keyestudio WiKi

keyestudio WiKi

Dec 06, 2023

# **KEYESTUDIO DOCS**

2       2.Kit list       3         3       3.Keyestudio ESP32 Mainboard       5         4       Getting started with Arduino       9         4.1       Windows System       9         4.2       Mac System:       32         5       Arduino Project       41         5.1       Download code files and Libraries files       41         5.2       Project 01: Hello World       41         5.3       Project 02: Turn On LED       48         5.4       Project 03: ED Flashing       60         5.5       Project 04: Breathing Led       65         5.6       Project 07: Flowing Water Light       77         5.8       Project 07: Flowing Water Light       80         5.10       Project 108-ID bigit Digital Tube       89         5.10       Project 108× Bot-matrix Display       92         5.12       Project 108× Bot-matrix Display       92         5.12       Project 138× Bot-matrix Display       92         5.11       Project 138× Buzzer       100         5.12       Project 138× Buzzer       100         5.13       Project 138× Buzzer       100         5.14       Project 138× Buzzer       103 <tdt< th=""><th>1</th><th>1.Inti</th><th>roduction</th><th>1</th></tdt<>	1	1.Inti	roduction	1
4       Getting started with Arduino       9         4.1       Windows System       9         4.2       Mac System:       32         5       Arduino Project       32         5       Arduino Project       41         5.1       Download code files and Libraries files       41         5.2       Project 01: Hello World       41         5.3       Project 03LED Flashing       60         5.4       Project 03LED Flashing       60         5.5       Project 05Traffic Lights       73         5.7       Project 05Traffic Lights       73         5.7       Project 081-Digit Digital Tube       80         5.9       Project 07: Flowing Water Light       80         5.0       Project 081-Digit Digital Tube       83         5.10       Project 084-Digit Digital Tube       89         5.11       Project 108×8 Dot-matrix Display       92         5.12       Project 108×8 Dot-matrix Display       92         5.12       Project 14/4CS95N Control 8 LEDs       96         5.13       Project 14/2Active Buzzer       100         5.14       Project 15Tit and LED       110         5.15       Project 1712C 128×32 LCD       116 </th <th>2</th> <th>2.Kit</th> <th>list</th> <th>3</th>	2	2.Kit	list	3
4.1       Windows System       9         4.2       Mac System:       32         5       Arduino Project       32         5       Arduino Project       32         5       Arduino Project       32         5       Arduino Project 01: Hello World       41         5.2       Project 01: Hello World       41         5.3       Project 02: Turn On LED       48         5.4       Project 03LED Flashing       60         5.5       Project 05Traffic Lights       73         5.7       Project 05Traffic Lights       73         5.8       Project 08 I-ED       77         5.8       Project 081-Digit Digital Tube       80         5.9       Project 081-Digit Digital Tube       83         5.10       Project 108-8 Dot-matrix Display       92         5.11       Project 108-8 Dot-matrix Display       92         5.12       Project 1174HC595N Control 8 LEDs       96         5.13       Project 13Passive Buzzer       100         5.14       Project 143Passive Buzzer       103         5.15       Project 15Tilt and LED       110         5.17       Project 18Small Fan       120         5.18       Pr	3	3.Key	vestudio ESP32 Mainboard	5
4.2       Mac System:       32         5       Arduino Project       41         5.1       Download code files and Libraries files       41         5.2       Project 01: Hello World       41         5.3       Project 02: Turn On LED       48         5.4       Project 01: Hello World       41         5.7       Project 05Traffic Lights       73         5.7       Project 06: RGB LED       77         5.8       Project 07: Flowing Water Light       80         5.9       Project 081-Digit Digital Tube       83         5.10       Project 104       84         5.11       Project 108×8 Dot-matrix Display       92         5.12       Project 112Active Buzzer       100         5.13       Project 13Passive Buzzer       103         5.14       Project 13Passive Buzzer       103         5.15       Project 13Passive Buzzer       103         5.1	4	Getti		9
5       Arduino Project       41         5.1       Download code files and Libraries files       41         5.2       Project 01: Hello World       41         5.3       Project 02: Turn On LED       48         5.4       Project 03LED Flashing       60         5.5       Project 04: Breathing Led       65         5.6       Project 05Traffic Lights       73         5.7       Project 07: Flowing Water Light       80         5.9       Project 081-Digit Digital Tube       83         5.10       Project 081-Digit Digital Tube       83         5.10       Project 081-Digit Digital Tube       89         5.11       Project 108-X8 Dot-matrix Display       92         5.12       Project 1174HC595N Control 8 LEDs       96         5.13       Project 12Active Buzzer       100         5.14       Project 13Passive Buzzer       103         5.15       Project 14: Mini Table Lamp       106         5.16       Project 172Active Buzzer       103         5.17       Project 18Mail Fan       120         5.18       Project 172C 128×32 LCD       116         5.19       Project 18Small Fan       120         5.20       Project 128small Fan <th></th> <th>4.1</th> <th>Windows System</th> <th>9</th>		4.1	Windows System	9
5.1       Download code files and Libraries files       41         5.2       Project 01: Hello World       41         5.3       Project 02: Turn On LED       48         5.4       Project 03LED Flashing       60         5.5       Project 04: Breathing Led       65         5.6       Project 05Traffic Lights       73         5.7       Project 06: RGB LED       77         5.8       Project 07: Flowing Water Light       80         5.9       Project 081-Digit Digital Tube       83         5.10       Project 084-Digit Digital Tube       83         5.10       Project 1084-Dot-matrix Display       92         5.12       Project 1174HC595N Control 8 LEDs       96         5.13       Project 13Passive Buzzer       100         5.14       Project 15Tilt and LED       110         5.15       Project 15Tilt and LED       110         5.16       Project 1172 L28x32 LCD       116         5.17       Project 18Small Fan       120         5.20       Project 128chag Alarm       124         5.21       Project 18Small Fan       120         5.22       Project 18Small Fan       120         5.23       Project 20Stepping Motor       <		4.2	Mac System:	32
5.2       Project 01: Hello World       41         5.3       Project 02: Turn On LED       48         5.4       Project 03LED Flashing       60         5.5       Project 04: Breathing Led       65         5.6       Project 05Traffic Lights       73         5.7       Project 06: RGB LED       77         5.8       Project 07: Flowing Water Light       80         5.9       Project 081-Digit Digital Tube       83         5.10       Project 094-Digit Digital Tube       83         5.11       Project 094-Digit Digital Tube       89         5.12       Project 1174HC595N Control 8 LEDs       96         5.13       Project 12Active Buzzer       100         5.14       Project 13Passive Buzzer       100         5.15       Project 14: Mini Table Lamp       110         5.16       Project 15Tilt and LED       110         5.17       Project 17 12C 128×32 LCD       116         5.18       Project 198eroy Sweep       124         5.20       Project 198eroy Sweep       135         5.23       Project 20Stepping Motor       129         5.23       Project 21Relay       136         5.24       Project 23Filame Alarm       136<	5	Ardu	ino Project	41
5.3       Project 02: Turn On LED       48         5.4       Project 03LED Flashing       60         5.5       Project 04: Breathing Led       65         5.6       Project 05Traffic Lights       73         5.7       Project 06: RGB LED       77         5.8       Project 07: Flowing Water Light       80         5.9       Project 081-Digit Digital Tube       83         5.10       Project 094-Digit Digital Tube       83         5.10       Project 108×8 Dot-matrix Display       92         5.12       Project 1174HC595N Control 8 LEDs       96         5.13       Project 12Active Buzzer       100         5.14       Project 157lit and LED       103         5.15       Project 16Burglar Alarm       114         5.16       Project 17 12C 128×32 LCD       116         5.17       Project 18Small Fan       120         5.20       Project 20Stepping Motor       129         5.21       Project 21Relay       135         5.23       Project 23Flame Alarm       141         5.24       Project 20Stepping Motor       129         5.23       Project 23Flame Alarm       141         5.24       Project 24 Night Lamp       146		5.1	Download code files and Libraries files	41
5.4       Project 03LED Flashing       60         5.5       Project 04: Breathing Led       65         5.6       Project 05Trafic Lights       73         5.7       Project 06: RGB LED       77         5.8       Project 07: Flowing Water Light       80         5.9       Project 081-Digit Digital Tube       83         5.10       Project 094-Digit Digital Tube       83         5.11       Project 108×8 Dot-matrix Display       92         5.12       Project 12Active Buzzer       100         5.13       Project 13Passive Buzzer       103         5.15       Project 13Passive Buzzer       103         5.16       Project 14: Mini Table Lamp       106         5.16       Project 15Tilt and LED       110         5.17       Project 108was 2LCD       116         5.19       Project 1172 L2 8x32 LCD       116         5.19       Project 19Servo Sweep       124         5.20       Project 20Stepping Motor       129         5.21       Project 21Relay       136         5.23       Project 23Flame Alarm       141         5.24       Project 23Flame Alarm       141         5.25       Project 24Night Lamp       146 <td></td> <td>5.2</td> <td>Project 01: Hello World</td> <td>41</td>		5.2	Project 01: Hello World	41
5.5       Project 04: Breathing Led       65         5.6       Project 05Traffic Lights       73         5.7       Project 06: RGB LED       77         5.8       Project 07: Flowing Water Light       80         5.9       Project 081-Digit Digital Tube       83         5.10       Project 094-Digit Digital Tube       83         5.11       Project 108×8 Dot-matrix Display       92         5.12       Project 1174HC595N Control 8 LEDs       96         5.13       Project 12Active Buzzer       100         5.14       Project 13Passive Buzzer       103         5.15       Project 14: Mini Table Lamp       106         5.16       Project 1712C 128×32 LCD       116         5.17       Project 18Small Fan       120         5.20       Project 19Servo Sweep       124         5.21       Project 20Stepping Motor       129         5.22       Project 21Relay       135         5.23       Project 23Flame Alarm       141         5.24       Project 23Flame Alarm       141         5.25       Project 24Night Lamp       136         5.26       Project 24Night Lamp       146         5.27       Project 24Night Lamp       150		5.3	Project 02: Turn On LED	48
5.5       Project 04: Breathing Led       65         5.6       Project 05Traffic Lights       73         5.7       Project 06: RGB LED       77         5.8       Project 07: Flowing Water Light       80         5.9       Project 081-Digit Digital Tube       83         5.10       Project 094-Digit Digital Tube       83         5.11       Project 108×8 Dot-matrix Display       92         5.12       Project 1174HC595N Control 8 LEDs       96         5.13       Project 12Active Buzzer       100         5.14       Project 13Passive Buzzer       103         5.15       Project 14: Mini Table Lamp       106         5.16       Project 1712C 128×32 LCD       116         5.17       Project 18Small Fan       120         5.20       Project 19Servo Sweep       124         5.21       Project 20Stepping Motor       129         5.22       Project 21Relay       135         5.23       Project 23Flame Alarm       141         5.24       Project 23Flame Alarm       141         5.25       Project 24Night Lamp       136         5.26       Project 24Night Lamp       146         5.27       Project 24Night Lamp       150		5.4	Project 03LED Flashing	60
5.7       Project 06: RGB LED       77         5.8       Project 07: Flowing Water Light       80         5.9       Project 081-Digit Digital Tube       83         5.10       Project 094-Digit Digital Tube       89         5.11       Project 108×8 Dot-matrix Display       92         5.12       Project 1174HC595N Control 8 LEDs       96         5.13       Project 12Active Buzzer       100         5.14       Project 13Passive Buzzer       103         5.15       Project 15Tilt and LED       106         5.16       Project 15Tilt and LED       110         5.17       Project 16Burglar Alarm       110         5.18       Project 17 12C 128×32 LCD       116         5.19       Project 18Small Fan       120         5.20       Project 18Small Fan       120         5.21       Project 20Stepping Motor       129         5.22       Project 21Relay       135         5.23       Project 22Dimming Light       136         5.24       Project 23Flame Alarm       141         5.25       Project 24Night Lamp       141         5.26       Project 24Night Lamp       146         5.27       Project 25Human Induction Lamp       150		5.5		65
5.8       Project 07: Flowing Water Light       80         5.9       Project 081-Digit Digital Tube       83         5.10       Project 094-Digit Digital Tube       83         5.11       Project 094-Digit Digital Tube       89         5.12       Project 108x8 Dot-matrix Display       92         5.12       Project 1174HC595N Control 8 LEDs       96         5.13       Project 12Active Buzzer       100         5.14       Project 13Passive Buzzer       103         5.15       Project 14: Mini Table Lamp       106         5.16       Project 15Tlit and LED       110         5.17       Project 16Burglar Alarm       114         5.18       Project 17 I2C 128×32 LCD       116         5.19       Project 18Small Fan       120         5.20       Project 19Servo Sweep       124         5.21       Project 20Stepping Motor       129         5.22       Project 21Relay       135         5.23       Project 23Flame Alarm       141         5.24       Project 23Flame Alarm       141         5.25       Project 24Might Lamp       146         5.26       Project 25Human Induction Lamp       150         5.27       Project 26Sound Contro		5.6	Project 05Traffic Lights	73
5.8       Project 07: Flowing Water Light       80         5.9       Project 081-Digit Digital Tube       83         5.10       Project 094-Digit Digital Tube       83         5.11       Project 094-Digit Digital Tube       89         5.12       Project 108x8 Dot-matrix Display       92         5.12       Project 1174HC595N Control 8 LEDs       96         5.13       Project 12Active Buzzer       100         5.14       Project 13Passive Buzzer       103         5.15       Project 15Tilt and LED       106         5.16       Project 15Tilt and LED       110         5.17       Project 16Burglar Alarm       110         5.18       Project 18Small Fan       120         5.20       Project 18Small Fan       120         5.21       Project 20Stepping Motor       129         5.22       Project 21Relay       135         5.23       Project 22Dimming Light       136         5.24       Project 23Flame Alarm       141         5.25       Project 24Night Lamp       146         5.26       Project 25Human Induction Lamp       150         5.27       Project 26Sound Control Fan       151         5.28       Project 27Temperature Mea		5.7	Project 06: RGB LED	77
5.9       Project 081-Digit Digital Tube       83         5.10       Project 094-Digit Digital Tube       89         5.11       Project 108×8 Dot-matrix Display       92         5.12       Project 1174HC595N Control 8 LEDs       96         5.13       Project 12Active Buzzer       100         5.14       Project 13Passive Buzzer       103         5.15       Project 157ilt and LED       106         5.16       Project 15Tilt and LED       110         5.17       Project 16Burglar Alarm       110         5.17       Project 17 I2C 128×32 LCD       116         5.19       Project 18Small Fan       120         5.20       Project 20Stepping Motor       129         5.21       Project 20Stepping Motor       129         5.22       Project 23Flame Alarm       136         5.24       Project 23Flame Alarm       146         5.25       Project 24Night Lamp       146         5.26       Project 24Night Lamp       150         5.27       Project 26Sound Control Fan       151         5.28       Project 27Temperature Measurement       155		5.8		80
5.10       Project 094-Digit Digital Tube       89         5.11       Project 108×8 Dot-matrix Display       92         5.12       Project 1174HC595N Control 8 LEDs       96         5.13       Project 12Active Buzzer       100         5.14       Project 13Passive Buzzer       103         5.15       Project 13Passive Buzzer       103         5.16       Project 15Tilt and LED       106         5.16       Project 15Tilt and LED       110         5.17       Project 16Burglar Alarm       114         5.18       Project 17 12C 128×32 LCD       116         5.19       Project 18Small Fan       120         5.20       Project 19Servo Sweep       124         5.21       Project 20Stepping Motor       129         5.22       Project 21Relay       135         5.23       Project 22Finming Light       136         5.24       Project 23Flame Alarm       141         5.25       Project 24Night Lamp       146         5.26       Project 25Human Induction Lamp       150         5.27       Project 26Sound Control Fan       151         5.28       Project 27Temperature Measurement       155		5.9		83
5.11       Project 108×8 Dot-matrix Display       92         5.12       Project 1174HC595N Control 8 LEDs       96         5.13       Project 12Active Buzzer       100         5.14       Project 13Passive Buzzer       103         5.15       Project 13Passive Buzzer       103         5.16       Project 14: Mini Table Lamp       106         5.17       Project 15Tilt and LED       110         5.17       Project 16Burglar Alarm       114         5.18       Project 17 I2C 128×32 LCD       116         5.19       Project 18Small Fan       120         5.20       Project 19Servo Sweep       124         5.21       Project 20Stepping Motor       129         5.22       Project 21Relay       135         5.23       Project 22Dimming Light       136         5.24       Project 23Flame Alarm       141         5.25       Project 24Night Lamp       146         5.26       Project 25Human Induction Lamp       150         5.27       Project 26Sound Control Fan       151         5.28       Project 27Temperature Measurement       155		5.10		89
5.13       Project 12Active Buzzer       100         5.14       Project 13Passive Buzzer       103         5.15       Project 14: Mini Table Lamp       106         5.16       Project 15Tilt and LED       110         5.17       Project 16Burglar Alarm       114         5.18       Project 16Burglar Alarm       114         5.19       Project 17 I2C 128×32 LCD       116         5.19       Project 18Small Fan       120         5.20       Project 19Servo Sweep       124         5.21       Project 20Stepping Motor       129         5.22       Project 21Relay       135         5.23       Project 22Dimming Light       136         5.24       Project 23Flame Alarm       141         5.25       Project 24Night Lamp       146         5.26       Project 25Human Induction Lamp       150         5.27       Project 26Sound Control Fan       151         5.28       Project 27Temperature Measurement       155		5.11		92
5.13       Project 12Active Buzzer       100         5.14       Project 13Passive Buzzer       103         5.15       Project 14: Mini Table Lamp       106         5.16       Project 15Tilt and LED       110         5.17       Project 16Burglar Alarm       114         5.18       Project 16Burglar Alarm       114         5.19       Project 17 I2C 128×32 LCD       116         5.19       Project 18Small Fan       120         5.20       Project 19Servo Sweep       124         5.21       Project 20Stepping Motor       129         5.22       Project 21Relay       135         5.23       Project 22Dimming Light       136         5.24       Project 23Flame Alarm       141         5.25       Project 24Night Lamp       146         5.26       Project 25Human Induction Lamp       150         5.27       Project 26Sound Control Fan       151         5.28       Project 27Temperature Measurement       155		5.12	5 1 5	96
5.14       Project 13Passive Buzzer.       103         5.15       Project 14: Mini Table Lamp       106         5.16       Project 15Tilt and LED.       110         5.17       Project 16Burglar Alarm       114         5.18       Project 17 I2C 128×32 LCD       116         5.19       Project 18Small Fan       120         5.20       Project 19Servo Sweep       124         5.21       Project 20Stepping Motor       129         5.22       Project 21Relay       135         5.23       Project 22Dimming Light       136         5.24       Project 23Flame Alarm       141         5.25       Project 24Night Lamp       146         5.26       Project 25Human Induction Lamp       150         5.27       Project 26Sound Control Fan       151         5.28       Project 27Temperature Measurement       155		5.13		100
5.15       Project 14: Mini Table Lamp       106         5.16       Project 15Tilt and LED       110         5.17       Project 16Burglar Alarm       114         5.18       Project 17 I2C 128×32 LCD       116         5.19       Project 18Small Fan       120         5.20       Project 19Servo Sweep       124         5.21       Project 20Stepping Motor       129         5.22       Project 21Relay       135         5.23       Project 22Dimming Light       136         5.24       Project 23Flame Alarm       141         5.25       Project 24Night Lamp       146         5.26       Project 25Human Induction Lamp       150         5.27       Project 26Sound Control Fan       151         5.28       Project 27Temperature Measurement       155		5.14		103
5.16       Project 15Tilt and LED       110         5.17       Project 16Burglar Alarm       114         5.18       Project 17 I2C 128×32 LCD       116         5.19       Project 18Small Fan       120         5.20       Project 19Servo Sweep       124         5.21       Project 20Stepping Motor       129         5.22       Project 21Relay       135         5.23       Project 22Dimming Light       136         5.24       Project 23Flame Alarm       141         5.25       Project 24Night Lamp       146         5.26       Project 25Human Induction Lamp       150         5.27       Project 26Sound Control Fan       151         5.28       Project 27Temperature Measurement       155		5.15		106
5.17       Project 16Burglar Alarm       114         5.18       Project 17 I2C 128×32 LCD       116         5.19       Project 18Small Fan       120         5.20       Project 19Servo Sweep       124         5.21       Project 20Stepping Motor       129         5.22       Project 21Relay       135         5.23       Project 22Dimming Light       136         5.24       Project 23Flame Alarm       141         5.25       Project 24Night Lamp       146         5.26       Project 25Human Induction Lamp       150         5.27       Project 26Sound Control Fan       151         5.28       Project 27Temperature Measurement       155		5.16		110
5.18       Project 17 I2C 128×32 LCD       116         5.19       Project 18Small Fan       120         5.20       Project 19Servo Sweep       124         5.21       Project 20Stepping Motor       129         5.22       Project 21Relay       135         5.23       Project 22Dimming Light       136         5.24       Project 23Flame Alarm       141         5.25       Project 24Night Lamp       146         5.26       Project 25Human Induction Lamp       150         5.27       Project 26Sound Control Fan       151         5.28       Project 27Temperature Measurement       155		5.17	5	
5.19Project 18Small Fan1205.20Project 19Servo Sweep1245.21Project 20Stepping Motor1295.22Project 21Relay1355.23Project 22Dimming Light1365.24Project 23Flame Alarm1415.25Project 24Night Lamp1465.26Project 25Human Induction Lamp1505.27Project 26Sound Control Fan1515.28Project 27Temperature Measurement155		5.18		116
5.20Project 19Servo Sweep1245.21Project 20Stepping Motor1295.22Project 21Relay1355.23Project 22Dimming Light1365.24Project 23Flame Alarm1415.25Project 24Night Lamp1465.26Project 25Human Induction Lamp1505.27Project 26Sound Control Fan1515.28Project 27Temperature Measurement155		5.19		120
5.21Project 20Stepping Motor1295.22Project 21Relay1355.23Project 22Dimming Light1365.24Project 23Flame Alarm1415.25Project 24Night Lamp1465.26Project 25Human Induction Lamp1505.27Project 26Sound Control Fan1515.28Project 27Temperature Measurement155		5.20		124
5.22Project 21Relay1355.23Project 22Dimming Light1365.24Project 23Flame Alarm1415.25Project 24Night Lamp1465.26Project 25Human Induction Lamp1505.27Project 26Sound Control Fan1515.28Project 27Temperature Measurement155			5 1	129
5.23Project 22Dimming Light1365.24Project 23Flame Alarm1415.25Project 24Night Lamp1465.26Project 25Human Induction Lamp1505.27Project 26Sound Control Fan1515.28Project 27Temperature Measurement155			5 11 0	
5.24Project 23Flame Alarm1415.25Project 24Night Lamp1465.26Project 25Human Induction Lamp1505.27Project 26Sound Control Fan1515.28Project 27Temperature Measurement155				136
5.25Project 24Night Lamp1465.26Project 25Human Induction Lamp1505.27Project 26Sound Control Fan1515.28Project 27Temperature Measurement155				141
5.26Project 25Human Induction Lamp1505.27Project 26Sound Control Fan1515.28Project 27Temperature Measurement155			5	
5.27Project 26Sound Control Fan1515.28Project 27Temperature Measurement155				
5.28 Project 27Temperature Measurement				
				-

	5.30	Project 29Temperature Humidity Meter
	5.31	Project 30Ultrasonic Ranger
	5.32	Project 31Temperature Instrument
	5.33	Project 32RFID
	5.34	Project 33Keypad Door
	5.35	Project 34IR Control Sound and LED
	5.36	Project 35Bluetooth
	5.37	Project 36WiFi Station Mode
	5.38	Project 37WiFi AP Mode
	5.39	Project 38WiFi Station+AP Mode
	5.40	Project 39: WiFi Test
	5.41	Project 40WiFi Smart Home
_	~	
6		ng started with Python 265
	6.1	1.Installing Thonny (Important)   265
	6.2	2. Basic Configuration of Thonny
	6.3	3.Installing CP2102 driver
	6.4	4.Burning Micropython Firmware (Important)
	6.5	5.Test Code
	6.6	6.Thonny Common operations
7	Pytho	on Project 305
	7.1	Download code files
	7.2	Project 01: Hello World
	7.3	Project 02: Turn on LED
	7.4	Project 03LED Flashing
	7.5	Project 04: Breathing Led
	7.6	Project 05Traffic Lights
	7.7	Project 06: RGB LED
	7.8	Project 07: Flowing Water Light
	7.9	Project 081-Digit Digital Tube
	7.10	Project 094-Digit Digital Tube
	7.11	Project 108×8 Dot-matrix Display
	7.12	Project 1174HC595N Control 8 LEDs
	7.12	Project 12Active Buzzer
	7.14	Project 13Passive Buzzer
	7.15	Project 14: Mini Table Lamp         390           Project 15TH         14 PD
		Project 15Tilt and LED
	7.17	Project 16Burglar Alarm
	7.18	Project 17 I2C 128×32 LCD
	7.19	Project 18Small Fan
	7.20	Project 19Servo Sweep
	7.21	Project 20Stepping Motor
	7.22	Project 21Relay
	7.23	Project 22Dimming Light
	7.24	Project 23Flame Alarm
	7.25	Project 24Night Lamp
	7.26	Project 25Human Induction Lamp
	7.27	Project 26Sound Control Fan
	7.28	Project 27Temperature Measurement
	7.29	Project 28Rocker control light
	7.30	Project 29Temperature Humidity Meter
	7.30	
		5
	7.32	Project 31Temperature Instrument

	7.33	Project 32RFID
	7.34	Project 33Keypad Door
	7.35	Project 34IR Control Sound and LED
		Project 35WiFi Station Mode
		Project result
		Project 36WiFi AP Mode
		Introduction
		Project wiring
	7.41	Project result
		Project 37WiFi Station+AP Mode
	2	
8	Getti	ng started with C (Raspberry Pi) 581
	8.1	Install the Raspberry Pi OS System
	8.2	Linux SystemRaspberry Pi
	8.3	Import the Arduino C library
9	Rasp	berryPi Arduino 645
	9.1	Download code files and Libraries files
	9.2	Project 01: Hello World
	9.3	Project 02: Turn On LED
	9.4	Project 03LED Flashing
	9.5	Project 04: Breathing Led
	9.6	Project 05Traffic Lights
	9.7	Project 06: RGB LED
	9.8	Project 07: Flowing Water Light
	9.9	Project 081-Digit Digital Tube
	9.10	Project 094-Digit Digital Tube
	9.11	Project 108×8 Dot-matrix Display
	9.11	Project 1174HC595N Control 8 LEDs
	9.12	Project 12Active Buzzer
	9.13	Project 13Passive Buzzer
	9.15	J I
	9.16	Project 15Tilt and LED
	9.17	Project 16Burglar Alarm
	9.18	Project 17 I2C 128×32 LCD
		Project 18Small Fan
		Project 19Servo Sweep
		Project 20Stepping Motor
		Project 21Relay
	9.23	Project 22Dimming Light
	9.24	Project 23Flame Alarm
	9.25	Project 24Night Lamp
	9.26	Project 25Human Induction Lamp
	9.27	Project 26Sound Control Fan
	9.28	Project 27Temperature Measurement
	9.29	Project 28Rocker control light
	9.30	Project 29Temperature Humidity Meter
	9.31	Project 30Ultrasonic Ranger
	9.32	Project 31Temperature Instrument
	9.33	Project 32RFID
	9.34	Project 33Keypad Door
	9.35	Project 34IR Control Sound and LED
	9.36	Project 35Bluetooth
	9.37	Project 35.1Classic Bluetooth
		· · · · · · · · · · · · · · · · · · ·

9.38	Project 35.2Bluetooth Control LED	89
9.39	Project 36WiFi Station Mode	92
9.40	Project 37WiFi AP Mode	97
9.41	Project 38WiFi Station+AP Mode	01
9.42	Project 39: WiFi Test	05
9.43	Project 40WiFi Smart Home	11

## **1.INTRODUCTION**

Do you want to learn about programming?

As long as you're passionate about science and dare to explore new things, this kit is surely the best choice for you. The Keyestudio ESP32 Learning Kit Ultimate Edition mainly contains some common electronic components/sensors/modules, a ESP32 mainboard and bread wires are also included.

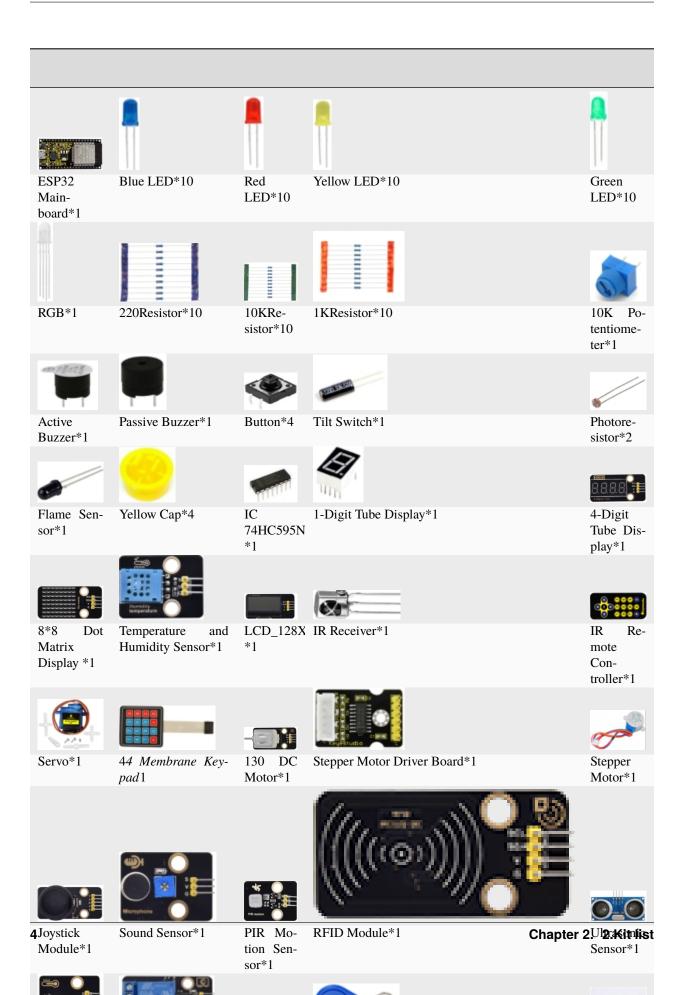
As many as 117 project tutorials are provided, which contain detailed wiring diagrams, components knowledge, and fascinating project code. Each project is produced using Thonny for Windows, Arduino IDE for Windows, and Arduino IDE for Raspberry Pi. It's easy to get started.

You can create numerous fascinating DIY experiments with one controller(ESP32), various of sensors/modules and electronics. These courses can give you a deeper understanding of programming methods, logic, electronic circuits and the Linux operating system (Raspberry Pi).

TWO

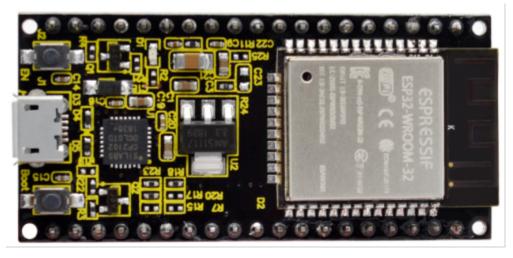
## 2.KIT LIST

When you received the kit, the first thing you see is a beautiful packaged box. Each accessory was safely and orderly packed in a small bag. Let's check them first:



THREE

## **3.KEYESTUDIO ESP32 MAINBOARD**



#### Introduction

Keyestudio ESP32 Core board is a Mini development board based on the ESP-WROOM-32 module. The board has brought out most I/O ports to pin headers of 2.54mm pitch. These provide an easy way of connecting peripherals according to your own needs.

When it comes to developing and debugging with the development board, the both side standard pin headers can make your operation more simple and handy.

The ESP-WROOM-32 module is the industry's leading integrated WiFi + Bluetooth solution with less than 10 external components. It integrates antenna switches, RF balun, power amplifiers, low noise amplifiers, filters as well as power management modules. At the same time, it also integrates TSMC's low-power 40nm technology, power performance and RF performance, making it safe, reliable and easy to expand to a variety of applications.

#### Specifications

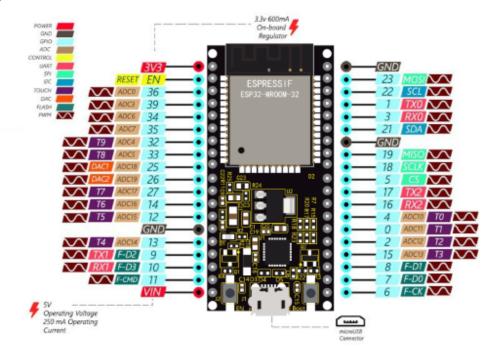
Microcontroller: ESP-WROOM-32Module USB to Serial Port Chip: CP2102-GMR Working Voltage: DC 5V Working Current80mAAverage Current Supply500mAMinimum Working Temperature Range : -40°C ~ +85°C WiFi ModeStation/SoftAP/SoftAP+Station/P2P WiFi Protocol 802.11 b/g/n/e/i802.11nSpeed up to 150 Mbps WiFi Frequency Range2.4 GHz ~ 2.5 GHz

Bluetooth Protocol conform to Bluetooth v4.2 BR/EDR and BLE Standard

Dimensions55\*26\*13mm

#### Weight9.3g

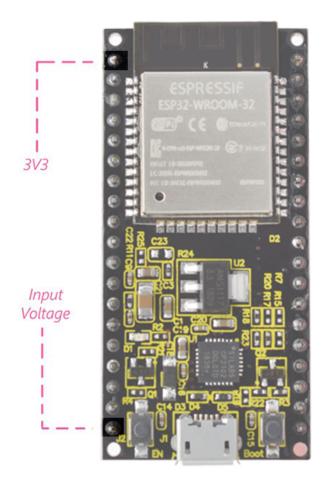
#### Pin out



ESP32 has fewer pins than commonly used processors, but it doesn't have any problems reusing multiple functions on pins.

Warning: The pin voltage level of the ESP32 is 3.3V. If you want to connect the ESP32 to another device with an operating voltage of 5V, you should use a level converter to convert the voltage level.

**Power Pins:** The module has two power pins +5V and 3.3V. You can use these two pins to power other devices and modules.



\*\* GND Pins\*\*The module has three grounded pins.

**Enable pin (EN) :** This pin is used to enable and disable modules. The pin enables module at high level and disables module at low level.

**Input/Output pins (GPIO) :** You can use 32 GPIO pins to communicate with LEDs, switches and other input/output devices. You can also pull these pins up or down internally.

**Note:** Though GPIO6 to GPIO11 pins (SCK/CLK, SDO/SD0, SDI/SD1, SHD/SD2, SWP/SD3 and SCS/CMD pins) are used for SPI communication for the internal module, which are not recommended.

**ADC:** You can use the 16 ADC pins on this module to convert analog voltages (the output of some sensors) into digital voltages. Some of these converters are connected to internal amplifiers and which are capable of measuring small voltages with high accuracy.

DAC: ESP32 module has two A/D converters with 8-bit precision.

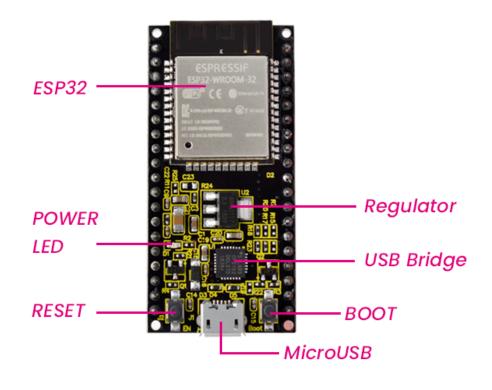
**Touch pad:** There are 10 pins on the ESP32 module that are sensitive to capacitance changes. You can attach these pins to certain PCB's pads and use them as touch switches.

**SPI:** There are two SPI interfaces on the module, which can be used to connect the display screen, SD/microSD memory card module as well as external flash memory, etc.

I2C: SDA and SCL pins are used for I2C communication.

**Serial Communication (UART) :** There are two UART serial interfaces on this module, which can be used to transfer up to 5Mbps of information between two devices . The UARTO also has CTS and RTS control functions.

**PWM:** Almost all ESP32 input/output pins can be used for PWM(pulse-width modulation). Using these pins can control the motor, LED lights and color changes for some other sensorsfor example: color sensor, etc.



Components

FOUR

## **GETTING STARTED WITH ARDUINO**

# 4.1 Windows System



## 4.1.1 1.1.Download and install Arduino software

1First, enter arduino's official website:https://www.arduino.cc/, and click "SOFTWARE" to enter the download page. As shown in the figure below

PROFESSIONAL	EDUCATION	STORE	Q Searc	h on Arduino.cc	SIGN II
©⊙	HARDWARE	SOFTWARE	CLOUD DOCUMENTATION - COMMU	NITY <del>v</del> blog about	
		JINO?	ARDUINO EDUVISION	Time to play Check out the best of our kip products, for all the fun of the	
BUY AN	I ARDUINO		October 21st -5pm CEST <u>Sign up now!</u>	electronics fair! Enter here for fun!	
00	HARDWARE	SOFTWARE	CLOUD DOCUMENTATION - COM	MUNITY <del>v</del> blog abo	

# Downloads



2Then, select and download the corresponding installer for your operating system. If you are a Windows user, please select "Windows Installer" to download to install the driver correctly.



Choose to click the **Windows Win7 and newer** to download Arduino 1.8.16 version installer, which requires manual installation. But when click the **Windows ZIP File**, the Arduino 1.8.16 zip file will be downloaded directly, just unzip it to complete the installation.



In general, you can click **JUST DOWNLOAD** to download it, although if you like, you can choose a small sponsorship to help the great Arduino open source cause.

3After the Arduino IDE is downloaded, continue the installation. When you receive the warning from the operating system, please allow the driver installation by clicking **I** Agree first, and then click **Next** after selecting the components to install.

💿 Arduino Setup: License Agreement	_		$\times$	
Please review the license agreement before in accept all terms of the agreement, click I Agre		ino. If you		
GNU LESSER GENERAL PUBLIC LICENSE			^	
Version 3, 29 June 2007				
Copyright (C) 2007 Free Software Foundation, Inc. < <u>http://fsf.org/</u> >				
Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.				
This version of the GNU Lesser General Public License incorporates the terms and conditions of version 3 of the GNU General Public License, supplemented				
and conditions of version 3 of the GNU General Public License, supplemented by the additional permissions listed below.				
Cancel Nullsoft Install System v3.0		I Agre	e	
Arduino Setup: Installation Options			×	
<ul> <li>Arduino Setup: Installation Options</li> <li>Check the components you want to install and you don't want to install. Click Next to continue</li> </ul>	– I uncheck the e.	componer		
Check the components you want to install and you don't want to install. Click Next to continu Select components to install: Install Arduino Create Start N Create Deskto	e. o software ver 4enu shortcu op shortcut			
Check the components you want to install and you don't want to install. Click Next to continu Select components to install:	e. o software ver 4enu shortcu op shortcut		nts	

4Select the installation directory (we recommend keeping the default directory), and then click Install.

💿 Arduino Setup: Installation Folder	_		$\times$	
Setup will install Arduino in the following fol folder, dick Browse and select another fold installation.				
Destination Folder				
C:\Users\Administrator\Desktop\Arduino Browse				
Concernational E42 AMP				
Space required: 543.4MB				
Space available: 14.7GB				
Cancel Nullsoft Install System v3.0	< Back	Inst	tall	

5Select Install if the following screen appears.

Windows Security
Would you like to install this device software?
Name: Arduino USB Driver Publisher: Arduino LLC
Always trust software from "Arduino LLC". Install Don't Install
You should only install driver software from publishers you trust. How can I decide which device software is safe to install?

This process extracts and installs all the necessary files to properly execute the Arduino software (IDE).

💿 Arduino Setup: Installing			$\times$
Extract: iox128a1u.h			
Show details			
Cancel Nullsoft Install System v3.0	< Back	Clos	se

After installation is complete, an Arduino Software shortcut will be generated in the desktop.

💿 Arduino Setup: Completed	_		
Completed			
Show details			
Cancel Nullsoft Install System v3.0	< Back	Clo	se

### 4.1.2 1.2.Install a driver on Windows

NoteIf you have installed the driver, just skip it

Before using the ESP32 board, you must install a driver, otherwise it will not communicate with computer. Unlike the USB series chip(ATMEGA8U2) of the Arduino UNO R3, the ESP32 board is used the CP2102 chip USB series chip and USB type C interface.

The driver of the CP2102 chip is included in 1.8.0 version and newer version of Arduino IDE. Usually, you connect the board to the computer and wait for Windows to begin its driver installation process. After a few moments, the process will succeed.

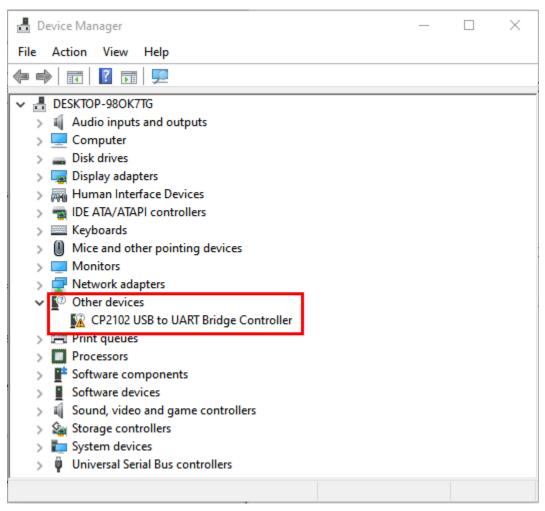
Note:

<sup>1.</sup> Please make sure that your IDE is updated to 1.8.0 or newer version

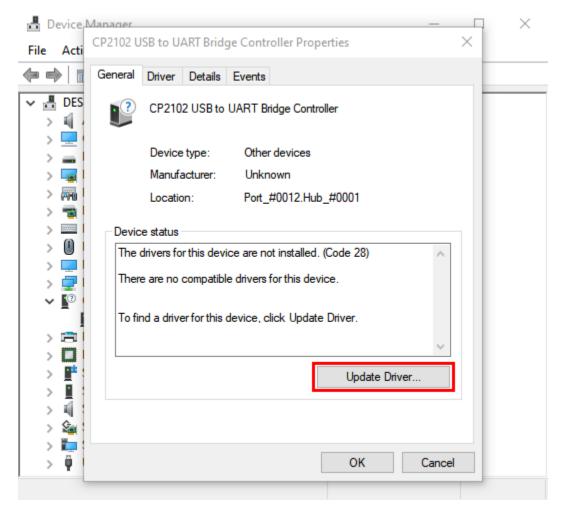
2. If the version of Arduino IDE you download is below 1.8, you should download the driver of CP2102 and install it manually.

Link to download the driver of CP2102:Download CP2102 Driver

If the driver installation process fail, you need to install the driver manually. Open device Manager for your computer and right-click "the computer"  $\rightarrow$  click "Properties"  $\rightarrow$  Click "Device Manager". Look under Ports (COM & LPT) or other device, a yellow exclamation mark means that the CP2102 driver installation failed.



It shows that the driver for CP2102 was not installed sucessfully with a yellow mark. Doubleclick CP2102 USB to UART Bridge Controller, and then click "Update drive…" to update the driver.

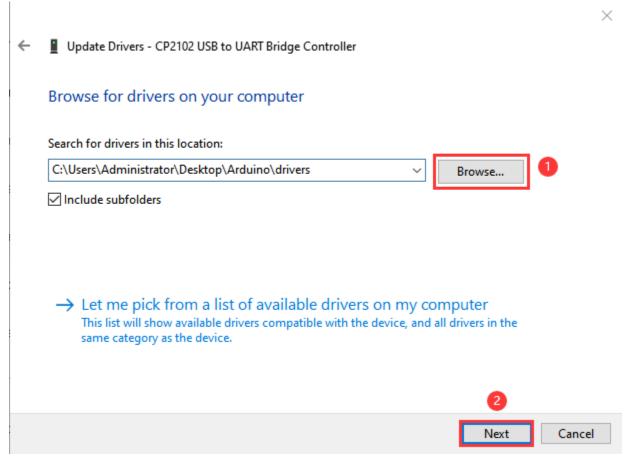


Click "Browse my computer for drivers" and find the Arduino software we installed or downloaded.

How	v do you want to search for drivers?		
→	Search automatically for drivers Windows will search your computer for the best your device.	available driver and install it on	
$\rightarrow$	Browse my computer for drivers Locate and install a driver manually.		

There is a **drivers** folder in Arduino software installed package Arduino ), open driver folder and you can see the driver of CP210X series chips.

Click"Browse...", then find the drivers folder, or you could enter"driver" to search in rectangular box, then click"Next".



After a while, the driver is installed successfully.

 $\times$ 

Update Drivers - Silicon Labs CP210x USB to UART Bridge (COM3)

Windows has successfully updated your drivers

Windows has finished installing the drivers for this device:



Silicon Labs CP210x USB to UART Bridge



Open the computer device Manager again, you can see that the CP2102 driver has been successfully installed, and find the yellow exclamation mark disappear.

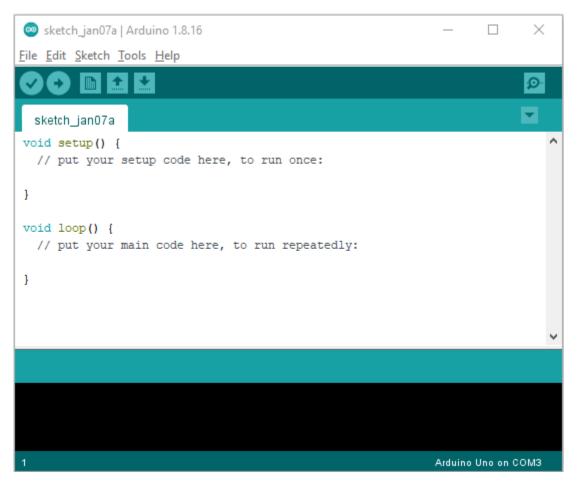
📇 Device Manager	_	$\times$
File Action View Help		
V 🗄 DESKTOP-980K7TG		
> 4 Audio inputs and outputs		
> 💻 Computer		
> 👝 Disk drives		
> 🏣 Display adapters		
> 🛺 Human Interface Devices		
> 📹 IDE ATA/ATAPI controllers		
> 🔤 Keyboards		
> U Mice and other pointing devices		
> 💻 Monitors		
> 🚽 Network adapters		
V 🛱 Ports (COM & LPT)		1
Silicon Labs CP210x USB to UART Bridge (COM3)		
> 🖻 Print queues		
> Processors		
> P Software components		
Software devices		
> 4 Sound, video and game controllers		
> 🍇 Storage controllers		
> 🏣 System devices		
> 🏺 Universal Serial Bus controllers		

## 4.1.3 1.3. Install the ESP32 on Arduino IDE

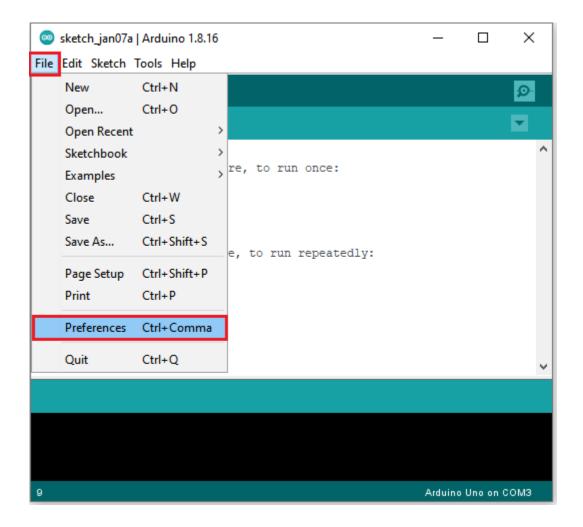
The installation process for ESP32 is almost the same as that for ESP8266. To install ESP32 on an Arduino IDE, follow these steps Noteyou need to download Arduino IDE 1.8.5 or advanced version to install the ESP32.



(1) Click the icon Arduino to open the Arduino IDE.



2Click"File"  $\rightarrow$  \*\*"Preferences"\*\*copy the website addresshttps://dl.espressif.com/dl/package\_esp32\_index.json in the "Additional Boards Manager URLs:"and then click" OK" to save the address.



Preferences	>
Settings Network	
Sketchbook location:	
C:\Users\Administrator\Doct	uments\Arduino Browse
Editor language:	System Default $\sim$ (requires restart of Arduino)
Editor font size:	12
Interface scale:	Automatic 100 - % (requires restart of Arduino)
Theme:	Default theme $\sim$ (requires restart of Arduino)
Show verbose output during:	compilation upload
Compiler warnings:	None 🗸
🗌 Display line numbers	Enable Code Folding
🗹 Verify code after uploa	d 🗌 Use external editor
Check for updates on st	artup 🗹 Save when verifying or uploading
🗌 Use accessibility featu	res
Additional Boards Manager U	JRLs: https://dl.espressif.com/dl/package_esp32_index.json 1
More preferences can be edi	ted directly in the file
C:\Users\Administrator\AppD	ata\Local\Arduino15\preferences.txt
(edit only when Arduino is	not running) 2
	OK Cancel

3First click "Tools"  $\rightarrow$  "Board:" and click "Boards Manager..." to enter "Boards Manager", enter "ESP32" in the box after "ALL", then select the latest version to Install, the installation package is not large, click "Install" to Install the plug-in, as shown in the figure below.

Boards Manager	×
Type All V ESP32	
esp32 by Espressif Systems Boards included in this package: ESP32 Dev Module, WEMOS LoLin32, WEMOS D1 MINI ESP32. <u>More Info</u>	2 1.0.6 V Install
	~
	Close

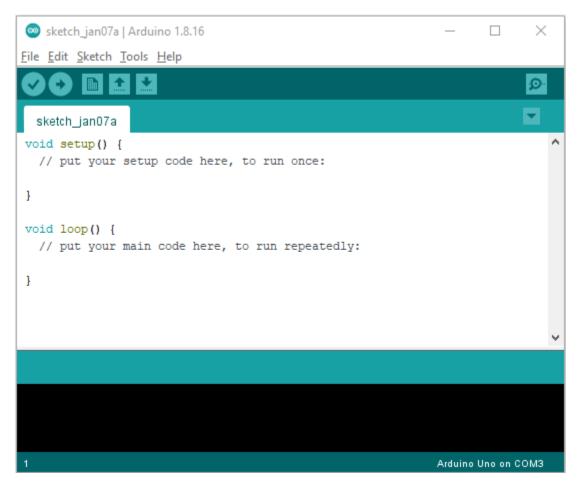
🞯 Boards Manager	$\times$
Type All ~ ESP32	
esp32 by Espressif Systems Boards included in this package: ESP32 Dev Module, WEMOS LoLin32, WEMOS D1 MINI ESP32. <u>More Info</u>	^
Installing	
	~
Downloading boards definitions. Downloaded 17,637kb of 51,126kb.	1

After successful installation, click "Close" to Close the page

### 4.1.4 1.4. Arduino IDE Setting:



1Click the icon Arduino to open the Arduino IDE.

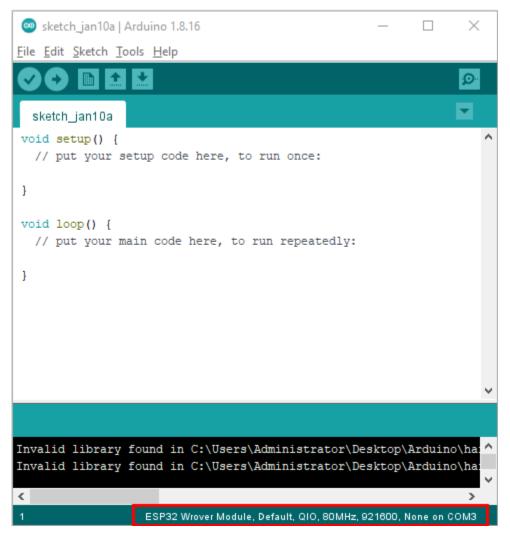


2When downloading the code to the board, you must select the correct name of Arduino board that matches the board connected to your computer, click "**Tools**"  $\rightarrow$  "**Board:**". As shown below ;

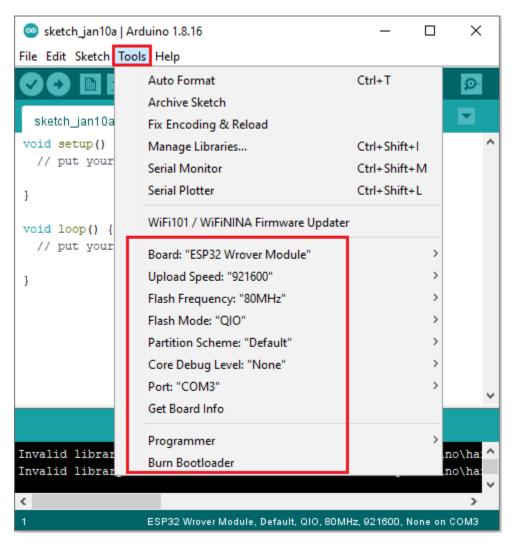
(Note: we use the ESP32 board in this tutorial; therefore, we select ESP32 Arduino\*\*)\*\*

### keyestudio WiKi

🥯 sketch_jan10a   A	rduino 1.8.16	- 0	×			
File Edit Sketch Too	ols Help					
sketch_jan10a	Auto Format Archive Sketch Fix Encoding & Reload Manage Libraries Serial Monitor	Ctrl+T Ctrl+Shift+I Ctrl+Shift+M	<b>1</b>			△ ESP32 Dev Module ESP32 Wrover Module ESP32 Pico Kit
} void loop() {	Serial Plotter WiFi101 / WiFiNINA Firmv	Ctrl+Shift+L vare Updater				TTGO LoRa32-OLED V1 XinaBox CW02 SparkFun ESP32 Thing
// put your	Board: "Arduino Uno"	;	Board	s Manager		u-blox NINA-W10 series (ESP32)
}	Port: "COM3" Get Board Info	*	Ardui	no AVR Boar no Mbed OS Arduino	ds RP2040 Boards	Widora AIR Electronic SweetPeas - ESP320
	Programmer: "AVRISP mk Burn Bootloader	II <sup>n</sup> \$	ESP32	Arduino	1	Nano32 LOLIN D32
			~			LOLIN D32 PRO WEMOS LOLIN32
						Dongsen Tech Pocket 32 "WeMos" WiFi&Bluetooth Battery ESPea32
						Noduino Quantum Node32s
		Arduino Uno o	n COM3			Hornbill ESP32 Dev

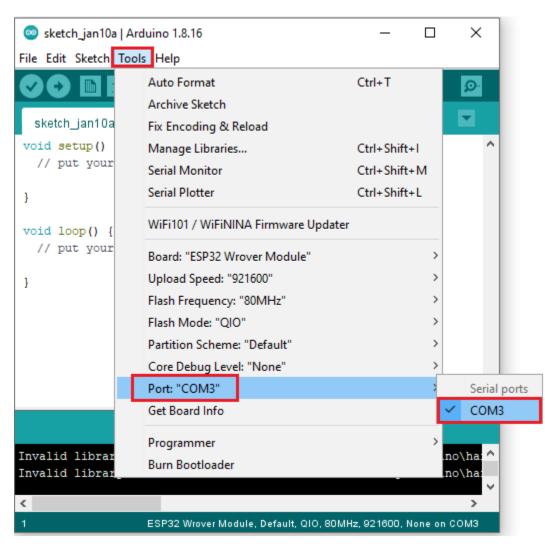


Set the board type as follows:

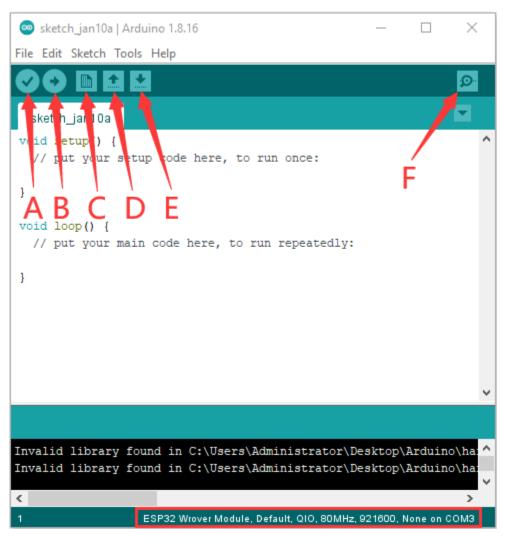


Then select the correct COM port (the corresponding COM port can be seen after the driver is installed successfully).

ᡖ Device Manager	_	$\times$
File Action View Help		
V 🗄 DESKTOP-980K7TG		
> 🐗 Audio inputs and outputs		
> 💻 Computer		
> 🚘 Disk drives		
> 🏣 Display adapters		
> 🏧 Human Interface Devices		
> 🦏 IDE ATA/ATAPI controllers		
> 🔤 Keyboards		
> II Mice and other pointing devices		
> 🛄 Monitors		
> 🚽 Network adapters		
✓		
Silicon Labs CP210x USB to UART Bridge (COM3)		
> 🚍 Print queues		
> Processors		
Software components		
Software devices		
Sound, video and game controllers		
> 🍇 Storage controllers		
> 🏣 System devices		
> 🏺 Universal Serial Bus controllers		



Before a code was uploaded to the ESP32 mainboard, we have to demonstrate the functionality of each symbol that appeared in the Arduino IDE toolbar.



- A- Used to verify whether there is any compiling mistakes or not.
- B- Used to upload the sketch to your Arduino board.
- C- Used to create shortcut window of a new sketch.
- D- Used to directly open an example sketch.
- E- Used to save the sketch.
- F- Used to send the serial data received from board to the serial monitor.

# 4.2 Mac System:



# 4.2.1 2.1.Download and install the Arduino IDE:

# Downloads



# Arduino IDE 1.8.16

The open-source Arduino Software (IDE) makes it easy to write code and upload it to the board. This software can be used with any Arduino board.

Refer to the Getting Started page for Installation instructions.

#### SOURCE CODE

Active development of the Arduino software is **hosted by GitHub**. See the instructions for **building the code**. Latest release source code archives are available **here**. The archives are PGP-signed so they can be verified using **this** gpg key.

#### DOWNLOAD OPTIONS

Windows Win 7 and newer Windows ZIP file

Windows app Win 8.1 or 10 Get

Linux 32 bits Linux 64 bits Linux ARM 32 bits Linux ARM 64 bits

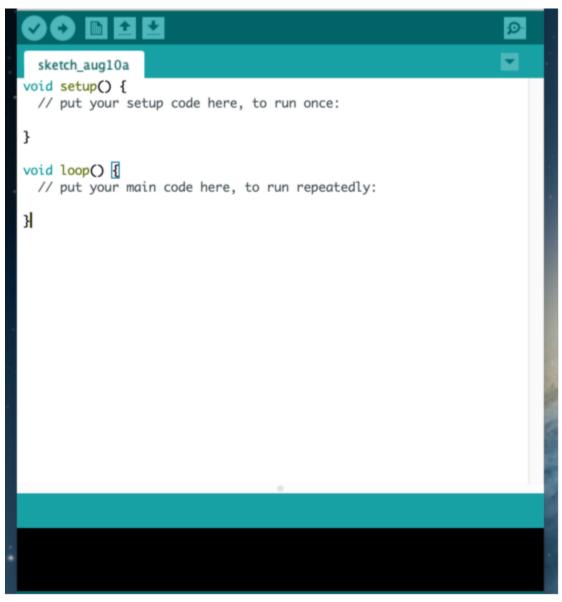
Mac OS X 10.10 or newer

Release Notes Checksums (sha512)

# 4.2.2 2.2. How to install the CP2102 driver

\*\*Note\*\*If you have installed the driver, just skip it

1Connect the ESP32 mainboard to your MacOS computer using a USB cable and open Arduino IDE.



Click "\*\*Tools"\*\*→"Board: ESP32 Dev Module "and "/dev/cu.usbserial-0001".

#### keyestudio WiKi

é Arduino File Edit Sketch	Tools Help	. d	M 8 9	# *		
	Auto Format					
	Archive Sketch					ſ
	Fix Encoding & Reload					
sketch_jan19a	Manage Libraries					
<pre>bid setup() { // put your setup code here, to ru</pre>	Serial Monitor					
// put your setup code here, to ru	Serial Plotter					
	WiFi101 / WiFiNINA Firmware Updater					
td loopO {	Merro ( / Wenning Pinning Opdate)					
// put your main code here, to run	ArduBlock					
	Board: "ESP32 Dev Module"					
	Upload Speed: "921600"					
	CPU Frequency: "240MHz (WiFi/BT)"					
	Flash Frequency: "80MHz"					
	Flash Mode: "QIO"					
	Flash Size: "4MB (32Mb)"					
	Partition Scheme: "Default 4MB with spiffs (1.2MB APP/1.5MB SPIFFS)"					
	Core Debug Level: "None"					
	PSRAM: "Disabled"					
	Get Board Info		/dev/cu.Bluet	ooth-incon	ing-Por	
	Dragenmar					
	Programmer				_	ľ
	Burn Bootloader					

Click to upload code.

**Note:** If code is uploaded unsuccessfully, you need to install driver of CP2102, please continue to follow the instructions as below:

2Download the driver of CP2102Download CP2102 Driver

3Click to download the MacOS version, as shown below.

### Download for WinCE

Platform	Software	Release Notes
MinCE 6.0 (2.1)	Download VCP (276 KB)	Download WinCE 6.0 Revision History
MinCE 5.0 (2.1)	Download VCP (271 KB)	Download WinCE 5.0 Revision History

# Download for Macintosh OSX (v5.3.5)

Platform	Software	Release Notes
🙀 Mac OSX	Download VCP (832 KB)	Download Mac VCP Revision History

## Download for Linux

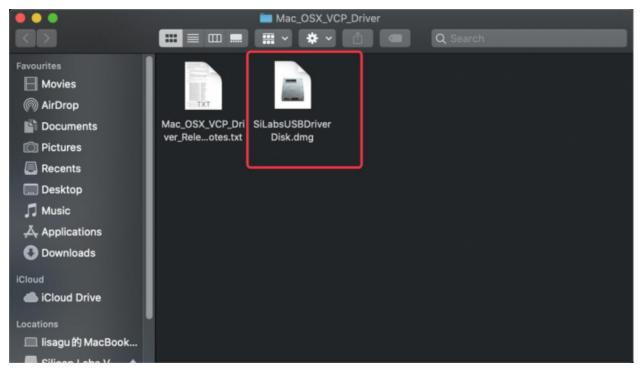
Platform	Software	Release Notes
Linux 3 x x and 4 x x	Download VCP (10.0 KB)	Download Unux 3xx and 4xx VCP Revision History
∆ Linux 2.6.x	Download VCP (10.2 K8)	Download Linux 2.6.x VCP Revision History

\*Note: The Linux 3xx and 4xx version of the driver is maintained in the current Linux 3xx and 4xx tree at www.kernel.org.

### 4Unzip the downloaded package.



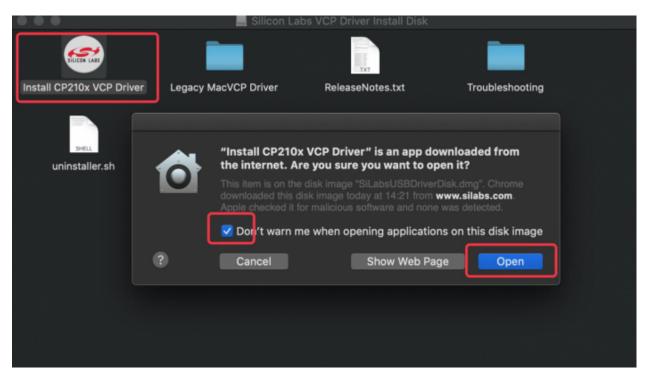
5Open folder and double-click"SiLabsUSBDriverDisk.dmg" file.



You will view the following files as follows:



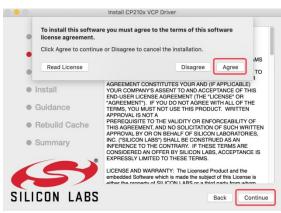
6Double-click\*\*"Install CP210x VCP Driver",\*\* check"**Don't warn me when opening application on this disk im-age**"and click"**Open**".



#### 7Click"Continue".



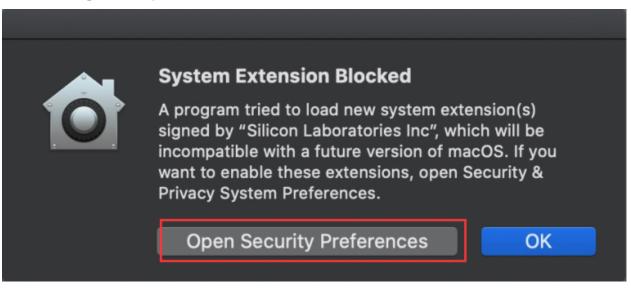
8Click "Agree" and then click "Continue".



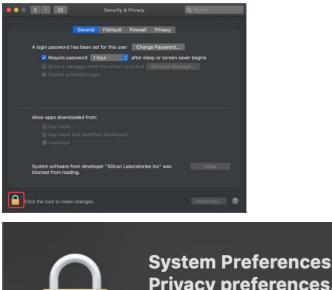
9Click "Continue" and enter your user password.

Introduction	To Be Installed: Version 5.3.5	
License	Currently Installed: None	
Info	Version 5.3.5 will be installed in /Library/Extensions/.	
Install		
<ul> <li>Guidance</li> </ul>		
Rebuild Cache		
<ul> <li>Summary</li> </ul>		
S	You will be prompted to enter your password.	
SILICON LABS	Back Continue	
	Install CP210x V helper tool. Enter your password	ng to install a new
	helper tool.	ng to install a new
	helper tool.	ng to install a new

10. Select "Open Security Preferences".



11Click the lock to unlock security & privacy preference, enter your user password to authorize, and then click"unlock".



# System Preferences is trying to unlock Security & Privacy preferences.

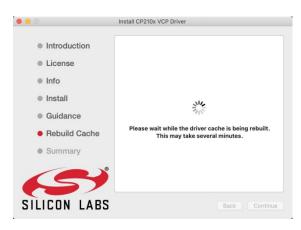
Enter your password to allow this.

Username: •		
Password:	••••	
	Car	ncel Unlock

12See the lock has been opened, click"Allow".

	Security & Privacy	Q Search
General	FileVault Firewall Privacy	
A login password has been set	for this user Change Password.	
Require password 1     Require password 1     Show a message when     Jisable automatic login	the screen is locked Set Lock Me	
Allow apps downloaded from: App Store App Store and identifie Anywhere	d developers	
System software from develop blocked from loading.	er "Silicon Laboratories Inc" was	Allow
ck the lock to prevent further cl	hanges.	Advanced ?

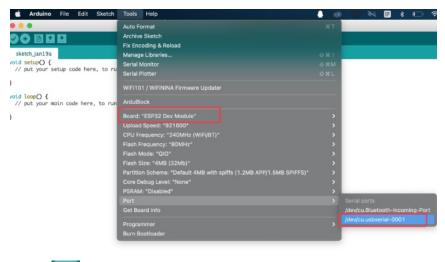
13Back to installation page, and wait to install.



#### 14Successfully installed.

	Install CP210x VCP Driver
<ul> <li>Introduction</li> <li>License</li> <li>Info</li> <li>Install</li> <li>Guidance</li> <li>Rebuild Cache</li> <li>Summary</li> </ul>	Success
SILICON LABS	Back Close

15Open arduinoIDEclick"Tools", select Board"ESP32 Dev Module" and port"/dev/cu.usbserial-0001".



16Click to upload code and show **Done uploading**".

## CHAPTER

# **FIVE**

# **ARDUINO PROJECT**

Click on the link to enter the Arduino IDE tutorial: Arduino IDE Tutorial

# 5.1 Download code files and Libraries files

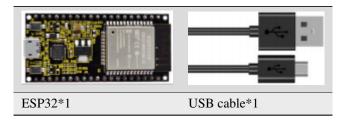
Click on the link to download the code file: Download Arduino C Codes file Click on the link to download the Libraries file: Download Libraries file

# 5.2 Project 01: Hello World

1. Introduction

For ESP32 beginners, we'll start with some simple things. In this project, you just need an ESP32 mainboard, USB cable and computer to complete "Hello World!" Project. It is not only a communication test for ESP32 mainboard and computer, but also a primary project for ESP32.

2. Components



## 3. Wiring

In this project, we use a USB cable to connect the ESP32 to the computer.



#### 4. Project code

You can open the code we provideIf you haven't downloaded the code file, please click on the link to download it:Download Arduino Codes

The code used in this project is saved in folder(path :) "Arduino-Codes\Project 01 HelloWorld\Project\_01\_Hello\_World".

```
/*
* Filename : Hello World
* Description : Enter the letter R, and the serial port displays"Hello World".
* Auther :http//www.keyestudio.com
*/
char val;// defines variable "val"
void setup()
{
Serial.begin(115200);// sets baudrate to 115200
}
void loop()
{
 if (Serial.available() > 0) {
  val=Serial.read();// reads symbols assigns to "val"
  if(val=='R')// checks input for the letter "R"
  { // if so,
   Serial.println("Hello World!");// shows "Hello World !".
  }
 }
}
```

Before uploading the project code to ESP32click"Tools"→"Board" and select"ESP32 Wrover Module".

Auto Format		Ctrl+T		ø		
Archive Sket	ch			_		Δ
ct_01_He Fix Encoding	& Reload					ESP32 Dev Module
Manage Libr	aries	Ctrl+Shift+I	*****	******	1.	ESP32 Wrover Module
Serial Monit	or	Ctrl+Shift+M				ESP32 Pico Kit
otio Serial Plotter		Ctrl+Shift+L	lays"Hello World".			TTGO LoRa32-OLED V1
WiFi101 / Wi	FiNINA Firmware Upo	later				XinaBox CW02
/ a	rinning rinning ope					SparkFun ESP32 Thing
) Board: "ESP3	2 Wrover Module"	;	Boards Manager			u-blox NINA-W10 series (ESP32
Upload Spee	d: "921600"	3	Arduino AVR Boards		;	Widora AIR
	ncy: "80MHz"	>	Arduino Mbed OS RP	2040 Boards	>	Electronic SweetPeas - ESP320
Flash Mode:	"QIO"	;	ESP32 Arduino		->	Nano32
Partition Sch	eme: "Default"	;	ESP8266 Boards (2.5.0	))	;	LOLIN D32
Core Debug	Level: "None"	>				LOLIN D32 PRO
Port: "COM3		>				WEMOS LOLIN32
Get Board In	fo					Dongsen Tech Pocket 32
Programme		>				"WeMos" WiFi&Bluetooth Batte
Burn Bootlo						ESPea32
		tor\Deskton\Ardu	ino\hardware\espressi	f\esn32\1		Noduino Ouantum
			ino\hardware\espressi		2	Node32s
					•	Hornbill ESP32 Dev
						Hornhill ESD22 Dev

Select the serial port.

Project_01_Hello_V	World   Arduino 1.8.16		- 🗆 X	
File Edit Sketch Too	ls Help			
	Auto Format	Ctrl+T		
	Archive Sketch			
Project_01_He	Fix Encoding & Reload			
//********	Manage Libraries	Ctrl+Shift+I	*****	^
/* * Filename	Serial Monitor	Ctrl+Shift+M		
* Descriptio	Serial Plotter	Ctrl+Shift+L	lays"Hello World".	
* Auther */	WiFi101 / WiFiNINA Firmware Updater			
char val;// d				
<pre>void setup()</pre>	Board: "ESP32 Wrover Module"	>		
{ Serial.begin(	Upload Speed: "921600"	>		
}	Flash Frequency: "80MHz"	>		
void loop()	Flash Mode: "QIO"	>		
{ if (Serial.	Partition Scheme: "Default"	>		
val=Seria	Core Debug Level: "None"	>		
if(val=='	Port: "COM3"	\$	Serial ports	
{ // if Serial.p	Get Board Info		COM3	
}	Programmer	>		~
	Burn Bootloader			
Invalid library f	ound in C:\Users\Administrator	Desktop\Ardu:	ino\hardware\espressif\esp32\1:	~
Invalid library f	ound in C:\Users\Administrator	Desktop\Ardu	ino\hardware\espressif\esp32\l:	
<				~
1	ESP32	Wrover Modul <u>e, De</u>	fault, QIO, 80MHz, 921600, None on COM3	

**Note:** For macOS users, if the uploading fails, please set the baud rate to 115200 before clicking

Project_01_Hello_V				_		×
File Edit Sketch Tool	s Help					
00 6	Auto Format	Ctrl+T				Ø
	Archive Sketch					
Project_01_He	Fix Encoding & Reload					
//*****	Manage Libraries	Ctrl+Shift+I	*****	******	******	rk 🖊
/* * Filename	Serial Monitor	Ctrl+Shift+M				
* Descriptio	Serial Plotter	Ctrl+Shift+L	lays"Hello	World".		
* Auther			-			
*/ char val;// d	WiFi101 / WiFiNINA Firmware Updater		_			
void setup()	Board: "ESP32 Wrover Module"	2	>			
{	Upload Speed: "921600"	;	921600			
Serial.begin(	Flash Frequency: "80MHz"	;	115200			
void loop()	Flash Mode: "QIO"	;	256000			
{	Partition Scheme: "Default"	;	230400			
if (Serial. val=Seria	Core Debug Level: "None"	;	512000			
if(val=='	Port: "COM3"	,	>	]		
{ // if	Get Board Info					
Serial.p			-			
1	Programmer	2	>			~
	Burn Bootloader					
1	ESP32	Wrover Module, D	efault, QIO, 80MH	Iz, 921600, N	lone on C	омз
lick to download	the code to ESP32.					

🥯 Project_01_Hello_World   Arduino 1.8.16			$\times$
<u>File Edit Sketch Tools H</u> elp			
			Ø
Project_01_Hello_World			
//******************	***********	******	** 🔺
/*			
<ul> <li>* Filename : Hello World</li> <li>* Description : Enter the letter R, and the serial port displays"He</li> </ul>	llo World"		
* Auther :http://www.keyestudio.com	siio world .		
*/			
char val;// defines variable "val"			
<pre>void setup()</pre>			
{			
<pre>Serial.begin(115200);// sets baudrate to 115200</pre>			
}			
<pre>void loop() {</pre>			
if (Serial.available() > 0) {			
<pre>val=Serial.read();// reads symbols assigns to "val"</pre>			
if(val=='R')// checks input for the letter "R"			
{ // if so,			
<pre>Serial.println("Hello World!");// shows "Hello World !".</pre>			
}			
}			
}			~
Uploading			
Hash of data verified. Compressed 17168 bytes to 10937			^
Wrote 17168 bytes (10937 compressed) at 0x00001000 in 0.2 seconds (6	effective 83	7.5 kbi	t/s)
Hash of data verified.			
Compressed 232464 bytes to 113899			
Writing at 0x00024000 (85 %)			$\sim$
< · · · · · · · · · · · · · · · · · · ·			>
1 ESP32 Wrover Module, Default, QIO,	80MHz, 921600,	None on (	сомз
Note: If uploading the code fails, you can press the Boot button on ESP32 after	alial and	release	the Roo
Note: If uproading the code rans, you can press the boot button on ESP32 after a	chekana, allu	release	
RESET	OOT		
button	after	the perc	entage of

button

uploading progress appears, as shown below:

after the percentage of

Uploading	
Hash of data verified.	
Compressed 17168 bytes to 10937	
Wrote 17168 bytes (10937 compressed) at	0x00001000 in 0.2 seconds (effective 832.4 kbit/s)
Hash of data verified.	
Compressed 232464 bytes to 113899	
Writing at 0x00024000 (85 %)	
<	>
1	ESP32 Wrover Module, Default, QIO, 80MHz, 921600, None on COM3

### The Project code is uploaded successfully

Project_01_Hello_World   Arduino 1.8.16		$\times$
<u>Eile Edit Sketch Tools H</u> elp		
		ø
Project_01_Hello_World		
//*************************************	*****	* ^
/* * Filename : Hello World		
* Description : Enter the letter R, and the serial port displays"Hello World".		
* Auther :http://www.keyestudio.com		
*/		
char val;// defines variable "val"		
<pre>void setup()</pre>		
{		
Serial.begin(115200);// sets baudrate to 115200		
}		
<pre>void loop() .</pre>		
{     if (Serial.available() > 0) {		
<pre>val=Serial.read();// reads symbols assigns to "val"</pre>		
if(val=='R')// checks input for the letter "R"		
{ // if so,		
Serial.println("Hello World!");// shows "Hello World !".		
}		
}		$\sim$
Done uploading.		
Invalid library found in C:\Users\Administrator\Desktop\Arduino\hardware\espress:	if\esp3	2\1:
Invalid library found in C:\Users\Administrator\Desktop\Arduino\hardware\espress:	if\esp3	2\1:
		$\sim$
<		>
1 ESP32 Wrover Module, Default, QIO, 80MHz, 921600, N	one on C	омз

5. Project result

After the project code is uploaded successfully, power up with a USB cable and click the icon to enter the serial monitor.

Set baud rate to 115200 and type "R" in the text box. Click "Send", and the serial monitor will display "Hello World!".

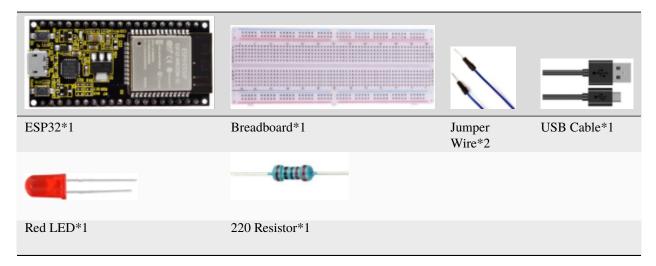
COM23		_	
R 2			Send
Hello World! Hello World!		3	^
		10	~
🗹 Autoscroll 🗌 Show timestamp	Newline ~	115200 baud 🗸	Clear output

# 5.3 Project 02: Turn On LED

1. Introduction

In this project, we will show you how to light up the LED. We use the ESP32's digital pin to turn on the LED so that the LED is lit up.

2. Components

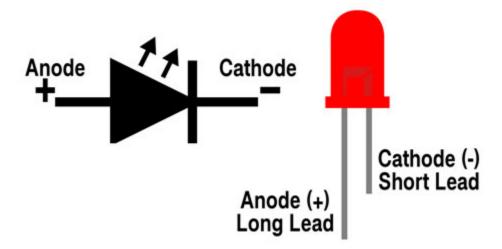


3. Component knowledge

1LED:

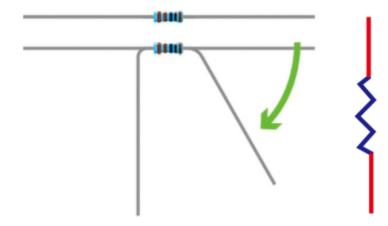


The LED is a semiconductor known as "light-emitting diode", which is an electronic device made from semiconducting materials(silicon, selenium, germanium, etc.). It has an anode and a cathode, the short lead is cathode, which connects to GND; the long lead is anode, which connects to 3.3V or 5V.



#### 2Five-color ring resistor

A resistor is an electronic component in a circuit that restricts or regulates the flow current flow. On the left is the appearance of the resistor and on the right is the symbol for the resistance in the circuit. Its unit is(). 1 m = 1000 k1k = 1000.



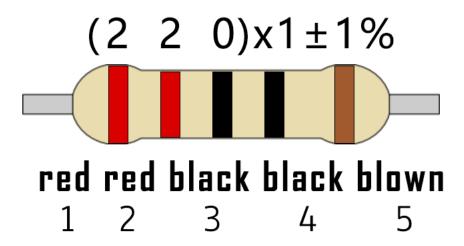
We can use resistors to protect sensitive components, such as LED. The strength of the resistance is marked on the body of the resistor with an electronic color code. Each color code represents a number, and you can refer to it in a resistance card.

- -Color 1 1st Digit.
- -Color 2 2nd Digit.
- -Color 3 3rd Digit.
- -Color 4 Multiplier.
- -Color 5 Tolerance.

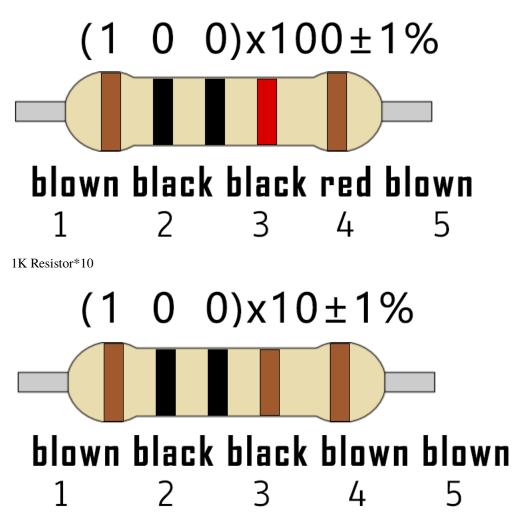
	1st Digit	2nd Digit	3rd Digit	Multiplier	Tolerance
Black		0	0	x1	
Brown	1	1	1	x10	± 1%
Red	2	2	2	x100	± 2%
Orange	3	3	3	x1K	± 3%
Yellow	4	4	4	x10K	± 4%
Green	5	5	5	x100K	±0.5%
Blue	6	6	6	x1M	±0.25%
Violet	7	7	7	x10M	±0.10%
Grey	8	8	8	x100M	±0.05%
White	9	9	9	x1G	
Gold				÷ 10	± 5%
Silver				÷ 100	± 10%

In this kit, we provide three Five-color ring resistor with different resistance values. Take three Five-color ring resistor as an example.

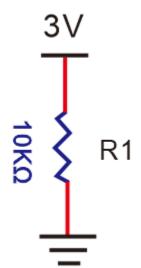
220 Resistor\*10



10K Resistor\*10



In the same voltage, there will be less current and more resistance. The connection between current(I), voltage(V), and resistance (a) can be expressed by the formula: I=U/R. In the figure below, if the voltage is 3V, the current through R1 is: I = U / R = 3 V / 10 K = 0.0003 A = 0.3 mA.



Don't connect a low resistance directly to the two poles of the power supply. as this will cause excessive current to

damage the electronic components. Resistors do not have positive and negative poles.

### **3Bread board**

Breadboards are used to build and test circuits quickly before completing any circuit design. There are many holes in the breadboard that can be inserted into circuit components such as integrated circuits and resistors. A typical breadboard is shown below



The breadboard has strips of metal, which run underneath the board and connect the holes on the top of the board. The metal strips are laid out as shown below. Note that the top and bottom rows of holes are connected horizontally, while the remaining holes are connected vertically.

media/b45e70b961537035c85878b73d371725.png

The first two rows (top) and the last two rows (bottom) of the breadboard are used for the positive pole (+) and negative pole (-) of the power supply respectively. The conductive layout of the breadboard is shown in the figure below:

media/d5478bd5eac558252cbc235479d979eb.png

When we connect DIP (Dual In-line Packages) components, such as integrated circuits, microcontrollers, chips and so on, we can see that a groove in the middle isolates the middle part, so the top and bottom of the groove is not connected. DIP components can be connected as shown in the following diagram:

media/50caf14e911c4244779e99445c658db6.png

media/9b66ae2199e77fbc99b7b278dac0b567.png

[Power](javascript:;) [Supply](javascript:;)

The ESP32 needs 3.3V-5V power supply. In this project, we connected the ESP32 to the computer by using a USB cable.



4. Wiring diagram

First, disconnect all power from the ESP32. Then build the circuit according to the wiring diagram. After the circuit is built and verified correct, connect the ESP32 to your computer by using a USB cable.

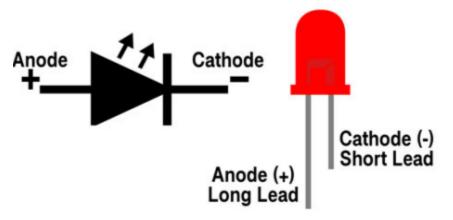
Note: Avoid any possible short circuits (especially connecting 3.3V and GND)!

**WARNING:** A short circuit can cause high current in your circuit, create excessive component heat and cause permanent damage to your hardware!

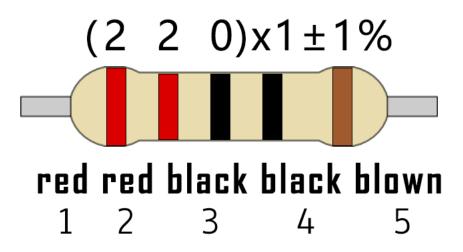


Note:

How to connect a LED



How to identify the 220 Five-color ring resistor



5. Project code

You can open the code we provideIf you haven't downloaded the code file, please click on the link to download it:Download Arduino Codes

The code used in this project is saved in folder(path:)"Arduino-Codes\Project 02Turn On LED\Project\_02\_Turn\_On\_LED".

```
/*
* Filename : Turn On LED
* Description : Make an led on.
* Auther
       : http//www.keyestudio.com
*/
##define LED_BUILTIN 15
// the setup function runs once when you press reset or power the board
void setup() {
 // initialize digital pin LED_BUILTIN as an output.
 pinMode(LED_BUILTIN, OUTPUT);
}
void loop() {
 digitalWrite(LED_BUILTIN, HIGH); // turn the LED on (HIGH is the voltage level)
}
```

Before uploading the project code to ESP32click"Tools"→"Board" and select"ESP32 Wrover Module".

ile Edit Sketch To	ols Help				
	Auto Format	Ctrl+T			
	Archive Sketch				Δ
Project_02_Tu	Fix Encoding & Reload				ESP32 Dev Module
//**********	Manage Libraries	Ctrl+Shift+I	*****	1.	ESP32 Wrover Module
* Filename	Serial Monitor	Ctrl+Shift+M			ESP32 Pico Kit
* Descriptio	Serial Plotter	Ctrl+Shift+L			TTGO LoRa32-OLED V1
* Auther	WiFi101 / WiFiNINA Firmware Updater	r			XinaBox CW02
define LED_B				-	SparkFun ESP32 Thing
	Board: "ESP32 Wrover Module"	3	Boards Manager	_	u-blox NINA-W10 series (ESP32)
// the setup	Upload Speed: "921600"	3	Arduino AVR Boards	2	Widora AIR
// initiali	Flash Frequency: "80MHz"	3	Arduino Mbed OS RP2040 Boards	;	Electronic SweetPeas - ESP320
pinMode(LED	Flash Mode: "QIO"	3	ESP32 Arduino	2	Nano32
} /oid loop() {	Partition Scheme: "Default"	3	ESP8266 Boards (2.5.0)	>	LOLIN D32
digitalWrit	Core Debug Level: "None"	>	H is the voltage level)		LOLIN D32 PRO
} //*****	Port: "COM3"	>	******		WEMOS LOLIN32
1	Get Board Info				Dongsen Tech Pocket 32
	Programmer	>			"WeMos" WiFi&Bluetooth Battery
	Burn Bootloader				ESPea32
					Noduino Quantum
					Node32s
					Hornbill ESP32 Dev
					Hornbill ESP32 Minima
	ESP32	Wrover Module, De	efault, QIO, 80MHz, 921600, None on COM3		EiroPoetlo ESD22

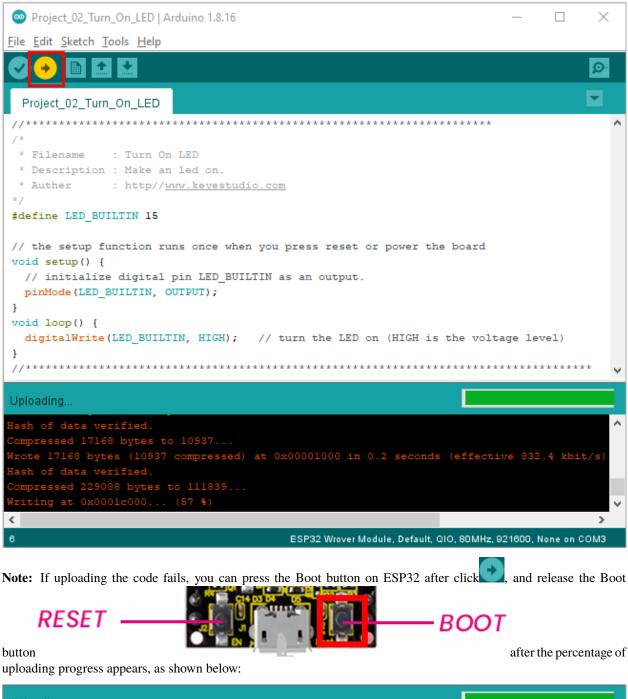
Select the serial port.

Project_02_Turn_	_On_LED   Arduino 1.8.16			_		×
File Edit Sketch To	ools Help					
	Auto Format Archive Sketch	Ctrl+T				Ø
Project_02_Tu	Fix Encoding & Reload					•
//*******	Manage Libraries	Ctrl+Shift+I	*****			^
/* * Filename	Serial Monitor	Ctrl+Shift+M				
* Descriptio	Serial Plotter	Ctrl+Shift+L				
* Auther */ #define LED B	WiFi101 / WiFiNINA Firmware Update	r				
#deline PED_P	Board: "ESP32 Wrover Module"	>				
// the setup	Upload Speed: "921600"	>	r the board			
<pre>void setup()     // initiali</pre>	Flash Frequency: "80MHz"	>				
pinMode(LED	Flash Mode: "QIO"	>				
<pre>} void loop() {</pre>	Partition Scheme: "Default"	>				
digitalWrit	Core Debug Level: "None"	>	H is the volta	age leve	1)	
}	Port: "COM3"	>	Serial ports	******	*****	
//	Get Board Info		🗸 сомз			
	Programmer	>				~
	Burn Bootloader					
	found in C:\Users\Administrator					
Invalid library	found in C:\Users\Administrator	\Desktop\Ardu	ino\hardware\e	spressif	\esp3	2\1:
<						>
1	ESP32	Wrover Module, De	fault, QIO, 80MHz, 9	21600, Non	e on CC	омз

Note: For macOS users, if the uploading fails, please set the baud rate to 115200 before clicking

	Auto Format	Ctrl+T			Ø
	Archive Sketch				
Project_02_Tu	Fix Encoding & Reload				
/*****	Manage Libraries	Ctrl+Shift+I	*****		
* * Filename	Serial Monitor	Ctrl+Shift+M			
* Descriptio	Serial Plotter	Ctrl+Shift+L			
* Auther					
/ define LED B	WiFi101 / WiFiNINA Firmware Updat	er			
aerine ppp_p	Board: "ESP32 Wrover Module"	>	•		
/ the setup	Upload Speed: "921600"	;	• 921600		
<pre>pid setup()     // initiali</pre>	Flash Frequency: "80MHz"	;	115200		
pinMode (LED	Flash Mode: "QIO"	;	256000		
	Partition Scheme: "Default"	;	230400		
<pre>digitalWrit</pre>	Core Debug Level: "None"	;	512000 1	tage level)	
-	Port: "COM3"	2	>		
/*****	Get Board Info		******	******	****
	Programmer	>	·		
	Burn Bootloader				

Click to download the code to ESP32.



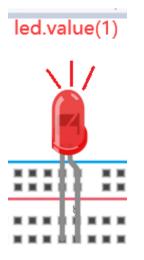
Uploading	
Hash of data verified.	~
Compressed 17168 bytes to 10937	
Wrote 17168 bytes (10937 compressed) at 0x00001000 in 0.2 seconds (effective 832.	.4 kbit/s)
Hash of data verified.	
Compressed 232464 bytes to 113899	
Writing at 0x00024000 (85 %)	$\sim$
<	>
1 ESP32 Wrover Module, Default, QIO, 80MHz, 921600, N	one on COM3

# The Project code is uploaded successfully

Project_02_Turn_On_LED   Arduino 1.8.16	_		$\times$
<u>File Edit Sketch Tools H</u> elp			
			Ø
Project_02_Turn_On_LED			
//*************************************			^
<pre>* Filename : Turn On LED * Description : Make an led on.</pre>			
* Auther : http://www.keyestudio.com			
*/ #define LED_BUILTIN 15			
<pre>// the setup function runs once when you press reset or power the board void setup() { // initialize digital pin LED_BUILTIN as an output. pinMode(LED BUILTIN, OUTPUT);</pre>			
}			
<pre>void loop() {     digitalWrite(LED_BUILTIN, HIGH); // turn the LED on (HIGH is the volt)</pre>	age lev	/el)	
} //***********************************	*****	******	* •
Done uploading.			
Invalid library found in C:\Users\Administrator\Desktop\Arduino\hardware\e			P 1
Invalid library found in C:\Users\Administrator\Desktop\Arduino\hardware\e	spress	if\esp3	2\1
<			> `
6 ESP32 Wrover Module, Default, QIO, 80MHz, 8	21600, N	lone on C	омз

6. Project result

After the project code was uploaded successfully, power up with a USB cable and the LED is lit up.

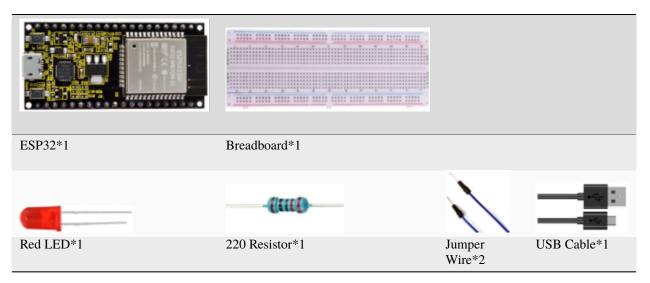


# 5.4 Project 03LED Flashing

1. Introduction

In this project, we will show you the LED flashing effect. We use the ESP32's digital pin to turn on the LED and make it flashing.

2. Components



3. Wiring diagram

First, disconnect all power from the ESP32. Then build the circuit according to the wiring diagram. After the circuit is built and verified correct, connect the ESP32 to your computer using a USB cable.

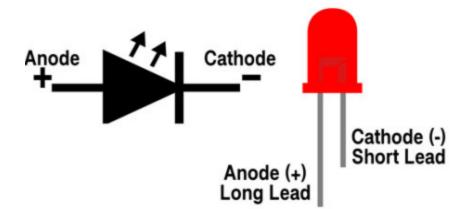
Note: Avoid any possible short circuits (especially connecting 3.3V and GND)!

**WARNING:** A short circuit can cause high current in your circuit, create excessive component heat and cause permanent damage to your hardware!

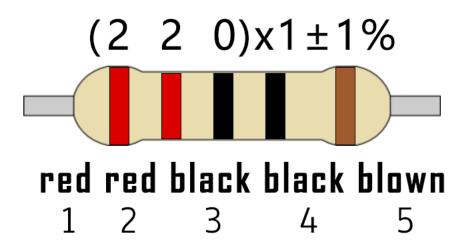


Note:

How to connect a LED



How to identify the 220 Five-color ring resistor



### 4. Project code

You can open the code we provideIf you haven't downloaded the code file, please click on the link to download it:Download Arduino Codes

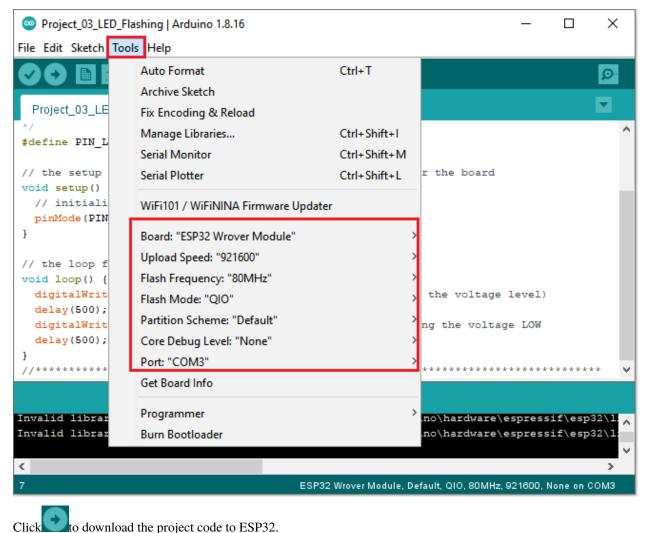
The code used in this project is saved in folder "Arduino-Codes\Project 03LED Flashing\Project\_03\_LED\_Flashing".

(continues on next page)

(continued from previous page)

Before uploading Project Code to ESP32, please check the configuration of Arduino IDE.

Click "Tools" to confirm the board type and port as shown below:



Project_03_LED_Flashing   Arduino 1.8.16	_		×
<u>Eile Edit Sketch Tools H</u> elp			
			Ø
Project_03_LED_Flashing			•
#define PIN_LED 15 //define the led pin			^
<pre>// the setup function runs once when you press reset or power the board void setup() {</pre>			
<pre>// initialize digital pin LED as an output. pinMode(PIN_LED, OUTPUT);</pre>			
}			
<pre>// the loop function runs over and over again forever void loop() {</pre>			
<pre>digitalWrite(PIN_LED, HIGH); // turn the LED on (HIGH is the voltage le delay(500); // wait for 0.5s</pre>	evel)		
<pre>digitalWrite(PIN_LED, LOW); // turn the LED off by making the voltage     delay(500); // wait for 0.5s</pre>	LOW		
}			
//*************************************	*****	*****	* ¥
Uploading			
Hash of data verified.			^
Compressed 229264 bytes to 107533 Writing at 0x00024000 (85 %)			~
<			>
7 ESP32 Wrover Module, Default, QIO, 80MHz, 92	1600, N	one on C(	омз

Note: If uploading the code fails, you can press the Boot button on ESP32 after click and release the Boot



after the percentage of

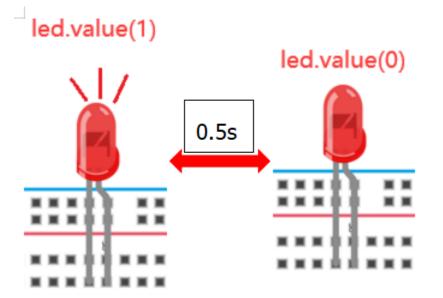
uploading progress appears, as shown below:

The Project code is uploaded successfully

```
Project 03 LED Flashing | Arduino 1.8.16
                                                                                 \times
File Edit Sketch Tools Help
                +
            ÷
                                                                                      Ø
     +
         -
  Project_03_LED_Flashing
#define PIN LED
                  15
                      //define the led pin
// the setup function runs once when you press reset or power the board
void setup() {
  // initialize digital pin LED as an output.
  pinMode(PIN_LED, OUTPUT);
}
// the loop function runs over and over again forever
void loop() {
  digitalWrite(PIN_LED, HIGH); // turn the LED on (HIGH is the voltage level)
                                   // wait for 0.5s
  delay(500);
  digitalWrite(PIN_LED, LOW); // turn the LED off by making the voltage LOW
  delay(500);
                                   // wait for 0.5s
}
                      **********
11
Done uploading.
Invalid liprary found in C:\Users\Administrator\Desktop\Arduino\hardware\espressif\esp32\1.
Invalid library found in C:\Users\Administrator\Desktop\Arduino\hardware\espressif\esp32\1
<
                                                                                       >
                                         ESP32 Wrover Module, Default, QIO, 80MHz, 921600, None on COM3
```

#### 5. Project result

After the project code was uploaded successfully, power up with a USB cable and the LED start flashing.



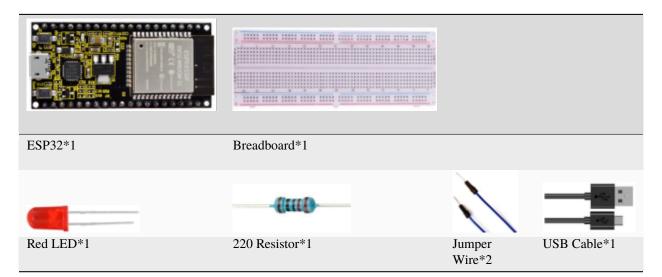
# 5.5 Project 04: Breathing Led

### 1. Introduction

In previous studies, we know that LEDs have on/off state, so how to enter the intermediate state? How to output an intermediate state to make the LED half bright? That's what we're going to learn.

Breathing light, that is, LED is turned from off to on gradually, and gradually from on to off, just like "breathing". So, how to control the brightness of a LED? We will use ESP32's PWM to achieve this target.

2. Components

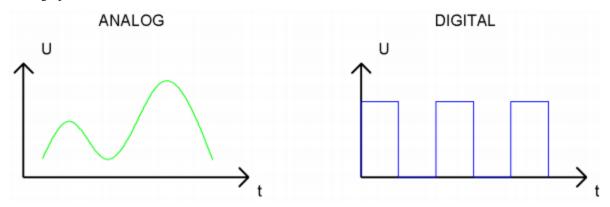


3. Component knowledge



### Analog & Digital

An Analog Signal is a continuous signal in both time and value. On the contrary, a Digital Signal or discrete time signal is a time series consisting of a sequence of quantities. Most signals in life are analog signals. A familiar example of an Analog Signal would be how the temperature throughout the day is continuously changing and could not suddenly change instantaneously from  $0^{\circ}$ C to  $10^{\circ}$ C. However, Digital Signals can instantaneously change in value. This change is expressed in numbers as 1 and 0 (the basis of binary code). Their differences can more easily be seen when compared when graphed as below.



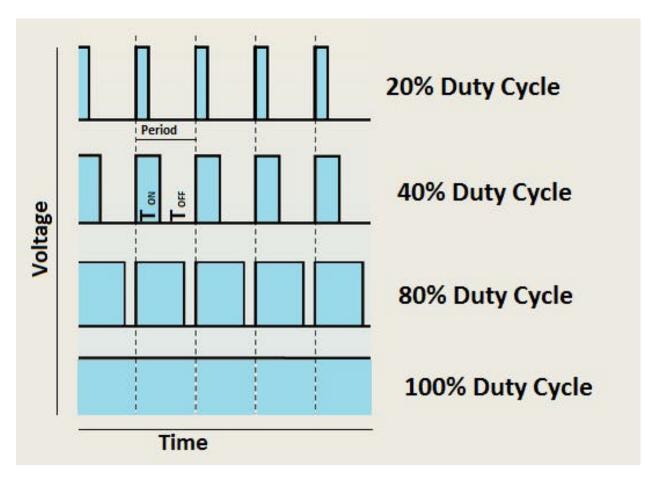
In practical application, we often use binary as the digital signal, that is a series of 0's and 1's. Since a binary signal only has two values (0 or 1), it has great stability and reliability. Lastly, both analog and digital signals can be converted into the other.

### PWM

PWM, Pulse-Width Modulation, is a very effective method for using digital signals to control analog circuits. Common processors cannot directly output analog signals. PWM technology makes it very convenient to achieve this conversion (translation of digital to analog signals).

PWM technology uses digital pins to send certain frequencies of square waves, that is, the output of high levels and low levels, which alternately last for a while. The total time for each set of high levels and low levels is generally fixed, which is called the period (Note: the reciprocal of the period is frequency). The time of high level outputs are generally called "pulse width", and the duty cycle is the percentage of the ratio of pulse duration, or pulse width (PW) to the total period(T) of the waveform.

The longer the output of high levels last, the longer the duty cycle and the higher the corresponding voltage in the analog signal will be. The following figures show how the analog signal voltages vary between 0V-3V3 (high level is 3V3) corresponding to the pulse width 0%-100%:

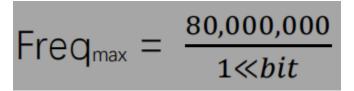


The longer the PWM duty cycle is, the higher the output power will be. Now that we understand this relationship, we can use PWM to control the brightness of an LED or the speed of DC motor and so on. It is evident from the above that PWM is not real analog, and the effective value of the voltage is equivalent to the corresponding analog. So, we can control the output power of the LED and other output modules to achieve different effects.

## ESP32 and PWM:

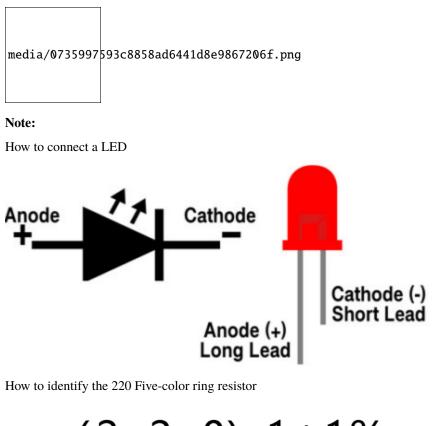
On ESP32, the LEDC(PWM) controller has 16 separate channels, each of which can independently control frequency, duty cycle, and even accuracy. Unlike traditional PWM pins, the PWM output pins of ESP32 are configurable, with one or more PWM output pins per channel. The relationship between the maximum

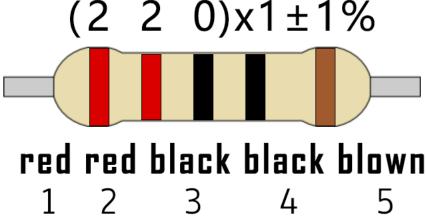
frequency and bit precision is shown in the following formula, where the maximum value of bit is 31.



For example, generate a PWM with an 8-bit precision (28=256. Values range from 0 to 255) with a maximum frequency of 80,000,000/255 = 312,500Hz.

4. Wiring diagram





5. Project code

The design of this project makes the GP15 output PWM, and the pulse width gradually increases from 0% to 100%, and then gradually decreases from 100% to 0%.

You can open the code we provideIf you haven't downloaded the code file, please click on the link to download it:Download Arduino Codes

The code used in this project is saved in folder "Arduino-Codes\Project 04Breathing Led\Project\_04\_Breathing\_Led".

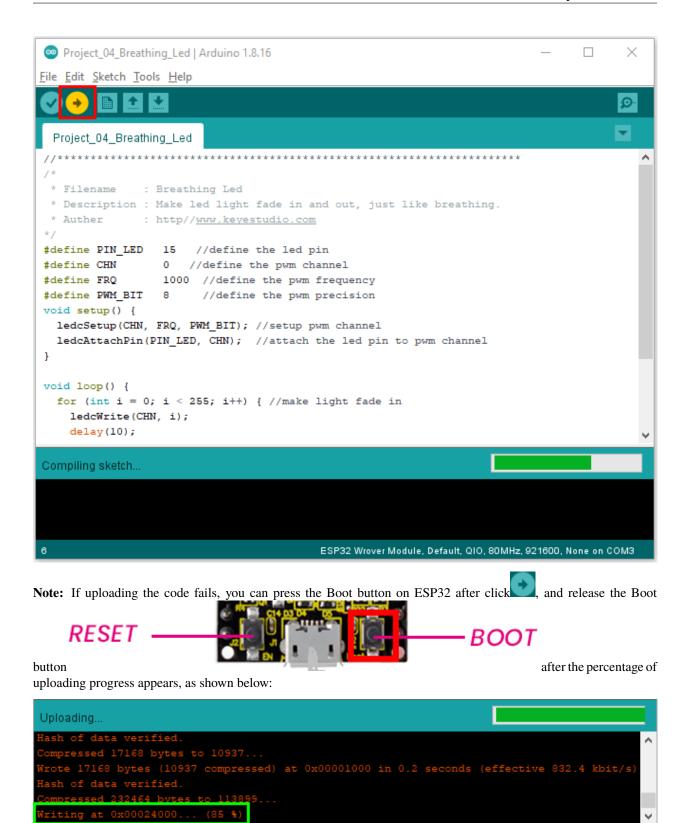
```
* Description : Make led light fade in and out, just like breathing.
* Auther
            : http//www.keyestudio.com
*/
##define PIN_LED 15 //define the led pin
##define CHN
                0 //define the pwm channel
                1000 //define the pwm frequency
##define FRQ
                     //define the pwm precision
##define PWM_BIT 8
void setup() {
 ledcSetup(CHN, FRQ, PWM_BIT); //setup pwm channel
 ledcAttachPin(PIN_LED, CHN); //attach the led pin to pwm channel
}
void loop() {
 for (int i = 0; i < 255; i++) { //make light fade in
   ledcWrite(CHN, i);
   delay(10);
 }
 for (int i = 255; i > -1; i--) { //make light fade out
   ledcWrite(CHN, i);
   delay(10);
 }
}
```

Before uploading Project Code to ESP32, please check the configuration of Arduino IDE.

Click "**Tools**" to confirm the board type and port as shown below:

Project_04_Brea	thing_Led   Arduino 1.8.16			_		$\times$
File Edit Sketch	pols Help					
	Auto Format	Ctrl+T				Ø
	Archive Sketch					
Project_04_Bre	Fix Encoding & Reload					
//*******	Manage Libraries	Ctrl+Shift+I	****			^
/* * Filename	Serial Monitor	Ctrl+Shift+M				
* Descriptio	Serial Plotter	Ctrl+Shift+L	reathing.			
* Auther	WiFi101 / WiFiNINA Firmware Updater					
#define PIN_L						
#define CHN	Board: "ESP32 Wrover Module"	>				
#define FRQ	Upload Speed: "921600"	>				
<pre>#define PWM_B void setup()</pre>	Flash Frequency: "80MHz"	>				
ledcSetup()	Flash Mode: "QIO"	>				
ledcAttachP	Partition Scheme: "Default"	>	channel			
}	Core Debug Level: "None"	>				- 1
<pre>void loop() {</pre>	Port: "COM3"	>				
for (int i ledcWrite	Get Board Info					
delay(10)	Programmer	>				~
	Burn Bootloader					
6	ESP32	Wrover Module, De	fault, QIO, 80MHz, 9	921600, N	ione on (	сомз
$\frown$						

Click to download the project code to ESP32.



The Project code is uploaded successfully!

<

х

Project_04_Breathing_Led   Arduino 1.8.16	_		×
<u>File Edit Sketch Tools H</u> elp			
			ø
Project_04_Breathing_Led			
//*************************************			^
* Filename : Breathing Led			
<pre>* Description : Make led light fade in and out, just like breathing. * Auther : http//www.keyestudio.com */</pre>			
#define PIN_LED 15 //define the led pin			
#define CHN 0 //define the pwm channel			
#define FRQ 1000 //define the pwm frequency			
#define PWM_BIT 8 //define the pwm precision			
<pre>void setup() {</pre>			
<pre>ledcSetup(CHN, FRQ, PWM_BIT); //setup pwm channel</pre>			
<pre>ledcAttachPin(PIN_LED, CHN); //attach the led pin to pwm channel</pre>			
}			
<pre>void loop() {    for (int i = 0; i &lt; 255; i++) { //make light fade in</pre>			
<pre>ledcWrite(CHN, i);</pre>			
delay(10);			
			~
Done uploading.			
Invalid library found in C:\Users\Administrator\Desktop\Arduino\hardware\e			
Invalid library found in C:\Users\Administrator\Desktop\Arduino\hardware\e	spress:	if\esp	
			<u> </u>
<			>
6 ESP32 Wrover Module, Default, QIO, 80MHz, 8	21600, N	one on C	юмз

### 6. Project result

After the project code was uploaded successfully, power up with a USB cable and the LED is turned from ON to OFF and then back from OFF to ON gradually like breathing.

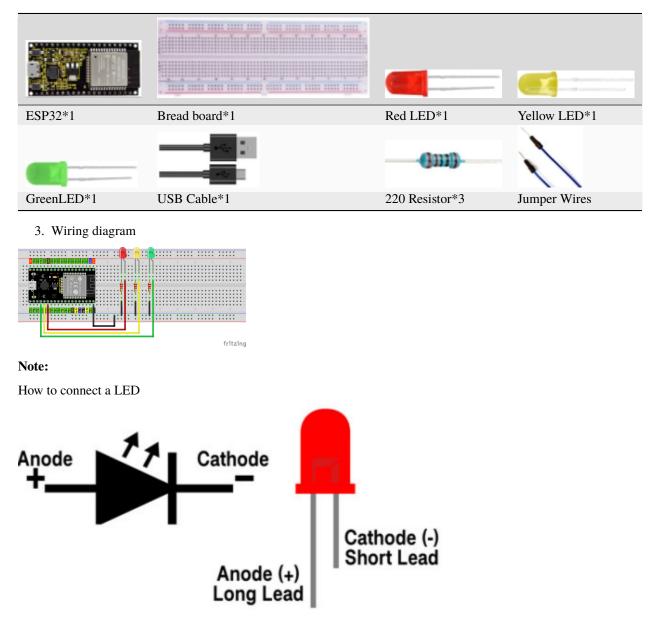


# 5.6 Project 05Traffic Lights

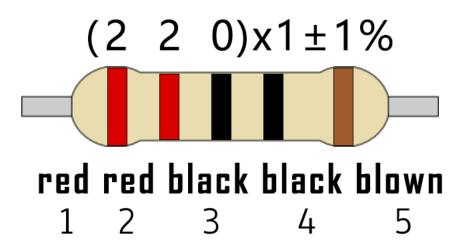
### 1. Introduction

Traffic lights are closely related to people's daily lives, which generally show red, yellow, and green. Everyone should obey the traffic rules, which can avoid many traffic accidents. In this project, we will use ESP32 and some LEDs (red, green and yellow) to simulate the traffic lights.

2. Components



How to identify the 220 Five-color ring resistor



4. Project code

You can open the code we provideIf you haven't downloaded the code file, please click on the link to download it:Download Arduino Codes

The code used in this project is saved in folder "Arduino-Codes\Project 05Traffic Lights\Project\_05\_Traffic\_Lights".

```
/*
* Filename : Traffic Lights
* Description : Simulated traffic lights.
 * Auther
         : http//www.keyestudio.com
*/
##define PIN_LED_RED 0 //define the red led pin
##define PIN_LED_YELLOW 2 //define the yellow led pin
##define PIN_LED_GREEN 15 //define the green led pin
void setup() {
  pinMode(PIN_LED_RED, OUTPUT);
 pinMode(PIN_LED_YELLOW, OUTPUT);
 pinMode(PIN_LED_GREEN, OUTPUT);
}
void loop() {
  digitalWrite(PIN_LED_RED, HIGH);// turns on the red led
  delay(5000);// delays 5 seconds
  digitalWrite(PIN_LED_GREEN, LOW); // turns off the green led
  for(int i=0;i<3;i++)// flashes 3 times.</pre>
{
  delay(500);// delays 0.5 second
  digitalWrite(PIN_LED_YELLOW, HIGH);// turns on the yellow led
  delay(500);// delays 0.5 second
  digitalWrite(PIN_LED_YELLOW, LOW);// turns off the yellow led
}
  delay(500);// delays 0.5 second
  digitalWrite(PIN_LED_GREEN, HIGH);// turns on the green led
  delay(5000);// delays 5 second
                                                                      (continues on next page)
```

```
digitalWrite(PIN_LED_RED, LOW);// turns off the red led
```

Before uploading Project Code to ESP32, please check the configuration of Arduino IDE.

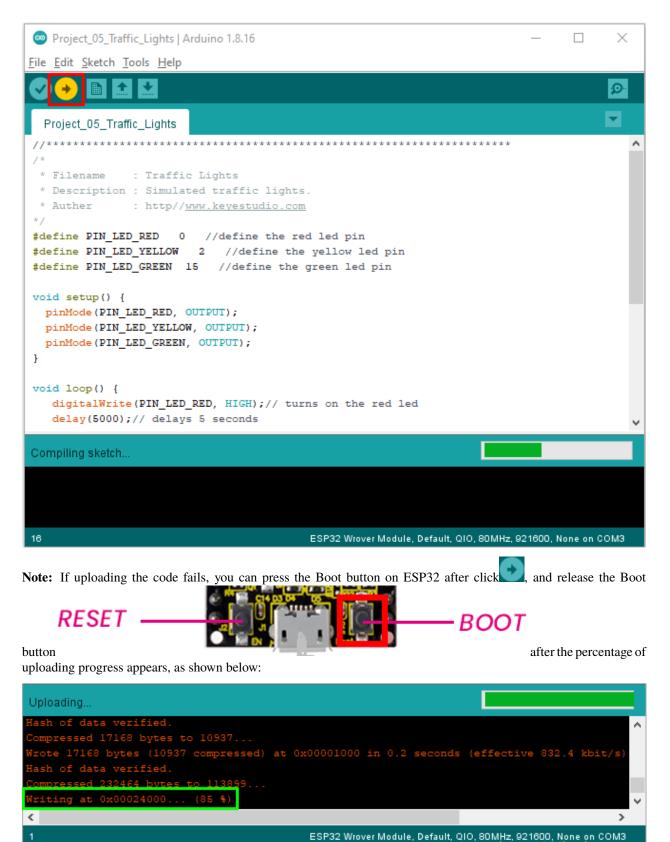
Click "Tools" to confirm the board type and port as shown below:

}

11

Project_05_Traffic_	Lights   Arduino 1.8.16			_		×
File Edit Sketch Tool	ls Help					
	Auto Format	Ctrl+T				Ø
	Archive Sketch					_
Project_05_Tra	Fix Encoding & Reload					
//*******	Manage Libraries	Ctrl+Shift+I	******			^
/* * Filename	Serial Monitor	Ctrl+Shift+M				
* Descriptio	Serial Plotter	Ctrl+Shift+L				
* Auther */	WiFi101 / WiFiNINA Firmware Updater					
<pre>#define PIN_L #define PIN_L</pre>	Board: "ESP32 Wrover Module"	>	1			
#define PIN_L	Upload Speed: "921600"	>				
void setup()	Flash Frequency: "80MHz"	>				
pinMode (PIN	Flash Mode: "QIO"	>				
pinMode(PIN pinMode(PIN	Partition Scheme: "Default"	>				
}	Core Debug Level: "None"	>				
	Port: "COM3"	>				
void loop() { digitalWri	Get Board Info					
delay(5000	Programmer	>				~
	Burn Bootloader					
	ound in C:\Users\Administrator\ ound in C:\Users\Administrator\					
						~
K						>
16	ESP32	Wrover Module, De	fault, QIO, 80MHz, 9	921600, N	ione on (	50M3

Click to download the project code to ESP32.



ESP32 wrover wrodule, Default, QIO, 80MH2, 921000, North

The Project code is uploaded successfully

<pre>Project_05_Traffic_Lights  //**********************************</pre>	Project_05_Traffic_Lights   Arduino 1.8.16	_		×
<pre>Project_05_Traffic_Lights //* * Filename : Traffic Lights * Description : Simulated traffic lights. * Auther : http://www.keyestudio.com */ tdefine PIN_LED_RED 0 //define the red led pin tdefine PIN_LED_RED 15 //define the yellow led pin tdefine PIN_LED_GREEN 15 //define the green led pin tdefine PIN_LED_RED, OUTPUT); pinMode(PIN_LED_RED, OUTPUT); pinMode(PIN_LED_CREEN, OUTPUT); } void loop() {     digitalWrite(PIN_LED_RED, HIGH);// turns on the red led     delay(5000);// delays 5 seconds </pre>	<u>F</u> ile <u>E</u> dit <u>S</u> ketch <u>T</u> ools <u>H</u> elp			
<pre>//***********************************</pre>				Ø
<pre>/* * Filename : Traffic Lights * Description : Simulated traffic lights. * Auther : http//www.kevestudio.com */ #define PIN_LED_RED 0 //define the red led pin #define PIN_LED_YELLOW 2 //define the yellow led pin #define PIN_LED_CREEN 15 //define the green led pin void setup() {     pinMode (PIN_LED_RED, OUTPUT);     pinMode (PIN_LED_RED, OUTPUT);     pinMode (PIN_LED_GREEN, OUTPUT); } void loop() {     digitalWrite (PIN_LED_RED, HIGH);// turns on the red led     delay(5000);// delays 5 seconds */ Done uploading. */ */ */ */ */ */ */ */ */ */ */ */ */</pre>	Project_05_Traffic_Lights			
<pre>* Filename : Traffic Lights * Description : Simulated traffic lights. * Auther : http//www.kevestudio.com */ #define PIN_LED_RED 0 //define the red led pin #define PIN_LED_YELLOW 2 //define the yellow led pin #define PIN_LED_GREEN 15 //define the green led pin void setup() { pinMode (PIN_LED_RED, OUTPUT); pinMode (PIN_LED_RED, OUTPUT); pinMode (PIN_LED_GREEN, OUTPUT); pinMode (PIN_LED_GREEN, OUTPUT); } void loop() { digitalWrite (PIN_LED_RED, HIGH);// turns on the red led delay(5000);// delays 5 seconds v</pre>	//*************************************			^
<pre>* Description : Simulated traffic lights. * Auther : http://www.keyestudio.com */ #define PIN_LED_RED 0 //define the red led pin #define PIN_LED_YELLOW 2 //define the yellow led pin #define PIN_LED_GREEN 15 //define the green led pin void setup() { pinMode (PIN_LED_RED, OUTPUT); pinMode (PIN_LED_YELLOW, OUTPUT); pinMode (PIN_LED_GREEN, OUTPUT); pinMode (PIN_LED_GREEN, OUTPUT); } void loop() { digitalWrite(PIN_LED_RED, HIGH);// turns on the red led delay(5000);// delays 5 seconds Done uploading. </pre>	/* * Tilesee - Teeffie Tielee			
<pre>* Auther : http//www.keyestudio.com */ #define PIN_LED_RED 0 //define the red led pin #define PIN_LED_YELLOW 2 //define the yellow led pin #define PIN_LED_GREEN 15 //define the green led pin void setup() {     pinMode (PIN_LED_RED, OUTPUT);     pinMode (PIN_LED_YELLOW, OUTPUT);     pinMode (PIN_LED_GREEN, OUTPUT);     pinMode (PIN_LED_GREEN, OUTPUT); } void loop() {     digitalWrite(PIN_LED_RED, HIGH);// turns on the red led     delay(5000);// delays 5 seconds  Done uploading</pre>				
<pre>#define PIN_LED_RED 0 //define the red led pin #define PIN_LED_YELLOW 2 //define the yellow led pin #define PIN_LED_GREEN 15 //define the green led pin void setup() {     pinMode(PIN_LED_RED, OUTPUT);     pinMode(PIN_LED_GREEN, OUTPUT);     pinMode(PIN_LED_GREEN, OUTPUT); } void loop() {     digitalWrite(PIN_LED_RED, HIGH);// turns on the red led     delay(5000);// delays 5 seconds v Done uploading. </pre>				
<pre>#define PIN_LED_YELLOW 2 //define the yellow led pin #define PIN_LED_GREEN 15 //define the green led pin void setup() {     pinMode(PIN_LED_RED, OUTPUT);     pinMode(PIN_LED_YELLOW, OUTPUT);     pinMode(PIN_LED_GREEN, OUTPUT); } void loop() {     digitalWrite(PIN_LED_RED, HIGH);// turns on the red led     delay(5000);// delays 5 seconds  Done uploading</pre>	*/			
<pre>#define PIN_LED_GREEN 15 //define the green led pin void setup() { pinMode(PIN_LED_RED, OUTPUT); pinMode(PIN_LED_YELLOW, OUTPUT); pinMode(PIN_LED_GREEN, OUTPUT); } void loop() { digitalWrite(PIN_LED_RED, HIGH);// turns on the red led delay(5000);// delays 5 seconds Done uploading. </pre>	<pre>#define PIN_LED_RED 0 //define the red led pin</pre>			
<pre>void setup() {     pinMode(PIN_LED_RED, OUTPUT);     pinMode(PIN_LED_YELLOW, OUTPUT);     pinMode(PIN_LED_GREEN, OUTPUT); } void loop() {     digitalWrite(PIN_LED_RED, HIGH);// turns on the red led     delay(5000);// delays 5 seconds  Done uploading.  Done uploading. </pre>	<pre>#define PIN_LED_YELLOW 2 //define the yellow led pin</pre>			
<pre>pinMode(PIN_LED_RED, OUTPUT); pinMode(PIN_LED_YELLOW, OUTPUT); pinMode(PIN_LED_GREEN, OUTPUT); } void loop() { digitalWrite(PIN_LED_RED, HIGH);// turns on the red led delay(5000);// delays 5 seconds v Done uploading. Done uploading. Wyartu Tiprary found in C:\Users\Administrator\Desktop\Arduino\hardware\espressif\esp32\1: ^</pre>	#define PIN_LED_GREEN 15 //define the green led pin			
<pre>pinMode(PIN_LED_RED, OUTPUT); pinMode(PIN_LED_YELLOW, OUTPUT); pinMode(PIN_LED_GREEN, OUTPUT); } void loop() { digitalWrite(PIN_LED_RED, HIGH);// turns on the red led delay(5000);// delays 5 seconds Done uploading. Wyartu Tiprary found in C:\Users\Administrator\Desktop\Arduino\hardware\espressif\esp32\1; ^</pre>	void setup() {			
<pre>pinMode(PIN_LED_GREEN, OUTPUT); } void loop() {     digitalWrite(PIN_LED_RED, HIGH);// turns on the red led     delay(5000);// delays 5 seconds  Done uploading</pre>				
<pre>} void loop() {     digitalWrite(PIN_LED_RED, HIGH);// turns on the red led     delay(5000);// delays 5 seconds v Done uploading</pre>	<pre>pinMode(PIN_LED_YELLOW, OUTPUT);</pre>			
<pre>void loop() {     digitalWrite(PIN_LED_RED, HIGH);// turns on the red led     delay(5000);// delays 5 seconds  Done uploading</pre>	<pre>pinMode(PIN_LED_GREEN, OUTPUT);</pre>			
<pre>digitalWrite(PIN_LED_RED, HIGH);// turns on the red led delay(5000);// delays 5 seconds Done uploading. </pre>	}			
<pre>digitalWrite(PIN_LED_RED, HIGH);// turns on the red led delay(5000);// delays 5 seconds v Done uploading</pre>				
<pre>delay(5000);// delays 5 seconds v Done uploading. nvarid fibrary found in C:\Users\Administrator\Desktop\Arduino\hardware\espressif\esp32\1;</pre>				
Done uploading. nvaru riprary found in C:\Users\Administrator\Desktop\Arduino\hardware\espressif\esp32\1:				
nvaru rorary found in C:\Users\Administrator\Desktop\Arduino\hardware\espressif\esp32\1				~
nvaru rorary found in C:\Users\Administrator\Desktop\Arduino\hardware\espressif\esp32\1	Done uploading.			
		press	if\esp	32\18
				$\sim$
	< <			>
16 ESP32 Wrover Module, Default, QIO, 80MHz, 921600, None on COM3	16 ESP32 Wrover Module, Default, QIO, 80MHz, 93	21600, N	one on (	сомз

5. Project result

After the project code was uploaded successfully, power up with a USB cable and you'll see are below:

First, the green light will be on for five seconds and then off;

Next, the yellow light blinks three times and then goes off;

Then, the red light goes on for five seconds and then goes off;

Repeat steps 1 to 3 above.

## 5.7 Project 06: RGB LED

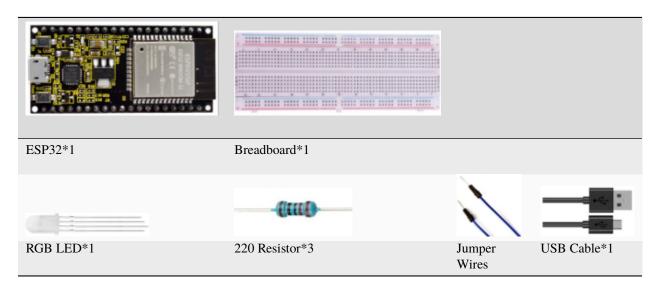
1. Introduction

media/94bdff69e438989d8e0934e57f2e5c00.png

RGB is composed of three colors (red, green and blue), which can emit different colors of light by mixing these three basic colors.

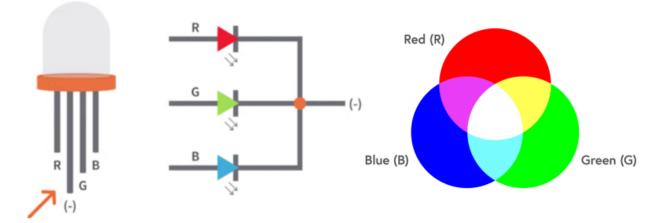
In this project, we will introduce the RGB and show you how to use ESP32 to control the RGB to emit different color light .RGB is pretty basic, but it's also a great way to learn the fundamentals of electronics and coding.

2. Components



#### 3. Component knowledge

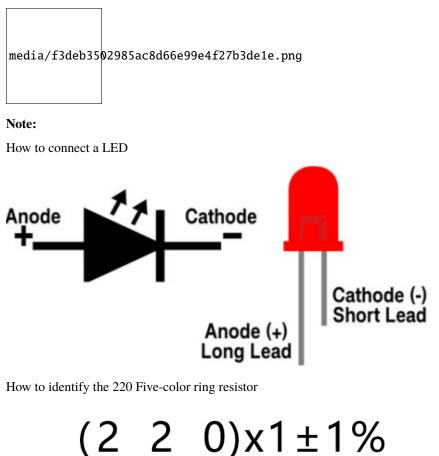
Most monitors adopt the RGB color standard, and all colors on a computer screen are a mixture of red, green and blue in varying proportions.

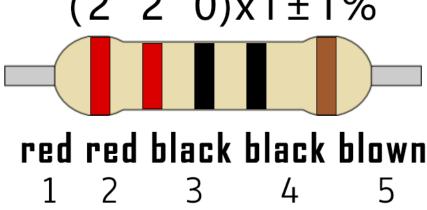


This RGB LED has 4 pins, each color (red, green, blue) and a common cathode, To change its brightness, we can use the PWM of the ESP32 pins, which can give different duty cycle signals to the RGB to produce different colors of light.

If we use three 10-bit PWM to control the RGB, in theory, we can create 2.10\*210\*210=1,073,741,824(1billion) colors through different combinations.

4. Wiring diagram





5. Project code

You can open the code we provideIf you haven't downloaded the code file, please click on the link to download it:Download Arduino Codes

The code used in this project is saved in folder "Arduino-Codes\Project 06RGB LED\Project\_06\_RGB\_LED".

```
(continued from previous page)
const byte chns[] = \{0, 1, 2\};
                                  //define the pwm channels
int red, green, blue;
void setup() {
 for (int i = 0; i < 3; i++) {
                             //setup the pwm channels,1KHz,8bit
   ledcSetup(chns[i], 1000, 8);
   ledcAttachPin(ledPins[i], chns[i]);
 }
}
void loop() {
 red = random(0, 256);
 green = random((0, 256);
 blue = random(\emptyset, 256);
 setColor(red, green, blue);
 delay(200);
}
void setColor(byte r, byte g, byte b) {
 ledcWrite(chns[0], 255 - r); //Common anode LED, low level to turn on the led.
 ledcWrite(chns[1], 255 - g);
 ledcWrite(chns[2], 255 - b);
}
```

### 6. Project result

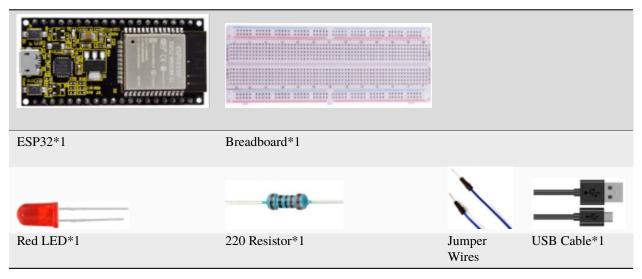
Compile and upload the code to ESP32, after the code is uploaded successfully, power up with a USB cable and you will see that the RGB LED starts to display random colors.

## 5.8 Project 07: Flowing Water Light

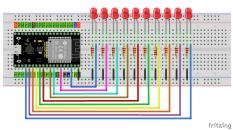
#### 1. Introduction

In our daily life, we can see many billboards composed of different colors of LED. They constantly change the light (like water) to attract customers' attention. In this project, we will use ESP32 to control 10 leds to achieve the effect of flowing water.

2. Components

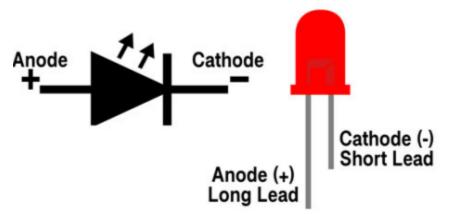


3. Wiring diagram:

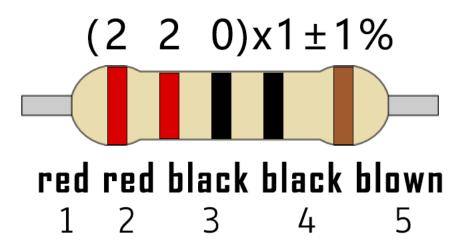


## Note:

How to connect a LED



How to identify the 220 Five-color ring resistor



4. Project code

This project is designed to make a flowing water lamp. Which are these actions: First turn LED #1 ON, then turn it OFF. Then turn LED #2 ON, and then turn it OFF... and repeat the same to all 10 LEDs until the last LED is turns OFF. This process is repeated to achieve the "movements" of flowing water.

You can open the code we provideIf you haven't downloaded the code file, please click on the link to download it:Download Arduino Codes

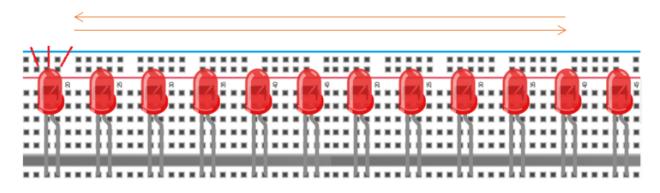
The code used in this project is saved in folder "Arduino-Codes\Project 07Flowing Water Light\Project\_07\_Flowing\_Water\_Light".

```
/*
* Filename
           : Flowing Water Light
* Description : Using ten leds to demonstrate flowing lamp.
* Auther
         : http//www.keyestudio.com
*/
byte ledPins[] = {22, 21, 19, 18, 17, 16, 4, 0, 2, 15};
int ledCounts;
void setup() {
 ledCounts = sizeof(ledPins);
 for (int i = 0; i < ledCounts; i++) {
   pinMode(ledPins[i], OUTPUT);
 }
}
void loop() {
 for (int i = 0; i < ledCounts; i++) {
   digitalWrite(ledPins[i], HIGH);
   delay(100);
   digitalWrite(ledPins[i], LOW);
 }
 for (int i = ledCounts - 1; i > -1; i--) {
   digitalWrite(ledPins[i], HIGH);
   delay(100);
   digitalWrite(ledPins[i], LOW);
 }
```



### 5. Project result

Compile and upload the code to ESP32, after the code is uploaded successfully, power up with a USB cable and you will see that 10 LEDs will light up from left to right and then back from right to left.

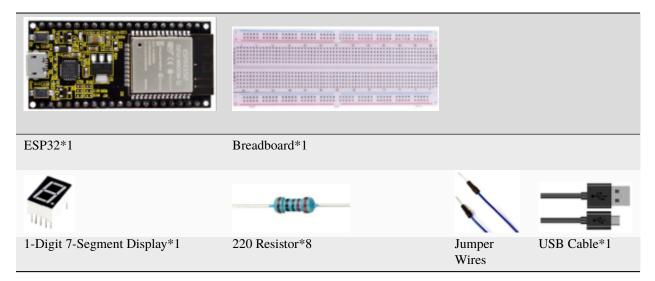


## 5.9 Project 081-Digit Digital Tube

1. Introduction

A 1-Digit 7-Segment Display is an electronic display device that displays decimal numbers. It is widely used in digital clocks, electronic meters, basic calculators and other electronic devices that display digital information. Eventhough they may not look modern enough, they are an alternative to more complex dot matrix displays and are easy to use in limited light conditions and strong sunlight. In this project, we will use ESP32 to control 1-Digit 7-segment display displays numbers.

2. Components



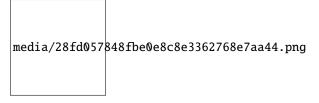
3. Component knowledge

media/e44a0f27beec739ee13e68c04865989f.png

**1-Digit 7-Segment Display principle:** Digital tube display is a semiconductor light emitting device, its basic unit is a light-emitting diode (LED). The digital tube display can be divided into 7-segment display and 8-segment display according to the number of segments. The 8-segment display has one more LED unit than the 7-segment display (used for decimal point display). Each segment of the 7-segment display is a separate LED. According to the connection mode of the LED unit, the digital tube can be divided into a common anode digital tube and a common cathode digital tube.

In the common cathode 7-segment display, all the cathodes (or negative electrodes) of the segmented LEDs are connected together, so you should connect the common cathode to GND. To light up a segmented LED, you can set its associated pin to "HIGH".

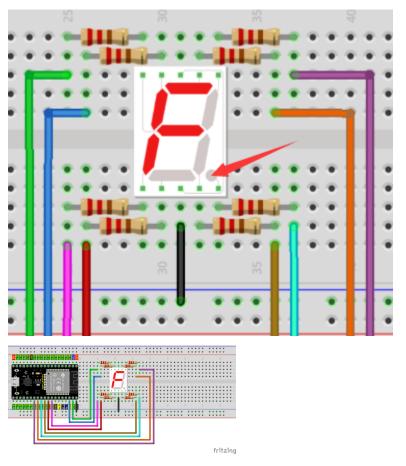
In the common anode 7-segment display, the LED anodes (positive electrodes) of all segments are connected together, so you should connect the common anode to"+5V". To light up a segmented LED, you can set its associated pin to"LOW".



Each part of the digital tube is composed of an LED. So when you use it, you also need to use a current limiting resistor. Otherwise, the LED will be damaged. In this experiment, we use an ordinary common cathode one-digit digital tube. As we mentioned above, you should connect the common cathode to GND. To light up a segmented LED, you can set its associated pin to "HIGH".

4. Wiring diagram

Note: The direction of the 7-segment display inserted into the breadboard is consistent with the wiring diagram, with one more point in the lower right corner.



5. Project code

The digital display is divided into 7 segments, and the decimal point display is divided into 1 segment. When certain numbers are displayed, the corresponding segment will be lit. For example, when the number 1 is displayed, segments b and c will be turned on.

You can open the code we provideIf you haven't downloaded the code file, please click on the link to download it:Download Arduino Codes

The code used in this project is saved in folder "Arduino-Codes\Project 081-Digit Digital Tube\Project\_08\_One\_Digit\_Digital\_Tube.

```
int dp=17; // digital PIN 17 for segment dp
void digital_0(void) // displays number 0
{
digitalWrite(a,HIGH);
digitalWrite(b,HIGH);
digitalWrite(c,HIGH);
digitalWrite(d,HIGH);
digitalWrite(e,HIGH);
digitalWrite(f,HIGH);
digitalWrite(g,LOW);
digitalWrite(dp,LOW);
}
void digital_1(void) // displays number 1
{
digitalWrite(a,LOW);
digitalWrite(b,HIGH);
digitalWrite(c,HIGH);
digitalWrite(d,LOW);
digitalWrite(e,LOW);
digitalWrite(f,LOW);
digitalWrite(g,LOW);
digitalWrite(dp,LOW);
}
void digital_2(void) // displays number 2
{
digitalWrite(a,HIGH);
digitalWrite(b,HIGH);
digitalWrite(c,LOW);
digitalWrite(d,HIGH);
digitalWrite(e,HIGH);
digitalWrite(f,LOW);
digitalWrite(g,HIGH);
digitalWrite(dp,LOW);
}
void digital_3(void) // displays number 3
{
digitalWrite(a,HIGH);
digitalWrite(b,HIGH);
digitalWrite(c,HIGH);
digitalWrite(d,HIGH);
digitalWrite(f,LOW);
digitalWrite(e,LOW);
digitalWrite(dp,LOW);
digitalWrite(g,HIGH);
}
void digital_4(void) // displays number 4
{
digitalWrite(a,LOW);
digitalWrite(b,HIGH);
digitalWrite(c,HIGH);
digitalWrite(d,LOW);
digitalWrite(e,LOW);
```

```
digitalWrite(f,HIGH);
digitalWrite(g,HIGH);
digitalWrite(dp,LOW);
}
void digital_5(void) // displays number 5
{
digitalWrite(a,HIGH);
digitalWrite(b,LOW);
digitalWrite(c,HIGH);
digitalWrite(d,HIGH);
digitalWrite(e,LOW);
digitalWrite(f,HIGH);
digitalWrite(g,HIGH);
digitalWrite(dp,LOW);
}
void digital_6(void) // displays number 6
{
digitalWrite(a,HIGH);
digitalWrite(b,LOW);
digitalWrite(c,HIGH);
digitalWrite(d,HIGH);
digitalWrite(e,HIGH);
digitalWrite(f,HIGH);
digitalWrite(g,HIGH);
digitalWrite(dp,LOW);
}
void digital_7(void) // displays number 7
{
digitalWrite(a,HIGH);
digitalWrite(b,HIGH);
digitalWrite(c,HIGH);
digitalWrite(d,LOW);
digitalWrite(e,LOW);
digitalWrite(f,LOW);
digitalWrite(g,LOW);
digitalWrite(dp,LOW);
}
void digital_8(void) // displays number 8
{
digitalWrite(a,HIGH);
digitalWrite(b,HIGH);
digitalWrite(c,HIGH);
digitalWrite(d,HIGH);
digitalWrite(e,HIGH);
digitalWrite(f,HIGH);
digitalWrite(g,HIGH);
digitalWrite(dp,LOW);
}
void digital_9(void) // displays number 9
ł
digitalWrite(a,HIGH);
digitalWrite(b,HIGH);
```

```
digitalWrite(c,HIGH);
digitalWrite(d,HIGH);
digitalWrite(e,LOW);
digitalWrite(f,HIGH);
digitalWrite(g,HIGH);
digitalWrite(dp,LOW);
}
void setup()
{
 // initialize digital pin LED as an output.
 pinMode(a, OUTPUT);
 pinMode(b, OUTPUT);
 pinMode(c, OUTPUT);
 pinMode(d, OUTPUT);
 pinMode(e, OUTPUT);
 pinMode(f, OUTPUT);
 pinMode(g, OUTPUT);
 pinMode(dp, OUTPUT);
}
void loop()
{
while(1)
{
digital_9();// displays number 9
delay(1000); // waits a sencond
digital_8();// displays number 8
delay(1000); // waits a sencond
digital_7();// displays number 7
delay(1000); // waits a sencond
digital_6();// displays number 6
delay(1000); // waits a sencond
digital_5();// displays number 5
delay(1000); // waits a sencond
digital_4();// displays number 4
delay(1000); // waits a sencond
digital_3();// displays number 3
delay(1000); // waits a sencond
digital_2();// displays number 2
delay(1000); // waits a sencond
digital_1();// displays number 1
delay(1000);// waits a sencond
digital_0();// displays number 0
delay(1000);// waits a sencond
}}
```

### 6. Project result

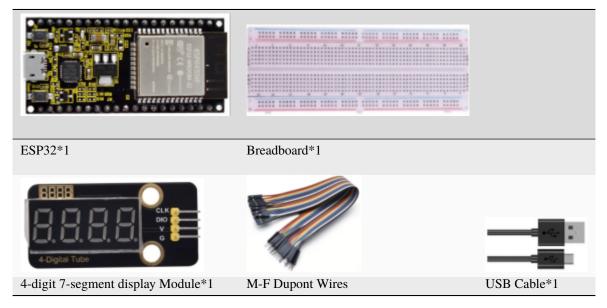
Compile and upload the code to ESP32, after the code is uploaded successfully, power up with a USB cable and you will see that the 1-Digit 7-Segment Display will display numbers from 9 to 0.

## 5.10 Project 094-Digit Digital Tube

### 1. Introduction

A 4-digit 7-segment display is a very practical display device and it is used for devices such as electronic clocks, score counters and the number of people in the park. Because of the low price, easy to use, more and more projects will use 4 Digit 7-segment display. In this project, we use ESP32 control 4-digit 7-segment display to display four digits.

2. Components



3. Component knowledge

**TM1650 4-digit 7-segment display** It is a 12-pin 4-digit 7-segment display module with clock dots. The driver chip is TM1650 which only needs 2 signal lines to enable the microcontroller to control the 4-digit 7-segment display. The control interface level can be 5V or 3.3V.

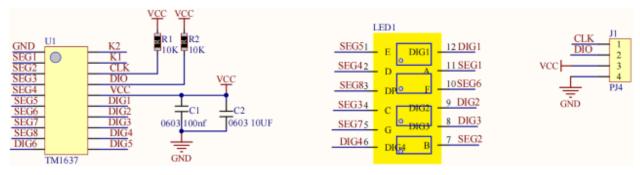
### Specifications of 4-bit 7-segment display module:

Working voltage: DC 3.3V-5V

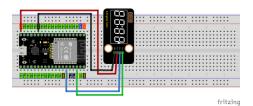
Maximum current: 100MA

Maximum power: 0.5W

### Schematic diagram of 4-digit 7-segment display module:



4. Wiring diagram



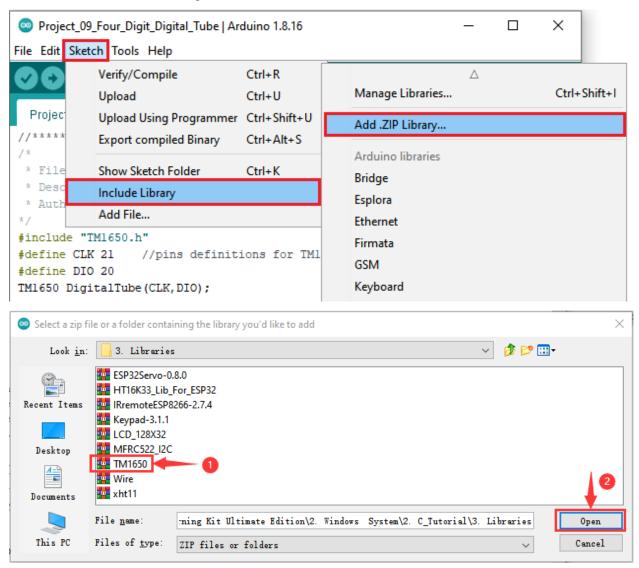
#### 5.Adding the TM1650 library

If you have not downloaded the library file, please click on the link to download it:Download Arduino Libraries

This code uses a library named "TM1650", if you haven't installed it yet, please do so before learning. The steps to add third-party libraries are as follows:

How to install the library

Open the Arduino IDEclick "Sketch"  $\rightarrow$  "Include Library"  $\rightarrow$  "Add .ZIP Library...". In the pop-up window, find the file named \*\*"2. Windows System 2. C\_Tutorial 3. Libraries \*\*TM1650.ZIP" which locates in this directory. Select the TM1650.ZIP file and then click"Open".



6. Project code

After the **TM1650** library is added, You can open the code we provide If you haven't downloaded the code file, please click on the link to download it:Download Arduino Codes

The code used in this project is saved in folder "Arduino-Codes\Project 094-Digit Digital Tube\Project\_09\_Four\_Digit\_Digital\_Tube".

```
/*
* Filename
           : 4-Digit Digital Tube
* Description : Four Digit Tube displays numbers from 1111 to 9999.
* Auther
            : http//www.keyestudio.com
*/
##include "TM1650.h"
##define CLK 22
                //pins definitions for TM1650 and can be changed to other ports
##define DIO 21
TM1650 DigitalTube(CLK,DIO);
void setup(){
 //DigitalTube.setBrightness(); //stes brightness from 0 to 7(default is 2)
 //DigitalTube.displayOnOFF(); // 0= off,1= on(default is 1)
 for(char b=1;b<5;b++){
   DigitalTube.clearBit(b);
                             //which bit to clear
 DigitalTube.displayDot(1,true); // displays the first number
 DigitalTube.displayDot(2,true);
 DigitalTube.displayDot(3,true);
 DigitalTube.displayDot(4,true);
 DigitalTube.displayBit(3,0); //which number to display. bit=1-4, number=0-9
}
void loop(){
 for(int num=0; num<10; num++){</pre>
   DigitalTube.displayBit(1,num);
   DigitalTube.displayBit(2,num);
   DigitalTube.displayBit(3,num);
   DigitalTube.displayBit(4,num);
   delay(1000);
 }
}
```

#### 7. Project result

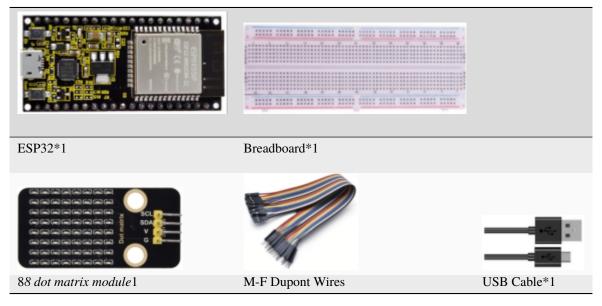
Compile and upload the code to ESP32, after the code is uploaded successfully, power up with a USB cable and you will see that 4-digit 7-segment display four digits and repeat these actions in an infinite loop.

## 5.11 Project 108×8 Dot-matrix Display

1. Introduction

Dot matrix display is an electronic digital display device that can display information on machine, clocks, public transport departure indicators and many other devices. In this project, we will use ESP32 control 8x8 LED dot matrix to display patterns.

2. Components



3. Component knowledge

**8\*8 dot matrix module** The 8\*8 dot matrix is composed of 64 LEDs, and each LED is placed at the intersection of a row and a column. When using the single chip microcomputer to drive an 8\*8 dot matrix, we need 16 digital ports in total, which greatly wastes the data of the single chip microcomputer. To this end, we specially designed this module, using the HT16K33 chip to drive an 8\*8 dot matrix, and only need to use the I2C communication port of the MCU to control the 8\*8 dot matrix, which greatly saving the MCU resources.

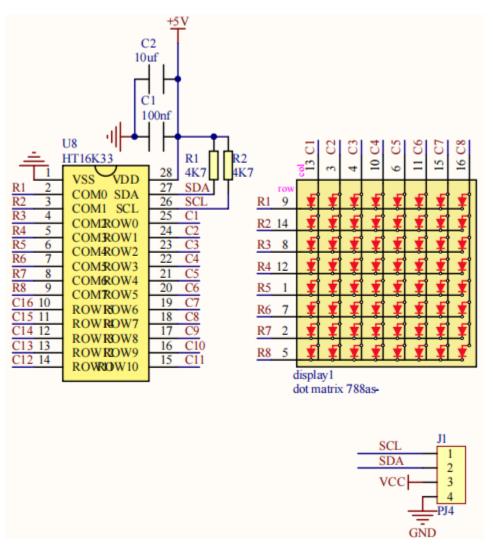
### Specifications of 8\*8 dot matrix module

Working voltage: DC 5V

Current: 200MA

Maximum power: 1W

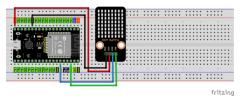
Schematic diagram of 8\*8 dot matrix module



Some modules have three DIP switches that you can toggle at will. These switches are used to set the I2C communication address, the setting method is as follows. The module has fixed the communication address. A0, A1 and A2 are connected to GND, and the address is 0x70.

A0 (1)	A1 (2)	A2 (3)	A0 (1)	A1 (2)	A2 (3)	A0 (1)	A1 (2)	A2 (3)	
0 (OFF)	0(OFF)	0 (OFF)	1 (ON)	0 (OFF)	0 (OFF)	0 (OFF)	1 (ON)	0 (OFF)	
0X70	0X70			0X71			0X72		
A0 (1)	A1 (2)	A2 (3)	A0 (1)	A1 (2)	A2 (3)	A0 (1)	A1 (2)	A2 (3)	
1 (ON)	1 (ON)	0 (OFF)	0(OFF)	0 (OFF)	1 (ON)	1 (ON)	0 (OFF)	1 (ON)	
0X73		0X74			0X75				
A0 (1)	A1 (2)	A2 (3)	A0 (1)	A1 (2)	A2 (3)				
0 (OFF)	1 (ON)	1 (ON)	1 (ON)	1 (ON)	1 (ON)				
0X76	•		0X77						

4. Wiring diagram



5. Adding the HT16K33\_Lib\_For\_ESP32 library

If you have not downloaded the library file, please click on the link to download it:Download Arduino Libraries

This code uses a library named "HT16K33\_Lib\_For\_ESP32", if you haven't installed it yet, please do so before learning.

How to install the library

Open the Arduino IDEclick"Sketch" $\rightarrow$ "Include Library" $\rightarrow$ "Add .ZIP Library...". In the pop-up window, find the file named\*\*"2. Windows System\2. C\_Tutorial\3.Libraries\\*\***HT16K33\_Lib\_For\_ESP32.ZIP**" which locates in this directory. Select the **HT16K33\_Lib\_For\_ESP32.ZIP** file and then click"Open".

	_8_8_Dot_Matrix_Displ	ay   Arduino 1.8.16		_		×
Project	ch Tools Help Verify/Compile Upload Upload Using Progra	Ctrl+R Ctrl+U mmer Ctrl+Shift+U	△ Manage Libraries Add .ZIP Library	Ctrl+Shift+I		₽ ▼
//****	Export compiled Bin	ary Ctrl+Alt+S	Arduino libraries		***	^
* File * Desc	Show Sketch Folder	Ctrl+K	Bridge			
* Auth	Include Library	;	Esplora			
*/ #include "H	Add File TI6K33_Lib_For_E	SP32.h"	Ethernet			
#define SDA			Firmata GSM			
#define SCI			LiquidCrystal			
💿 Select a zip f	ile or a folder containing t	he library you'd like to add	ł			×
Look <u>i</u> n:	3. Libraries			<ul> <li>Image: state of the state of th</li></ul>		
Recent Items	ESP32Servo-0.8.0 HT16K33_Lib_For_ES IRremoteESP8266-2. Keypad-3.1.1 LCD_128X32 MFRC522_I2C TM1650					
Documents	Wire xht11					2
	File <u>n</u> ame: :::::::::::::::::::::::::::::::::::	g Kit Ultimate Edition\2	2. Windows System\2. C_Tutori	al\3. Libraries	0	pen
This PC	Files of type: ZIP	files or folders		~	Ca	ncel

6. Project code

After the **HT16K33\_Lib\_For\_ESP32** library is added, You can open the code we provideIf you haven't downloaded the code file, please click on the link to download it:Download Arduino Codes

The code used in this project is saved in folder\*\*"Arduino-Codes\Project 108×8 Dot-matrix Display\Project\_10\_8×8\_Dot\_Matrix\_Display"\*\*.

```
//*
* Filename : 8×8 Dot-matrix Display
* Description : 8x8 LED dot matrix display"Heart" pattern.
* Auther : http//www.keyestudio.com
*/
##include "HT16K33_Lib_For_ESP32.h"
##define SDA 21
##define SCL 22
ESP32_HT16K33 matrix = ESP32_HT16K33();
```

```
//The brightness values can be set from 1 to 15, with 1 darkest and 15 brightest
##define A 15
byte result[8][8];
byte test1[8] = {0x00,0x42,0x41,0x09,0x09,0x41,0x42,0x00};
void setup()
{
 matrix.init(0x70, SDA, SCL);//Initialize matrix
 matrix.showLedMatrix(test1,0,0);
 matrix.show();
}
void loop()
{
 for (int i = 0; i <= 7; i++)
 ł
   matrix.setBrightness(i);
   delay(100);
 }
 for (int i = 7; i > 0; i--)
 {
   matrix.setBrightness(i);
   delay(100);
 }
}
```

7. Project result

Compile and upload the code to ESP32, after the code is uploaded successfully, power up with a USB cable and you will see that the 8\*8 dot matrix display"Smiling face"pattern.

# 5.12 Project 1174HC595N Control 8 LEDs

1. Introduction

In previous projects, we learned how to light up an LED.

With only 32 IO ports on ESP32, how do we light up a lot of leds? Sometimes it is possible to run out of pins on the ESP32, and you need to extend it with the shift register. You can use the 74HC595N chip to control 8 outputs at a time, taking up only a few pins on your microcontroller. In addition, you can also connect multiple registers together to further expand the output. In this project, we will use ESP3274HC595 chip and LED to make a flowing water light to understand the function of the 74HC595 chip.

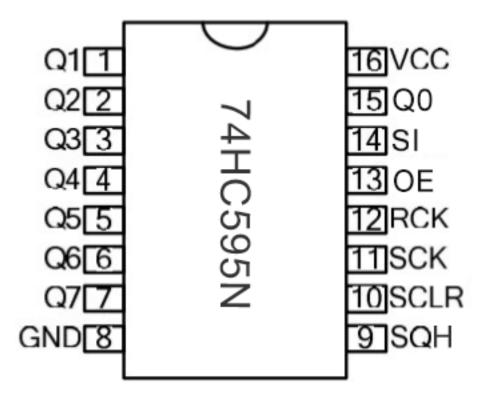
2. Components

			![img](file:///C:\Users\ADMINI~1\AppData\Local\Tem	ıp∖ksohtml16172
ESP32*1	Breadboard*1	74HC595N chip*1	Jumper Wires	
-(1111)-				
220Resistor*8	Red LED*8	USB Ca- ble*1		

3. Component knowledge



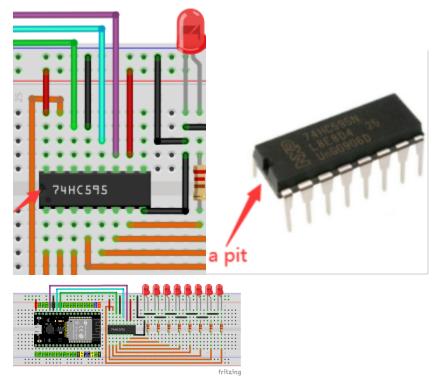
**74HC595N Chip:** The 74HC595 chip is used to convert serial data into parallel data. A 74HC595 chip can convert the serial data of one byte into 8 bits, and send its corresponding level to each of the 8 ports correspondingly. With this characteristic, the 74HC595 chip can be used to expand the IO ports of an ESP32. At least 3 ports are required to control the 8 ports of the 74HC595 chip.



The ports of the 74HC595 chip are described as follows

4. Wiring diagram

Note: Note the orientation in which the 74HC595N chip is inserted.



5. Project code

You can open the code we provideIf you haven't downloaded the code file, please click on the link to download it:Download Arduino Codes

The code used in this project is saved in folder "Arduino-Codes\Project 1174HC595N Control 8 LEDs\Project\_11\_74HC595N\_Control\_8\_LEDs".

```
/*
* Filename
              : 74HC595N Control 8 LEDs
* Description : Use 74HC575N to drive ten leds to display the flowing light.
* Auther
             : http//www.keyestudio.com
*/
int dataPin = 14; // Pin connected to DS of 74HC595(Pin14)
int latchPin = 12; // Pin connected to ST_CP of 74HC595(Pin12)
int clockPin = 13; // Pin connected to SH_CP of 74HC595(Pin11)
void setup() {
  // set pins to output
 pinMode(latchPin, OUTPUT);
 pinMode(clockPin, OUTPUT);
 pinMode(dataPin, OUTPUT);
}
void loop() {
  // Define a one-byte variable to use the 8 bits to represent the state of 8 LEDs of.
\rightarrow LED bar graph.
 // This variable is assigned to 0x01, that is binary 00000001, which indicates only.
\rightarrow one LED light on.
                 // 0b 0000 0001
 byte x = 0 \times 01;
 for (int j = 0; j < 8; j++) { // Let led light up from right to left
   writeTo595(LSBFIRST, x);
   \mathbf{x} \ll 1; // make the variable move one bit to left once, then the bright LED move.
\rightarrow one step to the left once.
   delay(50);
  }
 delay(100);
  x = 0x80;
                 //0b 1000 0000
  for (int j = 0; j < 8; j++) { // Let led light up from left to right
   writeTo595(LSBFIRST, x);
   x >>= 1;
   delay(50);
  }
  delay(100);
}
void writeTo595(int order, byte _data ) {
  // Output low level to latchPin
 digitalWrite(latchPin, LOW);
  // Send serial data to 74HC595
  shiftOut(dataPin, clockPin, order, _data);
 // Output high level to latchPin, and 74HC595 will update the data to the parallel.
\rightarrowoutput port.
  digitalWrite(latchPin, HIGH);
}
```

6. Project result

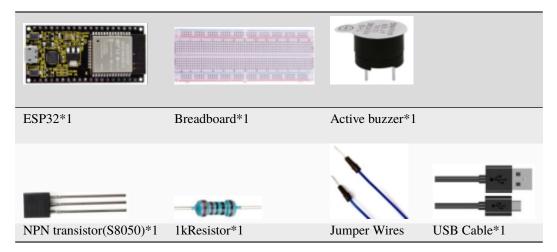
Compile and upload the code to ESP32, after the code is uploaded successfully, power up with a USB cable and you will see that the 8 LEDs start flashing in flowing water mode.

## 5.13 Project 12Active Buzzer

#### 1. Introduction

Active buzzer is a sound component that is widely used as a sound component for computersprintersalarmselectronic toys and phonestimers etc. It has an internal vibration source, just by connecting to a 5V power supply, it can continuously buzz. In this project, we will use ESP32 to control the active buzzer to beep.

2. Components



3. Component knowledge



Active buzzer: Active buzzer inside has a simple oscillator circuit, which can convert constant direct current into a certain frequency pulse signal. Once active buzzer receives a high level, it will produce sound. Passive buzzer is an internal without vibration source integrated electronic buzzer, it must be driven by 2k to 5k square wave, rather than a DC signal. The two buzzers are very similar in appearance, but one buzzer with a green circuit board is a passive buzzer, while the other buzzer with black tape is an active buzzer. Passive buzzers don't have positive polarity, but active buzzers have. As shown below:

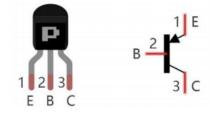


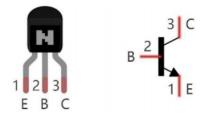
**Transistor:** 



Because the buzzer requires such large current that GPIO of ESP32 output capability cannot meet the requirement, a transistor of NPN type is needed here to amplify the current.

Transistor, the full name: semiconductor transistor, is a semiconductor device that controls current. Transistorcan be used to amplify weak signal, or works as a switch. It has three electrodes(PINs): base (b), collector © and emitter (e). When there is current passing between "be", "ce" will allow several-fold current (transistor magnification) pass, at this point, transistor works in the amplifying area. When current between "be" exceeds a certain value, "ce" will not allow current to increase any longer, at this point, transistor works in the saturation area. Transistor has two types as shown below: PNP and NPN,





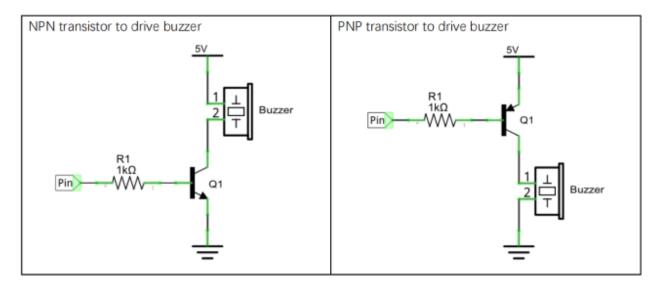
PNP transistor NPN transistor

In our kit, the PNP transistor is marked with 8550, and the NPN transistor is marked with 8050.

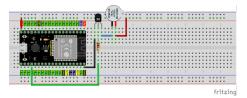
Based on the transistor's characteristics, it is often used as a switch in digital circuits. As micro-controller's capacity to output current is very weak, we will use transistor to amplify current and drive large-current components.

When using NPN transistor to drive buzzer, we often adopt the following method. If GPIO outputs high level, current will flow through R1, the transistor will get conducted, and the buzzer will sound. If GPIO outputs low level, no current flows through R1, the transistor will not be conducted, and buzzer will not sound.

When using PNP transistor to drive buzzer, we often adopt the following method. If GPIO outputs low level, current will flow through R1, the transistor will get conducted, and the buzzer will sound. If GPIO outputs high level, no current flows through R1, the transistor will not be conducted, and buzzer will not sound.



4. Wiring diagram



Note: The buzzer power supply in this circuit is 5V. On a 3.3V power supply, the buzzer can work, but will reduce the loudness.

5. Project code

You can open the code we provideIf you haven't downloaded the code file, please click on the link to download it:Download Arduino Codes

The code used in this project is saved in folder "Arduino-Codes\Project 12Active Buzzer\Project\_12\_Active\_Buzzer".

```
/*
* Filename : Active Buzzer
* Description : Active buzzer beeps.
* Auther
         : http//www.keyestudio.com
*/
##define buzzerPin 15 //define buzzer pins
void setup ()
{
 pinMode (buzzerPin, OUTPUT);
}
void loop ()
{
 digitalWrite (buzzerPin, HIGH);
 delay (500);
 digitalWrite (buzzerPin, LOW);
 delay (500);
```

(continued from previous page)



# 6. Project result

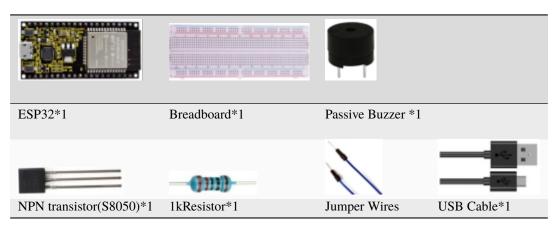
Compile and upload the code to ESP32, after the code is uploaded successfully, power up with a USB cable and you will see that the active buzzer beeps.

# 5.14 Project 13Passive Buzzer

1. Introduction:

In a previous project, we studied an active buzzer, which can only make a sound and may make you feel very monotonous. In this project, we will learn a passive buzzer and use the ESP32 control it to work. Unlike the active buzzer, the passive buzzer can emit sounds of different frequencies.

2. Components



3. Component knowledge

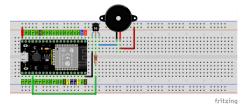


**Passive buzzer:** A passive buzzer is an integrated electronic buzzer with no internal vibration source and it has to be driven by 2K-5K square waves, not DC signals. The two buzzers are very similar in appearance, but one buzzer with a green circuit board is a passive buzzer and the other buzzer with black tape is an active buzzer. Passive buzzers cannot distinguish between positive polarity while active buzzers can.



Transistor: Please refer to Project 12.

4. Wiring diagram:



5. Project code

You can open the code we provideIf you haven't downloaded the code file, please click on the link to download it:Download Arduino Codes

The code used in this project is saved in folder "Arduino-Codes\Project 13Passive

Buzzer\Project\_13\_Passive\_Buzzer".

```
/*
* Filename
           : Passive Buzzer
* Description : Passive Buzzer sounds the alarm.
* Auther
            : http//www.keyestudio.com
*/
##define LEDC_CHANNEL_0 0
// LEDC timer uses 13 bit accuracy
##define LEDC_TIMER_13_BIT 13
// Define tool I/O ports
##define BUZZER PIN 15
//Create a musical melody list, Super Mario
int melody[] = {330, 330, 330, 262, 330, 392, 196, 262, 196, 165, 220, 247, 233, 220,
→196, 330, 392, 440, 349, 392, 330, 262, 294, 247, 262, 196, 165, 220, 247, 233, 220, L
→196, 330, 392,440, 349, 392, 330, 262, 294, 247, 392, 370, 330, 311, 330, 208, 220,
\rightarrow 262, 220, 262,
294, 392, 370, 330, 311, 330, 523, 523, 523, 392, 370, 330, 311, 330, 208, 220, 262,220,
→262, 294, 311, 294, 262, 262, 262, 262, 262, 294, 330, 262, 220, 196, 262, 262,262, .
→262, 294, 330, 262, 262, 262, 262, 294, 330, 262, 220, 196];
//Create a list of tone durations
\leftrightarrow4,8,4,8,4,8,2,8,4,4,8,4,1,8,4,4,8,4,8,4,8,2};
void setup() {
pinMode(BUZZER_PIN, OUTPUT); // Set the buzzer to output mode
}
void loop() {
 int noteDuration; //Create a variable of noteDuration
 for (int i = 0; i < sizeof(noteDurations); ++i)</pre>
 {
     noteDuration = 800/noteDurations[i];
     ledcSetup(LEDC_CHANNEL_0, melody[i]*2, LEDC_TIMER_13_BIT);
     ledcAttachPin(BUZZER_PIN, LEDC_CHANNEL_0);
     ledcWrite(LEDC_CHANNEL_0, 50);
```

(continues on next page)

(continued from previous page)

6. Project result

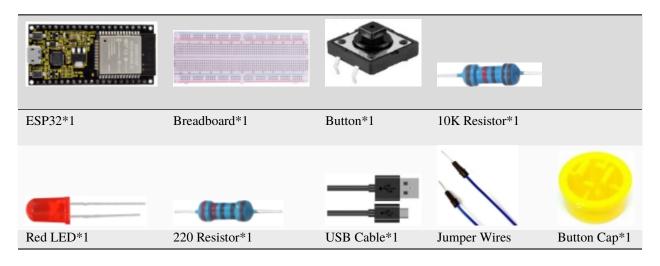
Compile and upload the code to ESP32, after the code is uploaded successfully, power up with a USB cable and you will see that the passive buzzer plays music.

# 5.15 Project 14: Mini Table Lamp

1. Introduction

Do you know that the ESP32 can light up an LED when you press a button? In this project, we will use ESP32a button switch and an LED to make a mini table lamp.

2. Components



# 3.Component knowledge



**Button:** A button can control the circuit on and off, the button is plugged into a circuit, the circuit is disconnected when the button is not pressed. The circuit works when you press the button, but breaks again when you release it. Why does it only work when you press it? It starts from the internal structure of the button, which don't allow current to travel from one end of the button to the other before it is pressed; When pressed, a metal strip inside the button connects the two sides to allow electricity to pass through.

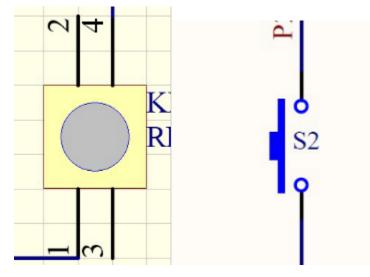


The internal structure of the button is shown in the figure

Before the button is pressed, 1 and 2 are on, 3 and 4 are also on, but 1, 3 or 1, 4 or 2, 3 or 2, 4 are off(not working). Only when the button is pressed, 1, 3 or 1, 4 or 2, 3 or 2, 4 are on.

The button switch is one of the most commonly used components in circuit design.

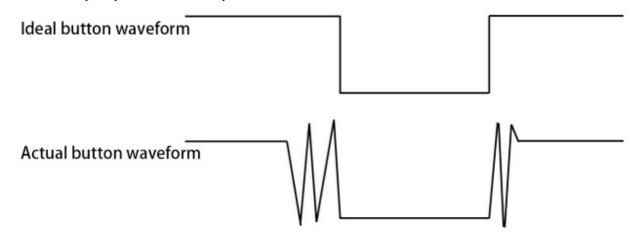
#### Schematic diagram of the button:



#### What is button [shake](javascript:;)?

We think of the switch circuit as "press the button and turn it on immediately", "press it again and turn it off immediately". In fact, this is not the case.

The button usually uses a mechanical elastic switch, and the mechanical elastic switch will produce a series of [shake](javascript:;) due to the elastic action at the moment when the mechanical contact is opened and closed (usually about 10ms). As a result, the button switch will not immediately and stably turn on the circuit when it is closed, and it will not be completely and instantaneously disconnected when it is turned off.

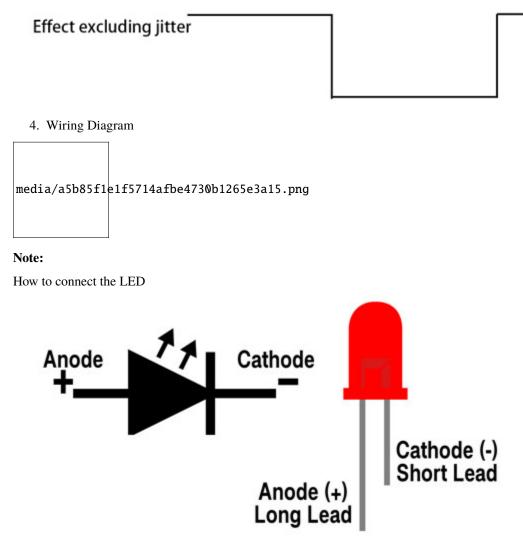


#### How to eliminate the [shake](javascript:;)?

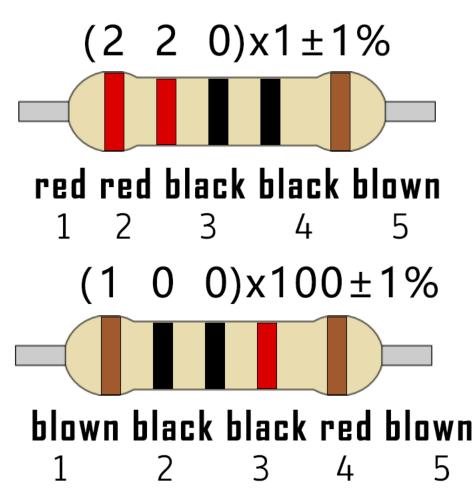
There are two common methods, namely fix [shake](javascript:;) in the software and hardware. We only discuss the [shake](javascript:;) removal in the software.

We already know that the [shake](javascript:;) time generated by elasticity is about 10ms, and the delay command can be used to delay the execution time of the command to achieve the effect of [shake](javascript:;) removal.

Therefore, we delay 0.02s in the code to achieve the key anti-shake function.



How to identify the 220 5-band resistor and 10K 5-band resistor



5. Project code

You can open the code we provideIf you haven't downloaded the code file, please click on the link to download it:Download Arduino Codes

The code used in this project is saved in folder "Arduino-Codes\Project 14Mini Table Lamp\Project\_14\_Mini\_Table\_Lamp".

(continues on next page)

(continued from previous page)

### 6. Project result

Compile and upload the code to ESP32, after the code is uploaded successfully, power up with a USB cable and you will see that press the push button switch, the LED turns on; When it is released, the LED is still on. Press it again, and the LED turns off. When it is released, the LED stays off. Doesn't it look like a mini table lamp?

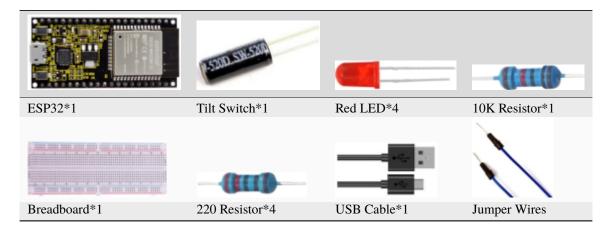
# 5.16 Project 15Tilt and LED

#### 1. Introduction

The ancients without electronic clock, so the hourglass are invented to measure time. The hourglass has a large capacity on both sides, and which is filled with fine sand on one side. What's more, there is a small channel in the middle, which can make the hourglass stand upright, the side with fine sand is on the top. due to the effect of gravity, the fine sand will flow down through the channel to the other side of the hourglass. When the sand reaches the bottom, turn it upside down and record the number of times it has gone through the hourglass, therefore, the next day we can know the approximate time of the day by it.

In this project, we will use ESP32 to control the tilt switch and LED lights to simulate an hourglass and make an electronic hourglass.

2. Components



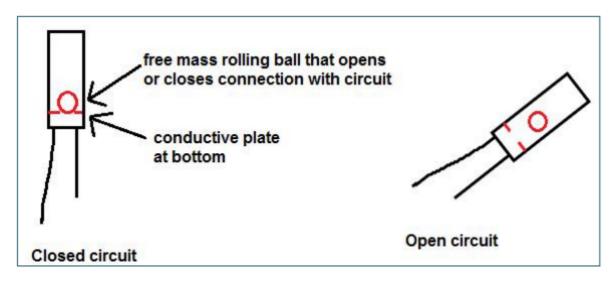
3. Component knowledge



Tilt switch is also called digital switch. Inside is a metal ball that can roll. The principle of rolling the metal ball to contact with the conductive plate at the bottom, which is used to control the on and off of the circuit. When it is a rolling ball tilt sensing switch with single directional trigger, the tilt sensor is tilted toward the trigger end (two gold-plated pin ends), the tilt switch is in a closed circuit and the voltage at the analog port is about 5V(binary number is 1023),

In this way, the LED will light up. When the tilting switch is in horizontal position or tilting to the other end, the tilting switch is in open state the voltage of the analog port is about 0V (binary number is 0), the LED will turn off. In the program, we judge the state of the switch based on whether the voltage value of the analog port is greater than 2.5V (binary number is 512).

The internal structure of the tilt switch is used here to illustrate how it works, as shown below:

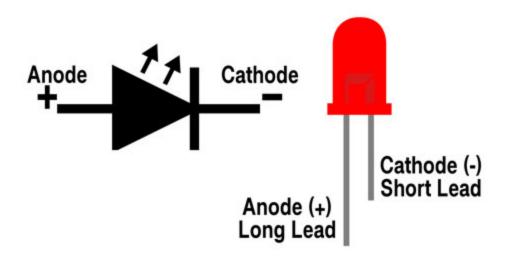


4. Wiring Diagram

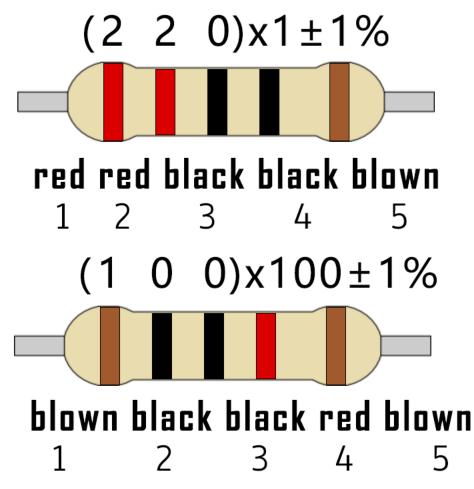


# Note:

How to connect the LED



How to identify the 220 5-band resistor and 10K 5-band resistor



5. Project code

You can open the code we provideIf you haven't downloaded the code file, please click on the link to download it:Download Arduino Codes

The code used in this project is saved in folder "Arduino-Codes\Project 15Tilt And LED\Project\_15\_Tilt\_And\_LED".

```
/*
* Filename : Tilt And LED
* Description : Tilt switches and four leds to simulate an hourglass.
          : http//www.keyestudio.com
* Auther
*/
##define SWITCH_PIN 15 // the tilt switch is connected to Pin15
byte switch_state = \emptyset;
void setup()
{
    for(int i=16;i<20;i++)</pre>
 {
      pinMode(i, OUTPUT);
 }
   pinMode(SWITCH_PIN, INPUT);
for(int i=16;i<20;i++)</pre>
 {
   digitalWrite(i, ♥);
 }
 Serial.begin(9600);
}
void loop()
{
switch_state = digitalRead(SWITCH_PIN);
Serial.println(switch_state);
if (switch_state == 0)
{
for(int i=16;i<20;i++)</pre>
 {
   digitalWrite(i,1);
   delay(500);
 }
 }
  if (switch_state == 1)
{
  for(int i=19;i>15;i--)
  {
   digitalWrite(i,0);
   delay(500);
  }
 }
}
```

# 6. Project result

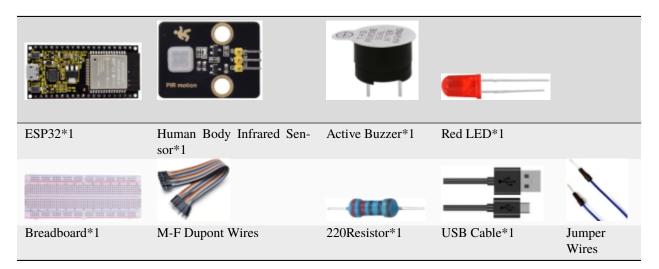
Compile and upload the code to ESP32, after the code is uploaded successfully, power up with a USB cable and you will see that when you tilt the breadboard to an angle, the LEDs will light up one by one. When you turn the breadboard to the original angle, the LEDs will turn off one by one. Like the hourglass, the sand will leak out over time.

# 5.17 Project 16Burglar Alarm

# 1. Introduction

The human body infrared sensor measures the thermal infrared (IR) light emitted by moving objects. The sensor can detect the movement of peopleanimals and cars to trigger safety alarms and lighting. They are used to detect movement and ideal for security such as burglar alarms and security lighting systems. In this project, we will use the ESP32 control human body infrared sensorbuzzer and LED to simulate burglar alarm.

2. Components



3. Component knowledge



**Human Body Infrared Sensor :** Its principle is that when some crystals, such as lithium tantalate and triglyceride sulfate are heated, the two ends of the crystal will generate an equal number of charges with opposite signs. These charges can be converted into voltage output by an amplifier. Due to the human body will release infrared light, although relatively weak, can still be detected. When the Human Body Infrared Sensor detects the movement of a nearby person, the sensor signal terminal outputs a high level 1, otherwise, it outputs low level 0.

Special attention should be paid to the fact that this sensor can detect peopleanimals and cars in motion, which cannot be detected in static, and the maximum detection distance is about 7 meters.

**Note:** Since vulnerable to radio frequency radiation and temperature changes, the PIR motion sensor should be kept away from heat sources like radiators, heaters and air conditioners, as well as direct irradiation of sunlight, headlights and incandescent light.

# Features:

Maximum input voltage: DC 3.3 ~ 5V Maximum operating current: 50MA Maximum power: 0.3W Operating temperature: -20 ~ 85°C Output high level is 3V, low level is 0V.

Delay time: about 2.3 to 3 seconds

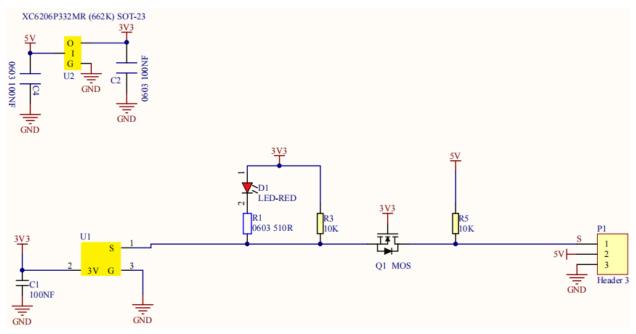
Detection Angle: about 100 degrees

Maximum detection distance: about 7 meters

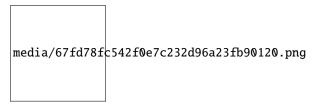
Indicator light output (when the output is high, it will light up)

Pin limiting current: 50MA

### Schematic diagram:



4. Wiring Diagram



5. Project code

You can open the code we provideIf you haven't downloaded the code file, please click on the link to download it:Download Arduino Codes

The code used in this project is saved in folder "Arduino-Codes\Project 16Burglar Alarm\Project\_16\_Burglar\_Alarm".

(continues on next page)

```
(continued from previous page)
```

```
##define ledPin
                      // the pin of the PIR motion sensor
                0
##define pirPin
                15
                       // the pin of the PIR motion sensor
byte pirStat = 0;
                  // the state of the PIR motion sensor
void setup() {
pinMode(buzzerPin, OUTPUT);
pinMode(ledPin, OUTPUT);
pinMode(pirPin, INPUT);
}
void loop()
{
pirStat = digitalRead(pirPin);
if (pirStat == HIGH)
            // if people or moving animals are detected
{
  digitalWrite(buzzerPin, HIGH); // the buzzer buzzes
  digitalWrite(ledPin, HIGH); // the led turn on
  delay(500);
  digitalWrite(buzzerPin, LOW); // the buzzer doesn't sound
  digitalWrite(ledPin, LOW); // the led turn off
  delay(500);
}
else {
  digitalWrite(buzzerPin, LOW); // if people or moving animals are not detected, turn_
\rightarrow off buzzers
  digitalWrite(ledPin, LOW); // the led turn off
}
}
```

# 6. Project result

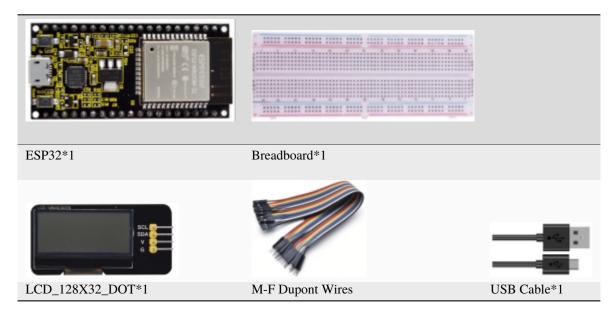
Compile and upload the code to ESP32, after the code is uploaded successfully, power up with a USB cable and you will see that if the human body infrared sensor detects someone moving nearby, the buzzer will continuously issue an alarm and the LED will continuously flash.

# 5.18 Project 17 I2C 128×32 LCD

1. Introduction

In everyday life, we can do all kinds of experiments with the display module and also DIY a variety of small objects. For example, you can make a temperature meter with a temperature sensor and display, or make a distance meter with an ultrasonic module and display. In this project, we will use the LCD\_128X32\_DOT module as the display and connect it to the ESP32, which will be used to control the LCD\_128X32\_DOT display to display various English words, common symbols and numbers.

2. Components

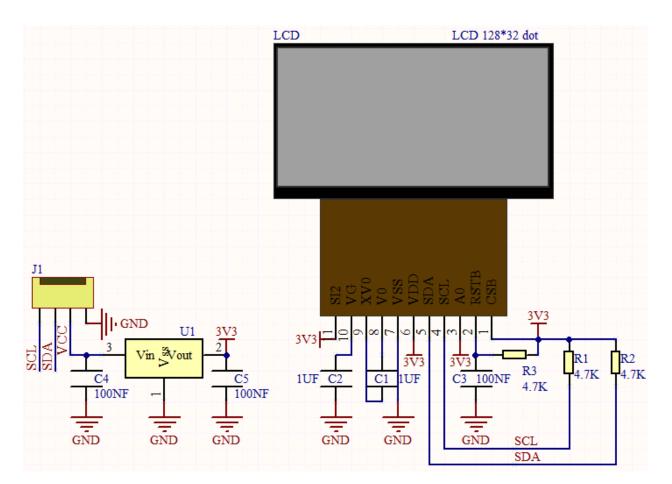


# 3.Component knowledge



**LCD\_128X32\_DOT:** It is an LCD module with 128\*32 pixels and its driver chip is ST7567A. The module uses the IIC communication mode, while the code contains a library of all alphabets and common symbols that can be called directly. When using, we can also set it in the code so that the English letters and symbols show different text sizes. To make it easy to set up the pattern display, we also provide a mold capture software that converts a specific pattern into control code and then copies it directly into the test code for use.

# Schematic diagram of LCD\_128X32\_DOT



# Features:

Pixel: 128\*32 character

Operating voltage(chip)4.5V to 5.5V

Operating current100mA (5.0V)

Optimal operating voltage(module):5.0V

4. Wiring Diagram

media/072d954dac310add077688398ad59af2.png

5. Adding the lcd128\_32\_io library

If you have not downloaded the library file, please click on the link to download it:Download Arduino Libraries

This code uses a library named "**lcd128\_32\_io**", if you haven't installed it yet, please do so before learning. The steps to add third-party libraries are as follows:

Open the Arduino IDEclick "Sketch"  $\rightarrow$  "Include Library"  $\rightarrow$  "Add .ZIP Library...". In the pop-up window, find the file named "2. Windows System 2. C\_Tutorial 3. Libraries LCD\_128X32.ZIP" which locates in this directory. Select the LCD\_128X32.ZIP file and then click "Open".

Project_17_12C_128_32_LCD   /	Arduino 1.8.16		_		×
File Edit Sketch Tools Help Verify/Compile	Ctrl+R				ø
Project Upload Upload Upload	Ctrl+U grammer Ctrl+Shift+U				
//****** /* * File Show Sketch Fold	-	∆ Manage Libraries	Ctrl+Shift+I	***	^
* File Show Sketch Fold * Desc Include Library	er Ctrl+K	Add .ZIP Library			
*/ Add File #include "lcdl28_32_io.h"		Arduino libraries Bridge			
<pre>//Create 1CD128 *32 pin, s lcd lcd(21, 22);</pre>	sda>21, scl>2	Esplora Ethernet			
<pre>void setup() {     lcd.Init(); //initializ</pre>		Firmata GSM			
<pre>lcd.Clear(); //clear }</pre>	-	LiquidCrystal Mouse			
<pre>void loop() {     lod Company(0, 4); ((Set))</pre>		Robot IR Remote			
<pre>lcd.Cursor(0, 4); //Set lcd.Display("KEYESTUDIO</pre>		Robot Motor			~
Select a zip file or a folder containin Look in: 3. Libraries	ng the library you'd like to add		🗸 🦸 📂 🖽	•	×
Look in. S. Libraries ESP32Servo-0.8.0 ESP32Servo-0.8.0 HT16K33_Lib_For IRremoteESP826 Keypad-3.1.1 LCD_128X32 Desktop Documents MFRC522_I2C Wire Wire Wire	r_ESP32				2
	ning Kit Ultimate Edition\2 IP files or folders	. Windows System\2. C_Tutorial\	3. Libraries		pen ncel

6. Project code

After the **lcd128\_32\_io** library was added, You can open the code we provideIf you haven't downloaded the code file, please click on the link to download it:Download Arduino Codes

The code used in this project is saved in folder "Arduino-Codes\Project 17 I2C 128×32 LCD2. C\_Tutorial\2. Projects\Project\_17\_I2C\_128\_32\_LCD".

(continues on next page)

(continued from previous page)

```
* Auther
             : http//www.keyestudio.com
*/
##include "lcd128_32_io.h"
//Create 1CD128 *32 pinsda--->21 scl--->22
lcd lcd(21, 22);
void setup() {
 lcd.Init(); //initialize
 lcd.Clear(); //clear
}
void loop() {
 lcd.Cursor(0, 4); //Set display position
 lcd.Display("KEYESTUDIO"); //Setting the display
 lcd.Cursor(1, 0);
 lcd.Display("ABCDEFGHIJKLMNOPQR");
 lcd.Cursor(2, 0);
 lcd.Display("123456789+-*/<>=$@");
 lcd.Cursor(3, 0);
 lcd.Display("%^&(){}:;'|?,.~\\[]");
}
```

7. Project result

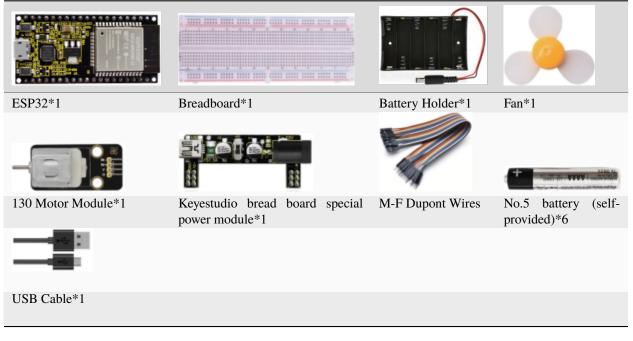
Compile and upload the code to ESP32, after the code is uploaded successfully, power up with a USB cable and you will see that the 128X32LCD module display will show "KEYESTUDIO" at the first line "ABCDEFGHIJKLMNOPQR" will be displayed at the second line "123456789 $\pm$ \*/<>=\$@" will be shown at the third line and "%^&(){}:;'?,.-\[]" will be displayed at the fourth line.

# 5.19 Project 18Small Fan

1. Introduction

In hot summer, we need electric fans to cool us down, so in this project, we will use ESP32 control 130 motor module and small fan blade to make a small electric fan.

2. Components

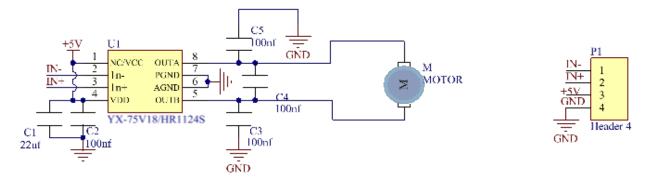


3. Component knowledge :



**130 motor module:** The motor control module uses the HR1124S motor control chip. which is a single-channel Hbridge driver chip for DC motor. The H-bridge driver part of the HR1124S uses low on-resistance PMOS and NMOS power tubes. The low on-resistance ensure low power loss of the chip and make the chip work safely for longer time In addition, In addition, the HR1124S has low standby current and low static operating current, which makes the HR1124S easy to use in toy solutions.

Features: Working voltage: 5V Working current: 200MA Working power: 2W Working temperature: -10°C~ +50°C Schematic diagram of 130 motor module



# Keyestudio Breadboard Power Supply Module



#### Introduction:

This breadboard power supply module is compatible with 5V and 3.3V, which can be applied to MB102 breadboard. The module contains two channels of independent control, powered by the USB all the way.

The output voltage is constant for the DC5V, and another way is powered by DC6.5-12V, output controlled by the slide switch, respectively for DC5V and DC3.3V.

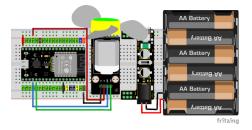
If the other power supply is DC 6.5-12v, when the slide switch is switched to +5V, the output voltages of the left and right lines of the module are DC 5V. When the slide switch is switched to +3V, the output voltage of the USB power supply terminal of the module is DC5V, and the output voltage of the DC 6.5-12V power supply terminal of the other power supply is DC3.3V.

#### **Specification:**

- Applied to MB102 breadboard;
- Input voltageDC 6.5-12V or powered by USB;
- Output voltage3.3V or 5V
- Max output current<700ma
- Up and down two channels of independent control, one of which can be switched to 3.3V or 5V;

Comes with two sets of DC output pins, easy for external use.

4. Wiring Diagram



(Note: Connect the wires and then install a small fan blade on the DC motor. )

5. Project code

You can open the code we provideIf you haven't downloaded the code file, please click on the link to download it:Download Arduino Codes

The code used in this project is saved in folder "2. Windows System\\*\*\*\*2. C\_Tutorial\2. Projects\Project 18Small Fan\Project\_18\_ Small\_Fan".

```
/*
* Filename
           : Small Fan
* Description : Fan clockwise rotation, stop, counterclockwise rotation, stop, cycle.
* Auther
          : http//www.keyestudio.com
*/
##define Motorla
               15 // the Motor_IN+ pin of the motor
##define Motorlb
                2 // the Motor_IN- pin of the motor
void setup(){
 pinMode(Motorla, OUTPUT);//set Motorla to OUTPUT
 pinMode(Motorlb, OUTPUT);//set Motorlb to OUTPUT
}
void loop(){
//Set to rotate for 5s anticlockwise
 digitalWrite(Motorla,HIGH);
 digitalWrite(Motorlb,LOW);
 delay(5000);
//Set to stop rotating for 2s
 digitalWrite(Motorla,LOW);
 digitalWrite(Motorlb,LOW);
 delay(2000);
//Set to rotate for 5s clockwise
 digitalWrite(Motorla,LOW);
 digitalWrite(Motorlb,HIGH);
 delay(5000);
//Set to stop rotating for 2s
 digitalWrite(Motorla,LOW);
 digitalWrite(Motorlb,LOW);
 delay(2000);
}
```

# 6. Project result

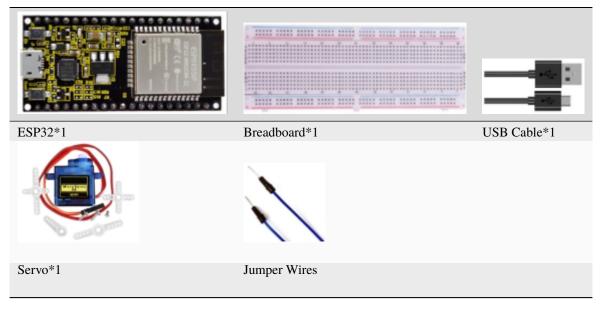
Compile and upload the code to ESP32, after the code is uploaded successfully, power up with a USB cable and you will see that the small fan turns counterclockwise for 5 seconds and stops for 2 seconds, and then turns clockwise for 5 seconds and stops for 2 seconds, which repeats in an endless loop.

# 5.20 Project 19Servo Sweep

### 1. Introduction

Servo is an electric motor that can rotate very precisely. At present, it has been widely used in toy cars, remote control helicoptersairplanesrobots, etc. In this project, we will use ESP32 to control the rotation of the servo.

2. Components



# 3. Component knowledge

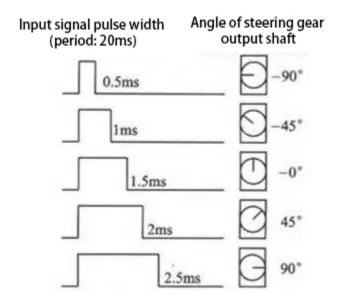
Servo



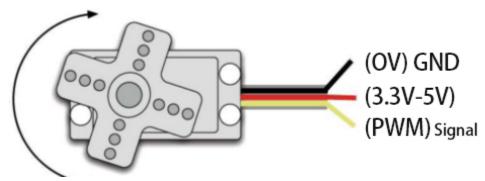
The servo is a kind of position servo driver, which is mainly composed of housingcircuit boardcopless motorgear and position detector. Its working principle is that the receiver or microcontroller sends a signal to the servo which has an internal reference circuit that generates a reference signal with a period of 20ms and a width of 1.5ms, and compares the DC bias voltage with the voltage of the potentiometer to output voltage difference. The IC on the circuit board determines the direction of rotation, and then drives the coreless motor to start rotation and transmits the power to the swing arm through the reduction gear, while the position detector sends back a signal to determine whether it has reached the positioning. It is suitable for those control systems that require constant change of angle and can be maintained.

When the motor rotates at a certain speed, the potentiometer is driven by the cascade reduction gear to rotate so that the voltage difference is 0 and the motor stops rotating. The angle range of general servo rotation is 0 to 180 degrees.

The pulse period for controlling the servo is 20ms, the pulse width is 0.5ms to 2.5ms, and the corresponding position is -90 degrees to +90 degrees. The following is an example of a 180 degree servo



Servo motors have many specifications, but they all have three connecting wires, which are brown, red, and orange (different brands may have different colors). The brown is GND, the red is the positive power supply, and the orange is the signal line.



# 4. Wiring Diagram

When supplying the servo, please note that the power supply voltage should be 3.3V-5V. Make sure there are no errors when connecting the servo to the power supply.



5. Adding the ESP32Servo library

If you have not downloaded the library file, please click on the link to download it:Download Arduino Libraries

This code uses a library named "**ESP32Servo**", If you haven't installed it yet, please do so before learning. The steps to add third-party libraries are as follows:

How to install the library

There are two ways to add libraries:

The first way, open the Arduino IDE, click "Sketch"  $\rightarrow$  "Include Library"  $\rightarrow$  "Manage Libraries". Enter "ESP32Servo" in the search box, select "ESP32Servo" and click "Update" to install. Please refer to the following operations :

Project	_19_Servo_Sweep   Arduino 1.	8.16		—		$\times$
ile Edit S	ketch Tools Help					
ØØ	Verify/Compile Upload	Ctrl+R Ctrl+U	∆ Manage Libraries	Ctrl+Shift+I		ø
Project	Upload Using Programm		Add .ZIP Library			
/* * File	Export compiled Binary	Ctrl+Alt+S	Arduino libraries			
* Desc	Show Sketch Folder	Ctrl+K	Bridge			
* Auth	Include Library		Esplora			
*/	Add File		Ethernet			
<pre>#include</pre>	<esp32servo.h></esp32servo.h>		Firmata			
Servo mys	servo; // create serv	o object to co	GSM			
_		LiquidCrystal				
-	<pre>int posVal = 0; // variable to store the int servoPin = 15; // Servo motor pin</pre>		Mouse			- 1
int serve	5Pin = 15; // Servo mo	tor pin	Robot IR Remote			
void setu	1p() {		Robot Motor			
-	.setPeriodHertz(50);	// s	SD			
myservo }	o.attach(servoPin, 500	, 2500); // a	Servo		serv	o obj
, void loop	p() {		SpacebrewYun			
<			TFT			>
Done uploa	dina		Temboo			
					-1	0.01.1
	ibrary found in C:\Use ibrary found in C:\Use		Contributed libraries			32\1: 32\1:
			Adafruit BuslO			
<			Adafruit DotStarMatrix	c .		>
7		ES	Ad-fault CLI10V		ne on (	сомз

🛛 Library Manager	$\times$
Type All v Topic All V ESP32Servo 1	
ESP32Servo	^
by Kevin Harrington Version 0.8.0 INSTALLED Allows ESP32 boards to control servo, tone and analogWrite motors using Arduino semantics. This library can control a many	
types of servos.	
It makes use of the ESP32 PWM timers: the library can control up to 16 servos on individual channels No attempt has been made to support multiple servos per channel.	
2	
Select version V Install Update	
by Sébastien Matos	
Initial development release Control Parallax Feedback 360° High Speed Servos with a ESP32 dev-board.	
More info	
	-
	~
Clos	se

The second wayopen the Arduino IDEclick "Sketch" $\rightarrow$ "Include Library" $\rightarrow$ "Add .ZIP Library...". In the pop-up window, find the file named\*\*"2. Windows System\2.C\_Tutorial\3.Libraries\\*\*\*\*ESP32Servo-0.8.0.ZIP\*\*which locates in this directory. Select the **ESP32Servo-0.80.ZIP** fileand then click"Open".

Project_19	_Servo_Sweep   Arduino 1.8.1	6		_		×
File Edit Sket	ch Tools Help		 		1	
	Verify/Compile	Ctrl+R	A	Chill Children		Ø
	Upload	Ctrl+U	 Manage Libraries	Ctrl+Shift+I		_
Project	Upload Using Programmer	Ctrl+Shift+U	Add .ZIP Library			
//****	Export compiled Binary	Ctrl+Alt+S	Arduino libraries	•		^
* File	Show Sketch Folder	Ctrl+K	Bridge			
* Desc * Auth	Include Library	;	Esplora			
*/	Add File		Ethernet			
<pre>#include <e< pre=""></e<></pre>	SP32Servo.h>		Firmata			
Servo myser		object to co	GSM			
			LiquidCrystal			
-	= 0; // variable to n = 15; // Servo moto:		Mouse			
100 5010011		- pin	Robot IR Remote			
void setup(			Robot Motor			
-	<pre>setPeriodHertz(50); stach(servoPin, 500, 1)</pre>	// s 2500) - // a	SD		servo	a obf
}	obach(Servorin, 500, 1	2000/ <b>,</b> // a	Servo		Serve	, 00-
<pre>void loop()</pre>	{		SpacebrewYun			~
<			TFT			>

Look <u>i</u> n:	3. Libraries	5					v 👌 📂 🛄 •	
	ESP32Servo-0.	.8.0	1					
	HT16K33_Lib_	For_ESP32						
Recent Items	IRremoteESP8	266-2.7.4						
	🚾 Keypad-3.1.1							
	LCD_128X32							
Desktop	MFRC522_I2C							
	TM1650							
	wire 🔛							2
Documents	🚾 xht11							<b>↓</b> _
	File <u>n</u> ame:	ning Kit Ultim	ate Edition\2.	Windows	System\2.	C_Tutorial\3.	Libraries	Open
This PC	Files of <u>t</u> ype:	ZIP files or f					~	Cancel

6. Project code

After the **ESP32Servo** library is added, You can open the code we provideIf you haven't downloaded the code file, please click on the link to download it:Download Arduino Codes

The code used in this project is saved in folder \*\*"2. Windows System\\*\*\*\*2. C\_Tutorial\2. Projects\Project 19Servo Sweep\Project\_19\_Servo\_Sweep".

```
/*
* Filename : Servo Sweep
* Description : Control the servo motor for sweeping
* Auther
           : http//www.keyestudio.com
*/
##include <ESP32Servo.h>
Servo myservo; // create servo object to control a servo
int posVal = 0; // variable to store the servo position
int servoPin = 15; // Servo motor pin
void setup() {
 myservo.setPeriodHertz(50);
                                   // standard 50 hz servo
 myservo.attach(servoPin, 500, 2500); // attaches the servo on servoPin to the servo
→object
}
void loop() {
 for (posVal = 0; posVal <= 180; posVal += 1) { // goes from 0 degrees to 180 degrees
   // in steps of 1 degree
   myservo.write(posVal);
                              // tell servo to go to position in variable 'pos'
   delay(15);
                              // waits 15ms for the servo to reach the position
 }
 for (posVal = 180; posVal >= 0; posVal -= 1) { // goes from 180 degrees to 0 degrees
   myservo.write(posVal);
                           // tell servo to go to position in variable 'pos'
   delay(15);
                              // waits 15ms for the servo to reach the position
 }
}
```

(continues on next page)

(continued from previous page)

### 6. Project result

Compile and upload the code to ESP32, after the code is uploaded successfully, power up with a USB cable and you will see that the Servo will rotate from 0 degrees to 180 degrees and then reverse the direction to make it rotate from 180 degrees to 0 degrees and repeat these actions in an endless loop.

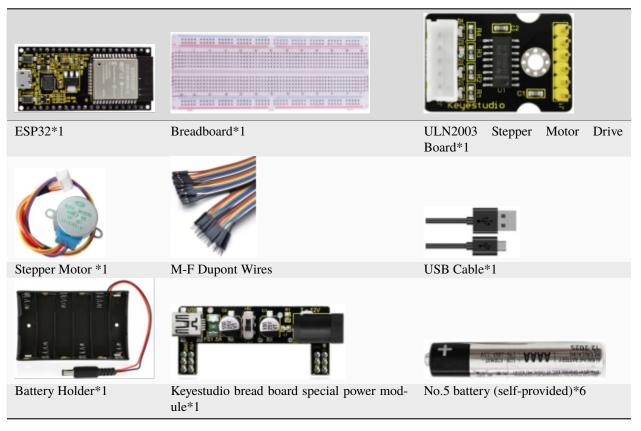


# 5.21 Project 20Stepping Motor

1. Introduction

Stepper motor is the most important part of industrial robot 3D printer lathes and other mechanical equipment with accurate positioning. In this project, we will use ESP32 control ULN2003 stepper motor drive board to drive the stepper motor to rotate.

2. Components



3. Component knowledge

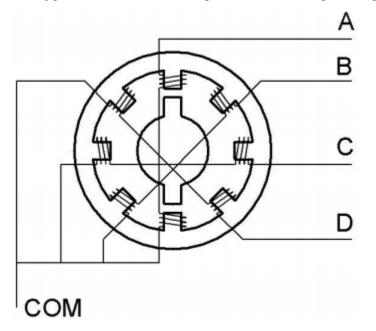


**Stepper motor:** It is a motor controlled by a series of electromagnetic coils. It can rotate by the exact number of degrees (or steps) needed, allowing you to move it to a precise position and keep it there. It does this by supplying power to the coil inside the motor in a very short time, but you must always supply power to the motor to keep it in the position you want. There are two basic types of stepping motors, namely unipolar stepping motor and bipolar stepping motor. In this project, we use a 28-BYJ48 unipolar stepper motor.



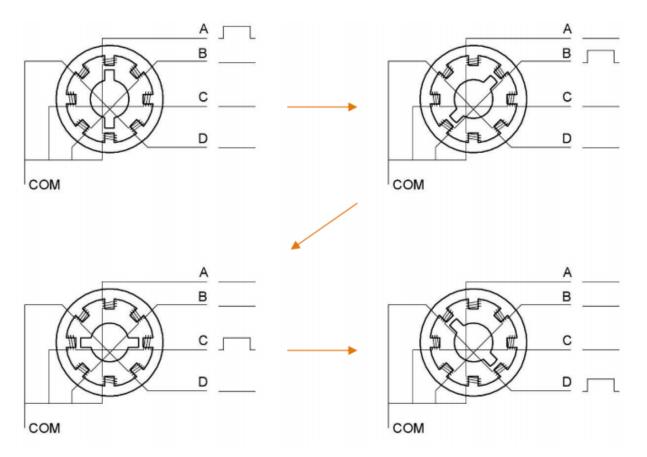
# Working Principle:

The stepper motor is mainly composed of a stator and a rotor. The stator is fixed. As shown in the figure below, the part of the coil group A, B, C, and D will generate a magnetic field when the coil group is energized. The rotor is the rotating part. As follows, the middle part of the stator, two poles are permanent magnets.



Single -phase four beat: At the beginning, the coils of group A are turned on, and the poles of the rotor point at A coil. Next, the group A coil are disconnected, and the group B coils are turned on. The rotor will turn clockwise to the group B. Then, group B is disconnected, group C is turned on, and the rotor is turned to group C. After that, group C is disconnected, and group D is turned on, and the rotor is turned to group D. Finally, group D is disconnected, group A coils. Therefore, rotor turns 180° and continuously rotates B-C-D-A, which means it runs a circle (eight phase). As shown below, he rotation principle of stepper motor is A - B C - D - A.

You make order inverse(D - C - B - A - D ...) if you want to make stepper motor rotate anticlockwise.



Half-phase and eight beat: 8 beat adopts single and dual beat wayA - AB B - BC - C - CD - D - DA - A  $\dots$  rotor will rotate half phase in this order. For example, when A coil is electrifiedrotor faces to A coil, then A and B coil are connected, on this condition, the strongest magnetic field produced lies in the central part of AB coil, which means rotating half-phase clockwise.

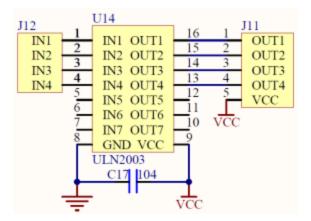
# **Stepper Motor Parameters:**

The rotor rotates one circle when the stepper motor we provide rotates 32 phases and with the output shaft driven by 1:64 reduction geared set. Therefore the rotation (a circle) of output shaft requires 32 \* 64 = 2048 phases.

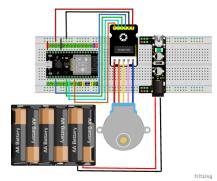
The step angle of 4-beat mode of 5V and 4-phase stepper motor is 11.25. And the step angle of 8-beat mode is 5.625, the reduction ratio is 1:64.

**ULN2003Stepper Motor Drive Board:** It is a stepper motor driver, which converts the weak signal into a stronger control signal to drive the stepper motor.

The following schematic diagram shows how to use the ULN2003 stepper motor driver board interface to connect a unipolar stepper motor to the pins of the ESP32, and shows how to use four TIP120 interfaces.



4. Wiring Diagram



5. Project code

You can open the code we provideIf you haven't downloaded the code file, please click on the link to download it:Download Arduino Codes

The code used in this project is saved in folder \*\*"2. Windows System\\*\*\*2. C\_Tutorial\2. Projects\Project 20Stepping Motor\Project\_20\_Stepping\_Motor".

```
/*
* Filename : Drive Stepper Motor
* Description : Use ULN2003 to drive the stepper motor.
* Auther
         : http//www.keyestudio.com
*/
// Conncet the port of the stepper motor driver
int outPorts[] = {15, 16, 17, 18};
void setup() {
 // set pins to output
 for (int i = 0; i < 4; i++) {
   pinMode(outPorts[i], OUTPUT);
 }
}
void loop()
{
 // Rotate a full turn
 moveSteps(true, 32 * 64, 3);
```

(continues on next page)

```
(continued from previous page)
```

```
delay(1000);
  // Rotate a full turn towards another direction
  moveSteps(false, 32 * 64, 3);
  delay(1000);
}
//Suggestion: the motor turns precisely when the ms range is between 3 and 20
void moveSteps(bool dir, int steps, byte ms) {
  for (unsigned long i = 0; i < steps; i++) {</pre>
    moveOneStep(dir); // Rotate a step
    delay(constrain(ms,3,20));
                                    // Control the speed
 }
}
void moveOneStep(bool dir) {
  // Define a variable, use four low bit to indicate the state of port
 static byte out = 0 \times 01;
  // Decide the shift direction according to the rotation direction
  if (dir) { // ring shift left
    out != 0 \times 08 ? out = out << 1 : out = 0 \times 01;
  }
              // ring shift right
  else {
    out != 0 \times 01 ? out = out >> 1 : out = 0 \times 08;
  }
  // Output singal to each port
  for (int i = 0; i < 4; i++) {
    digitalWrite(outPorts[i], (out & (0x01 << i)) ? HIGH : LOW);</pre>
  }
}
void moveAround(bool dir, int turns, byte ms){
  for(int i=0;i<turns;i++)</pre>
    moveSteps(dir,32*64,ms);
}
void moveAngle(bool dir, int angle, byte ms){
  moveSteps(dir,(angle*32*64/360),ms);
}
                            ************************************
```

# 6. Project result

Compile and upload the code to ESP32, after the code is uploaded successfully, power up with a USB cable and you will see that the four LEDs (D1,D2,D3,D4) on the ULN2003 drive module will light up. The stepper motor rotates clockwise first, then counterclockwise, and repeat these actions in an endless loop.

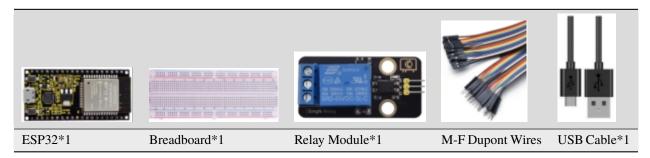


# 5.22 Project 21Relay

# 1. Introduction

In our daily life, we usually use communication to drive electrical equipments, and sometimes we use switches to control electrical equipments. If the switch is connected directly to the ac circuit, leakage occurs and people are in danger. Therefore, from the perspective of safety, we specially designed this relay module with NO(normally open) end and NC(normally closed) end. In this project, we will learn a relatively special and easy-to-use switch, which is the relay module.

2. Components



# 3. Component knowledge

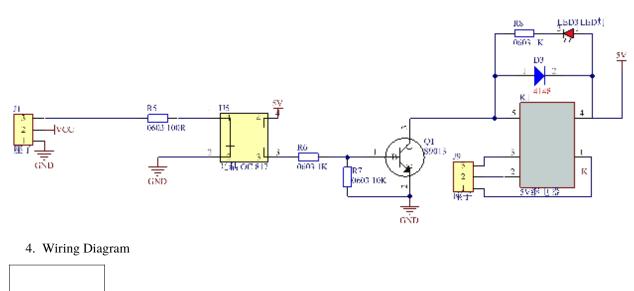
Relay: It is an "automatic switch" that uses a small current to control the operation of a large current.

Input voltage3.3V-5V

```
Rated load5A 250VAC (NO/NC) 5A 24VDC (NO/NC)
```

The rated load means that devices with dc voltage of 24V or AC voltage of 250V can be controlled using 3.3V-5V microcontrollers.

# Schematic diagram of Relay



# media/1741d3cb0405c740378ef7ef96df6072.png

5. Project code

You can open the code we provideIf you haven't downloaded the code file, please click on the link to download it:Download Arduino Codes

The code used in this project is saved in folder "Arduino-Codes\Project 21Relay\Project\_21\_Relay".

```
/*
* Filename
        : Relay
* Description : Relay turn on and off.
* Auther
        : http//www.keyestudio.com
*/
##define Relay 15 // defines digital 15
void setup()
{
pinMode(Relay, OUTPUT); // sets "Relay" to "output"
}
void loop()
{
digitalWrite(Relay, HIGH); // turns on the relay
delay(1000); //delays 1 seconds
digitalWrite(Relay, LOW); // turns off the relay
delay(1000); // delays 1 seconds
}
```

#### 6. Project result

Compile and upload the code to ESP32, after the code is uploaded successfully, power up with a USB cable and you will see that the relay will cycle on and off, on for 1 second, off for 1 second. At the same time, you can hear the sound of the relay on and off, and you can also see the change of the indicator light on the relay.

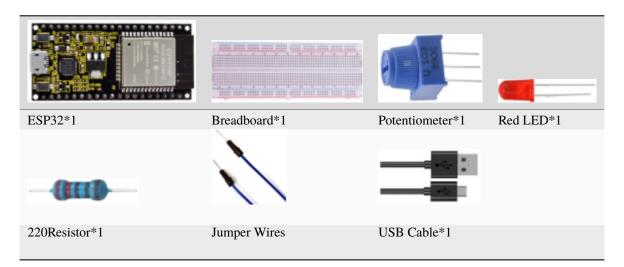
# 5.23 Project 22Dimming Light

1. Introduction

A potentiometer is a three-terminal resistor with sliding or rotating contacts that forms an adjustable voltage divider. It works by changing the position of the sliding contacts across a uniform resistance. In the potentiometer, the entire input voltage is applied across the whole length of the resistor, and the output voltage is the voltage drop between the fixed and sliding contact.

In this project, we will learn how to use ESP32 to read the values of the potentiometer, and make a dimming lamp with LED.

2. Components

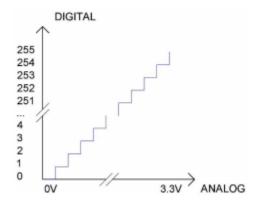


3. Component knowledge



Adjustable potentiometer: It is a kind of resistor and an analog electronic component, which has two states of 0 and 1(high level and low level). The analog quantity is different, its data state presents a linear state such as  $1 \sim 1024$ 

**ADC :** An ADC is an electronic integrated circuit used to convert analog signals such as voltages to digital or binary form consisting of 1s and 0s. The range of our ADC on ESP32 is 12 bits, that means the resolution is 2^12=4096, and it represents a range (at 3.3V) will be divided equally to 4096 parts. The rage of analog values corresponds to ADC values. So the more bits the ADC has, the denser the partition of analog will be and the greater the precision of the resulting conversion.



Subsection 1: the analog in rang of 0V—3.3/4095 V corresponds to digital 0;

Subsection 2: the analog in rang of 3.3/4095 V—2\*3.3 /4095V corresponds to digital 1;

• • •

The following analog will be divided accordingly.

The conversion formula is as follows:

$$ADCValue = \frac{Ana\log Voltage}{3.3} * 4095$$

**DAC** The reversing of this process requires a DAC, Digital-to-Analog Converter. The digital I/O port can output high level and low level (0 or 1), but cannot output an intermediate voltage value. This is where a DAC is useful. ESP32 has two DAC output pins with 8-bit accuracy, GPIO25 and GPIO26, which can divide VCC

(here is 3.3V) into  $2^8=256$  parts. For example, when the digital quantity is 1, the output voltage value is 3.3/256 \* 1 V, and when the digital quantity is 128, the output voltage value is 3.3/256\*128=1.65V, the higher the accuracy of DAC, the higher the accuracy of output voltage value will be.

The conversion formula is as follows:

$$Ana \log Voltage = \frac{DACValue}{255} * 3.3(V)$$

# ADC on ESP32

ESP32 has 16 pins can be used to measure analog signals. GPIO pin sequence number and analog pin definition are shown in the following table

# DAC on ESP32

ESP32 has two 8-bit digital analog converters to be connected to GPIO25 and GPIO26 pins, respectively, and it is immutable. As shown in the following table

The DAC pin number is already defined in ESP32's code base; for example, you can replace GPIO25 with DAC1 in the code.

#### 4. Read the ADC value, DAC value and voltage value of the potentiometer

We connect the potentiometer to the analog IO port of ESP32 to read the ADC value, DAC value and voltage value of the potentiometer, please refer to the wiring diagram below

media/0cda3256a0930404abc097ec8ffa3013.png

You can open the code we provideIf you haven't downloaded the code file, please click on the link to download it:Download Arduino Codes

The code used in this project is saved in folder "Arduino-Codes\Project 22Dimming Light\Project\_22.1\_Read\_Potentiometer\_Analog\_Value".

(continues on next page)

```
* Filename : Read Potentiometer Analog Value
* Description : Basic usage of ADCDAC and Voltage
* Auther
              : http//www.keyestudio.com
*/
##define PIN_ANALOG_IN 36 //the pin of the Potentiometer
void setup() {
 Serial.begin(115200);
}
//In loop()the analogRead() function is used to obtain the ADC value,
//and then the map() function is used to convert the value into an 8-bit precision DAC.
\rightarrow value.
//The input and output voltage are calculated according to the previous formula,
//and the information is finally printed out.
void loop() {
 int adcVal = analogRead(PIN_ANALOG_IN);
 int dacVal = map(adcVal, 0, 4095, 0, 255);
 double voltage = adcVal / 4095.0 * 3.3;
 Serial.printf("ADC Val: %d, \t DAC Val: %d, \t Voltage: %.2fV\n", adcVal, dacVal,
\rightarrow voltage);
 delay(200);
}
```

Compile and upload the code to ESP32, after the code is uploaded successfully, power up with a USB cable, open the serial monitor and set the baud rate to 115200. You will see that the serial port monitor window will print out the ADC value, DAC value and voltage value of the potentiometer. When turning the potentiometer handle, the ADC value, DAC value and voltage value will change. As shown below:

💿 сомз				_	
					Send
ADC Val: 76,	DAC Val: 4,	Voltage: 0.06V			^
ADC Val: 402,	DAC Val: 25,	Voltage: 0.32V			
ADC Val: 908,	DAC Val: 56,	Voltage: 0.73V			
ADC Val: 909,	DAC Val: 56,	Voltage: 0.73V			
ADC Val: 1577	, DAC Val: 98,	Voltage: 1.27V			
ADC Val: 1711	, DAC Val: 106,	Voltage: 1.38V			
ADC Val: 1904	, DAC Val: 118,	Voltage: 1.53V			
ADC Val: 2265	, DAC Val: 141,	Voltage: 1.83V			
ADC Val: 2271	, DAC Val: 141,	Voltage: 1.83V			
ADC Val: 2555	, DAC Val: 159,	Voltage: 2.06V			
ADC Val: 2547	, DAC Val: 158,	Voltage: 2.05V			
ADC Val: 3003	, DAC Val: 187,	Voltage: 2.42V			
ADC Val: 3327	, DAC Val: 207,	Voltage: 2.68V			
ADC Val: 4095	, DAC Val: 255,	Voltage: 3.30V			
ADC Val: 4095	, DAC Val: 255,	Voltage: 3.30V			
					~
Autoscroll [	Show timestamp		Newline $\vee$	115200 baud $~\sim$	Clear output

5. Wiring diagram of the dimming lamp

In the previous step, we read the ADC value, DAC value and voltage value of the potentiometer. Now we need to convert the ADC value of the potentiometer into the brightness of the LED to make a lamp that can adjust the brightness. The wiring diagram is as follows:



6. Project code

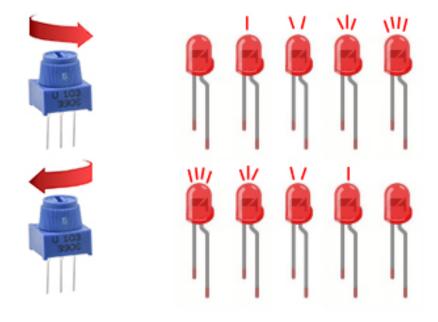
You can open the code we provideIf you haven't downloaded the code file, please click on the link to download it:Download Arduino Codes

The code used in this project is saved in folder "Arduino-Codes\Project 22Dimming Light\Project\_22.2\_Dimming\_Light".

```
/*
* Filename : Dimming Light
* Description : Controlling the brightness of LED by potentiometer.
         : http//www.keyestudio.com
* Auther
*/
##define PIN ANALOG IN
                  36 //the pin of the potentiometer
##define PIN_LED 15 // the pin of the LED
##define CHAN
                  0
void setup() {
 ledcSetup(CHAN, 1000, 12);
 ledcAttachPin(PIN_LED, CHAN);
}
void loop() {
 int adcVal = analogRead(PIN_ANALOG_IN); //read adc
 int pwmVal = adcVal;
                 // adcVal re-map to pwmVal
 ledcWrite(CHAN, pwmVal);
                     // set the pulse width.
 delay(10);
}
```

7. Project result

Compile and upload the code to ESP32, after the code is uploaded successfully, power up with a USB cable and you will see that turn the potentiometer handle and the brightness of the LED will change accordingly.

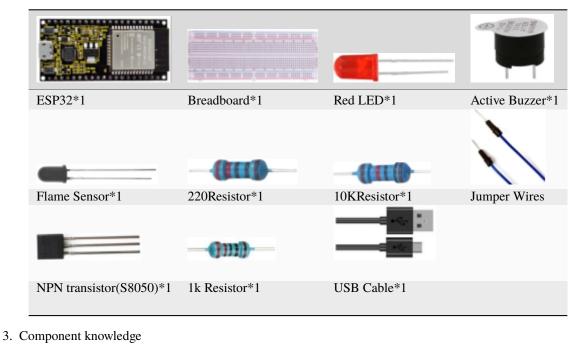


# 5.24 Project 23Flame Alarm

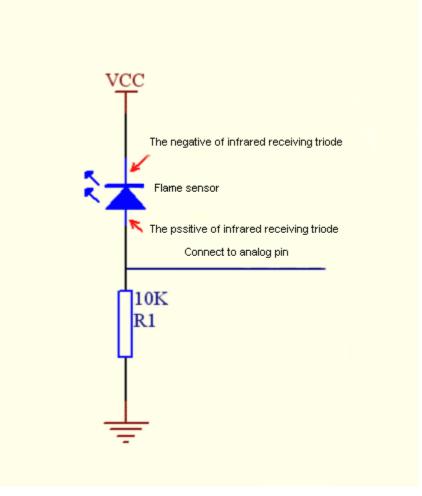
1. Introduction

Fire is a terrible disaster and fire alarm systems are very useful in housescommercial buildings and factories. In this project, we will use ESP32 to control a flame sensor, a buzzer and a LED to simulate fire alarm devices. This is a meaningful maker activity.

2. Components



The flame emits a certain amount IR light that is invisible to the human eye, but our flame sensor can detect it and alert a microcontroller(such as ESP32) that a fire has been detected. It has a specially designed infrared receiver tube to detect the flame and then convert the flame brightness into a fluctuating level signal. The short pin of the receiving triode is negative pole and the other long pin is positive pole. We should connect the short pin (negative) to 5V and the long pin (positive) to the analog pin, a resistor and GND. As shown in the figure below



**Note:** Since vulnerable to radio frequency radiation and temperature changes, the flame sensor should be kept away from heat sources like radiators, heaters and air conditioners, as well as direct irradiation of sunlight, headlights and incandescent light.

#### 4. Read the ADC value, DAC value and voltage value of the flame sensor

We first use a simple code to read the ADC value, DAC value and voltage value of the flame sensor and print them out. Please refer to the wiring diagram below



You can open the code we provideIf you haven't downloaded the code file, please click on the link to download it:Download Arduino Codes

The code used in this project is saved in folder " Arduino-Codes\Project 23Flame Alarm\Project\_23.1\_Read\_Analog\_Value\_Of\_Flame\_Sensor ".

(continues on next page)

```
* Filename
              : Read Analog Value Of Flame Sensor
 * Description : Basic usage of ADCDAC and Voltage
* Auther
              : http//www.keyestudio.com
*/
##define PIN_ANALOG_IN 36 //the pin of the Flame sensor
void setup() {
 Serial.begin(115200);
}
//In loop()the analogRead() function is used to obtain the ADC value,
//and then the map() function is used to convert the value into an 8-bit precision DAC.
\rightarrow value.
//The input and output voltage are calculated according to the previous formula,
//and the information is finally printed out.
void loop() {
 int adcVal = analogRead(PIN_ANALOG_IN);
 int dacVal = map(adcVal, 0, 4095, 0, 255);
 double voltage = adcVal / 4095.0 * 3.3;
 Serial.printf("ADC Val: %d, \t DAC Val: %d, \t Voltage: %.2fV\n", adcVal, dacVal,
\rightarrow voltage);
 delay(200);
}
```

Compile and upload the code to ESP32, after the code is uploaded successfully, power up with a USB cable, open the serial monitor and set the baud rate to 115200. You will see that the serial port monitor window will print out the ADC value, DAC value and voltage value of the flame sensor. When the sensor is closed to fire, the ADC value, DAC value and voltage value will get greater. Conversely, the ADC value, DAC value and voltage value decrease.

0	COM3					-			$\times$
								Se	end
ADC	Val:	76,	DAC Val: 4,	Voltage: 0.06V					^
ADC	Val:	402,	DAC Val: 25,	Voltage: 0.32V					
ADC	Val:	908,	DAC Val: 56,	Voltage: 0.73V					
ADC	Val:	909,	DAC Val: 56,	Voltage: 0.73V					
ADC	Val:	1577,	DAC Val: 98,	Voltage: 1.27V					
ADC	Val:	1711,	DAC Val: 106,	Voltage: 1.38V					
ADC	Val:	1904,	DAC Val: 118,	Voltage: 1.53V					
ADC	Val:	2265,	DAC Val: 141,	Voltage: 1.83V					
ADC	Val:	2271,	DAC Val: 141,	Voltage: 1.83V					
ADC	Val:	2555,	DAC Val: 159,	Voltage: 2.06V					
ADC	Val:	2547,	DAC Val: 158,	Voltage: 2.05V					
ADC	Val:	3003,	DAC Val: 187,	Voltage: 2.42V					
ADC	Val:	3327,	DAC Val: 207,	Voltage: 2.68V					
ADC	Val:	4095,	DAC Val: 255,	Voltage: 3.30V					
ADC	Val:	4095,	DAC Val: 255,	Voltage: 3.30V					
									$\checkmark$
<u>I</u>	Autosc:	roll 🗌	Show timestamp		Newline $\sim$	115200 baud	~	Clear o	utput

5. Wiring diagram of the flame alarm

Next, we will use a flame sensor, a buzzer, and a LED to make an interesting project, that is flame alarm. When flame is detected, the LED flashes and the buzzer alarms.

media/e9fa0e50df23c1f2e58fdd319ad21b4c.png

```
value = get value()
if value >500:
```

6. Project codeNote

codeNote **buzzer.value(1)** the threshold of 500 in the code can be reset itself as required)

You can open the code we provideIf you haven't downloaded the code file, please click on the link to download it:Download Arduino Codes

The code used in this project is saved in folder "Arduino-Codes\Project 23Flame Alarm\Project\_23.2\_Flame\_Alarm".

```
/*
* Filename : Flame Alarm
* Description : Controlling the buzzer and LED by flame sensor.
            : http//www.keyestudio.com
* Auther
*/
##define PIN_ADC0
                   36 //the pin of the flame sensor
##define PIN_LED
                   15 // the pin of the LED
##define PIN_BUZZER
                   4 // the pin of the buzzer
void setup() {
 pinMode(PIN_LED, OUTPUT);
 pinMode(PIN_BUZZER, OUTPUT);
 pinMode(PIN_ADC0, INPUT);
}
void loop() {
 int adcVal = analogRead(PIN_ADC0); //read the ADC value of flame sensor
 if (adcVal >= 500) {
   digitalWrite (PIN_BUZZER, HIGH); //turn on buzzer
   digitalWrite(PIN_LED, HIGH); // turn on LED
   delay(500); // wait a second.
   digitalWrite (PIN_BUZZER, LOW);
   digitalWrite(PIN_LED, LOW); // turn off LED
   delay(500); // wait a second
 }
else
{
   digitalWrite(PIN_LED, LOW); //turn off LED
   digitalWrite (PIN_BUZZER, LOW); //turn off buzzer
 }
}
```

7. Project result

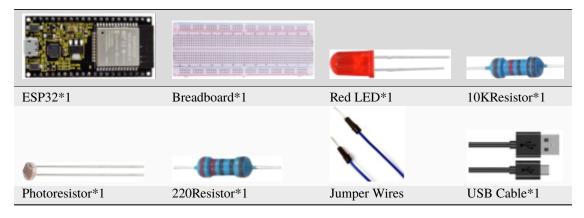
Compile and upload the code to ESP32, after the code is uploaded successfully, power up with a USB cable and you will see that when the flame sensor detects the flame, the LED will flash and the buzzer will alarm; otherwise, the LED does not light up and the buzzer does not sound.

# 5.25 Project 24Night Lamp

#### 1.Introduction

Sensors or components are ubiquitous in our daily life. For example, some public street lamps will automatically turn on at night and turn off during the day. Why? In fact, this make use of a photosensitive element that senses the intensity of external ambient light. When the outdoor brightness decreases at night, the street lights will turn on automatically. In the daytime, the street lights will automatically turn off. the principle of which is very simple, In this Project, we use ESP32 to control a LED to achieve the effect of the street light.

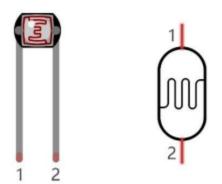
2. Components



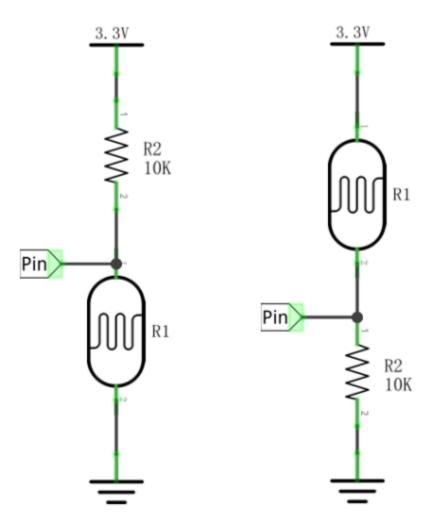
3. Component knowledge



**Photoresistor :** It is a kind of photosensitive resistance, its principle is that the photoresistor surface receives brightness (light) to reduce the resistance, the resistance value will change with the detected intensity of the ambient light. With this characteristic, we can use the photosensitive resistance to detect the light intensity. Photosensitive resistance and its electronic symbol are as follows



The following circuit is used to detect changes in resistance values of photoresistors



In the circuit above, when the resistance of the photoresistor changes due to the change of light intensity, the voltage between the photoresistor and resistance R2 will also change. Thus, the intensity of light can be obtained by measuring this voltage.

4. Read the ADC value, DAC value and voltage value of the photoresistor

We first use a simple code to read the ADC value, DAC value and voltage value of the photoresistor and print them out. Please refer to the following wiring diagram

media/b762098c798beb08e4d433137c317dc7.png

You can open the code we provideIf you haven't downloaded the code file, please click on the link to download it:Download Arduino Codes

The code used in this project is saved in folder "2. Windows System\\*\*\*\*2. C\_Tutorial\2. Projects\Project 24Night Lamp\Project\_24.1\_Read\_Photosensitive\_Analog\_Value".

```
* Description : Basic usage of ADC
 * Auther
             : http//www.keyestudio.com
*/
##define PIN_ANALOG_IN 36 //the pin of the photosensitive sensor
void setup() {
 Serial.begin(115200);
}
//In loop()the analogRead() function is used to obtain the ADC value,
//and then the map() function is used to convert the value into an 8-bit precision DAC_
\rightarrow value.
//The input and output voltage are calculated according to the previous formula,
//and the information is finally printed out.
void loop() {
 int adcVal = analogRead(PIN_ANALOG_IN);
 int dacVal = map(adcVal, 0, 4095, 0, 255);
 double voltage = adcVal / 4095.0 * 3.3;
 Serial.printf("ADC Val: %d, \t DAC Val: %d, \t Voltage: %.2fV\n", adcVal, dacVal,_
→voltage);
 delay(200);
}
```

Compile and upload the code to ESP32, after the code is uploaded successfully, power up with a USB cable, open the serial monitor and set the baud rate to 115200. You will see that the serial port monitor window will print out the ADC valueDAC value and voltage value of the photoresistor. When the light intensity around the photoresistor is gradually reduced, the ADC value, DAC value and voltage value will gradually increase. On the contrary, the ADC value, DAC value and voltage value decreases gradually.

0	COM3									_		$\times$
												Send
ADC	Val:	76,	DAC Val: 4	, Voltage:	0.0EV							^
ADC	Val:	402,	DAC Val: 2	5, Voltage:	0.32V							
ADC	Val:	908,	DAC Val: 5	6, Voltage:	0.73V							
ADC	Val:	909,	DAC Val: 5	6, Voltage:	0.73V							
ADC	Val:	1577,	DAC Val: 9	8, Voltage:	1.27V							
ADC	Val:	1711,	DAC Val: 1	.06, Voltage:	1.38V							
ADC	Val:	1904,	DAC Val: 1	18, Voltage:	1.53V							
ADC	Val:	2265,	DAC Val: 1	41, Voltage:	1.83V							
ADC	Val:	2271,	DAC Val: 1	41, Voltage:	1.83V							
ADC	Val:	2555,	DAC Val: 1	.59, Voltage:	2.06V							
ADC	Val:	2547,	DAC Val: 1	.58, Voltage:	2.05V							
ADC	Val:	3003,	DAC Val: 1	.87, Voltage:	2.42V							
ADC	Val:	3327,	DAC Val: 2	07, Voltage:	2.68V							
ADC	Val:	4095,	DAC Val: 2	55, Voltage:	3.30V							
ADC	Val:	4095,	DAC Val: 2	55, Voltage:	3.30V							
												×
	Autosc	roll 🗌	]Show timestam	p		Newline	~	115200	baud	$\sim$	Clear	output

5. Wiring diagram of the light-controlled lamp

We made a small dimming lamp in the front, now we will make a light controlled lamp. The principle is the same, that is, the ESP32 takes the ADC value of the sensor, and then adjusts the brightness of the LED.

media/77a0c534501f51e7fe7aa221e4db71d9.png

6. Project code

You can open the code we provideIf you haven't downloaded the code file, please click on the link to download it:Download Arduino Codes

The code used in this project is saved in folder\*\*"2. Windows System\*\*\*\*\2. C\_Tutorial\2. Projects\Project 24Night Lamp\Project\_24.2\_Night\_Lamp".

```
/*
* Filename
           : Night Lamp
* Description : Controlling the brightness of LED by photosensitive sensor.
* Auther
           : http//www.keyestudio.com
*/
##define PIN_ANALOG_IN 36 // the pin of the photosensitive sensor
##define PIN LED
               15 // the pin of the LED
##define CHAN
                   0
##define LIGHT_MIN
                   372
##define LIGHT_MAX
                   2048
void setup() {
 ledcSetup(CHAN, 1000, 12);
 ledcAttachPin(PIN_LED, CHAN);
}
void loop() {
 int adcVal = analogRead(PIN_ANALOG_IN); //read adc
 int pwmVal = map(constrain(adcVal, LIGHT_MIN, LIGHT_MAX), LIGHT_MIN, LIGHT_MAX, 0,_
→4095); // adcVal re-map to pwmVal
 ledcWrite(CHAN, pwmVal);
                     // set the pulse width.
 delay(10);
}
```

7. Project result

Compile and upload the code to ESP32, after the code is uploaded successfully, power up with a USB cable and you will see that when the intensity of light around the photoresistor is reduced, the LED will be bright, on the contraty, the LED will be dim.

# 5.26 Project 25Human Induction Lamp

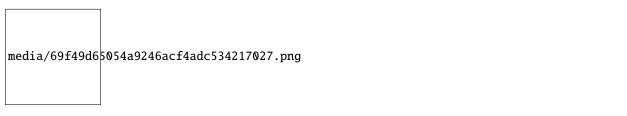
1. Introduction

Human body induction lamp is used commonly in the dark corridor area. With the development of science and technology, the use of the human body induction lamp is very common in our real life, such as the corridor of the community, the bedroom of the room, the garage of the dungeon, the bathroom and so on. The human induction lamp are generally composed of a human body infrared sensor, a led, a photoresistor sensor and so on.

In this project, we will learn how to use a Human Body Infrared Sensor, a led, and a photoresistor to make a human induction lamp.

- . . . ESP32\*1 Breadboard\*1 Red LED\*1 10KResistor\*1 Jumper USB Cable\*1 Wires Photoresistor\*1 Human Body Infrared 220Resistor\*1 M-F Dupont Sensor\*1 Wires
- 2. Components

3. Wiring Diagram



4. Project code

You can open the code we provideIf you haven't downloaded the code file, please click on the link to download it:Download Arduino Codes

The code used in this project is saved in folder "Arduino-Codes\Project 25Human Induction Lamp\Project\_25\_Human\_Induction\_Lamp".

(continues on next page)

```
// the pin of the PIR motion sensor
##define pirPin
                15
byte pirStat = 0;
                 // the state of the PIR motion sensor
void setup() {
 Serial.begin(115200);
 pinMode(PIN_LED, OUTPUT);
 pinMode(PIN_ADC0, INPUT);
 pinMode(pirPin, INPUT);
}
void loop() {
 int adcVal = analogRead(PIN_ADC0); //read the ADC value of photosensitive sensor
 pirStat = digitalRead(pirPin); //read the value of PIR motion sensor
 if (adcVal >= 2000) {
     if (pirStat == HIGH){
        digitalWrite(PIN_LED, HIGH);//turn on the LED
        }
     else{
        digitalWrite(PIN_LED, LOW);//turn off the LED
       }
 }
  else{
     digitalWrite(PIN_LED, LOW);//turn off the LED
     }
}
```

#### 5. Project result

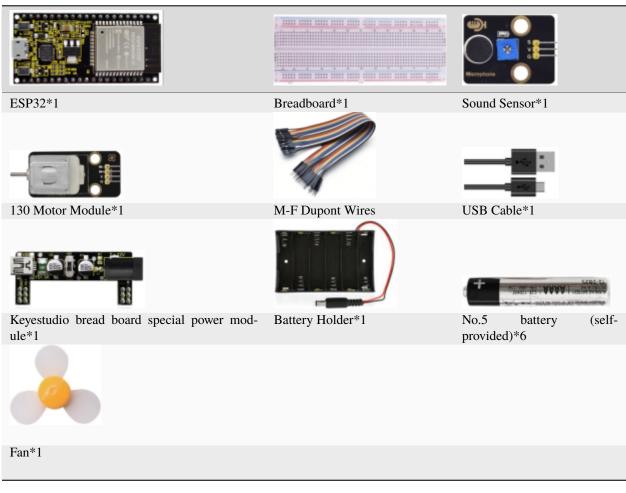
Compile and upload the code to ESP32, after the code is uploaded successfully, power up with a USB cable and you will see thatWhen your hand covers the photosensitive part of the photoresistor to simulate darkness, then shake your other hand in front of the Human Body Infrared Sensor, the external LED will light up. If the photosensitive part of the photoresistor is not covered, then shake your hand in front of the human infrared sensor and the LED is turned off.

## 5.27 Project 26Sound Control Fan

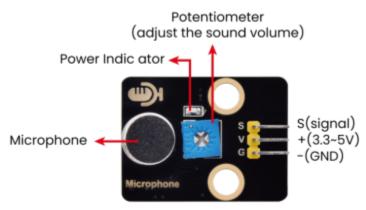
1. Introduction

The sound sensor has a built-in capacitive electret microphone and power amplifier which can be used to detect the sound intensity of the environment. In this project, we use ESP32 to control the sound sensor and the motor module to simulate a voice-controlled fan.

2. Components



3. Component knowledge



Sound sensor is usually used to detect the loudness of the sound in the surrounding environment. Microcontrol board can collect its output signal through the analog input interface. The S pin is an analog output, which is the real-time output of the microphone voltage signal. The sensor comes with a potentiometer so you can adjust the signal strength. It also has two fixing holes so that the sensor can be installed on any other equipment. You can use it to make some interactive works, such as voice-operated switches.

4. Read the ADC value, DAC value and voltage value of the sound sensor

We first use a simple code to read the ADC value, DAC value and voltage value of the sound sensor and print them out.

Please refer to the wiring diagram below

media/87fb44c475d1f53aa5905cebfed55ea2.png

You can open the code we provideIf you haven't downloaded the code file, please click on the link to download it:Download Arduino Codes

The code used in this project is saved in folder "Arduino-Codes\Project 26Sound Control Fan\Project\_26.1\_Read\_Sound\_Sensor\_Analog\_Value".

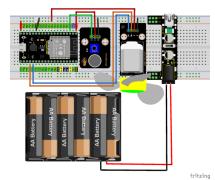
```
/*
* Filename : Read Sound Sensor Analog Value
* Description : Basic usage of ADC
* Auther
          : http//www.keyestudio.com
*/
##define PIN_ANALOG_IN 36 //the pin of the Sound Sensor
void setup() {
 Serial.begin(115200);
}
//In loop()the analogRead() function is used to obtain the ADC value,
//and then the map() function is used to convert the value into an 8-bit precision DAC.
\rightarrow value.
//The input and output voltage are calculated according to the previous formula,
//and the information is finally printed out.
void loop() {
 int adcVal = analogRead(PIN_ANALOG_IN);
 int dacVal = map(adcVal, 0, 4095, 0, 255);
 double voltage = adcVal / 4095.0 * 3.3;
 Serial.printf("ADC Val: %d, \t DAC Val: %d, \t Voltage: %.2fV\n", adcVal, dacVal,
→voltage);
 delay(200);
}
```

Compile and upload the code to ESP32, after the code is uploaded successfully, power up with a USB cable, open the serial monitor and set the baud rate to 115200. You will see that the serial port monitor window will print out the ADC valueDAC value and voltage value of the sound sensor. When you clap your hands to the sensor, the ADC valueDAC value and voltage value will change significantly.

🗠 сомз						Sei	×
						26	na
ADC Val: 72,	DAC Val: 4,	Voltage: 0.06V					1
ADC Val: 0,	DAC Val: 0,	Voltage: 0.00V					
ADC Val: 104,	DAC Val: 6,	Voltage: 0.08V					
ADC Val: 0,	DAC Val: 0,	Voltage: 0.00V					
ADC Val: 0,	DAC Val: 0,	Voltage: 0.00V					
ADC Val: 1741,	DAC Val: 108,	Voltage: 1.40V					
ADC Val: 1814,	DAC Val: 112,	Voltage: 1.46V					
ADC Val: 1328,	DAC Val: 82,	Voltage: 1.07V					
ADC Val: 0,	DAC Val: 0,	Voltage: 0.00V					
ADC Val: 0,	DAC Val: 0,	Voltage: 0.00V					
ADC Val: 1873,	DAC Val: 116,	Voltage: 1.51V					
ADC Val: 1216,	DAC Val: 75,	Voltage: 0.98V					
ADC Val: 0,	DAC Val: 0,	Voltage: 0.00V					
ADC Val: 0,	DAC Val: 0,	Voltage: 0.00V					
ADC Val: 0,	DAC Val: 0,	Voltage: 0.00V					- 1
Autoscroll	Show timestamp		Newline	~ 115	5200 baud 🗸	Clear ou	utpu

#### 5. Wiring diagram of the intelligent fan

Next, we officially entered the project. We used a sound sensor, a motor module and a fan blade to simulate a voicecontrolled fan. The wiring diagram is as follows



(Note: Connect the wires and then install a small fan blade on the DC motor. )

6. Project codeNote **if value >600**: The threshold 600 in the code can be reset itself as needed)

You can open the code we provideIf you haven't downloaded the code file, please click on the link to download it:Download Arduino Codes

The code used in this project is saved in folder "Arduino-Codes\Project 26Sound Control Fan\Project\_26.2\_Sound\_Control\_Fan".

```
##define PIN_Motorlb
                     2 // the Motor_IN- pin of the motor
void setup() {
 pinMode(PIN_Motorla, OUTPUT);//set Motorla to OUTPUT
 pinMode(PIN_Motorlb, OUTPUT);//set Motorlb to OUTPUT
 pinMode(PIN_ADC0, INPUT);//set PIN_ADC2 to INPUT
}
void loop() {
 int adcVal = analogRead(PIN_ADC0); //read the ADC value of Sound sensor
 if (adcVal > 600) {
   digitalWrite(PIN_Motorla,HIGH); //rotate
   digitalWrite(PIN_Motorlb,LOW);
   delay(5000); //delay 5S
 }
 else
 {
   digitalWrite(PIN_Motorla,LOW); //stop rotating
   digitalWrite(PIN_Motorlb,LOW);
 }
}
```

#### 7. Project result

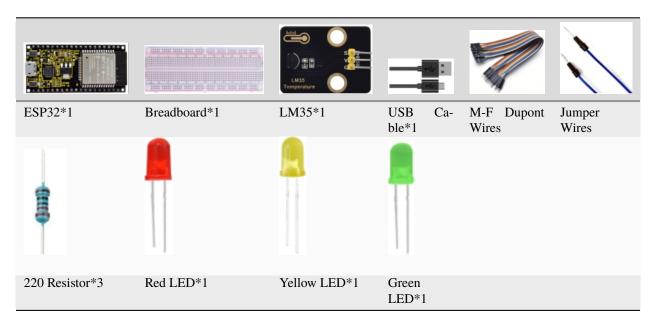
Compile and upload the code to ESP32, after the code is uploaded successfully, external power supply and power on. and then you will see that clap your hands to the sound sensor, and when the sound intensity exceeds a threshold, the small fan rotates; conversely, the small fan doesn't rotate.

### 5.28 Project 27Temperature Measurement

#### 1. Introduction

LM35 is a common used and easy-to-use temperature sensor. It doesn't require any other hardware and you only need an analog port. The difficulty lies in compiling the code and converting the analog values to Celsius temperature. In this project, we used a temperature sensor and 3 LEDs to make a temperature tester. When the temperature sensor touches different temperature objects, the LEDs will show different colors.

```
2. Components
```



3. Component knowledge

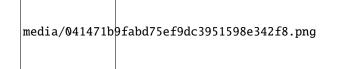


**Working principle of LM35 temperature sensor:** LM35 temperature sensor is a widely used temperature sensor with a variety of package types. At room temperature, it can achieve the accuracy of  $1/4^{\circ}$ C without additional calibration processing. LM35 temperature sensor can produce different voltage according to different temperatures, when the temperature is 0 °C, it output 0V; If increasing 1 °C, the output voltage will increase 10mv. The output temperature is 0 °C to 100°C, the conversion formula is as follows

$$V_{\text{out\_LM35}}(T) = 10 \,\text{mV}/_{\circ \text{C}} \times T^{\circ}\text{C}$$

4. Read the temperature value of LM35

We first use a simple code to read the value of the temperature sensor and printing them out, wiring diagram is shown below



LM35 output is given to analog pin GPIO36 of the ESP32, this analog voltage is converted to its digital form and processed to get the temperature reading.

You can open the code we provideIf you haven't downloaded the code file, please click on the link to download it:Download Arduino Codes

The code used in this project is saved in folder "Arduino-Codes\Project 27Temperature Measurement\Project\_27.1\_Read\_LM35\_Temperature\_Value".

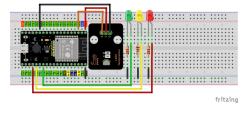
```
/*
* Filename
            : Read LM35 Temperature Value
* Description : ADC value is converted to LM35 temperature value
           : http//www.keyestudio.com
* Auther
*/
##define PIN_ANALOG_IN 36 //the pin of the Temperature Sensor
void setup() {
 Serial.begin(115200);
}
//In loop()the analogRead() function is used to obtain the ADC value,
//and then the map() function is used to convert the value into an 8-bit precision DAC.
\rightarrow value.
//Calculate the measured voltage value, Celsius and Fahrenheit values through the formula,
//and print these data through the serial port monitor.
void loop() {
 int adcVal = analogRead(PIN_ANALOG_IN);
 int dacVal = map(adcVal, 0, 4095, 0, 255);
 double voltage = adcVal / 4095.0 * 3.3;
 float temperatureC = (voltage * 1000.0) / 10.0 ;
 float temperatureF = (temperatureC * 1.8) + 32.0;
 Serial.print("ADC Value: " + String(adcVal));
 Serial.print("--DAC Value: " + String(dacVal));
 Serial.print("--Voltage Value: " + String(voltage) + "V");
 Serial.print("--temperatureC: " + String(temperatureC) + "C");
 Serial.println("--temperatureF: " + String(temperatureF) + "F");
 delay(200);
}
```

Compile and upload the code to ESP32, after the code is uploaded successfully, power up with a USB cable, open the serial monitor and set the baud rate to 115200. You will see that the serial port monitor window will print out the temperature values read by the LM35 temperature sensor. Hold the LM35 element by hand, the temperature value read by the LM35 temperature sensor will change.

0	COM3						_		×	(
									Send	
ADC	Value:	217DAC	Value:	13Voltage	Value:	0.17VtemperatureC:	17.49°Ctemperatu	reF: 63	.48F	^
ADC	Value:	213DAC	Value:	13Voltage	Value:	0.17VtemperatureC:	17.16°Ctemperatu	reF: 62	.90F	
ADC	Value:	216DAC	Value:	13Voltage	Value:	0.17VtemperatureC:	17.41°Ctemperatu	reF: 63	.33F	
ADC	Value:	216DAC	Value:	13Voltage	Value:	0.17VtemperatureC:	17.41°Ctemperatu	reF: 63	.33F	
ADC	Value:	216DAC	Value:	13Voltage	Value:	0.17VtemperatureC:	17.41°Ctemperatu	reF: 63	.33F	
ADC	Value:	224DAC	Value:	13Voltage	Value:	0.18VtemperatureC:	18.05°Ctemperatu	reF: 64	.49F	
ADC	Value:	217DAC	Value:	13Voltage	Value:	0.17VtemperatureC:	17.49°Ctemperatu	reF: 63	.48F	
ADC	Value:	216DAC	Value:	13Voltage	Value:	0.17VtemperatureC:	17.41°Ctemperatu	reF: 63	.33F	
ADC	Value:	218DAC	Value:	13Voltage	Value:	0.18VtemperatureC:	17.57°Ctemperatu	reF: 63	.62F	
ADC	Value:	224DAC	Value:	13Voltage	Value:	0.18VtemperatureC:	18.05°Ctemperatu	reF: 64	.49F	
ADC	Value:	218DAC	Value:	13Voltage	Value:	0.18VtemperatureC:	17.57°Ctemperatu	reF: 63	.62F	
ADC	Value:	216DAC	Value:	13Voltage	Value:	0.17VtemperatureC:	17.41°Ctemperatu	reF: 63	.33F	
ADC	Value:	218DAC	Value:	13Voltage	Value:	0.18VtemperatureC:	17.57°Ctemperatu	reF: 63	.62F	
ADC	Value:	208DAC	Value:	12Voltage	Value:	0.17VtemperatureC:	16.76°Ctemperatu	reF: 62	.17F	1
ADC	Value:	224DAC	Value:	13Voltage	Value:	0.18VtemperatureC:	18.05°Ctemperatu	reF: 64	.49F	
										Y
<b>⊻</b> !	Autoscrol	1 Show	timestam	p		Newline 🗸	$\sim$ 115200 baud $\sim$	Clear	output	

#### 5. Wiring diagram of the temperature measurement

Now we use a LM35 temperature sensor and three LED lights to do a temperature test. When the LM35 temperature sensor senses different temperatures, different LED lights will light up. Follow the diagram below for wiring.



#### 6. Project code

You can open the code we provideIf you haven't downloaded the code file, please click on the link to download it:Download Arduino Codes

The code used in this project is saved in folder "Arduino-Codes\Project 27Temperature Measurement\Project\_27.2\_Temperature\_Measurement".

(Note: The temperatureC threshold in the code can be reset itself as required.)

```
/*
* Filename
          : Temperature Measurement
* Description : Different leds light up when the LM35 senses different temperatures
* Auther
            : http//www.keyestudio.com
*/
##define PIN_ADC0
                    36
                         //the pin of the LM35 Sensor
##define PIN_GREENLED 4
                         //the pin of the Green led
                        //the pin of the Yellow led
##define PIN_YELLOWLED 2
##define PIN_REDLED 15
                         //the pin of the Red led
void setup() {
 Serial.begin(115200);
 pinMode(PIN_GREENLED, OUTPUT); //set PIN_GREENLED to OUTPUT
```

(continues on next page)

```
pinMode(PIN_YELLOWLED, OUTPUT);//set PIN_YELLOWLED to OUTPUT
 pinMode(PIN_REDLED, OUTPUT);//set PIN_REDLED to OUTPUT
 pinMode(PIN_ADC0, INPUT);//set PIN_ADC0 to INPUT
}
void loop() {
 int adcVal = analogRead(PIN_ADC0);
 double voltage = adcVal / 4095.0 * 3.3;
 float temperatureC = (voltage * 1000.0) / 10.0;
 float temperatureF = (temperatureC * 1.8) + 32.0;
 Serial.print("ADC Value: " + String(adcVal));
 Serial.print("---Voltage Value: " + String(voltage) + "V");
 Serial.print("---temperatureC: " + String(temperatureC) + "C");
 Serial.println("---temperatureF: " + String(temperatureF) + "F");
 if (temperatureC >= 25) {
   delay(100);
   digitalWrite(PIN_GREENLED, LOW);
   digitalWrite(PIN_YELLOWLED, LOW);
   digitalWrite(PIN_REDLED, HIGH);
 }
 else if (temperatureC >= 20 && temperatureC < 25) {</pre>
   digitalWrite(PIN_GREENLED, LOW);
   digitalWrite(PIN_YELLOWLED, HIGH);
   digitalWrite(PIN_REDLED, LOW);
 }
 else {
   digitalWrite(PIN_GREENLED, HIGH);
   digitalWrite(PIN_YELLOWLED, LOW);
   digitalWrite(PIN_REDLED, LOW);
 }
 delay(500);
}
  ******************
```

7. Project result

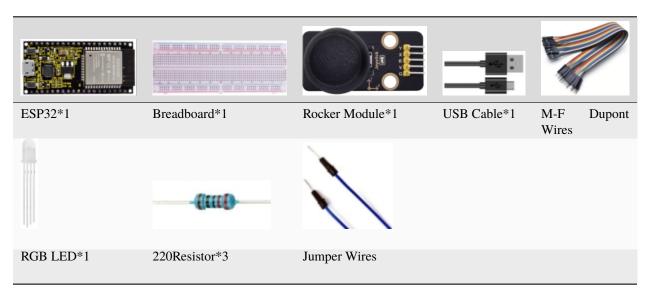
Compile and upload the code to ESP32, after the code is uploaded successfully, power up with a USB cable and you will see that the monitor displays the temperature values read by the LM35 temperature sensor. When the LM35 temperature sensor senses different temperatures, different LEDs will light up.

## 5.29 Project 28Rocker control light

1. Introduction

The rocker module is a component with two analog inputs and one digital input. It is widely used in areas such as game operationrobot control and drone control. In this project, we use ESP32 and a joystick module to control RGB, so that you can have a deeper understanding of the principle and operation of the joystick module in practice.

2. Components



3. Component knowledge



**Rocker module:** It mainly uses PS2 joystick components. In fact, the joystick module has 3 signal terminal pins, which simulate a three-dimensional space. The pins of the joystick module are GND, VCC, and signal terminals (B, X, Y). The signal terminals X and Y simulate the X-axis and Y-axis of the space. When controlling, the X and Y signal terminals of the module are connected to the analog port of the microcontroller. The signal terminal B simulates the Z axis of the space, it is generally connected to the digital port and used as a button.

VCC is connected to the microcontroller power output VCC (3.3V or 5V), GND is connected to the microcontroller GND, the voltage in the original state is about 1.65V or 2.5V. In the X-axis direction, when moving in the direction of the arrow, the voltage value increases, and the maximum voltage can be reached. Moving in the opposite direction of the arrow, the voltage value gradually decreases to the minimum voltage. In the Y-axis direction, the voltage value decreases gradually as it moves in the direction of the arrow on the module, decreasing to the minimum voltage. As the arrow is moved in the opposite direction, the voltage value increases and can reach the maximum voltage.

In the Z-axis direction, the signal terminal B is connected to the digital port and outputs 0 in the original state and outputs 1 when pressed. In this way, we can read the two analog values and the high and low level conditions of the digital port to determine the operating status of the joystick on the module.

#### Features:

Input VoltageDC 3.3V ~ 5V

Output SignalX/Y dual axis analog value +Z axis digital signal

[Range](javascript:;) [of](javascript:;) [Application](javascript:;)Suitable for control point coordinate movement in plane as well as control of two degrees of freedom steering gear, etc.

[product](javascript:;) [feature](javascript:;)sExquisite appearance, joystick feel superior, simple operation, sensitive response, long service life.

4. Read the value of the Rocker Module

We must use ESP32's analog IO port to read the value from the X/Y pin of the rocker module and use the digital IO port to read the digital signal of the button. Please connect the wires according to the wiring diagram below

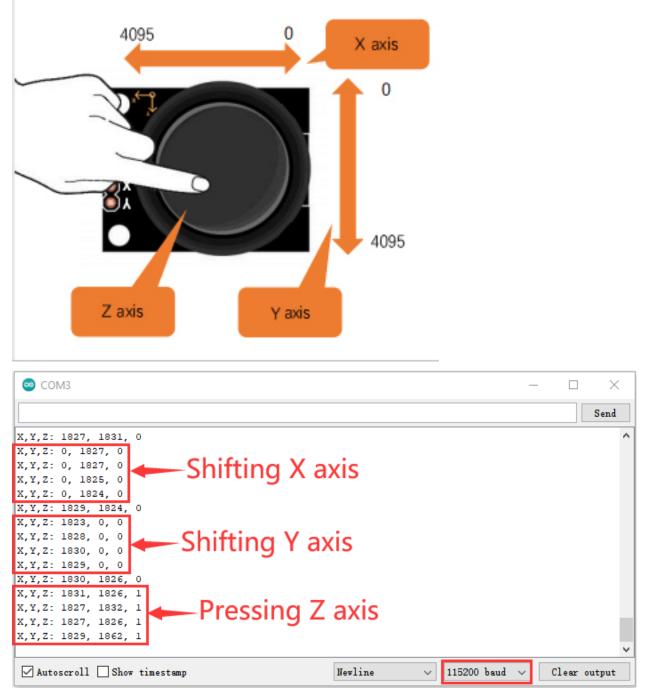
media/b611755eacc4c603e6c0555aced929cb.png

You can open the code we provideIf you haven't downloaded the code file, please click on the link to download it:Download Arduino Codes

The code used in this project is saved in folder\*\*"Arduino-Codes\Project 28Rocker control light\Project\_28.1\_Read\_Rocker\_Value"\*\*.

```
/*
* Filename : Read Rocker Value
* Description : Read data from Rocker.
         : http//www.keyestudio.com
* Auther
*/
int xyzPins[] = {36, 39, 14}; //x, y, z pins
void setup() {
 Serial.begin(115200);
 pinMode(xyzPins[0], INPUT); //x axis.
 pinMode(xyzPins[1], INPUT); //y axis.
 pinMode(xyzPins[2], INPUT_PULLUP); //z axis is a button.
}
// In loop(), use analogRead () to read the value of axes X and Y
//and use digitalRead () to read the value of axis Z, then display them.
void loop() {
 int xVal = analogRead(xyzPins[0]);
 int yVal = analogRead(xyzPins[1]);
 int zVal = digitalRead(xyzPins[2]);
 Serial.println("X,Y,Z: " + String(xVal) + ", " + String(yVal) + ", " + String(zVal));
 delay(500);
}
```

Compile and upload the code to ESP32, after the code is uploaded successfully, power up with a USB cable, open the serial monitor and set the baud rate to 115200. You will see that the serial port monitor window will print out the analog and digital values of the current joystick. Moving the joystick or pressing it will change the analog and digital values.



#### 5. Wiring diagram of Rocker control light

We just read the value of the rocker module, we need to do something with the rocker module and RGB here, Follow the diagram below for wiring

media/4ec49b488fedf216d03e49f83bc8443a.png

6. Project code

You can open the code we provideIf you haven't downloaded the code file, please click on the link to download it:Download Arduino Codes

The code used in this project is saved in folder "2. Windows System\\*\*\*\*2. C\_Tutorial\2. Projects\Project 28Rocker control lightProject\_28.2\_Rocker\_Control\_Light".

```
/*
* Filename
            : Rocker Control Light
* Description : Control RGB to light different colors by Rocker.
* Auther
          : http//www.keyestudio.com
*/
int x_Pin = 36;
                //x pin
int y_Pin = 39;
                //y pin
int z_Pin = 14; //z pin
int ledPins[] = {4, 0, 2}; //define red, green, blue led pins
const byte chns[] = {0, 1, 2};
                               //define the pwm channels
void setup() {
 pinMode(x_Pin, INPUT); //x axis.
 pinMode(y_Pin, INPUT); //y axis.
 pinMode(z_Pin, INPUT_PULLUP); //z axis is a button.
 for (int i = 0; i < 3; i++) { //setup the pwm channels,1KHz,8bit
   ledcSetup(chns[i], 1000, 8);
   ledcAttachPin(ledPins[i], chns[i]);
 }
}
// In loop(), use analogRead () to read the value of axes X and Y
//and use digitalRead () to read the value of axis Z, then display them.
void loop() {
 int xVal = analogRead(x_Pin);
 int yVal = analogRead(y_Pin);
 int zVal = digitalRead(z_Pin);
 if (xVal < 1000){
    ledcWrite(chns[0], 255); //Common cathode LED, high level to turn on the led.
    ledcWrite(chns[1], 0);
    ledcWrite(chns[2], 0);
  }
 else if (xVal > 3000){
    ledcWrite(chns[0], 0);
    ledcWrite(chns[1], 255);
    ledcWrite(chns[2], 0);
  }
 else if (yVal < 1000){
    ledcWrite(chns[0], 0);
    ledcWrite(chns[1], 0);
    ledcWrite(chns[2], 255);
  }
 else if (yVal > 3000){
    ledcWrite(chns[0], 255);
    ledcWrite(chns[1], 255);
```

(continues on next page)

ledcWrite(chns[2], 255);
}

7. Project result

Compile and upload the code to ESP32, after the code is uploaded successfully, power up with a USB cable and you will see that If the rocker is moved to the far left in the X direction, the RGB light turns red. If the rocker is moved to the far right in the X direction, the RGB light turns green. If the rocker is moved to the up in the Y direction, the RGB light turns white. If the rocker is moved to the down in the Y direction, the RGB light turns blue.

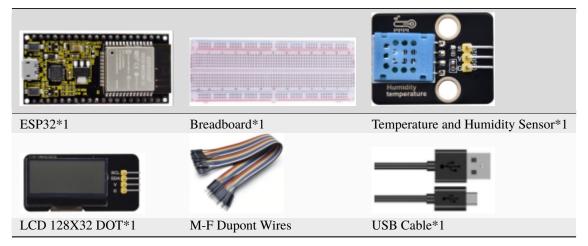
media/9c2d0d8777200827b16c49b752d45c4c.jpeg

# 5.30 Project 29Temperature Humidity Meter

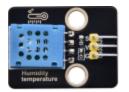
1. Introduction

In winter, the humidity in the air is very low, that is, the air is very dry, Coupled with cold, the skin of the human body is easy to be too dry and cracked, so you need to use a humidifier to increase the humidity of the air at home, but how do you know that the air is too dry? Then you need equipment to detect air humidity. In this Project, we will how to use the temperature and humidity sensor. We use the sensor to make a thermohygrometer, and also combined with a LCD 128X32 DOT to display the temperature and humidity values.

2. Components



3. Component knowledge



**Temperature and humidity sensor:** It is a temperature and humidity composite sensor with calibrated digital signal output, its precision humidity is $\pm$ 5%RH, temperature is $\pm$ 2°C, range humidity is 20 to 90%RH, and temperature is 0 to 50°C. The temperature and humidity sensor applies dedicated digital module acquisition technology and temperature and humidity sensing technology to ensure extremely high reliability and excellent long-term stability of the product. The temperature and humidity sensor includes a resistive-type humidity measurement and an NTC temperature measurement component, which is very suitable for temperature and humidity measurement applications where accuracy and real-time performance are not required.

The operating voltage is in the range of 3.3V to 5.5V.

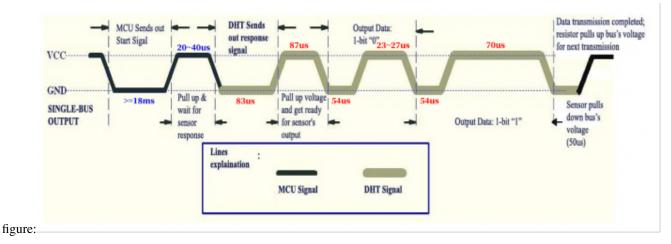
The temperature and humidity sensor has three pins, which are VCCGND and S. S is the pin for data output, using serial communication.

Single bus format definition of Temperature and Humidity Sensor

```
/*
* Filename
             : Temperature and Humidity Sensor
* Description : Use XHT11 to measure temperature and humidity.Print the result to the.
\leftrightarrow serial port.
* Auther
             : http//www.keyestudio.com
*/
##include "xht11.h"
//gpio13
xht11 xht(13);
unsigned char dht[4] = {0, 0, 0, 0; //Only the first 32 bits of data are received, not.
\rightarrow the parity bits
void setup() {
 Serial.begin(115200);//Start the serial port monitor and set baud rate to 115200
}
void loop() {
 if (xht.receive(dht)) { //Returns true when checked correctly
   Serial.print("RH:");
   Serial.print(dht[0]); //The integral part of humidity, DHT [0] is the fractional part
   Serial.print("% ");
   Serial.print("Temp:");
   Serial.print(dht[2]); //The integral part of temperature, DHT [3] is the fractional.
\rightarrow part
   Serial.println("C");
           //Read error
 } else {
   Serial.println("sensor error");
 }
 delay(1000); //It takes 1000ms to wait for the device to read
}
```

#### Data sequence diagram of Temperature and Humidity Sensor

When MCU sends a start signal, the Temperature and Humidity Sensor changes from the low-power-consumption mode to the high-speed mode, waiting for MCU completing the start signal. Once it is completed, the Temperature and Humidity Sensor sends a response signal of 40-bit data and triggers a signal acquisition. The signal is sent as shown in the



Combined with the code, you can understand better.

The XHT11 temperature and humidity sensor can easily add temperature and humidity data to your DIY electronic projects. It is perfect for remote weather stations, home environmental control systems, and farm or garden monitoring systems.

#### **Specification:**

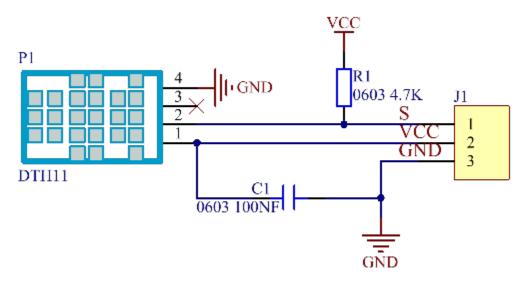
Working voltage: +5V

Temperature range: 0°C to 50°C, error of  $\pm$  2°C

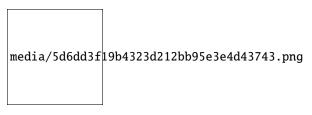
Humidity range: 20% to 90% RH, $\pm$  5% RH error

Digital interface

Schematic diagram of Temperature and Humidity Sensor:



4. Read temperature and humidity value



#### How to add the xht11 library

If you have not downloaded the library file, please click on the link to download it:Download Arduino Libraries

This code uses a library named "**xht11**", if you haven't installed it yet, please do so before learning. The steps to add third-party libraries are as follows:

Open the Arduino IDEclick"Sketch" $\rightarrow$ "Include Library" $\rightarrow$ "Add .zip Library...". In the pop-up window, find the file named "2 Windows System\2. C\_Tutorial\3.Libraries\xht11.ZIP" which locates in this directory. Select the xht11.ZIP file and then click"Open".

File Edit Sk	etch Tools Help							
$\odot \odot$	Verify/Compile	Ctrl+R					Ø	
	Upload	Ctrl+U	Δ					
Projec	Upload Using Programmer	Ctrl+Shift+U	Manage Libraries	Ctrl+Shift+I				
/****	Export compiled Binary	Ctrl+Alt+S	Add .ZIP Library		*****	****	*	ł
File	Show Sketch Folder	Ctrl+K	Add 21P Library					
Desc:	Include Library	;	Arduino libraries		resul	t to	the	
Auth	Add File		Bridge					
			Esplora					
	"xhtll.h"		Ethernet					
'gpiol3 ntll xht	(13):		Firmata					
	(20) /		GSM					
signed	char dht[4] = $\{0, 0, 0\}$	0, 0};//On	LiquidCrystal		recei	.ved,	not	
id setu			Mouse					
Serial.	begin(115200);//Star	Robot IR Remote		to 11	5200			
			SD					
id loop() {			Servo					
if (xht	.receive(dht)) { //R	eturns true	SpacebrewYun					
Seria	l.print("RH:");		TFT					

Look <u>i</u> n:	3. Libraries	s 🗸 🗸 💆	€ 💬 🛄 🕏
Q	ESP32Servo-0	.8.0	
	HT16K33_Lib_	_For_ESP32	
ecent Items	IRremoteESP8	3266-2.7.4	
	Keypad-3.1.1		
	LCD_128X32		
Desktop	MFRC522_12C		
	TM1650		
	Wire Wire		2
Documents	🔛 xht11 🔶	- 0	
	File <u>n</u> ame:	ning Kit Ultimate Edition\2. Windows System\2. C_Tutorial\3. Libra	aries Open
This PC	Files of type:	ZIP files or folders	Cancel

After the **xht11** library was added, You can open the code we provideIf you haven't downloaded the code file, please click on the link to download it:Download Arduino Codes

The code used in this project is saved in folder \*\*"Arduino-Codes\Project 29**Temperature Humidity Me**ter\Project\_29.1\_Detect\_Temperature\_Humidity".

```
/*
* Filename
           : Temperature Humidity Meter
* Description : LCD displays the value of temperature and humidity.
         : http//www.keyestudio.com
* Auther
*/
##include "xht11.h"
##include "lcd128_32_io.h"
//gpio13
xht11 xht(13);
unsigned char dht[4] = {0, 0, 0, 0}; //Only the first 32 bits of data are received, not
\rightarrow the parity bits
lcd lcd(21, 22); //Create lCD128 *32 pinsda->21 scl->22
void setup() {
 lcd.Init(); //initialize
 lcd.Clear(); //clear
}
char string[10];
//lcd displays humidity and temperature values
void loop() {
 if (xht.receive(dht)) { //Returns true when checked correctly
   }
 lcd.Cursor(0,0); //Set display position
 lcd.Display("Temper:"); //Setting the display
 lcd.Cursor(0,8);
 lcd.DisplayNum(dht[2]);
 lcd.Cursor(0,11);
```

(continues on next page)

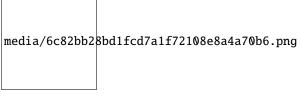
<pre>lcd.Display("C");</pre>
<pre>lcd.Cursor(2,0);</pre>
<pre>lcd.Display("humid:");</pre>
<pre>lcd.Cursor(2,8);</pre>
<pre>lcd.DisplayNum(dht[0]);</pre>
<pre>lcd.Cursor(2,11);</pre>
<pre>lcd.Display("%");</pre>
delay(200);
}
//*************************************

Compile and upload the code to ESP32, after the code is uploaded successfully, power up with a USB cable, and open the serial monitor and then set baud rate to 115200. You will see the current temperature and humidity value detected by the sensor from the serial monitor. As shown in the following figure:

💿 COM	3	-	$\Box$ $\times$
			Send
RH:65%	Temp:28C		^
RH:65%	Temp:28C		
RH:70%	Temp:28C		
RH:73%	Temp:28C		
RH:76%	Temp:28C		
RH:78%	Temp:28C		
RH:79%	Temp:28C		
RH:81%	Temp:28C		
RH:82%	Temp:28C		
RH:83%	Temp:28C		
RH:85%	Temp:29C		
RH:87%	Temp:29C		
RH:88%	Temp:29C		
RH:89%	Temp:29C		
RH:90%	Temp:29C		
			~
Autoso	roll 🗌 Show timestamp	Newline $\checkmark$ 115200 baud $\checkmark$	Clear output

5. Wiring diagram of the thermohygrometer

Now we start to print the values of the temperature and humidity sensor with LCD\_128X32\_DOT. We will see the corresponding values on the screen of LCD\_128X32\_DOT. Let's get started with this project. Please connect cables according to the following wiring diagram



6. Project code

The **xht11** and **lcd128\_32\_io** libraries have been added previously, so you don't need to add them again. If not, you need to add **xht11** and **lcd128\_32\_io** libraries. For the method of adding the **xht11** library, please refer to the above

method of this Project, and for the method of adding the lCD128\_32\_IO library, please refer to "Project 17: I2C 128 32 LCD".

After the **xht11** and **lcd128\_32\_io** libraries were added, You can open the code we provideIf you haven't downloaded the code file, please click on the link to download it:Download Arduino Codes

The code used in this project is saved in folder "Arduino-Codes\Project 29Temperature Humidity Meter\Project\_29.2\_Temperature\_Humidity\_Meter".

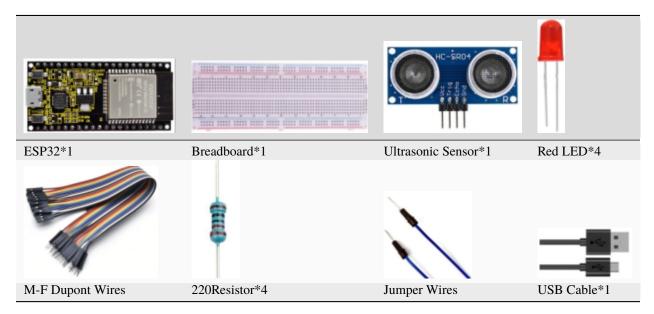
```
/*
* Filename
           : Temperature Humidity Meter
* Description : LCD displays the value of temperature and humidity.
          : http//www.keyestudio.com
* Auther
*/
##include "xht11.h"
##include "lcd128_32_io.h"
//gpio13
xht11 xht(13);
unsigned char dht[4] = {0, 0, 0, 0};//Only the first 32 bits of data are received, not
\rightarrow the parity bits
lcd lcd(21, 22); //Create lCD128 *32 pinsda->21 scl->22
void setup() {
 lcd.Init(); //initialize
 lcd.Clear(); //clear
}
char string[10];
//lcd displays humidity and temperature values
void loop() {
 if (xht.receive(dht)) { //Returns true when checked correctly
   }
 lcd.Cursor(0,0); //Set display position
 lcd.Display("Temper:"); //Setting the display
 lcd.Cursor(0,8);
 lcd.DisplayNum(dht[2]);
 lcd.Cursor(0,11);
 lcd.Display("C");
 lcd.Cursor(2,0);
 lcd.Display("humid:");
 lcd.Cursor(2,8);
 lcd.DisplayNum(dht[0]);
 lcd.Cursor(2,11);
 lcd.Display("%");
 delay(200);
}
```

#### 7.Project result

Compile and upload the code to ESP32, after the code is uploaded successfully, power up with a USB cable and you will see that the LCD 128X32 DOT will display temperature and humidity value in the current environment.

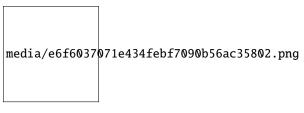
# 5.31 Project 30Ultrasonic Ranger

- 1. Introduction The HC-SR04 ultrasonic sensor is a very affordable distance sensor, mainly used for obstacle avoidance in various robotic projects. It is also used for water level sensing and even as a parking sensor. We treat the ultrasonic sensors as bat's eyes, in the dark, bats can still identify objects in front of them and directions through ultrasound. In this project, we use ESP32 to control a ultrasonic sensor and LEDs to simulate ultrasonic rangefinder.
- 2. Components



3. Component knowledge

**HC-SR04 Ultrasonic Sensor :** Like bats, sonar is used to determine the distance to an object. It provides accurate noncontact range detection, high-precision and stable readings. Its operation is not affected by sunlight or black materials, just like a precision camera (acoustically softer materials like cloth are difficult to detect). It has an ultrasonic transmitter and receiver.

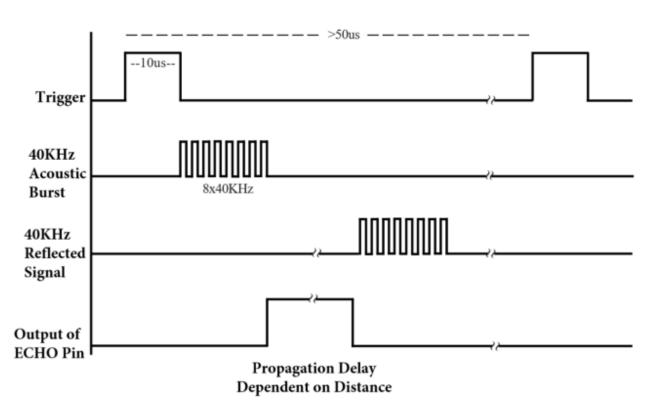


In front of the ultrasonic sensor are two metal cylinders, these are the converters. The converters convert the mechanical energy into an electrical signal. In the ultrasonic sensor, there are transmitting converters and receiving converters. The transmitting converter converts the electric signal into an ultrasonic pulse, and the receiving converter converts the reflected ultrasonic pulse back to an electric signal. If you look at the back of the ultrasonic sensor, you will see an IC behind the transmitting converter, which controls the transmitting converter. There is also an IC behind the receiving converter into a signal large enough to be transmitted to the Microcontroller.

#### Sequence diagrams:

The figure shows the sequence diagram of the HC-SR04. To start the measurement, the Trig of SR04 must receive at least 10us high pulse (5V), which will activate the sensor to emit 8 cycles of 40kHz ultrasonic pulses, and wait for the

reflected ultrasonic pulses. When the sensor detects ultrasound from the receiver, it sets the Echo pin to high (5V) and delays it by one cycle (width), proportional to the distance. To get the distance, measure the width of the Echo pin.



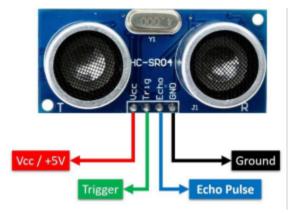
HC-SR04 ULTRASONIC MODULE

Time = Echo pulse width, its unit is "us" (microseconds) Distance(cm) = time/58

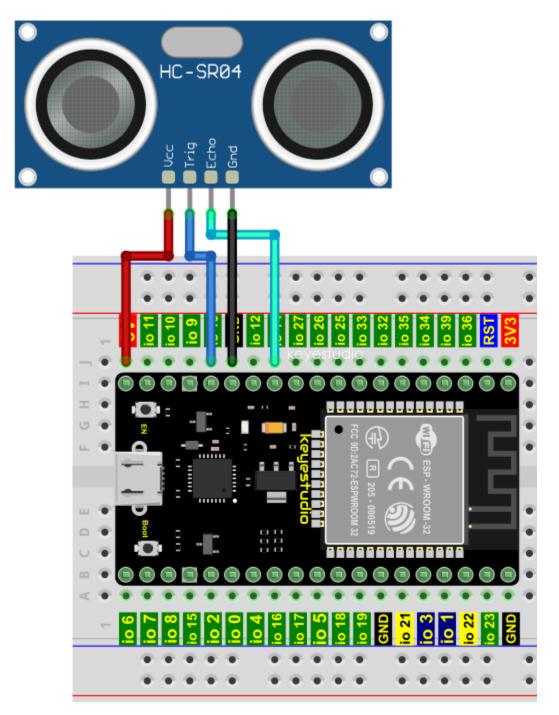
Distance(inch) = time/148

4. Read the distance value of the ultrasonic sensor:

We will start with a simple ultrasonic ranging and print the measured distance.



The HC-SR04 ultrasonic sensor has four pins, they are Vcc, Trig, Echo and GND. The Vcc pin provides the power source for generating ultrasonic pulses and is connected to Vcc (+5V). The GND pin is grounded. The Trig pin is where the Arduino sends a signal to start the ultrasonic pulse. The Echo pin is where the ultrasonic sensor sends information about the duration of the ultrasonic pulse to the Control board. Wiring as shown below:



You can open the code we provideIf you haven't downloaded the code file, please click on the link to download it:Download Arduino Codes

The code used in this project is saved in folder "Arduino-Codes\Project 30Ultrasonic Ranger\Project 30.1\_Ultrasonic\_Ranging".

(continues on next page)

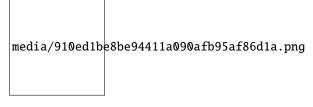
```
* Auther
              : http//www.keyestudio.com
*/
const int TrigPin = 13; // define TrigPin
const int EchoPin = 14; // define EchoPin.
int duration = 0; // Define the initial value of the duration to be 0
int distance = 0;//Define the initial value of the distance to be 0
void setup()
{
 pinMode(TrigPin , OUTPUT); // set trigPin to output mode
 pinMode(EchoPin , INPUT); // set echoPin to input mode
 Serial.begin(115200); // Open serial monitor at 115200 baud to see ping results.
}
void loop()
{
// make trigPin output high level lasting for 10s to triger HC_SR04
 digitalWrite(TrigPin , HIGH);
 delayMicroseconds(10);
 digitalWrite(TrigPin , LOW);
 // Wait HC-SR04 returning to the high level and measure out this waitting time
 duration = pulseIn(EchoPin , HIGH);
 // calculate the distance according to the time
 distance = (duration/2) / 28.5;
 Serial.print("Distance: ");
 Serial.print(distance); //Serial port print distance value
 Serial.println("cm");
 delay(300); // Wait 100ms between pings (about 20 pings/sec).
}
```

Compile and upload the code to ESP32, after the code is uploaded successfully, power up with a USB cable, open the serial monitor and set the baud rate to 115200. You will see that the serial port monitor window will print out the distance between the ultrasonic sensor and the object.

💿 сомз	- 🗆 X
	Send
Distance: 3cm	^
Distance: 3cm	
Distance: 3cm	
Distance: 4cm	
Distance: 5cm	
Distance: 6cm	
Distance: 6cm	
Distance: 7cm	
Distance: 8cm	
Distance: 8cm	
Distance: 9cm	
Distance: 9cm	
Distance: 10cm	
Distance: 10cm	
Distance: llcm	
	×
Autoscroll Show timestamp Newline	$\sim$ 115200 baud $\sim$ Clear output

5. Wiring diagram of the ultrasonic rangefinder

Next, we will use ESP32 to control an ultrasonic sensor and 4 LEDs to simulate ultrasonic rangefinder. Connect the line as shown below



6. Project code

You can open the code we provideIf you haven't downloaded the code file, please click on the link to download it:Download Arduino Codes

The code used in this project is saved in folder \*\*"2. Windows System\2. C\_Tutorial\2. Projects\Project 30Ultrasonic Ranger\Project\_30.2\_Ultrasonic\_Ranger".

```
/*
* Filename : Ultrasonic Ranger
* Description : four leds are controlled by ultrasonic ranging.
* Auther : http//www.keyestudio.com
*/
const int TrigPin = 13; // define TrigPin
const int EchoPin = 14; // define EchoPin.
const int PIN_LED1 = 4; // define PIN_LED1
const int PIN_LED2 = 0; // define PIN_LED2
const int PIN_LED3 = 2;
                        // define PIN_LED3
                        // define PIN_LED4
const int PIN_LED4 = 15;
int duration = 0; // define the initial value of the duration to be 0
int distance = 0; // define the initial value of the distance to be 0
void setup()
{
 pinMode(TrigPin , OUTPUT); // set trigPin to output mode
 pinMode(EchoPin , INPUT); // set echoPin to input mode
 pinMode(PIN_LED1 , OUTPUT); // set PIN_LED1 to output mode
 pinMode(PIN_LED2 , OUTPUT); // set PIN_LED2 to output mode
 pinMode(PIN_LED3 , OUTPUT); // set PIN_LED3 to output mode
 pinMode(PIN_LED4 , OUTPUT); // set PIN_LED4 to output mode
 Serial.begin(115200); // Open serial monitor at 115200 baud to see ping results.
}
void loop()
{
// make trigPin output high level lasting for 10s to triger HC_SR04
 digitalWrite(TrigPin , HIGH);
 delayMicroseconds(10);
 digitalWrite(TrigPin , LOW);
// Wait HC-SR04 returning to the high level and measure out this waitting time
 duration = pulseIn(EchoPin , HIGH);
// calculate the distance according to the time
 distance = (duration/2) / 28.5;
 Serial.print("Distance: ");
 Serial.print(distance); //Serial port print distance value
```

(continues on next page)

(continued from previous page)

```
Serial.println("cm");
 if ( distance <= 5 )
 {
   digitalWrite(PIN_LED1, HIGH);
 }
 else
 {
   digitalWrite(PIN_LED1, LOW);
 }
 if ( distance <= 10 )</pre>
 {
   digitalWrite(PIN_LED2, HIGH);
 }
 else
 {
   digitalWrite(PIN_LED2, LOW);
 }
 if ( distance <= 15 )</pre>
 {
   digitalWrite(PIN_LED3, HIGH);
 }
 else
 {
   digitalWrite(PIN_LED3, LOW);
 }
 if ( distance <= 20 )
 {
   digitalWrite(PIN_LED4, HIGH);
 }
 else
 {
   digitalWrite(PIN_LED4, LOW);
 }
}
```

7. Project result

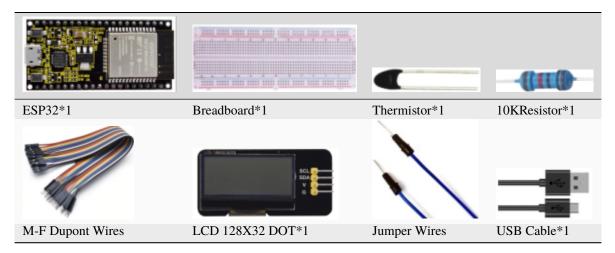
Compile and upload the code to ESP32, after the code is uploaded successfully, power up with a USB cable, open the serial monitor and set the baud rate to 115200. You will see that the serial port monitor window will print outthe distance between the ultrasonic sensor and the object, and the corresponding LED will light up when we move our hand in front of the ultrasonic sensor.

## 5.32 Project 31Temperature Instrument

### 1. Introduction

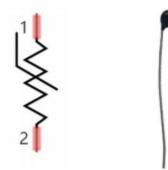
Thermistor is a kind of resistor whose resistance depends on temperature changes, which is widely used in gardening, home alarm system and other devices. Therefore, we can use the feature to make a temperature instrument.

2. Components



3. Component knowledge

**Thermistor:** A Thermistor is a temperature sensitive resistor. When it senses a change in temperature, the resistance of the Thermistor will change. We can take advantage of this characteristic by using a thermistor to detect temperature intensity. A Thermistor and its electronic symbol are shown below:



The relationship between resistance value and temperature of a thermistor is

 $Rt = R * EXP[B * \left(\frac{1}{T2} - \frac{1}{T1}\right)]$ 

### Where:

**Rt** is the thermistor resistance under T2 temperature;

**R** is the nominal resistance of thermistor under T1 temperature;

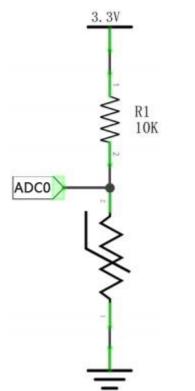
**EXP**[**n**] is nth power of e;

**B** is for thermal index;

**T1, T2** is Kelvin temperature (absolute temperature). Kelvin temperature=273.15 + Celsius temperature.

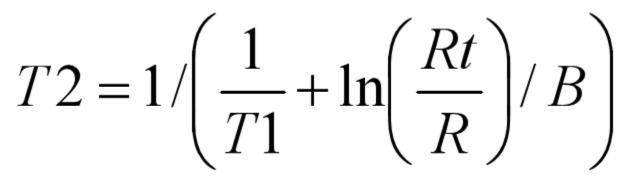
For the parameters of the Thermistor, we use: B=3950, R=10k, T1=25.

The circuit connection method of the Thermistor is similar to photoresistor, as the following



We can use the value measured by the ADC converter to obtain the resistance value of Thermistor, and then we can use the formula to obtain the temperature value.

Therefore, the temperature formula can be derived as:



4. Read the value of the Thermistor

First we will learn the thermistor to read the current ADC value, voltage value and temperature value and print them out. Please connect the wires according to the wiring diagram below

media/806fd81bf8a761b4ae1a638489c426ce.png

You can open the code we provideIf you haven't downloaded the code file, please click on the link to download it:Download Arduino Codes

The code used in this project is saved in folder "Arduino-Codes\Project 31Temperature Instrument\Project\_31.1\_Read\_the\_thermistor\_analog\_value".

```
/*
* Filename : Thermomter
* Description : Making a thermometer by thermistor.
* Auther : http//www.keyestudio.com
*/
##define PIN_ANALOG_IN
                    36
void setup() {
 Serial.begin(115200);
}
void loop() {
                                                    //read ADC pin
 int adcValue = analogRead(PIN_ANALOG_IN);
 double voltage = (float)adcValue / 4095.0 * 3.3;
                                                    // calculate voltage
 double Rt = 10 * voltage / (3.3 - voltage);
                                                     //calculate resistance...
\rightarrow value of thermistor
 double tempK = 1 / (1 / (273.15 + 25) + log(Rt / 10) / 3950.0); //calculate_
→temperature (Kelvin)
 double tempC = tempK - 273.15;
                                                     //calculate
→temperature (Celsius)
 Serial.printf("ADC value : %d,\tVoltage : %.2fV, \tTemperature : %.2fC\n", adcValue,_
→voltage, tempC);
 delay(1000);
}
```

Compile and upload the code to ESP32, after the code is uploaded successfully, power up with a USB cable, open the serial monitor and set the baud rate to 115200. You will see that the serial port monitor window prints out the thermistor's current ADC value, voltage value and temperature value. Try pinching the thermistor with your index finger and thumb (don't touch wires) for a while, and you will see the temperature increase.

💿 COM3			- 🗆 ×
			Send
ADC value : 1861,	Voltage : 1.50V,	Temperature : 29.17C	,
ADC value : 1870,	Voltage : 1.51V,	Temperature : 28.96C	
ADC value : 1872,	Voltage : 1.51V,	Temperature : 28.92C	
ADC value : 1871,	Voltage : 1.51V,	Temperature : 28.94C	
ADC value : 1876,	Voltage : 1.51V,	Temperature : 28.83C	
ADC value : 1874,	Voltage : 1.51V,	Temperature : 28.87C	
ADC value : 1875,	Voltage : 1.51V,	Temperature : 28.85C	
ADC value : 1875,	Voltage : 1.51V,	Temperature : 28.85C	
ADC value : 1887,	Voltage : 1.52V,	Temperature : 28.58C	
ADC value : 1909,	Voltage : 1.54V,	Temperature : 28.08C	
ADC value : 1851,	Voltage : 1.49V,	Temperature : 29.40C	
ADC value : 1879,	Voltage : 1.51V,	Temperature : 28.76C	
ADC value : 1883,	Voltage : 1.52V,	Temperature : 28.67C	
ADC value : 1862,	Voltage : 1.50V,	Temperature : 29.15C	
ADC value : 1810,	Voltage : 1.46V,	Temperature : 30.34C	
Autoscroll Show t	timestamp	Newline $\vee$ 115200 bat	ud v Clear output

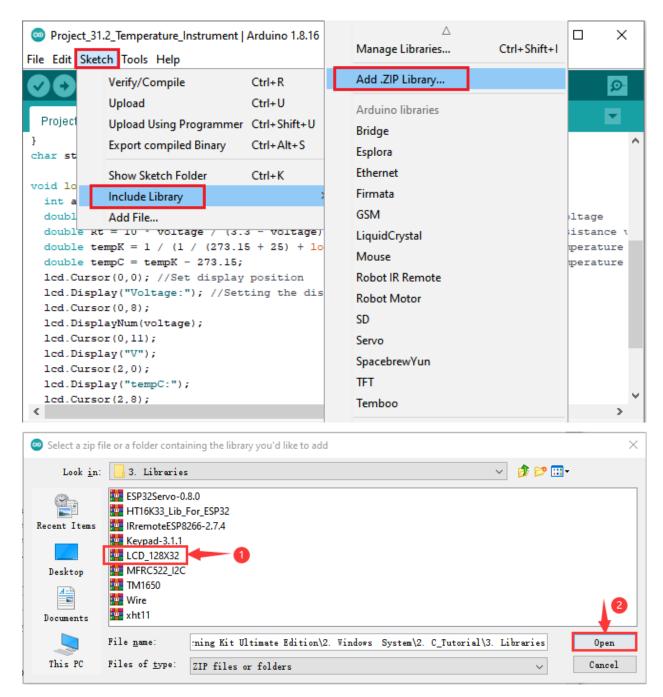
5. Wiring diagram of the temperature instrument

media/5a437bfdcad012211e15cab54e35dad7.png

#### 6. Adding the lcd128\_32\_io library

The **lcd128\_32\_io** library had been added previously, so you don't need to add it again. If not, you need to add the **lcd128\_32\_io** library. The steps to add third-party libraries are as follows:

Open the Arduino IDEclick "Sketch"  $\rightarrow$  "Include Library"  $\rightarrow$  "Add .ZIP Library...". In the pop-up window, find the file named "2. Windows System 2. C\_Tutorial 3. Libraries LCD\_128X32.ZIP" which locates in this directory. Select the LCD\_128X32.ZIP file and then click "Open".



#### 7. Project code

After the **lcd128\_32\_io** library was added, You can open the code we provideIf you haven't downloaded the code file, please click on the link to download it:Download Arduino Codes

The code used in this project is saved in folder "Arduino-Codes\Project 31Temperature Instrument\Project\_31.2\_Temperature\_Instrument".

```
//*
/*
* Filename : Temperature Instrument
* Description : LCD displays the temperature of thermistor.
```

(continues on next page)

```
(continued from previous page)
```

```
* Auther
              : http//www.keyestudio.com
*/
##include "lcd128_32_io.h"
##define PIN ANALOG IN
                       36
lcd lcd(21, 22); //Create lCD128 *32 pinsda->21 scl->22
void setup() {
 lcd.Init(); //initialize
 lcd.Clear(); //clear
}
char string[10];
void loop() {
 int adcValue = analogRead(PIN_ANALOG_IN);
                                                              //read ADC pin
 double voltage = (float)adcValue / 4095.0 * 3.3;
                                                              // calculate voltage
 double Rt = 10 * voltage / (3.3 - voltage);
                                                              //calculate resistance_
\rightarrow value of thermistor
 double tempK = 1 / (1 / (273.15 + 25) + log(Rt / 10) / 3950.0); //calculate_
→temperature (Kelvin)
 double tempC = tempK - 273.15;
                                                               //calculate.
→temperature (Celsius)
 lcd.Cursor(0,0); //Set display position
 lcd.Display("Voltage:"); //Setting the display
 lcd.Cursor(0,8);
 lcd.DisplayNum(voltage);
 lcd.Cursor(0,11);
 lcd.Display("V");
 lcd.Cursor(2, ◊);
 lcd.Display("tempC:");
 lcd.Cursor(2,8);
 lcd.DisplayNum(tempC);
 lcd.Cursor(2,11);
 lcd.Display("C");
 delay(200);
}
```

## 8. Project result

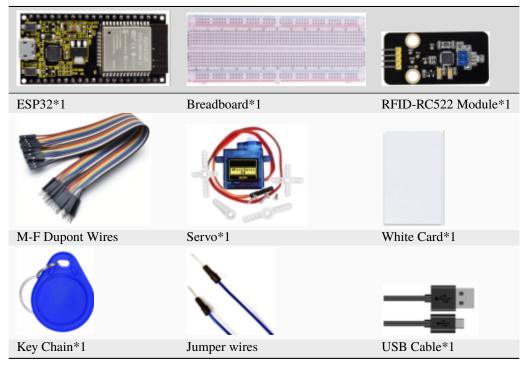
Compile and upload the code to ESP32, after the code is uploaded successfully, power up with a USB cable and you will see that the LCD 128X32 DOT displays the voltage value of the thermistor and the temperature value in the current environment.

## 5.33 Project 32RFID

## 1. Introduction

Nowadays, many residential districts use this function to open the door by swiping the card, which is very convenient. In this Project, we will learn how to use RFID(radio frequency identification) wireless communication technology and read and write the key chain card (white card) and control the steering gear rotation by RFID-MFRC522 module.

## 2. Components



3. Component knowledge

**RFID:** RFID (Radio Frequency Identification) is a wireless communication technology. A complete RFID system is generally composed of the responder and reader. Generally, we use tags as responders, and each tag has a unique code, which is attached to the object to identify the target object. The reader is a device for reading (or writing) tag information.

Products derived from RFID technology can be divided into three categories: passive RFID products, active RFID products and semi active RFID products. And Passive RFID products are the earliest, the most mature and most widely used products in the market among others. It can be seen everywhere in our daily life such as, the bus card, dining card, bank card, hotel access cards, etc., and all of these belong to close-range contact recognition. The main operating frequency of Passive RFID products are: 125KHZ (low frequency), 13.56MHZ (high frequency), 433MHZ (ultrahigh frequency) and 915MHZ (ultrahigh frequency). Active and semi active RFID products work at higher frequencies.

The RFID module we use is a passive RFID product with the operating frequency of 13.56MHz.

**RFID-RC522** Module The MFRC522 is a highly integrated reader/writer IC for contactless communication at 13.56MHz. The MFRC522's internal transmitter is able to drive a reader/writer antenna designed to communicate with ISO/IEC 14443 A/MIFARE cards and transponders without additional active circuitry. The receiver module provides a robust and efficient implementation for demodulating and decoding signals from ISO/IEC 14443 A/MIFARE compatible cards and transponders. The digital module manages the complete ISO/IEC 14443A framing and error detection(parity and CRC) functionality.

This RFID Module uses MFRC522 as the control chip and adopts I2C(Inter-Integrated Circuit) interface.



#### **Specifications:**

Operating voltage: DC 3.3V-5V

Operating current: 13—100mA/DC 5V

Idling current: 10-13mA/DC 5V

Sleep current: <80uA

Peak current: <100mA

Operating frequency: 13.56MHz

Maximum power: 0.5W

Supported card types: mifare1 S50, mifare1 S70, mifare UltraLight, mifare Pro, mifare Desfire.

Environmental operating temperature: -20 to 80 degrees Celsius.

Environment storage temperature: -40 to 85 degrees Celsius.

Relative Humidity: 5% to 95%.

Data transfer rate: The maximum is 10Mbit/s.

4. RFID Read UID

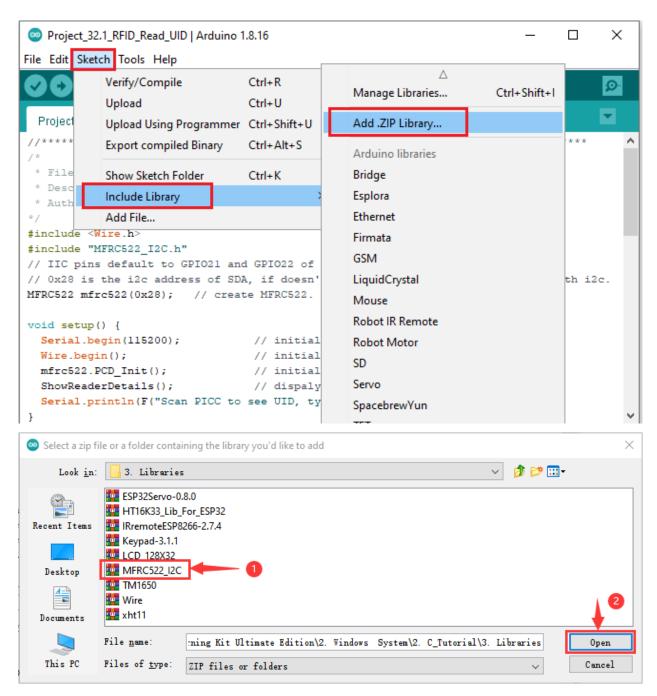
We will read the UNIQUE ID number (UID) of the RFID card and identify the type of the RFID card, and display the relevant information through the serial port. The wiring diagram is shown below

media/1cdb3ffd7f392f29451aeed5c3257133.png

## Adding the MFRC522\_I2C and Wire libraries

If you haven't installed the **MFRC522\_I2C** and **Wire** libraries yet, please do so before learning. The steps to add third-party libraries are as follows:

Open the Arduino IDEclick"Sketch" $\rightarrow$ "Include Library" $\rightarrow$ "Add .ZIP Library...". In the pop-up window, find the file named "2. Windows System\2. C\_Tutorial\3.Libraries\MFRC522\_I2C.ZIP" which locates in this directory. Select the MFRC522\_I2C.ZIPfile and then click"Open".



Then, in the pop-up window, find the file named \*\*\*\*\*2. Windows System\2. C\_Tutorial\3.Libraries\Wire.ZIP which locates in this directory. Select the Wire.Zip file and then click"Open".

🥯 Select a zip f	ile or a folder conta	ining the library you'd like to add	×
Look <u>i</u> n:	📙 3. Librarie	s 🗸 🕹 🖓 💭 🛄 •	,
	ESP32Servo-0	.8.0	
	HT16K33_Lib	For_ESP32	
Recent Items	IRremoteESP8	3266-2.7.4	
	🚾 Keypad-3.1.1		
	LCD_128X32		
Desktop	MFRC522_120		
4-9	TM1650	-	
	🛄 Wire 💶	- 0	2
Documents	wht11	-	
	File <u>n</u> ame:	ning Kit Ultimate Edition/2. Windows System/2. C_Tutorial/3. Libraries	Open
This PC	Files of <u>t</u> ype:	ZIP files or folders $\checkmark$	Cancel

After the **MFRC522\_I2C** and **Wire** libraries were added, You can open the code we provideIf you haven't downloaded the code file, please click on the link to download it:Download Arduino Codes

The code used in this project is saved in folder "Arduino-Codes\Project 32RFID\Project\_32.1\_RFID\_Read\_UID".

```
/*
* Filename : RFID
* Description : RFID reader UID
* Auther : http//www.keyestudio.com
*/
##include <Wire.h>
##include "MFRC522_I2C.h"
// IIC pins default to GPI021 and GPI022 of ESP32
// 0x28 is the i2c address of SDA, if doesn't matchplease check your address with i2c.
MFRC522 mfrc522(0x28); // create MFRC522.
void setup() {
                             // initialize and PC's serial communication
 Serial.begin(115200);
                             // initialize I2C
 Wire.begin();
                             // initialize MFRC522
 mfrc522.PCD_Init();
 ShowReaderDetails();
                             // dispaly PCD - MFRC522 read carder
 Serial.println(F("Scan PICC to see UID, type, and data blocks..."));
}
void loop() {
 //
 if ( ! mfrc522.PICC_IsNewCardPresent() || ! mfrc522.PICC_ReadCardSerial() ) {
   delay(50);
   return;
 }
 // select one of door cards. UID and SAK are mfrc522.uid.
 // save UID
 Serial.print(F("Card UID:"));
 for (byte i = 0; i < mfrc522.uid.size; i++) {
```

(continues on next page)

(continued from previous page)

```
Serial.print(mfrc522.uid.uidByte[i] < 0x10 ? " 0" : " ");</pre>
   Serial.print(mfrc522.uid.uidByte[i], HEX);
 }
 Serial.println();
}
void ShowReaderDetails() {
 // attain the MFRC522 software
 byte v = mfrc522.PCD_ReadRegister(mfrc522.VersionReg);
 Serial.print(F("MFRC522 Software Version: 0x"));
 Serial.print(v, HEX);
 if (v == 0x91)
   Serial.print(F(" = v1.0"));
 else if (v == 0x92)
   Serial.print(F(" = v2.0"));
 else
   Serial.print(F(" (unknown)"));
 Serial.println("");
 // when returning to 0x00 or 0xFF, may fail to transmit communication signals
 if ((v == 0x00) || (v == 0xFF)) {
   Serial.println(F("WARNING: Communication failure, is the MFRC522 properly connected?
'));
 }
}
```

Compile and upload the code to ESP32, after the code is uploaded successfully, power up with a USB cable, and open the serial monitor and then set baud rate to 115200. You will see that place the door card and key chain close to the module sensor area respectively, the serial monitor display the card number and key chain value respectively, as shown below:

💿 COM3		_	
			Send
Card UID: 93 AD F7 20			^
Card UID: 93 AD F7 20			
Card UID: 93 AD F7 20			
Card UID: 93 AD F7 20			
Card UID: 93 AD F7 20			
Card UID: 39 B6 46 C2			
Card UID: 39 B6 46 C2			
Card UID: 39 B6 46 C2			
Card UID: 39 B6 46 C2			
Card UID: 39 B6 46 C2			
Card UID: 39 B6 46 C2			
Card UID: 39 B6 46 C2			
Card UID: 39 B6 46 C2			
Card UID: 39 B6 46 C2			
Card UID: 39 B6 46 C2			
			~
Autoscroll 🗌 Show timestamp	Newline $\sim$	115200 baud $\ \lor$	Clear output

Note: the door card value and key chain value may be different for different RRFID -RC522 door cards and key chains.

5. Wiring diagram of the RFID MFRC522

Now we use the RFID -RC522 module, white card/key chain and Servo to simulate an intelligent access control system. When the white card/key chain close to the RFID -RC522 module induction area, the servo rotates. Wiring according to the figure below

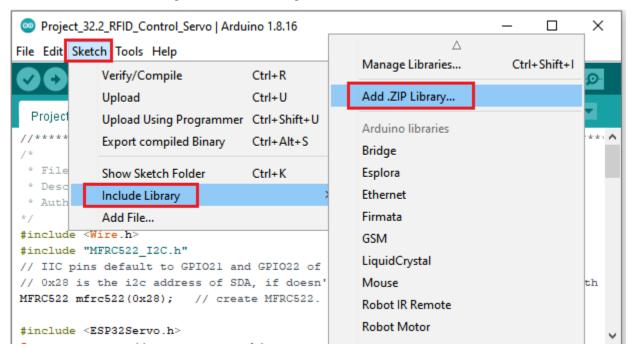
media/3ae6c0f1098d2aee34c51e7a96c25571.png

6. Adding the MFRC522\_I2C, Wire and ESP32Servo libraries

The MFRC522\_I2C, Wire and ESP32Servo libraries had been added previously, so you don't need to add it again. If not, you need to add the MFRC522\_I2C, Wire and ESP32Servo libraries. The steps to add third-party libraries are as follows:

First, add the ESP32Servo library

Open the Arduino IDEclick"Sketch" $\rightarrow$ "Include Library" $\rightarrow$ "Add .zip Library...". In the pop-up window, find the file named\*\*"2. Windows System\2. C\_Tutorial\3.Libraries\ ESP32Servo-0.8.0.Zip"\*\* which locates in this directory. Select the ESP32Servo-0.80.Zipfile and then click "Open".



Recent Items Recent Items Desktop Documents File name: ::ning Kit Ultimate Edition\2. Windows System\2. C_Tutorial\3. Libraries Open	Look <u>i</u> n:	3. Libraries		~	🤌 📂 🛄 •	
Recent Items IRremoteESP8266-2.7.4 Keypad-3.1.1 LCD_128X32 Desktop MFRC522_12C TM1650 Wire Documents xht11	9	ESP32Servo-0.8	0 🔶 🚺			
Image: Keypad-3.1.1           Image: LCD_128X32           Desktop           Image: MFRC522_I2C           Image: TM1650           Image: Wire           Image: Documents		HT16K33_Lib_F	r_ESP32			
Documents	Recent Items	IRremoteESP82	6-2.7.4			
Desktop MFRC522_I2C TM1650 Wire Documents xht11		Keypad-3.1.1				
Documents TM1650		LCD_128X32				
Documents Wire with the second secon	Desktop	MFRC522_12C				
Documents xht11		TM1650				
		🙀 Wire				2
File <u>n</u> ame: :::::::::::::::::::::::::::::::::::	Documents	🚾 xht11				
		File <u>n</u> ame:	ning Kit Ultimate Edition\2. Windows System\2	. C_Tutorial\3. Li	braries	Open

Then, add the MFRC522\_I2C and Wire libraries

If both of them are added, just skip this step.

7. Project code

After the **MFRC522\_I2CWire** and **ESP32Servo** libraries were added, You can open the code we provideIf you haven't downloaded the code file, please click on the link to download it:Download Arduino Codes

The code used in this project is saved in folder "Arduino-Codes\Project 32RFID\Project\_32.2\_RFID\_Control\_Servo".

**Note:** Different RFID-MFRC522 IC cards and keys have diverse values. You can substitute your own IC cards and keys values for the corresponding values read by the RFID-MFRC522 module in the program, otherwise the servo can't be controlled when uploading the test code to the ESP32.

```
if (rfid_str ==
                                                   "93adf720"
                                                                 || rfid_str == "39b646c2"
For example: You can replace the rfid_str of the
in the program code with your own IC cards and keys values read by the RFID-MFRC522 module.
/*
* Filename
             : RFID mfrc522 Control Servo
* Description : RFID controlled steering gear simulated door opening
* Auther
          : http//www.keyestudio.com
*/
##include <Wire.h>
##include "MFRC522_I2C.h"
// IIC pins default to GPI021 and GPI022 of ESP32
// 0x28 is the i2c address of SDA, if doesn't matchplease check your address with i2c.
MFRC522 mfrc522(0x28); // create MFRC522.
##include <ESP32Servo.h>
Servo myservo; // create servo object to control a servo
int servoPin = 15; // Servo motor pin
String rfid_str = "";
void setup() {
 Serial.begin(115200);
                                                                     (continues on next page)
```

```
(continued from previous page)
  Wire.begin();
  mfrc522.PCD_Init();
  ShowReaderDetails();
                                 // dispaly PCD - MFRC522 read carder
  Serial.println(F("Scan PICC to see UID, type, and data blocks..."));
  myservo.setPeriodHertz(50);
                                        // standard 50 hz servo
 myservo.attach(servoPin, 500, 2500); // attaches the servo on servoPin to the servo
→ object
 myservo.write(0);
  delay(500);
}
void loop() {
  if ( ! mfrc522.PICC_IsNewCardPresent() || ! mfrc522.PICC_ReadCardSerial() ) {
   delay(50);
   return:
 }
  // select one of door cards. UID and SAK are mfrc522.uid.
  // save UID
  rfid_str = ""; //String emptying
  Serial.print(F("Card UID:"));
  for (byte i = 0; i < mfrc522.uid.size; i++) {</pre>
   rfid_str = rfid_str + String(mfrc522.uid.uidByte[i], HEX); //Convert to string
   //Serial.print(mfrc522.uid.uidByte[i] < 0x10 ? " 0" : " ");</pre>
   //Serial.print(mfrc522.uid.uidByte[i], HEX);
  }
  Serial.println(rfid_str);
  if (rfid_str == "93adf720" || rfid_str == "39b646c2") {
   myservo.write(180);
   delay(500);
   Serial.println(" open the door!");
    }
}
void ShowReaderDetails() {
  // attain the MFRC522 software
 byte v = mfrc522.PCD_ReadRegister(mfrc522.VersionReg);
  Serial.print(F("MFRC522 Software Version: 0x"));
  Serial.print(v, HEX);
  if (v == 0x91)
    Serial.print(F(" = v1.0"));
  else if (v == 0x92)
   Serial.print(F(" = v2.0"));
  else
   Serial.print(F(" (unknown)"));
  Serial.println("");
  // when returning to 0x00 or 0xFF, may fail to transmit communication signals
  if ((v == 0x00) || (v == 0xFF)) \{
    Serial.println(F("WARNING: Communication failure, is the MFRC522 properly connected?
```

```
(continues on next page)
```

(continued from previous page)

}
}
//*************************************

#### 8. Project result

Compile and upload the code to ESP32, after the code is uploaded successfully, power up with a USB cable, and open the serial monitor and then set baud rate to 115200. You will see that when using the white card or a key card swiping, the serial port monitor displays white card or key card information and "open the door". As shown in the picture below, and the servo rotates to the corresponding angle to simulate opening the door.

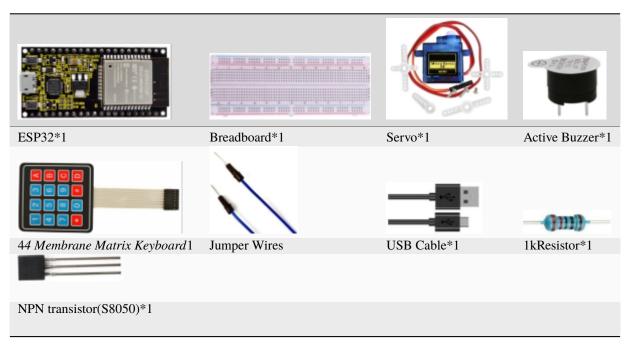
💿 СОМЗ	- 🗆 X
	Send
Card UID:93adf720	
open the door!	
Card UID:93adf720	
open the door!	
Card UID:93adf720	
open the door!	
Card UID:39b646c2	
open the door!	
Card UID:39b646c2	
open the door!	
Card UID:39b646c2	
open the door!	
Autoscroll Show timestamp Newlin	we $\sim$ 115200 baud $\sim$ Clear output

# 5.34 Project 33Keypad Door

1. Introduction

Commonly used digital button sensor, one button uses an IO port. However, it will occupy too many IO ports when we need a lot of buttons. In order to save the use of IO ports, the multiple buttons are made into a matrix type, through the control of the line and row to achieve less IO port control of multiple buttons. In this project, we will learn ESP32 and thin film 4\*4 matrix keyboard control a servo and a buzzer.

2. Components

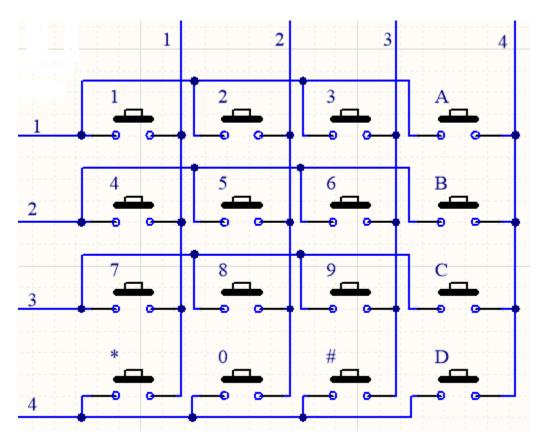


3. Component knowledge

**4\*4 Matrix keyboard** A Keypad Matrix is a device that integrates a number of keys in one package. As is shown below, a 4x4 Keypad Matrix integrates 16 keys:



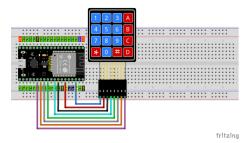
Similar to the integration of an LED Matrix, the 4x4 Keypad Matrix has each row of keys connected with one pin and this is the same for the columns. Such efficient connections reduce the number of processor ports required. The internal circuit of the Keypad Matrix is shown below.



The method of usage is similar to the Matrix LED, by using a row or column scanning method to detect the state of each key's position by column and row. Take column scanning method as an example, send low level to the first 4 column (Pin4), detect level state of row 1, 2, 3, 4 to judge whether the key A, B, C, D are pressed. Then send low level to column3, 2, 1 in turn to detect whether other keys are pressed. By this means, you can get the state of all of the keys.

4. Read the key value of the 4\*4 matrix keyboard

We start with a simple code to read the values of the 4\*4 matrix keyboard and print them in the serial monitor. Its wiring diagram is shown below



#### How to add the Keypad library

If you have not downloaded the library file, please click on the link to download it:Download Arduino Libraries

This code uses a library named "**Keypad**", if you haven't installed it yet, please do so before learning. The steps to add third-party libraries are as follows:

There are two ways to add libraries:

The first way, open the Arduino IDE, click "Sketch"  $\rightarrow$  "Include Library"  $\rightarrow$  "Manage Libraries...". Enter "Keypad" in the search box, select "Keypad" and click "Update" to install. Please refer to the following operations :

Droject 3	3.1_4x4_Matrix_Keypad_Dis	splay   Δrduino 1.81	16		_		×
	tch Tools Help	and Lengthing 1.01					~
		Ctrl+R					
	Verify/Compile						Ø
Project	Upload	Ctrl+U					
Project	Upload Using Programn			Δ		***	•
//****	Export compiled Binary	Ctrl+Alt+S		Manage Libraries	Ctrl+Shift+I		
* File	Show Sketch Folder	Ctrl+K					
* Desc * Auth	Include Library	;		Add .ZIP Library			
*/	Add File			Arduino libraries			
<pre>#include &lt;</pre>	Keypad.h>		1	Bridge			
// define	the symbols on the b	uttons of the		Esplora			
char keys[	[4] [4] = {			Ethernet			
	', '3', 'A'},			Firmata			
	5', '6', 'B'}, 5', '9', 'C'},			GSM			
	)', '#', 'D'}			LiquidCrystal			
};				Mouse			
byte rowPi	ns[4] = {22, 21, 19,	18}; // conne		Robot IR Remote			
-	ns[4] = {17, 16, 4,	-		Robot Motor		ıd	
				CD			~
🥯 Library Mana	ager						×
Type All	✓ Topic All	✓ Keypad	0				
by Gonçalo B		keypad	-				
		<b>eypads.</b> It includes v	rariou	us alphanumeric modes to proces	s text on phone-	like	^
More info							
Keypad							
Built-In by C	ommunity https://github.com						
				As of version 3.0 it now supports Hardware Abstraction. It improve			
by hiding the More info	e pinMode and digitalRead call	s for the user.					
Version 3.1	.0 v Install 2						
by Ben Arbla	actor						
A library for	I2C control of the LCD03 20x4			lules from Robot Electronics. It a			
				ards) , though some features of _CD03 including custom character			ed
the keypad. More info	Supports Arudino 1.0.0 and no	ewer.					~
							Close

The second wayopen the Arduino IDEclick "Sketch" $\rightarrow$ "Include Library" $\rightarrow$ "Add .ZIP Library...". In the pop-up window, find the file named\*\*"2. Windows System\\*\*\*2. C\_Tutorial\3.Libraries\\*\*\*\*Keypad-3.1.1.ZIP"\*\*which locates in this directory. Select the **Keypad-3.1.1.ZIP** fileand then click"Open".

	3.1_4x4_Matrix_Keypad_Dis	play   Arduino 1.8.1	6			_		×
00	Verify/Compile	Ctrl+R						ø
Project	Upload Upload Using Programm	Ctrl+U						
//****	Export compiled Binary			Δ			***	^
/* * File				Manage Libraries	Ct	rl+Shift+I		
* Desc	Show Sketch Folder	Ctrl+K	Г	Add .ZIP Library				
* Auth */	Add File	1		Arduino libraries				
#include <				Bridge				
// define t	the symbols on the b	uttons of the		Esplora				
char keys[4	4][4] = {			Ethernet				
	', '3', 'A'}, ', '6', 'B'},			Firmata				
	', '9', 'C'},			GSM				
};	', '#', 'D'}			LiquidCrystal				
hat a second	- (4) - (20 0) 10	101- (/		Mouse Robot IR Remote				
_	ns[4] = {22, 21, 19, ns[4] = {17, 16, 4,	-		Robot Motor			d	
				SD				~
Select a zip f	file or a folder containing the li	brary you'd like to add						×
Look <u>i</u> n:	_				~	🌶 📂 🖽	•	
Recent Items Desktop Documents	<ul> <li>ESP32Servo-0.8.0</li> <li>HT16K33_Lib_For_ESP32</li> <li>IRremoteESP8266-2.7.4</li> <li>Keypad-3.1.1</li> <li>LCD_128X32</li> <li>MFRC522_12C</li> <li>TM1650</li> <li>Wire</li> <li>xht11</li> </ul>	- 1						2
	File <u>n</u> ame: :ning Ki	t Ultimate Edition\2	. Wir	adows System\2. C_Tutor	ial\3. Li	braries	0	)pen
This PC	Files of type: ZIP file	es or folders				~	Ca	ancel

After the **Keypad** library is added, You can open the code we provideIf you haven't downloaded the code file, please click on the link to download it:Download Arduino Codes

The code used in this project is saved in folder \*\*"Arduino-Codes\Project 3 \*\*".

(continued from previous page)

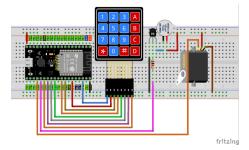
```
##include <Keypad.h>
// define the symbols on the buttons of the keypad
char keys[4][4] = \{
 {'1', '2', '3', 'A'},
 {'4', '5', '6', 'B'},
 {'7', '8', '9', 'C'},
 {'*', '0', '#', 'D'}
};
byte rowPins[4] = {22, 21, 19, 18}; // connect to the row pinouts of the keypad
byte colPins[4] = \{17, 16, 4, 0\}; // connect to the column pinouts of the keypad
// initialize an instance of class NewKeypad
Keypad myKeypad = Keypad(makeKeymap(keys), rowPins, colPins, 4, 4);
void setup() {
 Serial.begin(115200); // Initialize the serial port and set the baud rate to 115200
 Serial.println("ESP32 is ready!"); // Print the string "UNO is ready!"
}
void loop() {
 // Get the character input
 char keyPressed = myKeypad.getKey();
 // If there is a character input, sent it to the serial port
 if (keyPressed) {
   Serial.println(keyPressed);
 }
}
```

Compile and upload the code to ESP32, after the code is uploaded successfully, power up with a USB cable, and open the serial monitor and then set baud rate to 115200. You will see that press the keyboard and the serial port monitor prints the corresponding key value, as shown below.

💿 сомз	- 🗆 ×
	Send
1	^
2 3	
A	
4	
5	
e B	
7	
8	
9 C	
*	
0	
# D	U Contraction of the second
Autoscroll Show timestamp Newline	

### 5. Wiring diagram of the Keypad Door

In the last experiment, we have known the key values of the 4\*4 matrix keyboard. Next, we use it as the keyboard to control a servo and a buzzer.

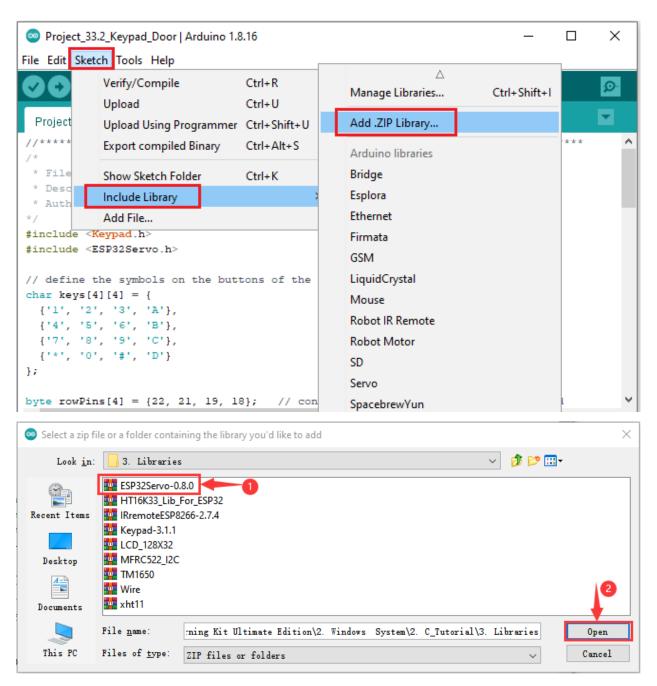


6. Adding the Keypad and ESP32Servo\*\* \*\*libraries

The **Keypad** and **ESP32Servo** libraries had been added previously, so you don't need to add them again. If not, you need to add **Keypad** and **ESP32Servo** libraries. The steps to add third-party Libraries are as follows:

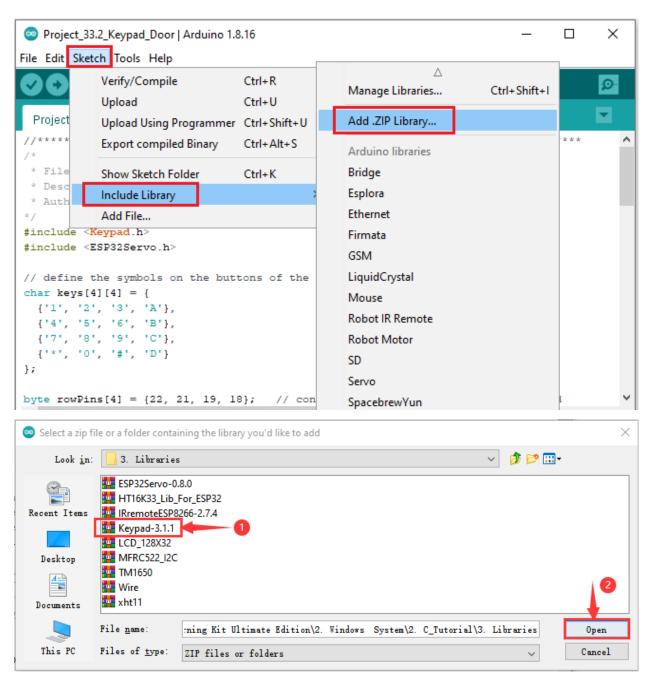
First, add the ESP32Servo library

Open the Arduino IDEclick"Sketch" $\rightarrow$ "Include Library" $\rightarrow$ "Add .ZIP Library...". In the pop-up window, find the file named "2. Windows System 2. C\_Tutorial 3.Libraries ESP32Servo-0.8.0.ZIP" which locates in this directory. Select the ESP32Servo-0.80.ZIP file and then click"Open".



Then, add the Keypad library

Open the Arduino IDEclick "Sketch" $\rightarrow$  "Include Library" $\rightarrow$  "Add .ZIP Library...". In the pop-up window, find the file named "2. Windows System\2. C\_Tutorial\3.Libraries\Keypad-3.1.1.ZIP" which locates in this directory. Select the Keypad-3.1.1.ZIP file and then click "Open".



7. Project code

After the **Keypad** and **ESP32Servo** libraries were added, You can open the code we provideIf you haven't downloaded the code file, please click on the link to download it:Download Arduino Codes

The code used in this project is saved in folder "2. Windows System\\*\*\*\*2. C\_Tutorial\2. Projects\Project 33Keypad Door\Project\_33.2\_Keypad\_Door".

\*/

(continued from previous page)

```
##include <Keypad.h>
##include <ESP32Servo.h>
// define the symbols on the buttons of the keypad
char keys[4][4] = \{
 {'1', '2', '3', 'A'},
 {'4', '5', '6', 'B'},
 {'7', '8', '9', 'C'},
 {'*', '0', '#', 'D'}
};
byte rowPins[4] = {22, 21, 19, 18}; // connect to the row pinouts of the keypad
byte colPins[4] = \{17, 16, 4, 0\}; // connect to the column pinouts of the keypad
// initialize an instance of class NewKeypad
Keypad myKeypad = Keypad(makeKeymap(keys), rowPins, colPins, 4, 4);
Servo myservo;
                 // Create servo object to control a servo
int servoPin = 15; // Define the servo pin
int buzzerPin = 2; // Define the buzzer pin
char passWord[] = {"1234"}; // Save the correct password
void setup() {
 myservo.setPeriodHertz(50);
                                      // standard 50 hz servo
 myservo.attach(servoPin, 500, 2500); // attaches the servo on servoPin to the servo.
→object
                                       // set the high level time range of the servo
→motor for an accurate 0 to 180 sweep
 myservo.write(0);
                                       // Set the starting position of the servo motor
 pinMode(buzzerPin, OUTPUT);
 Serial.begin(115200);
}
void loop() {
 static char keyIn[4]; // Save the input character
 static byte keyInNum = 0; // Save the the number of input characters
 char keyPressed = myKeypad.getKey(); // Get the character input
 // Handle the input characters
 if (keyPressed) {
   // Make a prompt tone each time press the key
   digitalWrite(buzzerPin, HIGH);
   delay(100);
   digitalWrite(buzzerPin, LOW);
   // Save the input characters
   keyIn[keyInNum++] = keyPressed;
   // Judge the correctness after input
   if (keyInNum == 4) {
     bool isRight = true;
                                    // Save password is correct or not
     for (int i = 0; i < 4; i++) { // Judge each character of the password is correct_
→or not
```

(continues on next page)

(continued from previous page)

```
if (keyIn[i] != passWord[i])
         isRight = false;
                                   // Mark wrong passageword if there is any wrong_
\hookrightarrow character.
     ł
     if (isRight) {
                                   // If the input password is right
       myservo.write(90);
                                  // Open the switch
                                   // Delay a period of time
       delay(2000);
                                  // Close the switch
       myservo.write(0);
       Serial.println("passWord right!");
     }
     else {
                                   // If the input password is wrong
       digitalWrite(buzzerPin, HIGH);// Make a wrong password prompt tone
       delay(1000);
       digitalWrite(buzzerPin, LOW);
       Serial.println("passWord error!");
     }
     keyInNum = 0; // Reset the number of the input characters to 0
   }
 }
}
```

#### 8. Project result

Compile and upload the code to ESP32, after the code is uploaded successfully, power up with a USB cable and you will see that press the keypad to input password with 4 characters. If the input is correct(Correct password :1234), the servo will move to a certain degree, and then return to the original position. If the input is wrong, an input error alarm will be generated.

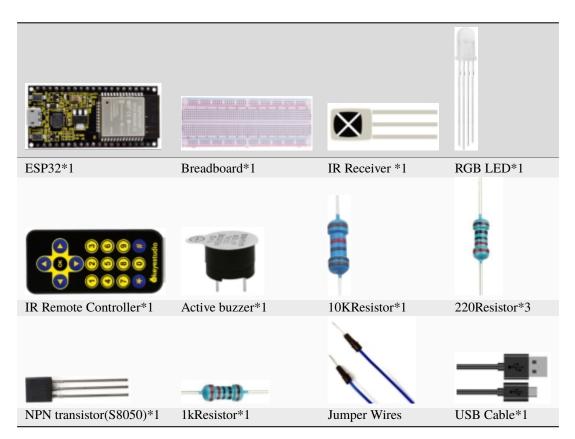
## 5.35 Project 34IR Control Sound and LED

#### 1. Introduction

An infrared(IR) remote control is a low-cost and easy-to-use wireless communication technology. IR light is very similar to visible light, except that its wavelength is slightly longer. This means that infrared rays cannot be detected by the human eye, which is perfect for wireless communication. For example, when you press a button on the TV remote control, an infrared LED will switch on and off repeatedly at a frequency of 38,000 times per second, transmitting information (such as volume or channel control) to the infrared sensor on the TV.

We'll start by explaining how common infrared communication protocols work. Then we will start the project with a remote control and an infrared receiver component.

2. Components



3. Component knowledge

\*\*Infrared Remote\*\*An infrared(IR) remote control is a device with a certain number of buttons. Pressing down different buttons will make the infrared emission tube, which is located in the front of the remote control, send infrared ray with different command. Infrared remote control technology is widely used in electronic products such as TVairconditioning, etc. Thus making it possible for you to switch TV programs and adjust the temperature of the air conditioning when away from them. The remote control we use is shown below:

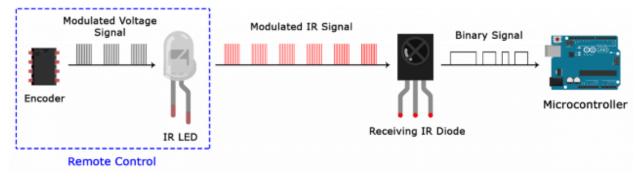
The infrared remote controller adopts NEC code and the signal cycle is 110ms.



**Infrared receiver** IR receiver is a component which can receive the infrared light, so we can use it to detect the signal emitted by the infrared remote control.

The infrared receiver demodules the received infrared signal and converts it back to binary, then passes the information to the microcontroller.

#### Infrared signal modulation process diagram



### **NEC Infrared communication protocol**

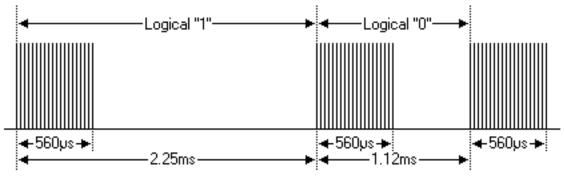
#### **NEC Protocol:**

To my knowledge the protocol I describe here was developed by NEC (Now Renesas). I've seen very similar protocol descriptions on the internet, and there the protocol is called Japanese Format. I do admit that I don't know exactly who developed it. What I do know is that it was used in my late VCR produced by Sanyo and was marketed under the name of Fisher. NEC manufactured the remote control IC. This description was taken from my VCR's service manual. Those were the days, when service manuals were filled with useful information!

#### **Features:**

- \* 8 bit address and 8 bit command length.
- \* Extended mode available, doubling the address size.
- \* Address and command are transmitted twice for reliability.
- \* Pulse distance modulation.
- \* Carrier frequency of 38kHz.
- \* Bit time of 1.125ms or 2.25ms.

#### Modulation:



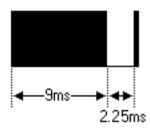
The NEC protocol uses pulse distance encoding of the bits. Each pulse is a 560µs long 38kHz carrier burst (about 21 cycles). A logical "1" takes 2.25ms to transmit, while a logical "0" is only half of that, being 1.125ms. The recommended carrier duty-cycle is 1/4 or 1/3.

#### **Protocol:**

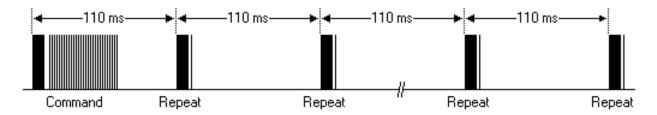


The picture above shows a typical pulse train of the NEC protocol. With this protocol the LSB is transmitted first. In this case Address 59andCommand16 is transmitted. A message is started by a 9ms AGC burst, which was used to set the gain of the earlier IR receivers. This AGC burst is then followed by a 4.5ms space, which is then followed by the Address and Command. Address and Command are transmitted twice. The second time all bits are inverted and can be used for verification of the received message. The total transmission time is constant because every bit is repeated with its inverted length. If you're not interested in this reliability you can ignore the inverted values, or you can expand the Address and Command to 16 bits each!

Keep in mind that one extra  $560\mu$ s burst has to follow at the end of the message in order to be able to determine the value of the last bit.



A command is transmitted only once, even when the key on the remote control remains pressed. Every 110ms a repeat code is transmitted for as long as the key remains down. This repeat code is simply a 9ms AGC pulse followed by a 2.25ms space and a 560µs burst.

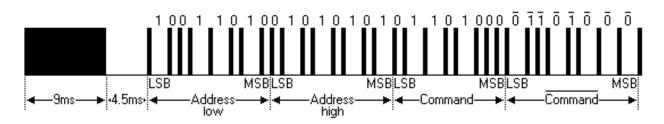


#### **Extended NEC protocol:**

The NEC protocol is so widely used that soon all possible addresses were used up. By sacrificing the address redundancy the address range was extended from 256 possible values to approximately 65000 different values. This way the address range was extended from 8 bits to 16 bits without changing any other property of the protocol.

By extending the address range this way the total message time is no longer constant. It now depends on the total number of 1's and 0's in the message. If you want to keep the total message time constant you'll have to make sure the number 1's in the address field is 8 (it automatically means that the number of 0's is also 8). This will reduce the maximum number of different addresses to just about 13000.

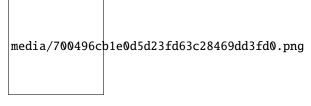
The command redundancy is still preserved. Therefore each address can still handle 256 different commands.



Keep in mind that 256 address values of the extended protocol are invalid because they are in fact normal NEC protocol addresses. Whenever the low byte is the exact inverse of the high byte it is not a valid extended address.

4. Decoded infrared signal

We connect the infrared receiving element to the ESP32, according to the wiring diagram below:



### How to add the IRremoteESP8266 library

If you have not downloaded the library file, please click on the link to download it:Download Arduino Libraries

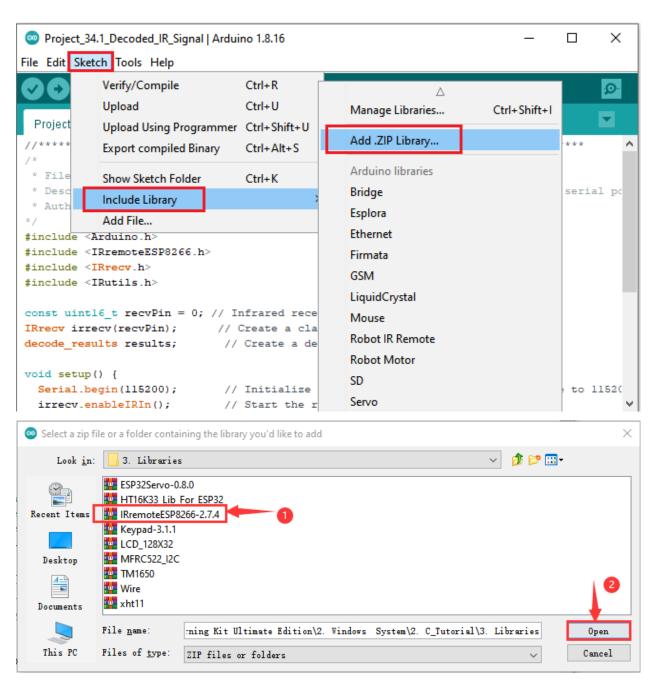
This code uses a library named "**IRremoteESP8266**", if you haven't installed it yet, please do so before learning. The steps to add third-party libraries are as follows:

There are two ways to add libraries:

The first way, open the Arduino IDE, click"Sketch" $\rightarrow$ "Include Library" $\rightarrow$ "Manage Libraries...". Enter"IRremoteESP8266" in the search box, select"IRremoteESP8266" and click"Update" to install. Please refer to the following operations :

	.1_Decoded_IR_Signal   Ardui	no 1.8.16			_		×
	ch Tools Help Verify/Compile	Ctrl+R	Г	∆ Manage Libraries	Ctrl+Shift+I		Ø
Project	Upload Upload Using Programmer			Add .ZIP Library		***	•
/* * File	Export compiled Binary Show Sketch Folder	Ctrl+Alt+S Ctrl+K		Arduino libraries Bridge			
* Desc * Auth	Include Library	;		Esplora Ethernet		seria	lpc
*/ #include <#	Add File Arduino.h> (RremoteESP8266.h>			Firmata			
<pre>#include <i #include="" <i<="" pre=""></i></pre>	[Rrecv.h>			GSM LiquidCrystal			
1	<pre>L6_t recvPin = 0; // In ecv(recvPin); // ()</pre>	nfrared rece Create a cla		Mouse Robot IR Remote			
_		Create a de		Robot Motor SD			
1	egin(115200); //	Initialize Start the r		Servo SpacebrewYun		to 1	152(
💿 Library Manag	ger						×
Type All	∨ Topic All	∨ IRremoteESF	8266				
INSTALLED Send and rece	avid Conran, Mark Szabo, Sebasti eive infrared signals with multipl als on an ESP8266 or an ESP32.			·	u to send and rece	1	^

The second wayopen the "Sketch"  $\rightarrow$  "Include Library"  $\rightarrow$  "Add .ZIP Library...". In the pop-up window, find the file named\*\*"2. Windows System 2. C\_Tutorial 3.Libraries IR remote ESP8266-2.7.4.ZIP"\*\* which locates in this directory. Select the **IR remote ESP8266-2.7.4.ZIP** fileand then click "Open".



After the **IRremoteESP8266** library is added, You can open the code we provideIf you haven't downloaded the code file, please click on the link to download it:Download Arduino Codes

The code used in this project is saved in folder "Arduino-Codes\Project 34IR Control Sound And LED\Project\_34.1\_Decoded\_IR\_Signal".

/\* \* Filename : Decoded IR Signal \* Description : Decode the infrared remote control and print it out through the serial.  $\rightarrow$  port. \* Auther : http//www.keyestudio.com

(continues on next page)

(continued from previous page)

```
*/
##include <Arduino.h>
##include <IRremoteESP8266.h>
##include <IRrecv.h>
##include <IRutils.h>
const uint16_t recvPin = 0; // Infrared receiving pin
IRrecv irrecv(recvPin); // Create a class object used to receive class
decode_results results;
                         // Create a decoding results class object
void setup() {
 Serial.begin(115200);
                     // Initialize the serial port and set the baud rate to.
→115200
 irrecv.enableIRIn();
                     // Start the receiver
 Serial.print("IRrecvDemo is now running and waiting for IR message on Pin ");
 Serial.println(recvPin); //print the infrared receiving pin
}
void loop() {
                                   // Waiting for decoding
 if (irrecv.decode(&results)) {
   serialPrintUint64(results.value, HEX);// Print out the decoded results
   Serial.println("");
   irrecv.resume();
                                    // Release the IRremote. Receive the next value
 }
 delay(1000);
}
```

Compile and upload the code to ESP32, after the code is uploaded successfully, power up with a USB cable, and open the serial monitor and then set baud rate to 115200. You will see that aim the infrared remote control transmitter at the infrared receiving head, press the button on the infrared controller, and the serial port monitor prints the current received key code values.

COM23					×
					Send
FF629D					^
FF22DD					
FFA857					
FFC23D					
FF02FD					
FF6897					
FF9867					
FFB04F					
FF30CF					
FF18E7					
FF7A85					
FF10EF					
FF38C7					
FF5AA5					
FF42BD					
FF4AB5					
FF52AD					~
Autoscroll 🗌 Show timestamp	Newline	∨ 115200 b	aud $\sim$	Clea	r output

Write down the code associated with each button, because you will need that information later.



5. Wiring diagram of the infrared remote control

media/4912c1622e0eaedb76ea3a9b8ed969c0.png

6. Project code

The **IRremoteESP8266** library had been added previously, so you don't need to add it again. If not, you need to add the **IRremoteESP8266** library. The steps to add third-party libraries are as shown above.

After the **IRremoteESP8266** library was added, You can open the code we provideIf you haven't downloaded the code file, please click on the link to download it:Download Arduino Codes

The code used in this project is saved in folder "Arduino-Codes\Project 34IR Control Sound And LED\Project\_34.2\_IR\_Control\_Sound\_And\_LED".

```
/*
* Filename : IR Control Sound And LED
* Description : Remote control RGB and Passive buzzer with infrared remote control.
* Auther : http//www.keyestudio.com
*/
##include <Arduino.h>
##include <IRremoteESP8266.h>
##include <IRrecv.h>
##include <IRutils.h>
const uint16_t recvPin = 0; // Infrared receiving pin
IRrecv irrecv(recvPin); // Create a class object used to receive class
decode_results results; // Create a decoding results class object
int ledPins[] = {22, 21, 4}; //define red, green, blue led pins
                                  //define the pwm channels
const byte chns[] = \{0, 1, 2\};
int buzzerPin = 15; // the number of the buzzer pin
void setup() {
 irrecv.enableIRIn();
                                    // Start the receiver
 pinMode(buzzerPin, OUTPUT);
 for (int i = 0; i < 3; i++) { //setup the pwm channels,1KHz,8bit</pre>
   ledcSetup(chns[i], 1000, 8);
   ledcAttachPin(ledPins[i], chns[i]);
 }
}
void loop() {
 if(irrecv.decode(&results)) { // Waiting for decoding
                                  // Handle the commands from remote control
   handleControl(results.value);
   irrecv.resume();
                                    // Receive the next value
 }
}
void handleControl(unsigned long value) {
 // Make a sound when it rereives commands
```

(continues on next page)

```
(continued from previous page)
```

```
digitalWrite(buzzerPin, HIGH);
delay(100);
digitalWrite(buzzerPin, LOW);
// Handle the commands
if (value == 0xFF6897) // Receive the number '1'
{
   ledcWrite(chns[0], 255); //Common cathode LED, high level to turn on the led.
   ledcWrite(chns[1], 0);
   ledcWrite(chns[2], 0);
    delay(1000);
}
else if (value == 0xFF9867) // Receive the number '2'
 {
   ledcWrite(chns[0], 0);
   ledcWrite(chns[1], 255);
   ledcWrite(chns[2], 0);
   delay(1000);
}
 else if (value == 0xFFB04F) // Receive the number '3'
 {
   ledcWrite(chns[0], 0);
   ledcWrite(chns[1], 0);
   ledcWrite(chns[2], 255);
   delay(1000);
 }
 else if (value == 0xFF30CF) // Receive the number '4'
 {
   ledcWrite(chns[0], 255);
   ledcWrite(chns[1], 255);
   ledcWrite(chns[2], 0);
   delay(1000);
 }
 else if (value == 0xFF18E7) // Receive the number '5'
{
   ledcWrite(chns[0], 255);
   ledcWrite(chns[1], 0);
   ledcWrite(chns[2], 255);
   delay(1000);
 }
 else if (value == 0xFF7A85) // Receive the number '6'
{
   ledcWrite(chns[0], 0);
   ledcWrite(chns[1], 255);
   ledcWrite(chns[2], 255);
   delay(1000);
 }
 else if (value == 0xFF10EF) // Receive the number '7'
 {
   ledcWrite(chns[0], 255);
    ledcWrite(chns[1], 255);
    ledcWrite(chns[2], 255);
    delay(1000);
```

(continues on next page)

### 7. Project result

Compile and upload the code to ESP32, after the code is uploaded successfully, power up with a USB cable and you will see that press the 1 to 7 key of the infrared remote controller, the buzzer will sound once, and the RGB light will be red, green, blue, yellow, red, blue, green and white respectively. Press another key (except 1 to 7 key), and the RGB light will go off.

(Note: Before use, we need to remove the plastic sheet from the bottom of the infrared remote controller.)

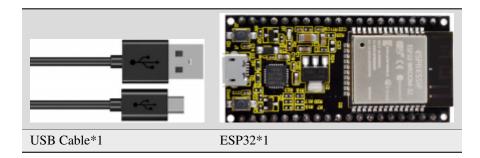


# 5.36 Project 35Bluetooth

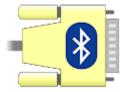
This chapter mainly introduces how to make simple data transmission through bluetooth of ESP32 and mobile phones. Project 35.1 is Classic Bluetooth and Project 35.2 is Bluetooth Control LED.

# **Project 35.1Classic Bluetooth**

1. Components



In this tutorial we need to use a Bluetooth APP called Serial Bluetooth Terminal to assist in the experiment. If you've not installed it yet, please do so by clicking: https://www.appsapk.com/serial-bluetooth-terminal/ .The following is its logo.



#### 2. Component knowledge

ESP32's integrated Bluetooth function Bluetooth is a short-distance communication system, which can be divided into two types, namely Bluetooth Low Energy(BLE) and Classic Bluetooth. There are two modes for simple data transmission: master mode and slave mode.

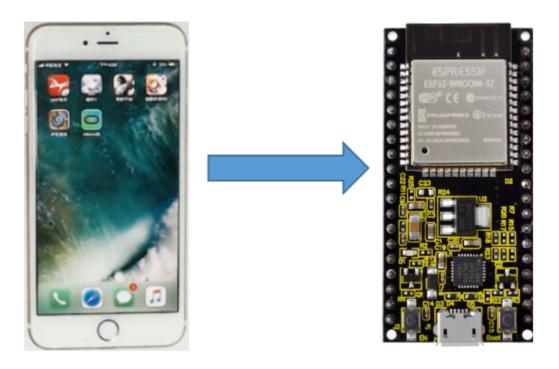
#### Master mode

In this mode, works are done in the master device and it can connect with a slave device. And we can search and select slave devices nearby to connect with. When a device initiates connection request in master mode, it requires information of the other Bluetooth devices including their address and pairing passkey. After finishing pairing, it can connect with them directly.

#### Slave mode

The Bluetooth module in slave mode can only accept connection request from a host computer, but cannot initiate a connection request. After connecting with a host device, it can send data to or receive from the host device.

Bluetooth devices can make data interaction with each other, as one is in master mode and the other in slave mode. When they are making datainteraction, the Bluetooth device in master mode searches and selects devices nearby to connect to. When establishing connection, they can exchange data. When mobile phones exchange data with ESP32, they are usually in master mode and ESP32 in slave mode.



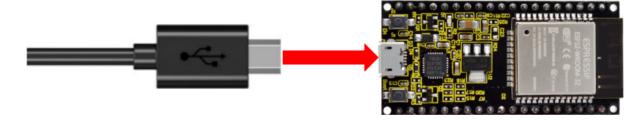
# Master



# **Master Slave**

3. Wiring Diagram

We can use a USB cable to connect ESP32 mainboard to the USB port on a computer.



4. Project code

You can open the code we provideIf you haven't downloaded the code file, please click on the link to download it:Download Arduino Codes

The code used in this project is saved in folder(path:) "Arduino-Codes\Project 35: Blue-tooth\Project\_35.1\_Classic\_Bluetooth".

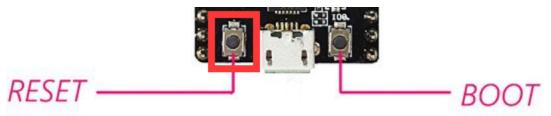
```
Project 35.1 Classic Bluetooth | Arduino 1.8.16
                                                                               \times
                                                                         File Edit Sketch Tools Help
               +
        ÷
    ÷)
                                                                              Ø
  Project_35.1_Classic_Bluetooth
 1*
  * Filename : SerialToSerialBT
  * Description : ESP32 communicates with the phone by bluetooth and print phone's data via
  * Auther
            : http//www.keyestudio.com
 */
 #include "BluetoothSerial.h"
 BluetoothSerial SerialBT;
 String buffer;
 void setup() {
  Serial.begin(115200);
  SerialBT.begin("ESP32test"); //Bluetooth device name
  Serial.println("\nThe device started, now you can pair it with bluetooth!");
 }
 void loop() {
 <
                                                                               >
 Done Saving.
Invalid library found in C:\Users\Administrator\Desktop\Arduino\hardware\espressif\esp32\1
Invalid library found in C:\Users\Administrator\Desktop\Arduino\hardware\espressif\esp32\1
<
                                                                               >
                                      ESP32 Wrover Module, Default, QIO, 80MHz, 921600, None on COM3
/*
* Filename : Classic Bluetooth--SerialToSerialBT
* Description : ESP32 communicates with the phone by bluetooth and print phone's data.
→via a serial port
* Auther : http//www.keyestudio.com
*/
##include "BluetoothSerial.h"
BluetoothSerial SerialBT;
String buffer;
void setup() {
 Serial.begin(115200):
 SerialBT.begin("ESP32test"); //Bluetooth device name
 Serial.println("\nThe device started, now you can pair it with bluetooth!");
}
void loop() {
 if (Serial.available()) {
   SerialBT.write(Serial.read());
```

(continues on next page)

	}
	<pre>if (SerialBT.available()) {</pre>
	<pre>Serial.write(SerialBT.read());</pre>
	}
	delay(20);
	}
,	//*************************************

#### 6. Project result:

Compile and upload the code to the ESP32. After uploading successfullywe will use a USB cable to power on. Open the serial monitor and set the baud rate to **115200**. When you see the serial monitor prints out the character string as below, it indicates that the Bluetooth of ESP32 is ready and waiting to connect with the mobile phone. (If open the serial monitor and set the baud rate to 115200, the information is not displayed, please press the RESET button of the ESP32)



© COM3 —		×
		Send
ets Jun 8 2016 00:22:57		^
rst:0x1 (POWERON_RESET),boot:0x13 (SPI_FAST_FLASH_BOOT)		
configsip: 0, SPIWP:0xee		
clk_drv:0x00,q_drv:0x00,d_drv:0x00,cs0_drv:0x00,hd_drv:0x00,wp_drv:0x00		
mode:DIO, clock div:1		
load:0x3fff0018,len:4		
load:0x3ff001c,len:1496		
load:0x40078000,len:8596		
load:0x40080400,len:6980		
entry 0x400806f4		
The device started, now you can pair it with bluetooth!		
		~
Autoscroll Show timestamp Newline Vill5200 baud V	lear o	utput

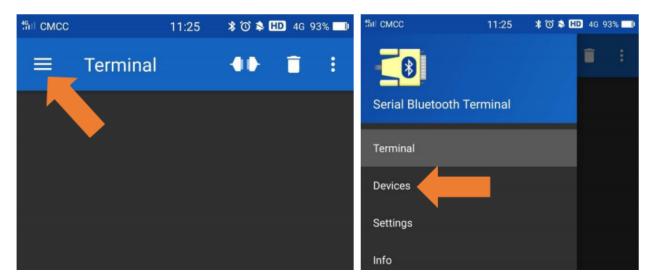
Make sure that the Bluetooth of your phone has been turned on and "Serial Bluetooth Terminal" has been installed.



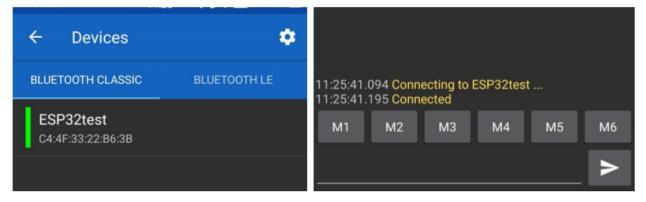
Click "Search" to search Bluetooth devices nearby and select "ESP32 test" to connect to.

46 II CMCC		11:24	*0*	4G 93%
<	Blu	uetootl	n	
Bluetoo	oth			-•
Renam MyBlueto	e this device			>
Paired de	vices			
Ō	ESP32test			ໃໃ
Available	devices			
*	4E:17:CF:AA	4:84:38		
With the E devices.	Bluetooth enable, th	is device w	ill be visible to	nearby

Turn on software APP, click the left of the terminal. Select "Devices" .



Select ESP32test in classic Bluetooth mode, and a successful connecting prompt will appear as shown on the right illustration.



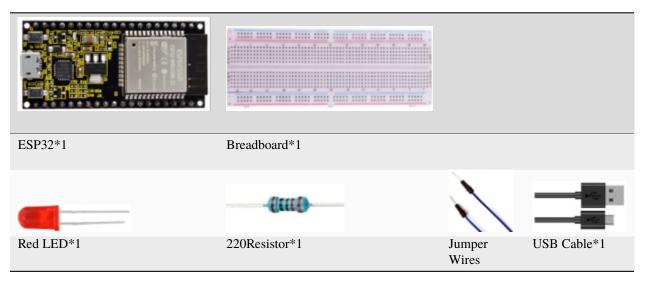
And now data can be transferred between your mobile phone and computer via ESP32.

Send "Hello!" from your phone, when the computer receives it, reply "Hi!" to your phone.

© COM3 —		$\times$
Hi!		Send
<pre>ets Jun 8 2016 00:22:57 rst:0x1 (POWERON_RESET),boot:0x13 (SPI_FAST_FLASH_BOOT) configsip: 0, SPIWP:0xee clk_drv:0x00,q_drv:0x00,d_drv:0x00,cs0_drv:0x00,hd_drv:0x00,wp_drv:0x00 mode:DIO, clock div:1 load:0x3fff0018,len:4 load:0x3fff001c,len:1496 load:0x40078000,len:8596 load:0x40080400,len:6980 entry 0x400806f4 The device started, now you can pair it with bluetooth! The device started, now you can pair it with bluetooth!</pre>		
Hello!		
Autoscroll Show timestamp Newline v 115200 baud v	Clear	outpu
Image: Second condition       11:26 <b>*</b> Image: Second condition       4G       92%       ■         Image: Second condition       Image: Second condit       Image: Second condition       Image: S		
11:25:41.094 Connecting to ESP32test 11:25:41.195 Connected 11:26:11.913 Hello! 11:26:24.759 Hi!		

# Project 35.2Bluetooth Control LED

1. Components



2. Wiring diagram



3. Project code

You can open the code we provideIf you haven't downloaded the code file, please click on the link to download it:Download Arduino Codes

The code used in this project is saved in folder(path:) "Arduino-Codes\Project 35Blue-tooth\Project\_35.2\_Bluetooth\_Control\_LED".

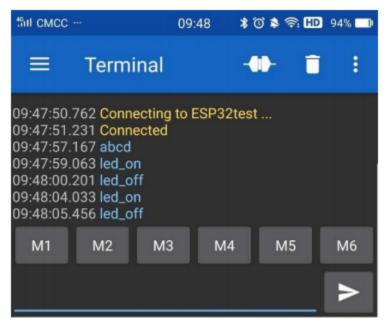
```
Project_35.2_Bluetooth_Control_LED | Arduino 1.8.16
                                                                                 \times
                                                                           \square
File Edit Sketch Tools Help
         +
     ÷)
                                                                                Ø
  Project_35.2_Bluetooth_Control_LED
 1*
  * Filename
             : Bluetooth Control LED
  * Description : The phone controls esp32's led via bluetooth.
               When the phone sends "LED on," ESP32's LED lights turn on.
               When the phone sends "LED_off," ESP32's LED lights turn off.
  * Auther
              : http//www.keyestudio.com
 */
 #include "BluetoothSerial.h"
 #include "string.h"
 #define LED 15
 BluetoothSerial SerialBT;
 char buffer[20];
 static int count = 0;
 void setup() {
  pinMode(LED, OUTPUT);
  SerialBT.begin("ESP32test"); //Bluetooth device name
  Serial.begin(115200);
             - ---- -
Invalid library found in C:\Users\Administrator\Desktop\Arduino\hardware\espressif\esp32\1.
Invalid library found in C:\Users\Administrator\Desktop\Arduino\hardware\espressif\esp32\1
<
                                                                                 х
                                       ESP32 Wrover Module, Default, QIO, 80MHz, 921600, None on COM3
/*
* Filename : Bluetooth Control LED
 * Description : The phone controls esp32's led via bluetooth.
               When the phone sends "LED_on," ESP32's LED lights turn on.
               When the phone sends "LED_off," ESP32's LED lights turn off.
             : http//www.keyestudio.com
* Auther
*/
##include "BluetoothSerial.h"
##include "string.h"
##define LED 15
BluetoothSerial SerialBT;
char buffer[20]:
static int count = 0;
void setup() {
 pinMode(LED, OUTPUT);
 SerialBT.begin("ESP32test"); //Bluetooth device name
 Serial.begin(115200);
 Serial.println("\nThe device started, now you can pair it with bluetooth!");
}
```

(continues on next page)

```
void loop() {
 while(SerialBT.available())
 {
   buffer[count] = SerialBT.read();
   count++;
 }
 if(count>0){
   Serial.print(buffer);
   if(strncmp(buffer,"led_on",6)==0){
     digitalWrite(LED,HIGH);
   }
   if(strncmp(buffer,"led_off",7)==0){
     digitalWrite(LED,LOW);
   }
   count=0;
   memset(buffer,0,20);
 }
}
//*****
```

### 4. Project result

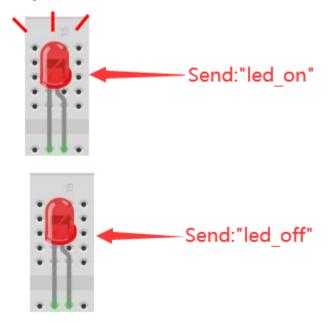
Compile and upload the code to the ESP32, after uploading successfullywe will use a USB cable to power on. The operation of the APP is the same as Project 35.1, you only need to change the sending content to "LED on" and "LED off" to operate LEDs on the ESP32. Data sent from mobile APP:



Display on the serial port of the computer:

SOM4		_	
			Send
abcd			^
led_on			
led_off			
led_on			
led_off			
			~
Autoscroll Show timestamp	Newline $\vee$	115200 baud $\lor$	Clear output

The phenomenon of LED



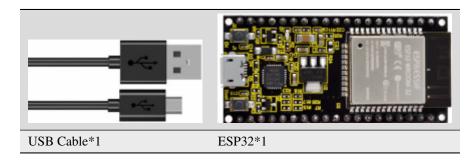
Attention: If the sending content isn't "led-on" or "led-off", then the state of LED will not change. If the LED is on, when receiving irrelevant content, it keeps on; Correspondingly, if the LED is off, when receiving irrelevant content, it keeps off.

# 5.37 Project 36WiFi Station Mode

#### 1. Introduction

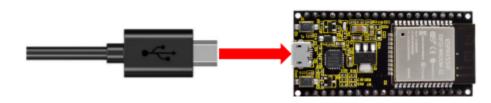
ESP32 has three different WiFi operating modes : Station modeAP mode and AP+Station mode. All WiFi programming projects must be configured with WiFi operating mode before using WiFi, otherwise WiFi cannot be used. In this project, we will learn about ESP32's WiFi Station mode.

### 2. Components



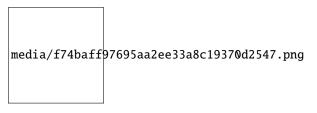
# 3. Wiring

Connect the ESP32 to the USB port on your computer using a USB cable.



4. Component knowledge

**Station mode:** When ESP32 selects Station mode, it acts as a WiFi client. It can connect to the router network and communicate with other devices on the router via WiFi connection. As shown below, the PC is connected to the router, and if ESP32 wants to communicate with the PC, it needs to be connected to the router.



5. Project code

You can open the code we provideIf you haven't downloaded the code file, please click on the link to download it:Download Arduino Codes

The code used in this project is saved in folder(path:) "Arduino-Codes\Project 36WiFi Station Mode\Project\_36\_WiFi\_Station\_Mode".

Project_36_WiFi_Station_Mode   Ard	uino 1.8.16				×
<u>File Edit Sketch Tools H</u> elp					
					Ø
Project_36_WiFi_Station_Mode					
//**********************	*****	*****	*****	*****	****
/*					
* Filename : WiFi Station * Description : Connect to yo	ur router using ESD32				
* Auther : http://www.key					
*/					
<pre>#include <wifi.h> //Include th</wifi.h></pre>	e WiFi Library header	file of ESP3	2.		
<pre>const char *password_Router = void setup() {    Serial.begin(115200);</pre>	"ChinaNet@233"; //Ent	er the router	passw	ord	
delay(2000);					
Serial.println("Setup start" WiFi.begin(ssid Router, pass		D32 in Statio	n mode	and c	0006
and single singl			n mode	and c	×
< c					>
1	ESP32 Wrover Module, Defaul		1800 No.		M20
	ESF32 Wrover Wrodure, Defaul	n, GIO, SUMHZ, 92	1000, Non		10120

Because the names and passwords of routers in various places are different, before the code runs, users need to enter the correct router's name and password in the box as shown in the illustration above.

```
\times
 Project 36 WiFi Station Mode | Arduino 1.8.16
                                                                      File Edit Sketch Tools Help
             +
                                                                            Ø
     +
  Project_36_WiFi_Station_Mode
                          //******
 1*
                                              Enter the correct Router
  * Filename
             : WiFi Station
  * Description : Connect to your router using ESP32
                                               name and password.
  * Auther : http//www.keyestudio.com
 */
 #include <WiFi.h> //Include the WiFi Library head file of ESP32.
 //Enter correct router name and password.
                           = "ChinaNet-2.4G-ODFO"; //Enter the router name
 const char *ssid_Router
 const char *password Router = "ChinaNet@233"; //Enter the router password
 void setup(){
  Serial.begin(115200);
  delay(2000);
  Serial.println("Setup start");
  WiFi.begin(ssid_Router, password_Router);//Set ESP32 in Station mode and connect
   Serial.println(String("Connecting to ")+ssid Router);
 //Check whether ESP32 has connected to router successfully every 0.5s.
 <
                                                                             >
 Done uploading.
Invalid library found in C:\Users\Administrator\Desktop\Arduino\hardware\espressif\
Invalid library found in C:\Users\Administrator\Desktop\Arduino\hardware\espressif\
 <
                                                                             >
                                ESP32 Wrover Module, Default, QIO, 80MHz, 921600, None on COM20
/*
* Filename : WiFi Station
* Description : Connect to your router using ESP32
            : http//www.keyestudio.com
* Auther
*/
##include <WiFi.h> //Include the WiFi Library header file of ESP32.
//Enter correct router name and password.
const char *ssid_Router = "ChinaNet-2.4G-0DF0"; //Enter the router name
const char *password_Router = "ChinaNet@233"; //Enter the router password
void setup(){
 Serial.begin(115200);
 delay(2000);
 Serial.println("Setup start");
 WiFi.begin(ssid_Router, password_Router);//Set ESP32 in Station mode and connect it to.
                                                                     (continues on next page)
```

```
-your router.
Serial.println(String("Connecting to ")+ssid_Router);
//Check whether ESP32 has connected to router successfully every 0.5s.
while (WiFi.status() != WL_CONNECTED){
    delay(500);
    Serial.print(".");
}
Serial.println("\nConnected, IP address: ");
Serial.println(WiFi.localIP());//Serial monitor prints out the IP address assigned to_
+ESP32.
Serial.println("Setup End");
}
void loop() {
}
```

### 6. Project result

After making sure the router name and password are entered correctly, compile and upload the code to ESP32, open serial monitor and set baud rate to 115200. When ESP32 successfully connects to "ssid\_Router", serial monitor will print out the IP address assigned to ESP32 by the router. (If open the serial monitor and set the baud rate to 115200, the information is not displayed, please press the RESET button of the ESP32)

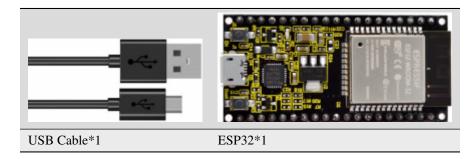
ية. 19		
RESET	BOOT	
👳 COM20	_	
		Send
clk drv:0x00,g drv:0x00,d drv:0x0	00,cs0_drv:0x00,hd_drv:0x00,wp_drv:0x00	
mode:DIO, clock div:1	,	^
load:0x3fff0018,len:4		
load:0x3fff001c,len:1496		
load:0x40078000,len:8596		
load:0x40080400,len:6980		
entry 0x400806f4		
Setup start		
Connecting to ChinaNet-2.4G-0DF0		
Connected, IP address:		
192.168.1.108		
Setup End		
		~
Autoscroll 🗌 Show timestamp	Newline $\checkmark$ 115200 baud $\checkmark$	Clear output

# 5.38 Project 37WiFi AP Mode

### 1. Introduction

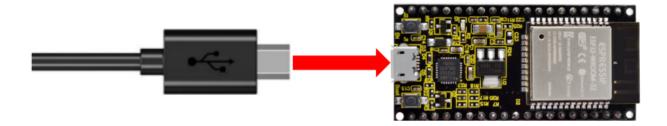
ESP32 has three different WiFi operating modes : Station modeAP mode and AP+Station mode. All WiFi programming projects must be configured with WiFi operating mode before using WiFi, otherwise WiFi cannot be used. In this project, we will learn about ESP32's WiFi AP mode.

### 2. Components



# 3. Wiring

Connect the ESP32 to the USB port on your computer using a USB cable.



# 4. Component knowledge

**AP mode :** When ESP32 selects AP mode, it creates a hotspot network that is separated from the Internet and waits for other WiFi devices to connect. As shown in the figure below, ESP32 is used as a hotspot. If a mobile phone or PC wants to communicate with ESP32, it must be connected to the hotspot of ESP32. Only after a connection is established with ESP32 can they communicate.



5. Project code

You can open the code we provideIf you haven't downloaded the code file, please click on the link to download it:Download Arduino Codes

The code used in this project is saved in folder(path:) "Arduino-Codes\Project 37WiFi AP Mode\Project\_37\_WiFi\_AP\_Mode".

Project_37_WiFi_AP_Mode   Arduino 1.8.16		٦
<u>F</u> ile <u>E</u> dit <u>S</u> ketch <u>T</u> ools <u>H</u> elp		
	ø	
Project_37_WiFi_AP_Mode		
//*************************************	******	^
* Filename : WiFi AP		
* Description : Set ESP32 to open an access point		
* Auther : http// <u>www.keyestudio.com</u> */		
<pre>*/ #include <wifi.h> //Include the WiFi Library header file of ESP32.</wifi.h></pre>		
<pre>const char *ssid_AP = "ESP32_WiFi"; //Enter the router name const char *password_AP = "12345678"; //Enter the router password</pre>		1
<pre>IPAddress local_IP(192,168,1,108);//Set the IP address of ESP32 itself IPAddress gateway(192,168,1,1); //Set the gateway of ESP32 itself IPAddress subnet(255,255,255,0); //Set the subnet mask for ESP32 itself</pre>		
<pre>void setup() {     Serial.begin(115200);</pre>		J
<	>	
1 ESP32 Wrover Module, Default, QIO, 80MHz, 921600, None	on COM20	

Before the code runs, you can make any changes to the AP name and password for ESP32 in the box as shown in the illustration above. Of course, you can leave it alone by default.

```
\times
 Project 37 WiFi AP Mode | Arduino 1.8.16
                                                                   File Edit Sketch Tools Help
                                                                        Ð.
     +
  Project_37_WiFi_AP_Mode
 //********
                       *************
 1*
                                                Set a name and a
  * Filename
            : WiFi AP
  * Description : Set ESP32 to open an access poipassword for ESP32 AP.
  * Auther : http//www.keyestudio.com
 */
 #include <WiFi.h> //Include the WiFi Library eader file of ESP32.
                       = "ESP32 WiFi"; //Enter the router name
 const char *ssid AP
 const char *password_AP = "12345678"; //Enter the router password
 IPAddress local_IP(192,168,1,108);//Set the IP address of ESP32 itself
 IPAddress gateway(192,168,1,1); //Set the gateway of ESP32 itself
 IPAddress subnet(255,255,255,0); //Set the subnet mask for ESP32 itself
 void setup(){
  Serial.begin(115200);
 <
                                                                         >
 Done Saving.
Invalid library found in C:\Users\Administrator\Desktop\Arduino\hardware\espress
Invalid library found in C:\Users\Administrator\Desktop\Arduino\hardware\espress
<
                                                                         >
                             ESP32 Wrover Module, Default, QIO, 80MHz, 921600, None on COM20
/*
* Filename : WiFi AP
* Description : Set ESP32 to open an access point
* Auther : http//www.keyestudio.com
*/
##include <WiFi.h> //Include the WiFi Library header file of ESP32.
const char *ssid_AP = "ESP32_WiFi"; //Enter the router name
const char *password_AP = "12345678"; //Enter the router password
IPAddress local_IP(192,168,1,108);//Set the IP address of ESP32 itself
IPAddress gateway(192,168,1,1); //Set the gateway of ESP32 itself
IPAddress subnet(255,255,255,0); //Set the subnet mask for ESP32 itself
void setup(){
 Serial.begin(115200);
 delay(2000);
```

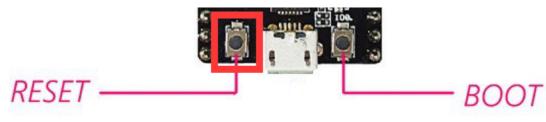
(continues on next page)

```
Serial.println("Setting soft-AP configuration ... ");
 WiFi.disconnect();
 WiFi.mode(WIFI_AP);
 Serial.println(WiFi.softAPConfig(local_IP, gateway, subnet) ? "Ready" : "Failed!");
 Serial.println("Setting soft-AP ... ");
 boolean result = WiFi.softAP(ssid_AP, password_AP);
 if(result){
   Serial.println("Ready");
   Serial.println(String("Soft-AP IP address = ") + WiFi.softAPIP().toString());
   Serial.println(String("MAC address = ") + WiFi.softAPmacAddress().c_str());
 }else{
   Serial.println("Failed!");
 }
 Serial.println("Setup End");
}
void loop() {
}
```

#### 6. Project result

Enter the ESP32 AP name and password correctly, compile and upload the code to ESP32, open the serial monitor and set the baud rate to **115200**. And then it will display as follows.

(If open the serial monitor and set the baud rate to 115200, the information is not displayed, please press the RESET button of the ESP32)



COM20	_			$\times$
			Se	nd
configsip: 0, SPIWP:0xee				~
clk_drv:0x00,q_drv:0x00,d_drv:0x00,cs0_drv:0x00,hd_drv:0x00,wp_drv:0x00				
mode:DIO, clock div:1				
load:0x3fff0018,len:4				
load:0x3fff001c,len:1496				
load:0x40078000,len:8596				
load:0x40080400,len:6980				
entry 0x400806f4				
Setting soft-AP configuration				
Ready				
Setting soft-AP				
Ready				
Soft-AP IP address = 192.168.1.108				
MAC address = 58:BF:25:8A:19:D1				
Setup End				
				~
Autoscroll Show timestamp Newline V115200 baud	~	Clea	r outp	ut

When observing the print information of the serial monitor, turn on the WiFi scanning function of your phone, and you can see the ssid\_AP on ESP32, which is called "ESP32\_Wifi" in this Code. You can enter the password "12345678" to connect it or change its AP name and password by modifying Code.

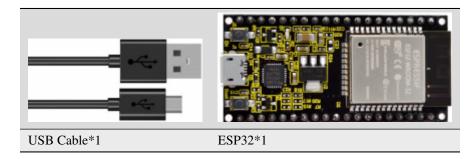
	Wi–Fi	
Settings	Wi-Fi View help	
Q Search	More settings	>
Log in to	AVAILABLE NETWORKS	
Access Cloud, AppGallery, and	ESP32_Wifi Connected	():-
more	ChinaNet-2.4G-0DF0	
WLAN ESP32_Wifi >	Saved, encrypted ChinaNet-Dsvv	0
	Encrynted	

# 5.39 Project 38WiFi Station+AP Mode

#### 1. Introduction

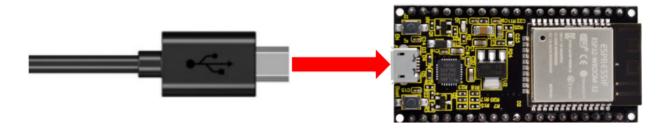
ESP32 has three different WiFi operating modes : Station modeAP mode and AP+Station mode. All WiFi programming projects must be configured with WiFi operating mode before using WiFi, otherwise WiFi cannot be used. In this project, we will learn ESP32's WiFi Station+AP mode.

### 2. Components



# 3. Wiring

Connect the ESP32 to the USB port on your computer using a USB cable.



#### 4. Component knowledge

**AP+Station mode:** In addition to AP mode and Station mode, ESP32 can also use AP mode and Station mode at the same time. This mode contains the functions of the previous two modes. Turn on ESP32's Station mode, connect it to the router network, and it can communicate with the Internet via the router. At the same time, turn on its AP mode to create a hotspot network. Other WiFi devices can choose to connect to the router network or the hotspot network to communicate with ESP32.

#### 5. Project code

You can open the code we provideIf you haven't downloaded the code file, please click on the link to download it:Download Arduino Codes

The code used in this project is saved in folder(path:) "Arduino-Codes\Project 38WiFi Station+AP Mode\Project\_38\_WiFi\_Station\_AP\_Mode".

🐵 Project_38_WiFi_Station_AP_Mode   Arduino 1.8.16 — 🗆 🗡	:
<u>F</u> ile <u>E</u> dit <u>Sketch</u> <u>T</u> ools <u>H</u> elp	
Project_38_WiFi_Station_AP_Mode	Â
<pre>ilename : WiFi AP+Station escription : ESP32 connects to the user's router, turning on an access point ather : http//<u>www.keyestudio.com</u></pre>	~
lude <wifi.h></wifi.h>	
<pre>t char *ssid_Router = "ChinaNet-2.4G-0DF0"; //Enter the router name t char *password_Router = "ChinaNet@233"; //Enter the router password t char *ssid_AP = "ESP32_WiFi"; //Enter the router name t char *password_AP = "12345678"; //Enter the router password</pre>	
<pre>setup() { rial.begin(115200); rial.println("Setting soft-AP configuration "); fi.disconnect();</pre>	~
< >>	
Done Saving.	
Invalid library found in C:\Users\Administrator\Desktop\Arduino\hardware\espress Invalid library found in C:\Users\Administrator\Desktop\Arduino\hardware\espress	
The sketch name had to be modified.	¥
< >	
43 ESP32 Wrover Module, Default, QIO, 80MHz, 921600, None on COM20	

It is analogous to Project 36 and Project 37. Before running the code, you need to modify ssid\_Router, password\_Router, ssid\_AP and password\_AP shown in the box of the illustration above.

```
Project 38 WiFi Station AP Mode | Arduino 1.8.16
                                                                    \times
File Edit Sketch Tools Help
                                                                         Ø
     +
  Project_38_WiFi_Station_AP_Mode
                           *****
 ilename : WiFi AP+Station
                                    Please enter the correct names
escription : ESP32 connects to the user
 ather : http://www.keyestudio.com and passwords of Router and AP.
lude <WiFi.h>
t char *ssid Router
                          "ChinaNet-2.4G-0DF0"; //Enter the router name
 t char *password_Router =
                          "ChinaNet@233"; //Enter the router password
 t char *ssid AP
                          "ESP32 WiFi"; //Enter the router name
                      =
                          "12345678"; //Enter the router password
 t char *password_AP
 setup() {
 rial.begin(115200);
 rial.println("Setting soft-AP configuration ... ");
 fi.disconnect();
 <
                                                                          >
 Done Saving.
Invalid library found in C:\Users\Administrator\Desktop\Arduino\hardware\espress
Invalid library found in C:\Users\Administrator\Desktop\Arduino\hardware\espress
The sketch name had to be modified.
 < |
                                                                          >
 43
                              ESP32 Wrover Module, Default, QIO, 80MHz, 921600, None on COM20
/*
* Filename : WiFi AP+Station
* Description : ESP32 connects to the user's router, turning on an access point
* Auther : http//www.keyestudio.com
*/
##include <WiFi.h>
const char *ssid_Router = "ChinaNet-2.4G-0DF0"; //Enter the router name
const char *password_Router = "ChinaNet@233"; //Enter the router password
const char *ssid_AP = "ESP32_WiFi"; //Enter the router name
const char *password_AP = "12345678"; //Enter the router password
void setup(){
 Serial.begin(115200);
 Serial.println("Setting soft-AP configuration ... ");
 WiFi.disconnect();
 WiFi.mode(WIFI_AP);
                                                                     (continues on next page)
```

```
Serial.println("Setting soft-AP ... ");
 boolean result = WiFi.softAP(ssid_AP, password_AP);
 if(result){
   Serial.println("Ready");
   Serial.println(String("Soft-AP IP address = ") + WiFi.softAPIP().toString());
   Serial.println(String("MAC address = ") + WiFi.softAPmacAddress().c_str());
 }else{
   Serial.println("Failed!");
 }
 Serial.println("\nSetting Station configuration ... ");
 WiFi.begin(ssid_Router, password_Router);
 Serial.println(String("Connecting to ")+ ssid_Router);
 while (WiFi.status() != WL_CONNECTED){
   delay(500);
   Serial.print(".");
 }
 Serial.println("\nConnected, IP address: ");
 Serial.println(WiFi.localIP());
 Serial.println("Setup End");
}
void loop() {
}
```

#### 6. Project result

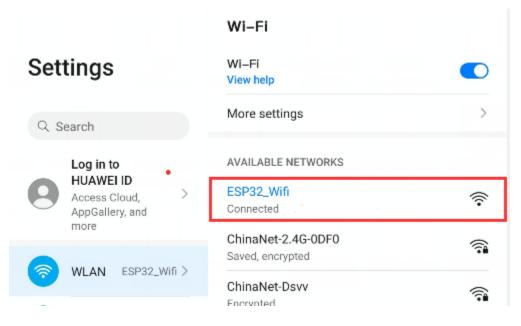
After making sure that the code is modified correctly, compile and upload the code to ESP32, open the serial monitor and set baud rate to 115200.

And then it will display as follows: (If open the serial monitor and set the baud rate to 115200, the information is not displayed, please press the RESET button of the ESP32)



💿 COM3				_		$\times$
						Send
load:0x40078000,len:8596						^
load:0x40080400,len:6980						
entry 0x400806f4						
Setting soft-AP configuration						
Setting soft-AP						
Ready						
Soft-AP IP address = 192.168.4.1						
MAC address = 58:BF:25:8A:19:D1						
Setting Station configuration						
Connecting to ChinaNet-2.4G-0DF0						
Connected, IP address:						
192.168.1.157						
Setup End						
						¥
Autoscroll Show timestamp	1	Newline $\sim$	115200 baud	$\sim$	Clear ou	itput

When observing the print information of the serial monitor, turn on the WiFi scanning function of your phone, and you can see the ssid\_AP on ESP32.

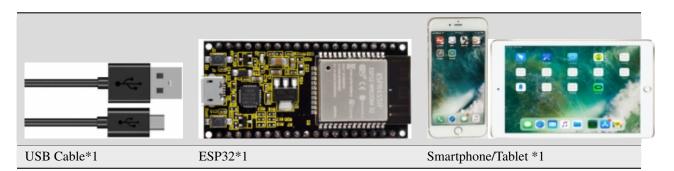


# 5.40 Project 39: WiFi Test

#### 1. Introduction

In this experiment, we first use the WiFi station mode of ESP32 to read the IP address of WiFi, and then connect WiFi through app to read the characters sent by each function button on App.

#### 2.Components



# 3. Wiring

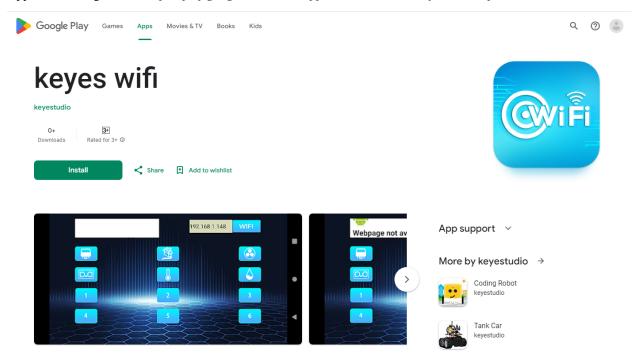
Connect the ESP32 to the USB port on your computer using a USB cable.



4. Install APP:

Android system (Smartphone/Tablet) APP: Go to Google Store to search for keys wifi

App link in Google Store: https://play.google.com/store/apps/details?id=com.keyestudio.esp8266\_web\_wifi2



# **Installation steps**

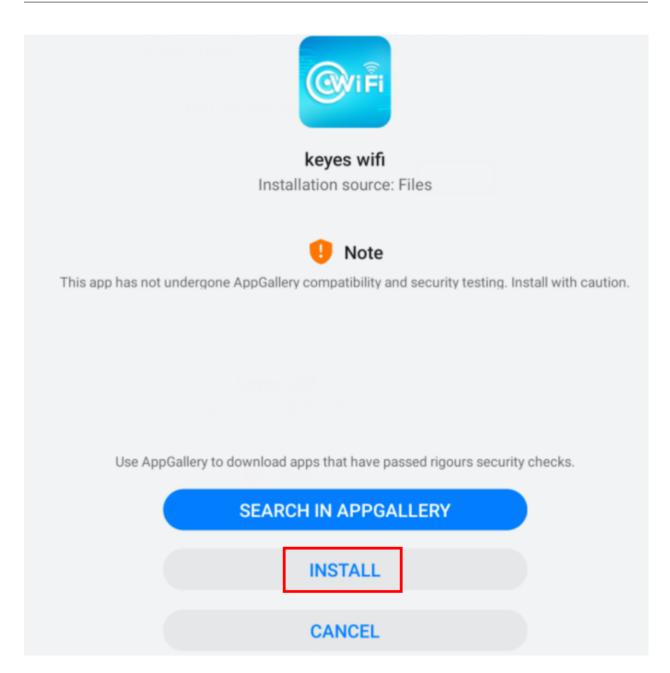
Now transfer the keyes wifi.apk file in the folder to android phone or tablet, click keyes wifi.apk file to enter the

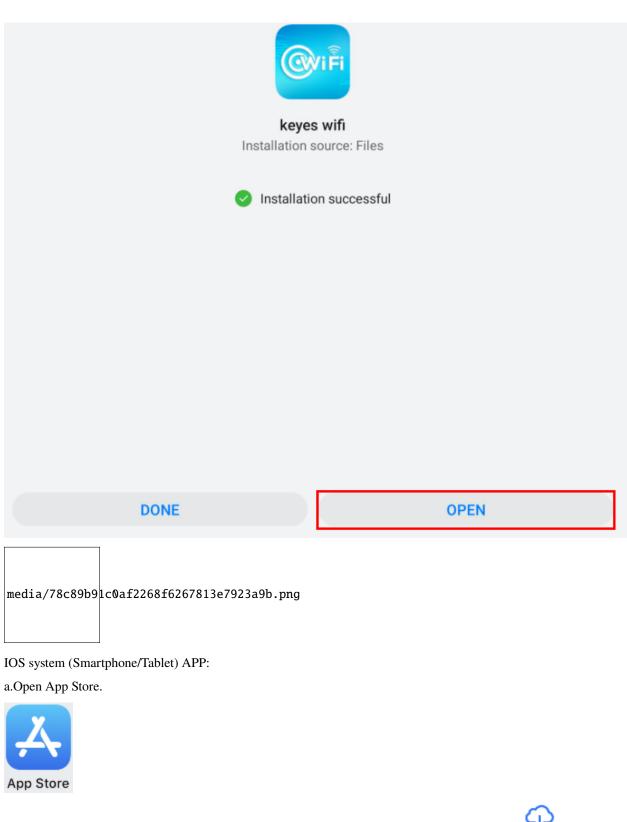
installation page, click"ALLOW", then click"INSTALL", after a while, click"OPEN" after the installation is completed to enter the APP interface.



keyes wifi.apk

	@vi Ēi	
keyes wifi		
Allow Files to install apps?		
Downloading apps f put your device and By touching ALLOW, these risks.	personal	data at greater risk.
Don't ask me agai	'n	
DENY		ALLOW





b. Enter keyes link in the search box and click Search. The download interface appears. Click" "" "to download and install the APP of keyes link. The following operations are similar to those of Android system, you can refer to the steps of Android system above for operation.

3. Project code

You can open the code we provideIf you haven't downloaded the code file, please click on the link to download it:Download Arduino Codes

The code used in this project is saved in folder(path:) "Arduino-Codes\Project 39WiFi Test\Project\_39\_WiFi\_Test".

```
/*
* Filename : WIFI Test
* Description : Wifi module test the ip of Wifi
* Auther
           : http//www.keyestudio.com
*/
// generated by KidsBlock
##include <Arduino.h>
##include <WiFi.h>
##include <ESPmDNS.h>
##include <WiFiClient.h>
String item = "0";
const char* ssid = "ChinaNet-2.4G-0DF0";
const char* password = "ChinaNet@233";
WiFiServer server(80);
void setup() {
 Serial.begin(115200);
 WiFi.begin(ssid, password);
 while (WiFi.status() != WL_CONNECTED) {
   delay(500);
   Serial.print(".");
 }
 Serial.println("");
 Serial.print("Connected to ");
 Serial.println(ssid);
 Serial.print("IP address: ");
 Serial.println(WiFi.localIP());
 server.begin();
 Serial.println("TCP server started");
 MDNS.addService("http", "tcp", 80);
}
void loop() {
 WiFiClient client = server.available();
 if (!client) {
     return;
 }
 while(client.connected() && !client.available()){
     delay(1);
 }
 String req = client.readStringUntil('\r');
 int addr_start = req.indexOf(' ');
 int addr_end = req.indexOf(' ', addr_start + 1);
 if (addr_start == -1 || addr_end == -1) {
     Serial.print("Invalid request: ");
```

(continues on next page)

```
Serial.println(req);
     return;
 }
 req = req.substring(addr_start + 1, addr_end);
 item=rea:
 Serial.println(item);
 String s;
 if (req == "/")
 {
     IPAddress ip = WiFi.localIP();
     String ipStr = String(ip[0]) + '.' + String(ip[1]) + '.' + String(ip[2]) + '.' +.
→String(ip[3]);
     s = "HTTP/1.1 200 OK\r\nContent-Type: text/html\r\n\r\n<!DOCTYPE HTML>\r\n<html>
→Hello from ESP32 at ";
     s += ipStr;
     s += "</html>\r\n':
     Serial.println("Sending 200");
     client.println(s);
 }
 //client.print(s);
 client.stop();
}
```

**Special attention:** you need to change the user's Wifi name and Wifi password in the experiment **const char\* ssid = "ChinaNet-2.4G-0DF0";** 

code const char\* password = "ChinaNet@233"; to your own Wifi name and Wifi password.

5. Project result

After making sure that the Code is modified correctly, compile and upload the code to ESP32.Note: If uploading the code fails, you can press the Boot button on ESP32 after clicked, and release the Boot



after the percentage of

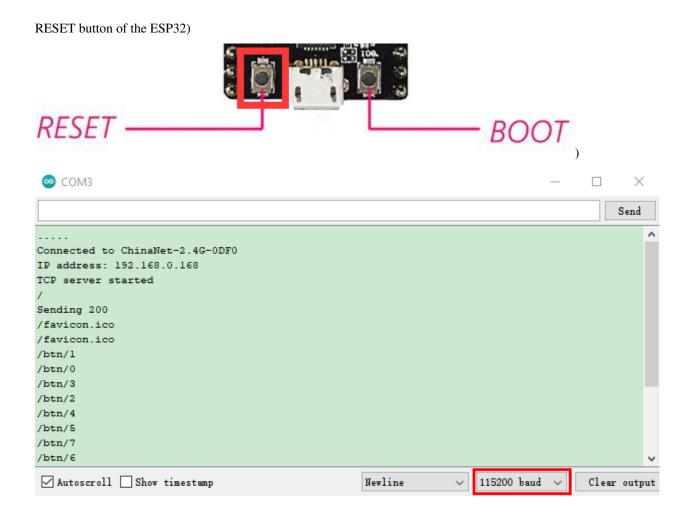
button

uploading progress appears)open the serial monitor and set baud rate to 115200.

In this way, the serial port monitor prints the detected WiFi IP address, then open the WiFi APP and enter the detected WiFi IP address in the text box in front of the WiFi button (for example, the IP address shown by the serial port monitor below :192.168.0.168), then click the WiFi button. "403 Forbidden" or "Webpage not available" will change to "192.168.0.168", indicating that the APP is already connected to WiFi.

media/ee42a1bde8dd1bc4cb554ccfef1ab9ae.png

Click each function button on the APP by hand, and then the serial port monitor will print the corresponding characters received. (If open the serial monitor and set the baud rate to 115200, the information is not displayed, please press the



## 5.41 Project 40WiFi Smart Home

#### 1. Introduction

In the previous experiment, we have learned the WiFi Station mode, WiFi AP mode and WiFi AP+Station mode of the ESP32. In this project, we will use ESP32's WiFi Station mode to control the work of multiple sensors/modules through APP connection with WiFi to achieve the effect of WiFi smart home.

2. Components



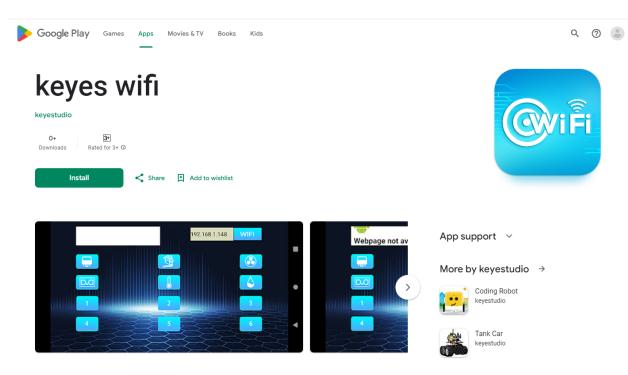
3. Wiring diagram



(Note: Connect the wires and then install a small fan blade on the DC motor. )

4. Install APP:

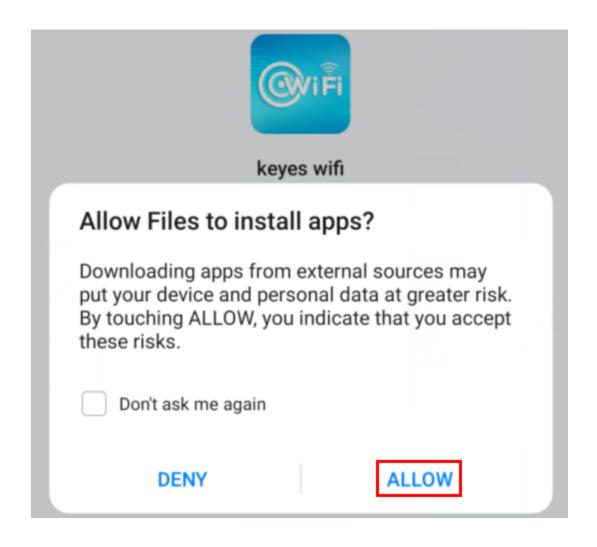
Android system (Smartphone/Tablet) APP:Go to Google Store to search for keys wifi

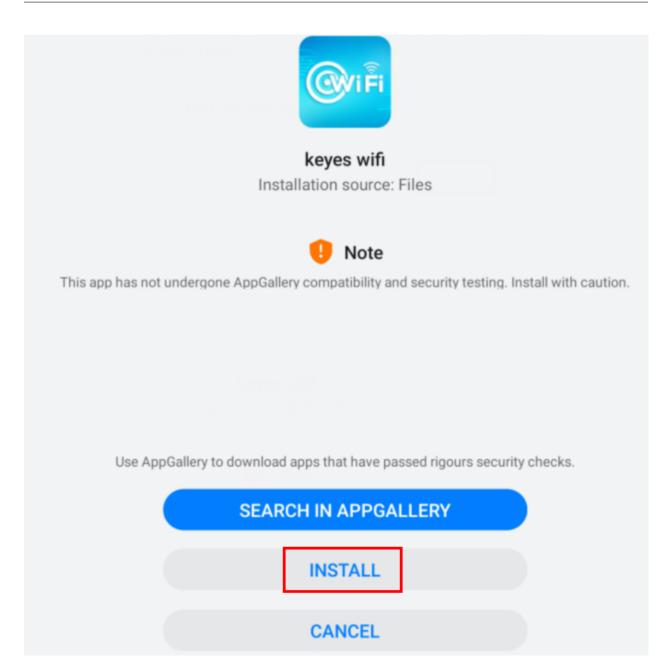


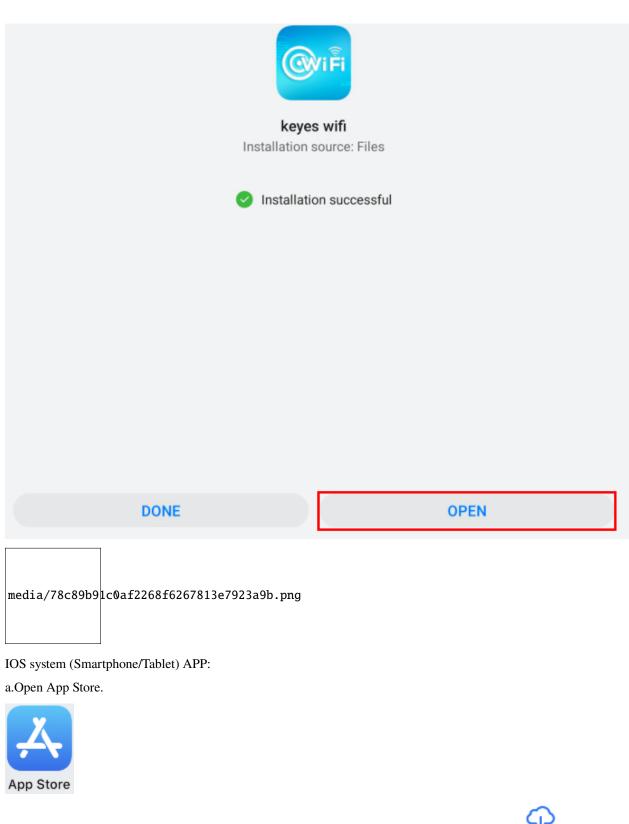
#### **Installation steps**



keyes wifi.apk







b. Enter keyes link in the search box and click Search. The download interface appears. Click" "" "to download and install the APP of keyes link. The following operations are similar to those of Android system, you can refer to the steps of Android system above for operation.

5. Add the xht11 and ESP32Servo libraries

This code uses two libraries named "**xht11**" and "**ESP32Servo**", if you haven't installed them yet, please do so before learning. The steps to add third-party libraries are as follows:

Add the xht11 library first:

Open the Arduino IDEclick"Sketch" $\rightarrow$ "Include Library" $\rightarrow$ "Add .zip Library...". In the pop-up window, find the file named "2 Windows System\2. C\_Tutorial\3.Libraries\xht11.ZIP" which locates in this directory. Select the xht11.ZIP file and then click"Open".

Project_40_WiFi_Smart_Home   Arduino 1.8.16						_		×
File Edit Sketch	Tools Help							
	/erify/Compile	Ctrl+R						Ø
ι 	Jpload	Ctrl+U						_
Project (	Upload Using Programm	er Ctrl+Shift+U		Δ				
	Export compiled Binary	Ctrl+Alt+S	Manage Librar	ries	Ctrl+Shift+I			^
xhtll - unsign S	Show Sketch Folder	Ctrl+K	Add .ZIP Libra	ry				
l.	nclude Library	;	Arduino librari	ies				
	Add File		Bridge	103				
Servo myser int servoPi			Esplora					
#define Rel			Ethernet					
	19 //IN1 corre	esponds to Il						
#define IN2 18 //IN2 corresponds to IN-			GSM					
#define trigPin 14								
#define echoPin 27		LiquidCrystal Mouse						
int distand	el:							
String dis	-		Robot IR Remo	ote				
int ip_flag	g = 1;		SD					~
💿 Select a zip fi	ile or a folder containing	the library you'd lik	e to add					×
Look <u>i</u> n:	3. Libraries				~	🏚 📂 🛄 <del>-</del>		
	ESP32Servo-0.8.0							
	HT16K33_Lib_For_E	SP32						
Recent Items	IRremoteESP8266-2	.7.4						
	🔛 Keypad-3.1.1							
	LCD_128X32							
Desktop	MFRC522_I2C TM1650							
4-	Wire						1	2
Documents	w xht11	0						
	File <u>n</u> ame: :nim	g Kit Ultimate Ed	ition\2. Windows \$	System\2.	C_Tutorial\3. Li	ibraries	0p	en
This PC	Files of type: ZIP	files or folders				~	Can	cel

Then Add the ESP32Servo library:

Open the Arduino IDEclick "Sketch"  $\rightarrow$  "Include Library"  $\rightarrow$  "Add .ZIP Library...". In the pop-up window, find the file named\*\*"2. Windows System 2. C\_Tutorial 3.Libraries ESP32Servo-0.8.0.ZIP"\*\* which locates in this directory. Select the **ESP32Servo-0.80.ZIP** fileand then click"Open".

Project_40_\	WiFi_Smart_Home   Ard	uino 1.8.16			_		×
File Edit Sketch	h Tools Help						
$\odot \odot$	Verify/Compile	Ctrl+R					ø
	Upload	Ctrl+U					_
Project	Upload Using Program	mer Ctrl+Shift+U	Δ				
//gpio xhtll	Export compiled Binary	Ctrl+Alt+S	Manage Libraries	Ctrl+Shift+I			^
	Show Sketch Folder	Ctrl+K	Add .ZIP Library				
# i m m 1 m	Include Library	;	Arduino libraries				
Servo myse	Add File		Bridge				
int servoP:	in = 4;		Esplora				
#define Re:	-		Ethernet				
-	1 19 //IN1 corn 2 18 //IN2 corre	-	Firmata				
#define IN.		sponds to IN-	GSM				
#define ech	-		LiquidCrystal				
			Mouse				
int distand			Robot IR Remote				
String dis int ip flag			SD				
	g 1,		<u>,</u>				*
🥯 Select a zip f	file or a folder containin	g the library you'd like	e to add				$\times$
Look <u>i</u> n	: 3. Libraries			~	🤌 📂 🛄	•	
Q-	ESP32Servo-0.8.0	-0					
	HT16K33_Lib_For_						
Recent Items	IRremoteESP8266	-2.7.4					
	LCD_128X32						
Desktop	MFRC522_12C						
<u>A-</u>	TM1650						0
	Wire						
Documents						_	
	File <u>n</u> ame: :ni	ng Kit Ultimate Ed	ition\2. Windows System	\2. C_Tutorial\3. Li	braries	01	pen
This PC	Files of type: ZI	P files or folders			$\sim$	Car	ncel

6. Project code

After the **xht11** and **ESP32Servo** libraries were added, You can open the code we provideIf you haven't downloaded the code file, please click on the link to download it:Download Arduino Codes

The code used in this project is saved in folder(path:) "Arduino-Codes\Project 40WiFi Smart Home\Project\_40\_WiFi\_Smart\_Home".

```
##include <WiFi.h>
##include <ESPmDNS.h>
##include <WiFiClient.h>
##include "xht11.h"
//gpio15
xht11 xht(15);
unsigned char dht[4] = {0, 0, 0, 0};
##include <ESP32Servo.h>
Servo myservo;
int servoPin = 4;
##define Relay 32
##define IN1 19 //IN1 corresponds to IN+
##define IN2 18 //IN2 corresponds to IN-
##define trigPin 14
##define echoPin 27
int distance1;
String dis_str;
int ip_flag = 1;
int ultra_state = 1;
int temp_state = 1;
int humidity_state = 1;
String item = "0";
const char* ssid = "ChinaNet-2.4G-0DF0"; //the name of user's wifi
const char* password = "ChinaNet@233"; //the password of user's wifi
WiFiServer server(80);
String unoData = "";
void setup() {
  Serial.begin(115200);
  pinMode(Relay, OUTPUT);
  myservo.setPeriodHertz(50);
 myservo.attach(servoPin, 500, 2500);
  pinMode(IN1, OUTPUT);
  pinMode(IN2, OUTPUT);
  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED) {
   delay(500);
   Serial.print(".");
  }
  Serial.println("");
  Serial.print("Connected to ");
  Serial.println(ssid);
  Serial.print("IP address: ");
  Serial.println(WiFi.localIP());
  server.begin();
  Serial.println("TCP server started");
  MDNS.addService("http", "tcp", 80);
```

```
digitalWrite(IN1, LOW);
  digitalWrite(IN2, LOW);
  digitalWrite(Relay, LOW);
 pinMode(trigPin, OUTPUT);
 pinMode(echoPin, INPUT);
}
void loop() {
  WiFiClient client = server.available();
  if (!client) {
     return:
  }
  while(client.connected() && !client.available()){
      delay(1);
  }
  String req = client.readStringUntil('\r');
  int addr_start = req.indexOf(' ');
  int addr_end = req.indexOf(' ', addr_start + 1);
  if (addr_start == -1 || addr_end == -1) {
      Serial.print("Invalid request: ");
      Serial.println(req);
     return;
  }
  req = req.substring(addr_start + 1, addr_end);
  item=req;
  Serial.println(item);
  String s;
  if (req == "/")
  {
      IPAddress ip = WiFi.localIP();
      String ipStr = String(ip[0]) + '.' + String(ip[1]) + '.' + String(ip[2]) + '.' +...

→String(ip[3]);

      s = "HTTP/1.1 200 OK\r\nContent-Type: text/html\r\n\r\n<!DOCTYPE HTML>\r\n<html>
→Hello from ESP32 at ";
      s += ipStr;
      s += "</html>\r\n\r\n";
      Serial.println("Sending 200");
      client.println(s);
  }
  else if(req == "/btn/0")
  {
   Serial.write('a');
   client.println(F("turn on the relay"));
   digitalWrite(Relay, HIGH);
  }
  else if(req == "/btn/1")
  {
   Serial.write('b');
   client.println(F("turn off the relay"));
   digitalWrite(Relay, LOW);
  }
```

```
else if(req == "/btn/2")
{
 Serial.write('c');
 client.println("Bring the steering gear over 180 degrees");
 myservo.write(180);
 delay(200);
}
else if(req == "/btn/3")
{
 Serial.write('d');
 client.println("Bring the steering gear over 0 degrees");
 myservo.write(0);
 delay(200);
}
else if(req == "/btn/4")
{
 Serial.write('e');
 client.println("esp32 already turn on the fans");
 digitalWrite(IN1, LOW);
 digitalWrite(IN2, HIGH);
}
else if(req == "/btn/5")
{
 Serial.write('f');
 client.println("esp32 already turn off the fans");
 digitalWrite(IN1, LOW);
 digitalWrite(IN2, LOW);
}
else if(req == "/btn/6")
{
 Serial.write('g');
 while(Serial.available() > 0)
  {
   unoData = Serial.readStringUntil('#');
    client.println("Data");
 }
 while(ultra_state>0)
     {
        Serial.print("Distance = ");
        Serial.print(checkdistance());
        Serial.println("#");
        Serial1.print("Distance = ");
        Serial1.print(checkdistance());
        Serial1.println("#");
        int t_val1 = checkdistance();
        client.print("Distance(cm) = ");
        client.println(t_val1);
        ultra_state = 0;
      }
}
else if(req == "/btn/7")
{
```

```
Serial.write('h');
 client.println("turn off the ultrasonic");
 ultra_state = 1;
}
else if(req == "/btn/8")
{
 Serial.write('i');
 while(Serial.available() > 0)
   {
   unoData = Serial.readStringUntil('#');
    client.println(unoData);
   }
 while(temp_state>0)
    {
      if (xht.receive(dht)) {
        Serial.print("Temperature = ");
        Serial.print(dht[2],1);
        Serial.println("#");
        Serial1.print("Temperature = ");
        Serial1.print(dht[2],1);
        Serial1.println("#");
        int t_val2 = dht[2];
        client.print("Temperature(°) = ");
        client.println(t_val2);
      }
      temp_state = 0;
   }
}
else if(req == "/btn/9")
{
 Serial.write('j');
 client.println("turn off the temperature");
 temp_state = 1;
}
else if(req == "/btn/10")
{
  Serial.write('k');
 while(Serial.available() > 0)
   {
     unoData = Serial.readStringUntil('#');
     client.println(unoData);
  }
 while(humidity_state > 0)
    {
      if (xht.receive(dht)) {
        Serial.print("Humidity = ");
        Serial.print(dht[0],1);
        Serial.println("#");
        Serial1.print("Humidity = ");
        Serial1.print(dht[0],1);
        Serial1.println("#");
        int t_val3 = dht[0];
```

```
client.print("Humidity(%) = ");
        client.println(t_val3);
      }
      humidity_state = 0;
     }
 }
 else if(req == "/btn/11")
 {
   Serial.write('1');
   client.println("turn off the humidity");
   humidity_state = 1;
   }
 //client.print(s);
 client.stop();
}
int checkdistance() {
 digitalWrite(14, LOW);
 delayMicroseconds(2);
 digitalWrite(14, HIGH);
 delayMicroseconds(10);
 digitalWrite(14, LOW);
 int distance = pulseIn(27, HIGH) / 58;
 delay(10);
 return distance;
}
```

**Special attention:** you need to change the user's Wifi name and Wifi password in the experiment code to your own Wifi name and Wifi password.

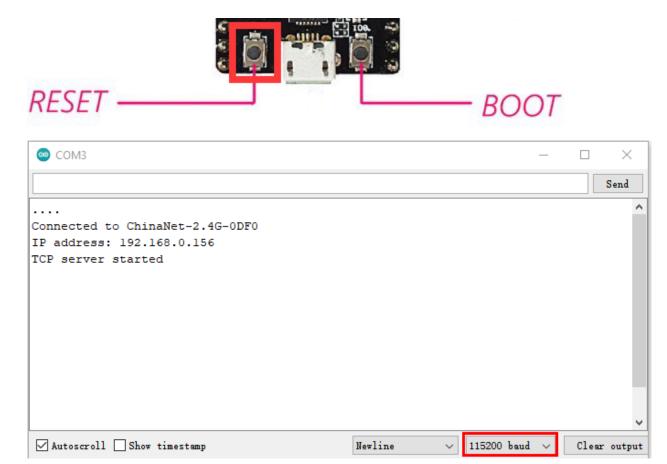
```
const char* ssid = "ChinaNet-2.4G-0DF0"; //the name of user's wifi
const char* password = "ChinaNet@233"; //the password of user's wifi
```

7. Project result

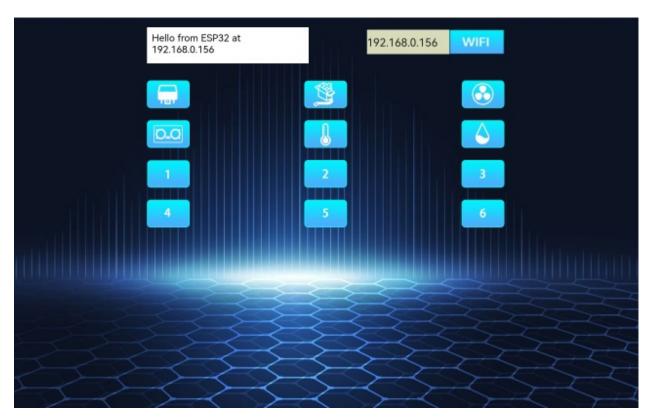
After making sure that the Code is modified correctly, external power supply and power on, and then compile and upload the code to ESP32.Note: If uploading the code fails, you can press the Boot button on ESP32 after click ,

and release the Boot button

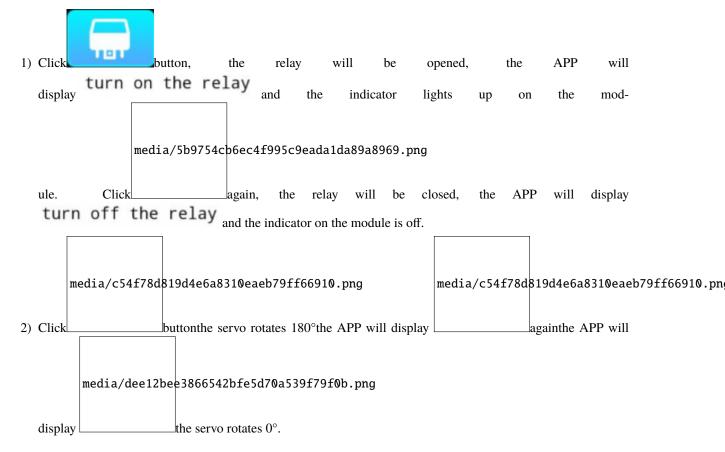
after the percentage of uploading progress appearsopen the serial monitor and set baud rate to 115200. In this way, the serial port monitor prints the detected WiFi IP address,(If open the serial monitor and set the baud rate to 115200, the information is not displayed, please press the RESET button of the ESP32)

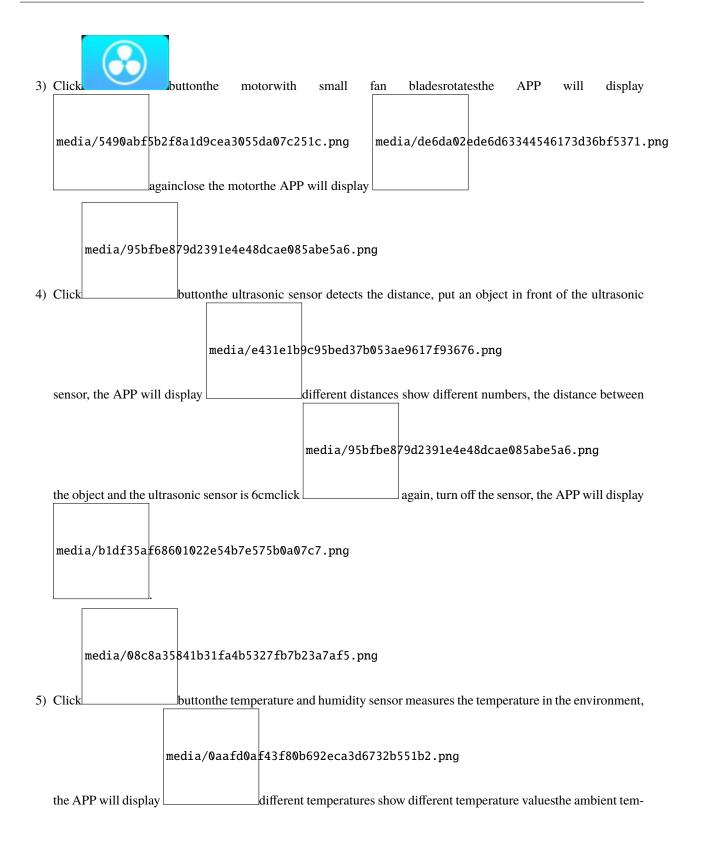


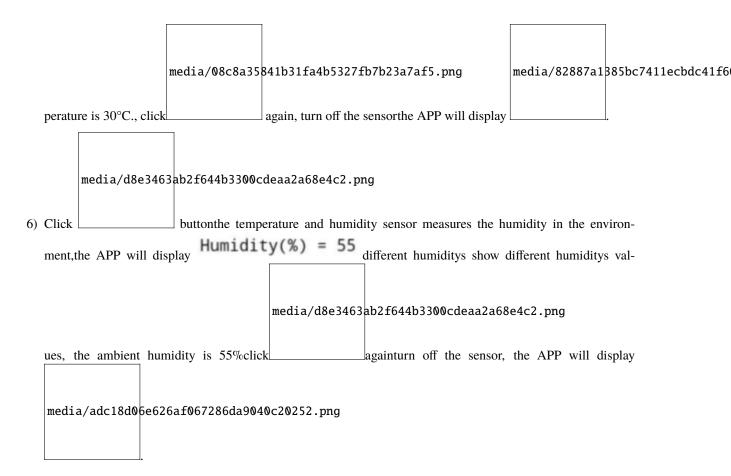
Then open the WiFi APP and enter the detected WiFi IP address in the text box in front of the WiFi button (for example, the IP address shownby the serial port monitor below :192.168.0.156), then click the WiFi button, "Hello from ESP32 at 192.168.0.156" is displayed in the text box next to the WiFi IP address, indicating that the APP is already connected to WiFi.(WiFi IP address sometimes changes, if the original IP address doesn't work, you need to re-check the WiFi IP address)



After the APP has been connected to WiFi, the following operations will be performed:







### CHAPTER

# **GETTING STARTED WITH PYTHON**

Before starting building the projects, you need to make some preparation first. Do not skip this step as it provides crucial information for installing.

# 6.1 1.Installing Thonny (Important)

Thonny is a free, open-source software platform with compact size, simple interface, simple operation and rich functions, making it a Python IDE for beginners. In this tutorial, we use this IDE to develop ESP32 during the whole process.

Thonny supports various operating system, including WindowsMac OSLinux.

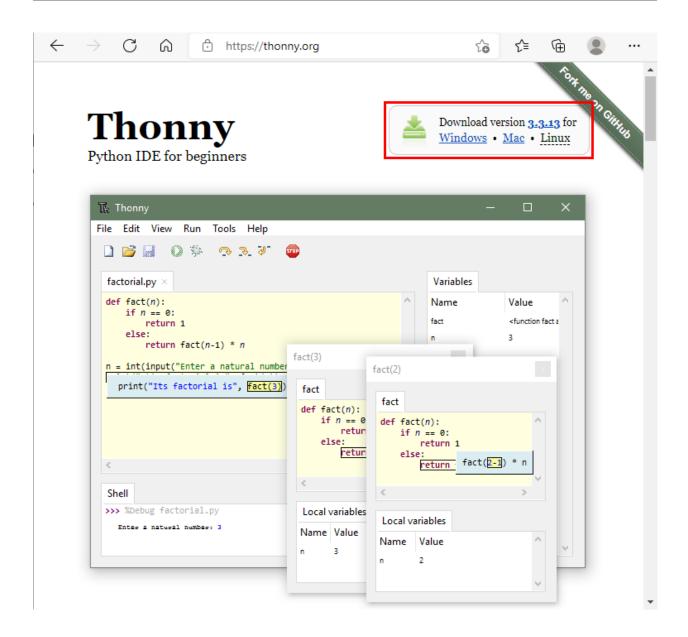
## 6.1.1 (1) Downloading Thonny

Enter the official website of Thonny: https://thonny.org and download the latest version of Thonny.

Open-source code repositories of Thonny: https://github.com/thonny/thonny

Follow the instruction of official website to install Thonny or click the links below to download and install. (Select the appropriate one based on your operating system.)

Oper- ating Sys- tem	Download links/methods
MAC OS	https://github.com/thonny/thonny/releases/download/v3.2.7/thonny-3.2.7.pkg
Win- dows	https://github.com/thonny/thonny/releases/download/v3.2.7/thonny-3.2.7.exe
Linux	The latest version:Binary bundle for PC (Thonny+Python): bash <(wget -O - https://thonny.org/installer-for-linux)With pip: pip3 install thonnyDistro packages (may not be the latest version):Debian, Rasbian, Ubuntu, Mint and others: sudo apt install thonnyFedora: sudo dnf install thonny



### 6.1.2 (2) Install Thonny on Windows

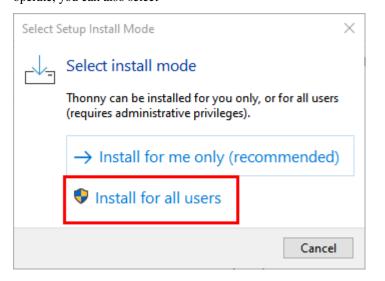
The icon of Thonny after downloading is as below.



thonny-3.3.13.exe

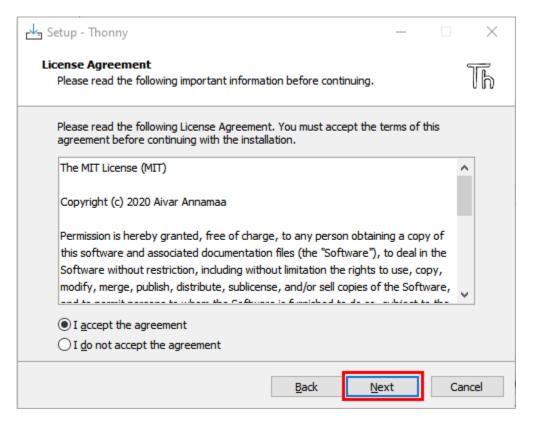
1. Double click"thonny-3.3.13.exe"the following dialog box will appear. I choose" **Install for all users** "to

operate, you can also select"  $\rightarrow$  Install for me only (recommended) ", to operate.



2. If you're not familiar with computer software installation, you can simply keep clicking "**Next**" until the installation completes.

4	Setup - Thonny	_		$\times$
D	📕 🖸 🅸 💮 🐎 🤇			
		Welcome to using Thonny!		
uess.	PY C			
1	import Candom	This wizard will install Thonny 3.3.13 for all users.		
2	n = ran randir			
3	U U.U.			
4 5	guess = int(input			
6	while n != "guess			
7	if guess < n:			
8	print("gu			
9	guess = i			
10	elif guess >			
11	print("gu			
12	guess = i			
13 14	else: print("yc			
14	рилисс ус			
		Next	Cance	4



3. If you want to change Thonny's installation path, you can click "**Browse...**" to modify it. After selecting installation path, click "**OK**". If you do not want to change it, just click "**Next**".

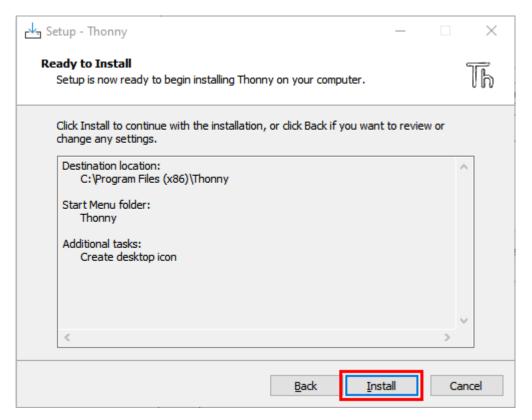
Setup - Thonny	_		$\times$
Select Destination Location Where should Thonny be installed?			Th
Setup will install Thonny into the following folder.			
To continue, click Next. If you would like to select a different folder,	click Bro	wse.	_
C:\Program Files (x86)\Thonny	Bro	owse	
2 At least 99.5 MB of free disk space is required.		0	
<u>B</u> ack <u>N</u> e	xt	Car	ncel

Setup - Thonny	_		$\times$
Select Start Menu Folder Where should Setup place the program's shortcuts?			Th
To continue, click Next. If you would like to select a different folder, o			
Thonny	B <u>r</u> o	wse	
<u>B</u> ack <u>N</u> ex	t	Can	icel

4. Check "Create desktop icon" and then it will generate a shortcut on your desktop to facilitate you to open Thonny later.

🛃 Setup - Thonny			$\times$
Select Additional Tasks Which additional tasks should be performed?			Th
Select the additional tasks you would like Setup to perform while ins then click Next.	stalling Th	onny,	
Create desktop icon			
	2		
<u>B</u> ack <u>N</u>	ext	Ca	incel

5. Click "**Install**" to install the software.



6. During the installation, you only need to wait for completion, and you should not click "**Cancel**", otherwise Thonny will fail to be installed.

Setup - Thonny	_		×
<b>Installing</b> Please wait while Setup installs Thonny on your computer.			Th
Extracting files C:\Program Files (x86)\Thonny\Lib\jdlelib\rpc.py			1
		Ca	incel

7. Once you see the interface as below, Thonny has been installed successfully. Click "Finish".

4	Setup - Thonny	- 🗆 ×
	🔝 🜔 🏇 💮 決 🤅	
		Great success!
uess.	ру	
1	impo t nandom	Thonny is now installed. Run it via shortcut or right-dick a *.py
2	n = ran randir	file and select "Edit with Thonny".
3		
4	guess = int(input	
5		
6	while n != "guess	
7	<pre>if guess &lt; n:</pre>	
8	print("gı	
9	guess = i	
10	<pre>elif guess &gt;</pre>	Computers are useless. They can only give you
11	print("gu	answers.
12	guess = i	– Pablo Picasso
13	else:	
14	print("yc	
		<u> </u>

8. If you've check" **Create desktop icon**" during the installation process, you can see the below icon on your desktop.



# 6.2 2. Basic Configuration of Thonny

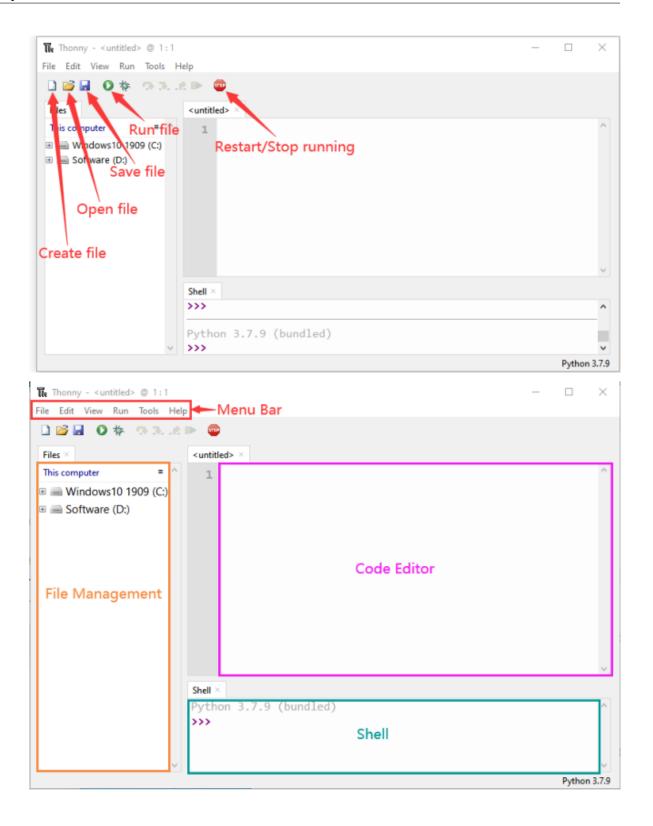
1. Click the desktop icon of Thonny and you can see the interface of it as follows, and we can also choose the language and initial settings. Once set, click "Let's Go!"

9		—		$\times$
Th	Language: Initial settings:	English Standard Let'	<ul> <li>✓</li> <li>S go!</li> </ul>	
1		_		$\times$
TH	Language: Initial settings:	English Čeština [BETA] Deutsch Eesti English Español Français Italiano Lietuvių Magyar [BETA] Norsk (Bokmål) Norsk (Nynorsk) Nederlands Polski Português (PT) Português (BR)	~	

0			—		×
Th	Language: Initial settings:	English Standard	Let	v v	

2. Select"View"→"Files"and"Shell".

🏗 Thonny - <untitled> @ 1:1</untitled>		– 🗆 ×
File Edit View Run Tools I	elp	
This com Heap Help		
Increase font size Decrease font size	Ctrl++ Ctrl+-	
Focus editor Focus shell	Alt+E Alt+S	~
L	<pre>Python 3.7.9 (bundled) &gt;&gt;&gt; %cd 'C:\Users' &gt;&gt;&gt; %cd 'C:\' &gt;&gt;&gt;</pre>	^
	·	×
		Python 3.7.9



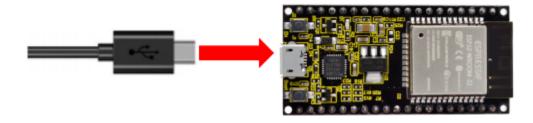
# 6.3 3.Installing CP2102 driver

ESP32 uses CP2102 to download codes. So before using it, we need to install CP2102 driver in our computers.

### 6.3.1 Windows System

Check whether CP2102 has been installed

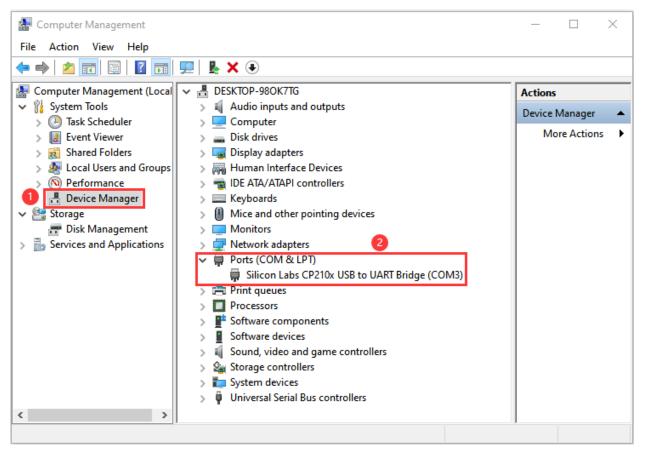
(1). Connect your computer and ESP32 with a USB cable.



(2). Turn to the main interface of your computer, select "This PC" and right-click to select "Manage".

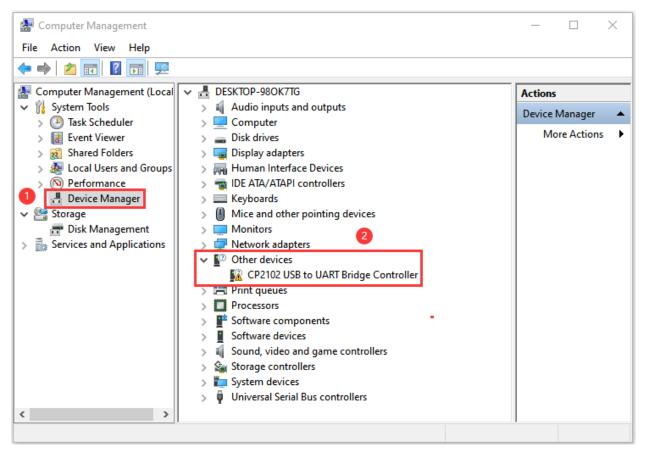
Recycle Bin		
This P	<b>Open</b> Pin to Quick access	
0025 0	Manage Pin to Start	
Contro	Map network drive Disconnect network drive	
Pane	Create shortcut Delete	
a <sup>l h</sup> Thomay	Properties	

(3). Click "**Device Manager**", your computer has installed CP2102 driveryou can see "Silicon Labs CP210x USB to UART Bridge (COMx)"



#### Installing CP2102 driver

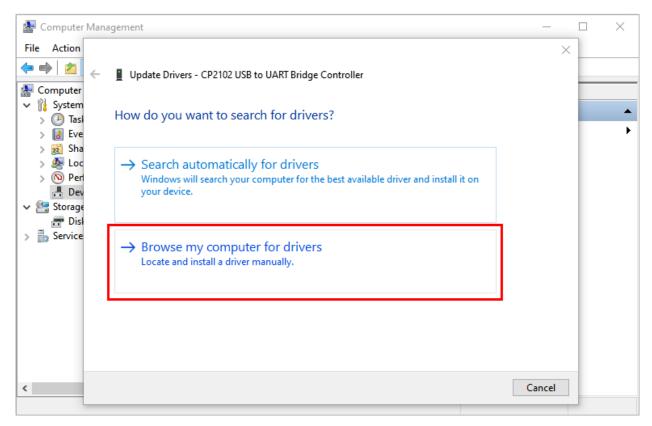
(1) If you have not yet installed the CP2102 driver, you'll see the following interface.



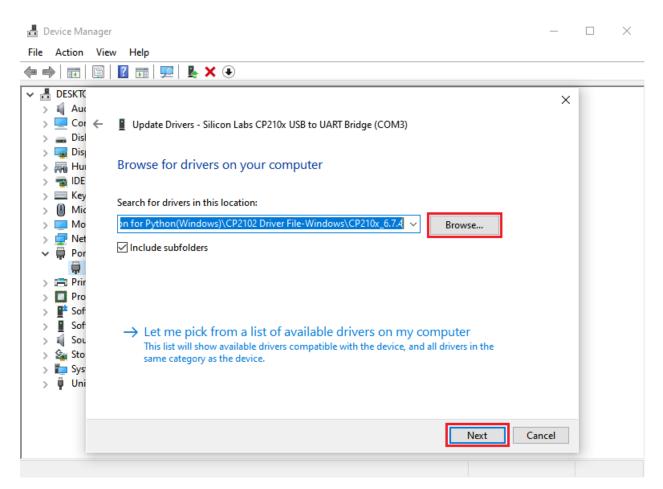
(2) Click "CP2102USB to UART Bridge Controller", and right-click to select "Update driver".

🔝 Computer Management		- 0	×
File Action View Help			
🗢 🔿   🖄 📰   📓 🗊   🖳 🖳 😓 😣			
😓 Computer Management (Local 🗸 🛃 DESKTOP-980K7TG		Actions	
System Tools M Audio inputs and outputs		Device Manad	ier 🔺
> 🕑 Task Scheduler 🛛 🔰 📃 Computer			,
> 🛃 Event Viewer 🛛 🔰 👝 Disk drives		More Act	ions 🕨
> 😥 Shared Folders 🛛 🔰 🥃 Display adapters	> 🙀 Display adapters		
> 👰 Local Users and Groups 🔰 🔋 🎆 Human Interface Devices	> 🙀 Human Interface Devices		
> 🔊 Performance > 📹 IDE ATA/ATAPI controllers	> 📷 IDE ATA/ATAPI controllers		
🕂 Device Manager 🔰 > 🔤 Keyboards	> 🔤 Keyboards		
Storage Mice and other pointing devices	> II Mice and other pointing devices		
📅 Disk Management 🔰 🔉 🛄 Monitors			
> 🍶 Services and Applications 🔰 🔉 💭 Network adapters	> 🚽 Network adapters		
✓ V Dther devices			
🙀 CP2102 USB to UART Bridge Controller	Update driver		
> 🚍 Print queues	Disable device		
> Processors			
> P Software components	Uninstall device	2	
> Software devices Scan for hardw		are changes	
> 🛍 Sound, video and game controllers			
> 🍇 Storage controllers	Properties		
> 🏣 System devices			
> 🏺 Universal Serial Bus controllers			
< >>			
aunches the Update Driver Wizard for the selected device.		, 	

(3) Click "Browse my computer for drivers".



(4) Click "Browse..." select CP210x\_6.7.4 (Driver path2. Windows System\1. Python\_Tutorial\1. Preparation for Python(Windows)\CP2102 Driver File-Windows), Click "Next".



(5) Wait for the CP2102 driver installation to be finished. When you see the following interface, which indicates that the CP2102 driver has been installed to your computer. You can close the interface.

😓 Computé	$\times$		×
File Action - Update Drivers - Silicon Labs CP210x USB to UART Bridge (COM3)			
Compute Vindows has successfully updated your drivers Syster	,		
> D Ta:			-
> 🔝 Ev Windows has finished installing the drivers for this device:			
> 😥 Sh > 🔊 Lo			
> No Pe Silicon Labs CP210X 056 to OAKT Bridge			
는 De < 2록 Storag			
T Die			
> 🛃 Servic			
	lose		
	lose	_	

(6) When ESP32 is connected to computer, the interface appears as follows.

🔚 Computer Management		—		Х
File Action View Help				
🔶 🤿 🙋 📰 🔄 🛛 💼	💻   🖡 🗙 🖲			
🜆 Computer Management (Local	V 🛃 DESKTOP-980K7TG	Actions		
✓	> 🐗 Audio inputs and outputs	Device Manager		
> 🕒 Task Scheduler	> 💻 Computer			
> 🛃 Event Viewer	> 👝 Disk drives	More Action	IS	)
> 👸 Shared Folders	> 🏣 Display adapters			
> 🜆 Local Users and Groups	> 🛺 Human Interface Devices			
> 🔊 Performance	> 🦏 IDE ATA/ATAPI controllers			
💾 Device Manager	> 🔤 Keyboards			
🗸 🚰 Storage	> II Mice and other pointing devices			
📻 Disk Management	> 🛄 Monitors			
> Loss Services and Applications	> 蓂 Network adapters			
	✓			
	Silicon Labs CP210x USB to UART Bridge (COM3)			
	> E Print queues			
	>  Processors			
	> J Software components			
	Software devices			
	> 👖 Sound, video and game controllers			
	> 🍇 Storage controllers			
	> 🏣 System devices			
<	> 🏺 Universal Serial Bus controllers			

## 6.3.2 MAC System

You can refer to the file **Get started with Arduino** in the folder to install the CP2102 driver. Path2. Windows System\2. C\_Tutorial\1. Preparation for C (Windows)

1. Preparation for C (Windows)

Share View				
« 2. Windows System > 2. C_Tutorial > 1. Preparation for C (Windows)				
Name	Date modified	Туре		
🔒 Arduino IDE	3/4/2022 4:28 PM	File folder		
CP2102 Driver File-MAC	2/17/2022 9:02 AM	File folder		
CP2102 Driver File-Windows	2/17/2022 9:02 AM	File folder		
Get started with Arduino	4/26/2022 11:49 AM	DOCX		

## 6.4 4.Burning Micropython Firmware (Important)

To run Python programs on ESP32, we need to burn a firmware to ESP32 first.

### 6.4.1 (1) Downloading Micropython firmware

Official website of microPythonhttp://micropython.org/

Webpage listing firmware of microPython for ESP32https://micropython.org/download/esp32/

# Firmware

Releases

v1.18 (2022-01-17) .bin [.elf] [.map] [Release notes] (latest)

v1.17 (2021-09-02) .bin [.elf] [.map] [Release notes] v1.16 (2021-06-23) .bin [.elf] [.map] [Release notes] v1.15 (2021-04-18) .bin [.elf] [.map] [Release notes] v1.14 (2021-02-02) .bin [.elf] [.map] [Release notes] v1.13 (2020-09-02) .bin [.elf] [.map] [Release notes] v1.12 (2019-12-20) .bin [.elf] [.map] [Release notes]

### Nightly builds

```
v1.18-121-gd8a7bf83c (2022-02-10) .bin [.elf] [.map]
v1.18-107-gaca40127b (2022-02-09) .bin [.elf] [.map]
v1.18-105-gada836b83 (2022-02-08) .bin [.elf] [.map]
v1.18-103-g6f7d6c567 (2022-02-08) .bin [.elf] [.map]
```

# Firmware (Compiled with IDF 3.x)

### Releases

Python\_Firmware

v1.14 (2021-02-02) .bin [.elf] [.map] [Release notes] (latest) v1.13 (2020-09-02) .bin [.elf] [.map] [Release notes] v1.12 (2019-12-20) .bin [.elf] [.map] [Release notes] v1.11 (2019-05-29) .bin [.elf] [.map] [Release notes] v1.10 (2019-01-25) .bin [.elf] [.map] [Release notes] v1.9.4 (2018-05-11) .bin [.elf] [.map] [Release notes]

Firmware used in this tutorial is esp32-20210902-v1.17.bin

Click the following link to download directlyhttps://micropython.org/resources/firmware/esp32-20210902-v1.17.bin

This file is also provided in our folder"2. Windows System/1. Python\_Tutorial/1. Preparation for Python(Windows)\Python\_Firmware".

```
      Share
      View

      Share
      1. Python_Tutorial > 1. Preparation for Python(Windows) > Python_Firmware

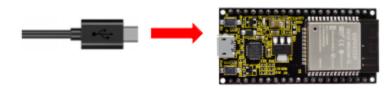
      Name
      Date modified

      Type

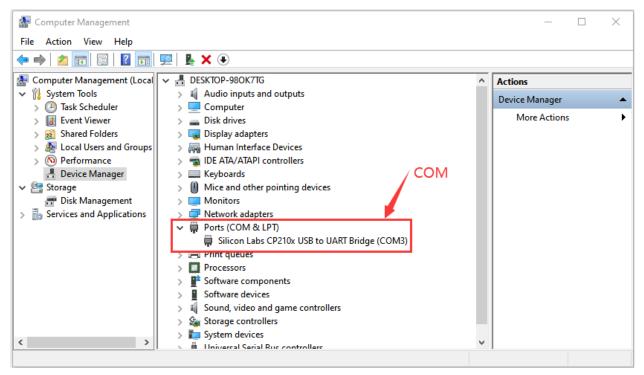
      esp32-20210902-v1.17.bin
      2/10/2022 11:23 AM
```

# 6.4.2 (2) Burning a Micropython Firmware

Connect your computer and ESP32 with a USB cable.

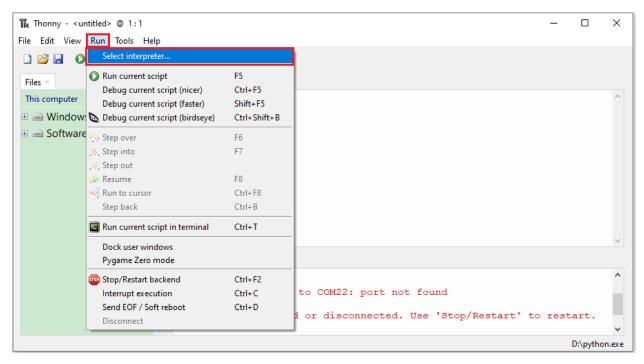


Make sure that the driver has been installed successfully and that it can recognize COM port correctly. Open device manager and expand "**Ports**(**COM&LPT**)".



Note: COM ports may be different on different computers

A. Open Thonnyclick "Run" and select "Select interpreter...".



B. Select "Micropython (ESP32)", select "Silicon Labs CP210x USB to UART Bridge(COM3)", and then click "Install or update firmware".

The Thonny options	×
General       Interpreter       Editor       Theme & Font       Run & Debug       Terminal       Shell       Assistant         Which interpreter or device should Thonny use for running your code?       1	
MicroPython (ESP32)         Details         Connecting via USB cable:         Connect your device to the computer and select corresponding port below         (look for your device name, "USB Serial" or "UART").         If you can't find it, you may need to install proper USB driver first.         Connecting via WebREPL (EXPERIMENTAL):         If your device supports WebREPL, first connect via serial, make sure WebREPL is enabled         (import webrepl_setup), connect your computer and device to same network and select         < WebREPL > below	
Port or WebREPI Silicon Labs CP210x USB to UART Bridge (COM3)	
3 Install or update firmware	
OK Cancel	

C. The following dialog box pops up, Select "Silicon Labs CP210x USB to UART Bridge(COM3)" for "Port", and then click "Browse...". Select the previous prepared microPython firmware "esp32-20210902-v1.17.bin". Check "Erase flash before installing" and "Flash mode", then click "Install" to wait for the prompt of finishing installation.

Note: If you fail to install the firmware, press the Boot on the ESP32 mainboard and click "Install" again.

					7
💦 ESP32 firm	nware installer			$\times$	
If you need Note that t at micropy	g allows installing or updating firm d to set other options, then please there are many variants of MicroP /thon.org/download doesn't work is look around in your device's	e use 'esptool' on the con Python for ESP devices. If k for your device, then the	nmand line. the firmware ere may exist	provided better	
Port	Silicon Labs CP210x USB to UAR	T Bridge (COM3)	~ Relo	ad	
Firmware	rted with Python/Python_Firmwa	are/esp32-20220117-v1.18	8.bin Brows	se	
	de				-
From in	mage file (keep) 🔿 Quad I/O (qio	p)			
◯ Dual I/	O (dio) O Dual Output	(dout)			
⊡ Erase fi	ash before installing		Install	Cancel	
😱 Open					×
$\leftarrow \rightarrow \checkmark \uparrow$	« 1. Python_Tutorial > 1. Preparation for Py	ython(Windows) > Python_Firmwa	are 🗸 i	ック Search P	ython_Firmware
Organize 👻 🛛 N	lew folder				= • 🔟 🕐
lene One Drive	^ Name	Date modified	Туре	Size	
💻 This PC	esp32-20210902-v1.17.bin	2/10/2022 11:23 AM	BIN File	1,492 KB	
3D Objects	_	0			
Desktop	×	×		2	
	File name: esp32-20210902-v1.17.bin			✓ bin-file	~
				Open	Cancel

This dialo If you nee Note that at microp	mware installer g allows installing or updating firmware on ESP32 using the most common setting ed to set other options, then please use 'esptool' on the command line. there are many variants of MicroPython for ESP devices. If the firmware provided ython.org/download doesn't work for your device, then there may exist better es look around in your device's documentation or at MicroPython forum.	×
Flash mo From Dual I,	Silicon Labs CP210x USB to UART Bridge (COM3)       Reload         on (Important) /Python_Firmware/esp32-20210902-v1.17.bin       Browse         ode       Image file (keep) Quad I/O (qio)         /O (dio)       Dual Output (dout)         lash before installing       Install	

D. Wait for the installation to be done, then click"Close"and"OK".

💦 ESP32 firm	nware installer	$\times$						
This dialog allows installing or updating firmware on ESP32 using the most common settings. If you need to set other options, then please use 'esptool' on the command line. Note that there are many variants of MicroPython for ESP devices. If the firmware provided at micropython.org/download doesn't work for your device, then there may exist better alternatives look around in your device's documentation or at MicroPython forum.								
Port	Silicon Labs CP210x USB to UART Bridge (COM3) V Reload							
Firmware	on (Important) /Python_Firmware/esp32-20210902-v1.17.bin Browse							
Flash mod								
From ir Dual I/0	nage file (keep) () Quad I/O (qio) O (dio) () Dual Output (dout)							
☑ Erase fla	Ariting at 0x0006521c (19 %)							

🔓 ESP32 fin	mware installer	$\times$
If you nee Note that at microp	g allows installing or updating firmware on ESP32 using the most common setting d to set other options, then please use 'esptool' on the command line. there are many variants of MicroPython for ESP devices. If the firmware provided ython.org/download doesn't work for your device, then there may exist better es look around in your device's documentation or at MicroPython forum.	5.
Port Firmware	Silicon Labs CP210x USB to UART Bridge (COM3) V Reload on (Important) /Python_Firmware/esp32-20210902-v1.17.bin Browse	
~	ode image file (keep) () Quad I/O (qio) 'O (dio) () Dual Output (dout)	
✓ Erase f	lash before installing	
<u>[</u>	Done! Install Close	

ត្រូ 📭	ionny								_		$\times$
File	🖫 Thonn	y options							×		
File Files This ⊕	General Which Micro Deta Con Con (loo If yo	Interpreter o Python (ESP3 ils necting via U nect your de k for your de u can't find i necting via V	ISB cable vice to th vice nam t, you m VebREPL	ne computer and ne, "USB Serial" o ay need to instal (EXPERIMENTAL	I select correspo r "UART"). I proper USB driv .):	vour code? nding port l ver first.	below		×	ľ	
	lf yc (im) < W Port	our device sup oort webrepl_ ebREPL > be or WebREPL	pports W setup), c low	ebREPL, first cor connect your cor to UART Bridge (	nect via serial, r nputer and devi				~		
								Install or update fi	irmware Cancel	\pytho	n.exe
											1

E. Close all dialog boxes, turn to main interface and click<sup>29</sup> "**STOP**". As shown in the illustration below:

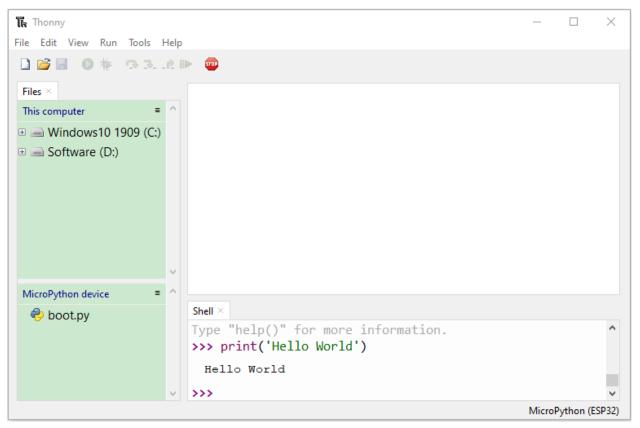
The Thonny	_		$\times$
File Edit View Run Tools Help			
🗋 🗃 📓 💿 🚸 👁 🗈 🗈 🖿 🔤 🛑 STOP Button			
Files ×			
This computer = ^			
🗉 🔿 Windows10 1909 (C:)			
🗉 🚔 Software (D:)			
/:Root directory			
Inter space of ESP32			
used to save files.			
MicroPython device Shell ×			
<pre>   boot.py   MicroPython v1.17 on 2021-09-02; ESP32 modul </pre>	e with	n ESP3	2
Type "help()" for more information.	C MICI	2013	
v >>>			~
	Micro	ython (E	SP32)

F. So far, all the preparations have been made.

# 6.5 5.Test Code

Testing the Shell commander

Enter "print('hello world')" in "Shell" and press Enter.



#### Running Online(Important)

ESP32 needs to be connected to a computer when it is run online. Users can use Thonny to writer and debug programs.

1. Open Thonny and click "Open...".

The Thonny	_		$\times$
File Edit View Run Tools Help			
□         □         ◆         •         ●         □           Files ×			
This computer = ^			
🗉 🛁 Software (D:)			
✓			
MicroPython device =			
🌏 boot.py Shell ×			
Type "help()" for more information.			^
<pre>&gt;&gt;&gt; print('Hello World')</pre>			
Hello World			
			~
	MicroP	ython (ES	SP32)

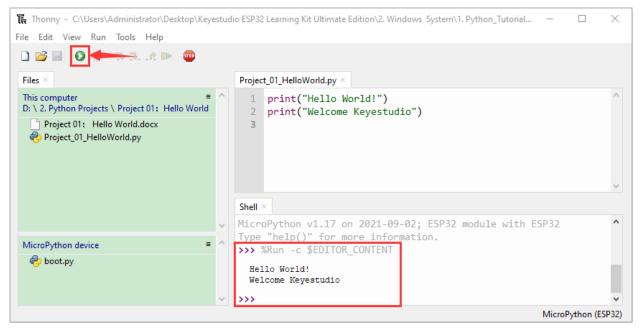
2. Click "This computer" on the newly pop-up window.



3. In the new dialog box, select "**Project\_01\_HelloWorld.py**" in "2. Windows System\1. Python\_Tutorial\2. Python Projects\Project 01Hello World" folder.

限 Open					×
	Python_Tutorial > 2. Python Projects > Proje	ct 01: Hello World	<b>∨ ບັ</b> ,⊃ Se	earch Project 01: Hel	lo W
Organize 🔻 New folde	r			· · · · · · · · · · · · · · · · · · ·	?
📰 Pictures 🛛 🖈 ^	Name	Date modified	Туре	Size	
<ul> <li>OneDrive</li> </ul>	Th Project_01_HelloWorld	10/15/2021 8:30 AM	Python file	1 KB	
💻 This PC					
🗊 3D Objects		X			
📃 Desktop 🗸 🗸				2	
File na	me: Project_01_HelloWorld		~ Pytho		$\sim$
			O	oen Canc	el:

4. Click C"Run current script" to execute the program and "Hello World" will be printed in "Shell".



**Note**When running online, if you press the Reset button of ESP32, user's code will not be executed again. If you wish to run the code automatically after resetting the code, please refer to the following Running Offline.

Running Offline(Important)

After ESP32 is reset, it runs the file boot.py in root directory first and then runs your code file, and finally it enters "**Shell**". Therefore, to make ESP32 execute user's programs after resetting, we need to add a guiding program in boot.py to execute user's code.

1. Move the program folder "2. Windows System/1. Python\_Tutorial/2. Python Projects" to disk(D) in advance with the path of "D:/2. Python Projects". Open "Thonny".

🌇 Thonny - <untitled> @ 1:1</untitled>		— 🗆	$\times$
File Edit View Run Tools Help			
🗋 💕 🛃 🛛 🐐 🗇 R. 12 🕨 👳			
Files ×		<until d=""> <math>\times</math></until>	
This computer	= ^	1	~
🗉 🛁 Windows10 1909 (C:)			
			$\sim$
	$\sim$	Shell ×	
MicroPython device	= ^		<b>^</b>
e boot.py		MicroPython v1.17 on 2021-09-02; ESP32 module with ESP.	
r,		Type "help()" for more information.	
	$\sim$	>>>	Υ.
		MicroPython (I	ESP32)

2. Expand "**Project 00: Boot**" in the "**2. Python Projects**" in the directory of disk(D), and double-click "**boot.py**", which is provided by us to enable programs in "**MicroPython device**" to run offline.

Thonny - D:\2. Python Projects\Project 00: Boot	\boo	t.py @	30 : 14 —	
File Edit View Run Tools Help				
🗋 😂 📕 🔕 🌞 🗇 3et 🕨 🥶				
Files $\times$		boot.p	y ×	
This computer ≡ D: \ 2. Python Projects	^	1 2	<pre>#!/opt/bin/lv_micropython import uos as os</pre>	^
🖃 <u> Project 00:</u> Boot		3	import uerrno as errno	
🚭 boot.py		4	<pre>iter = os.ilistdir()</pre>	
🕀 💫 Project 01: Hello World		5	$IS_{DIR} = 0 \times 4000$	
🕀 💫 Project 02: Turn On LED		6	$IS\_REGULAR = 0 \times 8000$	
Project 03: LED Flashing		7		
Project 04: Breathing LED		8	while True:	
⊕ ↓ Project 05: Traffic Lights     ⊕ ↓ Project 06: RGB LED		9 10	try:	
		11	entry = next(iter) filename = entry[0]	
	<b>.</b>	12	file type = entry[1]	
				*
MicroPython device =		Shell	<	
				^
		Micro	Python v1.17 on 2021-09-02; ESP32 module with ESP32	
		Type	"help()" for more information.	
	$\vee$	>>>		*
			MicroF	ython (ESP32)

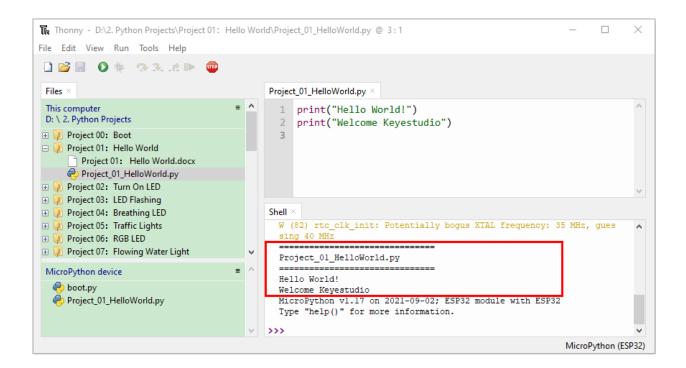
3. If you want your written programs to run offline, you need to upload "**boot.py**" we provided and all your codes to "MicroPython device" and press ESP32's Reset button. Here we use programs **Project 00** and **Project 01** as examples. Select "**boot.py**", right-click to select "**Upload to** /".

Thonny - D:\2. Python Projects\Project 00: Boot\boot	t.py @ 30:14 — 🗆	×
File Edit View Run Tools Help		
🗋 😂 🖩 🛛 🚸 🗇 3e 🕨 😑		
Files $\times$	boot.py $ imes$	
This computer = D: \ 2. Python Projects	1 #!/opt/bin/lv_micropython 2 import uos as os	^
	<pre>3 import uerrno as errno 4 iter = os.ilistdir() 5 IS_DIR = 0x4000 6 IS_REGULAR = 0x8000 7 8 while True:</pre>	
Project 05: Tra Move to Recycle Bin     Project 06: RG New directory     Project 07: Flo     Properties     Project 08: 1-[	<pre>9 try: 10 entry = next(iter) 11 filename = entry[0] 12 file type = entry[1]</pre>	~
MicroPython device	shell No code has been uploaded here.	
~	MicroPython v1.17 on 2021-09-02; ESP32 module with ESP32 Type "help()" for more information. >>>	×

4. Similarly, upload"Project\_01\_HelloWorld.py" to"MicroPython device".

限 Thonny - D:\2. Python Projects\Project	t 01: Hello	World\Proj	ect_01_	HelloWorld.py @ 3:1	-		×
File Edit View Run Tools Help							
🗋 📸 🖩 🗶 🔅 🗇 🕄 🕨	STOP						
Files $\times$		Proje	ct_01_H	lelloWorld.py ×			
This computer D: \ 2. Python Projects	=	^ 1 2		nt("Hello World!") nt("Welcome Keyestudio")			^
🕀 🔑 Project 00: Boot		3					
Project 01: Hello World		-					
Project 02: Turn On LED     Project 03: LED Flashing     Project 04: Breathing LED     Project 05: Traffic Lights     Project 06: RGB LED	Open in Tho Open in syste Configure .p J <mark>pload to /</mark> Nove to Rec New director	em default y files ycle Bin	app				<
MicroPython device P	roperties			boot.py has been upload	ed	her	e.
				non v1.17 on 2021-09-02; ESP32 module with [ Lp()" for more information.	SP32		~
					MicroP	ython (E	SP32)

5. Press ESP32's Reset button and in the box of the "Shell" below, you can see the code is executed.

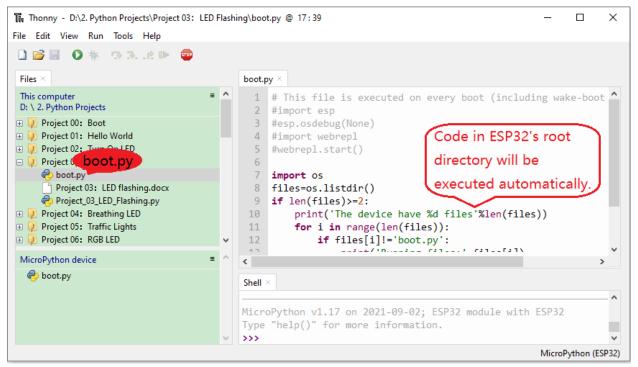


# 6.6 6.Thonny Common operations

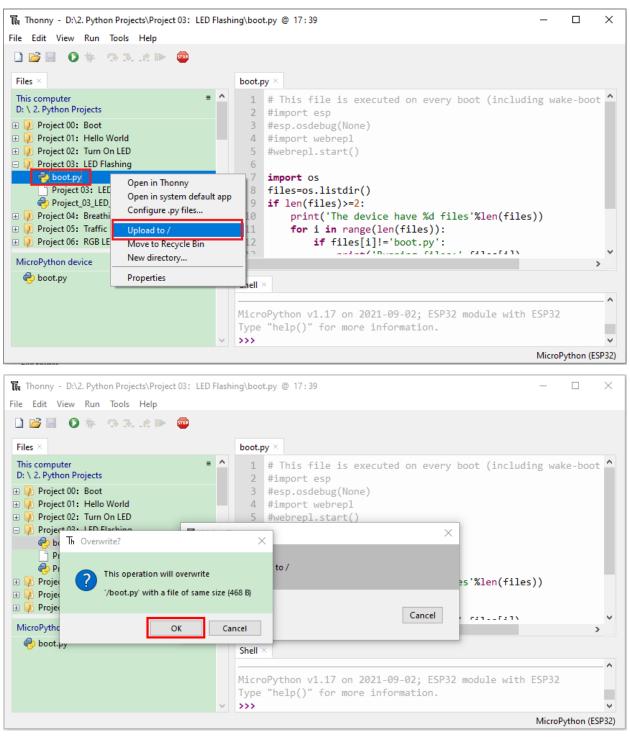
Upload the code to ESP32

For convenience, let's take the opertation on "boot.py" as an example.

If we add "**boot.py**" to each code directory, every time when ESP32 restarts, if there is a "**boot.py**" in the root directory, it will execute this code first.

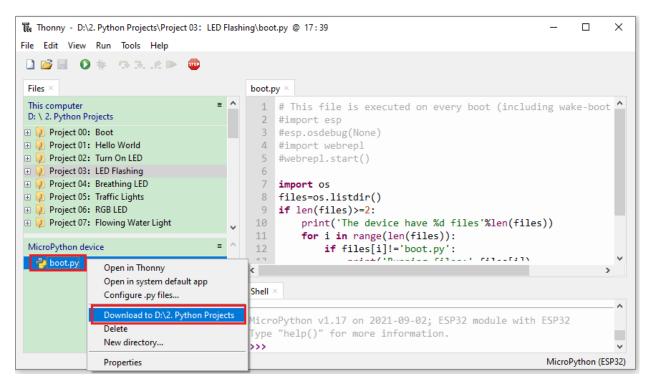


Select "**boot.py**" in "**Project 03LED Flashing**", right-click your mouse and select "**Upload to** /" to upload code to ESP32's root directory, select to click "**OK**".



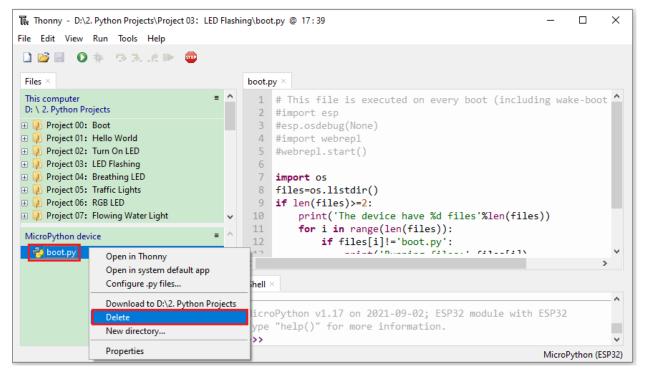
Downloading Code to Computer

Select "boot.py" in "MicroPython device", right-click to select "Download to ..." to download the code to your computer.



Delete files from ESP32 Root Directory

Select "**boot.py**" in "**MicroPython device**", right-click it and select "**Delete**" to delete "**boot.py**" from ESP32's root directory.



Select "boot.py" in "Project 03LED Flashing", right-click it and select "Move to Recycle Bin" to delete it from "Project 03LED Flashing".

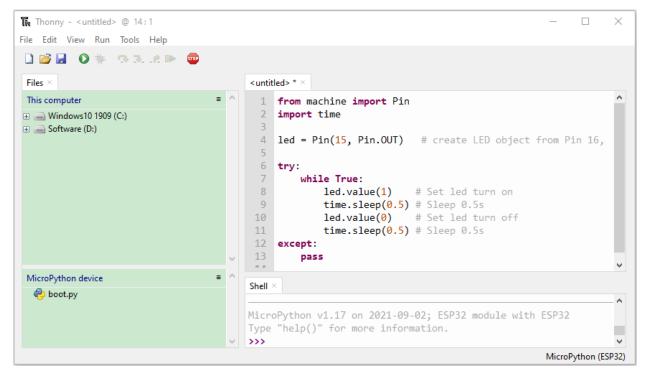
Thonny - D:\2. Python Projects\Project 03: LED Flashi File Edit View Run Tools Help	ng\boot.py @ 17:39 — 🗆	×
🗋 😂 📓 🔹 🔅 🦘 🐟 🦟 🕨 🚭	boot.py ×	
This computer D: \ 2. Python Projects D: \ 2. Python Project 00: Boot D: \ 2. Project 00: Boot D: \ 2. Project 01: Hello World D: \ 2. Project 02: Turn On LED D: \ 2. Project 03: LED Flashing	<pre>1 # This file is executed on every boot (including wake-boot 2 #import esp 3 #esp.osdebug(None) 4 #import webrepl 5 #webrepl.start() 6</pre>	. ^
<ul> <li>Project 03:</li> <li>Project 04:</li> <li>Project 05:</li> <li>Project 05:</li> <li>Project 05:</li> <li>Project 05:</li> <li>Project 05:</li> <li>Project 06:</li> <li>Project 06:</li> <li>Project 07:</li> <li>Project</li></ul>	<pre>7 import os 8 files=os.listdir() 9 if len(files)&gt;=2: 10 print('The device have %d files'%len(files)) 11 for i in range(len(files)): 12 if files[i]!='boot.py': 13 print('Purrier files' files[i])</pre>	¥
MicroPython device Properties	<pre>Shell × MicroPython v1.17 on 2021-09-02; ESP32 module with ESP32 Type "help()" for more information.</pre>	^
¥	>>> MicroPython (ES	♥ ♥32)

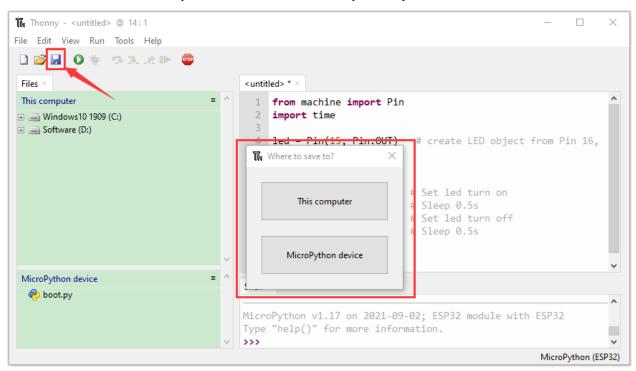
Creating and Saving the code

Click "File"  $\rightarrow$  "New" to create and write codes.

限 Thonny						_		×
File Edit View Ru	n Tools Help							
🗋 New	Ctrl+N	<b>.</b>						
💕 Open	Ctrl+0	1 -						
Recent files	+							
Close	Ctrl+W	= ^						
Close all	Ctrl+Shift+W							
Save	Ctrl+S							
Save All files	Ctrl+Alt+S							
Save as	Ctrl+Shift+S							
Save copy								
Move / rename								
Print	Ctrl+P							
Exit	Alt+F4							
MicroPython device		= ^						
			Shell ×					
			MicroPython v1.	17 on 2021-09.	-02: ESP32 modu	ule wit	h ESP	2
			Type "help()" f			AL WILL		
		~	>>>					~
						Micro	Python (E	SP32)

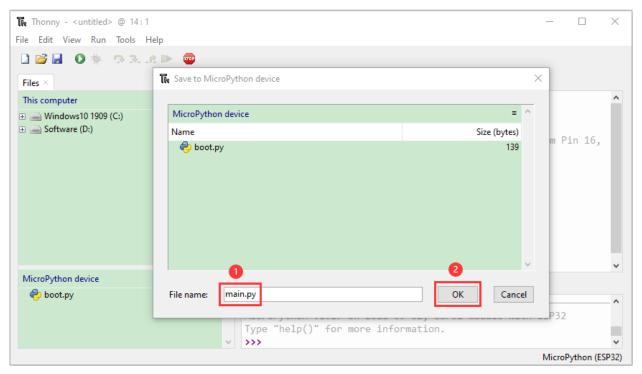
Enter codes in the newly opened file. Here we use codes of "Project\_03\_LED\_Flashing.py" as an example.





Click **Save**" on the menu bar, you can save the codes either to your computer or to ESP32.

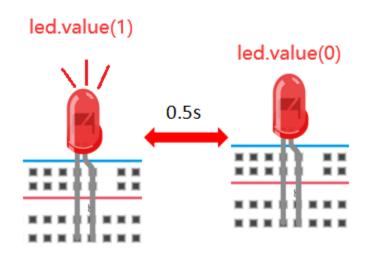
Select "MicroPython device", enter "main.py" in the newly pop-up window and click "OK".



You can see that the code has been uploaded to ESP32.



Disconnect and reconnect USB cable, and then you can see that LED is on for 0.5 s and then off for 0.5s.



### CHAPTER

### SEVEN

# **PYTHON PROJECT**

Click on the link to enter the Thonny tutorial: Thonny Tutorial

# 7.1 Download code files

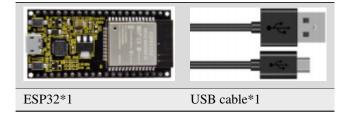
Click on the link to download the code file: Download Python Codes file

# 7.2 Project 01: Hello World

### 7.2.1 1. Introduction

For ESP32 beginners, we'll start with some simple things. In this project, you just need an ESP32 mainboard, USB cable and computer to complete "Hello World!" Project. It is not only a communication test for ESP32 mainboard and computer, but also a primary project for ESP32.

### 7.2.2 2. Components



### 7.2.3 3. Wiring

In this project, we use a USB cable to connect the ESP32 to the computer.



### 7.2.4 4. Running code online

To run ESP32 online, you need to connect it to computer, which allows you to use Thonny to compile or debug programs.

#### Advantages

- 1) You can use Thonny to compile or debug programs.
- 2) Through the "Shell" window, you can read the error information and output results generated during the running of the program and query related function information online to help improve the program.

#### Disadvantages

(1) To run ESP32 online, you have to be connected to a computer and run with Thonny.

(2) If ESP32 disconnects from computer, the program won't run again when they reconnect to each other.

#### **Basic Operation:**

1) Open Thonnyand click 📴 "	Ope	n".	
🏗 Thonny - <untitled> 🛛 1:1</untitled>		- 0	$\times$
File Edit View Run Tools Help	)		
🗋 📴 🖬 🛛 🐐 🔿 🛪 .et		🚭	
Files ×		<untitled> ×</untitled>	
This computer =	^	1	^
<ul> <li>■ Windows10 1909 (C:)</li> <li>■ Software (D:)</li> </ul>	~		
MicroPython device	^		
😔 boot.py			~
	~	Shell × MicroPython v1.17 on 2021-09-02; ESP32 module with ESP32 Type "help()" for more information. >>>	< >
		MicroPython (E	P32)

(2) On the newly pop-up window, click"This computer".

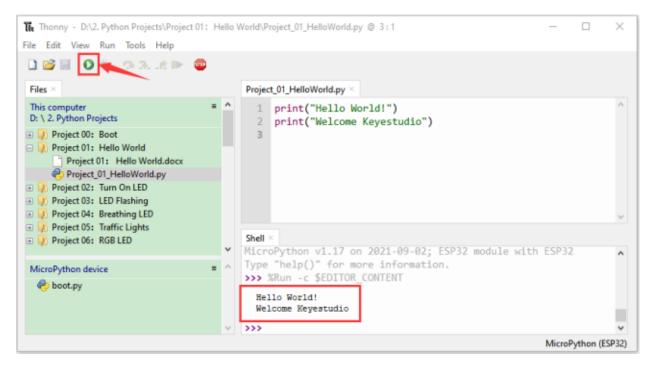
T	Where to open from?	$\times$
	This computer	
	MicroPython device	

2In the new dialog box, select "Project\_01\_HelloWorld.py", click "Open".

Codes used in this tutorial are saved in "2. Python Projects". If you haven't downloaded the code file, please click on the link to download it:Download Python Codes

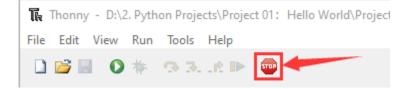
Te Open			×	<
← → × ↑ 📙 « Software (D:) → 2. Python Project	s > Project 01: Hello World v	ල 🔎 Search Pr	roject 01: Hello W	
Organize 🔻 New folder			💷 🔹 🔳 👔	
Downloads ^ Name ^	Date modified	Туре	Size	
Music The Project_01_HelloWorld	10/15/2021 8:30 AM	Python file	1 KB	
Pictures				
Videos Windows10 1905		0		
File name: Project_01_HelloWorld		<ul> <li>Pythor files</li> <li>Open</li> </ul>	Cancel	

(3) Click <sup>(2)</sup> "Run current script" to execute the program "Hello World!", "Welcome Keyestudio", which will be printed in the "**Shell**" window.



#### **Exit running online**

When running online, click "Stop /Restart Backend" or press "Ctrl+C" to exit the program.



### 7.2.5 5. Project code

```
print("Hello World!")
print("Welcome Keyestudio")
```

# 7.3 Project 02: Turn on LED

### 7.3.1 1.Introduction

In this project, we will show you how to light up the LED. We use the ESP32's digital pin to turn on the LED so that the LED is lit up.

## 7.3.2 2.Components

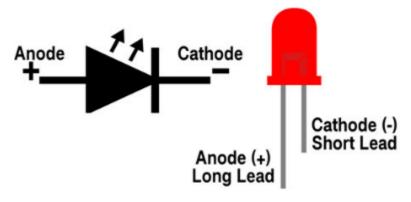
ESP32*1	Breadboard*1	USB Cable*1
	-((((())-	$\mathbf{X}$
Red LED*1	220 Resistor*1	Jumper Wire*2

# 7.3.3 3.Component knowledge

1LED:

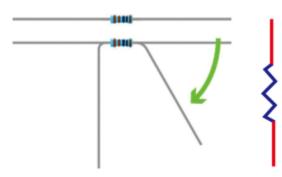


The LED is a semiconductor known as "light-emitting diode", which is an electronic device made from semiconducting materials(silicon, selenium, germanium, etc.). It has an anode and a cathode, the short lead is cathode, which connects to GND; the long lead is anode, which connects to 3.3V or 5V.



#### 2Five-color ring resistor

A resistor is an electronic component in a circuit that restricts or regulates the flow current flow. On the left is the appearance of the resistor and on the right is the symbol for the resistance in the circuit. Its unit is(). 1 m = 1000 k1k = 1000.



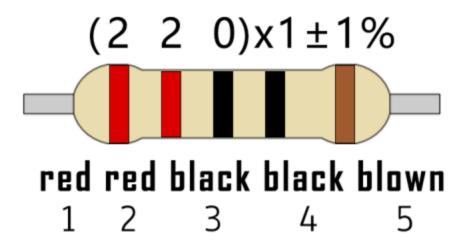
We can use resistors to protect sensitive components, such as LED. The strength of the resistance is marked on the body of the resistor with an electronic color code. Each color code represents a number, and you can refer to it in a resistance card.

- -Color 1 1st Digit.
- -Color 2 2nd Digit.
- -Color 3 3rd Digit.
- -Color 4 Multiplier.
- -Color 5 Tolerance.

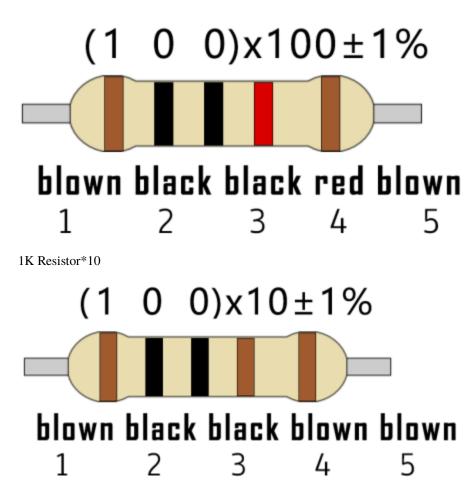
	1st Digit	2nd Digit	3rd Digit	Multiplier	Tolerance
Black		0	0	x1	
Brown	1	1	1	x10	± 1%
Red	2	2	2	x100	± 2%
Orange	3	3	3	x1K	± 3%
Yellow	4	4	4	x10K	± 4%
Green	5	5	5	x100K	±0.5%
Blue	6	6	6	x1M	±0.25%
Violet	7	7	7	x10M	±0.10%
Grey	8	8	8	x100M	±0.05%
White	9	9	9	x1G	
Gold				÷ 10	± 5%
Silver				÷ 100	± 10%

In this kit, we provide three Five-color ring resistor with different resistance values. Take three Five-color ring resistor as an example.

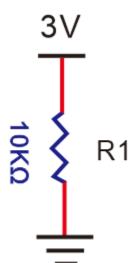
```
220 Resistor*10
```



10K Resistor\*10



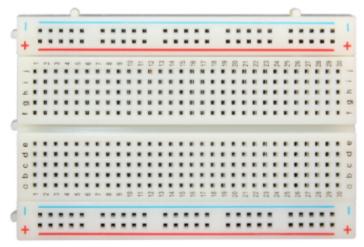
In the same voltage, there will be less current and more resistance. The connection between current(I), voltage(V), and resistance (a) can be expressed by the formula: I=U/R. In the figure below, if the voltage is 3V, the current through R1 is: I = U / R = 3 V / 10 K = 0.0003 A = 0.3 mA.



Don't connect a low resistance directly to the two poles of the power supply. as this will cause excessive current to damage the electronic components. Resistors do not have positive and negative poles.

#### **3Breadboard**

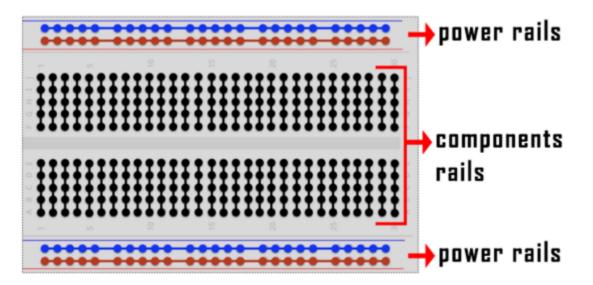
Breadboards are used to build and test circuits quickly before completing any circuit design. There are many holes in the breadboard that can be inserted into circuit components such as integrated circuit board and resistors. A typical breadboard is shown below



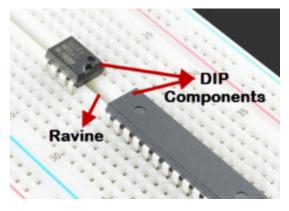
The breadboard has strips of metal, which run underneath the board and connect the holes on the top of the board. The metal strips are laid out as shown below. Note that the top and bottom rows of holes are connected horizontallywhile the remaining holes are connected vertically.

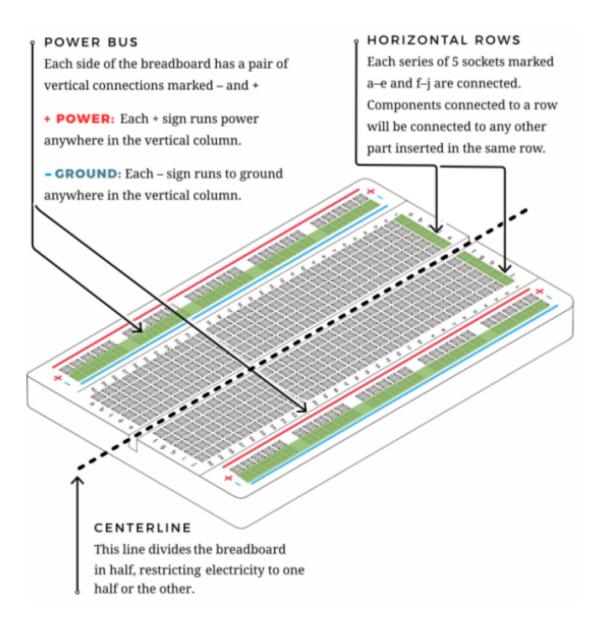


The first two rows (top) and the last two rows (bottom) of the breadboard are used for the positive pole (+) and negative pole (-) of the power supply respectively. The conductive layout of the breadboard is shown in the figure below:



When we connect DIP (Dual In-line Packages) components, such as integrated circuits, microcontrollers, chips and so on, we can see that a groove in the middle isolates the middle part, so the top and bottom of the groove is not connected. DIP components can be connected as shown in the following diagram:





#### (4) Power Supply

The ESP32 needs 3.3V-5V power supply. In this project, we connected the ESP32 to the computer by using a USB cable.

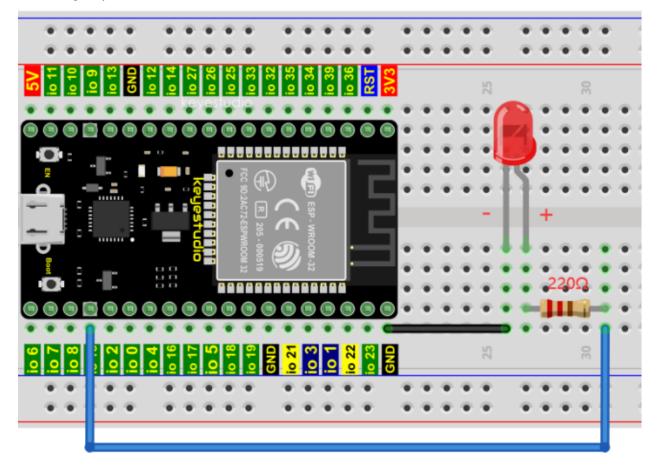


# 7.3.4 4.Wiring diagram

First, disconnect all power from the ESP32. Then build the circuit according to the wiring diagram. After the circuit is built and verified correct, connect the ESP32 to your computer by using a USB cable.

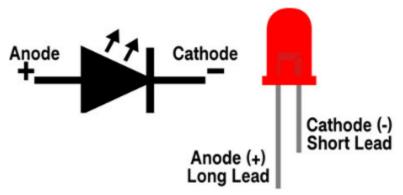
Note: Be careful to avoid short circuit when connecting 3.3V and GND!

**WARNING:** A short circuit can cause high current in your circuit, create excessive component heat and cause permanent damage to your hardware!

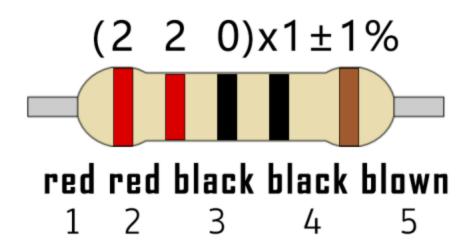


Note:

How to connect a LED



How to identify the 220 Five-color ring resistor



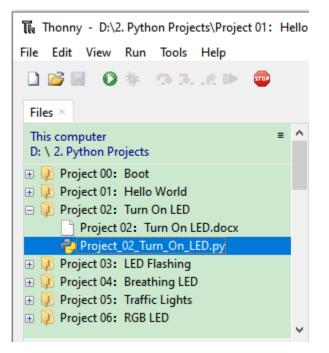
# 7.3.5 5.Project code

Codes used in this tutorial are saved in "2. Python Projects". If you haven't downloaded the code file, please click on the link to download it:Download Python Codes

2. Python Projects		_	$\Box$ $\times$
Share View			~ ?
→ This PC → Software (D:) → 2. Pyth	non Projects 🗸 Č	) 🔎 Search 2	2. Python
Name	Date modified	Туре	Size ^
Project 00: Boot	2/17/2022 10:21 AM	File folder	
Project 01: Hello World	2/17/2022 10:21 AM	File folder	
Project 02: Turn On LED	2/17/2022 11:10 AM	File folder	
Project 03: LED Flashing	2/17/2022 11:12 AM	File folder	
Project 04: Breathing LED	2/17/2022 10:21 AM	File folder	

Code running online:

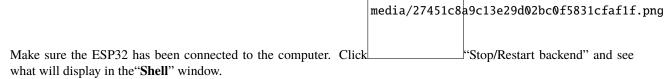
Open "Thonny"click "This computer"→"D:"→"2. Python Projects"→"Project 02Turn On LED".



Expand folder"Project 02Turn On LED"and click"Project\_02\_Turn\_On\_LED.py" to open it. As shown in the illustration below



```
from machine import Pin
import time
led = Pin(15, Pin.OUT)  # create LED object from Pin 15, Set Pin 15 to output
led.value(1)  # Set led turn on
```

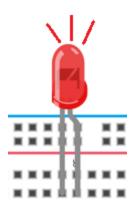


Thonny - D:\2. Python Projects\Project 02: Turn	On LED\Proj	ect_02_Turm_On_LED.py @ 8:1 -	$\times$
File Edit View Run Tools Help			
🗋 🗃 📕 🔕 🀐 🗇 3 t 🕨 🔤 🗲			
Files ×	Projec	t_02_Turn_On_LED.py ×	
This computer = D: \ 2. Python Projects	^ 1 2	from machine import Pin import time	^
Verify Project 00: Boot     Verify Project 01: Hello World     Verify Project 02: Turn On LED     Project 02: Turn On LED.docx     Project_02_Turn_On_LED.py	3 4 5 6 7	<pre>led = Pin(15, Pin.OUT) # create LED object from Pin 15, led.value(1) # Set led turn on</pre>	
	8		~
MicroPython device boot.py This indicates that the conection is successful.		oPython v1.17 on 2021-09-02; ESP32 module with ESP32 "help()" for more information.	^ ~
concertor is succession		MicroPython (ES	P32)

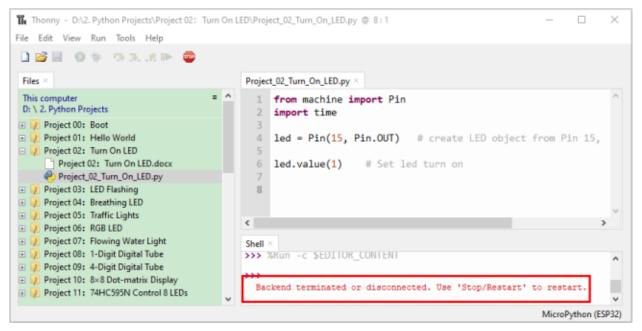
Click "Run current script", the code starts to be executed and the LED in the circuit lit up. Press "Ctrl+C" or click "Stop/Restart backend" to exit the program.



## led.value(1)



**Note**: This is the code running online. If you disconnect USB cable and repower ESP32 or press its reset button, LED is not bright and the following messages will be displayed in the "**Shell**" window of Thonny:

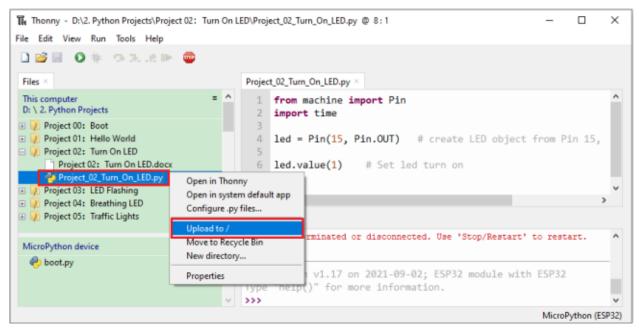


Code running offline (Upload the code to ESP32):

Make sure the ESP32 has been connected to the computer, click <sup>20</sup> "Stop/Restart backend".



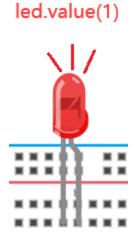
As shown below, right-click the file"Project\_02\_Turn\_On\_LED.py"select "Upload to /"to upload the code to ESP32.



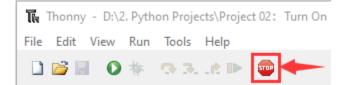
Upload "boot.py" in the same way.

Thonny - D:\2. Python Projects\Project 02: Turn	On LED\Project_02_Turn_On_LED.py @ 8:1 -	×
ile Edit View Run Tools Help		
🗋 💕 🖩 🛛 🕸 🕾 🔅 🕨 🤓		
Files ×	Project_02_Turn_On_LED.py ×	
This computer = D: \ 2. Python Projects	1 from machine import Pin 2 import time	^
Project 00: Boot     Doot.py     Open in Thonny     Project 01: He     Open in anthrough discharge	<pre>3 4 led = Pin(15, Pin.OUT) # create LED object from Pin 15 5</pre>	,
Open in system default app     Open in system default app	6 led.value(1) # Set led turn on 7	
Project 04: Bre Upload to / Project 05: Tra Move to Recycle Bin	8	
<ul> <li>Project 05: Tra Move to Recycle Bin</li> <li>Project 06: RG New directory</li> </ul>		
MicroPython device		
	¢	>
boot.py Project_02_Turn_On_LED.py	Shell ×	
		1
lake sure you have uploaded Proje	<pre>ct MicroPython v1.17 on 2021-09-02; ESP32 module with ESP32</pre>	
2_Turn_On_LED.py and boot.py he	e. Type "help()" for more information.	
	×	~
	MicroPython	(ESP32

Press the reset button of ESP32 and you can see LED is ON .



**Note**: Codes here is run offline. If you want to stop running offline and enter "**Shell**", just click <sup>1</sup> "Stop/Restart backend" in Thonny.

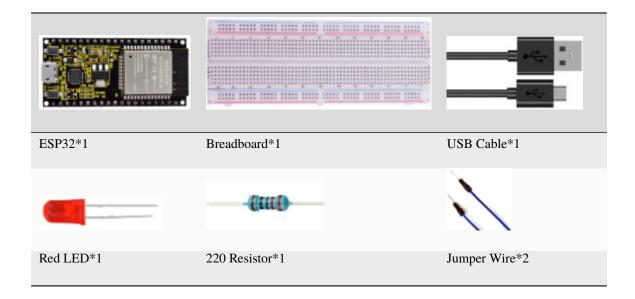


## 7.4 Project 03LED Flashing

### 7.4.1 1.Introduction

In this project, we will show you the LED flashing effect. We use the ESP32's digital pin to turn on the LED and make it flashing.

### 7.4.2 2.Components

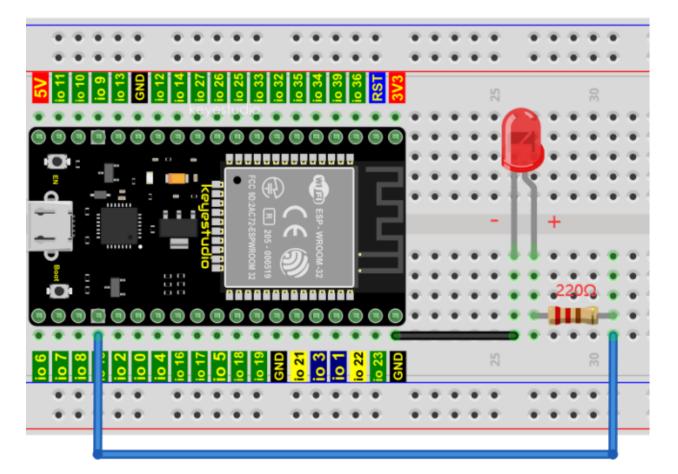


### 7.4.3 3.Wiring diagram

First, disconnect all power from the ESP32. Then build the circuit according to the wiring diagram. After the circuit is built and verified correct, connect the ESP32 to your computer using a USB cable.

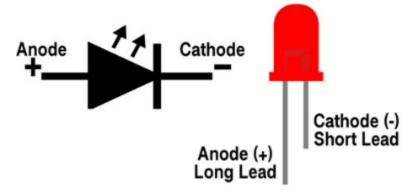
Note: Avoid any possible short circuits (especially connecting 3.3V and GND)!

**WARNING:** A short circuit can cause high current in your circuit, create excessive component heat and cause permanent damage to your hardware!

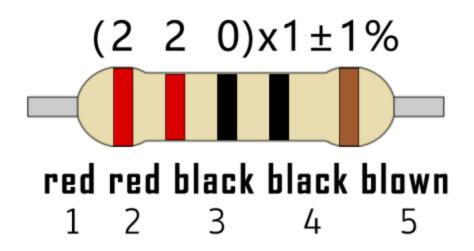


Note:

How to connect a LED



How to identify the 220 Five-color ring resistor



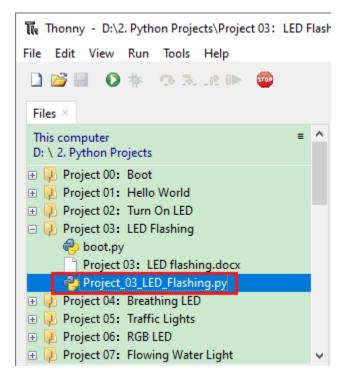
### 7.4.4 4.Project code

Codes used in this tutorial are saved in "2. Python Projects". If you haven't downloaded the code file, please click on the link to download it:Download Python Codes

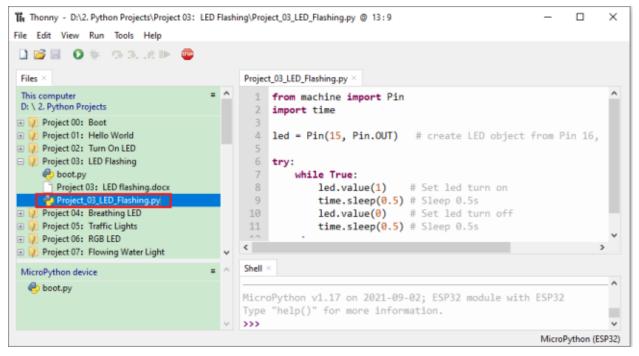
2. Python Projects		—	$\Box$ $\times$
Share View			~ ?
→ This PC → Software (D:) → 2. Pytł	non Projects 🗸 ぐ	, ○ Search 2	2. Python
Name	Date modified	Туре	Size ^
Project 00: Boot	2/17/2022 10:21 AM	File folder	
Project 01: Hello World	2/17/2022 10:21 AM	File folder	
Project 02: Turn On LED	2/17/2022 11:10 AM	File folder	
Project 03: LED Flashing	2/17/2022 11:12 AM	File folder	
Project 04: Breathing LED	2/17/2022 10:21 AM	File folder	

Code running online:

Open "Thonny" click "This computer" → "D:" → "2. Python Projects" → "Project 03LED Flashing".



Expand folder "Project 03LED Flashing" and click "Project\_03\_LED\_Flashing.py" to open it. As shown in the illustration below



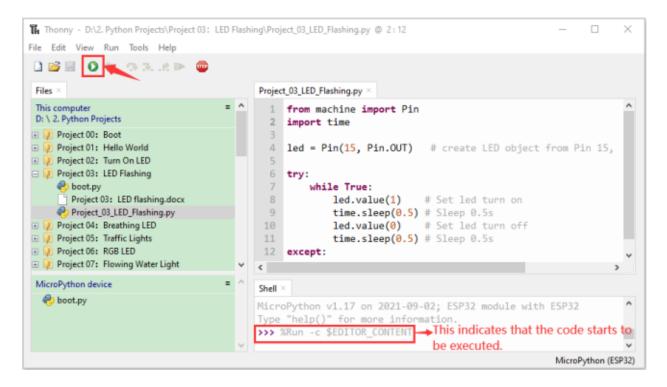
(continued from previous page)

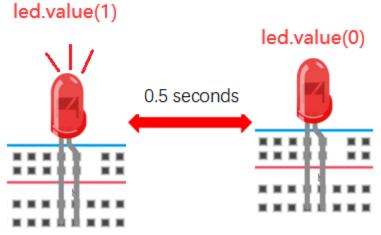
while True:	
<pre>led.value(1)</pre>	# Set led turn on
<pre>time.sleep(0.5)</pre>	# Sleep 0.5s
led.value(≬)	# Set led turn off
<pre>time.sleep(0.5)</pre>	# Sleep 0.5s
except:	
pass	

Make sure the ESP32 has been connected to the computer. Click <sup>we</sup> "Stop/Restart backend" and see what will display in the "Shell" window.

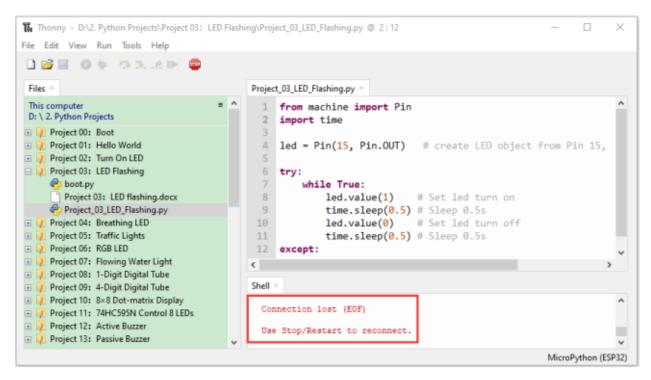


Click Click

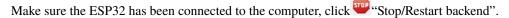


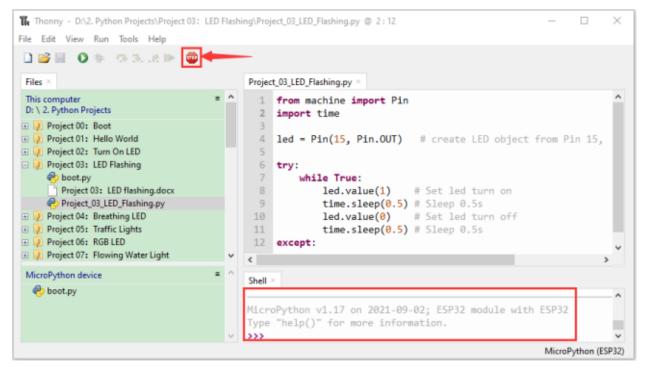


**Note**: This is the code running online. If you disconnect USB cable and repower ESP32 or press its reset button, the LED in the circuit stops flashing and the following messages will be displayed in the "Shell" window of Thonny:



Code running offlineUpload the code to ESP32





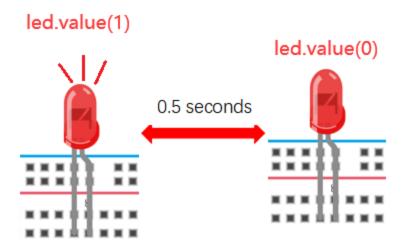
As shown below, right-click the file"Project\_03\_LED\_Flashing.py"select "Upload to /"to upload the code to ESP32.

) 💕 🖩 🛛 🐐 🔿 3e II iles ×	<ul> <li>••••••••••••••••••••••••••••••••••••</li></ul>	Projec	t_03_LED_Flashing.py ×
his computer ): \ 2. Python Projects	= ^	1 2	from machine import Pin import time
<ul> <li>Project 00: Boot</li> <li>Project 01: Hello World</li> <li>Project 02: Turn On LED</li> </ul>		3 4 5	<pre>led = Pin(15, Pin.OUT) # create LED object from Pin 15,</pre>
Project 03: LED Flashing boot.py Project 03: LED flashing.doc	×	6 7 8	<pre>try: while True: led.value(1)  # Set led turn on</pre>
<ul> <li>Project_03_LED_Flashing.py</li> <li>Project 04: Breathing LED</li> <li>Project 05: Traffic Lights</li> <li>Project 06: RGB LED</li> </ul>	Open in Tho Open in syst Configure .p	em defau	<pre>time.sleep(0.5) # Sleep 0.5s led.value(0) # Set led turn off time.sleep(0.5) # Sleep 0.5s</pre>
Project 07: Flowing Water Light	Upload to / Move to Rec	vole Bin	
	inforce to net	-	
AicroPython device	New directo	ry	

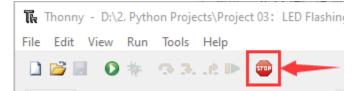
### Upload "boot.py" in the same way.

🕅 Thonny - D:\2. Python Projects\Project 03: LED F	ashing\Project_03_LED_Flashing.py @ 2:12 -	×
ile Edit View Run Tools Help		
🗋 😂 🖩 🛛 🕸 🕫 🕱 🕪 🖕 👘		
Files ×	Project_03_LED_Flashing.py ×	
This computer = D: \ 2. Python Projects □	1 from machine import Pin 2 import time 3	^
Copen in Thonny     Open in System default app     Project 01: Hel     Project 02: Tur     Project 03: LED	<pre>4 led = Pin(15, Pin.OUT) # create LED object from Pin 15 5 6 try: 7 while True:</pre>	,
Image: Project 04: Bre Upload to /	8 led.value(1) # Set led turn on	
Project 05: Trat Move to Recycle Bin	<pre>9 time.sleep(0.5) # Sleep 0.5s</pre>	
Project 06: RGE New directory	10 led.value(0) # Set led turn off	
Project 07: Flo     Properties	11 time.sleep(0.5) # Sleep 0.5s	
MicroPython device	12 except:	~
	<	>
boot.py Project_03_LED_Flashing.py	Shell ×	
Make sure you have uploaded		^
	MicroPython v1.17 on 2021-09-02; ESP32 module with ESP32	
Project_03_LED_Flashing.py and	Type "help()" for more information.	- 1
boot.py here.	>>>	~
	MicroPython	(ESP32

Press the reset button of ESP32 and you can see the LED is ON for 0.5 seconds and then OFF for 0.5 seconds, which repeats in an endless loop.



NoteCodes here is run offline. If you want to stop running offline and enter"Shell", just click <sup>1</sup> Stop/Restart backend"in Thonny.



## 7.5 Project 04: Breathing Led

### 7.5.1 1.Introduction

In previous studies, we know that LEDs have on/off state, so how to enter the intermediate state? How to output an intermediate state to make the LED half bright? That's what we're going to learn.

Breathing light, that is, LED is turned from off to on gradually, and gradually from on to off, just like "breathing". So, how to control the brightness of a LED? We will use ESP32's PWM to achieve this target.

### 7.5.2 2.Components

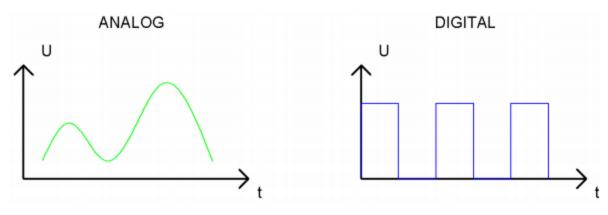
ESP32*1	Breadboard*1	USB Cable*1
	-(IIII)-	
Red LED*1	220 Resistor*1	Jumper Wire*2

### 7.5.3 3.Component knowledge



#### Analog & Digital:

An Analog Signal is a continuous signal in both time and value. On the contrary, a Digital Signal or discrete time signal is a time series consisting of a sequence of quantities. Most signals in life are analog signals. A familiar example of an Analog Signal would be how the temperature throughout the day is continuously changing and could not suddenly change instantaneously from 0°C to 10°C. However, Digital Signals can instantaneously change in value. This change is expressed in numbers as 1 and 0 (the basis of binary code). Their differences can more easily be seen when compared when graphed as below.



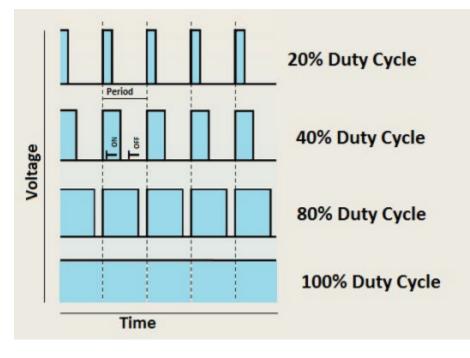
In practical application, we often use binary as the digital signal, that is a series of 0's and 1's. Since a binary signal only has two values (0 or 1), it has great stability and reliability. Lastly, both analog and digital signals can be converted into the other.

#### PWM

PWM, Pulse-Width Modulation, is a very effective method for using digital signals to control analog circuits. Common processors cannot directly output analog signals. PWM technology makes it very convenient to achieve this conversion (translation of digital to analog signals).

PWM technology uses digital pins to send certain frequencies of square waves, that is, the output of high levels and low levels, which alternately last for a while. The total time for each set of high levels and low levels is generally fixed, which is called the period (Note: the reciprocal of the period is frequency). The time of high level outputs are generally called "pulse width", and the duty cycle is the percentage of the ratio of pulse duration, or pulse width (PW) to the total period (T) of the waveform.

The longer the output of high levels last, the longer the duty cycle and the higher the corresponding voltage in the analog signal will be. The following figures show how the analog signal voltages vary between 0V-3V3 (high level is 3V3) corresponding to the pulse width 0%-100%:

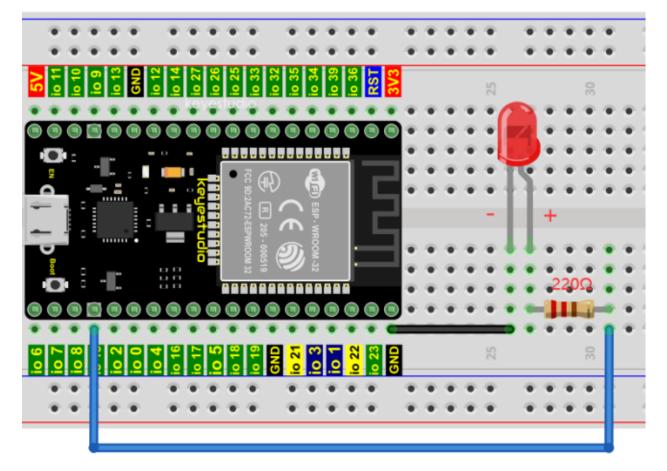


The longer the PWM duty cycle is, the higher the output power will be. Now that we understand this relationship, we can use PWM to control the brightness of an LED or the speed of DC motor and so on. It is evident from the above that PWM is not real analog, and the effective value of the voltage is equivalent to the corresponding analog. so, we can control the output power of the LED and other output modules to achieve different effects.

#### ESP32 and PWM:

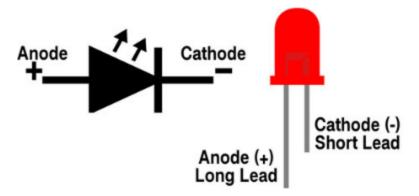
The ESP32 PWM controller has 8 independent channels, each of which can independently control frequency, duty cycle, and even accuracy. Unlike traditional PWM pins, the PWM output pins of ESP32 are configurable and they can be configured to PWM.

### 7.5.4 4.Wiring diagram

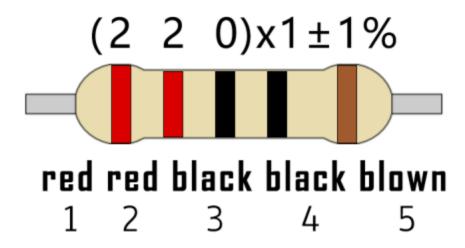


Note:

How to connect a LED



How to identify the 220 Five-color ring resistor



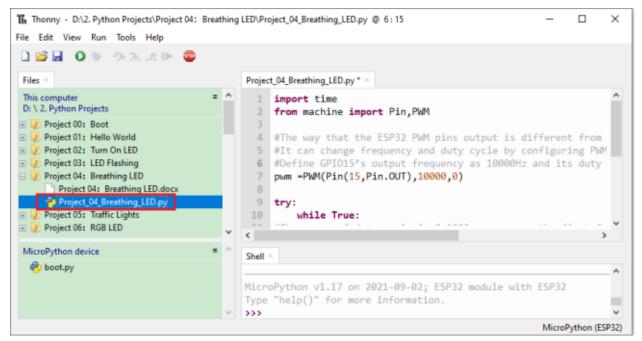
### 7.5.5 5. Project code

The design of this project makes the GP15 output PWM, and the pulse width gradually increases from 0% to 100%, and then gradually decreases from 100% to 0%.

Codes used in this tutorial are saved in "2. Python Projects". If you haven't downloaded the code file, please click on the link to download it:Download Python Codes

2. Python Projects			_		$\times$
Share View					~ ?
→ This PC → Software (D:) → 2. Pyth	non Projects 🗸 🗸 🗸	ې ب	Searc	h 2. Pytl	hon
Name	Date modified	Туре	:		Size ^
Project 00: Boot	2/17/2022 10:21 AN	1 Filet	folder		
Project 01: Hello World	2/17/2022 10:21 AN	1 Filet	folder		
Project 02: Turn On LED	2/17/2022 11:10 AN	1 Filet	folder		
Project 03: LED Flashing	2/17/2022 11:12 AN	1 Filet	folder		
Project 04: Breathing LED	2/17/2022 10:21 AN	1 Filet	folder		

Open"Thonny" click"This computer"  $\rightarrow$  "D:"  $\rightarrow$  "2. Python Projects"  $\rightarrow$  "Project 04Breathing Led", and double left-click "Project\_04\_Breathing\_LED.py".



# import time from machine import Pin,PWM

```
#The way that the ESP32 PWM pins output is different from traditionally controllers.
#It can change frequency and duty cycle by configuring PWM's parameters at the
→initialization stage.
#Define GPI015's output frequency as 10000Hz and its duty cycle as 0, and assign them to.
\rightarrow PWM.
pwm =PWM(Pin(15,Pin.OUT),10000,0)
try:
    while True:
#The range of duty cycle is 0-1023, so we use the first for loop to control PWM to
\rightarrow change the duty
#cycle value,making PWM output 0% -100%; Use the second for loop to make PWM output 100%-
\rightarrow 0\%.
        for i in range(0,1023):
            pwm.duty(i)
            time.sleep_ms(1)
        for i in range(0,1023):
            pwm.duty(1023-i)
            time.sleep_ms(1)
except:
#Each time PWM is used, the hardware Timer will be turned ON to cooperate it. Therefore,
\rightarrow after each use of PWM.
#deinit() needs to be called to turned OFF the timer. Otherwise, the PWM may fail to.
\rightarrow work next time.
    pwm.deinit()
```

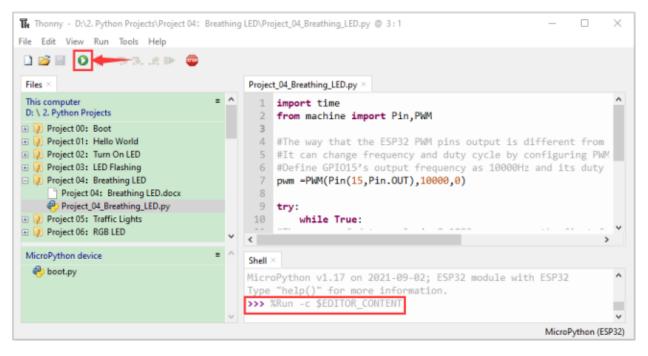
(continues on next page)

(continued from previous page)

### 7.5.6 6. Project result

Thonny - D:\2. Python Projects\Project 04: Breathin ile Edit View Run Tools Help	ng LED\Pro	oject_04_Breathing_LED.py @ 3 : 1 >
D 😂 🖬 🛛 🕸 🕾 🕸 🖬 🔤 🏎	-	
Files ×	Projec	t_04_Breathing_LED.py * ×
This computer = D: \ 2. Python Projects D: \ 2. Python Projects D: \ 2. Python Projects D: \ 2. Python Project 01: Hello World D: Project 01: Hello World D: Project 02: Turn On LED D: Project 02: Turn On LED D: Project 03: LED Flashing D: Project 04: Breathing LED D: Project 04: Breathing LED.docx D: Project 04: Breathing LED.docx D: Project 05: Traffic Lights D: Project 06: RGB LED	2 3 4 5 6 7 8 9 10	<pre>import time from machine import Pin,PWM #The way that the ESP32 PWM pins output is different from #It can change frequency and duty cycle by configuring PWM #Define GPI015's output frequency as 10000Hz and its duty pwm =PWM(Pin(15,Pin.OUT),10000,0) try:     while True:     while True:</pre>
MicroPython device = ^		<pre>x pPython v1.17 on 2021-09-02; ESP32 module with ESP32     "help()" for more information.</pre>

Click Click



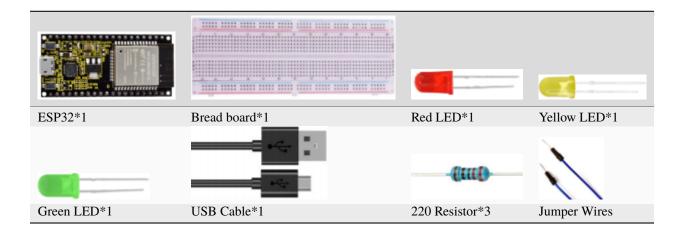


## 7.6 Project 05Traffic Lights

### 7.6.1 1.Introduction

Traffic lights are closely related to people's daily life, which generally show red, yellow, and green. Everyone should obey the traffic rules, which can avoid many traffic accidents. In this project, we will use ESP32 and some LEDs (red, green and yellow) to simulate the traffic lights.

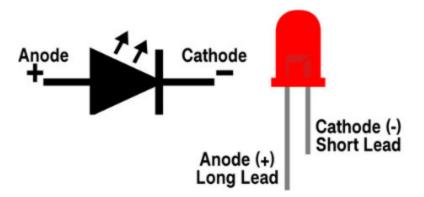
### 7.6.2 2.Components



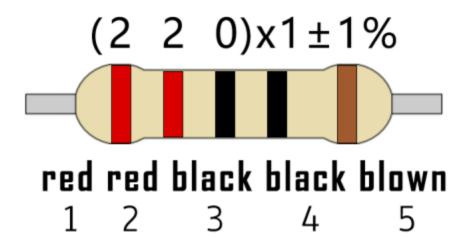
### 7.6.3 3. Wiring diagram

Note:

How to connect a LED



How to identify the 220 Five-color ring resistor

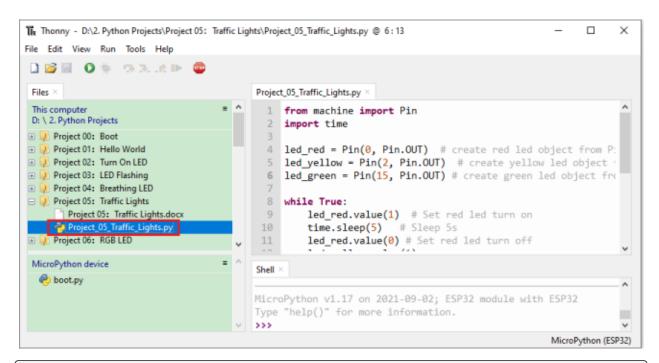


### 7.6.4 4. Project code

Codes used in this tutorial are saved in"**2. Python Projects**". If you haven't downloaded the code file, please click on the link to download it:Download Python Codes

2. Python Projects		-	
Share View			~ ?
→ This PC → Software (D:) → 2. Pyth	on Projects 🗸 さ	,○ Search 2.	Python
Name	Date modified	Туре	Size ^
Project 00: Boot	2/17/2022 10:21 AM	File folder	
Project 01: Hello World	2/17/2022 10:21 AM	File folder	
	2/17/2022 11:10 AM	File folder	
Project 03: LED Flashing	2/17/2022 11:12 AM	File folder	
Project 04: Breathing LED	2/17/2022 10:21 AM	File folder	

Open"Thonny" click"This computer"  $\rightarrow$  "D:"  $\rightarrow$  "2. Python Projects"  $\rightarrow$  "Project 05Traffic Lights". and double left-click "Project\_05\_Traffic\_Lights.py".



```
from machine import Pin
import time
```

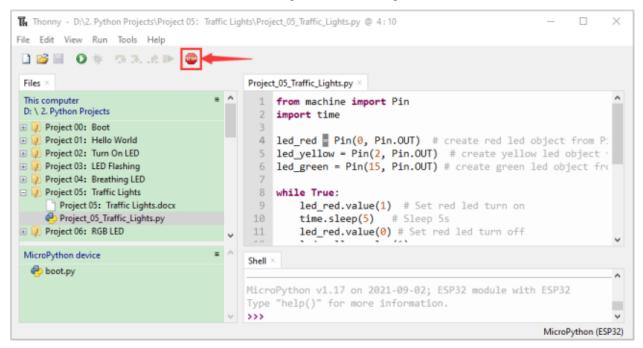
led\_red = Pin(0, Pin.OUT) # create red led object from Pin 0, Set Pin 0 to output led\_yellow = Pin(2, Pin.OUT) # create yellow led object from Pin 2, Set Pin 2 to output led\_green = Pin(15, Pin.OUT) # create green led object from Pin 15, Set Pin 15 to output

#### while True:

```
led_red.value(1) # Set red led turn on
time.sleep(5) # Sleep 5s
led_red.value(0) # Set red led turn off
led_yellow.value(1)
time.sleep(0.5)
led_yellow.value(0)
time.sleep(0.5)
led_yellow.value(1)
time.sleep(0.5)
led_yellow.value(0)
time.sleep(0.5)
led_yellow.value(1)
time.sleep(0.5)
led_yellow.value(0)
time.sleep((0.5))
led_green.value(1)
time.sleep(5)
led_green.value(♥)
```

### 7.6.5 5.Project result

Make sure the ESP32 has been connected to the computer, click "" "Stop/Restart backend".



Click <sup>O</sup> "Run current script", the code starts to be executed and you'll see are below:

First, the green light will be on for five seconds and then off;

Next, the yellow light blinks three times and then goes off;

Then, the red light goes on for five seconds and then goes off;

Repeat steps 1 to 3 above.

Press "Ctrl+C" or click "Stop/Restart backend" to exit the program.

🗋 😂 🖩 🚺 🏎 🤉 🤉 Lit 🕨 🥌 Files X		Projec	t_05_Traffic_Lights.py ×	
This computer         D: \ 2. Python Projects         Project 00: Boot         Project 01: Hello World         Project 02: Turn On LED         Project 03: LED Flashing         Project 04: Breathing LED         Project 05: Traffic Lights         Project 05: Traffic Lights.docx         Project 05: RGB LED	= ^	1 2 3 4 5 6 7 8 9 10 11	<pre>from machine import Pin import time led_red Pin(0, Pin.OUT) # create red led object from F led_yellow = Pin(2, Pin.OUT) # create yellow led object led_green = Pin(15, Pin.OUT) # create green led object fr while True:     led_red.value(1) # Set red led turn on     time.sleep(5) # Sleep 5s     led_red.value(0) # Set red led turn off</pre>	
MicroPython device	= ^	Type	× oPython v1.17 on 2021-09-02; ESP32 module with ESP32 "help()" for more information. %Run -c \$EDITOR_CONTENT	

## 7.7 Project 06: RGB LED

### 7.7.1 1.Introduction



RGB is composed of three colors (red, green and blue), which can emit different colors of light by mixing these three basic colors.

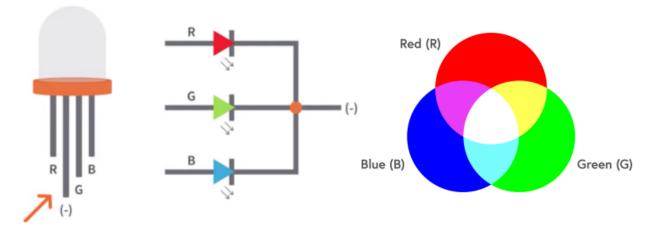
In this project, we will introduce the RGB and show you how to use ESP32 to control the RGB to emit different color light. RGB is pretty basic, but it's also a great way to learn the fundamentals of electronics and coding.

### 7.7.2 2.Components

ESP32*1	Breadboard*1	USB Cable*1
	-((((())-	
RGB LED*1	220 Resistor*3	Jumper Wires

### 7.7.3 3. Component knowledge

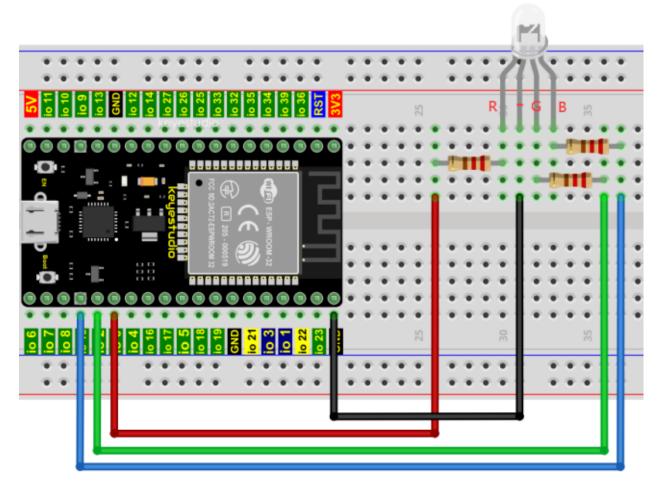
Most monitors adopt the RGB color standard, and all colors on a computer screen are a mixture of red, green and blue in varying proportions.



This RGB LED has 4 pins, each color (red, green, blue) and a common cathode, To change its brightness, we can use the PWM of the ESP32 pins, which can give different duty cycle signals to the RGB to produce different colors of light.

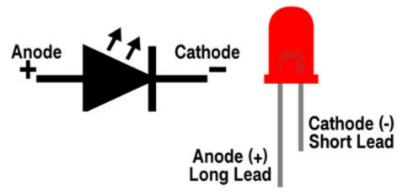
If we use three 10-bit PWM to control the RGB, in theory, we can create  $2 \ 10*210*210 = 1,073,741,824$  (1 billion) colors through different combinations.

### 7.7.4 4.Wiring diagram

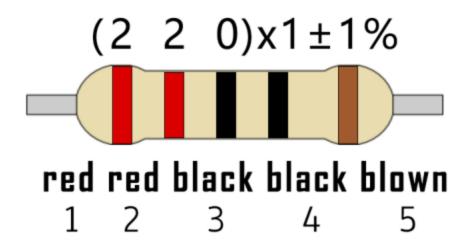


Note:

How to connect a LED



How to identify the 220 Five-color ring resistor

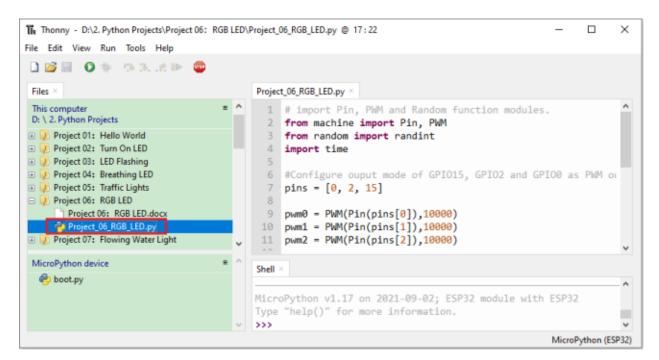


### 7.7.5 5.Project code

Codes used in this tutorial are saved in"**2. Python Projects**". If you haven't downloaded the code file, please click on the link to download it:Download Python Codes

2. Python Projects		— (	
Share View			~ 🕐
→ This PC → Software (D:) → 2. Pyth	non Projects 🗸 진		Python
Name	Date modified	Туре	Size ^
Project 00: Boot	2/17/2022 10:21 AM	File folder	
Project 01: Hello World	2/17/2022 10:21 AM	File folder	
Project 02: Turn On LED	2/17/2022 11:10 AM	File folder	
Project 03: LED Flashing	2/17/2022 11:12 AM	File folder	
Project 04: Breathing LED	2/17/2022 10:21 AM	File folder	

Open"Thonny", click "This computer"  $\rightarrow$  "D:"  $\rightarrow$  "2. Python Projects"  $\rightarrow$  "Project 06RGB LED", and double left-click "Project\_06\_RGB\_LED.py".



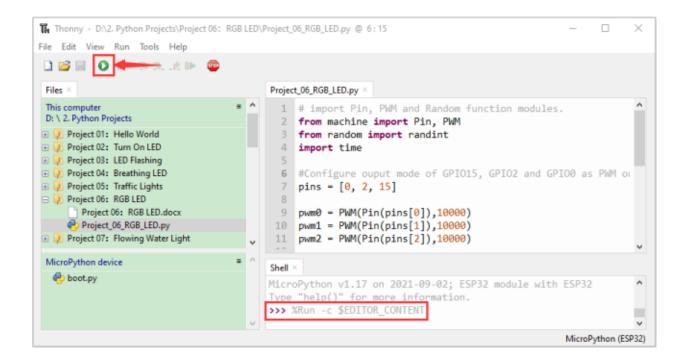
```
## import Pin, PWM and Random function modules.
from machine import Pin, PWM
from random import randint
import time
#Configure ouput mode of GPI015, GPI02 and GPI00 as PWM output and PWM frequency as_
\rightarrow 10000Hz.
pins = [0, 2, 15]
pwm0 = PWM(Pin(pins[0]), 10000)
pwm1 = PWM(Pin(pins[1]), 10000)
pwm2 = PWM(Pin(pins[2]), 10000)
#define a function to set the color of RGBLED.
def setColor(r, g, b):
    pwm0.duty(1023-r)
    pwm1.duty(1023-g)
    pwm2.duty(1023-b)
try:
    while True:
        red = randint(0, 1023)
        green = randint((0, 1023))
        blue = randint(0, 1023)
        setColor(red, green, blue)
        time.sleep_ms(200)
except:
    pwm0.deinit()
    pwm1.deinit()
    pwm2.deinit()
```

### 7.7.6 Project result



File Edit View Run Tools Help	-	
Files × This computer D: \ 2. Python Projects Project 01: Hello World Project 02: Turn On LED Project 03: LED Flashing Project 04: Breathing LED Project 05: Traffic Lights Project 05: RGB LED Project 06: RGB LED.docx Project 06: RGB LED.docx Project 07: Flowing Water Light	-	<pre>Project_06_RGB_LED.py × 1 # import Pin, PWM and Random function modules. 2 from machine import Pin, PWM 3 from random import randint 4 import time 5 6 #Configure ouput mode of GPI015, GPI02 and GPI00 as PWM or 7 pins = [0, 2, 15] 8 9 pwm0 = PWM(Pin(pins[0]),10000) 10 pwm1 = PWM(Pin(pins[1]),10000) 11 pwm2 = PWM(Pin(pins[2]),10000) </pre>
MicroPython device	•	Shell × MicroPython v1.17 on 2021-09-02; ESP32 module with ESP32 Type "help()" for more information.

Click Click

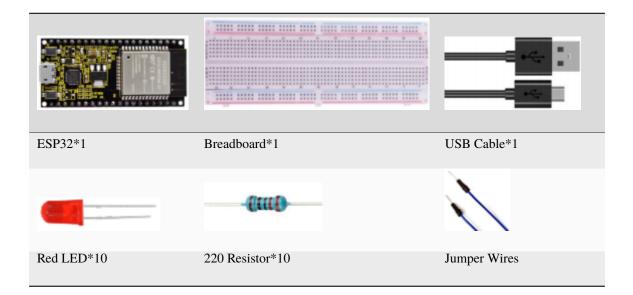


## 7.8 Project 07: Flowing Water Light

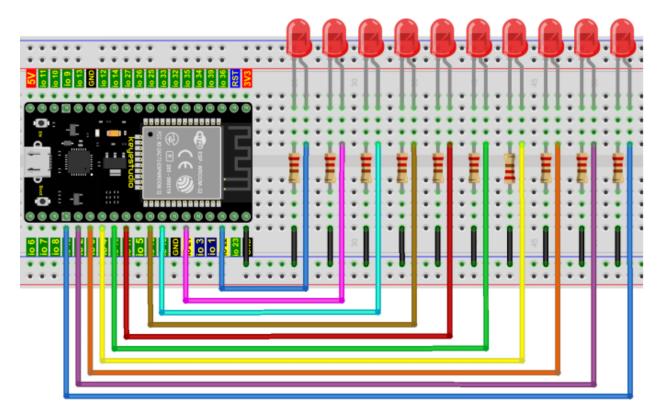
### 7.8.1 1.Introduction

In our daily life, we can see many billboards composed of different colors of LED. They constantly change the light (like water) to attract customers' attention. In this project, we will use ESP32 to control 10 leds to achieve the effect of flowing water.

### 7.8.2 2.Components

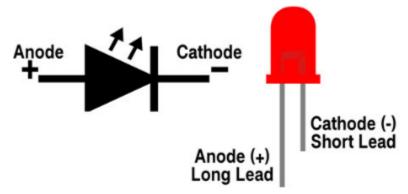


### 7.8.3 3.Wiring diagram :

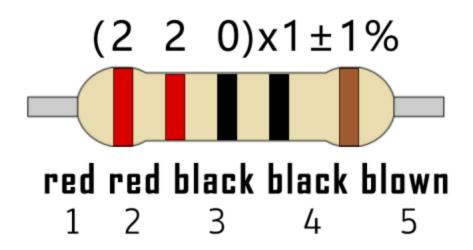


Note:

How to connect a LED



How to identify the 220 Five-color ring resistor



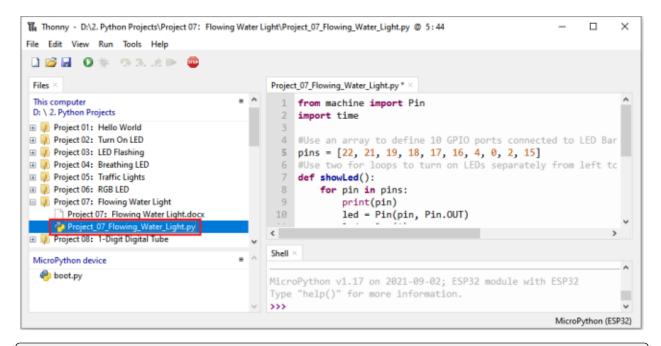
### 7.8.4 4.Project code

This project is designed to make a flowing water lamp. Which are these actions: First turn LED #1 ON, then turn it OFF. Then turn LED #2 ON, and then turn it OFF... and repeat the same to all 10 LEDs until the last LED is turns OFF. This process is repeated to achieve the "movements" of flowing water.

Codes used in this tutorial are saved in"2. Python Projects". If you haven't downloaded the code file, please click on the link to download it:Download Python Codes

2. Python Projects		_	$\Box$ $\times$
Share View			~ ?
→ This PC → Software (D:) → 2. Pytł	non Projects 🛛 🗸 🗸	ට 🔎 Sear	ch 2. Python
Name	Date modified	Туре	Size ^
Project 00: Boot	2/17/2022 10:21 AM	A File folder	
Project 01: Hello World	2/17/2022 10:21 AM	A File folder	
Project 02: Turn On LED	2/17/2022 11:10 AM	A File folder	
Project 03: LED Flashing	2/17/2022 11:12 AM	A File folder	
Project 04: Breathing LED	2/17/2022 10:21 AM	A File folder	

Open"Thonny" click"This computer"  $\rightarrow$  "D:"  $\rightarrow$  "2. Python Projects"  $\rightarrow$  "Project 07Flowing Water Light", and double left-click "Project\_07\_Flowing\_Water\_Light.py".

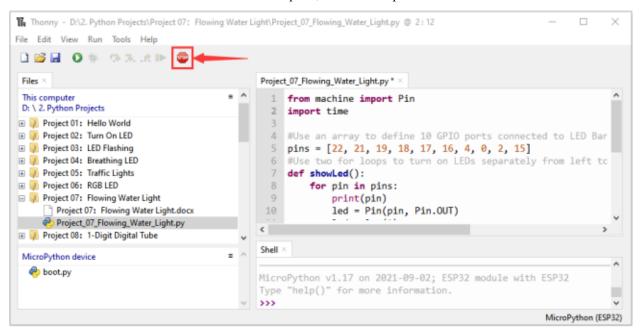


# from machine import Pin import time

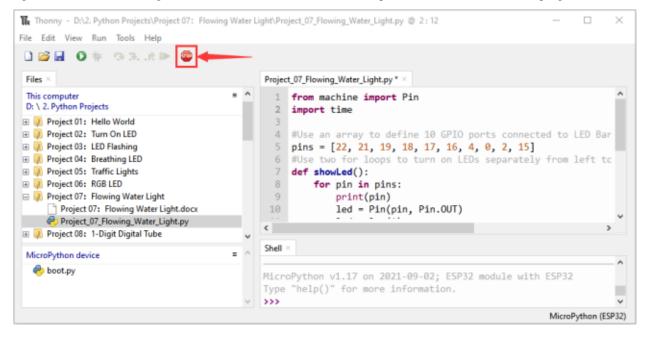
```
#Use an array to define 10 GPIO ports connected to LED Bar Graph for easier operation.
pins = [22, 21, 19, 18, 17, 16, 4, 0, 2, 15]
#Use two for loops to turn on LEDs separately from left to right and then back from.
\rightarrow right to left
def showLed():
    for pin in pins:
        print(pin)
        led = Pin(pin, Pin.OUT)
        led.value(1)
        time.sleep_ms(100)
        led.value(0)
        time.sleep_ms(100)
    for pin in reversed(pins):
        print(pin)
        led = Pin(pin, Pin.OUT)
        led.value(1)
        time.sleep_ms(100)
        led.value(0)
        time.sleep_ms(100)
while True:
    showLed()
```

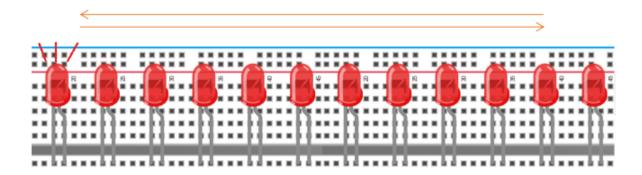
### 7.8.5 5. Project result

Make sure the ESP32 has been connected to the computer, click "" "Stop/Restart backend".



Click "Run current script", the code starts to be executed and you'll see that 10 LEDs will light up from left to right and then back from right to left. Press"Ctrl+C" or click "Stop/Restart backend" to exit the program.



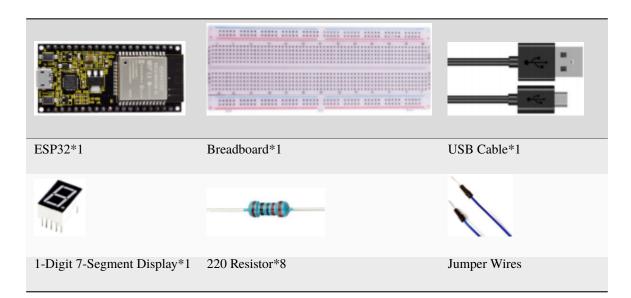


## 7.9 Project 081-Digit Digital Tube

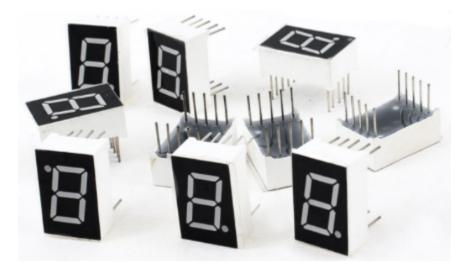
### 7.9.1 1.Introduction

A 1-Digit 7-Segment Display is an electronic display device that displays decimal numbers. It is widely used in digital clocks, electronic meters, basic calculators and other electronic devices that display digital information. Eventhough they may not look modern enough, they are an alternative to more complex dot matrix displays and are easy to use in limited light conditions and strong sunlight. In this project, we will use ESP32 to control 1-Digit 7-segment display displays numbers.

### 7.9.2 2.Components



# 7.9.3 3. Component knowledge

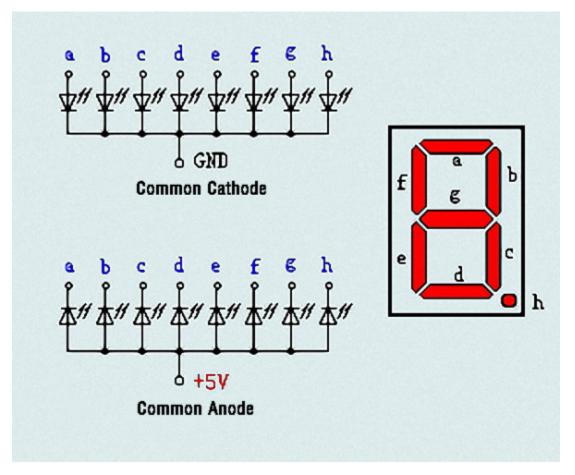


#### 1-Digit 7-Segment Display principle:

Digital tube display is a semiconductor light emitting device, its basic unit is a light-emitting diode (LED). The digital tube display can be divided into 7-segment display and 8-segment display according to the number of segments. The 8-segment display has one more LED unit than the 7-segment display (used for decimal point display). Each segment of the 7-segment display is a separate LED. According to the connection mode of the LED unit, the digital tube can be divided into a common anode digital tube and a common cathode digital tube.

In the common cathode 7-segment display, all the cathodes (or negative electrodes) of the segmented LEDs are connected together, so you should connect the common cathode to GND. To light up a segmented LED, you can set its associated pin to "HIGH".

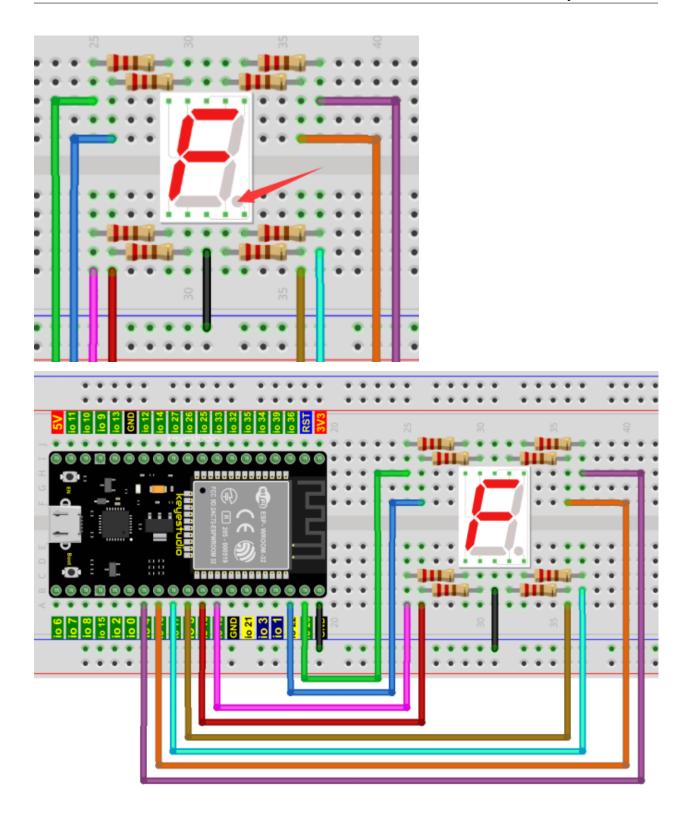
In the common anode 7-segment display, the LED anodes (positive electrodes) of all segments are connected together, so you should connect the common anode to"+5V". To light up a segmented LED, you can set its associated pin to"LOW".



Each part of the digital tube is composed of an LED. So when you use it, you also need to use a current limiting resistor. Otherwise, the LED will be damaged. In this experiment, we use an ordinary common cathode one-digit digital tube. As we mentioned above, you should connect the common cathode to GND. To light up a segmented LED, you can set its associated pin to "HIGH".

# 7.9.4 4.Wiring diagram

Note: The direction of the 7-segment display inserted into the breadboard is consistent with the wiring diagram, with one more point in the lower right corner.



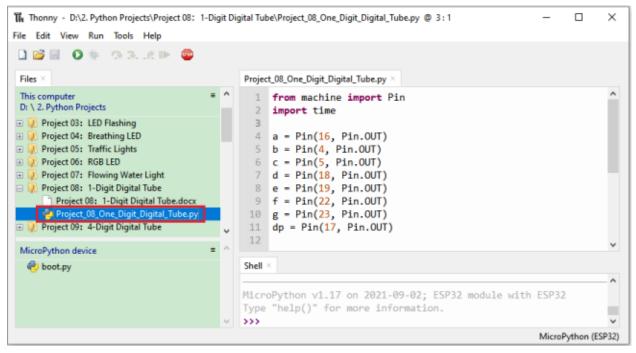
# 7.9.5 5.Project code

The digital display is divided into 7 segments, and the decimal point display is divided into 1 segment. When certain numbers are displayed, the corresponding segment will be lit. For example, when the number 1 is displayed, segments b and c will be turned on.

Codes used in this tutorial are saved in "2. Python Projects". If you haven't downloaded the code file, please click on the link to download it:Download Python Codes

2. Python Projects		_	$\square$ ×
Share View			~ ?
→ This PC → Software (D:) → 2. Pyth	on Projects v さ	,	Python
Name	Date modified	Туре	Size ^
Project 00: Boot	2/17/2022 10:21 AM	File folder	
Project 01: Hello World	2/17/2022 10:21 AM	File folder	
Project 02: Turn On LED	2/17/2022 11:10 AM	File folder	
Project 03: LED Flashing	2/17/2022 11:12 AM	File folder	
Project 04: Breathing LED	2/17/2022 10:21 AM	File folder	

Open"Thonny" click"This computer"  $\rightarrow$  "D:"  $\rightarrow$  "2. Python Projects"  $\rightarrow$  "Project 081-Digit Digital Tube", and double left-click "Project\_08\_One\_Digit\_Digital\_Tube.py".

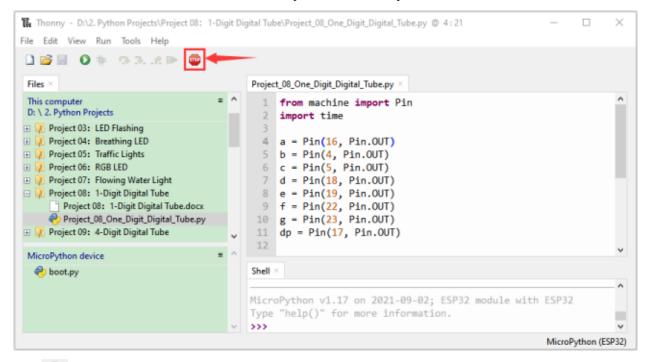




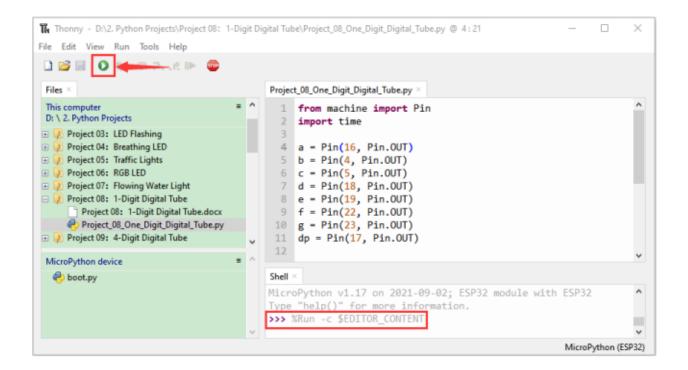
```
c = Pin(5, Pin.OUT)
d = Pin(18, Pin.OUT)
e = Pin(19, Pin.OUT)
f = Pin(22, Pin.OUT)
g = Pin(23, Pin.OUT)
dp = Pin(17, Pin.OUT)
pins = [Pin(id,Pin.OUT) for id in [16, 4, 5, 18, 19, 22, 23, 17]]
def show(code):
    for i in range(0, 8):
        pins[i].value(~code & 1)
        code = code >> 1
#Select code from 0 to 9
mask_digits = [0xc0, 0xf9, 0xa4, 0xb0, 0x99, 0x92, 0x82, 0xf8,0x80, 0x90]
for code in reversed(mask_digits):
    show(code)
   time.sleep(1)
```

# 7.9.6 6.Project result

Make sure the ESP32 has been connected to the computer, click <sup>22</sup> "Stop/Restart backend".



Click Click

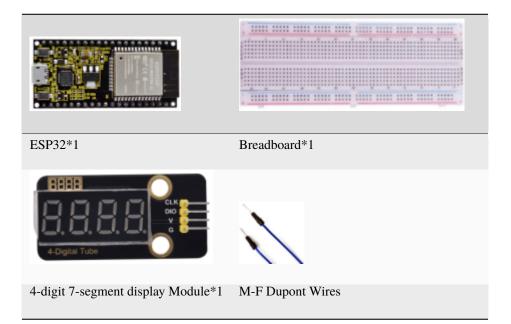


# 7.10 Project 094-Digit Digital Tube

### 7.10.1 1.Introduction

A 4-digit 7-segment display is a very practical display device and it is used for devices such as electronic clocksscore counters and the number of people in the park. Because of the low price, easy to use, more and more projects will use 4 Digit 7-segment display. In this project, we use ESP32 control 4-digit 7-segment display to display four digits.

# 7.10.2 2.Components



# 7.10.3 3.Component knowledge

**TM1650** \*\*4-digit 7-segment display\*\*It is a 12-pin 4-digit 7-segment display module with clock dots. The driver chip is TM1650 which only needs 2 signal lines to enable the microcontroller to control the 4-digit 7-segment display. The control interface level can be 5V or 3.3V.

#### Specifications of 4-bit 7-segment display module:

Working voltage: DC 3.3V-5V

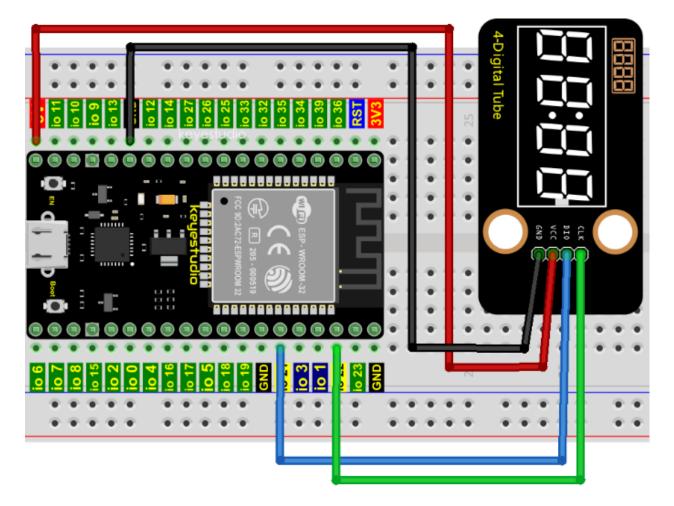
Maximum current: 100MA

Maximum power: 0.5W

#### Schematic diagram of 4-digit 7-segment display module:

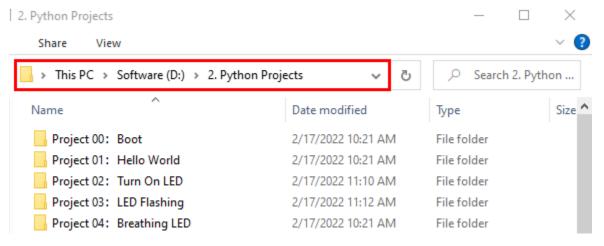


# 7.10.4 4.Wiring diagram

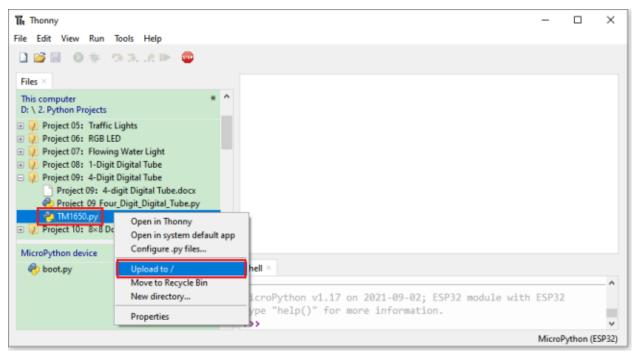


### 7.10.5 5. Project code

Codes used in this tutorial are saved in"**2. Python Projects**". If you haven't downloaded the code file, please click on the link to download it:Download Python Codes



Open"Thonny" click"This computer"  $\rightarrow$  "D:"  $\rightarrow$  "2. Python Projects"  $\rightarrow$  "Project 094-Digit Digital Tube". and then double left-click "Project\_09\_Four\_Digital\_Tube.py".



```
from machine import Pin
import time
```

```
## definitions for TM1650
ADDR_DIS = 0x48 #mode command
ADDR_KEY = 0x49 #read key value command
## definitions for brightness
```

```
BRIGHT_DARKEST = 0
BRIGHT_TYPICAL = 2
BRIGHTEST
              = 7
on = 1
off = 0
## number:0~9
NUM = [0x3f, 0x06, 0x5b, 0x4f, 0x66, 0x6d, 0x7d, 0x07, 0x7f, 0x6f]
## DIG = [0x68,0x6a,0x6c,0x6e]
DIG = [0x6e,0x6c,0x6a,0x68]
DOT = [0, 0, 0, 0]
clkPin = 22
dioPin = 21
clk = Pin(clkPin, Pin.OUT)
dio = Pin(dioPin, Pin.OUT)
DisplayCommand = 0
```

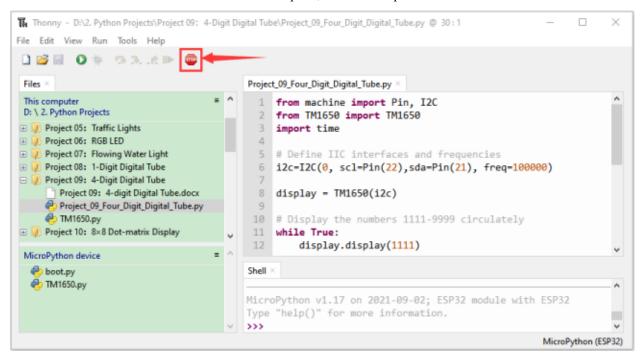
```
def writeByte(wr_data):
   global clk,dio
   for i in range(8):
        if(wr_data & 0x80 == 0x80):
            dio.value(1)
        else:
            dio.value(♥)
        clk.value(♥)
        time.sleep(0.0001)
        clk.value(1)
        time.sleep(0.0001)
        clk.value(♥)
        wr_data <<= 1
   return
def start():
   global clk,dio
   dio.value(1)
   clk.value(1)
   time.sleep(0.0001)
   dio.value(♥)
   return
def ack():
   global clk,dio
   dy = ≬
   clk.value(♥)
   time.sleep(0.0001)
   dio = Pin(dioPin, Pin.IN)
   while(dio.value() == 1):
        time.sleep(0.0001)
        dy += 1
        if(dy>5000):
            break
   clk.value(1)
   time.sleep(0.0001)
   clk.value(♥)
   dio = Pin(dioPin, Pin.OUT)
   return
def stop():
   global clk,dio
   dio.value(♥)
   clk.value(1)
   time.sleep(0.0001)
   dio.value(1)
   return
def displayBit(bit, num):
   global ADDR_DIS
   if(num > 9 and bit > 4):
        return
```

```
start()
    writeByte(ADDR_DIS)
    ack()
    writeByte(DisplayCommand)
    ack()
    stop()
    start()
   writeByte(DIG[bit-1])
    ack()
    if(DOT[bit-1] == 1):
        writeByte(NUM[num] | 0x80)
    else:
        writeByte(NUM[num])
    ack()
    stop()
    return
def clearBit(bit):
    if(bit > 4):
        return
    start()
    writeByte(ADDR_DIS)
    ack()
    writeByte(DisplayCommand)
    ack()
    stop()
    start()
    writeByte(DIG[bit-1])
    ack()
    writeByte(0x00)
    ack()
    stop()
    return
def setBrightness(b = BRIGHT_TYPICAL):
    global DisplayCommand, brightness
    DisplayCommand = (DisplayCommand & 0x0f)+(b<<4)
    return
def setMode(segment = 0):
    global DisplayCommand
    DisplayCommand = (DisplayCommand & 0xf7)+(segment<<3)</pre>
    return
def displayOnOFF(OnOff = 1):
    global DisplayCommand
    DisplayCommand = (DisplayCommand & 0xfe)+OnOff
    return
def displayDot(bit, OnOff):
    if(bit > 4):
```

```
return
    if(0n0ff == 1):
        DOT[bit-1] = 1;
    else:
        DOT[bit-1] = 0;
    return
def InitDigitalTube():
    setBrightness(2)
    setMode(♥)
    displayOnOFF(1)
    for _ in range(4):
        clearBit(_)
    return
def ShowNum(num): #0~9999
    displayBit(1,num%10)
    if(num < 10):
        clearBit(2)
        clearBit(3)
        clearBit(4)
    if(num > 9 and num < 100):
        displayBit(2,num//10%10)
        clearBit(3)
        clearBit(4)
    if(num > 99 and num < 1000):
        displayBit(2,num//10%10)
        displayBit(3,num//100%10)
        clearBit(4)
    if(num > 999 and num < 10000):
        displayBit(2,num//10%10)
        displayBit(3,num//100%10)
        displayBit(4,num//1000)
InitDigitalTube()
while True:
    #displayDot(1,on)
                         # on or off, DigitalTube.Display(bit,number); bit=1---4 _
\rightarrownumber=0---9
    for i in range(0,9999):
        ShowNum(i)
        time.sleep(0.01)
```

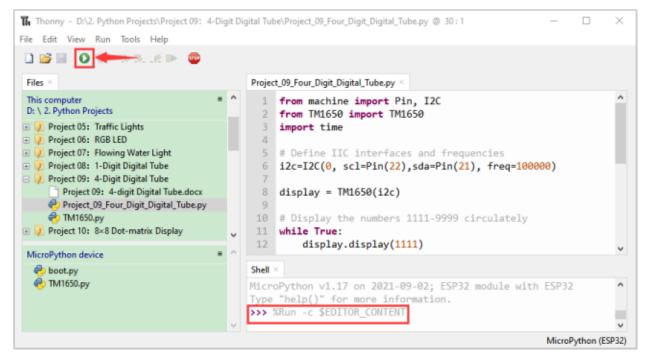
# 7.10.6 6. Project result

Make sure the ESP32 has been connected to the computer, click "" "Stop/Restart backend".



Click Click

digits the number increments by one and repeat these actions in an infinite loop. Press "Ctrl+C" or click <sup>100</sup> "Stop/Restart backend" to exit the program.

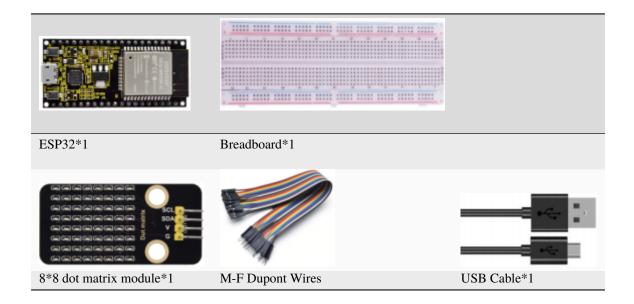


# 7.11 Project 108×8 Dot-matrix Display

# 7.11.1 1.Introduction

Dot matrix display is an electronic digital display device that can display information on machine, clocks, public transport departure indicators and many other devices. In this project, we will use ESP32 control 8x8 LED dot matrix to display patterns.

# 7.11.2 2.Components



# 7.11.3 3.Component knowledge

#### 8\*8 dot matrix module

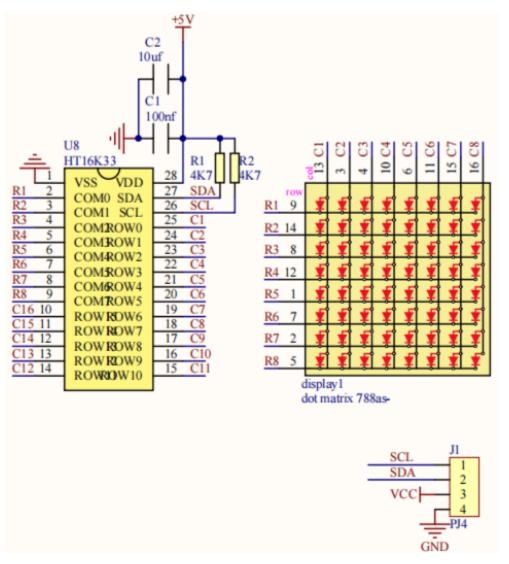
The 8\*8 dot matrix is composed of 64 LEDs, and each LED is placed at the intersection of a row and a column. When using the single chip microcomputer to drive an 8\*8 dot matrix, we need 16 digital ports in total, which greatly wastes the data of the single chip microcomputer.

To this end, we specially designed this module, using the HT16K33 chip to drive an 8\*8 dot matrix, and only need to use the I2C communication port of the MCU to control the 8\*8 dot matrix, which greatly saving the MCU resources.

#### Specifications of 8\*8 dot matrix module

- Working voltage: DC 5V
- Current: 200MA
- Maximum power: 1W

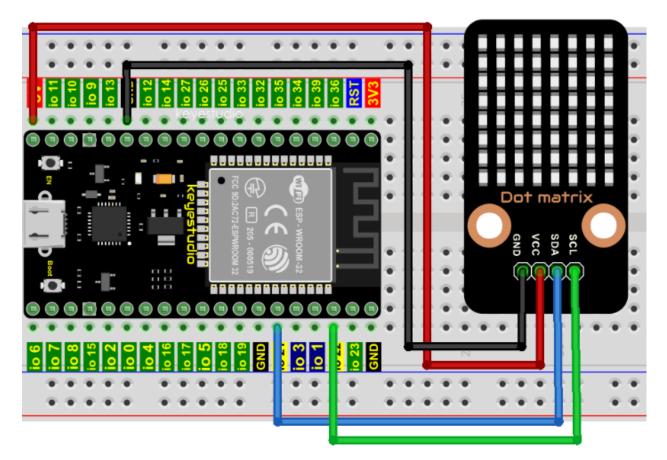
#### Schematic diagram of 8\*8 dot matrix module



Some modules have three DIP switches that you can toggle at will. These switches are used to set the I2C communication address, the setting method is as follows. The module has fixed the communication address. A0, A1 and A2 are connected to GND, and the address is 0x70.

A0 (1)	A1 (2)	A2 (3)	A0 (1)	A1 (2)	A2 (3)	A0 (1)	A1 (2)	A2 (3)	
0(OFF)	0 (OFF)	0 (OFF)	1 (ON)	0 (OFF)	0 (OFF)	0 (OFF)	1 (ON)	0(OFF)	
0X70			0X71			0X72			
A0 (1)	A1 (2)	A2 (3)	A0 (1)	A1 (2)	A2 (3)	A0 (1)	A1 (2)	A2 (3)	
1 (ON)	1 (ON)	0 (OFF)	0 (OFF)	0 (OFF)	1 (ON)	1 (ON)	0 (OFF)	1 (ON)	
0X7	0X73			0X74			0X75		
A0 (1)	A1 (2)	A2 (3)	A0 (1)	A1 (2)	A2 (3)				
0(OFF)	1 (ON)								
0X7	5		0X77						

# 7.11.4 4.Wiring diagram



# 7.11.5 5.Project code

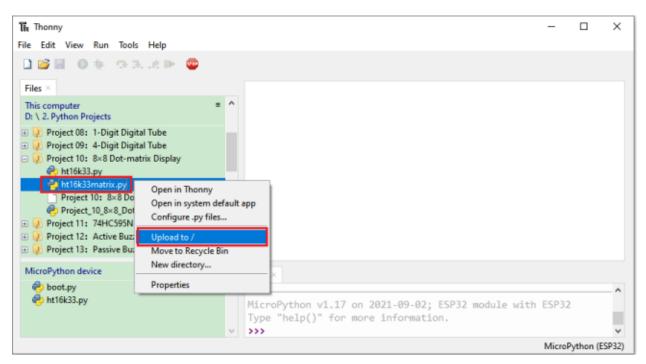
Codes used in this tutorial are saved in "2. Python Projects". If you haven't downloaded the code file, please click on the link to download it:Download Python Codes

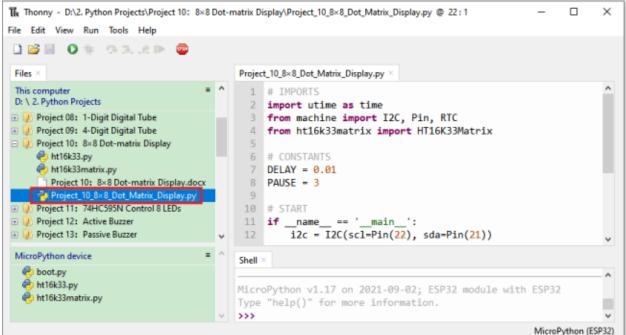
2. Python Projects		_	$\Box$ $\times$
Share View			~ 🕐
→ This PC → Software (D:) → 2. Pytł	non Projects 🗸 🖑	,○ Search 2	. Python
Name	Date modified	Туре	Size ^
Project 00: Boot	2/17/2022 10:21 AM	File folder	
Project 01: Hello World	2/17/2022 10:21 AM	File folder	
Project 02: Turn On LED	2/17/2022 11:10 AM	File folder	
Project 03: LED Flashing	2/17/2022 11:12 AM	File folder	
Project 04: Breathing LED	2/17/2022 10:21 AM	File folder	

Open"Thonny"click"This computer"→"D:"→"2. Python Projects"→"Project 108×8 Dot-matrix Display".

Select "ht16k33matrix.py" and "ht16k33.py", right-click your mouse to select "Upload to /", wait for "ht16k33matrix.py" and "ht16k33.py" to be uploaded to ESP32, and then double left-click "Project\_10\_8×8\_Dot\_Matrix\_Display.py".

The Thonny			-		×
File Edit View Run Tools Help					
🗋 🐸 🖩 🛛 🌴 🧐 3e 🕨 🚭					
Files ×					
This computer D: \ 2. Python Projects	• ^				
Project 08: 1-Digit Digital Tube					
Project 09: 4-Digit Digital Tube     Vertical State     Vertical					
Copen in Thonny th16k33matris Denin system defaul	арр				
Project 10: 8 Configure .py files					
Project 11: 74HC Upload to /		J			
⊕ Project 12: Activ Move to Recycle Bin     ⊕ Project 13: Passi New directory					
MicroPython device Properties		Shell ×			
😔 boot.py					^
		MicroPython v1.17 on 2021-09-02; ESP32 module with Type "help()" for more information.	ESP32	2	÷
	$\vee$	>>>			$\sim$
			MicroF	ython (i	SP32





```
## IMPORTS
import utime as time
from machine import I2C, Pin, RTC
from ht16k33matrix import HT16K33Matrix
## CONSTANTS
DELAY = 0.01
PAUSE = 3
```

```
## START
if __name__ == '__main__':
    i2c = I2C(scl=Pin(22), sda=Pin(21))
    display = HT16K33Matrix(i2c)
    display.set_brightness(2)

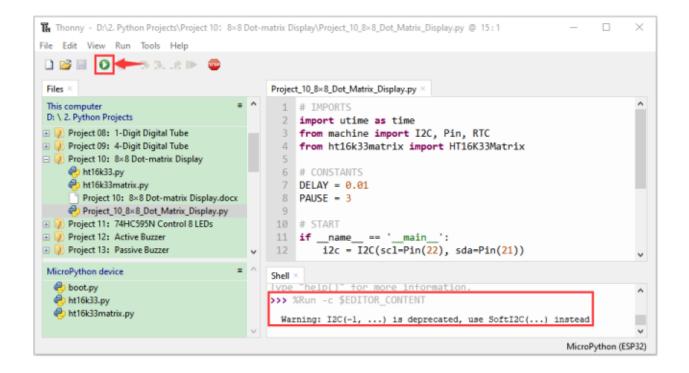
# Draw a custom icon on the LED
    icon = b"\x00\x66\x00\x00\x18\x42\x3c\x00"
    display.set_icon(icon).draw()
# Rotate the icon
    display.set_angle(0).draw()
    time.sleep(PAUSE)
```

### 7.11.6 6.Project result

🟗 Thonny - D:\2. Python Projects\Project 10: 8×8 Dot-matrix Display\Project\_10\_8×8\_Dot\_Matrix\_Display.py @ 15:1 × File Edit View Run Tools Help 🗋 💕 🖩 🛛 🕸 🔿 3. ..e 🕨 Files × Project\_10\_8×8\_Dot\_Matrix\_Display.py This computer 1 # IMPORTS D: \ 2. Python Projects import utime as time 🗉 뒞 Project 08: 1-Digit Digital Tube 3 from machine import I2C, Pin, RTC 🗉 📜 Project 09: 4-Digit Digital Tube 4 from ht16k33matrix import HT16K33Matrix Project 10: 8×8 Dot-matrix Display ht16k33.py 6 # CONSTANTS ht16k33matrix.py 7 DELAY = 0.01Project 10: 8×8 Dot-matrix Display.docx 8 PAUSE = 3 Project\_10\_8×8\_Dot\_Matrix\_Display.py 9 Project 11: 74HC595N Control 8 LEDs 10 # START 11 if \_ 🗉 🕖 Project 12: Active Buzzer name --- ' main ': 🗉 🕖 Project 13: Passive Buzzer i2c = I2C(scl=Pin(22), sda=Pin(21)) MicroPython device = Shell boot.py ht16k33.py MicroPython v1.17 on 2021-09-02; ESP32 module with ESP32 ht16k33matrix.py Type "help()" for more information. >>> MicroPython (ESP32)

Click Click

Make sure the ESP32 has been connected to the computer, click <sup>999</sup> "Stop/Restart backend".



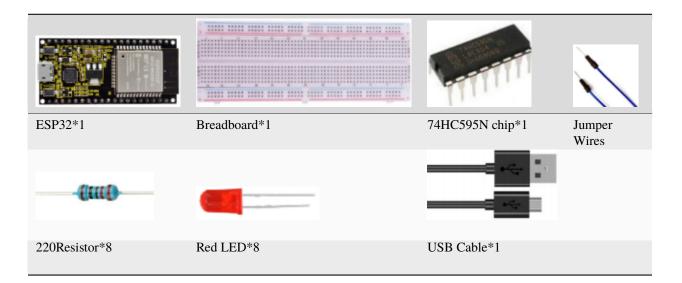
# 7.12 Project 1174HC595N Control 8 LEDs

### 7.12.1 1.Introduction

In previous projects, we learned how to light up an LED. With only 32 IO ports on ESP32, how do we light up a lot of leds? Sometimes it is possible to run out of pins on the ESP32, and you need to extend it with the shift register. You can use the 74HC595N chip to control 8 outputs at a time, taking up only a few pins on your microcontroller.

In addition, you can also connect multiple registers together to further expand the output. In this project, we will use ESP32, 74HC595 chip and LED to make a flowing water light to understand the function of the 74HC595 chip.

# 7.12.2 2.Components



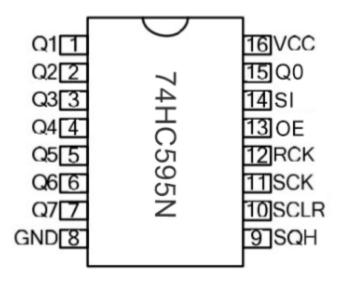
# 7.12.3 3.Component knowledge



#### 74HC595N Chip:

The 74HC595 chip is used to convert serial data into parallel data. A 74HC595 chip can convert the serial data of one byte into 8 bits, and send its corresponding level to each of the 8 ports correspondingly.

With this characteristic, the 74HC595 chip can be used to expand the IO ports of an ESP32. At least 3 ports are required to control the 8 ports of the 74HC595 chip.

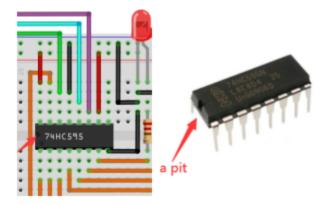


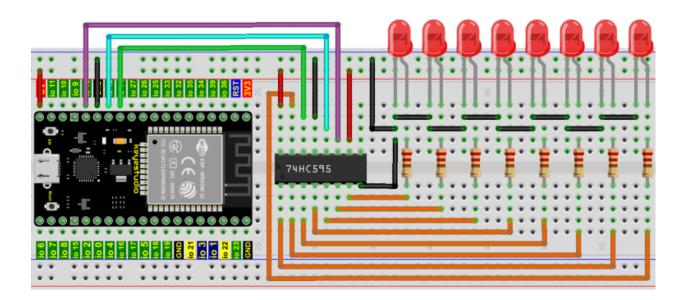
The ports of the 74HC595 chip are described as follows

PIN	FUNCTION
Pin 13–OE	Enable output, When this pin is in high level, Q0-Q7 is in high resistance state. When this pin is in low level, Q0-Q7 is in output mode.
Pin 14—SI	Serial data Input, only enter one bit at a time, so you can enter eight consecutive times to form one byte.
Pin	Remove shift register: When this pin is in low level, the content in shift register will be cleared.In
10—SCLR	this experiment, we connect VCC to maintain a high level.
Pin 11—SCK	Serial shift clock: when its electrical level is rising, serial data input register will do a shift.
Pin 12—RCK	Parallel Update Output: when its electrical level is rising, it will update the parallel data output. In
	this case, the data is output from ports Q0 to Q7 in parallel
Pin 9—SQH	Serial data output: it can be connected to more 74HC595 in series.
Q0–Q7(Pin 15Pin 1-7)	Parallel data output, can directly control the 8 segments of the digital tube.

# 7.12.4 4. Wiring diagram

Note: Note the orientation in which the 74HC595N chip is inserted.





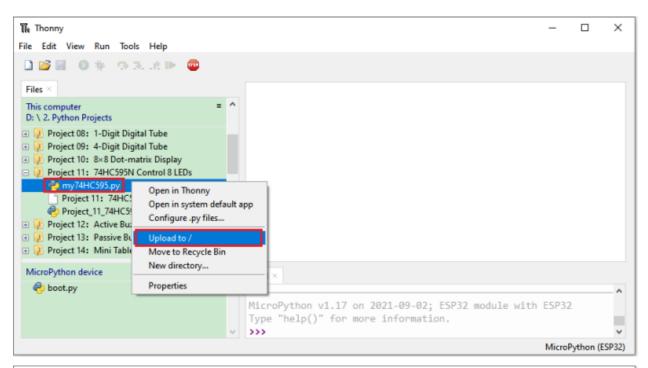
# 7.12.5 5. Project code

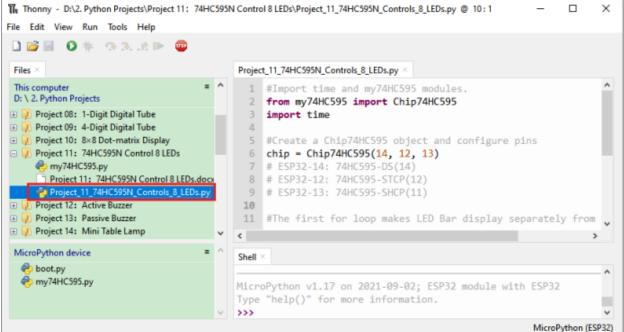
Codes used in this tutorial are saved in "**2**. **Python Projects**". If you haven't downloaded the code file, please click on the link to download it:Download Python Codes

2. Python Projects			—		$\times$
Share View				`	?
🔒 > This PC > Software (D:) > 2. Pyth	on Projects 🛛 🗸	ۍ	⊖ Search	2. Python	
Name	Date modified	Ту	pe	5	ize ^
Project 00: Boot	2/17/2022 10:21 AI	M Fil	e folder		
Project 01: Hello World	2/17/2022 10:21 Al	M Fil	e folder		
Project 02: Turn On LED	2/17/2022 11:10 AI	M Fil	e folder		
Project 03: LED Flashing	2/17/2022 11:12 AI	M Fil	e folder		
Project 04: Breathing LED	2/17/2022 10:21 AI	M Fil	e folder		

Open"Thonny"click"This computer"→"D:"→"2. Python Projects"→"Project 1174HC595N Control 8 LEDs".

Select"my74HC595.py", right click your mouse to select"Upload to /"wait for"my74HC595.py"to be uploaded to ESP32, and then double left-click "Project\_11\_74HC595N\_Controls\_8\_LEDs.py".





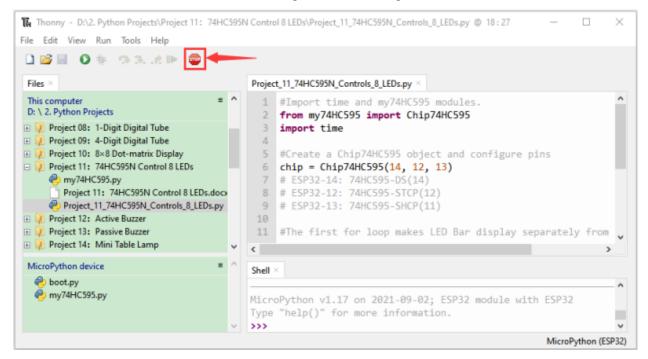
```
#Import time and my74HC595 modules.
from my74HC595 import Chip74HC595
import time
#Create a Chip74HC595 object and configure pins
chip = Chip74HC595(14, 12, 13)
## ESP32-14: 74HC595-DS(14)
## ESP32-12: 74HC595-STCP(12)
```

## ESP32-13: 74HC595-SHCP(11)

```
#The first for loop makes LED Bar display separately from left to right
#while the second for loop make it display separately from right to left.
while True:
    x = 0x01
    for count in range(8):
        chip.shiftOut(1, x)
        x = x<<1;
        time.sleep_ms(300)
    x = 0x01
    for count in range(8):
        chip.shiftOut(0, x)
        x = x<<1
        time.sleep_ms(300)</pre>
```

### 7.12.6 6.Project result

Make sure the ESP32 has been connected to the computer, click <sup>222</sup> "Stop/Restart backend".



Click Click Run current script", the code starts to be executed and you'll see that the 8 LEDs start flashing in flowing water mode. Press"Ctrl+C"or click ""Stop/Restart backend"to exit the program.

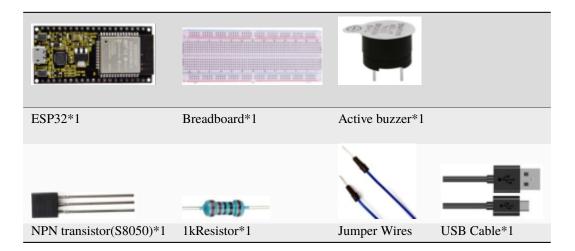
Files ×		Project_11_74HC595N_Controls_8_LEDs.py ×
This computer = D: \ 2. Python Projects D: \ 2. Python Projects D: \ 2. Python Project 08: 1-Digit Digital Tube D: Project 09: 4-Digit Digital Tube D: Project 10: 8×8 Dot-matrix Display D: Project 11: 74HC595N Control 8 LEDs Project 11: 74HC595N Control 8 LEDs.docs Project 11: 74HC595N Controls 8 LEDs.py D: Project 12: Active Buzzer D: Project 13: Passive Buzzer D: Project 14: Mini Table Lamp		<pre>1 #Import time and my74HC595 modules. 2 from my74HC595 import Chip74HC595 3 import time 4 5 #Create a Chip74HC595 object and configure pins 6 chip = Chip74HC595(14, 12, 13) 7 # ESP32-14: 74HC595-05(14) 8 # ESP32-12: 74HC595-STCP(12) 9 # ESP32-13: 74HC595-STCP(11) 10 11 #The first for loop makes LED Bar display separately from &lt;</pre>
MicroPython device =	^	Shell ×
🥹 boot.py 餋 my74HC595.py		MicroPython v1.17 on 2021-09-02; ESP32 module with ESP32 Type "help()" for more information. >>> %Run -c \$EDITOR_CONTENT

# 7.13 Project 12Active Buzzer

### 7.13.1 1.Introduction

Active buzzer is a sound component that is widely used as a sound component for computersprintersalarmselectronic toys and phonestimers etc. It has an internal vibration source, just by connecting to a 5V power supply, it can continuously buzz. In this project, we will use ESP32 to control the active buzzer to beep.

# 7.13.2 2.Components



### 7.13.3 3. Component knowledge



#### Active buzzer:

Active buzzer inside has a simple oscillator circuit, which can convert constant direct current into a certain frequency pulse signal. Once active buzzer receives a high level, it will produce sound. Passive buzzer is an internal without vibration source integrated electronic buzzer, it must be driven by 2k to 5k square wave, rather than a DC signal.

The two buzzers are very similar in appearance, but one buzzer with a green circuit board is a passive buzzer, while the other buzzer with black tape is an active buzzer. Passive buzzers don't have positive polarity, but active buzzers have. As shown below:



#### **Transistor:**



Because the buzzer requires such large current that GPIO of ESP32 output capability cannot meet the requirement, a transistor of NPN type is needed here to amplify the current.

Transistor, the full name: semiconductor transistor, is a semiconductor device that controls current. Transistorcan be used to amplify weak signal, or works as a switch. It has three electrodes(PINs): base (b), collector © and emitter (e).

When there is current passing between "be", "ce" will allow several-fold current (transistor magnification) pass, at this point, transistor works in the amplifying area. When current between "be" exceeds a certain value, "ce" will not allow

current to increase any longer, at this point, transistor works in the saturation area. Transistor has two types as shown below: PNP and NPN,



**PNP transistor** 

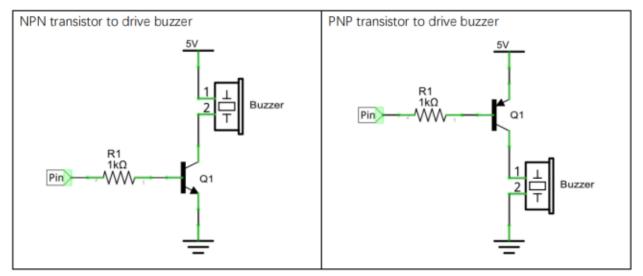


In our kit, the PNP transistor is marked with 8550, and the NPN transistor is marked with 8050.

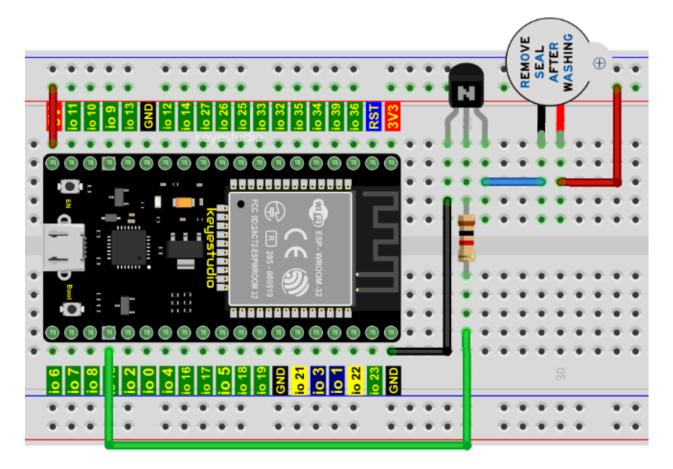
Based on the transistor's characteristics, it is often used as a switch in digital circuits. As micro-controller's capacity to output current is very weak, we will use transistor to amplify current and drive large-current components.

When using NPN transistor to drive buzzer, we often adopt the following method. If GPIO outputs high level, current will flow through R1, the transistor will get conducted, and the buzzer will sound. If GPIO outputs low level, no current flows through R1, the transistor will not be conducted, and buzzer will not sound.

When using PNP transistor to drive buzzer, we often adopt the following method. If GPIO outputs low level, current will flow through R1, the transistor will get conducted, and the buzzer will sound. If GPIO outputs high level, no current flows through R1, the transistor will not be conducted, and buzzer will not sound.



# 7.13.4 4.Wiring diagram



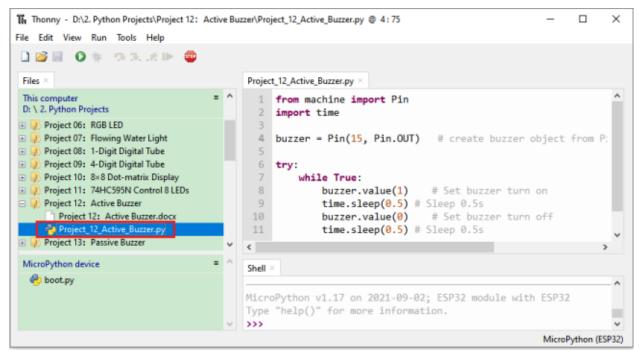
Note: The buzzer power supply in this circuit is 5V. On a 3.3V power supply, the buzzer can work, but will reduce the loudness.

# 7.13.5 5.Project code

Codes used in this tutorial are saved in "2. Python Projects". If you haven't downloaded the code file, please click on the link to download it:Download Python Codes

2. Python Projects			$\Box$ $\times$
Share View			~ 🕐
→ This PC → Software (D:) → 2. Pyth	on Projects 🛛 🗸 🗸	ට ,⊃ Search	n 2. Python
Name	Date modified	Туре	Size ^
Project 00: Boot	2/17/2022 10:21 AM	1 File folder	
Project 01: Hello World	2/17/2022 10:21 AN	1 File folder	
Project 02: Turn On LED	2/17/2022 11:10 AN	1 File folder	
Project 03: LED Flashing	2/17/2022 11:12 AN	1 File folder	
Project 04: Breathing LED	2/17/2022 10:21 AM	1 File folder	

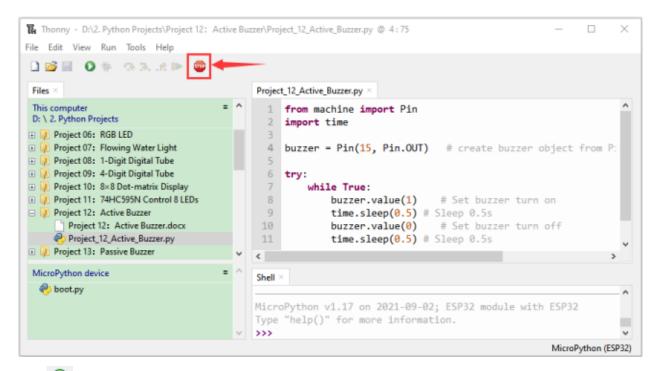
Open "Thonny" click "This computer"  $\rightarrow$  "D:"  $\rightarrow$  "2. Python Projects"  $\rightarrow$  "Project 12Active Buzzer", and then double left-click "Project\_12\_Active\_Buzzer.py".



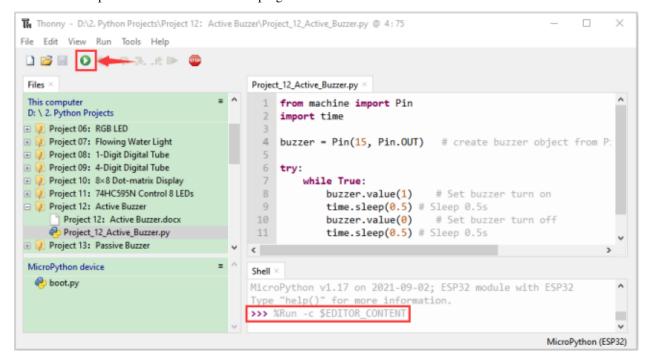
```
from machine import Pin
import time
buzzer = Pin(15, Pin.OUT)  # create buzzer object from Pin 15, Set Pin 15 to output
try:
    while True:
        buzzer.value(1)  # Set buzzer turn on
        time.sleep(0.5) # Sleep 0.5s
        buzzer.value(0)  # Set buzzer turn off
        time.sleep(0.5) # Sleep 0.5s
except:
    pass
```

# 7.13.6 6.Project result

Make sure the ESP32 has been connected to the computer, click 🖤 "Stop/Restart backend".



Click Click

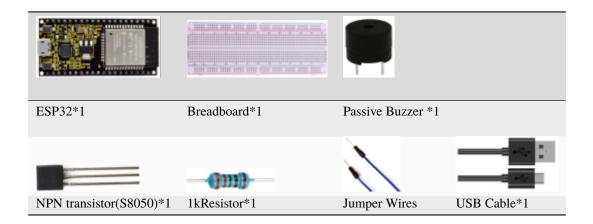


# 7.14 Project 13Passive Buzzer

# 7.14.1 1.Introduction:

In a previous project, we studied an active buzzer, which can only make a sound and may make you feel very monotonous. In this project, we will learn a passive buzzer and use the ESP32 control it to work. Unlike the active buzzer, the passive buzzer can emit sounds of different frequencies.

## 7.14.2 2.Components



# 7.14.3 3.Component knowledge



#### Passive buzzer:

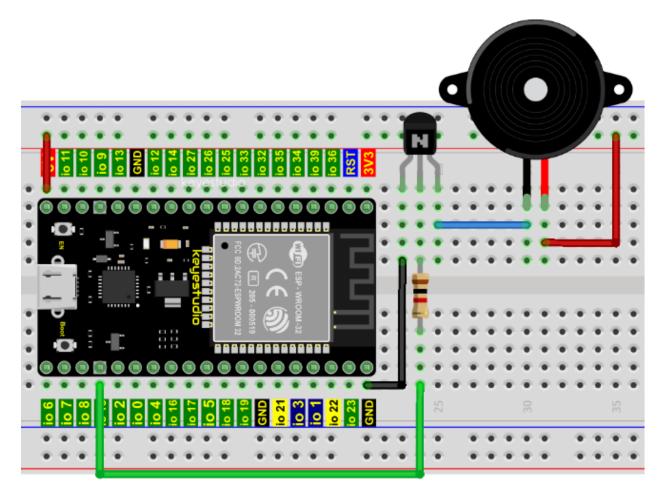
A passive buzzer is an integrated electronic buzzer with no internal vibration source and it has to be driven by 2K-5K square waves, not DC signals.

The two buzzers are very similar in appearance, but one buzzer with a green circuit board is a passive buzzer and the other buzzer with black tape is an active buzzer. Passive buzzers cannot distinguish between positive polarity while active buzzers can.



Transistor: Please refer to Project 12.

# 7.14.4 4.Wiring diagram:

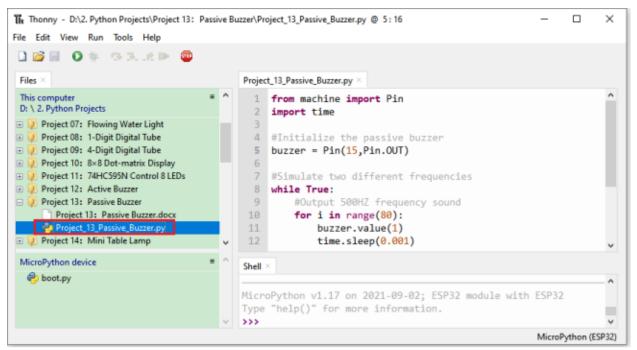


# 7.14.5 5.Project code

Codes used in this tutorial are saved in "2. Python Projects". If you haven't downloaded the code file, please click on the link to download it:Download Python Codes

2. Python Projects		_	
Share View			~ ?
→ This PC → Software (D:) → 2. Pytho	n Projects 🗸 진		. Python
Name	Date modified	Туре	Size ^
Project 00: Boot	2/17/2022 10:21 AM	File folder	
Project 01: Hello World	2/17/2022 10:21 AM	File folder	
Project 02: Turn On LED	2/17/2022 11:10 AM	File folder	
Project 03: LED Flashing	2/17/2022 11:12 AM	File folder	
Project 04: Breathing LED	2/17/2022 10:21 AM	File folder	

Open "Thonny", click "This computer" $\rightarrow$ "D:" $\rightarrow$ "2. Python Projects" $\rightarrow$ "Project 13Passive Buzzer", and then double left-click "Project\_13\_Passive\_Buzzer.py".

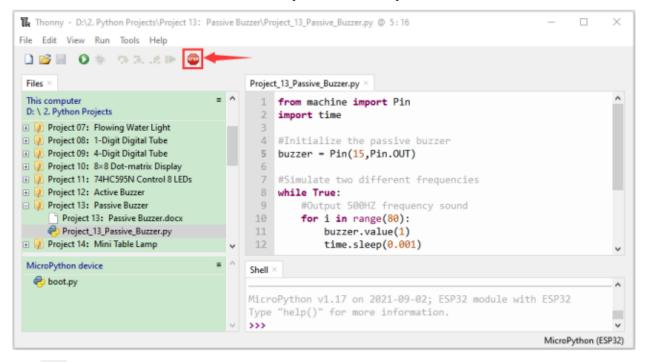




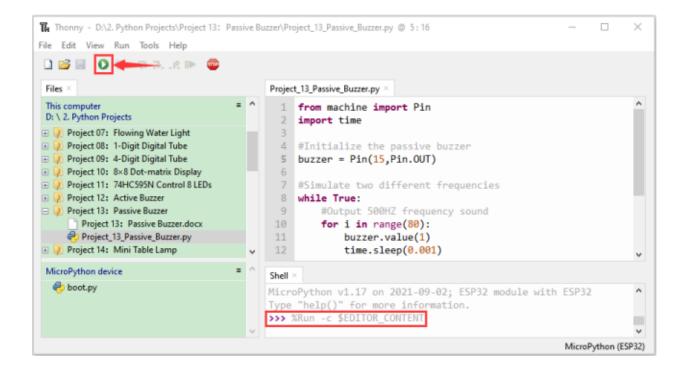
```
time.sleep(0.001)
#Output 250HZ frequency sound
for i in range(100):
    buzzer.value(1)
    time.sleep(0.002)
    buzzer.value(0)
    time.sleep(0.002)
```

# 7.14.6 6.Project result

Make sure the ESP32 has been connected to the computer, click <sup>99</sup> "Stop/Restart backend".



Click Click

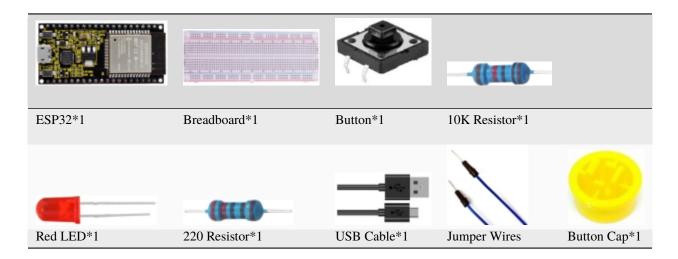


# 7.15 Project 14: Mini Table Lamp

### 7.15.1 1.Introduction

Do you know that the ESP32 can light up an LED when you press a button? In this project, we will use ESP32, a button switch and an LED to make a mini table lamp.

# 7.15.2 2.Components



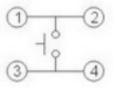
### 7.15.3 3. Component knowledge



#### **Button:**

A button can control the circuit on and off, the button is plugged into a circuit, the circuit is disconnected when the button is not pressed. The circuit works when you press the button, but breaks again when you release it.

Why does it only work when you press it? It starts from the internal structure of the button, which don't allow current to travel from one end of the button to the other before it is pressed; When pressed, a metal strip inside the button connects the two sides to allow electricity to pass through.

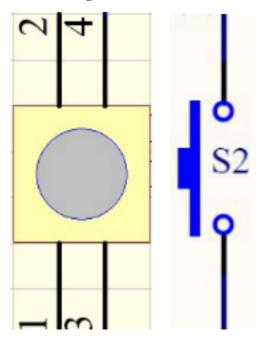


The internal structure of the button is shown in the figure

Before the button is pressed, 1 and 2 are on, 3 and 4 are also on, but 1, 3 or 1, 4 or 2, 3 or 2, 4 are off (not working). Only when the button is pressed, 1, 3 or 1, 4 or 2, 3 or 2, 4 are on.

The button switch is one of the most commonly used components in circuit design.

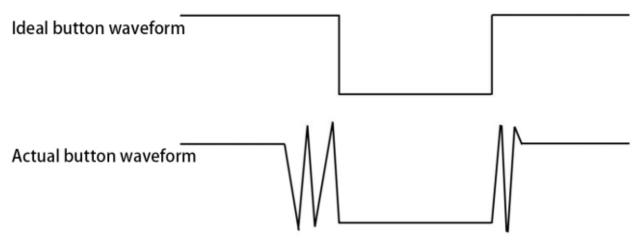
#### Schematic diagram of the button:



#### What is button shake?

We think of the switch circuit as "press the button and turn it on immediately", "press it again and turn it off immediately". In fact, this is not the case.

The button usually uses a mechanical elastic switch, and the mechanical elastic switch will produce a series of shake due to the elastic action at the moment when the mechanical contact is opened and closed (usually about 10ms). As a result, the button switch will not immediately and stably turn on the circuit when it is closed, and it will not be completely and instantaneously disconnected when it is turned off.



#### How to eliminate the shake?

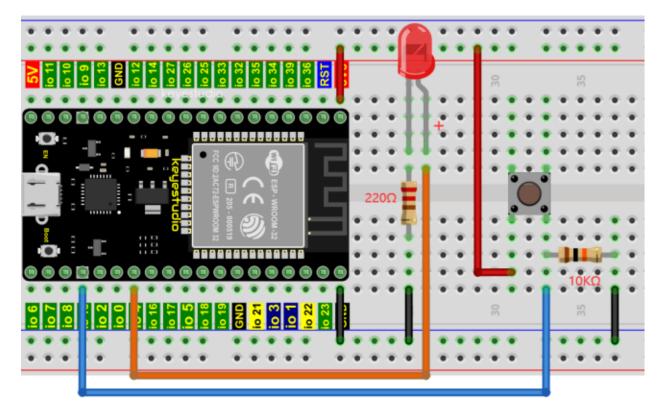
There are two common methods, namely fix shake in the software and hardware. We only discuss the shake removal in the software.

We already know that the shake time generated by elasticity is about 10ms, and the delay command can be used to delay the execution time of the command to achieve the effect of shake removal.

Therefore, we delay 0.02s in the code to achieve the key anti-shake function.

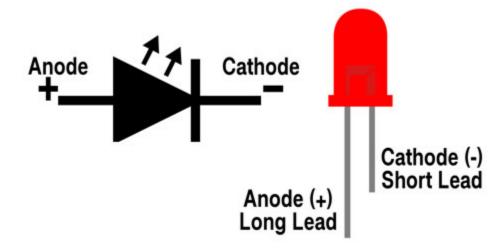
### Effect excluding jitter

### 7.15.4 4.Wiring Diagram

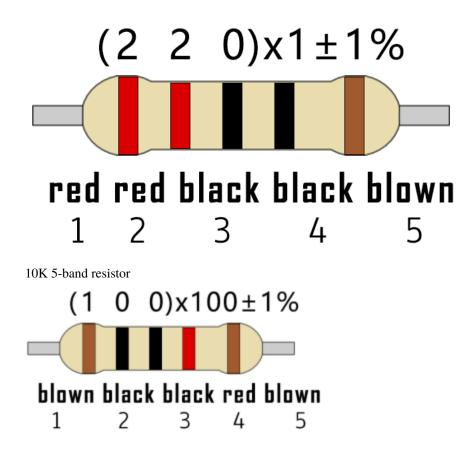


Note:

Connect the LED



220 5-band resistor

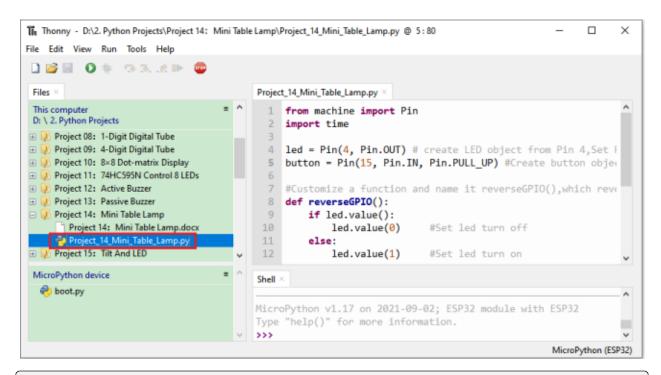


### 7.15.5 5.Project code

Codes used in this tutorial are saved in "2. Python Projects". If you haven't downloaded the code file, please click on the link to download it:Download Python Codes

2. Python Projects		—	$\Box$ $\times$
Share View			~ 🕐
📙 > This PC > Software (D:) > 2. Pytł	hon Projects 🗸 で		. Python
Name	Date modified	Туре	Size ^
Project 00: Boot	2/17/2022 10:21 AM	File folder	
Project 01: Hello World	2/17/2022 10:21 AM	File folder	
Project 02: Turn On LED	2/17/2022 11:10 AM	File folder	
Project 03: LED Flashing	2/17/2022 11:12 AM	File folder	
Project 04: Breathing LED	2/17/2022 10:21 AM	File folder	

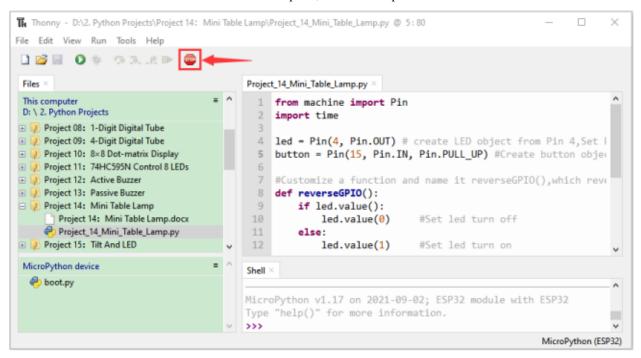
 $Open ``Thonny" click ``This computer" \rightarrow ``D:" \rightarrow ``2. Python Projects" \rightarrow ``Project 14Mini Table Lamp", and then double left-click ``Project_14_Mini_Table_Lamp.py".$ 



```
from machine import Pin
import time
led = Pin(4, Pin.OUT) # create LED object from Pin 4,Set Pin 4 to output
button = Pin(15, Pin.IN, Pin.PULL_UP) #Create button object from Pin15, Set GP15 to input
#Customize a function and name it reverseGPIO(), which reverses the output level of the
\rightarrow LED
def reverseGPIO():
    if led.value():
        led.value(0)
                          #Set led turn off
    else:
        led.value(1)
                         #Set led turn on
try:
    while True:
        if not button.value():
            time.sleep_ms(20)
            if not button.value():
                reverseGPIO()
                while not button.value():
                    time.sleep_ms(20)
except:
    pass
```

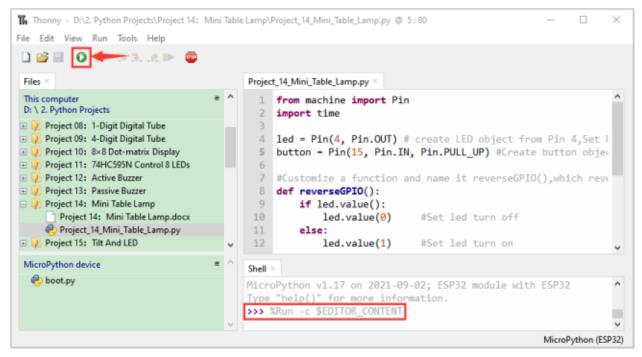
### 7.15.6 6.Project result

Make sure the ESP32 has been connected to the computer, click "" "Stop/Restart backend".



Click "Run current script", the code starts to be executed and you'll see that press the push button switch, the LED turns on; When it is released, the LED is still on.

Press it again, and the LED turns off. When it is released, the LED stays off. Doesn't it look like a mini table lamp? Press "Ctrl+C" or click "Stop/Restart backend" to exit the program.



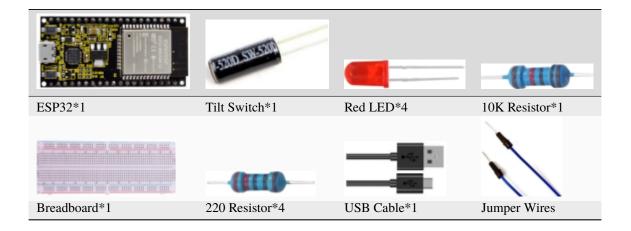
# 7.16 Project 15Tilt and LED

### 7.16.1 1.Introduction

The ancients without electronic clock, so the hourglass are invented to measure time. The hourglass has a large capacity on both sides, and which is filled with fine sand on one side. What's more, there is a small channel in the middle, which can make the hourglass stand upright, the side with fine sand is on the top.

Due to the effect of gravity, the fine sand will flow down through the channel to the other side of the hourglass. When the sand reaches the bottom, turn it upside down and record the number of times it has gone through the hourglass, therefore, the next day we can know the approximate time of the day by it. In this project, we will use ESP32 to control the tilt switch and LED lights to simulate an hourglass and make an electronic hourglass.

### 7.16.2 2.Components



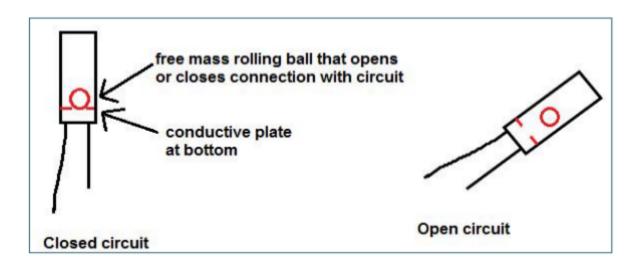
### 7.16.3 3.Component knowledge



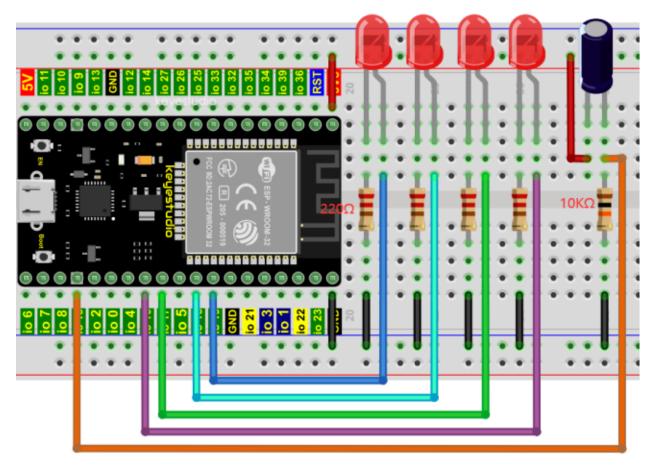
Tilt switch is also called digital switch. Inside is a metal ball that can roll. The principle of rolling the metal ball to contact with the conductive plate at the bottom, which is used to control the on and off of the circuit. When it is a rolling ball tilt sensing switch with single directional trigger, the tilt sensor is tilted toward the trigger end (two gold-plated pin ends), the tilt switch is in a closed circuit and the voltage at the analog port is about 5V(binary number is 1023),

In this way, the LED will light up. When the tilting switch is in horizontal position or tilting to the other end, the tilting switch is in open state the voltage of the analog port is about 0V (binary number is 0), the LED will turn off. In the program, we judge the state of the switch based on whether the voltage value of the analog port is greater than 2.5V (binary number is 512).

The internal structure of the tilt switch is used here to illustrate how it works, as shown below:

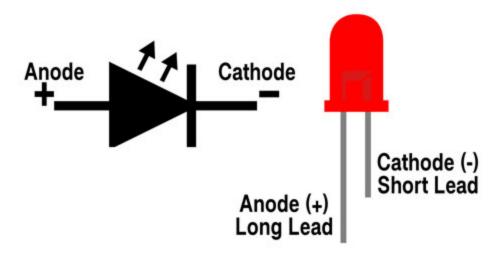


### 7.16.4 4. Wiring Diagram

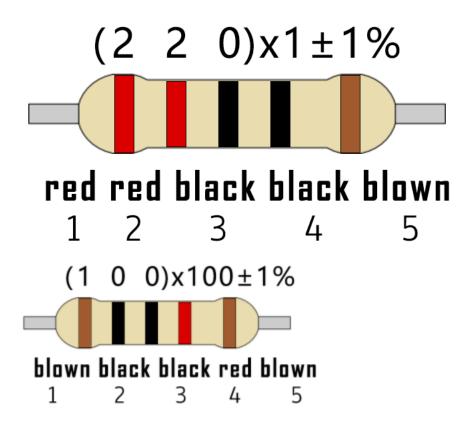


#### Note:

How to connect the LED



How to identify the 220 5-band resistor and 10K 5-band resistor

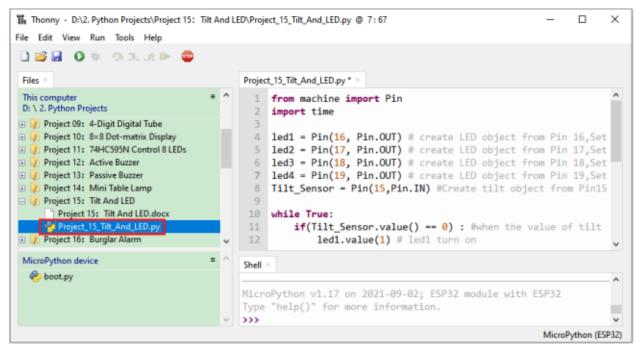


### 7.16.5 5.Project code

Codes used in this tutorial are saved in "2. Python Projects". If you haven't downloaded the code file, please click on the link to download it:Download Python Codes

2. Python Projects		— [	$\square$ ×
Share View			~ ?
→ This PC → Software (D:) → 2. Pyth	on Projects v さ		Python
Name	Date modified	Туре	Size ^
Project 00: Boot	2/17/2022 10:21 AM	File folder	
Project 01: Hello World	2/17/2022 10:21 AM	File folder	
Project 02: Turn On LED	2/17/2022 11:10 AM	File folder	
Project 03: LED Flashing	2/17/2022 11:12 AM	File folder	
Project 04: Breathing LED	2/17/2022 10:21 AM	File folder	

Open"Thonny" click"This computer"  $\rightarrow$  "D:"  $\rightarrow$  "2. Python Projects"  $\rightarrow$  "Project 15Tilt And LED", and then double left-click "Project\_15\_Tilt\_And\_LED.py".

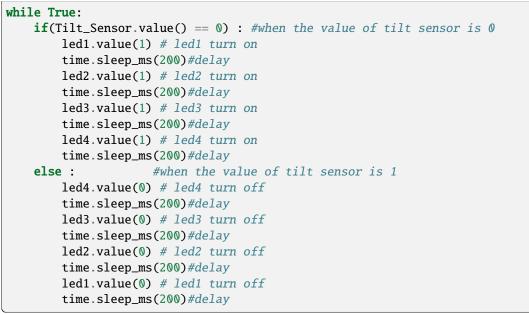


# from machine import Pin import time

```
led1 = Pin(16, Pin.OUT) # create LED object from Pin 2,Set Pin 2 to output
led2 = Pin(17, Pin.OUT) # create LED object from Pin 0,Set Pin 0 to output
led3 = Pin(18, Pin.OUT) # create LED object from Pin 4,Set Pin 4 to output
led4 = Pin(19, Pin.OUT) # create LED object from Pin 16,Set Pin 16 to output
Tilt_Sensor = Pin(15,Pin.IN) #Create tilt object from Pin15,Set GP15 to input
```

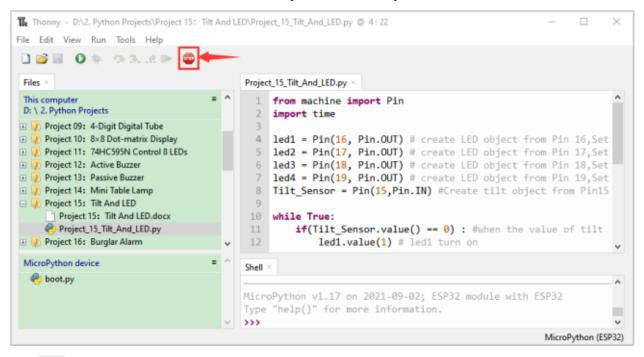
(continues on next page)

(continued from previous page)



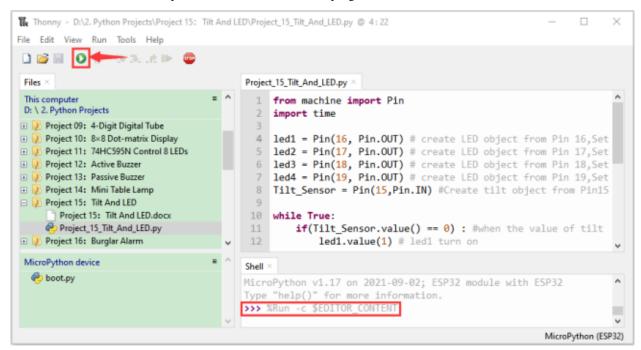
### 7.16.6 6.Project result

Make sure the ESP32 has been connected to the computer, click W "Stop/Restart backend".



Click "Run current script", the code starts to be executed and you'll see that when you tilt the breadboard to an angle, the LEDs will light up one by one.

When you turn the breadboard to the original angle, the LEDs will turn off one by one. Like the hourglass, the sand will leak out over time.



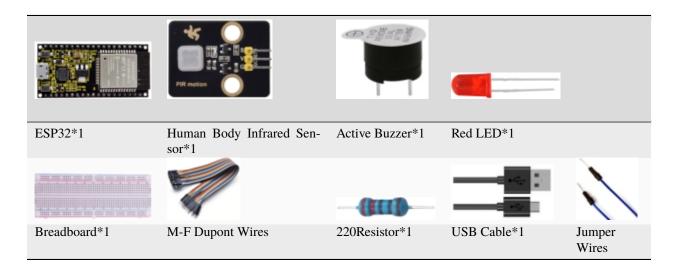
Press"Ctrl+C"or click <sup>222</sup> "Stop/Restart backend" to exit the program.

# 7.17 Project 16Burglar Alarm

### 7.17.1 1.Introduction

The human body infrared sensor measures the thermal infrared (IR) light emitted by moving objects. The sensor can detect the movement of peopleanimals and carsto trigger safety alarms and lighting. They are used to detect movement and ideal for security such as burglar alarms and security lighting systems. In this project, we will use the ESP32 control human body infrared sensorbuzzer and LED to simulate burglar alarm.

### 7.17.2 2.Components



### 7.17.3 3.Component knowledge



#### Human Body Infrared Sensor :

Its principle is that when some crystals, such as lithium tantalate and triglyceride sulfate are heated, the two ends of the crystal will generate an equal number of charges with opposite signs. These charges can be converted into voltage output by an amplifier. Due to the human body will release infrared light, although relatively weak, can still be detected. When the Human Body Infrared Sensor detects the movement of a nearby person, the sensor signal terminal outputs a high level 1, otherwise, it outputs low level 0.

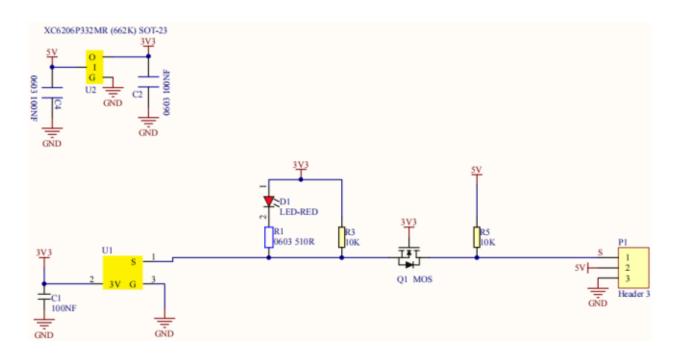
Special attention should be paid to the fact that this sensor can detect peopleanimals and cars in motion, which cannot be detected in static, and the maximum detection distance is about 7 meters.

**Note:** Since vulnerable to radio frequency radiation and temperature changes, the PIR motion sensor should be kept away from heat sources like radiators, heaters and air conditioners, as well as direct irradiation of sunlight, headlights and incandescent light.

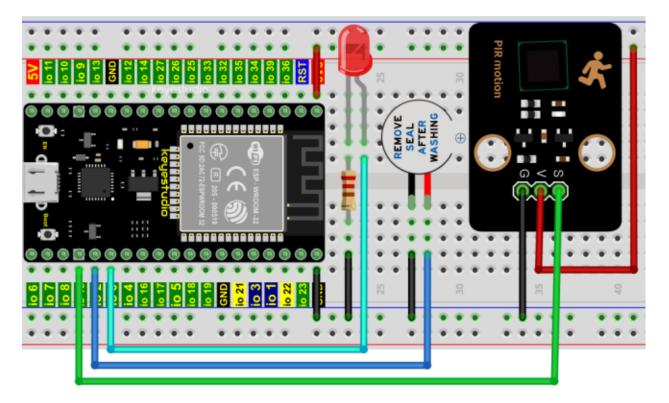
#### Features:

- Maximum input voltage: DC 3.3 ~ 5V.
- Maximum operating current: 50MA.
- Maximum power: 0.3W.
- Operating temperature: -20 ~ 85°C.
- Output high level is 3V, low level is 0V.
- Delay time: about 2.3 to 3 seconds.
- Detection Angle: about 100 degrees.
- Maximum detection distance: about 7 meters.
- Indicator light output (when the output is high, it will light up).
- Pin limiting current: 50MA.

#### Schematic diagram:



### 7.17.4 4. Wiring Diagram

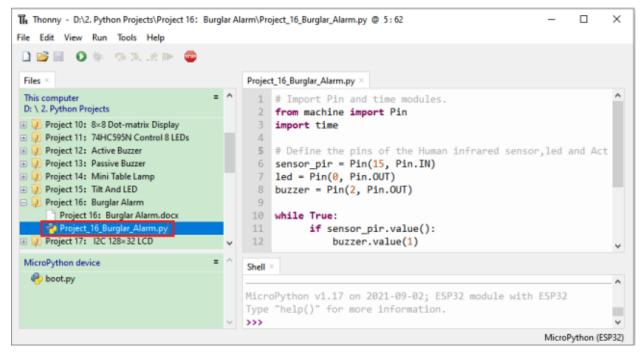


### 7.17.5 5.Project code

Codes used in this tutorial are saved in "2. Python Projects". If you haven't downloaded the code file, please click on the link to download it:Download Python Codes

2. Python Projects		_	$\Box$ $\times$
Share View			~ 🕐
→ This PC → Software (D:) → 2. Pyth	non Projects 🗸 🗸	ව 🔎 Search 2	2. Python
Name	Date modified	Туре	Size ^
	2/17/2022 10:21 AM	File folder	
Project 01: Hello World	2/17/2022 10:21 AM	File folder	
Project 02: Turn On LED	2/17/2022 11:10 AM	File folder	
Project 03: LED Flashing	2/17/2022 11:12 AM	File folder	
Project 04: Breathing LED	2/17/2022 10:21 AM	File folder	

Open "Thonny" click "This computer"  $\rightarrow$  "D:"  $\rightarrow$  "2. Python Projects"  $\rightarrow$  "Project 16Burglar Alarm", and then double left-click "Project\_16\_Burglar\_Alarm.py".



```
## Import Pin and time modules.
from machine import Pin
import time
## Define the pins of the Human infrared sensor,led and Active buzzer.
sensor_pir = Pin(15, Pin.IN)
led = Pin(0, Pin.OUT)
buzzer = Pin(2, Pin.OUT)
```

(continues on next page)

(continued from previous page)

ile True:	
<pre>if sensor_pir.value():</pre>	
<pre>buzzer.value(1)</pre>	
<pre>led.value(1)</pre>	
<pre>time.sleep(0.2)</pre>	
<pre>buzzer.value(0)</pre>	
led.value(0)	
<pre>time.sleep(0.2)</pre>	
else:	
<pre>buzzer.value(0)</pre>	
led.value(∅)	

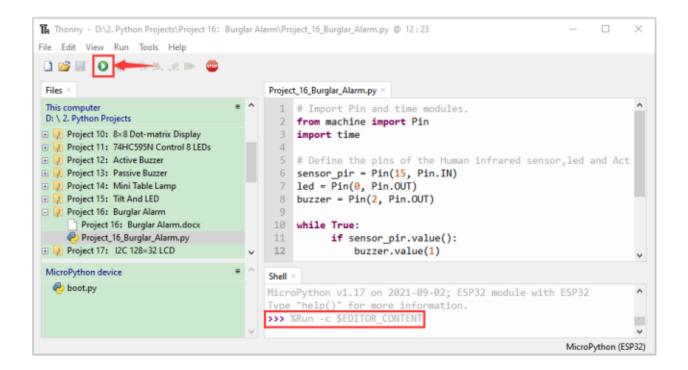
### 7.17.6 6.Project result

Make sure the ESP32 has been connected to the computer, click <sup>999</sup> "Stop/Restart backend".

🗋 📸 📓 🛛 🕸 🖘 3e 🕨 🚭 🖣 Files ×		Proje	rt_16_Burglar_Alarm.py ×	
This computer D: \ 2. Python Projects D: \ 2. Python Projects Project 10: 8×8 Dot-matrix Display Project 11: 74HC595N Control 8 LEDs Project 12: Active Buzzer Project 13: Passive Buzzer Project 13: Passive Buzzer Project 14: Mini Table Lamp Project 14: Mini Table Lamp Project 15: Tilt And LED Project 15: Tilt And LED Project 16: Burglar Alarm.docx Project 16: Burglar Alarm.py Project 17: 12C 128×32 LCD	= *	1 2 3 4 5 6 7 8 9 10 11 12	<pre># Import Pin and time modules. from machine import Pin import time # Define the pins of the Human infrared sensor,led and A sensor_pir = Pin(15, Pin.IN) led = Pin(0, Pin.OUT) buzzer = Pin(2, Pin.OUT) while True:     if sensor_pir.value():         buzzer.value(1)</pre>	Ict
MicroPython device	-		<pre>x oPython v1.17 on 2021-09-02; ESP32 module with ESP32 "help()" for more information.</pre>	

Click Circle Click

Press"Ctrl+C"or click "Stop/Restart backend" to exit the program.



# 7.18 Project 17 I2C 128×32 LCD

### 7.18.1 1.Introduction

In everyday life, we can do all kinds of experiments with the display module and also DIY a variety of small objects. For example, you can make a temperature meter with a temperature sensor and display, or make a distance meter with an ultrasonic module and display. In this project, we will use the LCD\_128X32\_DOT module as the display and connect it to the ESP32, which will be used to control the LCD\_128X32\_DOT display to display various English words, common symbols and numbers.

### 7.18.2 2.Components

ESP32*1	Breadboard*1	
SCL SCL SCL G		
LCD_128X32_DOT*1	M-F Dupont Wires	USB Cable*1

### 7.18.3 3.Component knowledge

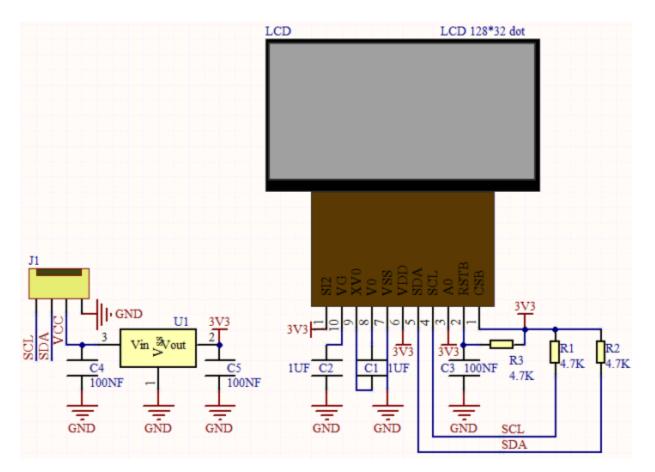


#### LCD\_128X32\_DOT:

It is an LCD module with 128\*32 pixels and its driver chip is ST7567A. The module uses the IIC communication mode, while the code contains a library of all alphabets and common symbols that can be called directly. When using, we can also set it in the code so that the English letters and symbols show different text sizes.

To make it easy to set up the pattern display, we also provide a mold capture software that converts a specific pattern into control code and then copies it directly into the test code for use.

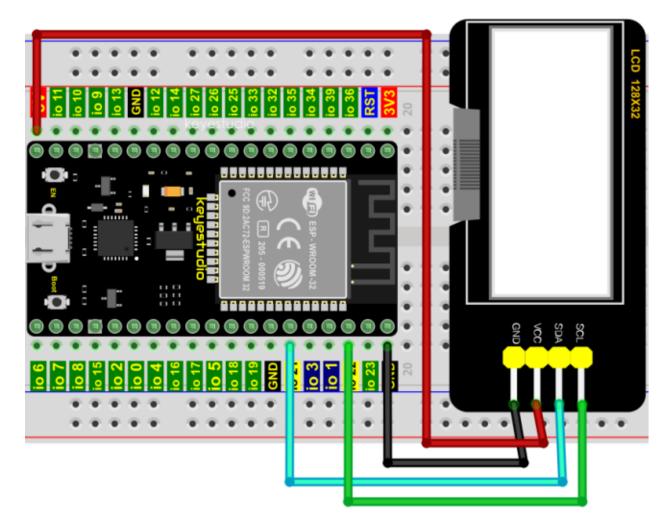
#### Schematic diagram of LCD\_128X32\_DOT



**Features:** 

- Pixel: 128\*32 character
- Operating voltage(chip)4.5V to 5.5V
- Operating current100mA (5.0V)
- Optimal operating voltage(module):5.0V

### 7.18.4 4.Wiring Diagram

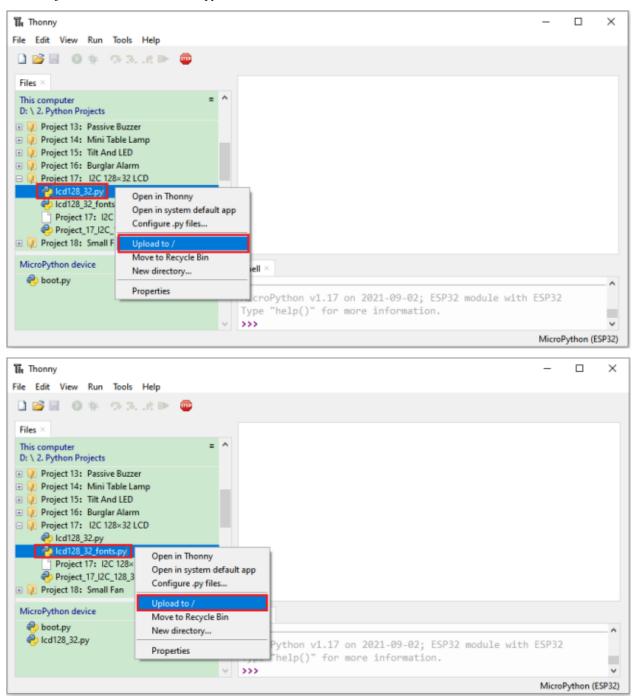


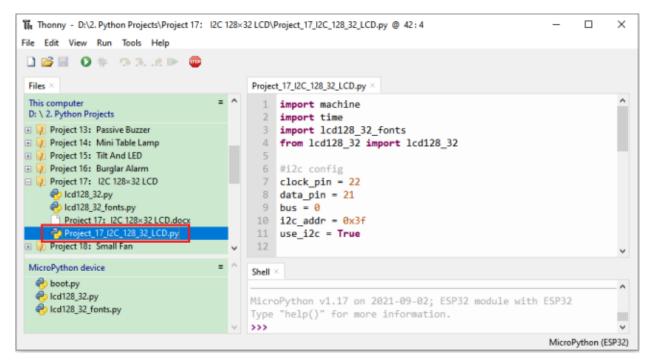
### 7.18.5 5.Project code

Codes used in this tutorial are saved in"**2. Python Projects**". If you haven't downloaded the code file, please click on the link to download it:Download Python Codes

> This PC > Software (D:) > 2. Pyth	non Projects 🗸 🗸	Ū		. Python
Name	Date modified		Туре	Size
Project 00: Boot	2/17/2022 10:21	MA	File folder	
Project 01: Hello World	2/17/2022 10:21	AM	File folder	
Project 02: Turn On LED	2/17/2022 11:10	AM	File folder	
Project 03: LED Flashing	2/17/2022 11:12	AM	File folder	
Project 04: Breathing LED	2/17/2022 10:21	AM	File folder	

Open"Thonny" click" This 128×32 computer"  $\rightarrow$  "D:"  $\rightarrow$  "2. Python Projects"→"Project 17I2C LCD". mouse Select"lcd128\_32.py"and "lcd128\_32\_fonts.py"click your select"Upload to to"wait uploaded ESP32and for"lcd128\_32.py"and"lcd128\_32\_fonts.py"to be to then click"Project\_17\_I2C\_128\_32\_LCD.py".





```
import machine
```

```
import time
import lcd128_32_fonts
from lcd128_32 import lcd128_32
#i2c config
clock_pin = 22
data_pin = 21
bus = 0
i2c_addr = 0x3f
use_i2c = True
def scan_for_devices():
   i2c = machine.I2C(bus,sda=machine.Pin(data_pin),scl=machine.Pin(clock_pin))
   devices = i2c.scan()
   if devices:
        for d in devices:
            print(hex(d))
   else:
        print('no i2c devices')
   if use_i2c:
        scan_for_devices()
        lcd = lcd128_32(data_pin, clock_pin, bus, i2c_addr)
   lcd.Clear()
   lcd.Cursor(0, 4)
lcd.Display("KEYESTUDIO")
lcd.Cursor(1, ◊)
lcd.Display("ABCDEFGHIJKLMNOPQR")
lcd.Cursor(2, ∅)
```

(continues on next page)

(continued from previous page)

```
lcd.Display("123456789+-*/<>=$@")
lcd.Cursor(3, 0)
lcd.Display("%^&(){}:;'|?,.~\\[]")
"""
while True:
    scan_for_devices()
    time.sleep(0.5)
"""
```

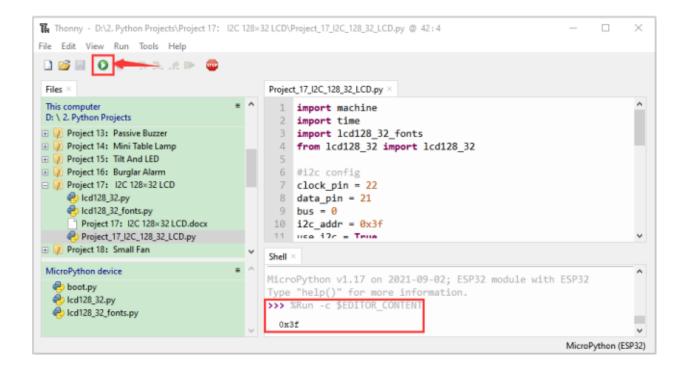
### 7.18.6 6.Project result

Make sure the ESP32 has been connected to the computer, click "" "Stop/Restart backend".



Click "Run current script", the code starts to be executed and you'll see that the 128X32 LCD module display will show "KEYESTUDIO" at the first line, "ABCDEFGHIJKLMNOPQR" will be displayed at the second line, "123456789 $\pm$ \*/<>=\$@" will be shown at the third line and "%^&(){}:;'P,-~[]" will be displayed at the fourth line.

Press "Ctrl+C" or click "" "Stop/Restart backend" to exit the program.

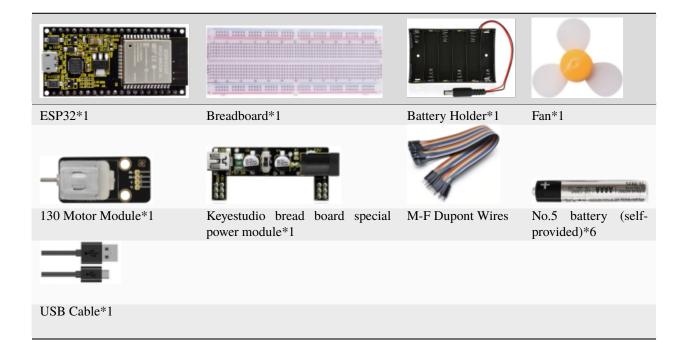


# 7.19 Project 18Small Fan

### 7.19.1 1.Introduction

In hot summer, we need electric fans to cool us down, so in this project, we will use ESP32 control 130 motor module and small fan blade to make a small electric fan.

### 7.19.2 2.Components



### 7.19.3 3.Component knowledge :

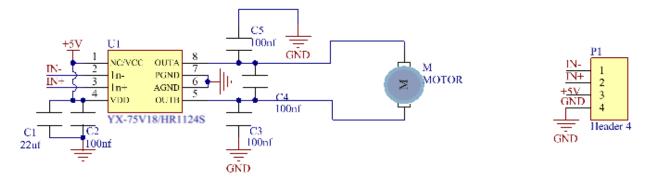


**130 motor module:** The motor control module uses the HR1124S motor control chip. which is a single-channel Hbridge driver chip for DC motor. The H-bridge driver part of the HR1124S uses low on-resistance PMOS and NMOS power tubes. The low on-resistance ensure low power loss of the chip and make the chip work safely for longer time In addition, the HR1124S has low standby current and low static operating current, which makes the HR1124S easy to use in toy solutions.

#### Features:

- Working voltage: 5V
- Working current: 200MA
- Working power: 2W
- Working temperature: -10°C~ +50°C

#### Schematic diagram of 130 motor module



#### Keyestudio Breadboard Power Supply Module



#### Introduction:

This breadboard power supply module is compatible with 5V and 3.3V, which can be applied to MB102 breadboard. The module contains two channels of independent control, powered by the USB all the way.

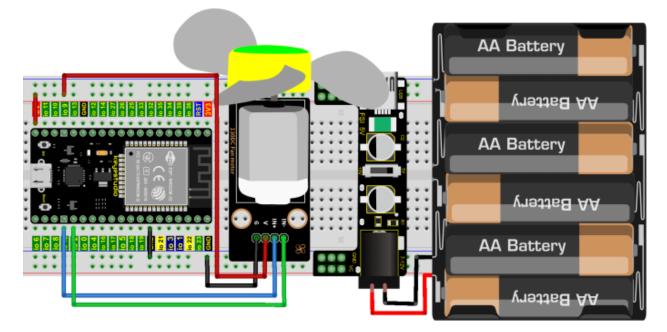
The output voltage is constant for the DC5V, and another way is powered by DC6.5-12V, output controlled by the slide switch, respectively for DC5V and DC3.3V.

If the other power supply is DC 6.5-12v, when the slide switch is switched to +5V, the output voltages of the left and right lines of the module are DC 5V. When the slide switch is switched to +3V, the output voltage of the USB power supply terminal of the module is DC5V, and the output voltage of the DC 6.5-12V power supply terminal of the other power supply is DC3.3V.

#### Specification:

- Applied to MB102 breadboard;
- Input voltageDC 6.5-12V or powered by USB;
- Output voltage3.3V or 5V
- Max output current<700ma
- Up and down two channels of independent control, one of which can be switched to 3.3V or 5V;
- Comes with two sets of DC output pins, easy for external use.

### 7.19.4 4. Wiring Diagram



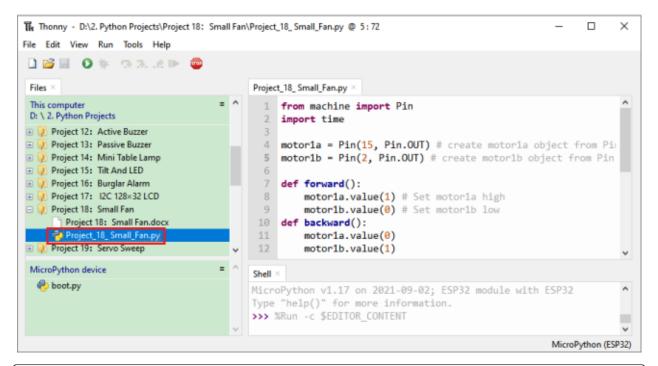
(Note: Connect the wires and then install a small fan blade on the DC motor. )

### 7.19.5 5.Project code

Codes used in this tutorial are saved in "2. Python Projects". If you haven't downloaded the code file, please click on the link to download it:Download Python Codes

Python Projects			_	
Share View				~ ?
> This PC > Software (D:) > 2. Py	thon Projects 🛛 🗸	۹ ن	Search 2	. Python
Name	Date modified	Туре		Size ^
Project 00: Boot	2/17/2022 10:21 A	M File f	older	
Project 01: Hello World	2/17/2022 10:21 A	M File f	older	
Project 02: Turn On LED	2/17/2022 11:10 A	M File f	older	
Project 03: LED Flashing	2/17/2022 11:12 A	M File f	older	
Project 04: Breathing LED	2/17/2022 10:21 A	M File f	older	

Open"Thonny"click"This computer" $\rightarrow$ "D:" $\rightarrow$ "2. Python Projects" $\rightarrow$ "Project 18Small Fan", and then double left-click"Project\_18\_Small\_Fan.py".



```
from machine import Pin
import time
motor1a = Pin(15, Pin.OUT) # create motor1a object from Pin 15, Set Pin 15 to output
motor1b = Pin(2, Pin.OUT) # create motor1b object from Pin 2, Set Pin 2 to output
def forward():
   motor1a.value(1) # Set motor1a high
    motor1b.value(0) # Set motor1b low
def backward():
   motor1a.value(0)
   motor1b.value(1)
def stop():
   motor1a.value(0)
   motor1b.value(0)
def test():
   forward() # motor forward
   time.sleep(5) #delay
   stop() # motor stop
   time.sleep(2)
   backward()# motor backward
   time.sleep(5)
   stop()
   time.sleep(2)
for i in range(5):
   test()
```

### 7.19.6 6.Project result

Make sure the ESP32 has been connected to the computer,  $click^{22}$  "Stop/Restart backend".

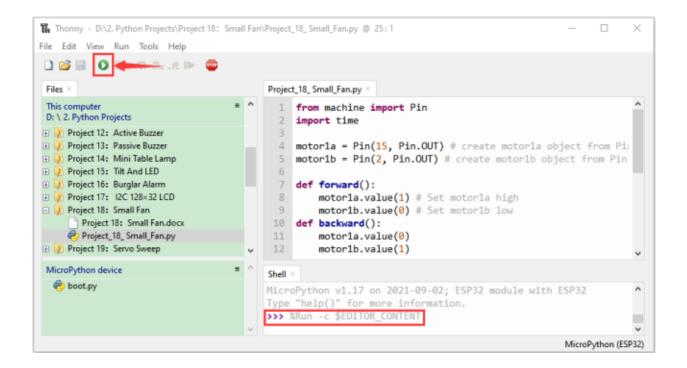


External power supply and power on.

Click Circle Click Circle Click Circle Content and Stops for 2 seconds, and then turns clockwise for 5 seconds and stops for 2 seconds, and then turns clockwise for 5 seconds and stops for 2 seconds.

Repeat this rule for 5 times and then the small fan stops.

Press"Ctrl+C"or click <sup>22</sup> "Stop/Restart backend" to exit the program.

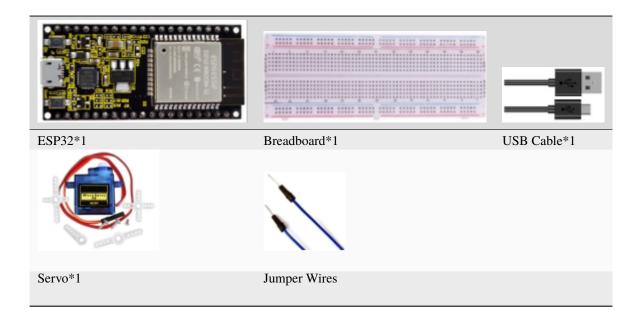


# 7.20 Project 19Servo Sweep

### 7.20.1 1.Introduction

Servo is an electric motor that can rotate very precisely. At present, it has been widely used in toy carsremote control helicoptersairplanesrobots, etc. In this project, we will use ESP32 to control the rotation of the servo.

### 7.20.2 2.Components



### 7.20.3 3.Component knowledge

Servo



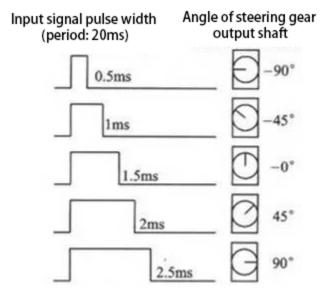
The servo is a kind of position servo driver, which is mainly composed of housing, circuit board, copless motor, gear and position detector.

Its working principle is that the receiver or microcontroller sends a signal to the servo which has an internal reference circuit that generates a reference signal with a period of 20ms and a width of 1.5ms, and compares the DC bias voltage with the voltage of the potentiometer to output voltage difference. The IC on the circuit board determines the direction of rotation, and then drives the coreless motor to start rotation and transmits the power to the swing arm through the reduction gear, while the position detector sends back a signal to determine whether it has reached the positioning.

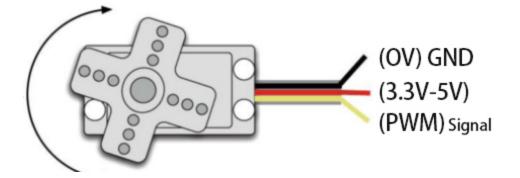
It is suitable for those control systems that require constant change of angle and can be maintained.

When the motor rotates at a certain speed, the potentiometer is driven by the cascade reduction gear to rotate so that the voltage difference is 0 and the motor stops rotating. The angle range of general servo rotation is 0 to 180 degrees.

The pulse period for controlling the servo is 20ms, the pulse width is 0.5ms to 2.5ms, and the corresponding position is -90 degrees to +90 degrees. The following is an example of a 180 degree servo

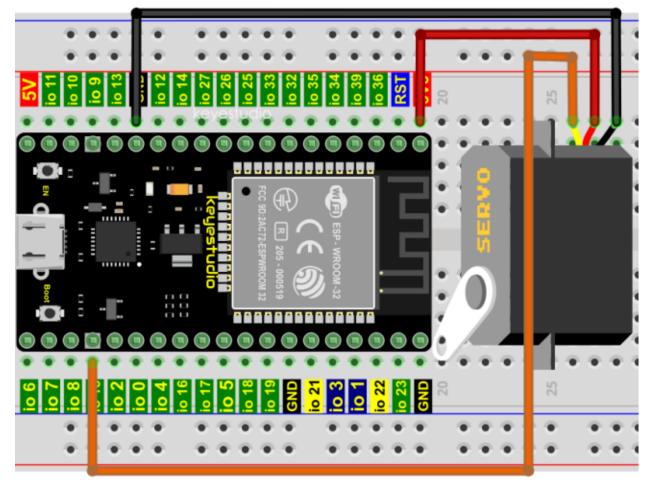


Servo motors have many specifications, but they all have three connecting wires, which are brown, red, and orange (different brands may have different colors). The brown is GND, the red is the positive power supply, and the orange is the signal line.



### 7.20.4 4.Wiring Diagram

When supplying the servo, please note that the power supply voltage should be 3.3V-5V. Make sure there are no errors when connecting the servo to the power supply.



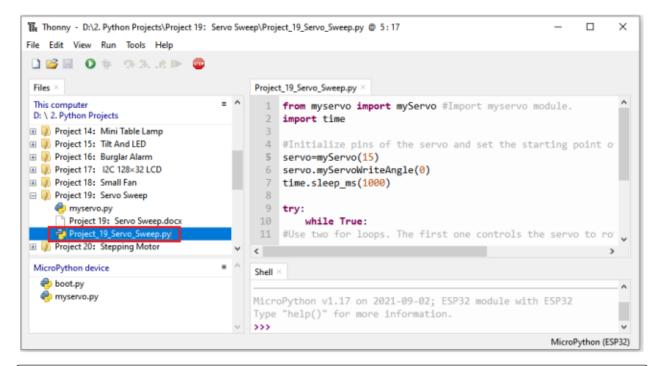
### 7.20.5 5.Project code

Codes used in this tutorial are saved in "2. Python Projects". If you haven't downloaded the code file, please click on the link to download it:Download Python Codes

. Python Projects		_	$\Box$ $\times$
Share View			~ ?
> This PC > Software (D:) > 2. Pyth	non Projects 🗸 🗸 🗸	ට 🔎 Search	2. Python
Name	Date modified	Туре	Size ^
Project 00: Boot	2/17/2022 10:21 AM	File folder	
Project 01: Hello World	2/17/2022 10:21 AM	File folder	
Project 02: Turn On LED	2/17/2022 11:10 AM	File folder	
Project 03: LED Flashing	2/17/2022 11:12 AM	File folder	
Project 04: Breathing LED	2/17/2022 10:21 AM	File folder	

Open"Thonny" click"This computer"  $\rightarrow$  "D:"  $\rightarrow$  "2. Python Projects"  $\rightarrow$  "Project 19Servo Sweep". Select"myservo.py" right-click your mouse to select"Upload to /" wait for "myservo.py" to be uploaded to ESP32 and then click"Project\_19\_Servo\_Sweep.py".

The Thonny		-		×
File Edit View Run Tools Help				
D 22 ■ 0 ★ 0 3 ● 2 Files ×				
This computer D: \ 2. Python Projects	= ^			
<ul> <li></li></ul>		_		
Project 19: Ser Project 19: Ser Project_19_Serv Project_20: Steppir Configure .py files				
MicroPython device         Upload to /           Doot.py         Move to Recycle Bin           New directory         New directory		nell ×		^
Properties	~ >	<pre>icroPython v1.17 on 2021-09-02; ESP32 module with ESP32 ype "help()" for more information. &gt;&gt;</pre>		*
		Micro	ython (E	SP32)



```
from myservo import myServo #Import myservo module.
import time
```

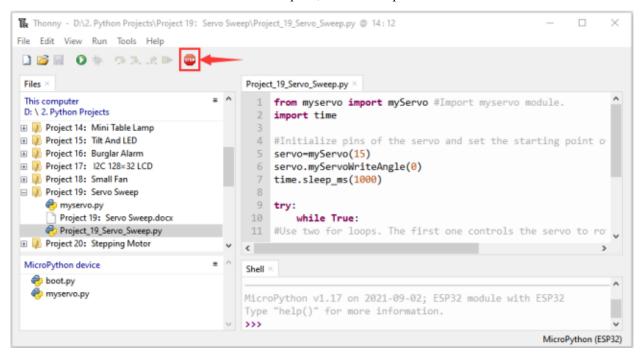
```
#Initialize pins of the servo and set the starting point of the servo to 0 degree.
servo=myServo(15)
servo.myServoWriteAngle(0)
time.sleep_ms(1000)
```

#### try:

```
while True:
#Use two for loops. The first one controls the servo to rotate from 0 degree to 180_
...degrees
#while the other controls it to rotate back from 180 degrees to 0 degree.
for i in range(0,180,1):
    servo.myServoWriteAngle(i) #Control the servo to rotate to a specified angle_
...within the range of 0-180 degrees.
    time.sleep_ms(15)
    for i in range(180,0,-1):
        servo.myServoWriteAngle(i)
        time.sleep_ms(15)
except:
    servo.deinit()
```

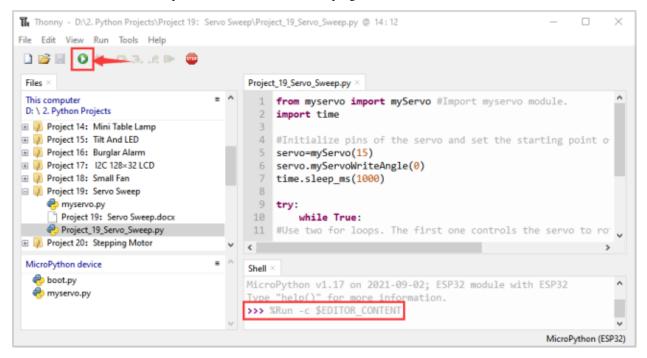
### 7.20.6 6.Project result

Make sure the ESP32 has been connected to the computer, click "" "Stop/Restart backend".



Click "Run current script", the code starts to be executed and you'll see that the Servo will rotate from 0 degrees to 180 degrees and then reverse the direction to make it rotate from 180 degrees to 0 degrees and repeat these actions in an endless loop.

Press"Ctrl+C"or click W "Stop/Restart backend" to exit the program.



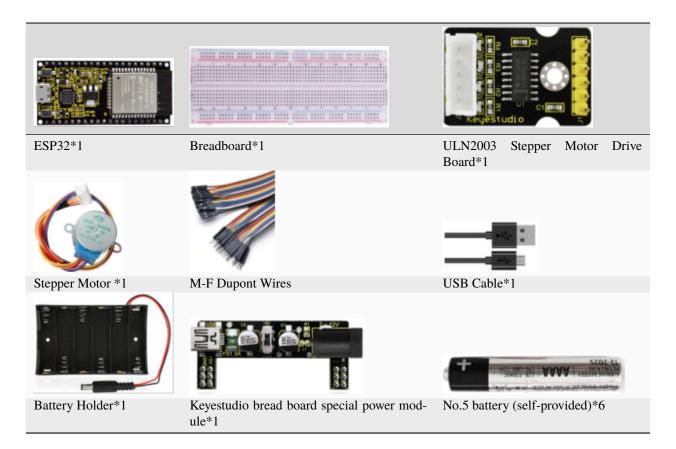


# 7.21 Project 20Stepping Motor

### 7.21.1 1.Introduction

Stepper motor is the most important part of industrial robot 3D printer lathes and other mechanical equipment with accurate positioning. In this project, we will use ESP32 control ULN2003 stepper motor drive board to drive the stepper motor to rotate.

### 7.21.2 2.Components



### 7.21.3 3.Component knowledge :



**Stepper motor:** 

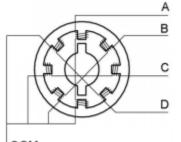
It is a motor controlled by a series of electromagnetic coils. It can rotate by the exact number of degrees (or steps) needed, allowing you to move it to a precise position and keep it there. It does this by supplying power to the coil inside the motor in a very short time, but you must always supply power to the motor to keep it in the position you want.

There are two basic types of stepping motors, namely unipolar stepping motor and bipolar stepping motor. In this project, we use a 28-BYJ48 unipolar stepper motor.



#### Working Principle:

The stepper motor is mainly composed of a stator and a rotor. The stator is fixed. As shown in the figure below, the part of the coil group A, B, C, and D will generate a magnetic field when the coil group is energized. The rotor is the rotating part. As follows, the middle part of the stator, two poles are permanent magnets.



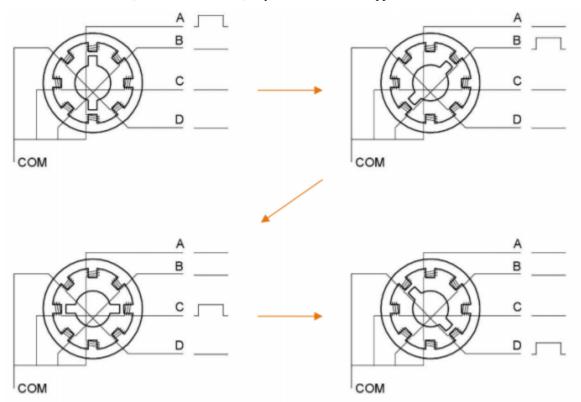
COM

Single -phase four beat:

At the beginning, the coils of group A are turned on, and the poles of the rotor point at A coil. Next, the group A coil are disconnected, and the group B coils are turned on. The rotor will turn clockwise to the group B. Then, group B is disconnected, group C is turned on, and the rotor is turned to group C. After that, group C is disconnected, and group D is turned on, and the rotor is turned to group D is disconnected, group A coils. Therefore, rotor turns 180° and continuously rotates B-C-D-A, which means it runs a circle (eight phase).

As shown below, he rotation principle of stepper motor is A - B C - D - A.

You make order inverse(D - C - B - A - D ...) if you want to make stepper motor rotate anticlockwise.



Half-phase and eight beat:

8 beat adopts single and dual beat wayA - AB B - BC - C - CD - D - DA - A ... rotor will rotate half phase in this order. For example, when A coil is electrifiedrotor faces to A coil , then A and B coil are connected, on this condition, the strongest magnetic field produced lies in the central part of AB coil, which means rotating half-phase clockwise.

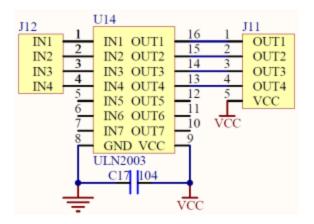
#### **Stepper Motor Parameters:**

The rotor rotates one circle when the stepper motor we provide rotates 32 phases and with the output shaft driven by 1:64 reduction geared set. Therefore the rotation (a circle) of output shaft requires 32 \* 64 = 2048 phases.

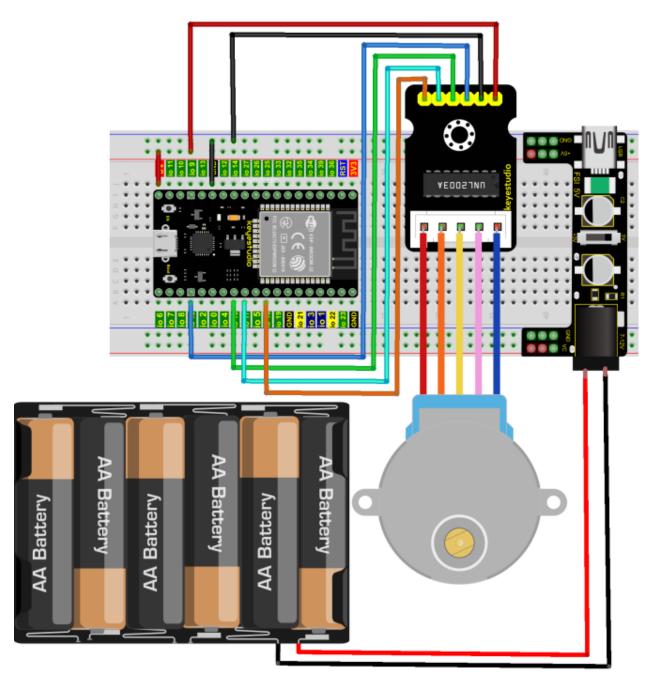
The step angle of 4-beat mode of 5V and 4-phase stepper motor is 11.25. And the step angle of 8-beat mode is 5.625, the reduction ratio is 1:64.

**ULN2003Stepper Motor Drive Board:** It is a stepper motor driver, which converts the weak signal into a stronger control signal to drive the stepper motor.

The following schematic diagram shows how to use the ULN2003 stepper motor driver board interface to connect a unipolar stepper motor to the pins of the ESP32, and shows how to use four TIP120 interfaces.



# 7.21.4 4.Wiring Diagram

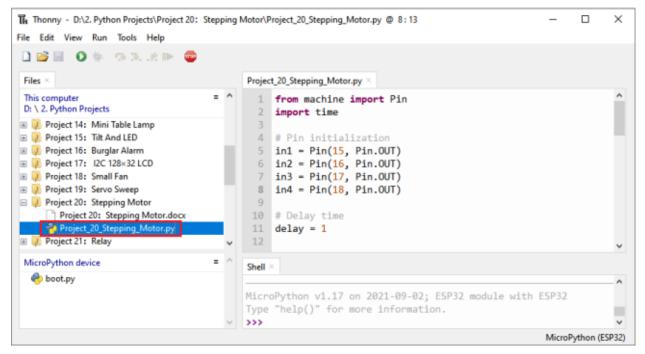


# 7.21.5 5.Project code

Codes used in this tutorial are saved in "2. Python Projects". If you haven't downloaded the code file, please click on the link to download it:Download Python Codes

2. Python Projects		_	$\Box$ $\times$
Share View			~ ?
→ This PC → Software (D:) → 2. Pyth	on Projects 🗸 진		. Python
Name	Date modified	Туре	Size ^
Project 00: Boot	2/17/2022 10:21 AM	File folder	
Project 01: Hello World	2/17/2022 10:21 AM	File folder	
Project 02: Turn On LED	2/17/2022 11:10 AM	File folder	
Project 03: LED Flashing	2/17/2022 11:12 AM	File folder	
Project 04: Breathing LED	2/17/2022 10:21 AM	File folder	

Open"Thonny" click"This computer"  $\rightarrow$  "D:"  $\rightarrow$  "2. Python Projects"  $\rightarrow$  "Project 20Stepping Motor", and then double left-click "Project\_20\_Stepping\_Motor.py".



```
from machine import Pin
import time
## Pin initialization
in1 = Pin(15, Pin.OUT)
in2 = Pin(16, Pin.OUT)
in3 = Pin(17, Pin.OUT)
in4 = Pin(18, Pin.OUT)
```

(continues on next page)

(continued from previous page)

```
## Delay time
delay = 1
## The number of steps required for the motor to rotate one revolution, (about 360^\circ),
\rightarrow with a slight deviation
ROUND_VALUE = 509
## The sequence value of the four-phase eight-beat stepper motor: A-AB-B-BC-C-CD-D-DA-A
STEP_VALUE = [
    [1, 0, 0, 0],
    [1, 1, 0, 0],
    [0, 1, 0, 0],
    [0, 1, 1, 0],
    [0, 0, 1, 0],
    [0, 0, 1, 1],
    [0, 0, 0, 1],
    [1, 0, 0, 1],
]
## Pin output low level
def reset():
    in1(0)
    in2(≬)
    in3(≬)
    in4(≬)
## If count is positive integers turn clockwise, if count is negative integers turn.
→counterclockwise
def step_run(count):
    direction = 1
                   # turn clockwise
    if count < 0:
        direction = -1 # turn counterclockwise
        count = -count
    for x in range(count):
        for bit in STEP_VALUE[::direction]:
            in1(bit[0])
            in2(bit[1])
            in3(bit[2])
            in4(bit[3])
            time.sleep_ms(delay)
    reset()
## If a is positive integers turn clockwise, if a is negative integers turn.
→counterclockwise
def step_angle(a):
    step_run(int(ROUND_VALUE * a / 360))
## Cycle: turn clockwise one circle, then counterclockwise one circle.
while True:
    step_run(509)
    step_run(-509)
    step_angle(360)
```

(continues on next page)

(continued from previous page)

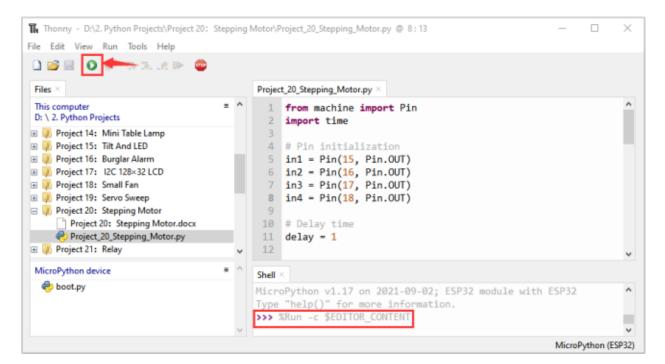
step\_angle(-360)

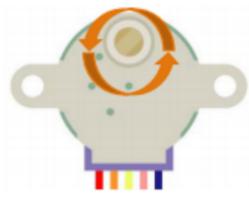
### 7.21.6 6.Project result

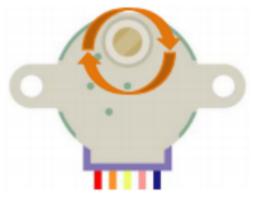
Make sure the ESP32 has been connected to the computer, click "" "Stop/Restart backend". 🖡 Thonny - D:\2. Python Projects\Project 20: Stepping Motor\Project\_20\_Stepping\_Motor.py @ 8:13  $\times$ File Edit View Run Tools Help 🗋 💕 🔙 🖸 🎋 🔿 3. .e 🕨 Files Project\_20\_Stepping\_Motor.py This computer = ^ from machine import Pin 1 D: \ 2. Python Projects import time 🗉 🕕 Project 14: Mini Table Lamp 🗉 🕖 Project 15: Tilt And LED # Pin initialization 4 🗉 🔑 Project 16: Burglar Alarm 5 in1 = Pin(15, Pin.OUT) 표 🕕 Project 17: I2C 128×32 LCD 6 in2 = Pin(16, Pin.OUT) 🗉 🕖 Project 18: Small Fan in3 = Pin(17, Pin.OUT) 7 🗉 🕕 Project 19: Servo Sweep 8 in4 = Pin(18, Pin.OUT) 😑 🔑 Project 20: Stepping Motor Project 20: Stepping Motor.docx 10 # Delay time Project\_20\_Stepping\_Motor.py delay = 1 🗉 🕖 Project 21: Relay MicroPython device = Shell \varTheta boot.py MicroPython v1.17 on 2021-09-02; ESP32 module with ESP32 Type "help()" for more information. >>> MicroPython (ESP32)

External power supply and power on. Click "Run current script", the code starts to be executed and you'll see that the four LEDs (D1,D2,D3,D4) on the ULN2003 drive module will light up. The stepper motor rotates clockwise first,

then counterclockwise, and repeat these actions in an endless loop. Press"Ctrl+C"or click "" "Stop/Restart backend" to exit the program.







# 7.22 Project 21Relay

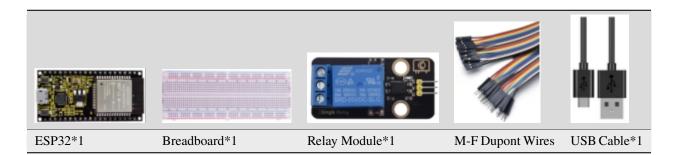
## 7.22.1 1.Introduction

In our daily life, we usually use communication to drive electrical equipments, and sometimes we use switches to control electrical equipments. If the switch is connected directly to the ac circuit, leakage occurs and people are in danger.

Therefore, from the perspective of safety, we specially designed this relay module with NO(normally open) end and NC(normally closed) end.

In this project, we will learn a relatively special and easy-to-use switch, which is the relay module.

# 7.22.2 2.Components



# 7.22.3 3.Component knowledge

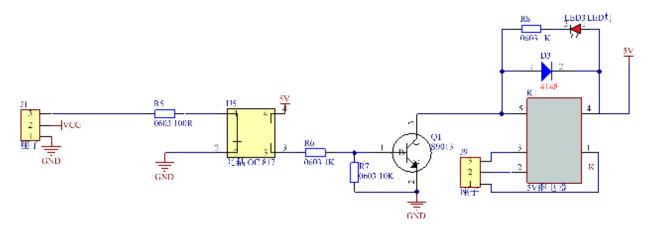
#### **Relay:**

It is an "automatic switch" that uses a small current to control the operation of a large current.

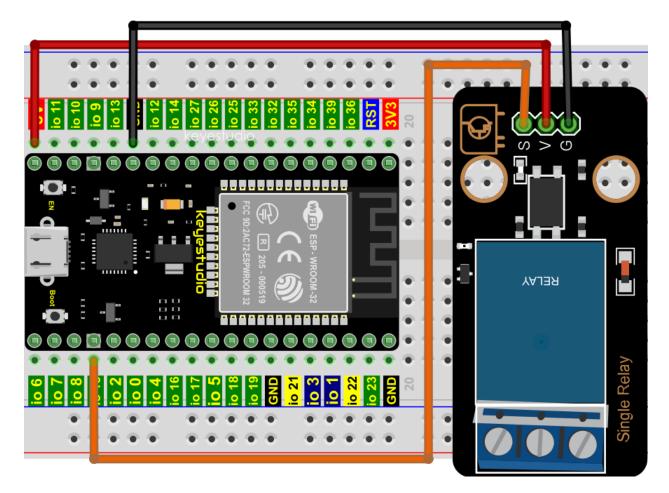
- Input voltage3.3V-5V
- Rated load5A 250VAC (NO/NC) 5A 24VDC (NO/NC)

The rated load means that devices with dc voltage of 24V or AC voltage of 250V can be controlled using 3.3V-5V microcontrollers.

#### Schematic diagram of Relay



# 7.22.4 4. Wiring Diagram

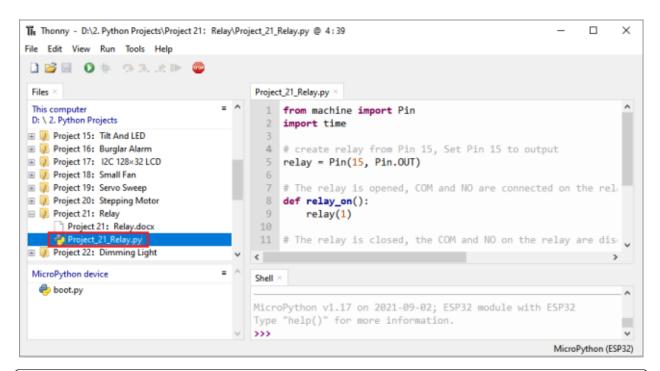


# 7.22.5 5.Project code

Codes used in this tutorial are saved in "2. Python Projects". If you haven't downloaded the code file, please click on the link to download it:Download Python Codes

2. Python Projects		_	$\Box$ $\times$
Share View			~ ?
→ This PC → Software (D:) → 2. Pyt	hon Projects 🗸 ぐ	🔎 Search 2	. Python
Name	Date modified	Туре	Size ^
Project 00: Boot	2/17/2022 10:21 AM	File folder	
Project 01: Hello World	2/17/2022 10:21 AM	File folder	
Project 02: Turn On LED	2/17/2022 11:10 AM	File folder	
Project 03: LED Flashing	2/17/2022 11:12 AM	File folder	
	2/17/2022 10:21 AM	File folder	

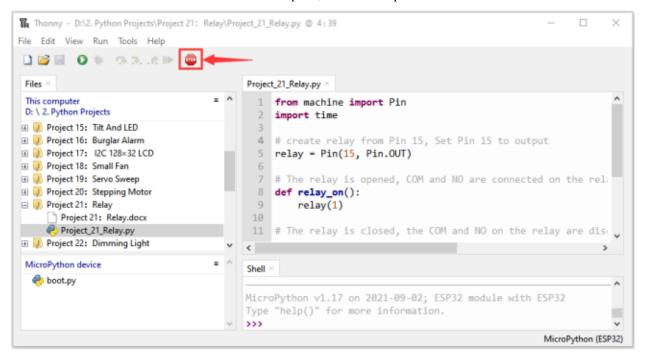
Open "Thonny", click "This computer"  $\rightarrow$  "D:"  $\rightarrow$  "2. Python Projects"  $\rightarrow$  "Project 21Relay", and then click "Project\_21\_Relay.py".



```
from machine import Pin
import time
## create relay from Pin 15, Set Pin 15 to output
relay = Pin(15, Pin.OUT)
## The relay is opened, COM and NO are connected on the relay, and COM and NC are.
\rightarrow disconnected.
def relay_on():
    relay(1)
## The relay is closed, the COM and NO on the relay are disconnected, and the COM and NC.
\rightarrow are connected.
def relay_off():
    relay(≬)
## Loop, the relay is on for one second and off for one second
while True:
   relay_on()
    time.sleep(1)
    relay_off()
    time.sleep(1)
```

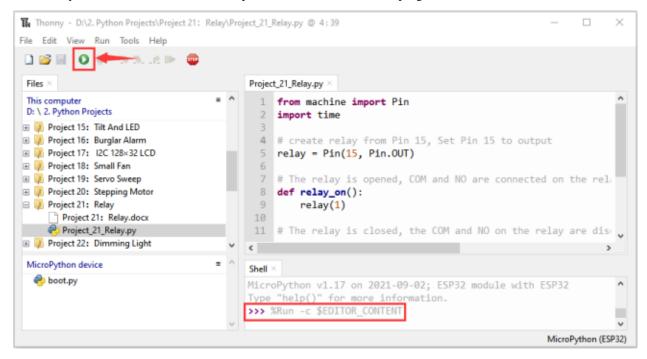
### 7.22.6 6.Project result

Make sure the ESP32 has been connected to the computer, click "" "Stop/Restart backend".



Click "Run current script", the code starts to be executed and you'll see that the relay will cycle on and off, on for 1 second, off for 1 second.

At the same time, you can hear the sound of the relay on and off, and you can also see the change of the indicator light on the relay. Press"Ctrl+C"or click "Stop/Restart backend" to exit the program.



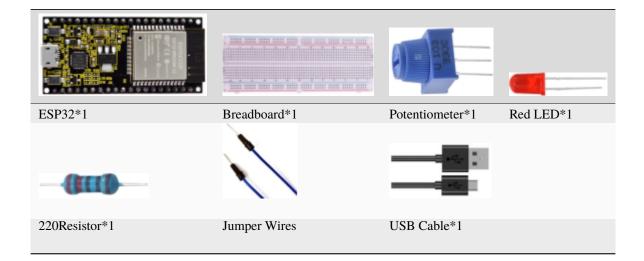
# 7.23 Project 22Dimming Light

# 7.23.1 1.Introduction

A potentiometer is a three-terminal resistor with sliding or rotating contacts that forms an adjustable voltage divider. It works by changing the position of the sliding contacts across a uniform resistance. In the potentiometer, the entire input voltage is applied across the whole length of the resistor, and the output voltage is the voltage drop between the fixed and sliding contact.

In this project, we will learn how to use ESP32 to read the values of the potentiometer, and make a dimming lamp with LED.

# 7.23.2 2.Components



## 7.23.3 3.Component knowledge



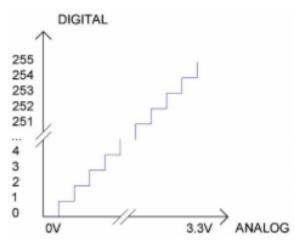
#### Adjustable potentiometer:

It is a kind of resistor and an analog electronic component, which has two states of 0 and 1(high level and low level). The analog quantity is different, its data state presents a linear state such as  $1 \sim 1024$ .

#### ADC:

An ADC is an electronic integrated circuit used to convert analog signals such as voltages to digital or binary form consisting of 1s and 0s. The range of our ADC on ESP32 is 12 bits, that means the resolution is 2^12=4096, and it represents a range (at 3.3V) will be divided equally to 4096 parts. The rage of analog values corresponds to ADC

values. So the more bits the ADC has, the denser the partition of analog will be and the greater the precision of the resulting conversion.



Subsection 1: the analog in rang of 0V—3.3/4095 V corresponds to digital 0;

Subsection 2: the analog in rang of 3.3/4095 V—2\*3.3 /4095V corresponds to digital 1;

•••

The following analog will be divided accordingly.

The conversion formula is as follows: ADC = AnalogVoltage/3.3 \* 4095 DAC

The reversing of this process requires a DAC, Digital-to-Analog Converter. The digital I/O port can output high level and low level (0 or 1), but cannot output an intermediate voltage value. This is where a DAC is useful. ESP32 has two DAC output pins with 8-bit accuracy, GPIO25 and GPIO26, which can divide VCC (here is 3.3V) into 2^8=256 parts. For example, when the digital quantity is 1, the output voltage value is 3.3/256 \*1 V, and when the digital quantity is 128, the output voltage value is 3.3/256\*128=1.65V, the higher the accuracy of DAC, the higher the accuracy of output voltage value will be.

The conversion formula is as follows: AnalogVoltage = DAC/255 \* 3.3(V) ADC on ESP32

ESP32 has 16 pins can be used to measure analog signals. GPIO pin sequence number and analog pin definition are shown in the following table

ADC number in ESP32	ESP32 GPIO number
ADC0	GPIO 36
ADC3	GPIO 39
ADC4	GPIO 32
ADC5	GPIO33
ADC6	GPIO34
ADC7	GPIO 35
ADC10	GPIO 4
ADC11	GPIO0
ADC12	GPIO2
ADC13	GPIO15
ADC14	GPIO13
ADC15	GPIO 12
ADC16	GPIO 14
ADC17	GPIO27
ADC18	GPIO25
ADC19	GPIO26

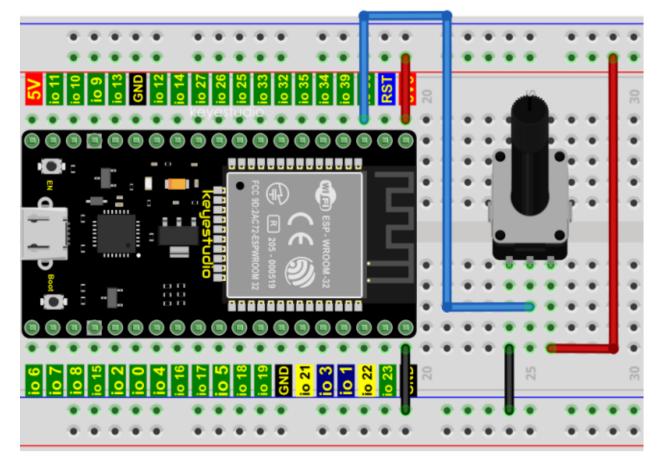
#### DAC on ESP32

ESP32 has two 8-bit digital analog converters to be connected to GPIO25 and GPIO26 pins, respectively, and it is immutable. As shown in the following table

Simulate pin number	GPIO number
DAC1	GPIO25
DAC2	GPIO26

#### 7.23.4 4.Read the ADC value, DAC value and voltage value of the potentiometer

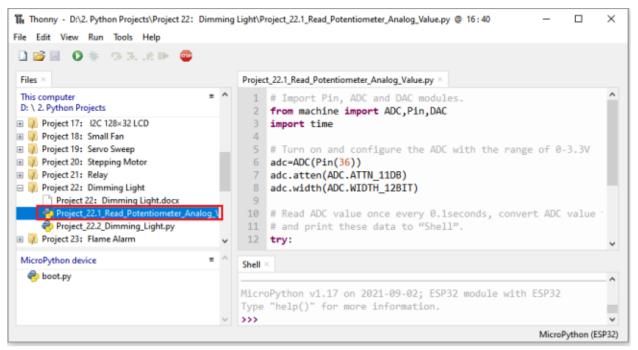
We connect the potentiometer to the analog IO port of ESP32 to read the ADC value, DAC value and voltage value of the potentiometer, please refer to the wiring diagram below



Codes used in this tutorial are saved in "2. Python Projects". If you haven't downloaded the code file, please click on the link to download it:Download Python Codes

2. Python Projects		_	
Share View			~ 🕐
→ This PC → Software (D:) → 2. Pyth	on Projects 🗸 진	,○ Search 2	2. Python
Name	Date modified	Туре	Size ^
	2/17/2022 10:21 AM	File folder	
Project 01: Hello World	2/17/2022 10:21 AM	File folder	
Project 02: Turn On LED	2/17/2022 11:10 AM	File folder	
Project 03: LED Flashing	2/17/2022 11:12 AM	File folder	
Project 04: Breathing LED	2/17/2022 10:21 AM	File folder	

 $Open ``Thonny" click ``This computer" \rightarrow ``D:" \rightarrow ``2. Python Projects" \rightarrow ``Project 22Dimming Light" and then double left-click ``Project_22.1_Read_Potentiometer_Analog_Value.py".$ 



(continued from previous page)

```
dacVal=adcVal//16
voltage = adcVal / 4095.0 * 3.3
print("ADC Val:",adcVal,"DACVal:",dacVal,"Voltage:",voltage,"V")
time.sleep(0.1)
except:
pass
```

Make sure the ESP32 has been connected to the computer,  $click^{\textcircled{m}}$  "Stop/Restart backend".

Files ×		Projec	rt_22.1_Read_Potentiometer_Analog_Value.py ×
This computer D: \ 2. Python Projects D: \ 2. Python Projects D: \ 2. Python Project 32 Project 17: 12C 128×32 LCD D: Project 19: Servo Sweep D: Project 21: Stepping Motor D: Project 22: Stepping Motor D: Project 22: Dimming Light D: Project 22: Dimming Light.docx D: Project 22: Dimming Light.docx D: Project 22: Dimming Light.docx D: Project 22: Dimming Light.py D: Project 23: Flame Alarm	= ^	1 2 3 4 5 6 <b>7</b> 8 9 10 11 12	<pre># Import Pin, ADC and DAC modules. from machine import ADC,Pin,DAC import time # Turn on and configure the ADC with the range of 0-3.3V adc-ADC(Pin(36)) adc.atten(ADC.ATTN_11DB) adc.width(ADC.WIDTH_12BIT) # Read ADC value once every 0.1seconds, convert ADC value ' # and print these data to "Shell". try:</pre>
MicroPython device		Shell	
eboot.py		Snell	

Click C"Run current script", the code starts to be executed and you'll see that the "Shell" window of Thonny IDE will print the ADC value, DAC value and voltage value of the potentiometer, turn the potentiometer handle, the ADC value

and voltage value will change. Press "Ctrl+C" or click ""Stop/Restart backend" to exit the program.

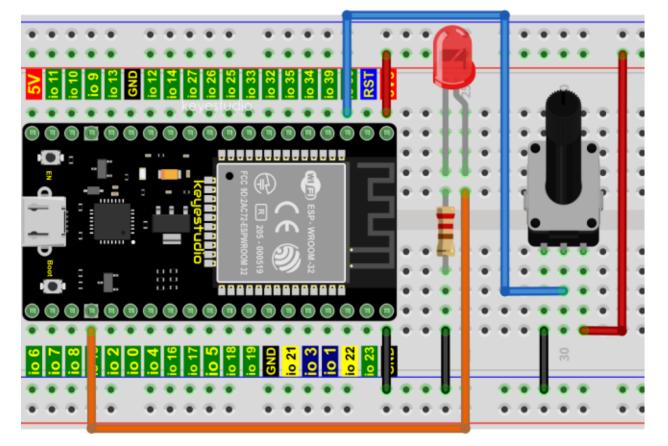
Files ×     =     ^       This computer     =     ^       D: \ 2. Python Projects     =     ^       IIII: D: \ 2. Python Project 17:     I2C 128×32 LCD     IIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIII	Project_22.1_Read_Potentiometer_Analog_Value.py × 1 # Import Pin, ADC and DAC modules. 2 from machine import ADC,Pin,DAC 3 import time 4 5 # Turn on and configure the ADC with the range of 0-3.3V 6 adc=ADC(Pin(36))	Â
	<pre>7 adc.atten(ADC.ATTN_11DB) 8 adc.width(ADC.WIDTH_12BIT) 9 10 # Read ADC value once every 0.1seconds, convert ADC value 11 # and print these data to "Shell". 12 trv:</pre>	. ,
MicroPython device = ^	Shell × MicroPython v1.17 on 2021-09-02; ESP32 module with ESP32 Type "help()" for more information. >>> %Run -c \$EDITOR_CONTENT ADC Val: 112 DACVal: 7 Voltage: 0.0902564 V ADC Val: 113 DACVal: 7 Voltage: 0.09106227 V	· · · · ·

Shell ×

onen		
	0 DACVal: 0 Voltage: 0.0 V	^
ADC Val:	23 DACVal: 1 Voltage: 0.0185348 V	
ADC Val:	48 DACVal: 3 Voltage: 0.03868132 V	
ADC Val:	268 DACVal: 16 Voltage: 0.2159707 V	
ADC Val:	559 DACVal: 34 Voltage: 0.4504762 V	
ADC Val:	0 DACVal: 0 Voltage: 0.0 V	
ADC Val:	553 DACVal: 34 Voltage: 0.445641 V	
ADC Val:	810 DACVal: 50 Voltage: 0.6527472 V	
ADC Val:	1294 DACVal: 80 Voltage: 1.042784 V	
ADC Val:	1280 DACVal: 80 Voltage: 1.031502 V	
ADC Val:	1287 DACVal: 80 Voltage: 1.037143 V	
ADC Val:	1514 DACVal: 94 Voltage: 1.220073 V	
ADC Val:	2160 DACVal: 135 Voltage: 1.740659 V	
	2162 DACVal: 135 Voltage: 1.742271 V	
	2171 DACVal: 135 Voltage: 1.749524 V	
	2467 DACVal: 154 Voltage: 1.988059 V	
	2642 DACVal: 165 Voltage: 2.129084 V	
	2640 DACVal: 165 Voltage: 2.127473 V	
	2723 DACVal: 170 Voltage: 2.194359 V	
	2911 DACVal: 181 Voltage: 2.345861 V	
	3008 DACVal: 188 Voltage: 2.424029 V	
	3029 DACVal: 189 Voltage: 2.440952 V	
	3140 DACVal: 196 Voltage: 2.530403 V	
	3271 DACVal: 204 Voltage: 2.635971 V	
	3583 DACVal: 223 Voltage: 2.887399 V	
ADC Val:	3664 DACVal: 229 Voltage: 2.952674 V	۷

# 7.23.5 5. Wiring diagram of the dimming lamp

In the previous step, we read the ADC value, DAC value and voltage value of the potentiometer. Now we need to convert the ADC value of the potentiometer into the brightness of the LED to make a lamp that can adjust the brightness. The wiring diagram is as follows:



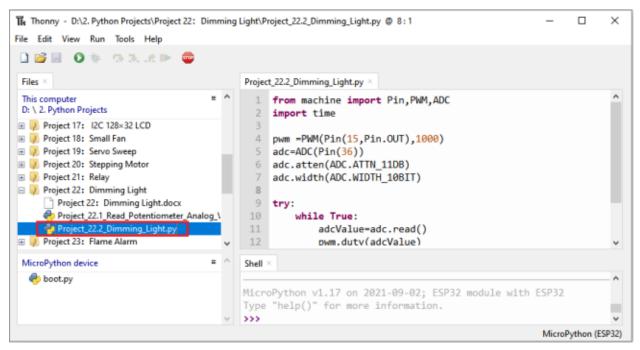
# 7.23.6 6.Project code

Codes used in this tutorial are saved in "2. Python Projects". If you haven't downloaded the code file, please click on the link to download it:Download Python Codes

2. Python Projects		_	$\Box$ $\times$
Share View			~ 🕐
→ This PC → Software (D:) → 2. Pyth	on Projects 🗸 🗸	ク Search 2	2. Python
Name	Date modified	Туре	Size ^
Project 00: Boot	2/17/2022 10:21 AM	File folder	
	2/17/2022 10:21 AM	File folder	
	2/17/2022 11:10 AM	File folder	
	2/17/2022 11:12 AM	File folder	
Project 04: Breathing LED	2/17/2022 10:21 AM	File folder	

Open"Thonny"click"This computer" $\rightarrow$ "D:" $\rightarrow$ "2. click"Project\_22.2\_Dimming\_Light.py".

Python Projects" $\rightarrow$ "Project 22Dimming Light" and then



```
from machine import Pin,PWM,ADC
import time

pwm =PWM(Pin(15,Pin.OUT),1000)
adc=ADC(Pin(36))
adc.atten(ADC.ATTN_11DB)
adc.width(ADC.WIDTH_10BIT)

try:
    while True:
        adcValue=adc.read()
        pwm.duty(adcValue)
```

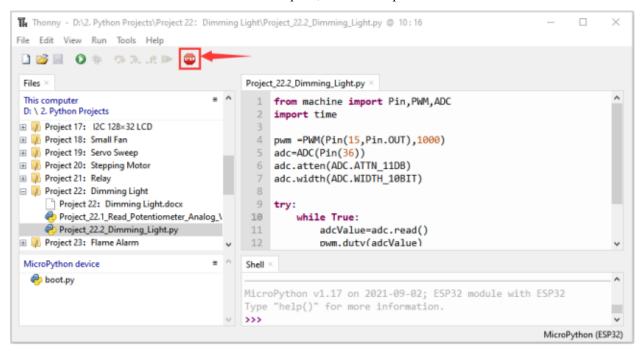
print(adc.read())

```
time.sleep_ms(100)
except:
```

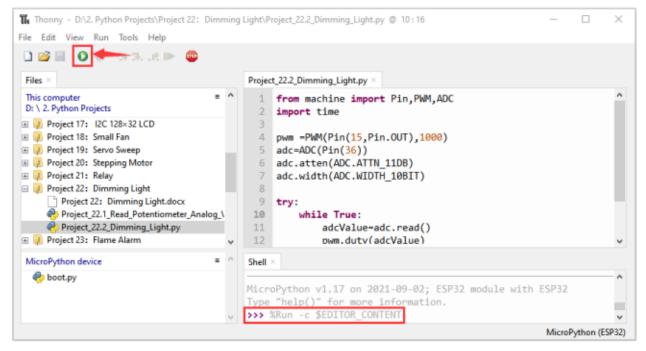
```
pwm.deinit()
```

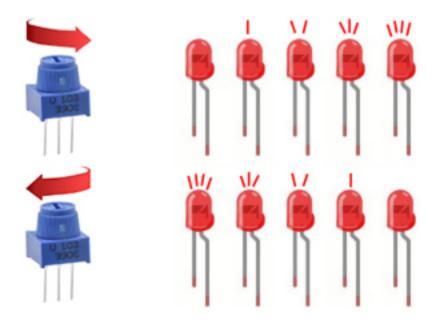
# 7.23.7 7.Project result

Make sure the ESP32 has been connected to the computer, click "Stop/Restart backend".



Click<sup>©</sup> "Run current script", the code starts to be executed and you'll see that turn the potentiometer handle and the brightness of the LED will change accordingly. Press "Ctrl+C" or click<sup>©</sup> "Stop/Restart backend" to exit the program.





# 7.24 Project 23Flame Alarm

# 7.24.1 1.Introduction

Fire is a terrible disaster and fire alarm systems are very useful in housescommercial buildings and factories. In this project, we will use ESP32 to control a flame sensor, a buzzer and a LED to simulate fire alarm devices. This is a meaningful maker activity.

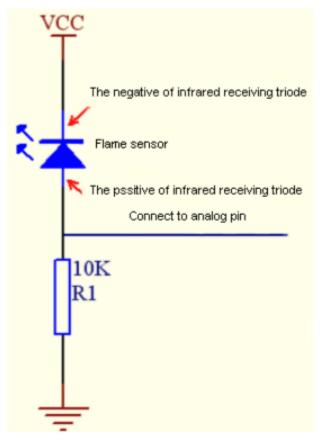
## 7.24.2 2.Components

			C PERCENT
ESP32*1	Breadboard*1	Red LED*1	Active Buzzer*1
		<b>(m</b> )-	
Flame Sensor*1	220Resistor*1	10KResistor*1	Jumper Wires
	-9110		
NPN transistor(S8050)*1	1k Resistor*1	USB Cable*1	

## 7.24.3 3.Component knowledge



The flame emits a certain amount IR light that is invisible to the human eye, but our flame sensor can detect it and alert a microcontroller (such as ESP32) that a fire has been detected. It has a specially designed infrared receiver tube to detect the flame and then convert the flame brightness into a fluctuating level signal. The short pin of the receiving triode is negative pole and the other long pin is positive pole. We should connect the short pin (negative) to 5V and the long pin (positive) to the analog pin, a resistor and GND. As shown in the figure below

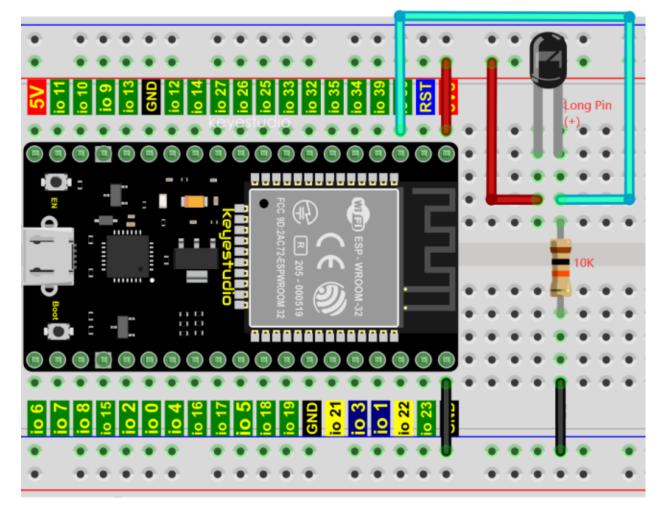


#### Note:

Since vulnerable to radio frequency radiation and temperature changes, the flame sensor should be kept away from heat sources like radiators, heaters and air conditioners, as well as direct irradiation of sunlight, headlights and incandescent light.

## 7.24.4 4.Read the ADC value, DAC value and voltage value of the flame sensor

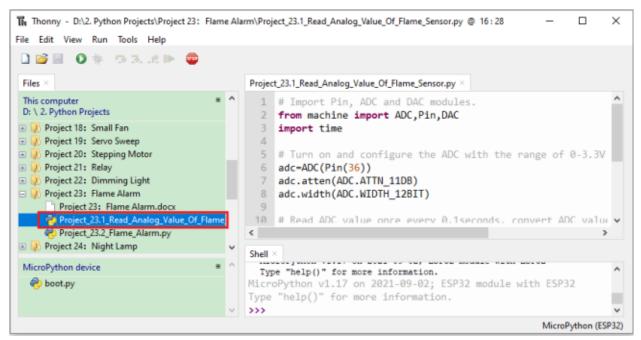
We first use a simple code to read the ADC value, DAC value and voltage value of the flame sensor and print them out. Please refer to the wiring diagram below



Codes used in this tutorial are saved in "**2. Python Projects**". If you haven't downloaded the code file, please click on the link to download it:Download Python Codes

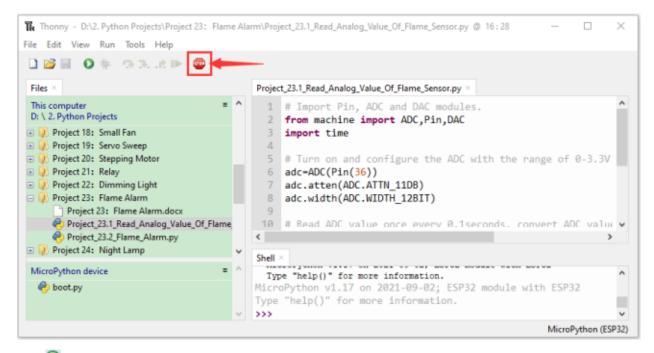
2. Python Projects		_	$\Box$ $\times$
Share View			~ 🕐
→ This PC → Software (D:) → 2. Pyth	non Projects 🗸 🗸 🗸	ල 🔎 Search	n 2. Python
Name	Date modified	Туре	Size ^
	2/17/2022 10:21 AM	File folder	
Project 01: Hello World	2/17/2022 10:21 AM	File folder	
Project 02: Turn On LED	2/17/2022 11:10 AM	File folder	
Project 03: LED Flashing	2/17/2022 11:12 AM	File folder	
Project 04: Breathing LED	2/17/2022 10:21 AM	File folder	

Open"Thonny", click "This computer"  $\rightarrow$  "D:"  $\rightarrow$  "2. Python Projects"  $\rightarrow$  "Project 23Flame Alarm" and then double left-click "Project\_23.1\_Read\_Analog\_Value\_Of\_Flame\_Sensor.py".



```
## Import Pin, ADC and DAC modules.
from machine import ADC,Pin,DAC
import time
## Turn on and configure the ADC with the range of 0-3.3V
adc=ADC(Pin(36))
adc.atten(ADC.ATTN_11DB)
adc.width(ADC.WIDTH_12BIT)
## Read ADC value once every 0.1seconds, convert ADC value to DAC value and output it,
## and print these data to "Shell".
try:
   while True:
        adcVal=adc.read()
        dacVal=adcVal//16
        voltage = adcVal / 4095.0 * 3.3
        print("ADC Val:",adcVal,"DACVal:",dacVal,"Voltage:",voltage,"V")
        time.sleep(0.1)
except:
   pass
```

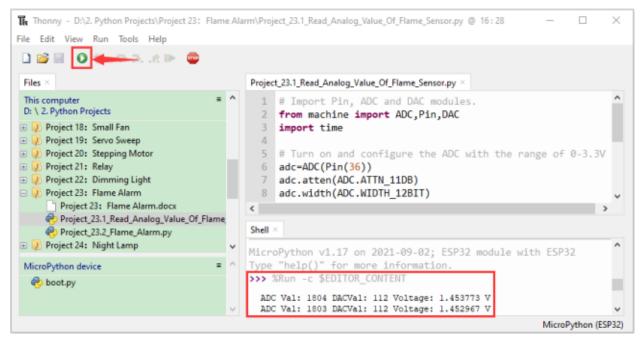
Make sure the ESP32 has been connected to the computer, click <sup>11</sup> "Stop/Restart backend".



Click "Run current script", the code starts to be executed and you'll see that the "Shell" window of Thonny IDE will print the ADC valueDAC value and voltage value of the flame sensor.

When the flame is close to the flame sensor, the ADC value, DAC value and voltage value increase; Conversely, the

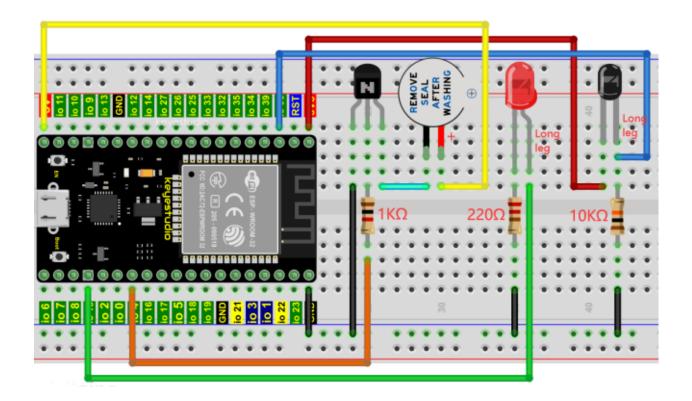
ADC value, DAC value and voltage value decrease. Press "Ctrl+C" or click ""Stop/Restart backend" to exit the program.



Shell ×	
ADC Val: 0 DACVal: 0 Voltage: 0.0 V	~
ADC Val: 23 DACVal: 1 Voltage: 0.0185348 V	
ADC Val: 48 DACVal: 3 Voltage: 0.03868132 V	
ADC Val: 268 DACVal: 16 Voltage: 0.2159707 V	
ADC Val: 559 DACVal: 34 Voltage: 0.4504762 V	
ADC Val: 0 DACVal: 0 Voltage: 0.0 V	
ADC Val: 553 DACVal: 34 Voltage: 0.445641 V	
ADC Val: 810 DACVal: 50 Voltage: 0.6527472 V	
ADC Val: 1294 DACVal: 80 Voltage: 1.042784 V	
ADC Val: 1280 DACVal: 80 Voltage: 1.031502 V	
ADC Val: 1287 DACVal: 80 Voltage: 1.037143 V	
ADC Val: 1514 DACVal: 94 Voltage: 1.220073 V	
ADC Val: 2160 DACVal: 135 Voltage: 1.740659 V	
ADC Val: 2162 DACVal: 135 Voltage: 1.742271 V	
ADC Val: 2171 DACVal: 135 Voltage: 1.749524 V	
ADC Val: 2467 DACVal: 154 Voltage: 1.988059 V	
ADC Val: 2642 DACVal: 165 Voltage: 2.129084 V	
ADC Val: 2640 DACVal: 165 Voltage: 2.127473 V	
ADC Val: 2723 DACVal: 170 Voltage: 2.194359 V	
ADC Val: 2911 DACVal: 181 Voltage: 2.345861 V	
ADC Val: 3008 DACVal: 188 Voltage: 2.424029 V	
ADC Val: 3029 DACVal: 189 Voltage: 2.440952 V	
ADC Val: 3140 DACVal: 196 Voltage: 2.530403 V	
ADC Val: 3271 DACVal: 204 Voltage: 2.635971 V	
ADC Val: 3583 DACVal: 223 Voltage: 2.887399 V	
ADC Val: 3664 DACVal: 229 Voltage: 2.952674 V	$\sim$

# 7.24.5 5. Wiring diagram of the flame alarm

Next, we will use a flame sensor, a buzzer, and a LED to make an interesting project, that is flame alarm. When flame is detected, the LED flashes and the buzzer alarms.



# 7.24.6 6.Project code

Note:

1

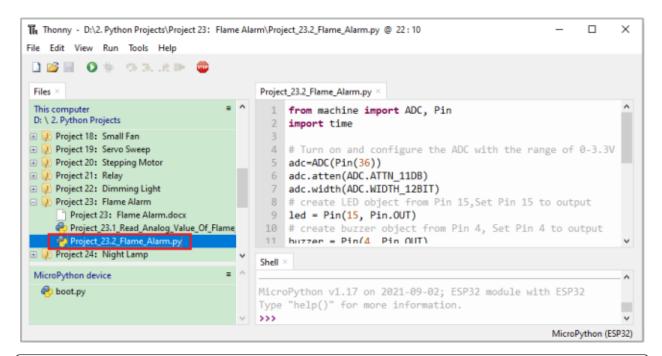
value >500:

the threshold of 500 in the code can be reset itself as required)

Codes used in this tutorial are saved in "2. Python Projects". If you haven't downloaded the code file, please click on the link to download it:Download Python Codes

2. Python Projects		_	
Share View			~ 🕐
> This PC > Software (D:) > 2. Pyth	non Projects 🗸 진	, ○ Search 2	. Python
Name	Date modified	Туре	Size ^
Project 00: Boot	2/17/2022 10:21 AM	File folder	
- Project 01: Hello World	2/17/2022 10:21 AM	File folder	
	2/17/2022 11:10 AM	File folder	
Project 03: LED Flashing	2/17/2022 11:12 AM	File folder	
Project 04: Breathing LED	2/17/2022 10:21 AM	File folder	

 $Open "Thonny" click "This computer" \rightarrow "D:" \rightarrow "2. Python Projects" \rightarrow "Project 23Flame Alarm", and then double left-click "Project_23.2_Flame_Alarm.py".$ 



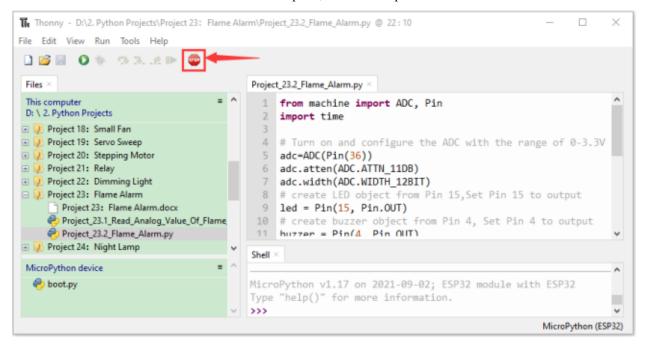
```
from machine import ADC, Pin
import time
## Turn on and configure the ADC with the range of 0-3.3V
adc=ADC(Pin(36))
```

```
adc.atten(ADC.ATTN_11DB)
adc.width(ADC.WIDTH_12BIT)
## create LED object from Pin 15, Set Pin 15 to output
led = Pin(15, Pin.OUT)
## create buzzer object from Pin 4, Set Pin 4 to output
buzzer = Pin(4, Pin.OUT)
## If the flame sensor detects a flame, the buzzer will beep
## and the LED will blink when the analog value is greater than 500
## Otherwise, the buzzer does not sound and the LED goes off
while True:
    adcVal=adc.read()
   if adcVal >500:
                         # Set buzzer turn on
       buzzer.value(1)
                      # Set led turn on
        led.value(1)
        time.sleep(0.5) # Sleep 0.5s
       buzzer.value(0)
                      # Set led turn off
        led.value(0)
        time.sleep(0.5) # Sleep 0.5s
   else:
       buzzer.value(0) # Set buzzer turn off
                        # Set led turn off
```

led.value(♥)

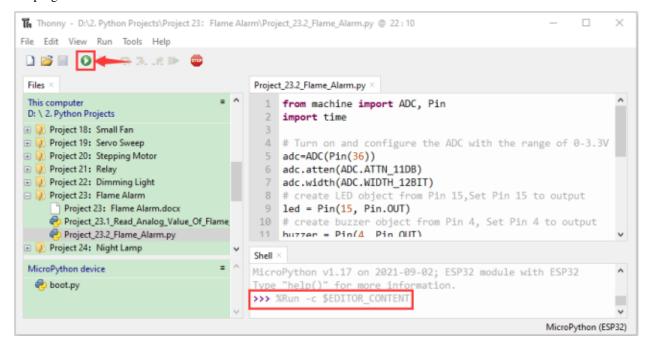
### 7.24.7 7.Project result

Make sure the ESP32 has been connected to the computer, click <sup>1</sup> "Stop/Restart backend".



Click "Run current script", the code starts to be executed and you'll see that when the flame sensor detects the flame, the LED flashes and the buzzer alarms.

Otherwise, the LED does not light, the buzzer does not sound.Press"Ctrl+C"or click <sup>\*\*\*</sup> "Stop/Restart backend" to exit the program.



# 7.25 Project 24Night Lamp

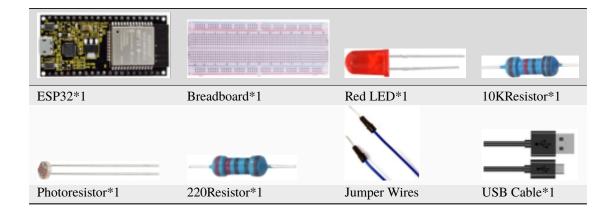
### 7.25.1 1.Introduction

Sensors or components are ubiquitous in our daily life. For example, some public street lamps will automatically turn on at night and turn off during the day.

Why? In fact, this make use of a photosensitive element that senses the intensity of external ambient light.

When the outdoor brightness decreases at night, the street lights will turn on automatically; In the daytime, the street lights will automatically turn off. the principle of which is very simple, In this Project, we use ESP32 to control a LED to achieve the effect of the street light.

# 7.25.2 2.Components



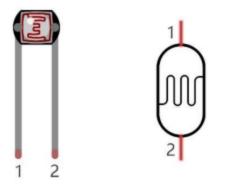
## 7.25.3 3.Component knowledge



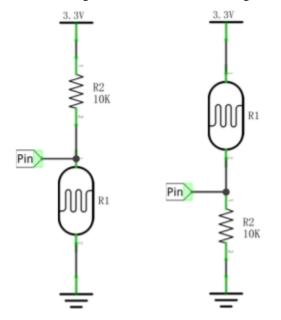
#### **Photoresistor :**

It is a kind of photosensitive resistance, its principle is that the photoresistor surface receives brightness (light) to reduce the resistance, the resistance value will change with the detected intensity of the ambient light. With this characteristic, we can use the photosensitive resistance to detect the light intensity.

Photosensitive resistance and its electronic symbol are as follows



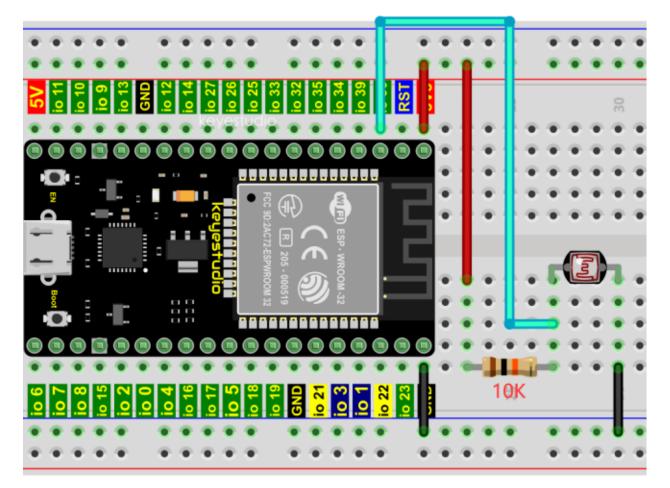
The following circuit is used to detect changes in resistance values of photoresistors



In the circuit above, when the resistance of the photoresistor changes due to the change of light intensity, the voltage between the photoresistor and resistance R2 will also change. Thus, the intensity of light can be obtained by measuring this voltage.

#### 7.25.4 4.Read the ADC value, DAC value and voltage value of the photoresistor

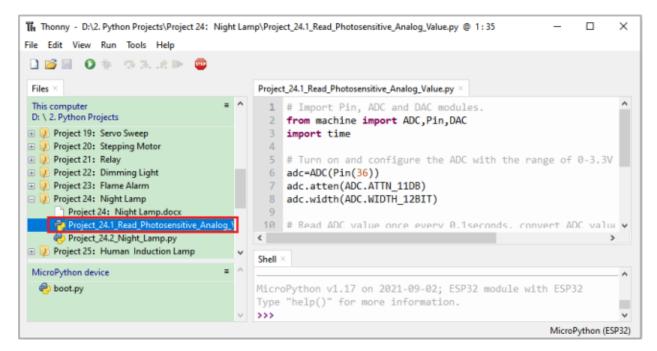
We first use a simple code to read the ADC value, DAC value and voltage value of the photoresistor and print them out. Please refer to the following wiring diagram



Codes used in this tutorial are saved in "2. Python Projects". If you haven't downloaded the code file, please click on the link to download it:Download Python Codes

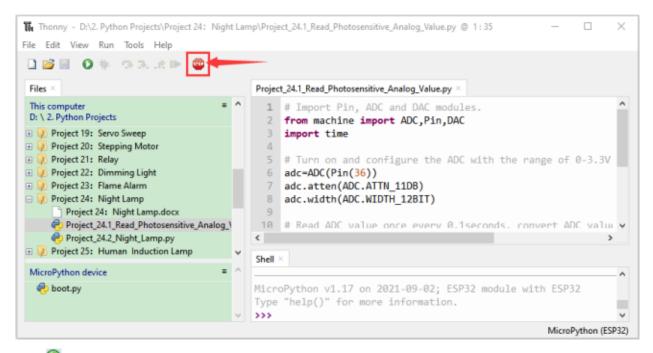
 $\times$ 2. Python Projects Share View ? This PC > Software (D:) > 2. Python Projects Search 2. Python ... Ū Q > ~ Name Size ^ Date modified Type Project 00: Boot 2/17/2022 10:21 AM File folder Project 01: Hello World File folder 2/17/2022 10:21 AM Project 02: Turn On LED 2/17/2022 11:10 AM File folder Project 03: LED Flashing 2/17/2022 11:12 AM File folder Project 04: Breathing LED 2/17/2022 10:21 AM File folder

Open"Thonny"click"This computer" $\rightarrow$ "D:" $\rightarrow$ "2. Python Projects" $\rightarrow$ "Project 24Night Lamp"and then double left-click "Project\_24.1\_Read\_Photosensitive\_Analog\_Value.py".



```
## Import Pin, ADC and DAC modules.
from machine import ADC,Pin,DAC
import time
## Turn on and configure the ADC with the range of 0-3.3V
adc=ADC(Pin(36))
adc.atten(ADC.ATTN_11DB)
adc.width(ADC.WIDTH_12BIT)
## Read ADC value once every 0.1seconds, convert ADC value to DAC value and output it,
## and print these data to "Shell".
try:
   while True:
        adcVal=adc.read()
        dacVal=adcVal//16
        voltage = adcVal / 4095.0 * 3.3
        print("ADC Val:",adcVal,"DACVal:",dacVal,"Voltage:",voltage,"V")
        time.sleep((0.1))
except:
   pass
```

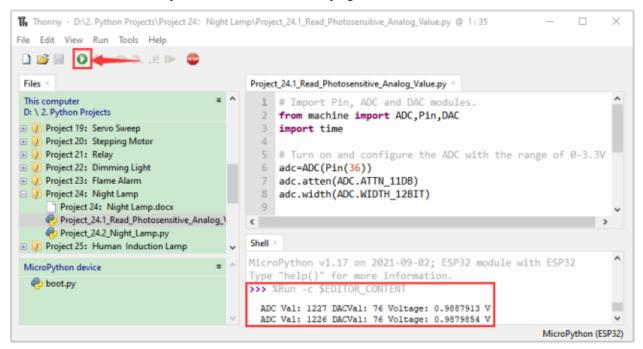
Make sure the ESP32 has been connected to the computer, click "Stop/Restart backend".



Click Circle Click Circle Content and Second and Second Circle Content and Second Circle Circ

When the light intensity around the photoresistor is gradually reduced, the ADC valueDAC value and voltage value will gradually increase. On the contrary, the ADC value, DAC value and voltage value decreases gradually.

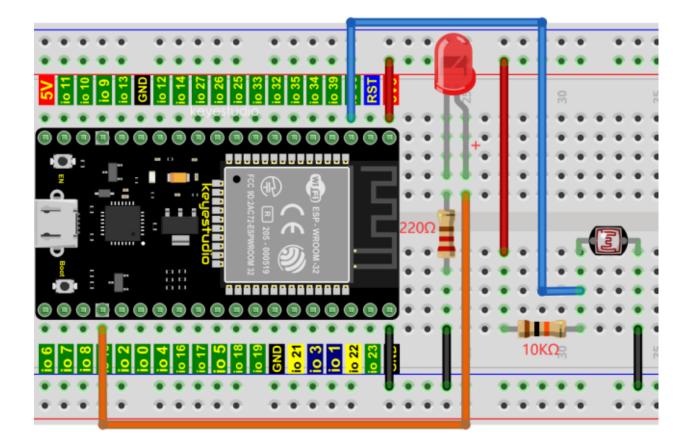
Press"Ctrl+C"or click "Stop/Restart backend" to exit the program.



Shell ×				
ADC Val:	1472 DA	ACVal: 92 V	Voltage: 1.186227 V	~
			Voltage: 1.325641 V	
ADC Val:	1847 DA	ACVal: 115	Voltage: 1.488425 V	
ADC Val:	2043 DA	ACVal: 127	Voltage: 1.646374 V	
ADC Val:	2254 DA	ACVal: 140	Voltage: 1.81641 V	
ADC Val:	2442 DA	ACVal: 152	Voltage: 1.967912 V	
ADC Val:	2625 DA	ACVal: 164	Voltage: 2.115385 V	
ADC Val:	2752 DA	ACVal: 172	Voltage: 2.217729 V	
ADC Val:	2832 DA	ACVal: 177	Voltage: 2.282198 V	
ADC Val:	2880 DA	ACVal: 180	Voltage: 2.320879 V	
ADC Val:	2887 DA	ACVal: 180	Voltage: 2.32652 V	
			Voltage: 2.315238 V	
ADC Val:	2922 DA	ACVal: 182	Voltage: 2.354725 V	
ADC Val:	2991 DA	ACVal: 186	Voltage: 2.41033 V	
ADC Val:	3051 DA	ACVal: 190	Voltage: 2.458681 V	
ADC Val:	3103 DA	ACVal: 193	Voltage: 2.500586 V	
ADC Val:	3145 DA	ACVal: 196	Voltage: 2.534432 V	
ADC Val:	3163 DA	ACVal: 197	Voltage: 2.548938 V	
ADC Val:	3181 DA	ACVal: 198	Voltage: 2.563443 V	
ADC Val:	3187 DA	ACVal: 199	Voltage: 2.568278 V	1.00
ADC Val:	3214 DA	ACVal: 200	Voltage: 2.590037 V	
ADC Val:	3314 DA	ACVal: 207	Voltage: 2.670623 V	$\sim$

# 7.25.5 5.Wiring diagram of the light-controlled lamp

We made a small dimming lamp in the front, now we will make a light controlled lamp. The principle is the same, that is, the ESP32 takes the ADC value of the sensor, and then adjusts the brightness of the LED.

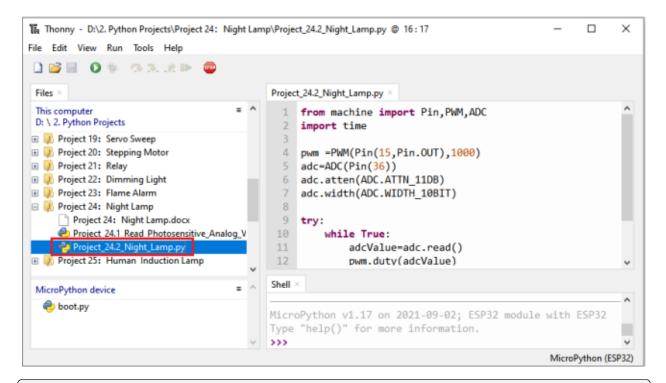


## 7.25.6 6.Project code

Codes used in this tutorial are saved in "2. Python Projects". If you haven't downloaded the code file, please click on the link to download it:Download Python Codes

2. Python Projects		— C	) ×
Share View			~ ?
→ This PC → Software (D:) → 2. Pythe	on Projects 🗸 진		Python
Name	Date modified	Туре	Size ^
Project 00: Boot	2/17/2022 10:21 AM	File folder	
Project 01: Hello World	2/17/2022 10:21 AM	File folder	
Project 02: Turn On LED	2/17/2022 11:10 AM	File folder	
Project 03: LED Flashing	2/17/2022 11:12 AM	File folder	
Project 04: Breathing LED	2/17/2022 10:21 AM	File folder	

 $\label{eq:constraint} Open ``Thonny" click ``This computer" \rightarrow ``D:" \rightarrow ``2. Python Projects" \rightarrow ``Project 24Night Lamp" and then double left-click ``Project_24.2_Night_Lamp.py".$ 



```
from machine import Pin,PWM,ADC
import time

pwm =PWM(Pin(15,Pin.OUT),1000)
adc=ADC(Pin(36))
adc.atten(ADC.ATTN_11DB)
adc.width(ADC.WIDTH_10BIT)

try:
    while True:
        adcValue=adc.read()
        pwm.duty(adcValue)
        print(adc.read())
        time.sleep_ms(100)

except:
        pwm.deinit()
```

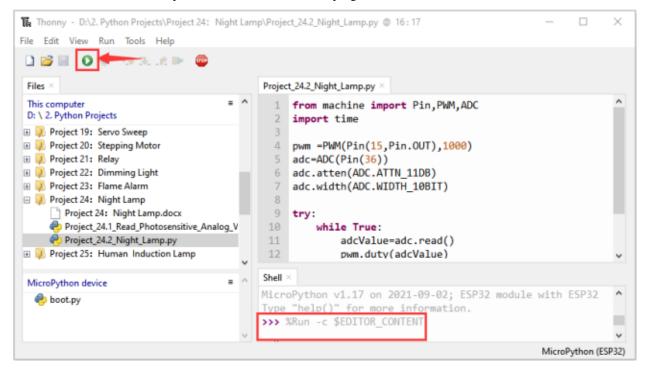
#### 7.25.7 7.Project result

Make sure the ESP32 has been connected to the computer,  $click \stackrel{\text{result}}{=} "Stop/Restart backend"$ .

The Thonny - D:\2. Python Projects\Project 24: N File Edit View Run Tools Help	ight Lan	amp\Project_24.2_Night_Lamp.py @ 16:17 —	×
		Project_24.2_Night_Lamp.py ×	
This computer D: \ 2. Python Projects Project 19: Servo Sweep Project 20: Stepping Motor Project 21: Relay Project 22: Dimming Light Project 23: Flame Alarm Project 24: Night Lamp Project 24: Night Lamp. Project 24: Night Lamp.docx Project 24: Night Lamp.docx Project 24: Night Lamp.py Project 24: Night Lamp.py Project 25: Human Induction Lamp	= ^	<pre>from machine import Pin,PWM,ADC import time pwm =PWM(Pin(15,Pin.OUT),1000) adc=ADC(Pin(36)) adc.atten(ADC.ATTN_11DB) adc.width(ADC.WIDTH_10BIT) try: while True: adcValue=adc.read() pwm.duty(adcValue)</pre>	~
MicroPython device	= ^	Shell × MicroPython v1.17 on 2021-09-02; ESP32 module with ESP32	^
	~	Type "help()" for more information. >>> MicroPython (ESI	v P32)

Click "Run current script", the code starts to be executed and you'll see that when the intensity of light around the photoresistor is reduced, the LED will be bright, on the contraty, the LED will be dim.

Press"Ctrl+C"or click "Stop/Restart backend" to exit the program.



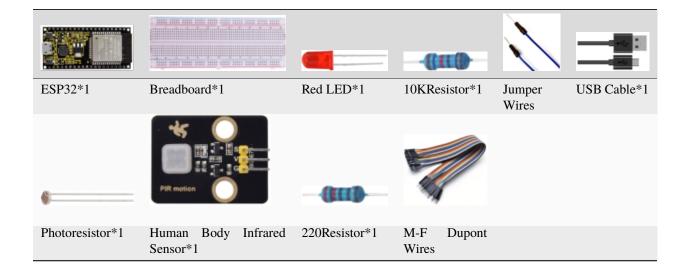
# 7.26 Project 25Human Induction Lamp

#### 7.26.1 1.Introduction

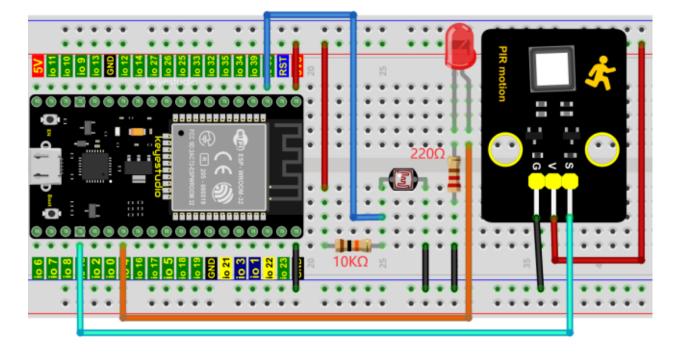
Human body induction lamp is used commonly in the dark corridor area. With the development of science and technology, the use of the human body induction lamp is very common in our real life, such as the corridor of the community, the bedroom of the room, the garage of the dungeon, the bathroom and so on. The human induction lamp are generally composed of a human body infrared sensor, a led, a photoresistor sensor and so on.

In this project, we will learn how to use a Human Body Infrared Sensor, a led, and a photoresistor to make a human induction lamp.

### 7.26.2 2.Components



### 7.26.3 3.Wiring Diagram

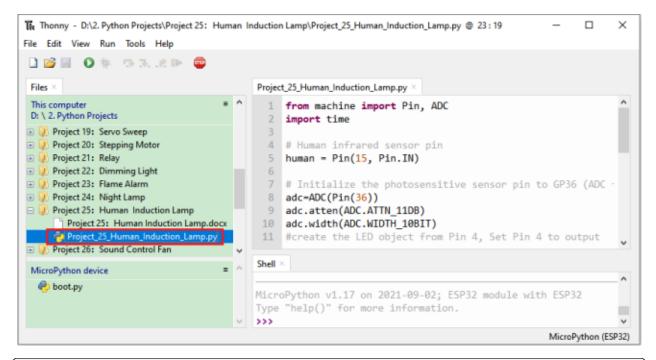


## 7.26.4 4.Project code

Codes used in this tutorial are saved in "2. Python Projects". If you haven't downloaded the code file, please click on the link to download it:Download Python Codes

> This PC ⇒ Software (D:) ⇒ 2. Pyt	hon Projects 🗸 🗸	ک 🔎 🔎 Search	2. Python
Name	Date modified	Туре	Size
Project 00: Boot	2/17/2022 10:21 AM	File folder	
Project 01: Hello World	2/17/2022 10:21 AM	File folder	
Project 02: Turn On LED	2/17/2022 11:10 AM	File folder	
Project 03: LED Flashing	2/17/2022 11:12 AM	File folder	
Project 04: Breathing LED	2/17/2022 10:21 AM	File folder	

Open"Thonny", click"This computer" $\rightarrow$ "D:" $\rightarrow$ "2. Python Projects" $\rightarrow$ "Project 25Human Induction Lamp"and then double left-click "Project\_25\_Human\_Induction\_Lamp.py".



```
from machine import Pin, ADC
import time
## Human infrared sensor pin
human = Pin(15, Pin.IN)
## Initialize the photosensitive sensor pin to GP36 (ADC function)
adc=ADC(Pin(36))
adc.atten(ADC.ATTN_11DB)
adc.width(ADC.WIDTH_10BIT)
#create the LED object from Pin 4, Set Pin 4 to output
led = Pin(4, Pin.OUT)
def detect_someone():
   if human.value() == 1:
       return True
   return False
abc = ≬
while True:
   adcVal=adc.read()
   if adcVal >= 500:
        if detect_someone() == True:
            abc += 1
            led.value(1)
            print("value=", abc)
            time.sleep(1)
        else:
            if abc != 0:
                abc = ≬
```

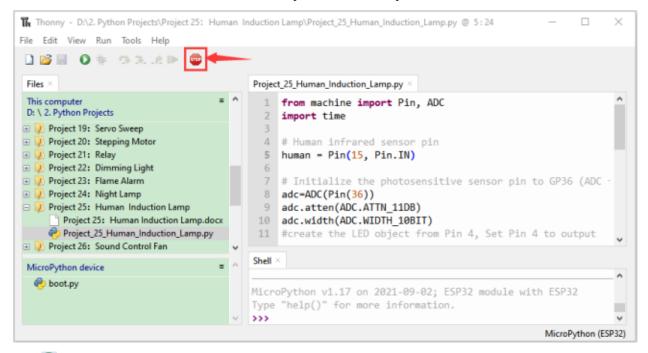
(continues on next page)

(continued from previous page)

led.value(0)
else:
 led.value(0)
time.sleep(0.1)

### 7.26.5 5.Project result

Make sure the ESP32 has been connected to the computer, click "Stop/Restart backend".



Click "Run current script", the code starts to be executed and you'll see that When your hand covers the photosensitive part of the photoresistor to simulate darkness, then shake your other hand in front of the Human Body Infrared Sensor, the external LED will light up, and after a delay of a few seconds, the external LED will automatically turn off.

At the same time, the "Shell" window of Thonny IDE will print the delay time when the external LED lights up. If the photosensitive part of the photoresistor is not covered, then shake your hand in front of the human infrared sensor and

the LED is turned off. Press "Ctrl+C" or click ""Stop/Restart backend" to exit the program.

Files × This computer D: \ 2. Python Projects	^	Projec	t_25_Human_Induction_Lamp.py × from machine import Pin, ADC import time	í
<ul> <li>Project 19: Servo Sweep</li> <li>Project 20: Stepping Motor</li> <li>Project 21: Relay</li> </ul>		3 4 5	<pre># Human infrared sensor pin human = Pin(15, Pin.IN)</pre>	ļ
<ul> <li>Project 22: Dimming Light</li> <li>Project 23: Flame Alarm</li> <li>Project 24: Night Lamp</li> </ul>		6 7 8	<pre># Initialize the photosensitive sensor pin to GP36 (ADC - adc-ADC(Pin(36))</pre>	
<ul> <li>Were Project 25: Human Induction Lamp</li> <li>Project 25: Human Induction Lamp.docs</li> <li>Project 25: Human Induction_Lamp.py</li> <li>Project 26: Sound Control Fan</li> </ul>		9 10 11	<pre>adc.atten(ADC.ATTN_11DB) adc.width(ADC.WIDTH_10BIT) #create the LED object from Pin 4, Set Pin 4 to output</pre>	,
MicroPython device		Shell		

Shell ×	
<pre>&gt;&gt;&gt; %Run -c \$EDITOR_CONTENT</pre>	^
value= 1	
value= 1	
value= 2	
value= 3	
value= 4	
value= 5	
value= 6	
value= 7	
value= 8	
	~

# 7.27 Project 26Sound Control Fan

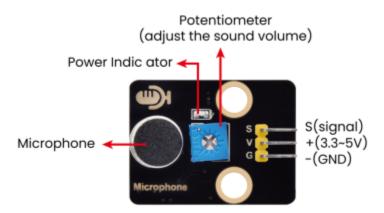
#### 7.27.1 1.Introduction

The sound sensor has a built-in capacitive electret microphone and power amplifier which can be used to detect the sound intensity of the environment. In this project, we use ESP32 to control the sound sensor and the motor module to simulate a voice-controlled fan.

## 7.27.2 2.Components

		Warsham
ESP32*1	Breadboard*1	Sound Sensor*1
130 Motor Module*1	M-F Dupont Wires	USB Cable*1
Keyestudio bread board special power mod- ule*1	Battery Holder*1	No.5 battery (self- provided)*6
Fan*1		

#### 7.27.3 3.Component knowledge

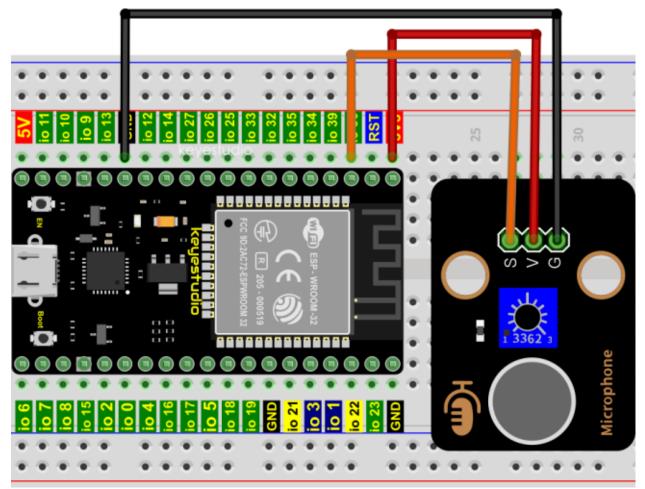


Sound sensor is usually used to detect the loudness of the sound in the surrounding environment. Microcontrol board can collect its output signal through the analog input interface. The S pin is an analog output, which is the real-time output of the microphone voltage signal. The sensor comes with a potentiometer so you can adjust the signal strength. It also has two fixing holes so that the sensor can be installed on any other equipment. You can use it to make some

interactive works, such as voice-operated switches.

#### 7.27.4 4.Read the ADC value, DAC value and voltage value of the sound sensor

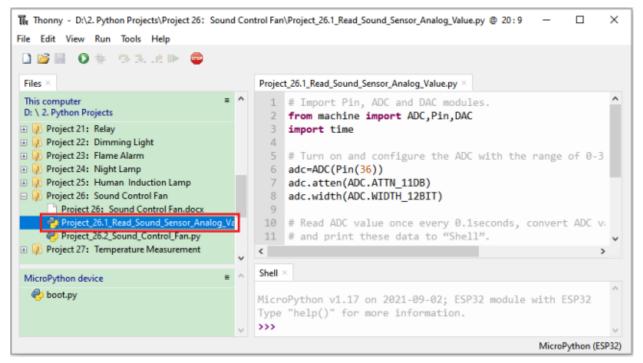
We first use a simple code to read the ADC value, DAC value and voltage value of the sound sensor and print them out. Please refer to the wiring diagram below



Codes used in this tutorial are saved in "2. Python Projects". If you haven't downloaded the code file, please click on the link to download it:Download Python Codes

. Python Projects		_	$\Box$ $\times$
Share View			~ ?
→ This PC → Software (D:) → 2. Pyth	on Projects 🗸 こ	,○ Search 2	. Python
Name	Date modified	Туре	Size ^
Project 00: Boot	2/17/2022 10:21 AM	File folder	
Project 01: Hello World	2/17/2022 10:21 AM	File folder	
Project 02: Turn On LED	2/17/2022 11:10 AM	File folder	
Project 03: LED Flashing	2/17/2022 11:12 AM	File folder	
Project 04: Breathing LED	2/17/2022 10:21 AM	File folder	

 $\label{eq:constraint} Open ``Thonny'' click'' This computer'' \rightarrow ``D:'' \rightarrow ``2. Python Projects'' \rightarrow ``Project 26Sound Control Fan'', and then double left-click ``Project_26.1_Read_Sound_Sensor_Analog_Value.py''.$ 



(continued from previous page)

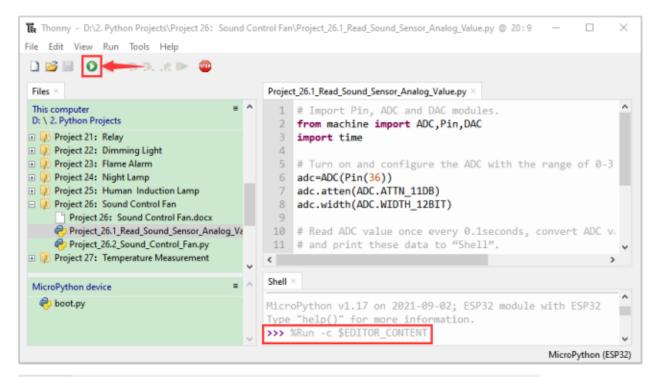


Thonny - D:\2. Python Projects\Project 26: Sound Control Fan\Project 26.1 Read Sound Sensor Analog Value.py @ 20:9 × File Edit View Run Tools Help 🔘 🎄 🖓 3. .e 🕨 🗋 📂 🔲 Files Project\_26.1\_Read\_Sound\_Sensor\_Analog\_Value.py # Import Pin, ADC and DAC modules. This computer 1 D: \ 2. Python Projects from machine import ADC, Pin, DAC 2 🗄 📜 Project 21: Relay import time 🗉 🔑 Project 22: Dimming Light 🗉 🕖 Project 23: Flame Alarm # Turn on and configure the ADC with the range of 0-3 🗉 🕖 Project 24: Night Lamp adc=ADC(Pin(36)) Project 25: Human Induction Lamp adc.atten(ADC.ATTN\_11DB) Project 26: Sound Control Fan adc.width(ADC.WIDTH\_12BIT) 8 Project 26: Sound Control Fan.docx Project\_26.1\_Read\_Sound\_Sensor\_Analog\_Va 10 # Read ADC value once every 0.1seconds, convert ADC Project\_26.2\_Sound\_Control\_Fan.py 11 # and print these data to "Shell". Project 27: Temperature Measurement < || Shell MicroPython device doot.py MicroPython v1.17 on 2021-09-02; ESP32 module with ESP32 Type "help()" for more information. >>> MicroPython (ESP32)

Make sure the ESP32 has been connected to the computer, click "" "Stop/Restart backend".

Click "Run current script", the code starts to be executed and you'll see that the "Shell" window of Thonny IDE will print the ADC valueDAC value and voltage value of the sound sensor.

When you clap your hands to the sensor, the ADC value, DAC value and voltage value will change significantly. Press"Ctrl+C"or click "Stop/Restart backend"to exit the program.



```
Shell ×

>>> %Run -c $EDITOR_CONTENT

ADC Val: 0 DACVal: 0 Voltage: 0.0 V

ADC Val: 23 DACVal: 1 Voltage: 0.0185348 V

ADC Val: 1520 DACVal: 95 Voltage: 1.224908 V

ADC Val: 551 DACVal: 34 Voltage: 0.4440293 V

ADC Val: 2285 DACVal: 142 Voltage: 1.841392 V

ADC Val: 1395 DACVal: 87 Voltage: 1.124176 V

ADC Val: 0 DACVal: 0 Voltage: 0.0 V

ADC Val: 1902 DACVal: 118 Voltage: 1.532747 V

ADC Val: 0 DACVal: 0 Voltage: 0.0 V

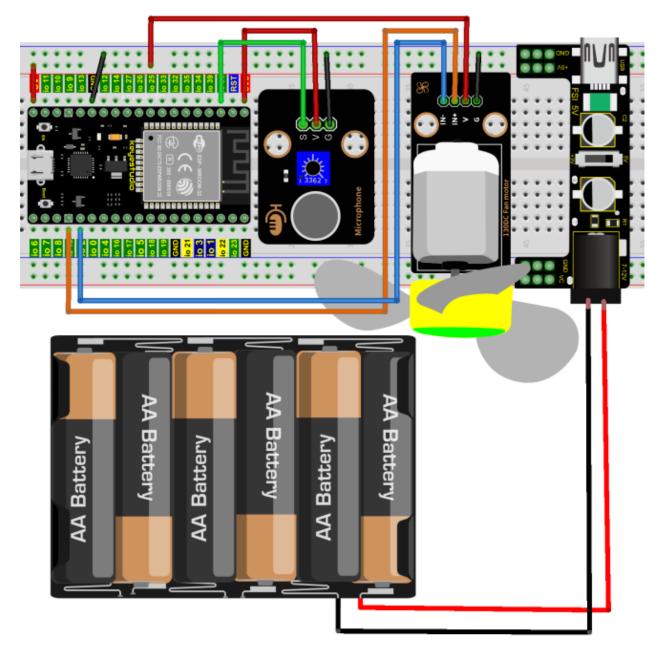
ADC Val: 0 DACVal: 0 Voltage: 0.0 V

ADC Val: 0 DACVal: 0 Voltage: 0.0 V

ADC Val: 0 DACVal: 39 Voltage: 0.5101099 V
```

#### 7.27.5 5. Wiring diagram of the intelligent fan

Next, we officially entered the project. We used a sound sensor, a motor module and a fan blade to simulate a voicecontrolled fan. The wiring diagram is as follows



(Note: Connect the wires and then install a small fan blade on the DC motor. )

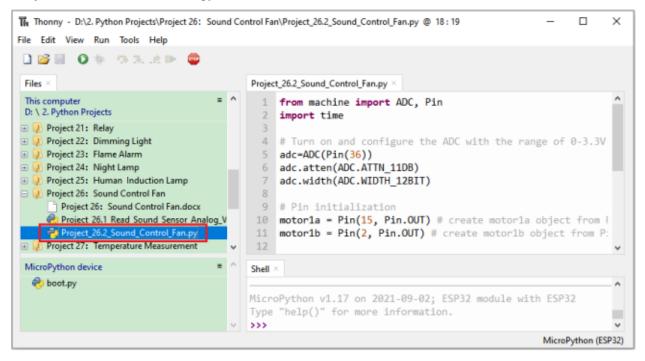
#### 7.27.6 6.Project code

Note if value >600: The threshold 600 in the code can be reset itself as needed)

Codes used in this tutorial are saved in "**2**. **Python Projects**". If you haven't downloaded the code file, please click on the link to download it:Download Python Codes

2. Python Projects		_	$\Box$ $\times$
Share View			~ 🕐
→ This PC → Software (D:) → 2. Pyth	on Projects 🗸 🗸 🗸	ට 🔎 Search	2. Python
Name	Date modified	Туре	Size ^
Project 00: Boot	2/17/2022 10:21 AM	File folder	
	2/17/2022 10:21 AM	File folder	
	2/17/2022 11:10 AM	File folder	
Project 03: LED Flashing	2/17/2022 11:12 AM	File folder	
Project 04: Breathing LED	2/17/2022 10:21 AM	File folder	

Open"Thonny", click "This computer"  $\rightarrow$  "D:"  $\rightarrow$  "2. Python Projects"  $\rightarrow$  "Project 26Sound Control Fan" and then click "Project\_26.2\_Sound\_Control\_Fan.py".



from machine import ADC, Pin
import time
## Turn on and configure the ADC with the range of 0-3.3V
adc=ADC(Pin(36))
adc.atten(ADC.ATTN\_11DB)

(continues on next page)

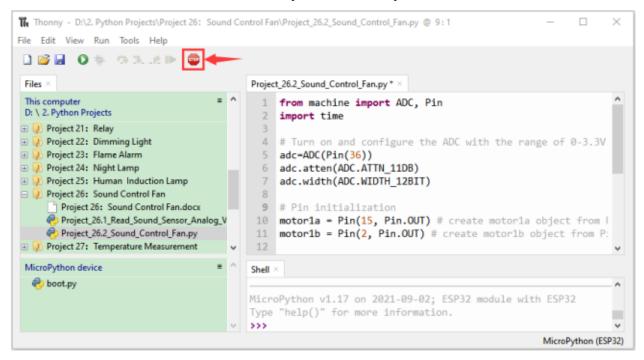
(continued from previous page)

```
adc.width(ADC.WIDTH_12BIT)
```

```
## Pin initialization
motor1a = Pin(15, Pin.OUT) # create motor1a object from Pin 15, Set Pin 15 to output
motor1b = Pin(2, Pin.OUT) # create motor1b object from Pin 2, Set Pin 2 to output
## If the Sound sensor detects Sounds, and the motor will rotate
## when the analog value is greater than 600,0therwise, the motor does not rotate.
while True:
   adcVal=adc.read()
   print(adcVal)
   time.sleep(0.5)
   if adcVal >600:
        motor1a.value(1) # Set motor1a high
        motor1b.value(0) # Set motor1b low
        time.sleep(5) # delay time
   else:
        motor1a.value(0)
       motor1b.value(0)
```

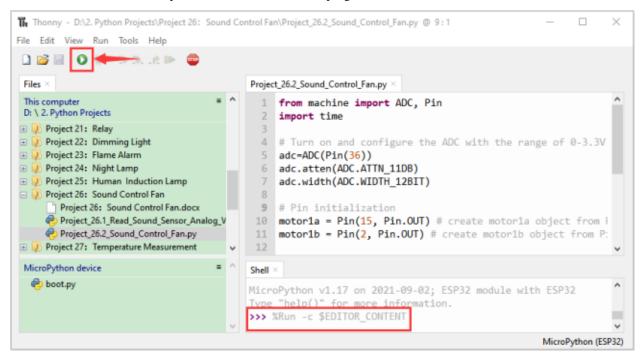
### 7.27.7 7.Project result

Make sure the ESP32 has been connected to the computer, click "" "Stop/Restart backend".



External power supply and power on.

Click "Run current script", the code starts to be executed and you'll see that clap your hands to the sound sensor, and when the sound intensity exceeds a threshold, the small fan rotates; conversely, the small fan doesn't rotate.



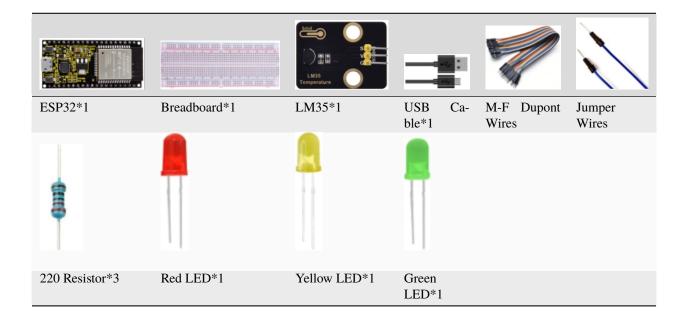
#### Press"Ctrl+C"or click<sup>999</sup>"Stop/Restart backend"to exit the program.

# 7.28 Project 27Temperature Measurement

#### 7.28.1 1.Introduction

LM35 is a common used and easy-to-use temperature sensor. It doesn't require any other hardware and you only need an analog port. The difficulty lies in compiling the code and converting the analog values to Celsius temperature. In this project, we used a temperature sensor and 3 LEDs to make a temperature tester. When the temperature sensor touches different temperature objects, the LEDs will show different colors.

## 7.28.2 2.Components



#### 7.28.3 3.Component knowledge



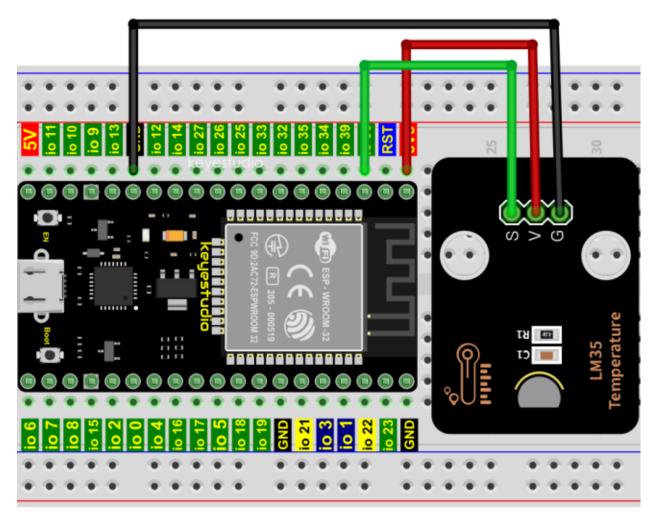
#### Working principle of LM35 temperature sensor:

LM35 temperature sensor is a widely used temperature sensor with a variety of package types. At room temperature, it can achieve the accuracy of  $1/4^{\circ}$ C without additional calibration processing. LM35 temperature sensor can produce different voltage according to different temperatures, when the temperature is 0 °C, it output 0V; If increasing 1 °C, the output voltage will increase 10mv. The output temperature is 0°C to 100°C, the conversion formula is as follows

$$V_{\text{out\_LM35}}(T) = 10 \,\text{mV}/_{\odot} \times T^{\circ}\text{C}$$

#### 7.28.4 4.Read the temperature value of the LM35

We first use a simple code to read the value of the temperature sensor and printing them out, wiring diagram is shown below

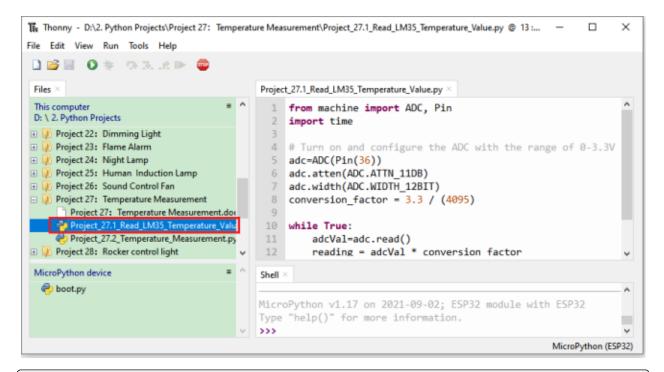


LM35 output is given to analog pin GPIO36 of the ESP32, this analog voltage is converted to its digital form and processed to get the temperature reading.

Codes used in this tutorial are saved in "2. Python Projects". If you haven't downloaded the code file, please click on the link to download it:Download Python Codes

2. Python Projects			$\square$ ×
Share View			~ ?
→ This PC → Software (D:) → 2. Pytho	n Projects 🗸 진	,	Python
Name	Date modified	Туре	Size ^
Project 00: Boot	2/17/2022 10:21 AM	File folder	
	2/17/2022 10:21 AM	File folder	
Project 02: Turn On LED	2/17/2022 11:10 AM	File folder	
Project 03: LED Flashing	2/17/2022 11:12 AM	File folder	
Project 04: Breathing LED	2/17/2022 10:21 AM	File folder	

Open "Thonny" click "This computer"  $\rightarrow$  "D:"  $\rightarrow$  "2. Python Projects"  $\rightarrow$  "Project 27Temperature Measurement", and then double left-click "Project\_27.1\_Read\_LM35\_Temperature\_Value.py".



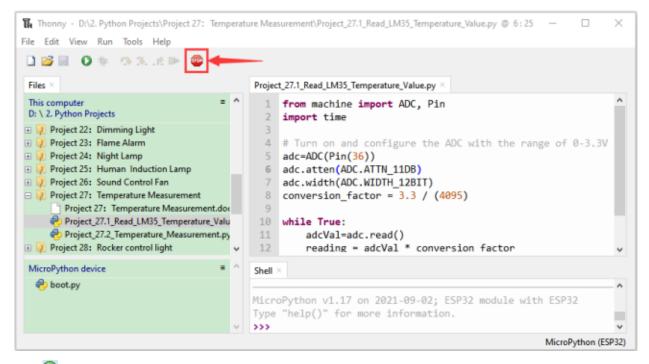
```
from machine import ADC, Pin
import time

## Turn on and configure the ADC with the range of 0-3.3V
adc=ADC(Pin(36))
adc.atten(ADC.ATTN_11DB)
adc.width(ADC.WIDTH_12BIT)
conversion_factor = 3.3 / (4095)

while True:
    adcVal=adc.read()
    reading = adcVal * conversion_factor
    temperature = reading * 102.4
    print(temperature)
```

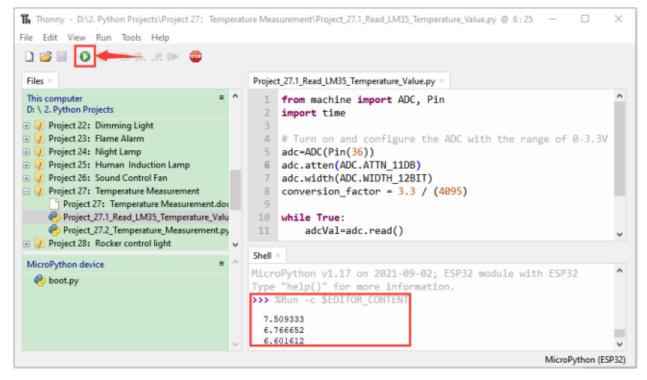
Make sure the ESP32 has been connected to the computer, click "Stop/Restart backend".

time.sleep(1)



Click "Run current script", the code starts to be executed and you'll see that the "Shell" window of Thonny IDE will print the temperature values read by the LM35 temperature sensor.

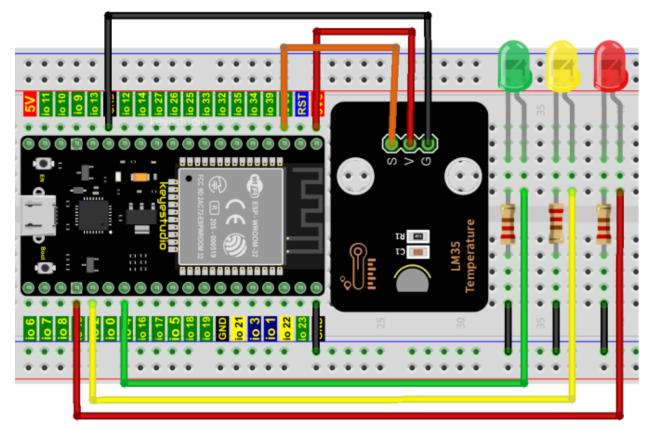
Hold the LM35 element by hand, the temperature value read by the LM35 temperature sensor will change. Press "Ctrl+C" or click "Stop/Restart backend" to exit the program.



Shell $ imes$	
7.591854	^
7.179253	
7.344293	
8.334535	
9.407297	
10.56258	
11.0577	
13.1207	
13.03818	
14.11095	
14.52355	
14.93615	
15.34875	
16.00891	
15.76135	
16.66907	~

### 7.28.5 5.Diagram of the temperature measurement

Now we use a LM35 temperature sensor and three LED lights to do a temperature test. When the LM35 temperature sensor senses different temperatures, different LED lights will light up. Follow the diagram below for wiring.



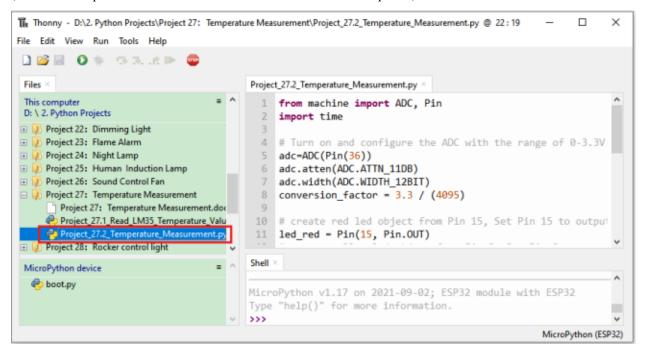
#### 7.28.6 6.Project code

Codes used in this tutorial are saved in "2. Python Projects". If you haven't downloaded the code file, please click on the link to download it:Download Python Codes

2. Python Projects		—	
Share View			~ 🕐
→ This PC → Software (D:) → 2. Pyth	non Projects 🗸 ぐ	,○ Search 2	. Python
Name	Date modified	Туре	Size ^
Project 00: Boot	2/17/2022 10:21 AM	File folder	
Project 01: Hello World	2/17/2022 10:21 AM	File folder	
	2/17/2022 11:10 AM	File folder	
Project 03: LED Flashing	2/17/2022 11:12 AM	File folder	
Project 04: Breathing LED	2/17/2022 10:21 AM	File folder	

Open"Thonny" click"This computer"  $\rightarrow$  "D:"  $\rightarrow$  "2. Python Projects"  $\rightarrow$  "Project 27Temperature Measurement" and then double left-click "Project\_27.2\_Temperature\_Measurement.py".

(Note: The temperature threshold in the code can be reset itself as required.)



```
from machine import ADC, Pin
import time
```

```
## Turn on and configure the ADC with the range of 0-3.3V
adc=ADC(Pin(36))
adc.atten(ADC.ATTN_11DB)
adc.width(ADC.WIDTH_12BIT)
conversion_factor = 3.3 / (4095)
```

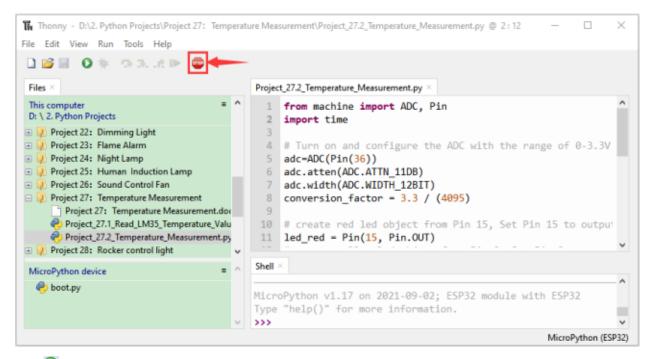
(continues on next page)

(continued from previous page)

```
## create red led object from Pin 15, Set Pin 15 to output
led_red = Pin(15, Pin.OUT)
## create yellow led object from Pin 2, Set Pin 2 to output
led_yellow = Pin(2, Pin.OUT)
## create green led object from Pin 4, Set Pin 4 to output
led_green = Pin(4, Pin.OUT)
while True:
   adcVal=adc.read()
   reading = adcVal * conversion_factor
   temperature = reading * 102.4
   print(temperature)
   time.sleep((0.2))
   if temperature <20:
       led_red.value(1) # Set red led turn on
        led_yellow.value(0) # Set yellow led turn off
       led_green.value(0) # Set green led turn off
   elif temperature >=20 and temperature <25:</pre>
       led_red.value(0) # Set red led turn off
       led_yellow.value(1) # Set yellow led turn on
       led_green.value(0) # Set green led turn off
   else:
       led_red.value(0) # Set red led turn off
       led_yellow.value(0) # Set yellow led turn off
       led_green.value(1) # Set green led turn on
```

## 7.28.7 7.Project result

Make sure the ESP32 has been connected to the computer, click "" "Stop/Restart backend".



Click "Run current script", the code starts to be executed and you'll see that the "Shell" window of Thonny IDE will print the temperature values read by the LM35 temperature sensor.

When the LM35 temperature sensor senses different temperatures, different LEDS will light up. Press"Ctrl+C"or click "Stop/Restart backend"to exit the program.

🏗 Thonny - D:\2. Python Projects\Project 27: Temperature Measurement\Project_27.2_Temperature_Measurement.py @ 2:12 - 🛛 🗙					
File Edit View Run Tools Help					
🗋 🗃 🖬 💽 🛑 🗮 👘 🖤					
Files ×		Project_27.2_Temperature_Measurement.py $\times$			
This computer = D: \ 2. Python Projects	^	<pre>1 from machine import ADC, Pin 2 import time</pre>	^		
<ul> <li></li></ul>		<pre>3 4 # Turn on and configure the ADC with the range of 0-3.3</pre>	v		
<ul> <li></li></ul>		<pre>5 adc=ADC(Pin(36)) 6 adc.atten(ADC.ATTN_11DB) </pre>			
Project 26: Sound Control Fan     Project 27: Temperature Measurement     Project 27: Temperature Measurement do		<pre>7 adc.width(ADC.WIDTH_12BIT) 8 conversion_factor = 3.3 / (4095) 9</pre>			
Project 27: Temperature Measurement.do Project_27.1_Read_LM35_Temperature_Value Project_27.2_Temperature_Measurement.project_27.2_Temperature_Measure_Maasure_Maasure_Meas		<pre>9 10 # create red led object from Pin 15, Set Pin 15 to outp</pre>	ut 🗸		
⊕ Project 28: Rocker control light	~	Shell × MicroPython v1.17 on 2021-09-02; ESP32 module with ESP32	^		
MicroPython device =	^	Type "help()" for more information.			
🕏 boot.py	~	>>> %Run -c \$EDITOR_CONTENT 13.1207 14.85363 15.67883	*		
		MicroPython (	ESP32)		

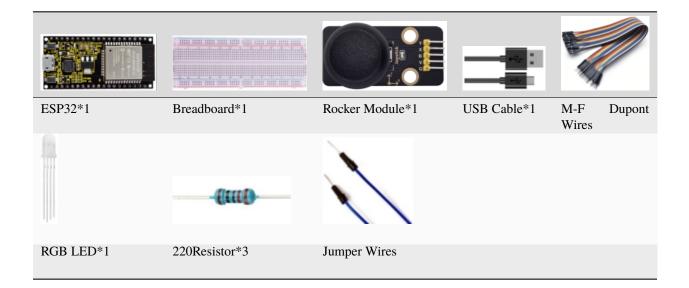
# 7.29 Project 28Rocker control light

#### 7.29.1 1.Introduction

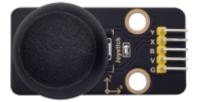
The rocker module is a component with two analog inputs and one digital input. It is widely used in areas such as game operation, robot control and drone control.

In this project, we use ESP32 and a joystick module to control RGB, so that you can have a deeper understanding of the principle and operation of the joystick module in practice.

### 7.29.2 2.Components



#### 7.29.3 3.Component knowledge



#### **Rocker module:**

It mainly uses PS2 joystick components. In fact, the joystick module has 3 signal terminal pins, which simulate a three-dimensional space.

The pins of the joystick module are GND, VCC, and signal terminals (B, X, Y). The signal terminals X and Y simulate the X-axis and Y-axis of the space. When controlling, the X and Y signal terminals of the module are connected to the analog port of the microcontroller. The signal terminal B simulates the Z axis of the space, it is generally connected to the digital port and used as a button.

VCC is connected to the microcontroller power output VCC (3.3V or 5V), GND is connected to the microcontroller GND, the voltage in the original state is about 1.65V or 2.5V.

In the X-axis direction, when moving in the direction of the arrow, the voltage value increases, and the maximum voltage can be reached. Moving in the opposite direction of the arrow, the voltage value gradually decreases to the minimum voltage.

In the Y-axis direction, the voltage value decreases gradually as it moves in the direction of the arrow on the module, decreasing to the minimum voltage. As the arrow is moved in the opposite direction, the voltage value increases and can reach the maximum voltage.

In the Z-axis direction, the signal terminal B is connected to the digital port and outputs 0 in the original state and outputs 1 when pressed.

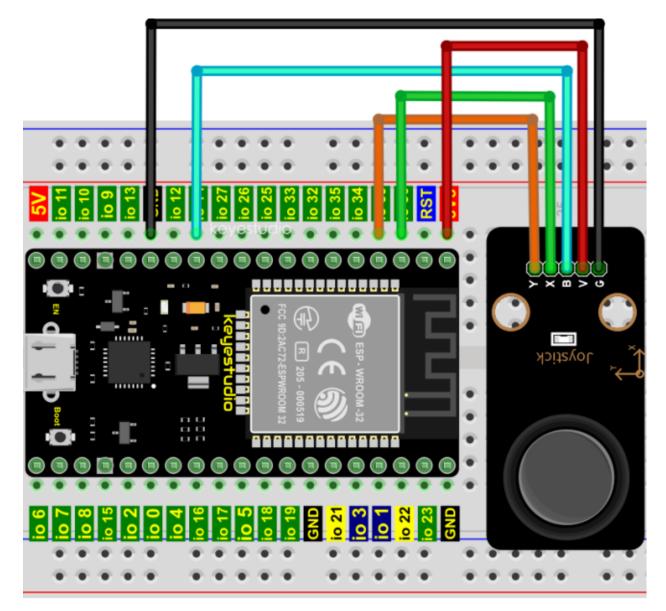
In this way, we can read the two analog values and the high and low level conditions of the digital port to determine the operating status of the joystick on the module.

#### Features:

- Input Voltage: DC 3.3V ~ 5V.
- Output Signal: X/Y dual axis analog value +Z axis digital signal.
- Range of Allication: Suitable for control point coordinate movement in plane as well as control of two degrees of freedom steering gear, etc.
- Product features: Exquisite appearance, joystick feel superior, simple operation, sensitive response, long service life.

#### 7.29.4 4.Read the value of the Rocker Module

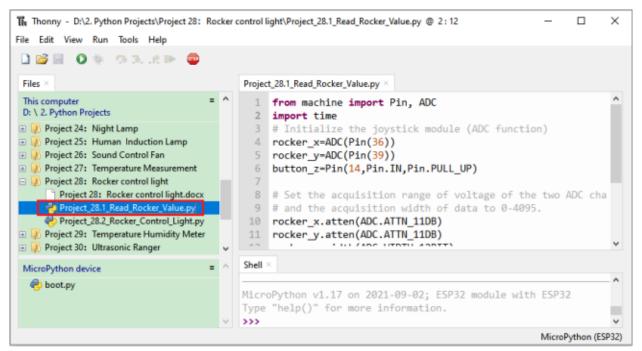
We must use ESP32's analog IO port to read the value from the X/Y pin of the rocker module and use the digital IO port to read the digital signal of the button. Please connect the wires according to the wiring diagram below



Codes used in this tutorial are saved in "2. Python Projects". If you haven't downloaded the code file, please click on the link to download it:Download Python Codes

→ This PC → Software (D:) → 2. Pytł	non Proiects 🗸 🗸	o o	Search 2. I	Python
^				-
Name	Date modified	Туре		Size
Project 00: Boot	2/17/2022 10:21 AM	√l File fo	older	
Project 01: Hello World	2/17/2022 10:21 AM	√l File fo	older	
Project 02: Turn On LED	2/17/2022 11:10 AM	√l File fo	older	
Project 03: LED Flashing	2/17/2022 11:12 AM	√l File fo	older	
Project 04: Breathing LED	2/17/2022 10:21 AM	∧ File fo	older	

Open"Thonny" click"This computer"  $\rightarrow$  "D:"  $\rightarrow$  "2. Python Projects"  $\rightarrow$  "Project 28Rocker control light", and then double left-click "Project\_28.1\_Read\_Rocker\_Value.py".



```
from machine import Pin, ADC
import time
## Initialize the joystick module (ADC function)
rocker_x=ADC(Pin(36))
rocker_y=ADC(Pin(39))
button_z=Pin(14,Pin.IN,Pin.PULL_UP)
## Set the acquisition range of voltage of the two ADC channels to 0-3.3V,
## and the acquisition width of data to 0-4095.
rocker_x.atten(ADC.ATTN_11DB)
rocker_y.atten(ADC.ATTN_11DB)
rocker_x.width(ADC.WIDTH_12BIT)
rocker_y.width(ADC.WIDTH_12BIT)
## In the code, configure Z_Pin to pull-up input mode.
## In loop(), use Read () to read the value of axes X and Y
## and use value() to read the value of axis Z, and then display them.
while True:
   print("X,Y,Z:",rocker_x.read(),",",rocker_y.read(),",",button_z.value())
    time.sleep(0.5)
```

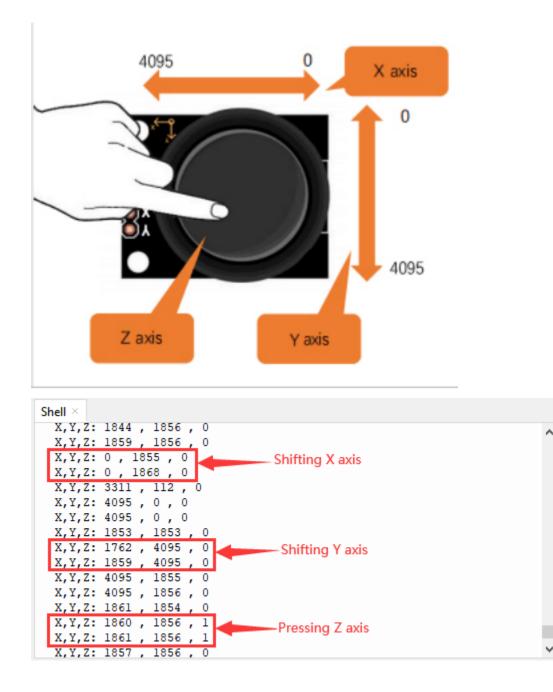
Make sure the ESP32 has been connected to the computer, click "Stop/Restart backend".

<pre> Project_28.1_Read_Rocker_Value.py Project_28.2_Rocker_Control_Light.py  Project 29: Temperature Humidity Meter  # and the acquisition width of data to 0-4095.  rocker_x.atten(ADC.ATTN_11DB) rocker_y.atten(ADC.ATTN_11DB) </pre>	2       import time         3       # Initialize the joystick module (ADC function)         an Induction Lamp       4         d Control Fan       5         berature Measurement       6         button_z=Pin(14,Pin.IN,Pin.PULL_UP)         r         ocker control light.docx         Read_Rocker_Value.py         9       # and the acquisition width of data to 0-4095.
Project 30: Ultrasonic Ranger	perature Humidity Meter 11 rocker_y.atten(ADC.ATTN_11DB)
MicroPython device = A Shell ×	

Click Circle Click

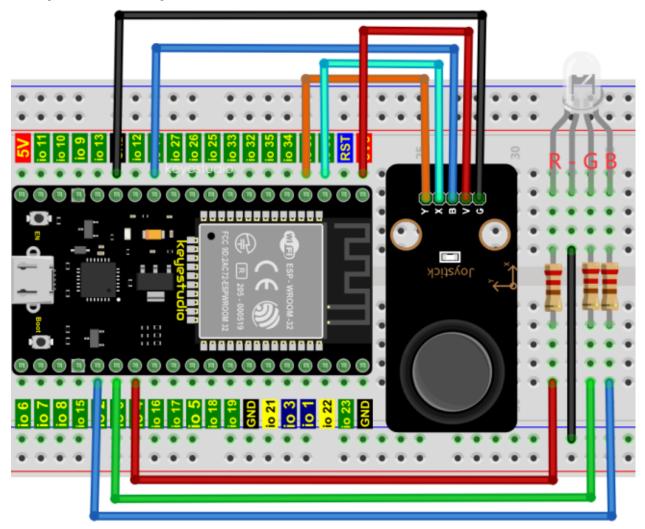
Moving the joystick or pressing it will change the analog and digital values in "Shell". Press"Ctrl+C"or click "Stop/Restart backend" to exit the program.

Thonny - D:\2. Python Projects\Project 28:	Rocke	control light\Project_28.1_Read_Rocker_Value.py @ 20:19 —	×
File Edit View Run Tools Help			
🗋 🗃 🔲 💽 🛑 🖘 3e 🕨 👳			
Files $\times$		Project_28.1_Read_Rocker_Value.py ×	
This computer D: \ 2. Python Projects	= ^	1 from machine import Pin, ADC 2 import time	î
Project 24: Night Lamp     Project 25: Human Induction Lamp     Project 25: Sound Control Fan     Project 26: Sound Control Fan     Project 27: Temperature Measurement     Project 28: Rocker control light     Project 28: Rocker control light.docx     Project 28: Rocker_Control Light.py     Project 29: Temperature Humidity Meter     Project 30: Ultrasonic Ranger		<pre>3 # Initialize the joystick module (ADC function) 4 rocker_x=ADC(Pin(36)) 5 rocker_y=ADC(Pin(39)) 6 button_z=Pin(14,Pin.IN,Pin.PULL_UP) 7 8 # Set the acquisition range of voltage of the two ADC ch 9 # and the acquisition width of data to 0-4095. 10 rocker_x.atten(ADC.ATTN_11DB) 11 rocker_y.atten(ADC.ATTN_11DB)</pre>	ha
IProject 30: Ultrasonic Ranger  MicroPython device	= ^	Shell ×	
e boot.py		MicroPython v1.17 on 2021-09-02; ESP32 module with ESP32 Type "help()" for more information. >>> %Run -c \$EDITOR_CONTENT X,Y,Z: 1844, 1856, 0 X,Y,Z: 1859, 1856, 0 X,Y,Z: 0, 1855, 0	^
	~	V V Z+ 0 1969 0 MicroPython (	ESP32



## 7.29.5 5.Wiring diagram

We just read the value of the rocker module, we need to do something with the rocker module and RGB here, Follow the diagram below for wiring

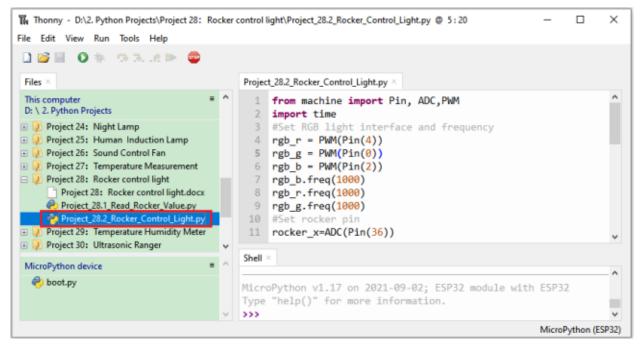


#### 7.29.6 6.Project code

Codes used in this tutorial are saved in "2. Python Projects". If you haven't downloaded the code file, please click on the link to download it:Download Python Codes

2. Python Projects		_	
Share View			~ ?
→ This PC → Software (D:) → 2. Pyth	on Projects 🗸 진	, P Search 2	. Python
Name	Date modified	Туре	Size ^
Project 00: Boot	2/17/2022 10:21 AM	File folder	
Project 01: Hello World	2/17/2022 10:21 AM	File folder	
Project 02: Turn On LED	2/17/2022 11:10 AM	File folder	
Project 03: LED Flashing	2/17/2022 11:12 AM	File folder	
Project 04: Breathing LED	2/17/2022 10:21 AM	File folder	

 $Open "Thonny" click "This computer" \rightarrow "D:" \rightarrow "2. Python Projects" \rightarrow "Project 28 Rocker control light" and then double left-click "Project_28.2_Rocker_Control_Light.py".$ 



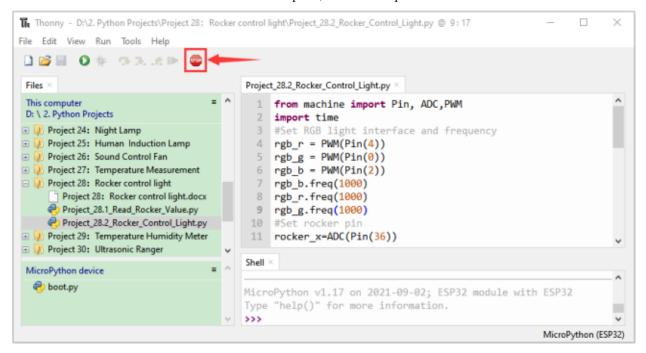
```
from machine import Pin, ADC,PWM
import time
#Set RGB light interface and frequency
rgb_r = PWM(Pin(4))
rgb_g = PWM(Pin(0))
rgb_b = PWM(Pin(2))
rgb_b.freq(1000)
rgb_r.freq(1000)
rgb_g.freq(1000)
#Set rocker pin
rocker_x=ADC(Pin(36))
rocker_y=ADC(Pin(36))
## Set the acquisition range of voltage of the two ADC channels to 0-3.3V,
## and the acquisition width of data to 0-4095.
```

(continued from previous page)

```
rocker_x.atten(ADC.ATTN_11DB)
rocker_y.atten(ADC.ATTN_11DB)
rocker_x.width(ADC.WIDTH_12BIT)
rocker_y.width(ADC.WIDTH_12BIT)
while True:
   y = rocker_y.read()#Get Y value of rocker
   x = rocker_x.read()#Get X value of rocker
   if x < 1000:
                    #left
        rgb_b.duty())
        rgb_r.duty(1023)
        rgb_g.duty())
   elif x > 3000:
                      #right
        rgb_b.duty())
        rgb_r.duty(≬)
        rgb_g_duty(1023)
   elif y < 1000:
                      #down
        rgb_b.duty(1023)
        rgb_r.duty())
        rgb_g.duty())
    elif y > 3000:
                      #up
        rgb_b.duty(1023)
        rgb_r.duty(1023)
        rgb_g.duty(1023)
    time.sleep(0.01)
```

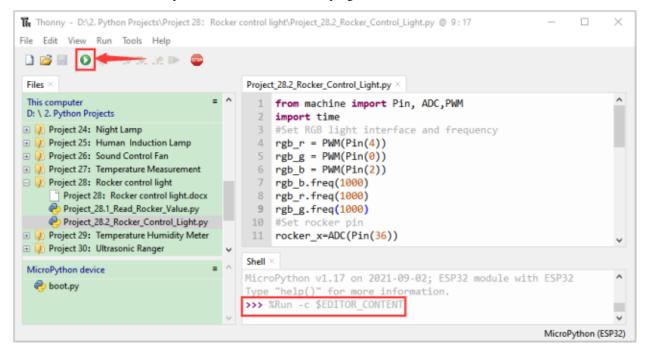
## 7.29.7 7.Project result

Make sure the ESP32 has been connected to the computer, click "Stop/Restart backend".



Click Circle Click

Press"Ctrl+C"or click <sup>22</sup> "Stop/Restart backend" to exit the program.







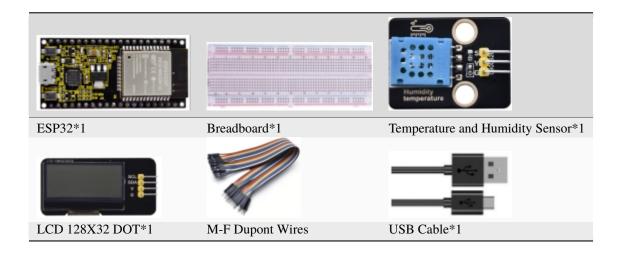


# 7.30 Project 29Temperature Humidity Meter

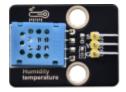
#### 7.30.1 1.Introduction

In winter, the humidity in the air is very low, that is, the air is very dry, Coupled with cold, the skin of the human body is easy to be too dry and cracked, so you need to use a humidifier to increase the humidity of the air at home, but how do you know that the air is too dry? Then you need equipment to detect air humidity. In this Project, we will how to use the temperature and humidity sensor. We use the sensor to make a thermohygrometer, and also combined with a LCD 128X32 DOT to display the temperature and humidity values.

#### 7.30.2 2.Components



#### 7.30.3 3.Component knowledge



#### Temperature and humidity sensor:

It is a temperature and humidity composite sensor with calibrated digital signal output, its precision humidity is $\pm 5\%$ RH, temperature is $\pm 2$ °C, range humidity is 20 to 90%RH, and temperature is 0 to 50°C. The temperature and humidity sensor applies dedicated digital module acquisition technology and temperature and humidity sensing technology to ensure extremely high reliability and excellent long-term stability of the product.

The temperature and humidity sensor includes a resistive-type humidity measurement and an NTC temperature measurement component, which is very suitable for temperature and humidity measurement applications where accuracy and real-time performance are not required. The operating voltage is in the range of 3.3V to 5.5V.

The temperature and humidity sensor has three pins, which are VCCGND and S. S is the pin for data output, using serial communication.

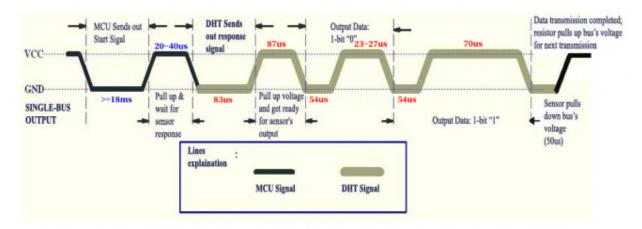
#### Single bus format definition of Temperature and Humidity Sensor

De- scrip- tion	Definition
Start signal	Microprocessor pulls data bus (SDA) down at least 18ms for a period of time(Maximum is 30ms), noti- fying the sensor to prepare data.
Re- sponse signal	The sensor pulls the data bus (SDA) low for $83\mu$ s, and then pulls up for $87\mu$ s to respond to the host's start signal.
Hu- midity	The high humidity is an integer part of the humidity data, and the low humidity is a fractional part of the humidity data.
Tem- pera- ture	The high temperature is the integer part of the temperature data, the low temperature is the fractional part of the temperature data. And the low temperature Bit8 is 1, indicating a negative temperature, otherwise, it is a positive temperature.
Parity bit	Parity bit=Humidity high bit+ Humidity low bit+temperature high bit+temperature low bit

#### Data sequence diagram of Temperature and Humidity Sensor

When MCU sends a start signal, the Temperature and Humidity Sensor changes from the low-power-consumption mode to the high-speed mode, waiting for MCU completing the start signal. Once it is completed, the Temperature and Humidity Sensor sends a response signal of 40-bit data and triggers a signal acquisition.

The signal is sent as shown in the figure:



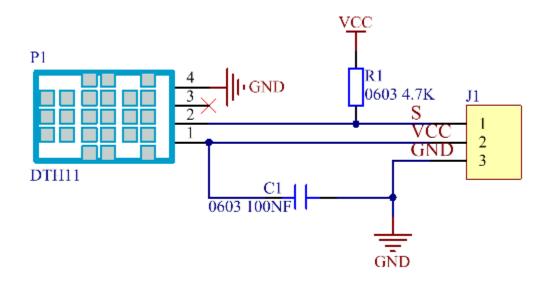
Combined with the code, you can understand better.

The XHT11 temperature and humidity sensor can easily add temperature and humidity data to your DIY electronic projects. It is perfect for remote weather stations, home environmental control systems, and farm or garden monitoring systems.

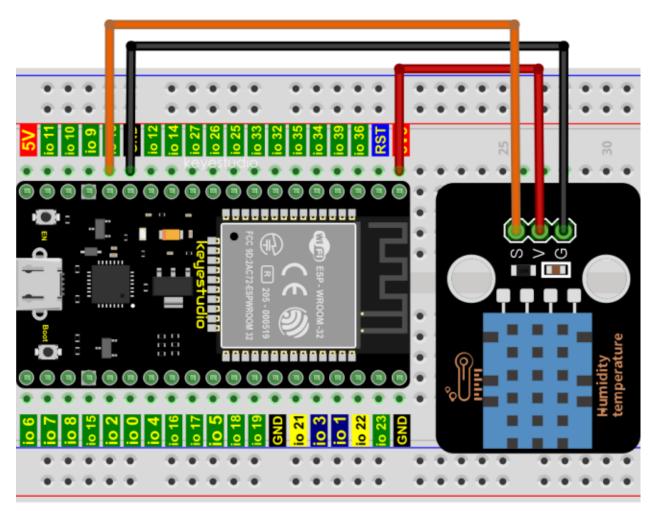
#### Specification:

- Working voltage: +5V
- Temperature range: 0°C to 50°C, error of  $\pm$  2°C
- Humidity range: 20% to 90% RH, $\pm$  5% RH error
- Interface: Digital interface

#### Schematic diagram of Temperature and Humidity Sensor:



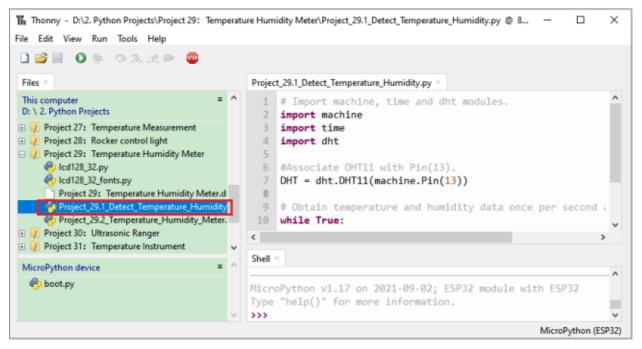
7.30.4 4.Read temperature and humidity value

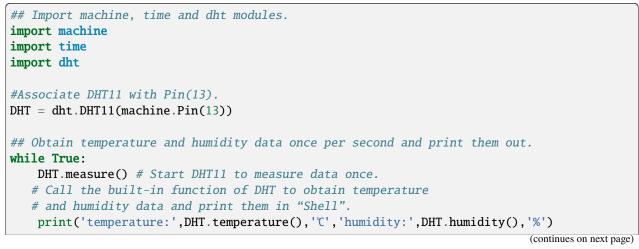


Codes used in this tutorial are saved in "2. Python Projects". If you haven't downloaded the code file, please click on the link to download it:Download Python Codes

2. Python Projects		_	
Share View			~ 🕐
→ This PC → Software (D:) → 2. Pyth	on Projects 🗸 さ		. Python
Name	Date modified	Туре	Size ^
Project 00: Boot	2/17/2022 10:21 AM	File folder	
	2/17/2022 10:21 AM	File folder	
	2/17/2022 11:10 AM	File folder	
Project 03: LED Flashing	2/17/2022 11:12 AM	File folder	
Project 04: Breathing LED	2/17/2022 10:21 AM	File folder	

Open"Thonny" click"This computer"  $\rightarrow$  "D:"  $\rightarrow$  "2. Python Projects"  $\rightarrow$  "Project 29Temperature Humidity Meter", and then double left-click "Project\_29.1\_Detect\_Temperature\_Humidity.py".

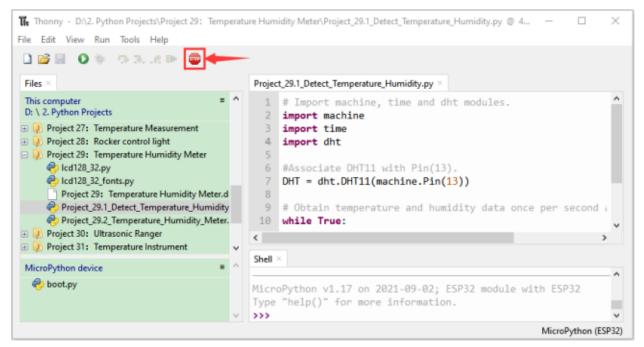




(continued from previous page)

time.sleep\_ms(1000)

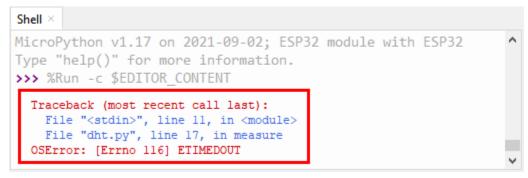
Make sure the ESP32 has been connected to the computer, click "Stop/Restart backend".

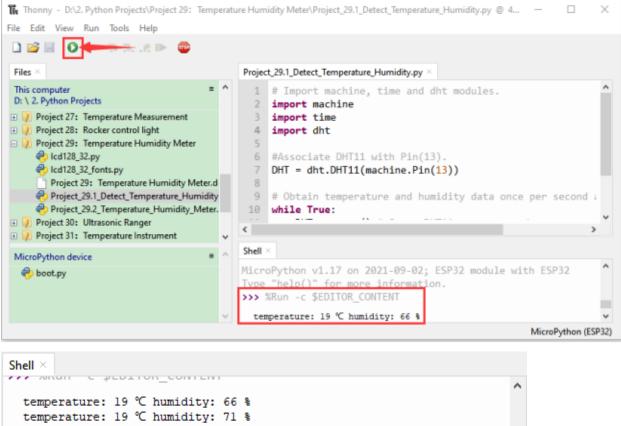


Click Circle Click Circle Connected, the following information is displayed in the "Shell" window. Please make sure your circuit is properly connected.

Click Qagain, the code starts to be executed and you'll see that the "Shell" window of Thonny IDE will print the temperature and humidity datas in the current surroundings, as shown in the following figure.

Press"Ctrl+C"or click <sup>22</sup> "Stop/Restart backend" to exit the program.

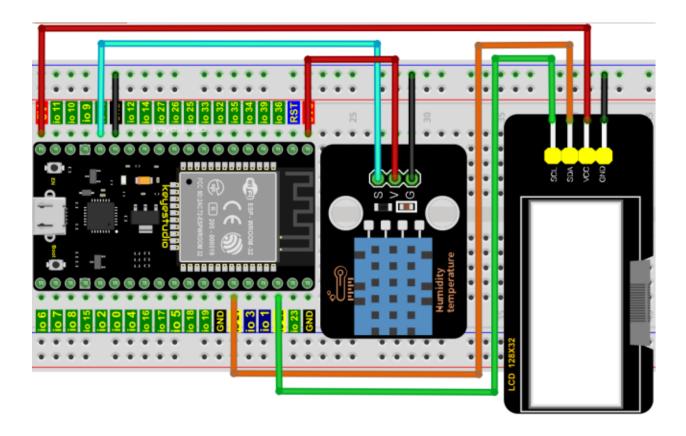




cemperature.	19	~	numitarcy.	00	•
temperature:	19	°C	humidity:	71	ŝ
temperature:	19	°C	humidity:	72	Ŷ
temperature:	19	°C	humidity:	73	ş
temperature:	19	°C	humidity:	74	ş
temperature:	19	°C	humidity:	93	ş
temperature:	20	°C	humidity:	93	ş
temperature:	21	°C	humidity:	94	ş
temperature:	23	°C	humidity:	95	ş
temperature:	24	°C	humidity:	95	ş
temperature:	25	°C	humidity:	95	ŝ
temperature:	25	°C	humidity:	95	ŝ
temperature:	27	°C	humidity:	95	\$
temperature:	27	°C	humidity:	93	00
temperature:	28	°C	humidity:	95	elo

## 7.30.5 5. Wiring diagram of the thermohygrometer

Now we start to print the values of the temperature and humidity sensor with LCD\_128X32\_DOT. We will see the corresponding values on the screen of LCD\_128X32\_DOT. Let's get started with this project. Please connect cables according to the following wiring diagram



#### 7.30.6 6.Project code

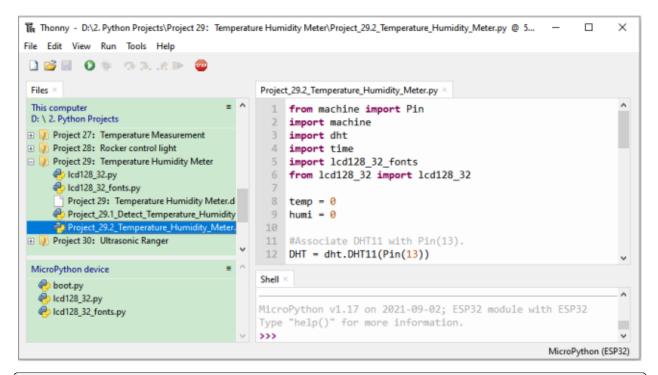
Codes used in this tutorial are saved in "2. Python Projects". If you haven't downloaded the code file, please click on the link to download it:Download Python Codes

2. Python Projects		-	$\square$ ×
Share View			~ ?
→ This PC → Software (D:) → 2. Pytł	non Projects v さ	,○ Search 2.	Python
Name	Date modified	Туре	Size ^
Project 00: Boot	2/17/2022 10:21 AM	File folder	
	2/17/2022 10:21 AM	File folder	
	2/17/2022 11:10 AM	File folder	
Project 03: LED Flashing	2/17/2022 11:12 AM	File folder	
Project 04: Breathing LED	2/17/2022 10:21 AM	File folder	

Open"Thonny"click"This computer" $\rightarrow$ "D:" $\rightarrow$ "2. Python Projects" $\rightarrow$ "Project 29Temperature Humidity Meter". Select"lcd128\_32.py"and"lcd128\_32\_fonts.py"right-click your mouse to select"Upload to /"wait for "lcd128\_32.py" and "lcd128\_32\_fonts.py" to be uploaded to ESP32and double left-click "Project\_29.2\_Temperature\_Humidity\_Meter.py".

Ter Thonny	-		$\times$
File Edit View Run Tools Help			
D B B O W O B. Lt D B B			
This computer = ^ D: \ 2. Python Projects			
•          •          •			
Icd128_32.py       Open in Thonny         Project 29:       Open in system default app         Project 29.1 De       Configure .py files			
Project_29.2_Te Upload to /           Image: Project 30: Ultraso         Move to Recycle Bin			
New directory			
MicroPython device Properties hell ×			
<pre>boot.py MicroPython v1.17 on 2021-09-02; ESP32 module with Type "help()" for more information. &gt;&gt;&gt;</pre>	h ESP3	2	< >
	MicroPy	thon (ES	P32)

Tin Thonny		_		×
File Edit View Run Tools	Help			
🗋 💕 🖩 0 🎄 🔿 3. Files ×	.e ▶ 👄			
This computer D: \ 2. Python Projects	= ^			
Project 27: Temperature N     Project 28: Rocker contro     Project 29: Temperature N     Project 29: Temperature N     Polect 29: Temperature N     Polect 29: Temperature N	l light			
Project 29: Temper Project 29: Temper Project 29.1_Detect	Open in Thonny Open in system default app Configure .py files			
🗉 🕖 Project 30: Ultrasonic R	Upload to / Move to Recycle Bin			
MicroPython device	New directory			
eboot.py	Properties			^
🤲 lcd128_32.py	<pre>MicroPython v1.17 on 2021-09-02; ESP32 module with Type "help()" for more information. &gt;&gt;&gt;</pre>			~
		Micro	Python (E	SP32)



```
from machine import Pin
import machine
import dht
import time
import lcd128_32_fonts
from lcd128_32 import lcd128_32
temp = 
humi = 
#Associate DHT11 with Pin(13).
DHT = dht.DHT11(Pin(13))
#i2c config
clock_pin = 22
data_pin = 21
bus = 0
i2c_addr = 0x3f
use i2c = True
def scan_for_devices():
   i2c = machine.I2C(bus,sda=machine.Pin(data_pin),scl=machine.Pin(clock_pin))
   devices = i2c.scan()
   if devices:
        for d in devices:
            print(hex(d))
   else:
       print('no i2c devices')
try:
   while True:
```

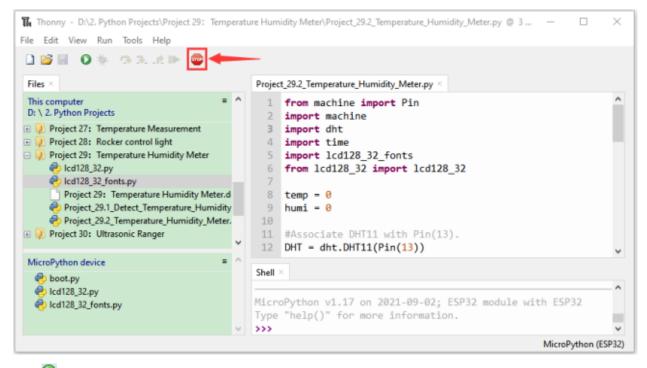
(continues on next page)

(continued from previous page)

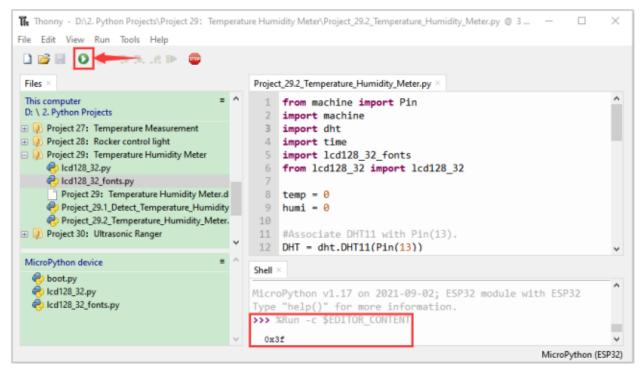
```
DHT.measure()
        temp = int(DHT.temperature())
        humi = int(DHT.humidity())
        if use_i2c:
            scan_for_devices()
            lcd = lcd128_32(data_pin, clock_pin, bus, i2c_addr)
        lcd.Clear()
        lcd.Cursor(∅, ∅)
        lcd.Display("temper:")
        lcd.Cursor(0, 8)
        lcd.Display(str(temp))
        lcd.Cursor(0, 11)
        lcd.Display("C")
        lcd.Cursor(2, ≬)
        lcd.Display("Humid:")
        lcd.Cursor(2, 7)
        lcd.Display(str(humi))
        lcd.Cursor(2, 10)
        lcd.Display("%")
        time.sleep((0.1))
except:
    pass
```

#### 7.30.7 7.Project result

Make sure the ESP32 has been connected to the computer, click "" "Stop/Restart backend".



Click "Run current script", the code starts to be executed and you'll see that the LCD 128X32 DOT will display temperature and humidity value in the current environment.



Press"Ctrl+C"or click ""Stop/Restart backend" to exit the program.

# 7.31 Project 30Ultrasonic Ranger

#### 7.31.1 Introduction

The HC-SR04 ultrasonic sensor is a very affordable distance sensor, mainly used for obstacle avoidance in various robotic projects. It is also used for water level sensing and even as a parking sensor. We treat the ultrasonic sensors as bat's eyes, in the dark, bats can still identify objects in front of them and directions through ultrasound.

In this project, we use ESP32 to control a ultrasonic sensor and LEDs to simulate ultrasonic rangefinder.

# 7.31.2 Components

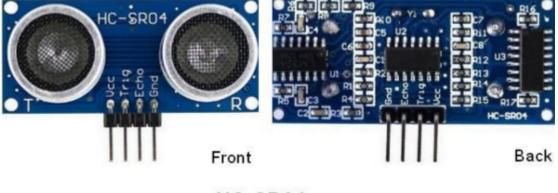
ESP32*1	Breadboard*1	Ultrasonic Sensor*1	Red LED*4
M-F Dupont Wires	220Resistor*4	Jumper Wires	USB Cable*1

## 7.31.3 Component knowledge

#### HC-SR04 Ultrasonic Sensor :

Like bats, sonar is used to determine the distance to an object. It provides accurate non-contact range detection, high-precision and stable readings.

Its operation is not affected by sunlight or black materials, just like a precision camera (acoustically softer materials like cloth are difficult to detect). It has an ultrasonic transmitter and receiver.



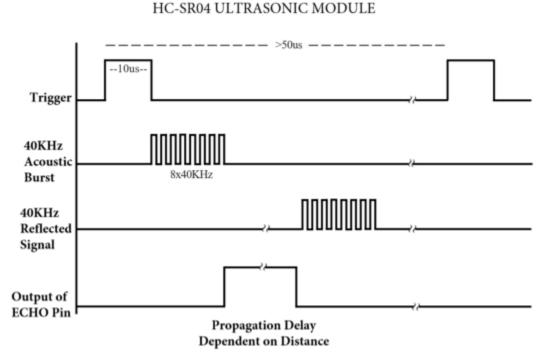
HC-SR04

In front of the ultrasonic sensor are two metal cylinders, these are the converters. The converters convert the mechanical energy into an electrical signal. In the ultrasonic sensor, there are transmitting converters and receiving converters. The transmitting converter converts the electric signal into an ultrasonic pulse, and the receiving converter converts the reflected ultrasonic pulse back to an electric signal. If you look at the back of the ultrasonic sensor, you will see an IC behind the transmitting converter, which controls the transmitting converter.

There is also an IC behind the receiving converter, which is a quad operational amplifier that amplifies the signal generated by the receiving converter into a signal large enough to be transmitted to the Microcontroller.

#### Sequence diagrams:

The figure shows the sequence diagram of the HC-SR04. To start the measurement, the Trig of SR04 must receive at least 10us high pulse (5V), which will activate the sensor to emit 8 cycles of 40kHz ultrasonic pulses, and wait for the reflected ultrasonic pulses. When the sensor detects ultrasound from the receiver, it sets the Echo pin to high (5V) and delays it by one cycle (width), proportional to the distance. To get the distance, measure the width of the Echo pin.



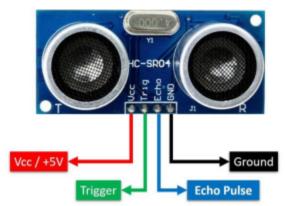
Time = Echo pulse width, its unit is"us" (microseconds)

Distance(cm) = time/58

Distance(inch) = time/148

## 7.31.4 Read the distance value of the ultrasonic sensor:

We will start with a simple ultrasonic ranging and print the measured distance.



The HC-SR04 ultrasonic sensor has four pins, they are Vcc, Trig, Echo and GND.

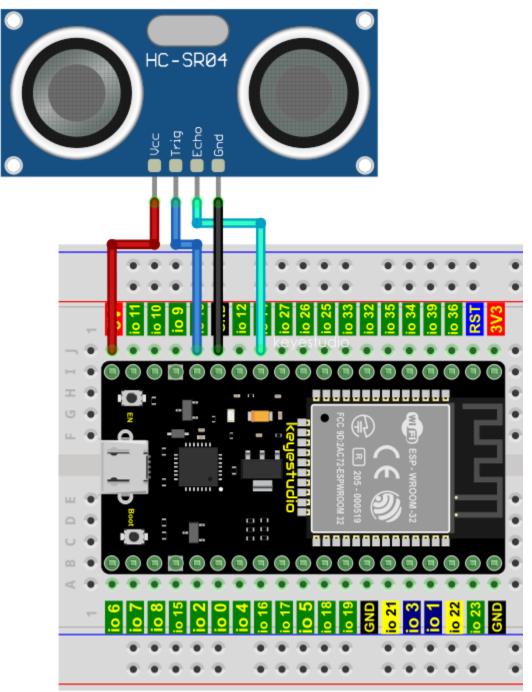
The Vcc pin provides the power source for generating ultrasonic pulses and is connected to Vcc (+5V).

The GND pin is grounded.

The Trig pin is where the Arduino sends a signal to start the ultrasonic pulse.

The Echo pin is where the ultrasonic sensor sends information about the duration of the ultrasonic pulse to the Control board.

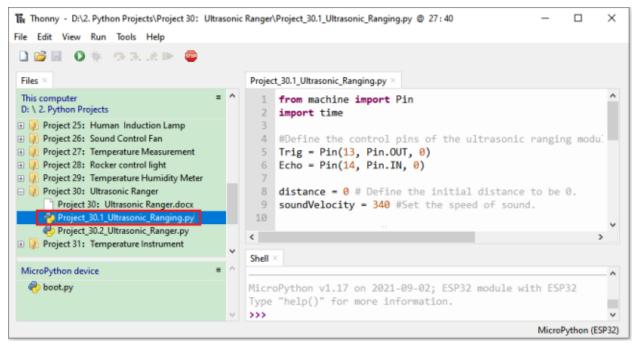
Wiring as shown below:



Codes used in this tutorial are saved in "**2. Python Projects**". If you haven't downloaded the code file, please click on the link to download it:Download Python Codes

2. Python Projects		_	
Share View			~ ?
→ This PC → Software (D:) → 2. Pytho	on Projects 🗸 진		. Python
Name	Date modified	Туре	Size ^
Project 00: Boot	2/17/2022 10:21 AM	File folder	
Project 01: Hello World	2/17/2022 10:21 AM	File folder	
Project 02: Turn On LED	2/17/2022 11:10 AM	File folder	
Project 03: LED Flashing	2/17/2022 11:12 AM	File folder	
Project 04: Breathing LED	2/17/2022 10:21 AM	File folder	

Open"Thonny" click"This computer"  $\rightarrow$  "D:"  $\rightarrow$  "2. Python Projects"  $\rightarrow$  "Project 30Ultrasonic Ranger", and then double left-click"Project 30.1\_Ultrasonic\_Ranging.py".



# from machine import Pin import time

```
## Define the control pins of the ultrasonic ranging module.
Trig = Pin(13, Pin.OUT, 0)
Echo = Pin(14, Pin.IN, 0)
distance = 0 # Define the initial distance to be 0.
soundVelocity = 340 #Set the speed of sound.
## The getDistance() function is used to drive the ultrasonic module to measure distance,
## the Trig pin keeps at high level for 10us to start the ultrasonic module.
## Echo.value() is used to read the status of ultrasonic module's Echo pin,
## and then use timestamp function of the time module to calculate the duration of Echo
(continues on next page)
```

(continued from previous page)

```
## pin's high level,calculate the measured distance based on time and return the value.
def getDistance():
   Trig.value(1)
   time.sleep_us(10)
   Trig.value(♥)
   while not Echo.value():
        pass
   pingStart = time.ticks_us()
   while Echo.value():
        pass
   pingStop = time.ticks_us()
   pingTime = time.ticks_diff(pingStop, pingStart) // 2
   distance = int(soundVelocity * pingTime // 10000)
   return distance
## Delay for 2 seconds and wait for the ultrasonic module to stabilize,
## Print data obtained from ultrasonic module every 500 milliseconds.
time.sleep(2)
while True:
   time.sleep_ms(500)
   distance = getDistance()
   print("Distance: ", distance, "cm")
```

Make sure the ESP32 has been connected to the computer, click <sup>999</sup> "Stop/Restart backend".

🗋 🗃 📓 🔕 🌸 🗇 3e 🕨 🤤	-	-			
Files ×			Proje	t_30.1_Ultrasonic_Ranging.py ×	
This computer D: \ 2. Python Projects	-	^	1	<pre>from machine import Pin import time</pre>	í
🛛 💫 Project 25: Human Induction Lamp			3		
Project 26: Sound Control Fan			4	aperate ene concrea parts or ene eactopenace renDariD mon	u:
Project 27: Temperature Measurement			5	Trig = $Pin(13, Pin.OUT, 0)$	
Project 28: Rocker control light			6	Echo = Pin(14, Pin.IN, 0)	
Project 29: Temperature Humidity Meter			7	Materia All Define the Initial distance to be 0	
Project 30: Ultrasonic Ranger Project 30: Ultrasonic Ranger.docx			8	distance = 0 # Define the initial distance to be 0.	
Project 30.1 Ultrasonic Ranging.py			10	<pre>soundVelocity = 340 #Set the speed of sound.</pre>	
Project_30.2_Ultrasonic_Ranger.py			10		
Project 31: Temperature Instrument			<		>
		~	Shell	×	
MicroPython device	=	$\sim$			
boot.py			Micr	oPython v1.17 on 2021-09-02; ESP32 module with ESP32	
				"help()" for more information.	
			>>>		

Click "Run current script", the code starts to be executed and you can use it to measure the distance between the ultrasonic sensor and the object.

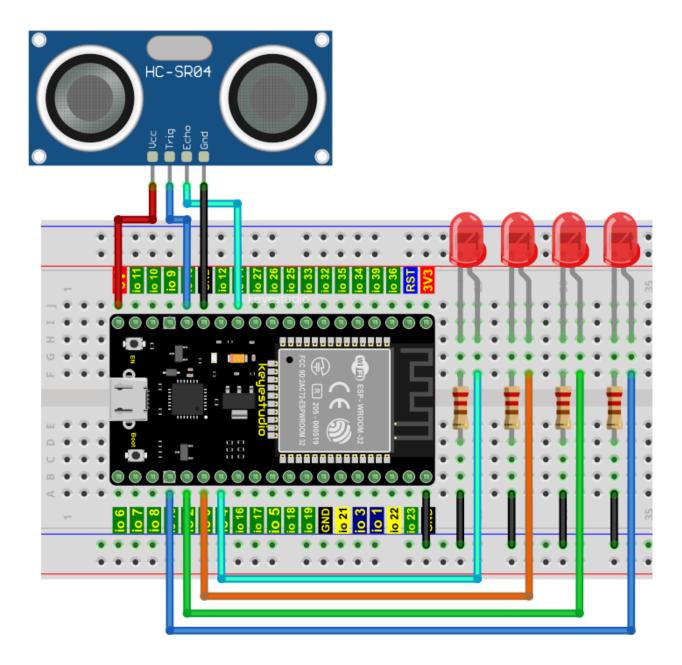
Press"Ctrl+C"or click ""Stop/Restart backend"to exit the program.

1 🖄 🖩 🖸 🔶 🙃 🕨 🚳				
Files × This computer D: \ 2. Python Projects	= ^	Proje	<pre>tt_30.1_Ultrasonic_Ranging.py × from machine import Pin import time</pre>	ŕ
Project 25: Human Induction Lamp  Project 26: Sound Control Fan  Project 27: Temperature Measurement  Project 28: Rocker control light  Project 29: Temperature Humidity Meter		3 4 5 6 7	<pre>#Define the control pins of the ultrasonic ranging modu Trig = Pin(13, Pin.OUT, 0) Echo = Pin(14, Pin.IN, 0)</pre>	c
Project 30: Ultrasonic Ranger  Project 30: Ultrasonic Ranger.docx  Project 30: Ultrasonic Ranging.py		8 9	<pre>distance = 0 # Define the initial distance to be 0. soundVelocity = 340 #Set the speed of sound.</pre>	
<ul> <li>Project_30.2_Ultrasonic_Ranger.py</li> <li>Project 31: Temperature Instrument</li> </ul>	v	< Shell	×	>
MicroPython device Ә boot.py	= ^	Type	oPython v1.17 on 2021-09-02; ESP32 module with ESP32 "help()" for more information. %Run -c \$EDITOR_CONTENT stance: 5 cm	

Shell ×			
	4 Cm		
Distance:	3 cm		
Distance:	4 cm		
Distance:	5 cm		
Distance:	6 cm		
Distance:	6 cm		
Distance:	7 cm		
Distance:	7 cm		
Distance:	8 cm		
Distance:	9 cm		
Distance:	9 cm		
Distance:	10 cm		
Distance:	11 cm		
Distance:	11 cm		
Distance:	12 cm		

# 7.31.5 Wiring diagram of the ultrasonic rangefinder

Next, we will use ESP32 to control an ultrasonic sensor and 4 LEDs to simulate ultrasonic rangefinder. Connect the line as shown below

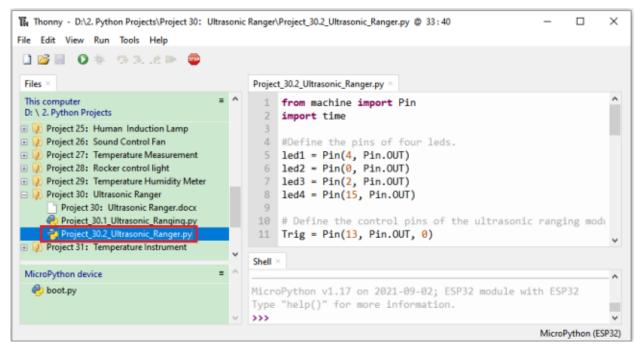


# 7.31.6 Project code

Codes used in this tutorial are saved in "2. Python Projects". If you haven't downloaded the code file, please click on the link to download it:Download Python Codes

2. Python Projects		_	
Share View			~ 🕐
→ This PC → Software (D:) → 2. Pythe	on Projects 🗸 진	,	2. Python
Name	Date modified	Туре	Size ^
Project 00: Boot	2/17/2022 10:21 AM	File folder	
Project 01: Hello World	2/17/2022 10:21 AM	File folder	
Project 02: Turn On LED	2/17/2022 11:10 AM	File folder	
Project 03: LED Flashing	2/17/2022 11:12 AM	File folder	
Project 04: Breathing LED	2/17/2022 10:21 AM	File folder	

 $Open "Thonny" click "This computer" \rightarrow "D:" \rightarrow "2. Python Projects" \rightarrow "Project 30Ultrasonic Ranger" and double left-click "Project_30.2_Ultrasonic_Ranger.py".$ 



```
from machine import Pin
import time
```

#Define the pins of four leds. led1 = Pin(4, Pin.OUT) led2 = Pin(0, Pin.OUT) led3 = Pin(2, Pin.OUT) led4 = Pin(15, Pin.OUT) ## Define the control pins of the ultrasonic ranging module. Trig = Pin(13, Pin.OUT, 0) Echo = Pin(14, Pin.IN, 0) distance = 0 # Define the initial distance to be 0.

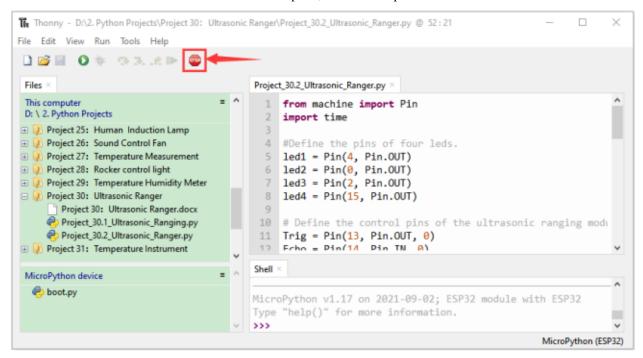
(continues on next page)

(continued from previous page)

```
soundVelocity = 340 #Set the speed of sound.
## The getDistance() function is used to drive the ultrasonic module to measure distance,
## the Trig pin keeps at high level for 10us to start the ultrasonic module.
## Echo.value() is used to read the status of ultrasonic module's Echo pin,
## and then use timestamp function of the time module to calculate the duration of Echo
## pin's high level, calculate the measured distance based on time and return the value.
def getDistance():
    Trig.value(1)
    time.sleep_us(10)
    Trig.value(♥)
    while not Echo.value():
        pass
    pingStart = time.ticks_us()
    while Echo.value():
        pass
    pingStop = time.ticks_us()
    pingTime = time.ticks_diff(pingStop, pingStart) // 2
    distance = int(soundVelocity * pingTime // 10000)
    return distance
## Delay for 2 seconds and wait for the ultrasonic module to stabilize,
## Print data obtained from ultrasonic module every 500 milliseconds.
time.sleep(2)
while True:
    time.sleep_ms(500)
    distance = getDistance()
    print("Distance: ", distance, "cm")
    if distance <= 5:</pre>
       led1.value(1)
    else:
       led1.value(0)
    if distance \leq 10:
       led2.value(1)
    else:
       led2.value(♥)
    if distance <= 15:</pre>
       led3.value(1)
    else:
       led3.value(♥)
    if distance \leq 20:
       led4.value(1)
    else:
       led4.value(♥)
```

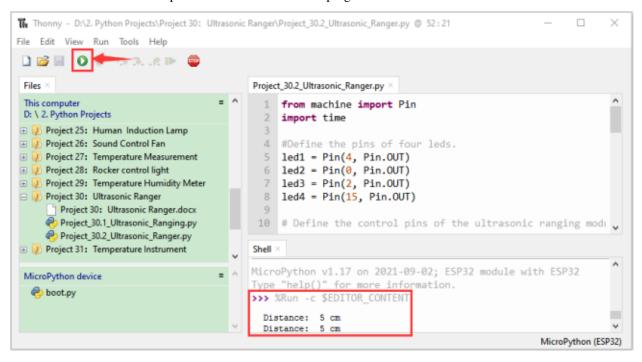
## 7.31.7 Project result

Make sure the ESP32 has been connected to the computer, click "" "Stop/Restart backend".



Click "Run current script", the code starts to be executed and you'll see that the "Shell" window of Thonny IDE will print the distance between the ultrasonic sensor and the object, and the corresponding LED will light up when we move our hand in front of the ultrasonic sensor.

Press"Ctrl+C"or click ""Stop/Restart backend" to exit the program.

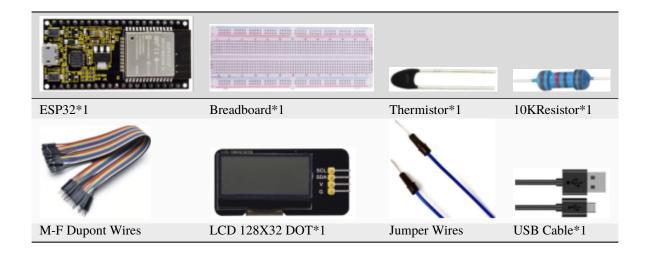


# 7.32 Project 31Temperature Instrument

## 7.32.1 Introduction

Thermistor is a kind of resistor whose resistance depends on temperature changes, which is widely used in gardening, home alarm system and other devices. Therefore, we can use the feature to make a temperature instrument.

# 7.32.2 Components



## 7.32.3 Component knowledge

#### **Thermistor:**

A Thermistor is a temperature sensitive resistor. When it senses a change in temperature, the resistance of the Thermistor will change. We can take advantage of this characteristic by using a Thermistor to detect temperature intensity. A Thermistor and its electronic symbol are shown below:



The relationship between resistance value and temperature of a thermistor is

# $Rt = R * EXP[B * \left(\frac{1}{T2} - \frac{1}{T1}\right)]$

Rt is the thermistor resistance under T2 temperature;

**R** is the nominal resistance of thermistor under T1 temperature;

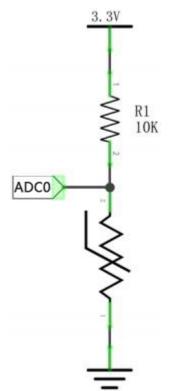
**EXP**[**n**] is nth power of e;

**B** is for thermal index;

T1, T2 is Kelvin temperature (absolute temperature). Kelvin temperature=273.15 + Celsius temperature.

For the parameters of the Thermistor, we use: B=3950, R=10k, T1=25.

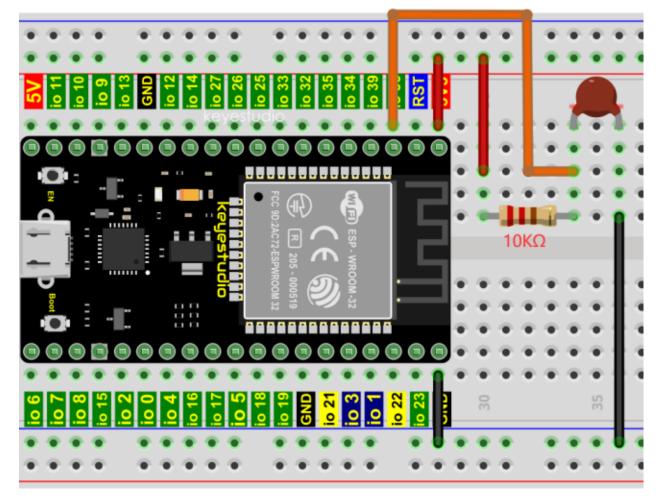
The circuit connection method of the Thermistor is similar to photoresistor, as the following



We can use the value measured by the ADC converter to obtain the resistance value of Thermistor, and then we can use the formula to obtain the temperature value. Therefore, the temperature formula can be derived as:

# 7.32.4 Read the value of the Thermistor

First we will learn the thermistor to read the current ADC value, voltage value and temperature value and print them out. Please connect the wires according to the wiring diagram below

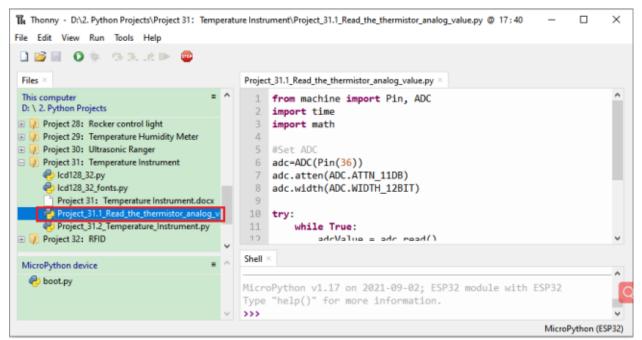


Codes used in this tutorial are saved in "**2. Python Projects**". If you haven't downloaded the code file, please click on the link to download it:Download Python Codes

2. Python Projects		—	$\times$
Share View			~ ?
→ This PC → Software (D:) → 2. Python Proj	ects 🗸 🤇	ら Search 2. Pyth	on
Name	Date modified	Туре	Size ^
Project 00: Boot	2/17/2022 10:21 AM	File folder	
Project 01: Hello World	2/17/2022 10:21 AM	File folder	
Project 02: Turn On LED	2/17/2022 11:10 AM	File folder	
Project 03: LED Flashing	2/17/2022 11:12 AM	File folder	
Project 04: Breathing LED	2/17/2022 10:21 AM	File folder	

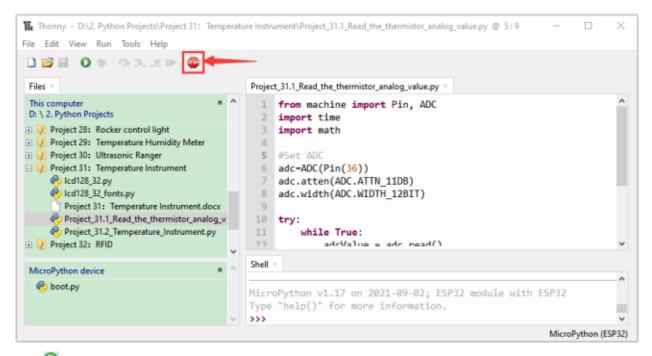
Open"Thonny"click"This computer" $\rightarrow$ "D:" $\rightarrow$ "2. Python Projects" $\rightarrow$ "Project 31Temperature Instrument", and then

double left-click "Project\_31.1\_Read\_the\_thermistor\_analog\_value.py".



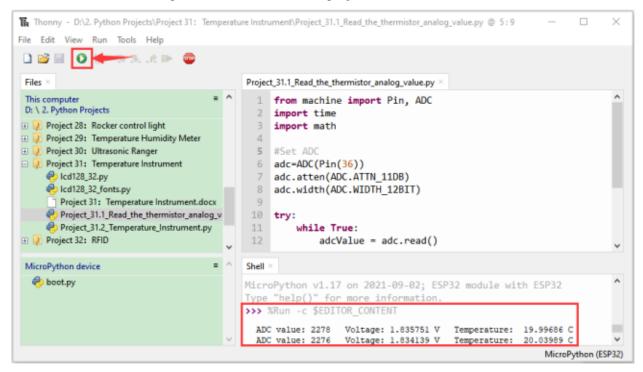
```
from machine import Pin, ADC
import time
import math
#Set ADC
adc=ADC(Pin(36))
adc.atten(ADC.ATTN_11DB)
adc.width(ADC.WIDTH_12BIT)
try:
   while True:
       adcValue = adc.read()
        voltage = adcValue / 4095 * 3.3
       Rt = 10 * voltage / (3.3-voltage)
        tempK = (1 / (1 / (273.15+25) + (math.log(Rt/10)) / 3950))
        tempC = (tempK - 273.15)
       print("ADC value:",adcValue," Voltage:",voltage,"V"," Temperature: ",tempC,"C
time.sleep(1)
except:
   pass
```

Make sure the ESP32 has been connected to the computer, click <sup>10</sup> "Stop/Restart backend".



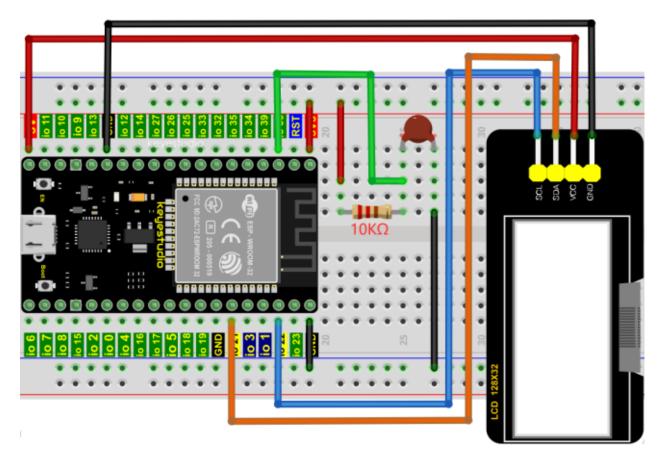
Click "Run current script", the code starts to be executed and you'll see that the "Shell" window of Thonny IDE will continuously display the thermistor's current ADC value, voltage value and temperature value. Try pinching the thermistor with your index finger and thumb (don't touch wires) for a while, and you will see the temperature increase.

Press"Ctrl+C"or click ""Stop/Restart backend"to exit the program.



Shell ×						
>>> %Run -c	\$EDI	TOR_CONTE	NT			^
ADC value:	2305	Voltage:	1.857509 V	Temperature:	19.41592 C	
ADC value:	2287	Voltage:	1.843004 V	Temperature:	19.80316 C	
ADC value:	2256	Voltage:	1.818022 V	Temperature:	20.47055 C	
ADC value:	2246	Voltage:	1.809963 V	Temperature:	20.68604 C	
ADC value:	2271	Voltage:	1.83011 V	Temperature:	20.14752 C	
ADC value:	2269	Voltage:	1.828498 V	Temperature:	20.19058 C	
ADC value:	2197	Voltage:	1.770476 V	Temperature:	21.74371 C	
ADC value:	2218	Voltage:	1.787399 V	Temperature:	21.29001 C	
ADC value:	2251	Voltage:	1.813993 V	Temperature:	20.57831 C	
ADC value:	2227	Voltage:	1.794652 V	Temperature:	21.0958 C	
ADC value:	2227	Voltage:	1.794652 V	Temperature:	21.0958 C	
ADC value:	2247	Voltage:	1.810769 V	Temperature:	20.66449 C	
ADC value:	2257	Voltage:	1.818828 V	Temperature:	20.44904 C	
						~

# 7.32.5 Wiring diagram of the temperature instrument



## 7.32.6 Project code

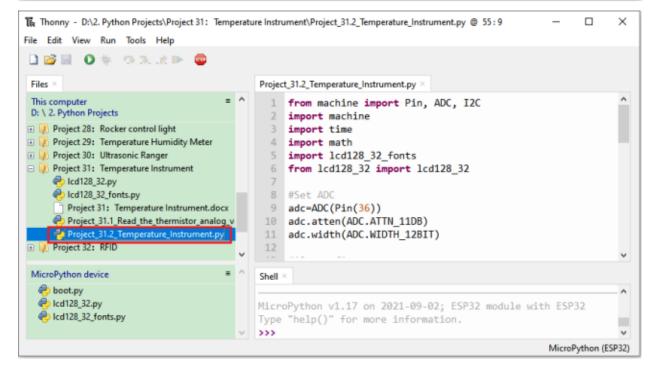
Codes used in this tutorial are saved in "2. Python Projects". If you haven't downloaded the code file, please click on the link to download it:Download Python Codes

2. Python Projects		_	$\Box$ $\times$
Share View			~ ?
→ This PC → Software (D:) → 2. Pyth	non Projects 🗸 진	⊖ Search 2	. Python
Name	Date modified	Туре	Size ^
Project 00: Boot	2/17/2022 10:21 AM	File folder	
Project 01: Hello World	2/17/2022 10:21 AM	File folder	
Project 02: Turn On LED	2/17/2022 11:10 AM	File folder	
Project 03: LED Flashing	2/17/2022 11:12 AM	File folder	
Project 04: Breathing LED	2/17/2022 10:21 AM	File folder	

Open "Thonny", click "This computer"  $\rightarrow$  "D:"  $\rightarrow$  "2. Python Projects"  $\rightarrow$  "Project 31Temperature Instrument". Select "lcd128\_32\_py" and "lcd128\_32\_fonts.py", right-click your mouse to select "Upload to /", wait for "lcd128\_32.py" and "lcd128\_32\_fonts.py" to be uploaded to ESP32, and double left-click "Project\_31.2\_Temperature\_Instrument.py".

The Thonny	_		×
File Edit View Run Tools Help			
🗋 💕 🖩 🕘 🀐 🔿 3e 🕨 🥶			
Files ×			
This computer = ^ D: \ 2. Python Projects			
Project 28: Rocker control light			
Vroject 29: Temperature Humidity Meter     Vroject 30: Ultrasonic Ranger			
IV Project 31: Temperature Instrument			
Copen in Thonny Copen in Thonny			
Project 31: Ten Configure py files			
Project_31.1_Re			
Image: Project 32: RFID Move to Recycle Bin			
MicroPython device nell ×			
Properties			^
MicroPython v1.17 on 2021-09-02; ESP32 module with	ESP3	2	
Type "help()" for more information.			
V >>>	MicroPy	thon (E	SP32)

Te Thonny		-		×
File Edit View Run Tools	Help			
	.et 🕪 🤓			
Files ×				
This computer D: \ 2. Python Projects	= ^			
Project 28: Rocker control     Project 29: Temperature H	umidity Meter			
Project 30: Ultrasonic Range				
Project 31: Temperature In Oct 128_32.py	istrument			
lcd128_32_fonts.py				
Project 31: Temperat	Open in Thonny			
Project_31.1_Read_the	Open in system default app Configure .py files			
Project_31.2_Tempera				
Project 32: RFID	Upload to /			
	Move to Recycle Bin			
MicroPython device	New directory			
😴 boot.py	Properties			_ ^
🤣 lcd128_32.py	MicroPython v1.17 on 2021-09-02; ESP32 module with	h ESP3	32	
	Type "help()" for more information.			
	✓ >>>			~
		MicroP	ython (E	SP32)



from machine import Pin, ADC, I2C
import machine
import time
import math
import lcd128\_32\_fonts
from lcd128\_32 import lcd128\_32

#Set ADC

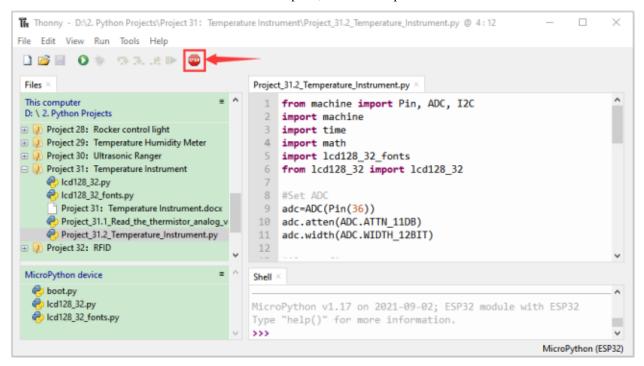
(continues on next page)

(continued from previous page)

```
adc=ADC(Pin(36))
adc.atten(ADC.ATTN_11DB)
adc.width(ADC.WIDTH_12BIT)
#i2c config
clock_pin = 22
data_pin = 21
bus = 0
i2c_addr = 0x3f
use_i2c = True
def scan_for_devices():
    i2c = machine.I2C(bus,sda=machine.Pin(data_pin),scl=machine.Pin(clock_pin))
    devices = i2c.scan()
    if devices:
        for d in devices:
            print(hex(d))
    else:
        print('no i2c devices')
try:
    while True:
        adcValue = adc.read()
        voltage = adcValue / 4095 * 3.3
        Rt = 10 \times \text{voltage} / (3.3 - \text{voltage})
        tempK = (1 / (1 / (273.15+25) + (math.log(Rt/10)) / 3950))
        tempC = int(tempK - 273.15)
        if use_i2c:
            scan_for_devices()
            lcd = lcd128_32(data_pin, clock_pin, bus, i2c_addr)
        lcd.Clear()
        lcd.Cursor(∅, ∅)
        lcd.Display("Voltage:")
        lcd.Cursor(0, 8)
        lcd.Display(str(voltage))
        lcd.Cursor(0, 20)
        lcd.Display("V")
        lcd.Cursor(2, ◊)
        lcd.Display("Temperature:")
        lcd.Cursor(2, 12)
        lcd.Display(str(tempC))
        lcd.Cursor(2, 15)
        lcd.Display("C")
        time.sleep((0.5))
except:
    pass
```

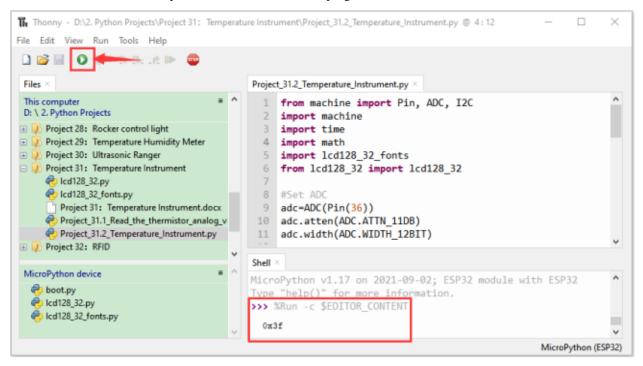
## 7.32.7 Project result

Make sure the ESP32 has been connected to the computer, click "Stop/Restart backend".



Click "Run current script", the code starts to be executed and you'll see that the LCD 128X32 DOT displays the voltage value of the thermistor and the temperature value in the current environment.

Press"Ctrl+C"or click <sup>22</sup> "Stop/Restart backend" to exit the program.

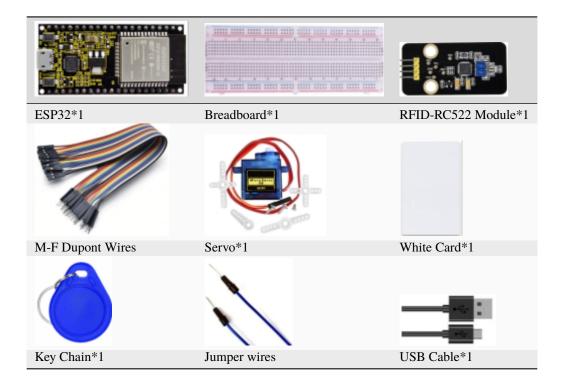


# 7.33 Project 32RFID

#### 7.33.1 Introduction

Nowadays, many residential districts use this function to open the door by swiping the card, which is very convenient. In this Project, we will learn how to use RFID(radio frequency identification) wireless communication technology and read and write the key chain card (white card) and control the steering gear rotation by RFID-MFRC522 module.

#### 7.33.2 Components



## 7.33.3 Component knowledge

#### RFID

RFID (Radio Frequency Identification) is a wireless communication technology. A complete RFID system is generally composed of the responder and reader. Generally, we use tags as responders, and each tag has a unique code, which is attached to the object to identify the target object. The reader is a device for reading (or writing) tag information.

Products derived from RFID technology can be divided into three categories: passive RFID products, active RFID products and semi active RFID products. And Passive RFID products are the earliest, the most mature and most widely used products in the market among others. It can be seen everywhere in our daily life such as, the bus card, dining card, bank card, hotel access cards, etc., and all of these belong to close-range contact recognition. The main operating frequency of Passive RFID products are: 125KHZ (low frequency), 13.56MHZ (high frequency), 433MHZ (ultrahigh frequency) and 915MHZ (ultrahigh frequency). Active and semi active RFID products work at higher frequencies.

The RFID module we use is a passive RFID product with the operating frequency of 13.56MHz.

#### **RFID-RC522** Module

The MFRC522 is a highly integrated reader/writer IC for contactless communication at 13.56MHz. The MFRC522's internal transmitter is able to drive a reader/writer antenna designed to communicate with ISO/IEC 14443 A/MIFARE cards and transponders without additional active circuitry. The receiver module provides a robust and efficient implementation for demodulating and decoding signals from ISO/IEC 14443 A/MIFARE compatible cards and transponders. The digital module manages the complete ISO/IEC 14443A framing and error detection (parity and CRC) functionality.

This RFID Module uses MFRC522 as the control chip and adopts I2C (Inter-Integrated Circuit) interface.

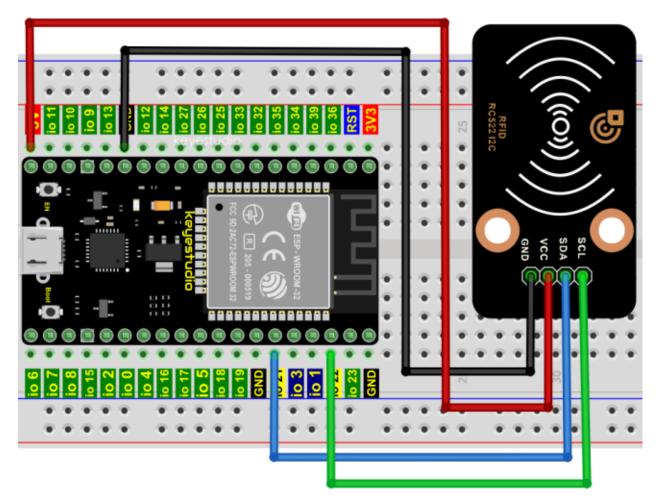


#### **Specifications:**

- Operating voltage: DC 3.3V-5V
- Operating current: 13—100mA/DC 5V
- Idling current: 10-13mA/DC 5V
- Sleep current: <80uA
- Peak current: <100mA
- Operating frequency: 13.56MHz
- Maximum power: 0.5W
- Supported card types: mifare1 S50, mifare1 S70, mifare UltraLight, mifare Pro, mifare Desfire.
- Environmental operating temperature: -20 to 80 degrees Celsius.
- Environment storage temperature: -40 to 85 degrees Celsius.
- Relative Humidity: 5% to 95%.
- Data transfer rate: The maximum is 10Mbit/s.

#### 7.33.4 RFID Read UID

We will read the UNIQUE ID number (UID) of the RFID card and identify the type of the RFID card, and display the relevant information through the serial port. The wiring diagram is shown below



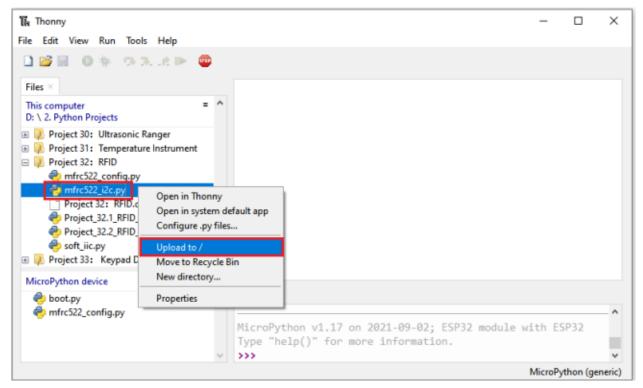
Codes used in this tutorial are saved in "2. Python Projects". If you haven't downloaded the code file, please click on the link to download it:Download Python Codes

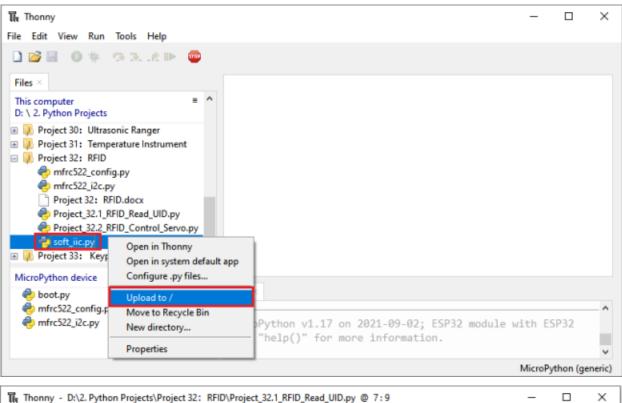
2. Python Projects		— [	
Share View			~ 🕐
→ This PC → Software (D:) → 2. Python	Projects 🗸 🗸	,○ Search 2.	Python
Name	Date modified	Туре	Size ^
Project 00: Boot	2/17/2022 10:21 AM	File folder	
	2/17/2022 10:21 AM	File folder	
Project 02: Turn On LED	2/17/2022 11:10 AM	File folder	
Project 03: LED Flashing	2/17/2022 11:12 AM	File folder	
Project 04: Breathing LED	2/17/2022 10:21 AM	File folder	

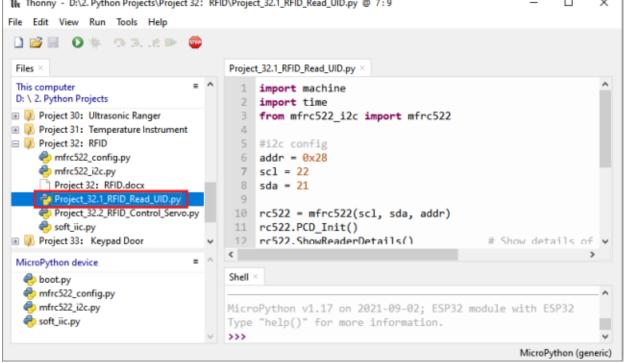
Open "Thonny", click "This computer"  $\rightarrow$  "D:"  $\rightarrow$  "2. Python Projects"  $\rightarrow$  "Project 32RFID".

Select "mfrc522\_config.py", "mfrc522\_i2c.py" and "soft\_iic.py", right-click your mouse to select "Upload to /", wait for "mfrc522\_config.py", "mfrc522\_i2c.py" and "soft\_iic.py" to be uploaded to ESP32, and double left-click "Project\_32.1\_RFID\_Read\_UID.py".

🏹 Thonny					_		×
File Edit View Run Tools Help							
0 🗃 🖬 0 🐐 🕫 3e 🕨							
Files ×							
This computer D: \ 2. Python Projects	= ^						
🗉 📜 Project 30: Ultrasonic Ranger							
🗉 🕖 Project 31: Temperature Instrume	ent						
🖃 📜 Project 32: RFID							
mfrc522_config.py Op	en in Thonny	y					
e mfrc522_i2c.py Op	en in system	default app					
Project 32: RFID.docx Project_32.1_RFID_Rea	nfigure .py fi	iles					
	load to /						
	ove to Recycl	e Bin					
	w directory						
MicroPython device Pro	operties						_
eboot.py		Shell					
							^
		MicroPythor	n v1.17 on 2021	L-09-02; ESP32	module with ES	5P32	
		Type "help	()" for more in	formation.			
	$\sim$	>>>					~
					MicroPy	thon (ge	neric)







import machine import time from mfrc522\_i2c import mfrc522

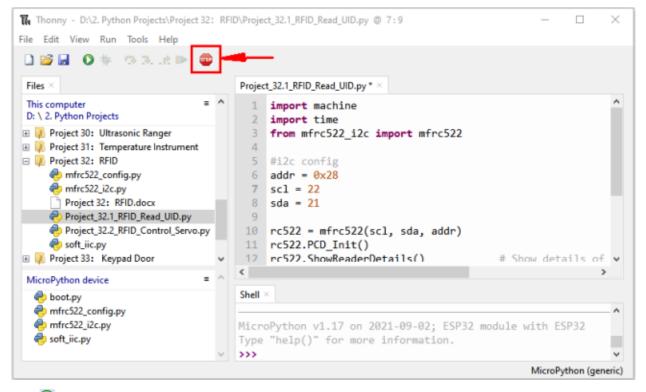
#i2c config

(continues on next page)

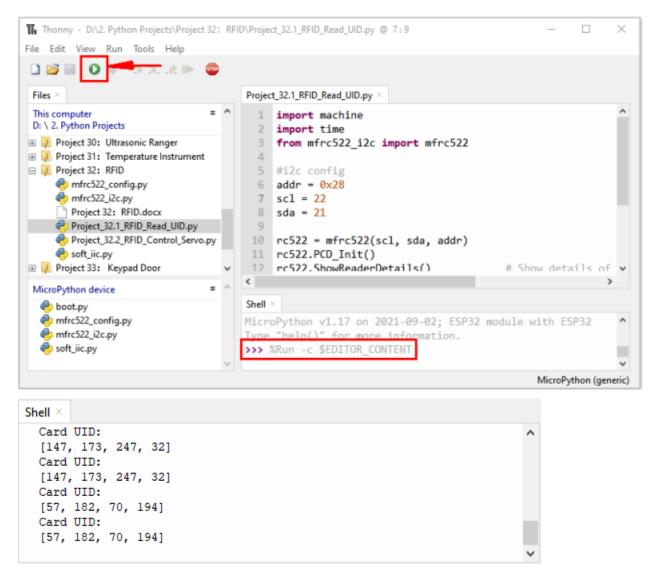
(continued from previous page)

```
addr = 0x28
scl = 22
sda = 21
rc522 = mfrc522(scl, sda, addr)
rc522.PCD_Init()
rc522.ShowReaderDetails()  # Show details of PCD - MFRC522 Card Reader details
while True:
    if rc522.PICC_IsNewCardPresent():
        #print("Is new card present!")
        if rc522.PICC_ReadCardSerial() == True:
            print("Card UID:")
            print(rc522.uid.uidByte[0 : rc522.uid.size])
    #time.sleep(1)
```

Make sure the ESP32 has been connected to the computer, click <sup>10</sup> "Stop/Restart backend".



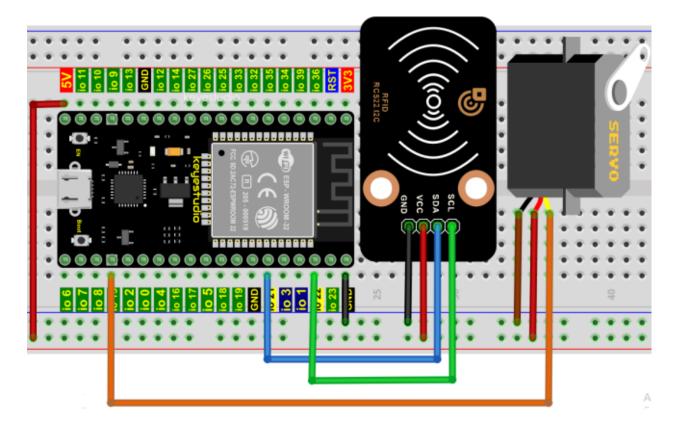
Click Circle Click



Note: the door card value and key chain value may be different for different RRFID -RC522 door cards and key chains.

# 7.33.5 Wiring diagram of the RFID MFRC522

Now we use the RFID -RC522 module, white card/key chain and Servo to simulate an intelligent access control system. When the white card/key chain close to the RFID -RC522 module induction area, the servo rotates. Wiring according to the figure below



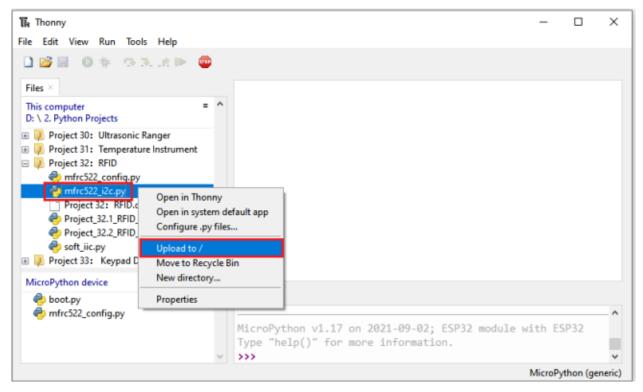
## 7.33.6 Project code

Codes used in this tutorial are saved in "2. Python Projects". If you haven't downloaded the code file, please click on the link to download it:Download Python Codes

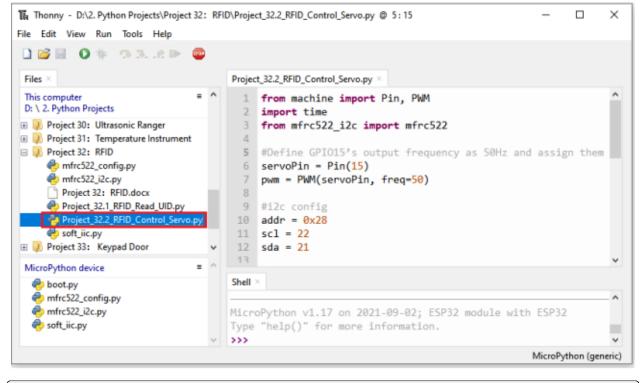
2. Python Projects		-	$\square$ ×
Share View			~ 🕐
→ This PC → Software (D:) → 2. Pyth	non Projects 🗸 진		Python
Name	Date modified	Туре	Size ^
	2/17/2022 10:21 AM	File folder	
Project 01: Hello World	2/17/2022 10:21 AM	File folder	
	2/17/2022 11:10 AM	File folder	
Project 03: LED Flashing	2/17/2022 11:12 AM	File folder	
Project 04: Breathing LED	2/17/2022 10:21 AM	File folder	

Open"Thonny", click"This computer" $\rightarrow$ "D:" $\rightarrow$ "2. Python Projects" $\rightarrow$ "Project 32RFID". Select "mfrc522\_config.py", "mfrc522\_i2c.py" and "soft\_iic.py", click your mouse to select "Upload to /", wait for "mfrc522\_config.py", "mfrc522\_i2c.py" and "soft\_iic.py" to be uploaded to ESP32, and click "Project\_32.2\_RFID\_Control\_Servo.py".

The Thonny					_		×
File Edit View Run Tools Help	)						
🗋 💕 🖬 🛛 🌞 🗇 3t. i Files ×	▶ 😳						
This computer D: \ 2. Python Projects	= ^						
Project 30: Ultrasonic Ranger     Project 31: Temperature Instrum     Project 32: RFID	ment						
Project 32: REID docr	Open in Thonny Open in system Configure .py fi	default app					
🔿 soft_iic.py 🛛 🛛	Jpload to / Move to Recycle New directory						
MicroPython device Pr	roperties						
🗢 boot.py		Shell ×					
			n v1.17 on 2 ()" for more	 ESP32 module n.	with ES	5P32	~
					MicroPy	thon (ge	neric)



🏹 Thonny		-	
ile Edit View Run T	Tools Help		
🗋 💕 🖩 🛛 🎄 🧐	3 3. in 🕨 👳		
This computer D: \ 2. Python Projects	= ^		
	ature Instrument .py D.docx		
B Project 33: Keyp	Open in Thonny Open in system default	upp	
MicroPython device	Configure .py files		
<ul> <li>boot.py</li> <li>mfrc522_config.p</li> <li>mfrc522_i2c.py</li> </ul>	Upload to / Move to Recycle Bin New directory	Python v1.17 on 2021-09-02; ESP32 module with E "help()" for more information.	SP32
	Properties		~
		MicroP	ython (generic)



# from machine import Pin, PWM import time from mfrc522\_i2c import mfrc522

#Define GPI015's output frequency as 50Hz and assign them to PWM.

(continues on next page)

(continued from previous page)

```
servoPin = Pin(15)
pwm = PWM(servoPin, freq=50)
#i2c config
addr = 0x28
scl = 22
sda = 21
rc522 = mfrc522(scl, sda, addr)
rc522.PCD_Init()
rc522.ShowReaderDetails()
                                      # Show details of PCD - MFRC522 Card Reader details
uid1 = [147, 173, 247, 32]
uid2 = [57, 182, 70, 194]
pwm = PWM(servoPin, freq=50)
pwm.duty(128)
time.sleep(1)
while True:
   if rc522.PICC_IsNewCardPresent():
        #print("Is new card present!")
        if rc522.PICC_ReadCardSerial() == True:
            print("Card UID:", end=' ')
            print(rc522.uid.uidByte[0 : rc522.uid.size])
            if rc522.uid.uidByte[0 : rc522.uid.size] == uid1 or rc522.uid.uidByte[0 :_

→rc522.uid.size] == uid2:

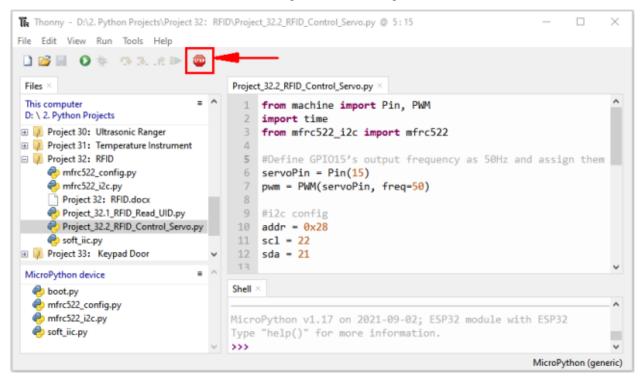
                pwm = PWM(servoPin, freq=50)
                pwm.duty(25)
            else :
                pwm = PWM(servoPin, freq=50)
                pwm.duty(128)
            time.sleep(500)
```

Note: Different RFID-RC522 modules, ID cards and key chains may cause different uid1 values and uid2 values. The UID1 and UID2 values of the white card and key chain read by your RRFID RC522 module can be replaced by the corresponding values in the program code. If not, click "Run current script" to run the code may cause your own white card and key chain to fail to control the servo.

For	example:	You	can	replace	the	UID1	and	UID2	values	in	the	program
	uid1 =	[147	, 1	73, 24	47,	32]						
code	uid2 =	[57,	18	2, 70	, 1	94]	with you	ur own wk	ite card ar	ad kov	ahain 1	201000
coue							with yo	ui owii wi	inte caru ai	iu key	chann v	alues.

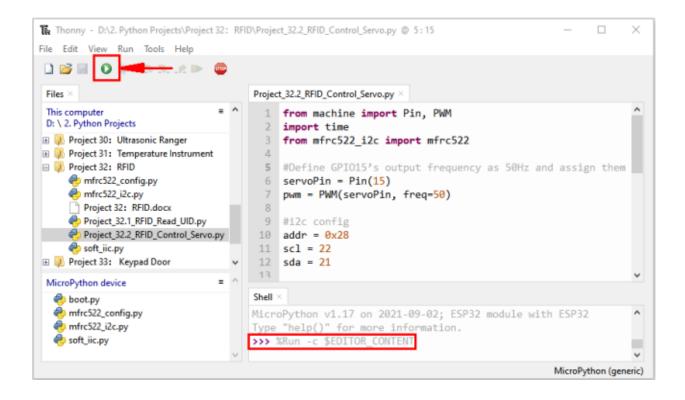
# 7.33.7 Project result

Make sure the ESP32 has been connected to the computer, click "" "Stop/Restart backend".



Click "Run current script", the code starts to be executed and you'll see that when using the white card or a key card swiping, the "Shell" window of Thonny IDE displays the card number value respectively, and at the same time, the servo rotates to the corresponding angle to simulate opening the door.

Press"Ctrl+C"or click "Stop/Restart backend" to exit the program.



# 7.34 Project 33Keypad Door

# 7.34.1 Introduction

Commonly used digital button sensor, one button uses an IO port. However, it will occupy too many IO ports when we need a lot of buttons. In order to save the use of IO ports, the multiple buttons are made into a matrix type, through the control of the line and row to achieve less IO port control of multiple buttons. In this project, we will learn ESP32 and thin film 4\*4 matrix keyboard control a servo and a buzzer.

# 7.34.2 Components

ESP32*1	Breadboard*1	Servo*1	Active Buzzer*1
		14 <sup>1</sup> 12 11 11	-(111)
44 Membrane Matrix Keyboard1	Jumper Wires	USB Cable*1	1kResistor*1
NPN transistor(S8050)*1			

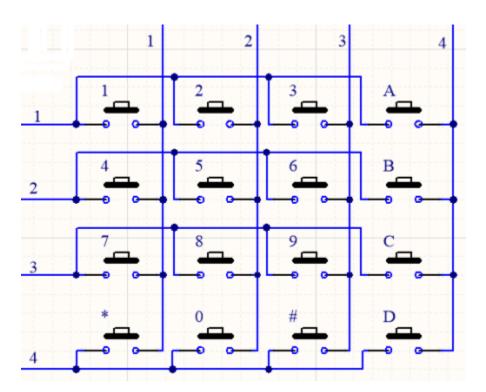
# 7.34.3 Component knowledge

#### 4\*4 Matrix keyboard

A Keypad Matrix is a device that integrates a number of keys in one package. As is shown below, a 4x4 Keypad Matrix integrates 16 keys:



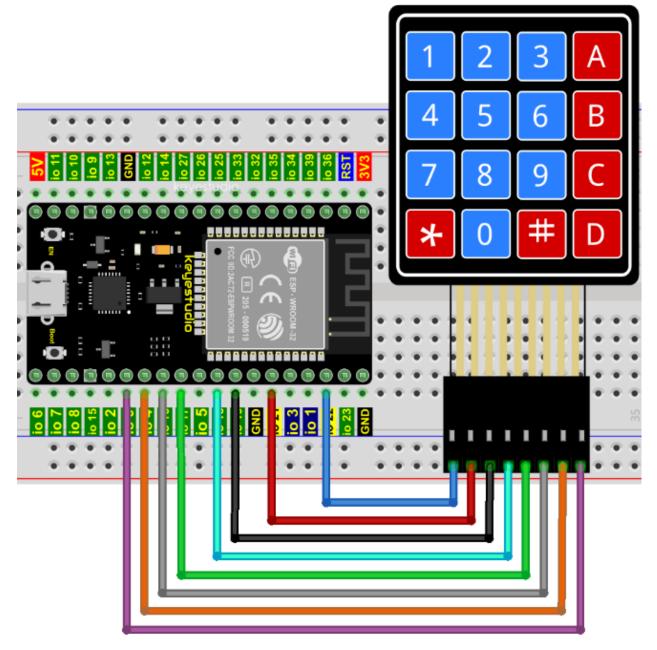
Similar to the integration of an LED Matrix, the 4x4 Keypad Matrix has each row of keys connected with one pin and this is the same for the columns. Such efficient connections reduce the number of processor ports required. The internal circuit of the Keypad Matrix is shown below.



The method of usage is similar to the Matrix LED, by using a row or column scanning method to detect the state of each key's position by column and row. Take column scanning method as an example, send low level to the first 4 column (Pin4), detect level state of row 1, 2, 3, 4 to judge whether the key A, B, C, D are pressed. Then send low level to column3, 2, 1 in turn to detect whether other keys are pressed. By this means, you can get the state of all of the keys.

# 7.34.4 Read the key value of the 4\*4 matrix keyboard

We start with a simple code to read the values of the 4\*4 matrix keyboard and print them in the serial monitor. Its wiring diagram is shown below

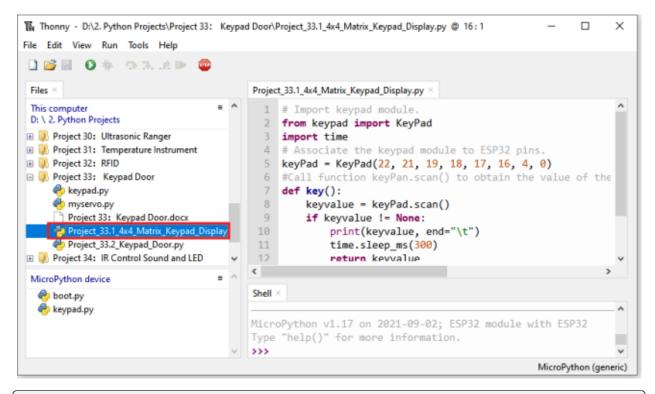


Codes used in this tutorial are saved in "2. Python Projects". If you haven't downloaded the code file, please click on the link to download it:Download Python Codes

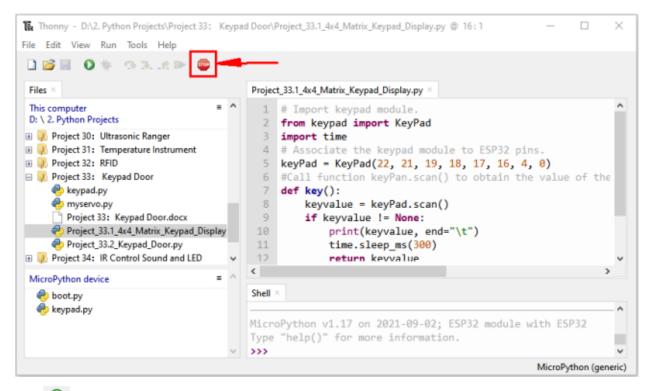
2. Python Projects		_	$\Box$ $\times$
Share View			~ 🕐
→ This PC → Software (D:) → 2. Pyth	on Projects 🗸 진	, ○ Search 2	. Python
Name	Date modified	Туре	Size ^
Project 00: Boot	2/17/2022 10:21 AM	File folder	
	2/17/2022 10:21 AM	File folder	
	2/17/2022 11:10 AM	File folder	
Project 03: LED Flashing	2/17/2022 11:12 AM	File folder	
Project 04: Breathing LED	2/17/2022 10:21 AM	File folder	

Open"Thonny", click "This computer"  $\rightarrow$  "D:"  $\rightarrow$  "2. Python Projects"  $\rightarrow$  "Project 33Keypad Door". Select "keypad.py", select "Upload to l", wait for "keypad.py" to be uploaded to ESP32, and click "Project\_33.1\_4x4\_Matrix\_Keypad\_Display.py".

TR Thonny			-		$\times$
File Edit View Run T	ools Help				
	3e 🕨 🐵				
Files ×					
This computer D: \ 2. Python Projects	= ^				
🗉 🔑 Project 30: Ultrasor	-				
Project 31: Temper	ture Instrument				
⊕ Project 32: RFID     ⊖ Project 33: Keypad	Door				
Reypad.py Reypad.py myservo.py Project 33: K Project_33.1_4	Open in Thonny Open in system default app Configure .py files				
Project_33.2_	Upload to /				
🗉 🔑 Project 34: IR Cc	Move to Recycle Bin				
MicroPython device	New directory				_
🌏 boot.py	Properties ell ×				
		ython v1.17 on 2021-09-02; help()" for more informatio	n.		~
			Micro	Python (ge	neric)

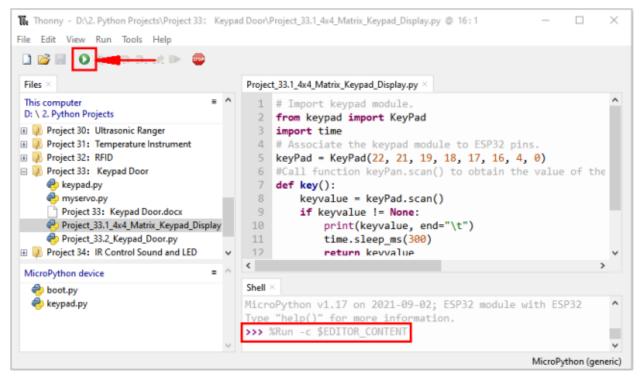


Make sure the ESP32 has been connected to the computer, click <sup>1</sup> Stop/Restart backend'.



Click Click

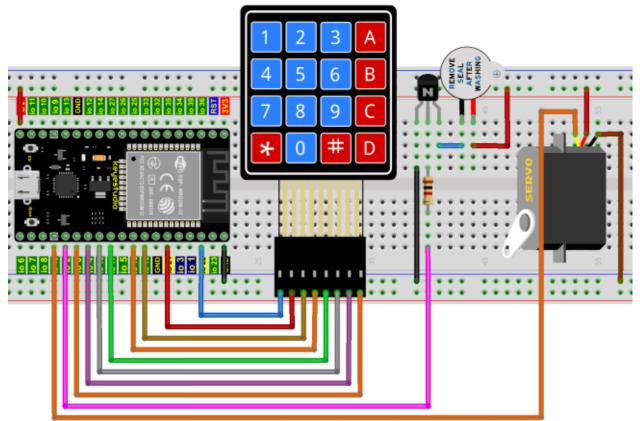
window of Thonny IDE prints the corresponding key value, as shown below. Press"Ctrl+C"or click <sup>11</sup> "Stop/Restart backend" to exit the program.



Shell ×							
>>> %Ru	n -c \$E	DITOR_CO	NTENT				
1	2	3	A	4	5	6	в
7	8	9	С	*	0	#	D

# 7.34.5 Wiring diagram of the Keypad Door

In the last experiment, we have known the key values of the 4\*4 matrix keyboard. Next, we use it as the keyboard to control a servo and a buzzer.



# 7.34.6 Project code

Codes used in this tutorial are saved in "2. Python Projects". If you haven't downloaded the code file, please click on the link to download it:Download Python Codes

2. Python Projects		_	$\Box$ $\times$
Share View			~ ?
→ This PC → Software (D:) → 2. Pyth	hon Projects 🗸 🗸	ට 🔎 Search 2	2. Python
Name	Date modified	Туре	Size ^
Project 00: Boot	2/17/2022 10:21 AM	File folder	
Project 01: Hello World	2/17/2022 10:21 AM	File folder	
Project 02: Turn On LED	2/17/2022 11:10 AM	File folder	
Project 03: LED Flashing	2/17/2022 11:12 AM	File folder	
Project 04: Breathing LED	2/17/2022 10:21 AM	File folder	

Open "Thonny", click "This computer"  $\rightarrow$  "D:"  $\rightarrow$  "2. Python Projects"  $\rightarrow$  "Project 33 Keypad Door". Select "keypad.py" and "myservo.py", click your mouse to select "Upload to /", wait for "keypad.py" and "myservo.py" to be uploaded to ESP32, and , click "Project\_33.2\_Keypad\_Door.py".

🙀 Thonny		-		×
File Edit View Run 1	Tools Help			
0 😸 🖩 0 🚸 🤄	8 Be 🕨 🥶			
Files ×				
This computer D: \ 2. Python Projects	= ^			
🗉 📜 Project 30: Ultrasor	-			
Project 31: Temper	ature Instrument			
Image: Book of the second	Door			
🛃 keypad.py	Open in Thonny			
💨 myservo.py	Open in system default ap	P		
Project 33: K	Configure .py files			
Project_33.2	Upload to /			
🗉 📜 Project 34: IR Cc	Move to Recycle Bin	-		
MicroPython device	New directory			
e boot.py	Properties	ell ×		
				^
		MicroPython v1.17 on 2021-09-02; ESP32 module with	ESP32	
	~	Type "help()" for more information.		~
			oPython (ger	neric)

The Thonny				-	
				- 0	×
File Edit View Run Tools Help	-				
0 🖉 🖩 0 🐐 🧿 3e 🕨	5700				
Files ×					
This computer D: \ 2. Python Projects	=	^			
🗄 📜 Project 30: Ultrasonic Ranger					
<ul> <li>Project 31: Temperature Instrume</li> <li>Project 32: RFID</li> </ul>	ent				
Project 33: Keypad Door					
le keypad.py			_		
Project 33: K			1		
Project_33.1_4 Configure 1		ult app			
Project_33.2_K	y mes		1		
Move to Re	cycle Bin				
MicroPython device New directo					
boot.py Reverse Properties					^
				Python v1.17 on 2021-09-02; ESP32 module with ESP32	
				"help()" for more information.	
		~	>>>	MicroPython (g	
				incorport (g	enency
Thonny - D:\2. Python Projects\Proje	ect 33: K	eypad (	)oor\P	Project_33.2_Keypad_Door.py @ 45:19	×
File Edit View Run Tools Help					
🗋 💕 🖩 🛛 🐐 🔿 3e 🕨	5700				
Files ×			Proiect	_33.2_Keypad_Door.py ×	
This computer	=	^	1	<pre>from myservo import myServo #Import myservo module.</pre>	^
D: \ 2. Python Projects			2	from keypad import KeyPad	
Project 30: Ultrasonic Ranger     Project 31: Temperature Instrume	ent		3	<pre>from machine import Pin import time</pre>	
🗉 📜 Project 32: RFID			5	amport came	
			6	<pre>keyPad = KeyPad(22, 21, 19, 18, 17, 16, 4, 0)</pre>	
Project 33: Keypad Door					
🔿 keypad.py			7	servo=myServo(15)	
<ul> <li>keypad.py</li> <li>myservo.py</li> <li>Project 33: Keypad Door.docx</li> </ul>		d		<pre>servo=myServo(15) servo.myServoWriteAngle(0) time.sleep_ms(1000)</pre>	
<ul> <li>keypad.py</li> <li>myservo.py</li> <li>Project 33: Keypad Door.docx</li> <li>Project 33.1 4x4 Matrix Keypad</li> </ul>	ad_Display	,	7 8 9 10	<pre>servo=myServo(15) servo.myServoWriteAngle(0)</pre>	
<ul> <li>keypad.py</li> <li>myservo.py</li> <li>Project 33: Keypad Door.docx</li> </ul>	ad_Display	Ţ	7 8 9	<pre>servo=myServo(15) servo.myServoWriteAngle(0) time.sleep_ms(1000)</pre>	
<ul> <li>keypad.py</li> <li>myservo.py</li> <li>Project 33: Keypad Door.docx</li> <li>Project 33.1 4x4 Matrix Keypa</li> <li>Project 33.2 Keypad_Door.py</li> <li>Project 34: IR Control Sound and</li> </ul>	ad_Display	*	7 8 9 10 11	<pre>servo=myServo(15) servo.myServoWriteAngle(0) time.sleep_ms(1000) activeBuzzer = Pin(2, Pin.OUT)</pre>	4
<ul> <li>keypad.py</li> <li>myservo.py</li> <li>Project 33: Keypad Door.docx</li> <li>Project 33.1 4x4 Matrix Keypa</li> <li>Project _33.2_Keypad_Door.py</li> </ul>	ad_Display LED	<b>*</b>	7 8 9 10 11 12	<pre>servo=myServo(15) servo.myServoWriteAngle(0) time.sleep_ms(1000) activeBuzzer = Pin(2, Pin.OUT) # Define an array and set the password. nassWord = "1234"</pre>	¢
<ul> <li>keypad.py</li> <li>myservo.py</li> <li>Project 33: Keypad Door.docx</li> <li>Project 33.1 4x4 Matrix Keypad</li> <li>Project 33.2 Keypad_Door.py</li> <li>Project 34: IR Control Sound and</li> <li>MicroPython device</li> <li>boot.py</li> <li>keypad.py</li> </ul>	ad_Display LED	*	7 8 9 10 11 12 13	<pre>servo=myServo(15) servo.myServoWriteAngle(0) time.sleep_ms(1000) activeBuzzer = Pin(2, Pin.OUT) # Define an array and set the password. nassWord = "1234"</pre>	~
<ul> <li>keypad.py</li> <li>myservo.py</li> <li>Project 33: Keypad Door.docx</li> <li>Project 33.1 4x4 Matrix Keypa</li> <li>Project 33.2 Keypad_Door.py</li> <li>Project 34: IR Control Sound and</li> <li>MicroPython device</li> <li>boot.py</li> </ul>	ad_Display LED	<b>*</b>	7 8 9 10 11 12 13 Shell	<pre>servo=myServo(15) servo.myServoWriteAngle(0) time.sleep_ms(1000) activeBuzzer = Pin(2, Pin.OUT) # Define an array and set the password. nassWord = "1234" </pre>	~
<ul> <li>keypad.py</li> <li>myservo.py</li> <li>Project 33: Keypad Door.docx</li> <li>Project 33.1 4x4 Matrix Keypad</li> <li>Project 33.2 Keypad_Door.py</li> <li>Project 34: IR Control Sound and</li> <li>MicroPython device</li> <li>boot.py</li> <li>keypad.py</li> </ul>	ad_Display LED	*	7 8 9 10 11 12 13 Shell	<pre>servo=myServo(15) servo.myServoWriteAngle(0) time.sleep_ms(1000) activeBuzzer = Pin(2, Pin.OUT) # Define an array and set the password. nassWord = "1234"</pre>	>

```
from myservo import myServo #Import myservo module.
from keypad import KeyPad
from machine import Pin
import time
keyPad = KeyPad(22, 21, 19, 18, 17, 16, 4, 0)
```

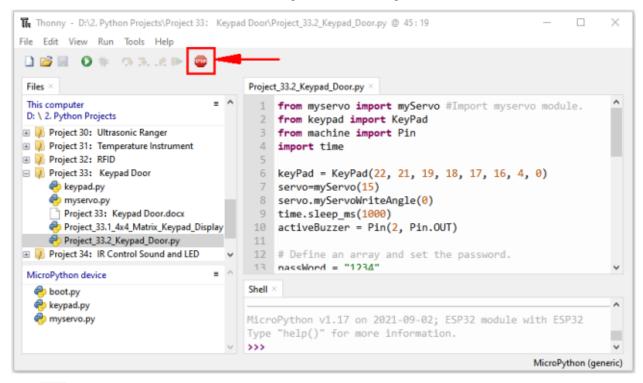
(continues on next page)

(continued from previous page)

```
servo=myServo(15)
servo.myServoWriteAngle(0)
time.sleep_ms(1000)
activeBuzzer = Pin(2, Pin.OUT)
## Define an array and set the password.
passWord = "1234"
keyIn = ""
def key():
    keyvalue = keyPad.scan()
    if keyvalue != None:
        print('Your input:', keyvalue)
        time.sleep_ms(200)
        return keyvalue
while True:
 # Each time a key is pressed, the buzzer will short beep once,
# and the key value of the key will be stored in the keyIn array.
    keydata = key()
    if keydata != None:
        activeBuzzer.value(1)
        time.sleep ms(100)
        activeBuzzer.value(0)
        keyIn += keydata
## When 4 keys are pressed, it will judge whether the password is correct.
## If it is correct, the servo will rotate 90 degrees, and then turn back after 1 second.
## If the password is wrong, the buzzer will long beep once and the keyInNum value will.
\rightarrow be cleared.
    if len(keyIn) == 4:
        if keyIn == passWord:
            print("passWord right!")
            servo.myServoWriteAngle(90)
            time.sleep_ms(1000)
            servo.myServoWriteAngle(0)
        else:
            print("passWord error!")
            activeBuzzer.value(1)
            time.sleep_ms(1000)
            activeBuzzer.value(0)
        keyIn = ""
```

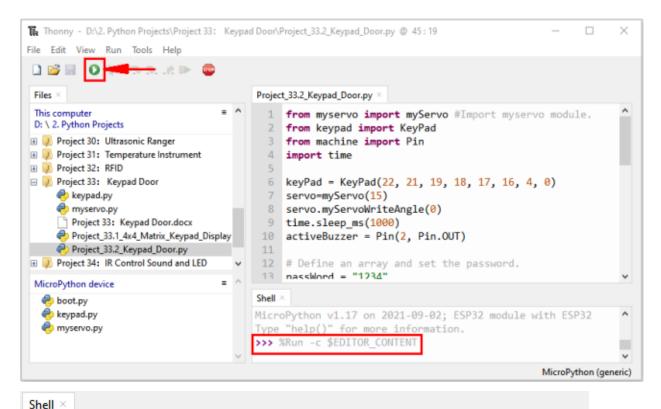
# 7.34.7 Project result

Make sure the ESP32 has been connected to the computer, click <sup>1</sup> "Stop/Restart backend".



Click Click

Press"Ctrl+C"or click <sup>22</sup>"Stop/Restart backend"to exit the program.



>>> %Run -c \$EDITOR\_CONTENT Your input: 1 Your input: 2 Your input: 3 Your input: 4

Your input: 4 passWord right! Your input: 1 Your input: 2 Your input: 4 Your input: 5 passWord error!

# 7.35 Project 34IR Control Sound and LED

## 7.35.1 Introduction

An infrared(IR) remote control is a low-cost and easy-to-use wireless communication technology. IR light is very similar to visible light, except that its wavelength is slightly longer. This means that infrared rays cannot be detected by the human eye, which is perfect for wireless communication. For example, when you press a button on the TV remote control, an infrared LED will switch on and off repeatedly at a frequency of 38,000 times per second, transmitting information (such as volume or channel control) to the infrared sensor on the TV.

We'll start by explaining how common infrared communication protocols work. Then we will start the project with a remote control and an infrared receiver component.

# 7.35.2 Components

ESP32*1	Breadboard*1	IR Receiver *1	RGB LED*1
A C C C C C C C C C C C C C C C C C C C			(IIII)
IR Remote Controller*1	Active buzzer*1	10KResistor*1	220Resistor*3
NPN transistor(S8050)*1	1kResistor*1	Jumper Wires	USB Cable*1

# 7.35.3 Component knowledge

#### **Infrared Remote**

An infrared (IR) remote control is a device with a certain number of buttons. Pressing down different buttons will make the infrared emission tube, which is located in the front of the remote control, send infrared ray with different command. Infrared remote control technology is widely used in electronic products such as TV, airconditioning, etc. Thus making it possible for you to switch TV programs and adjust the temperature of the air conditioning when away from them. The remote control we use is shown below:

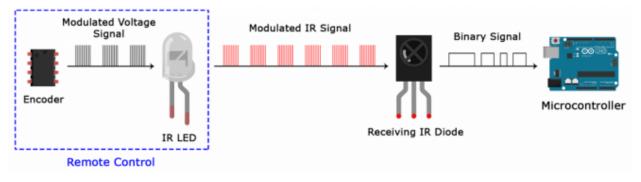
The infrared remote controller adopts NEC code and the signal cycle is 110ms.



\*\*Infrared receiver\*\*receiver is a component which can receive the infrared light, so we can use it to detect the signal emitted by the infrared remote control.

The infrared receiver demodules the received infrared signal and converts it back to binary, then passes the information to the microcontroller.

#### Infrared signal modulation process diagram



#### **NEC Infrared communication protocol**

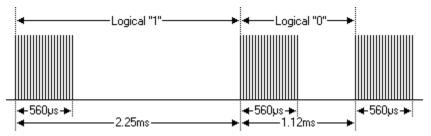
#### **NEC Protocol:**

To my knowledge the protocol I describe here was developed by NEC (Now Renesas). I've seen very similar protocol descriptions on the internet, and there the protocol is called Japanese Format. I do admit that I don't know exactly who developed it. What I do know is that it was used in my late VCR produced by Sanyo and was marketed under the name of Fisher. NEC manufactured the remote control IC. This description was taken from my VCR's service manual. Those were the days, when service manuals were filled with useful information!

#### Features:

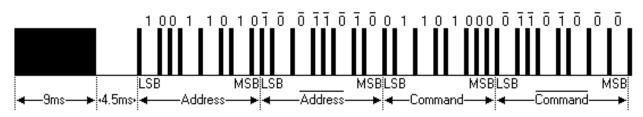
- 8 bit address and 8 bit command length.
- Extended mode available, doubling the address size.
- Address and command are transmitted twice for reliability.
- Pulse distance modulation.
- Carrier frequency of 38kHz.
- Bit time of 1.125ms or 2.25ms.

#### Modulation:

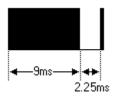


The NEC protocol uses pulse distance encoding of the bits. Each pulse is a 560µs long 38kHz carrier burst (about 21 cycles). A logical "1" takes 2.25ms to transmit, while a logical "0" is only half of that, being 1.125ms. The recommended carrier duty-cycle is 1/4 or 1/3.

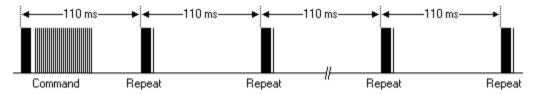
#### **Protocol:**



The picture above shows a typical pulse train of the NEC protocol. With this protocol the LSB is transmitted first. In this case Address 59andCommand16 is transmitted. A message is started by a 9ms AGC burst, which was used to set the gain of the earlier IR receivers. This AGC burst is then followed by a 4.5ms space, which is then followed by the Address and Command. Address and Command are transmitted twice. The second time all bits are inverted and can be used for verification of the received message. The total transmission time is constant because every bit is repeated with its inverted length. If you're not interested in this reliability you can ignore the inverted values, or you can expand the Address and Command to 16 bits each! Keep in mind that one extra 560µs burst has to follow at the end of the message in order to be able to determine the value of the last bit.



A command is transmitted only once, even when the key on the remote control remains pressed. Every 110ms a repeat code is transmitted for as long as the key remains down. This repeat code is simply a 9ms AGC pulse followed by a 2.25ms space and a 560µs burst.

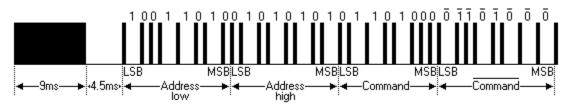


#### **Extended NEC protocol:**

The NEC protocol is so widely used that soon all possible addresses were used up. By sacrificing the address redundancy the address range was extended from 256 possible values to approximately 65000 different values. This way the address range was extended from 8 bits to 16 bits without changing any other property of the protocol.

By extending the address range this way the total message time is no longer constant. It now depends on the total number of 1's and 0's in the message. If you want to keep the total message time constant you'll have to make sure the number 1's in the address field is 8 (it automatically means that the number of 0's is also 8). This will reduce the maximum number of different addresses to just about 13000.

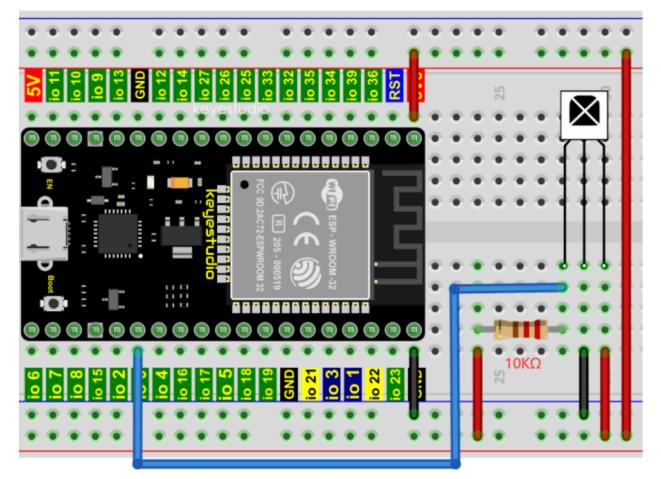
The command redundancy is still preserved. Therefore each address can still handle 256 different commands.



Keep in mind that 256 address values of the extended protocol are invalid because they are in fact normal NEC protocol addresses. Whenever the low byte is the exact inverse of the high byte it is not a valid extended address.

# 7.35.4 Decoded infrared signal

We connect the infrared receiving element to the ESP32, according to the wiring diagram below:



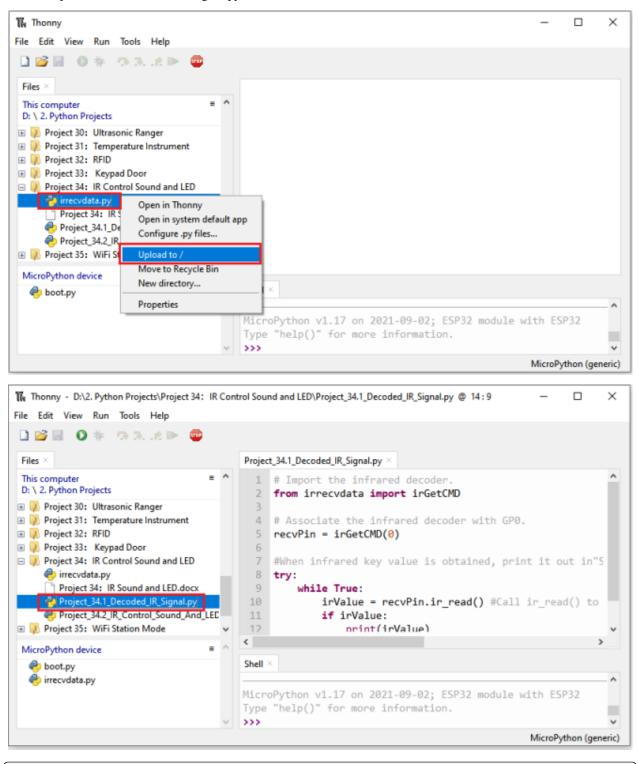
Codes used in this tutorial are saved in "2. Python Projects". If you haven't downloaded the code file, please click on the link to download it:Download Python Codes

If you haven't downloaded the code file, please click on the link to download it:Download Python Codes

2. Python Projects		—	$\Box$ $\times$
Share View			~ ?
→ This PC → Software (D:) → 2. Pyth	on Projects 🗸 🗸	, Search	2. Python
Name	Date modified	Туре	Size ^
Project 00: Boot	2/17/2022 10:21 AM	File folder	
	2/17/2022 10:21 AM	File folder	
Project 02: Turn On LED	2/17/2022 11:10 AM	File folder	
Project 03: LED Flashing	2/17/2022 11:12 AM	File folder	
Project 04: Breathing LED	2/17/2022 10:21 AM	File folder	

Open "Thonny", click "This computer"→"D:"→"2. Python Projects"→"Project 34IR Control Sound and LED".

Select "irrecvdata.py", click your mouse to select "Upload to /", wait for "irrecvdata.py" to be uploaded to ESP32, and click "Project\_34.1\_Decoded\_IR\_Signal.py".



## Import the infrared decoder.
from irrecvdata import irGetCMD

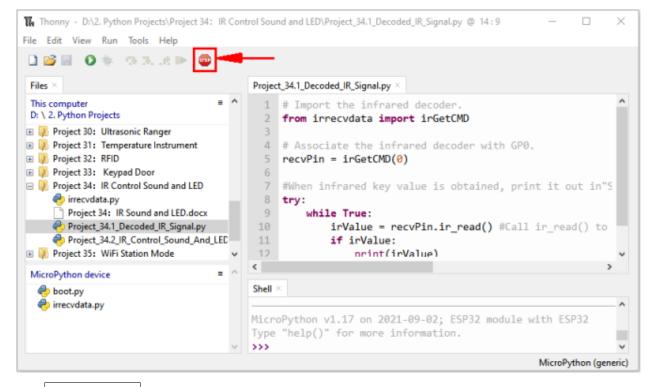
(continues on next page)

```
## Associate the infrared decoder with GP0.
recvPin = irGetCMD(0)
#When infrared key value is obtained, print it out in"Shell".
try:
    while True:
        irValue = recvPin.ir_read() #Call ir_read() to read the value of the pressed key.
        →and assign it to IRValue.
        if irValue:
            print(irValue)
except:
        pass
```

media/27451c8a9c13e29d02bc0f5831cfaf1f.png

(continued from previous page)

Make sure the ESP32 has been connected to the computer, click "Stop/Restart backend".

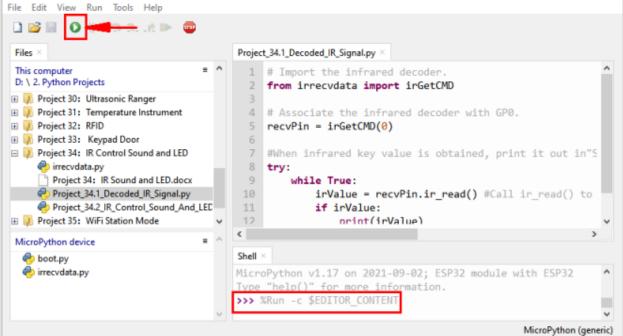


media/da852227207616ccd9aff28f19e02690.png

Click "Run current script", the code starts to be executed and you'll see that aim the infrared remote control transmitter at the infrared receiving head, press the button on the infrared controller, and the "Shell" window of Thonny IDE prints the current received key code values.

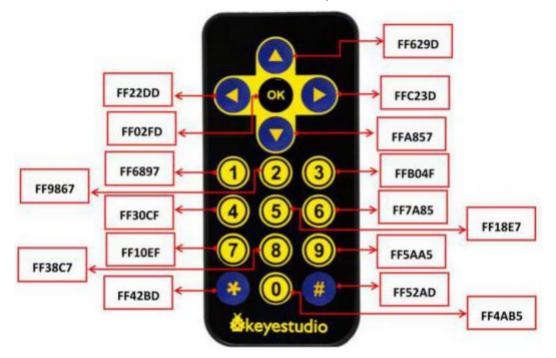
 $\times$ 





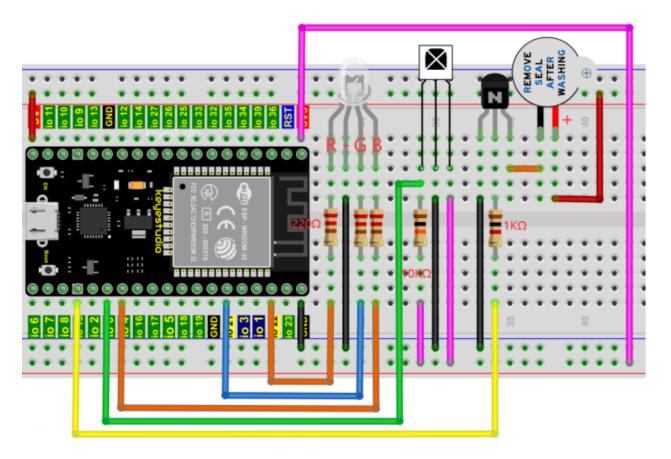
>>> %Run -c \$EDITOR_CONTENT	^
0xff629d	
0xff22dd	
0xffa857	
0xff02fd	
0xffc23d	
0xff6897	
0xff9867	
0xffb04f	
0xff30cf	
0xff18e7	
0xff7a85	
0xff10ef	
0xff38c7	
0xff5aa5	
0xff42bd	
0xff4ab5	
0xff52ad	
	~

Shell ×



Write down the code associated with each button, because you will need that information later.

7.35.5 Wiring diagram of the infrared remote control



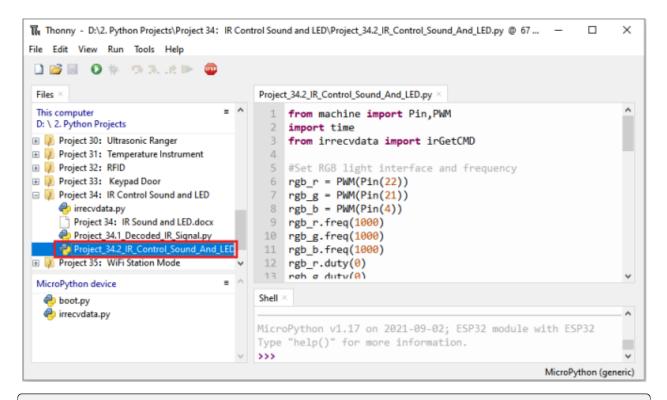
# 7.35.6 Project code

Codes used in this tutorial are saved in "2. Python Projects". If you haven't downloaded the code file, please click on the link to download it:Download Python Codes

2. Python Projects		_	$\Box$ $\times$
Share View			~ ?
→ This PC → Software (D:) → 2. Pyth	non Projects 🗸 🗸	,	. Python
Name	Date modified	Туре	Size ^
	2/17/2022 10:21 AM	File folder	
Project 01: Hello World	2/17/2022 10:21 AM	File folder	
	2/17/2022 11:10 AM	File folder	
Project 03: LED Flashing	2/17/2022 11:12 AM	File folder	
Project 04: Breathing LED	2/17/2022 10:21 AM	File folder	

Open"Thonny", click "This computer"  $\rightarrow$  "D:"  $\rightarrow$  "2. Python Projects"  $\rightarrow$  "Project 34IR Control Sound and LED". Select "irrecvdata.py", click your mouse to select "Upload to /", wait for "irrecvdata.py" to be uploaded to ESP32, and click "Project\_34.2\_IR\_Control\_Sound\_And\_LED.py".

The Thonny		-		×
File Edit View Run To	pols Help			
2 ≥ ■ 0 * ○	) 3. Lé 🕪 🐵			
Files ×				
This computer D: \ 2. Python Projects	= ^			
	-			
Project 33: Keypad				
🖃 🚺 Project 34: IR Contro	ol Sound and LED			
Project 34: IRS	Open in Thonny			
Project_34.1_De	Open in system default app			
Project_34.2_IR	Configure .py files			
🗉 🚶 Project 35: WiFi St	Upload to /			
MicroPython device	Move to Recycle Bin			_
eboot.py -	New directory			
	Properties			^
	MicroPython v1.17 on 2021-09-02; ESP32 module wi	th ES	P32	
	Type "help()" for more information.			
	V >>>	1.C		*
		MICTOPY	thon (ge	neric)



```
from machine import Pin, PWM
import time
from irrecvdata import irGetCMD
#Set RGB light interface and frequency
rgb_r = PWM(Pin(22))
rgb_g = PWM(Pin(21))
rgb_b = PWM(Pin(4))
rgb_r.freq(1000)
rgb_g.freq(1000)
rgb_b.freq(1000)
rgb_r.duty(0)
rgb_g.duty(≬)
rgb_b.duty(♥)
## Initialize the buzzer pin
buzzer=Pin(15, Pin.OUT)
#Configure infrared receiving pin and library
recvPin = irGetCMD(0)
while True:
    irValue = recvPin.ir read() # Read remote control data
## Determine whether there is a button that meets the needs
   if irValue:
        print(irValue)
        buzzer.value(1)
        time.sleep(0.1)
        buzzer.value(0)
        if irValue == '0xff6897':
                                    #1
```

(continues on next page)

(continued from previous page)

```
rgb_r.duty(1023)
  rgb_g.duty(≬)
  rgb_b.duty())
  print('1')
elif irValue == '0xff9867': #2
    rgb_r.duty())
    rgb_g.duty(1023)
    rgb_b.duty())
    print('2')
elif irValue == '0xffb04f': #3
    rgb_r.duty(♥)
    rgb_g.duty())
    rgb_b.duty(1023)
    print('3')
elif irValue == '0xff30cf': #4
    rgb_r.duty(1023)
    rgb_g.duty(1023)
    rgb_b.duty(≬)
    print('4')
elif irValue == '0xff18e7': #5
    rgb_r.duty(1023)
    rgb_g.duty(≬)
    rgb_b.duty(1023)
    print('5')
elif irValue == '0xff7a85': #6
    rgb_r.duty(0)
    rgb_g.duty(1023)
    rgb_b.duty(1023)
    print('6')
elif irValue == '0xff10ef': #7
    rgb_r.duty(1023)
    rgb_g.duty(1023)
    rgb_b.duty(1023)
    print('7')
else:
    rgb_r.duty(≬)
    rgb_g.duty())
    rgb_b.duty())
```

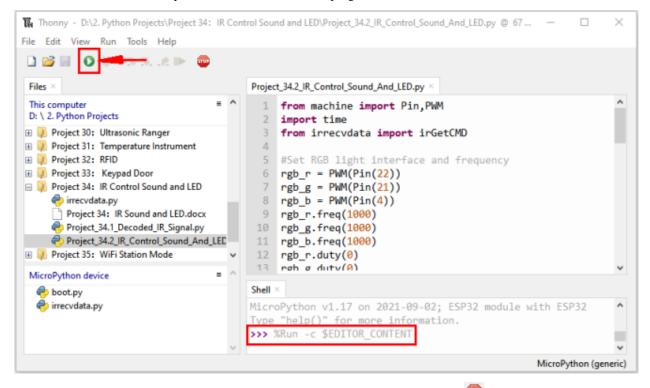
## 7.35.7 Project result

Make sure the ESP32 has been connected to the computer, click "Stop/Restart backend".

🗋 😂 🖩 🛛 🐐 🗇 3e 🕨 🥮 🔫	Projec	t_34.2_IR_Control_Sound_And_LED.py ×	
This computer       =         D: \ 2. Python Projects         Project 30: Ultrasonic Ranger         Project 31: Temperature Instrument         Project 32: RFID         Project 33: Keypad Door         Project 34: IR Control Sound and LED         Project 34: IR Sound and LED.docx         Project 34: IR Sound and LED.docx         Project 34.2 IR_Control_Sound_And_LEE         Project 35: WiFi Station Mode	1 2 3 4 5 6 7 8 9 10 11 12 13	<pre>from machine import Pin,PWM import time from irrecvdata import irGetCMD #Set RGB light interface and frequency rgb_r = PWM(Pin(22)) rgb_g = PWM(Pin(21)) rgb_b = PWM(Pin(4)) rgb_r.freq(1000) rgb_g.freq(1000) rgb_b.freq(1000) rgb_b.freq(1000) rgb_r.duty(0) rgb_g duty(0)</pre>	
MicroPython device = ^			
<ul> <li>♦ boot.py</li> <li>♦ irrecvdata.py</li> </ul>		<pre>oPython v1.17 on 2021-09-02; ESP32 module with ESP32 "help()" for more information.</pre>	- ^

Click "Run current script", the code starts to be executed and you'll see that press the 1 to 7 key of the infrared remote controller, the buzzer will sound once, and the RGB light will be red, green, blue, yellow ,red, blue ,green and white respectively. Press another key (except 1 to 7 key), and the RGB light will go off.

Press"Ctrl+C"or click <sup>22</sup> "Stop/Restart backend" to exit the program.



Note When the code is running, the following prompt appears, you just need to click "Stop/Restart backend", then

click<sup>O</sup>"Run current script" to make the code run again.

>>> %Run -c \$EDITOR_CONTENT
0xff4ab5 0xff4ab5
<pre>Traceback (most recent call last):     File "<stdin>", line 48, in <module></module></stdin></pre>
File "irrecvdata.py", line 32, in ir read IndexError: list index out of range

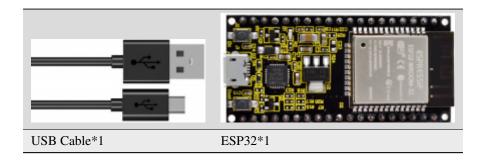
(Note: Before use, we need to remove the plastic sheet from the bottom of the infrared remote controller.)

# 7.36 Project 35WiFi Station Mode

## 7.36.1 Introduction

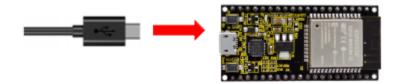
ESP32 has three different WiFi operating modes : Station modeAP mode and AP+Station mode. All WiFi programming projects must be configured with WiFi operating mode before using WiFi, otherwise WiFi cannot be used. In this project, we will learn about ESP32's WiFi Station mode.

# 7.36.2 Components



# 7.36.3 Project wiring

Connect the ESP32 to the USB port on your computer using a USB cable.



# 7.36.4 Component knowledge

**Station mode:** When ESP32 selects Station mode, it acts as a WiFi client. It can connect to the router network and communicate with other devices on the router via WiFi connection. As shown below, the PC is connected to the router, and if ESP32 wants to communicate with the PC, it needs to be connected to the router.

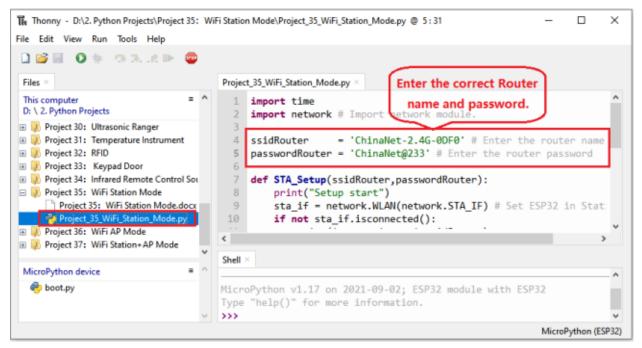


## 7.36.5 Project code

Codes used in this tutorial are saved in "2. Python Projects". If you haven't downloaded the code file, please click on the link to download it:Download Python Codes

2. Python Projects		_	
Share View			~ ?
→ This PC → Software (D:) → 2. Pyth	on Projects 🗸 さ		. Python
Name	Date modified	Туре	Size ^
	2/17/2022 10:21 AM	File folder	
	2/17/2022 10:21 AM	File folder	
	2/17/2022 11:10 AM	File folder	
Project 03: LED Flashing	2/17/2022 11:12 AM	File folder	
Project 04: Breathing LED	2/17/2022 10:21 AM	File folder	

Open"Thonny", click "This computer"  $\rightarrow$  "D:"  $\rightarrow$  "2. Python Projects"  $\rightarrow$  "Project 35WiFi Station Mode", and double left-click "Project\_35\_WiFi\_Station\_Mode.py".



```
import time
import network # Import network module.
ssidRouter = 'ChinaNet-2.4G-0DF0' # Enter the router name
passwordRouter = 'ChinaNet@233' # Enter the router password
def STA_Setup(ssidRouter,passwordRouter):
    print("Setup start")
    sta_if = network.WLAN(network.STA_IF) # Set ESP32 in Station mode.
(continues on next page)
```

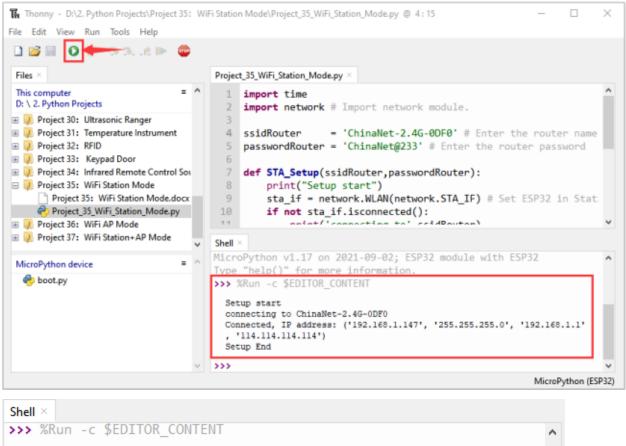
(continued from previous page)

```
if not sta_if.isconnected():
       print('connecting to',ssidRouter)
  # Activate ESP32's Station mode, initiate a connection request to the router
  # and enter the password to connect.
        sta_if.active(True)
        sta_if.connect(ssidRouter,passwordRouter)
  #Wait for ESP32 to connect to router until they connect to each other successfully.
                                                                                          <u>ل</u>
4
        while not sta_if.isconnected():
            pass
  # Print the IP address assigned to ESP32-WROVER in "Shell".
   print('Connected, IP address:', sta_if.ifconfig())
   print("Setup End")
try:
    STA_Setup(ssidRouter,passwordRouter)
except:
    sta_if.disconnect()
```

# 7.37 Project result

Because the names and passwords of routers in various places are different, before the code runs, users need to enter the correct router's name and password in the box as shown in the illustration above.

After making sure the router name and password are entered correctly, click <sup>Q</sup> "Run current script", the code starts to be executed and wait for ESP32 to connect to your router and print the IP address assigned by the router to ESP32 in the "Shell" window of Thonny IDE.



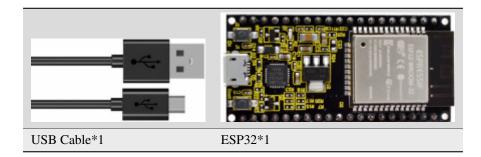
```
>>> %Run -C $EDITOR_CONTENT
Setup start
connecting to ChinaNet-2.4G-0DF0
Connected, IP address: ('192.168.1.147', '255.255.255.0', '192.168.1.1'
, '114.114.114.114')
Setup End
>>>
```

# 7.38 Project 36WiFi AP Mode

# 7.39 Introduction

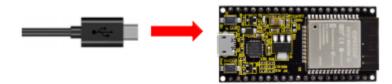
ESP32 has three different WiFi operating modes : Station modeAP mode and AP+Station mode. All WiFi programming projects must be configured with WiFi operating mode before using WiFi, otherwise WiFi cannot be used. In this project, we will learn about ESP32's WiFi AP mode.

## 7.39.1 Components



# 7.40 Project wiring

Connect the ESP32 to the USB port on your computer using a USB cable.



## 7.40.1 Component knowledge

#### AP mode :

When ESP32 selects AP mode, it creates a hotspot network that is separated from the Internet and waits for other WiFi devices to connect. As shown in the figure below, ESP32 is used as a hotspot. If a mobile phone or PC wants to communicate with ESP32, it must be connected to the hotspot of ESP32. Only after a connection is established with ESP32 can they communicate.



# 7.40.2 Project code

Codes used in this tutorial are saved in "2. Python Projects". If you haven't downloaded the code file, please click on the link to download it:Download Python Codes

2. Python Projects		_	$\Box$ $\times$
Share View			~ ?
→ This PC → Software (D:) → 2. Pyth	non Projects 🗸 🗸 🗸	ට 🔎 Searc	h 2. Python
Name	Date modified	Туре	Size ^
Project 00: Boot	2/17/2022 10:21 AM	I File folder	
	2/17/2022 10:21 AM	1 File folder	
	2/17/2022 11:10 AM	1 File folder	
Project 03: LED Flashing	2/17/2022 11:12 AM	1 File folder	
Project 04: Breathing LED	2/17/2022 10:21 AM	1 File folder	

 $\label{eq:constraint} Open ``Thonny" click" This computer" \rightarrow ``D:" \rightarrow ``2. Python Projects" \rightarrow ``Project 36WiFi AP Mode", and double left-click "Project_36_WiFi_AP_Mode.py".$ 

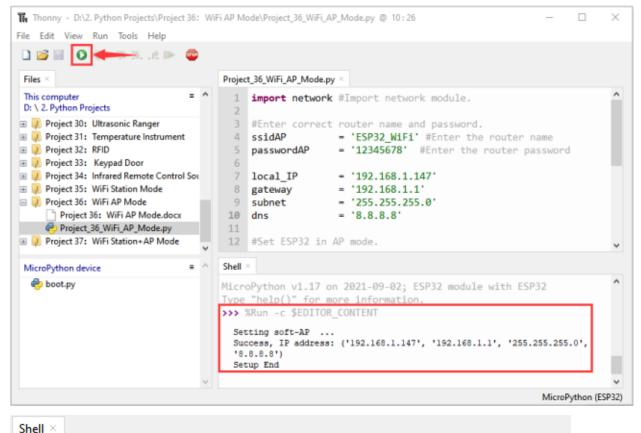
Thonny - D:\2. Python Projects\Project 36: W	ViFi AP M	ode\Project 36 WiFi AP Mode.pv @ 10:26 -	ı x
ile Edit View Run Tools Help			
🗋 🐸 📓 🔕 🌞 🧐 🔭 🖄 👘 🥶			
Files ×	Projec	t_36_WiFi_AP_Mode.py ×	
This computer = ^ D: \ 2. Python Projects	1	import network #Import network module.	^
Image: Project 30: Ultrasonic Ranger	3	#Enter correct router name and pressure.	
Project 31: Temperature Instrument	-4	<pre>ssidAP = 'ESP32_WiFi' #Enter the router name</pre>	
🗉 🔑 Project 32: RFID	5	passwordAP = '12345678' #Enter the router password	
Project 33: Keypad Door	6		
Project 34: Infrared Remote Control Sou	7	local_IP = '192.168.1.147'	
Project 35: WiFi Station Mode	8	gateway = '192.168.1.1'	
Project 36: WiFi AP Mode	9	subnet = '255.255.255.0'	
Project 36: WiFi AP Mode.docx	10	dns = '8.8.8.8'	
Project_36_WiFi_AP_Mode.py	11	AC-A CCD20 in AD and	
Project 37: WiFi Station+AP Mode     V		#Set ESP32 in AP mode.	
MicroPython device = ^	13	<pre>ap_if = network.WLAN(network.AP_IF)</pre>	~
loot.py			
- bootpy	Shell	K.	
			^
		oPython v1.17 on 2021-09-02; ESP32 module with ESP32	
		"help()" for more information.	
~	>>>		~
		MicroPytho	on (ESP32

```
import network #Import network module.
#Enter correct router name and password.
ssidAP
          = 'ESP32_WiFi' #Enter the router name
             = '12345678' #Enter the router password
passwordAP
local_IP
              = '192.168.1.147'
gateway
             = '192.168.1.1'
subnet
              = '255.255.255.0'
              = '8.8.8.8'
dns
#Set ESP32 in AP mode.
ap_if = network.WLAN(network.AP_IF)
def AP_Setup(ssidAP,passwordAP):
   ap_if.ifconfig([local_IP,gateway,subnet,dns])
   print("Setting soft-AP ... ")
   ap_if.config(essid=ssidAP,authmode=network.AUTH_WPA_WPA2_PSK, password=passwordAP)
   ap_if.active(True)
   print('Success, IP address:', ap_if.ifconfig())
   print("Setup End\n")
try:
   AP_Setup(ssidAP,passwordAP)
except:
   print("Failed, please disconnect the power and restart the operation.")
   ap_if.disconnect()
```

# 7.41 Project result

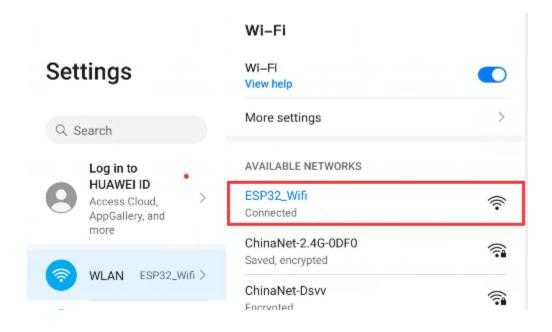
Before the code runs, you can make any changes to the AP name and password for ESP32 in the box as shown in the illustration above. Of course, you can leave it alone by default.

Click "Run current script", the code starts to be executed and open the AP function of ESP32 and print the access point information in the "Shell" window of Thonny IDE.



```
>>> %Run -c $EDITOR_CONTENT
Setting soft-AP ...
Success, IP address: ('192.168.1.147', '192.168.1.1', '255.255.255.0',
'8.8.8.8')
Setup End
>>>
```

Turn on the WiFi scanning function of your phone, and you can see the ssid\_AP on ESP32, which is called "ESP32\_Wifi" in this code. You can enter the password "12345678" to connect it or change its AP name and password by modifying Code.

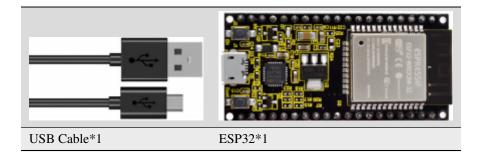


# 7.42 Project 37WiFi Station+AP Mode

## 7.42.1 Introduction

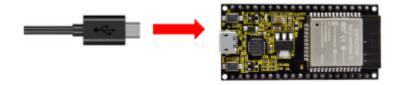
ESP32 has three different WiFi operating modes : Station modeAP mode and AP+Station mode. All WiFi programming projects must be configured with WiFi operating mode before using WiFi, otherwise WiFi cannot be used. In this project, we will learn ESP32's WiFi Station+AP mode.

# 7.42.2 Components



# 7.42.3 Project wiring

Connect the ESP32 to the USB port on your computer using a USB cable.



## 7.42.4 Component knowledge

**AP+Station mode:** In addition to AP mode and Station mode, ESP32 can also use AP mode and Station mode at the same time. This mode contains the functions of the previous two modes. Turn on ESP32's Station mode, connect it to the router network, and it can communicate with the Internet via the router. At the same time, turn on its AP mode to create a hotspot network. Other WiFi devices can choose to connect to the router network or the hotspot network to communicate with ESP32.

## 7.42.5 Project code

Codes used in this tutorial are saved in "2. Python Projects". If you haven't downloaded the code file, please click on the link to download it:Download Python Codes

2. Python Projects		_	$\Box$ $\times$
Share View			~ ?
→ This PC → Software (D:) → 2. Pyth	on Projects 🛛 🗸	ට 🔎 Search	2. Python
Name	Date modified	Туре	Size ^
Project 00: Boot	2/17/2022 10:21 AN	I File folder	
Project 01: Hello World	2/17/2022 10:21 AN	1 File folder	
Project 02: Turn On LED	2/17/2022 11:10 AN	File folder	
Project 03: LED Flashing	2/17/2022 11:12 AN	File folder	
Project 04: Breathing LED	2/17/2022 10:21 AN	File folder	

Open"Thonny" click"This computer"  $\rightarrow$  "D:"  $\rightarrow$  "2. Python Projects"  $\rightarrow$  "Project 37WiFi Station+AP Mode" and double left-click "Project\_37\_WiFi\_Station+AP\_Mode.py".

1 😂 🖩 🛛 🐐 🖘 3e 🕨 🥶 -	ject_37_WiFi_Station+AP_Mode.py	Please enter the correct names and passwords
This computer = ^ D: \ 2. Python Projects		t ne worof Router and AP.
Project 30: Ultrasonic Ranger     Project 31: Temperature Instrument     Project 32: RFID     Project 33: Keypad Door     Project 34: Infrared Remote Control Soc	passwordRouter = 'Chi ssidAP = 'ESP passwordAP = '123	InaNet-2.4G-0DF0' #Enter the router name InaNet@233' #Enter the router password P32_WIF1'#Enter the AP name 845678' #Enter the AP password
Project 35: WiFi Station Mode Project 36: WiFi AP Mode Project 37: WiFi Station+AP Mode Project 37: WiFi Station+AP Mode.c Project 37: WiFi Station+AP Mode.c	gateway = '192	2.168.4.147' 2.168.1.1' 5.255.255.0' 8 8 8'
MicroPython device = ^	×	
🐣 boot.py	roPython v1.17 on 2021- De "help()" for more inf	09-02; ESP32 module with ESP32 formation.

ap\_if.config(essid=ssidAP,authmode=network.AUTH\_WPA\_WPA2\_PSK, password=passwordAP)

```
import network #Import network module.
              = 'ChinaNet-2.4G-0DF0' #Enter the router name
ssidRouter
passwordRouter = 'ChinaNet@233' #Enter the router password
ssidAP
              = 'ESP32_WiFi'#Enter the AP name
passwordAP
              = '12345678' #Enter the AP password
local_IP
              = '192.168.4.147'
              = '192.168.1.1'
gateway
subnet
              = '255.255.255.0'
dns
              = '8.8.8.8'
sta_if = network.WLAN(network.STA_IF)
ap_if = network.WLAN(network.AP_IF)
def STA_Setup(ssidRouter,passwordRouter):
   print("Setting soft-STA ... ")
   if not sta_if.isconnected():
       print('connecting to',ssidRouter)
        sta_if.active(True)
        sta_if.connect(ssidRouter,passwordRouter)
       while not sta_if.isconnected():
           pass
   print('Connected, IP address:', sta_if.ifconfig())
   print("Setup End")
def AP_Setup(ssidAP,passwordAP):
    ap_if.ifconfig([local_IP,gateway,subnet,dns])
```

```
(continues on next page)
```

print("Setting soft-AP ... ")

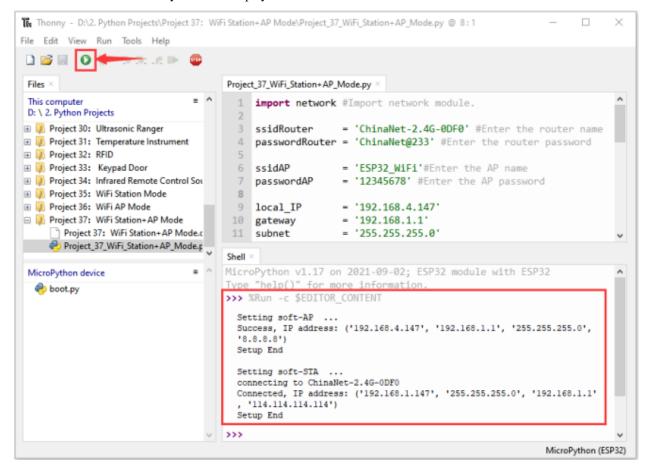
(continued from previous page)

```
ap_if.active(True)
print('Success, IP address:', ap_if.ifconfig())
print("Setup End\n")
try:
    AP_Setup(ssidAP,passwordAP)
    STA_Setup(ssidRouter,passwordRouter)
except:
    sta_if.disconnect()
    ap_if.idsconnect()
```

## 7.42.6 Project result

It is analogous to Project 35 and project 36. Before running the code, you need to modify ssidRouter, passwordRouter, ssidAP and passwordAP shown in the box of the illustration above.

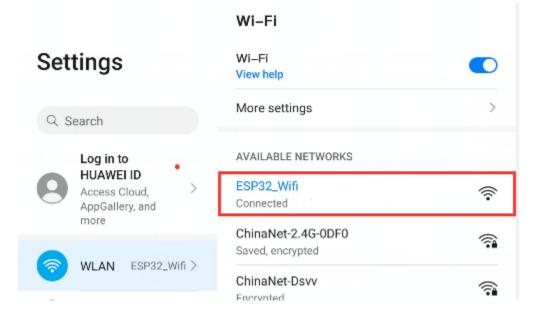
After making sure that the code is modified correctly, click <sup>()</sup> "Run current script" the code starts to be executed and the "Shell" window of Thonny IDE will display as follows:



```
\mathbf{Shell} \ \times
```

```
MicroPython v1.17 on 2021-09-02; ESP32 module with ESP32
Type "help()" for more information.
>>> %Run -c $EDITOR_CONTENT
Setting soft-AP ...
Success, IP address: ('192.168.4.147', '192.168.1.1', '255.255.255.0',
'8.8.8.8')
Setup End
Setting soft-STA ...
connecting to ChinaNet-2.4G-0DF0
Connected, IP address: ('192.168.1.147', '255.255.255.0', '192.168.1.1'
, '114.114.114.114')
Setup End
>>>
```

Turn on the WiFi scanning function of your phone, and you can see the ssidAP on ESP32.



CHAPTER

EIGHT

# **GETTING STARTED WITH C (RASPBERRY PI)**

About raspberry pi

Raspberry Pi is a card computer whose official system is Raspberry Pi OS, and can be installed on the Raspberry Pi, such as: ubuntu, Windows IoT. Raspberry Pi can be used as a personal server, performing camera monitoring and recognition, as well as voice interaction by connecting a camera and a voice interactive assistant. Also, Raspberry Pi leads out 40Pin pins that can be connected to various sensors and control LEDs, motors, etc. This can be used to make a robot with a Raspberry Pi.

# 8.1 Install the Raspberry Pi OS System

## 8.1.1 1. Tools needed for the Raspberry Pi system

#### 1.1. Hardware Tool

1Raspberry Pi 4B/3B/2B 2Above 16G TFT Memory Card 3Card Reader 4Computer and other parts

#### 1.2. Software tools that need to be installed

Windows System

#### **1Install putty**

Download linkhttps://www.chiark.greenend.org.uk/~sgtatham/putty/



# **PuTTY: a free SSH and Telnet client**

Home | FAQ | Feedback | Licence | Updates | Mirrors | Keys | Links | Team Download: Stable · Snapshot | Docs | Changes | Wishlist

PuTTY is a free implementation of SSH and Telnet for Windows and Unix platforms, along with an xterm terminal emulator. It is written and maintained primarily by <u>Simon Tatham</u>.

The latest version is 0.74 **Download it here**.

**LEGAL WARNING**: Use of PuTTY, PSCP, PSFTP and Plink is illegal in countries where encryption is outlawed. We believe it is legal to use PuTTY, PSCP, PSFTP and Plink in England and Wales and in many other countries, but we are not lawyers, and so if in doubt you should seek legal advice before downloading it. You may find useful information at <u>cryptolaw.org</u>, which collects information on cryptography laws in many countries, but we can't vouch for its correctness.

Use of the Telnet-only binary (PuTTYtel) is unrestricted by any cryptography laws.

### Latest news

### 2020-11-22 Primary git branch renamed

The primary branch in the PuTTY git repository is now called main, instead of git's default of master. For now, both branch names continue to exist, and are kept automatically in sync by a symbolic-ref on the server. In a few months' time, the alias master will be withdrawn.

P Dov	vnload Pi	uTTY: latest release (C × +			
$\leftarrow \rightarrow$	G	chiark.greenend.org.uk/~sgtatham/putty/latest.html	Q	☆	θ

## Download PuTTY: latest release (0.74)

<u>Home</u> | FAQ | Feedback | Licence | Updates | Mirrors | Keys | Links | Team Download: Stable - Snapshot | Docs | Changes | Wishlist

This page contains download links for the latest released version of PuTTY. Currently this is 0.74, released on 2020-06-27.

When new releases come out, this page will update to contain the latest, so this is a good page to bookmark or link to. Alternatively, here is a <u>permanent link to the 0.74 release</u>.

Release versions of PuTTY are versions we think are reasonably likely to work well. However, they are often not the most up-to-date version of the code available. If you have a problem with this release, then it might be worth trying out the <u>development snapshots</u>, to see if the problem has already been fixed in those versions.

Package file	25			
	want one of these. They include versions of ther you want the 32-bit or the 64-bit version			
MSI ('Windo	ws Installer')			
32-bit:	putty-0.74-installer.msi	(or by FTP)	(signature)	
64-bit:	<u>putty-64bit-0.74-installer.msi</u>	<u>(or by FTP)</u>	(signature)	
Unix source a	rchive			
.tar.gz:	<u>putty-0.74.tar.gz</u>	(or by FTP)	(signature)	

1. After downloading the package file putty-64bit-0.74-installer , double-click it and tap "Next".

	🕌 PuTTY release 0.74 (64-bit)	Setup	_		×
	đ	Welcome to the Pu (64-bit) Setup Wiza		0.74	
		The Setup Wizard allows you release 0.74 (64-bit) feature or to remove it from your cor Cancel to exit the Setup Wiza	s are installed on nputer. Click Nex	your comput	ter or
	Ż				
		Back	Next	Cancel	
2.	Click "Next".				
	🔀 PuTTY release 0.74 (64-bit)	Setup	_		×
	Destination Folder Click Next to install to the d	efault folder or click Change to c	hoose another.	Ĺ	
	Install PuTTY release 0.74 (6	4-bit) to:			
	C:\Program Files\PuTTY\				
	Change				
		Back	Next	Cancel	

3. Select"Install Putty files" and click"Install".

🖟 PuTTY release 0.74 (64-bit) Setup	_		×
Product Features Select the way you want features to be installed.		Ē	ł
Add shortcut to PuTTY files Add shortcut to PuTTY on the Desktop Put install directory on the PATH for command prompts Associate .PPK files with PuTTYgen and Pageant			
This feature requires 3914KB on your hard drive.			
Back 💙 Install		Cance	<u>!</u>

4. After a few seconds, the installation is complete, click "Finish".

PuTTY release 0.74 (64-bit)	Setup – 🗆 🗙
ð	Completed the PuTTY release 0.74 (64-bit) Setup Wizard
	Click the Finish button to exit the Setup Wizard.
	View README file
	Back Finish Cancel

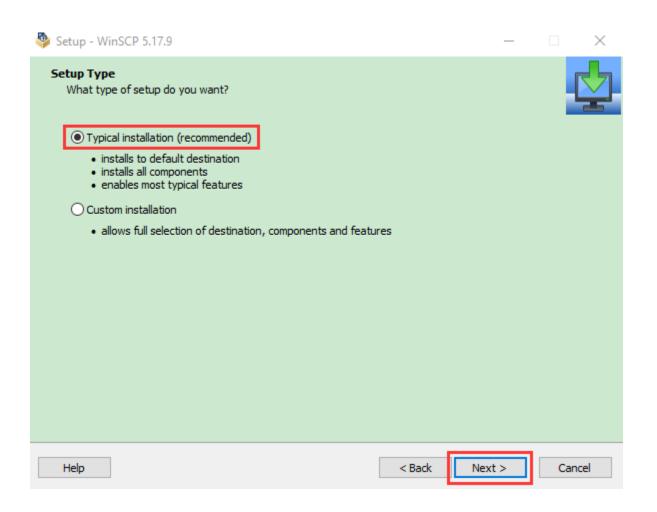
## 2Remote Login software -WinSCP

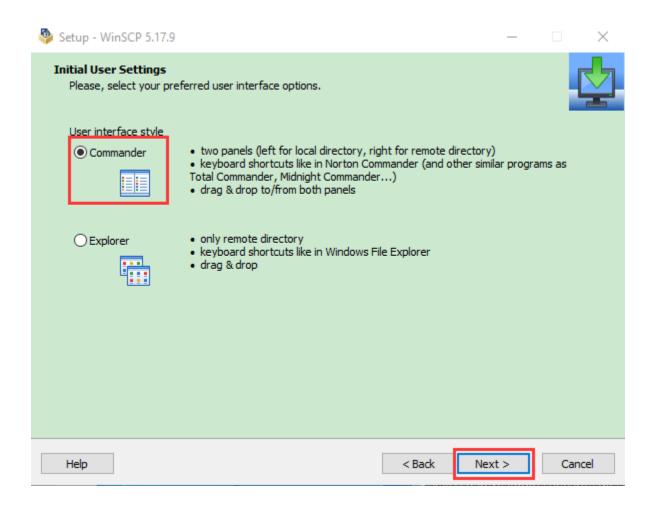
Download linkhttps://winscp.net/eng/download.php After downloading the WinSCP software file, double-click the WinSCP software file WinSCP-5.17.9-Setup.exe and click 🍄 WinSCP-5.17.9-Setup.exe

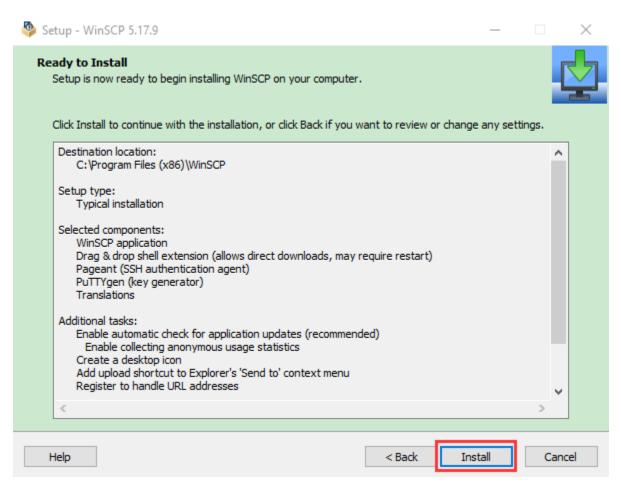
Select S	Setup Install Mode	$\times$
•	Select install mode	
Ť	WinSCP can be installed for all users (requires administrative privileges), or for you only.	
	Install for all users (recommended)	
	$\rightarrow$ Install for me only	
	Cancel	

Click"Accept"then select the appropriate option and click"Next", then Click"Install".

Setup - WinSCP 5.17.9	_		$\times$
License Agreement Please read the following important information before continuing.			¢.
Please read the following License Agreement. You must accept the terms of this agreement b continuing with the installation.	efore		
You can also review this license and further details online at:		~	
https://winscp.net/eng/docs/license			
A. GNU General Public License B. License of WinSCP Icon Set C. Privacy Policy			
A. GNU GENERAL PUBLIC LICENSE Version 3, 29 June 2007			
Copyright (C) 2007 Free Software Foundation, Inc. < <u>https://www.fsf.org/</u> > Everyone is permitted to copy and distribute verbatim copies of this license document, but d is not allowed.	hanging	it V	,
Help Accept >		Car	ncel







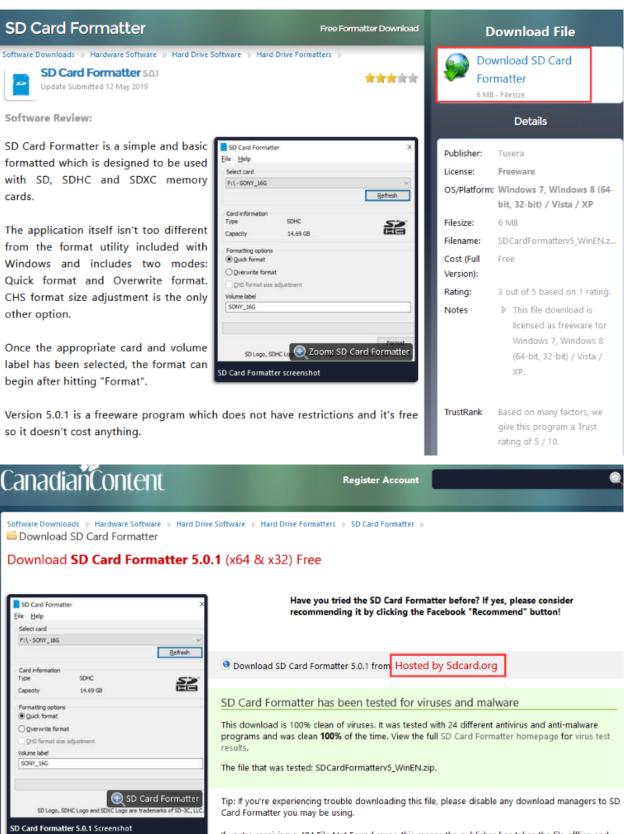
After a few seconds, the installation is complete, click "Finish".



### 3Format TFT card tool- SD Card Formatter

#### Download link

 $http://www.canadiancontent.net/tech/download/SD\_Card\_Formatter.html$ 



has not updated their links with us for SD Card Formatter. Please do drop us a note in the event of a missing file.

Unzip the SDCardFormatterv5\_WinEN package and double-click the SD Card Formatter file SD Card Formatter 5.0.1 Setup.exe to run it.

SD Card Formatter - InstallShield Wizard

	Preparing to Install SD Card Formatter Setup is preparing the InstallShield Wizard, which will guide you through the program setup process. Please wait. Extracting: SD Card Formatter Setup.msi
I accent to	Cancel
Click"Next"select" Taccept u	he terms in the license agreement "and click"Next".
	Welcome to the InstallShield Wizard for SD Card Formatter
	The InstallShield(R) Wizard will install SD Card Formatter on your computer. To continue, click Next.
	WARNING: This program is protected by copyright law and international treaties.
	< Back Next > Cancel

🔀 SD Card Formatter - InstallShield Wiza	rd		×
License Agreement Please read the following license agreeme	ent carefully.		と
END USER LIC	CENSE AGREEM	MENT	^
NOTICE: BY DOWNLOADING, INSTALLING OR USIN ENTERING INTO THIS AGREEMENT AGRE IF YOU DO NOT AGREE WITH ANY OF T OR USE THE PRODUCT; PROMPTLY RET SDA OR YOUR SDA DISTRIBUTOR. IF YO ACQUIRE ANY LICENSE TO USE THE PROD	EES TO BE BOU THESE TERMS, TURN (IF APPI OU REJECT TH	JND BY THE FOLLOV , DO NOT DOWNLO LICABLE) THE PROD	VING TERMS. DAD, INSTALL, DUCT TO THE
I accept the terms in the license agreement I do not accept the terms in the license agreement			Print
InstallShield			
[	< Back	Next >	Cancel
Click "Next" again, and then click "Install"			
🔀 SD Card Formatter - InstallShield Wiza	rd		$\times$
Destination Folder Click Next to install to this folder, or click	Change to insta	Il to a different folde	

$\triangleright$	Install SD Card Formatter to: C:\Program Files (x86)\SDA\SD Card Fo	ormatter\		Change
				changern
InstallShield -				
	< Ba	ack	Next >	Cancel

BD Card Formatter - InstallShield Wizard	×
Ready to Install the Program	4.
The wizard is ready to begin installation.	
If you want to review or change any of your installation settings, click Ba exit the wizard.	ck. Click Cancel to
Current Settings:	
Setup Type:	
Typical	
Destination Folder:	
C:\Program Files (x86)\SDA\SD Card Formatter\	
User Information:	
Name:	
Company:	
InstallShield	
< Back	Cancel

After a few seconds, the installation is complete, click "Finish".

🔛 SD Card Formatter - InstallSh	🛃 SD Card Formatter - InstallShield Wizard 🛛 🕹				
2	InstallShield Wizard Completed	٦			
	The InstallShield Wizard has successfully installed SD Card Formatter. Click Finish to exit the wizard.				
	☑ Launch the program				
	< Back Finish Cancel				

#### 4Burn mirror system software tool— Win32DiskImager

Download linkhttps://sourceforge.net/projects/win32diskimager/

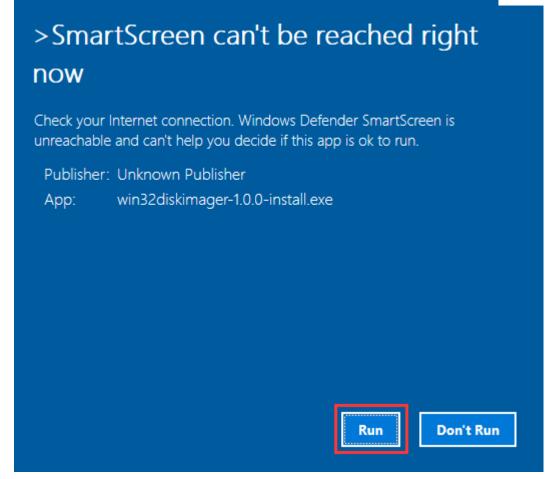
Home / Browse / S	ystem Adminis	stration / Storage /	Win32 Disk Imager						PROJECT
	Win32 Disk Imager								MONTH MAR 2014
- 🐋									-
	Brough	it to you by: g	ruemaster, tux	inator200	19				
	+			Devent		V1-			40.04.07
****	★ 112	Reviews		Downl	oads: 42,251 This V	Veek	I	Last Update: 20	18-06-07
💩 Do	Share This								
Summary	Files	Reviews	Support	Wiki	Feature Requests	Bugs	Code	Mailing Lists	Blog
	0				able device or backup a			0	<i>'</i>

modify this program. Patches are always welcome.

This release is for Windows 7/8.1/10. It will should also work on Windows Server 2008/2012/2016 (although not tested by the developmers). For Windows XP/Vista, please use v0.9 (in the files archive).

1. After downloading the software file win32diskimager-1.0.0-install.exe double-click Win32DiskImagersoftware file win32diskimager-1.0.0-install.exe and then click"Run".

 $\times$ 



2. After selecting I accept the agreement and click "Next".

👒 Setup - Win32DiskImager —		$\times$
License Agreement Please read the following important information before continuing.	¢	
Please read the following License Agreement. You must accept the terms of this agreement before continuing with the installation.		
This program is licensed under the GNU GPL Version 2 License. Included libraries	•	
are licensed under GPL v2 and LGPL v2.1 accordingly.		
GNU GENERAL PUBLIC LICENSE		
Version 2, June 1991		
Copyright (C) 1989, 1991 Free Software Foundation, Inc.,	~	
• I accept the agreement		
○ I do not accept the agreement		
Next >	Cano	el

#### 3. Click "Browse..." select the location where Win32DiskImager is installed and click "Next".

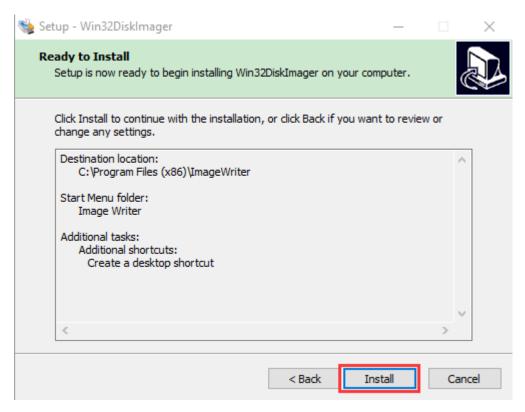
🐞 Setup - Win32DiskImager	_		$\times$
Select Destination Location Where should Win32DiskImager be installed?			
Setup will install Win32DiskImager into the following folder.			
To continue, click Next. If you would like to select a different folder,	click Bro	wse.	
C:\Program Files (x86)\ImageWriter	Br	owse	
At least 44.2 MB of free disk space is required.			
< Back Next	>	Car	ncel

4. Click "Browse..." select the location where Win32DiskImager is installed and click "Next".

👒 Setup - Win32Disklmager	—		$\times$
Select Start Menu Folder Where should Setup place the program's shortcuts?		¢	
Setup will create the program's shortcuts in the following	Start Men	u folder.	
To continue, click Next. If you would like to select a different fold	er, dick Bro	owse.	_
Image Writer	Br	owse	
< Back N	ext >	Car	ncel

5. Select Create a desktop shortcut and click "Next" and then click again "Install".

🐋 Setup - Win32Disklmager	_		$\times$
Select Additional Tasks Which additional tasks should be performed?		¢	
Select the additional tasks you would like Setup to perform w Win32DiskImager, then click Next. Additional shortcuts: Create a desktop shortcut	hile installing		
< Back	Next >	Car	ncel



After a few seconds, the installation is complete, click "Finish".

👒 Setup - Win32Disklmager	– 🗆 ×
	Completing the Win32DiskImager on your computer. The application may be launched by selecting the installed shortcuts.         Click Finish to exit Setup.         View README.txt         Launch Win32DiskImager
	Finish

5Scan for IP address software tool-WNetWatcher Download Linkhttp://www.nirsoft.net/utils/wnetwatcher.zip

#### 1.3. Raspberry PI mirror system

Download link for the latest version

https://www.raspberrypi.org/downloads/raspberry-pi-os/

Download link for the old version

- Raspbianhttps://downloads.raspberrypi.org/raspbian/images/
- Raspbian fullhttps://downloads.raspberrypi.org/raspbian\_full/images/
- Raspbian litehttps://downloads.raspberrypi.org/raspbian\_lite/images/

We use the 2020.05.28 version in the tutorial and recommend you to use this version(Please download this version as shown in the picture below.)

https://downloads.raspberrypi.org/raspios\_full\_armhf/images/raspios\_full\_armhf-2021-05-28/

# Index of /raspios\_full\_armhf/images/raspios\_full\_armhf-2021-05-28

Na	ıme	Last modi	fied	<u>Size</u>	<b>Description</b>
Parent Directory				-	
2021-05-07-raspios-bus	ter-armhf-full.info	2021-05-07	16:23	288K	
2021-05-07-raspios-bus	ter-armhf-full.zip	2021-05-07	16:35	2.8G	-
2021-05-07-raspios-bus	ter-armhf-full.zip.sha1	2021-05-28	15:49	83	
2021-05-07-raspios-bus	ter-armhf-full.zip.sha256	2021-05-28	15:49	107	
2021-05-07-raspios-bus	ter-armhf-full.zip.sig	2021-05-28	15:00	488	
2021-05-07-raspios-bus	ter-armhf-full.zip.torrent	2021-05-28	15:50	28K	

### 8.1.2 2. Install Raspberry Pi OS system on Raspberry Pi 4B:

2.1. Interface the TFT memory card with a card reader, then plug the card reader into a computer's USB port.

## 2.2. Use the SD Card Formatter to format a TFT memory card, as illustrated below

SD Card Formatter		×
File Help		
Select card		
E:\-boot		~
		Refresh
Card information		
Туре	SDXC	52
Capacity	59.48 GB	XC
Formatting options Quick format		
Overwrite format		
CHS format size adju	ustment	
Volume label		
boot		
		Format
SD Logo, SDHC	Logo and SDXC Logo	are trademarks of SD-3C, LLC.

SD Card Formatter	$\times$
File Help	
Select card	
E:\-boot	~
	Refresh
SD Card Formatter	$\times$
Do you want to cont Note: As formatting overwrite option is s	e all data on this card. tinue? can take some time (especially when selected), please make sure that your ted to a power supply and that sleep
	Yes No
boot	
DOOL	
SD Logo, SDHC Logo an	Format of SDXC Logo are trademarks of SD-3C, LLC.
SD Card Formatter File Help	×
Select card	
E:\-boot	~
SD Card Formatter	X
Capa Volume infor File system: e Capacity: 59. O Free space: 5	exFAT 45 GB (63,831,015,424 bytes) 59,45 GB (63,830,622,208 bytes) 128 kilobytes
Volun boot	ОК
SD Logo, SDHC Logo an	Format od SDXC Logo are trademarks of SD-3C, LLC.

## 2.3. Burn system:

1Use Win32DiskImager to burn the official Raspberry Pi OS mirror to the TFT memory card.

👒 Win32 Disk Imager - 1.0	Choose the correct letter
Image File	Det c1
inistrator/Desktop/2020-12-02-raspi	os-buster-armhf-full.img 📄 [E:\] 🔻
Kash	0
None 🔻 Generate Copy	Click, and then find the mirror
	system(".img"file)that you
Read Only Allocated Partitions	download and unzipped
Progress	
Click Write to	o write the system
Cancel Read Write	Verify Only Exit
Waiting for a task.	
👒 Win32 Disk Imager - 1.0	
Winsz Disk imager - 1.0	- 🗆 X
Image File	- L ×
	Device
Image File	Device
Image File inistrator/Desktop/2020-12-02-raspi Hash Sconfirm overwrite - 1.0	os-buster-armhf-full.img E [E:\] • al device can corrupt the device. ] "boot")
Image File inistrator/Desktop/2020-12-02-raspin Hash None Vriting to a physice (Target Device: [E:\]	os-buster-armhf-full.img E [E:\] • al device can corrupt the device. ] "boot") vant to continue?
Image File inistrator/Desktop/2020-12-02-raspin Hash None None Writing to a physice (Target Device: [E:\) Are you sure you w	os-buster-armhf-full.img E [E:\] • al device can corrupt the device. ] "boot")
Image File inistrator/Desktop/2020-12-02-raspine Hash Confirm overwrite - 1.0 None Writing to a physice (Target Device: [E:\] Are you sure you we	os-buster-armhf-full.img E [E:\] • al device can corrupt the device. ] "boot") vant to continue?
Image File inistrator/Desktop/2020-12-02-raspine Hash Confirm overwrite - 1.0 None Writing to a physice (Target Device: [E:\] Are you sure you we	os-buster-armhf-full.img E [E:\] • al device can corrupt the device. ] "boot") vant to continue?

👒 Win32 Disk Imager - 1.0		_	
Image File inistrator/Desktop/2020-	12-02-raspios-buster-armhf-ful	1. img 📔	Device [E:\] ▼
Hash None 🔻 Generate	Somplete - 1.0 ×		
Read Only Allocated 1 Progress	ОК		
Cancel Read	Write Verify Or	Jу	Exit 08:40/08:40

2After the mirror system is burned, don't pull out the card reader, use Notepad to create a file named **SSH**, delete\*\*.txt\*\*, and then copy it to the boot directory of the TFT card, so that you can open SSH login function, as shown in the following figure:

→ * ↑				ڻ ~	,O Search boo
	^	Name	Date modified	Туре	Size
Quick access		start.elf	11/26/2020 5:30 PM	ELF File	2,869 KB
Desktop	*	start_cd.elf	11/26/2020 5:30 PM	ELF File	771 KB
Downloads	1	start_db.elf	11/26/2020 5:30 PM	ELF File	4,674 KB
🗄 Documents	*	start_x.elf	11/26/2020 5:30 PM	ELF File	3,610 KB
Pictures	*	start4.elf	11/26/2020 5:30 PM	ELF File	2,162 KB
		start4cd.elf	11/26/2020 5:30 PM	ELF File	771 KB
This PC		start4db.elf	11/26/2020 5:30 PM	ELF File	3,627 KB
3D Objects		start4x.elf	11/26/2020 5:30 PM	ELF File	2,904 KB
Desktop		bcm2708-rpi-b.dtb	11/26/2020 5:30 PM	DTB File	25 KB
Documents		bcm2708-rpi-b-plus.dtb	11/26/2020 5:30 PM	DTB File	25 KB
- Downloads		bcm2708-rpi-b-rev1.dtb	11/26/2020 5:30 PM	DTB File	25 KB
Music		bcm2708-rpi-cm.dtb	11/26/2020 5:30 PM	DTB File	25 KB
Pictures		bcm2708-rpi-zero.dtb	11/26/2020 5:30 PM	DTB File	25 KB
		bcm2708-rpi-zero-w.dtb	11/26/2020 5:30 PM	DTB File	26 KB
Videos		bcm2709-rpi-2-b.dtb	11/26/2020 5:30 PM	DTB File	26 KB
Windows10 1909 (C:)		bcm2710-rpi-2-b.dtb	11/26/2020 5:30 PM	DTB File	26 KB
新加卷(D:)		bcm2710-rpi-3-b.dtb	11/26/2020 5:30 PM	DTB File	28 KB
boot (E:)		bcm2710-rpi-3-b-plus.dtb	11/26/2020 5:30 PM	DTB File	28 KB
USB Drive (F:)		bcm2710-rpi-cm3.dtb	11/26/2020 5:30 PM	DTB File	26 KB
New folder (\\desktop-eng) (Z:)		bcm2711-rpi-4-b.dtb	11/26/2020 5:30 PM	DTB File	47 KB
		bcm2711-rpi-400.dtb	11/26/2020 5:30 PM	DTB File	47 KB
boot (E:)		bcm2711-rpi-cm4.dtb	11/26/2020 5:30 PM	DTB File	47 KB
overlays		bootcode.bin	11/26/2020 5:30 PM	BIN File	52 KB
USB Drive (F:)		fixup.dat	11/26/2020 5:30 PM	DAT File	8 KB
030 0110 (1)		fixup_cd.dat	11/26/2020 5:30 PM	DAT File	4 KB
Network		fixup_db.dat	11/26/2020 5:30 PM	DAT File	11 KB
DESKTOP-1V3JQ2C		fixup_x.dat	11/26/2020 5:30 PM	DAT File	11 KB
DESKTOP-901C3HI		fixup4.dat	11/26/2020 5:30 PM	DAT File	6 KB
DESKTOP-BDC71VT		fixup4cd.dat	11/26/2020 5:30 PM	DAT File	4 KB
DESKTOP-CMPIKJB		fixup4db.dat	11/26/2020 5:30 PM	DAT File	9 K.B
DESKTOP-SMT1PCK		fixup4x.dat	11/26/2020 5:30 PM	DAT File	9 KB
		LICENCE.broadcom	9/30/2020 12:00 PM	BROADCOM File	2 KB
DYJ		COPYING.linux	5/27/2020 10:57 AM	LINUX File	19 KB
FTJ69C9C26XOD0S		overlays	12/2/2020 12:39 PM	File folder	
HK4KUMDY9PBVSC0		E SSH	12/8/2020 11:48 AM	Text Document	0 KB

3Pull out card reader.

#### 2.4. Log in system:

The following operations require raspberry to be on the same LOCAL area network as the PC

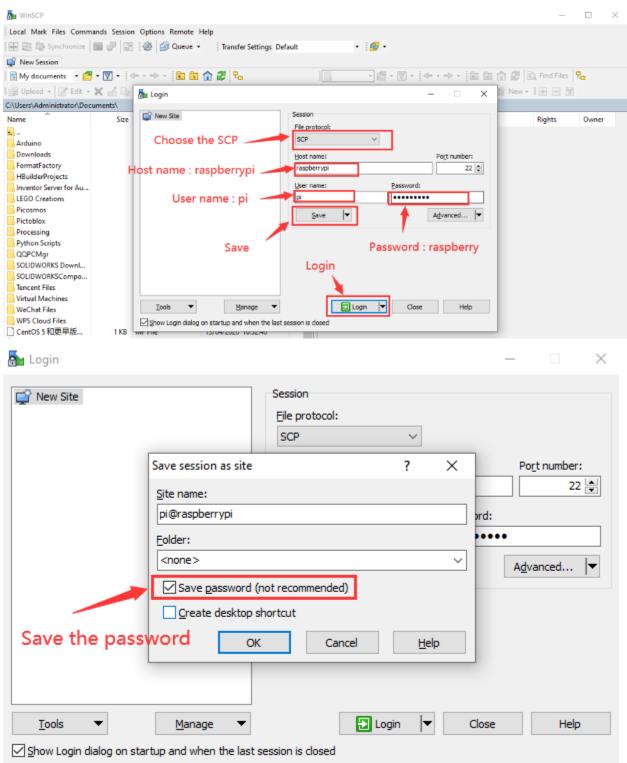
#### **1Preparation**

Insert the burned TFT memory card into the Raspberry Pi, connect internet cable and plug in power. If you have a screen and a HDMI cable of Raspberry Pi, connect the screen, and you can see the Raspberry Pi OS startup screen. If you don't have an HDMI cable of Raspberry Pi, you can enter the desktop of Raspberry Pi via SSH remote login software—WinSCP and xrdp.

#### 2Remote login

Use WinSCP to log in using the default Raspberry Pi system name, default user name, default password.

Note that only a raspberry pi can be connected to a network.



## 3View the ip address and mac address

🛚 / - pi@raspberry - WinSCP							
ocal <u>M</u> ark <u>F</u> iles <u>C</u> ommand	s Session Options Remote	Help					
🗄 🔁 📚 Synchronize 🛛 🖬	🧬 💽 🔅 🛞 🖓 Queu	• Transfer Settin	gs Default 🔹 🥩 🔹				
pi@raspberry 🗙 🚅 New	Session Click of	oen terminal					
De - 🚰 - 🕎 - 🗺 -			/ <root> • 🚰 • 🕅 • 🖛 - 🔿 -</root>	a ia 🏠 🔊	🔁 Find Files 🛛 😓		
🗑 Upload 👻 📝 Edit 👻 🗙			i 🖓 Download - 📝 Edit - 🗶 🚮 🕞 Prope				
Users\Administrator\Desktop							
· · · · ·	-		/ 				
ame	Size Type	Changed ^	Name	Size	Changed	Rights	Owner
	Parent directory	19/10/2020 09:5			20/08/2020 12:09:38	FWXF-XF-X	root
3D8S_CSharp_Control	File folder	07/08/2020 14:1	bin		20/08/2020 11:36:31	FWXF-XF-X	root
4wd_motor_test	File folder	17/08/2020 17:5	boot		01/01/1970 01:00:00	FWXF-XF-X	root
7大课程收获	File folder	29/09/2020 17:2	dev .		20/08/2020 12:09:38	FWXF-XF-X	root
15_ble_all	File folder	15/09/2020 11:5	etc		20/08/2020 12:09:48	rwxr-xr-x	root
16and8Game	File folder	16/09/2020 11:5	home		20/08/2020 11:31:10	rwxr-xr-x	root
alienzhangyw-BlockP	File folder	11/08/2020 11:4	lib		20/08/2020 11:45:50	rwxr-xr-x	root
arduino-esp32-esp32s2	File folder	31/08/2020 15:1	lost+found		20/08/2020 12:08:08	rwx	root
Arm_car	File folder	31/07/2020 15:5	media		20/08/2020 11:26:08	rwxr-xr-x	root
AutoCAD	File folder	21/09/2020 18:0	mnt		20/08/2020 11:26:08	rwxr-xr-x	root
bluetooth test	File folder	24/08/2020 16:1	opt		20/08/2020 11:43:02	rwxr-xr-x	root
car_test	File folder	04/09/2020 13:5	proc		01/01/1970 01:00:00	r-xr-xr-x	root
EB0028 原理图	File folder	15/10/2020 17:1	root		20/08/2020 12:09:43	rwx	root
esp	File folder	12/09/2020 10:0	run		20/08/2020 12:10:05	FWXF-XF-X	root
esp32 test	File folder	14/09/2020 10:2	sbin		20/08/2020 11:45:50	FWXF-XF-X	root
esp8266_arduino	File folder	09/10/2020 15:4	SIV		20/08/2020 11:26:08	FWXF-XF-X	root
esp8266 test	File folder	09/10/2020 15:3	5/5		01/01/1970 01:00:01	F-XF-XF-X	root
esp-idf	File folder	03/09/2020 16:3	tmp		20/08/2020 12:09:45	rwxrwxrwt	root
Grove-Beginner-Kit-f	File folder	09/10/2020 09:1	usr		20/08/2020 11:38:05	DAXE-XE-X	root
HT16K33 8x16dot ma	File folder	28/08/2020 14:5	var		20/08/2020 12:09:38	DWXF-XF-X	root
keyestudio-e-Paper	File folder	19/06/2020 11:2			20,00,2020 12/03/30	INAL-AL-A	1001
M_car	File folder	28/08/2020 16:2					
otto1	File folder	16/10/2020 08:4					
	File folder	10/10/2020 08:2					
processing_test	File folder	31/07/2020 15:5	The surface Club to all				
ps2_arm_car RGB点阵	File folder		The system file for the ra	spberry P	i system		
		24/08/2020 17:0					
TS1693 CQRobot PAJ	File folder	07/09/2020 17:5					
TS1695 CQRobot BM	File folder	07/09/2020 14:4					
TS1727 CQROBOT AD	File folder	27/09/2020 10:4					

Click to open terminal and input the password: raspberry, and tap"Enter" on keyboard.



After successful login, open the terminal, input ip a and tap"Enter"keyboard to view the ip address and mac address.

🛃 pi@raspberrypi: ~

```
Wi-Fi is currently blocked by rfkill.
Use raspi-config to set the country before use.
pi@raspberrypi:~ $ ip a
1: lo: <LOOPBACK,UP,LOWER UP> mtu 65536 qdisc noqueue state UNKNOWN group defaul
 glen 1000
   link/loopback 00:00:00:00:00 brd 00:00:00:00:00
   inet 127.0.0.1/8 scope host lo
      valid lft forever preferred lft forever
   inet6 ::1/128 scope host
      valid lft forever preferred lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER UP> mtu 1500 qdisc mq state UP group defa
ult qlen 1000
    link/ether dc:a6:32:17:61:9c brd ff:ff:ff:ff:ff:ff
    inet 192.168.1.128/24 ord 192.168.1.255 scope global dynamic noprefixroute e
th0
      valid lft 1357sec preferred lft 1132sec
   inet6 fe80::1e7d:5653:59e9:3262/64 scope link
       valid lft forever preferred lft forever
3: wlan0: <BROADCAST,MULTICAST> mtu 1500 qdisc noop state DOWN group default qle
 1000
   link/ether dc:a6:32:17:61:9d brd ff:ff:ff:ff:ff:ff
oi@raspberrypi:~ $
```

From the above figure, mac address of this Raspberry Pi is a6:32:17:61:9c, and ip address is 192.168.1.128(use ip address to finish xrdp remote login).

Since mac address never changes, you could confirm ip via mac address when not sure which ip it is.

#### 4Fix the IP address of Raspberry Pi

IP address is changeable, therefore, we need to make IP address fixed for convenient use.

Follow the below steps: Switch to root user If without root user's password Set root password Input password in the terminal: sudo passwd root to set password. Switch to root user su root Fix the configuration file of IP address Firstly change IP address of the following configuration file. #New IP address: address 192.168.1.99 Copy the above new address to terminal and tap"**Enter**"keyboard. Configuration File\*\*:\*\* echo -e ' ×

auto eth0

iface eth0 inet static

#Change IP address

address 192.168.1.99

netmask 255.255.255.0

gateway 192.168.1.1

network 192.168.1.0

broadcast 192.168.1.255

dns-domain 119.29.29.29

dns-nameservers 119.29.29.29

metric 0

mtu 1492

'>/etc/network/interfaces.d/eth0

Example operation diagram, as follows

pi@raspberrypi:~ \$ su root
Password:
root@raspberrypi:/home/pi# echo -e '
> auto eth0
> iface eth0 inet static
> #Change IP address
> address 192.168.1.99
> netmask 255.255.255.0
> gateway 192.168.1.1
> network 192.168.1.0
> broadcast 192.168.1.255
> dns-domain 119.29.29.29
> dns-nameservers 119.29.29.29
> metric 0
> mtu 1492
<pre>&gt; '&gt;/etc/network/interfaces.d/eth0</pre>
root@raspberrypi:/home/pi#

Reboot the system to activate the configuration file. Input the restart command in the terminal: sudo reboot You could log in via fixed IP afterwards. Check IP and insure IP address fixed well.

pi@raspberrypi:~ \$ ip a
1: lo: <loopback, lower_up="" up,=""> mtu 65536 qdisc noqueue state UNKNOWN group defaul</loopback,>
t qlen 1000
<pre>k link/loopback 00:00:00:00:00 brd 00:00:00:00:00:00</pre>
inet 127.0.0.1/8 scope host lo
valid lft forever preferred lft forever
inet6 ::1/128 scope host
valid lft forever preferred lft forever
2: eth0: <broadcast,multicast,up,lower_up> mtu 1492 qdisc mq state UP group defa</broadcast,multicast,up,lower_up>
Bult glen 1000
link/ether dc:a6:32:17:61:9c brd ff:ff:ff:ff:ff:ff
inet 192.168.1.99/24 brd 192.168.1.255 scope global eth0
valid_lft forever preferred_lft forever
inet 192.168.1.128/24 brd 192.168.1.255 scope global secondary dynamic nopre
fixroute eth0
valid_lft 1730sec preferred_lft 1505sec
inet6 fe80::1e7d:5653:59e9:3262/64 scope link
valid lft forever preferred lft forever
3: wlan0: <broadcast,multicast> mtu 1500 qdisc noop state DOWN group default qle</broadcast,multicast>
n 1000
link/ether dc:a6:32:17:61:9d brd ff:ff:ff:ff:ff:ff
pi@raspberrypi:~ \$

#### 5Log in desktop on Raspberry Pi wirelessly

If we don't have an HDMI cable to connect to the display, can we wirelessly log in to the Raspberry Pi desktop from the Windows desktop? Yes, there are many methods, VNC and Xrdp are commonly used to log in desktop of Raspberry Pi wirelessly.

Let's take an example of Xrdp.

Install Xrdp Service in the terminal

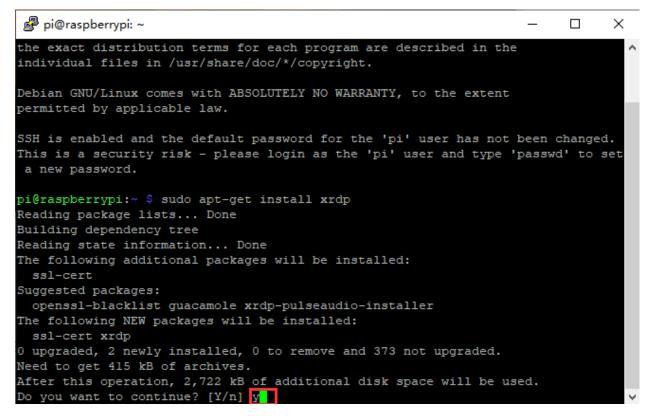
Installation commands:

Switch to root User: su root

Installation commands: apt-get install xrdp

Enter y and tap"Enter"keyboard...

As shown below:



Open the remote desktop connection on Windows

Press WIN+R on keyboard and enter mstsc.exe.

#### As shown below:

🖅 Run		×
	Type the name of a program, folder, document, or Internet resource, and Windows will open it for you.	t
<u>O</u> pen:	mstsc.exe	~
	OK Cancel <u>B</u> rowse	

Enter the IP address of the Raspberry Pi, as shown below. Click "Connect" and then click "Connect"again. 192.168.1.99 is the ip address we use, you could change it into your IP address.

Nemote Desktop Connection	$\times$
Do you trust this remote connection?	
This remote connection could harm your local or remote computer. Make sure that you trust the remote computer before you connect.	
Type: Remote Desktop Connection Remote computer: 192.168.1.99	
Don't ask me again for connections to this computer	
Show Details Connect Canc	el

A prompt will appear and you can click"Yes".

Nemote Desktop Connection	×
The identity of the remote computer cannot be verified. Do you want connect anyway?	to
This problem can occur if the remote computer is running a version of Windows that is earlier than Windows Vista, or if the remote computer is not configured to support server authentication.	
For assistance, contact your network administrator or the owner of the remote computer.	е
Don't ask me again for connections to this computer	
Yes No	

Then enter the user name: pi ,and the default password: raspberry, as shown below:

No. 192.168.1.253 - 远程桌面连接	_	×
Login to raspberrypi         Just         Just         Just         connecting         username         pi         password         WK密码: raspberry         OK		
<		> .::

Click"OK" or tap"Enter" keyboard, you will view the desktop of Raspberry Pi OS, as shown below:



Now, we finish the basic configuration of the Raspberry Pi OS system.

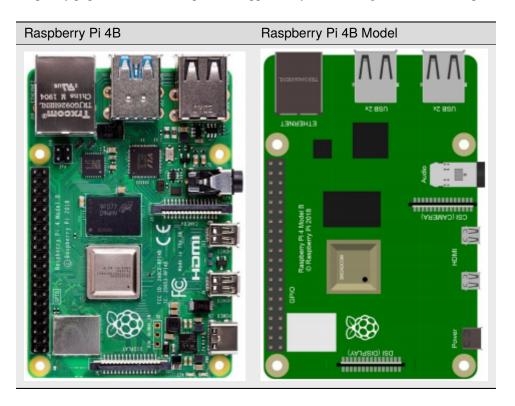
## 8.1.3 3. Preparation of C language control basic hardware:

C language is a programming language with a considerably fast running speed. There are numerous software and system core code written in it, such as Linux system. Notably, hardware MCU and embedded class are not exception. Thereby, it makes sense to learn the C language to control hardware.

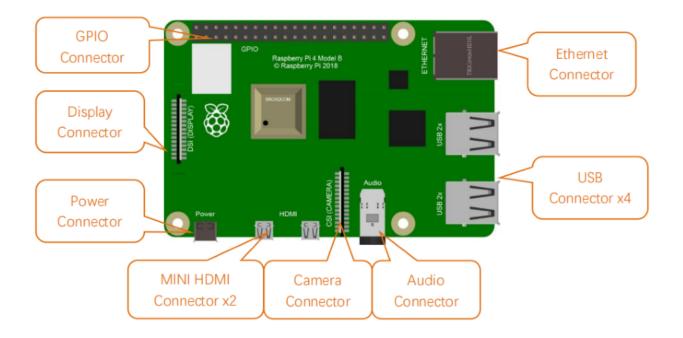
#### (1)Description of basic raspberry pi accessories

#### **Raspberry Pi 4B**

Below are the raspberry pi pictures and model pictures supported by this learning kit. There are 40 pins.



**Hardware Interfaces** 

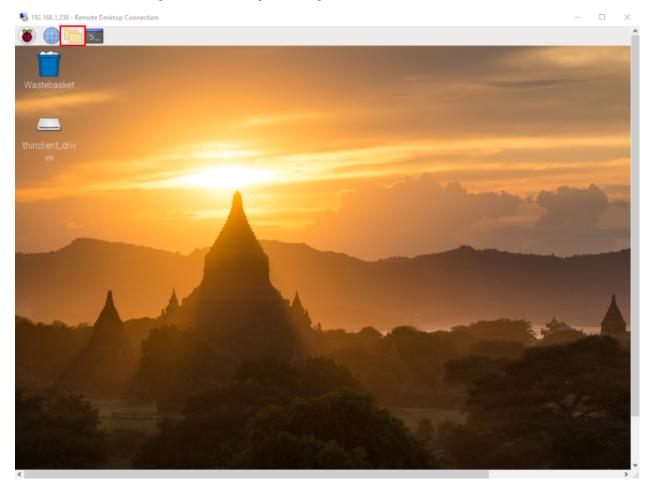


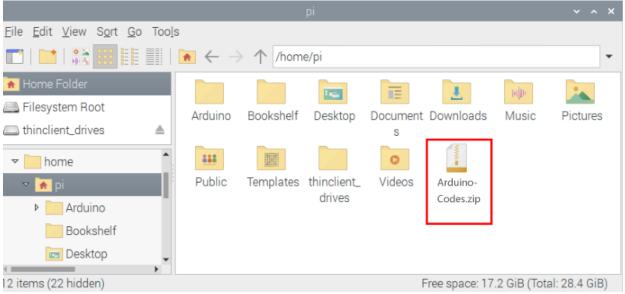
### (2)Raspberry Pi +ESP32 mainboard + breadboard +USB cable, as shown below

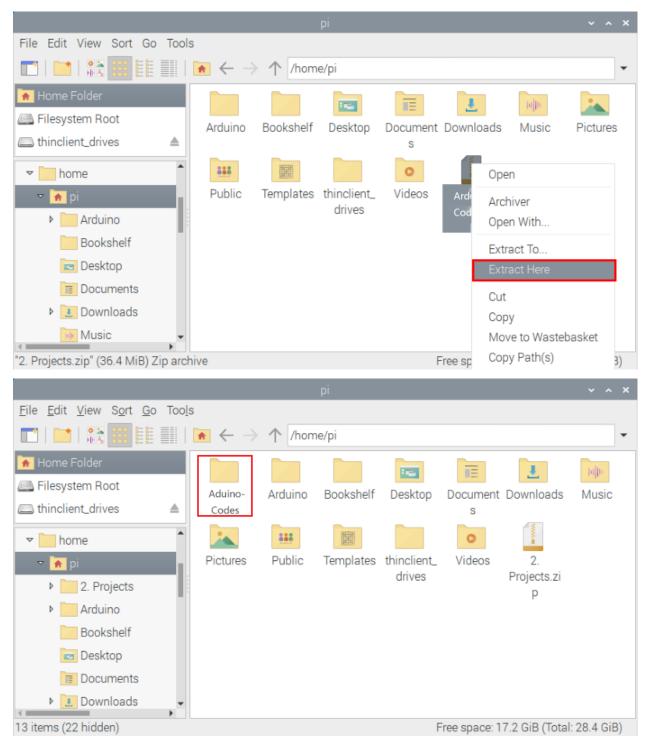


#### (3)Copy Example Code Folder to Raspberry Pi

Place example code folder to the pi folder of Raspberry Pi. and Just copy and paste the **Arduino-Codes.zip** file (the default is **ZIP** file) that we provided into user pi and unzip it, as shown below:







Double-click the Arduino-Codes file, as shown below.

		2.	Projects				~ ~	×
<u>File E</u> dit <u>V</u> iew S <u>o</u> rt <u>G</u> o Too <u>l</u>	s							
	$\blacktriangleright \leftarrow \rightarrow$	↑ /hom	e/pi/ Arduinc	o-Codes				•
觰 Home Folder								Î
📖 Filesystem Root	Project	Project	Project	Project	Project	Project	Project	
limit_drives	01 : Hello	02 : Turn	03 : LED	04 :	05 :	06 : RGB	07:	
- UCY	World	On LED	Flashing	Breathing	Traffic Lig	LED	Flowing	
▶ etc								
▼ home	Project	Project	Project	Project	Project	Project	Project	
🗢 🏦 pi	08:1-	09:4-	10:8×8	11:	12:	13:	14 : Mini	
- 2. Projects	Digit Digit	Digit Digit	Dot-matri	74HC595	Active Bu	Passive B	Table La	
Project 01 : Hello								
· · · · · · · · · · · · · · · · · · ·	Project	Project	Project	Project	Project	Project	Project	-
40 items					Free space	e: 17.2 GiB (T	otal: 28.4 Gi	B)

# 8.2 Linux SystemRaspberry Pi

## 8.2.1 1. Download and install Arduino IDE

1First, click on Raspberry Pi's browser.



2. Enter the Official Arduino website in your browserwww.arduino.cc/en/software , as shown below:

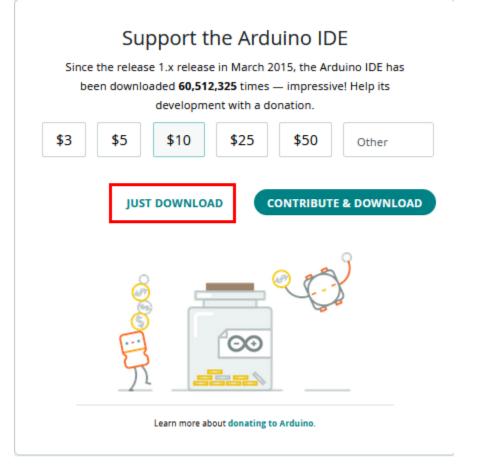
ng 192.168.0.167 - Remote Desktop Connection	_		$\times$
🛞 🌐 🥅 🗾 💽 New Tab - Chromium			^
New Tab - Chromium			
New Tab × +			
← → C ( www.arduino.cc/en/software			
٢			> .:
ng 192.168.0.167 - Remote Desktop Connection	_		$\times$
🕉) (  ) 🦳 🔽 📀 Software   Arduino - C			^
Software   Arduino - Chromium			<b>1</b>
Software   Arduino × +			
← → C arduino.cc	Q A	r)	U
HARDWARE SOFTWARE CLOUD DOCUMENTATION - COMMUNITY - BLOG ABOUT			-
Downloads			-
Downloads			
Arduino IDE 1.8.19 DownLoad options			
Arduino IDE 1.8.19 Windows Win 7 and newer Windows ZIP file			
The open-source Arduino Software (IDE) makes it easy to write code Windows app Win 8.1 or 10 Ge	8		
and upload it to the board. This software can be used with any Arduino board. Linux 32 bits Linux 64 bits			
Refer to the Getting Started page for Installation Instructions.			
SOURCE CODE Mac OS X 10.10 or newer			
Active development of the Arduino software is <b>hosted by GitHub</b> . See the instructions for <b>building the code</b> . Latest release source			
code archives are available <b>here</b> . The archives are PGP-signed so Checklard (Ma312) they can be verified using <b>this</b> gpg key.			
they can be verned using this gpg key.			
<			> .:

3There are various versions of IDE for Arduino. Just download a version compatible with your system (install the lasted Arduino IDE) and click "Linux ARM 32 bits".

# Downloads



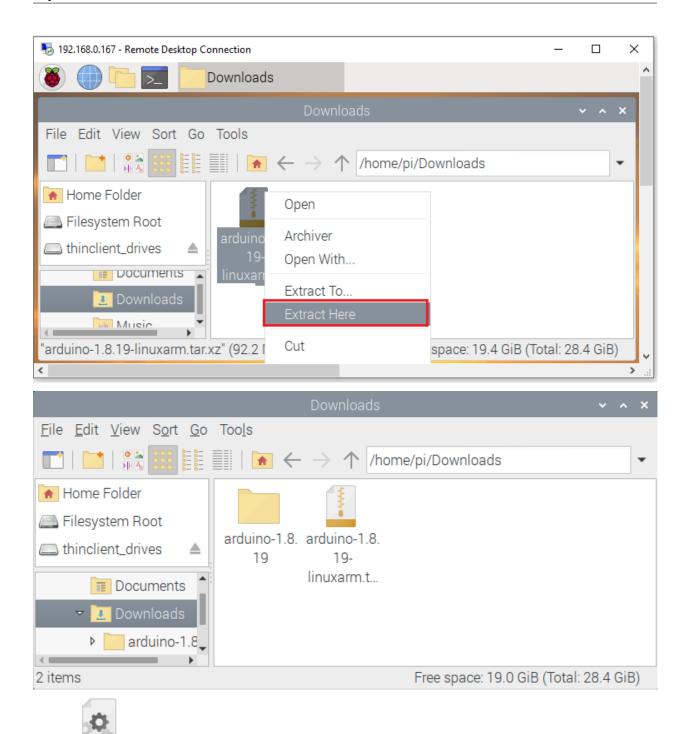
In general, you can click **JUST DOWNLOAD** to download, although if you like, you can choose a small sponsorship to help the great Arduino open source cause.



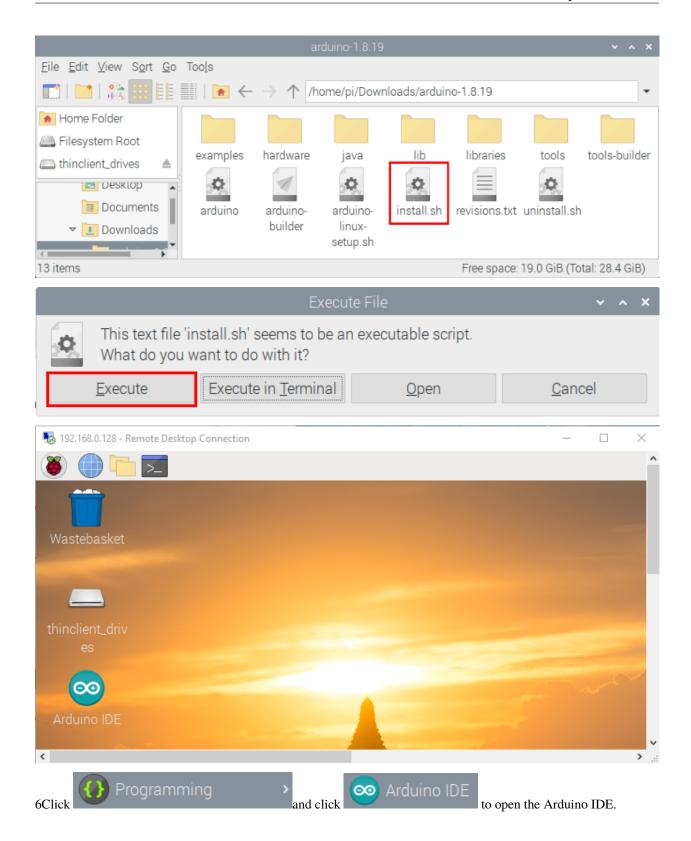
After a few seconds, the lasted Arduino IDEArduino 1.8.19 versionzip file can be directly downloaded.

(4) Click , then find the **Downloads** file from the pi folder and click it. In the **Downloads** folder, you can see the package"arduino-1.8.19-linuxarm.tar.xz" that you just downloaded. Then unzip the package"arduino-1.8.19-linuxarm.tar.xz", after a while, the package is unzipped.

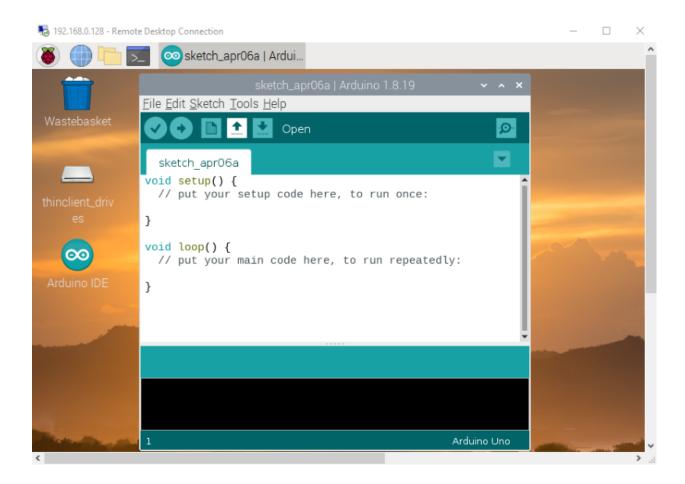
192.168.0.167 - Remote Desktop Connection			—	) ×
🗕 🛑 🗾				^
				_
Wastebasket				
and the second				
thinclient_driv				U U
<				> .::
192.168.0.167 - Remote Desktop Connection			— [	
🥘 🛑 🗾 🛄 pi				^
	pi		~	~ X
<u>F</u> ile <u>E</u> dit <u>V</u> iew S <u>o</u> rt <u>G</u> o Too <u>l</u> s				
🗖 📄   🔛   👬 🎫 🗄 🗐   🖿 🔹	$\leftrightarrow \rightarrow \wedge$ /	home/pi		•
觰 Home Folder			a dh	
🚐 Filesystem Root		10 L	- Ind the	- 11
☐ thinclient_drives ▲ Bookshel	f Desktop	Documents Downloads	Music	- 11
▶ etc			0	
✓ home Pictures	Public	Templates thinclient_dr	Videos	
		ives		-
10 items (21 hidden)		Free space: 19.4 GiB (	(Total: 28.4	GiB)



5Click install.sh file and tap itclick "Execute" in the dialog that appears to install the Arduino IDE. Once installed, an Arduino software shortcut is generated in the desktop.



💀 192.168.0.128 - Remote Desktop	Connection	_	$\times$
🕘 🛑 💽 👘			^
Programming	> 🐼 Arduino IDE		
😸 Education	> 🐔 BlueJ Java IDE		
🥠 Office	> 🤯 Geany Programmer's Editor		
Internet	> 🕇 Greenfoot Java IDE		
🔛 Sound & Video	> 🔆 Mathematica		
🚏 Graphics	> mu		
Games	> Rede-RED		
Accessories	> 😸 Scratch		
🔂 Help	, Scratch 3		
	Sense HAT Emulator		
Preferences	(n))) Sonic Pi		
Run	Thonny Python IDE		
Shutdown	🚯 Wolfram		~
<			>:



## 8.2.2 2. Install the ESP32 on Arduino IDE

Note: You need to download Arduino IDE 1.8.5 or advanced version to install the ESP32.



sketch_apr06a   Arduino 1.8.19	~ ^ X
<u>F</u> ile <u>E</u> dit <u>S</u> ketch <u>T</u> ools <u>H</u> elp	_
	<mark>,₽</mark> -
sketch_apr06a	
<pre>void setup() {    // put your setup code here, to run once:</pre>	î
}	- 1
<pre>void loop() {     // put your main code here, to run repeatedly:</pre>	- 1
}	- 1
	Ļ
1 Ardu	iino Uno

2) Click\*\*"File\*\*"→\*\*"Preferences"\*\*copy the website address https://dl.espressif.com/dl/package\_esp32\_ index.json in the"Additional Boards Manager URLs:"and click"OK" to save the address.

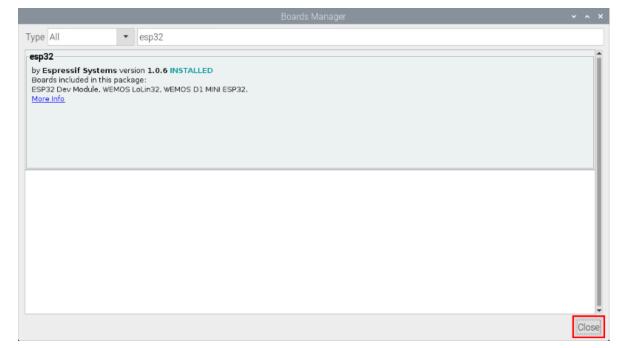
	sketch_a	apr06a   Arduino 1.8.19 🛛 🗸 🔺 🗙
<u>File</u> <u>E</u> dit <u>S</u> ketch	n <u>T</u> ools <u>H</u> elp	
New	Ctrl+N	
Open	Ctrl+O	
Open Recent	>	
Sketchbook	>	e here, to run once:
Examples	>	
Close	Ctrl+W	
Save	Ctrl+S	here, to run repeatedly:
Save As	Ctrl+Shift+S	
Page Setup	Ctrl+Shift+P	
Print	Ctrl+P	
Preferences	Ctrl+Comma	
Quit	Ctrl+Q	
1		Arduino Uno

	Preferences 🗸 🗸 🗙
Settings Network	
Sketchbook location:	
/home/pi/Arduino	Browse
Editor language:	System Default  v (requires restart of Arduino)
Editor font size:	12
Interface scale:	✓ Automatic 100 <sup>●</sup> / <sub>●</sub> % (requires restart of Arduino)
Theme:	Default theme    (requires restart of Arduino)
Show verbose output during:	compilation upload
Compiler warnings:	None 💌
<ul> <li>Display line numbers</li> <li>Verify code after upload</li> <li>Check for updates on start</li> <li>Use accessibility features</li> </ul>	<ul> <li>Enable Code Folding</li> <li>Use external editor</li> <li>up</li> <li>✓ Save when verifying or uploading</li> </ul>
Additional Boards Manager U	RLs: https://dl.espressif.com/dl/package_esp32_index.json 1
More preferences can be edite	d directly in the file
/home/pi/.arduino15/preferen	ces.txt
(edit only when Arduino is not	running) (2
	OK Cancel

3. Click"**Tools**"→"**Board:**",then click "**Boards Manager...**"to enter "**Boards Manager**" page . Enter"**esp32**"as follows and select the latest version to Install. The installation package is not large, click"**Install**"to start the installation.

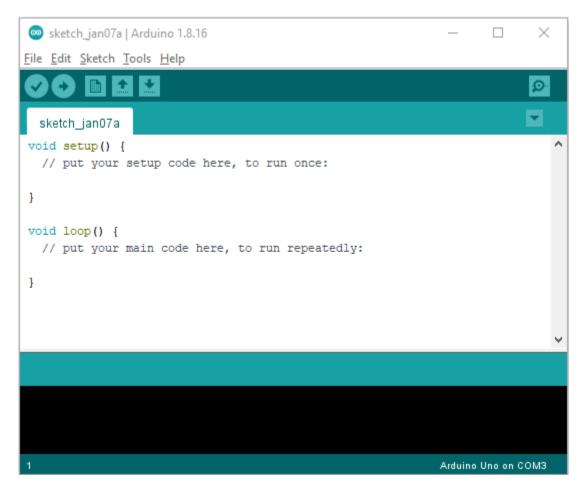
	sketch_apr06a   Arduino 1.8.19	× ^	×
<u>F</u> ile <u>E</u> dit <u>S</u> ketch			
	Auto Format	Ctrl+T	
	Archive Sketch		Boards Manager
sketch_apr00	Fix Encoding & Reload		Boards Manager
<pre>void setup()     // put your</pre>	Manage Libraries	Ctrl+Shift+I	Δ
1	Serial Monitor	Ctrl+Shift+M	Arduino Yún
}	Serial Plotter	Ctrl+Shift+L	<ul> <li>Arduino Uno</li> </ul>
<pre>void loop() // put your</pre>	WiFi101 / WiFiNINA Firmware Updater		Arduino Duemilanove or Diecimila Arduino Nano
}	Board: "Arduino Uno"	>	Arduino Mega or Mega 2560
	Port	>	Arduino Mega ADK
	Get Board Info		Arduino Leonardo
	Programmer: "AVRISP mkII"	>	Arduino Leonardo ETH
	Burn Bootloader	,	Arduino Micro
	Buin Bootioader		Arduino Esplora
			Arduino Mini
1		Arduino Uno	
	Boards Mar	nager	~ ^ X
Type All	<ul> <li>✓ esp32</li> <li>①</li> </ul>		
by Espressif System Boards included in this			2 1.0.6 ▼ Install
	Boards Mar	nager	~ ^ X
Type All	▼ esp32		
esp32			1
by E <b>spressif Syster</b> Boards included in thi ESP32 Dev Module, W <u>More Info</u>			Installing
	Downloading boards definitions. Downloa	ded 23 526kb of 51	126kb. Cancel
	bownousing boards demittions. Downloa	303 20,02000 0101,	Califer

4. After a while, the ESP32 installation package is installed. Click "Close" to Close the page.



## 8.2.3 3. Arduino IDE Settings and toolbars:





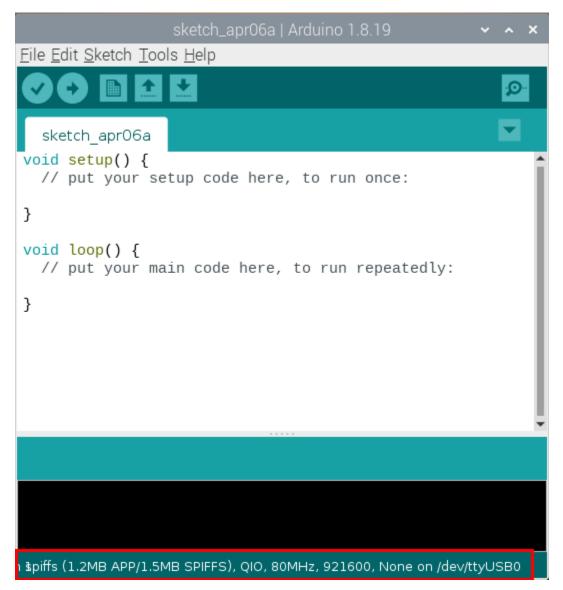
2When downloading the code to the board, you must select the correct name of Arduino board that matches the board connected to the Raspberry Pi,click"Tools"→"Board:". As shown below ;

(Note: we use the ESP32 board in this tutorial; therefore, we select ESP32 Arduino\*\*)\*\* File Edit Sketch Tools Help Auto Format Ctrl+T Archive Sketch sketch\_apr04 Fix Encoding & Reload void setup() // put your Manage Libraries... Ctrl+Shift+I Ctrl+Shift+M Serial Monitor  $\triangle$ } ESP32 Dev Module Serial Plotter Ctrl+Shift+L void loop() { // put your WiFi101 / WiFiNINA Firmware Updater ESP32 Pico Kit } Boards Manager... TinyPICO Port > Arduino AVR Boards S.ODI Ultra v1 > Get Board Info MagicBit Programmer: "AVRISP mkII" > Turta IoT Node Burn Bootloader TTGO LoRa32-OLED V1 TTGO T1 TTGO T7 V1.3 Mini32 Arduino Uno

sketch_apr06a   Arduino 1.8.19 🛛 🗸 🔺 🗙
<u>F</u> ile <u>E</u> dit <u>S</u> ketch <u>T</u> ools <u>H</u> elp
sketch_apr06a
<pre>void setup() {     // put your setup code here, to run once:</pre>
}
<pre>void loop() {     // put your main code here, to run repeatedly:</pre>
}
Default 4MB with spiffs (1.2MB APP/1.5MB SPIFFS), QIO, 80MHz, 921600, None

Then select the correct COM port (After connecting the ESP32 mainboard to the Raspberry Pi via USB cable, you can see the corresponding COM port).

	sketch_apr06a   Arduino 1.8.19 🛛 🗸 🔺 🗙		
<u>F</u> ile <u>E</u> dit <u>S</u> ketch	Tools Help	_	
	Auto Format	Ctrl+T	
	Archive Sketch		
sketch_apr06	Fix Encoding & Reload		
<pre>void setup()   // put your</pre>	Manage Libraries	Ctrl+Shift+I	
	Serial Monitor	Ctrl+Shift+M	
}	Serial Plotter	Ctrl+Shift+L	
<pre>void loop() {     // put your</pre>	WiFi101 / WiFiNINA Firmware Updater		
}	Board: "ESP32 Wrover Module"	>	
	Upload Speed: "921600"	>	
	Flash Frequency: "80MHz"	>	
	Flash Mode: "QIO"	>	
	Partition Scheme: "Default 4MB with spiffs (1.2MB APP/1.5MB SPIFFS)	)" >	
	Core Debug Level: "None"	>	
	Port	>	Serial ports
	Get Board Info		/dev/ttyAMA0
	Programmer	>	/dev/ttyUSB0
Default 4MB with s	Burn Bootloader		a sea of the sea



Before a code was uploaded to the ESP32 mainboard, we have to demonstrate the functionality of each symbol that appeared in the Arduino IDE toolbar.

sketch_apr06a   Arduino 1.8.19 🗸 🗸	×
<u>File Edit Sketch Tools H</u> elp	
sketch_apr06a	
<pre>void setup() {     // put your setup code here, to run once:     F</pre>	Î
ABCDE	ł
<pre>void loop() {     // put your main code here, to run repeatedly:</pre>	
}	I
	Ŧ
ו spiffs (1.2MB APP/1.5MB SPIFFS), QIO, 80MHz, 921600, None on /dev/ttyUSB0	

- A- Used to verify whether there is any compiling mistakes or not.
- B- Used to upload the sketch to your Arduino board.
- C- Used to create shortcut window of a new sketch.
- D- Used to directly open an example sketch.
- E- Used to save the sketch.
- F- Used to send the serial data received from board to the serial monitor.

## 8.3 Import the Arduino C library

Before starting the course, we also need to install the Arduino C library files required by the code in the course.

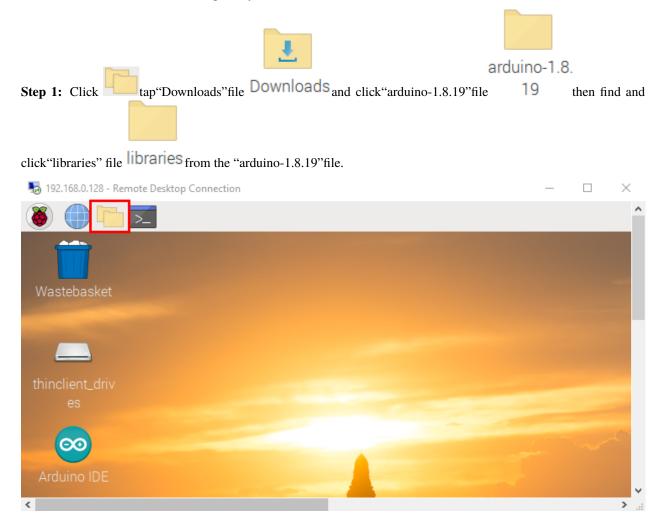
## 8.3.1 What are Libraries ?

Librariesare a collection of code that make it easy for you to connect sensors, displays, modules, etc.

For example, the built-in LiquidCrystal library helps talk to LCD displays. There are hundreds of additional libraries available on the Internet for download.

The built-in libraries and some of these additional libraries are listed in the reference. (https://www.arduino.cc/en/Reference/Libraries)

Here we will introduce the most simple way to add libraries .

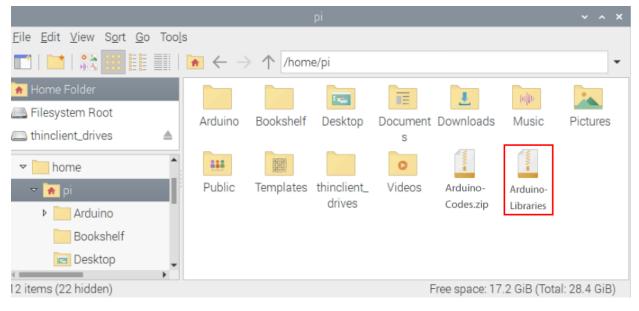


		p	i de la companya de l			~ ^ X
<u>F</u> ile <u>E</u> dit <u>V</u> iew S <u>o</u> rt <u>G</u> o	Too <u>l</u> s					
	🖬 🤶	ightarrow  ightarrow /hc	ome/pi			•
🖌 Home Folder					Ł	pd]p
thinclient_drives	Arduino	Bookshelf	Desktop	Documents	Downloads	Music
			PETRI		0	I
🗢 🏫 pi	Pictures	Public	Tomplates	thinclient_dr	Videos	
Arduino	Fictures	Fublic	Templates	ives	VIGEOS	
11 items (18 hidden)				Free spac	e: 48.9 GiB (T	otal: 58.2 GiB)
		Down	loads			~ ^ X
<u>File Edit View Sort G</u> o	Too <u>l</u> s					
	•	ightarrow  /hc	ome/pi/Dowr	loads		•
🔒 Home Folder		www.				
🖾 Filesystem Root	and view 1.0					
🖾 thinclient_drives 🛛 🛔	arduino-1.8. 19	arduino-1.8. 19-				
Documents		linuxarm.t				
▼ Ł Downloads						
0 items				<b></b>	40.0.C.D./T	
2 items				Free space	:e: 48.9 GIB (1	Total: 58.2 GiB)
		arduino	-1.8.19			~ ^ X
<u>File Edit View Sort Go</u>	lools					
		$\rightarrow \uparrow$ /hc	ome/pi/Down	loads/arduino	-1.8.19	•
觰 Home Folder						
Filesystem Root	examples	hardware	java	lib	libraries	tools
thinclient_drives	cxampico		Java	-		
		<b>••</b>	$\sim$	· <b>Q</b>	<b>.</b>	
■ Documents ■ Downloads	tools-builder	arduino	arduino- builder	arduino- linux-	install.sh	revisions.txt
			o and o	setup.sh		
examples	0					
A hardware	uninstall.sh					
	announon					
13 items				Free spac	e: 48.9 GiB (T	otal: 58.2 GiB)

		libr	aries			~ ^ >	ĸ
<u>File Edit View Sort Go</u>	Too <u>l</u> s						
	🖬 🤶	ightarrow $ ightarrow$ /h	ome/pi/Down	loads/arduin	o-1.8.19/librar	ries	-
🖌 Home Folder 📖 Filesystem Root							Î
☐ thinclient_drives ▲	Adafruit_Cir cuit_Playgro	Bridge	Esplora	Ethernet	examples	Firmata	
▶ <mark>i</mark> java ▶ ib	und						l
✓ libraries Adafrui	GSM	Keyboard	LiquidCrysta I	Mouse	Robot_Contr ol	RobotIRrem ote	l
▶ Bridge							
Esplora	Robot_Moto	SD	Servo	SpacebrewY	Stepper	Temboo	•
21 items				Free spa	ce: 48.9 GiB (T	Total: 58.2 GiB)	

**Step 2 :** Copy and paste the Arduino C library ZIP file (the default is ZIP file) from the provided Arduino Libraries folder into the libraries file opened in the first stepthe route is/home/pi/Downloads/arduino-1.8.19/libraries.

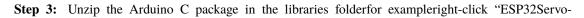
Click on the link to download the Libraries file: Download Libraries file



Arduino Libraries

Share View

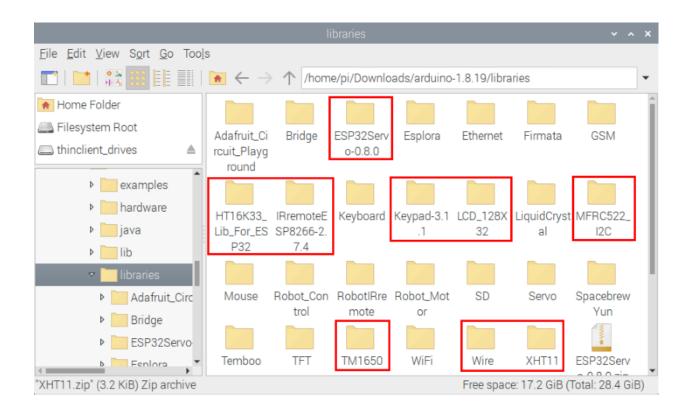
Share View							
📙 « Arduino-Libraries						· · · · · · · · · · · · · · · · · · ·	~
Name	D	ate modified		Туре	Size		
ESP32Servo-0.8.0	4,	/22/2022 9:53	AM	WinRAR ZIF	·	24 KB	
HT16K33_Lib_For_ESP32	4,	/22/2022 9:53	AM	WinRAR ZIF	·	7 KB	
IRremoteESP8266-2.7.4	4,	/22/2022 9:53	AM	WinRAR ZIF	·	665 KB	
🏧 Keypad-3.1.1	4,	/22/2022 9:53	AM	WinRAR ZIF	·	31 KB	
LCD_128X32	4,	/22/2022 9:53	AM	WinRAR ZIF	·	5 KB	
MFRC522_12C	4,	/22/2022 9:52	AM	WinRAR ZIF	·	27 KB	
TM1650	4,	/22/2022 9:52	AM	WinRAR ZIF	·	471 KB	
🚾 Wire	4,	/22/2022 9:52	AM	WinRAR ZIF	·	14 KB	
🚾 XHT11	4,	/22/2022 10:03	AM	WinRAR ZIF	·	4 KB	
		libraries				~ ^	×
<u>F</u> ile <u>E</u> dit <u>V</u> iew S <u>o</u> rt <u>G</u> o Too <u>l</u> s							
💼   🔛   👬 🧱 🔡   🖬	$\epsilon \leftrightarrow \pm$ /hc	me/pi/Downlo	ads/arduind	-1.8.19/libra	ies		Ŧ
觰 Home Folder							Î
Filesystem Root	.dafruit_Ci Bridge	Esplora	Ethernet	Firmata	GSM	Keyboard	I
	uit_Playg	Lopiora	Eulenier	Timata	00111	Reyboard	I
naranara	round						I
▶ 📄 java							I
ا 📄 lib Li	iquidCryst Mouse	Robot_Con	RobotIRre	Robot_Mot	SD	Servo	1
👻 📃 libraries	al	trol	mote	or		_	1
Adafruit_Circ							1
▶ 🔜 Bridge S	pacebrew Temboo	TFT	WiFi	ESP32Serv			1
▶ Esplora	Yun			o-0.8.0.zip			
▶ Ethernet	<b>E</b>	*	1	1	P32.zip	7.4.zip	
Firmata					101 <b>-</b>		
▶ GSM		X MFRC522_		Wire.zip	xht11.zip		
<	.1.zip 32.zip	I2C.zip	р				Ŧ
27 items				Free space	:: 17.2 GiB (	Total: 28.4 GiB)	)





0.8.0.zip"file <sup>0-0.8.0.zip</sup>, select and tap "Extract Here"to unzip the "ESP32Servo-0.8.0.zip"file. Similarly, unzip the remaining library files in the same way.). So you can see all the decompressed Arduino C library files.

		I	ibraries				~ ^ X
File Edit View Sort Go Tool	s						
📑   📑   👬 🔛 🗄 📰	$\blacktriangleright \leftarrow \rightarrow$	1/hom	e/pi/Downlo	ads/arduind	-1.8.19/libra	ries	•
🔶 Home Folder							Î
Filesystem Root	Adafruit_Ci	Bridge	Esplora	Ethernet	Firmata	GSM	Keyboard
☐ thinclient_drives ≜	rcuit_Playg round	5					,
▶ java	Touriu						
▶ <mark>ib</mark>	LiquidCryst	Mouse	Robot Con	RobotiBre	Robot_Mot	SD	Servo
👻 📃 libraries	al	modec	trol	mote	or	00	
Adafruit_Circ					WWA =	WWW.=	
Bridge	Spacebrew	Temboo	TFT	WiFi		HT16K33_	
▶ <mark>Esplora</mark>	Yun				o-0.8.0.zip	Lib_For_ES P32.zip	SP8266-2. 7.4.zip
▶ Ethernet							
▶ Firmata	Keypad-3.1	LCD_128X	MFRC522_		Wire.zip	xht11.zip	
► GSM	.1.zip	32.zip	l2C.zip	р			Ļ
27 items					Free space	e: 17.2 GiB (T	Total: 28.4 GiB)
		I	ibraries				~ ^ X
File Edit View Sort Go Tool							
				ads/arduinc			
				ads/arduind			
Home Folder	Adafruit_Ci			ads/arduind Ethernet			
	A ← →	↑ /hom	e/pi/Downlo		o-1.8.19/libra	ries	~ ^ ×
Home Folder Filesystem Root thinclient_drives	Adafruit_Ci rcuit_Playg	↑ /hom	e/pi/Downlo		o-1.8.19/libra	ries	~ ^ ×
Home Folder Filesystem Root thinclient_drives	Adafruit_Ci rcuit_Playg round	↑ /hom Bridge	e/pi/Downlo	Ethernet	-1.8.19/libra	ries GSM	~ ^ ×
Home Folder Filesystem Root thinclient_drives	Adafruit_Ci rcuit_Playg round	↑ /hom Bridge	e/pi/Downlo Esplora	Ethernet	-1.8.19/libra	ries GSM	Keyboard
<ul> <li>Home Folder</li> <li>Filesystem Root</li> <li>thinclient_drives</li> <li>java</li> <li>lib</li> <li>lib</li> <li>Adafruit_Circ</li> </ul>	Adafruit_Ci rcuit_Playg round	↑ /hom Bridge	e/pi/Downlo Esplora	Ethernet RobotlRre	-1.8.19/libra Firmata Robot_Mot or Op	ries GSM	Keyboard Servo
<ul> <li>Home Folder</li> <li>Filesystem Root</li> <li>thinclient_drives</li> <li>java</li> <li>lib</li> <li>lib</li> <li>Adafruit_Circ</li> <li>Bridge</li> </ul>	<ul> <li>Adafruit_Circuit_Playg</li> <li>LiquidCryst</li> <li>Spacebrew</li> </ul>	↑ /hom Bridge	e/pi/Downlo Esplora	Ethernet RobotlRre	P-1.8.19/libra	ries GSM SD SD	Keyboard Servo
<ul> <li>Home Folder</li> <li>Filesystem Root</li> <li>thinclient_drives</li> <li>ibava</li> <li>lib</li> <li>lib</li> <li>libraries</li> <li>Adafruit_Circ</li> <li>Bridge</li> <li>Esplora</li> </ul>	<ul> <li>Adafruit_Ci rcuit_Playg round</li> <li>LiquidCryst al</li> </ul>	↑ /hom Bridge	e/pi/Downlo Esplora Robot_Con trol	Ethernet RobotIRre mote	P-1.8.19/libra	ries GSM SD	Keyboard
<ul> <li>Home Folder</li> <li>Filesystem Root</li> <li>thinclient_drives</li> <li>java</li> <li>lib</li> <li>lib</li> <li>libraries</li> <li>Adafruit_Circ</li> <li>Bridge</li> <li>Esplora</li> <li>Ethernet</li> </ul>	<ul> <li>Adafruit_Circuit_Playg</li> <li>LiquidCryst</li> <li>Spacebrew</li> </ul>	↑ /hom Bridge	e/pi/Downlo Esplora Robot_Con trol	Ethernet RobotIRre mote	P-1.8.19/libra Firmata Robot_Mot or Op ESP3 Ard Op ESP3 Ard Op	ries GSM SD ben chiver ben With	Keyboard Servo
<ul> <li>Home Folder</li> <li>Filesystem Root</li> <li>thinclient_drives</li> <li>java</li> <li>lib</li> <li>lib</li> <li>libraries</li> <li>Adafruit_Circ</li> <li>Bridge</li> <li>Esplora</li> <li>Ethernet</li> <li>Firmata</li> </ul>	<ul> <li>Adafruit_Circuit_Playg</li> <li>LiquidCryst</li> <li>Spacebrew</li> <li>Yun</li> <li>Keypad-3.1</li> </ul>	/hom Bridge Bridge Mouse Temboo LCD_128X	e/pi/Downlo Esplora Robot_Con trol TFT	Ethernet RobotIRre mote WiFi WiFi	P-1.8.19/libra Firmata Robot_Mot or Op ESP3 Arc Op ESP3 Arc Op ESP3 Con Op	ries GSM SD sD chiver chiver ven With tract To tract Here	Keyboard Servo
<ul> <li>Home Folder</li> <li>Filesystem Root</li> <li>thinclient_drives</li> <li>java</li> <li>lib</li> <li>lib</li> <li>lib</li> <li>Adafruit_Circ</li> <li>Bridge</li> <li>Esplora</li> <li>Ethernet</li> <li>Firmata</li> </ul>	<ul> <li>▲ ← →</li> <li>Adafruit_Circuit_Playg</li> <li>round</li> <li>LiquidCryst</li> <li>al</li> <li>Spacebrew</li> <li>Yun</li> <li>Keypad-3.1</li> <li>.1.zip</li> </ul>	↑ /hom Bridge Bridge Mouse Temboo	e/pi/Downlo Esplora Robot_Con trol TFT	Ethernet RobotIRre mote WiFi	P-1.8.19/libra	ries GSM SD sD chiver chiver ven With tract To tract Here	Keyboard Servo



#### CHAPTER

## NINE

# **RASPBERRYPI ARDUINO**

Click on the link to enter the Raspberry Pi Arduino IDE tutorial: Raspberry Pi Arduino IDE Tutoria

# 9.1 Download code files and Libraries files

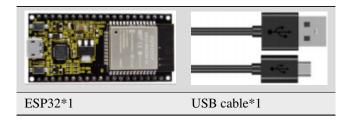
Click on the link to download the code file: Download Arduino C Codes file Click on the link to download the Libraries file: Download Libraries file

# 9.2 Project 01: Hello World

## 9.2.1 1.Introduction

For ESP32 beginners, we'll start with some simple things. In this project, you just need an ESP32 mainboard, USB cable and Raspberry Pi to complete "Hello World!" Project. It is not only a communication test for ESP32 mainboard and Raspberry Pi, but also a primary project for ESP32.

## 9.2.2 2.Components



# 9.2.3 3.Components

In this project, we use a USB cable to connect the ESP32 to the Raspberry Pi.



## 9.2.4 4.Project code

You can open the code we provide If you haven't downloaded the code file, please click on the link to download itDownload Arduino C Codes file

The code used in this project is saved in folder(path :) "Arduino-Codes\Project 01Hello World\Project\_01\_Hello\_World".

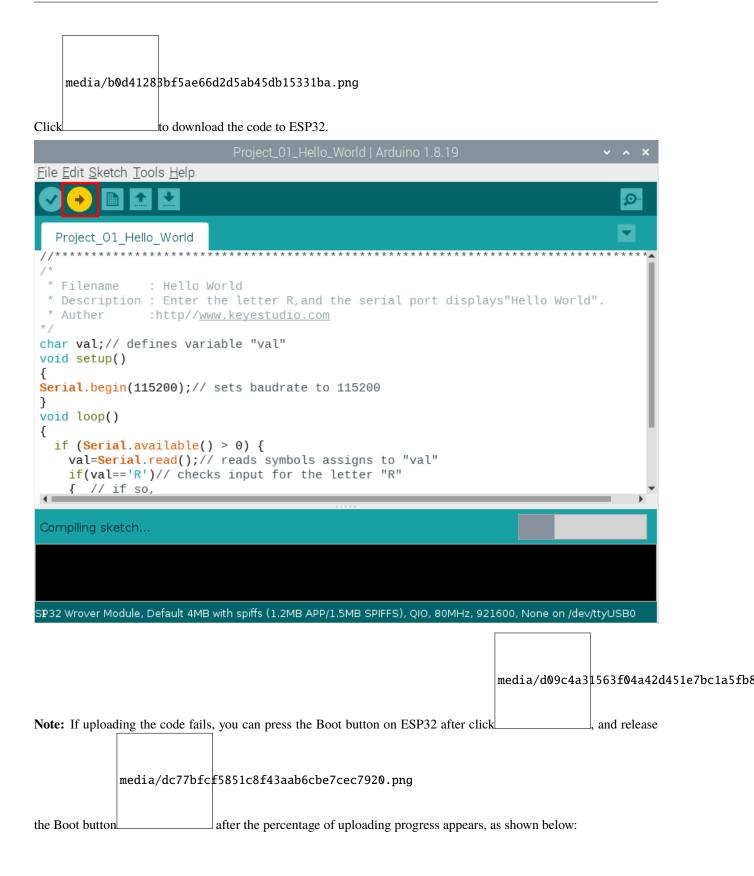
```
/*
* Filename : Hello World
* Description : Enter the letter R, and the serial port displays"Hello World".
* Auther :http//www.keyestudio.com
*/
char val;// defines variable "val"
void setup()
{
Serial.begin(115200);// sets baudrate to 115200
}
void loop()
{
 if (Serial.available() > 0) {
  val=Serial.read();// reads symbols assigns to "val"
  if(val=='R')// checks input for the letter "R"
  { // if so,
   Serial.println("Hello World!");// shows "Hello World !".
  }
 }
}
```

Before uploading the project code to ESP32click "Tools"→"Board" and select"ESP32 Wrover Module".

_	Project_01_Hello_World   Arduino 1.8.19	¥ ^	×	
e <u>E</u> dit <u>S</u> ketch	iools Help			
	Auto Format	Ctrl+T		
	Archive Sketch			
Project_01_H	Fix Encoding & Reload			
	Manage Libraries	Ctrl+Shift+I		
	Serial Monitor	Ctrl+Shift+M	-	Δ
Descriptic Auther	Serial Plotter	Ctrl+Shift+L	-	ESP32 Dev Module
ar val·// ()	WiFi101 / WiFiNINA Firmware Updater		-	<ul> <li>ESP32 Wrover Module</li> </ul>
id setup()			Statistics in the local of the	ESP32 Pico Kit
rial benin	Board: "ESP32 Wrover Module"	>	Boards Manager	TinyPICO
	Upload Speed: "921600"	>	Arduino AVR Boards >	S.ODI Ultra v1
Ld loop()	Flash Frequency: "80MHz"	>	ESP32 Arduino >	MagicBit
if (Serial.	Flash Mode: "QIO"	>		Turta IoT Node
val= <mark>Seria</mark> if(val=='	Partition Scheme: "Default 4MB with spiffs (1.2MB APP/1.5MB SPIFFS)"	>		TTGO LoRa32-OLED V1
	Core Debug Level: "None"	>	.*	TTGO T1
	Port: "/dev/ttyUSB0"	>		TTG0 T7 V1.3 Mini32
	Get Board Info		and the second s	TTG0 T7 V1.4 Mini32
or downloa	Programmer	>		XinaBox CW02
	Burn Bootloader		and the second second	SparkFun ESP32 Thing
	), Default 4MB with spiffs (1.2MB APP/1.5MB SPIFFS), QIO, 80MHz, 921600, None	no Monattul ISBA		SparkFun ESP32 Thing Plus

# Select the serial port.

Project_01_Hello_World   Arduino 1.8.19	~ ^ X
Eile Edit Sketch Tools Help	
Auto Format	Ctrl+T
Archive Sketch	
Project_01_F Fix Encoding & Reload	
//*********** Manage Libraries	Ctrl+Shift+I
* Filename Serial Monitor	Ctrl+Shift+M
* Descriptio * Auther Serial Plotter	Ctrl+Shift+L
*/ char val;// c WiFi101 / WiFiNINA Firmware Updater	
( Board: "ESP32 Wrover Module"	>
Serial.begin( Upload Speed: "921600"	>
void loop() Flash Frequency: "80MHz"	>
if (Serial, Flash Mode: "QIO"	>
<pre>val=Seria if(val==) Partition Scheme: "Default 4MB with spiffs (1.2MB APP/1.5MB SPIF)</pre>	FS)" >
{ // if Core Debug Level: "None"	>
Port: "/dev/ttyUSB0"	>
Get Board Info	
Error downloa Error downloa Programmer	, ~
Burn Bootloader	
P32 Wrover Module, Default 4MB with spiffs (1.2MB APP/1.5MB SPIFFS), QIO, 80MHz, 921600,	None on /dev/ttyUSB0



media/157ee2e7687559d9812d24edec758150.png

The Project code is uploaded successfully

# 9.2.5 5.Project result

media/2f6bca5	6f724e45a855335cb53a

After the project code is uploaded successfully, power up with a USB cable and click the icon to enter the serial monitor.

Set baud rate to 115200 and type "R" in the text box. Click "Send", and the serial monitor will display "Hello World!".

(Note: If you enter"R" in the text box and click"Send", the serial monitor does not print"Hello World!", you need to press the RESET button on the ESP32 mainboard and repeat the above operation.)

media/1fd21fa	fd84d2b529931a89d21a03d6a	nna
meara/ mazira		• prig

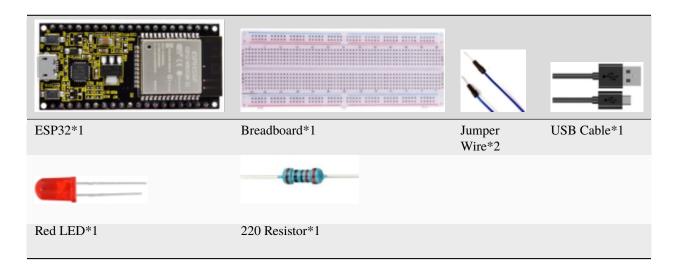
Uploading	(
Hash of data verified.	^
Compressed 17168 bytes to 10937	
Wrote 17168 bytes (10937 compressed) at 0x00001000 in 0.2 seconds (effective 832.4 kbit/s)	
Hash of data verified.	
Compressed 232464 bytes to 113899	
Writing at 0x00024000 (85 %)	¥
< >>	
1 ESP32 Wrover Module, Default, QIO, 80MHz, 921600, None on COM3	

# 9.3 Project 02: Turn On LED

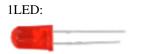
## 9.3.1 1.Introduction

In this project, we will show you how to light up the LED. We use the ESP32's digital pin to turn on the LED so that the LED is lit up.

# 9.3.2 2.Components



# 9.3.3 3.Component knowledge

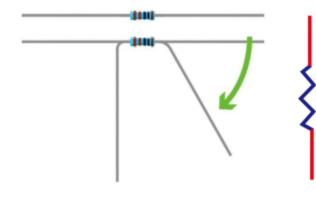


The LED is a semiconductor known as "light-emitting diode", which is an electronic device made from semiconducting materials(silicon, selenium, germanium, etc.). It has an anode and a cathode, the short lead is cathode, which connects to GND; the long lead is anode, which connects to 3.3V or 5V.

media/f70404aa49540fd7aecae944c7c01f83.jpeg

2Five-color ring resistor

A resistor is an electronic component in a circuit that restricts or regulates the flow current flow. On the left is the appearance of the resistor and on the right is the symbol for the resistance in the circuit. Its unit is(). 1 m = 1000 k1k = 1000.



We can use resistors to protect sensitive components, such as LED. The strength of the resistance is marked on the body of the resistor with an electronic color code. Each color code represents a number, and you can refer to it in a resistance card.

-Color 1 – 1st Digit.

-Color 2 - 2nd Digit.

-Color 3 – 3rd Digit.

-Color 4 – Multiplier.

-Color 5 - Tolerance.

media/c3df005312cd9f6d4cdae6abf3cddb83.png

In this kit, we provide three Five-color ring resistor with different resistance values. Take three Five-color ring resistor as an example.

220 Resistor\*10

media/55c0199544e9819328f6d5778f10d7d0.png

10K Resistor\*10

media/246cf3885dc837c458a28123885c9f7b.png

1K Resistor\*10

media/19f5dfc51adfd79b04c3b164529767ed.png

In the same voltage, there will be less current and more resistance. The connection between current(I), voltage(V), and resistance<sup>®</sup> can be expressed by the formula: I=U/R. In the figure below, if the voltage is 3V, the current through R1 is: I = U / R = 3 V / 10 K = 0.0003A = 0.3mA.

media/b3eec552e4dfad361833730698621776.png

Don't connect a low resistance directly to the two poles of the power supply. as this will cause excessive current to damage the electronic components. Resistors do not have positive and negative poles.

3Bread board

Breadboards are used to build and test circuits quickly before completing any circuit design. There are many holes in the breadboard that can be inserted into circuit components such as integrated circuits and resistors. A typical breadboard is shown below

media/612c1381811b2d780d5f6ed6a7ec3701.png

The breadboard has strips of metal, which run underneath the board and connect the holes on the top of the board. The metal strips are laid out as shown below. Note that the top and bottom rows of holes are connected horizontallywhile the remaining holes are connected vertically.

media/b45e70b961537035c85878b73d371725.png

The first two rows (top) and the last two rows (bottom) of the breadboard are used for the positive pole (+) and negative pole (-) of the power supply respectively. The conductive layout of the breadboard is shown in the figure below:

media/d5478bd5eac558252cbc235479d979eb.png

When we connect DIP (Dual In-line Packages) components, such as integrated circuits, microcontrollers, chips and so on, we can see that a groove in the middle isolates the middle part, so the top and bottom of the groove is not connected. DIP components can be connected as shown in the following diagram:

media/50caf14e911c4244779e99445c658db6.png

media/9b66ae2199e77fbc99b7b278dac0b567.png

Power Supply

In this project, we connected the ESP32 to the Raspberry Pi by using USB cable.



# 9.3.4 4.Wiring diagram

First, disconnect all power from the ESP32. Then build the circuit according to the wiring diagram. After the circuit is built and verified correct, connect the ESP32 to the Raspberry Pi by using a USB cable.

Note: Avoid any possible short circuits (especially connecting 3.3V and GND)!

**WARNING:** A short circuit can cause high current in your circuit, create excessive component heat and cause permanent damage to your hardware!

media/0735997593c8858ad6441d8e9867206f.png

Note:

How to connect a LED

media/42ff6f405dfa128593827de5aa03e94b.png

How to identify the 220 Five-color ring resistor

media/55c0199544e9819328f6d5778f10d7d0.png

# 9.3.5 5.Project code

You can open the code we provideIf you haven't downloaded the code file, please click on the link to download itDownload Arduino C Codes file

The code used in this project is saved in folder(path:) "Arduino-Codes\Project 02Turn On LED\Project\_02\_Turn\_On\_LED".

```
/*
* Filename : Turn On LED
* Description : Make an led on.
* Auther : http//www.keyestudio.com
*/
#define LED_BUILTIN 15
// the setup function runs once when you press reset or power the board
void setup() {
 // initialize digital pin LED_BUILTIN as an output.
 pinMode(LED_BUILTIN, OUTPUT);
}
void loop() {
 digitalWrite(LED_BUILTIN, HIGH); // turn the LED on (HIGH is the voltage level)
}
```

Before uploading the project code to ESP32click "Tools"→"Board" and select"ESP32 Wrover Module".

media/fe73bbd726dbe38bb24f9c0d735c90ec.png

Select the serial port.

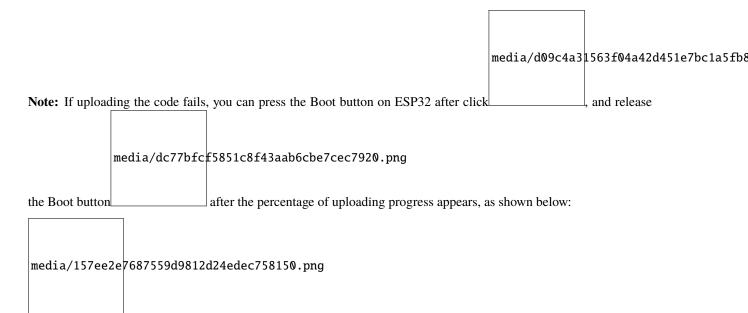
	Project_02_Turn_On_LED   Arduino 1.8.19	× •	×	
e <u>E</u> dit <u>S</u> ketch	Tools Help			
00 🖬	Auto Format	Ctrl+T		
	Archive Sketch			
Project_02_T	Fix Encoding & Reload			
	Manage Libraries	Ctrl+Shift+I	î	
	Serial Monitor	Ctrl+Shift+M	and the second second	Δ
Descriptic Auther	Serial Plotter	Ctrl+Shift+L	and the second second	ESP32 Dev Module
fine LED C	WiFi101 / WiFiNINA Firmware Updater		-	<ul> <li>ESP32 Wrover Module</li> </ul>
			Statements in the local division of	ESP32 Pico Kit
	Board: "ESP32 Wrover Module"	,	Boards Manager	TinyPICO
<pre>// initiali</pre>	Upload Speed: "921600"	>	Arduino AVR Boards	S.ODI Ultra v1
	Flash Frequency: "80MHz"	>	ESP32 Arduino	> MagicBit
id loop() {	Flash Mode: "QIO"	>		Turta IoT Node
digitalWrit	Partition Scheme: "Default 4MB with spiffs (1.2MB APP/1.5MB SPIFFS)"	>	, I	TTGO LoRa32-OLED V1
	Core Debug Level: "None"	>		TTGO T1
	Port: "/dev/ttyUSB0"	>		TTGO T7 V1.3 Mini32
	Get Board Info			TTGO T7 V1.4 Mini32
	Programmer	>		XinaBox CW02
	Bum Bootloader			SparkFun ESP32 Thing
Wrover Modu	e, Default 4MB with spiffs (1.2MB APP/1.5MB SPIFFS), QIO, 80MHz, 921600, None	on Identity ISB0		SparkFun ESP32 Thing Plus

# media/b0d41283bf5ae66d2d5ab45db15331ba.png

Click

#### to download the code to ESP32.

	Project_02_Turn_On_LED   Arduino 1.8.19	~ <i>.</i>	×	
<u>File E</u> dit <u>S</u> ketch	Tools Help			
	Auto Format	Ctrl+		
	Archive Sketch			
Project_02_T	Fix Encoding & Reload			
//*	Manage Libraries	Ctrl+Shift+	1	
* Filename	Serial Monitor	Ctrl+Shift+N	1	-
* Descriptio * Auther	Serial Plotter	Ctrl+Shift+	. 1	-
*/ #define LED_E	WiFi101 / WiFiNINA Firmware Updater			-
	Board: "ESP32 Wrover Module"	2		
<pre>void setup()     // initiali</pre>	Upload Speed: "921600"	3		
	Flash Frequency: "80MHz"	2		-
void loop() {	Flash Mode: "QIO"	3		
digitalWrit	Partition Scheme: "Default 4MB with spiffs (1.2MB APP/1.5MB SPIFFS)"	2		
//*********	Core Debug Level: "None"	3		
	Port: "/dev/ttyUSB0"	\$		Serial ports
	Get Board Info			/dev/ttyAMA0
	Programmer	2	~	/dev/ttyUSB0
	Burn Bootloader			
S₽32 Wrover Modu	e, Default 4MB with spiffs (1.2MB APP/1.5MB SPIFFS), QIO, 80MHz, 921600, None	on /dev/ttyUSE	0	

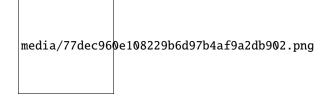


The Project code is uploaded successfully

Project_02_Turn_On_LED   Arduino 1.8.19	~ ^ X
<u>File Edit Sketch Tools H</u> elp	
	<mark>,⊘</mark> ,
Project_02_Turn_On_LED	
<pre>//***********************************</pre>	ĺ
<pre>// the setup function runs once when you press reset or power the board void setup() { // initialize digital pin LED_BUILTIN as an output. pinMode(LED_BUILTIN, OUTPUT); } void loop() { digitalWrite(LED_BUILTIN, HIGH); // turn the LED on (HIGH is the voltage }</pre>	level)
4	••••••••••••••••••••••••••••••••••••••
Done uploading.	
Leaving Hard resetting via RTS pin	Ĵ
SP32 Wrover Module, Default 4MB with spiffs (1 2MB APP/1 5MB SPIEFS), OIO, 80MHz, 921600, None on /dev	

# 9.3.6 6.Project result

After the project code was uploaded successfully, power up with a USB cable and the LED is lit up.



# 9.4 Project 03LED Flashing

## 9.4.1 Introduction

In this project, we will show you the LED flashing effect. We use the ESP32's digital pin to turn on the LED and make it flashing.

## 9.4.2 Components

ESP32*1	Breadboard*1		
	-((111))	$\mathbf{X}$	
Red LED*1	220 Resistor*1	Jumper Wire*2	USB Cable*1

# 9.4.3 Wiring diagram

First, disconnect all power from the ESP32. Then build the circuit according to the wiring diagram. After the circuit is built and verified correct, connect the ESP32 to your computer using a USB cable.

Note: Avoid any possible short circuits (especially connecting 3.3V and GND)!

**WARNING:** A short circuit can cause high current in your circuit, create excessive component heat and cause permanent damage to your hardware!

media/0735997593c8858ad6441d8e9867206f.png

Note:

How to connect a LED

media/42ff6f405dfa128593827de5aa03e94b.png

How to identify the 220 Five-color ring resistor

media/55c0199544e9819328f6d5778f10d7d0.png

### 9.4.4 Project code

You can open the code we provide: If you haven't downloaded the code file, please click on the link to download itDownload Arduino C Codes file

The code used in this project is saved in folder "Arduino-Codes\Project 03LED Flashing\Project\_03\_LED\_Flashing".

```
/*
* Filename : External LED flashing
* Description : Make an led blinking.
* Auther : http//www.keyestudio.com
*/
#define PIN_LED 15 //define the led pin
// the setup function runs once when you press reset or power the board
void setup() {
 // initialize digital pin LED as an output.
 pinMode(PIN_LED, OUTPUT);
}
// the loop function runs over and over again forever
void loop() {
 digitalWrite(PIN_LED, HIGH); // turn the LED on (HIGH is the voltage level)
                              // wait for 0.5s
 delay(500);
 digitalWrite(PIN_LED, LOW);
                            // turn the LED off by making the voltage LOW
                               // wait for 0.5s
 delay(500);
```

(continues on next page)

(continued from previous page)

Before uploading Project Code to ESP32, please check the configuration of Arduino IDE.

Click "Tools" to confirm the board type and port as shown below:

	Project_03_LED_Flashing   Arduino 1.8.19	~ ^	×
<u>F</u> ile <u>E</u> dit <u>S</u> ketch	Tools Help		
	Auto Format	Ctrl+T	
	Archive Sketch		
Project_03_L	Fix Encoding & Reload		
/*	Manage Libraries	Ctrl+Shift+I	1
* Filename	Serial Monitor	Ctrl+Shift+M	
* Description * Auther	Serial Plotter	Ctrl+Shift+L	
*/ #define PIN_L	WiFi101 / WiFiNINA Firmware Updater		
	Board: "ESP32 Wrover Module"	>	11
<pre>void setup()     // initial:</pre>	Upload Speed: "921600"	>	ш
pinMode(PI	Flash Frequency: "80MHz"	>	
}	Flash Mode: "QIO"	>	
<pre>// the loop ' void loop() ·</pre>	Partition Scheme: "Default 4MB with spiffs (1.2MB APP/1.5MB SPIFFS)"	>	
digitalWrit	Core Debug Level: "None"	>	
	Port: "/dev/ttyUSB0"	>	
	Get Board Info		
	Programmer	>	
	Burn Bootloader		
S₽32 Wrover Modu	le, Default 4MB with spiffs (1.2MB APP/1.5MB SPIFFS), QIO, 80MHz, 921600, None	on /dev/ttyUSB0	)

media/b0d41288bf5ae66d2d5ab45db15331ba.png

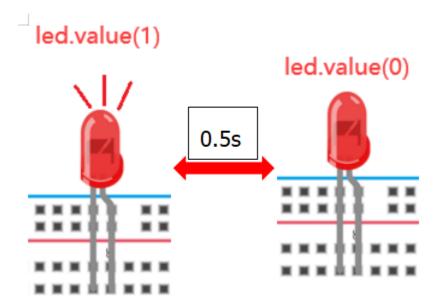
Click to download the project code to ESP32.

	Project_03_LED_Flashing   Ard	uino 1.8.19	~ ^ X
<u>F</u> ile <u>E</u> dit <u>S</u> ketch <u>T</u> ools <u>H</u> elp			
			an a
Project_03_LED_Flashing			
	*****	*****	•
/* * Filename : Extern	al LED flashing		
* Description : Make a * Auther : http//	n led blinking.		
*/			
#define PIN_LED 15	-		
<pre>// the setup function r void setup() {</pre>	uns once when you press res	et or power the board	
<pre>// initialize digital pinMode(PIN_LED, OUTP</pre>	pin LED as an output. UT):		
}			
	ns over and over again fore	ver	
<pre>void loop() {     digitalWrite(PIN LED,</pre>	HIGH); // turn the LED o	on (HIGH is the voltage	level) -
		_	
Compiling sketch			
SP32 Wrover Module, Default 4ME	with spiffs (1.2MB APP/1.5MB SPIFFS	), QIO, 80MHz, 921600, None on	/dev/ttyUSB0
		media/d090	24a31563f04a42d451e7bc
Note: If uploading the code fail	s, you can press the Boot button on	ESP32 after click	, and release
media/dc77bfc	f5851c8f43aab6cbe7cec7920.p	ng	
the Boot button	after the percentage of uploading p	rogress appears, as shown belo	w:
media/157ee2e7687559d9812	2d24edec758150 ppg		
	an reace, 20120, bud		
The Project code is uploaded suc	cessfully		

Project_03_LED_Flashing   Arduino 1.8.19 🗸	~ ×
<u>File Edit Sketch Tools H</u> elp	
	<b>9</b> 0-
Project_03_LED_Flashing	
<pre>//***********************************</pre>	Î
#define PIN_LED 15 //define the led pin	- 1
<pre>// the setup function runs once when you press reset or power the board void setup() {     // initialize digital pin LED as an output.     pinMode(PIN_LED, OUTPUT); }</pre>	
<pre>// the loop function runs over and over again forever void loop() {     digitalWrite(PIN LED, HIGH); // turn the LED on (HIGH is the voltage level) </pre>	
Done uploading.	
Leaving Hard resetting via RTS pin	Î
▲ SP32 Wrover Module, Default 4MB with spiffs (1.2MB APP/1.5MB SPIFFS), QIO, 80MHz, 921600, None on /dev/ttyU	SB0

# 9.4.5 Project result

After the project code was uploaded successfully, power up with a USB cable and the LED start flashing.



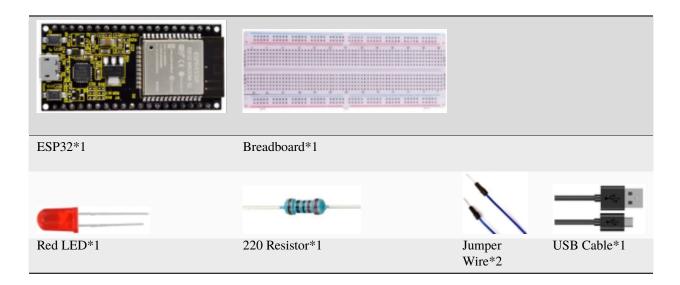
# 9.5 Project 04: Breathing Led

# 9.5.1 Introduction

In previous studies, we know that LEDs have on/off state, so how to enter the intermediate state? How to output an intermediate state to make the LED half bright? That's what we're going to learn.

Breathing light, that is, LED is turned from off to on gradually, and gradually from on to off, just like "breathing". So, how to control the brightness of a LED? We will use ESP32's PWM to achieve this target.

# 9.5.2 Components



## 9.5.3 Component knowledge



#### **Analog & Digital**

An Analog Signal is a continuous signal in both time and value. On the contrary, a Digital Signal or discrete time signal is a time series consisting of a sequence of quantities. Most signals in life are analog signals. A familiar example of an Analog Signal would be how the temperature throughout the day is continuously changing and could not suddenly change instantaneously from  $0^{\circ}$ C to  $10^{\circ}$ C. However, Digital Signals can instantaneously change in value. This change is expressed in numbers as 1 and 0 (the basis of binary code). Their differences can more easily be seen when compared when graphed as below.

media/4bdf6127e563b453a1fd8953b4ebb277.png

In practical application, we often use binary as the digital signal, that is a series of 0's and 1's. Since a binary signal only has two values (0 or 1), it has great stability and reliability. Lastly, both analog and digital signals can be converted into the other.

#### PWM

PWM, Pulse-Width Modulation, is a very effective method for using digital signals to control analog circuits. Common processors cannot directly output analog signals. PWM technology makes it very convenient to achieve this conversion (translation of digital to analog signals).

PWM technology uses digital pins to send certain frequencies of square waves, that is, the output of high levels and low levels, which alternately last for a while. The total time for each set of high levels and low levels is generally fixed, which is called the period (Note: the reciprocal of the period is frequency). The time of high level outputs are generally called "pulse width", and the duty cycle is the percentage of the ratio of pulse duration, or pulse width (PW) to the total period (T) of the waveform.

The longer the output of high levels last, the longer the duty cycle and the higher the corresponding voltage in the analog signal will be. The following figures show how the analog signal voltages vary between 0V-3V3 (high level is 3V3) corresponding to the pulse width 0%-100%:

media/a439e1bd8a4578b43b7188c821d58594.jpeg

The longer the PWM duty cycle is, the higher the output power will be. Now that we understand this relationship, we can use PWM to control the brightness of an LED or the speed of DC motor and so on. It is evident from the above

that PWM is not real analog, and the effective value of the voltage is equivalent to the corresponding analog. So, we can control the output power of the LED and other output modules to achieve different effects.

#### ESP32 and PWM:

On ESP32, the LEDC(PWM) controller has 16 separate channels, each of which can independently control frequency, duty cycle, and even accuracy. Unlike traditional PWM pins, the PWM output pins of ESP32 are configurable, with one or more PWM output pins per channel. The relationship between the maximum frequency and bit precision is shown in the following formula, where the maximum value of bit is 31.

media/f79af745d3c726ee5ca07932d2ca6d5e.png

For example, generate a PWM with an 8-bit precision (28=256. Values range from 0 to 255) with a maximum frequency of 80,000,000/255 = 312,500Hz.

### 9.5.4 Wiring diagram

media/0735997593c8858ad6441d8e9867206f.png

Note:

How to connect a LED

media/42ff6f405dfa128593827de5aa03e94b.png

How to identify the 220 Five-color ring resistor

media/55c0199544e9819328f6d5778f10d7d0.png

## 9.5.5 Project code

The design of this project makes the GP15 output PWM, and the pulse width gradually increases from 0% to 100%, and then gradually decreases from 100% to 0%.

You can open the code we provideIf you haven't downloaded the code file, please click on the link to download itDownload Arduino C Codes file

The code used in this project is saved in folder "Arduino-Codes\Project 04Breathing Led\Project\_04\_Breathing\_Led".

```
/*
* Filename : Breathing Led
* Description : Make led light fade in and out, just like breathing.
* Auther : http//www.keyestudio.com
*/
#define PIN_LED 15 //define the led pin
#define CHN
              0 //define the pwm channel
#define FRQ
              1000 //define the pwm frequency
#define PWM_BIT 8
                  //define the pwm precision
void setup() {
 ledcSetup(CHN, FRQ, PWM_BIT); //setup pwm channel
 ledcAttachPin(PIN_LED, CHN); //attach the led pin to pwm channel
}
void loop() {
 for (int i = 0; i < 255; i++) { //make light fade in</pre>
   ledcWrite(CHN, i);
   delay(10);
 }
 for (int i = 255; i > -1; i--) { //make light fade out
   ledcWrite(CHN, i);
   delay(10);
 }
}
```

Before uploading Project Code to ESP32, please check the configuration of Arduino IDE.

Click "Tools" to confirm the board type and port as shown below:

### keyestudio WiKi

	Project_04_Breathing_Led   Arduino 1.8.19	~ ^ X
<u>F</u> ile <u>E</u> dit <u>S</u> ketch	Tools Help	
	Auto Format	Ctrl+T .
	Archive Sketch	
Project_04_E	Fix Encoding & Reload	
/*	Manage Libraries	Ctrl+Shift+I
* Filename	Serial Monitor	Ctrl+Shift+M
* Descriptio * Auther	Serial Plotter	Ctrl+Shift+L
*/ #define PIN_L	WiFi101 / WiFiNINA Firmware Updater	
	Board: "ESP32 Wrover Module"	>
<pre>#define PWM_E void setup()</pre>	Upload Speed: "921600"	>
ledcSetup(		>
}	Flash Mode: "QIO"	>
<pre>void loop() {</pre>	Partition Scheme: "Default 4MB with spiffs (1.2MB APP/1.5MB SPIFFS)"	>
for (int i	Core Debug Level: "None"	>
	Port: "/dev/ttyUSB0"	> <b>•</b>
	Get Board Info	
	Programmer	>
	Bum Bootloader	
SP32 Wrover Modu	le, Default 4MB with spiffs (1.2MB APP/1.5MB SPIFFS), QIO, 80MHz, 921600, None	on /dev/ttyUSB0

## media/b0d41283bf5ae66d2d5ab45db15331ba.png

to download the project code to ESP32.

Click

Project_04_Breathing_Led   Arduino 1.8.19	~ ^ X
<u>File Edit Sketch Tools Help</u>	
	<b>-</b> ₽
Project_04_Breathing_Led	
//************************************	***
<pre>* Filename : Breathing Led * Description : Make led light fade in and out, just like breathing. * Auther : http//<u>www.keyestudio.com</u> */</pre>	
<pre>#define PIN_LED 15 //define the led pin #define CHN 0 //define the pwm channel #define FRQ 1000 //define the pwm frequency #define PWM_BIT 8 //define the pwm precision void setup() { ledcSetup(CHN, FRQ, PWM_BIT); //setup pwm channel</pre>	
<pre>ledcAttachPin(PIN_LED, CHN); //attach the led pin to pwm channel }</pre>	
<pre>void loop() {    for (int i = 0; i &lt; 255; i++) { //make light fade in</pre>	, -
Compiling sketch	
SP32 Wrover Module, Default 4MB with spiffs (1.2MB APP/1.5MB SPIFFS), QIO, 80MHz, 921600, Nor	ne on /dev/ttyUSB0
media/	/d09c4a31563f04a42d451e7bc1a5fb8
Note: If uploading the code fails, you can press the Boot button on ESP32 after click	, and release
media/dc77bfcf5851c8f43aab6cbe7cec7920.png	
the Boot button after the percentage of uploading progress appears, as shown	below:
media/157ee2e7687559d9812d24edec758150.png	
The Project code is uploaded successfully	

Project_04_Breathing_Led   Arduino 1.8.19	~ ^ X
<u>Eile Edit Sketch Tools H</u> elp	
	<mark>,⊘</mark> -
Project_04_Breathing_Led	
V/************************************	1
<pre>/*  * Filename : Breathing Led  * Description : Make led light fade in and out, just like breathing.  * Auther : http//www.keyestudio.com  */</pre>	
<pre>#define PIN_LED 15 //define the led pin #define CHN 0 //define the pwm channel #define FRQ 1000 //define the pwm frequency #define PWM_BIT 8 //define the pwm precision void setup() {    ledcSetup(CHN, FRQ, PWM_BIT); //setup pwm channel    ledcAttachPin(PIN_LED, CHN); //attach the led pin to pwm channel }</pre>	
<pre>void loop() {    for (int i = 0; i &lt; 255; i++) { //make light fade in</pre>	
Done uploading.	
Leaving Hard resetting via RTS pin	Ĵ
S₽32 Wrover Module, Default 4MB with spiffs (1.2MB APP/1.5MB SPIFFS), QIO, 80MHz, 921600, None on /dev	/ttyUSB0

# 9.5.6 Project result

After the project code was uploaded successfully, power up with a USB cable and the LED is turned from ON to OFF and then back from OFF to ONgradually like breathing.

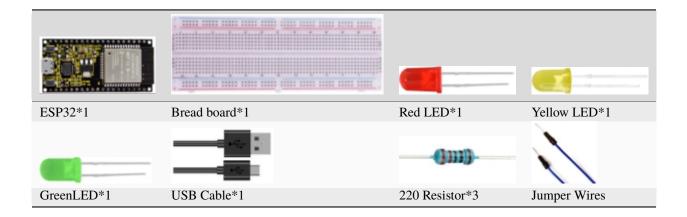


# 9.6 Project 05Traffic Lights

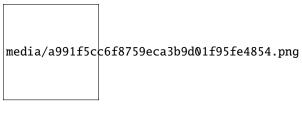
## 9.6.1 Introduction

Traffic lights are closely related to people's daily lives, which generally show red, yellow, and green. Everyone should obey the traffic rules, which can avoid many traffic accidents. In this project, we will use ESP32 and some LEDs (red, green and yellow) to simulate the traffic lights.

# 9.6.2 Components



# 9.6.3 Wiring diagram



Note:

How to connect a LED

media/42ff6f405dfa128593827de5aa03e94b.png

How to identify the 220 Five-color ring resistor

media/55c0199544e9819328f6d5778f10d7d0.png

# 9.6.4 Project code

You can open the code we provide: If you haven't downloaded the code file, please click on the link to download itDownload Arduino C Codes file

The code used in this project is saved in folder "Arduino-Codes\Project 05Traffic Lights\Project\_05\_Traffic\_Lights".

```
/*
* Filename
           : Traffic Lights
* Description : Simulated traffic lights.
* Auther
           : http//www.keyestudio.com
*/
#define PIN_LED_RED 0 //define the red led pin
#define PIN_LED_YELLOW 2 //define the yellow led pin
#define PIN_LED_GREEN 15 //define the green led pin
void setup() {
 pinMode(PIN_LED_RED, OUTPUT);
 pinMode(PIN_LED_YELLOW, OUTPUT);
 pinMode(PIN_LED_GREEN, OUTPUT);
}
void loop() {
  digitalWrite(PIN_LED_RED, HIGH);// turns on the red led
  delay(5000);// delays 5 seconds
  digitalWrite(PIN_LED_GREEN, LOW); // turns off the green led
  for(int i=0;i<3;i++)// flashes 3 times.</pre>
{
  delay(500);// delays 0.5 second
  digitalWrite(PIN_LED_YELLOW, HIGH);// turns on the yellow led
  delay(500);// delays 0.5 second
  digitalWrite(PIN_LED_YELLOW, LOW);// turns off the yellow led
}
  delay(500);// delays 0.5 second
  digitalWrite(PIN_LED_GREEN, HIGH);// turns on the green led
  delay(5000);// delays 5 second
  digitalWrite(PIN_LED_RED, LOW);// turns off the red led
}
```

Before uploading Project Code to ESP32, please check the configuration of Arduino IDE.

Click "Tools" to confirm the board type and port as shown below:

	Project_05_Traffic_Lights   Arduino 1.8.19	~ ^ X
<u>F</u> ile <u>E</u> dit <u>S</u> ketch	<u>Iools H</u> elp	_
	Auto Format	Ctrl+T
	Archive Sketch	
Project_05_T	Fix Encoding & Reload	
/*	Manage Libraries	Ctrl+Shift+I
* Filename	Serial Monitor	Ctrl+Shift+M
* Description * Auther	Serial Plotter	Ctrl+Shift+L
*/ #define PIN_L	WiFi101 / WiFiNINA Firmware Updater	
#define PIN_ #define PIN_	Board: "ESP32 Wrover Module"	>
<pre>void setup()     pinMode(PII</pre>	Upload Speed: "921600"	>
	Flash Frequency: "80MHz"	>
pinMode(PI pinMode(PI	Flash Mode: "QIO"	>
}	Partition Scheme: "Default 4MB with spiffs (1.2MB APP/1.5MB SPIFFS)"	>
void loop()	Core Debug Level: "None"	>
	Port: "/dev/ttyUSB0"	> <b>•</b>
	Get Board Info	
	Programmer	>
	Burn Bootloader	
SP32 Wrover Modu	le, Default 4MB with spiffs (1.2MB APP/1.5MB SPIFFS), QIO, 80MHz, 921600, None	on /dev/ttyUSB0

## media/b0d41283bf5ae66d2d5ab45db15331ba.png

\_\_\_\_\_to download the project code to ESP32.

Click

Project_05_Traffic_Lights   Arduino 1.8.19	~ ^ ×
<u>File Edit Sketch Tools H</u> elp	
	<b>₽</b>
Project_05_Traffic_Lights	
/*  * Filename : Traffic Lights	Î
<pre>* Description : Simulated traffic lights. * Auther : http//www.keyestudio.com */</pre>	
<pre>#define PIN_LED_RED 0 //define the red led pin #define PIN_LED_YELLOW 2 //define the yellow led pin #define PIN_LED_GREEN 15 //define the green led pin</pre>	
<pre>void setup() {     pinMode(PIN_LED_RED, OUTPUT);     pinMode(PIN_LED_YELLOW, OUTPUT);     pinMode(PIN_LED_GREEN, OUTPUT); }</pre>	
void loop() {	
	<b>,</b>
Compiling sketch	
SP32 Wrover Module, Default 4MB with spiffs (1.2MB APP/1.5MB SPIFFS), QIO, 80MHz, 921600, None on /dev	/ttyUSB0
SP32 Wrover Module, Default 4MB with spiffs (1.2MB APP/1.5MB SPIFFS), QIO, 80MHz, 921600, None on /dev	//ttyUSB0
	//ttyUSB0 1563f04a42d451e7bc1a51
media/d09c4a3	1563f04a42d451e7bc1a51
media/d09c4a3	1563f04a42d451e7bc1a51
Note: If uploading the code fails, you can press the Boot button on ESP32 after click media/dc77bfcf5851c8f43aab6cbe7cec7920.png	1563f04a42d451e7bc1a51
Note: If uploading the code fails, you can press the Boot button on ESP32 after click	1563f04a42d451e7bc1a51
Note: If uploading the code fails, you can press the Boot button on ESP32 after click media/dc77bfcf5851c8f43aab6cbe7cec7920.png	1563f04a42d451e7bc1a5
Note: If uploading the code fails, you can press the Boot button on ESP32 after click media/dc77bfcf5851c8f43aab6cbe7cec7920.png	1563f04a42d451e7bc1a5
Note: If uploading the code fails, you can press the Boot button on ESP32 after click media/dc77bfcf5851c8f43aab6cbe7cec7920.png the Boot button after the percentage of uploading progress appears, as shown below:	1563f04a42d451e7bc1a5:

Project_05_Traffic_Lights   Arduino 1.8.19	~ ^ X
<u>File Edit Sketch T</u> ools <u>H</u> elp	
	<mark>,⊘</mark> ,
Project_05_Traffic_Lights	
<pre>//* * Filename : Traffic Lights * Description : Simulated traffic lights. * Auther : http//www.keyestudio.com */ #define PIN_LED_RED 0 //define the red led pin #define PIN_LED_YELLOW 2 //define the yellow led pin #define PIN_LED_GREEN 15 //define the green led pin Void setup() {     pinMode(PIN_LED_RED, OUTPUT);     pinMode(PIN_LED_YELLOW, OUTPUT);     pinMode(PIN_LED_GREEN, OUTPUT); } </pre>	
<pre>void loop() {</pre>	,*
Done uploading.	
Leaving Hard resetting via RTS pin	Ĵ
▲ SP32 Wrover Module, Default 4MB with spiffs (1.2MB APP/1.5MB SPIFFS), QIO, 80MHz, 921600, None on /dev	/ttyUSB0

# 9.6.5 Project result

After the project code was uploaded successfully, power up with a USB cable and you'll see are below:

First, the green light will be on for five seconds and then off;

Next, the yellow light blinks three times and then goes off;

Then, the red light goes on for five seconds and then goes off;

Repeat steps 1 to 3 above.

# 9.7 Project 06: RGB LED

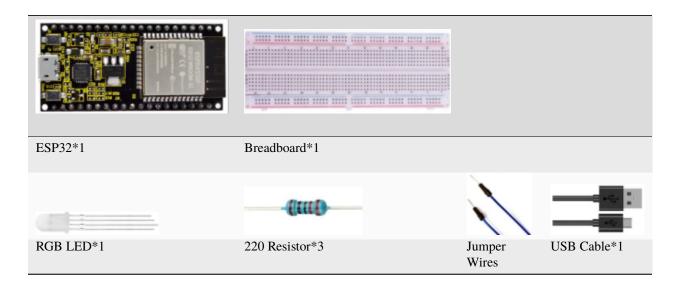
# 9.7.1 Introduction

media/94bdff69e438989d8e0934e57f2e5c00.png

RGB is composed of three colors (red, green and blue), which can emit different colors of light by mixing these three basic colors.

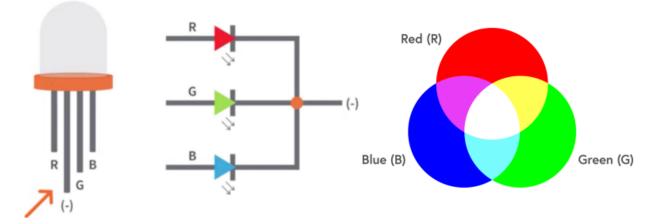
In this project, we will introduce the RGB and show you how to use ESP32 to control the RGB to emit different color light .RGB is pretty basic, but it's also a great way to learn the fundamentals of electronics and coding.

# 9.7.2 Components



# 9.7.3 Component knowledge

Most monitors adopt the RGB color standard, and all colors on a computer screen are a mixture of red, green and blue in varying proportions.



This RGB LED has 4 pins, each color (red, green, blue) and a common cathode, To change its brightness, we can use the PWM of the ESP32 pins, which can give different duty cycle signals to the RGB to produce different colors of light.

If we use three 10-bit PWM to control the RGB, in theory, we can create  $2 \ 10^{210} = 1,073,741,824(1 \text{ billion})$  colors through different combinations.

# 9.7.4 Wiring diagram

media/f3deb3502985ac8d66e99e4f27b3de1e.png

Note:

How to connect a LED

media/42ff6f405dfa128593827de5aa03e94b.png

How to identify the 220 Five-color ring resistor

media/55c0199544e9819328f6d5778f10d7d0.png

# 9.7.5 Project code

You can open the code we provide: If you haven't downloaded the code file, please click on the link to download itDownload Arduino C Codes file

The code used in this project is saved in folder "Arduino-Codes\Project 06RGB LED\Project\_06\_RGB\_LED".

```
/*
* Filename : RGB LED
* Description : Use RGBLED to show random color.
* Auther : http//www.keyestudio.com
*/
int ledPins[] = {0, 2, 15}; //define red, green, blue led pins
const byte chns[] = \{0, 1, 2\};
                           //define the pwm channels
int red, green, blue;
void setup() {
 for (int i = 0; i < 3; i++) { //setup the pwm channels,1KHz,8bit</pre>
   ledcSetup(chns[i], 1000, 8);
   ledcAttachPin(ledPins[i], chns[i]);
 }
}
void loop() {
 red = random(0, 256);
 green = random(0, 256);
 blue = random(\emptyset, 256);
 setColor(red, green, blue);
 delay(200);
}
void setColor(byte r, byte g, byte b) {
 ledcWrite(chns[0], 255 - r); //Common anode LED, low level to turn on the led.
 ledcWrite(chns[1], 255 - g);
 ledcWrite(chns[2], 255 - b);
}
```

# 9.7.6 Project result

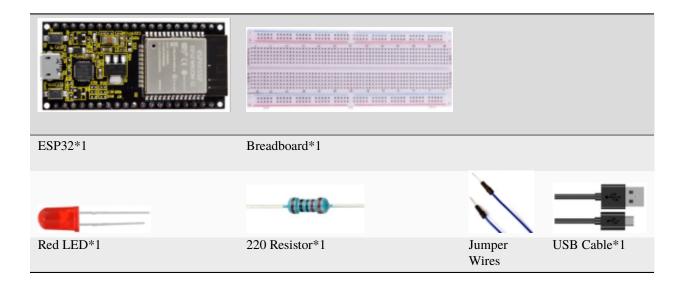
Compile and upload the code to ESP32, after the code is uploaded successfully, power up with a USB cable and you will see that the RGB LED starts to display random colors.

# 9.8 Project 07: Flowing Water Light

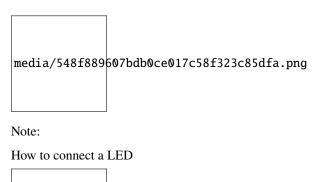
# 9.8.1 Introduction

In our daily life, we can see many billboards composed of different colors of LED. They constantly change the light (like water) to attract customers' attention. In this project, we will use ESP32 to control 10 leds to achieve the effect of flowing water.

# 9.8.2 Components



# 9.8.3 Wiring diagram:



media/42ff6f405dfa128593827de5aa03e94b.png

How to identify the 220 Five-color ring resistor

media/55c0199544e9819328f6d5778f10d7d0.png

# 9.8.4 Project code

This project is designed to make a flowing water lamp. Which are these actions: First turn LED #1 ON, then turn it OFF. Then turn LED #2 ON, and then turn it OFF... and repeat the same to all 10 LEDs until the last LED is turns OFF. This process is repeated to achieve the "movements" of flowing water.

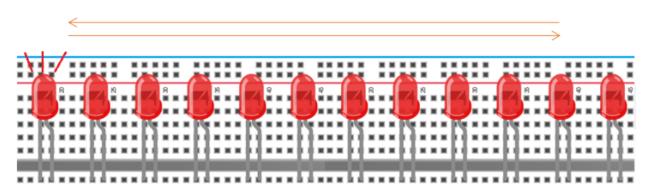
You can open the code we provide: If you haven't downloaded the code file, please click on the link to download itDownload Arduino C Codes file

The code used in this project is saved in folder "Arduino-Codes\Project 07Flowing Water Light\Project\_07\_Flowing\_Water\_Light"

```
/*
* Filename : Flowing Water Light
* Description : Using ten leds to demonstrate flowing lamp.
* Auther : http//www.keyestudio.com
*/
byte ledPins[] = {22, 21, 19, 18, 17, 16, 4, 0, 2, 15};
int ledCounts;
void setup() {
 ledCounts = sizeof(ledPins);
 for (int i = 0; i < ledCounts; i++) {
   pinMode(ledPins[i], OUTPUT);
 }
}
void loop() {
 for (int i = 0; i < ledCounts; i++) {
   digitalWrite(ledPins[i], HIGH);
   delay(100);
   digitalWrite(ledPins[i], LOW);
 }
 for (int i = ledCounts - 1; i > -1; i--) {
   digitalWrite(ledPins[i], HIGH);
   delay(100);
   digitalWrite(ledPins[i], LOW);
 }
}
```

## 9.8.5 Project result

Compile and upload the code to ESP32, after the code is uploaded successfully, power up with a USB cable and you will see that 10 LEDs will light up from left to right and then back from right to left.

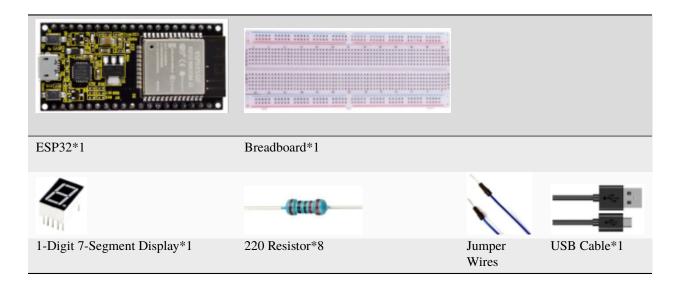


# 9.9 Project 081-Digit Digital Tube

#### 9.9.1 Introduction

A 1-Digit 7-Segment Display is an electronic display device that displays decimal numbers. It is widely used in digital clocks, electronic meters, basic calculators and other electronic devices that display digital information. Eventhough they may not look modern enough, they are an alternative to more complex dot matrix displays and are easy to use in limited light conditions and strong sunlight. In this project, we will use ESP32 to control 1-Digit 7-segment display displays numbers.

### 9.9.2 Components



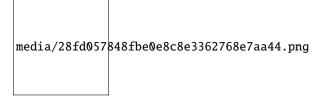
## 9.9.3 Component knowledge

media/e44a0f27beec739ee13e68c04865989f.png

**1-Digit 7-Segment Display principle:** Digital tube display is a semiconductor light emitting device, its basic unit is a light-emitting diode (LED). The digital tube display can be divided into 7-segment display and 8-segment display according to the number of segments. The 8-segment display has one more LED unit than the 7-segment display (used for decimal point display). Each segment of the 7-segment display is a separate LED. According to the connection mode of the LED unit, the digital tube can be divided into a common anode digital tube and a common cathode digital tube.

In the common cathode 7-segment display, all the cathodes (or negative electrodes) of the segmented LEDs are connected together, so you should connect the common cathode to GND. To light up a segmented LED, you can set its associated pin to "HIGH".

In the common anode 7-segment display, the LED anodes (positive electrodes) of all segments are connected together, so you should connect the common anode to "+5V". To light up a segmented LED, you can set its associated pin to "LOW".



Each part of the digital tube is composed of an LED. So when you use it, you also need to use a current limiting resistor. Otherwise, the LED will be damaged. In this experiment, we use an ordinary common cathode one-digit digital tube. As we mentioned above, you should connect the common cathode to GND. To light up a segmented LED, you can set its associated pin to "HIGH".

## 9.9.4 Wiring diagram

Note: The direction of the 7-segment display inserted into the breadboard is consistent with the wiring diagram, with one more point in the lower right corner.

media/631ee0861da60ed02d191de0e0e210d9.png

media/5f01d1eea2bb207f19dee4f437f93bc8.png

#### 9.9.5 Project code

The digital display is divided into 7 segments, and the decimal point display is divided into 1 segment. When certain numbers are displayed, the corresponding segment will be lit. For example, when the number 1 is displayed, segments b and c will be turned on.

You can open the code we provide: If you haven't downloaded the code file, please click on the link to download itDownload Arduino C Codes file

The code used in this project is saved in folder "Arduino-Codes\Project 081-Digit Digital Tube\Project\_08\_One\_Digit\_Digital\_Tube.

```
/*
* Filename : 1-Digit Digital Tube
* Description : One Digit Tube displays numbers from 9 to 0.
* Auther : http//www.keyestudio.com
*/
// sets the IO PIN for every segment
int a=16; // digital PIN 16 for segment a
int b=4; // digital PIN 4 for segment b
int c=5; // digital PIN 5 for segment c
int d=18; // digital PIN 18 for segment d
int e=19; // digital PIN 19 for segment e
int f=22; // digital PIN 22 for segment f
int g=23; // digital PIN 23 for segment g
int dp=17; // digital PIN 17 for segment dp
void digital_0(void) // displays number 0
{
digitalWrite(a,HIGH);
digitalWrite(b,HIGH);
digitalWrite(c,HIGH);
digitalWrite(d,HIGH);
digitalWrite(e,HIGH);
digitalWrite(f,HIGH);
digitalWrite(g,LOW);
digitalWrite(dp,LOW);
}
void digital_1(void) // displays number 1
{
digitalWrite(a,LOW);
digitalWrite(b,HIGH);
digitalWrite(c,HIGH);
digitalWrite(d,LOW);
digitalWrite(e,LOW);
digitalWrite(f,LOW);
digitalWrite(g,LOW);
digitalWrite(dp,LOW);
}
void digital_2(void) // displays number 2
{
digitalWrite(a,HIGH);
digitalWrite(b,HIGH);
digitalWrite(c,LOW);
                                                                       (continues on next page)
```

```
digitalWrite(d,HIGH);
digitalWrite(e,HIGH);
digitalWrite(f,LOW);
digitalWrite(g,HIGH);
digitalWrite(dp,LOW);
}
void digital_3(void) // displays number 3
{
digitalWrite(a,HIGH);
digitalWrite(b,HIGH);
digitalWrite(c,HIGH);
digitalWrite(d,HIGH);
digitalWrite(f,LOW);
digitalWrite(e,LOW);
digitalWrite(dp,LOW);
digitalWrite(g,HIGH);
}
void digital_4(void) // displays number 4
{
digitalWrite(a,LOW);
digitalWrite(b,HIGH);
digitalWrite(c,HIGH);
digitalWrite(d,LOW);
digitalWrite(e,LOW);
digitalWrite(f,HIGH);
digitalWrite(g,HIGH);
digitalWrite(dp,LOW);
}
void digital_5(void) // displays number 5
{
digitalWrite(a,HIGH);
digitalWrite(b,LOW);
digitalWrite(c,HIGH);
digitalWrite(d,HIGH);
digitalWrite(e,LOW);
digitalWrite(f,HIGH);
digitalWrite(g,HIGH);
digitalWrite(dp,LOW);
}
void digital_6(void) // displays number 6
{
digitalWrite(a,HIGH);
digitalWrite(b,LOW);
digitalWrite(c,HIGH);
digitalWrite(d,HIGH);
digitalWrite(e,HIGH);
digitalWrite(f,HIGH);
digitalWrite(g,HIGH);
digitalWrite(dp,LOW);
}
void digital_7(void) // displays number 7
{
```

(continues on next page)

```
digitalWrite(a,HIGH);
digitalWrite(b,HIGH);
digitalWrite(c,HIGH);
digitalWrite(d,LOW);
digitalWrite(e,LOW);
digitalWrite(f,LOW);
digitalWrite(g,LOW);
digitalWrite(dp,LOW);
}
void digital_8(void) // displays number 8
{
digitalWrite(a,HIGH);
digitalWrite(b,HIGH);
digitalWrite(c,HIGH);
digitalWrite(d,HIGH);
digitalWrite(e,HIGH);
digitalWrite(f,HIGH);
digitalWrite(g,HIGH);
digitalWrite(dp,LOW);
}
void digital_9(void) // displays number 9
{
digitalWrite(a,HIGH);
digitalWrite(b,HIGH);
digitalWrite(c,HIGH);
digitalWrite(d,HIGH);
digitalWrite(e,LOW);
digitalWrite(f,HIGH);
digitalWrite(g,HIGH);
digitalWrite(dp,LOW);
}
void setup()
{
  // initialize digital pin LED as an output.
 pinMode(a, OUTPUT);
 pinMode(b, OUTPUT);
  pinMode(c, OUTPUT);
  pinMode(d, OUTPUT);
  pinMode(e, OUTPUT);
  pinMode(f, OUTPUT);
  pinMode(g, OUTPUT);
  pinMode(dp, OUTPUT);
}
void loop()
{
while(1)
{
digital_9();// displays number 9
delay(1000); // waits a sencond
digital_8();// displays number 8
delay(1000); // waits a sencond
digital_7();// displays number 7
```

(continues on next page)

<pre>delay(1000); // waits a sencond</pre>
<pre>digital_6();// displays number 6</pre>
<pre>delay(1000); // waits a sencond</pre>
<pre>digital_5();// displays number 5</pre>
<pre>delay(1000); // waits a sencond</pre>
<pre>digital_4();// displays number 4</pre>
<pre>delay(1000); // waits a sencond</pre>
<pre>digital_3();// displays number 3</pre>
<pre>delay(1000); // waits a sencond</pre>
<pre>digital_2();// displays number 2</pre>
<pre>delay(1000); // waits a sencond</pre>
<pre>digital_1();// displays number 1</pre>
<pre>delay(1000);// waits a sencond</pre>
<pre>digital_0();// displays number 0</pre>
<pre>delay(1000);// waits a sencond</pre>
}}
//*************************************

### 9.9.6 Project result

Compile and upload the code to ESP32, after the code is uploaded successfully, power up with a USB cable and you will see that the display will display numbers from 9 to 0.

## 9.10 Project 094-Digit Digital Tube

## 9.10.1 Introduction

A 4-digit 7-segment display is a very practical display device and it is used for devices such as electronic clocks, score counters and the number of people in the park. Because of the low price, easy to use, more and more projects will use 4 Digit 7-segment display. In this project, we use ESP32 control 4-digit 7-segment display to display four digits.

### 9.10.2 Components

ESP32*1	Breadboard*1	
4-Digital Tube		
4-digit 7-segment display Module*1	M-F Dupont Wires	USB Cable*1

## 9.10.3 Component knowledge

\*\*TM1650 4-digit 7-segment display\*\*It is a 12-pin 4-digit 7-segment display module with clock dots. The driver chip is TM1650 which only needs 2 signal lines to enable the microcontroller to control the 4-digit 7-segment display. The control interface level can be 5V or 3.3V.

Specifications of 4-bit 7-segment display module:

Working voltage: DC 3.3V-5V

Maximum current: 100MA

Maximum power: 0.5W

Schematic diagram of 4-digit 7-segment display module:

media/5f400887c90fc00098a3e77beca656ef.png

### 9.10.4 Wiring diagram

media/a7721c08ed3b73b21a997d43d2e3addd.png

## 9.10.5 Adding the TM1650 library

This code uses a library named "TM1650", if you haven't installed it yet, please do so before learning. The steps to add third-party libraries are as follows:

## 9.10.6 Project code

After the **TM1650** library is added, You can open the code we provide If you haven't downloaded the code file, please click on the link to download itDownload Arduino C Codes file

The code used in this project is saved in folder "Arduino-Codes\Project 094-Digit Digital Tube\Project\_09\_Four\_Digit\_Digital\_Tube".

```
/*
* Filename : 4-Digit Digital Tube
* Description : Four Digit Tube displays numbers from 1111 to 9999.
* Auther
          : http//www.keyestudio.com
*/
#include "TM1650.h"
               //pins definitions for TM1650 and can be changed to other ports
#define CLK 22
#define DIO 21
TM1650 DigitalTube(CLK,DIO);
void setup(){
 //DigitalTube.setBrightness(); //stes brightness from 0 to 7(default is 2)
 //DigitalTube.displayOnOFF(); // 0= off,1= on(default is 1)
 for(char b=1;b<5;b++){
   DigitalTube.clearBit(b); //which bit to clear
 DigitalTube.displayDot(1,true); // displays the first number
 DigitalTube.displayDot(2,true);
 DigitalTube.displayDot(3,true);
 DigitalTube.displayDot(4,true);
 DigitalTube.displayBit(3,0); //which number to display. bit=1-4, number=0-9
}
void loop(){
 for(int num=0; num<10; num++){</pre>
   DigitalTube.displayBit(1,num);
   DigitalTube.displayBit(2,num);
   DigitalTube.displayBit(3,num);
   DigitalTube.displayBit(4,num);
   delay(1000);
 }
}
```

## 9.10.7 Project result

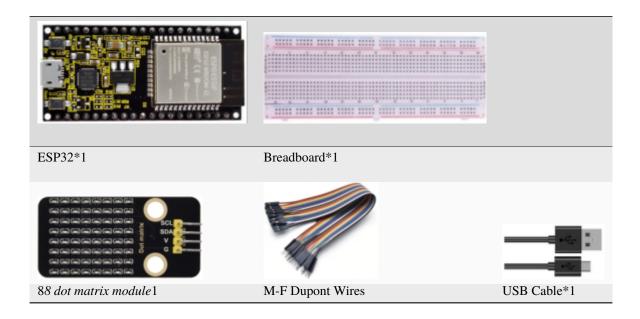
Compile and upload the code to ESP32, after the code is uploaded successfully, power up with a USB cable and you will see that 4-digit 7-segment display four digits four digits and repeat these actions in an infinite loop.

## 9.11 Project 108×8 Dot-matrix Display

### 9.11.1 Introduction

Dot matrix display is an electronic digital display device that can display information on machine, clocks, public transport departure indicators and many other devices. In this project, we will use ESP32 control 8x8 LED dot matrix to display patterns.

### 9.11.2 Components



## 9.11.3 Component knowledge

#### 8\*8 dot matrix module

The 8\*8 dot matrix is composed of 64 LEDs, and each LED is placed at the intersection of a row and a column. When using the single chip microcomputer to drive an 8\*8 dot matrix, we need 16 digital ports in total, which greatly wastes the data of the single chip microcomputer. To this end, we specially designed this module, using the HT16K33 chip to drive an 8\*8 dot matrix, and only need to use the I2C communication port of the MCU to control the 8\*8 dot matrix, which greatly saving the MCU resources.

Specifications of 8\*8 dot matrix module

- Working voltage: DC 5V
- Current: 200MA
- Maximum power: 1W

• Schematic diagram of 8\*8 dot matrix module

media/b04fe5e60695365a23644395aaef5085.png

Some modules have three DIP switches that you can toggle at will. These switches are used to set the I2C communication address, the setting method is as follows. The module has fixed the communication address. A0, A1 and A2 are connected to GND, and the address is 0x70.

A0 (1)	A1 (2)	A2 (3)	A0 (1)	A1 (2)	A2 (3)	A0 (1)	A1 (2)	A2 (3)
0 (OFF)	0(OFF)	0 (OFF)	1 (ON)	0 (OFF)	0 (OFF)	0 (OFF)	1 (ON)	0 (OFF)
0X70	0X70 0X71		0X71		0X72			
A0 (1)	A1 (2)	A2 (3)	A0 (1)	A1 (2)	A2 (3)	A0 (1)	A1 (2)	A2 (3)
1 (ON)	1 (ON)	0 (OFF)	0(OFF)	0 (OFF)	1 (ON)	1 (ON)	0 (OFF)	1 (ON)
<b>0X7</b> 3			0X74	4		0X75		
A0 (1)	A1 (2)	A2 (3)	A0 (1)	A1 (2)	A2 (3)			
0 (OFF)	1 (ON)	1 (ON)	1 (ON)	1 (ON)	1 (ON)			
0X76	•	•	0X77					

## 9.11.4 Wiring diagram

media/78a74a4a920791b492bcd398dc8dc82b.png

## 9.11.5 Adding the HT16K33\_Lib\_For\_ESP32 library

This code uses a library named "HT16K33\_Lib\_For\_ESP32", if you haven't installed it yet, please do so before learning. The steps to add third-party libraries are as follows:

#### 9.11.6 Project code

After the **HT16K33\_Lib\_For\_ESP32** library is added, You can open the code we provide f you haven't downloaded the code file, please click on the link to download itDownload Arduino C Codes file

The code used in this project is saved in folder "Arduino-Codes\Project 108×8 Dot-matrix Display\Project\_10\_8×8\_Dot\_Matrix\_Display".

```
/*
* Filename : 8×8 Dot-matrix Display
* Description : 8x8 LED dot matrix display"Heart" pattern.
* Auther : http//www.keyestudio.com
*/
#include "HT16K33_Lib_For_ESP32.h"
#define SDA 21
#define SCL 22
ESP32_HT16K33 matrix = ESP32_HT16K33();
//The brightness values can be set from 1 to 15, with 1 darkest and 15 brightest
#define A 15
byte result[8][8];
byte test1[8] = {0x00,0x42,0x41,0x09,0x09,0x41,0x42,0x00};
void setup()
{
 matrix.init(0x70, SDA, SCL);//Initialize matrix
 matrix.showLedMatrix(test1,0,0);
 matrix.show();
}
void loop()
{
 for (int i = 0; i <= 7; i++)
 {
   matrix.setBrightness(i);
   delay(100);
 }
 for (int i = 7; i > 0; i - -)
 {
   matrix.setBrightness(i);
   delay(100);
 }
}
```

## 9.11.7 Project result

Compile and upload the code to ESP32, after the code is uploaded successfully, power up with a USB cable and you will see that the 8\*8 dot matrix display "Smiling face" pattern.

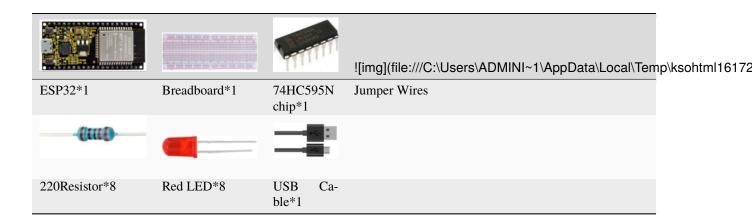
## 9.12 Project 1174HC595N Control 8 LEDs

### 9.12.1 Introduction

In previous projects, we learned how to light up an LED.

With only 32 IO ports on ESP32, how do we light up a lot of leds? Sometimes it is possible to run out of pins on the ESP32, and you need to extend it with the shift register. You can use the 74HC595N chip to control 8 outputs at a time, taking up only a few pins on your microcontroller. In addition, you can also connect multiple registers together to further expand the output. In this project, we will use an ESP32, a 74HC595 chip and LEDs to make a flowing water light to understand the function of the 74HC595 chip.

## 9.12.2 Components



### 9.12.3 Component knowledge

media/6921c6d60135e072ed4bd24564ec4a6d.png

**74HC595N Chip:** The 74HC595 chip is used to convert serial data into parallel data. A 74HC595 chip can convert the serial data of one byte into 8 bits, and send its corresponding level to each of the 8 ports correspondingly. With this characteristic, the 74HC595 chip can be used to expand the IO ports of an ESP32. At least 3 ports are required to control the 8 ports of the 74HC595 chip.

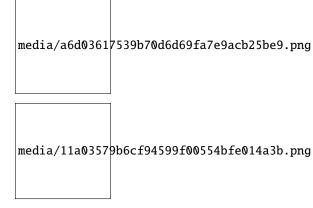
#### media/858b189f06ad68afe051b15043b2affd.png

The ports of the 74HC595 chip are described as follows

PIN	FUNCTION
Pin 13–OE	Enable output, When this pin is in high level, Q0-Q7 is in high resistance state. When this pin is in low level, Q0-Q7 is in output mode.
Pin 14—SI	Serial data Input, only enter one bit at a time, so you can enter eight consecutive times to form one byte.
Pin	Remove shift register: When this pin is in low level, the content in shift register will be cleared
10—SCLR	In this experiment, we connect VCC to maintain a high level.
Pin 11—SCK	Serial shift clock: when its electrical level is rising, serial data input register will do a shift.
Pin 12—RCK	Parallel Update Output: when its electrical level is rising, it will update the parallel data output. In this case, the data is output from ports Q0 to Q7 in parallel
Pin 9—SQH	Serial data output: it can be connected to more 74HC595 in series.
Q0–Q7(Pin 15Pin 1-7)	Parallel data output, can directly control the 8 segments of the digital tube.

### 9.12.4 Wiring diagram

Note: Note the orientation in which the 74HC595N chip is inserted.



## 9.12.5 Project code

You can open the code we provideIf you haven't downloaded the code file, please click on the link to download itDownload Arduino C Codes file

The code used in this project is saved in folder "Arduino-Codes\Project 1174HC595N Control 8 LEDs\Project\_11\_74HC595N\_Control\_8\_LEDs.

(continues on next page)

```
(continued from previous page)
```

```
* Description : Use 74HC575N to drive ten leds to display the flowing light.
* Auther
             : http//www.keyestudio.com
*/
int dataPin = 14; // Pin connected to DS of 74HC595(Pin14)
int latchPin = 12; // Pin connected to ST_CP of 74HC595(Pin12)
int clockPin = 13; // Pin connected to SH_CP of 74HC595(Pin11)
void setup() {
     // set pins to output
     pinMode(latchPin, OUTPUT);
     pinMode(clockPin, OUTPUT);
     pinMode(dataPin, OUTPUT);
}
void loop() {
 // Define a one-byte variable to use the 8 bits to represent the state of 8 LEDs of _{\rm L}
\leftrightarrow LED bar graph.
// This variable is assigned to 0x01, that is binary 00000001, which indicates only.
\rightarrow one LED light on.
 byte x = 0x01;
                 // Ob 0000 0001
 for (int j = 0; j < 8; j++) { // Let led light up from right to left
   writeTo595(LSBFIRST, x);
   \mathbf{x} \ll 1; // make the variable move one bit to left once, then the bright LED move
\rightarrow one step to the left once.
   delay(50);
 }
 delay(100);
 x = 0x80;
                //0b 1000 0000
 for (int j = 0; j < 8; j++) { // Let led light up from left to right
   writeTo595(LSBFIRST, x);
   x >>= 1;
   delay(50);
 }
 delay(100);
}
void writeTo595(int order, byte _data ) {
 // Output low level to latchPin
 digitalWrite(latchPin, LOW);
 // Send serial data to 74HC595
 shiftOut(dataPin, clockPin, order, _data);
 // Output high level to latchPin, and 74HC595 will update the data to the parallel.
\rightarrowoutput port.
 digitalWrite(latchPin, HIGH);
}
```

## 9.12.6 Project result

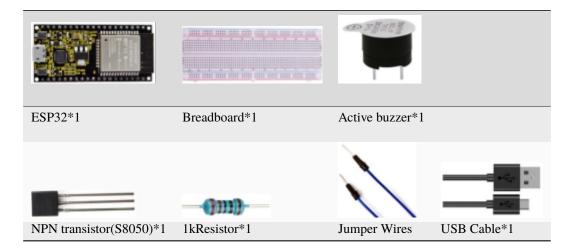
Compile and upload the code to ESP32, after the code is uploaded successfully, power up with a USB cable and you will see that the 8 LEDs start flashing in flowing water mode.

## 9.13 Project 12Active Buzzer

### 9.13.1 Introduction

Active buzzer is a sound component that is widely used as a sound component for computers, printers, alarms, electronic toys, phones and timers. It has an internal vibration source, just by connecting to a 5V power supply, it can continuously buzz. In this project, we will use ESP32 to control the active buzzer to beep.

## 9.13.2 Components



### 9.13.3 Component knowledge



#### Active buzzer:

Active buzzer inside has a simple oscillator circuit, which can convert constant direct current into a certain frequency pulse signal. Once active buzzer receives a high level, it will produce sound. Passive buzzer is an internal without vibration source integrated electronic buzzer, it must be driven by 2k to 5k square wave, rather than a DC signal. The two buzzers are very similar in appearance, but one buzzer with a green circuit board is a passive buzzer, while the other buzzer with black tape is an active buzzer. Passive buzzers don't have positive polarity, but active buzzers have. As shown below:

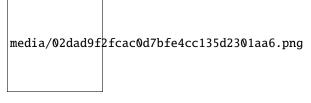
media/0f9825969867ac2d65bb1a19ed0ad2ab.png

Transistor:



Because the buzzer requires such large current that GPIO of ESP32 output capability cannot meet the requirement, a transistor of NPN type is needed here to amplify the current.

Transistor, the full name: semiconductor transistor, is a semiconductor device that controls current. Transistorcan be used to amplify weak signal, or works as a switch. It has three electrodes(PINs): base (b), collector © and emitter (e). When there is current passing between "be", "ce" will allow several-fold current (transistor magnification) pass, at this point, transistor works in the amplifying area. When current between "be" exceeds a certain value, "ce" will not allow current to increase any longer, at this point, transistor works in the saturation area. Transistor has two types as shown below: PNP and NPN, here are the PNP transistor and NPN transistor.



In our kit, the PNP transistor is marked with 8550, and the NPN transistor is marked with 8050.

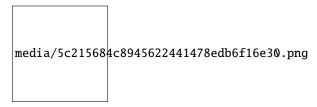
Based on the transistor's characteristics, it is often used as a switch in digital circuits. As micro-controller's capacity to output current is very weak, we will use transistor to amplify current and drive large-current components.

When using NPN transistor to drive buzzer, we often adopt the following method. If GPIO outputs high level, current will flow through R1, the transistor will get conducted, and the buzzer will sound. If GPIO outputs low level, no current flows through R1, the transistor will not be conducted, and buzzer will not sound.

When using PNP transistor to drive buzzer, we often adopt the following method. If GPIO outputs low level, current will flow through R1, the transistor will get conducted, and the buzzer will sound. If GPIO outputs high level, no current flows through R1, the transistor will not be conducted, and buzzer will not sound.



### 9.13.4 Wiring diagram



Note: The buzzer power supply in this circuit is 5V. On a 3.3V power supply, the buzzer can work, but will reduce the loudness.

## 9.13.5 Project code

You can open the code we provide: If you haven't downloaded the code file, please click on the link to download itDownload Arduino C Codes file

The code used in this project is saved in folder "Arduino-Codes\Project 12Active Buzzer\Project\_12\_Active\_Buzzer".

```
/*
* Filename
        : Active Buzzer
* Description : Active buzzer beeps.
* Auther : http//www.keyestudio.com
*/
#define buzzerPin 15 //define buzzer pins
void setup ()
{
 pinMode (buzzerPin, OUTPUT);
}
void loop ()
{
 digitalWrite (buzzerPin, HIGH);
 delay (500);
 digitalWrite (buzzerPin, LOW);
 delay (500);
}
```

### 9.13.6 Project result

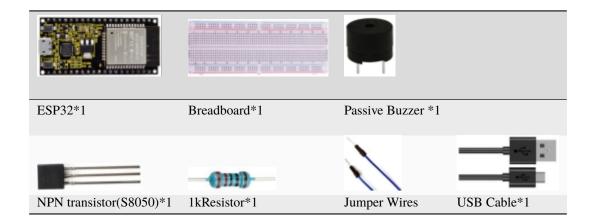
Compile and upload the code to ESP32, after the code is uploaded successfully, power up with a USB cable and you will see that the active buzzer beeps.

# 9.14 Project 13Passive Buzzer

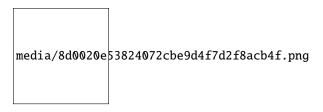
## 9.14.1 Introduction:

In a previous project, we studied an active buzzer, which can only make a sound and may make you feel very monotonous. In this project, we will learn a passive buzzer and use the ESP32 control it to work. Unlike the active buzzer, the passive buzzer can emit sounds of different frequencies.

## 9.14.2 Components



### 9.14.3 Component knowledge



**Passive buzzer:** A passive buzzer is an integrated electronic buzzer with no internal vibration source and it has to be driven by 2K-5K square waves, not DC signals. The two buzzers are very similar in appearance, but one buzzer with a green circuit board is a passive buzzer and the other buzzer with black tape is an active buzzer. Passive buzzers cannot distinguish between positive polarity while active buzzers can.

media/fc42c5ed014609ff0b290ee5361bb2fd.png

Transistor: Please refer to Project 12.

#### 9.14.4 Wiring diagram:



## 9.14.5 Project code

You can open the code we provide: If you haven't downloaded the code file, please click on the link to download itDownload Arduino C Codes file

The code used in this project is saved in folder "Arduino-Codes\Project 13Passive Buzzer\Project\_13\_Passive\_Buzzer.

```
/*
* Filename
            : Passive Buzzer
* Description : Passive Buzzer sounds the alarm.
            : http//www.keyestudio.com
* Auther
*/
#define LEDC_CHANNEL_0 0
// LEDC timer uses 13 bit accuracy
#define LEDC_TIMER_13_BIT 13
// Define tool I/O ports
#define BUZZER PIN 15
//Create a musical melody list, Super Mario
int melody[] = {330, 330, 330, 262, 330, 392, 196, 262, 196, 165, 220, 247, 233, 220,
→196, 330, 392, 440, 349, 392, 330, 262, 294, 247, 262, 196, 165, 220, 247, 233, 220, 1
→196, 330, 392,440, 349, 392, 330, 262, 294, 247, 392, 370, 330, 311, 330, 208, 220, .
\rightarrow 262, 220, 262,
294, 392, 370, 330, 311, 330, 523, 523, 523, 392, 370, 330, 311, 330, 208, 220, 262,220,
→262, 294, 311, 294, 262, 262, 262, 262, 262, 294, 330, 262, 220, 196, 262, 262, 262, 2
→262, 294, 330, 262, 262, 262, 262, 294, 330, 262, 220, 196};
//Create a list of tone durations
\rightarrow 8,4,8,4,3,8,8,2,8,8,8,4,4,8,8,4,8,8,3,8,8,8,4,4,4,8,2,8,8,8,4,4,8,8,4,8,8,3,3,3,1,8,4,
\leftrightarrow4,8,4,8,4,8,2,8,4,4,8,4,1,8,4,4,8,4,8,4,8,2};
void setup() {
pinMode(BUZZER_PIN, OUTPUT); // Set the buzzer to output mode
}
```

(continues on next page)

```
void loop() {
    int noteDuration; //Create a variable of noteDuration
    for (int i = 0; i < sizeof(noteDurations); ++i)
    {
        noteDuration = 800/noteDurations[i];
        ledcSetup(LEDC_CHANNEL_0, melody[i]*2, LEDC_TIMER_13_BIT);
        ledcAttachPin(BUZZER_PIN, LEDC_CHANNEL_0);
        ledcWrite(LEDC_CHANNEL_0, 50);
        delay(noteDuration * 1.30); //delay
    }
}</pre>
```

## 9.14.6 Project result

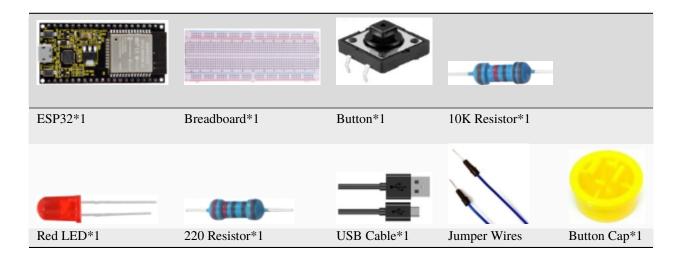
Compile and upload the code to ESP32, after the code is uploaded successfully, power up with a USB cable and you will see that the passive buzzer plays music.

## 9.15 Project 14: Mini Table Lamp

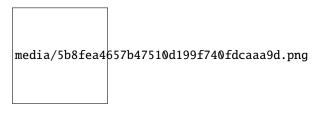
### 9.15.1 Introduction

Do you know that the ESP32 can light up an LED when you press a button? In this project, we will use ESP32a button switch and an LED to make a mini table lamp.

## 9.15.2 Components



## 9.15.3 Component knowledge



**Button:** A button can control the circuit on and off, the button is plugged into a circuit, the circuit is disconnected when the button is not pressed. The circuit works when you press the button, but breaks again when you release it. Why does it only work when you press it? It starts from the internal structure of the button, which don't allow current to travel from one end of the button to the other before it is pressed; When pressed, a metal strip inside the button connects the two sides to allow electricity to pass through.

media/d2a204e61c768f18924150db58aee093.png

The internal structure of the button is shown in the figure. Before the button is pressed, 1 and 2 are on, 3 and 4 are also on, but 1, 3 or 1, 4 or 2, 3 or 2, 4 are off (not working). Only when the button is pressed, 1, 3 or 1, 4 or 2, 3 or 2, 4 are on.

The button switch is one of the most commonly used components in circuit design.

Schematic diagram of the button:



What is button jitteritch circuit as "press the button and turn it on immediately", "press it again and turn it off immediately". In fact,

We think of the switch circuit as "press the button and turn it on immediately", "press it again and turn it off immediately". In fact, this is not the case.

The button usually uses a mechanical elastic switch, and the mechanical elastic switch will produce a series of [shake](javascript:;) due to the elastic action at the moment when the mechanical contact is opened and closed (usually about 10ms). As a result, the button switch will not immediately and stably turn on the circuit when it is closed, and it will not be completely and instantaneously disconnected when it is turned off.

media/7e7ac82db8bb810a7ee1de4181ceaa2d.jpeg

How to eliminate the jitter?

There are two common methods, namely fix jitter in the software and hardware. We only discuss the jitter removal in the software.

We already know that the jitter time generated by elasticity is about 10ms, and the delay command can be used to delay the execution time of the command to achieve the effect of jitter removal.

Therefore, we delay 0.02s in the code to achieve the key anti-shake function.



#### 9.15.4 Wiring Diagram

media/a5b85f1e1f5714afbe4730b1265e3a15.png

Note:

How to connect the LED

media/f70404aa49540fd7aecae944c7c01f83.jpeg

How to identify the 220 5-band resistor and 10K 5-band resistor

media/55c0199544e9819328f6d5778f10d7d0.png media/246cf3885dc837c458a28123885c9f7b.png

#### 9.15.5 Project code

You can open the code we provide: If you haven't downloaded the code file, please click on the link to download itDownload Arduino C Codes file

The code used in this project is saved in folder "Arduino-Codes\Project 14Mini Table Lamp\Project\_14\_Mini\_Table\_Lamp".

```
/*
* Filename : Mini Table Lamp
* Description : Make a table lamp.
* Auther : http//www.keyestudio.com
*/
#define PIN_LED
              4
#define PIN BUTTON 15
bool ledState = false;
void setup() {
 // initialize digital pin PIN_LED as an output.
 pinMode(PIN_LED, OUTPUT);
 pinMode(PIN_BUTTON, INPUT);
}
// the loop function runs over and over again forever
void loop() {
 if (digitalRead(PIN_BUTTON) == LOW) {
   delay(20);
   if (digitalRead(PIN_BUTTON) == LOW) {
    reverseGPIO(PIN_LED);
   }
   while (digitalRead(PIN_BUTTON) == LOW);
 }
}
void reverseGPIO(int pin) {
 ledState = !ledState;
 digitalWrite(pin, ledState);
}
```

## 9.15.6 Project result

Compile and upload the code to ESP32, after the code is uploaded successfully, power up with a USB cable and you will see that press the push button switch, the LED turns on; When it is released, the LED is still on. Press it again, and the LED turns off. When it is released, the LED stays off. Doesn't it look like a mini table lamp?

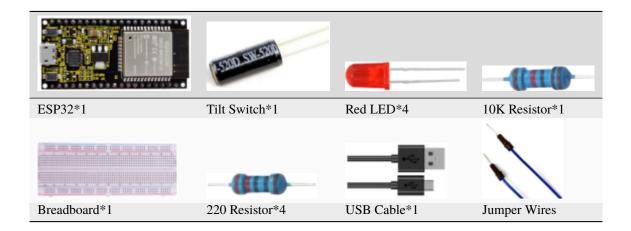
## 9.16 Project 15Tilt and LED

## 9.16.1 Introduction

The ancients without electronic clock, so the hourglass are invented to measure time. The hourglass has a large capacity on both sides, and which is filled with fine sand on one side. What's more, there is a small channel in the middle, which can make the hourglass stand upright, the side with fine sand is on the top. due to the effect of gravity, the fine sand will flow down through the channel to the other side of the hourglass.

When the sand reaches the bottom, turn it upside down and record the number of times it has gone through the hourglass, therefore, the next day we can know the approximate time of the day by it. In this project, we will use ESP32 to control the tilt switch and LED lights to simulate an hourglass and make an electronic hourglass.

## 9.16.2 Components



### 9.16.3 Component knowledge



Tilt switch is also called digital switch. Inside is a metal ball that can roll. The principle of rolling the metal ball to contact with the conductive plate at the bottom, which is used to control the on and off of the circuit. When it is a rolling

ball tilt sensing switch with single directional trigger, the tilt sensor is tilted toward the trigger end (two gold-plated pin ends), the tilt switch is in a closed circuit and the voltage at the analog port is about 5V(binary number is 1023),

In this way, the LED will light up. When the tilting switch is in horizontal position or tilting to the other end, the tilting switch is in open state the voltage of the analog port is about 0V (binary number is 0), the LED will turn off. In the program, we judge the state of the switch based on whether the voltage value of the analog port is greater than 2.5V (binary number is 512).

The internal structure of the tilt switch is used here to illustrate how it works, as shown below:



4. Wiring Diagram

media/a46c0b8be898ba596308ce56993c26ba.png

Note:

How to connect the LED

media/f70404aa49540fd7aecae944c7c01f83.jpeg

How to identify the 220 5-band resistor and 10K 5-band resistor

media/55c0199544e9819328f6d5778f10d7d0.png

media/246cf3885dc837c458a28123885c9f7b.png

## 9.16.4 Project code

You can open the code we provide: If you haven't downloaded the code file, please click on the link to download itDownload Arduino C Codes file

The code used in this project is saved in folder "Arduino-Codes\Project 15Tilt And LED\Project\_15\_Tilt\_And\_LED.

```
/*
* Filename
          : Tilt And LED
* Description : Tilt switches and four leds to simulate an hourglass.
* Auther : http//www.keyestudio.com
*/
#define SWITCH_PIN 15 // the tilt switch is connected to Pin15
byte switch_state = \emptyset;
void setup()
{
    for(int i=16;i<20;i++)</pre>
 {
      pinMode(i, OUTPUT);
 }
   pinMode(SWITCH_PIN, INPUT);
for(int i=16;i<20;i++)</pre>
 {
   digitalWrite(i, ◊);
 }
 Serial.begin(9600);
}
void loop()
{
switch_state = digitalRead(SWITCH_PIN);
Serial.println(switch_state);
if (switch_state == 0)
{
for(int i=16;i<20;i++)</pre>
 {
   digitalWrite(i,1);
   delay(500);
 }
 }
  if (switch_state == 1)
{
  for(int i=19;i>15;i--)
  {
   digitalWrite(i,0);
   delay(500);
  }
 }
}
```

## 9.16.5 Project result

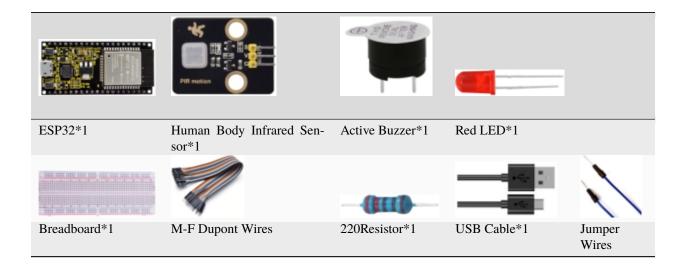
Compile and upload the code to ESP32, after the code is uploaded successfully, power up with a USB cable and you will see that when you tilt the breadboard to an angle, the LEDs will light up one by one. When you turn the breadboard to the original angle, the LEDs will turn off one by one. Like the hourglass, the sand will leak out over time.

# 9.17 Project 16Burglar Alarm

## 9.17.1 Introduction

The PIR motion sensor measures the thermal infrared (IR) light emitted by moving objects. The sensor can detect the movement of peopleanimals and carsto trigger safety alarms and lighting. They are used to detect movement and ideal for security such as burglar alarms and security lighting systems. In this project, we will use the ESP32 to control PIR motion sensorbuzzer and LED to simulate burglar alarm.

## 9.17.2 Components



## 9.17.3 Component knowledge



**PIR Motion Sensor :** Its principle is that when some crystals, such as lithium tantalate and triglyceride sulfate are heated, the two ends of the crystal will generate an equal number of charges with opposite signs. These charges can be converted into voltage output by an amplifier. Due to the human body will release infrared light, although relatively weak, can still be detected. When the Human Body Infrared Sensor detects the movement of a nearby person, the sensor signal terminal outputs a high level 1, otherwise, it outputs low level 0.

Special attention should be paid to the fact that this sensor can detect peopleanimals and cars in motion, which cannot be detected in static, and the maximum detection distance is about 7 meters.

**Note:** Since vulnerable to radio frequency radiation and temperature changes, the PIR motion sensor should be kept away from heat sources like radiators, heaters and air conditioners, as well as direct irradiation of sunlight, headlights and incandescent light.

Features:

- Maximum input voltage: DC 3.3 ~ 5V
- Maximum operating current: 50MA
- Maximum power: 0.3W
- Operating temperature: -20 ~ 85°C
- Output high level is 3V, low level is 0V.
- Delay time: about 2.3 to 3 seconds
- Detection Angle: about 100 degrees
- Maximum detection distance: about 7 meters
- Indicator light output (when the output is high, it will light up)
- Pin limiting current: 50MA

Schematic diagram:

media/9e1ec604aa6f9d4a3c1fe41d4bccd699.png

### 9.17.4 Wiring Diagram

media/67fd78fc542f0e7c232d96a23fb90120.png

#### 9.17.5 Project code

You can open the code we provide: If you haven't downloaded the code file, please click on the link to download itDownload Arduino C Codes file

The code used in this project is saved in folder "Arduino-Codes\Project 16Burglar Alarm\Project\_16\_Burglar\_Alarm".

(continues on next page)

```
* Description : Human infrared sensor buzzer and LED to simulate burglar alarm.
 * Auther
             : http//www.keyestudio.com
*/
#define buzzerPin 2 // the pin of the buzzer
                     // the pin of the PIR motion sensor
#define ledPin
               0
                      // the pin of the PIR motion sensor
#define pirPin
              15
byte pirStat = 0; // the state of the PIR motion sensor
void setup() {
pinMode(buzzerPin, OUTPUT);
pinMode(ledPin, OUTPUT);
pinMode(pirPin, INPUT);
}
void loop()
{
pirStat = digitalRead(pirPin);
if (pirStat == HIGH)
{
            // if people or moving animals are detected
  digitalWrite(buzzerPin, HIGH); // the buzzer buzzes
  digitalWrite(ledPin, HIGH); // the led turn on
  delay(500);
  digitalWrite(buzzerPin, LOW); // the buzzer doesn't sound
  digitalWrite(ledPin, LOW); // the led turn off
  delay(500);
}
else {
  digitalWrite(buzzerPin, LOW); // if people or moving animals are not detected, turn_
→off buzzers
  digitalWrite(ledPin, LOW); // the led turn off
}
}
```

### 9.17.6 Project result

Compile and upload the code to ESP32, after the code is uploaded successfully, power up with a USB cable and you will see that if the sensor detects someone moving nearby, the buzzer will continuously issue an alarm and the LED will continuously flash.

## 9.18 Project 17 I2C 128×32 LCD

#### 9.18.1 Introduction

In everyday life, we can do all kinds of experiments with the display module and also DIY a variety of small objects. For example, you can make a temperature meter with a temperature sensor and display, or make a distance meter with an ultrasonic module and display. In this project, we will use the LCD\_128X32\_DOT module as the display and connect it to the ESP32, which will be used to control the LCD\_128X32\_DOT display to display various English words, common symbols and numbers.

## 9.18.2 Components

ESP32*1	Breadboard*1	
SCL SDA g		
LCD_128X32_DOT*1	M-F Dupont Wires	USB Cable*1

## 9.18.3 Component knowledge



**LCD\_128X32\_DOT:** It is an LCD module with 128\*32 pixels and its driver chip is ST7567A. The module uses the IIC communication mode, while the code contains a library of all alphabets and common symbols that can be called directly. When using, we can also set it in the code so that the English letters and symbols show different text sizes. To make it easy to set up the pattern display, we also provide a mold capture software that converts a specific pattern into control code and then copies it directly into the test code for use.

Schematic diagram of LCD\_128X32\_DOT

media/5451aed32bc5b7b30fbd5613ad09a65b.png

Features:

Pixel: 128\*32 character

Operating voltage(chip)4.5V to 5.5V

Operating current100mA (5.0V)

Optimal operating voltage(module):5.0V

### 9.18.4 Wiring Diagram



## 9.18.5 Adding the lcd128\_32\_io library

If you have not downloaded the library file, please click on the link to download it:Download Arduino Libraries

This code uses a library named "**lcd128\_32\_io**", if you haven't installed it yet, please do so before learning. The steps to add third-party libraries are as follows:

## 9.18.6 Project code

After the **lcd128\_32\_io** library was added, You can open the code we provideIf you haven't downloaded the code file, please click on the link to download itDownload Arduino C Codes file

The code used in this project is saved in folder "Arduino-Codes\Project 17 I2C 128×32 LCD2.C\_Tutorial\2. Projects\Project\_17\_I2C\_128\_32\_LCD".

```
/*
* Filename : LCD 128*32
* Description : LCD 128*32 display string
* Auther : http//www.keyestudio.com
*/
#include "lcd128_32_io.h"
//Create 1CD128 *32 pinsda--->21 scl--->22
lcd lcd(21, 22);
void setup() {
 lcd.Init(); //initialize
 lcd.Clear(); //clear
}
void loop() {
 lcd.Cursor(0, 4); //Set display position
 lcd.Display("KEYESTUDIO"); //Setting the display
 lcd.Cursor(1, 0);
 lcd.Display("ABCDEFGHIJKLMNOPQR");
 lcd.Cursor(2, 0);
 lcd.Display("123456789+-*/<>=$@");
 lcd.Cursor(3, ∅);
 lcd.Display("%^&(){}:;'|?,.~\\[]");
}
```

## 9.18.7 Project result

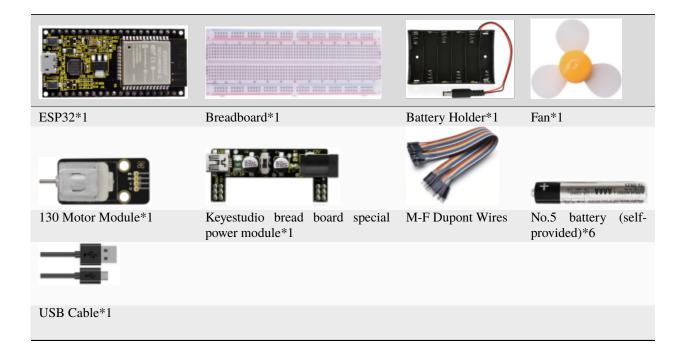
Compile and upload the code to ESP32, after the code is uploaded successfully, power up with a USB cable and you will see that the 128X32LCD module display will show "KEYESTUDIO" at the first line, "ABCDE-FGHIJKLMNOPQR" will be displayed at the second line, "123456789 $\pm$ \*/<>=\$@" will be shown at the third line and "%^&(){}:;'?,-~[]" will be displayed at the fourth line.

## 9.19 Project 18Small Fan

## 9.19.1 Introduction

In hot summer, we need electric fans to cool us down, so in this project, we will use ESP32 control 130 motor module and small fan blade to make a small electric fan.

## 9.19.2 Components



## 9.19.3 Component knowledge :



**130 motor module:** The motor control module uses the HR1124S motor control chip. which is a single-channel Hbridge driver chip for DC motor. The H-bridge driver part of the HR1124S uses low on-resistance PMOS and NMOS power tubes. The low on-resistance ensure low power loss of the chip and make the chip work safely for longer time In addition, In addition, the HR1124S has low standby current and low static operating current, which makes the HR1124S easy to use in toy solutions.

Features:

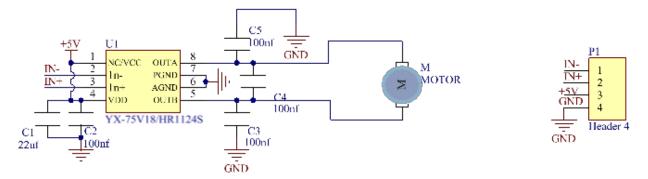
Working voltage: 5V

Working current: 200MA

Working power: 2W

Working temperature:  $-10^{\circ}C \sim +50^{\circ}C$ 

Schematic diagram of 130 motor module



Keyestudio Breadboard Power Supply Module



Introduction:

This breadboard power supply module is compatible with 5V and 3.3V, which can be applied to MB102 breadboard. The module contains two channels of independent control, powered by the USB all the way.

The output voltage is constant for the DC5V, and another way is powered by DC6.5-12V, output controlled by the slide switch, respectively for DC 5V and DC 3.3V.

If the other power supply is DC 6.5-12v, when the slide switch is switched to +5V, the output voltages of the left and right lines of the module are DC 5V. When the slide switch is switched to +3V, the output voltage of the USB power supply terminal of the module is DC5V, and the output voltage of the DC 6.5-12V power supply terminal of the other power supply is DC3.3V.

Specification:

- Applied to MB102 breadboard;
- Input voltageDC 6.5-12V or powered by USB;
- Output voltage3.3V or 5V
- Max output current<700ma
- Up and down two channels of independent control, one of which can be switched to 3.3V or 5V;

Comes with two sets of DC output pins, easy for external use.

## 9.19.4 Wiring Diagram



## 9.19.5 Project code

You can open the code we provide: If you haven't downloaded the code file, please click on the link to download itDownload Arduino C Codes file

The code used in this project is saved in folder "Arduino-Codes\Project 18Small Fan\Project\_18\_ Small\_Fan".

```
/*
* Filename : Small Fan
* Description : Fan clockwise rotation, stop, counterclockwise rotation, stop, cycle.
         : http//www.keyestudio.com
* Auther
*/
#define Motorla 15 // the Motor_IN+ pin of the motor
#define Motorlb 2 // the Motor_IN- pin of the motor
void setup(){
 pinMode(Motorla, OUTPUT);//set Motorla to OUTPUT
 pinMode(Motorlb, OUTPUT);//set Motorlb to OUTPUT
}
void loop(){
//Set to rotate for 5s anticlockwise
 digitalWrite(Motorla,HIGH);
 digitalWrite(Motorlb,LOW);
 delay(5000);
//Set to stop rotating for 2s
 digitalWrite(Motorla,LOW);
 digitalWrite(Motorlb,LOW);
 delay(2000);
//Set to rotate for 5s clockwise
 digitalWrite(Motorla,LOW);
 digitalWrite(Motorlb,HIGH);
 delay(5000);
//Set to stop rotating for 2s
 digitalWrite(Motorla,LOW);
 digitalWrite(Motorlb,LOW);
 delay(2000);
}
```

## 9.19.6 Project result

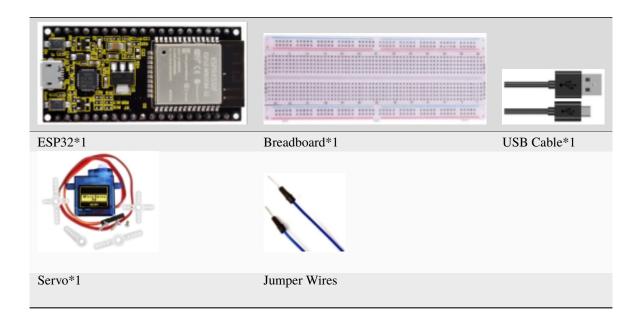
Compile and upload the code to ESP32, after the code is uploaded successfully, power up with a USB cable and you will see that small fan turns counterclockwise for 5 seconds and stops for 2 seconds, and then turns clockwise for 5 seconds and stops for 2 seconds, which repeats in an endless loop.

## 9.20 Project 19Servo Sweep

## 9.20.1 Introduction

Servo is an electric motor that can rotate very precisely. At present, it has been widely used in toy cars, remote control helicoptersairplanesrobots, etc. In this project, we will use ESP32 to control the rotation of the servo.

### 9.20.2 Components



## 9.20.3 Component knowledge

Servo



The servo is a kind of position servo driver, which is mainly composed of a housing circuit board copless motor gear and position detector. Its working principle is that the receiver or microcontroller sends a signal to the servo which has

an internal reference circuit that generates a reference signal with a period of 20ms and a width of 1.5ms, and compares the DC bias voltage with the voltage of the potentiometer to output voltage difference.

The IC on the circuit board determines the direction of rotation, and then drives the coreless motor to start rotation and transmits the power to the swing arm through the reduction gear, while the position detector sends back a signal to determine whether it has reached the positioning. It is suitable for those control systems that require constant change of angle and can be maintained.

When the motor rotates at a certain speed, the potentiometer is driven by the cascade reduction gear to rotate so that the voltage difference is 0 and the motor stops rotating. The angle range of general servo rotation is 0 to 180 degrees.

The pulse period for controlling the servo is 20ms, the pulse width is 0.5ms to 2.5ms, and the corresponding position is -90 degrees to +90 degrees. The following is an example of a 180 degree servo

media/708316fde05c62113a3024e0efb0c237.jpeg

Servo motors have many specifications, but they all have three connecting wires, which are brown, red, and orange (different brands may have different colors). The brown is GND, the red is the positive power supply, and the orange is the signal line.

media/3f5bc31305e64108bed3b3619d602891.jpeg

### 9.20.4 Wiring Diagram

When supplying the servo, please note that the power supply voltage should be 3.3V-5V. Make sure there are no errors when connecting the servo to the power supply.

media/39621cc861e5f7c189a047b7f0bbd0be.png

### 9.20.5 Adding the ESP32Servo library

If you have not downloaded the library file, please click on the link to download it:Download Arduino Libraries

This code uses a library named "**ESP32Servo**", If you haven't installed it yet, please do so before learning. The steps to add third-party libraries are as follows:

## 9.20.6 Project code

After the **ESP32Servo** library is added, You can open the code we provideIf you haven't downloaded the code file, please click on the link to download itDownload Arduino C Codes file

The code used in this project is saved in folder "Arduino-Codes\Project 19Servo Sweep\Project\_19\_Servo\_Sweep".

```
/*
* Filename : Servo Sweep
* Description : Control the servo motor for sweeping
* Auther : http//www.keyestudio.com
*/
#include <ESP32Servo.h>
Servo myservo; // create servo object to control a servo
int posVal = 0; // variable to store the servo position
int servoPin = 15; // Servo motor pin
void setup() {
 myservo.setPeriodHertz(50);
                                 // standard 50 hz servo
 myservo.attach(servoPin, 500, 2500); // attaches the servo on servoPin to the servo.
→object
}
void loop() {
 for (posVal = 0; posVal <= 180; posVal += 1) \{ // goes from 0 degrees to 180 degrees
   // in steps of 1 degree
                           // tell servo to go to position in variable 'pos'
   myservo.write(posVal);
                            // waits 15ms for the servo to reach the position
   delay(15);
 }
 for (posVal = 180; posVal >= 0; posVal -= 1) { // goes from 180 degrees to 0 degrees
   myservo.write(posVal); // tell servo to go to position in variable 'pos'
   delay(15);
                            // waits 15ms for the servo to reach the position
 }
}
```

#### 9.20.7 Project result

Compile and upload the code to ESP32, after the code is uploaded successfully, power up with a USB cable and you will see that the Servo will rotate from 0 degrees to 180 degrees and then reverse the direction to make it rotate from 180 degrees to 0 degrees and repeat these actions in an endless loop.

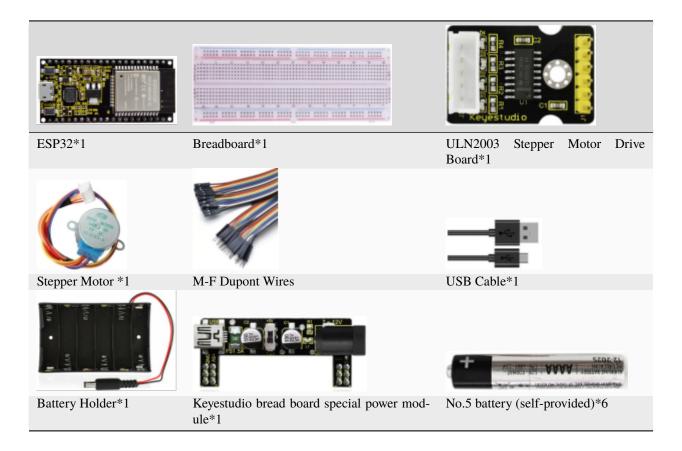
media/c5250405a4290ecb2d758ff1097310c7.png

# 9.21 Project 20Stepping Motor

## 9.21.1 Introduction

Stepper motor is the most important part of industrial robot 3D printer lathes and other mechanical equipment with accurate positioning. In this project, we will use ESP32 control ULN2003 stepper motor drive board to drive the stepper motor to rotate.

## 9.21.2 Components



## 9.21.3 Component knowledge



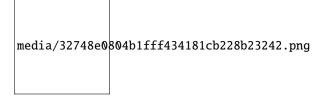
**Stepper motor:** It is a motor controlled by a series of electromagnetic coils. It can rotate by the exact number of degrees (or steps) needed, allowing you to move it to a precise position and keep it there. It does this by supplying

power to the coil inside the motor in a very short time, but you must always supply power to the motor to keep it in the position you want. There are two basic types of stepping motors, namely unipolar stepping motor and bipolar stepping motor. In this project, we use a 28-BYJ48 unipolar stepper motor.



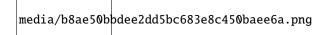
Working Principle:

The stepper motor is mainly composed of a stator and a rotor. The stator is fixed. As shown in the figure below, the part of the coil group A, B, C, and D will generate a magnetic field when the coil group is energized. The rotor is the rotating part. As follows, the middle part of the stator, two poles are permanent magnets.



Single -phase four beat: At the beginning, the coils of group A are turned on, and the poles of the rotor point at A coil. Next, the group A coil are disconnected, and the group B coils are turned on. The rotor will turn clockwise to the group B. Then, group B is disconnected, group C is turned on, and the rotor is turned to group C. After that, group C is disconnected, and group D is turned on, and the rotor is turned to group D. Finally, group D is disconnected, group A coils. Therefore, rotor turns 180° and continuously rotates B-C-D-A, which means it runs a circle (eight phase). As shown below, he rotation principle of stepper motor is A - B C - D - A.

You make order inverse(D - C - B - A - D ...) if you want to make stepper motor rotate anticlockwise.



Half-phase and eight beat: 8 beat adopts single and dual beat wayA - AB B - BC - C - CD - D - DA - A  $\dots$  rotor will rotate half phase in this order. For example, when A coil is electrifiedrotor faces to A coil, then A and B coil are connected, on this condition, the strongest magnetic field produced lies in the central part of AB coil, which means rotating half-phase clockwise.

Stepper Motor Parameters:

The rotor rotates one circle when the stepper motor we provide rotates 32 phases and with the output shaft driven by 1:64 reduction geared set. Therefore the rotation (a circle) of output shaft requires 32 \* 64 = 2048 phases.

The step angle of 4-beat mode of 5V and 4-phase stepper motor is 11.25. And the step angle of 8-beat mode is 5.625, the reduction ratio is 1:64.

**ULN2003Stepper Motor Drive Board:** It is a stepper motor driver, which converts the weak signal into a stronger control signal to drive the stepper motor.

The following schematic diagram shows how to use the ULN2003 stepper motor driver board interface to connect a unipolar stepper motor to the pins of the ESP32, and shows how to use four TIP120 interfaces.

media/6fa632d2b70e97dd55565d23ec15d245.png

## 9.21.4 Wiring Diagram



## 9.21.5 Project code

You can open the code we provide: If you haven't downloaded the code file, please click on the link to download itDownload Arduino C Codes file

The code used in this project is saved in folder "Arduino-Codes\Project 20Stepping Motor\Project\_20\_Stepping\_Motor".

```
/*
* Filename : Drive Stepper Motor
* Description : Use ULN2003 to drive the stepper motor.
* Auther : http://www.keyestudio.com
*/
// Conncet the port of the stepper motor driver
int outPorts[] = {15, 16, 17, 18};
void setup() {
 // set pins to output
 for (int i = 0; i < 4; i++) {
   pinMode(outPorts[i], OUTPUT);
 }
}
void loop()
{
 // Rotate a full turn
 moveSteps(true, 32 * 64, 3);
 delay(1000);
 // Rotate a full turn towards another direction
```

(continues on next page)

(continued from previous page)

```
moveSteps(false, 32 * 64, 3);
  delay(1000);
}
//Suggestion: the motor turns precisely when the ms range is between 3 and 20
void moveSteps(bool dir, int steps, byte ms) {
  for (unsigned long i = 0; i < steps; i++) {</pre>
   moveOneStep(dir); // Rotate a step
   delay(constrain(ms,3,20));
                                    // Control the speed
 }
}
void moveOneStep(bool dir) {
  // Define a variable, use four low bit to indicate the state of port
 static byte out = 0 \times 01;
 // Decide the shift direction according to the rotation direction
 if (dir) { // ring shift left
   out != 0 \times 08 ? out = out << 1 : out = 0 \times 01;
  }
           // ring shift right
  else {
   out != 0 \times 01 ? out = out >> 1 : out = 0 \times 08;
 // Output singal to each port
 for (int i = 0; i < 4; i++) {
    digitalWrite(outPorts[i], (out & (0x01 << i)) ? HIGH : LOW);</pre>
  }
}
void moveAround(bool dir, int turns, byte ms){
  for(int i=0;i<turns;i++)</pre>
   moveSteps(dir,32*64,ms);
}
void moveAngle(bool dir, int angle, byte ms){
 moveSteps(dir,(angle*32*64/360),ms);
}
```

## 9.21.6 Project result

Compile and upload the code to ESP32, after the code is uploaded successfully, power up with a USB cable and you will see that the four LEDs (D1,D2,D3,D4) on the ULN2003 drive module will light up. The stepper motor rotates clockwise first, then counterclockwise, and repeat these actions in an endless loop.

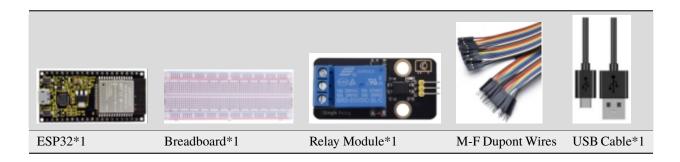
media/8dc4a0547390e0108c3960c31d330ee7.png

# 9.22 Project 21Relay

## 9.22.1 Introduction

In our daily life, we usually use communication to drive electrical equipments, and sometimes we use switches to control electrical equipments. If the switch is connected directly to the ac circuit, leakage occurs and people are in danger. Therefore, from the perspective of safety, we specially designed this relay module with NO(normally open) end and NC(normally closed) end. In this project, we will learn a relatively special and easy-to-use switch, which is the relay module.

## 9.22.2 Components



## 9.22.3 Component knowledge

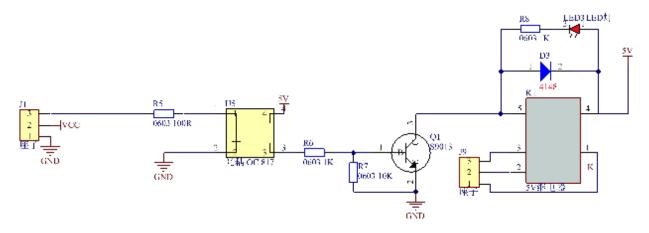
Relay: It is an "automatic switch" that uses a small current to control the operation of a large current.

Input voltage3.3V-5V

Rated load5A 250VAC (NO/NC) 5A 24VDC (NO/NC)

The rated load means that devices with dc voltage of 24V or AC voltage of 250V can be controlled using 3.3V-5V microcontrollers.

Schematic diagram of Relay



## 9.22.4 Wiring Diagram



## 9.22.5 Project code

You can open the code we provide: If you haven't downloaded the code file, please click on the link to download itDownload Arduino C Codes file

The code used in this project is saved in folder "Arduino-Codes\Project 21Relay\Project\_21\_Relay".

```
/*
* Filename : Relay
* Description : Relay turn on and off.
* Auther : http//www.keyestudio.com
*/
#define Relay 15 // defines digital 15
void setup()
{
pinMode(Relay, OUTPUT); // sets "Relay" to "output"
}
void loop()
{
digitalWrite(Relay, HIGH); // turns on the relay
delay(1000); //delays 1 seconds
digitalWrite(Relay, LOW); // turns off the relay
delay(1000); // delays 1 seconds
}
```

## 9.22.6 Project result

Compile and upload the code to ESP32, after the code is uploaded successfully, power up with a USB cable and you will see that the relay will cycle on and off, on for 1 second, off for 1 second. At the same time, you can hear the sound of the relay on and off, and you can also see the change of the indicator light on the relay.

# 9.23 Project 22Dimming Light

## 9.23.1 Introduction

A potentiometer is a three-terminal resistor with sliding or rotating contacts that forms an adjustable voltage divider. It works by changing the position of the sliding contacts across a uniform resistance. In the potentiometer, the entire input voltage is applied across the whole length of the resistor, and the output voltage is the voltage drop between the fixed and sliding contact.

In this project, we will learn how to use ESP32 to read the values of the potentiometer, and make a dimming lamp with LED.

## 9.23.2 Components

ESP32*1	Breadboard*1	Potentiometer*1	Red LED*1
220Resistor*1	Jumper Wires	USB Cable*1	

## 9.23.3 Component knowledge



Adjustable potentiometer: It is a kind of resistor and an analog electronic component, which has two states of 0 and 1(high level and low level). The analog quantity is different, its data state presents a linear state such as  $1 \sim 1024$ 

**ADC :** An ADC is an electronic integrated circuit used to convert analog signals such as voltages to digital or binary form consisting of 1s and 0s. The range of our ADC on ESP32 is 12 bits, that means the resolution is 2^12=4096, and it represents a range (at 3.3V) will be divided equally to 4096 parts. The rage of analog values corresponds to ADC values. So the more bits the ADC has, the denser the partition of analog will be and the greater the precision of the resulting conversion.



Subsection 1: the analog in rang of 0V—3.3/4095 V corresponds to digital 0;

Subsection 2: the analog in rang of 3.3/4095 V—2\*3.3 /4095V corresponds to digital 1;

The following analog will be divided accordingly.

The conversion formula is as follows:

$$ADCValue = \frac{Ana\log Voltage}{3.3} * 4095$$

\*\*DAC\*\*The reversing of this process requires a DAC, Digital-to-Analog Converter. The digital I/O port can output high level and low level (0 or 1), but cannot output an intermediate voltage value. This is where a DAC is useful. ESP32 has two DAC output pins with 8-bit accuracy, GPIO25 and GPIO26, which can divide VCC

(here is 3.3V) into  $2^8=256$  parts. For example, when the digital quantity is 1, the output voltage value is 3.3/256 \* 1 V, and when the digital quantity is 128, the output voltage value is 3.3/256\*128=1.65V, the higher the accuracy of DAC, the higher the accuracy of output voltage value will be.

The conversion formula is as follows:

Ana log Voltage =  $\frac{DACValue}{255}$  \* 3.3(V)

#### ADC on ESP32

ESP32 has 16 pins can be used to measure analog signals. GPIO pin sequence number and analog pin definition are shown in the following table

ADC number in ESP32	ESP32 GPIO number
ADC0	GPIO 36
ADC3	GPIO 39
ADC4	GPIO 32
ADC5	GPIO33
ADC6	GPIO34
ADC7	GPIO 35
ADC10	GPIO 4
ADC11	GPIO0
ADC12	GPIO2
ADC13	GPIO15
ADC14	GPIO13
ADC15	GPIO 12
ADC16	GPIO 14
ADC17	GPIO27
ADC18	GPIO25
ADC19	GPIO26

#### DAC on ESP32

ESP32 has two 8-bit digital analog converters to be connected to GPIO25 and GPIO26 pins, respectively, and it is immutable. As shown in the following table

Simulate pin number	GPIO number
DAC1	GPIO25
DAC2	GPIO26

The DAC pin number is already defined in ESP32's code base; for example, you can replace GPIO25 with DAC1 in the code.

#### 9.23.4 Read the ADC valueDAC value and voltage value of the potentiometer

We connect the potentiometer to the analog IO port of ESP32 to read the ADC valueDAC value and voltage value of the potentiometer, please refer to the wiring diagram below

media/0cda3256a0930404abc097ec8ffa3013.png

You can open the code we provide: If you haven't downloaded the code file, please click on the link to download itDownload Arduino C Codes file

The code used in this project is saved in folder "Arduino-Codes\Project 22Dimming Light\Project\_22.1\_Read\_Potentiometer\_Analog\_Value".

(continues on next page)

(continued from previous page)

```
* Description : Basic usage of ADCDAC and Voltage
 * Auther
              : http//www.keyestudio.com
*/
#define PIN_ANALOG_IN 36 //the pin of the Potentiometer
void setup() {
 Serial.begin(115200);
}
//In loop()the analogRead() function is used to obtain the ADC value, and then the map()_
\rightarrow function is used to convert the value into an 8-bit precision DAC value. The input and
-output voltage are calculated according to the previous formula, and the information.
\rightarrow is finally printed out.
void loop() {
 int adcVal = analogRead(PIN_ANALOG_IN);
 int dacVal = map(adcVal, 0, 4095, 0, 255);
 double voltage = adcVal / 4095.0 * 3.3;
 Serial.printf("ADC Val: %d, \t DAC Val: %d, \t Voltage: %.2fV\n", adcVal, dacVal,
→voltage);
 delay(200);
}
```

Compile and upload the code to ESP32, after the code is uploaded successfully, power up with a USB cable, open the serial monitor and set the baud rate to 115200. You need to press the reset button on the ESP32 mainboard first, and then you will see that the serial port monitor window will print out the ADC valueDAC value and voltage value of the potentiometer. When turning the potentiometer handle, the ADC valueDAC value and voltage value will change. As shown below:

	/dev/ttyUSB0	~ ^ X
		Send
ADC Val:         38,         DAC Val:         2,           ADC Val:         443,         DAC Val:         28,           ADC Val:         477,         DAC Val:         28,           ADC Val:         477,         DAC Val:         30,           ADC Val:         490,         DAC Val:         31,           ADC Val:         753,         DAC Val:         47,           ADC Val:         928,         DAC Val:         58,           ADC Val:         925,         DAC Val:         58,           ADC Val:         1359,         DAC Val:         85,           ADC Val:         1421,         DAC Val:         88,	Voltage: 0.61V Voltage: 0.75V Voltage: 0.75V Voltage: 1.10V Voltage: 1.15V Voltage: 1.18V Voltage: 1.43V	
<ul> <li>✓ Autoscroll □ Show timestamp</li> </ul>	Newline • 11520	0 baud 🔹 Clear output

## 9.23.5 Wiring diagram of the dimming lamp

In the previous step, we read the ADC valueDAC value and voltage value of the potentiometer. Now we need to convert the ADC value of the potentiometer into the brightness of the LED to make a lamp that can adjust the brightness. The wiring diagram is as follows:

```
media/3396bd77169711de6e15da73f14c8afb.png
```

## 9.23.6 Project code

You can open the code we provide: If you haven't downloaded the code file, please click on the link to download itDownload Arduino C Codes file

The code used in this project is saved in folder "Arduino-Codes\Project 22Dimming Light\Project\_22.2\_Dimming\_Light".

```
/*
* Filename : Dimming Light
* Description : Controlling the brightness of LED by potentiometer.
       : http//www.keyestudio.com
* Auther
*/
#define PIN_ANALOG_IN
                36 //the pin of the potentiometer
#define PIN_LED 15 // the pin of the LED
#define CHAN
                 0
void setup() {
 ledcSetup(CHAN, 1000, 12);
 ledcAttachPin(PIN_LED, CHAN);
}
void loop() {
 int adcVal = analogRead(PIN_ANALOG_IN); //read adc
                 // adcVal re-map to pwmVal
 int pwmVal = adcVal;
                     // set the pulse width.
 ledcWrite(CHAN, pwmVal);
 delay(10);
}
```

## 9.23.7 Project result

Compile and upload the code to ESP32, after the code is uploaded successfully, power up with a USB cable and you will see that turn the potentiometer handle and the brightness of the LED will change accordingly.

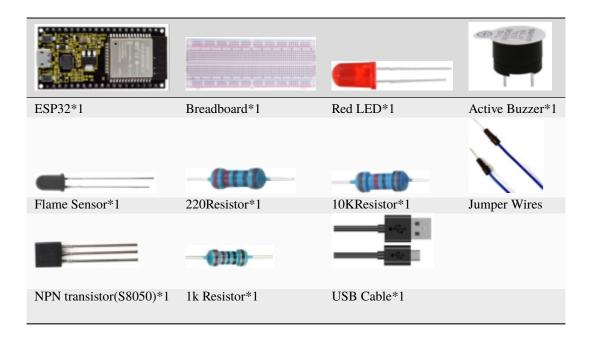
media/eca30dead3f4923afa0dcb0306db2319.jpeg

# 9.24 Project 23Flame Alarm

## 9.24.1 Introduction

Fire is a terrible disaster and fire alarm systems are very useful in housescommercial buildings and factories. In this project, we will use ESP32 to control a flame sensor, a buzzer and a LED to simulate fire alarm devices. This is a meaningful maker activity.

## 9.24.2 Components



## 9.24.3 Component knowledge



The flame emits a certain amount IR light that is invisible to the human eye, but our flame sensor can detect it and alert a microcontroller(such as ESP32) that a fire has been detected. It has a specially designed infrared receiver tube to detect the flame and then convert the flame brightness into a fluctuating level signal. The short pin of the receiving triode is negative pole and the other long pin is positive pole. We should connect the short pin (negative) to 5V and the long pin(positive) to the analog pin, a resistor and GND. As shown in the figure below

media/87bd204db523c602c80745266c1ee452.png

**Note:** Since vulnerable to radio frequency radiation and temperature changes, the flame sensor should be kept away from heat sources like radiators, heaters and air conditioners, as well as direct irradiation of sunlight, headlights and incandescent light.

## 9.24.4 Read the ADC valueDAC value and voltage value of the flame sensor

We first use a simple code to read the ADC valueDAC value and voltage value of the flame sensor and print them out. Please refer to the wiring diagram below

media/76ce57355da1df27e049bdc6e19f0650.png

You can open the code we provide: If you haven't downloaded the code file, please click on the link to download itDownload Arduino C Codes file

The code used in this project is saved in folder "Arduino-Codes\Project 23Flame Alarm\Project\_23.1\_Read\_Analog\_Value\_Of\_Flame\_Sensor".

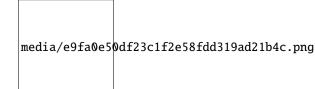
```
/*
* Filename
          : Read Analog Value Of Flame Sensor
* Description : Basic usage of ADCDAC and Voltage
           : http//www.keyestudio.com
* Auther
*/
#define PIN_ANALOG_IN 36 //the pin of the Flame sensor
void setup() {
 Serial.begin(115200);
}
//In loop()the analogRead() function is used to obtain the ADC value, and then the map()_
\rightarrow function is used to convert the value into an 8-bit precision DAC value. The input and
\rightarrowoutput voltage are calculated according to the previous formula, and the information.
\rightarrow is finally printed out.
void loop() {
 int adcVal = analogRead(PIN_ANALOG_IN);
 int dacVal = map(adcVal, 0, 4095, 0, 255);
 double voltage = adcVal / 4095.0 * 3.3;
 Serial.printf("ADC Val: %d, \t DAC Val: %d, \t Voltage: %.2fV\n", adcVal, dacVal,
→voltage);
 delay(200);
}
```

Compile and upload the code to ESP32, after the code is uploaded successfully, power up with a USB cable, open the serial monitor and set the baud rate to 115200. You need to press the reset button on the ESP32 mainboard first, and then you will see that the serial port monitor window will print out the ADC valueDAC value and voltage value of the flame sensor. When the sensor is closed to fire, the ADC valueDAC value and voltage value will get greater. Conversely, the ADC valueDAC value and voltage value decrease.

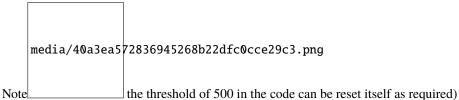
		/dev/ttyUSB0	~ ^ X
			Send
ADC Val: 58, ADC Val: 43, ADC Val: 38, ADC Val: 443, ADC Val: 447, ADC Val: 490, ADC Val: 490, ADC Val: 753, ADC Val: 928, ADC Val: 925, ADC Val: 925, ADC Val: 1359, ADC Val: 1421, ADC Val: 1775, ADC Val: 1915,	DAC Val: 2, DAC Val: 28, DAC Val: 30, DAC Val: 31, DAC Val: 47, DAC Val: 58, DAC Val: 58, DAC Val: 58, DAC Val: 85, DAC Val: 88, DAC Val: 91, DAC Val: 111,	Voltage: 0.05V Voltage: 0.03V Voltage: 0.03V Voltage: 0.36V Voltage: 0.38V Voltage: 0.39V Voltage: 0.61V Voltage: 0.75V Voltage: 0.75V Voltage: 1.16V Voltage: 1.18V Voltage: 1.43V Voltage: 1.54V	
<ul> <li>✓ Autoscroll □ 5</li> </ul>	Show timestamp	Newline • 115	200 baud 🔹 Clear output

### 9.24.5 Wiring diagram of the flame alarm

Next, we will use a flame sensor, a buzzer, and a LED to make an interesting project, that is flame alarm. When flame is detected, the LED flashes and the buzzer alarms.



## 9.24.6 Project code



You can open the ends we provide. If you haven't developed the ends file places click on the link

You can open the code we provide: If you haven't downloaded the code file, please click on the link to download itDownload Arduino C Codes file

The code used in this project is saved in folder "Arduino-Codes\Project 23Flame Alarm\Project\_23.2\_Flame\_Alarm".

```
/*
* Filename
            : Flame Alarm
* Description : Controlling the buzzer and LED by flame sensor.
           : http//www.keyestudio.com
* Auther
*/
                  36 //the pin of the flame sensor
#define PIN_ADC0
#define PIN_LED
                  15 // the pin of the LED
#define PIN_BUZZER
                  4 // the pin of the buzzer
void setup() {
 pinMode(PIN_LED, OUTPUT);
 pinMode(PIN_BUZZER, OUTPUT);
 pinMode(PIN_ADC0, INPUT);
}
void loop() {
 int adcVal = analogRead(PIN_ADC0); //read the ADC value of flame sensor
 if (adcVal >= 500) {
   digitalWrite (PIN_BUZZER, HIGH); //turn on buzzer
   digitalWrite(PIN_LED, HIGH); // turn on LED
   delay(500); // wait a second.
   digitalWrite (PIN_BUZZER, LOW);
   digitalWrite(PIN_LED, LOW); // turn off LED
   delay(500); // wait a second
 }
else
{
   digitalWrite(PIN_LED, LOW); //turn off LED
   digitalWrite (PIN_BUZZER, LOW); //turn off buzzer
 }
}
```

## 9.24.7 Project result

Compile and upload the code to ESP32, after the code is uploaded successfully, power up with a USB cable and you will see that when the flame sensor detects the flame, the LED will flash and the buzzer will alarm; otherwise, the LED does not light up and the buzzer does not sound.

## 9.25 Project 24Night Lamp

## 9.25.1 Introduction

Sensors or components are ubiquitous in our daily life. For example, some public street lamps will automatically turn on at night and turn off during the day. Why? In fact, this make use of a photosensitive element that senses the intensity of external ambient light. When the outdoor brightness decreases at night, the street lights will turn on automatically; In the daytime, the street lights will automatically turn off. the principle of which is very simple, In this Project, we use ESP32 to control a LED to achieve the effect of the street light.

## 9.25.2 Components

			-(111)
ESP32*1	Breadboard*1	Red LED*1	10KResistor*1
6			
Photoresistor*1	220Resistor*1	Jumper Wires	USB Cable*1

## 9.25.3 Component knowledge



**Photoresistor :** It is a kind of photosensitive resistance, its principle is that the photoresistor surface receives brightness (light) to reduce the resistance, the resistance value will change with the detected intensity of the ambient light. With this characteristic, we can use the photosensitive resistance to detect the light intensity. Photosensitive resistance and its electronic symbol are as follows

media/7d575da675a2f6cb511d28b801e2abaa.png

The following circuit is used to detect changes in resistance values of photoresistors



In the circuit above, when the resistance of the photoresistor changes due to the change of light intensity, the voltage between thephotoresistor and resistance R2 will also change. Thus, the intensity of light can be obtained by measuring this voltage.

#### 4. Read the ADC valueDAC value and voltage value of the photoresistor

We first use a simple code to read the ADC valueDAC value and voltage value of the photoresistor and print them out. Please refer to the following wiring diagram

media/b762098c798beb08e4d433137c317dc7.png

You can open the code we provide: If you haven't downloaded the code file, please click on the link to download itDownload Arduino C Codes file

The code used in this project is saved in folder "Arduino-Codes\Project 24Night Lamp\Project\_24.1\_Read\_Photosensitive\_Analog\_Value".

```
/*
* Filename
          : Read Photosensitive Analog Value
* Description : Basic usage of ADC
* Auther
          : http//www.keyestudio.com
*/
#define PIN_ANALOG_IN 36 //the pin of the photosensitive sensor
void setup() {
 Serial.begin(115200);
}
//In loop()the analogRead() function is used to obtain the ADC value, and then the map()_
\rightarrow function is used to convert the value into an 8-bit precision DAC value. The input and
-output voltage are calculated according to the previous formula, and the information.
\rightarrow is finally printed out.
void loop() {
 int adcVal = analogRead(PIN_ANALOG_IN);
 int dacVal = map(adcVal, 0, 4095, 0, 255);
 double voltage = adcVal / 4095.0 * 3.3;
 Serial.printf("ADC Val: %d, \t DAC Val: %d, \t Voltage: %.2fV\n", adcVal, dacVal,
\rightarrow voltage);
 delay(200);
}
           *******
```

Compile and upload the code to ESP32, after the code is uploaded successfully, power up with a USB cable, open the serial monitor and set the baud rate to 115200. You need to press the reset button on the ESP32 mainboard first, and then you will see that the serial port monitor window will print out the ADC value, DAC value and voltage value of the photoresistor. When the light intensity around the photoresistor is gradually reduced, the ADC value, DAC value and voltage value

		/dev/ttyUSB0	~ ^ X
			Send
DC Val: 58,	DAC Val: 4,	Voltage: 0.05V	
DC Val: 43,	DAC Val: 3,	Voltage: 0.03V	
DC Val: 38,	DAC Val: 2,	Voltage: 0.03V	
DC Val: 443,	DAC Val: 28,	Voltage: 0.36V	
DC Val: 477,	DAC Val: 30,	Voltage: 0.38V	
DC Val: 490,	DAC Val: 31,	Voltage: 0.39V	
DC Val: 753,	DAC Val: 47,	Voltage: 0.61V	
DC Val: 928,	DAC Val: 58,	Voltage: 0.75V	
DC Val: 925,	DAC Val: 58,	Voltage: 0.75V	
DC Val: 1359,	DAC Val: 85,	Voltage: 1.10V	
DC Val: 1421,	DAC Val: 88,	Voltage: 1.15V	
DC Val: 1463,	DAC Val: 91,	Voltage: 1.18V	
DC Val: 1775,	DAC Val: 111,	Voltage: 1.43V	
DC Val: 1915,	DAC Val: 119,	Voltage: 1.54V	
			•
✓ Autoscroll □ :	Show timestamp	Newline • 115	200 baud 👻 Clear output
Autoacion	onow unestamp	Newline	200 badd

## 9.25.4 Wiring diagram of the light-controlled lamp

We made a small dimming lamp in the front, now we will make a light controlled lamp. The principle is the same, that is, the ESP32 takes the ADC value of the sensor, and then adjusts the brightness of the LED.

media/77a0c534501f51e7fe7a	aa221e4db71d9.png
----------------------------	-------------------

## 9.25.5 Project code

You can open the code we provide: If you haven't downloaded the code file, please click on the link to download itDownload Arduino C Codes file

The code used in this project is saved in folder\*\*"Arduino-Codes\Project 24Night Lamp\Project\_24.2\_Night\_Lamp"\*\*.

```
/*
* Filename
          : Night Lamp
* Description : Controlling the brightness of LED by photosensitive sensor.
* Auther
        : http//www.keyestudio.com
*/
#define PIN_ANALOG_IN
                   36 // the pin of the photosensitive sensor
#define PIN_LED 15 // the pin of the LED
#define CHAN
                   0
#define LIGHT_MIN
                   372
#define LIGHT_MAX
                   2048
void setup() {
 ledcSetup(CHAN, 1000, 12);
```

(continues on next page)

(continued from previous page)

## 9.25.6 Project result

Compile and upload the code to ESP32, after the code is uploaded successfully, power up with a USB cable and you will see that when the intensity of light around the photoresistor is reduced, the LED will be bright, on the contraty, the LED will be dim.

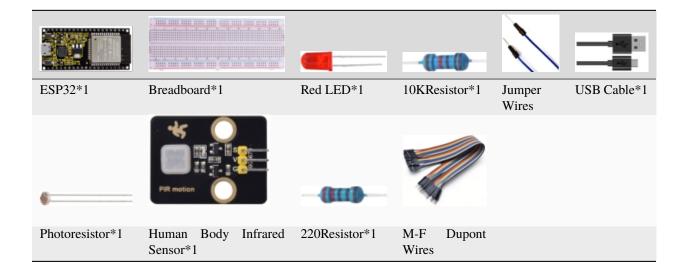
## 9.26 Project 25Human Induction Lamp

## 9.26.1 Introduction

Human body induction lamp is used commonly in the dark corridor area. With the development of science and technology, the use of the human body induction lamp is very common in our real life, such as the corridor of the community, the bedroom of the room, the garage of the dungeon, the bathroom and so on. The human induction lamp are generally composed of a human body infrared sensor, a led, a photoresistor sensor and so on.

In this project, we will learn how to use a Human Body Infrared Sensor, a led, and a photoresistor to make a human induction lamp.

## 9.26.2 Components



## 9.26.3 Wiring Diagram



## 9.26.4 Project code

You can open the code we provide: If you haven't downloaded the code file, please click on the link to download itDownload Arduino C Codes file

The code used in this project is saved in folder "Arduino-Codes\Project 25Human Induction Lamp\Project\_25\_Human\_Induction\_Lamp".

```
/*
* Filename
           : Human Induction Lamp
* Description : Controlling the LED by photosensitive sensor and PIR motion sensor.
* Auther
           : http//www.keyestudio.com
*/
#define PIN_ADC0 36 //the pin of the photosensitive sensor
#define PIN_LED 4 // the pin of the LED
#define pirPin 15
                   // the pin of the PIR motion sensor
byte pirStat = 0; // the state of the PIR motion sensor
void setup() {
 Serial.begin(115200);
 pinMode(PIN_LED, OUTPUT);
 pinMode(PIN_ADC0, INPUT);
 pinMode(pirPin, INPUT);
}
void loop() {
 int adcVal = analogRead(PIN_ADC0); //read the ADC value of photosensitive sensor
 pirStat = digitalRead(pirPin); //read the value of PIR motion sensor
 if (adcVal >= 2000) {
    if (pirStat == HIGH){
       digitalWrite(PIN_LED, HIGH);//turn on the LED
       }
     else{
       digitalWrite(PIN_LED, LOW);//turn off the LED
      }
 }
  else{
     digitalWrite(PIN_LED, LOW);//turn off the LED
     }
}
```

## 9.26.5 Project result

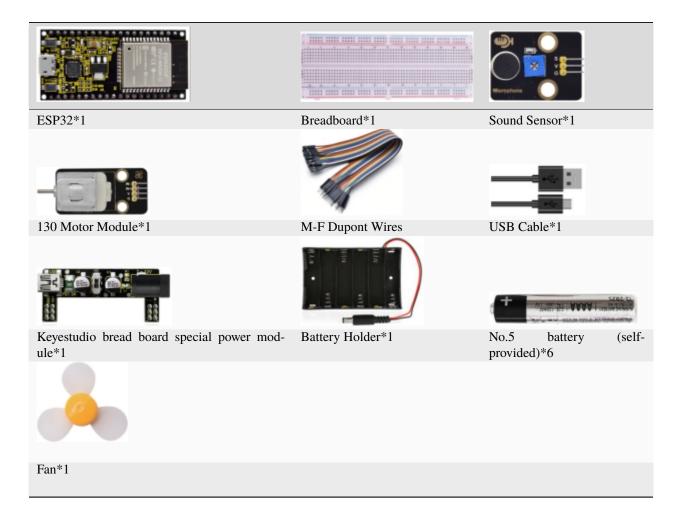
Compile and upload the code to ESP32, after the code is uploaded successfully, power up with a USB cable and you will see thatWhen your hand covers the photosensitive part of the photoresistor to simulate darkness, then shake your other hand in front of the Human Body Infrared Sensor, the external LED will light up. If the photosensitive part of the photoresistor is not covered, then shake your hand in front of the human infrared sensor and the LED is turned off.

# 9.27 Project 26Sound Control Fan

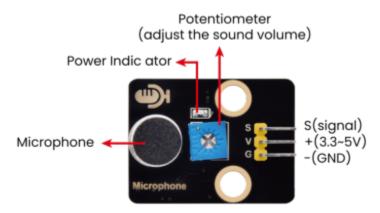
## 9.27.1 Introduction

The sound sensor has a built-in capacitive electret microphone and power amplifier which can be used to detect the sound intensity of the environment. In this project, we use ESP32 to control the sound sensor and the motor module to simulate a voice-controlled fan.

## 9.27.2 Components



## 9.27.3 Component knowledge



Sound sensor is usually used to detect the loudness of the sound in the surrounding environment. Microcontrol board can collect its output signal through the analog input interface. The S pin is an analog output, which is the real-time output of the microphone voltage signal. The sensor comes with a potentiometer so you can adjust the signal strength. It also has two fixing holes so that the sensor can be installed on any other equipment. You can use it to make some interactive works, such as voice-operated switches.

## 9.27.4 Read the ADC valueDAC value and voltage value of the sound sensor

We first use a simple code to read the ADC valueDAC value and voltage value of the sound sensor and print them out. Please refer to the wiring diagram below



You can open the code we provide: If you haven't downloaded the code file, please click on the link to download itDownload Arduino C Codes file

The code used in this project is saved in folder "Arduino-Codes\Project 26Sound Control Fan\Project\_26.1\_Read\_Sound\_Sensor\_Analog\_Value".

```
/*
* Filename
           : Read Sound Sensor Analog Value
 * Description : Basic usage of ADC
* Auther
          : http//www.keyestudio.com
*/
#define PIN_ANALOG_IN 36 //the pin of the Sound Sensor
void setup() {
 Serial.begin(115200);
}
//In loop()the analogRead() function is used to obtain the ADC value, and then the map()_
\rightarrow function is used to convert the value into an 8-bit precision DAC value. The input and
-output voltage are calculated according to the previous formula, and the information.
                                                                  (continues on next page)
```

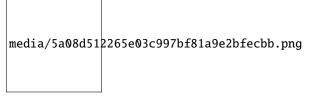
(continued from previous page)

Compile and upload the code to ESP32, after the code is uploaded successfully, power up with a USB cable, open the serial monitor and set the baud rate to 115200. You need to press the reset button on the ESP32 mainboard first, and then you will see that the serial port monitor window will print out the ADC valueDAC value and voltage value of the sound sensor. When you clap your hands to the sensor, the ADC valueDAC value and voltage value will change significantly.

						JSB0					
											Send
ADC Val: ADC Val: ADC Val: ADC Val: ADC Val: ADC Val: ADC Val: ADC Val: ADC Val: ADC Val:	0,         D/           0,         D/           1187,         D/           551,         D/           0,         D/           2112,         D/           0,         D/           34,         D/           35,         D/           0,         D/	AC Val: AC Val:	0, 0, 74, 34, 0, 0, 132, 0, 2, 2, 0, 0, 0,	Voltage: Voltage: Voltage: Voltage: Voltage: Voltage: Voltage: Voltage: Voltage:	0.00V 0.96V 0.44V 0.00V 0.00V 1.70V 0.00V 0.00V 0.03V 0.03V 0.00V 0.00V						
<ul> <li>Autose</li> </ul>	croll 🗌 Show	r timestan	np			Newline	•	115200 baud	j 🔹	Clear	output

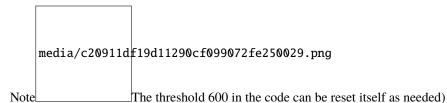
## 9.27.5 Wiring diagram of the intelligent fan

Next, we officially entered the project. We used a sound sensora motor module and a fan blade to simulate a voicecontrolled fan. The wiring diagram is as follows



(Note: Connect the wires and then install a small fan blade on the DC motor. )

### 9.27.6 Project code



You can open the code we provide: If you haven't downloaded the code file, please click on the link to download itDownload Arduino C Codes file

The code used in this project is saved in folder "Arduino-Codes\Project 26Sound Control Fan\Project\_26.2\_Sound\_Control\_Fan".

```
//***********************
/*
* Filename : Sound Control Fan
* Description : Controlling the fan by Sound sensor.
* Auther : http//www.keyestudio.com
*/
#define PIN_ADC0 36 //the pin of the Sound sensor
#define PIN_Motorla 15 // the Motor_IN+ pin of the motor
#define PIN_Motorlb 2 // the Motor_IN- pin of the motor
void setup() {
 pinMode(PIN_Motorla, OUTPUT);//set Motorla to OUTPUT
 pinMode(PIN_Motorlb, OUTPUT);//set Motorlb to OUTPUT
 pinMode(PIN_ADC0, INPUT);//set PIN_ADC2 to INPUT
}
void loop() {
 int adcVal = analogRead(PIN_ADC0); //read the ADC value of Sound sensor
 if (adcVal > 600) {
   digitalWrite(PIN_Motorla,HIGH); //rotate
   digitalWrite(PIN_Motorlb,LOW);
   delay(5000); //delay 5S
 }
 else
 {
   digitalWrite(PIN_Motorla,LOW); //stop rotating
   digitalWrite(PIN_Motorlb,LOW);
 }
}
```

## 9.27.7 Project result

Compile and upload the code to ESP32, after the code is uploaded successfully, external power supply and power on. and then you will see that clap your hands to the sound sensor, and when the sound intensity exceeds a threshold, the small fan rotates; conversely, the small fan doesn't rotate.

## 9.28 Project 27Temperature Measurement

## 9.28.1 Introduction

LM35 is a common used and easy-to-use temperature sensor. It doesn't require any other hardware and you only need an analog port. The difficulty lies in compiling the code and converting the analog values to Celsius temperature. In this project, we used a temperature sensor and 3 LEDs to make a temperature tester. When the temperature sensor touches different temperature objects, the LEDs will show different colors.

# ESP32\*1 Breadboard\*1 LM35\*1 USB Cable\*1 M-F Dupont Wires Jumper Wires 220 Resistor\*3 Red LED\*1 Yellow LED\*1 Green LED\*1 Green LED\*1

## 9.28.2 Components

## 9.28.3 Component knowledge



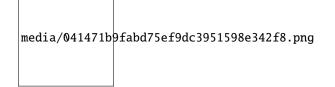
**Working principle of LM35 temperature sensor:** LM35 temperature sensor is a widely used temperature sensor with a variety of package types. At room temperature, it can achieve the accuracy of 1/4°C without additional calibration processing. LM35 temperature sensor can produce different voltage according to different temperatures, when the

temperature is 0 °C, it output 0V; If increasing 1 °C, the output voltage will increase 10mv. The output temperature is  $0^{\circ}$ C to 100°C, the conversion formula is as follows

```
media/0dfa07fa69f2a98658a3822c2da93bf7.jpeg
```

#### 9.28.4 Read the temperature value of LM35

We first use a simple code to read the value of the temperature sensor and printing them out, wiring diagram is shown below



LM35 output is given to analog pin GPIO36 of the ESP32, this analog voltage is converted to its digital form and processed to get the temperature reading.

You can open the code we provide: If you haven't downloaded the code file, please click on the link to download itDownload Arduino C Codes file

The code used in this project is saved in folder "Arduino-Codes\Project 27Temperature Measurement\Project\_27.1\_Read\_LM35\_Temperature\_Value.

```
/*
* Filename : Read LM35 Temperature Value
* Description : ADC value is converted to LM35 temperature value
* Auther : http//www.keyestudio.com
*/
#define PIN_ANALOG_IN 36 //the pin of the Temperature Sensor
void setup() {
 Serial.begin(115200);
}
//In loop() the analogRead() function is used to obtain the ADC value, and then the map()
→ function is used to convert the value into an 8-bit precision DAC value. Calculate the
\rightarrow measured voltage value. Celsius and Fahrenheit values through the formula, and print
→these data through the serial port monitor.
void loop() {
 int adcVal = analogRead(PIN_ANALOG_IN);
 int dacVal = map(adcVal, 0, 4095, 0, 255);
 double voltage = adcVal / 4095.0 * 3.3;
 float temperatureC = (voltage * 1000.0) / 10.0 ;
 float temperatureF = (temperatureC * 1.8) + 32.0;
 Serial.print("ADC Value: " + String(adcVal));
 Serial.print("--DAC Value: " + String(dacVal));
```

(continues on next page)

(continued from previous page)

## 9.28.5 Project result

Compile and upload the code to ESP32, after the code is uploaded successfully, power up with a USB cable, open the serial monitor and set the baud rate to 115200. You need to press the reset button on the ESP32 mainboard first, and then you will see that the serial port monitor window will print out the temperature values read by the LM35 temperature sensor. Hold the LM35 element by hand, the temperature value read by the LM35 temperature sensor will change.

					/d	ev/ttyUSB0		~	~ >	×
									Sen	d
ADC ADC ADC ADC ADC ADC ADC ADC ADC ADC	Value: Value: Value: Value: Value: Value: Value: Value: Value: Value: Value:	240 DAC 240 DAC 240 DAC 240 DAC 242 DAC 233 DAC 240 DAC 240 DAC 240 DAC 240 DAC 240 DAC 240 DAC	Value: Value: Value: Value: Value: Value: Value: Value: Value: Value: Value:	15Voltage 15Voltage 15Voltage 15Voltage 15Voltage 15Voltage 15Voltage 15Voltage 15Voltage 15Voltage 15Voltage 15Voltage	Value: Value: Value: Value: Value: Value: Value: Value: Value: Value: Value:	0.19VtemperatureC: 0.19VtemperatureC: 0.19VtemperatureC: 0.20VtemperatureC: 0.19VtemperatureC: 0.19VtemperatureC: 0.19VtemperatureC: 0.19VtemperatureC: 0.19VtemperatureC: 0.19VtemperatureC: 0.19VtemperatureC: 0.19VtemperatureC:	19.26℃temperatureF: 19.34℃temperatureF: 19.34℃temperatureF: 19.34℃temperatureF: 19.50℃temperatureF: 19.34℃temperatureF: 19.34℃temperatureF: 19.34℃temperatureF: 19.34℃temperatureF: 19.34℃temperatureF: 19.34℃temperatureF: 19.34℃temperatureF: 19.34℃temperatureF: 19.34℃temperatureF: 19.34℃temperatureF: 19.34℃temperatureF:	66.8 66.8 66.8 66.8 66.8 66.8 66.8 66.8	31F 31F 31F 30F 31F 31F 31F 31F 31F 31F 31F	
•						N			•	•
<b>~</b>	Autoscro	II Show t	timestam	p		Newline	<ul> <li>▼ 115200 baud</li> <li>▼ C</li> </ul>	lear o	utpu	t

## 9.28.6 Wiring diagram

Now we use a LM35 temperature sensor and three LED lights to do a temperature test. When the LM35 temperature sensor senses different temperatures, different LED lights will light up. Follow the diagram below for wiring.

media/831ebf50a9b59c744cbd93ac2170d64b.png

#### 9.28.7 Project code

You can open the code we provide: If you haven't downloaded the code file, please click on the link to download itDownload Arduino C Codes file

The code used in this project is saved in folder "Arduino-Codes\Project 27Temperature Measurement\Project\_27.2\_Temperature\_Measurement".

(Note: The temperatureC threshold in the code can be reset itself as required.)

```
/*
* Filename
             : Temperature Measurement
* Description : Different leds light up when the LM35 senses different temperatures
* Auther
            : http//www.keyestudio.com
*/
#define PIN_ADC0
                      36
                            //the pin of the LM35 Sensor
#define PIN_GREENLED 4
                            //the pin of the Green led
#define PIN_YELLOWLED 2
                            //the pin of the Yellow led
                   15
#define PIN_REDLED
                            //the pin of the Red led
void setup() {
 Serial.begin(115200);
 pinMode(PIN_GREENLED, OUTPUT); //set PIN_GREENLED to OUTPUT
 pinMode(PIN_YELLOWLED, OUTPUT);//set PIN_YELLOWLED to OUTPUT
 pinMode(PIN_REDLED, OUTPUT);//set PIN_REDLED to OUTPUT
 pinMode(PIN_ADC0, INPUT);//set PIN_ADC0 to INPUT
}
void loop() {
 int adcVal = analogRead(PIN_ADC0);
 double voltage = adcVal / 4095.0 * 3.3;
 float temperatureC = (voltage * 1000.0) / 10.0 ;
 float temperatureF = (temperatureC * 1.8) + 32.0;
 Serial.print("ADC Value: " + String(adcVal));
 Serial.print("---Voltage Value: " + String(voltage) + "V");
 Serial.print("---temperatureC: " + String(temperatureC) + ""C");
 Serial.println("---temperatureF: " + String(temperatureF) + "F");
 if (temperatureC >= 25) {
   delay(100);
   digitalWrite(PIN_GREENLED, LOW);
   digitalWrite(PIN_YELLOWLED, LOW);
   digitalWrite(PIN_REDLED, HIGH);
 }
 else if (temperatureC >= 20 && temperatureC < 25) {</pre>
   digitalWrite(PIN_GREENLED, LOW);
   digitalWrite(PIN_YELLOWLED, HIGH);
   digitalWrite(PIN_REDLED, LOW);
 }
 else {
   digitalWrite(PIN_GREENLED, HIGH);
   digitalWrite(PIN_YELLOWLED, LOW);
   digitalWrite(PIN_REDLED, LOW);
 }
```

(continues on next page)

(continued from previous page)

delay(500);
}
//*************************************

## 9.28.8 Project result

Compile and upload the code to ESP32, after the code is uploaded successfully, power up with a USB cable and you will see that the monitor displays the temperature values read by the LM35 temperature sensor. When the LM35 temperature sensor senses different temperatures, different LEDS will light up.

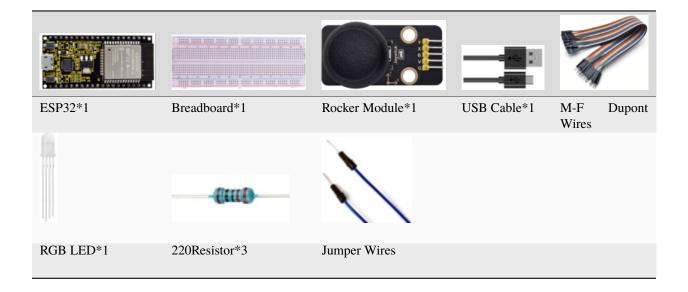
# 9.29 Project 28Rocker control light

## 9.29.1 Introduction

The rocker module is a component with two analog inputs and one digital input. It is widely used in areas such as game operation, robot control and drone control.

In this project, we use ESP32 and a joystick module to control RGB, so that you can have a deeper understanding of the principle and operation of the joystick module in practice.

## 9.29.2 Components



## 9.29.3 Component knowledge



**Rocker module:** It mainly uses PS2 joystick components. In fact, the joystick module has 3 signal terminal pins, which simulate a three-dimensional space. The pins of the joystick module are GND, VCC, and signal terminals (B, X, Y). The signal terminals X and Y simulate the X-axis and Y-axis of the space. When controlling, the X and Y signal terminals of the module are connected to the analog port of the microcontroller. The signal terminal B simulates the Z axis of the space, it is generally connected to the digital port and used as a button.

VCC is connected to the microcontroller power output VCC (3.3V or 5V), GND is connected to the microcontroller GND, the voltage in the original state is about 1.65V or 2.5V. In the X-axis direction, when moving in the direction of the arrow, the voltage value increases, and the maximum voltage can be reached. Moving in the opposite direction of the arrow, the voltage value gradually decreases to the minimum voltage. In the Y-axis direction, the voltage value decreases gradually as it moves in the direction of the arrow on the module, decreasing to the minimum voltage. As the arrow is moved in the opposite direction, the voltage value increases and can reach the maximum voltage. In the Z-axis direction, the signal terminal B is connected to the digital port and outputs 0 in the original state and outputs 1 when pressed. In this way, we can read the two analog values and the high and low level conditions of the digital port to determine the operating status of the joystick on the module.

Features:

- Input VoltageDC 3.3V ~ 5V
- Output SignalX/Y dual axis analog value +Z axis digital signal
- Rang of ApplicationSuitable for control point coordinate movement in plane as well as control of two degrees of freedom steering gear, etc.
- Product FeaturesExquisite appearance, joystick feel superior, simple operation, sensitive response, long service life.

## 9.29.4 Read the value of the Rocker Module

We must use ESP32's analog IO port to read the value from the X/Y pin of the rocker module and use the digital IO port to read the digital signal of the button. Please connect the wires according to the wiring diagram below

media/b611755eacc4c603e6c0555aced929cb.png
--

You can open the code we provide: If you haven't downloaded the code file, please click on the link to download itDownload Arduino C Codes file

The code used in this project is saved in folder\*\*"Arduino-Codes\Project 28Rocker control light\Project\_28.1\_Read\_Rocker\_Value"\*\*.

```
/*
* Filename : Read Rocker Value
* Description : Read data from Rocker.
        : http//www.keyestudio.com
* Auther
*/
int xyzPins[] = {36, 39, 14}; //x,y,z pins
void setup() {
 Serial.begin(115200);
 pinMode(xyzPins[0], INPUT); //x axis.
 pinMode(xyzPins[1], INPUT); //y axis.
 pinMode(xyzPins[2], INPUT_PULLUP); //z axis is a button.
}
// In loop(), use analogRead () to read the value of axes X and Y and use digitalRead ().
\rightarrow to read the value of axis Z, then display them.
void loop() {
 int xVal = analogRead(xyzPins[0]);
 int yVal = analogRead(xyzPins[1]);
 int zVal = digitalRead(xyzPins[2]);
 Serial.println("X,Y,Z: " + String(xVal) + ", " + String(yVal) + ", " + String(zVal));
 delay(500);
}
```

Compile and upload the code to ESP32, after the code is uploaded successfully, power up with a USB cable, open the serial monitor and set the baud rate to 115200. You need to press the reset button on the ESP32 mainboard first, and then you will see that the serial port monitor window will print out the analog and digital values of the current joystick. Moving the joystick or pressing it will change the analog and digital values.

media/06a9de681779df5cfc7e6bc24a928a3a.jpeg



## 9.29.5 Wiring diagram of Rocker control light

We just read the value of the rocker module, we need to do something with the rocker module and RGB here, Follow the diagram below for wiring



## 9.29.6 Project code

You can open the code we provide: If you haven't downloaded the code file, please click on the link to download itDownload Arduino C Codes file

The code used in this project is saved in folder "Arduino-Codes\Project 28Rocker control lightProject\_28.2\_Rocker\_Control\_Light".

```
/*
* Filename : Rocker Control Light
* Description : Control RGB to light different colors by Rocker.
        : http//www.keyestudio.com
* Auther
*/
int x_Pin = 36;
             //x pin
int y_Pin = 39;
            //y pin
int z_Pin = 14; //z pin
int ledPins[] = {4, 0, 2}; //define red, green, blue led pins
const byte chns[] = \{0, 1, 2\};
                         //define the pwm channels
void setup() {
```

(continued from previous page)

```
pinMode(x_Pin, INPUT); //x axis.
 pinMode(y_Pin, INPUT); //y axis.
 pinMode(z_Pin, INPUT_PULLUP);
                               //z axis is a button.
                                //setup the pwm channels,1KHz,8bit
 for (int i = 0; i < 3; i++) {
   ledcSetup(chns[i], 1000, 8);
   ledcAttachPin(ledPins[i], chns[i]);
 }
}
// In loop(), use analogRead () to read the value of axes X and Y and use digitalRead ().
\hookrightarrow to read the value of axis Z, then display them.
void loop() {
 int xVal = analogRead(x_Pin);
 int yVal = analogRead(y_Pin);
 int zVal = digitalRead(z_Pin);
 if (xVal < 1000){
    ledcWrite(chns[0], 255); //Common cathode LED, high level to turn on the led.
    ledcWrite(chns[1], 0);
    ledcWrite(chns[2], 0);
  }
 else if (xVal > 3000){
    ledcWrite(chns[0], 0);
    ledcWrite(chns[1], 255);
    ledcWrite(chns[2], 0);
  }
 else if (yVal < 1000){
    ledcWrite(chns[0], 0);
    ledcWrite(chns[1], 0);
    ledcWrite(chns[2], 255);
  }
 else if (yVal > 3000){
    ledcWrite(chns[0], 255);
    ledcWrite(chns[1], 255);
    ledcWrite(chns[2], 255);
  }
}
```

## 9.29.7 Project result

Compile and upload the code to ESP32, after the code is uploaded successfully, power up with a USB cable and you will see that If the rocker is moved to the far left in the X direction, the RGB light turns red. If the rocker is moved to the far right in the X direction, the RGB light turns green. If the rocker is moved to the up in the Y direction, the RGB light turns white. If the rocker is moved to the down in the Y direction, the RGB light turns blue.

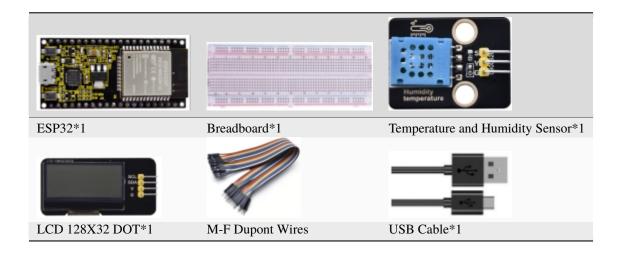
media/9c2d0d8777200827b16c49b752d45c4c.jpeg

# 9.30 Project 29Temperature Humidity Meter

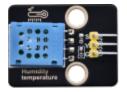
## 9.30.1 Introduction

In winter, the humidity in the air is very low, that is, the air is very dry, Coupled with cold, the skin of the human body is easy to be too dry and cracked, so you need to use a humidifier to increase the humidity of the air at home, but how do you know that the air is too dry? Then you need equipment to detect air humidity. In this Project, we will how to use the temperature and humidity sensor. We use the sensor to make a thermohygrometer, and also combined with a LCD 128X32 DOT to display the temperature and humidity values.

## 9.30.2 Components



## 9.30.3 Component knowledge



**Temperature and humidity sensor:** It is a temperature and humidity composite sensor with calibrated digital signal output, its precision humidity is $\pm$ 5%RH, temperature is $\pm$ 2°C, range humidity is 20 to 90%RH, and temperature is 0 to 50°C. The temperature and humidity sensor applies dedicated digital module acquisition technology and temperature and humidity sensing technology to ensure extremely high reliability and excellent long-term stability of the product. The temperature and humidity sensor includes a resistive-type humidity measurement and an NTC temperature measurement component, which is very suitable for temperature and humidity measurement applications where accuracy and real-time performance are not required.

The operating voltage is in the range of 3.3V to 5.5V.

The temperature and humidity sensor has three pins, which are VCC, GND and S. S is the pin for data output, using serial communication.

Single bus format definition of Temperature and Humidity Sensor

De- scrip- tion	Definition
Start signal	Microprocessor pulls data bus (SDA) down at least 18ms for a period of time(Maximum is 30ms), noti- fying the sensor to prepare data.
Re- sponse signal	The sensor pulls the data bus (SDA) low for $83\mu$ s, and then pulls up for $87\mu$ s to respond to the host's start signal.
Hu- midity	The high humidity is an integer part of the humidity data, and the low humidity is a fractional part of the humidity data.
Tem- pera- ture	The high temperature is the integer part of the temperature data, the low temperature is the fractional part of the temperature data. And the low temperature Bit8 is 1, indicating a negative temperature, otherwise, it is a positive temperature.
Parity bit	Parity bit=Humidity high bit+ Humidity low bit+temperature high bit+temperature low bit

```
/*
* Filename
            : Temperature and Humidity Sensor
* Description : Use XHT11 to measure temperature and humidity.Print the result to the.
→serial port.
* Auther
         : http//www.keyestudio.com
*/
#include "xht11.h"
//gpio13
xht11 xht(13);
unsigned char dht[4] = {0, 0, 0, 0}; //Only the first 32 bits of data are received, not
\rightarrow the parity bits
void setup() {
 Serial.begin(115200);//Start the serial port monitor and set baud rate to 115200
}
void loop() {
 if (xht.receive(dht)) { //Returns true when checked correctly
   Serial.print("RH:");
   Serial.print(dht[0]); //The integral part of humidity, DHT [0] is the fractional part
   Serial.print("% ");
   Serial.print("Temp:");
   Serial.print(dht[2]); //The integral part of temperature, DHT [3] is the fractional.
→part
   Serial.println("C");
 } else {
           //Read error
   Serial.println("sensor error");
 }
 delay(1000); //It takes 1000ms to wait for the device to read
}
```

Data sequence diagram of Temperature and Humidity Sensor

When MCU sends a start signal, the Temperature and Humidity Sensor changes from the low-power-consumption

mode to the high-speed mode, waiting for MCU completing the start signal. Once it is completed, the Temperature and Humidity Sensor sends a response signal of 40-bit data and triggers a signal acquisition. The signal is sent as shown

media/933ac5e5a5e921d4b16c7c48091ba75a.png

in the figure:

Combined with the code, you can understand better.

The XHT11 temperature and humidity sensor can easily add temperature and humidity data to your DIY electronic projects. It is perfect for remote weather stations, home environmental control systems, and farm or garden monitoring systems.

Specification:

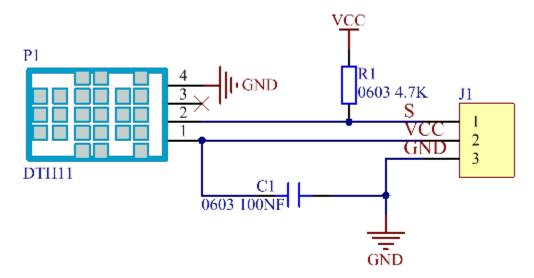
Working voltage: +5V

Temperature range:  $0^{\circ}$ C to  $50^{\circ}$ C, error of  $\pm 2^{\circ}$ C

Humidity range: 20% to 90% RH, $\pm$  5% RH error

Digital interface

Schematic diagram of Temperature and Humidity Sensor:



#### 9.30.4 Read temperature and humidity value



How to add the xht11 library

If you have not downloaded the library file, please click on the link to download it:Download Arduino Libraries

This code uses a library named "**xht11**", if you haven't installed it yet, please do so before learning. The steps to add third-party libraries are as follows:

After the **xht11** library was added, You can open the code we provideIf you haven't downloaded the code file, please click on the link to download itDownload Arduino C Codes file

The code used in this project is saved in folder "Arduino-Codes\Project 29Temperature Humidity Meter\Project\_29.1\_Detect\_Temperature\_Humidity".

```
/*
* Filename
             : Temperature and Humidity Sensor
* Description : Use XHT11 to measure temperature and humidity.Print the result to the.
\leftrightarrow serial port.
* Auther
             : http//www.keyestudio.com
*/
#include "xht11.h"
//apio13
xht11 xht(13);
unsigned char dht[4] = {0, 0, 0, 0}; //Only the first 32 bits of data are received, not
\rightarrow the parity bits
void setup() {
 Serial.begin(115200);//Start the serial port monitor and set baud rate to 115200
}
void loop() {
 if (xht.receive(dht)) { //Returns true when checked correctly
   Serial.print("RH:");
   Serial.print(dht[0]); //The integral part of humidity, DHT [0] is the fractional part
   Serial.print("% ");
   Serial.print("Temp:");
   Serial.print(dht[2]); //The integral part of temperature, DHT [3] is the fractional.
\rightarrow part
   Serial.println("C");
 } else { //Read error
   Serial.println("sensor error");
 }
 delay(1000); //It takes 1000ms to wait for the device to read
}
```

Compile and upload the code to ESP32, after the code is uploaded successfully, power up with a USB cable, and open the serial monitor and then set baud rate to 115200. You need to press the reset button on the ESP32 mainboard first, and then you will see the current temperature and humidity value detected by the sensor from the serial monitor. As shown in the following figure:

		/dev/t	tyUSB0			~ ^ X
						Send
RH:66%	Temp:31C					-
RH:67%	Temp:31C					
RH:67%	Temp:31C					
RH:67%	Temp:31C					
RH:67%	Temp:31C					
RH:66%	Temp:31C					
RH:66%	Temp:31C					
RH:65%	Temp:31C					
RH:66%	Temp:31C					
RH:72%	Temp:31C					
RH:77%	Temp:31C					
RH:79%	Temp:31C					
RH:82%	Temp:31C					
RH:85%	Temp:31C					
RH:82%	Temp:31C					
Auto:	scroll 🗌 Show timestamp		Newline	• 115	200 baud 🔹	Clear output
U Auto	onon aneoump		. termine	110	200 0000	ordar output

#### 5. Wiring diagram of the thermohygrometer

Now we start to print the values of the temperature and humidity sensor with LCD\_128X32\_DOT. We will see the corresponding values on the screen of LCD\_128X32\_DOT. Let's get started with this project. Please connect cables according to the following wiring diagram

media/6c82bb28bd1fcd7a1f72108e8a4a70b6.png

#### 9.30.5 Project code

If you have not downloaded the library file, please click on the link to download it:Download Arduino Libraries

The **xht11** and **lcd128\_32\_io** libraries have been added previously, so you don't need to add them again. If not, you need to add the **xht11** and **lcd128\_32\_io** libraries. The steps to add third-party Libraries are as follows:

After the **xht11** and **lcd128\_32\_io** libraries were added, You can open the code we provideIf you haven't downloaded the code file, please click on the link to download itDownload Arduino C Codes file

The code used in this project is saved in folder "Arduino-Codes\Project 29Temperature Humidity Meter\Project\_29.2\_Temperature\_Humidity\_Meter".

```
//*
* Filename : Temperature Humidity Meter
* Description : LCD displays the value of temperature and humidity.
* Auther : http//www.keyestudio.com
*/
#include "xht11.h"
#include "lcd128_32_io.h"
```

```
//gpio13
xht11 xht(13);
unsigned char dht[4] = {0, 0, 0, 0}; //Only the first 32 bits of data are received, not
\rightarrow the parity bits
lcd lcd(21, 22); //Create lCD128 *32 pinsda->21 scl->22
void setup() {
 lcd.Init(); //initialize
 lcd.Clear(); //clear
}
char string[10];
//lcd displays humidity and temperature values
void loop() {
 if (xht.receive(dht)) { //Returns true when checked correctly
   }
 lcd.Cursor(0,0); //Set display position
 lcd.Display("Temper:"); //Setting the display
 lcd.Cursor(0,8);
 lcd.DisplayNum(dht[2]);
 lcd.Cursor(0,11);
 lcd.Display("C");
 lcd.Cursor(2, ◊);
 lcd.Display("humid:");
 lcd.Cursor(2,8);
 lcd.DisplayNum(dht[0]);
 lcd.Cursor(2,11);
 lcd.Display("%");
 delay(200);
}
```

# 9.30.6 Project result

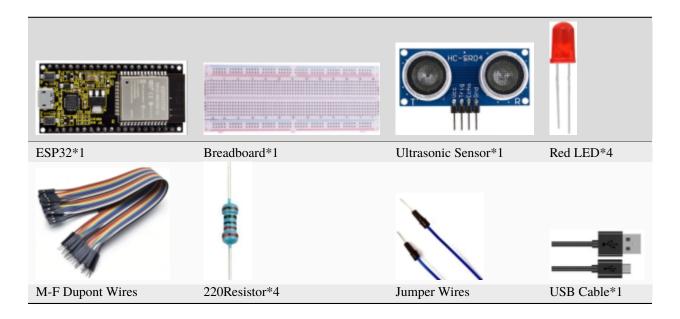
Compile and upload the code to ESP32, after the code is uploaded successfully, power up with a USB cable and you will see that the LCD 128X32 DOT will display temperature and humidity value in the current environment.

# 9.31 Project 30Ultrasonic Ranger

# 9.31.1 Introduction

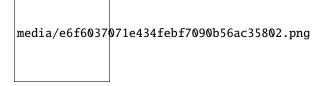
The HC-SR04 ultrasonic sensor is a very affordable distance sensor, mainly used for obstacle avoidance in various robotic projects. It is also used for water level sensing and even as a parking sensor. We treat the ultrasonic sensors as bat's eyes, in the dark, bats can still identify objects in front of them and directions through ultrasound. In this project, we use ESP32 to control a ultrasonic sensor and LEDs to simulate ultrasonic rangefinder.

# 9.31.2 Components



# 9.31.3 Component knowledge

**HC-SR04 Ultrasonic Sensor :** Like bats, sonar is used to determine the distance to an object. It provides accurate noncontact range detection, high-precision and stable readings. Its operation is not affected by sunlight or black materials, just like a precision camera(acoustically softer materials like cloth are difficult to detect). It has an ultrasonic transmitter and receiver.



In front of the ultrasonic sensor are two metal cylinders, these are the converters. The converters convert the mechanical energy into an electrical signal. In the ultrasonic sensor, there are transmitting converters and receiving converters. The transmitting converter converts the electric signal into an ultrasonic pulse, and the receiving converter converts the reflected ultrasonic pulse back to an electric signal. If you look at the back of the ultrasonic sensor, you will see an IC behind the transmitting converter, which controls the transmitting converter. There is also an IC behind the receiving converter into a signal large enough to be transmitted to the Microcontroller.

Sequence diagrams:

The figure shows the sequence diagram of the HC-SR04. To start the measurement, the Trig of SR04 must receive at least 10us high pulse(5V), which will activate the sensor to emit 8 cycles of 40kHz ultrasonic pulses, and wait for the reflected ultrasonic pulses. When the sensor detects ultrasound from the receiver, it sets the Echo pin to high (5V) and delays it by one cycle (width), proportional to the distance. To get the distance, measure the width of the Echo pin.

media/4114885ac4b6214953e3224d8c1d52c4.png

Time = Echo pulse width, its unit is "us" (microseconds) Distance in centimeters = time / 58

Distance in inches = time / 148

# 9.31.4 Read the distance value of the ultrasonic sensor:

We will start with a simple ultrasonic ranging and print the measured distance.



The HC-SR04 ultrasonic sensor has four pins, they are Vcc, Trig, Echo and GND. The Vcc pin provides the power source for generating ultrasonic pulses and is connected to Vcc (+5V). The GND pin is grounded. The Trig pin is where the Arduino sends a signal to start the ultrasonic pulse. The Echo pin is where the ultrasonic sensor sends information about the duration of the ultrasonic pulse to the control board, as shown below:

media/a8d408be3629a2d288dbb30bd60007af.png

You can open the code we provide: If you haven't downloaded the code file, please click on the link to download itDownload Arduino C Codes file

The code used in this project is saved in folder "Arduino-Codes\Project 30Ultrasonic Ranger\Project 30.1\_Ultrasonic\_Ranging".

```
Serial.begin(115200); // Open serial monitor at 115200 baud to see ping results.
}
void loop()
{
// make trigPin output high level lasting for 10s to triger HC_SR04
 digitalWrite(TrigPin , HIGH);
 delayMicroseconds(10);
 digitalWrite(TrigPin , LOW);
 // Wait HC-SR04 returning to the high level and measure out this waitting time
 duration = pulseIn(EchoPin , HIGH);
 // calculate the distance according to the time
 distance = (duration/2) / 28.5;
 Serial.print("Distance: ");
 Serial.print(distance); //Serial port print distance value
 Serial.println("cm");
 delay(300); // Wait 100ms between pings (about 20 pings/sec).
}
```

Compile and upload the code to ESP32, after the code is uploaded successfully, power up with a USB cable, open the serial monitor and set the baud rate to 115200. You need to press the reset button on the ESP32 mainboard first, and then you will see that the serial port monitor window will print out the distance between the ultrasonic sensor and the object.

	/dev/ttyUSB0				~ ^ X
					Send
Distance: 3cm					
Distance: 2cm					
Distance: 2cm					
Distance: 2cm					
Distance: 2cm					
Distance: 2cm					
Distance: 2cm					
Distance: 3cm					
Distance: 3cm					
Distance: 4cm					
Distance: 5cm					
Distance: 6cm					
Distance: 6cm					
Distance: 7cm					
Distance: 7cm					ļ
✓ Autoscroll	Newline	-	115200 baud	•	Clear output

#### 9.31.5 Wiring diagram of the ultrasonic rangefinder

Next, we will use ESP32 to control an ultrasonic sensor and 4 LEDs to simulate ultrasonic rangefinder. Connect the line as shown below

```
media/910ed1be8be94411a090afb95af86d1a.png
```

#### 9.31.6 Project code

You can open the code we provide: If you haven't downloaded the code file, please click on the link to download itDownload Arduino C Codes file

The code used in this project is saved in folder "Arduino-Codes\Project 30Ultrasonic Ranger\Project\_30.2\_Ultrasonic\_Ranger".

```
//********
/*
* Filename : Ultrasonic Ranger
* Description : four leds are controlled by ultrasonic ranging.
            : http//www.keyestudio.com
* Auther
*/
const int TrigPin = 13; // define TrigPin
const int EchoPin = 14; // define EchoPin.
const int PIN_LED1 = 4;
                        // define PIN_LED1
                      // define PIN_LED2
const int PIN_LED2 = 0;
                      // define PIN_LED3
const int PIN_LED3 = 2;
const int PIN_LED4 = 15;
                       // define PIN_LED4
int duration = 0;  // define the initial value of the duration to be 0
int distance = 0; // define the initial value of the distance to be 0
void setup()
{
 pinMode(TrigPin , OUTPUT); // set trigPin to output mode
 pinMode(EchoPin , INPUT); // set echoPin to input mode
 pinMode(PIN_LED1 , OUTPUT); // set PIN_LED1 to output mode
 pinMode(PIN_LED2 , OUTPUT); // set PIN_LED2 to output mode
 pinMode(PIN_LED3 , OUTPUT); // set PIN_LED3 to output mode
 pinMode(PIN_LED4 , OUTPUT); // set PIN_LED4 to output mode
 Serial.begin(115200); // Open serial monitor at 115200 baud to see ping results.
}
void loop()
{
// make trigPin output high level lasting for 10s to triger HC_SR04
 digitalWrite(TrigPin , HIGH);
 delayMicroseconds(10);
 digitalWrite(TrigPin , LOW);
// Wait HC-SR04 returning to the high level and measure out this waitting time
 duration = pulseIn(EchoPin , HIGH);
// calculate the distance according to the time
```

```
distance = (duration/2) / 28.5;
  Serial.print("Distance: ");
  Serial.print(distance); //Serial port print distance value
  Serial.println("cm");
  if ( distance <= 5 )</pre>
  {
   digitalWrite(PIN_LED1, HIGH);
  }
  else
  {
   digitalWrite(PIN_LED1, LOW);
  }
  if ( distance <= 10 )
  {
   digitalWrite(PIN_LED2, HIGH);
  }
  else
  {
   digitalWrite(PIN_LED2, LOW);
  }
  if ( distance <= 15 )</pre>
  {
   digitalWrite(PIN_LED3, HIGH);
  }
  else
  {
   digitalWrite(PIN_LED3, LOW);
  }
  if ( distance <= 20 )</pre>
  {
   digitalWrite(PIN_LED4, HIGH);
  }
  else
  {
    digitalWrite(PIN_LED4, LOW);
  }
}
                         *************************
```

#### 9.31.7 Project result

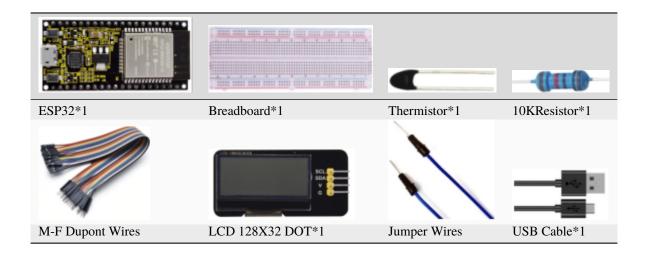
Compile and upload the code to ESP32, after the code is uploaded successfully, power up with a USB cable, open the serial monitor and set the baud rate to 115200. You will see that the serial port monitor window will print outthe distance between the ultrasonic sensor and the object, and the corresponding LED will light up when we move our hand in front of the ultrasonic sensor.

# 9.32 Project 31Temperature Instrument

# 9.32.1 Introduction

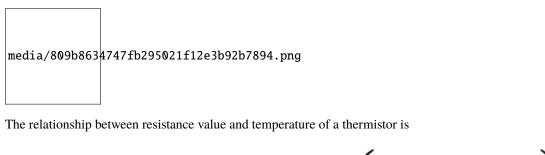
Thermistor is a kind of resistor whose resistance depends on temperature changes, which is widely used in gardening, home alarm system and other devices. Therefore, we can use the feature to make a temperature instrument.

# 9.32.2 Components



## 9.32.3 Component knowledge

**Thermistor:** A Thermistor is a temperature sensitive resistor. When it senses a change in temperature, the resistance of the Thermistor will change. We can take advantage of this characteristic by using a thermistor to detect temperature intensity. A Thermistor and its electronic symbol are shown below:



$$Rt = R * EXP[B * \left(\frac{1}{T2} - \frac{1}{T1}\right)]$$

Where:

Rt is the thermistor resistance under T2 temperature;

**R** is the nominal resistance of thermistor under T1 temperature;

**EXP**[**n**] is nth power of e;

**B** is for thermal index;

**T1, T2** is Kelvin temperature (absolute temperature). Kelvin temperature=273.15 + Celsius temperature.

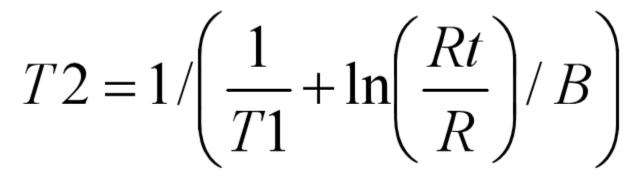
For the parameters of the Thermistor, we use: B=3950, R=10k, T1=25.

The circuit connection method of the Thermistor is similar to photoresistor, as the following

media/b0f80e9bd350a8b7390a73756ac1ac8c.jpeg

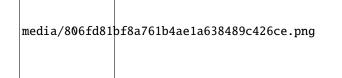
We can use the value measured by the ADC converter to obtain the resistance value of Thermistor, and then we can use the formula to obtain the temperature value.

Therefore, the temperature formula can be derived as:



#### 9.32.4 Read the value of the Thermistor

First we will learn the thermistor to read the current ADC valuevoltage value and temperature value and print them out. Please connect the wires according to the wiring diagram below



You can open the code we provide: If you haven't downloaded the code file, please click on the link to download itDownload Arduino C Codes file

The code used in this project is saved in folder "Arduino-Codes\Project 31Temperature Instrument\Project\_31.1\_Read\_the\_thermistor\_analog\_value".

\* Auther

void loop() {

double tempC = tempK - 273.15;

\*/

}

```
(continued from previous page)
 * Description : Making a thermometer by thermistor.
               : http//www.keyestudio.com
#define PIN_ANALOG_IN
                         36
void setup() {
  Serial.begin(115200);
  int adcValue = analogRead(PIN_ANALOG_IN);
                                                                    //read ADC pin
  double voltage = (float)adcValue / 4095.0 * 3.3;
                                                                    // calculate voltage
  double Rt = 10 * voltage / (3.3 - voltage);
                                                                    //calculate resistance...
```

```
\rightarrow value of thermistor
 double tempK = 1 / (1 / (273.15 + 25) + log(Rt / 10) / 3950.0); //calculate_
→temperature (Kelvin)
```

→temperature (Celsius) Serial.printf("ADC value : %d,\tVoltage : %.2fV, \tTemperature : %.2fC\n", adcValue,\_ →voltage, tempC); delay(1000); } 

Compile and upload the code to ESP32, after the code is uploaded successfully, power up with a USB cable, open the serial monitor and set the baud rate to 115200. You need to press the reset button on the ESP32 mainboard first, and then you will see that the serial port monitor window prints out the thermistor's current ADC valuevoltage value and temperature value. Try pinching the thermistor with your index finger and thumb (don't touch wires) for a while, and you will see the temperature increase.

	/de	ev/ttyUSB0	~ ^ X
			Send
ADC value : 1841, ADC value : 1834, ADC value : 1834, ADC value : 1861, ADC value : 1835, ADC value : 1802, ADC value : 1802, ADC value : 1834, ADC value : 1843, ADC value : 1843,	Voltage : 1.67V, Voltage : 1.48V, Voltage : 1.48V, Voltage : 1.50V, Voltage : 1.48V, Voltage : 1.45V, Voltage : 1.45V, Voltage : 1.43V, Voltage : 1.44V, Voltage : 1.44V, Voltage : 1.46V, Voltage : 1.49V, Voltage : 1.51V, Voltage : 1.74V,	Temperature : 29.63C Temperature : 29.79C Temperature : 29.17C Temperature : 29.76C Temperature : 30.52C Temperature : 31.13C Temperature : 29.79C Temperature : 31.85C Temperature : 30.38C Temperature : 29.58C	
Autoscroll 🗌 Sho	w timestamp	Newline   115200 baud	Clear output

//calculate

## 9.32.5 Wiring diagram of the temperature instrument



# 9.32.6 Adding the lcd128\_32\_io library

If you have not downloaded the library file, please click on the link to download it:Download Arduino Libraries

The **lcd128\_32\_io** library had been added previously, so you don't need to add it again. If not, you need to add the lcd128\_32\_io library. The steps to add third-party libraries are as follows:

# 9.32.7 Project code

After the **lcd128\_32\_io** library was added, You can open the code we provideIf you haven't downloaded the code file, please click on the link to download itDownload Arduino C Codes file

The code used in this project is saved in folder "Arduino-Codes\Project 31Temperature Instrument\Project\_31.2\_Temperature\_Instrument".

```
/*
* Filename : Temperature Instrument
* Description : LCD displays the temperature of thermistor.
          : http//www.keyestudio.com
* Auther
*/
#include "lcd128_32_io.h"
#define PIN ANALOG IN 36
lcd lcd(21, 22); //Create lCD128 *32 pinsda->21 scl->22
void setup() {
 lcd.Init(); //initialize
 lcd.Clear(); //clear
}
char string[10];
void loop() {
 int adcValue = analogRead(PIN_ANALOG_IN);
                                                           //read ADC pin
 double voltage = (float)adcValue / 4095.0 * 3.3;
                                                           // calculate voltage
 double Rt = 10 * voltage / (3.3 - voltage);
                                                            //calculate resistance...
\rightarrow value of thermistor
 double tempK = 1 / (1 / (273.15 + 25) + log(Rt / 10) / 3950.0); //calculate_
→temperature (Kelvin)
                                                            //calculate
 double tempC = tempK - 273.15;
→temperature (Celsius)
 lcd.Cursor(0,0); //Set display position
```

```
lcd.Display("Voltage:"); //Setting the display
lcd.Cursor(0,8);
lcd.DisplayNum(voltage);
lcd.Cursor(0,11);
lcd.Display("V");
lcd.Cursor(2,0);
lcd.Cursor(2,0);
lcd.Display("tempC:");
lcd.Cursor(2,8);
lcd.DisplayNum(tempC);
lcd.Cursor(2,11);
lcd.Display("C");
delay(200);
}
```

#### 9.32.8 Project result

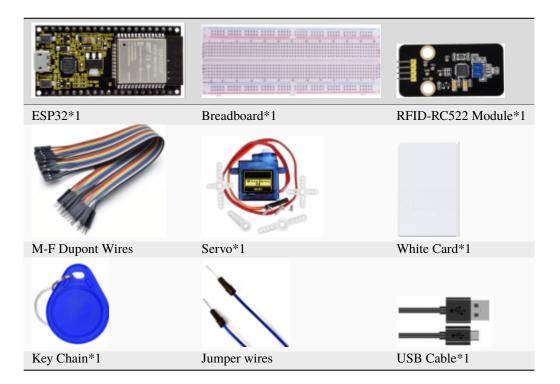
Compile and upload the code to ESP32, after the code is uploaded successfully, power up with a USB cable and you will see that the LCD 128X32 DOT displays the voltage value of the thermistor and the temperature value in the current environment.

# 9.33 Project 32RFID

## 9.33.1 Introduction

Nowadays, many residential districts use this function to open the door by swiping the card, which is very convenient. In this Project, we will learn how to use RFID(radio frequency identification) wireless communication technology and read and write the key chain card (white card) and control the steering gear rotation by RFID-MFRC522 module.

## 9.33.2 Components



## 9.33.3 Component knowledge

\*\*RFID \*\*RFID (Radio Frequency Identification) is a wireless communication technology. A complete RFID system is generally composed of the responder and reader. Generally, we use tags as responders, and each tag has a unique code, which is attached to the object to identify the target object. The reader is a device for reading (or writing) tag information.

Products derived from RFID technology can be divided into three categories: passive RFID productsactive RFID products and semi active RFID products. And Passive RFID products are the earliest, the most mature and most widely used products in the market among others. It can be seen everywhere in our daily life such as, the bus card, dining card, bank card, hotel access cards, etc., and all of these belong to close-range contact recognition. The main operating frequency of Passive RFID products are: 125KHZ (low frequency), 13.56MHZ (high frequency), 433MHZ (ultrahigh frequency) and 915MHZ (ultrahigh frequency). Active and semi active RFID products work at higher frequencies.

The RFID module we use is a passive RFID product with the operating frequency of 13.56MHz.

\*\*RFID-RC522 Module\*\*The MFRC522 is a highly integrated reader/writer IC for contactless communication at 13.56MHz. The MFRC522's internal transmitter is able to drive a reader/writer antenna designed to communicate with ISO/IEC 14443 A/MIFARE cards and transponders without additional active circuitry. The receiver module provides a robust and efficient implementation for demodulating and decoding signals from ISO/IEC 14443 A/MIFARE compatible cards and transponders. The digital module manages the complete ISO/IEC 14443A framing and error detection (parity and CRC) functionality.

This RFID Module uses MFRC522 as the control chip and adopts I2C (Inter-Integrated Circuit) interface.

media/5a19d0dd224c2cdc78871f11e8951045.png

Specifications:

- Operating voltage: DC 3.3V-5V
- Operating current: 13—100mA/DC 5V
- Idling current: 10-13mA/DC 5V
- Sleep current: <80uA
- Peak current: <100mA
- Operating frequency: 13.56MHz
- Maximum power: 0.5W
- Supported card types: mifare1 S50mifare1 S70, mifare UltraLight, mifare Pro, mifare Desfire.
- Environmental operating temperature: -20 to 80 degrees Celsius.
- Environment storage temperature: -40 to 85 degrees Celsius.
- Relative Humidity: 5% to 95%.
- Data transfer rate: The maximum is 10Mbit/s.

## 9.33.4 RFID Read UID

We will read the UNIQUE ID number (UID) of the RFID card and identify the type of the RFID card, and display the relevant information through the serial port. The wiring diagram is shown below

```
media/1cdb3ffd7f392f29451aeed5c3257133.png
```

Adding the MFRC522\_I2C and Wire libraries

If you have not downloaded the library file, please click on the link to download it:Download Arduino Libraries

If you haven't installed the MFRC522\_I2C and Wire libraries yet, please do so before learning. The steps to add third-party libraries are as follows:

After the **MFRC522\_I2C** and **Wire** libraries were added, You can open the code we provideIf you haven't downloaded the code file, please click on the link to download itDownload Arduino C Codes file

The code used in this project is saved in folder "Arduino-Codes\Project 32RFID\Project\_32.1\_RFID\_Read\_UID".

```
* Auther
               : http//www.keyestudio.com
*/
#include <Wire.h>
#include "MFRC522_I2C.h"
// IIC pins default to GPI021 and GPI022 of ESP32
// 0x28 is the i2c address of SDA, if doesn't matchplease check your address with i2c.
MFRC522 mfrc522(0x28); // create MFRC522.
void setup() {
                                // initialize and PC's serial communication
  Serial.begin(115200);
                                // initialize I2C
  Wire.begin();
 mfrc522.PCD_Init();
                                 // initialize MFRC522
  ShowReaderDetails();
                                 // dispaly PCD - MFRC522 read carder
  Serial.println(F("Scan PICC to see UID, type, and data blocks..."));
}
void loop() {
  11
 if ( ! mfrc522.PICC_IsNewCardPresent() || ! mfrc522.PICC_ReadCardSerial() ) {
   delay(50);
   return;
  }
 // select one of door cards. UID and SAK are mfrc522.uid.
  // save UID
  Serial.print(F("Card UID:"));
  for (byte i = 0; i < mfrc522.uid.size; i++) {
   Serial.print(mfrc522.uid.uidByte[i] < 0x10 ? " 0" : " ");</pre>
   Serial.print(mfrc522.uid.uidByte[i], HEX);
  }
 Serial.println();
}
void ShowReaderDetails() {
  // attain the MFRC522 software
  byte v = mfrc522.PCD_ReadRegister(mfrc522.VersionReg);
  Serial.print(F("MFRC522 Software Version: 0x"));
  Serial.print(v, HEX);
  if (v == 0x91)
   Serial.print(F(" = v1.0"));
  else if (v == 0x92)
   Serial.print(F(" = v2.0"));
  else
    Serial.print(F(" (unknown)"));
  Serial.println("");
  // when returning to 0x00 or 0xFF, may fail to transmit communication signals
 if ((v == 0x00) || (v == 0xFF)) {
   Serial.println(F("WARNING: Communication failure, is the MFRC522 properly connected?

→"));

 }
}
```

Compile and upload the code to ESP32, after the code is uploaded successfully, power up with a USB cable, and open the serial monitor and then set baud rate to 115200. You need to press the reset button on the ESP32 mainboard first, and then you will see that place the door card and key chain close to the module sensor area respectively, the serial monitor display the card number and key chain value respectively, as shown below:

								/d	lev/ttyl	USB0							~	^ X	
							 											Send	
Card	UID:	4C	09	6B	6E														
Card	UID:	4C	Θ9	6B	6E														
Card	UID:	4C	Θ9	6B	6E														
Card	UID:	4C	Θ9	6B	6E														
Card	UID:	4C	Θ9	6B	6E														
Card	UID:	4C	Θ9	6B	6E														
Card	UID:	4C	09	6B	6E														
Card	UID:	4C	Θ9	6B	6E														
Card	UID:	ED	F7	94	5A														
Card	UID:	ED	F7	94	5A														
Card	UID:	ED	F7	94	5A														
Card	UID:	ED	F7	94	5A														
Card	UID:	ED	F7	94	5A														L
Card	UID:	ED	F7	94	5A														L
Card	UID:	ED	F7	94	5A														L
																			,
✓ A	utoscr	oll	S	now	times	tamp				Newli	ne	•	11:	5200 b	aud	•	Clear	output	

Note: the door card value and key chain value may be different for different RRFID -RC522 door cards and key chains.

# 9.33.5 Wiring diagram of the RFID MFRC522

Now we use the RFID -RC522 modulewhite card/key chain and Servo to simulate an intelligent access control system. When the white card/key chain close to the RFID -RC522 module induction area, the servo rotates. Wiring according to the figure below

media/3ae6c0f1098d2aee34c51e7a96c25571.png

# 9.33.6 Adding the MFRC522\_I2C, Wire and ESP32Servo libraries

If you have not downloaded the library file, please click on the link to download it:Download Arduino Libraries

The MFRC522\_I2C, Wire and ESP32Servo libraries had been added previously, so you don't need to add it again. If not, you need to add the MFRC522\_I2C, Wire and ESP32Servo libraries. The steps to add third-party Libraries are as follows:

## 9.33.7 Project code

After the **MFRC522\_I2C**, **Wire** and **ESP32Servo** libraries were added, You can open the code we provideIf you haven't downloaded the code file, please click on the link to download itDownload Arduino C Codes file

The code used in this project is saved in folder "Arduino-Codes\Project 32RFID\Project\_32.2\_RFID\_Control\_Servo".

**Note:** Different RFID-MFRC522 IC cards and keys have diverse values. You can substitute your own IC cards and keys values for the corresponding values read by the RFID-MFRC522 module in the program, otherwise the servo can't be controlled when uploading the test code to the ESP32.

media/85170ea6c47d9350dd05efa60be8c808.png

For example: You can replace the rfid\_str of the \_\_\_\_\_\_ in the program code with your own IC cards and keys values read by the RFID-MFRC522 module.

```
/*
* Filename : RFID mfrc522 Control Servo
* Description : RFID controlled steering gear simulated door opening
* Auther : http//www.keyestudio.com
*/
#include <Wire.h>
#include "MFRC522_I2C.h"
// IIC pins default to GPI021 and GPI022 of ESP32
// 0x28 is the i2c address of SDA, if doesn't matchplease check your address with i2c.
MFRC522 mfrc522(0x28); // create MFRC522.
#include <ESP32Servo.h>
Servo myservo; // create servo object to control a servo
int servoPin = 15; // Servo motor pin
String rfid_str = "";
void setup() {
 Serial.begin(115200);
 Wire.begin();
 mfrc522.PCD_Init();
 ShowReaderDetails();
                              // dispaly PCD - MFRC522 read carder
 Serial.println(F("Scan PICC to see UID, type, and data blocks..."));
 myservo.setPeriodHertz(50);
                                   // standard 50 hz servo
 myservo.attach(servoPin, 500, 2500); // attaches the servo on servoPin to the servo
→object
 myservo.write(0);
 delay(500);
}
void loop() {
  if ( ! mfrc522.PICC_IsNewCardPresent() || ! mfrc522.PICC_ReadCardSerial() ) {
   delay(50);
```

```
return;
 }
 // select one of door cards. UID and SAK are mfrc522.uid.
 // save UID
 rfid_str = ""; //String emptying
 Serial.print(F("Card UID:"));
 for (byte i = 0; i < mfrc522.uid.size; i++) {</pre>
   rfid_str = rfid_str + String(mfrc522.uid.uidByte[i], HEX); //Convert to string
   //Serial.print(mfrc522.uid.uidByte[i] < 0x10 ? " 0" : " ");</pre>
   //Serial.print(mfrc522.uid.uidByte[i], HEX);
 }
 Serial.println(rfid_str);
 if (rfid_str == "93adf720" || rfid_str == "39b646c2") {
   myservo.write(180);
   delay(500);
   Serial.println(" open the door!");
   }
}
void ShowReaderDetails() {
 // attain the MFRC522 software
 byte v = mfrc522.PCD_ReadRegister(mfrc522.VersionReg);
 Serial.print(F("MFRC522 Software Version: 0x"));
 Serial.print(v, HEX);
 if (v == 0x91)
   Serial.print(F(" = v1.0"));
 else if (v == 0x92)
   Serial.print(F(" = v2.0"));
 else
   Serial.print(F(" (unknown)"));
 Serial.println("");
 // when returning to 0x00 or 0xFF, may fail to transmit communication signals
 if ((v == 0x00) || (v == 0xFF)) {
   Serial.println(F("WARNING: Communication failure, is the MFRC522 properly connected?
'));
 }
}
```

# 9.33.8 Project result

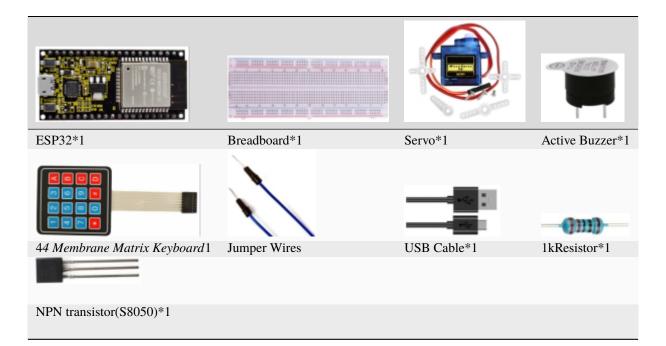
Compile and upload the code to ESP32, after the code is uploaded successfully, power up with a USB cable, and open the serial monitor and then set baud rate to 115200. You need to press the reset button on the ESP32 mainboard first, and then you will see that when using the white card or a key card swiping, the serial port monitor displays white card or key card information and "open the door". As shown in the picture below, and the servo rotates to the corresponding angle to simulate opening the door.

# 9.34 Project 33Keypad Door

# 9.34.1 Introduction

Commonly used digital button sensor, one button uses an IO port. However, it will occupy too many IO ports when we need a lot of buttons. In order to save the use of IO ports, the multiple buttons are made into a matrix type, through the control of the line and row to achieve less IO port control of multiple buttons. In this project, we will learn ESP32 and thin film 4\*4 matrix keyboard control a servo and a buzzer.

# 9.34.2 Components

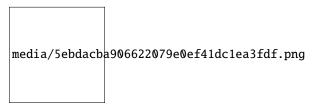


#### 9.34.3 Component knowledge

\*\*4\*4 Matrix keyboard\*\*A Keypad Matrix is a device that integrates a number of keys in one package. As is shown below, a 4x4 Keypad Matrix integrates 16 keys:



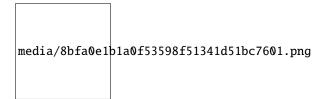
Similar to the integration of an LED Matrix, the 4x4 Keypad Matrix has each row of keys connected with one pin and this is the same for the columns. Such efficient connections reduce the number of processor ports required. The internal circuit of the Keypad Matrix is shown below.



The method of usage is similar to the Matrix LED, by using a row or column scanning method to detect the state of each key's position by column and row. Take column scanning method as an example, send low level to the first 4 column (Pin4), detect level state of row 1234 to judge whether the key A, B, C, D are pressed. Then send low level to column321 in turn to detect whether other keys are pressed. By this means, you can get the state of all of the keys.

#### 9.34.4 Read the key value of the 4\*4 matrix keyboard

We start with a simple code to read the values of the 4\*4 matrix keyboard and print them in the serial monitor. Its wiring diagram is shown below



How to add the Keypad library

If you have not downloaded the library file, please click on the link to download it:Download Arduino Libraries

This code uses a library named "**Keypad**", if you haven't installed it yet, please do so before learning. The steps to add third-party libraries are as follows:

After the **Keypad** library is added, You can open the code we provideIf you haven't downloaded the code file, please click on the link to download itDownload Arduino C Codes file

The code used in this project is saved in folder "Arduino-Codes\Project 33:Key-pad\DoorProject\_33.1\_4x4\_Matrix\_Keypad\_Display".

```
*/
#include <Keypad.h>
// define the symbols on the buttons of the keypad
char keys[4][4] = \{
 {'1', '2', '3', 'A'},
 {'4', '5', '6', 'B'},
 {'7', '8', '9', 'C'},
 {'*', '0', '#', 'D'}
};
byte rowPins[4] = {22, 21, 19, 18}; // connect to the row pinouts of the keypad
byte colPins[4] = \{17, 16, 4, 0\}; // connect to the column pinouts of the keypad
// initialize an instance of class NewKeypad
Keypad myKeypad = Keypad(makeKeymap(keys), rowPins, colPins, 4, 4);
void setup() {
 Serial.begin(115200); // Initialize the serial port and set the baud rate to 115200
 Serial.println("ESP32 is ready!"); // Print the string "UNO is ready!"
}
void loop() {
 // Get the character input
 char keyPressed = myKeypad.getKey();
 // If there is a character input, sent it to the serial port
 if (keyPressed) {
   Serial.println(keyPressed);
 }
}
```

Compile and upload the code to ESP32, after the code is uploaded successfully, power up with a USB cable, and open the serial monitor and then set baud rate to 115200. You need to press the reset button on the ESP32 mainboard first, and then you will see that press the keyboard and the serial port monitor prints the corresponding key value, as shown below.

/d	ev/ttyUSB0		~	<b>^</b>	×
				Sen	d
ESP32 is ready! 1 2 3 A 4 5 6 8 9 C * 0 # D					
Autoscroll Show timestamp	Newline	• 115200 baud •	Clear	outpu	Jt

# 9.34.5 Wiring diagram of the Keypad Door

In the last experiment, we have known the key values of the 4\*4 matrix keyboard. Next, we use it as the keyboard to control a servo and a buzzer.

media/862e840117a46c1174522a734e28e2f0.png

6. Adding the Keypad and ESP32Servo libraries

If you have not downloaded the library file, please click on the link to download it:Download Arduino Libraries

The **Keypad** and **ESP32Servo** libraries had been added previously, so you don't need to add them again. If not, you need to add Keypad and **ESP32Servo** libraries. The steps to add third-party Libraries are as follows:

## 9.34.6 Project code

After the **Keypad** and **ESP32Servo** libraries were added, You can open the code we provideIf you haven't downloaded the code file, please click on the link to download itDownload Arduino C Codes file

The code used in this project is saved in folder "Arduino-Codes\Project 33Keypad Door\Project\_33.2\_Keypad\_Door".

```
#include <Keypad.h>
#include <ESP32Servo.h>
// define the symbols on the buttons of the keypad
char keys[4][4] = \{
  {'1', '2', '3', 'A'},
 {'4', '5', '6',
                 'B'},
 {'7', '8', '9', 'C'},
 {'*', '0', '#', 'D'}
}:
byte rowPins[4] = {22, 21, 19, 18}; // connect to the row pinouts of the keypad
byte colPins[4] = \{17, 16, 4, 0\}; // connect to the column pinouts of the keypad
// initialize an instance of class NewKeypad
Keypad myKeypad = Keypad(makeKeymap(keys), rowPins, colPins, 4, 4);
                  // Create servo object to control a servo
Servo myservo;
int servoPin = 15; // Define the servo pin
int buzzerPin = 2; // Define the buzzer pin
char passWord[] = {"1234"}; // Save the correct password
void setup() {
 myservo.setPeriodHertz(50);
                                       // standard 50 hz servo
 myservo.attach(servoPin, 500, 2500); // attaches the servo on servoPin to the servo_
→ object
                                       // set the high level time range of the servo
→motor for an accurate 0 to 180 sweep
 myservo.write(0);
                                       // Set the starting position of the servo motor
 pinMode(buzzerPin, OUTPUT);
 Serial.begin(115200);
}
void loop() {
  static char keyIn[4]; // Save the input character
  static byte keyInNum = 0; // Save the the number of input characters
  char keyPressed = myKeypad.getKey(); // Get the character input
  // Handle the input characters
  if (keyPressed) {
   // Make a prompt tone each time press the key
   digitalWrite(buzzerPin, HIGH);
   delay(100);
   digitalWrite(buzzerPin, LOW);
   // Save the input characters
   keyIn[keyInNum++] = keyPressed;
   // Judge the correctness after input
   if (keyInNum == 4) {
     bool isRight = true;
                                    // Save password is correct or not
     for (int i = 0; i < 4; i++) { // Judge each character of the password is correct_
\rightarrow or not
       if (keyIn[i] != passWord[i])
```

```
(continued from previous page)
         isRight = false;
                                    // Mark wrong passageword if there is any wrong_
\hookrightarrow character.
     }
     if (isRight) {
                                    // If the input password is right
       myservo.write(90);
                                   // Open the switch
                                    // Delay a period of time
       delay(2000);
                                   // Close the switch
       myservo.write(0);
       Serial.println("passWord right!");
     }
     else {
                                    // If the input password is wrong
       digitalWrite(buzzerPin, HIGH);// Make a wrong password prompt tone
       delay(1000);
       digitalWrite(buzzerPin, LOW);
       Serial.println("passWord error!");
     }
     keyInNum = 0; // Reset the number of the input characters to 0
   }
 }
}
```

# 9.34.7 Project result

Compile and upload the code to ESP32, after the code is uploaded successfully, power up with a USB cable and you will see that press the keypad to input password with 4 characters. If the input is correct(Correct password :1234), the servo will move to a certain degree, and then return to the original position. If the input is wrong, an input error alarm will be generated.

# 9.35 Project 34IR Control Sound and LED

# 9.35.1 Introduction

An infrared(IR) remote control is a low-cost and easy-to-use wireless communication technology. IR light is very similar to visible light, except that its wavelength is slightly longer. This means that infrared rays cannot be detected by the human eye, which is perfect for wireless communication. For example, when you press a button on the TV remote control, an infrared LED will switch on and off repeatedly at a frequency of 38,000 times per second, transmitting information (such as volume or channel control) to the infrared sensor on the TV.

We'll start by explaining how common infrared communication protocols work. Then we will start the project with a remote control and an infrared receiver component.

## 9.35.2 Components

ESP32*1	Breadboard*1	IR Receiver *1	RGB LED*1
			(IIII)
IR Remote Controller*1	Active buzzer*1	10KResistor*1	220Resistor*3
NPN transistor(S8050)*1	1kResistor*1	Jumper Wires	USB Cable*1

## 9.35.3 Component knowledge

\*\*Infrared Remote\*\*An infrared(IR) remote control is a device with a certain number of buttons. Pressing down different buttons will make the infrared emission tube, which is located in the front of the remote control, send infrared ray with different command. Infrared remote control technology is widely used in electronic products such as TVairconditioning, etc. Thus making it possible for you to switch TV programs and adjust the temperature of the air conditioning when away from them. The remote control we use is shown below:

The infrared remote controller adopts NEC code and the signal cycle is 110ms.

media/3c9d76c	b0d24d9861811ce2cb0bb6ae4.png

\*\*Infrared receiver\*\*receiver is a component which can receive the infrared light, so we can use it to detect the signal emitted by the infrared remote control.

The infrared receiver demodules the received infrared signal and converts it back to binary, then passes the information to the microcontroller.

Infrared signal modulation process diagram



NEC Infrared communication protocol

NEC Protocol:

To my knowledge the protocol I describe here was developed by NEC (Now Renesas). I've seen very similar protocol descriptions on the internet, and there the protocol is called Japanese Format. I do admit that I don't know exactly who developed it. What I do know is that it was used in my late VCR produced by Sanyo and was marketed under the name of Fisher. NEC manufactured the remote control IC. This description was taken from my VCR's service manual. Those were the days, when service manuals were filled with useful information!

Features:

- \* 8 bit address and 8 bit command length.
- \* Extended mode available, doubling the address size.
- \* Address and command are transmitted twice for reliability.
- \* Pulse distance modulation.
- \* Carrier frequency of 38kHz.
- \* Bit time of 1.125ms or 2.25ms.

#### Modulation:



The NEC protocol uses pulse distance encoding of the bits. Each pulse is a 560µs long 38kHz carrier burst (about 21 cycles). A logical "1" takes 2.25ms to transmit, while a logical "0" is only half of that, being 1.125ms. The recommended carrier duty-cycle is 1/4 or 1/3.

Protocol:



The picture above shows a typical pulse train of the NEC protocol. With this protocol the LSB is transmitted first. In this case Address 59andCommand16 is transmitted. A message is started by a 9ms AGC burst, which was used to set the gain of the earlier IR receivers. This AGC burst is then followed by a 4.5ms space, which is then followed by the Address and Command.

Address and Command are transmitted twice. The second time all bits are inverted and can be used for verification of the received message. The total transmission time is constant because every bit is repeated with its inverted length. If you're not interested in this reliability you can ignore the inverted values, or you can expand the Address and Command to 16 bits each!

Keep in mind that one extra  $560\mu$ s burst has to follow at the end of the message in order to be able to determine the value of the last bit.

media/63364daf21e5522c64eb8dfa82f2cef2.png

A command is transmitted only once, even when the key on the remote control remains pressed. Every 110ms a repeat code is transmitted for as long as the key remains down. This repeat code is simply a 9ms AGC pulse followed by a 2.25ms space and a 560µs burst.

media/afea92a3b5cc1aa2457d2b118b157c84.png

Extended NEC protocol:

The NEC protocol is so widely used that soon all possible addresses were used up. By sacrificing the address redundancy the address range was extended from 256 possible values to approximately 65000 different values. This way the address range was extended from 8 bits to 16 bits without changing any other property of the protocol. By extending the address range this way the total message time is no longer constant. It now depends on the total number of 1's and 0's in the message. If you want to keep the total message time constant you'll have to make sure the number 1's in the address field is 8 (it automatically means that the number of 0's is also 8). This will reduce the maximum number of different addresses to just about 13000.

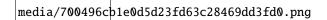
The command redundancy is still preserved. Therefore each address can still handle 256 different commands.



Keep in mind that 256 address values of the extended protocol are invalid because they are in fact normal NEC protocol addresses. Whenever the low byte is the exact inverse of the high byte it is not a valid extended address.

## 9.35.4 Decoded infrared signal

We connect the infrared receiving element to the ESP32, according to the wiring diagram below:



How to add the IRremoteESP8266 library

If you have not downloaded the library file, please click on the link to download it:Download Arduino Libraries

This code uses a library named "**IRremoteESP8266**", if you haven't installed it yet, please do so before learning. The steps to add third-party libraries are as follows:

After the **IRremoteESP8266** library is added, You can open the code we provideIf you haven't downloaded the code file, please click on the link to download itDownload Arduino C Codes file

The code used in this project is saved in folder "Arduino-Codes\Project 34IR Control Sound And LED\Project\_34.1\_Decoded\_IR\_Signal".

```
/*
* Filename : Decoded IR Signal
* Description : Decode the infrared remote control and print it out through the serial.
\rightarrow port.
* Auther
            : http//www.keyestudio.com
*/
#include <Arduino.h>
#include <IRremoteESP8266.h>
#include <IRrecv.h>
#include <IRutils.h>
const uint16_t recvPin = 0; // Infrared receiving pin
IRrecv irrecv(recvPin); // Create a class object used to receive class
decode_results results;
                       // Create a decoding results class object
void setup() {
 Serial.begin(115200); // Initialize the serial port and set the baud rate to.
→115200
 irrecv.enableIRIn(); // Start the receiver
 Serial.print("IRrecvDemo is now running and waiting for IR message on Pin ");
 Serial.println(recvPin); //print the infrared receiving pin
}
void loop() {
 if (irrecv.decode(&results)) { // Waiting for decoding
   serialPrintUint64(results.value, HEX);// Print out the decoded results
   Serial.println("");
                                   // Release the IRremote. Receive the next value
   irrecv.resume();
 }
 delay(1000);
}
```

Compile and upload the code to ESP32, after the code is uploaded successfully, power up with a USB cable, and open the serial monitor and then set baud rate to 115200. You need to press the reset button on the ESP32 mainboard first, and then you will see that aim the infrared remote control transmitter at the infrared receiving head, press the button on the infrared controller, and the serial port monitor prints the current received key code values.

	/dev/ttyUSB0		~ ^ X
			Send
FF629D FF22DD FF4857 FFC23D FF6897 FF9867 FFB04F FF30CF			
Autoscroll Show timestamp	Newline	<ul> <li>▼ 115200 baud</li> <li>▼</li> </ul>	Clear output

Write down the code associated with each button, because you will need that information later.

media/ebcf0cb2055f7784505f76ceeaef9f47.jpeg

## 9.35.5 Wiring diagram of the infrared remote control



## 9.35.6 Project code

If you have not downloaded the library file, please click on the link to download it:Download Arduino Libraries

The **IRremoteESP8266** library had been added previously, so you don't need to add it again. If not, you need to add the IRremoteESP8266 library. The steps to add third-party libraries are as shown above.

After the **IRremoteESP8266** library was added, You can open the code we provideIf you haven't downloaded the code file, please click on the link to download itDownload Arduino C Codes file

The code used in this project is saved in folder "Arduino-Codes\Project 34IR Control Sound And LED\Project\_34.2\_IR\_Control\_Sound\_And\_LED".

```
* Auther
               : http//www.keyestudio.com
*/
#include <Arduino.h>
#include <IRremoteESP8266.h>
#include <IRrecv.h>
#include <IRutils.h>
const uint16_t recvPin = 0; // Infrared receiving pin
IRrecv irrecv(recvPin); // Create a class object used to receive class
decode_results results;
                             // Create a decoding results class object
int ledPins[] = {22, 21, 4}; //define red, green, blue led pins
const byte chns[] = \{0, 1, 2\};
                                     //define the pwm channels
int buzzerPin = 15; // the number of the buzzer pin
void setup() {
 irrecv.enableIRIn();
                                       // Start the receiver
  pinMode(buzzerPin, OUTPUT);
  for (int i = 0; i < 3; i++) { //setup the pwm channels,1KHz,8bit</pre>
   ledcSetup(chns[i], 1000, 8);
   ledcAttachPin(ledPins[i], chns[i]);
  }
}
void loop() {
 if(irrecv.decode(&results)) {
                                  // Waiting for decoding
                                      // Handle the commands from remote control
   handleControl(results.value);
   irrecv.resume();
                                       // Receive the next value
 }
}
void handleControl(unsigned long value) {
  // Make a sound when it rereives commands
  digitalWrite(buzzerPin, HIGH);
  delay(100):
  digitalWrite(buzzerPin, LOW);
  // Handle the commands
  if (value == 0xFF6897) // Receive the number '1'
  {
      ledcWrite(chns[0], 255); //Common cathode LED, high level to turn on the led.
     ledcWrite(chns[1], 0);
     ledcWrite(chns[2], 0);
     delay(1000);
  }
  else if (value == 0xFF9867) // Receive the number '2'
  {
      ledcWrite(chns[0], 0);
      ledcWrite(chns[1], 255);
     ledcWrite(chns[2], 0);
     delay(1000);
  }
   else if (value == 0xFFB04F) // Receive the number '3'
```

```
{
     ledcWrite(chns[0], 0);
     ledcWrite(chns[1], 0);
     ledcWrite(chns[2], 255);
     delay(1000);
   }
   else if (value == 0xFF30CF) // Receive the number '4'
  {
     ledcWrite(chns[0], 255);
     ledcWrite(chns[1], 255);
     ledcWrite(chns[2], 0);
     delay(1000);
   }
   else if (value == 0xFF18E7) // Receive the number '5'
  {
     ledcWrite(chns[0], 255);
     ledcWrite(chns[1], 0);
     ledcWrite(chns[2], 255);
     delay(1000);
   }
   else if (value == 0xFF7A85) // Receive the number '6'
  {
     ledcWrite(chns[0], 0);
     ledcWrite(chns[1], 255);
     ledcWrite(chns[2], 255);
     delay(1000);
   }
   else if (value == 0xFF10EF) // Receive the number '7'
  {
     ledcWrite(chns[0], 255);
     ledcWrite(chns[1], 255);
     ledcWrite(chns[2], 255);
     delay(1000);
   }
   else{
     ledcWrite(chns[0], 0);
     ledcWrite(chns[1], 0);
     ledcWrite(chns[2], 0);
     delay(1000);
     }
 }
//**********
```

# 9.35.7 Project result

Compile and upload the code to ESP32, after the code is uploaded successfully, power up with a USB cable and you will see that press the 1 to 7 key of the infrared remote controller, the buzzer will sound once, and the RGB light will be redgreenblueyellowredbluegreen and white respectively. Press another key (except 1 to 7 key), and the RGB light will go off.

(Note: Before use, we need to remove the plastic sheet from the bottom of the infrared remote controller.)

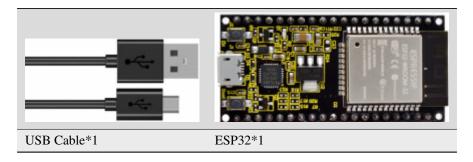


# 9.36 Project 35Bluetooth

This chapter mainly introduces how to make simple data transmission through bluetooth of ESP32 and mobile phones. Project 35.1 is Classic Bluetooth and Project 35.2 is Bluetooth Control LED.

# 9.37 Project 35.1Classic Bluetooth

# 9.37.1 Components



In this tutorial we need to use a Bluetooth APP called Serial Bluetooth Terminal to assist in the experiment. If you've not installed it yet, please do so by clicking: https://www.appsapk.com/serial-bluetooth-terminal/ .The following is its logo.

media/7b98d6708888b0a6f38f85ffca484857.png

# 9.37.2 Component knowledge

ESP32's integrated Bluetooth function Bluetooth is a short-distance communication system, which can be divided into two types, namely Bluetooth Low Energy(BLE) and Classic Bluetooth. There are two modes for simple data transmission: master mode and slave mode.

Master mode

In this mode, works are done in the master device and it can connect with a slave device. And we can search and select slave devices nearby to connect with. When a device initiates connection request in master mode, it requires information of the other Bluetooth devices including their address and pairing passkey. After finishing pairing, it can connect with them directly.

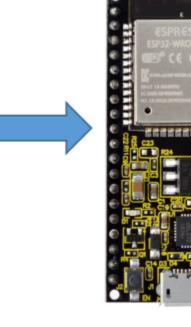
Slave mode

The Bluetooth module in slave mode can only accept connection request from a host computer, but cannot initiate a connection request. After connecting with a host device, it can send data to or receive from the host device.

Bluetooth devices can make data interaction with each other, as one is in master mode and the other in slave mode. When they are making data interaction, the Bluetooth device in master mode searches and selects devices nearby to connect to. When establishing connection, they can exchange data. When mobile phones

exchange data with ESP32, they are usually in master mode and ESP32 in slave mode.





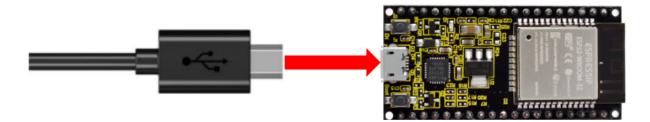
Master

Slave

Master Slave

# 9.37.3 Wiring Diagram

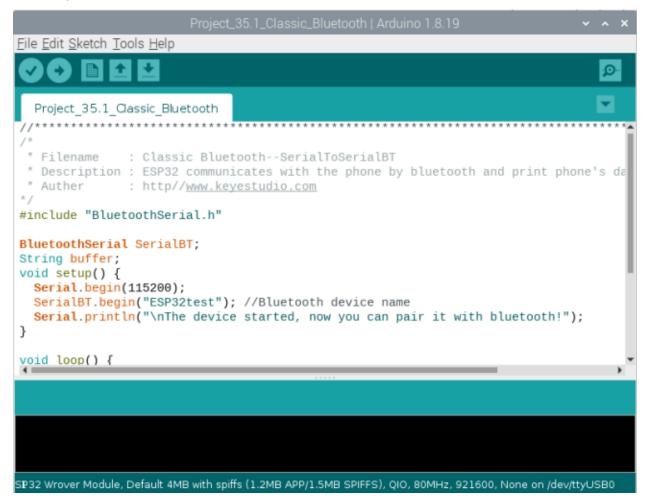
We can use a USB cable to connect ESP32 mainboard to the USB port on the Raspberry Pi.



# 9.37.4 Project code

You can open the code we provideIf you haven't downloaded the code file, please click on the link to download itDownload Arduino C Codes file

The code used in this project is saved in folder(path:) "Arduino-Codes\Project 35:lue-tooth\Project\_35.1\_Classic\_Bluetooth".



```
/*
* Filename
           : Classic Bluetooth--SerialToSerialBT
* Description : ESP32 communicates with the phone by bluetooth and print phone's data.
→via a serial port
* Auther
          : http//www.keyestudio.com
*/
#include "BluetoothSerial.h"
BluetoothSerial SerialBT;
String buffer;
void setup() {
 Serial.begin(115200);
 SerialBT.begin("ESP32test"); //Bluetooth device name
 Serial.println("\nThe device started, now you can pair it with bluetooth!");
}
void loop() {
 if (Serial.available()) {
   SerialBT.write(Serial.read());
 }
 if (SerialBT.available()) {
   Serial.write(SerialBT.read());
 }
 delay(20);
}
```

# 9.37.5 Project result

Compile and upload the code to the ESP32. After uploading successfullywe will use a USB cable to power on. Open the serial monitor and set the baud rate to **115200**. You need to press the reset button on the ESP32 mainboard first, and when you see the serial monitor prints out the character string as below, it indicates that the Bluetooth of ESP32 is ready and waiting to connect with the mobile phone. (If open the serial monitor and set the baud rate to 115200, the information is not displayed, please press the RESET button of the ESP32)

media/1fd21fafd84d2b529931a89d21a03d6a.png

/dev/ttyUSB0	~ ^ X
	Send
<pre>rst:0x1 (POWERON_RESET),boot:0x13 (SPI_FAST_FLASH_BOOT) configsip: 0, SPIWP:0xee clk_drv:0x00,q_drv:0x00,d_drv:0x00,cs0_drv:0x00,hd_drv:0x00,wp_drv:0x00 mode:DIO, clock div:1 load:0x3fff0018,len:4 load:0x3fff001c,len:1216 ho 0 tail 12 room 4 load:0x40078000,len:10944 load:0x40080400,len:6388 entry 0x400806b4 E (137) psram: PSRAM ID read error: 0xffffffff</pre>	Î
The device started, now you can pair it with bluetooth!	
✓ Autoscroll Show timestamp Newline • 115200 baud •	Clear output

Make sure that the Bluetooth of your phone has been turned on and "Serial Bluetooth Terminal" has been installed.

media/382529e	def3989e60264cad217d88e6f.png

Click "Search" to search Bluetooth devices nearby and select "ESP32 test" to connect to.

media/0608c9a78b5f56d4c8f1994c55c9cd46.png

Turn on software APP, click the left of the terminal. Select "Devices" .

media/32b8c3abd51fc538ba854b1d72e1165e.png

Select ESP32test in classic Bluetooth mode, and a successful connecting prompt will appear as shown on the right illustration.

media/00f9b335cb512704763e3621e7c598b2.png

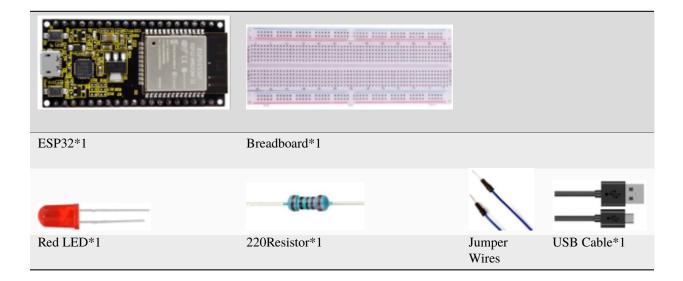
And now data can be transferred between your mobile phone and Raspberry Pi via ESP32.

Send "Hello!" from your phone, when the computer receives it, reply "Hi!" to your phone.

/dev/ttyUSB0	~ ^ X
Hil	Send
configsip: 0, SPIWP:0xee clk_drv:0x00,q_drv:0x00,d_drv:0x00,cs0_drv:0x00,hd_drv:0x00,wp_drv:0x00	^
<pre>mode:DI0, clock div:1 load:0x3fff0018,len:4 load:0x3fff001c,len:1216 ho 0 tail 12 room 4 load:0x40078000,len:10944 load:0x40080400,len:6388 entry 0x400806b4 E (137) psram: PSRAM ID read error: 0xfffffff</pre>	
The device started, now you can pair it with bluetooth! Hello!	
✓ Autoscroll Show timestamp Newline • 115200 baud •	Clear output
media/4f4e6b4e45996ccbde4da17219f02d00.png	

# 9.38 Project 35.2Bluetooth Control LED

# 9.38.1 Components



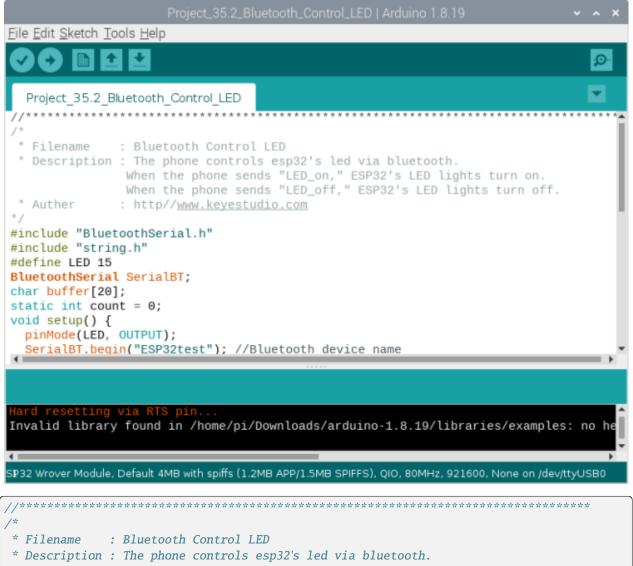
#### 9.38.2 Wiring diagram



#### 9.38.3 Project code

You can open the code we provideIf you haven't downloaded the code file, please click on the link to download itDownload Arduino C Codes file

The code used in this project is saved in folder(path:) "Arduino-Codes\Project 35:Blue-tooth\Project\_35.2\_Bluetooth\_Control\_LED".



```
When the phone sends "LED_on," ESP32's LED lights turn on.
              When the phone sends "LED_off," ESP32's LED lights turn off.
* Auther
              : http//www.keyestudio.com
*/
#include "BluetoothSerial.h"
#include "string.h"
#define LED 15
BluetoothSerial SerialBT;
char buffer[20];
static int count = 0;
void setup() {
 pinMode(LED, OUTPUT);
 SerialBT.begin("ESP32test"); //Bluetooth device name
 Serial.begin(115200);
 Serial.println("\nThe device started, now you can pair it with bluetooth!");
}
void loop() {
 while(SerialBT.available())
 {
   buffer[count] = SerialBT.read();
   count++;
 }
 if(count>0){
   Serial.print(buffer);
   if(strncmp(buffer, "led_on", 6)==0){
     digitalWrite(LED,HIGH);
   }
   if(strncmp(buffer, "led_off", 7)==0){
     digitalWrite(LED,LOW);
   }
   count=0;
   memset(buffer,0,20);
 }
}
```

#### 9.38.4 Project result

Compile and upload the code to the ESP32, after uploading successfullywe will use a USB cable to power on. The operation of the APP is the same as Project 35.1, you only need to change the sending content to "LED on" and "LED off" to operate LEDs on the ESP32. Data sent from mobile APP:

media/21ec63eBabe43a119ab8a3d4634894f0.png

Display on the serial port of the computer:

/dev/ttyUSB0	~	^	×
		Sen	d
load:0x3fff001c,len:1216 ho 0 tail 12 room 4 load:0x40078000,len:10944 load:0x40080400,len:6388 entry 0x400806b4 E (137) psram: PSRAM ID read error: 0xffffffff S The device started, now you can pair it with bluetooth! abcd led_on led_off led_off led_off			
✓ Autoscroll _ Show timestamp       Newline <ul> <li>115200 baud</li> <li>Classical</li> <li>Clastical</li> <li>Classical</li> <li></li></ul>	ear o	outp	ut

The phenomenon of LED

# media/a1e66d46e5797995a1b66a761c7857f8.png

Attention: If the sending content isn't "led-on" or "led-off", then the state of LED will not change. If the LED is on, when receiving irrelevant content, it keeps on; Correspondingly, if the LED is off, when receiving irrelevant content, it keeps off.

# 9.39 Project 36WiFi Station Mode

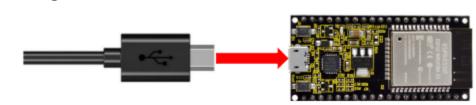
#### 9.39.1 Introduction

ESP32 has three different WiFi operating modes : Station modeAP mode and AP+Station mode. All WiFi programming projects must be configured with WiFi operating mode before using WiFi, otherwise WiFi cannot be used. In this project, we will learn about ESP32's WiFi Station mode.

# 9.39.2 Components

USB Cable*1	ESP32*1

## 9.39.3 Wiring



Connect the ESP32 to the USB port on your raspberry pi using a USB cable.

#### 9.39.4 Component knowledge

**Station mode:** When ESP32 selects Station mode, it acts as a WiFi client. It can connect to the router network and communicate with other devices on the router via WiFi connection. As shown below, the PC is connected to the router, and if ESP32 wants to communicate with the PC, it needs to be connected to the router.

media/f74baff97695aa2ee33a8c19370d2547.png

#### 9.39.5 Project code

You can open the code we provideIf you haven't downloaded the code file, please click on the link to download itDownload Arduino C Codes file

The code used in this project is saved in folder(path:) "Arduino-Codes\Project 36WiFi Station Mode\Project\_36\_WiFi\_Station\_Mode".

Project_36_WiFi_Station_Mode   Arduino 1.8.19 🔷 🗸	×
<u>File Edit Sketch Tools H</u> elp	
Serial Monitor 🖌 Serial Monitor	<b>D</b> -
Project_36_WiFi_Station_Mode	1
<pre>//***********************************</pre>	Î
<pre>#include <wifi.h> //Include the WiFi Library header file of ESP32.</wifi.h></pre>	
<pre>//Enter correct router name and password. const char *ssid_Router = "ChinaNet-2.4G-0DF0"; //Enter the router name const char *password_Router = "ChinaNet@233"; //Enter the router password</pre>	1
<pre>void setup(){    Serial.begin(115200);    delay(2000);    Serial.println("Setup start");    WiFi.begin(ssid Router, password Router);//Set ESP32 in Station mode and connect </pre>	t_i≖ ≯
Hard resetting via RTS pin Invalid library found in /home/pi/Downloads/arduino-1.8.19/libraries/examples: no	he
▲ SP32 Wrover Module, Default 4MB with spiffs (1.2MB APP/1.5MB SPIFFS), QIO, 80MHz, 921600, None on /dev/ttyUSB	•

Because the names and passwords of routers in various places are different, before the code runs, users need to enter the correct router's name and password in the box as shown in the illustration above.

Project_3	36_WiFi_Station_Mode   Arduino 1.8.19 🛛 🗸 🔺 🗙
<u>File E</u> dit <u>S</u> ketch <u>T</u> ools <u>H</u> elp	
	Serial Monitor 😥
Project_36_WiFi_Station_Mode	■
//************************************	*******
* Filename : WiFi Station * Description : Connect to you	ur router using ESD22
* Auther : http// <u>www.key</u>	estudio.com Enter the correct Router
*/ #include <wifi.h> //Include the</wifi.h>	e WiFi Library header file of ESP32.
	d password "ChinaNet-2.4G-0DF0"; //Enter the router name "ChinaNet@233"; //Enter the router password
<pre>void setup(){    Serial.begin(115200);    delay(2000);    Serial.println("Setup start"    WiFi begin(ssid Router</pre>	); word Router);//Set ESP32 in Station mode and connect i▼
A MIPI. Degin(SSIG Router, passi	word Router), // Set ESP32 In Station mode and connect i
Hard resetting via RTS pin Invalid library found in /home/	/pi/Downloads/arduino-1.8.19/libraries/examples: no he
SP32 Wrover Module, Default 4MB with spiff	s (1.2MB APP/1.5MB SPIFFS), QIO, 80MHz, 921600, None on /dev/ttyUSB0
//**********************************	*******
/* * Filename : WiFi Station	
<pre>* Description : Connect to your * Auther : http//www.keyes</pre>	
*/ #include <wifi.h> //Include the</wifi.h>	WiFi Library header file of ESP32.
//Enter correct router name and p	password.
<pre>const char *ssid_Router = "Character"</pre>	hinaNet-2.4G-0DF0"; //Enter the router name hinaNet@233"; //Enter the router password
<pre>void setup(){    Serial.begin(115200);    delaw(2000);</pre>	
<pre>delay(2000); Serial.println("Setup start"); WiFi.begin(ssid_Router, password)</pre>	rd_Router);//Set ESP32 in Station mode and connect it to.
→your router. Serial.println(String("Connect:	ing to ")+ssid Router).
	ted to router successfully every 0.5s.
delay(500);	(continues on next page)
	(continues on next page)

Project result

After making sure the router name and password are entered correctly, compile and upload the code to ESP32, open serial monitor and set baud rate to 115200. You need to press the reset button on the ESP32 mainboard first, and when ESP32 successfully connects to "ssid\_Router", serial monitor will print out the IP address assigned to ESP32 by the router. (If open the serial monitor and set the baud rate to 115200, the information is not displayed, please press the RESET button of the ESP32)

media/1fd21fafd84d2b529931a89d21a03d6a.png

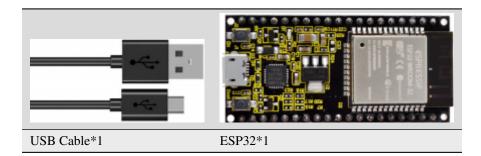
	/dev/ttyUSB0		~ ^ X
			Send
<pre>load:0x3fff0018,len:4 load:0x3fff001c,len:1216 ho 0 tail 12 room 4 load:0x40078000,len:10944 load:0x40080600,len:6388 entry 0x400806b4 F (52) psram: PSRAM ID read error: 0xff Setup start Connecting to ChinaNet-2.4G-0DF0  Connected, IP address: 192.168.0.154 Setup End</pre>	fffff		Î
Autoscroll 🗌 Show timestamp	Newline	▼ 115200 baud ▼	Clear output

# 9.40 Project 37WiFi AP Mode

#### 9.40.1 Introduction

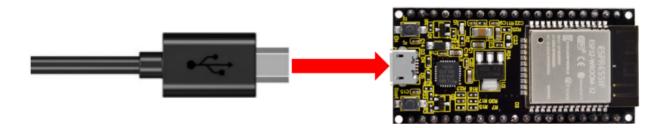
ESP32 has three different WiFi operating modes : Station modeAP mode and AP+Station mode. All WiFi programming projects must be configured with WiFi operating mode before using WiFi, otherwise WiFi cannot be used. In this project, we will learn about ESP32's WiFi AP mode.

## 9.40.2 Components



#### 9.40.3 Wiring

Connect the ESP32 to the USB port on your raspberry pi using a USB cable.



# 9.40.4 Component knowledge

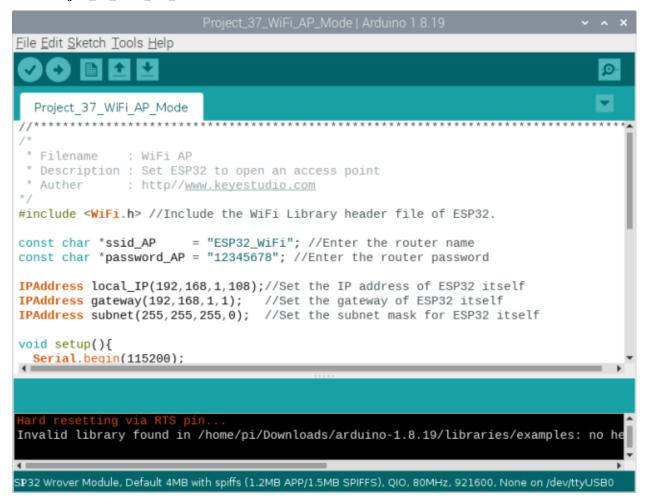
**AP mode :** When ESP32 selects AP mode, it creates a hotspot network that is separated from the Internet and waits for other WiFi devices to connect. As shown in the figure below, ESP32 is used as a hotspot. If a mobile phone or PC wants to communicate with ESP32, it must be connected to the hotspot of ESP32. Only after a connection is established with ESP32 can they communicate.

media/35d90f1ce10814ea1897ba63f8bd7ad9.png

## 9.40.5 Project code

You can open the code we provideIf you haven't downloaded the code file, please click on the link to download itDownload Arduino C Codes file

The code used in this project is saved in folder(path:) "Arduino-Codes\Project 37WiFi AP Mode\Project\_37\_WiFi\_AP\_Mode".



Before the code runs, you can make any changes to the AP name and password for ESP32 in the box as shown in the illustration above. Of course, you can leave it alone by default.

Project_37_WiFi_AP_Mode   Arduino 1.8.19 🗸 🔺 🗙
<u>File Edit Sketch Tools H</u> elp
Project_37_WiFi_AP_Mode
//*************************************
<pre>* Filename : WiFi AP Set a name and a * Description : Set ESP32 to open an access point * Auther : http//www.keyestudio.com password for ESP32 AP. */ #include <kifi b=""> //Include the kiFi Library beader file of ESP32</kifi></pre>
<pre>#include <wifi.h> //Include the WiFi Library header file of ESP32.</wifi.h></pre>
<pre>const char *ssid_AP = "ESP32_WiFi"; //Enter the router name const char *password_AP = "12345678"; //Enter the router password</pre>
<pre>IPAddress local_IP(192,168,1,108);//Set the IP address of ESP32 itself IPAddress gateway(192,168,1,1); //Set the gateway of ESP32 itself IPAddress subnet(255,255,255,0); //Set the subnet mask for ESP32 itself</pre>
<pre>void setup(){     Serial.begin(115200); </pre>
4 · · · · · · · · · · · · · · · · · · ·
Hard resetting via RTS pin Invalid library found in /home/pi/Downloads/arduino-1.8.19/libraries/examples: no he SP32 Wrover Module, Default 4MB with spiffs (1.2MB APP/1.5MB SPIFFS), QIO, 80MHz, 921600, None on /dev/ttyUSB0
//*************************************
/* * Filename : WiFi AP
* Description : Set ESP32 to open an access point * Auther : http://www.keyestudio.com
*/ #include <wifi.h> //Include the WiFi Library header file of ESP32.</wifi.h>
<pre>const char *ssid_AP = "ESP32_WiFi"; //Enter the router name</pre>
<pre>const char *password_AP = "12345678"; //Enter the router password</pre>
<pre>IPAddress local_IP(192,168,1,108);//Set the IP address of ESP32 itself IPAddress gateway(192,168,1,1); //Set the gateway of ESP32 itself IPAddress subnet(255,255,255,0); //Set the subnet mask for ESP32 itself</pre>
<pre>void setup(){    Serial.begin(115200);    delew(2000); </pre>
<pre>delay(2000); Serial.println("Setting soft-AP configuration "); WiFi.disconnect();</pre>
WiFi.mode(WIFI_AP);
<pre>Serial.println(WiFi.softAPConfig(local_IP, gateway, subnet) ? "Ready" : "Failed!");</pre>

```
Serial.println("Setting soft-AP ... ");
boolean result = WiFi.softAP(ssid_AP, password_AP);
if(result){
   Serial.println("Ready");
   Serial.println(String("Soft-AP IP address = ") + WiFi.softAPIP().toString());
   Serial.println(String("MAC address = ") + WiFi.softAPmacAddress().c_str());
}else{
   Serial.println("Failed!");
}
Serial.println("Setup End");
}
void loop() {
}
```

#### 9.40.6 Project result

Enter the ESP32 AP name and password correctly, compile and upload the code to ESP32, open the serial monitor and set the baud rate to **115200**. You need to press the reset button on the ESP32 mainboard first, and then it will display as follows.

(If open the serial monitor and set the baud rate to 115200, the information is not displayed, please press the RESET button of the ESP32)

media/1fd21fafd84d2b529931a89d21a03d6a.png

	/dev/ttyUSB0 🗸		
		Ser	nd
load:0x3fff001c,len:1216 ho 0 tail 12 room 4 load:0x40078000,len:10944 load:0x40080400,len:6388 entry 0x400806b4			^
F (53) psram: PSRAM ID read error: Setting soft-AP configuration Ready Setting soft-AP Ready Soft-AP IP address = 192.168.1.108 MAC address = 58:BF:25:8A:2F:E1 Setup End	Ðxffffffff		
Autoscroll 🗌 Show timestamp	Newline • 115200 baud • Clear	r outp	ut

When observing the print information of the serial monitor, turn on the WiFi scanning function of your phone, and you can see the ssid\_AP on ESP32, which is called "ESP32\_Wifi" in this Code. You can enter the password "12345678" to connect it or change its AP name and password by modifying Code.

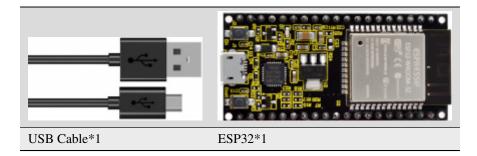
media/3e0ad895bea7f5100cc02a415adcace7.png

# 9.41 Project 38WiFi Station+AP Mode

## 9.41.1 Introduction

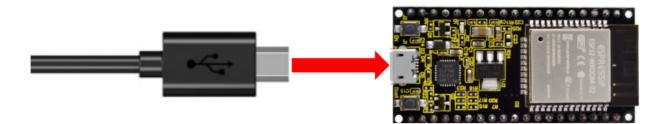
ESP32 has three different WiFi operating modes : Station modeAP mode and AP+Station mode. All WiFi programming projects must be configured with WiFi operating mode before using WiFi, otherwise WiFi cannot be used. In this project, we will learn ESP32's WiFi Station+AP mode.

# 9.41.2 Components



# 9.41.3 Wiring

Connect the ESP32 to the USB port on your raspberry pi using a USB cable.



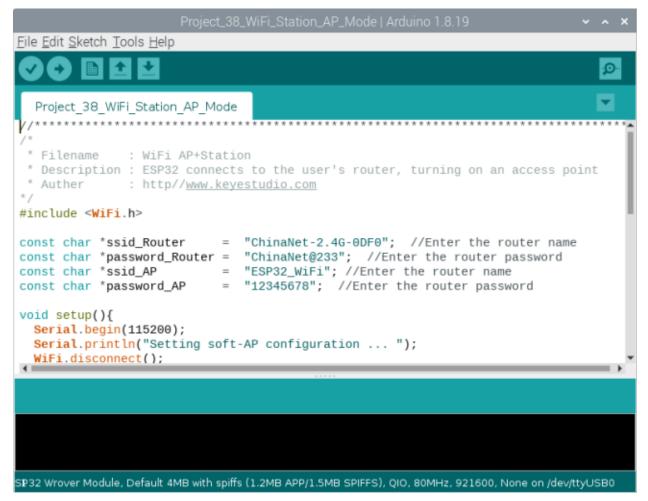
#### 9.41.4 Component knowledge

**AP+Station mode:** In addition to AP mode and Station mode, ESP32 can also use AP mode and Station mode at the same time. This mode contains the functions of the previous two modes. Turn on ESP32's Station mode, connect it to the router network, and it can communicate with the Internet via the router. At the same time, turn on its AP mode to create a hotspot network. Other WiFi devices can choose to connect to the router network or the hotspot network to communicate with ESP32.

#### 9.41.5 Project code

You can open the code we provideIf you haven't downloaded the code file, please click on the link to download itDownload Arduino C Codes file

The code used in this project is saved in folder(path:) "Arduino-Codes\Project 38WiFi Station+AP Mode\Project\_38\_WiFi\_Station\_AP\_Mode".



It is analogous to Project 36 and Project 37. Before running the code, you need to modify ssid\_Router, password\_Router, ssid\_AP and password\_AP shown in the box of the illustration above.

Project_38_WiFi_Station_AP_Mode   Arduino 1.8.19
<u>File Edit Sketch Tools H</u> elp
Project_38_WiFi_Station_AP_Mode
/* * Filename : WiFi AP+Station Please enter the correct names * Description : ESP32 connects to the user's router, turning on an access point * Auther : http://www.keyestudio.com and passwords of Router and AP. */
#include <wifi.h></wifi.h>
<pre>const char *ssid_Router = "ChinaNet-2.4G-0DF0"; //Enter the router name const char *password_Router = "ChinaNet@233"; //Enter the router password const char *ssid_AP = "ESP32_WiFi"; //Enter the router name const char *password_AP = "12345678"; //Enter the router password</pre>
<pre>void setup(){    Serial.begin(115200);    Serial.println("Setting soft-AP configuration ");    WiFi.disconnect(); </pre>
SP32 Wrover Module, Default 4MB with spiffs (1.2MB APP/1.5MB SPIFFS), QIO, 80MHz, 921600, None on /dev/ttyUSB0
/* * Filename : WiFi AP+Station
* Description : ESP32 connects to the user's router, turning on an access point
* Auther : http//www.keyestudio.com
*/ #include <wifi.h></wifi.h>
<pre>const char *ssid_Router = "ChinaNet-2.4G-0DF0"; //Enter the router name const char *password_Router = "ChinaNet@233"; //Enter the router password const char *ssid_AP = "ESP32_WiFi"; //Enter the router name const char *password_AP = "12345678"; //Enter the router password</pre>
void setun(){
<pre>void setup(){    Serial.begin(115200);</pre>
<pre>Serial.begin(115200); Serial.println("Setting soft-AP configuration ");</pre>
<pre>Serial.begin(115200); Serial.println("Setting soft-AP configuration "); WiFi.disconnect();</pre>
<pre>Serial.begin(115200); Serial.println("Setting soft-AP configuration ");</pre>
<pre>Serial.begin(115200); Serial.println("Setting soft-AP configuration "); WiFi.disconnect(); WiFi.mode(WIFI_AP); Serial.println("Setting soft-AP "); boolean result = WiFi.softAP(ssid_AP, password_AP);</pre>
<pre>Serial.begin(115200); Serial.println("Setting soft-AP configuration "); WiFi.disconnect(); WiFi.mode(WIFI_AP); Serial.println("Setting soft-AP ");</pre>

```
Serial.println(String("Soft-AP IP address = ") + WiFi.softAPIP().toString());
   Serial.println(String("MAC address = ") + WiFi.softAPmacAddress().c_str());
 }else{
   Serial.println("Failed!");
 }
 Serial.println("\nSetting Station configuration ... ");
 WiFi.begin(ssid_Router, password_Router);
 Serial.println(String("Connecting to ")+ ssid_Router);
 while (WiFi.status() != WL_CONNECTED){
   delay(500);
   Serial.print(".");
 }
 Serial.println("\nConnected, IP address: ");
 Serial.println(WiFi.localIP());
 Serial.println("Setup End");
}
void loop() {
}
  ************************
```

# 9.41.6 Project result

After making sure that the code is modified correctly, compile and upload the code to ESP32, open the serial monitor and set baud rate to 115200.

You need to press the reset button on the ESP32 mainboard first, and then it will display as follows: (If open the serial monitor and set the baud rate to 115200, the information is not displayed, please press the RESET button of the ESP32)

media/1fd21fafd84d2b529931a89d21a03d6a.png

	/dev/ttyUSB0	~ ^ X
		Send
F (53) psram: PSRAM ID read error: Setting soft-AP configuration Setting soft-AP Ready Soft-AP IP address = 192.168.4.1 MAC address = 58:BF:25:8A:2F:E1 Setting Station configuration Connecting to ChinaNet-2.4G-0DF0  Connected, IP address: 192.168.0.154 Setup End		
✓ Autoscroll  Show timestamp	Newline 🔹 115200 baud 💌 Cle	ear output

When observing the print information of the serial monitor, turn on the WiFi scanning function of your phone, and you can see the ssid\_AP on ESP32.

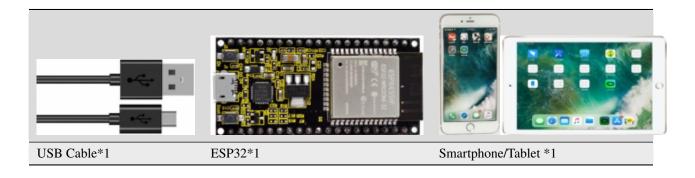
media/3e0ad895bea7f5100cc02a415adcace7.png

# 9.42 Project 39: WiFi Test

#### 9.42.1 Introduction

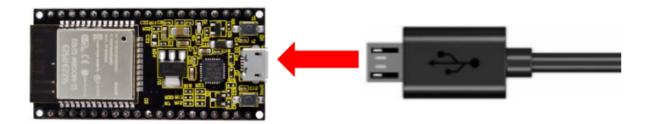
In this experiment, we first use the WiFi station mode of ESP32 to read the IP address of WiFi, and then connect WiFi through app to read the characters sent by each function button on App.

# 9.42.2 Components



## 9.42.3 Wiring

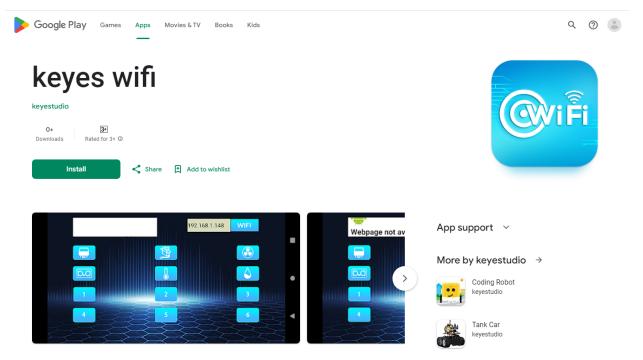
Connect the ESP32 to the USB port on your raspberry pi using a USB cable.



#### 9.42.4 Install APP:

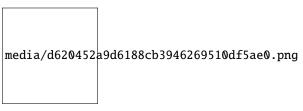
Android system (Smartphone/Tablet) APP:Go to Google Store to search for keys wifi

App link in Google Store:https://play.google.com/store/apps/details?id=com.keyestudio.esp8266\_web\_wifi2



Installation steps

Now transfer the **keyes wifi.apk** file in the folder to android phone or tablet, click **keyes wifi.apk** file to enter the installation page, click"ALLOW", then click"INSTALL", after a while, click"OPEN"after the installation is completed to enter the APP interface.



media/b311329042f5bbd2880841127b91ebf8.png

media/7c5cfc935371c8e2ab30e999775d5f8f.png

media/d48c065ebaf1c5ca652eb72b15d3e596.png

media/78c89b91c0af2268f6267813e7923a9b.png

IOS system (Smartphone/Tablet) APP:

a.Open App Store.

media/27924fdb3d67692df7c63d8d0fb72287.png

media/962a57f92b78eea1f0e3e814

b. Enter keyes link in the search box and click Search. The download interface appears. Click"\_\_\_\_\_\_'to download and install the APP of keyes link. The following operations are similar to those of Android system, you can refer to the steps of Android system above for operation.

# 9.42.5 Project code

You can open the code we provideIf you haven't downloaded the code file, please click on the link to download itDownload Arduino C Codes file

The code used in this project is saved in folder(path:) "Arduino-Codes\Project 39WiFi Test\Project\_39\_WiFi\_Test".

```
/*
* Filename : WIFI Test
* Description : Wifi module test the ip of Wifi
* Auther : http//www.keyestudio.com
*/
// generated by KidsBlock
#include <Arduino.h>
#include <WiFi.h>
#include <ESPmDNS.h>
#include <WiFiClient.h>
String item = "0";
const char* ssid = "ChinaNet-2.4G-0DF0";
const char* password = "ChinaNet@233";
WiFiServer server(80);
void setup() {
 Serial.begin(115200);
 WiFi.begin(ssid, password);
 while (WiFi.status() != WL_CONNECTED) {
   delay(500);
   Serial.print(".");
 }
 Serial.println("");
 Serial.print("Connected to ");
 Serial.println(ssid);
 Serial.print("IP address: ");
 Serial.println(WiFi.localIP());
 server.begin();
 Serial.println("TCP server started");
 MDNS.addService("http", "tcp", 80);
}
void loop() {
 WiFiClient client = server.available();
 if (!client) {
     return;
 }
 while(client.connected() && !client.available()){
     delay(1);
 }
 String req = client.readStringUntil('\r');
 int addr_start = req.indexOf(' ');
 int addr_end = req.indexOf(' ', addr_start + 1);
 if (addr_start == -1 || addr_end == -1) {
     Serial.print("Invalid request: ");
```

```
Serial.println(req);
     return;
 }
 req = req.substring(addr_start + 1, addr_end);
 item=rea:
 Serial.println(item);
 String s;
 if (req == "/")
 {
     IPAddress ip = WiFi.localIP();
     String ipStr = String(ip[0]) + '.' + String(ip[1]) + '.' + String(ip[2]) + '.' +.
→String(ip[3]);
     s = "HTTP/1.1 200 OK\r\nContent-Type: text/html\r\n\r\n<!DOCTYPE HTML>\r\n<html>
→Hello from ESP32 at ";
     s += ipStr;
     s += "</html>\r\n';
     Serial.println("Sending 200");
     client.println(s);
 }
 //client.print(s);
 client.stop();
}
```

Special attention: you need to change the user's Wifi name and Wifi password in the experiment

	media/8555128	5d37d87fc8decadc09f968ec3.png
code		to your own Wifi name and Wifi password.

5. Project result

After making sure that the Code is modified correctly, compile and upload the code to ESP32.Note: If upload-

media/d09c4a31563f04a42d451e7bc1a5fb8a.png

ing the code fails, you can press the Boot button on ESP32 after click, and release the Boot

media/dc77bfcf5851c8f43aab6cbe7cec7920.png

button after the percentage of uploading progress appears)open the serial monitor and set baud rate to 115200.

You need to press the reset button on the ESP32 mainboard first. In this way, the serial port monitor prints the detected WiFi IP address, then open the WiFi APP and enter the detected WiFi IP address in the text box in front of the WiFi button (for example, the IP address shown by the serial port monitor below :192.168.0.119), then click the WiFi button."403 Forbidden" or "Webpage not available" will change to "192.168.0.119", indicating that the APP is already

connected to WiFi.



Click each function button on the APP by hand, and then the serial port monitor will print the corresponding characters received. (If open the serial monitor and set the baud rate to 115200, the information is not displayed, please press the RESET button of the ESP32)

# media/1fd21fafd84d2b529931a89d21a03d6a.png

	/dev/ttyUSB0	~ ^ X )
		Send
Connected to ChinaNet-2.4G-0DF0 IP address: 192.168.0.119 TCP server started / Sending 200 /favicon.ico /favicon.ico /btn/0 Sending 200 /favicon.ico /btn/1 /btn/0		
Autoscroll 🗌 Show timestamp	Newline • 115200	baud 👻 Clear output

# 9.43 Project 40WiFi Smart Home

# 9.43.1 Introduction

In the previous experiment, we have learned the WiFi Station mode, WiFi AP mode and WiFi AP+Station mode of the ESP32. In this project, we will use ESP32's WiFi Station mode to control the work of multiple sensors/modules through APP connection with WiFi to achieve the effect of WiFi smart home.

# 9.43.2 Components



# 9.43.3 Wiring diagram

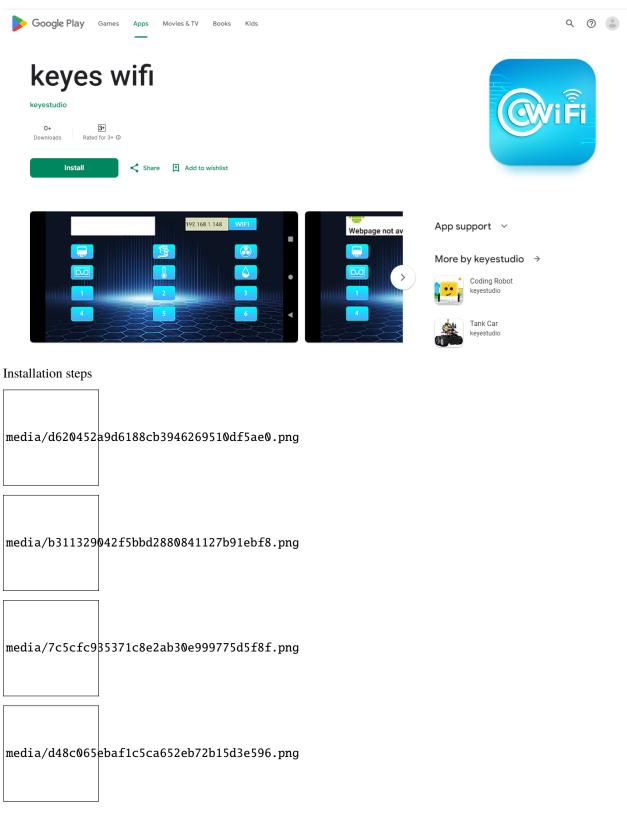
Relay Module	ESP32	Temperature and Humidity Sensor	ESP32
G	G	G	G
V	5V	V	3V3
S	IO32	S	IO15
Ultrasonic Sensor	ESP32	130 Motor	ESP32
Vcc	5V	G	G
Trig	IO14	V	5V
Echo	IO27	IN+	IO19
Gnd	G	IN-	IO18
Servo	ESP32		
Red line	5V		
Brown line	G		
Orange line	IO4		

media/fd7a7efd4bc365e1524011b68217dee5.png

(Note: Connect the wires and then install a small fan blade on the DC motor. )

# 9.43.4 Install APP:

Android system (Smartphone/Tablet) APP:Go to Google Store to search for keys wifi



media/78c89b91c0af2268f6267813e7923a9b.png

IOS system (Smartphone/Tablet) APP:

a.Open App Store.

media/27924fdb3d67692df7c63d8d0fb72287.png

b. Enter keyes link in the search box and click Search. The download interface appears. Click download and install the APP of keyes link. The following operations are similar to those of Android system, you can refer to the steps of Android system above for operation.

#### 9.43.5 Add the xht11 and ESP32Servo libraries

If you have not downloaded the library file, please click on the link to download it:Download Arduino Libraries

This code uses two libraries named "**xht11**" and "**ESP32Servo**", if you haven't installed them yet, please do so before learning. The steps to add third-party libraries are as follows:

#### 9.43.6 Project code

After the **xht11** and **ESP32Servo** libraries were added, You can open the code we provideIf you haven't downloaded the code file, please click on the link to download itDownload Arduino C Codes file

The code used in this project is saved in folder(path:) "Arduino-Codes\Project 40WiFi Smart Home\Project\_40\_WiFi\_Smart\_Home".

(continues on next page)

media/962a57f92b78eea1f0e3e81

```
#include "xht11.h"
//gpio15
xht11 xht(15);
unsigned char dht[4] = \{0, 0, 0, 0\};
#include <ESP32Servo.h>
Servo myservo;
int servoPin = 4;
#define Relay 32
#define IN1 19 //IN1 corresponds to IN+
#define IN2 18 //IN2 corresponds to IN-
#define trigPin 14
#define echoPin 27
int distance1;
String dis_str;
int ip_flag = 1;
int ultra_state = 1;
int temp_state = 1;
int humidity_state = 1;
String item = "0";
const char* ssid = "ChinaNet-2.4G-0DF0"; //the name of user's wifi
const char* password = "ChinaNet@233"; //the password of user's wifi
WiFiServer server(80);
String unoData = "";
void setup() {
  Serial.begin(115200);
  pinMode(Relay, OUTPUT);
  myservo.setPeriodHertz(50);
  myservo.attach(servoPin, 500, 2500);
  pinMode(IN1, OUTPUT);
  pinMode(IN2, OUTPUT);
  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED) {
   delay(500);
   Serial.print(".");
  }
  Serial.println("");
  Serial.print("Connected to ");
  Serial.println(ssid);
  Serial.print("IP address: ");
  Serial.println(WiFi.localIP());
  server.begin();
  Serial.println("TCP server started");
  MDNS.addService("http", "tcp", 80);
  digitalWrite(IN1, LOW);
  digitalWrite(IN2, LOW);
  digitalWrite(Relay, LOW);
```

```
pinMode(trigPin, OUTPUT);
 pinMode(echoPin, INPUT);
}
void loop() {
  WiFiClient client = server.available();
  if (!client) {
     return;
  }
  while(client.connected() && !client.available()){
      delay(1);
  }
  String req = client.readStringUntil('\r');
  int addr_start = req.indexOf(' ');
  int addr_end = req.indexOf(' ', addr_start + 1);
  if (addr_start == -1 || addr_end == -1) {
      Serial.print("Invalid request: ");
      Serial.println(req);
      return;
  }
  req = req.substring(addr_start + 1, addr_end);
  item=req;
  Serial.println(item);
  String s;
  if (req == "/")
  {
      IPAddress ip = WiFi.localIP();
      String ipStr = String(ip[0]) + '.' + String(ip[1]) + '.' + String(ip[2]) + '.' +...

→String(ip[3]);

      s = "HTTP/1.1 200 OK\r\nContent-Type: text/html\r\n\r\n<!DOCTYPE HTML>\r\n<html>
→Hello from ESP32 at ";
      s += ipStr;
      s += "</html>\r\n\r\n";
      Serial.println("Sending 200");
      client.println(s);
  }
  else if(req == "/btn/0")
  {
   Serial.write('a');
   client.println(F("turn on the relay"));
   digitalWrite(Relay, HIGH);
  }
  else if(req == "/btn/1")
  {
   Serial.write('b');
   client.println(F("turn off the relay"));
   digitalWrite(Relay, LOW);
  }
  else if(req == "/btn/2")
  ł
   Serial.write('c');
    client.println("Bring the steering gear over 180 degrees");
```

```
myservo.write(180);
 delay(200);
}
else if(req == "/btn/3")
{
 Serial.write('d');
 client.println("Bring the steering gear over 0 degrees");
 myservo.write(0);
 delay(200);
}
else if(req == "/btn/4")
ł
 Serial.write('e');
 client.println("esp32 already turn on the fans");
 digitalWrite(IN1, LOW);
 digitalWrite(IN2, HIGH);
}
else if(req == "/btn/5")
{
 Serial.write('f');
 client.println("esp32 already turn off the fans");
 digitalWrite(IN1, LOW);
 digitalWrite(IN2, LOW);
}
else if(req == "/btn/6")
{
 Serial.write('g');
 while(Serial.available() > 0)
  {
   unoData = Serial.readStringUntil('#');
    client.println("Data");
 }
 while(ultra_state>0)
     {
        Serial.print("Distance = ");
        Serial.print(checkdistance());
        Serial.println("#");
        Serial1.print("Distance = ");
        Serial1.print(checkdistance());
        Serial1.println("#");
        int t_val1 = checkdistance();
        client.print("Distance(cm) = ");
        client.println(t_val1);
        ultra_state = 0;
      }
}
else if(req == "/btn/7")
{
 Serial.write('h');
  client.println("turn off the ultrasonic");
 ultra_state = 1;
}
```

```
else if(req == "/btn/8")
{
 Serial.write('i');
 while(Serial.available() > 0)
   {
   unoData = Serial.readStringUntil('#');
   client.println(unoData);
  }
 while(temp_state>0)
    {
      if (xht.receive(dht)) {
        Serial.print("Temperature = ");
        Serial.print(dht[2],1);
        Serial.println("#");
        Serial1.print("Temperature = ");
        Serial1.print(dht[2],1);
        Serial1.println("#");
        int t_val2 = dht[2];
        client.print("Temperature(°C) = ");
        client.println(t_val2);
      }
      temp_state = 0;
    }
}
else if(req == "/btn/9")
{
 Serial.write('j');
 client.println("turn off the temperature");
  temp_state = 1;
}
else if(req == "/btn/10")
{
 Serial.write('k');
 while(Serial.available() > 0)
   {
     unoData = Serial.readStringUntil('#');
     client.println(unoData);
  }
 while(humidity_state > 0)
    {
      if (xht.receive(dht)) {
        Serial.print("Humidity = ");
        Serial.print(dht[0],1);
        Serial.println("#");
        Serial1.print("Humidity = ");
        Serial1.print(dht[0],1);
        Serial1.println("#");
        int t_val3 = dht[0];
        client.print("Humidity(%) = ");
        client.println(t_val3);
      }
     humidity_state = 0;
```

```
}
 }
 else if(req == "/btn/11")
 {
   Serial.write('1');
   client.println("turn off the humidity");
   humidity_state = 1;
   }
 //client.print(s);
 client.stop();
}
int checkdistance() {
 digitalWrite(14, LOW);
 delayMicroseconds(2);
 digitalWrite(14, HIGH);
 delayMicroseconds(10);
 digitalWrite(14, LOW);
 int distance = pulseIn(27, HIGH) / 58;
 delay(10);
 return distance;
}
```

**Special attention:** you need to change the user's Wifi name and Wifi password in the experiment code to your own Wifi name and Wifi password.

media/9ddee42d7e41abd8a6db60d447cd9f68.png

#### 9.43.7 Project result

After making sure that the Code is modified correctly, external power supply and power on, and then compile and upload the code to ESP32.Note: If uploading the code fails, you can press the Boot button on ESP32 after



pears)open the serial monitor and set baud rate to 115200.

You need to press the reset button on the ESP32 mainboard first. In this way, the serial port monitor prints the detected WiFi IP address,(If open the serial monitor and set the baud rate to 115200, the information is not displayed, please press the RESET button of the ESP32)

media/1fd21fafd84d2b529931a89d21a03d6a.png									
/dev/ttyUSB0									
					Send				
<pre>clk_drv:0x00,q_drv:0x00,d_drv:0x00,cs0_drv:0x00, mode:DIO, clock div:1 load:0x3fff0018,len:4 load:0x3fff001c,len:1216 ho 0 tail 12 room 4 load:0x40078000,len:10944 load:0x40080400,len:6388 entry 0x400806b4 E (62) psram: PSRAM ID read error: 0xffffffff  Connected to ChinaNet-2.4G-0DF0 IP address: 192.168.0.156 TCP server started</pre>	hd_drv:0x00,w	vp_drv∶	0x00						
✓ Autoscroll	Newline	• 11	15200 baud	▼ Clear	routput				

Then open the WiFi APP and enter the detected WiFi IP address in the text box in front of the WiFi button (for example, the IP address shown by the serial port monitor below :192.168.0.156), then click the WiFi button, "Hello from ESP32 at 192.168.0.156" is displayed in the text box next to the WiFi IP address, indicating that the APP is already connected to WiFi.(WiFi IP address sometimes changes, if the original IP address doesn't work, you need to re-check the WiFi IP address)

media/ac1bd20a153c3abc5c0c62a416446f52.jpeg

After the APP has been connected to WiFi, the following operations will be performed:

