keyestudio WiKi

keyestudio WiKi

Dec 06, 2023

KEYESTUDIO DOCS

CHAPTER

ONE

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.

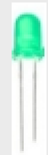
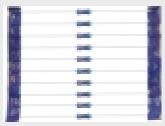
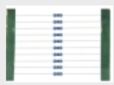
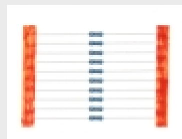
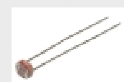
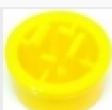
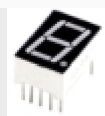
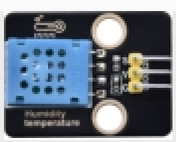
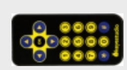
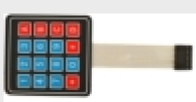
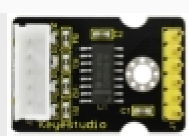
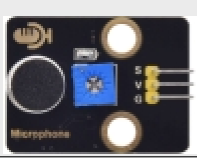
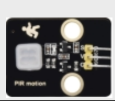
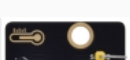
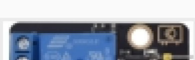
As many as 117 project tutorials are provided, which contain detailed wiring diagrams, components knowledge, and fascinating project code. Each project is produced using Thonny for Windows, Arduino IDE for Windows, and Arduino IDE for Raspberry Pi. It's easy to get started.

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).

CHAPTER TWO

2.KIT LIST

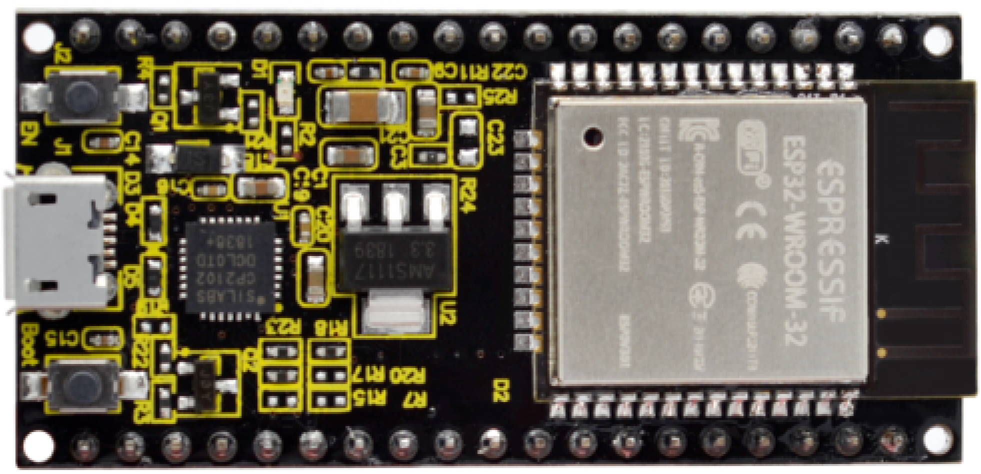
When you received the kit , the first thing you see is a beautiful packaged box. Each accessory was safely and orderly packed in a small bag. Let's check them first:

				
ESP32 Mainboard*1	Blue LED*10	Red LED*10	Yellow LED*10	Green LED*10
				
RGB*1	220Resistor*10	10KResistor*10	1KResistor*10	10K Potentiometer*1
				
Active Buzzer*1	Passive Buzzer*1	Button*4	Tilt Switch*1	Photoresistor*2
				
Flame Sensor*1	Yellow Cap*4	IC 74HC595N *1	1-Digit Tube Display*1	4-Digit Tube Display*1
				
8*8 Dot Matrix Display *1	Temperature and Humidity Sensor*1	LCD_128X *1	IR Receiver*1	IR Remote Controller*1
				
Servo*1	*44 Membrane Keypad*1	130 DC Motor*1	Stepper Motor Driver Board*1	Stepper Motor*1
				
Joystick Module*1	Sound Sensor*1	PIR Motion Sensor*1	RFID Module*1	Ultrasonic Sensor*1
				

CHAPTER

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 Current80mA*Average*

Current Supply500mA*Minimum*

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

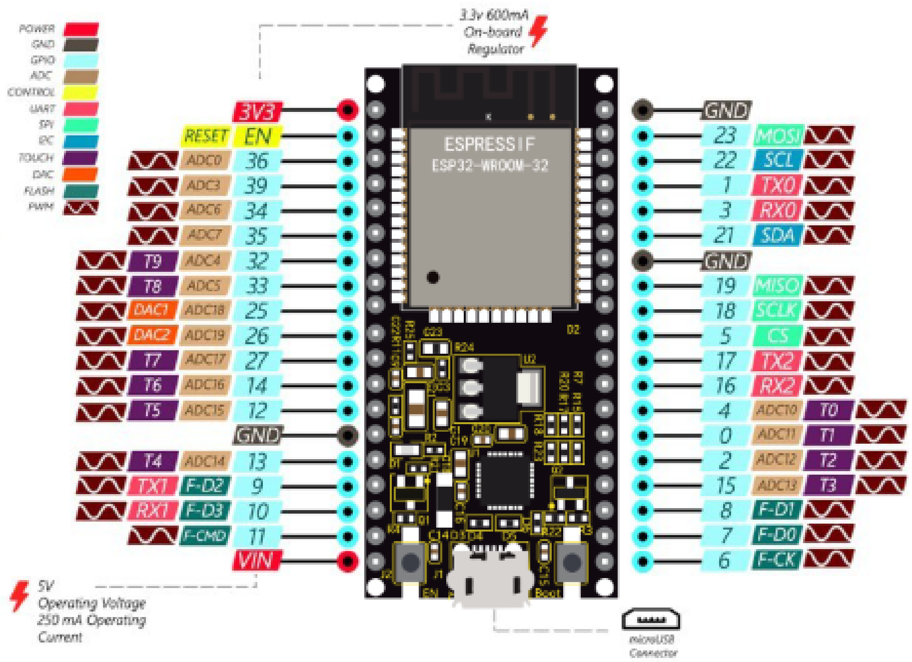
WiFi Frequency Range2.4 GHz ~ 2.5 GHz

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

Dimensions55*26*13mm

Weight9.3g

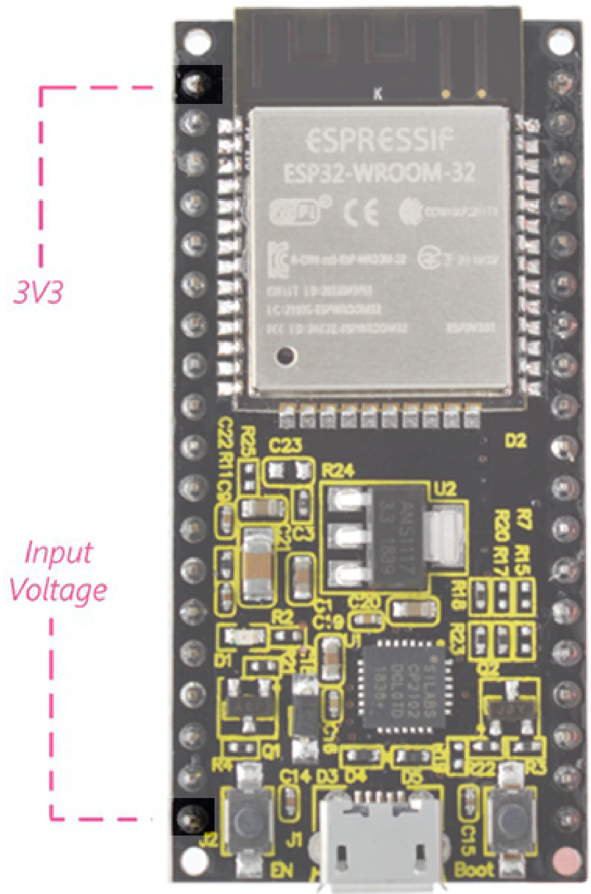
Pin out



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.



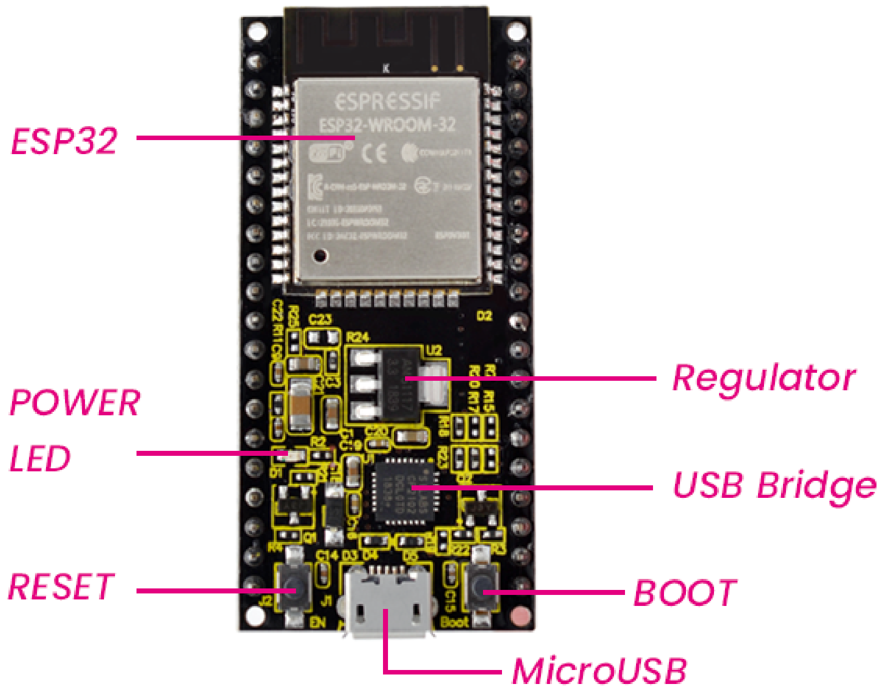
** GND Pins** The module has three grounded pins.
Enable pin (EN) : This pin is used to enable and disable modules. The pin enables module at high level and disables module at low level.

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 UART0 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

CHAPTER

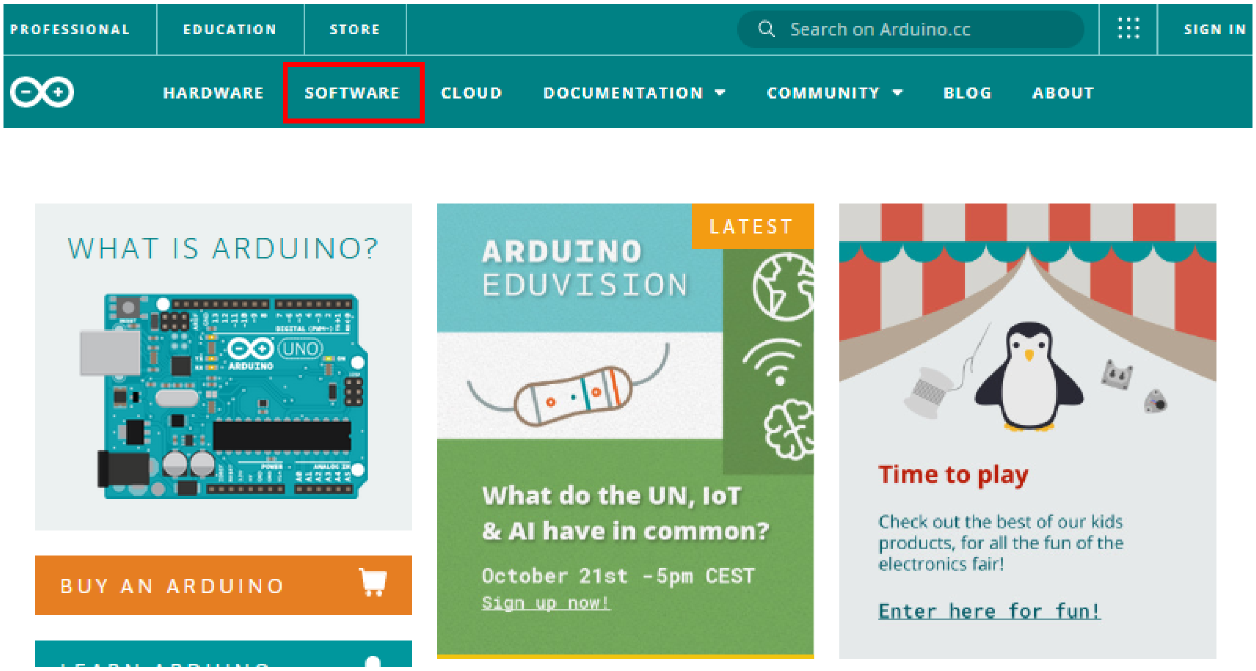
FOUR

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

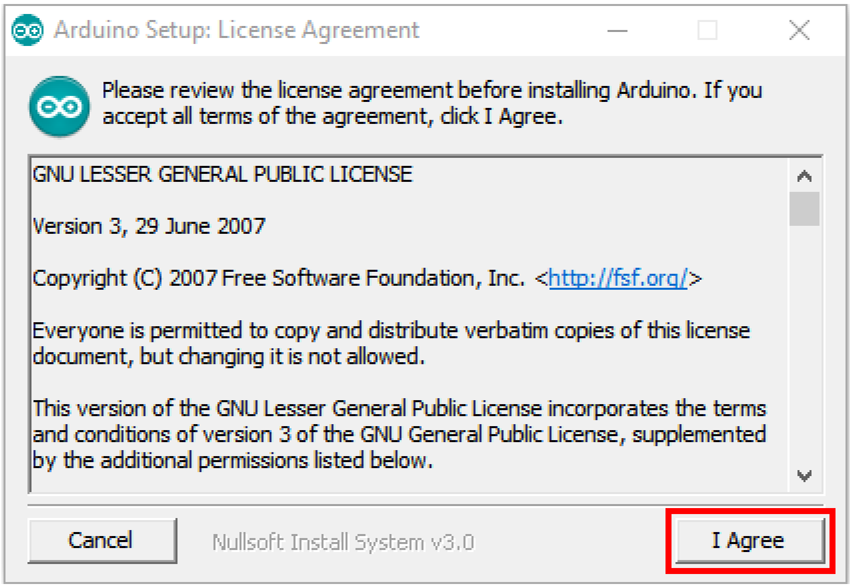


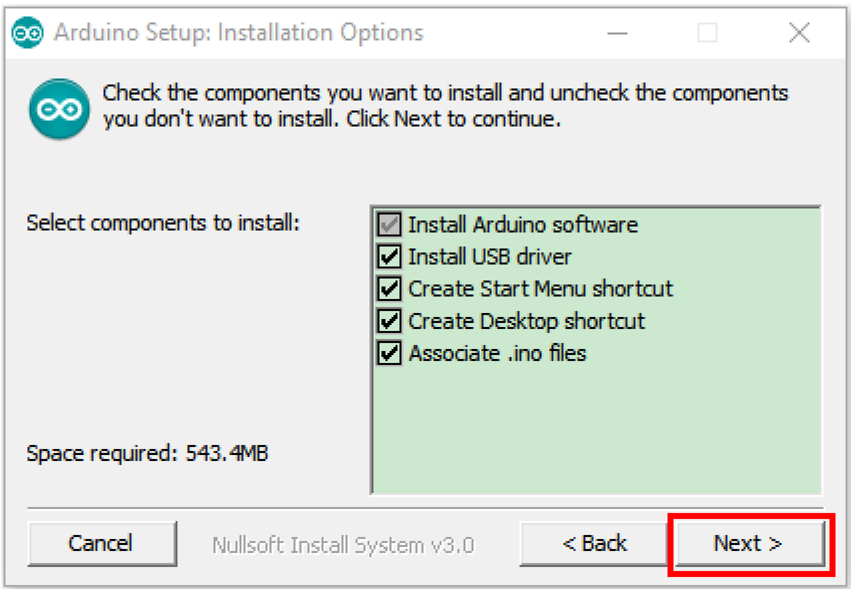
2Then, select and download the corresponding installer for your operating system. If you are a Windows user, please select "Windows Installer" to download to install the driver correctly.

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.

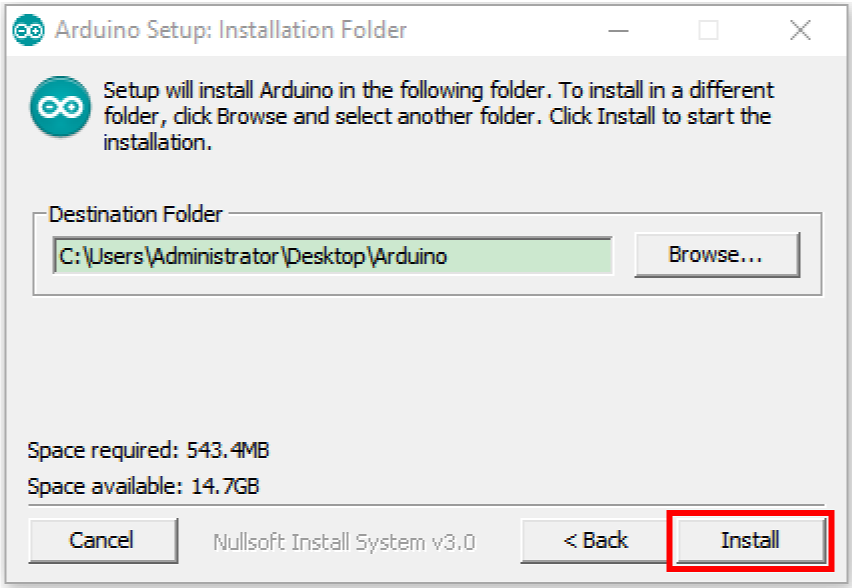
In general, you can click **JUST DOWNLOAD** to download it, although if you like, you can choose a small sponsorship to help the great Arduino open source cause.

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.

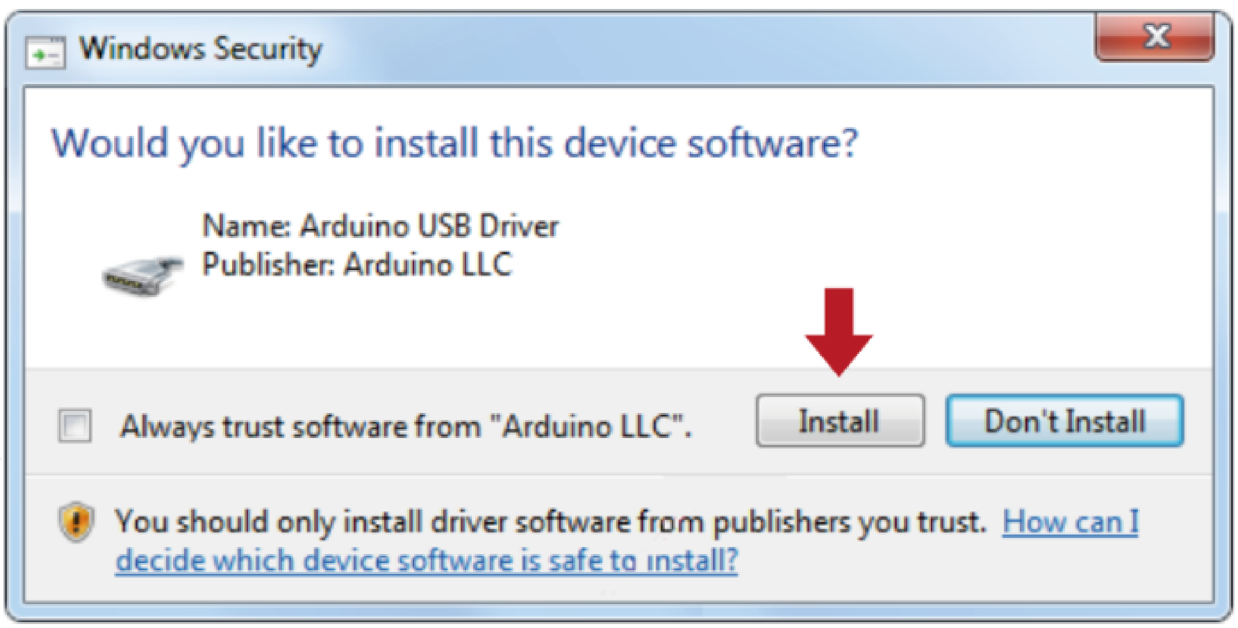




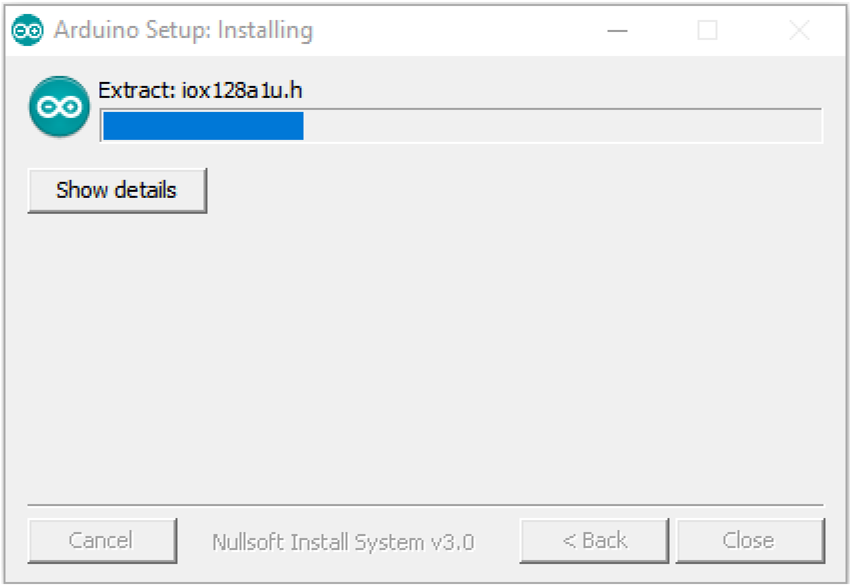
4Select the installation directory (we recommend keeping the default directory), and then click **Install**.



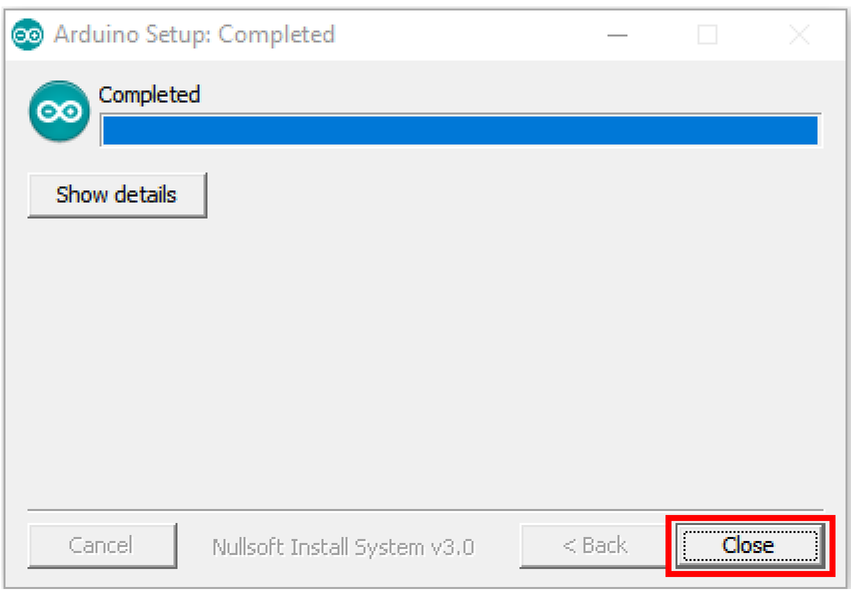
5Select Install if the following screen appears.



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



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



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.

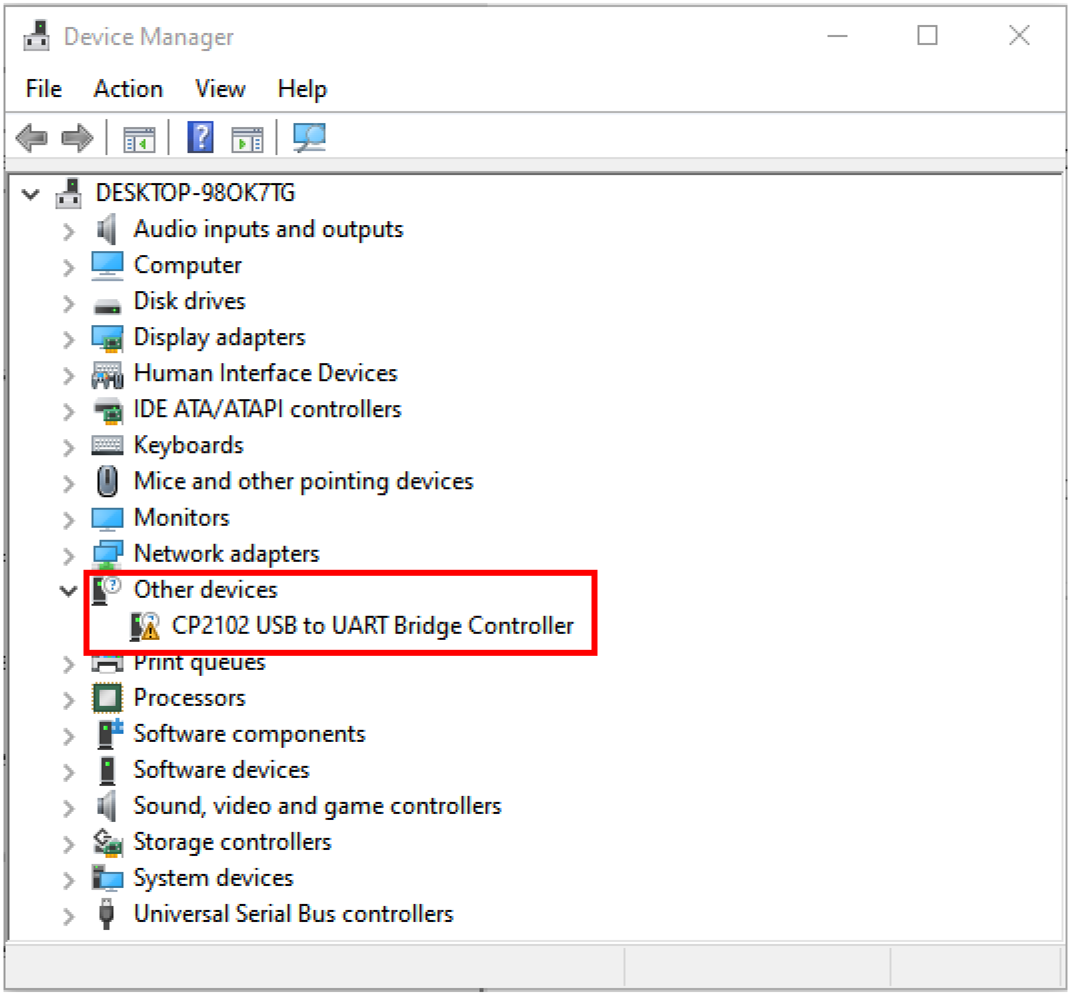
Note:

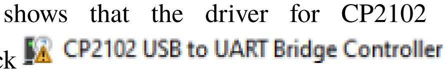
1. Please make sure that your IDE is updated to 1.8.0 or newer version

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

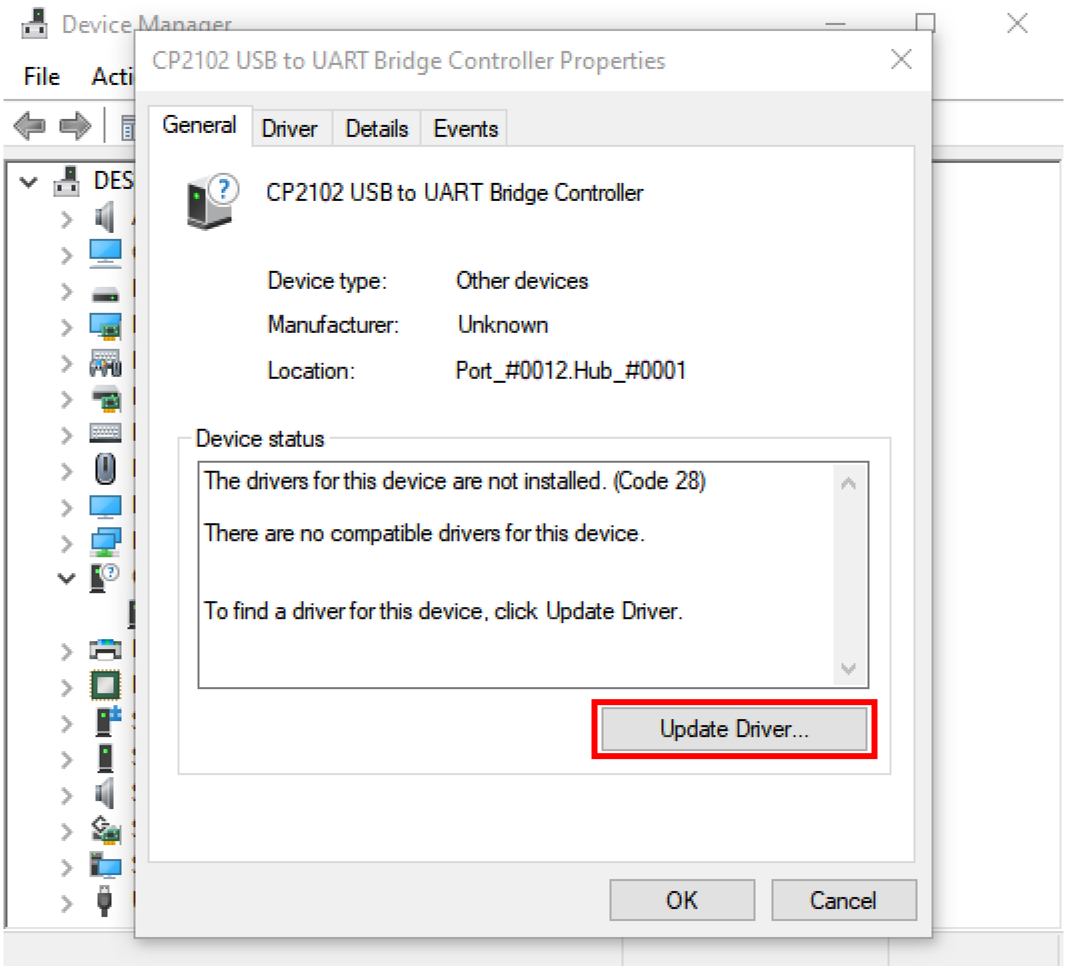
Link to download the driver of CP2102:Download CP2102 Driver

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

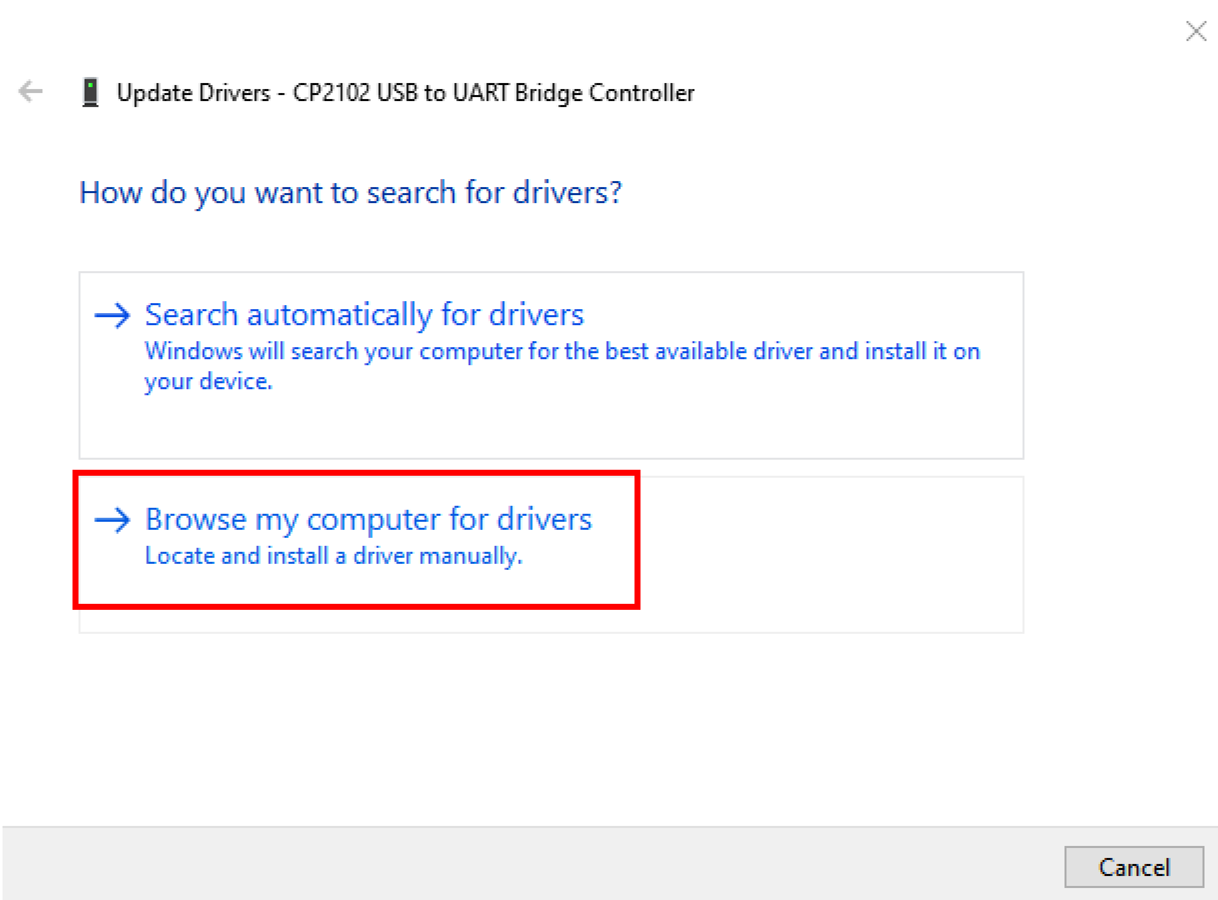


It shows that the driver for CP2102 was not installed sucessfully with a yellow mark. Double-click 

, and then click "**Update drive…**" to update the driver.

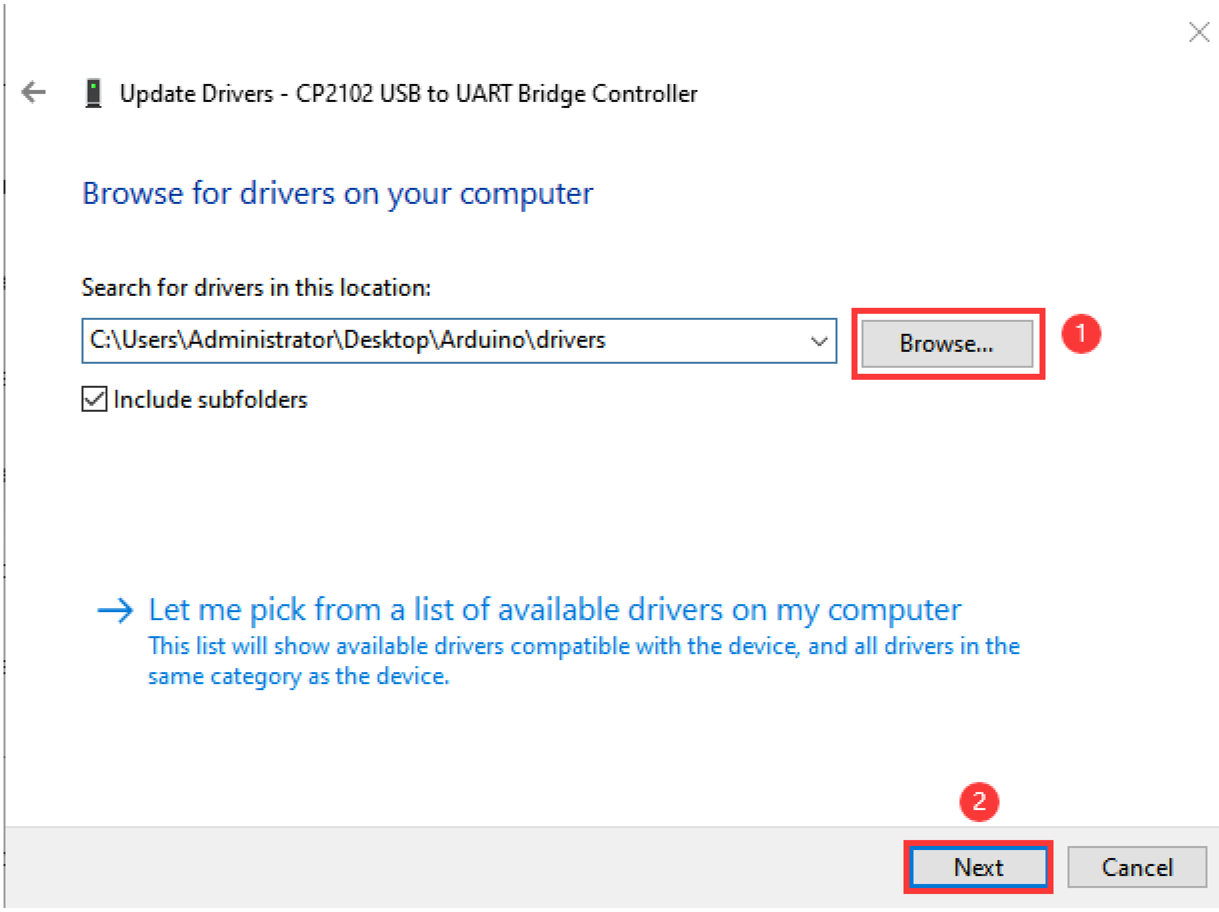


Click “**Browse my computer for drivers**”and find the Arduino software we installed or downloaded.

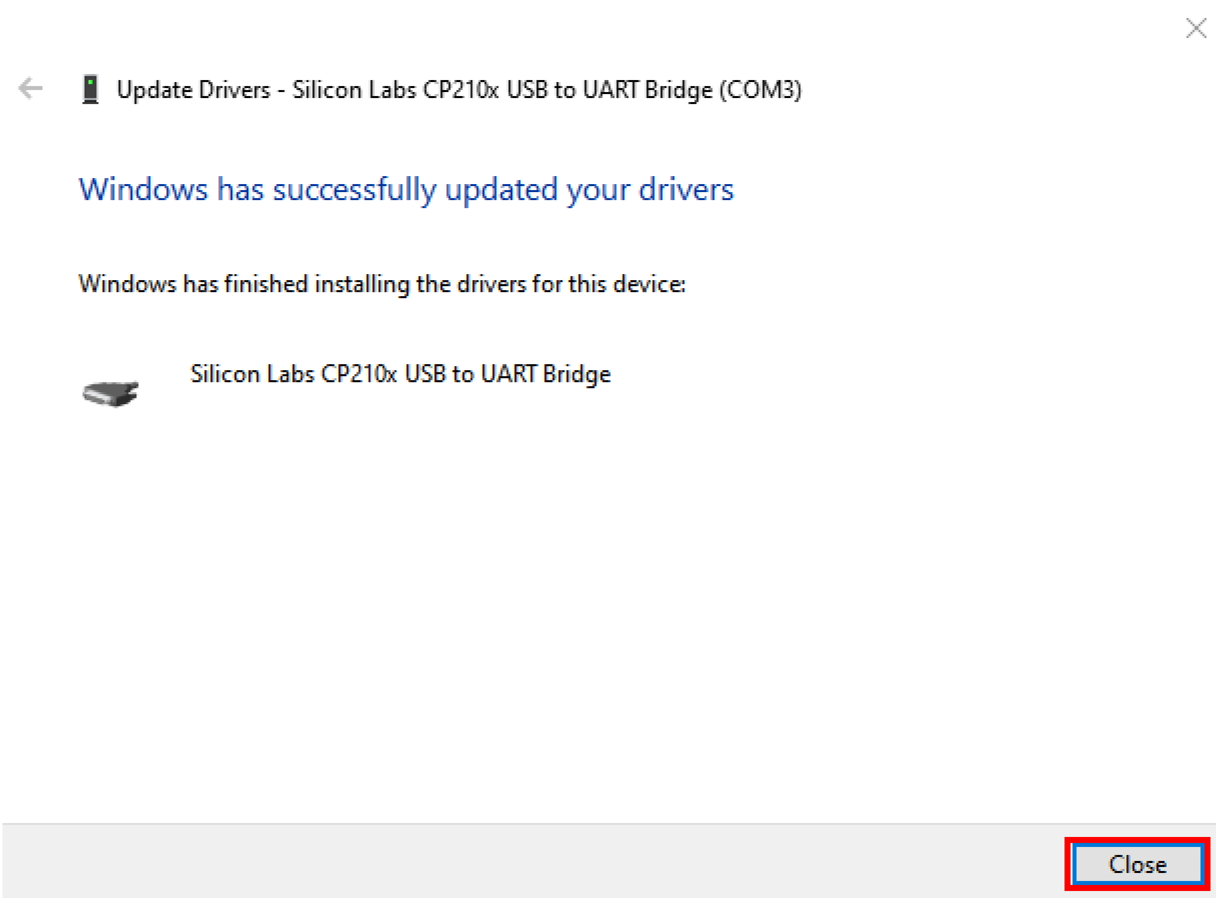


There is a **drivers** folder in Arduino software installed package Arduino), open driver folder and you can see the driver of CP210X series chips.

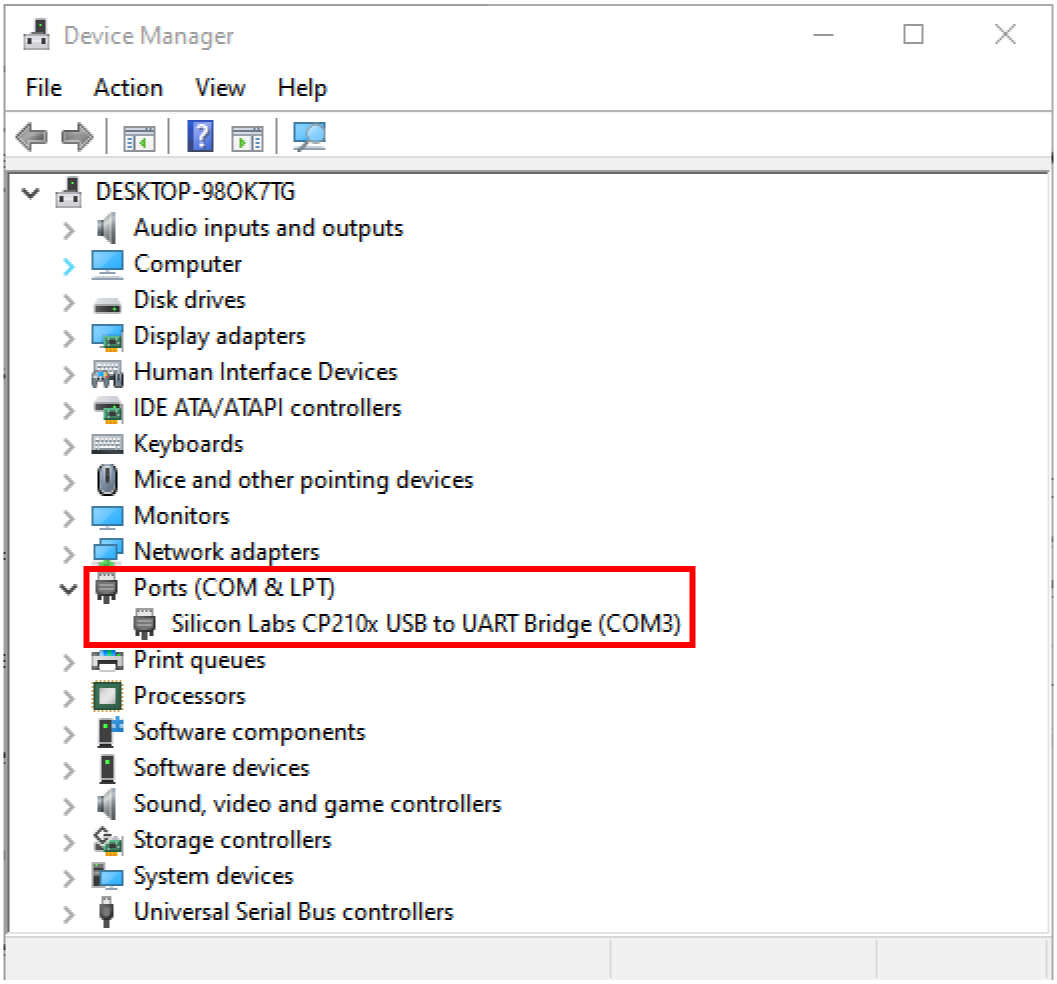
Click"**Browse…**", then find the drivers folder, or you could enter"driver"to search in rectangular box, then click"**Next**".



After a while, the driver is installed successfully.



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

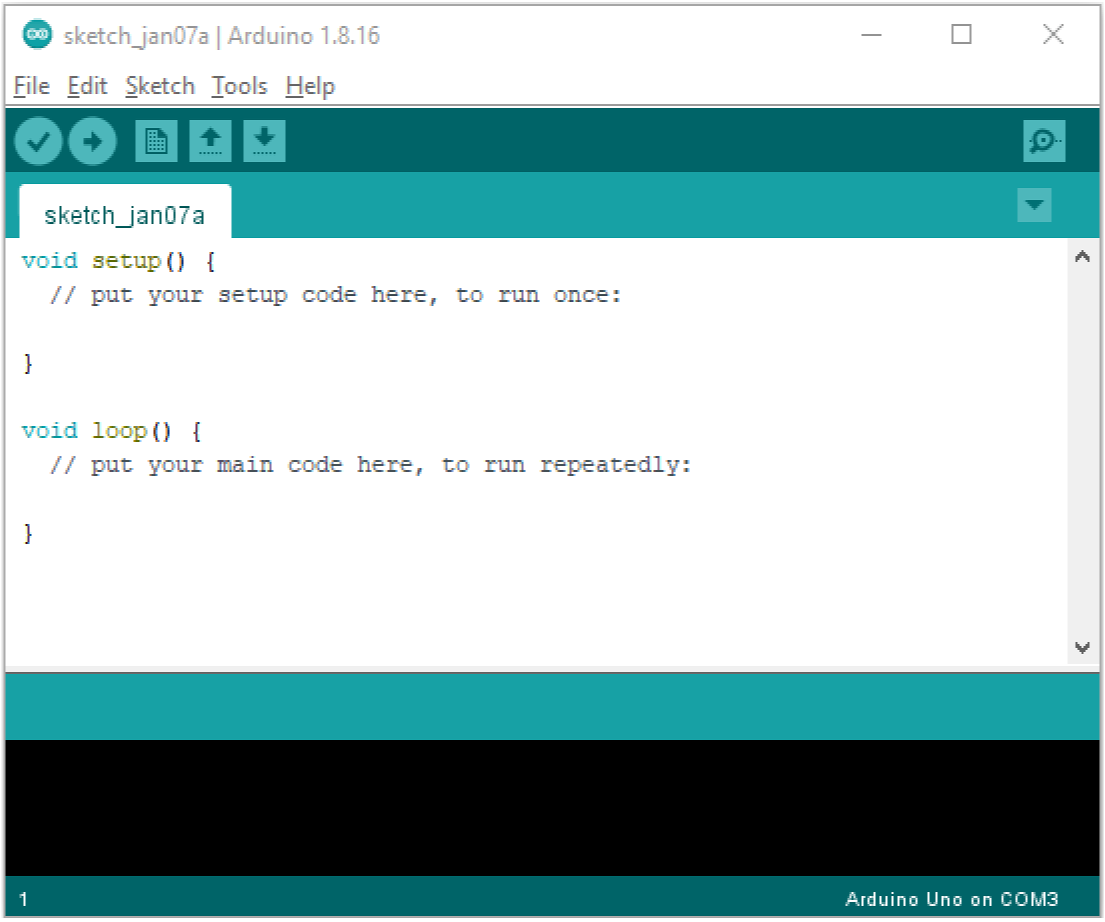


4.1.3 1.3. Install the ESP32 on Arduino IDE

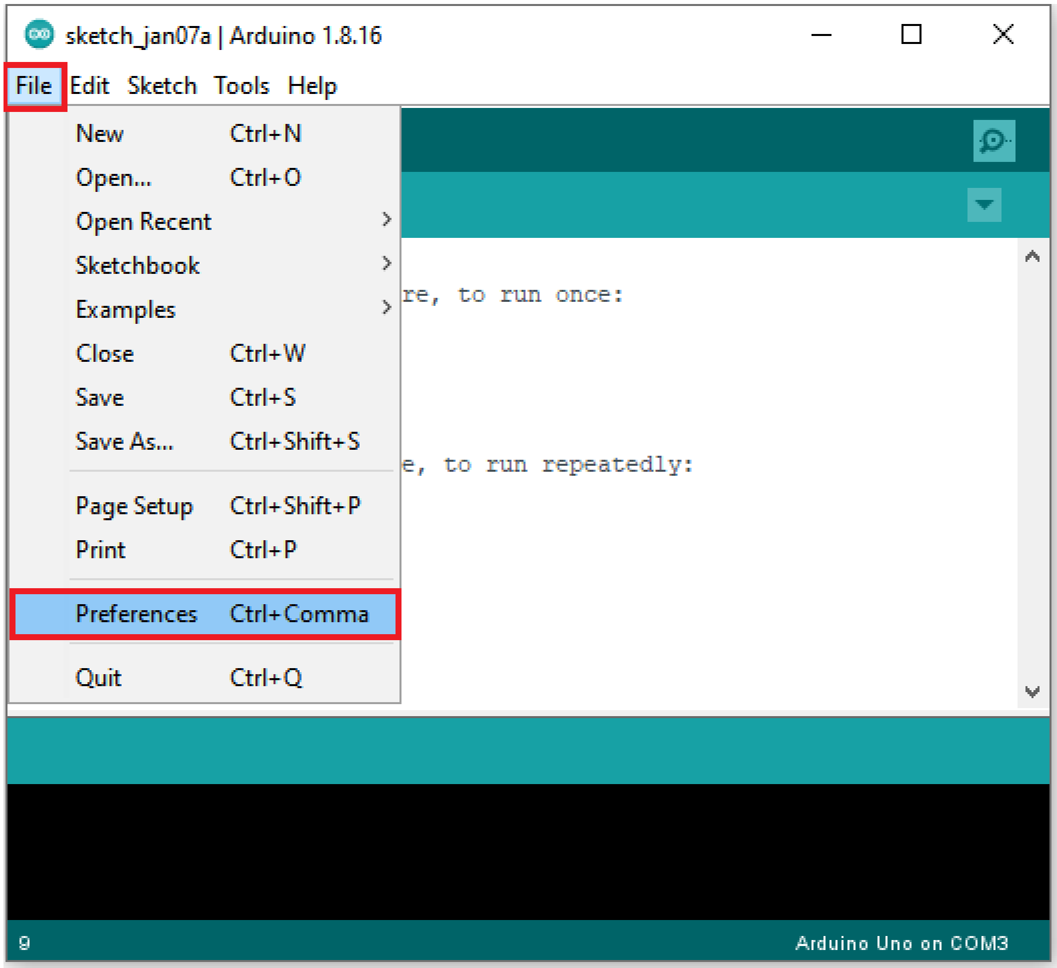
The installation process for ESP32 is almost the same as that for ESP8266. To install ESP32 on an Arduino IDE, follow these steps Noteyou need to download Arduino IDE 1.8.5 or advanced version to install the ESP32.

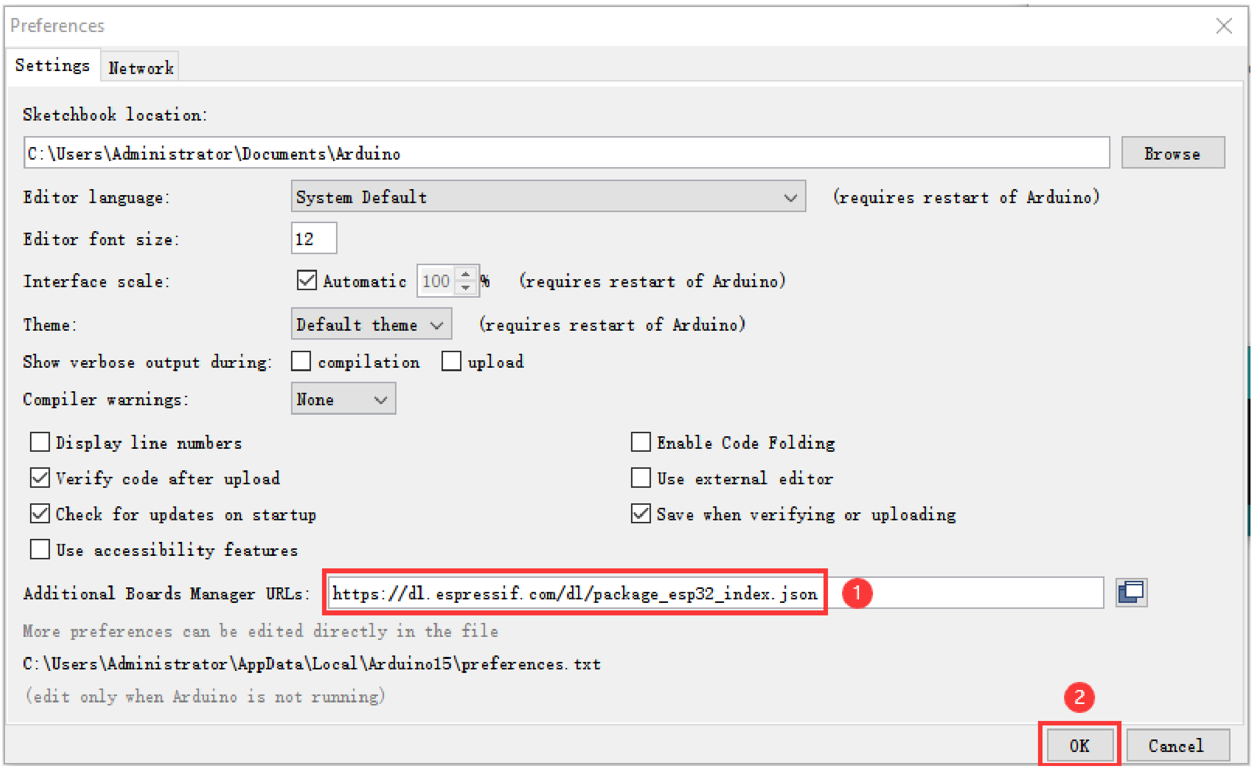
(1) Click the icon 

to open the Arduino IDE.

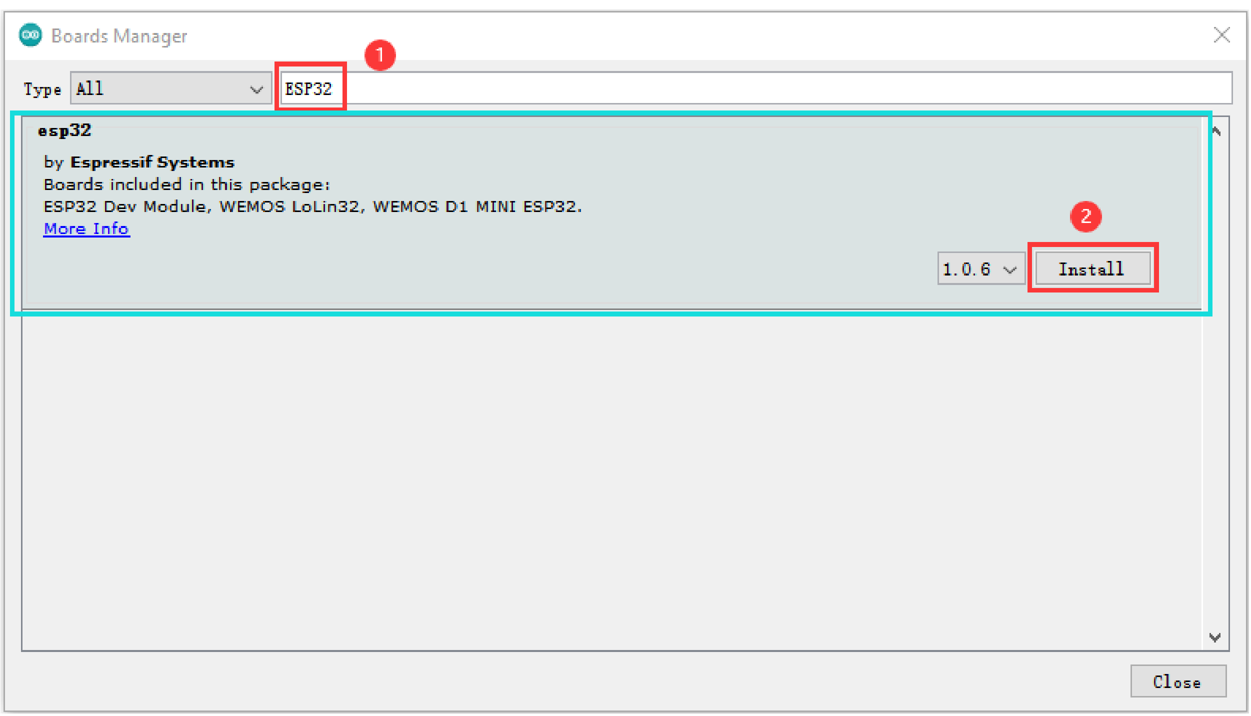


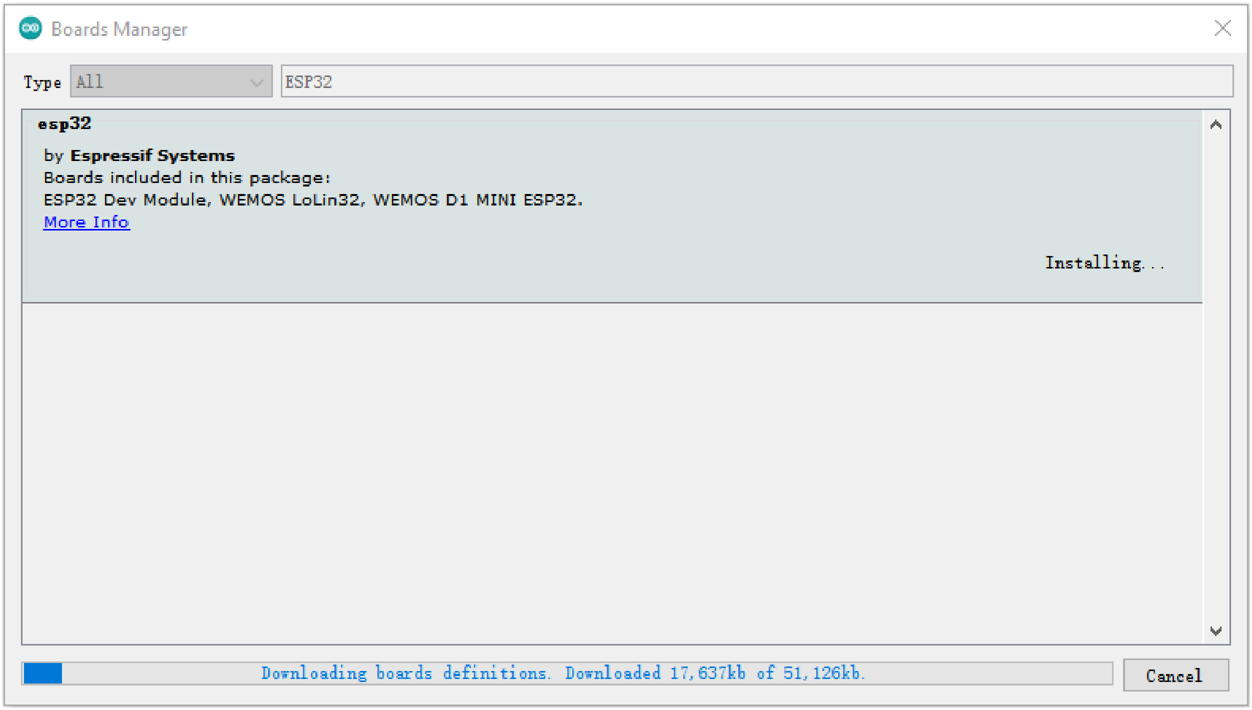
2Click“**File**” →**“Preferences”**copy the website addresshttps://dl.espressif.com/dl/package_esp32_index.json in the “**Additional Boards Manager URLs:**”and then click“**OK**” to save the address.


sketch_jan07a | Arduino 1.8.16
File Edit Sketch Tools Help
New Ctrl+N
Open... Ctrl+O
Open Recent
Sketchbook
Examples
Close Ctrl+W
Save Ctrl+S
Save As... Ctrl+Shift+S
Page Setup Ctrl+Shift+P
Print Ctrl+P
Preferences Ctrl+Comma
Quit Ctrl+Q
re, to run once:
e, to run repeatedly:
9
Arduino Uno on COM3



3First click "**Tools**"→"**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.



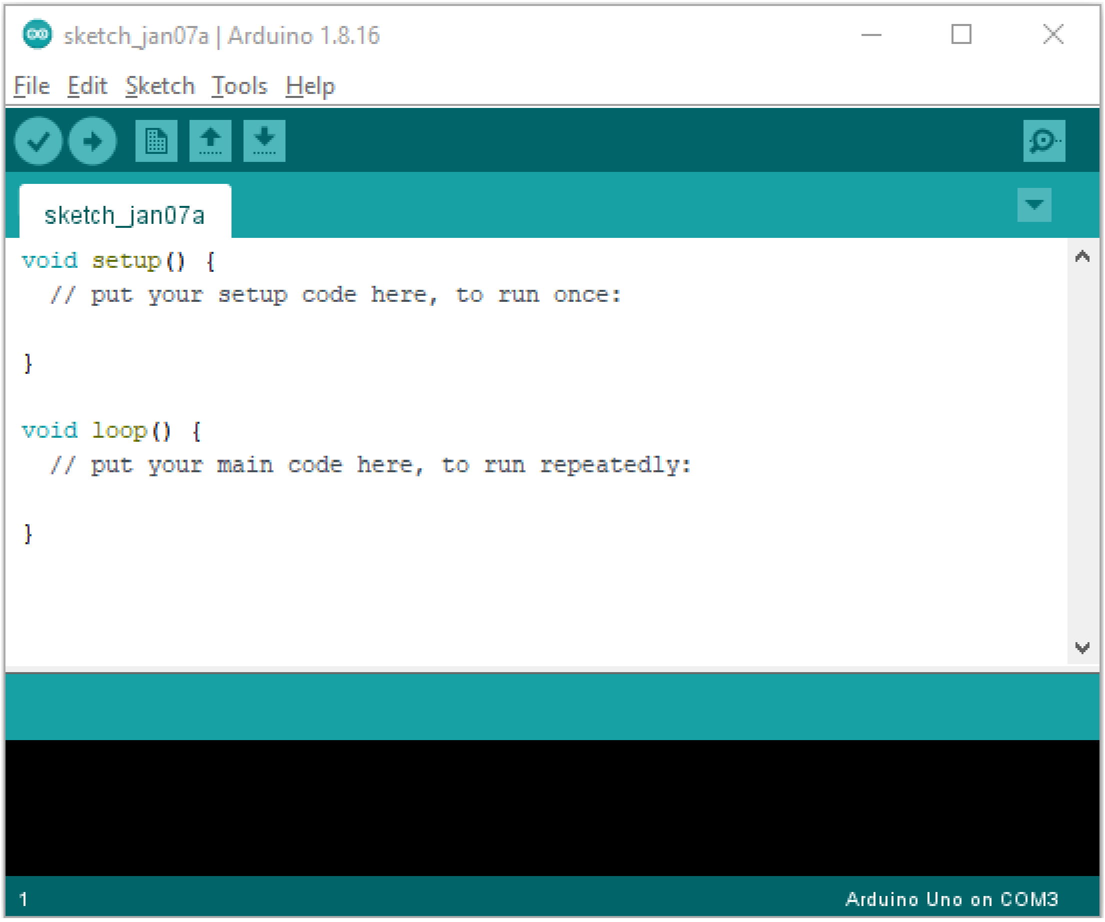


After successful installation, click "**Close**" to Close the page

4.1.4 1.4. Arduino IDE Setting:

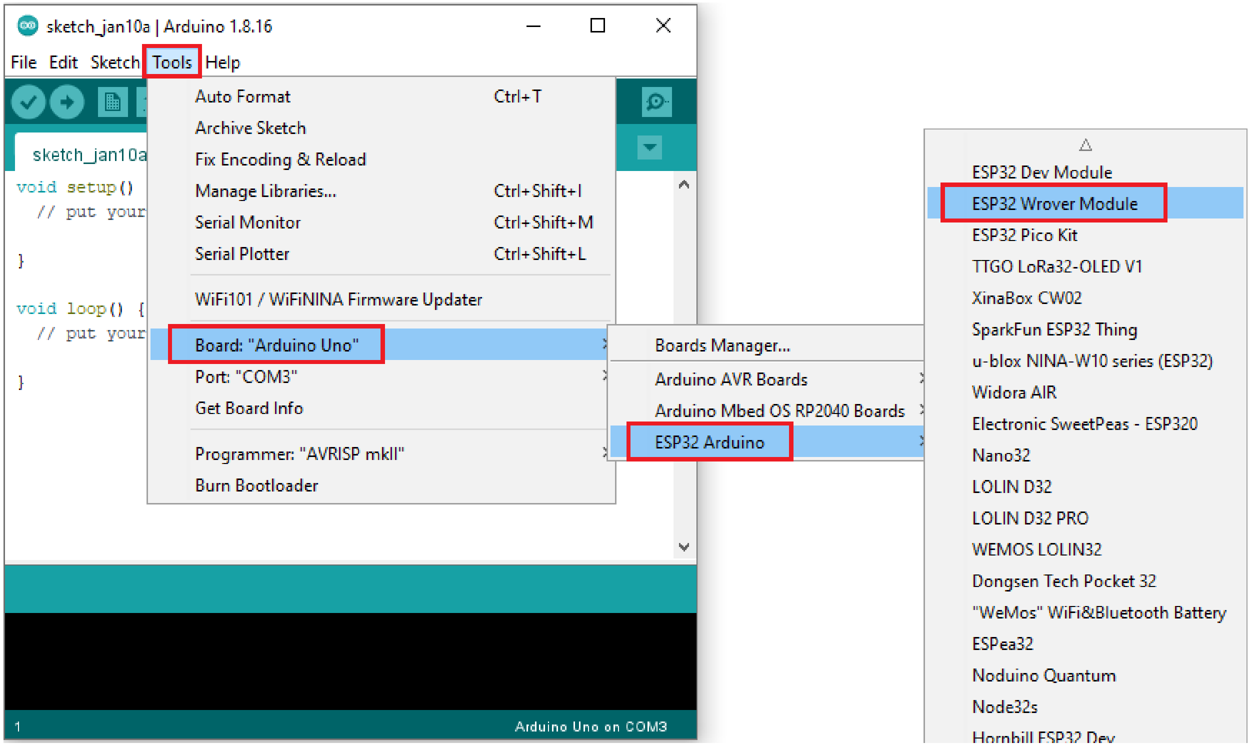
1Click the icon

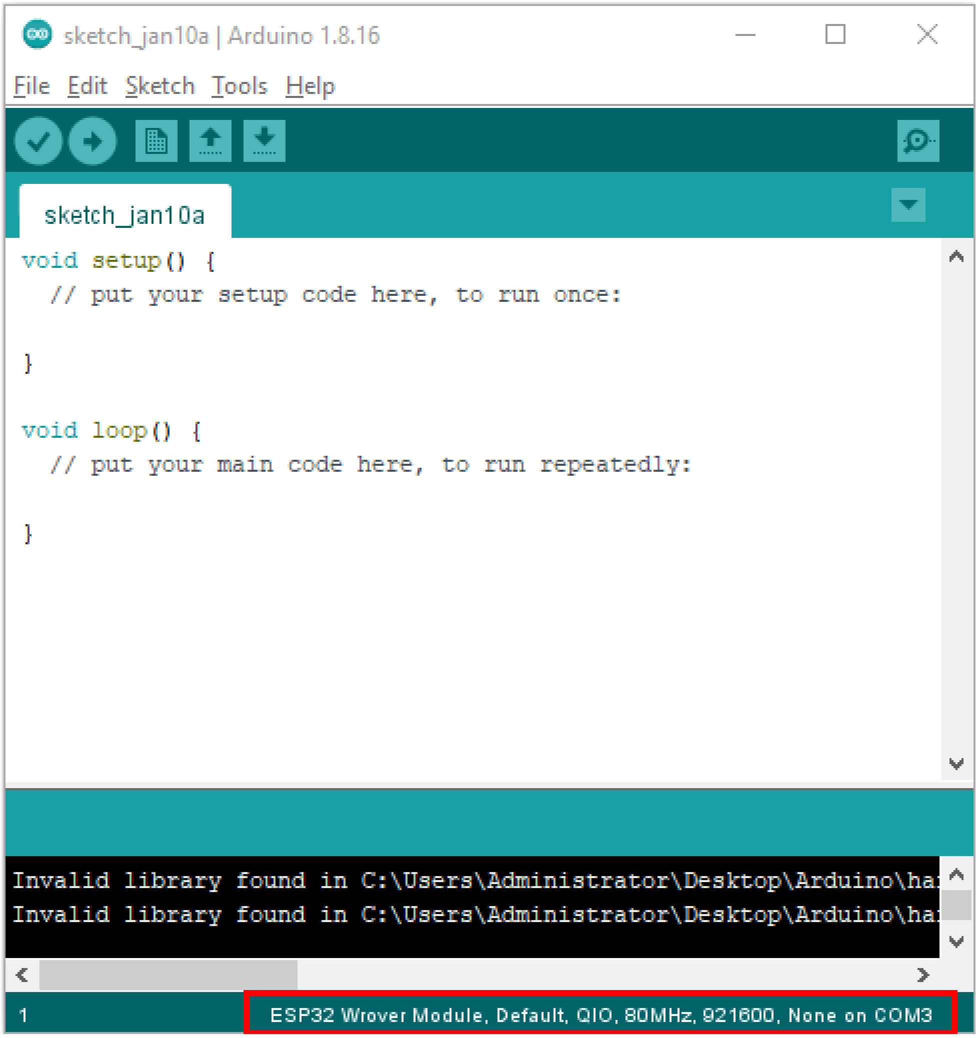
to open the Arduino IDE.



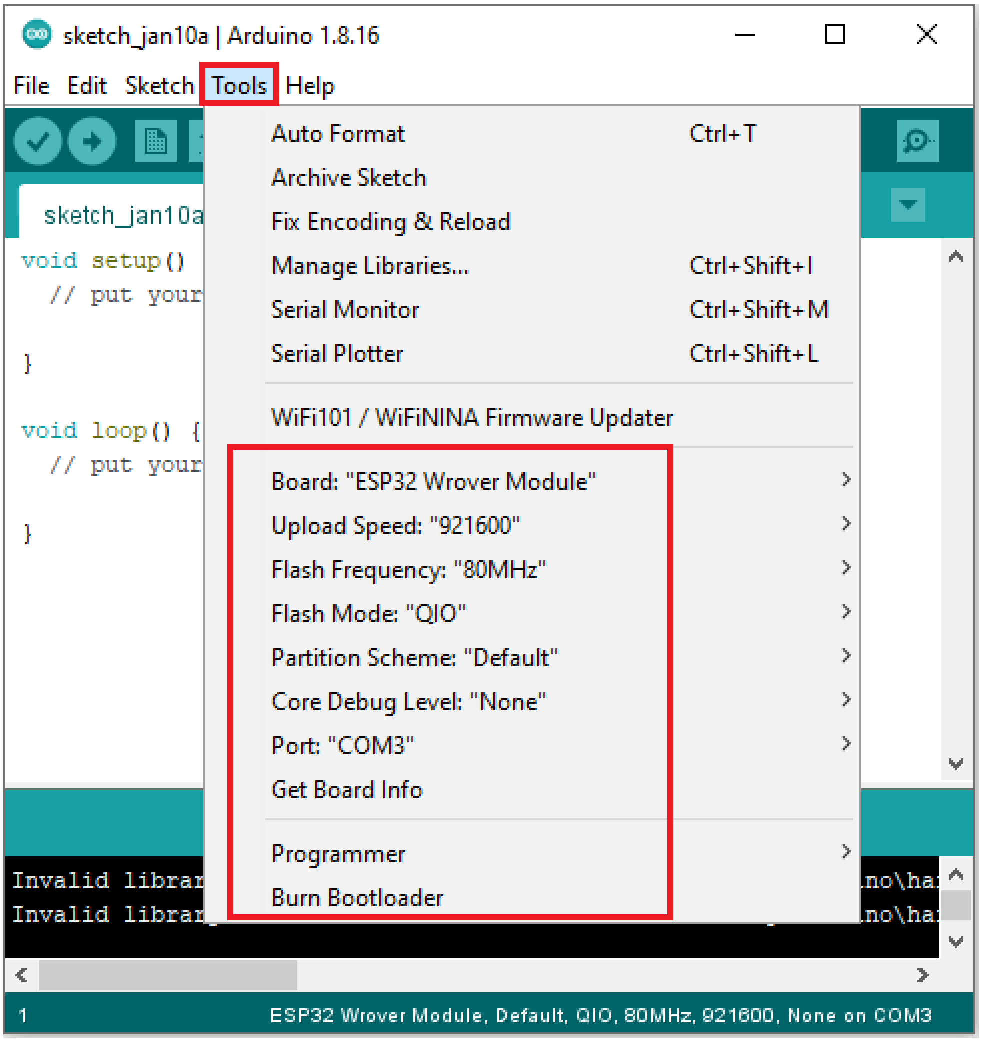
2When downloading the code to the board, you must select the correct name of Arduino board that matches the board connected to your computer,click"**Tools**"→"**Board:**". As shown below ;

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

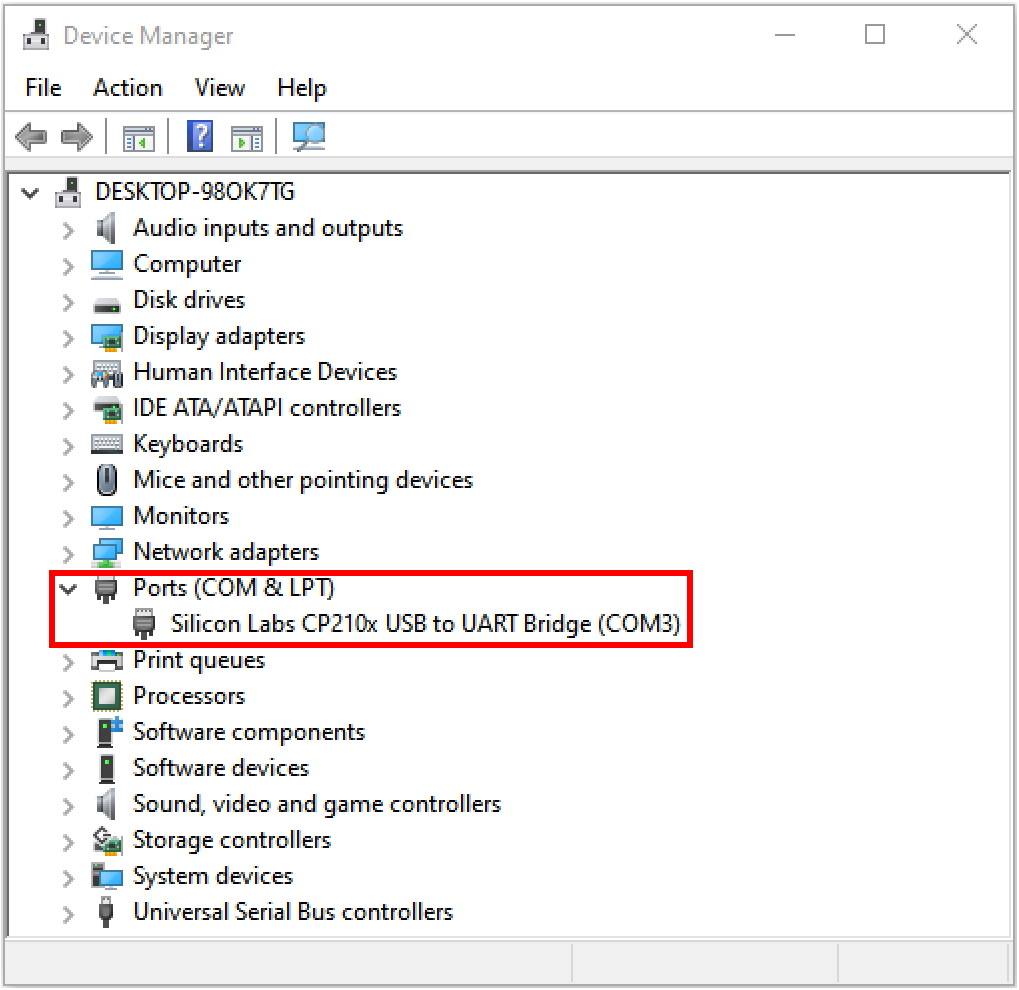

sketch_jan10a | Arduino 1.8.16
File Edit Sketch Tools Help
Auto Format Ctrl+T
Archive Sketch
Fix Encoding & Reload
Manage Libraries... Ctrl+Shift+I
Serial Monitor Ctrl+Shift+M
Serial Plotter Ctrl+Shift+L
WiFi101 / WiFiNINA Firmware Updater
Board: "Arduino Uno"
Port: "COM3"
Get Board Info
Programmer: "AVRISP mkII"
Burn Bootloader
Boards Manager...
Arduino AVR Boards
Arduino Mbed OS RP2040 Boards
ESP32 Arduino
ESP32 Dev Module
ESP32 Wrover Module
ESP32 Pico Kit
TTGO LoRa32-OLED V1
XinaBox CW02
SparkFun ESP32 Thing
u-blox NINA-W10 series (ESP32)
Widora AIR
Electronic SweetPeas - ESP320
Nano32
LOLIN D32
LOLIN D32 PRO
WEMOS LOLIN32
Dongsen Tech Pocket 32
"WeMos" WiFi&Bluetooth Battery
ESPea32
Noduino Quantum
Node32s
Arduino Uno on COM3

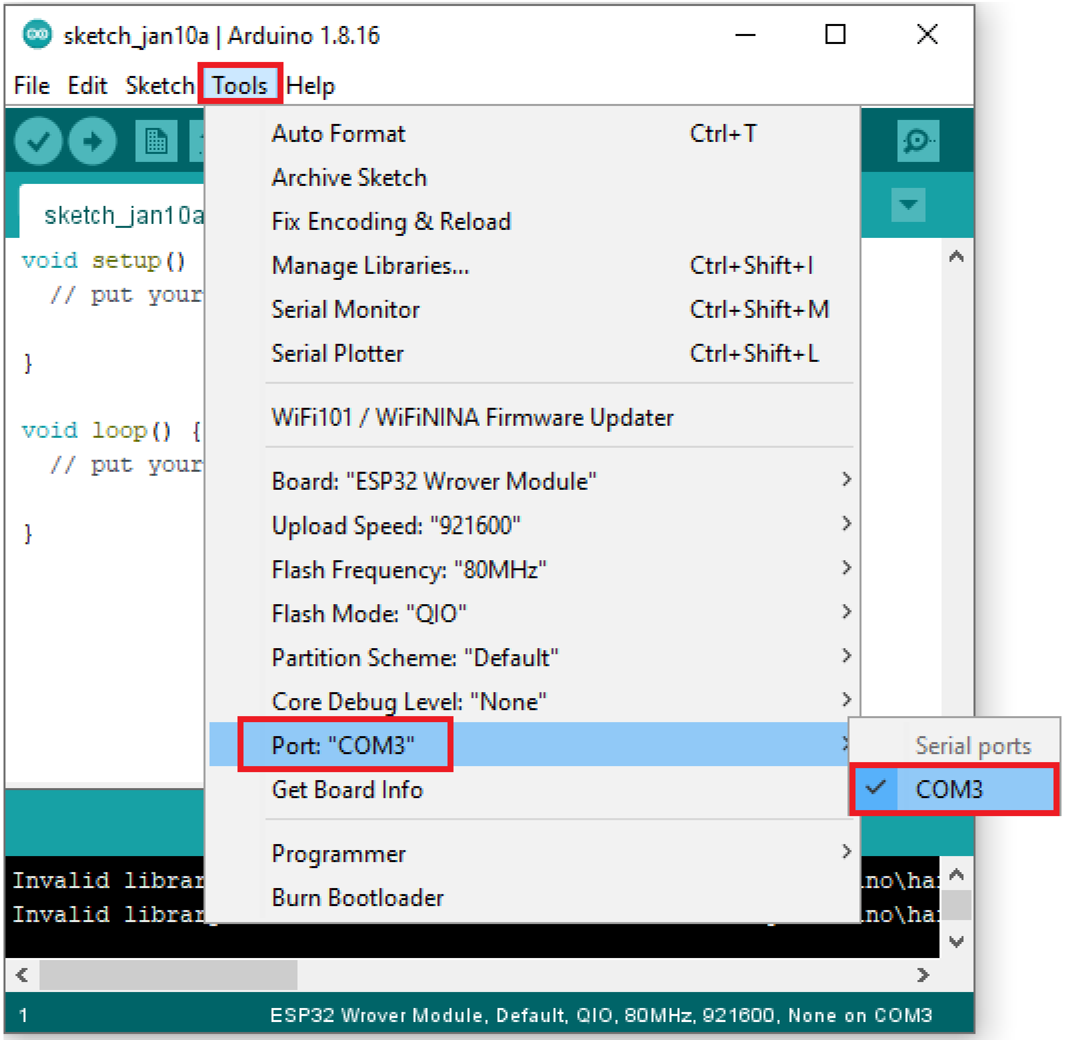


Set the board type as follows:

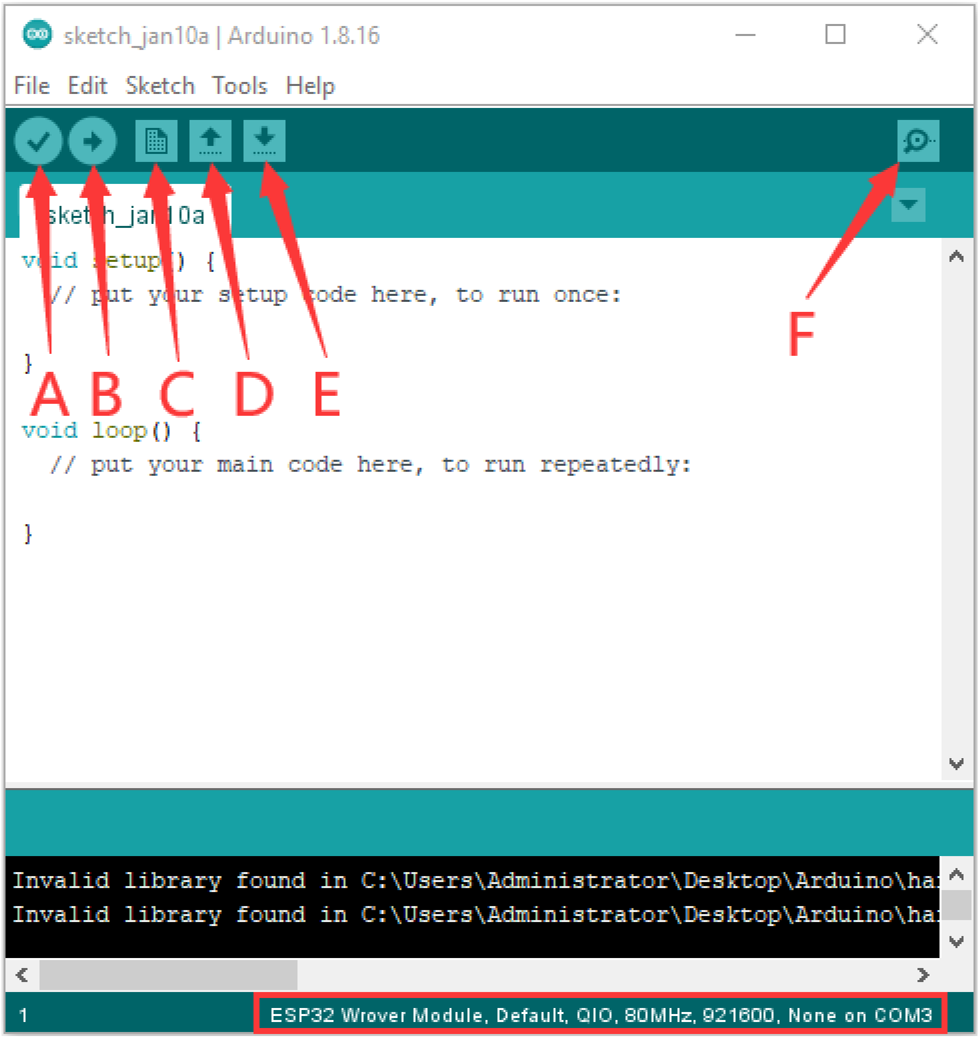


Then select the correct COM port (the corresponding COM port can be seen after the driver is installed successfully).


Device Manager
File Action View Help
DESKTOP-98OK7TG
Audio inputs and outputs
Computer
Disk drives
Display adapters
Human Interface Devices
IDE ATA/ATAPI controllers
Keyboards
Mice and other pointing devices
Monitors
Network adapters
Ports (COM & LPT)
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.

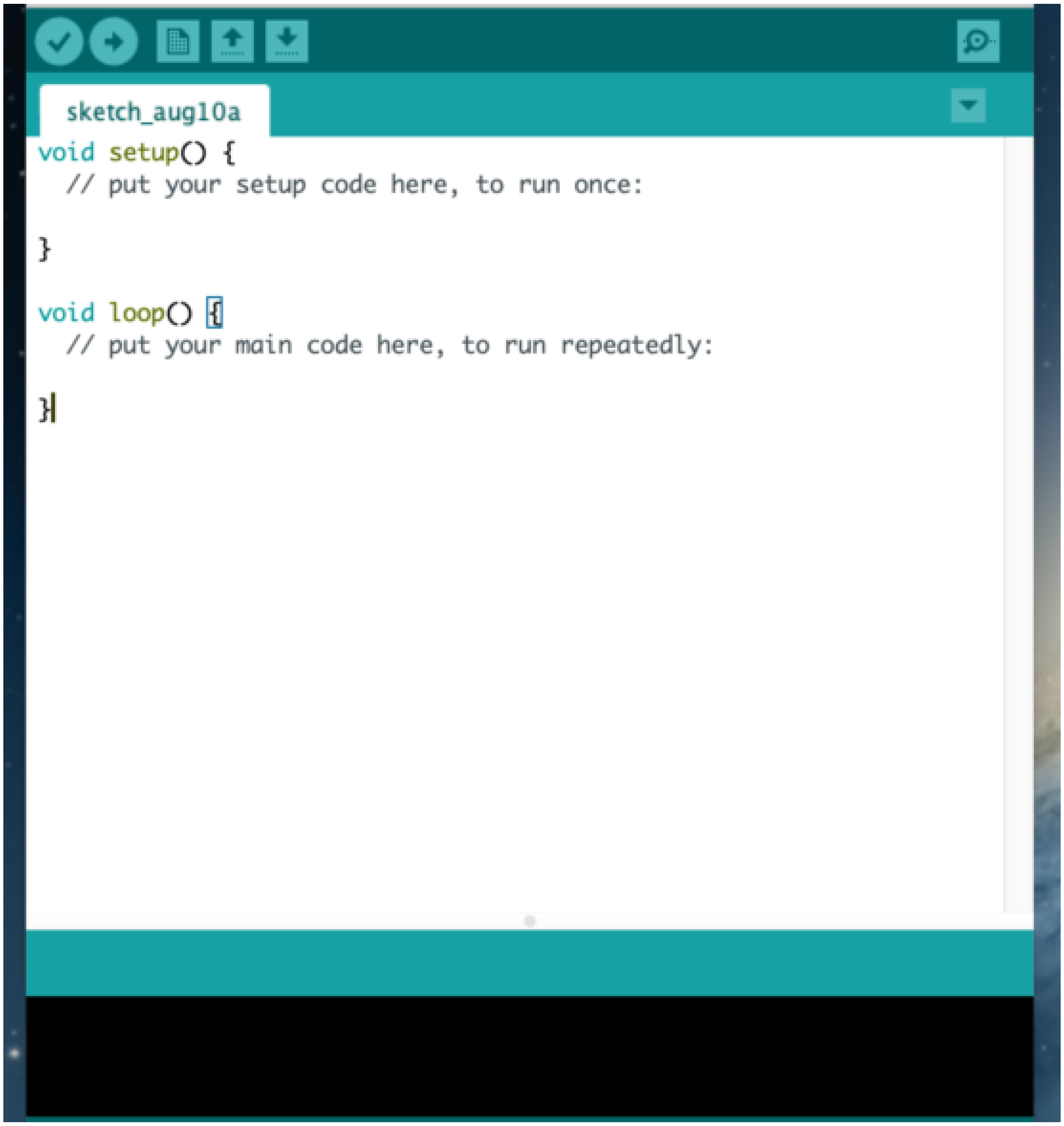
4.2 Mac System:

4.2.1 2.1.Download and install the Arduino IDE:

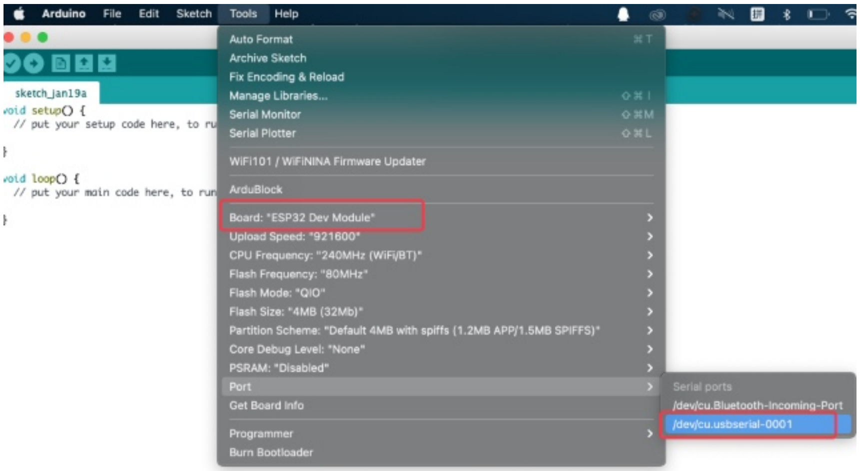
4.2.2 2.2.How to install the CP2102 driver

NoteIf 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"**.

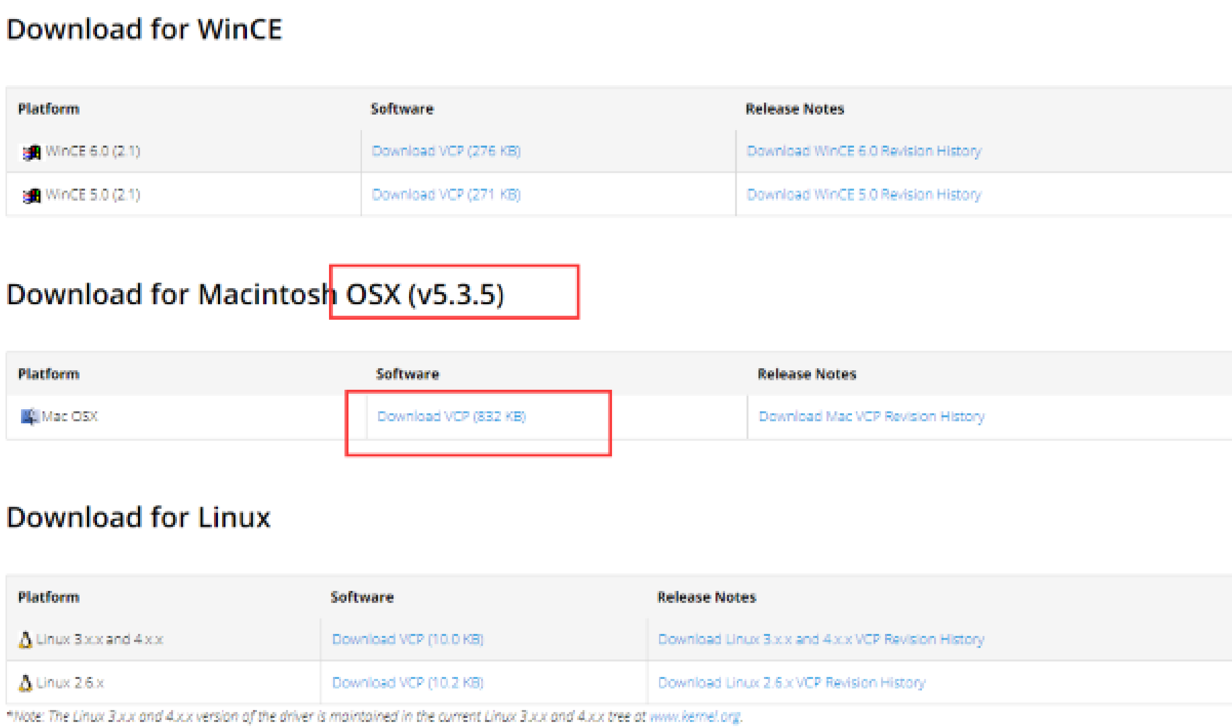


Click  to upload code.

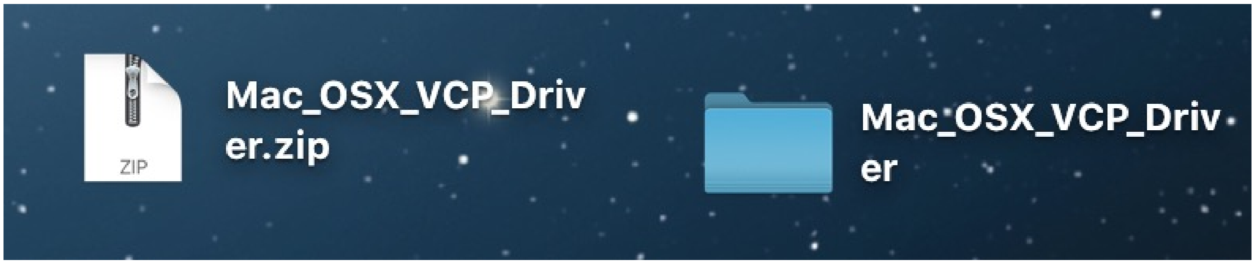
Note: If code is uploaded unsuccessfully, you need to install driver of CP2102, please continue to follow the instructions as below:

2Download the driver of CP2102Download CP2102 Driver

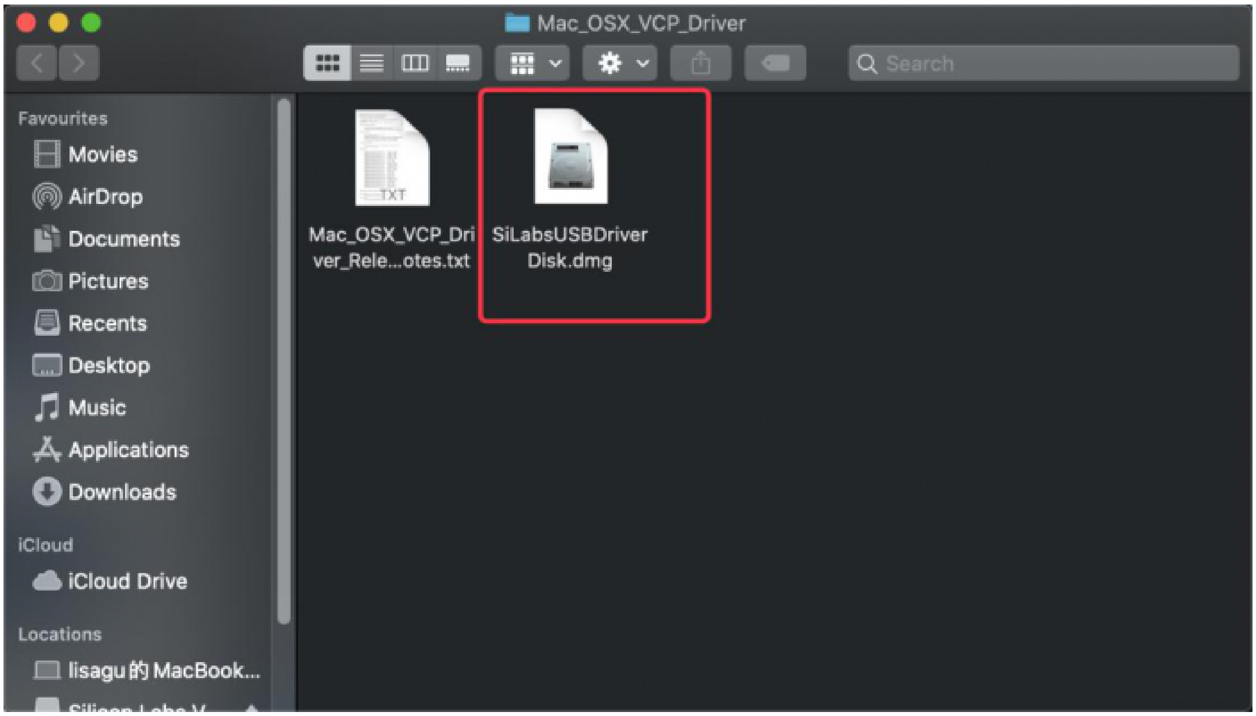
3Click to download the MacOS version, as shown below.



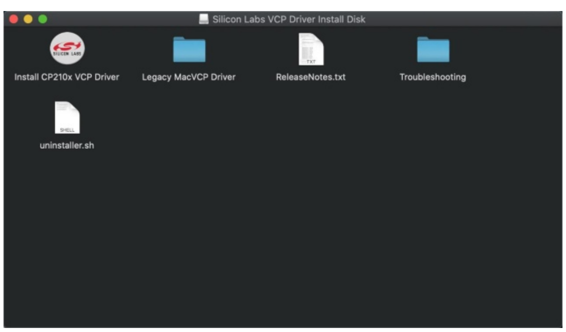
4Unzip the downloaded package.



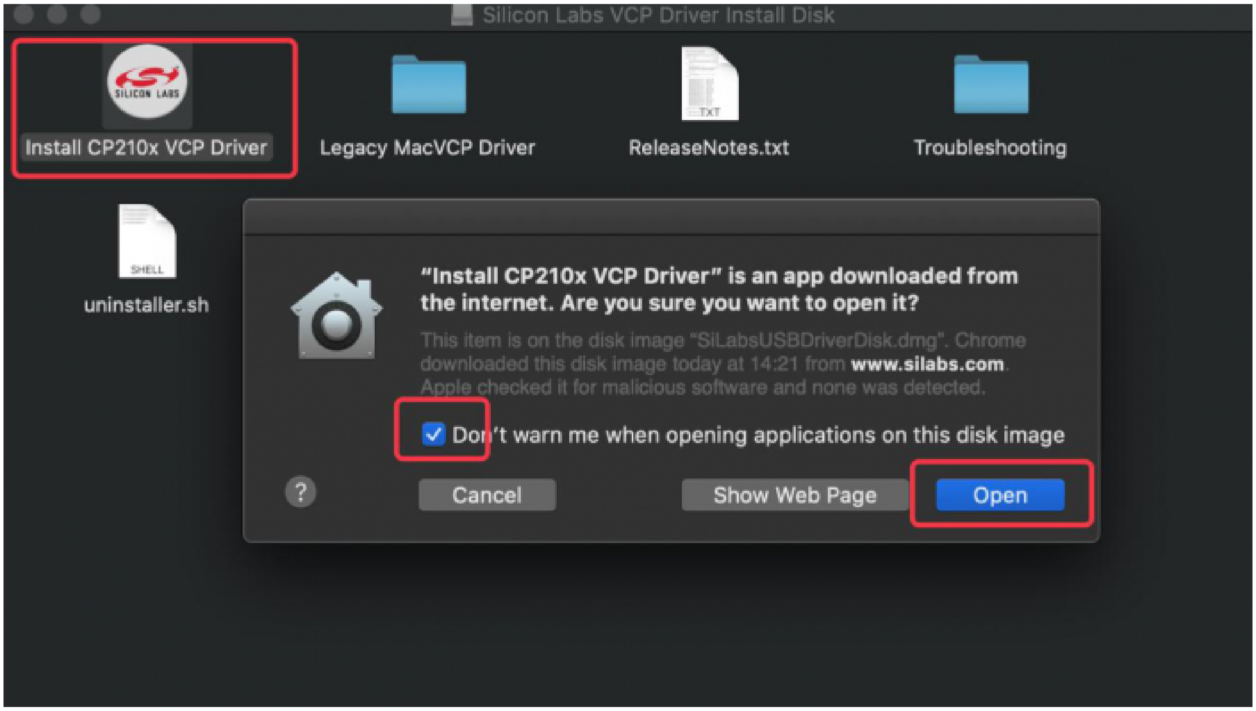
5Open folder and double-click“**SiLabsUSBDriverDisk.dmg**” file.



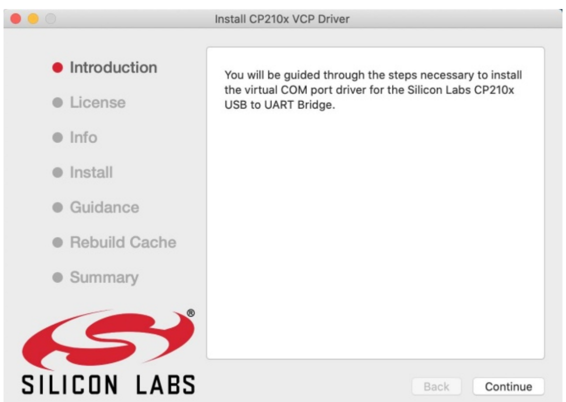
You will view the following files as follows:



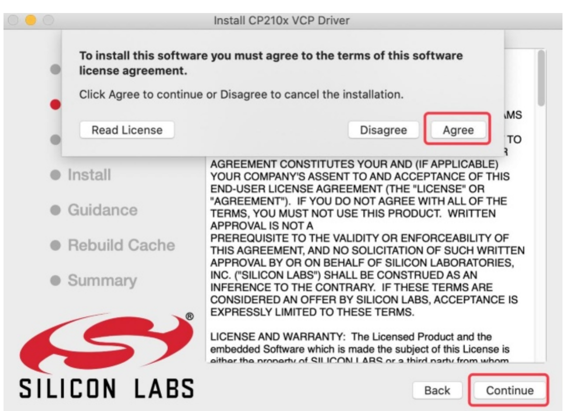
6Double-click**“Install CP210x VCP Driver”,** check“**Don’t warn me when opening application on this disk image**”and click“**Open**”.



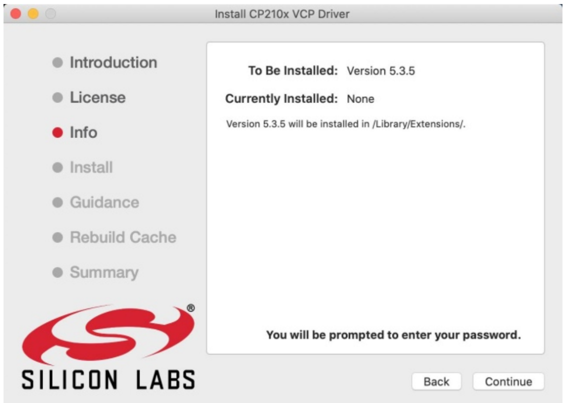
7Click"**Continue**".

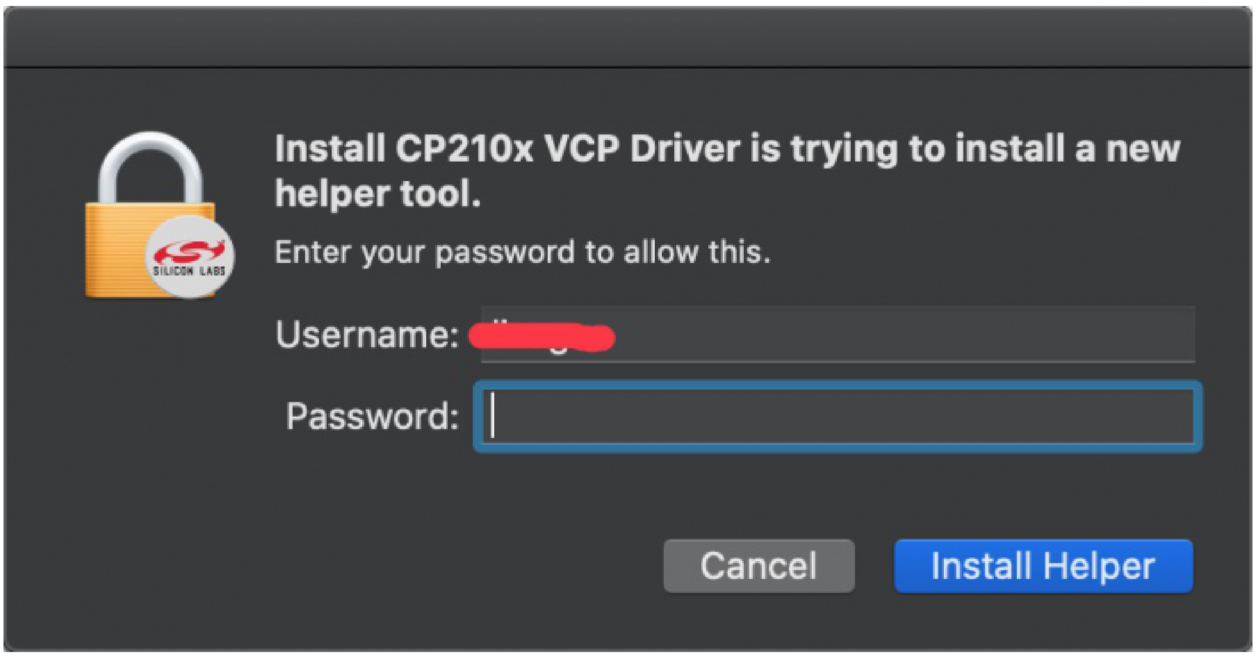


8Click "**Agree**" and then click "**Continue**".

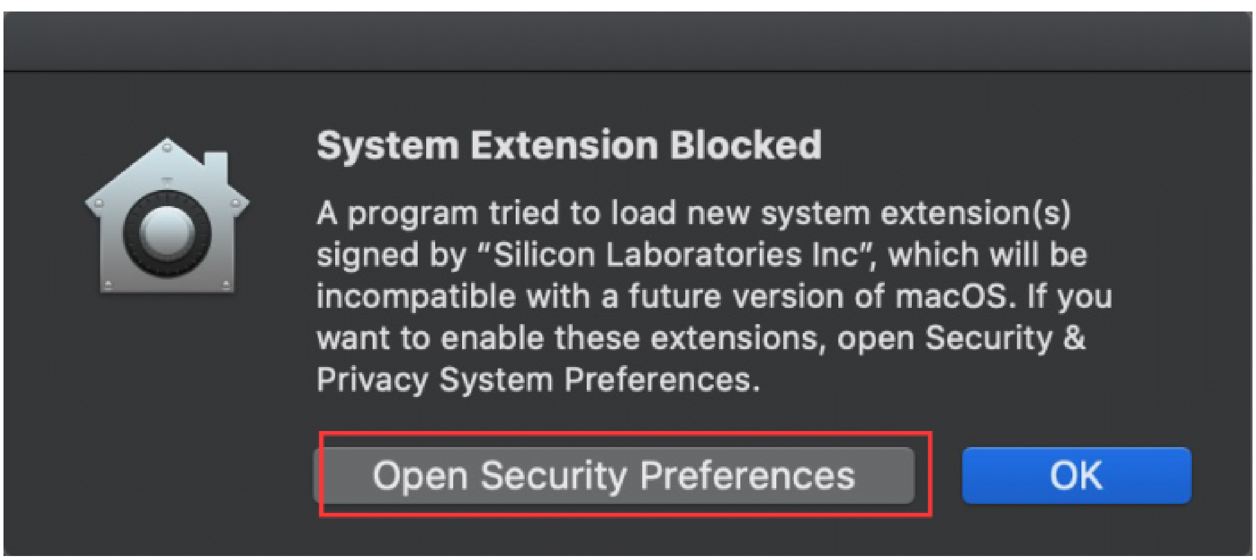


9Click "**Continue**" and enter your user password.

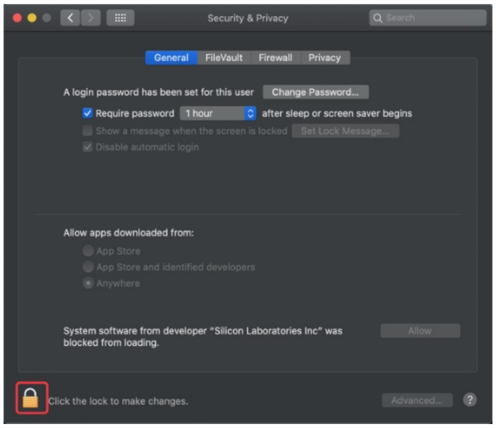


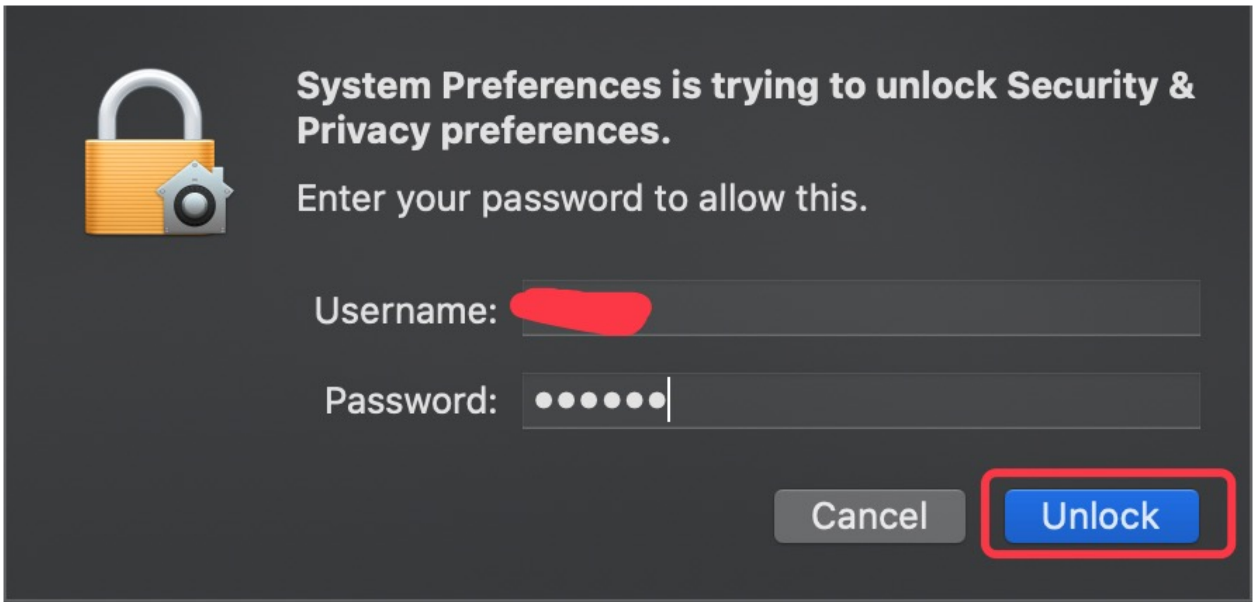


10. Select “**Open Security Preferences**”.

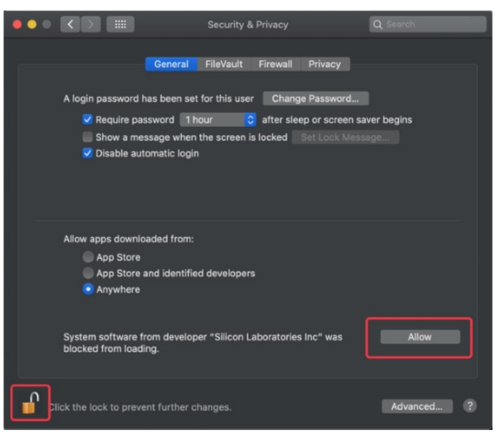


11Click the lock to unlock security & privacy preference, enter your user password to authorize, and then click“**unlock**”.

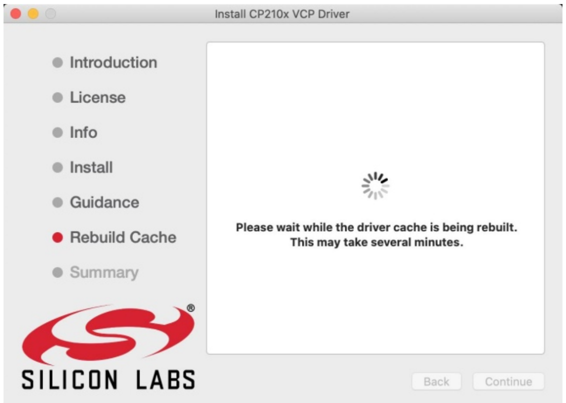




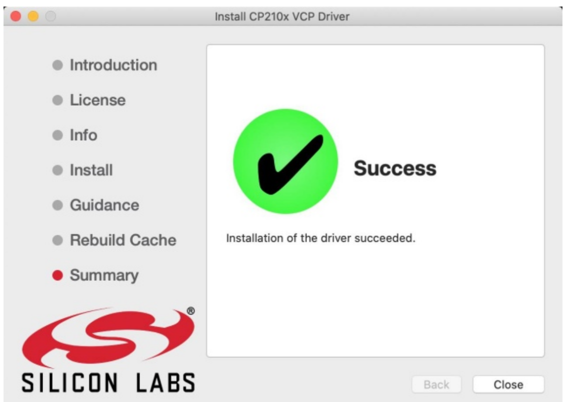
12See the lock has been opened, click“**Allow**”.



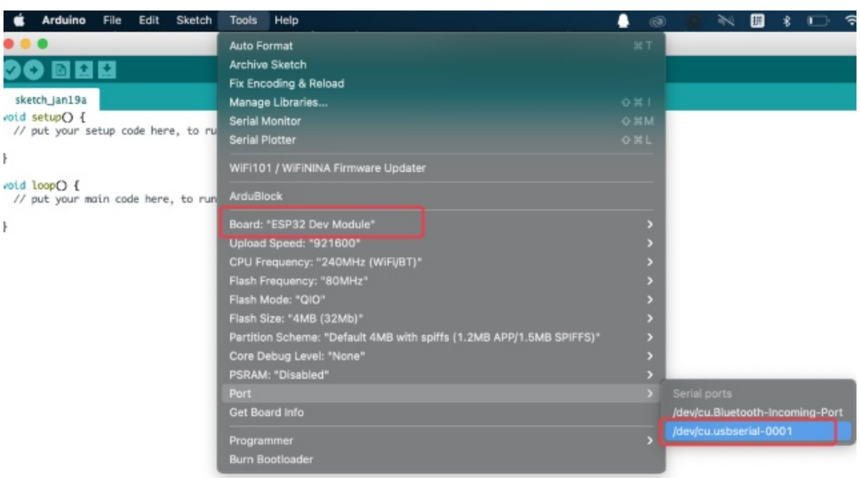
13Back to installation page, and wait to install.



14Successfully installed.



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

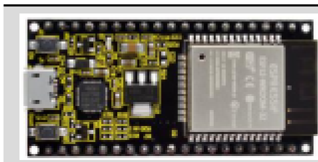
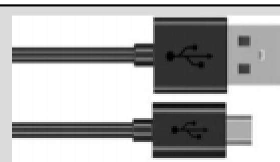
Click on the link to download the Libraries file: Download Libraries file

5.2 Project 01: Hello World

1. Introduction

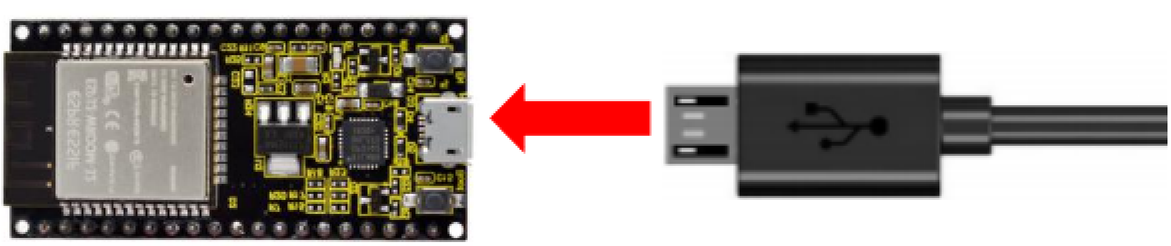
For ESP32 beginners, we'll start with some simple things. In this project, you just need an ESP32 mainboard, USB cable and computer to complete"Hello World!"Project. It is not only a communication test for ESP32 mainboard and computer, but also a primary project for ESP32.

2. Components

	
ESP32*1	USB cable*1

3. Wiring

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



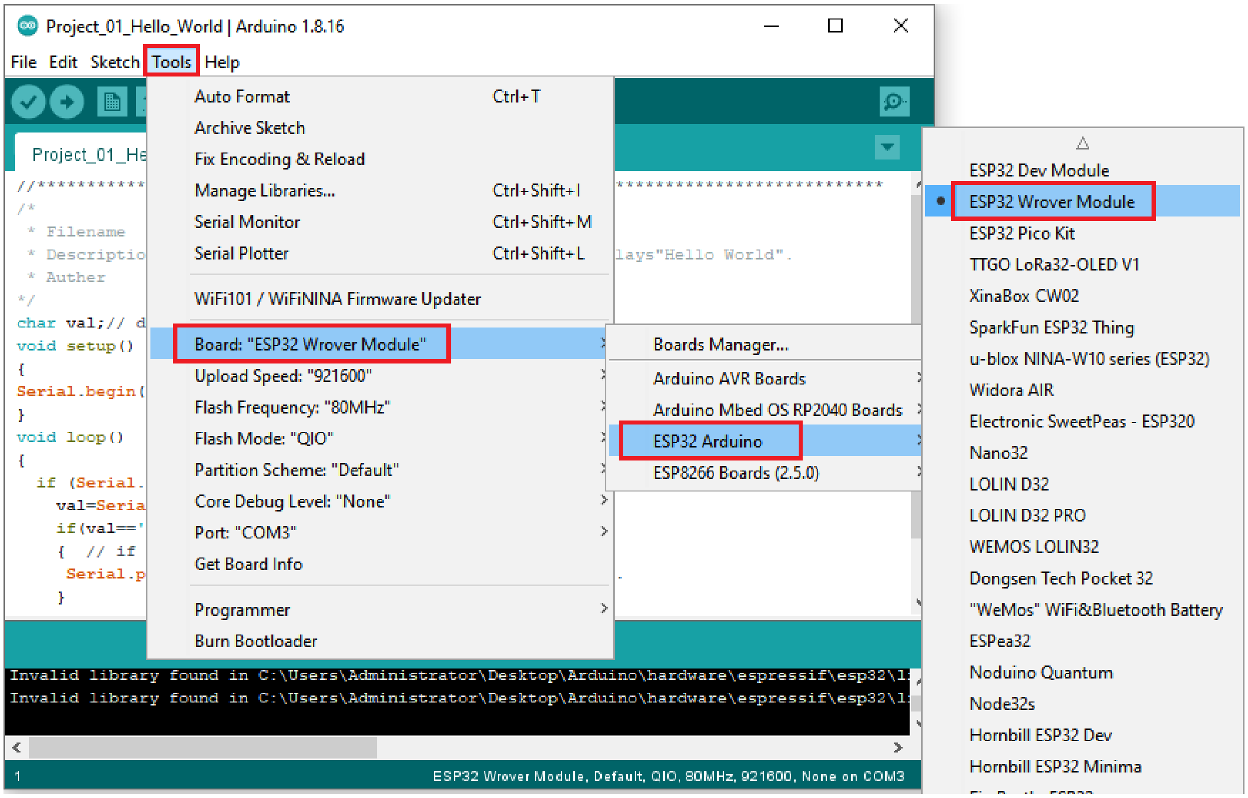
4. Project code

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

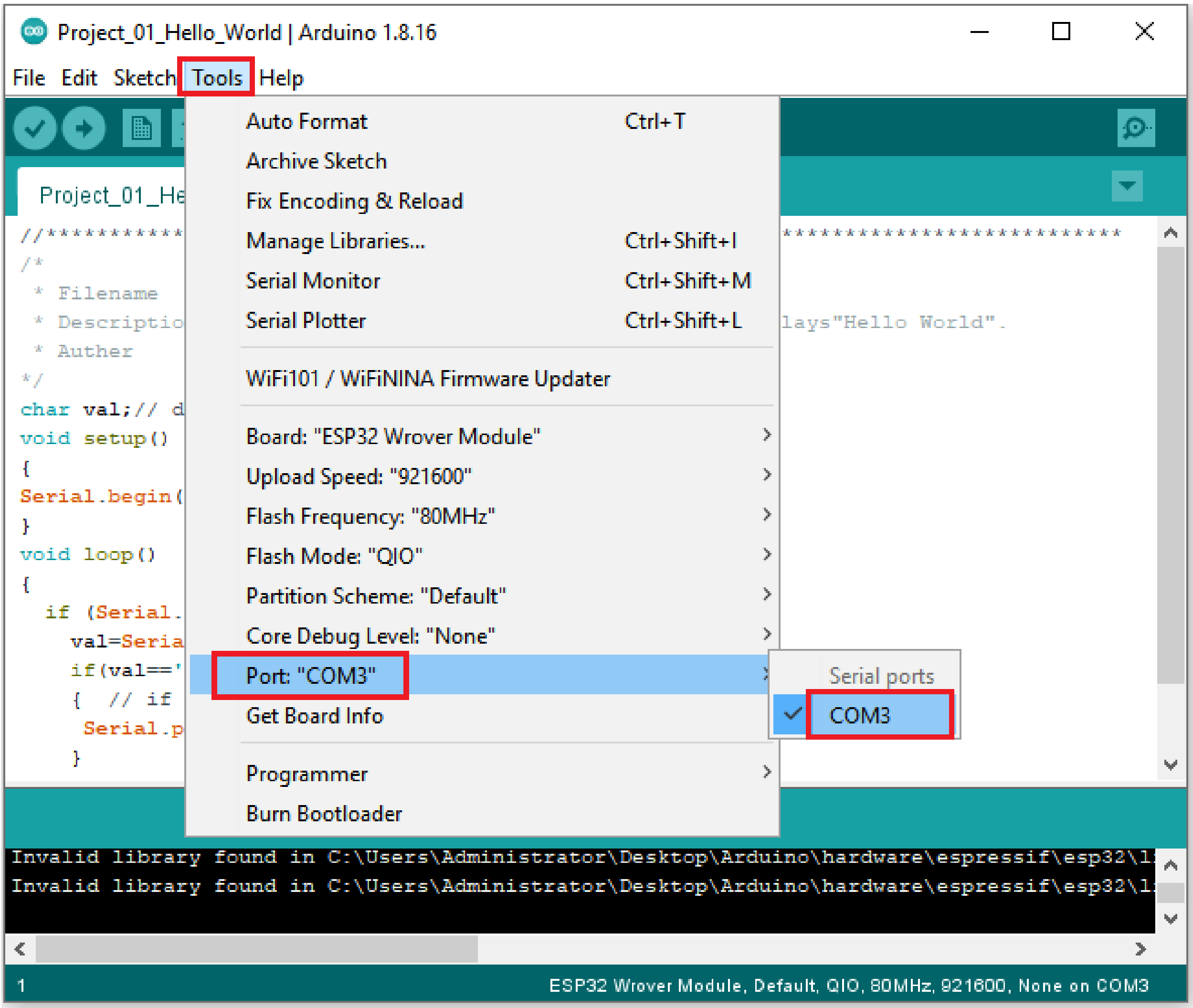
The code used in this project is saved in folder(path :) **"Arduino-Codes\Project 01 HelloWorld\Project_01_Hello_World"**.

```
//**********************************************************************************
/*
 * 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 !".
    }
  }
}
//**********************************************************************************
```

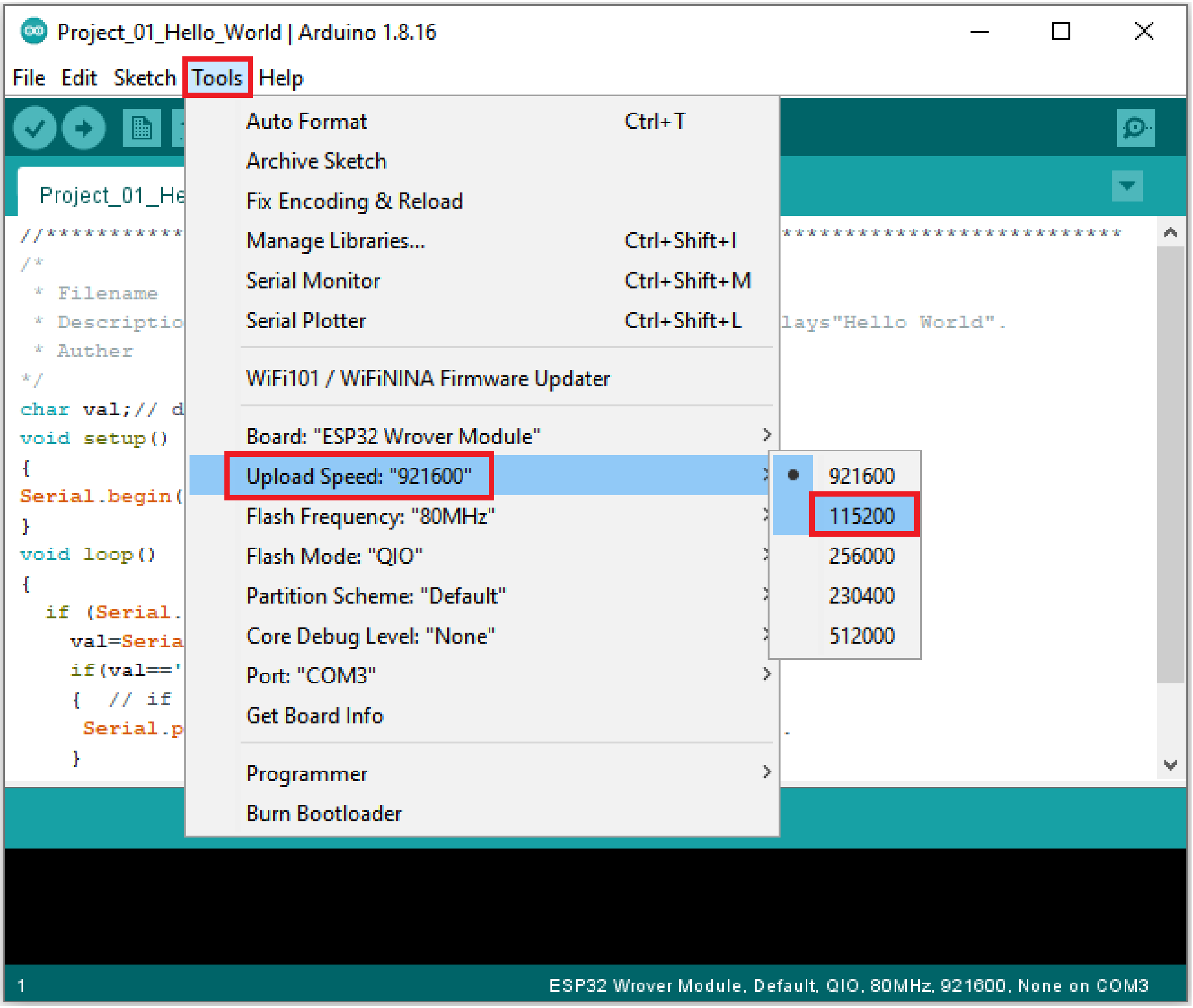
Before uploading the project code to ESP32click"Tools"→"Board" and select"ESP32 Wrover Module".



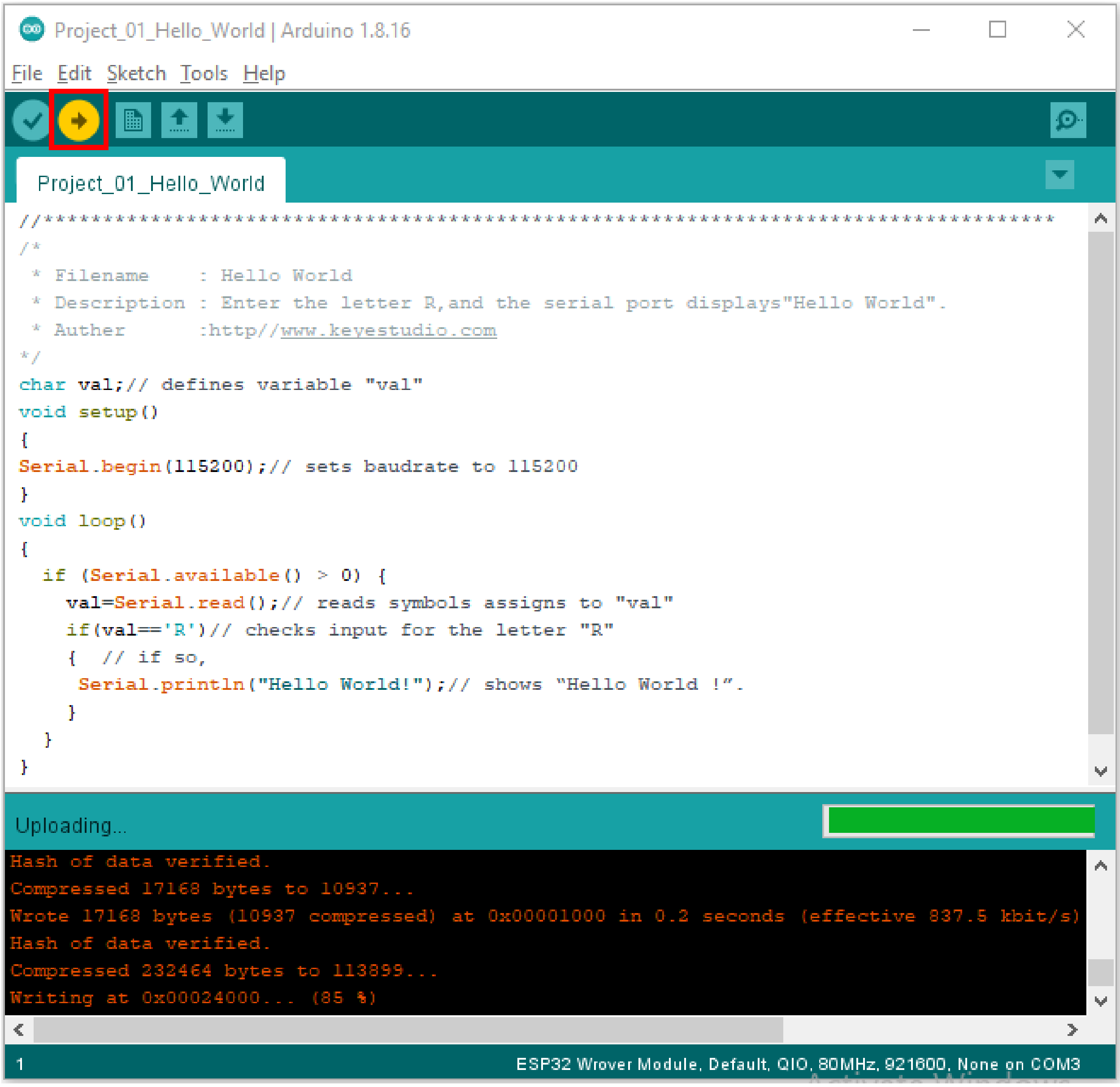
Select the serial port.

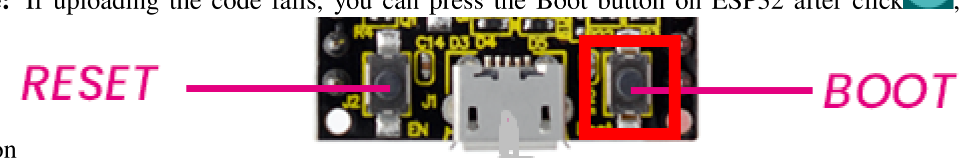


Note: For macOS users, if the uploading fails, please set the baud rate to 115200 before clicking .

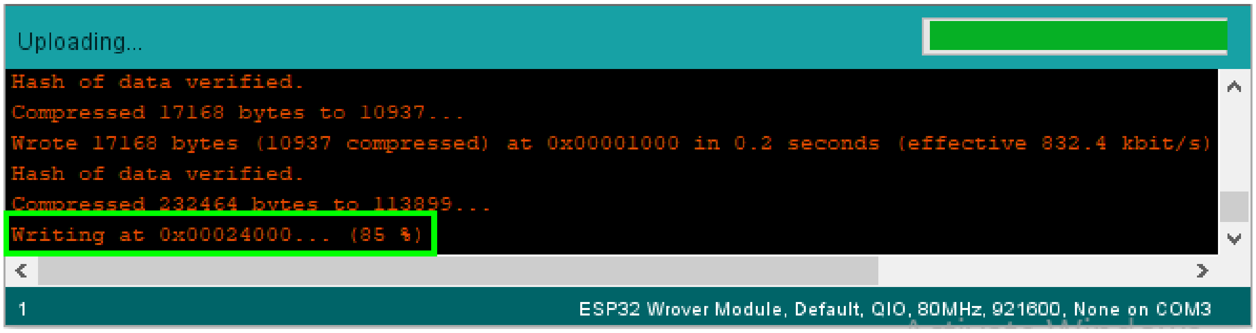


Click  to download the code to ESP32.



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 appears, as shown below:

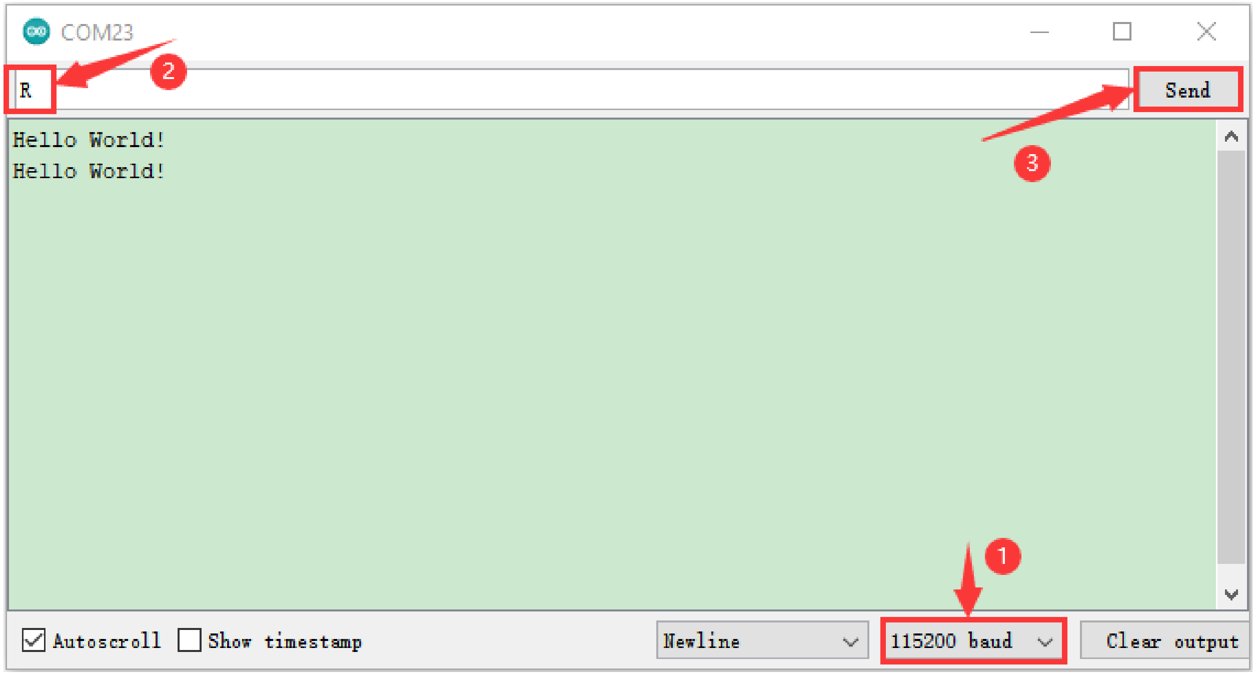


The Project code is uploaded successfully

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!".

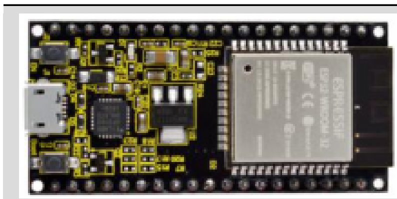
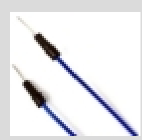
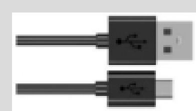
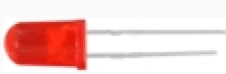
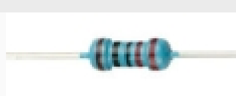


5.3 Project 02: Turn On LED

1. Introduction

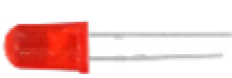
In this project, we will show you how to light up the LED. We use the ESP32's digital pin to turn on the LED so that the LED is lit up.

2. Components

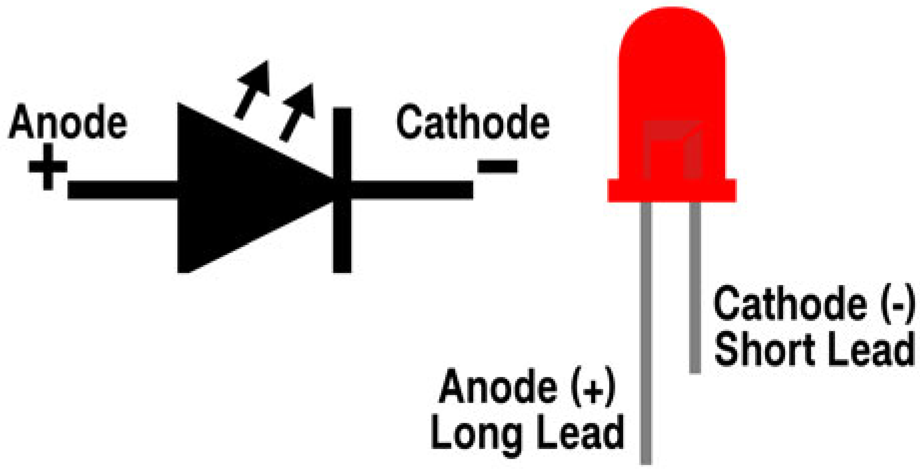
			
ESP32*1	Breadboard*1	Jumper Wire*2	USB Cable*1
			
Red LED*1	220 Resistor*1		

3. Component knowledge

1LED:

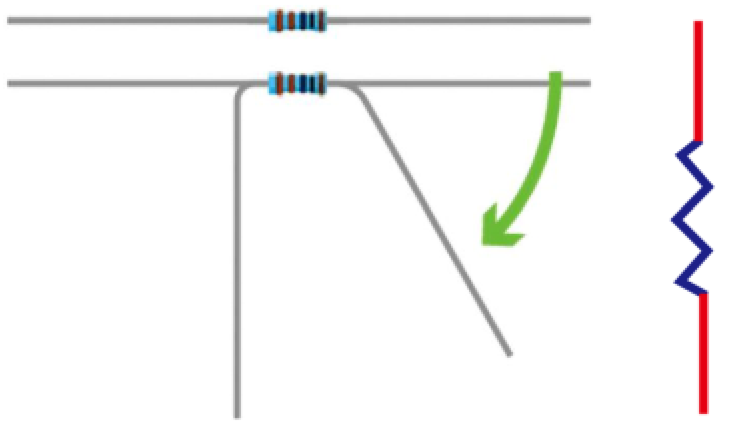


The LED is a semiconductor known as “light-emitting diode” , which is an electronic device made from semiconducting materials(silicon, selenium, germanium, etc.). It has an anode and a cathode, the short lead is cathode, which connects to GND; the long lead is anode, which connects to 3.3V or 5V.



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.

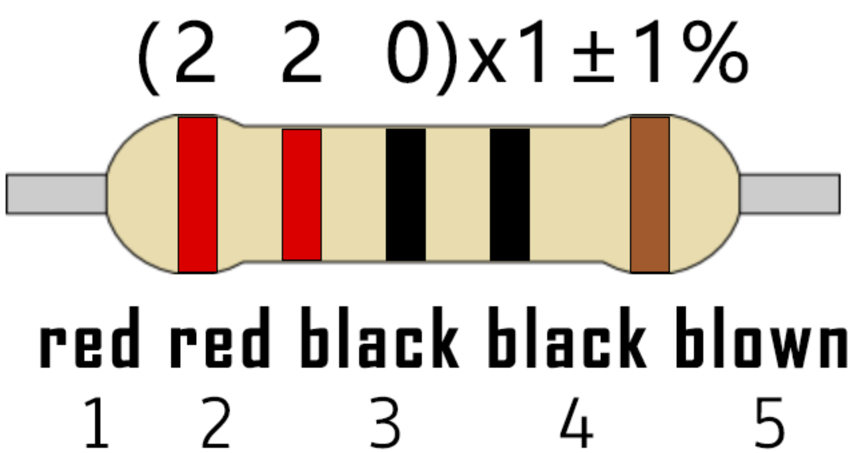
-Color 4 – Multiplier.

-Color 5 – Tolerance.

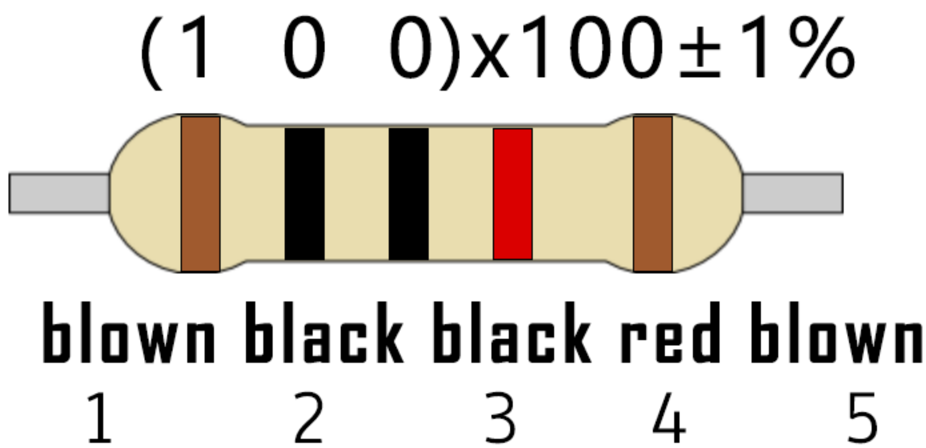
	1st Digit	2nd Digit	3rd Digit	Multiplier	Tolerance
Black		0	0	x1	
Brown	1	1	1	x10	± 1%
Red	2	2	2	x100	± 2%
Orange	3	3	3	x1K	± 3%
Yellow	4	4	4	x10K	± 4%
Green	5	5	5	x100K	± 0.5%
Blue	6	6	6	x1M	± 0.25%
Violet	7	7	7	x10M	± 0.10%
Grey	8	8	8	x100M	± 0.05%
White	9	9	9	x1G	
Gold				÷ 10	± 5%
Silver				÷ 100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

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

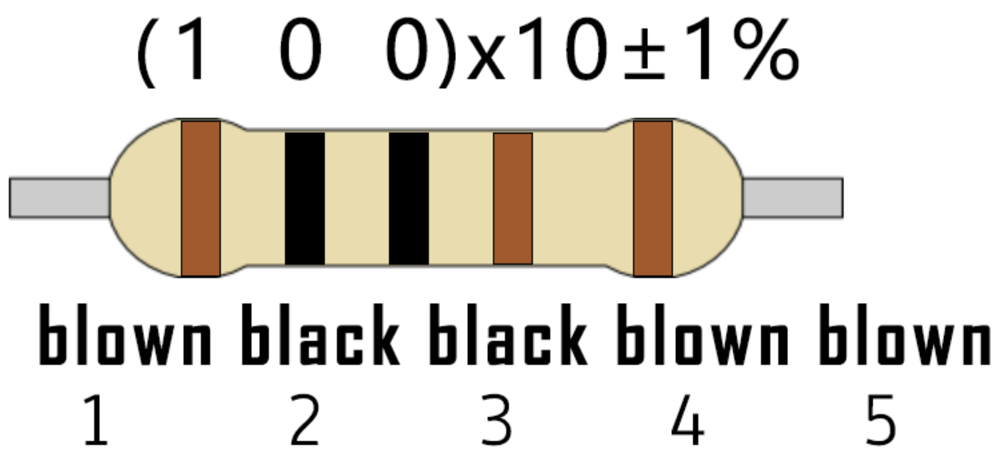
220 Resistor*10



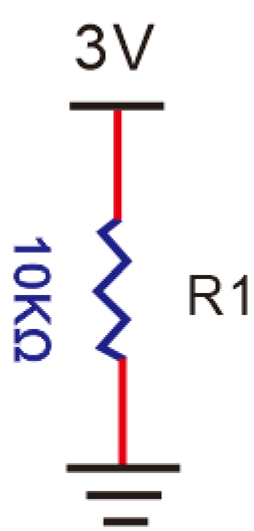
10K Resistor*10



1K Resistor*10



In the same voltage, there will be less current and more resistance. The connection between current(I), voltage(V), and resistance® 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.



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 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

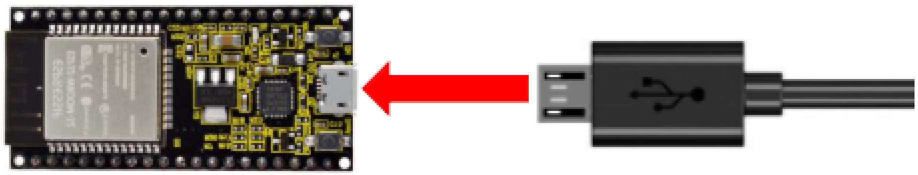
When we connect DIP (Dual In-line Packages) components, such as integrated circuits, microcontrollers, chips and so on, we can see that a groove in the middle isolates the middle part, so the top and bottom of the groove is not connected. DIP components can be connected as shown in the following diagram:

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.

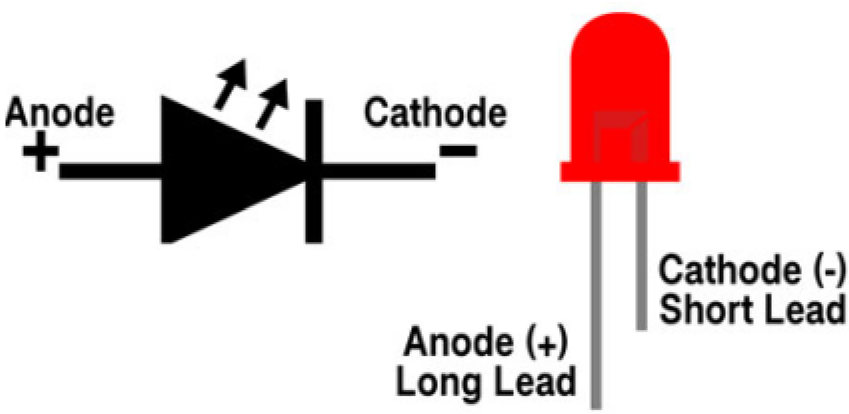
Note: Avoid any possible short circuits (especially connecting 3.3V and GND)!

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

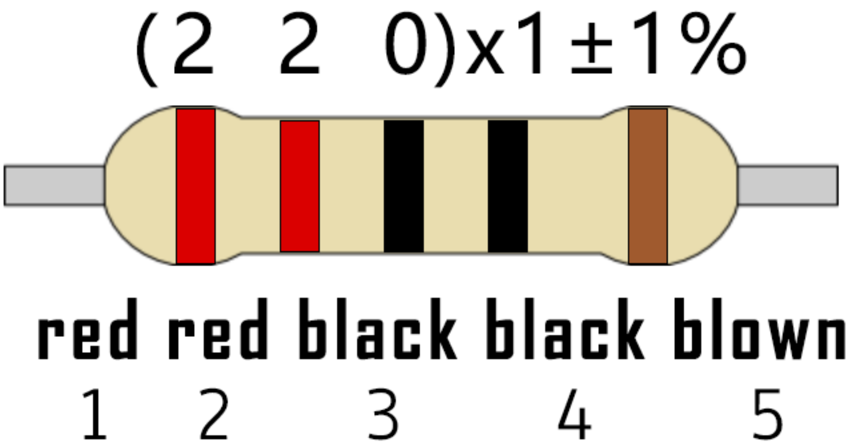
media/0735997593c8858ad6441d8e9867206f.png

Note:

How to connect a LED



How to identify the 220 Five-color ring resistor



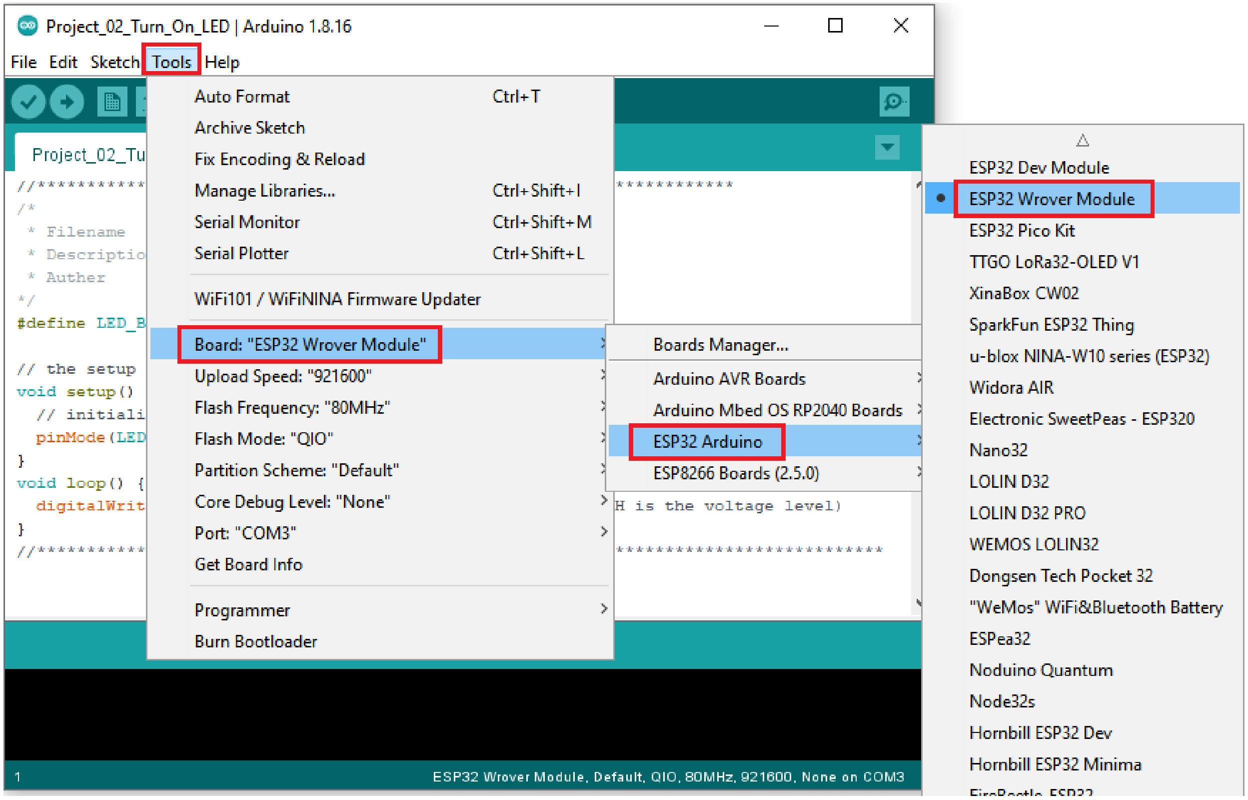
5. Project code

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

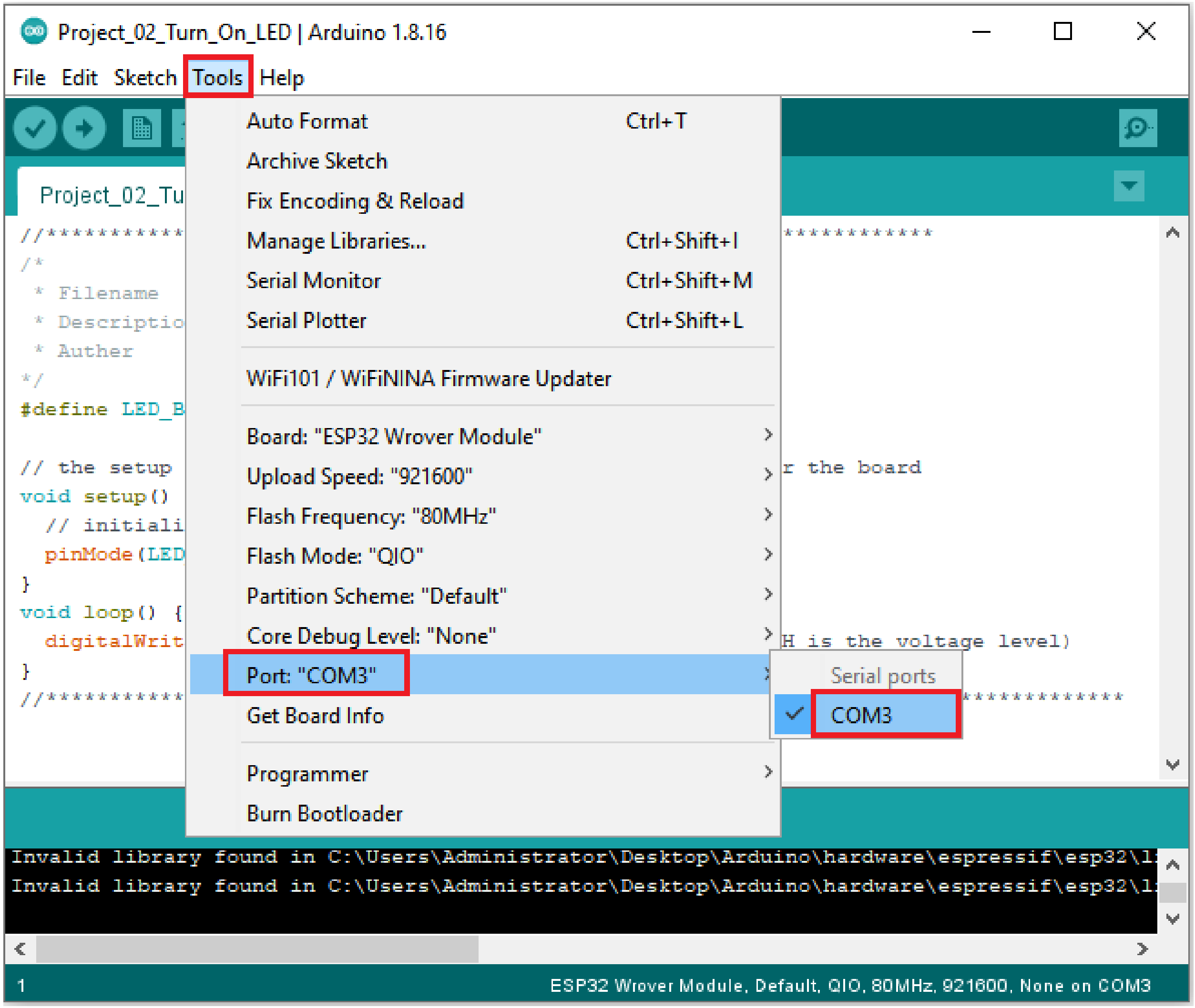
The code used in this project is saved in folder(path:)**"Arduino-Codes\Project 02Turn On LED\Project_02_Turn_On_LED"**.

```
//**********************************************************************
/*
 * 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)
}
//*********************************************************************************
```

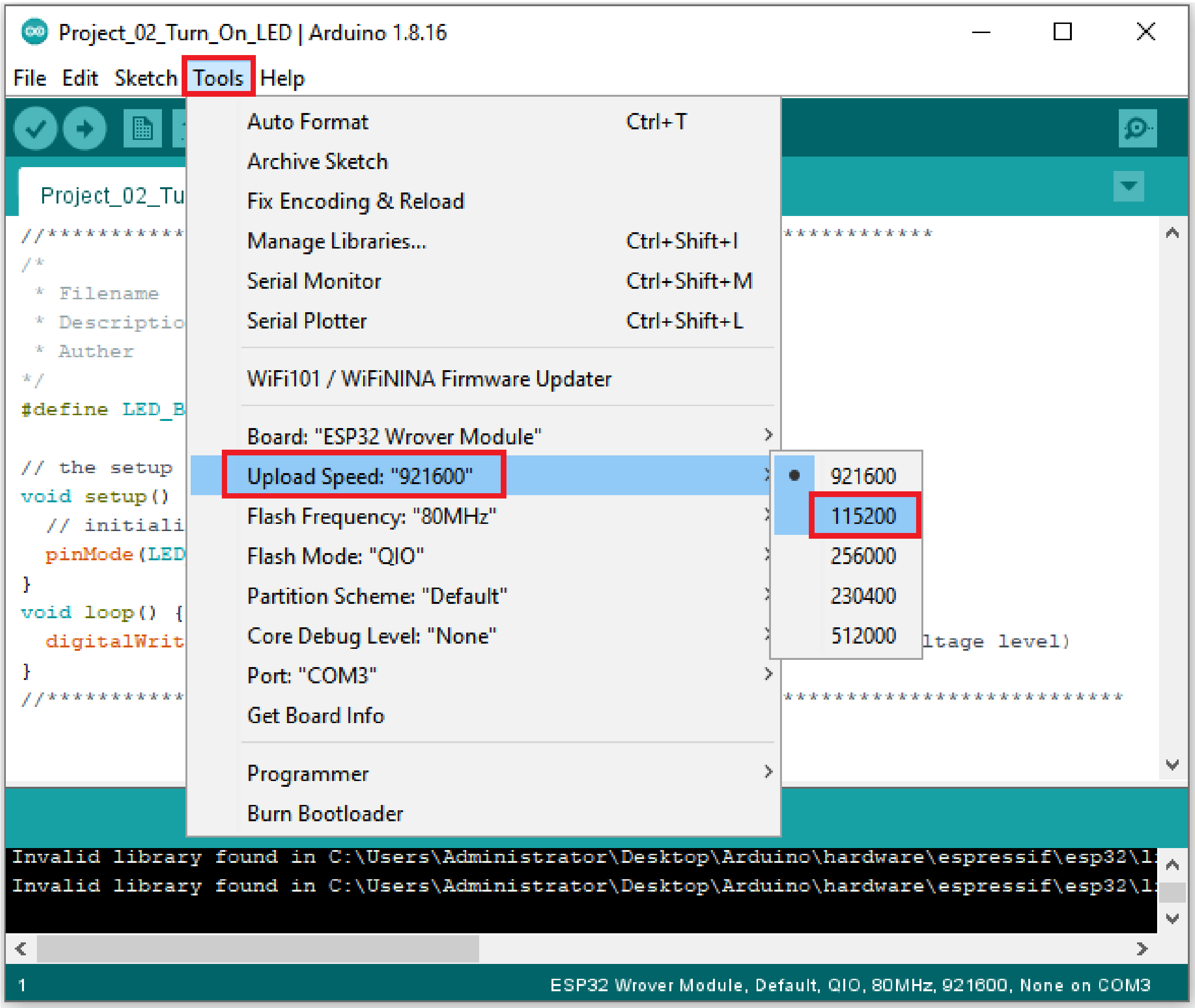
Before uploading the project code to ESP32click"Tools"→"Board" and select"ESP32 Wrover Module".



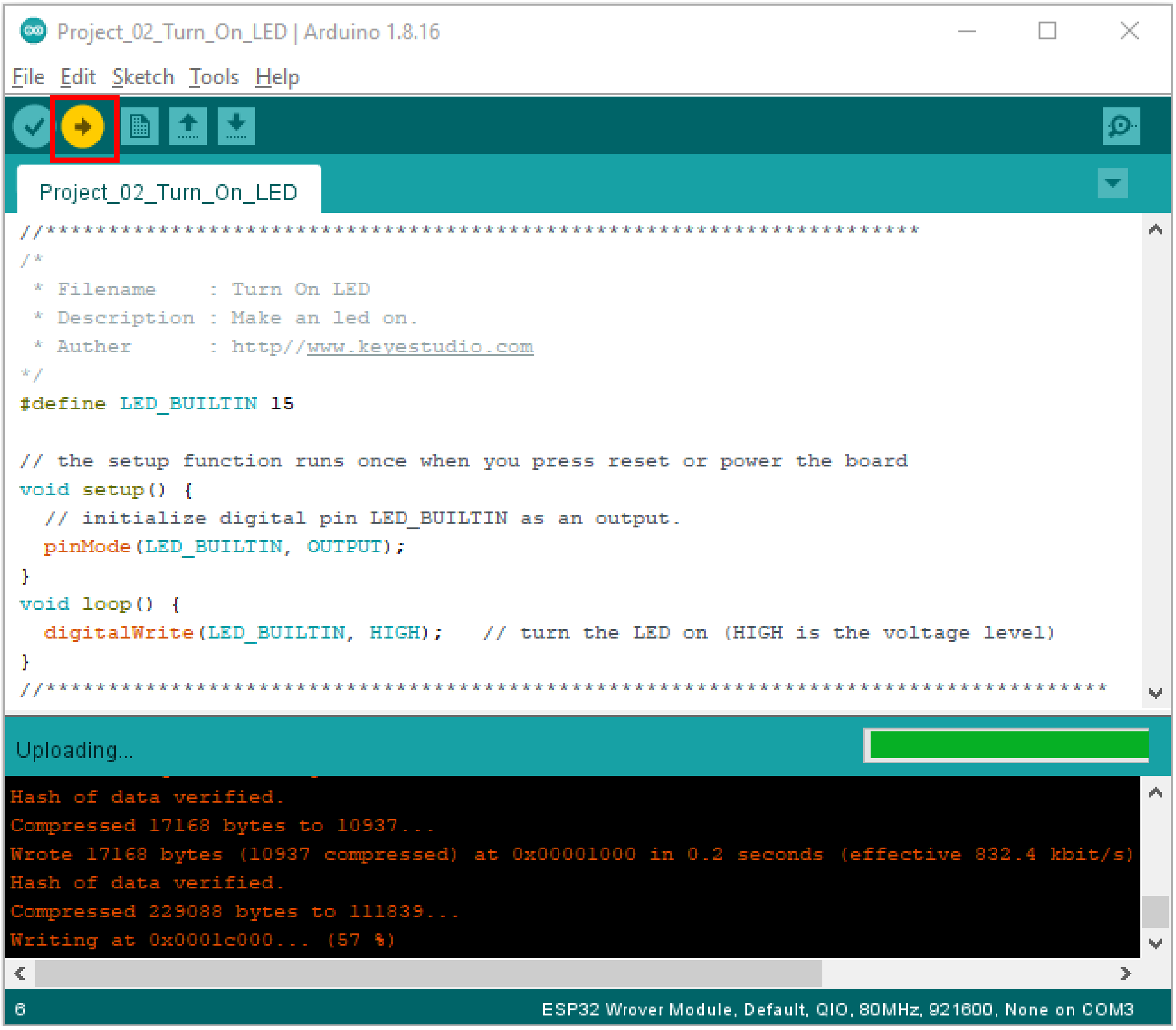
Select the serial port.



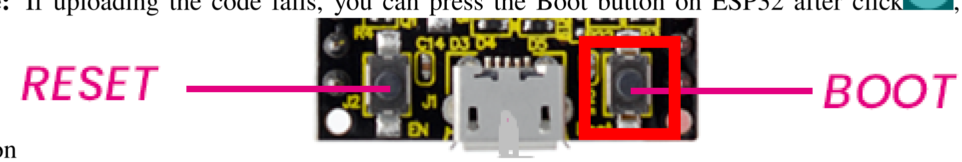
Note: For macOS users, if the uploading fails, please set the baud rate to 115200 before clicking .

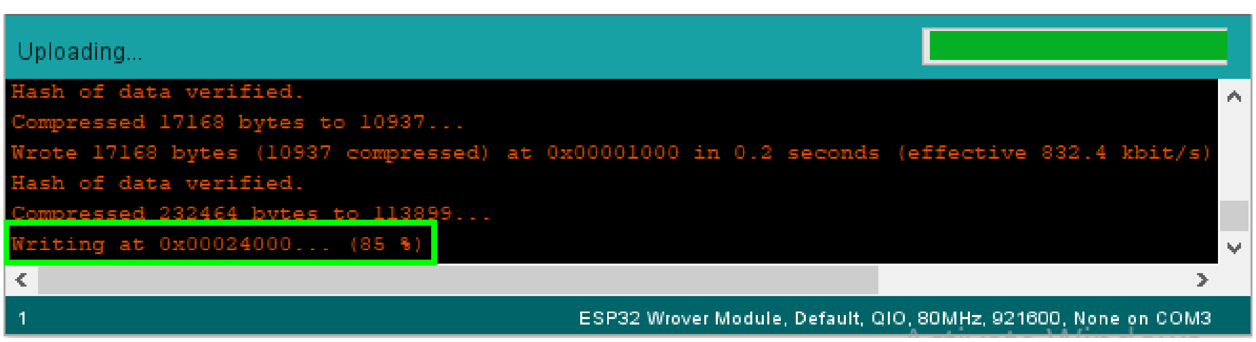


Click  to download the code to ESP32.

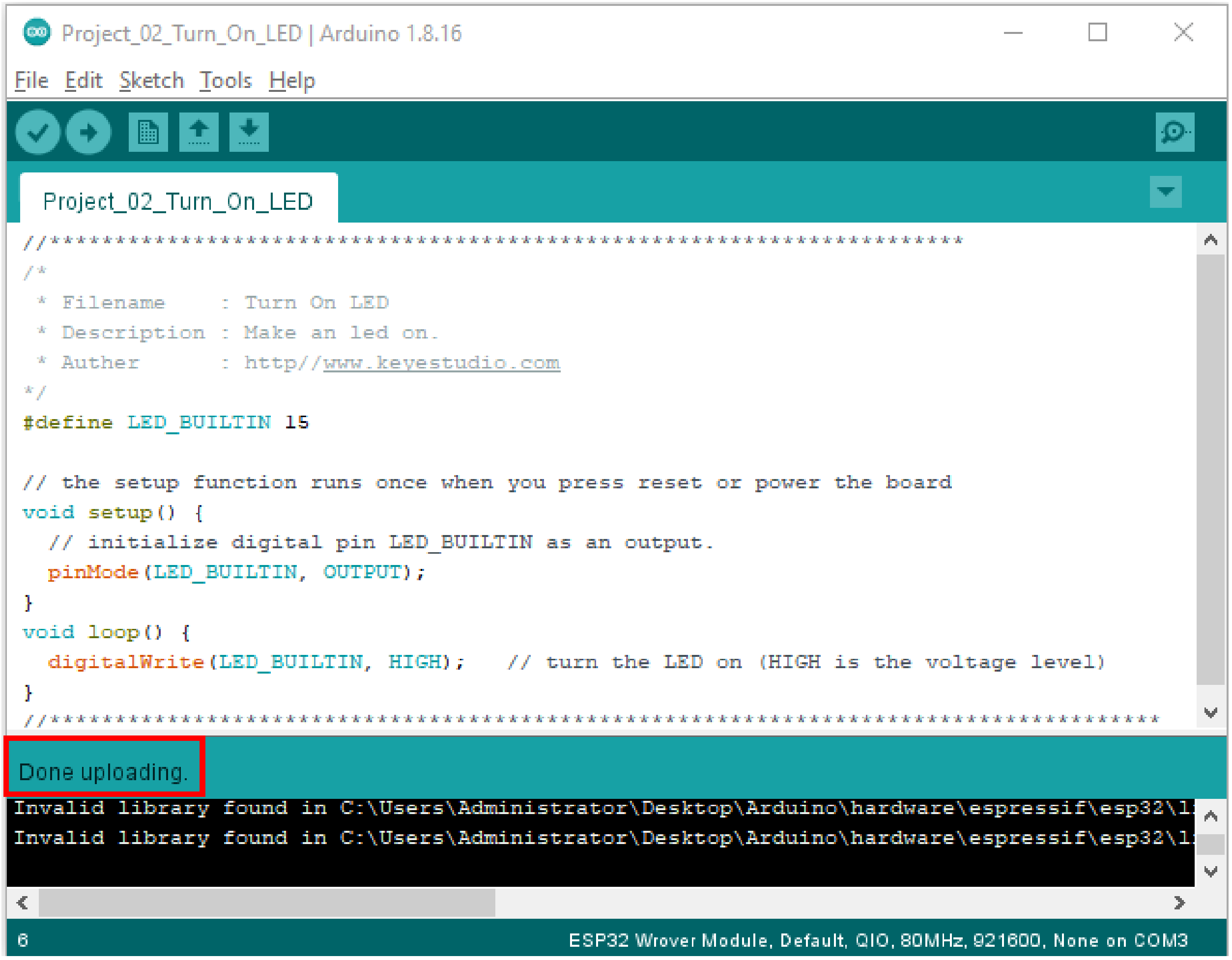


Note: If uploading the code fails, you can press the Boot button on ESP32 after click , and release the Boot button after the percentage of uploading progress appears, as shown below:



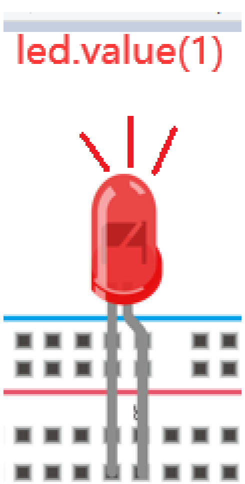


The Project code is uploaded successfully



6. Project result

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

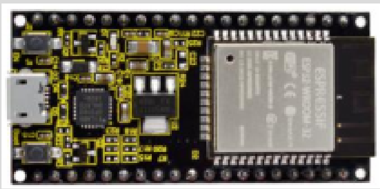
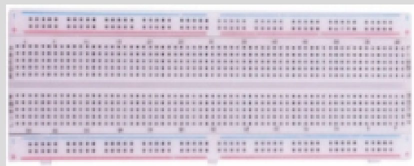
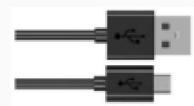


5.4 Project 03LED Flashing

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

			
ESP32*1	Breadboard*1		
			
Red LED*1	220 Resistor*1	Jumper Wire*2	USB Cable*1

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.

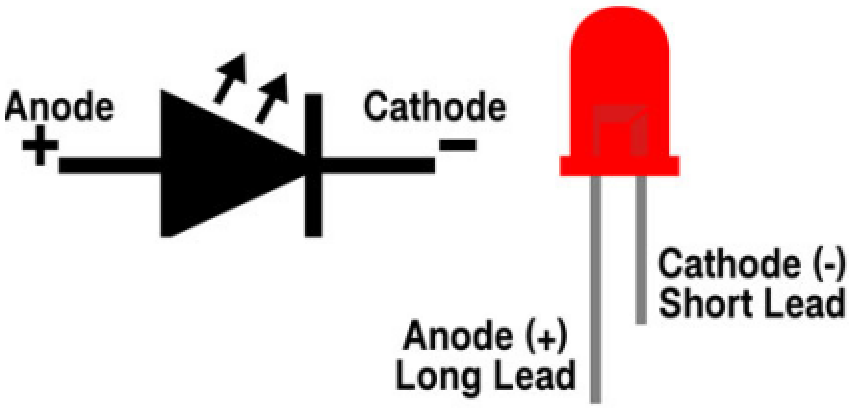
Note: Avoid any possible short circuits (especially connecting 3.3V and GND)!

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

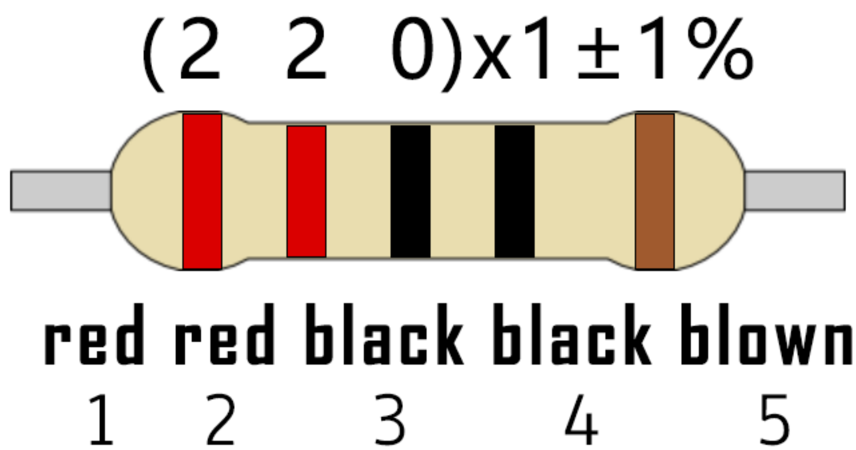
media/0735997593c8858ad6441d8e9867206f.png

Note:

How to connect a LED



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"**.

```
//*************************************************************************
/*
 * 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
```

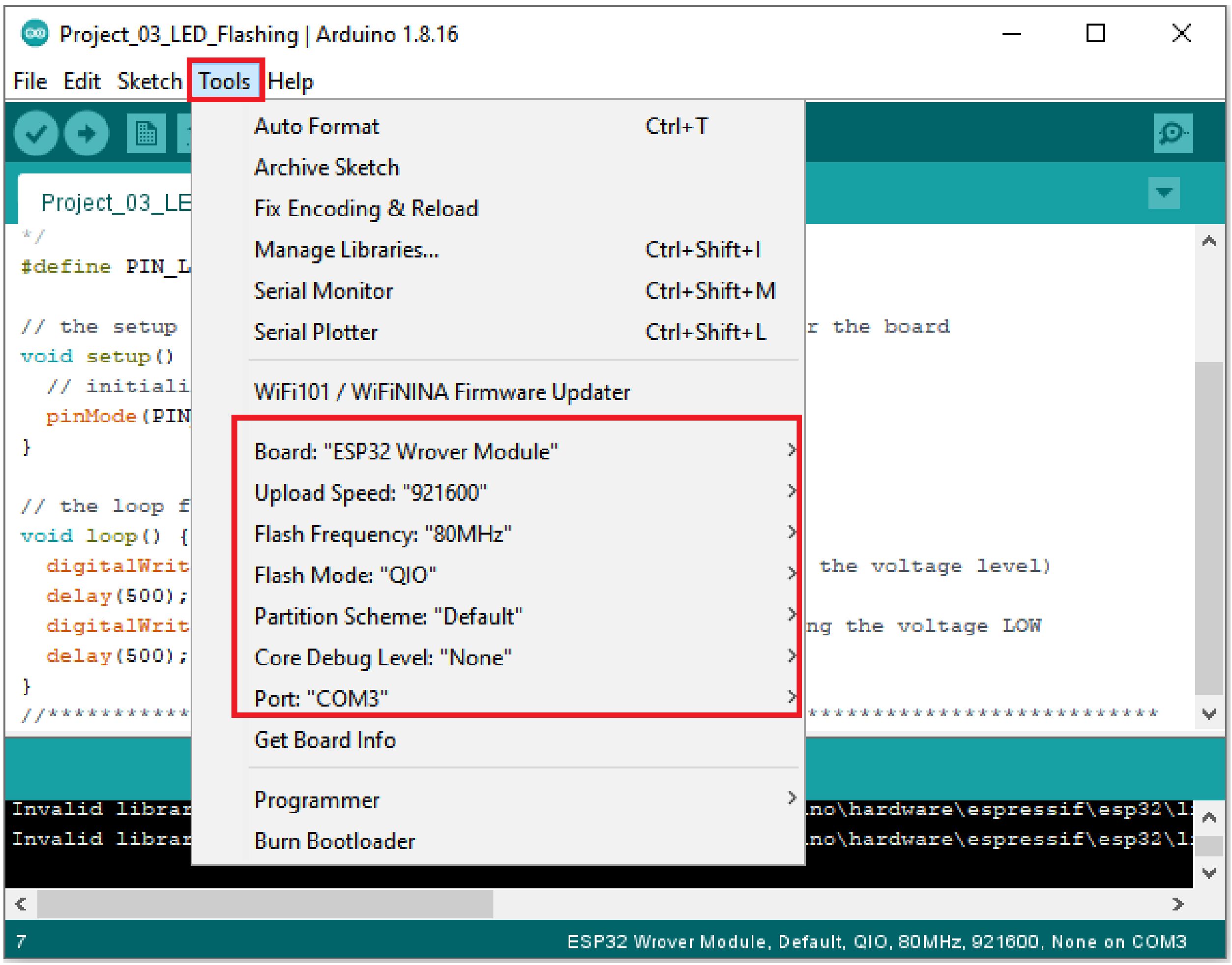
(continues on next page)

(continued from previous page)

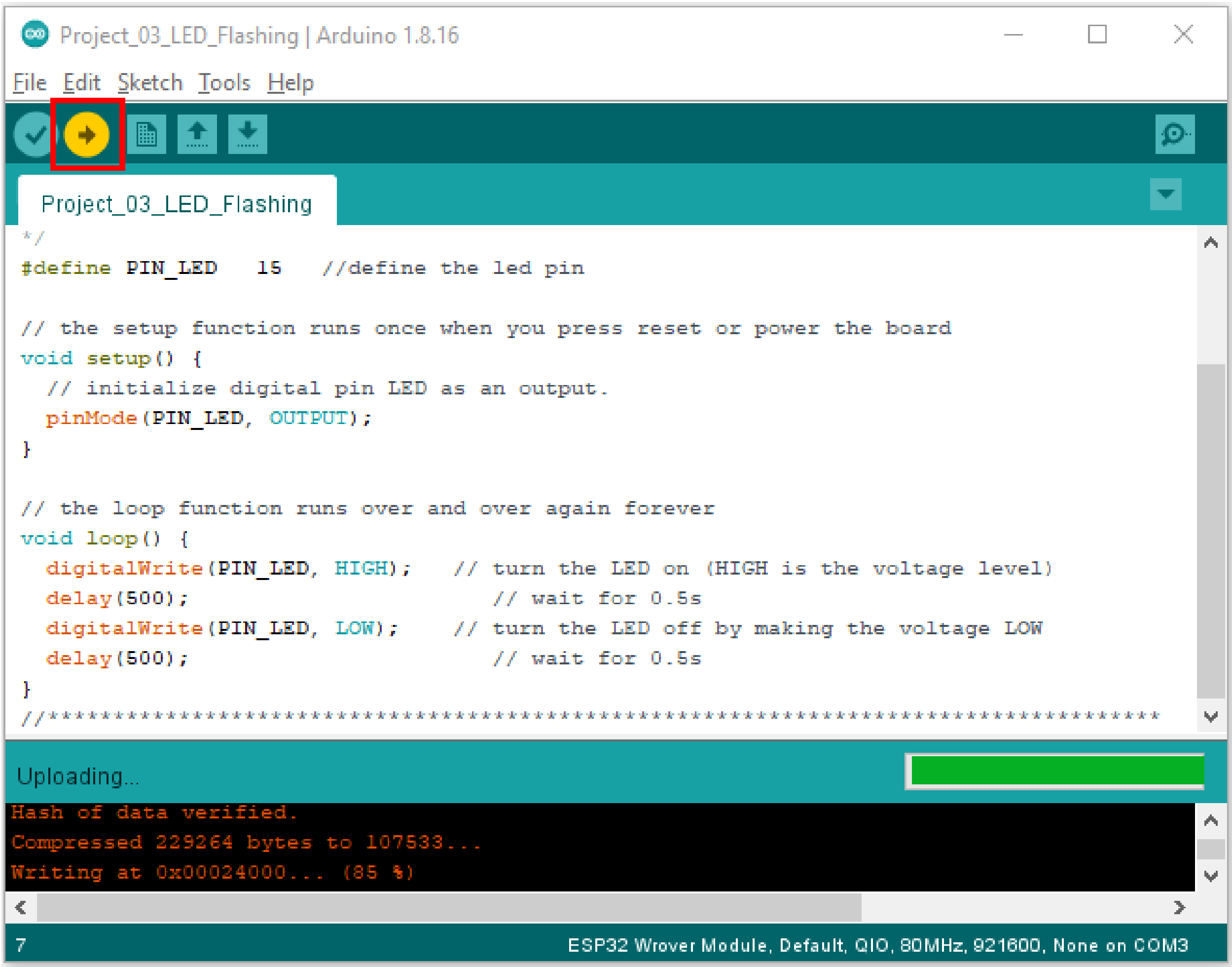
```
void loop() {
  digitalWrite(PIN_LED, HIGH);   // turn the LED on (HIGH is the voltage level)
  delay(500);                       // wait for 0.5s
  digitalWrite(PIN_LED, LOW);    // turn the LED off by making the voltage LOW
  delay(500);                       // wait for 0.5s
}
//*************************************************************************************
```

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

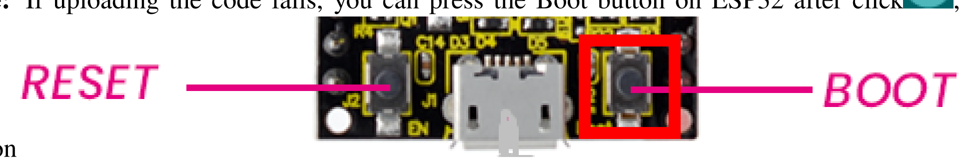
Click “**Tools**” to confirm the board type and port as shown below:



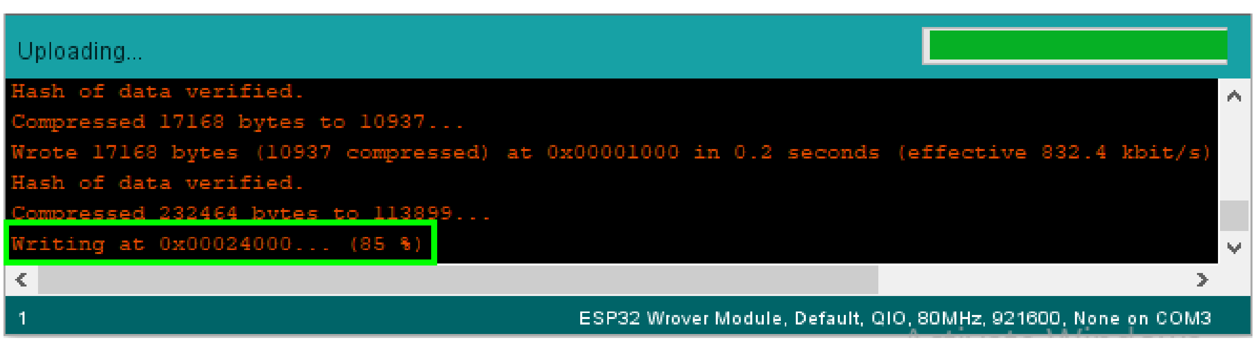
Click  to download the project code to ESP32.



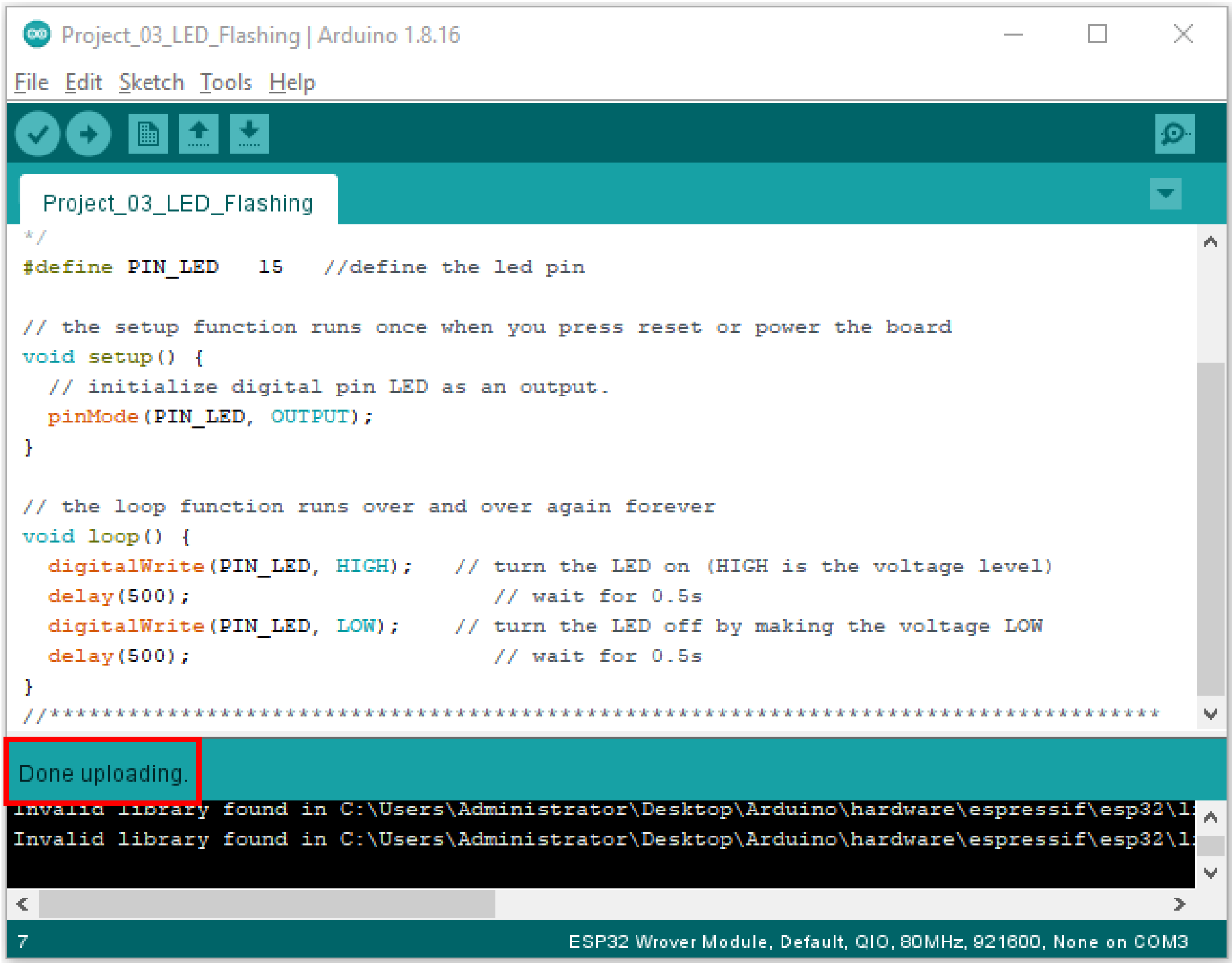
Note: If uploading the code fails, you can press the Boot button on ESP32 after click , and release the Boot button



after the percentage of uploading progress appears, as shown below:

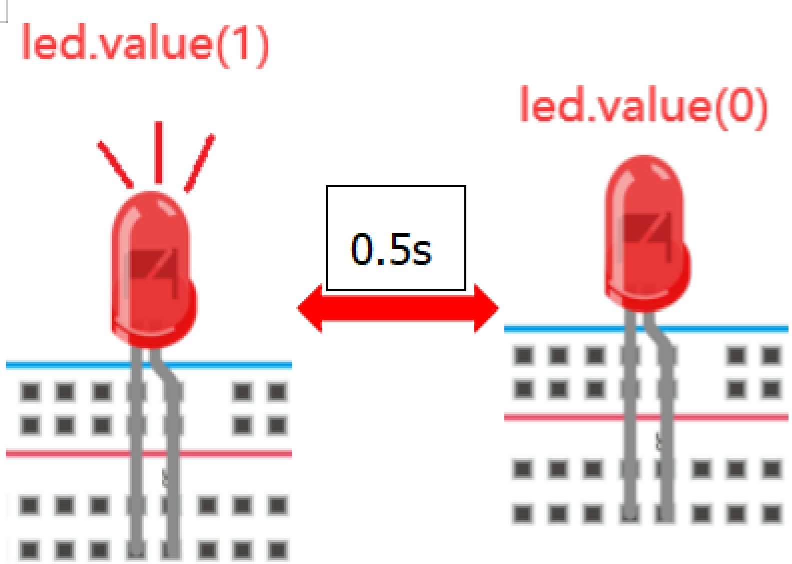


The Project code is uploaded successfully



5. Project result

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



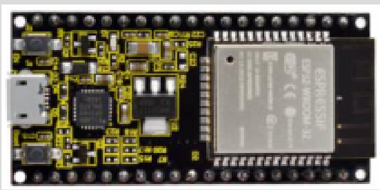
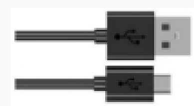
5.5 Project 04: Breathing Led

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.

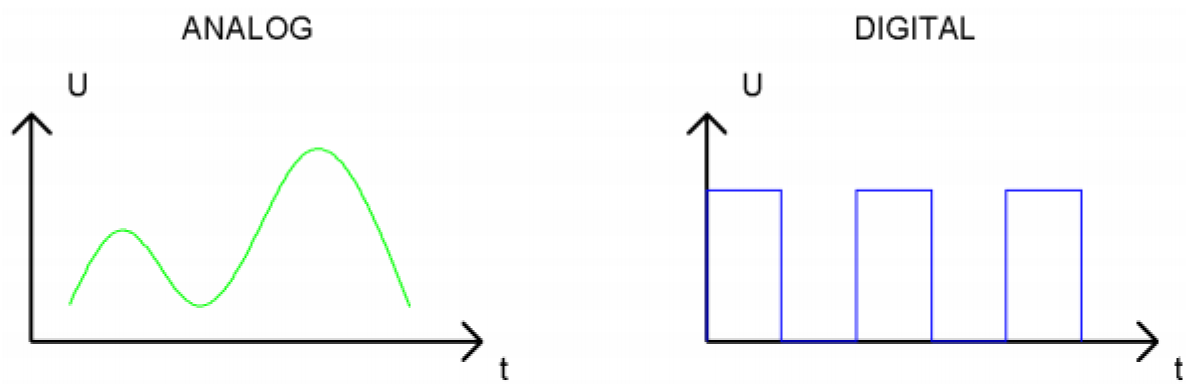
2. Components

			
ESP32*1	Breadboard*1		
			
Red LED*1	220 Resistor*1	Jumper Wire*2	USB Cable*1

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℃ to 10℃. 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.



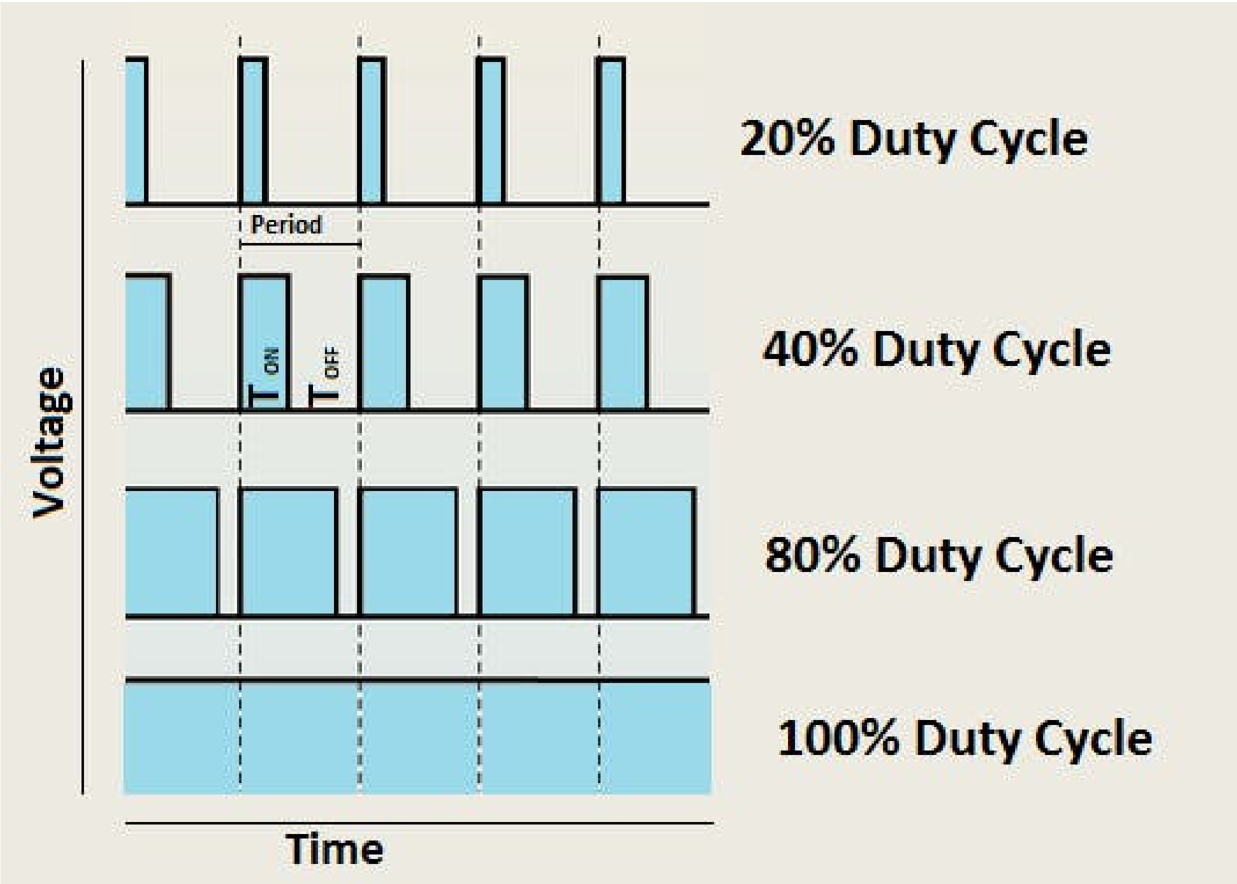
In practical application, we often use binary as the digital signal, that is a series of 0's and 1's. Since a binary signal only has two values (0 or 1), it has great stability and reliability. Lastly, both analog and digital signals can be converted into the other.

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.

$$\mathrm{Freq}_{max} = \frac{80,000,000}{1 \ll bit}$$

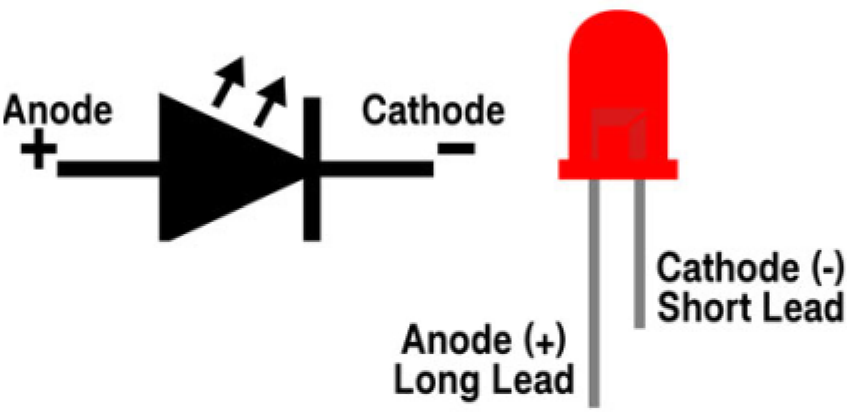
For example, generate a PWM with an 8-bit precision (28=256. Values range from 0 to 255) with a maximum frequency of 80,000,000/255 =312,500Hz.

4. Wiring diagram

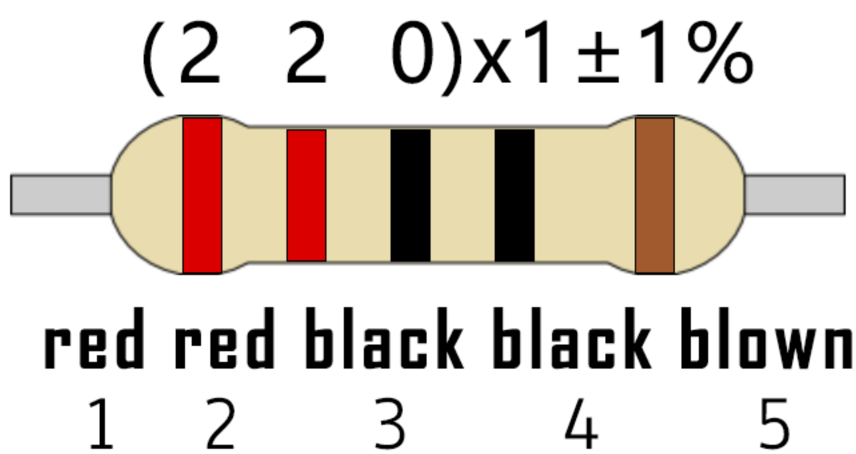
media/0735997593c8858ad6441d8e9867206f.png

Note:

How to connect a LED



How to identify the 220 Five-color ring resistor



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"**.

```
//**********************************************************************
/*
 * Filename    : Breathing Led
```

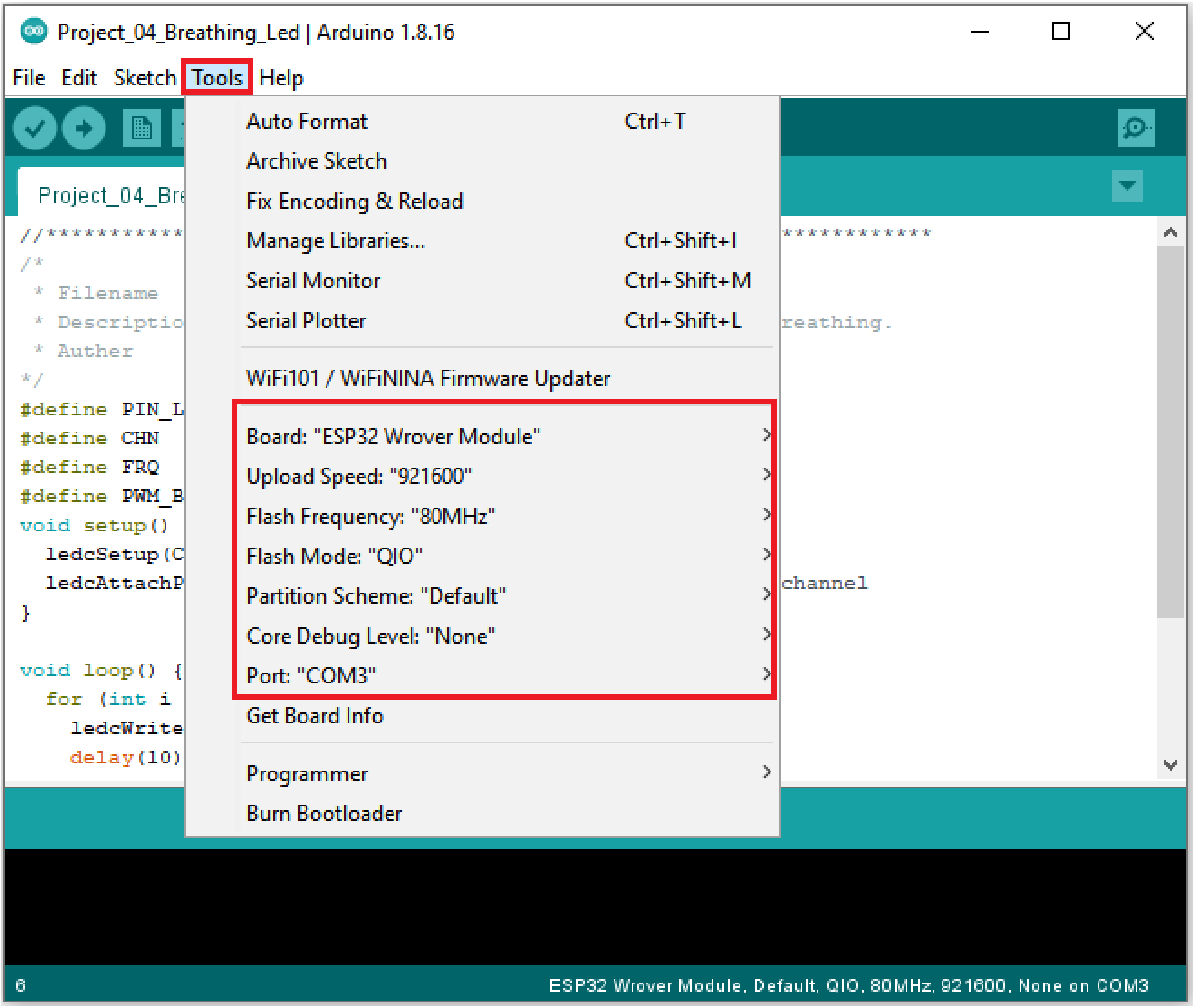
(continues on next page)

(continued from previous page)

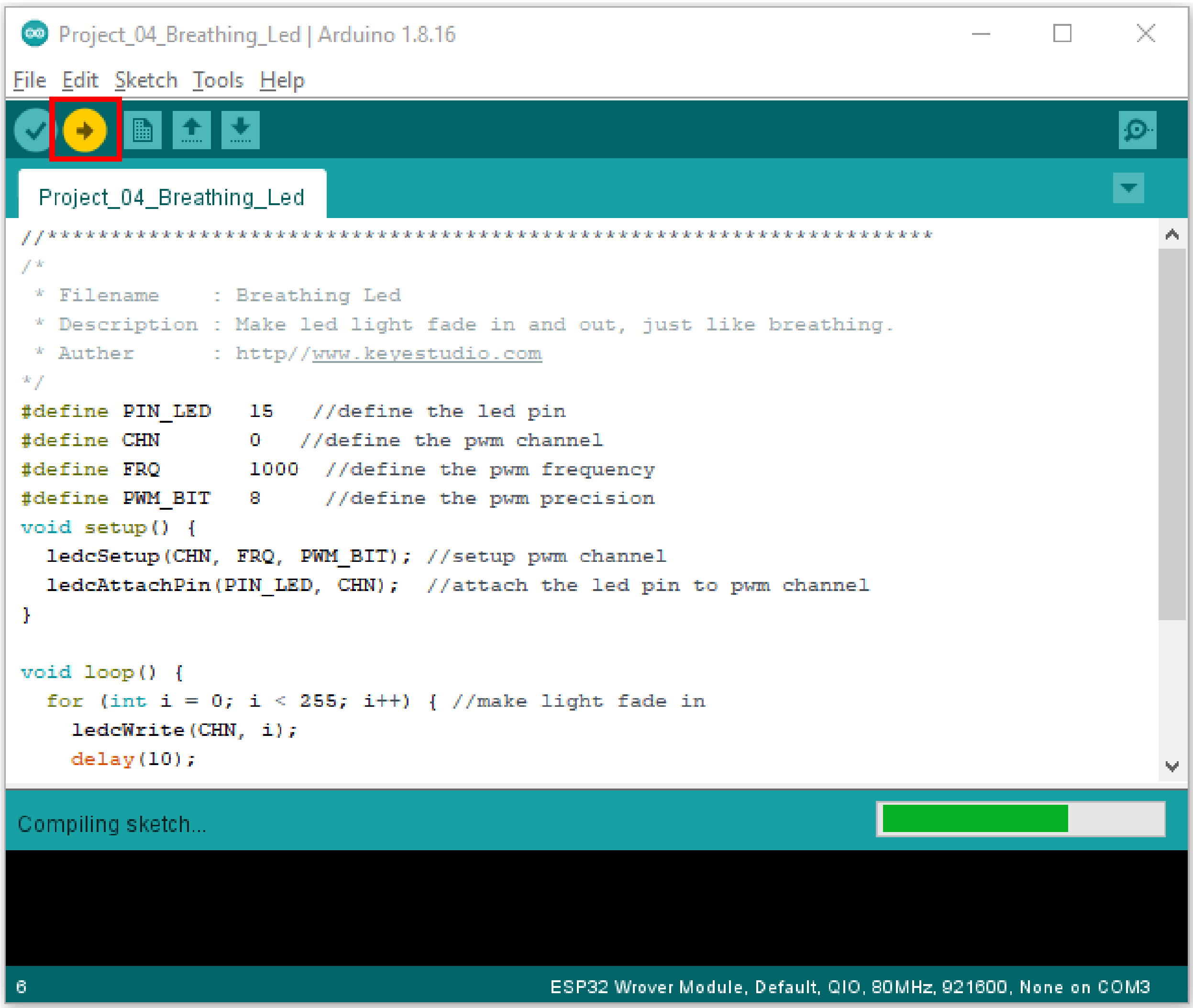
```
 * Description : Make led light fade in and out, just like breathing.
 * Auther      : http//www.keyestudio.com
*/
##define PIN_LED   15   //define the led pin
##define CHN       0   //define the pwm channel
##define FRQ       1000  //define the pwm frequency
##define PWM_BIT   8     //define the 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
    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.

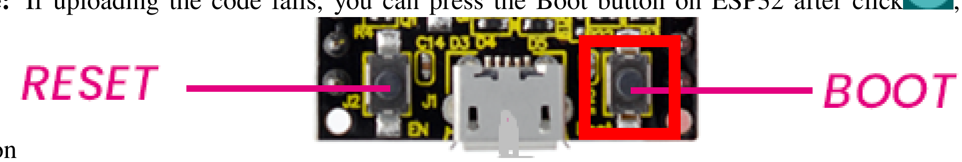
Click “**Tools**” to confirm the board type and port as shown below:



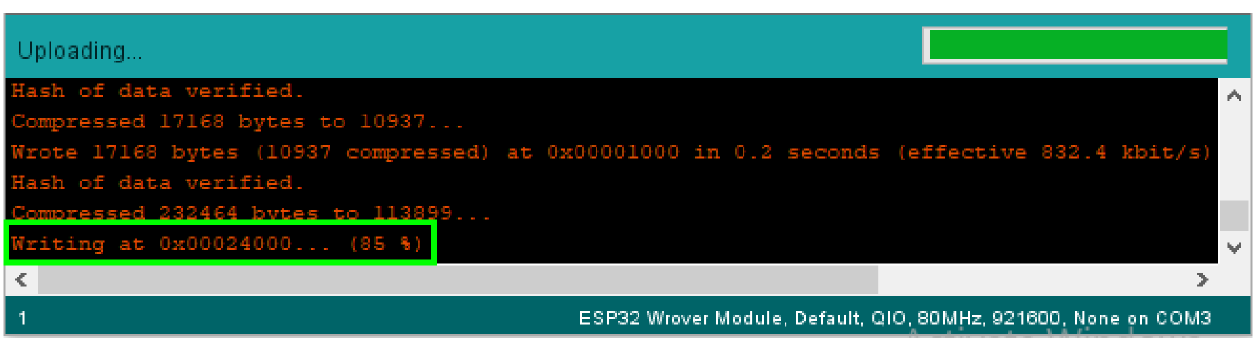
Click  to download the project code to ESP32.



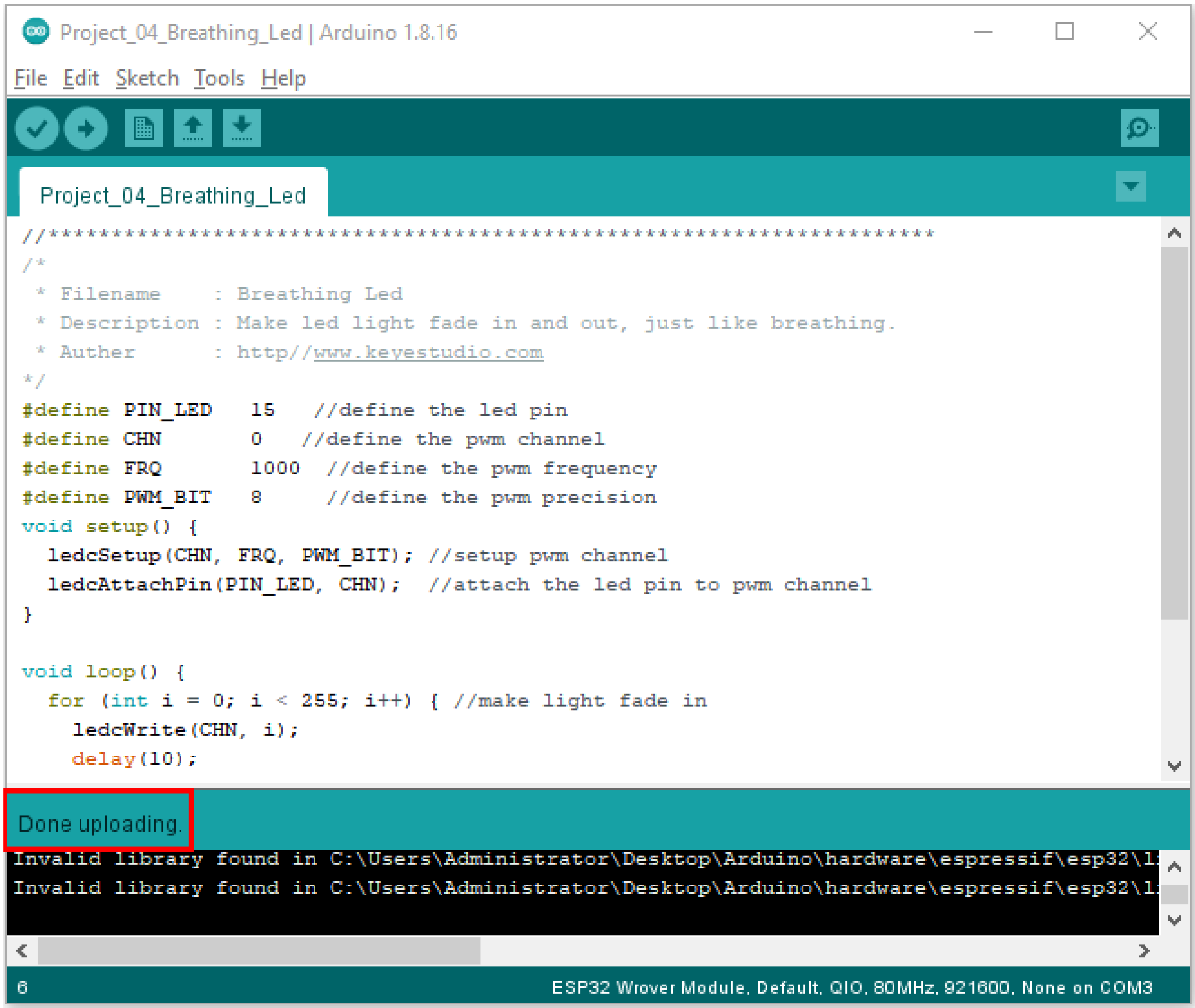
Note: If uploading the code fails, you can press the Boot button on ESP32 after click , and release the Boot button



after the percentage of uploading progress appears, as shown below:



The Project code is uploaded successfully!



6. Project result

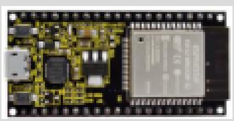
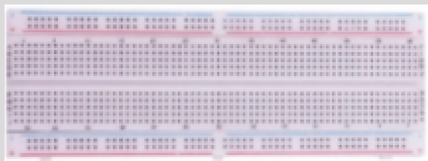
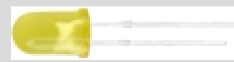
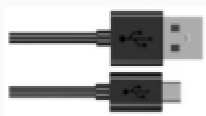
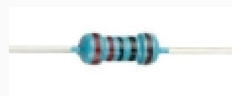
After the project code was uploaded successfully, power up with a USB cable and the LED is turned from ON to OFF and then back from OFF to ON gradually like breathing.

5.6 Project 05Traffic Lights

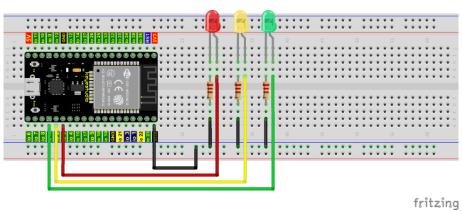
1. Introduction

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

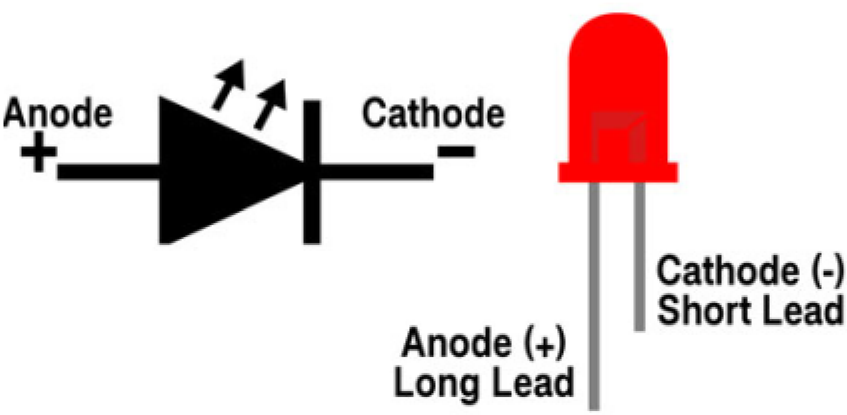
			
ESP32*1	Bread board*1	Red LED*1	Yellow LED*1
			
GreenLED*1	USB Cable*1	220 Resistor*3	Jumper Wires

3. Wiring diagram

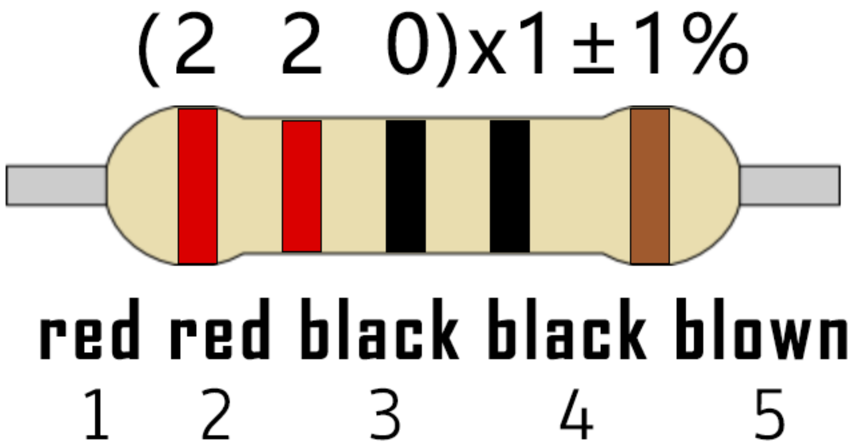


Note:

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 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.
{
   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
```

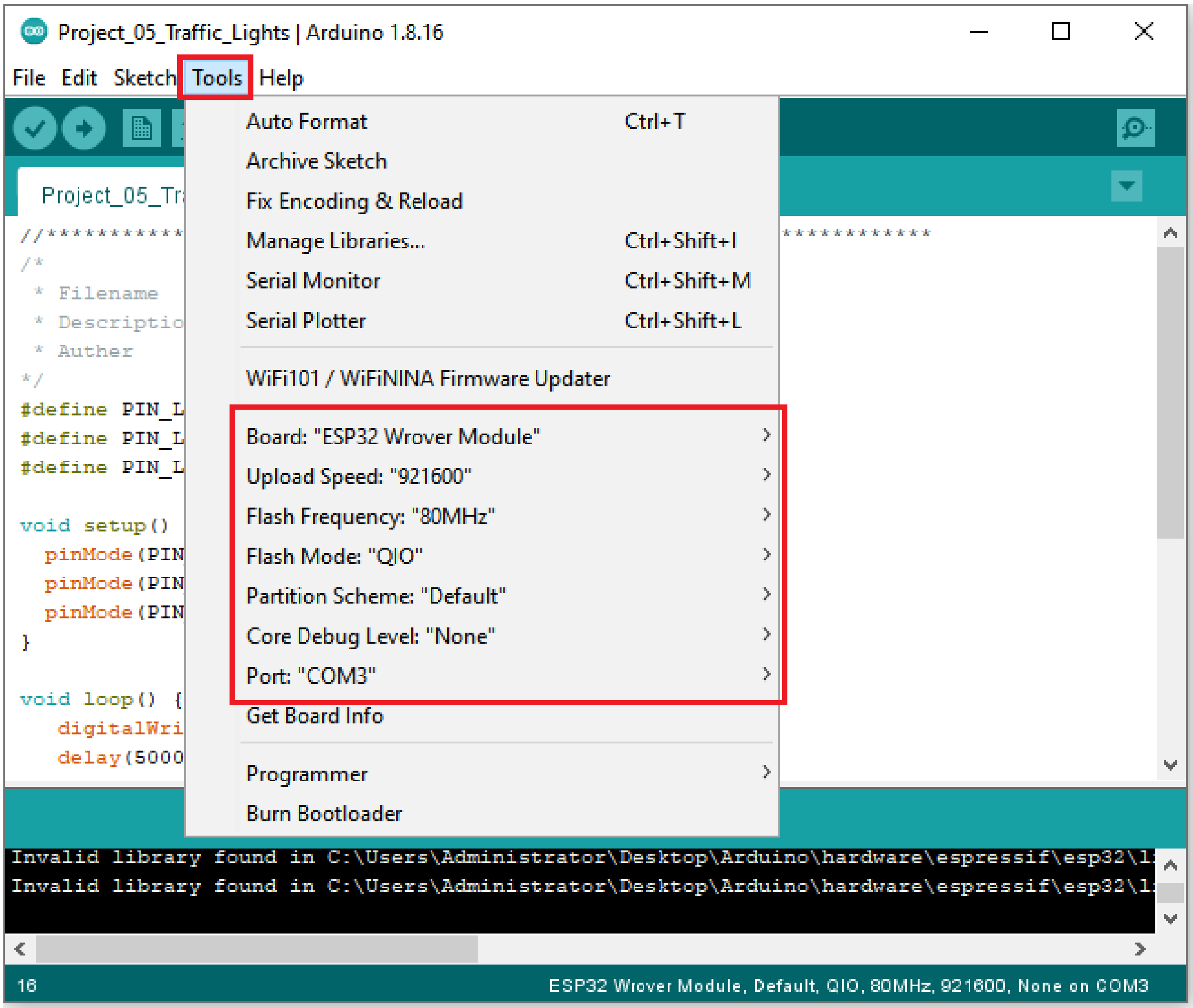
(continues on next page)

(continued from previous page)

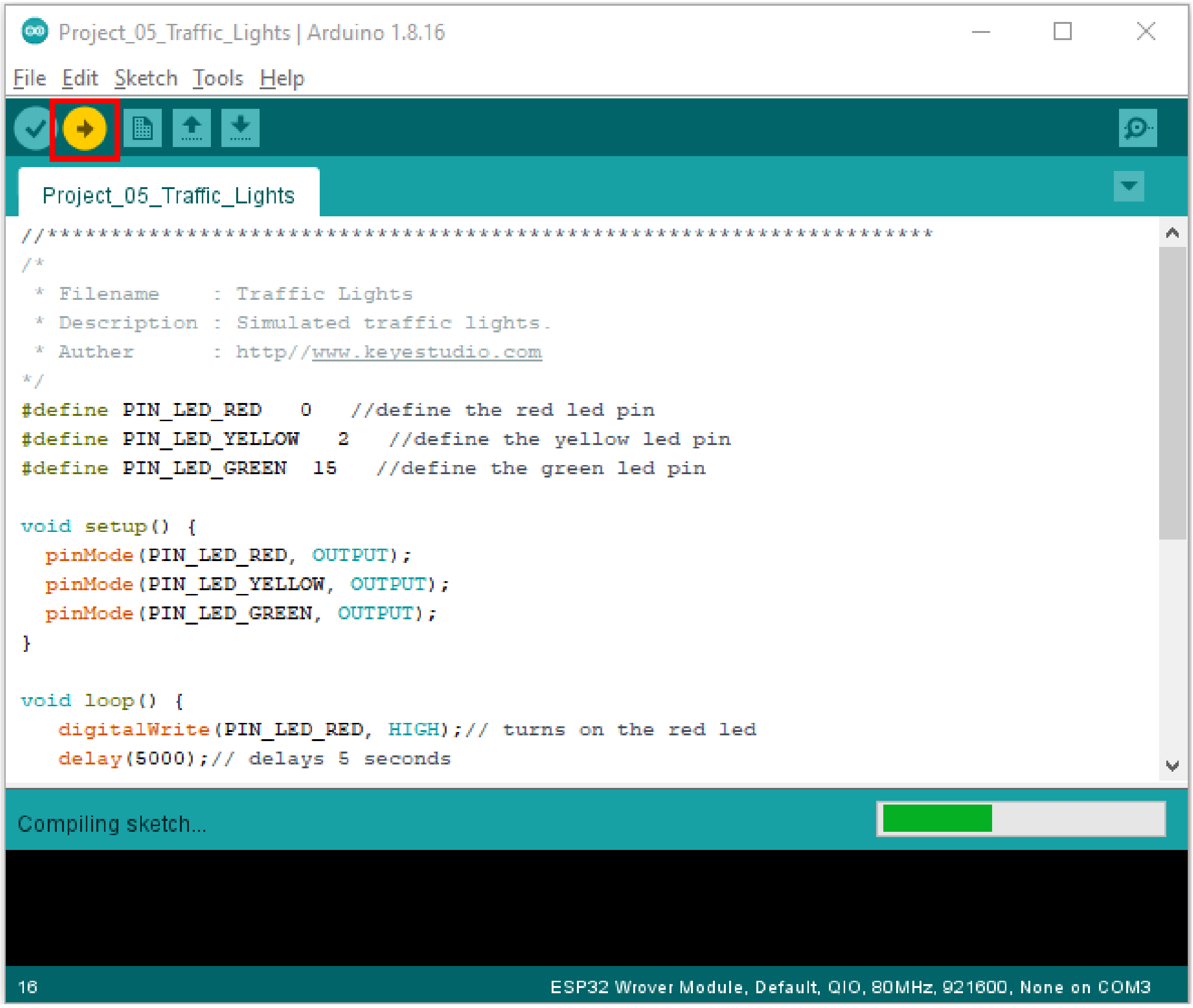
```
    digitalWrite(PIN_LED_RED, LOW);// turns off the red led
}
//*************************************************************************************
```

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

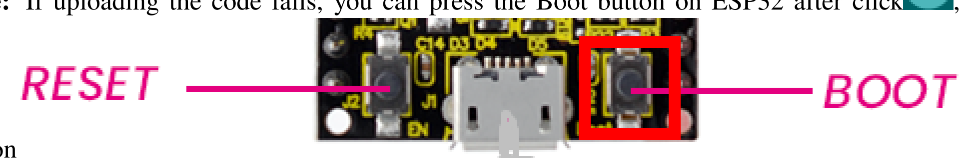
Click "**Tools**" to confirm the board type and port as shown below:



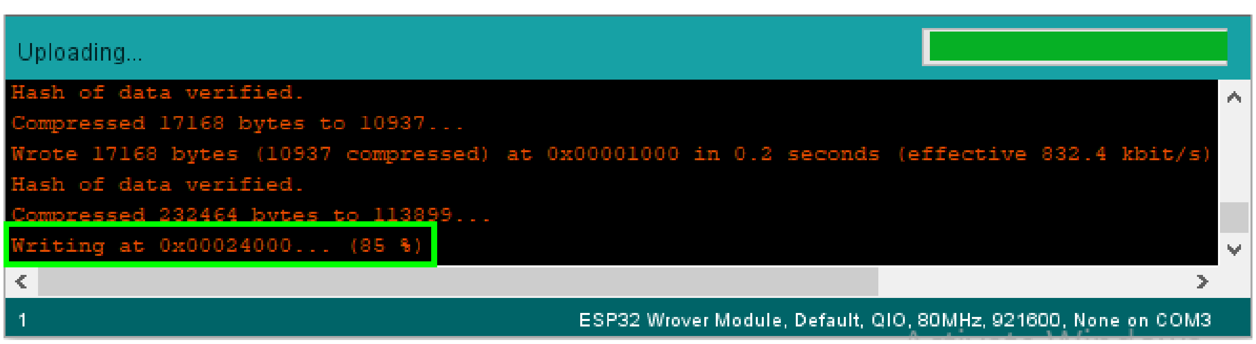
Click  to download the project code to ESP32.



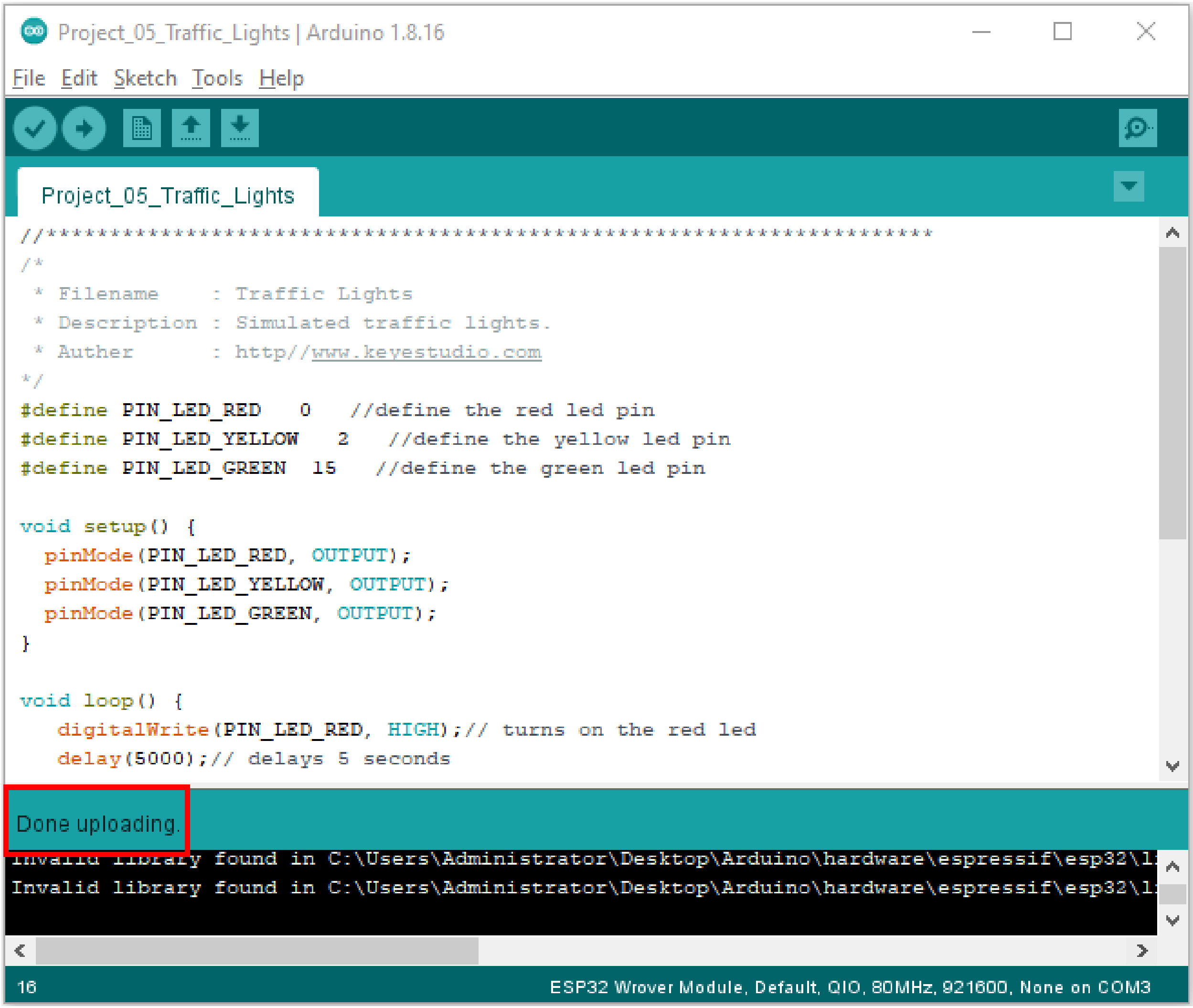
Note: If uploading the code fails, you can press the Boot button on ESP32 after click , and release the Boot button



after the percentage of uploading progress appears, as shown below:



The Project code is uploaded successfully



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

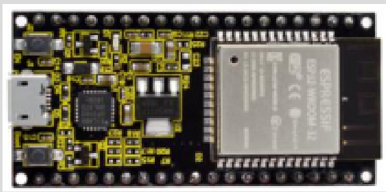
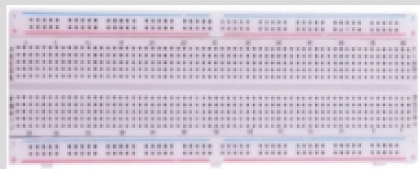
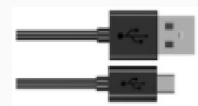
1. Introduction

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.

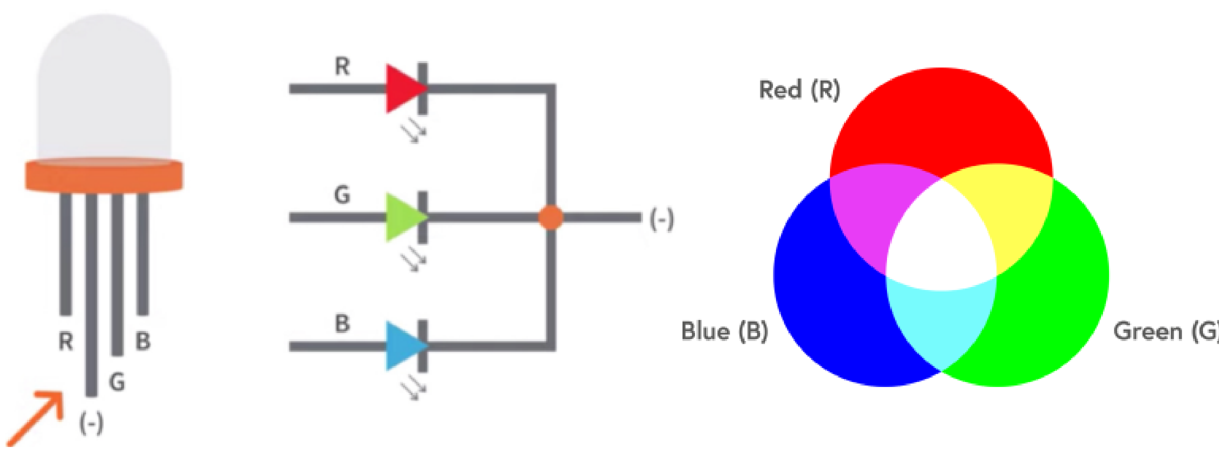
In this project, we will introduce the RGB and show you how to use ESP32 to control the RGB to emit different color light .RGB is pretty basic, but it's also a great way to learn the fundamentals of electronics and coding.

2. Components

			
ESP32*1	Breadboard*1		
			
RGB LED*1	220 Resistor*3	Jumper Wires	USB Cable*1

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.

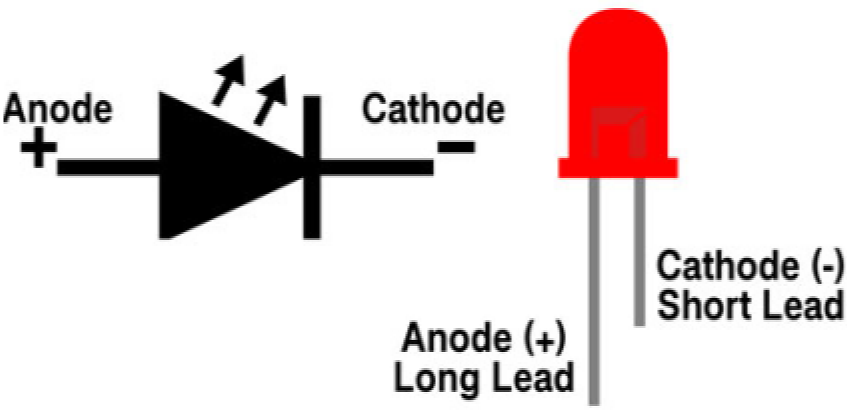
If we use three 10-bit PWM to control the RGB, in theory, we can create 2 10*210*210 =1,073,741,824(1billion) colors through different combinations.

4. Wiring diagram

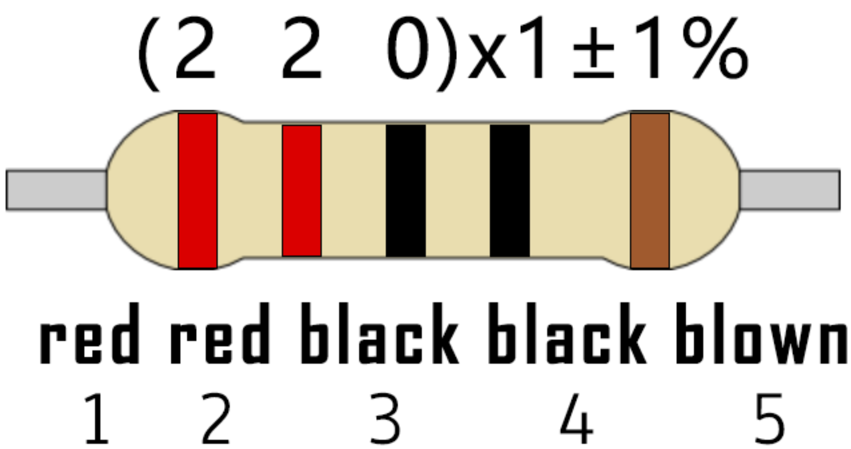
media/f3deb3502985ac8d66e99e4f27b3de1e.png

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 **"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
```

(continues on next page)

(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(0, 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

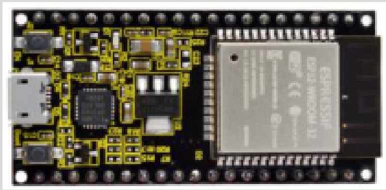
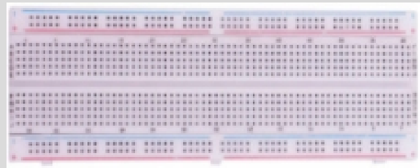
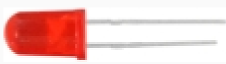
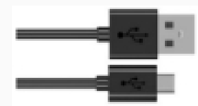
Compile and upload the code to ESP32, after the code is uploaded successfully, power up with a USB cable and you will see that the RGB LED starts to display random colors.

5.8 Project 07: Flowing Water Light

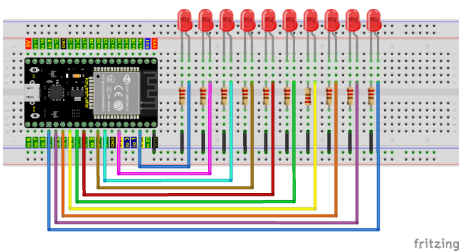
1. Introduction

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

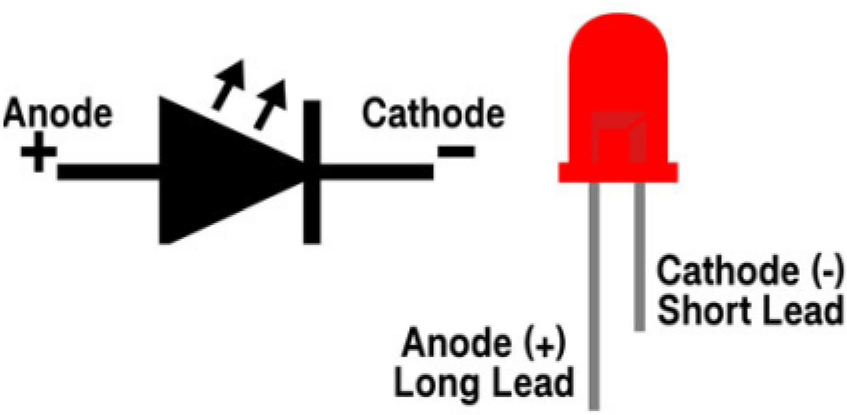
			
ESP32*1	Breadboard*1		
			
Red LED*1	220 Resistor*1	Jumper Wires	USB Cable*1

3. Wiring diagram:

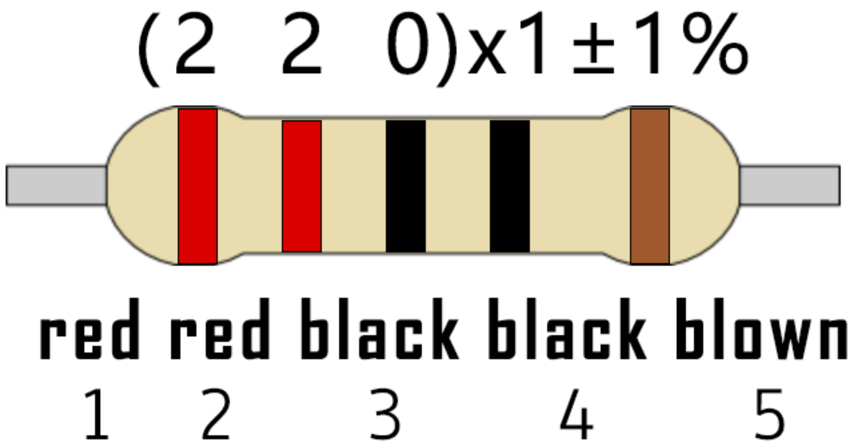


Note:

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);
  }
```

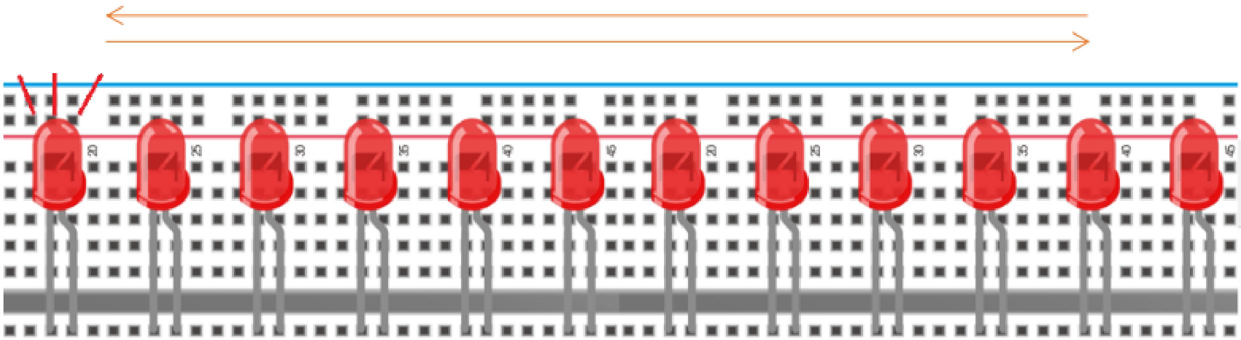
(continues on next page)

(continued from previous page)

```
}
//*************************************************************************
```

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.

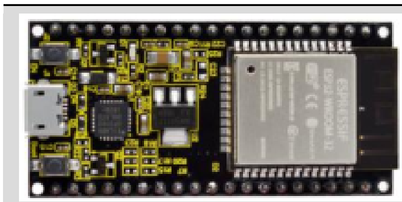
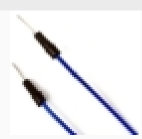
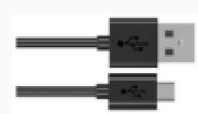


5.9 Project 081-Digit Digital Tube

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

			
ESP32*1	Breadboard*1		
			
1-Digit 7-Segment Display*1	220 Resistor*8	Jumper Wires	USB Cable*1

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). Thedigital 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".

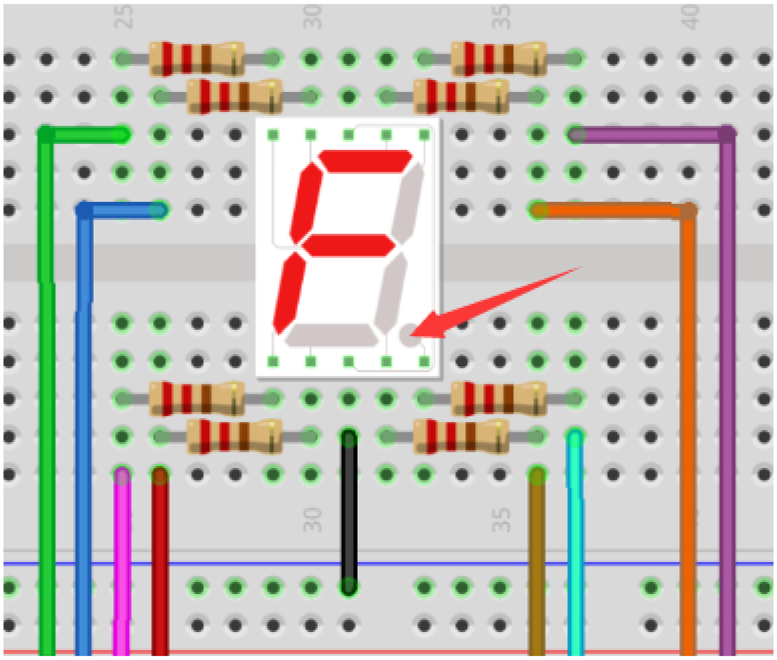
In the common anode 7-segment display, the LED anodes (positive electrodes) of all segments are connected together, so you should connect the common anode to"+5V". To light up a segmented LED, you can set its associated pin to"LOW".

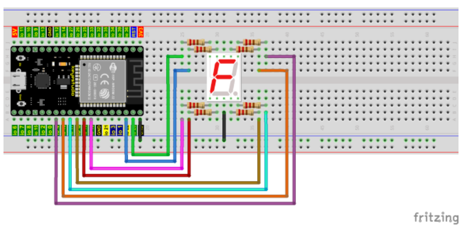
media/28fd057848fbe0e8c8e3362768e7aa44.png

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**.

```
//**********************************************************************
/*
 * 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
```

(continues on next page)

(continued from previous page)

```
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);
```

(continues on next page)

(continued from previous page)

```
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
{
digitalWrite(a,HIGH);
digitalWrite(b,HIGH);
```

(continues on next page)

(continued from previous page)

```
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

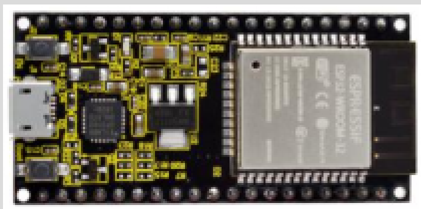
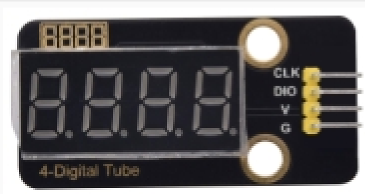
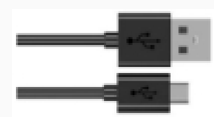
Compile and upload the code to ESP32, after the code is uploaded successfully, power up with a USB cable and you will see that the 1-Digit 7-Segment Display will display numbers from 9 to 0.

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

		
ESP32*1	Breadboard*1	
		
4-digit 7-segment display Module*1	M-F Dupont Wires	USB Cable*1

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.

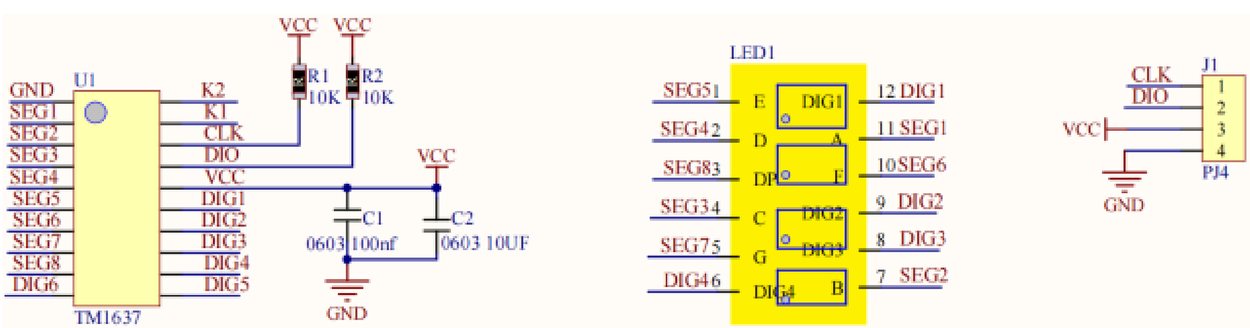
Specifications of 4-bit 7-segment display module:

Working voltage: DC 3.3V-5V

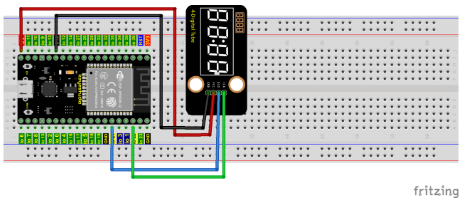
Maximum current: 100MA

Maximum power: 0.5W

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



4. Wiring diagram



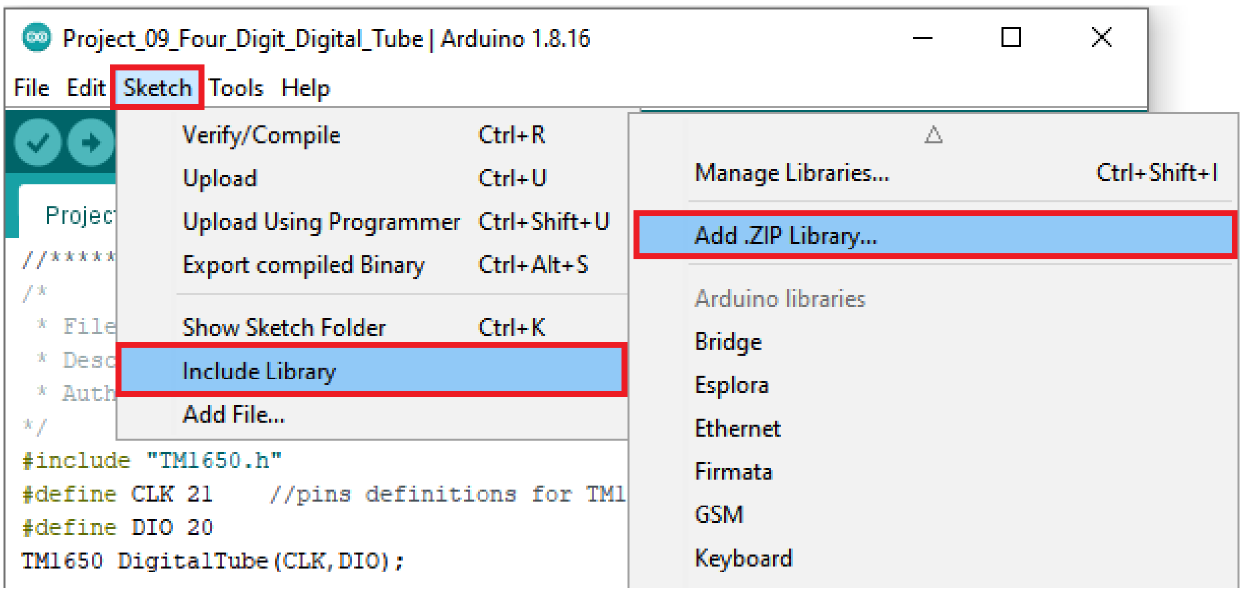
5.Adding the TM1650 library

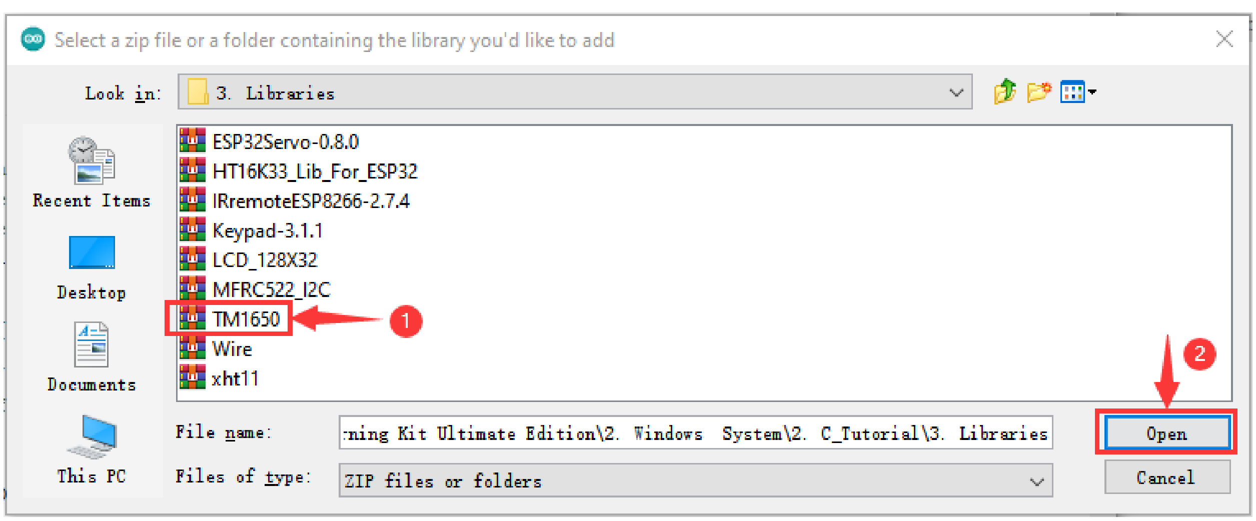
If you have not downloaded the library file, please click on the link to download it:Download Arduino Libraries

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"→"Include Library"→"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 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 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++){
    DigitalTube.displayBit(1,num);
    DigitalTube.displayBit(2,num);
    DigitalTube.displayBit(3,num);
    DigitalTube.displayBit(4,num);
    delay(1000);
  }
 }
//************************************************************************************
```

7. Project result

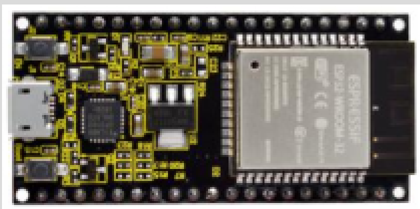
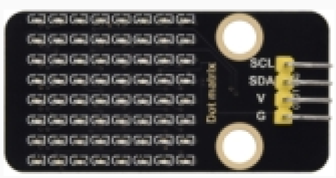
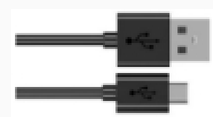
Compile and upload the code to ESP32, after the code is uploaded successfully, power up with a USB cable and you will see that 4-digit 7-segment display displays four digitsand repeat these actions in an infinite loop.

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

		
ESP32*1	Breadboard*1	
		
88 *dot matrix module*1	M-F Dupont Wires	USB Cable*1

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.

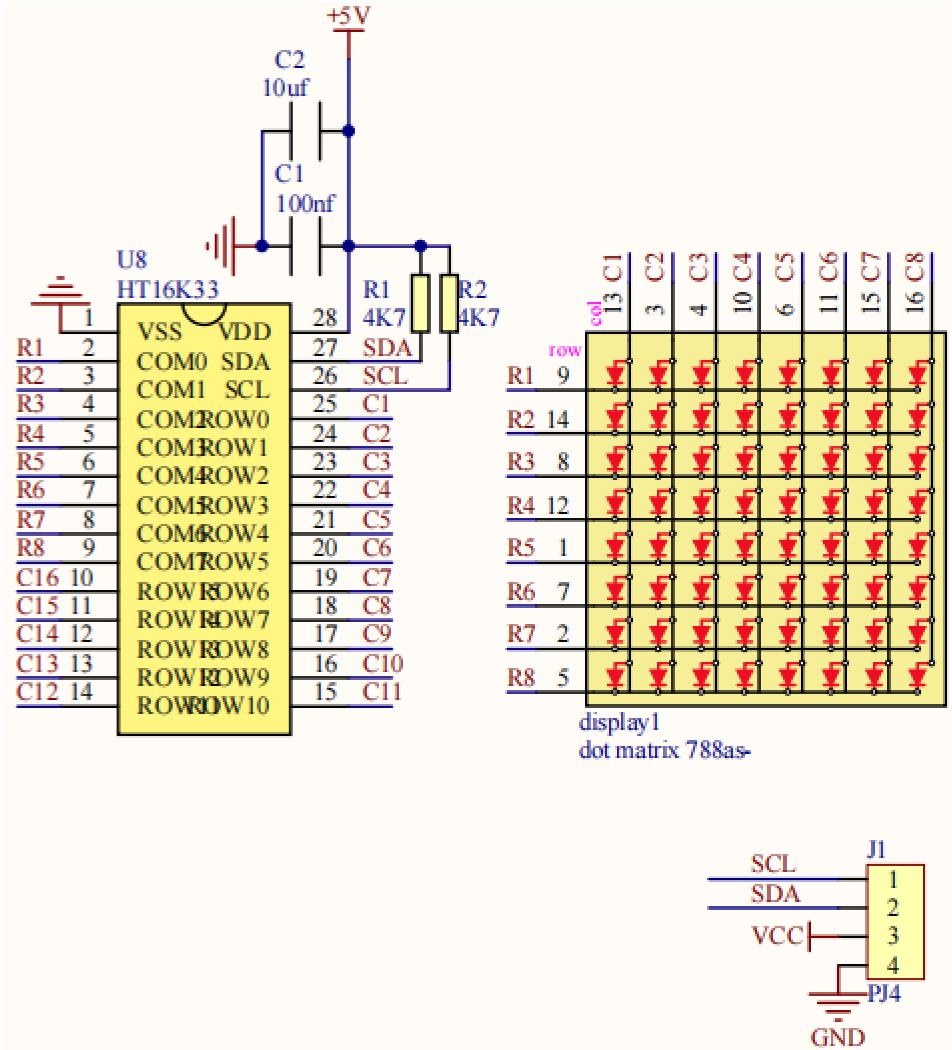
Specifications of 8*8 dot matrix module

Working voltage: DC 5V

Current: 200MA

Maximum power: 1W

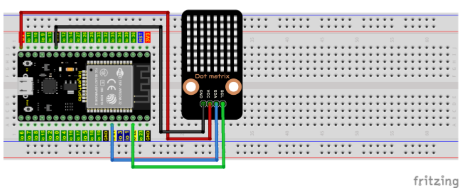
Schematic diagram of 8*8 dot matrix module



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)
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



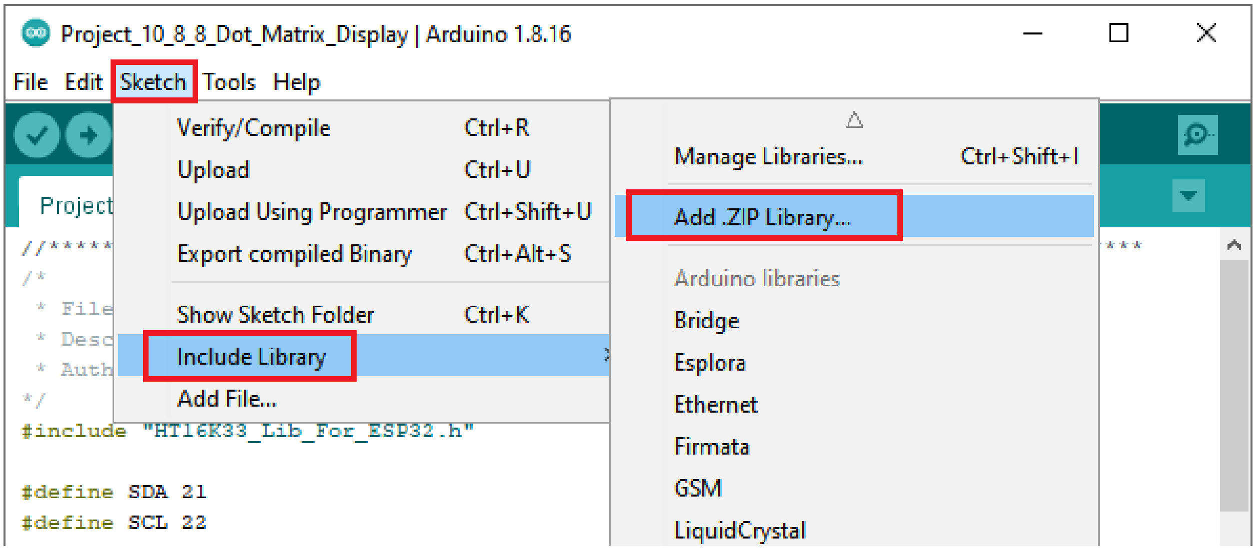
5. Adding the HT16K33_Lib_For_ESP32 library

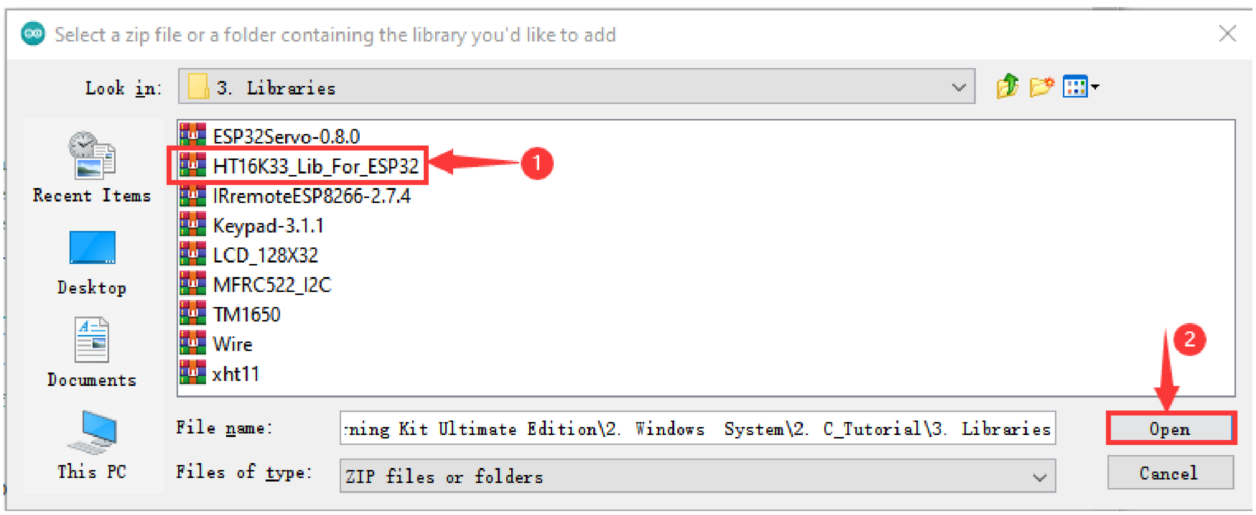
   If you have not downloaded the library file, please click on the link to download it:Download Arduino Libraries

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"→"Include Library"→"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".





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();
```

(continues on next page)

(continued from previous page)

```
//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);
  }
}
//************************************************************************************
```

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.

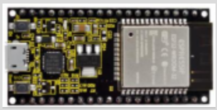
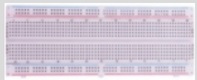
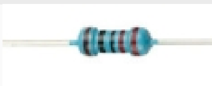
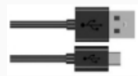
5.12 Project 1174HC595N Control 8 LEDs

1. Introduction

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

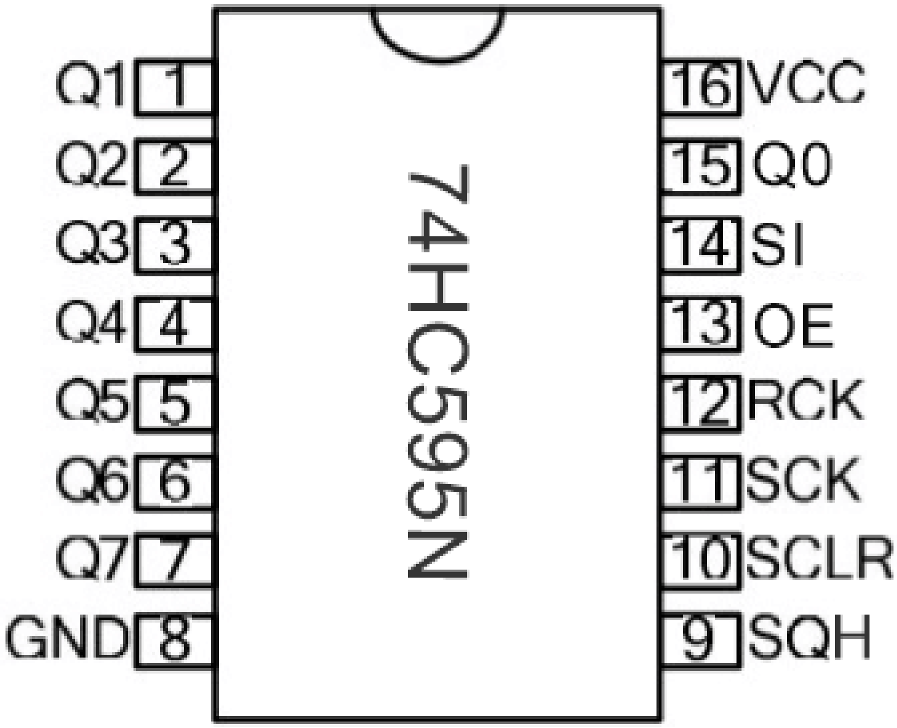
With only 32 IO ports on ESP32, how do we light up a lot of leds? Sometimes it is possible to run out of pins on the ESP32, and you need to extend it with the shift register.You can use the 74HC595N chip to control 8 outputs at a time, taking up only a few pins on your microcontroller. In addition, you 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\Temp\ksohtml16172
ESP32*1	Breadboard*1	74HC595N chip*1	Jumper Wires
			
220Resistor*8	Red LED*8	USB Cable*1	

3. Component knowledge

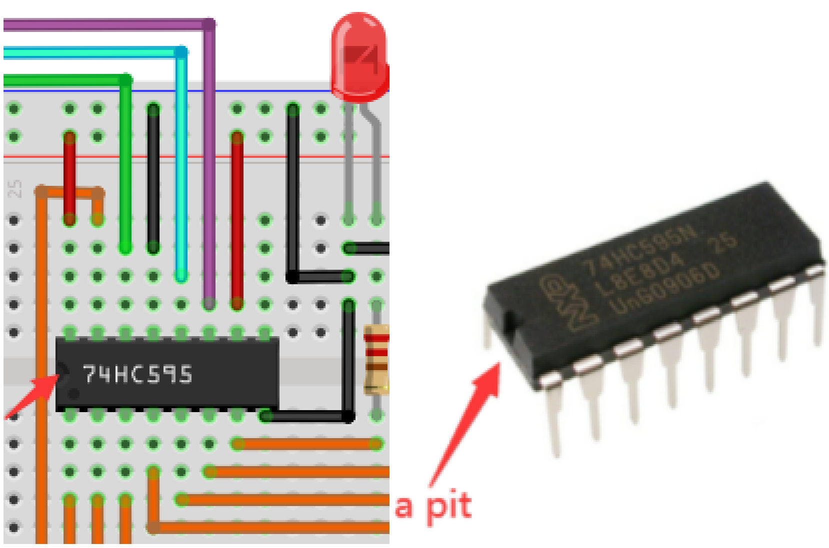
74HC595N Chip: The 74HC595 chip is used to convert serial data into parallel data. A 74HC595 chip can convert the serial data of one byte into 8 bits, and send its corresponding level to each of the 8 ports correspondingly. With this characteristic, the 74HC595 chip can be used to expand the IO ports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

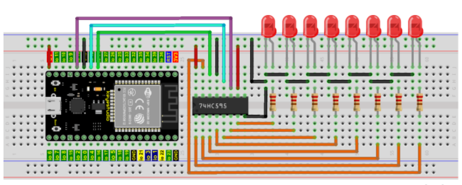


The ports of the 74HC595 chip are described as follows

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 LED bar graph.
  // This variable is assigned to 0x01, that is binary 00000001, which indicates only one LED light on.
  byte x = 0x01;    // 0b 0000 0001
  for (int j = 0; j < 8; j++) { // Let led light up from right to left
    writeTo595(LSBFIRST, x);
    x <<= 1; // make the variable move one bit to left once, then the bright LED move 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 output port.
  digitalWrite(latchPin, HIGH);
}
```

(continues on next page)

(continued from previous page)

```
//*************************************************************************************
```

6. Project result

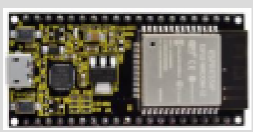
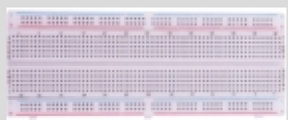
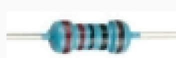
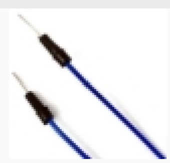
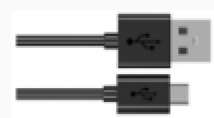
Compile and upload the code to ESP32, after the code is uploaded successfully, power up with a USB cable and you will see that the 8 LEDs start flashing in flowing water mode.

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

			
ESP32*1	Breadboard*1	Active buzzer*1	
			
NPN transistor(S8050)*1	1kResistor*1	Jumper Wires	USB Cable*1

3. Component knowledge

media/11ec5ddc982db9928341e858aab94652.png

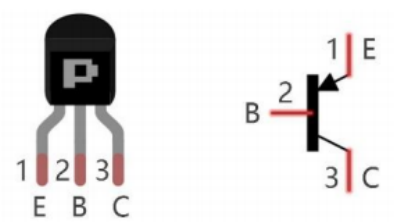
Active buzzer: Active buzzer inside has a simple oscillator circuit, which can convert constant direct current into a certain frequency pulse signal. Once active buzzer receives a high level, it will produce sound. Passive buzzer is an internal without vibration source integrated electronic buzzer, it 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:

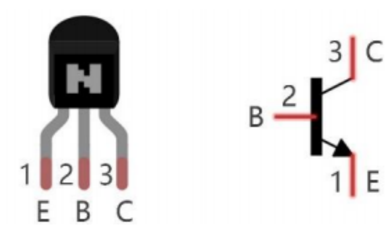
media/0f9825969867ac2d65bb1a19ed0ad2ab.png

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,





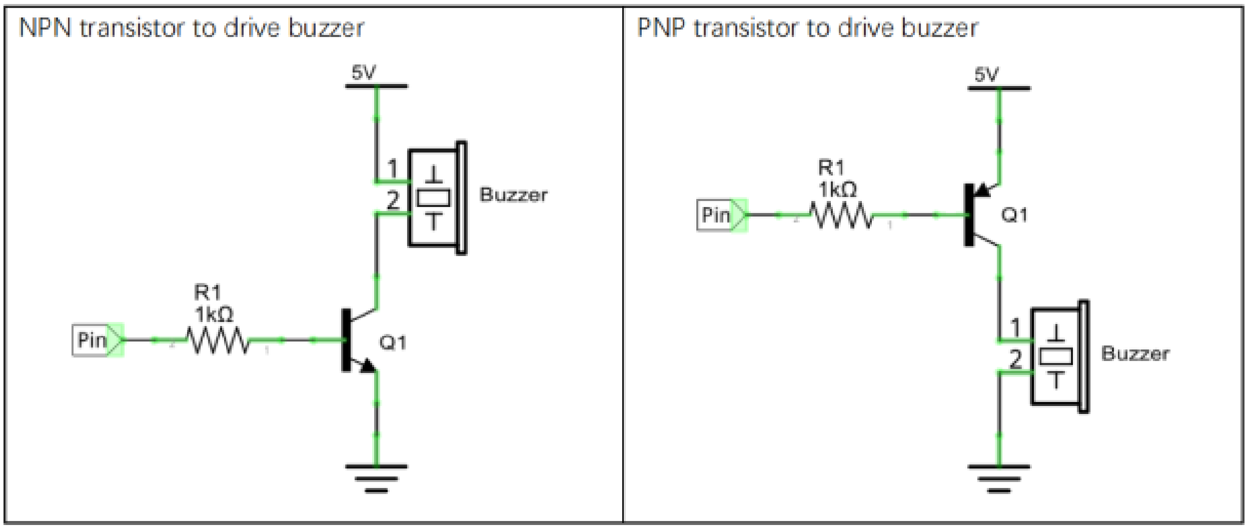
PNP transistor NPN transistor

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

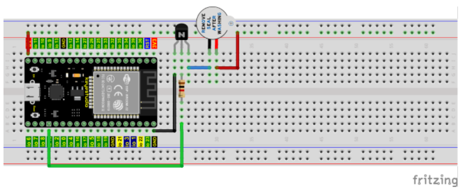
Based on the transistor's characteristics, it is often used as a switch in digital circuits. As micro-controller's capacity to output current is very weak, we will use transistor to amplify current and drive large-current components.

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);
```

(continues on next page)

(continued from previous page)

```
}
//*************************************************************************************
```

6. Project result

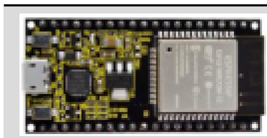
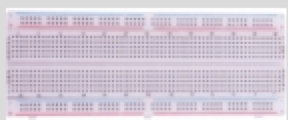
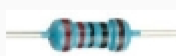
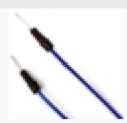
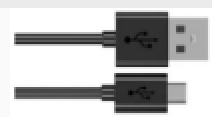
Compile and upload the code to ESP32, after the code is uploaded successfully, power up with a USB cable and you will see that the active buzzer beeps.

5.14 Project 13Passive Buzzer

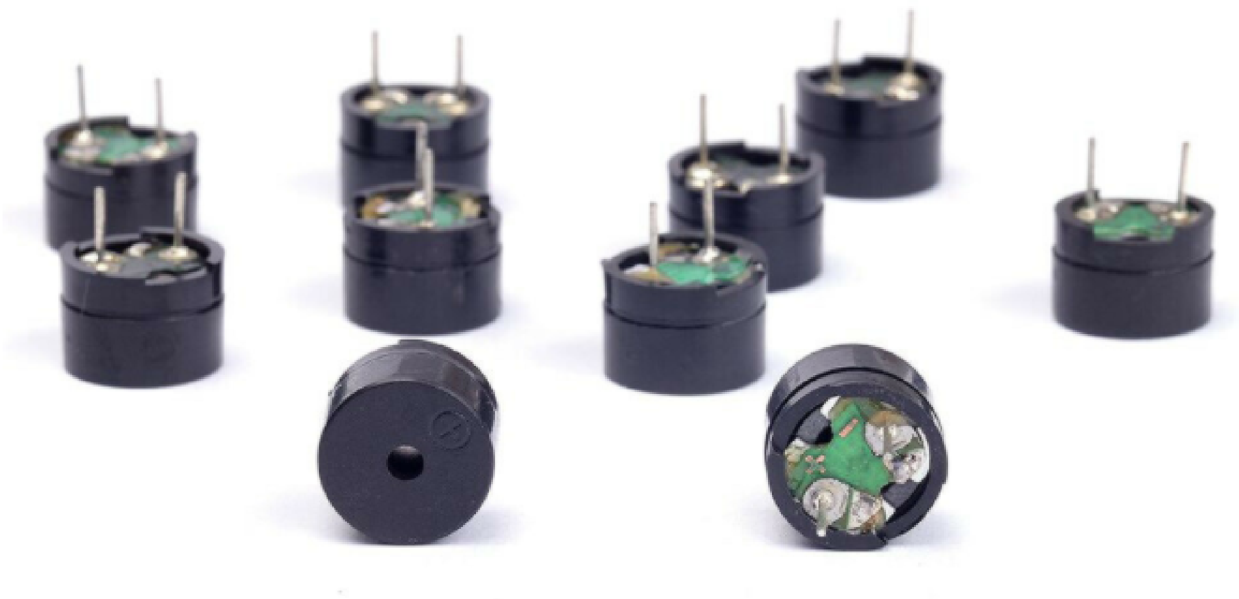
1. Introduction:

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

			
ESP32*1	Breadboard*1	Passive Buzzer *1	
			
NPN transistor(S8050)*1	1kResistor*1	Jumper Wires	USB Cable*1

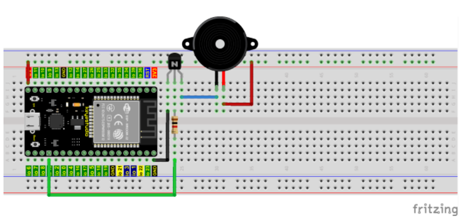
3. Component knowledge



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,␣
↪196, 330, 392,440, 349, 392, 330, 262, 294, 247, 392, 370, 330, 311, 330, 208, 220,␣
↪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

int noteDurations[] = {8,4,4,8,4,2,2,3,3,3,4,4,8,4,8,8,8,4,8,4,3,8,8,3,3,3,3,4,4,8,4,8,8,
↪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,
↪4,8,4,8,4,8,2,8,4,4,8,4,1,8,4,4,8,4,8,4,8,2};
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)

  {
      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)

```
      delay(noteDuration * 1.30); //delay
  }
}
//*************************************************************************
```

6. Project result

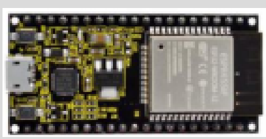
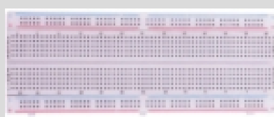
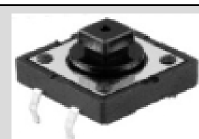
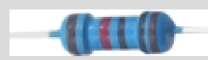
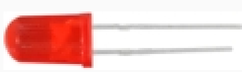
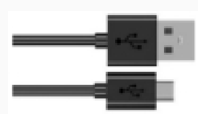
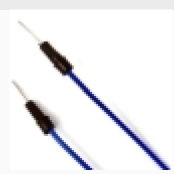
Compile and upload the code to ESP32, after the code is uploaded successfully, power up with a USB cable and you will see that the passive buzzer plays music.

5.15 Project 14: Mini Table Lamp

1. Introduction

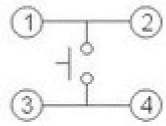
Do you know that the ESP32 can light up an LED when you press a button? In this project, we will use ESP32a button switch and an LED to make a mini table lamp.

2. Components

				
ESP32*1	Breadboard*1	Button*1	10K Resistor*1	
				
Red LED*1	220 Resistor*1	USB Cable*1	Jumper Wires	Button Cap*1

3.Component knowledge

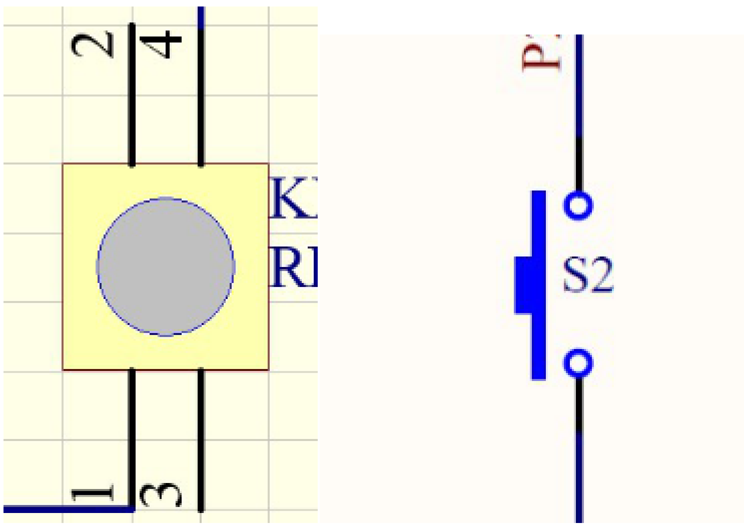
Button: A button can control the circuit on and off, the button is plugged into a circuit, the circuit is disconnected when the button is not pressed. The circuit works when you press the button, but breaks again when you release it. Why does it only work when you press it? It starts from the internal 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.

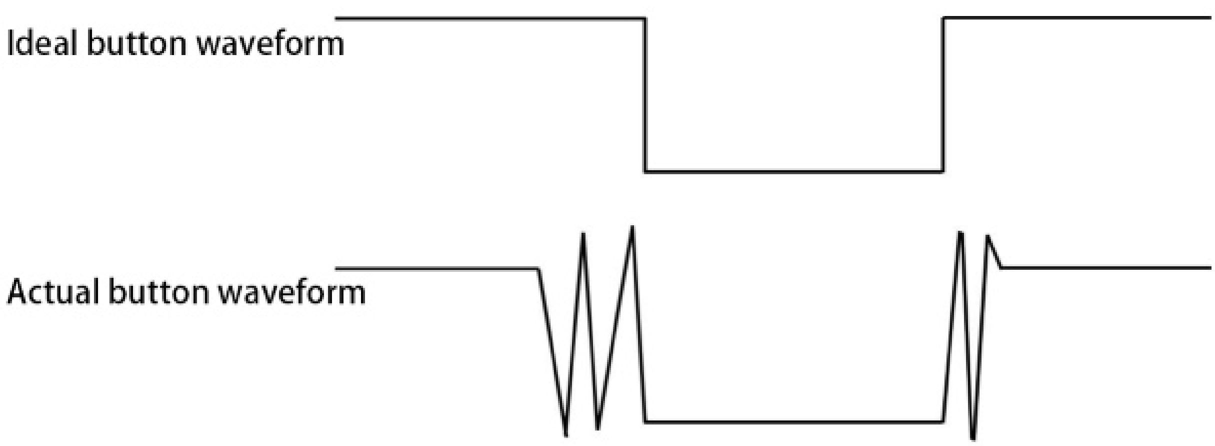
Schematic diagram of the button:



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.

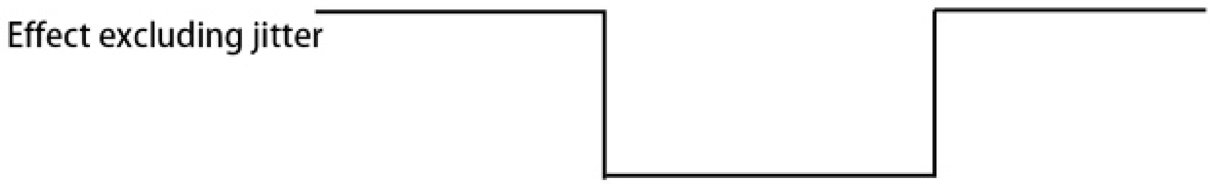


How to eliminate the [shake](javascript:;)?

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.

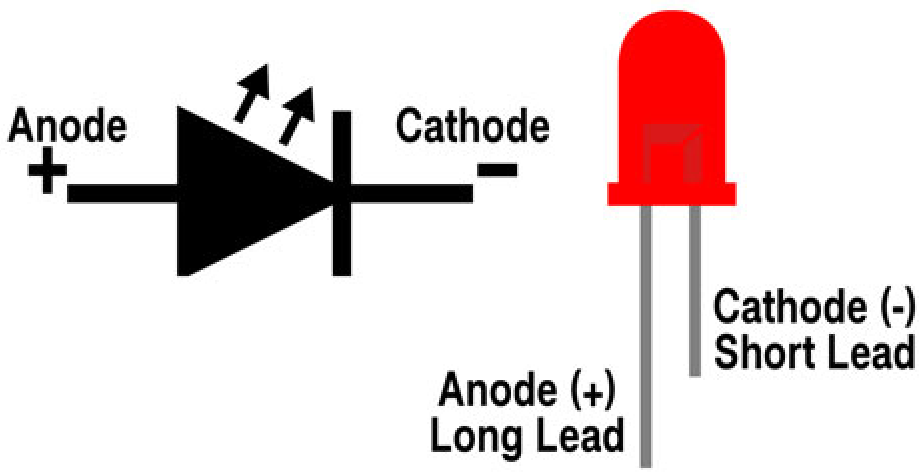


4. Wiring Diagram

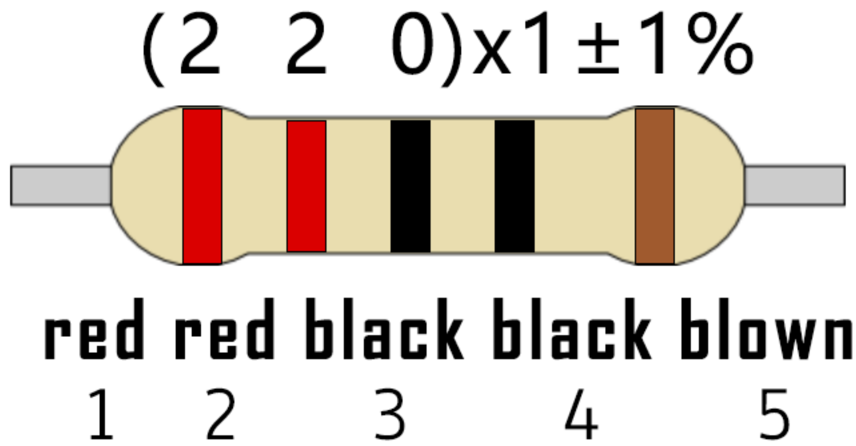
media/a5b85f1e1f5714afbe4730b1265e3a15.png

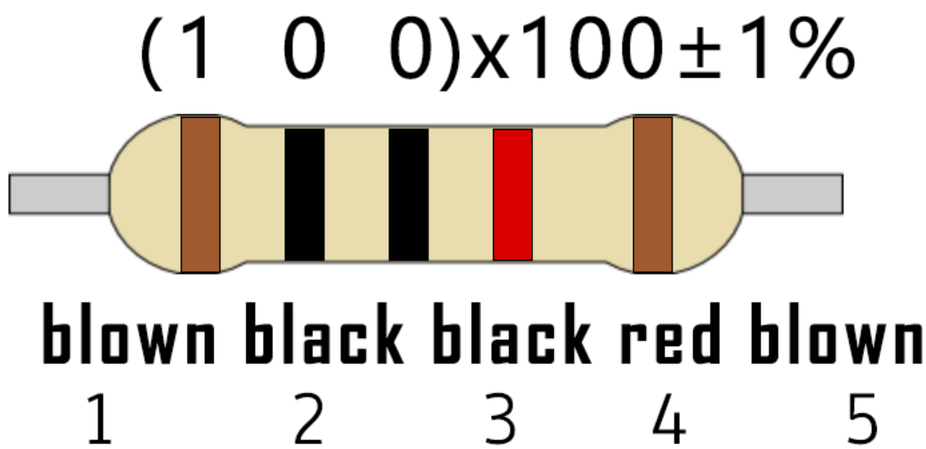
Note:

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 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);
}
```

(continues on next page)

(continued from previous page)

```
// 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);
}
//*************************************************************************
```

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?

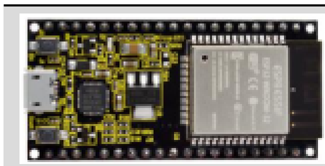
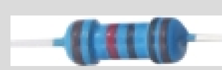
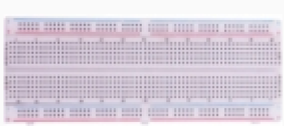
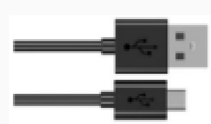
5.16 Project 15Tilt and LED

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

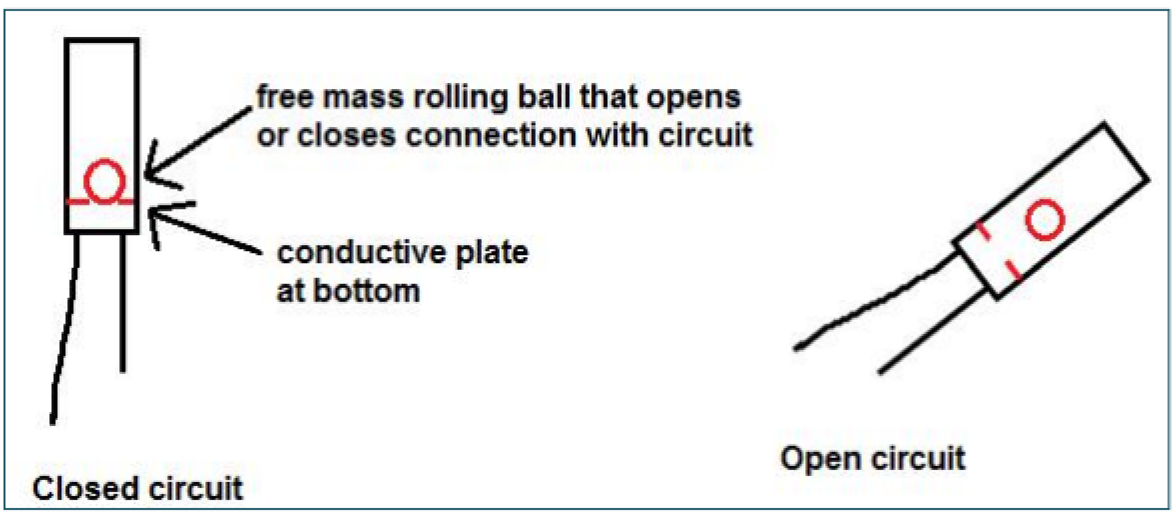
			
ESP32*1	Tilt Switch*1	Red LED*4	10K Resistor*1
			
Breadboard*1	220 Resistor*4	USB Cable*1	Jumper Wires

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:

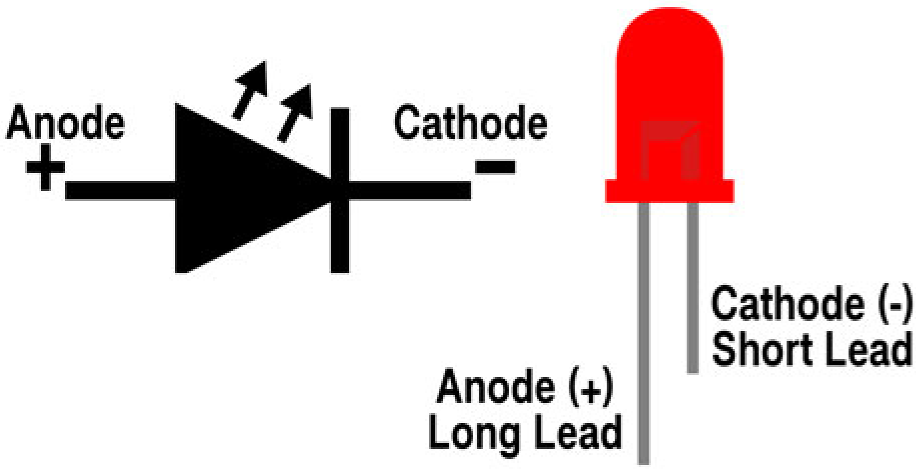


4. Wiring Diagram

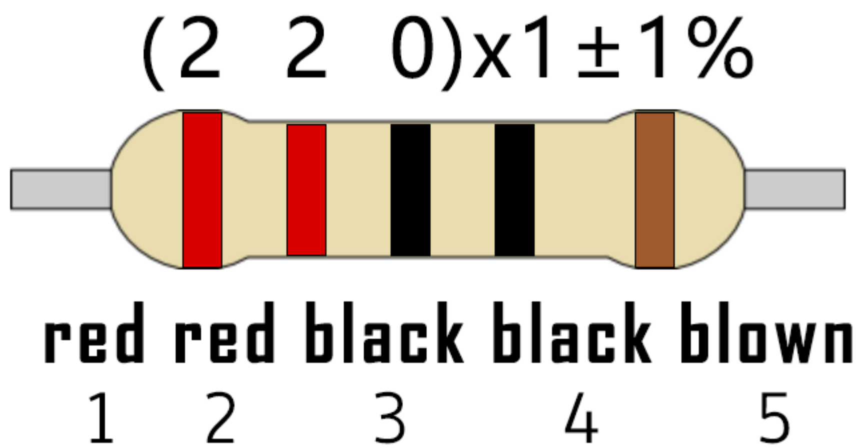
media/a46c0b8be898ba596308ce56993c26ba.png

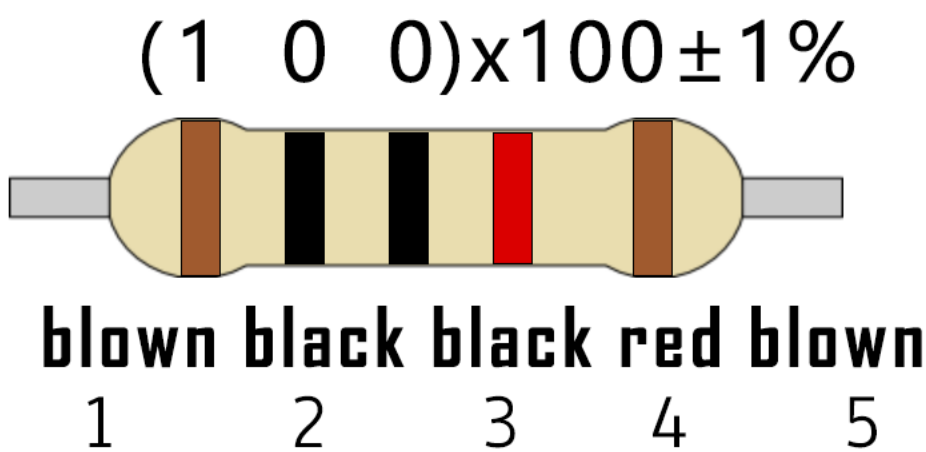
Note:

How to connect the LED



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.
 * Auther      : http//www.keyestudio.com
*/
##define SWITCH_PIN  15  // the tilt switch is connected to Pin15
byte switch_state = 0;
void setup()
{
     for(int i=16;i<20;i++)
  {
        pinMode(i, OUTPUT);
  }
    pinMode(SWITCH_PIN, INPUT);
 for(int i=16;i<20;i++)
  {
    digitalWrite(i,0);
  }
  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++)
  {
    digitalWrite(i,1);
    delay(500);
  }
  }
   if (switch_state == 1)
 {
   for(int i=19;i>15;i--)
   {
    digitalWrite(i,0);
    delay(500);
   }
  }
}
//*************************************************************************************
```

6. Project result

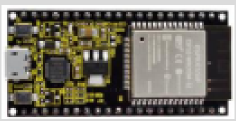
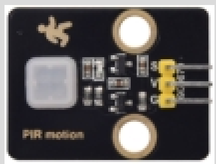
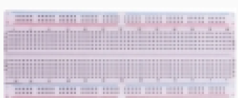
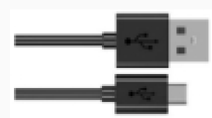
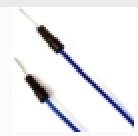
Compile and upload the code to ESP32, after the code is uploaded successfully, power up with a USB cable and you will see that when you tilt the breadboard to an angle, the LEDs will light up one by one. When you turn the breadboard to the original angle, the LEDs will turn off one by one. Like the hourglass,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

5.17 Project 16Burglar Alarm

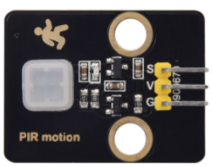
1. Introduction

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

				
ESP32*1	Human Body Infrared Sensor*1	Active Buzzer*1	Red LED*1	
				
Breadboard*1	M-F Dupont Wires	220Resistor*1	USB Cable*1	Jumper Wires

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

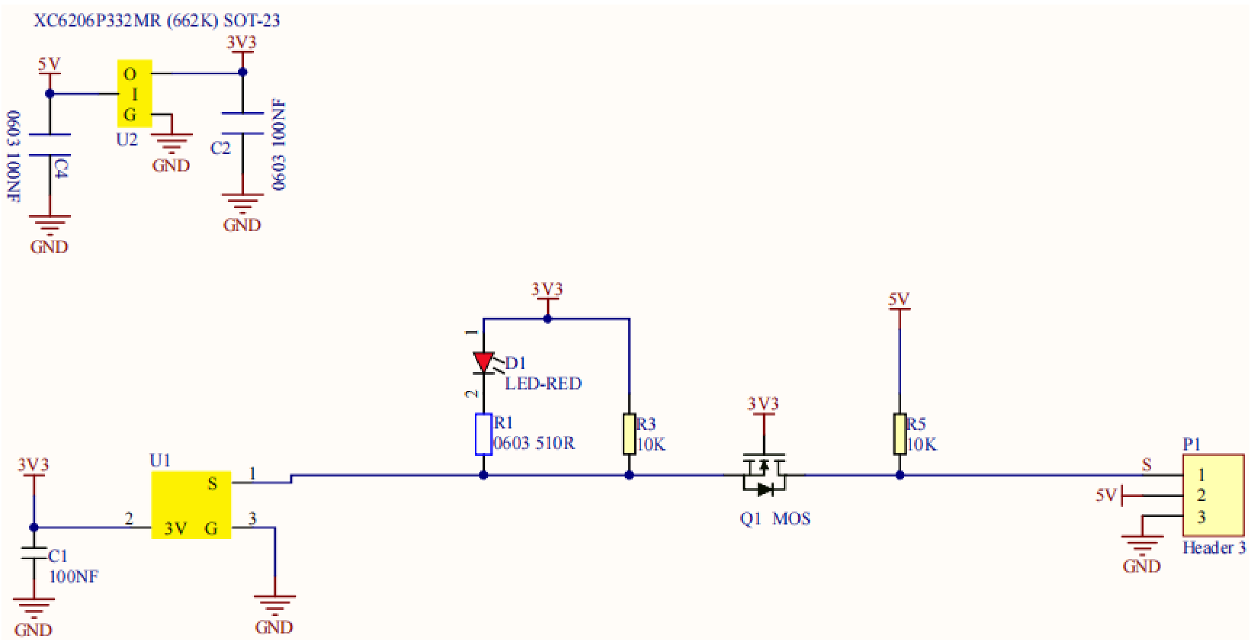
Detection Angle: about 100 degrees

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

media/67fd78fc542f0e7c232d96a23fb90120.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 16Burglar Alarm\Project_16_Burglar_Alarm"**.

```
//**********************************************************************
/*
 * Filename    : Burglar Alarm
 * Description : Human infrared sensor buzzer and LED to simulate burglar alarm.
 * Auther      : http//www.keyestudio.com
*/
##define buzzerPin   2   // the pin of the buzzer
```

(continues on next page)

(continued from previous page)

```
##define ledPin   0     // the pin of the PIR motion sensor
##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␣
↪off buzzers
   digitalWrite(ledPin, LOW);  // the led turn off
 }
}
//**********************************************************************************
```

6. Project result

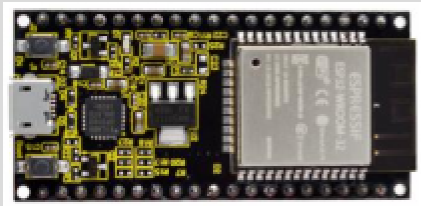
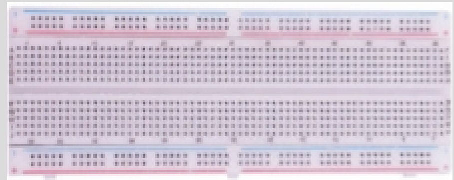
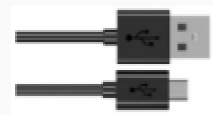
Compile and upload the code to ESP32, after the code is uploaded successfully, power up with a USB cable and you will see that if the human body infrared sensor detects someone moving nearby, the buzzer will continuously issue an alarm and the LED will continuously flash.

5.18 Project 17 I2C 128×32 LCD

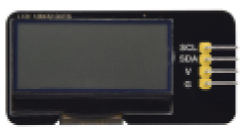
1. Introduction

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

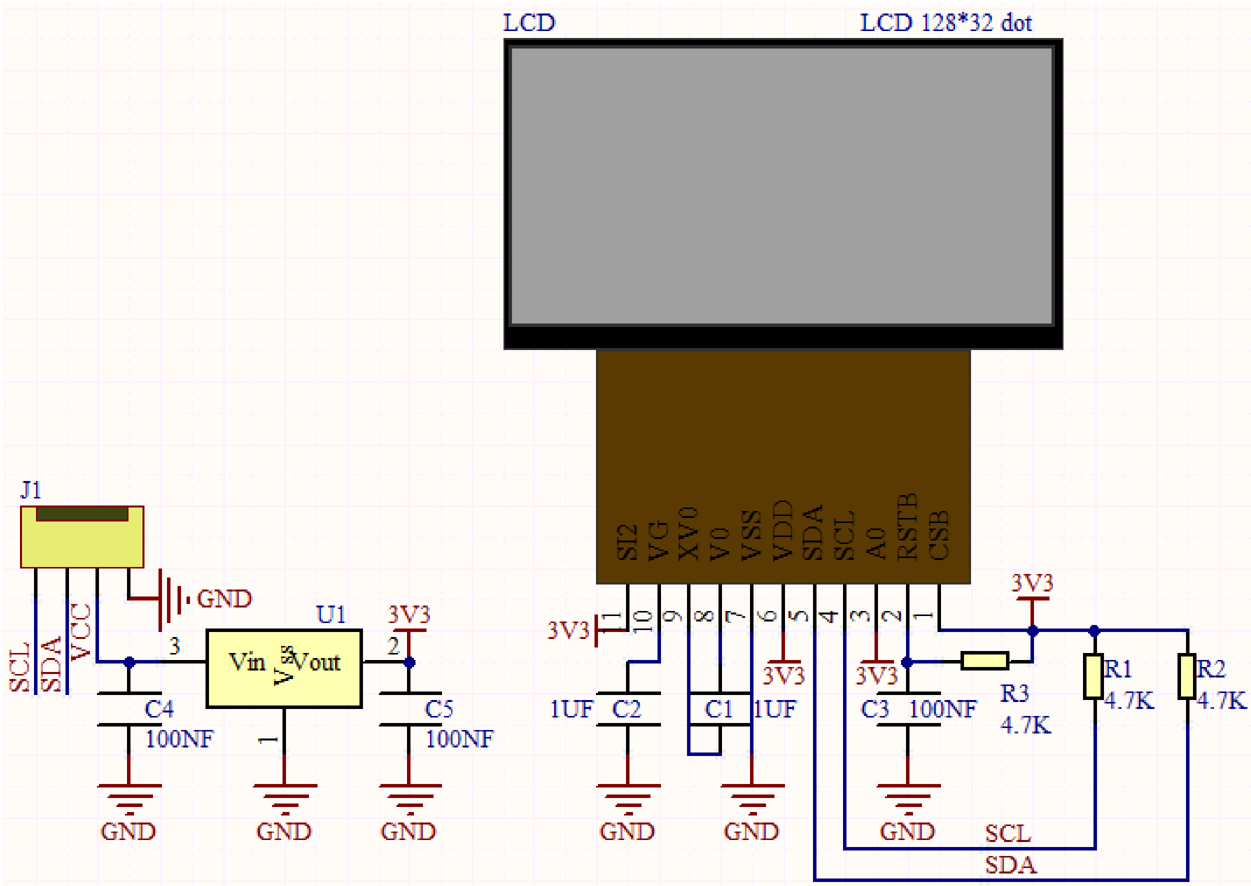
		
ESP32*1	Breadboard*1	
		
LCD_128X32_DOT*1	M-F Dupont Wires	USB Cable*1

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

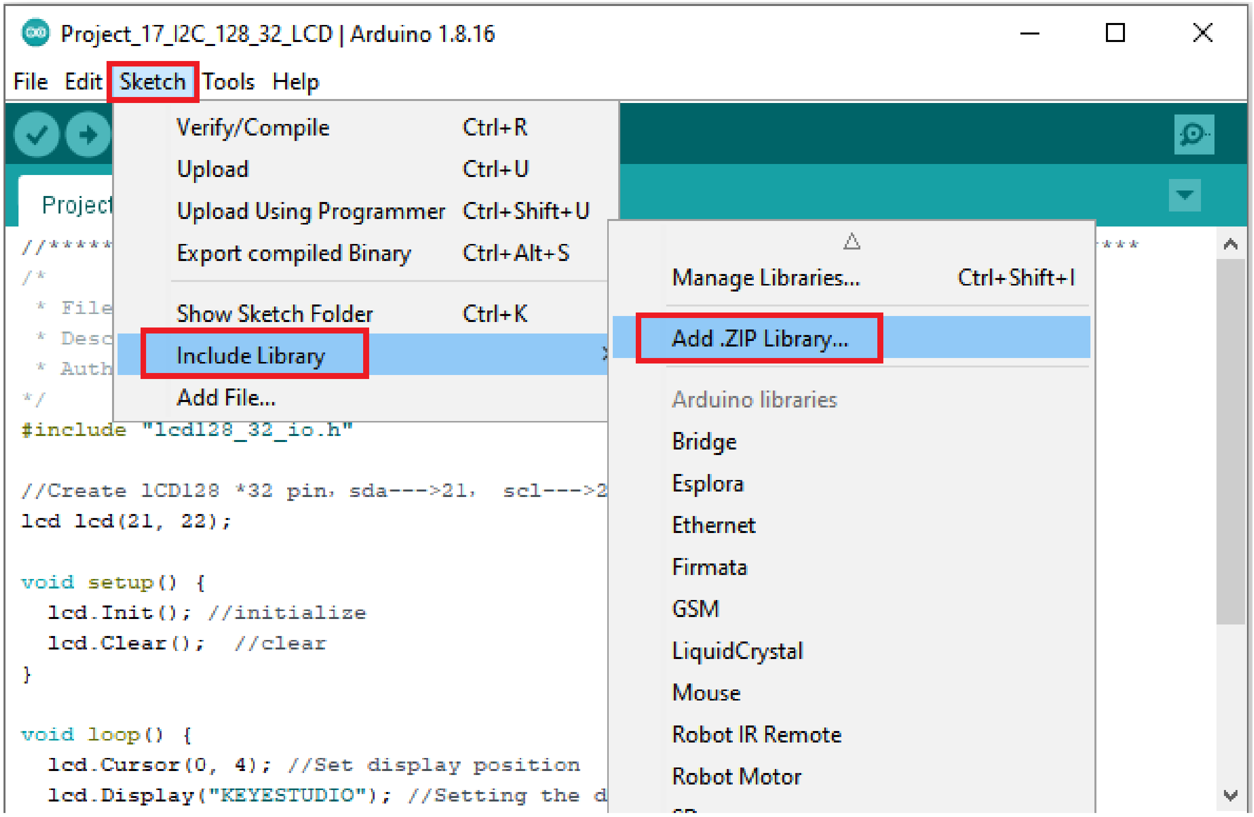
media/072d954dac310add077688398ad59af2.png

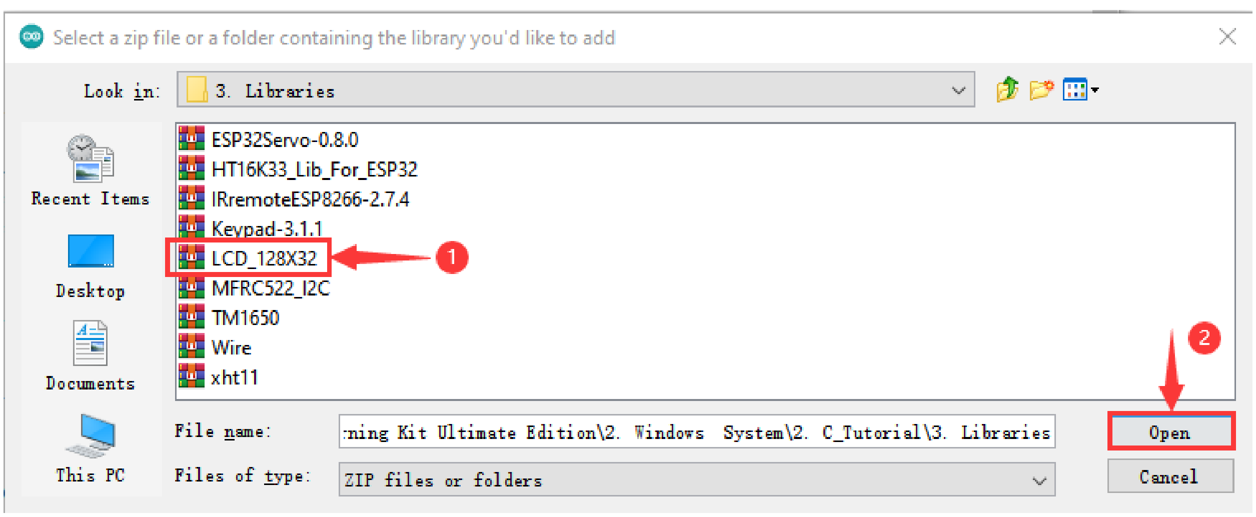
5. Adding the lcd128_32_io library

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"→"Include Library"→"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".





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"**.

```
//**********************************************************************************
/*
 * Filename    : LCD 128*32
 * Description : LCD 128*32 display string
```

(continues on next page)

(continued from previous page)

```
 * Auther      : http//www.keyestudio.com
*/
##include "lcd128_32_io.h"

//Create lCD128 *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

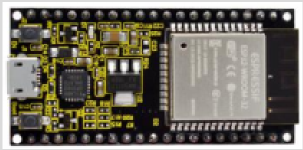
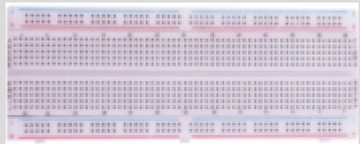
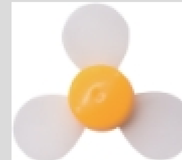
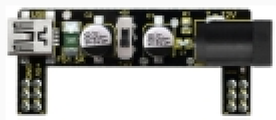
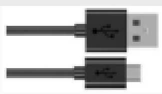
Compile and upload the code to ESP32, after the code is uploaded successfully, power up with a USB cable and you will see that the 128X32LCD module display will show"KEYESTUDIO"at the first line"ABCDEFGHIJKLMNOPQR"will be displayed at the second line"123456789±*/<>=$@"will be shown at the third line 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

			
ESP32*1	Breadboard*1	Battery Holder*1	Fan*1
			
130 Motor Module*1	Keyestudio bread board special power module*1	M-F Dupont Wires	No.5 battery (self-provided)*6
			
USB Cable*1			

3. Component knowledge :

130 motor module: The motor control module uses the HR1124S motor control chip. which is a single-channel H-bridge 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:

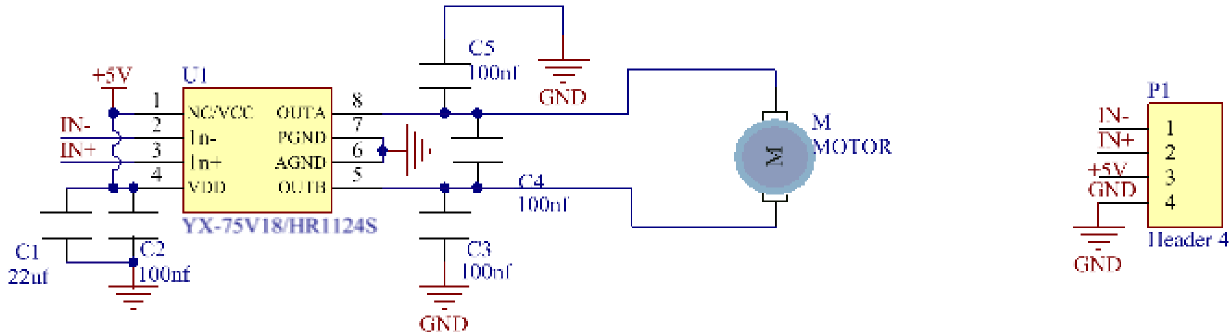
Working voltage: 5V

Working current: 200MA

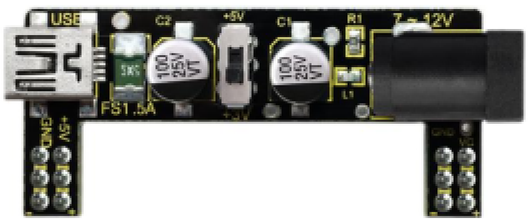
Working power: 2W

Working temperature: -10℃~ +50℃

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.

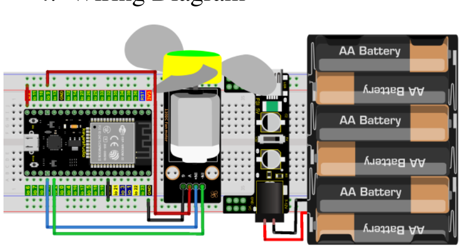
If the other power supply is DC 6.5-12v, when the slide switch is switched to +5V, the output voltages of the left and right lines of the module are DC 5V. When the slide switch is switched to +3V, the output voltage of the USB power supply terminal of the module is DC5V , and the output voltage of the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

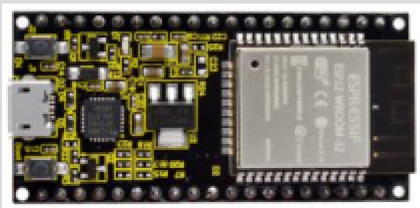
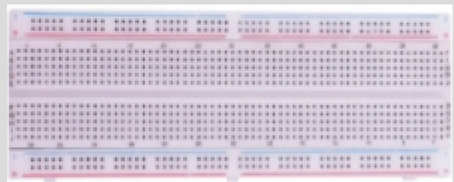
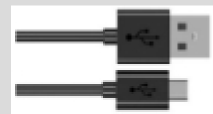
Compile and upload the code to ESP32, after the code is uploaded successfully, power up with a USB cable and you will see thatthe small fan turns counterclockwise for 5 seconds and stops for 2 seconds, and then turns clockwise for 5 seconds and stops for 2 seconds, which repeats in an endless loop.

5.20 Project 19Servo Sweep

1. Introduction

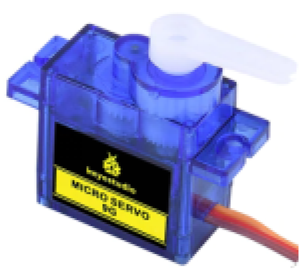
Servo is an electric motor that can rotate very precisely. At present, it has been widely used in toy cars, remote control helicoptersairplanesrobots, etc. In this project, we will use ESP32 to control the rotation of the servo.

2. Components

		
ESP32*1	Breadboard*1	USB Cable*1
		
Servo*1	Jumper Wires	

3. Component knowledge

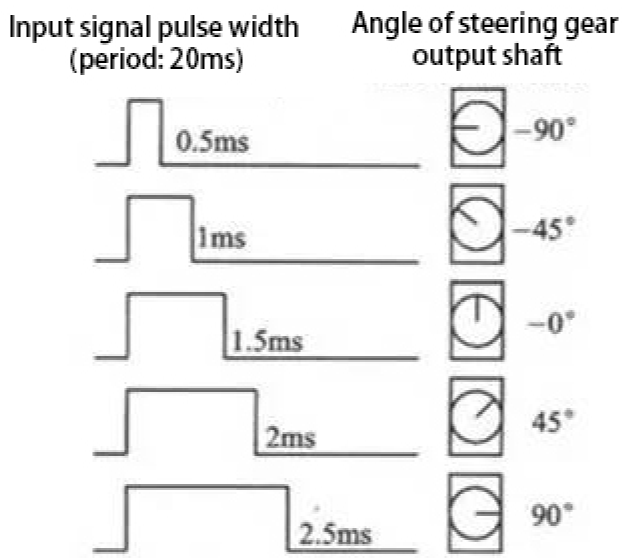
Servo



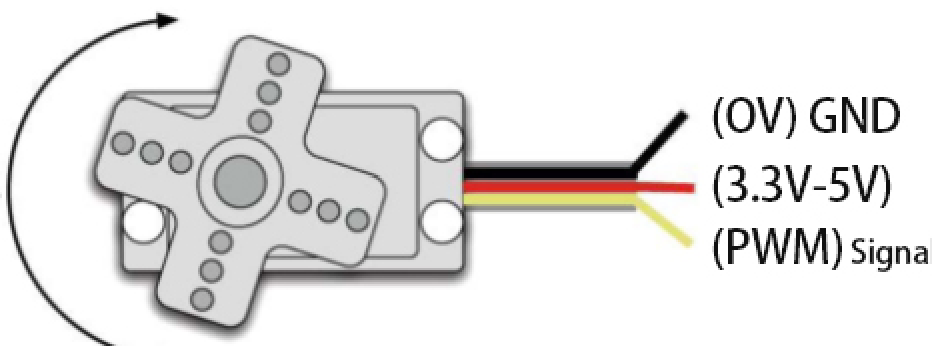
The servo is a kind of position servo driver, which is mainly composed of housingcircuit boardcopless motorgear and position detector. Its working principle is that the receiver or microcontroller sends a signal to the servo which has an internal reference circuit that generates a reference signal with 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.

media/39621cc861e5f7c189a047b7f0bbd0be.png

5. Adding the ESP32Servo library

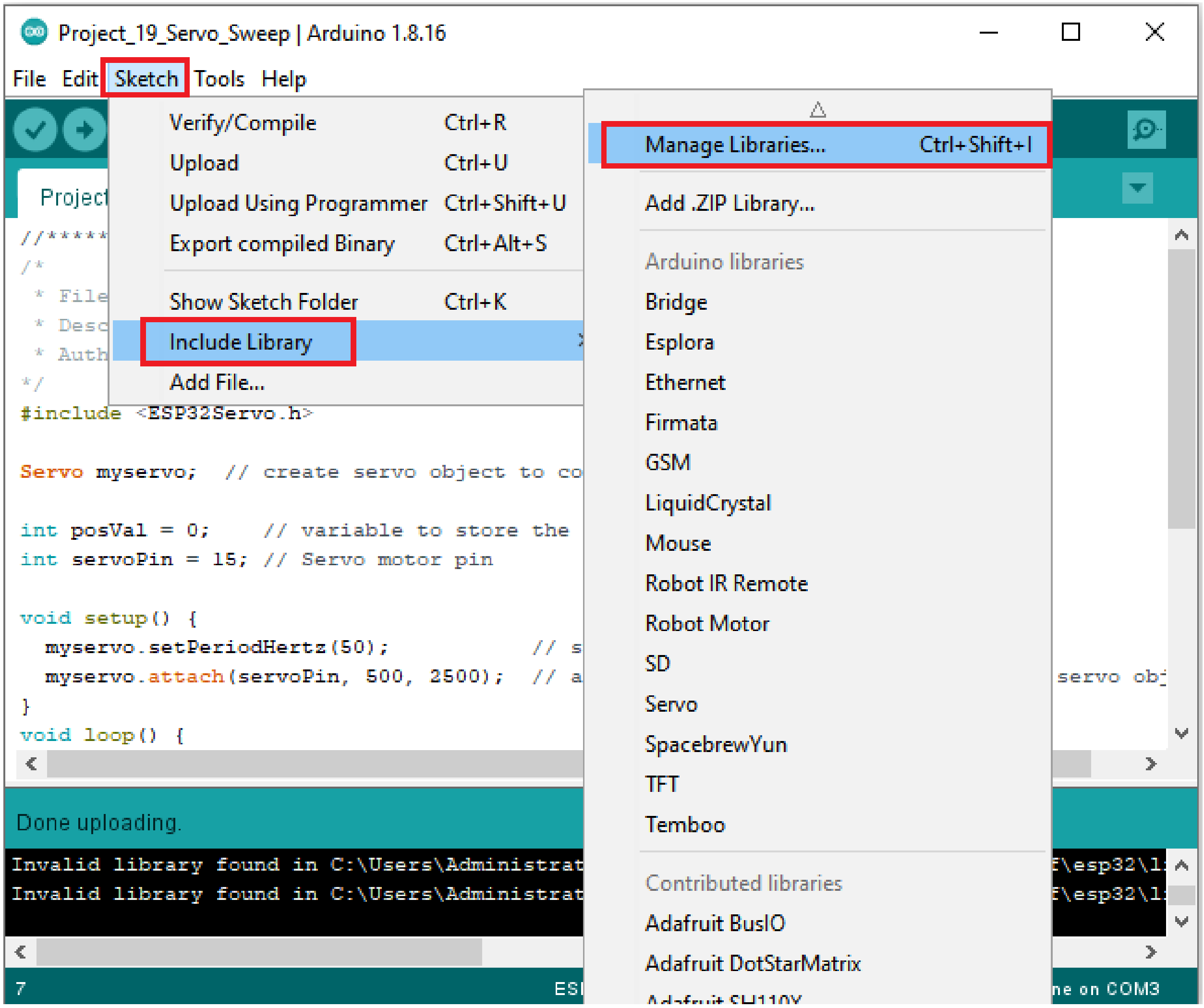
If you have not downloaded the library file, please click on the link to download it:Download Arduino Libraries

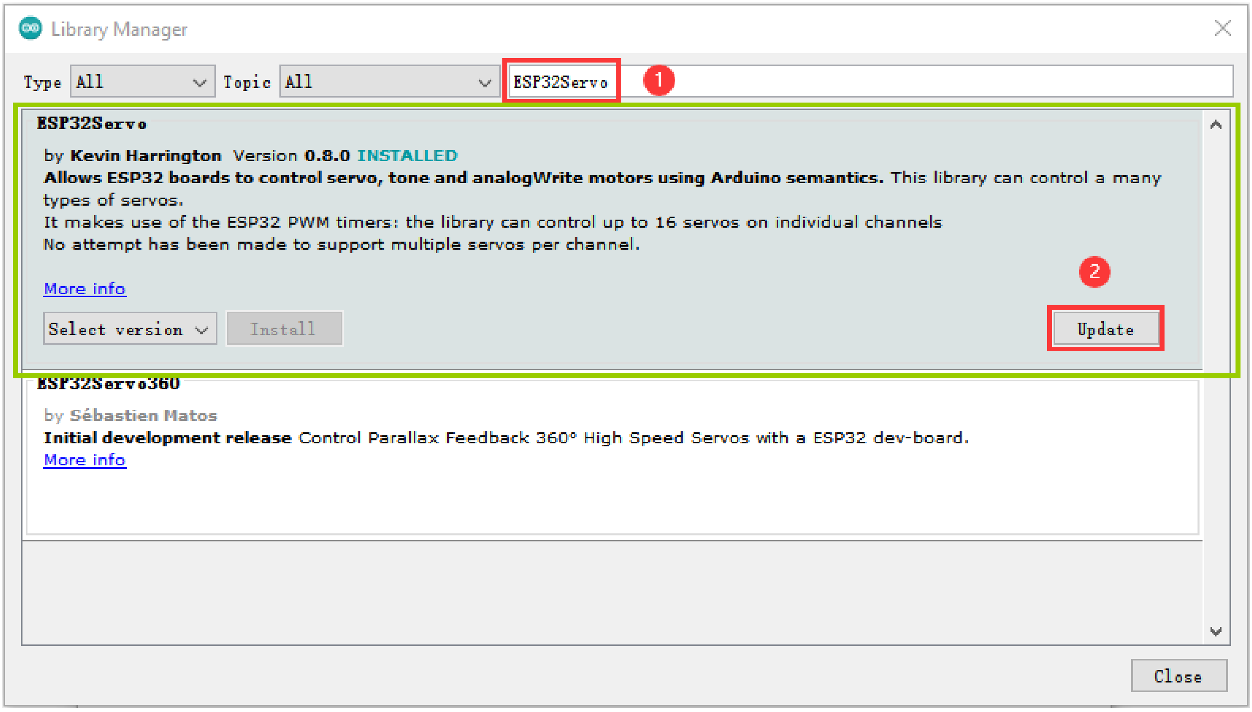
This code uses a library named "**ESP32Servo**", If you haven't installed it yet, please do so before learning. The steps to add third-party libraries are as follows:

How to install the library

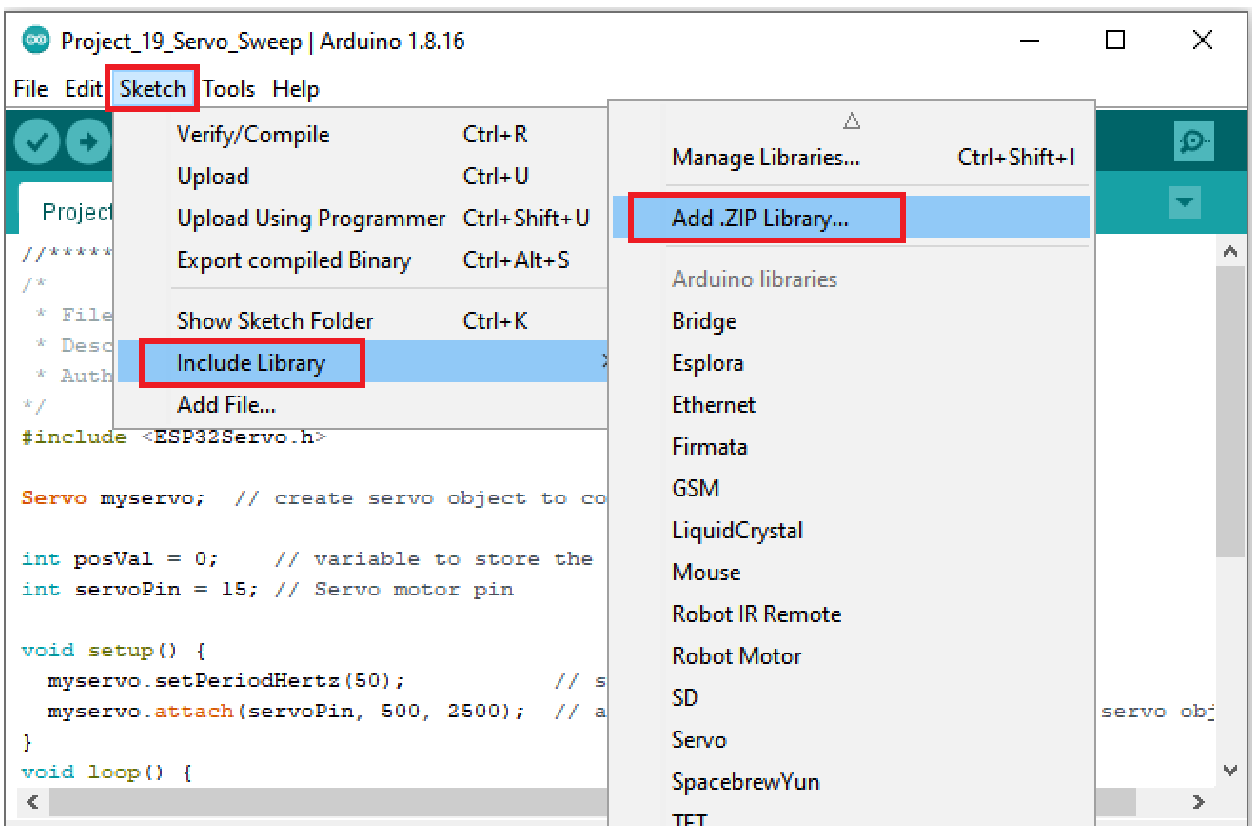
There are two ways to add libraries:

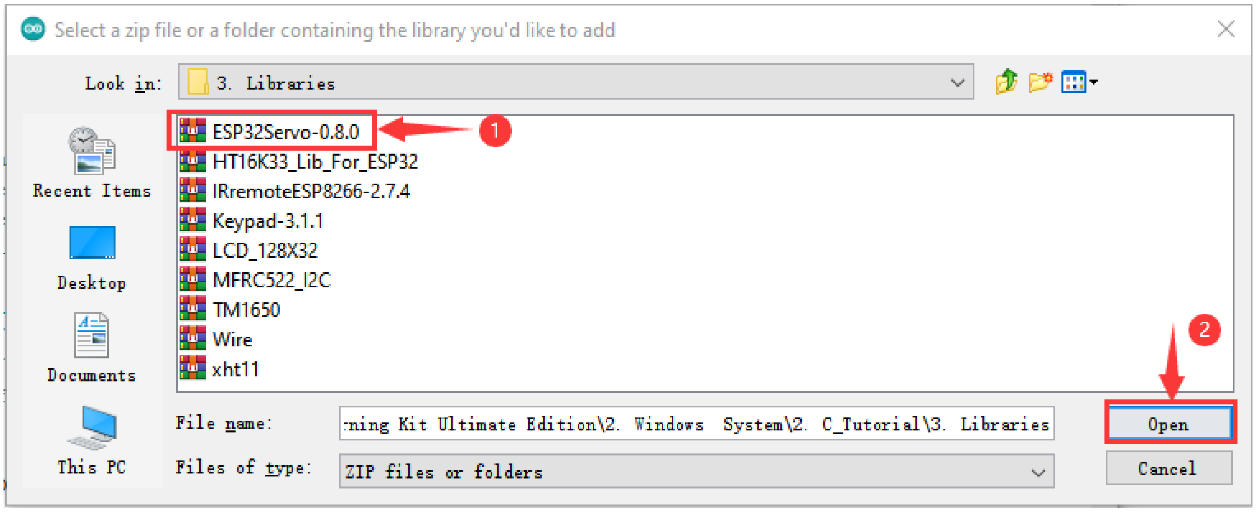
The first way, open the Arduino IDE, click"Sketch"→"Include Library"→"Manage Libraries". Enter "ESP32Servo" in the search box, select "ESP32Servo" and click "Update" to install. Please refer to the following operations :





The second wayopen the Arduino IDEclick “Sketch”→“Include Library”→“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”.





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

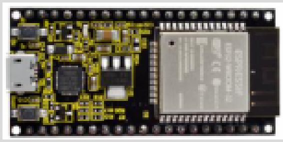
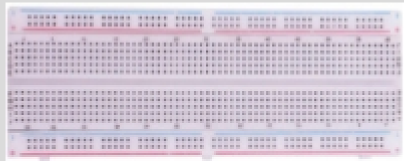
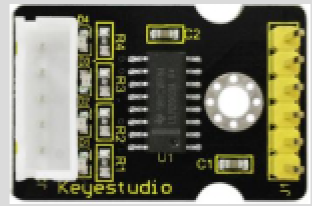
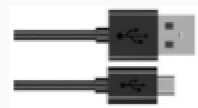
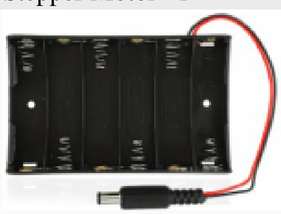
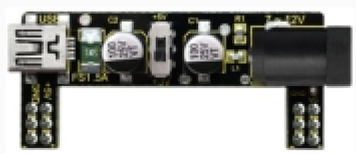
Compile and upload the code to ESP32, after the code is uploaded successfully, power up with a USB cable and you will see that the Servo will rotate from 0 degrees to 180 degrees and then reverse the direction to make it rotate from 180 degrees to 0 degrees and repeat these actions in an endless loop.

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

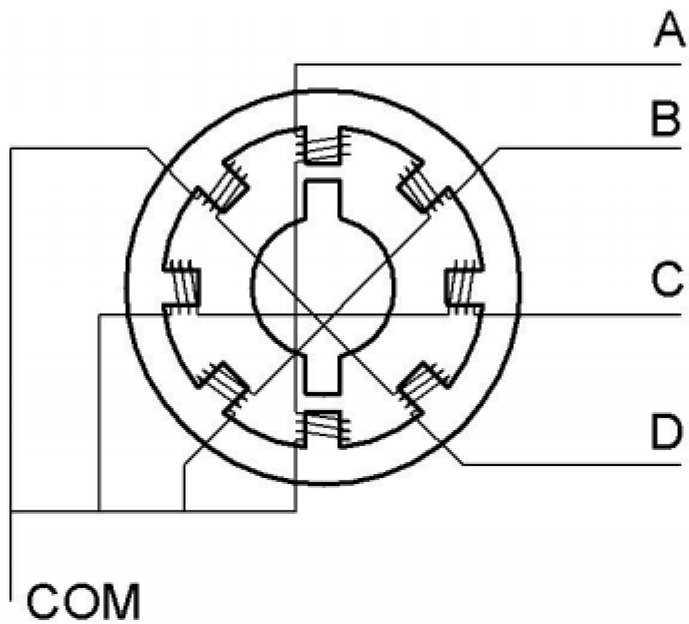
		
ESP32*1	Breadboard*1	ULN2003 Stepper Motor Drive Board*1
		
Stepper Motor *1	M-F Dupont Wires	USB Cable*1
		
Battery Holder*1	Keyestudio bread board special power module*1	No.5 battery (self-provided)*6

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.

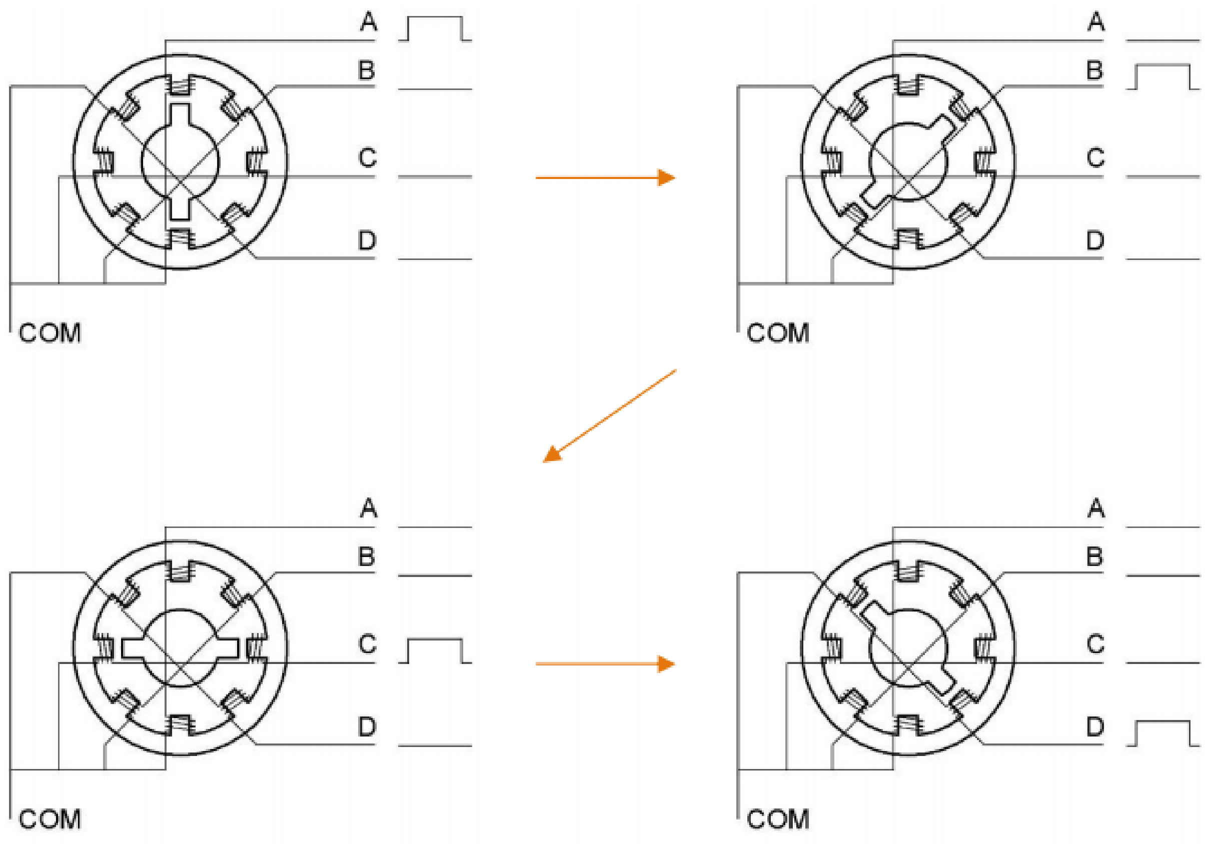
Working Principle:

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 is turned on, and the rotor is turned to 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.

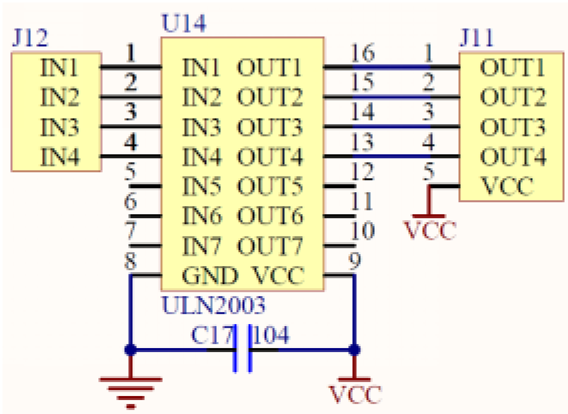
Stepper Motor Parameters:

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.

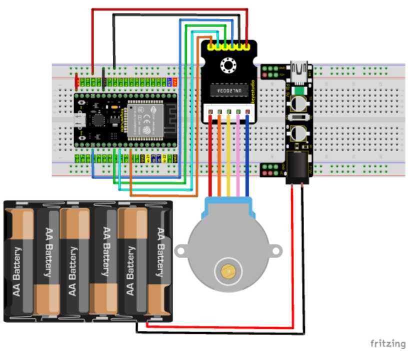
The step angle of 4-beat mode of 5V and 4-phase stepper motor is 11.25. And the step angle of 8-beat mode is 5.625, the reduction ratio is 1:64.

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.



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);
```

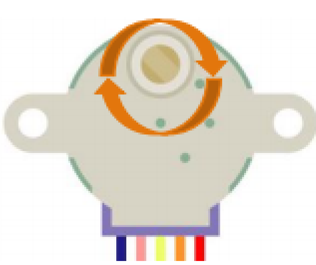
(continues on next page)

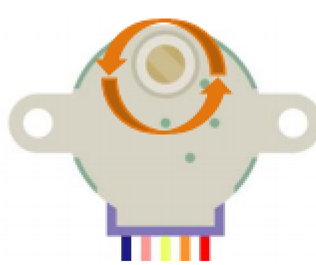
(continued from previous page)

```
  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++) {
    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 = 0x01;
  // Decide the shift direction according to the rotation direction
  if (dir) {  // ring shift left
    out != 0x08 ? out = out << 1 : out = 0x01;
  }
  else {      // ring shift right
    out != 0x01 ? out = out >> 1 : out = 0x08;
  }
  // Output singal to each port
  for (int i = 0; i < 4; i++) {
    digitalWrite(outPorts[i], (out & (0x01 << i)) ? HIGH : LOW);
  }
}

void moveAround(bool dir, int turns, byte ms){
  for(int i=0;i<turns;i++)
    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.



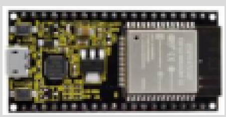
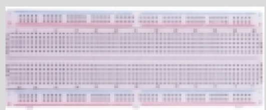
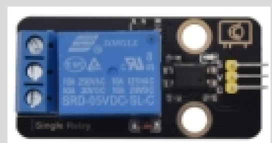
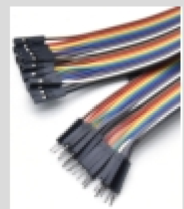
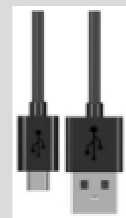


5.22 Project 21Relay

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

				
ESP32*1	Breadboard*1	Relay Module*1	M-F Dupont Wires	USB Cable*1

3. Component knowledge

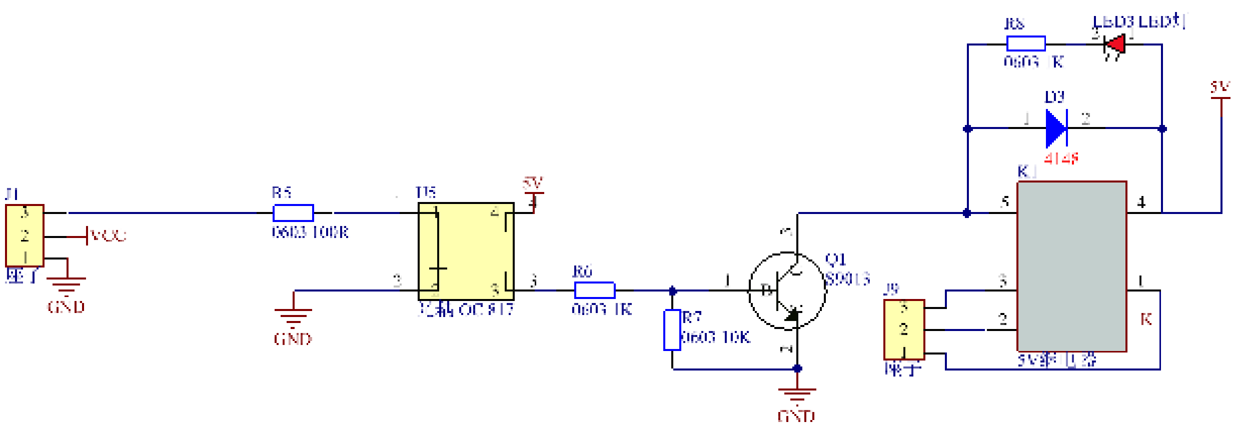
Relay: It is an "automatic switch" that uses a small current to control the operation of a large current.

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



4. Wiring Diagram

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.

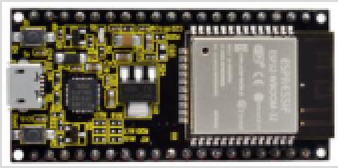
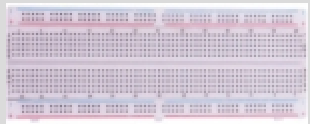
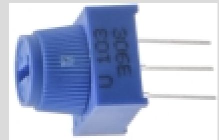
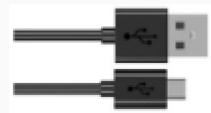
5.23 Project 22Dimming Light

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.

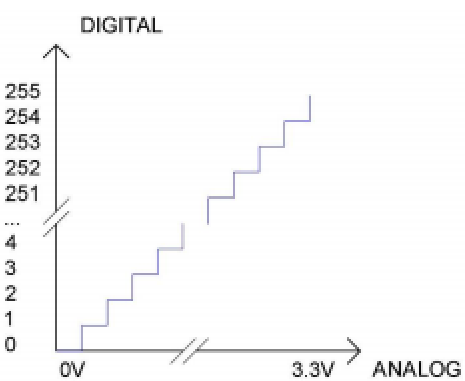
2. Components

			
ESP32*1	Breadboard*1	Potentiometer*1	Red LED*1
			
220Resistor*1	Jumper Wires	USB Cable*1	

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 ~ 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"**.

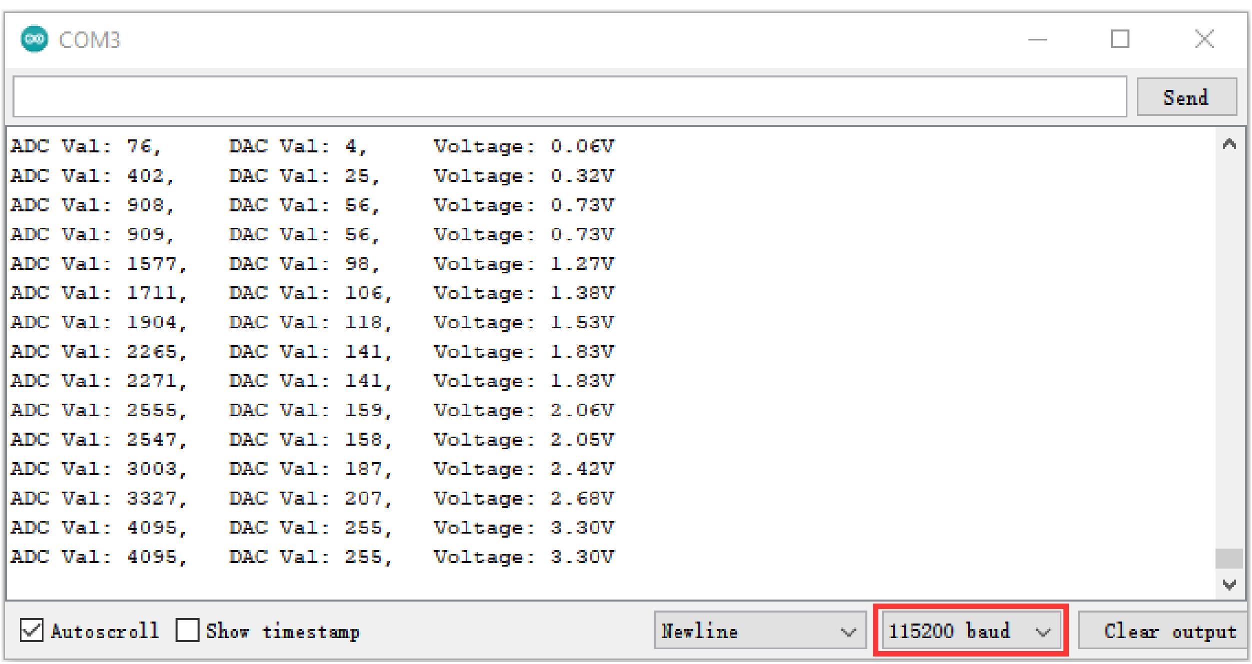
```
//**********************************************************************************
/*
```

(continues on next page)

(continued from previous 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 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 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:



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:

media/3396bd77169711de6e15da73f14c8afb.png

6. Project code

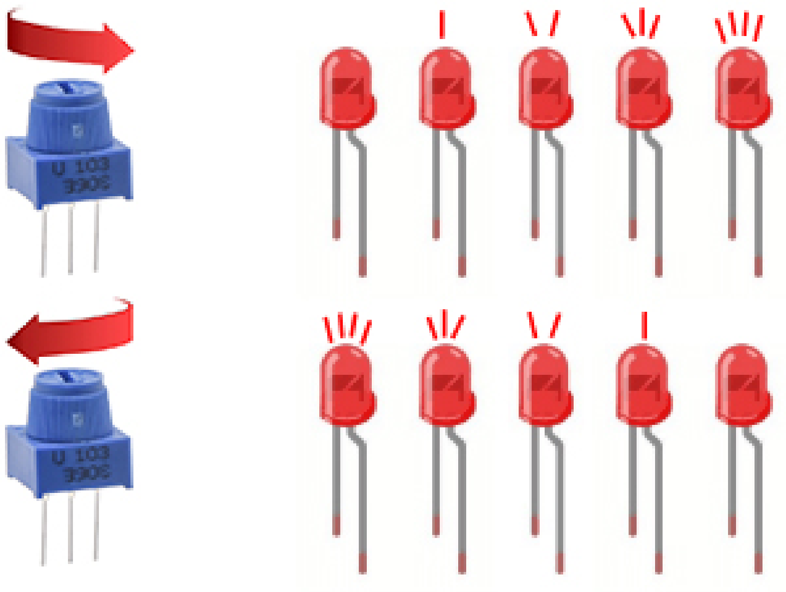
You can open the code we provideIf you haven't downloaded the code file, please click on the link to download it:Download Arduino Codes

The code used in this project is saved in folder **"Arduino-Codes\Project 22Dimming Light\Project_22.2_Dimming_Light"**.

```
//**********************************************************************************
/*
 * Filename    : Dimming Light
 * Description : Controlling the brightness of LED by potentiometer.
 * Auther      : http//www.keyestudio.com
*/
##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.

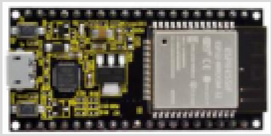
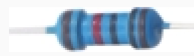
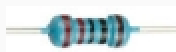
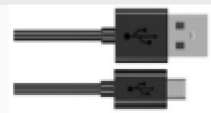


5.24 Project 23Flame Alarm

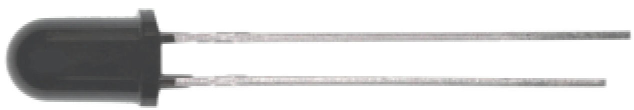
1. Introduction

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.

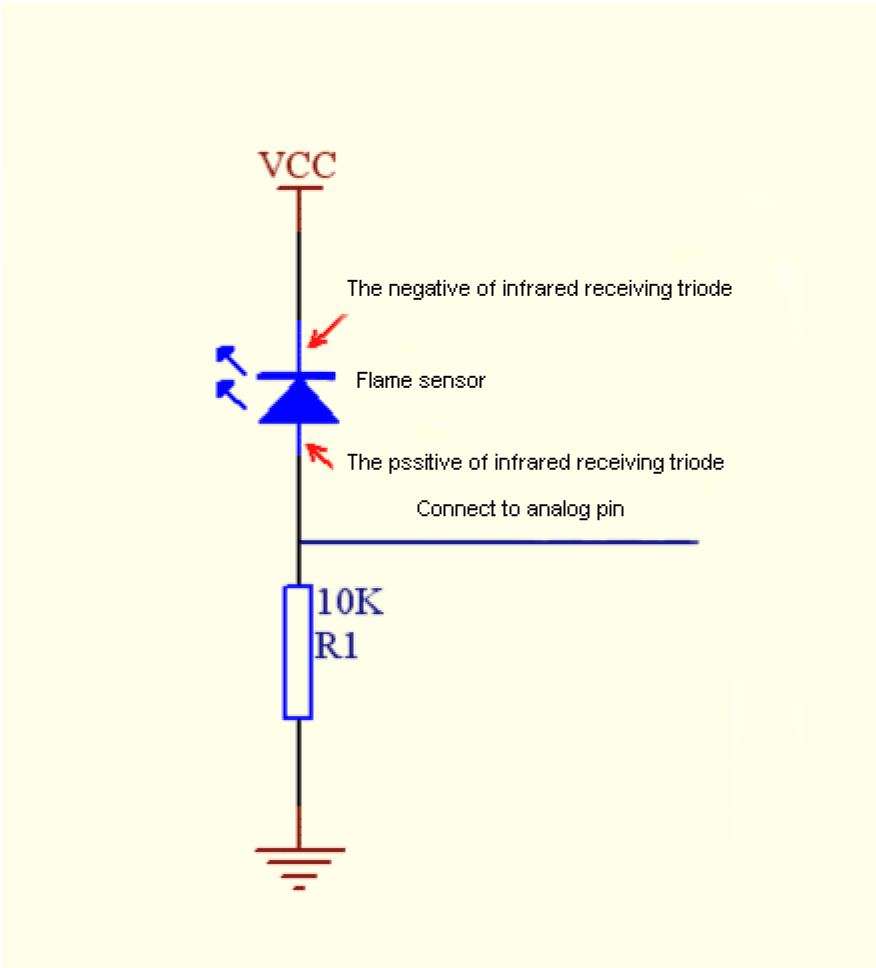
2. Components

			
ESP32*1	Breadboard*1	Red LED*1	Active Buzzer*1
			
Flame Sensor*1	220Resistor*1	10KResistor*1	Jumper Wires
			
NPN transistor(S8050)*1	1k Resistor*1	USB Cable*1	

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.

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

media/76ce57355da1df27e049bdc6e19f0650.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 23Flame Alarm\Project_23.1_Read_Analog_Value_Of_Flame_Sensor** ".

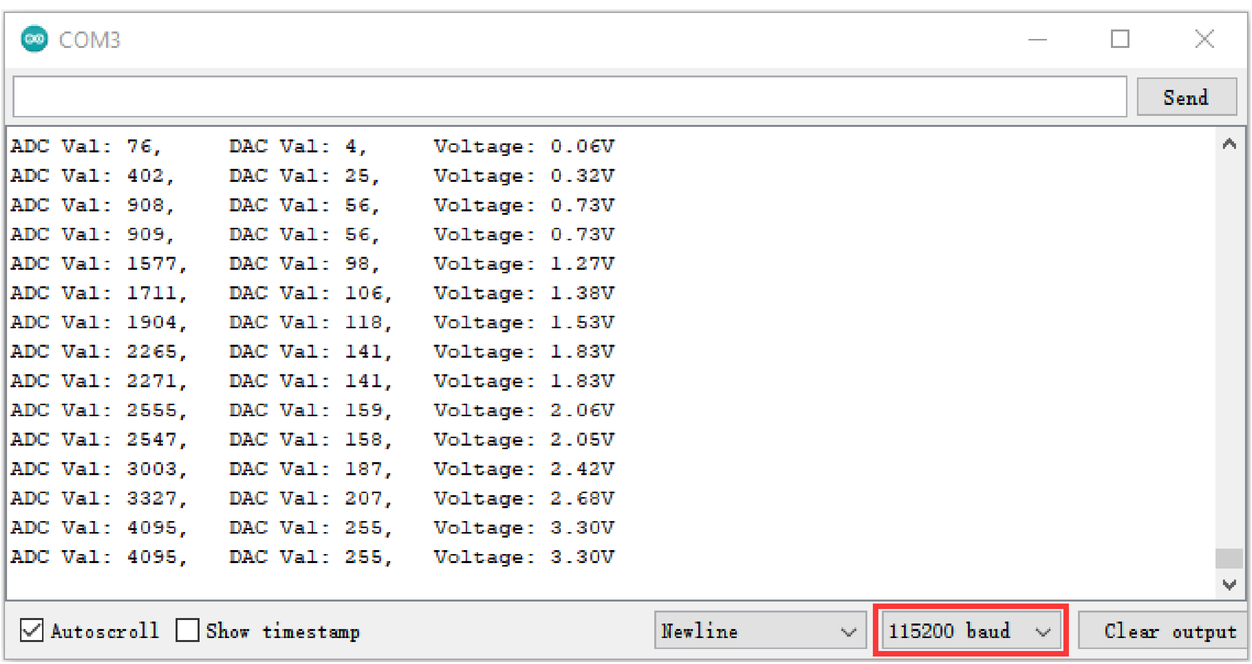
```
//**********************************************************************************
/*
```

(continues on next page)

(continued from previous 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 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);
}
//*********************************************************************************
```

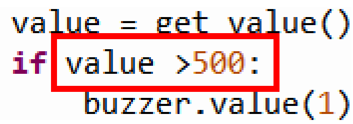
Compile and upload the code to ESP32, after the code is uploaded successfully, power up with a USB cable, open the serial monitor and set the baud rate to 115200. You will see that the serial port monitor window will print out the ADC value, DAC value and voltage value of the flame sensor. When the sensor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



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

6. Project codeNote 

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.
 * Auther      : http//www.keyestudio.com
*/
##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

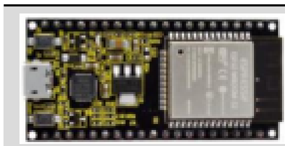
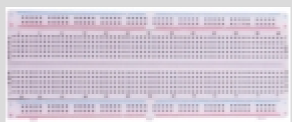
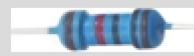
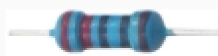
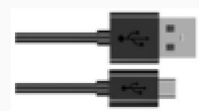
Compile and upload the code to ESP32, after the code is uploaded successfully, power up with a USB cable and you will see that when the flame sensor detects the flame, the LED will flash and the buzzer will alarm; otherwise, the LED does not light up and the buzzer does not sound.

5.25 Project 24Night Lamp

1.Introduction

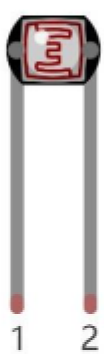
Sensors or components are ubiquitous in our daily life. For example, some public street lamps will automatically turn on at night and turn off during the day. Why? In fact, this make use of a photosensitive element that senses the intensity of external ambient light. When the outdoor brightness decreases 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.

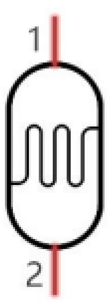
2. Components

			
ESP32*1	Breadboard*1	Red LED*1	10KResistor*1
			
Photoresistor*1	220Resistor*1	Jumper Wires	USB Cable*1

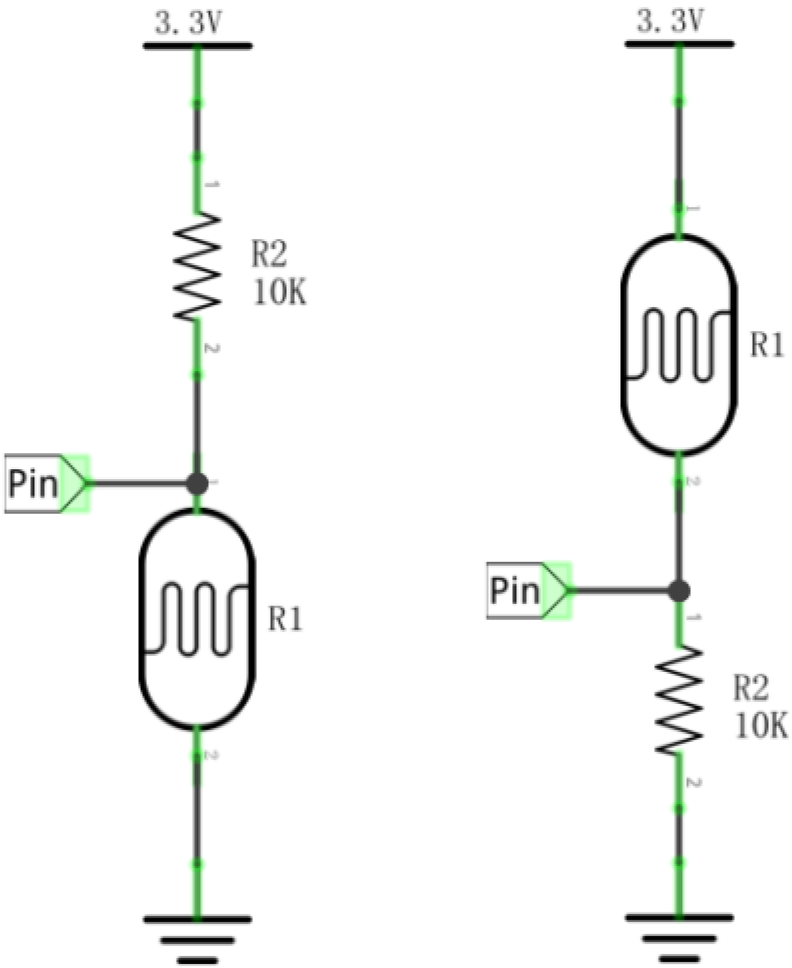
3. Component knowledge

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"**.

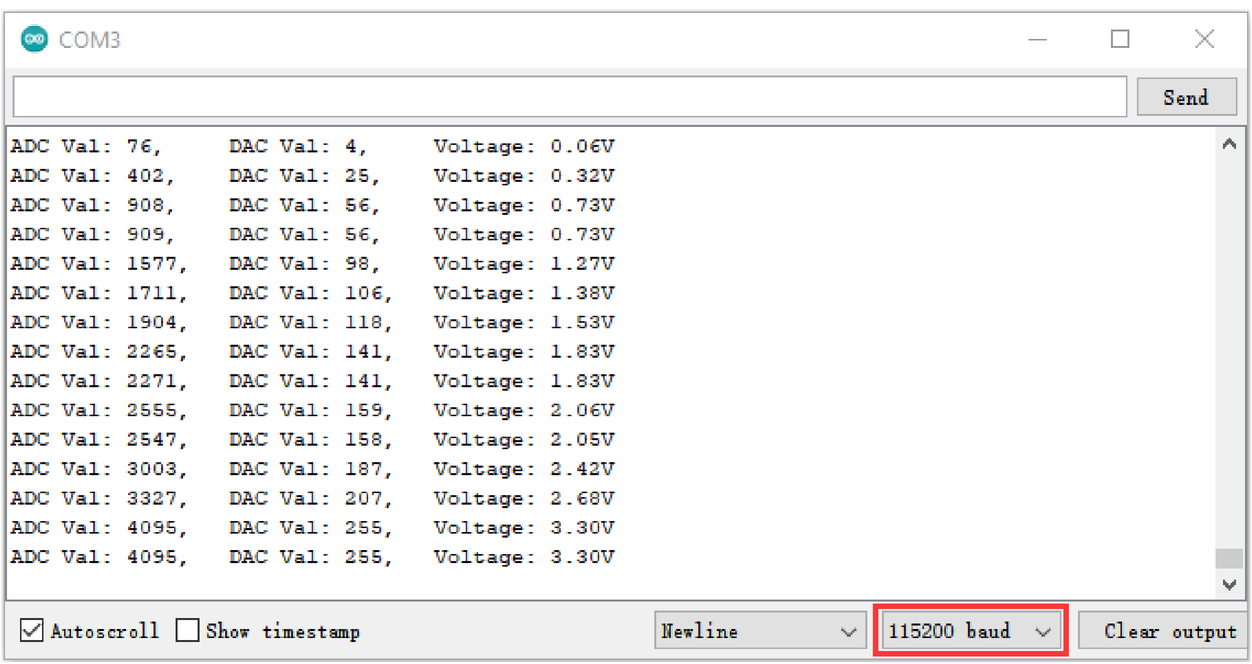
```
//**********************************************************************************
/*
 * Filename    : Read Photosensitive Analog Value
```

(continues on next page)

(continued from previous page)

```
 * 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 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.



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 thatwhen the intensity of light around the photoresistor is reduced, the LED will be bright, on the contraty, the LED will be dim.

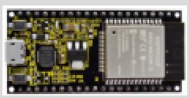
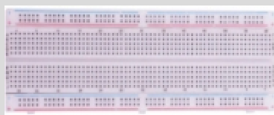
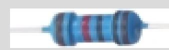
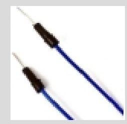
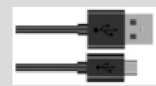
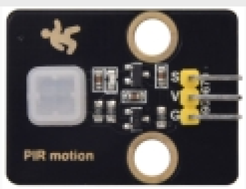
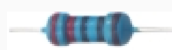
5.26 Project 25Human Induction Lamp

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.

2. Components

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

3. Wiring Diagram

media/69f49d65054a9246acf4adc534217027.png

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".**

```
//**********************************************************************************
/*
 * 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
```

(continues on next page)

(continued from previous page)

```
##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
      }
}
//*************************************************************************************
```

5. Project result

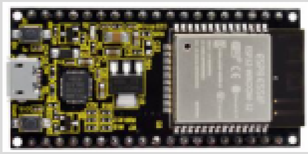
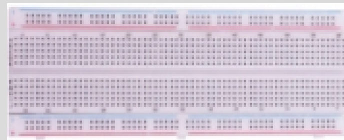
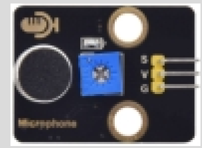
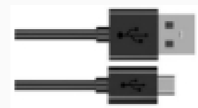
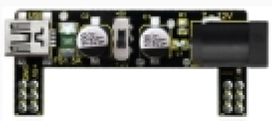
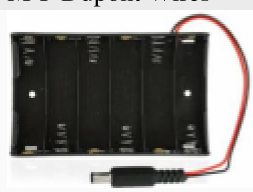
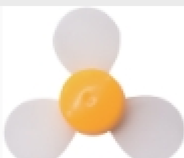
Compile and upload the code to ESP32, after the code is uploaded successfully, power up with a USB cable and you will see thatWhen your hand covers the photosensitive part of the photoresistor to simulate darkness, then shake your other hand in front of the Human Body Infrared Sensor, the external LED 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

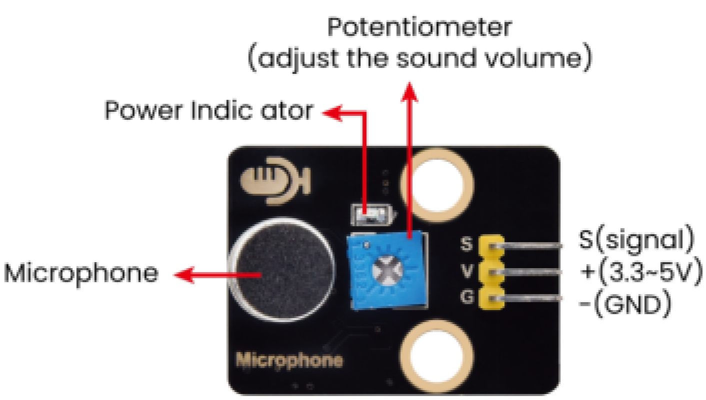
1. Introduction

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

		
ESP32*1	Breadboard*1	Sound Sensor*1
		
130 Motor Module*1	M-F Dupont Wires	USB Cable*1
		
Keyestudio bread board special power module*1	Battery Holder*1	No.5 battery (self-provided)*6
		
Fan*1		

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

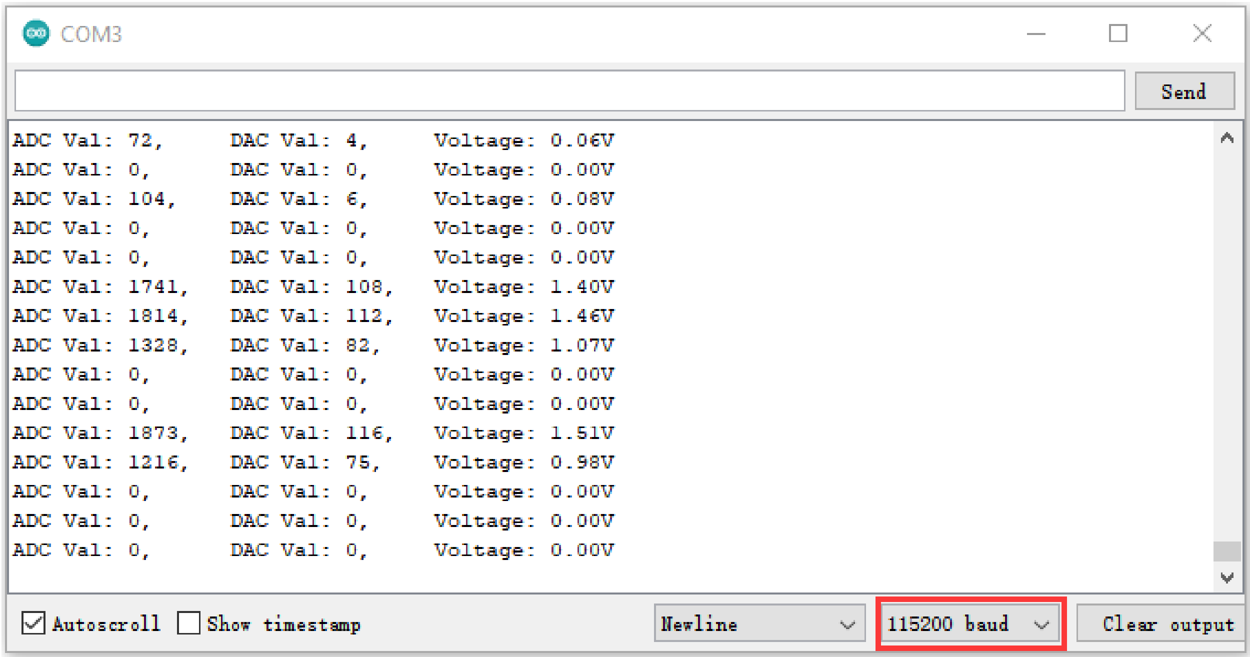
media/87fb44c475d1f53aa5905cebfed55ea2.png

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"**.

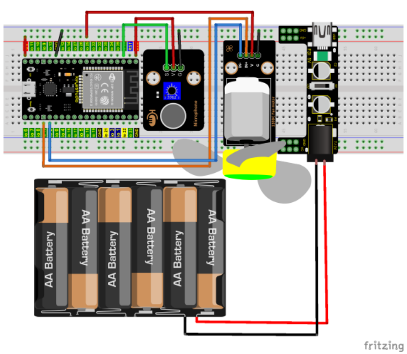
```
//**********************************************************************************
/*
 * Filename    : Read Sound Sensor Analog Value
 * Description : Basic usage of ADC
 * Auther      : http//www.keyestudio.com
*/
##define PIN_ANALOG_IN  36  //the pin of the Sound Sensor

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 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.



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 voice-controlled 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"**.

```
//**********************************************************************************
/*
 * 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
```

(continues on next page)

(continued from previous page)

```
##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

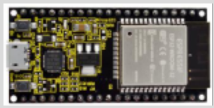
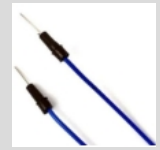
Compile and upload the code to ESP32, after the code is uploaded successfully, external power supply and power on. and then you will see that clap your hands to the sound sensor, and when the sound intensity exceeds a threshold, the small fan rotates; conversely, the small fan doesn't rotate.

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

					
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		

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 ℃, it output 0V; If increasing 1 ℃, the output voltage will increase 10mv. The output temperature is 0°C to 100°C, the conversion formula is as follows

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

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

media/041471b9fabd75ef9dc3951598e342f8.png

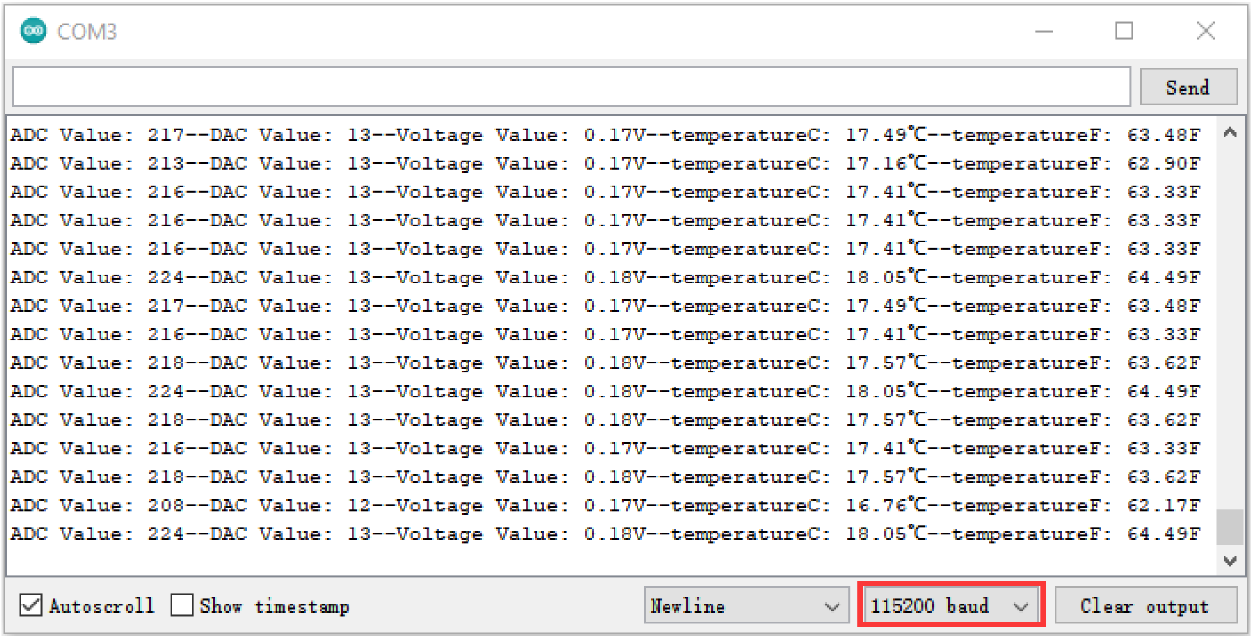
LM35 output is given to analog pin GPIO36 of the ESP32, this analog voltage is converted to its digital form and processed to get the temperature reading.

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"**.

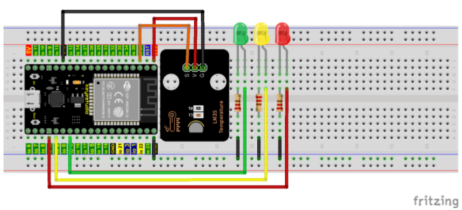
```
//**********************************************************************************
/*
 * Filename    : Read LM35 Temperature Value
 * Description : ADC value is converted to LM35 temperature value
 * Auther      : http//www.keyestudio.com
*/
##define PIN_ANALOG_IN  36  //the pin of the Temperature 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 measured voltage value,Celsius and Fahrenheit valuesthrough 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) + "℃");
  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.



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
##define PIN_YELLOWLED   2      //the pin of the Yellow led
##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)

(continued from previous 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) + "℃");
  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) {
    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

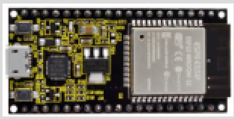
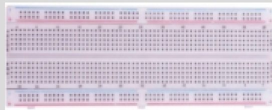
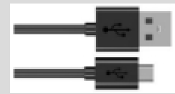
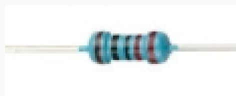
Compile and upload the code to ESP32, after the code is uploaded successfully, power up with a USB cable and you will see that the monitor displays the temperature values read by the LM35 temperature sensor. When the LM35 temperature sensor senses different temperatures, different LEDs will light up.

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

				
ESP32*1	Breadboard*1	Rocker Module*1	USB Cable*1	M-F Dupont Wires
				
RGB LED*1	220Resistor*3	Jumper Wires		

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

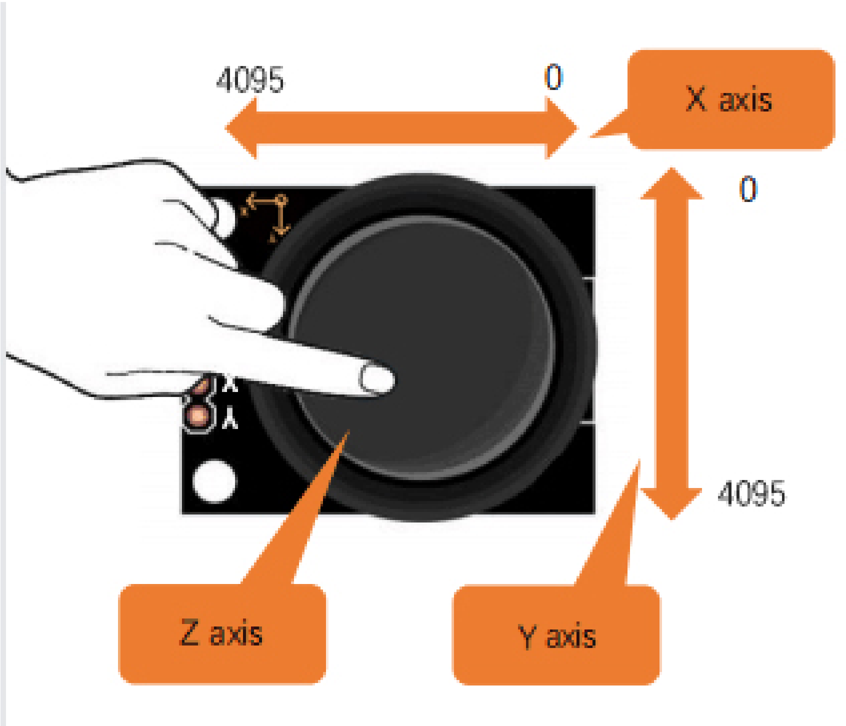
media/b611755eacc4c603e6c0555aced929cb.png

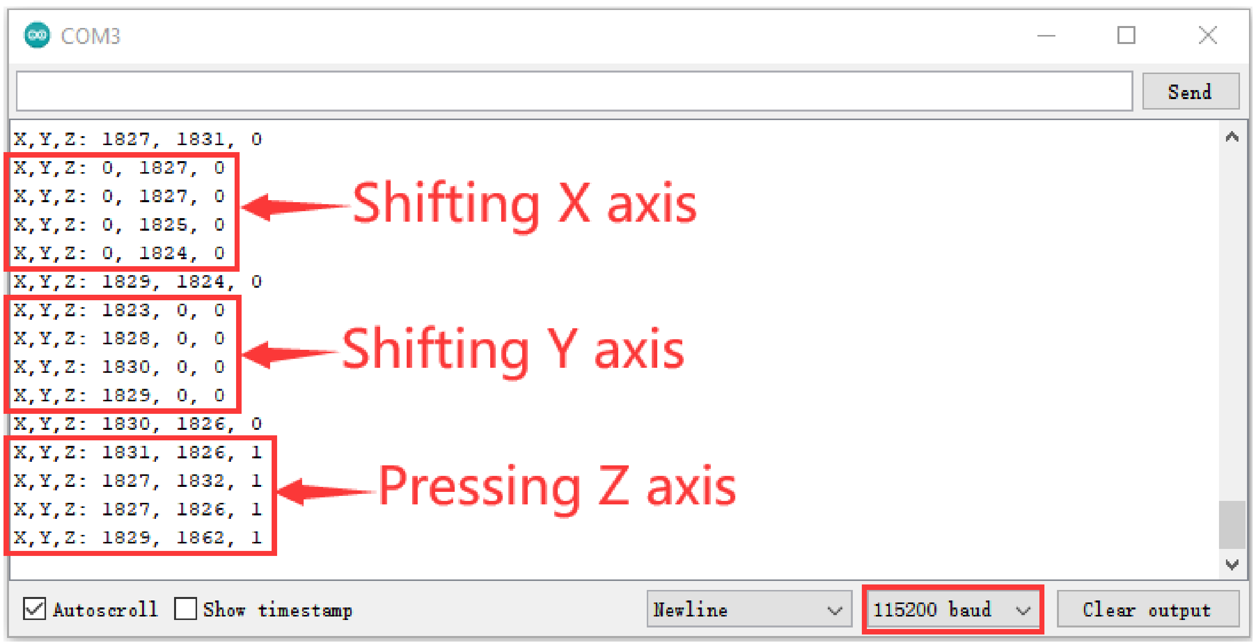
You can open the code we provideIf you haven't downloaded the code file, please click on the link to download it:Download Arduino Codes

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.
 * Auther      : http//www.keyestudio.com
*/
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)

(continued from previous 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.

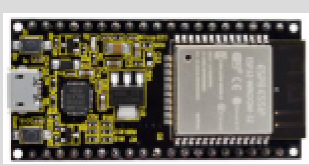
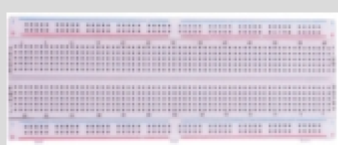
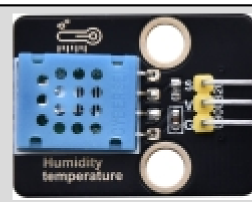
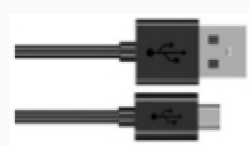
media/9c2d0d8777200827b16c49b752d45c4c.jpeg

5.30 Project 29Temperature Humidity Meter

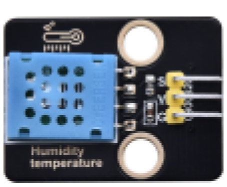
1. Introduction

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

		
ESP32*1	Breadboard*1	Temperature and Humidity Sensor*1
		
LCD 128X32 DOT*1	M-F Dupont Wires	USB Cable*1

3. Component knowledge



Temperature and humidity sensor: It is a temperature and humidity composite sensor with calibrated digital signal output, its precision humidity is±5%RH, temperature is±2℃, range humidity is 20 to 90%RH, and temperature is 0 to 50℃. 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␣
↪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␣
↪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 in the

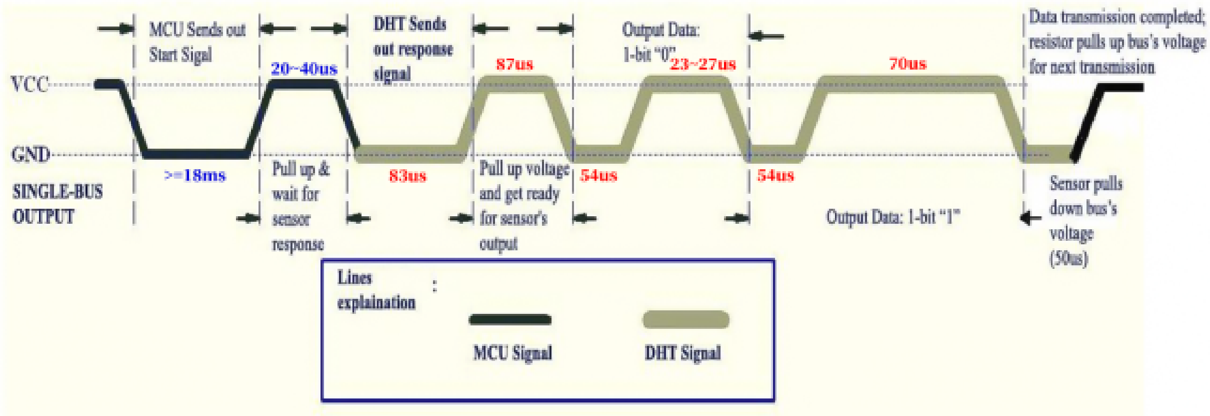


figure:

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:

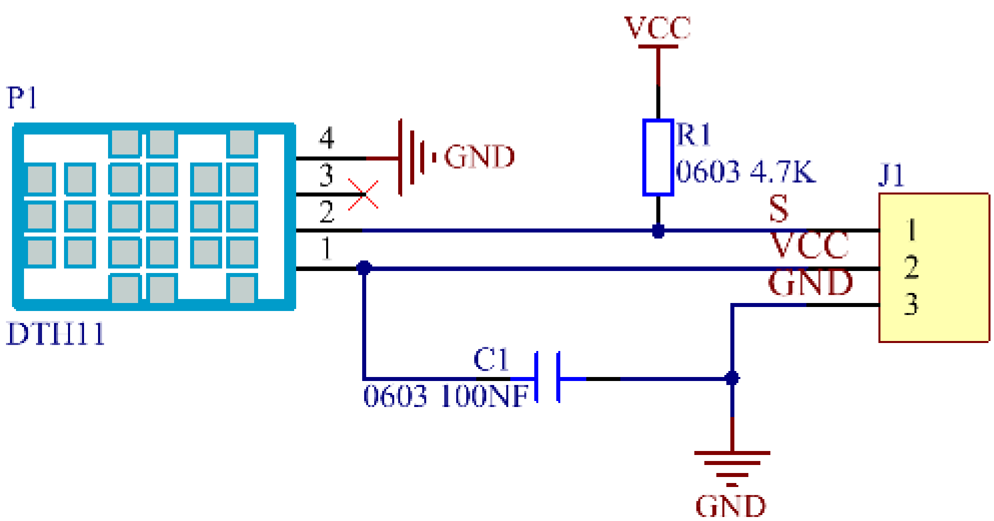
Working voltage: +5V

Temperature range: 0°C to 50°C, error of ± 2°C

Humidity range: 20% to 90% RH,± 5% RH error

Digital interface

Schematic diagram of Temperature and Humidity Sensor:



4. Read temperature and humidity value

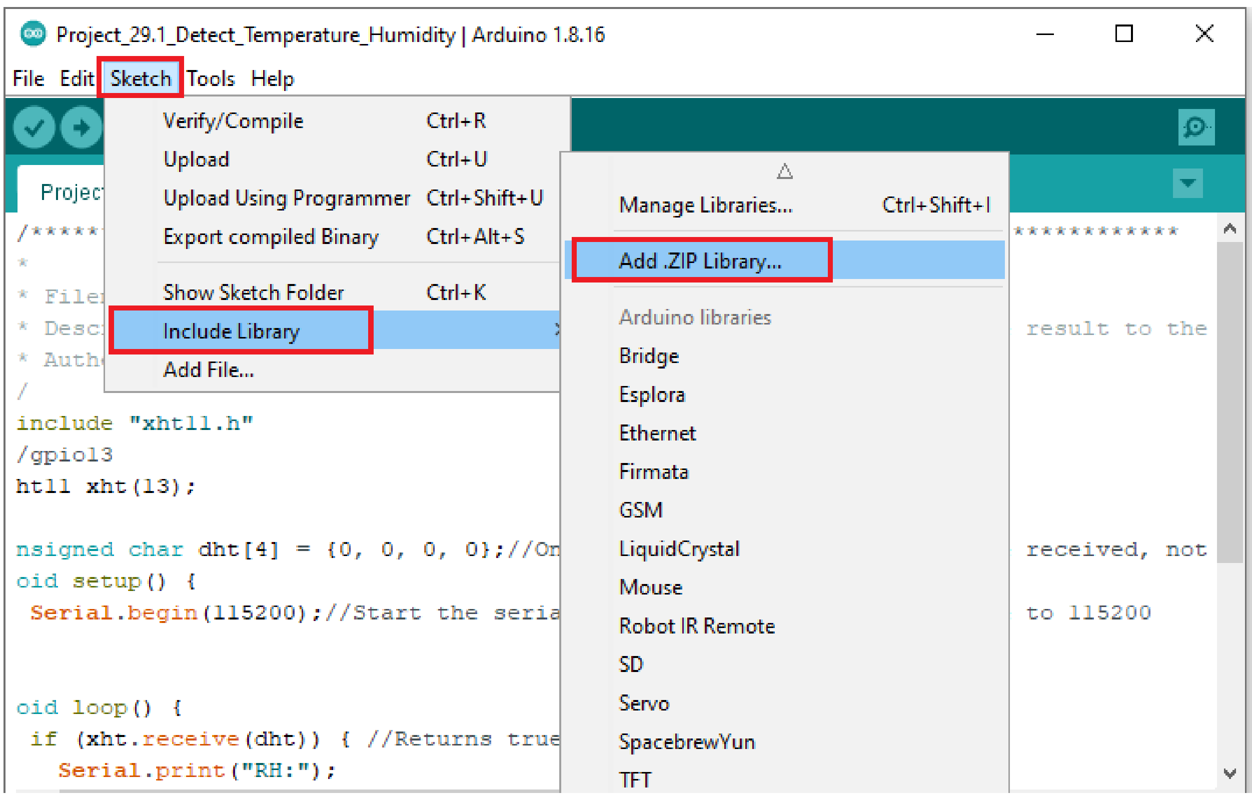
media/5d6dd3f19b4323d212bb95e3e4d43743.png

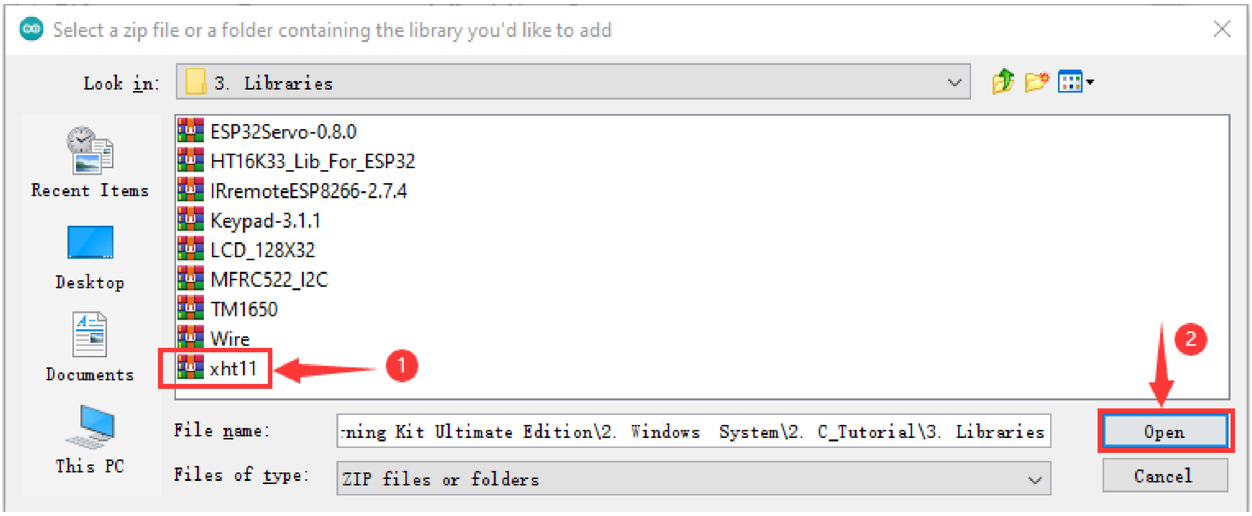
How to add the xht11 library

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"→"Include Library"→"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".





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 Meter\Project_29.1_Detect_Temperature_Humidity"**.

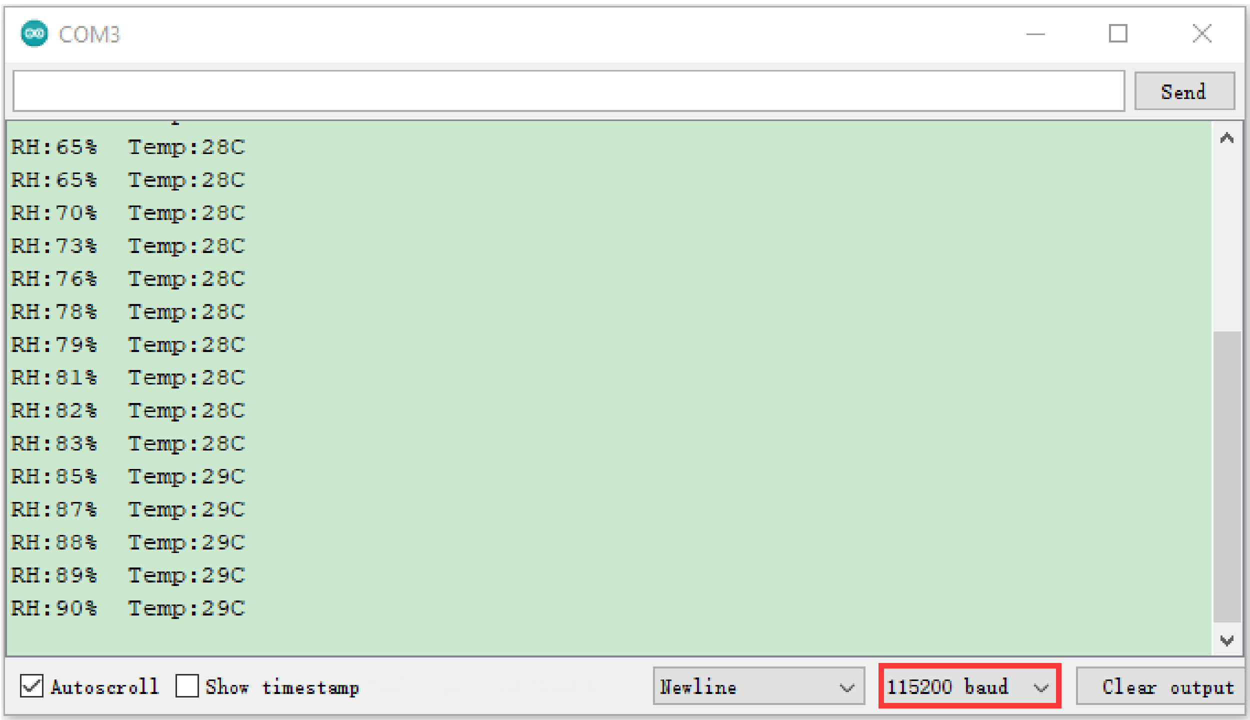
```
//**********************************************************************************
/*
 * Filename    : Temperature Humidity Meter
 * Description : LCD displays the value of temperature and humidity.
 * Auther      : http//www.keyestudio.com
*/
##include "xht11.h"
##include "lcd128_32_io.h"

//gpio13
xht11 xht(13);
unsigned char dht[4] = {0, 0, 0, 0};//Only the first 32 bits of data are received, not␣
↪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)

(continued from previous page)

```
  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);
}
//*************************************************************************************
```

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:



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

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"**.

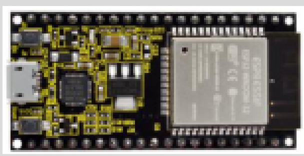
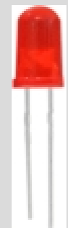
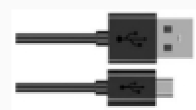
```
//**********************************************************************
/*
 * Filename    : Temperature Humidity Meter
 * Description : LCD displays the value of temperature and humidity.
 * Auther      : http//www.keyestudio.com
*/
##include "xht11.h"
##include "lcd128_32_io.h"

//gpio13
xht11 xht(13);
unsigned char dht[4] = {0, 0, 0, 0};//Only the first 32 bits of data are received, not␣
↪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

			
ESP32*1	Breadboard*1	Ultrasonic Sensor*1	Red LED*4
			
M-F Dupont Wires	220Resistor*4	Jumper Wires	USB Cable*1

3. Component knowledge

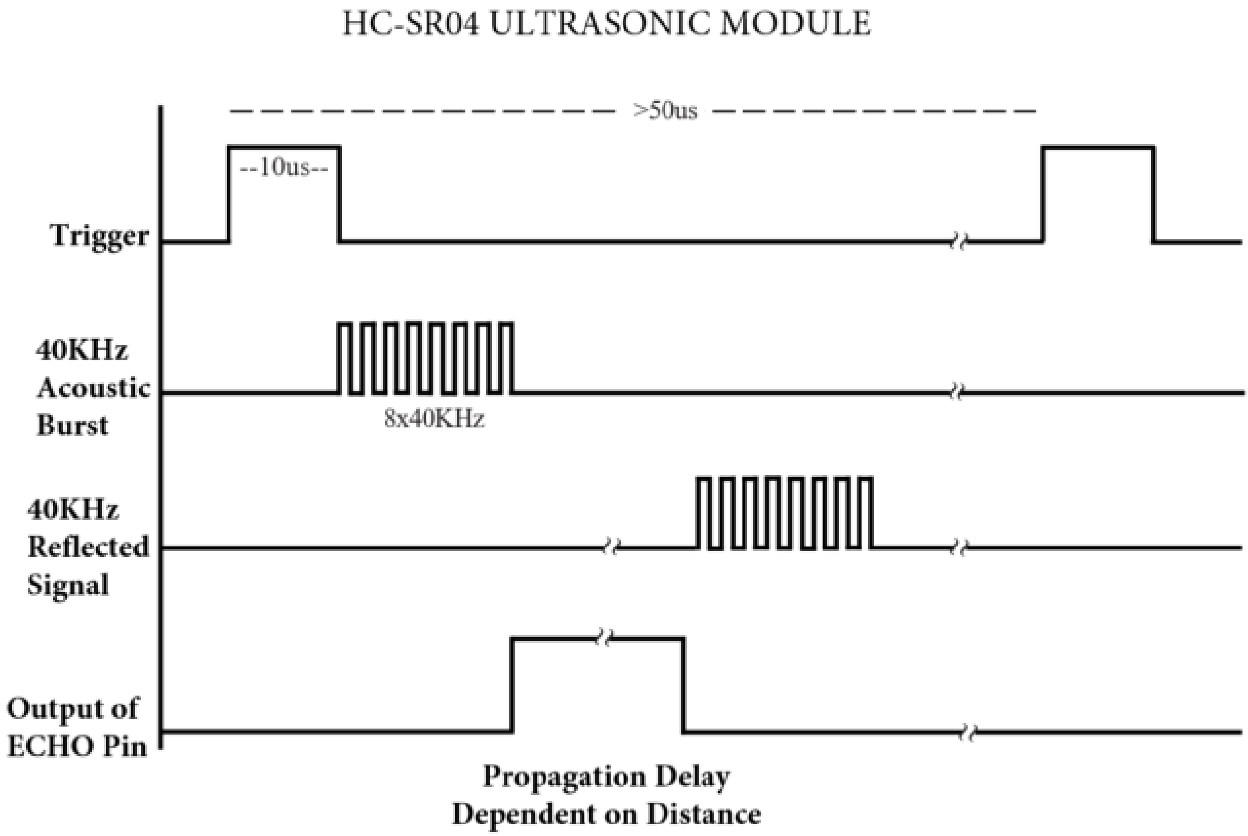
HC-SR04 Ultrasonic Sensor : Like bats, sonar is used to determine the distance to an object. It provides accurate non-contact range detection, high-precision and stable readings. Its operation is not affected by sunlight or black materials, just like a precision camera (acoustically softer materials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

media/e6f6037071e434febf7090b56ac35802.png

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.

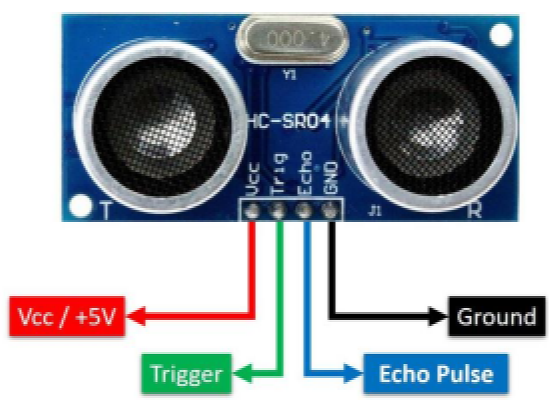


Time = Echo pulse width, its unit is "us" (microseconds) $Distance(cm) = time/58$

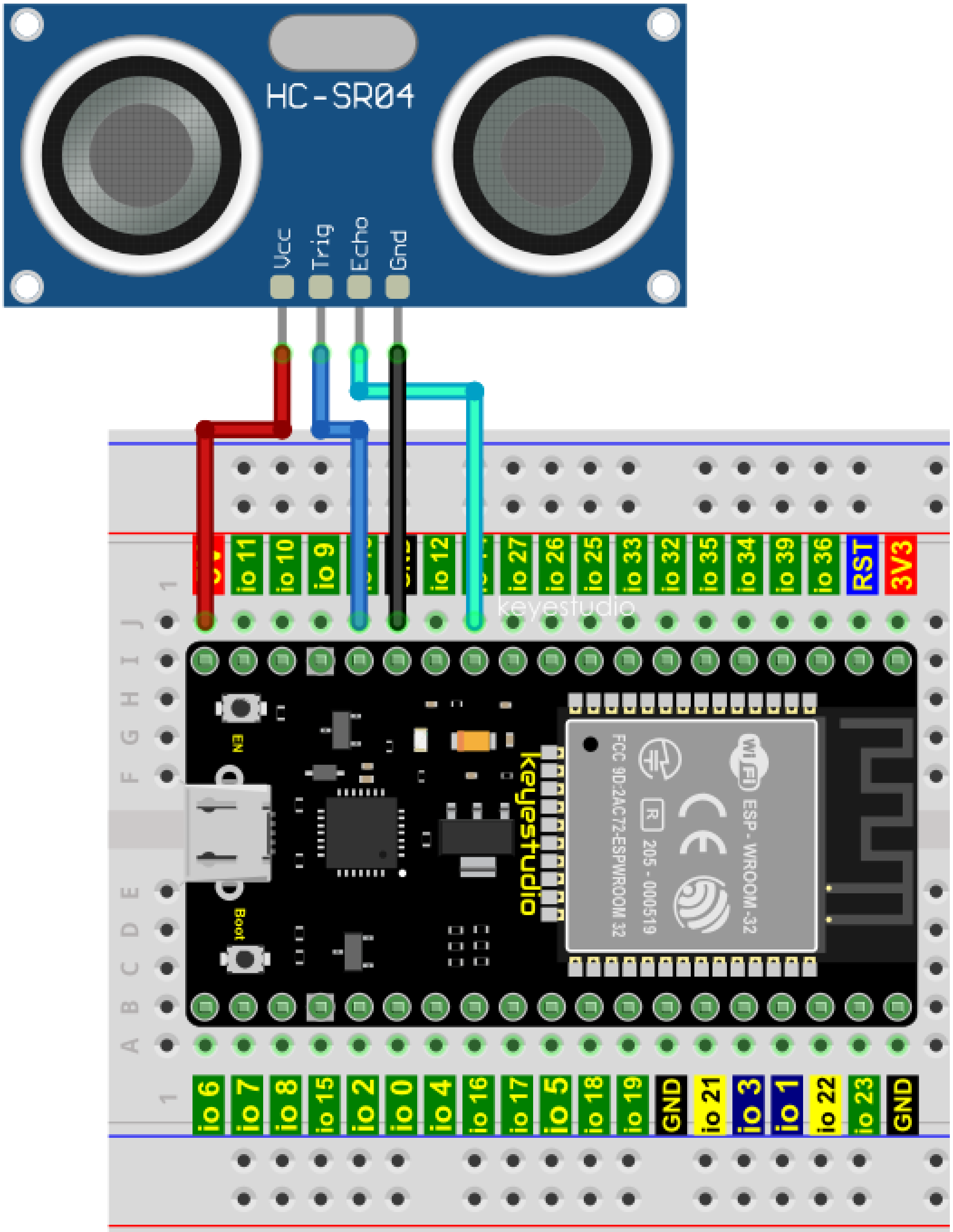
$$Distance(inch) = time/148$$

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"**.

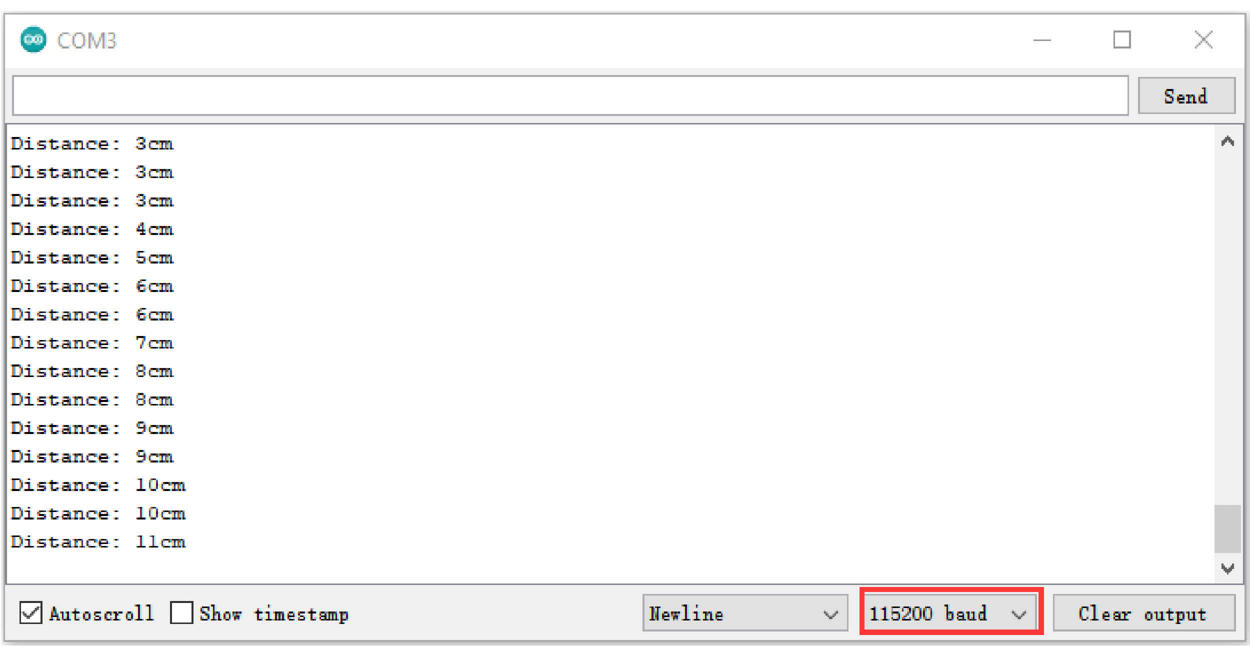
```
//**********************************************************************************
/*
 * Filename    : Ultrasonic Ranging
 * Description : Use the ultrasonic module to measure the distance.
```

(continues on next page)

(continued from previous 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.



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

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
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
```

(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 )
  {
    digitalWrite(PIN_LED2, HIGH);
  }
  else
  {
    digitalWrite(PIN_LED2, LOW);
  }
  if ( distance <= 15 )
  {
    digitalWrite(PIN_LED3, HIGH);
  }
  else
  {
    digitalWrite(PIN_LED3, LOW);
  }
  if ( distance <= 20 )
  {
    digitalWrite(PIN_LED4, HIGH);
  }
  else
  {
    digitalWrite(PIN_LED4, LOW);
  }
}
//*************************************************************************************
```

7. Project result

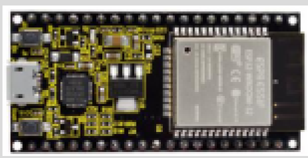
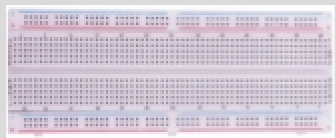
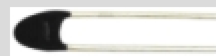
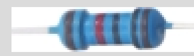
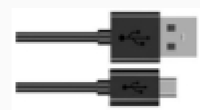
Compile and upload the code to ESP32, after the code is uploaded successfully, power up with a USB cable, open the serial monitor and set the baud rate to 115200. You will see that the serial port monitor window will print outthe distance between the ultrasonic sensor and the object, and the corresponding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

5.32 Project 31Temperature Instrument

1. Introduction

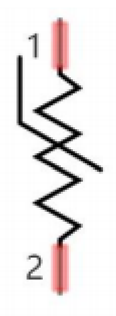
Thermistor is a kind of resistor whose resistance depends on temperature changes, which is widely used in gardening, home alarm system and other devices. Therefore, we can use the feature to make a temperature instrument.

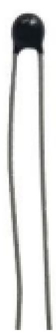
2. Components

			
ESP32*1	Breadboard*1	Thermistor*1	10KResistor*1
			
M-F Dupont Wires	LCD 128X32 DOT*1	Jumper Wires	USB Cable*1

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;

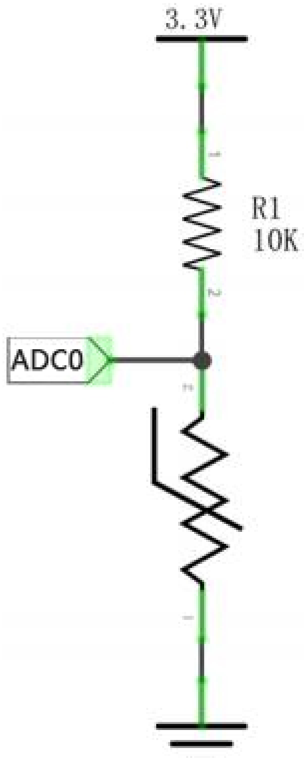
EXP[n] is nth power of e;

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:

$$T2 = 1 / \left(\frac{1}{T1} + \ln\left(\frac{Rt}{R} \right) / B \right)$$

4. Read the value of the Thermistor

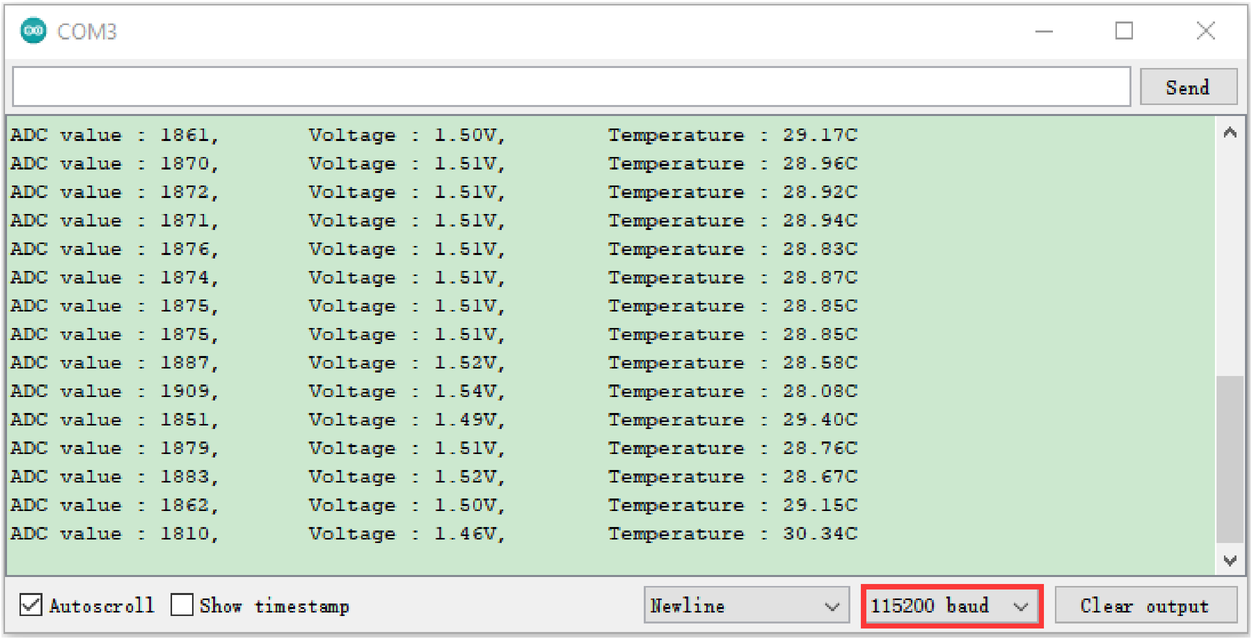
First we will learn the thermistor to read the current ADC value, voltage value and temperature value and print them out. Please connect the wires according to the wiring diagram below media/806fd81bf8a761b4ae1a638489c426ce.png

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() {
  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 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.



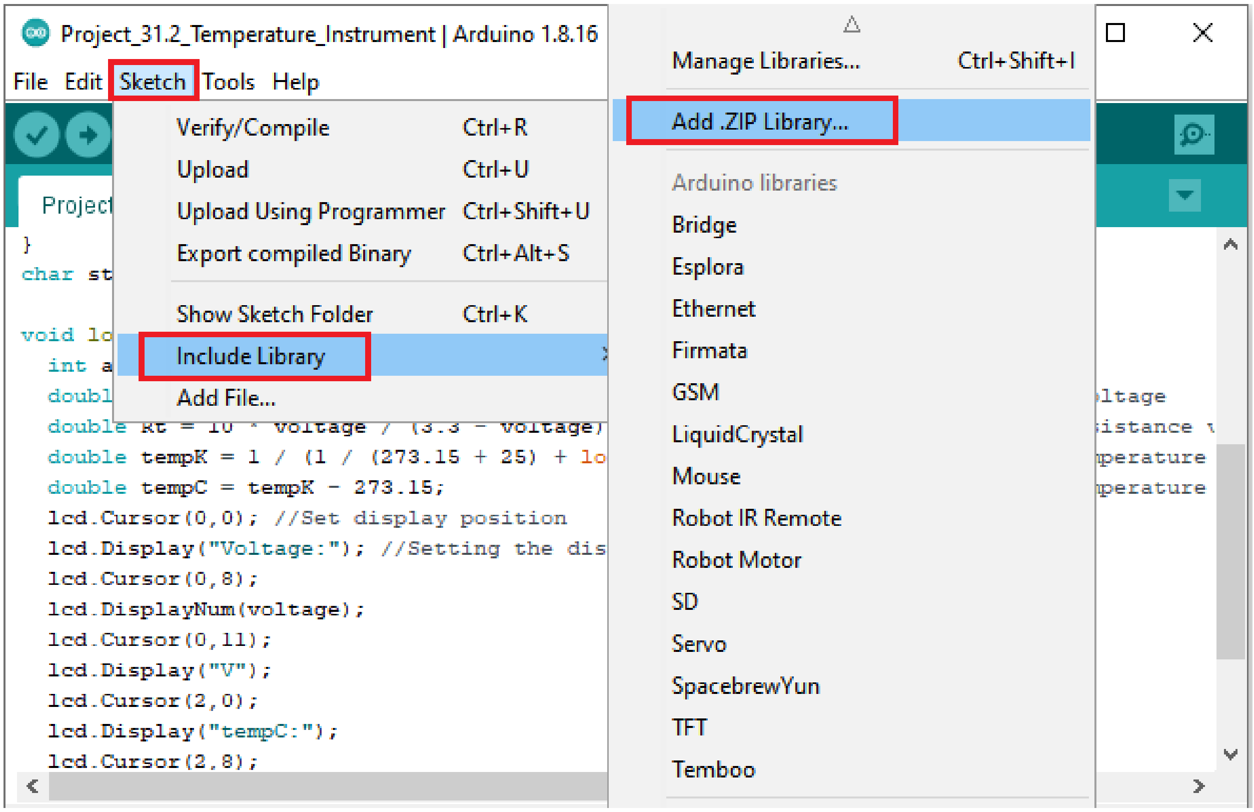
5. Wiring diagram of the temperature instrument

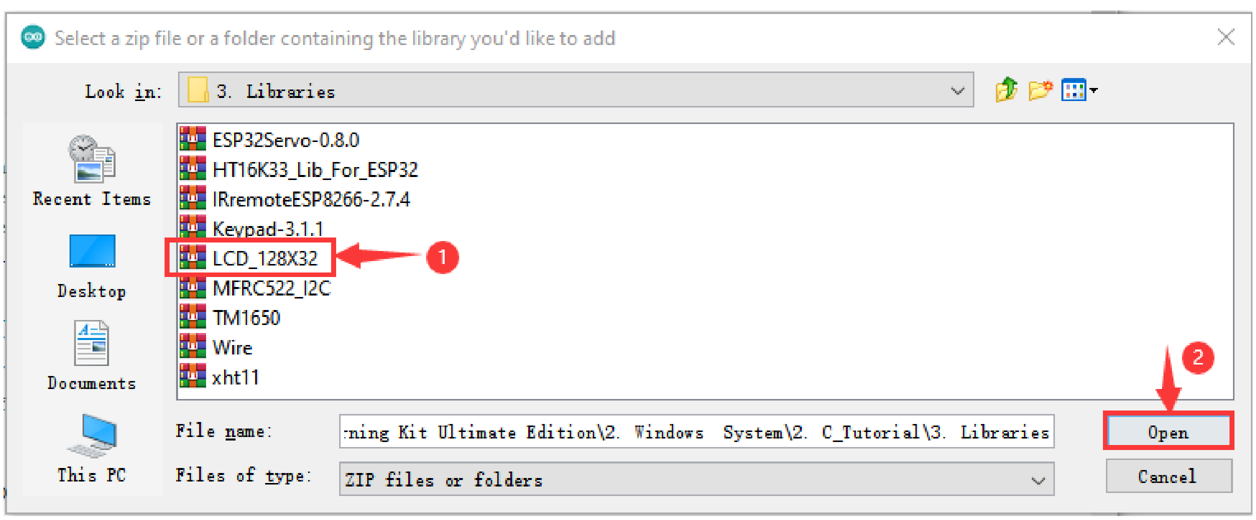
media/5a437bfdcad012211e15cab54e35dad7.png

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"→"Include Library"→"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␣
↪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(2,0);
  lcd.Display("tempC:");
  lcd.Cursor(2,8);
  lcd.DisplayNum(tempC);
  lcd.Cursor(2,11);
  lcd.Display("C");
  delay(200);
}
//**********************************************************************************
```

8. Project result

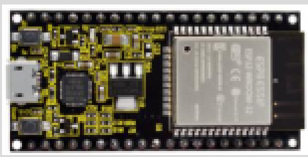
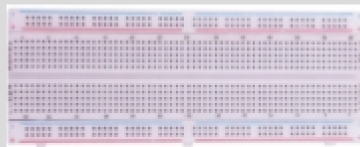
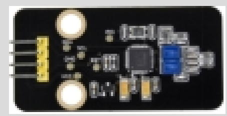
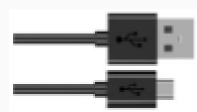
Compile and upload the code to ESP32, after the code is uploaded successfully, power up with a USB cable and you will see that the LCD 128X32 DOT displays the voltage value of the thermistor and the temperature value in the current environment.

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

		
ESP32*1	Breadboard*1	RFID-RC522 Module*1
		
M-F Dupont Wires	Servo*1	White Card*1
		
Key Chain*1	Jumper wires	USB Cable*1

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

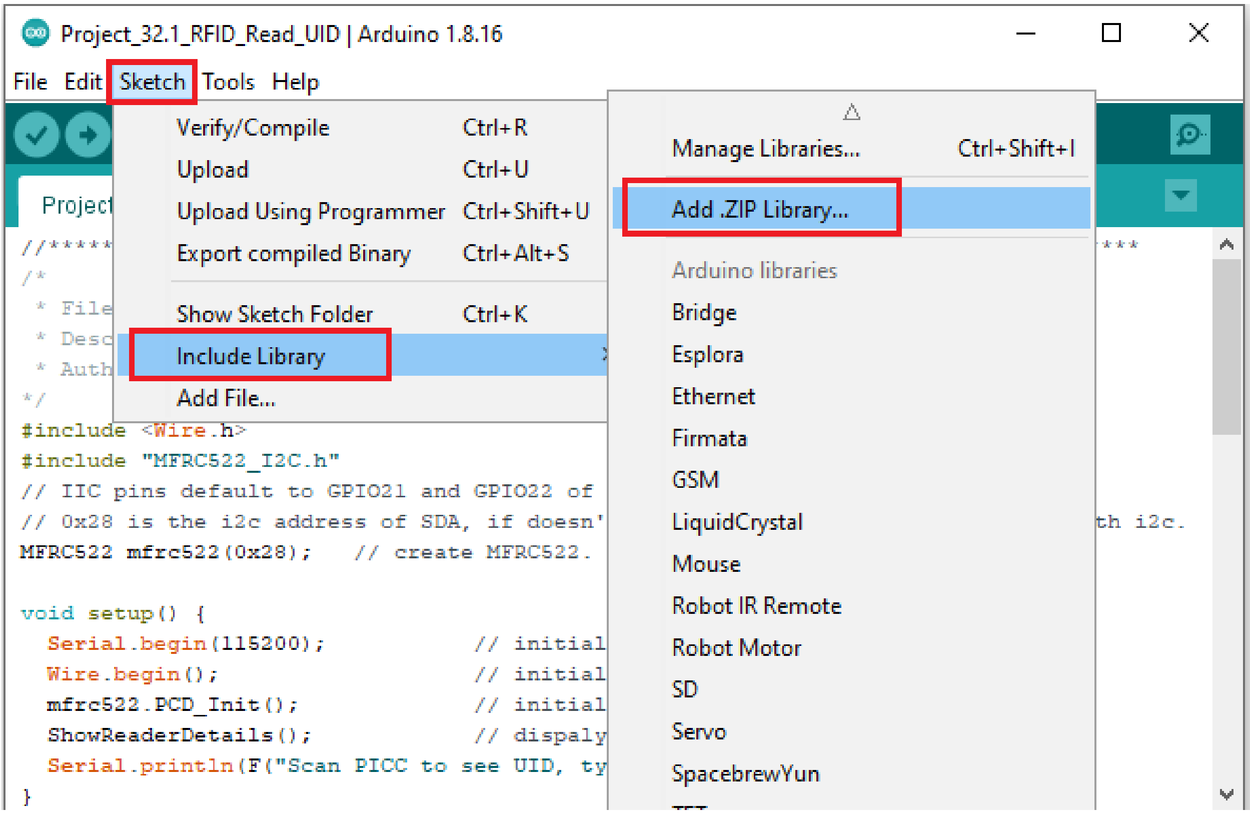
We will read the UNIQUE ID number (UID) of the RFID card and identify the type of the RFID card, and display the relevant information through the serial port. The wiring diagram is shown below

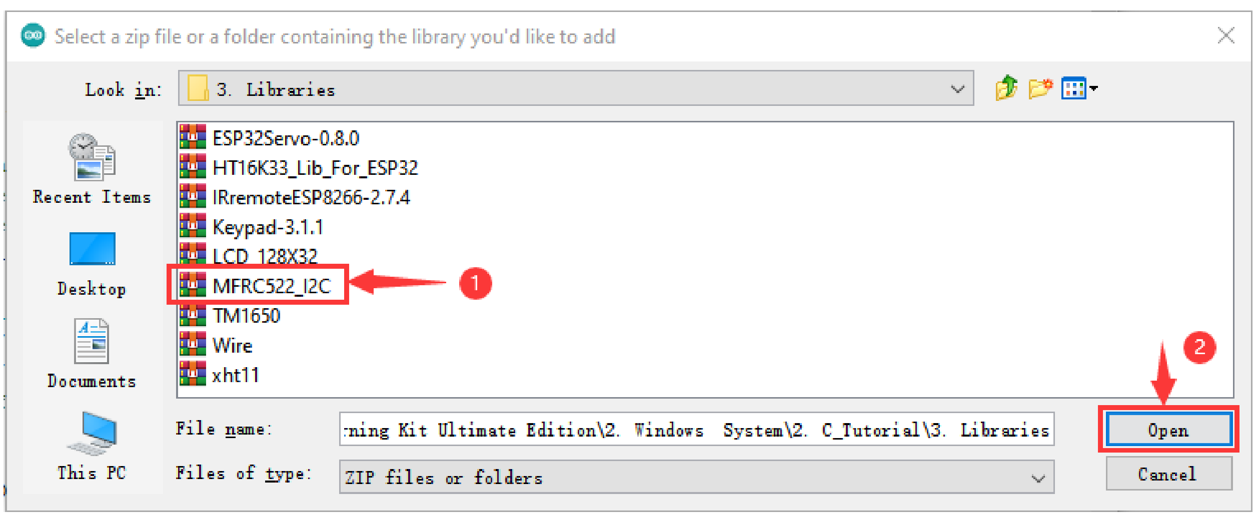
media/1cdb3ffd7f392f29451aeed5c3257133.png

Adding the MFRC522_I2C and Wire libraries

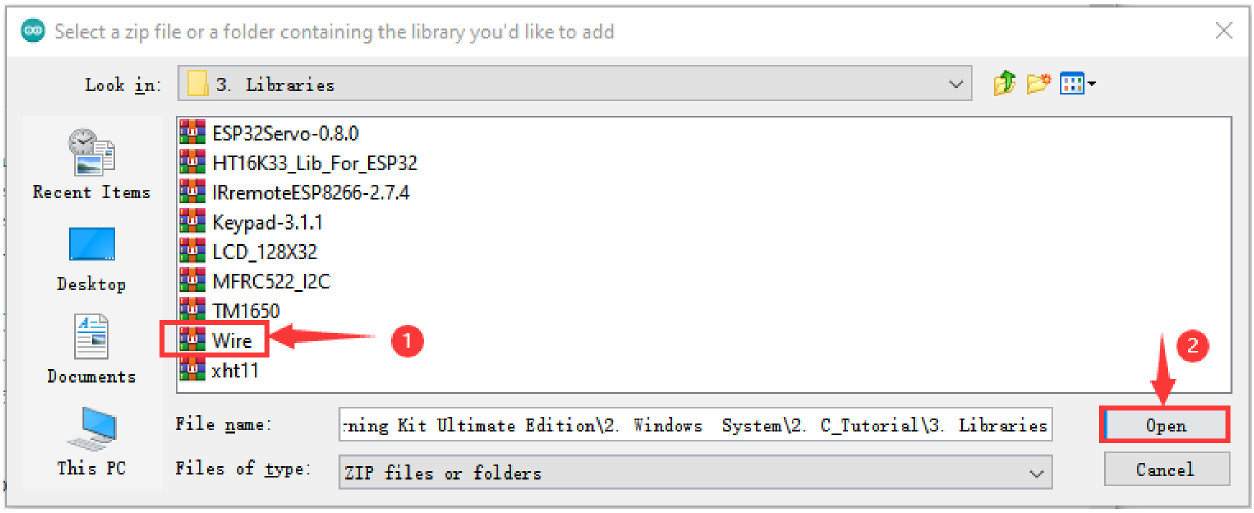
If you haven't installed the **MFRC522_I2C** and **Wire** libraries yet, please do so before learning. The steps to add third-party libraries are as follows:

Open the Arduino IDEclick"Sketch"→"Include Library"→"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”.



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"**.

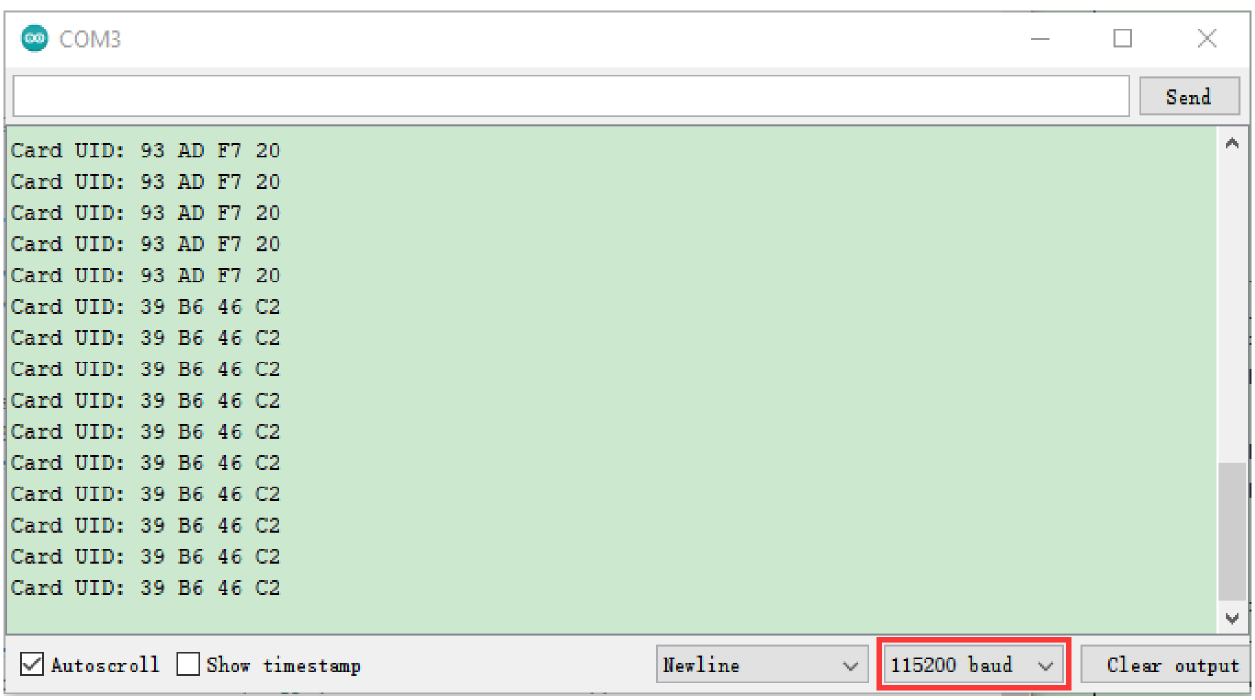
```
//**********************************************************************************
/*
 * Filename    : RFID
 * Description : RFID reader UID
 * Auther      : http//www.keyestudio.com
*/
##include <Wire.h>
##include "MFRC522_I2C.h"
// IIC pins default to GPIO21 and GPIO22 of ESP32
// 0x28 is the i2c address of SDA, if doesn't matchplease check your address with i2c.
MFRC522 mfrc522(0x28);   // create MFRC522.

void setup() {
  Serial.begin(115200);           // initialize and PC's serial communication
  Wire.begin();                   // initialize I2C
  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() {
  //
  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" : " ");
    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:



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

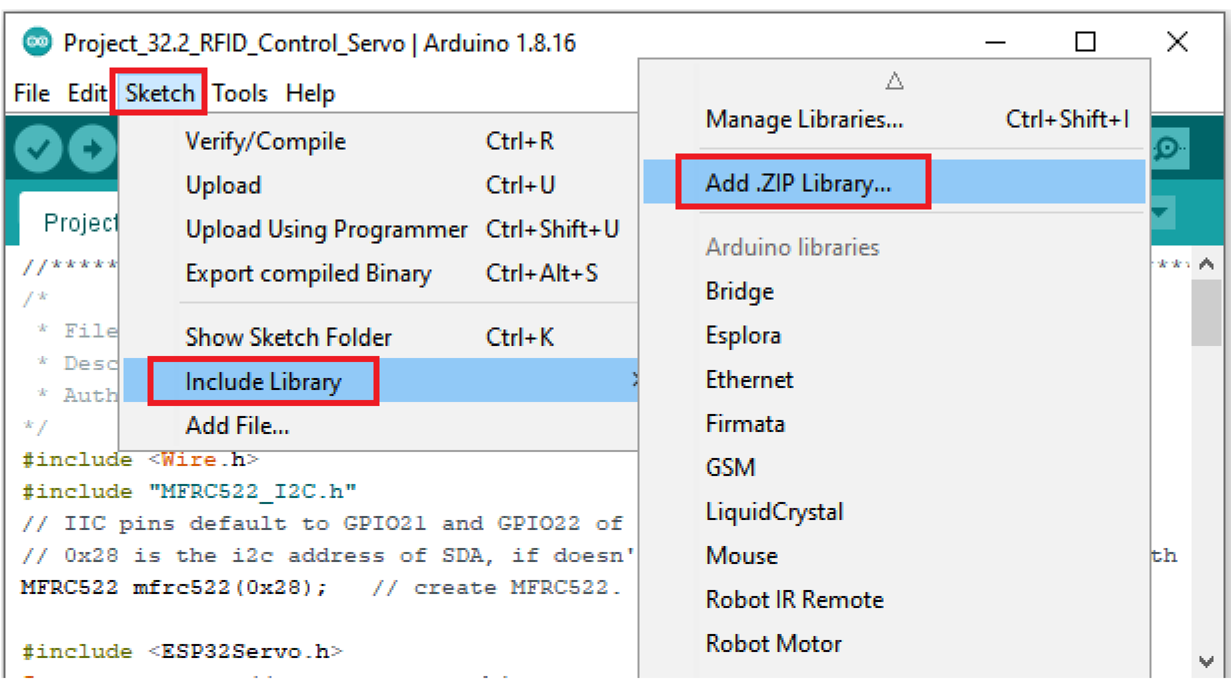
media/3ae6c0f1098d2aee34c51e7a96c25571.png

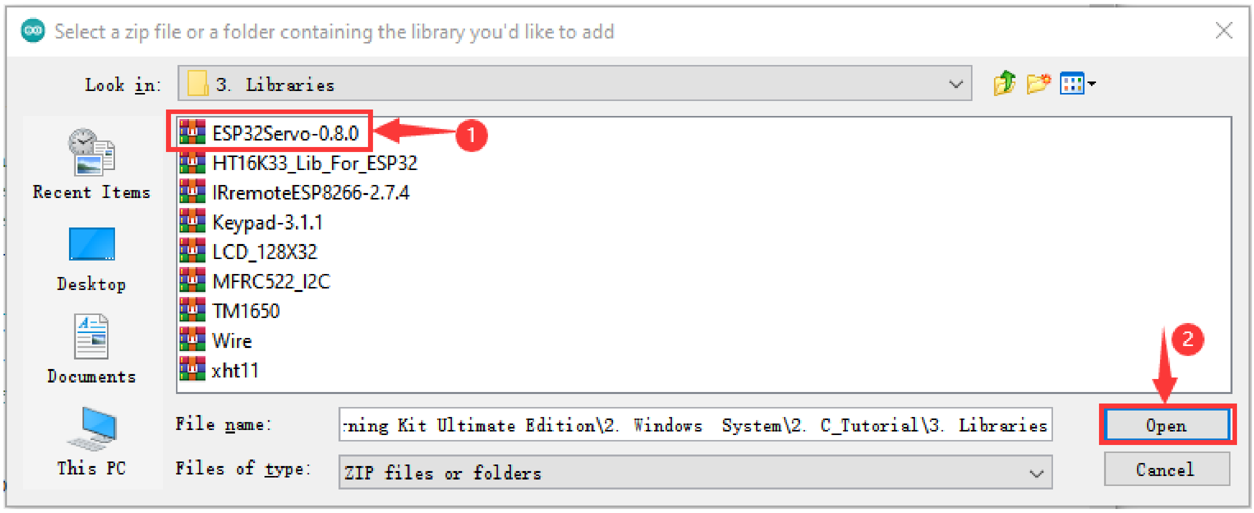
6. Adding the MFRC522_I2C, Wire and ESP32Servo libraries

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"→"Include Library"→"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".





Then, add the **MFRC522_I2C** and **Wire** libraries

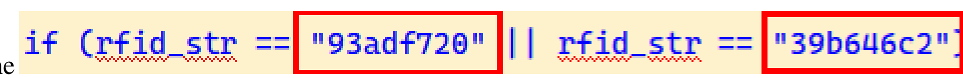
If both of them are added, just skip this step.

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.

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 GPIO21 and GPIO22 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++) {
    rfid_str = rfid_str + String(mfrc522.uid.uidByte[i], HEX);  //Convert to string
    //Serial.print(mfrc522.uid.uidByte[i] < 0x10 ? " 0" : " ");
    //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?
```

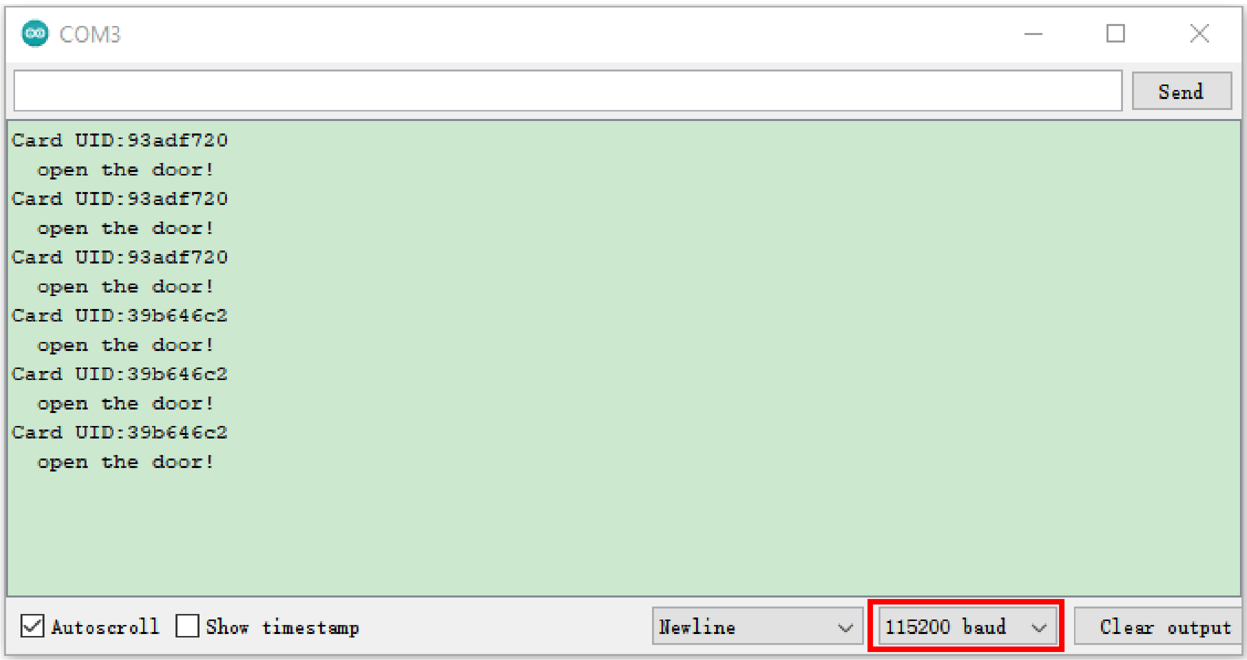
(continues on next page)

(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.

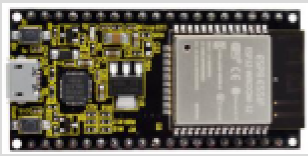
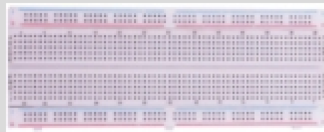
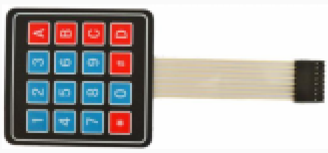
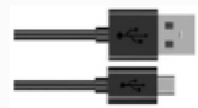
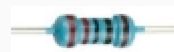


5.34 Project 33Keypad Door

1. Introduction

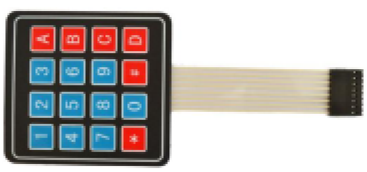
Commonly used digital button sensor, one button uses an IO port. However, it will occupy too many IO ports when we need a lot of buttons. In order to save the use of IO ports, the multiple buttons are made into a matrix type, through the control of the line and row to achieve less IO port control of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

2. Components

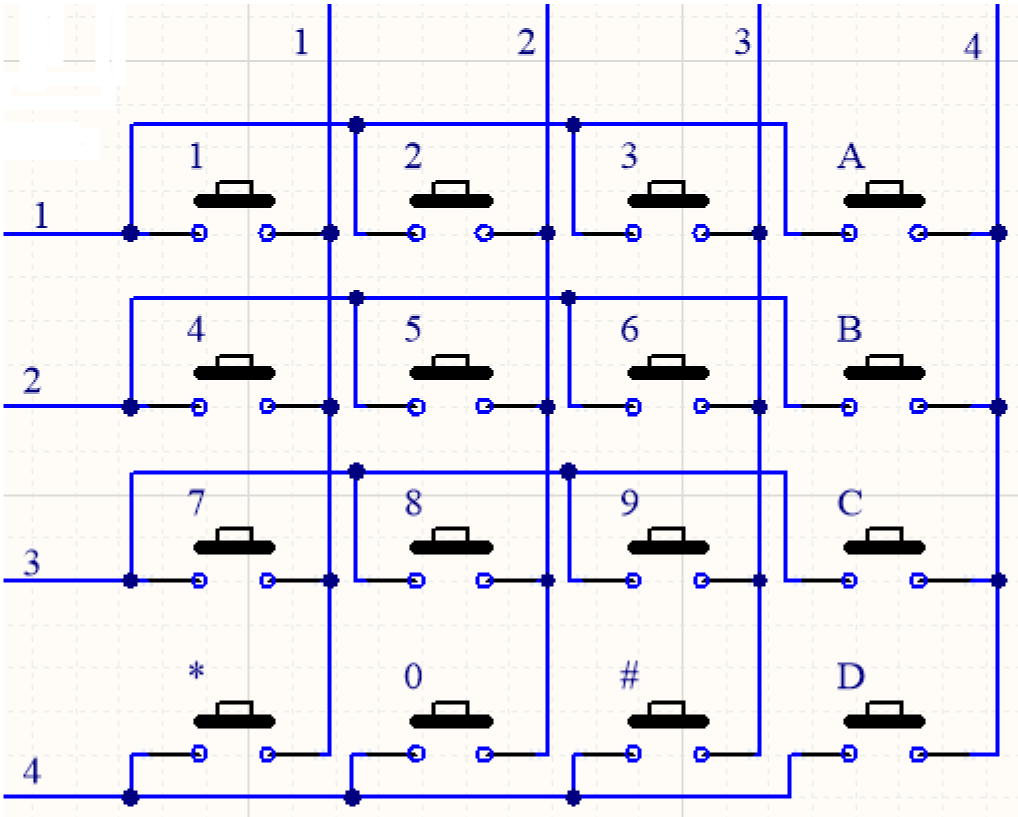
			
ESP32*1	Breadboard*1	Servo*1	Active Buzzer*1
			
44 *Membrane Matrix Keyboard*1	Jumper Wires	USB Cable*1	1kResistor*1
			
NPN transistor(S8050)*1			

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:



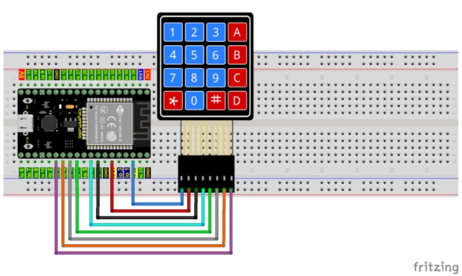
Similar to the integration of an LED Matrix, the 4x4 Keypad Matrix has each row of keys connected with one pin and this is the same for the columns. Such efficient connections reduce the number of processor ports required. The internal circuit of the Keypad Matrix is shown below.



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



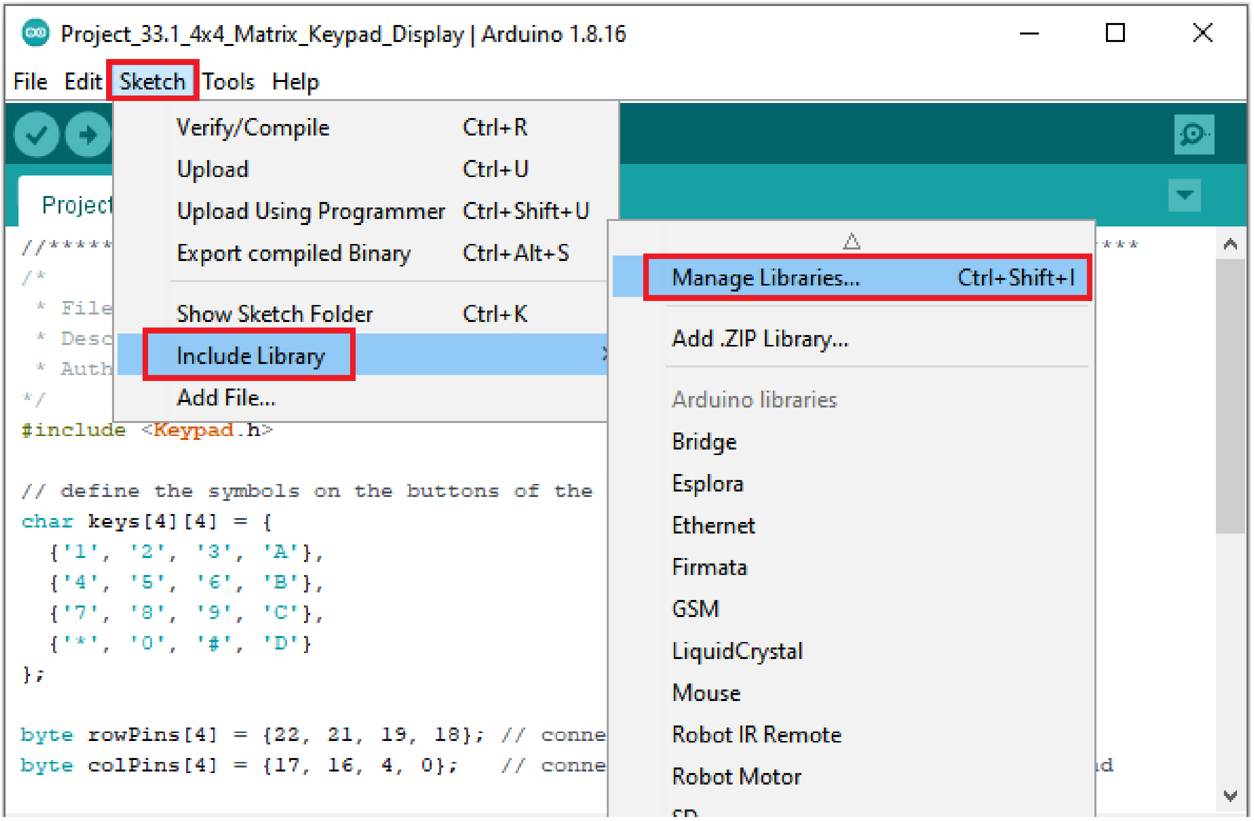
How to add the Keypad library

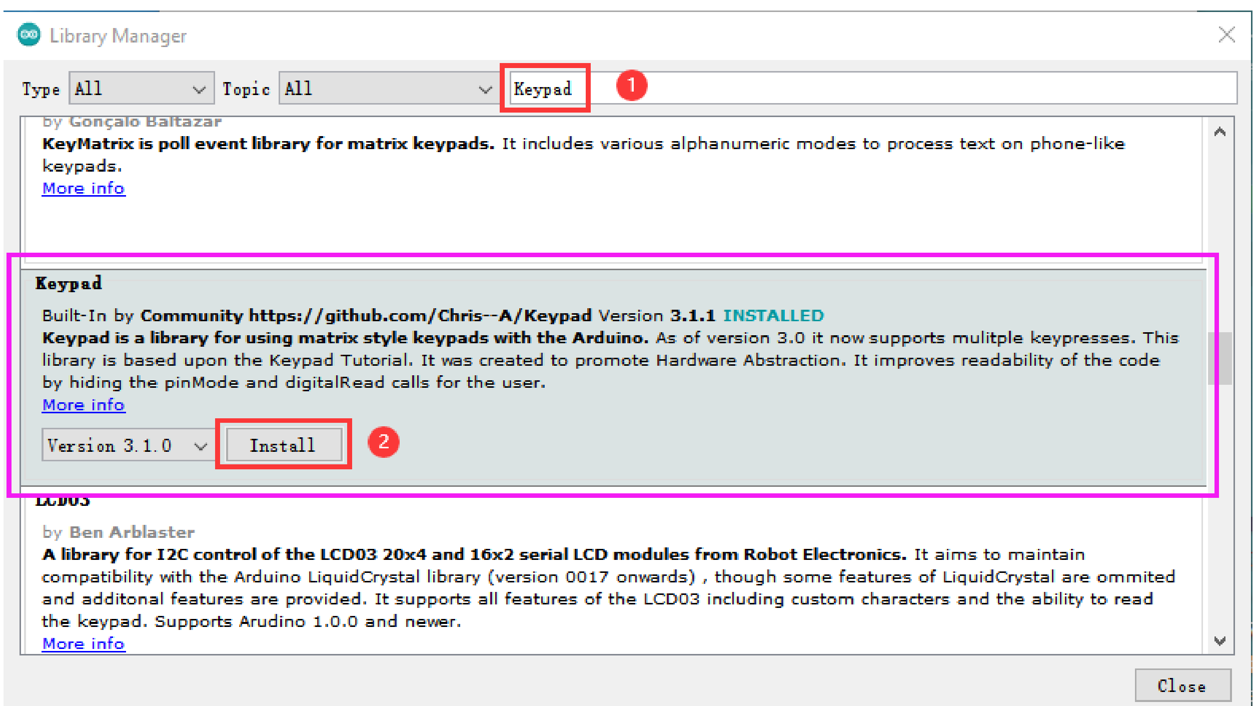
If you have not downloaded the library file, please click on the link to download it:Download Arduino Libraries

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:

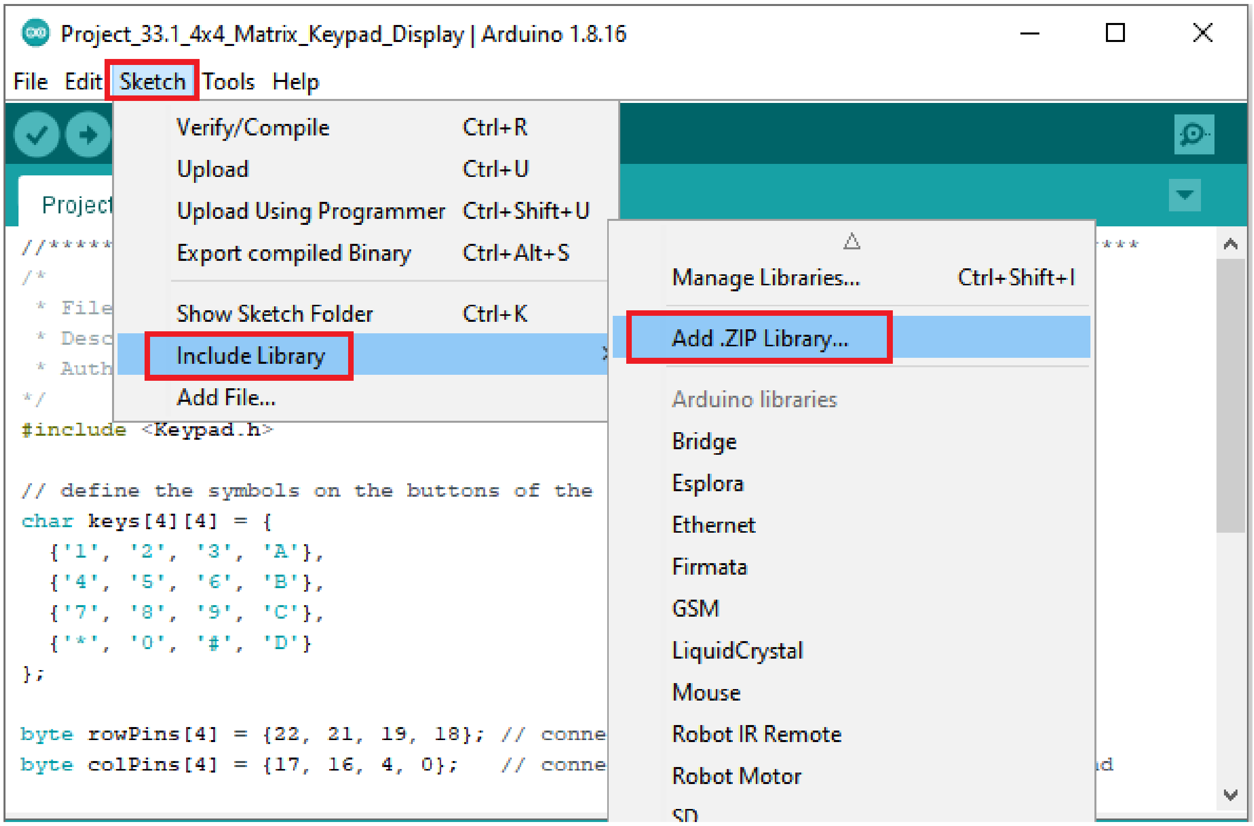
There are two ways to add libraries:

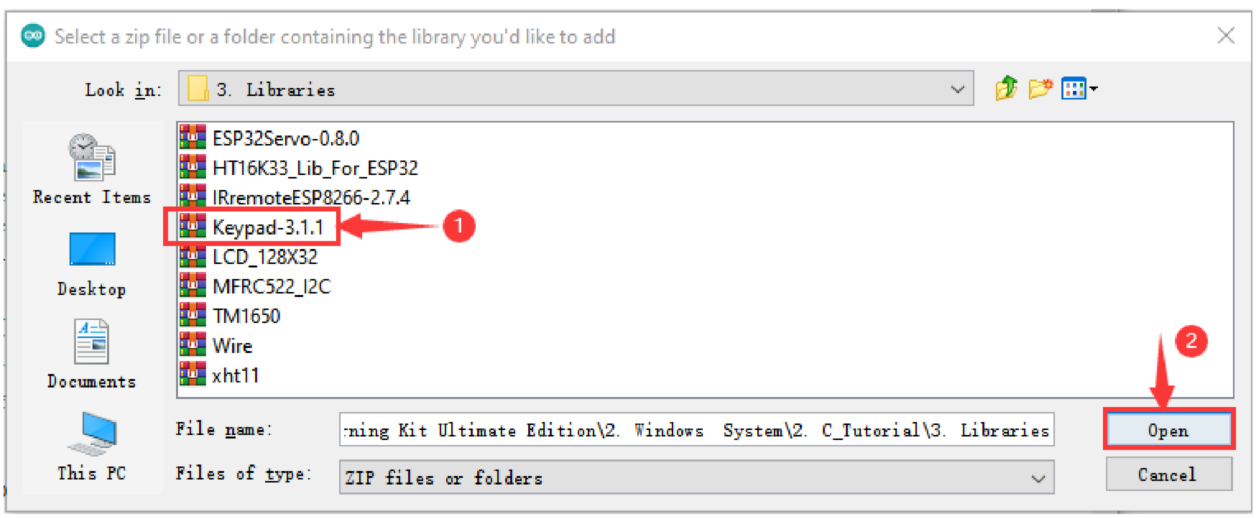
The first way, open the Arduino IDE, click"Sketch"→"Include Library"→"Manage Libraries…". Enter"Keypad"in the search box, select"Keypad"and click"Update"to install. Please refer to the following operations :





The second wayopen the Arduino IDEclick “Sketch”→“Include Library”→“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”.





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 **".

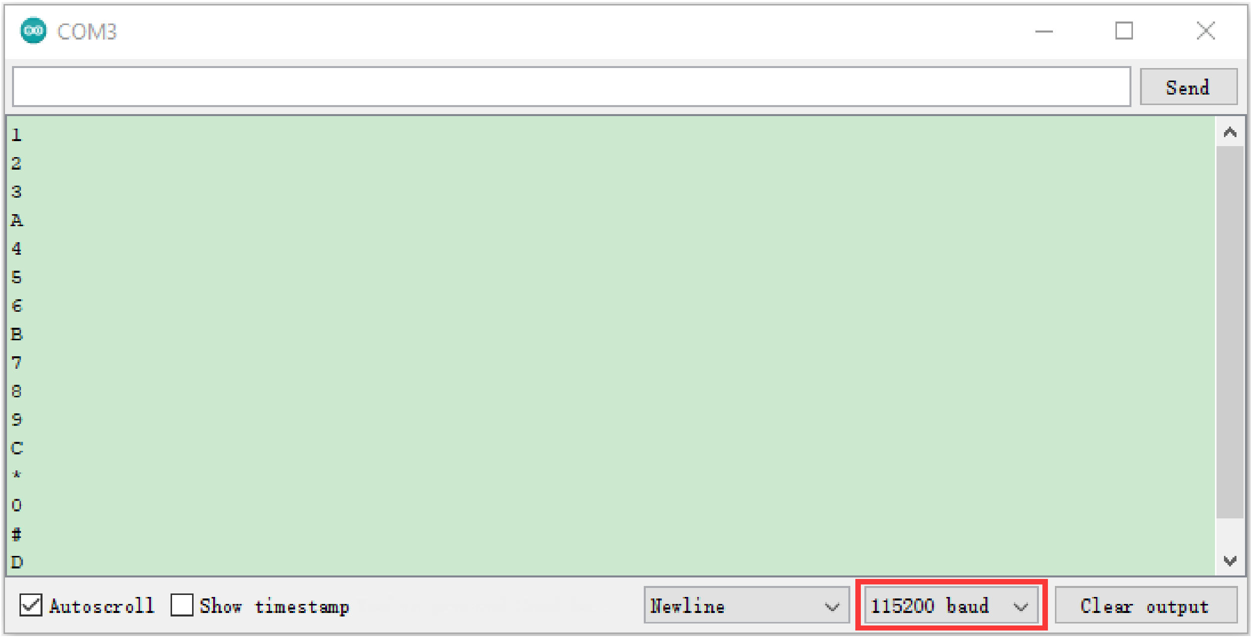
```
//*************************************************************************************
/*
 * Filename    : 4x4 Matrix Keypad Display
 * Description : Get the value for the matrix keyboard
 * Auther      : http//www.keyestudio.com
*/
```

(continues on next page)

(continued from previous page)

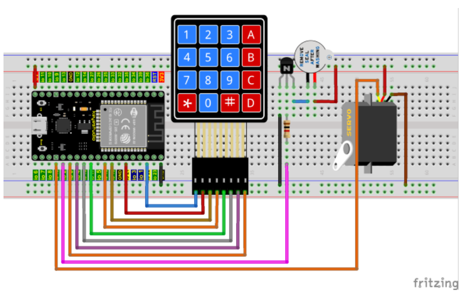
```
##include <Keypad.h>

// 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.



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.

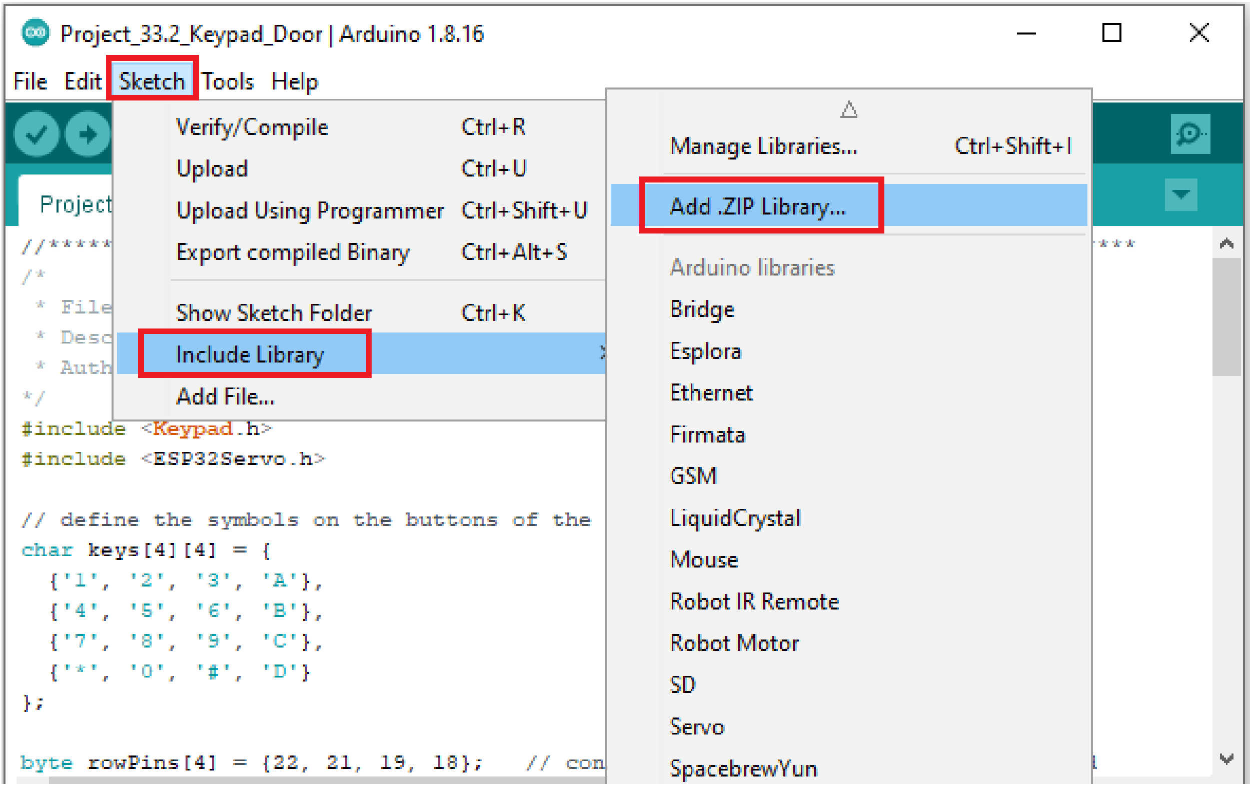


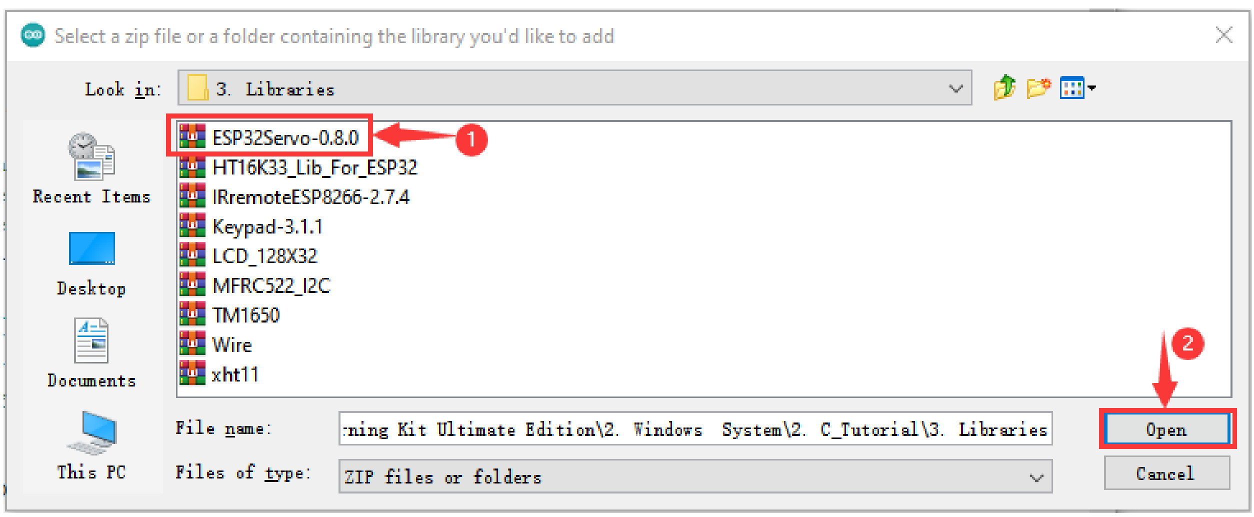
6. Adding the Keypad and ESP32Servo** **libraries

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

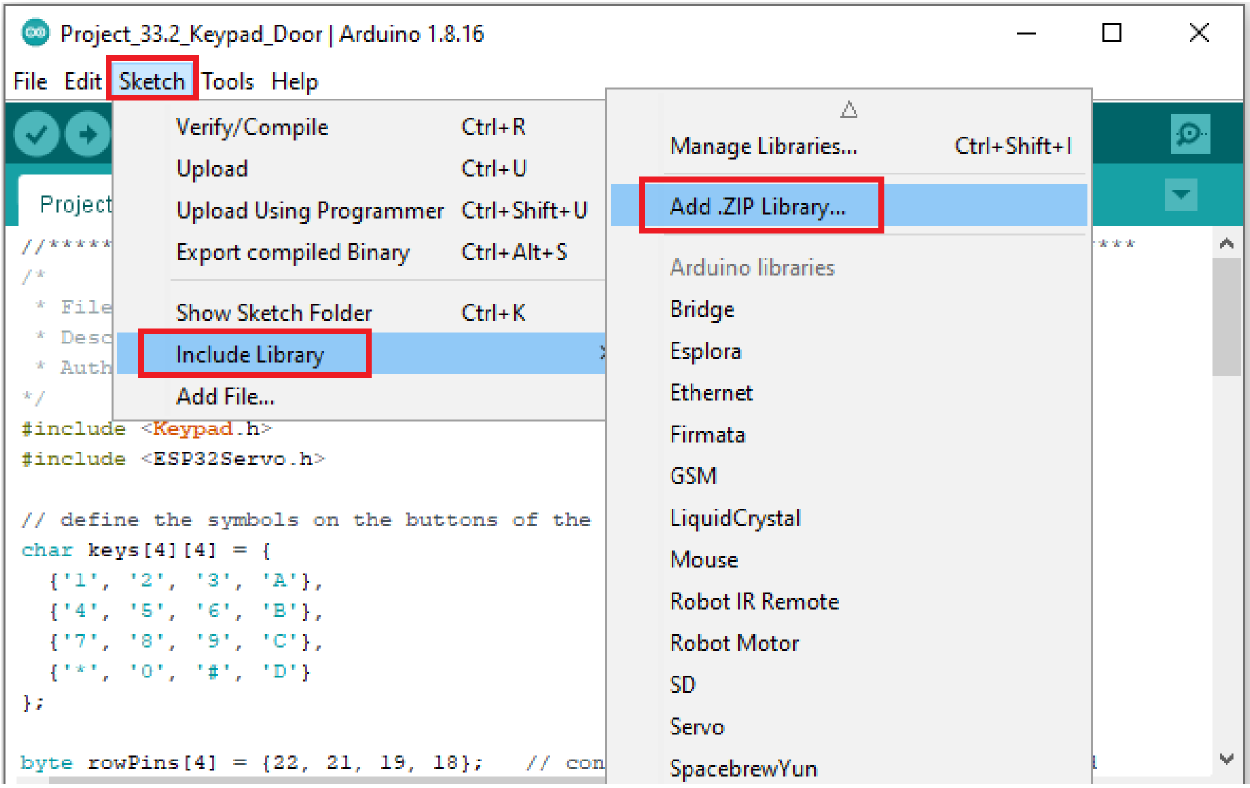
Open the Arduino IDEclick"Sketch"→"Include Library"→"Add .ZIP Library…". In the pop-up window, find the file named **"2. Windows System\2. C_Tutorial\3.Libraries\ESP32Servo-0.8.0.ZIP"** which locates in this directory. Select the **ESP32Servo-0.80.ZIP** file and then click"Open".

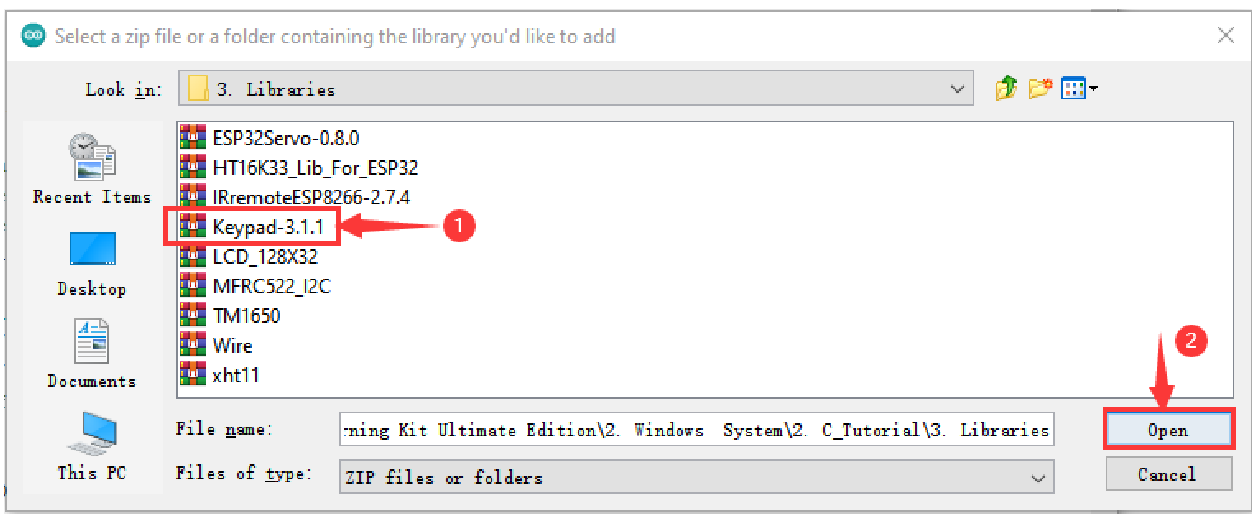




Then, add the **Keypad** library

Open the Arduino IDEclick “Sketch”→“Include Library”→“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"**.

```
//**********************************************************************************
/*
 * Filename    : Keypad_Door
 * Description : Make a simple combination lock.
 * Auther      : http//www.keyestudio.com
```

(continues on next page)

(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␣
↪character.
      }
      if (isRight) {                  // If the input password is right
        myservo.write(90);           // Open the switch
        delay(2000);                  // Delay a period of time
        myservo.write(0);            // Close the switch
        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.

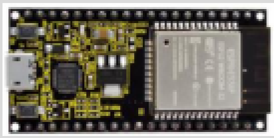
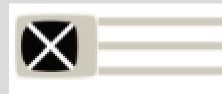
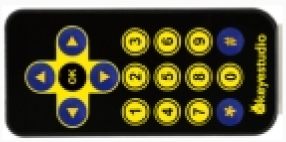
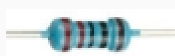
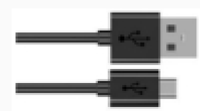
5.35 Project 34IR Control Sound and LED

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.

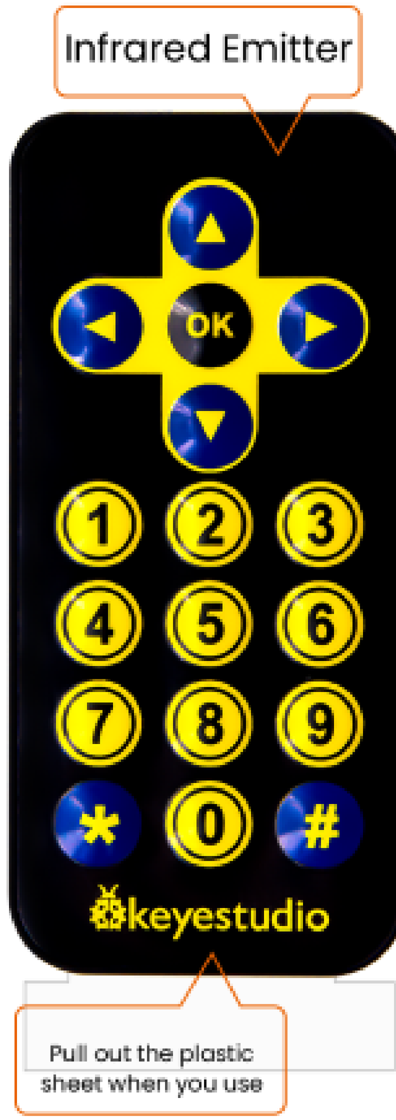
2. Components

			
ESP32*1	Breadboard*1	IR Receiver *1	RGB LED*1
			
IR Remote Controller*1	Active buzzer*1	10KResistor*1	220Resistor*3
			
NPN transistor(S8050)*1	1kResistor*1	Jumper Wires	USB Cable*1

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:

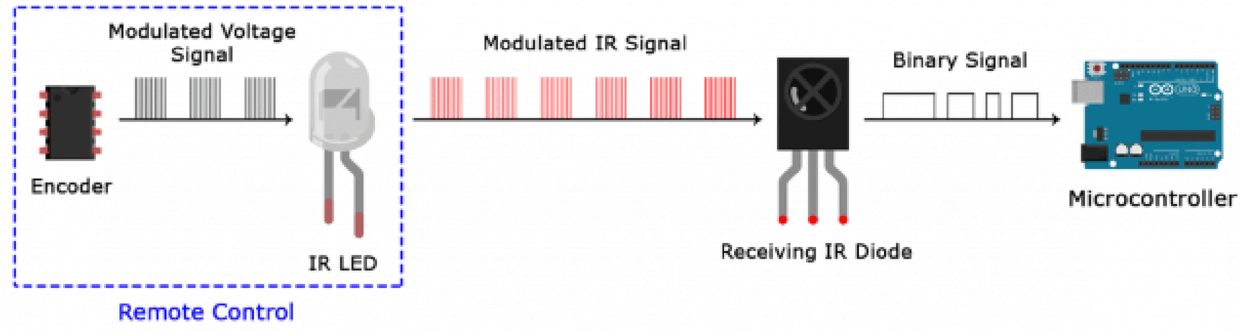
The infrared remote controller adopts NEC code and the signal cycle is 110ms.



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

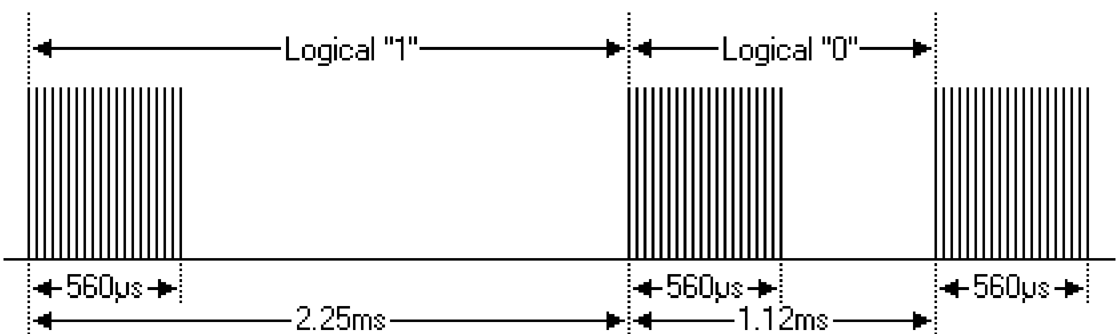
NEC Protocol:

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:

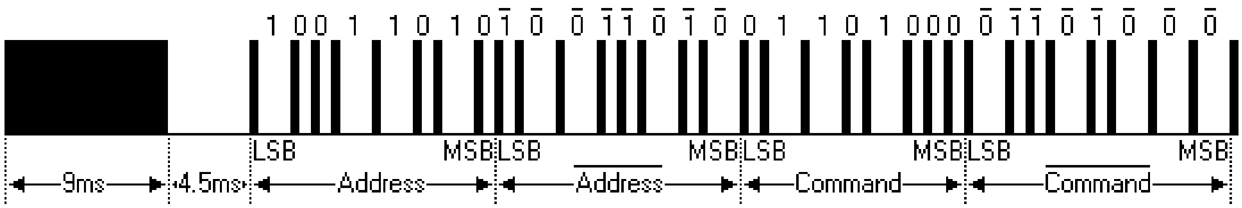
- * 8 bit address and 8 bit command length.
- * Extended mode available, doubling the address size.
- * Address and command are transmitted twice for reliability.
- * Pulse distance modulation.
- * Carrier frequency of 38kHz.
- * Bit time of 1.125ms or 2.25ms.

Modulation:



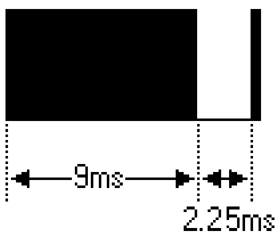
The NEC protocol uses pulse distance encoding of the bits. Each pulse is a 560µs long 38kHz carrier burst (about 21 cycles). A logical "1" takes 2.25ms to transmit, while a logical "0" is only half of that, being 1.125ms. The recommended carrier duty-cycle is 1/4 or 1/3.

Protocol:

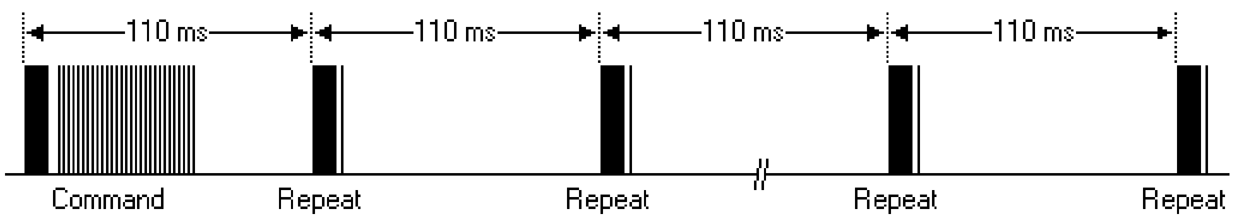


The picture above shows a typical pulse train of the NEC protocol. With this protocol the LSB is transmitted first. In this case Address $59andCommand16$ is transmitted. A message is started by a 9ms AGC burst, which was used to set the gain of the earlier IR receivers. This AGC burst is then followed 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 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.

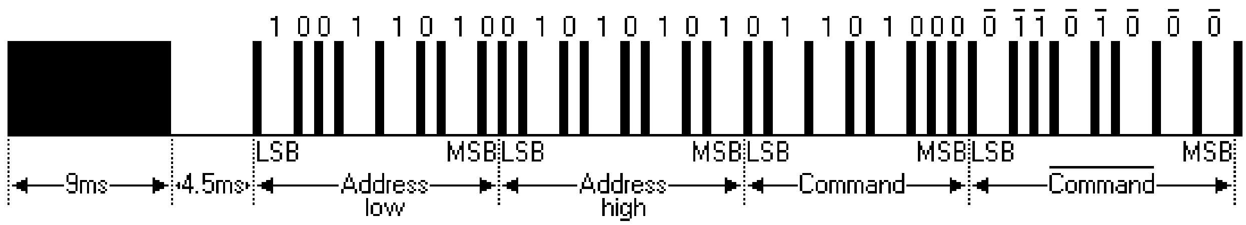


Extended NEC protocol:

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.

4. Decoded infrared signal

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

media/700496cb1e0d5d23fd63c28469dd3fd0.png

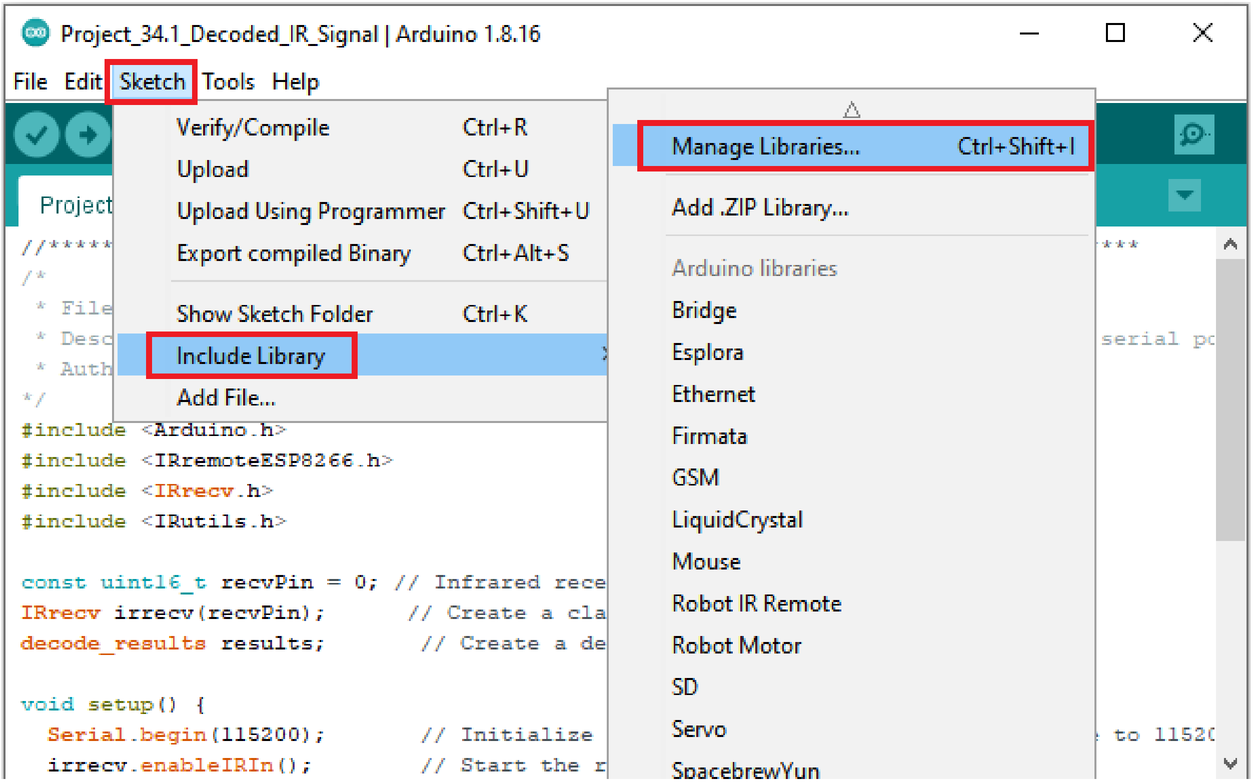
How to add the IRremoteESP8266 library

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

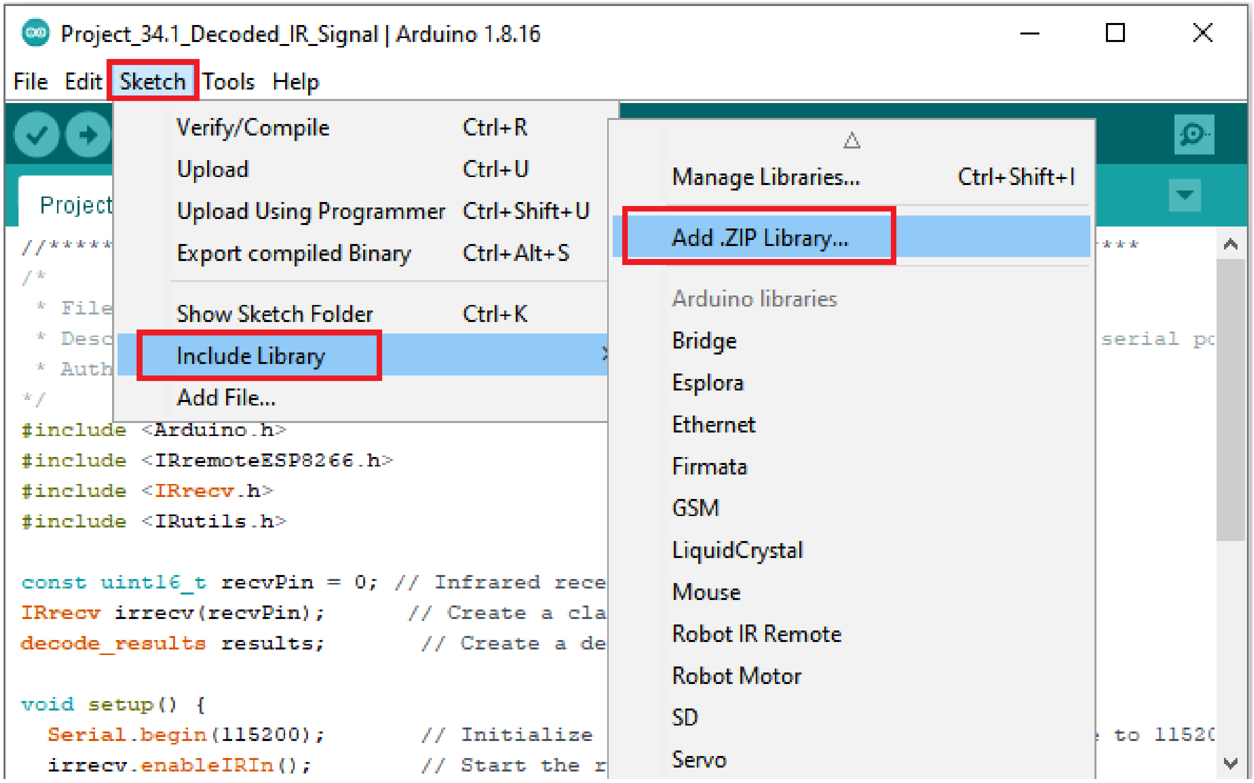
This code uses a library named "**IRremoteESP8266**", if you haven't installed it yet, please do so before learning. The steps to add third-party libraries are as follows:

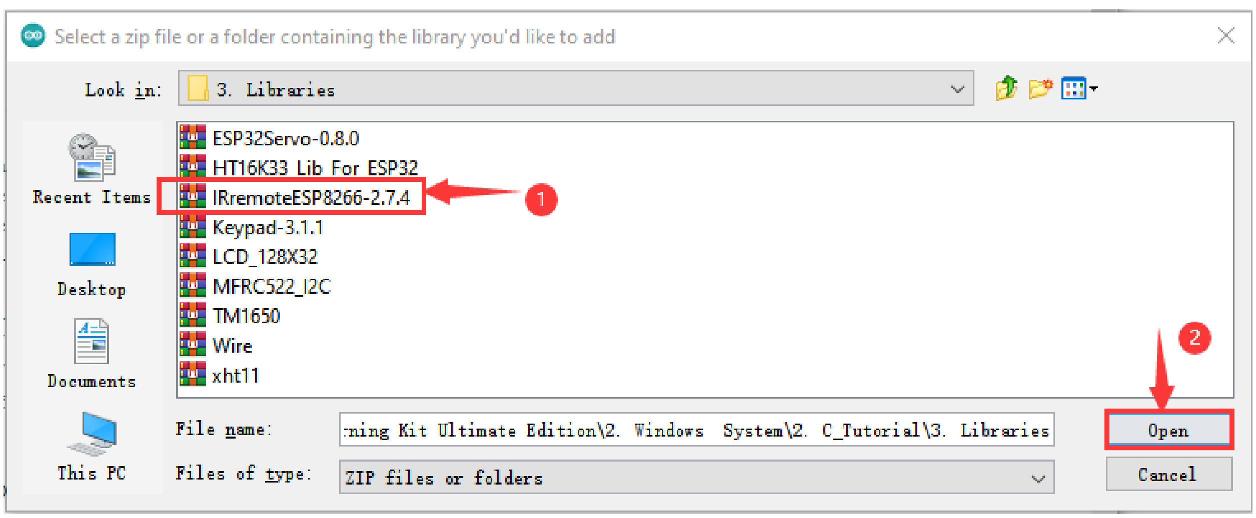
There are two ways to add libraries:

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



The second wayopen the“Sketch”→“Include Library”→“Add .ZIP Library…”. In the pop-up window, find the file named**“2. Windows System\2. C_Tutorial\3.Libraries\IRremoteESP8266-2.7.4.ZIP”**which locates in this directory. Select the **IRremoteESP8266-2.7.4.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"**.

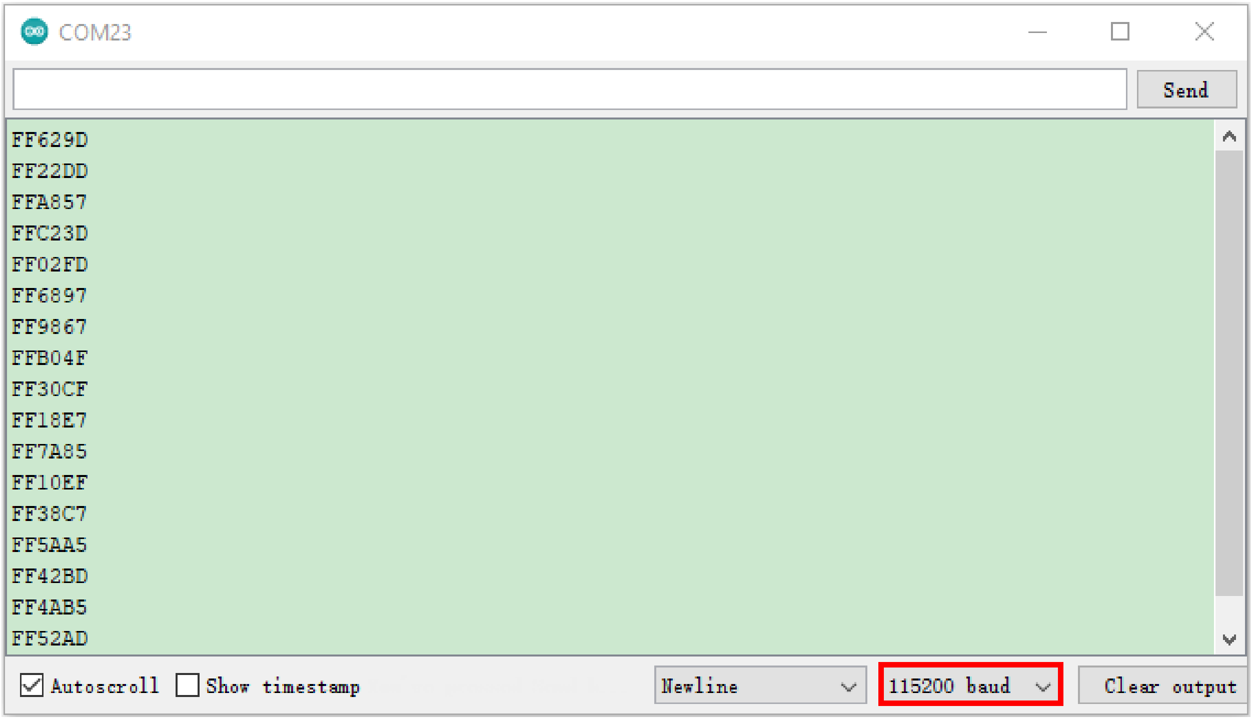
```
//**************************************************************************
/*
 * Filename    : Decoded IR Signal
 * Description : Decode the infrared remote control and print it out through the serial␣
↪port.
 * Auther      : http//www.keyestudio.com
```

(continues on next page)

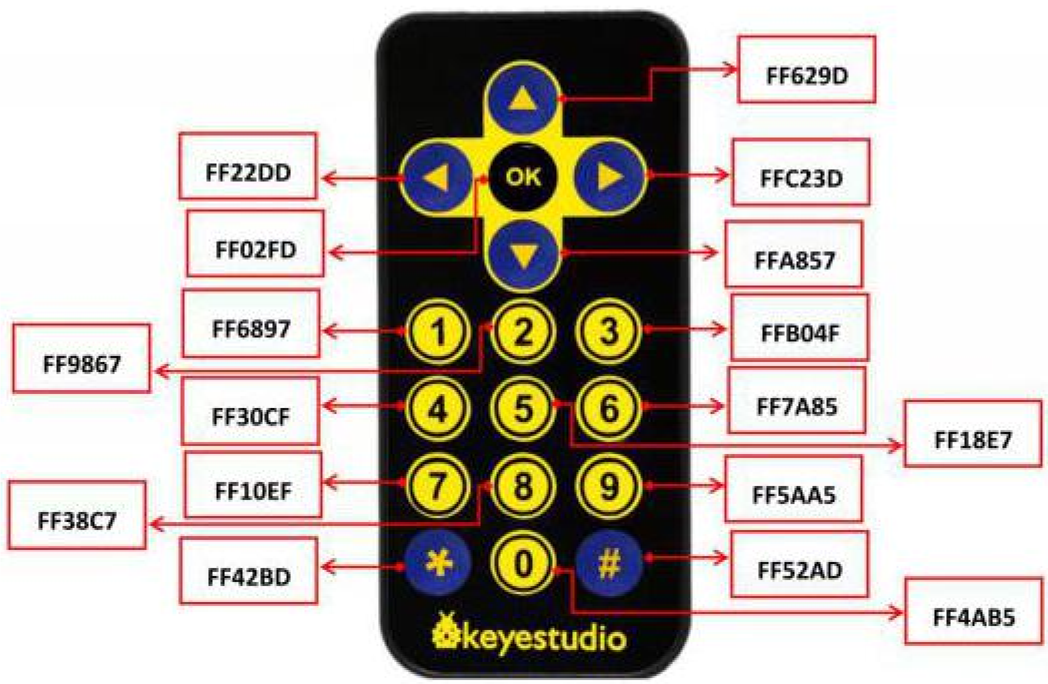
(continued from previous page)

```
*/
##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() {
  if (irrecv.decode(&results)) {          // Waiting for decoding
    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.



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
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
    ledcSetup(chns[i], 1000, 8);
    ledcAttachPin(ledPins[i], chns[i]);
  }
}

void loop() {
  if(irrecv.decode(&results)) {        // Waiting for decoding
    handleControl(results.value);       // Handle the commands from remote control
    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)

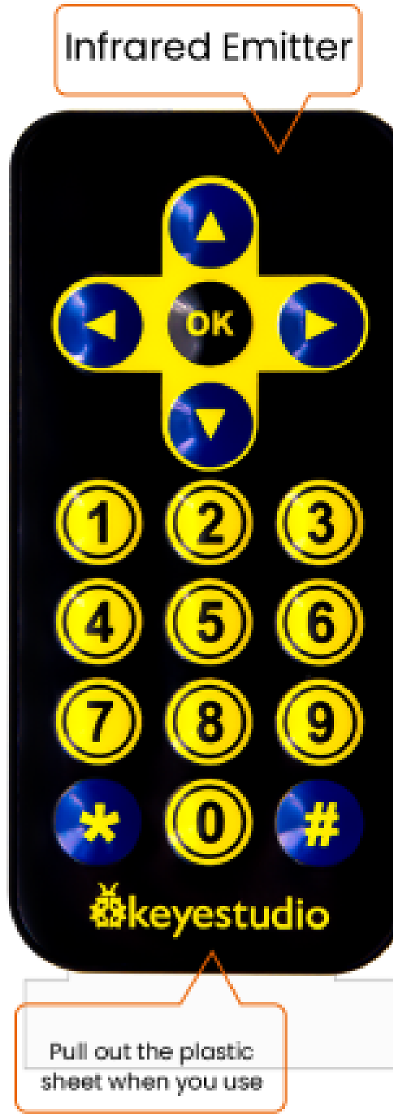
(continued from previous page)

```
    }
    else{
      ledcWrite(chns[0], 0);
      ledcWrite(chns[1], 0);
      ledcWrite(chns[2], 0);
      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 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.)

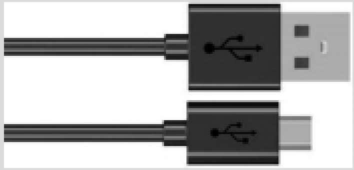
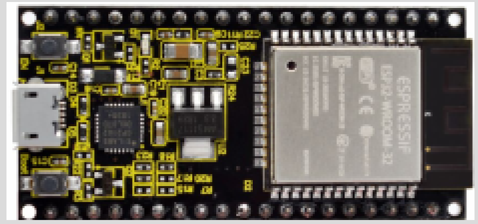


5.36 Project 35Bluetooth

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

	
USB Cable*1	ESP32*1

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.

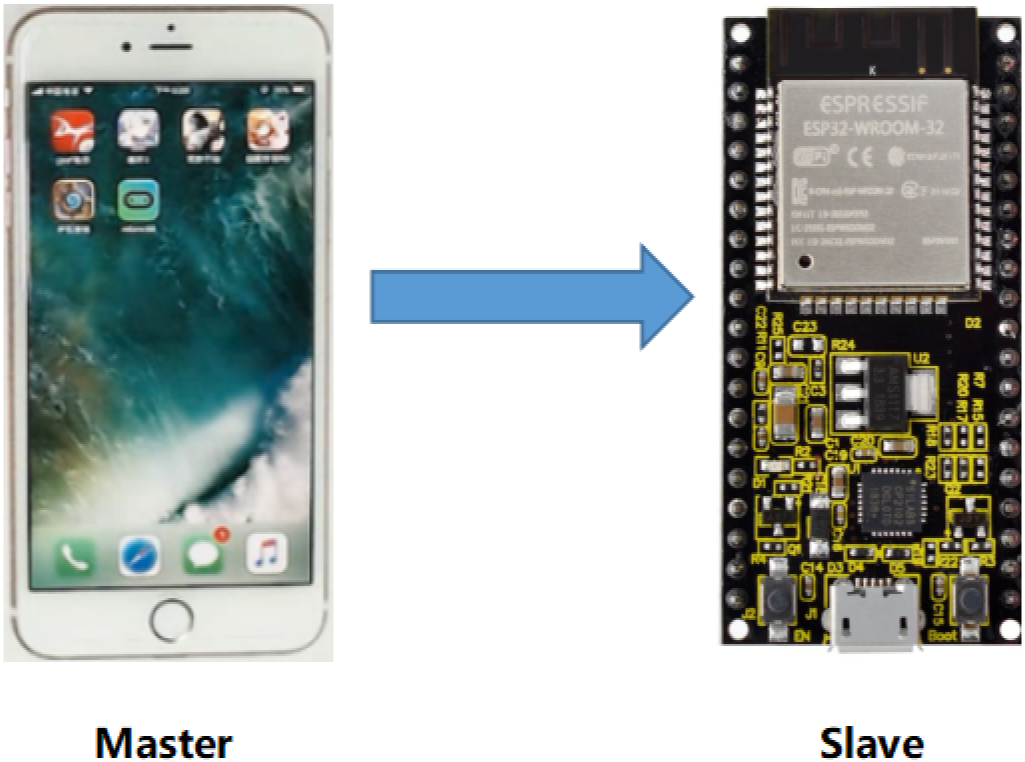
Master mode

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.

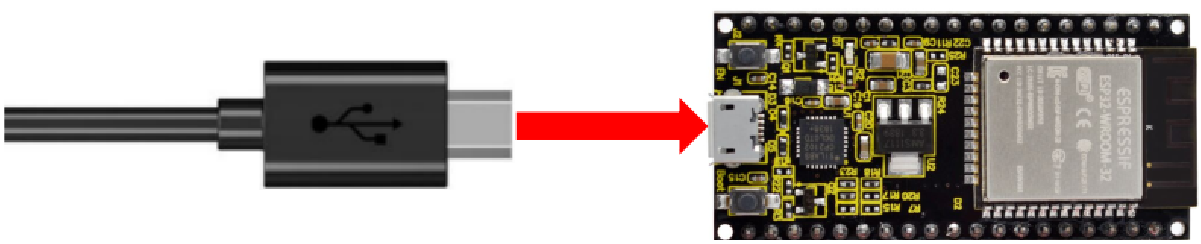
Bluetooth devices can make data interaction with each other, as one is in master mode and the other in slave mode. When they are making datainteraction, the Bluetooth device in master mode searches and selects devices nearby to connect to. When establishing connection, they can exchange data. When mobile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



Master Slave

3. Wiring Diagram

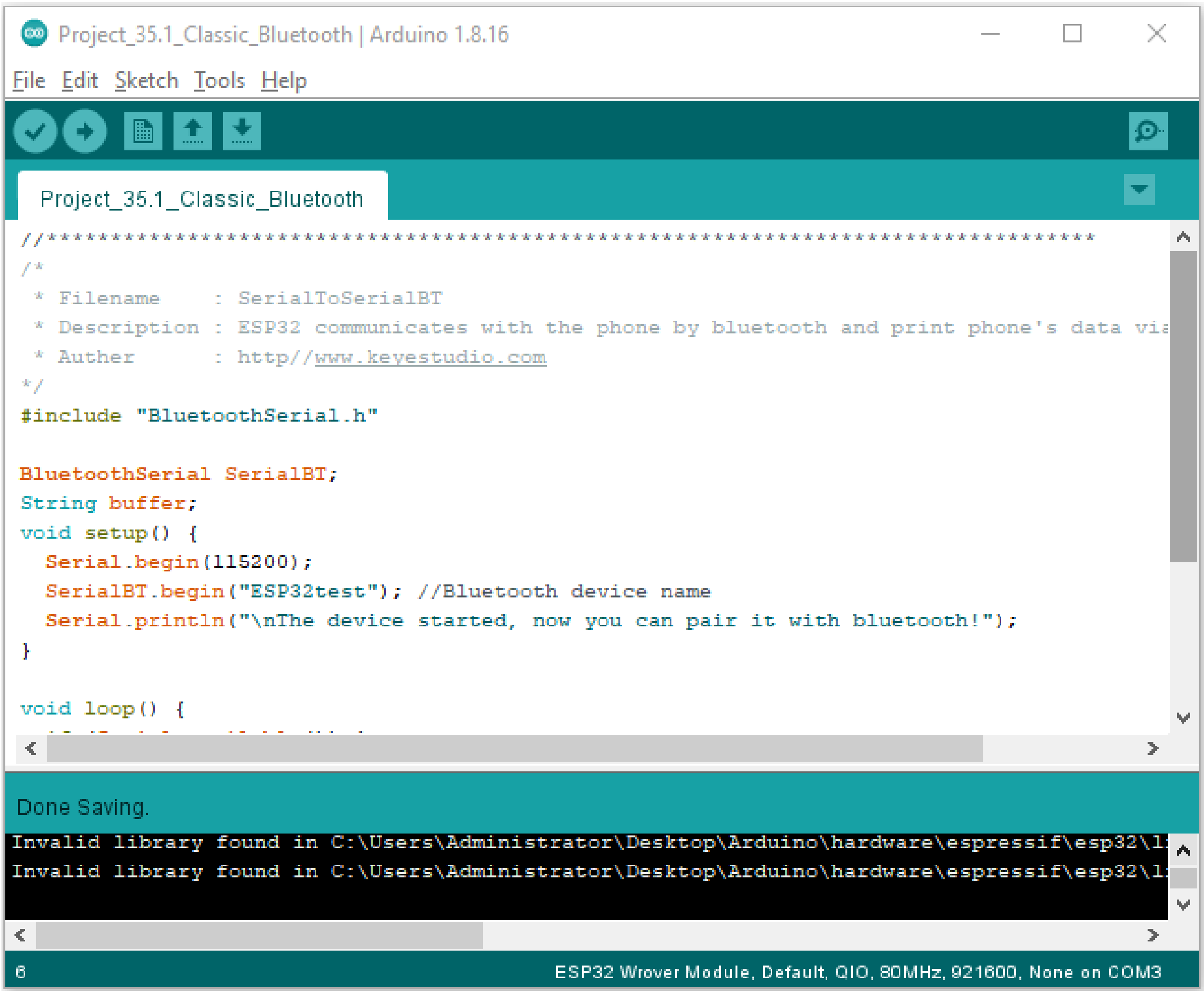
We can use a USB cable to connect ESP32 mainboard to the USB port on a computer.



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: Bluetooth\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());
```

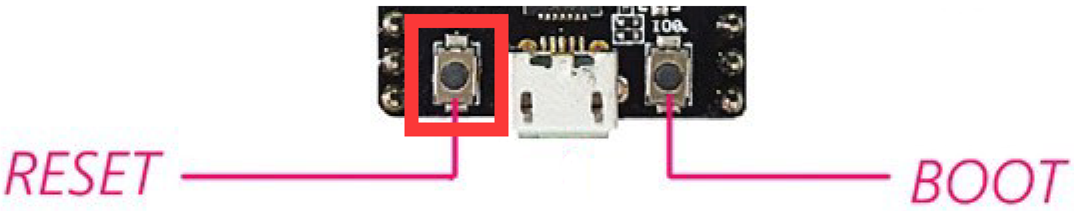
(continues on next page)

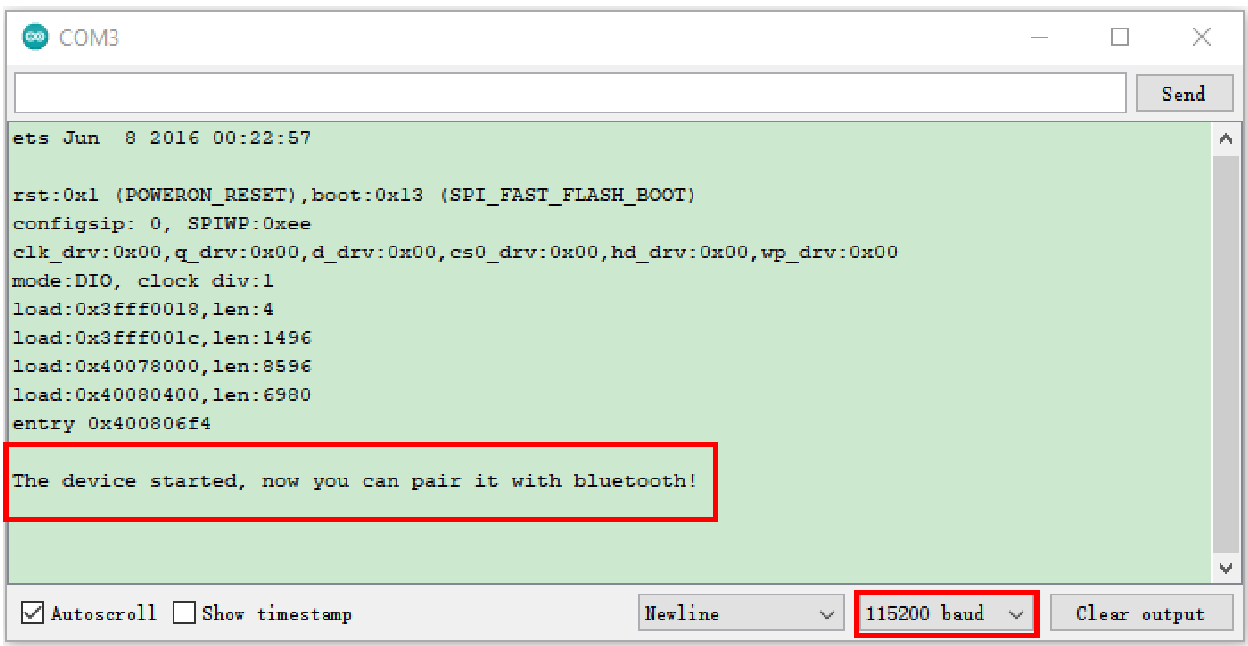
(continued from previous page)

```
  }
  if (SerialBT.available()) {
    Serial.write(SerialBT.read());
  }
  delay(20);
}
//**********************************************************************************
```

6. Project result:

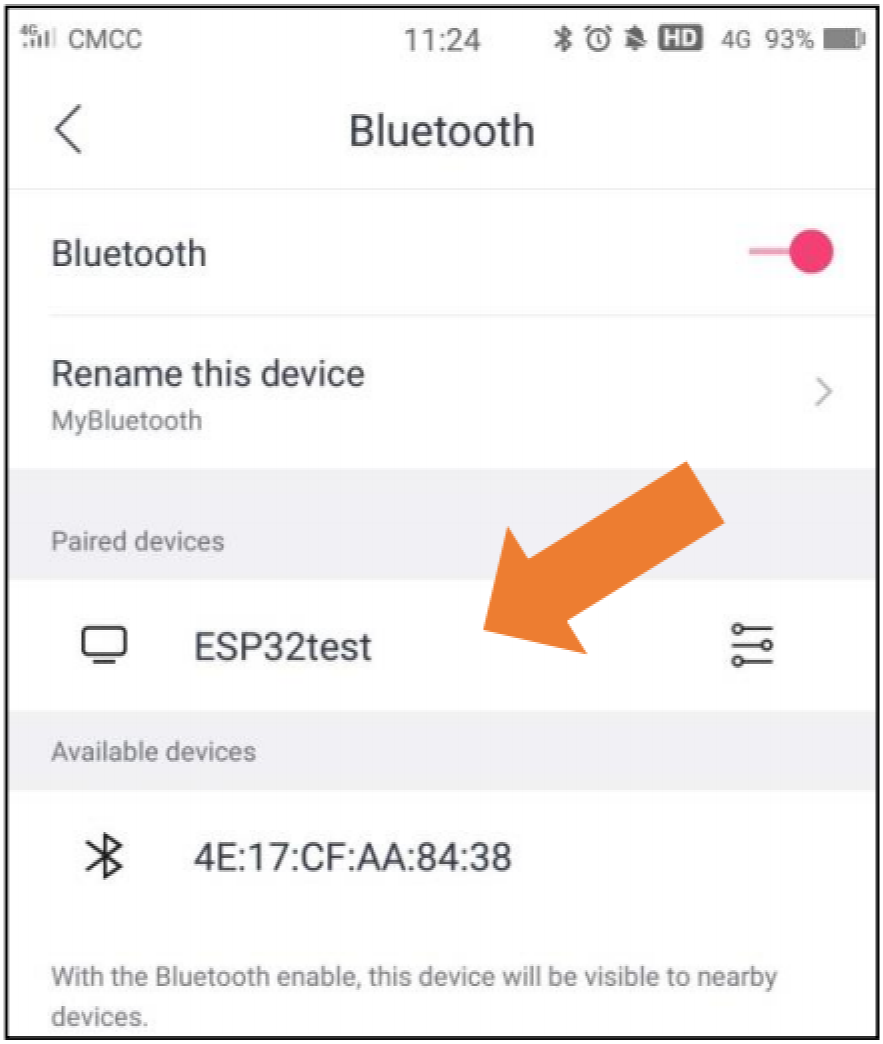
Compile and upload the code to the ESP32. After uploading successfullywe will use a USB cable to power on. Open the serial monitor and set the baud rate to **115200**. When you see the serial monitor prints out the character string as below, it indicates that the Bluetooth of ESP32 is ready and waiting 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)



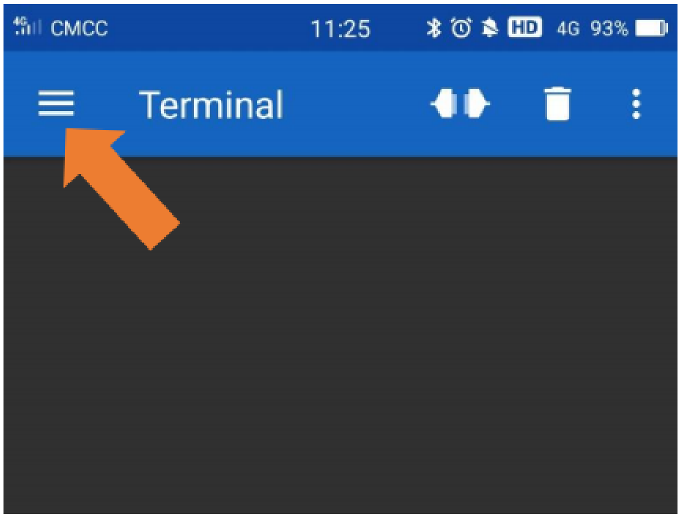


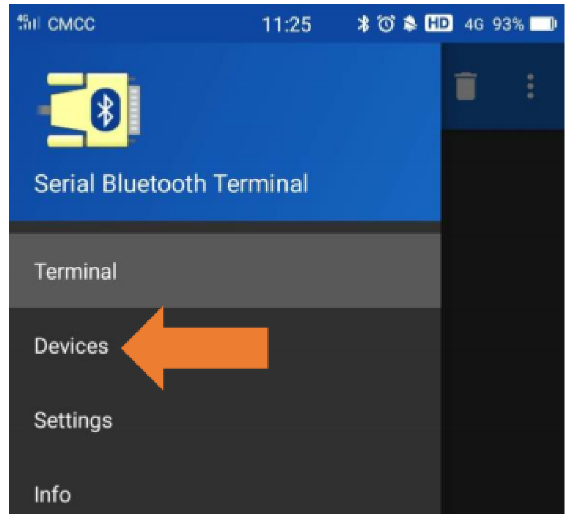
Make sure that the Bluetooth of your phone has been turned on and "Serial Bluetooth Terminal" has been installed.

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

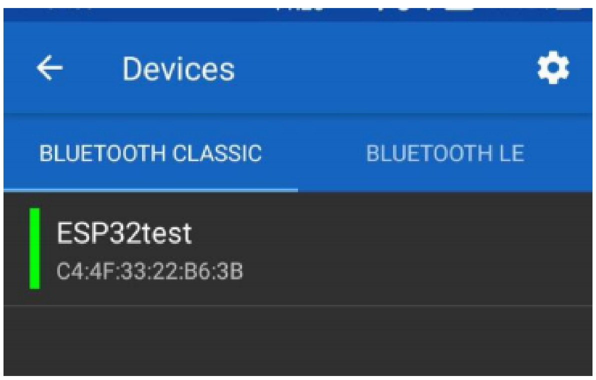


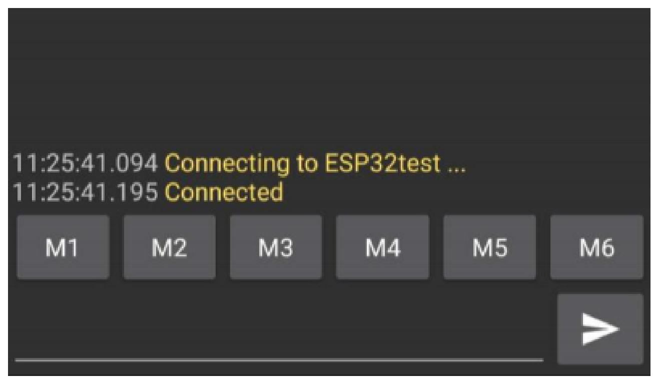
Turn on software APP, click the left of the terminal. Select "**Devices**" .





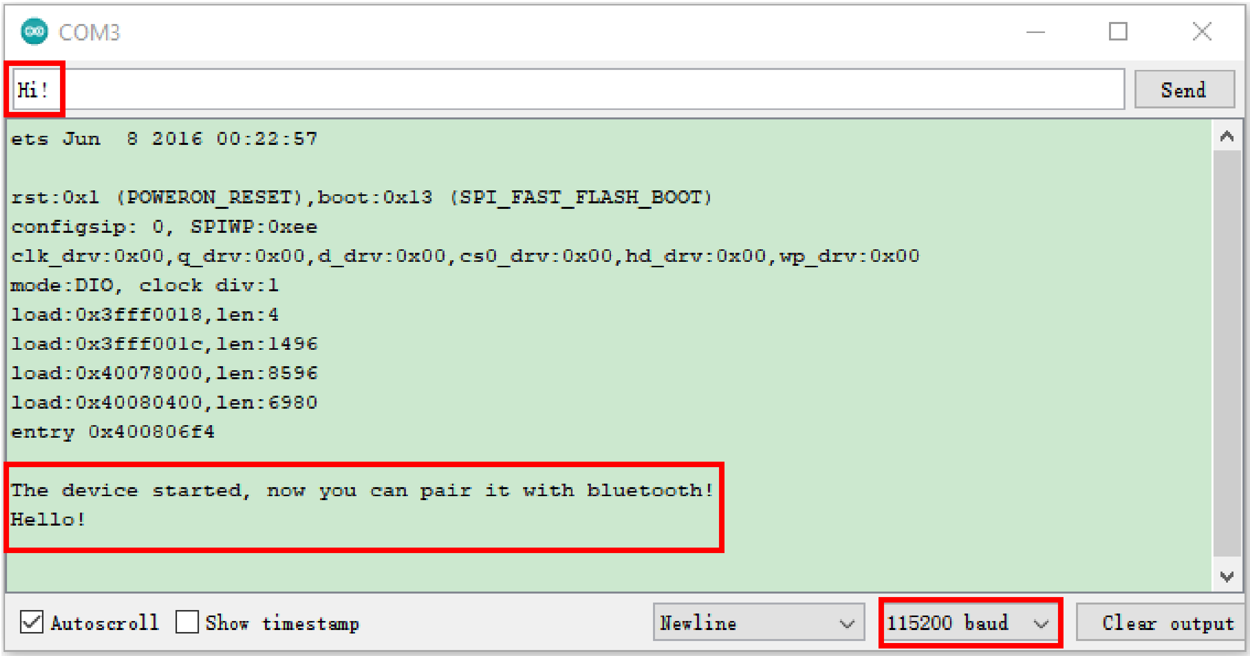
Select ESP32test in classic Bluetooth mode, and a successful connecting prompt will appear as shown on the right illustration.

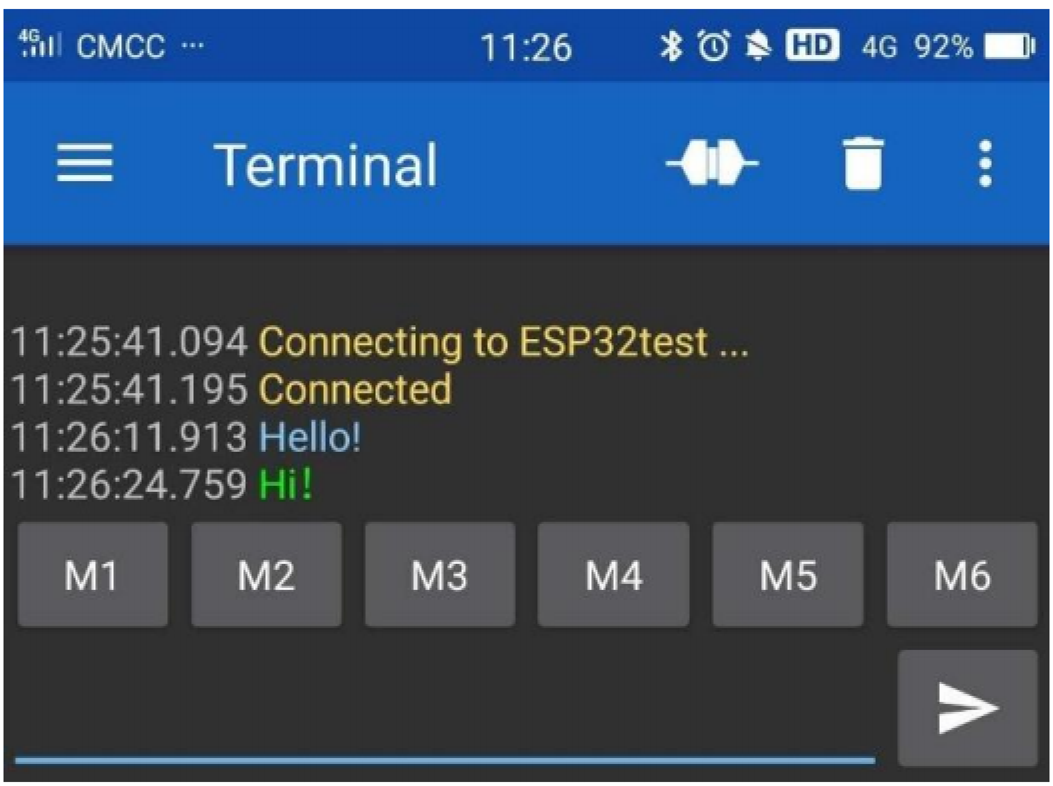




And now data can be transferred between your mobile phone and computer via ESP32.

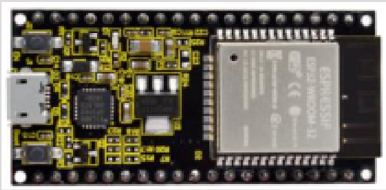
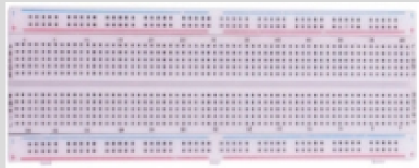
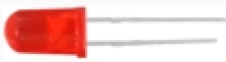
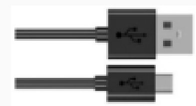
Send “**Hello!**” from your phone, when the computer receives it, reply “**Hi!**” to your phone.





Project 35.2Bluetooth Control LED

1. Components

			
ESP32*1	Breadboard*1		
			
Red LED*1	220Resistor*1	Jumper Wires	USB Cable*1

2. Wiring diagram

media/0735997593c8858ad6441d8e9867206f.png

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 35Bluetooth\Project_35.2_Bluetooth_Control_LED"**.


```
//*************************************************************************************
/*
 * 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);
  Serial.println("\nThe device started, now you can pair it with bluetooth!");
}
```

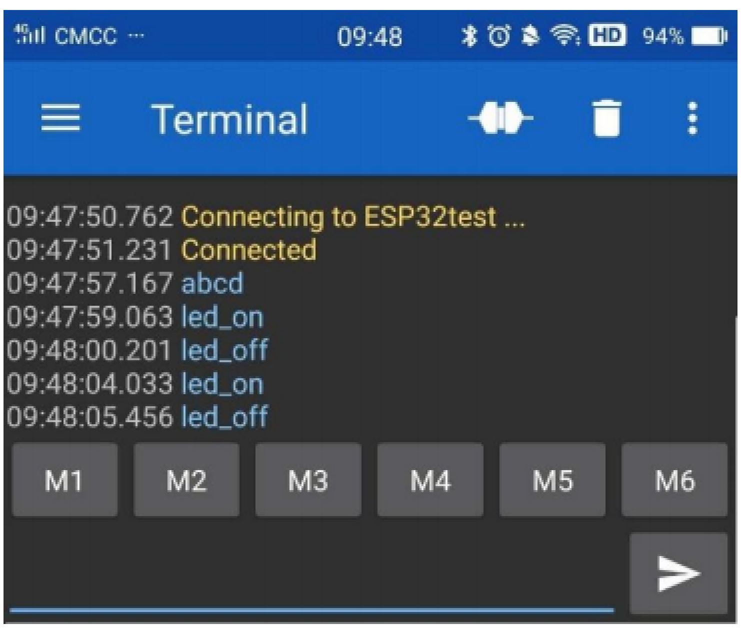
(continues on next page)

(continued from previous page)

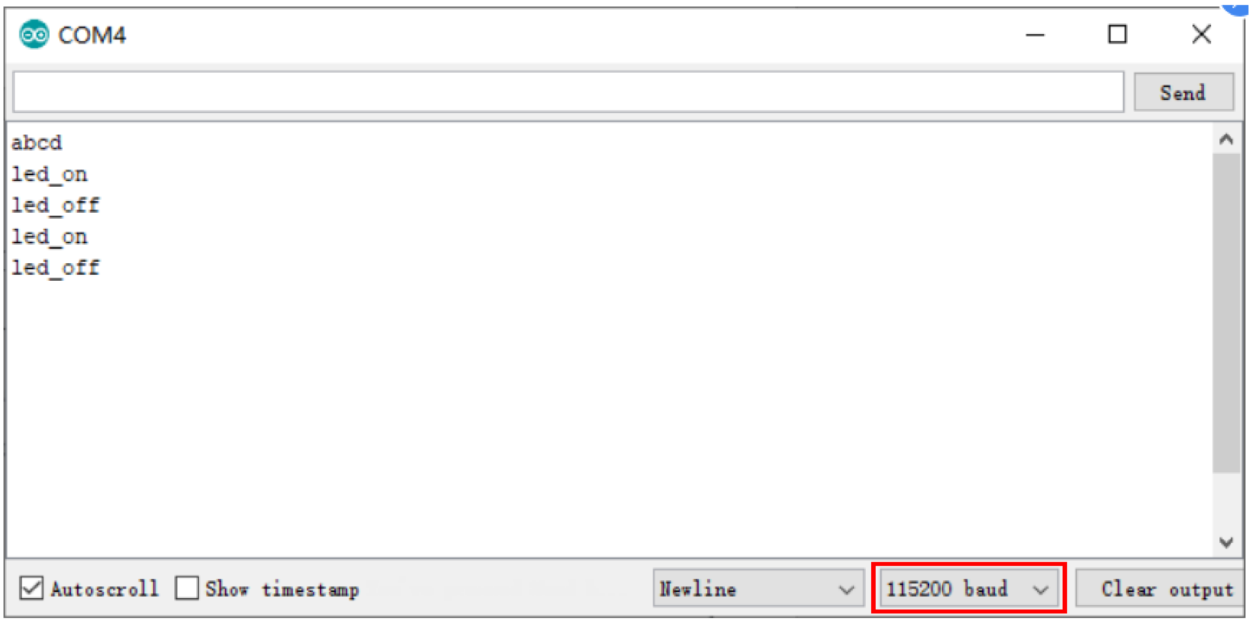
```
void loop() {
  while(SerialBT.available())
  {
    buffer[count] = SerialBT.read();
    count++;
  }
  if(count>0){
    Serial.print(buffer);
    if(strncmp(buffer,"led_on",6)==0){
      digitalWrite(LED,HIGH);
    }
    if(strncmp(buffer,"led_off",7)==0){
      digitalWrite(LED,LOW);
    }
   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

4. Project result

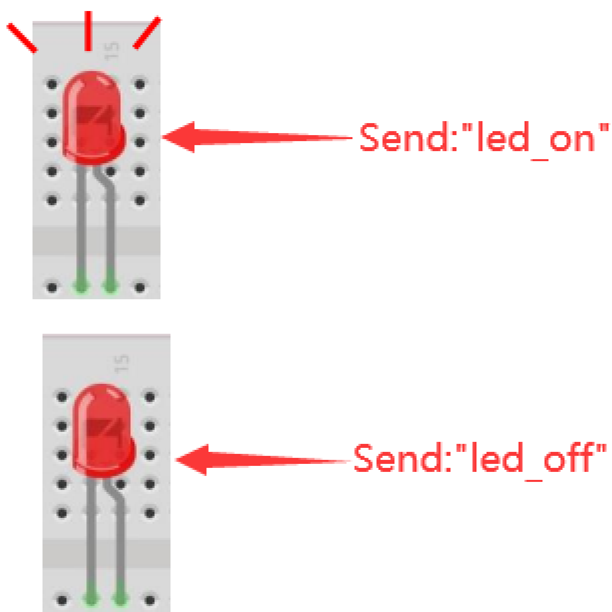
Compile and upload the code to the ESP32, after uploading successfullywe will use a USB cable to power on. The operation of the APP is the same as Project 35.1, you only need to change the sending content to “LED on”and “LED off” to operate LEDs on the ESP32. Data sent from mobile APP:



Display on the serial port of the computer:



The phenomenon of LED



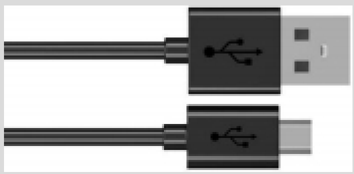
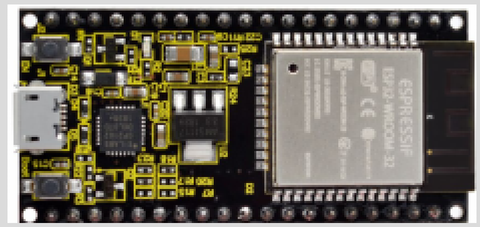
Attention: If the sending content isn't "led-on"or "led-off", then the state of LED will not change. If the LED is on, when receiving irrelevant content, it keeps on; Correspondingly, if the LED is off, when receiving irrelevant content, it keeps off.

5.37 Project 36WiFi Station Mode

1. Introduction

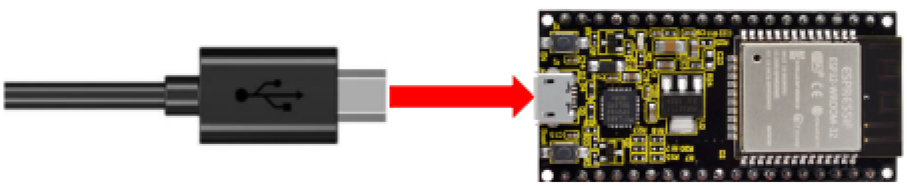
ESP32 has three different WiFi operating modes : Station modeAP mode and AP+Station mode. All WiFi programming projects must be configured with WiFi operating mode before using WiFi, otherwise WiFi cannot be used. In this project, we will learn about ESP32's WiFi Station mode.

2. Components

	
USB Cable*1	ESP32*1

3. Wiring

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



4. Component knowledge

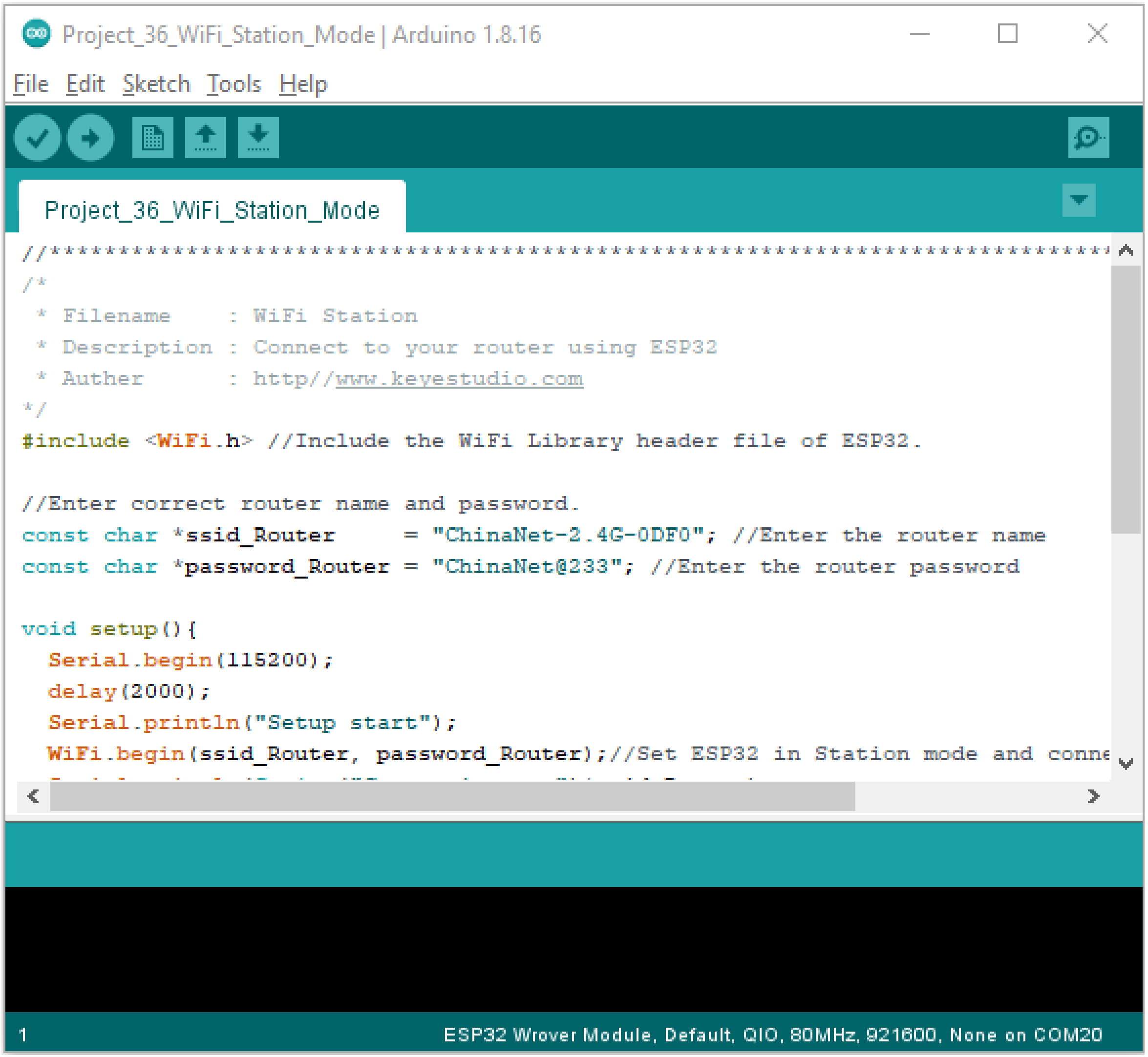
Station mode: When ESP32 selects Station mode, it acts as a WiFi client. It can connect to the router network and communicate with other devices on the router via WiFi connection. As shown below, the PC is connected to the router, and if ESP32 wants to communicate with the PC, it needs to be connected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

media/f74baff97695aa2ee33a8c19370d2547.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(path:) **"Arduino-Codes\Project 36WiFi Station Mode\Project_36_WiFi_Station_Mode"**.



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.


```
//**********************************************************************************
/*
 * Filename    : WiFi Station
 * Description : Connect to your router using ESP32
 * Auther      : http//www.keyestudio.com
*/
##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␣
```

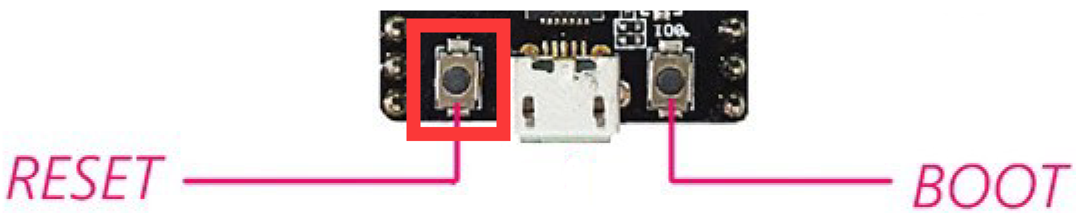
(continues on next page)

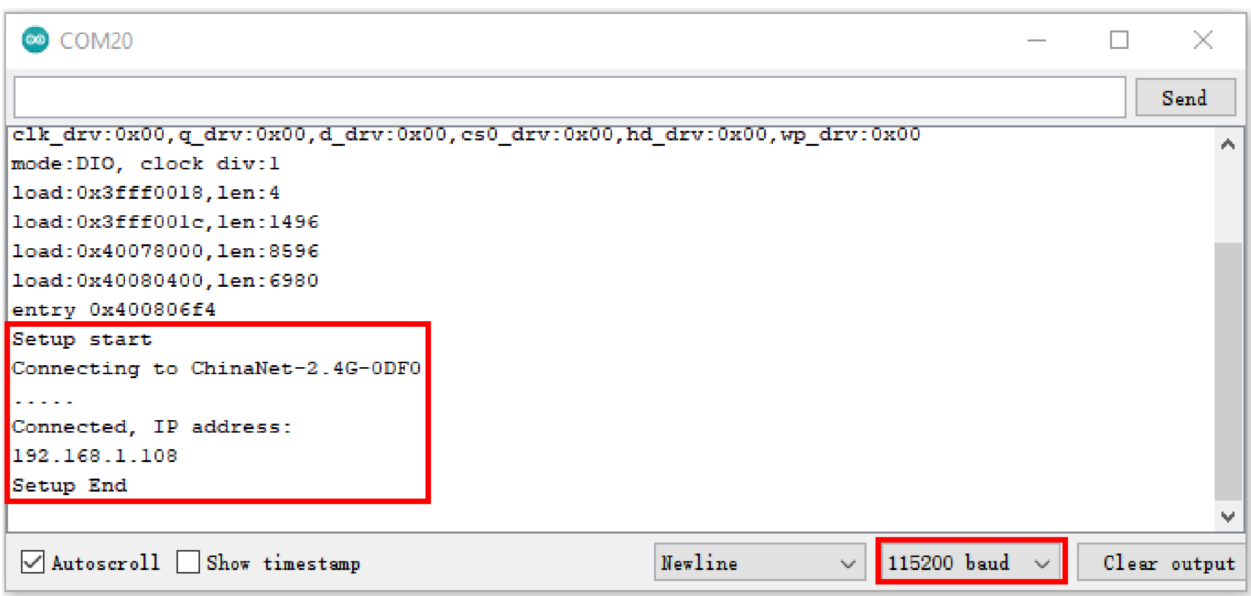
(continued from previous page)

```
↪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)



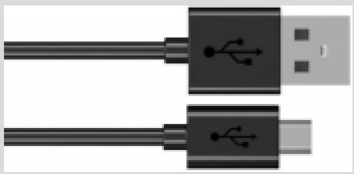
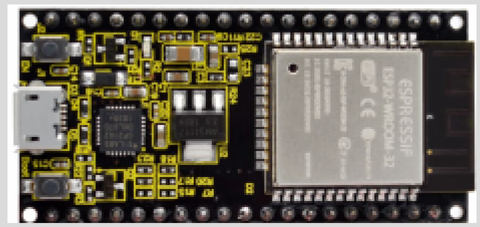


5.38 Project 37WiFi AP Mode

1. Introduction

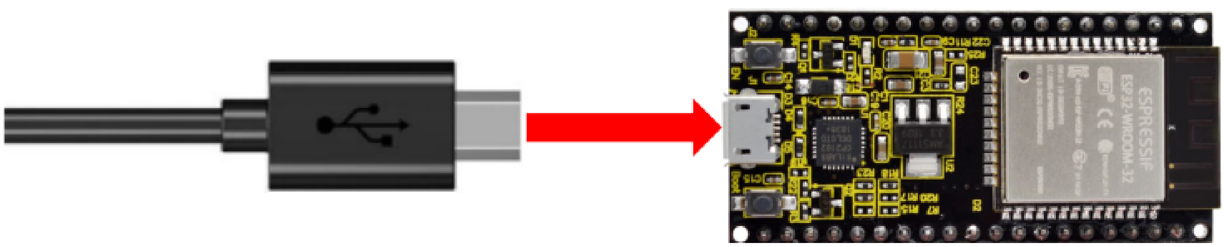
ESP32 has three different WiFi operating modes : Station modeAP mode and AP+Station mode. All WiFi programming projects must be configured with WiFi operating mode before using WiFi, otherwise WiFi cannot be used. In this project, we will learn about ESP32's WiFi AP mode.

2. Components

	
USB Cable*1	ESP32*1

3. Wiring

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



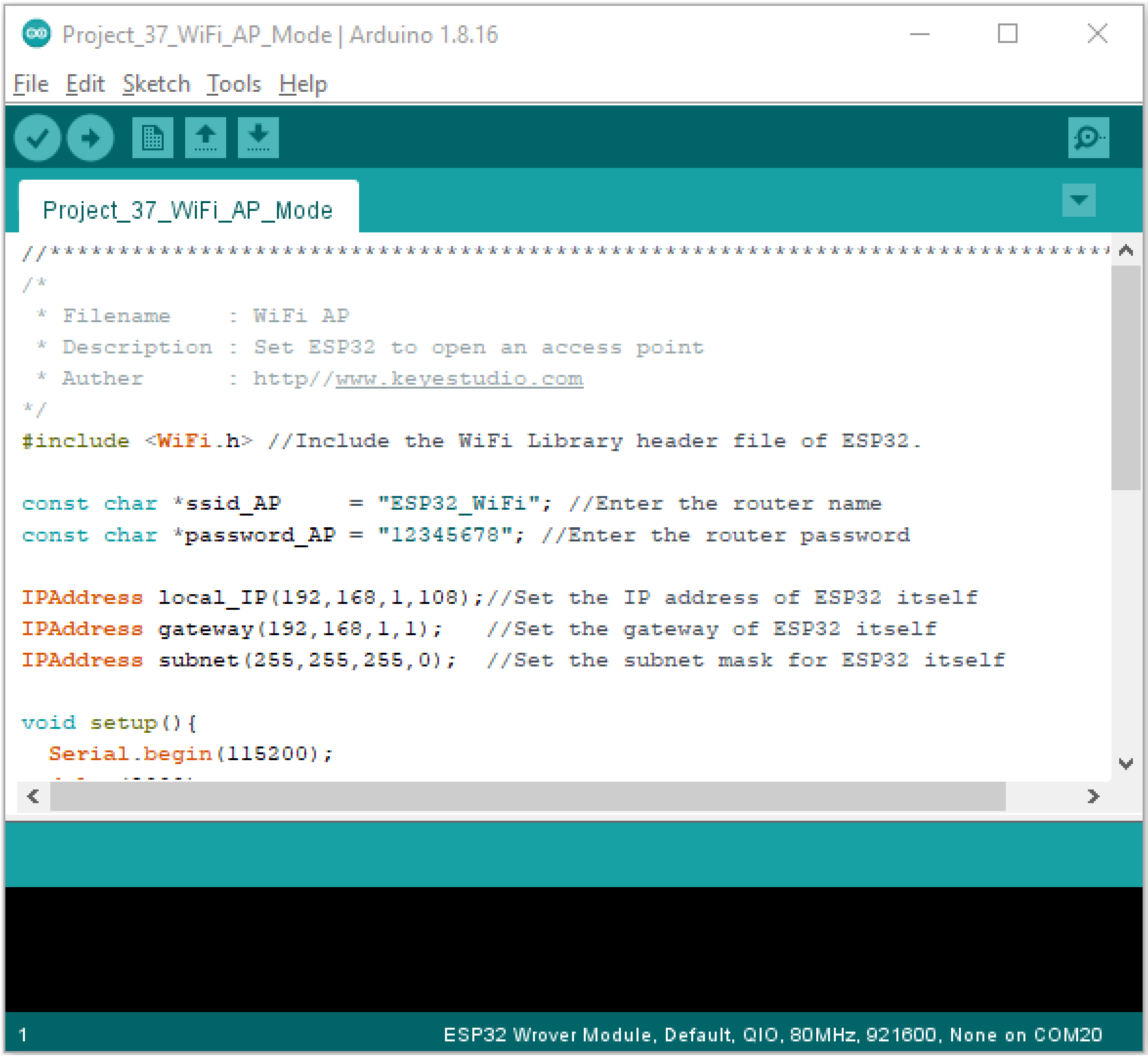
4. Component knowledge

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"**.



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.


```
//**************************************************************************
/*
 * 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)

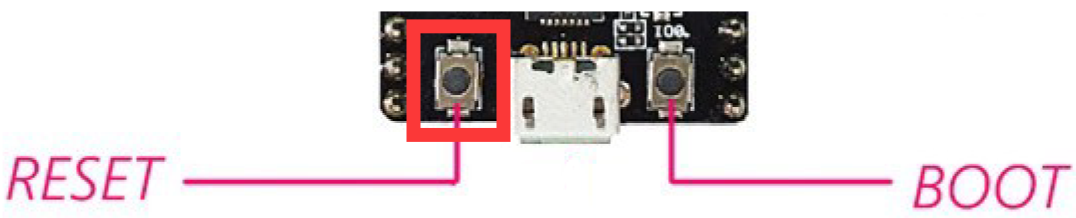
(continued from previous page)

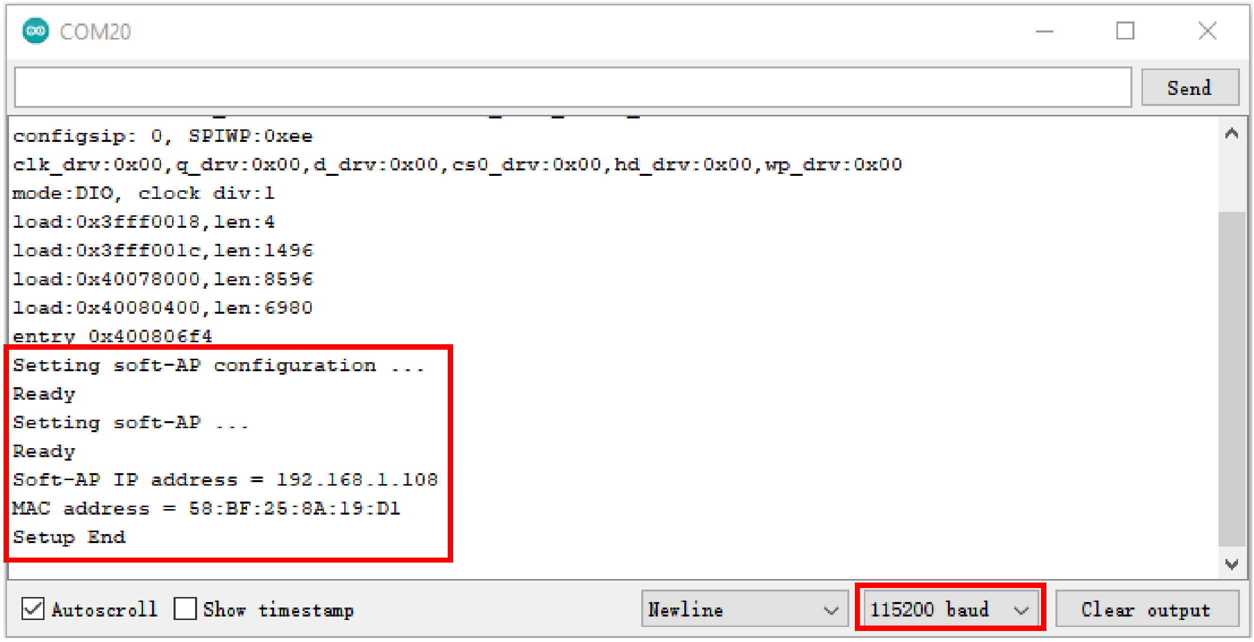
```
  Serial.println("Setting soft-AP configuration ... ");
  WiFi.disconnect();
  WiFi.mode(WIFI_AP);
  Serial.println(WiFi.softAPConfig(local_IP, gateway, subnet) ? "Ready" : "Failed!");
  Serial.println("Setting soft-AP ... ");
  boolean result = WiFi.softAP(ssid_AP, password_AP);
  if(result){
    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

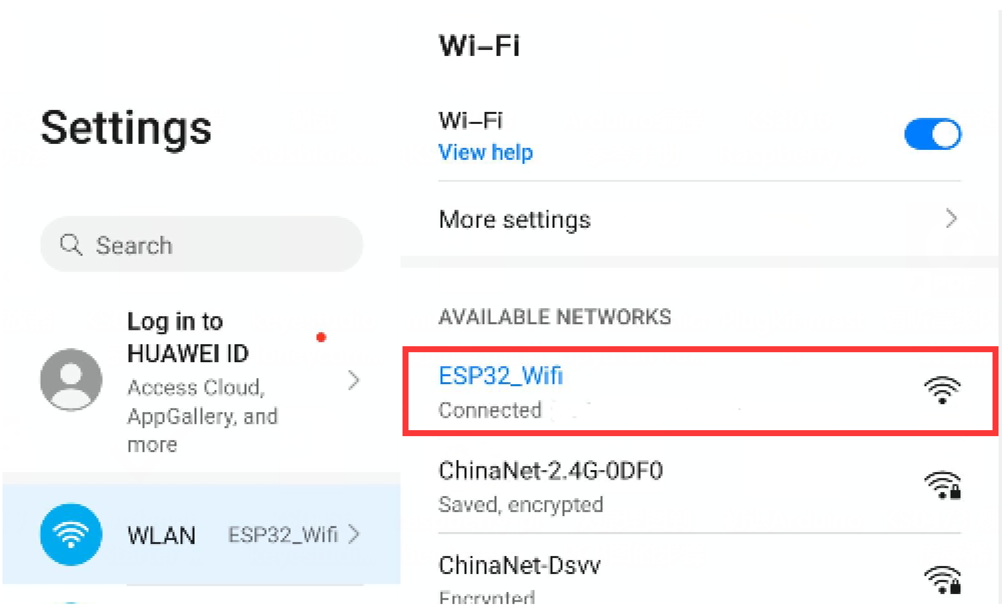
Enter the ESP32 AP name and password correctly, compile and upload the code to ESP32, open the serial monitor and set the baud rate to **115200**. And then it will display as follows.

(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)





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.

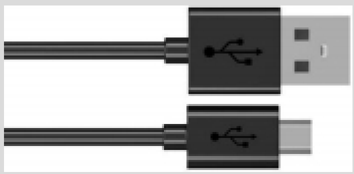
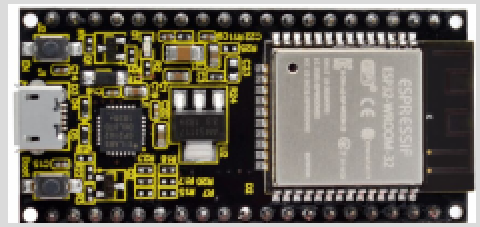


5.39 Project 38WiFi Station+AP Mode

1. Introduction

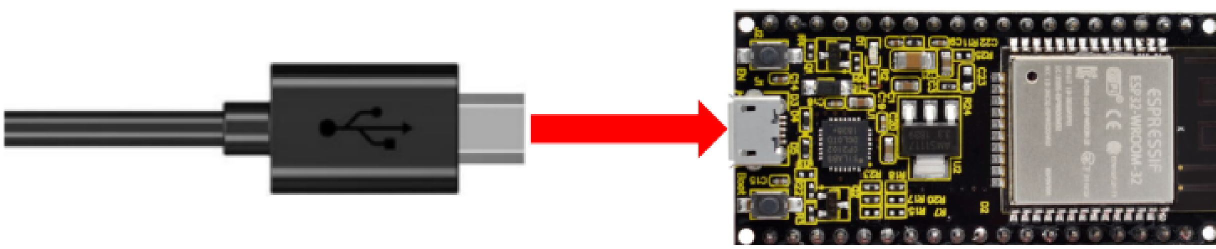
ESP32 has three different WiFi operating modes : Station modeAP mode and AP+Station mode. All WiFi programming projects must be configured with WiFi operating mode before using WiFi, otherwise WiFi cannot be used. In this project, we will learn ESP32's WiFi Station+AP mode.

2. Components

	
USB Cable*1	ESP32*1

3. Wiring

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



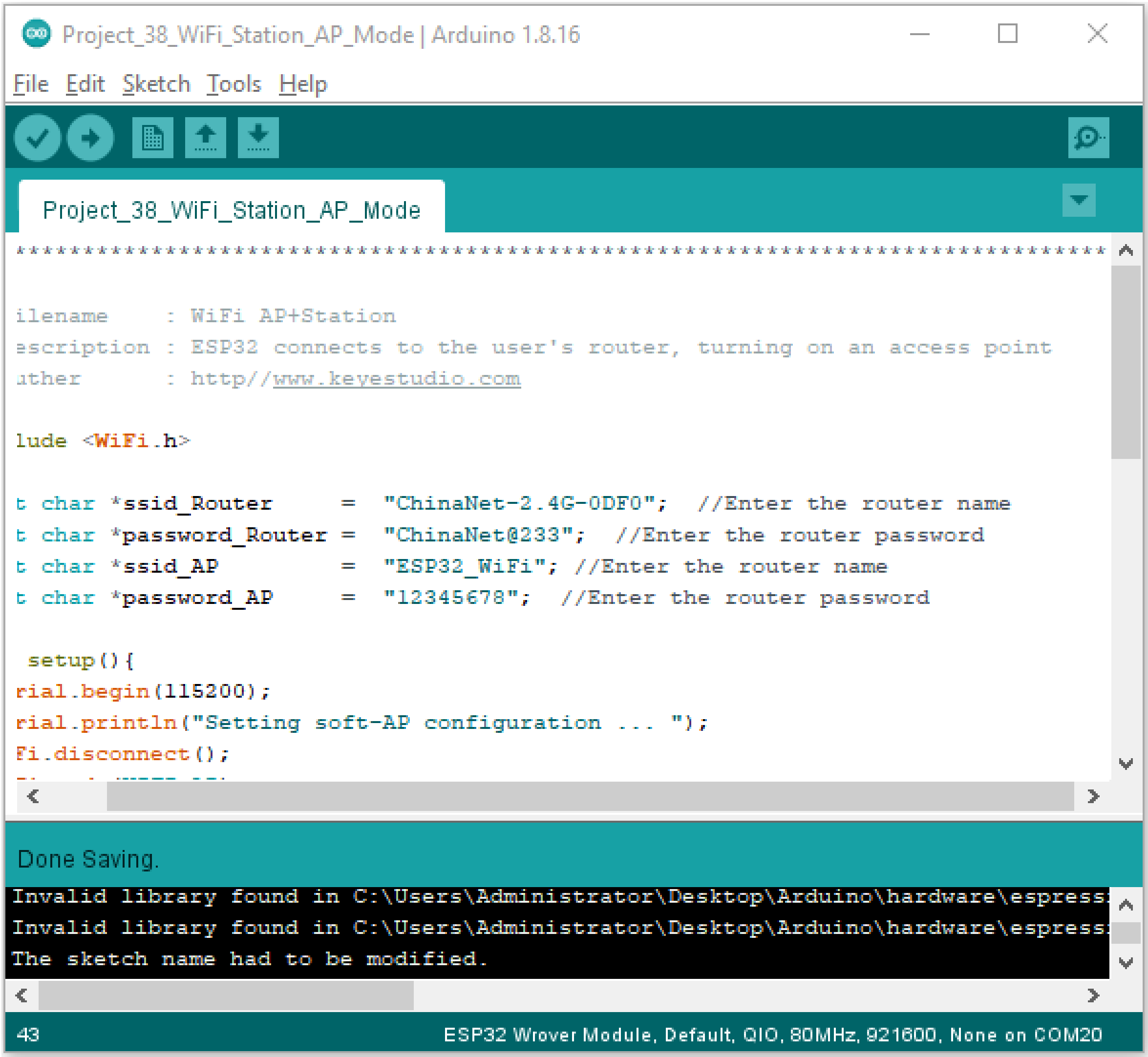
4. Component knowledge

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.

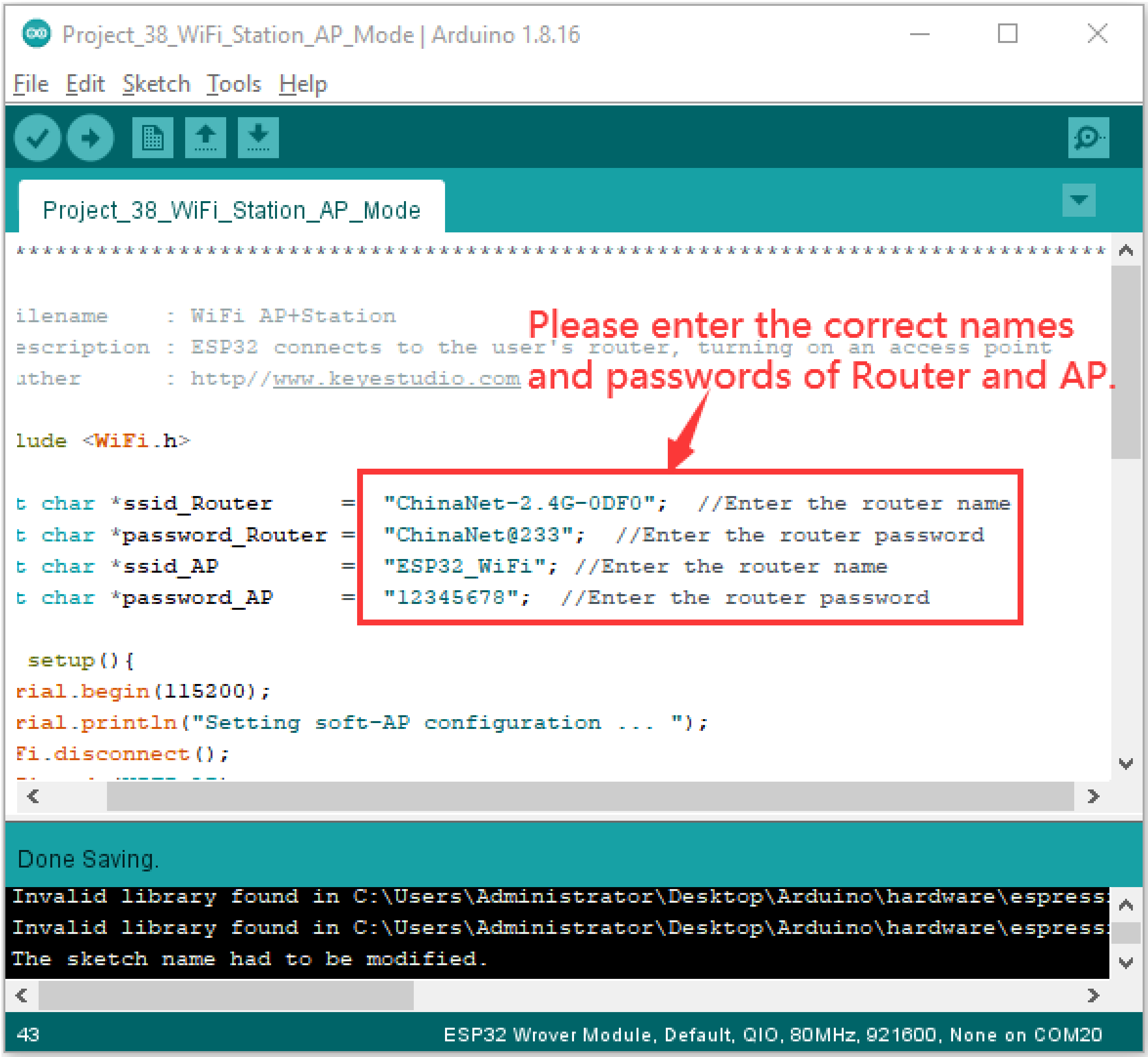
5. Project code

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"**.



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.




```
//**********************************************************************************
/*
 * 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)

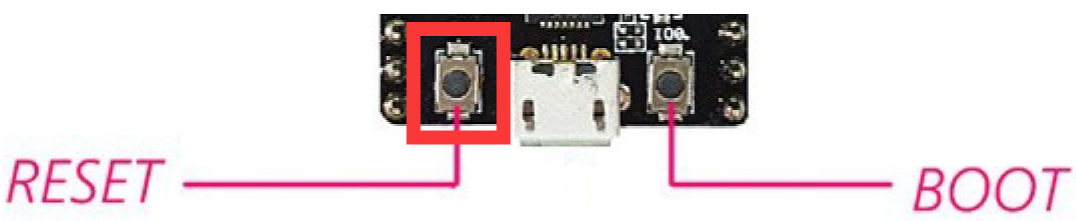
(continued from previous page)

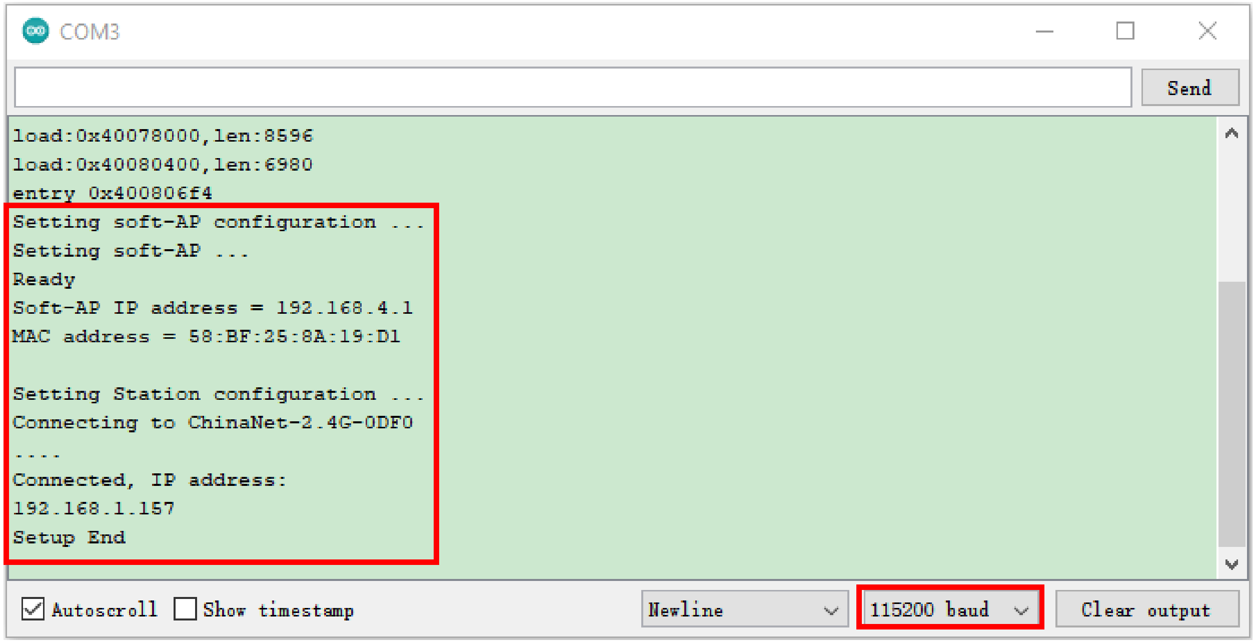
```
  Serial.println("Setting soft-AP ... ");
  boolean result = WiFi.softAP(ssid_AP, password_AP);
  if(result){
    Serial.println("Ready");
    Serial.println(String("Soft-AP IP address = ") + WiFi.softAPIP().toString());
    Serial.println(String("MAC address = ") + WiFi.softAPmacAddress().c_str());
  }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

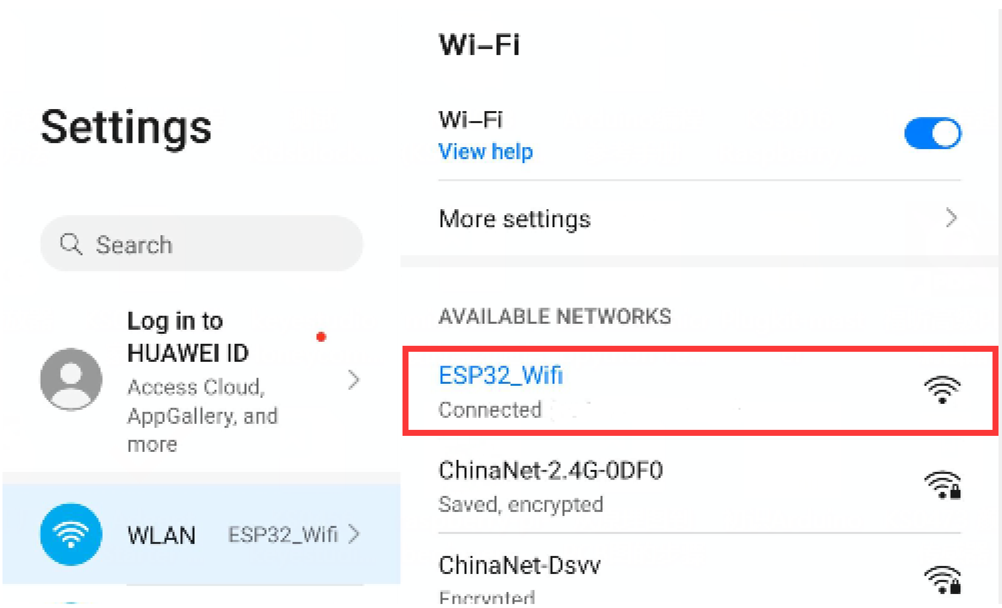
After making sure that the code is modified correctly, compile and upload the code to ESP32, open the serial monitor and set baud rate to 115200.

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)





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.

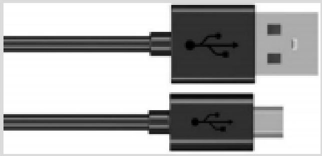
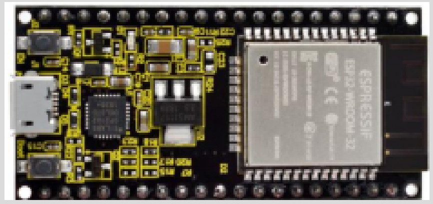


5.40 Project 39: WiFi Test

1. Introduction

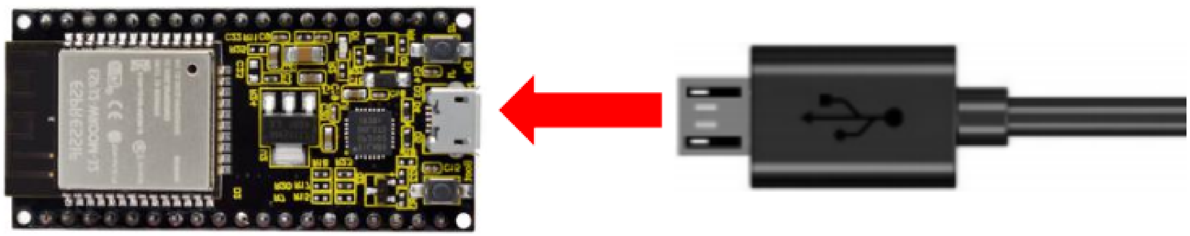
In this experiment, we first use the WiFi station mode of ESP32 to read the IP address of WiFi, and then connect WiFi through app to read the characters sent by each function button on App.

2.Components

		
USB Cable*1	ESP32*1	Smartphone/Tablet *1

3. Wiring

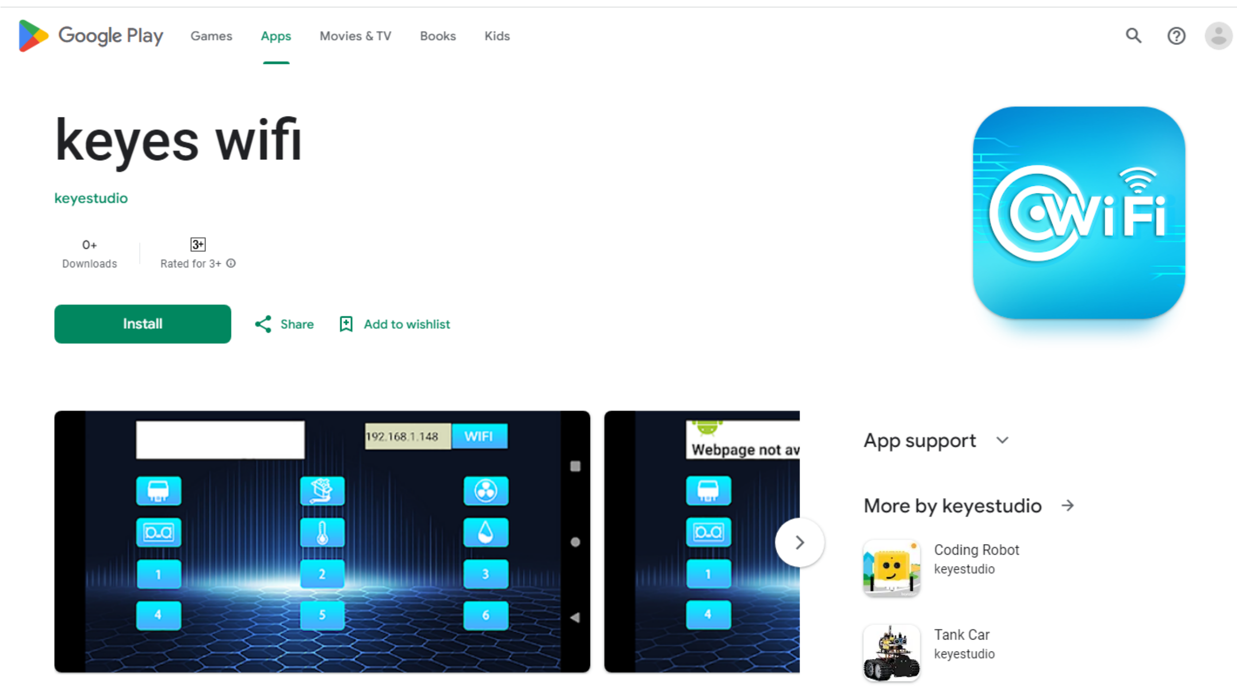
Connect the ESP32 to the USB port on your computer using a USB cable.



4. Install APP:

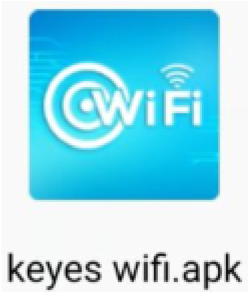
Android system (Smartphone/Tablet) APP: Go to Google Store to search for keys wifi

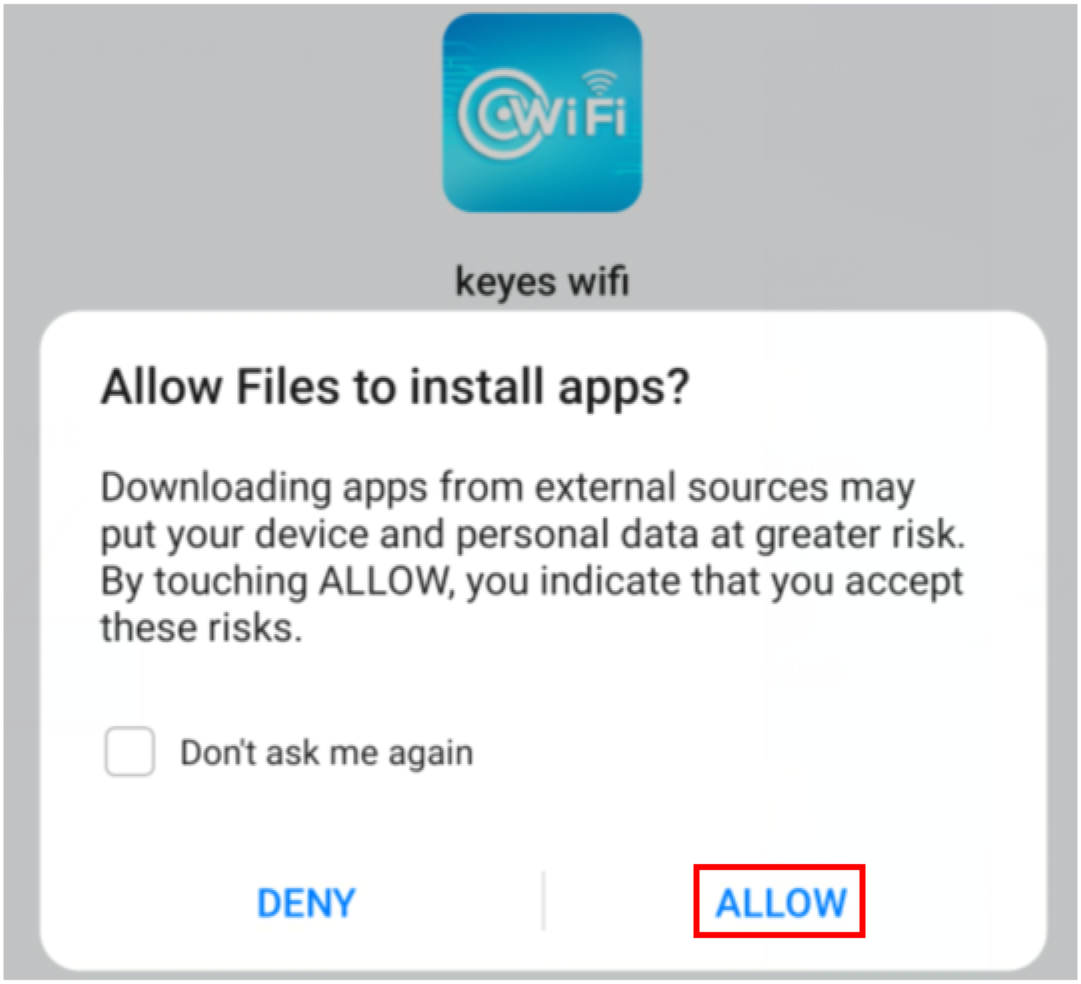
App link in Google Store:https://play.google.com/store/apps/details?id=com.keyestudio.esp8266_web_wifi2

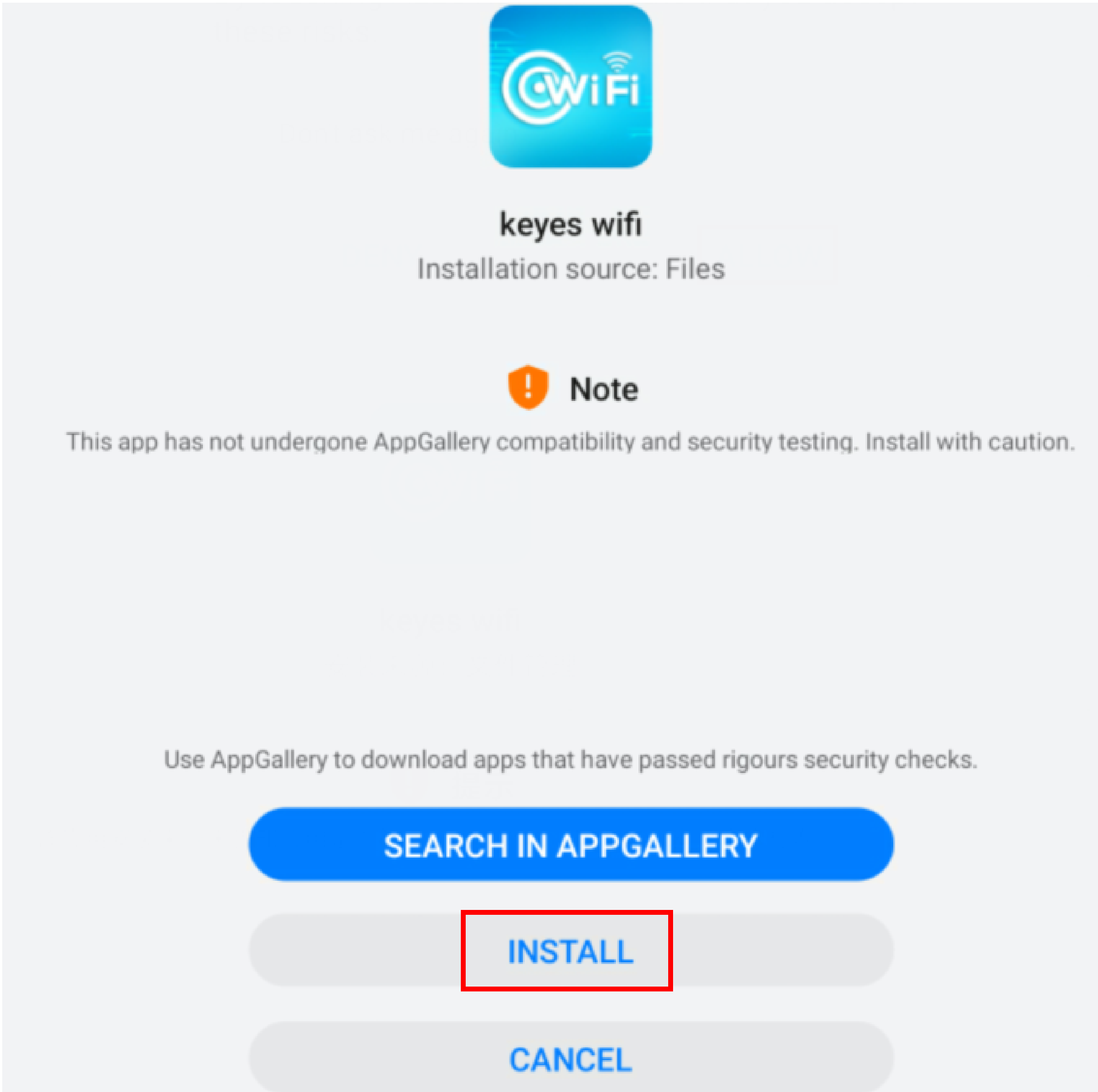


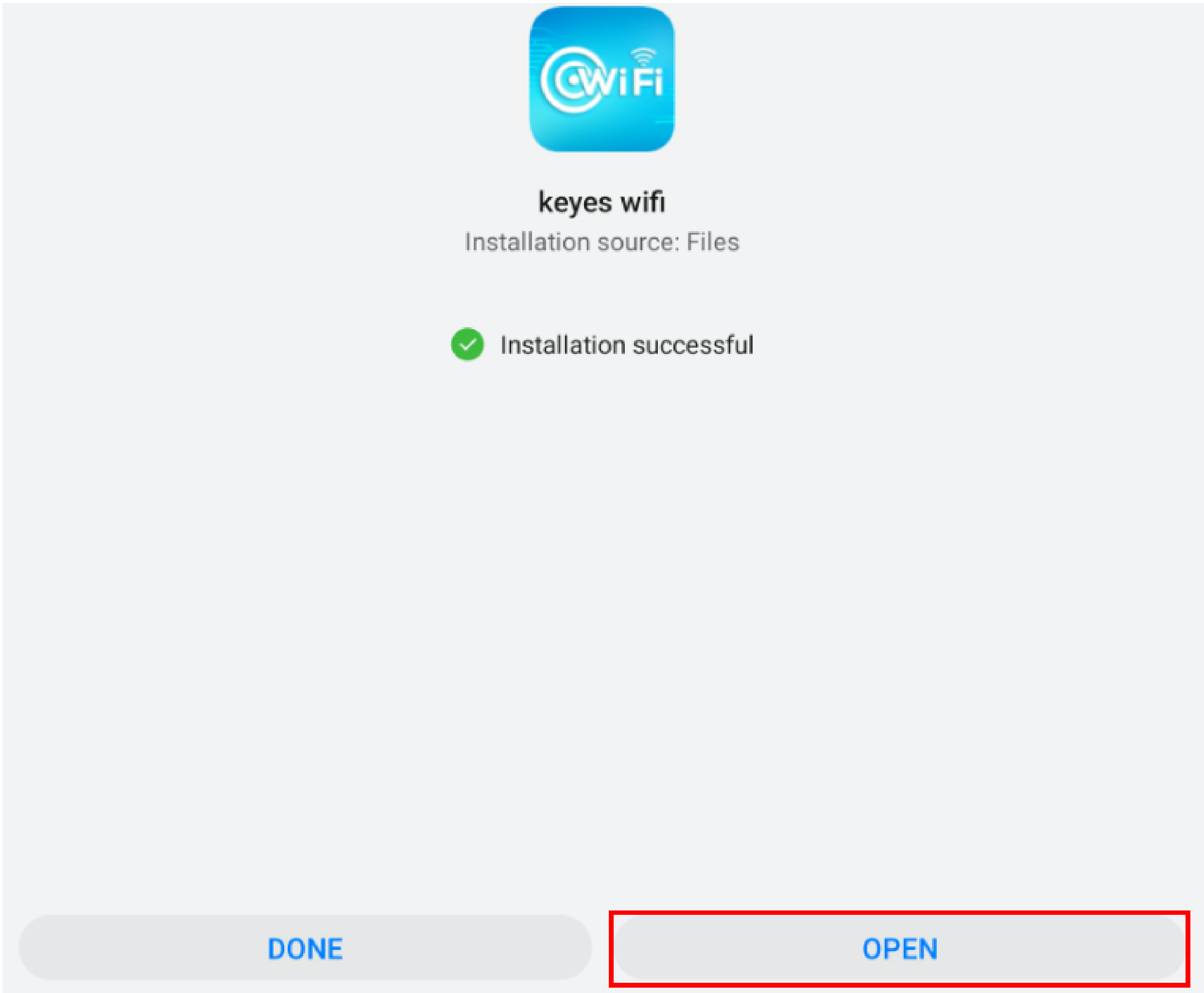
Installation steps

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.





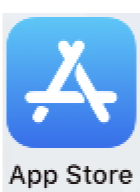

WiFi
keyes wifi
Installation source: Files
Note
This app has not undergone AppGallery compatibility and security testing. Install with caution.
Use AppGallery to download apps that have passed rigours security checks.
SEARCH IN APPGALLERY
INSTALL
CANCEL



media/78c89b91c0af2268f6267813e7923a9b.png

IOS system (Smartphone/Tablet) APP:

a.Open App Store.



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: ");
```

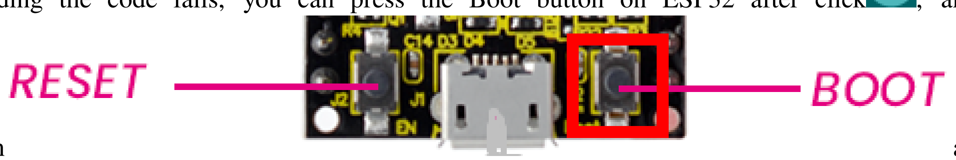
(continues on next page)

(continued from previous page)

```
        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);
    }
    //client.print(s);
    client.stop();
}
//************************************************************************************
```

Special attention: you need to change the user's Wifi name and Wifi password in the experiment code `const char* ssid = "ChinaNet-2.4G-0DF0";` `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 click, and release the Boot button 

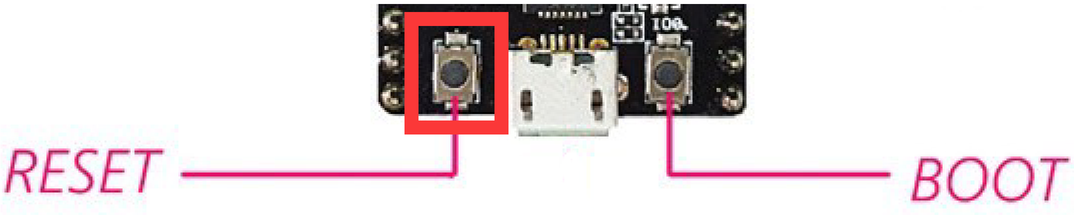
after the percentage of uploading progress appears)open the serial monitor and set baud rate to 115200.

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.

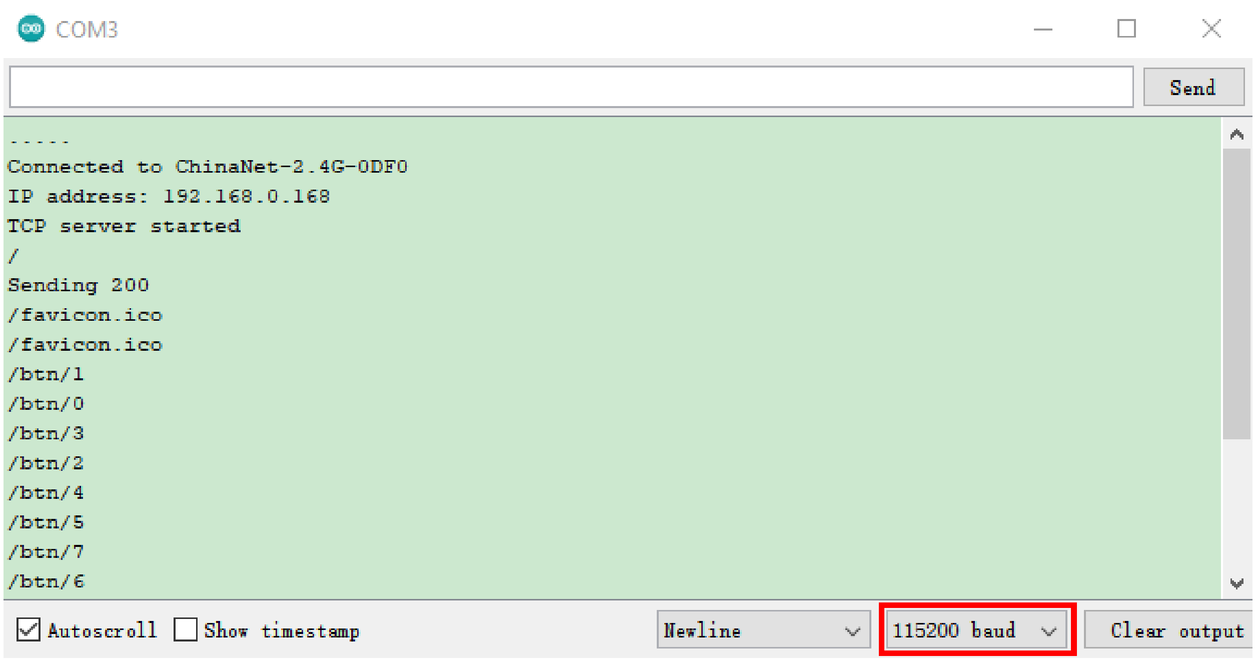
media/ee42a1bde8dd1bc4cb554ccfef1ab9ae.png

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

RESET button of the ESP32)



)

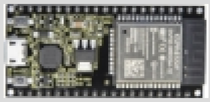
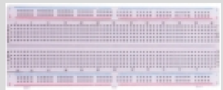
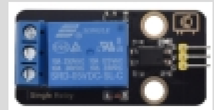
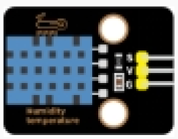
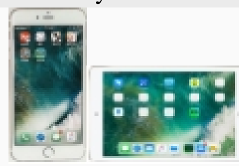
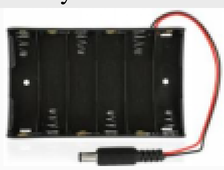
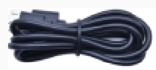
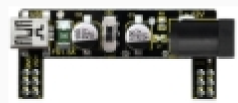
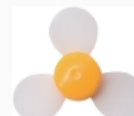


5.41 Project 40WiFi Smart Home

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

				
ESP32*1	Breadboard*1	130 Motor*1	5V Relay Module*1	Servo*1
				
Temperature and Humidity Sensor*1	HC-SR04 Ultrasonic Sensor*1	M-F Dupont Wires	Smartphone/Tablet*1	Jumper Wires
				
Battery Holder*1	Battery(self-provided)*6	USB Cable*1	Keyestudio breadboard special power module*1	Fan*1

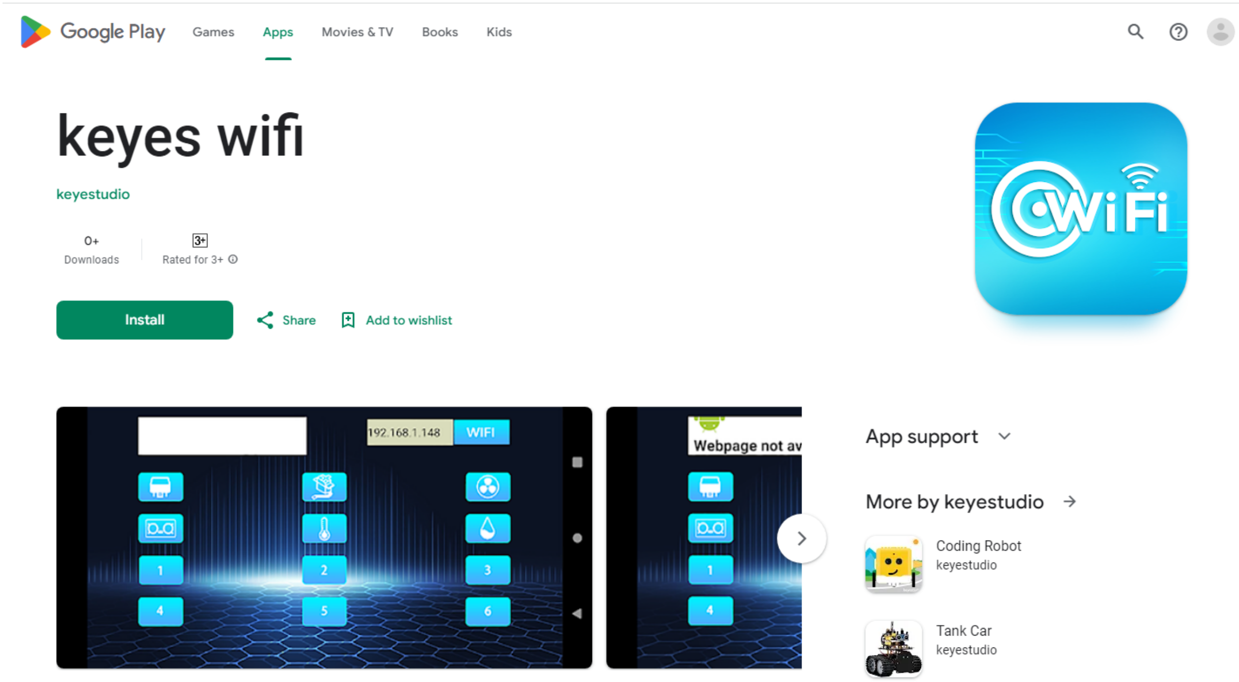
3. Wiring diagram

media/fd7a7efd4bc365e1524011b68217dee5.png

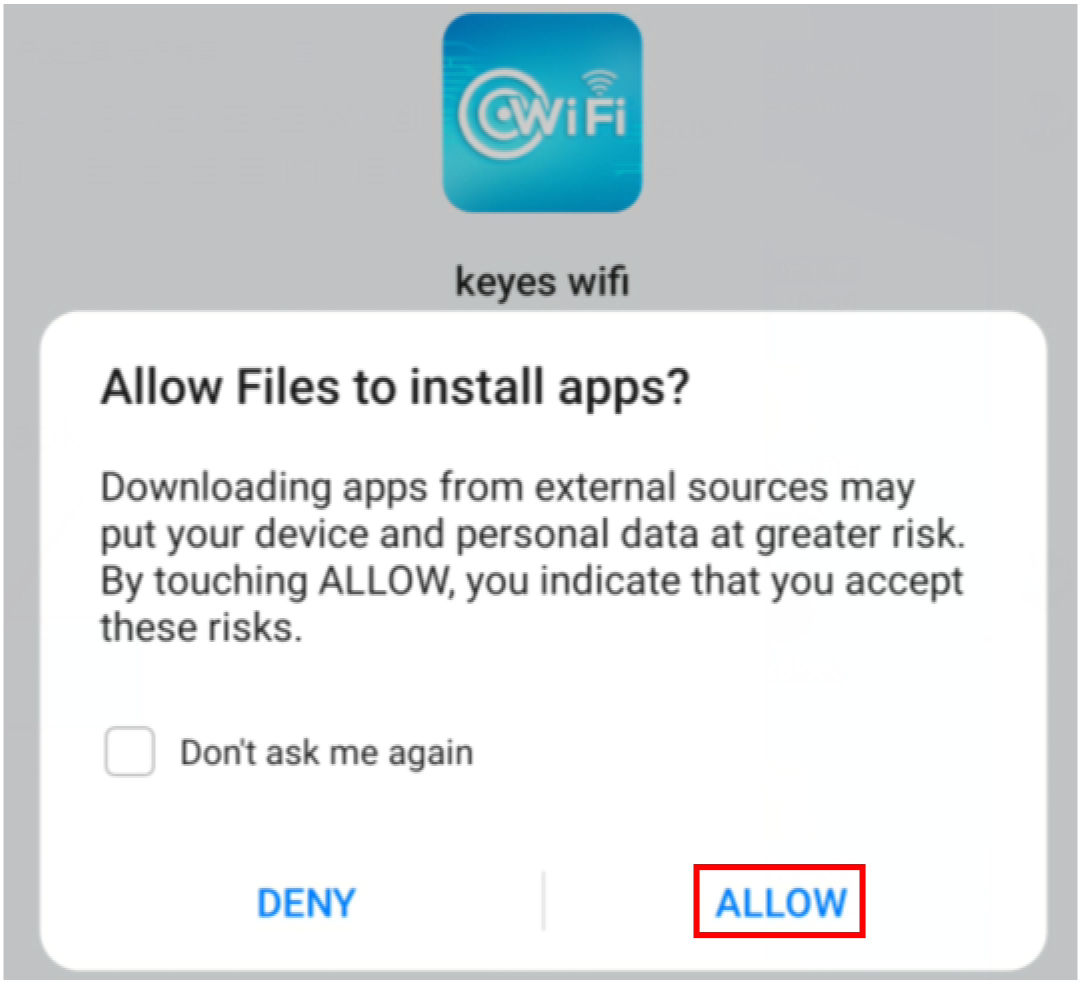
(Note: Connect the wires and then install a small fan blade on the DC motor.)

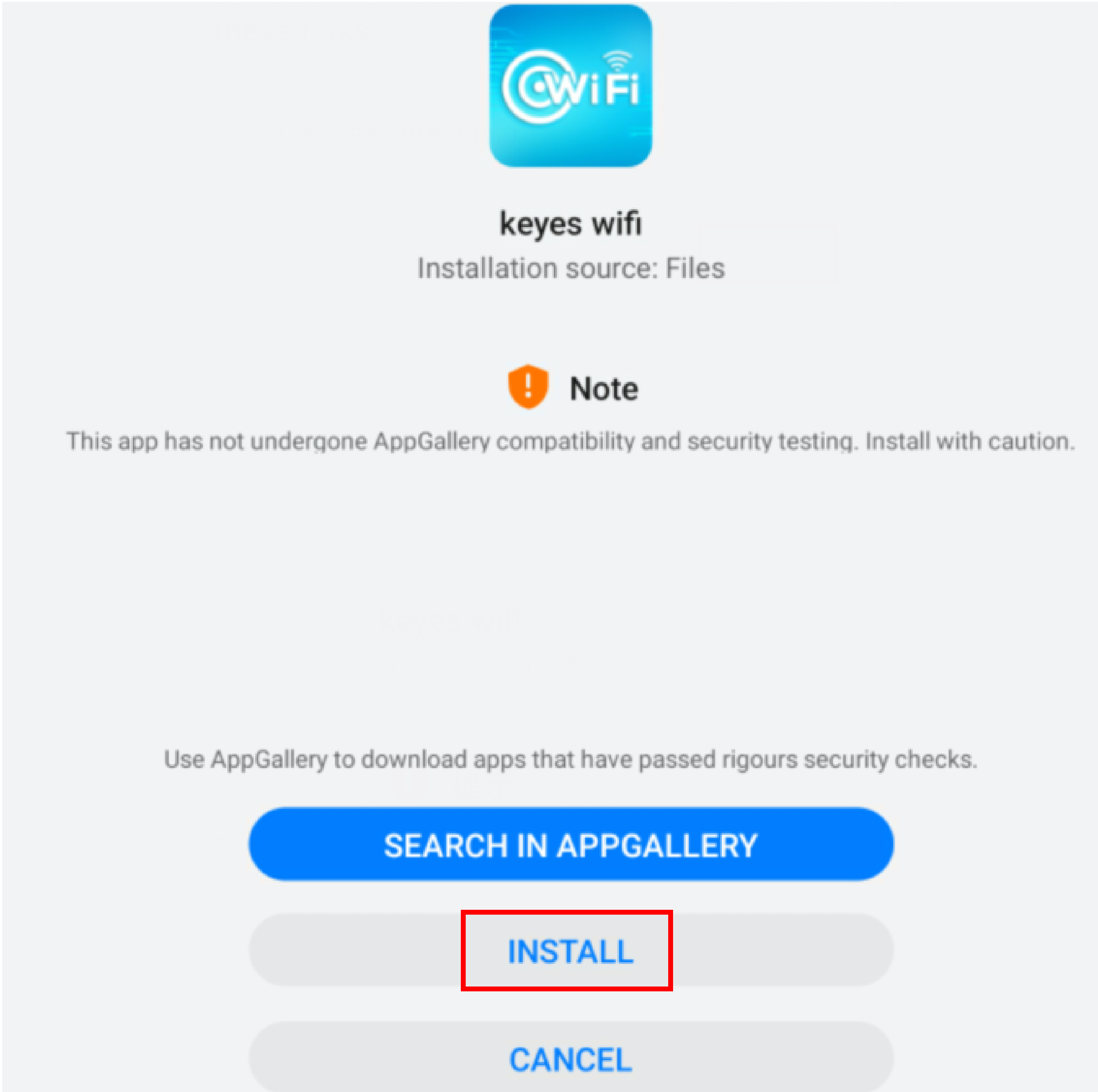
4. Install APP:

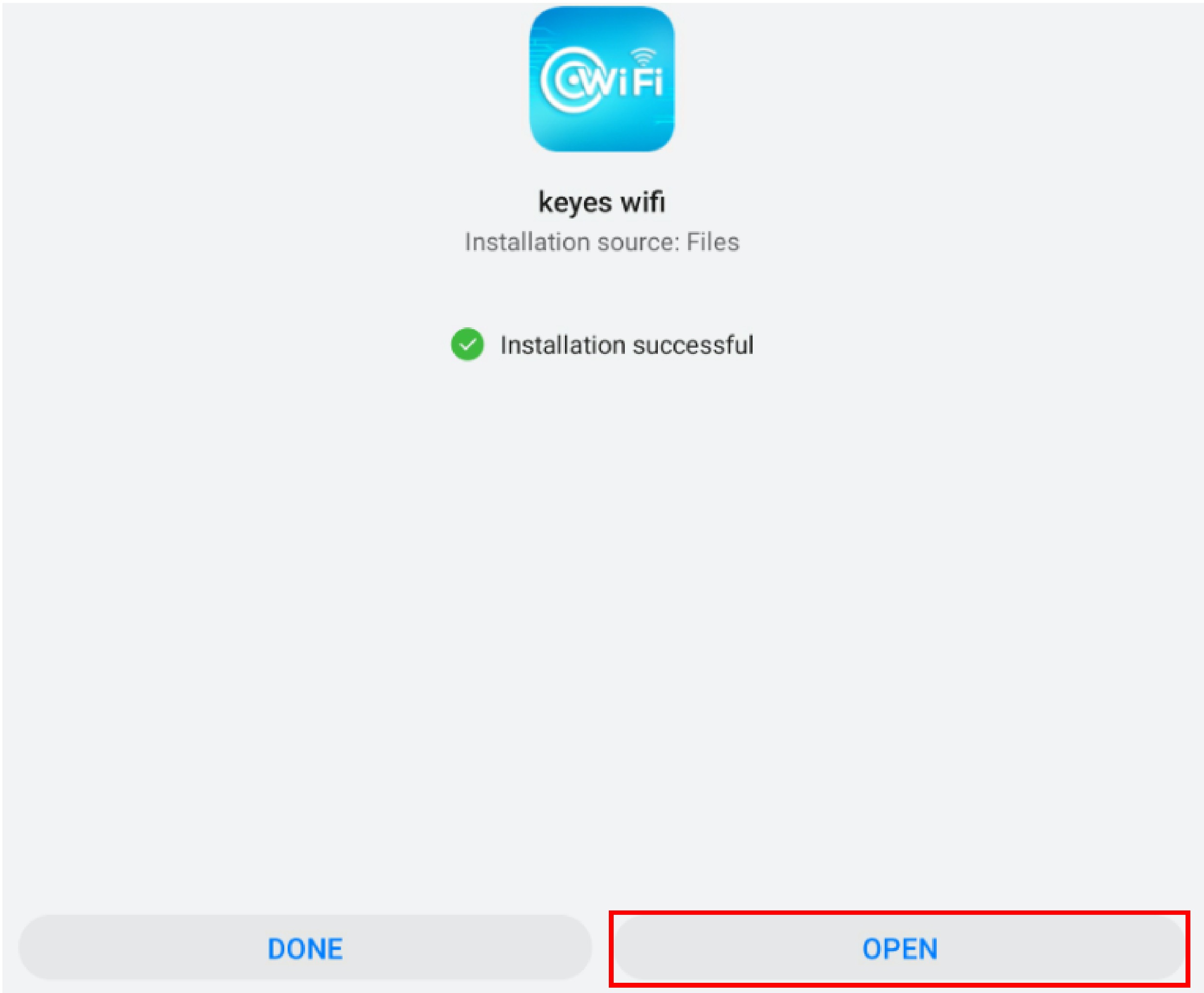
Android system (Smartphone/Tablet) APP:Go to Google Store to search for keys wifi



Installation steps


WiFi
keyes wifi
Allow Files to install apps?
Downloading apps from external sources may put your device and personal data at greater risk. By touching ALLOW, you indicate that you accept these risks.
Don't ask me again
DENY
ALLOW

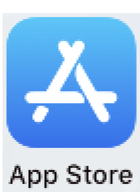

WiFi
keyes wifi
Installation source: Files
Note
This app has not undergone AppGallery compatibility and security testing. Install with caution.
Use AppGallery to download apps that have passed rigours security checks.
SEARCH IN APPGALLERY
INSTALL
CANCEL



media/78c89b91c0af2268f6267813e7923a9b.png

IOS system (Smartphone/Tablet) APP:

a.Open App Store.



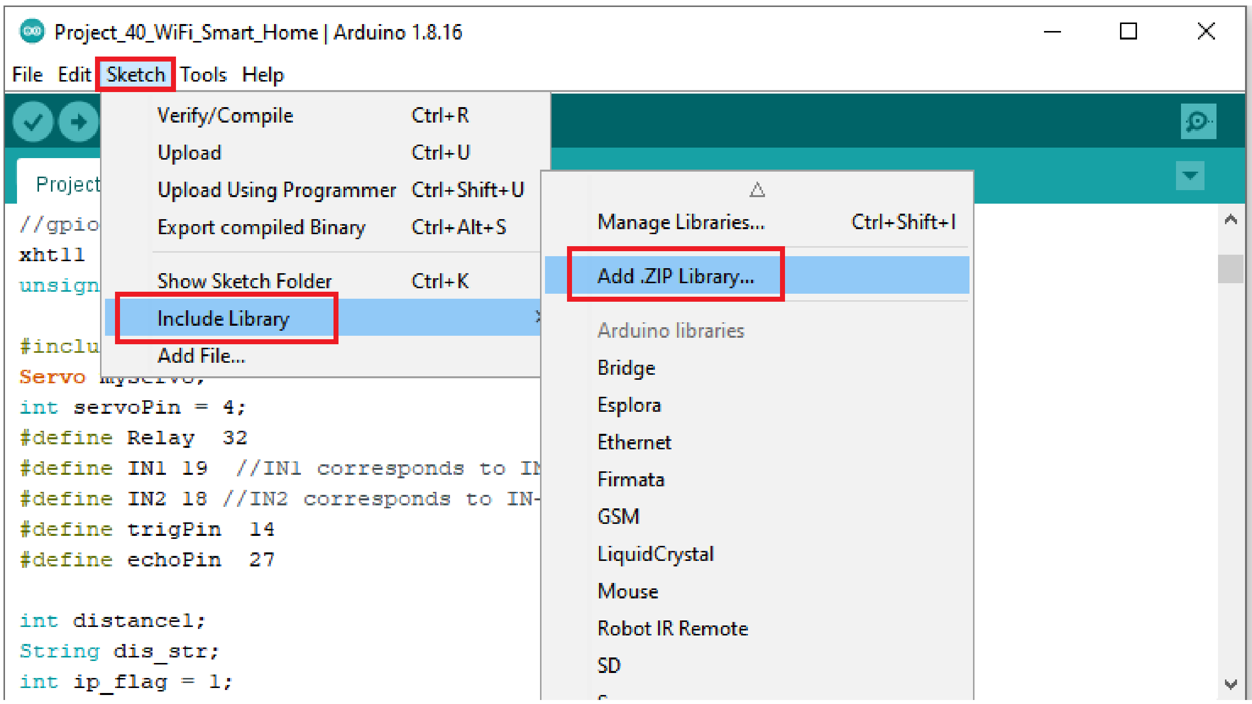
b. Enter keyes link in the search box and click Search. The download interface appears. Click“  ”to download and install the APP of keyes link. The following operations are similar to those of Android system, you can refer to the steps of Android system above for operation.

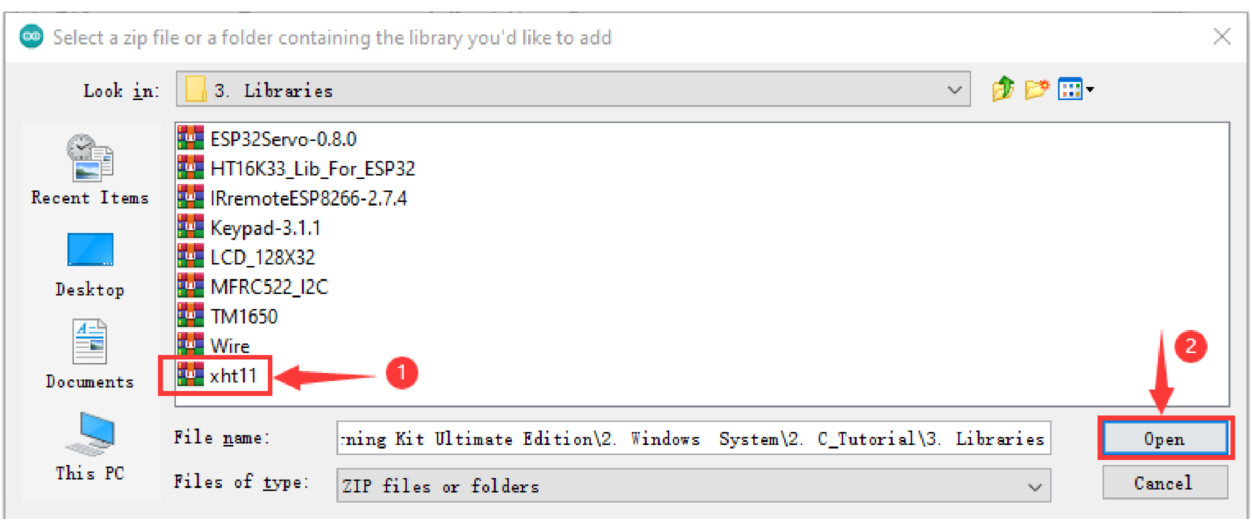
5. Add the xht11 and ESP32Servo libraries

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:

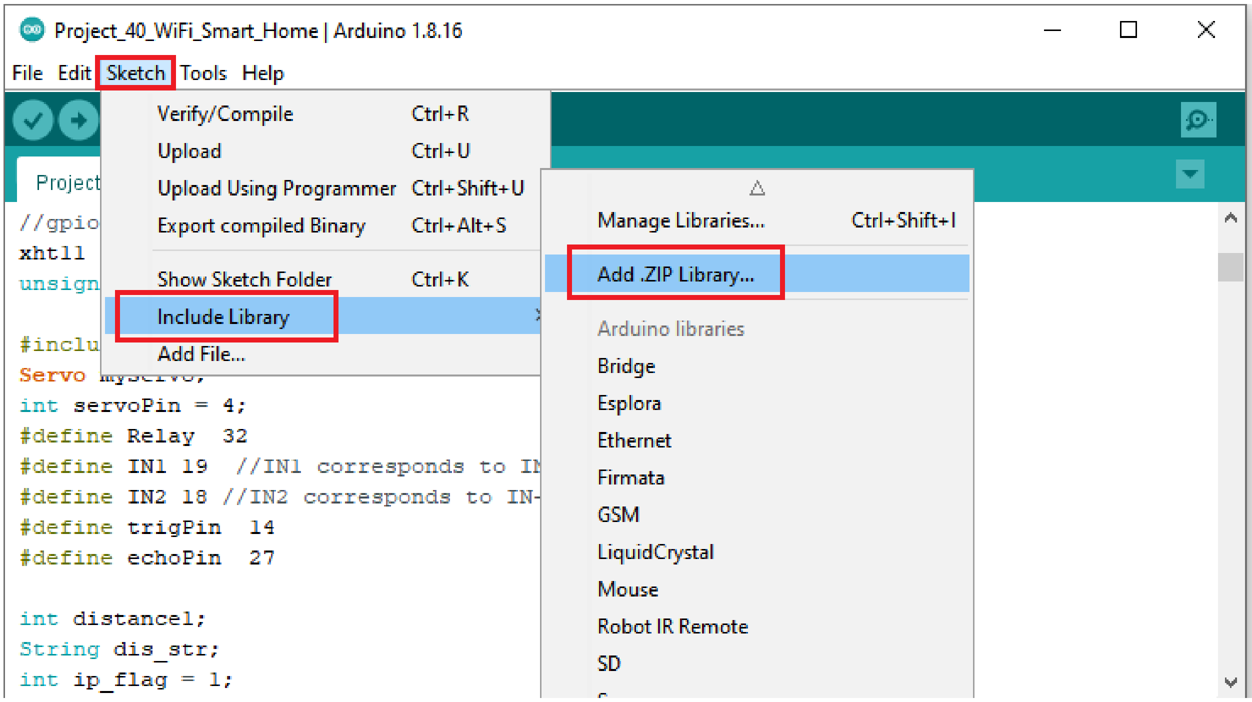
Open the Arduino IDEclick"Sketch"→"Include Library"→"Add .zip Library…". In the pop-up window, find the file named **"2 Windows System\2. C_Tutorial\3.Libraries\xht11.ZIP"** which locates in this directory. Select the **xht11.ZIP** file and then click"Open".

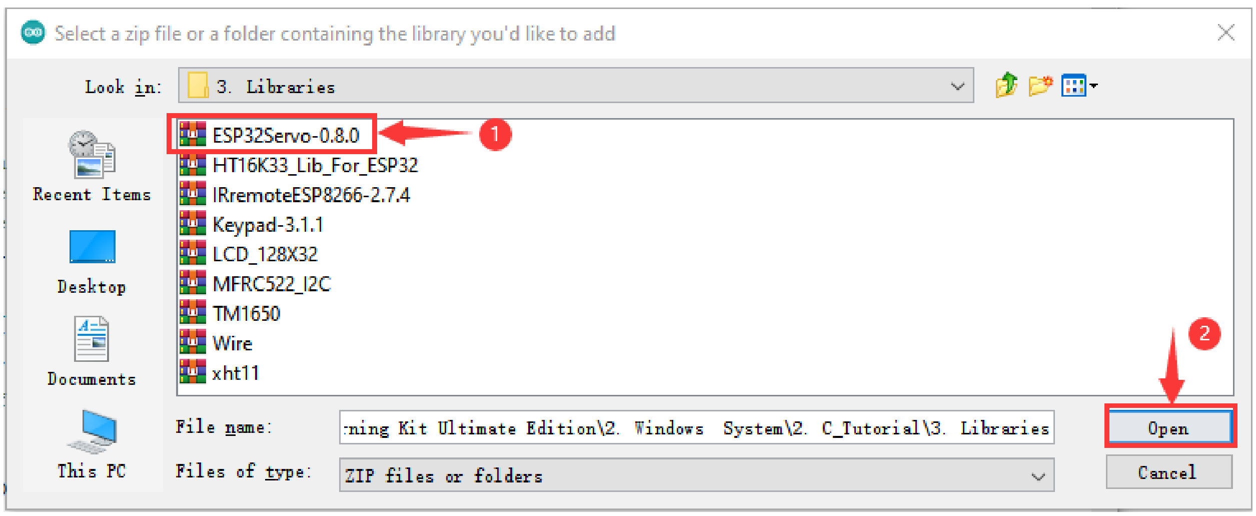




Then Add the ESP32Servo library:

Open the Arduino IDEclick "Sketch"→"Include Library"→"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".





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"**.

```
//**********************************************************************************
/*
 * Filename    : WiFi Smart Home.
 * Description : WiFi APP controls Multiple sensors/modules work to achieve the effect of WiFi smart home.
 * Auther      : http//www.keyestudio.com
*/
##include <Arduino.h>
```

(continues on next page)

(continued from previous page)

```
##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);
```

(continues on next page)

(continued from previous page)

```
  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);
  }
```

(continues on next page)

(continued from previous page)

```
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")
{
```

(continues on next page)

(continued from previous page)

```
    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];
```

(continues on next page)

(continued from previous page)

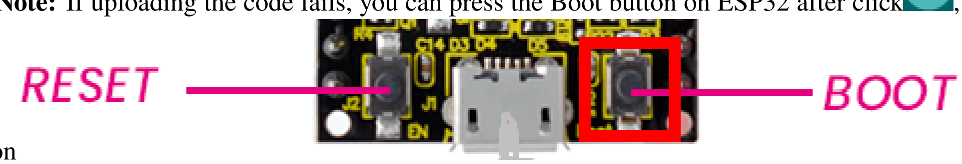
```
          client.print("Humidity(%) = ");
          client.println(t_val3);
        }
        humidity_state = 0;
      }
  }
  else if(req == "/btn/11")
  {
    Serial.write('l');
    client.println("turn off the humidity");
    humidity_state = 1;
    }
  //client.print(s);
  client.stop();
}

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.

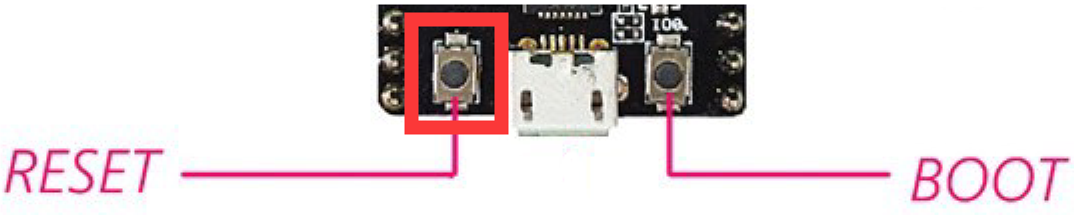
```
const char* ssid = "ChinaNet-2.4G-0DF0"; //the name of user's wifi
const char* password = "ChinaNet@233";  //the password of user's wifi
```

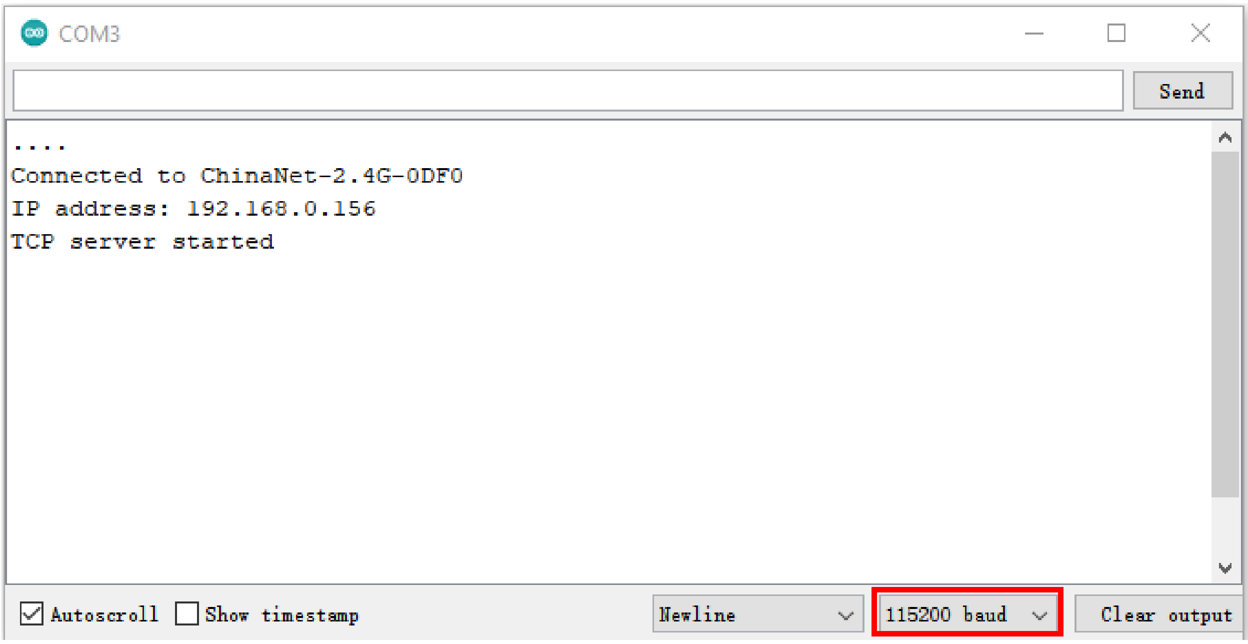
7. Project result

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 ,

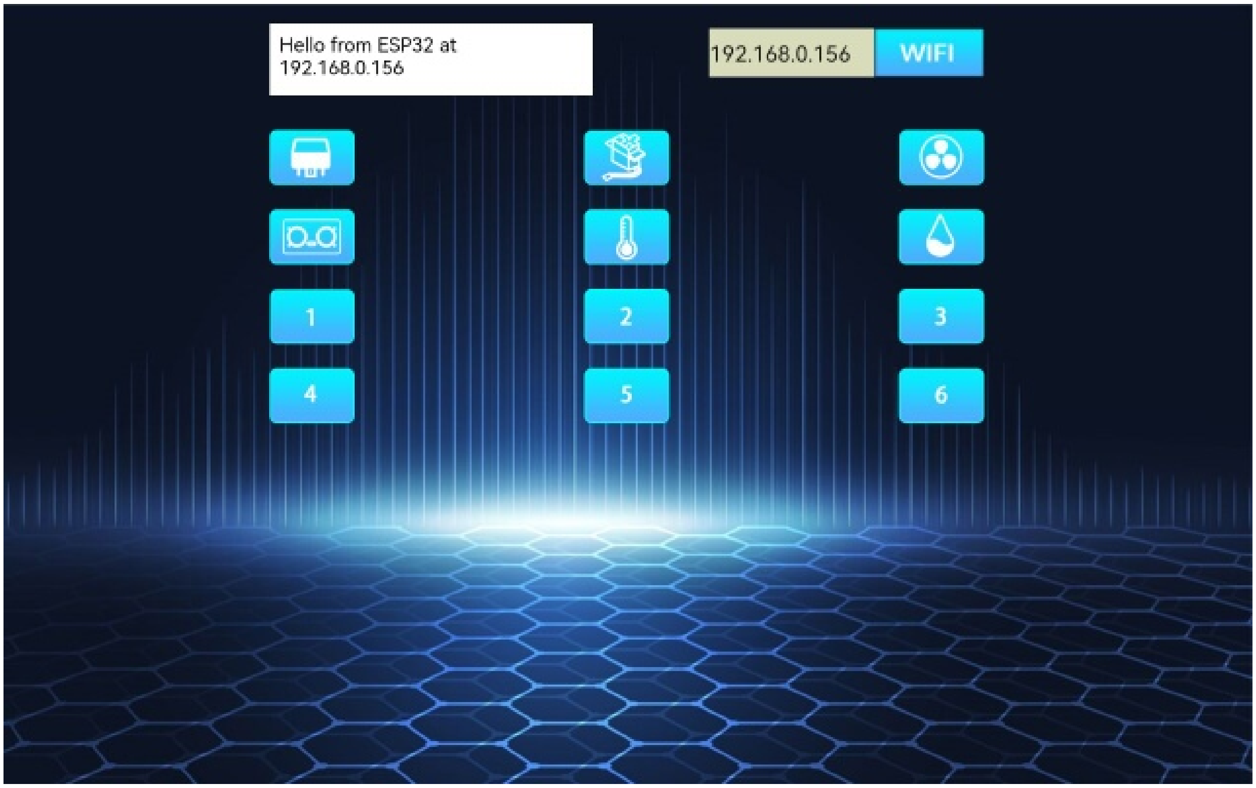


and release the Boot button after the percentage of uploading progress appearsopen the serial monitor and set baud rate to 115200. In this way, the serial port monitor prints the detected WiFi IP address,(If open the serial monitor and set the baud rate to 115200, the information is not displayed, please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

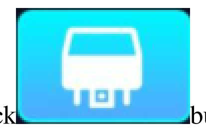




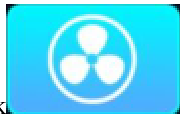
Then open the WiFi APP and enter the detected WiFi IP address in the text box in front of the WiFi button (for example, the IP address shownby the serial port monitor below :192.168.0.156), then click the WiFi button, "Hello from ESP32 at 192.168.0.156"is displayed in the text box next to the WiFi IP 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:

1) Click  button, the relay will be opened, the APP will display `turn on the relay` and the indicator lights up on the module. Click media/5b9754cb6ec4f995c9eada1da89a8969.png again, the relay will be closed, the APP will display `turn off the relay` and the indicator on the module is off.

2) Click media/c54f78d819d4e6a8310eaeb79ff66910.png buttonthe servo rotates 180°the APP will display media/c54f78d819d4e6a8310eaeb79ff66910.png againthe APP will display media/dee12bee3866542bfe5d70a539f79f0b.png the servo rotates 0°.

3) Click  buttonthe motorwith small fan bladesrotatesthe APP will display media/5490abf5b2f8a1d9cea3055da07c251c.png againclose the motorthe APP will display media/de6da02ede6d63344546173d36bf5371.png

4) Click media/95bfbe879d2391e4e48dcae085abe5a6.png buttonthe ultrasonic sensor detects the distance, put an object in front of the ultrasonic sensor, the APP will display media/e431e1b9c95bed37b053ae9617f93676.png different distances show different numbers, the distance between the object and the ultrasonic sensor is 6cmclick media/95bfbe879d2391e4e48dcae085abe5a6.png again, turn off the sensor, the APP will display media/b1df35af68601022e54b7e575b0a07c7.png.

5) Click media/08c8a35841b31fa4b5327fb7b23a7af5.png buttonthe temperature and humidity sensor measures the temperature in the environment, the APP will display media/0aafd0af43f80b692eca3d6732b551b2.png different temperatures show different temperature valuesthe ambient tem-

perature is 30°C., click media/08c8a35841b31fa4b5327fb7b23a7af5.png again, turn off the sensorthe APP will display media/82887a1385bc7411ecbdc41f6.

6) Click media/d8e3463ab2f644b3300cdeaa2a68e4c2.png buttonthe temperature and humidity sensor measures the humidity in the environment,the APP will display Humidity(%) = 55 different humiditys show different humiditys values, the ambient humidity is 55%click media/d8e3463ab2f644b3300cdeaa2a68e4c2.png againturn off the sensor, the APP will display media/adc18d06e626af067286da9040c20252.png.

CHAPTER

SIX

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.

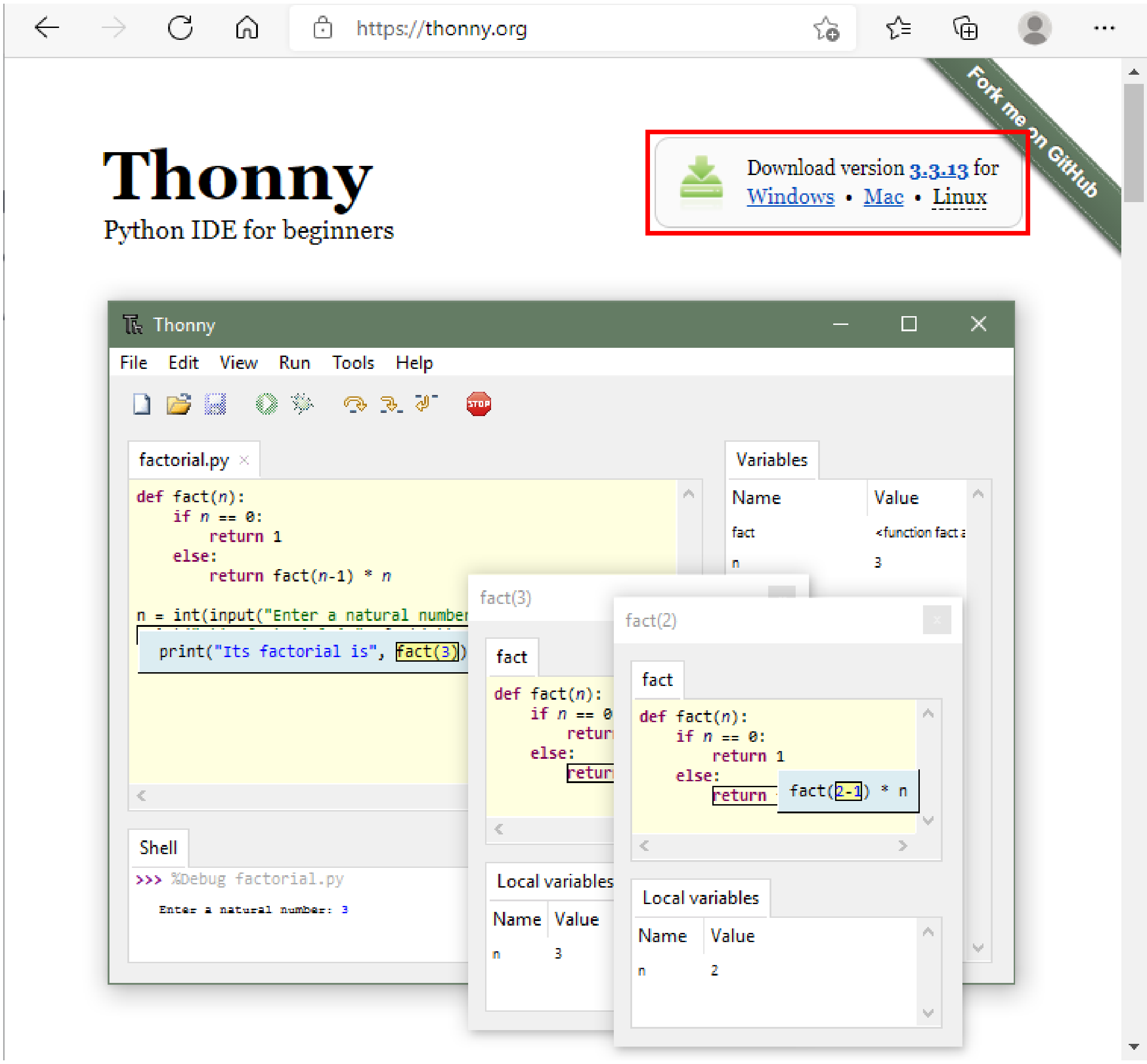
6.1.1 (1) Downloading Thonny

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

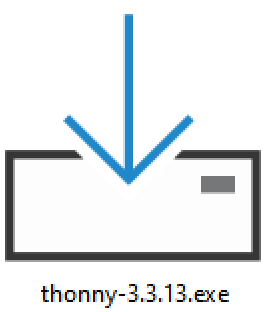
Follow the instruction of official website to install Thonny or click the links below to download and install. (Select the appropriate one based on your operating system.)

Operating System	Download links/methods
MAC OS	https://github.com/thonny/thonny/releases/download/v3.2.7/thonny-3.2.7.pkg
Windows	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 thonny**Distro packages (may not be the latest version):Debian, Rasbian, Ubuntu, Mint and others:** sudo apt install thonny**Fedora:** sudo dnf install thonny



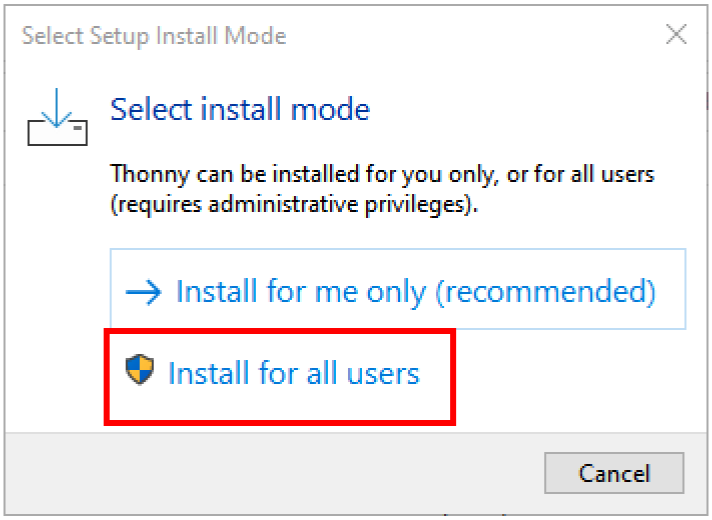
6.1.2 (2) Install Thonny on Windows

The icon of Thonny after downloading is as below.

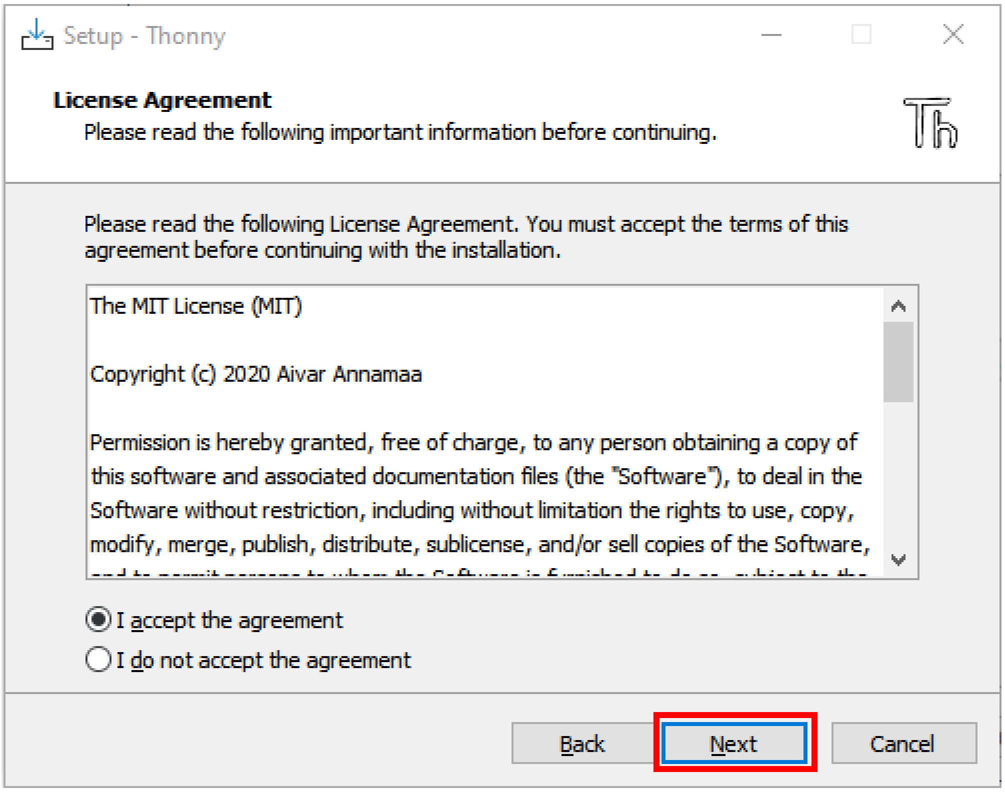


1. Double click"thonny-3.3.13.exe"the following dialog box will appear. I choose"  "to

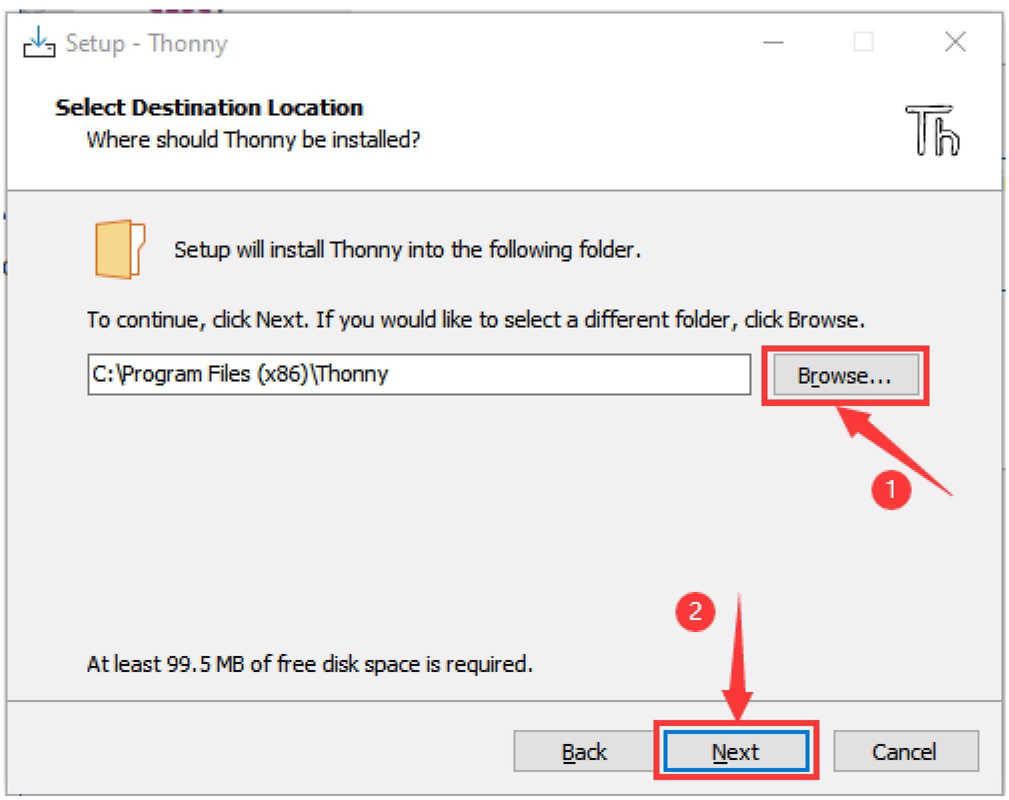
operate, you can also select" → Install for me only (recommended) " to operate.

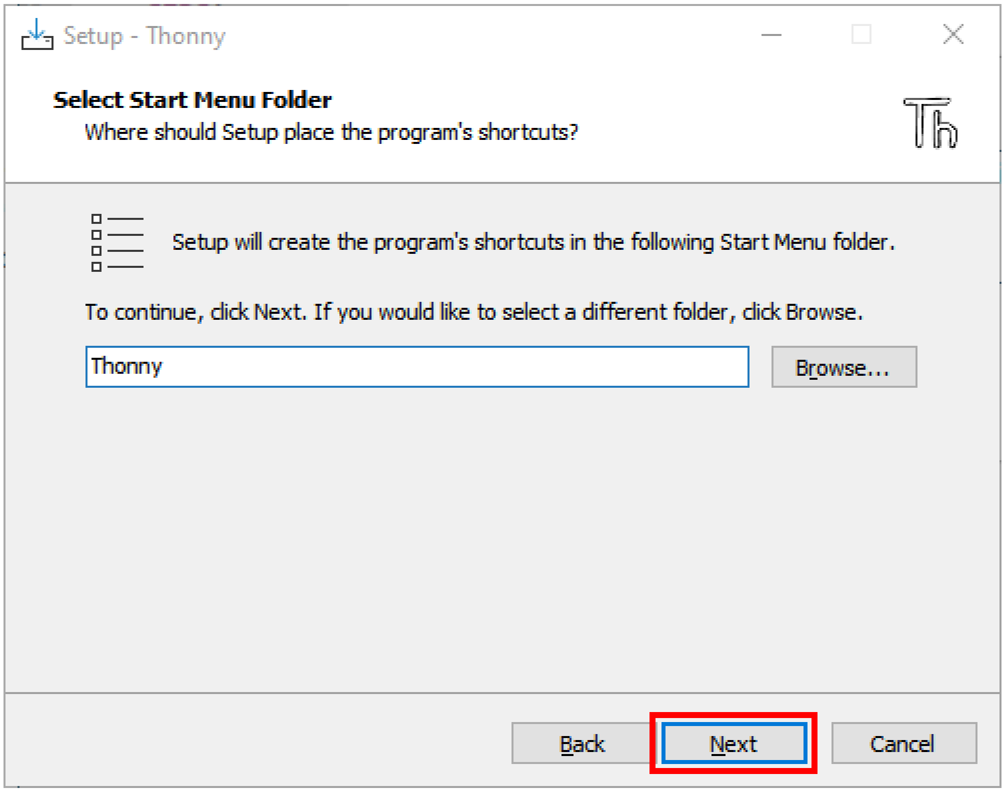


2. If you're not familiar with computer software installation, you can simply keep clicking "**Next**" until the installation completes.

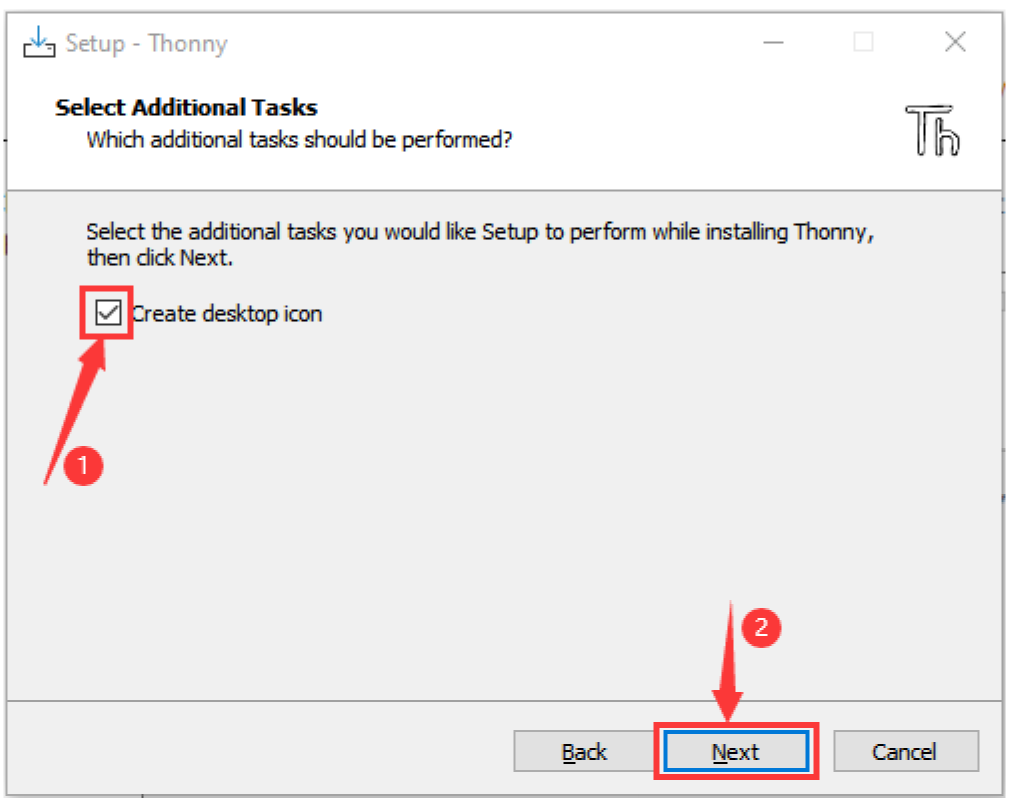


3. If you want to change Thonny's installation path, you can click "**Browse. . .**" to modify it. After selecting installation path, click "**OK**". If you do not want to change it, just click "**Next**".

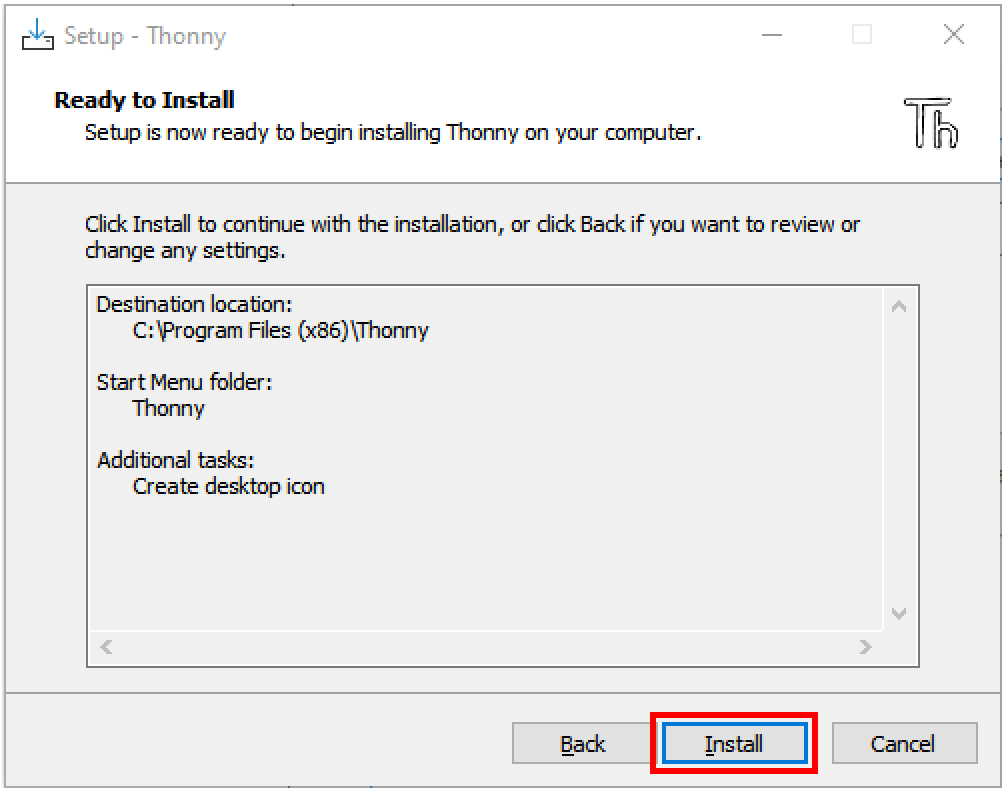




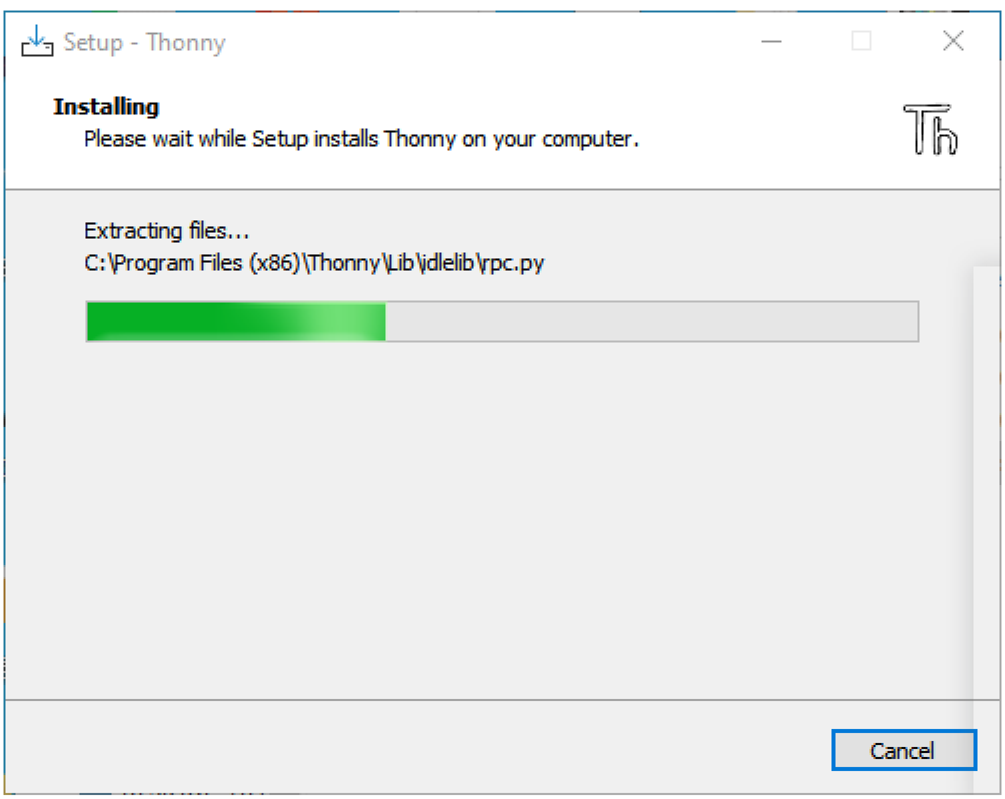
4. Check "**Create desktop icon**" and then it will generate a shortcut on your desktop to facilitate you to open Thonny later.



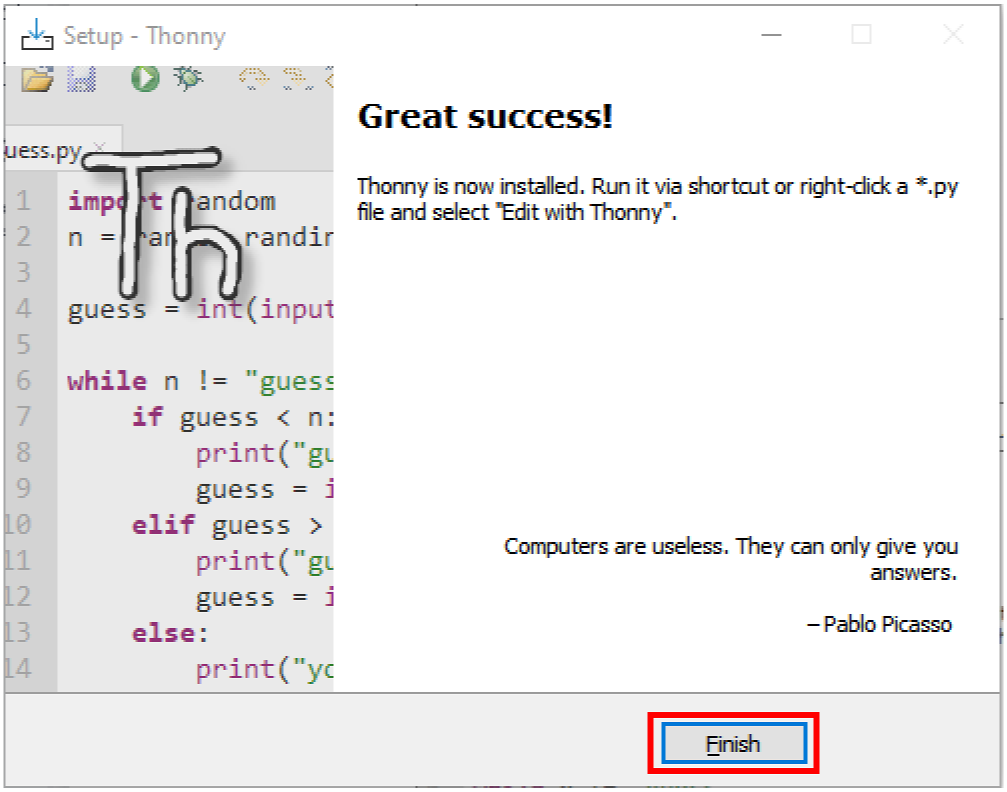
5. Click "**Install**" to install the software.



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



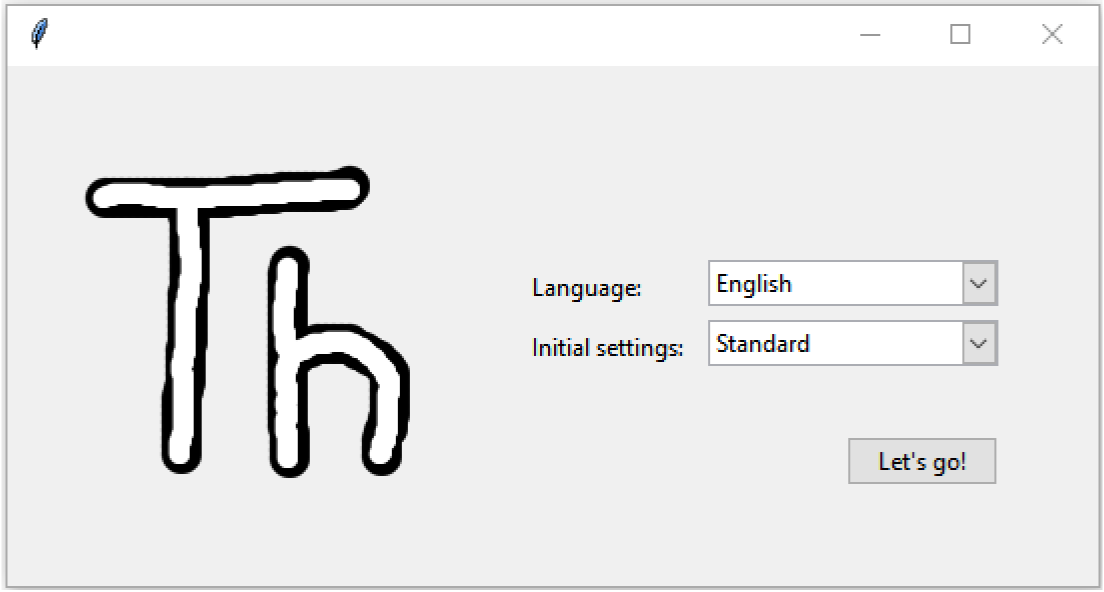
7. Once you see the interface as below, Thonny has been installed successfully. Click "**Finish**".

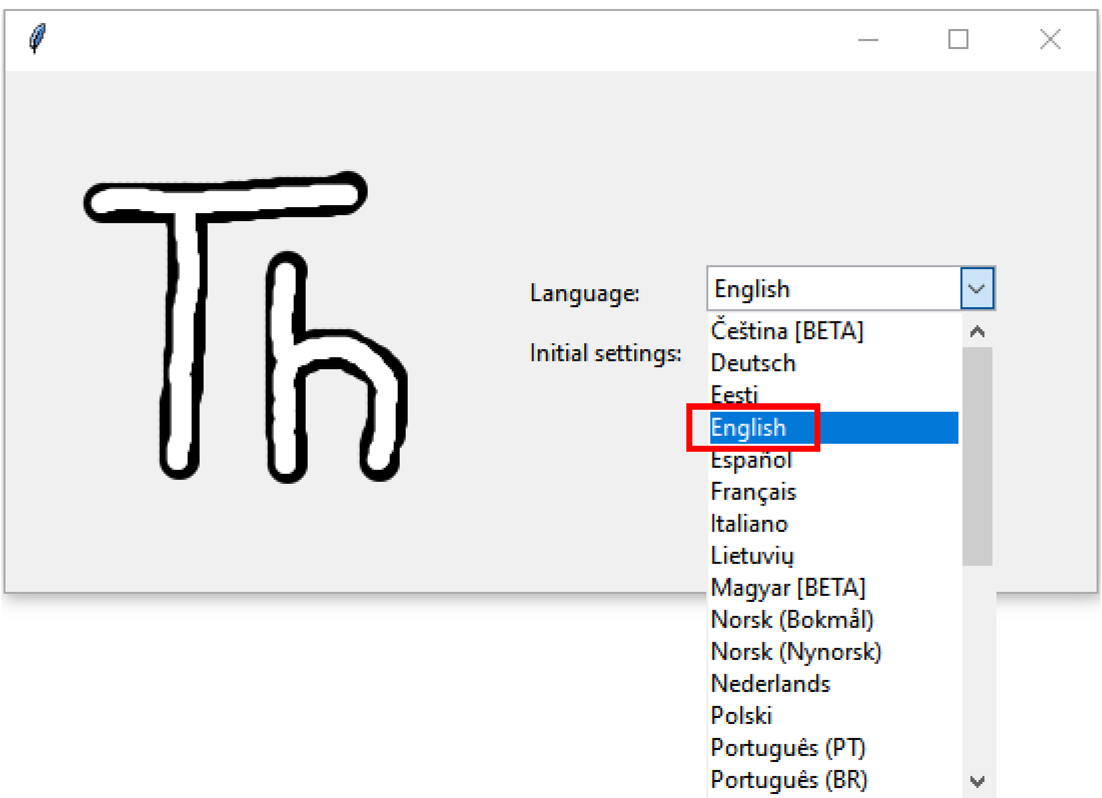


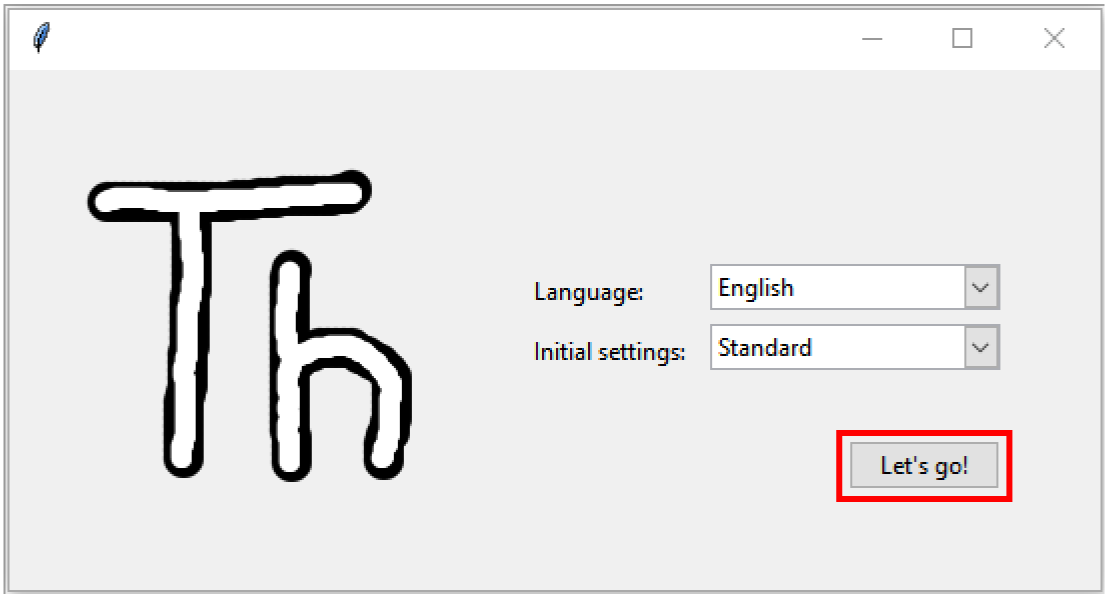
8. If you've check"**Create desktop icon**"during the installation process, you can see the below icon on your desktop.

6.2 2. Basic Configuration of Thonny

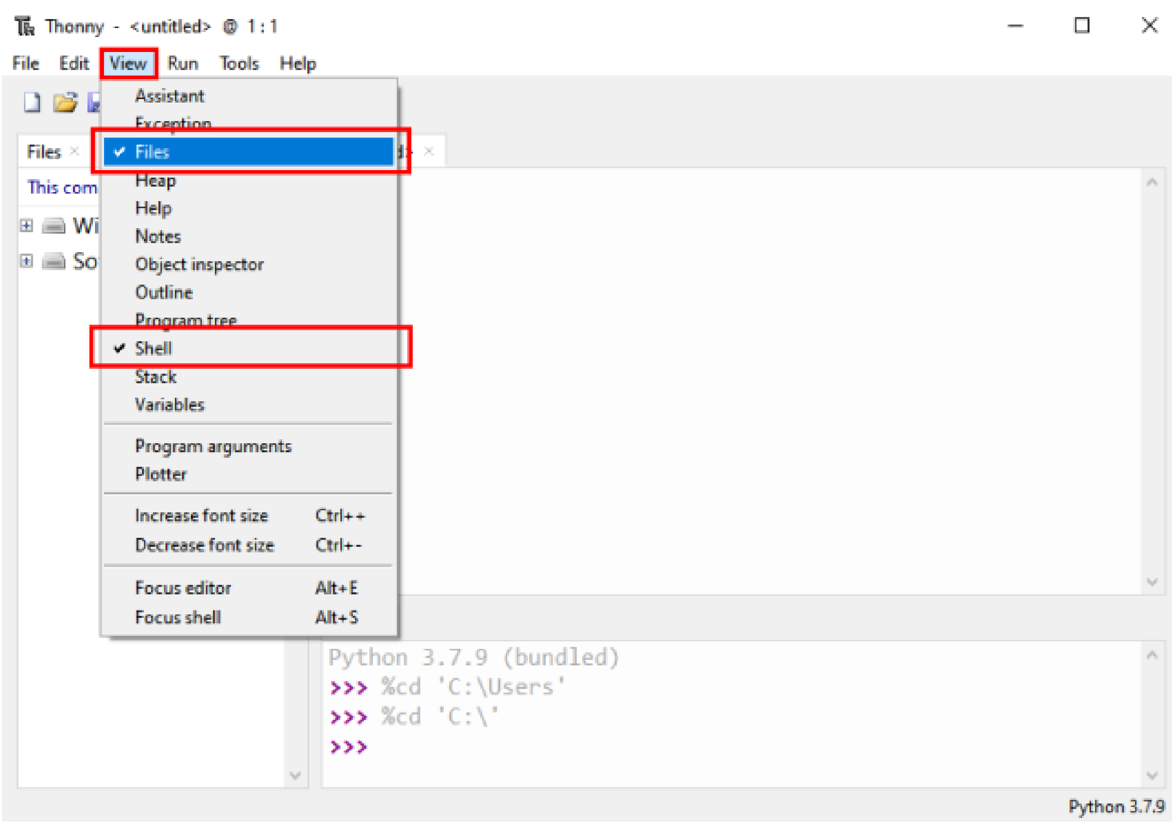
1. Click the desktop icon of Thonny and you can see the interface of it as follows, and we can also choose the language and initial settings. Once set, click "**Let's Go!**"

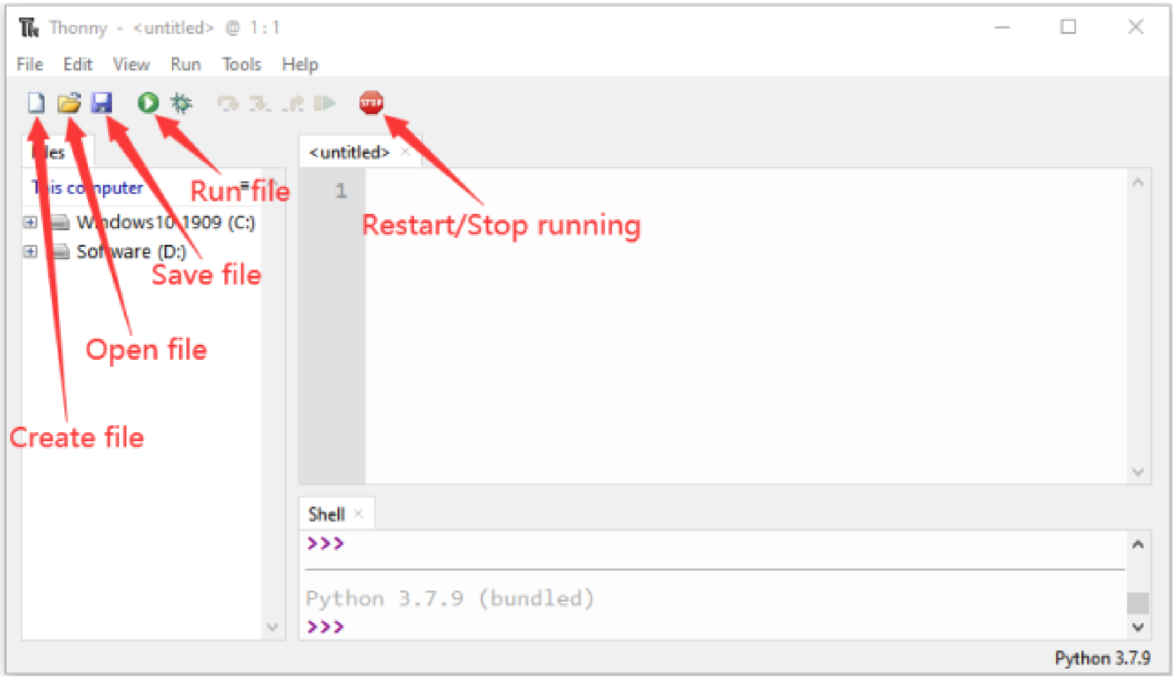

Language:
English
Initial settings:
Standard
Let's go!

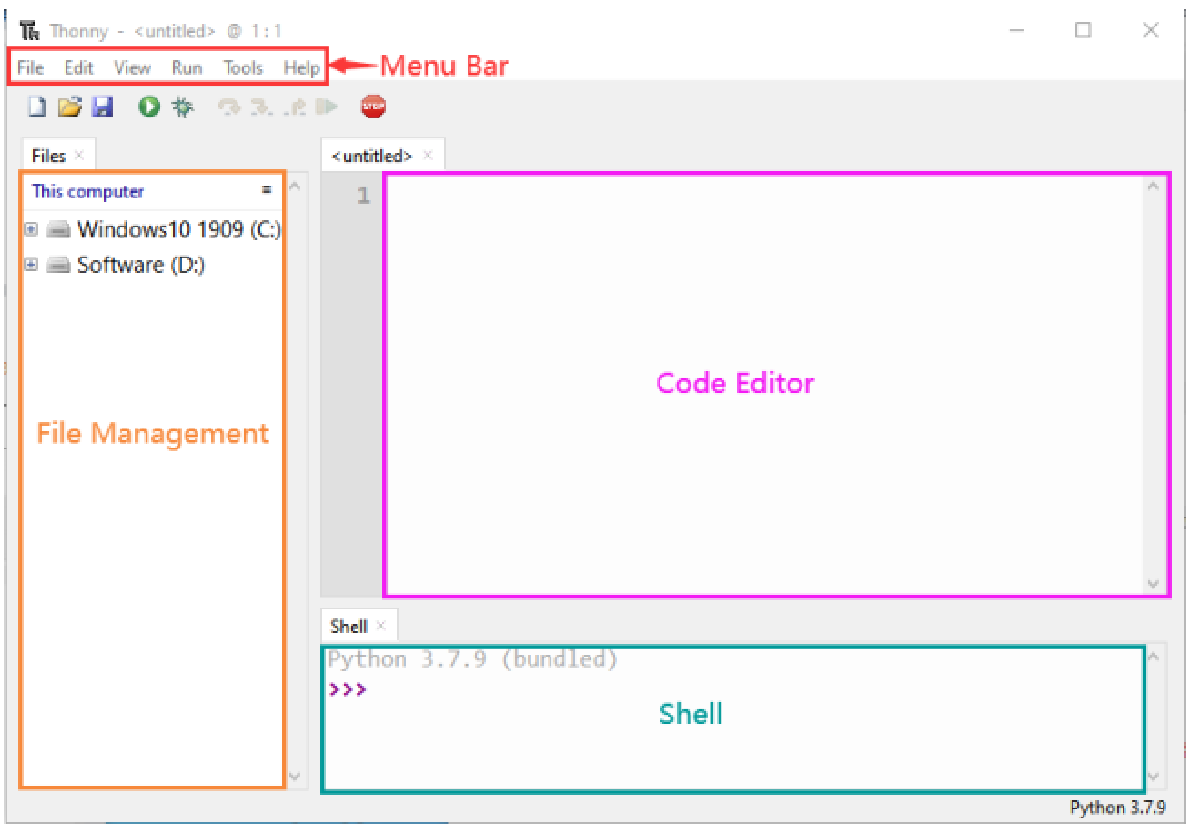

Language:
English
Initial settings:
Čeština [BETA]
Deutsch
Eesti
English
Español
Français
Italiano
Lietuvių
Magyar [BETA]
Norsk (Bokmål)
Norsk (Nynorsk)
Nederlands
Polski
Português (PT)
Português (BR)



2. Select"**View**"→"**Files**"and"**Shell**".




Thonny - <untitled> @ 1 : 1
File Edit View Run Tools Help
Files
This computer
Windows10 1909 (C:)
Software (D:)
<untitled>
Run file
Restart/Stop running
Save file
Open file
Create file
Shell
>>>
Python 3.7.9 (bundled)
>>>
Python 3.7.9


Thonny - <untitled> @ 1 : 1
File Edit View Run Tools Help
Menu Bar
Files
<untitled>
This computer
Windows10 1909 (C:)
Software (D:)
Code Editor
File Management
Shell
Python 3.7.9 (bundled)
>>>
Shell
Python 3.7.9

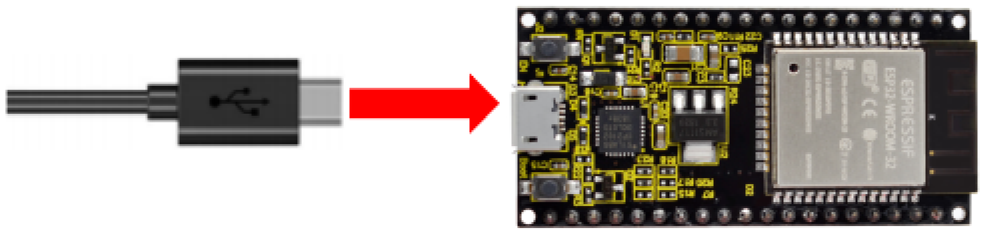
6.3 3.Installing CP2102 driver

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

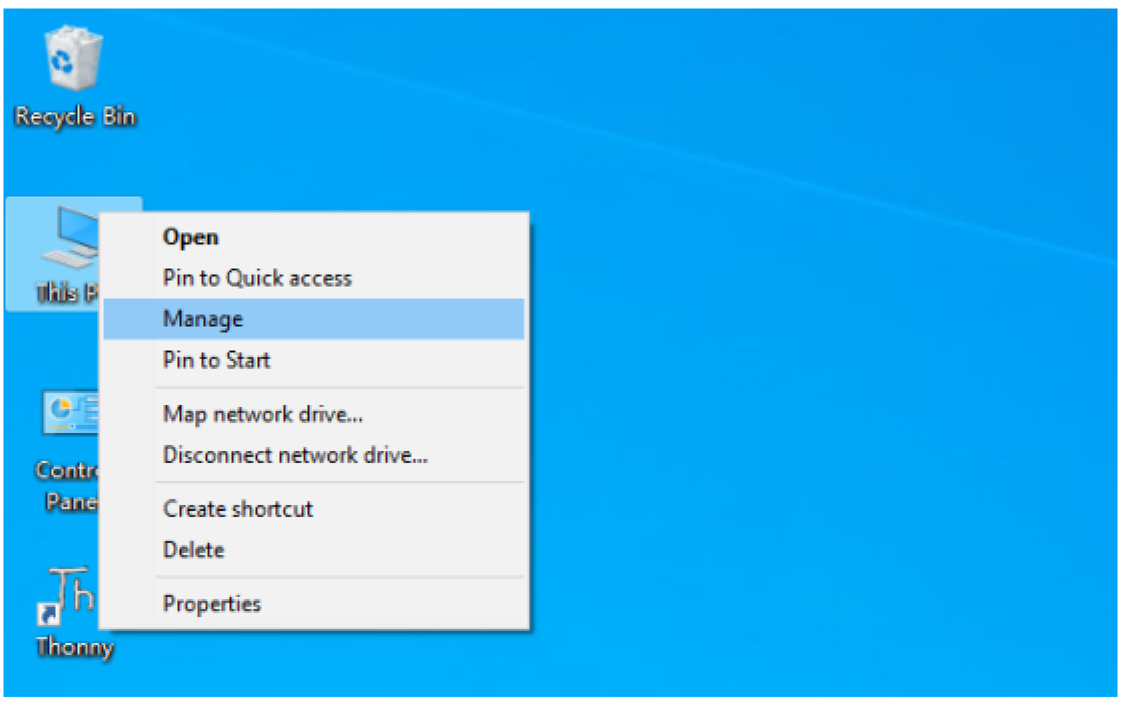
6.3.1 Windows System

Check whether CP2102 has been installed

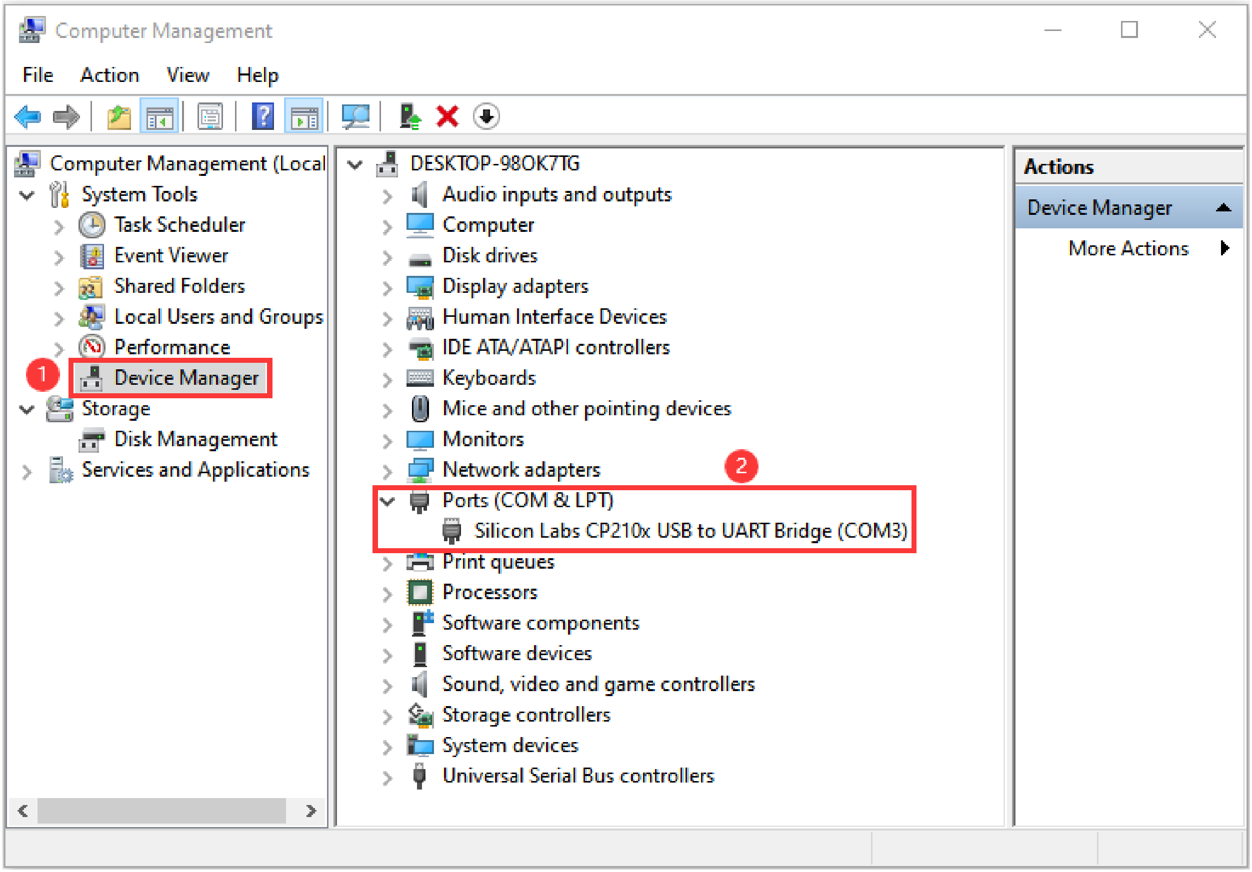
(1). Connect your computer and ESP32 with a USB cable.



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

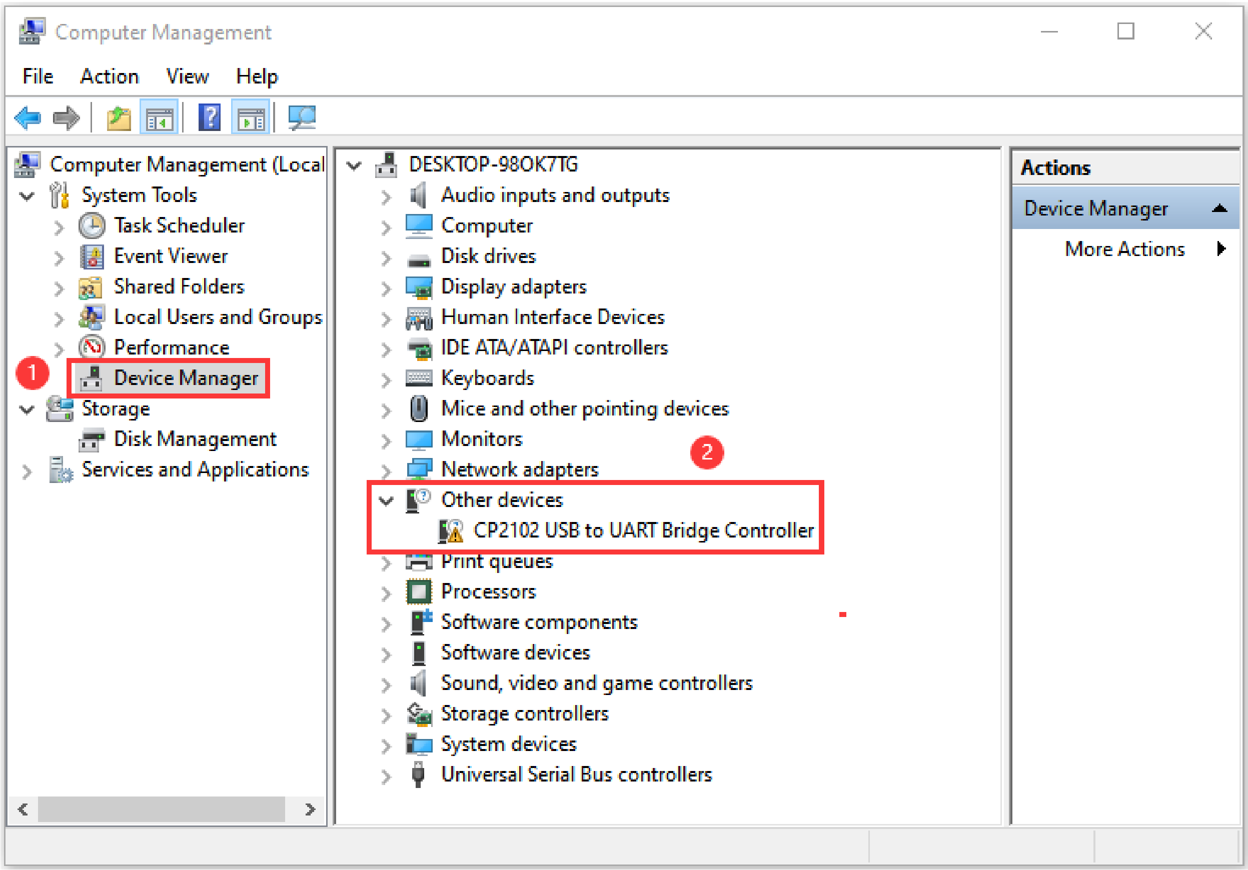


(3). Click "**Device Manager**", your computer has installed CP2102 driveryou can see "Silicon Labs CP210x USB to UART Bridge (COMx)"

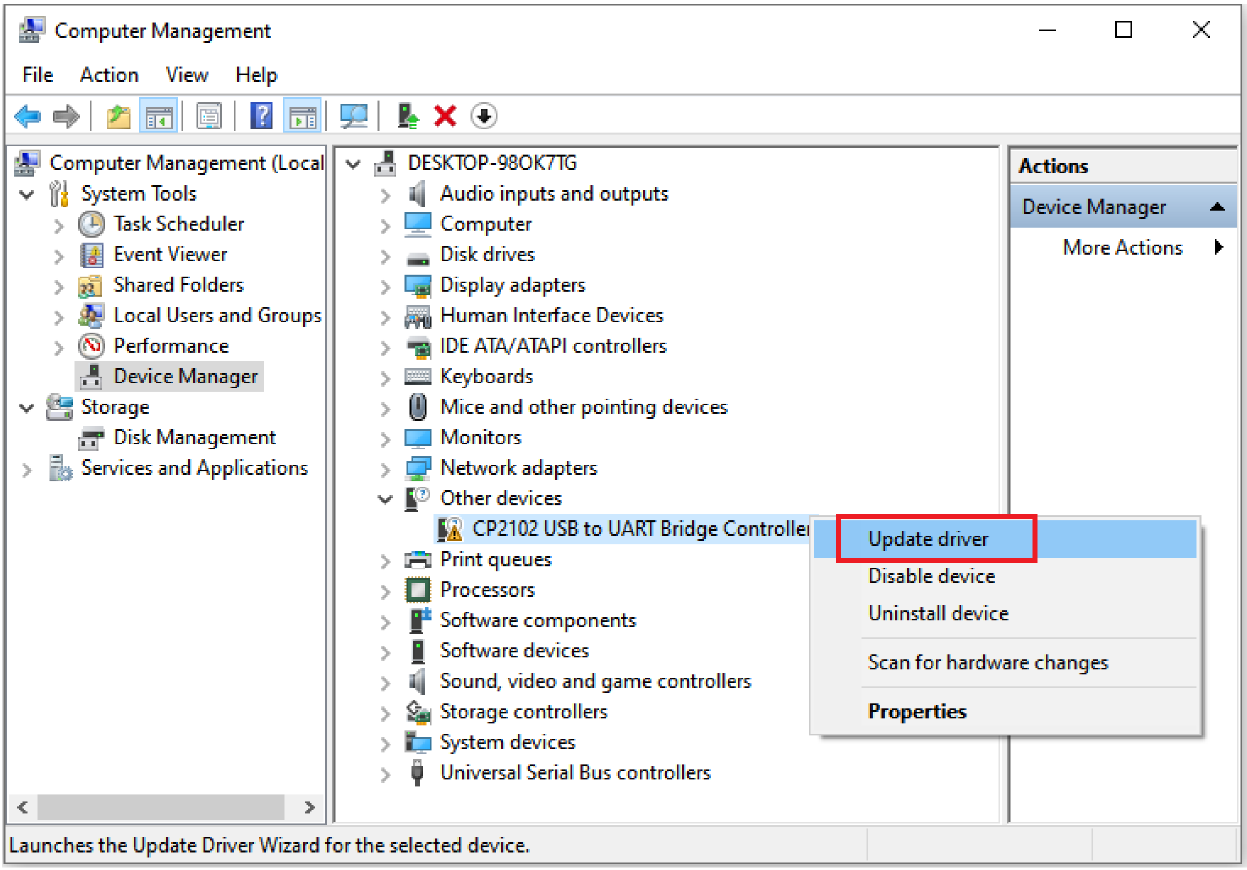


Installing CP2102 driver

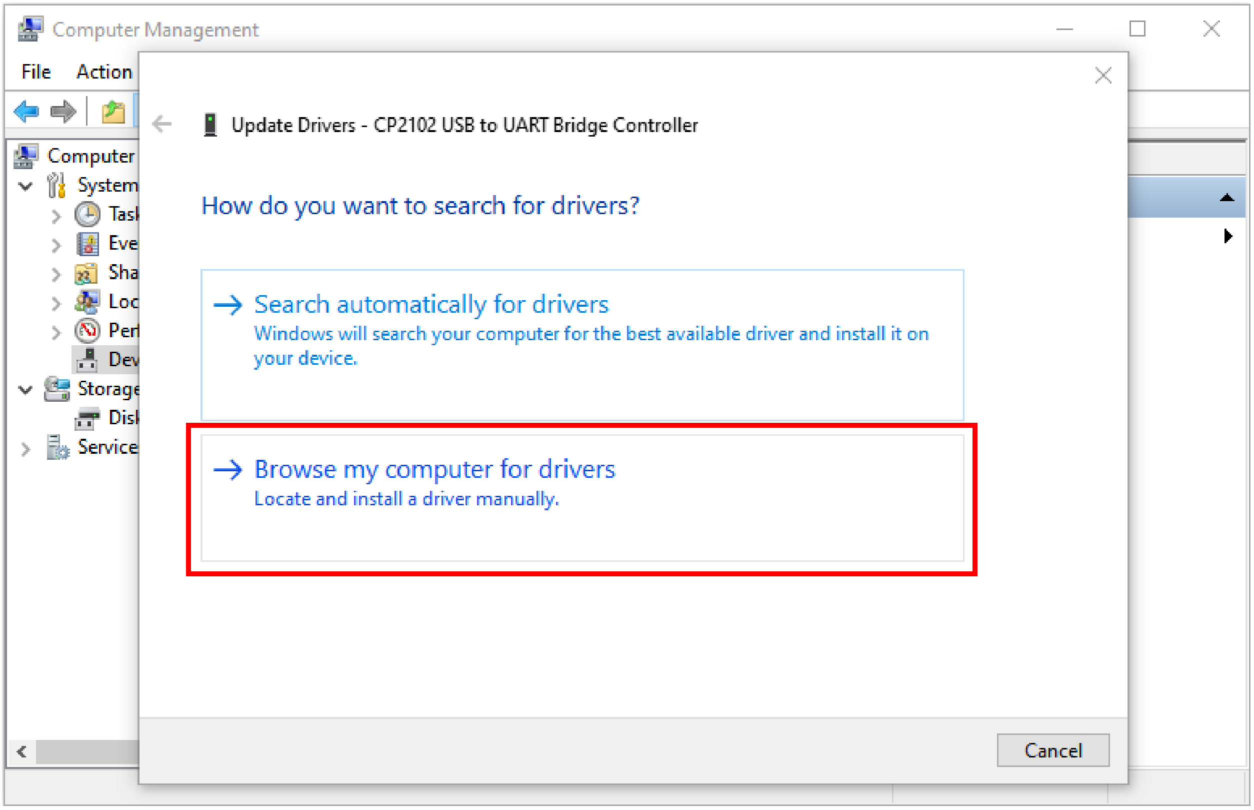
(1) If you have not yet installed the CP2102 driver, you'll see the following interface.



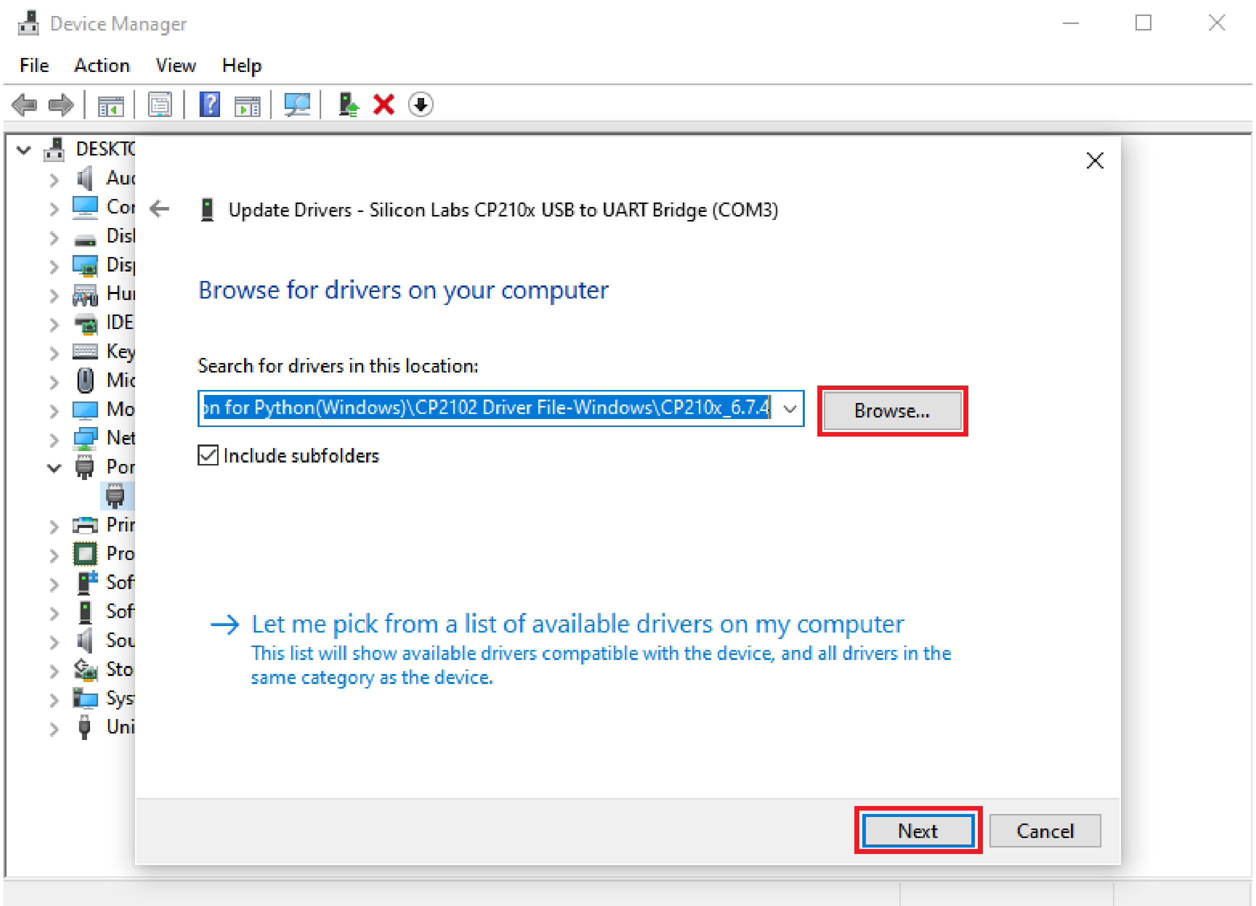
(2) Click "CP2102USB to UART Bridge Controller", and right-click to select "Update driver".



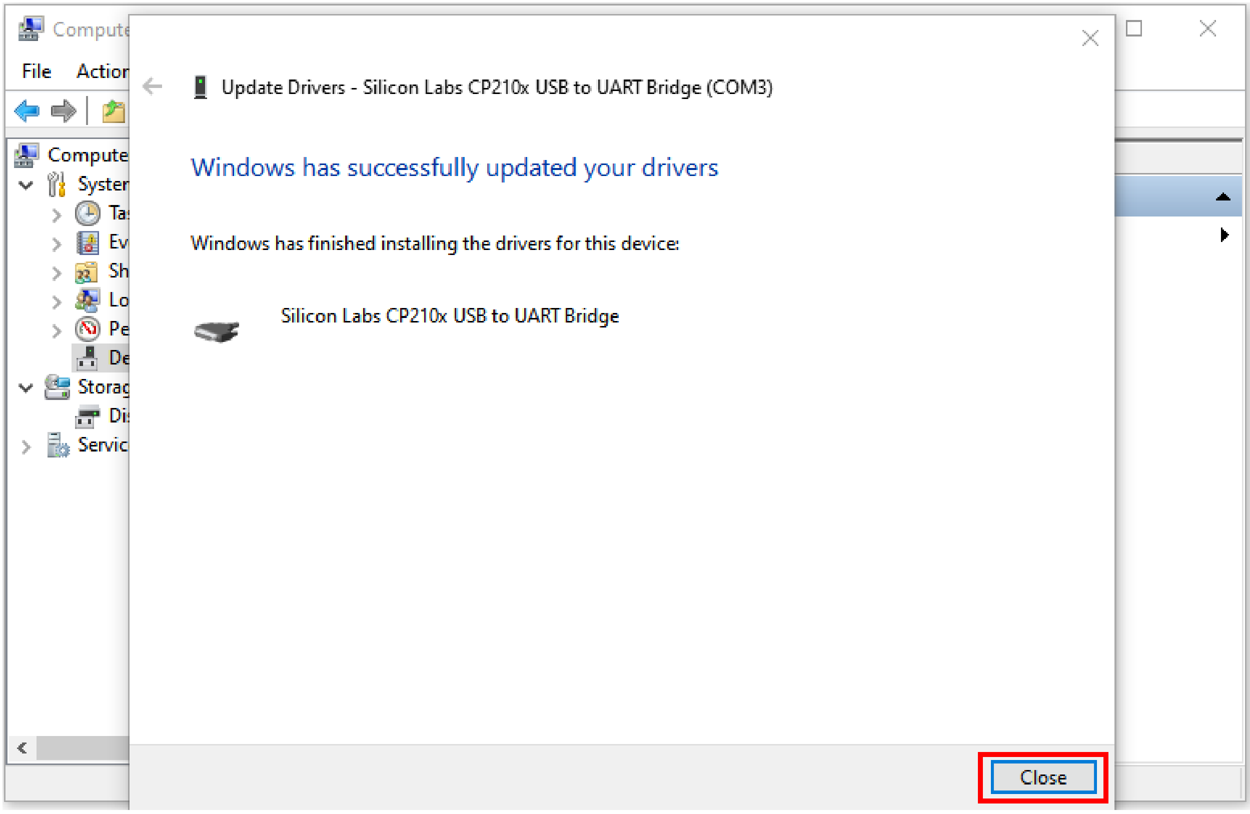
(3) Click “Browse my computer for drivers ”.



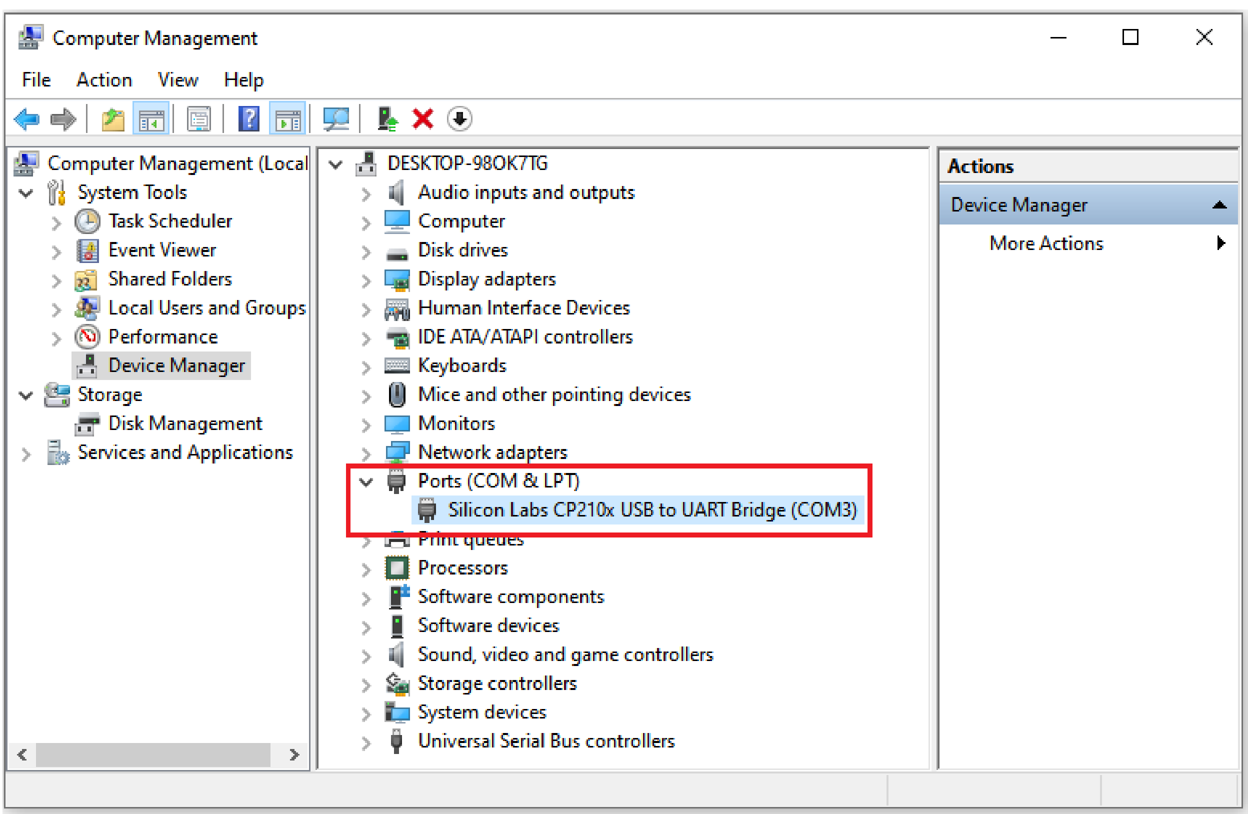
(4) Click "Browse…" select CP210x_6.7.4 (Driver path2. Windows System\1. Python_Tutorial\1. Preparation for Python(Windows)\CP2102 Driver File-Windows), Click "Next".



(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.

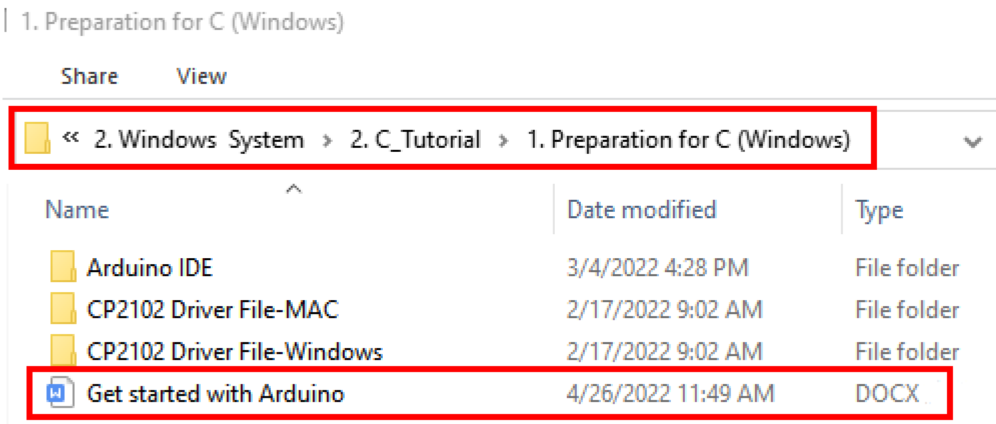


(6) When ESP32 is connected to computer, the interface appears as follows.



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)



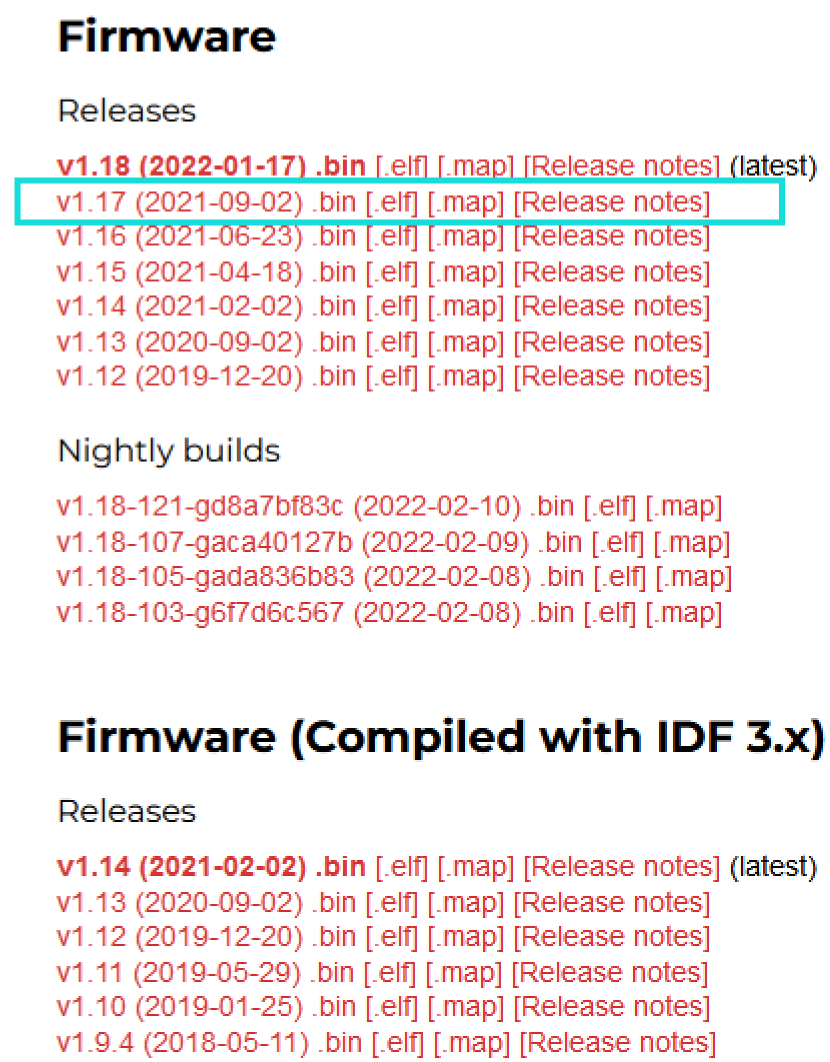
6.4 4.Burning Micropython Firmware (Important)

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/

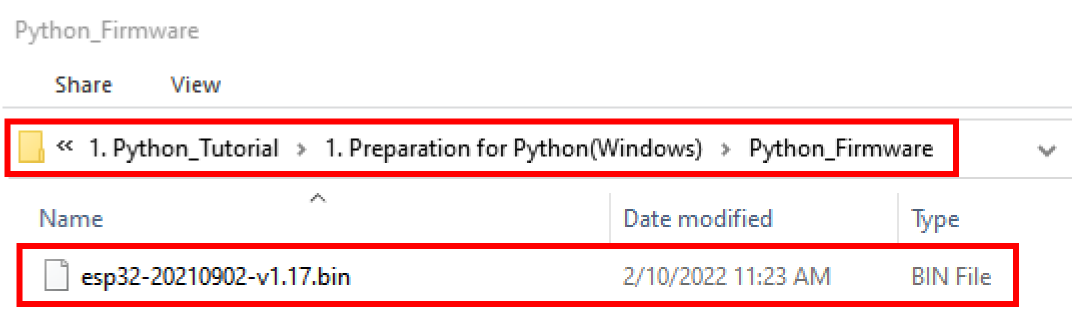
Webpage listing firmware of microPython for ESP32https://micropython.org/download/esp32/



Firmware used in this tutorial is **esp32-20210902-v1.17.bin**

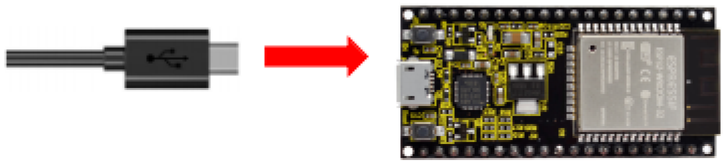
Click the following link to download directlyhttps://micropython.org/resources/firmware/esp32-20210902-v1.17.bin

This file is also provided in our folder“2. Windows System\1. Python_Tutorial\1. Preparation for Python(Windows)\Python_Firmware”.

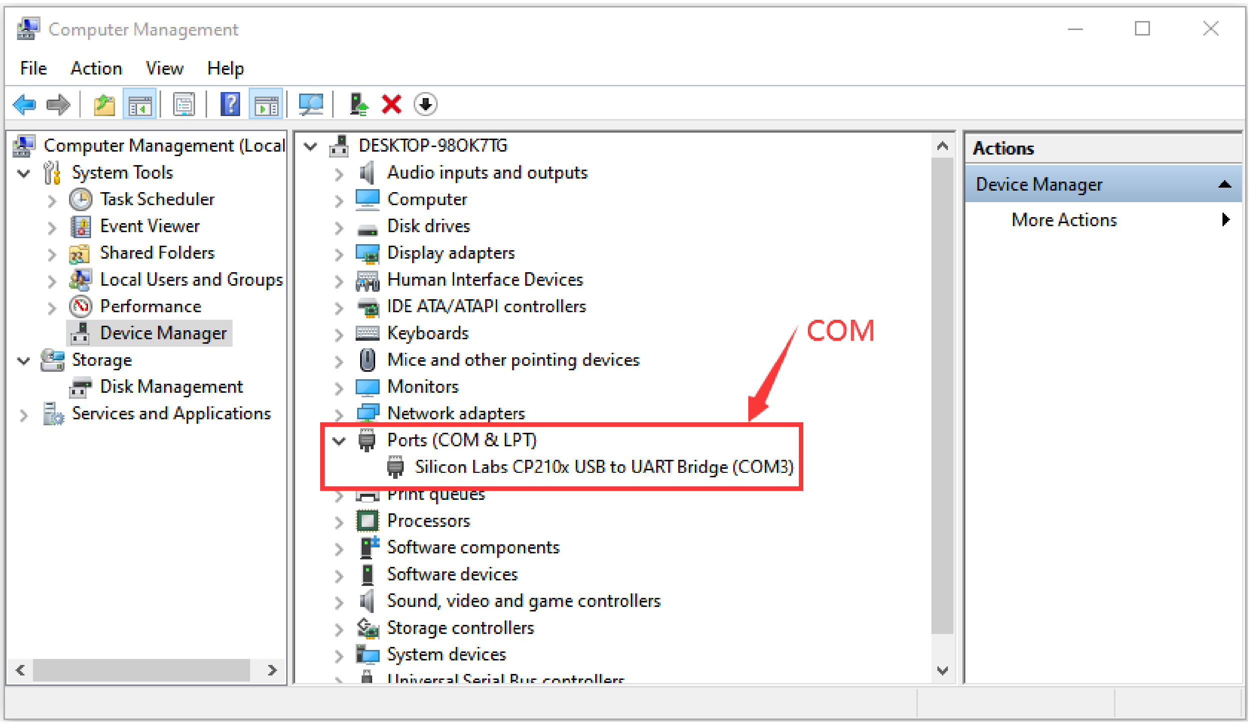


6.4.2 (2) Burning a Micropython Firmware

Connect your computer and ESP32 with a USB cable.

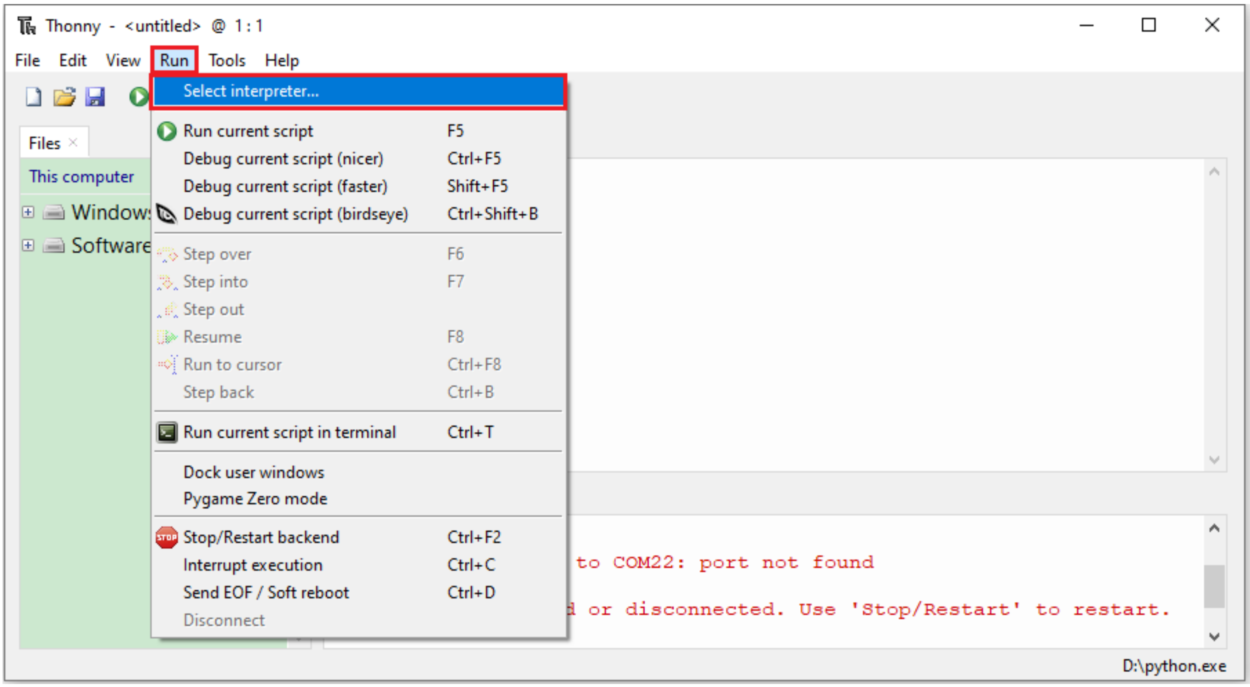


Make sure that the driver has been installed successfully and that it can recognize COM port correctly. Open device manager and expand "**Ports(COM&LPT)**".

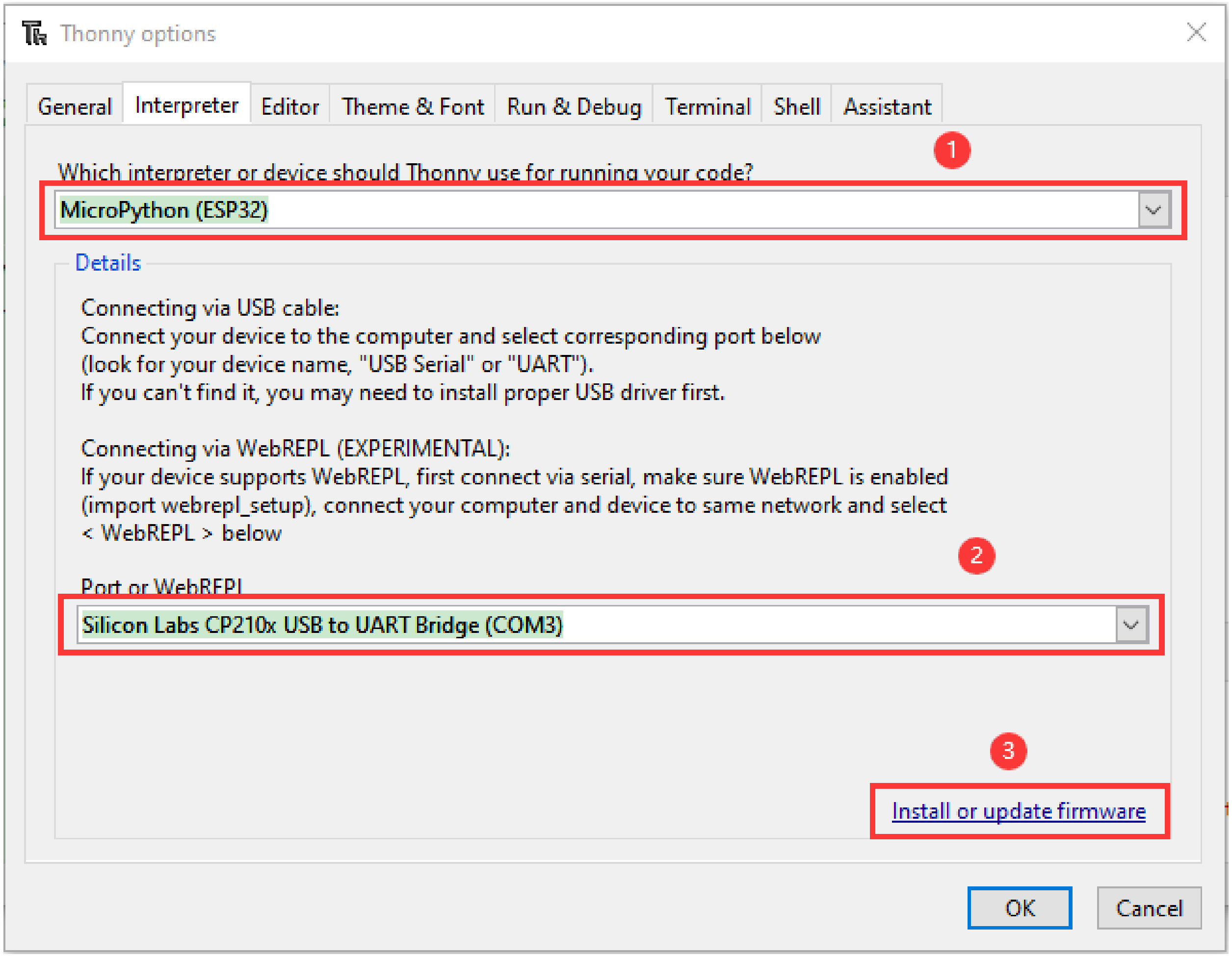


Note: COM ports may be different on different computers

A. Open Thonnyclick "**Run**" and select "**Select interpreter. . .**".

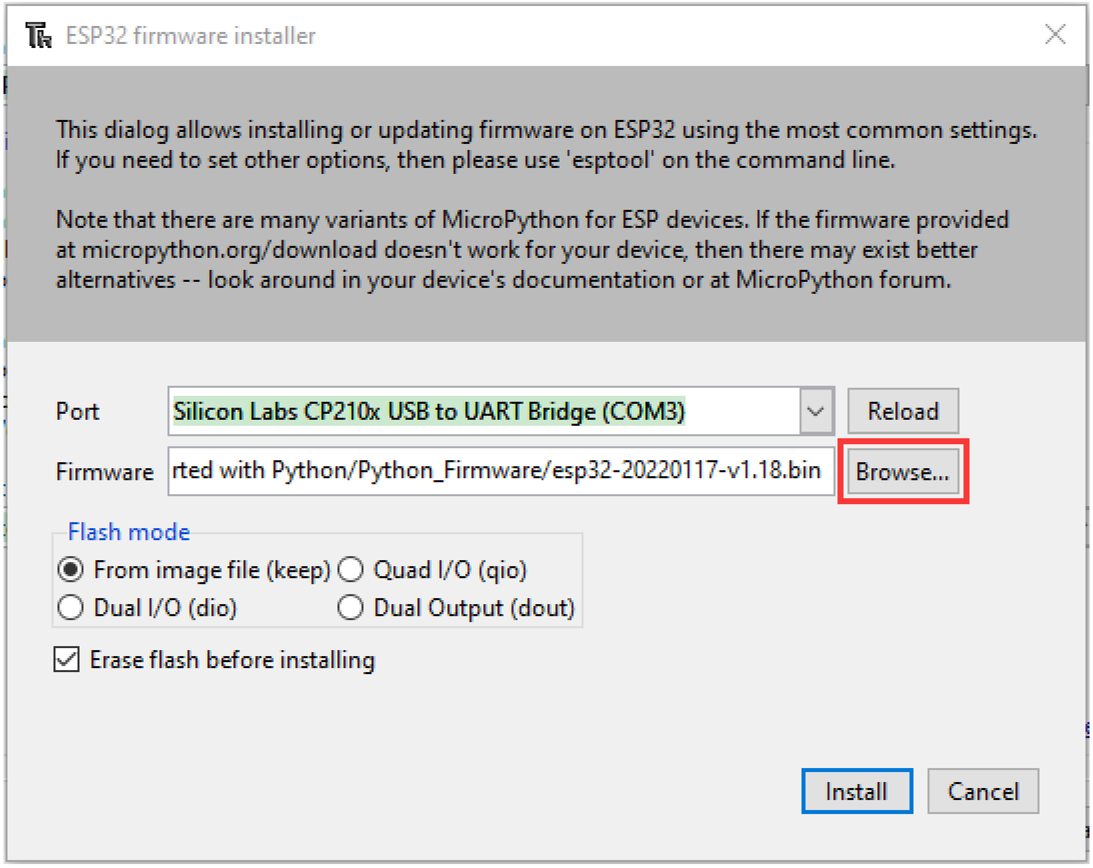


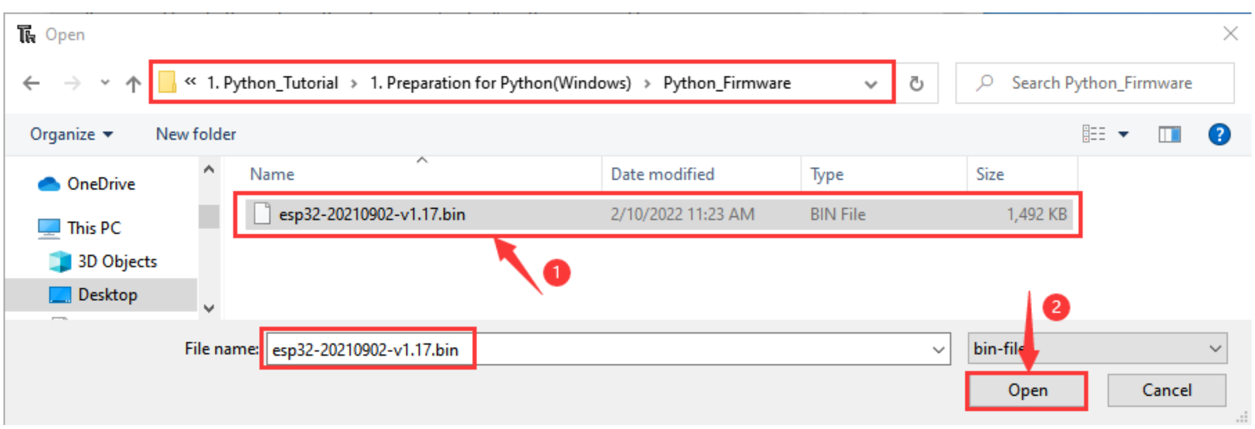
B. Select "**Micropython (ESP32)**", select "**Silicon Labs CP210x USB to UART Bridge(COM3)**", and then click "**Install or update firmware**".

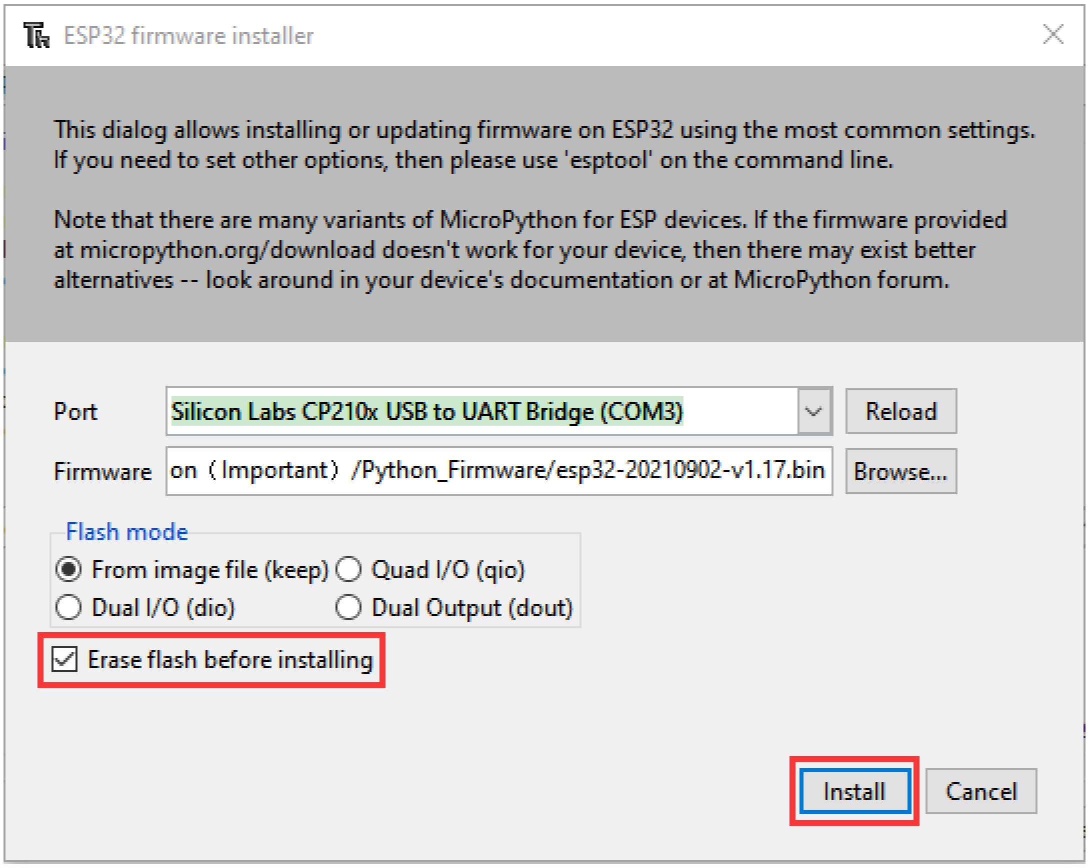


C. The following dialog box pops up, Select "**Silicon Labs CP210x USB to UART Bridge(COM3)**" for "**Port**", and then click "**Browse...**". Select the previous prepared microPython firmware "**esp32-20210902-v1.17.bin**". Check "**Erase flash before installing**" and "**Flash mode**", then click "**Install**" to wait for the prompt of finishing installation.

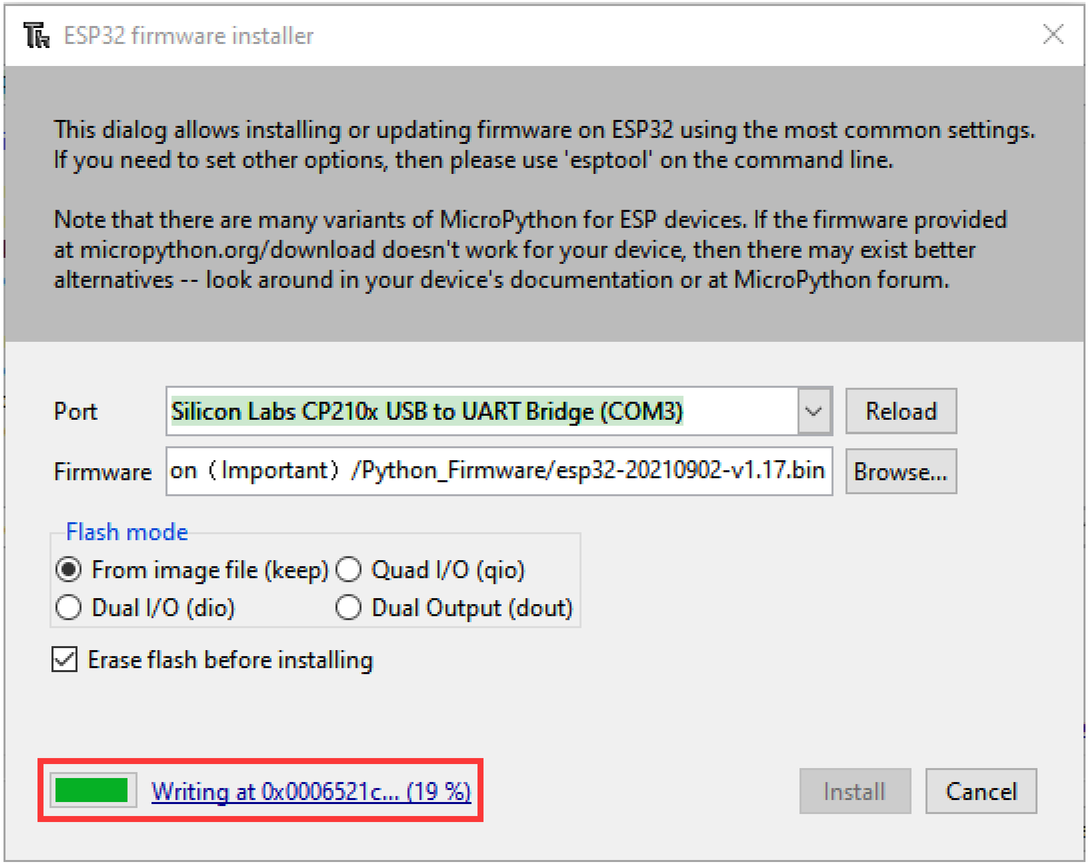
Note: If you fail to install the firmware, press the Boot on the ESP32 mainboard and click "Install" again.

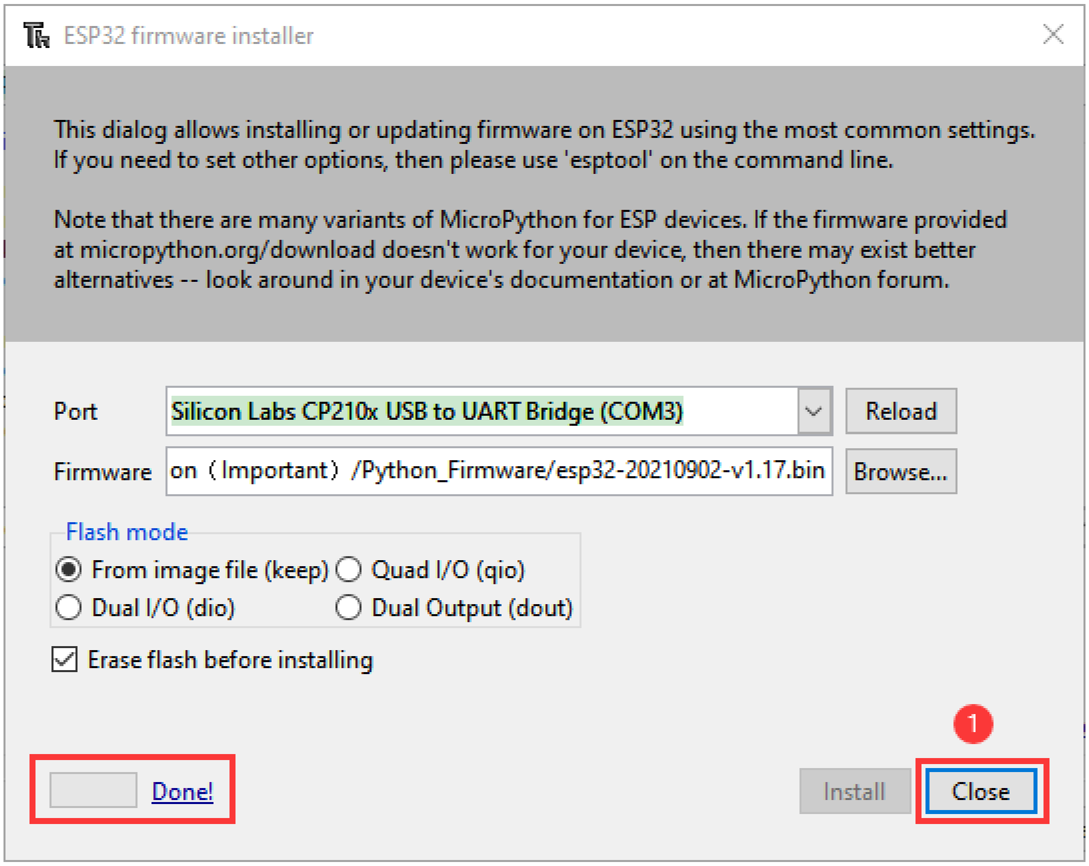

ESP32 firmware installer
This dialog allows installing or updating firmware on ESP32 using the most common settings.
If you need to set other options, then please use 'esptool' on the command line.
Note that there are many variants of MicroPython for ESP devices. If the firmware provided at micropython.org/download 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)
Reload
Firmware
rted with Python/Python_Firmware/esp32-20220117-v1.18.bin
Browse...
Flash mode
From image file (keep)
Quad I/O (qio)
Dual I/O (dio)
Dual Output (dout)
Erase flash before installing
Install
Cancel

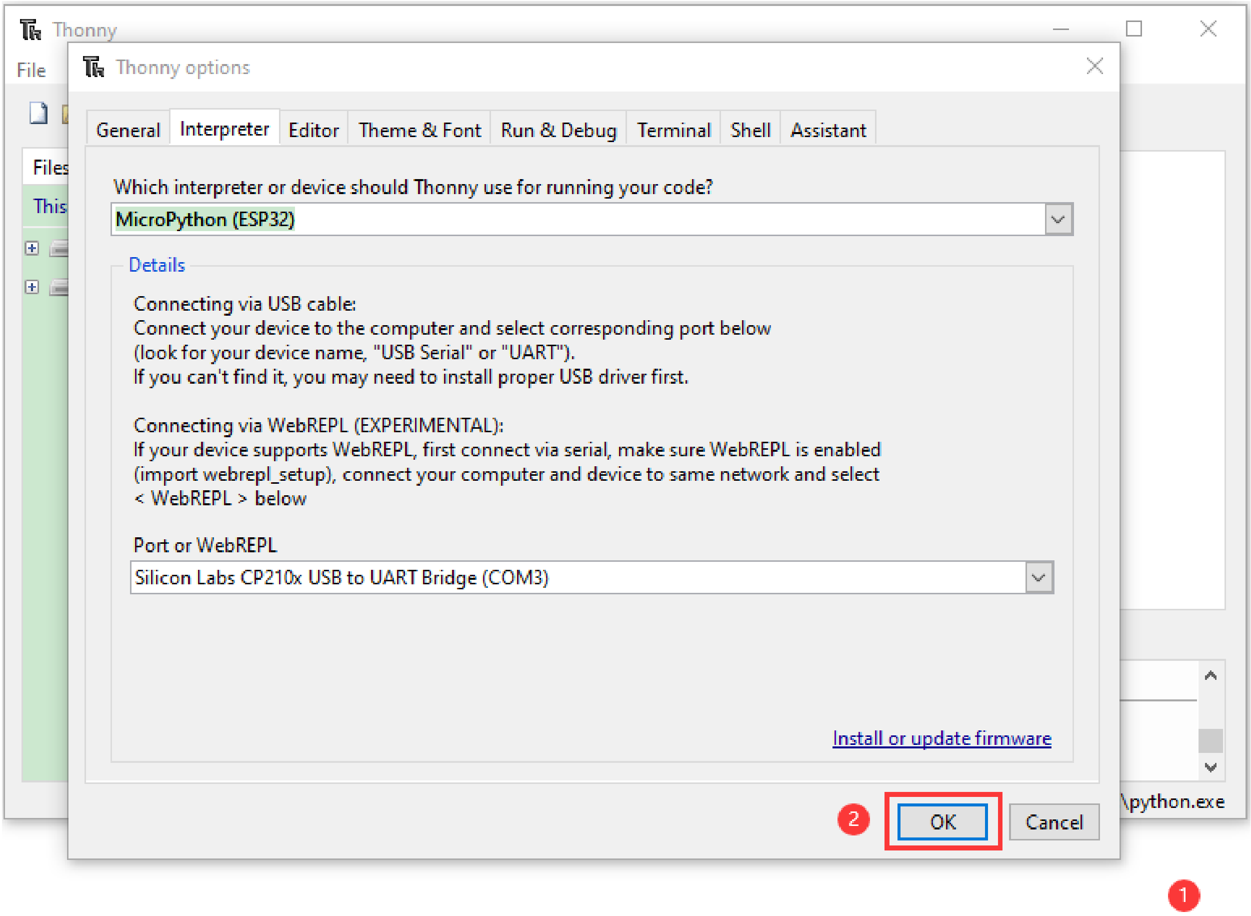

Open
« 1. Python_Tutorial › 1. Preparation for Python(Windows) › Python_Firmware
Search Python_Firmware
Organize
New folder
OneDrive
This PC
3D Objects
Desktop
Name
Date modified
Type
Size
esp32-20210902-v1.17.bin
2/10/2022 11:23 AM
BIN File
1,492 KB
1
2
File name:
esp32-20210902-v1.17.bin
bin-file
Open
Cancel



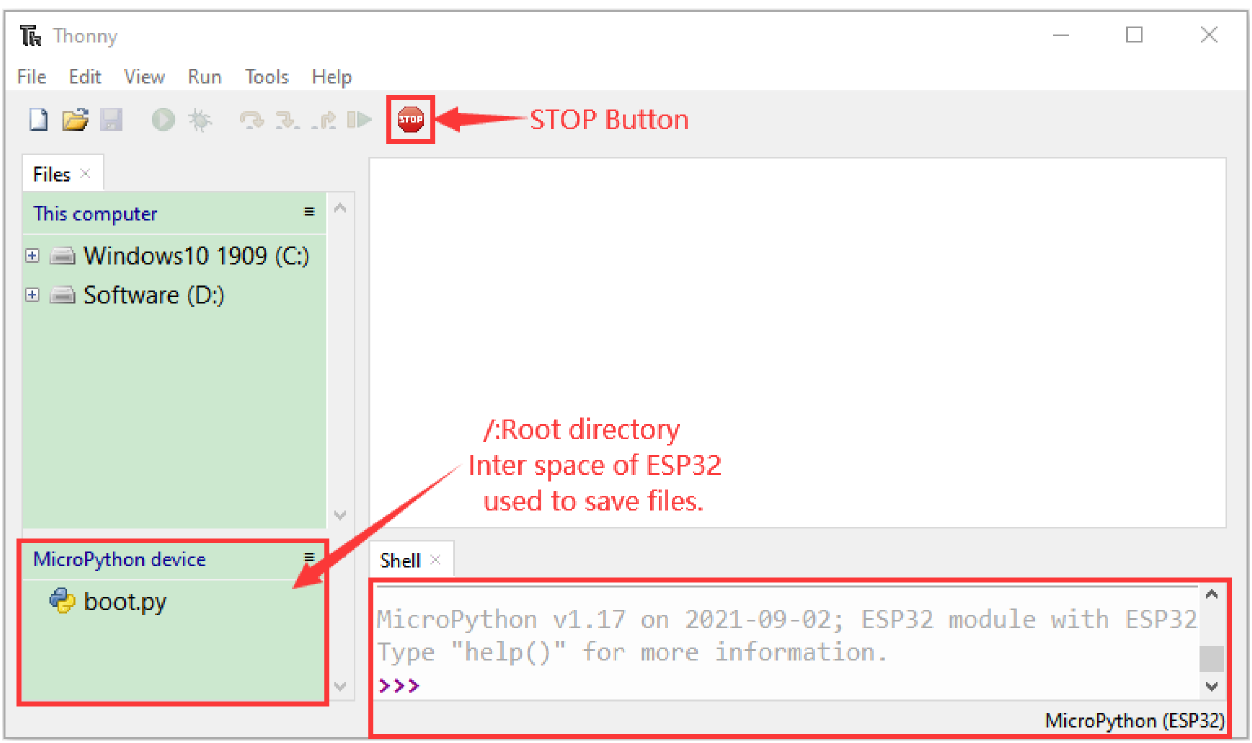
D. Wait for the installation to be done, then click“**Close**”and“**OK**”.


ESP32 firmware installer
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)
Reload
Firmware
on（Important）/Python_Firmware/esp32-20210902-v1.17.bin
Browse...
Flash mode
From image file (keep)
Quad I/O (qio)
Dual I/O (dio)
Dual Output (dout)
Erase flash before installing
Writing at 0x0006521c... (19 %)
Install
Cancel


ESP32 firmware installer
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)
Reload
Firmware
on（Important）/Python_Firmware/esp32-20210902-v1.17.bin
Browse...
Flash mode
From image file (keep)
Quad I/O (qio)
Dual I/O (dio)
Dual Output (dout)
Erase flash before installing
1
Done!
Install
Close



E. Close all dialog boxes, turn to main interface and click "**STOP**". As shown in the illustration below:

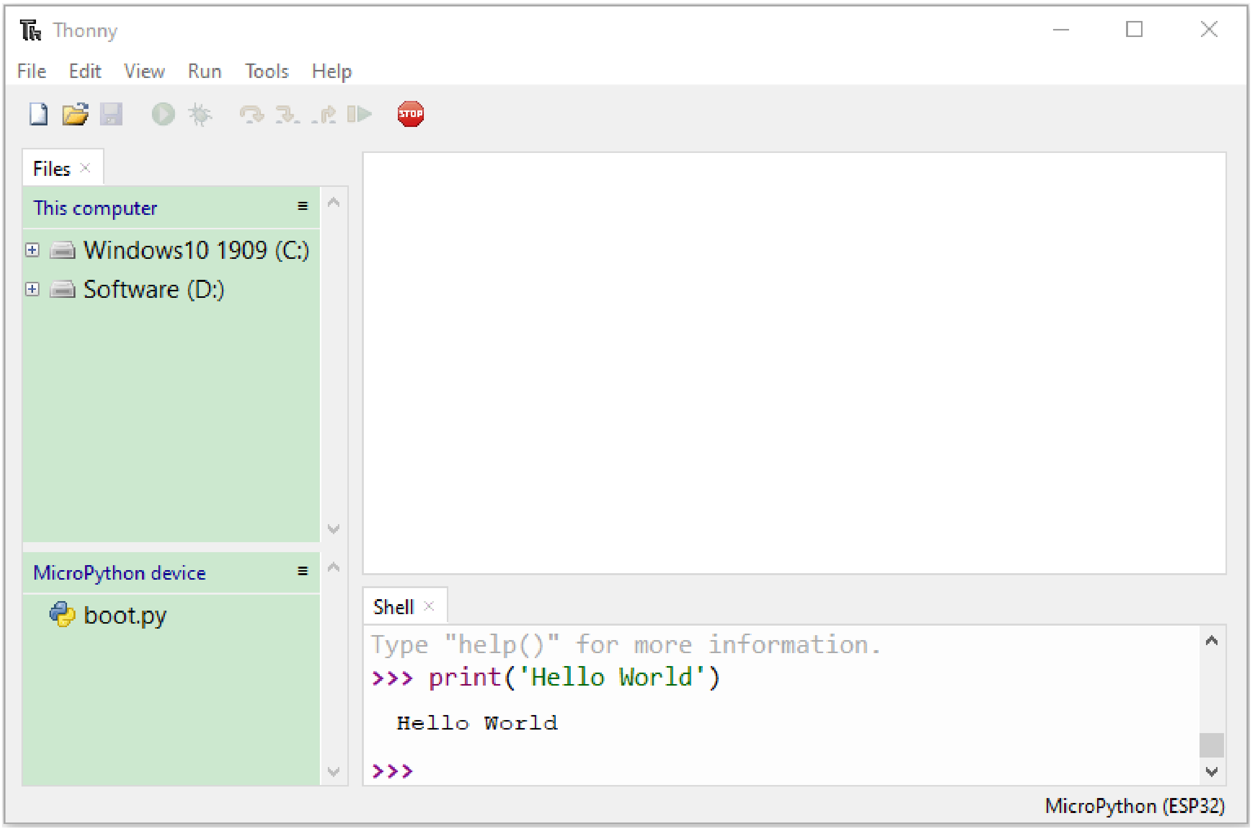


F. So far, all the preparations have been made.

6.5 5.Test Code

Testing the Shell commander

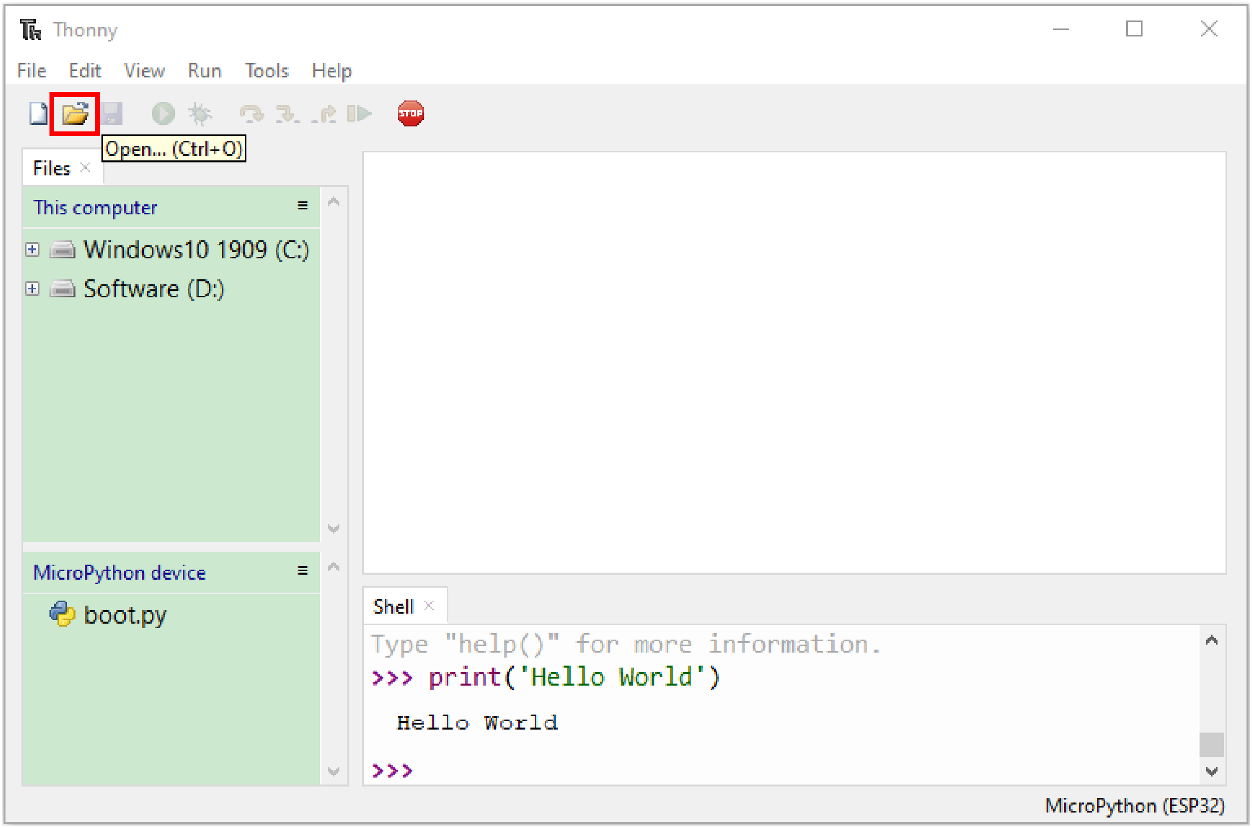
Enter “**print(‘hello world’)**” in “**Shell**” and press Enter.



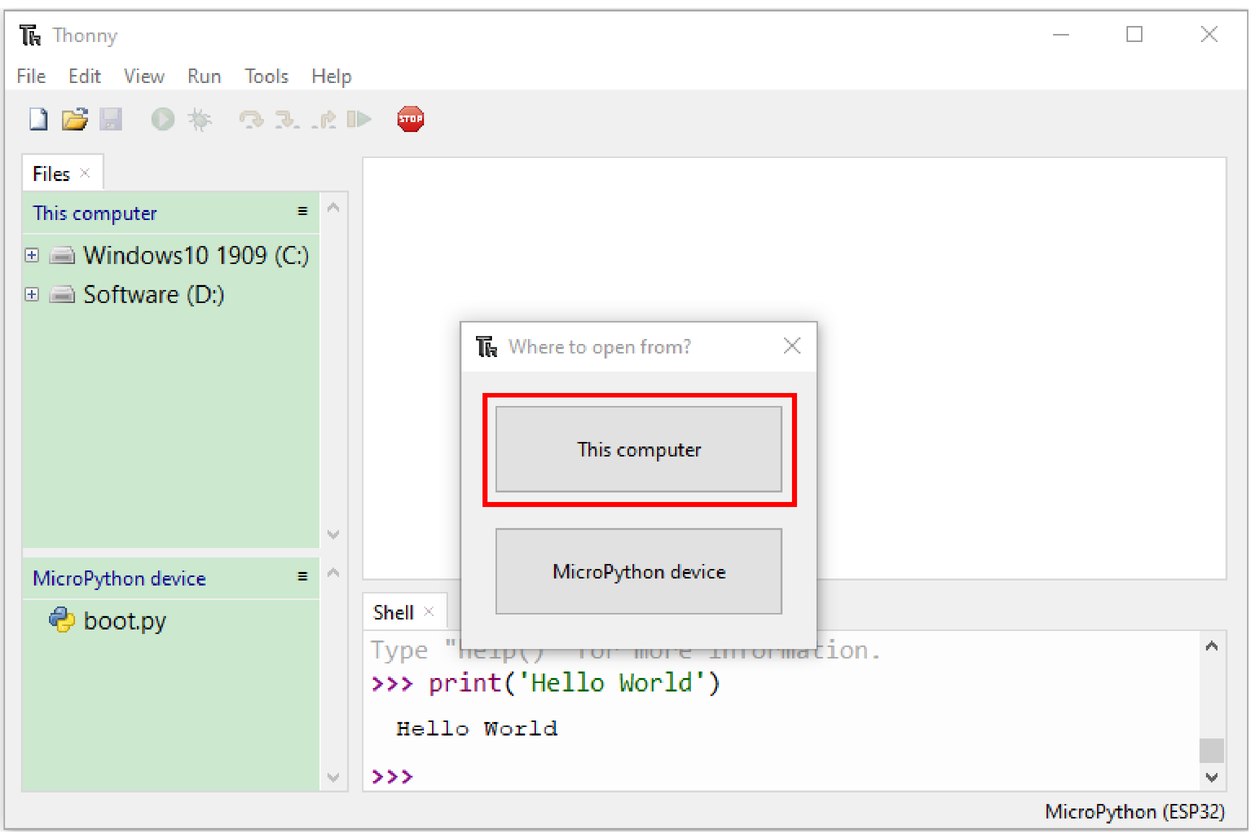
Running Online(Important)

ESP32 needs to be connected to a computer when it is run online. Users can use Thonny to writer and debug programs.

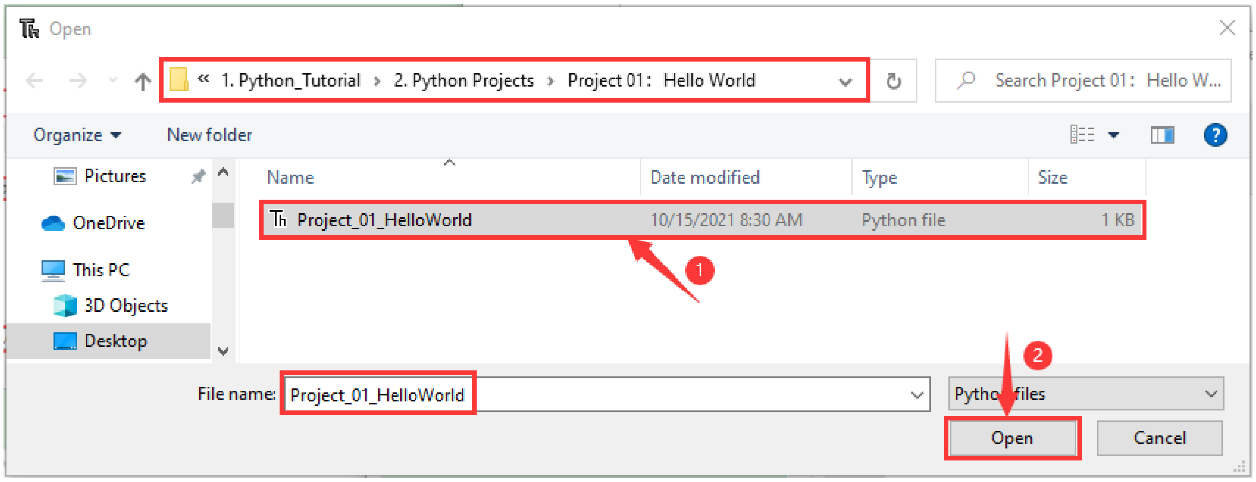
1. Open Thonny and click “**Open. . .** ”.



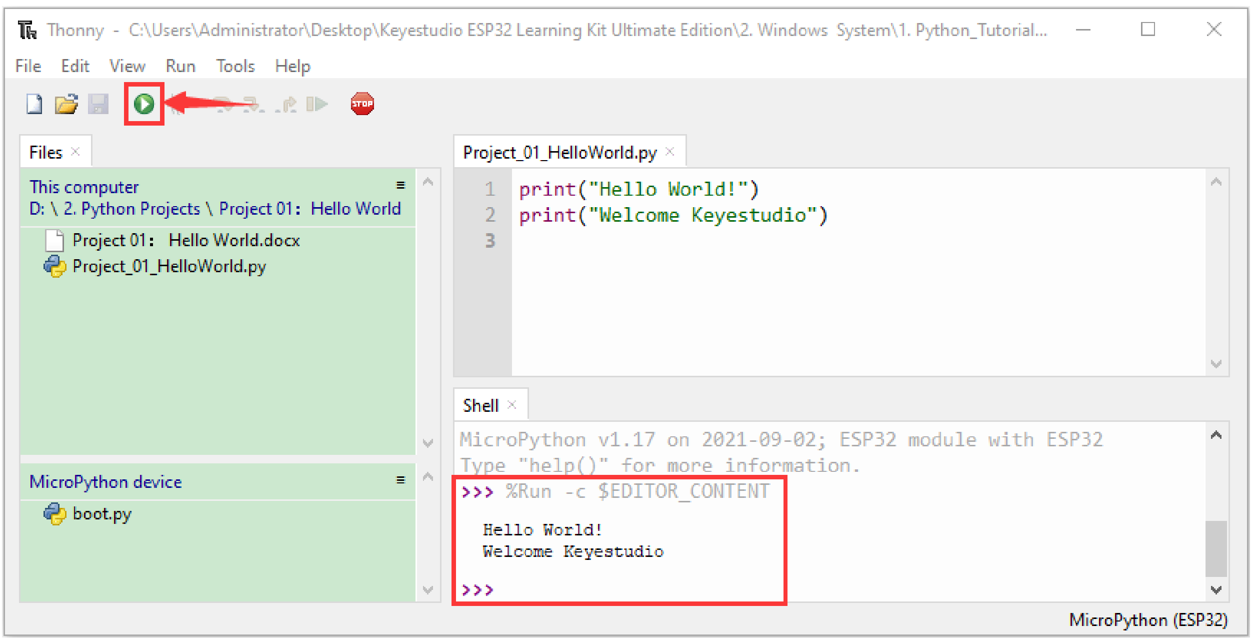
2. Click “**This computer**”on the newly pop-up window.



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.



4. Click "Run current script"to execute the program and "**Hello World**" will be printed in"**Shell**".

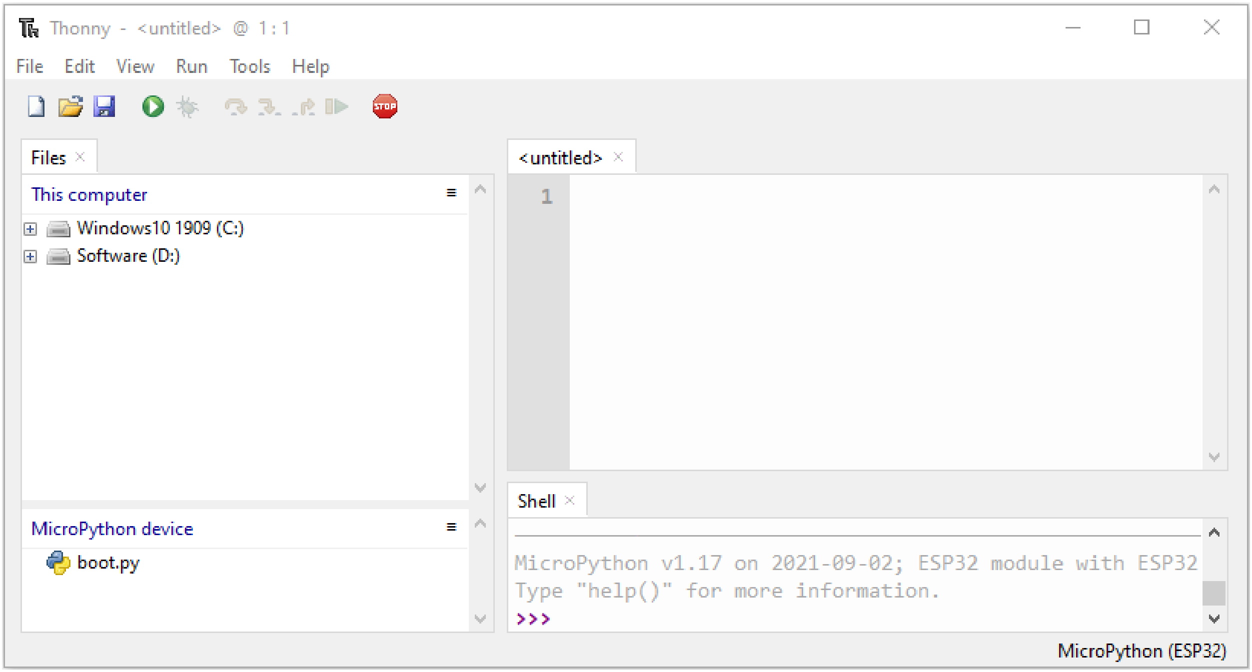


NoteWhen running online, if you press the Reset button of ESP32, user's code will not be executed again. If you wish to run the code automatically after resetting the code, please refer to the following Running Offline.

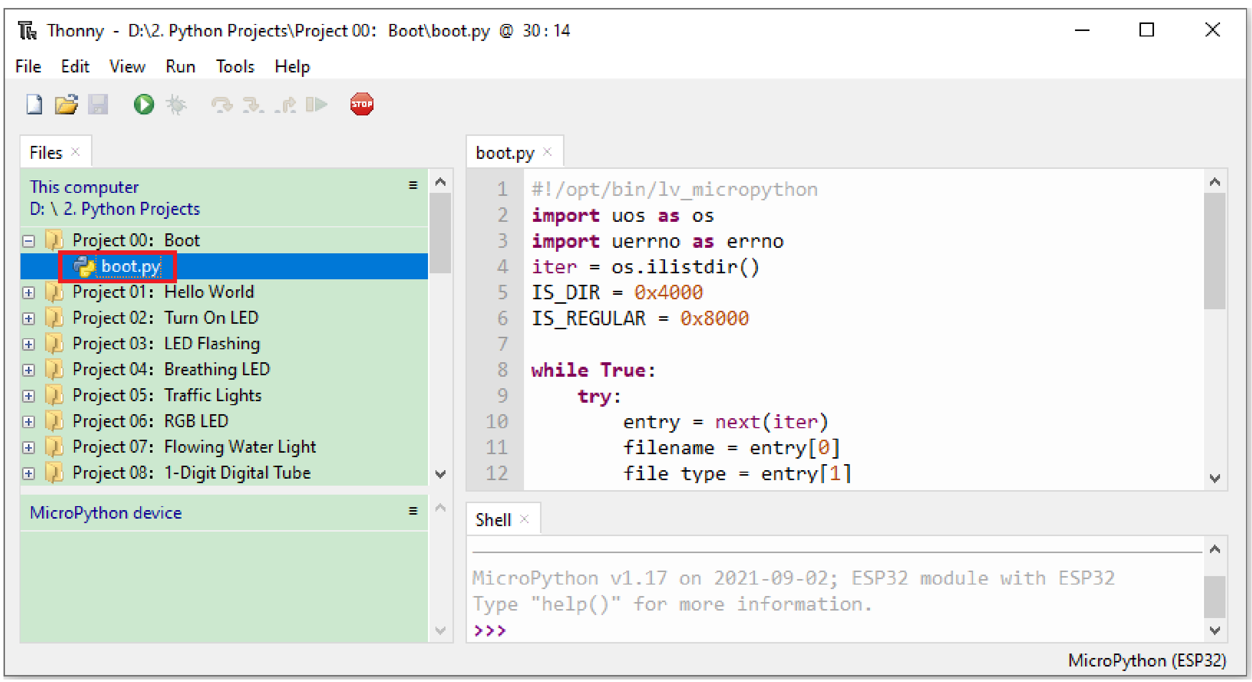
Running Offline(Important)

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.

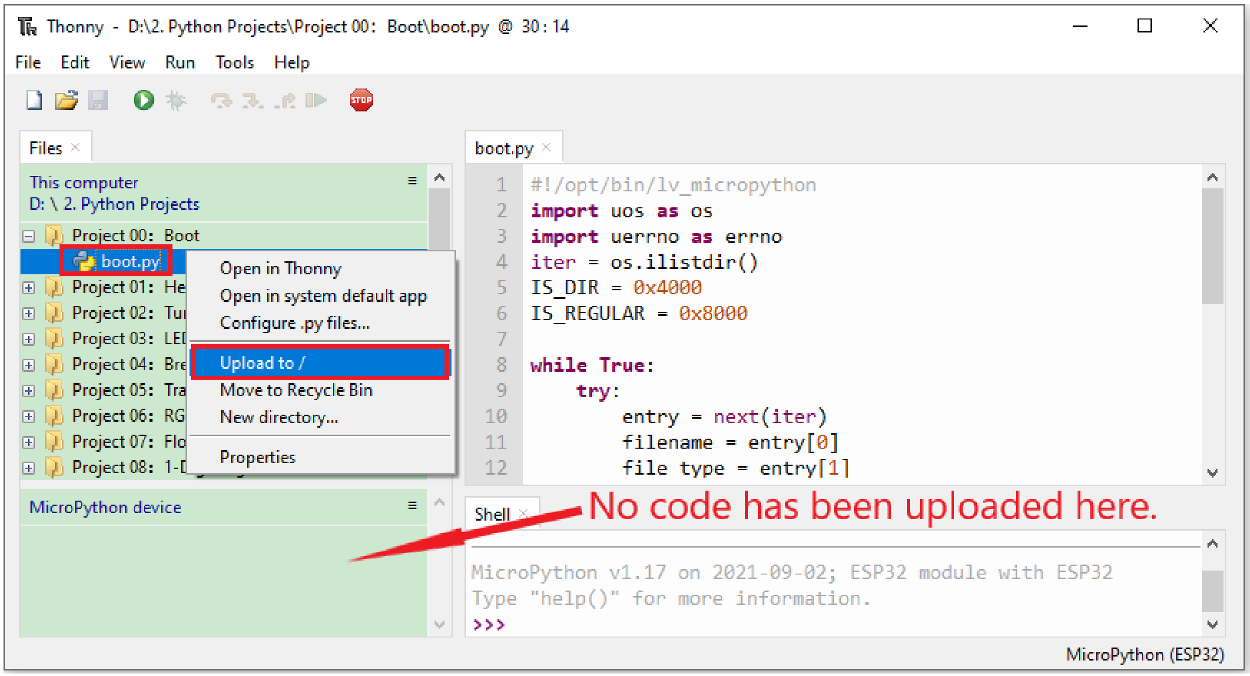
1. Move the program folder "2. Windows System\1. Python_Tutorial\2. Python Projects" to disk(D) in advance with the path of "D:/2. Python Projects". Open "**Thonny**".



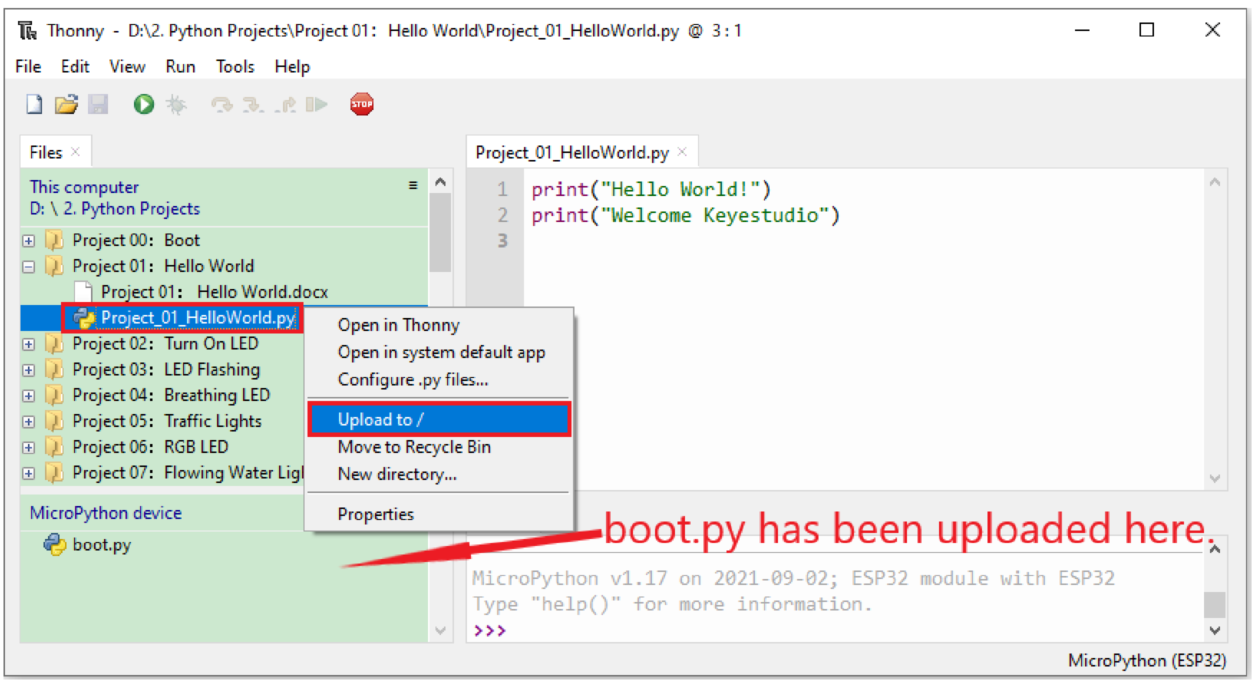
2. Expand "**Project 00: Boot**" in the "**2. Python Projects**" in the directory of disk(D), and double-click "**boot.py**", which is provided by us to enable programs in"**MicroPython device**" to run offline.



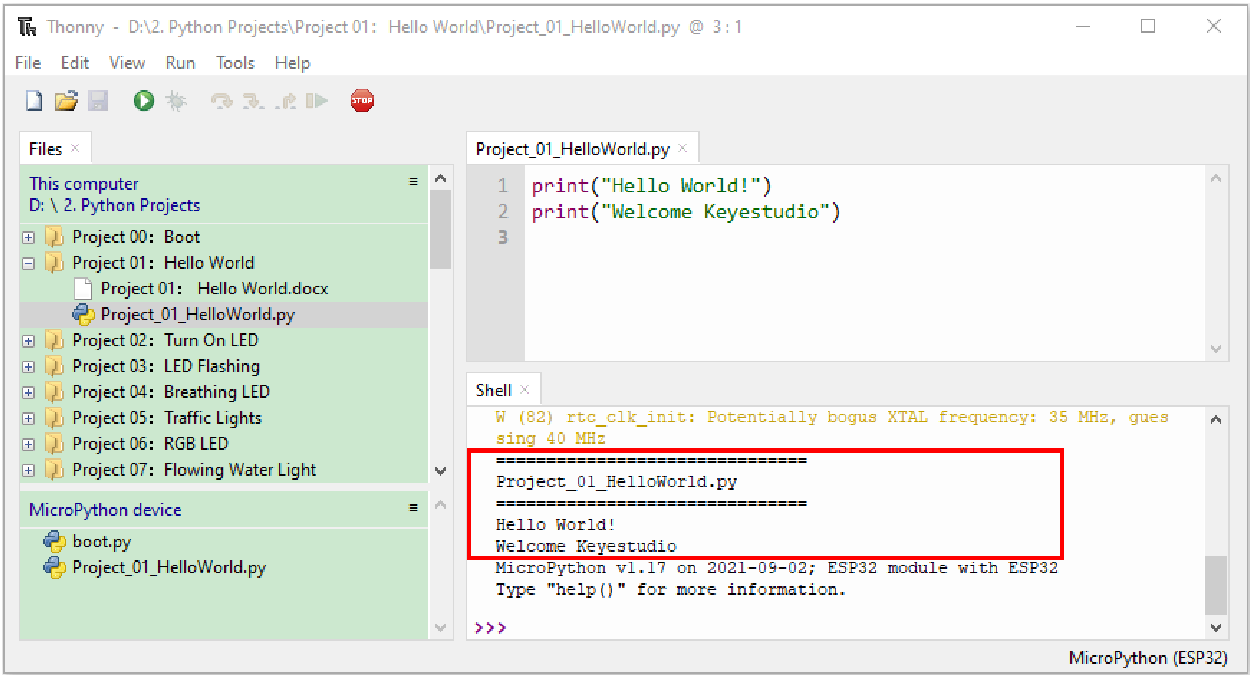
3. If you want your written programs to run offline, you need to upload "**boot.py"** we provided and all your codes to "MicroPython device" and press ESP32's Reset button. Here we use programs **Project 00** and **Project 01** as examples. Select "**boot.py**", right-click to select "**Upload to /**".



4. Similarly, upload"**Project_01_HelloWorld.py**" to"**MicroPython device**".



5. Press ESP32's Reset button and in the box of the "**Shell**" below, you can see the code is executed.

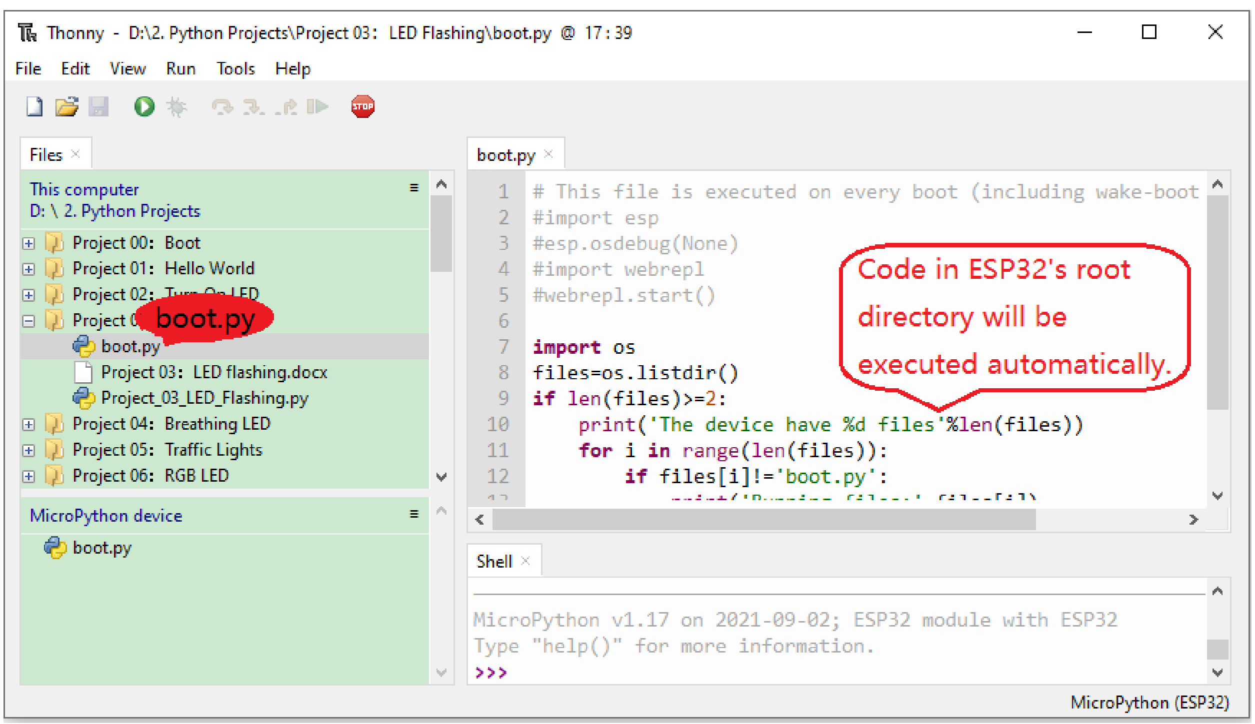


6.6 6.Thonny Common operations

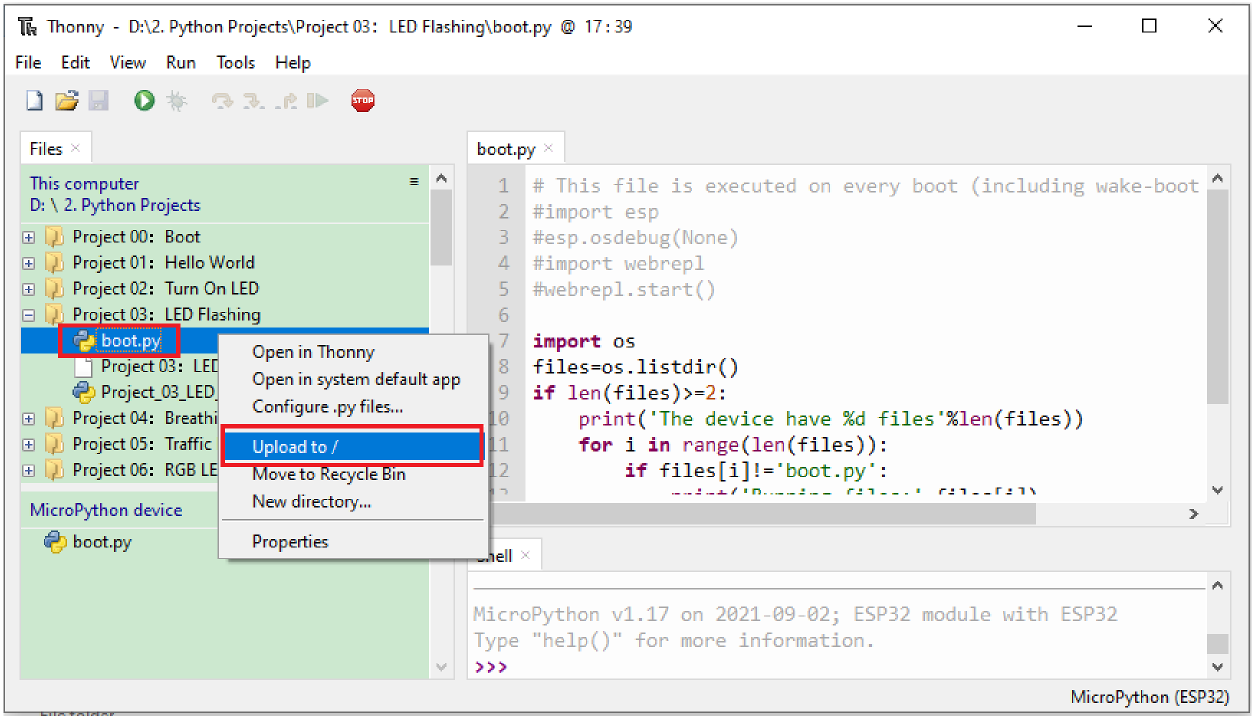
Upload the code to ESP32

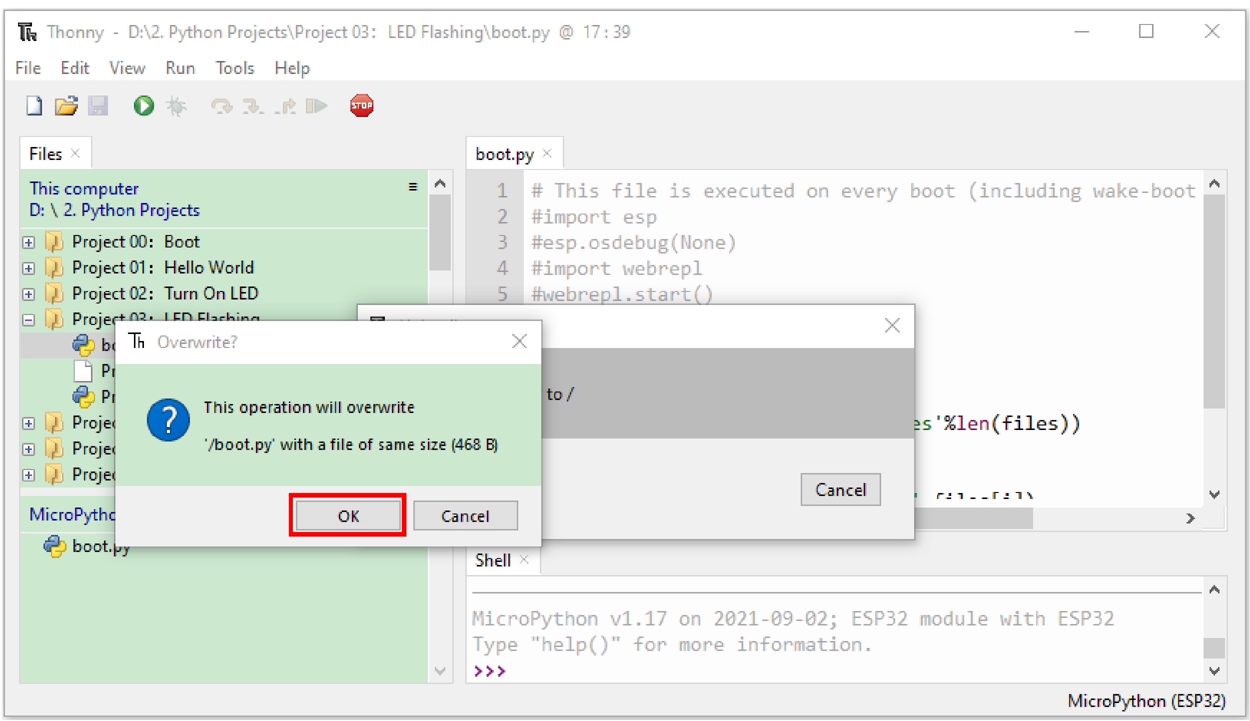
For convenience, let's take the opertation on "**boot.py**" as an example.

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.



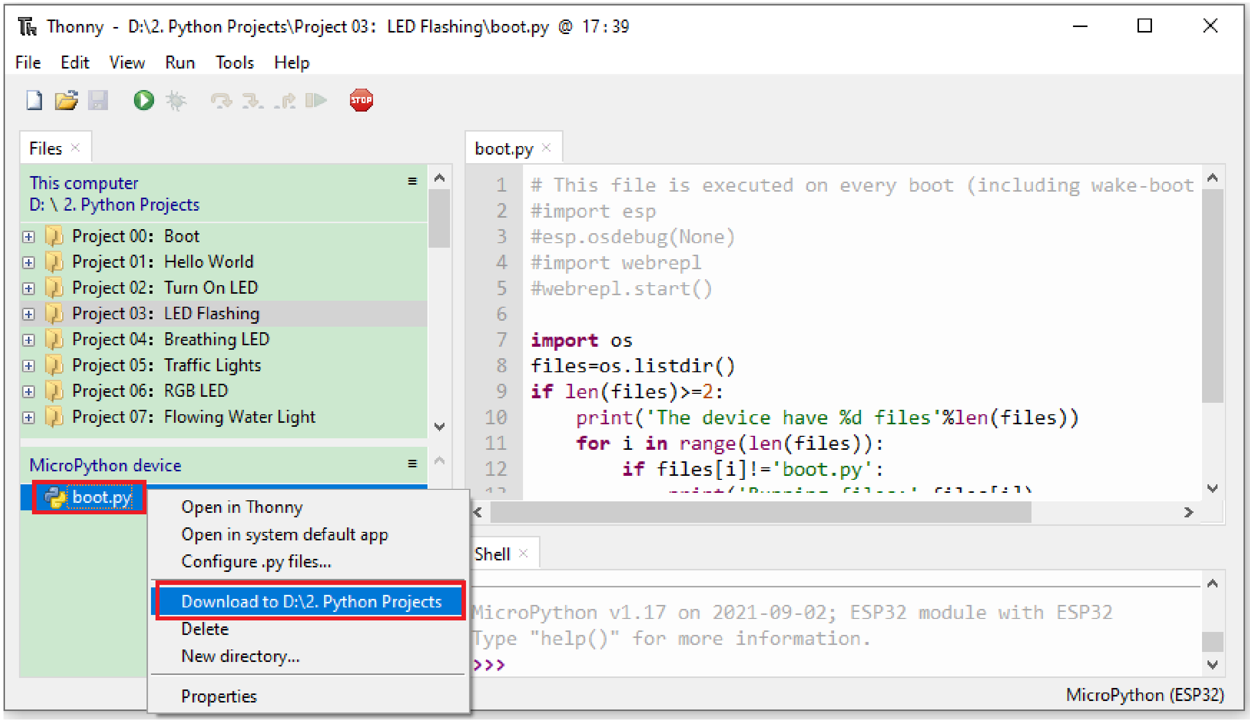
Select "**boot.py**" in "**Project 03LED Flashing**", right-click your mouse and select "**Upload to /**" to upload code to ESP32's root directory, select to click "**OK**".





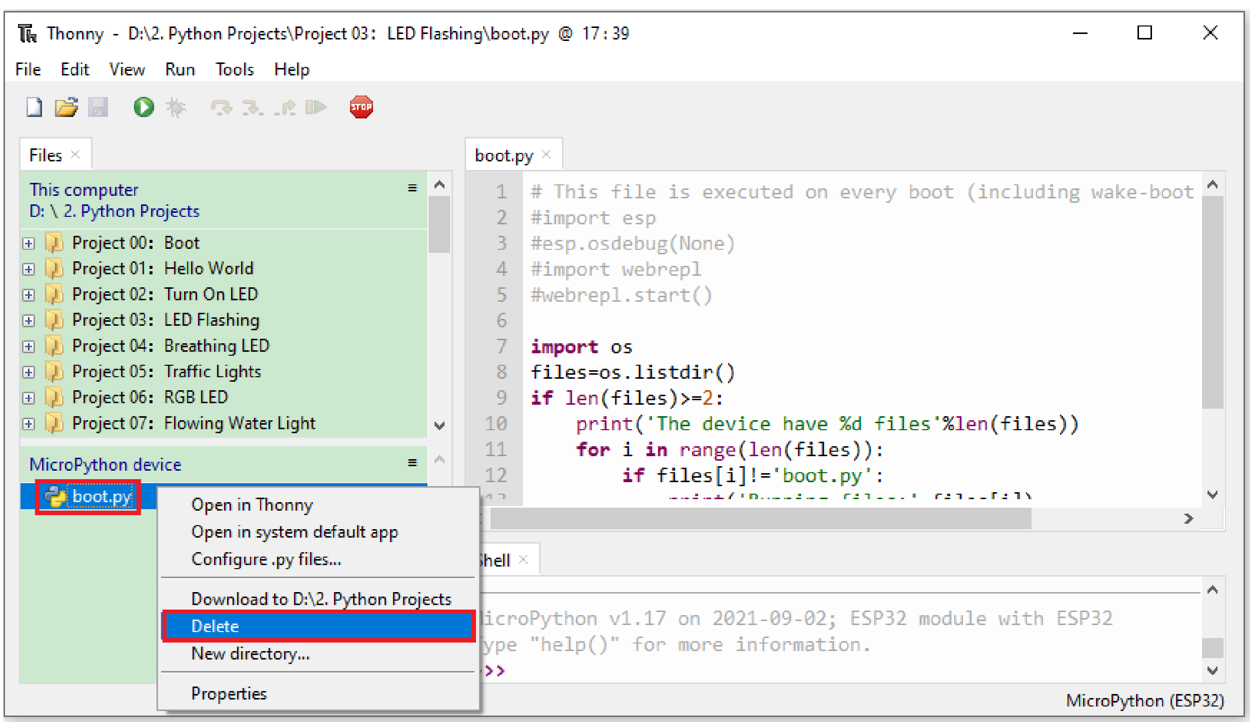
Downloading Code to Computer

Select "**boot.py**" in "**MicroPython device**", right-click to select "**Download to ...**" to download the code to your computer.

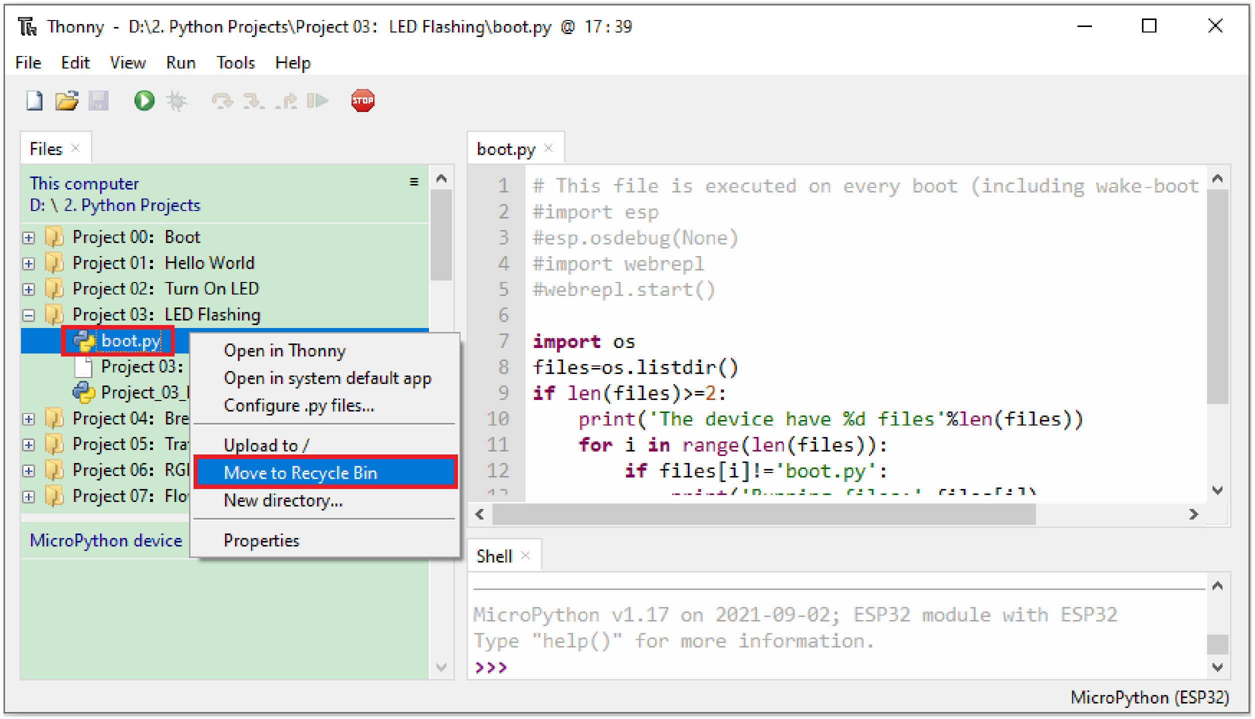


Delete files from ESP32 Root Directory

Select "**boot.py**" in "**MicroPython device**", right-click it and select "**Delete**" to delete "**boot.py**" from ESP32's root directory.

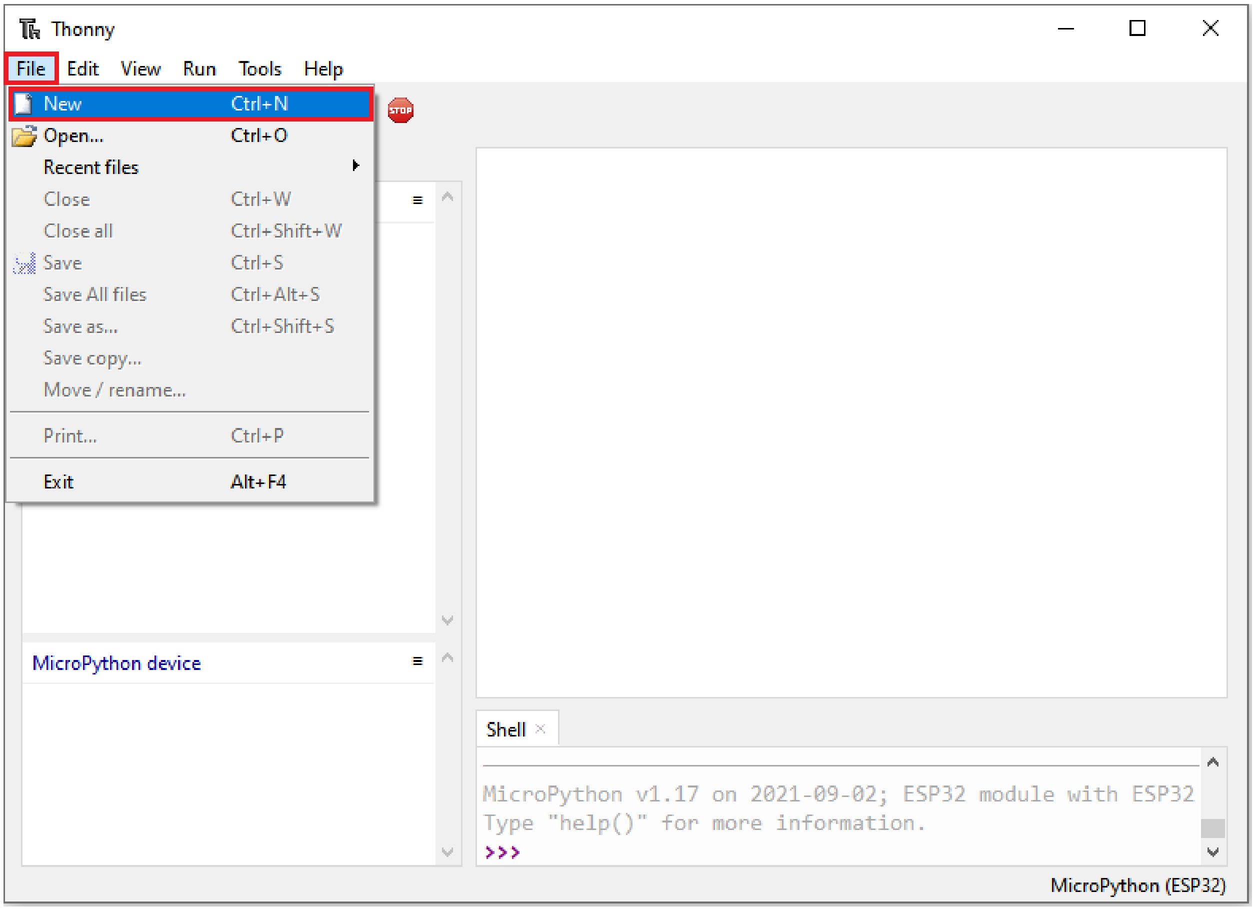


Select "**boot.py**" in "**Project 03LED Flashing**", right-click it and select "**Move to Recycle Bin**" to delete it from "**Project 03LED Flashing**".

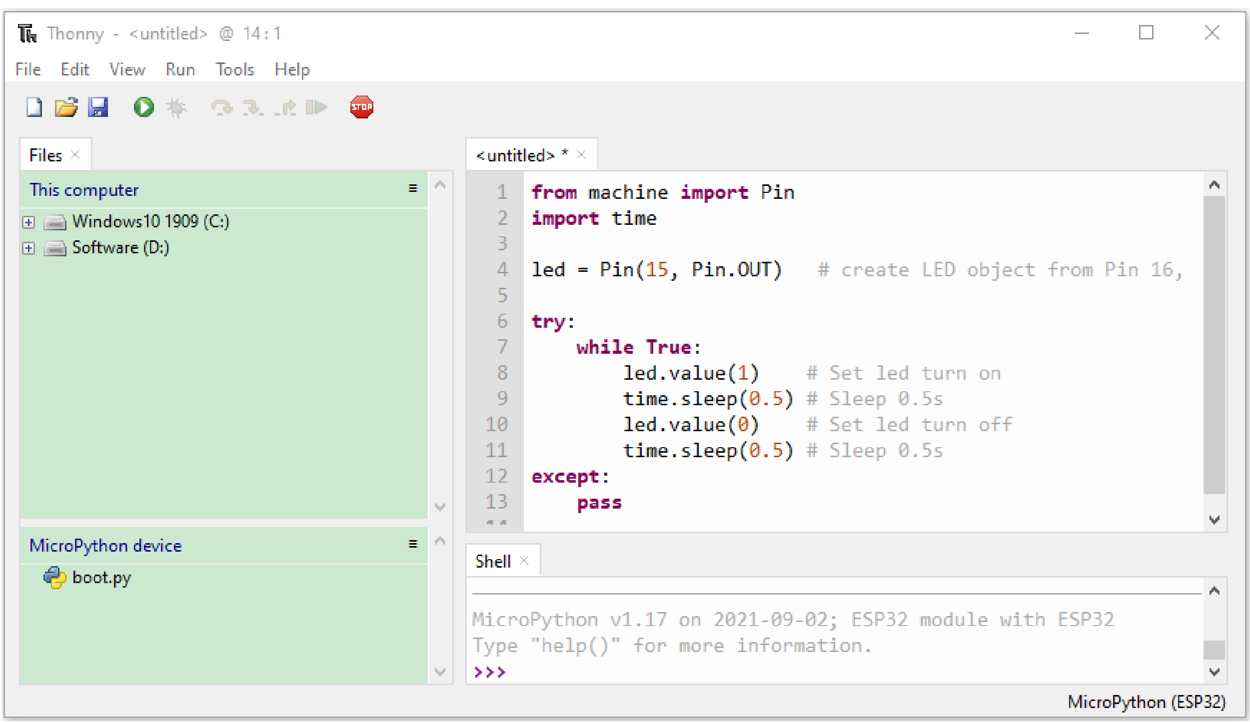


Creating and Saving the code

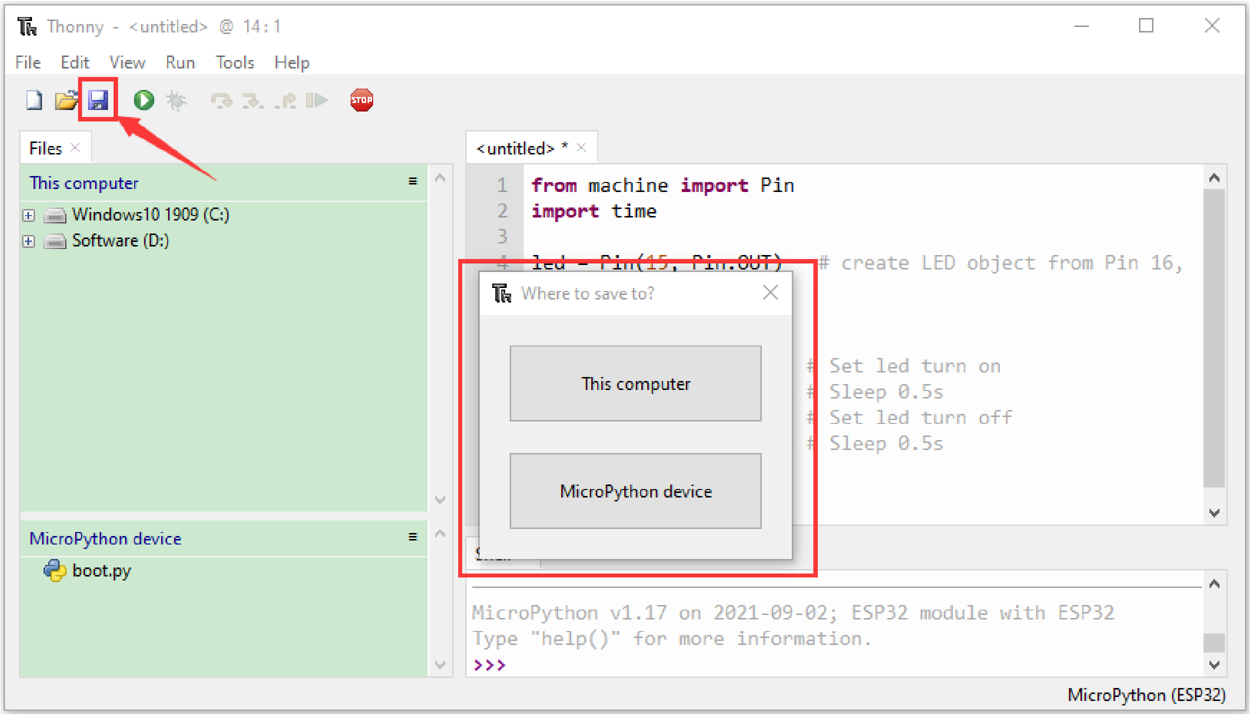
Click “**File**” → “**New**” to create and write codes.



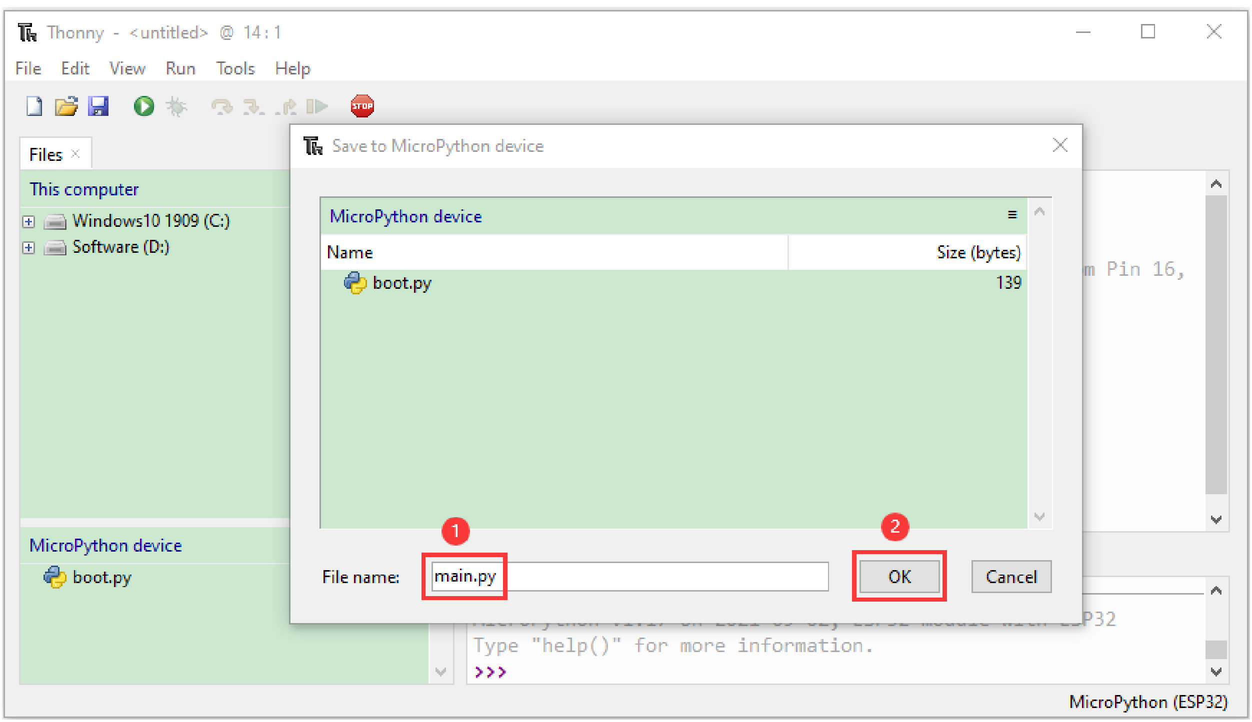
Enter codes in the newly opened file. Here we use codes of "**Project_03_LED_Flashing.py**" as an example.



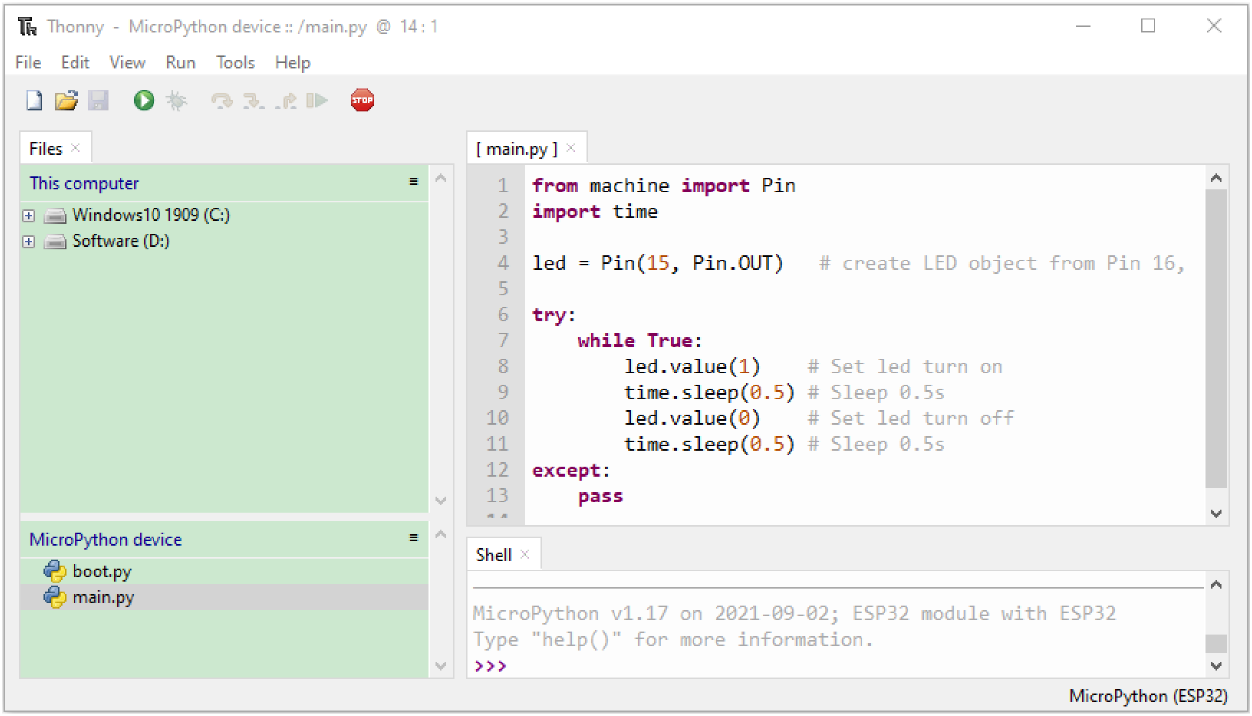
Click “**Save**”on the menu bar, you can save the codes either to your computer or to ESP32.



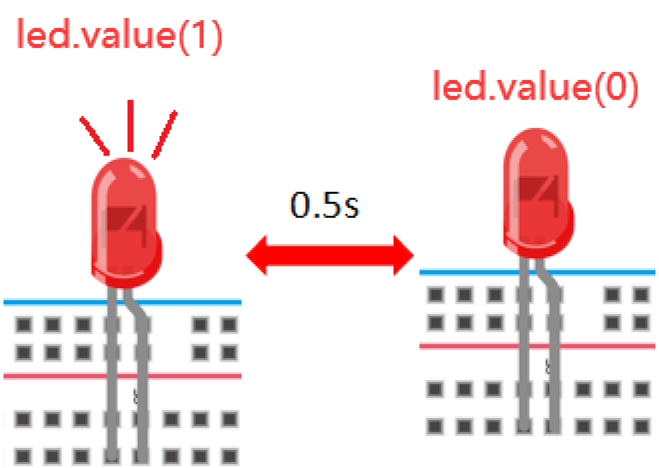
Select “**MicroPython device**”, enter “**main.py**” in the newly pop-up window and click “**OK**”.



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

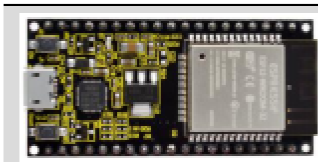
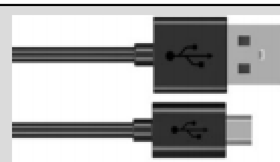
Click on the link to download the code file: Download Python Codes file

7.2 Project 01: Hello World

7.2.1 1. Introduction

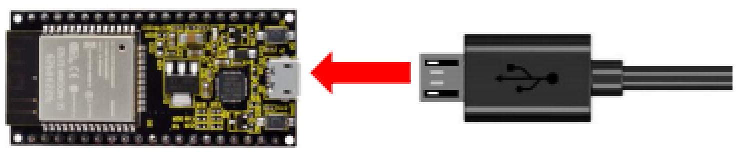
For ESP32 beginners, we'll start with some simple things. In this project, you just need an ESP32 mainboard, USB cable and computer to complete "Hello World!" Project. It is not only a communication test for ESP32 mainboard and computer, but also a primary project for ESP32.

7.2.2 2. Components

	
ESP32*1	USB cable*1

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.

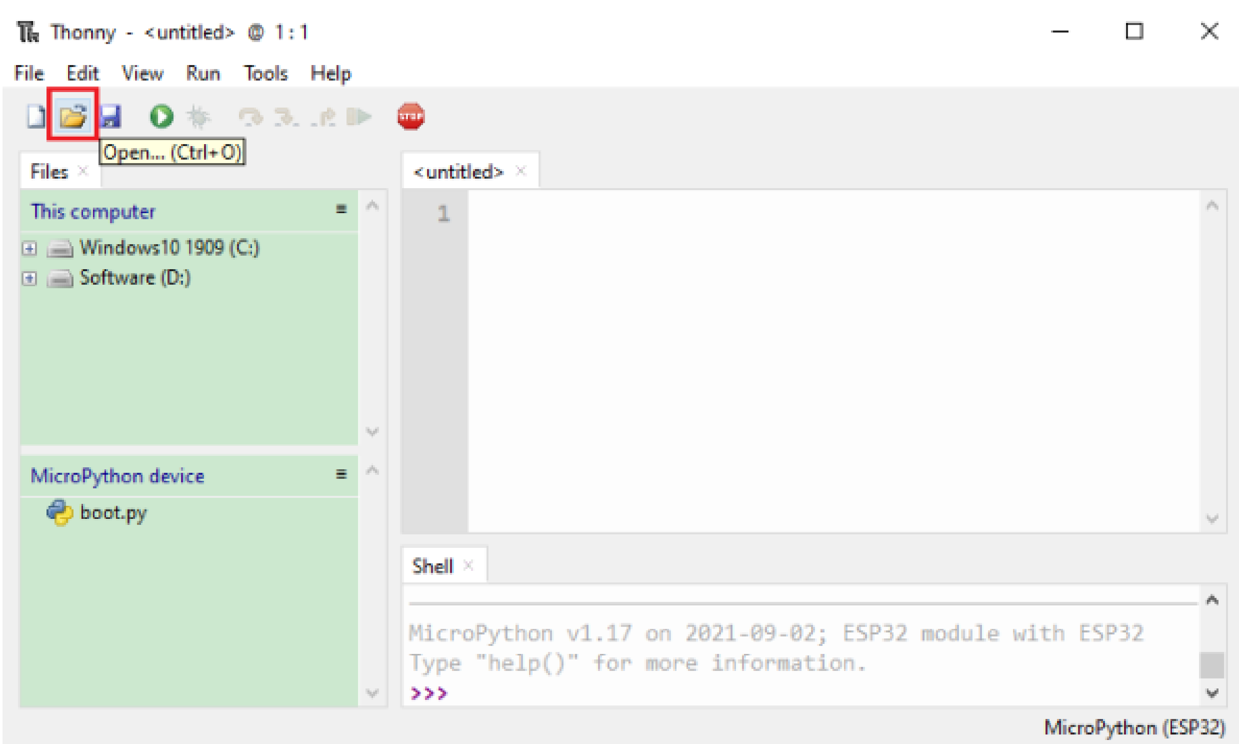
Disadvantages

(1) To run ESP32 online, you have to be connected to a computer and run with Thonny.

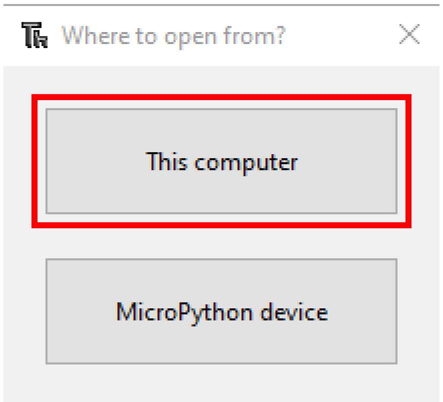
(2) If ESP32 disconnects from computer, the program won't run again when they reconnect to each other.

Basic Operation:

1) Open Thonnyand click  "Open…".

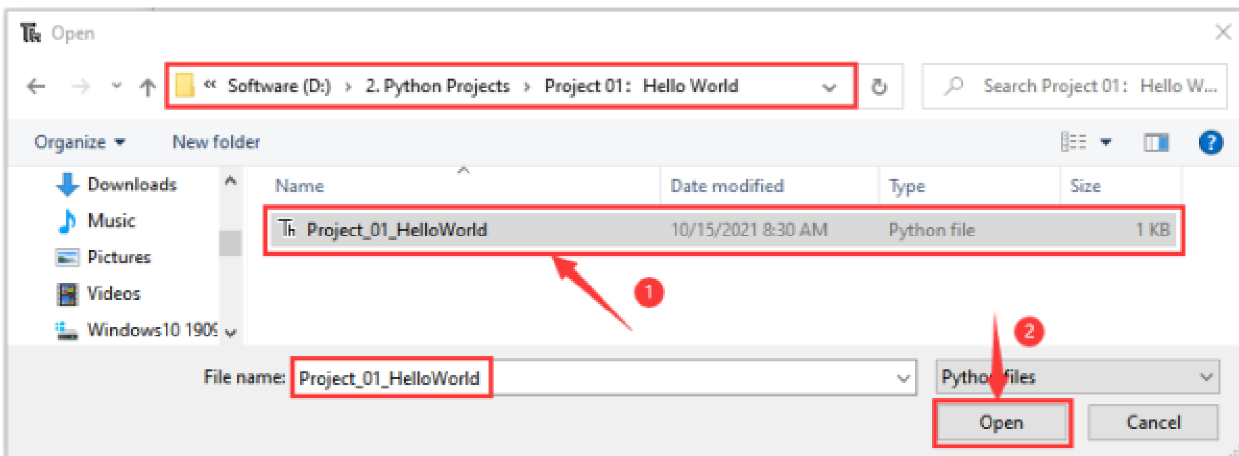


(2) On the newly pop-up window, click"**This computer**".

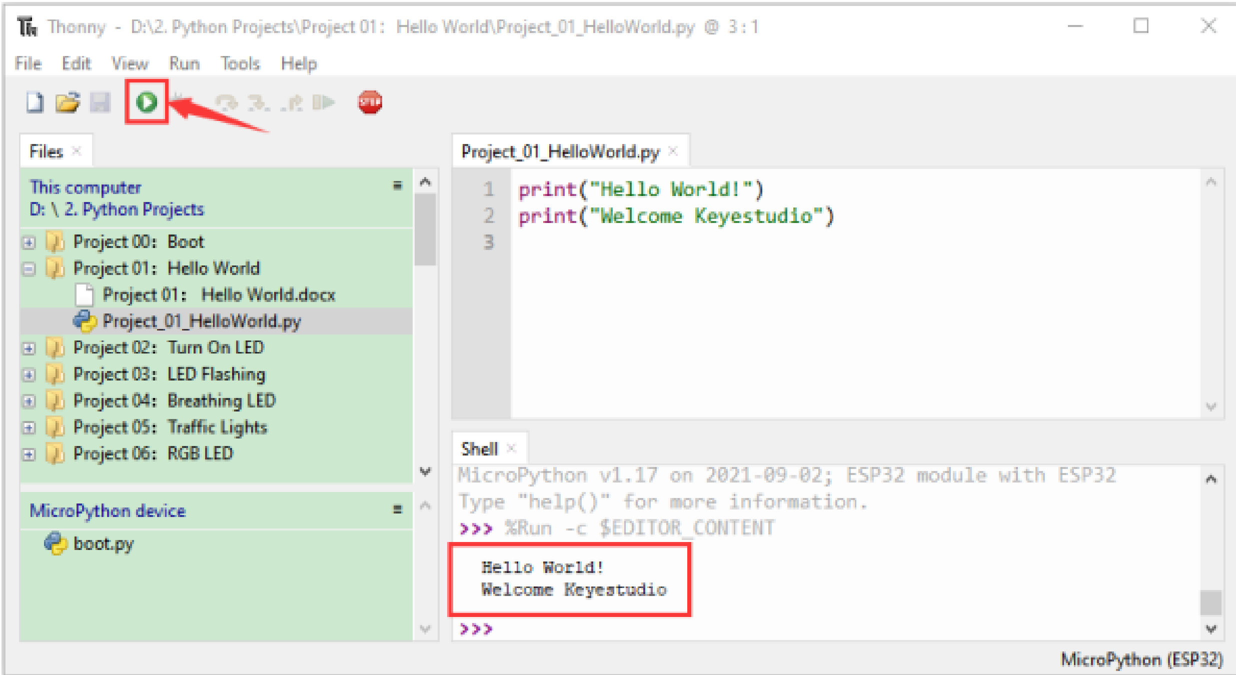


2In the new dialog box, select "Project_01_HelloWorld.py", click "Open".

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

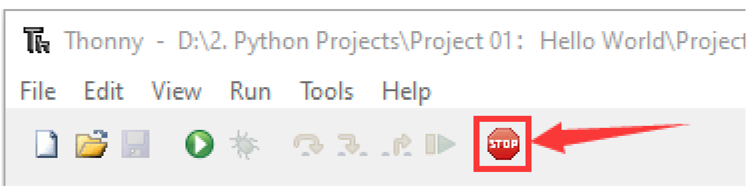


(3) Click  "Run current script" to execute the program "Hello World!", "Welcome Keyestudio", which will be printed in the "**Shell**" window.



Exit running online

When running online, click "**Stop /Restart Backend**" or press "Ctrl+C" to exit the program.



7.2.5 5. Project code

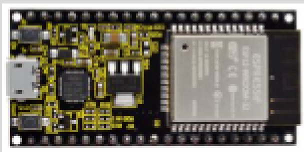
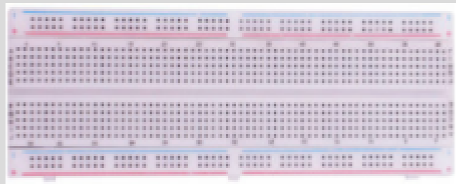
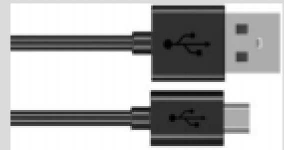
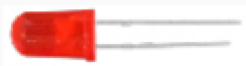
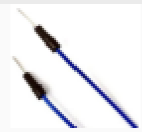
```
print("Hello World!")
print("Welcome Keyestudio")
```

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.

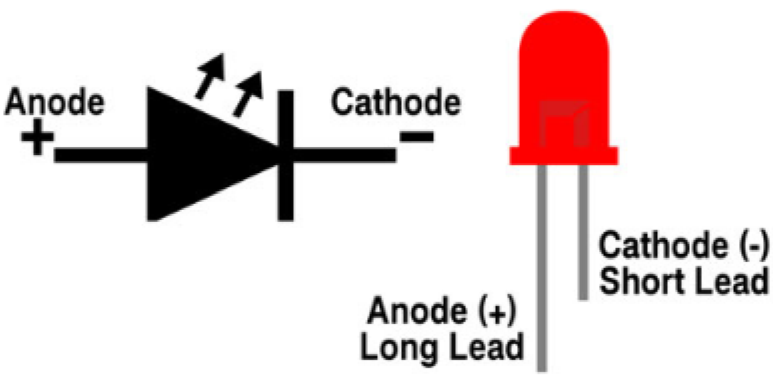
7.3.2 2.Components

		
ESP32*1	Breadboard*1	USB Cable*1
		
Red LED*1	220 Resistor*1	Jumper Wire*2

7.3.3 3.Component knowledge

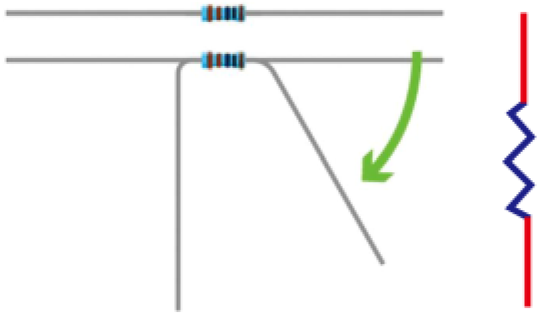
1LED:

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 to3.3V 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.

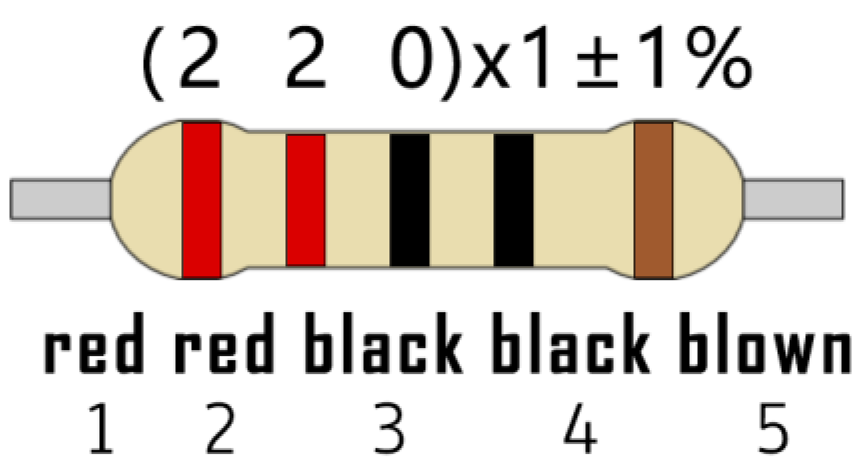
-Color 4 – Multiplier.

-Color 5 – Tolerance.

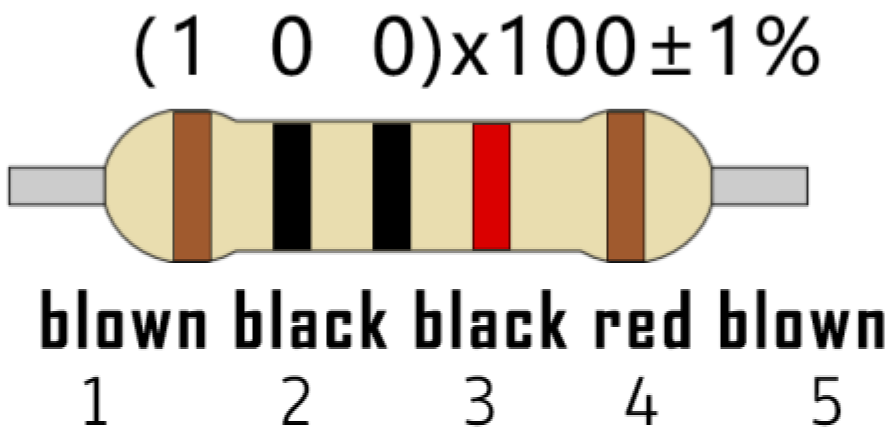
	1st Digit	2nd Digit	3rd Digit	Multiplier	Tolerance
Black		0	0	x1	
Brown	1	1	1	x10	± 1%
Red	2	2	2	x100	± 2%
Orange	3	3	3	x1K	± 3%
Yellow	4	4	4	x10K	± 4%
Green	5	5	5	x100K	± 0.5%
Blue	6	6	6	x1M	± 0.25%
Violet	7	7	7	x10M	± 0.10%
Grey	8	8	8	x100M	± 0.05%
White	9	9	9	x1G	
Gold				÷ 10	± 5%
Silver				÷ 100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

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

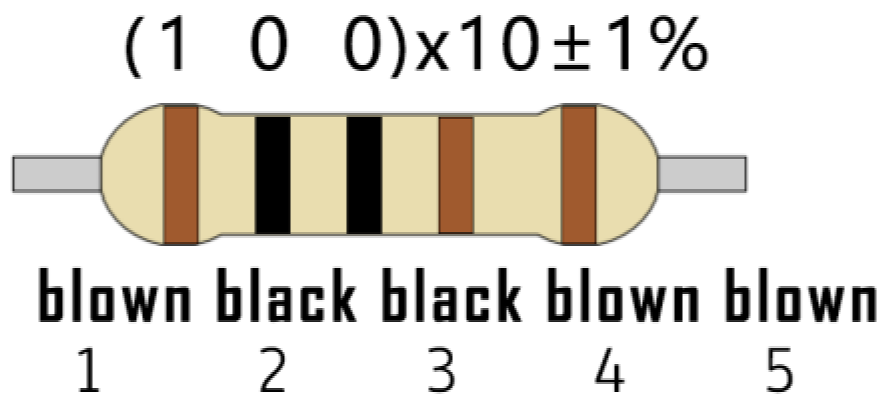
220 Resistor*10



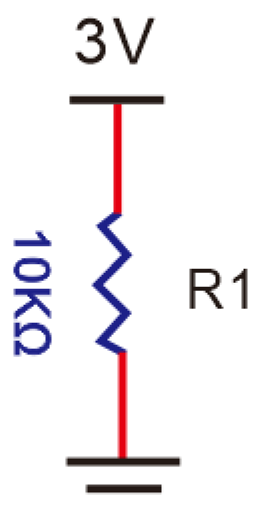
10K Resistor*10



1K Resistor*10



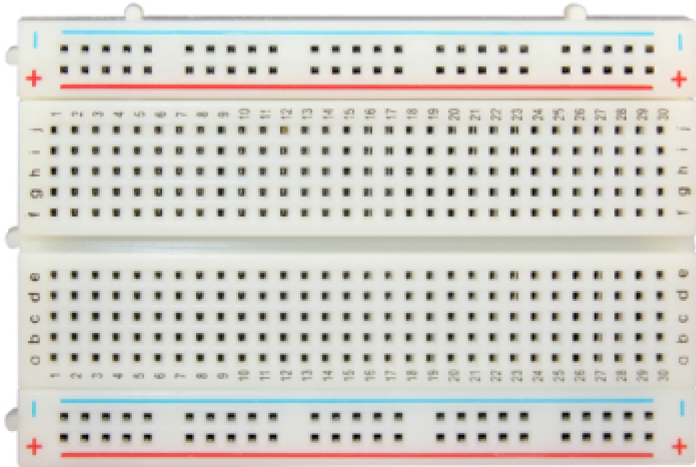
In the same voltage, there will be less current and more resistance. The connection between current(I), voltage(V), and resistance® can be expressed by the formula: I=U/R. In the figure below, if the voltage is 3V, the current through R1 is: I = U / R = 3 V / 10 K= 0.0003A= 0.3mA.



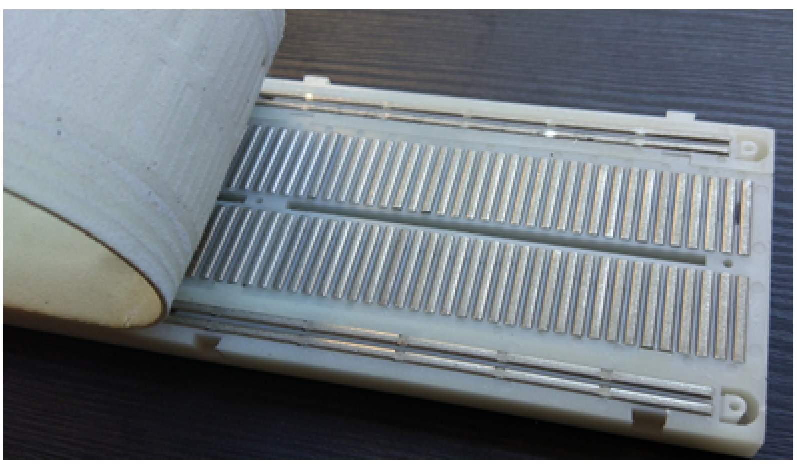
Don't connect a low resistance directly to the two poles of the power supply. as this will cause excessive current to damage the electronic components. Resistors do not have positive and negative poles.

3Breadboard

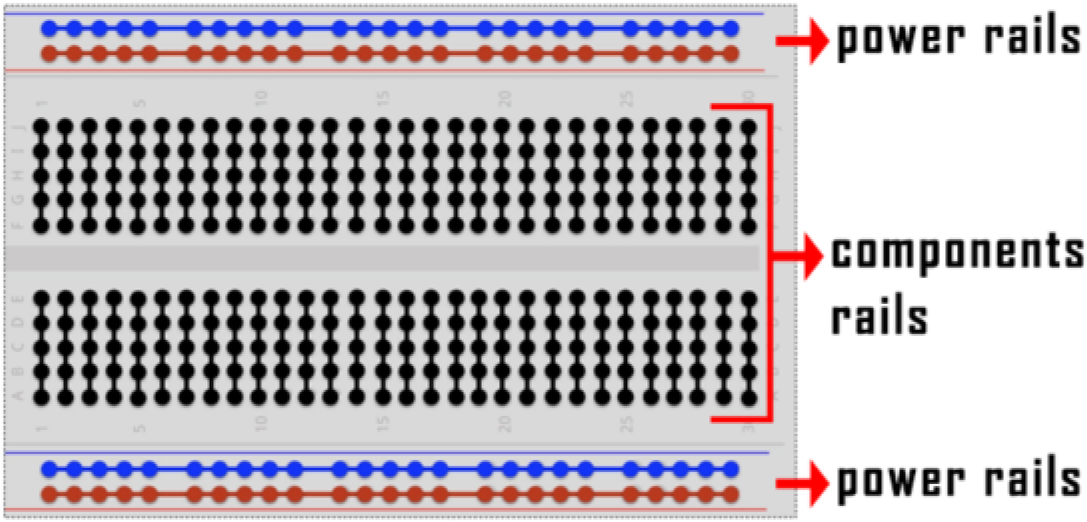
Breadboards are used to build and test circuits quickly before completing any circuit design. There are many holes in the breadboard that can be inserted into circuit components such as integrated circuit board and resistors. A typical breadboard is shown below



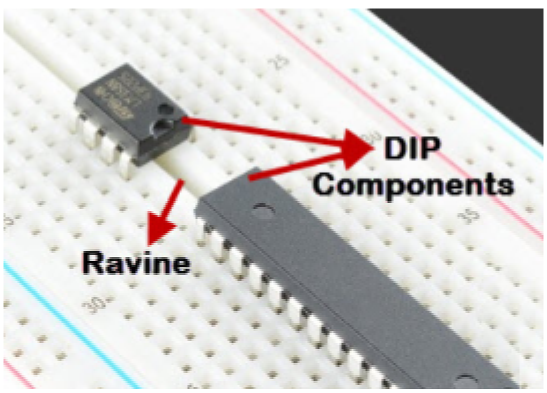
The breadboard has strips of metal , which run underneath the board and connect the holes on the top of the board. The metal strips are laid out as shown below. Note that the top and bottom rows of holes are connected horizontallywhile the remaining holes are connected vertically.

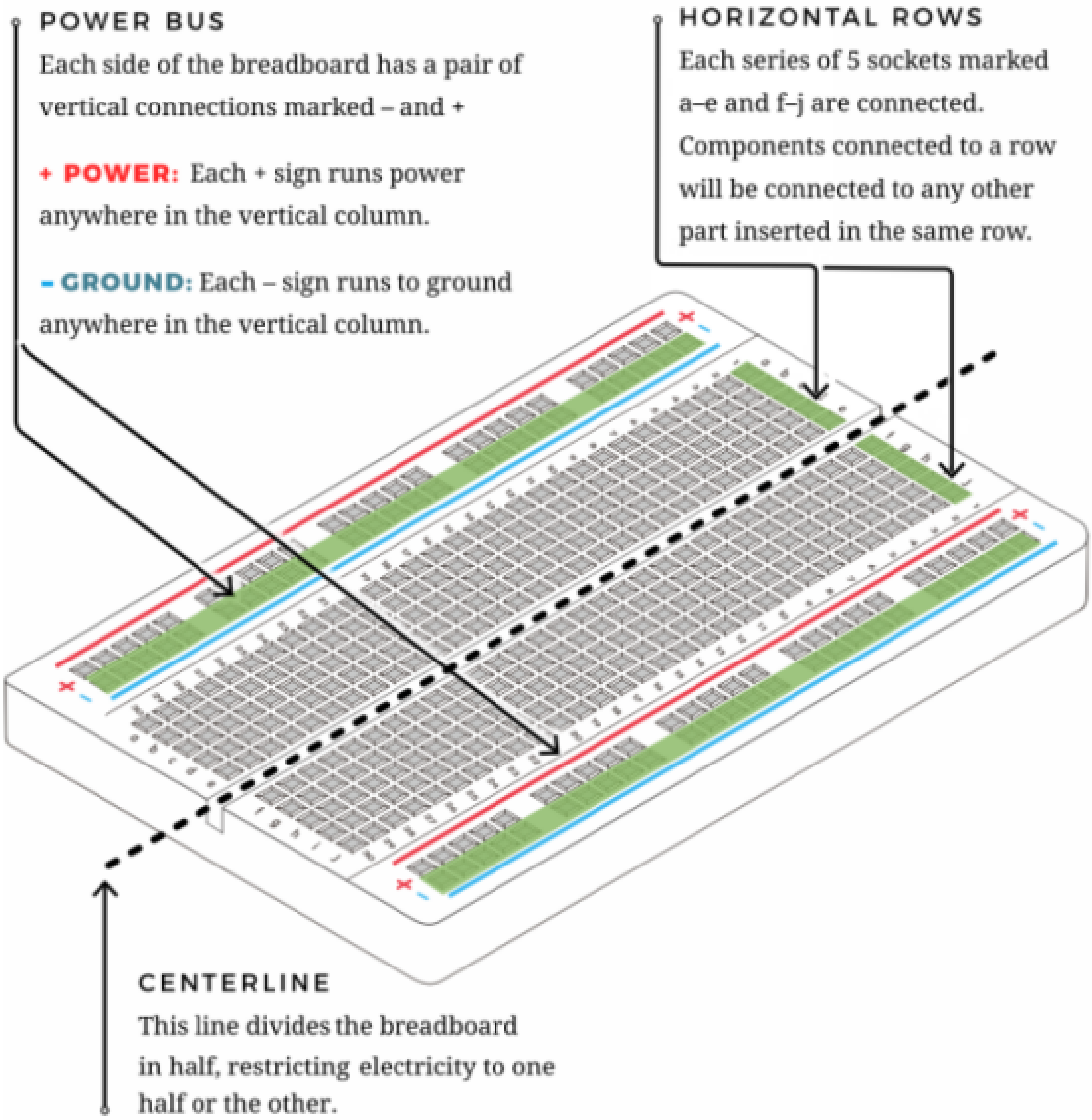


The first two rows (top) and the last two rows (bottom) of the breadboard are used for the positive pole (+) and negative pole (-) of the power supply respectively. The conductive layout of the breadboard is shown in the figure below:



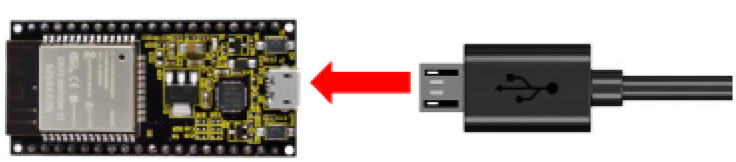
When we connect DIP (Dual In-line Packages) components, such as integrated circuits, microcontrollers, chips and so on, we can see that a groove in the middle isolates the middle part, so the top and bottom of the groove is not connected. DIP components can be connected as shown in the following diagram:





(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.

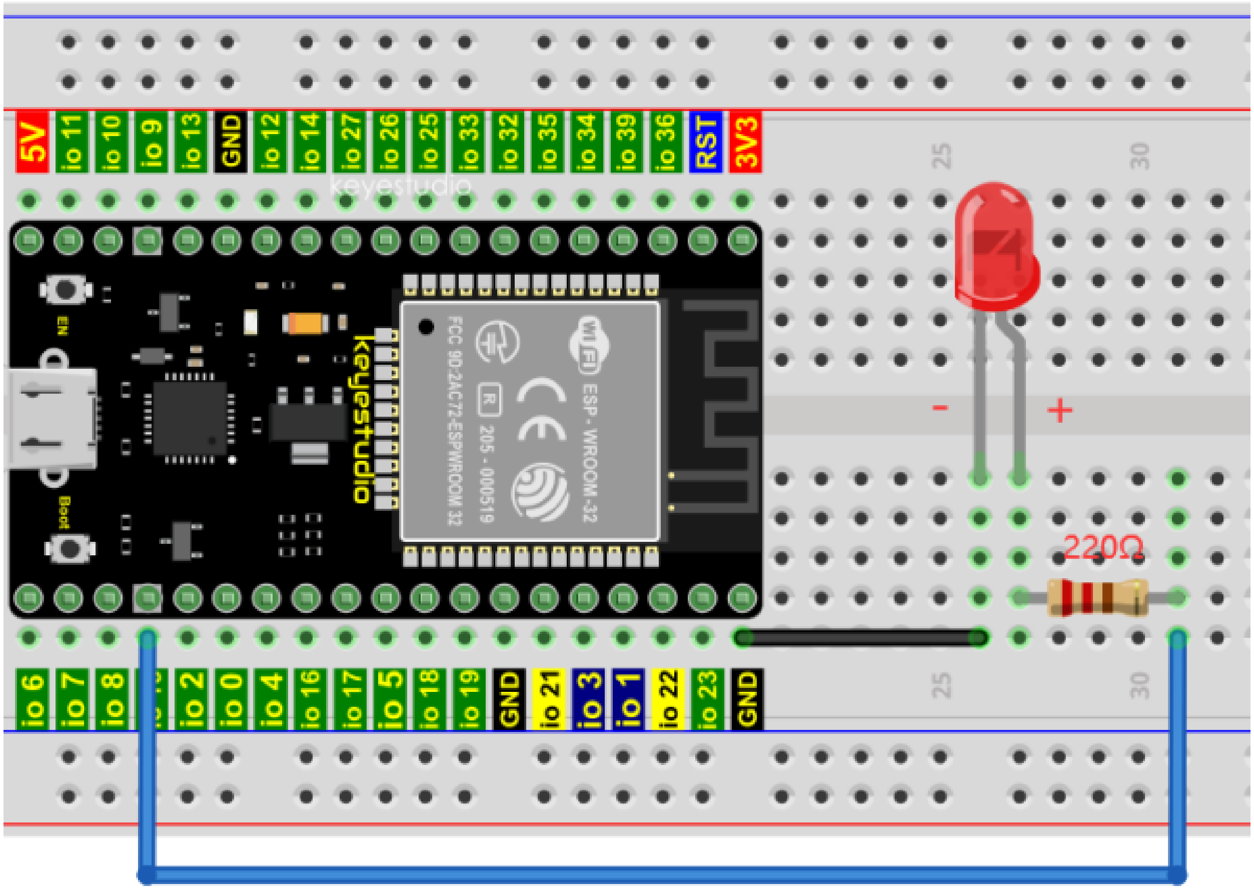


7.3.4 4.Wiring diagram

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.

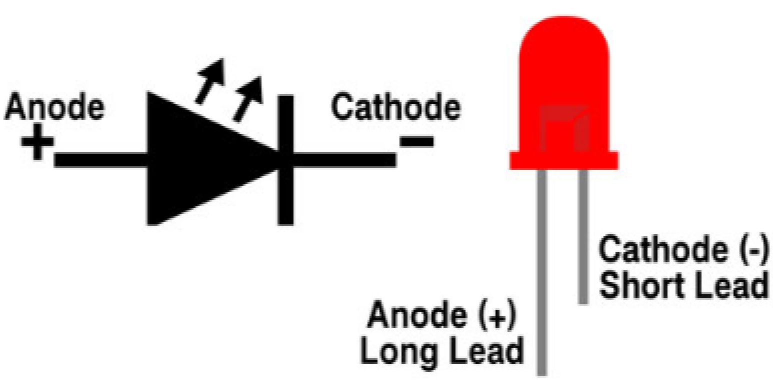
Note: Be careful to avoid short circuit when connecting 3.3V and GND!

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

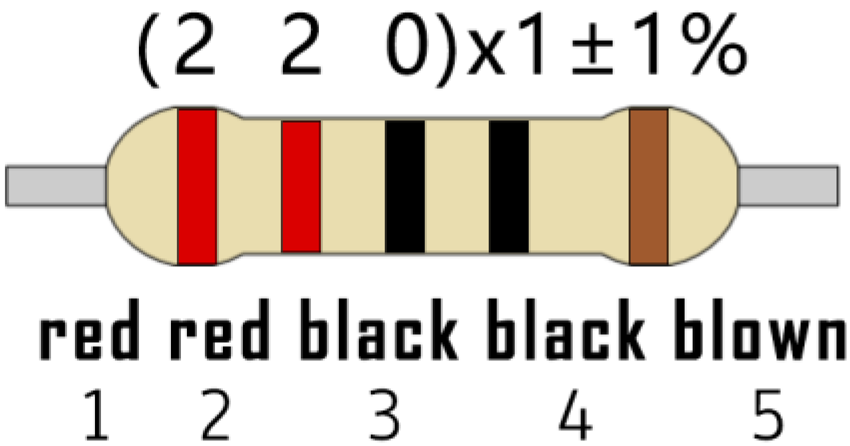


Note:

How to connect a LED

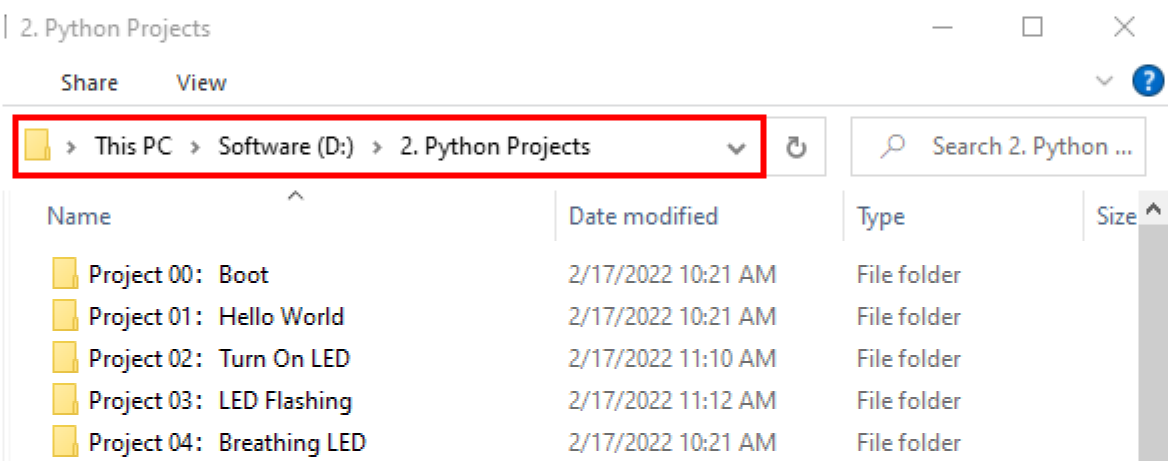


How to identify the 220 Five-color ring resistor



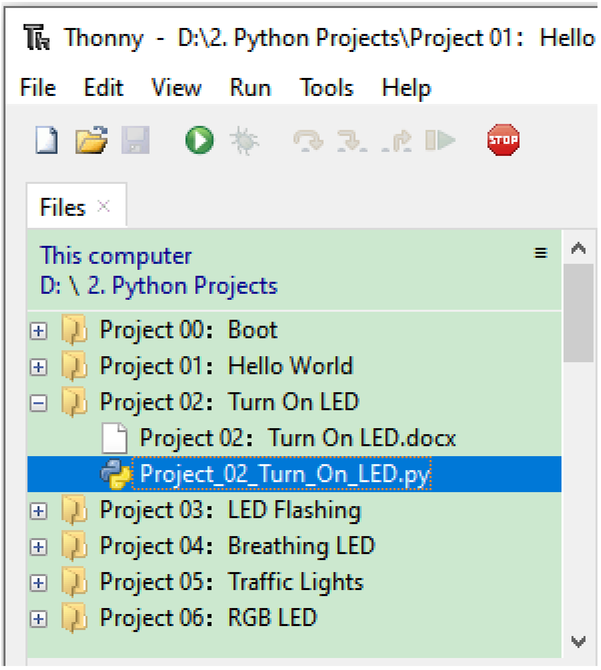
7.3.5 5.Project code

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

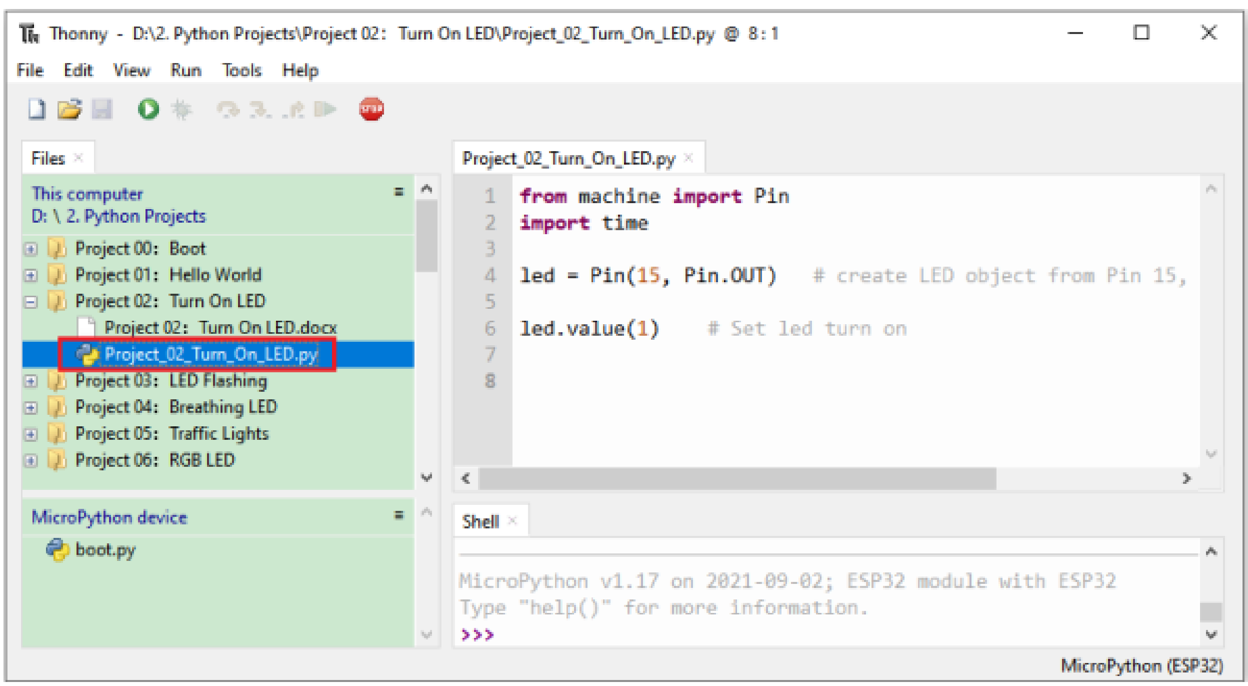


Code running online:

Open "Thonny"click "This computer"→"D:"→"2. Python Projects"→"Project 02Turn On LED".



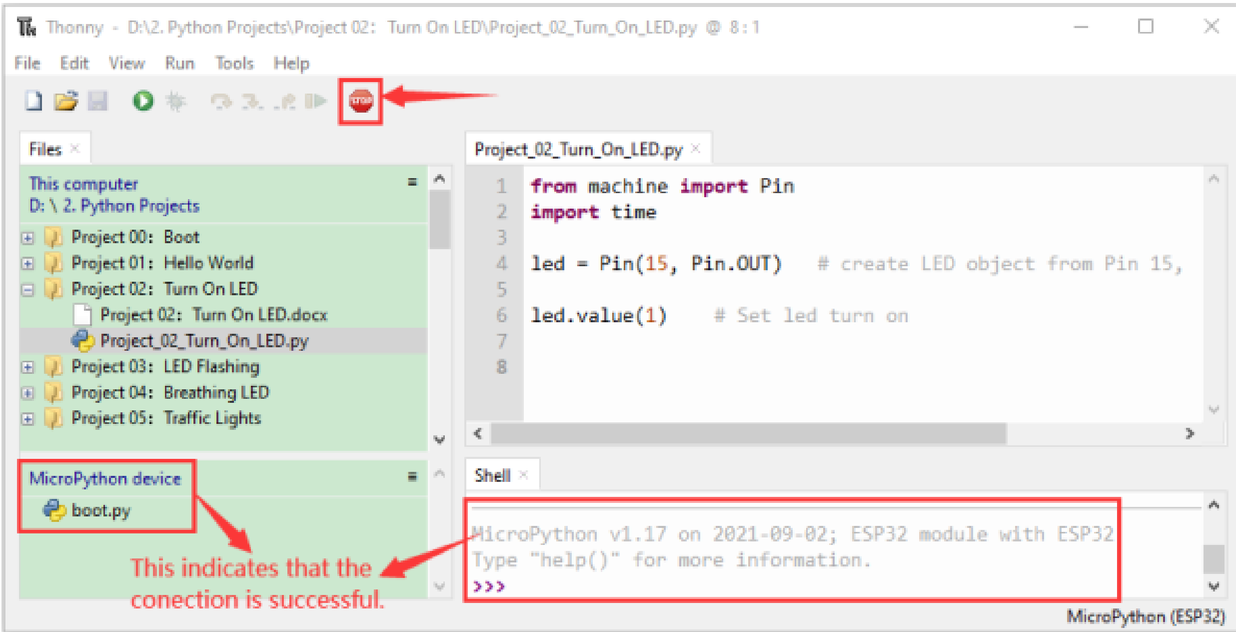
Expand folder"Project 02Turn On LED"and click"Project_02_Turn_On_LED.py" to open it. As shown in the illustration below



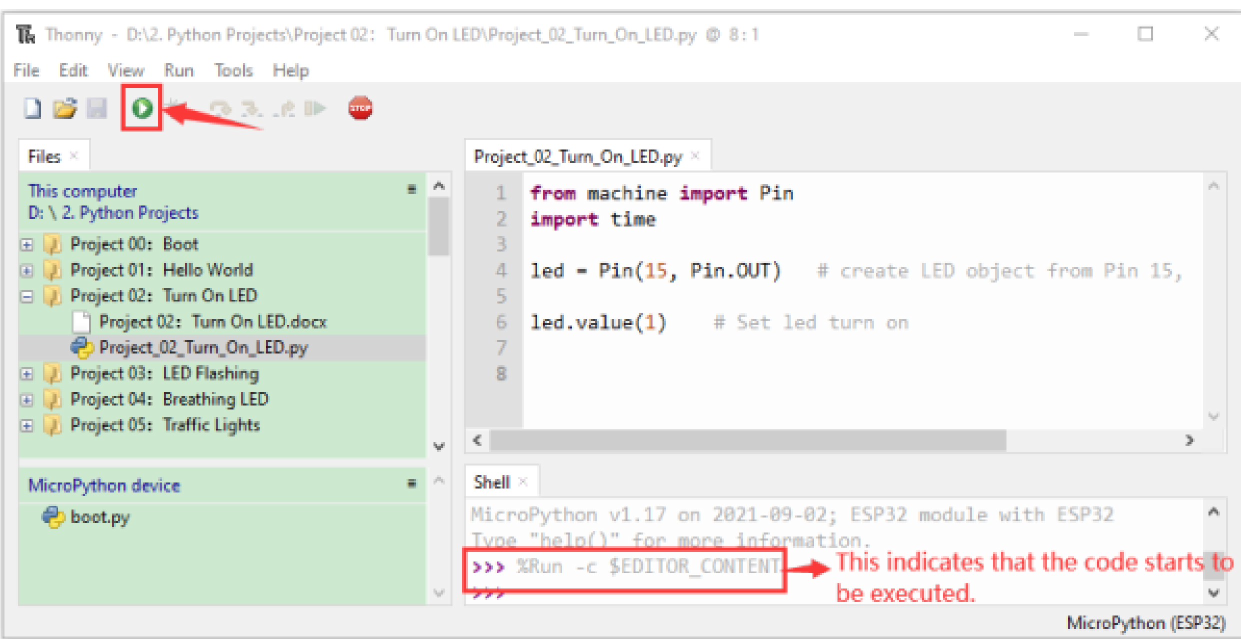

```python
from machine import Pin
import time

led = Pin(15, Pin.OUT)    # create LED object from Pin 15, Set Pin 15 to output

led.value(1)     # Set led turn on
```

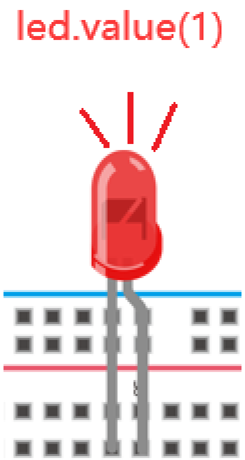
media/27451c8a9c13e29d02bc0f5831cfaf1f.png

Make sure the ESP32 has been connected to the computer. Click “Stop/Restart backend” and see what will display in the“**Shell**” window.

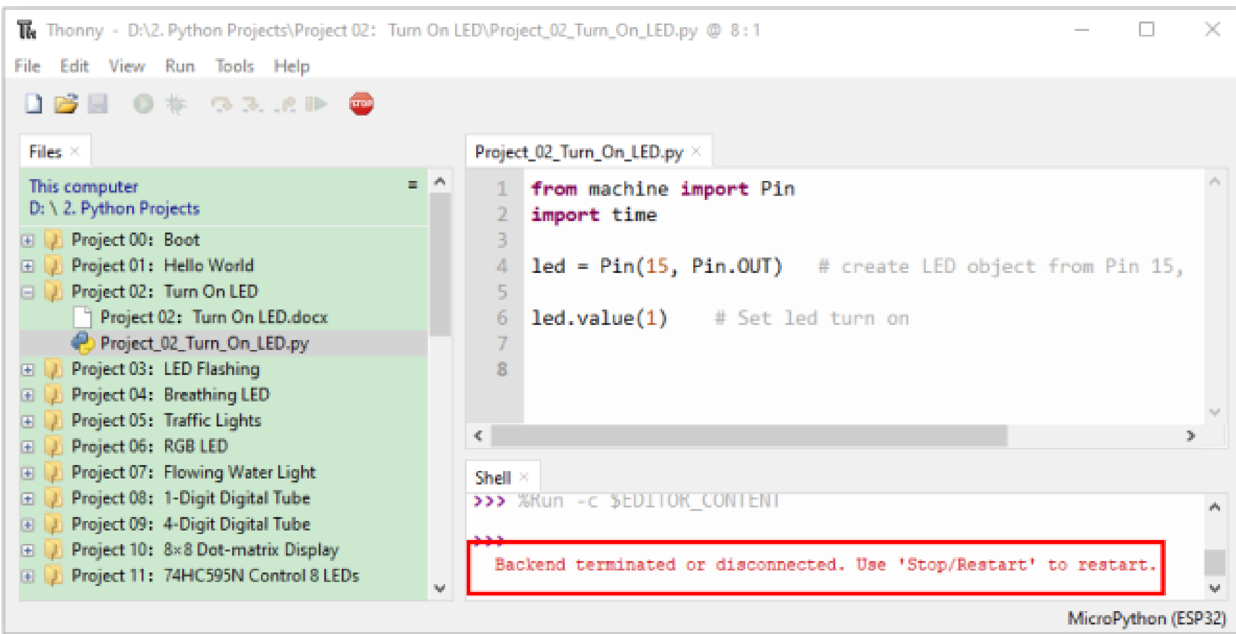


Click “Run current script”, the code starts to be executed and the LED in the circuit lit up. Press “Ctrl+C” or click “Stop/Restart backend” to exit the program.



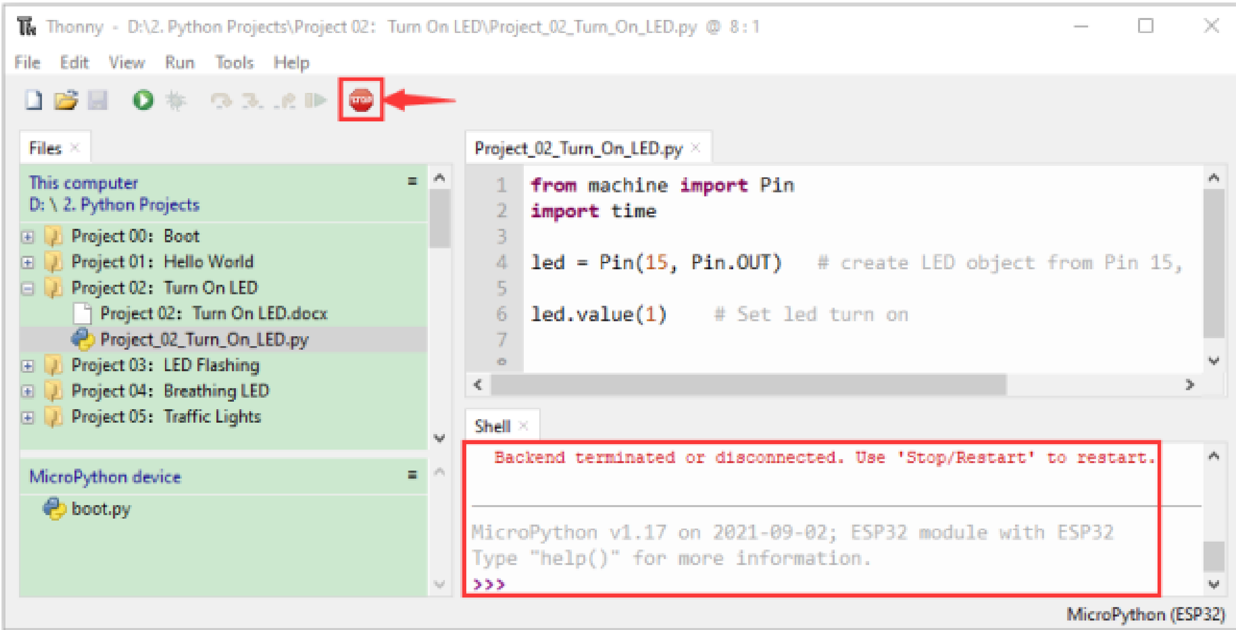


Note: This is the code running online. If you disconnect USB cable and repower ESP32 or press its reset button, LED is not bright and the following messages will be displayed in the "**Shell**" window of Thonny:

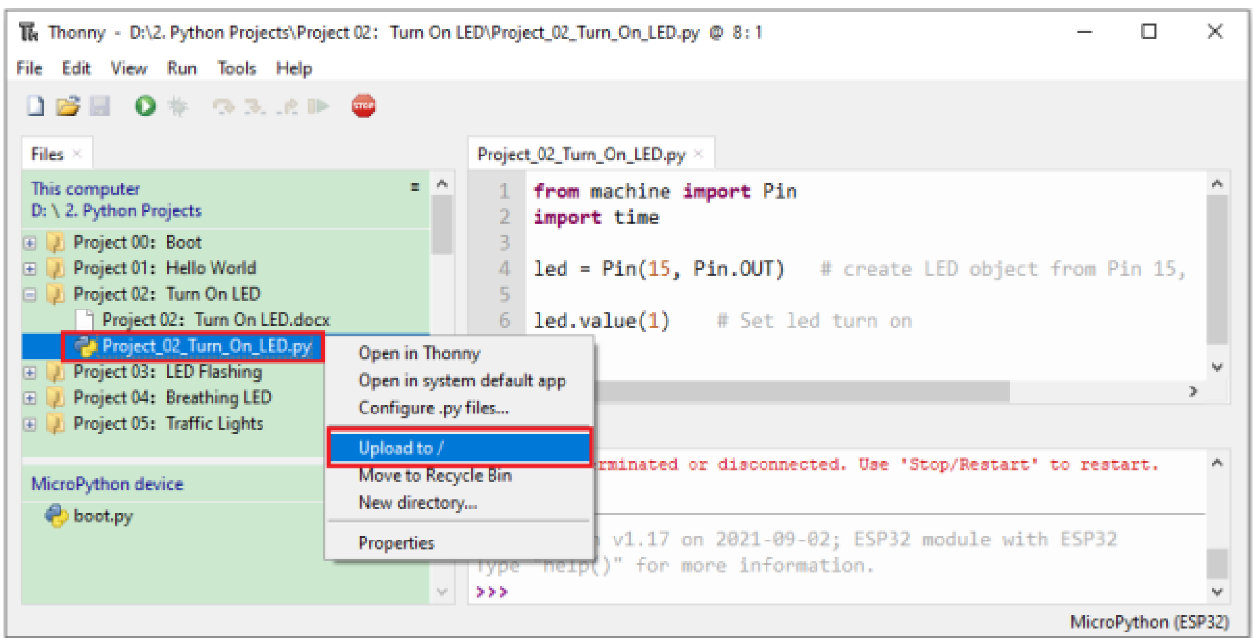


Code running offline (Upload the code to ESP32):

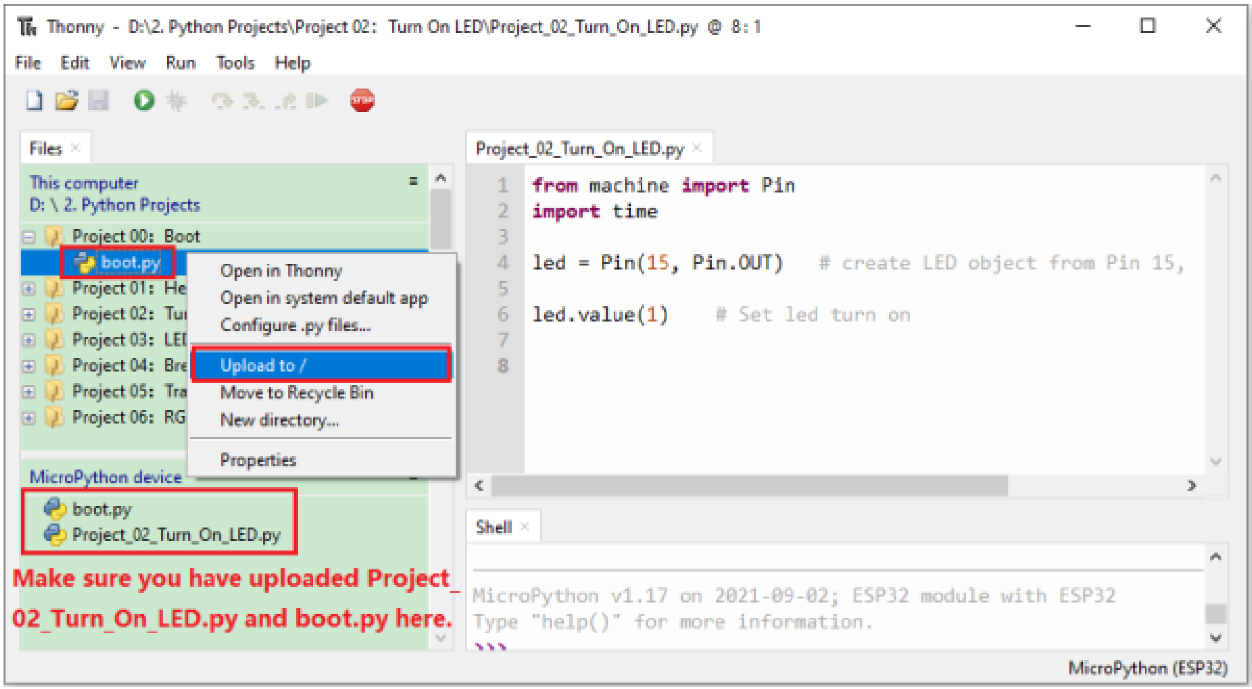
Make sure the ESP32 has been connected to the computer, click  "Stop/Restart backend".



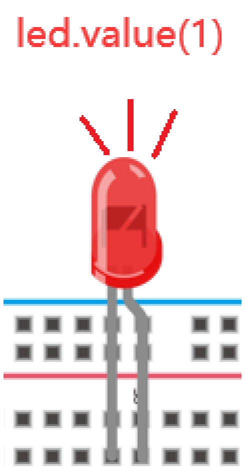
As shown below, right-click the file“Project_02_Turn_On_LED.py”select “**Upload to /**”to upload the code to ESP32.



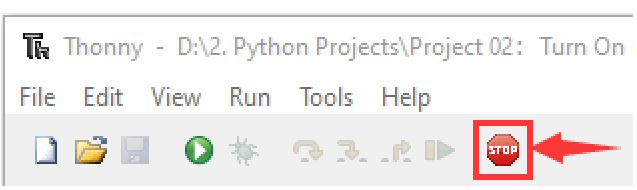
Upload “boot.py” in the same way.



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 "Stop/Restart backend"in Thonny.

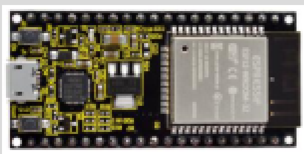
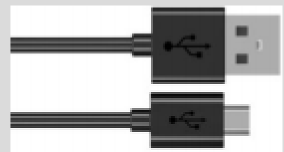
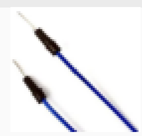


7.4 Project 03LED Flashing

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

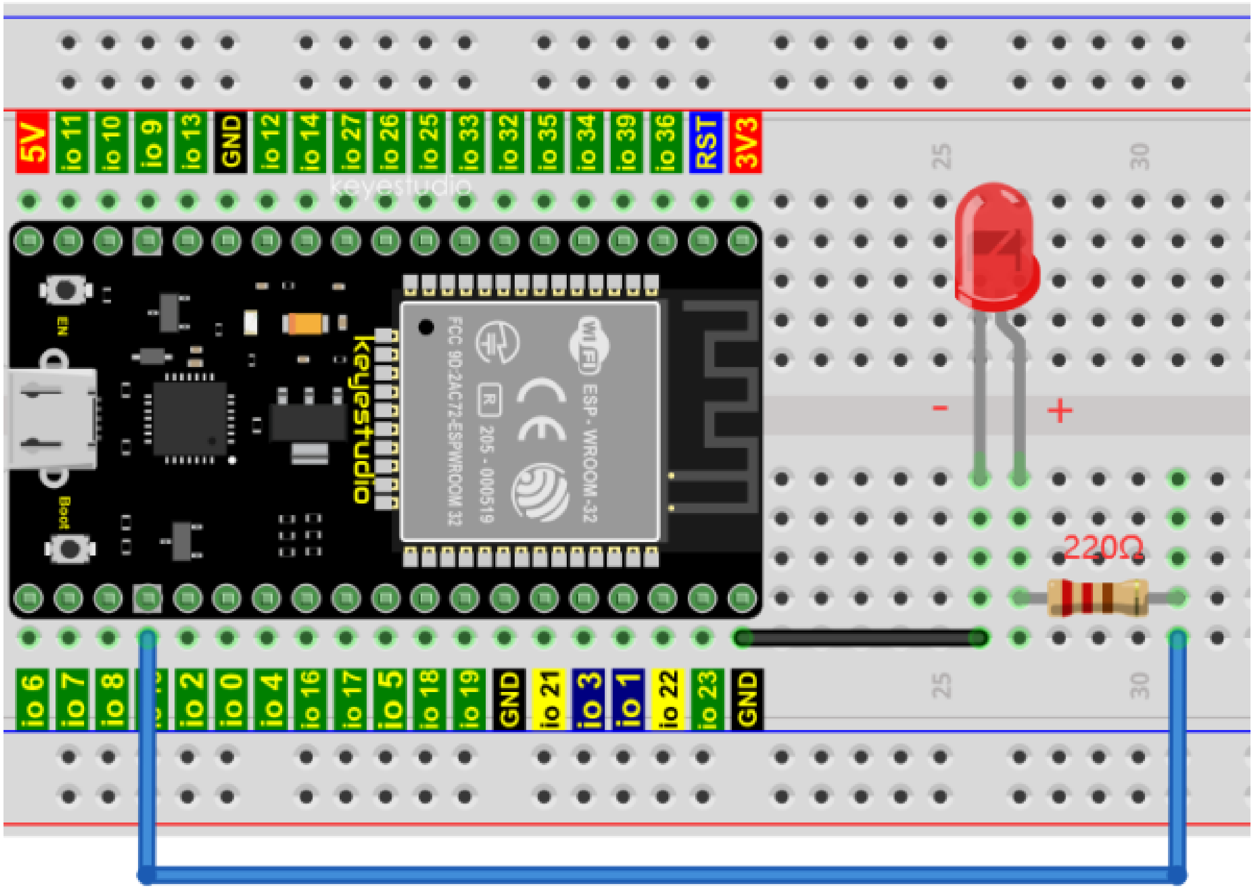
		
ESP32*1	Breadboard*1	USB Cable*1
		
Red LED*1	220 Resistor*1	Jumper Wire*2

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.

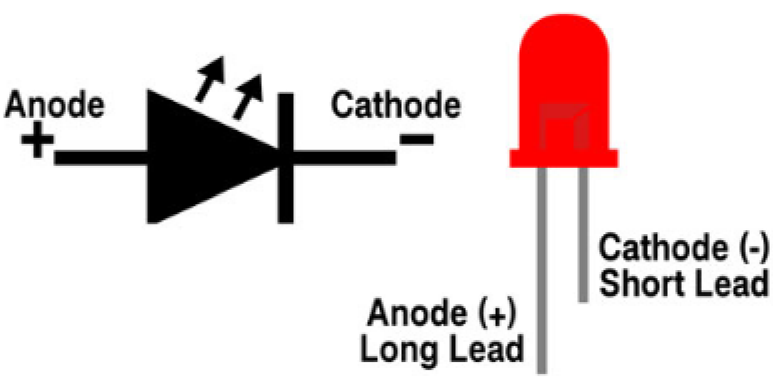
Note: Avoid any possible short circuits (especially connecting 3.3V and GND)!

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

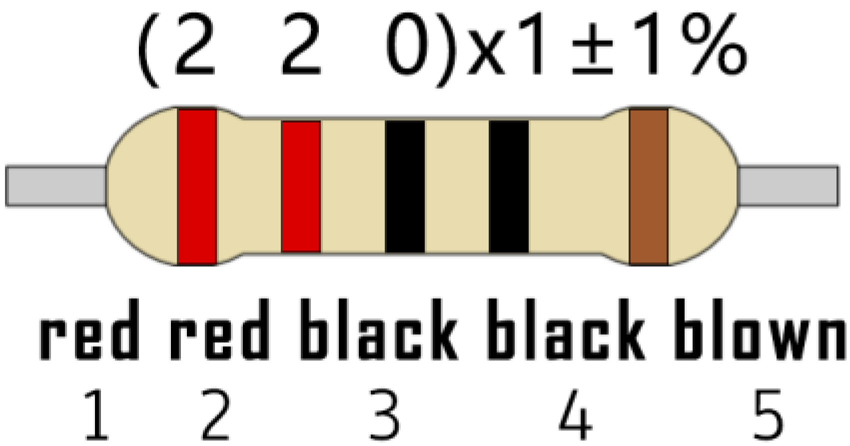


Note:

How to connect a LED

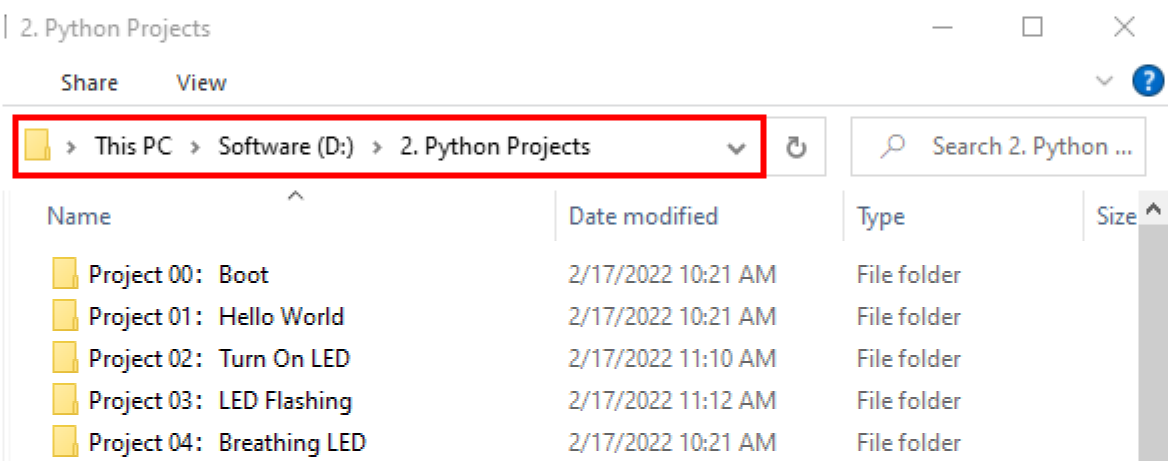


How to identify the 220 Five-color ring resistor



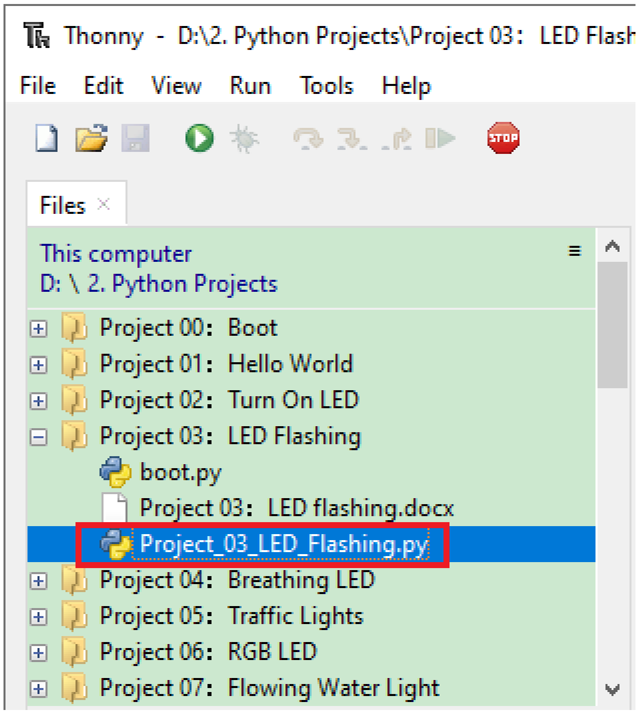
7.4.4 4.Project code

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

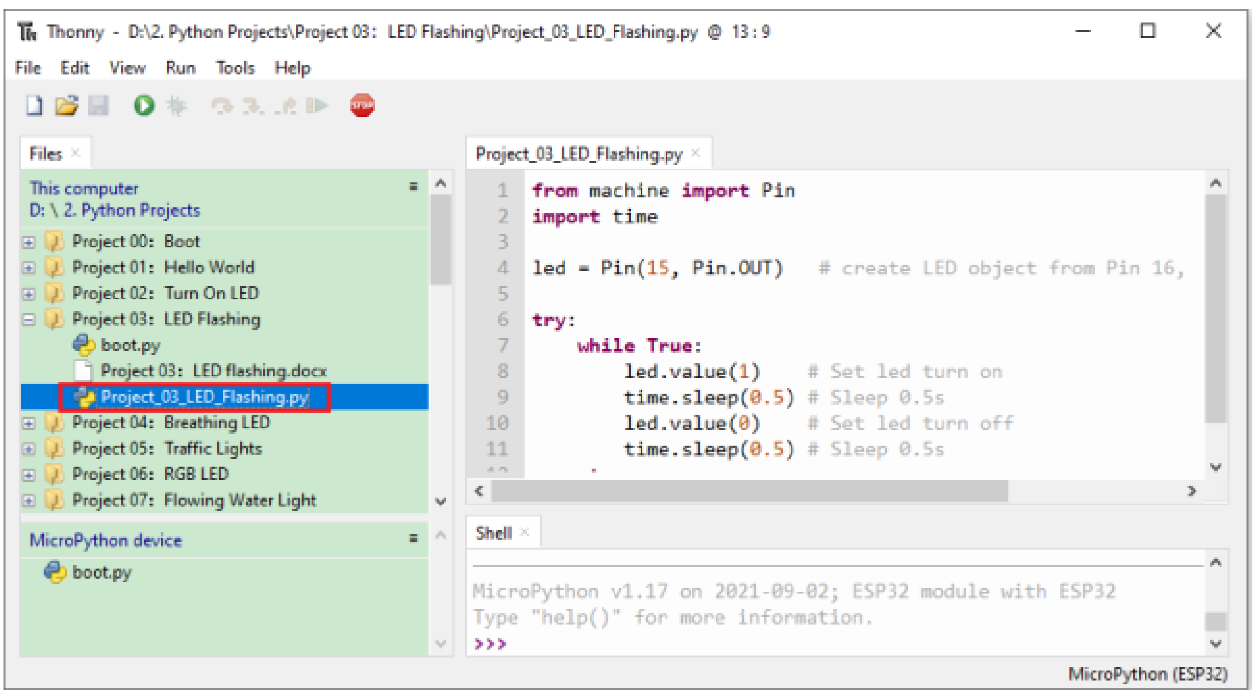


Code running online:

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



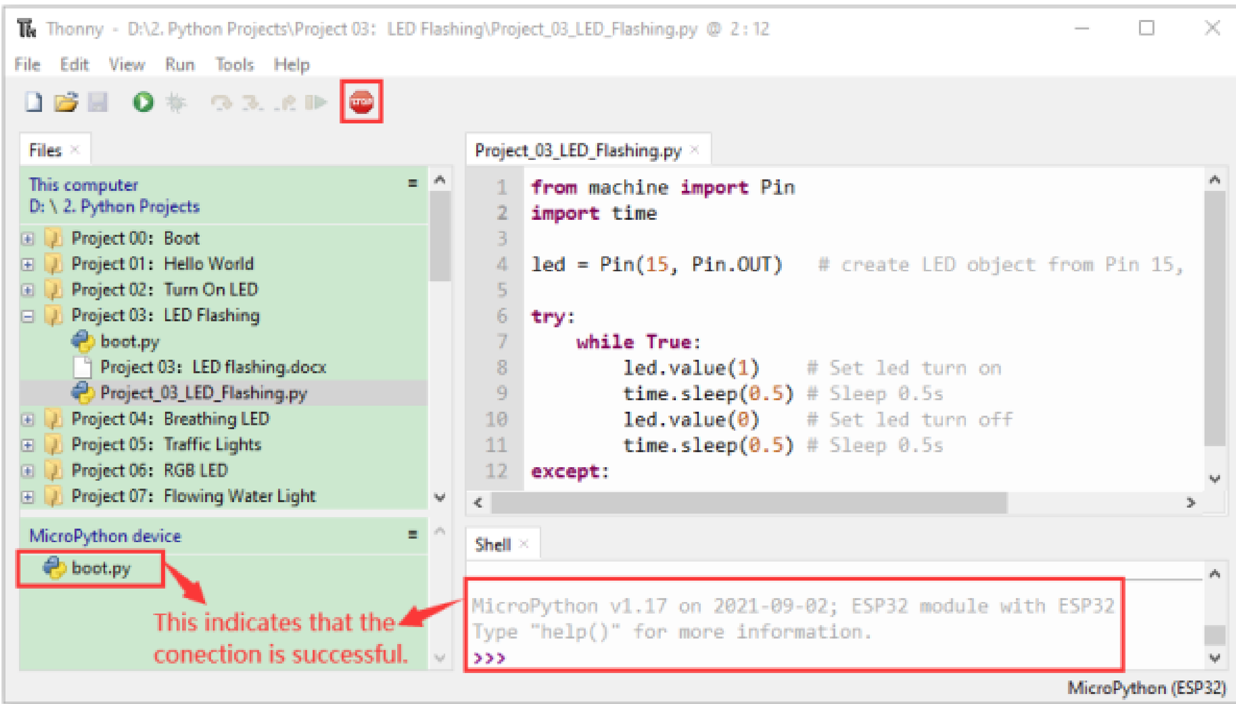

```
from machine import Pin
import time

led = Pin(15, Pin.OUT)   # create LED object from Pin 15, Set Pin 15 to output

try:
```

(continues on next page)

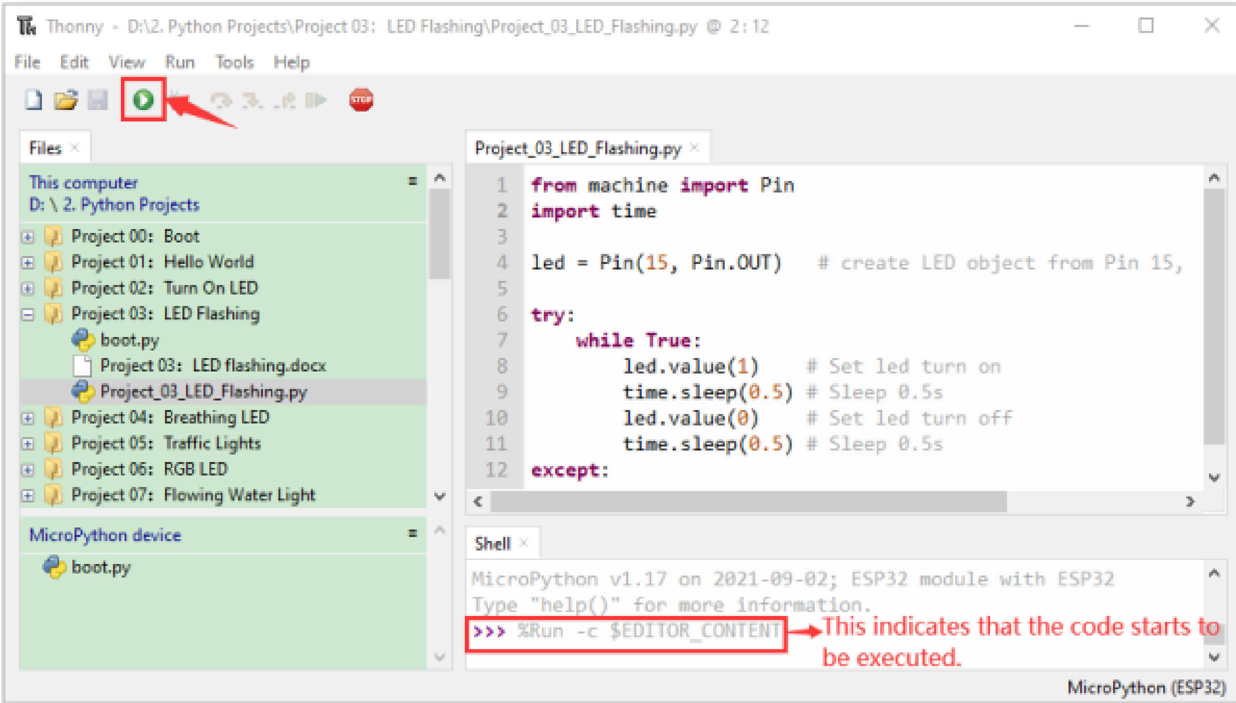
(continued from previous page)

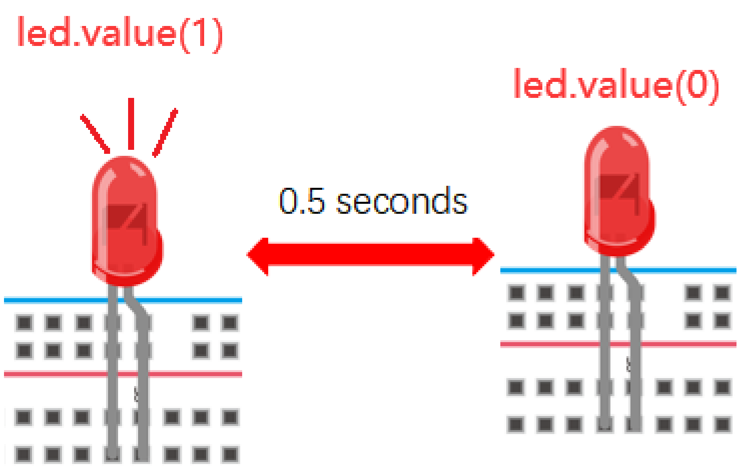
```
    while True:
            led.value(1)    # Set led turn on
            time.sleep(0.5) # Sleep 0.5s
            led.value(0)    # Set led turn off
            time.sleep(0.5) # Sleep 0.5s
except:
        pass
```

Make sure the ESP32 has been connected to the computer. Click "Stop/Restart backend" and see what will display in the "**Shell**" window.

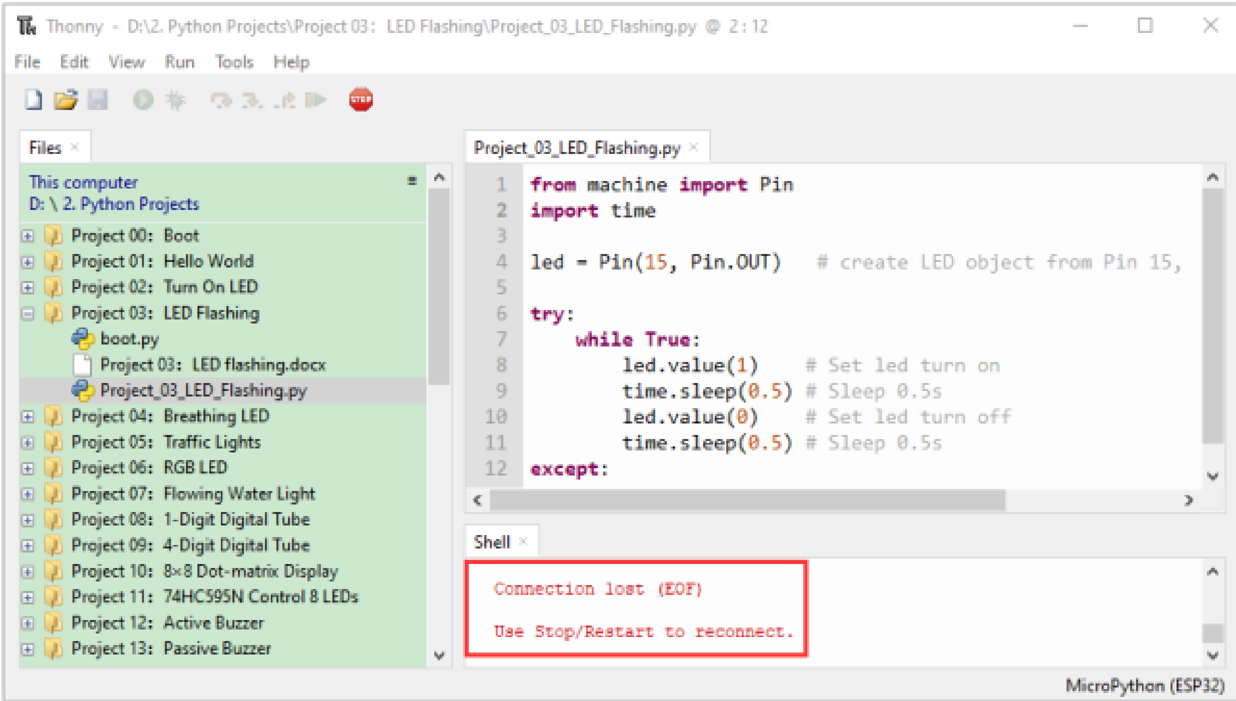


Click "Run current script", the code starts to be executed and you can see the LED is ON for 0.5s and then OFF for 0.5s, which repeats in an endless loop. Press"Ctrl+C"or click "Stop/Restart backend" to exit the program.



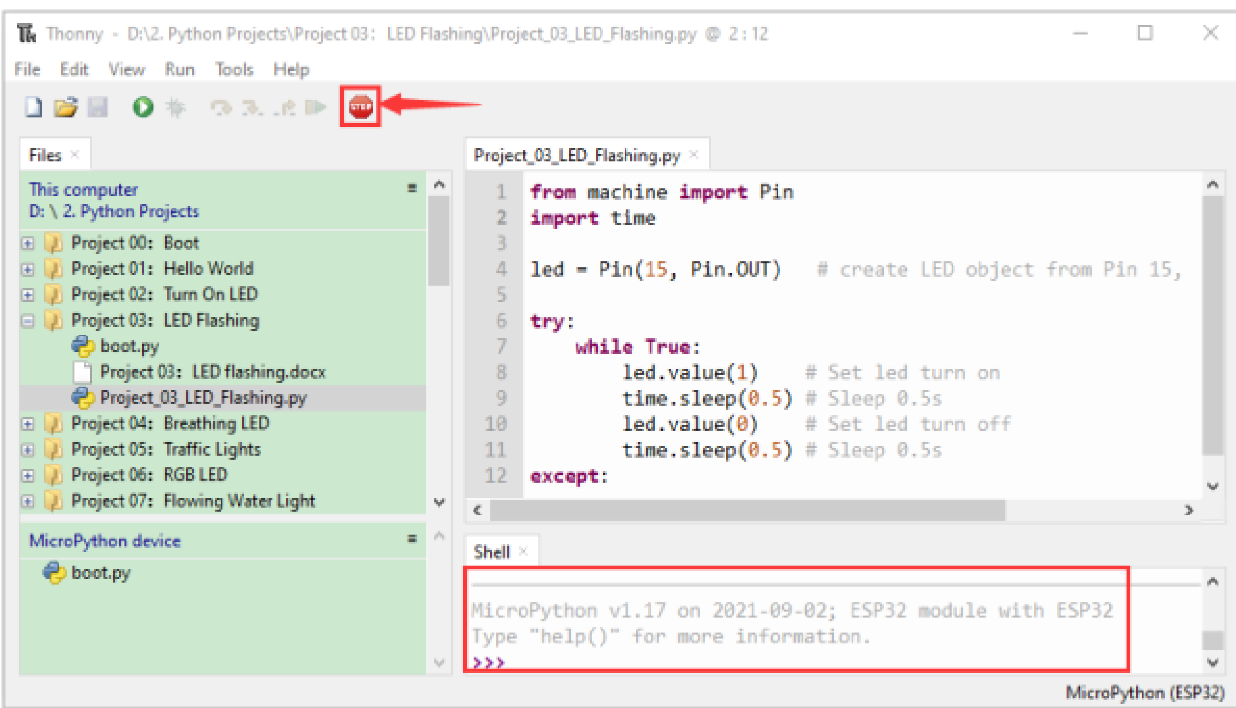


Note: This is the code running online. If you disconnect USB cable and repower ESP32 or press its reset button, the LED in the circuit stops flashing and the following messages will be displayed in the "Shell" window of Thonny:

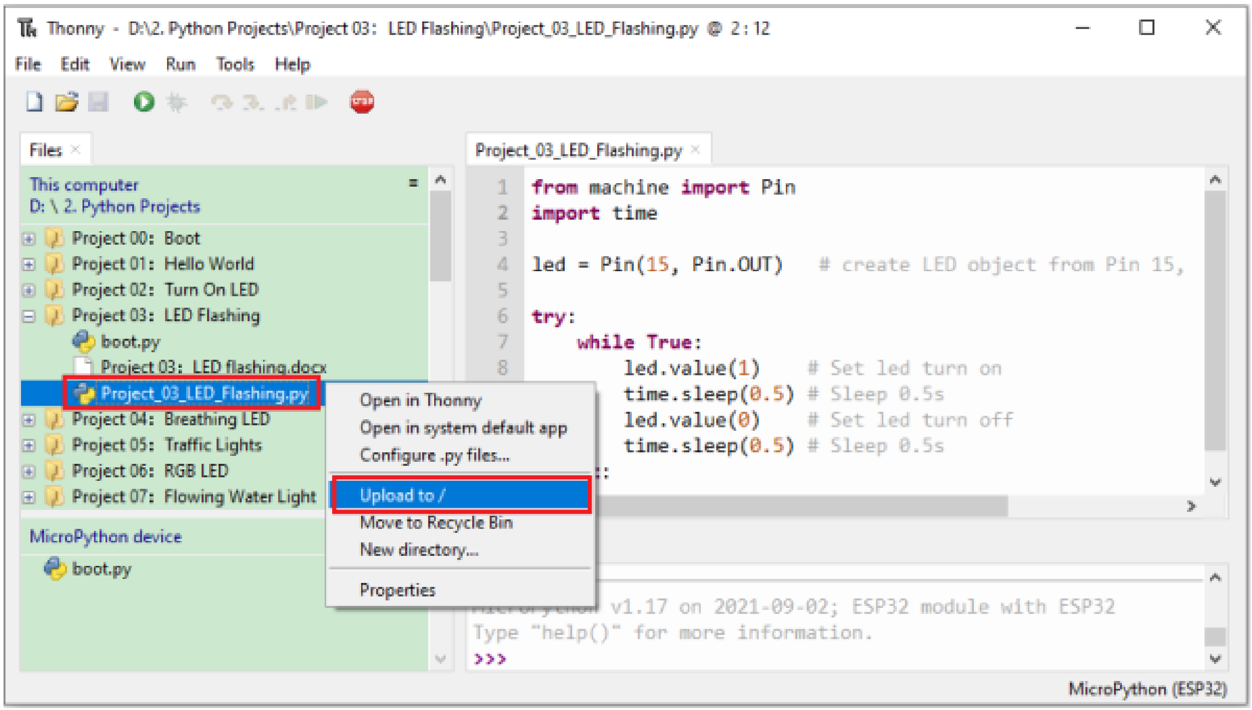


Code running offlineUpload the code to ESP32

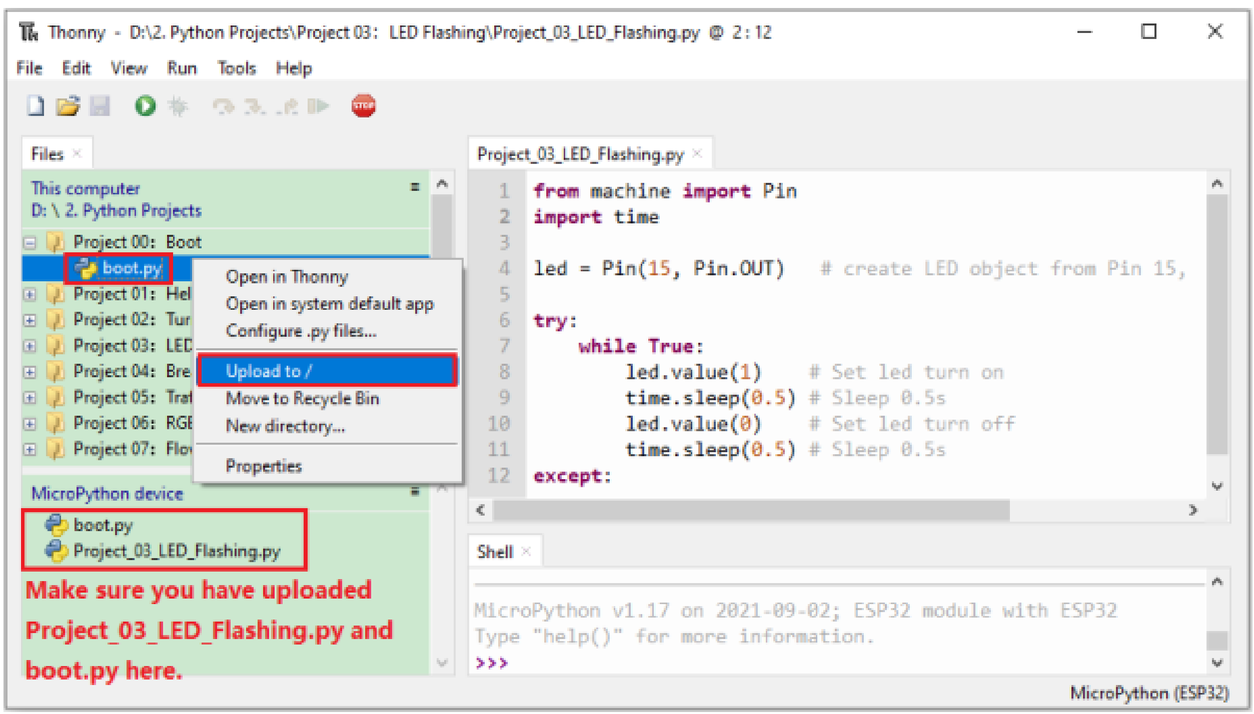
Make sure the ESP32 has been connected to the computer, click “Stop/Restart backend”.



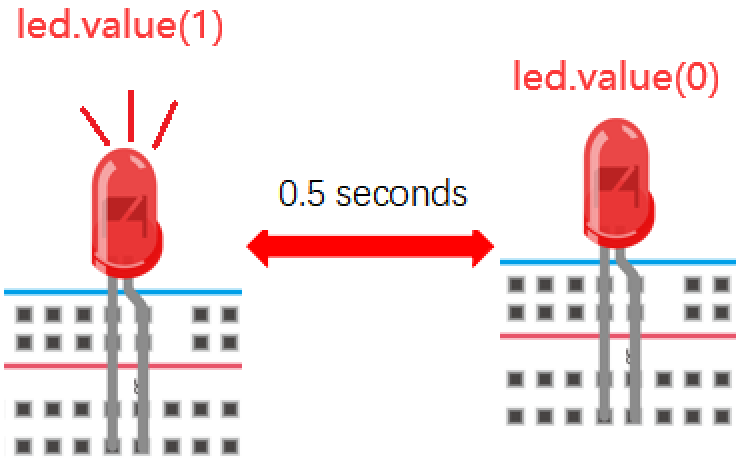
As shown below, right-click the file“Project_03_LED_Flashing.py”select “**Upload to /**”to upload the code to ESP32.



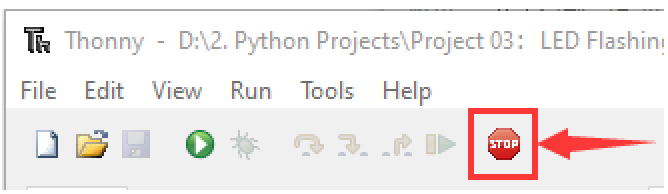
Upload "boot.py" in the same way.



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 "Stop/Restart backend"in Thonny.



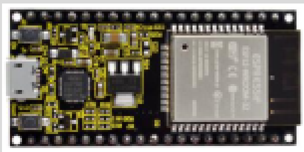
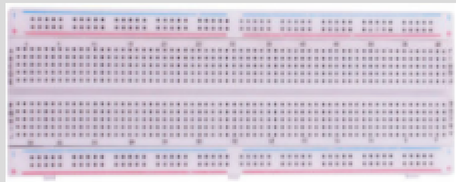
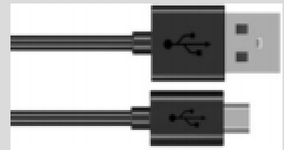
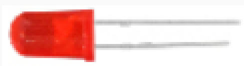
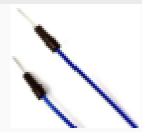
7.5 Project 04: Breathing Led

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.

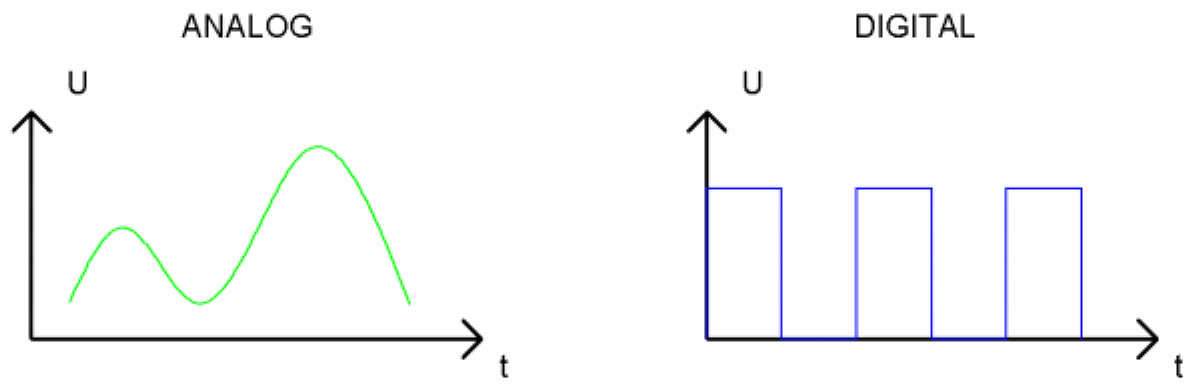
7.5.2 2.Components

		
ESP32*1	Breadboard*1	USB Cable*1
		
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℃ to 10℃. 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.



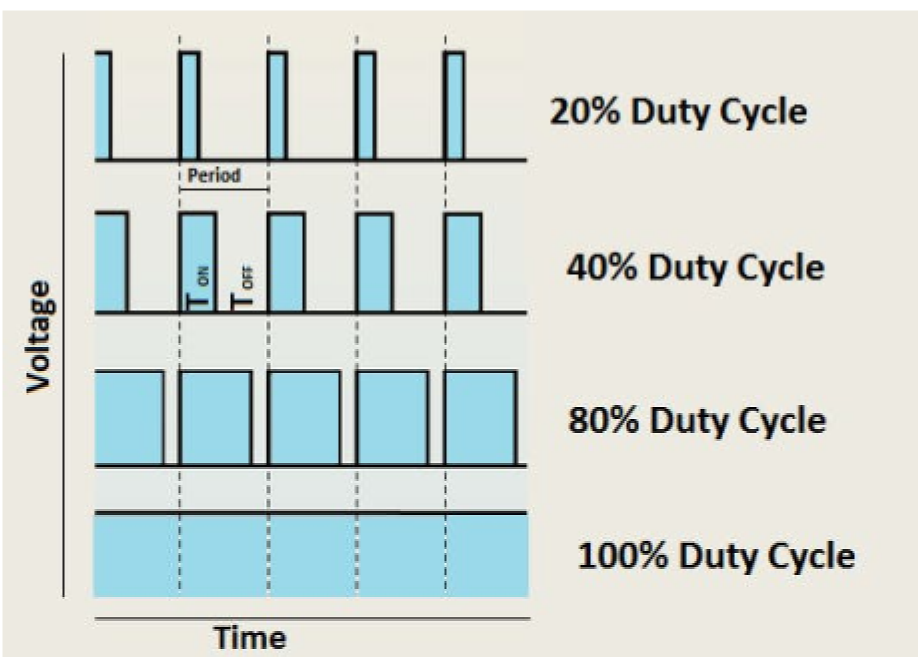
In practical application, we often use binary as the digital signal, that is a series of 0's and 1's. Since a binary signal only has two values (0 or 1), it has great stability and reliability. Lastly, both analog and digital signals can be converted into the other.

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%:

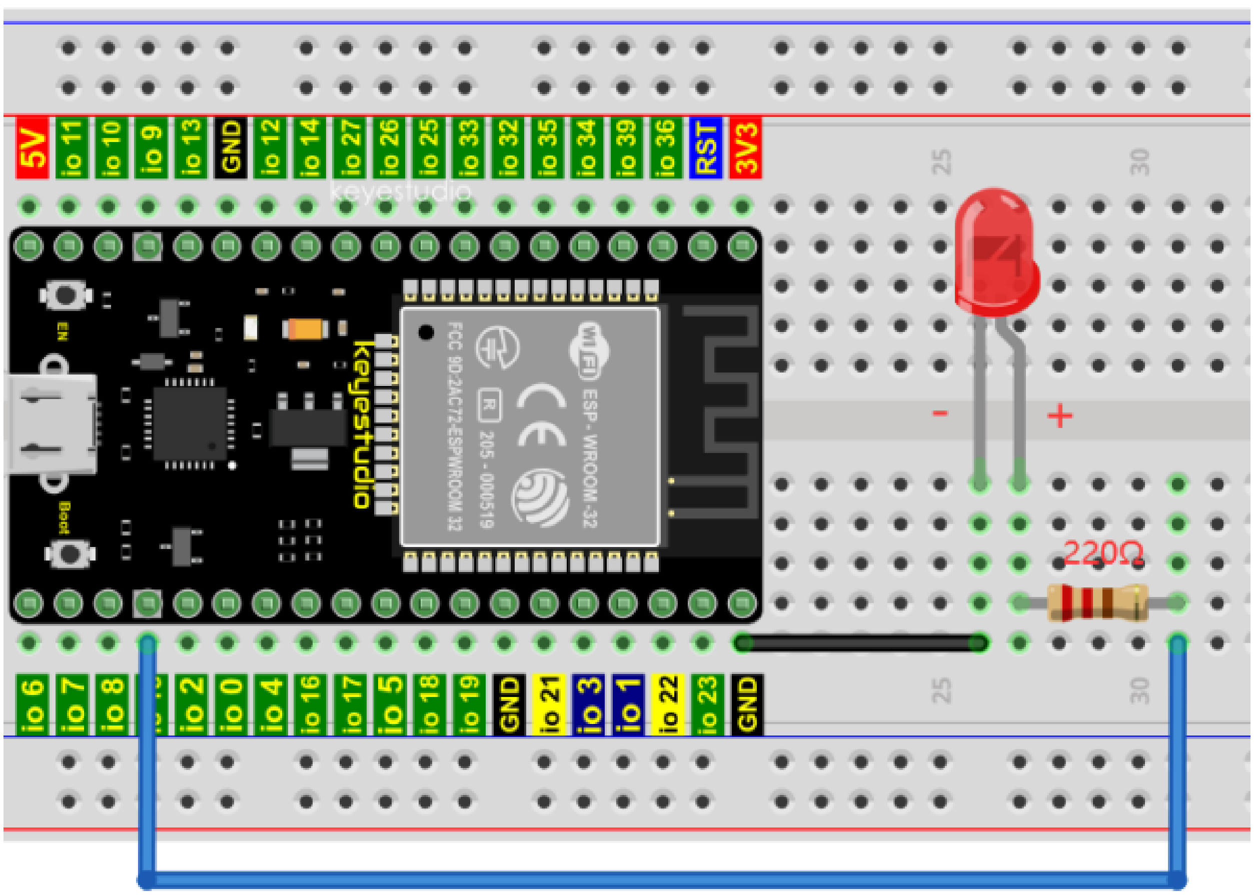


The longer the PWM duty cycle is, the higher the output power will be. Now that we understand this relationship, we can use PWM to control the brightness of an LED or the speed of DC motor and so on. It is evident from the above that PWM is not real analog, and the effective value of the voltage is equivalent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

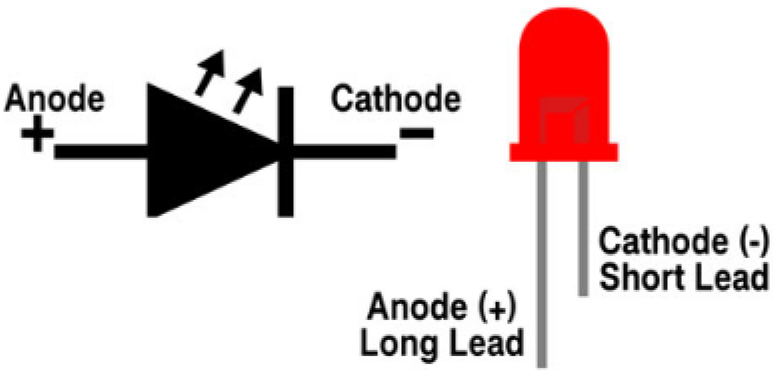
The ESP32 PWM controller has 8 independent channels, each of which can independently control frequency, duty cycle, and even accuracy. Unlike traditional PWM pins, the PWM output pins of ESP32 are configurable and they can be configured to PWM.

7.5.4 4.Wiring diagram

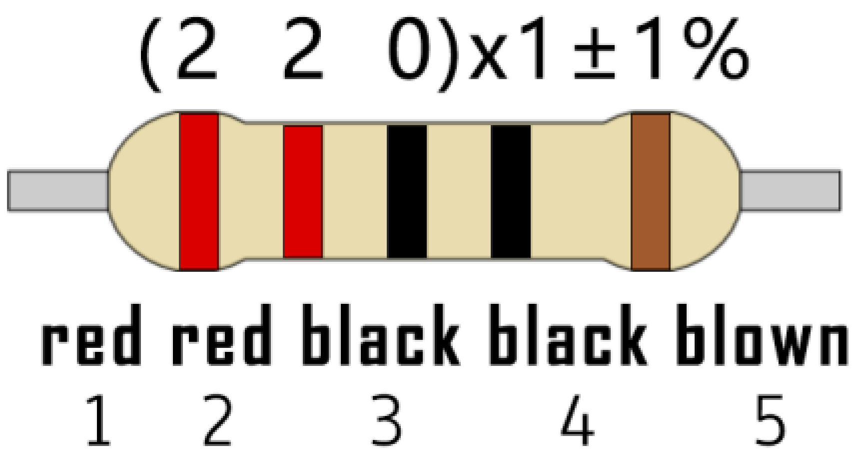


Note:

How to connect a LED



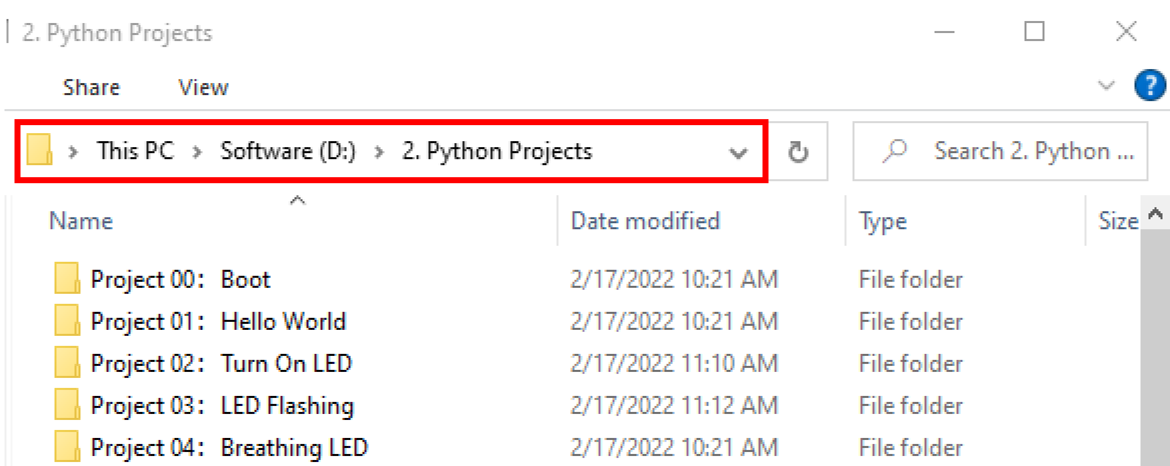
How to identify the 220 Five-color ring resistor



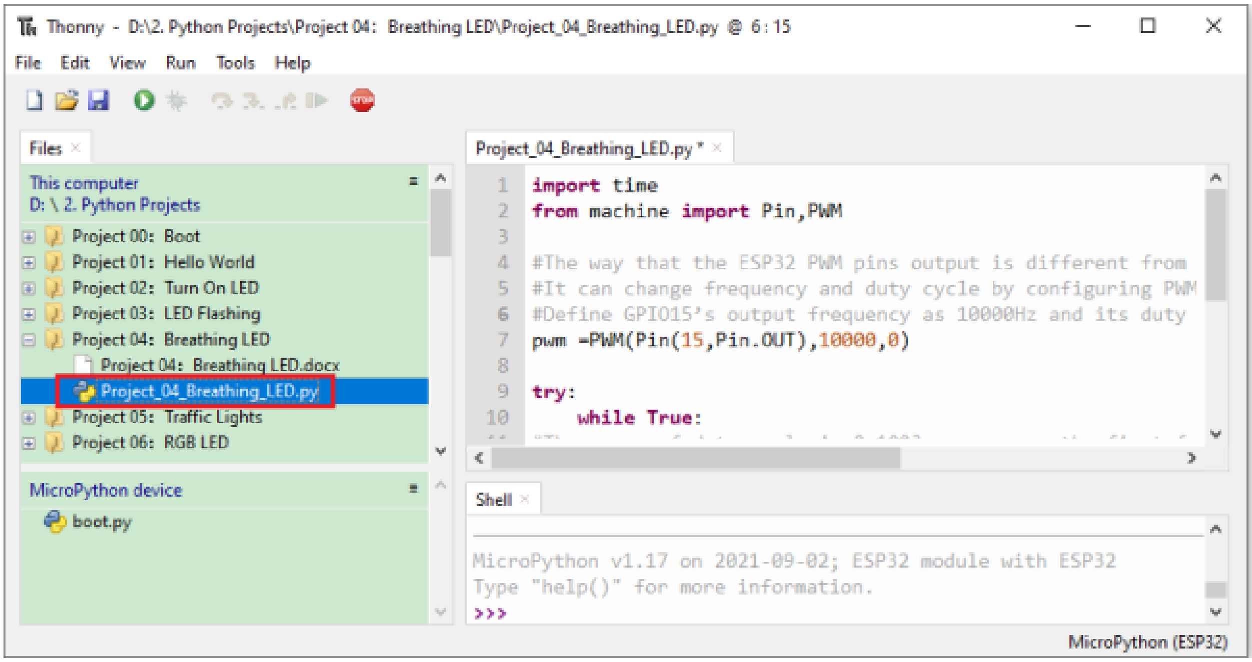
7.5.5 5. Project code

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



Open“Thonny”click“This computer”→“D:”→“2. Python Projects”→“Project 04Breathing Led”, and double left-click “Project_04_Breathing_LED.py”.



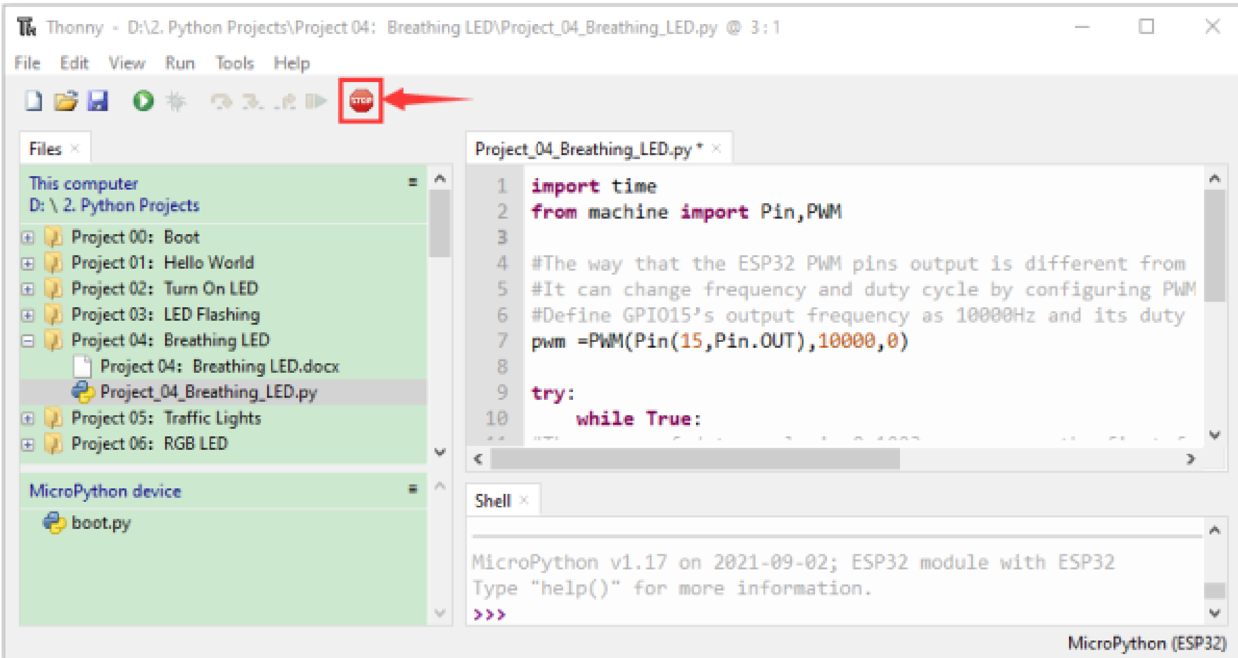

```
import time
from machine import Pin,PWM

#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 GPIO15’s output frequency as 10000Hz and its duty cycle as 0, and assign them to␣
↪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␣
↪change the duty
#cycle value,making PWM output 0% -100%; Use the second for loop to make PWM output 100%-
↪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,␣
↪after each use of PWM,
#deinit() needs to be called to turned OFF the timer. Otherwise, the PWM may fail to␣
↪work next time.
    pwm.deinit()
```

(continues on next page)

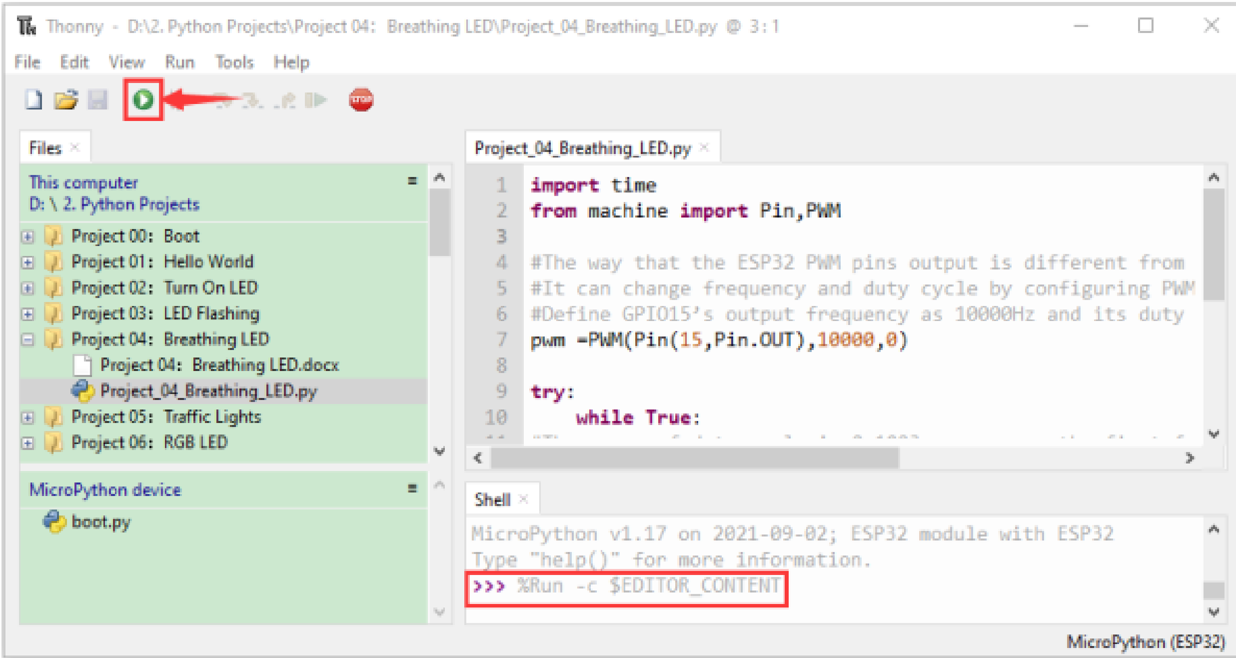
(continued from previous page)

7.5.6 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 LED is turned from ON to OFF and then back from OFF to ON gradually like breathing. Press "Ctrl+C" or click "Stop/Restart backend" to exit the program.

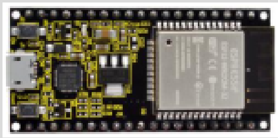
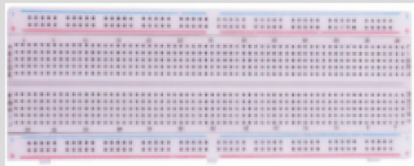
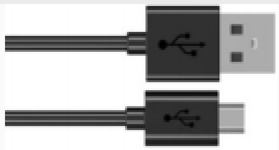
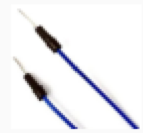


7.6 Project 05Traffic Lights

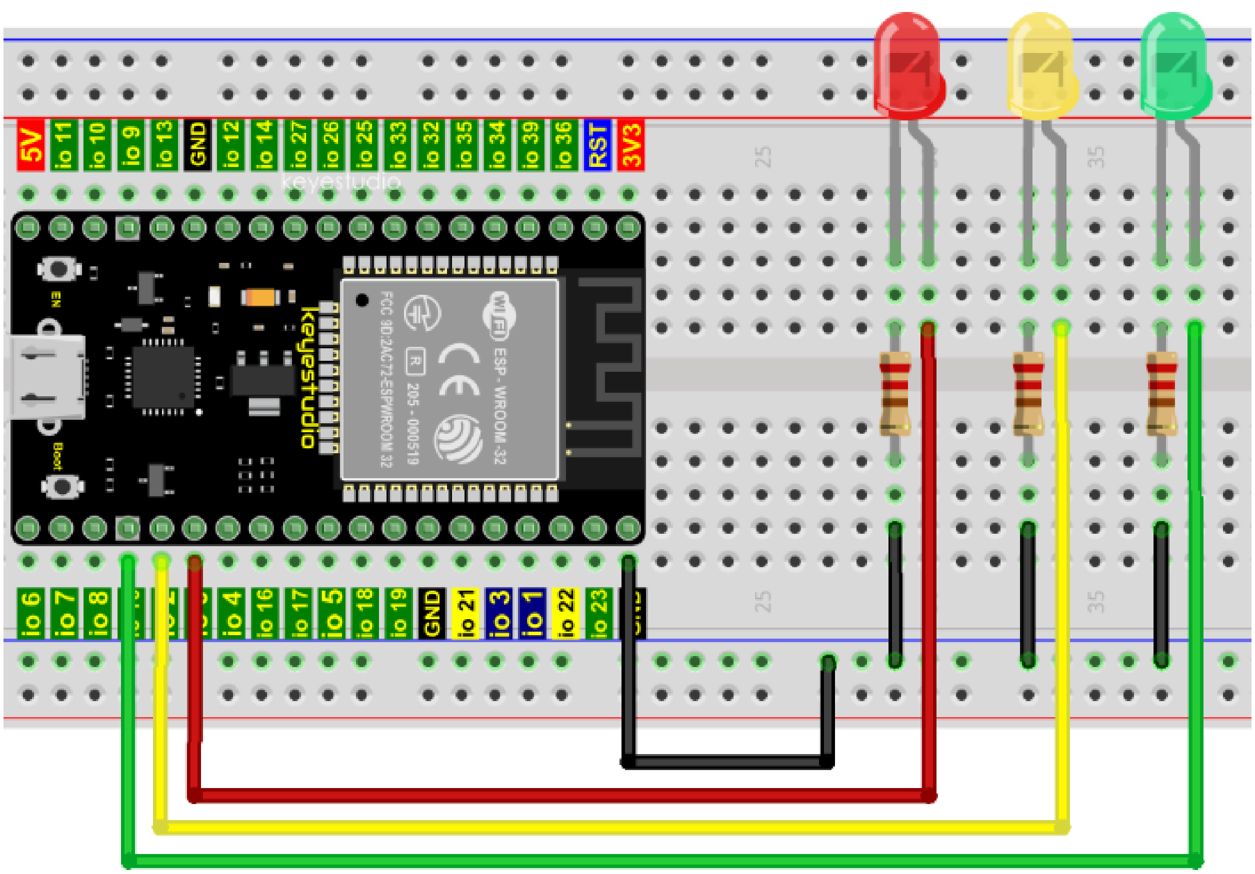
7.6.1 1.Introduction

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

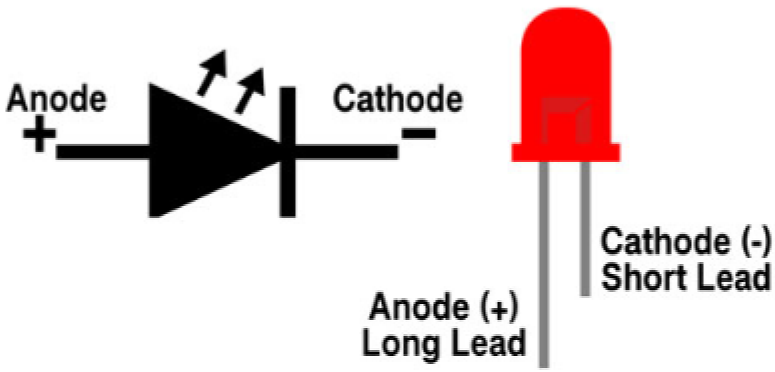
			
ESP32*1	Bread board*1	Red LED*1	Yellow LED*1
			
Green LED*1	USB Cable*1	220 Resistor*3	Jumper Wires

7.6.3 3. Wiring diagram

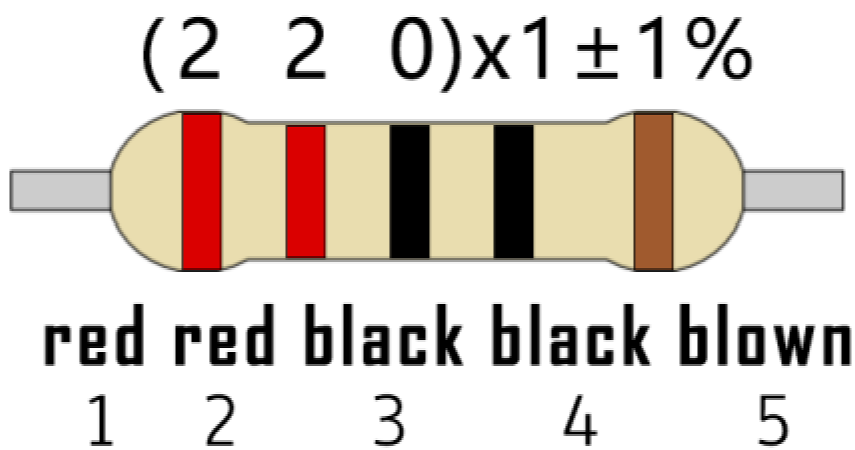


Note:

How to connect a LED

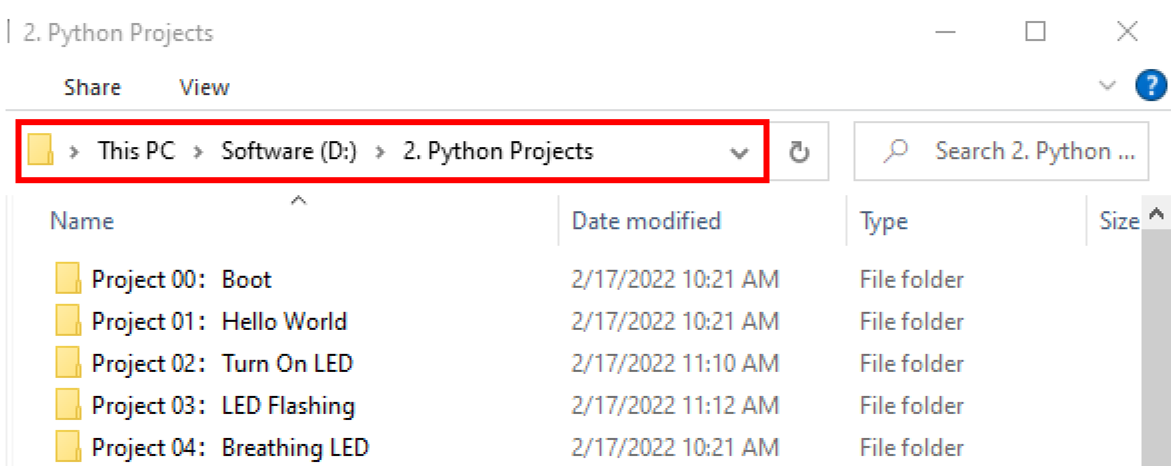


How to identify the 220 Five-color ring resistor

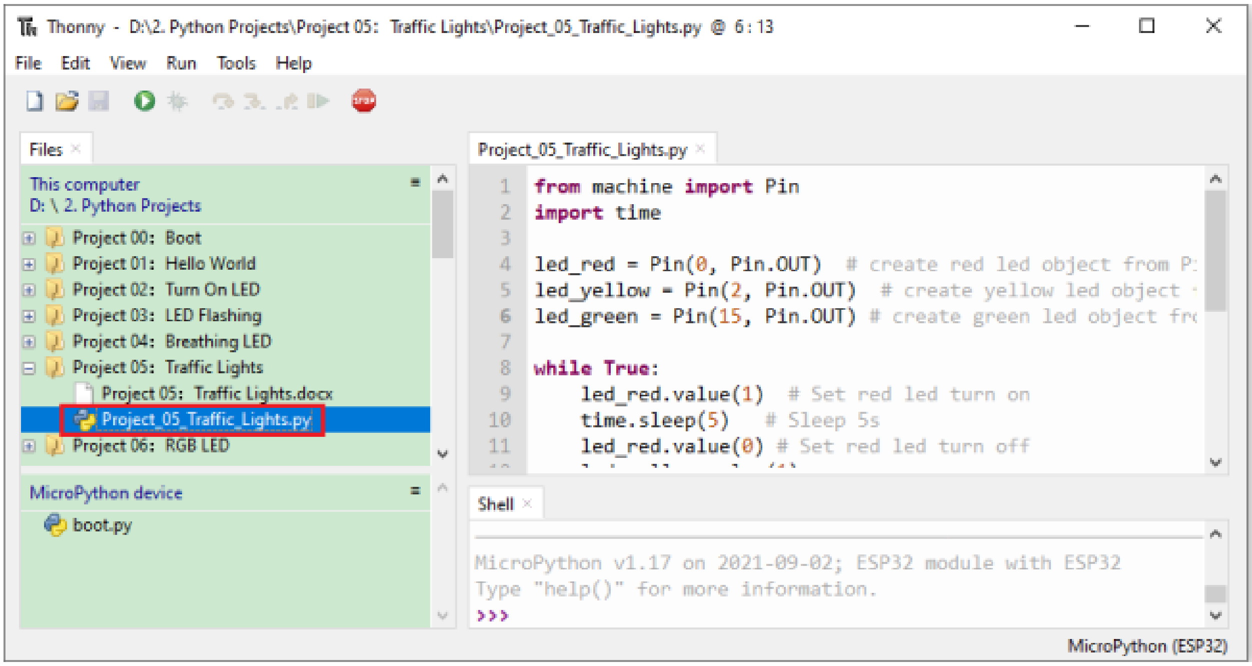


7.6.4 4. Project code

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



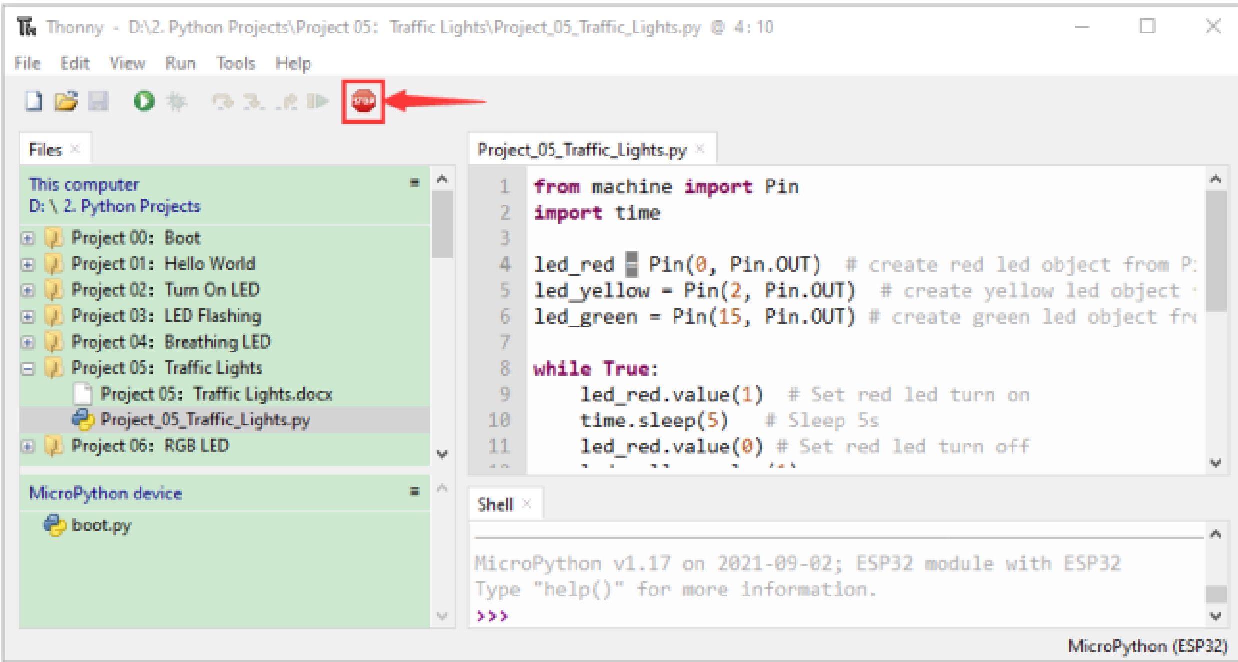
Open"Thonny"click"This computer"→"D:"→"2. Python Projects"→"Project 05Traffic Lights". and double left-click "Project_05_Traffic_Lights.py".




```
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(0)
```

7.6.5 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 are below:

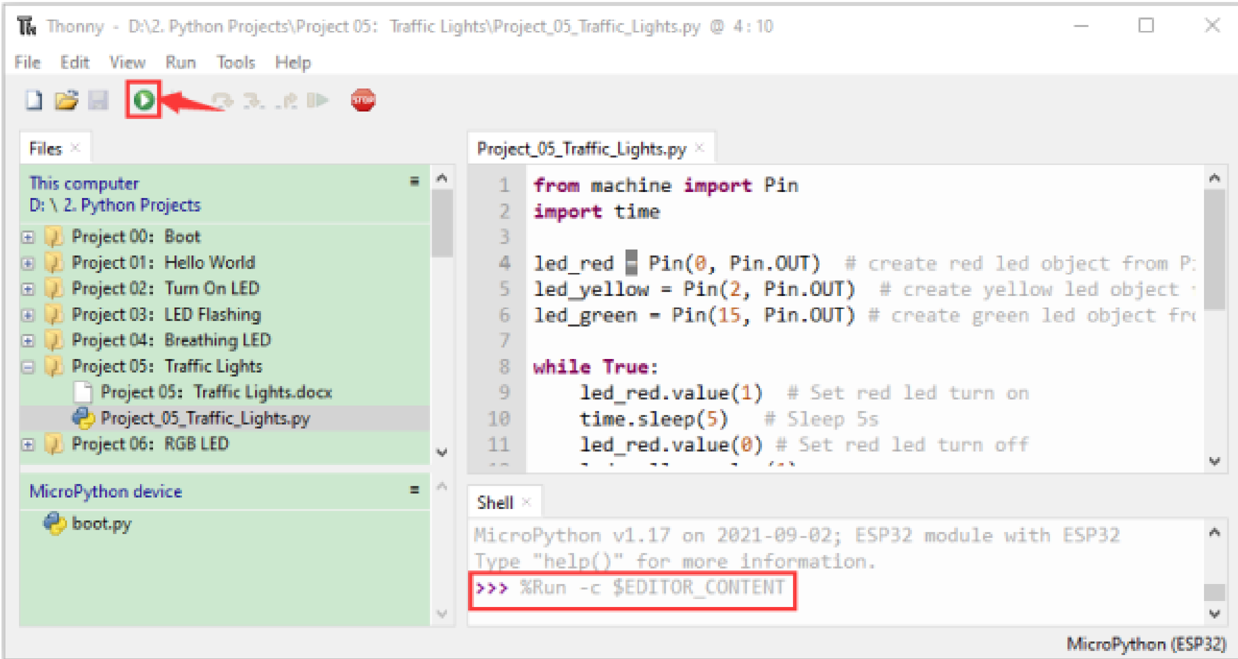
First, the green light will be on for five seconds and then off;

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

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

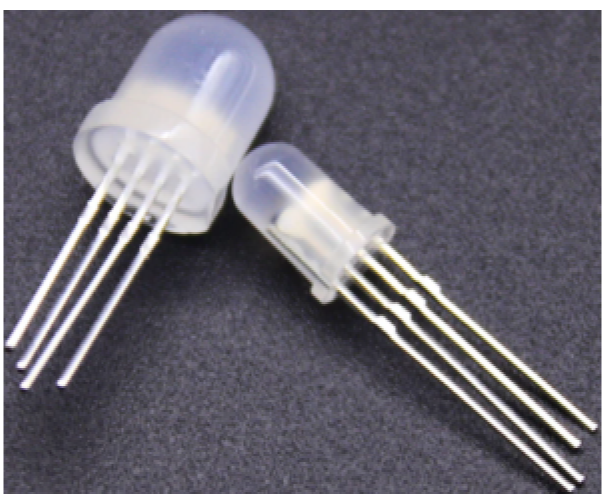
Repeat steps 1 to 3 above.

Press “Ctrl+C” or click “Stop/Restart backend” to exit the program.



7.7 Project 06: RGB LED

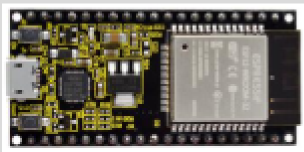
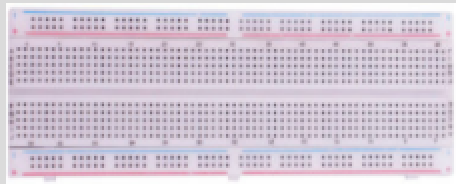
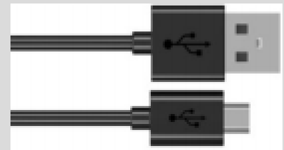
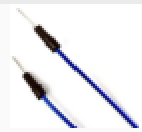
7.7.1 1.Introduction



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

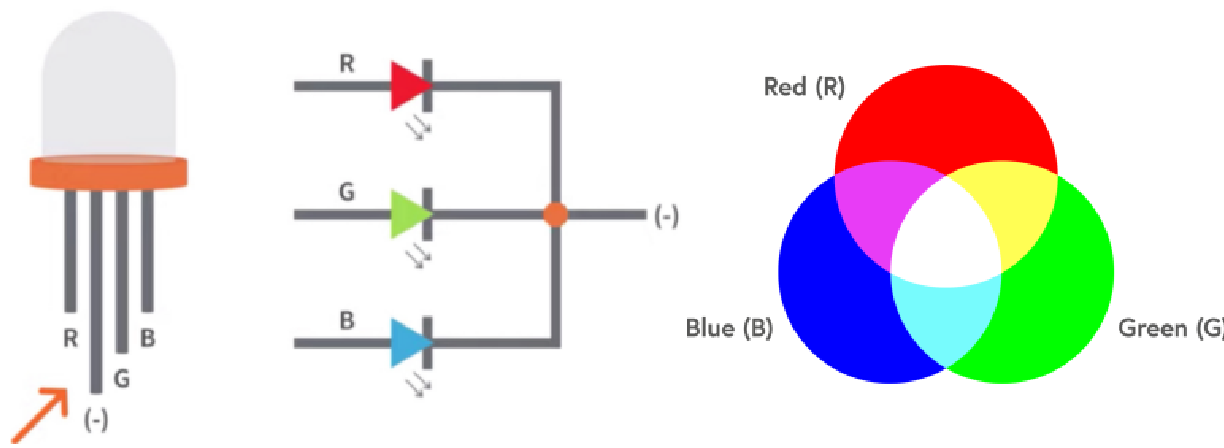
In this project, we will introduce the RGB and show you how to use ESP32 to control the RGB to emit different color light. RGB is pretty basic, but it's also a great way to learn the fundamentals of electronics and coding.

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

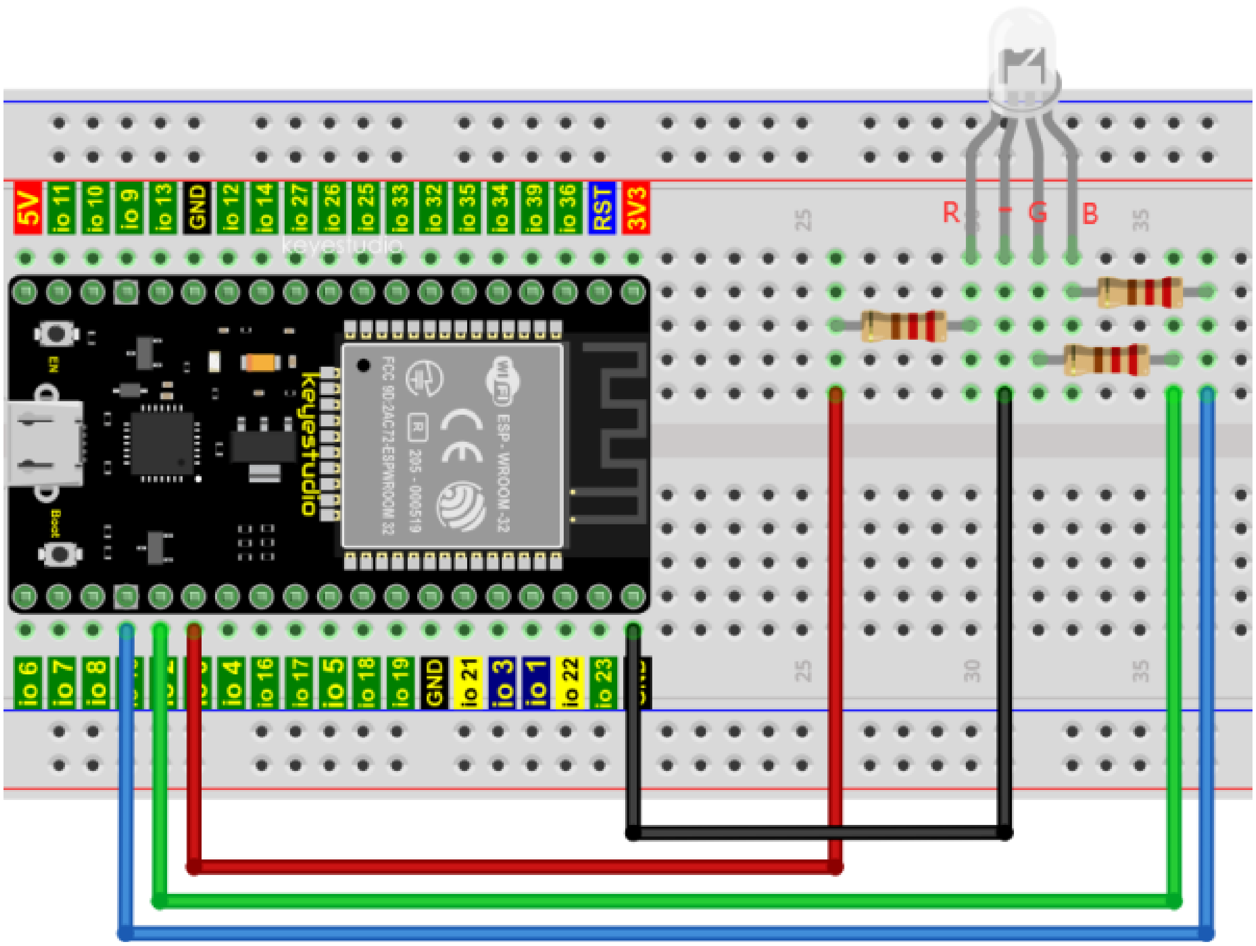
Most monitors adopt the RGB color standard, and all colors on a computer screen are a mixture of red, green and blue in varying proportions.



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.

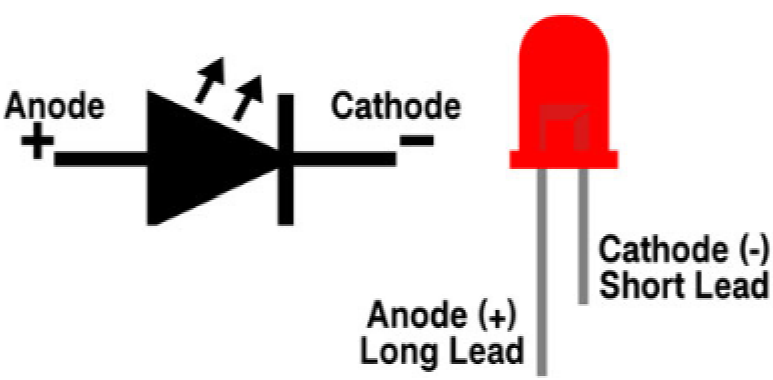
If we use three 10-bit PWM to control the RGB, in theory, we can create 2 10*210*210 = 1,073,741,824 (1 billion) colors through different combinations.

7.7.4 4.Wiring diagram

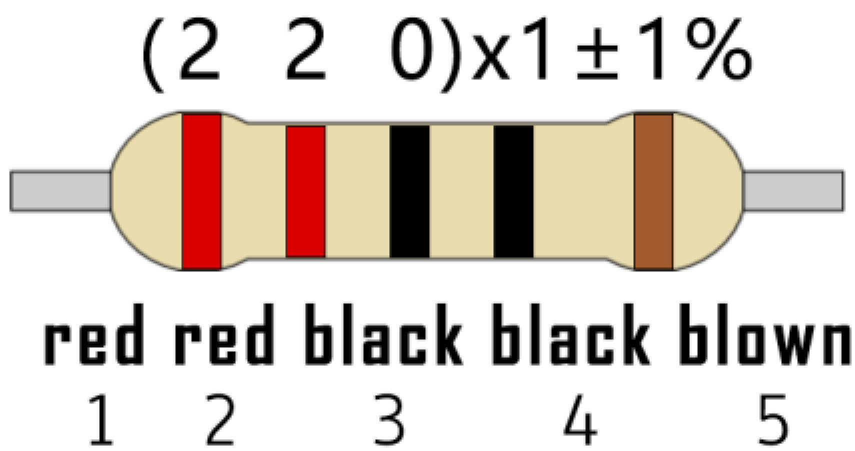


Note:

How to connect a LED

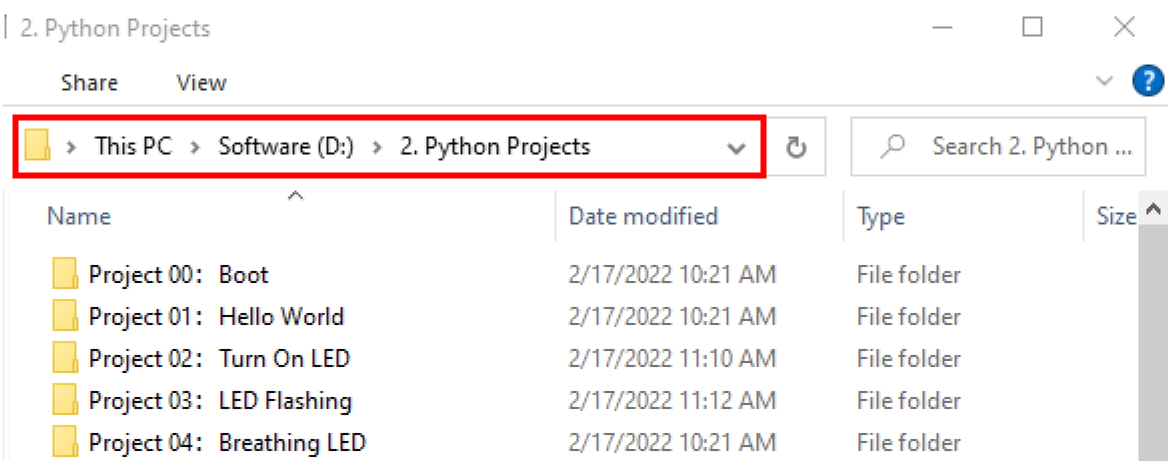


How to identify the 220 Five-color ring resistor

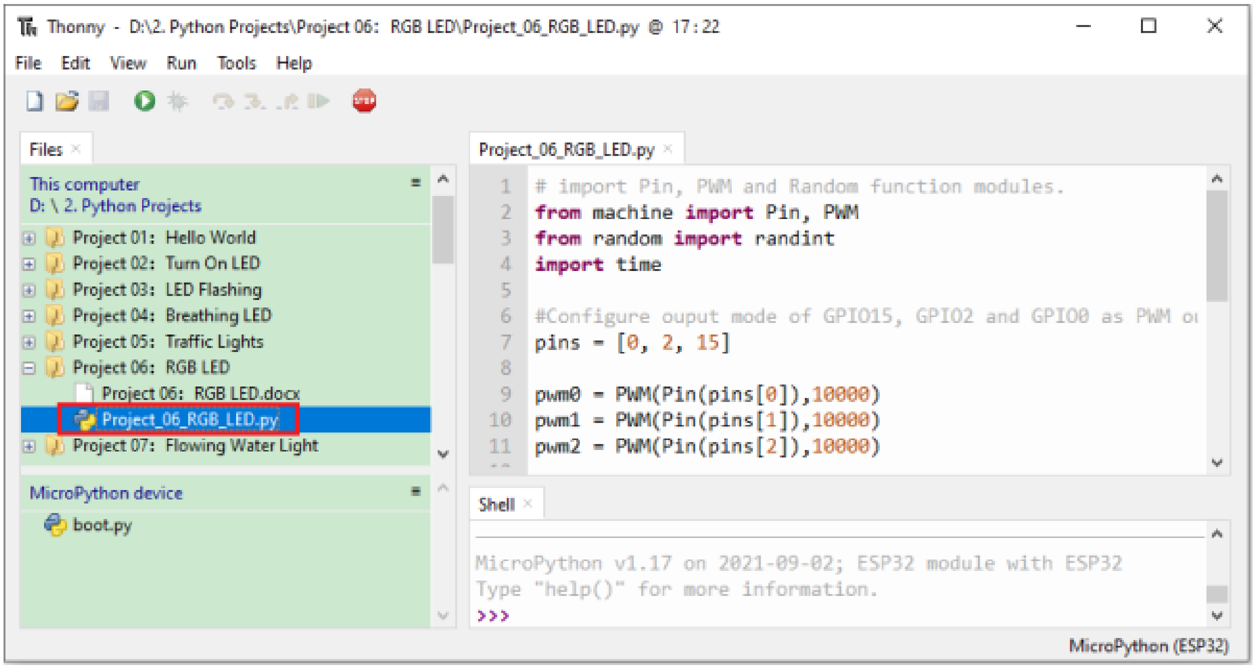


7.7.5 5.Project code

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



Open“Thonny”, click “This computer” → “D:” → “2. Python Projects” → “Project 06RGB LED”, and double left-click “Project_06_RGB_LED.py”.

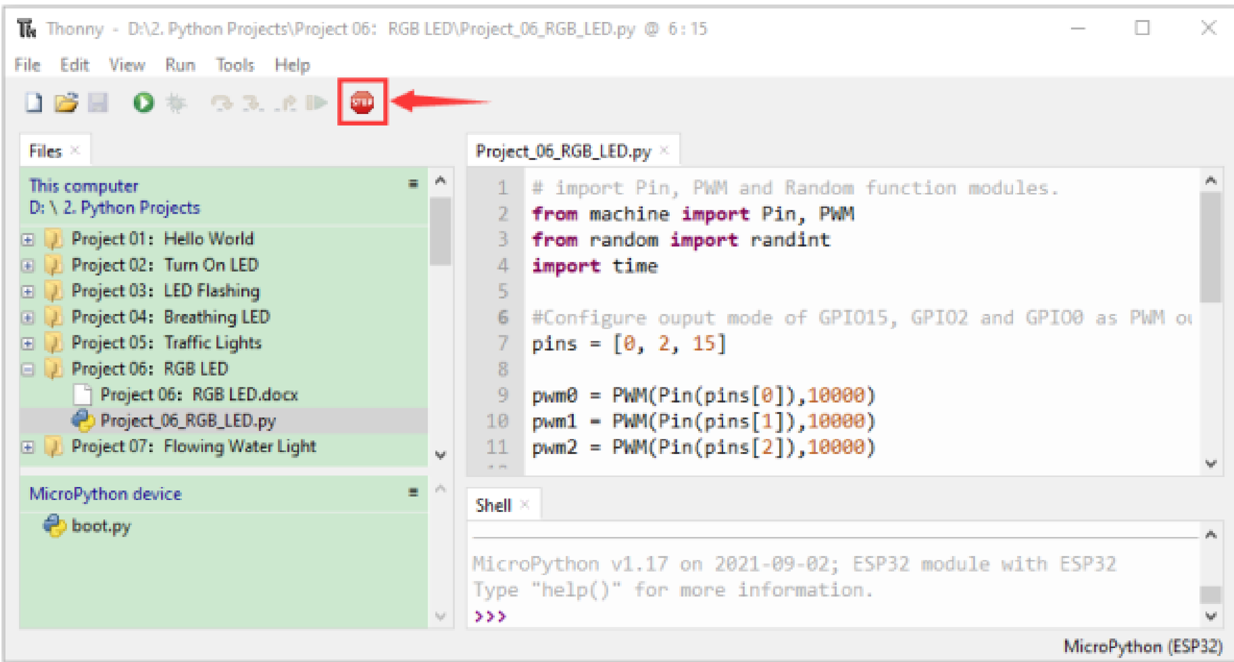



```
## import Pin, PWM and Random function modules.
from machine import Pin, PWM
from random import randint
import time

#Configure ouput mode of GPIO15, GPIO2 and GPIO0 as PWM output and PWM frequency as 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()
```

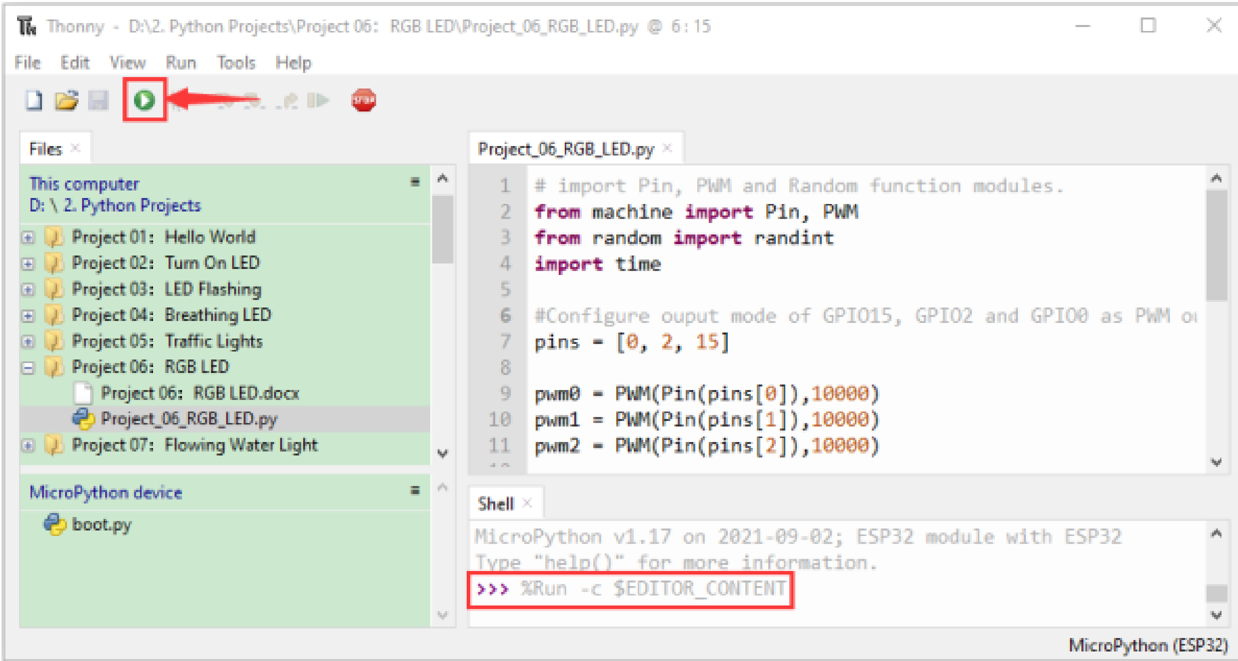
7.7.6 Project result

media/27451c8a9c13e29d02bc0f5831cfaf1f.png

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 RGB begins to display random colors. Press "Ctrl+C" or click  "Stop/Restart backend" to exit the program.

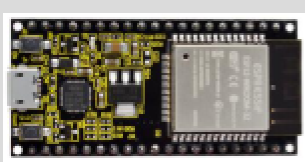
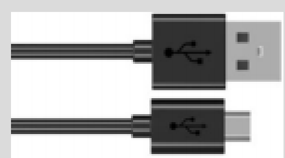
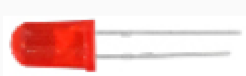
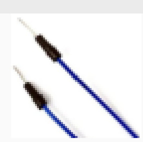


7.8 Project 07: Flowing Water Light

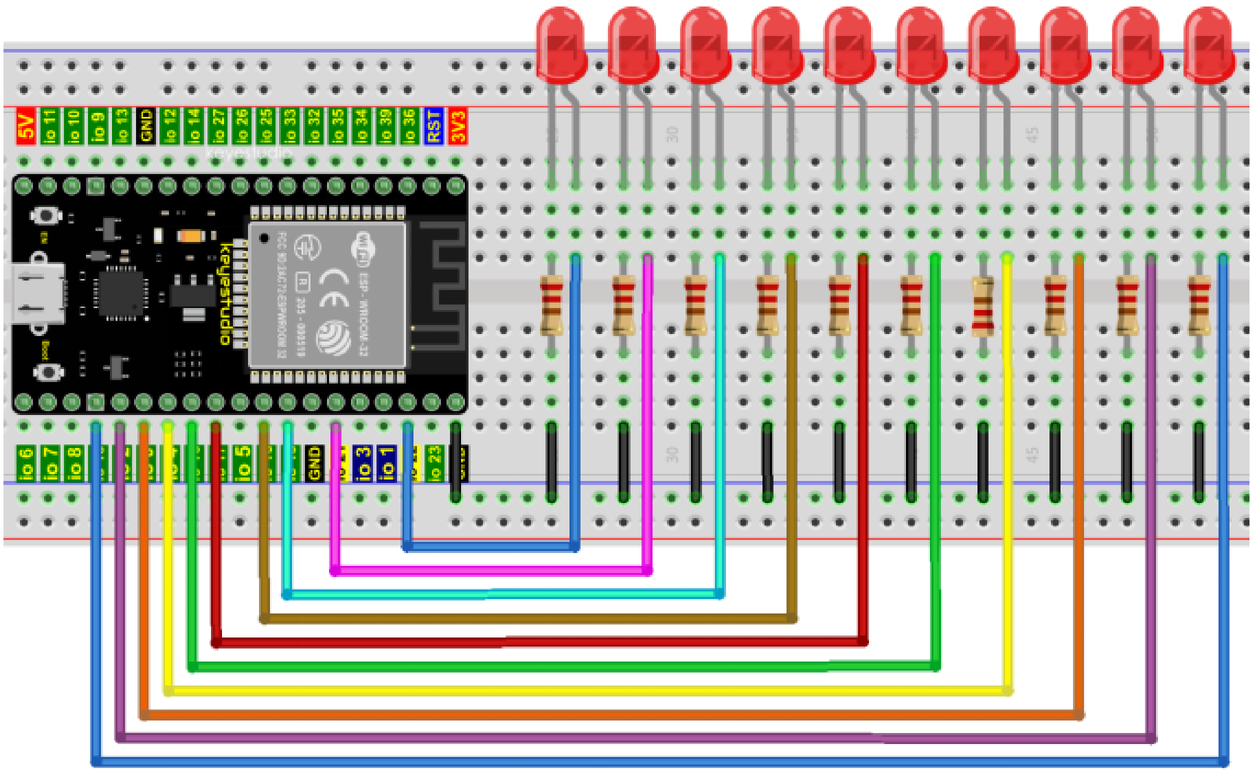
7.8.1 1.Introduction

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

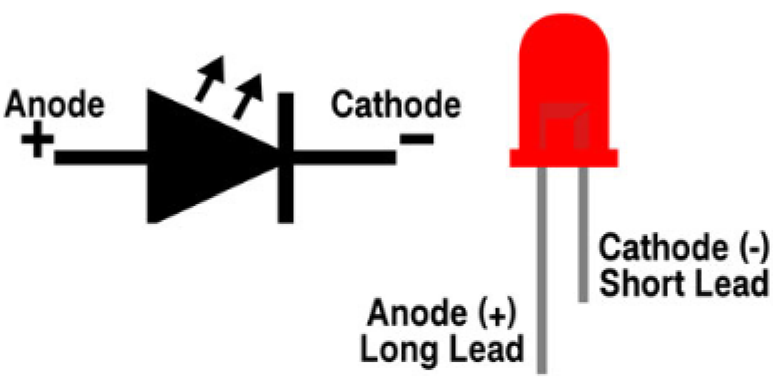
		
ESP32*1	Breadboard*1	USB Cable*1
		
Red LED*10	220 Resistor*10	Jumper Wires

7.8.3 3.Wiring diagram :

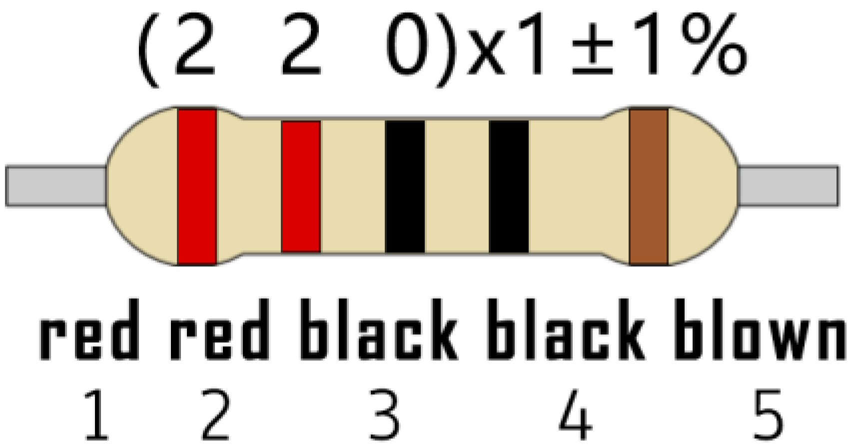


Note:

How to connect a LED



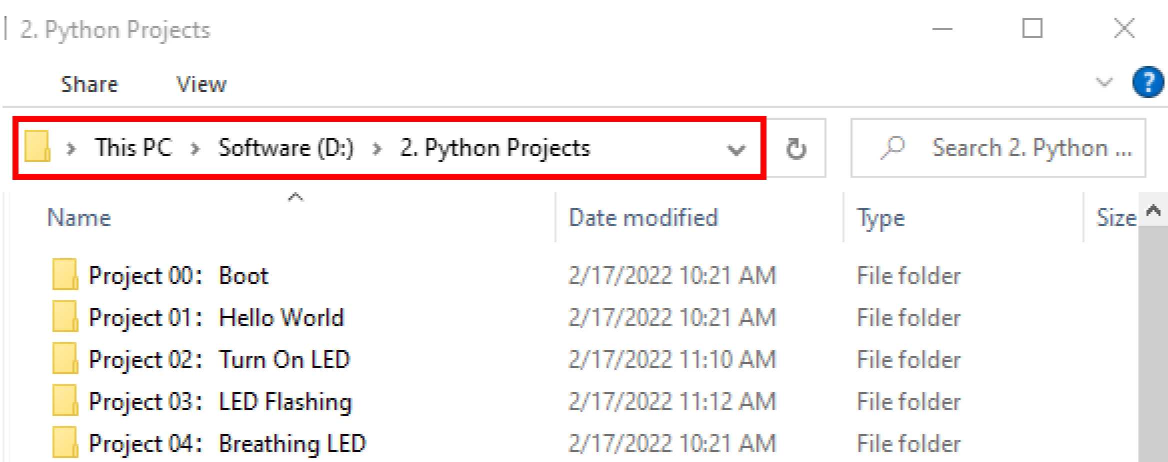
How to identify the 220 Five-color ring resistor



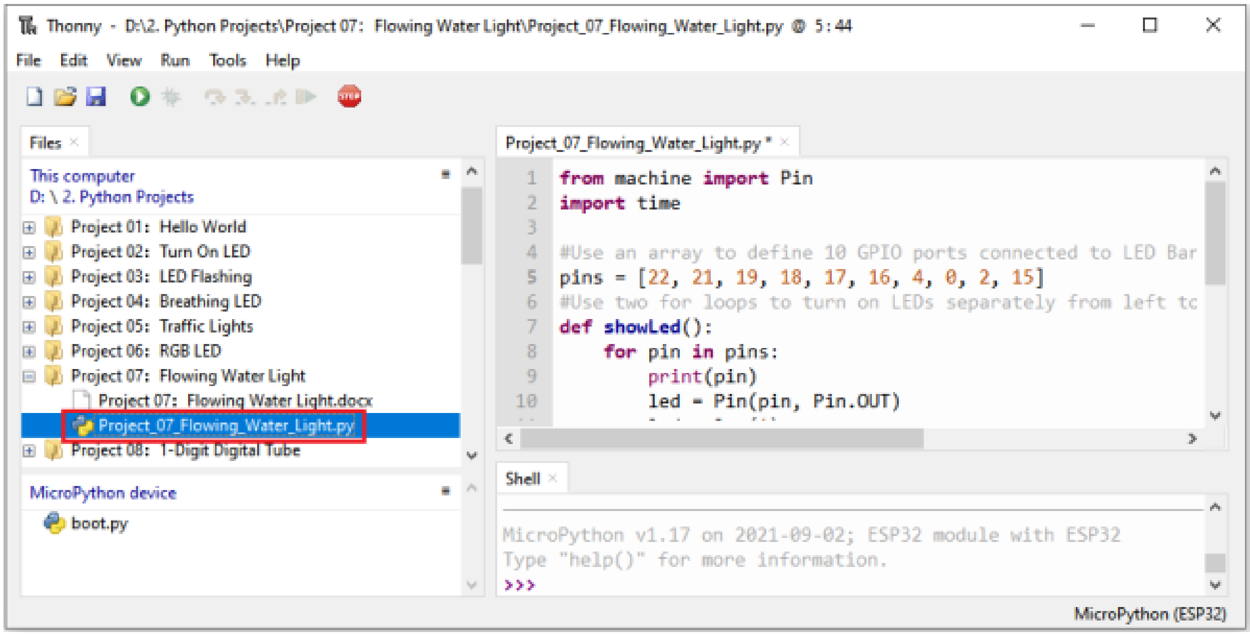
7.8.4 4.Project code

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



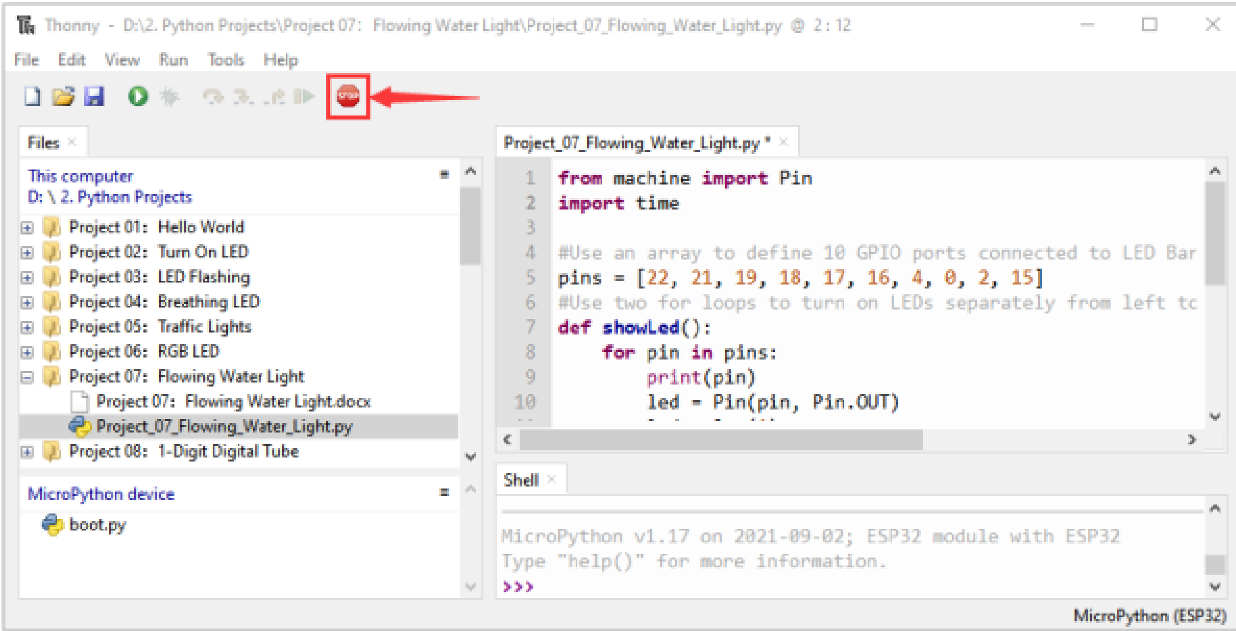
Open"Thonny"click"This computer"→"D:"→"2. Python Projects"→"Project 07Flowing Water Light", and double left-click "Project_07_Flowing_Water_Light.py".



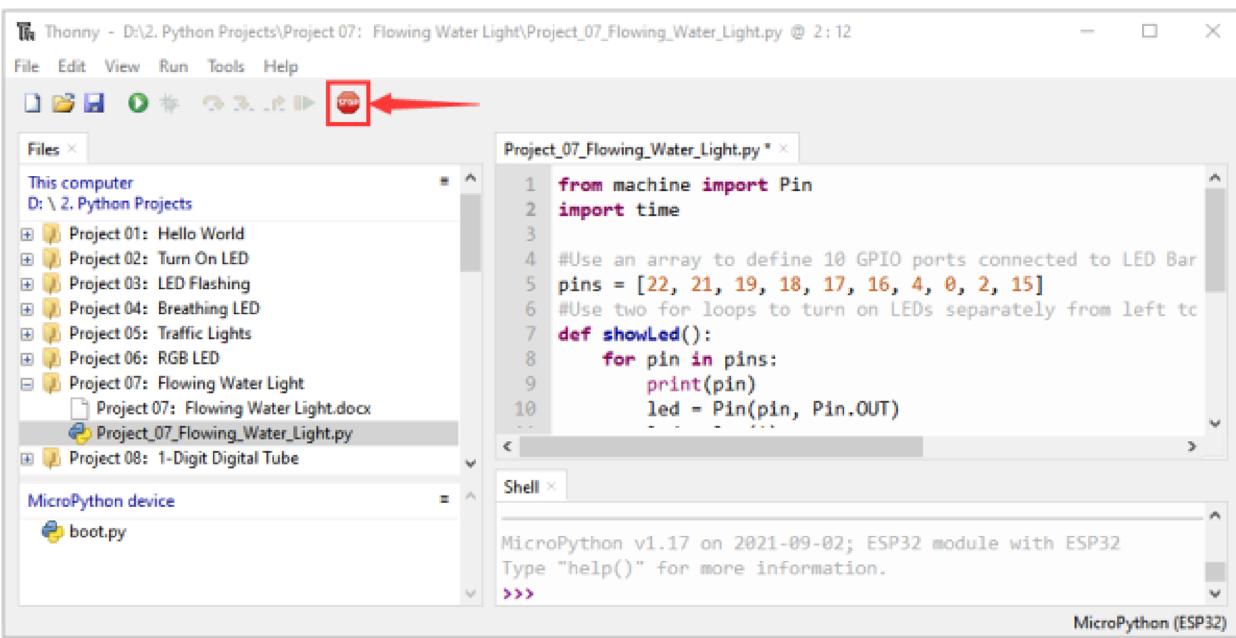

```python
from machine import Pin
import time

#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 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()
```

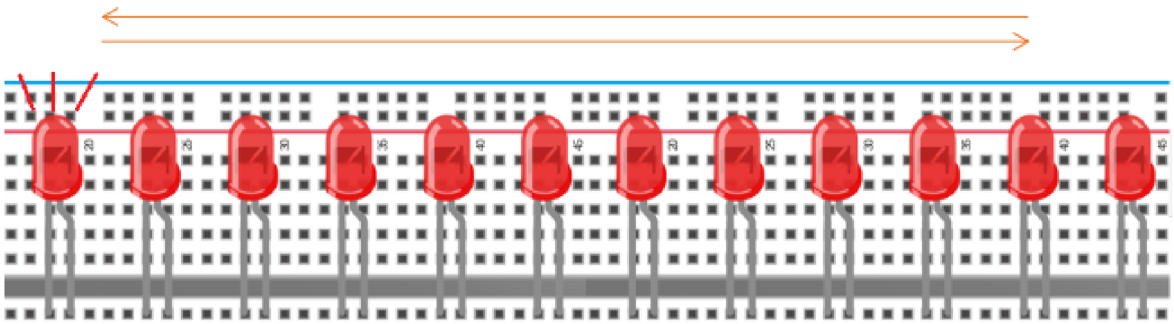
7.8.5 5. Project result

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.



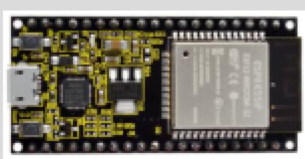
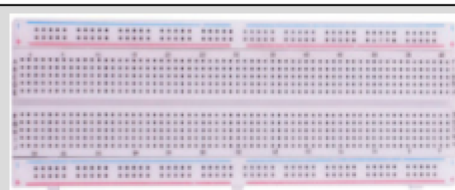
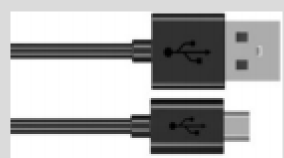
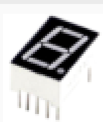
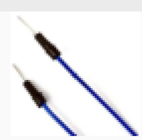


7.9 Project 081-Digit Digital Tube

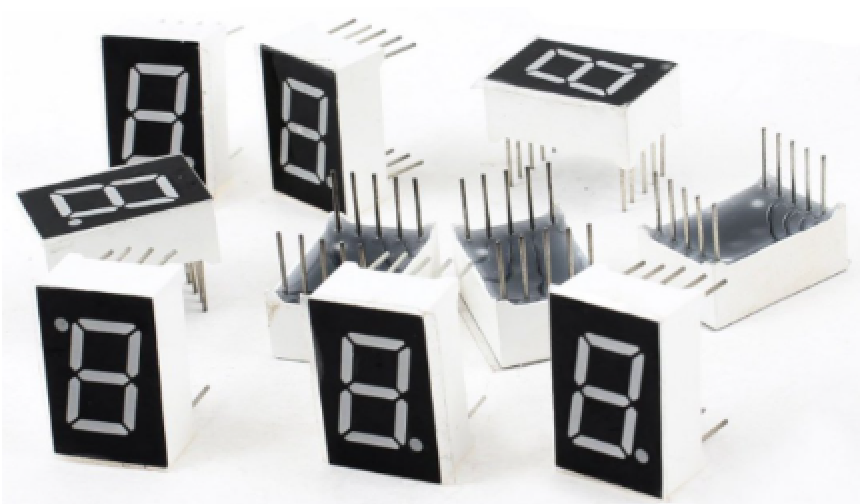
7.9.1 1.Introduction

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

		
ESP32*1	Breadboard*1	USB Cable*1
		
1-Digit 7-Segment Display*1	220 Resistor*8	Jumper Wires

7.9.3 3. Component knowledge

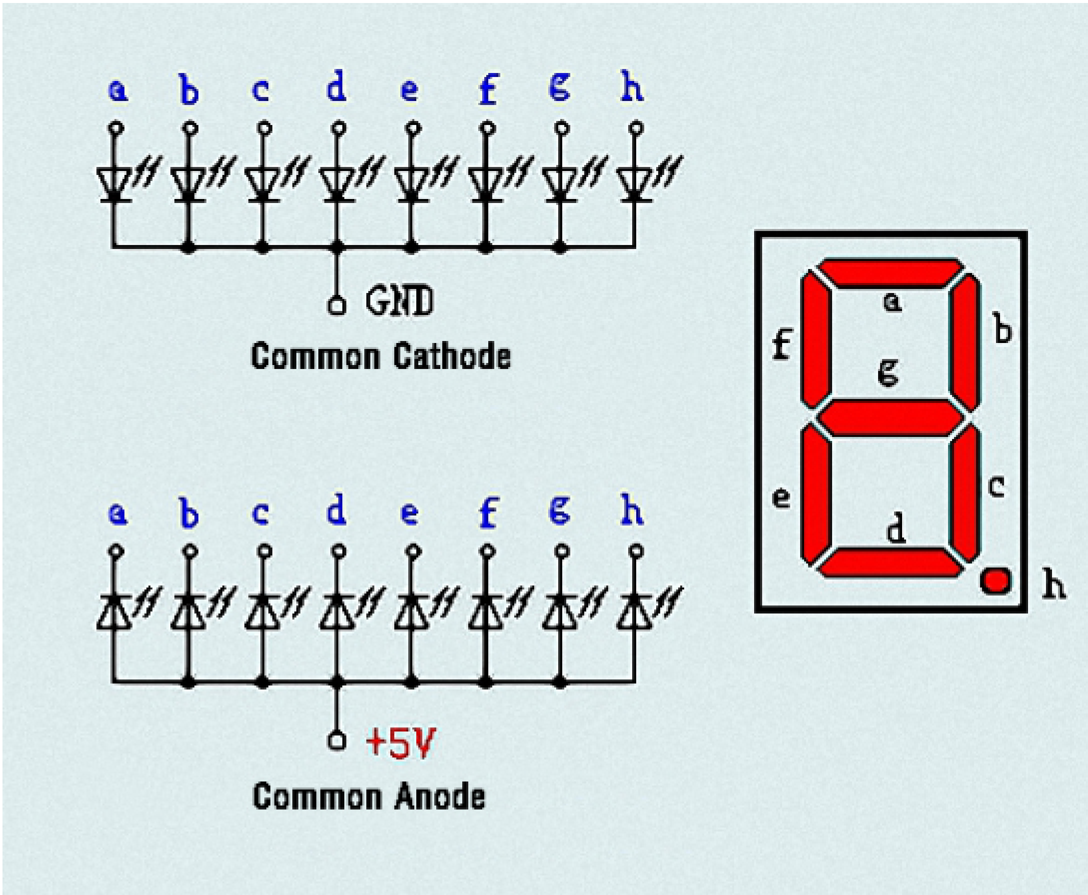


1-Digit 7-Segment Display principle:

Digital tube display is a semiconductor light emitting device,its basic unit is a light-emitting diode (LED). Thedigital 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".

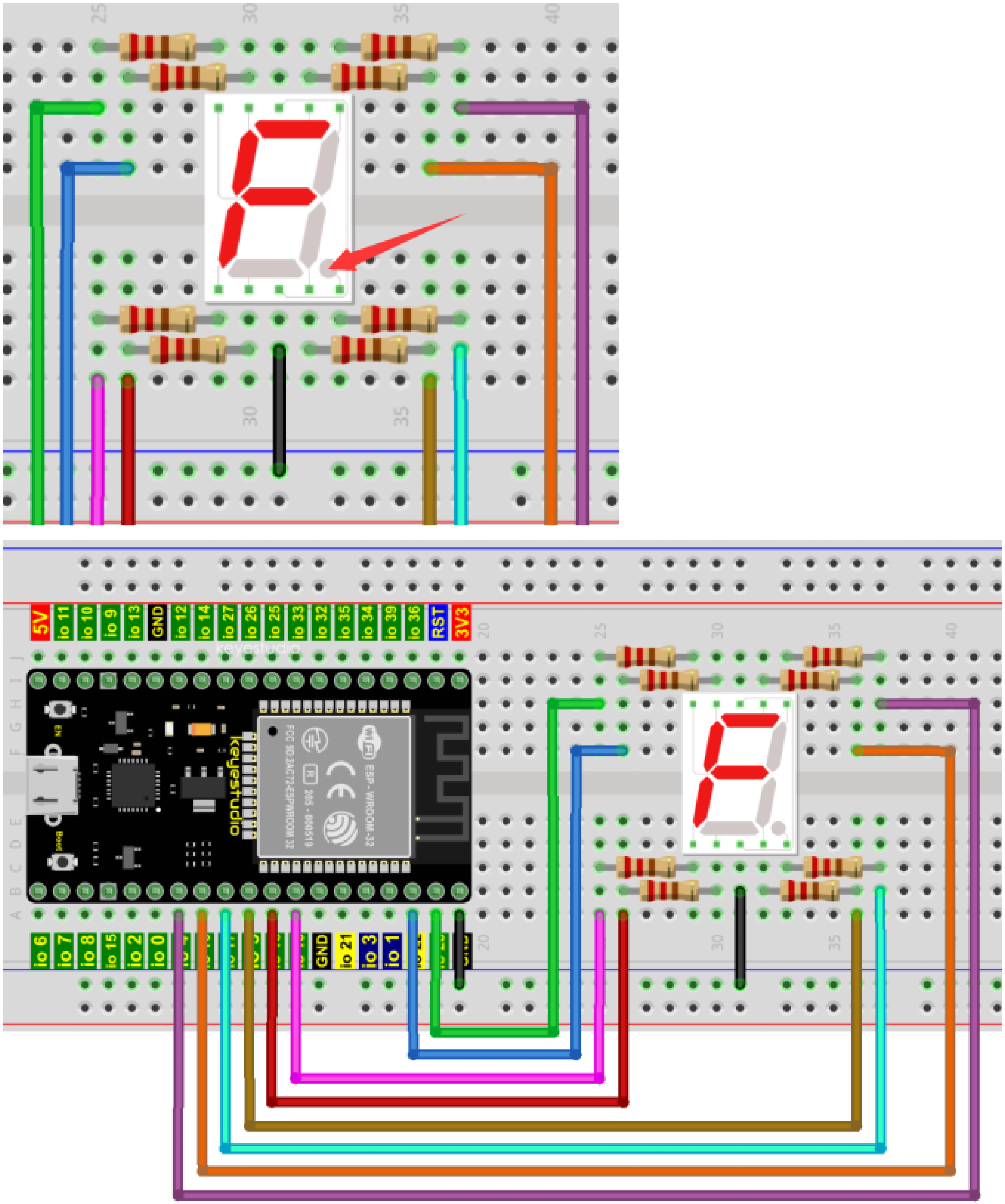
In the common anode 7-segment display, the LED anodes (positive electrodes) of all segments are connected together, so you should connect the common anode to"+5V". To light up a segmented LED, you can set its associated pin to"LOW".



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

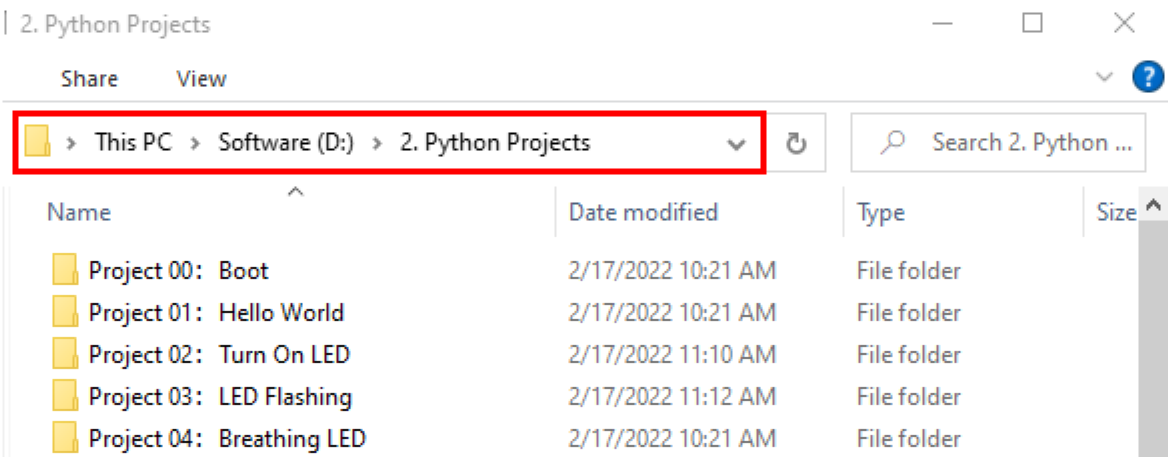
Note: The direction of the 7-segment display inserted into the breadboard is consistent with the wiring diagram, with one more point in the lower right corner.


5V
io 11
io 10
io 9
io 13
GND
io 12
io 14
io 27
io 26
io 25
io 33
io 32
io 35
io 34
io 39
io 36
RST
3V3
keyestudio
io 6
io 7
io 8
io 15
io 2
io 0
GND
io 21
io 3
io 1

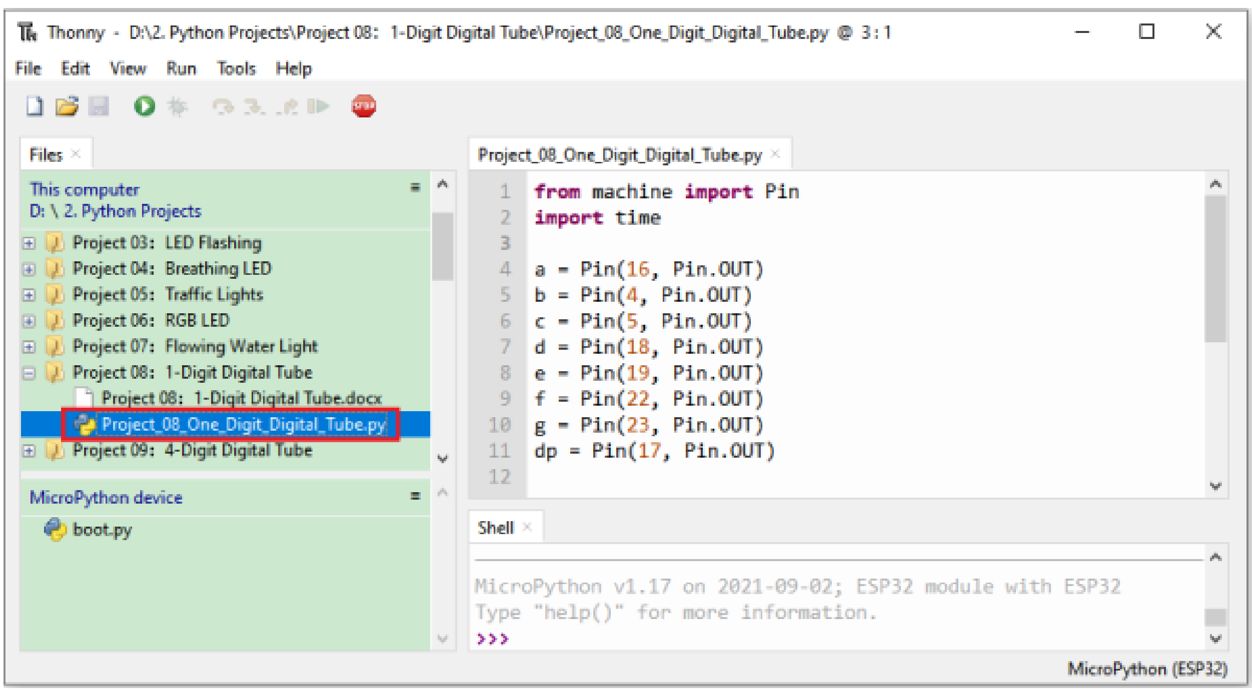
7.9.5 5.Project code

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



Open"Thonny"click"This computer"→"D:"→"2. Python Projects"→"Project 081-Digit Digital Tube", and double left-click "Project_08_One_Digit_Digital_Tube.py".




```
from machine import Pin
import time

a = Pin(16, Pin.OUT)
b = Pin(4, Pin.OUT)
```

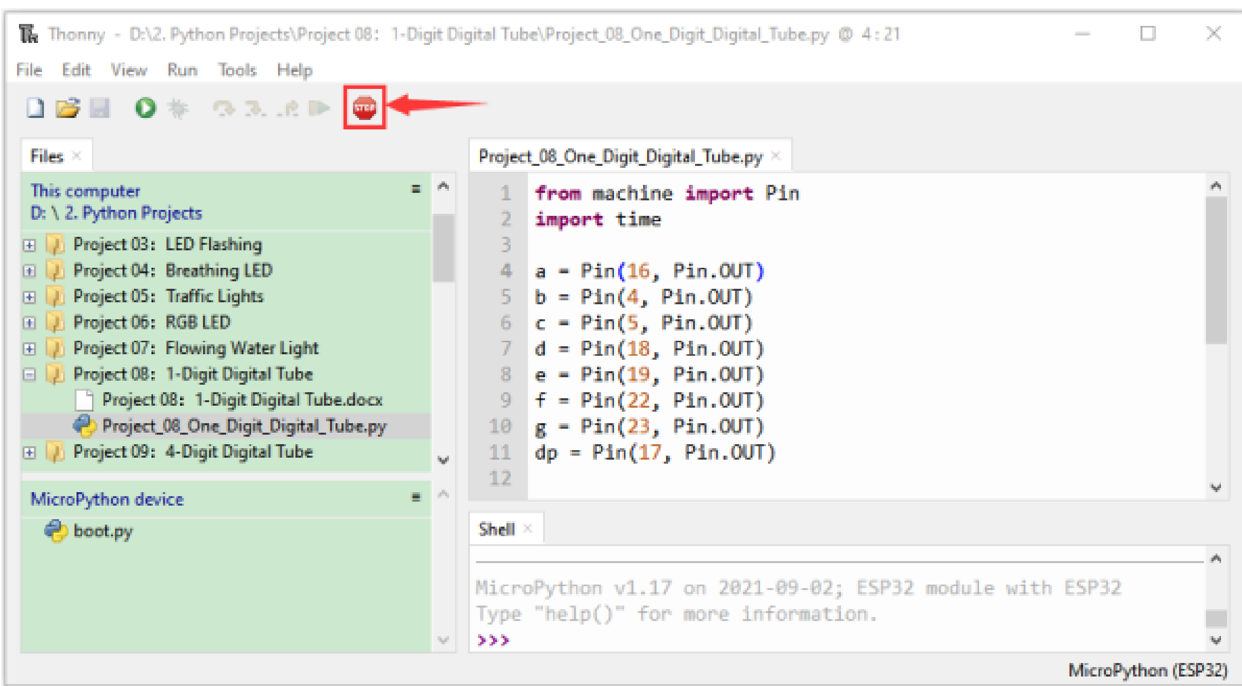
(continues on next page)

(continued from previous page)

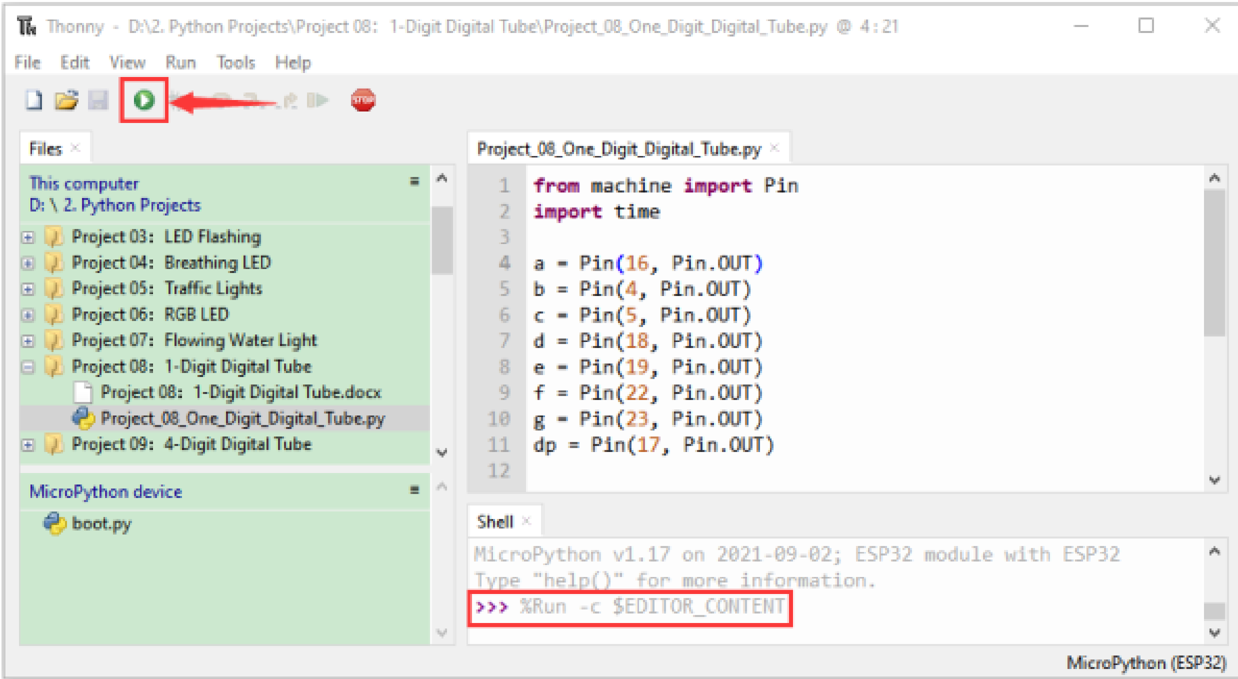
```
c = Pin(5, Pin.OUT)
d = Pin(18, Pin.OUT)
e = Pin(19, Pin.OUT)
f = Pin(22, Pin.OUT)
g = Pin(23, Pin.OUT)
dp = Pin(17, Pin.OUT)

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 "Stop/Restart backend".



Click "Run current script", the code starts to be executed and you'll see that the 1-Digit 7-Segment Display will display numbers from 9 to 0. Press "Ctrl+C" or click "Stop/Restart backend" to exit the program.

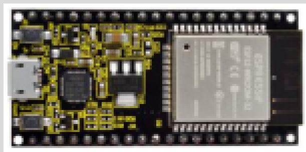
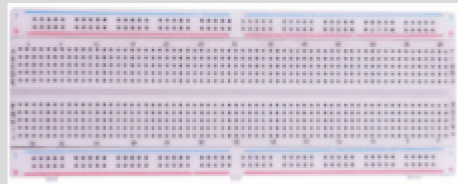
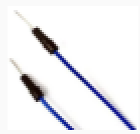


7.10 Project 094-Digit Digital Tube

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

	
ESP32*1	Breadboard*1
	
4-digit 7-segment display Module*1	M-F Dupont Wires

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.

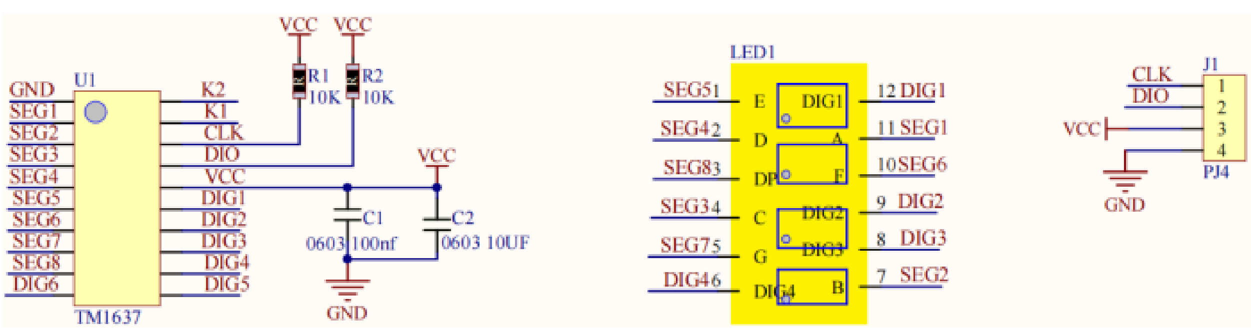
Specifications of 4-bit 7-segment display module:

Working voltage: DC 3.3V-5V

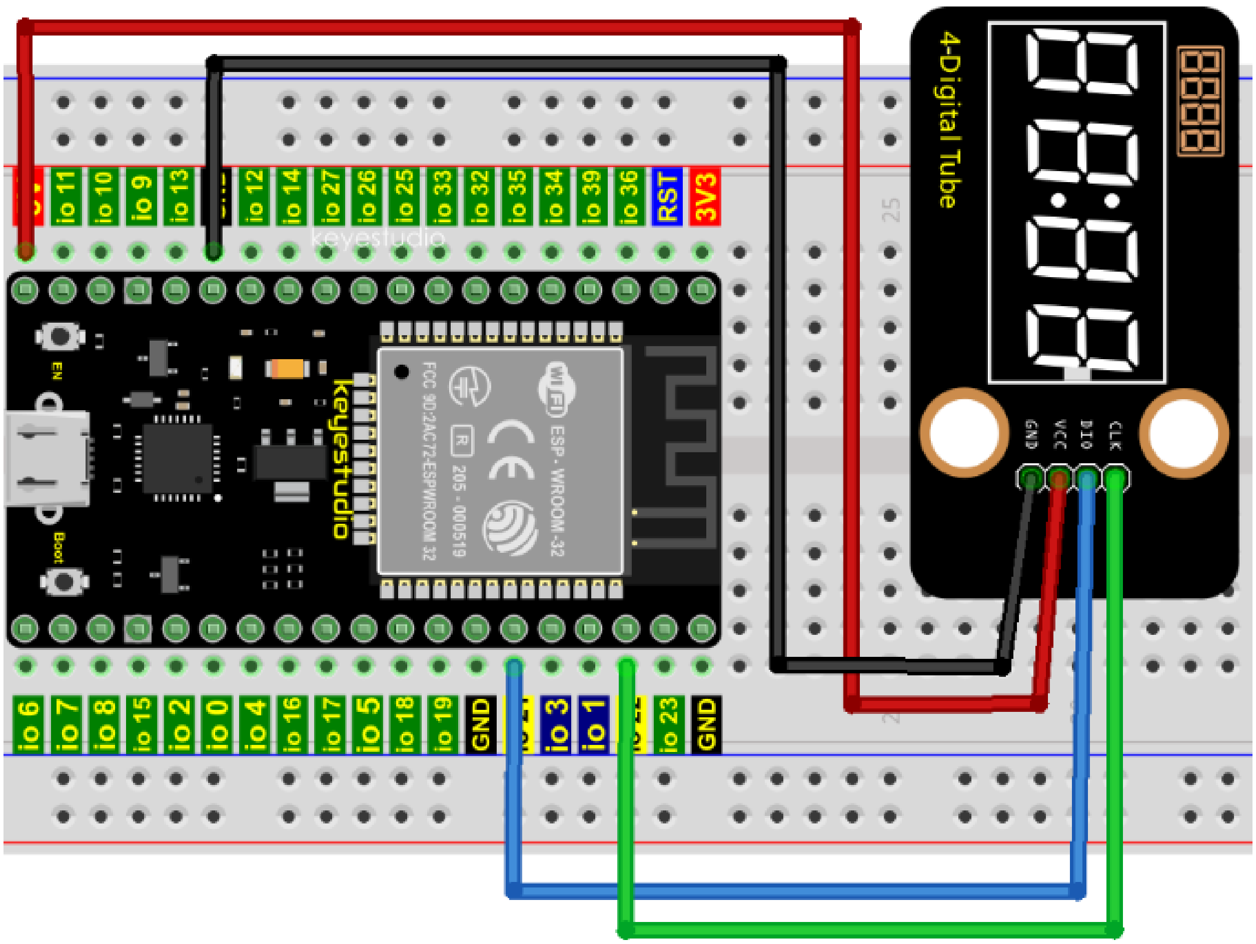
Maximum current: 100MA

Maximum power: 0.5W

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

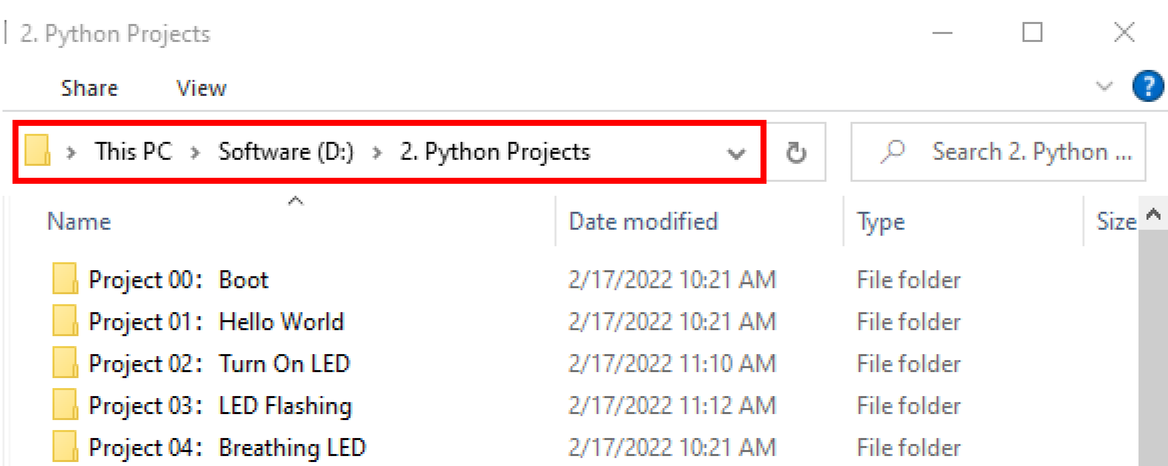


7.10.4 4.Wiring diagram

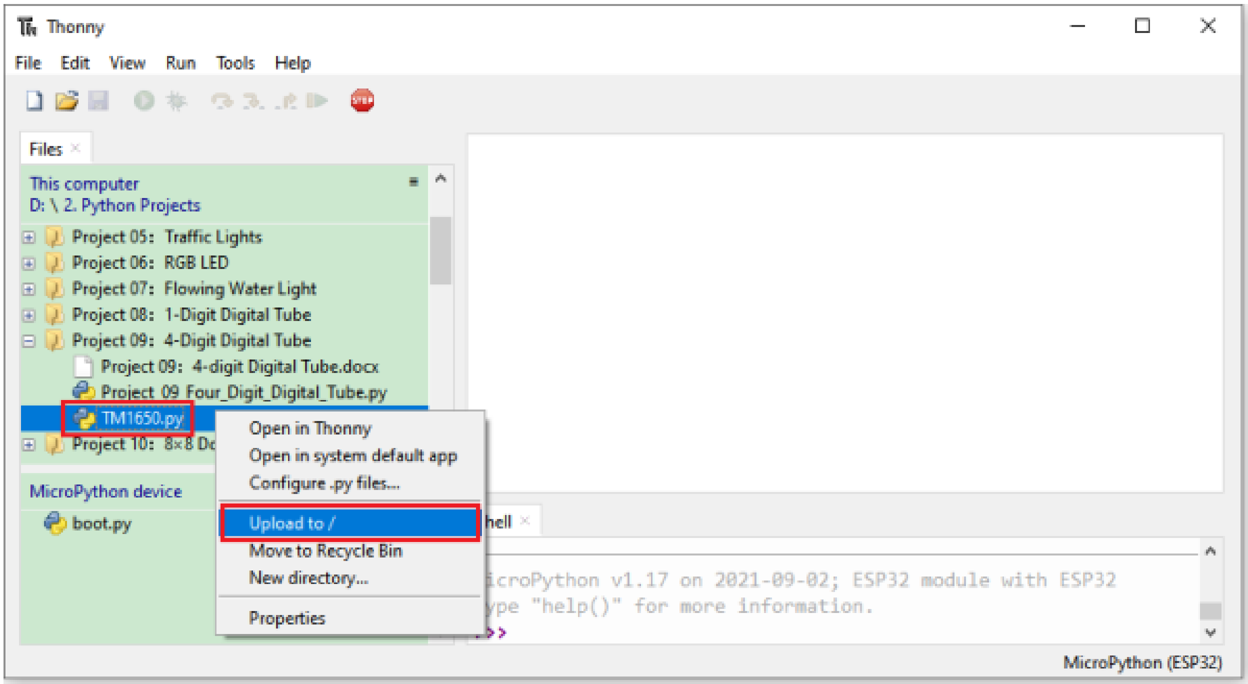


7.10.5 5. Project code

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"→"D:"→"2. Python Projects"→"Project 094-Digit Digital Tube". and then double left-click "Project_09_Four_Digit_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
```

(continues on next page)

(continued from previous page)

```
def writeByte(wr_data):
    global clk,dio
    for i in range(8):
        if(wr_data & 0x80 == 0x80):
            dio.value(1)
        else:
            dio.value(0)
        clk.value(0)
        time.sleep(0.0001)
        clk.value(1)
        time.sleep(0.0001)
        clk.value(0)
        wr_data <<= 1
    return

def start():
    global clk,dio
    dio.value(1)
    clk.value(1)
    time.sleep(0.0001)
    dio.value(0)
    return

def ack():
    global clk,dio
    dy = 0
    clk.value(0)
    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(0)
    dio = Pin(dioPin, Pin.OUT)
    return

def stop():
    global clk,dio
    dio.value(0)
    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
```

(continues on next page)

(continued from previous page)

```
    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)
    return

def displayOnOFF(OnOff = 1):
    global DisplayCommand
    DisplayCommand = (DisplayCommand & 0xfe)+OnOff
    return

def displayDot(bit, OnOff):
    if(bit > 4):
```

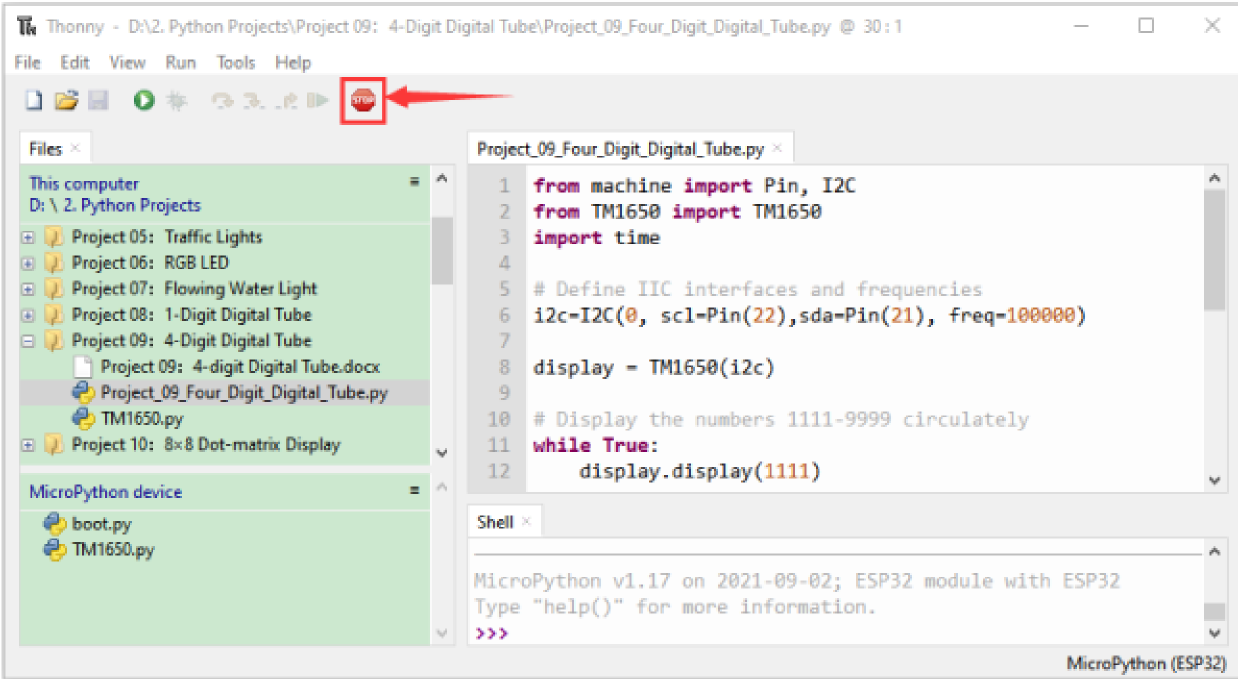
(continues on next page)

(continued from previous page)

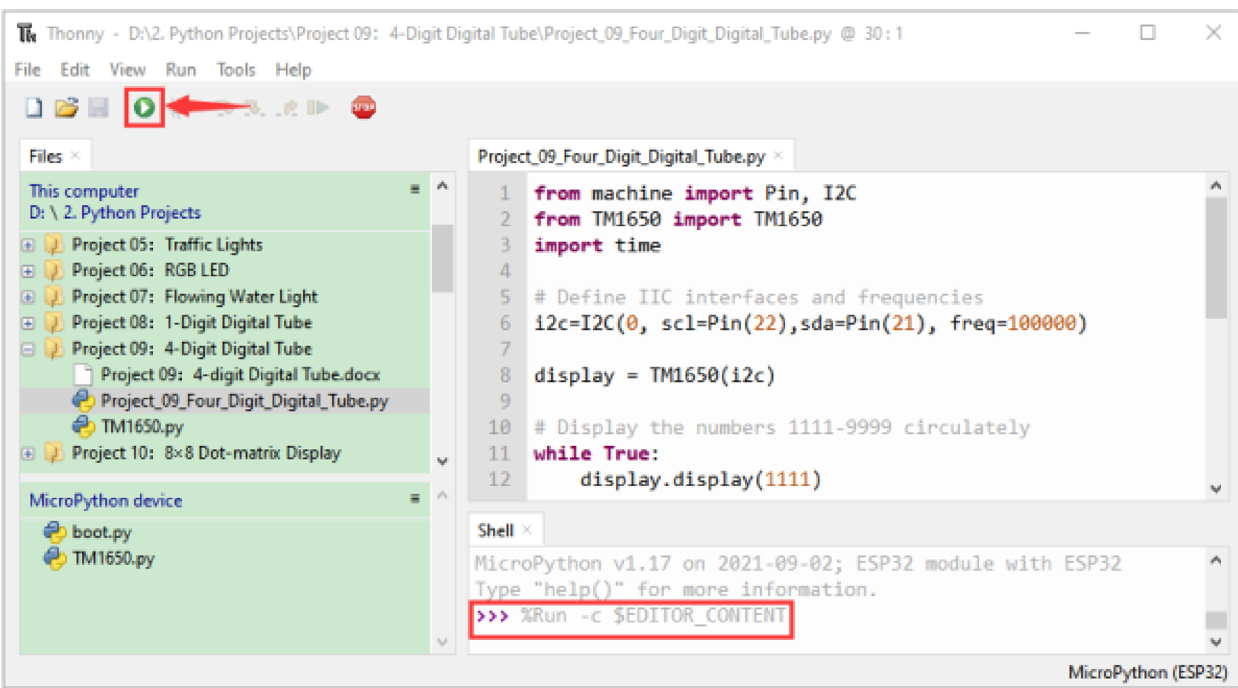
```
        return
    if(OnOff == 1):
        DOT[bit-1] = 1;
    else:
        DOT[bit-1] = 0;
    return

def InitDigitalTube():
    setBrightness(2)
    setMode(0)
    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 ␣
↪number=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 "Run current script", the code starts to be executed and you'll see that 4-digit 7-segment display displays digitsthe number increments by one and repeat these actions in an infinite loop. Press "Ctrl+C" or click "Stop/Restart backend" to exit the program.

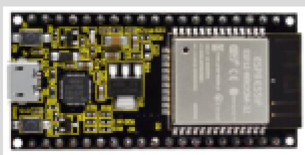
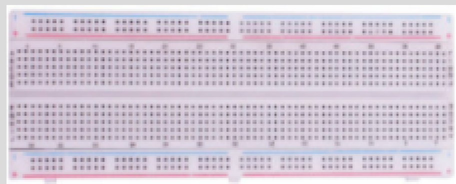
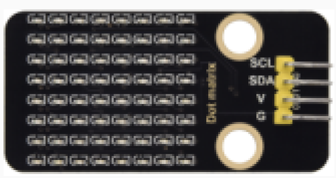
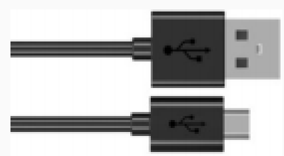


7.11 Project 108×8 Dot-matrix Display

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

		
ESP32*1	Breadboard*1	
		
8*8 dot matrix module*1	M-F Dupont Wires	USB Cable*1

7.11.3 3.Component knowledge

8*8 dot matrix module

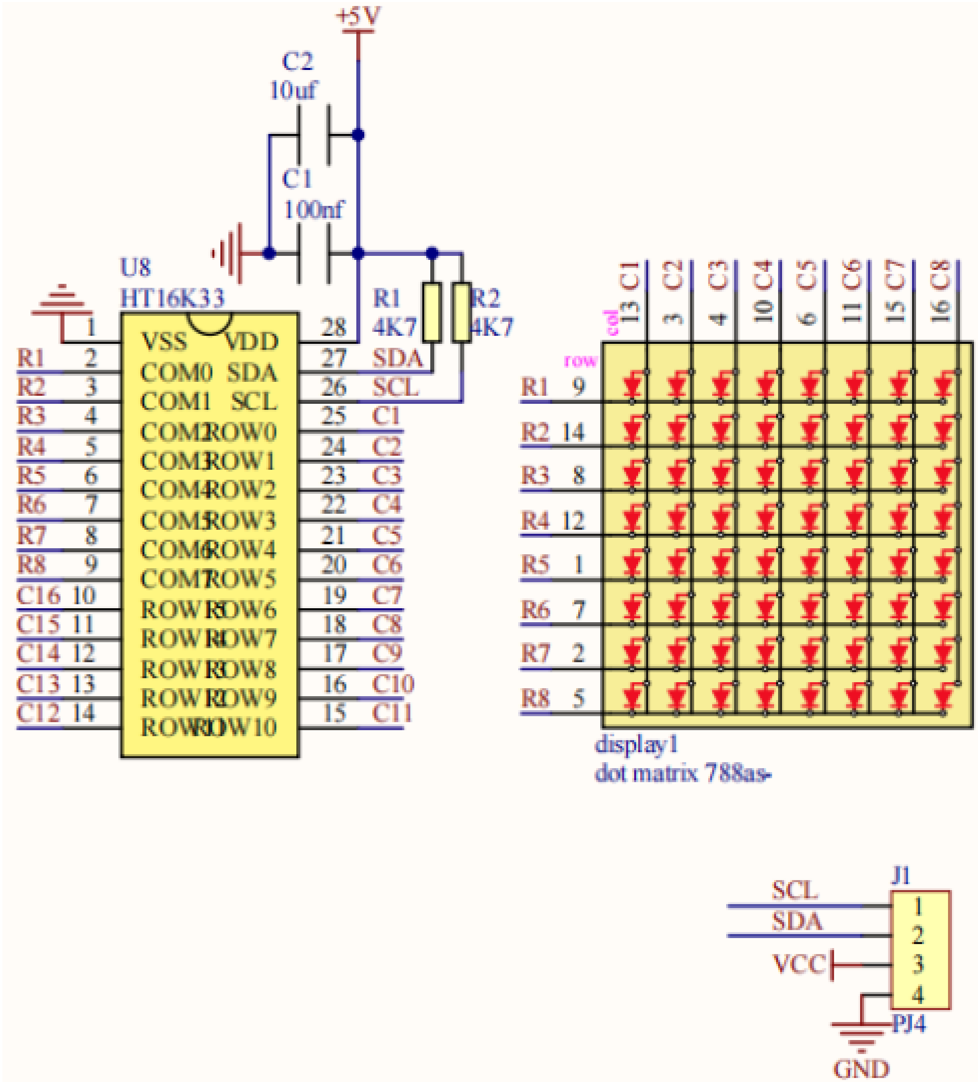
The 8*8 dot matrix is composed of 64 LEDs, and each LED is placed at the intersection of a row and a column. When using the single chip microcomputer to drive an 8*8 dot matrix, we need 16 digital ports in total, which greatly wastes the data of the single chip microcomputer.

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

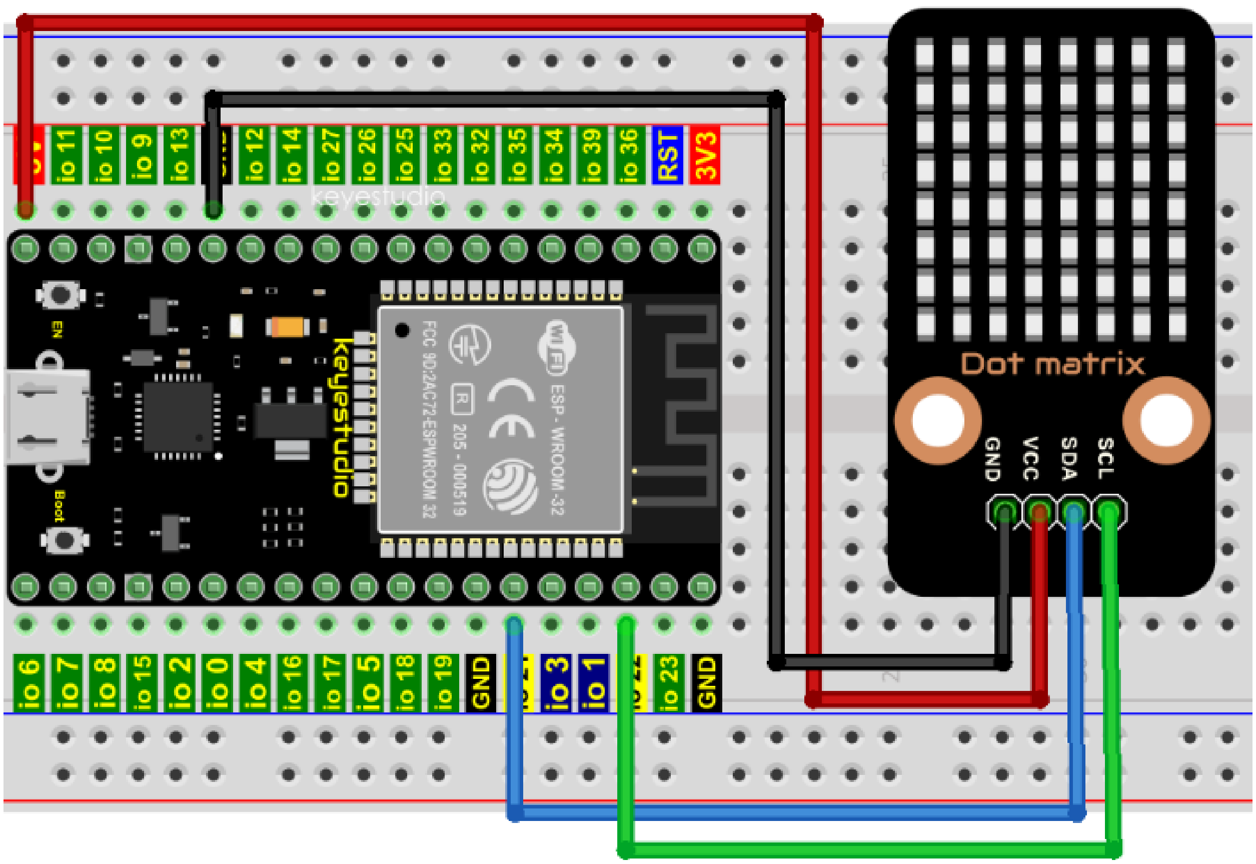
Schematic diagram of 8*8 dot matrix module



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.

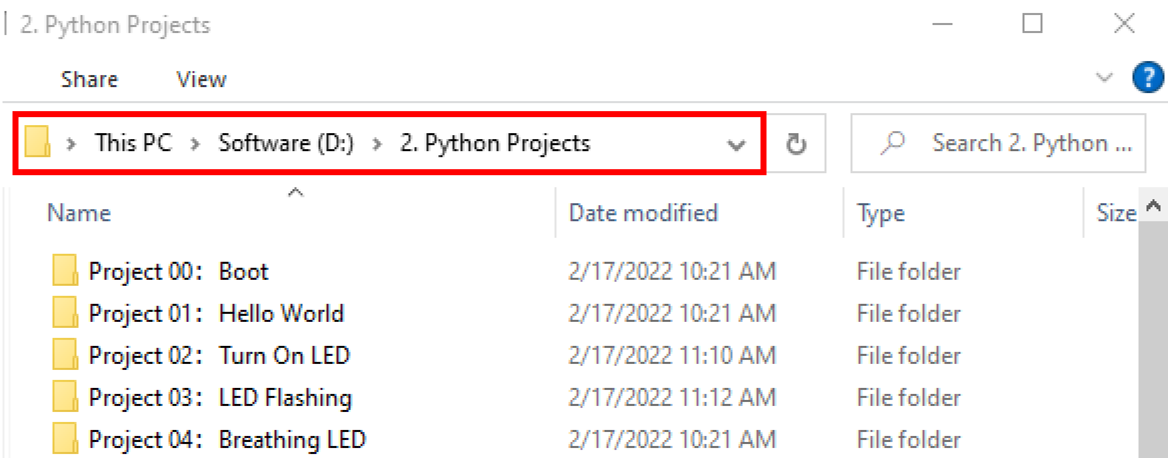
A0 (1)	A1 (2)	A2 (3)	A0 (1)	A1 (2)	A2 (3)	A0 (1)	A1 (2)	A2 (3)
0(OFF)	0 (OFF)	0 (OFF)	1 (ON)	0 (OFF)	0 (OFF)	0 (OFF)	1 (ON)	0(OFF)
0X70			0X71			0X72		
A0 (1)	A1 (2)	A2 (3)	A0 (1)	A1 (2)	A2 (3)	A0 (1)	A1 (2)	A2 (3)
1 (ON)	1 (ON)	0 (OFF)	0 (OFF)	0 (OFF)	1 (ON)	1 (ON)	0 (OFF)	1 (ON)
0X73			0X74			0X75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

7.11.4 4.Wiring diagram



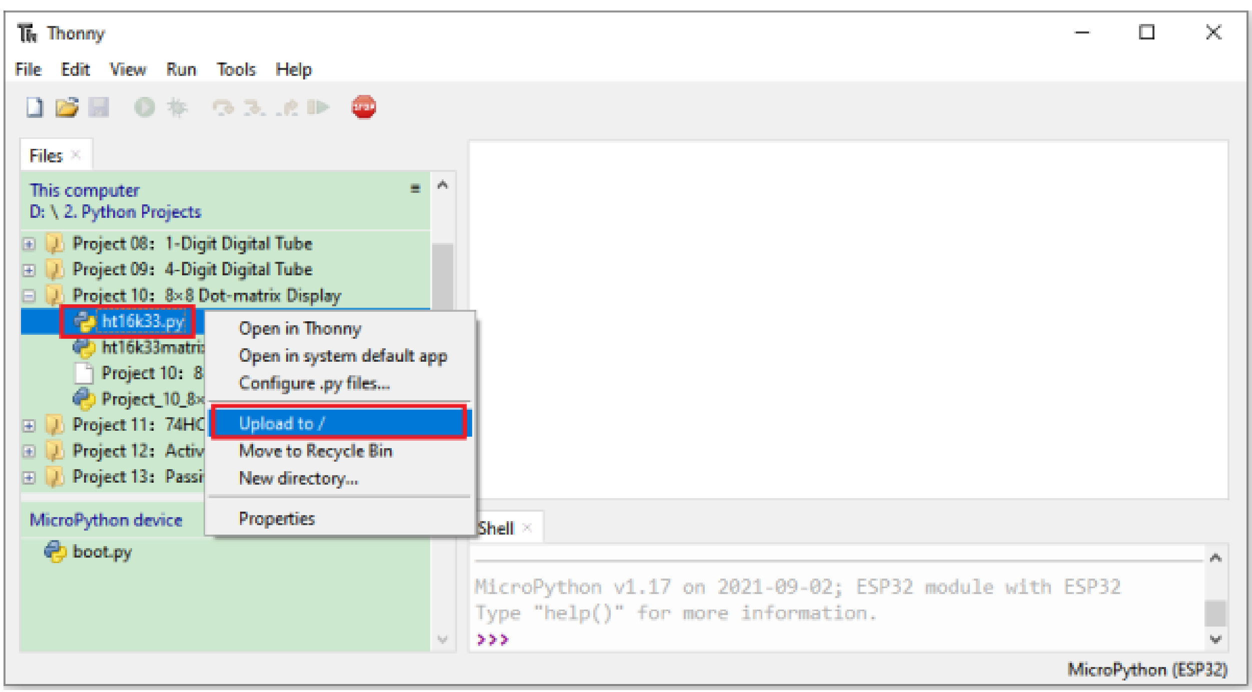
7.11.5 5.Project code

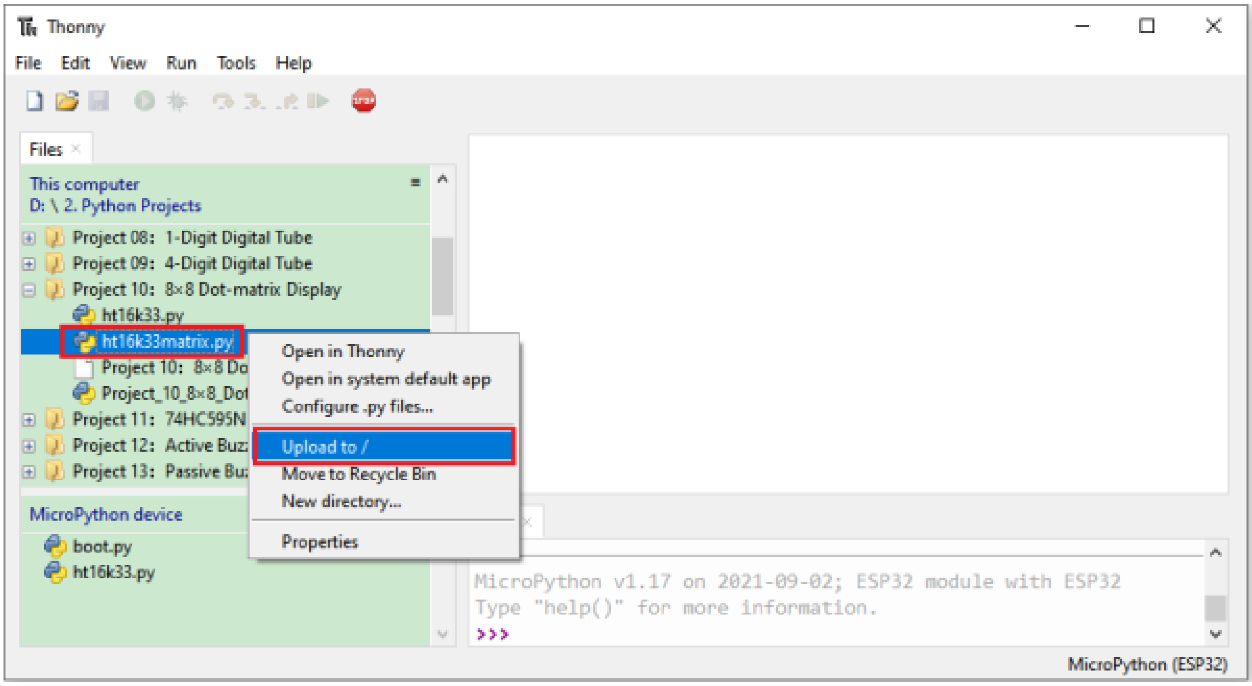
Codes used in this tutorial are saved in“**2. Python Projects**”. If you haven’t downloaded the code file, please click on the link to download it:Download Python Codes

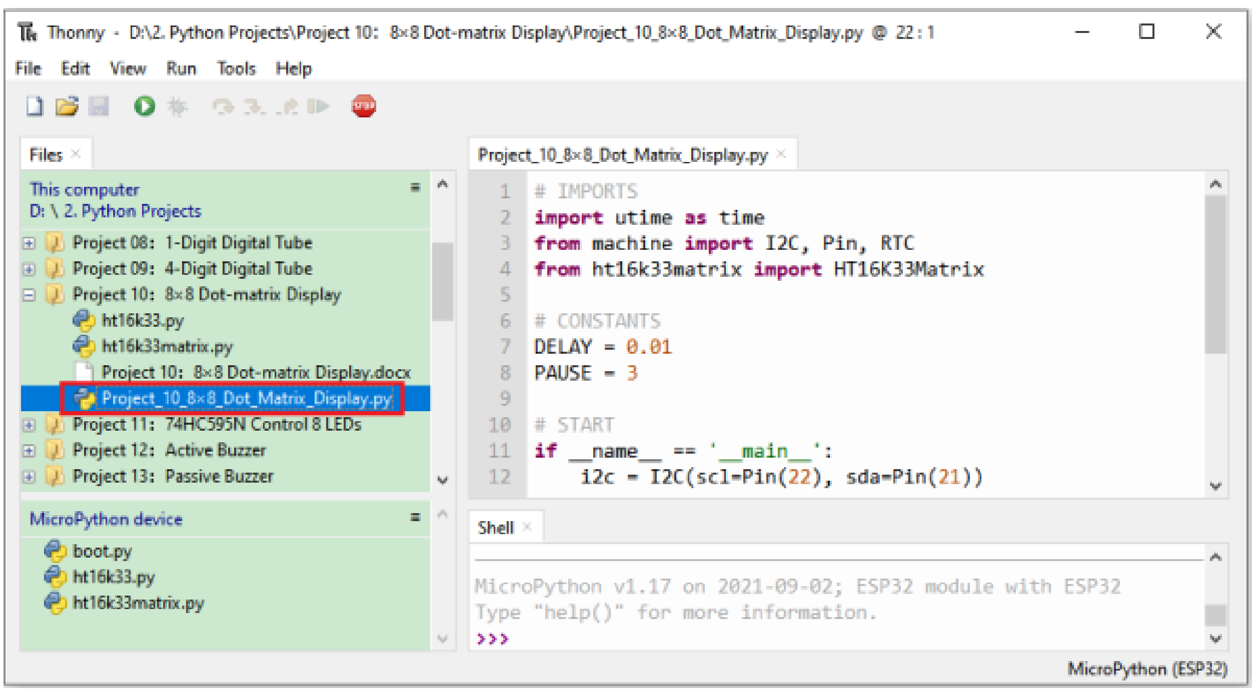


Open“Thonny”click“This computer”→“D:”→“2. Python Projects”→“Project 108×8 Dot-matrix Display”.

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”.








```python
## IMPORTS
import utime as time
from machine import I2C, Pin, RTC
from ht16k33matrix import HT16K33Matrix

## CONSTANTS
DELAY = 0.01
PAUSE = 3

```

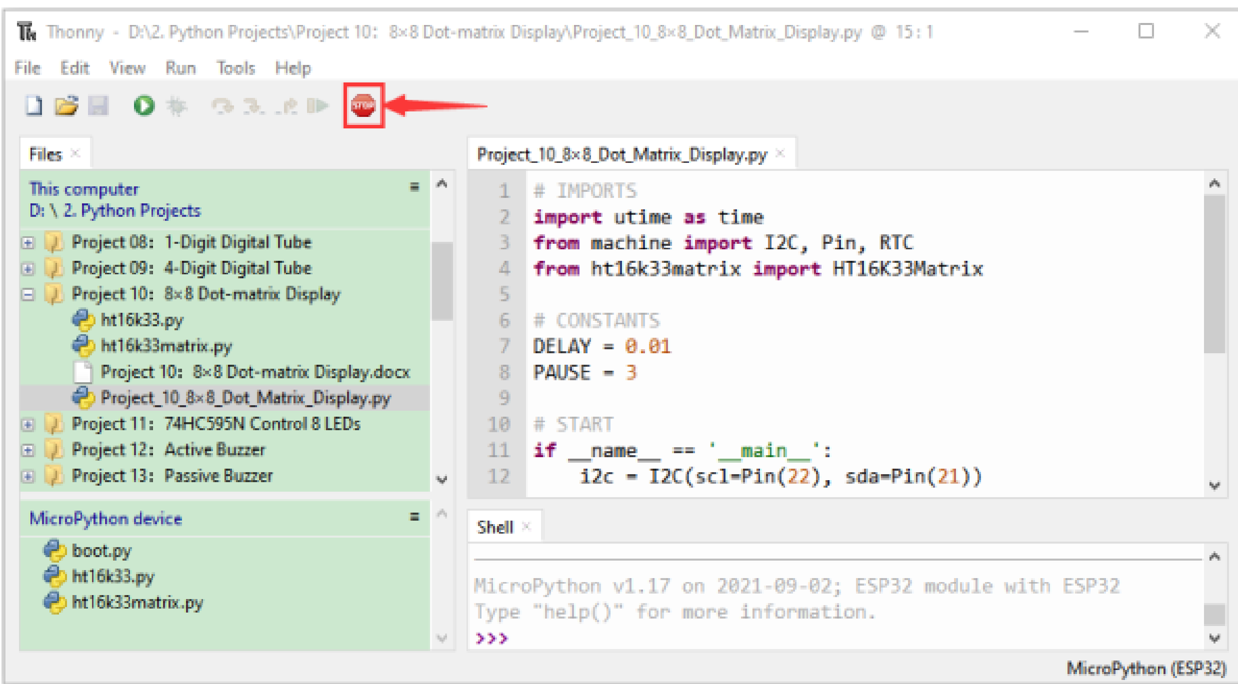
(continues on next page)

(continued from previous page)

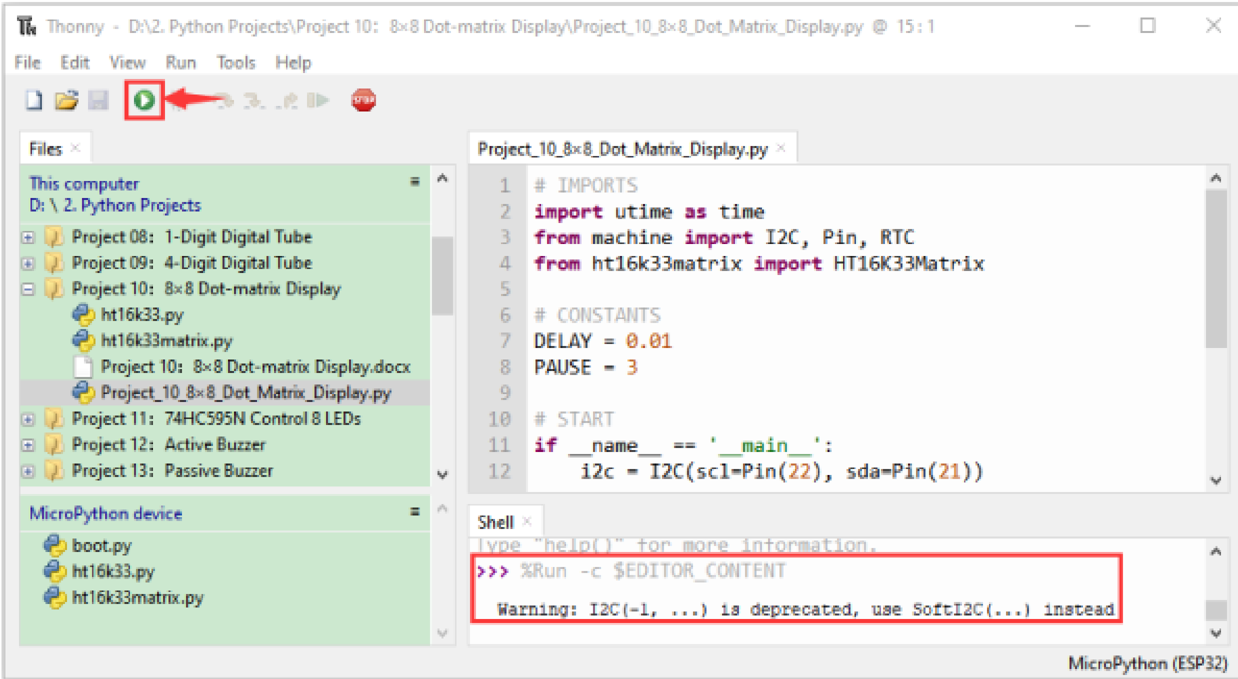
```
## START
if __name__ == '__main__':
    i2c = I2C(scl=Pin(22), sda=Pin(21))
    display = HT16K33Matrix(i2c)
    display.set_brightness(2)

    # 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

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 8*8 dot matrix displays"Smiling face" pattern. Press"Ctrl+C" or click "Stop/Restart backend"to exit the program.



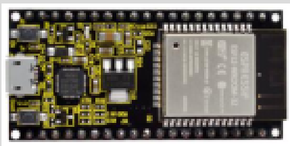
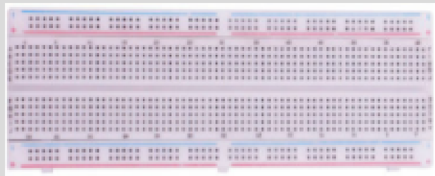
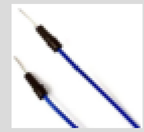
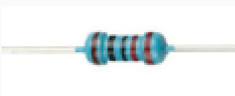
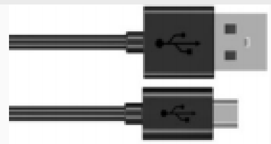
7.12 Project 1174HC595N Control 8 LEDs

7.12.1 1.Introduction

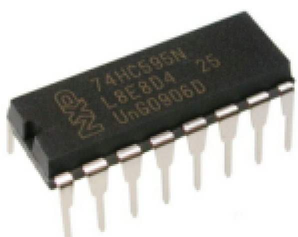
In previous projects, we learned how to light up an LED. With only 32 IO ports on ESP32, how do we light up a lot of leds? Sometimes it is possible to run out of pins on the ESP32, and you need to extend it with the shift register.You can use the 74HC595N chip to control 8 outputs at a time, taking up 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

			
ESP32*1	Breadboard*1	74HC595N chip*1	Jumper Wires
			
220Resistor*8	Red LED*8	USB Cable*1	

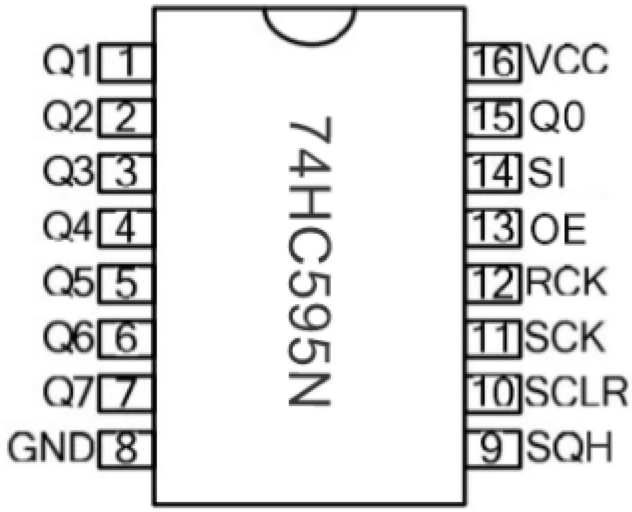
7.12.3 3.Component knowledge



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.

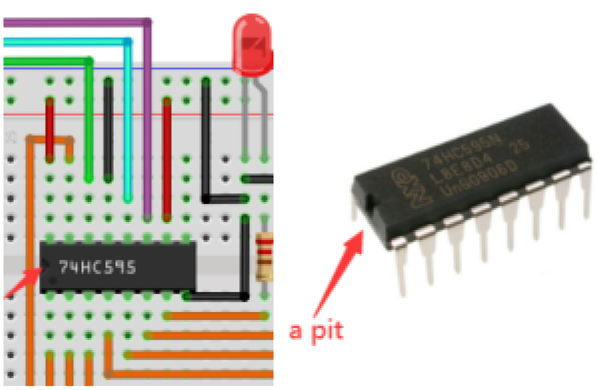


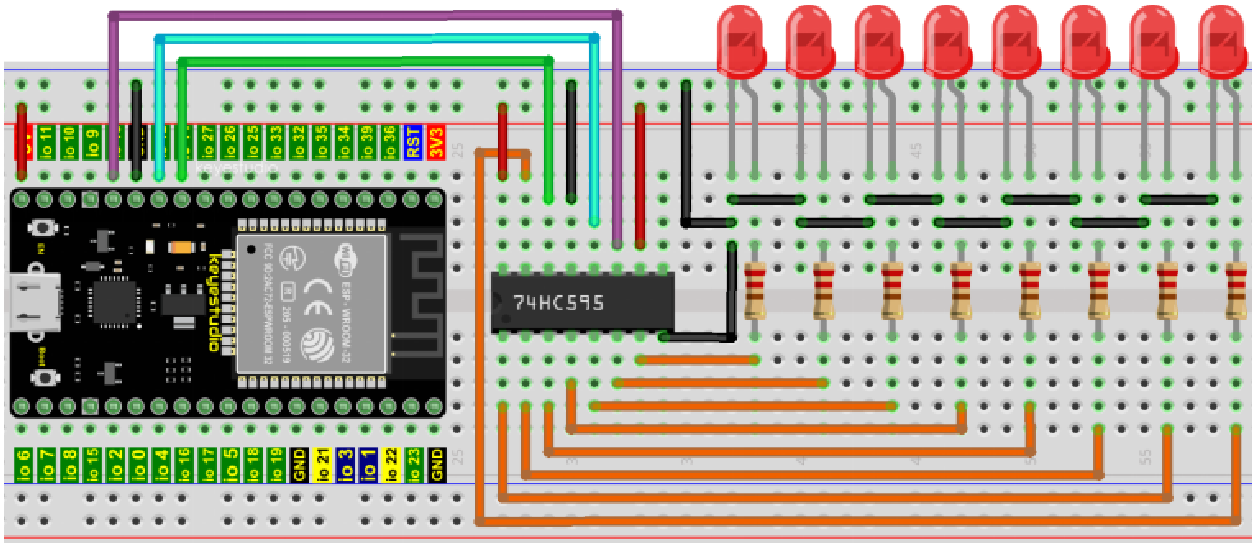
The ports of the 74HC595 chip are described as follows

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 10—SCLR	Remove shift register: When this pin is in low level, the content in shift register will be cleared.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.

7.12.4 4. Wiring diagram

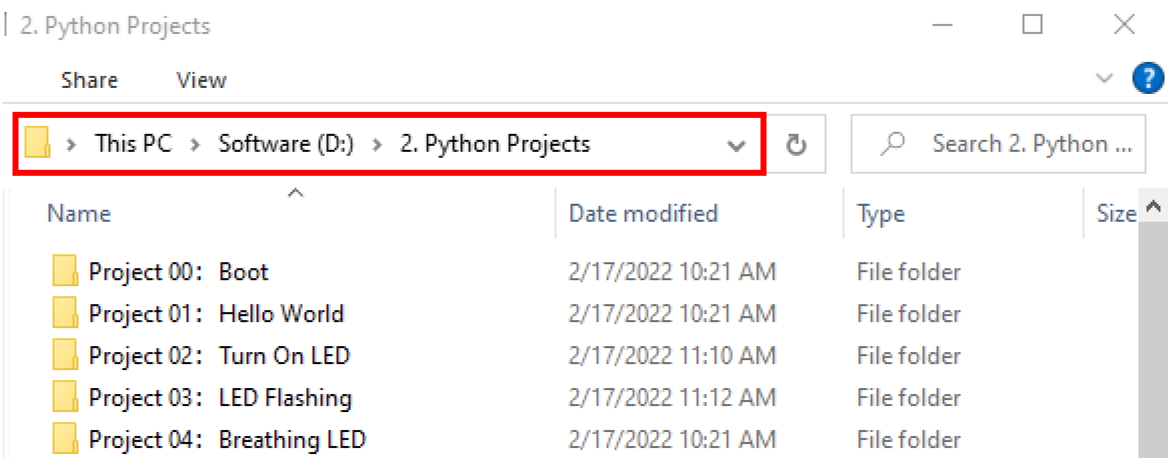
Note: Note the orientation in which the 74HC595N chip is inserted.





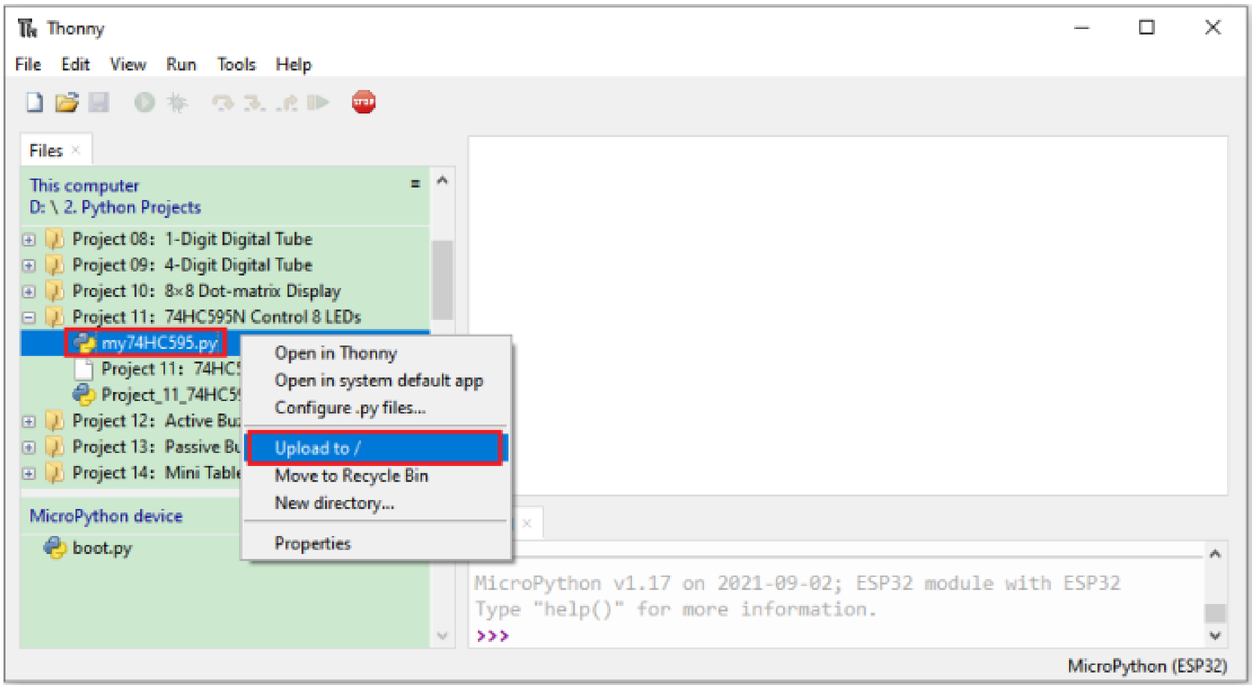
7.12.5 5. Project code

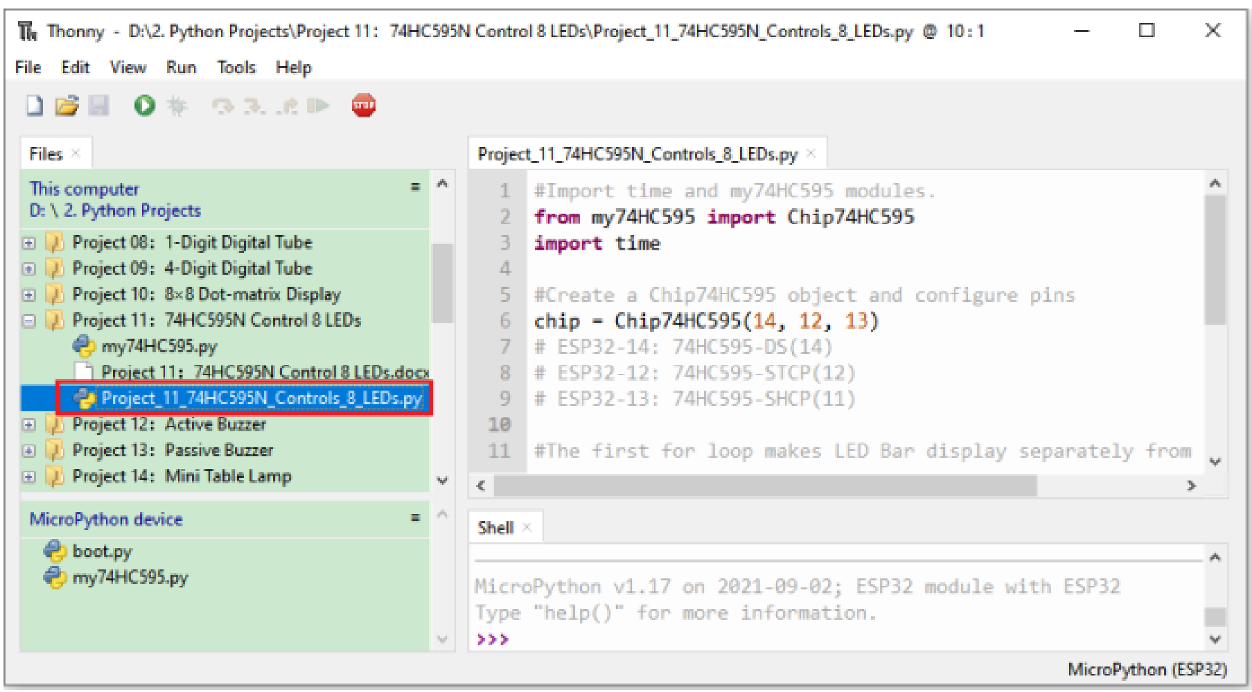
Codes used in this tutorial are saved in "**2. Python Projects**". If you haven't downloaded the code file, please click on the link to download it:Download Python Codes



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".






```python
#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)
```

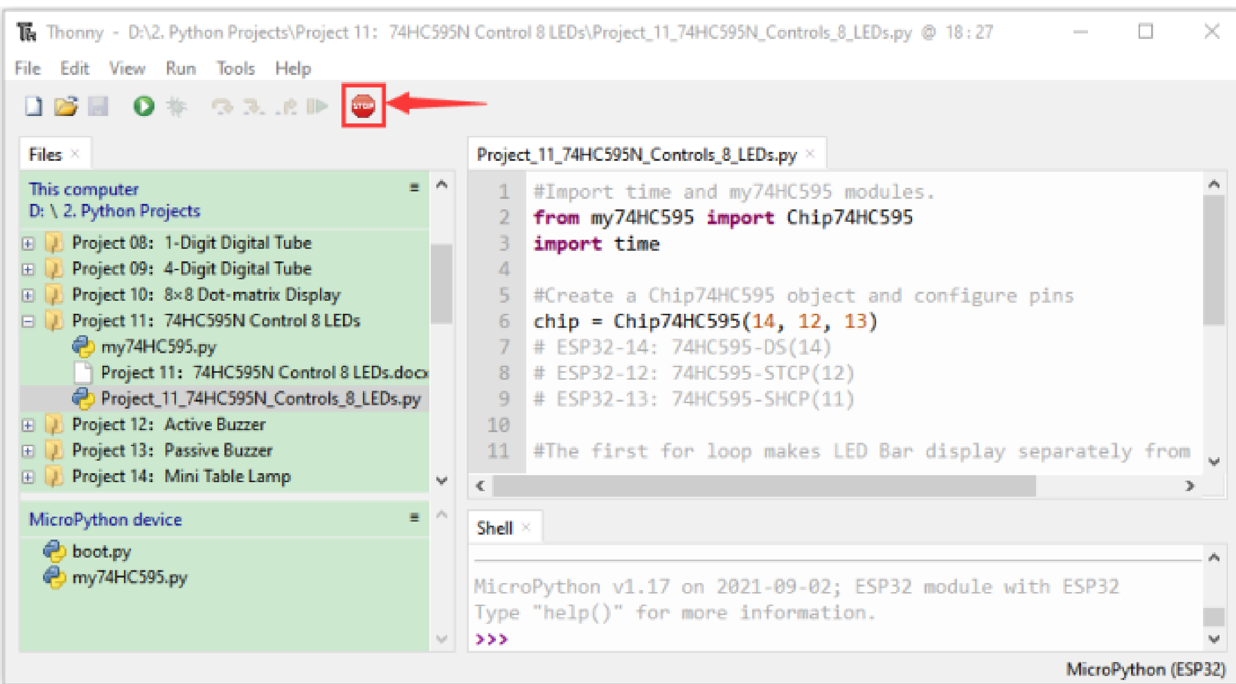
(continues on next page)

(continued from previous page)

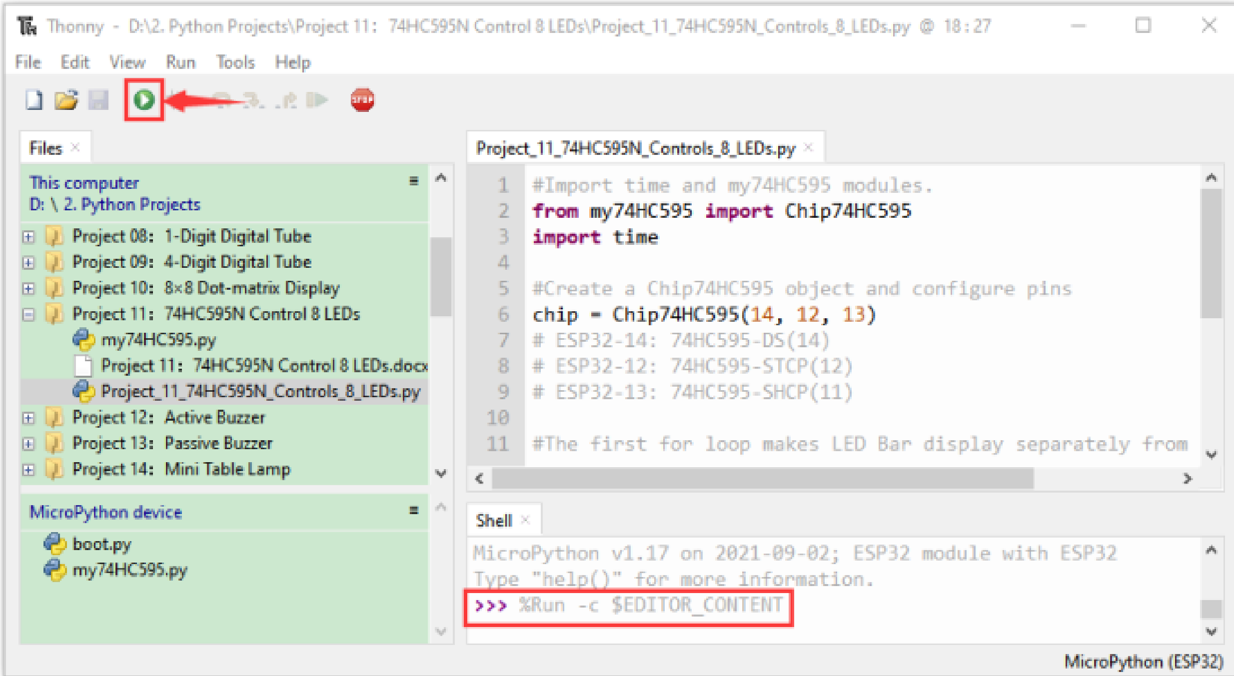
```
#The first for loop makes LED Bar display separately from left to right
#while the second for loop make it display separately from right to left.
while True:
    x = 0x01
    for count in range(8):
        chip.shiftOut(1, x)
        x = x<<1;
        time.sleep_ms(300)
    x = 0x01
    for count in range(8):
        chip.shiftOut(0, x)
        x = x<<1
        time.sleep_ms(300)
```

7.12.6 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 8 LEDs start flashing in flowing water mode. Press“Ctrl+C”or click “Stop/Restart backend”to exit the program.

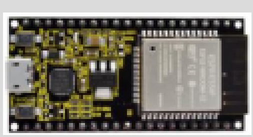
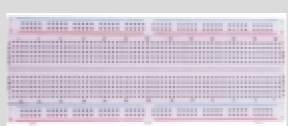
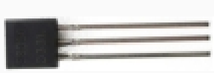
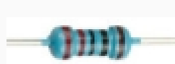
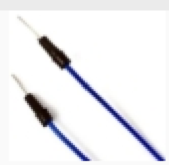
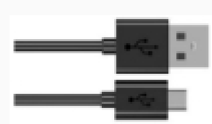


7.13 Project 12Active Buzzer

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

			
ESP32*1	Breadboard*1	Active buzzer*1	
			
NPN transistor(S8050)*1	1kResistor*1	Jumper Wires	USB Cable*1

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).

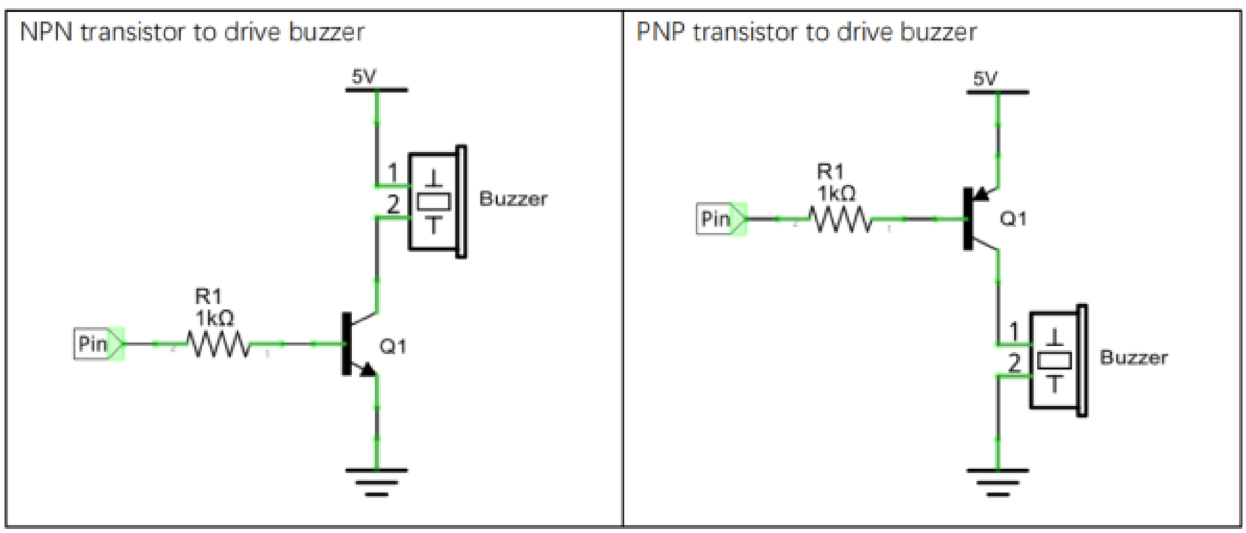
When there is current passing between "be", "ce" will allow several-fold current (transistor magnification) pass, at this point, transistor works in the amplifying area. When current between "be" exceeds a certain value, "ce" will not allow current to increase any longer, at this point, transistor works in the saturation area. Transistor has two types as shown below: PNP and NPN,

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

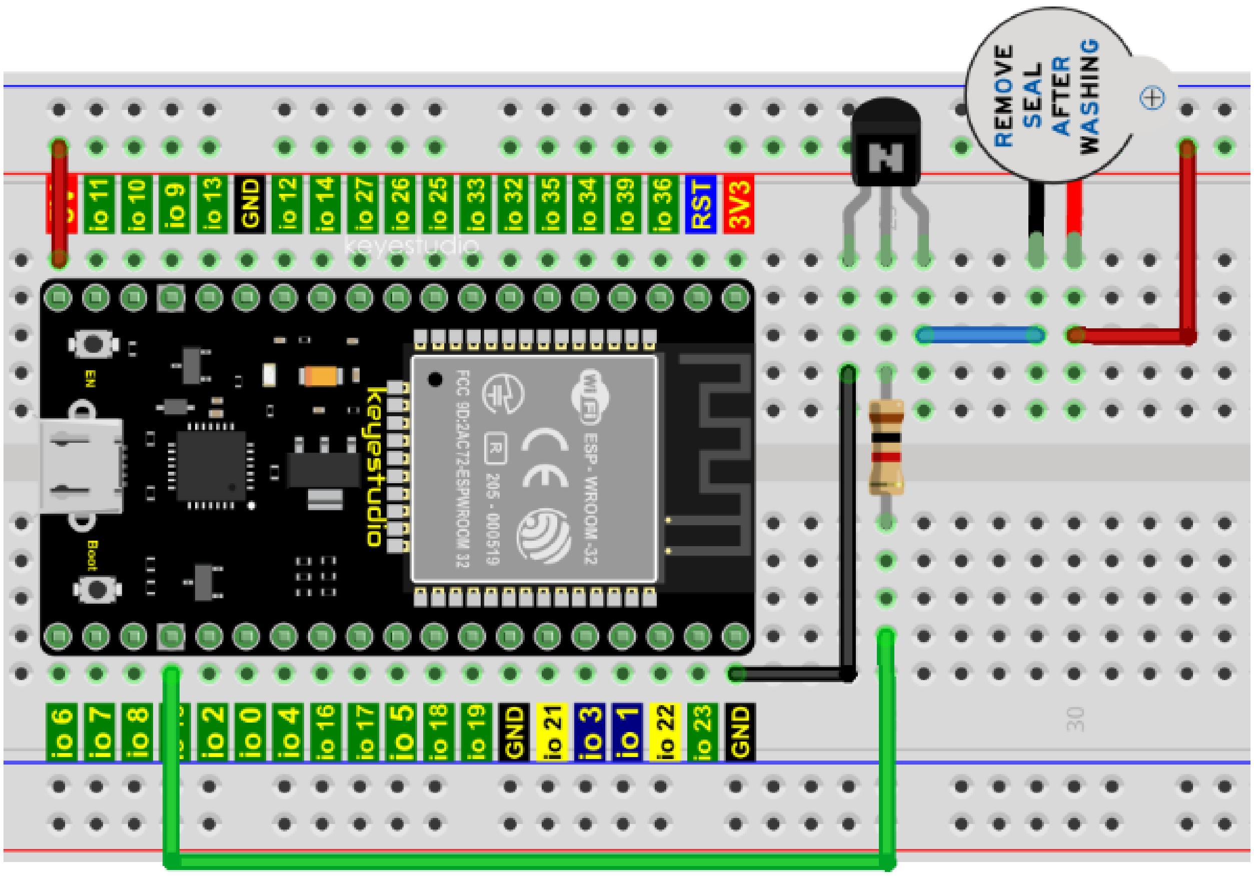
Based on the transistor's characteristics, it is often used as a switch in digital circuits. As micro-controller's capacity to output current is very weak, we will use transistor to amplify current and drive large-current components.

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.



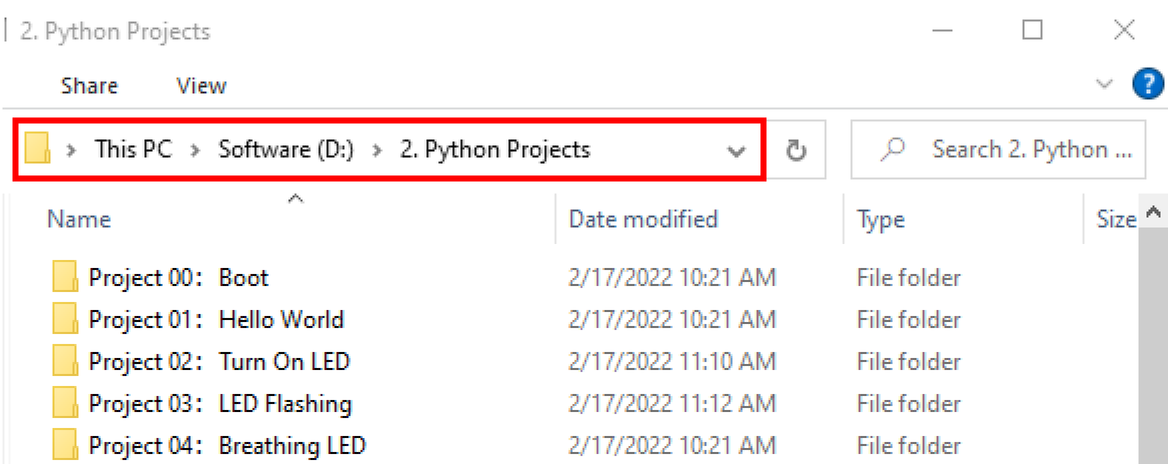
7.13.4 4.Wiring diagram



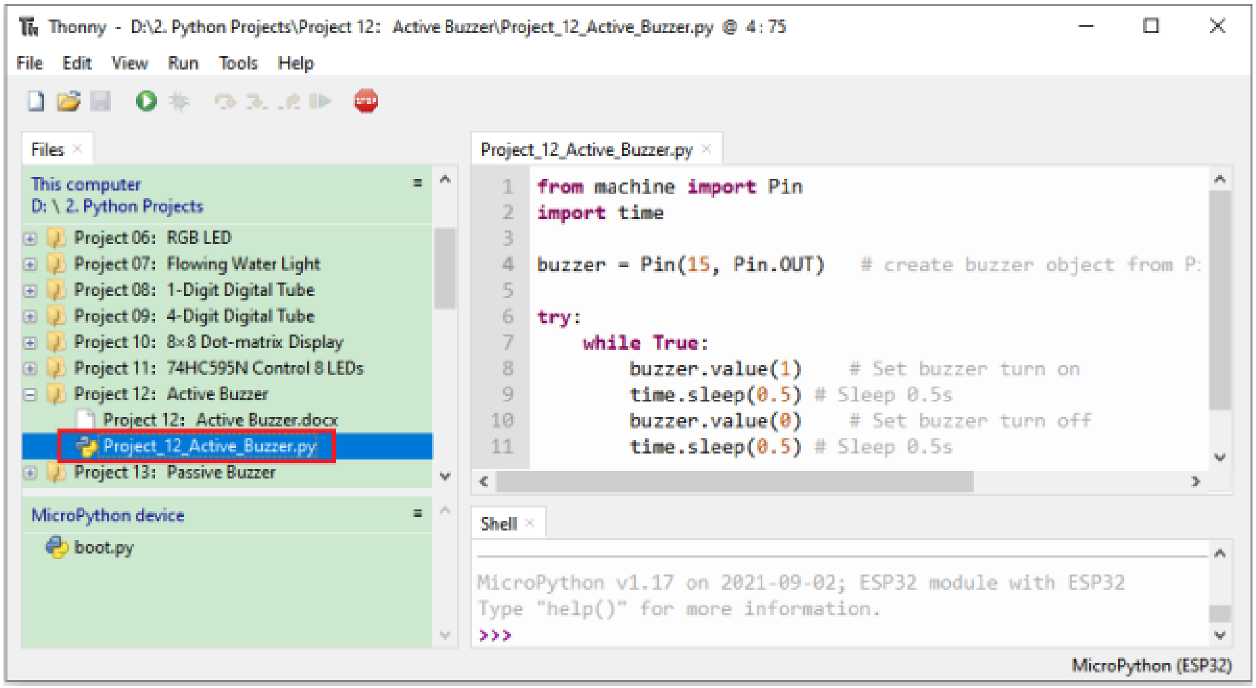
Note: The buzzer power supply in this circuit is 5V. On a 3.3V power supply, the buzzer can work, but will reduce the loudness.

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



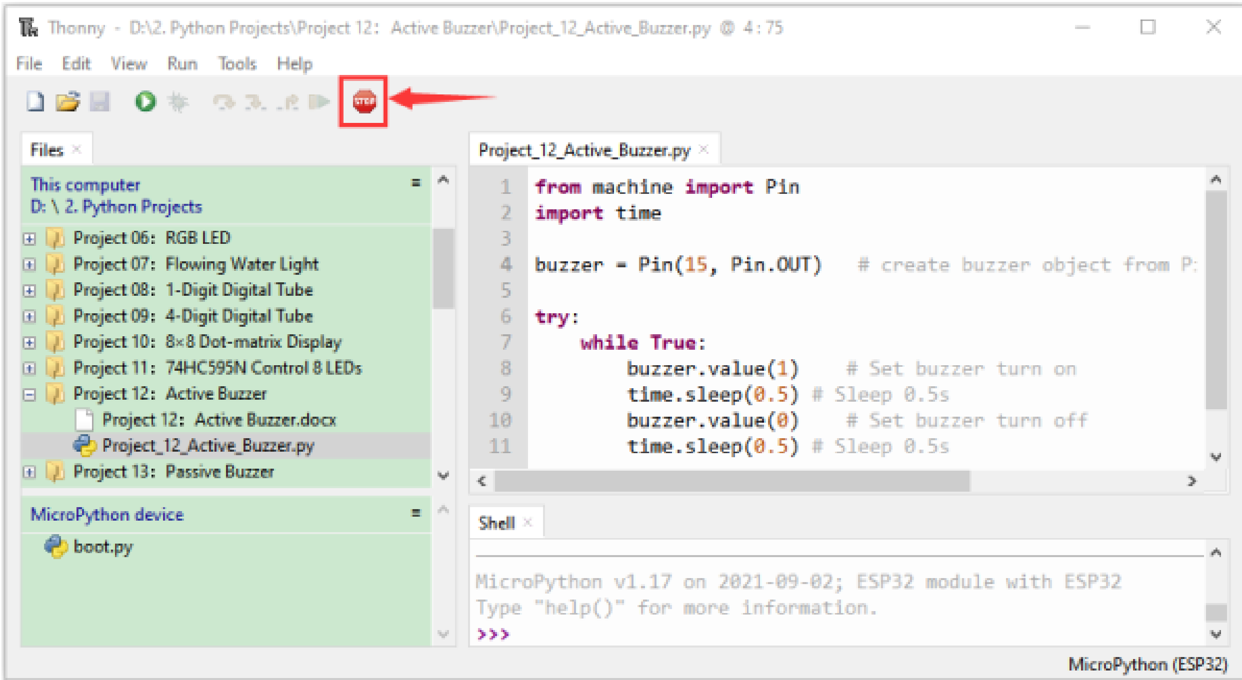
Open“Thonny”click“This computer”→“D:”→“2. Python Projects”→“Project 12Active Buzzer”, and then double left-click “Project_12_Active_Buzzer.py”.



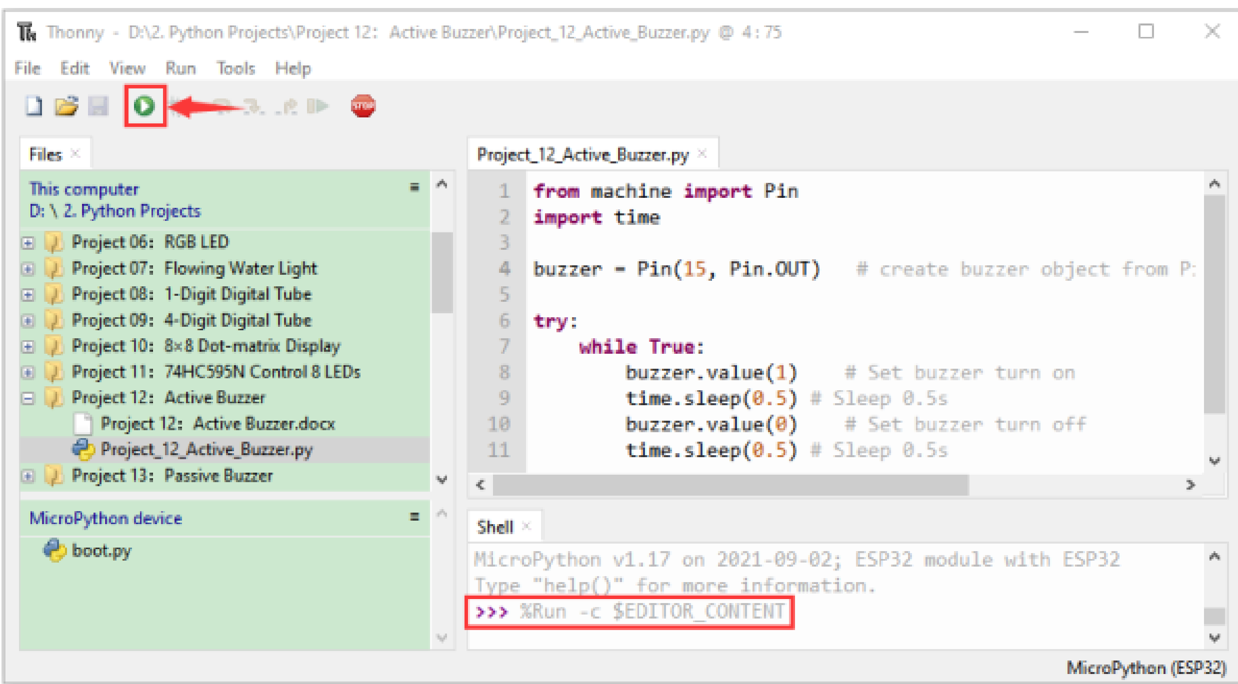

```python
from machine import Pin
import time

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 “Run current script”, the code starts to be executed and you’ll see that the active buzzer beeps. Press “Ctrl+C” or click “Stop/Restart backend” to exit the program.

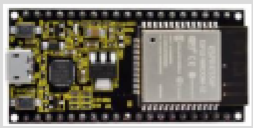
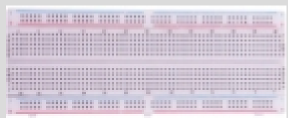
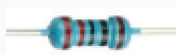
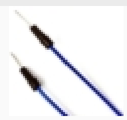
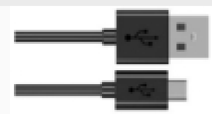


7.14 Project 13Passive Buzzer

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

			
ESP32*1	Breadboard*1	Passive Buzzer *1	
			
NPN transistor(S8050)*1	1kResistor*1	Jumper Wires	USB Cable*1

7.14.3 3.Component knowledge

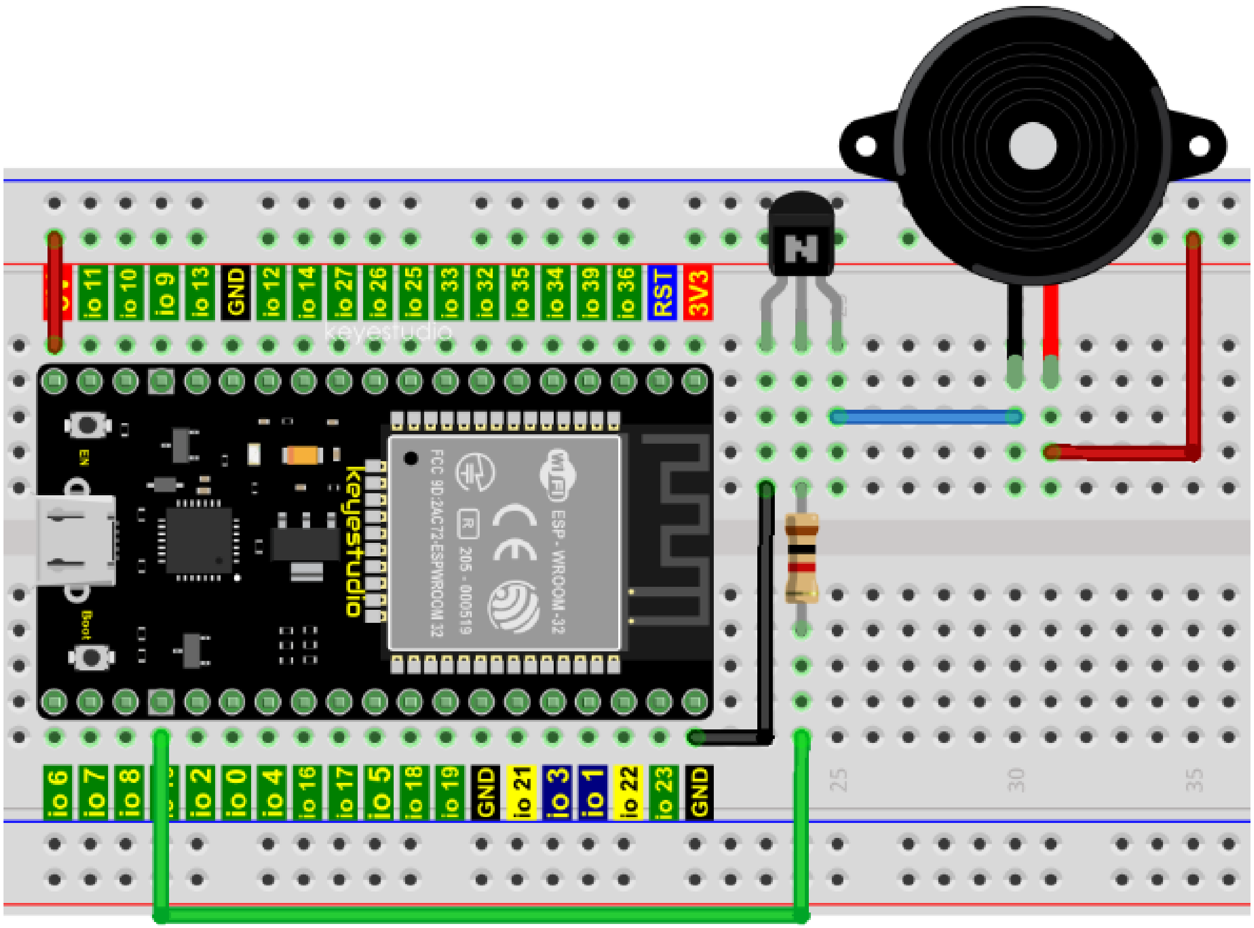
Passive buzzer:

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.

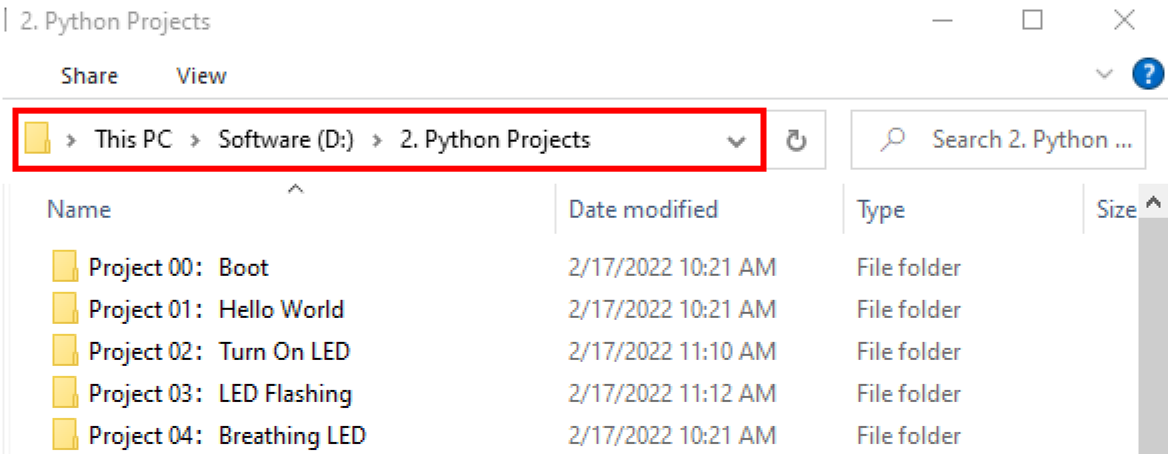
Transistor: Please refer to Project 12.

7.14.4 4.Wiring diagram:

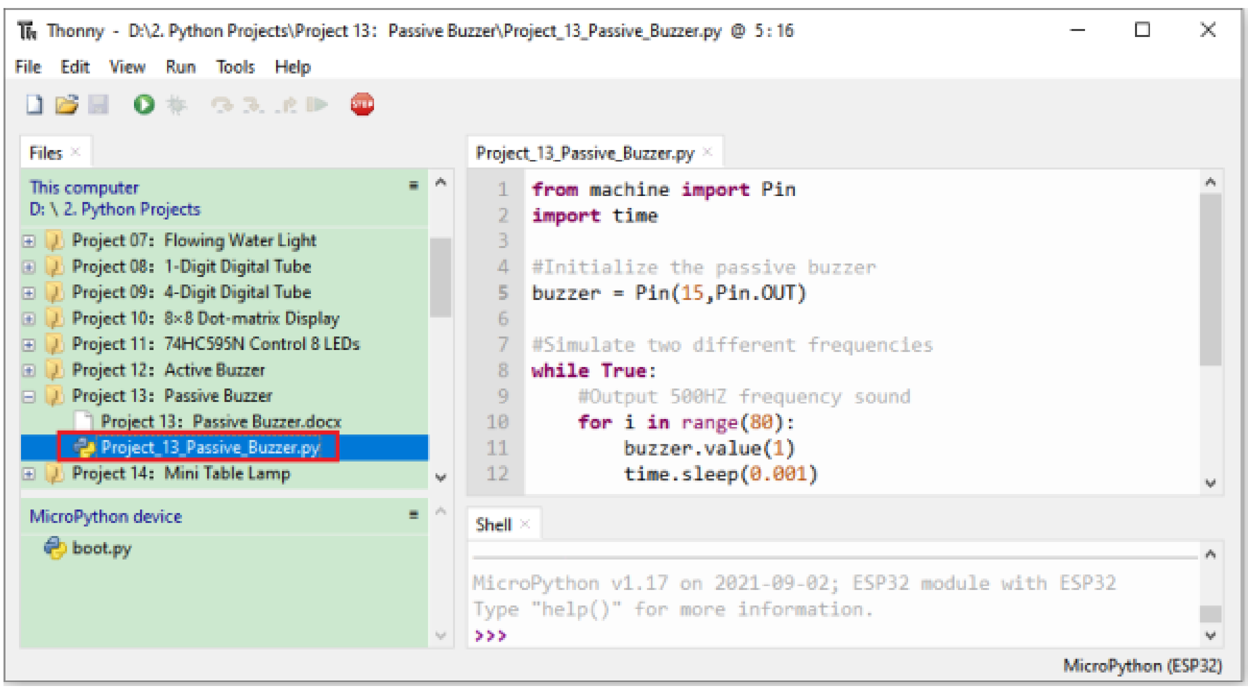


7.14.5 5.Project code

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



Open "Thonny", click "This computer"→"D:"→"2. Python Projects"→"Project 13Passive Buzzer", and then double left-click "Project_13_Passive_Buzzer.py".




```python
from machine import Pin
import time

#Initialize the passive buzzer
buzzer = Pin(15,Pin.OUT)

#Simulate two different frequencies
while True:
    #Output 500HZ frequency sound
    for i in range(80):
        buzzer.value(1)
        time.sleep(0.001)
        buzzer.value(0)
```

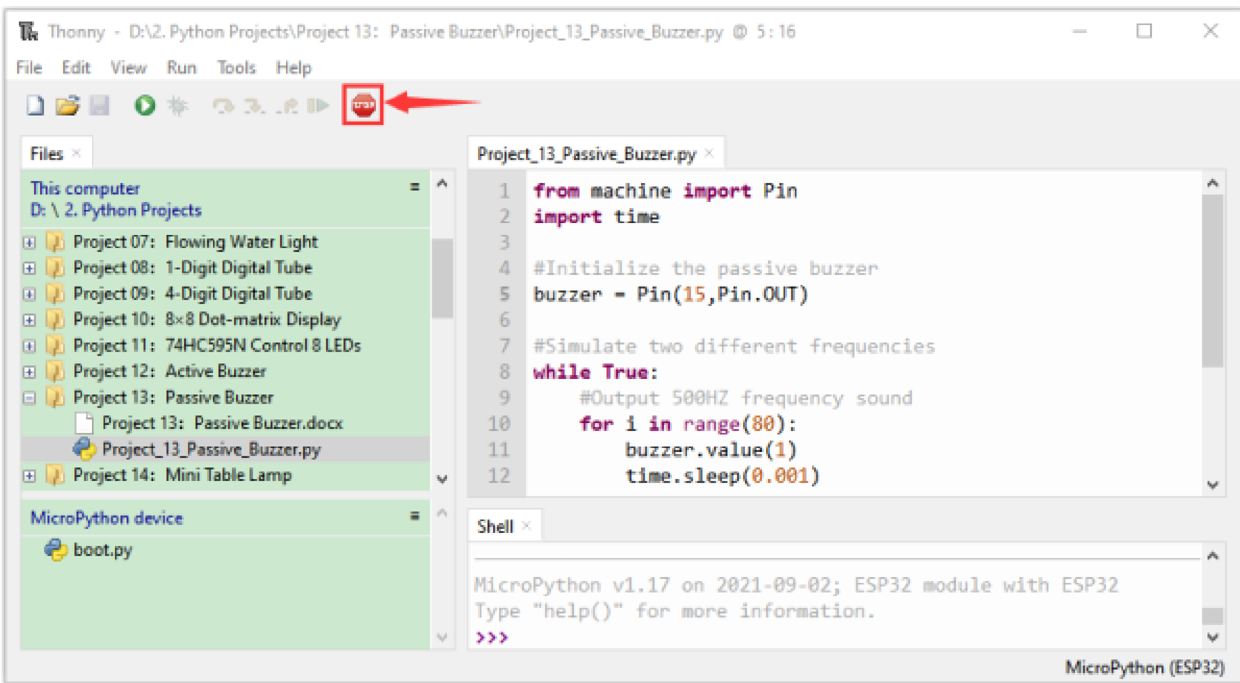
(continues on next page)

(continued from previous page)

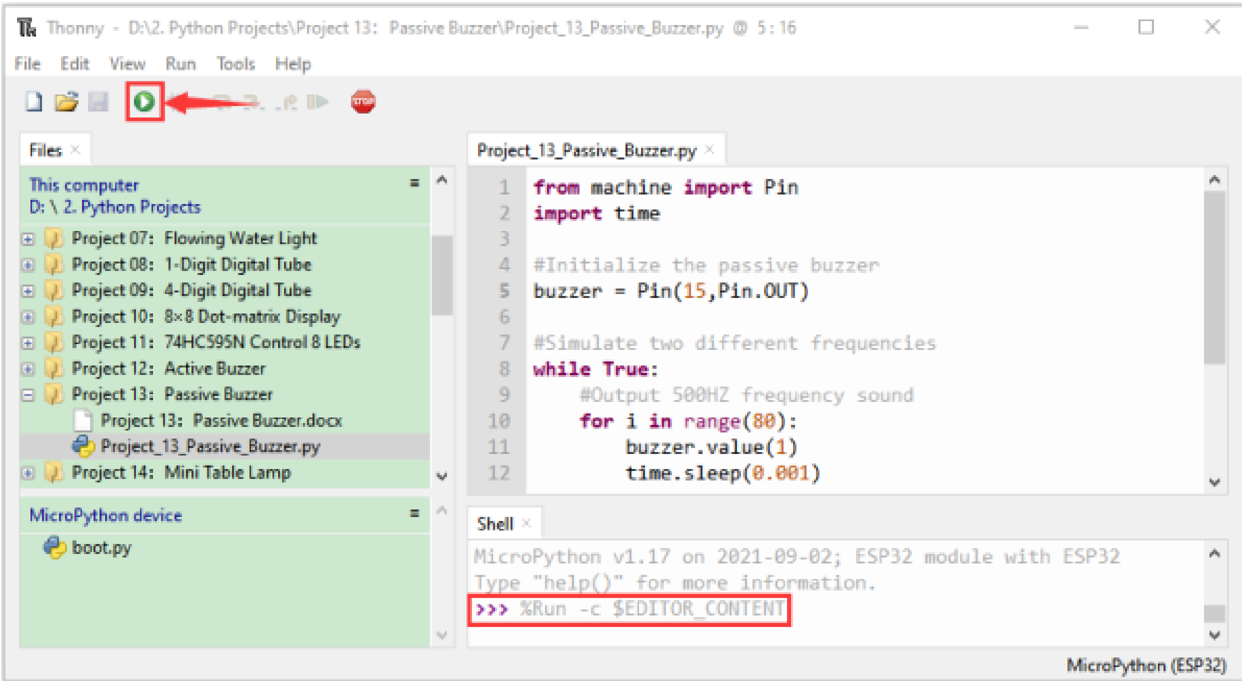
```
        time.sleep(0.001)
    #Output 250HZ frequency sound
    for i in range(100):
        buzzer.value(1)
        time.sleep(0.002)
        buzzer.value(0)
        time.sleep(0.002)
```

7.14.6 6.Project result

Make sure the ESP32 has been connected to the computer, click “Stop/Restart backend”.



Click “Run current script”, the code starts to be executed and you’ll see that the passive buzzer sounds alarm. Press “Ctrl+C” or click “Stop/Restart backend”to exit the program.

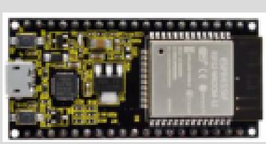
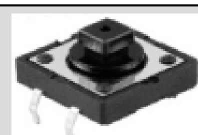
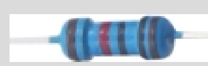
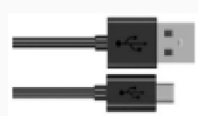


7.15 Project 14: Mini Table Lamp

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

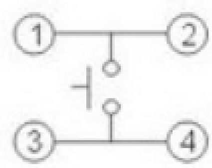
				
ESP32*1	Breadboard*1	Button*1	10K Resistor*1	
				
Red LED*1	220 Resistor*1	USB Cable*1	Jumper Wires	Button Cap*1

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.

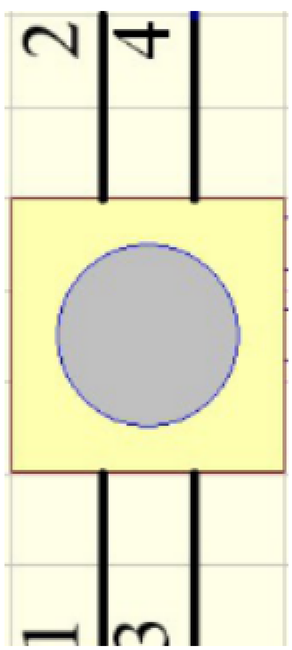
The internal structure of the button is shown in the figure 

.

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.

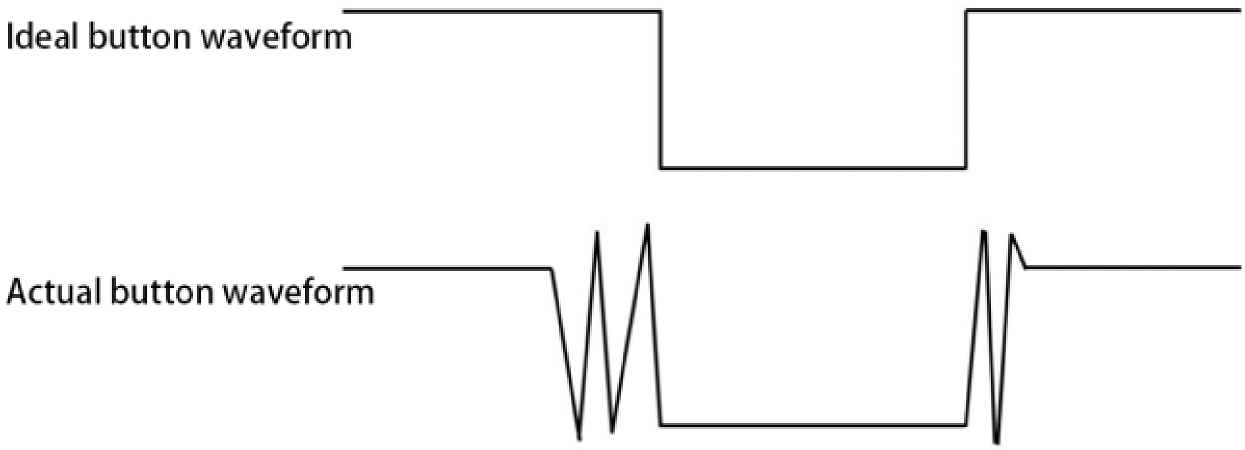
Schematic diagram of the button:



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.

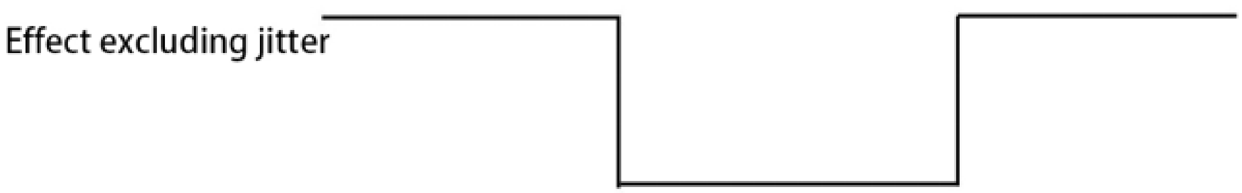


How to eliminate the shake?

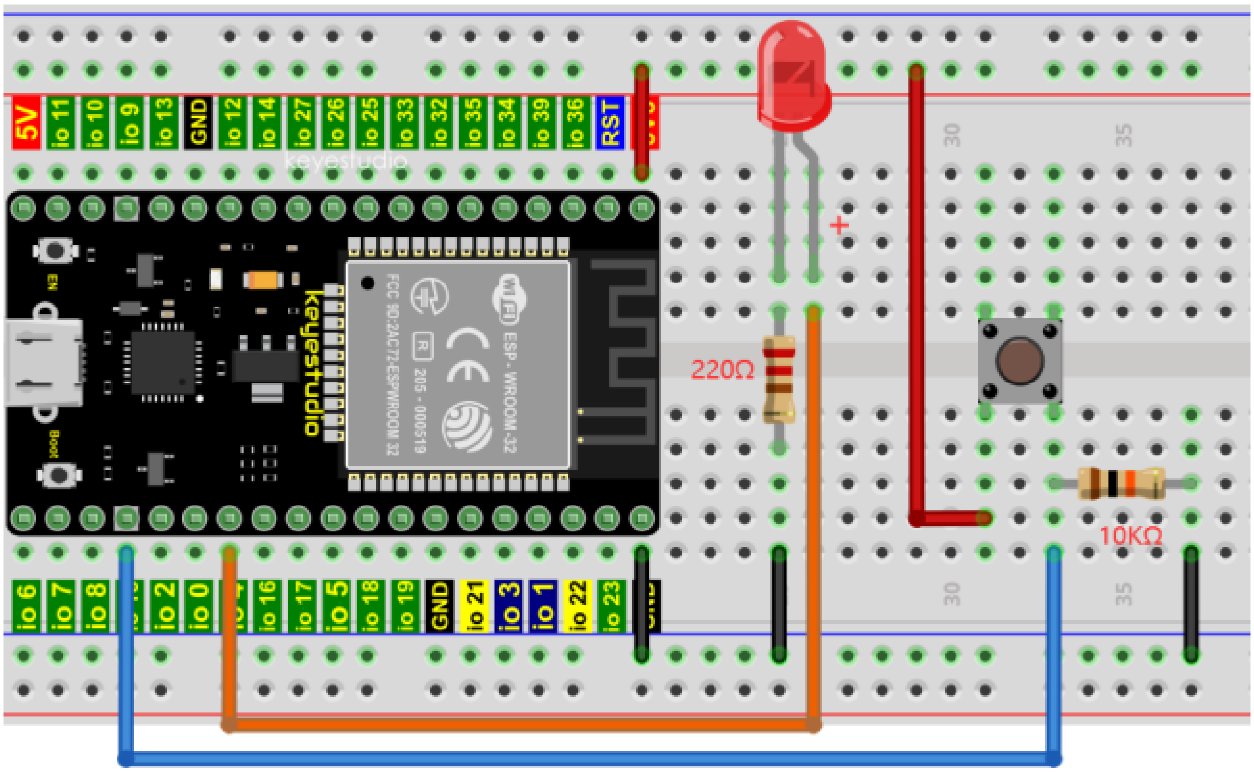
There are two common methods, namely fix shake in the software and hardware. We only discuss the shake removal in the software.

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.

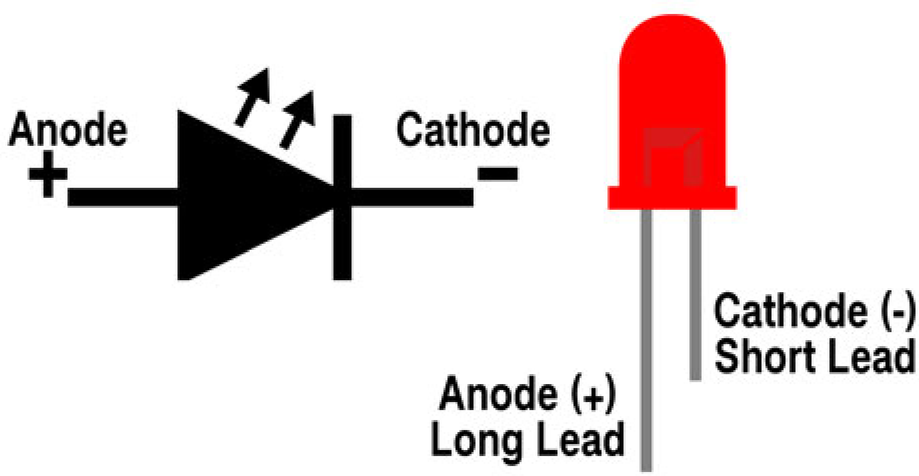


7.15.4 4.Wiring Diagram

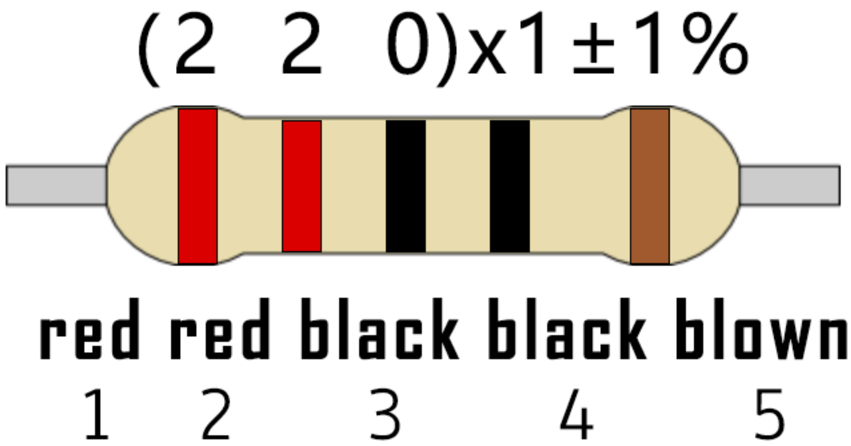


Note:

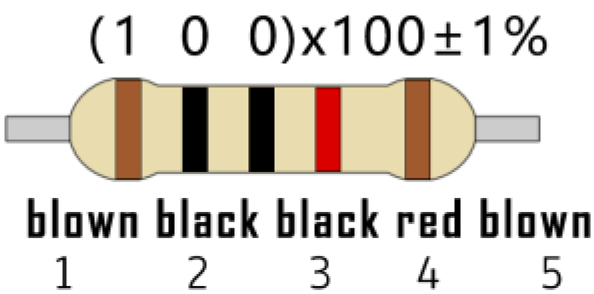
Connect the LED



220 5-band resistor

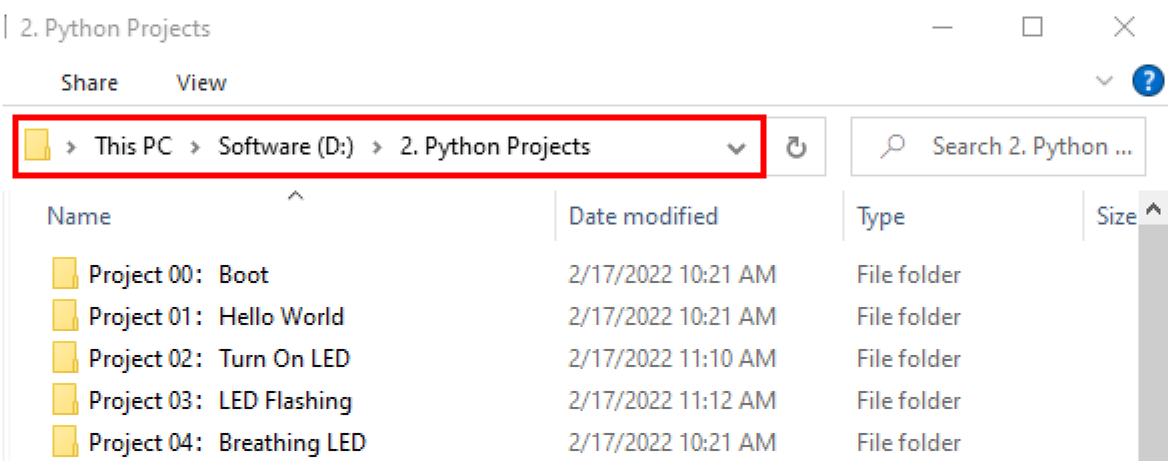


10K 5-band resistor

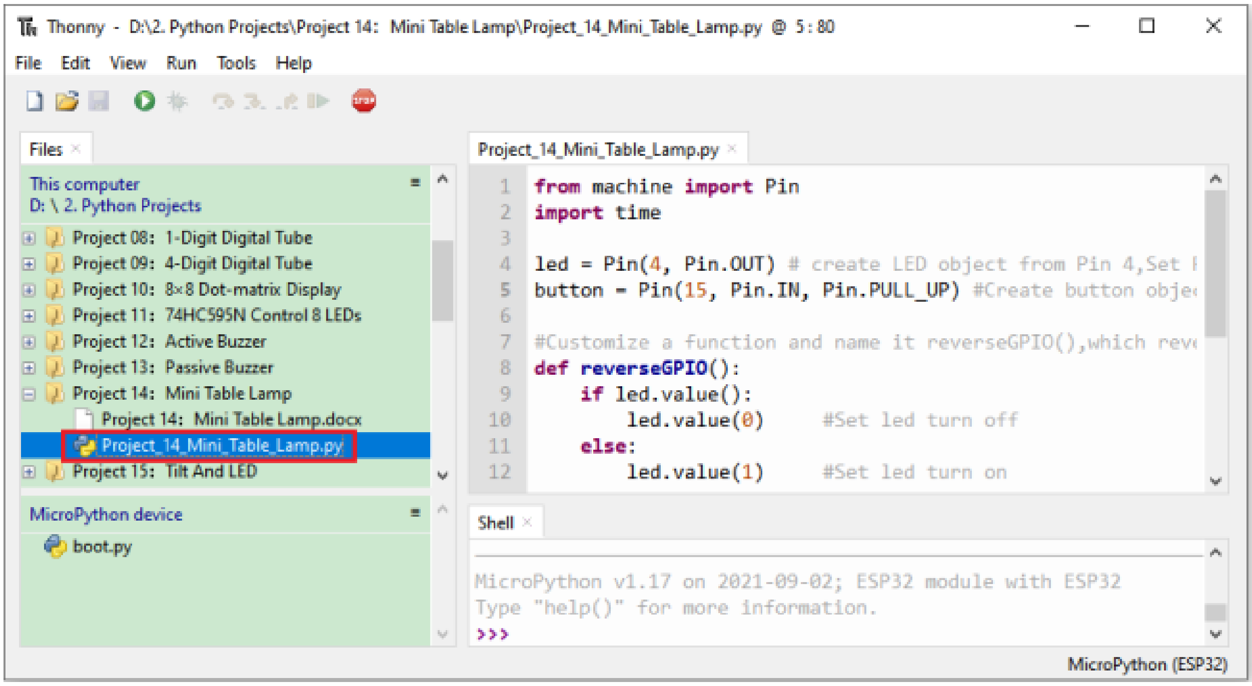


7.15.5 5.Project code

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



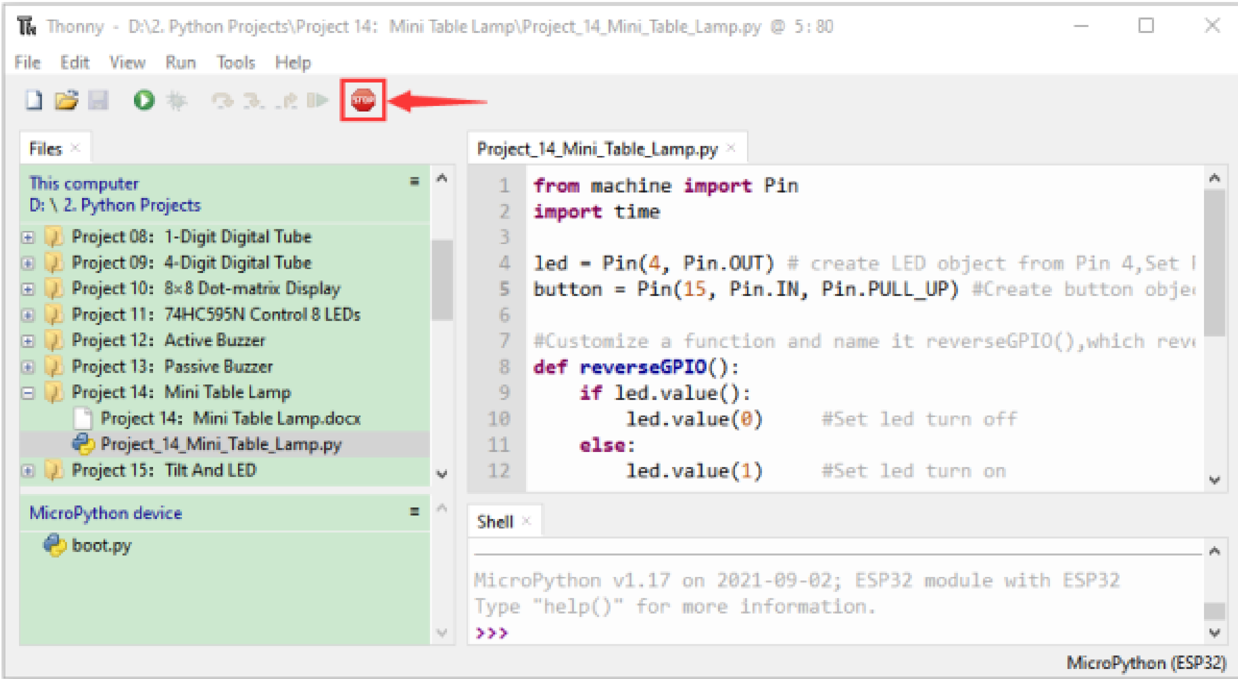
Open“Thonny”click“This computer”→“D:”→“2. Python Projects”→“Project 14Mini Table Lamp”, and then double left-click “Project_14_Mini_Table_Lamp.py”.




```python
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 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
```

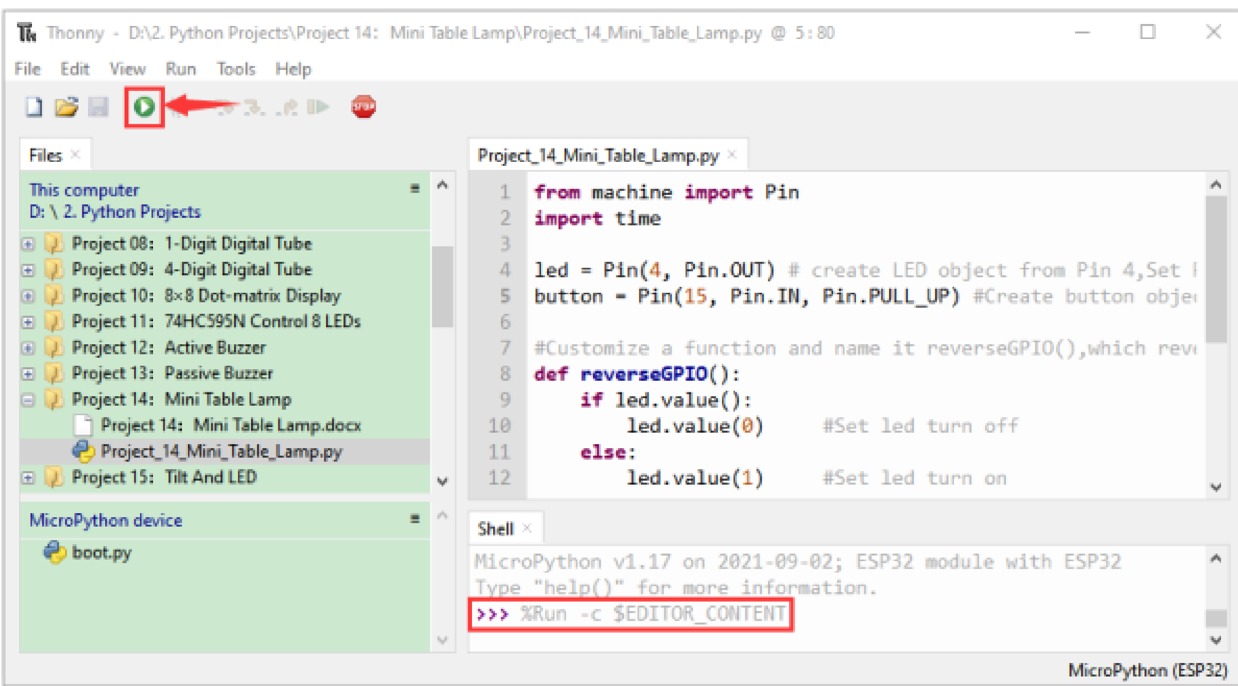
7.15.6 6.Project result

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.



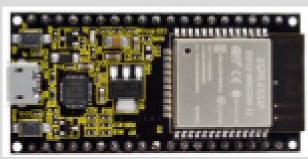
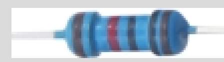
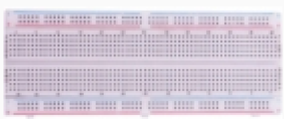
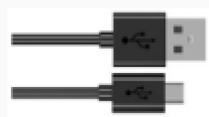
7.16 Project 15Tilt and LED

7.16.1 1.Introduction

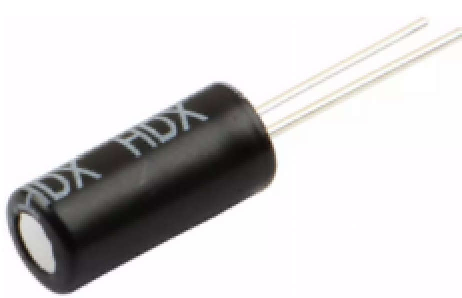
The ancients without electronic clock, so the hourglass are invented to measure time. The hourglass has a large capacity on both sides, and which is filled with fine sand on one side. What's more, there is a small channel in the middle, which can make the hourglass stand upright , the side with fine 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

			
ESP32*1	Tilt Switch*1	Red LED*4	10K Resistor*1
			
Breadboard*1	220 Resistor*4	USB Cable*1	Jumper Wires

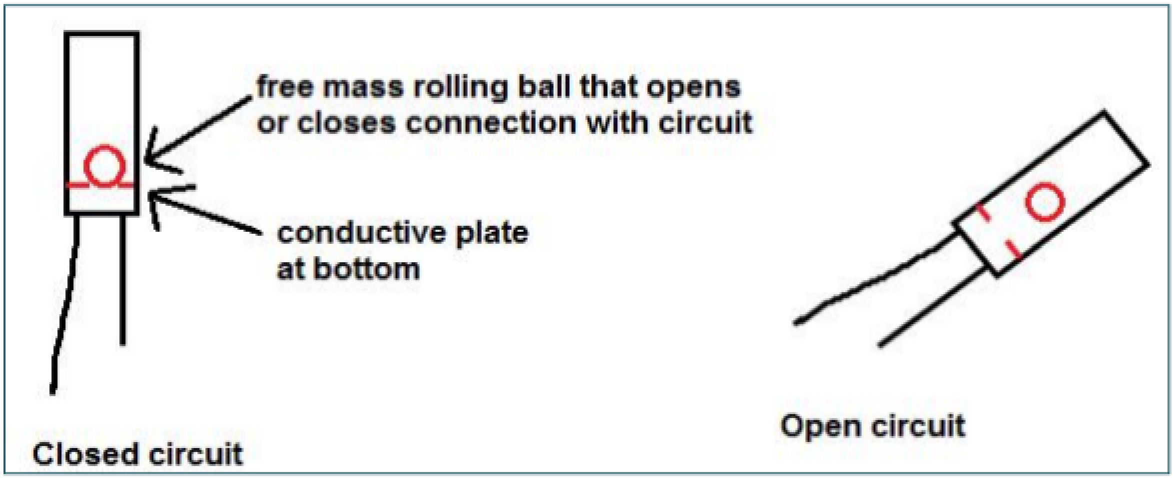
7.16.3 3.Component knowledge



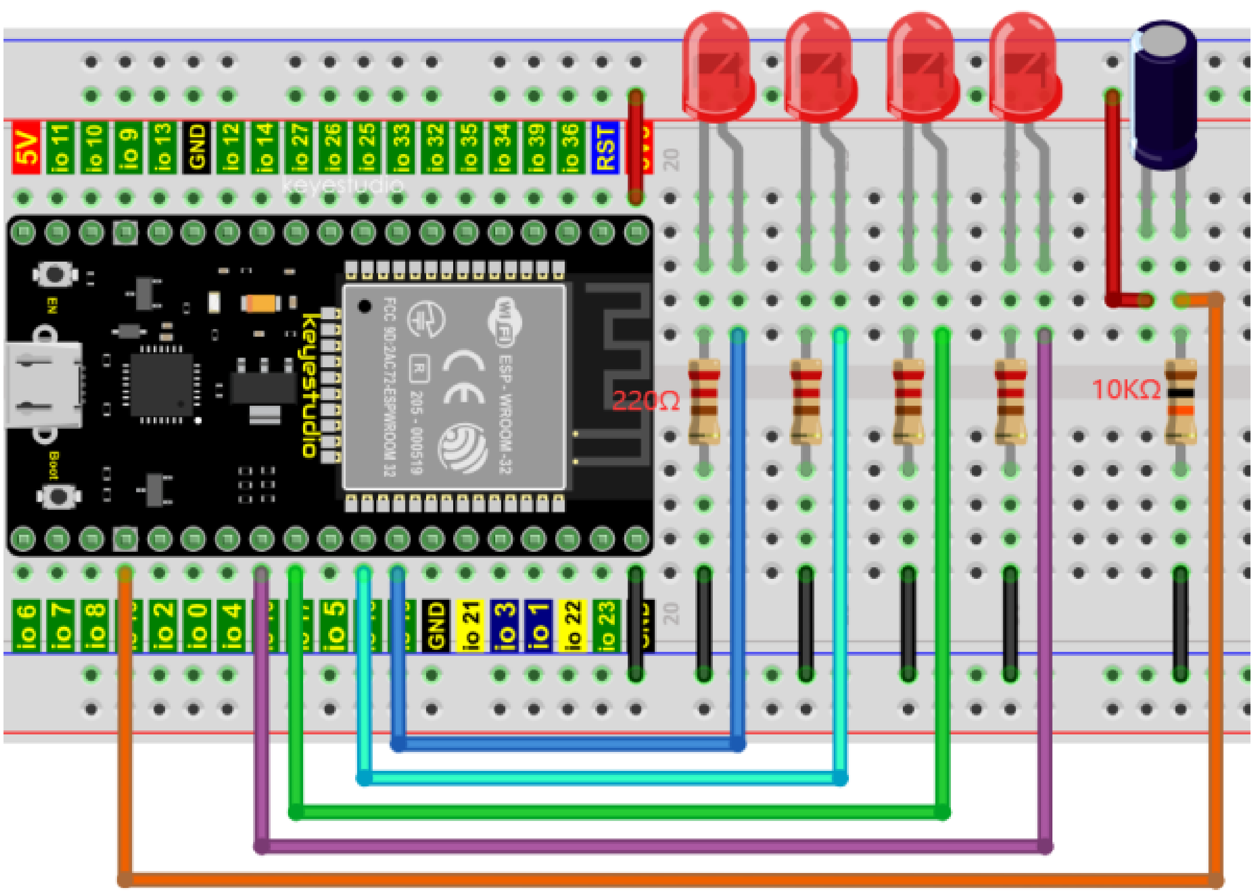
Tilt switch is also called digital switch. Inside is a metal ball that can roll. The principle of rolling the metal ball to contact with the conductive plate at the bottom, which is used to control the on and off of the circuit. When it is a rolling ball tilt sensing switch with single directional trigger, 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:

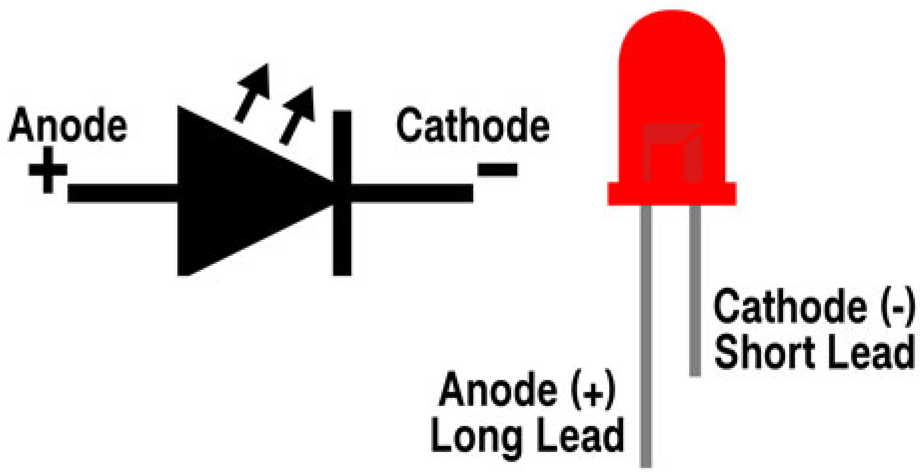


7.16.4 4. Wiring Diagram

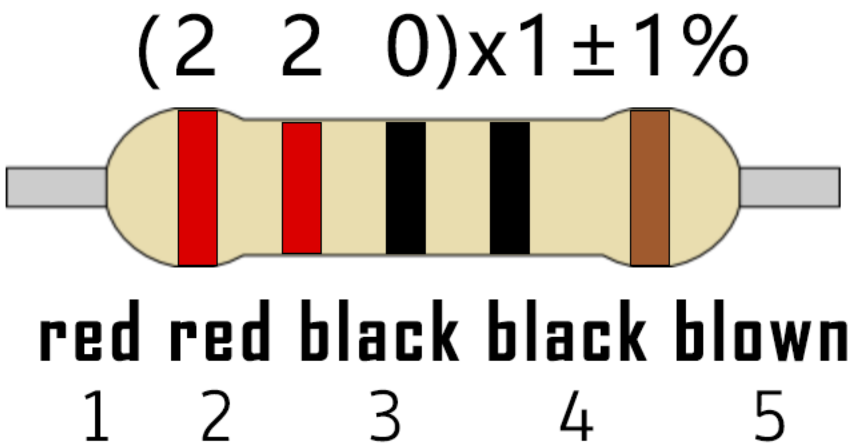


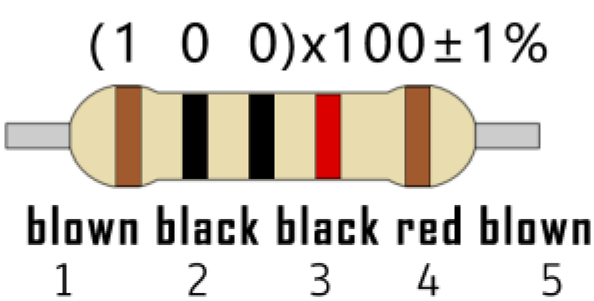
Note:

How to connect the LED



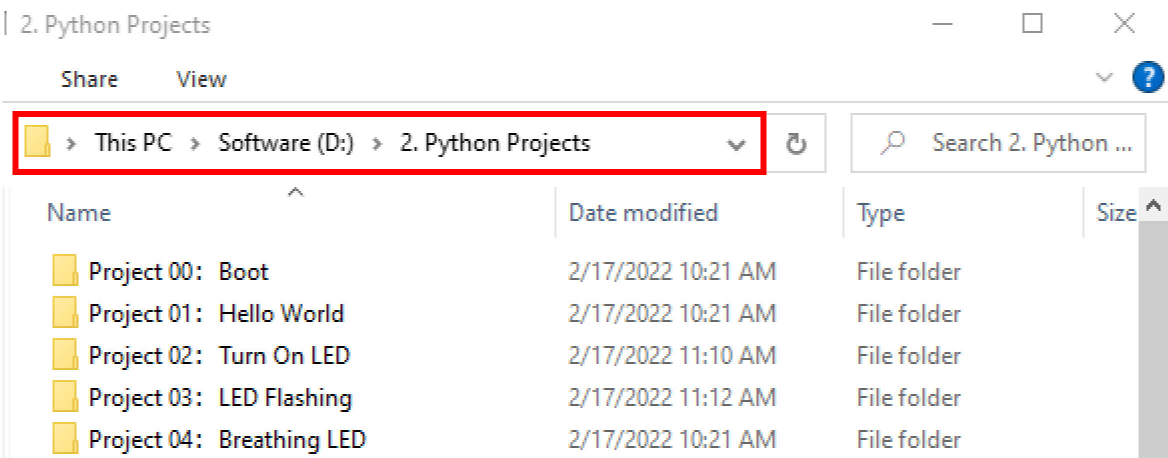
How to identify the 220 5-band resistor and 10K 5-band resistor



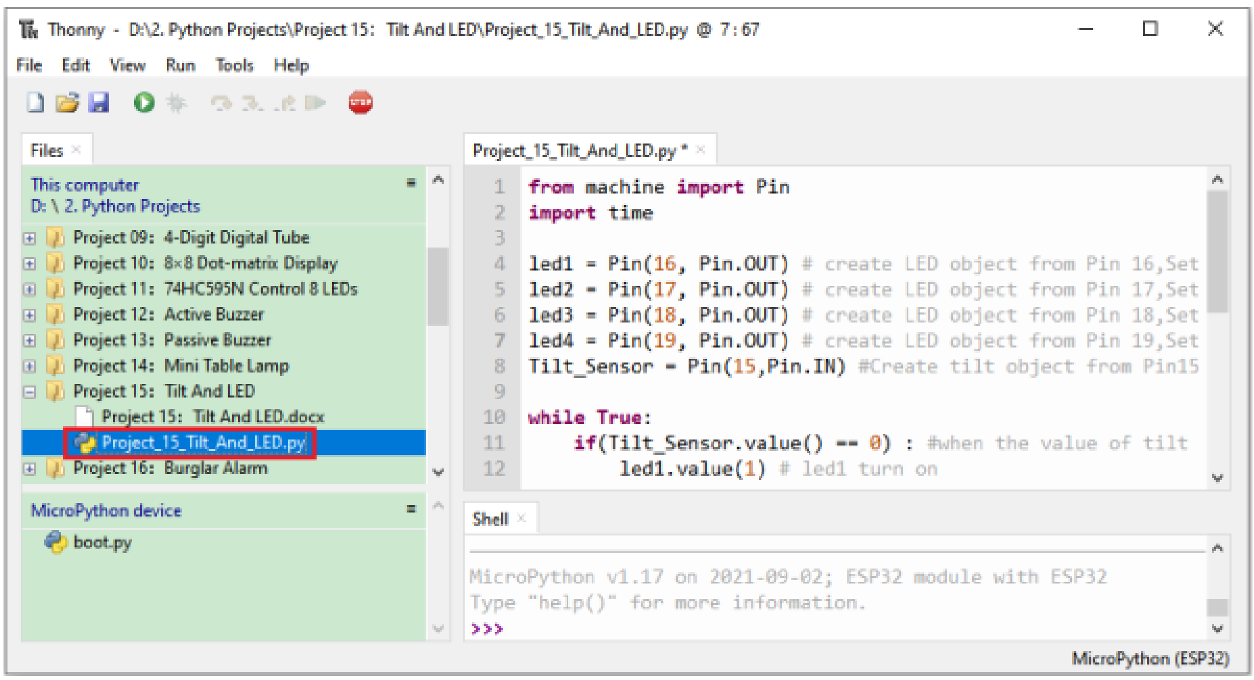


7.16.5 5.Project code

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



Open“Thonny”click“This computer”→“D:”→“2. Python Projects”→“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
```

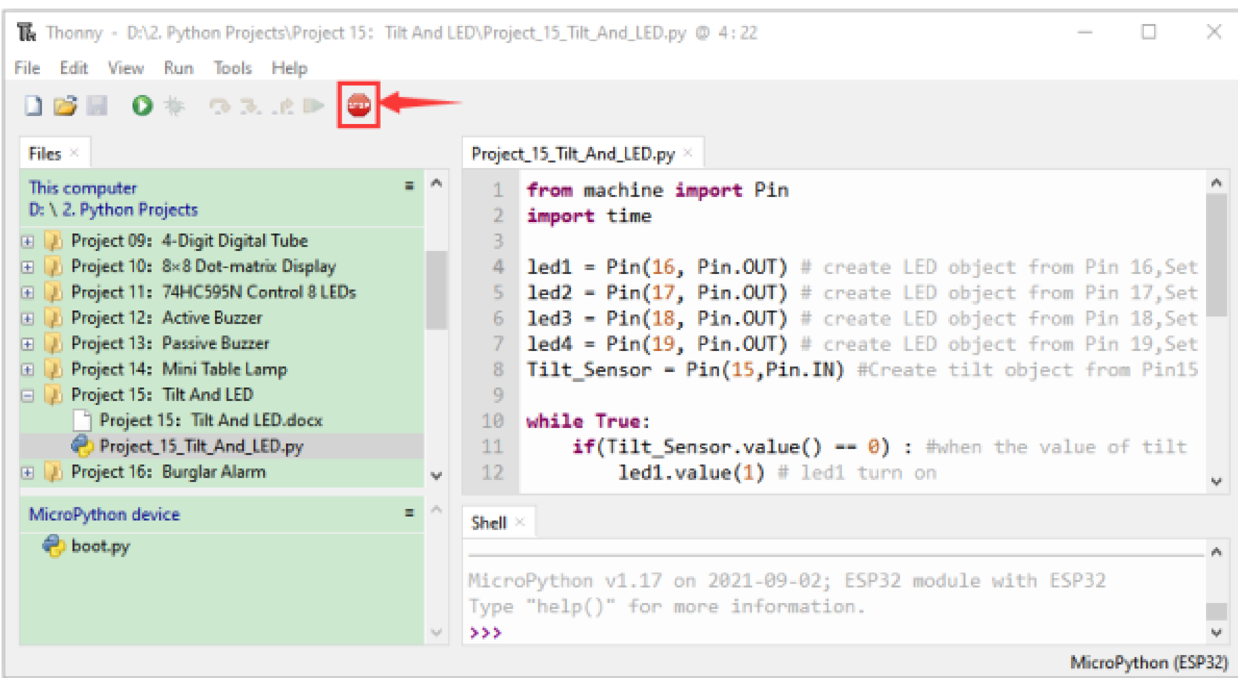
(continues on next page)

(continued from previous page)

```
while True:
    if(Tilt_Sensor.value() == 0) : #when the value of tilt sensor is 0
        led1.value(1) # led1 turn on
        time.sleep_ms(200)#delay
        led2.value(1) # led2 turn on
        time.sleep_ms(200)#delay
        led3.value(1) # led3 turn on
        time.sleep_ms(200)#delay
        led4.value(1) # led4 turn on
        time.sleep_ms(200)#delay
    else :           #when the value of tilt sensor is 1
        led4.value(0) # led4 turn off
        time.sleep_ms(200)#delay
        led3.value(0) # led3 turn off
        time.sleep_ms(200)#delay
        led2.value(0) # led2 turn off
        time.sleep_ms(200)#delay
        led1.value(0) # led1 turn off
        time.sleep_ms(200)#delay
```

7.16.6 6.Project result

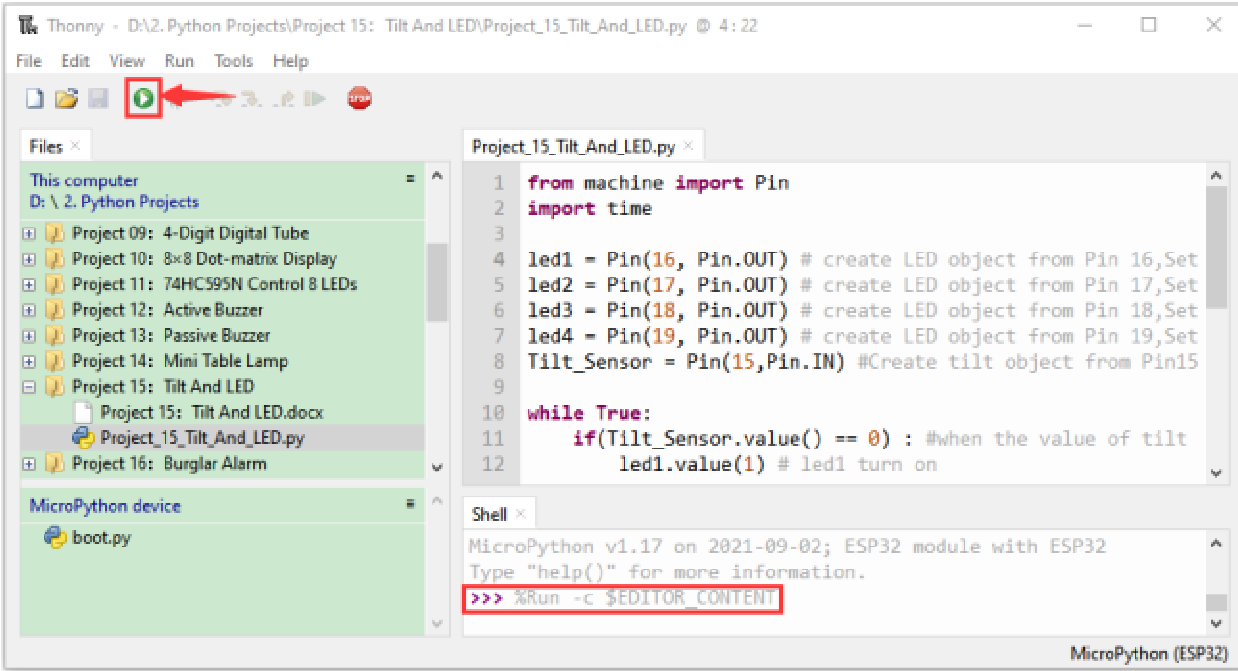
Make sure the ESP32 has been connected to the computer, click  “Stop/Restart backend” .



Click  “Run current script”, the code starts to be executed and you’ll see that 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 “Stop/Restart backend” to exit the program.

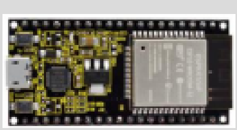
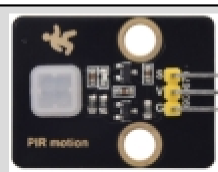
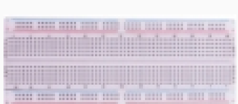
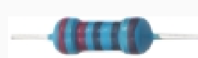
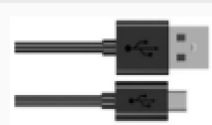
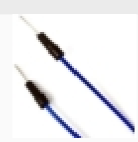


7.17 Project 16Burglar Alarm

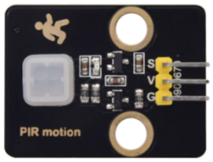
7.17.1 1.Introduction

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

				
ESP32*1	Human Body Infrared Sensor*1	Active Buzzer*1	Red LED*1	
				
Breadboard*1	M-F Dupont Wires	220Resistor*1	USB Cable*1	Jumper Wires

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.

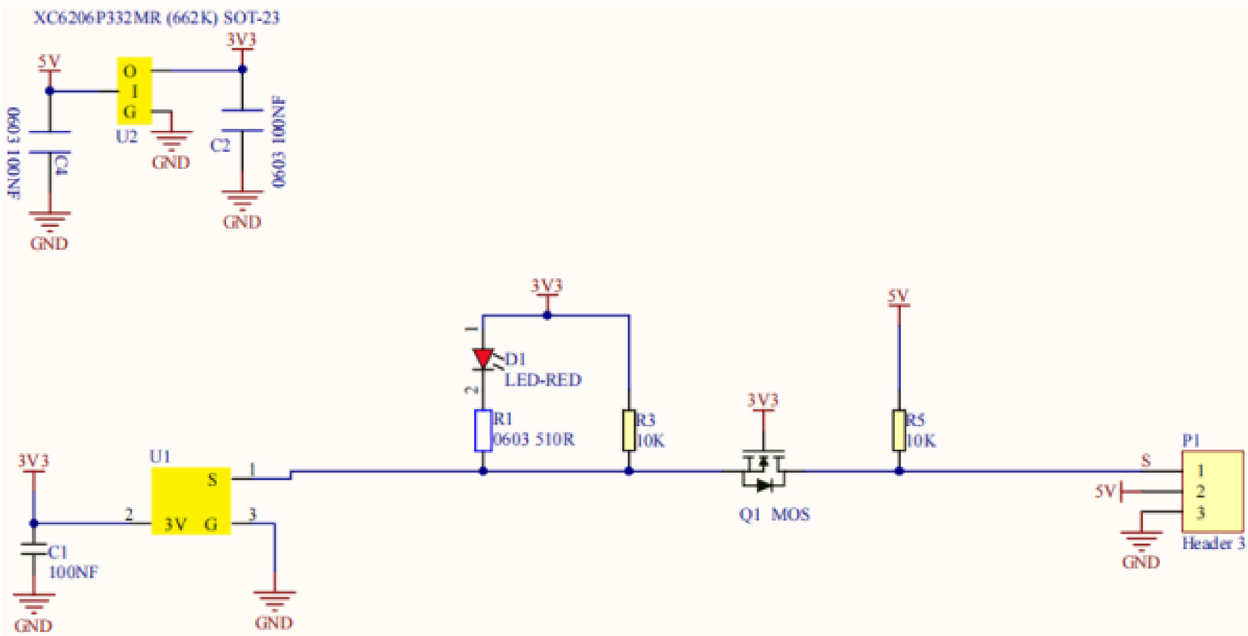
Special attention should be paid to the fact that this sensor can detect peopleanimals and cars in motion, which cannot be detected in static, and the maximum detection distance is about 7 meters.

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.

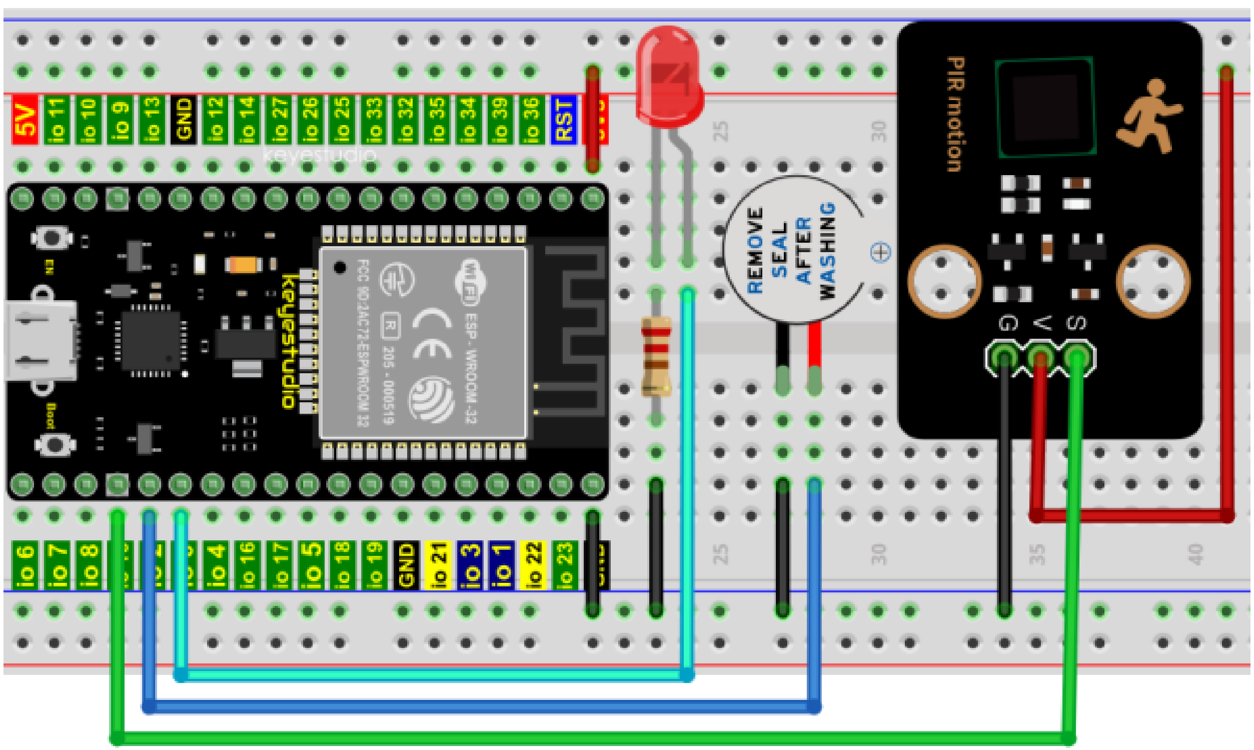
Features:

- 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:

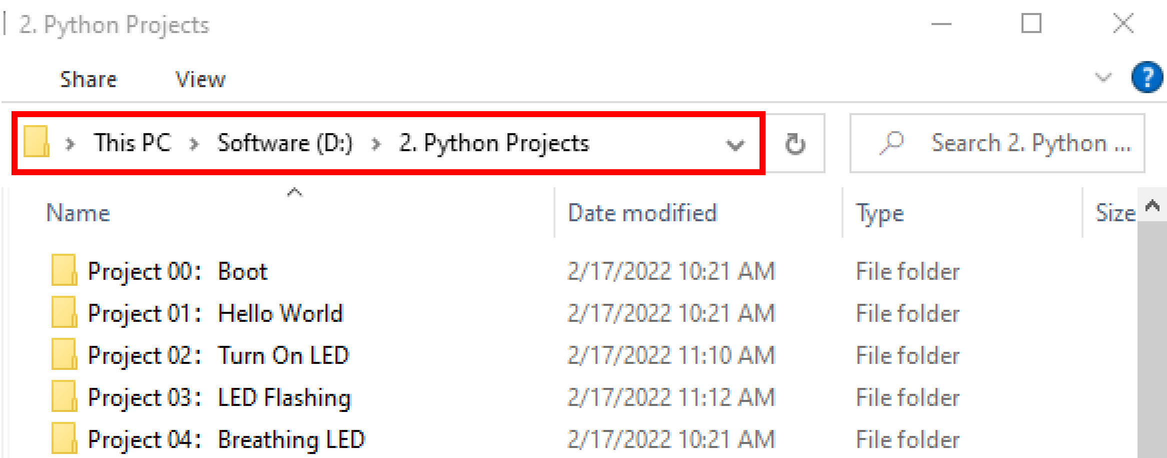


7.17.4 4. Wiring Diagram

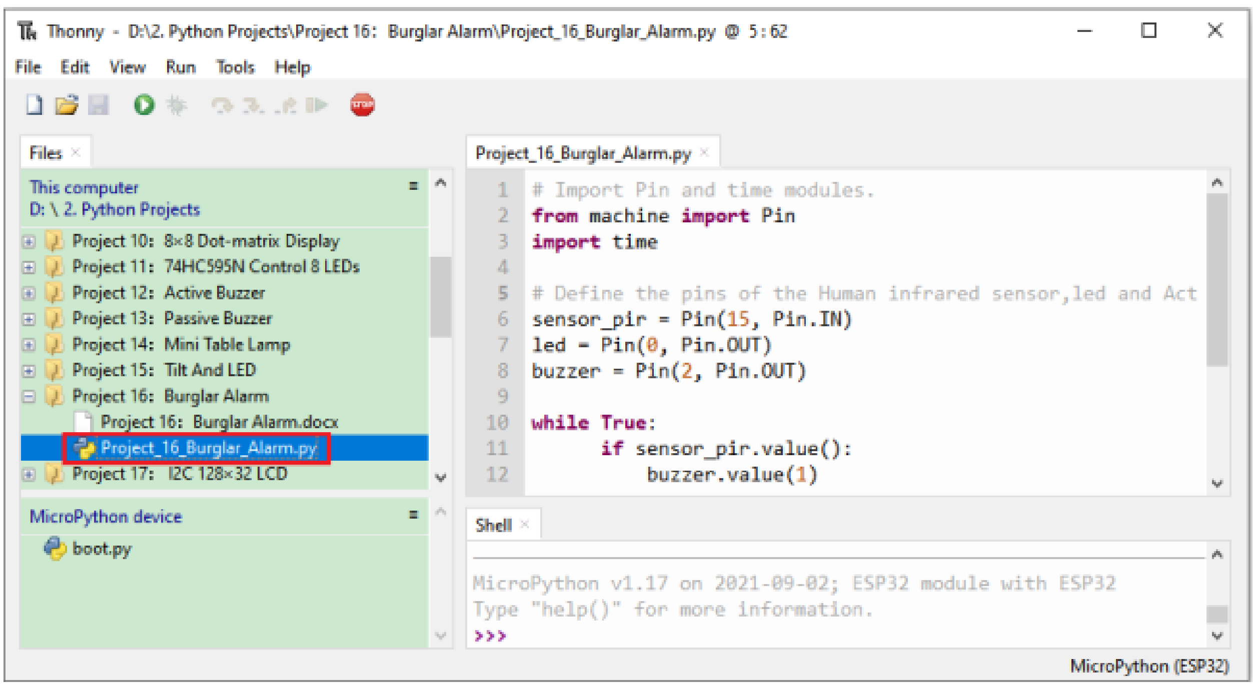


7.17.5 5.Project code

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



Open"Thonny"click"This computer"→"D:"→"2. Python Projects"→"Project 16Burglar Alarm", and then double left-click "Project_16_Burglar_Alarm.py".




```python
## 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)

```

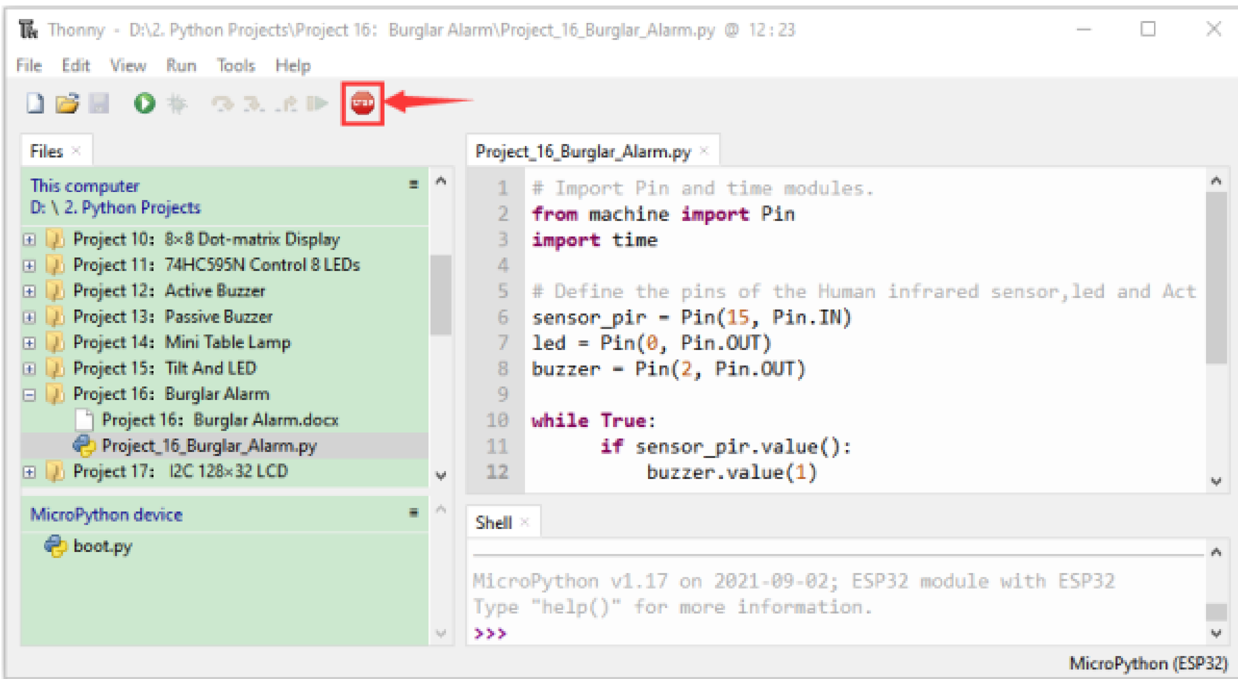
(continues on next page)

(continued from previous page)

```
while True:
      if sensor_pir.value():
          buzzer.value(1)
          led.value(1)
          time.sleep(0.2)
          buzzer.value(0)
          led.value(0)
          time.sleep(0.2)
      else:
          buzzer.value(0)
          led.value(0)
```

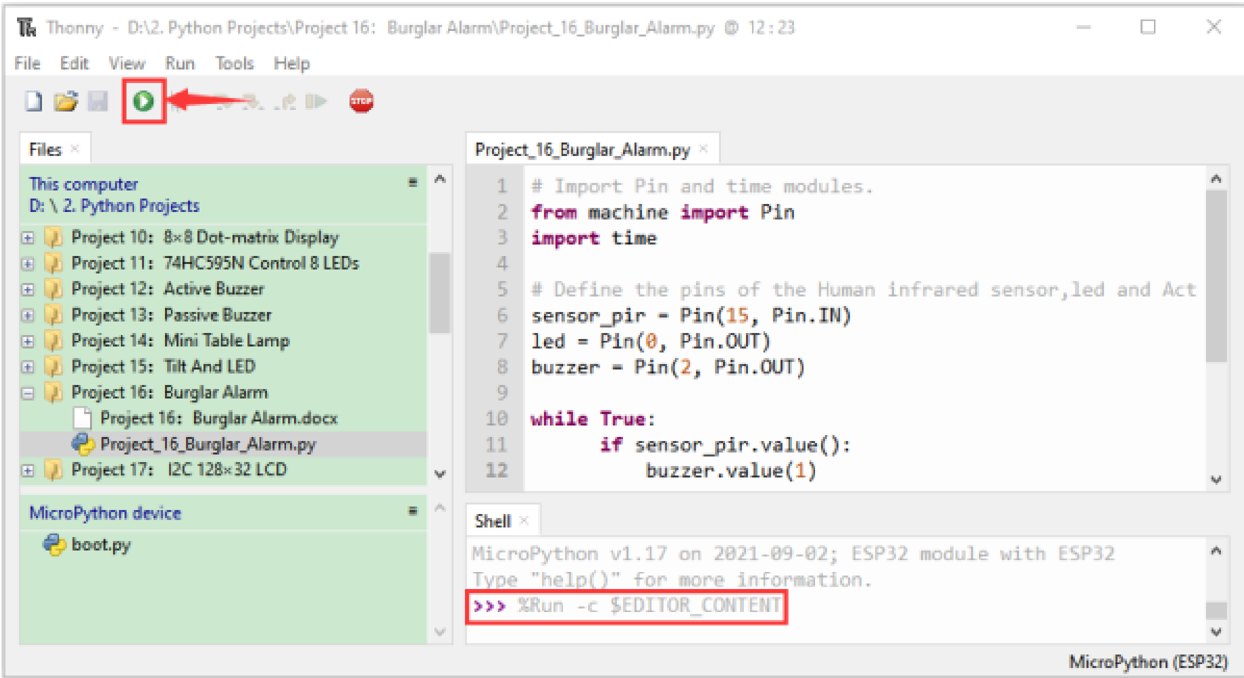
7.17.6 6.Project result

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 if the human body infrared sensor detects someone moving nearby, the buzzer will continuously issue an alarm and the LED will continuously flash.

Press"Ctrl+C"or click "Stop/Restart backend" to exit the program.

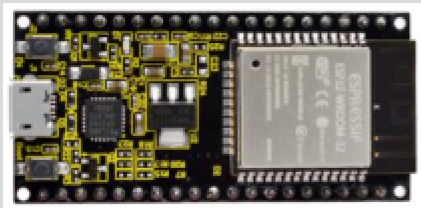
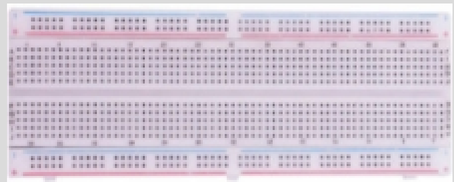
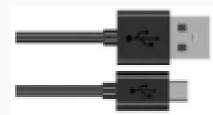


7.18 Project 17 I2C 128×32 LCD

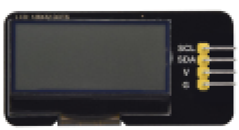
7.18.1 1.Introduction

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	
		
LCD_128X32_DOT*1	M-F Dupont Wires	USB Cable*1

7.18.3 3.Component knowledge

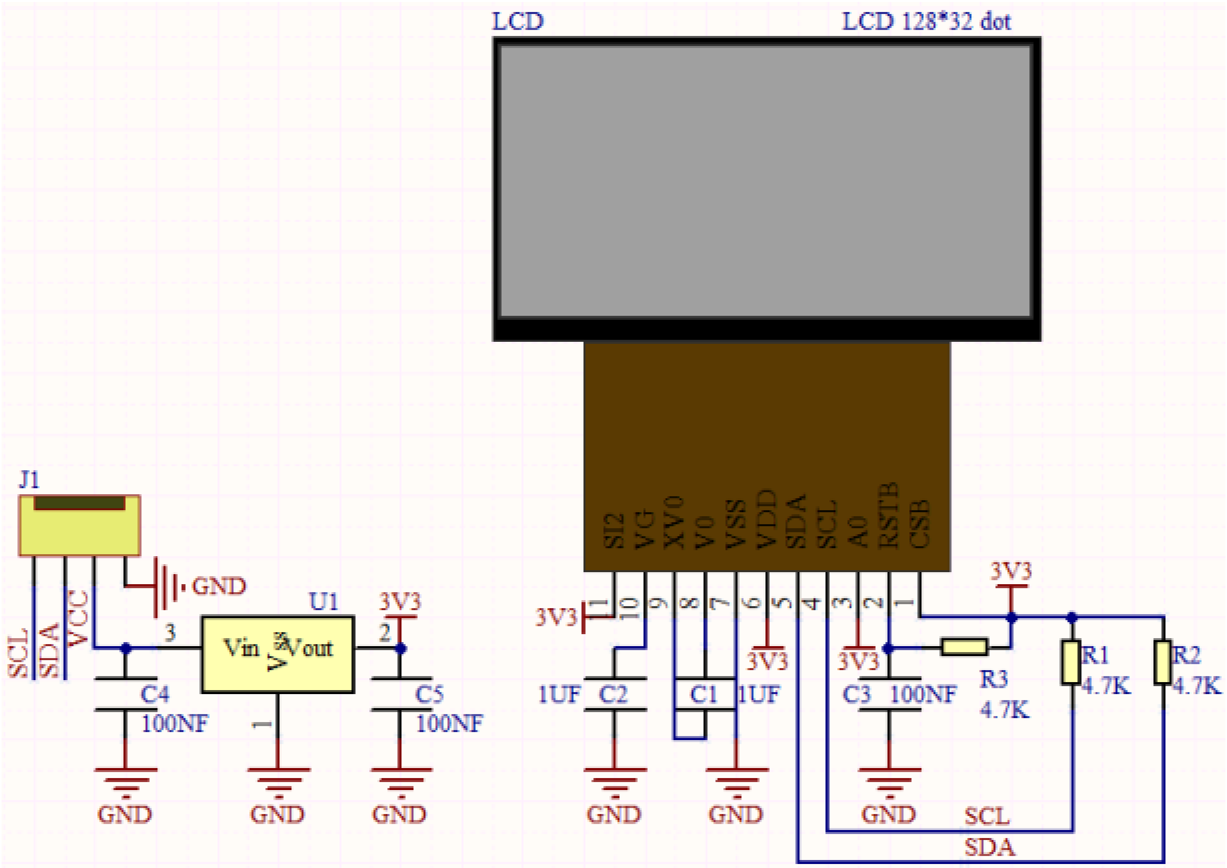


LCD_128X32_DOT:

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.

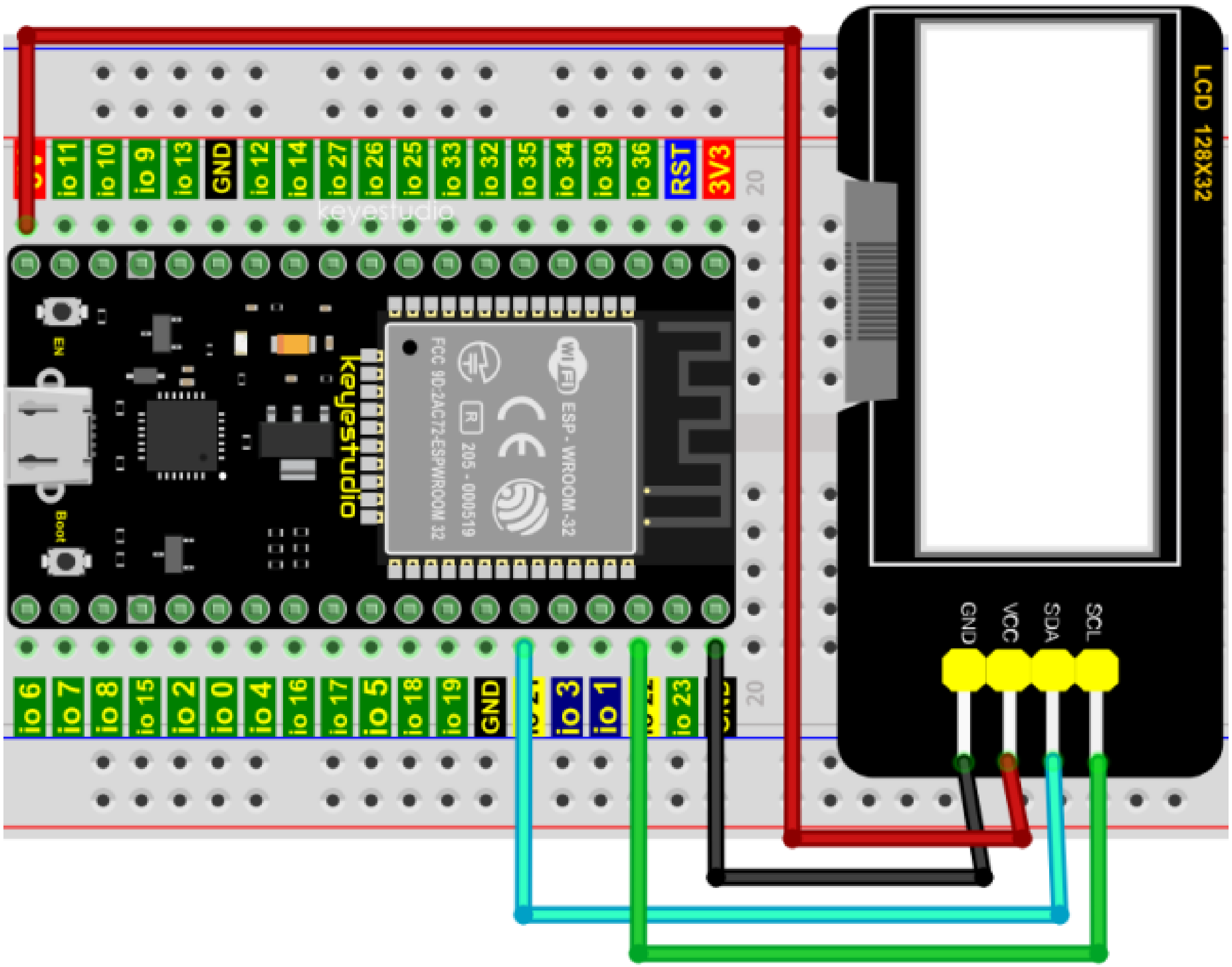
Schematic diagram of LCD_128X32_DOT



Features:

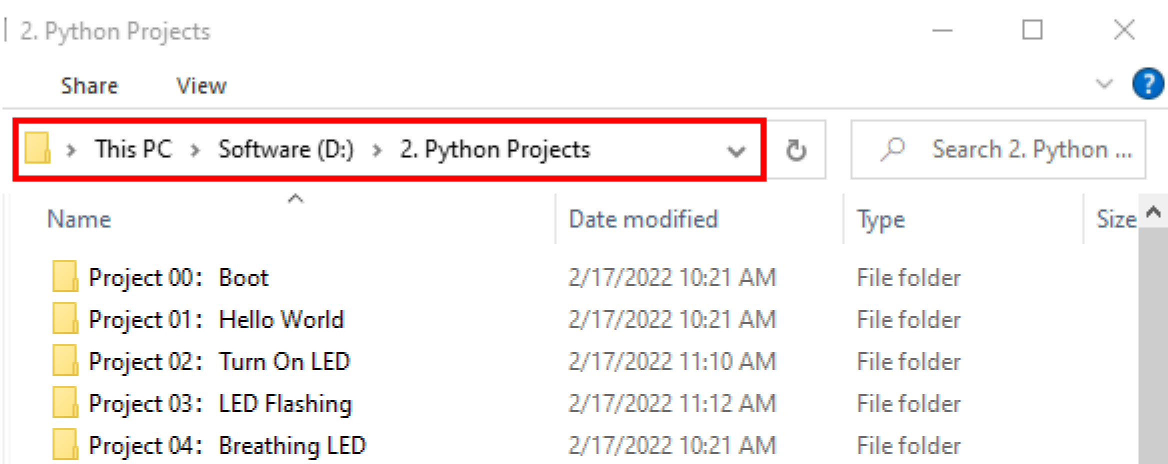
- Pixel: 128*32 character
- Operating voltage(chip)4.5V to 5.5V
- Operating current100mA (5.0V)
- Optimal operating voltage(module):5.0V

7.18.4 4.Wiring Diagram

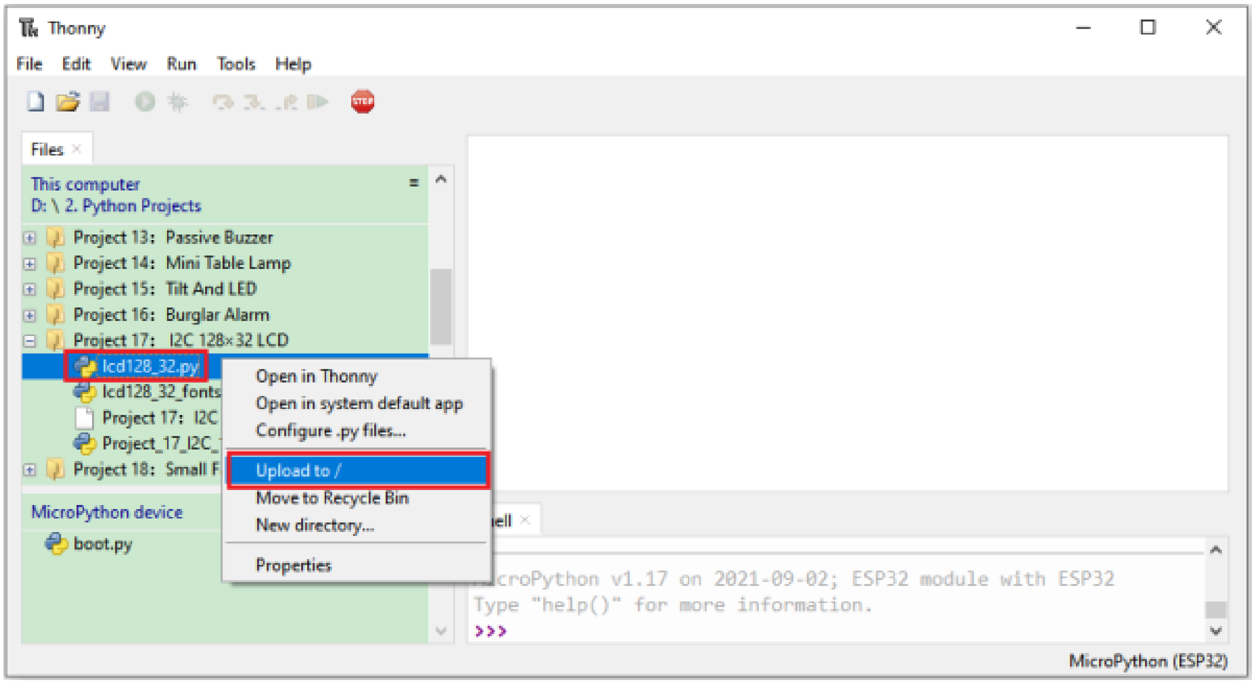


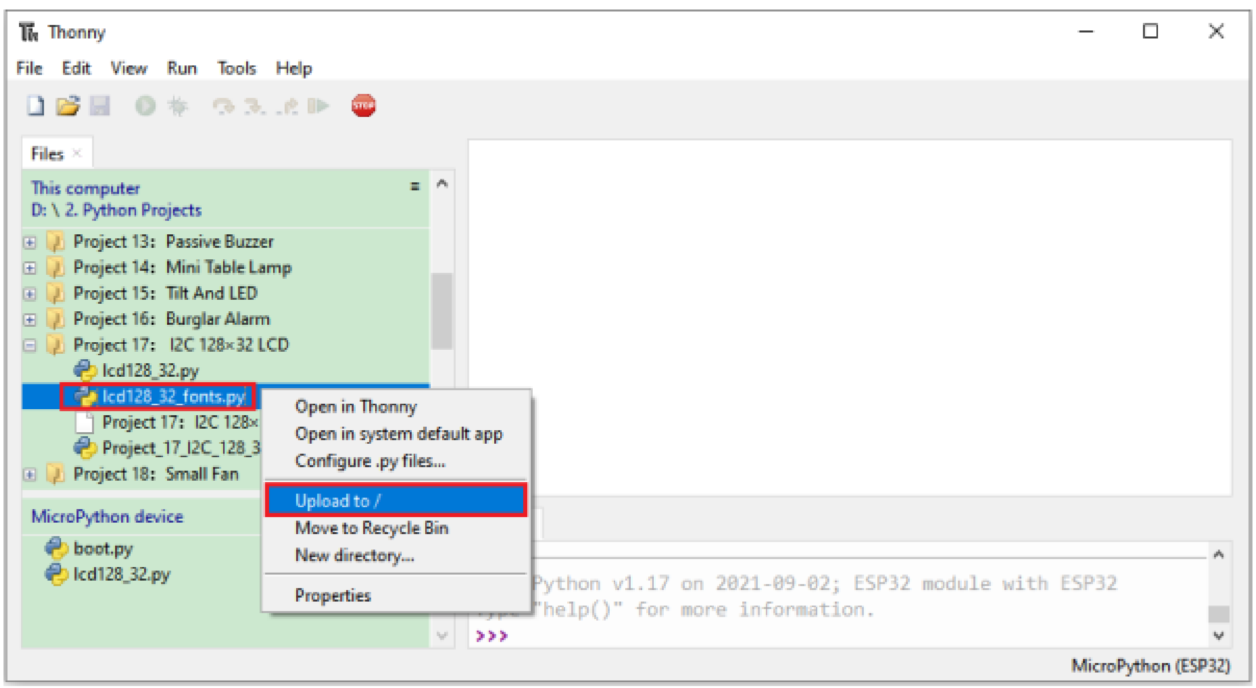
7.18.5 5.Project code

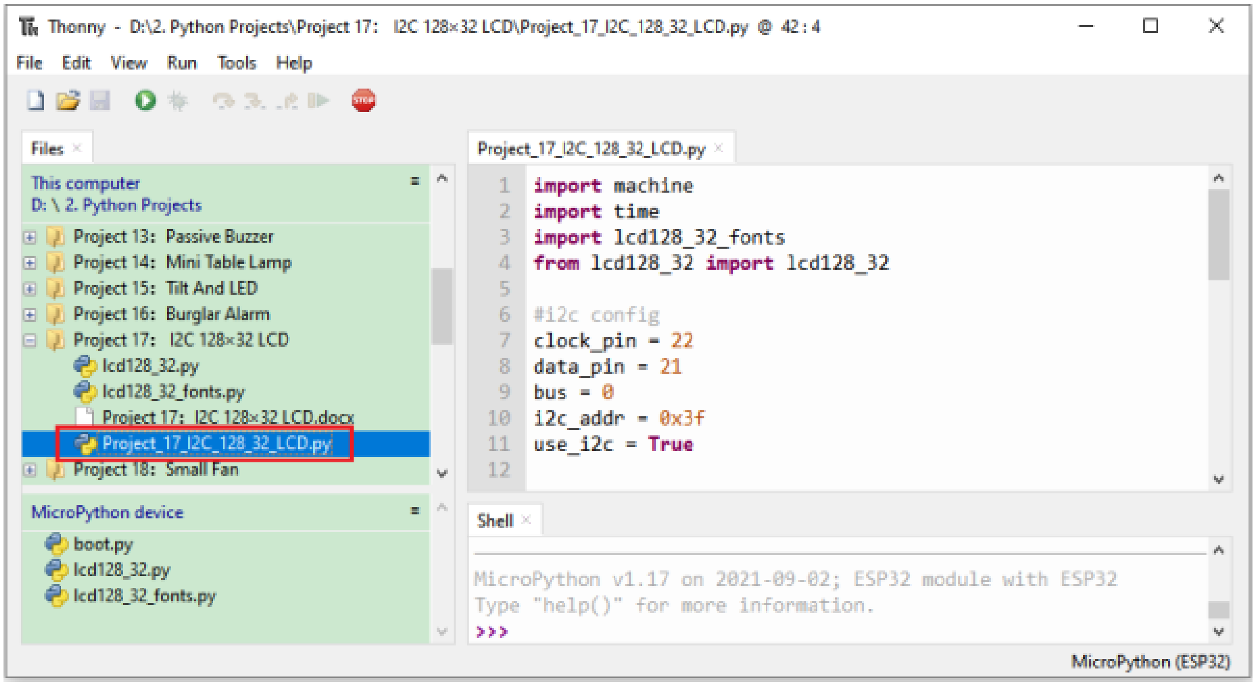
Codes used in this tutorial are saved in“**2. Python Projects**”. If you haven’t downloaded the code file, please click on the link to download it:Download Python Codes



Open“Thonny”click“This computer”→“D:”→“2. Python Projects”→“Project 17I2C 128×32 LCD”. Select“lcd128_32.py”and “lcd128_32_fonts.py”click your mouse to select“Upload to”wait for“lcd128_32.py”and“lcd128_32_fonts.py”to be uploaded to ESP32and 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, 0)
lcd.Display("ABCDEFGHIJKLMNOPQR")
lcd.Cursor(2, 0)
```

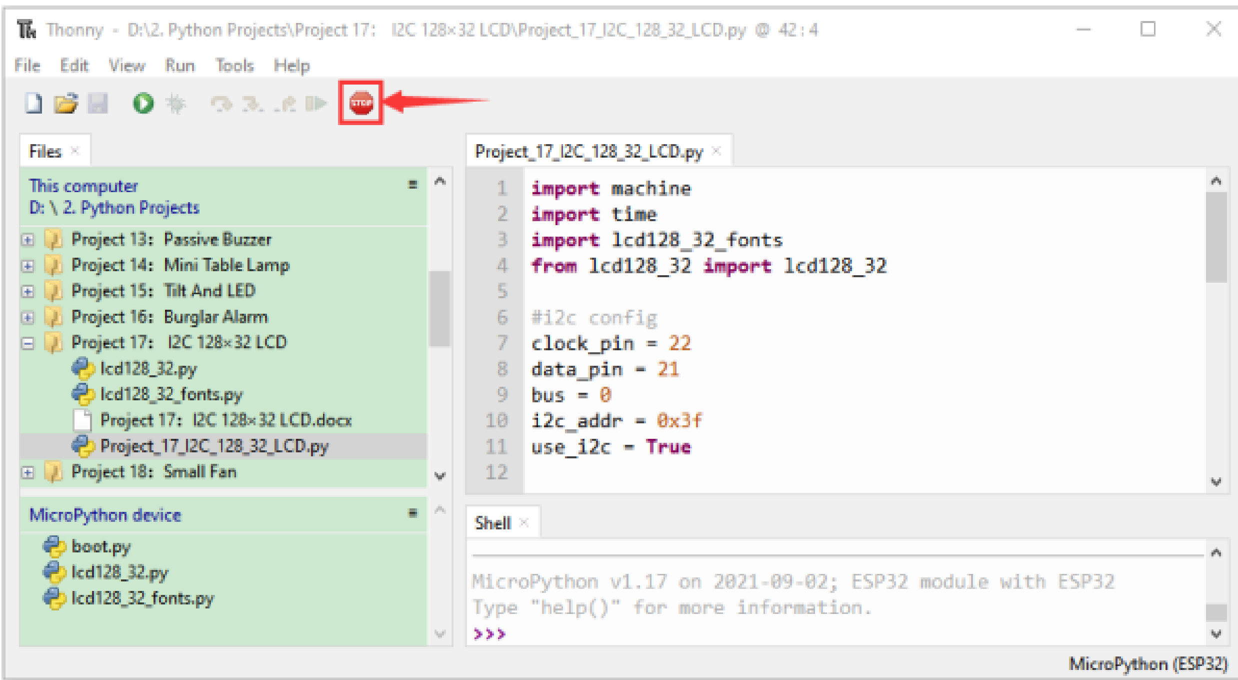
(continues on next page)

(continued from previous page)

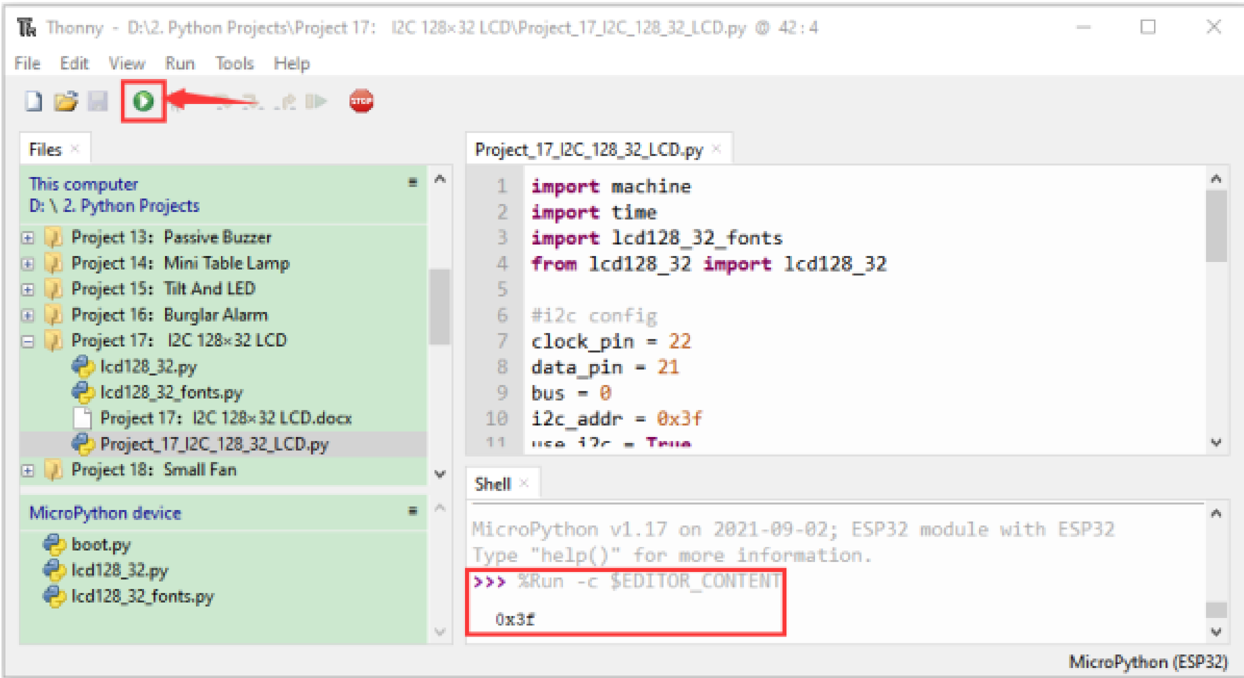
```
lcd.Display("123456789+-*/<>=$@")
lcd.Cursor(3, 0)
lcd.Display("%^&(){}:;'|?,.~\\[]")
"""
while True:
    scan_for_devices()
    time.sleep(0.5)
"""
```

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±*/<>=$@" will be shown at the third line and "%^&(){}:;'|?,.~\[]" will be displayed at the fourth line. Press "Ctrl+C" or click "Stop/Restart backend" to exit the program.

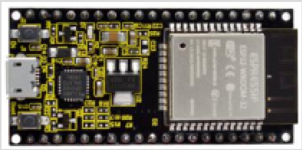
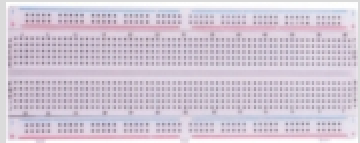
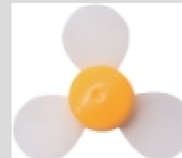
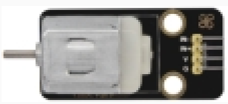
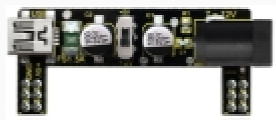
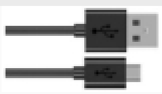


7.19 Project 18Small Fan

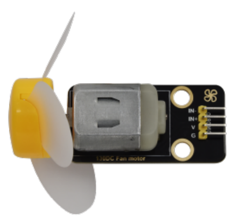
7.19.1 1.Introduction

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

			
ESP32*1	Breadboard*1	Battery Holder*1	Fan*1
			
130 Motor Module*1	Keyestudio bread board special power module*1	M-F Dupont Wires	No.5 battery (self-provided)*6
			
USB Cable*1			

7.19.3 3.Component knowledge :

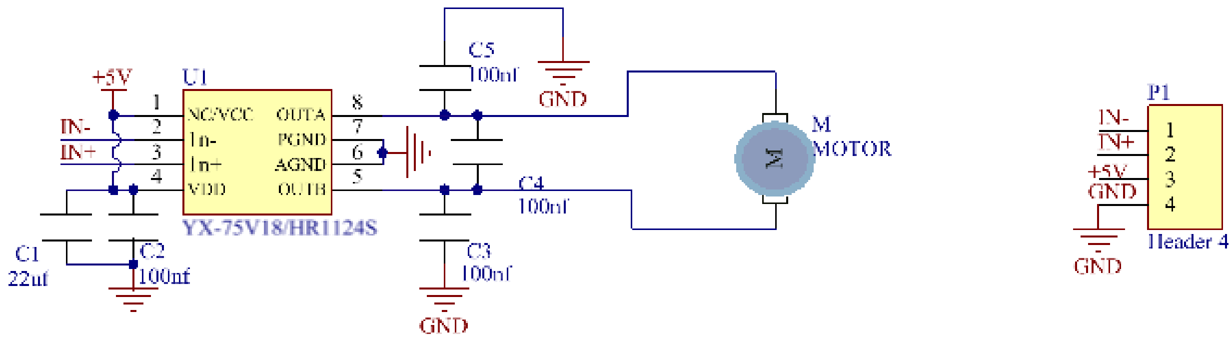


130 motor module: The motor control module uses the HR1124S motor control chip. which is a single-channel H-bridge driver chip for DC motor. The H-bridge driver part of the HR1124S uses low on-resistance PMOS and NMOS power tubes. The low on-resistance ensure low power loss of the chip and make the chip 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.

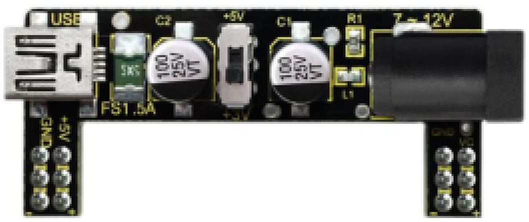
Features:

- Working voltage: 5V
- Working current: 200MA
- Working power: 2W
- Working temperature: -10℃~ +50℃

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.

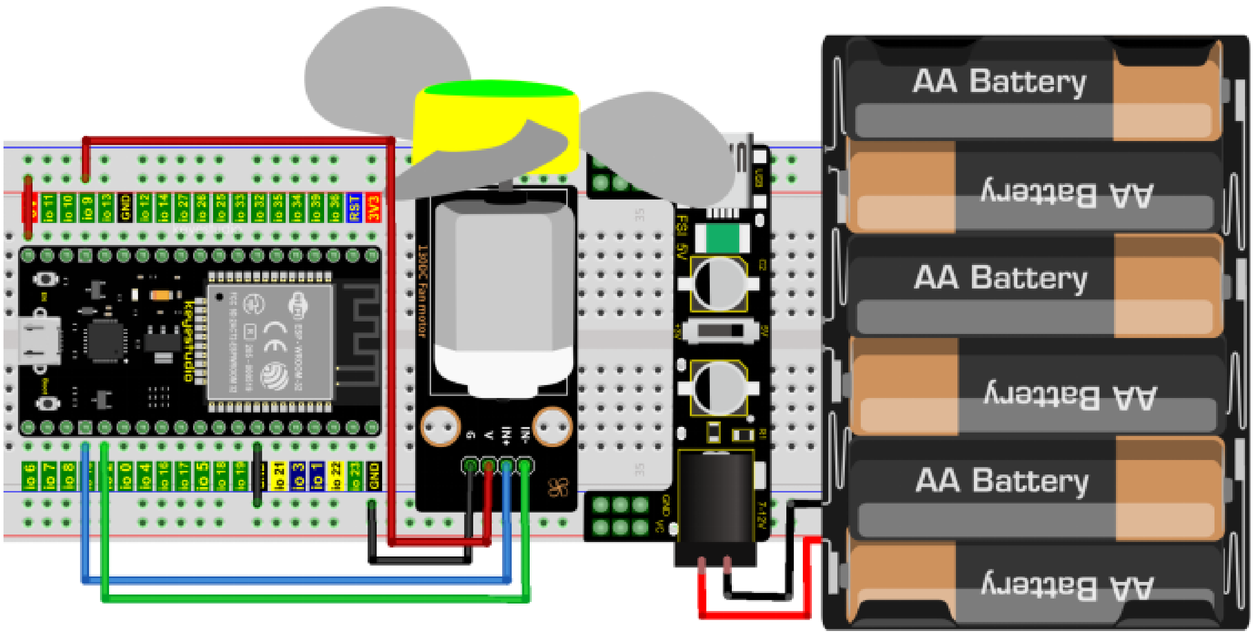
The output voltage is constant for the DC5V, and another way is powered by DC6.5-12V, output controlled by the slide switch, respectively for DC5V and DC3.3V.

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.

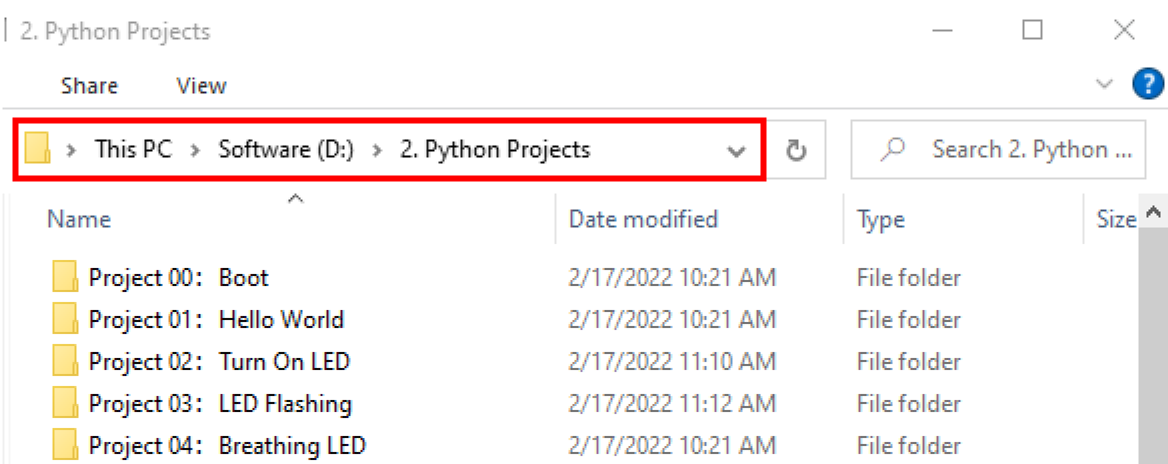
7.19.4 4. Wiring Diagram



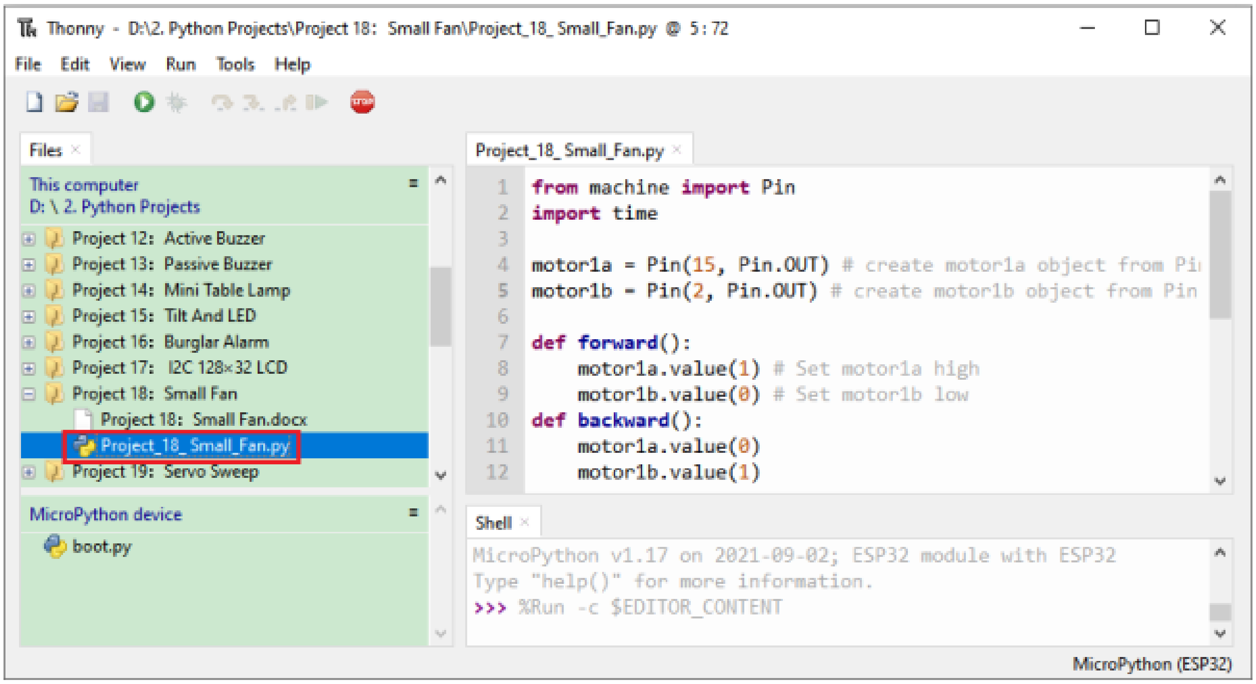
(Note: Connect the wires and then install a small fan blade on the DC motor.)

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



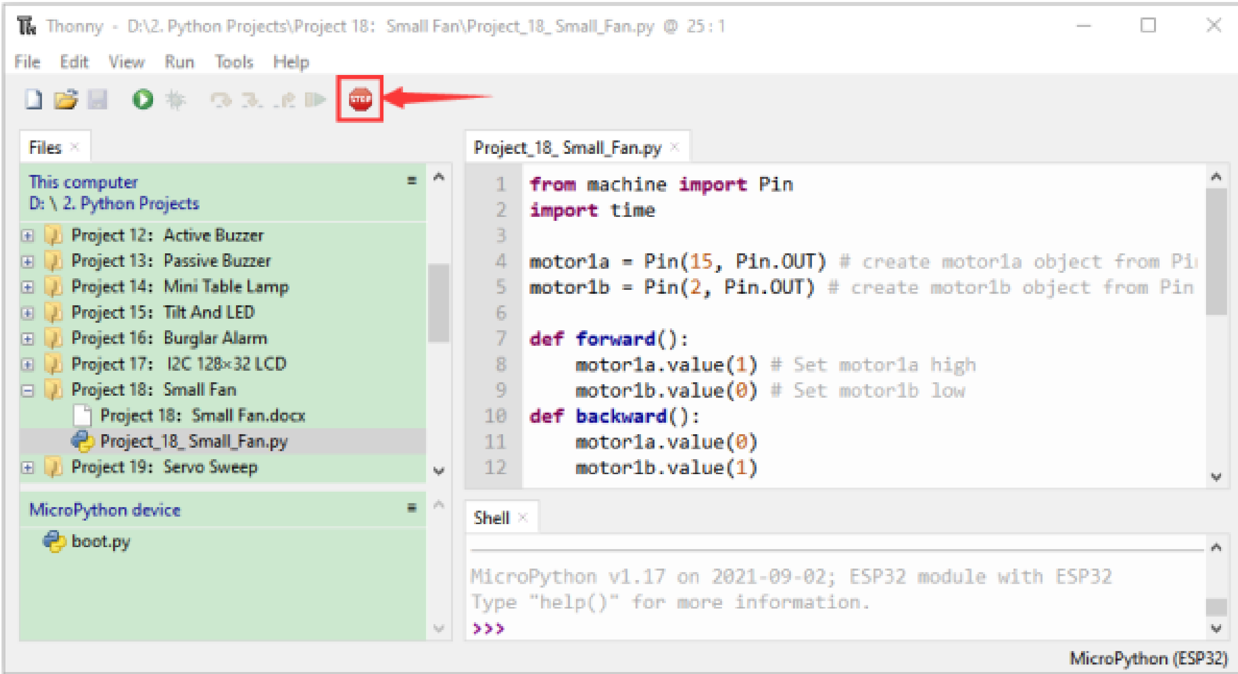
Open“Thonny”click“This computer”→“D:”→“2. Python Projects”→“Project 18Small Fan”, and then double left-click“Project_18_ Small_Fan.py”.



```
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 “Stop/Restart backend” .

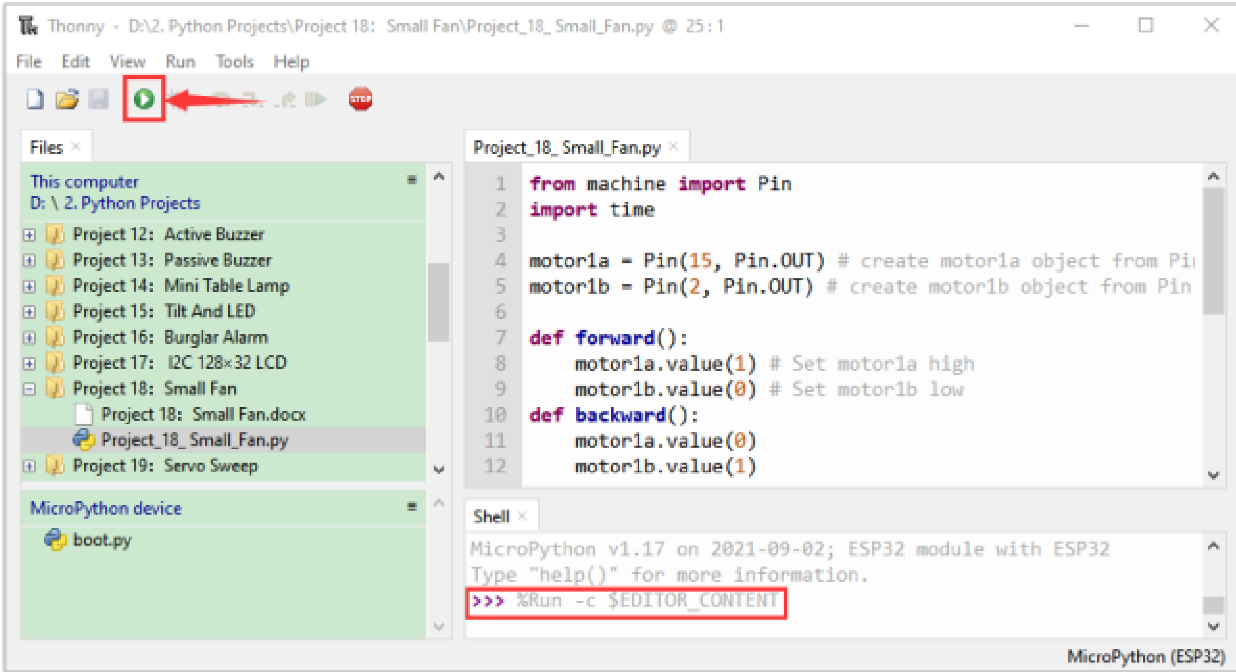


External power supply and power on.

Click “Run current script”, the code starts to be executed and you’ll 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.

Repeat this rule for 5 times and then the small fan stops.

Press“Ctrl+C”or click “Stop/Restart backend”to exit the program.

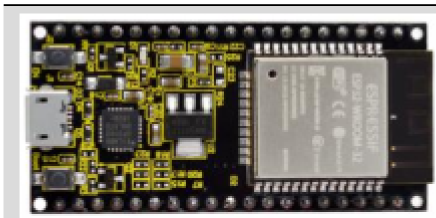
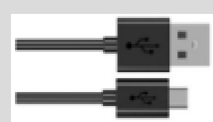
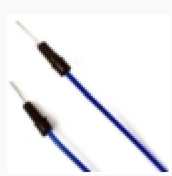


7.20 Project 19Servo Sweep

7.20.1 1.Introduction

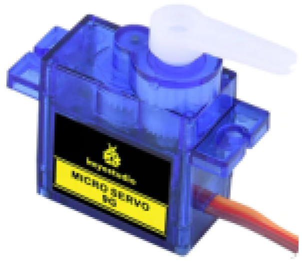
Servo is an electric motor that can rotate very precisely. At present, it has been widely used in toy carsremote control helicoptersairplanesrobots, etc. In this project, we will use ESP32 to control the rotation of the servo.

7.20.2 2.Components

		
ESP32*1	Breadboard*1	USB Cable*1
		
Servo*1	Jumper Wires	

7.20.3 3.Component knowledge

Servo



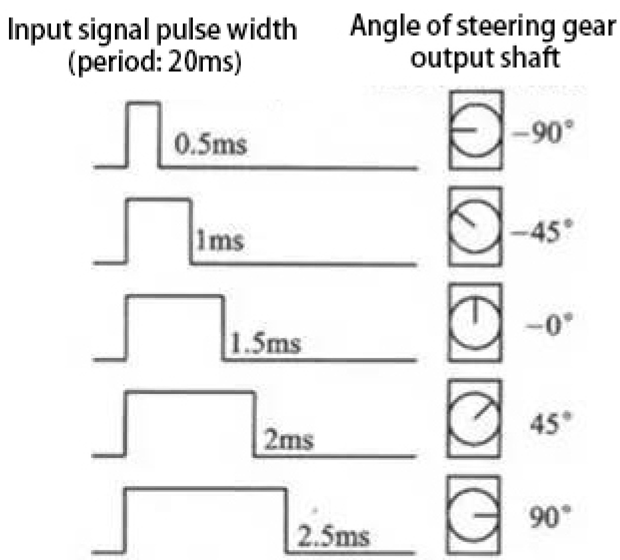
The servo is a kind of position servo driver, which is mainly composed of housing, circuit board, copless motor, gear and position detector.

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.

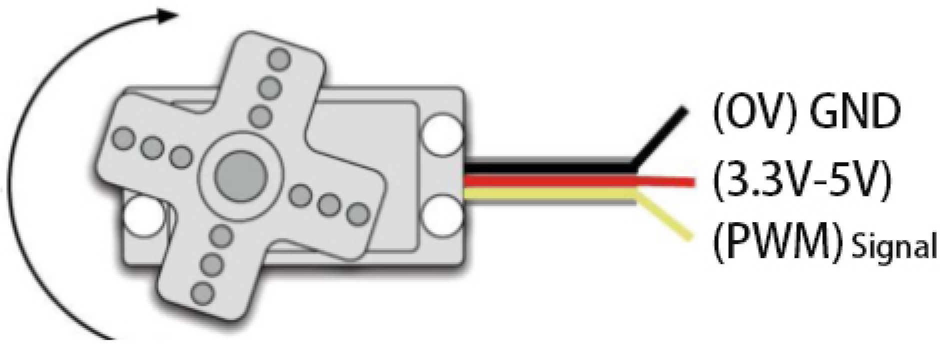
It is suitable for those control systems that require constant change of angle and can be maintained.

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

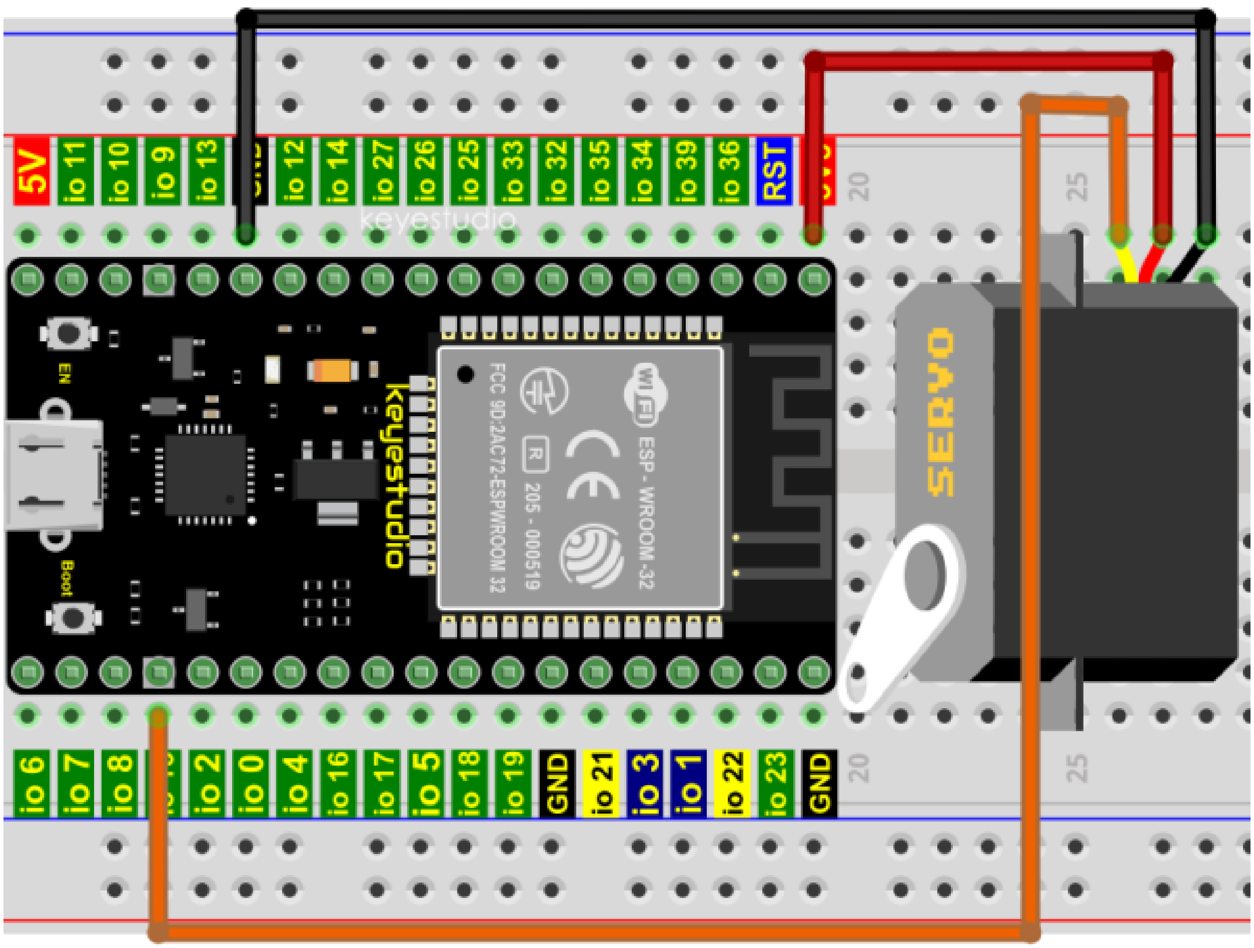


Servo motors have many specifications, but they all have three connecting wires, which are brown, red, and orange (different brands may have different colors). The brown is GND, the red is the positive power supply, and the orange is the signal line.



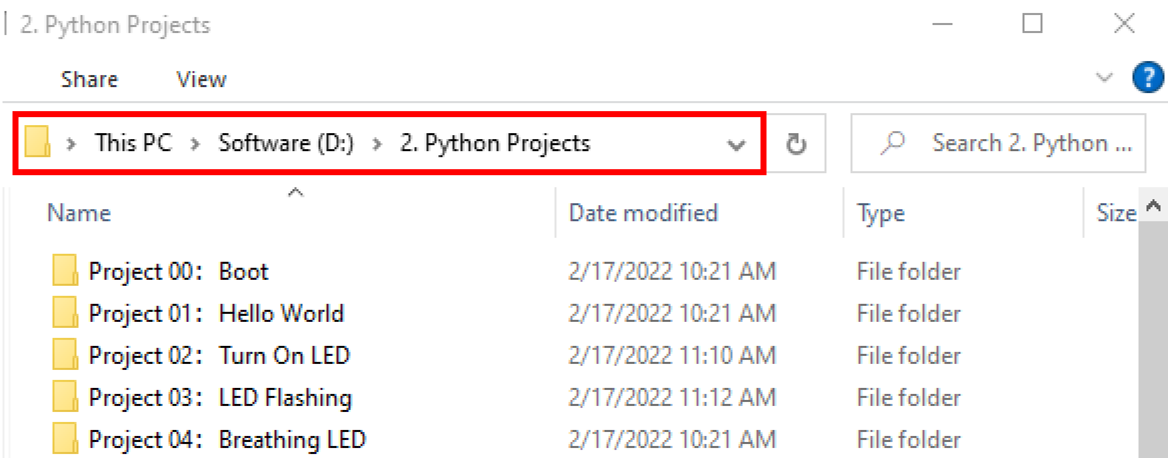
7.20.4 4.Wiring Diagram

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.

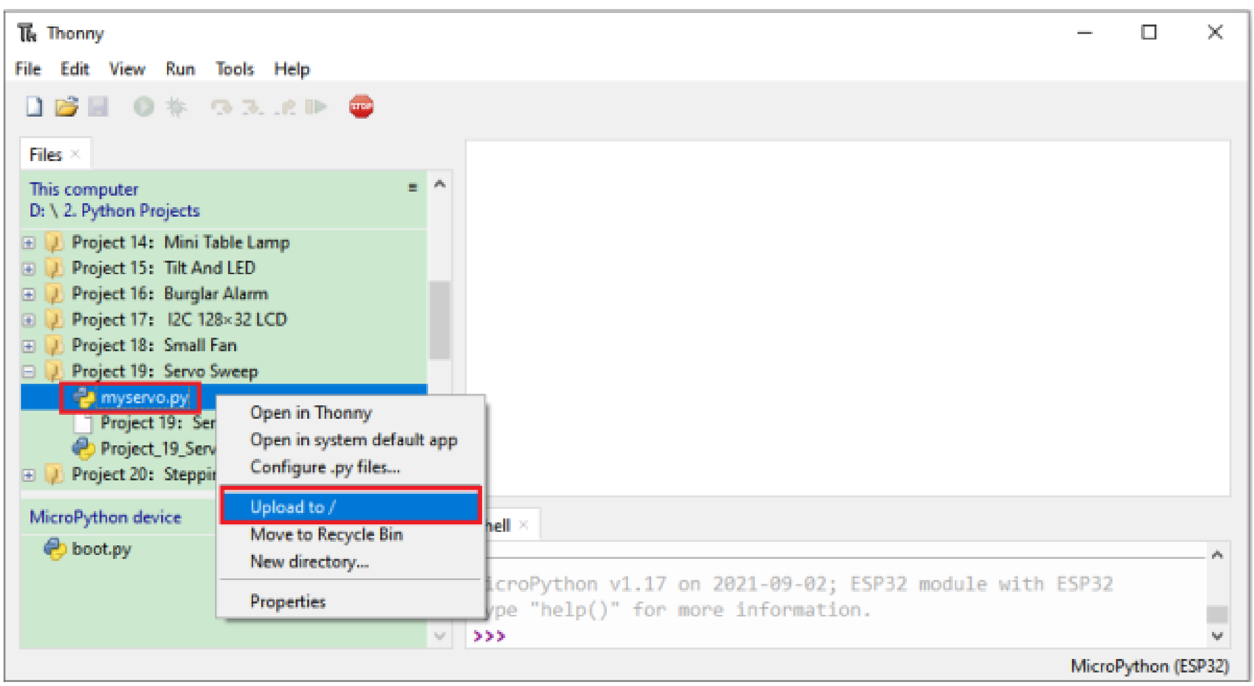


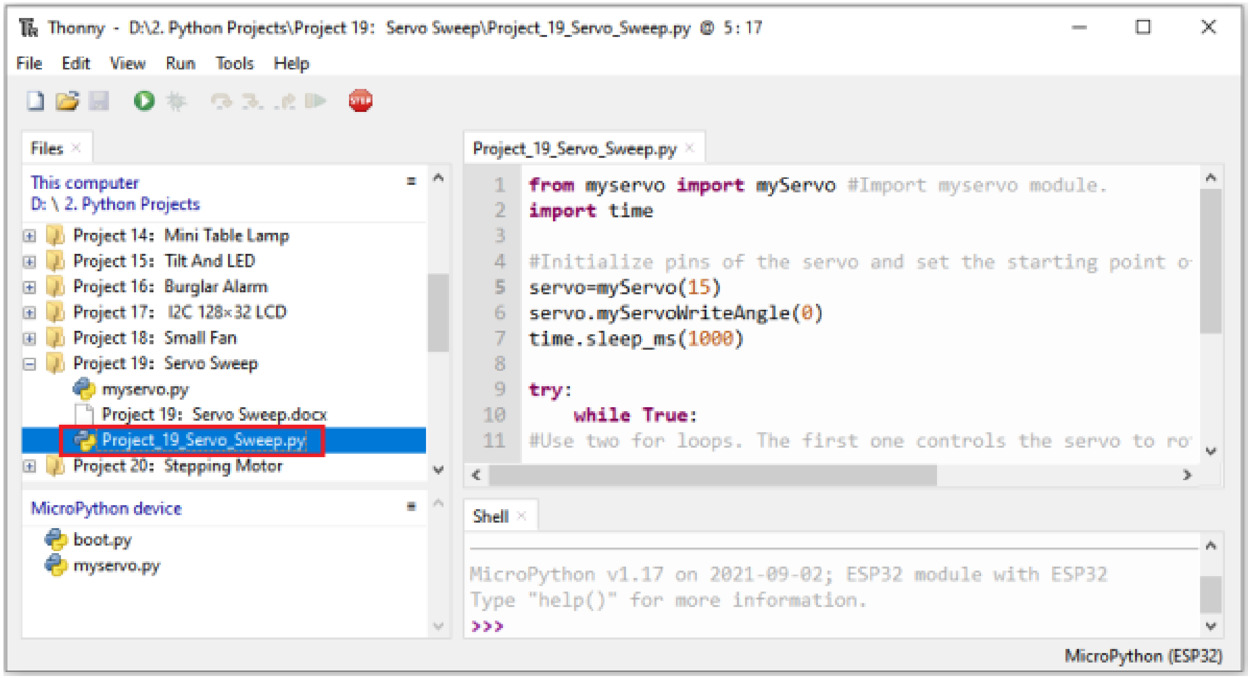
7.20.5 5.Project code

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



Open“Thonny”click“This computer”→“D:”→“2. Python Projects”→“Project 19Servo Sweep”. Select“myservo.py”right-click your mouse to select“Upload to /”wait for“myservo.py”to be uploaded to ESP32and then click“Project_19_Servo_Sweep.py”.

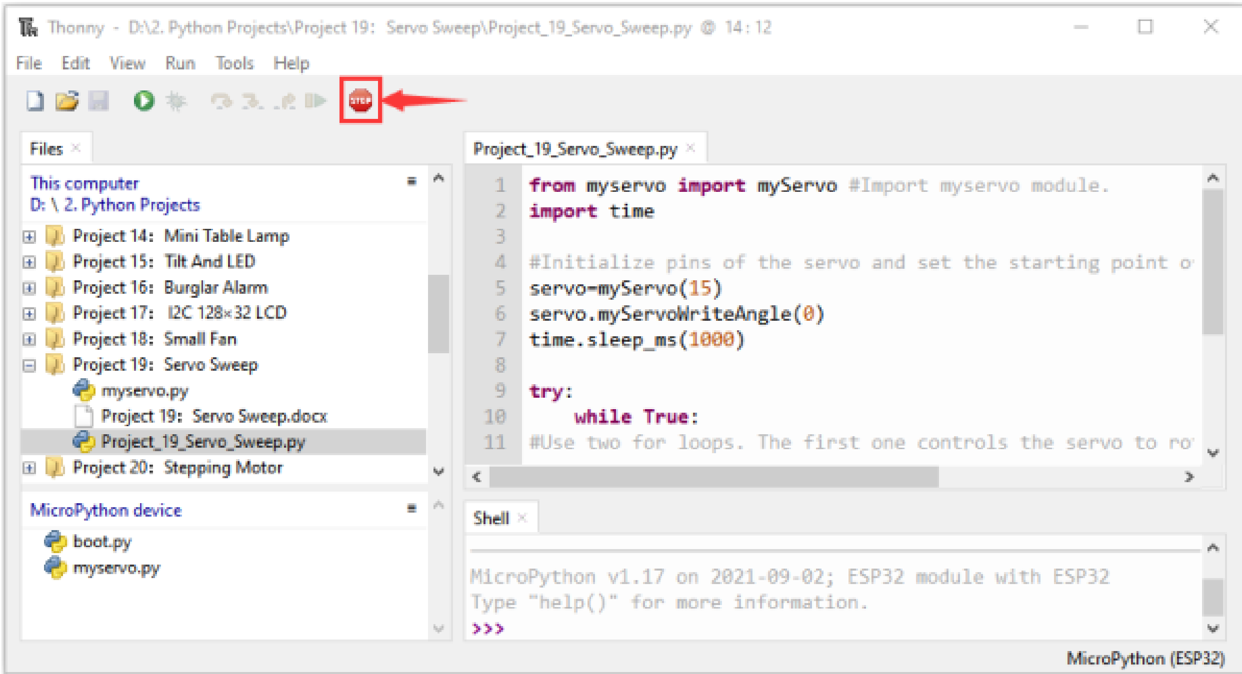





```python
from myservo import myServo #Import myservo module.
import time

#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()
```

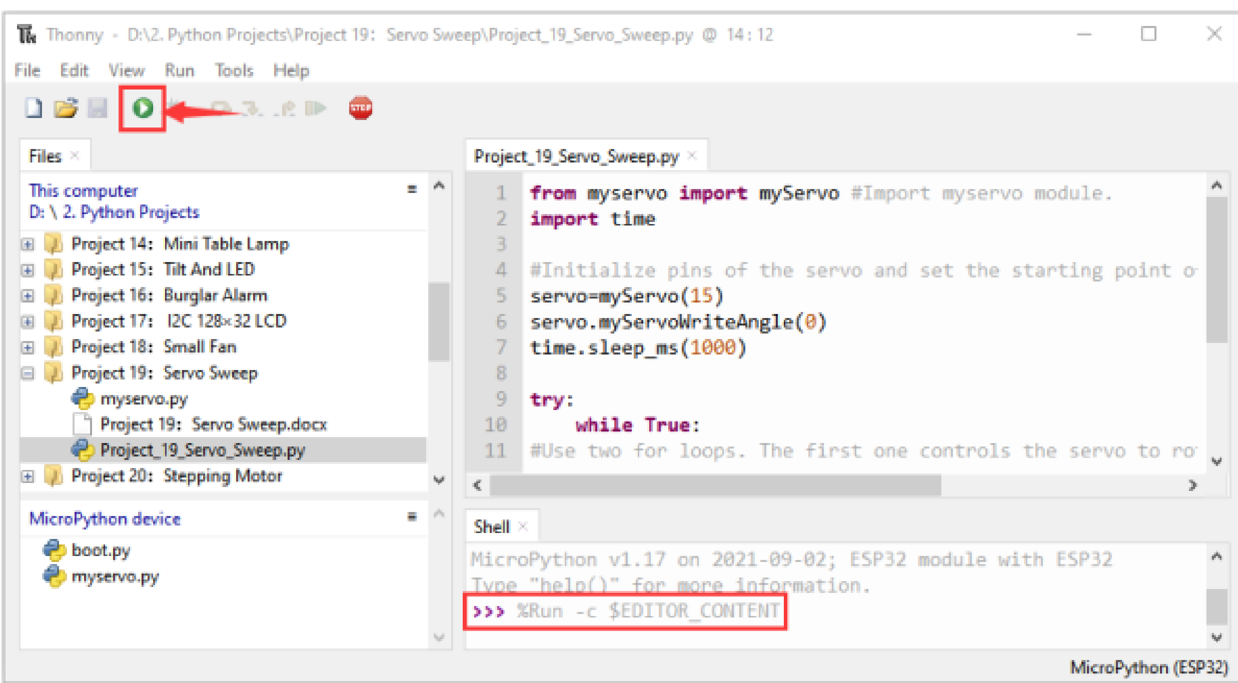
7.20.6 6.Project result

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“Stop/Restart backend” to exit the program.

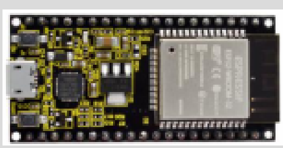
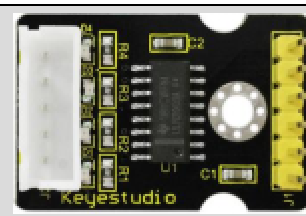
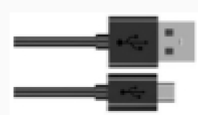
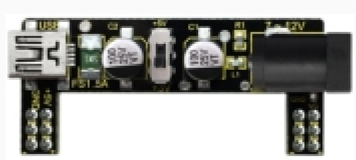


7.21 Project 20Stepping Motor

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

		
ESP32*1	Breadboard*1	ULN2003 Stepper Motor Drive Board*1
		
Stepper Motor *1	M-F Dupont Wires	USB Cable*1
		
Battery Holder*1	Keyestudio bread board special power module*1	No.5 battery (self-provided)*6

7.21.3 3.Component knowledge :

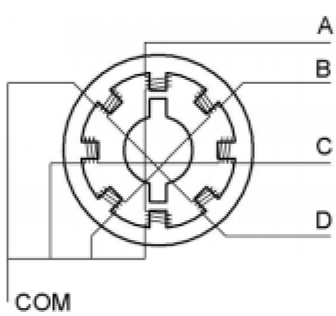
Stepper motor:

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.

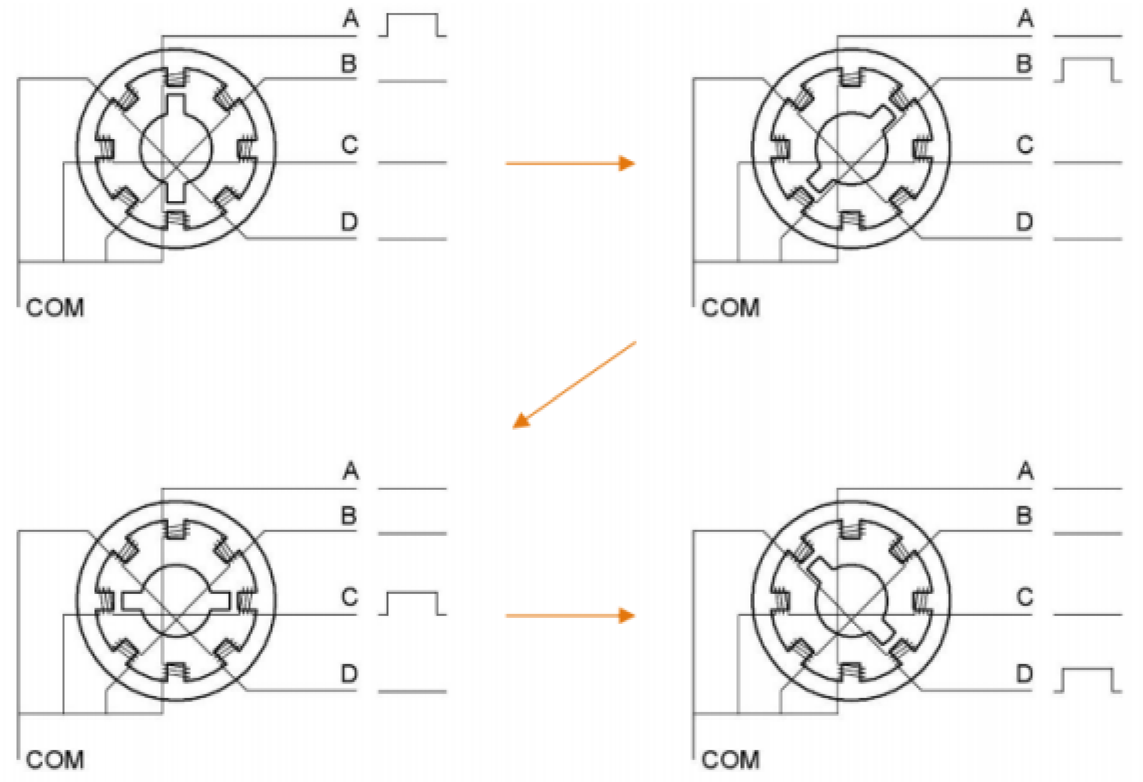


Single -phase four beat:

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 is turned on, and the rotor is turned to 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.

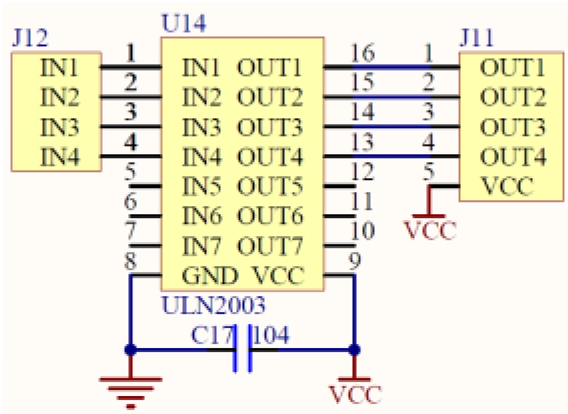
Stepper Motor Parameters:

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.

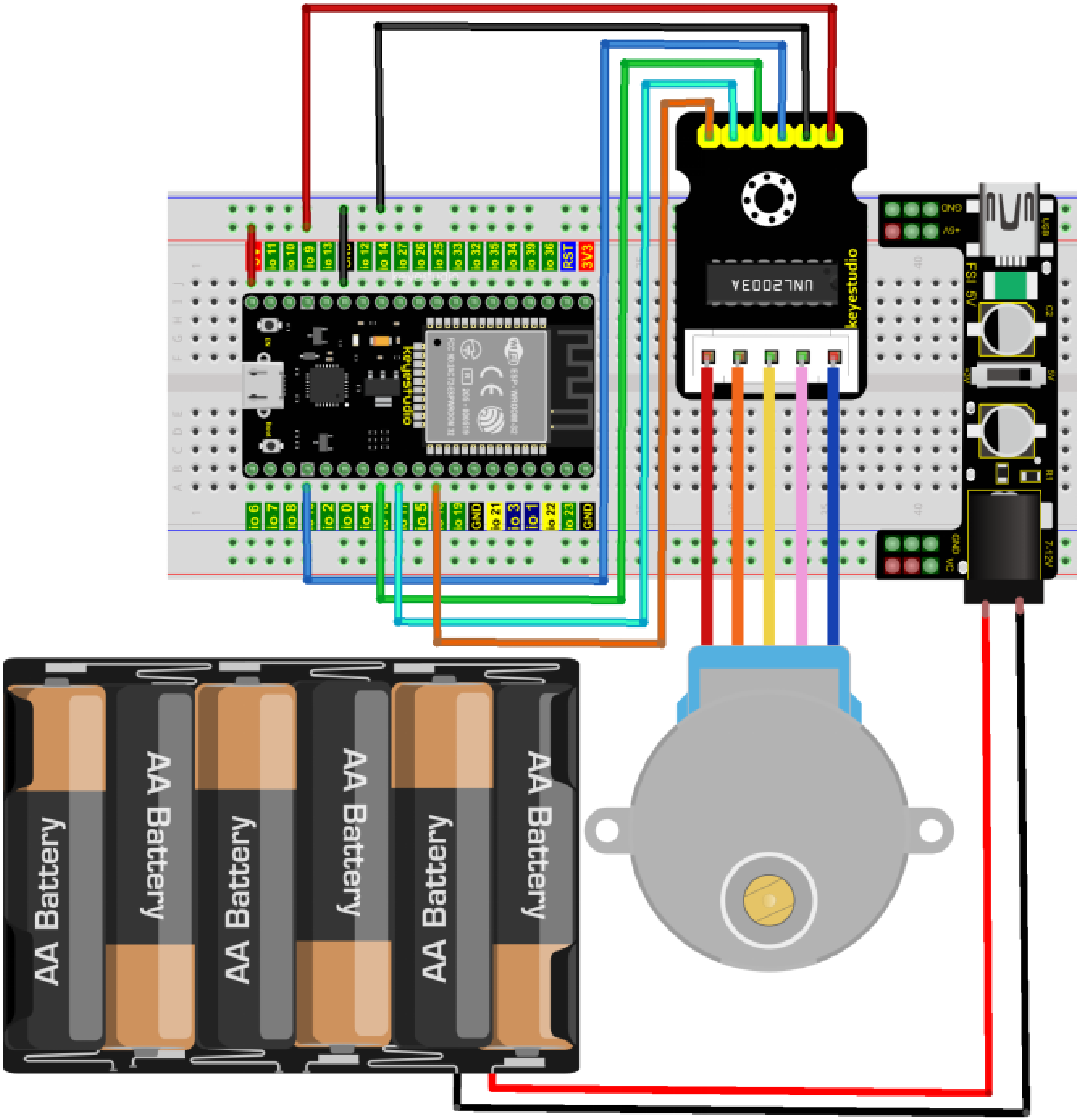
The step angle of 4-beat mode of 5V and 4-phase stepper motor is 11.25. And the step angle of 8-beat mode is 5.625, the reduction ratio is 1:64.

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.

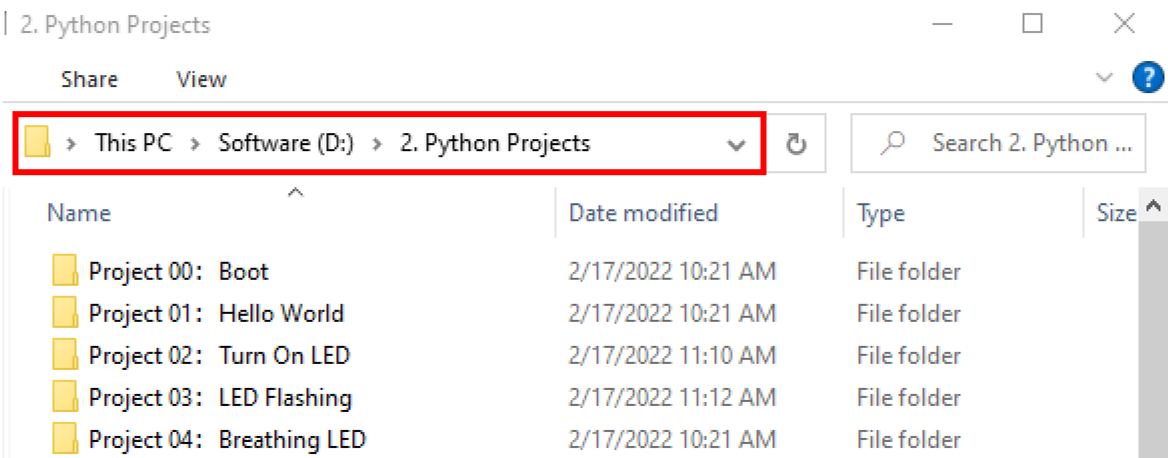

J12
IN1
IN2
IN3
IN4
U14
IN1 OUT1
IN2 OUT2
IN3 OUT3
IN4 OUT4
IN5 OUT5
IN6 OUT6
IN7 OUT7
GND VCC
ULN2003
J11
OUT1
OUT2
OUT3
OUT4
VCC
VCC
C17 104
VCC

7.21.4 4.Wiring Diagram

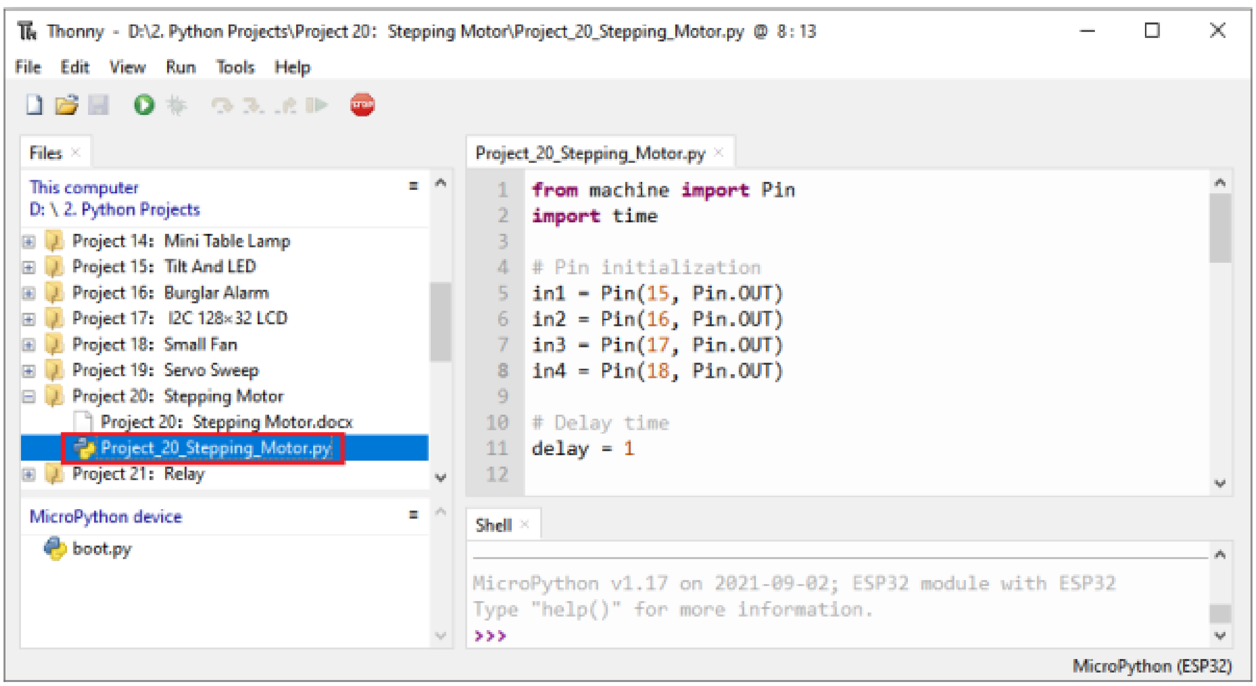


7.21.5 5.Project code

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



Open"Thonny"click"This computer"→"D:"→"2. Python Projects"→"Project 20Stepping Motor", and then double left-click "Project_20_Stepping_Motor.py".




```python
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°),␣
↪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(0)
    in3(0)
    in4(0)

## 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)
```

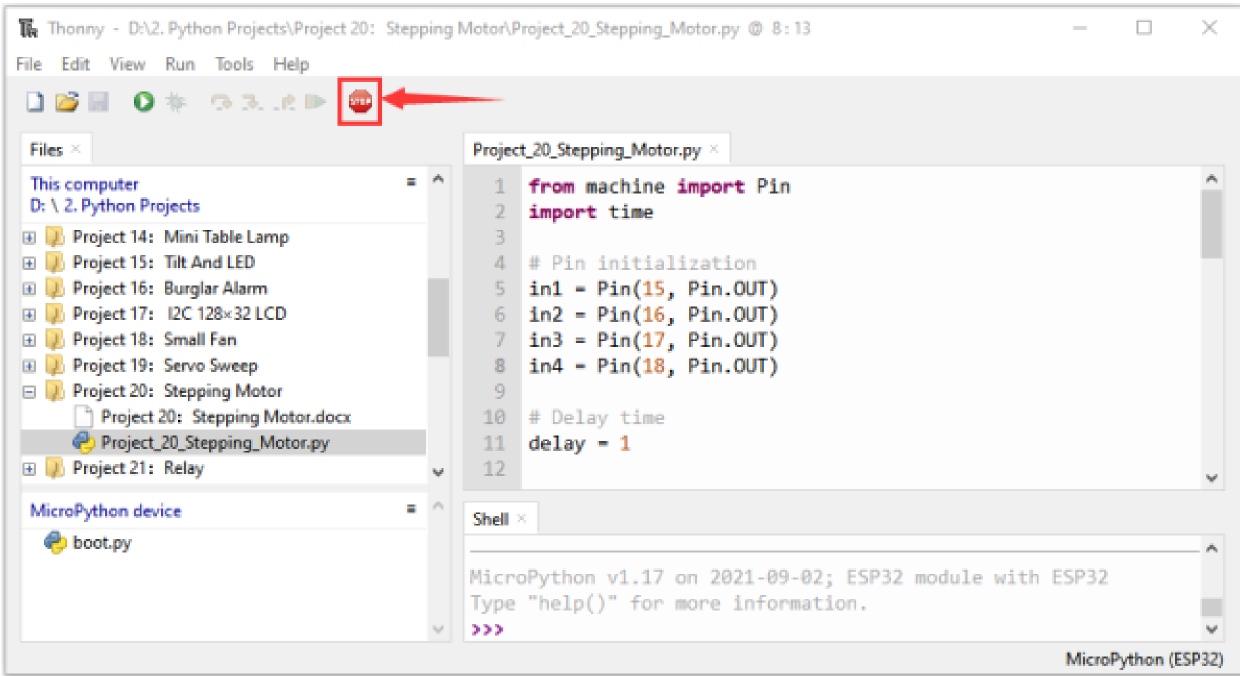
(continues on next page)

(continued from previous page)

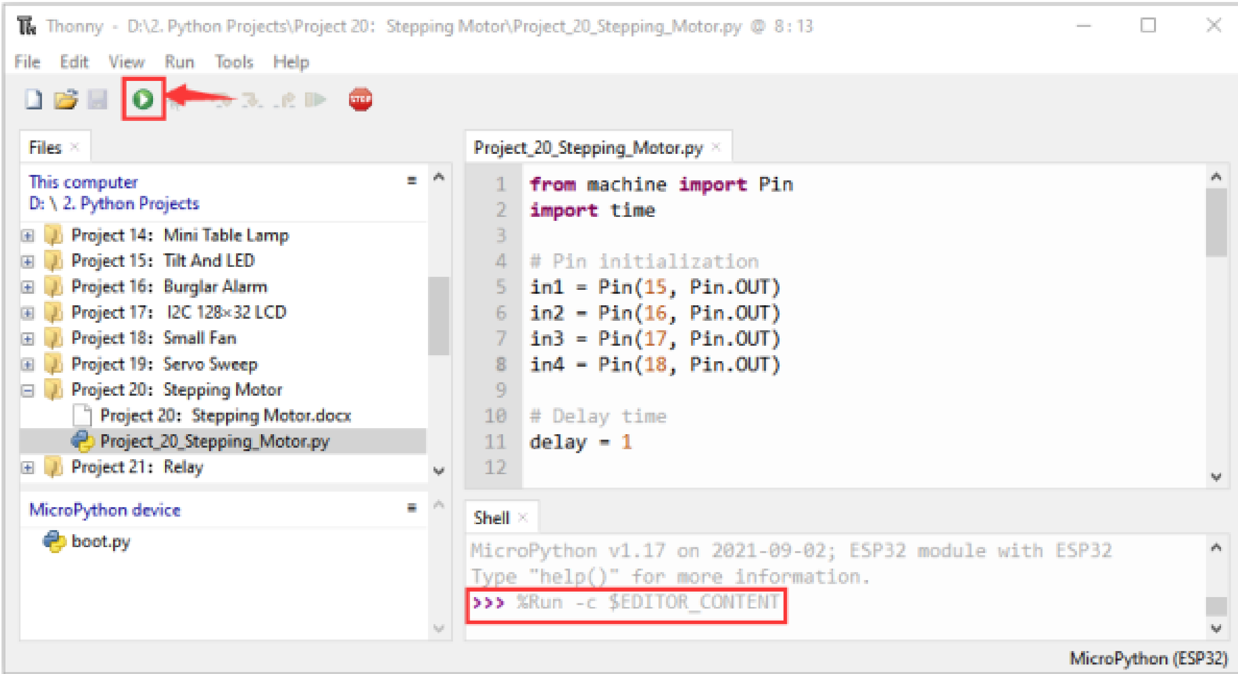
```
    step_angle(-360)
```

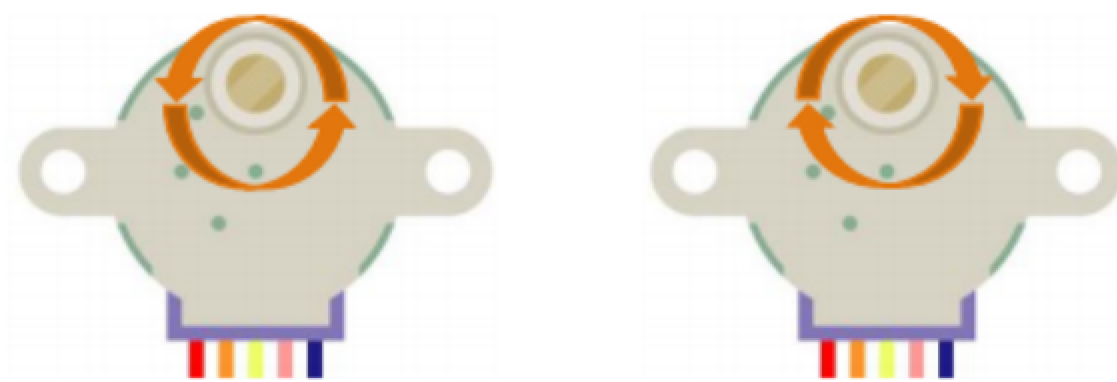
7.21.6 6.Project result

Make sure the ESP32 has been connected to the computer, click “Stop/Restart backend” .



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

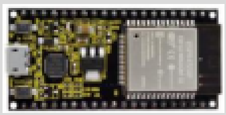
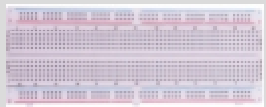
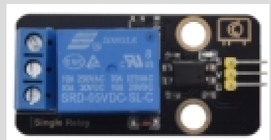
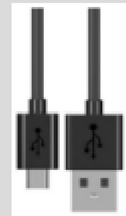
7.22.1 1.Introduction

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

				
ESP32*1	Breadboard*1	Relay Module*1	M-F Dupont Wires	USB Cable*1

7.22.3 3.Component knowledge

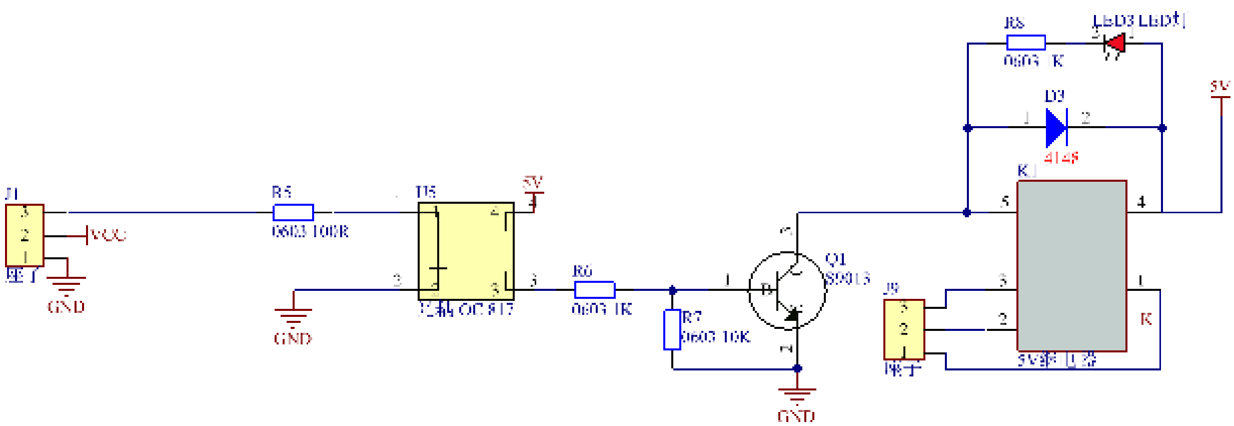
Relay:

It is an “automatic switch” that uses a small current to control the operation of a large current.

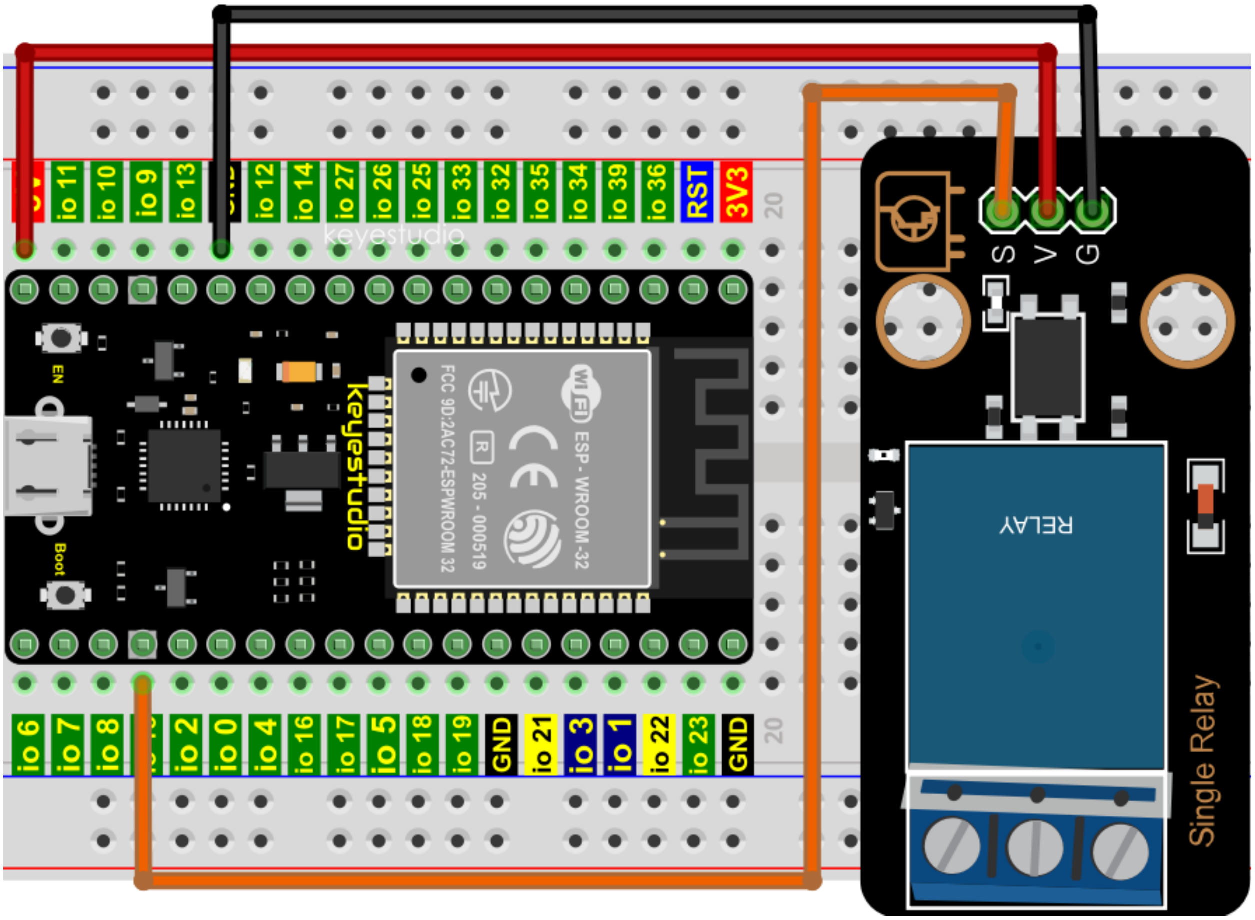
- Input voltage3.3V-5V
- Rated load5A 250VAC (NO/NC) 5A 24VDC (NO/NC)

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

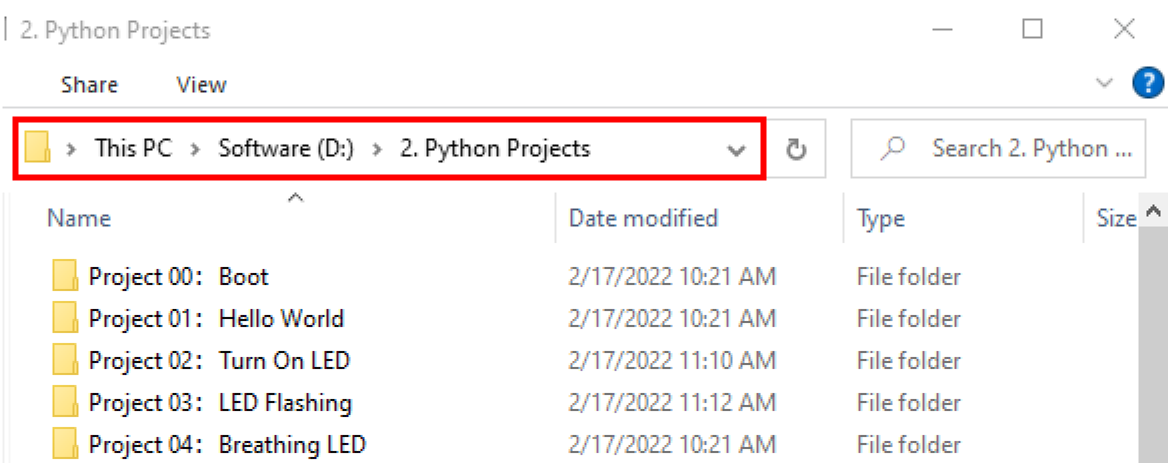


7.22.4 4. Wiring Diagram

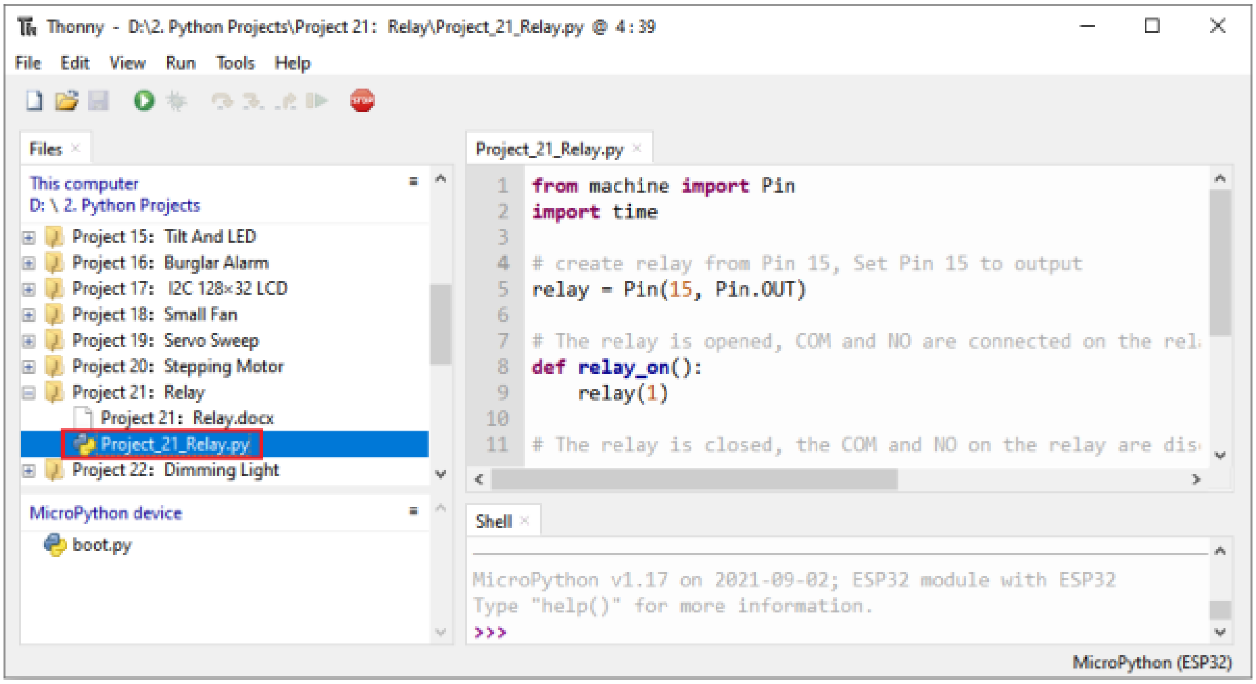


7.22.5 5.Project code

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



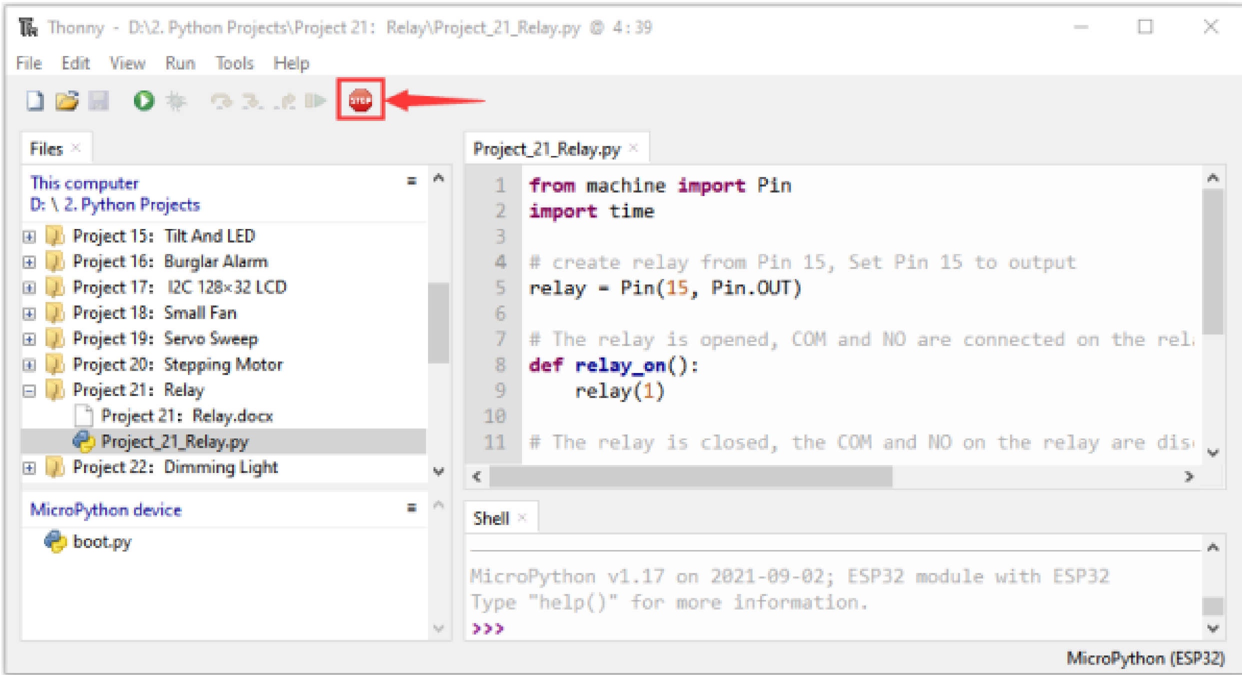
Open “Thonny”, click “This computer” → “D:” → “2. Python Projects” → “Project 21Relay”, and then click “Project_21_Relay.py”.




```
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␣
↪disconnected.
def relay_on():
    relay(1)

## The relay is closed, the COM and NO on the relay are disconnected, and the COM and NC␣
↪are connected.
def relay_off():
    relay(0)

## 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)
```

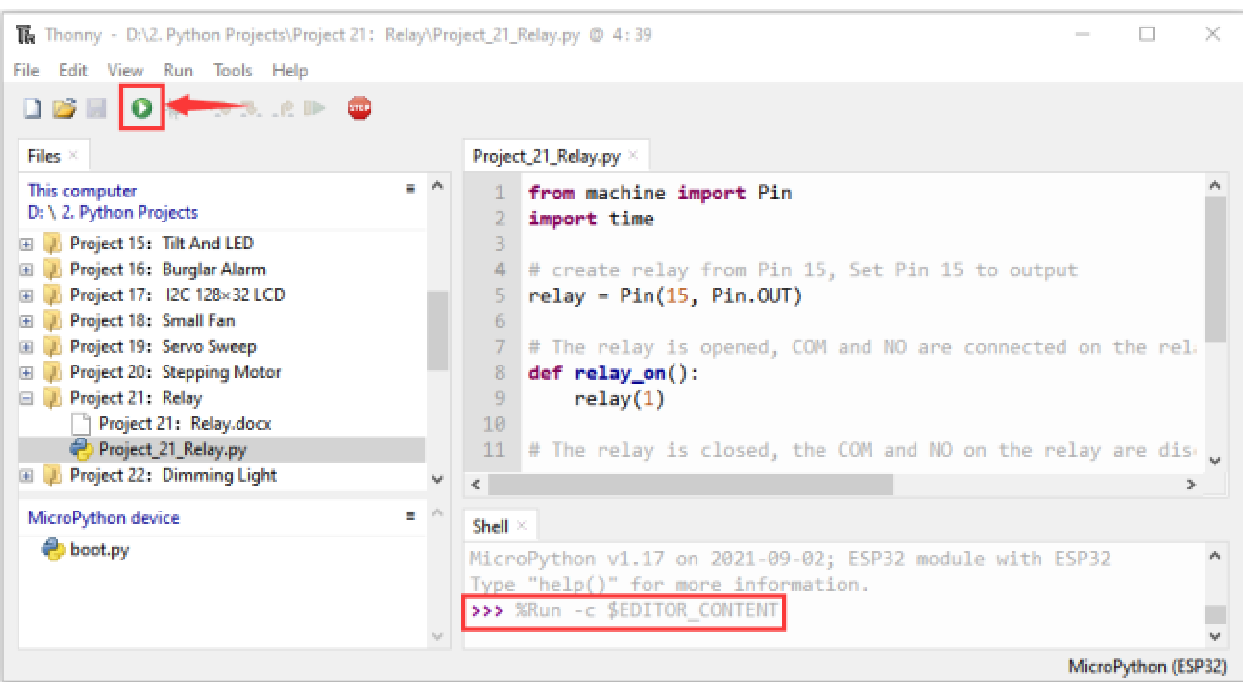
7.22.6 6.Project result

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.



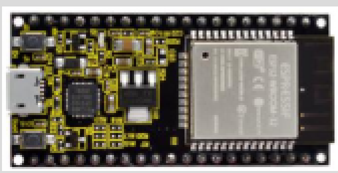
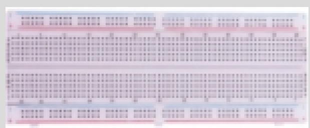
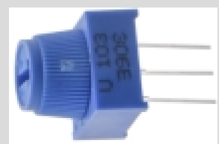
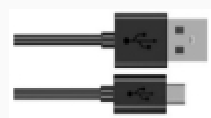
7.23 Project 22Dimming Light

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

			
ESP32*1	Breadboard*1	Potentiometer*1	Red LED*1
			
220Resistor*1	Jumper Wires	USB Cable*1	

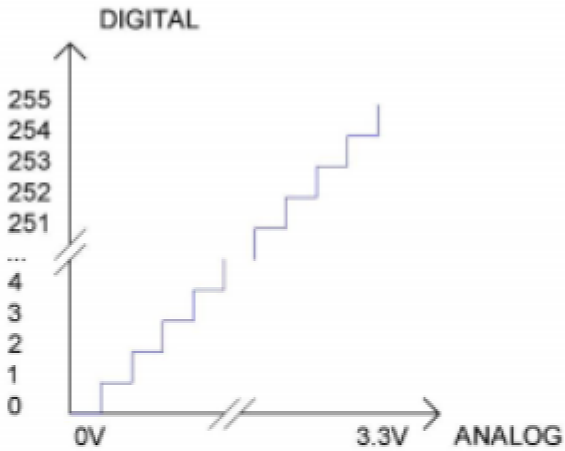
7.23.3 3.Component knowledge

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 ~ 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

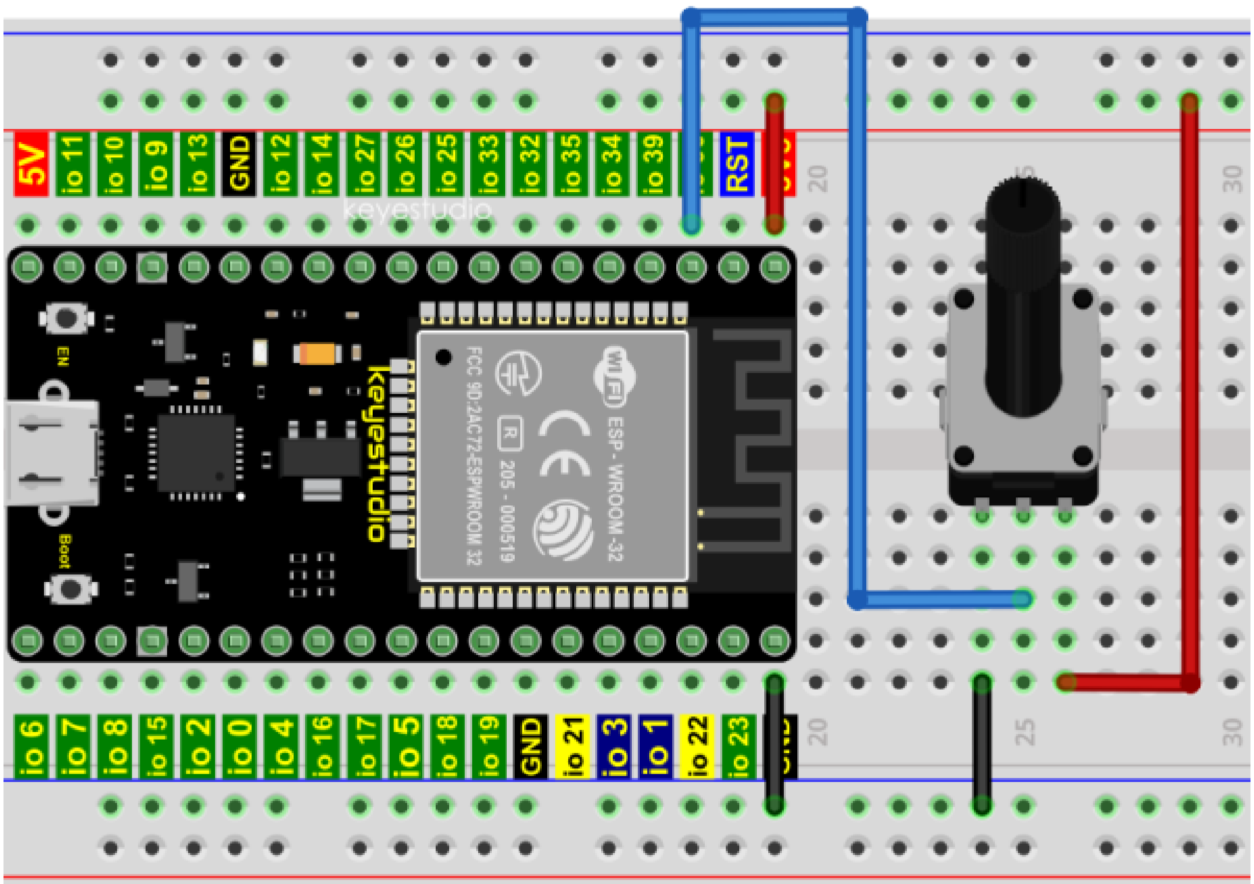
DAC on ESP32

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

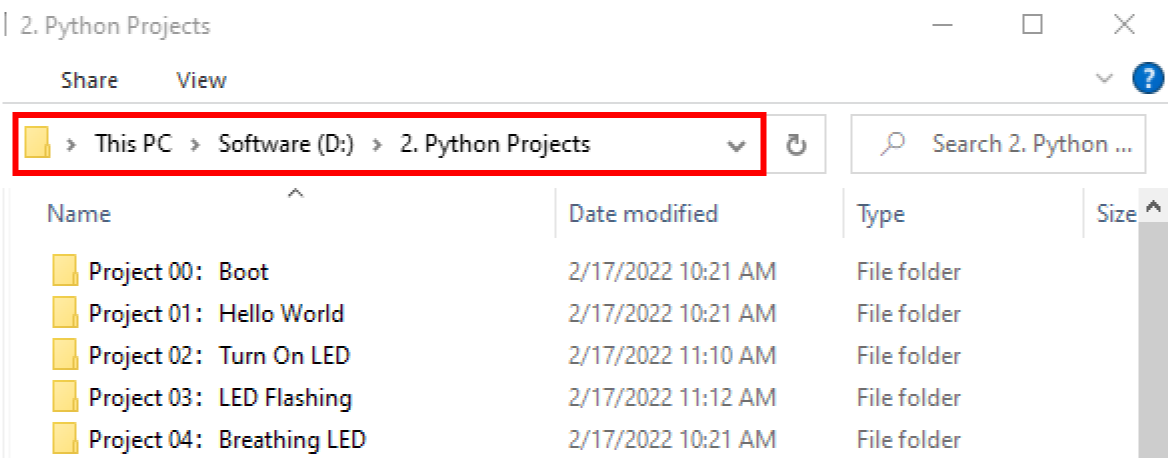
Simulate pin number	GPIO number
DAC1	GPIO25
DAC2	GPIO26

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

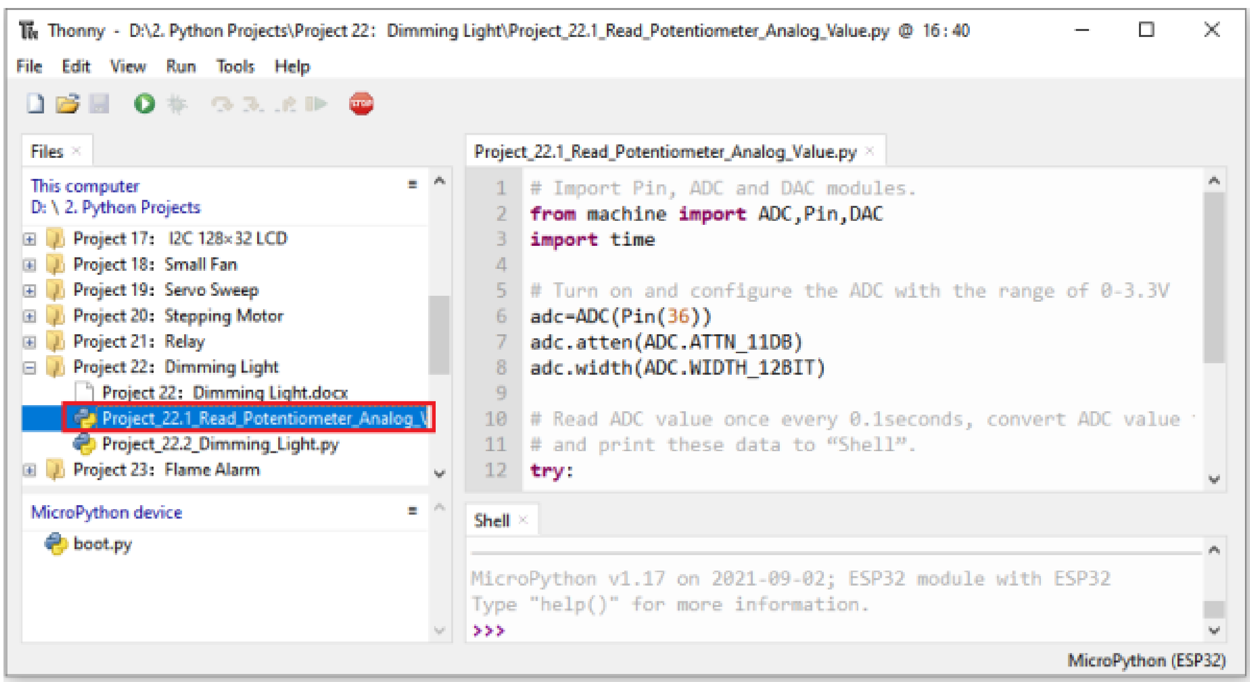
We connect the potentiometer to the analog IO port of ESP32 to read the ADC value, DAC value and voltage value of the potentiometer, please refer to the wiring diagram below



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



Open"Thonny"click"This computer"→"D:"→"2. Python Projects"→"Project 22Dimming Light"and then double left-click "Project_22.1_Read_Potentiometer_Analog_Value.py".



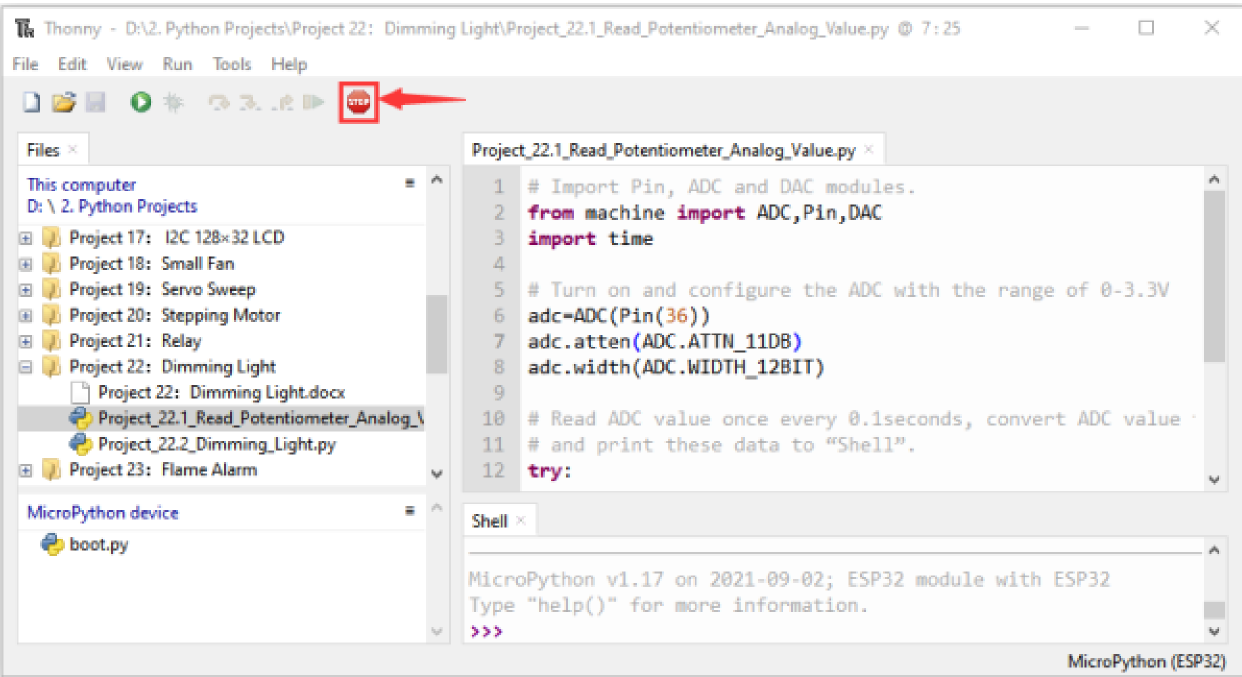

```
## Import Pin, ADC and DAC modules.
from machine import ADC,Pin,DAC
import time

## 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()
```

(continues on next page)

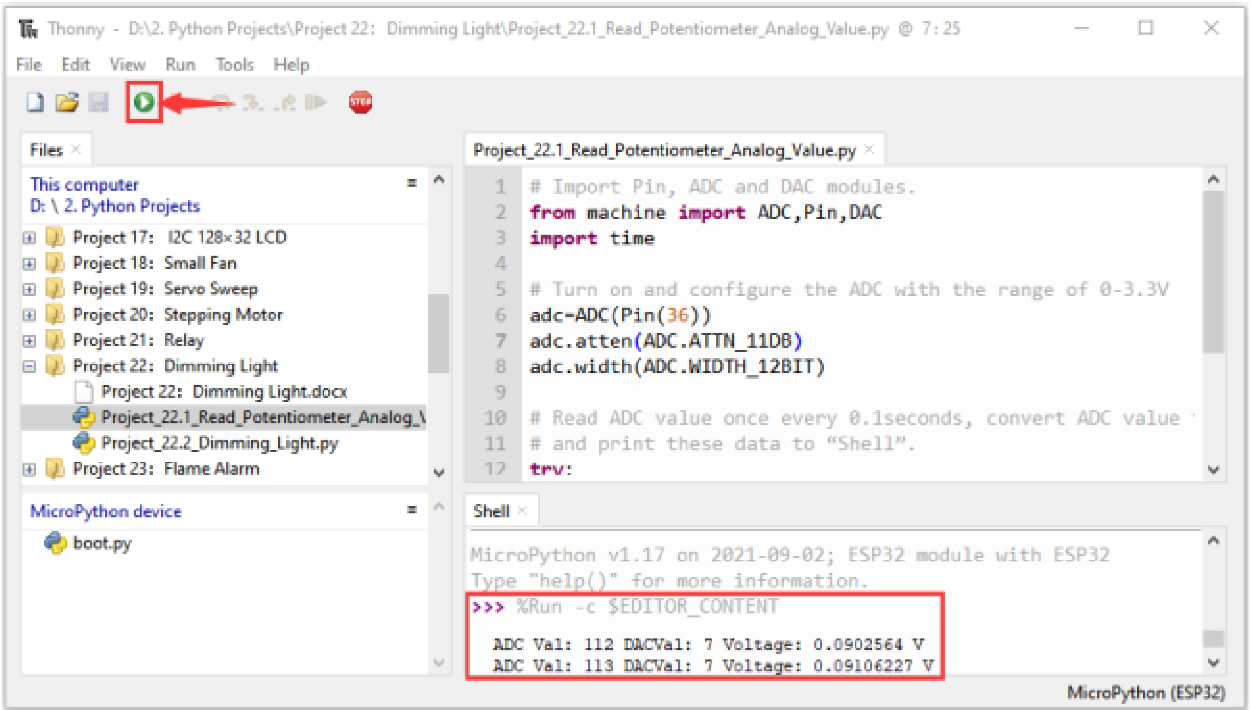
(continued from previous page)

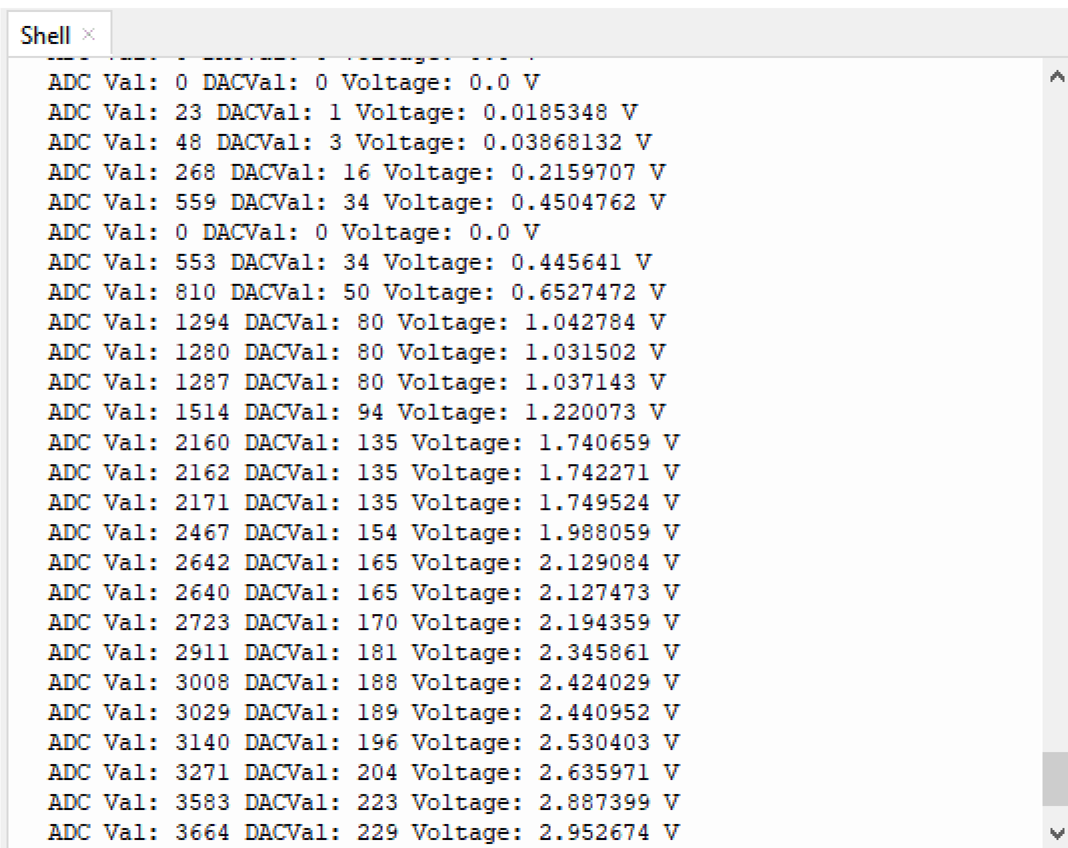
```
        dacVal=adcVal//16
        voltage = adcVal / 4095.0 * 3.3
        print("ADC Val:",adcVal,"DACVal:",dacVal,"Voltage:",voltage,"V")
        time.sleep(0.1)
except:
    pass
```

Make sure the ESP32 has been connected to the computer, click "Stop/Restart backend" .



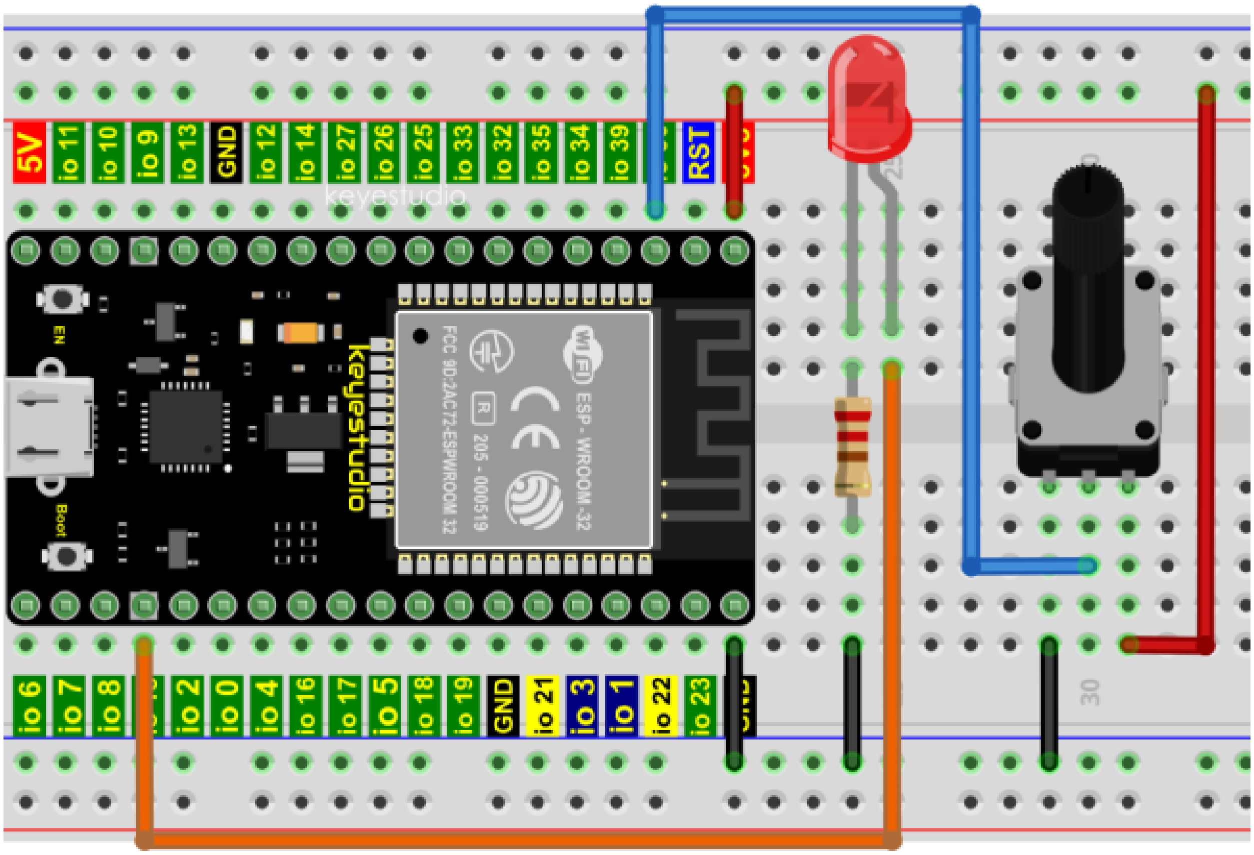
Click "Run current script", the code starts to be executed and you'll see that the "Shell" window of Thonny IDE will print the ADC value, DAC value and voltage value of the potentiometer, turn the potentiometer handle, the ADC value and voltage value will change. Press "Ctrl+C" or click "Stop/Restart backend" to exit the program.


Thonny - D:\2. Python Projects\Project 22: Dimming Light\Project_22.1_Read_Potentiometer_Analog_Value.py @ 7 : 25
File Edit View Run Tools Help
Files
This computer
D: \ 2. Python Projects
Project 17: I2C 128×32 LCD
Project 18: Small Fan
Project 19: Servo Sweep
Project 20: Stepping Motor
Project 21: Relay
Project 22: Dimming Light
Project 22: Dimming Light.docx
Project_22.1_Read_Potentiometer_Analog_\
Project_22.2_Dimming_Light.py
Project 23: Flame Alarm
MicroPython device
boot.py
Project_22.1_Read_Potentiometer_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
# and print these data to "Shell".
try:
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
MicroPython (ESP32)


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

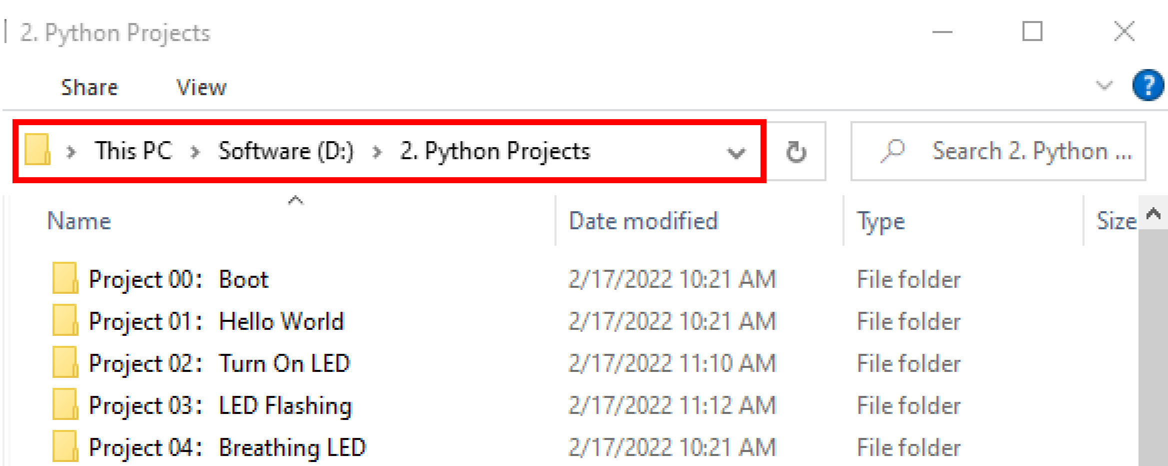
7.23.5 5.Wiring diagram of the dimming lamp

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:

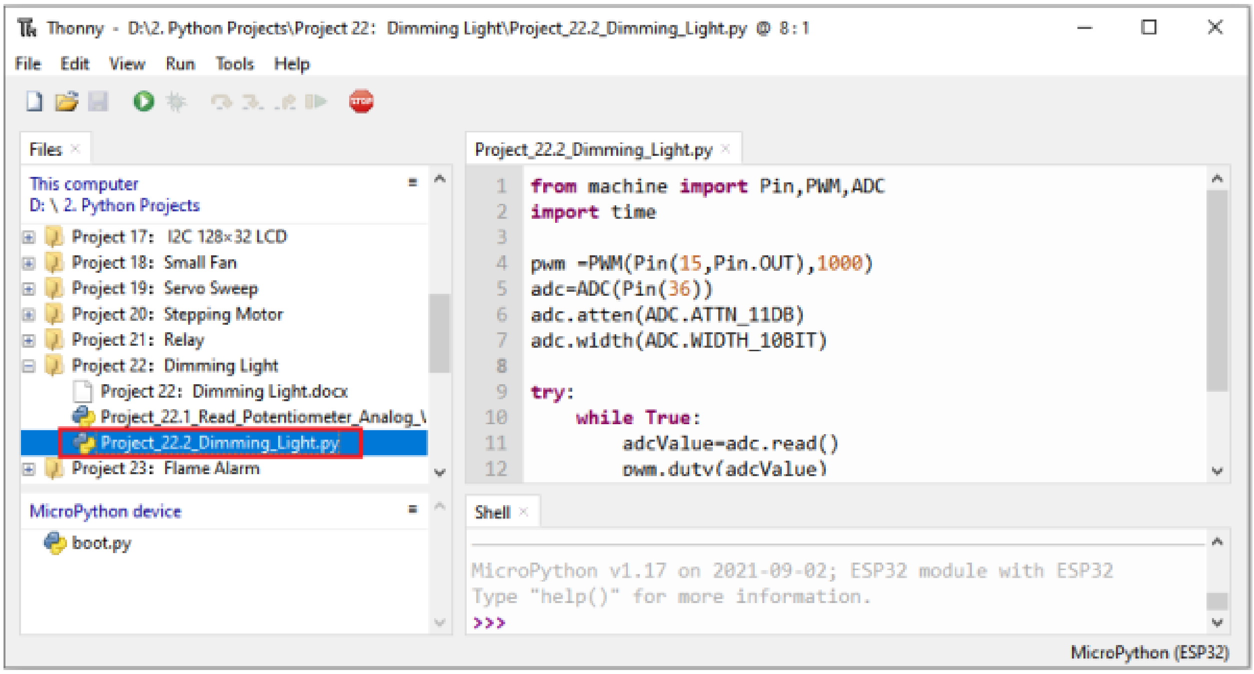


7.23.6 6.Project code

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



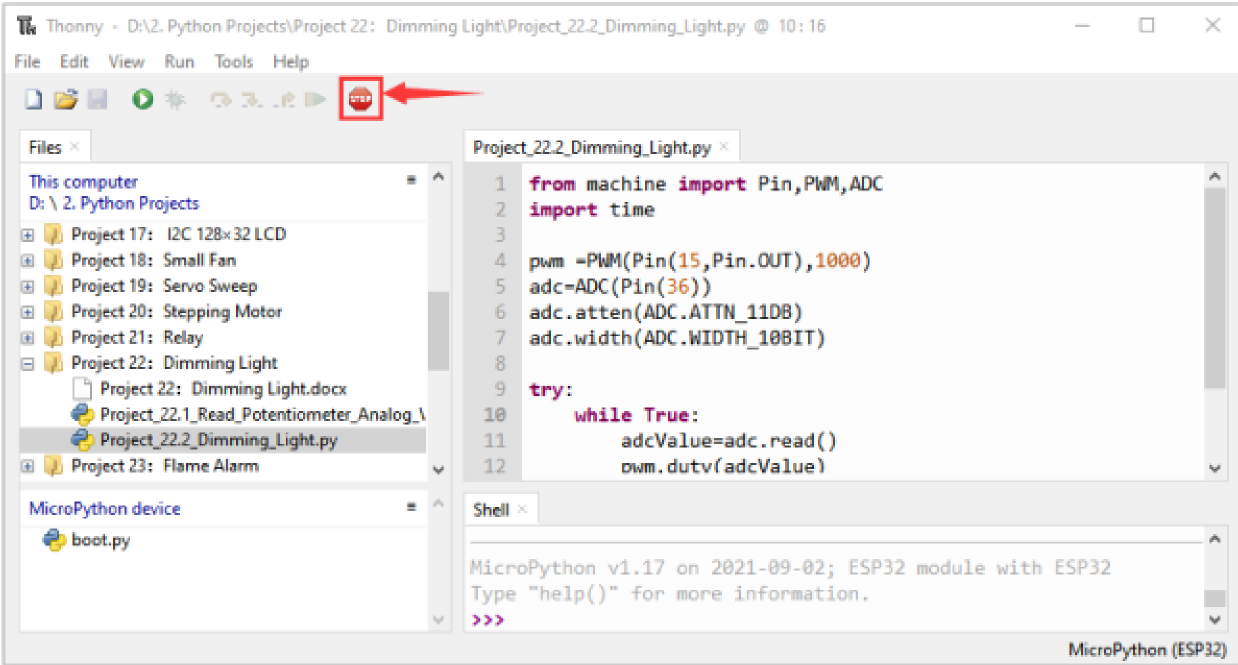
Open“Thonny”click“This computer”→“D:”→“2. Python Projects”→“Project 22Dimming Light”and then click“Project_22.2_Dimming_Light.py”.



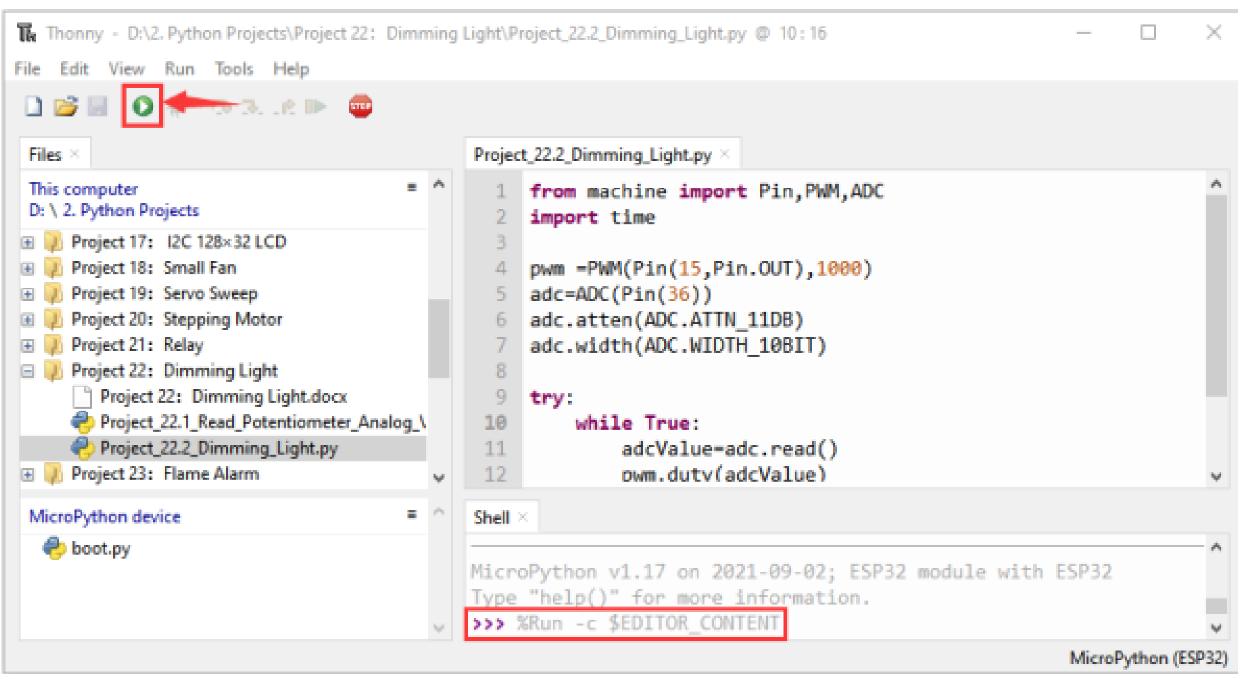
```
from machine import Pin,PWM,ADC
import time

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()
```

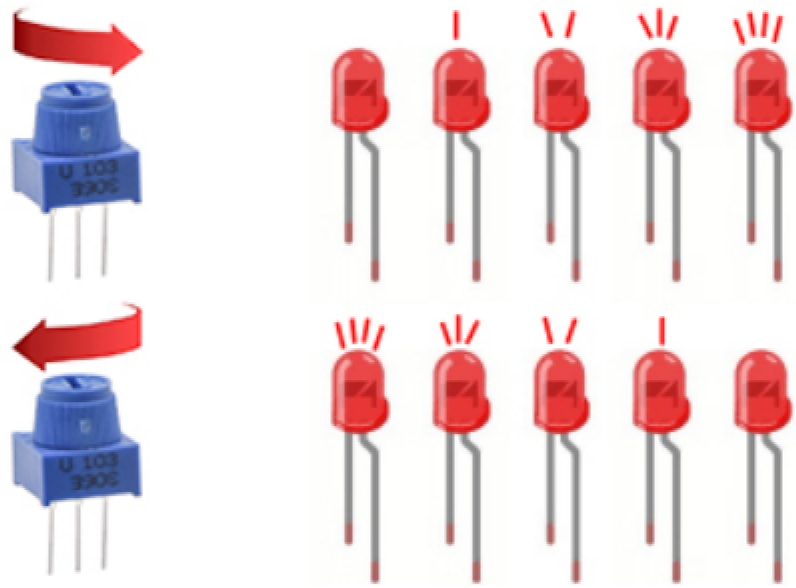
7.23.7 7.Project result

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 turn the potentiometer handle and the brightness of the LED will change accordingly. Press "Ctrl+C" or click "Stop/Restart backend" to exit the program.



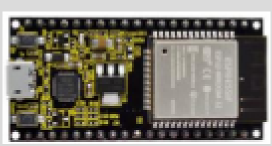
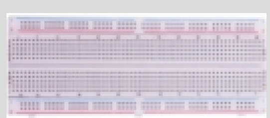
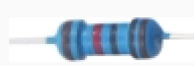
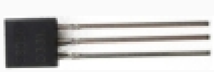
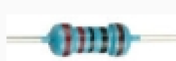
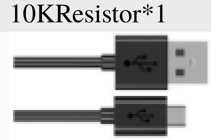


7.24 Project 23Flame Alarm

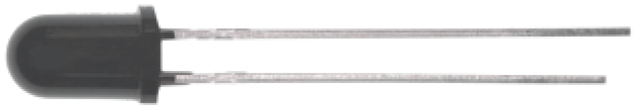
7.24.1 1.Introduction

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.

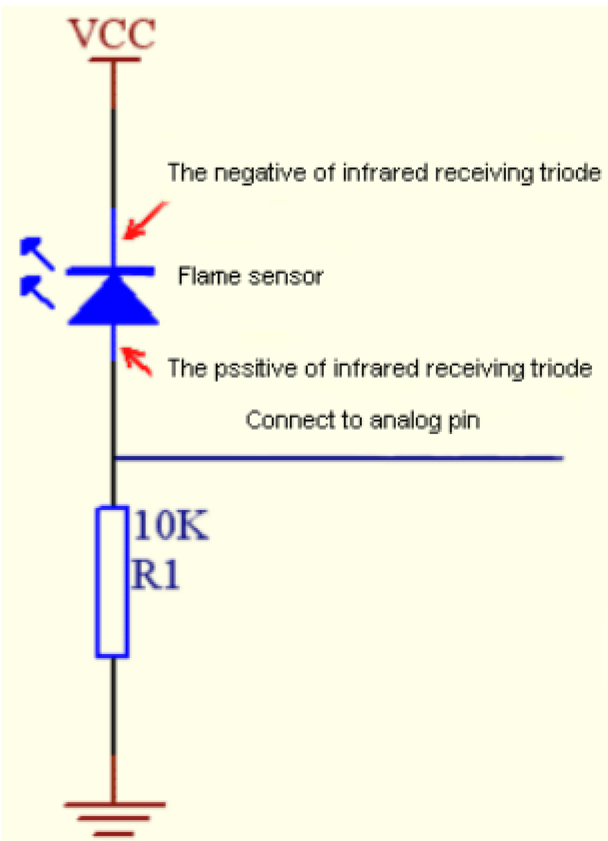
7.24.2 2.Components

			
ESP32*1	Breadboard*1	Red LED*1	Active Buzzer*1
			
Flame Sensor*1	220Resistor*1	10KResistor*1	Jumper Wires
			
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

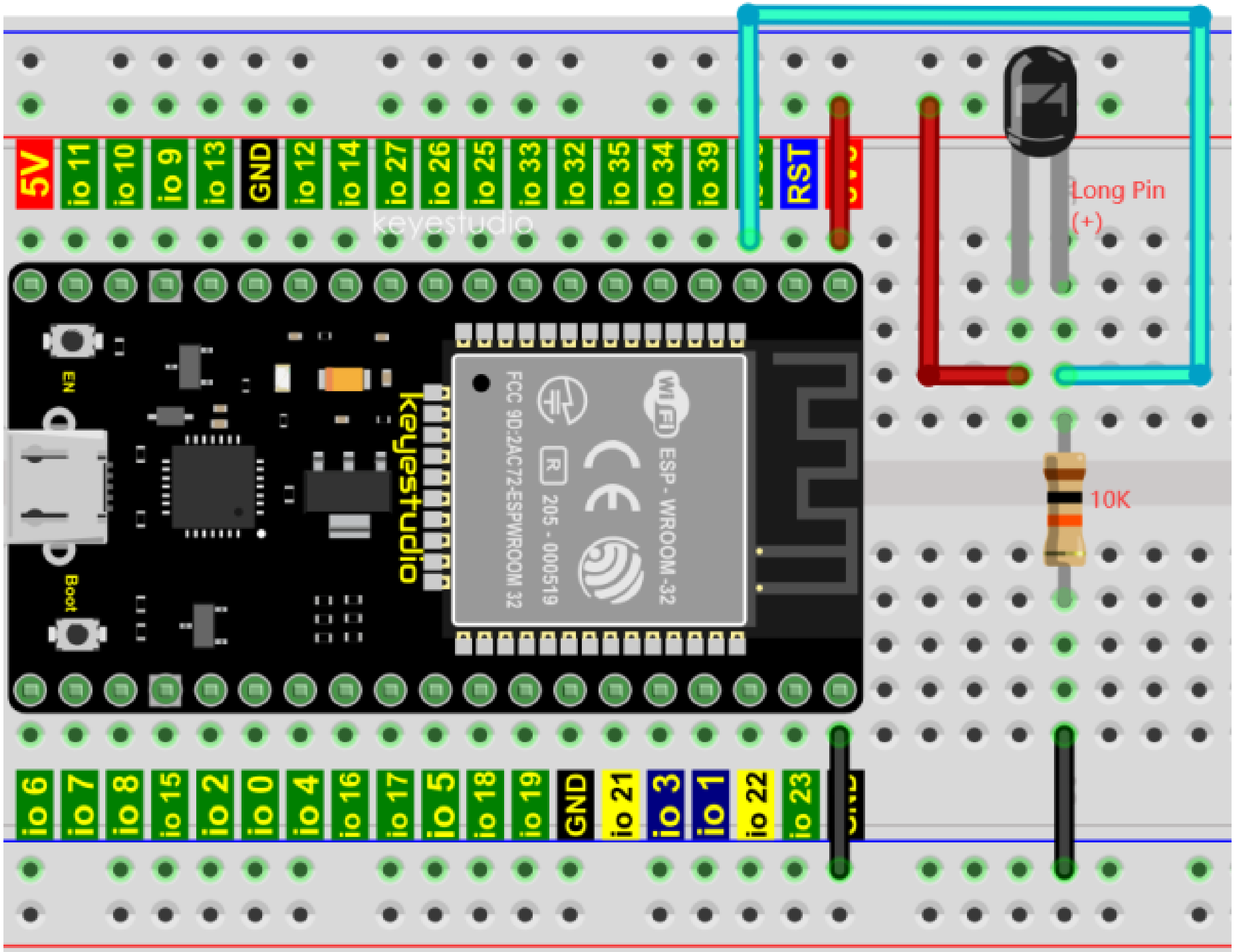


Note:

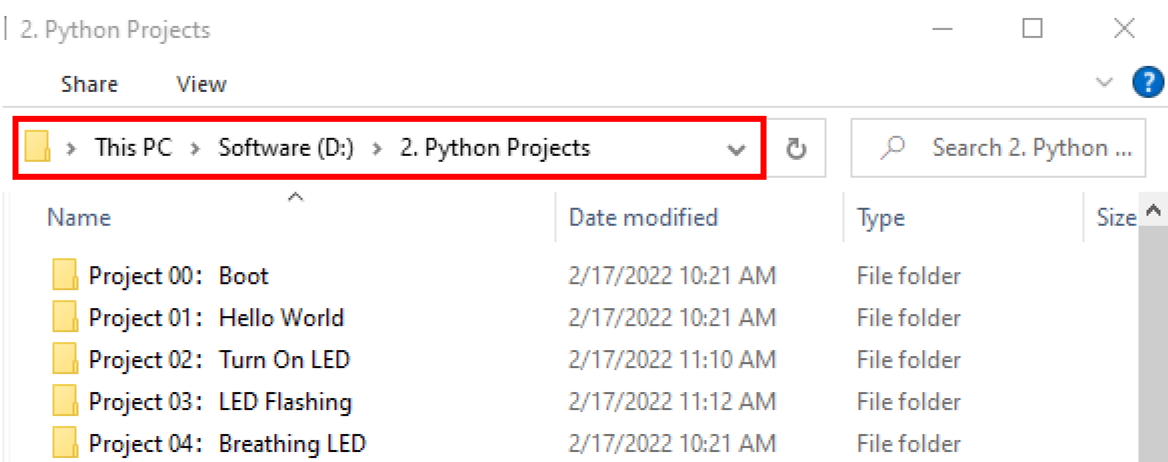
Since vulnerable to radio frequency radiation and temperature changes, the flame sensor should be kept away from heat sources like radiators, heaters and air conditioners, as well as direct irradiation of sunlight, headlights and incandescent light.

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

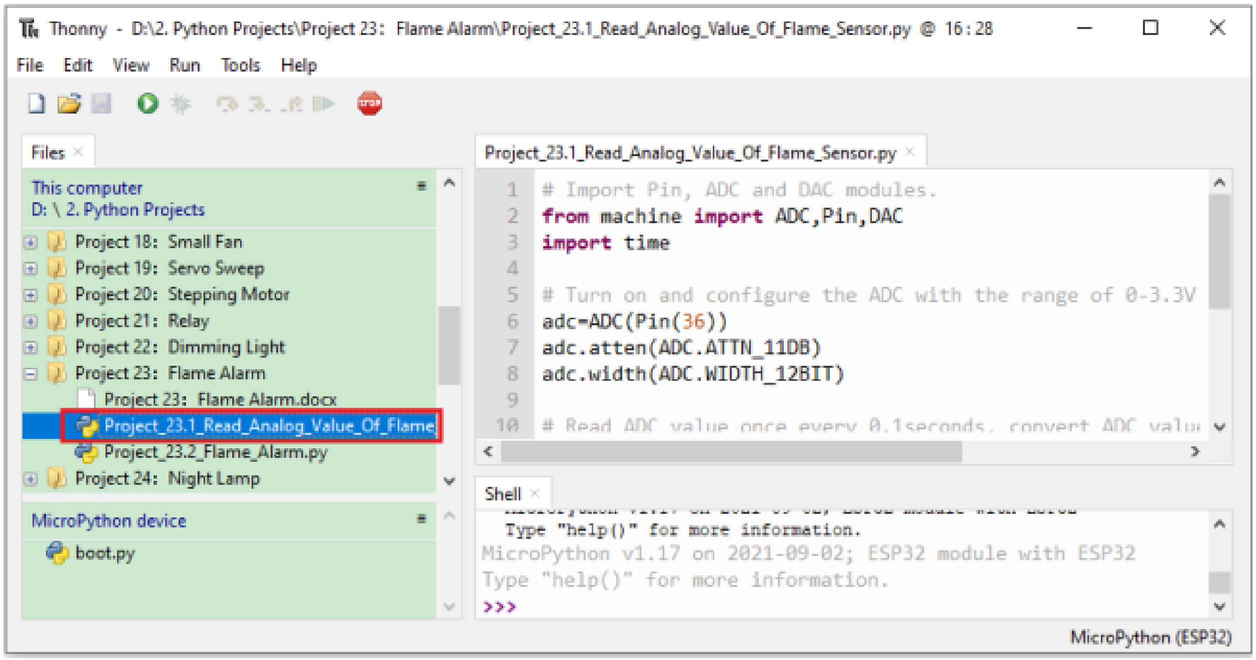
We first use a simple code to read the ADC value, DAC value and voltage value of the flame sensor and print them out. Please refer to the wiring diagram below



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

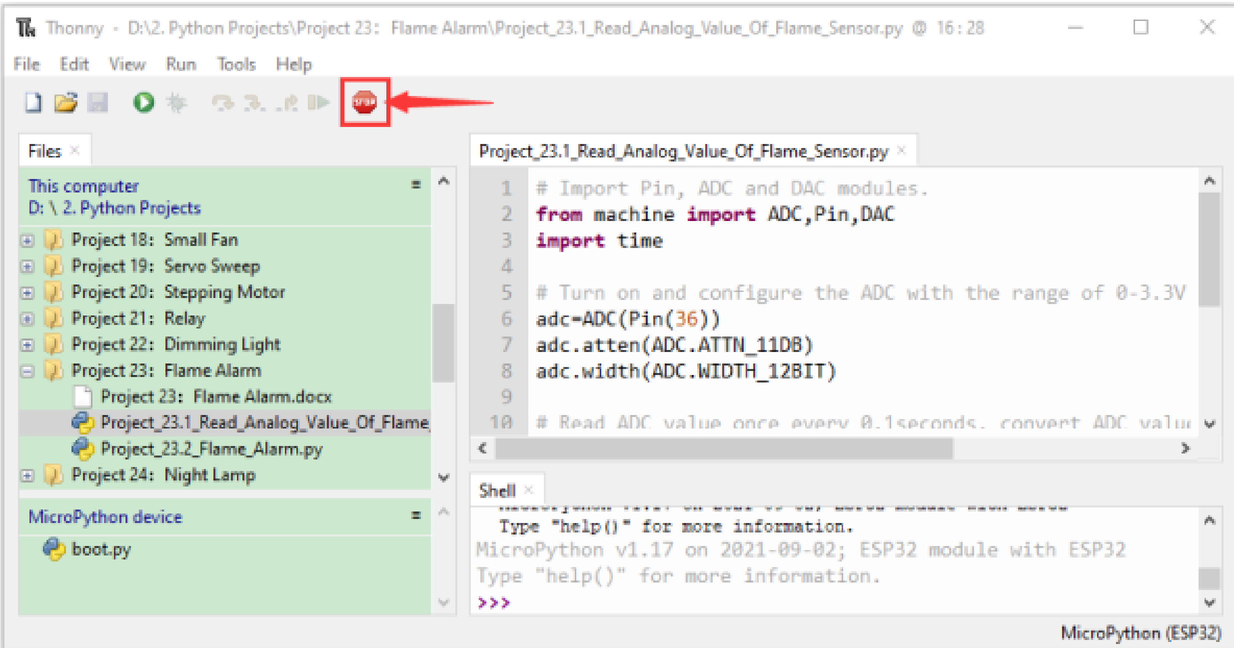


Open"Thonny", click "This computer"→"D:"→"2. Python Projects"→"Project 23Flame Alarm"and then double left-click "Project_23.1_Read_Analog_Value_Of_Flame_Sensor.py".



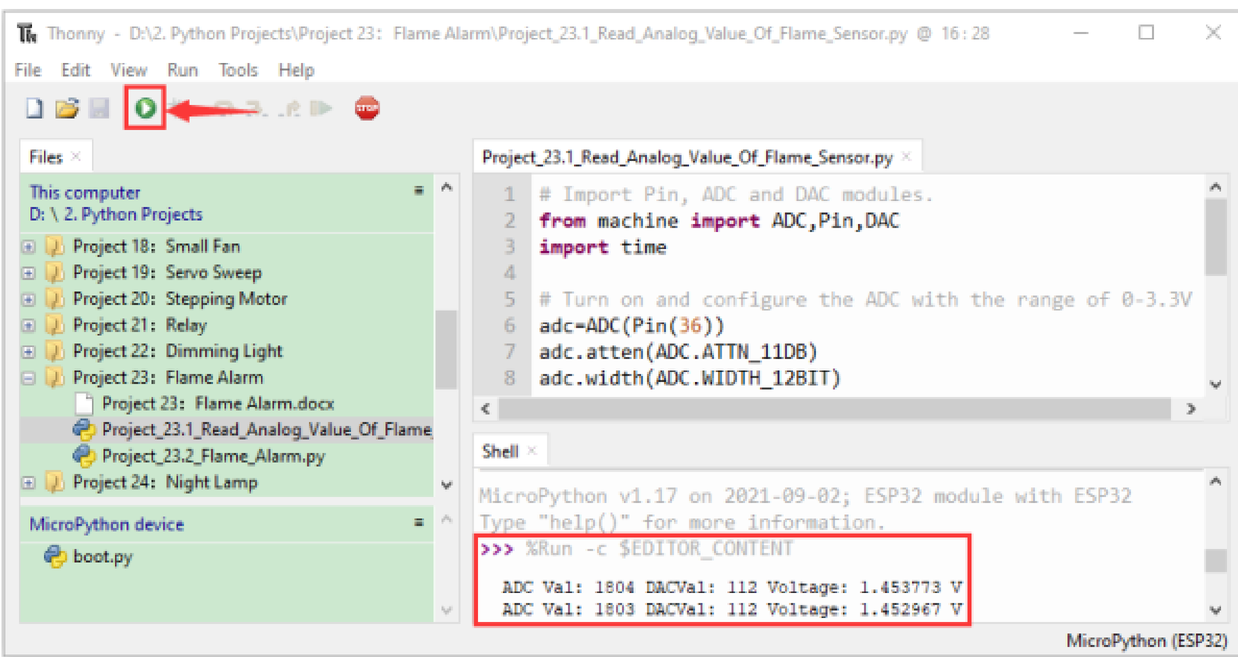

```python
## Import Pin, ADC and DAC modules.
from machine import ADC,Pin,DAC
import time

## 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 "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.

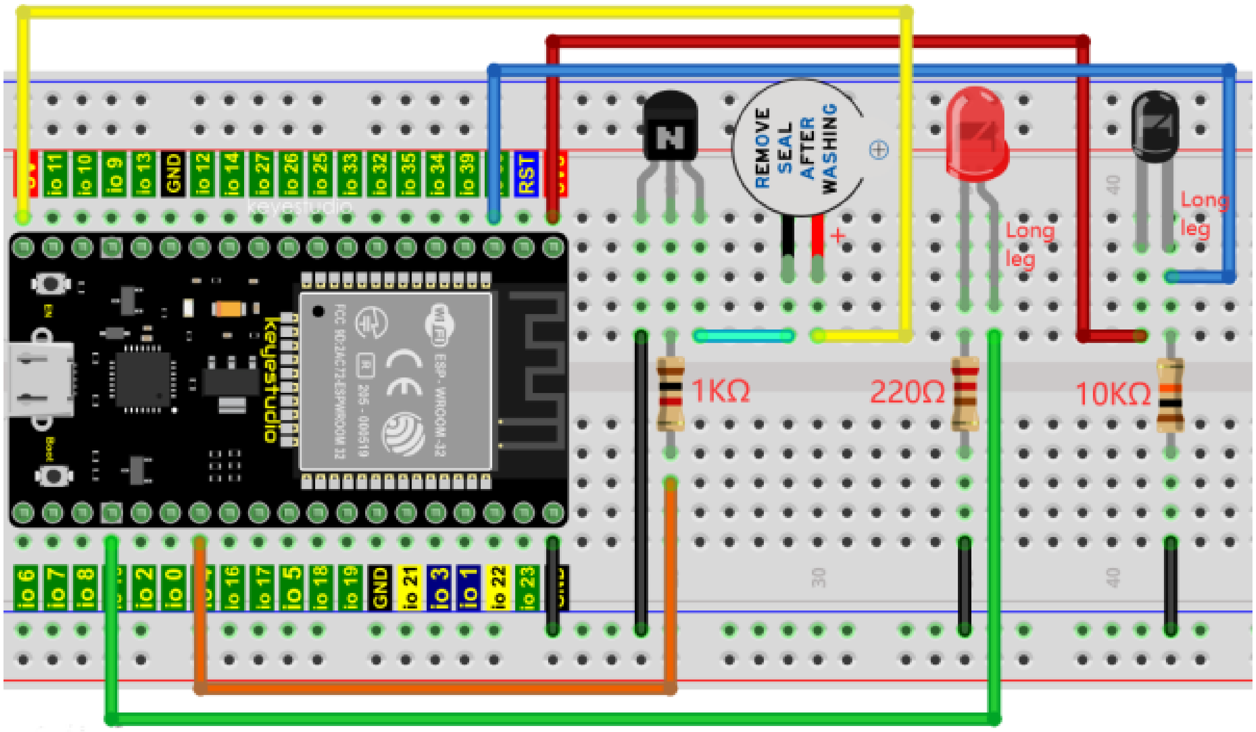
When the flame is close to the flame sensor, the ADC value, DAC value and voltage value increase; Conversely, the ADC value, DAC value and voltage value decrease. Press "Ctrl+C" or click "Stop/Restart backend" to exit the program.



```
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
```

7.24.5 5.Wiring diagram of the flame alarm

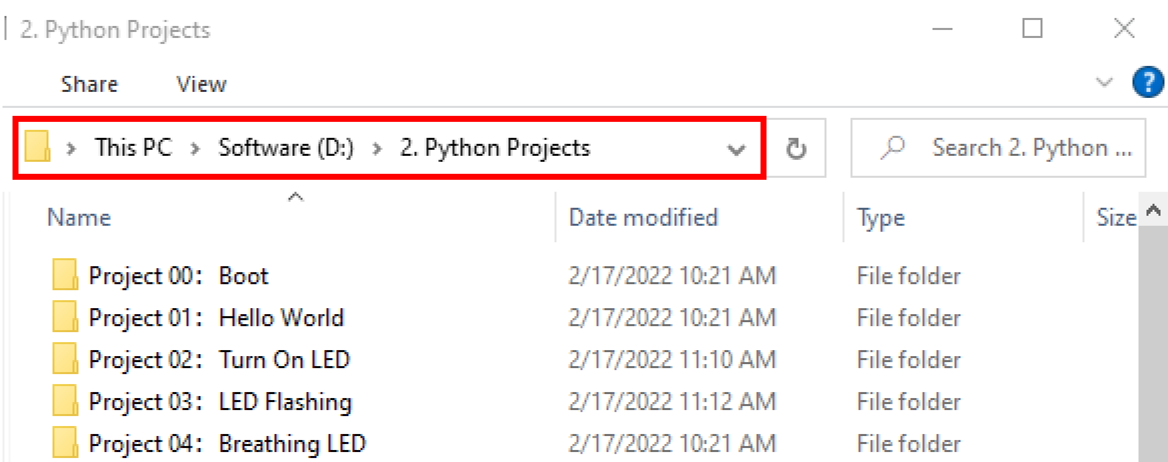
Next, we will use a flame sensor, a buzzer, and a LED to make an interesting project, that is flame alarm. When flame is detected, the LED flashes and the buzzer alarms.



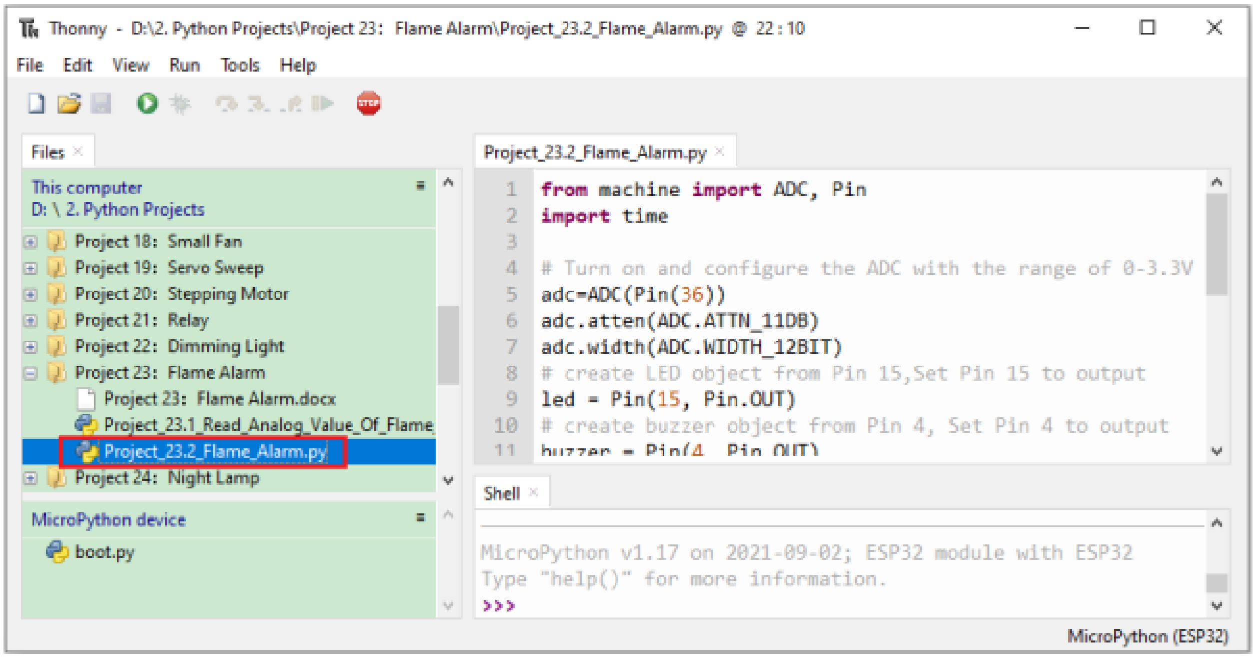
7.24.6 6.Project code

Note: `if 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



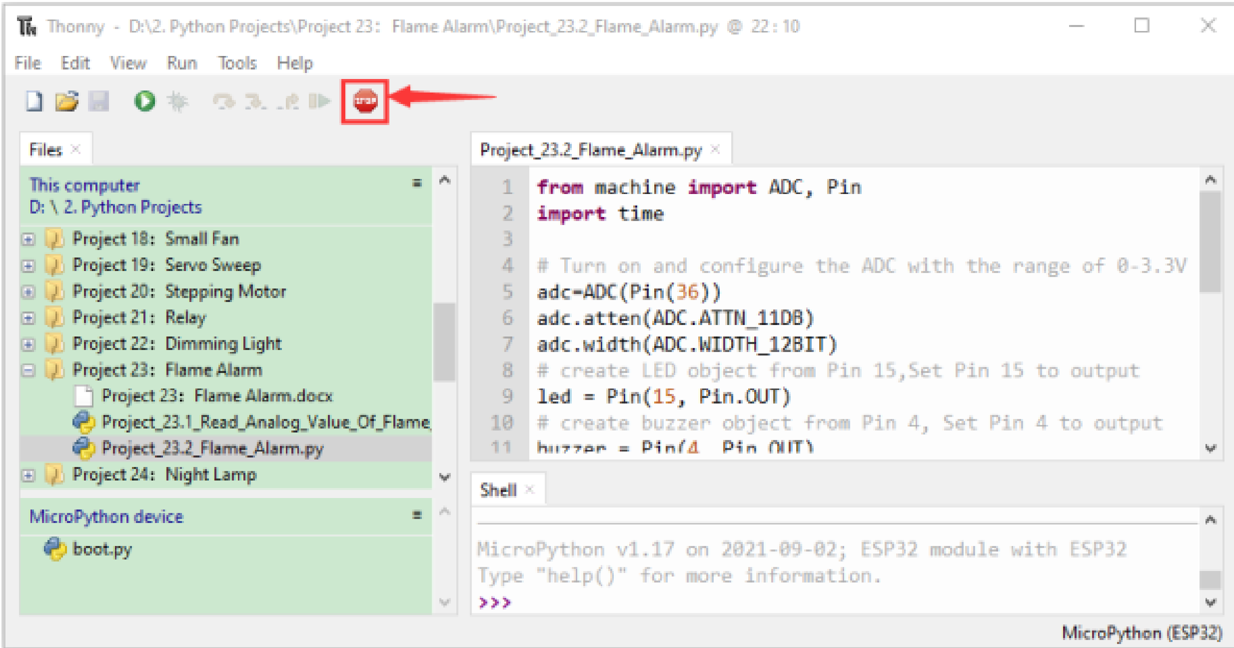
Open“Thonny”click“This computer”→“D:”→“2. Python Projects”→“Project 23Flame Alarm”, and then double left-click “Project_23.2_Flame_Alarm.py”.




```
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:
        buzzer.value(1)    # Set buzzer turn on
        led.value(1)    # Set led turn on
        time.sleep(0.5) # Sleep 0.5s
        buzzer.value(0)
        led.value(0)    # Set led turn off
        time.sleep(0.5) # Sleep 0.5s
    else:
        buzzer.value(0)    # Set buzzer turn off
        led.value(0)    # Set led turn off
```

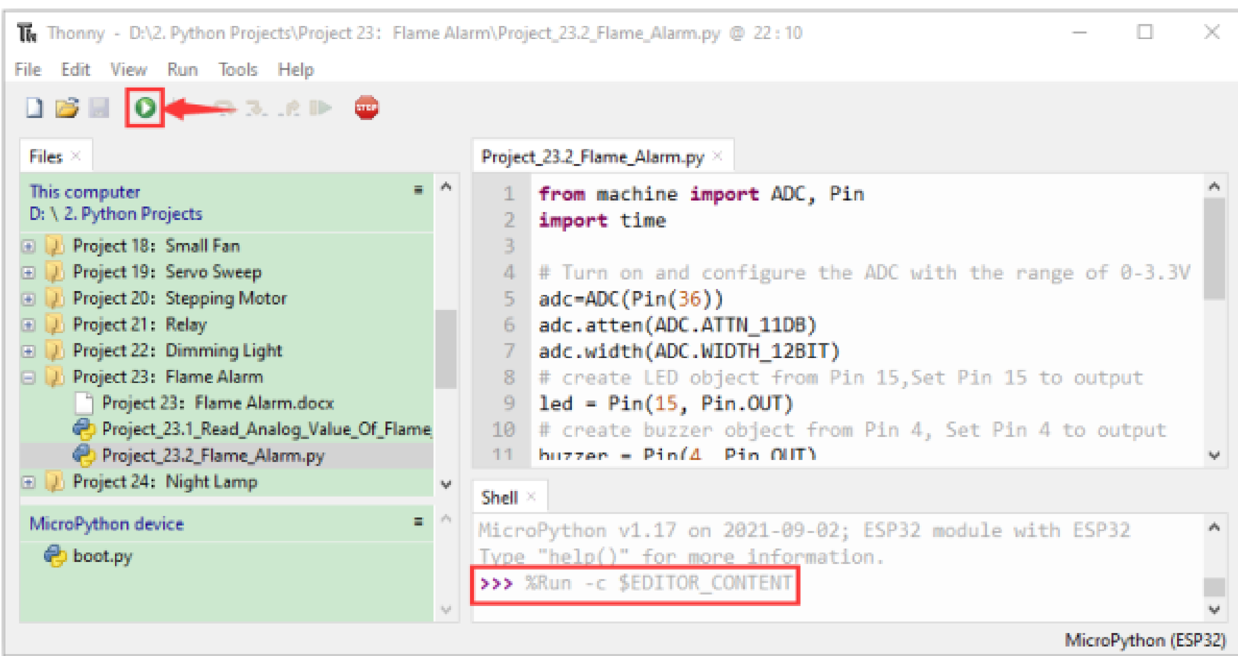
7.24.7 7.Project result

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 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 "Stop/Restart backend" to exit the program.



7.25 Project 24Night Lamp

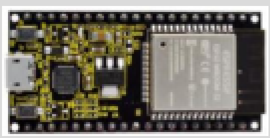
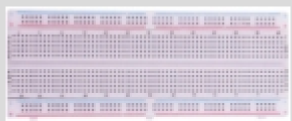
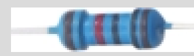
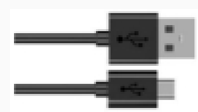
7.25.1 1.Introduction

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

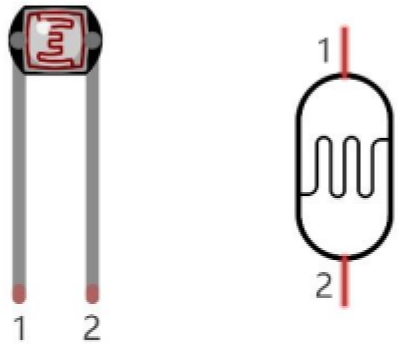
			
ESP32*1	Breadboard*1	Red LED*1	10KResistor*1
			
Photoresistor*1	220Resistor*1	Jumper Wires	USB Cable*1

7.25.3 3.Component knowledge

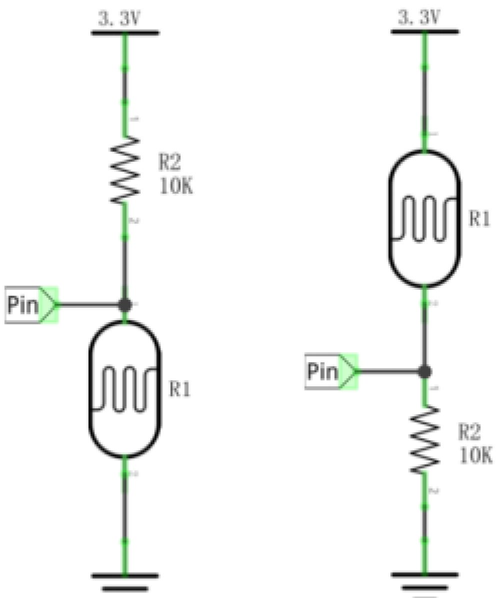
Photoresistor :

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



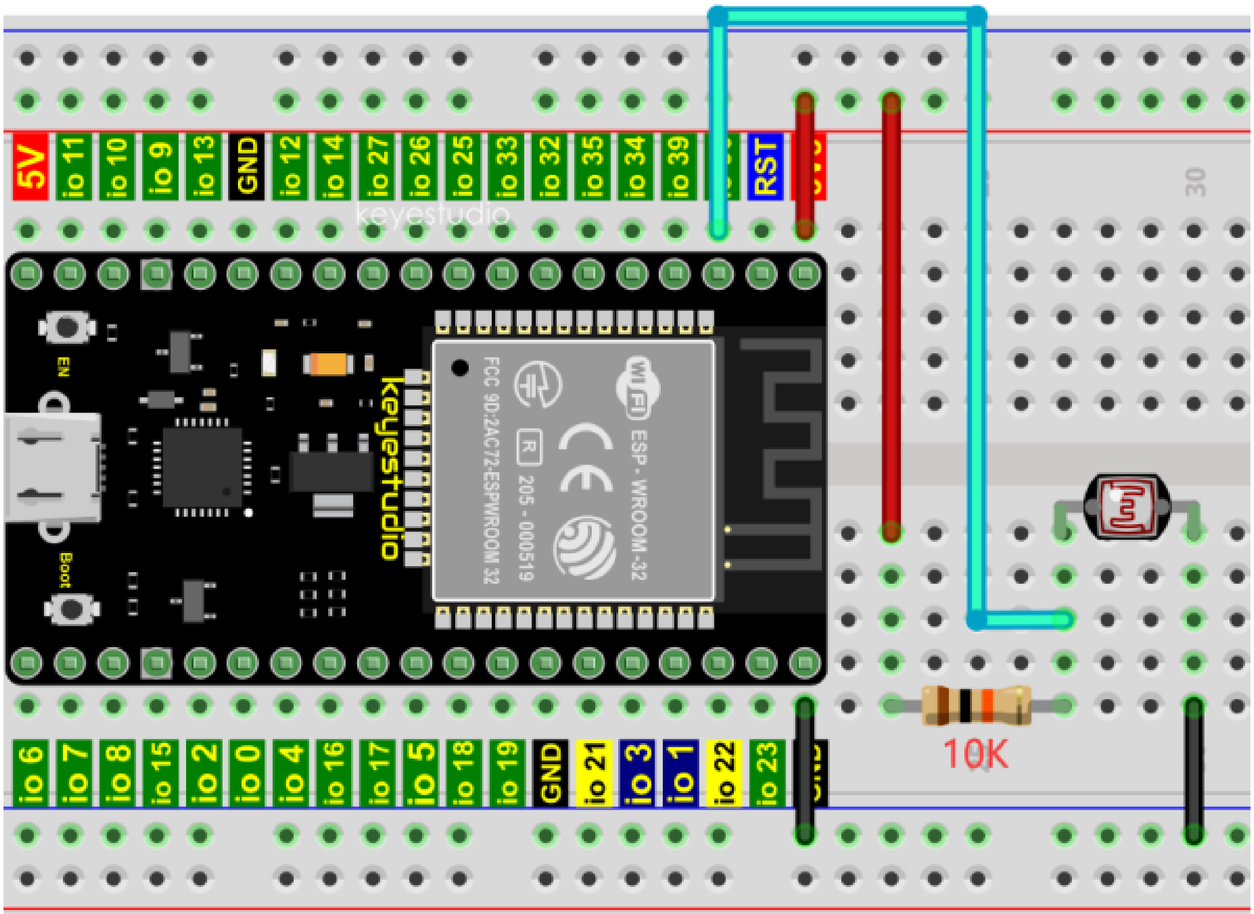
The following circuit is used to detect changes in resistance values of photoresistors



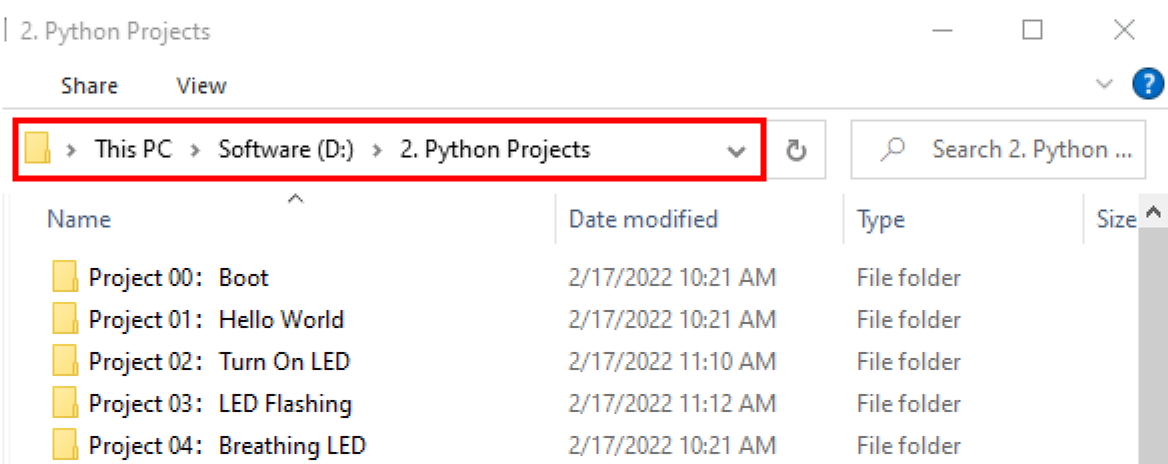
In the circuit above, when the resistance of the photoresistor changes due to the change of light intensity, the voltage between the photoresistor and resistance R2 will also change. Thus, the intensity of light can be obtained by measuring this voltage.

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

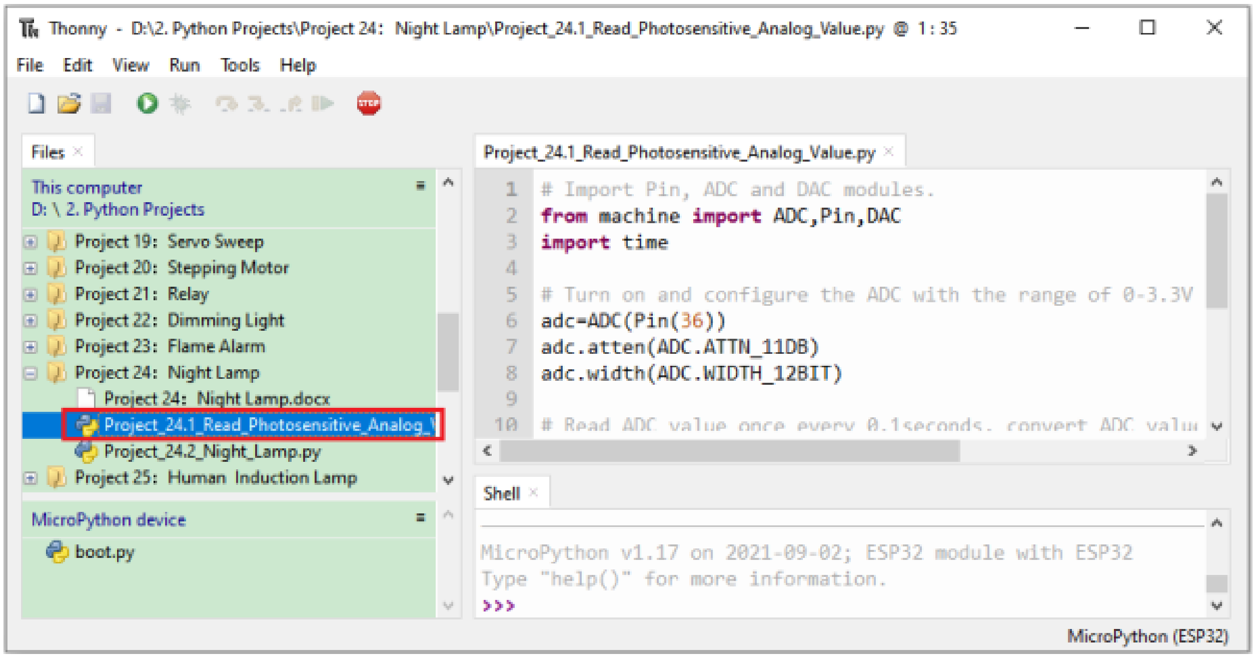
We first use a simple code to read the ADC value, DAC value and voltage value of the photoresistor and print them out. Please refer to the following wiring diagram



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

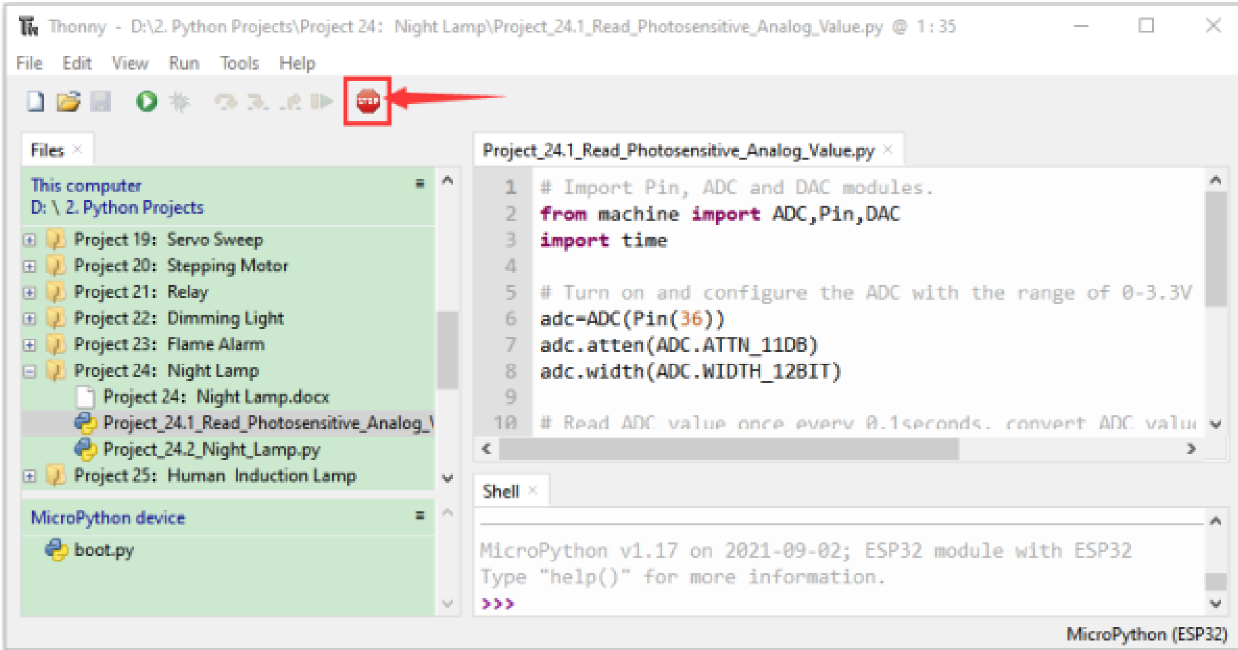


Open“Thonny”click“This computer”→“D:”→“2. Python Projects”→“Project 24Night Lamp”and then double left-click “Project_24.1_Read_Photosensitive_Analog_Value.py”.




```
## 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
```

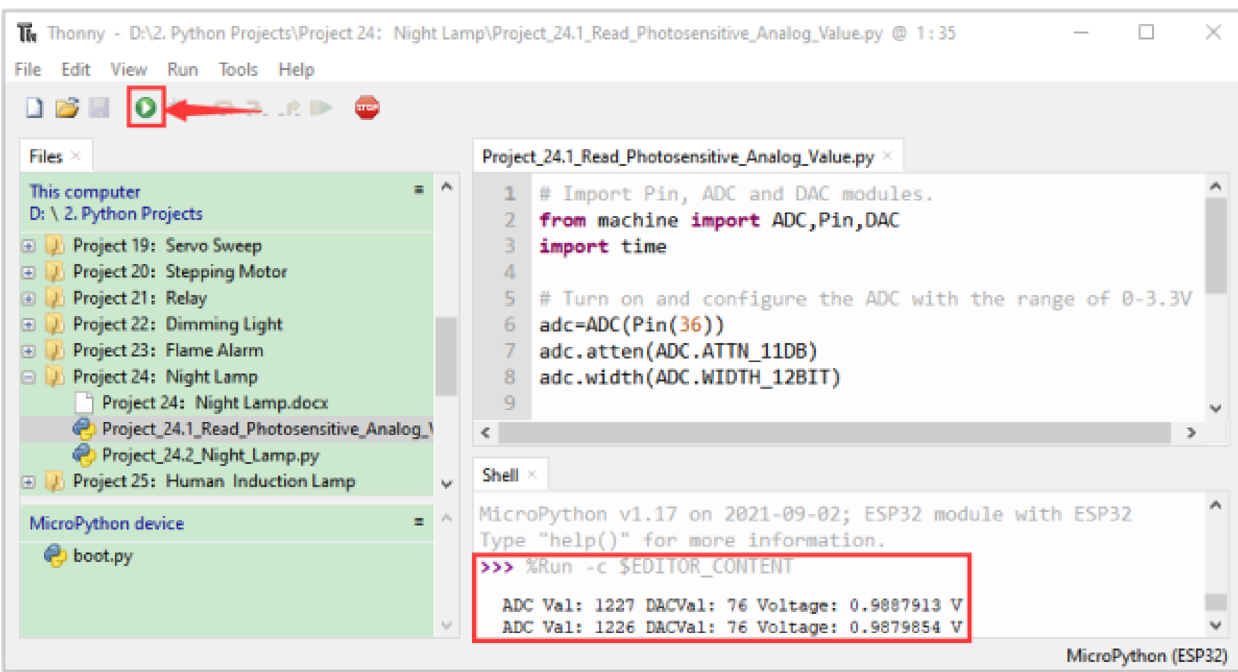
Make sure the ESP32 has been connected to the computer, click “Stop/Restart backend” .



Click “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 photoresistor.

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.

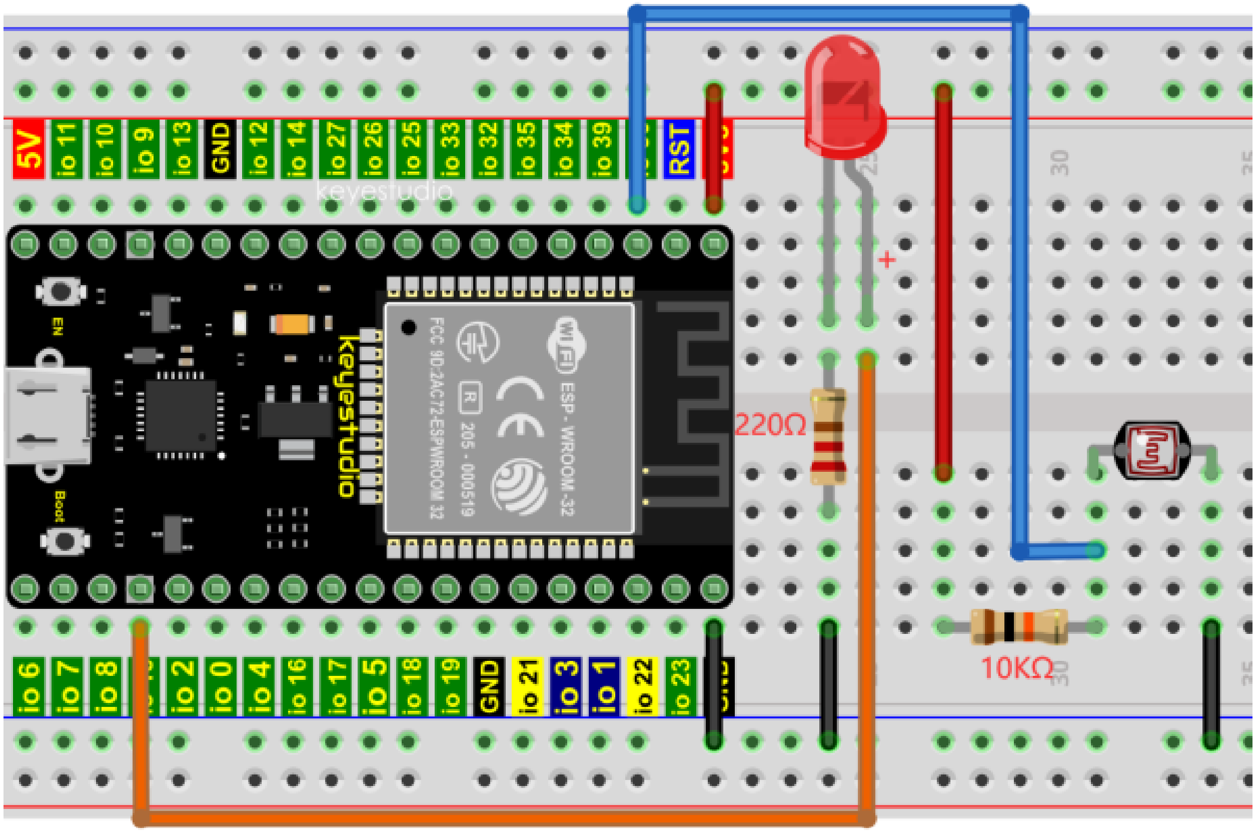
Press“Ctrl+C”or click “Stop/Restart backend” to exit the program.



```
Shell
ADC Val: 1472 DACVal: 92 Voltage: 1.186227 V
ADC Val: 1645 DACVal: 102 Voltage: 1.325641 V
ADC Val: 1847 DACVal: 115 Voltage: 1.488425 V
ADC Val: 2043 DACVal: 127 Voltage: 1.646374 V
ADC Val: 2254 DACVal: 140 Voltage: 1.81641 V
ADC Val: 2442 DACVal: 152 Voltage: 1.967912 V
ADC Val: 2625 DACVal: 164 Voltage: 2.115385 V
ADC Val: 2752 DACVal: 172 Voltage: 2.217729 V
ADC Val: 2832 DACVal: 177 Voltage: 2.282198 V
ADC Val: 2880 DACVal: 180 Voltage: 2.320879 V
ADC Val: 2887 DACVal: 180 Voltage: 2.32652 V
ADC Val: 2873 DACVal: 179 Voltage: 2.315238 V
ADC Val: 2922 DACVal: 182 Voltage: 2.354725 V
ADC Val: 2991 DACVal: 186 Voltage: 2.41033 V
ADC Val: 3051 DACVal: 190 Voltage: 2.458681 V
ADC Val: 3103 DACVal: 193 Voltage: 2.500586 V
ADC Val: 3145 DACVal: 196 Voltage: 2.534432 V
ADC Val: 3163 DACVal: 197 Voltage: 2.548938 V
ADC Val: 3181 DACVal: 198 Voltage: 2.563443 V
ADC Val: 3187 DACVal: 199 Voltage: 2.568278 V
ADC Val: 3214 DACVal: 200 Voltage: 2.590037 V
ADC Val: 3314 DACVal: 207 Voltage: 2.670623 V
```

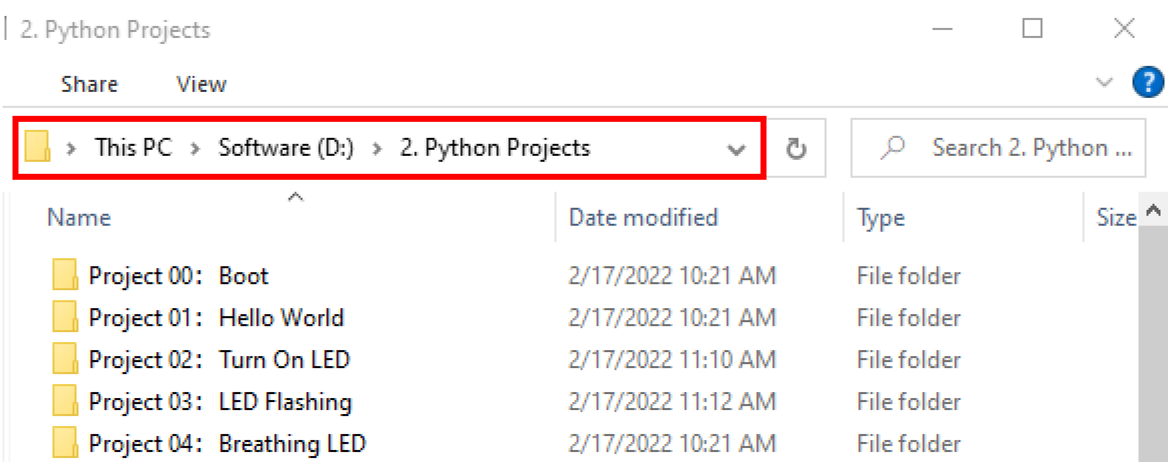
7.25.5 5.Wiring diagram of the light-controlled lamp

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.

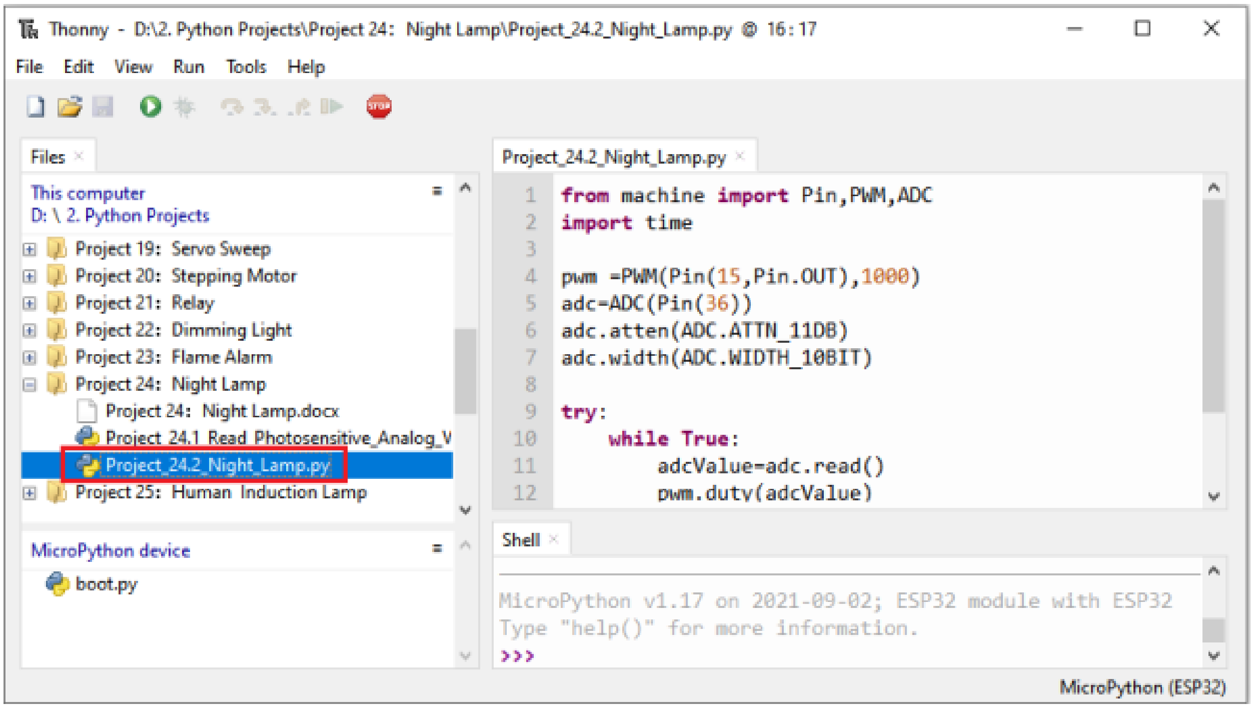


7.25.6 6.Project code

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



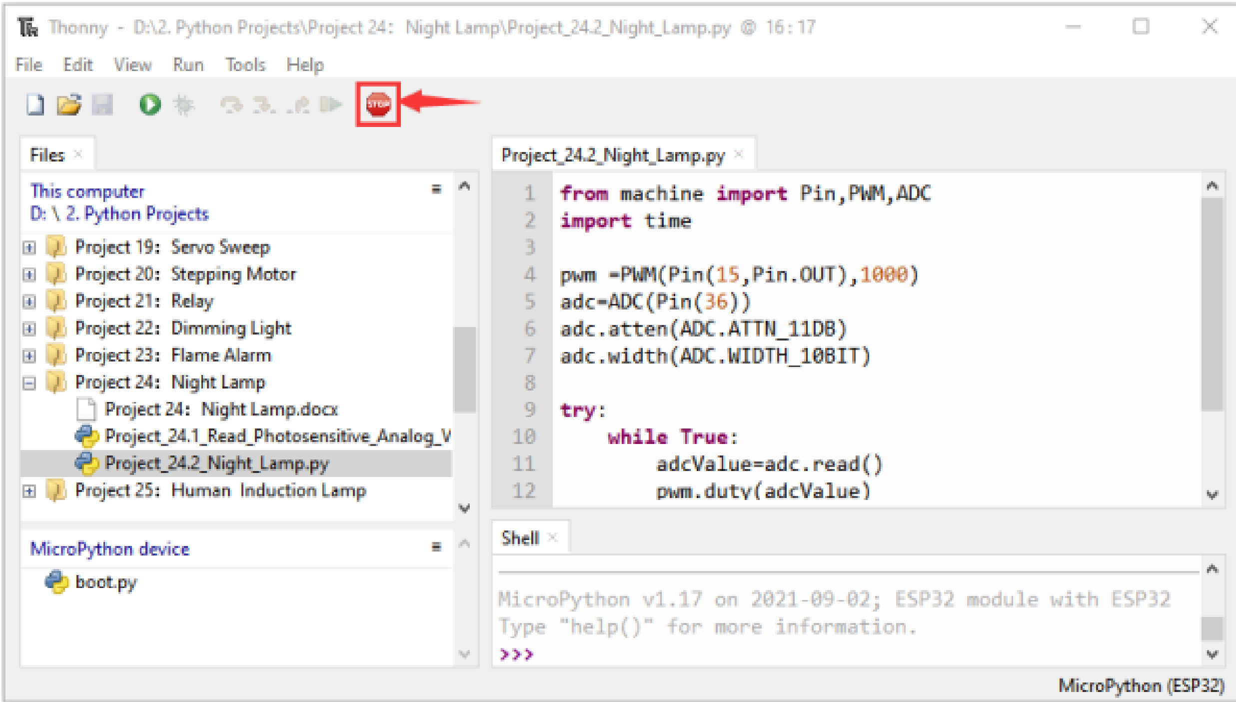
Open“Thonny”click“This computer”→“D:”→“2. Python Projects”→“Project 24Night Lamp”and then double left-click “Project_24.2_Night_Lamp.py”.




```
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()
```

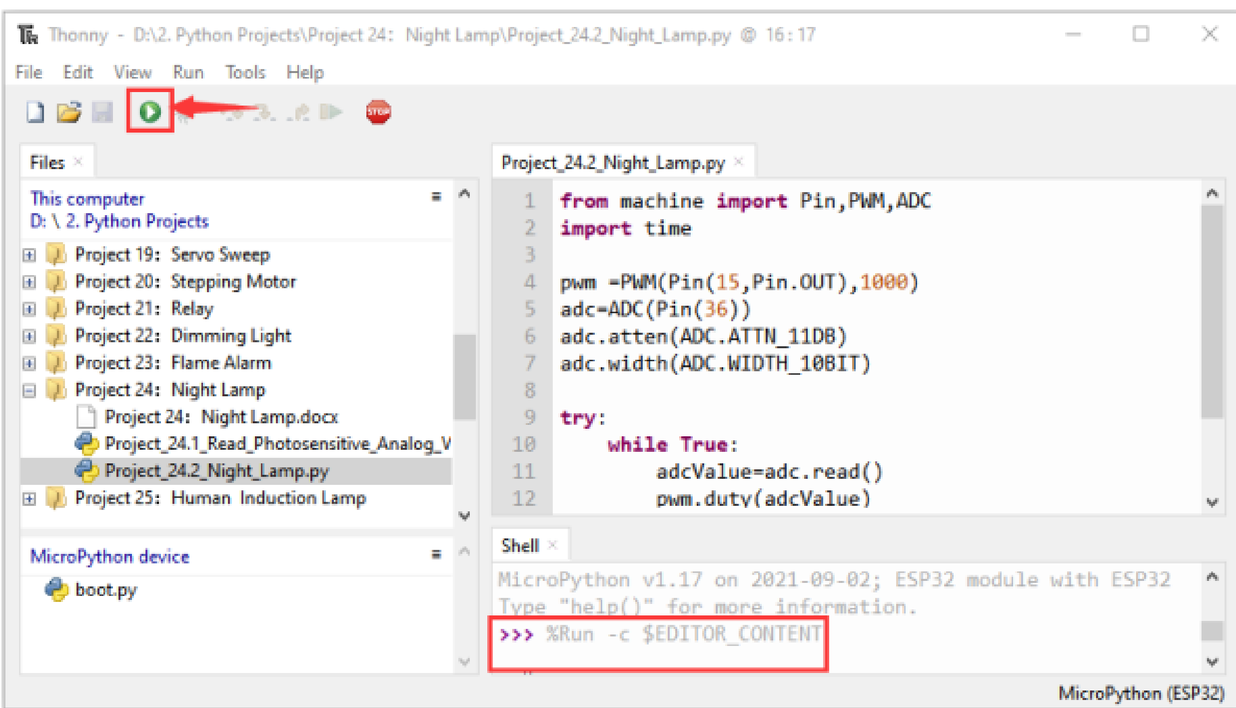
7.25.7 7.Project result

Make sure the ESP32 has been connected to the computer, click "Stop/Restart backend" .



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.



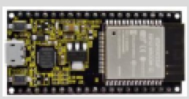
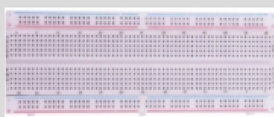
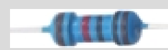
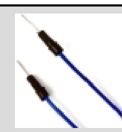
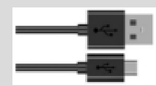
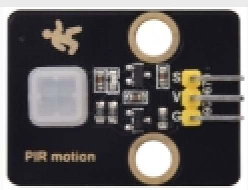
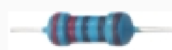
7.26 Project 25Human Induction Lamp

7.26.1 1.Introduction

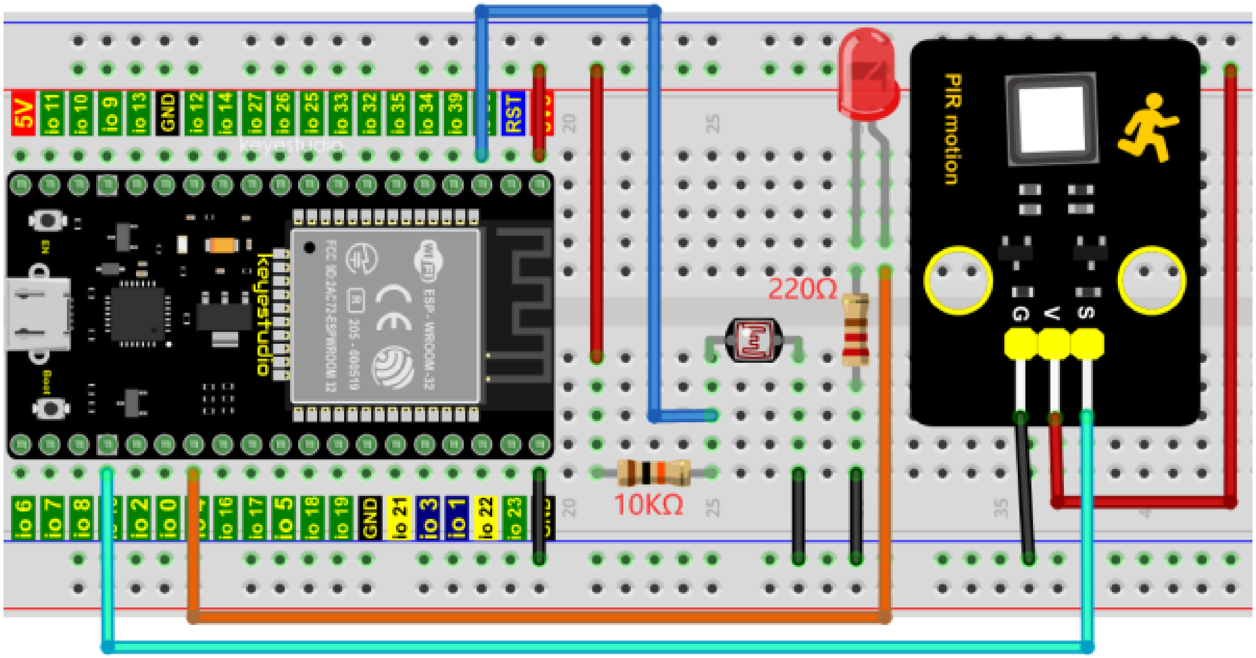
Human body induction lamp is used commonly in the dark corridor area. With the development of science and technology, the use of the human body induction lamp is very common in our real life, such as the corridor of the community, the bedroom of the room, the garage of the dungeon, the bathroom and so 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

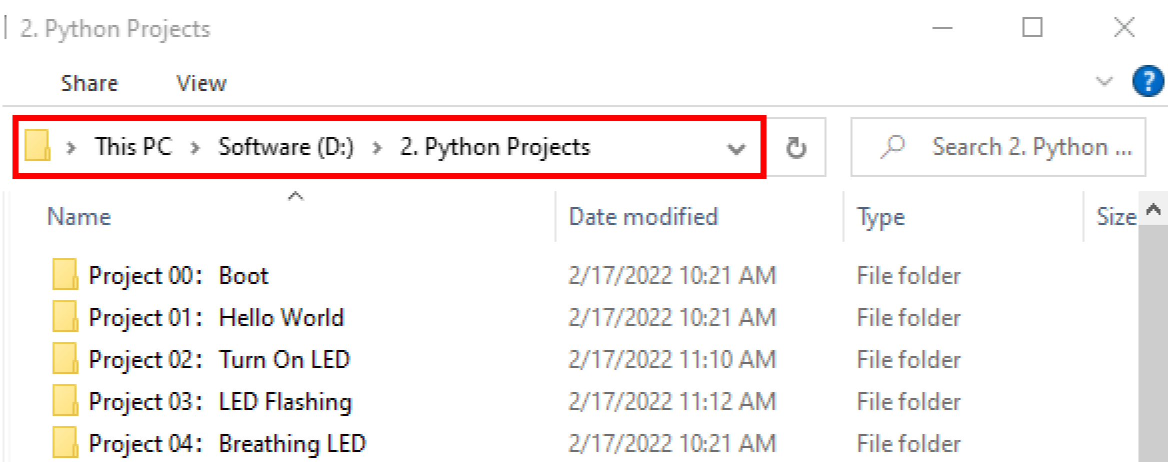
					
ESP32*1	Breadboard*1	Red LED*1	10KResistor*1	Jumper Wires	USB Cable*1
					
Photoresistor*1	Human Body Infrared Sensor*1	220Resistor*1	M-F Dupont Wires		

7.26.3 3.Wiring Diagram

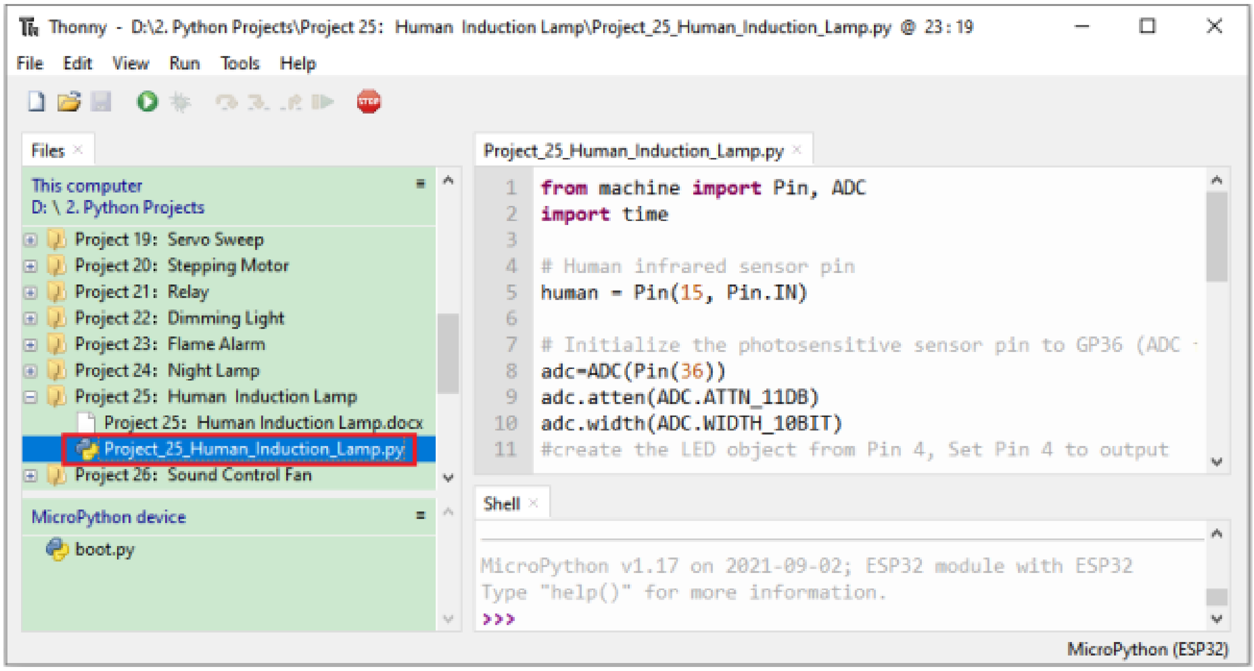


7.26.4 4.Project code

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



Open"Thonny", click"This computer"→"D:"→"2. Python Projects"→"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 = 0

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 = 0
```

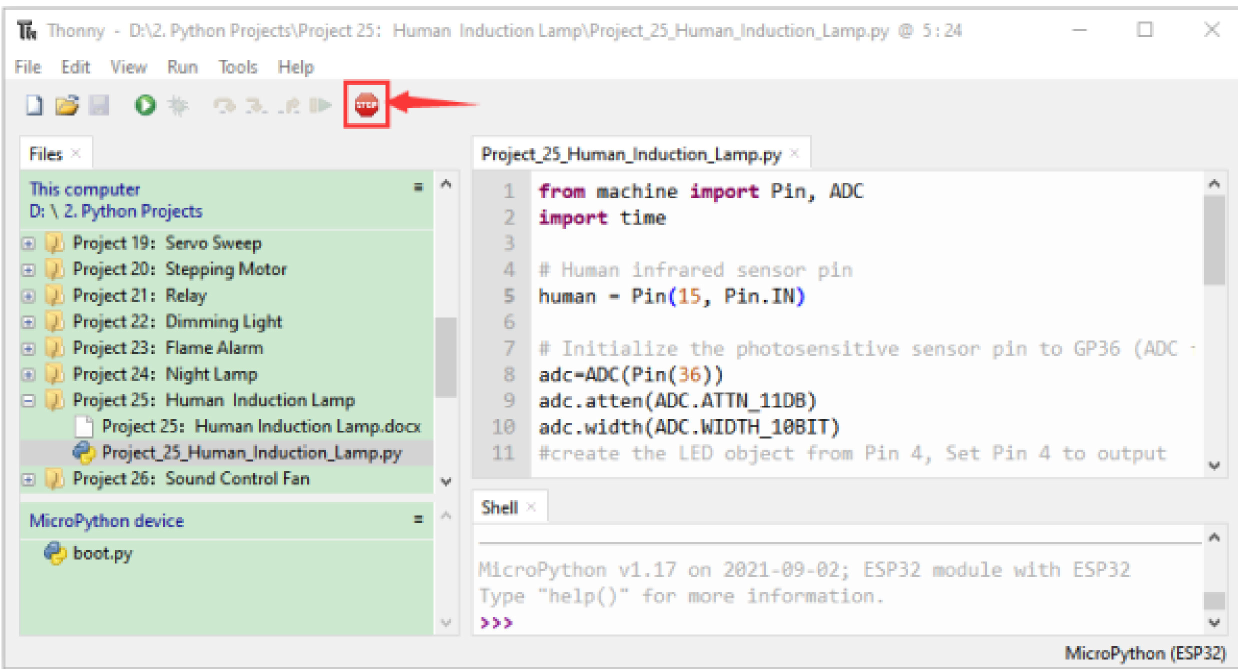
(continues on next page)

(continued from previous page)

```
                led.value(0)
    else:
        led.value(0)

    time.sleep(0.1)
```

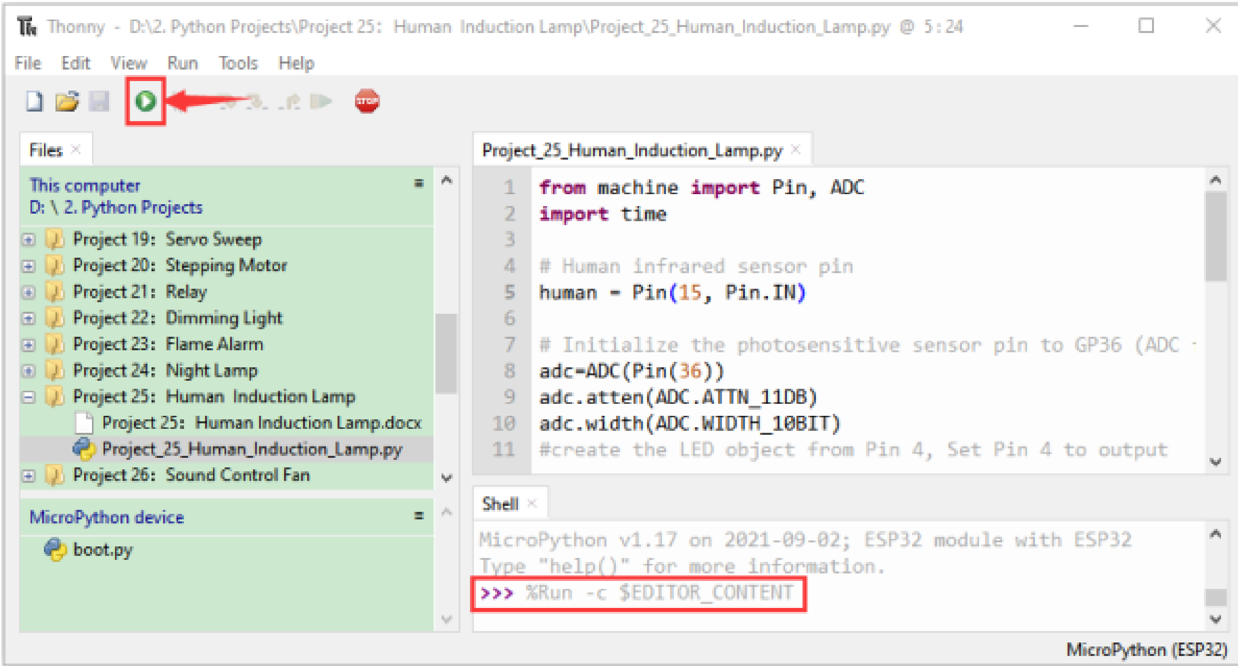
7.26.5 5.Project result

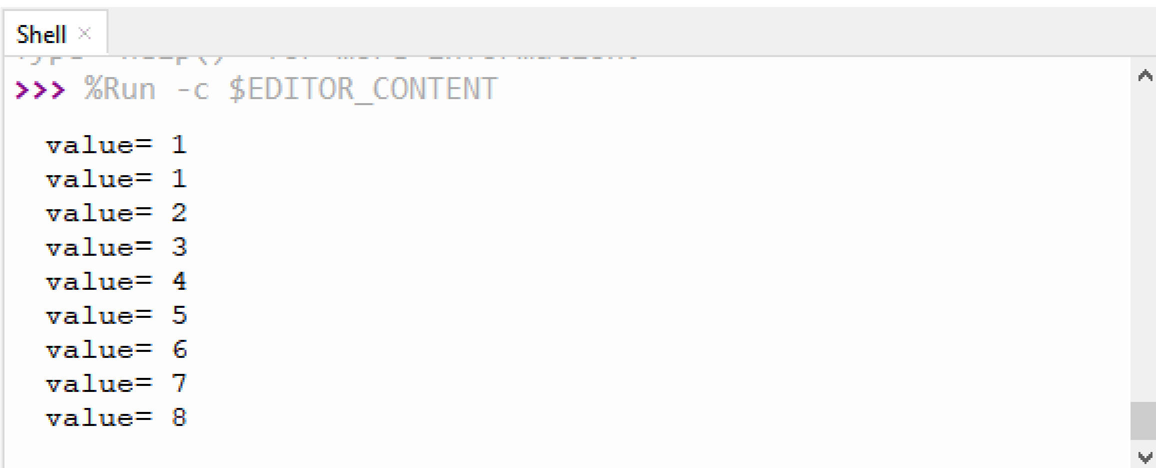
Make sure the ESP32 has been connected to the computer, click "Stop/Restart backend" .



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.



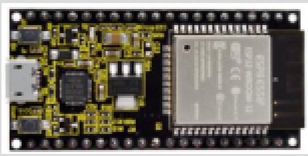
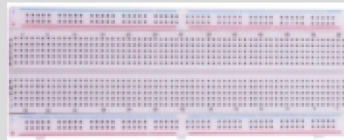
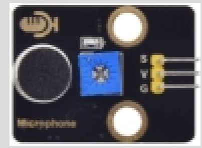
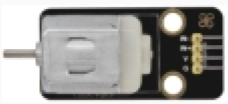
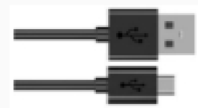
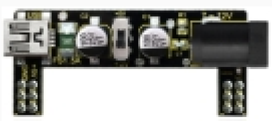
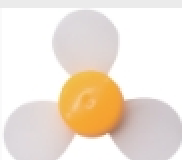


7.27 Project 26Sound Control Fan

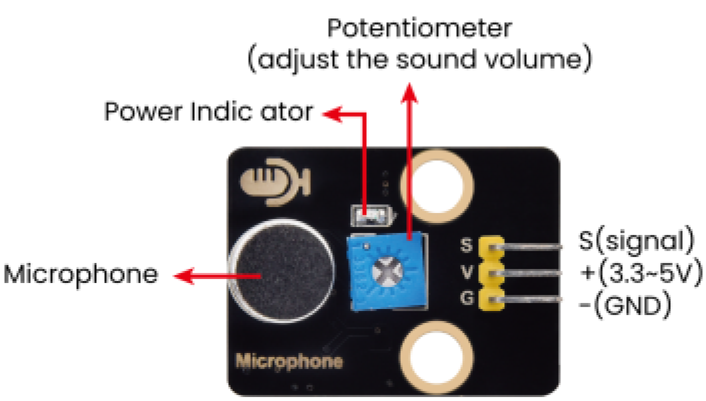
7.27.1 1.Introduction

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

		
ESP32*1	Breadboard*1	Sound Sensor*1
		
130 Motor Module*1	M-F Dupont Wires	USB Cable*1
		
Keyestudio bread board special power module*1	Battery Holder*1	No.5 battery (self-provided)*6
		
Fan*1		

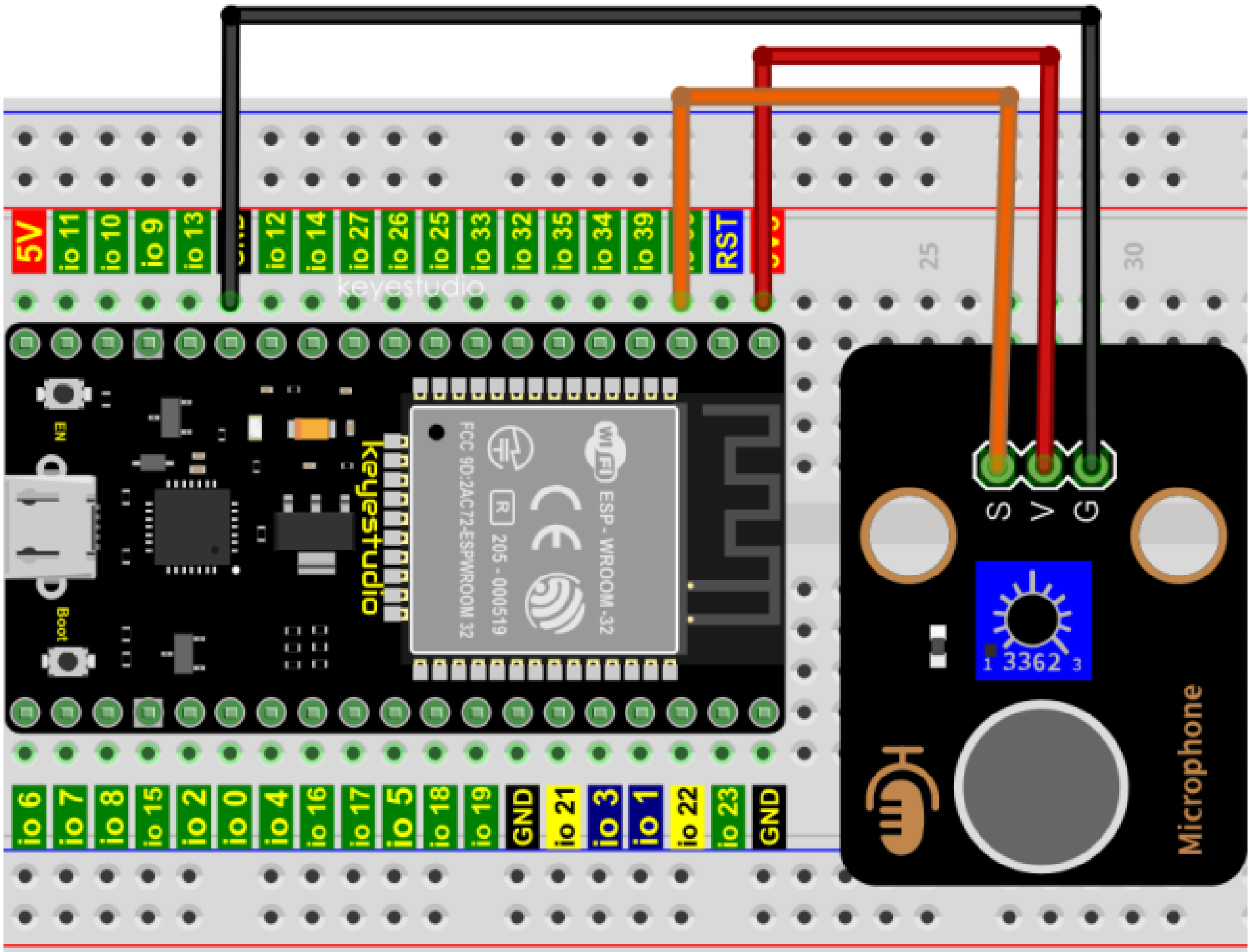
7.27.3 3.Component knowledge



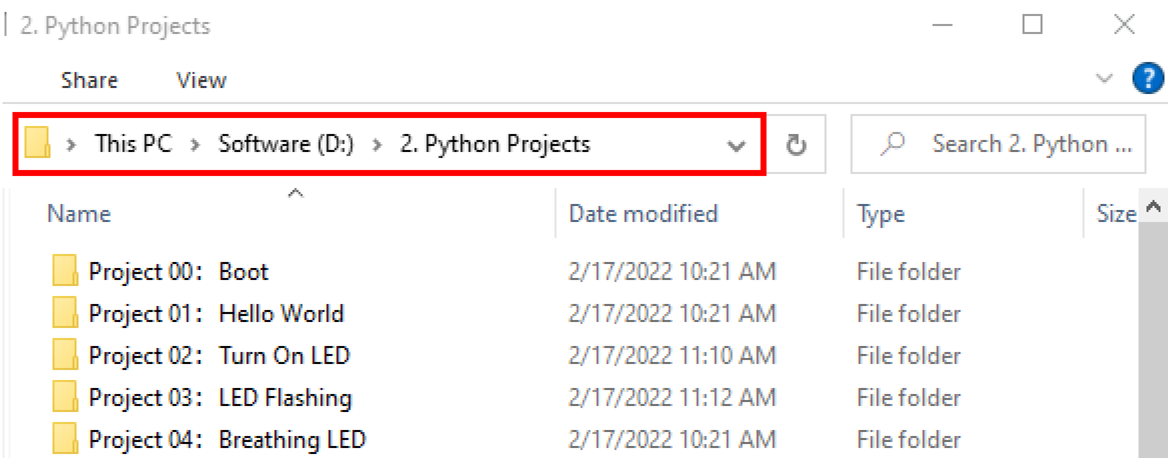
Sound sensor is usually used to detect the loudness of the sound in the surrounding environment. Microcontrol board can collect its output signal through the analog input interface.The S pin is an analog output, which is the real-time output of the microphone voltage signal. The sensor comes with a potentiometer 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

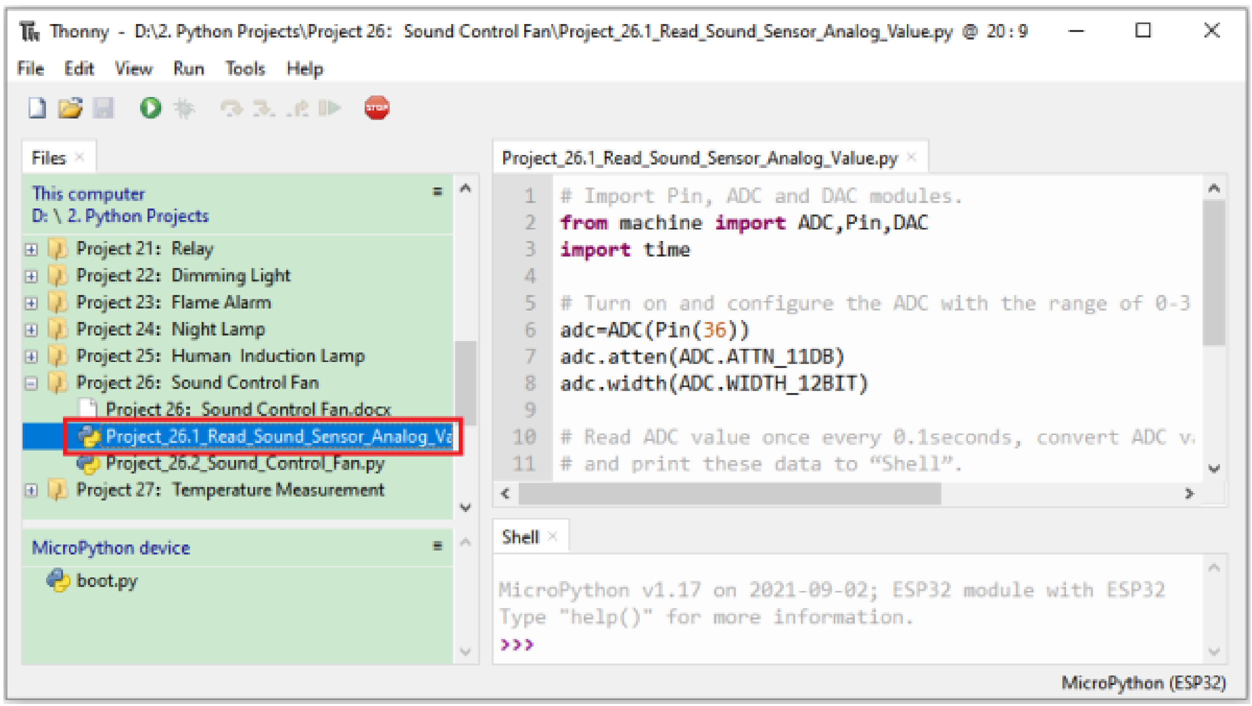
We first use a simple code to read the ADC value, DAC value and voltage value of the sound sensor and print them out. Please refer to the wiring diagram below



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



Open“Thonny”click“This computer”→“D:”→“2. Python Projects”→“Project 26Sound Control Fan”, and then double left-click “Project_26.1_Read_Sound_Sensor_Analog_Value.py”.



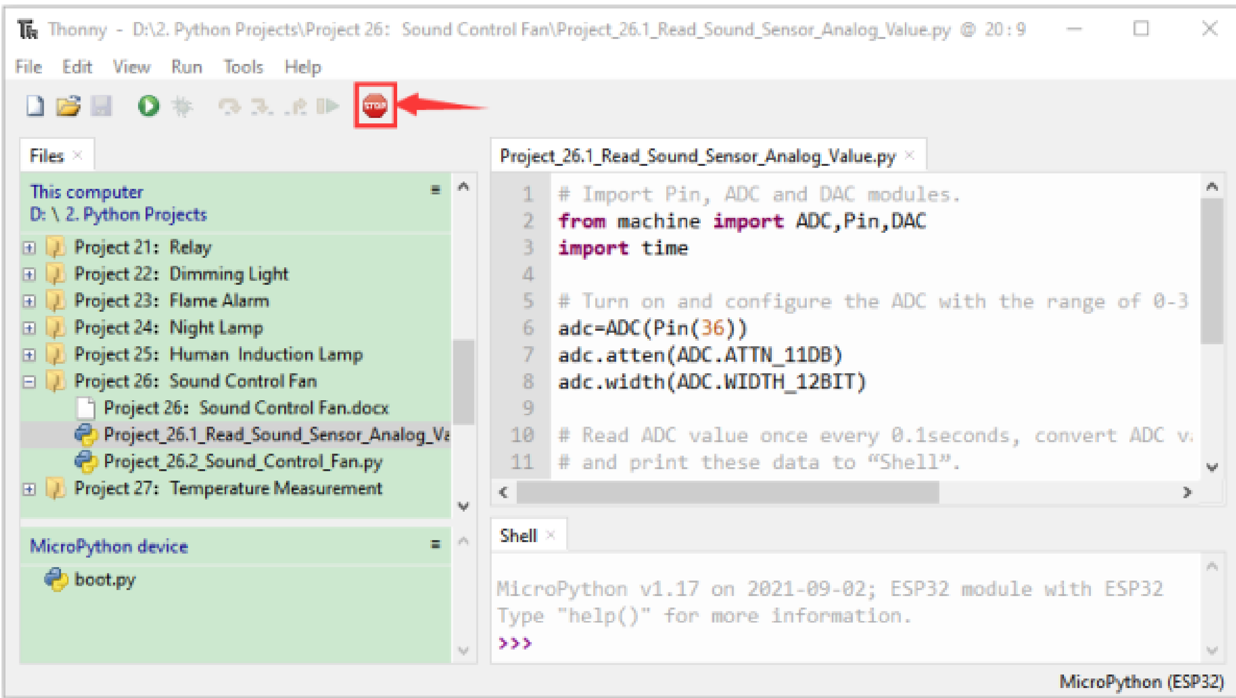

```
## Import Pin, ADC and DAC modules.
from machine import ADC,Pin,DAC
import time

## 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:
```

(continues on next page)

(continued from previous page)

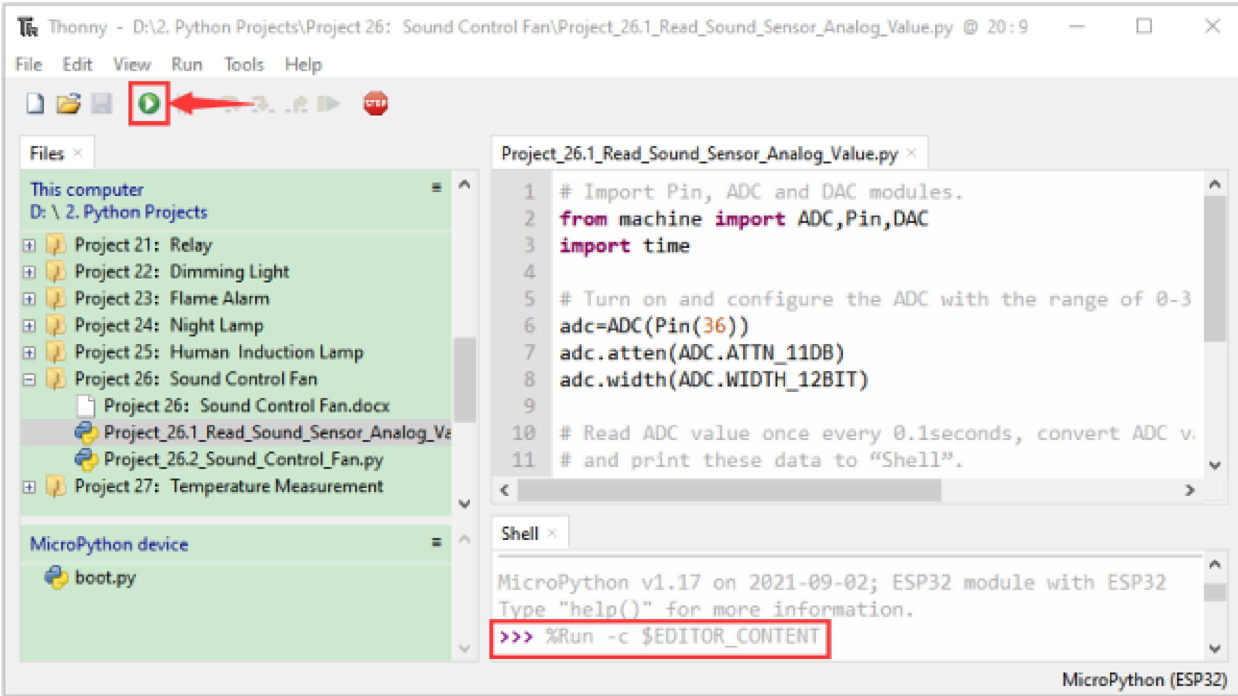
```
        adcVal=adc.read()
        dacVal=adcVal//16
        voltage = adcVal / 4095.0 * 3.3
        print("ADC Val:",adcVal,"DACVal:",dacVal,"Voltage:",voltage,"V")
        time.sleep(0.1)
except:
    pass
```

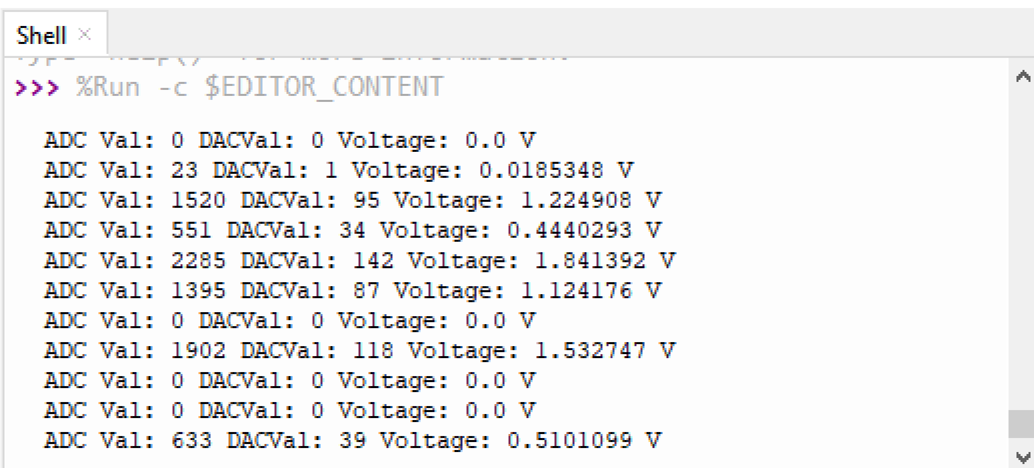
Make sure the ESP32 has been connected to the computer, click "Stop/Restart backend" .



Click "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.

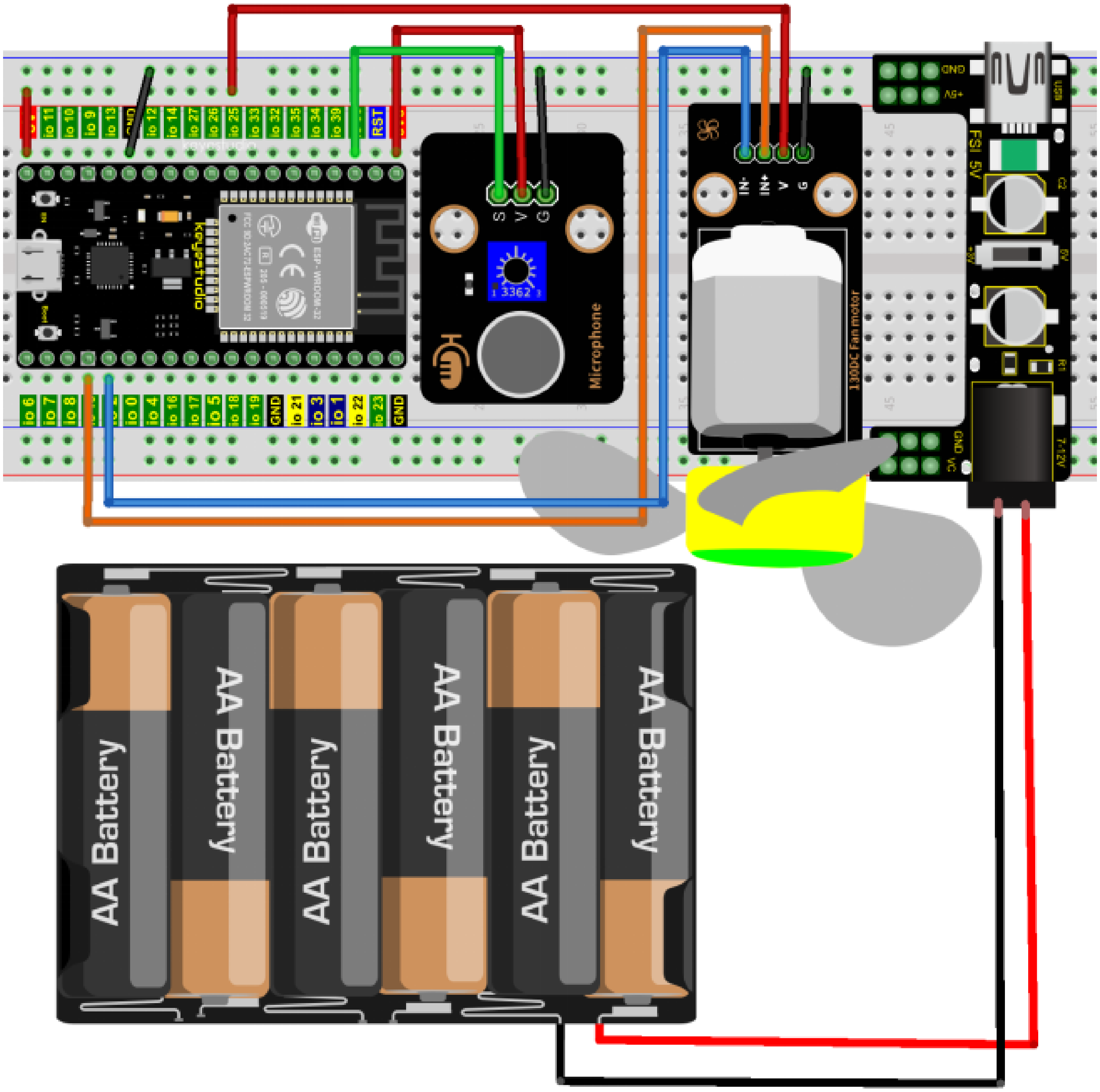
When you clap your hands to the sensor, the ADC value, DAC value and voltage value will change significantly. Press"Ctrl+C"or click "Stop/Restart backend"to exit the program.





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 voice-controlled fan. The wiring diagram is as follows

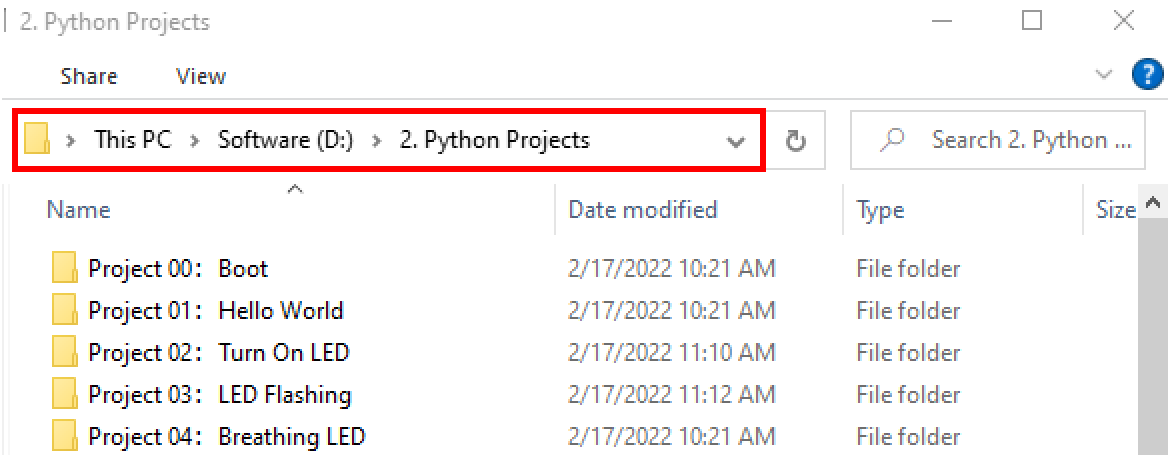


(Note: Connect the wires and then install a small fan blade on the DC motor.)

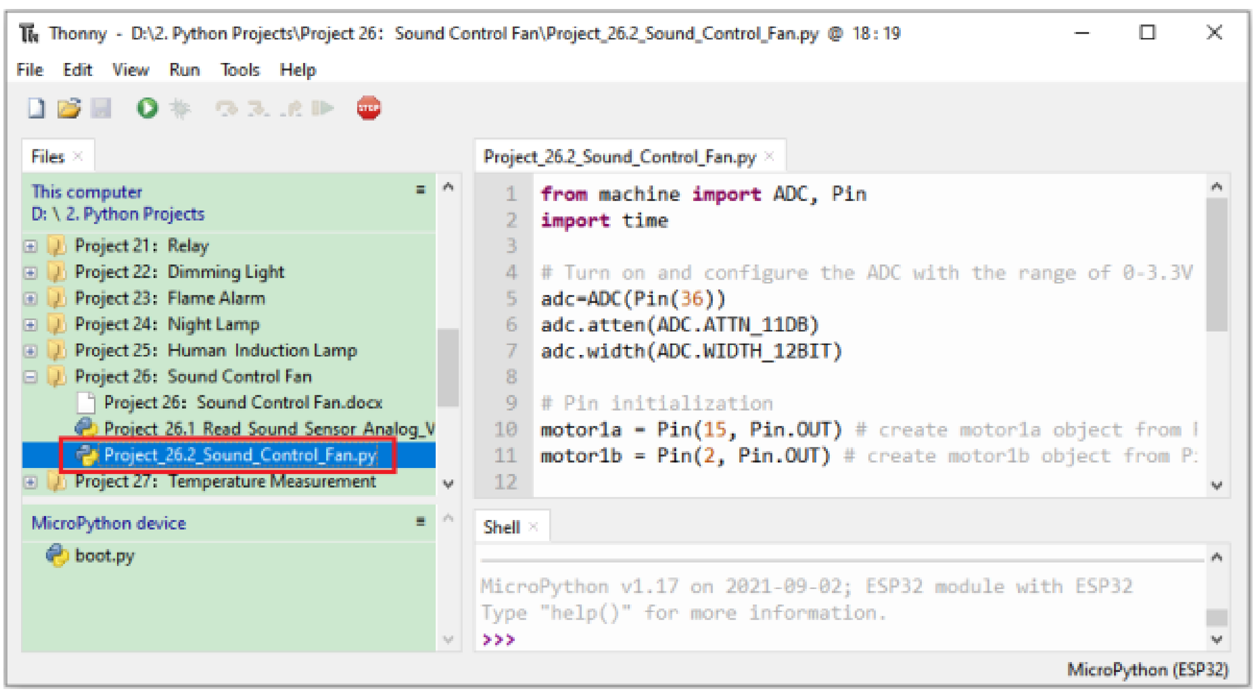
7.27.6 6.Project code

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



Open"Thonny", click "This computer"→"D:"→"2. Python Projects"→"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)
```

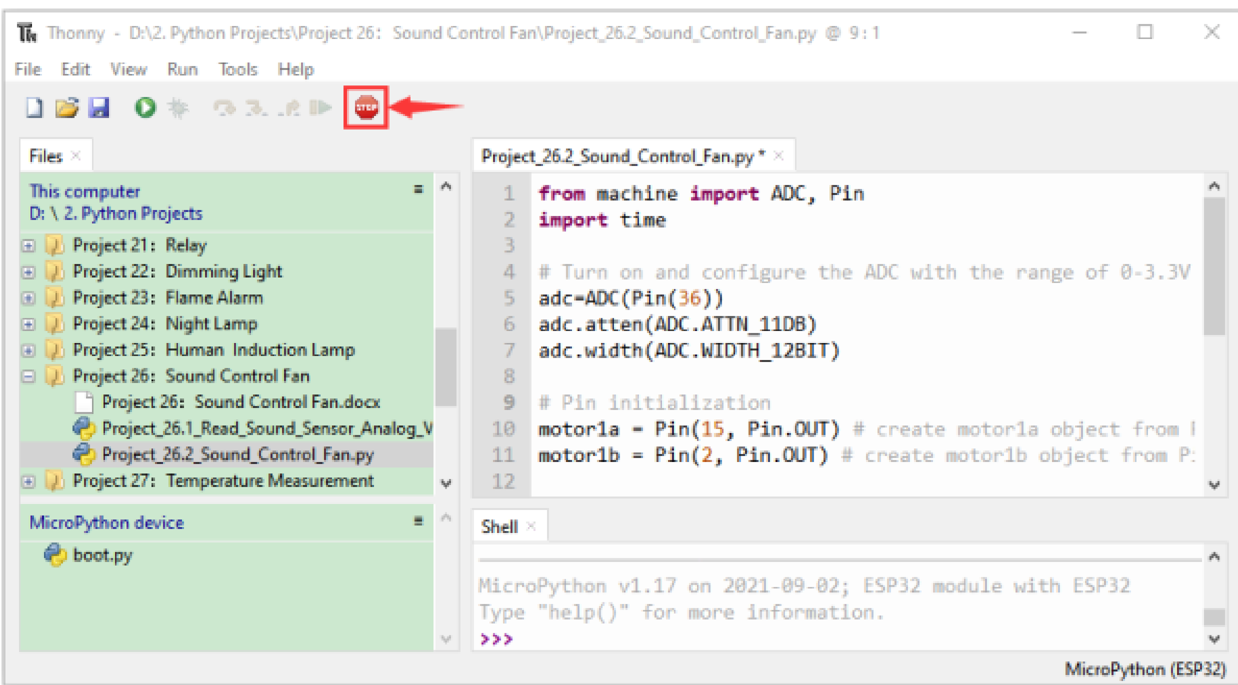
(continues on next page)

(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,Otherwise, 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

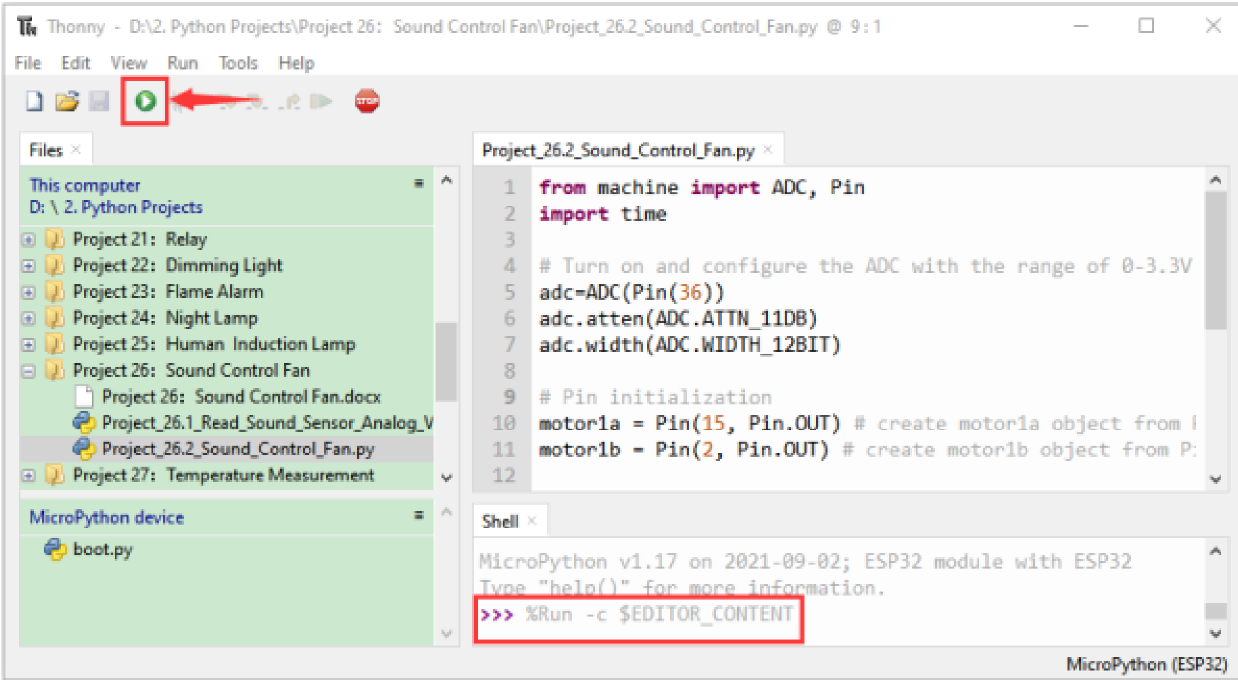
Make sure the ESP32 has been connected to the computer, click "Stop/Restart backend" .



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 "Stop/Restart backend"to exit the program.

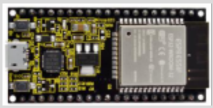
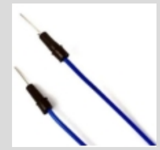


7.28 Project 27Temperature Measurement

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

					
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		

7.28.3 3.Component knowledge

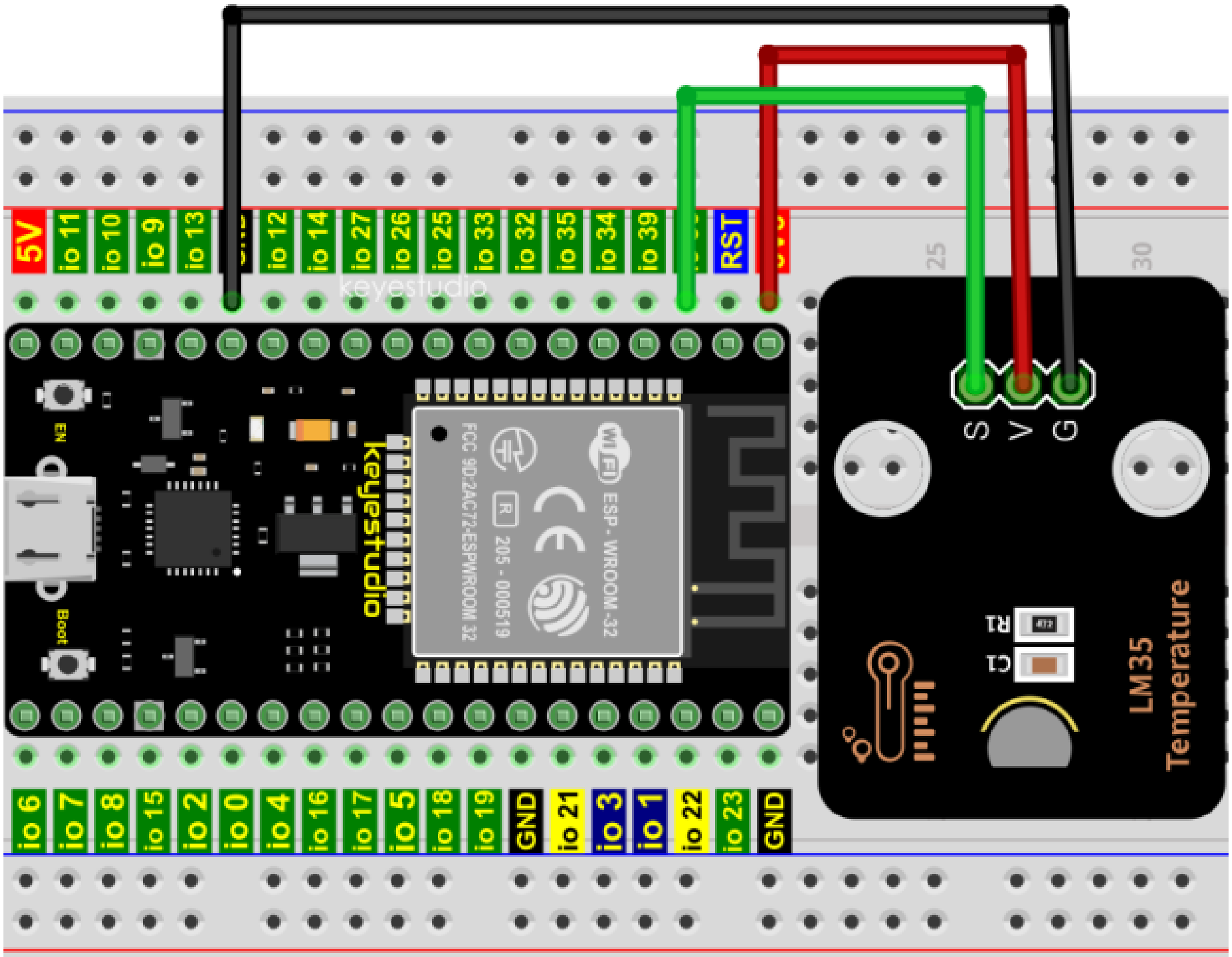
Working principle of LM35 temperature sensor:

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°C without additional calibration processing. LM35 temperature sensor can produce different voltage according to different temperatures, when the temperature is 0 ℃, it output 0V; If increasing 1 ℃, the output voltage will increase 10mv. The output temperature is 0°C to 100°C, the conversion formula is as follows

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

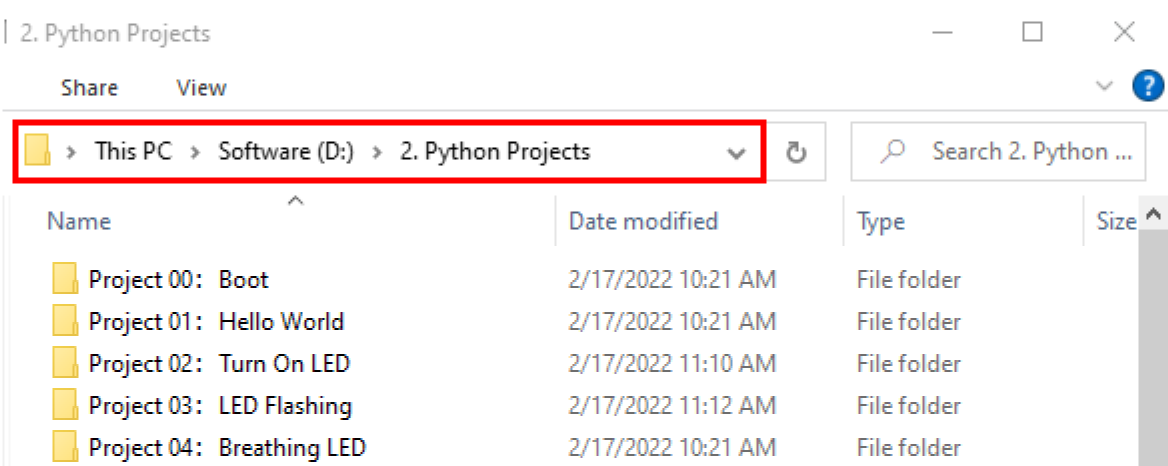
7.28.4 4.Read the temperature value of the LM35

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

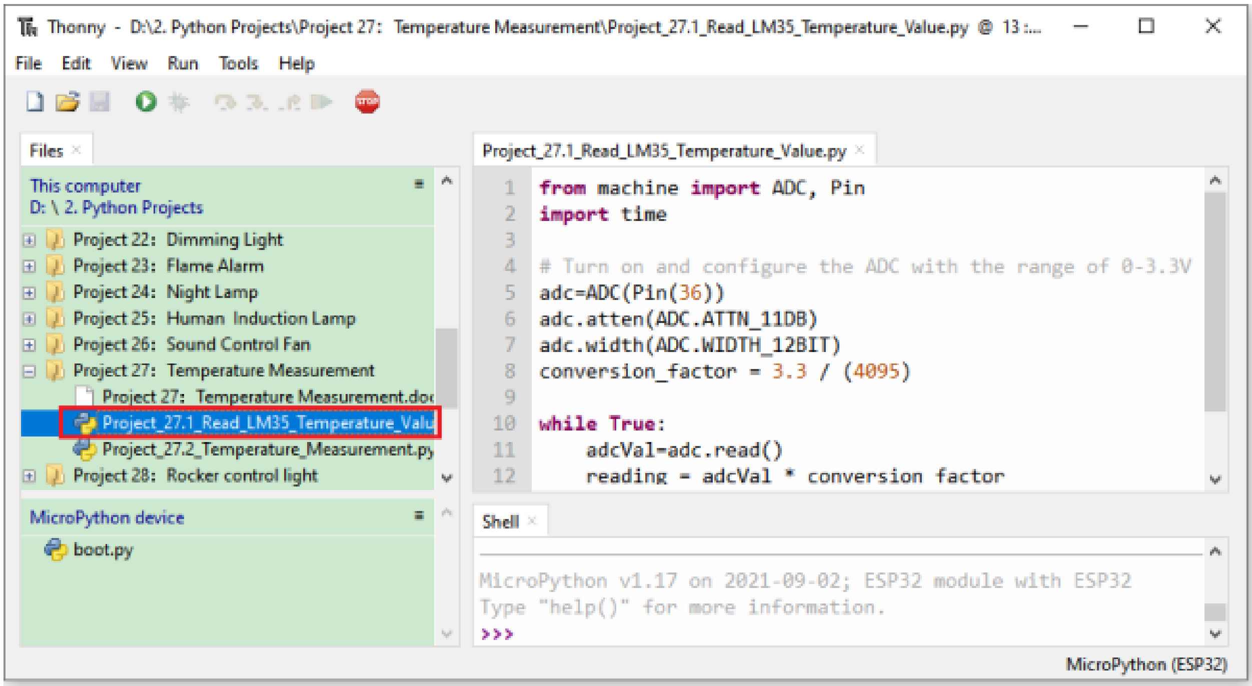


LM35 output is given to analog pin GPIO36 of the ESP32, this analog voltage is converted to its digital form and processed to get the temperature reading.

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

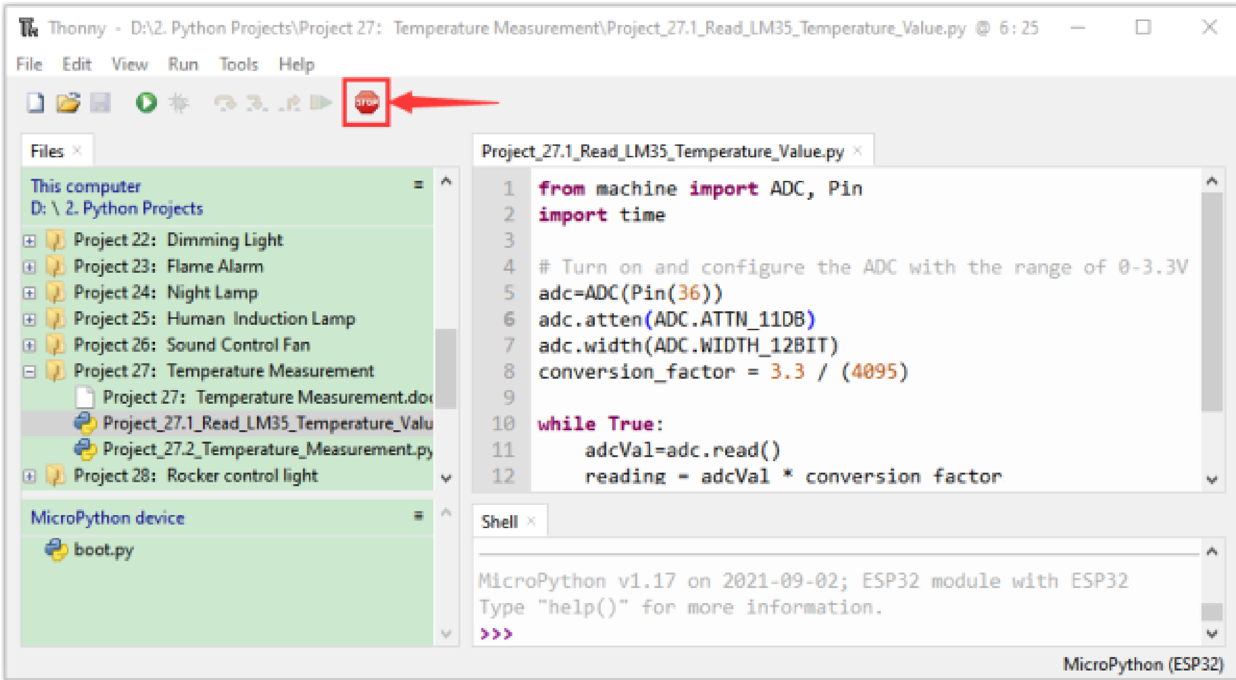


Open “Thonny”click “This computer”→“D:”→“2. Python Projects”→“Project 27Temperature Measurement”, and then double left-click “Project_27.1_Read_LM35_Temperature_Value.py”.



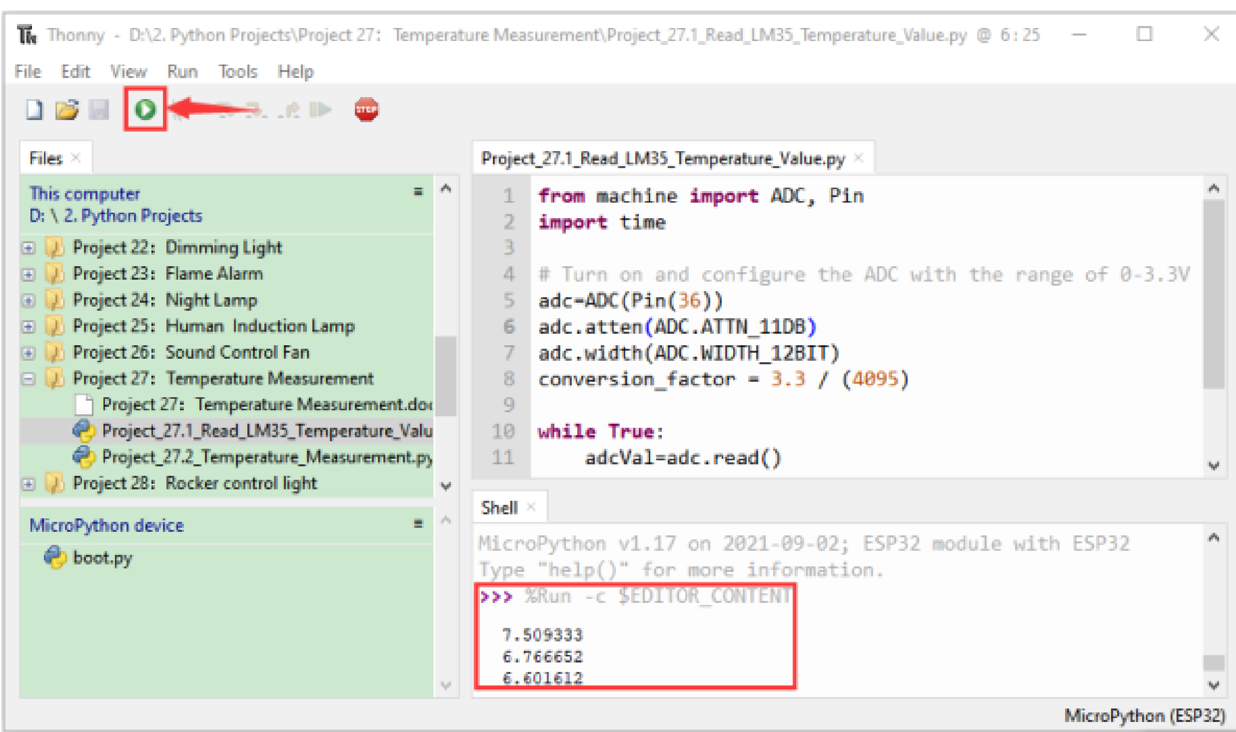

```python
from machine import ADC, Pin
import time

## 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)
    time.sleep(1)
```

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.

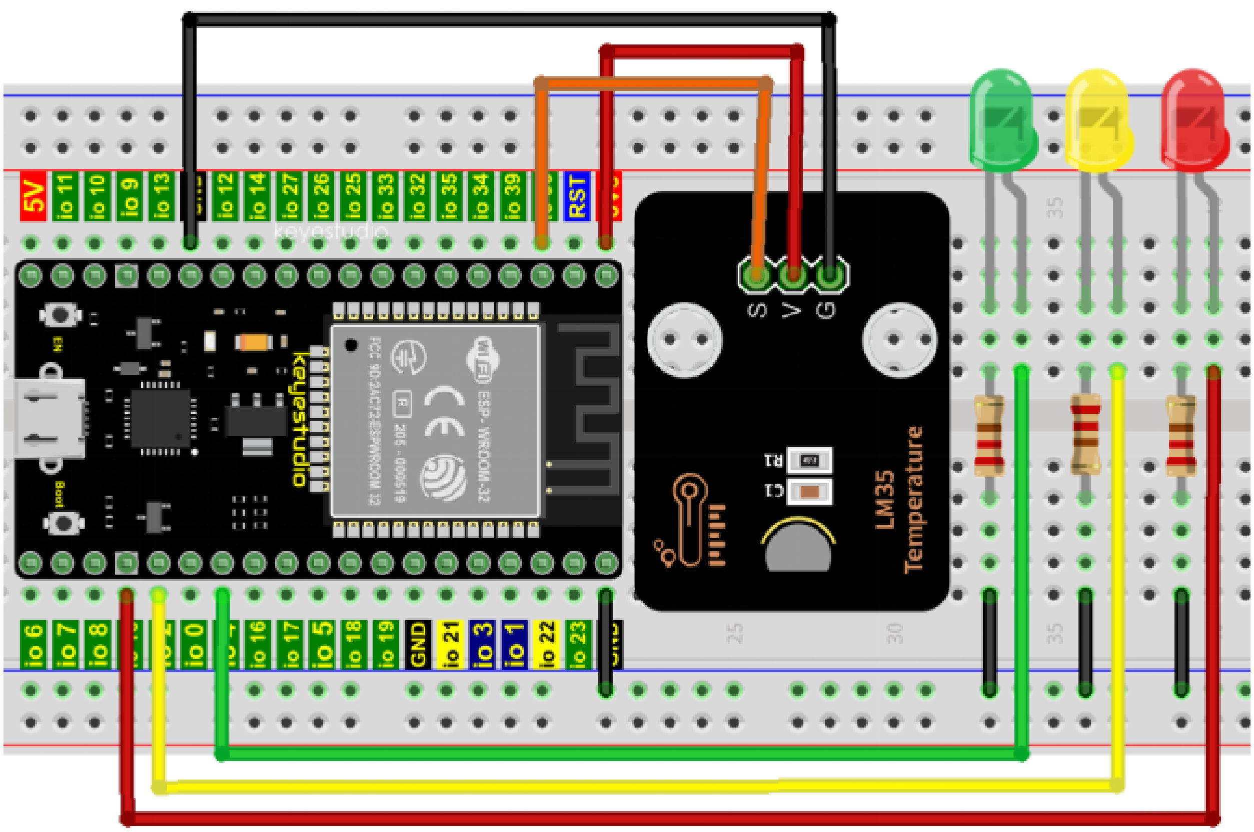
Hold the LM35 element by hand, the temperature value read by the LM35 temperature sensor will change. Press “Ctrl+C” or click“Stop/Restart backend” to exit the program.



```
Shell ×
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
```

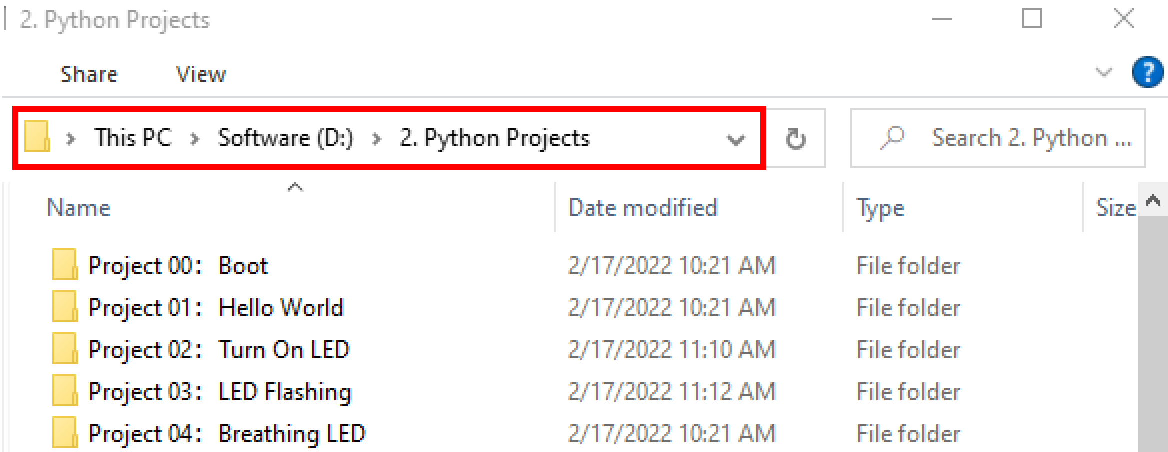
7.28.5 5.Diagram of the temperature measurement

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.



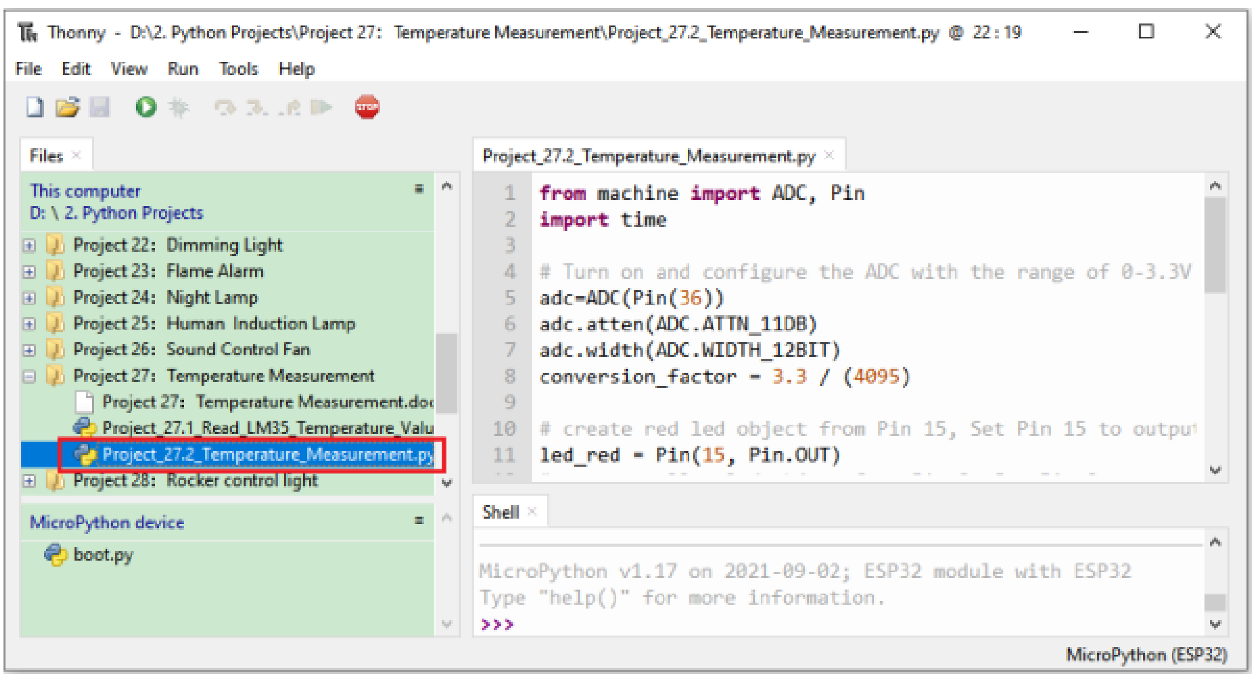
7.28.6 6.Project code

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



Open"Thonny"click"This computer"→"D:"→"2. Python Projects"→"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)

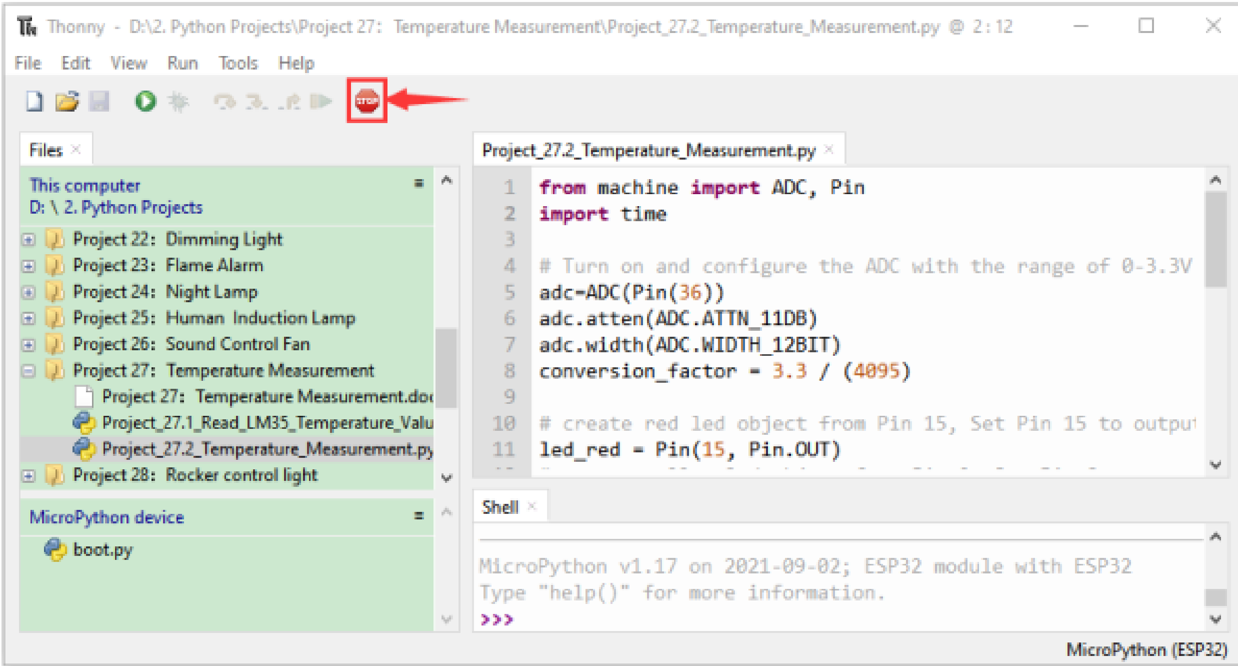
(continued from previous page)

```
## 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:
        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

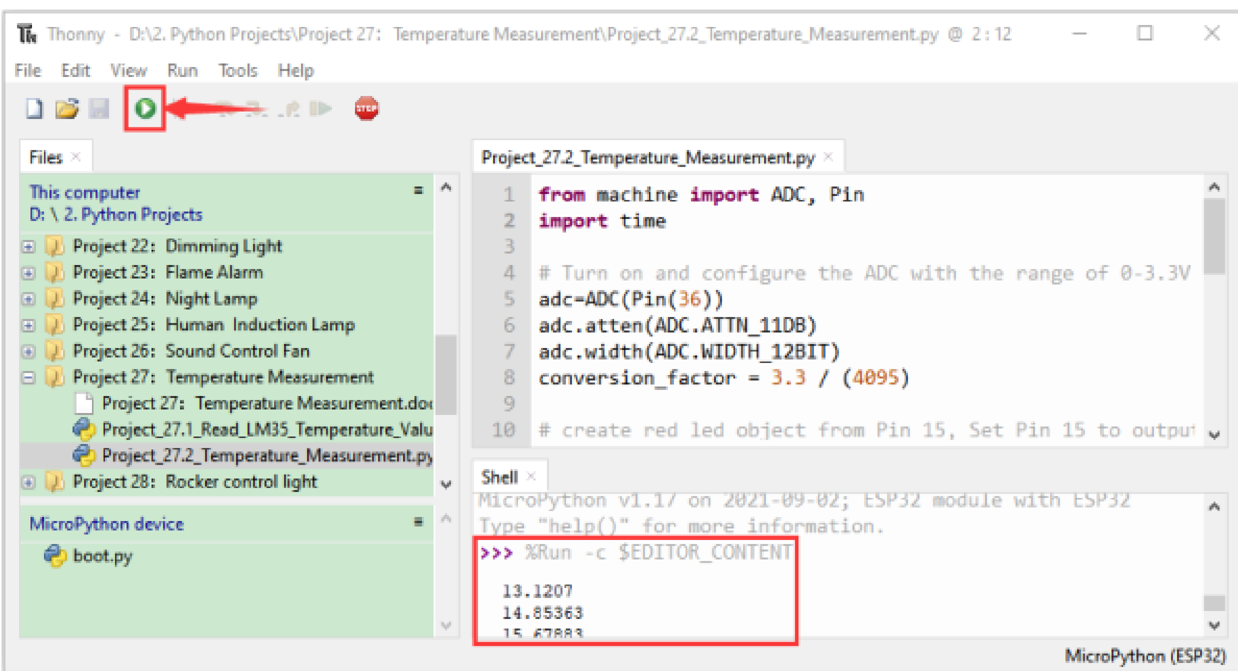
Make sure the ESP32 has been connected to the computer, click 

"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.



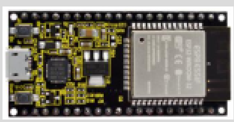
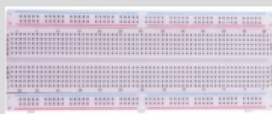
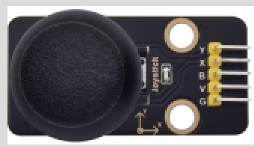
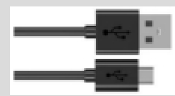
7.29 Project 28Rocker control light

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

				
ESP32*1	Breadboard*1	Rocker Module*1	USB Cable*1	M-F Dupont Wires
				
RGB LED*1	220Resistor*3	Jumper Wires		

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.

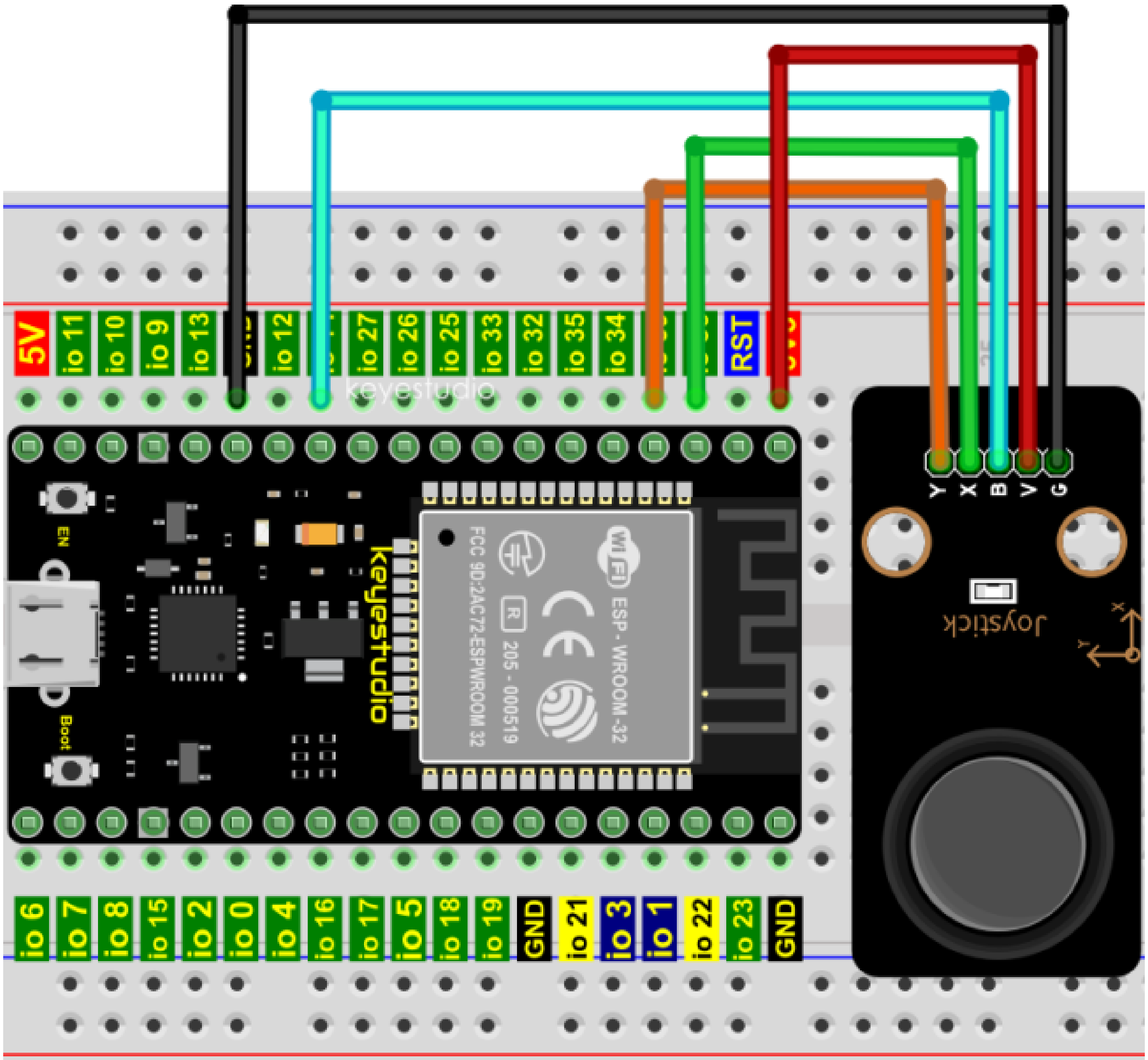
In this way, we can read the two analog values and the high and low level conditions of the digital port to determine the operating status of the joystick on the module.

Features:

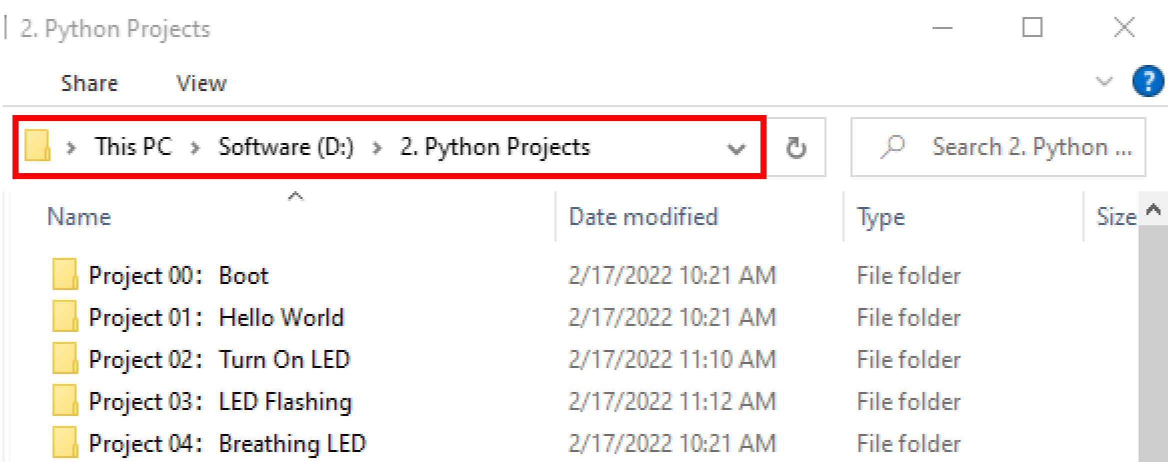
- Input Voltage: DC 3.3V ~ 5V.
- Output Signal: X/Y dual axis analog value +Z axis digital signal.
- Range of Allication: Suitable for control point coordinate movement in plane as well as control of two degrees of freedom steering gear, etc.
- Product features: Exquisite appearance, joystick feel superior, simple operation, sensitive response, long service life.

7.29.4 4.Read the value of the Rocker Module

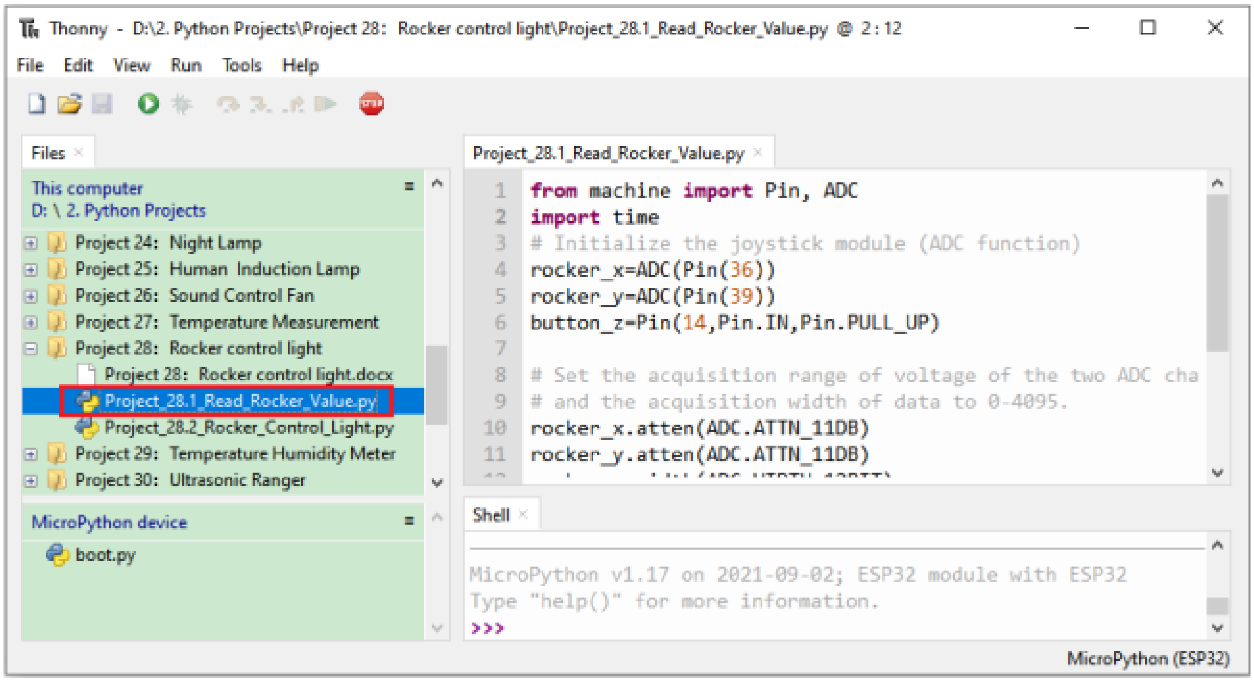
We must use ESP32's analog IO port to read the value from the X/Y pin of the rocker module and use the digital IO port to read the digital signal of the button. Please connect the wires according to the wiring diagram below



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

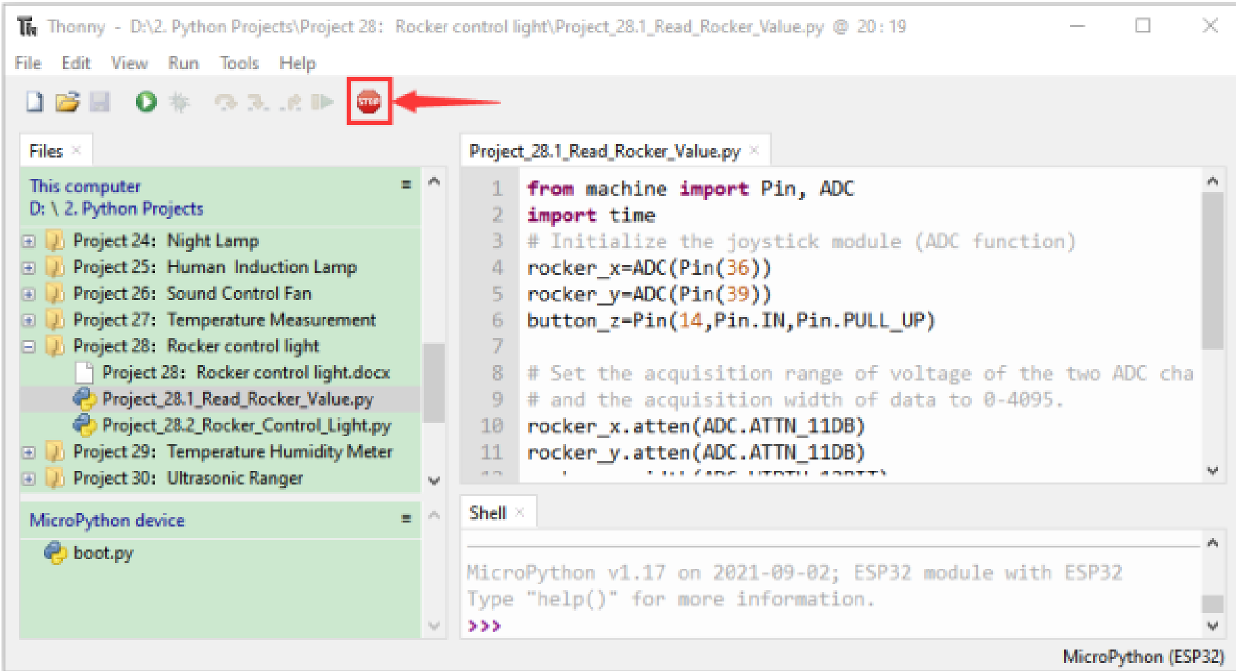


Open“Thonny”click“This computer”→“D:”→“2. Python Projects”→“Project 28Rocker control light”, and then double left-click “Project_28.1_Read_Rocker_Value.py”.



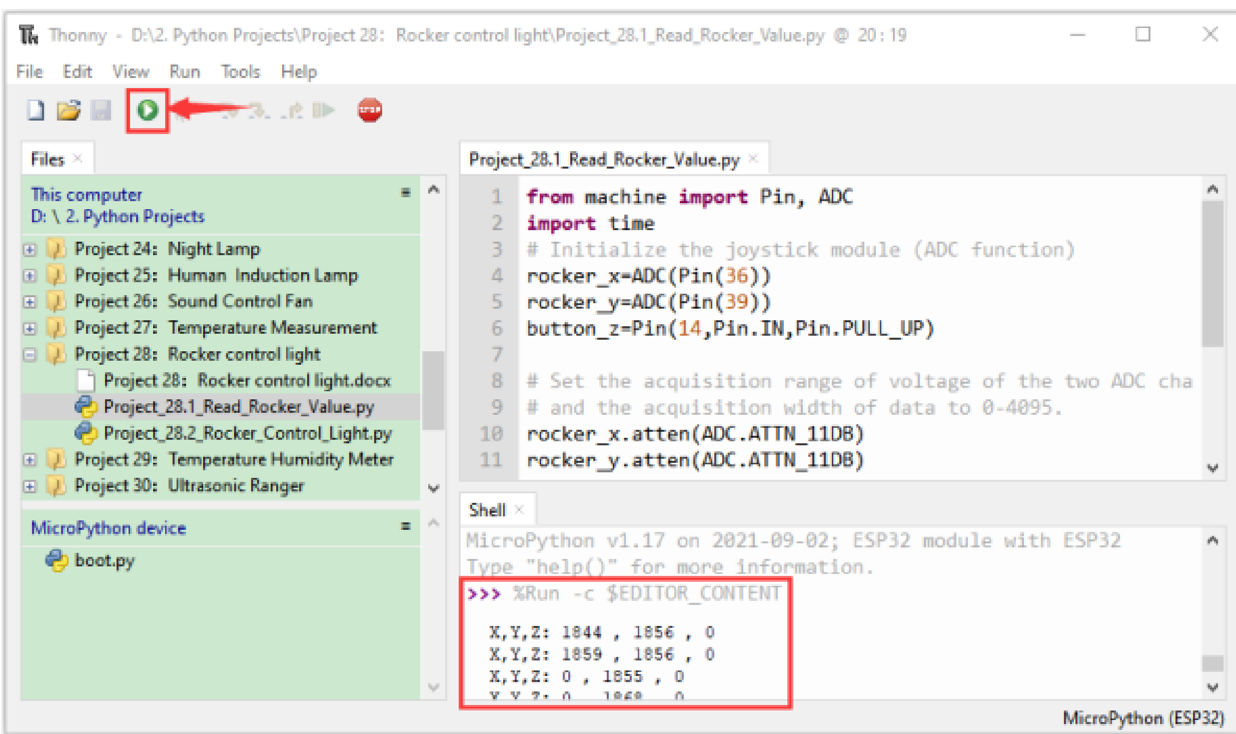

```
from machine import Pin, ADC
import time
## Initialize the joystick module (ADC function)
rocker_x=ADC(Pin(36))
rocker_y=ADC(Pin(39))
button_z=Pin(14,Pin.IN,Pin.PULL_UP)

## 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)
```

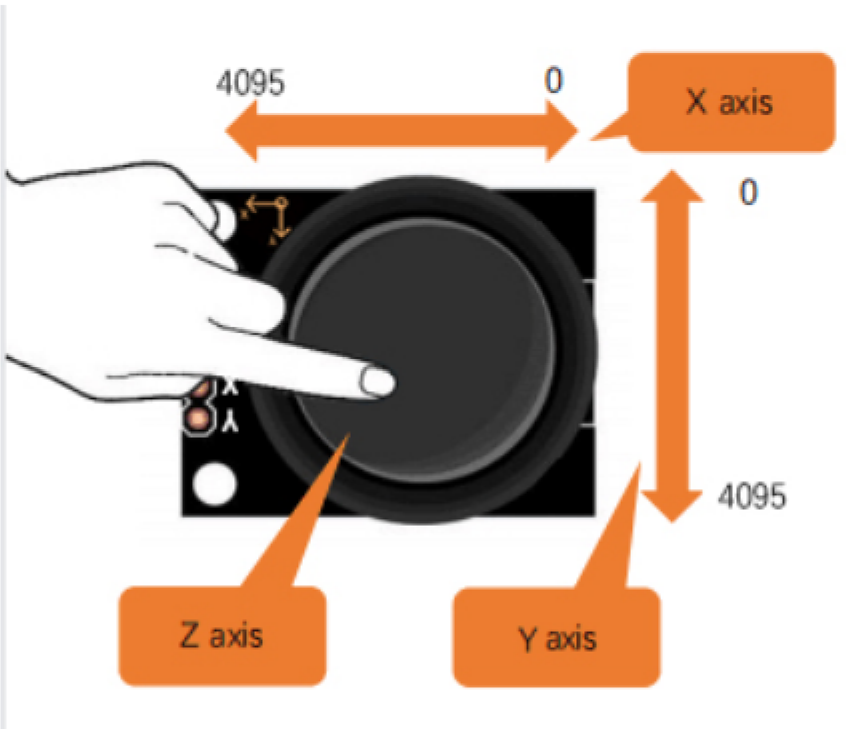
Make sure the ESP32 has been connected to the computer, click“Stop/Restart backend” .

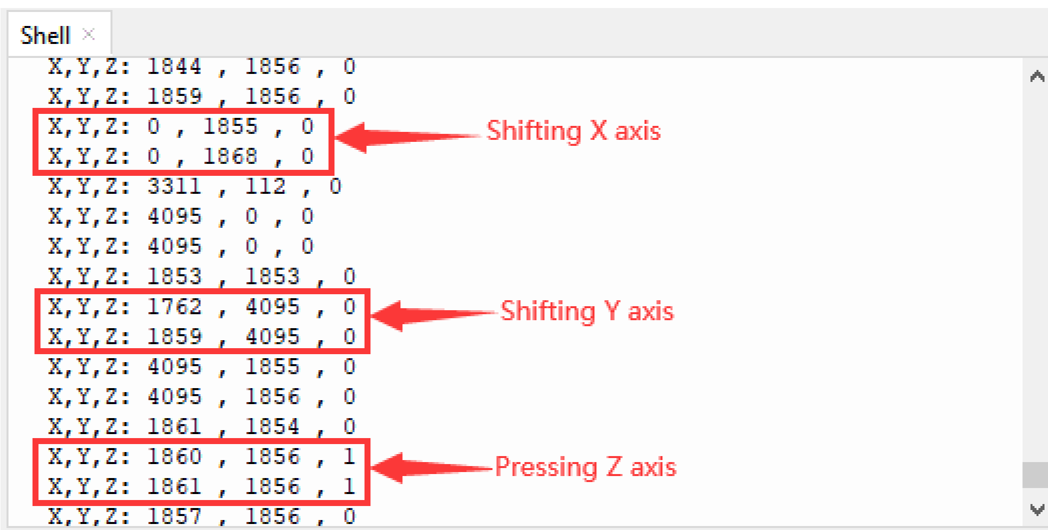


Click “Run current script”, the code starts to be executed and you’ll see that the “Shell” window of Thonny IDE will print the analog and digital values of the current joystick.

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.

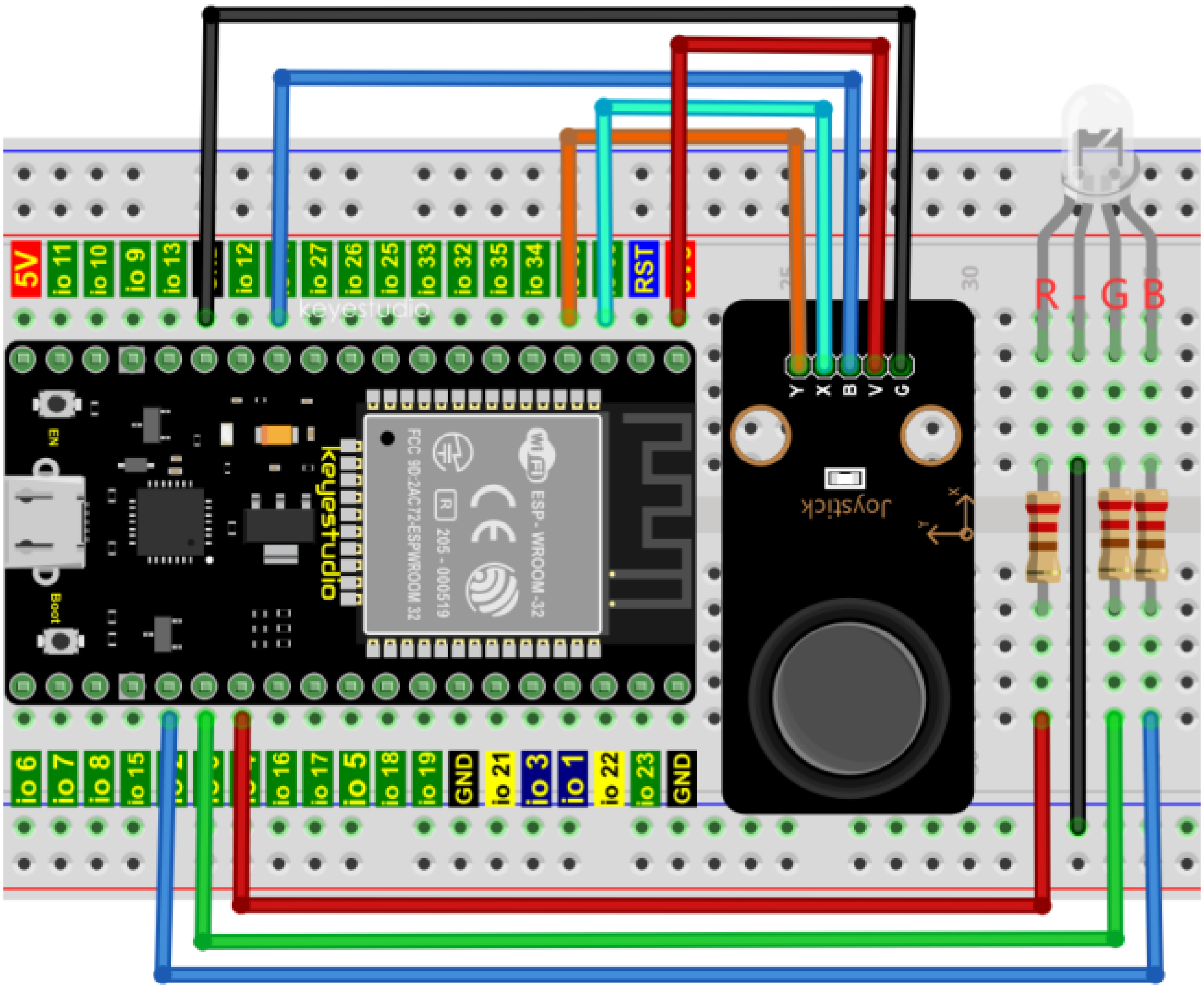



4095
0
X axis
0
4095
Z axis
Y axis


Shell
X,Y,Z: 1844 , 1856 , 0
X,Y,Z: 1859 , 1856 , 0
X,Y,Z: 0 , 1855 , 0
X,Y,Z: 0 , 1868 , 0
Shifting X axis
X,Y,Z: 3311 , 112 , 0
X,Y,Z: 4095 , 0 , 0
X,Y,Z: 4095 , 0 , 0
X,Y,Z: 1853 , 1853 , 0
X,Y,Z: 1762 , 4095 , 0
X,Y,Z: 1859 , 4095 , 0
Shifting Y axis
X,Y,Z: 4095 , 1855 , 0
X,Y,Z: 4095 , 1856 , 0
X,Y,Z: 1861 , 1854 , 0
X,Y,Z: 1860 , 1856 , 1
X,Y,Z: 1861 , 1856 , 1
Pressing Z axis
X,Y,Z: 1857 , 1856 , 0

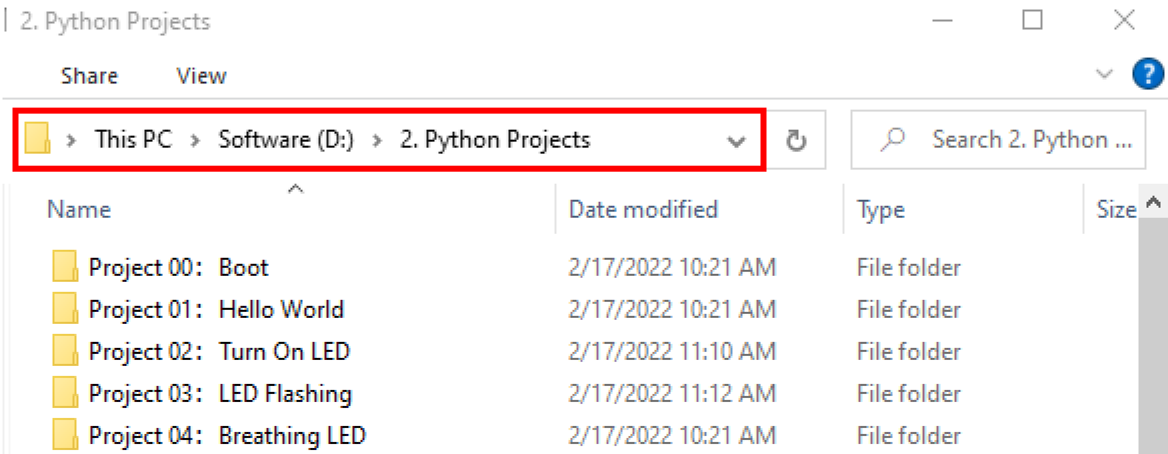
7.29.5 5.Wiring diagram

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

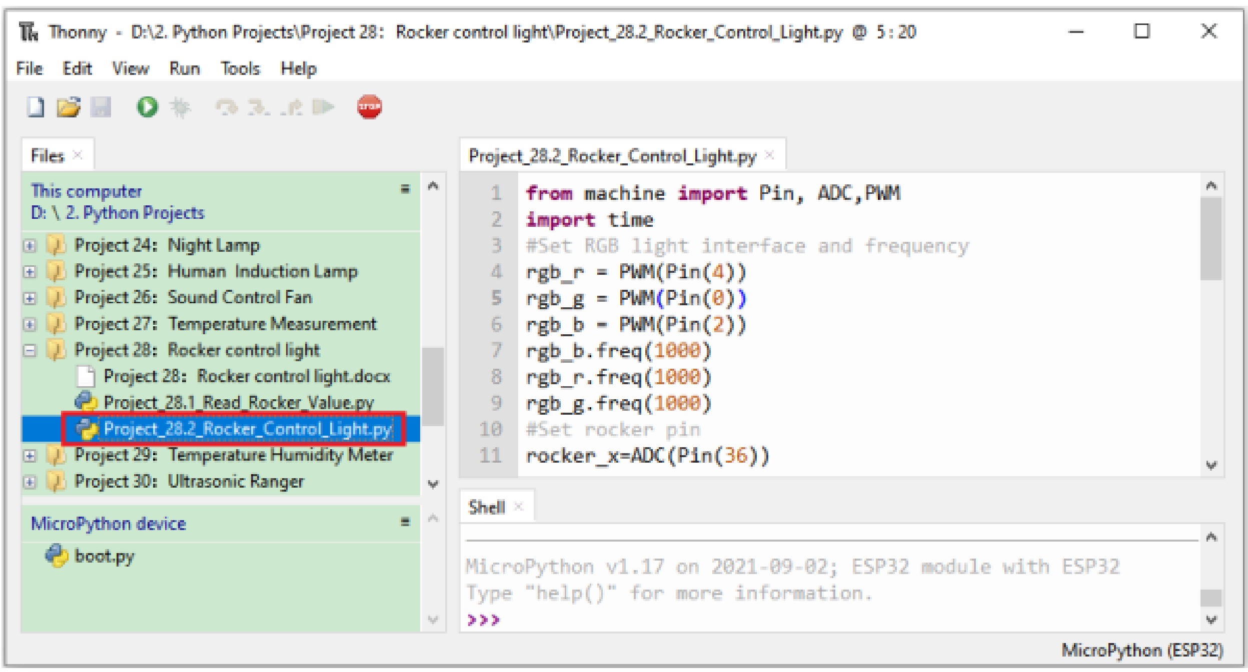


7.29.6 6.Project code

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



Open"Thonny"click"This computer"→"D:"→"2. Python Projects"→"Project 28Rocker control light"and then double left-click "Project_28.2_Rocker_Control_Light.py".




```python
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(39))
## Set the acquisition range of voltage of the two ADC channels to 0-3.3V,
## and the acquisition width of data to 0-4095.
```

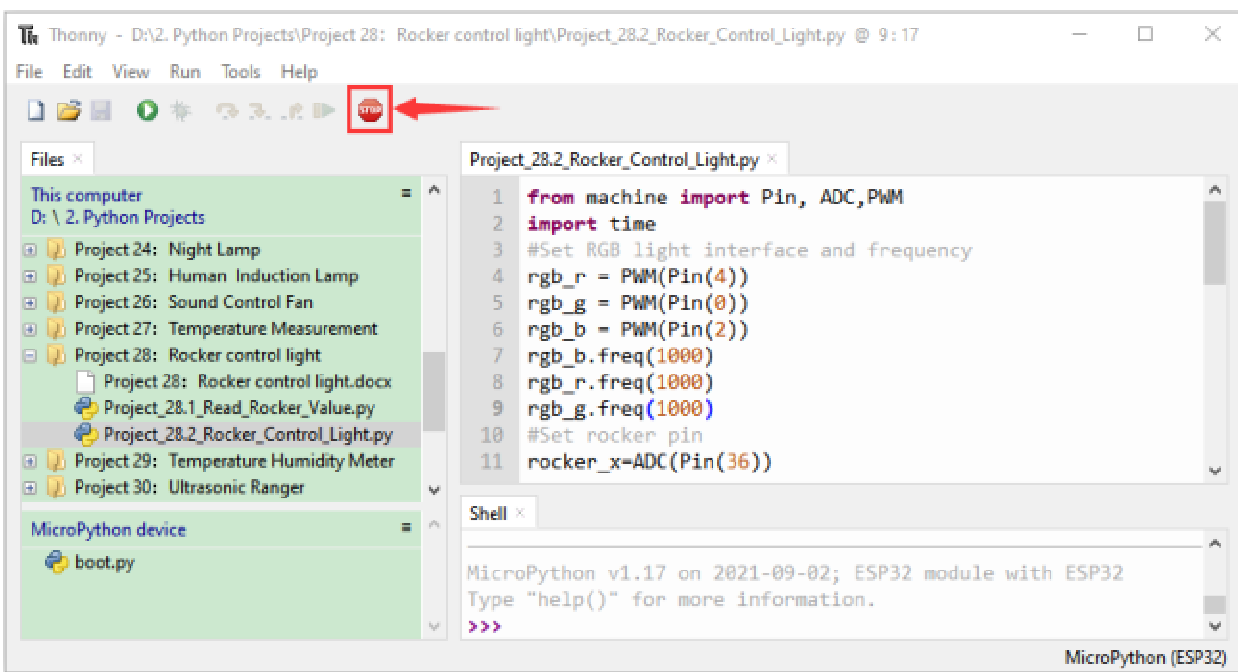
(continues on next page)

(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(0)
        rgb_r.duty(1023)
        rgb_g.duty(0)
    elif x > 3000:     #right
        rgb_b.duty(0)
        rgb_r.duty(0)
        rgb_g.duty(1023)
    elif y < 1000:     #down
        rgb_b.duty(1023)
        rgb_r.duty(0)
        rgb_g.duty(0)
    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 “Run current script”, the code starts to be executed and you’ll see that

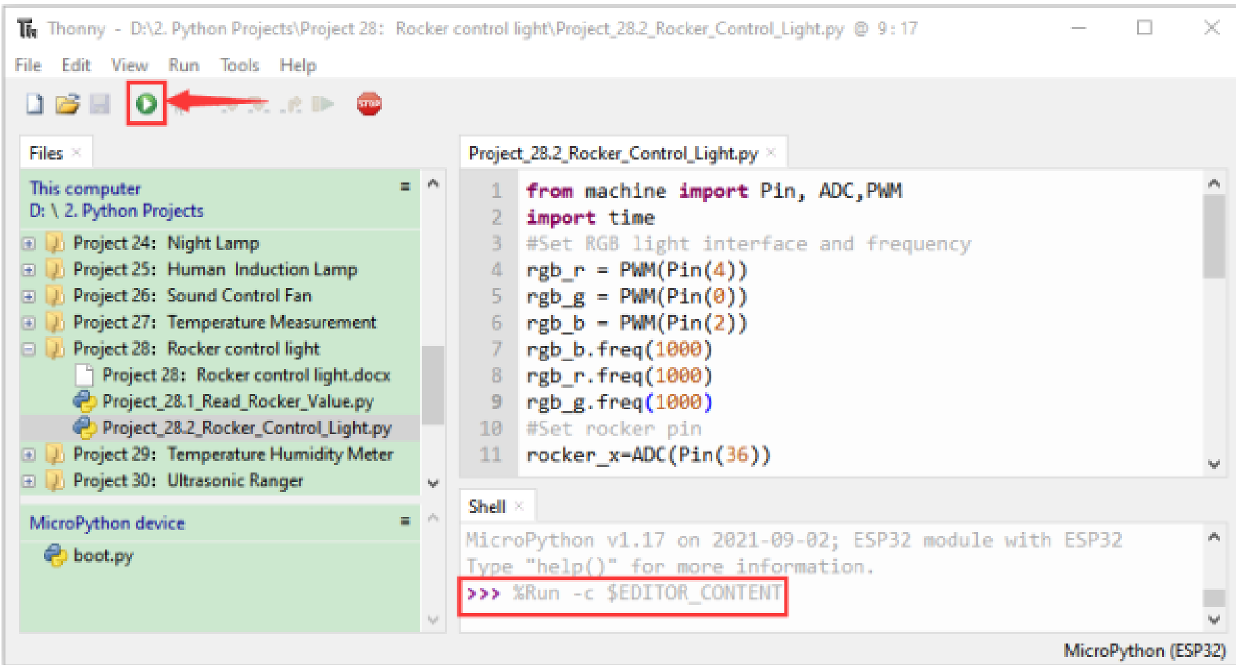
If the rocker is moved to the far left in the X direction, the RGB light turns red.

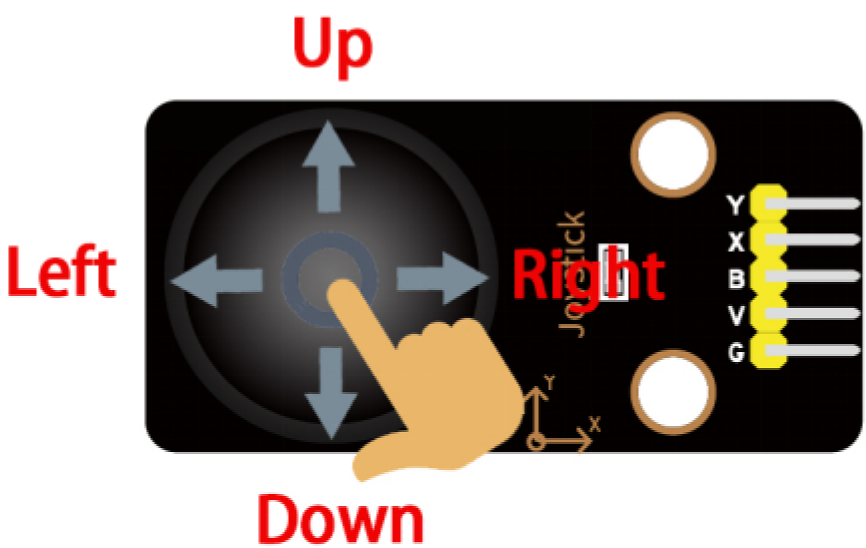
If the rocker is moved to the far right in the X direction, the RGB light turns green.

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.

Press“Ctrl+C”or click “Stop/Restart backend”to exit the program.



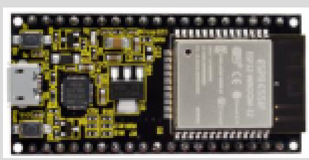
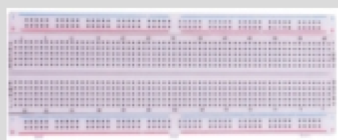
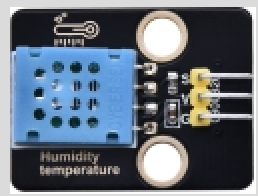
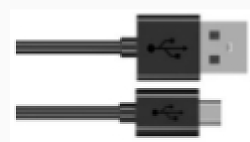


7.30 Project 29Temperature Humidity Meter

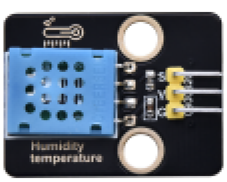
7.30.1 1.Introduction

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

		
ESP32*1	Breadboard*1	Temperature and Humidity Sensor*1
		
LCD 128X32 DOT*1	M-F Dupont Wires	USB Cable*1

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±5%RH, temperature is±2℃, range humidity is 20 to 90%RH, and temperature is 0 to 50℃. 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.

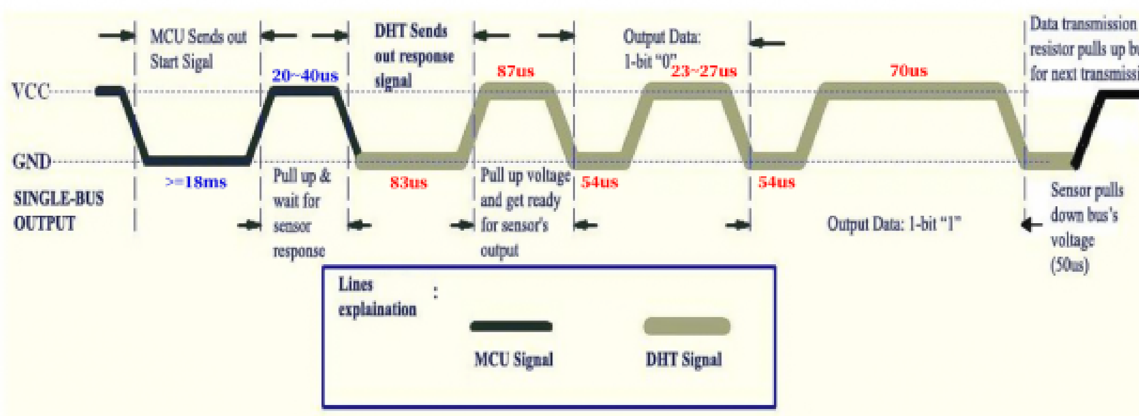
Single bus format definition of Temperature and Humidity Sensor

Description	Definition
Start signal	Microprocessor pulls data bus (SDA) down at least 18ms for a period of time(Maximum is 30ms), notifying the sensor to prepare data.
Response signal	The sensor pulls the data bus (SDA) low for 83µs, and then pulls up for 87µs to respond to the host's start signal.
Humidity	The high humidity is an integer part of the humidity data, and the low humidity is a fractional part of the humidity data.
Temperature	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:



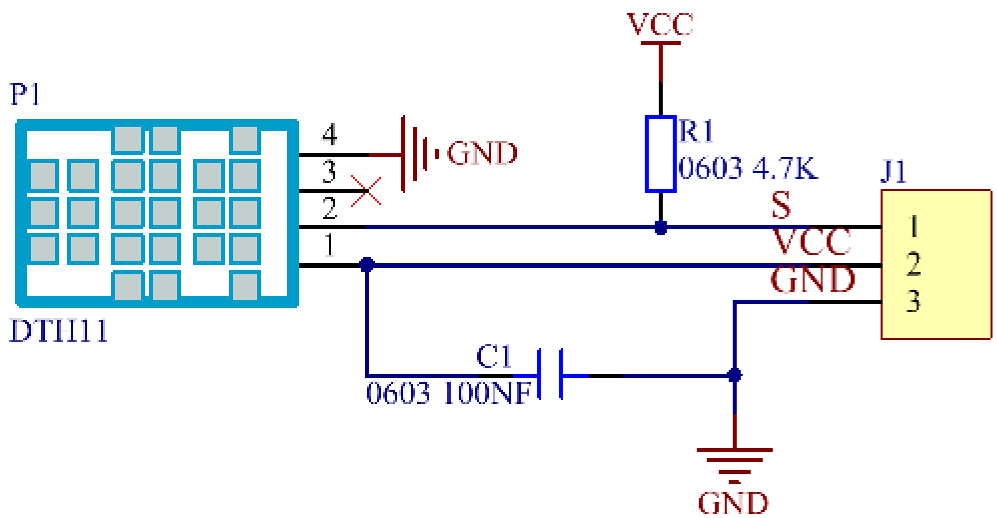
Combined with the code, you can understand better.

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.

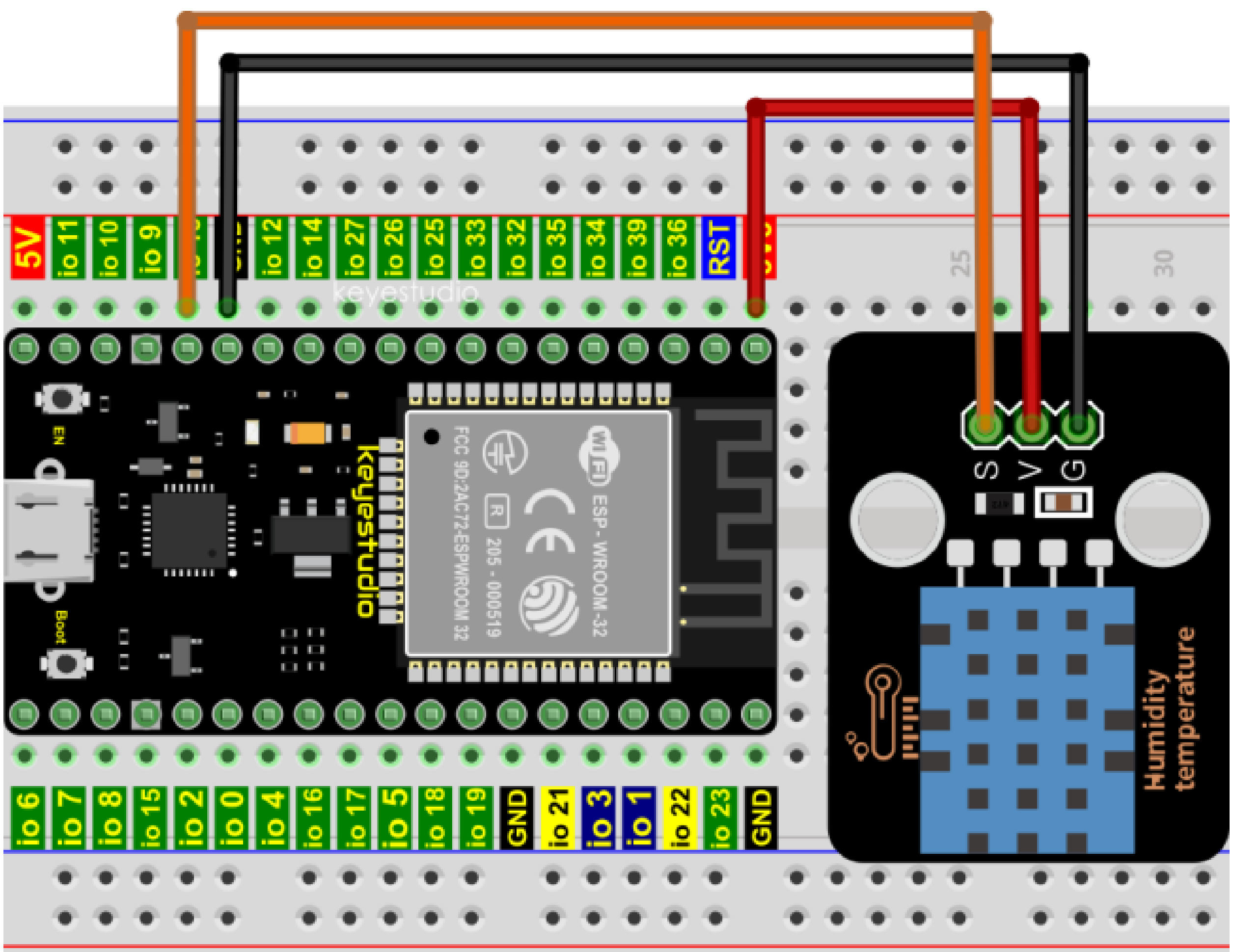
Specification:

- Working voltage: +5V
- Temperature range: 0°C to 50°C, error of ± 2°C
- Humidity range: 20% to 90% RH,± 5% RH error
- Interface: Digital interface

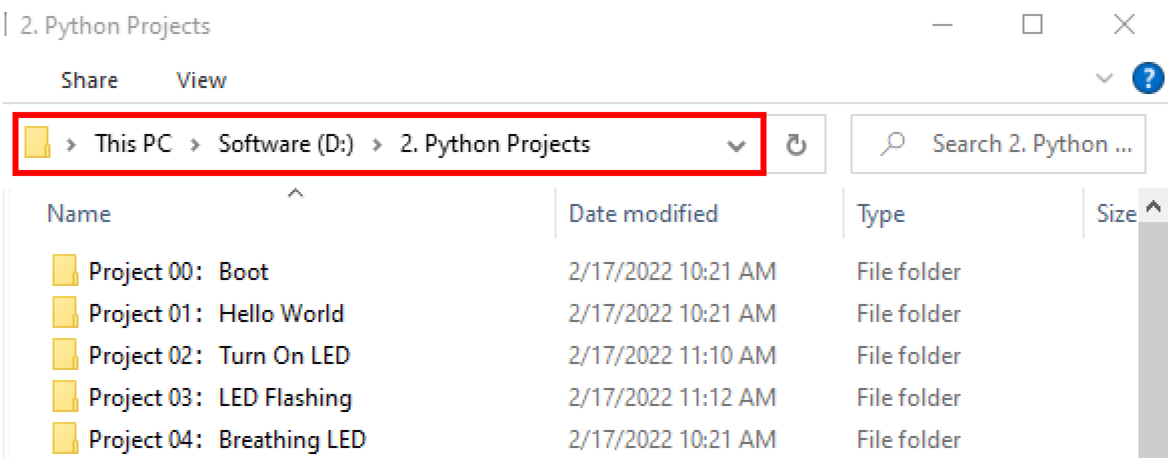
Schematic diagram of Temperature and Humidity Sensor:



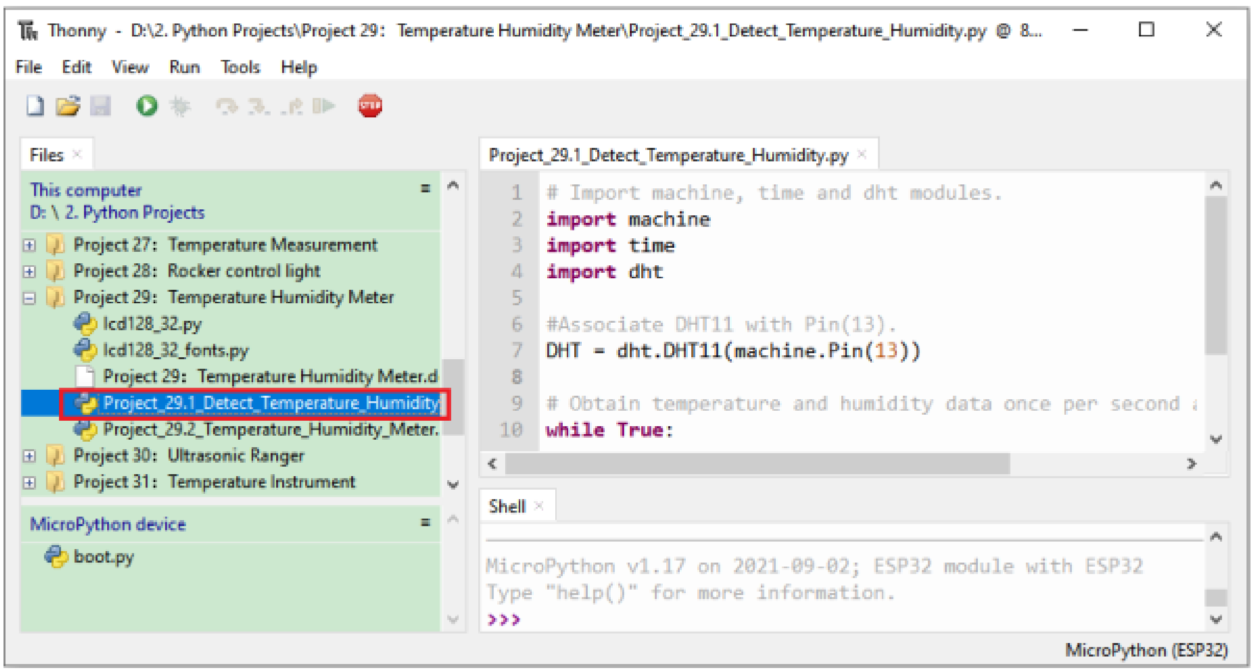
7.30.4 4.Read temperature and humidity value



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



Open"Thonny"click"This computer"→"D:"→"2. Python Projects"→"Project 29Temperature Humidity Meter", and then double left-click "Project_29.1_Detect_Temperature_Humidity.py".



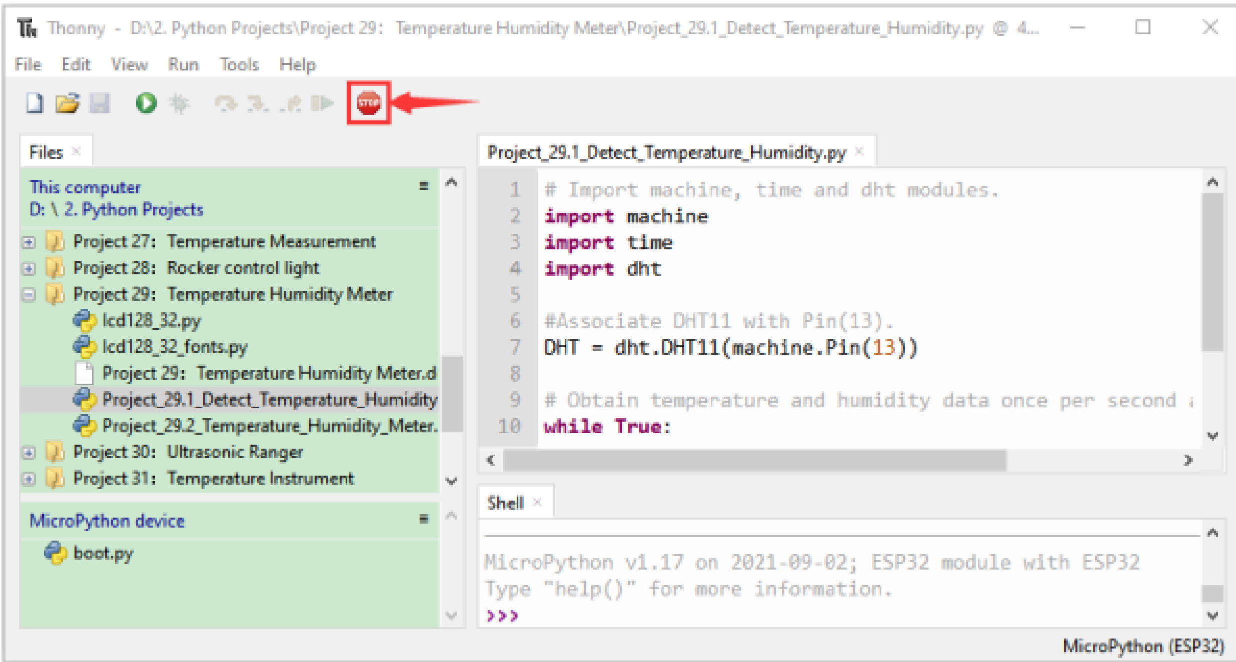

```
## Import machine, time and dht modules.
import machine
import time
import dht

#Associate DHT11 with Pin(13).
DHT = dht.DHT11(machine.Pin(13))

## Obtain temperature and humidity data once per second and print them out.
while True:
    DHT.measure() # Start DHT11 to measure data once.
   # Call the built-in function of DHT to obtain temperature
   # and humidity data and print them in "Shell".
    print('temperature:',DHT.temperature(),'℃','humidity:',DHT.humidity(),'%')
```

(continues on next page)

(continued from previous page)

```
    time.sleep_ms(1000)
```

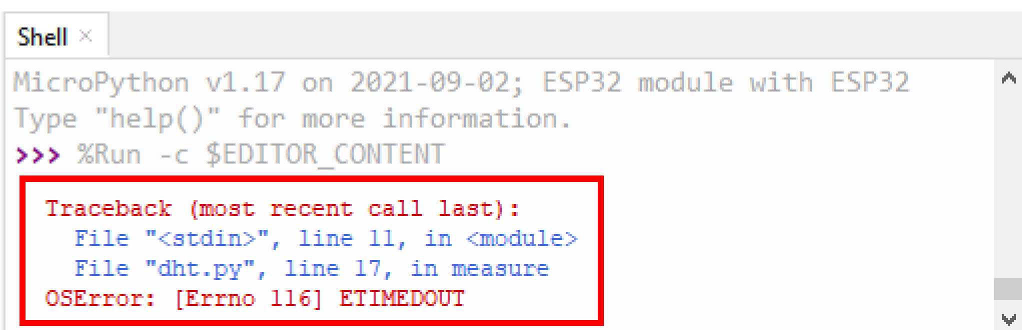
Make sure the ESP32 has been connected to the computer, click  "Stop/Restart backend" .

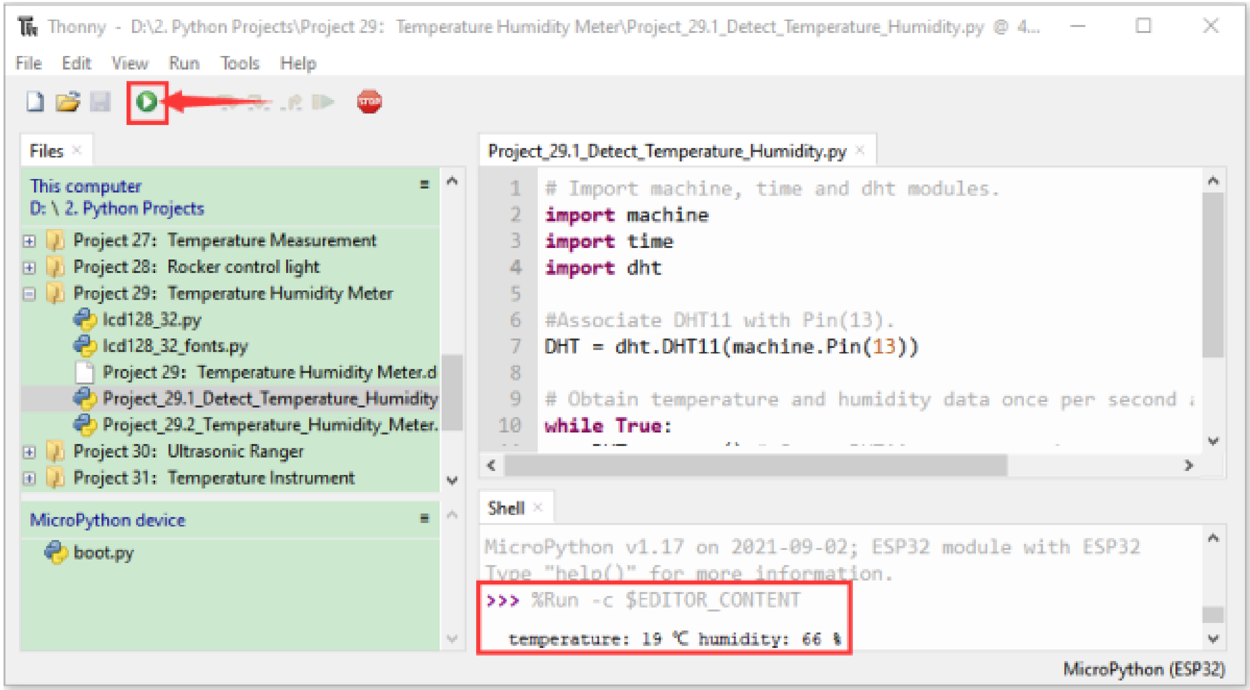


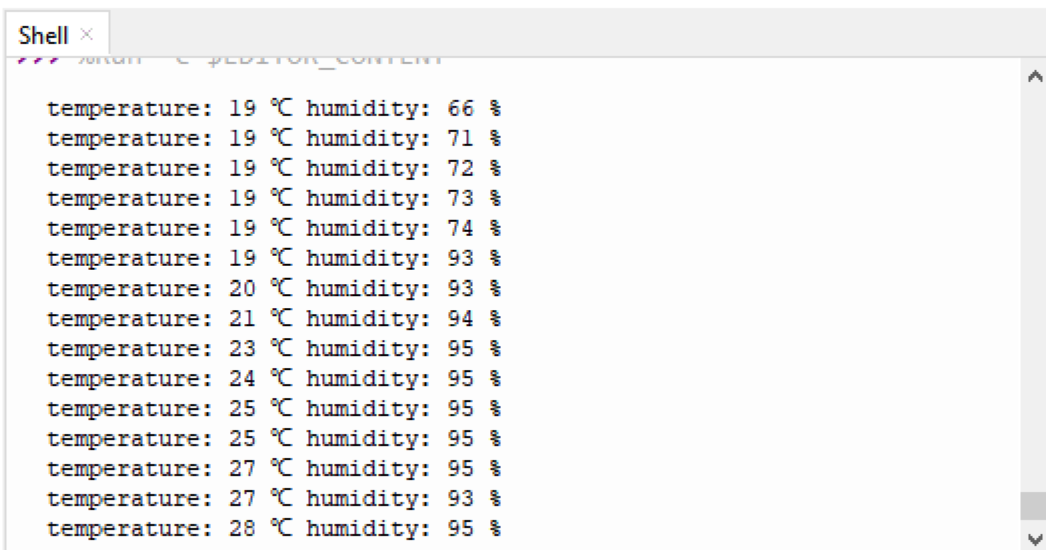
Click  "Run current script", if the temperature and humidity sensor is incorrectly connected, the following information is displayed in the "Shell"window. Please make sure your circuit is properly connected.

Click  again, 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  "Stop/Restart backend" to exit the program.

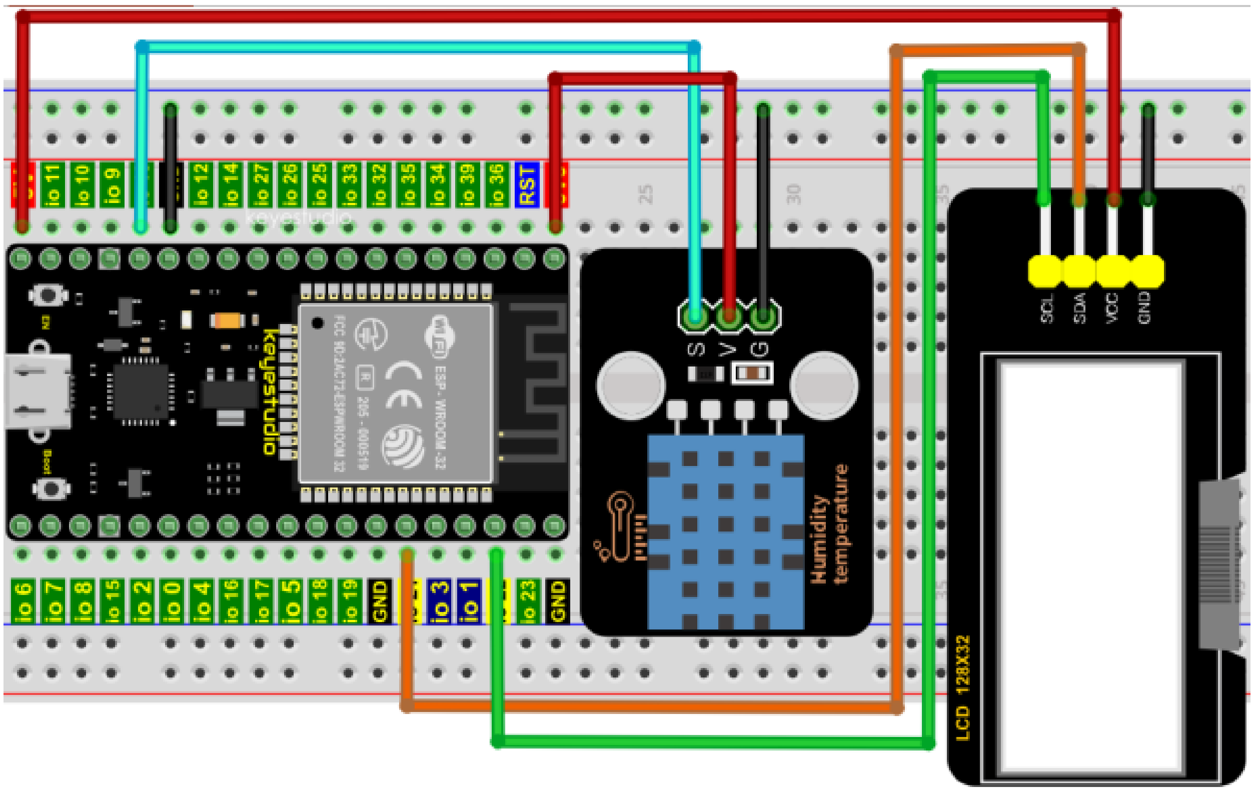






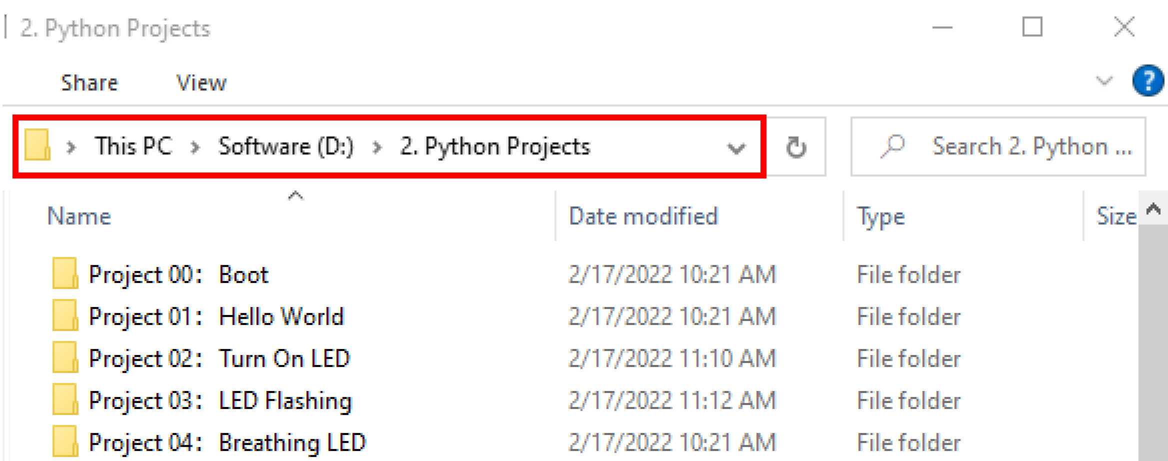
7.30.5 5.Wiring diagram of the thermohygrometer

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

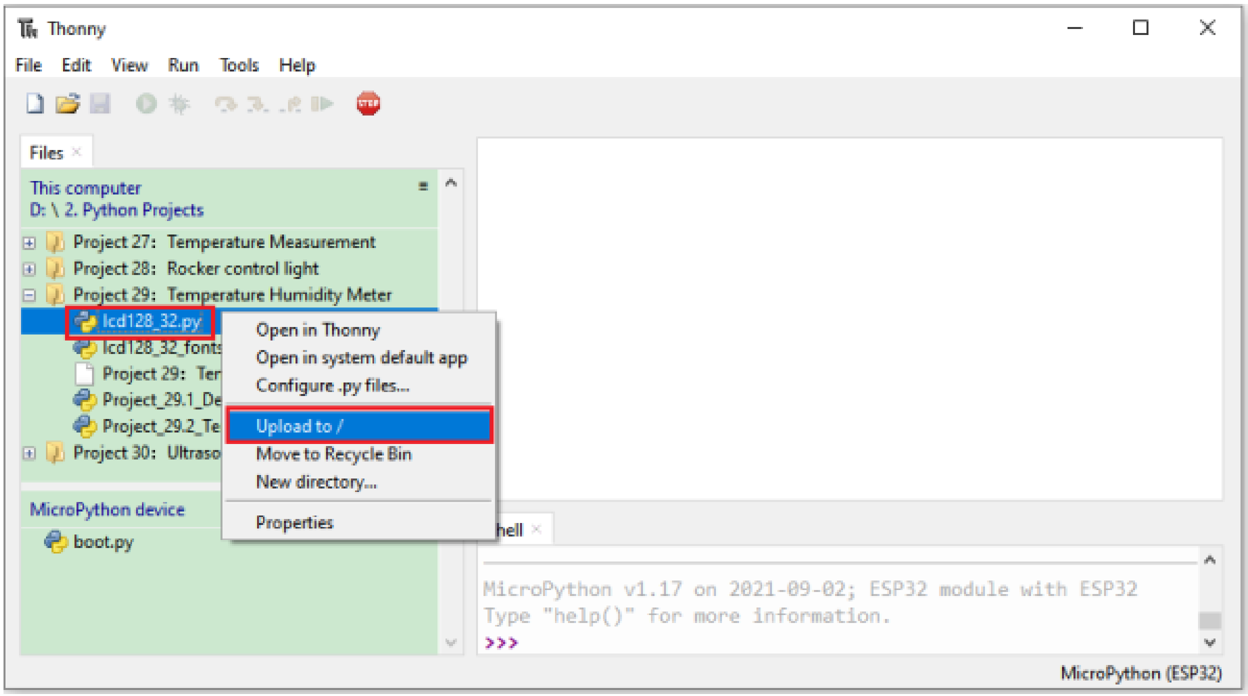


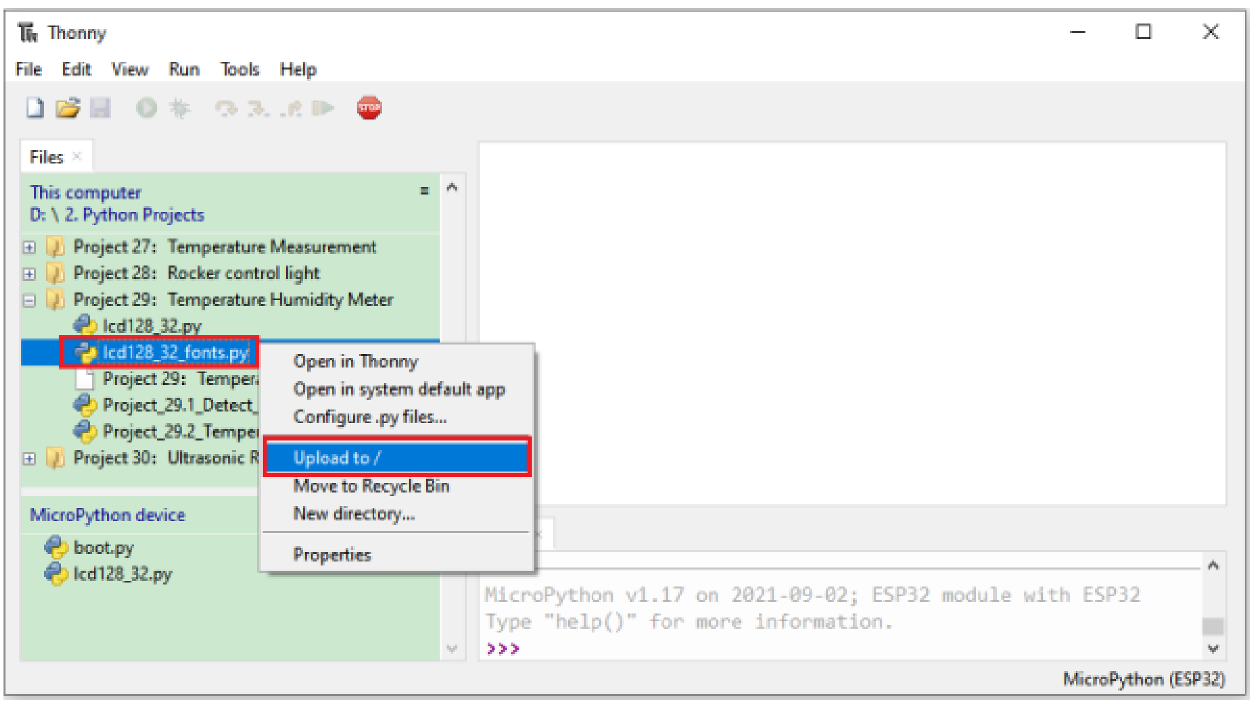
7.30.6 6.Project code

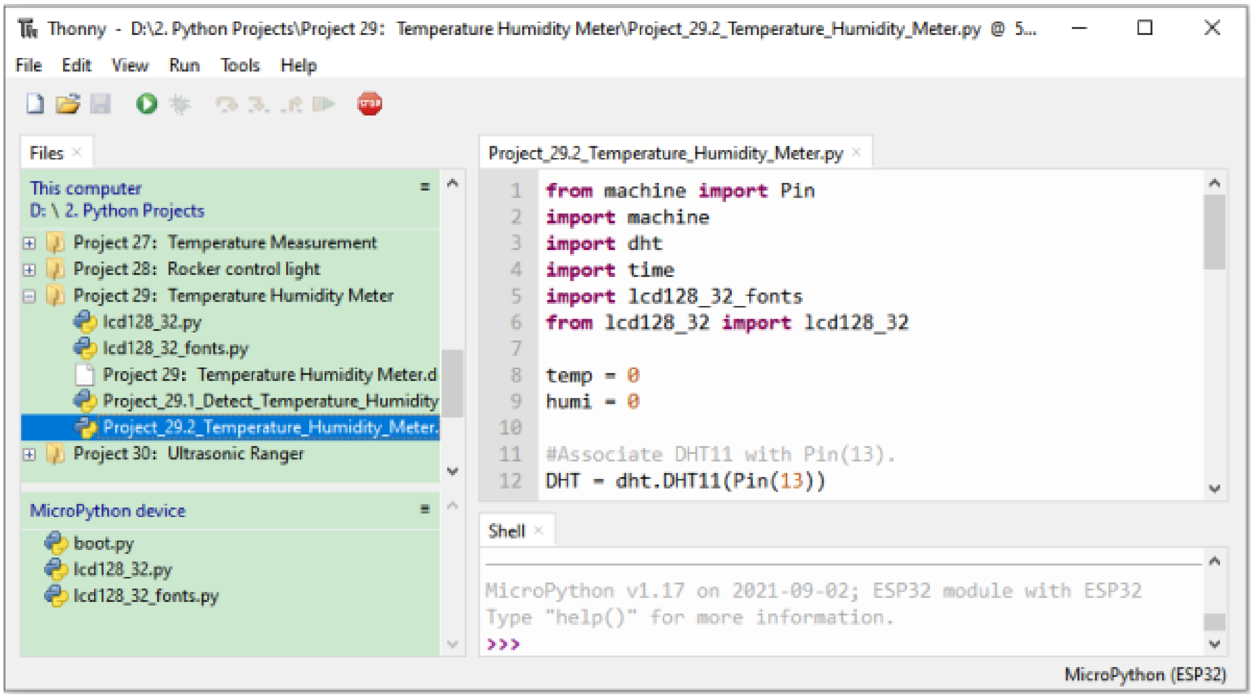
Codes used in this tutorial are saved in"**2. Python Projects**". If you haven't downloaded the code file, please click on the link to download it:Download Python Codes



Open"Thonny"click"This computer"→"D:"→"2. Python Projects"→"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".


Thonny
File Edit View Run Tools Help
Files
This computer
D: \ 2. Python Projects
Project 27: Temperature Measurement
Project 28: Rocker control light
Project 29: Temperature Humidity Meter
lcd128_32.py
Open in Thonny
Open in system default app
Configure .py files...
Upload to /
Move to Recycle Bin
New directory...
Properties
MicroPython device
boot.py
MicroPython v1.17 on 2021-09-02; ESP32 module with ESP32
Type "help()" for more information.
>>>
MicroPython (ESP32)


Thonny
File Edit View Run Tools Help
Files
This computer
D: \ 2. Python Projects
Project 27: Temperature Measurement
Project 28: Rocker control light
Project 29: Temperature Humidity Meter
lcd128_32.py
lcd128_32_fonts.py
Open in Thonny
Open in system default app
Configure .py files...
Upload to /
Move to Recycle Bin
New directory...
Properties
MicroPython device
boot.py
lcd128_32.py
MicroPython v1.17 on 2021-09-02; ESP32 module with ESP32
Type "help()" for more information.
>>>
MicroPython (ESP32)




```
from machine import Pin
import machine
import dht
import time
import lcd128_32_fonts
from lcd128_32 import lcd128_32

temp = 0
humi = 0

#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:
```

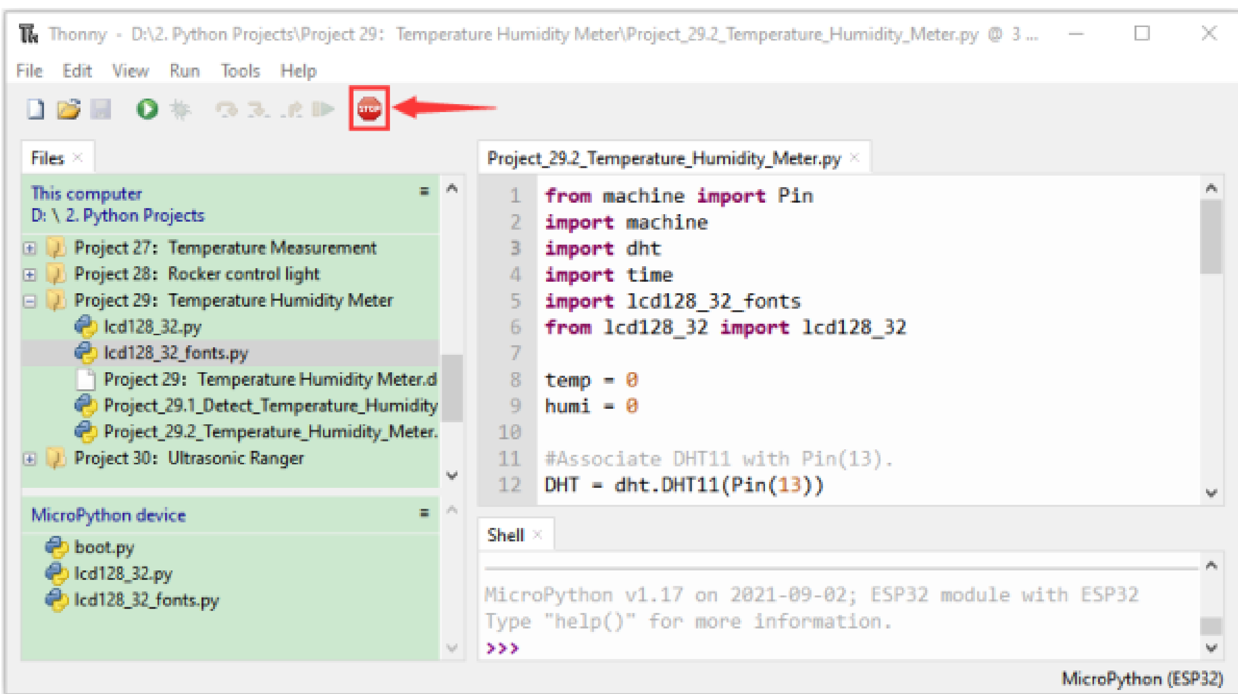
(continues on next page)

(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(0, 0)
        lcd.Display("temper:")
        lcd.Cursor(0, 8)
        lcd.Display(str(temp))
        lcd.Cursor(0, 11)
        lcd.Display("C")
        lcd.Cursor(2, 0)
        lcd.Display("Humid:")
        lcd.Cursor(2, 7)
        lcd.Display(str(humi))
        lcd.Cursor(2, 10)
        lcd.Display("%")
        time.sleep(0.1)
except:
    pass
```

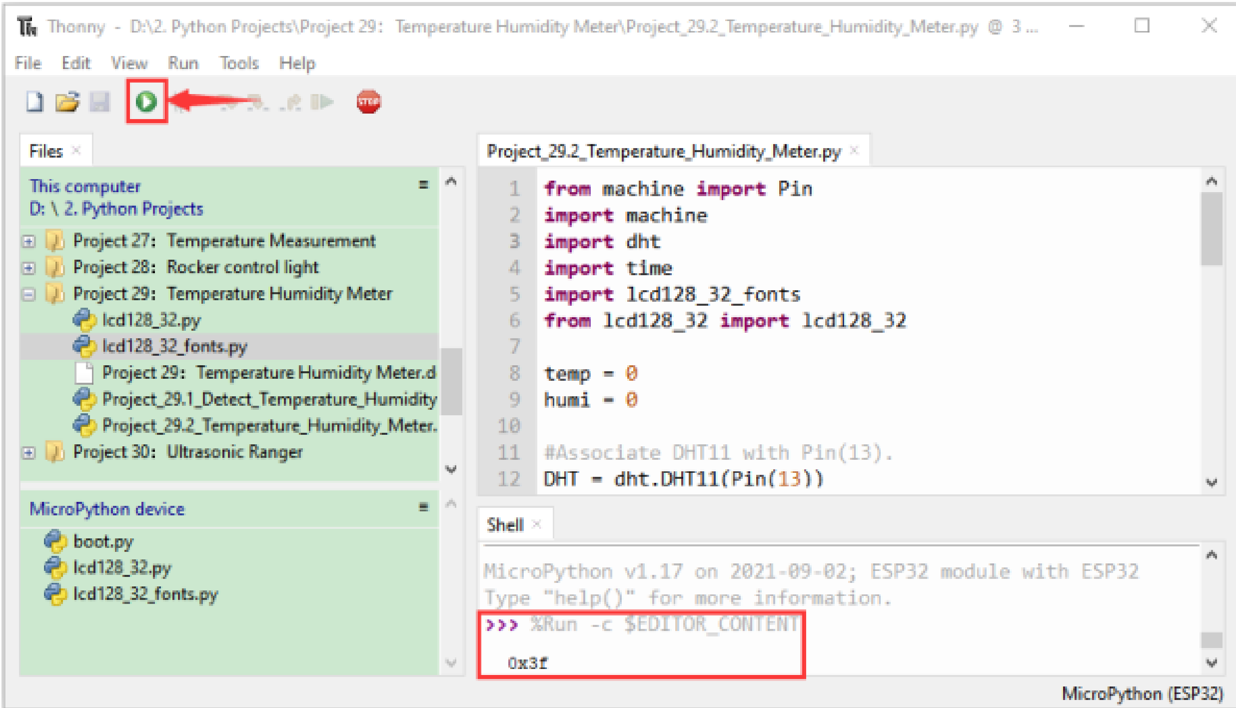
7.30.7 7.Project result

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.



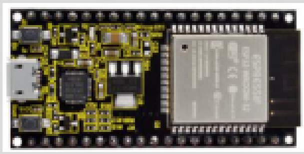
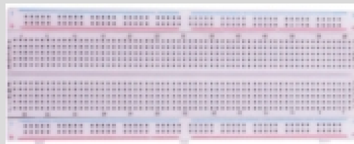
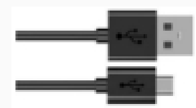
7.31 Project 30Ultrasonic Ranger

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.

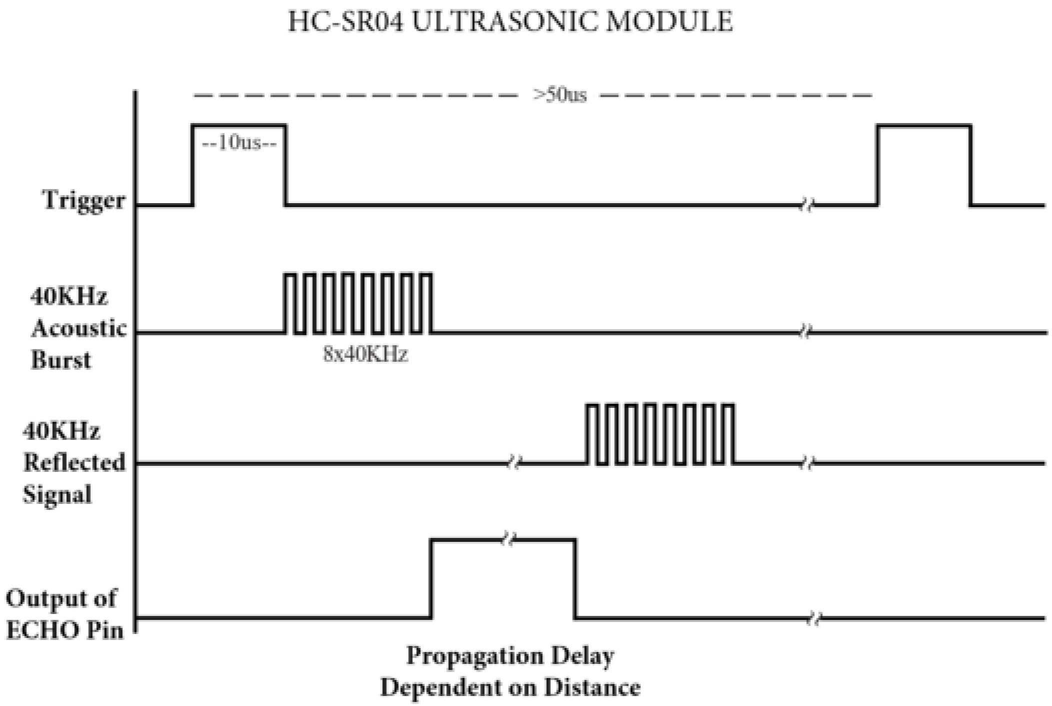
Its operation is not affected by sunlight or black materials, just like a precision camera (acoustically softer materials like cloth are difficult to detect). It has an ultrasonic transmitter and receiver.

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.



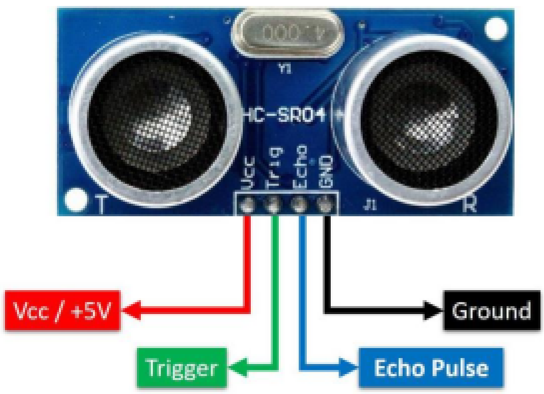
Time = Echo pulse width, its unit is"us" (microseconds)

$$Distance(cm) = time/58$$

$$Distance(inch) = time/148$$

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.

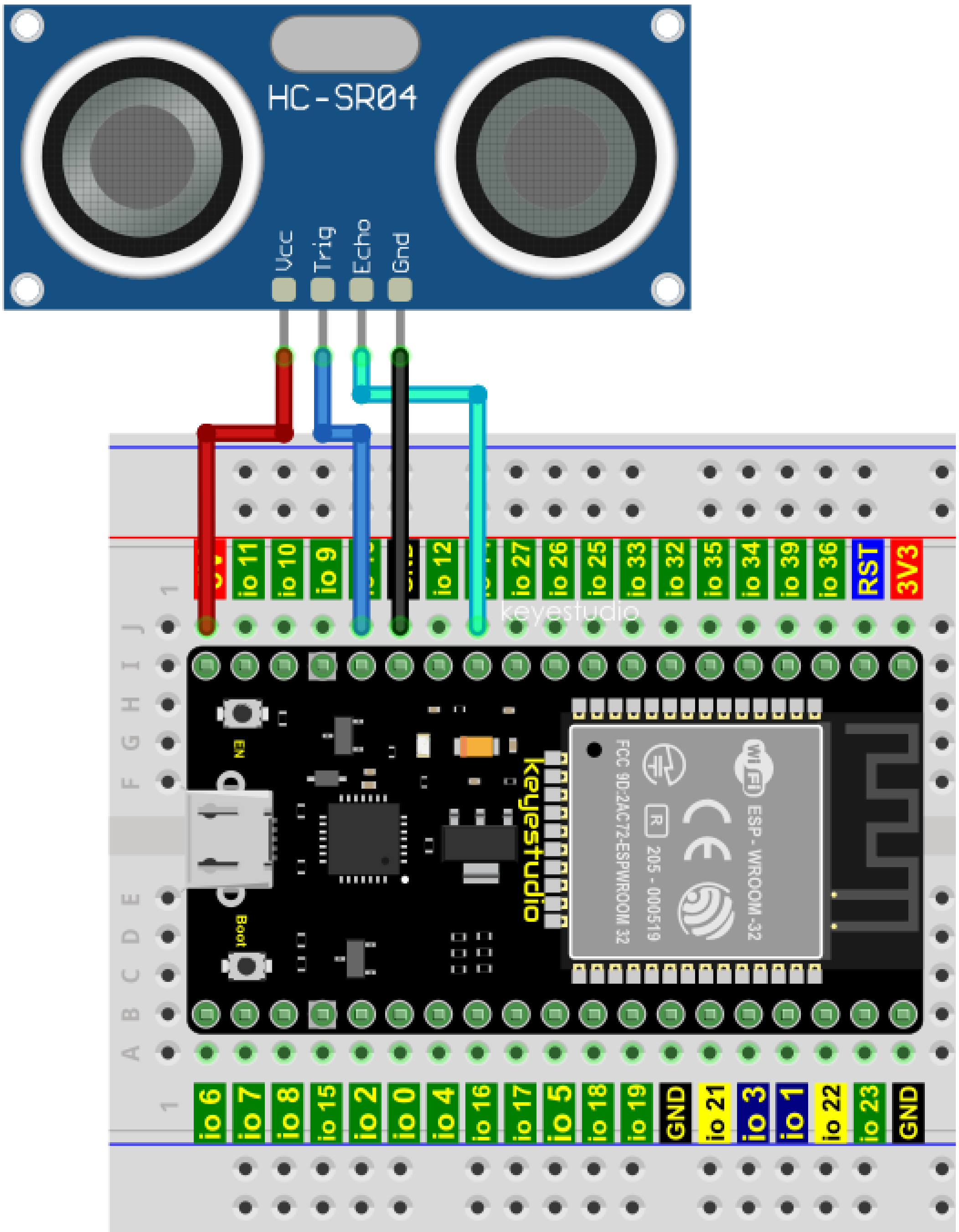
The Vcc pin provides the power source for generating ultrasonic pulses and is connected to Vcc (+5V).

The GND pin is grounded.

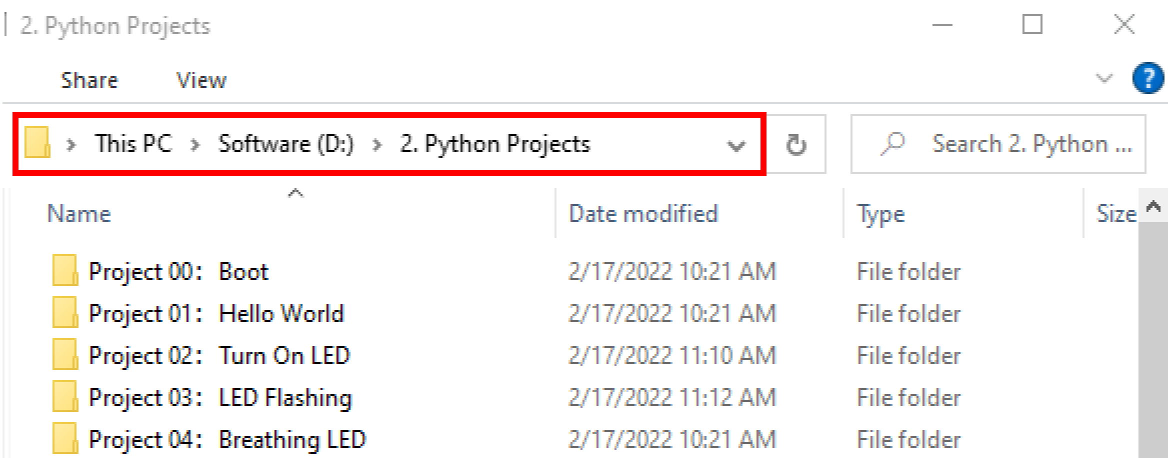
The Trig pin is where the Arduino sends a signal to start the ultrasonic pulse.

The Echo pin is where the ultrasonic sensor sends information about the duration of the ultrasonic pulse to the Control board.

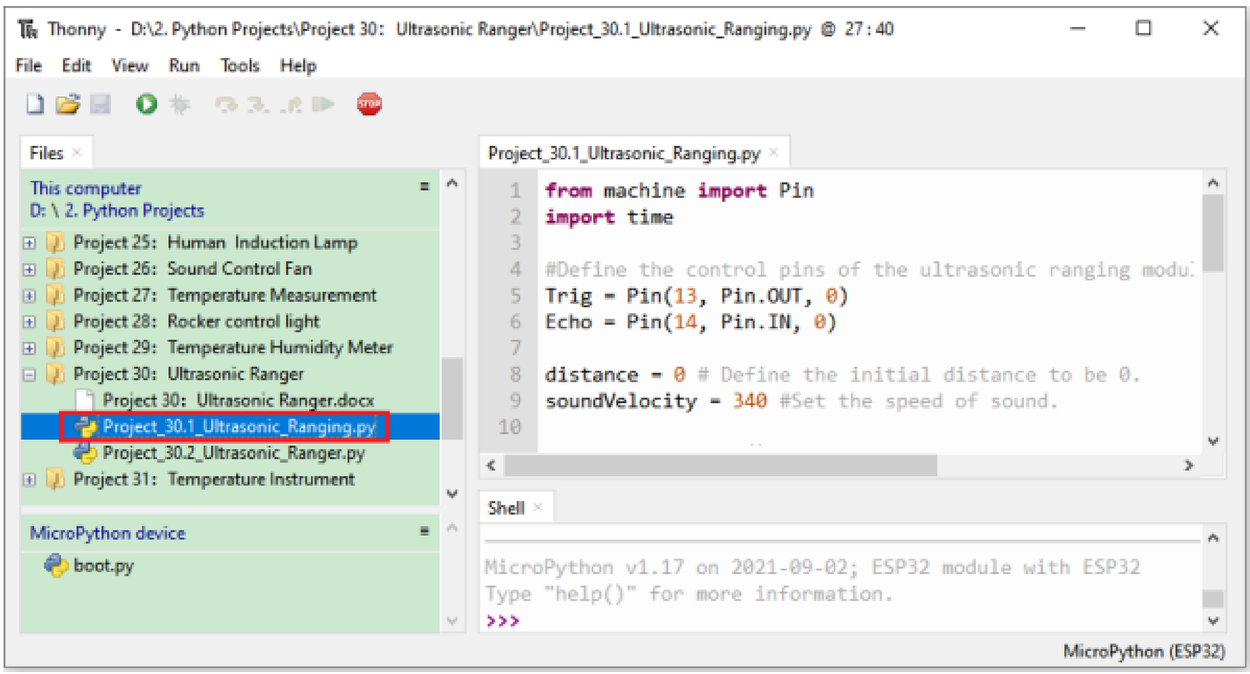
Wiring as shown below:



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



Open“Thonny”click“This computer”→“D:”→“2. Python Projects”→“Project 30Ultrasonic Ranger”, and then double left-click“Project 30.1_Ultrasonic_Ranging.py”.



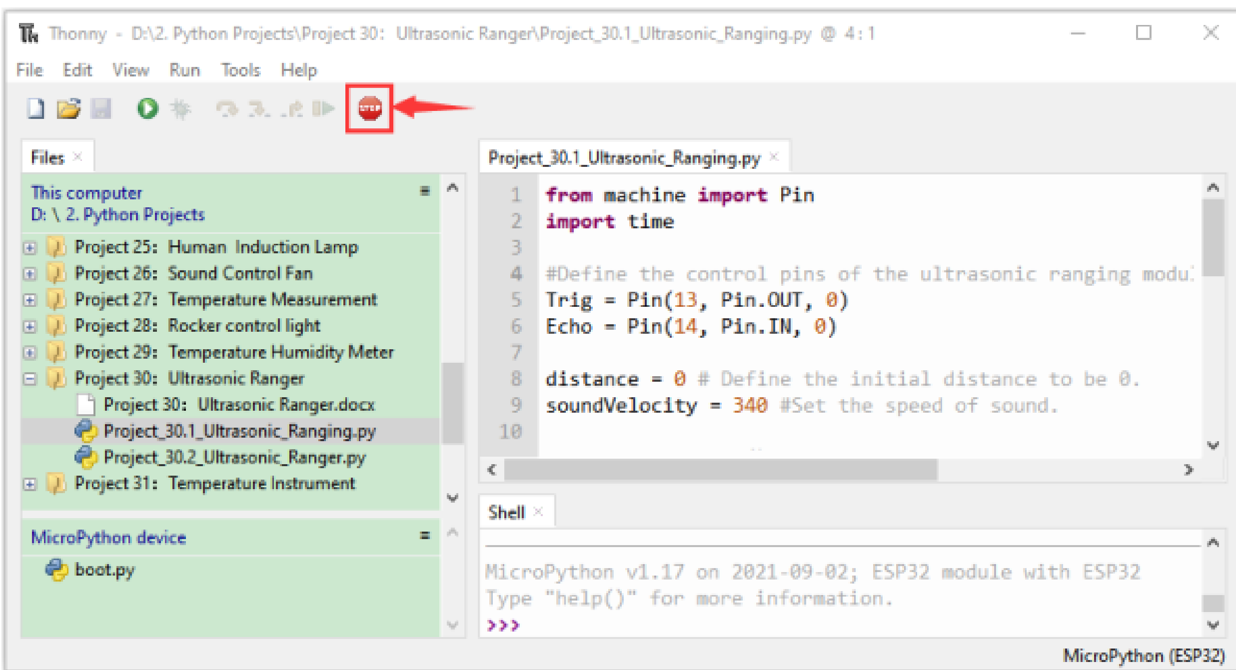

```
from machine import Pin
import time

## 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)

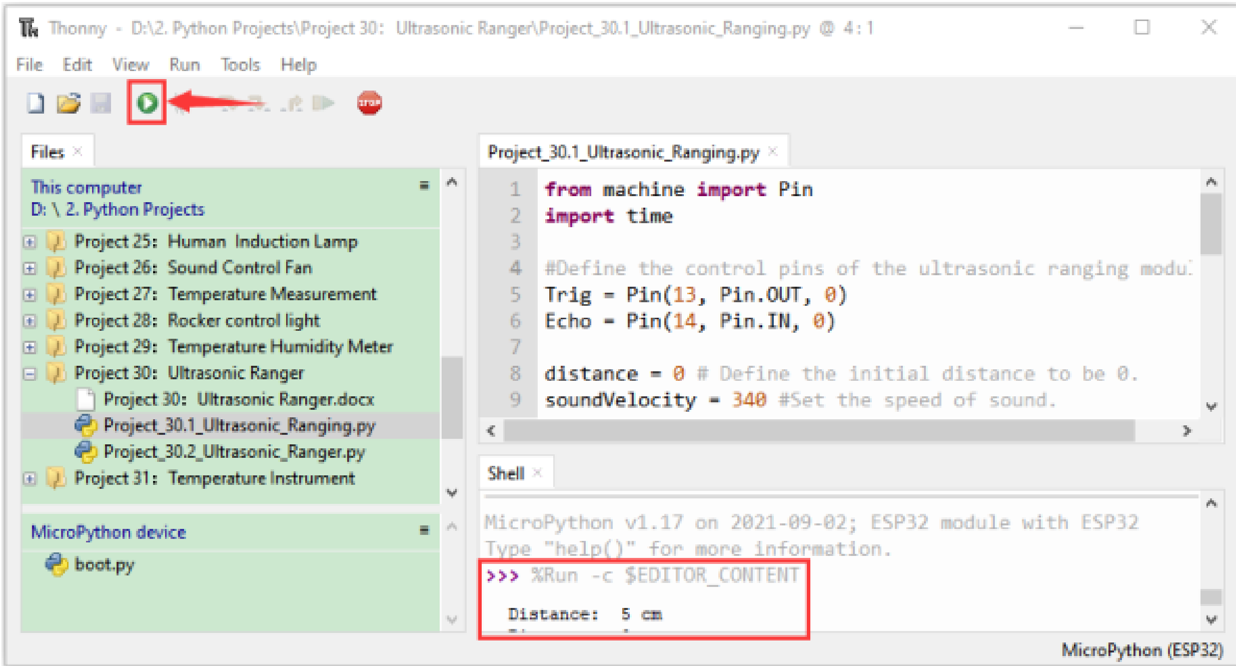
```
## pin's high level,calculate the measured distance based on time and return the value.
def getDistance():
    Trig.value(1)
    time.sleep_us(10)
    Trig.value(0)
    while not Echo.value():
        pass
    pingStart = time.ticks_us()
    while Echo.value():
        pass
    pingStop = time.ticks_us()
    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")
```

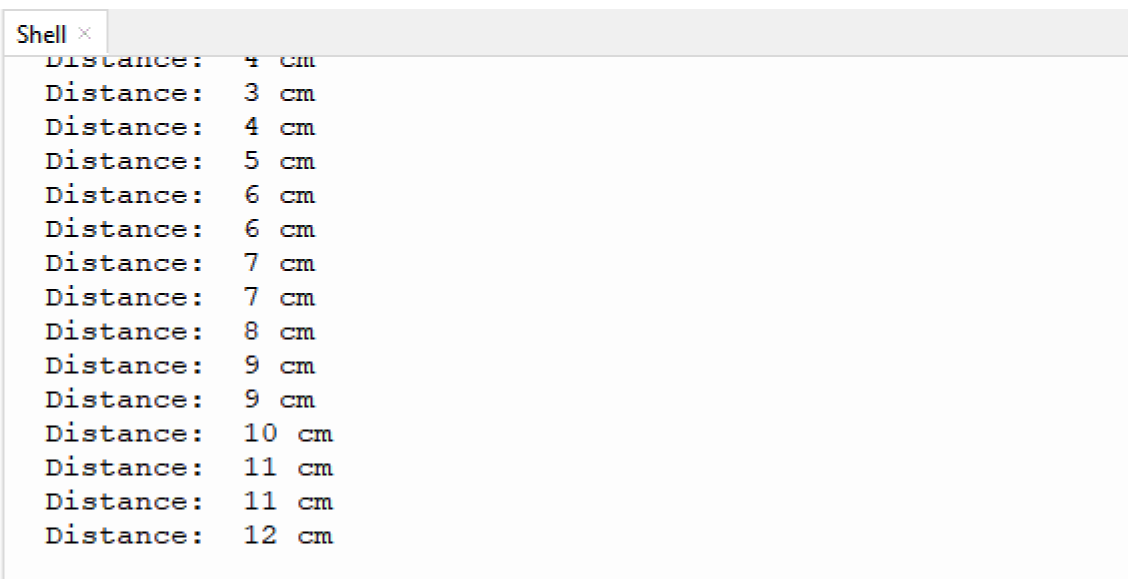
Make sure the ESP32 has been connected to the computer, click "Stop/Restart backend" .



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.

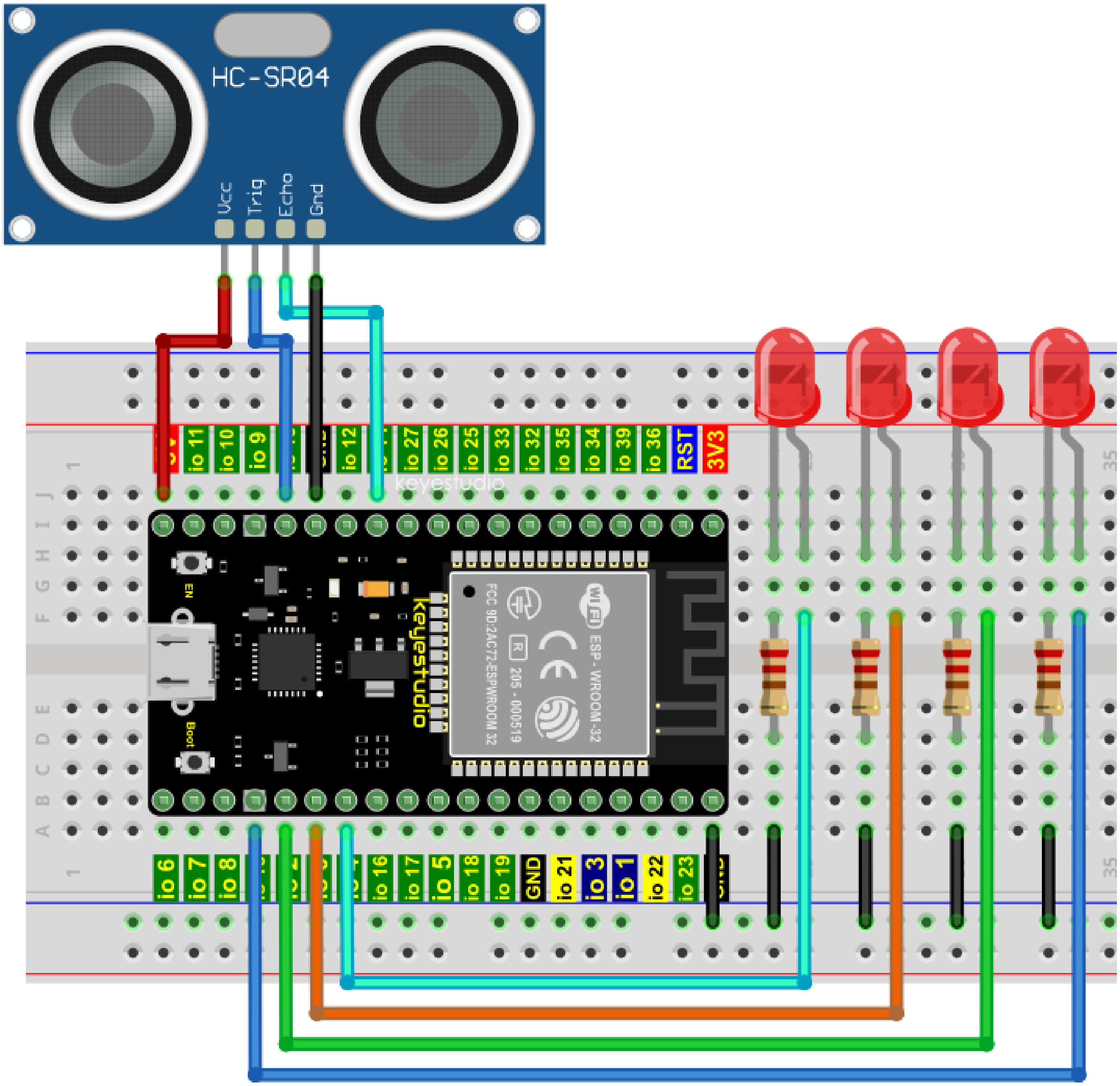
Press"Ctrl+C"or click "Stop/Restart backend"to exit the program.





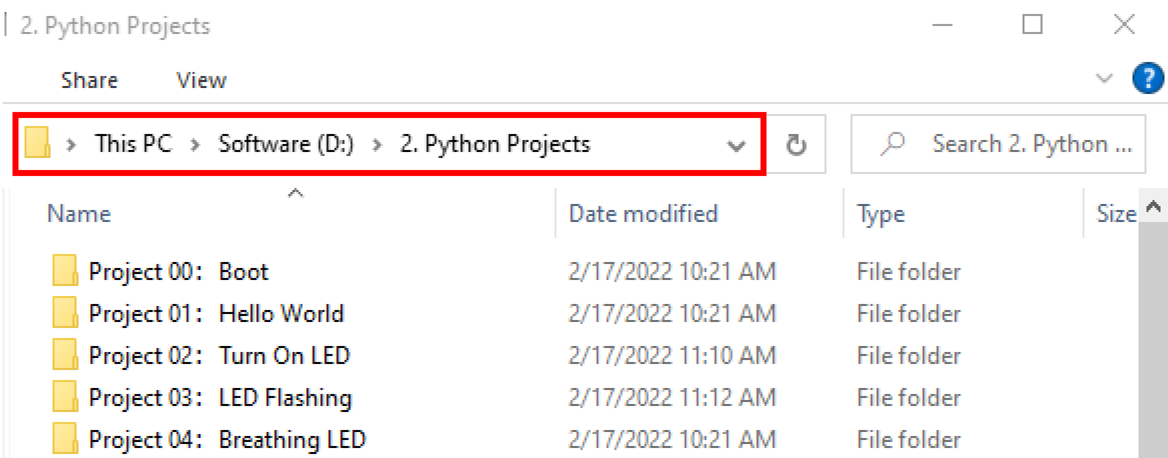
7.31.5 Wiring diagram of the ultrasonic rangefinder

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

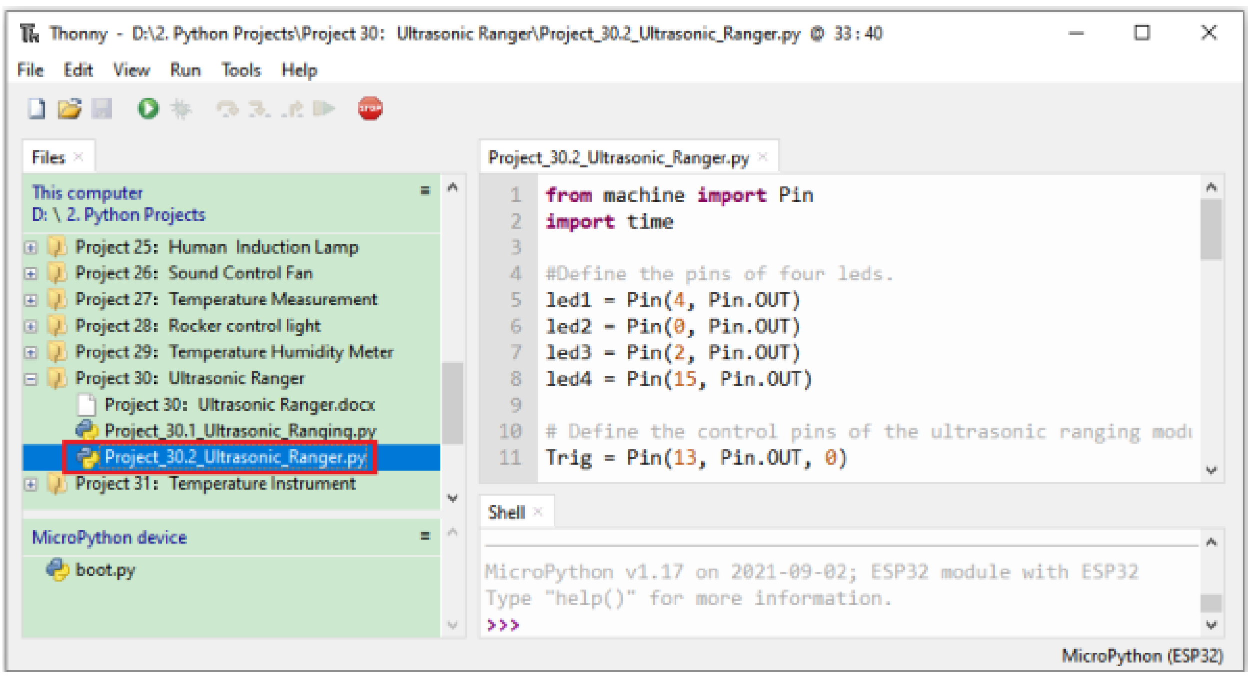


7.31.6 Project code

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



Open"Thonny"click"This computer"→"D:"→"2. Python Projects"→"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.
```

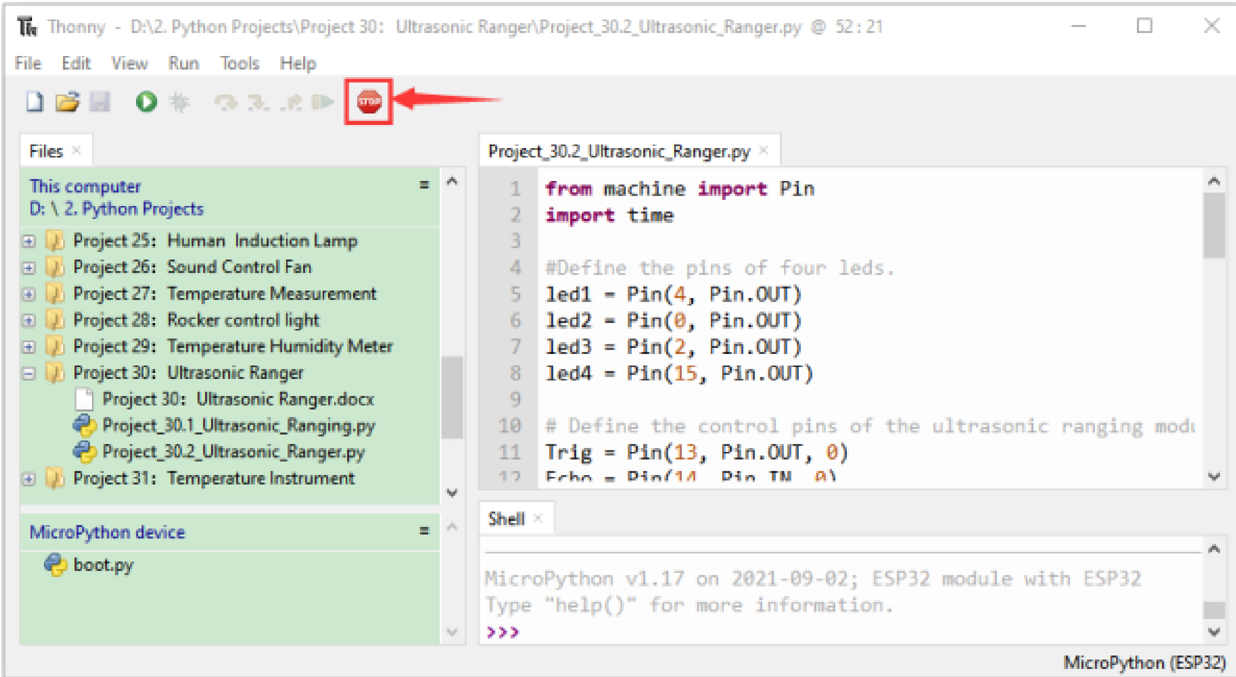
(continues on next page)

(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(0)
    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:
       led1.value(1)
    else:
       led1.value(0)
    if distance <= 10:
       led2.value(1)
    else:
       led2.value(0)
    if distance <= 15:
       led3.value(1)
    else:
       led3.value(0)
    if distance <= 20:
       led4.value(1)
    else:
       led4.value(0)
```

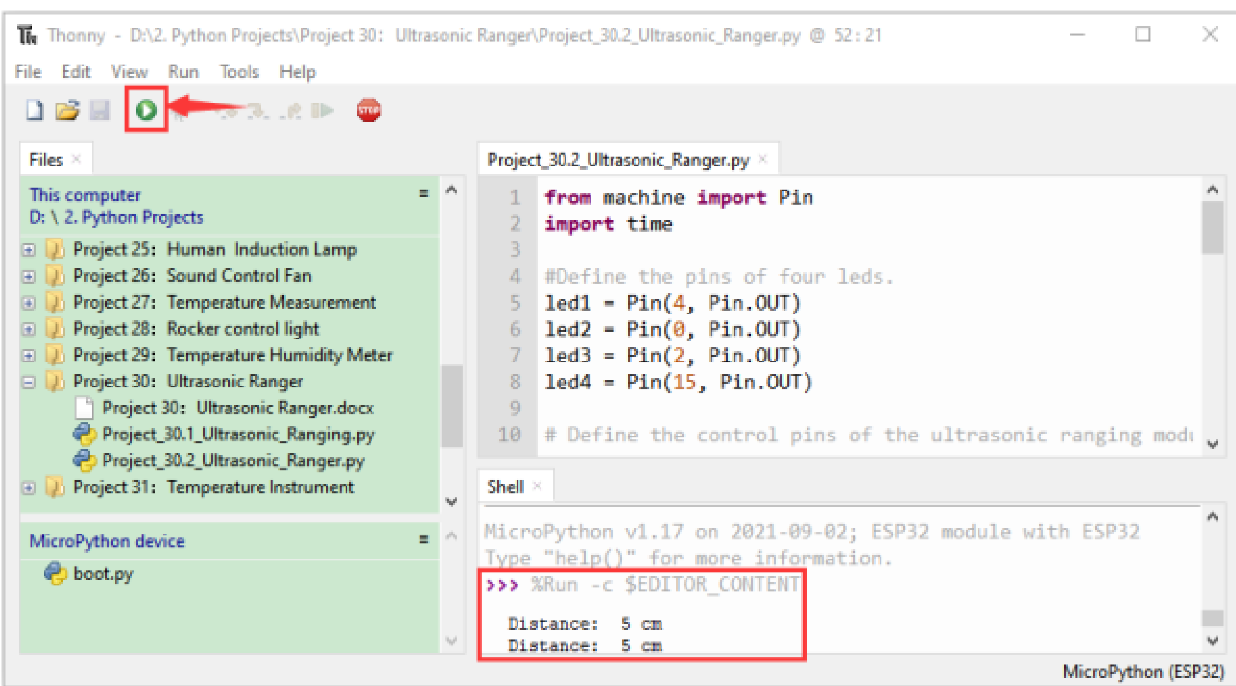
7.31.7 Project result

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.

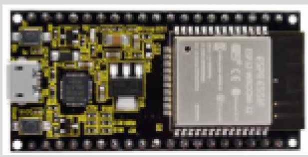
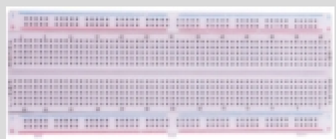
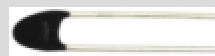
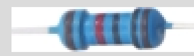
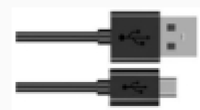


7.32 Project 31Temperature Instrument

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.

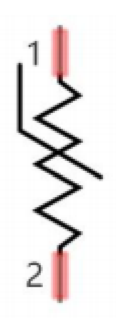
7.32.2 Components

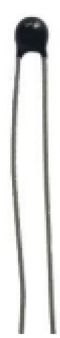
			
ESP32*1	Breadboard*1	Thermistor*1	10KResistor*1
			
M-F Dupont Wires	LCD 128X32 DOT*1	Jumper Wires	USB Cable*1

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;

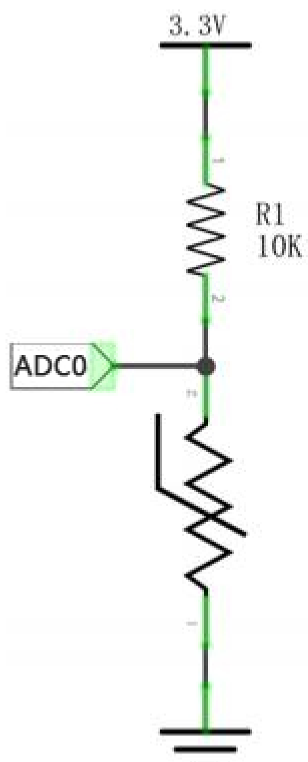
EXP[n] is nth power of e;

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.

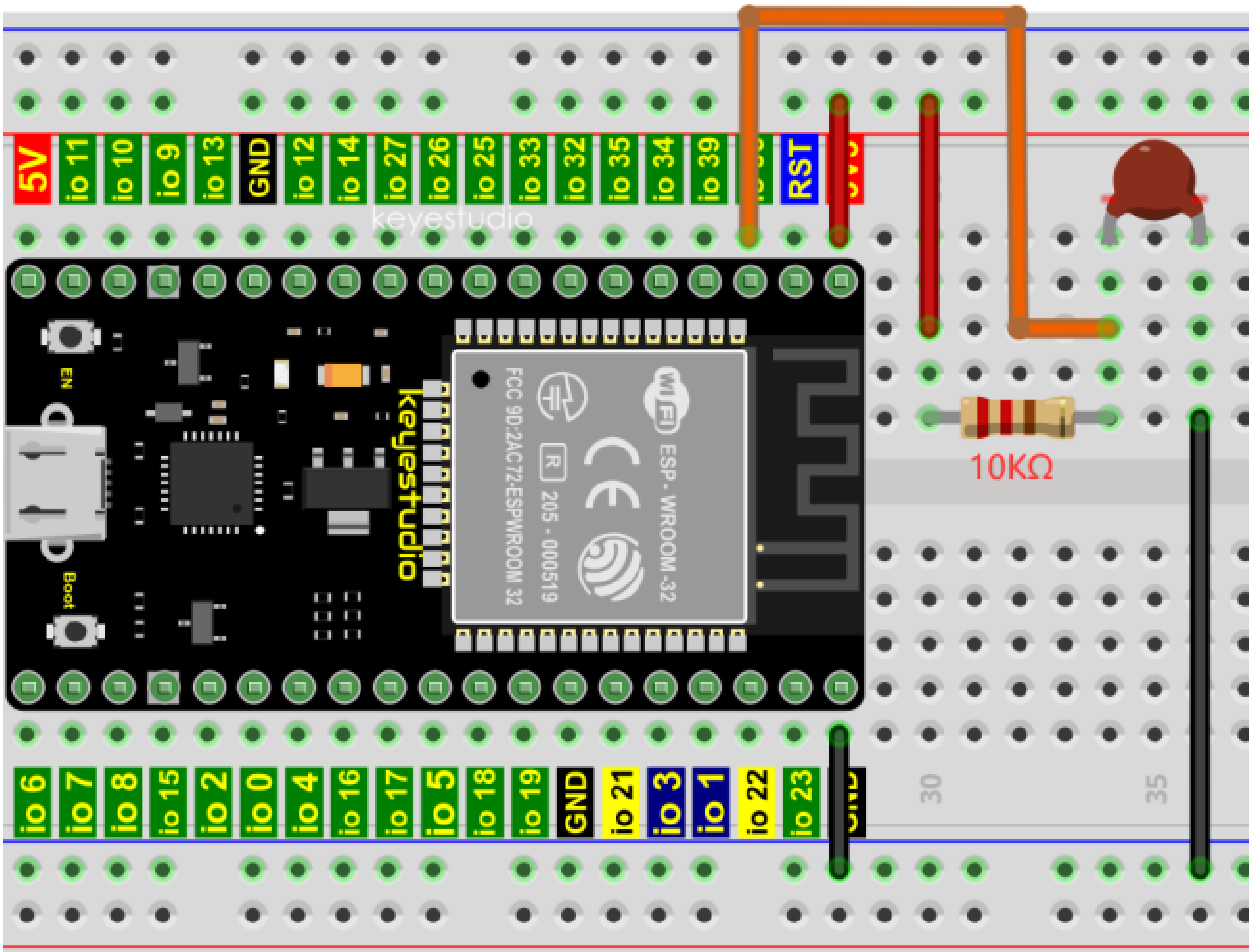
The circuit connection method of the Thermistor is similar to photoresistor, as the following



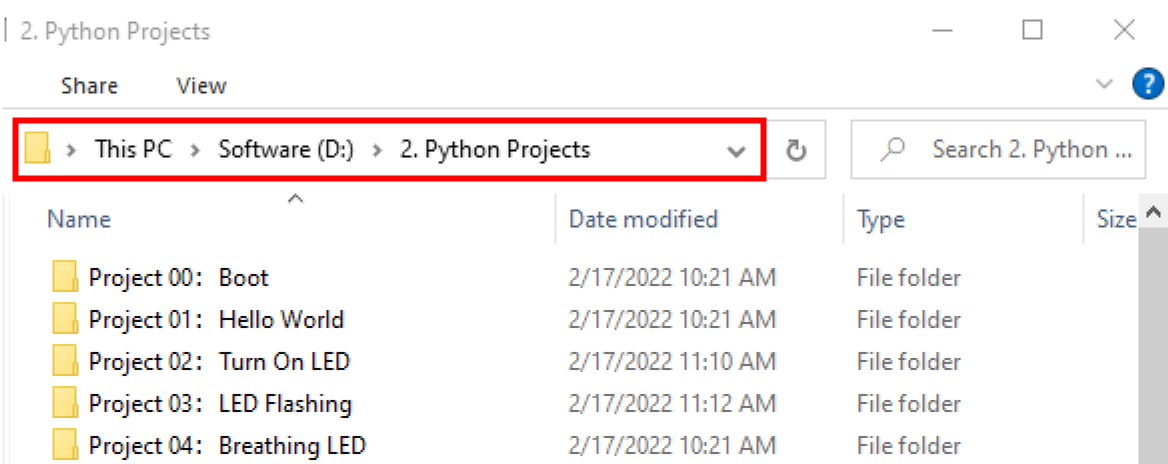
We can use the value measured by the ADC converter to obtain the resistance value of Thermistor, and then we can use the formula to obtain the temperature value. Therefore, the temperature formula can be derived as:

7.32.4 Read the value of the Thermistor

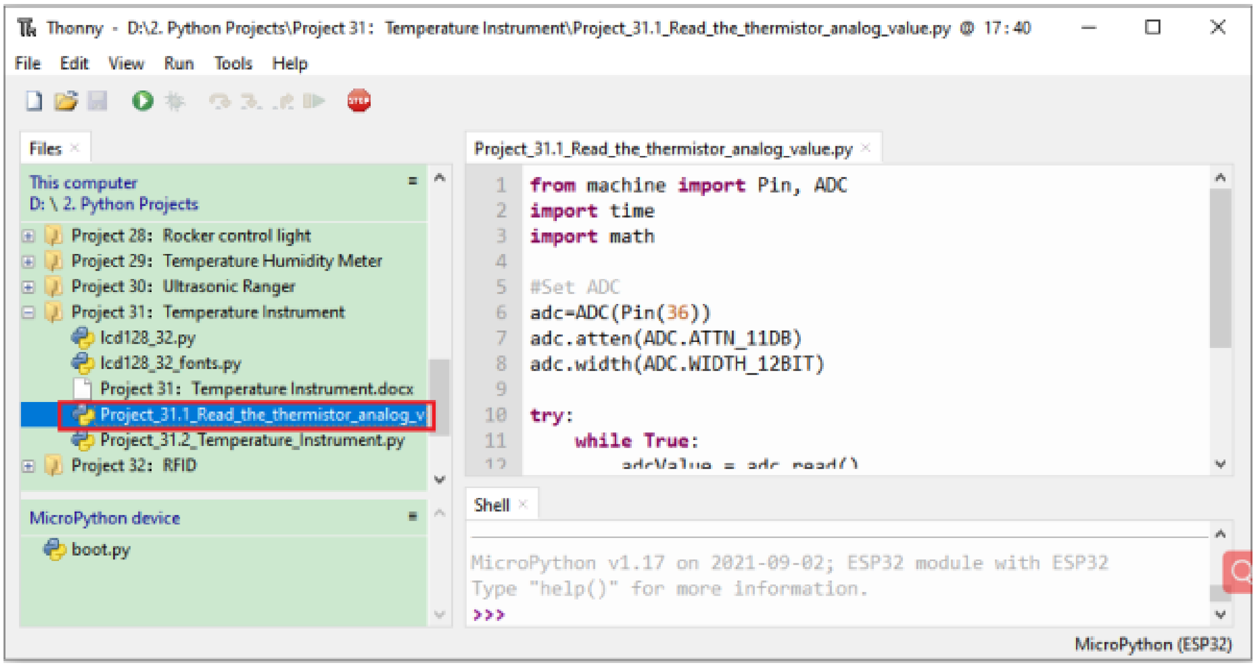
First we will learn the thermistor to read the current ADC value, voltage value and temperature value and print them out. Please connect the wires according to the wiring diagram below



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

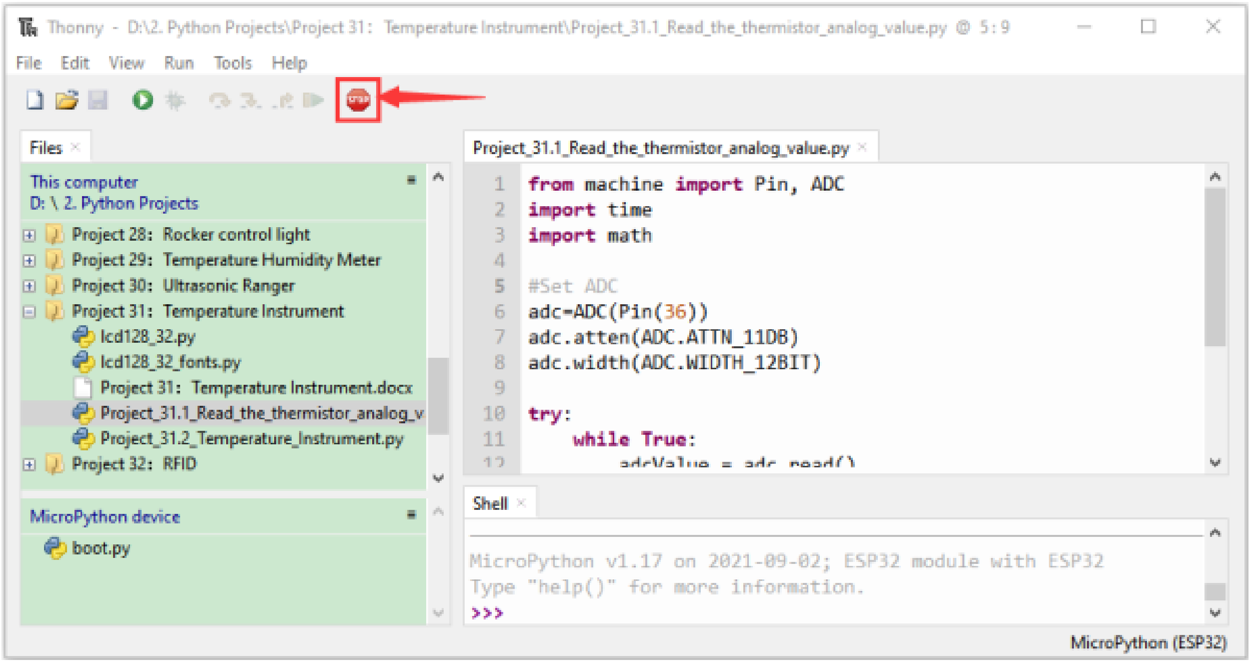


Open"Thonny"click"This computer"→"D:"→"2. Python Projects"→"Project 31Temperature Instrument", and then double left-click "Project_31.1_Read_the_thermistor_analog_value.py".



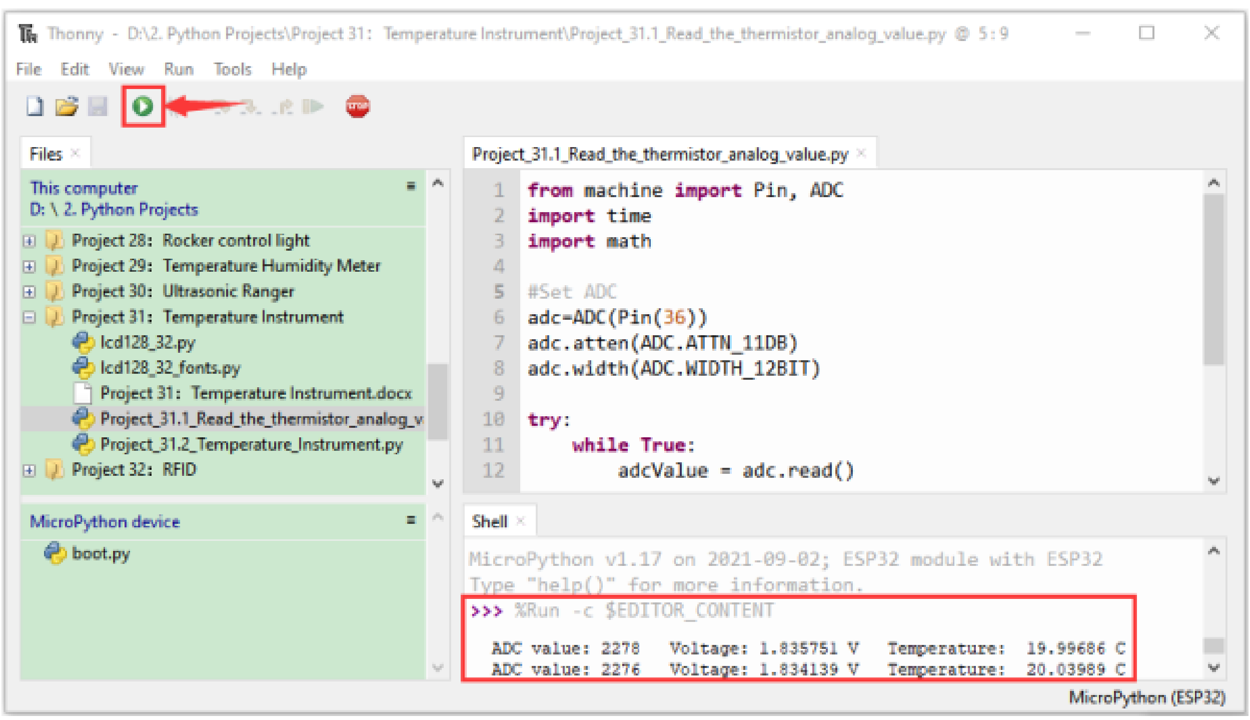

```python
from machine import Pin, ADC
import time
import math

#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 "Stop/Restart backend".



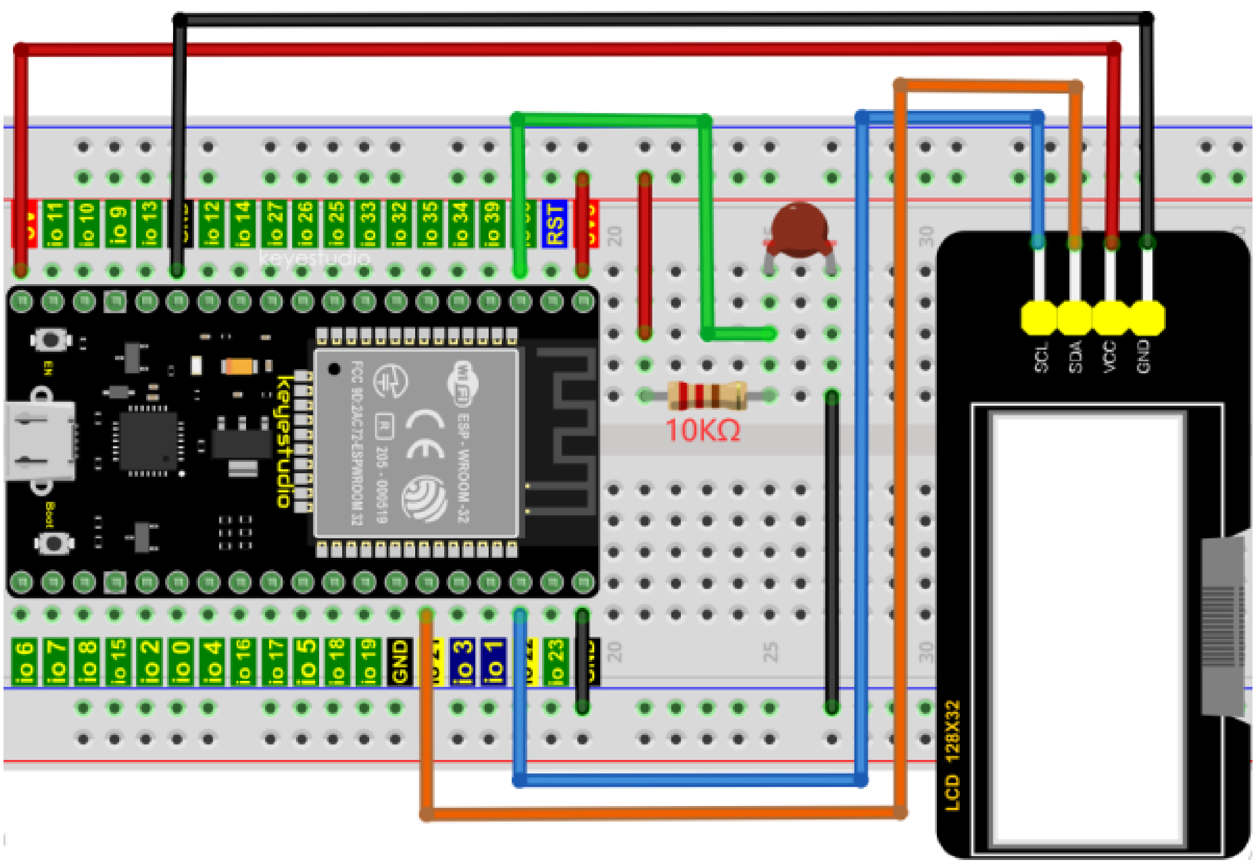
Click "Run current script", the code starts to be executed and you'll see that the "Shell" window of Thonny IDE will continuously display the thermistor's current ADC value, voltage value and temperature value. Try pinching the thermistor with your index finger and thumb (don't touch wires) for a while, and you will see the temperature increase.

Press"Ctrl+C"or click "Stop/Restart backend"to exit the program.



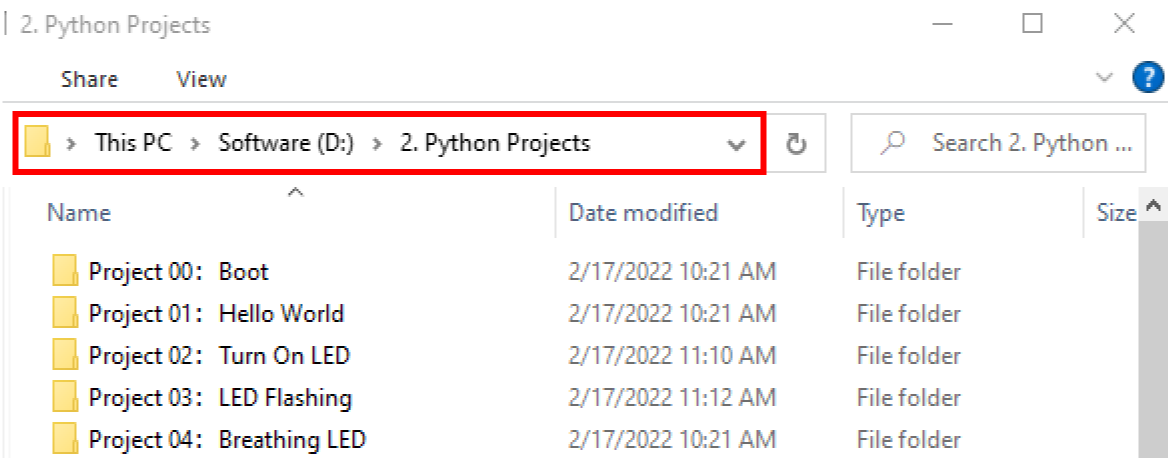
```
Shell ×
>>> %Run -c $EDITOR_CONTENT

  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

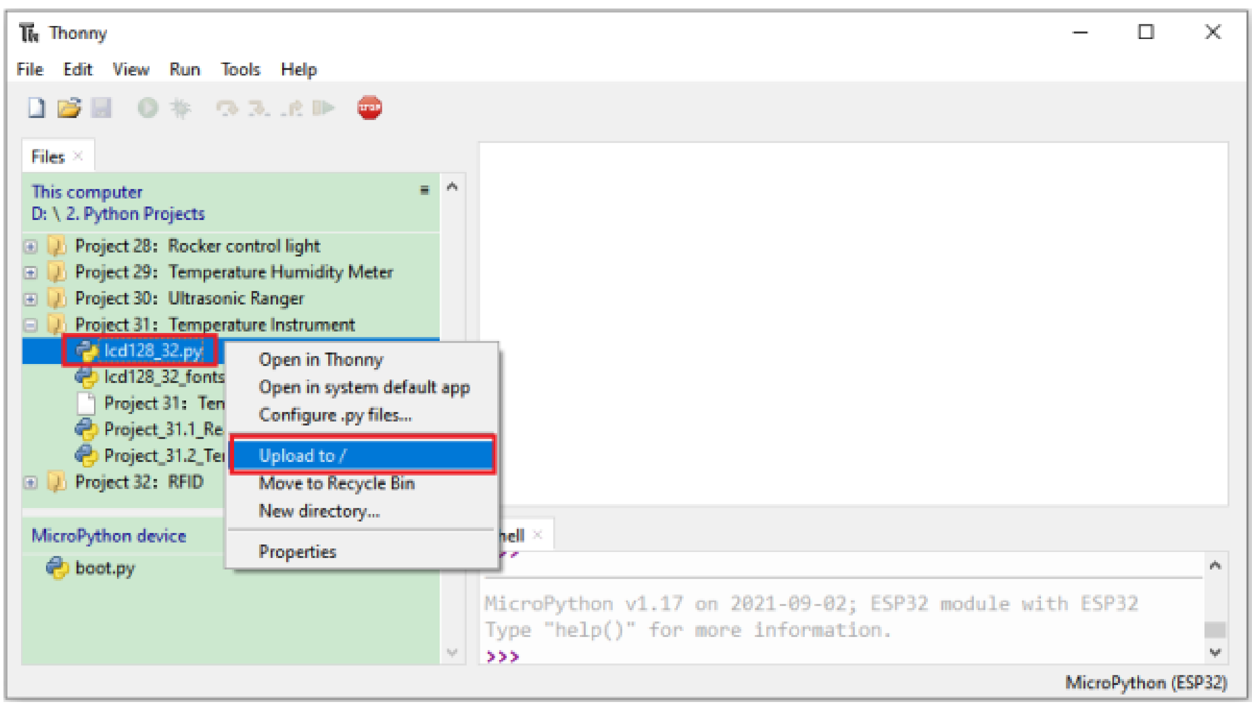


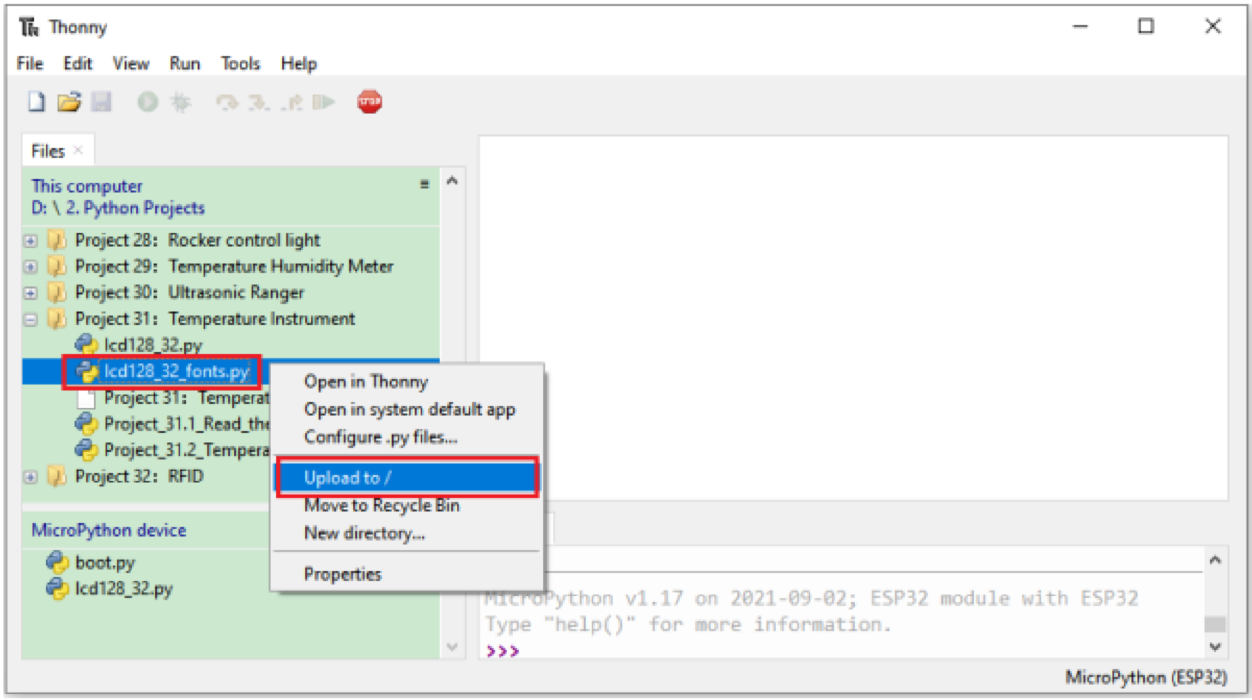
7.32.6 Project code

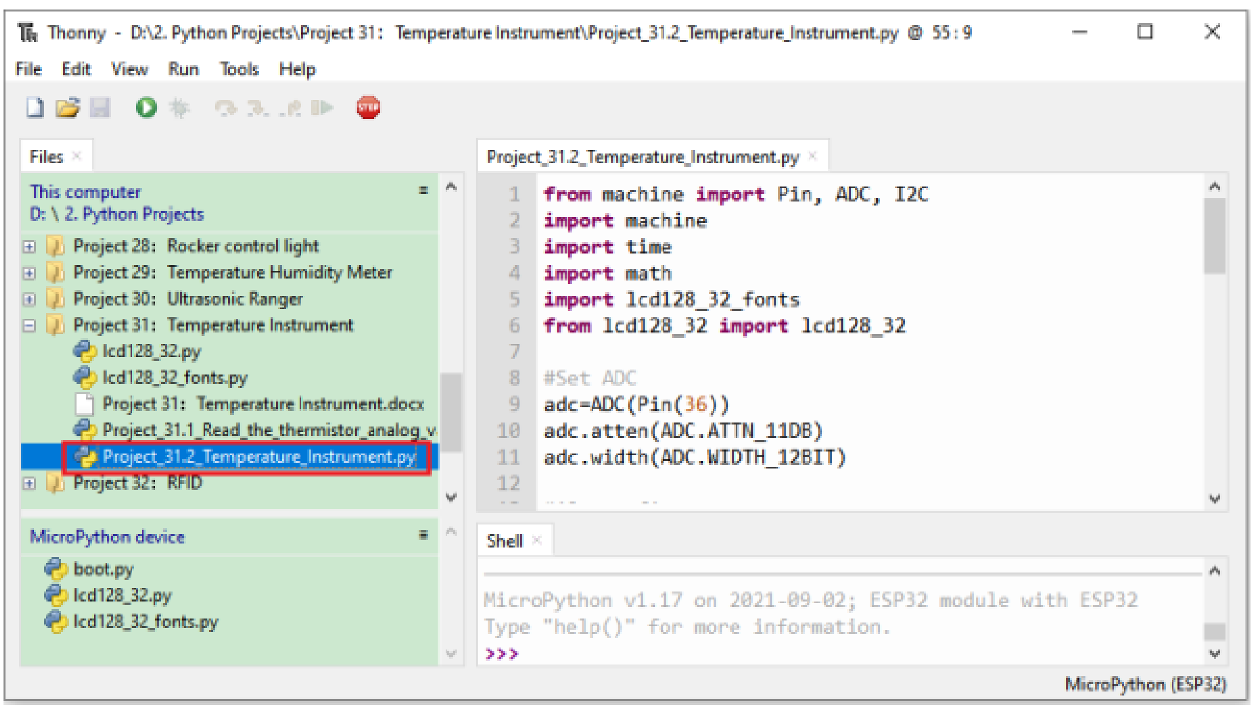
Codes used in this tutorial are saved in"**2. Python Projects**". If you haven't downloaded the code file, please click on the link to download it:Download Python Codes



Open "Thonny", click "This computer" → "D:" → "2. Python Projects" → "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".








```
from machine import Pin, ADC, I2C
import machine
import time
import math
import lcd128_32_fonts
from lcd128_32 import lcd128_32

#Set ADC
```

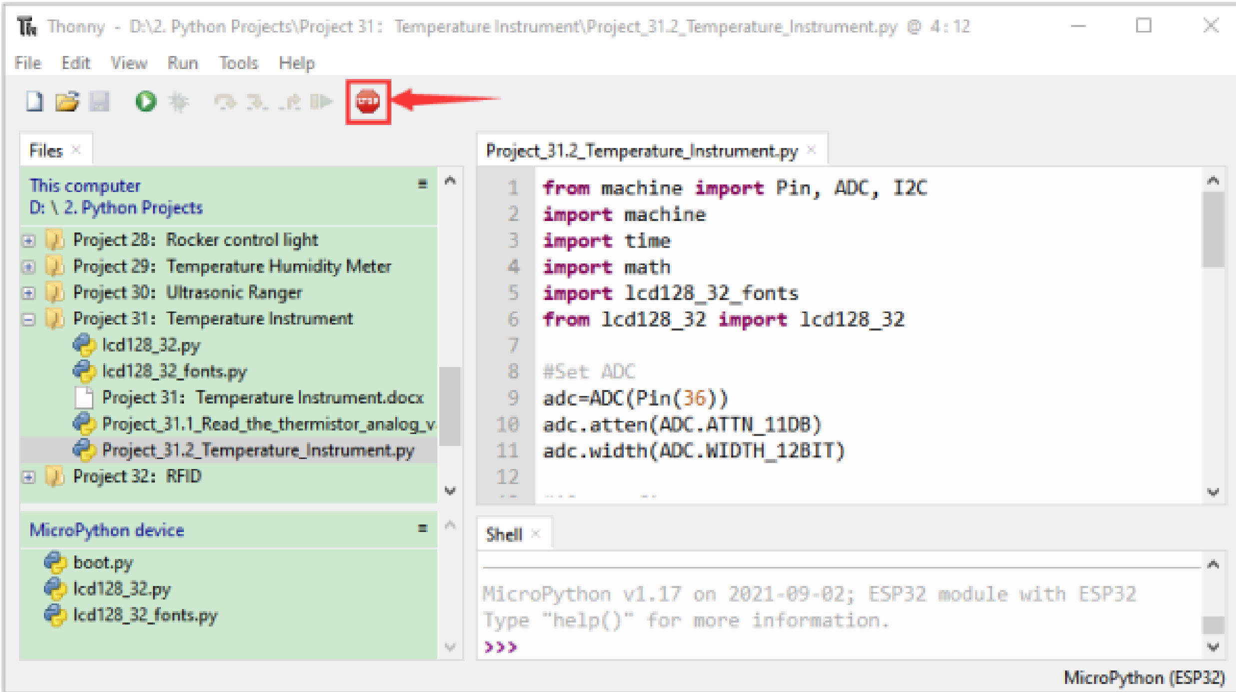
(continues on next page)

(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 * voltage / (3.3-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(0, 0)
        lcd.Display("Voltage:")
        lcd.Cursor(0, 8)
        lcd.Display(str(voltage))
        lcd.Cursor(0, 20)
        lcd.Display("V")
        lcd.Cursor(2, 0)
        lcd.Display("Temperature:")
        lcd.Cursor(2, 12)
        lcd.Display(str(tempC))
        lcd.Cursor(2, 15)
        lcd.Display("C")
        time.sleep(0.5)
except:
    pass
```

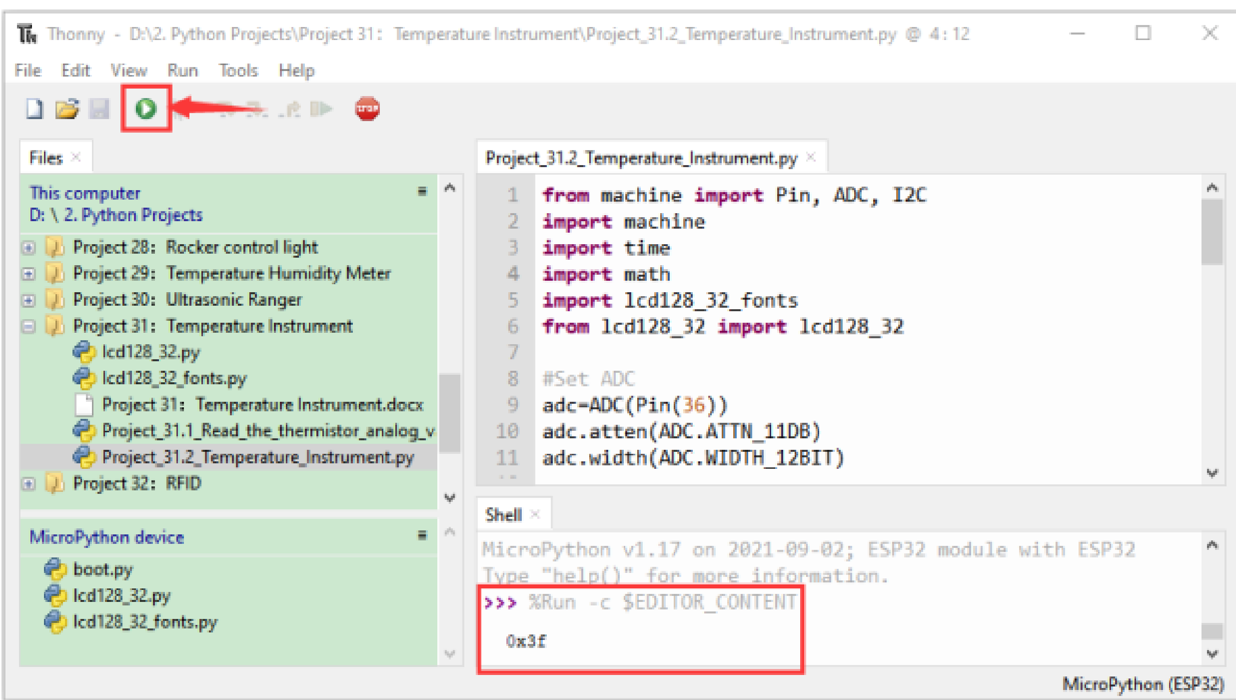
7.32.7 Project result

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 "Stop/Restart backend"to exit the program.

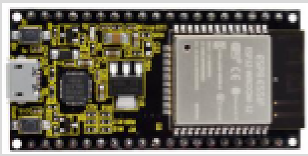
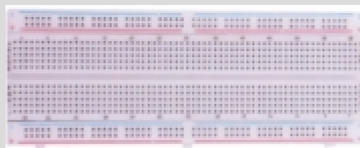
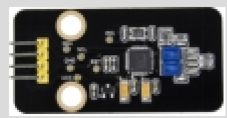
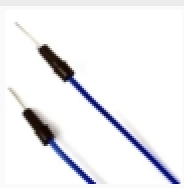
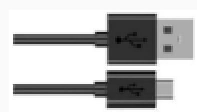


7.33 Project 32RFID

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

		
ESP32*1	Breadboard*1	RFID-RC522 Module*1
		
M-F Dupont Wires	Servo*1	White Card*1
		
Key Chain*1	Jumper wires	USB Cable*1

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.

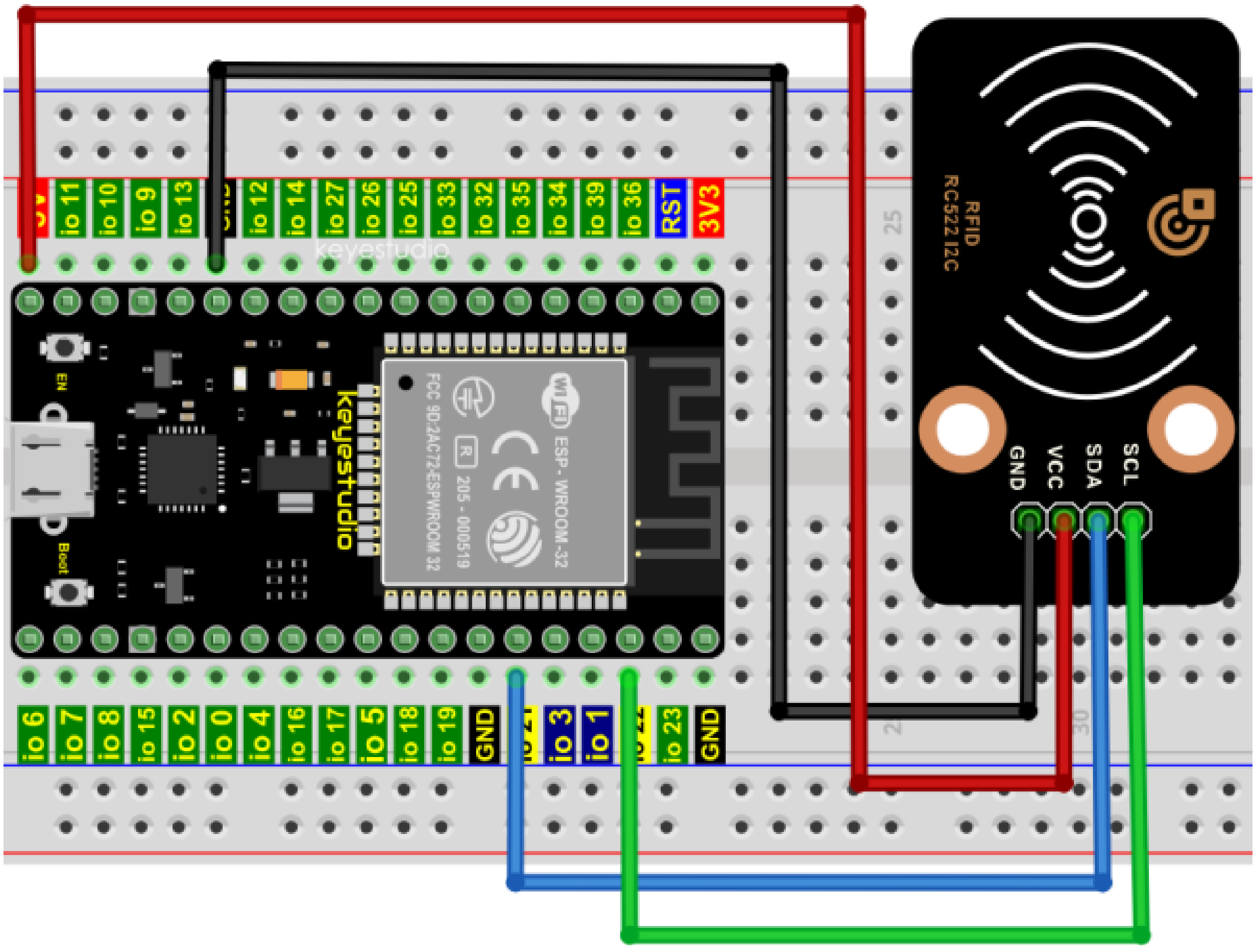
This RFID Module uses MFRC522 as the control chip and adopts I2C (Inter-Integrated Circuit) interface.

Specifications:

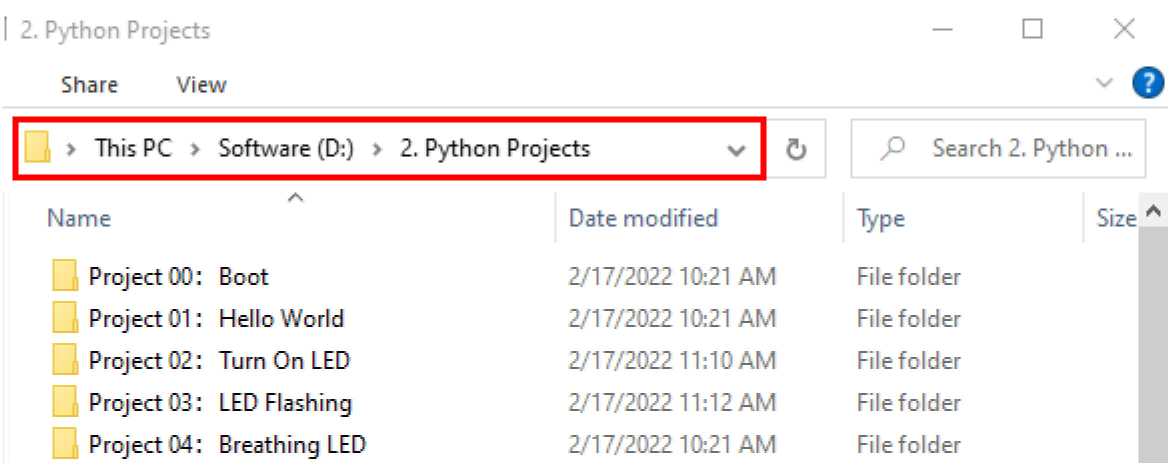
- Operating voltage: DC 3.3V-5V
- Operating current: 13—100mA/DC 5V
- Idling current: 10-13mA/DC 5V
- Sleep current: <80uA
- Peak current: <100mA
- Operating frequency: 13.56MHz
- Maximum power: 0.5W
- Supported card types: mifare1 S50, mifare1 S70, mifare UltraLight, mifare Pro, mifare Desfire.
- Environmental 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

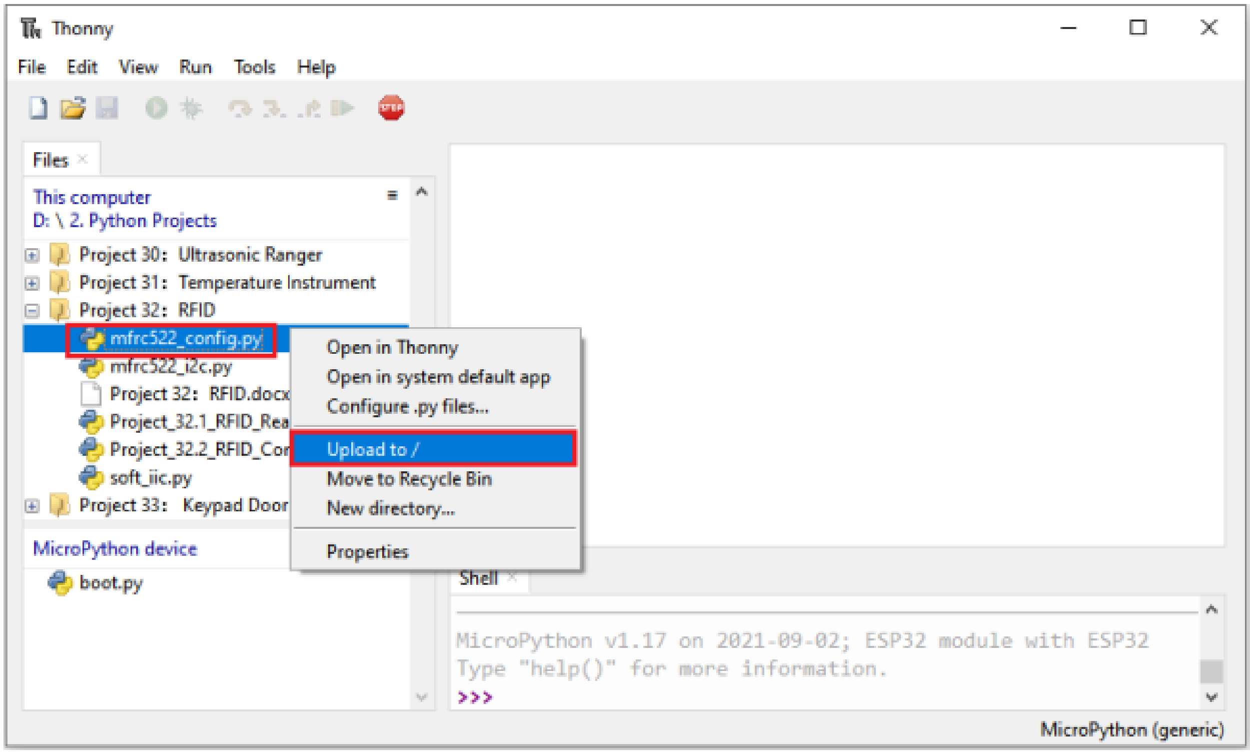


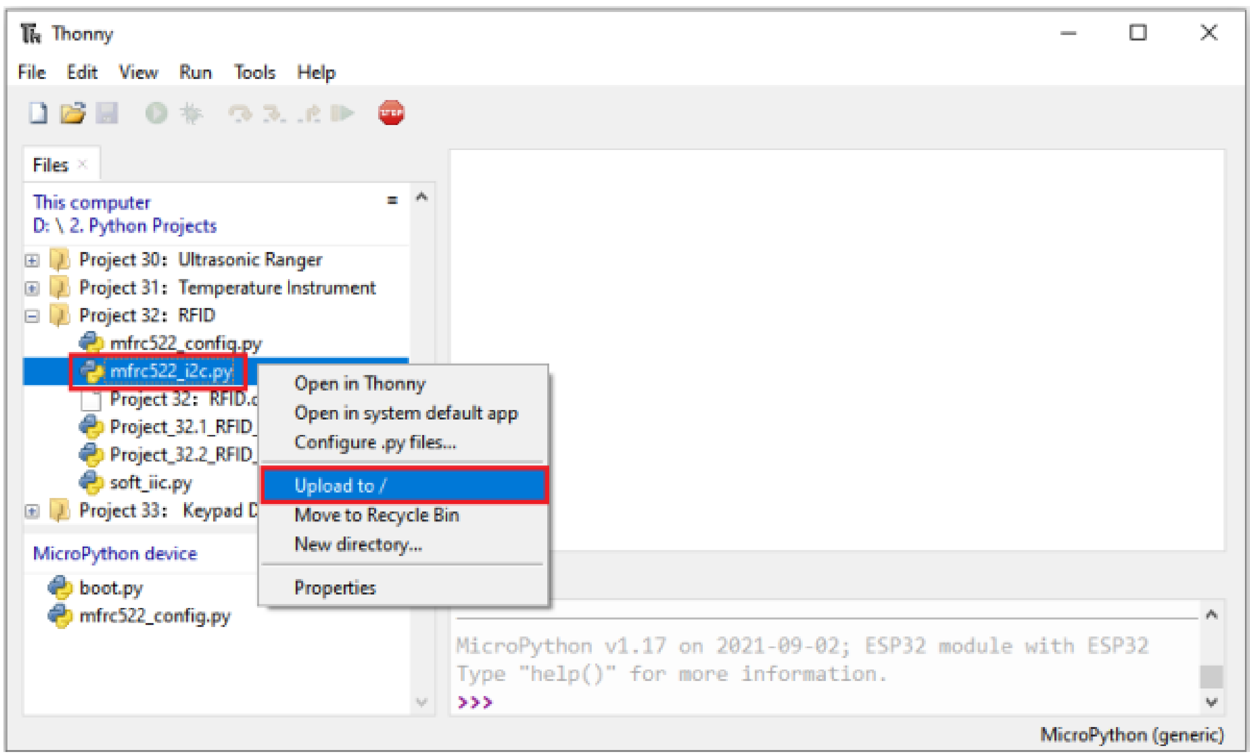
Codes used in this tutorial are saved in“**2. Python Projects**”. If you haven’t downloaded the code file, please click on the link to download it:Download Python Codes

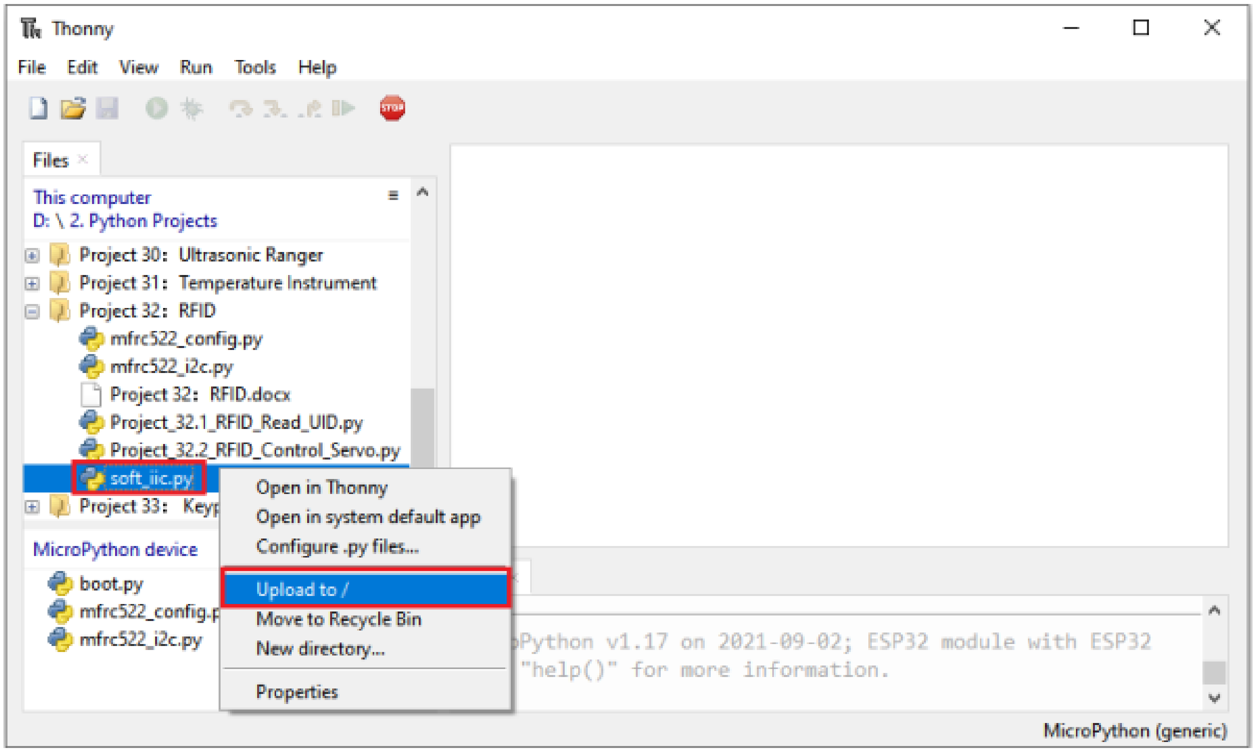


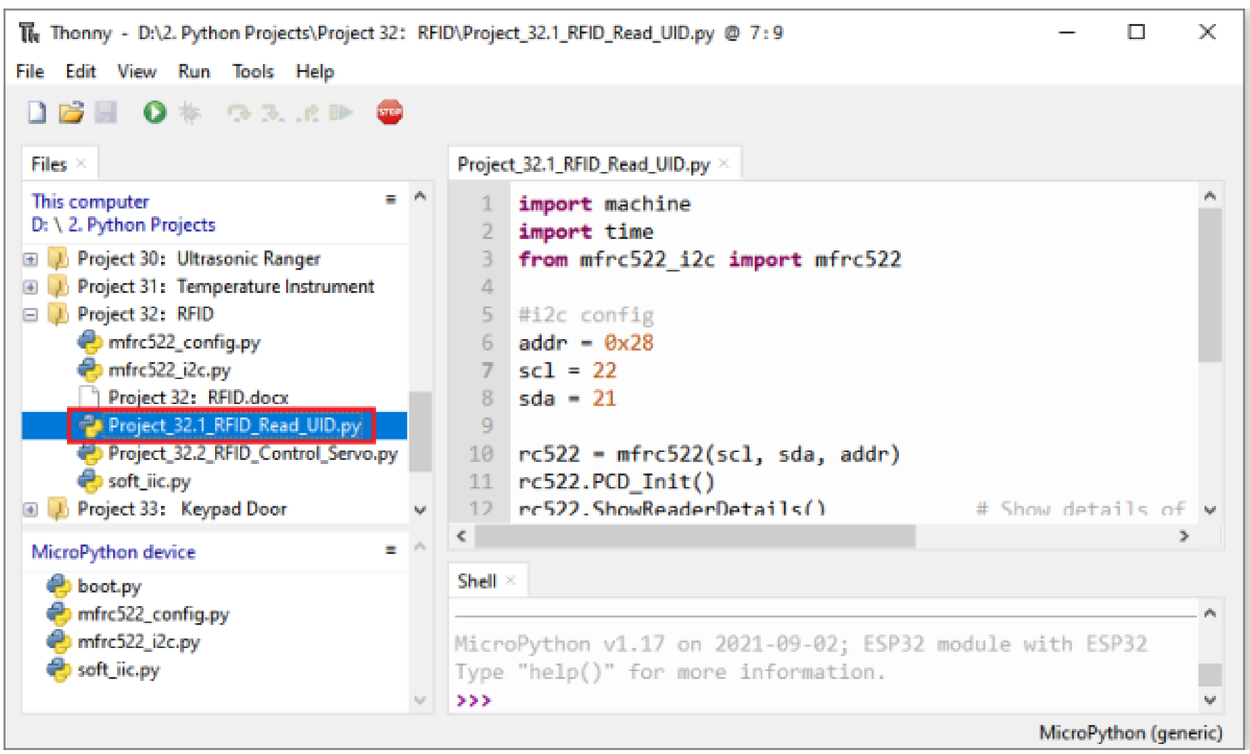
Open “Thonny”, click “This computer” → “D:” → “2. Python Projects” → “Project 32RFID”.

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
Files
This computer
D: \ 2. Python Projects
Project 30: Ultrasonic Ranger
Project 31: Temperature Instrument
Project 32: RFID
mfrc522_config.py
mfrc522_i2c.py
Project 32: RFID.docx
Project_32.1_RFID_Rea
Project_32.2_RFID_Cor
soft_iic.py
Project 33: Keypad Door
MicroPython device
boot.py
Open in Thonny
Open in system default app
Configure .py files...
Upload to /
Move to Recycle Bin
New directory...
Properties
Shell
MicroPython v1.17 on 2021-09-02; ESP32 module with ESP32
Type "help()" for more information.
>>>
MicroPython (generic)


Thonny
File Edit View Run Tools Help
Files
This computer
D: \ 2. Python Projects
Project 30: Ultrasonic Ranger
Project 31: Temperature Instrument
Project 32: RFID
mfrc522_config.py
mfrc522_i2c.py
Project 32: RFID.d
Project_32.1_RFID_
Project_32.2_RFID_
soft_iic.py
Project 33: Keypad D
MicroPython device
boot.py
mfrc522_config.py
Open in Thonny
Open in system default app
Configure .py files...
Upload to /
Move to Recycle Bin
New directory...
Properties
MicroPython v1.17 on 2021-09-02; ESP32 module with ESP32
Type "help()" for more information.
>>>
MicroPython (generic)





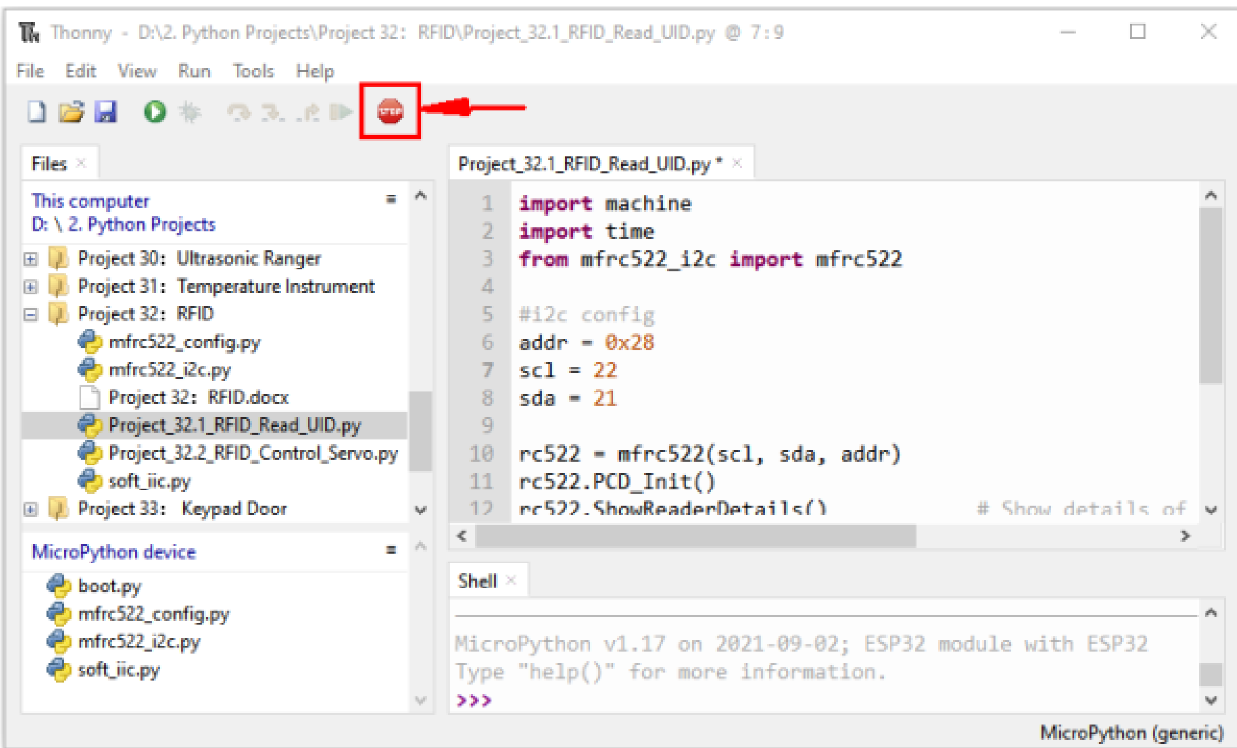

```
import machine
import time
from mfrc522_i2c import mfrc522

#i2c config
```

(continues on next page)

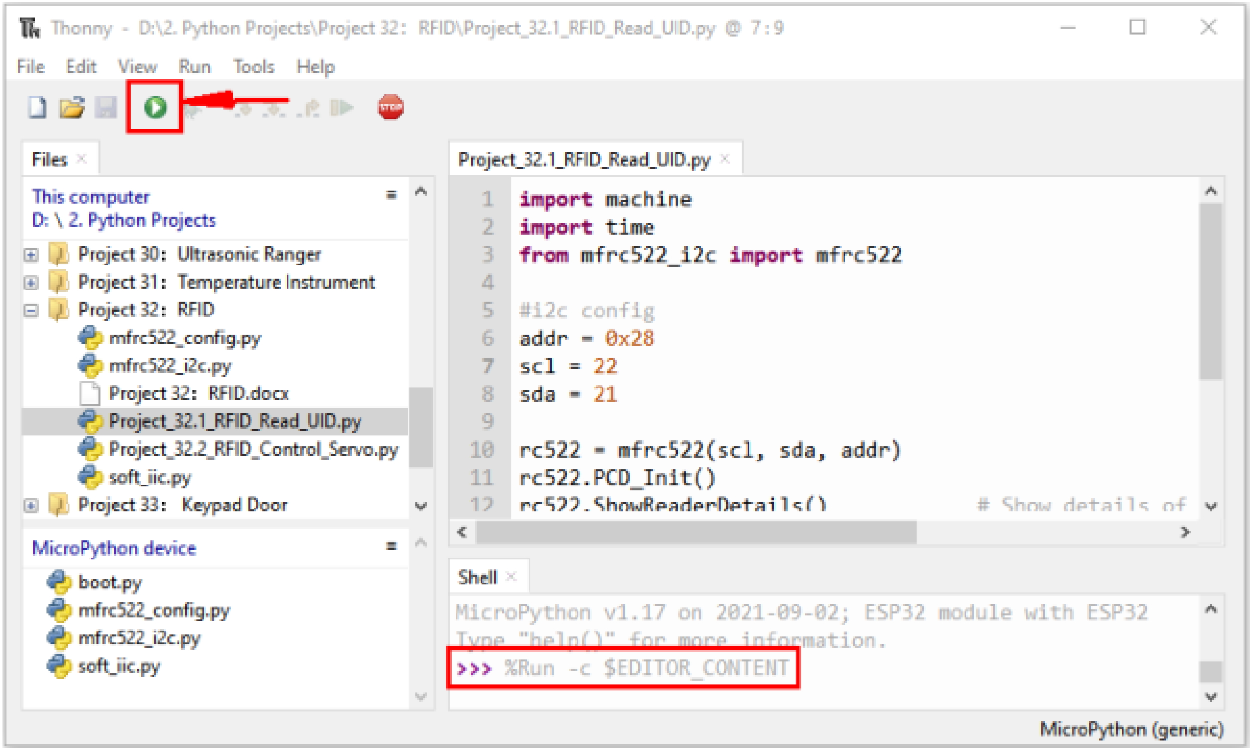
(continued from previous page)

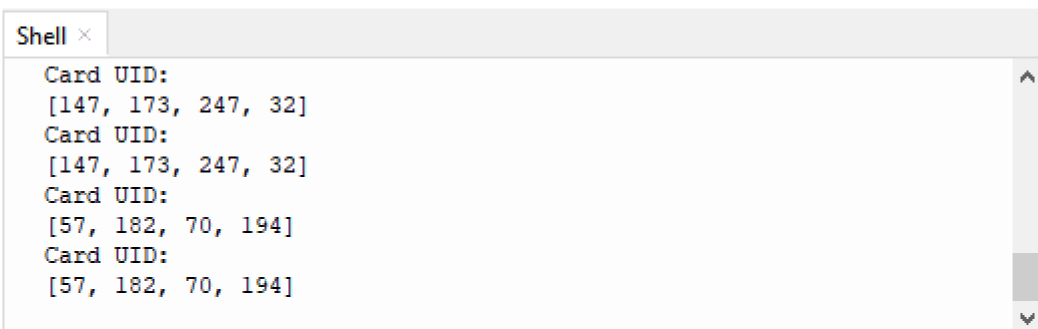
```
addr = 0x28
scl = 22
sda = 21

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 "Stop/Restart backend".



Click "Run current script", the code starts to be executed and you'll see that place the door card and key chain close to the module sensor area respectively, the "Shell" window of Thonny IDE will display the card number and key chain value respectively, as shown below. Press"Ctrl+C"or click "Stop/Restart backend"to exit the program.

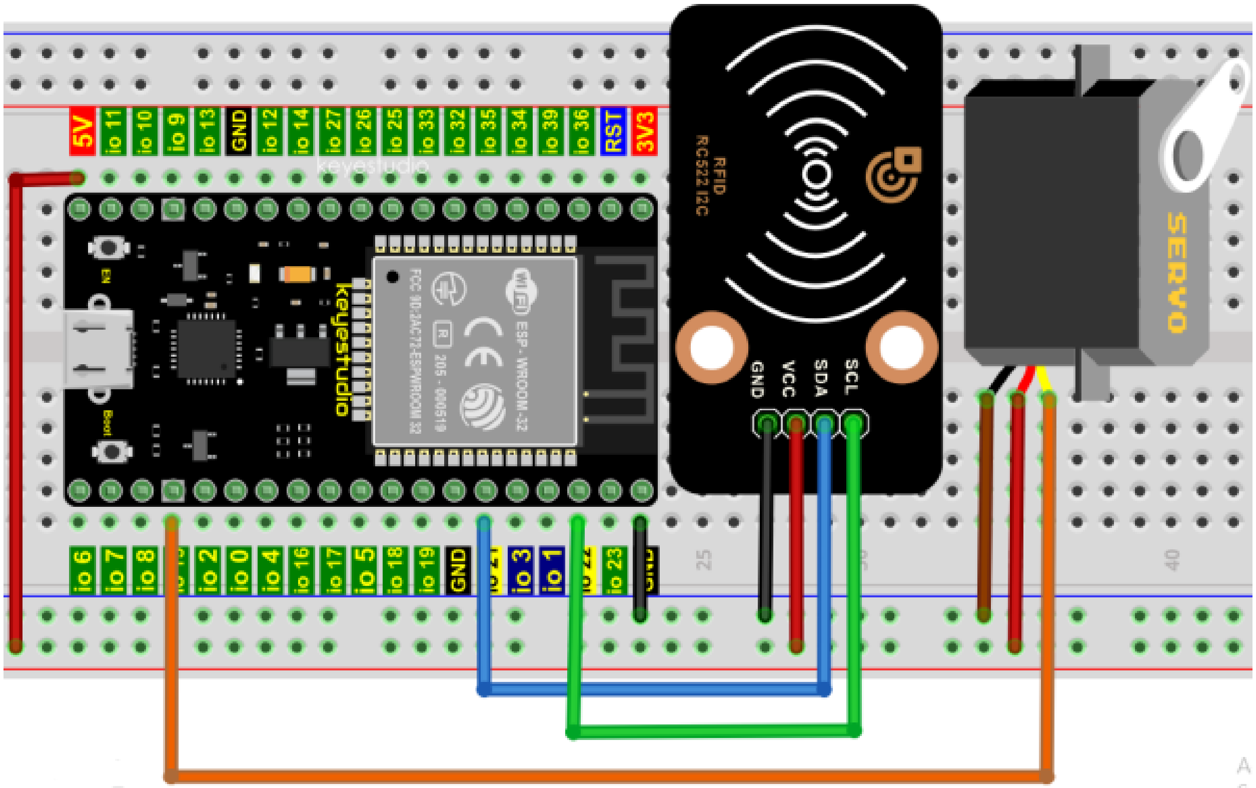




Note: the door card value and key chain value may be different for different RRFID -RC522 door cards and key chains.

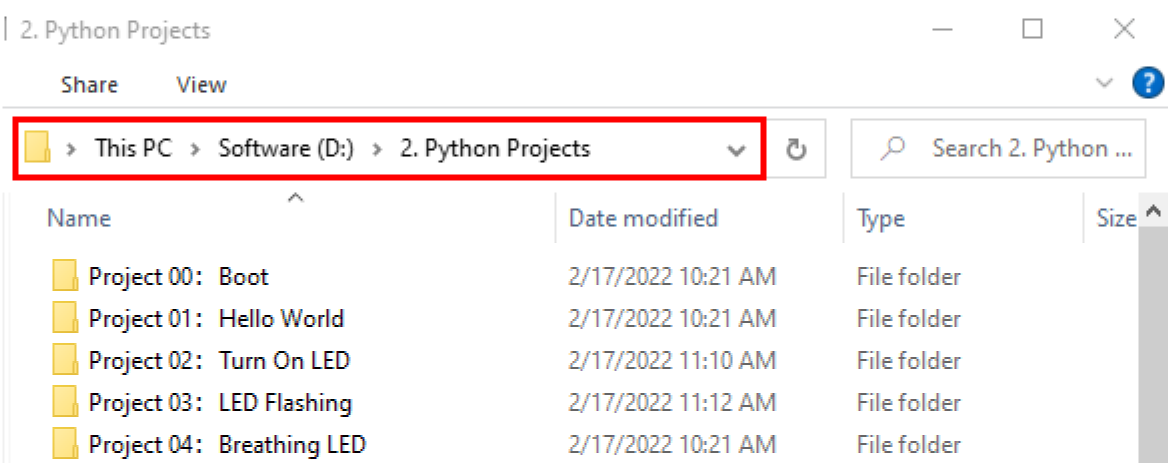
7.33.5 Wiring diagram of the RFID MFRC522

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

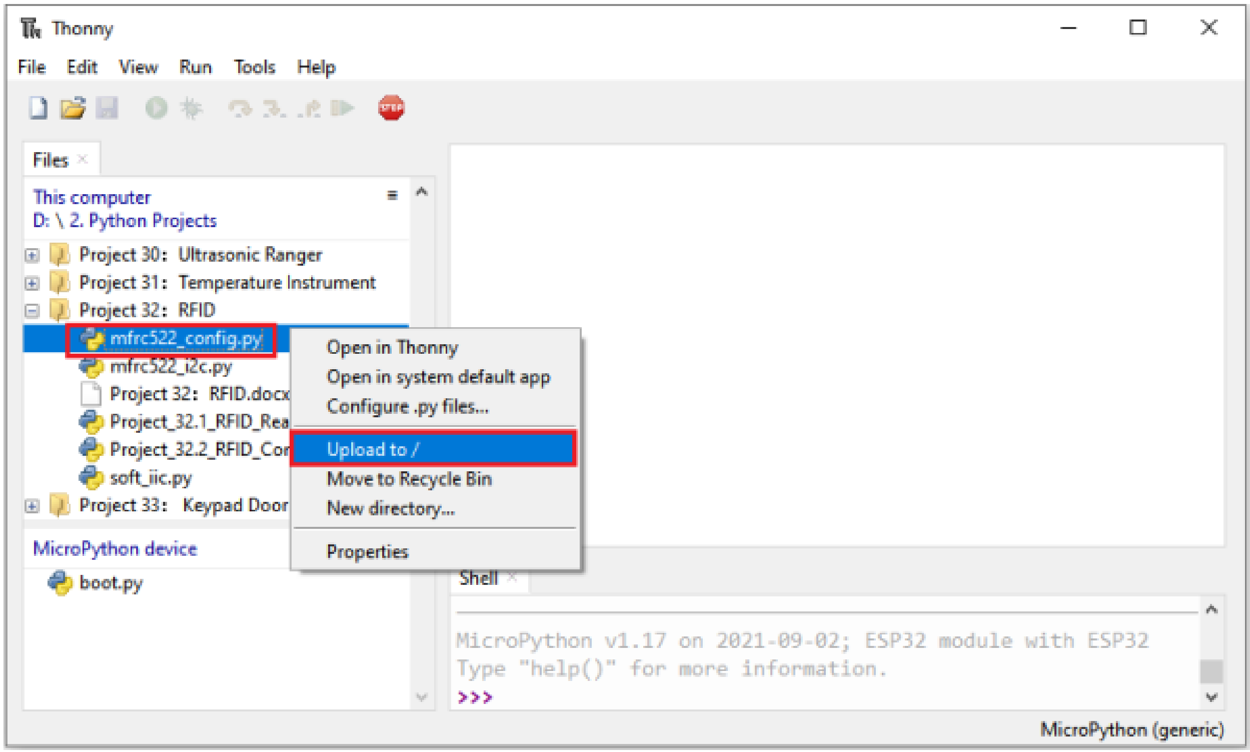


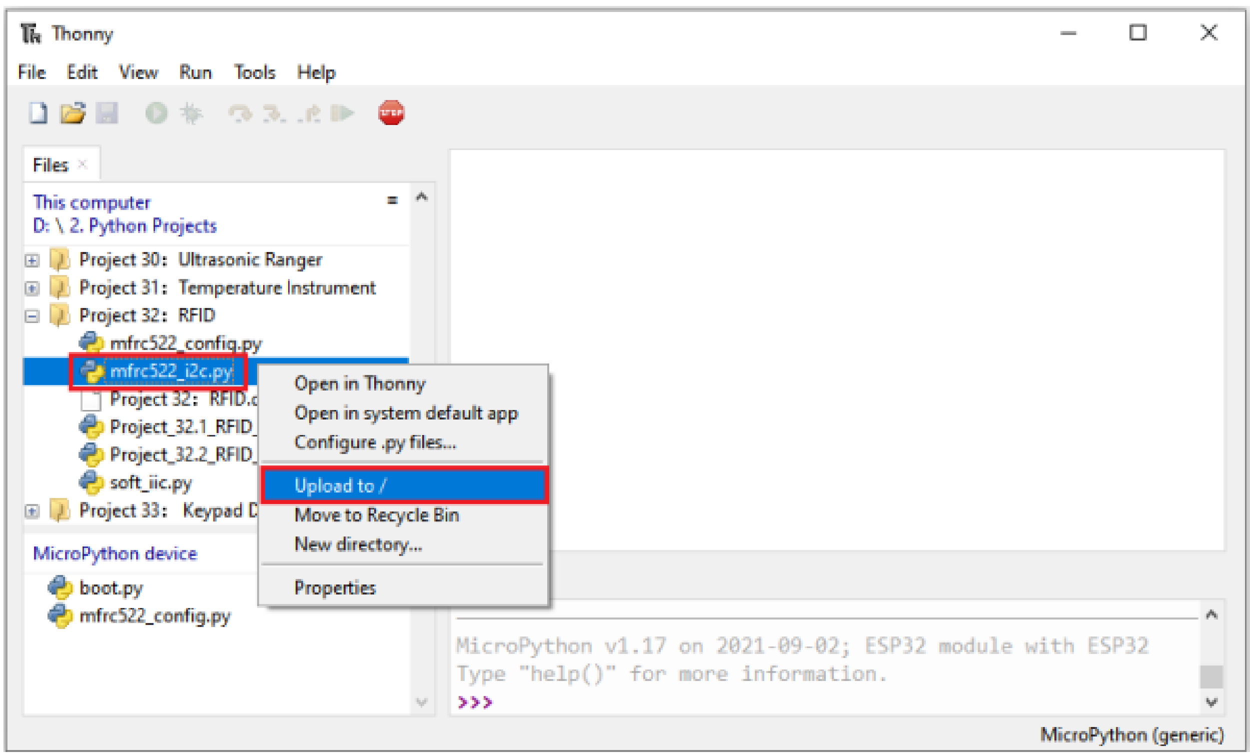
7.33.6 Project code

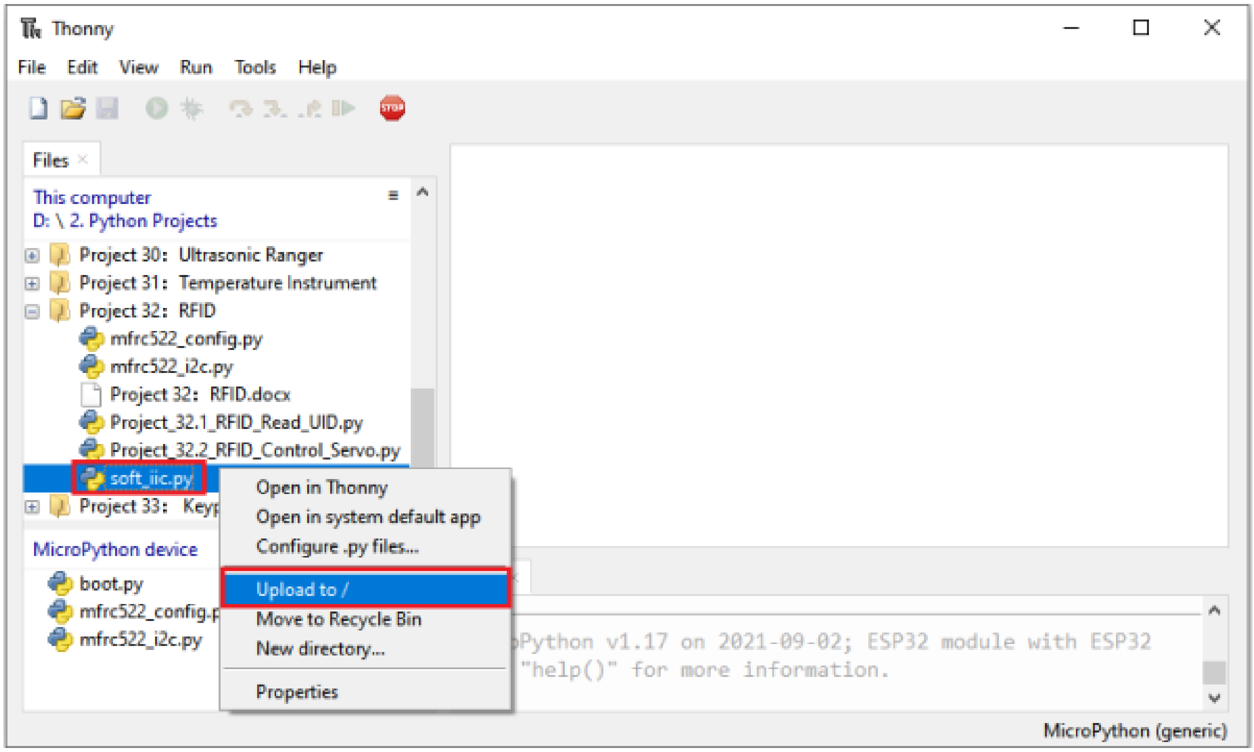
Codes used in this tutorial are saved in“**2. Python Projects**”. If you haven’t downloaded the code file, please click on the link to download it:Download Python Codes

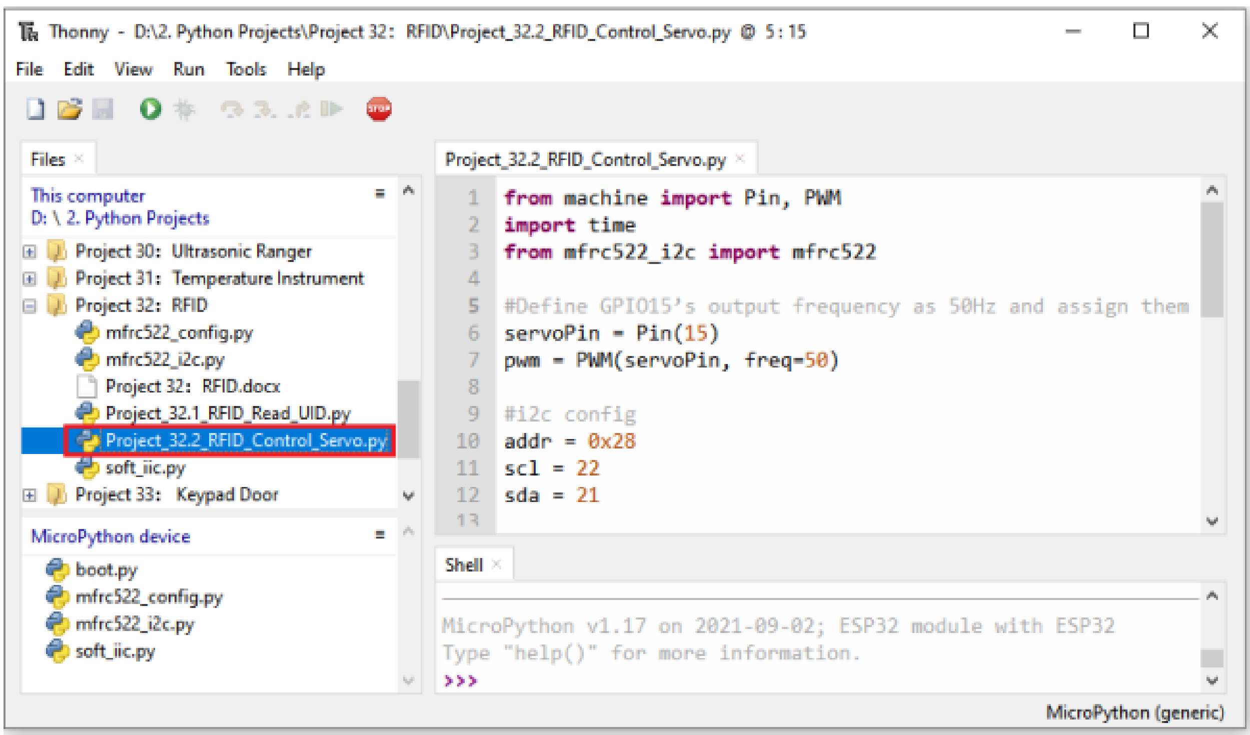


Open“Thonny”, click“This computer”→“D:”→“2. Python Projects”→“Project 32RFID”. Select “mfrc522_config.py”, “mfrc522_i2c.py” and “soft_iic.py”, click your mouse to select “Upload to /”, wait for “mfrc522_config.py”, “mfrc522_i2c.py” and “soft_iic.py” to be uploaded to ESP32, and click “Project_32.2_RFID_Control_Servo.py”.


Thonny
File Edit View Run Tools Help
Files
This computer
D: \ 2. Python Projects
Project 30: Ultrasonic Ranger
Project 31: Temperature Instrument
Project 32: RFID
mfrc522_config.py
mfrc522_i2c.py
Project 32: RFID.docx
soft_iic.py
MicroPython device
boot.py
Open in Thonny
Open in system default app
Configure .py files...
Upload to /
Move to Recycle Bin
New directory...
Properties
Shell
MicroPython v1.17 on 2021-09-02; ESP32 module with ESP32
Type "help()" for more information.
>>>
MicroPython (generic)


Thonny
File Edit View Run Tools Help
Files
This computer
D: \ 2. Python Projects
Project 30: Ultrasonic Ranger
Project 31: Temperature Instrument
Project 32: RFID
mfrc522_config.py
mfrc522_i2c.py
soft_iic.py
MicroPython device
boot.py
mfrc522_config.py
Open in Thonny
Open in system default app
Configure .py files...
Upload to /
Move to Recycle Bin
New directory...
Properties
MicroPython v1.17 on 2021-09-02; ESP32 module with ESP32
Type "help()" for more information.
>>>
MicroPython (generic)






```
from machine import Pin, PWM
import time
from mfrc522_i2c import mfrc522

#Define GPIO15'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.

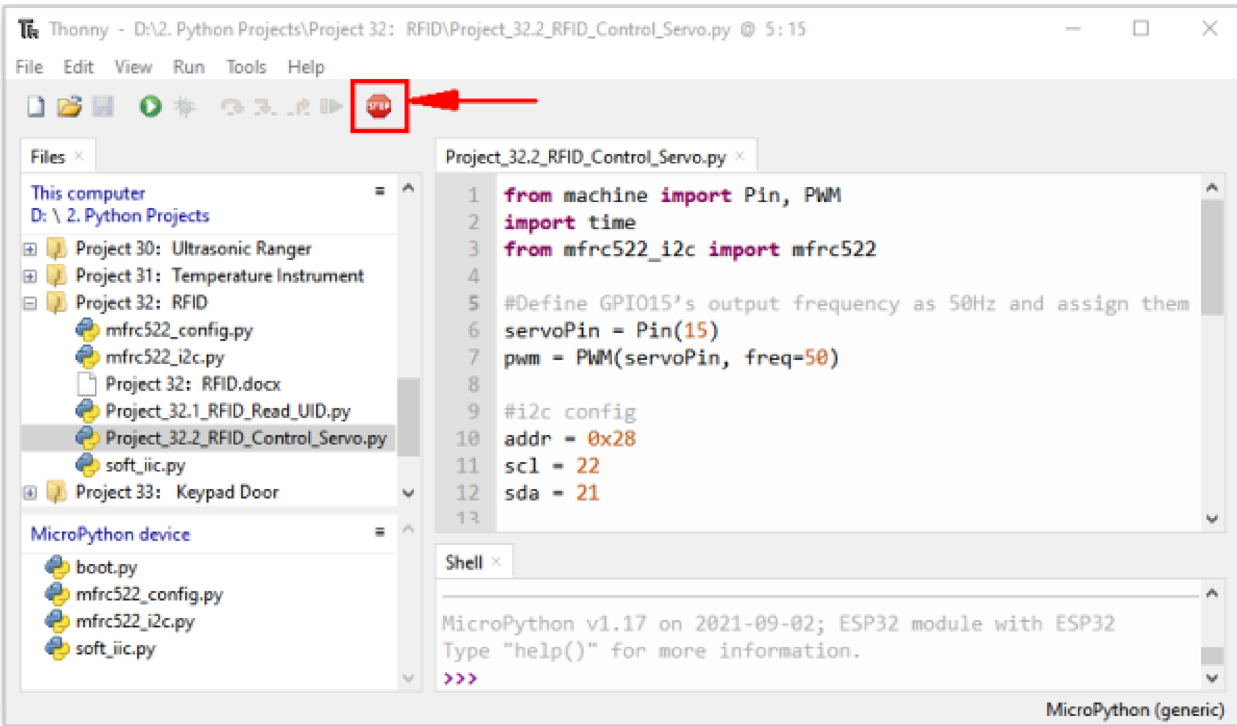
For example: You can replace the UID1 and UID2 values in the program code

```
uid1 = [147, 173, 247, 32]
uid2 = [57, 182, 70, 194]
```

with your own white card and key chain values.

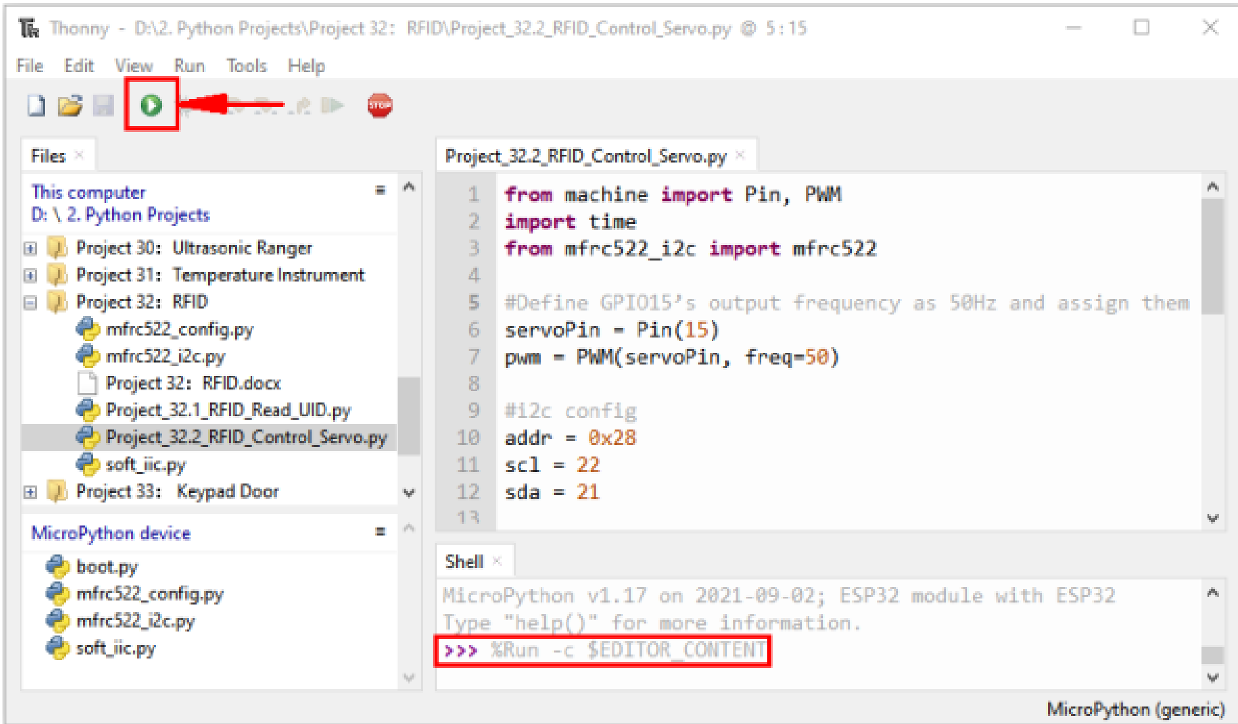
7.33.7 Project result

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.

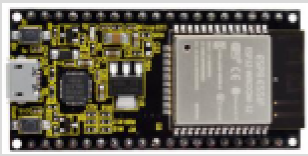
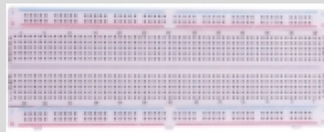
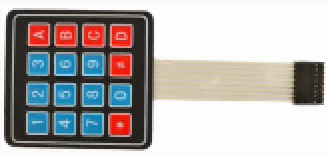
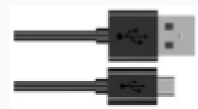
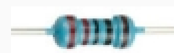


7.34 Project 33Keypad Door

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.

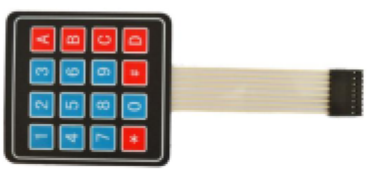
7.34.2 Components

			
ESP32*1	Breadboard*1	Servo*1	Active Buzzer*1
			
*44 Membrane Matrix Keyboard*1	Jumper Wires	USB Cable*1	1kResistor*1
			
NPN transistor(S8050)*1			

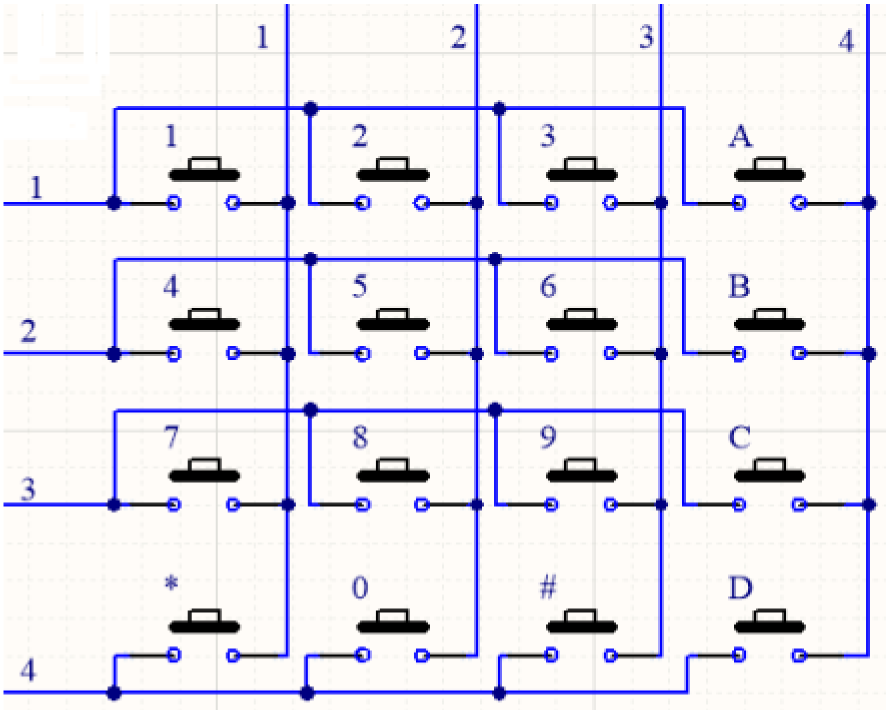
7.34.3 Component knowledge

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:



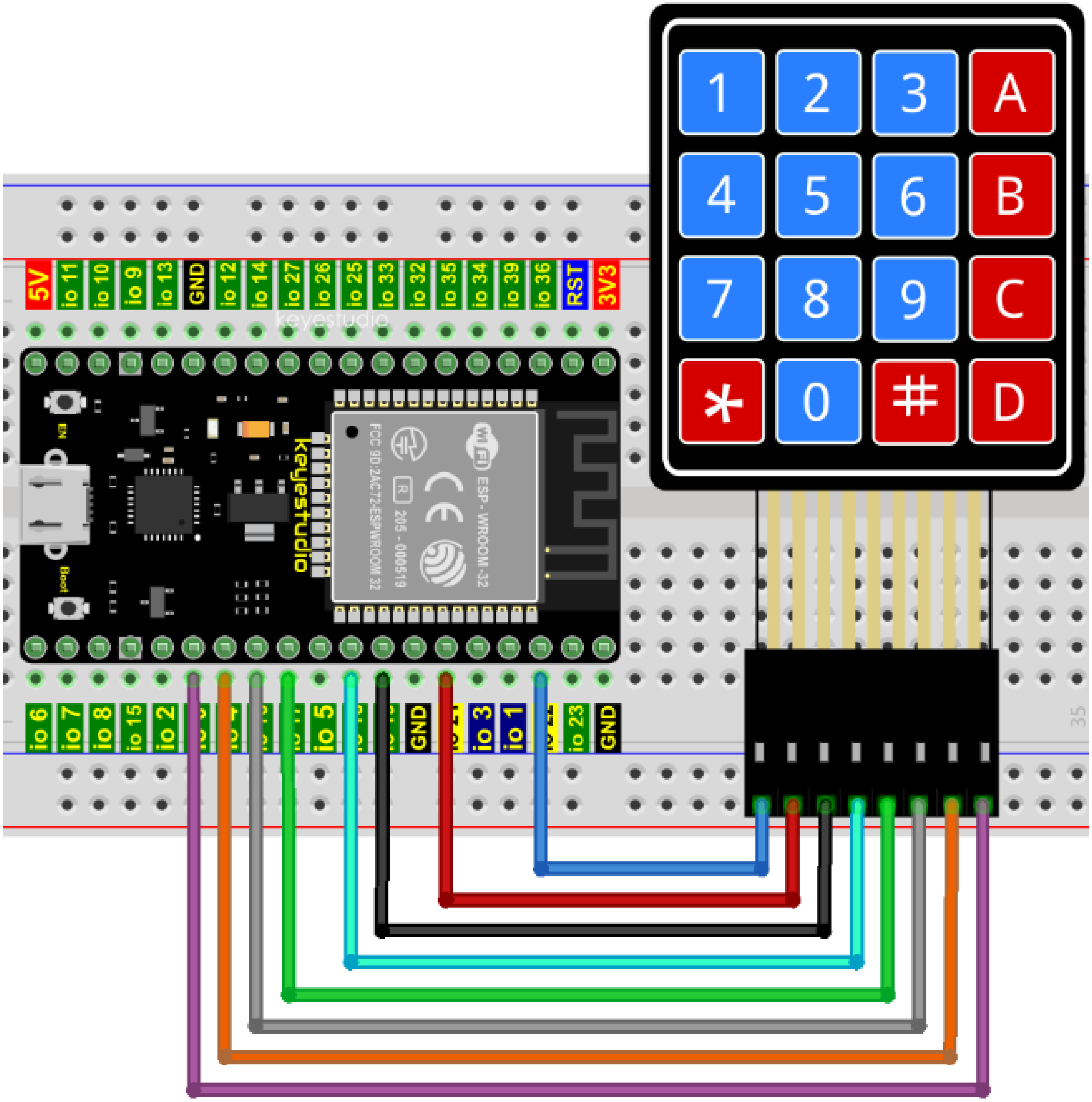
Similar to the integration of an LED Matrix, the 4x4 Keypad Matrix has each row of keys connected with one pin and this is the same for the columns. Such efficient connections reduce the number of processor ports required. The internal circuit of the Keypad Matrix is shown below.



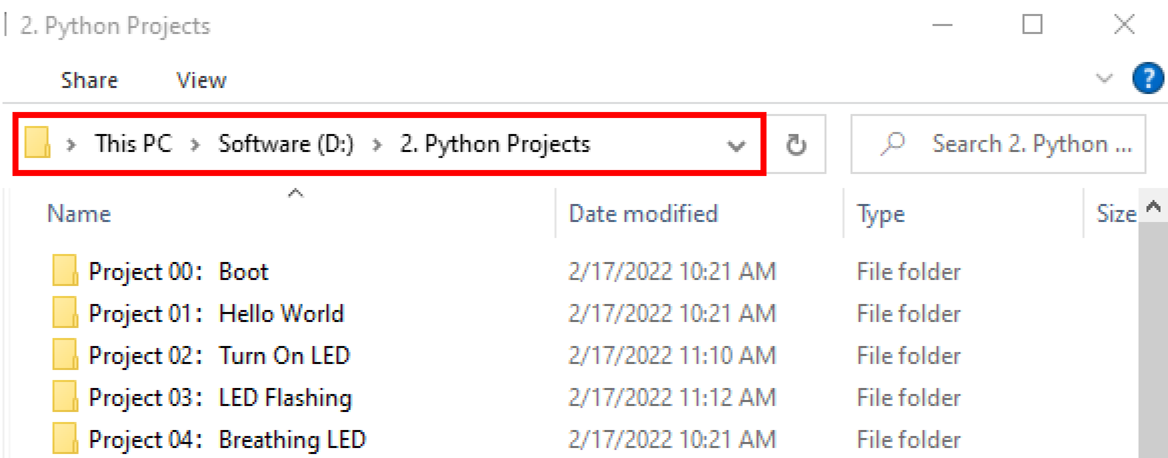
The method of usage is similar to the Matrix LED, by using a row or column scanning method to detect the state of each key's position by column and row. Take column scanning method as an example, send low level to the first 4 column (Pin4), detect level state of row 1, 2, 3, 4 to judge whether the key 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

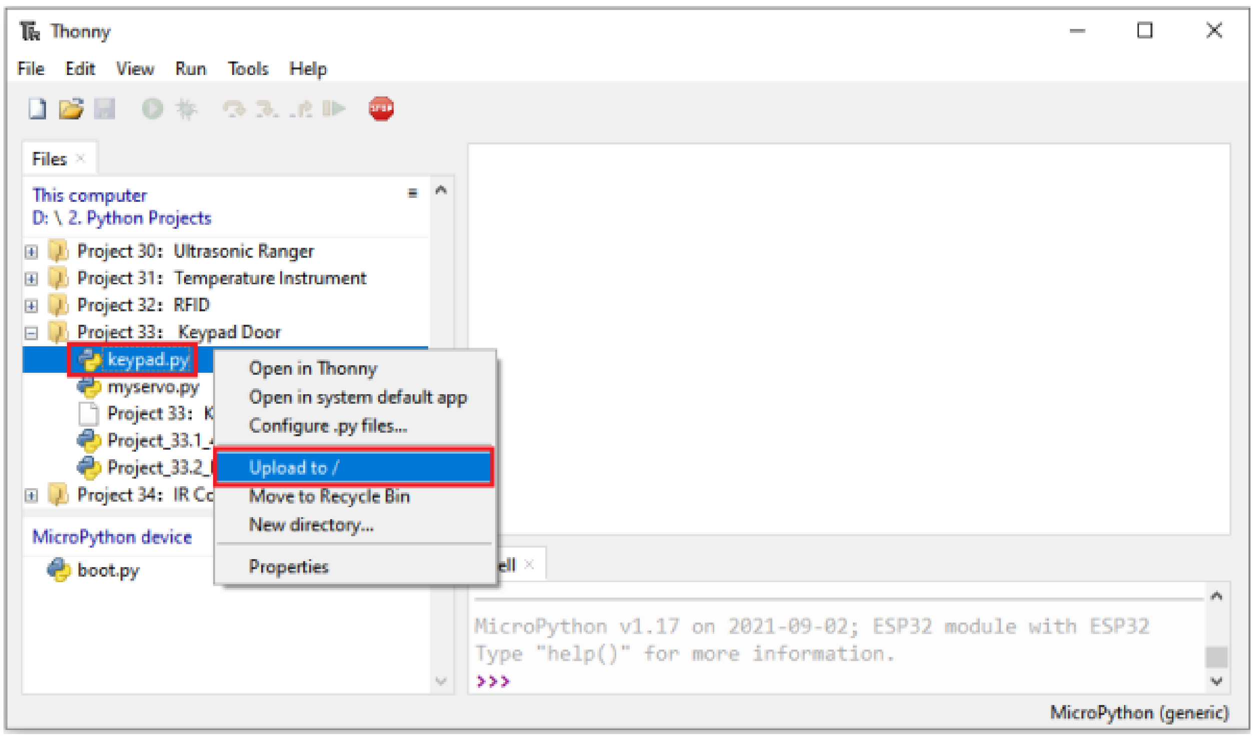
We start with a simple code to read the values of the 4*4 matrix keyboard and print them in the serial monitor. Its wiring diagram is shown below

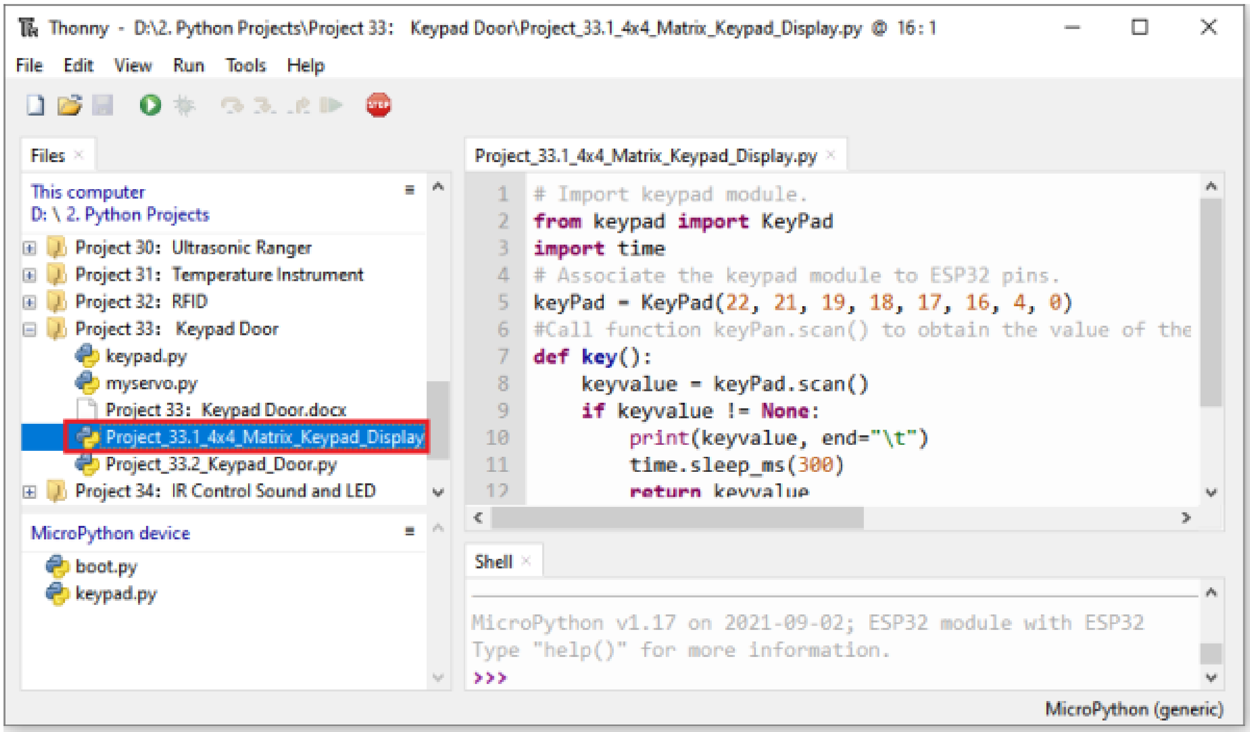


Codes used in this tutorial are saved in“**2. Python Projects**”. If you haven’t downloaded the code file, please click on the link to download it:Download Python Codes



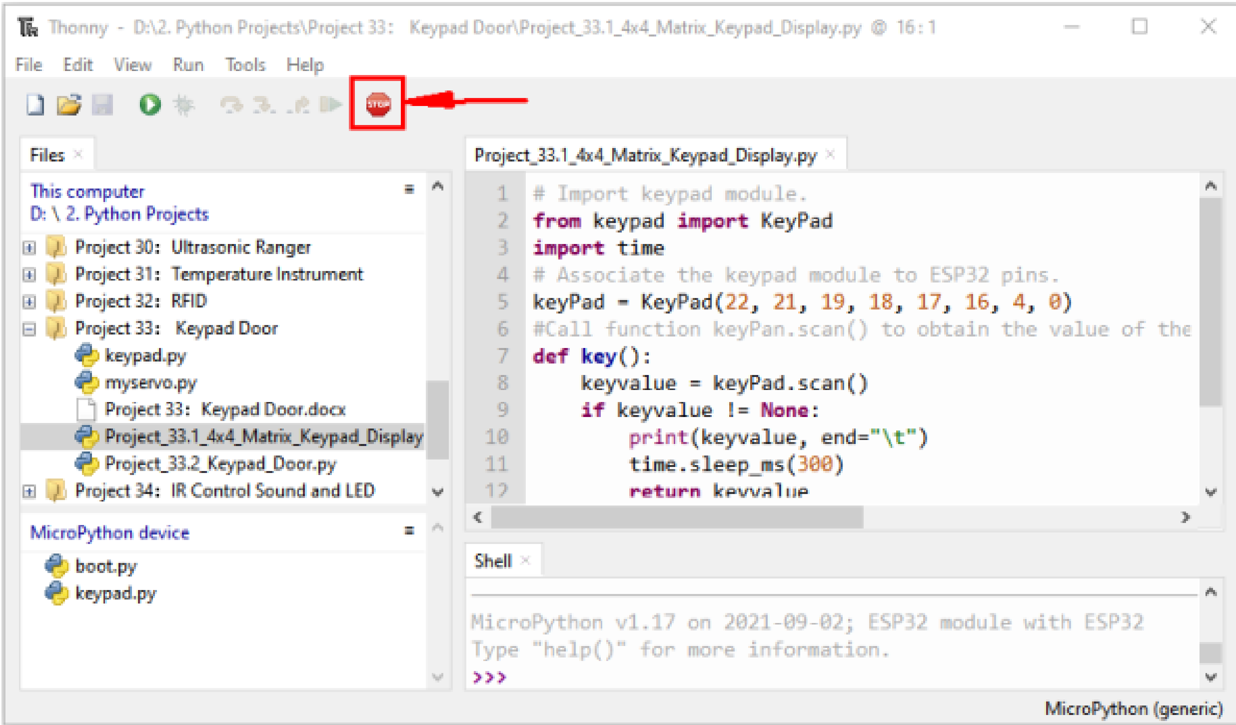
Open“Thonny”, click “This computer”→“D:”→“2. Python Projects”→“Project 33Keypad Door”. Select “keypad.py”, select “Upload to /”, wait for “keypad.py” to be uploaded to ESP32, and click “Project_33.1_4x4_Matrix_Keypad_Display.py”.



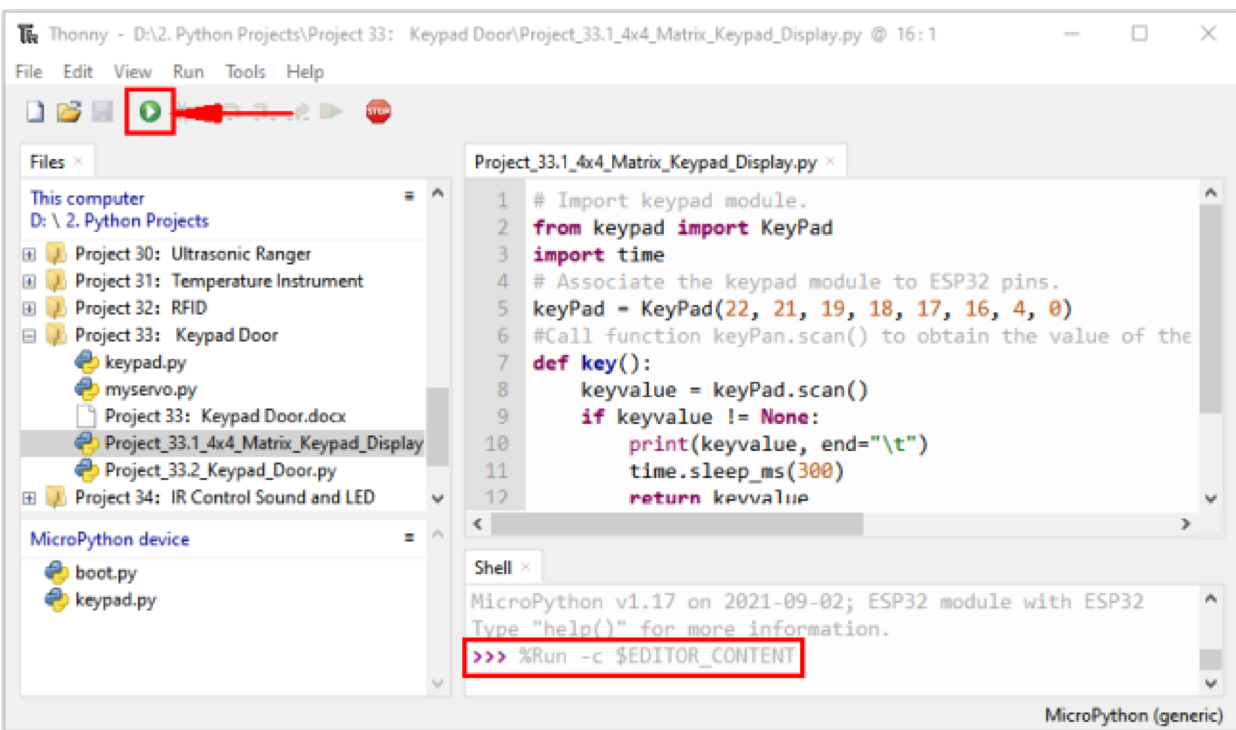


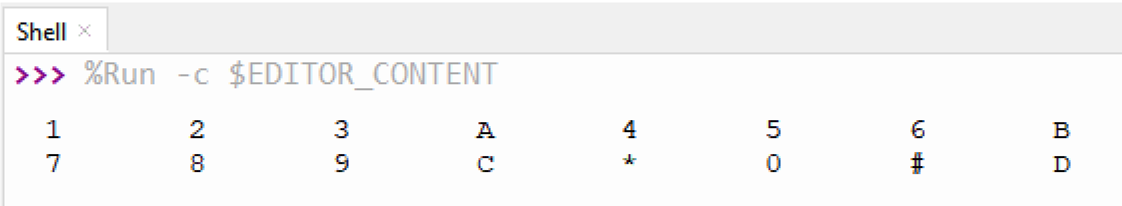

```
## Import keypad module.
from keypad import KeyPad
import time
## Associate the keypad module to ESP32 pins.
keyPad = KeyPad(22, 21, 19, 18, 17, 16, 4, 0)
#Call function keyPan.scan() to obtain the value of the pressed key. Once it is obtained,
↪ print it out.
def key():
    keyvalue = keyPad.scan()
    if keyvalue != None:
        print(keyvalue, end="\t")
        time.sleep_ms(300)
        return keyvalue

while True:
    key()
```

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 keyboard and the “Shell” window of Thonny IDE prints the corresponding key value, as shown below. Press“Ctrl+C”or click “Stop/Restart backend” to exit the program.

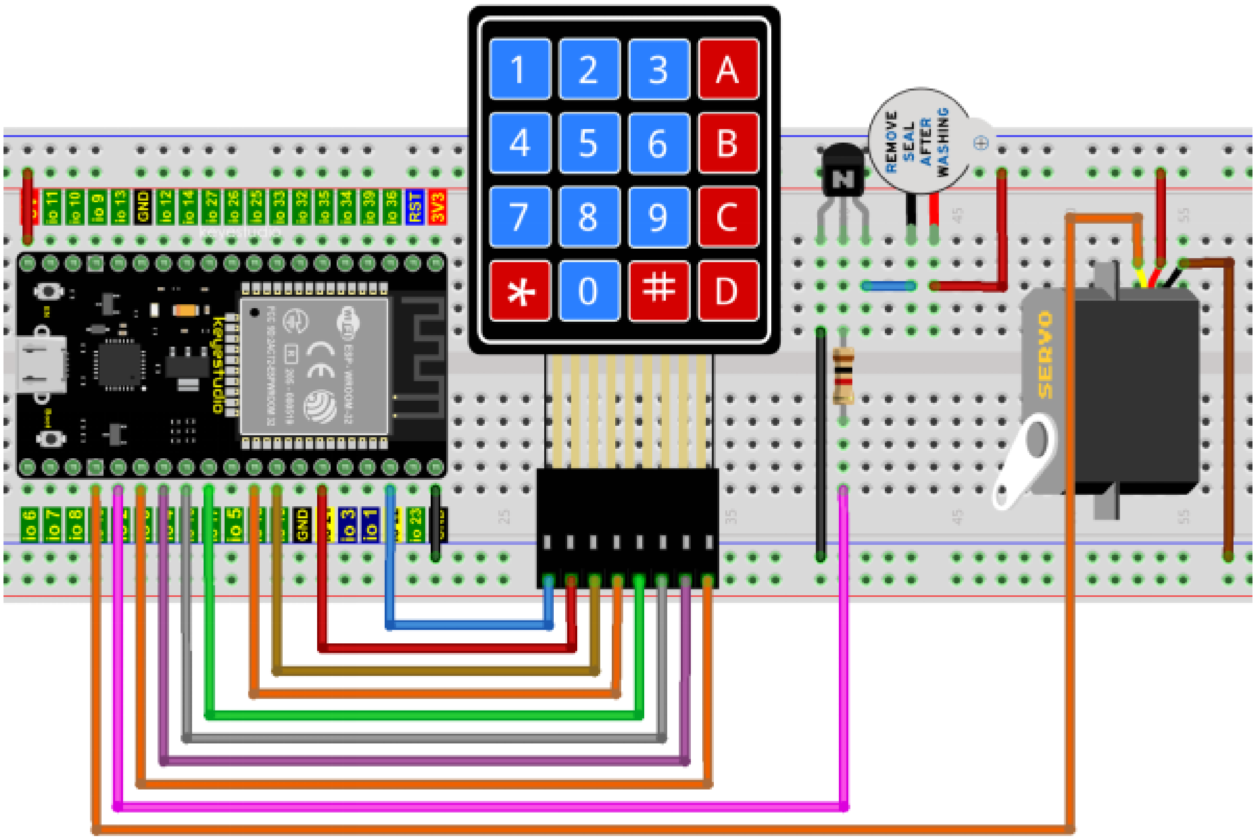




```
Shell
>>> %Run -c $EDITOR_CONTENT

  1         2         3         A         4         5         6         B
  7         8         9         C         *         0         #         D
```

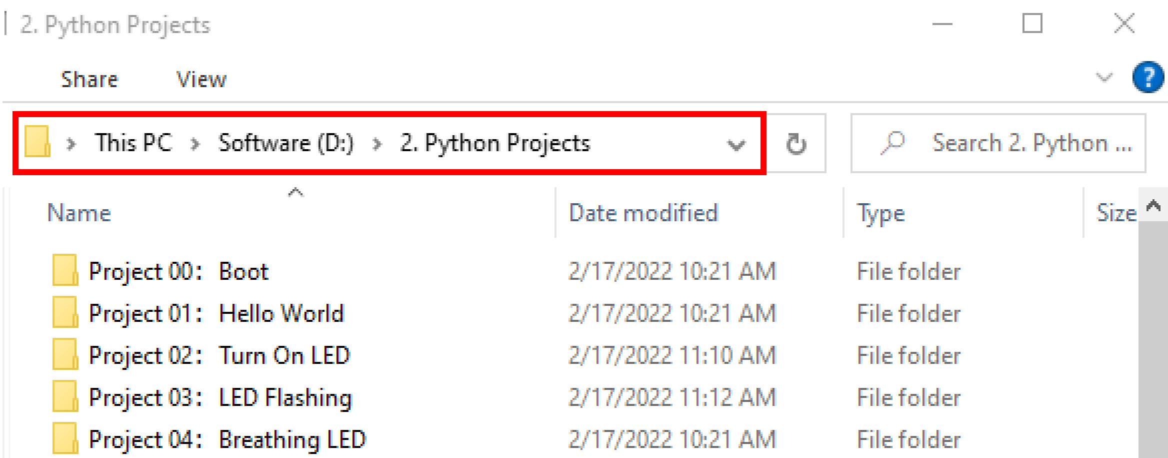
7.34.5 Wiring diagram of the Keypad Door

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.

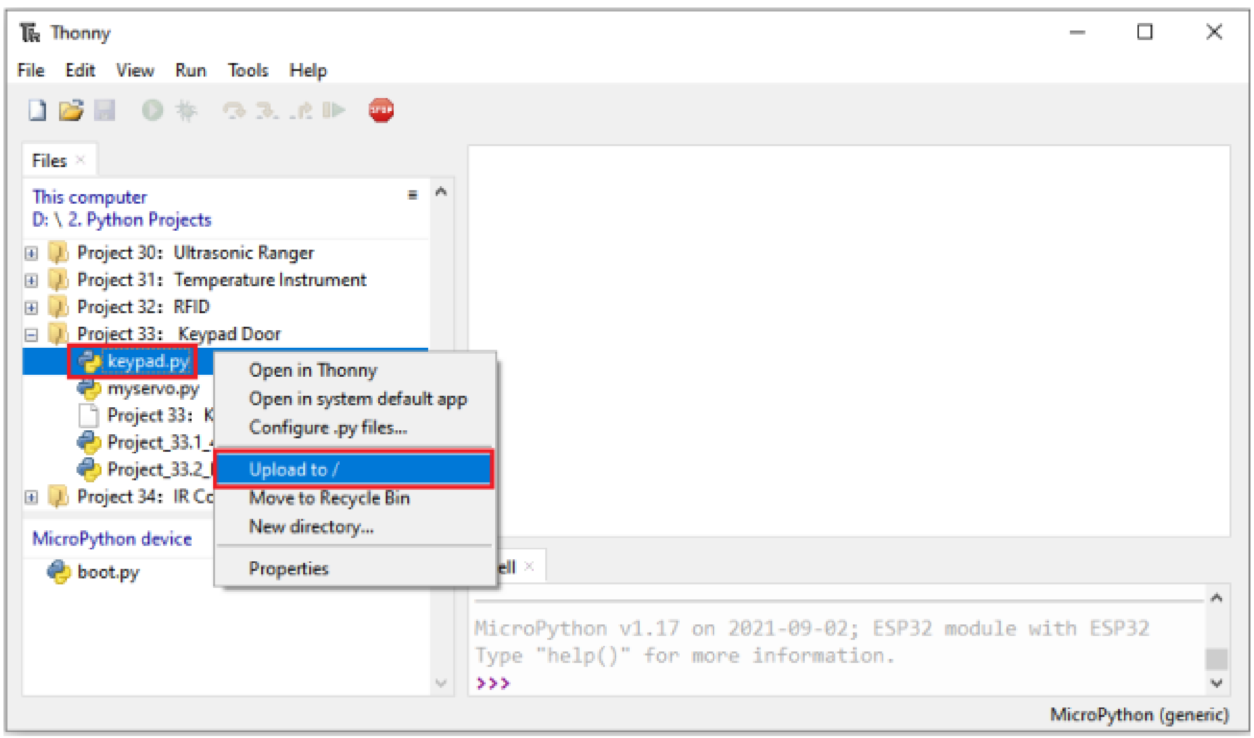


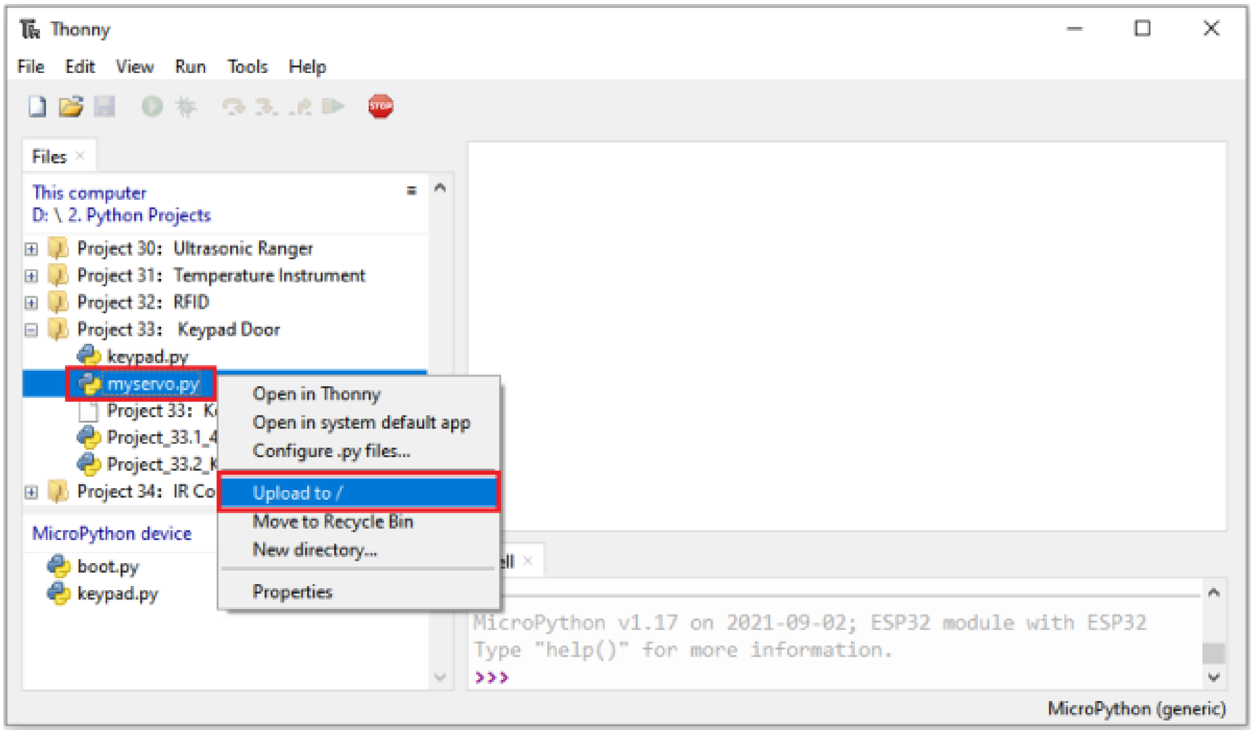
7.34.6 Project code

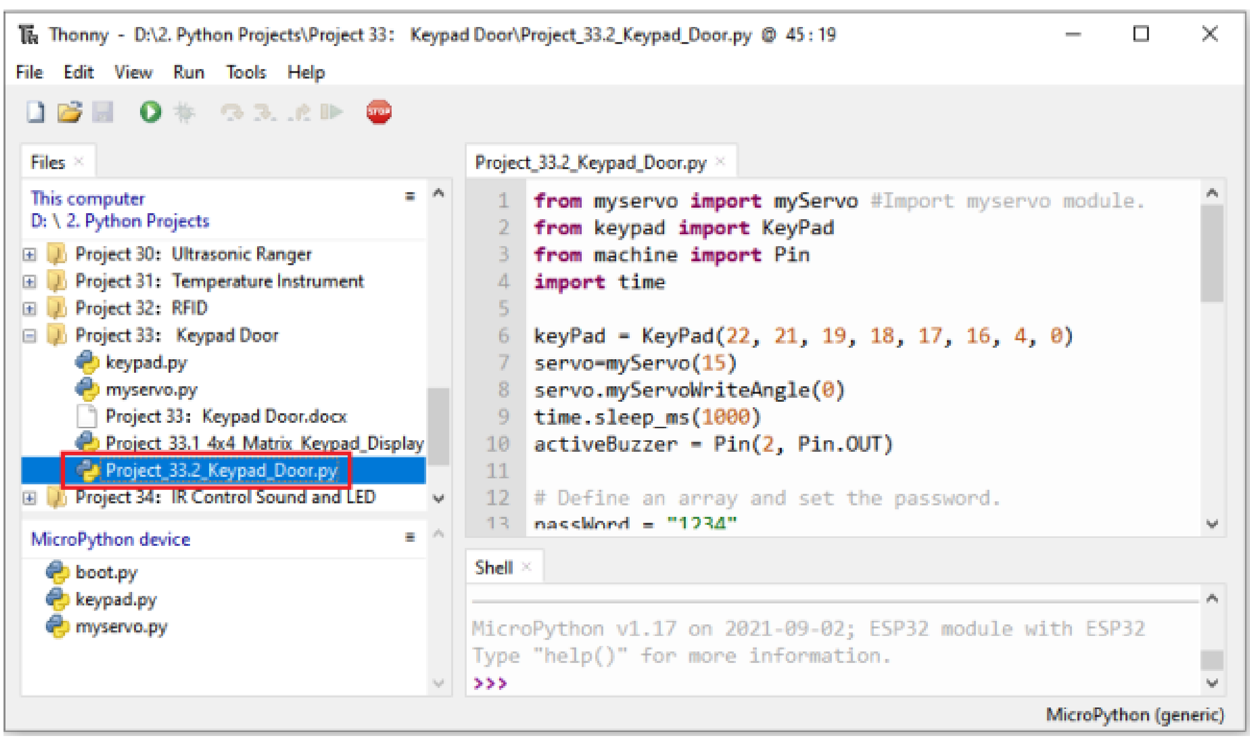
Codes used in this tutorial are saved in"**2. Python Projects**". If you haven't downloaded the code file, please click on the link to download it:Download Python Codes



Open "Thonny", click"This computer"→"D:"→"2. Python Projects"→"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".








```python
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)
```

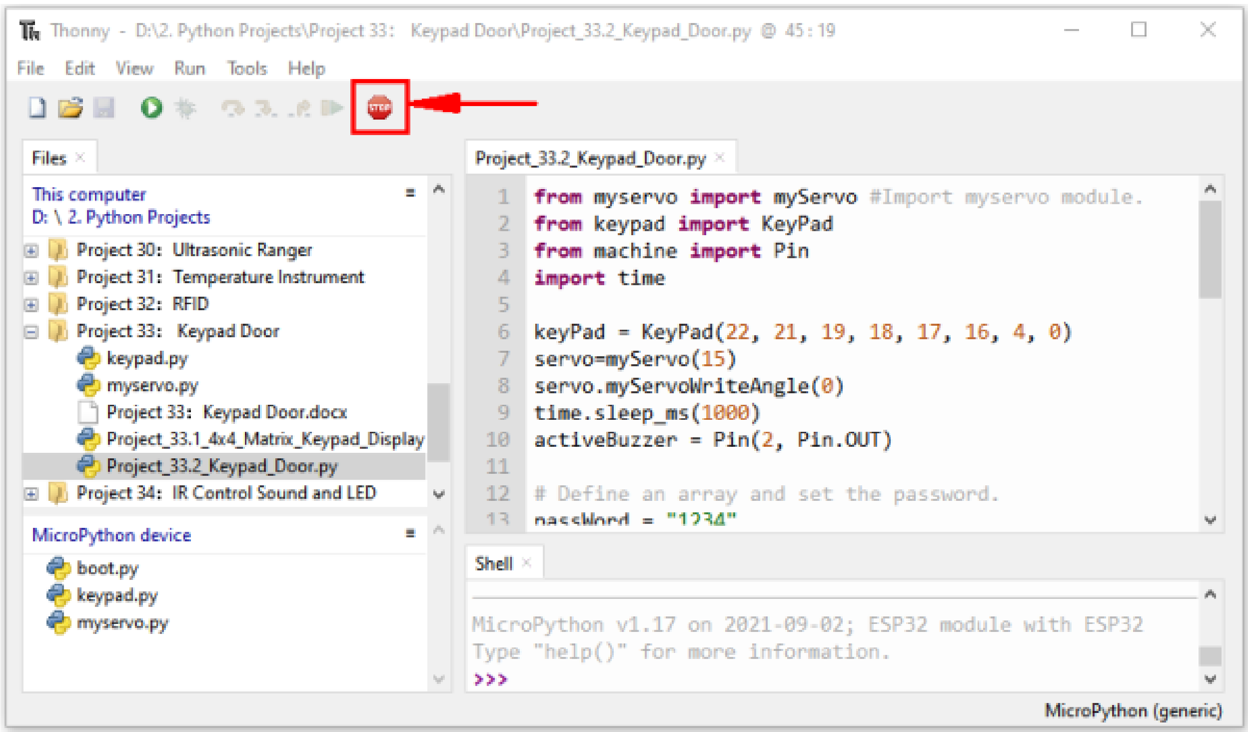
(continues on next page)

(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␣
↪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 = ""
```

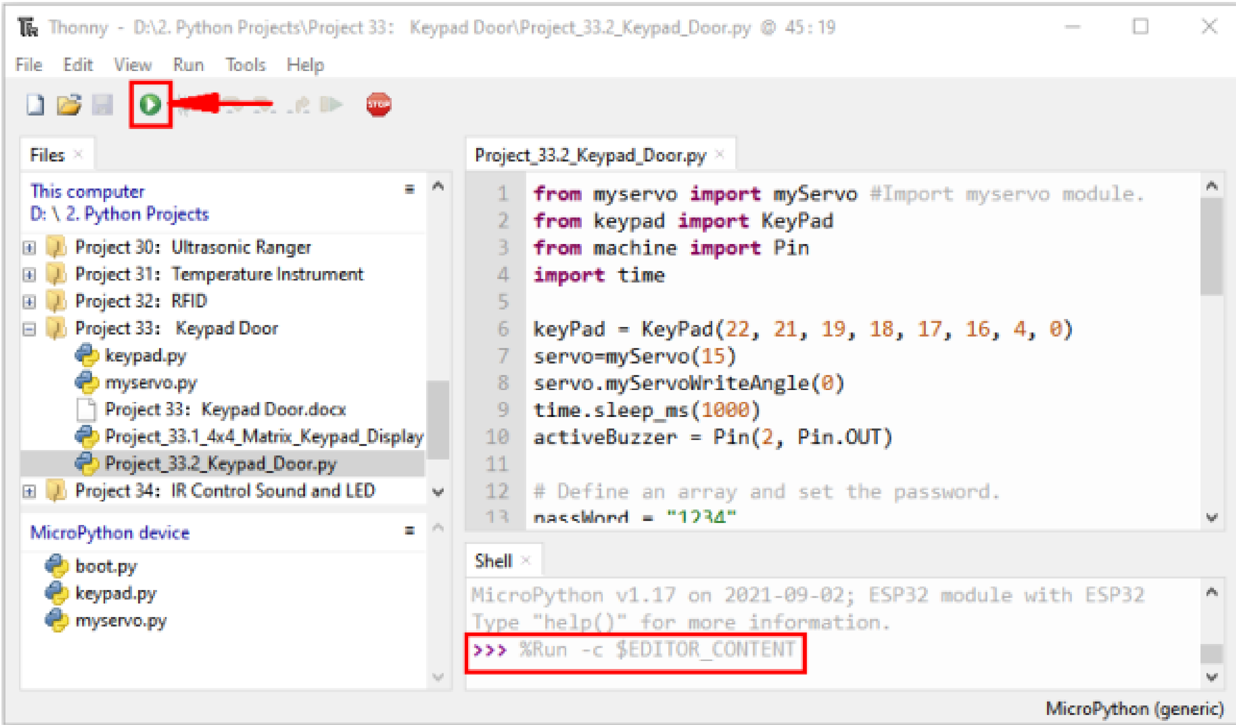
7.34.7 Project result

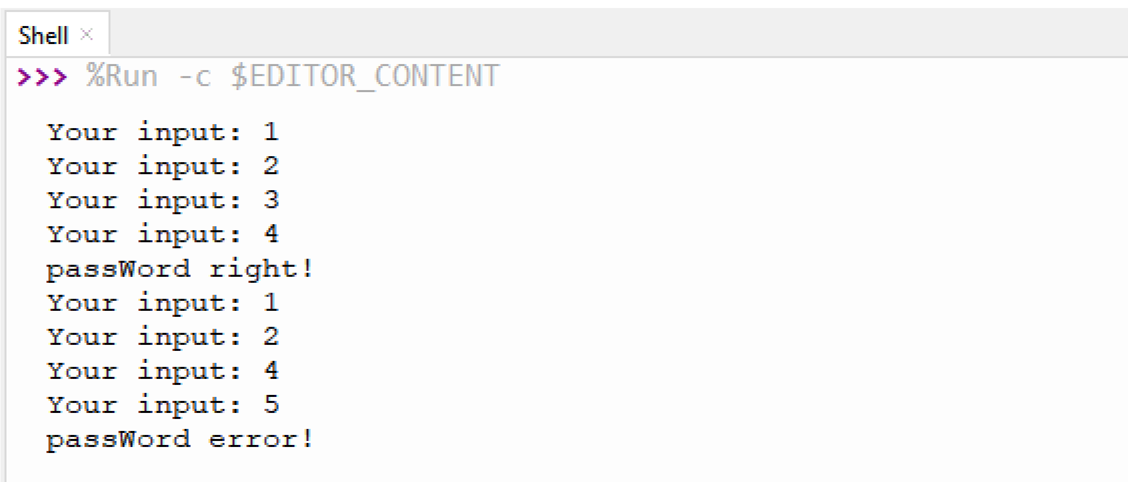
Make sure the ESP32 has been connected to the computer, click “Stop/Restart backend”.



Click “Run current script”, the code starts to be executed and you’ll 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.

Press“Ctrl+C”or click “Stop/Restart backend”to exit the program.





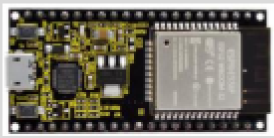
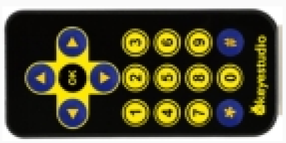
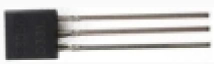
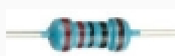
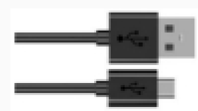
7.35 Project 34IR Control Sound and LED

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

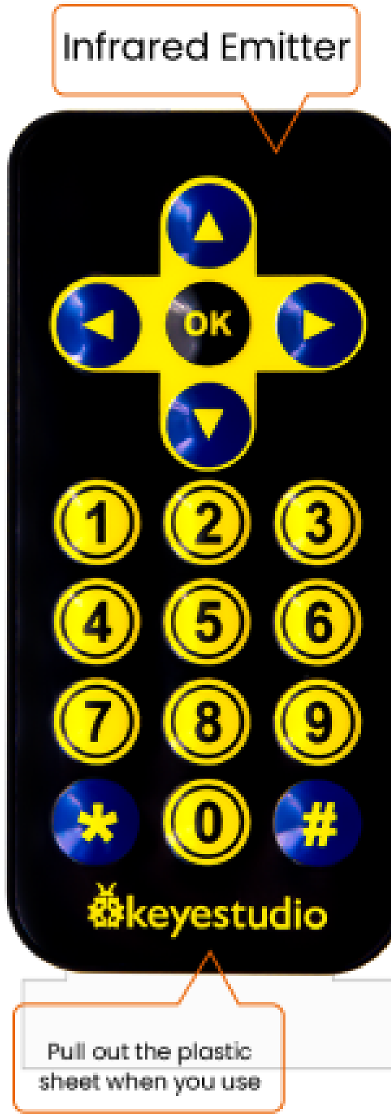
			
ESP32*1	Breadboard*1	IR Receiver *1	RGB LED*1
			
IR Remote Controller*1	Active buzzer*1	10KResistor*1	220Resistor*3
			
NPN transistor(S8050)*1	1kResistor*1	Jumper Wires	USB Cable*1

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:

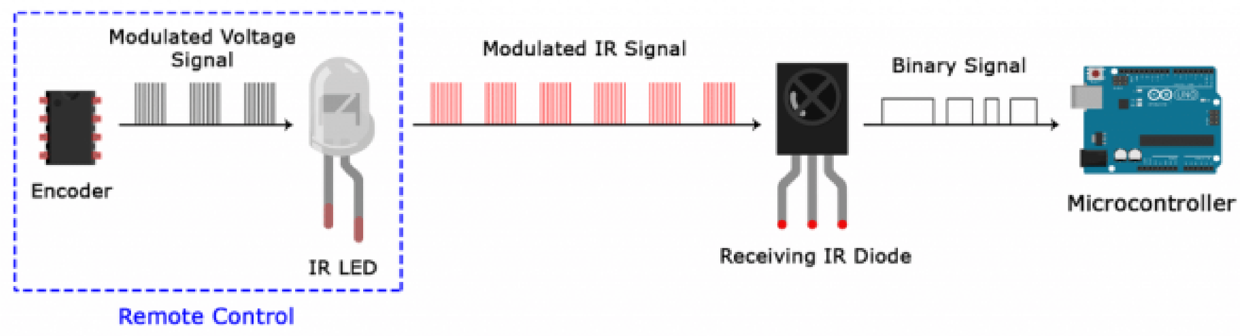
The infrared remote controller adopts NEC code and the signal cycle is 110ms.



Infrared receiverreceiver 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

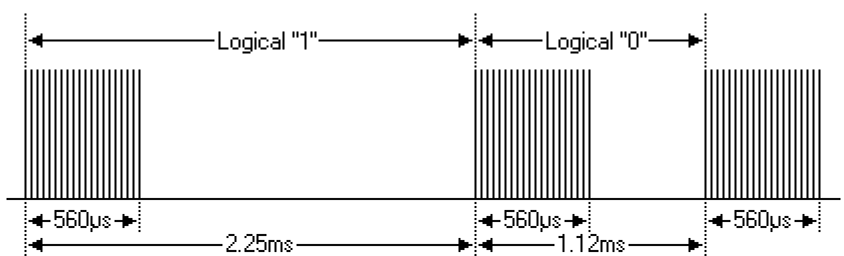
NEC Protocol:

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:

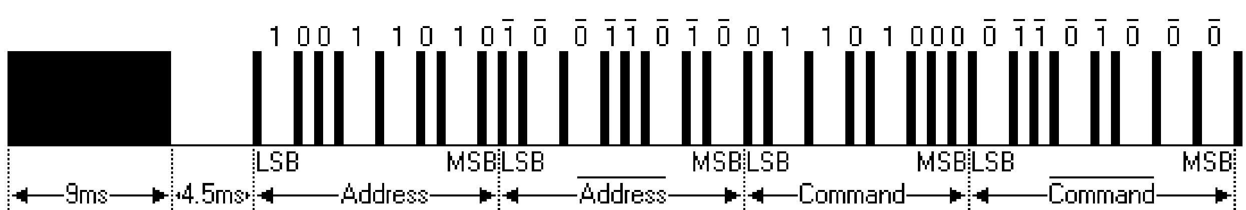
- 8 bit address and 8 bit command length.
- Extended mode available, doubling the address size.
- Address and command are transmitted twice for reliability.
- Pulse distance modulation.
- Carrier frequency of 38kHz.
- Bit time of 1.125ms or 2.25ms.

Modulation:

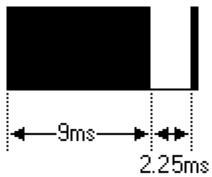


The NEC protocol uses pulse distance encoding of the bits. Each pulse is a 560µs long 38kHz carrier burst (about 21 cycles). A logical "1" takes 2.25ms to transmit, while a logical "0" is only half of that, being 1.125ms. The recommended carrier duty-cycle is 1/4 or 1/3.

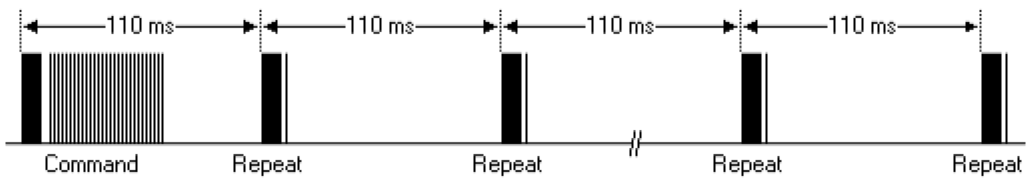
Protocol:



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.

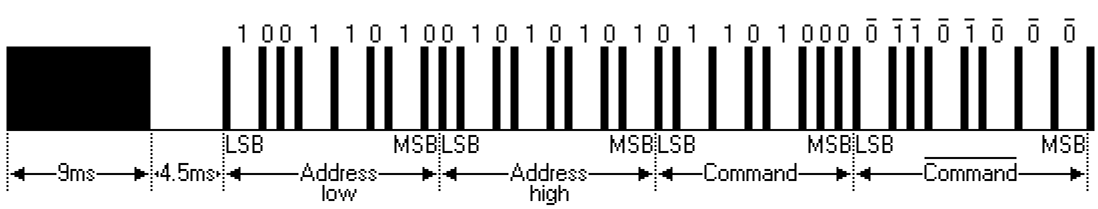


Extended NEC protocol:

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.

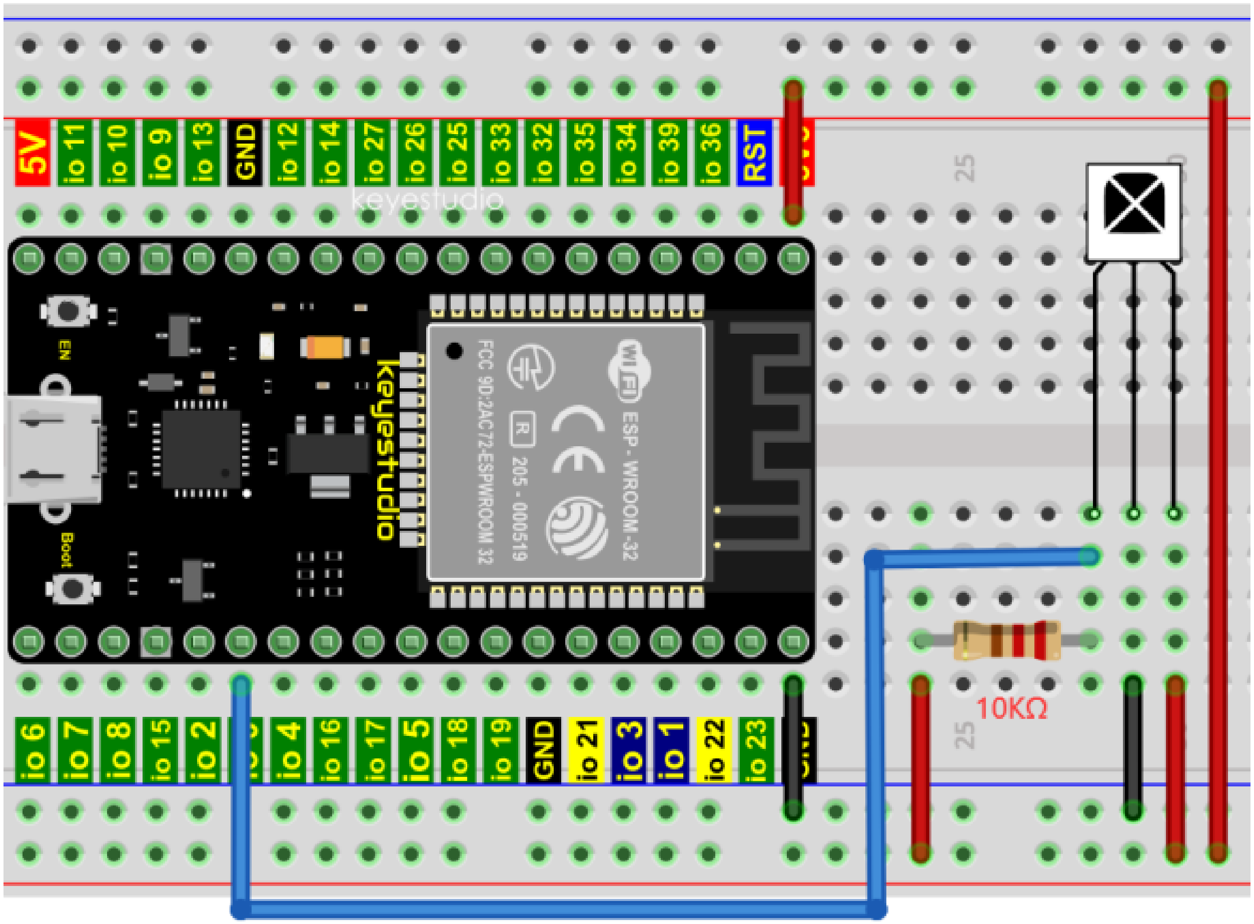
The command redundancy is still preserved. Therefore each address can still handle 256 different commands.



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.

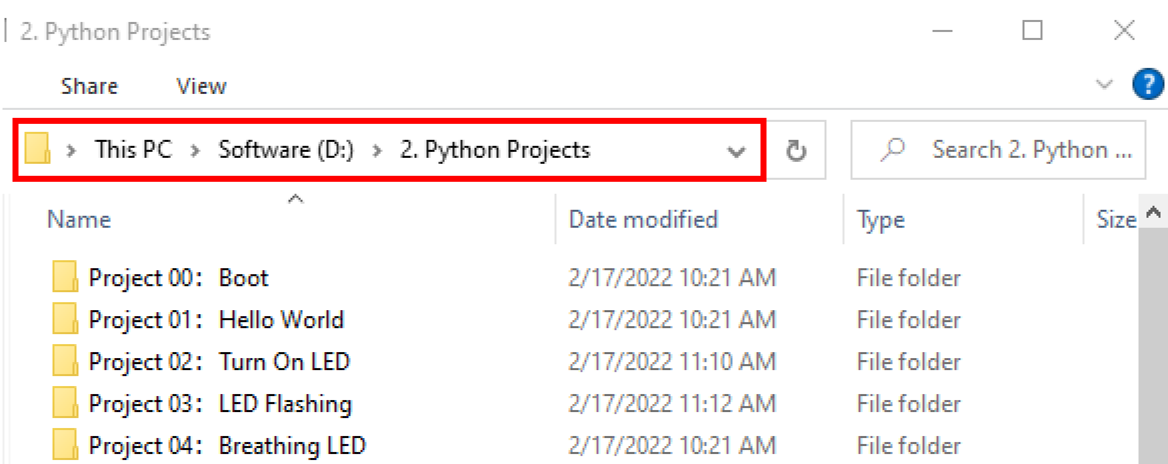
7.35.4 Decoded infrared signal

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



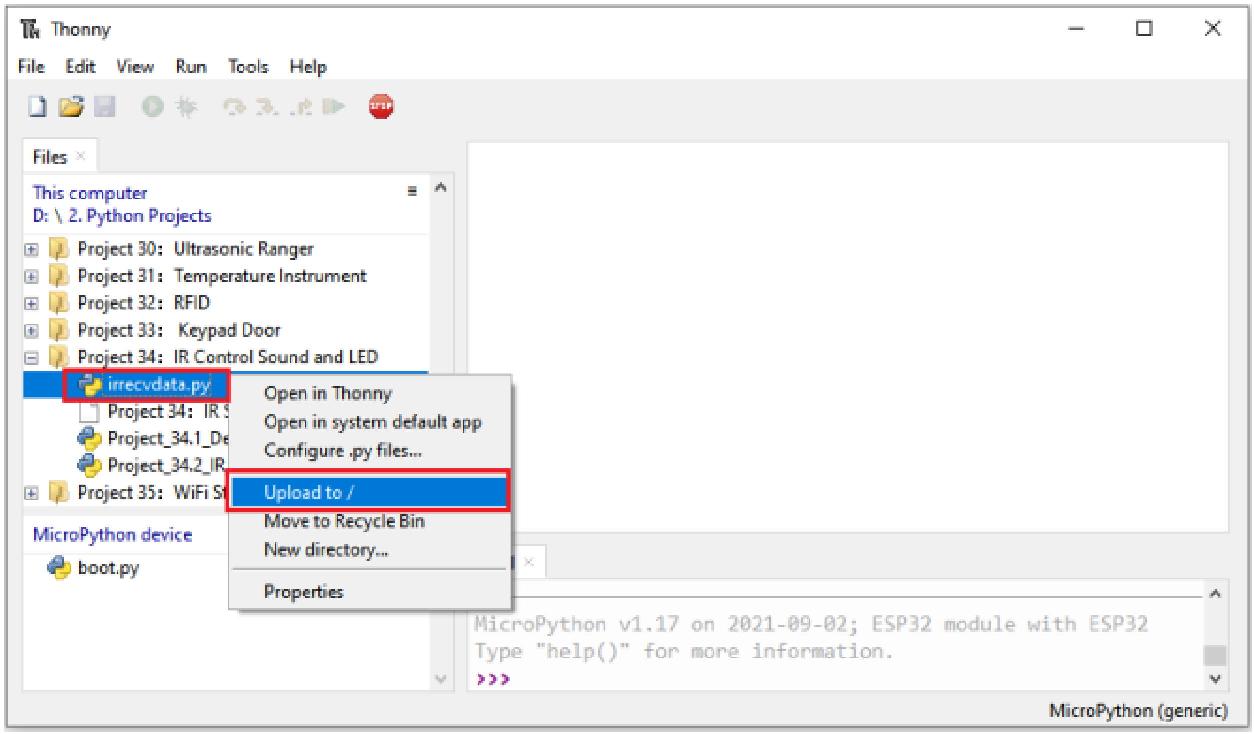
Codes used in this tutorial are saved in“**2. Python Projects**”. If you haven’t downloaded the code file, please click on the link to download it:Download Python Codes

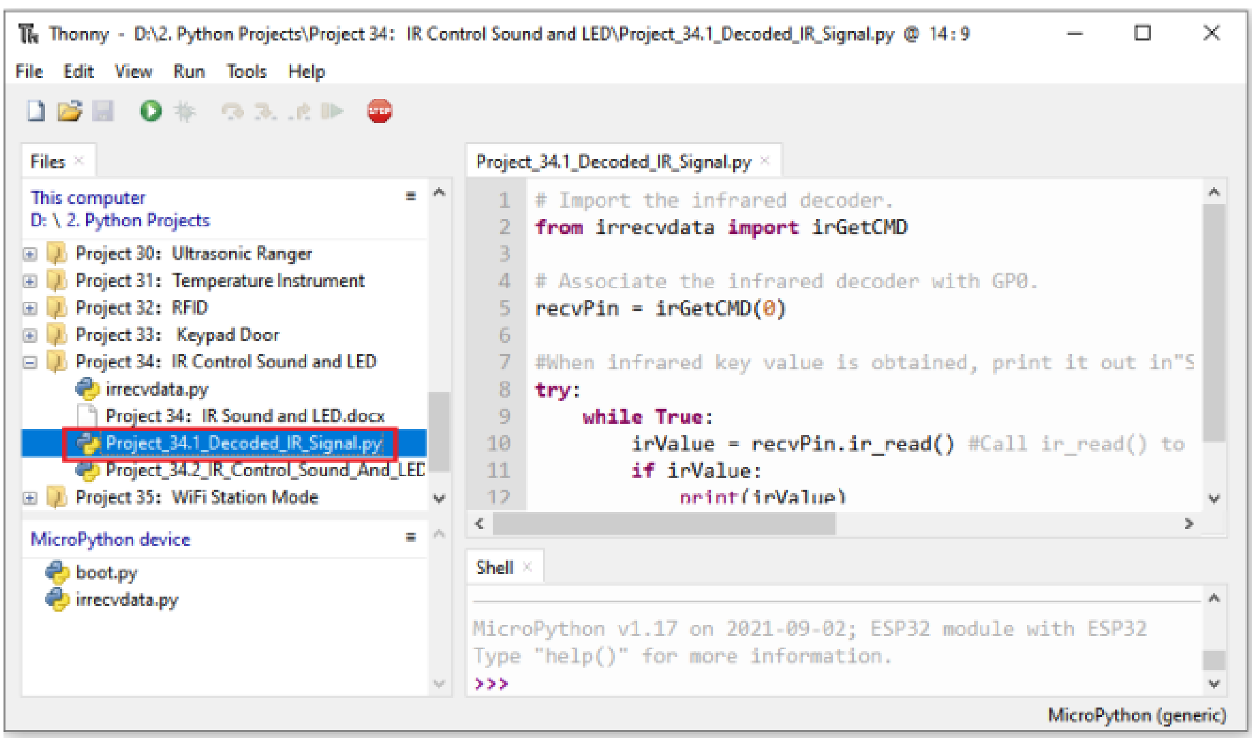
If you haven’t downloaded the code file, please click on the link to download it:Download Python Codes



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".






```python
## Import the infrared decoder.
from irrecvdata import irGetCMD
```

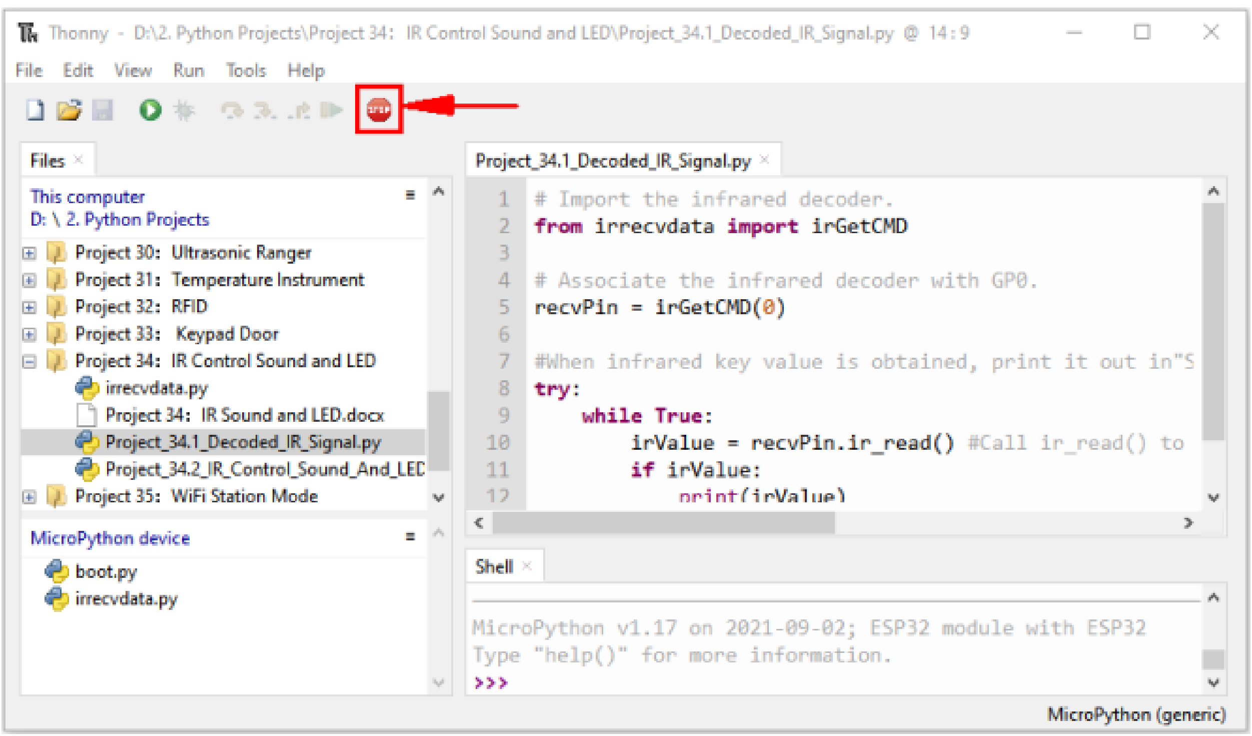
(continues on next page)

(continued from previous 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

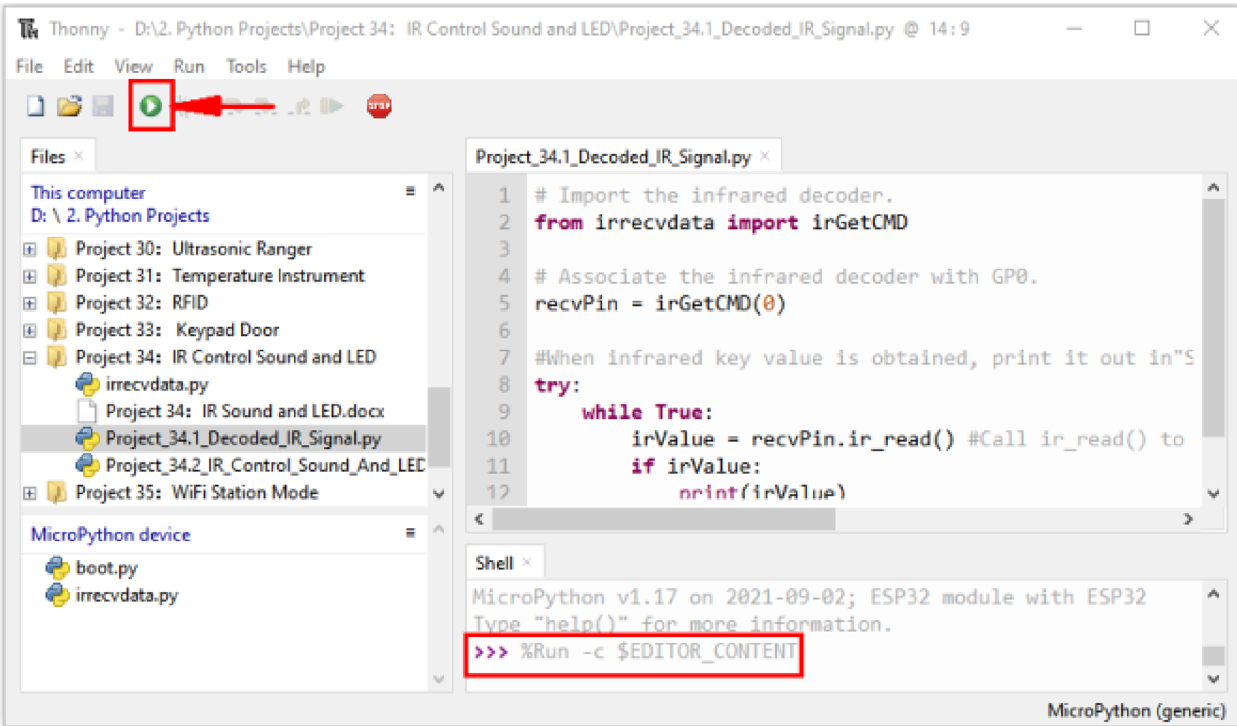
Make sure the ESP32 has been connected to the computer, click “Stop/Restart backend”.

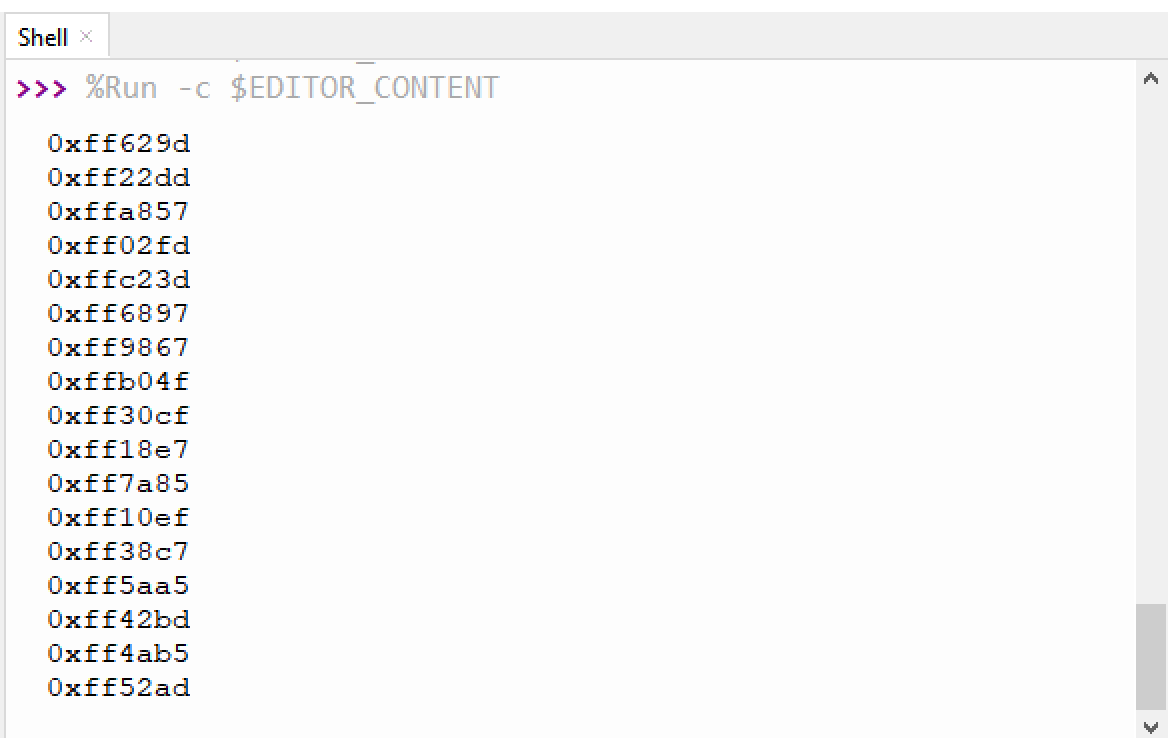


media/da852227207616ccd9aff28f19e02690.png

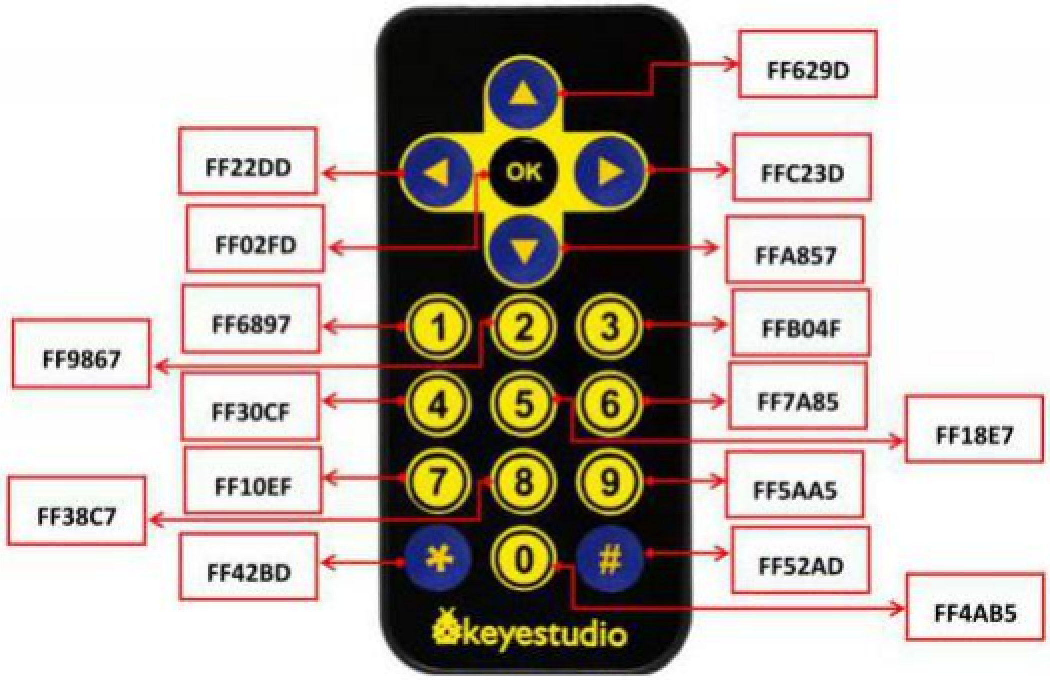
Click “Run current script”, the code starts to be executed and you’ll see that aim the infrared remote control transmitter at the infrared receiving head, press the button on the infrared controller, and the “Shell” window of Thonny IDE prints the current received key code values.

Press"Ctrl+C"or click media/27451c8a9c13e29d02bc0f5831cfaf1f.png "Stop/Restart backend"to exit the program.

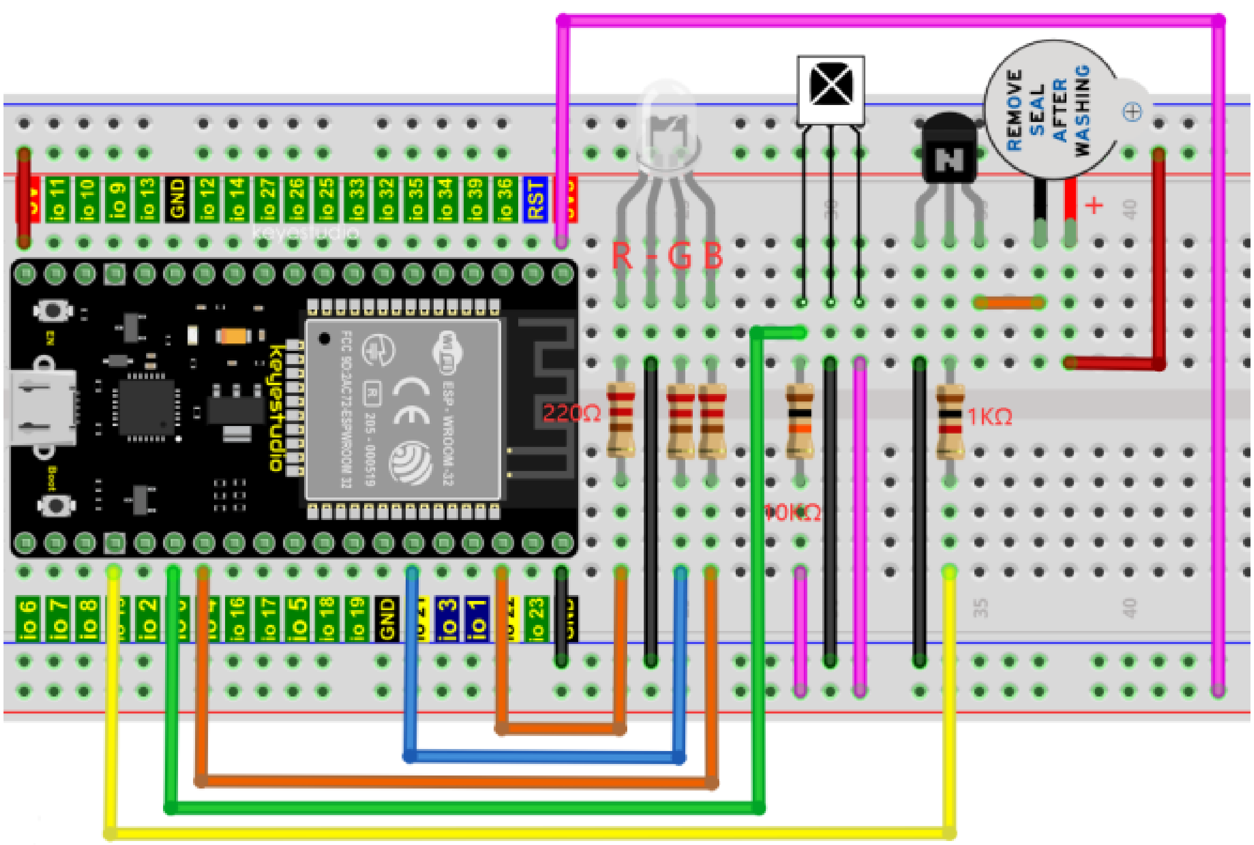




Write down the code associated with each button, because you will need that information later.

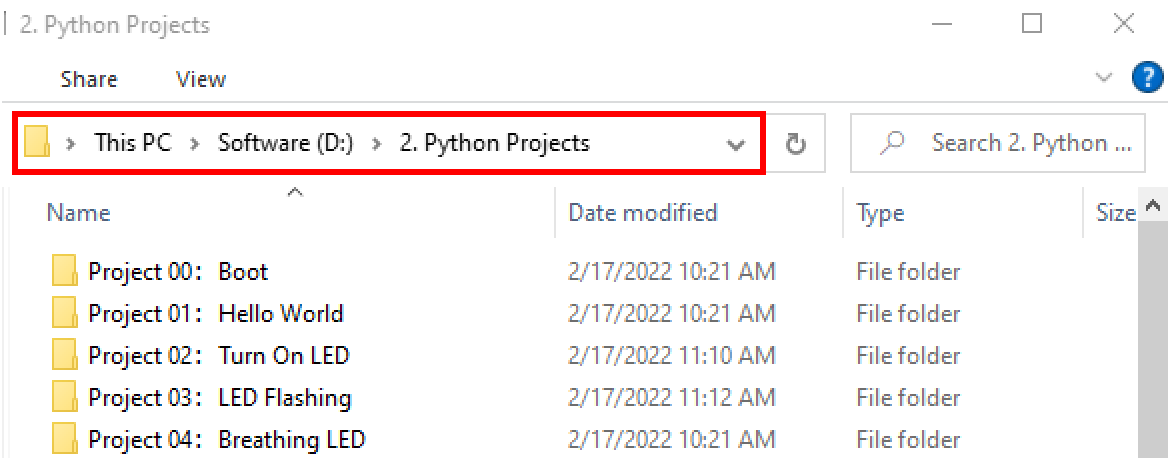


7.35.5 Wiring diagram of the infrared remote control

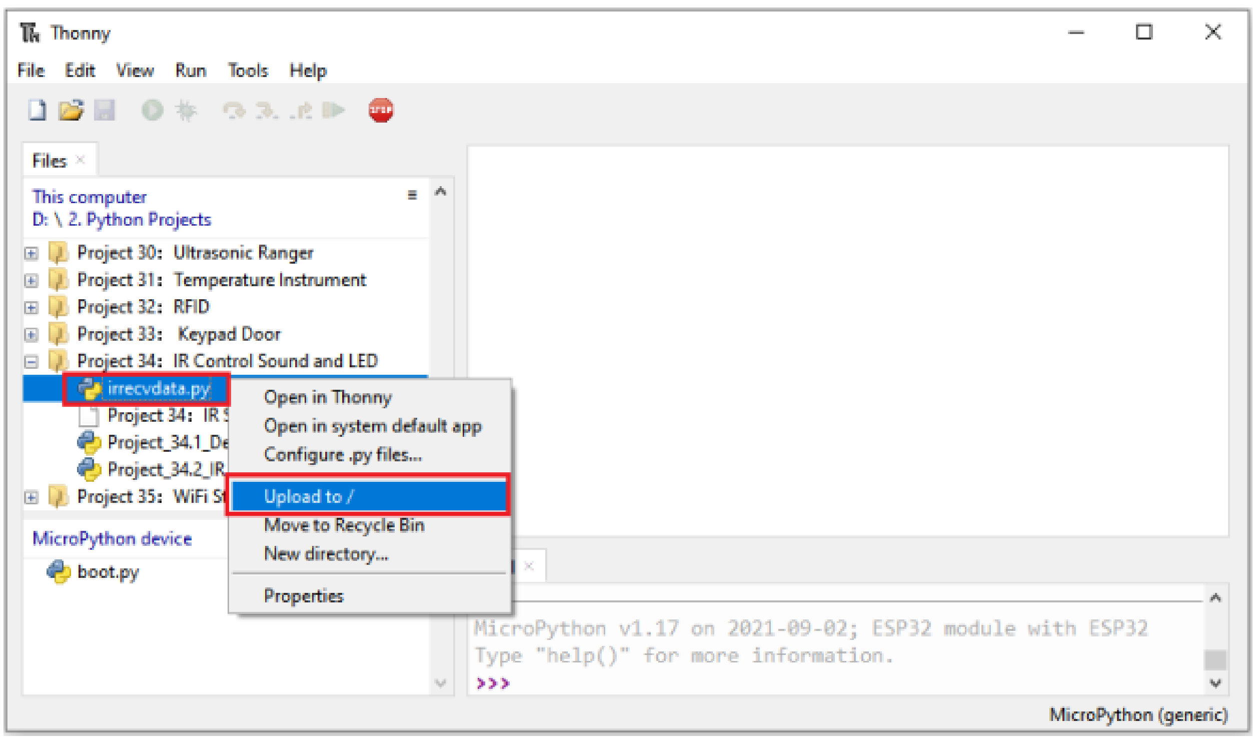


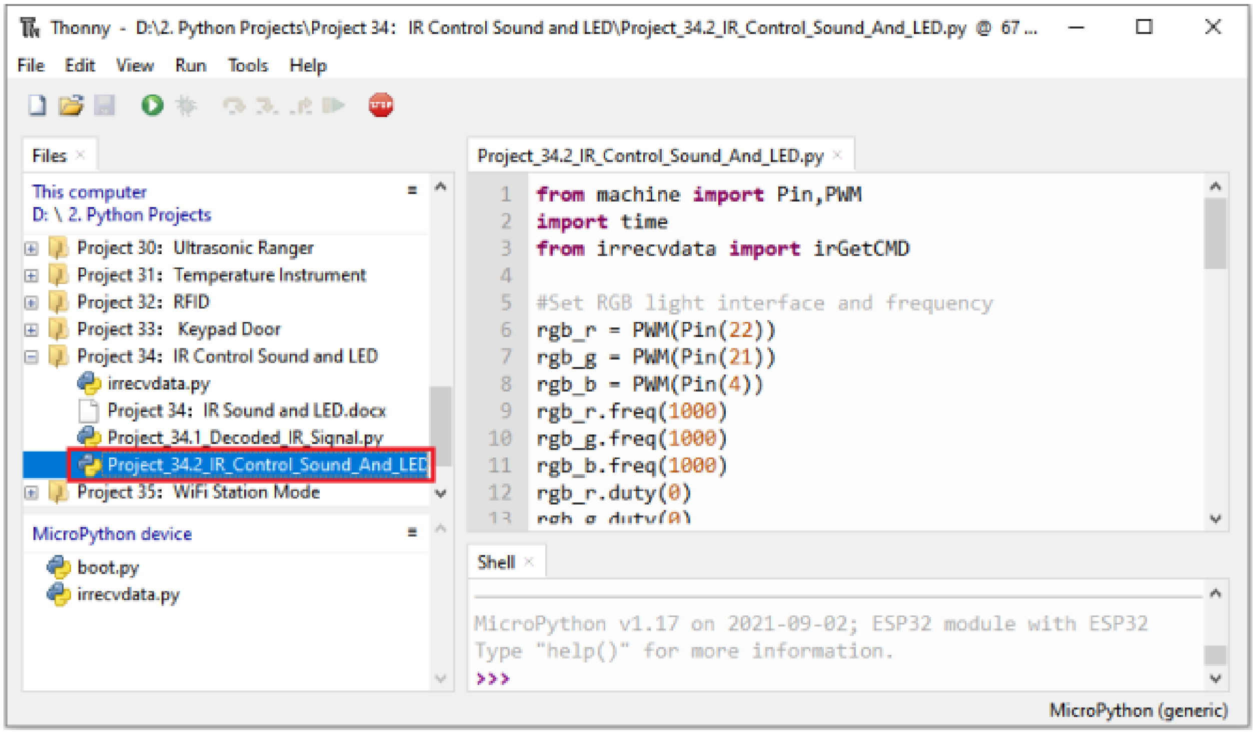
7.35.6 Project code

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



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_34.2_IR_Control_Sound_And_LED.py”.






```
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(0)
rgb_b.duty(0)
## 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
```

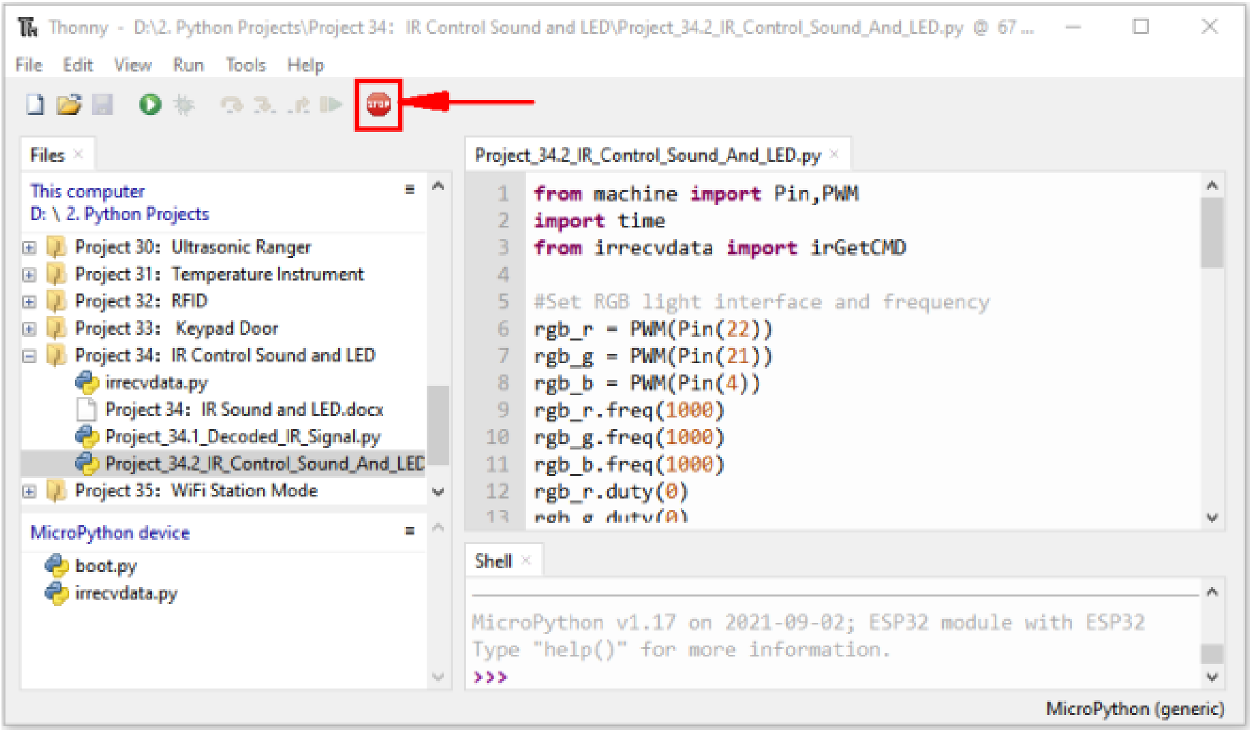
(continues on next page)

(continued from previous page)

```
           rgb_r.duty(1023)
           rgb_g.duty(0)
           rgb_b.duty(0)
           print('1')
        elif irValue == '0xff9867': #2
            rgb_r.duty(0)
            rgb_g.duty(1023)
            rgb_b.duty(0)
            print('2')
        elif irValue == '0xffb04f': #3
            rgb_r.duty(0)
            rgb_g.duty(0)
            rgb_b.duty(1023)
            print('3')
        elif irValue == '0xff30cf': #4
            rgb_r.duty(1023)
            rgb_g.duty(1023)
            rgb_b.duty(0)
            print('4')
        elif irValue == '0xff18e7': #5
            rgb_r.duty(1023)
            rgb_g.duty(0)
            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(0)
            rgb_g.duty(0)
            rgb_b.duty(0)
```

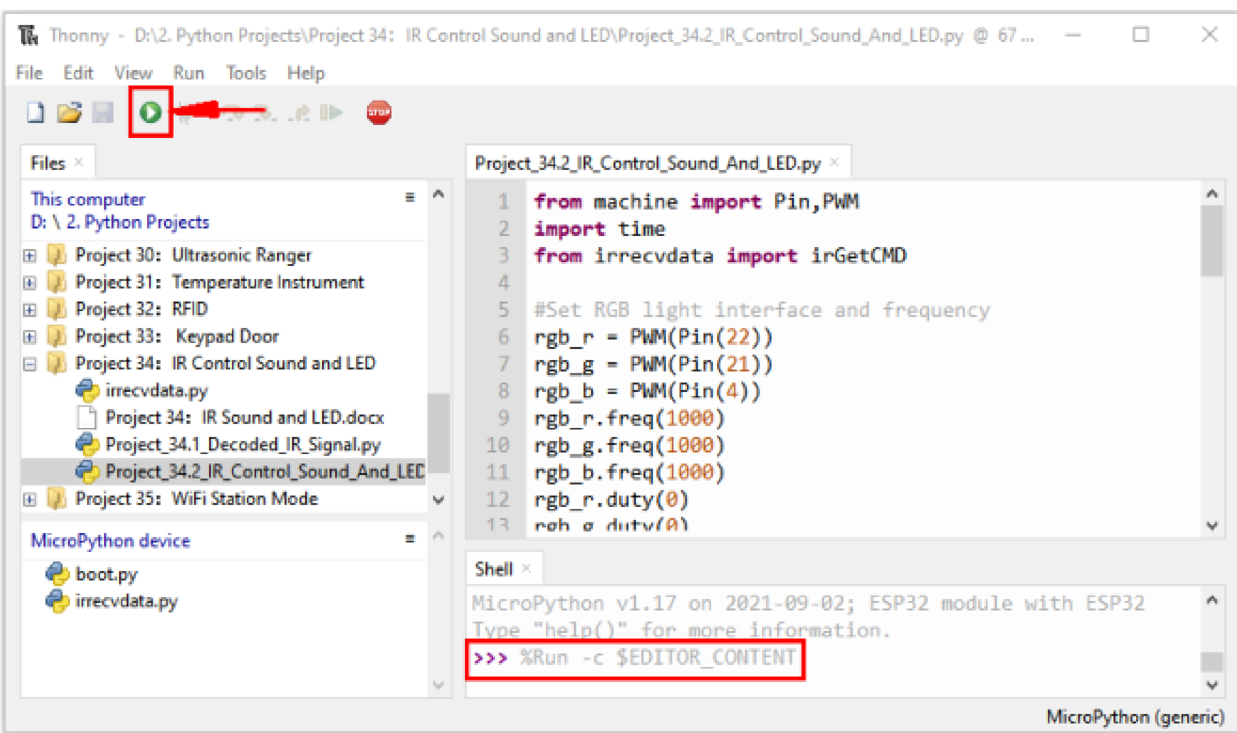
7.35.7 Project result

Make sure the ESP32 has been connected to the computer, click “Stop/Restart backend”.

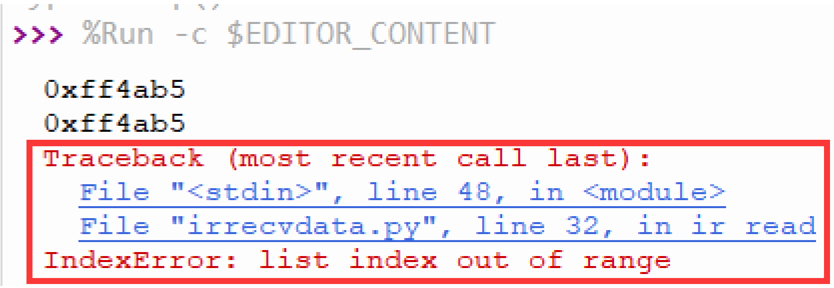


Click "Run current script", the code starts to be executed and you'll see that press the 1 to 7 key of the infrared remote controller, the buzzer will sound once, and the RGB light will be red, green, blue, yellow ,red, blue ,green and white respectively. Press another key (except 1 to 7 key), and the RGB light will go off.

Press"Ctrl+C"or click "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“Run current script” to make the code run again.



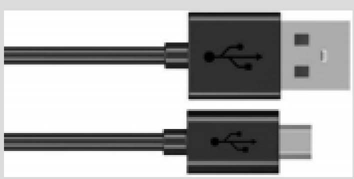
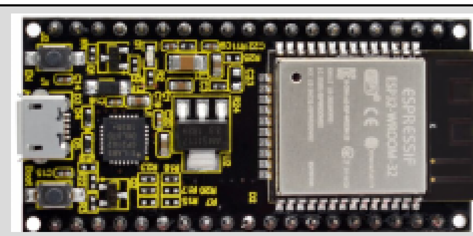
(Note: Before use, we need to remove the plastic sheet from the bottom of the infrared remote controller.)

7.36 Project 35WiFi Station Mode

7.36.1 Introduction

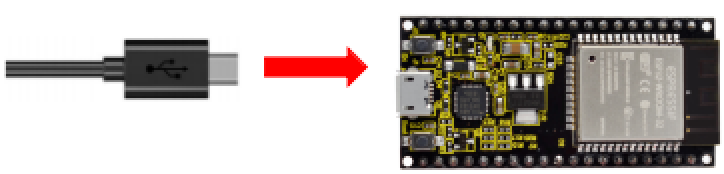
ESP32 has three different WiFi operating modes : Station modeAP mode and AP+Station mode. All WiFi programming projects must be configured with WiFi operating mode before using WiFi, otherwise WiFi cannot be used. In this project, we will learn about ESP32’s WiFi Station mode.

7.36.2 Components

	
USB Cable*1	ESP32*1

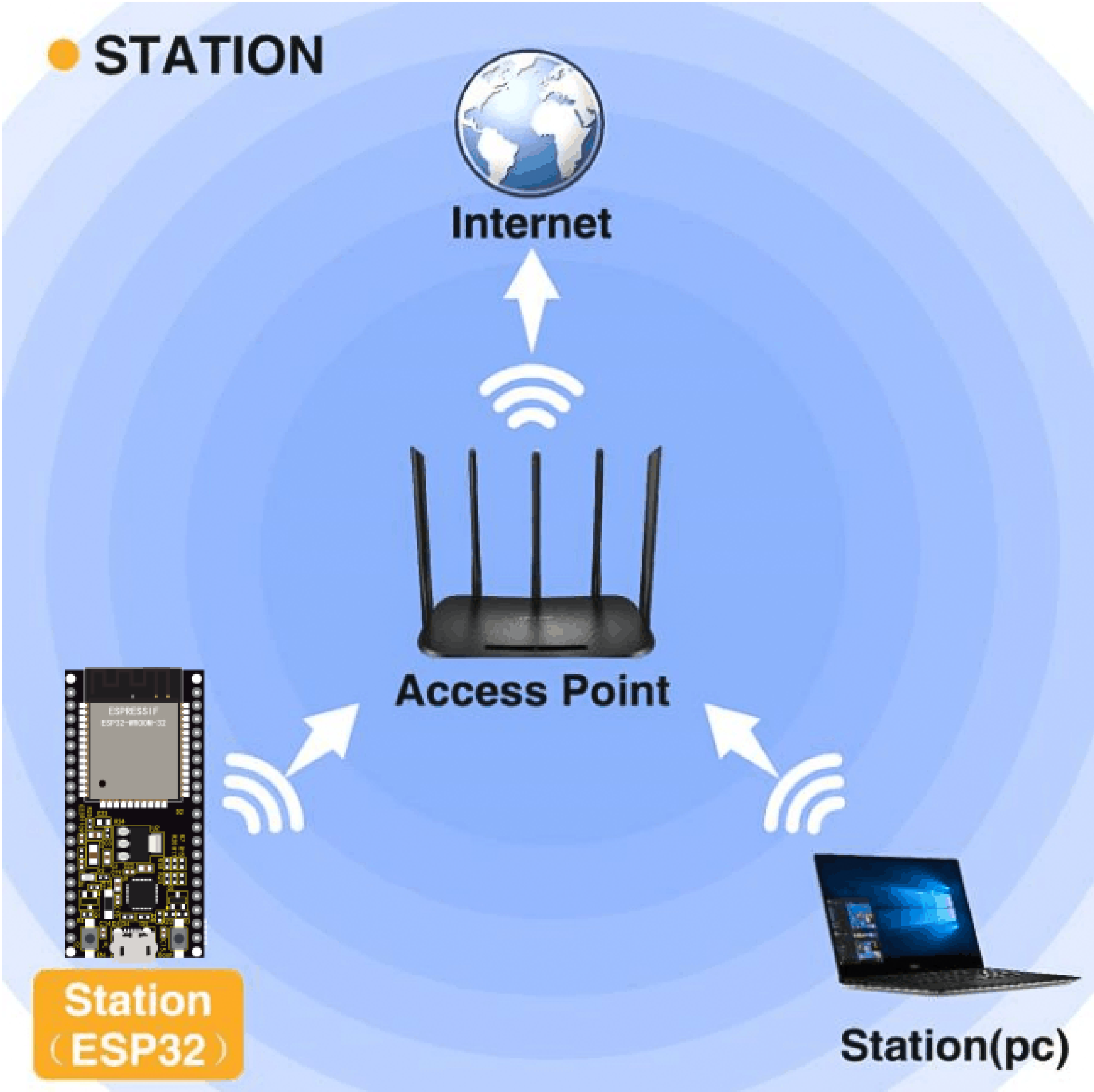
7.36.3 Project wiring

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



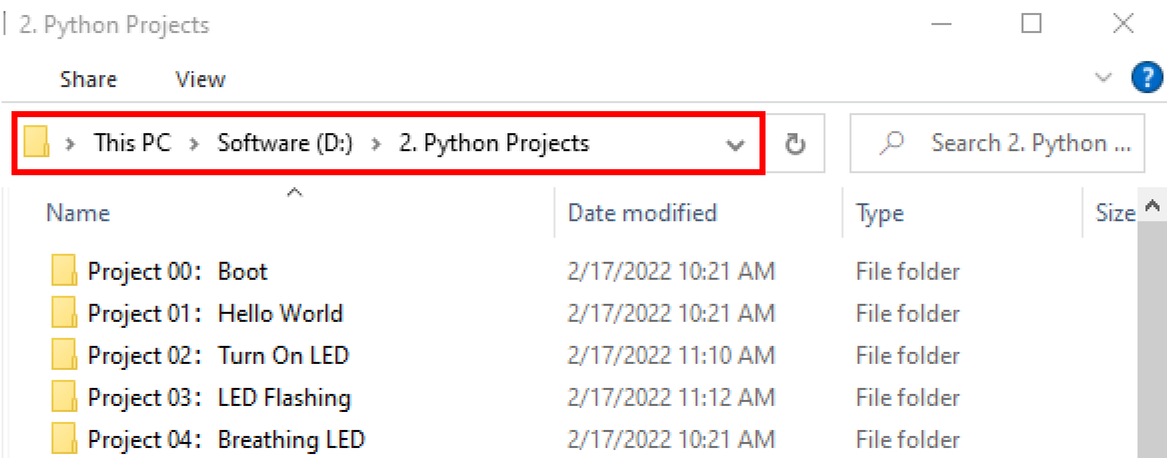
7.36.4 Component knowledge

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.

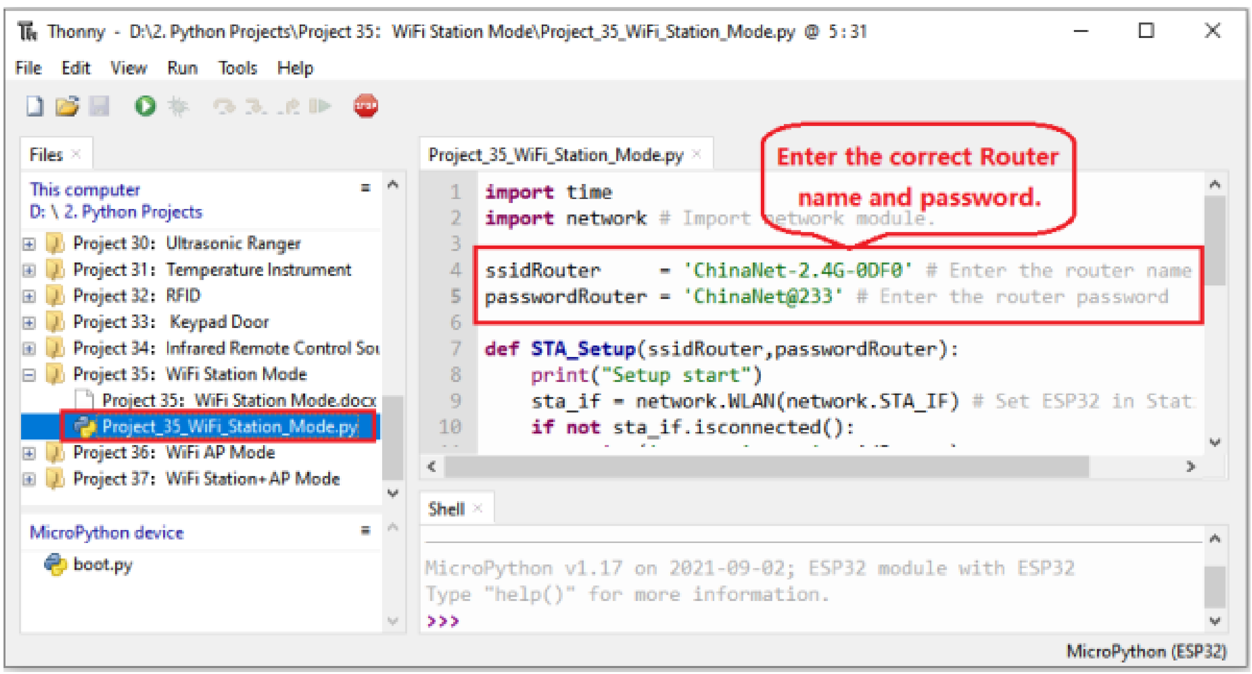


7.36.5 Project code

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



Open“Thonny”, click “This computer” → “D:” → “2. Python Projects” → “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)

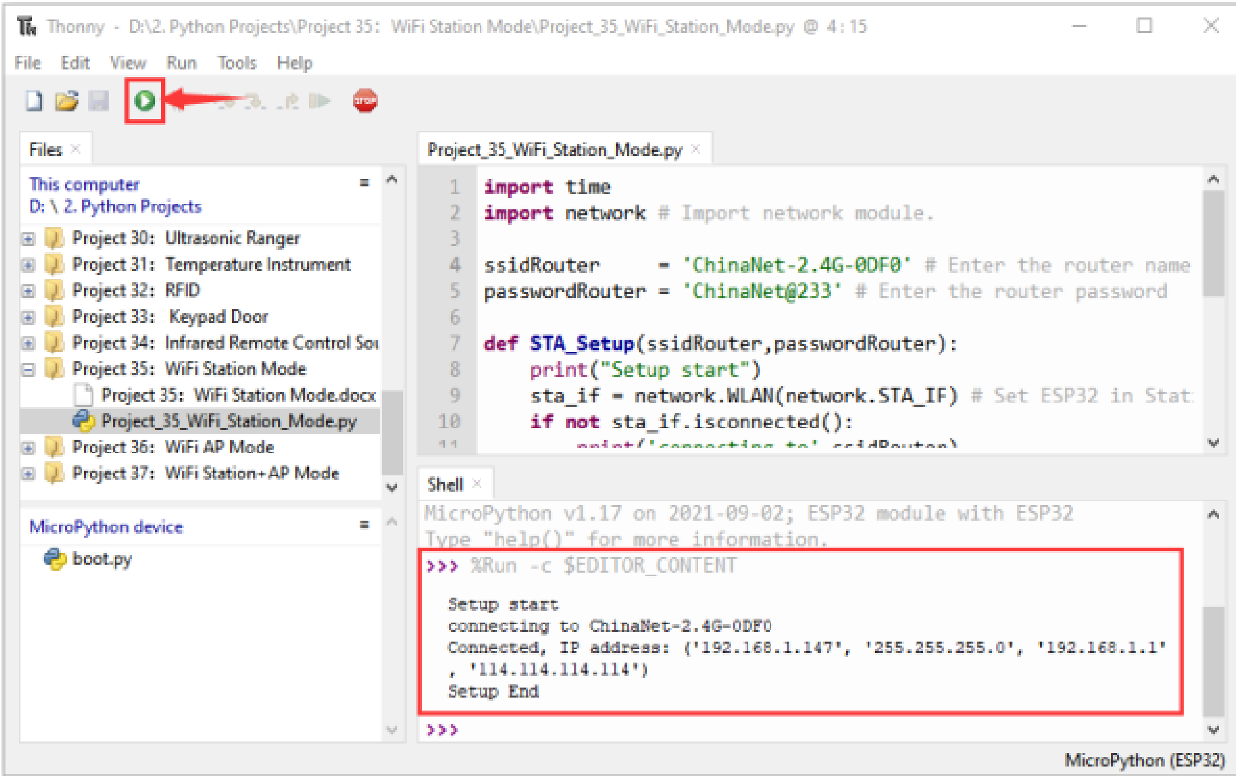
(continued from previous page)

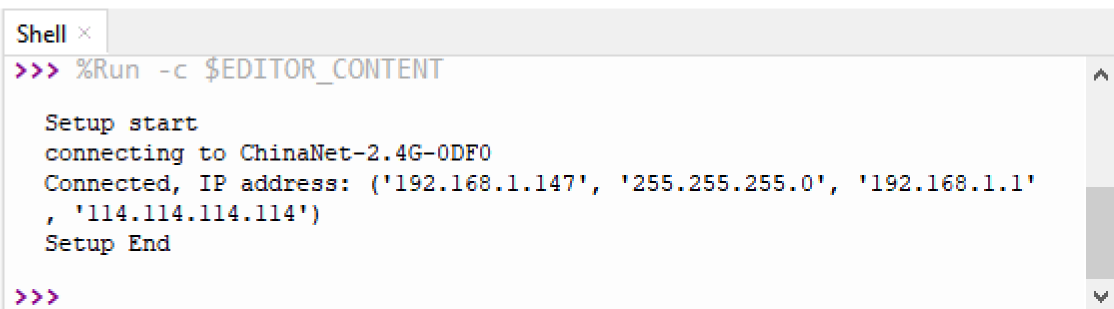
```
    if not sta_if.isconnected():
        print('connecting to',ssidRouter)
  # Activate ESP32's Station mode, initiate a connection request to the router
  # and enter the password to connect.
        sta_if.active(True)
        sta_if.connect(ssidRouter,passwordRouter)
  #Wait for ESP32 to connect to router until they connect to each other successfully.
        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  "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.



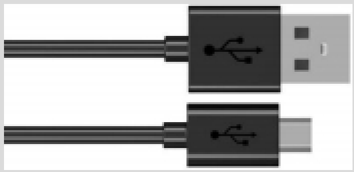
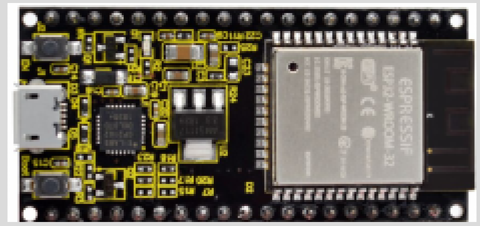


7.38 Project 36WiFi AP Mode

7.39 Introduction

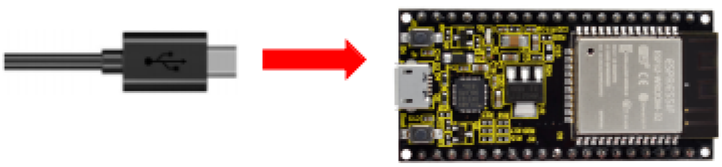
ESP32 has three different WiFi operating modes : Station modeAP mode and AP+Station mode. All WiFi programming projects must be configured with WiFi operating mode before using WiFi, otherwise WiFi cannot be used. In this project, we will learn about ESP32's WiFi AP mode.

7.39.1 Components

	
USB Cable*1	ESP32*1

7.40 Project wiring

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



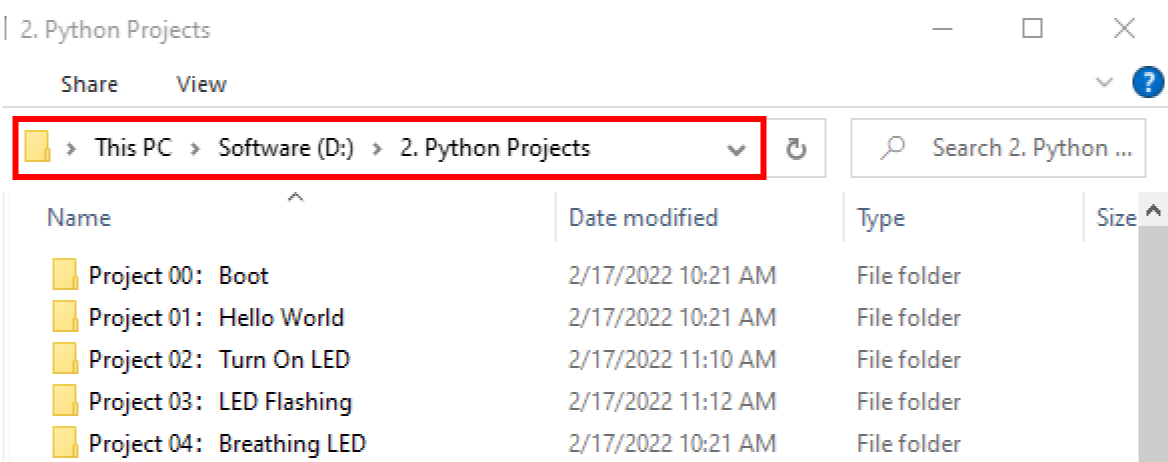
7.40.1 Component knowledge

AP mode :

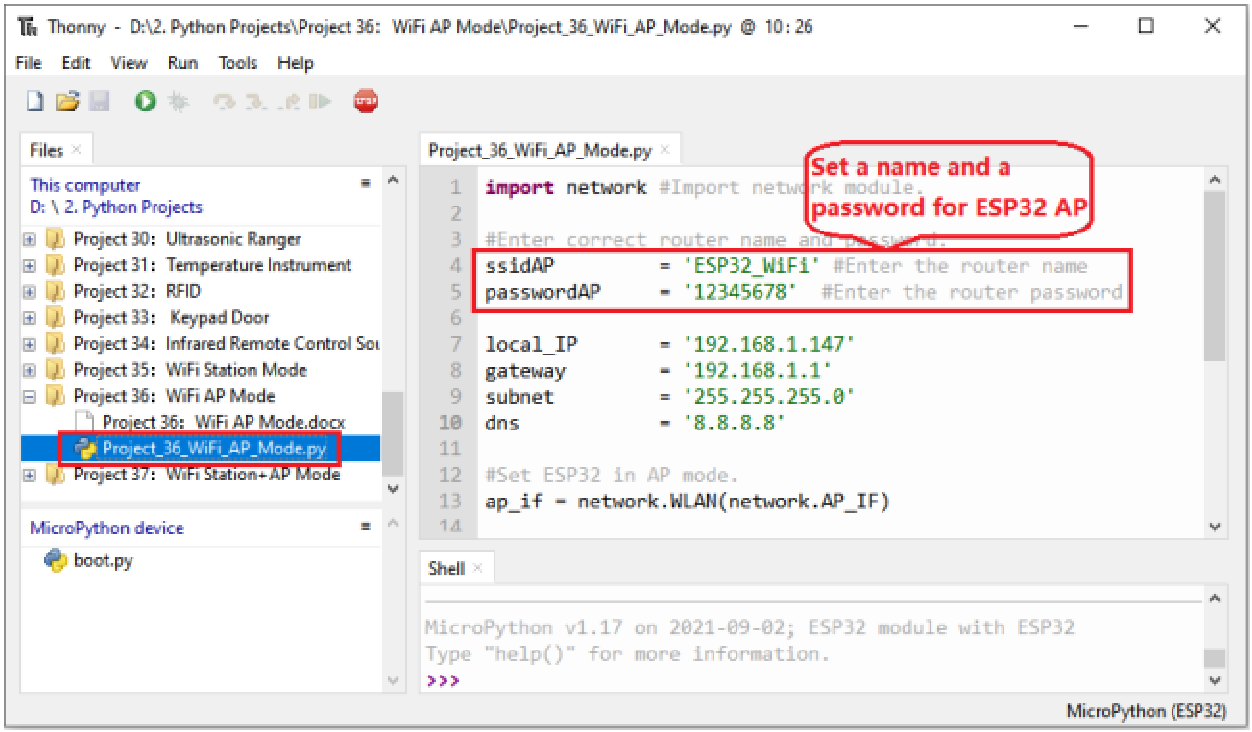
When ESP32 selects AP mode, it creates a hotspot network that is separated from the Internet and waits for other WiFi devices to connect. As shown in the figure below, ESP32 is used as a hotspot. If a mobile phone or PC wants to communicate with ESP32, it must be connected to the hotspot of ESP32. Only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



Open“Thonny”click“This computer”→“D:”→“2. Python Projects”→“Project 36WiFi AP Mode”, and double left-click “Project_36_WiFi_AP_Mode.py”.

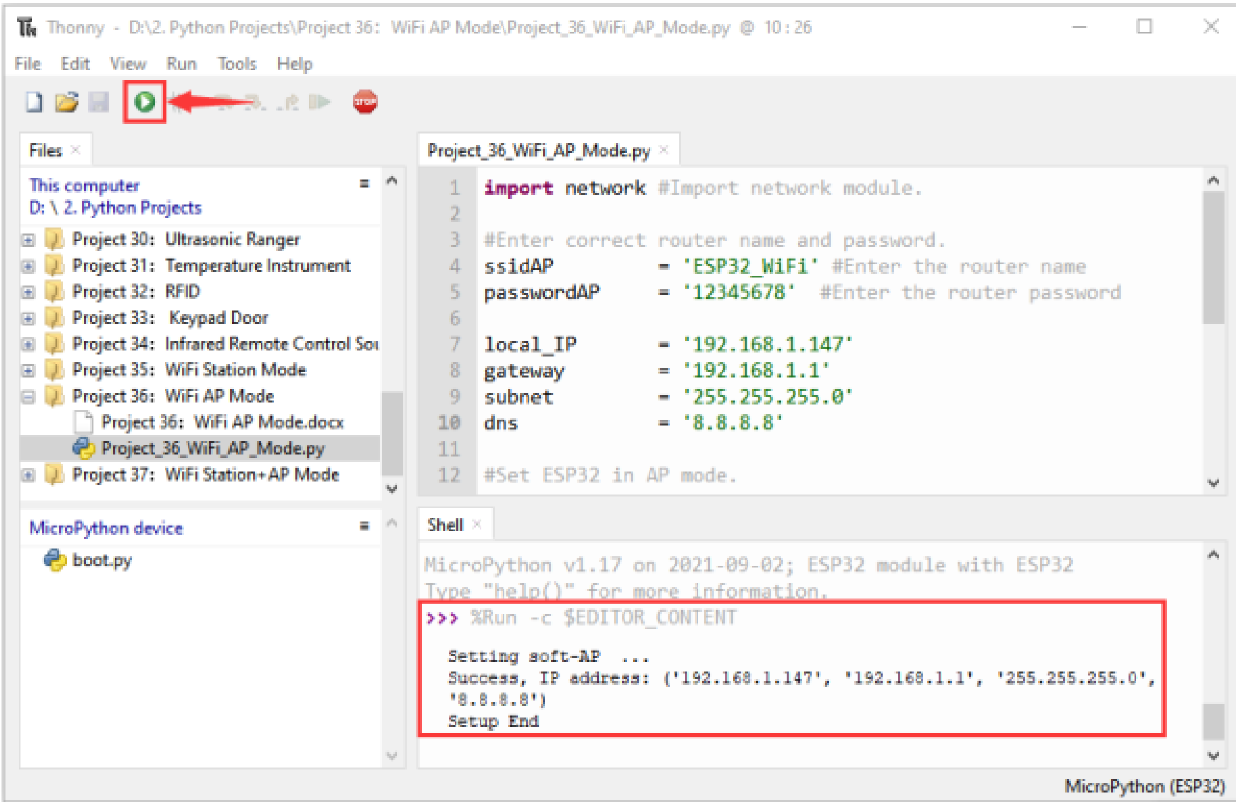


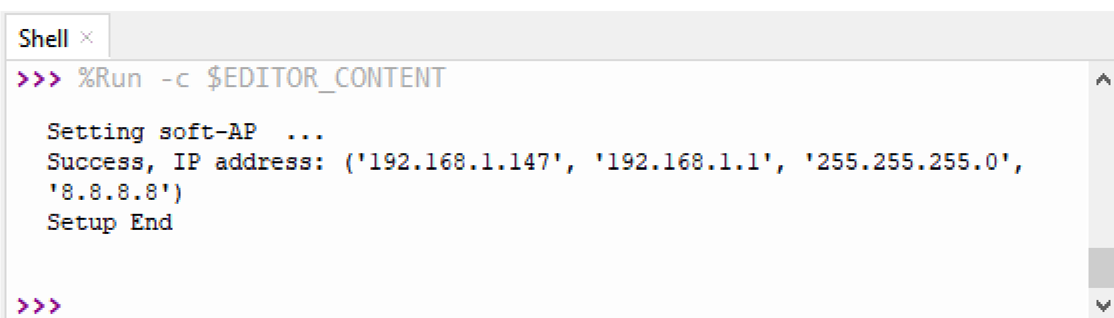

```
import network #Import network module.

#Enter correct router name and password.
ssidAP          = 'ESP32_WiFi' #Enter the router name
passwordAP      = '12345678'  #Enter the router password

local_IP        = '192.168.1.147'
gateway         = '192.168.1.1'
subnet          = '255.255.255.0'
dns             = '8.8.8.8'

#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

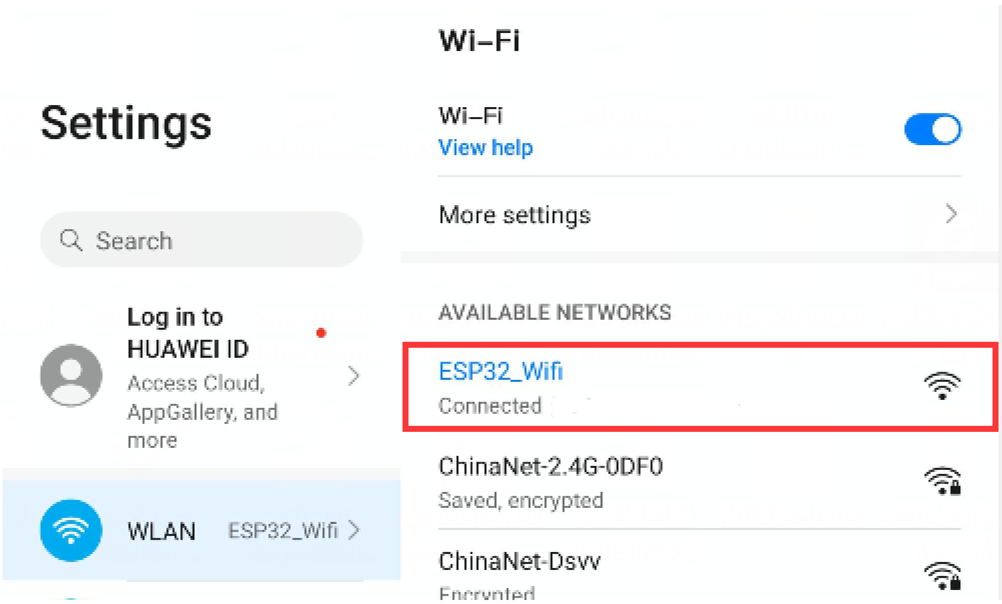
Before the code runs, you can make any changes to the AP name and password for ESP32 in the box as shown in the illustration above. Of course, you can leave it alone by default.

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.





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.

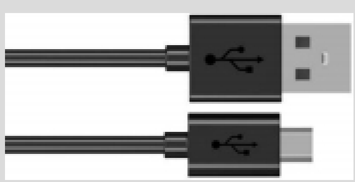
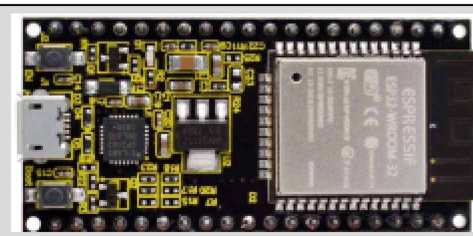


7.42 Project 37WiFi Station+AP Mode

7.42.1 Introduction

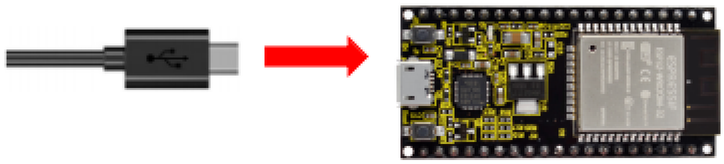
ESP32 has three different WiFi operating modes : Station modeAP mode and AP+Station mode. All WiFi programming projects must be configured with WiFi operating mode before using WiFi, otherwise WiFi cannot be used. In this project, we will learn ESP32's WiFi Station+AP mode.

7.42.2 Components

	
USB Cable*1	ESP32*1

7.42.3 Project wiring

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

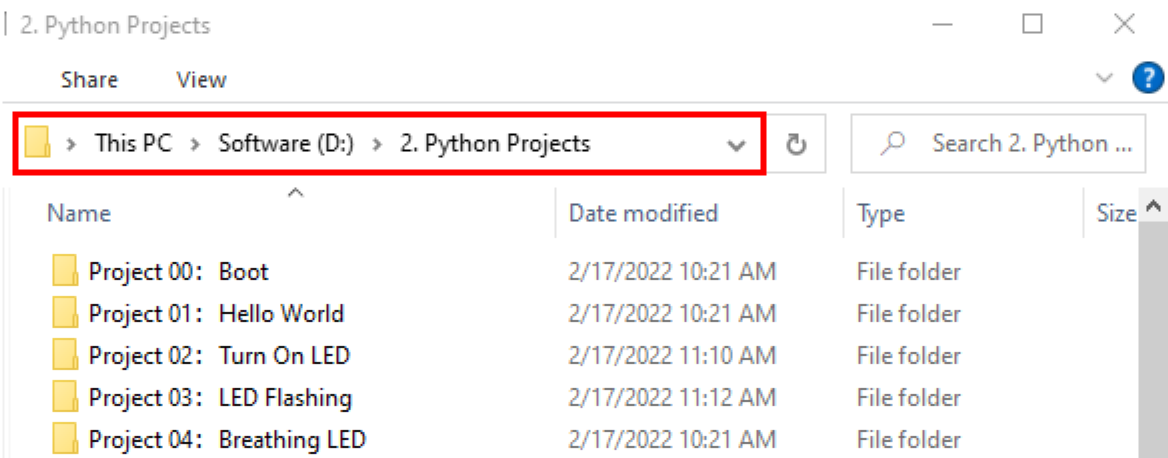


7.42.4 Component knowledge

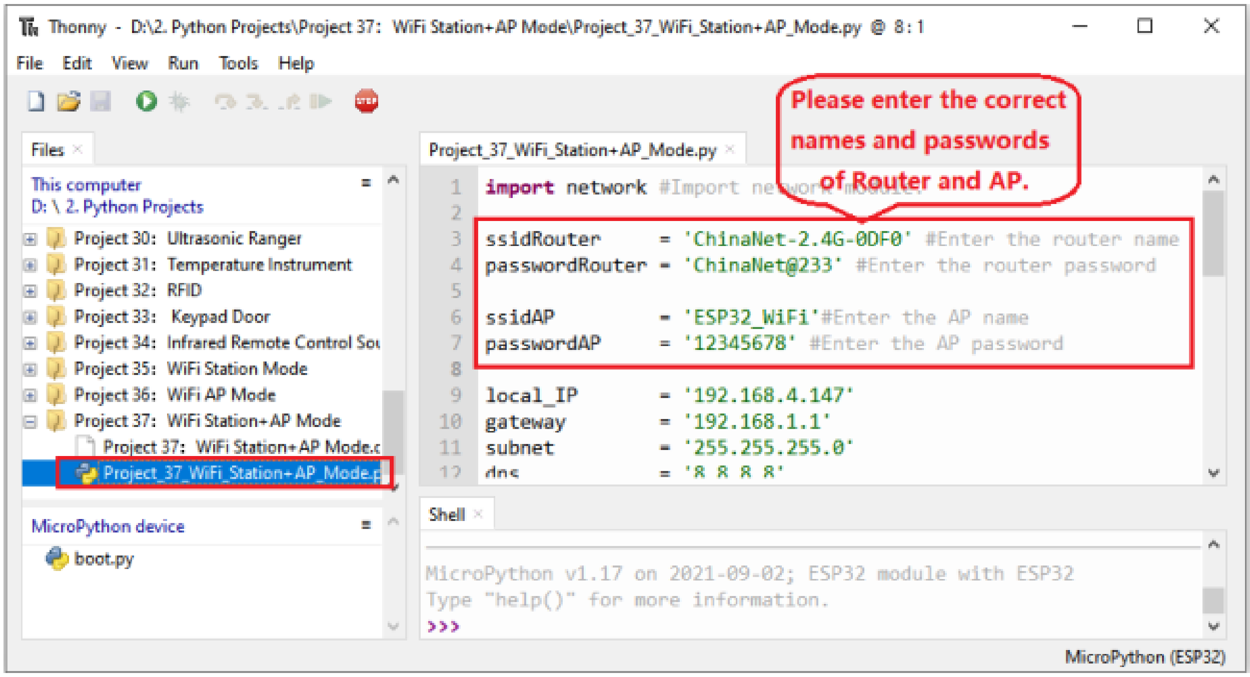
AP+Station mode: In addition to AP mode and Station mode, ESP32 can also use AP mode and Station mode at the same time. This mode contains the functions of the previous two modes. Turn on ESP32's Station mode, connect it to the router network, and it can communicate with the Internet via the router. 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



Open"Thonny"click"This computer"→"D:"→"2. Python Projects"→"Project 37WiFi Station+AP Mode"and double left-click "Project_37_WiFi_Station+AP_Mode.py".




```
import network #Import network module.

ssidRouter     = 'ChinaNet-2.4G-0DF0' #Enter the router name
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'
gateway        = '192.168.1.1'
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])
    print("Setting soft-AP  ... ")
    ap_if.config(essid=ssidAP,authmode=network.AUTH_WPA_WPA2_PSK, password=passwordAP)
```

(continues on next page)

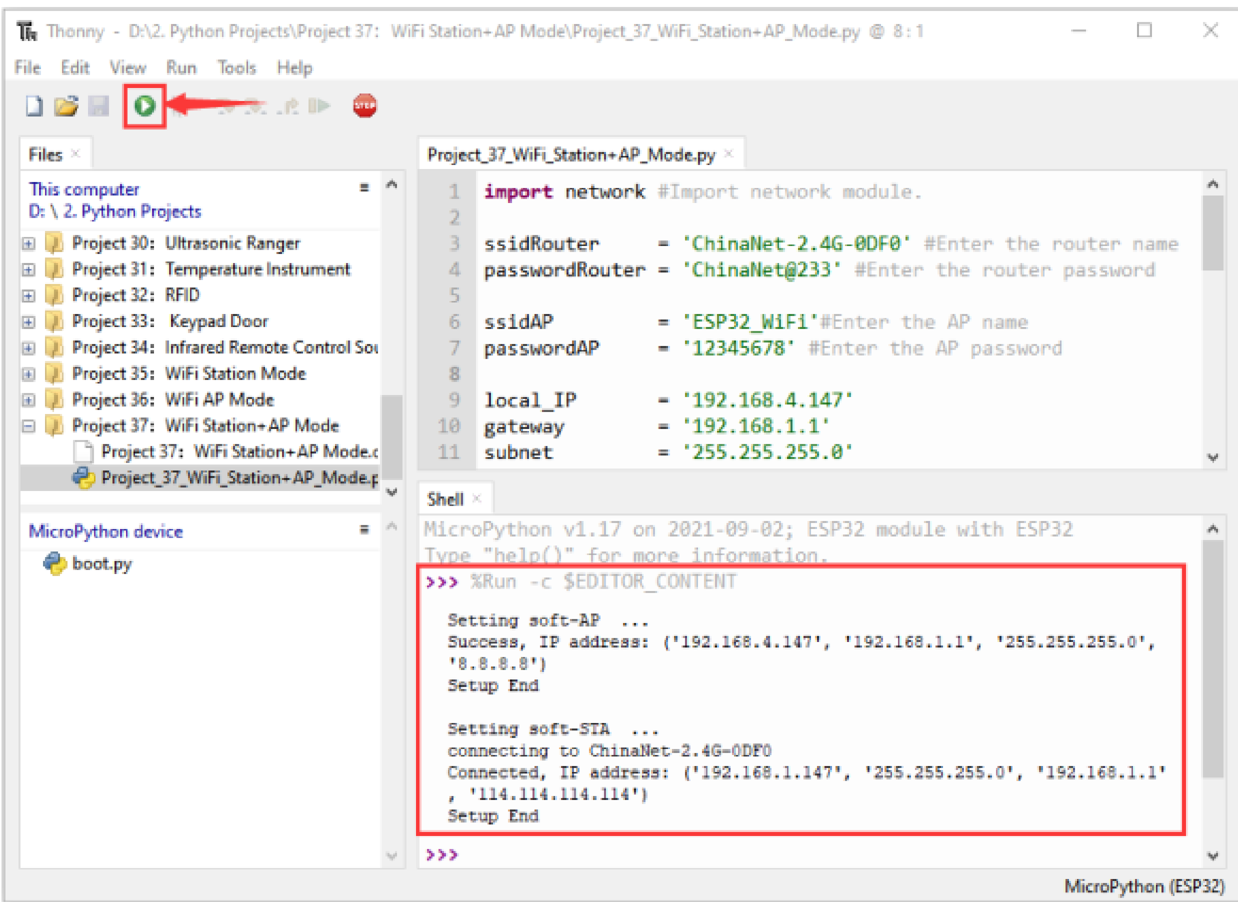
(continued from previous page)

```
    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

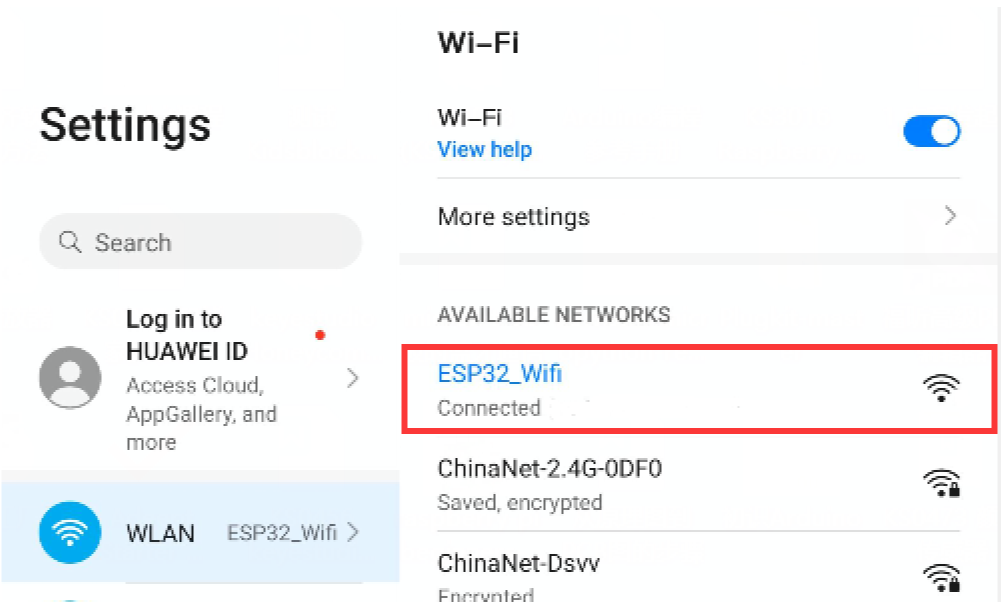
It is analogous to Project 35 and project 36. Before running the code, you need to modify ssidRouter, passwordRouter, ssidAP and passwordAP shown in the box of the illustration above.

After making sure that the code is modified correctly, click  "Run current script" the code starts to be executed and the "Shell" window of Thonny IDE will display as follows:



```
Shell ×
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

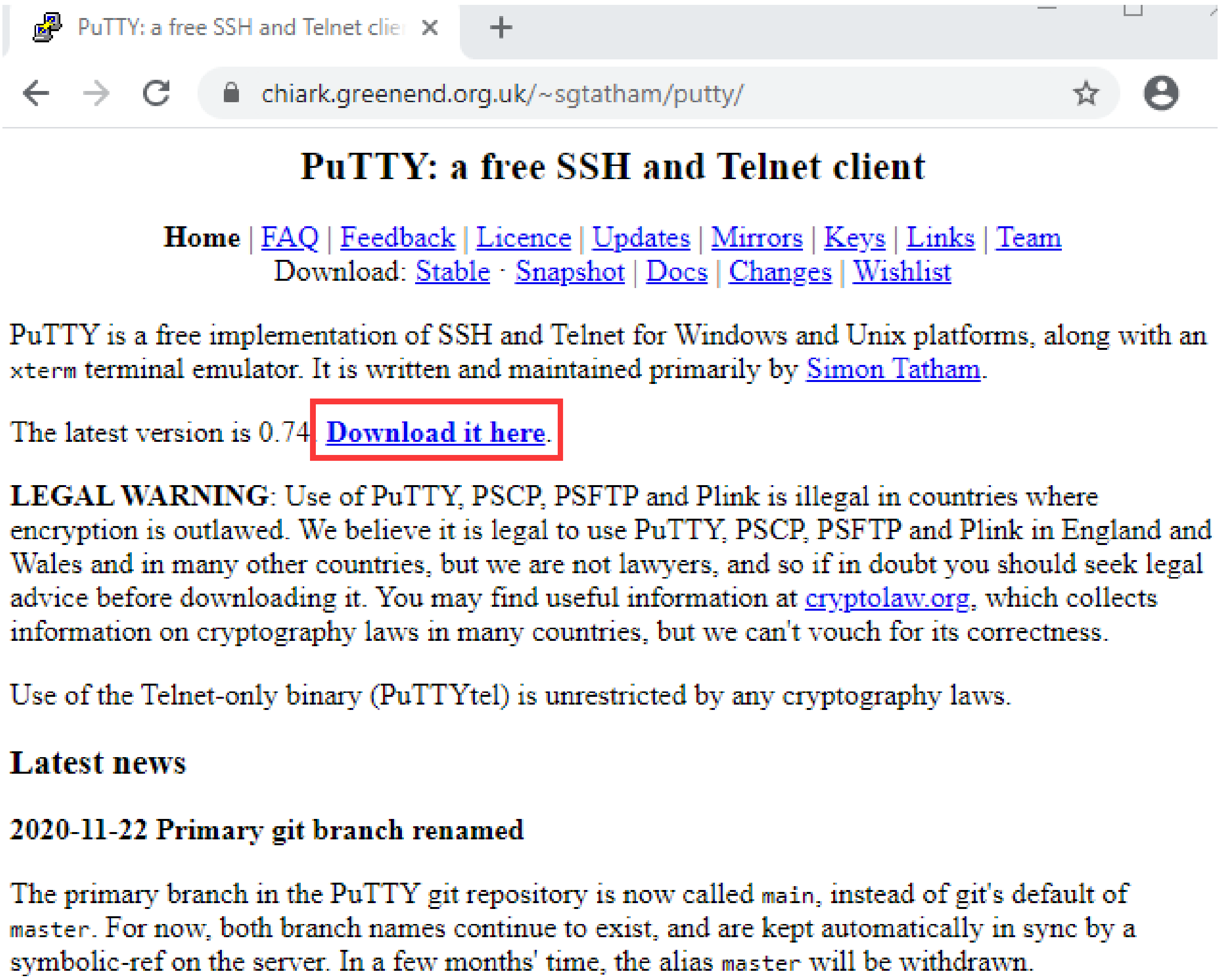
1Raspberry Pi 4B/3B/2B 2Above 16G TFT Memory Card 3Card Reader 4Computer and other parts

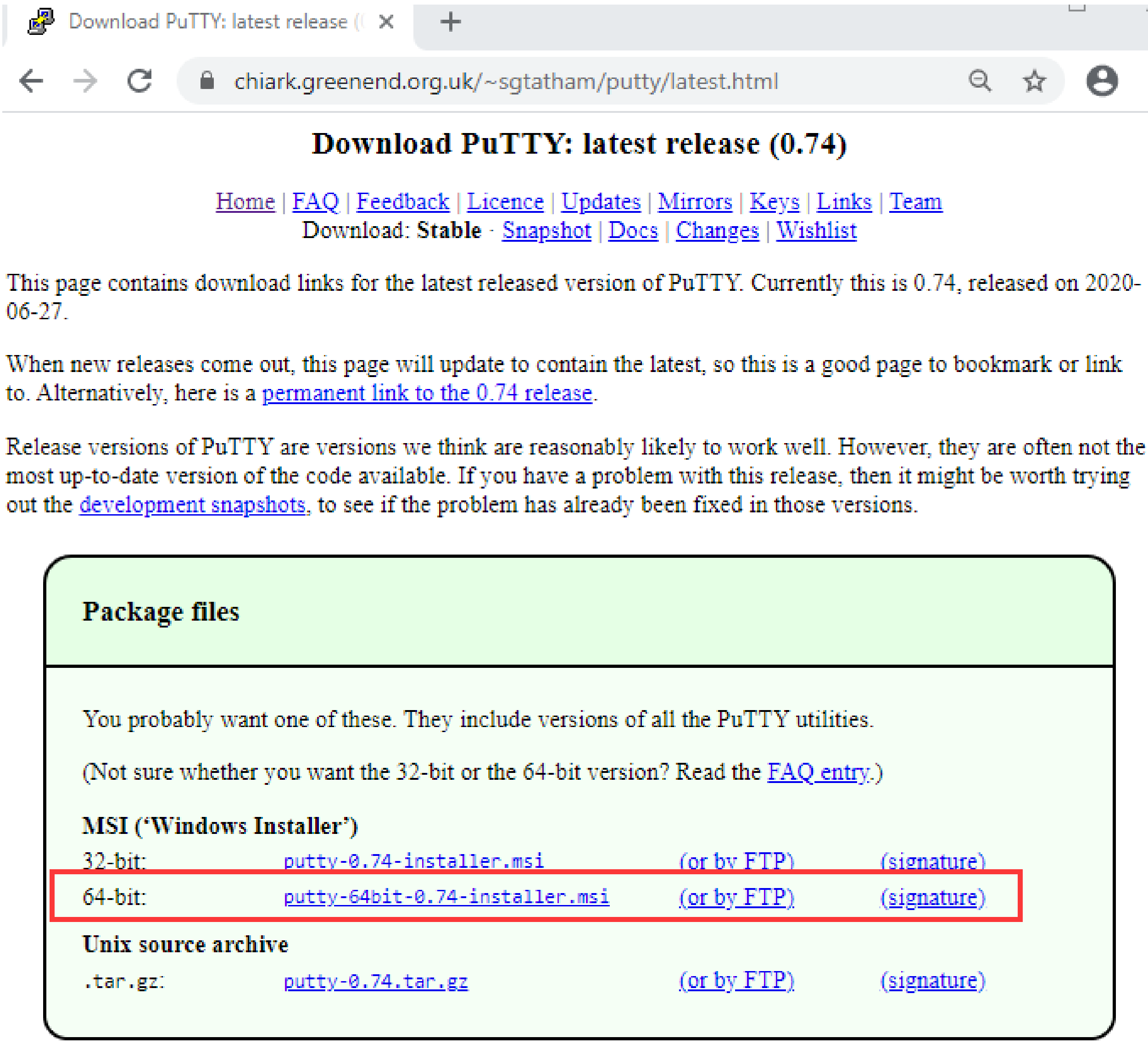
1.2. Software tools that need to be installed

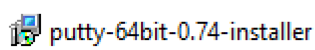
Windows System

1Install putty

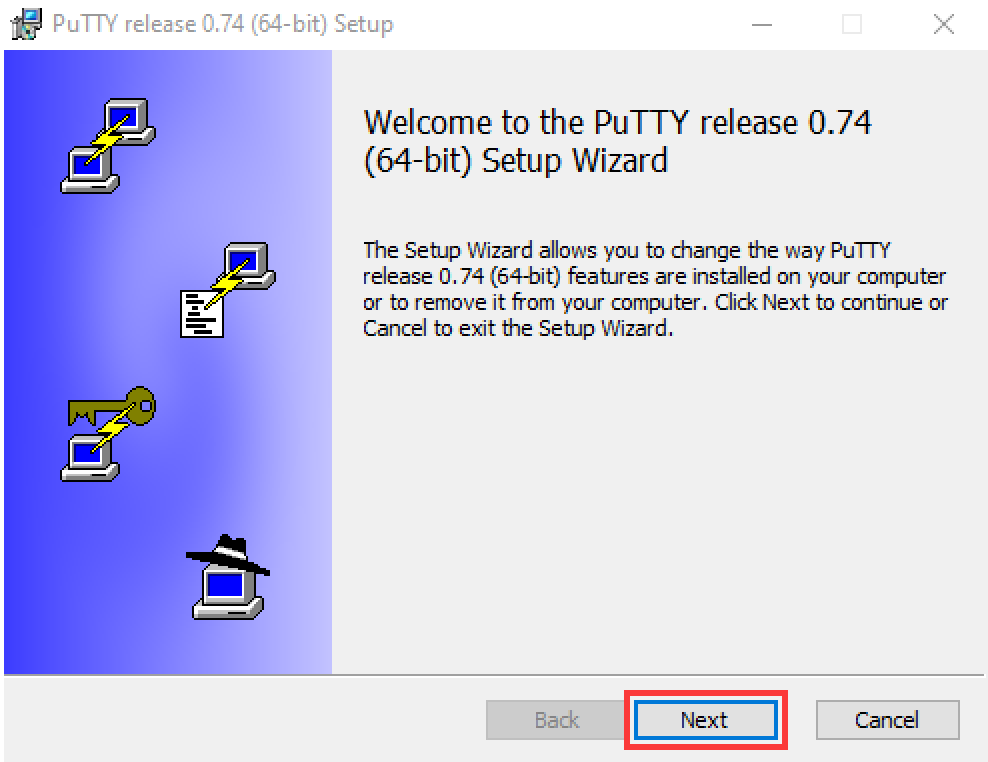
Download linkhttps://www.chiark.greenend.org.uk/~sgtatham/putty/


PuTTY: a free SSH and Telnet clie
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 Simon Tatham.
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 cryptolaw.org, 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.

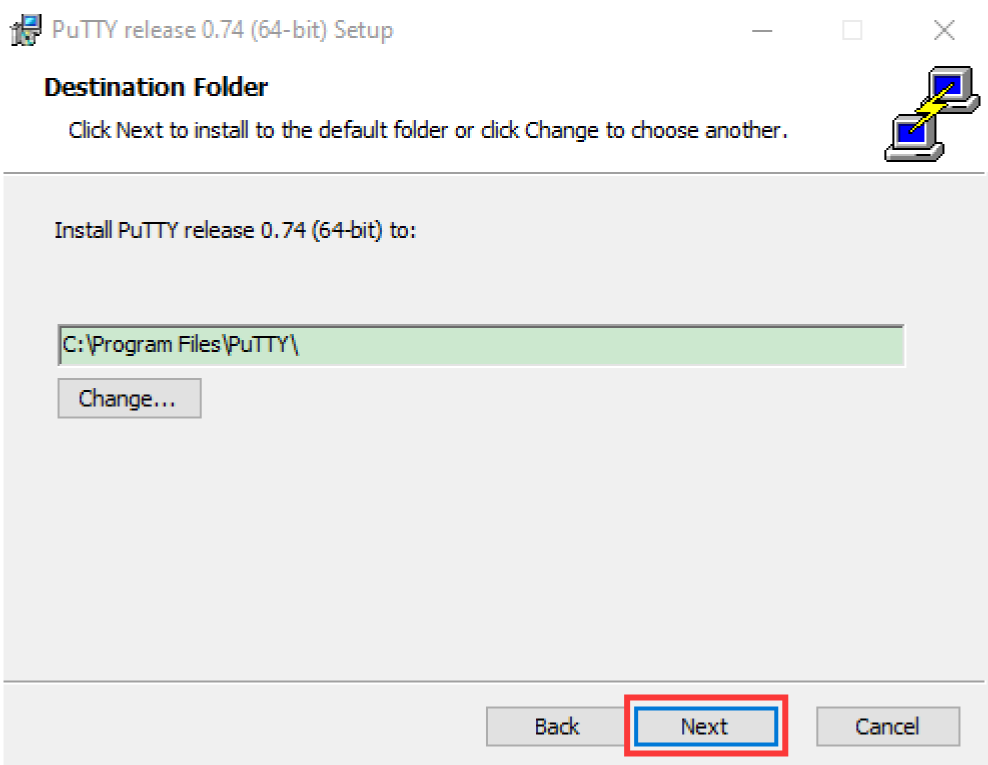


1. After downloading the package file 

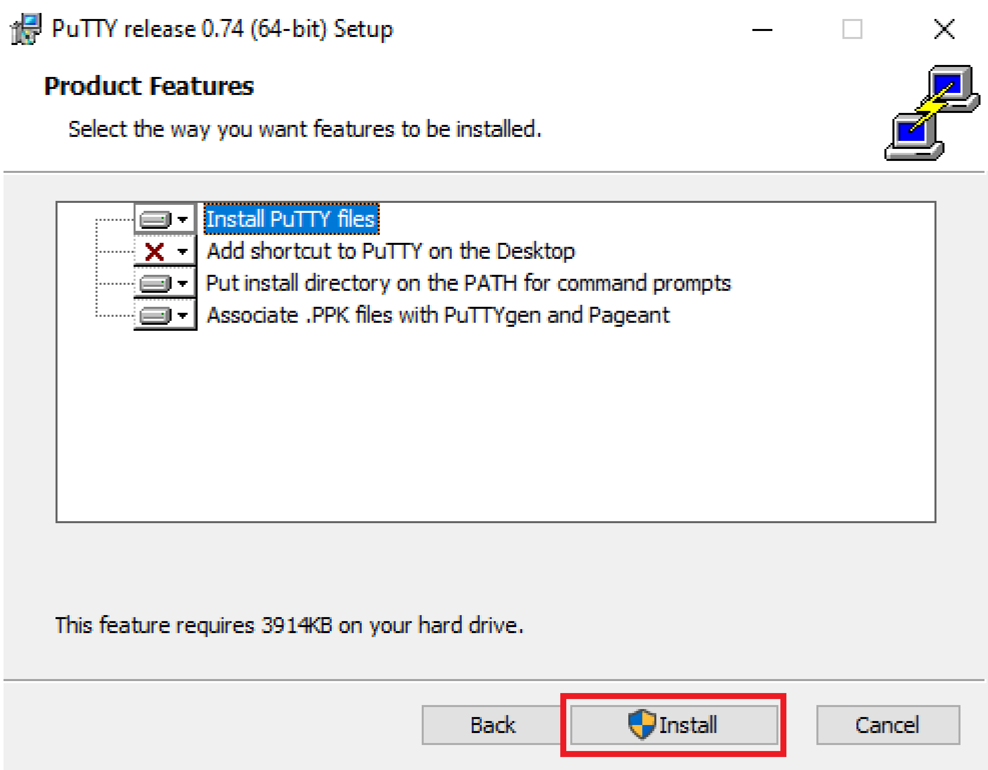
, double-click it and tap "Next".



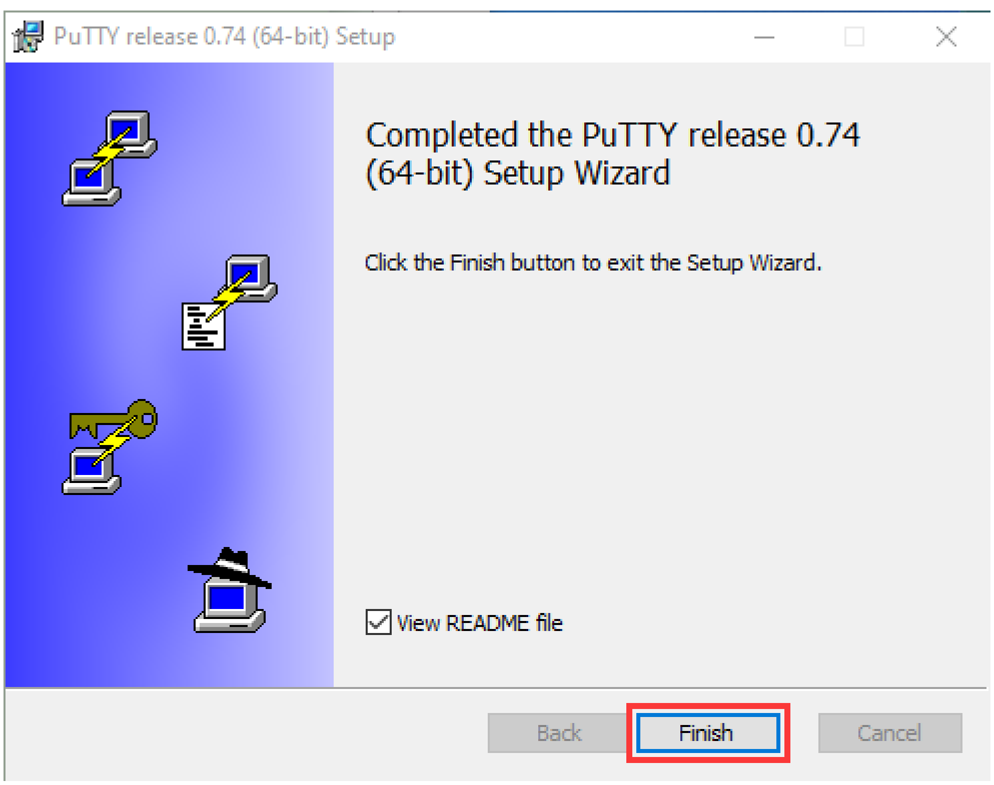
2. Click "Next".



3. Select"Install Putty files" and click"Install".



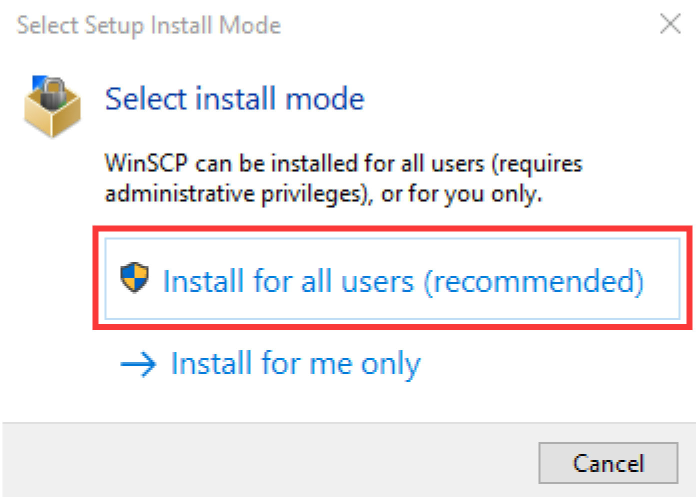
4. After a few seconds, the installation is complete, click "Finish".



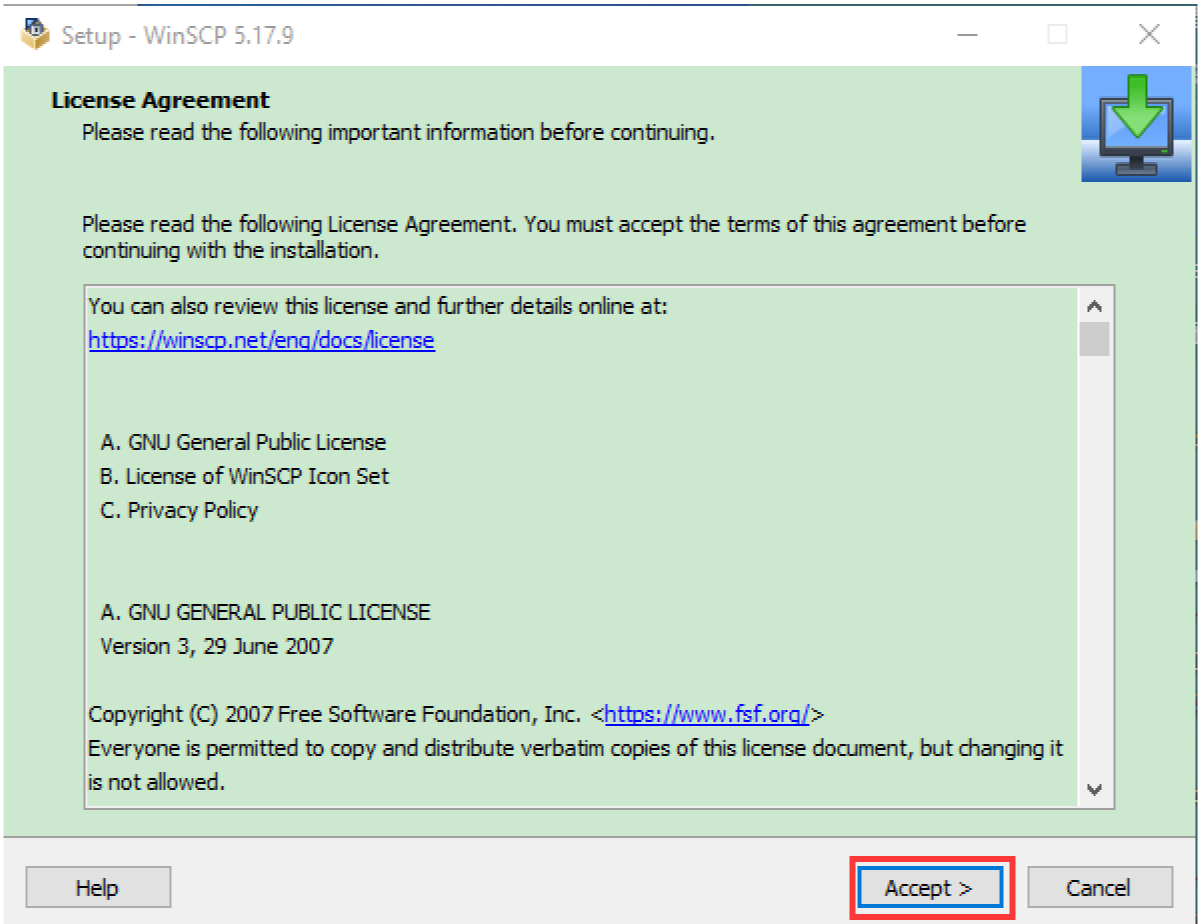
2Remote Login software -WinSCP

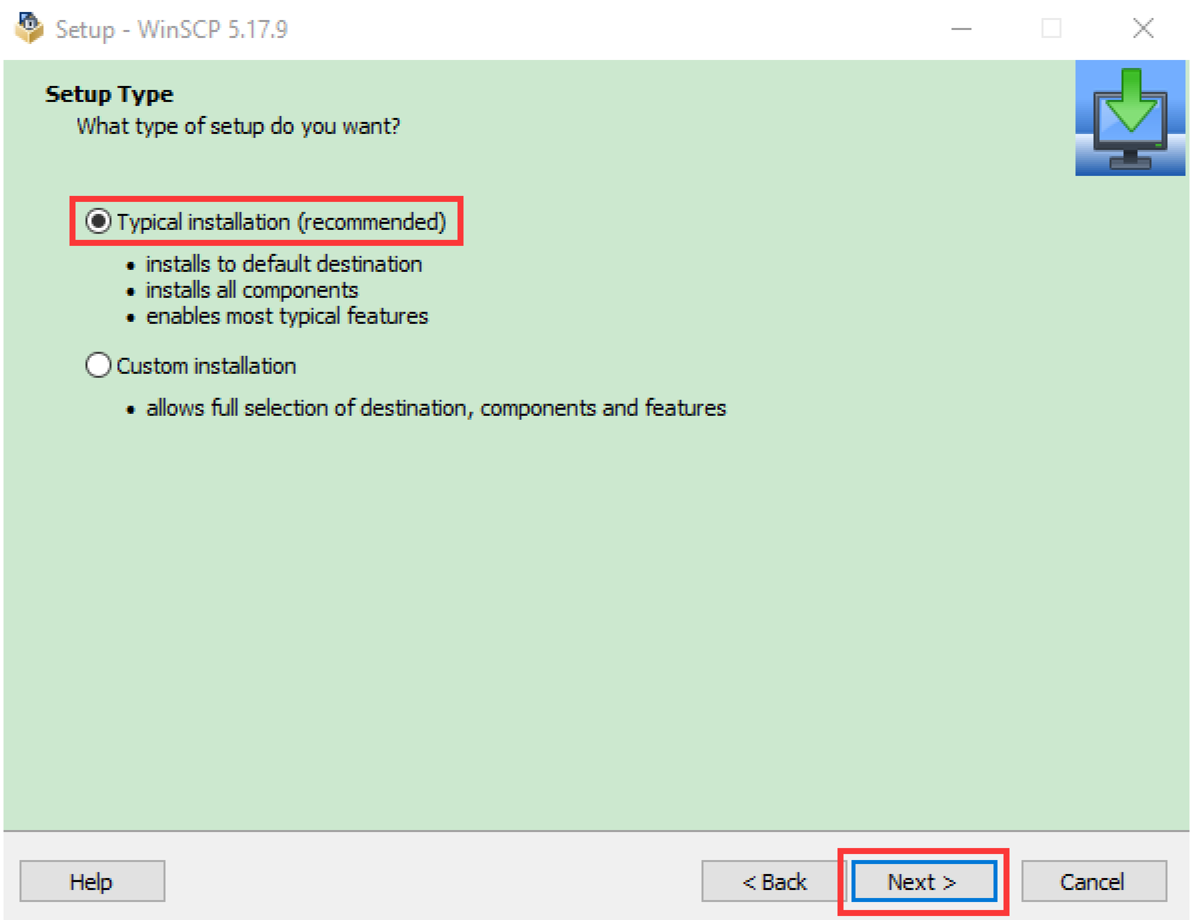
Download linkhttps://winscp.net/eng/download.php

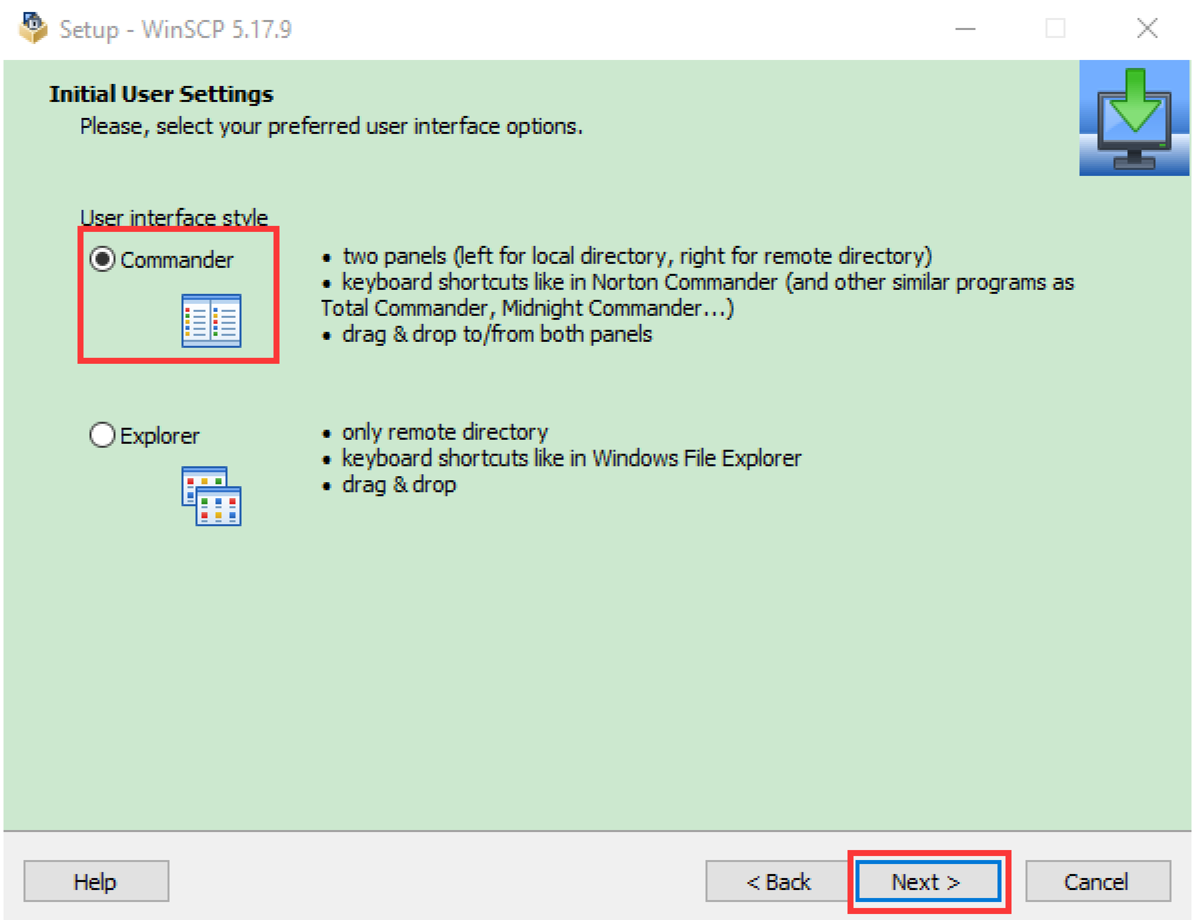
After downloading the WinSCP software file, double-click the WinSCP software file WinSCP-5.17.9-Setup.exe and click WinSCP-5.17.9-Setup.exe .

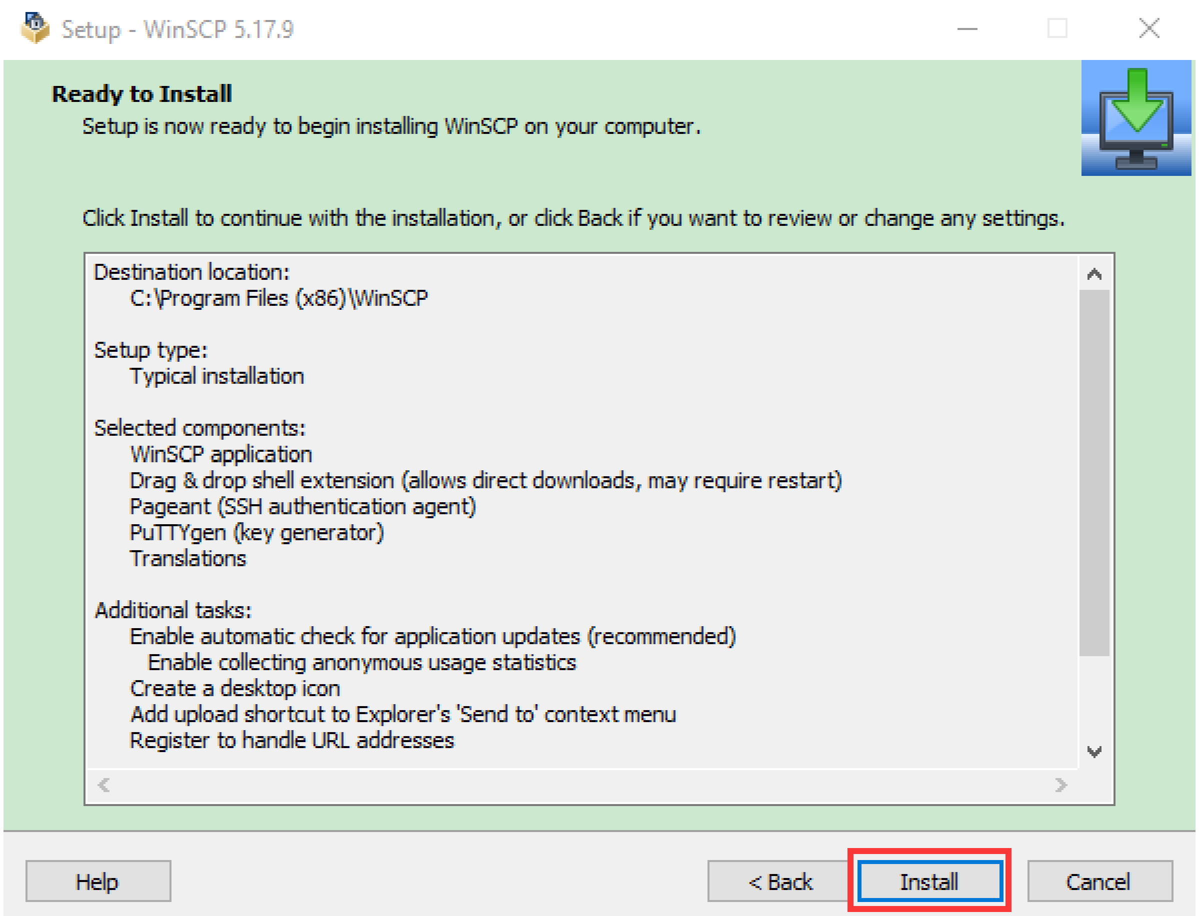


Click“Accept”then select the appropriate option and click“Next”, then Click“Install”.


Setup - WinSCP 5.17.9
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.
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. <https://www.fsf.org/>
Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.
Help
Accept >
Cancel


Setup - WinSCP 5.17.9
Setup Type
What type of setup do you want?
Typical installation (recommended)
• installs to default destination
• installs all components
• enables most typical features
Custom installation
• allows full selection of destination, components and features
Help
< Back
Next >
Cancel


Setup - WinSCP 5.17.9
Initial User Settings
Please, select your preferred user interface options.
User interface style
Commander
• two panels (left for local directory, right for remote directory)
• keyboard shortcuts like in Norton Commander (and other similar programs as Total Commander, Midnight Commander…)
• drag & drop to/from both panels
Explorer
• only remote directory
• keyboard shortcuts like in Windows File Explorer
• drag & drop
Help
< Back
Next >
Cancel

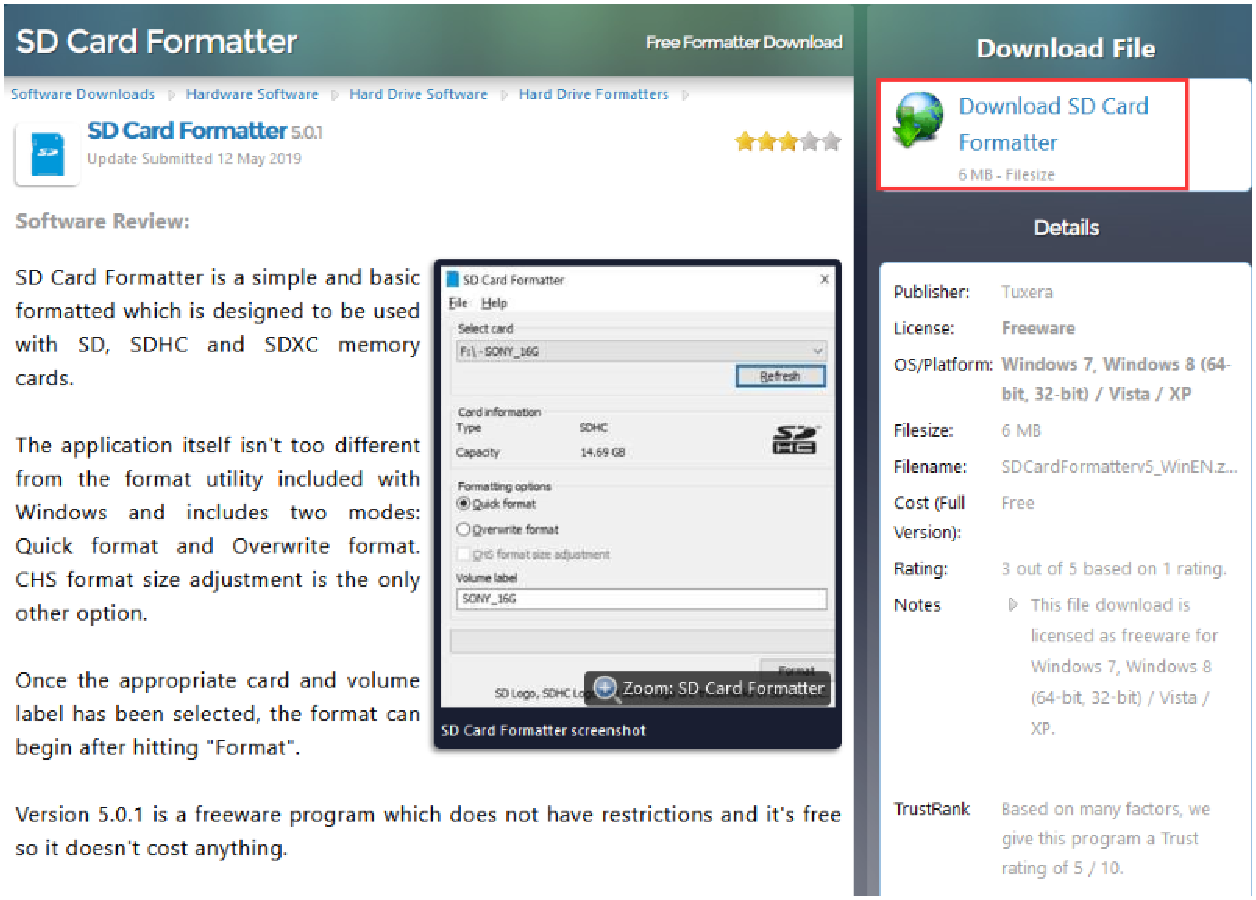


After a few seconds, the installation is complete, click "Finish".

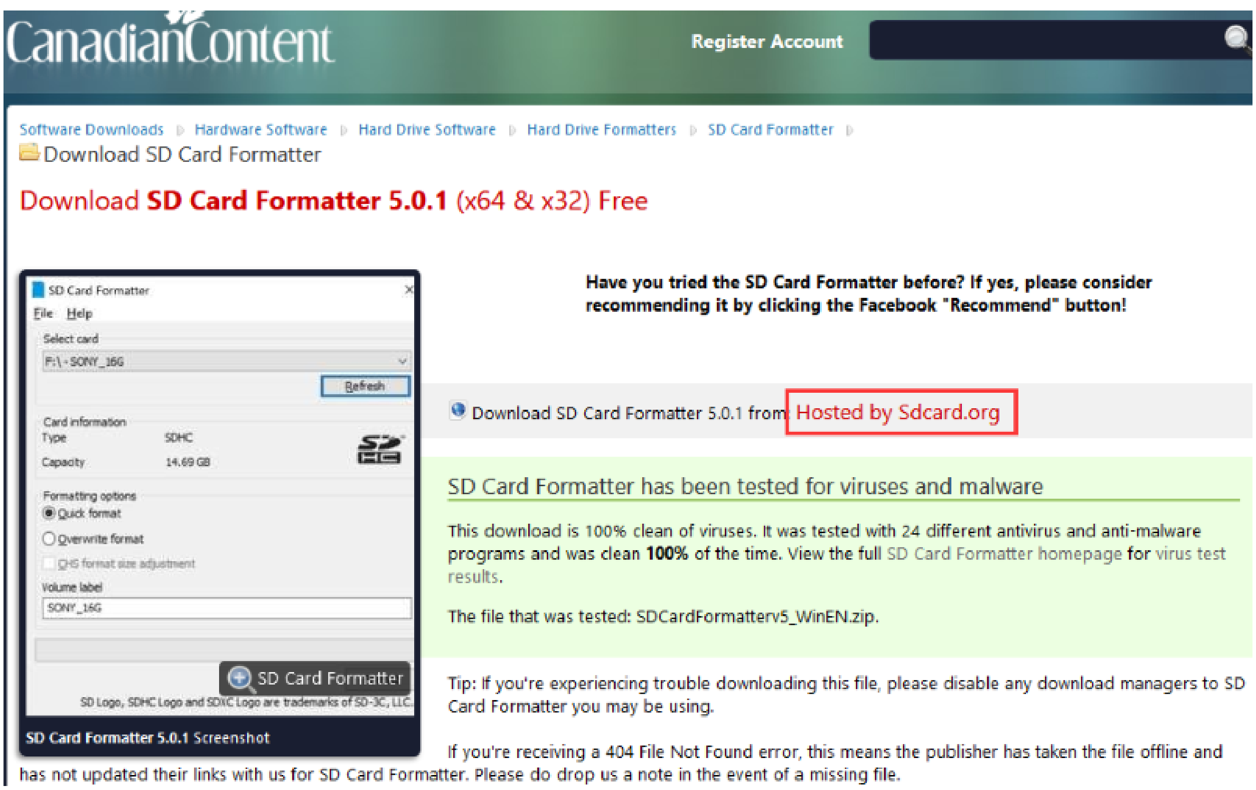
3Format TFT card tool– SD Card Formatter

Download link

http://www.canadiancontent.net/tech/download/SD_Card_Formatter.html

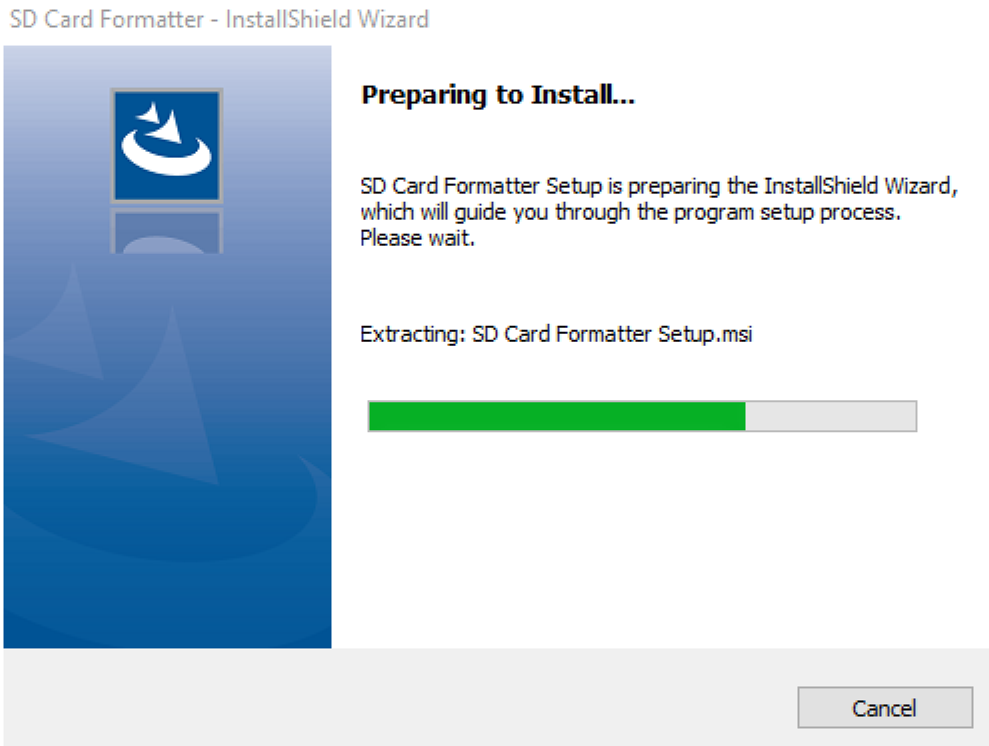


SD Card Formatter
Free Formatter Download
Software Downloads Hardware Software Hard Drive Software Hard Drive Formatters
SD Card Formatter 5.0.1
Update Submitted 12 May 2019
Software Review:
SD Card Formatter is a simple and basic formatted which is designed to be used with SD, SDHC and SDXC memory cards.
The application itself isn't too different from the format utility included with Windows and includes two modes: Quick format and Overwrite format. CHS format size adjustment is the only other option.
Once the appropriate card and volume label has been selected, the format can begin after hitting "Format".
Version 5.0.1 is a freeware program which does not have restrictions and it's free so it doesn't cost anything.
Zoom: SD Card Formatter
SD Card Formatter screenshot
Download File
Download SD Card Formatter
6 MB - Filesize
Details
Publisher: Tuxera
License: Freeware
OS/Platform: Windows 7, Windows 8 (64-bit, 32-bit) / Vista / XP
Filesize: 6 MB
Filename: SDCardFormatterv5_WinEN.z...
Cost (Full Version): Free
Rating: 3 out of 5 based on 1 rating.
Notes This file download is licensed as freeware for Windows 7, Windows 8 (64-bit, 32-bit) / Vista / XP.
TrustRank Based on many factors, we give this program a Trust rating of 5 / 10.

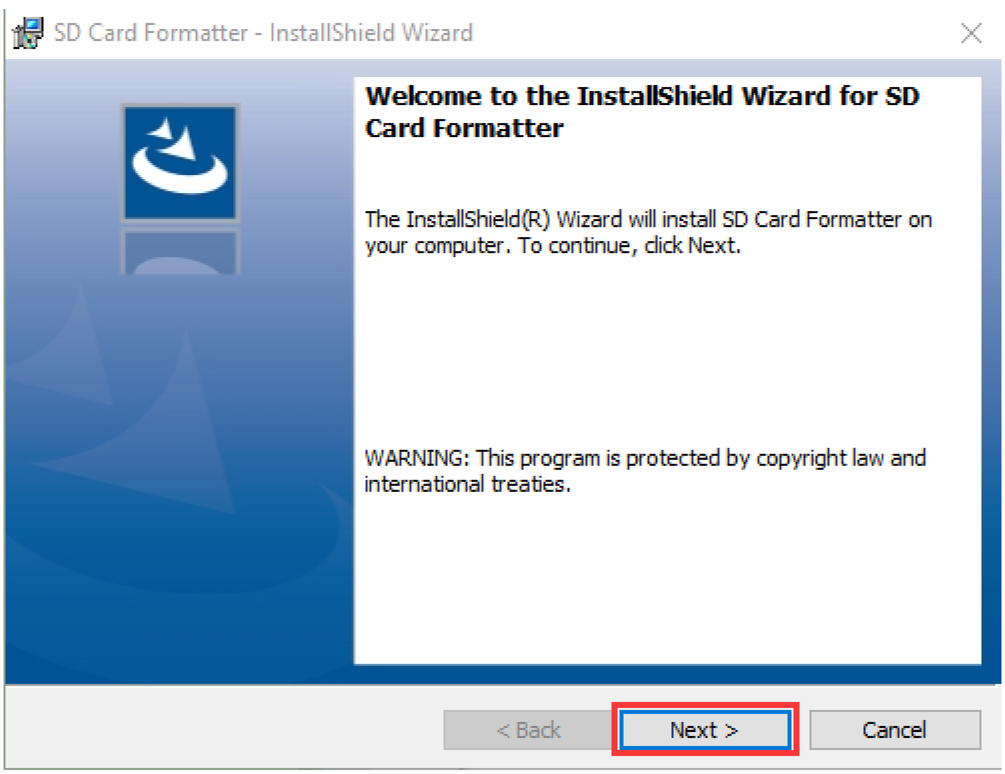


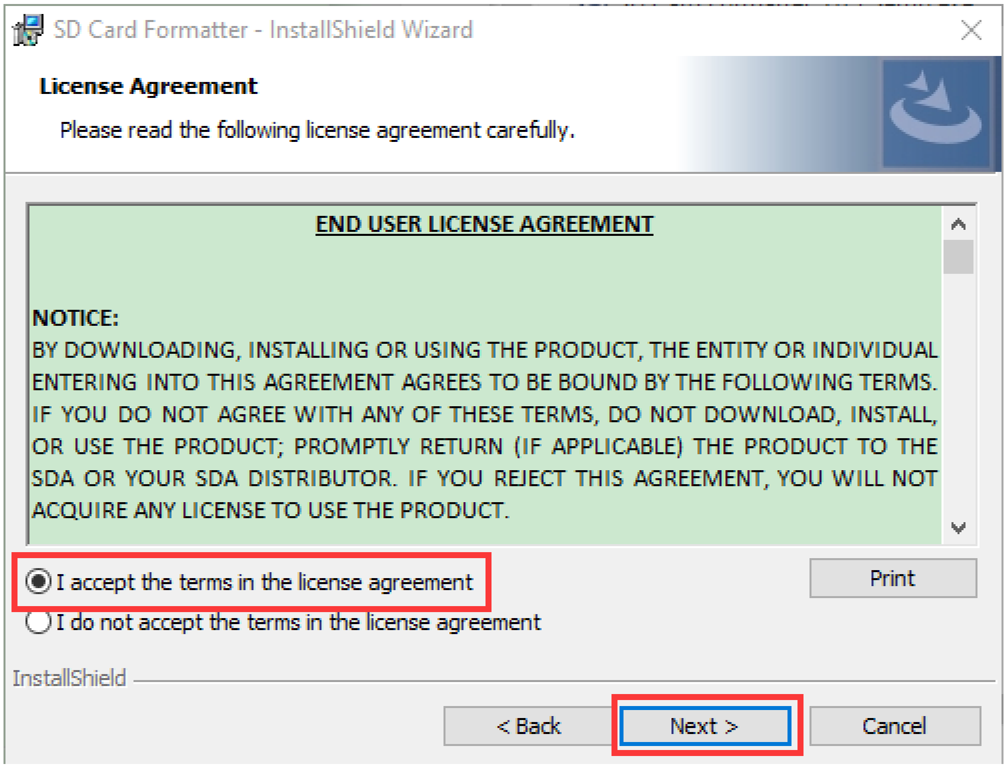
CanadianContent
Register Account
Software Downloads Hardware Software Hard Drive Software Hard Drive Formatters SD Card Formatter
Download SD Card Formatter
Download SD Card Formatter 5.0.1 (x64 & x32) Free
Have you tried the SD Card Formatter before? If yes, please consider recommending it by clicking the Facebook "Recommend" button!
Download SD Card Formatter 5.0.1 from Hosted by Sdcard.org
SD Card Formatter has been tested for viruses and malware
This download is 100% clean of viruses. It was tested with 24 different antivirus and anti-malware programs and was clean 100% of the time. View the full SD Card Formatter homepage for virus test results.
The file that was tested: SDCardFormatterv5_WinEN.zip.
Tip: If you're experiencing trouble downloading this file, please disable any download managers to SD Card Formatter you may be using.
If you're receiving a 404 File Not Found error, this means the publisher has taken the file offline and has not updated their links with us for SD Card Formatter. Please do drop us a note in the event of a missing file.
SD Card Formatter 5.0.1 Screenshot

Unzip the SDCardFormatterv5_WinEN package and double-click the SD Card Formatter file SD Card Formatter 5.0.1 Setup.exe to run it.

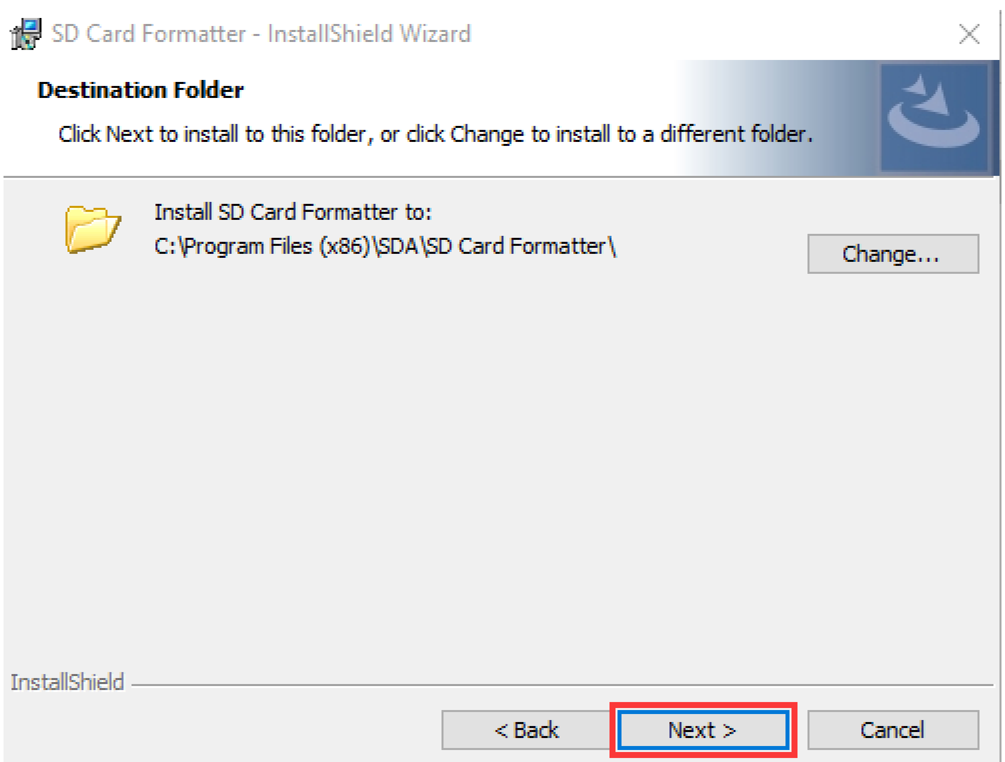


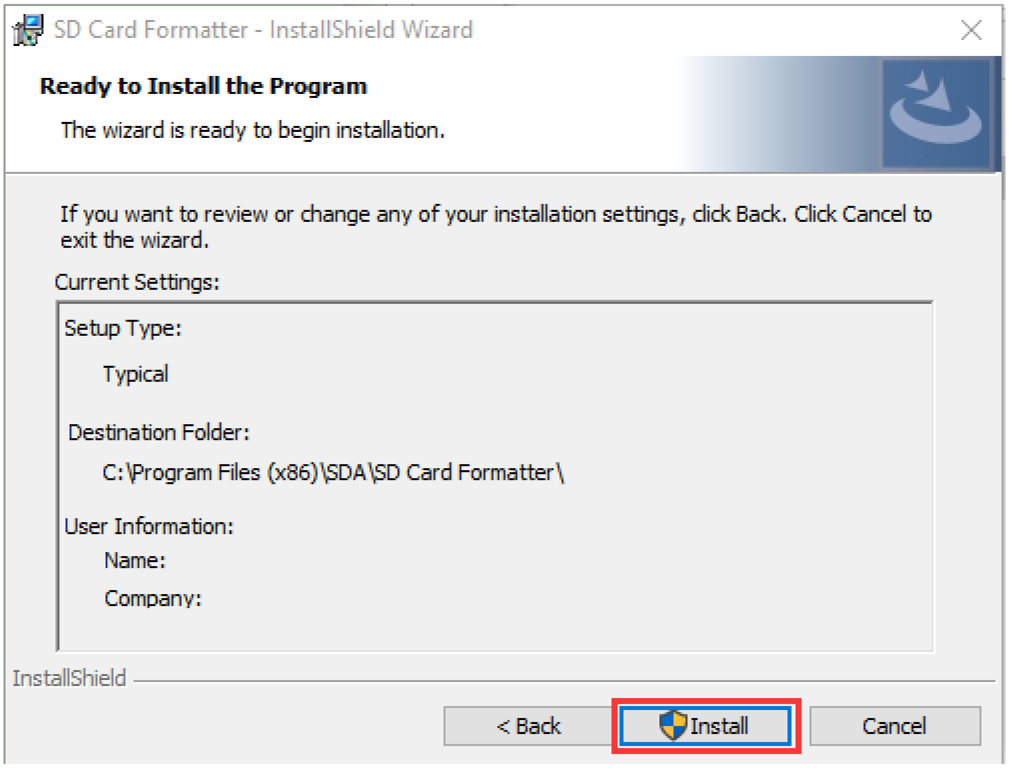
Click"Next"select" I accept the terms in the license agreement "and click"Next".



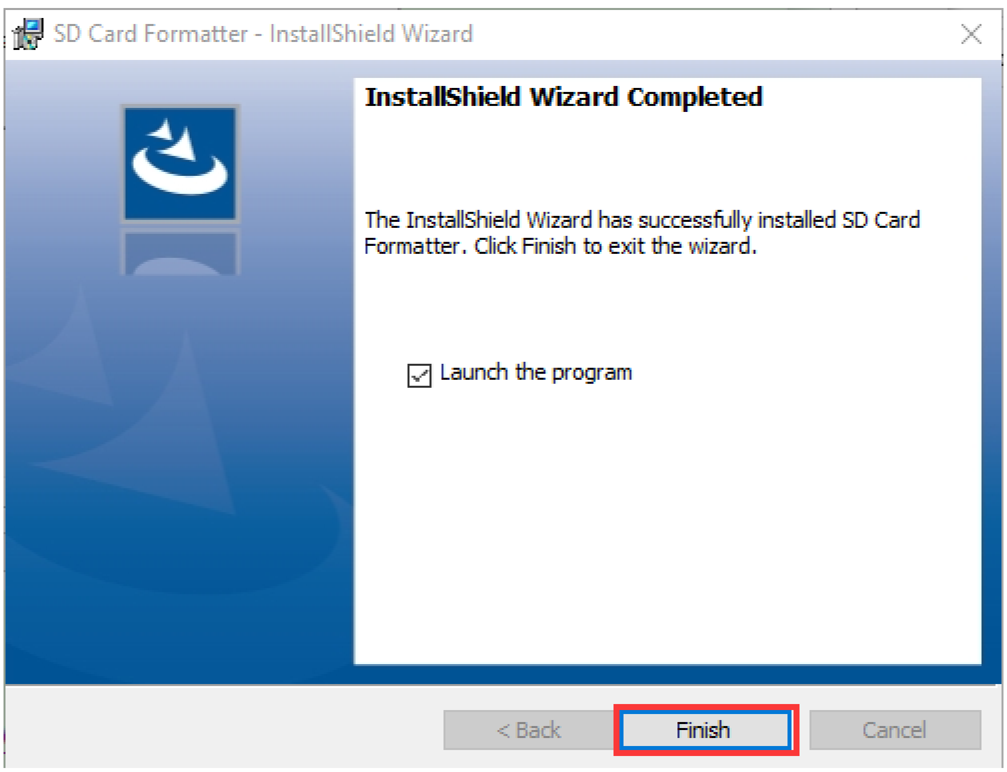


Click "Next"again, and then click "Install".



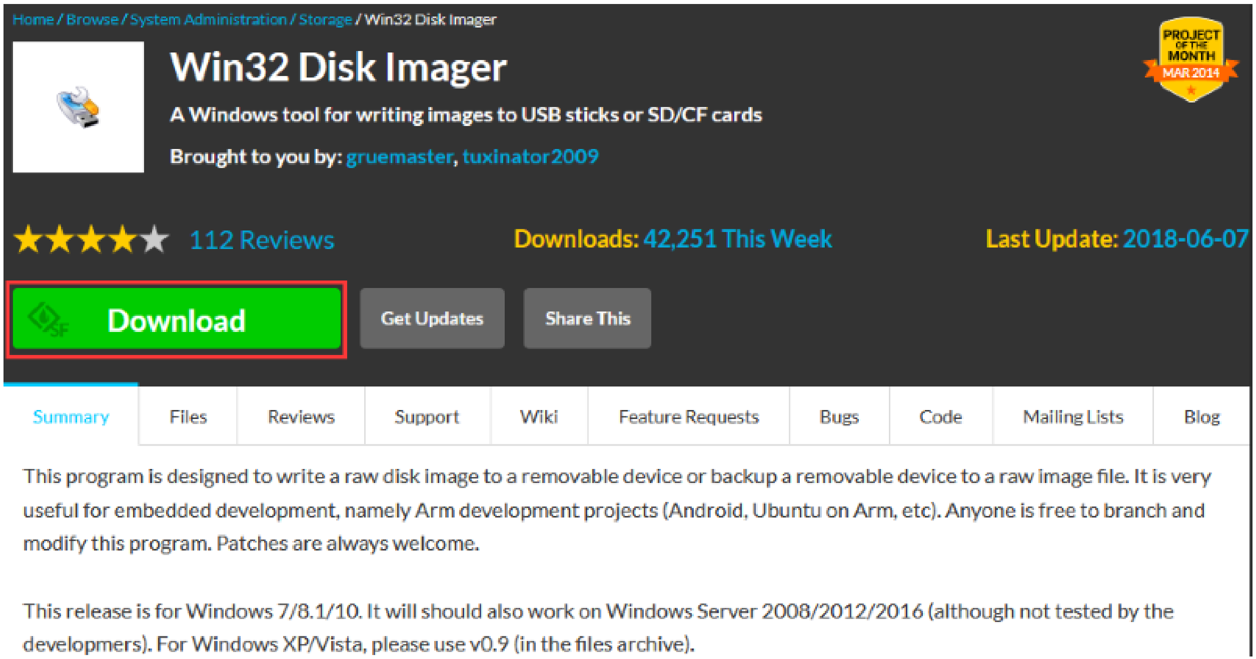


After a few seconds, the installation is complete, click "Finish".

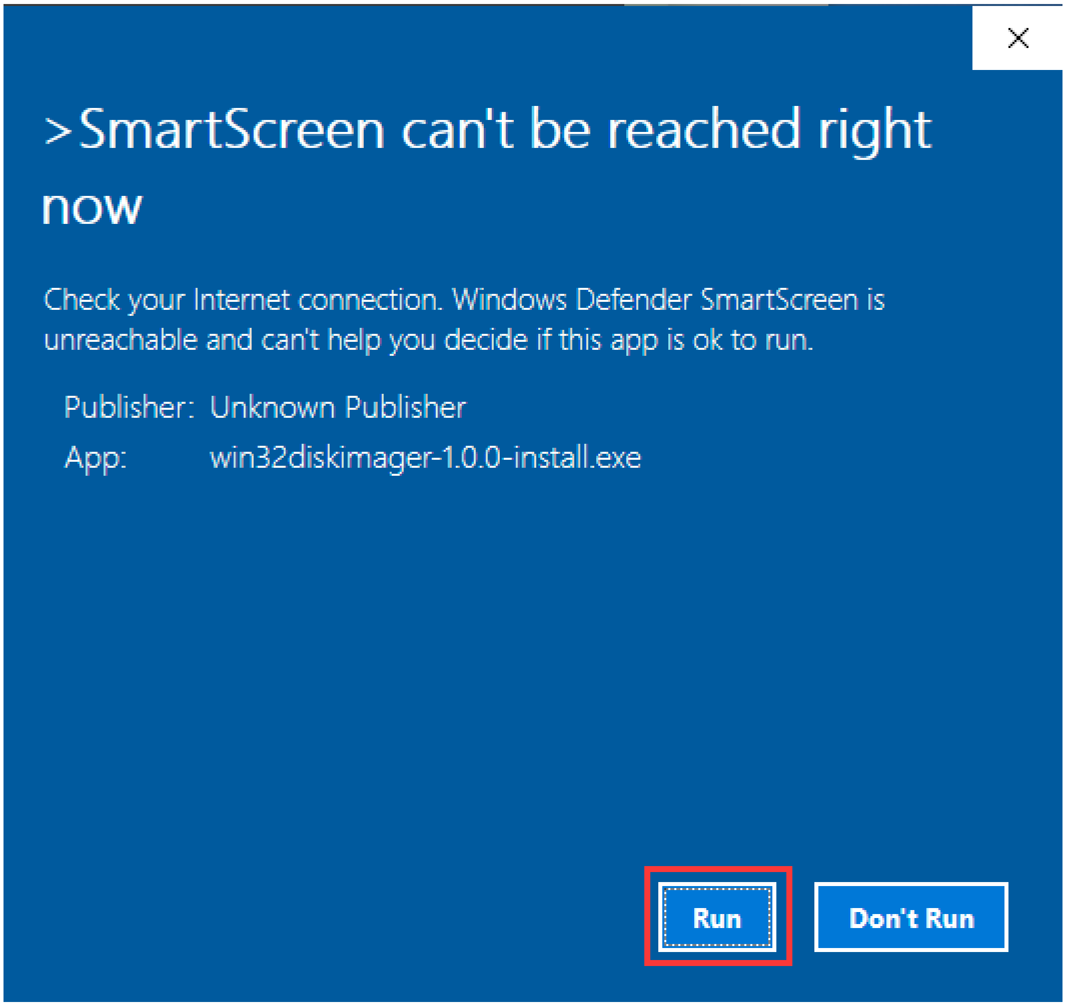


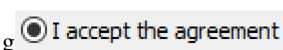
4Burn mirror system software tool— Win32DiskImager

Download linkhttps://sourceforge.net/projects/win32diskimager/

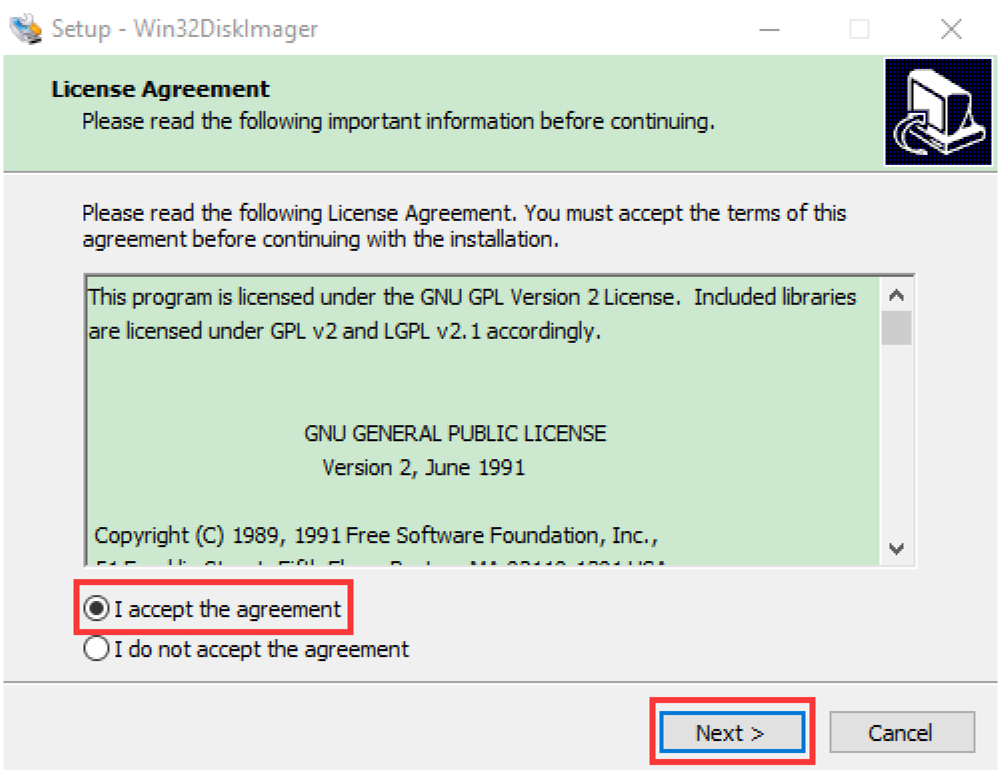


1. After downloading the software file win32diskimager-1.0.0-install.exe double-click Win32DiskImagersoftware file win32diskimager-1.0.0-install.exe and then click"Run".

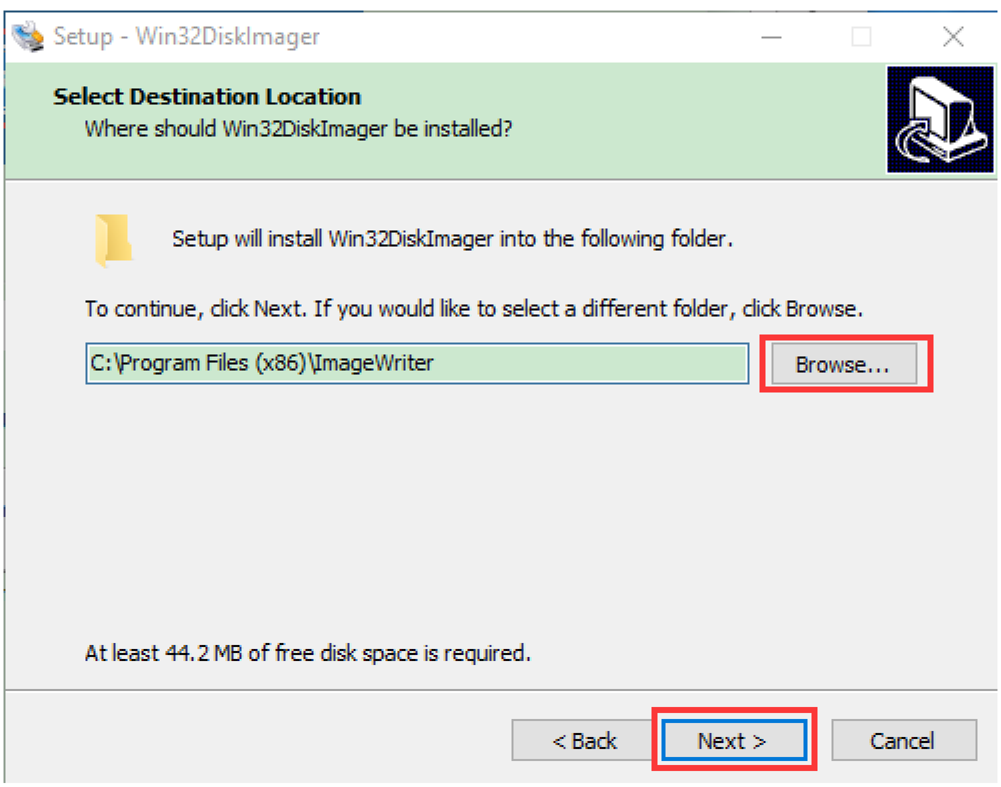


2. After selecting 

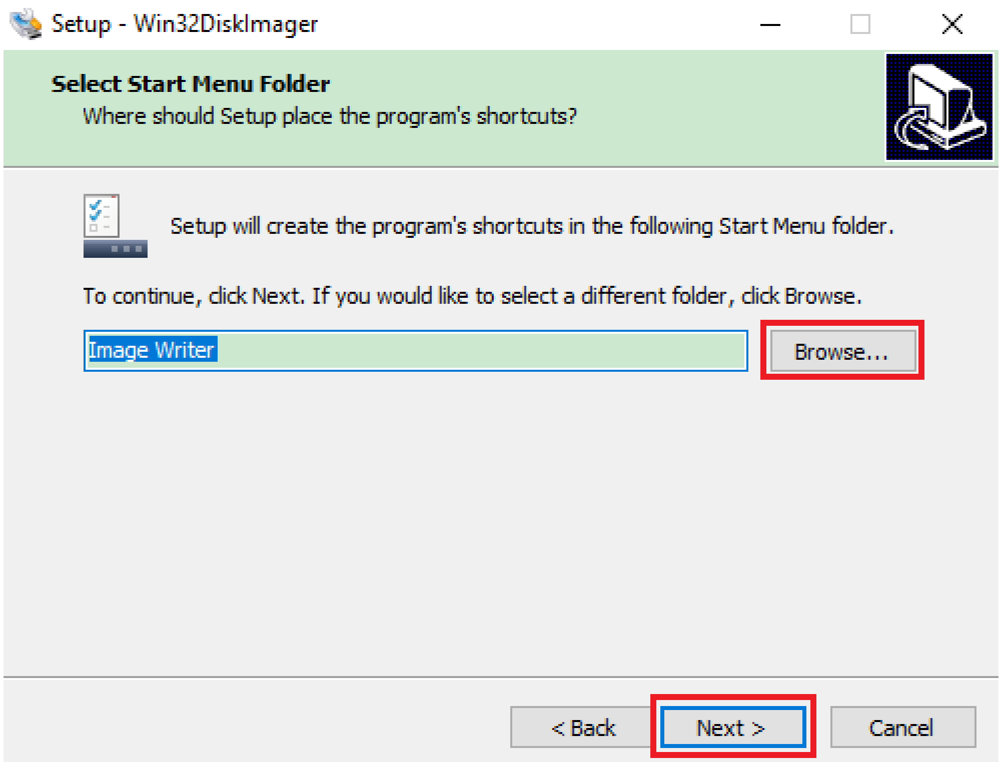
and click“Next”.



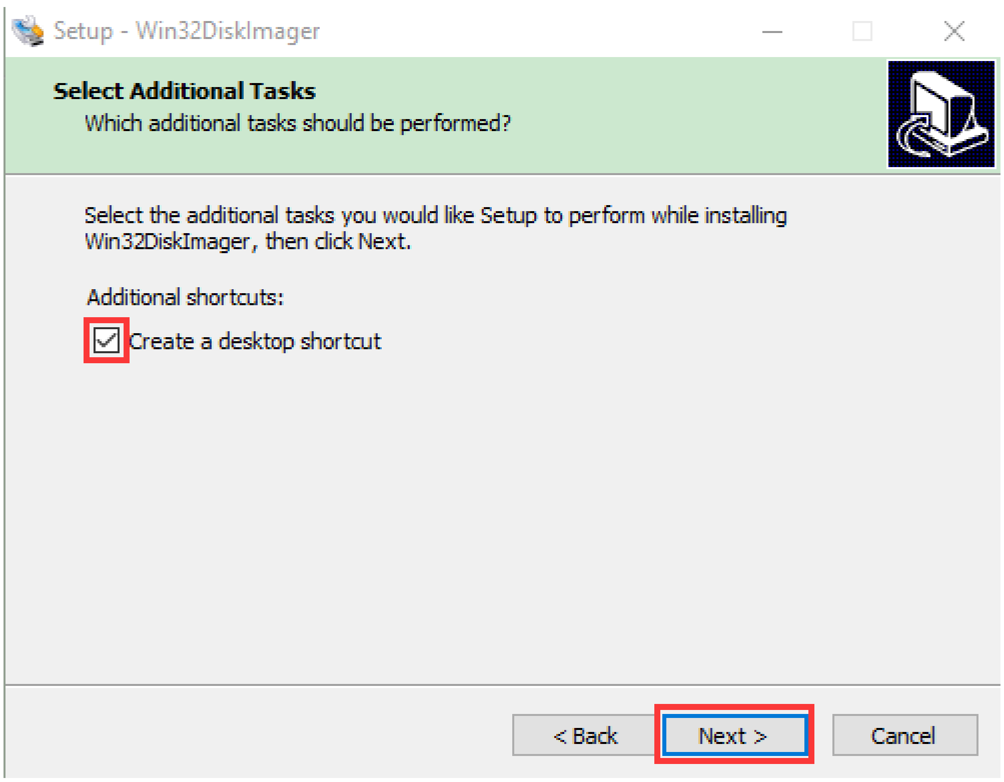
3. Click "Browse…"select the location where Win32DiskImager is installed and click"Next".

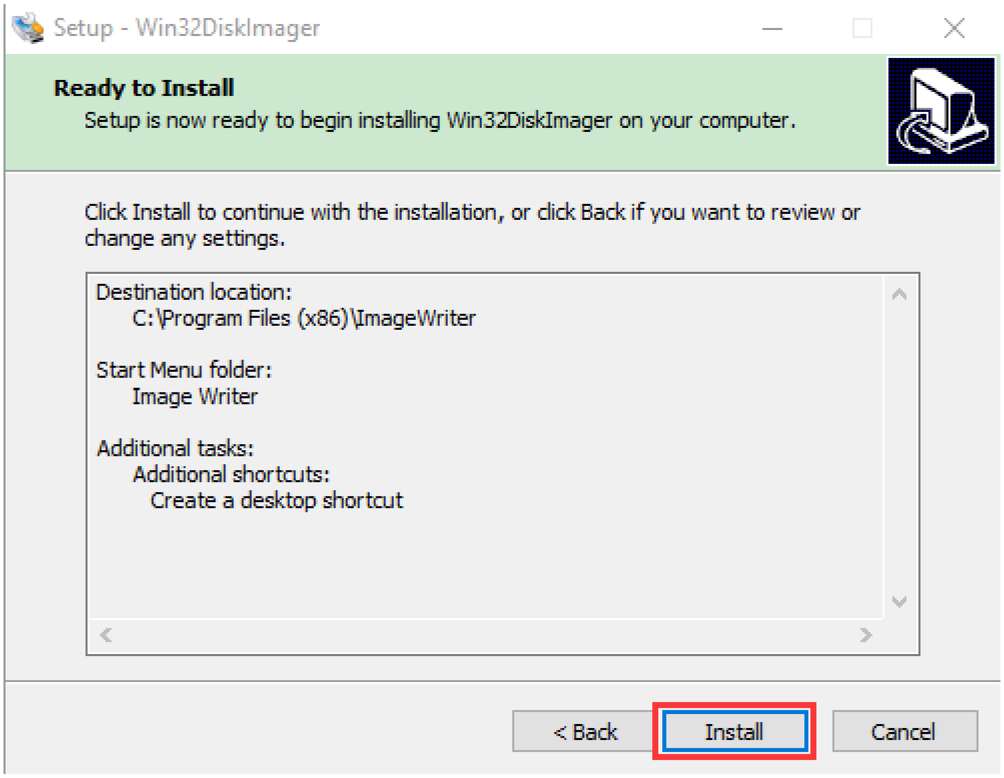


4. Click "Browse…"select the location where Win32DiskImager is installed and click"Next".

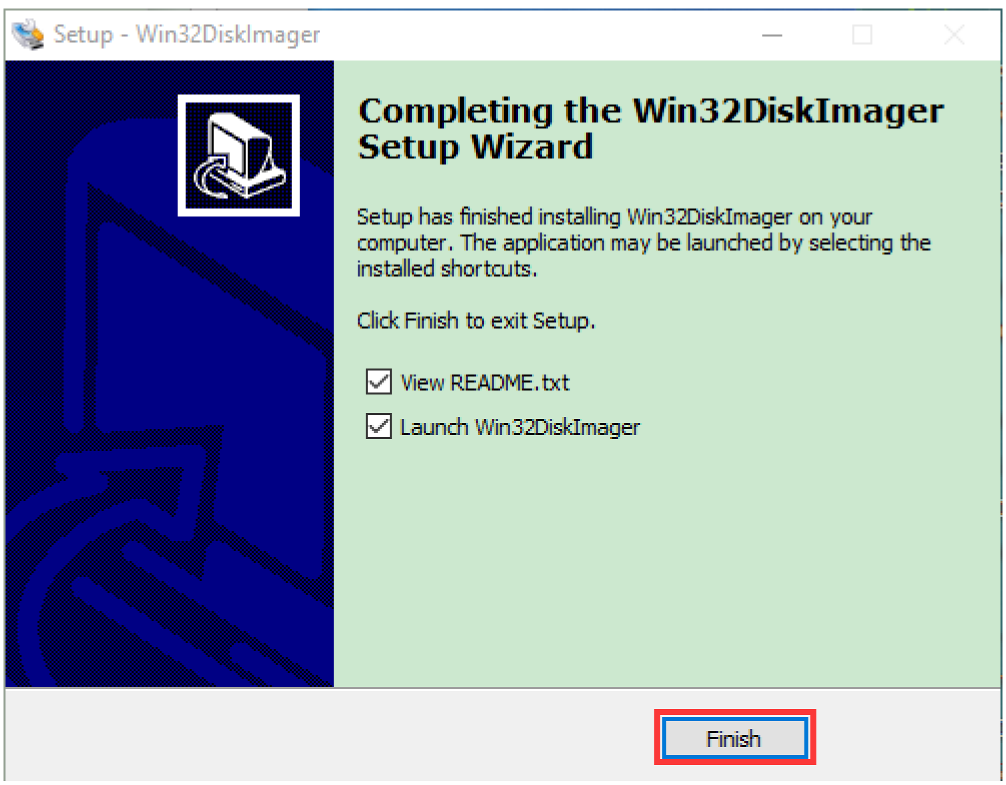


5. Select Create a desktop shortcut and click“Next”and then click again“Install”.





After a few seconds, the installation is complete, click "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

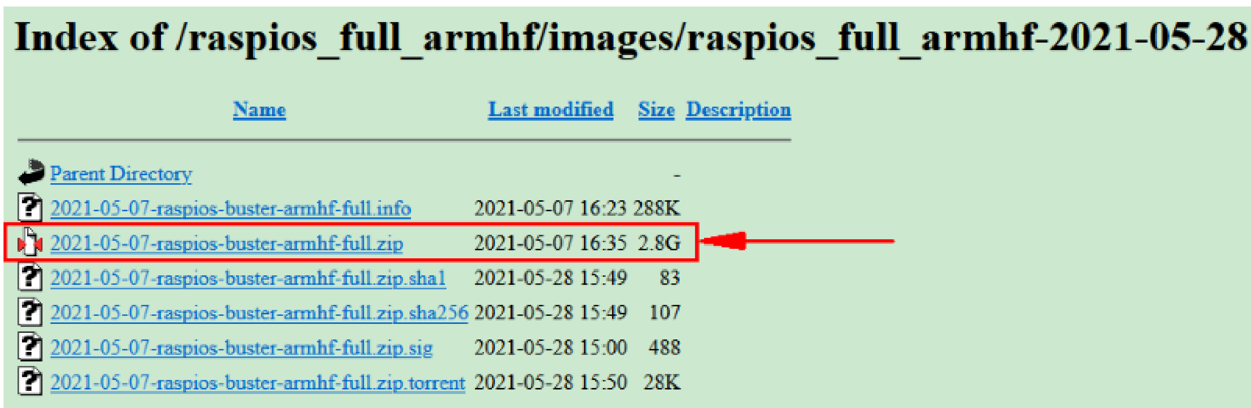
https://www.raspberrypi.org/downloads/raspberry-pi-os/

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/



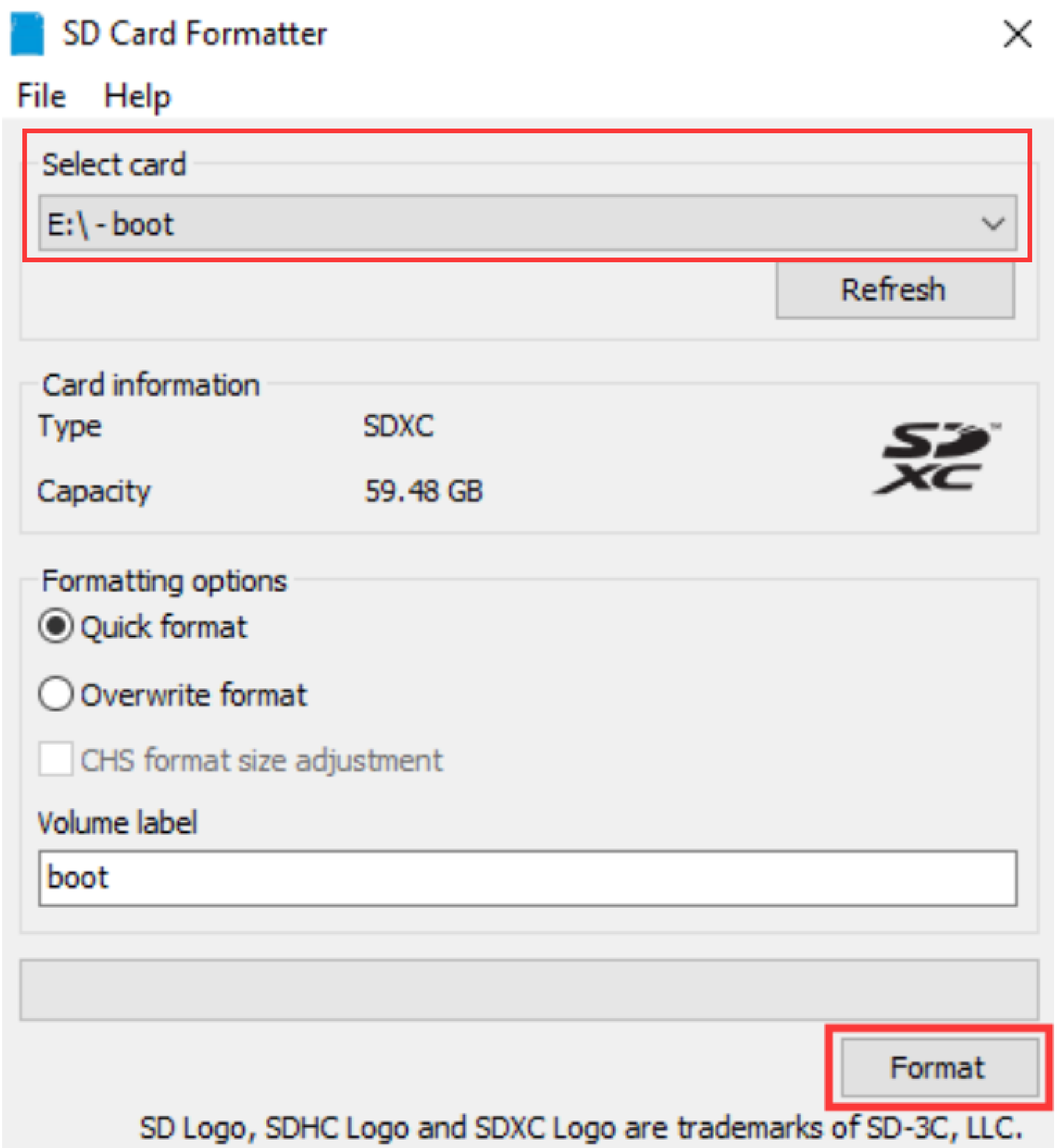
Index of /raspios_full_armhf/images/raspios_full_armhf-2021-05-28

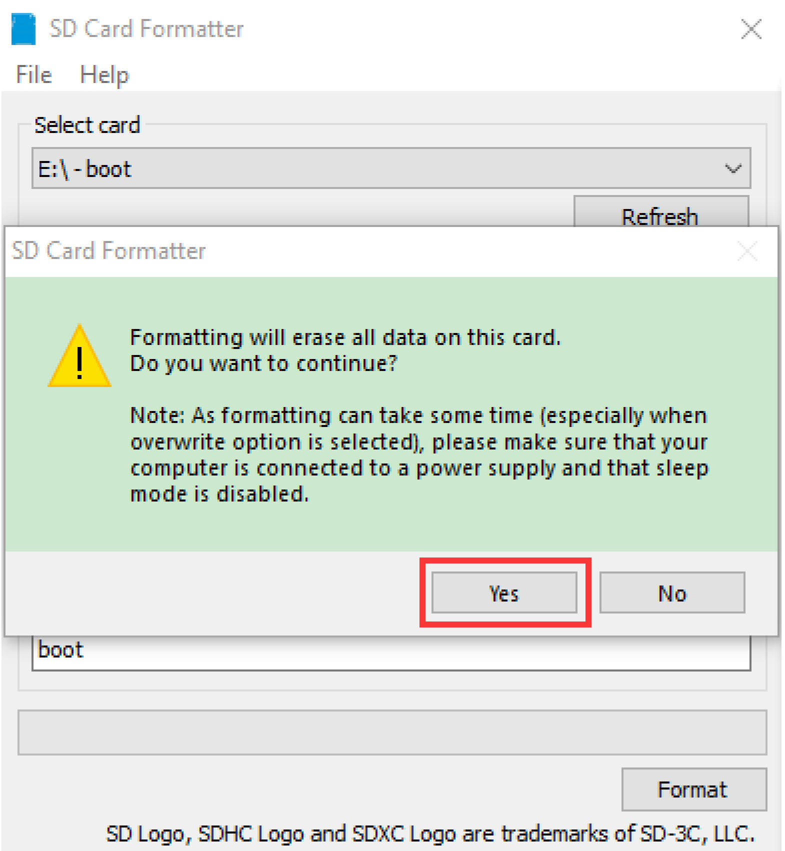
Name	Last modified	Size	Description
Parent Directory		-	
2021-05-07-raspios-buster-armhf-full.info	2021-05-07 16:23	288K	
2021-05-07-raspios-buster-armhf-full.zip	2021-05-07 16:35	2.8G	
2021-05-07-raspios-buster-armhf-full.zip.sha1	2021-05-28 15:49	83	
2021-05-07-raspios-buster-armhf-full.zip.sha256	2021-05-28 15:49	107	
2021-05-07-raspios-buster-armhf-full.zip.sig	2021-05-28 15:00	488	
2021-05-07-raspios-buster-armhf-full.zip.torrent	2021-05-28 15:50	28K	

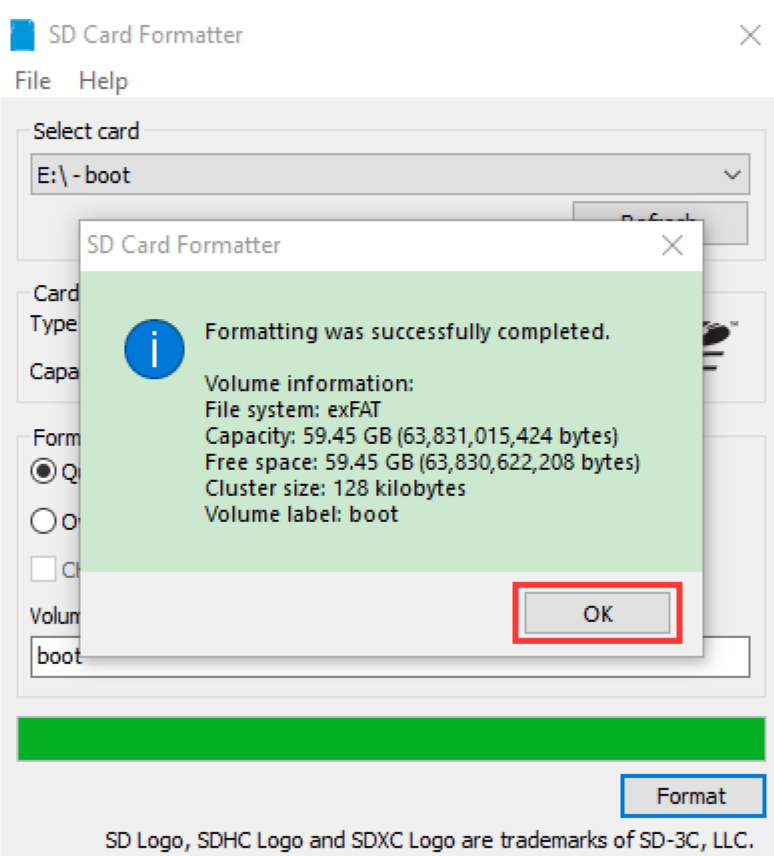
8.1.2 2. Install Raspberry Pi OS system on Raspberry Pi 4B:

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

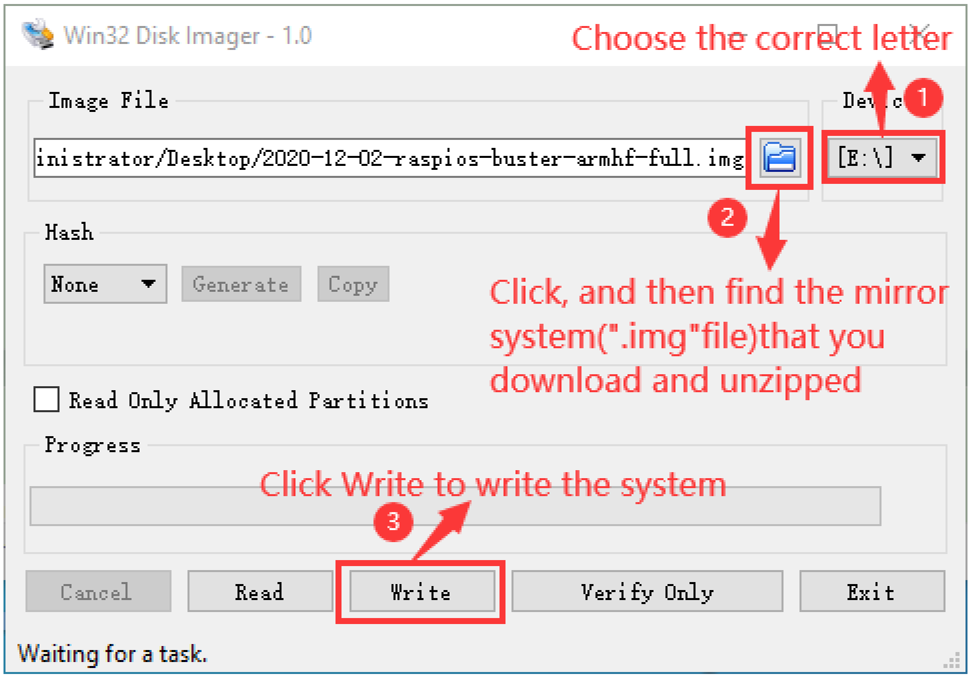


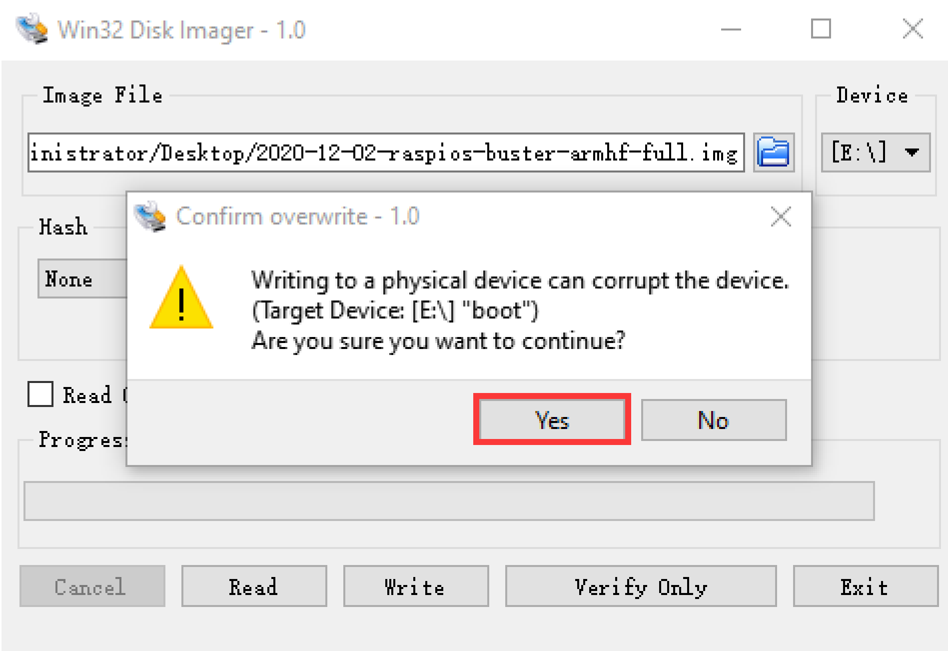

SD Card Formatter
File Help
Select card
E:\ - boot
Refresh
SD Card Formatter
Formatting will erase all data on this card.
Do you want to continue?
Note: As formatting can take some time (especially when overwrite option is selected), please make sure that your computer is connected to a power supply and that sleep mode is disabled.
Yes
No
boot
Format
SD Logo, SDHC Logo and SDXC Logo are trademarks of SD-3C, LLC.

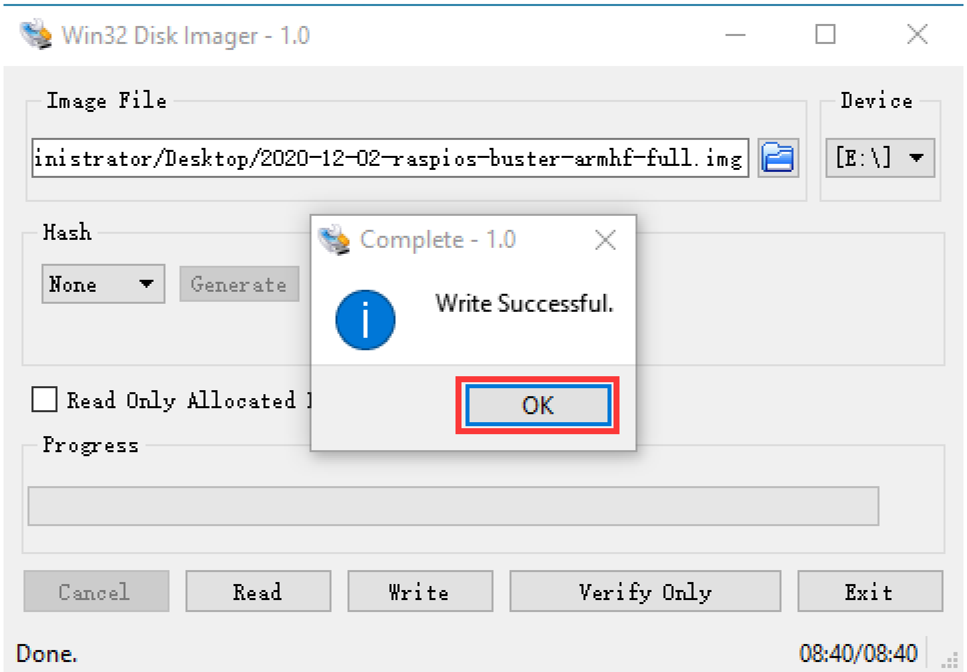

SD Card Formatter
File Help
Select card
E:\ - boot
SD Card Formatter
Card
Type
Formatting was successfully completed.
Volume information:
File system: exFAT
Capacity: 59.45 GB (63,831,015,424 bytes)
Free space: 59.45 GB (63,830,622,208 bytes)
Cluster size: 128 kilobytes
Volume label: boot
OK
boot
Format
SD Logo, SDHC Logo and SDXC Logo are trademarks of SD-3C, LLC.

2.3. Burn system:

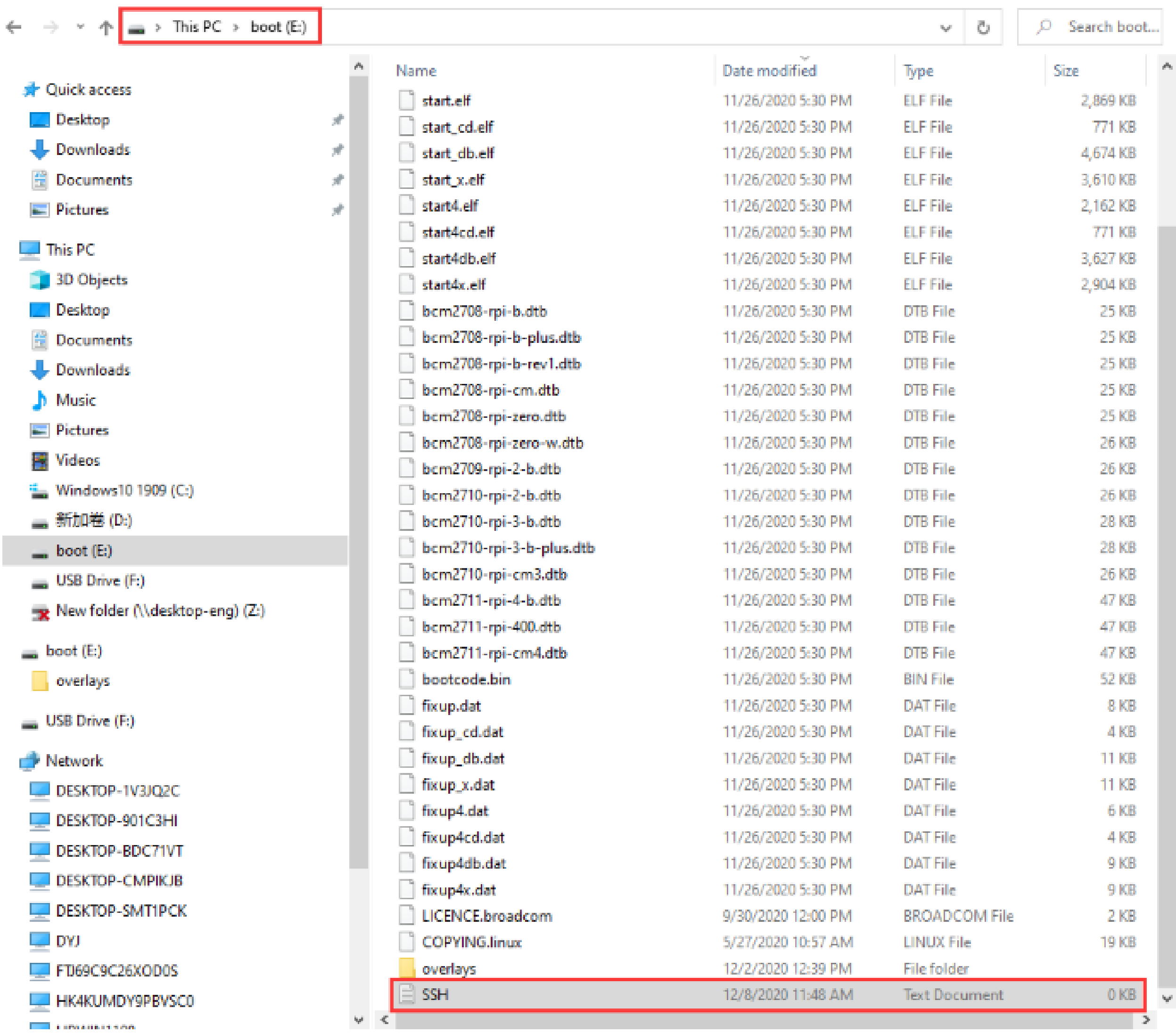
1Use **Win32DiskImager** to burn the official **Raspberry Pi OS** mirror to the TFT memory card.







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:



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

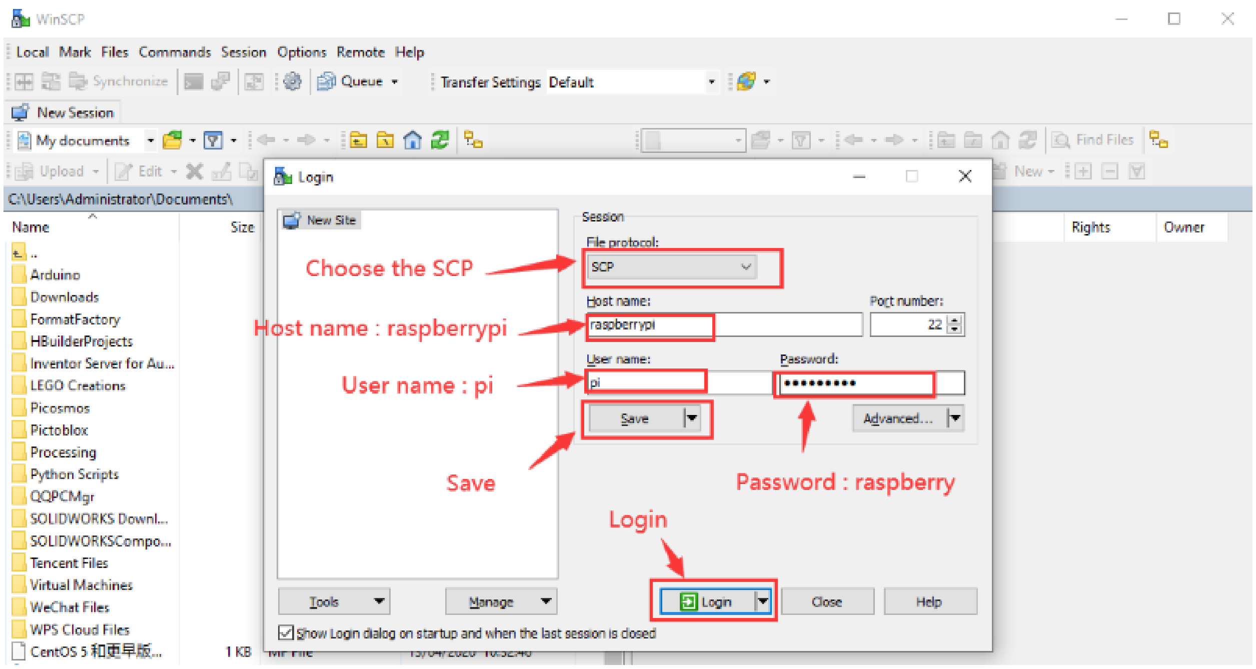
1Preparation

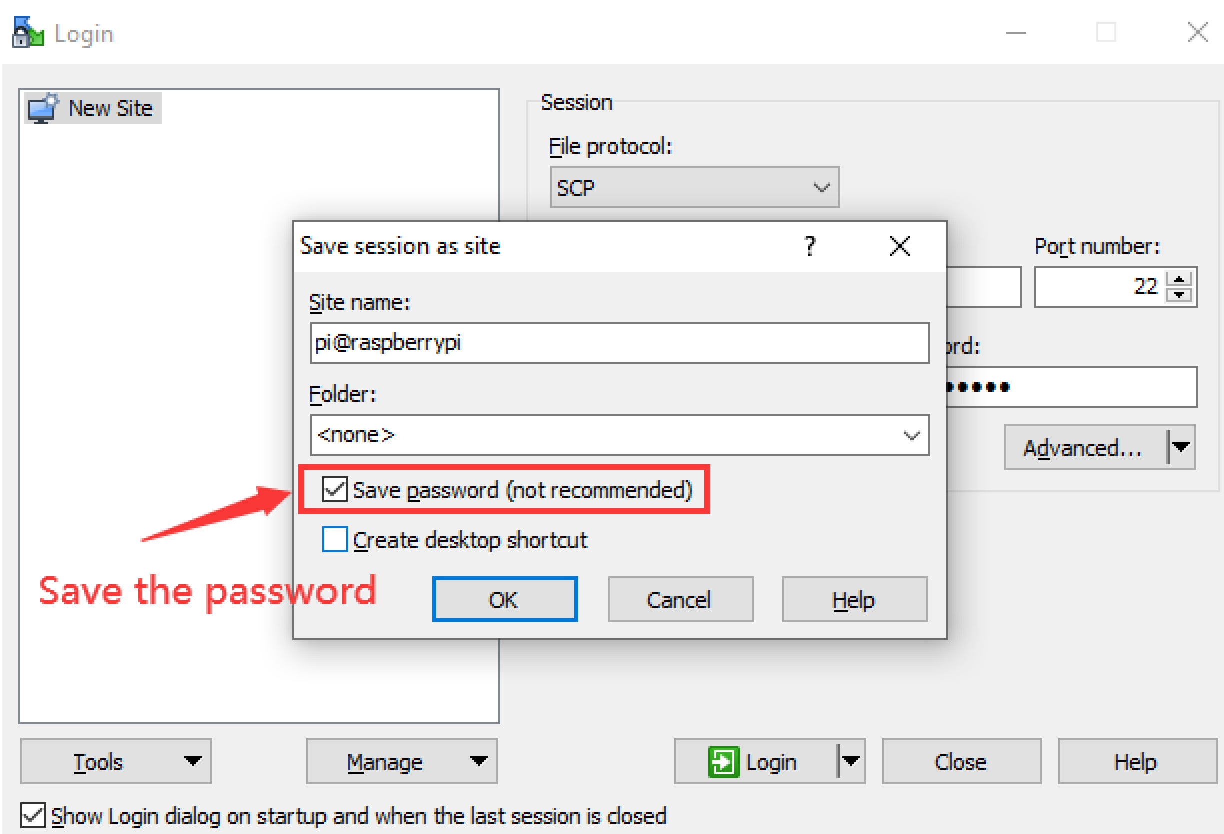
Insert the burned TFT memory card into the Raspberry Pi, connect internet cable and plug in power. If you have a screen and a HDMI cable of Raspberry Pi, connect the screen, and you can see the Raspberry Pi OS startup screen. If you don't have an HDMI cable of Raspberry Pi, you can enter the desktop of Raspberry Pi via SSH remote login software—WinSCP and xrdp.

2Remote login

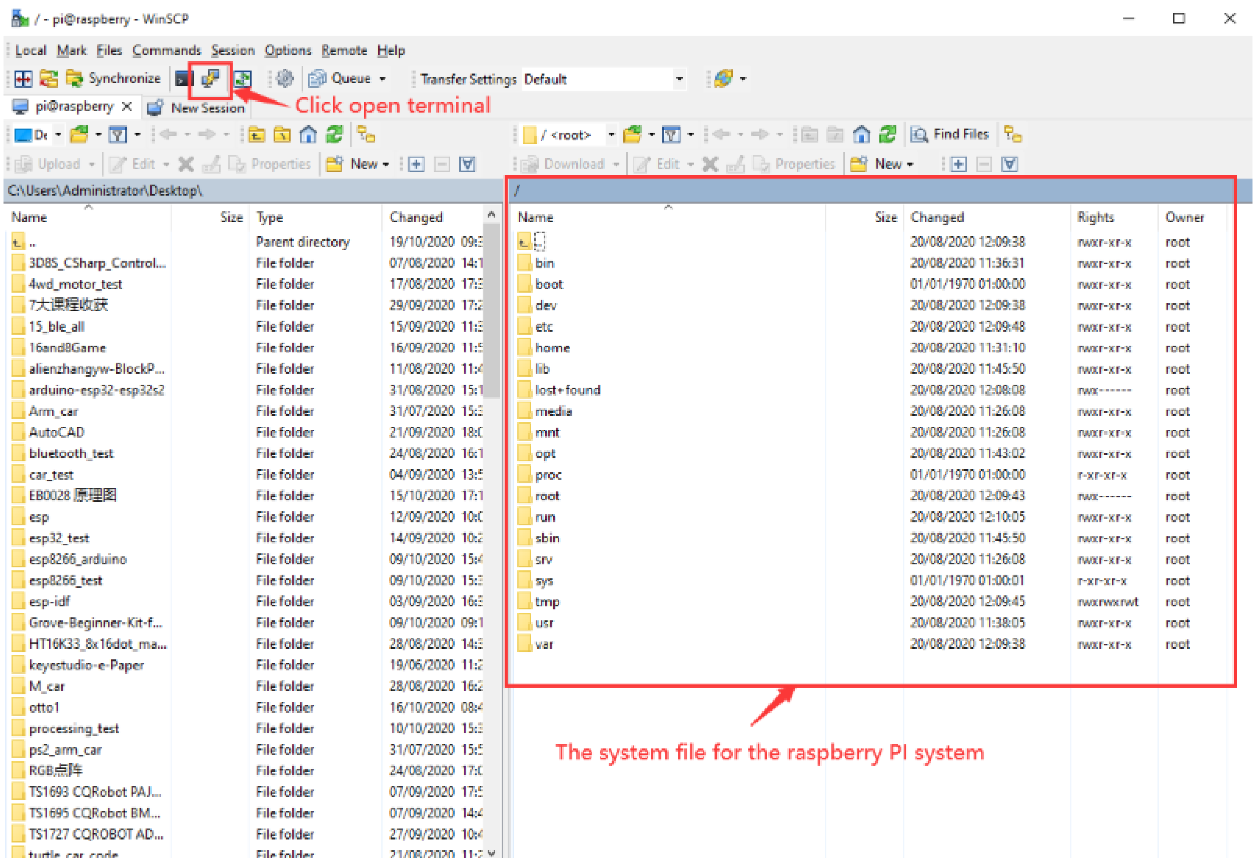
Use WinSCP to log in using the default Raspberry Pi system name, default user name, default password.

Note that only a raspberry pi can be connected to a network.

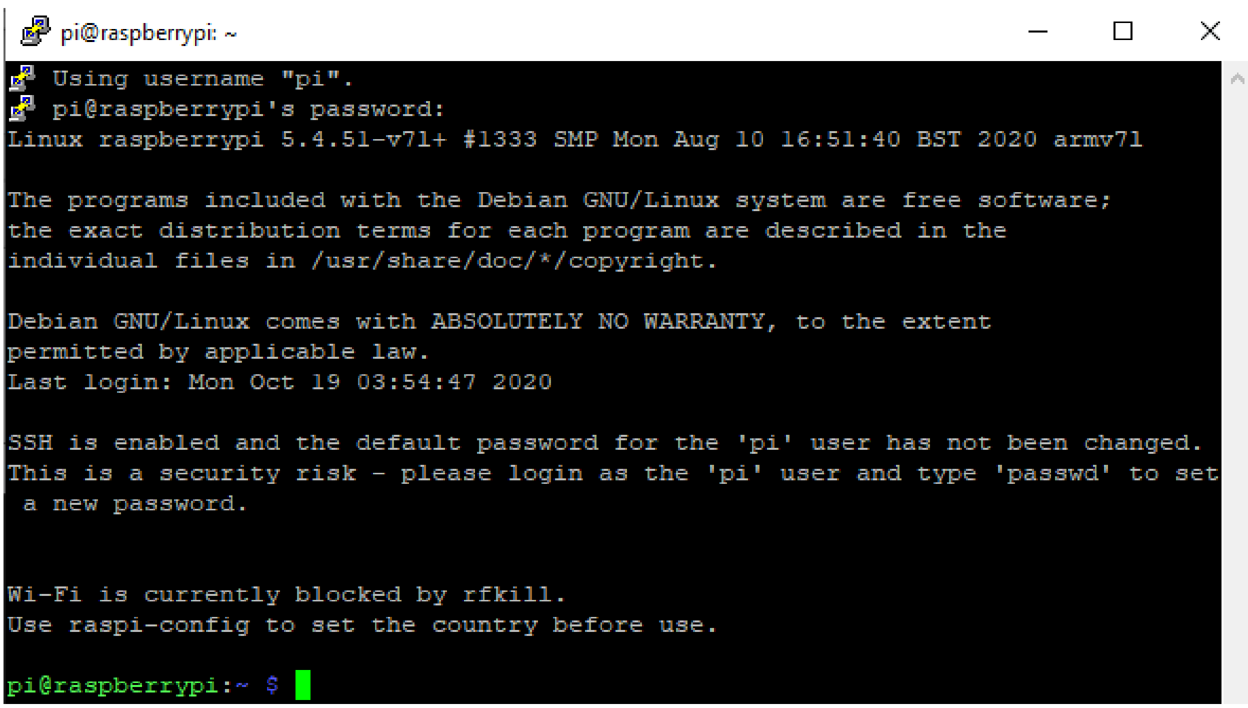




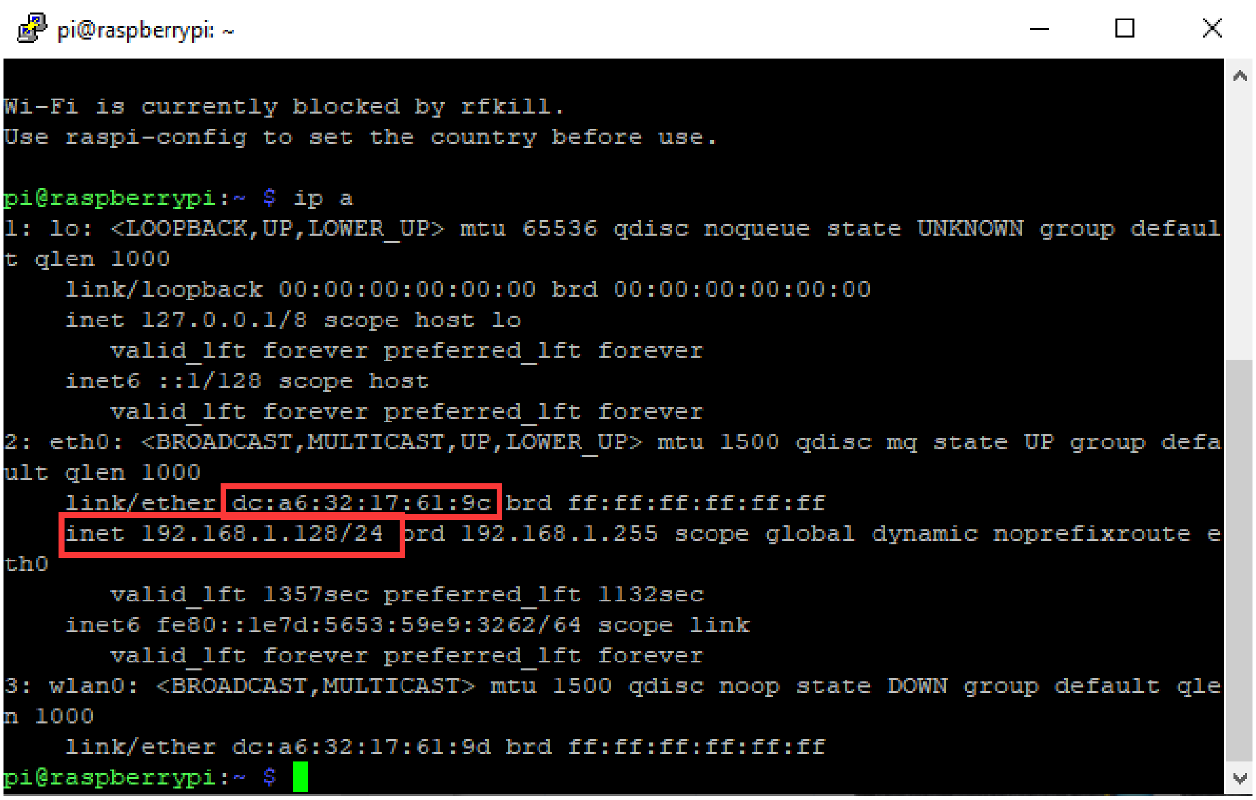
3View the ip address and mac address



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.



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

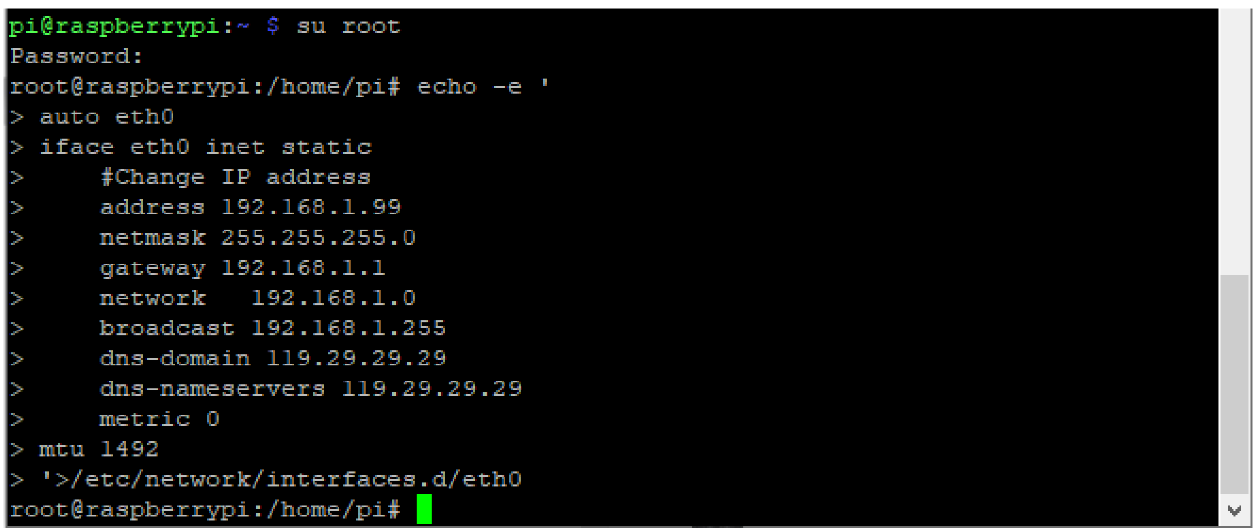
dns-nameservers 119.29.29.29

metric 0

mtu 1492

'>/etc/network/interfaces.d/eth0

Example operation diagram, as follows



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,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group defaul
t qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    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 1492 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 192.168.1.99/24 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
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

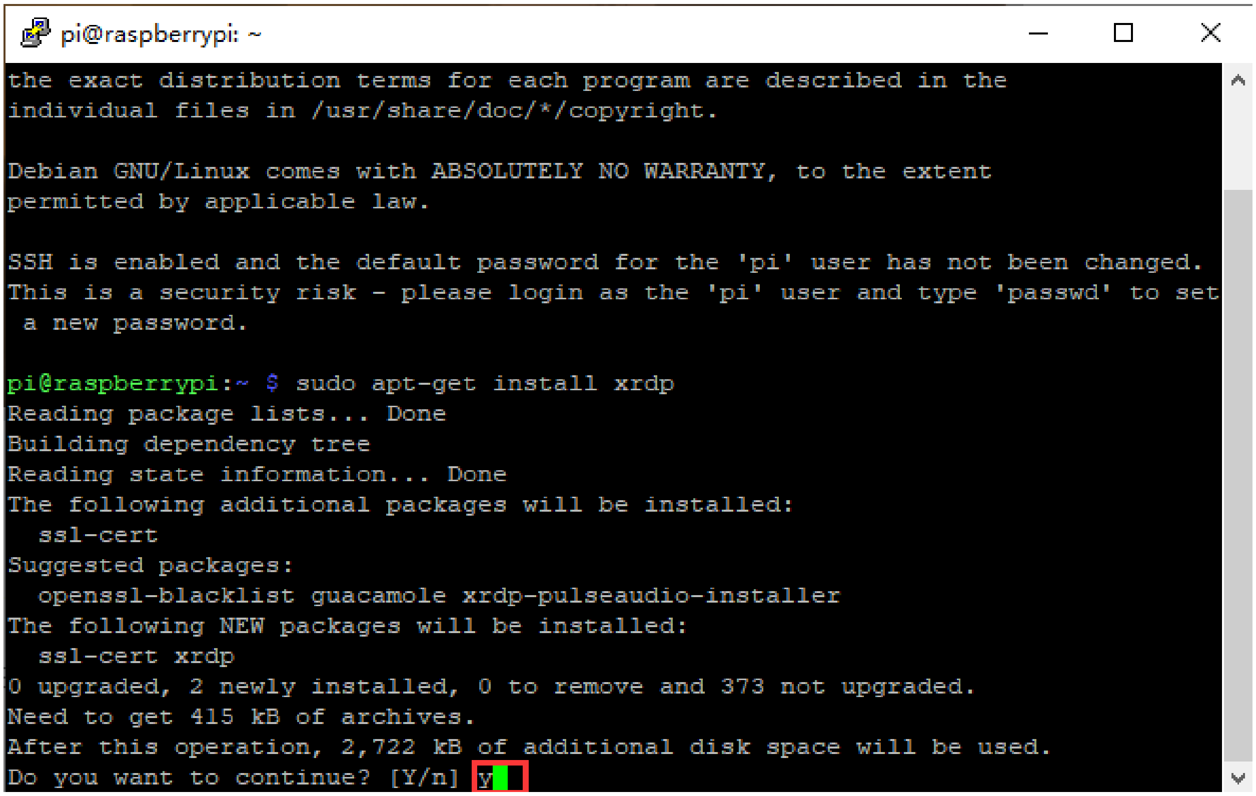
Installation commands:

Switch to root User: su root

Installation commands: apt-get install xrdp

Enter y and tap“**Enter**”keyboard. . .

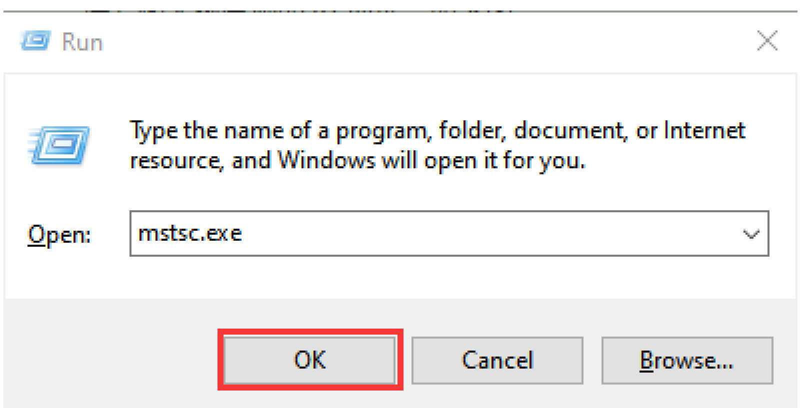
As shown below:



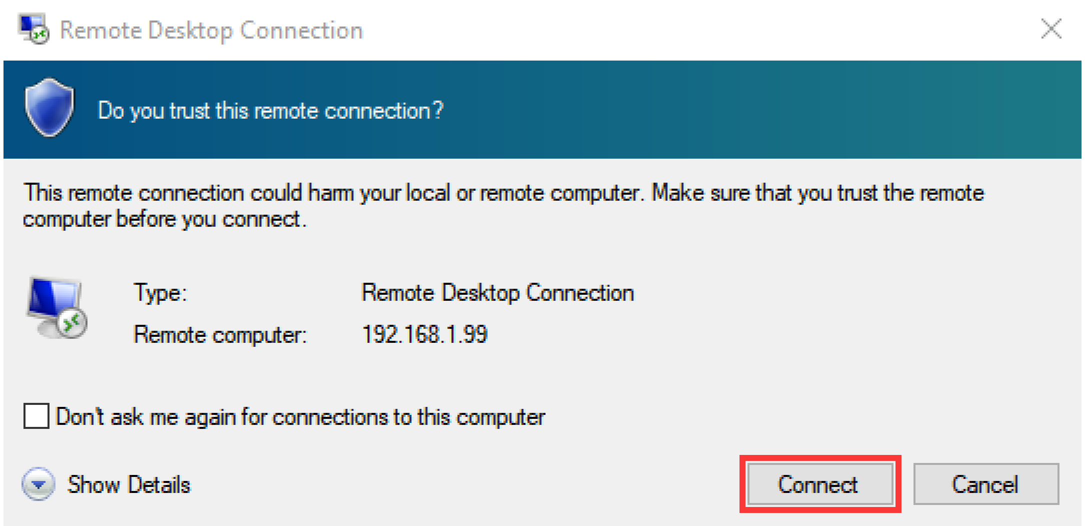
Open the remote desktop connection on Windows

Press **WIN+R** on keyboard and enter mstsc.exe.

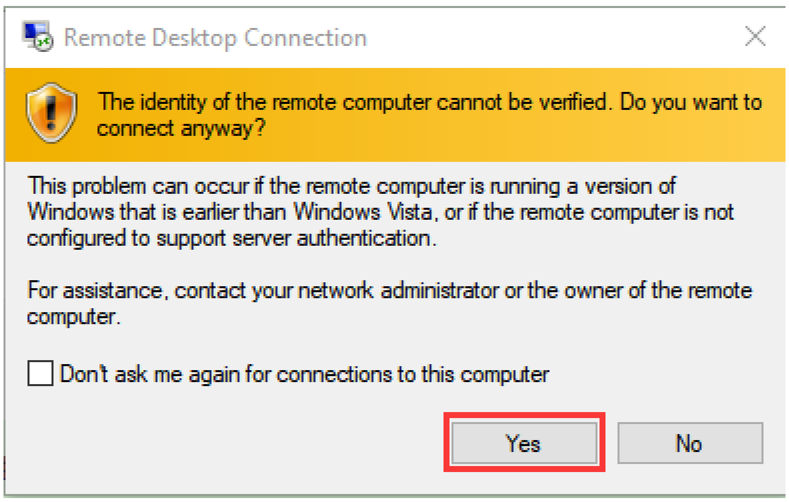
As shown below:



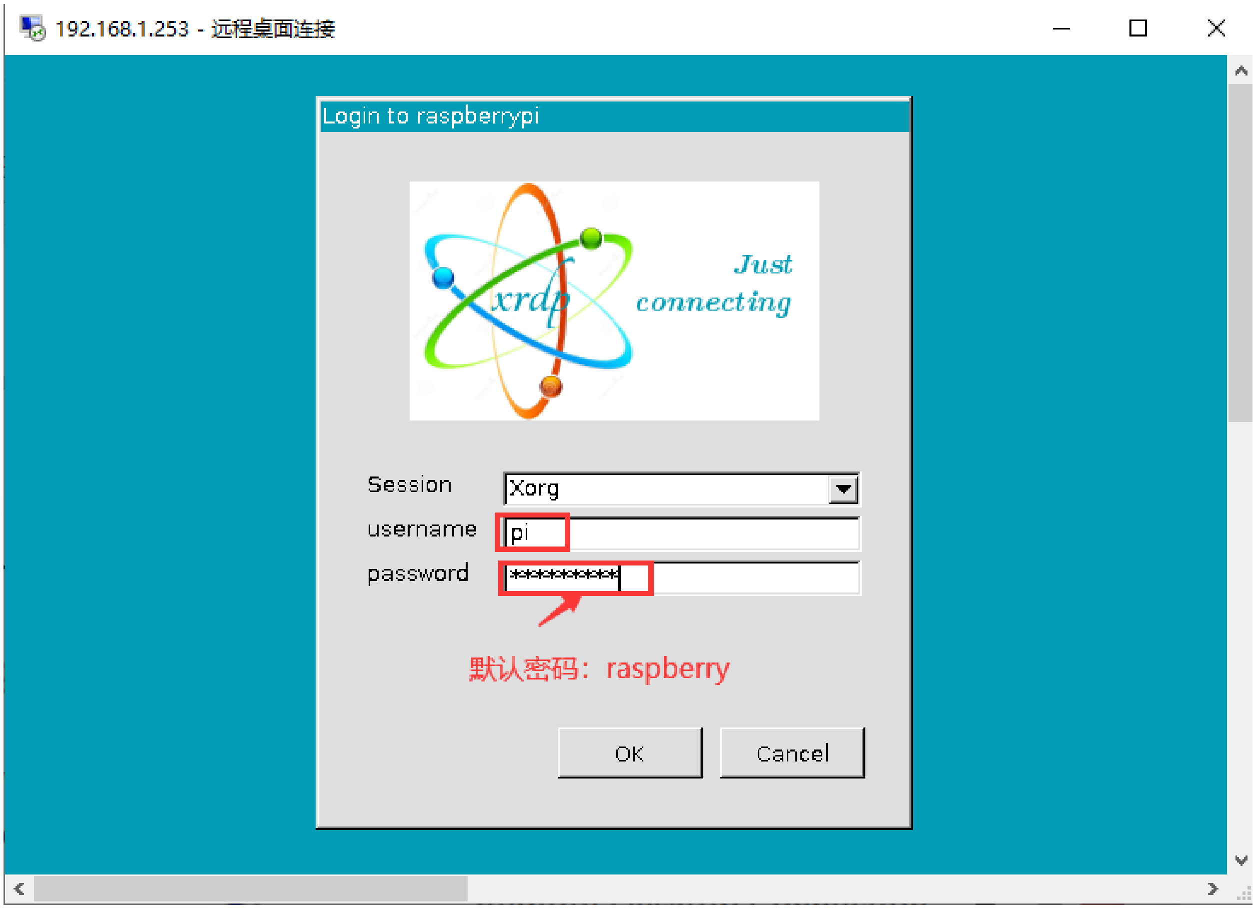
Enter the IP address of the Raspberry Pi, as shown below. Click "Connect" and then click "Connect"again. 192.168.1.99 is the ip address we use, you could change it into your IP address.



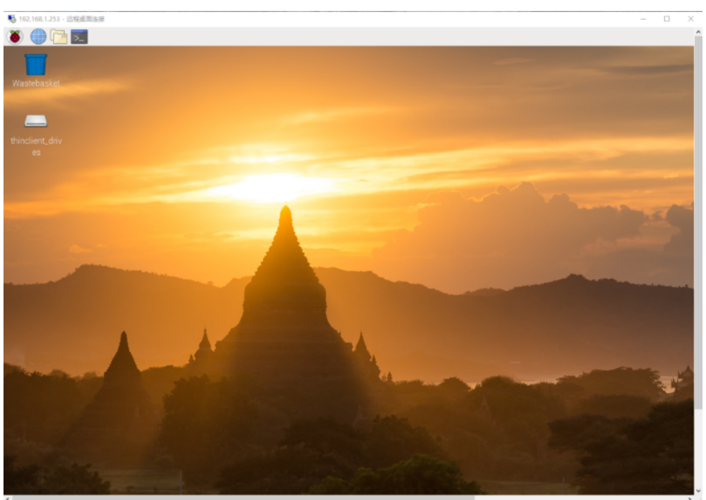
A prompt will appear and you can click"Yes".



Then enter the user name: pi ,and the default password: raspberry, as shown below:



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.

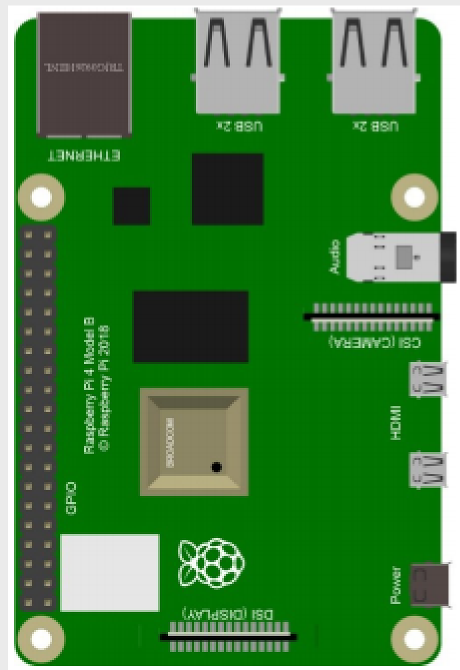
8.1.3 3. Preparation of C language control basic hardware:

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.

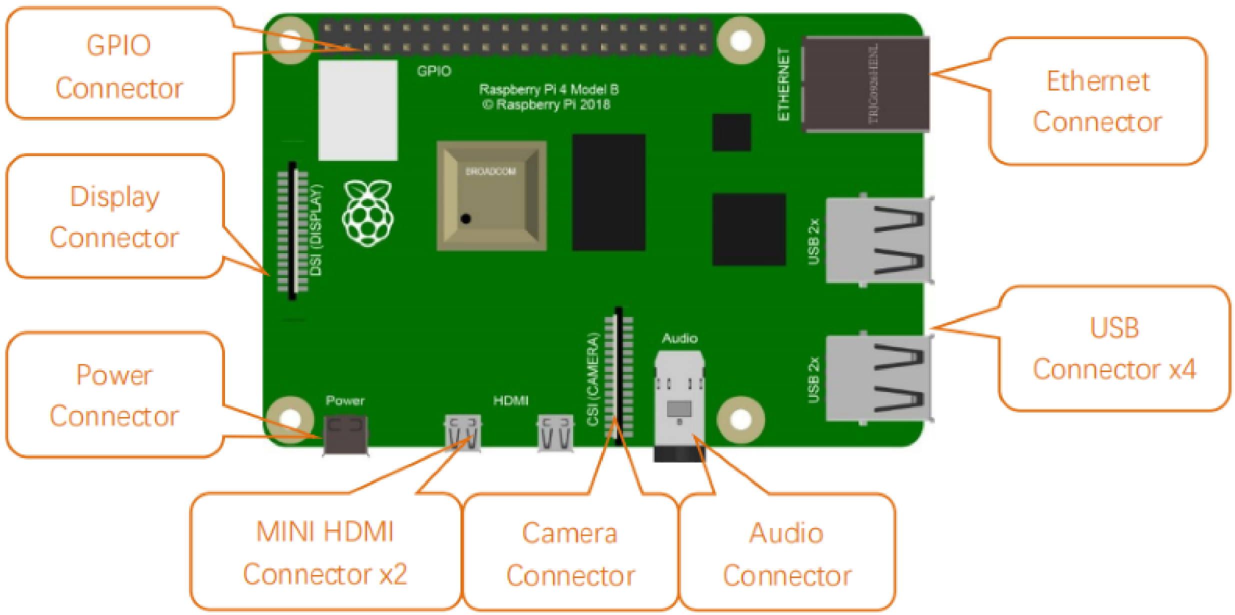
(1)Description of basic raspberry pi accessories

Raspberry Pi 4B

Below are the raspberry pi pictures and model pictures supported by this learning kit. There are 40 pins.

Raspberry Pi 4B	Raspberry Pi 4B Model
	

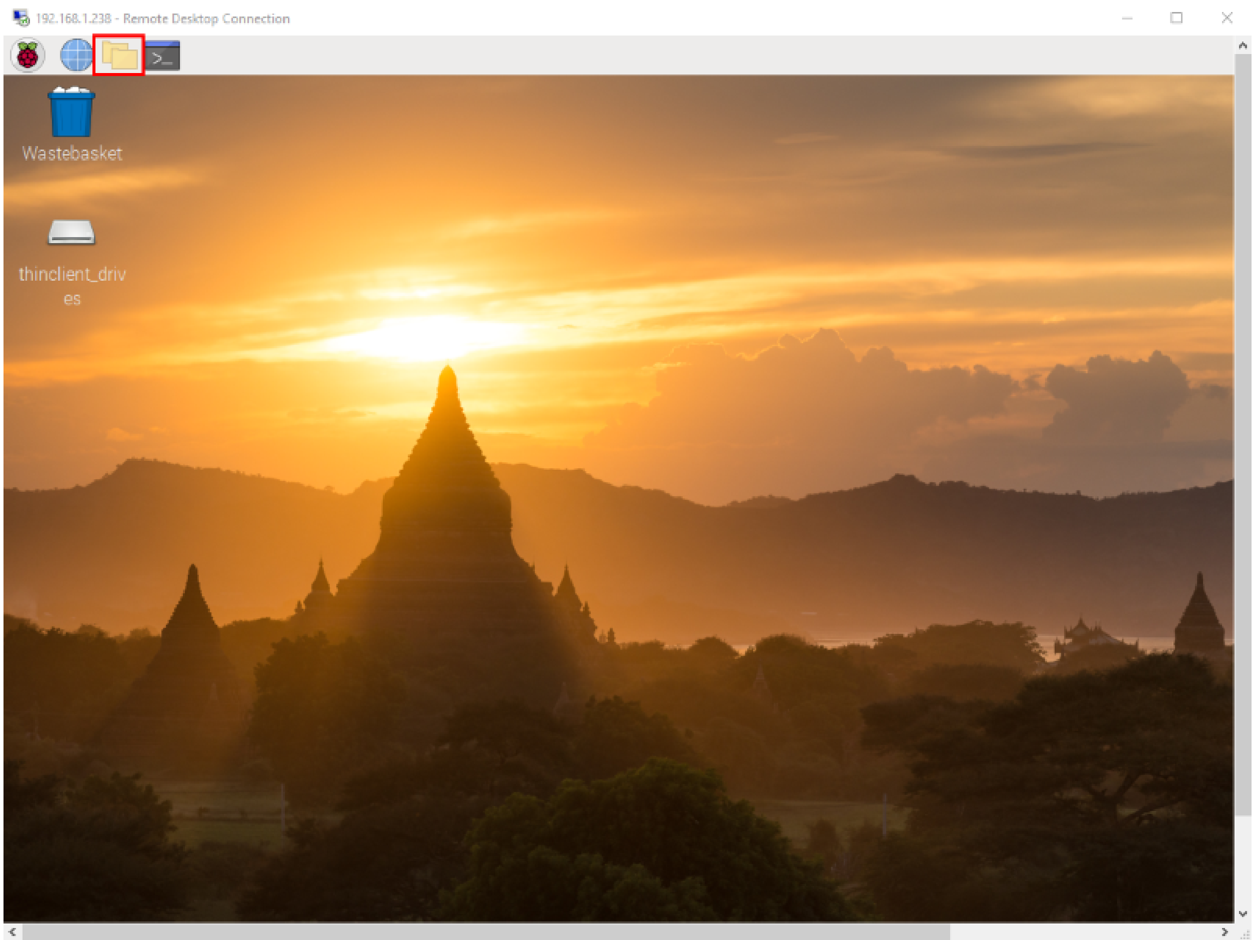
Hardware Interfaces

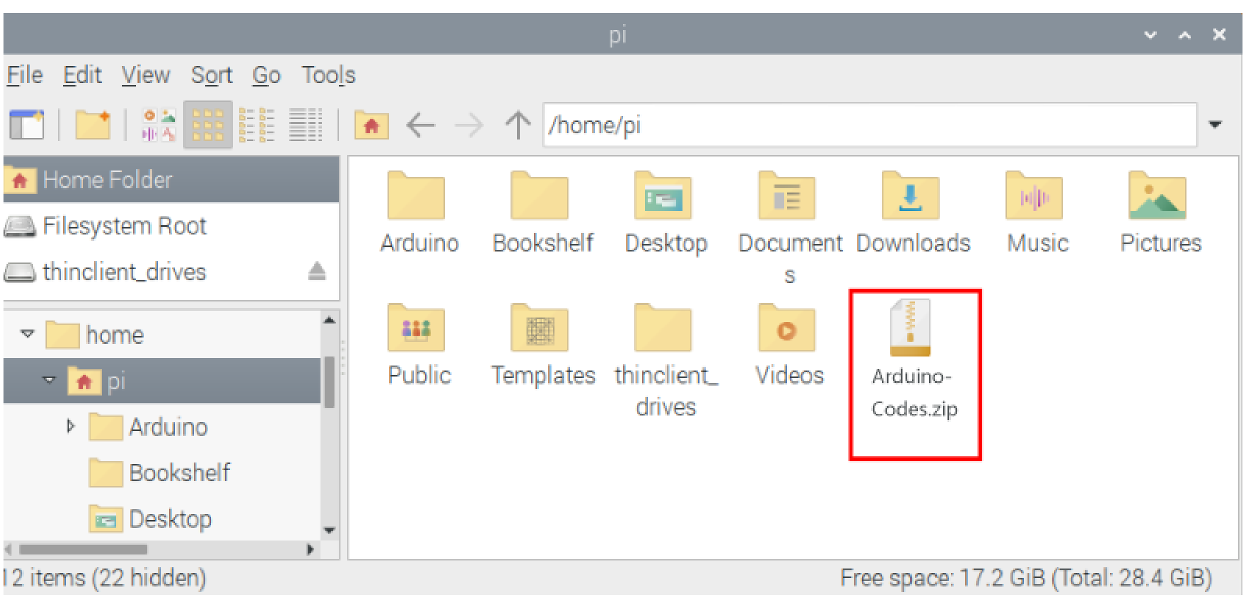


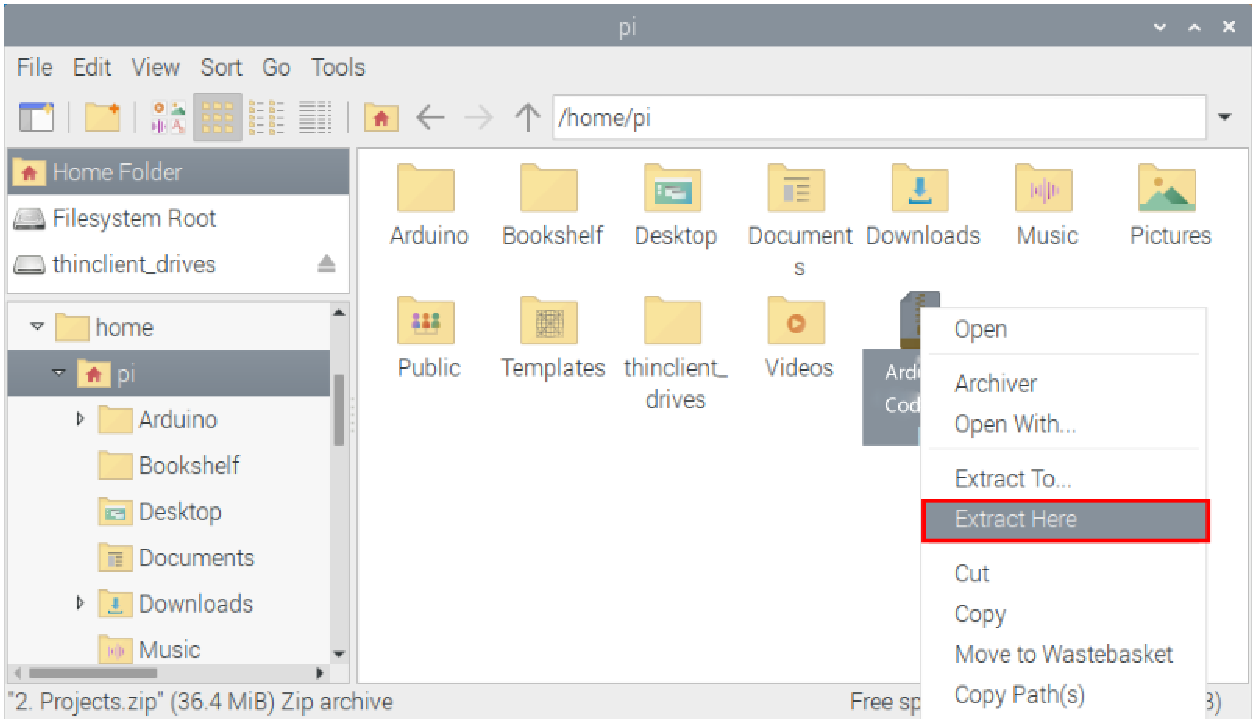
(2)Raspberry Pi +ESP32 mainboard + breadboard +USB cable, as shown below

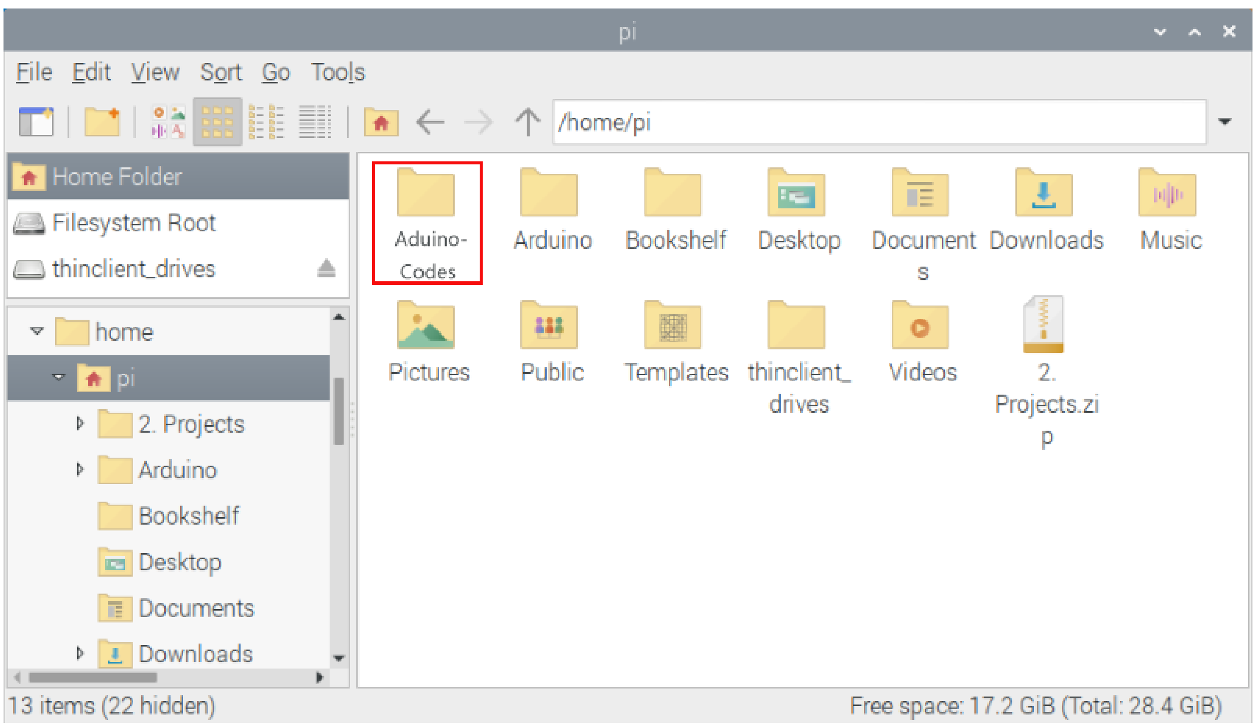
(3)Copy Example Code Folder to Raspberry Pi

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:

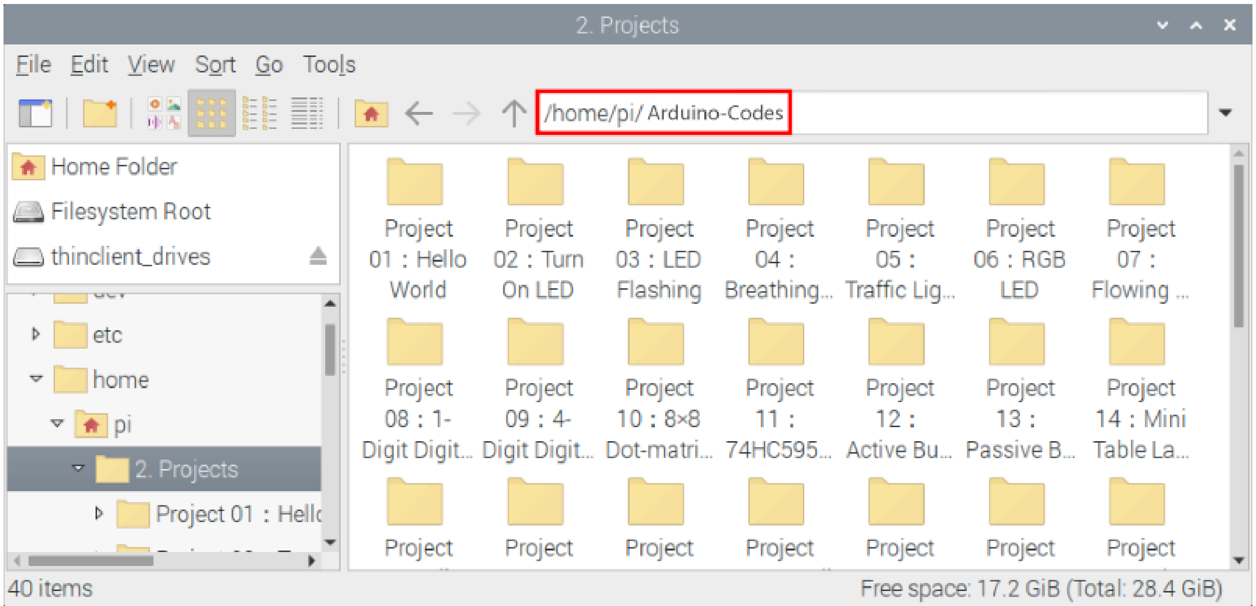








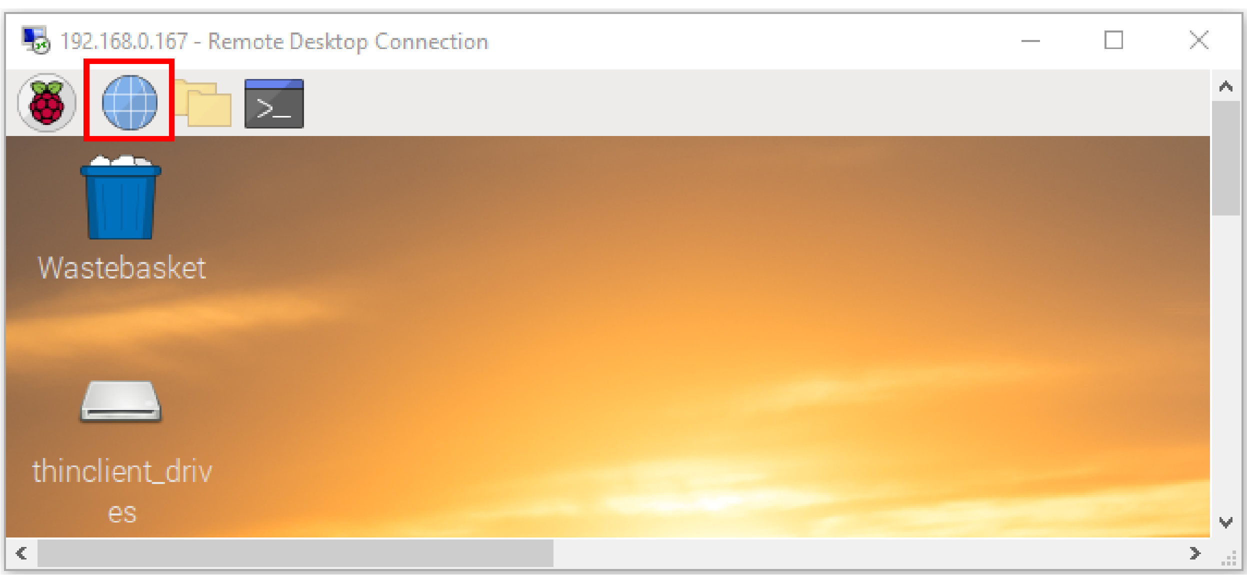
Double-click the **Arduino-Codes** file, as shown below.



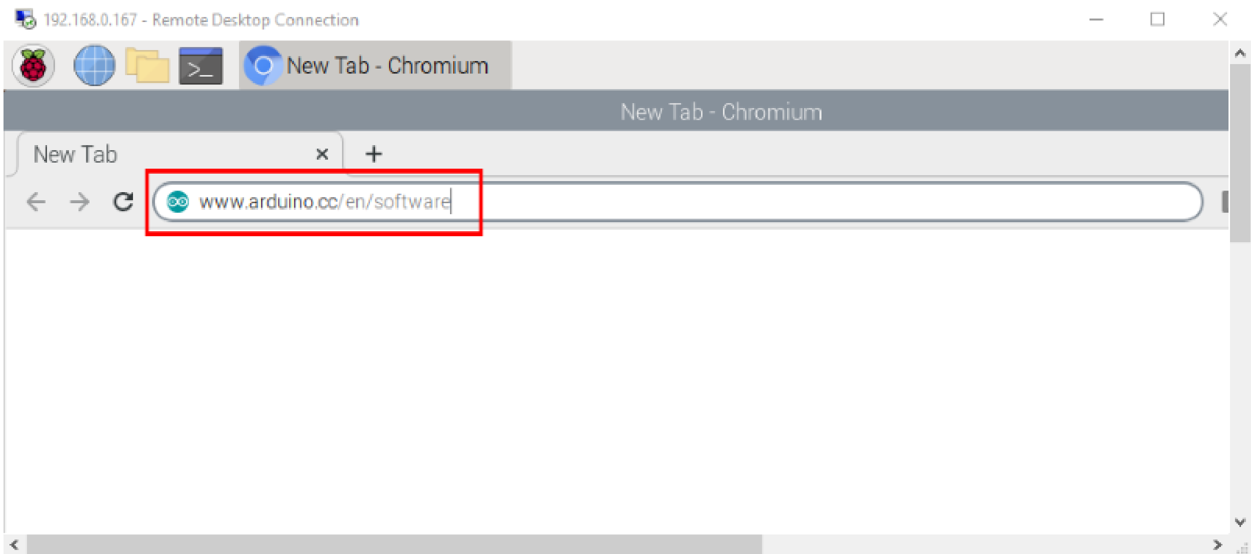
8.2 Linux SystemRaspberry Pi

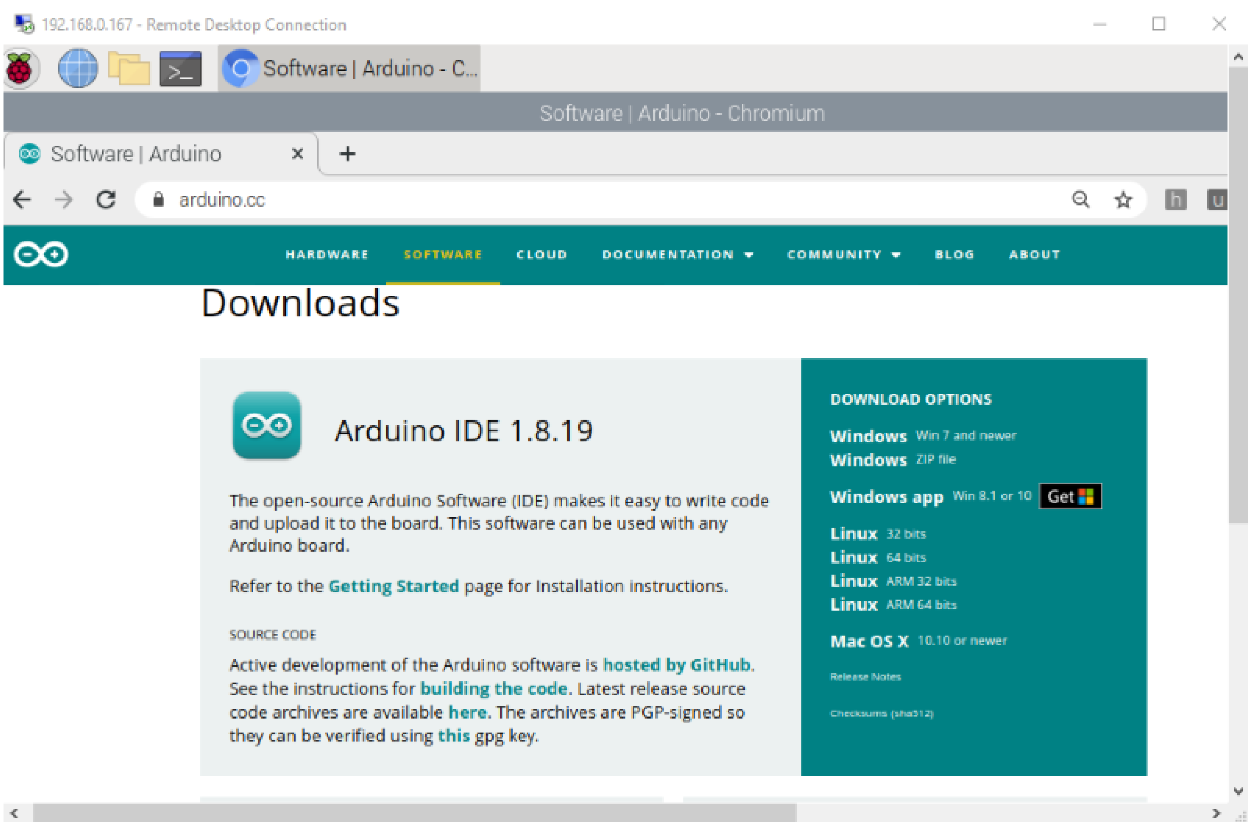
8.2.1 1. Download and install Arduino IDE

1First, click on Raspberry Pi's browser.

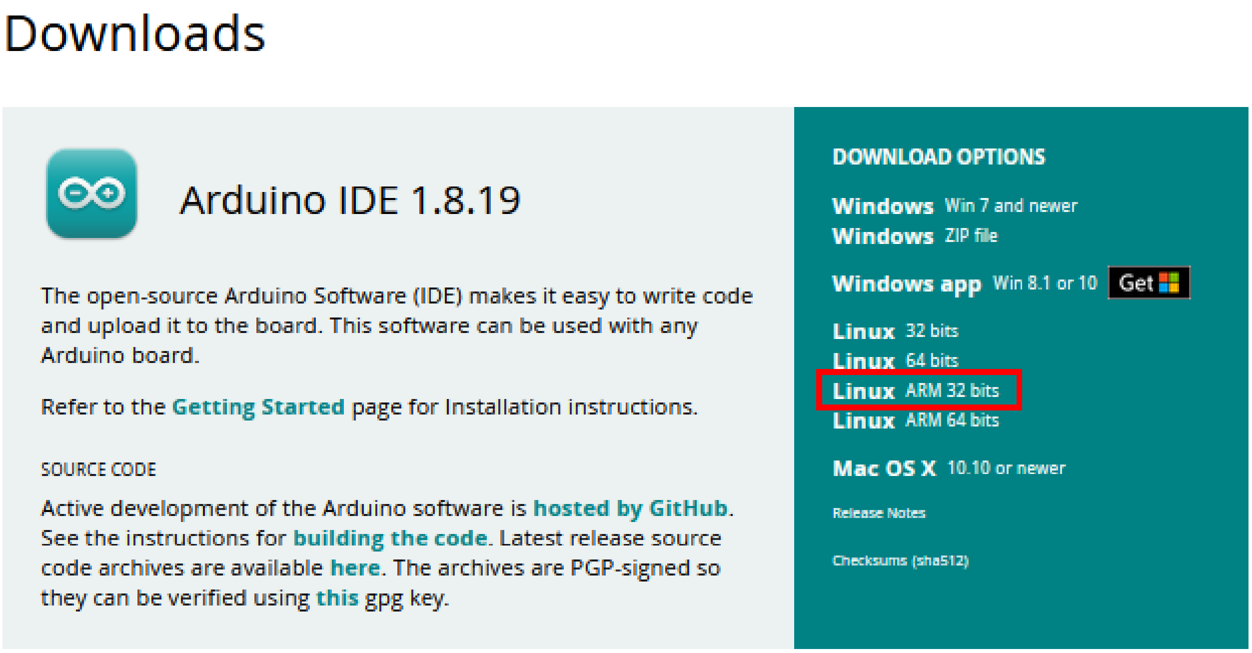


2. Enter the Official Arduino website in your browserwww.arduino.cc/en/software , as shown below:

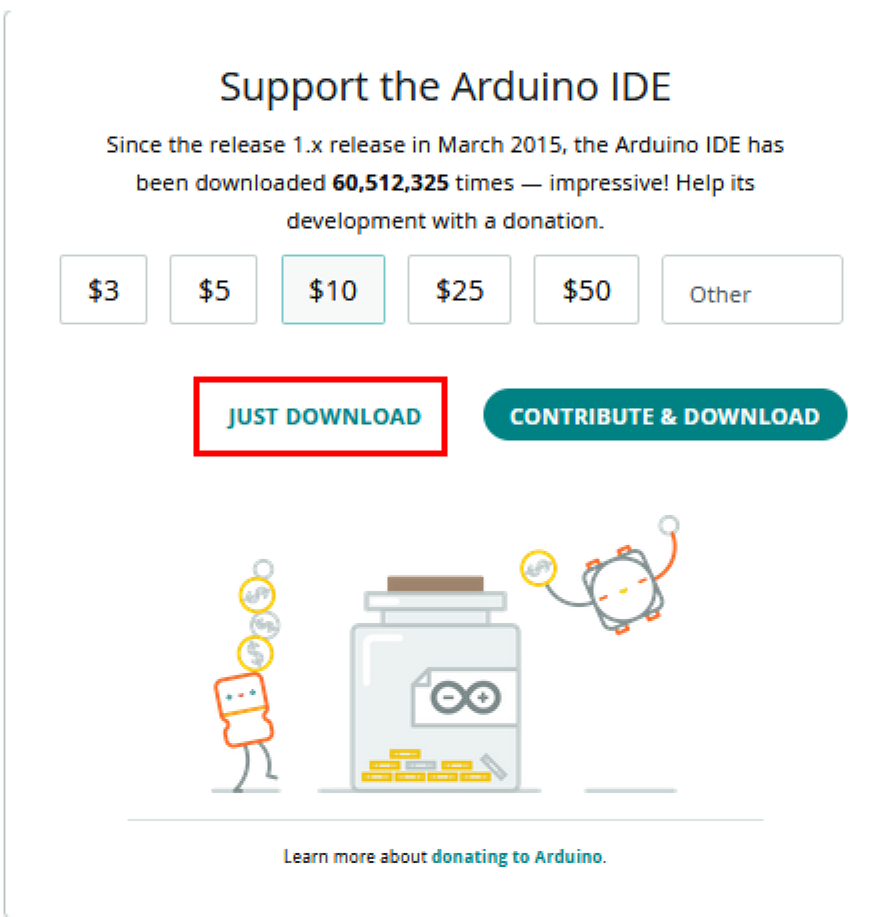




3There are various versions of IDE for Arduino. Just download a version compatible with your system (install the lasted Arduino IDE) and click "Linux ARM 32 bits".

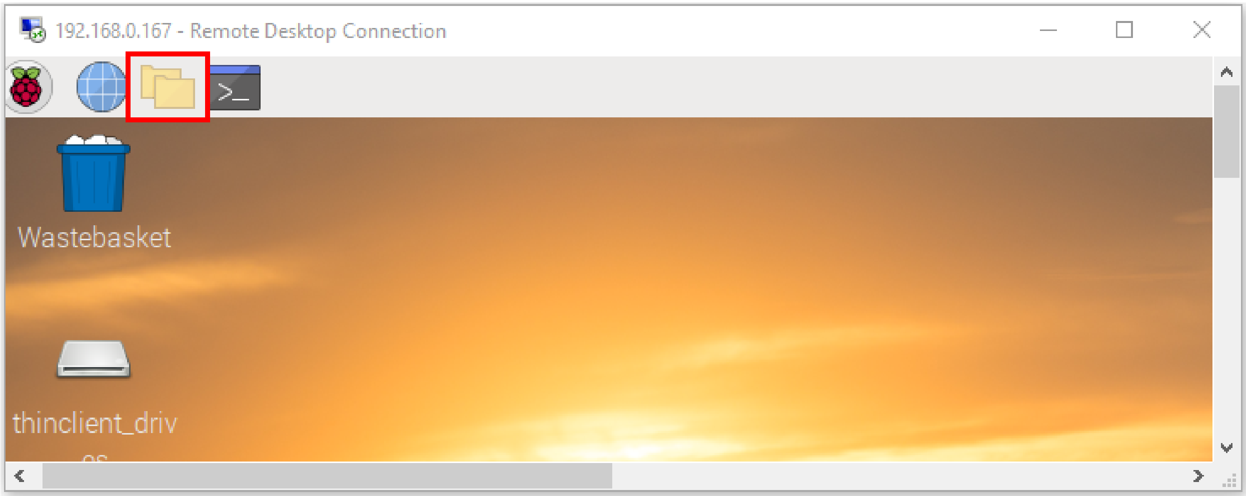


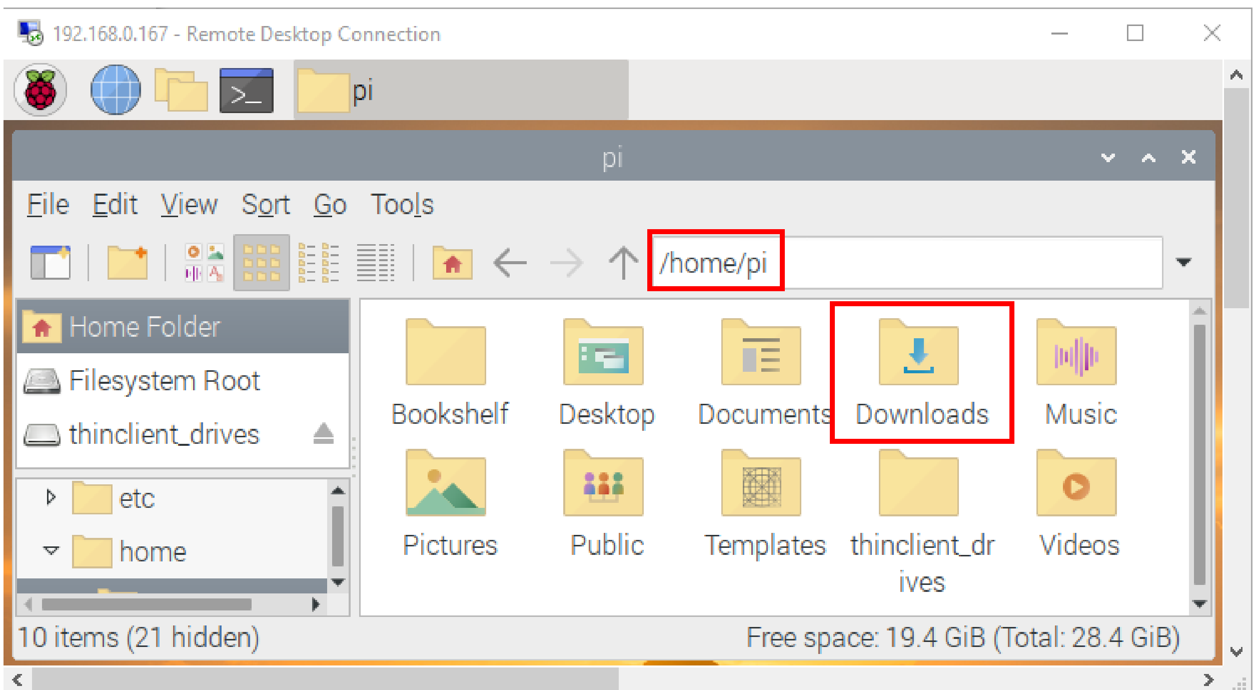
In general, you can click **JUST DOWNLOAD** to download, although if you like, you can choose a small sponsorship to help the great Arduino open source cause.

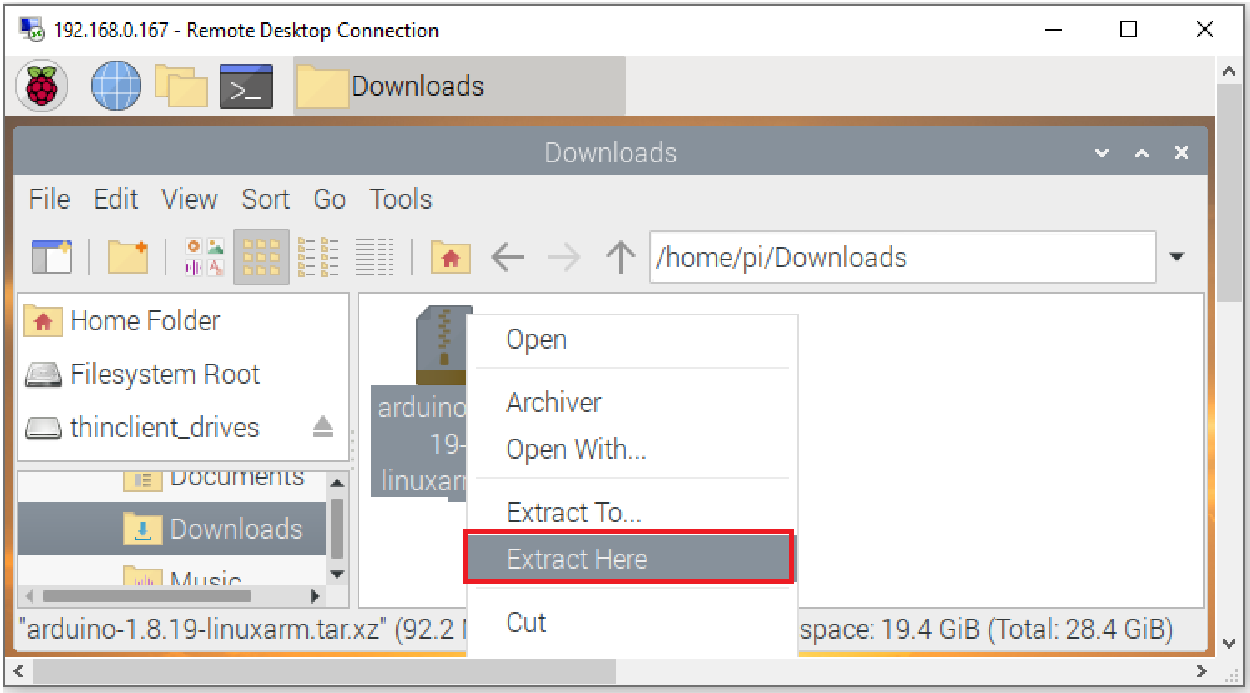


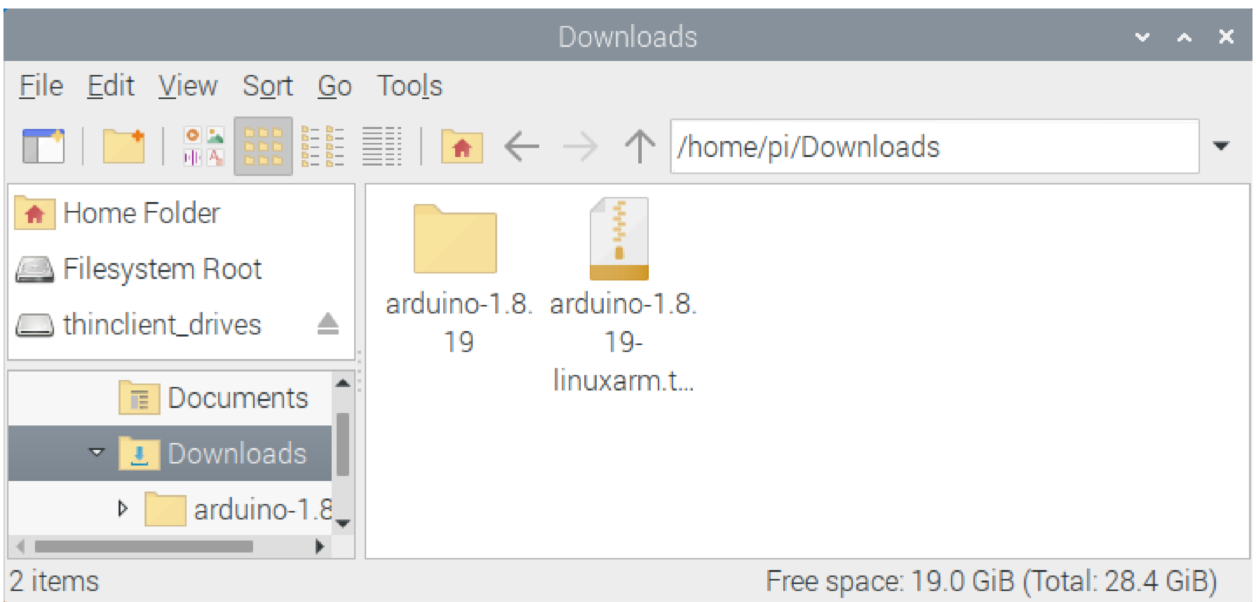
After a few seconds, the lasted Arduino IDEArduino 1.8.19 versionzip file can be directly downloaded.

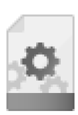
(4) Click , then find the **Downloads** file from the pi folder and click it. In the **Downloads** folder, you can see the package“arduino-1.8.19-linuxarm.tar.xz”that you just downloaded. Then unzip the package“arduino-1.8.19-linuxarm.tar.xz”, after a while, the package is unzipped.



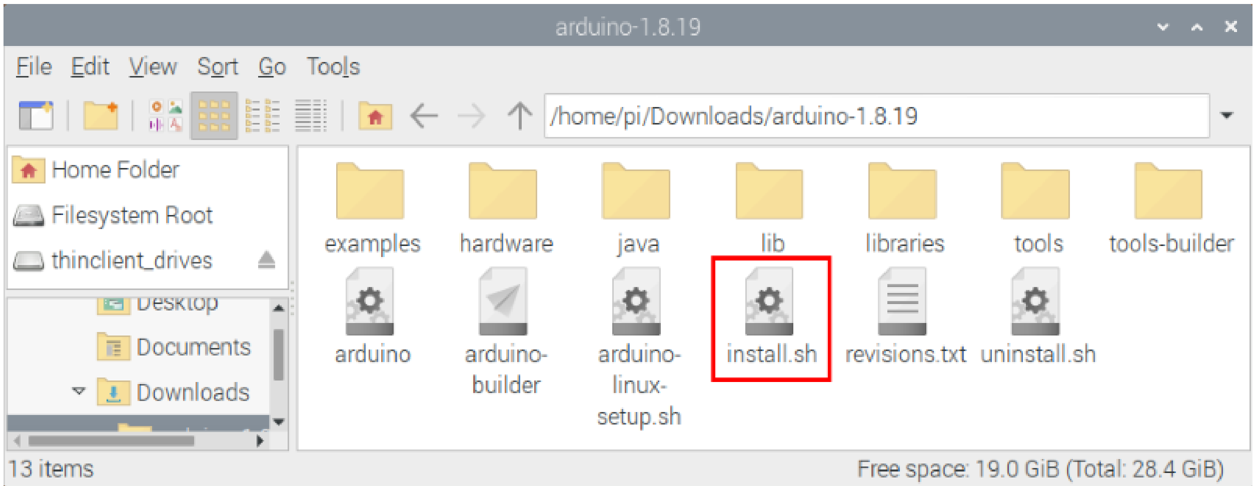


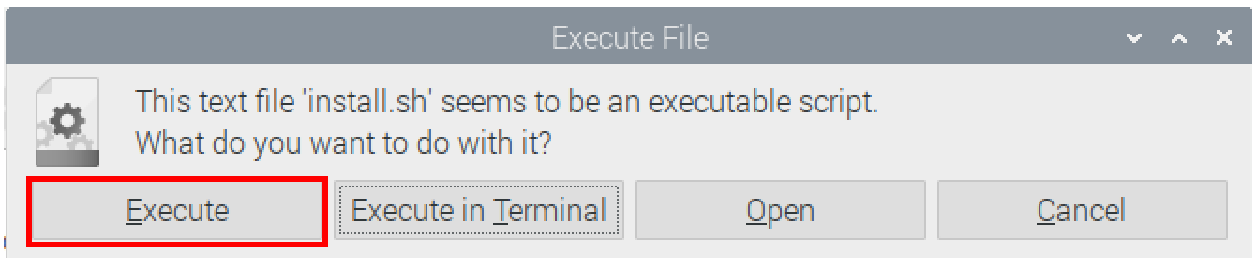


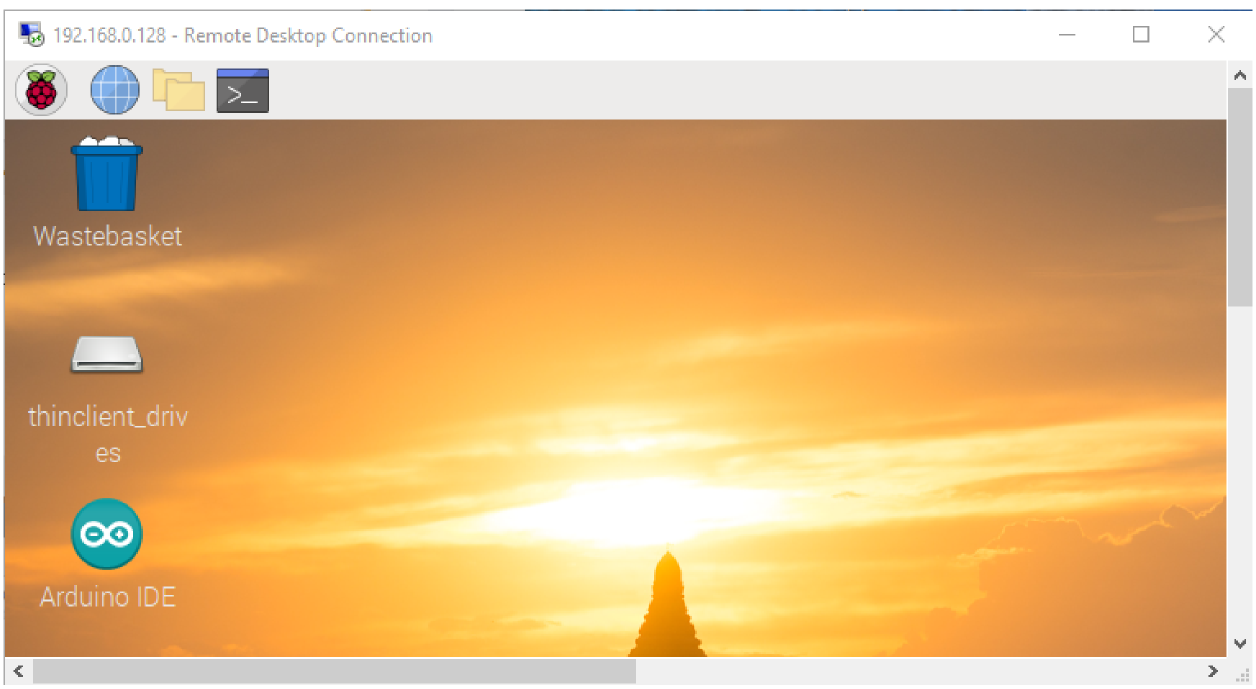




5Click install.sh file and tap itclick"Execute"in the dialog that appears to install the Arduino IDE. Once installed, an Arduino software shortcut is generated in the desktop.



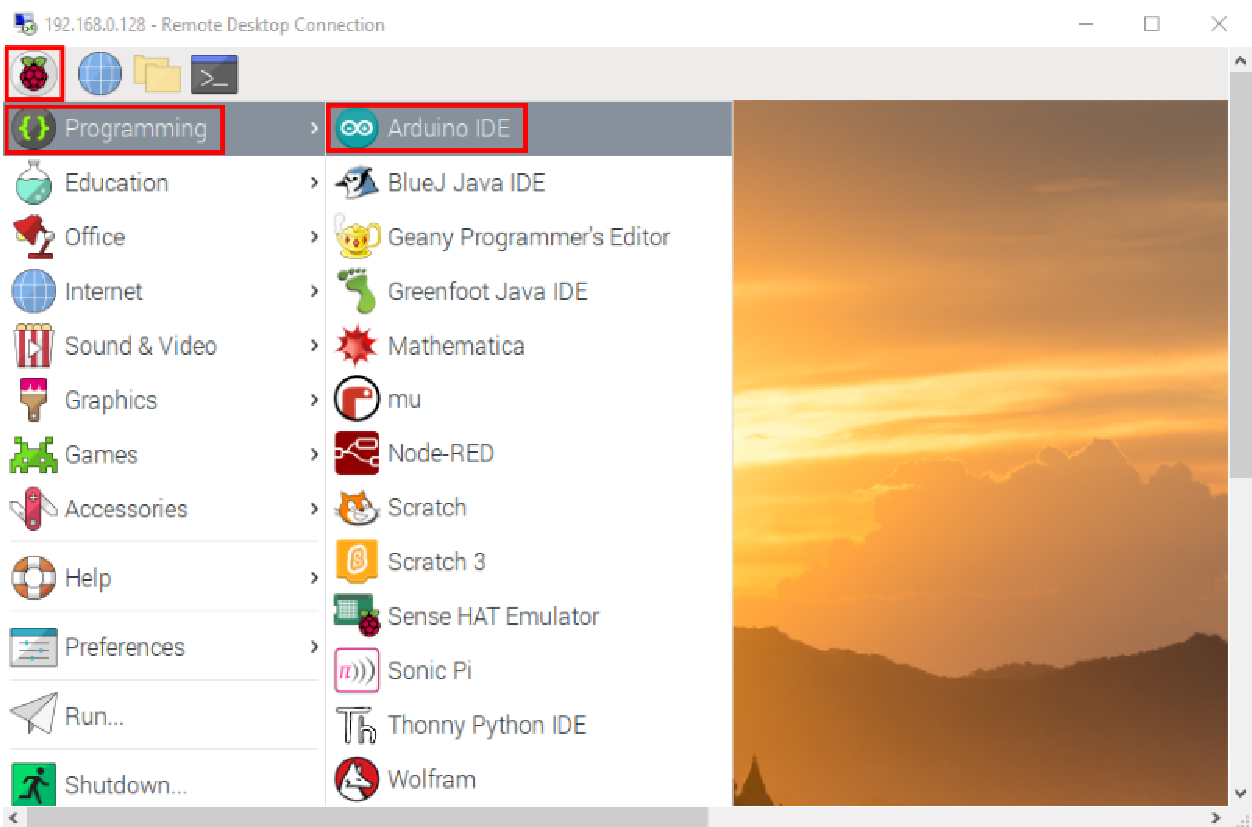


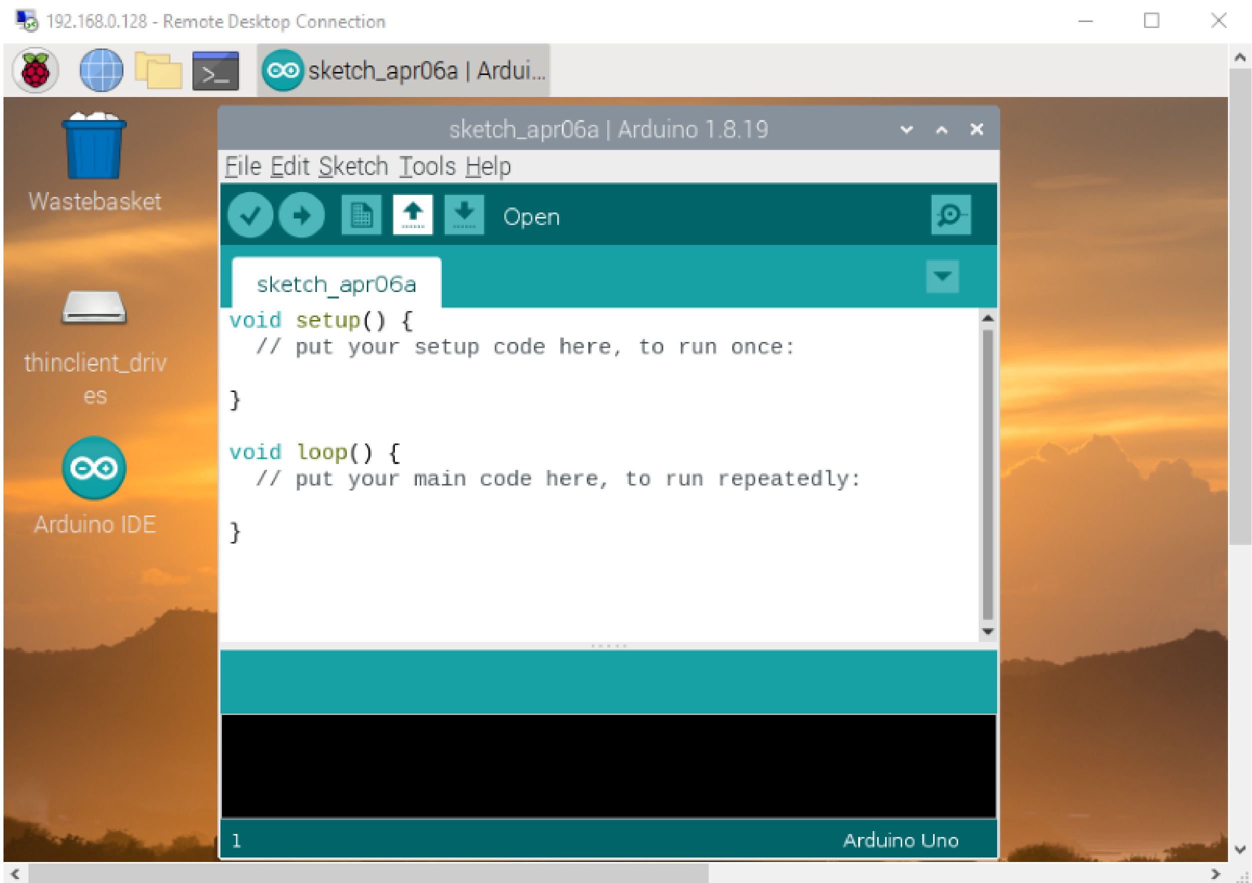


6Click 

and click 

to open the Arduino IDE.

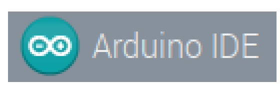

192.168.0.128 - Remote Desktop Connection
Programming
Education
Office
Internet
Sound & Video
Graphics
Games
Accessories
Help
Preferences
Run...
Shutdown...
Arduino IDE
BlueJ Java IDE
Geany Programmer's Editor
Greenfoot Java IDE
Mathematica
mu
Node-RED
Scratch
Scratch 3
Sense HAT Emulator
Sonic Pi
Thonny Python IDE
Wolfram



8.2.2 2. Install the ESP32 on Arduino IDE

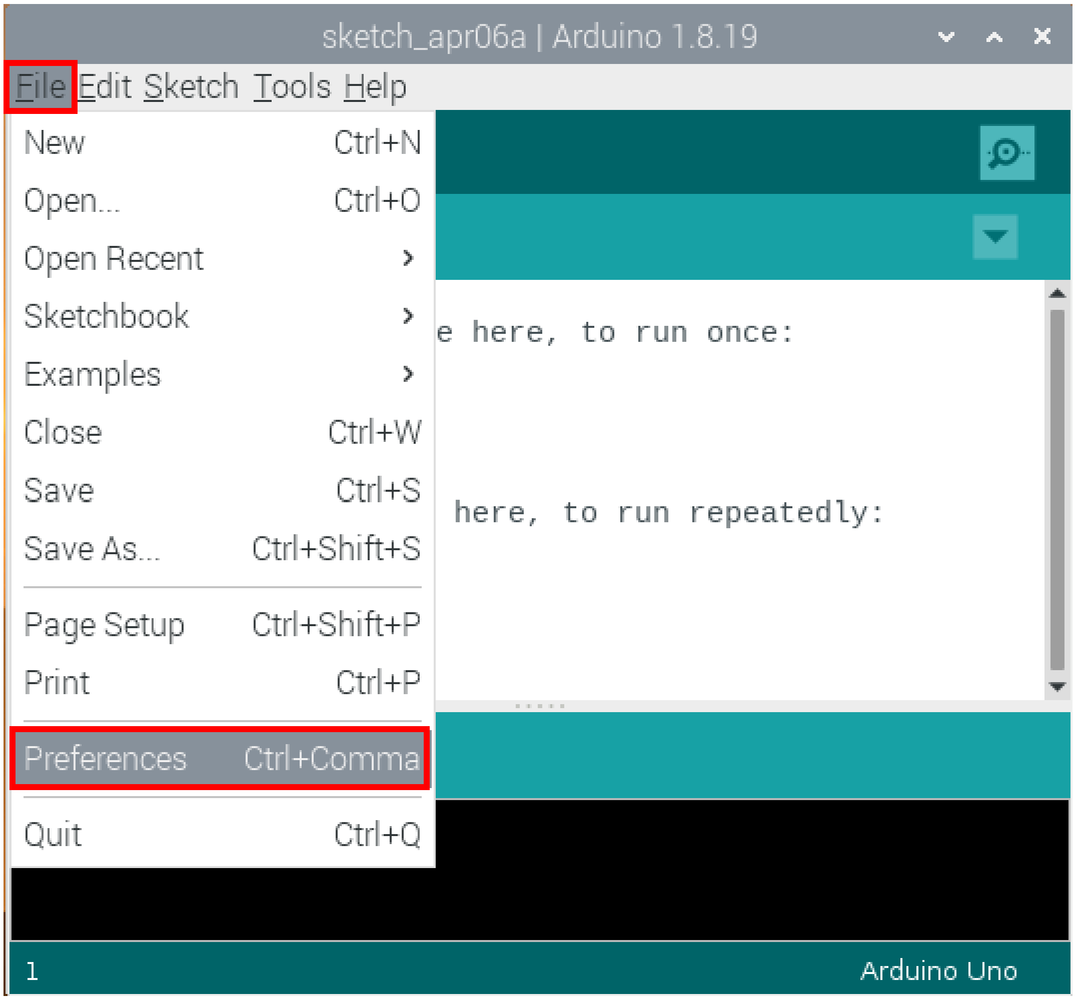
Note: You need to download Arduino IDE 1.8.5 or advanced version to install the ESP32.

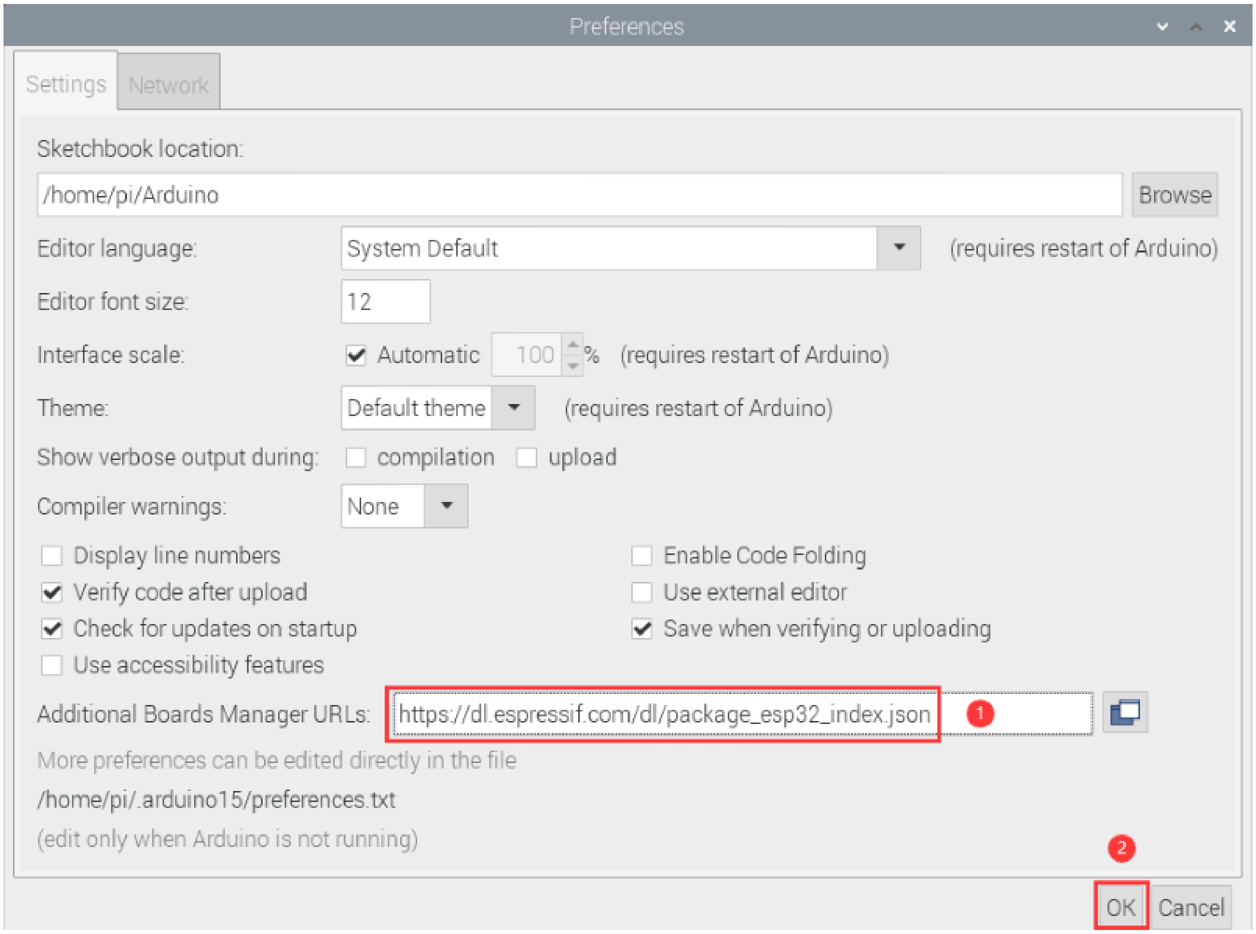
1) Click 

and 

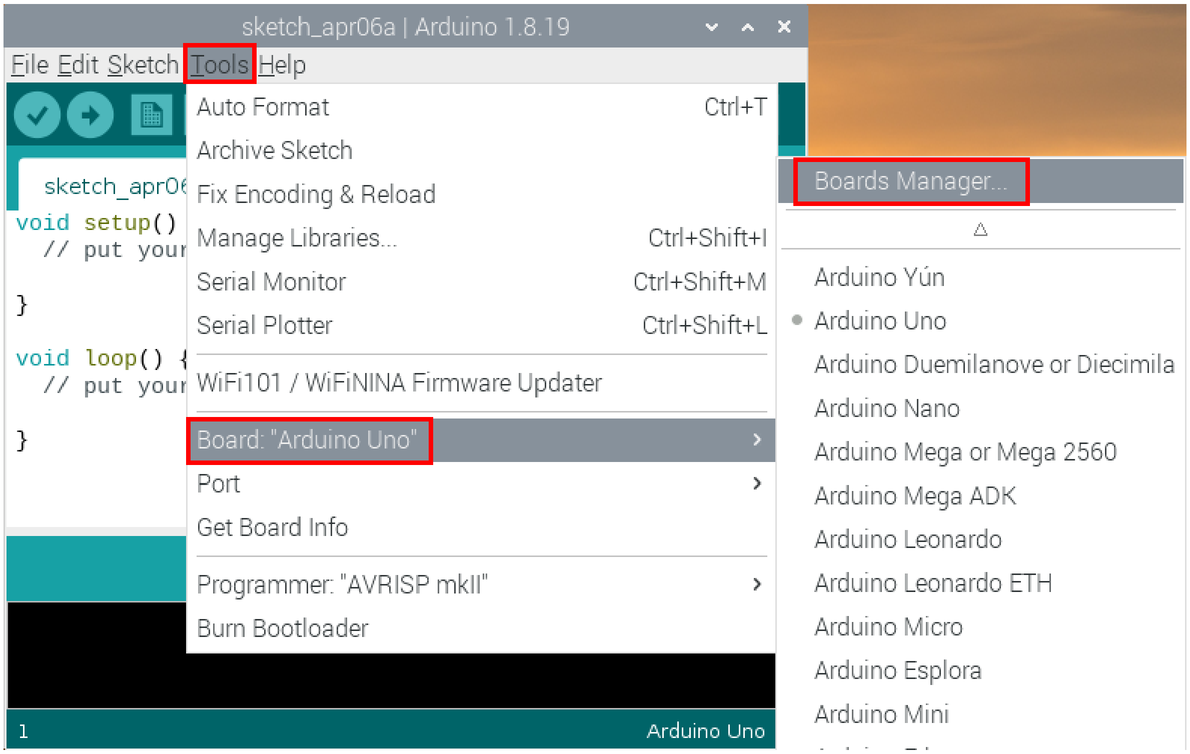
to open the Arduino IDE.

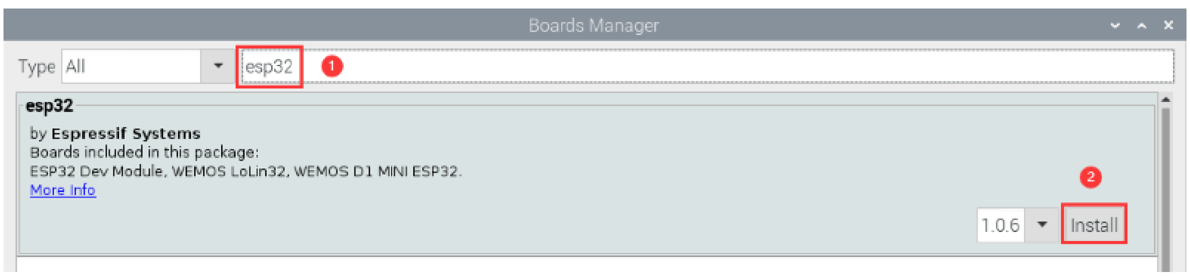
2) Click**"File**"→**"Preferences"**copy the website address https://dl.espressif.com/dl/package_esp32_index.json in the"**Additional Boards Manager URLs:**"and click"**OK**" to save the address.

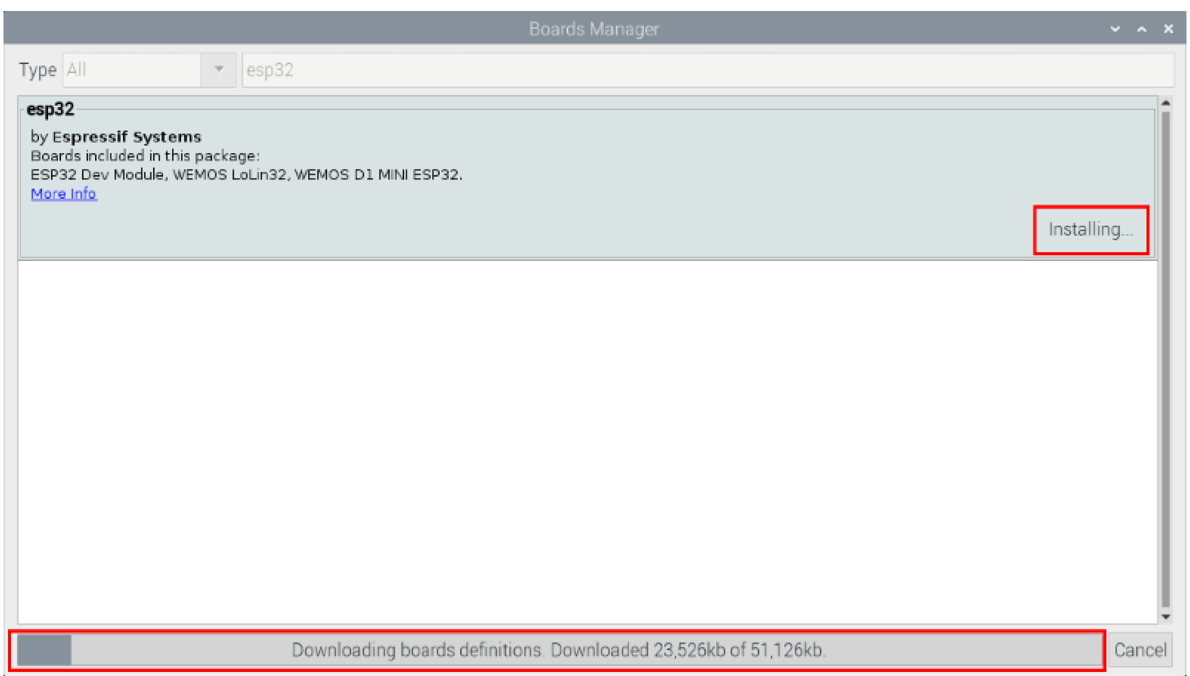

sketch_apr06a | Arduino 1.8.19
File Edit Sketch Tools Help
New Ctrl+N
Open... Ctrl+O
Open Recent
Sketchbook
Examples
Close Ctrl+W
Save Ctrl+S
Save As... Ctrl+Shift+S
Page Setup Ctrl+Shift+P
Print Ctrl+P
Preferences Ctrl+Comma
Quit Ctrl+Q
e here, to run once:
here, to run repeatedly:
1
Arduino Uno



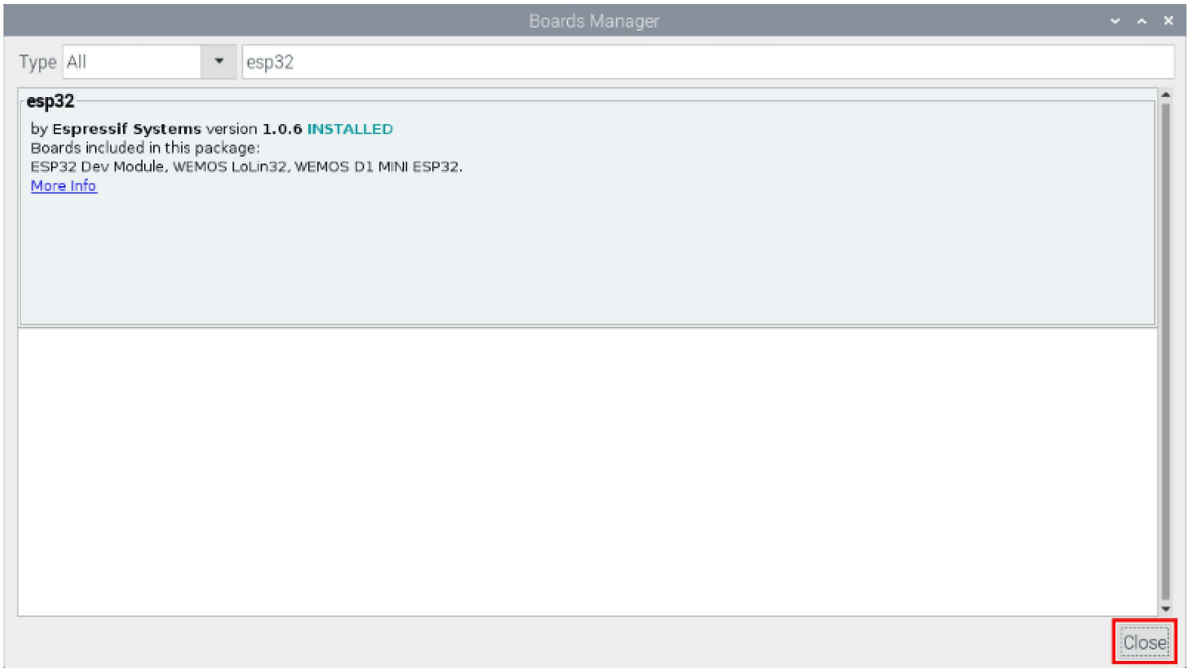
3. Click"**Tools**"→"**Board:**",then click "**Boards Manager…**"to enter "**Boards Manager**" page . Enter"**esp32**"as follows and select the latest version to Install. The installation package is not large, click"**Install**"to start the installation.


sketch_apr06a | Arduino 1.8.19
File Edit Sketch Tools Help
Auto Format Ctrl+T
Archive Sketch
Fix Encoding & Reload
Manage Libraries... Ctrl+Shift+I
Serial Monitor Ctrl+Shift+M
Serial Plotter Ctrl+Shift+L
WiFi101 / WiFiNINA Firmware Updater
Board: "Arduino Uno"
Port
Get Board Info
Programmer: "AVRISP mkII"
Burn Bootloader
Boards Manager...
Arduino Yún
Arduino Uno
Arduino Duemilanove or Diecimila
Arduino Nano
Arduino Mega or Mega 2560
Arduino Mega ADK
Arduino Leonardo
Arduino Leonardo ETH
Arduino Micro
Arduino Esplora
Arduino Mini
1
Arduino Uno


Boards Manager
Type All
esp32
1
esp32
by Espressif Systems
Boards included in this package:
ESP32 Dev Module, WEMOS LoLin32, WEMOS D1 MINI ESP32.
More Info
2
1.0.6
Install


Boards Manager
Type All
esp32
esp32
by Espressif Systems
Boards included in this package:
ESP32 Dev Module, WEMOS LoLin32, WEMOS D1 MINI ESP32.
More Info
Installing...
Downloading boards definitions. Downloaded 23,526kb of 51,126kb.
Cancel

4. After a while, the ESP32 installation package is installed. Click "**Close**" to Close the page.

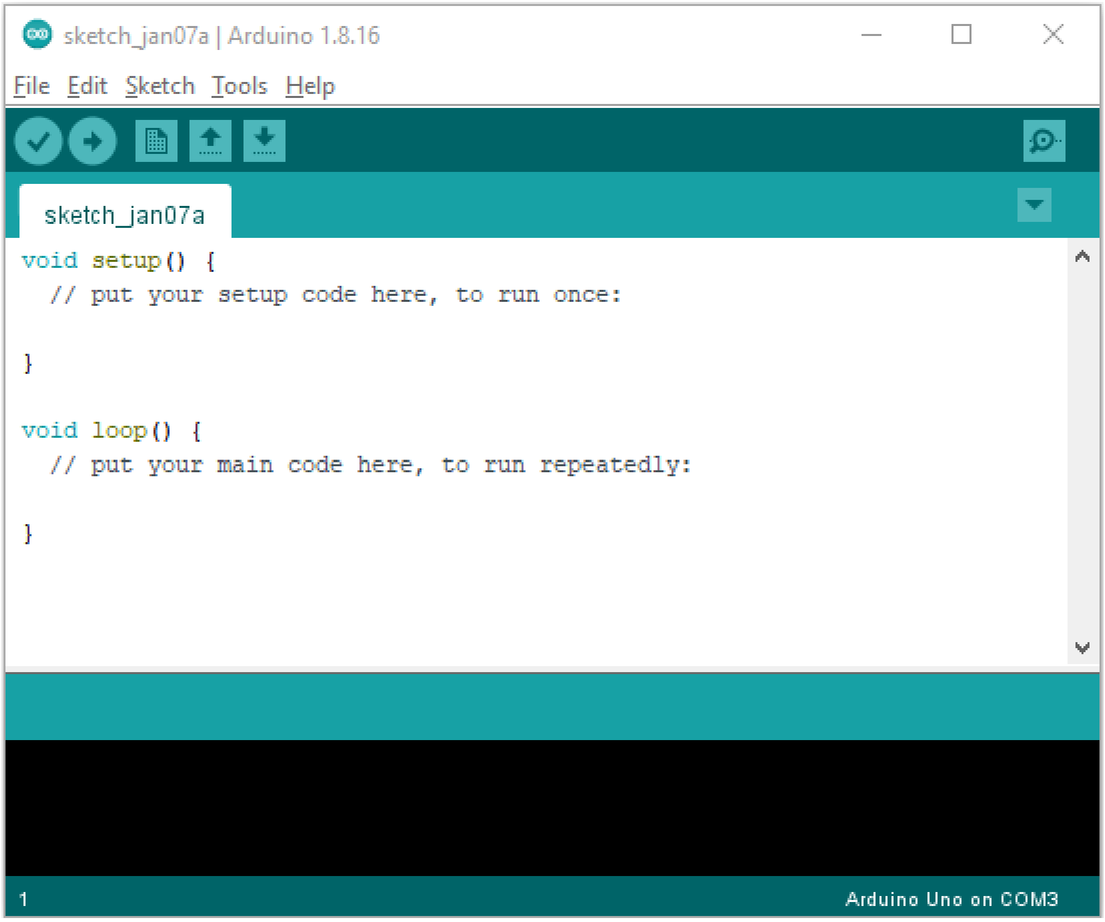


8.2.3 3. Arduino IDE Settings and toolbars:

1Click 

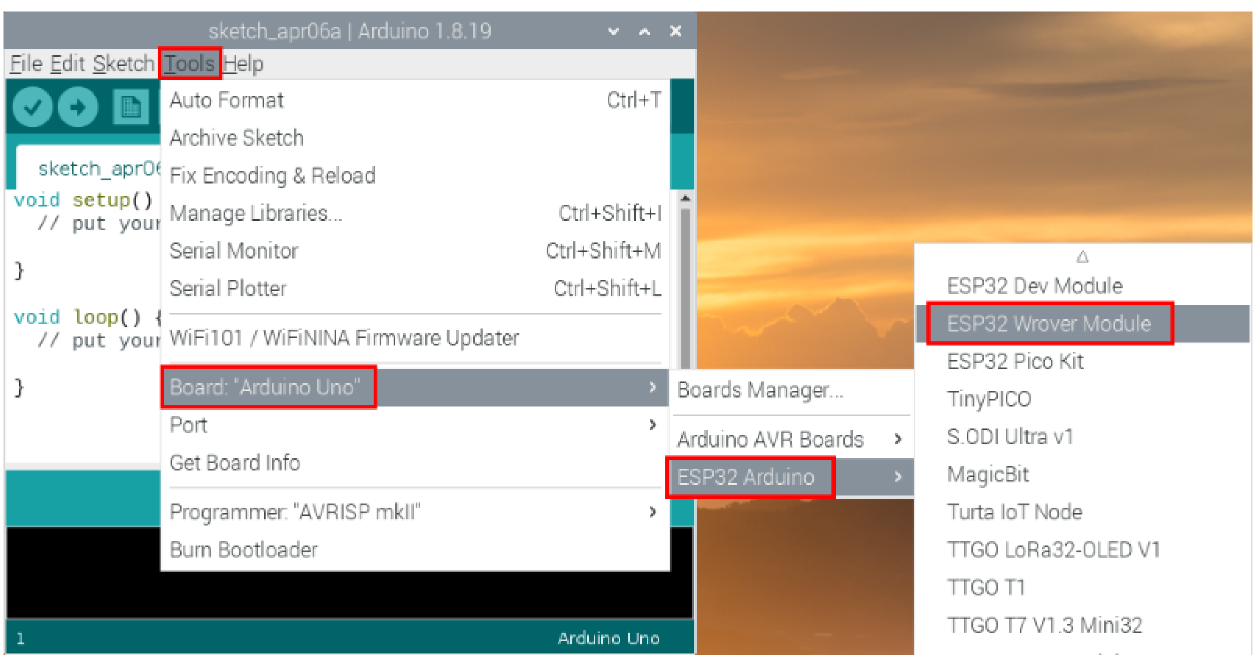
and 

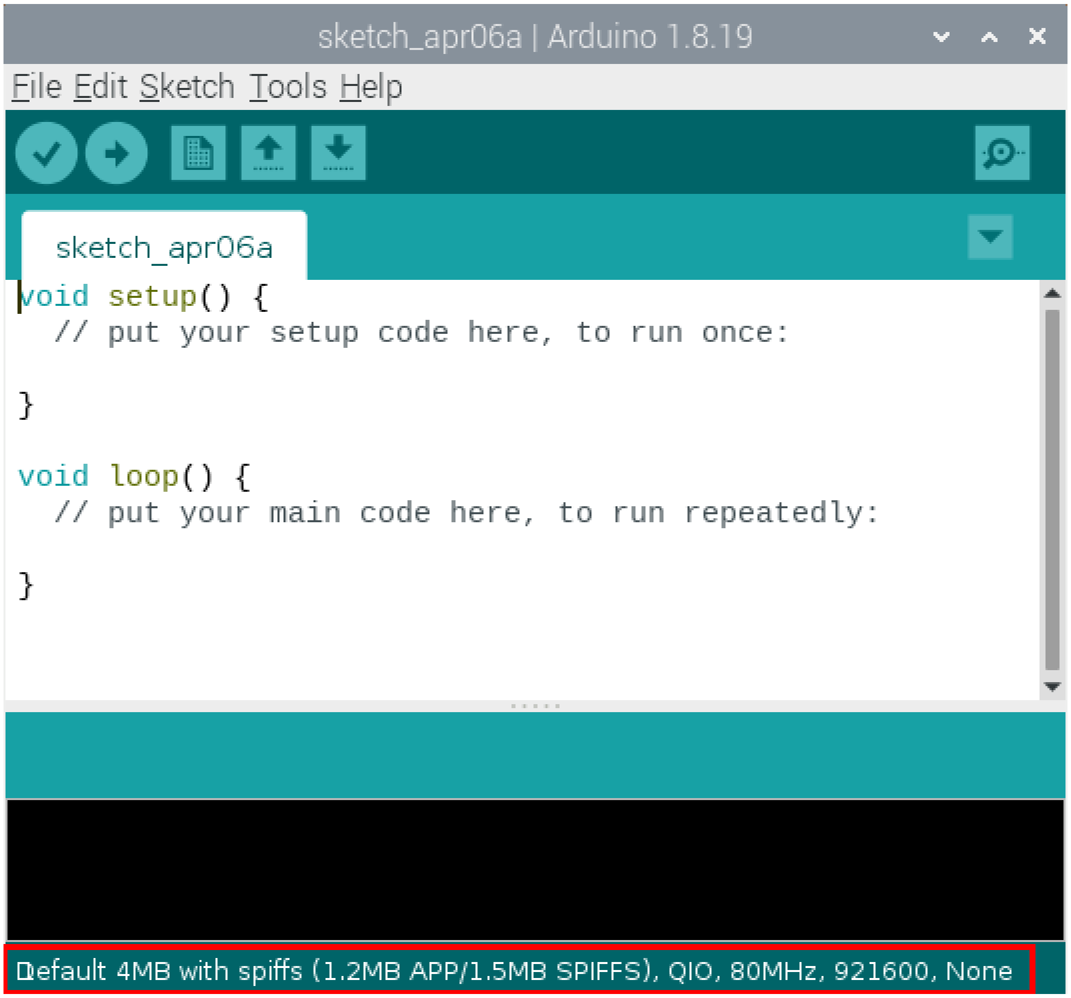
to open the Arduino IDE.



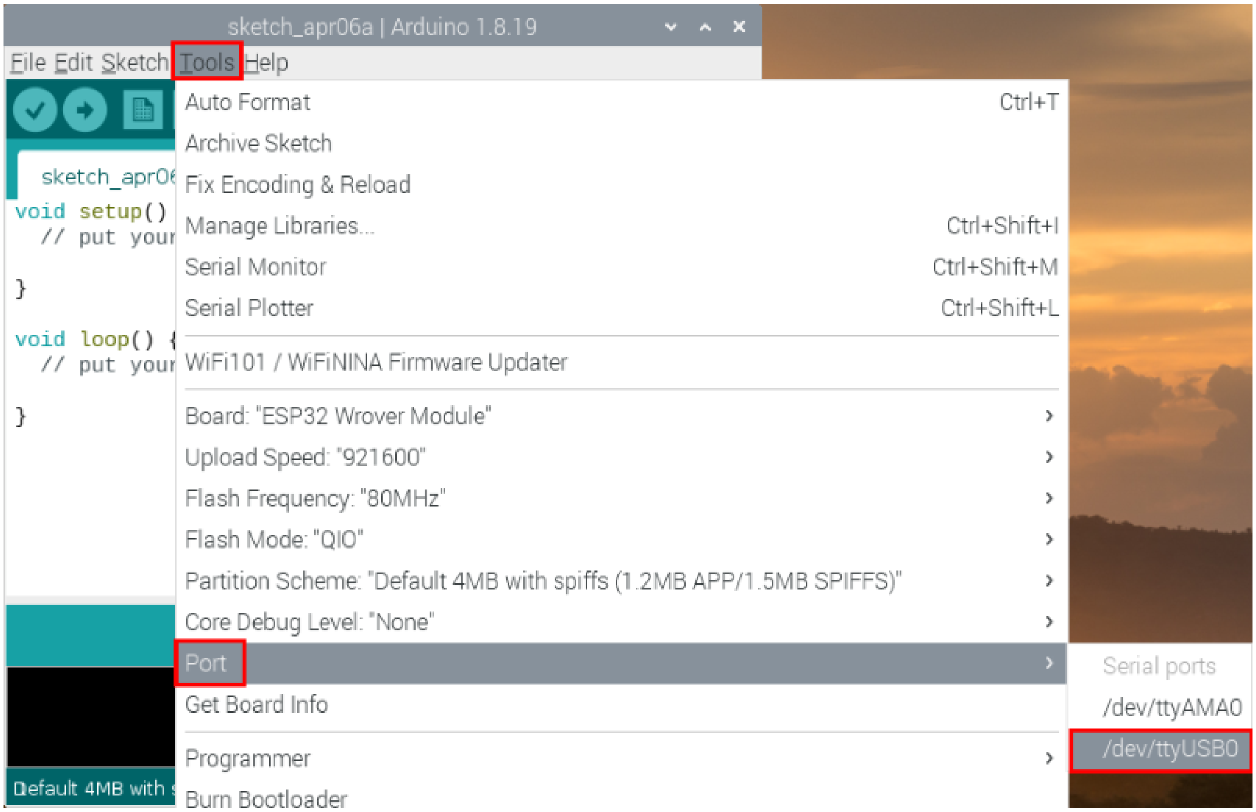
2When downloading the code to the board, you must select the correct name of Arduino board that matches the board connected to the Raspberry Pi,click“**Tools**”→“**Board:**”. As shown below ;

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

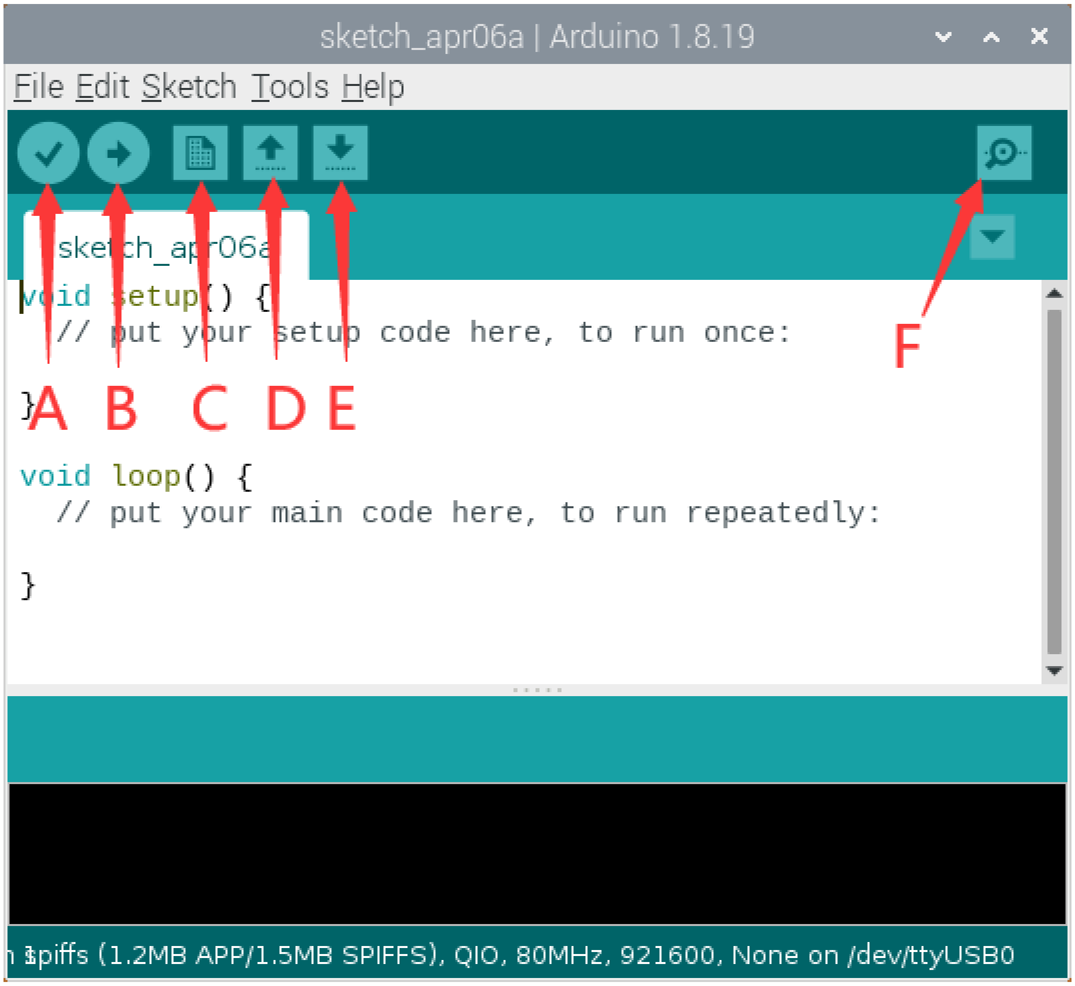




Then select the correct COM port (After connecting the ESP32 mainboard to the Raspberry Pi via USB cable, you can see the corresponding COM port).


sketch_apr06a | Arduino 1.8.19
File Edit Sketch Tools Help
sketch_apr06
void setup() {
// put your
}
void loop() {
// put your
}
Default 4MB with s
Auto Format Ctrl+T
Archive Sketch
Fix Encoding & Reload
Manage Libraries... Ctrl+Shift+I
Serial Monitor Ctrl+Shift+M
Serial Plotter Ctrl+Shift+L
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
Get Board Info
Programmer
Burn Bootloader
Serial ports
/dev/ttyAMA0
/dev/ttyUSB0

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.

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.

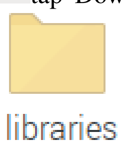
For example, the built-in LiquidCrystal library helps talk to LCD displays. There are hundreds of additional libraries available on the Internet for download.

The built-in libraries and some of these additional libraries are listed in the reference. (https://www.arduino.cc/en/Reference/Libraries)

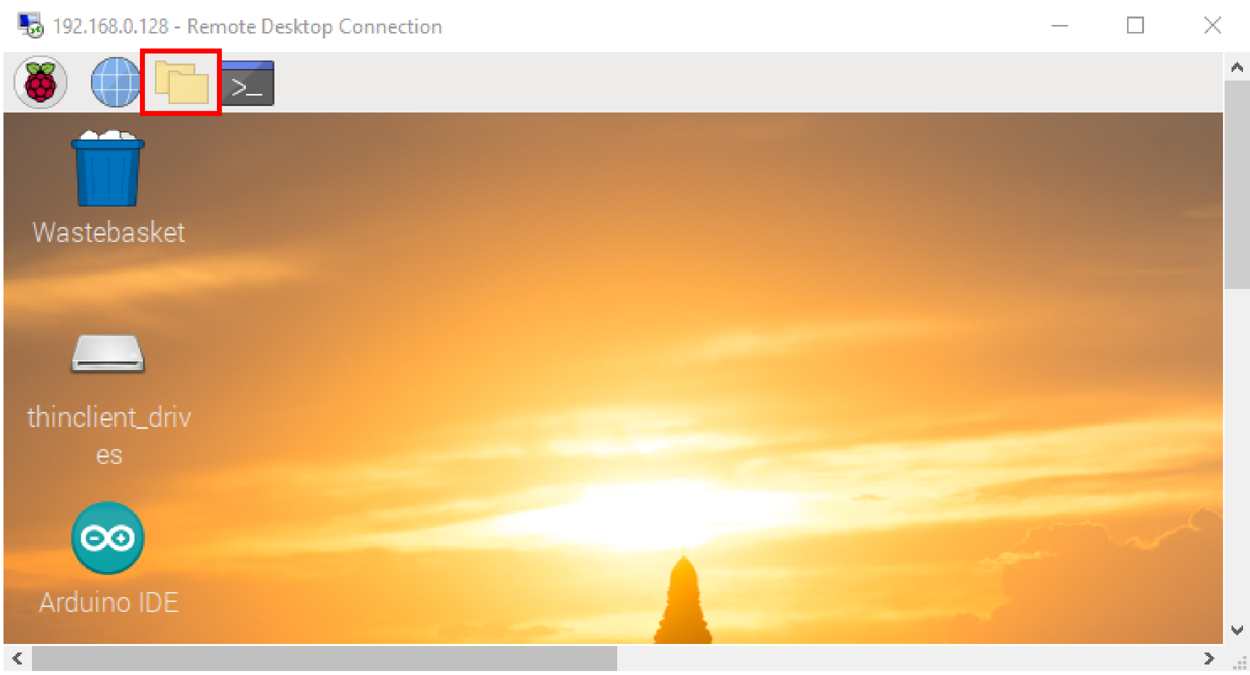
Here we will introduce the most simple way to add libraries .

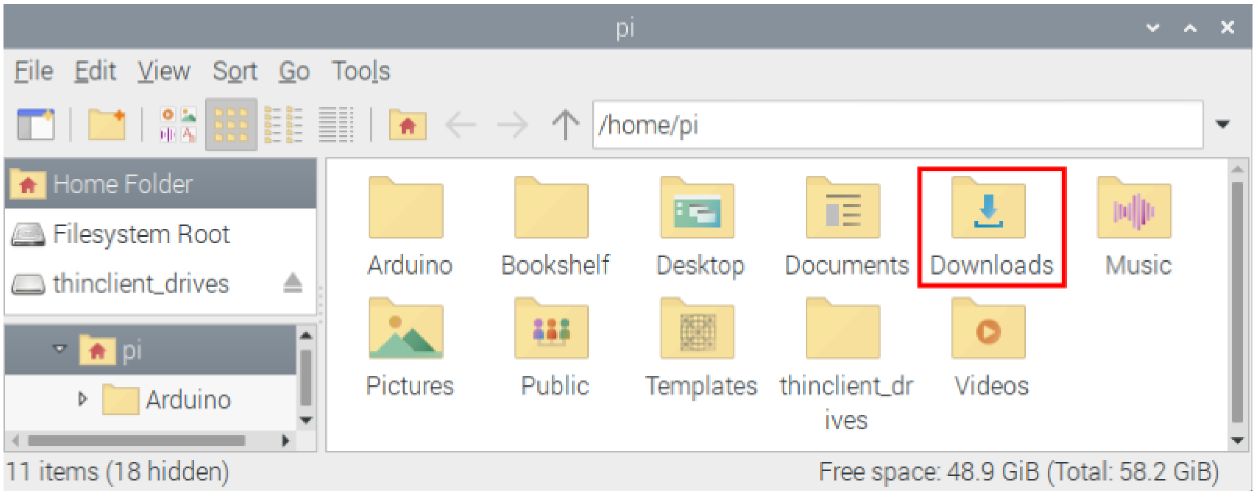
Step 1: Click  tap“Downloads”file 

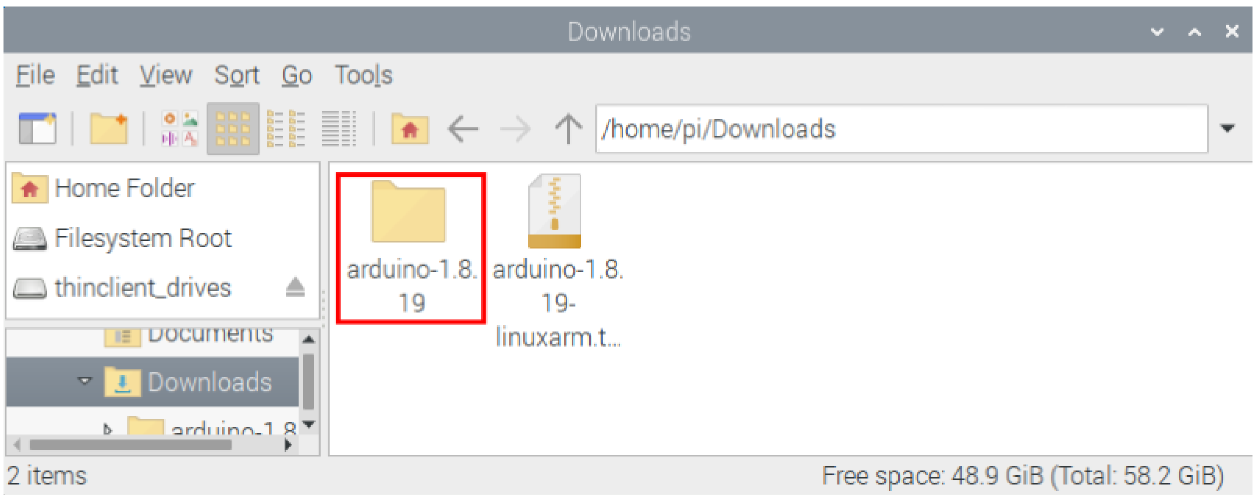
and click“arduino-1.8.19”file 

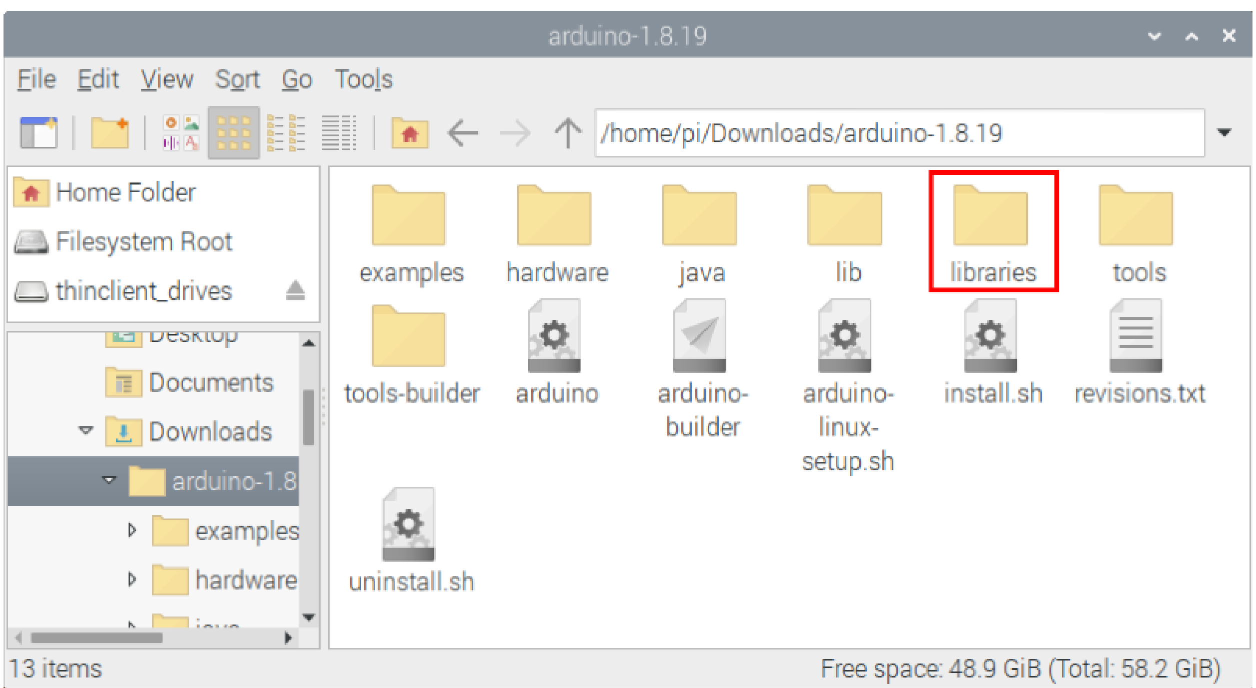
then find and click“libraries” file 

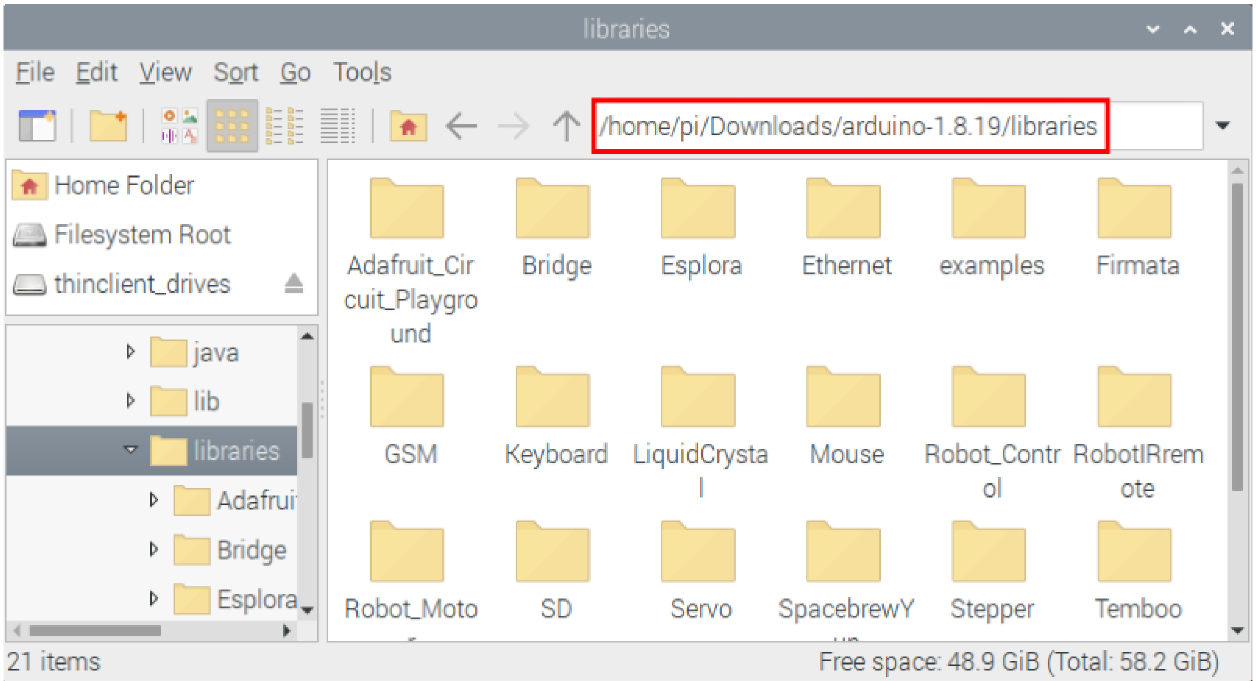
from the “arduino-1.8.19”file.




pi
File Edit View Sort Go Tools
/home/pi
Home Folder
Filesystem Root
thinclient_drives
pi
Arduino
Arduino
Bookshelf
Desktop
Documents
Downloads
Music
Pictures
Public
Templates
thinclient_drives
Videos
11 items (18 hidden)
Free space: 48.9 GiB (Total: 58.2 GiB)

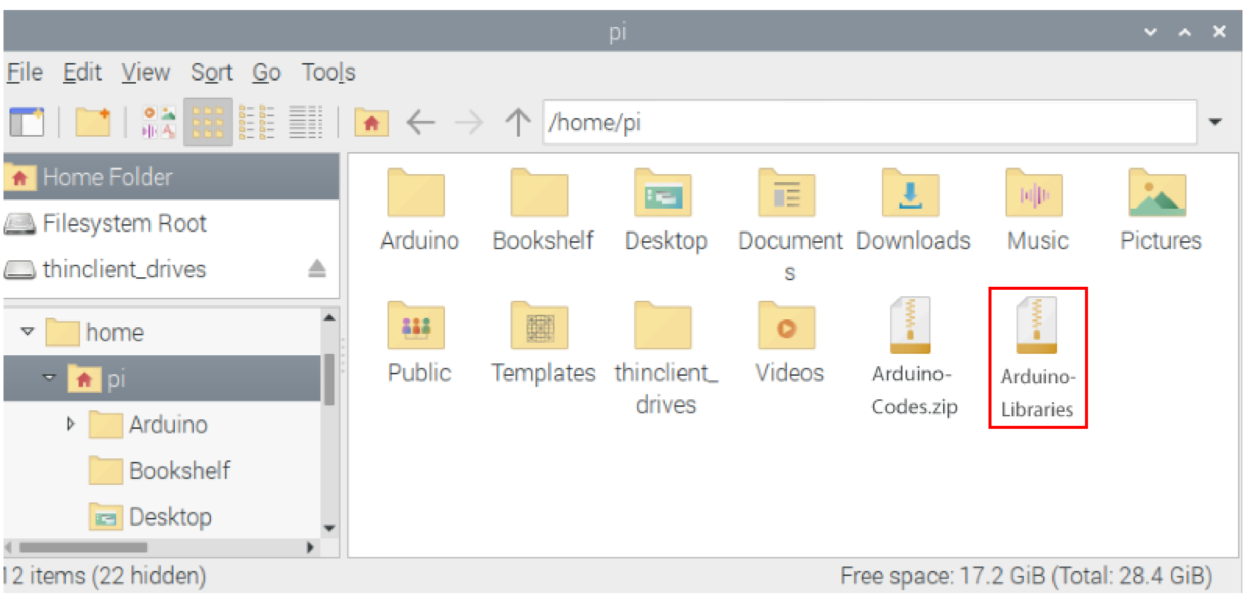

Downloads
File Edit View Sort Go Tools
/home/pi/Downloads
Home Folder
Filesystem Root
thinclient_drives
Downloads
arduino-1.8.19
arduino-1.8.19-linuxarm.t...
2 items
Free space: 48.9 GiB (Total: 58.2 GiB)

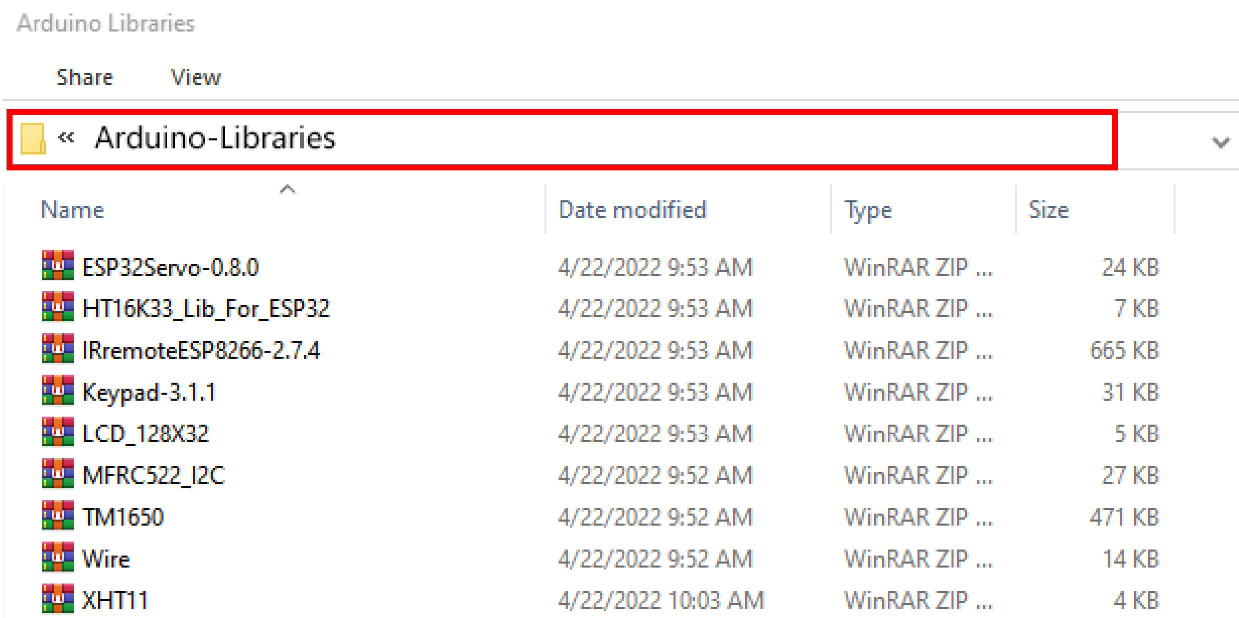

arduino-1.8.19
File Edit View Sort Go Tools
/home/pi/Downloads/arduino-1.8.19
Home Folder
Filesystem Root
thinclient_drives
Documents
Downloads
arduino-1.8
examples
hardware
examples
hardware
java
lib
libraries
tools
tools-builder
arduino
arduino-builder
arduino-linux-setup.sh
install.sh
revisions.txt
uninstall.sh
13 items
Free space: 48.9 GiB (Total: 58.2 GiB)

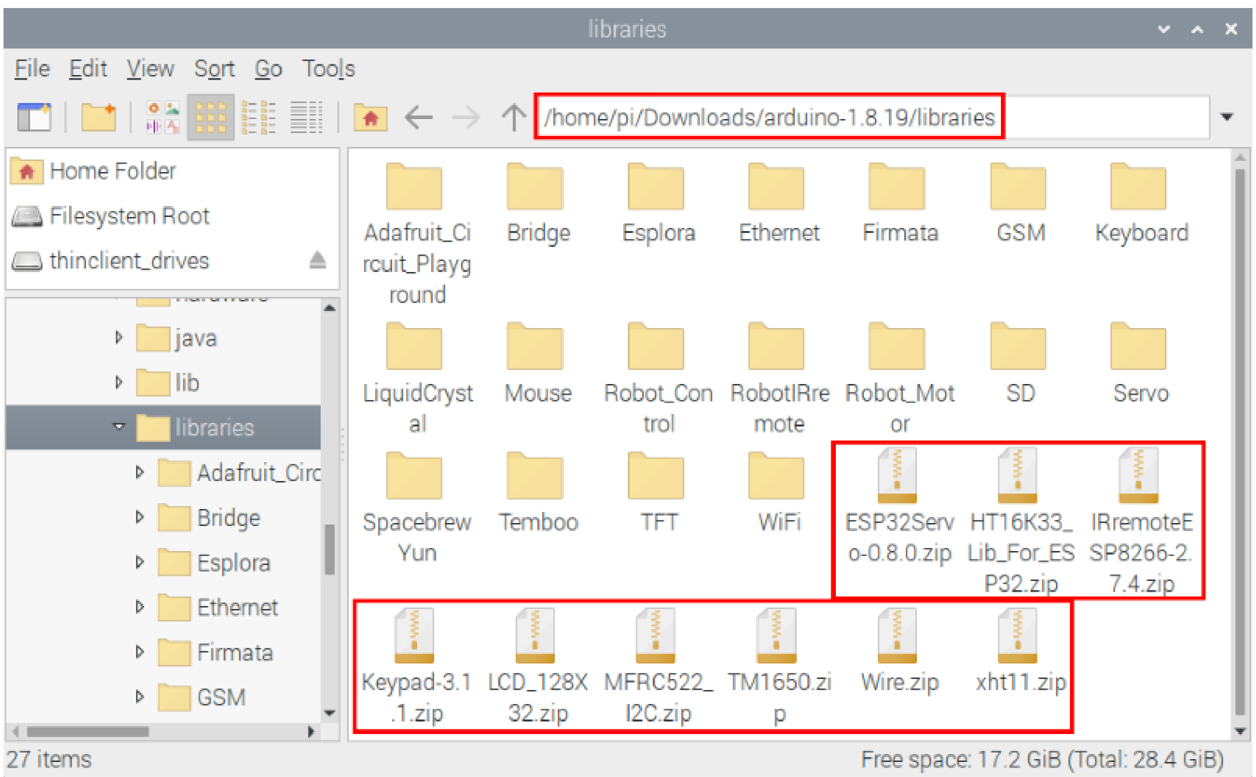


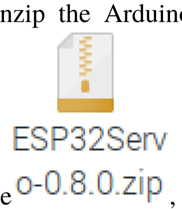
Step 2 : Copy and paste the Arduino C library ZIP file (the default is ZIP file) from the provided Arduino Libraries folder into the libraries file opened in the first stepthe route is/home/pi/Downloads/arduino-1.8.19/libraries.

Click on the link to download the Libraries file: Download Libraries file

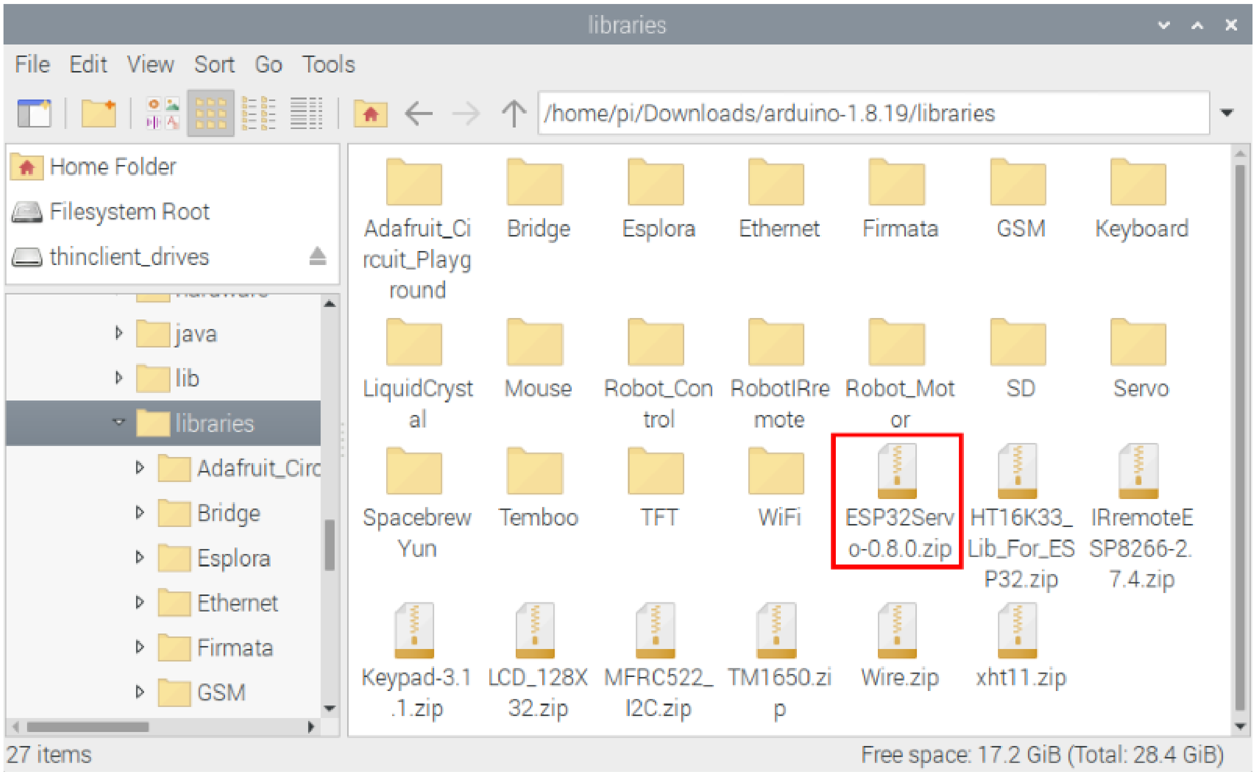


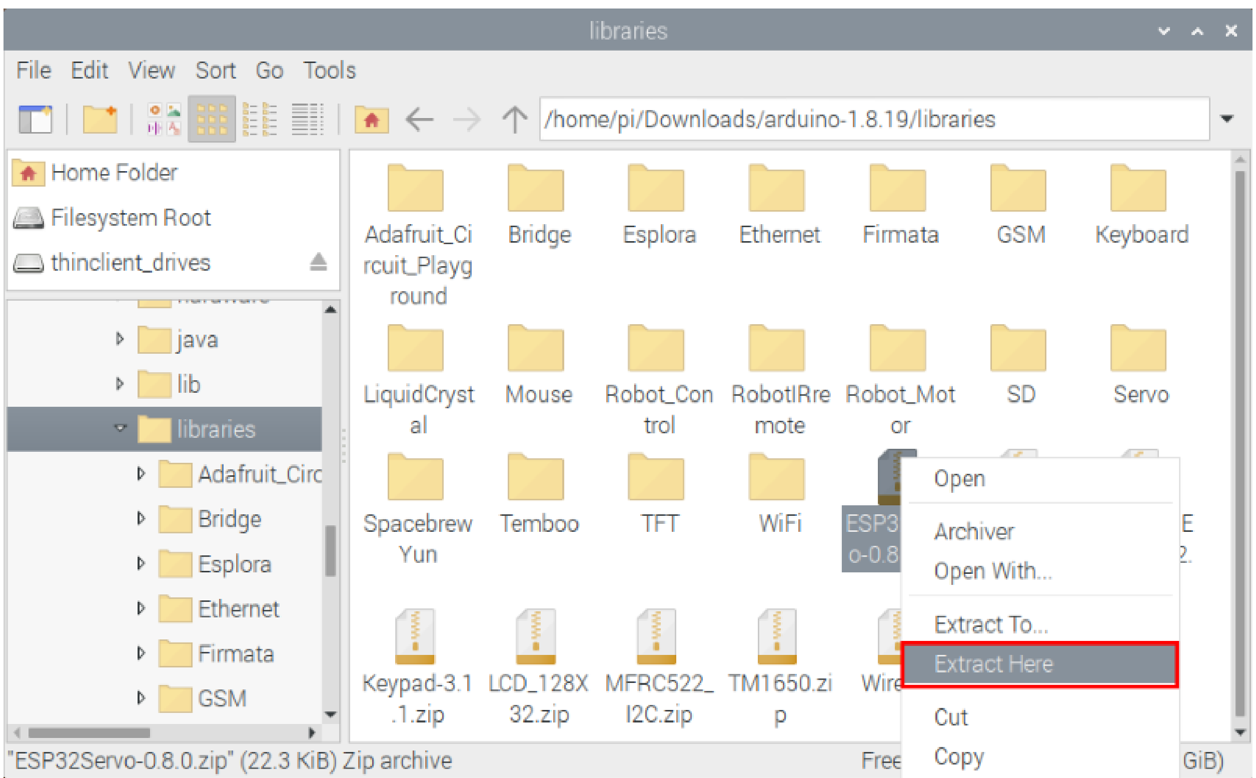


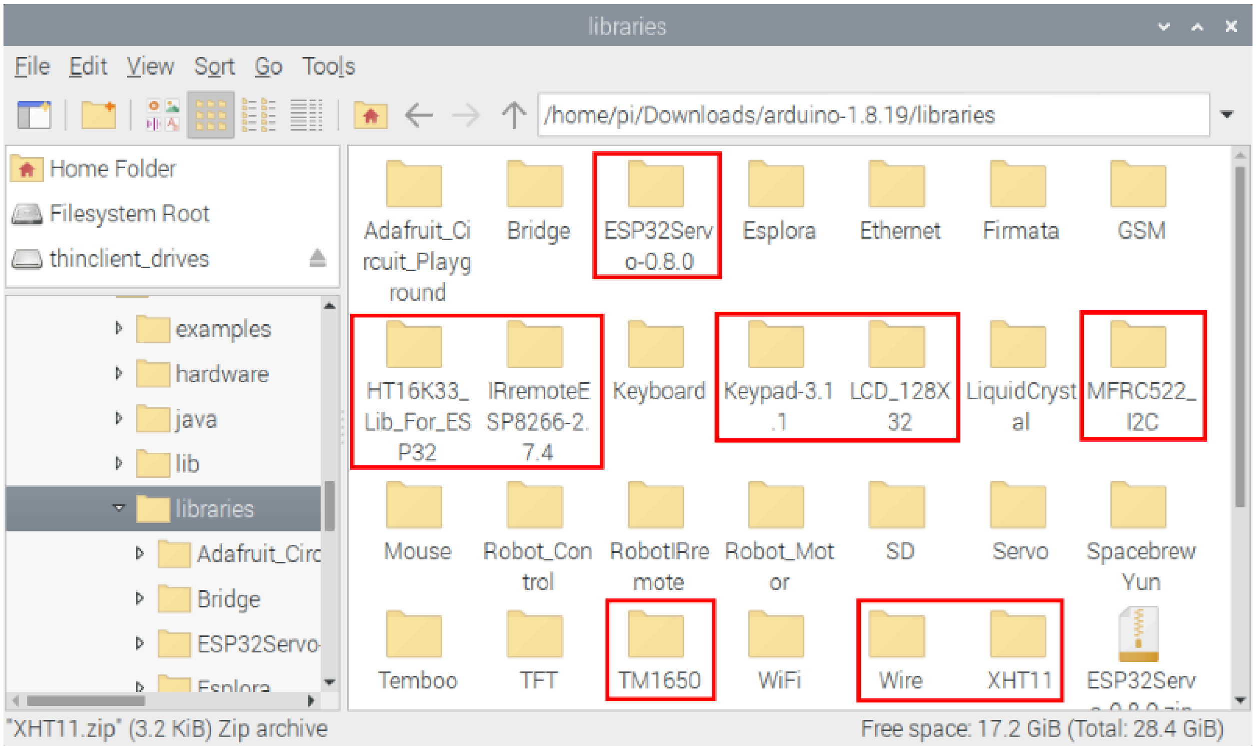


Step 3: Unzip the Arduino C package in the libraries folderfor exampleright-click “ESP32Servo-0.8.0.zip”file 

, 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.


libraries
File Edit View Sort Go Tools
/home/pi/Downloads/arduino-1.8.19/libraries
Home Folder
Filesystem Root
thinclient_drives
java
lib
libraries
Adafruit_Circ
Bridge
Esplora
Ethernet
Firmata
GSM
Adafruit_Circuit_Playground
Bridge
Esplora
Ethernet
Firmata
GSM
Keyboard
LiquidCrystal
Mouse
Robot_Control
RobotIRremote
Robot_Motor
SD
Servo
SpacebrewYun
Temboo
TFT
WiFi
ESP32Servo-0.8.0.zip
HT16K33_Lib_For_ESP32.zip
IRremoteESP8266-2.7.4.zip
Keypad-3.1.1.zip
LCD_128X32.zip
MFRC522_I2C.zip
TM1650.zip
Wire.zip
xht11.zip
27 items
Free space: 17.2 GiB (Total: 28.4 GiB)


libraries
File Edit View Sort Go Tools
/home/pi/Downloads/arduino-1.8.19/libraries
Home Folder
Filesystem Root
thinclient_drives
Open
Archiver
Open With...
Extract To...
Extract Here
Cut
Copy
"ESP32Servo-0.8.0.zip" (22.3 KiB) Zip archive


libraries
File Edit View Sort Go Tools
/home/pi/Downloads/arduino-1.8.19/libraries
Home Folder
Filesystem Root
thinclient_drives
examples
hardware
java
lib
libraries
Adafruit_Circ
Bridge
ESP32Servo
Adafruit_Circuit_Playground
Bridge
ESP32Servo-0.8.0
Esplora
Ethernet
Firmata
GSM
HT16K33_Lib_For_ESP32
IRremoteESP8266-2.7.4
Keyboard
Keypad-3.1.1
LCD_128X32
LiquidCrystal
MFRC522_I2C
Mouse
Robot_Control
RobotIRremote
Robot_Motor
SD
Servo
SpacebrewYun
Temboo
TFT
TM1650
WiFi
Wire
XHT11
ESP32Serv
"XHT11.zip" (3.2 KiB) Zip archive
Free space: 17.2 GiB (Total: 28.4 GiB)

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

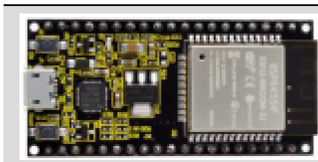
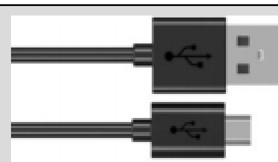
Click on the link to download the Libraries file: Download Libraries file

9.2 Project 01: Hello World

9.2.1 1.Introduction

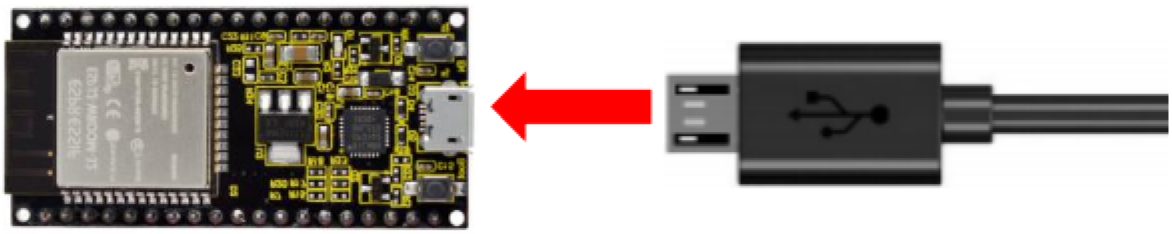
For ESP32 beginners, we'll start with some simple things. In this project, you just need an ESP32 mainboard, USB cable and Raspberry Pi to complete "Hello World!" Project. It is not only a communication test for ESP32 mainboard and Raspberry Pi, but also a primary project for ESP32.

9.2.2 2.Components

	
ESP32*1	USB cable*1

9.2.3 3.Components

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



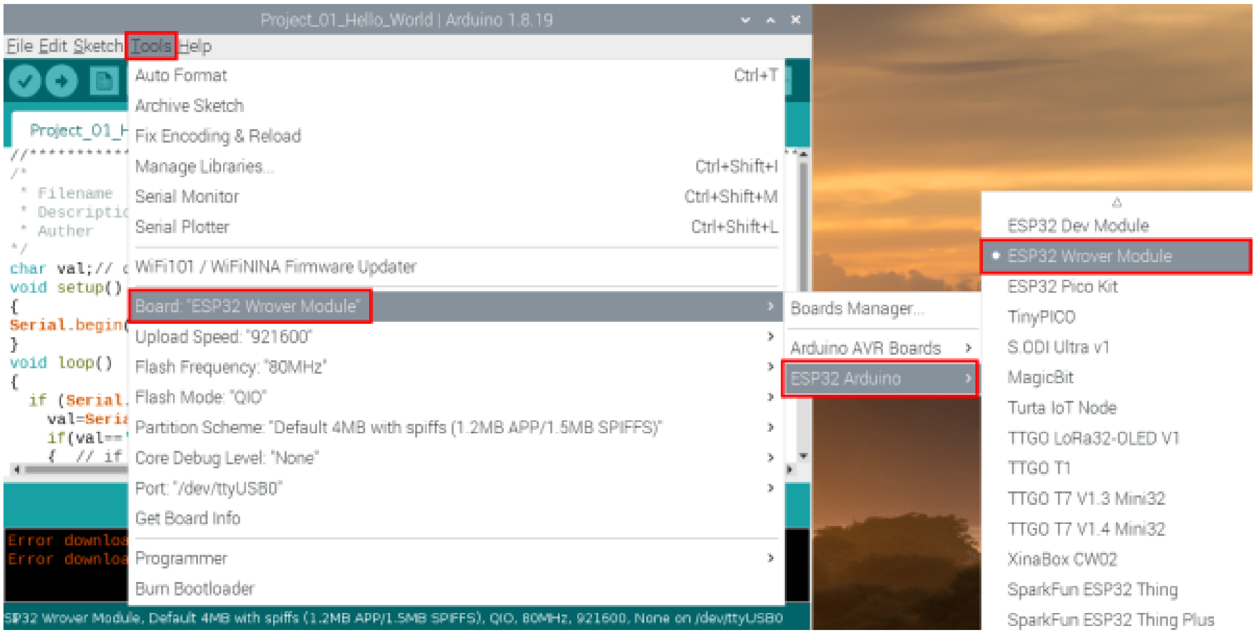
9.2.4 4.Project code

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

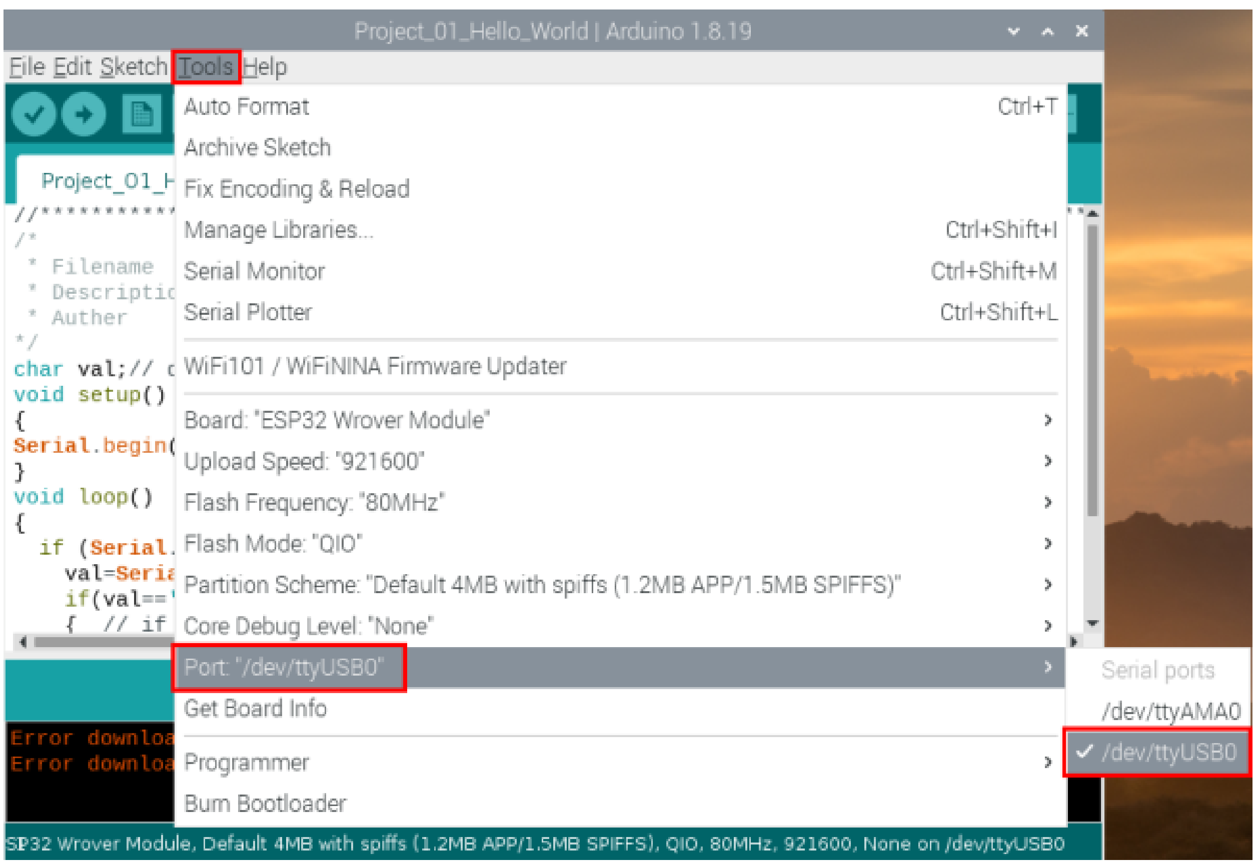
The code used in this project is saved in folder(path :) **"Arduino-Codes\Project 01Hello World\Project_01_Hello_World"**.

```
//**********************************************************************************
/*
 * 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".



Select the serial port.



Click media/b0d41283bf5ae66d2d5ab45db15331ba.png to download the code to ESP32.

Note: If uploading the code fails, you can press the Boot button on ESP32 after click media/d09c4a31563f04a42d451e7bc1a5fb8, and release the Boot button media/dc77bfcf5851c8f43aab6cbe7cec7920.png after the percentage of uploading progress appears, as shown below:

media/157ee2e7687559d9812d24edec758150.png

The Project code is uploaded successfully

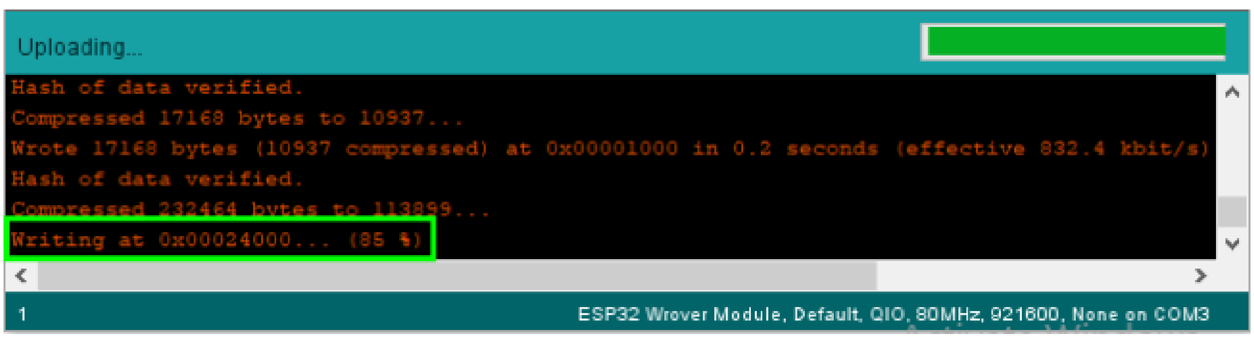
9.2.5 5.Project result

After the project code is uploaded successfully, power up with a USB cable and click the icon media/2f6bca56f724e45a855335cb53a 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/1fd21fafd84d2b529931a89d21a03d6a.png

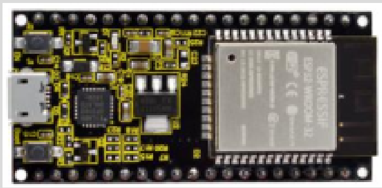
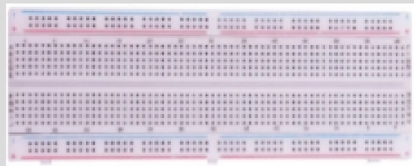
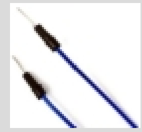
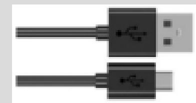
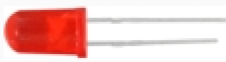


9.3 Project 02: Turn On LED

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

			
ESP32*1	Breadboard*1	Jumper Wire*2	USB Cable*1
			
Red LED*1	220 Resistor*1		

9.3.3 3.Component knowledge

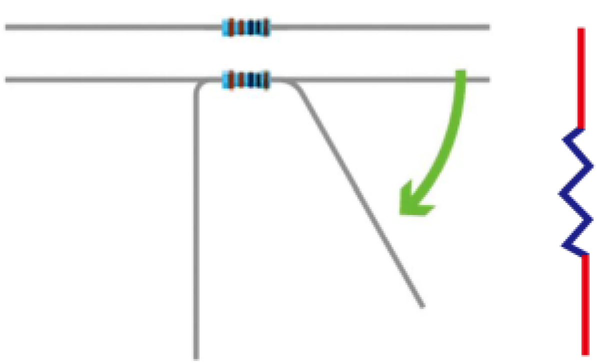
1LED:

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.

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® 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

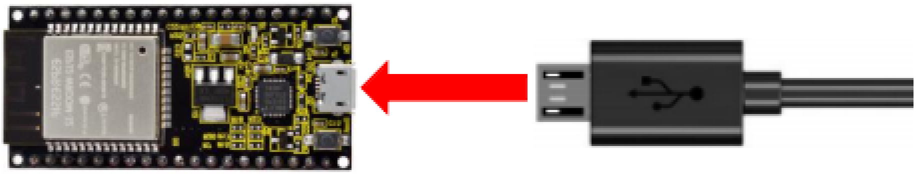
When we connect DIP (Dual In-line Packages) components, such as integrated circuits, microcontrollers, chips and so on, we can see that a groove in the middle isolates the middle part, so the top and bottom of the groove is not connected. DIP components can be connected as shown in the following diagram:

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/07359975 93c8858ad6441d8e9867206f.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

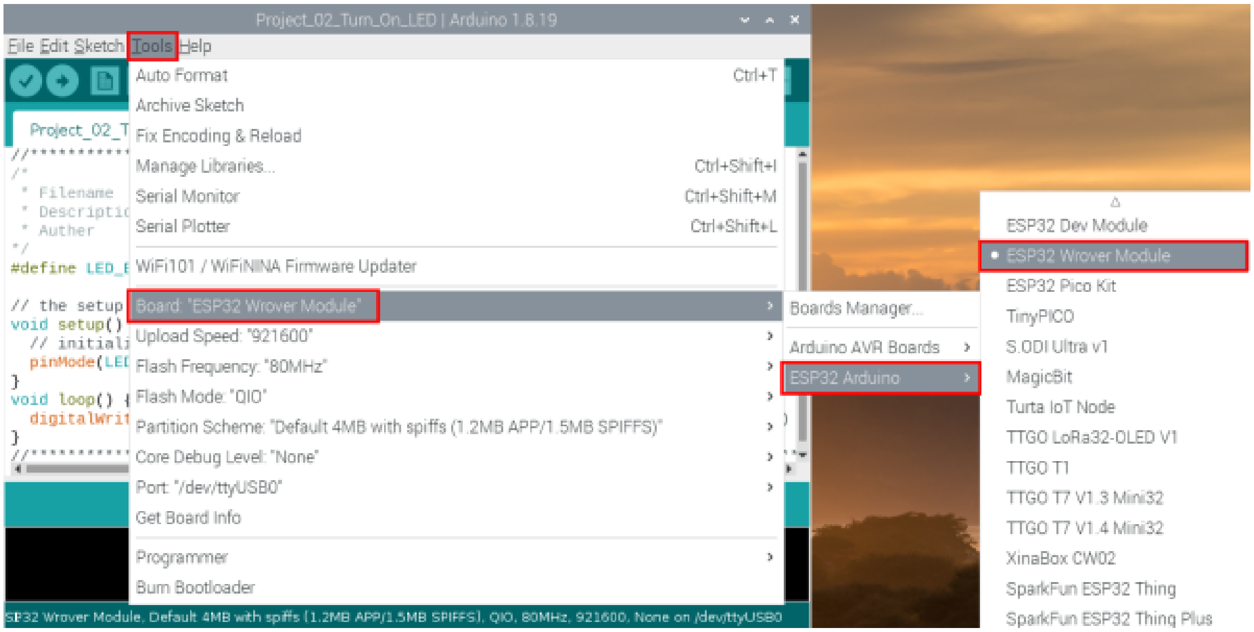
The code used in this project is saved in folder(path:) **"Arduino-Codes\Project 02Turn On LED\Project_02_Turn_On_LED"**.

```
//**************************************************************************
/*
 * 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".

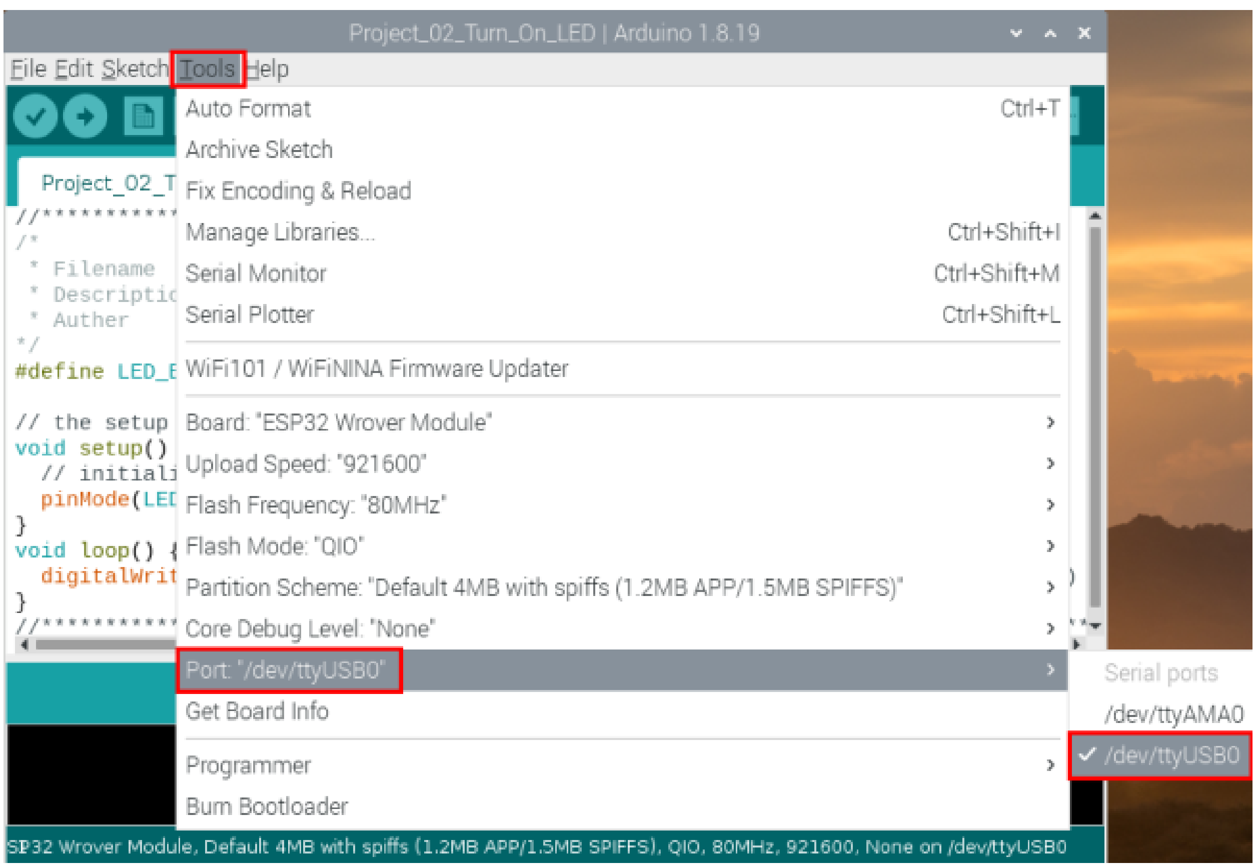
media/fe73bbd726dbe38bb24f9c0d735c90ec.png

Select the serial port.



media/b0d41283bf5ae66d2d5ab45db15331ba.png

Click to download the code to ESP32.



Note: If uploading the code fails, you can press the Boot button on ESP32 after click media/d09c4a31563f04a42d451e7bc1a5fb8, and release the Boot button media/dc77bfcf5851c8f43aab6cbe7cec7920.png after the percentage of uploading progress appears, as shown below:

media/157ee2e7687559d9812d24edec758150.png

The Project code is uploaded successfully

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.

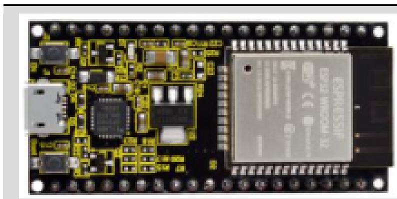
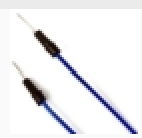
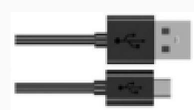
media/77dec960e108229b6d97b4af9a2db902.png

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		
			
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)
  delay(500);                       // wait for 0.5s
  digitalWrite(PIN_LED, LOW);    // turn the LED off by making the voltage LOW
  delay(500);                       // wait for 0.5s
```

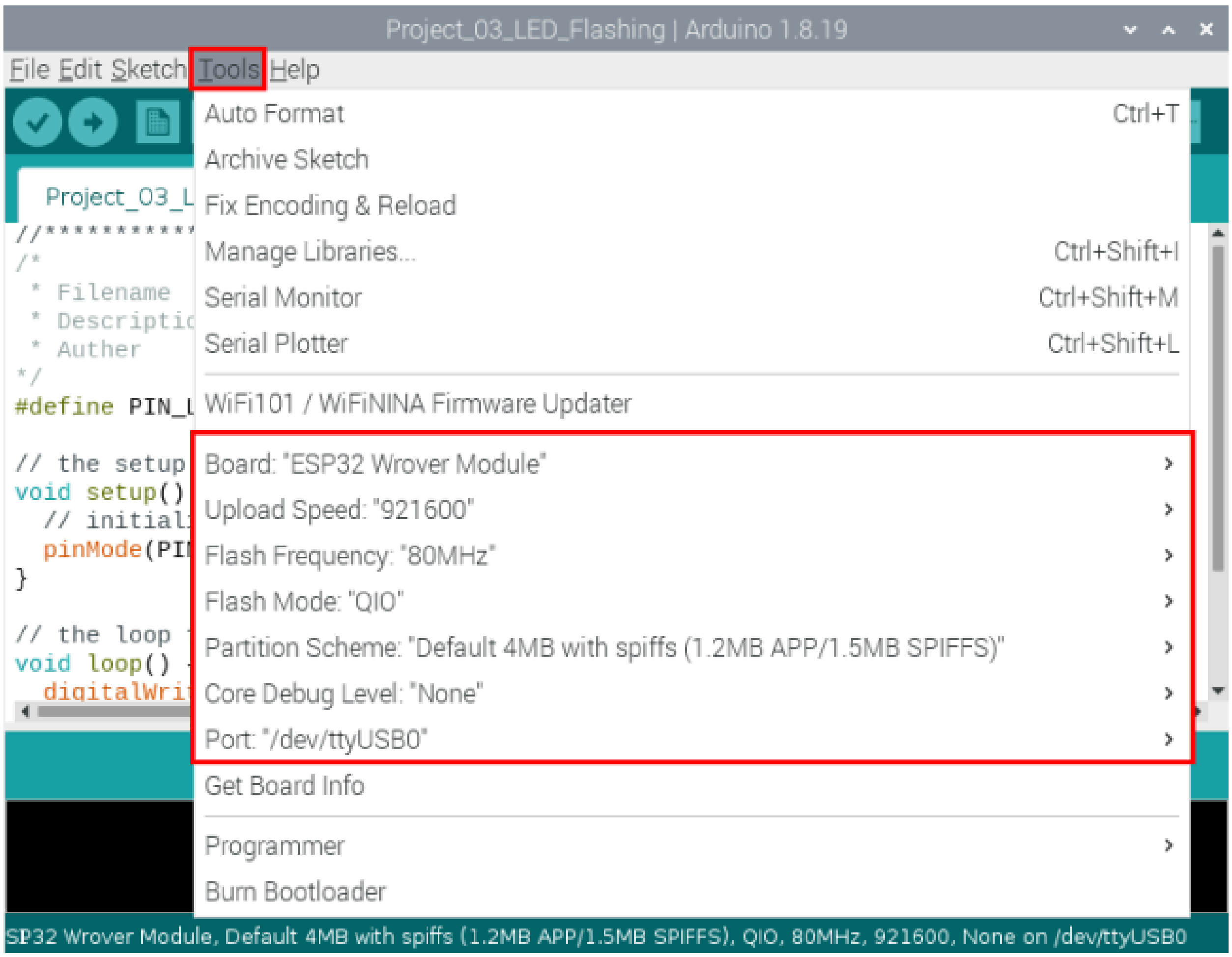
(continues on next page)

(continued from previous page)

```
}
//*************************************************************************************
```

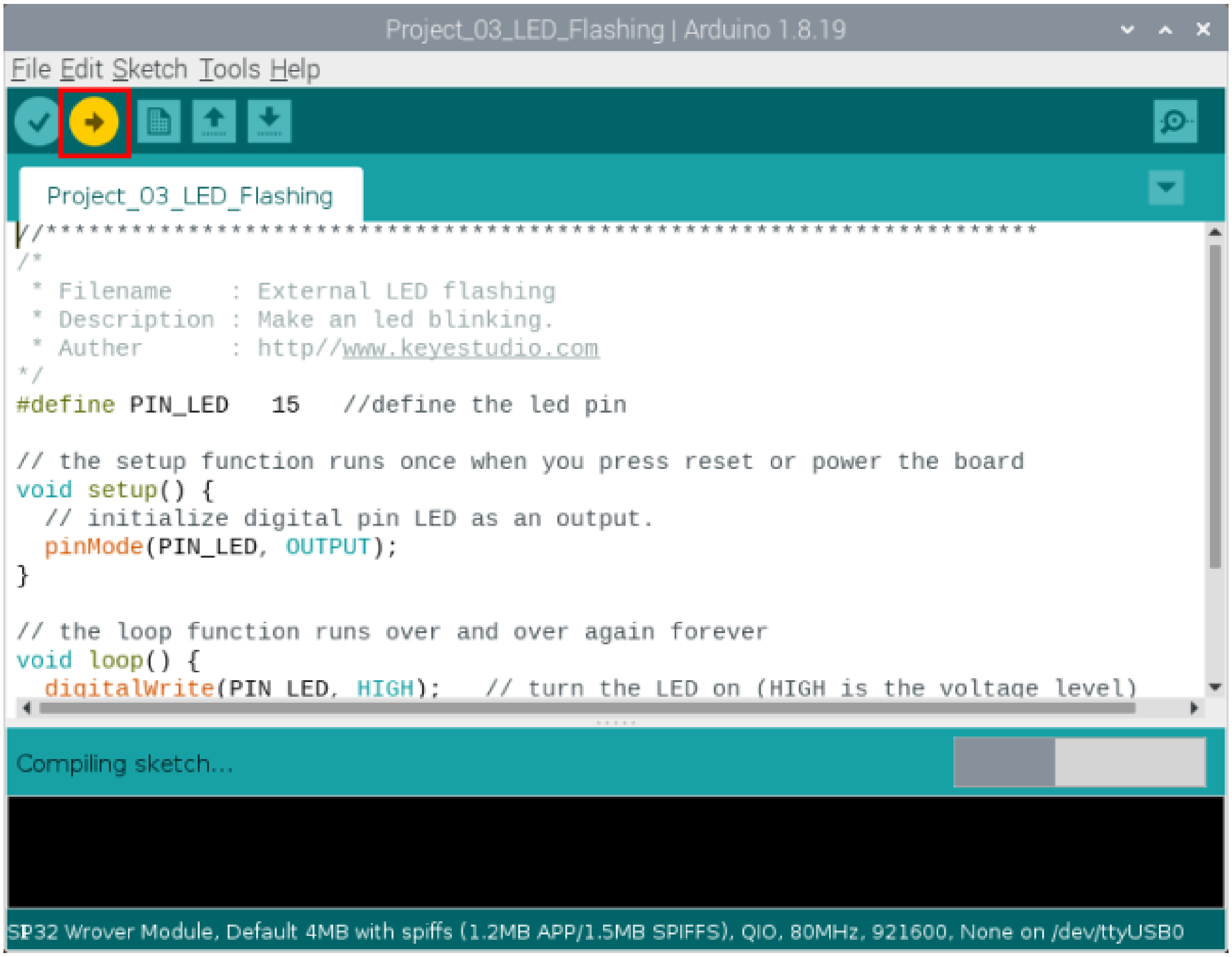
Before uploading Project Code to ESP32, please check the configuration of Arduino IDE.

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



media/b0d41283bf5ae66d2d5ab45db15331ba.png

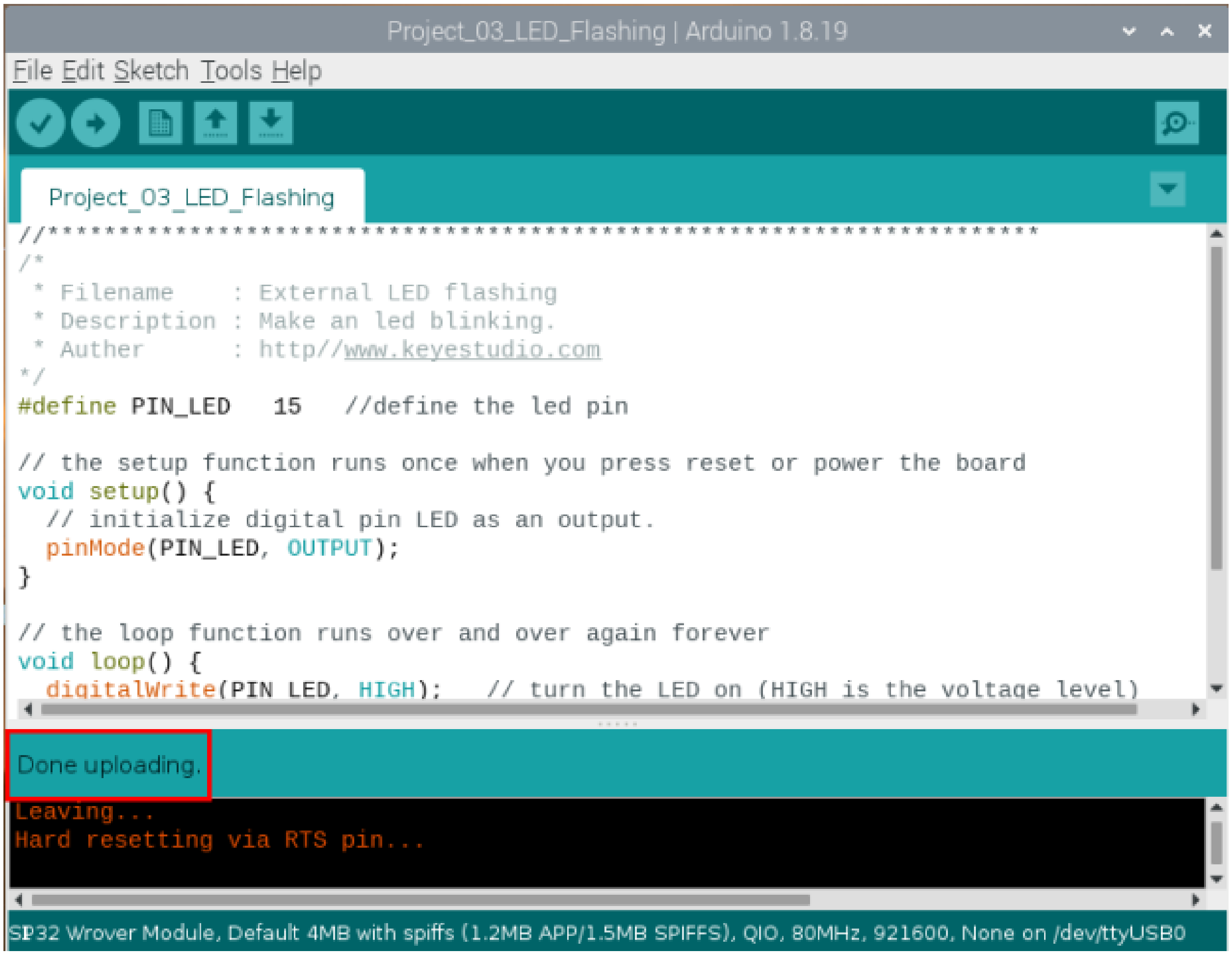
Click to download the project code to ESP32.



Note: If uploading the code fails, you can press the Boot button on ESP32 after click media/d09c4a31563f04a42d451e7bc1a5fb8, and release the Boot button media/dc77bfcf5851c8f43aab6cbe7cec7920.png after the percentage of uploading progress appears, as shown below:

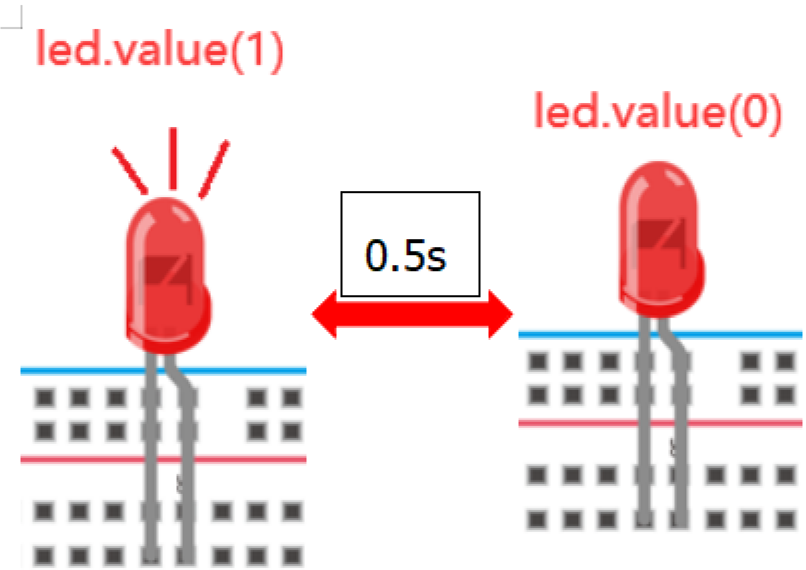
media/157ee2e7687559d9812d24edec758150.png

The Project code is uploaded successfully



9.4.5 Project result

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



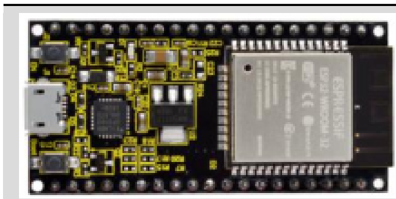
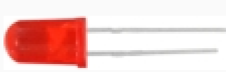
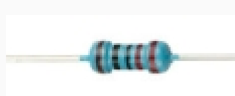
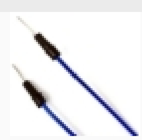
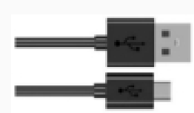
9.5 Project 04: Breathing Led

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

			
ESP32*1	Breadboard*1		
			
Red LED*1	220 Resistor*1	Jumper Wire*2	USB Cable*1

9.5.3 Component knowledge

media/6549bdbfd4e7b6b2b341012105d655e8.png

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.

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%.

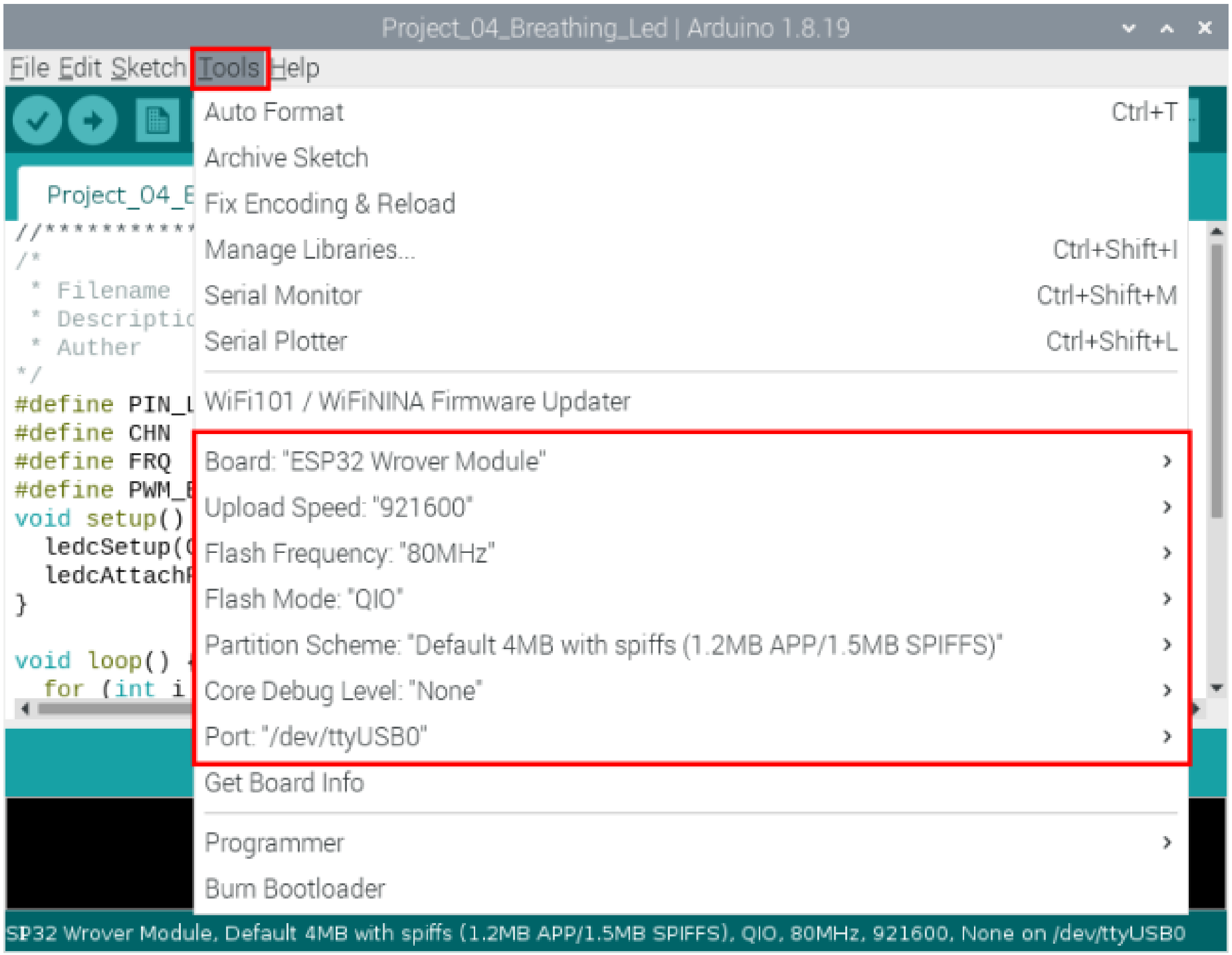
You can open the code we provideIf you haven't downloaded the code file, please click on the link to download itDownload Arduino C Codes file

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
    ledcWrite(CHN, i);
    delay(10);
  }
  for (int i = 255; i > -1; i--) {  //make light fade out
    ledcWrite(CHN, i);
    delay(10);
  }
}
//*************************************************************************************
```

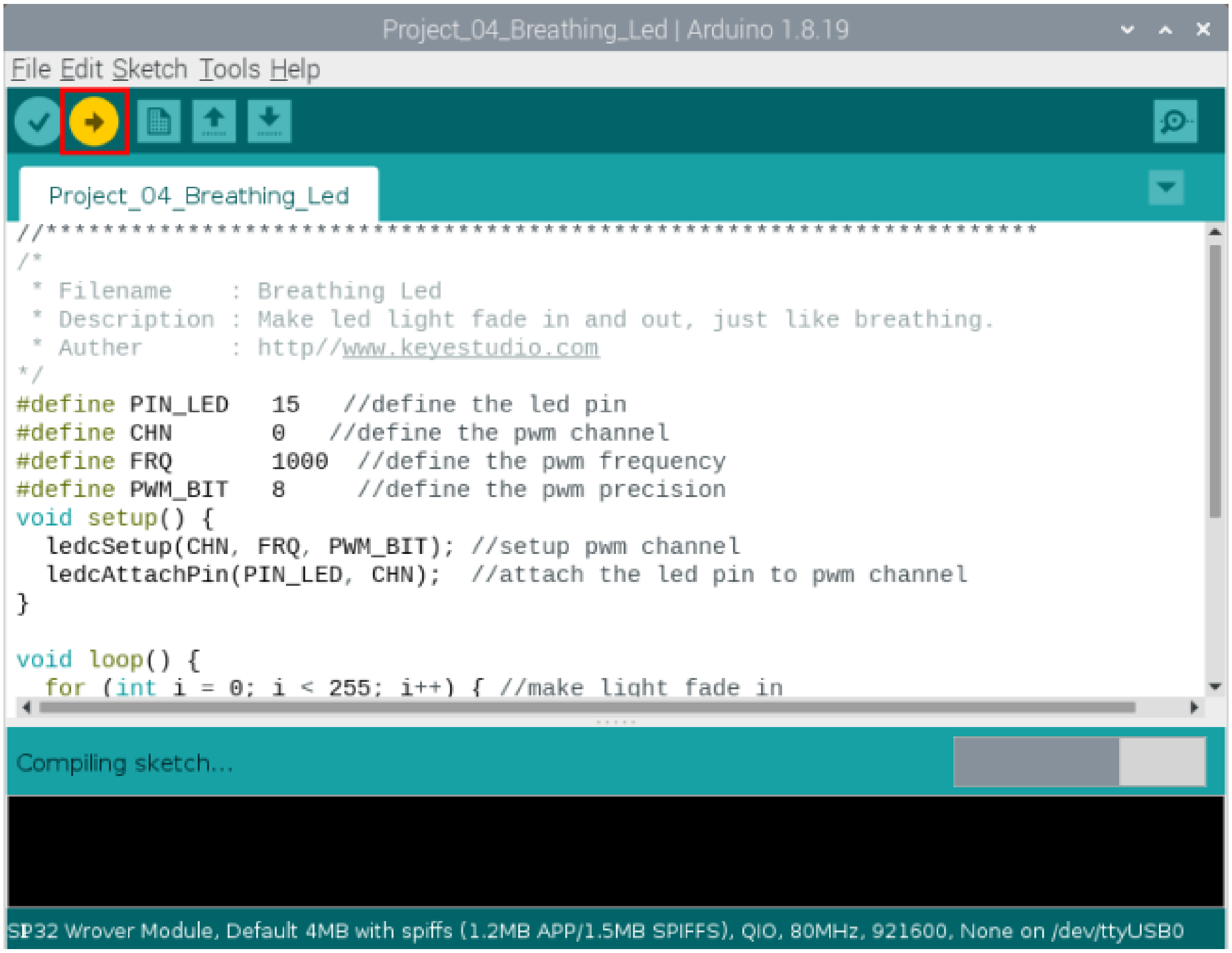
Before uploading Project Code to ESP32, please check the configuration of Arduino IDE.

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



media/b0d41283bf5ae66d2d5ab45db15331ba.png

Click to download the project code to ESP32.



media/d09c4a31563f04a42d451e7bc1a5fb8

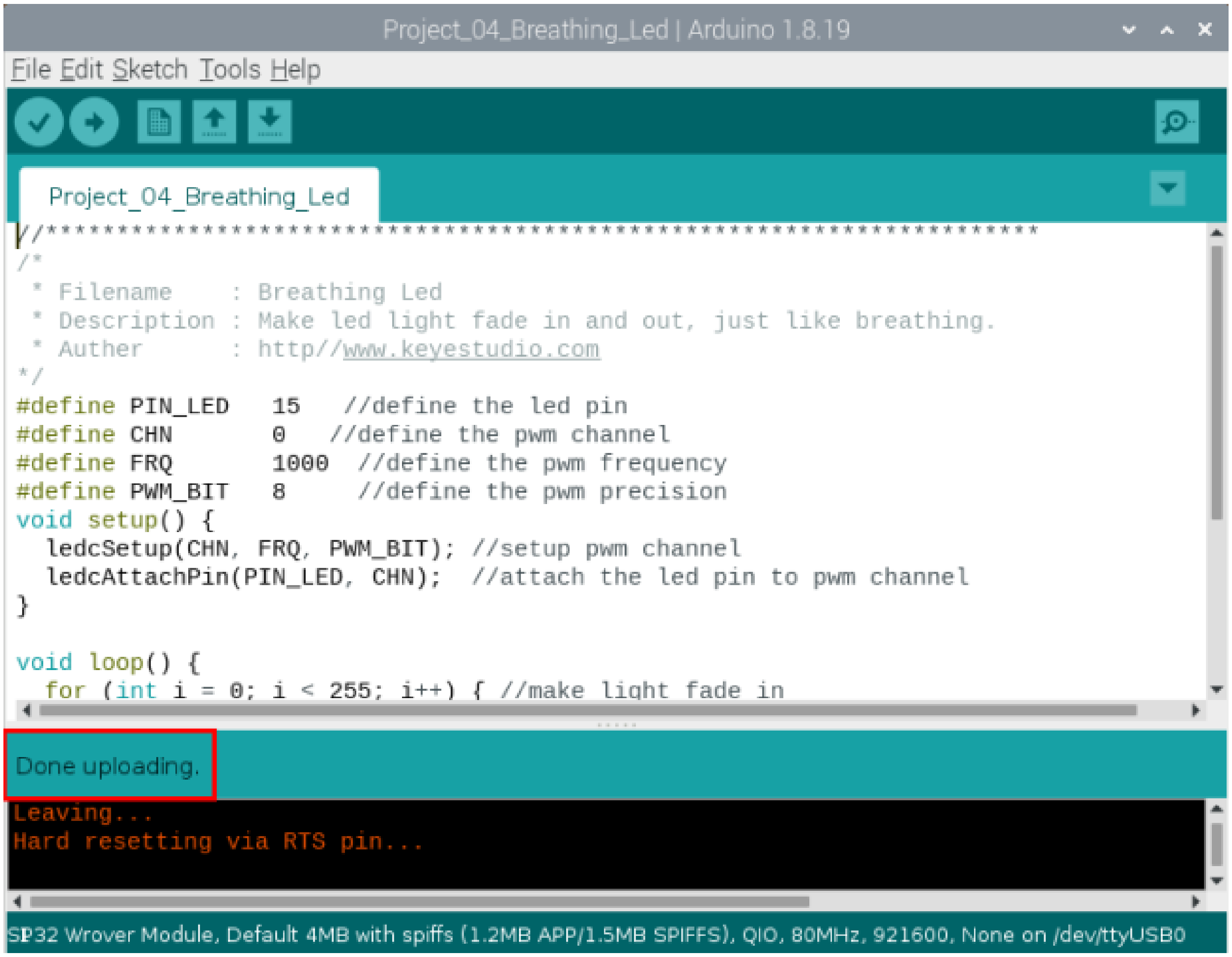
Note: If uploading the code fails, you can press the Boot button on ESP32 after click, and release

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



9.5.6 Project result

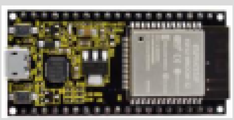
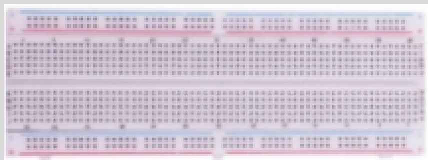
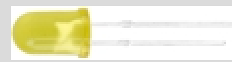
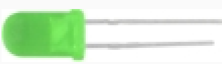
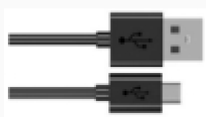
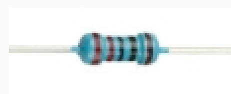
After the project code was uploaded successfully, power up with a USB cable and the LED is turned from ON to OFF and then back from OFF to ONgradually like breathing.

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

			
ESP32*1	Bread board*1	Red LED*1	Yellow LED*1
			
GreenLED*1	USB Cable*1	220 Resistor*3	Jumper Wires

9.6.3 Wiring diagram

media/a991f5cc6f8759eca3b9d01f95fe4854.png

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

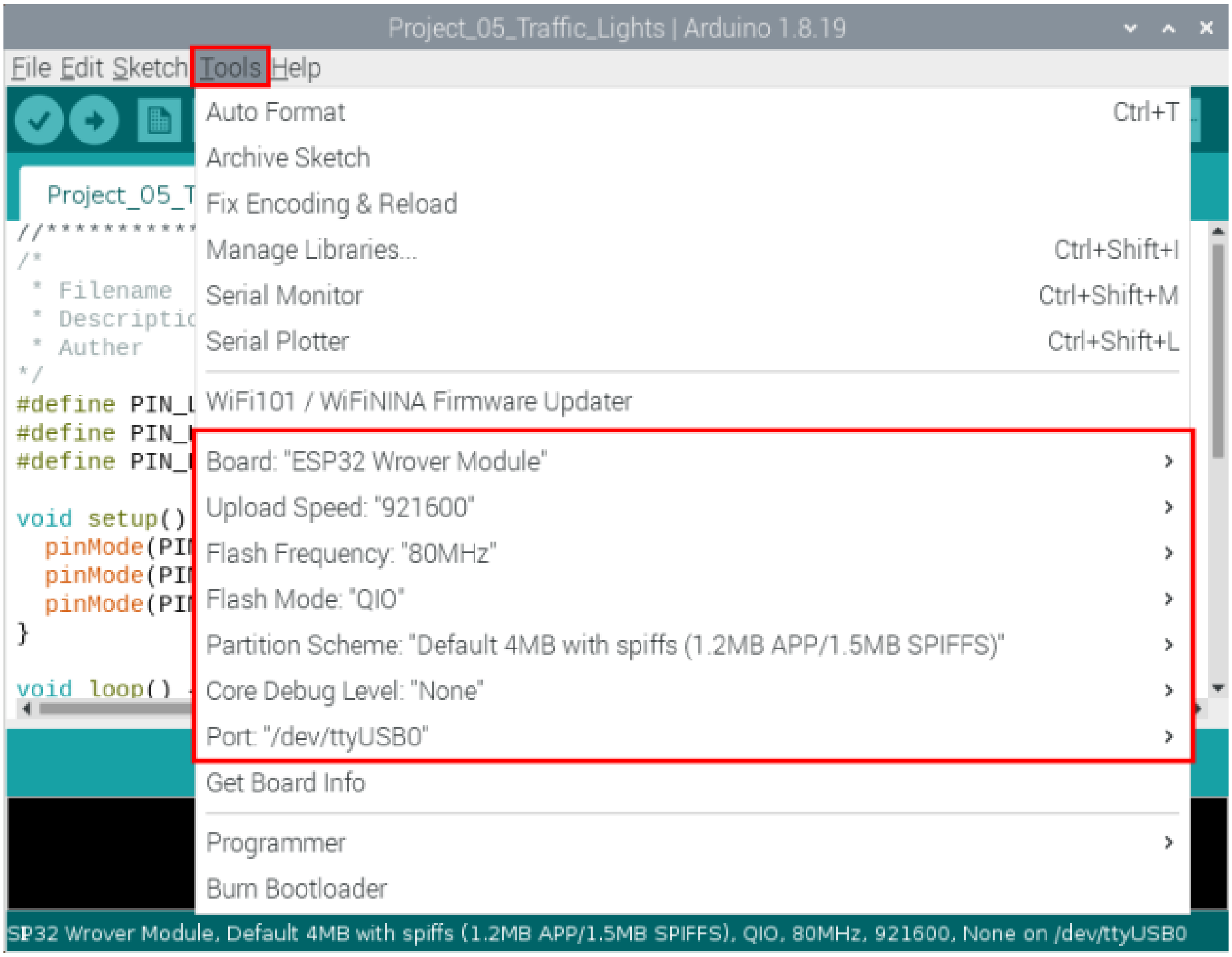
You can open the code we provide: If you haven't downloaded the code file, please click on the link to download itDownload Arduino C Codes file

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.
{
   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
}
//*****************************************************************************************
```

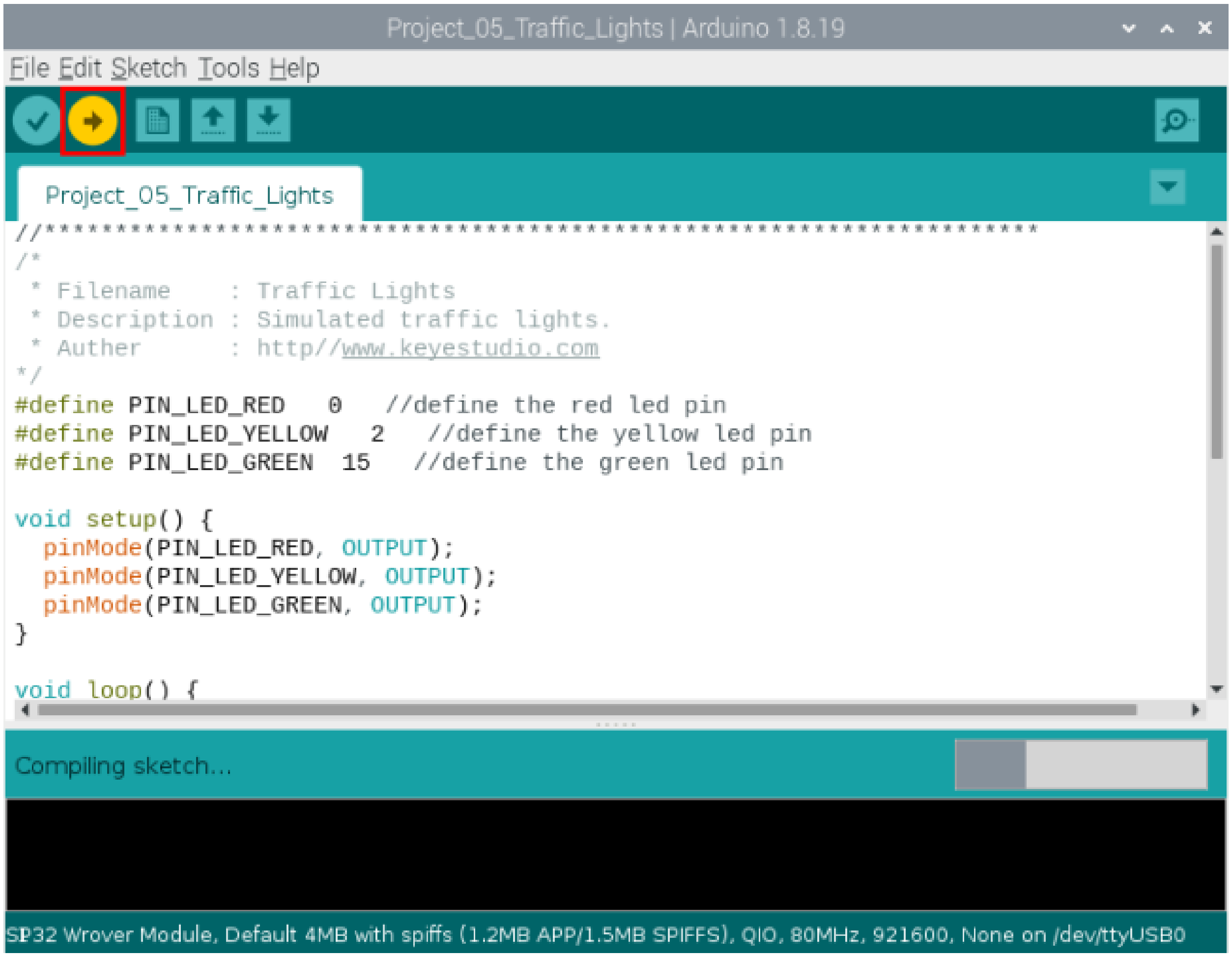
Before uploading Project Code to ESP32, please check the configuration of Arduino IDE.

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



media/b0d41283bf5ae66d2d5ab45db15331ba.png

Click to download the project code to ESP32.

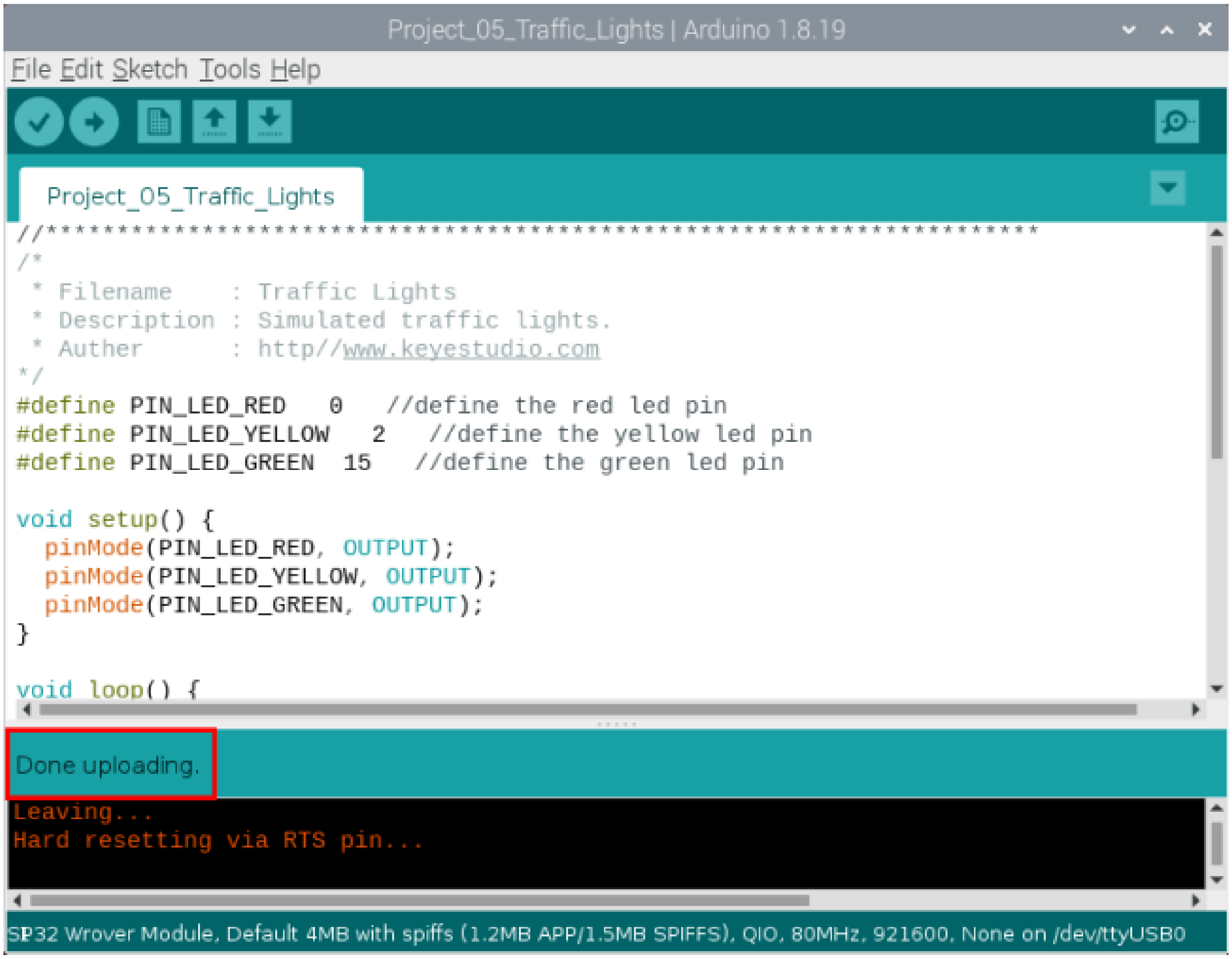


media/d09c4a31563f04a42d451e7bc1a5fb8

Note: If uploading the code fails, you can press the Boot button on ESP32 after click , and release the Boot button media/dc77bfcf5851c8f43aab6cbe7cec7920.png after the percentage of uploading progress appears, as shown below:

media/157ee2e7687559d9812d24edec758150.png

The Project code is uploaded successfully



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

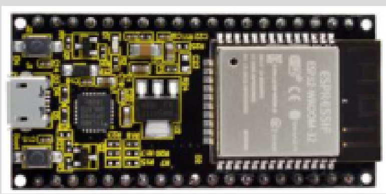
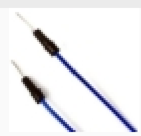
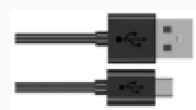
9.7.1 Introduction

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.

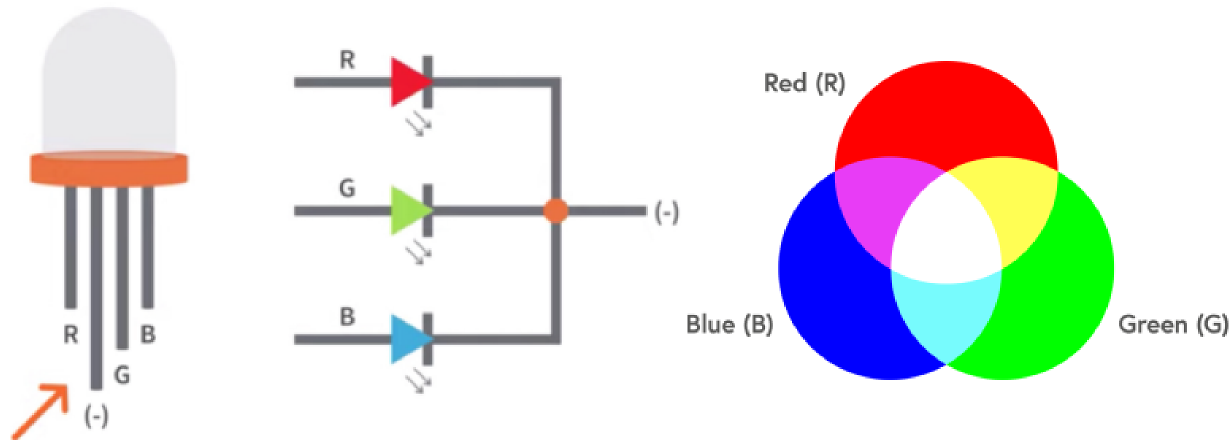
In this project, we will introduce the RGB and show you how to use ESP32 to control the RGB to emit different color light .RGB is pretty basic, but it’s also a great way to learn the fundamentals of electronics and coding.

9.7.2 Components

			
ESP32*1	Breadboard*1		
			
RGB LED*1	220 Resistor*3	Jumper Wires	USB Cable*1

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*210=1,073,741,824(1 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
    ledcSetup(chns[i], 1000, 8);
    ledcAttachPin(ledPins[i], chns[i]);
  }
}

void loop() {
  red = random(0, 256);
  green = random(0, 256);
  blue = random(0, 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

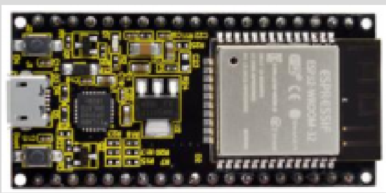
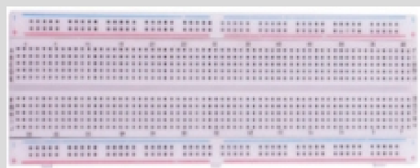
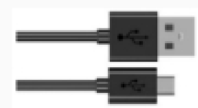
Compile and upload the code to ESP32, after the code is uploaded successfully, power up with a USB cable and you will see that the RGB LED starts to display random colors.

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

			
ESP32*1	Breadboard*1		
			
Red LED*1	220 Resistor*1	Jumper Wires	USB Cable*1

9.8.3 Wiring diagram:

media/548f889607bdb0ce017c58f323c85dfa.png

Note:

How to connect a LED

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.

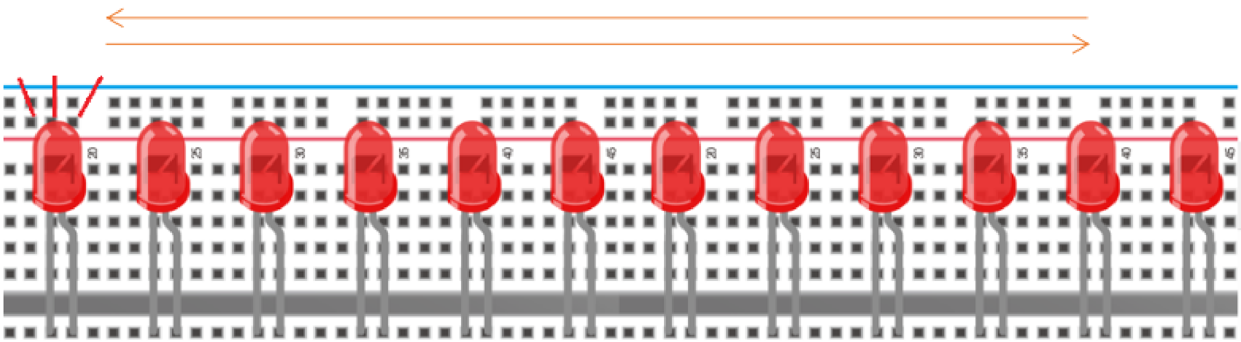
You can open the code we provide: If you haven’t downloaded the code file, please click on the link to download itDownload Arduino C Codes file

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.

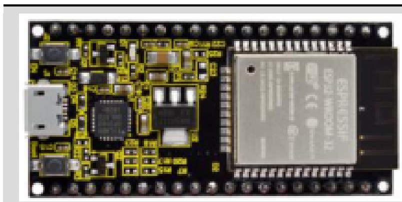
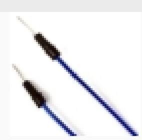
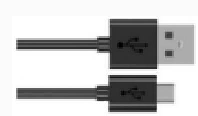


9.9 Project 081-Digit Digital Tube

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

			
ESP32*1	Breadboard*1		
			
1-Digit 7-Segment Display*1	220 Resistor*8	Jumper Wires	USB Cable*1

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). Thedigital 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".

media/28fd057848fbe0e8c8e3362768e7aa44.png

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)

(continued from previous 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)

(continued from previous 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)

(continued from previous page)

```
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
}}
//*********************************************************************
```

9.9.6 Project result

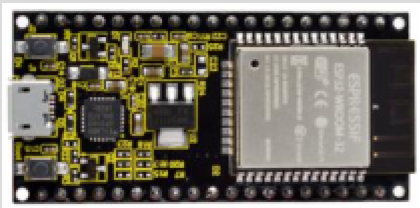
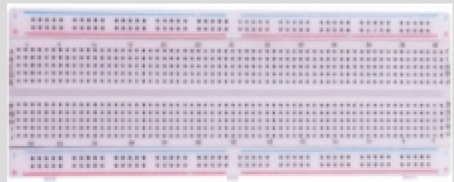
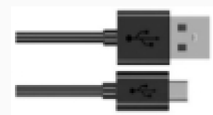
Compile and upload the code to ESP32, after the code is uploaded successfully, power up with a USB cable and you will see that the display will display numbers from 9 to 0.

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-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 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 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++){
    DigitalTube.displayBit(1,num);
    DigitalTube.displayBit(2,num);
    DigitalTube.displayBit(3,num);
    DigitalTube.displayBit(4,num);
    delay(1000);
  }
 }
//**************************************************************************************
```

9.10.7 Project result

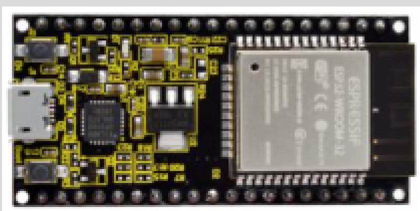
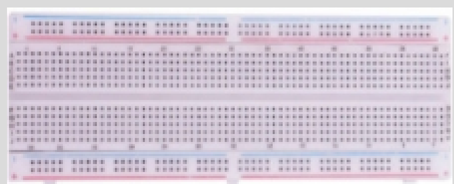
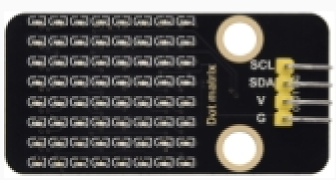
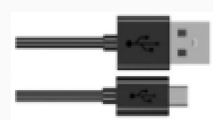
Compile and upload the code to ESP32, after the code is uploaded successfully, power up with a USB cable and you will see that 4-digit 7-segment display displays four digitsand repeat these actions in an infinite loop.

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

		
ESP32*1	Breadboard*1	
		
88 *dot matrix module*1	M-F Dupont Wires	USB Cable*1

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			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					

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 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 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.

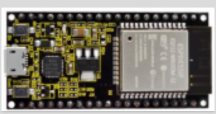
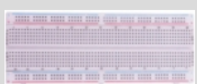
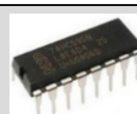
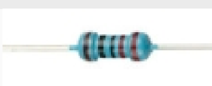
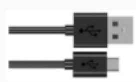
9.12 Project 1174HC595N Control 8 LEDs

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

			![img](file:///C:\Users\ADMINI~1\AppData\Local\Temp\ksohtml16172
ESP32*1	Breadboard*1	74HC595N chip*1	Jumper Wires
			
220Resistor*8	Red LED*8	USB Cable*1	

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 10—SCLR	Remove shift register: When this pin is in low level, the content in shift register will be cleared. . . 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.

media/a6d03617539b70d6d69fa7e9acb25be9.png

media/11a03579b6cf94599f00554bfe014a3b.png

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**.

```
//**********************************************************************
/*
 * Filename    : 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␣
↪LED bar graph.
  // This variable is assigned to 0x01, that is binary 00000001, which indicates only␣
↪one LED light on.
  byte x = 0x01;    // 0b 0000 0001
  for (int j = 0; j < 8; j++) { // Let led light up from right to left
    writeTo595(LSBFIRST, x);
    x <<= 1; // make the variable move one bit to left once, then the bright LED move␣
↪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␣
↪output port.
  digitalWrite(latchPin, HIGH);
}
//*********************************************************************************
```

9.12.6 Project result

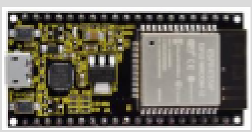
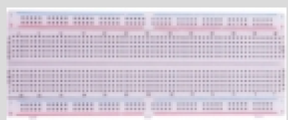
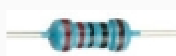
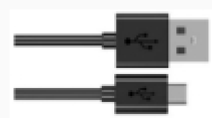
Compile and upload the code to ESP32, after the code is uploaded successfully, power up with a USB cable and you will see that the 8 LEDs start flashing in flowing water mode.

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

			
ESP32*1	Breadboard*1	Active buzzer*1	
			
NPN transistor(S8050)*1	1kResistor*1	Jumper Wires	USB Cable*1

9.13.3 Component knowledge

media/11ec5ddc982db9928341e858aab94652.png

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.

media/02dad9f2fcac0d7bfe4cc135d2301aa6.png

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.

media/2a9755ec14ab58c67d7d8341601d8dbc.png

9.13.4 Wiring diagram

media/5c215684c8945622441478edb6f16e30.png

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

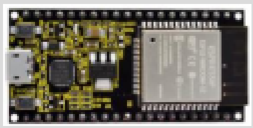
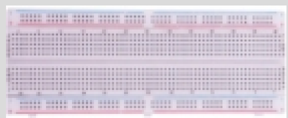
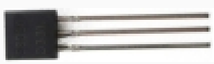
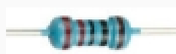
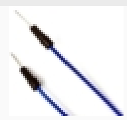
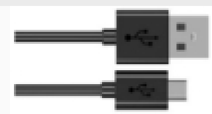
Compile and upload the code to ESP32, after the code is uploaded successfully, power up with a USB cable and you will see that the active buzzer beeps.

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

			
ESP32*1	Breadboard*1	Passive Buzzer *1	
			
NPN transistor(S8050)*1	1kResistor*1	Jumper Wires	USB Cable*1

9.14.3 Component knowledge

media/8d0020e53824072cbe9d4f7d2f8acb4f.png

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:

media/9c12d89ce3f10c838e63f1334f41fc9e.png

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.
 * 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,␣
↪196, 330, 392,440, 349, 392, 330, 262, 294, 247, 392, 370, 330, 311, 330, 208, 220,␣
↪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

int noteDurations[] = {8,4,4,8,4,2,2,3,3,3,4,4,8,4,8,8,8,4,8,4,3,8,8,3,3,3,3,4,4,8,4,8,8,
↪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,
↪4,8,4,8,4,8,2,8,4,4,8,4,1,8,4,4,8,4,8,4,8,2};
void setup() {
pinMode(BUZZER_PIN, OUTPUT); // Set the buzzer to output mode
}
```

(continues on next page)

(continued from previous 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
  }
}
//*************************************************************************
```

9.14.6 Project result

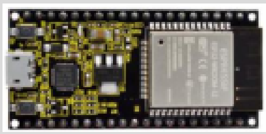
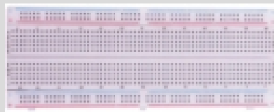
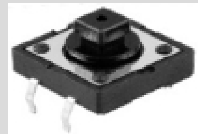
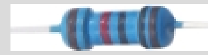
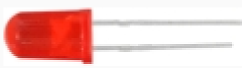
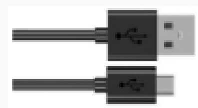
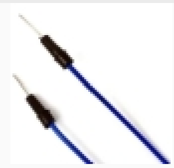
Compile and upload the code to ESP32, after the code is uploaded successfully, power up with a USB cable and you will see that the passive buzzer plays music.

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

				
ESP32*1	Breadboard*1	Button*1	10K Resistor*1	
				
Red LED*1	220 Resistor*1	USB Cable*1	Jumper Wires	Button Cap*1

9.15.3 Component knowledge

media/5b8fea4657b47510d199f740fdcaaa9d.png

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 media/d2a204e61c768f18924150db58aee093.png. 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:

media/5e42fde9[illegible]d66ba3a460.png media/be87[illegible].png

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.

media/c0d68d1134b0b4097e8983ed2cac07fc.jpeg

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?

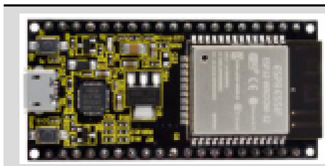
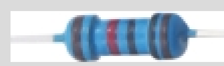
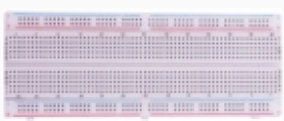
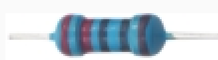
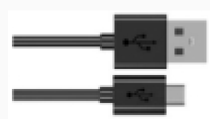
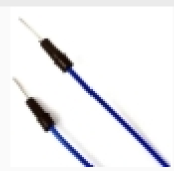
9.16 Project 15Tilt and LED

9.16.1 Introduction

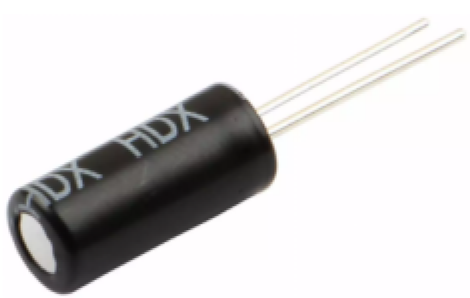
The ancients without electronic clock, so the hourglass are invented to measure time. The hourglass has a large capacity on both sides, and which is filled with fine sand on one side. What's more, there is a small channel in the middle, which can make the hourglass stand upright , the side with fine 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

			
ESP32*1	Tilt Switch*1	Red LED*4	10K Resistor*1
			
Breadboard*1	220 Resistor*4	USB Cable*1	Jumper Wires

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:

media/bf8b10ad248ac939ac4ef96d02ed87c7.png

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 = 0;
void setup()
{
     for(int i=16;i<20;i++)
  {
        pinMode(i, OUTPUT);
  }
    pinMode(SWITCH_PIN, INPUT);
 for(int i=16;i<20;i++)
  {
    digitalWrite(i,0);
  }
  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++)
  {
    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

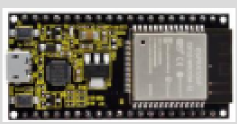
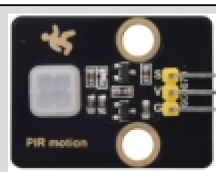
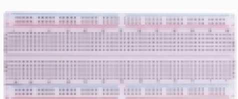
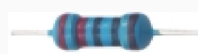
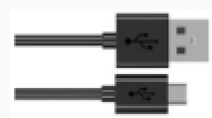
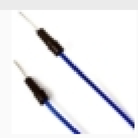
Compile and upload the code to ESP32, after the code is uploaded successfully, power up with a USB cable and you will see that when you tilt the breadboard to an angle, the LEDs will light up one by one. When you turn the breadboard to the original angle, the LEDs will turn off one by one. Like the hourglass,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

				
ESP32*1	Human Body Infrared Sensor*1	Active Buzzer*1	Red LED*1	
				
Breadboard*1	M-F Dupont Wires	220Resistor*1	USB Cable*1	Jumper Wires

9.17.3 Component knowledge

media/51dc6a366ebb066960c32263c37e0244.png

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"**.

```
//**********************************************************************
/*
 * Filename    : Burglar Alarm
```

(continues on next page)

(continued from previous 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
#define ledPin   0     // the pin of the PIR motion sensor
#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␣
↪off buzzers
   digitalWrite(ledPin, LOW);  // the led turn off
 }
}
//*************************************************************************************
```

9.17.6 Project result

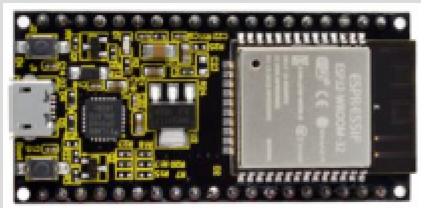
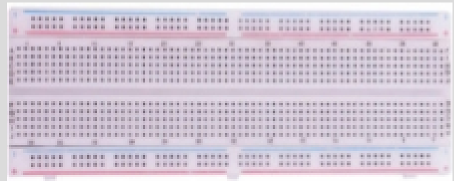
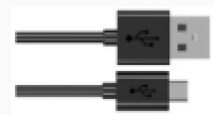
Compile and upload the code to ESP32, after the code is uploaded successfully, power up with a USB cable and you will see that if the sensor detects someone moving nearby, the buzzer will continuously issue an alarm and the LED will continuously flash.

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	
		
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

media/072d954dac310add077688398ad59af2.png

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 lCD128 *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("%^&(){}:;'|?,.~\\[]");
}
//**********************************************************************************
```

9.18.7 Project result

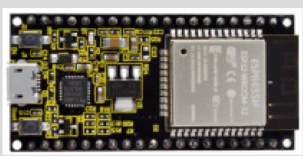
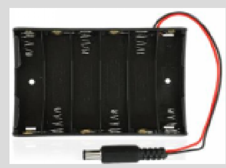
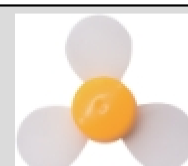
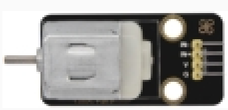
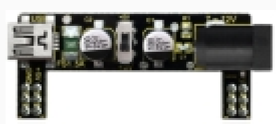
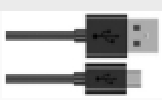
Compile and upload the code to ESP32, after the code is uploaded successfully, power up with a USB cable and you will see that the 128X32LCD module display will show“KEYESTUDIO”at the first line, “ABCDEFGHIJKLMNOPQR”will be displayed at the second line, “123456789±*/<>=$@”will be shown at the third line 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

			
ESP32*1	Breadboard*1	Battery Holder*1	Fan*1
			
130 Motor Module*1	Keyestudio bread board special power module*1	M-F Dupont Wires	No.5 battery (self-provided)*6
			
USB Cable*1			

9.19.3 Component knowledge :

media/75a9140bdb671783bb79a71ca76fae69.png

130 motor module: The motor control module uses the HR1124S motor control chip. which is a single-channel H-bridge 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:

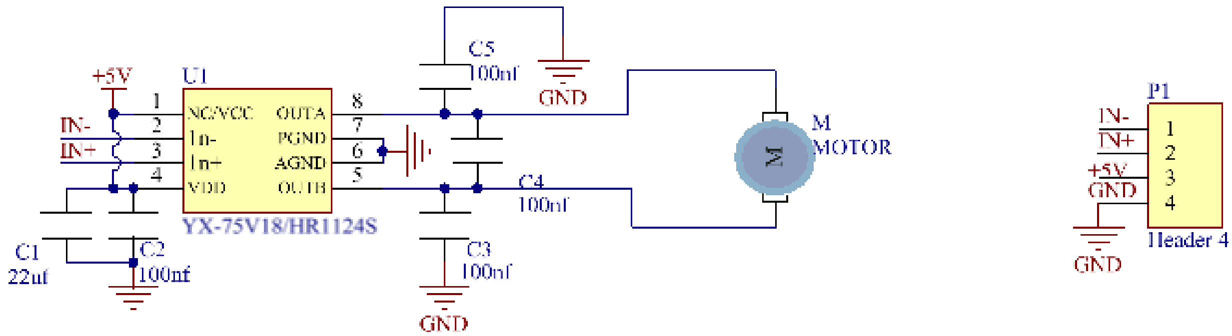
Working voltage: 5V

Working current: 200MA

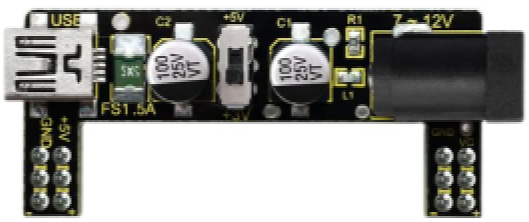
Working power: 2W

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 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

media/83f622be72c1b5501ffeab7bd1caf421.png

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.
 * 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);
}
//*******************************************************************************
```

9.19.6 Project result

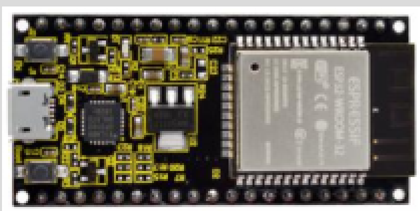
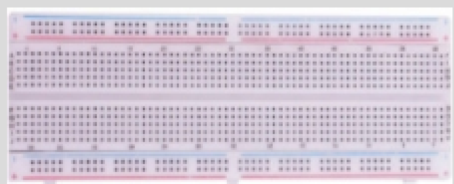
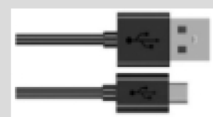
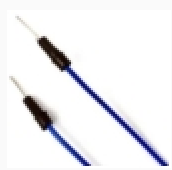
Compile and upload the code to ESP32, after the code is uploaded successfully, power up with a USB cable and you will see thatthe small fan turns counterclockwise for 5 seconds and stops for 2 seconds, and then turns clockwise for 5 seconds and stops for 2 seconds, which repeats in an endless loop.

9.20 Project 19Servo Sweep

9.20.1 Introduction

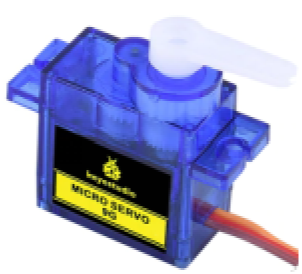
Servo is an electric motor that can rotate very precisely. At present, it has been widely used in toy cars, remote control helicoptersairplanesrobots, etc. In this project, we will use ESP32 to control the rotation of the servo.

9.20.2 Components

		
ESP32*1	Breadboard*1	USB Cable*1
		
Servo*1	Jumper Wires	

9.20.3 Component knowledge

Servo



The servo is a kind of position servo driver, which is mainly composed of a housinga circuit boarda copless motora 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
    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
  }
}
//********************************************************************************
```

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.

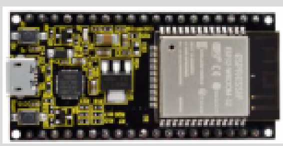
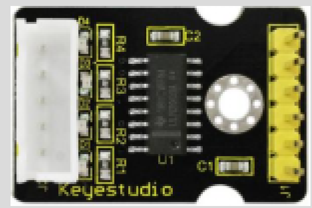
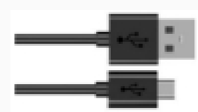
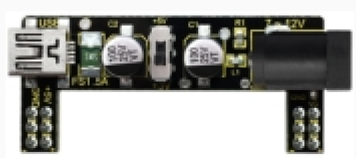
media/c5250405a4290ecb2d758ff1097310c7.png

9.21 Project 20Stepping Motor

9.21.1 Introduction

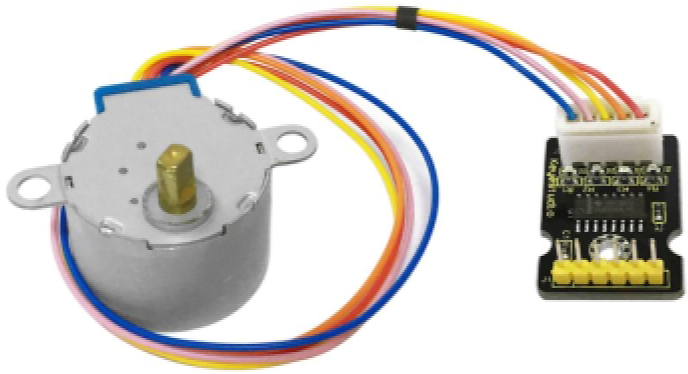
Stepper motor is the most important part of industrial robot 3D printer lathes and other mechanical equipment with accurate positioning. In this project, we will use ESP32 control ULN2003 stepper motor drive board to drive the stepper motor to rotate.

9.21.2 Components

		
ESP32*1	Breadboard*1	ULN2003 Stepper Motor Drive Board*1
		
Stepper Motor *1	M-F Dupont Wires	USB Cable*1
		
Battery Holder*1	Keyestudio bread board special power module*1	No.5 battery (self-provided)*6

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.

media/32748e0804b1fff434181cb228b23242.png

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 is turned on, and the rotor is turned to 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.

media/b8ae50bbdee2dd5bc683e8c450baee6a.png

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.

media/6fa632d2b70e97dd55565d23ec15d245.png

9.21.4 Wiring Diagram

media/6333a59ee8dd57f7ceb5eaaec8d588df.png

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++) {
    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 = 0x01;
  // Decide the shift direction according to the rotation direction
  if (dir) {  // ring shift left
    out != 0x08 ? out = out << 1 : out = 0x01;
  }
  else {      // ring shift right
    out != 0x01 ? out = out >> 1 : out = 0x08;
  }
  // Output singal to each port
  for (int i = 0; i < 4; i++) {
    digitalWrite(outPorts[i], (out & (0x01 << i)) ? HIGH : LOW);
  }
}

void moveAround(bool dir, int turns, byte ms){
  for(int i=0;i<turns;i++)
    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.

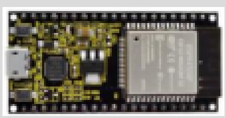
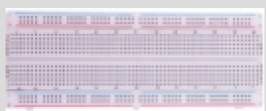
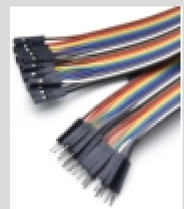
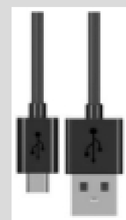
media/8dc4a0547390e0108c3960c31d330ee7.png

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

				
ESP32*1	Breadboard*1	Relay Module*1	M-F Dupont Wires	USB Cable*1

9.22.3 Component knowledge

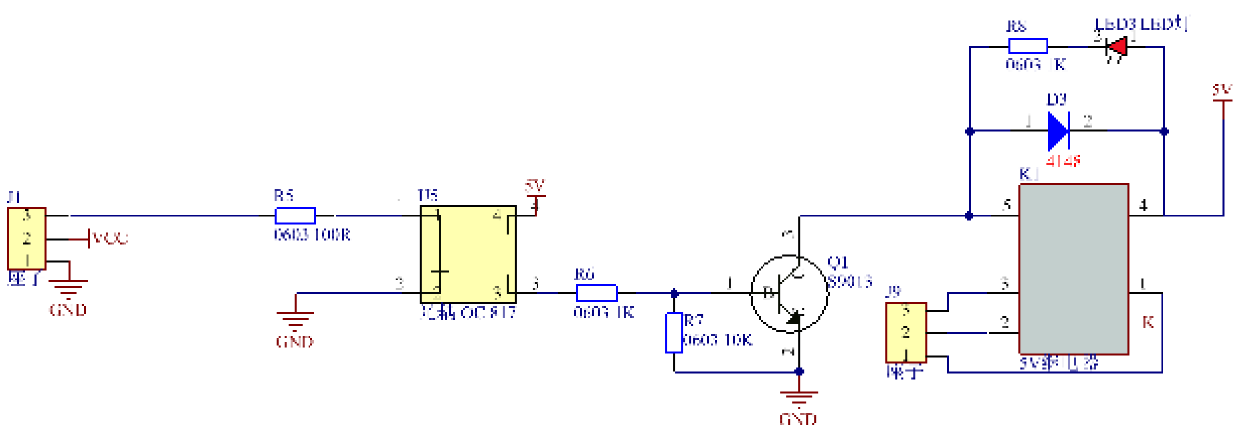
Relay: It is an “automatic switch” that uses a small current to control the operation of a large current.

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

media/1741d3cb0405c740378ef7ef96df6072.png

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.

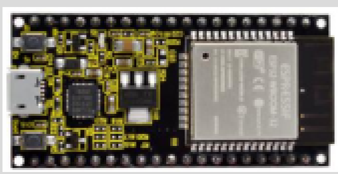
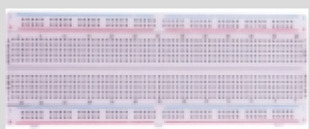
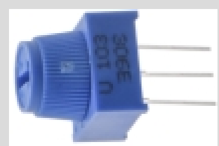
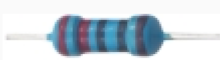
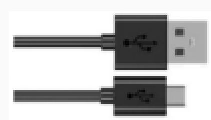
9.23 Project 22Dimming Light

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 ~ 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.

media/f6c45550f4adf8373d7f1d01daec2c64.png

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:

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

DACThe 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"**.

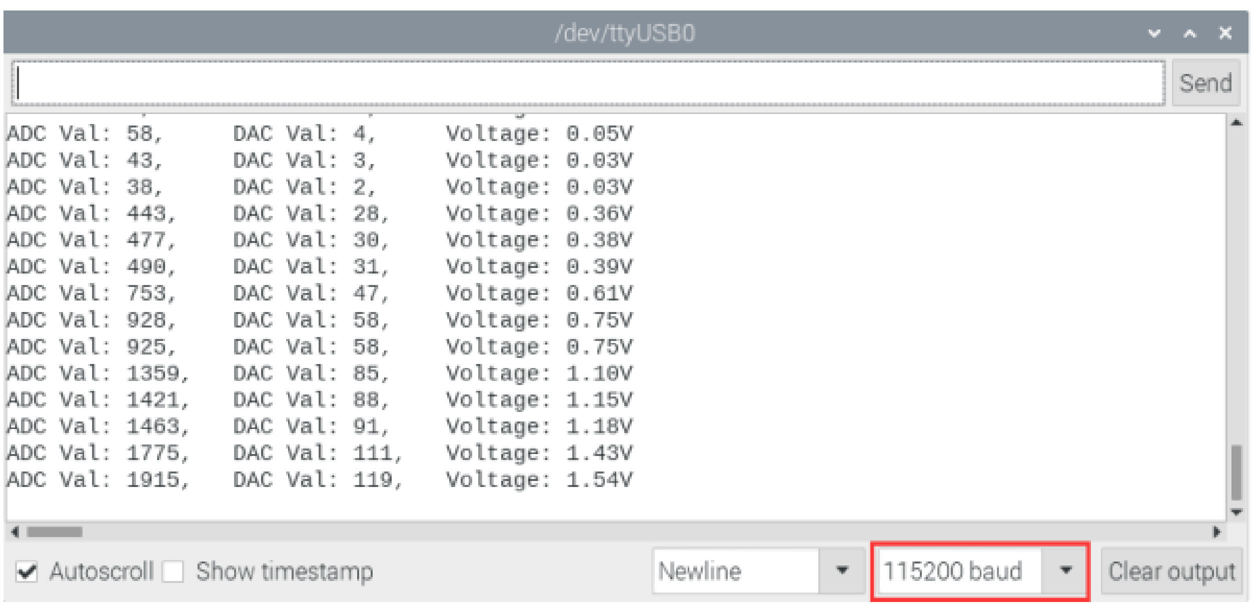
```
//**********************************************************************************
/*
 * Filename    : Read Potentiometer Analog Value
```

(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() 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 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:



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.
 * Auther      : http//www.keyestudio.com
*/
#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);
}
//**********************************************************************************
```

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.

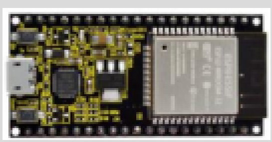
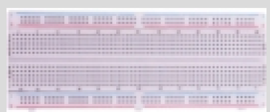
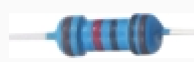
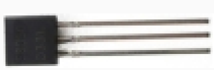
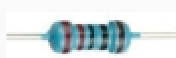
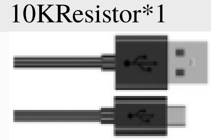
media/eca30dead3f4923afa0dcb0306db2319.jpeg

9.24 Project 23Flame Alarm

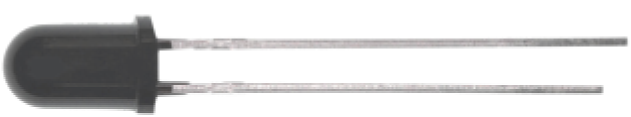
9.24.1 Introduction

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

			
ESP32*1	Breadboard*1	Red LED*1	Active Buzzer*1
			
Flame Sensor*1	220Resistor*1	10KResistor*1	Jumper Wires
			
NPN transistor(S8050)*1	1k Resistor*1	USB Cable*1	

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

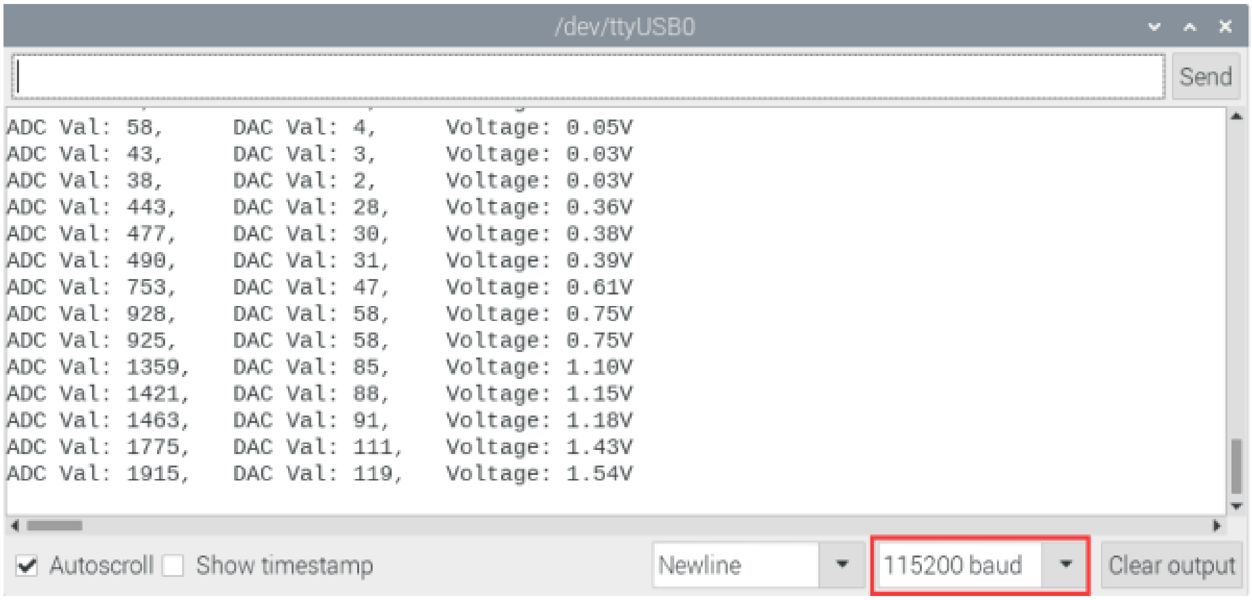
media/76ce57355da1df27e049bdc6e19f0650.png

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
 * 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 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 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.



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.

media/e9fa0e50df23c1f2e58fdd319ad21b4c.png

9.24.6 Project code

media/40a3ea572836945268b22dfc0cce29c3.png

Note the threshold of 500 in the code can be reset itself as required)

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.
 * Auther      : http//www.keyestudio.com
*/
#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
  }
}
//**********************************************************************************
```

9.24.7 Project result

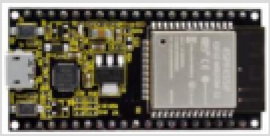
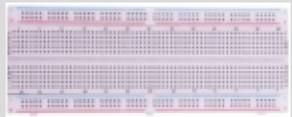
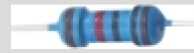
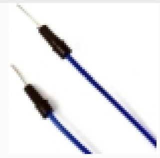
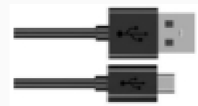
Compile and upload the code to ESP32, after the code is uploaded successfully, power up with a USB cable and you will see that when the flame sensor detects the flame, the LED will flash and the buzzer will alarm; otherwise, the LED does not light up and the buzzer does not sound.

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

			
ESP32*1	Breadboard*1	Red LED*1	10KResistor*1
			
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

media/5a7f7e641eb78007760a94151c1d80a5.png

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**

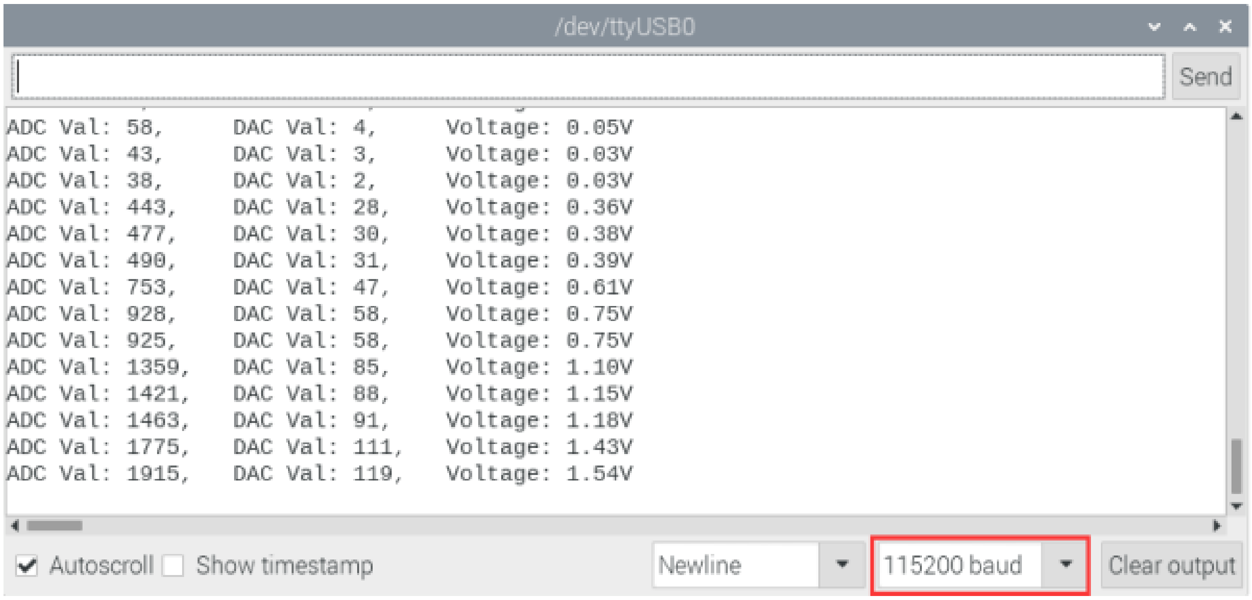
We first use a simple code to read the ADC valueDAC value and voltage value of the photoresistor and print them out. Please refer to the following wiring diagram media/b762098c798beb08e4d433137c317dc7.png

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() 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 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 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 will gradually increase. On the contrary, the ADC valueDAC value and voltage value decreases gradually.



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/77a0c534501f51e7fe7aa221e4db71d9.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)

```
  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);
}
//*********************************************************************************
```

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.

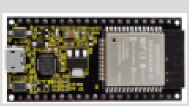
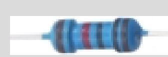
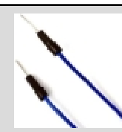
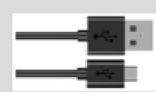
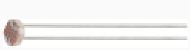
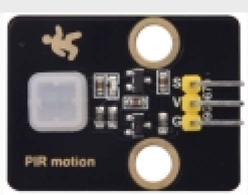
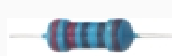
9.26 Project 25Human Induction Lamp

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

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

9.26.3 Wiring Diagram

media/69f49d65054a9246acf4adc534217027.png

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

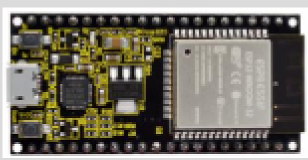
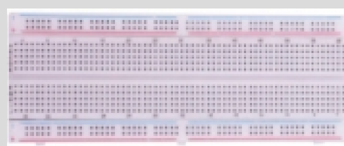
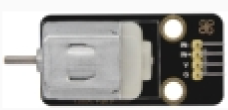
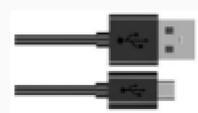
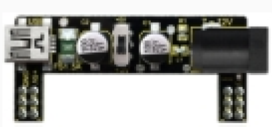
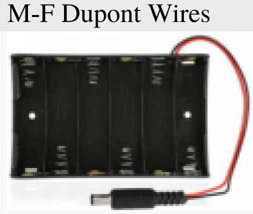
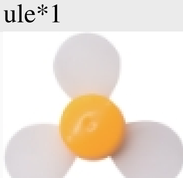
Compile and upload the code to ESP32, after the code is uploaded successfully, power up with a USB cable and you will see thatWhen your hand covers the photosensitive part of the photoresistor to simulate darkness, then shake your other hand in front of the Human Body Infrared Sensor, the external LED 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

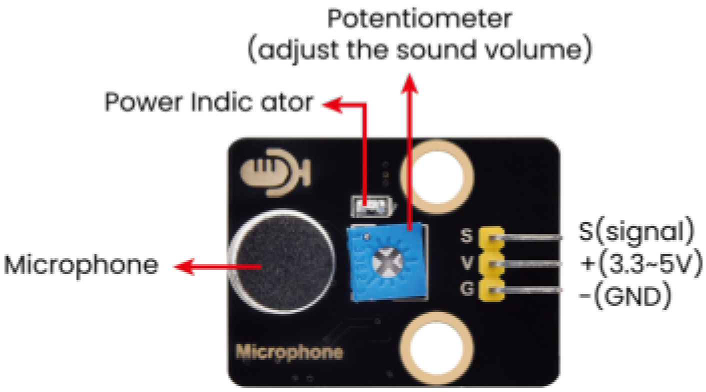
9.27.1 Introduction

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

		
ESP32*1	Breadboard*1	Sound Sensor*1
		
130 Motor Module*1	M-F Dupont Wires	USB Cable*1
		
Keyestudio bread board special power module*1	Battery Holder*1	No.5 battery (self-provided)*6
		
Fan*1		

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

media/87fb44c475d1f53aa5905cebfed55ea2.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 26Sound Control Fan\Project_26.1_Read_Sound_Sensor_Analog_Value"**.

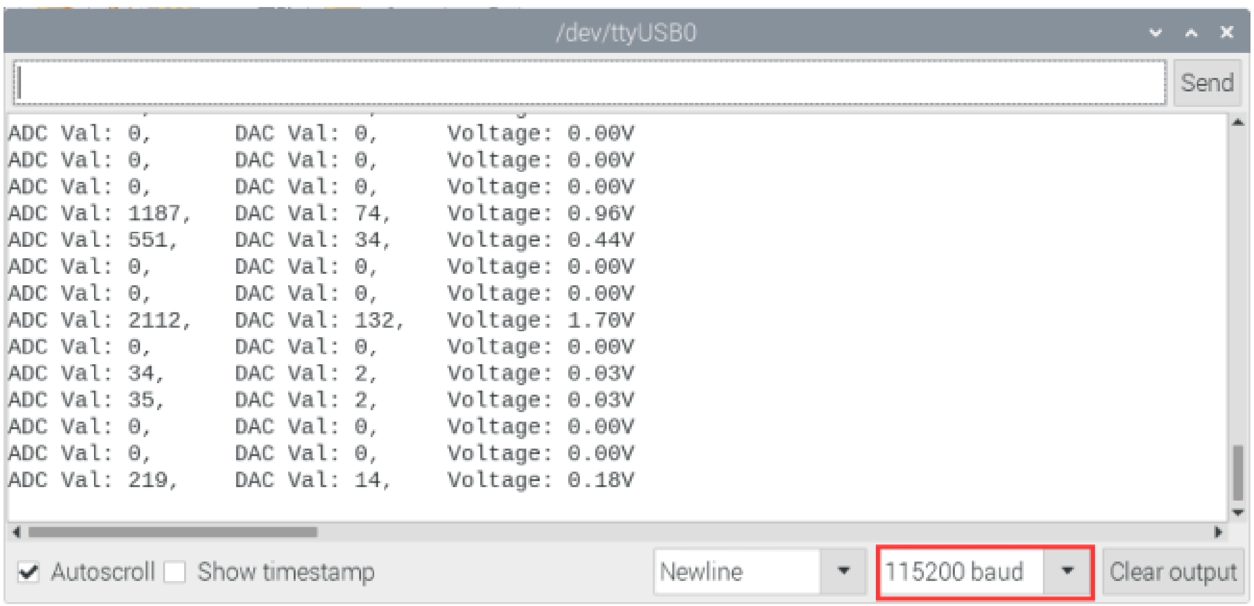
```
//**********************************************************************************
/*
 * Filename    : Read Sound Sensor Analog Value
 * Description : Basic usage of ADC
 * Auther      : http//www.keyestudio.com
*/
#define PIN_ANALOG_IN  36  //the pin of the Sound Sensor

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 value. The input and output voltage are calculated according to the previous formula, and the information
```

(continues on next page)

(continued from previous page)

```
↪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 sound sensor.When you clap your hands to the sensor, the ADC valueDAC value and voltage value will change significantly.



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 voice-controlled fan. The wiring diagram is as follows

media/5a08d512265e03c997bf81a9e2bfecbb.png

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

9.27.6 Project code

media/c20911df19d11290cf099072fe250029.png

Note The threshold 600 in the code can be reset itself as needed)

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

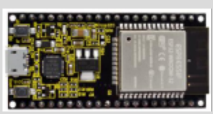
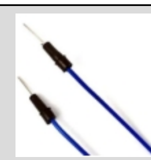
Compile and upload the code to ESP32, after the code is uploaded successfully, external power supply and power on. and then you will see that clap your hands to the sound sensor, and when the sound intensity exceeds a threshold, the small fan rotates; conversely, the small fan doesn't rotate.

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.

9.28.2 Components

					
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		

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 ℃, it output 0V; If increasing 1 ℃, the output voltage will increase 10mv. The output temperature is 0℃ to 100℃, 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

media/041471b9fabd75ef9dc3951598e342f8.png

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 measured voltage value,Celsius and Fahrenheit valuesthrough 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));
```

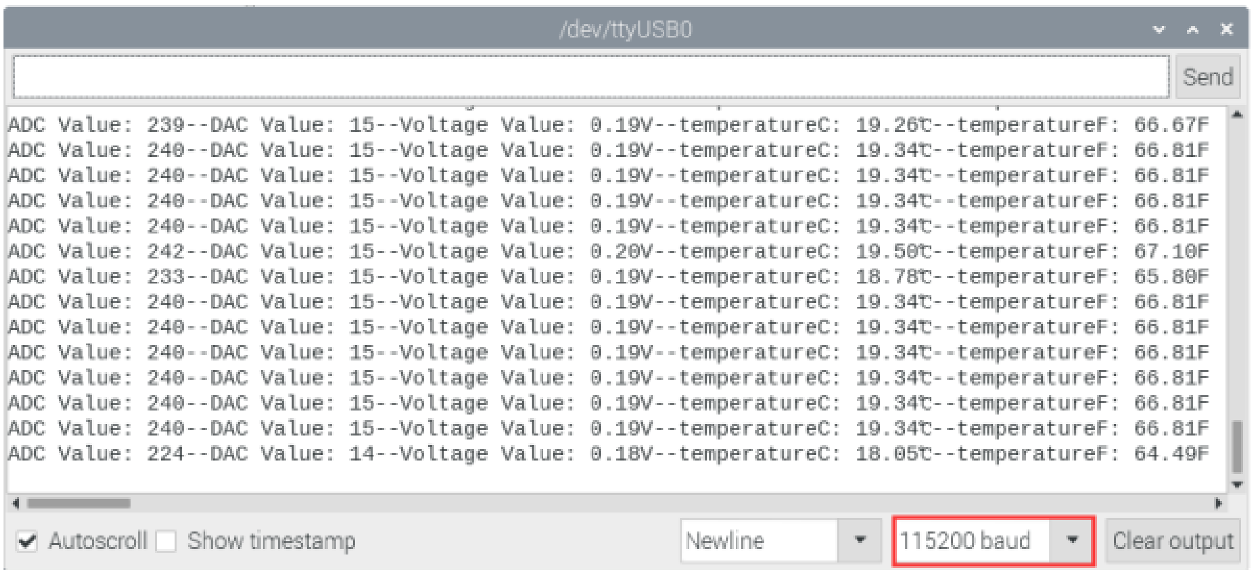
(continues on next page)

(continued from previous page)

```
  Serial.print("--Voltage Value: " + String(voltage) + "V");
  Serial.print("--temperatureC: " + String(temperatureC) + "℃");
  Serial.println("--temperatureF: " + String(temperatureF) + "F");
  delay(200);
}
//*************************************************************************************
```

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.



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
#define PIN_REDLED     15      //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) + "℃");
  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) {
    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.

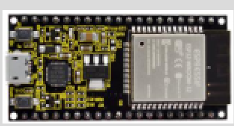
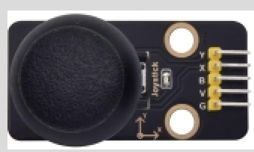
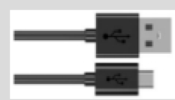
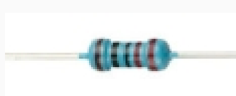
9.29 Project 28Rocker control light

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

				
ESP32*1	Breadboard*1	Rocker Module*1	USB Cable*1	M-F Dupont Wires
				
RGB LED*1	220Resistor*3	Jumper Wires		

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

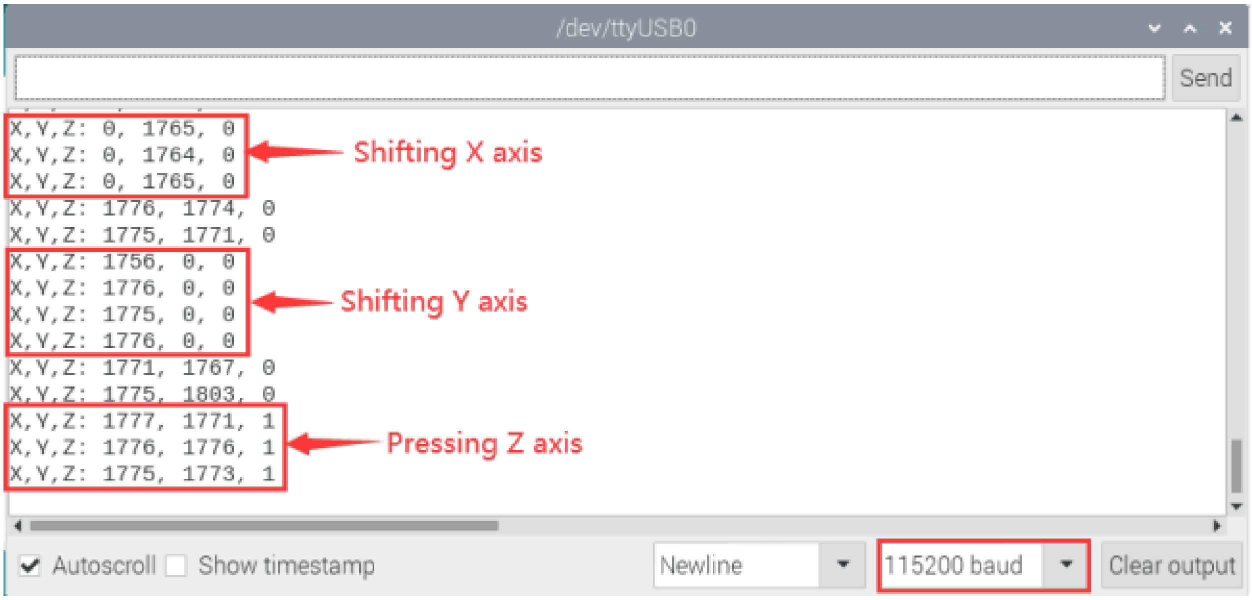
You can open the code we provide: If you haven't downloaded the code file, please click on the link to download itDownload Arduino C Codes file

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.
 * Auther      : http//www.keyestudio.com
*/
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 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

media/4ec49b488fedf216d03e49f83bc8443a.png

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.
 * 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() {
```

(continues on next page)

(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.
  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);
     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.

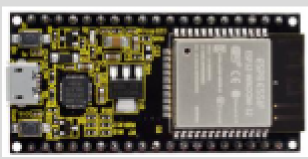
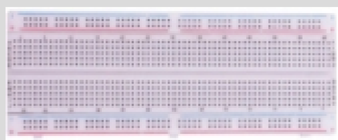
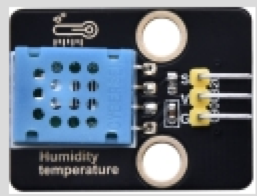
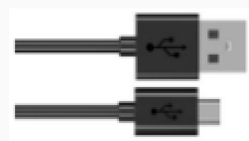
media/9c2d0d8777200827b16c49b752d45c4c.jpeg

9.30 Project 29Temperature Humidity Meter

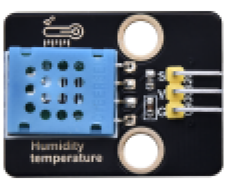
9.30.1 Introduction

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

		
ESP32*1	Breadboard*1	Temperature and Humidity Sensor*1
		
LCD 128X32 DOT*1	M-F Dupont Wires	USB Cable*1

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±5%RH, temperature is±2℃, range humidity is 20 to 90%RH, and temperature is 0 to 50℃. 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

Description	Definition
Start signal	Microprocessor pulls data bus (SDA) down at least 18ms for a period of time(Maximum is 30ms), notifying the sensor to prepare data.
Response signal	The sensor pulls the data bus (SDA) low for 83µs, and then pulls up for 87µs to respond to the host's start signal.
Humidity	The high humidity is an integer part of the humidity data, and the low humidity is a fractional part of the humidity data.
Temperature	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␣
↪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 in the figure: media/933ac5e5a5e921d4b16c7c48091ba75a.png

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:

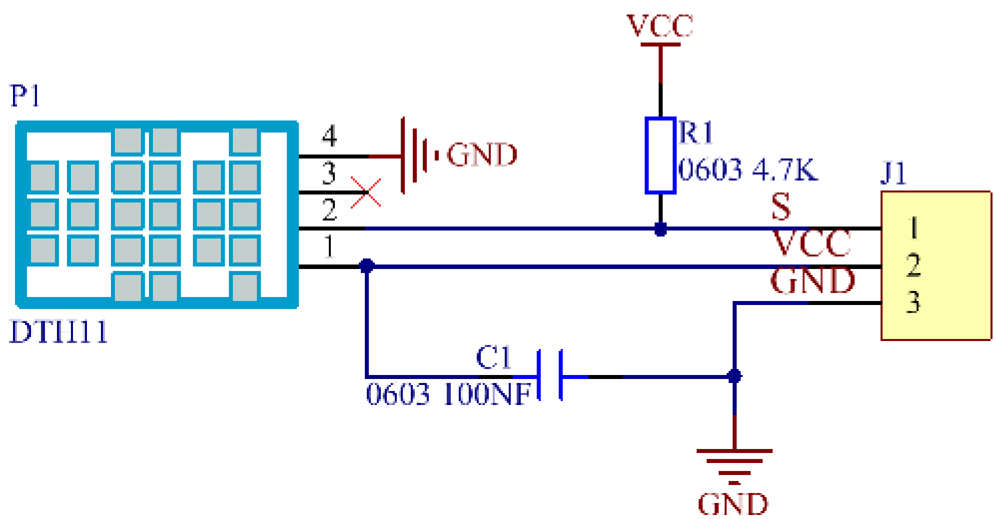
Working voltage: +5V

Temperature range: 0°C to 50°C, error of ± 2°C

Humidity range: 20% to 90% RH,± 5% RH error

Digital interface

Schematic diagram of Temperature and Humidity Sensor:



9.30.4 Read temperature and humidity value

media/5d6dd3f19b4323d212bb95e3e4d43743.png

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

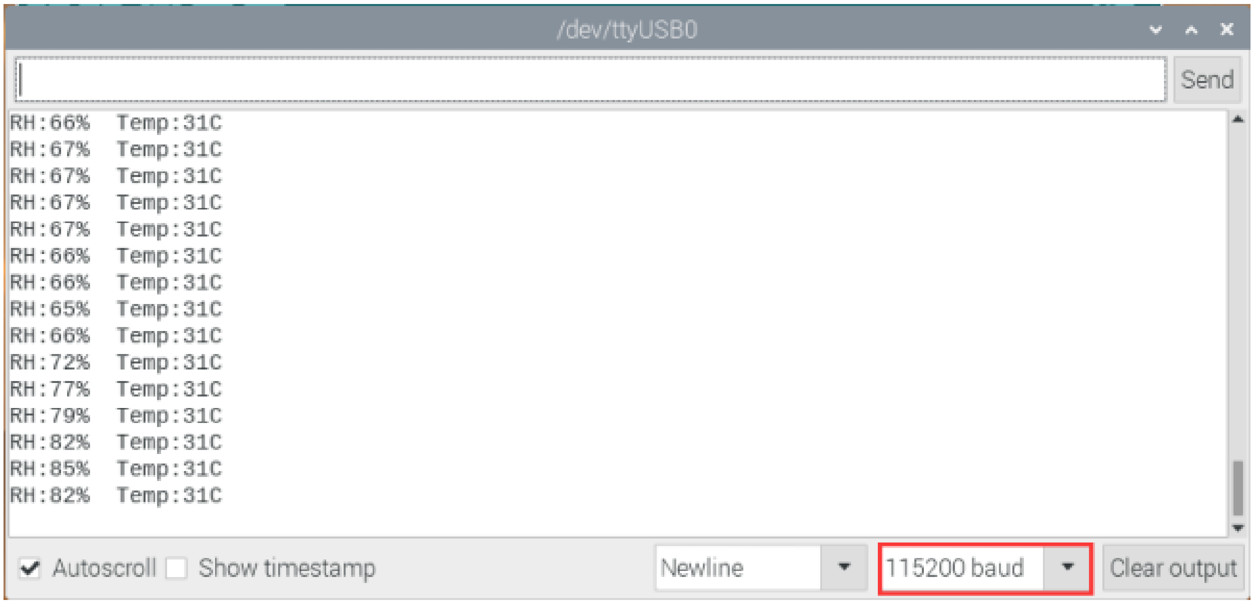
This code uses a library named "**xht11**", if you haven't installed it yet, please do so before learning. The steps to add third-party libraries are as follows:

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␣
↪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␣
↪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
}
//**********************************************************************************
```

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:



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"
```

(continues on next page)

(continued from previous page)

```
//gpio13
xht11 xht(13);
unsigned char dht[4] = {0, 0, 0, 0};//Only the first 32 bits of data are received, not␣
↪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);
}
//*************************************************************************************
```

9.30.6 Project result

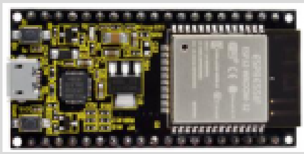
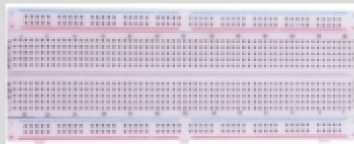
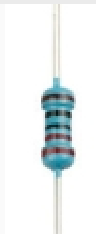
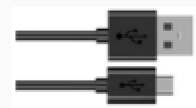
Compile and upload the code to ESP32, after the code is uploaded successfully, power up with a USB cable and you will see that the LCD 128X32 DOT will display temperature and humidity value in the current environment.

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

			
ESP32*1	Breadboard*1	Ultrasonic Sensor*1	Red LED*4
			
M-F Dupont Wires	220Resistor*4	Jumper Wires	USB Cable*1

9.31.3 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.

media/e6f6037071e434febf7090b56ac35802.png

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.

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.

media/db430baa07e2e4d9ac9efca1950b953a.jpeg

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"**.

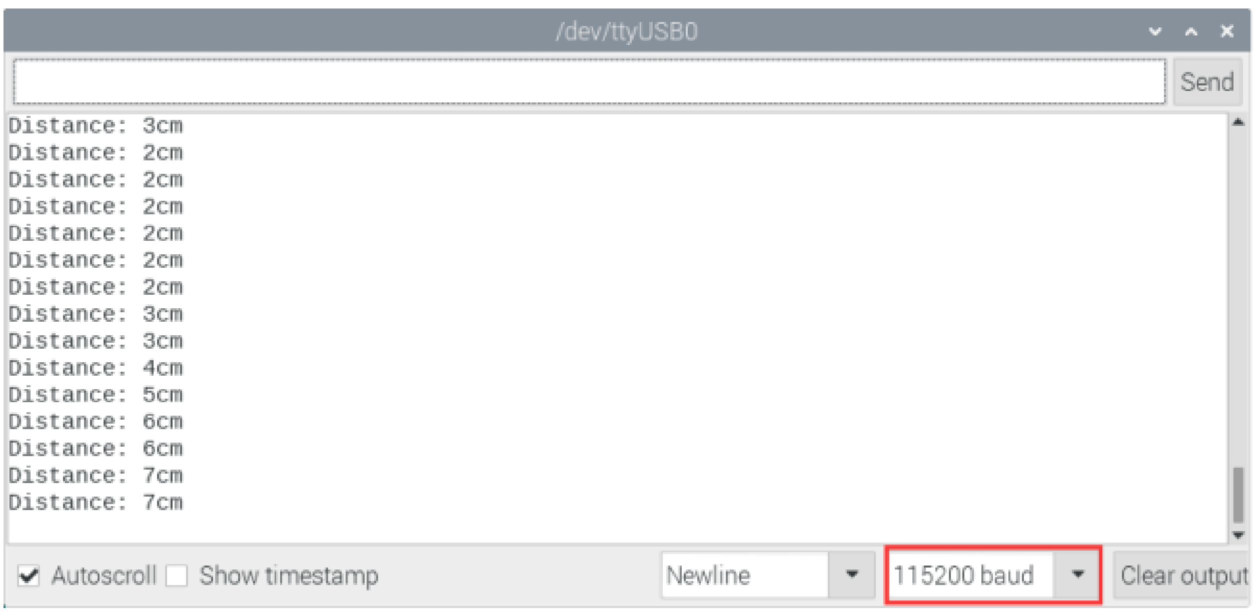
```
//**********************************************************************************
/*
 * Filename    : Ultrasonic Ranging
 * Description : Use the ultrasonic module to measure the distance.
 * Auther      : http//www.keyestudio.com
*/
const int TrigPin = 13; // define TrigPin
const int EchoPin = 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
```

(continues on next page)

(continued from previous page)

```
  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.



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.
 * 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
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
```

(continues on next page)

(continued from previous page)

```
  distance = (duration/2) / 28.5 ;
  Serial.print("Distance: ");
  Serial.print(distance); //Serial port print distance value
  Serial.println("cm");
  if ( distance <= 5 )
  {
    digitalWrite(PIN_LED1, HIGH);
  }
  else
  {
    digitalWrite(PIN_LED1, LOW);
  }
  if ( distance <= 10 )
  {
    digitalWrite(PIN_LED2, HIGH);
  }
  else
  {
    digitalWrite(PIN_LED2, LOW);
  }
  if ( distance <= 15 )
  {
    digitalWrite(PIN_LED3, HIGH);
  }
  else
  {
    digitalWrite(PIN_LED3, LOW);
  }
  if ( distance <= 20 )
  {
    digitalWrite(PIN_LED4, HIGH);
  }
  else
  {
    digitalWrite(PIN_LED4, LOW);
  }
}
//*********************************************************************************
```

9.31.7 Project result

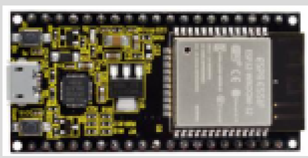
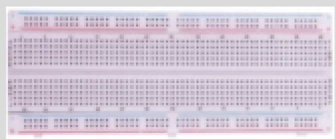
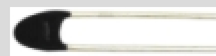
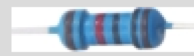
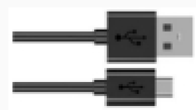
Compile and upload the code to ESP32, after the code is uploaded successfully, power up with a USB cable, open the serial monitor and set the baud rate to 115200. You will see that the serial port monitor window will print outthe distance between the ultrasonic sensor and the object, and the corresponding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

			
ESP32*1	Breadboard*1	Thermistor*1	10KResistor*1
			
M-F Dupont Wires	LCD 128X32 DOT*1	Jumper Wires	USB Cable*1

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:

media/809b8634747fb295021f12e3b92b7894.png

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

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:

$$T2 = 1/\left(\frac{1}{T1} + \ln\left(\frac{Rt}{R}\right)/B\right)$$

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

media/806fd81bf8a761b4ae1a638489c426ce.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 31Temperature Instrument\Project_31.1_Read_the_thermistor_analog_value"**.

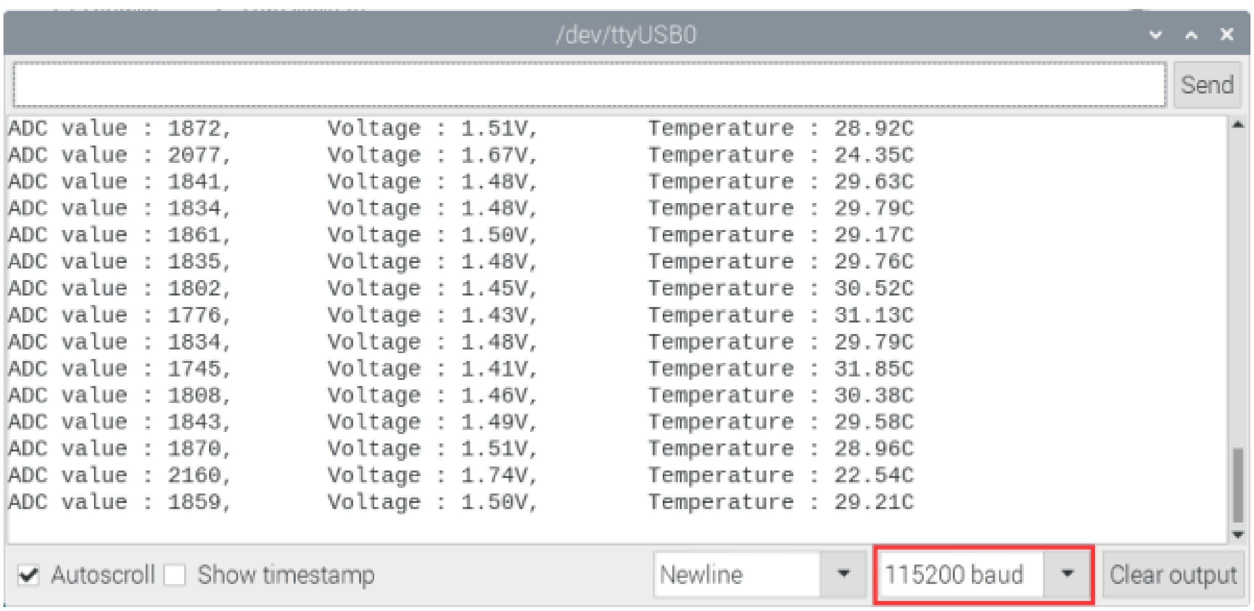
```
//**********************************************************************************
/*
 * Filename    : Thermomter
```

(continues on next page)

(continued from previous page)

```
 * Description : Making a thermometer by thermistor.
 * Auther      : http//www.keyestudio.com
*/
#define PIN_ANALOG_IN   36
void setup() {
  Serial.begin(115200);
}

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 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 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.



9.32.5 Wiring diagram of the temperature instrument

media/5a437bfdcad012211e15cab54e35dad7.png

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.
 * 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␣
↪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
```

(continues on next page)

(continued from previous page)

```
  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.Display("tempC:");
  lcd.Cursor(2,8);
  lcd.DisplayNum(tempC);
  lcd.Cursor(2,11);
  lcd.Display("C");
  delay(200);
}
//*************************************************************************************
```

9.32.8 Project result

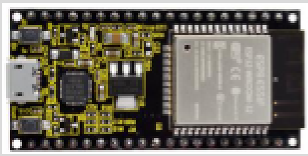
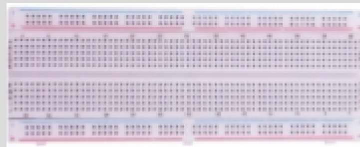
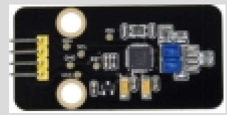
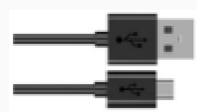
Compile and upload the code to ESP32, after the code is uploaded successfully, power up with a USB cable and you will see that the LCD 128X32 DOT displays the voltage value of the thermistor and the temperature value in the current environment.

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

		
ESP32*1	Breadboard*1	RFID-RC522 Module*1
		
M-F Dupont Wires	Servo*1	White Card*1
		
Key Chain*1	Jumper wires	USB Cable*1

9.33.3 Component knowledge

RFIDRFID (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"**.

```
//**********************************************************************************
/*
 * Filename    : RFID
 * Description : RFID reader UID
```

(continues on next page)

(continued from previous page)

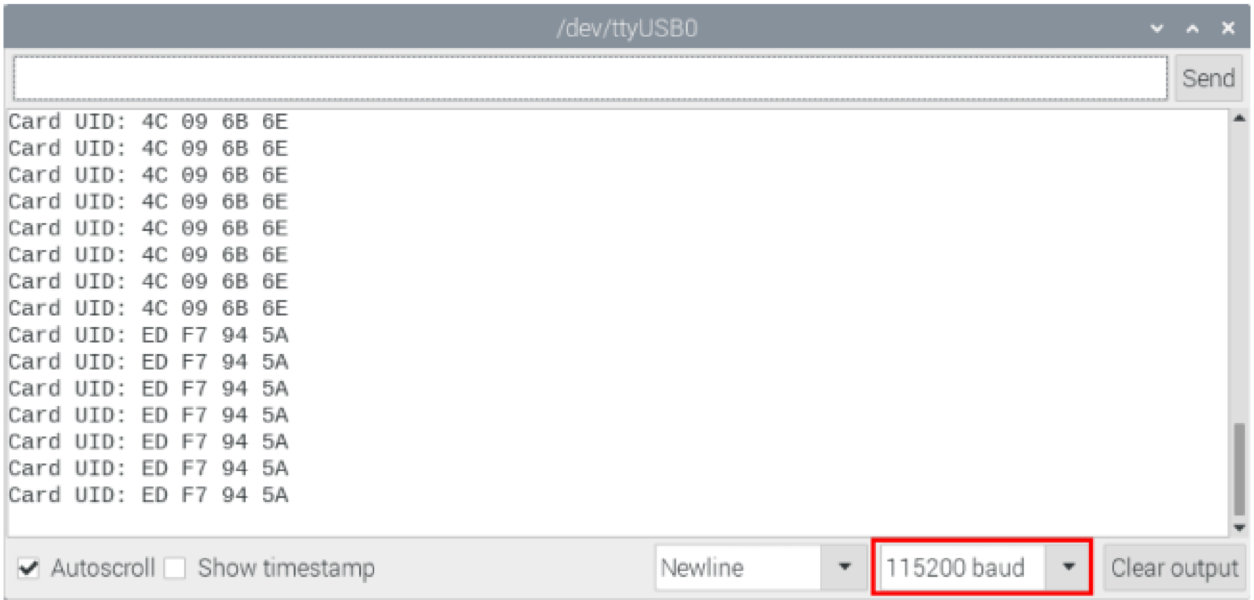
```
 * Auther      : http//www.keyestudio.com
*/
#include <Wire.h>
#include "MFRC522_I2C.h"
// IIC pins default to GPIO21 and GPIO22 of ESP32
// 0x28 is the i2c address of SDA, if doesn't matchplease check your address with i2c.
MFRC522 mfrc522(0x28);   // create MFRC522.

void setup() {
  Serial.begin(115200);           // initialize and PC's serial communication
  Wire.begin();                   // initialize I2C
  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() {
  //
  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" : " ");
    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?
↪"));
  }
}
```

(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, 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:



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.

For example: You can replace the rfid_str of the media/85170ea6c47d9350dd05efa60be8c808.png 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 GPIO21 and GPIO22 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);
```

(continues on next page)

(continued from previous page)

```
    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++) {
    rfid_str = rfid_str + String(mfrc522.uid.uidByte[i], HEX);  //Convert to string
    //Serial.print(mfrc522.uid.uidByte[i] < 0x10 ? " 0" : " ");
    //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

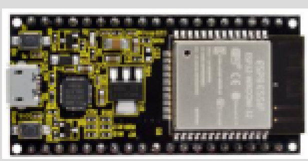
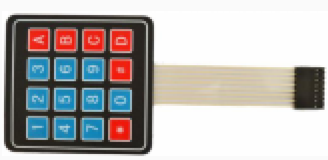
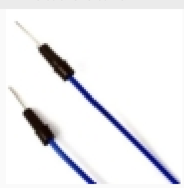
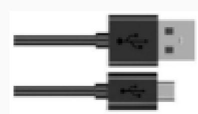
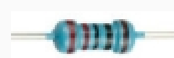
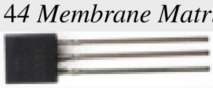
Compile and upload the code to ESP32, after the code is uploaded successfully, power up with a USB cable, and open the serial monitor and then set baud rate to 115200. You need to press the reset button on the ESP32 mainboard first, and then you will see that when using the white card or a key card swiping, 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

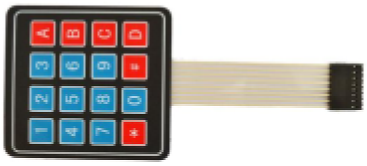
Commonly used digital button sensor, one button uses an IO port. However, it will occupy too many IO ports when we need a lot of buttons. In order to save the use of IO ports, the multiple buttons are made into a matrix type, through the control of the line and row to achieve less IO port control of 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

			
ESP32*1	Breadboard*1	Servo*1	Active Buzzer*1
			
44 *Membrane Matrix Keyboard*1	Jumper Wires	USB Cable*1	1kResistor*1
			
NPN transistor(S8050)*1			

9.34.3 Component knowledge

4*4 Matrix keyboardA 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.

media/5ebdacba906622079e0ef41dc1ea3fdf.png

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

media/8bfa0e1b1a0f53598f51341d51bc7601.png

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:Keypad\DoorProject_33.1_4x4_Matrix_Keypad_Display"**.

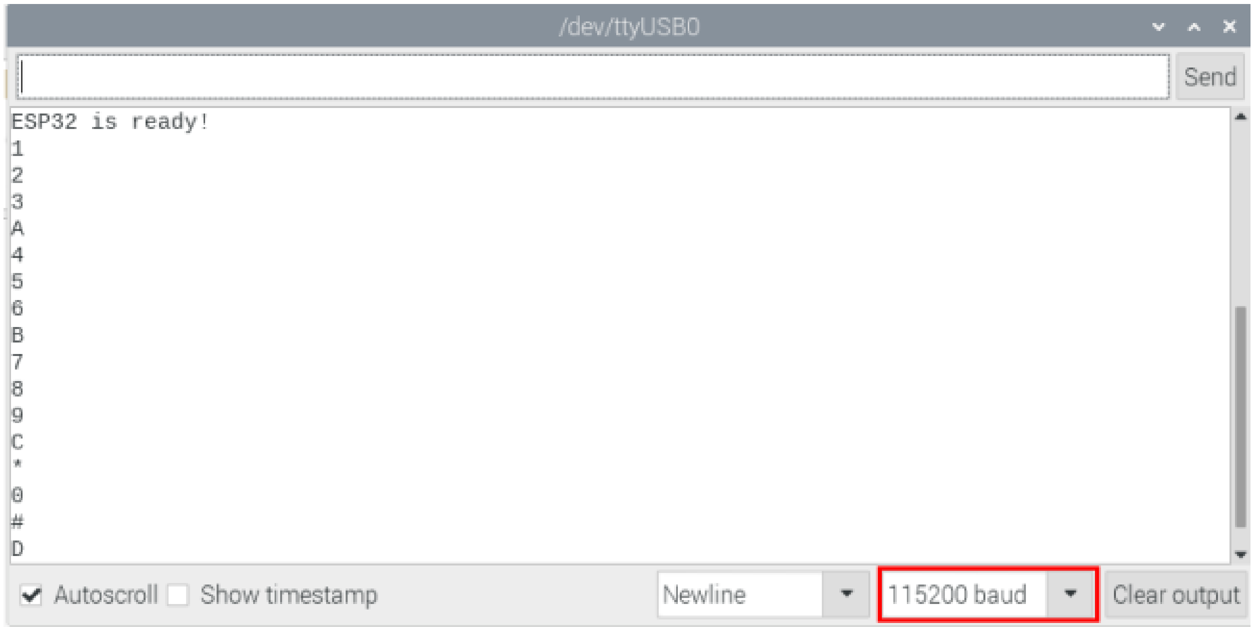
```
//**********************************************************************************
/*
 * Filename    : 4x4 Matrix Keypad Display
 * Description : Get the value for the matrix keyboard
 * Auther      : http//www.keyestudio.com
```

(continues on next page)

(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 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.



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"**.

```
//**********************************************************************************
/*
 * Filename    : Keypad_Door
 * Description : Make a simple combination lock.
 * Auther      : http//www.keyestudio.com
*/
```

(continues on next page)

(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
        if (keyIn[i] != passWord[i])
```

(continues on next page)

(continued from previous page)

```
          isRight = false;            // Mark wrong passageword if there is any wrong␣
↪character.
      }
      if (isRight) {                  // If the input password is right
        myservo.write(90);           // Open the switch
        delay(2000);                  // Delay a period of time
        myservo.write(0);            // Close the switch
        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.

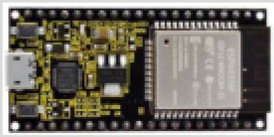
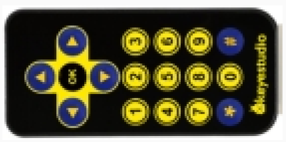
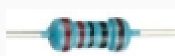
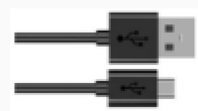
9.35 Project 34IR Control Sound and LED

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
			
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/3c9d76cb0d24d9861811ce2cb0bb6ae4.png

Infrared receiverreceiver 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 media/3da1969e509f53706643c77d0534eb73.png

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:

media/da33571c6f0afb94b1ec1d91caba3edb.png

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:

media/f970404e7bbfb5711fea5c776f689f3a.png

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.

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.

media/2f78d1ce7f001926f6b90ad966796e91.png

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:

media/700496cb1e0d5d23fd63c28469dd3fd0.png

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␣
↪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("");
    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 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.

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

media/4912c1622e0eaedb76ea3a9b8ed969c0.png

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"**.

```
//**********************************************************************************
/*
 * Filename    : IR Control Sound And LED
 * Description : Remote control RGB and Passive buzzer with infrared remote control.
```

(continues on next page)

(continued from previous page)

```
 * 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
    ledcSetup(chns[i], 1000, 8);
    ledcAttachPin(ledPins[i], chns[i]);
  }
}

void loop() {
  if(irrecv.decode(&results)) {        // Waiting for decoding
    handleControl(results.value);       // Handle the commands from remote control
    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'
```

(continues on next page)

(continued from previous page)

```
    {
      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.)

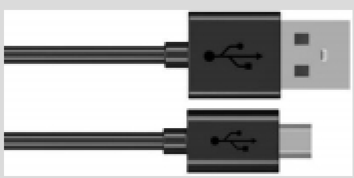
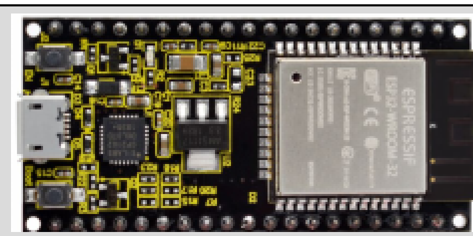
media/3c9d76cb0d24d9861811ce2cb0bb6ae4.png

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

	
USB Cable*1	ESP32*1

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

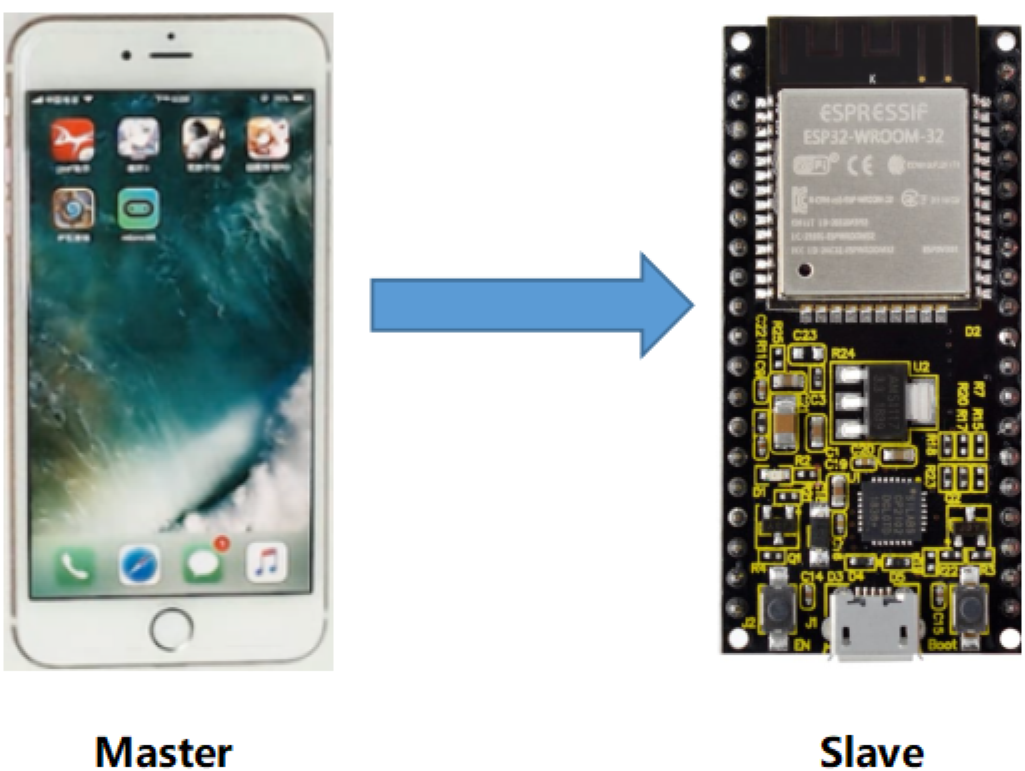
In this mode, works are done in the master device and it can connect with a slave device. And we can search and select slave devices nearby to connect with. When a device initiates connection request in master mode, it requires information of the other Bluetooth devices including their address and pairing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

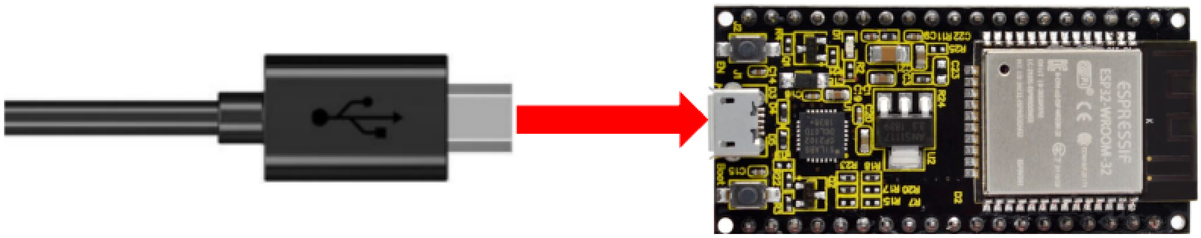
exchange data with ESP32, they are usually in master mode and ESP32 in slave mode.



Master Slave

9.37.3 Wiring Diagram

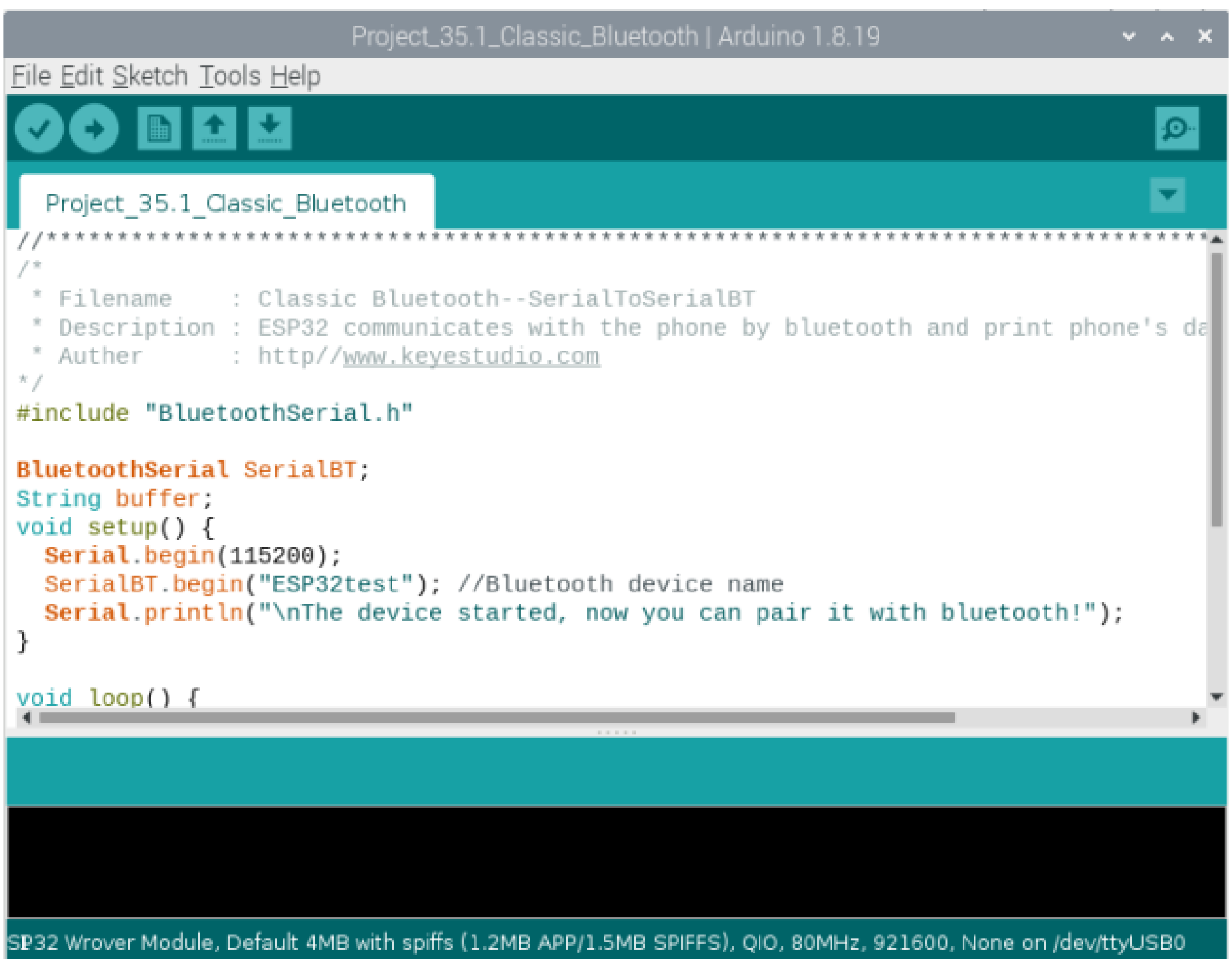
We can use a USB cable to connect ESP32 mainboard to the USB port on the Raspberry Pi.



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:luetooth\Project_35.1_Classic_Bluetooth"**.

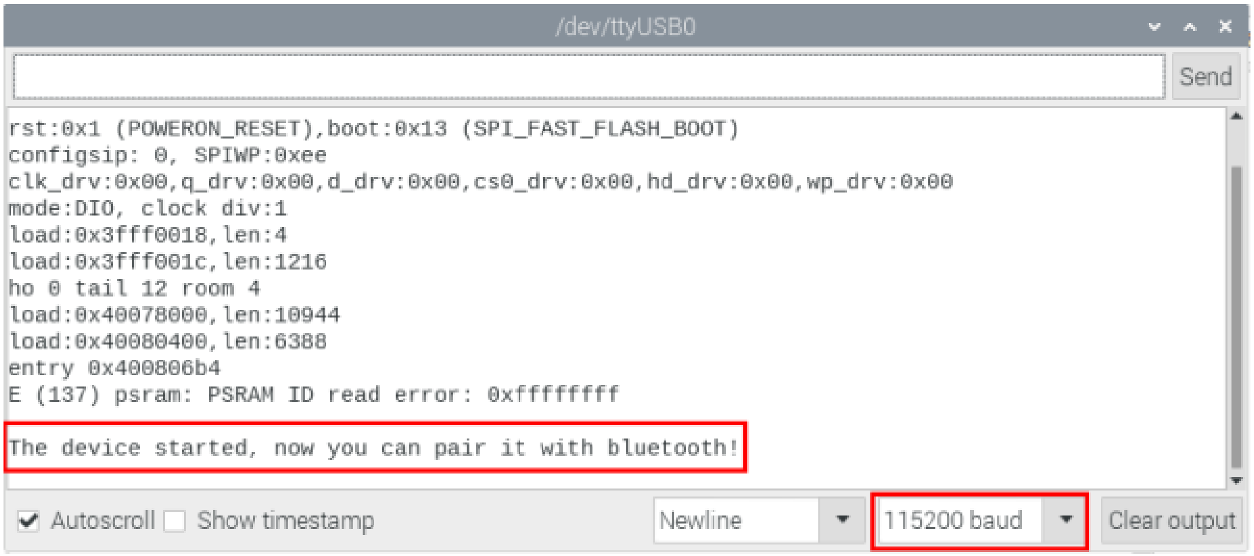


```
//**********************************************************************************
/*
 * Filename    : Classic Bluetooth--SerialToSerialBT
 * Description : ESP32 communicates with the phone by bluetooth and print phone's data via a serial port
 * Auther      : http//www.keyestudio.com
*/
#include "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



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

media/382529edef3989e60264cad217d88e6f.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**" .

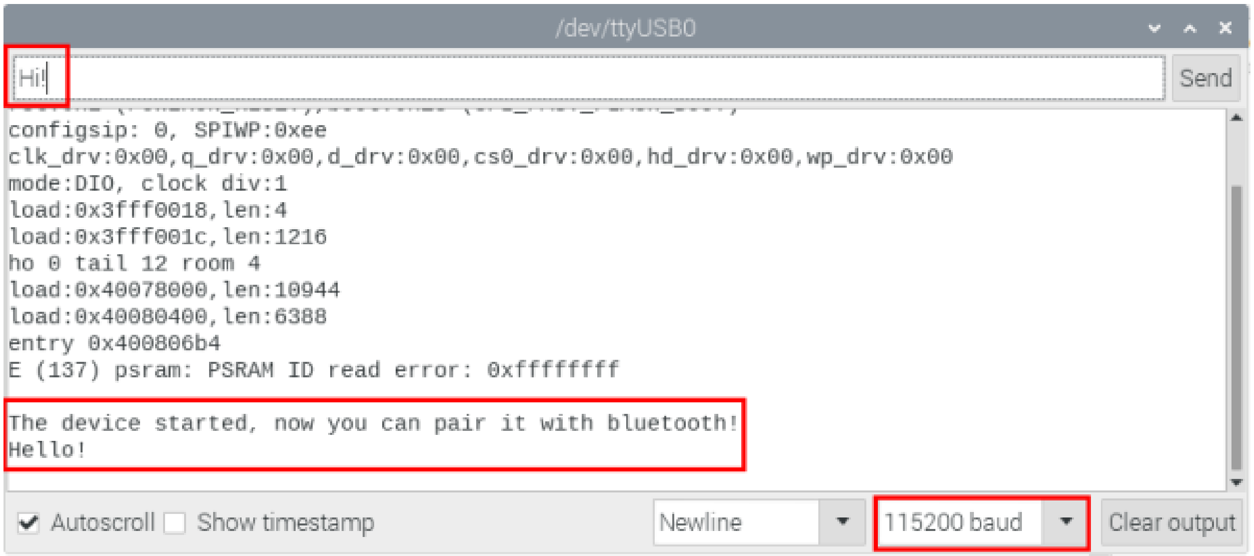
media/32b8c3abd51fc538ba854b1d72e1165e.png

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.

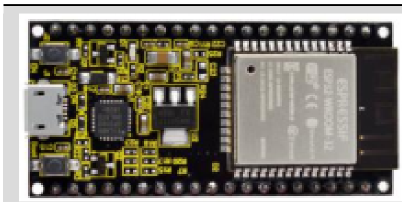
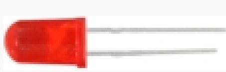
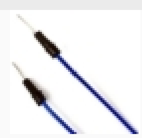
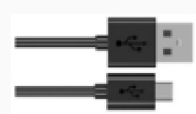
Send "**Hello!**" from your phone, when the computer receives it, reply "**Hi!**" to your phone.



media/4f4e6b4e45996ccbde4da17219f02d00.png

9.38 Project 35.2Bluetooth Control LED

9.38.1 Components

			
ESP32*1	Breadboard*1		
			
Red LED*1	220Resistor*1	Jumper Wires	USB Cable*1

9.38.2 Wiring diagram

media/0735997593c8858ad6441d8e9867206f.png

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:Bluetooth\Project_35.2_Bluetooth_Control_LED"**.


```
//**********************************************************************************
/*
 * Filename    : Bluetooth Control LED
 * Description : The phone controls esp32's led via bluetooth.
```

(continues on next page)

(continued from previous page)

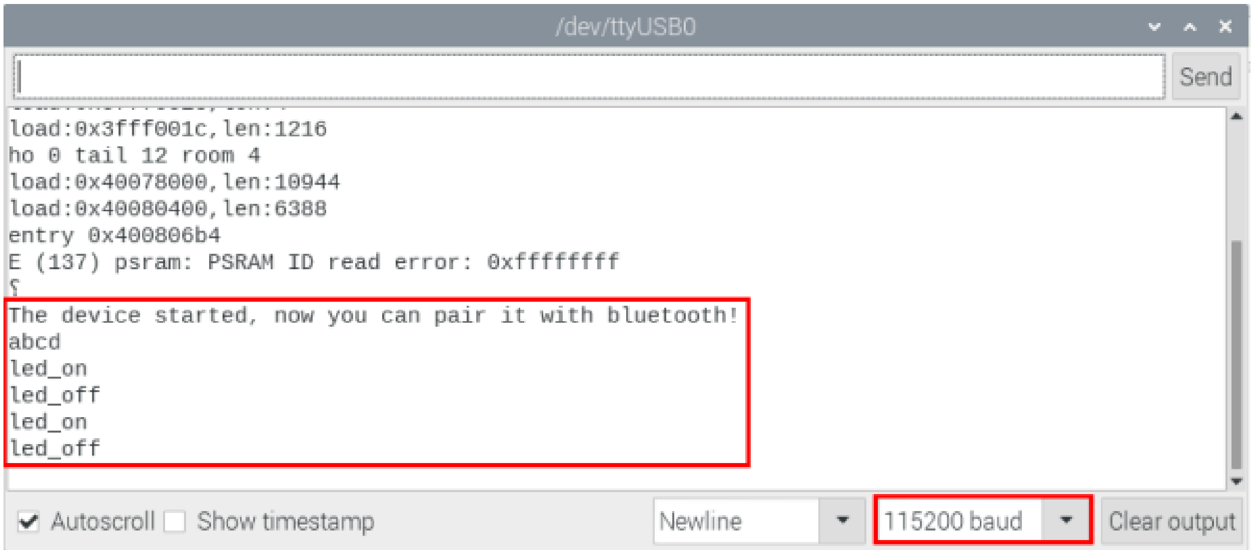
```
                When the phone sends "LED_on," ESP32's LED lights turn on.
                When the phone sends "LED_off," ESP32's LED lights turn off.
 * Auther      : http//www.keyestudio.com
*/
#include "BluetoothSerial.h"
#include "string.h"
#define LED 15
BluetoothSerial SerialBT;
char buffer[20];
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/21ec63e3abe43a119ab8a3d4634894f0.png

Display on the serial port of the computer:



The phenomenon of LED

media/a1e66d46e5797995a1b66a761c7857f8.png

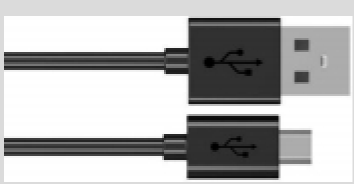
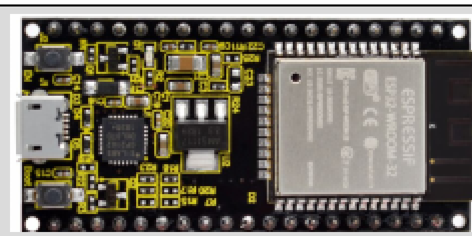
Attention: If the sending content isn't "led-on" or"led-off", then the state of LED will not change. If the LED is on, when receiving irrelevant content, it keeps on; Correspondingly, if the LED is off, when receiving irrelevant content, it keeps off.

9.39 Project 36WiFi Station Mode

9.39.1 Introduction

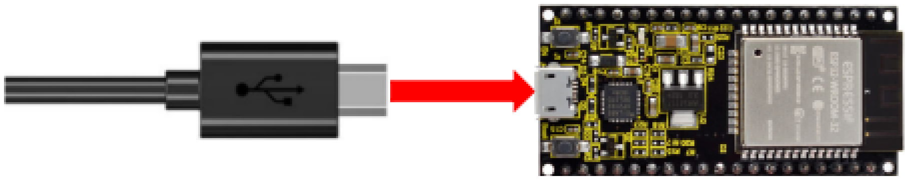
ESP32 has three different WiFi operating modes : Station modeAP mode and AP+Station mode. All WiFi programming projects must be configured with WiFi operating mode before using WiFi, otherwise WiFi cannot be used. In this project, we will learn about ESP32's WiFi Station mode.

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"**.

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.


```
//**********************************************************************************
/*
 * Filename    : WiFi Station
 * Description : Connect to your router using ESP32
 * Auther      : http//www.keyestudio.com
*/
#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␣
↪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);
```

(continues on next page)

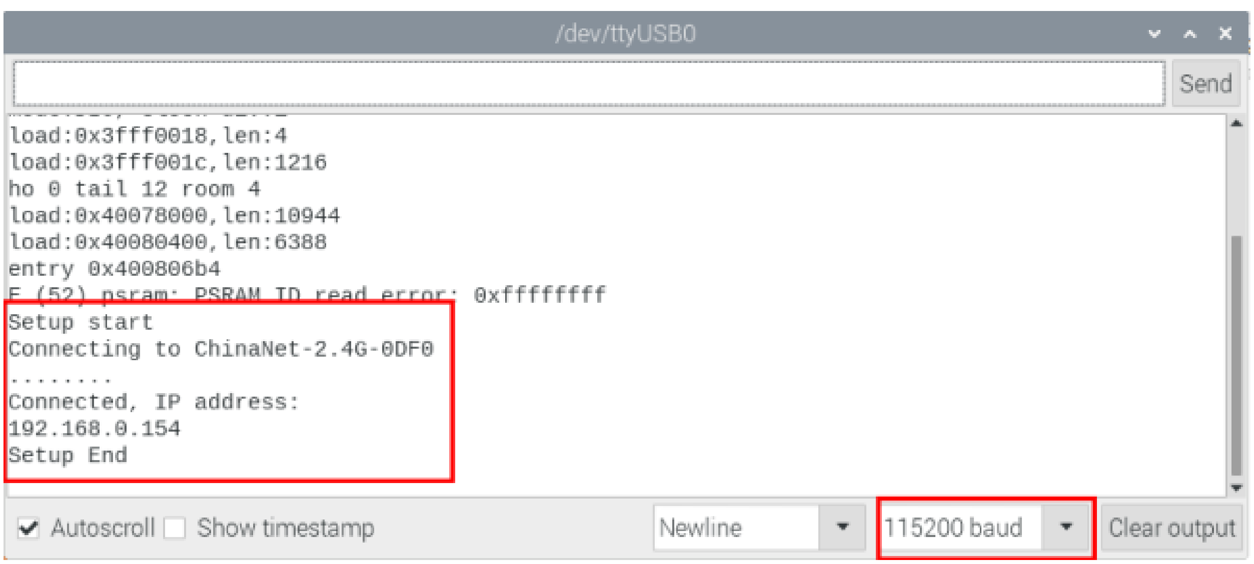
(continued from previous page)

```
    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() {
}
//*************************************************************************************
```

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

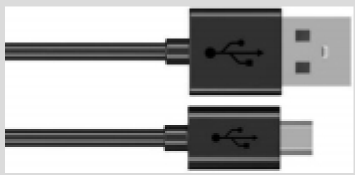
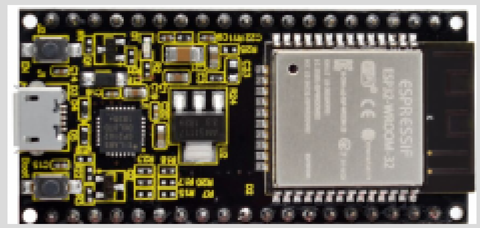


9.40 Project 37WiFi AP Mode

9.40.1 Introduction

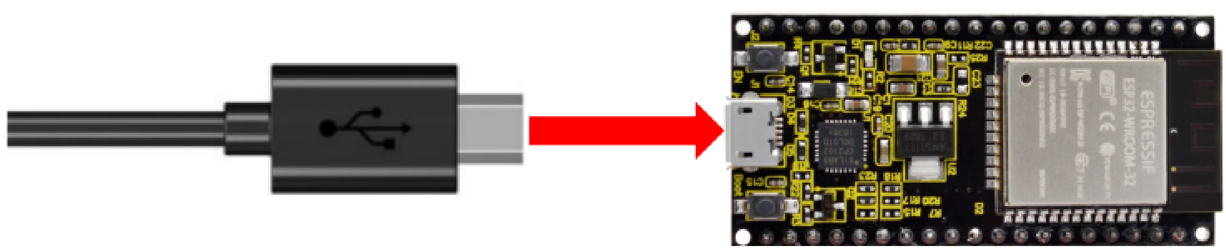
ESP32 has three different WiFi operating modes : Station modeAP mode and AP+Station mode. All WiFi programming projects must be configured with WiFi operating mode before using WiFi, otherwise WiFi cannot be used. In this project, we will learn about ESP32's WiFi AP mode.

9.40.2 Components

	
USB Cable*1	ESP32*1

9.40.3 Wiring

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



9.40.4 Component knowledge

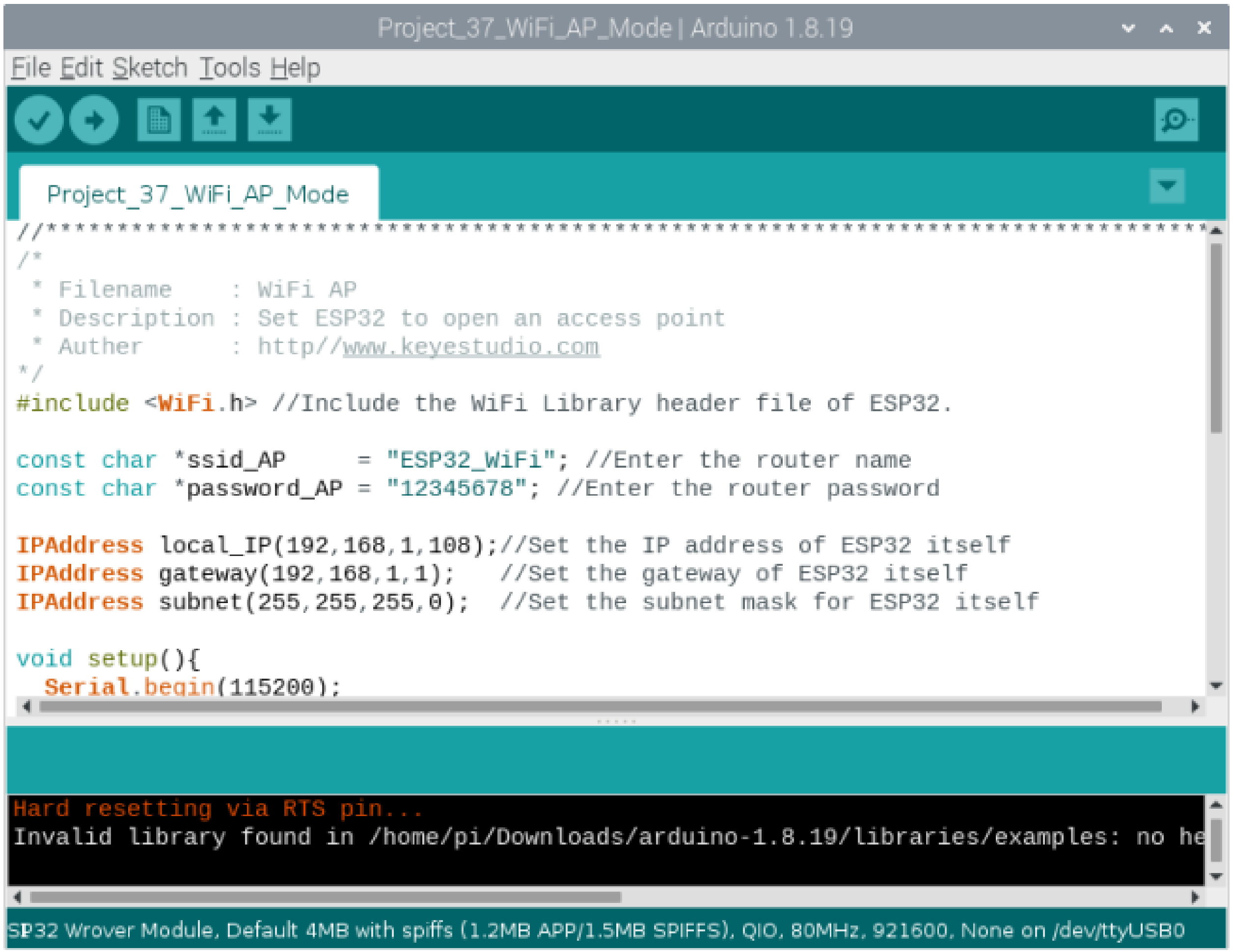
AP mode : When ESP32 selects AP mode, it creates a hotspot network that is separated from the Internet and waits for other WiFi devices to connect. As shown in the figure below, ESP32 is used as a hotspot. If a mobile phone or PC wants to communicate with ESP32, it must be connected to the hotspot of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


```
//**********************************************************************************
/*
 * 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);
  Serial.println("Setting soft-AP configuration ... ");
  WiFi.disconnect();
  WiFi.mode(WIFI_AP);
  Serial.println(WiFi.softAPConfig(local_IP, gateway, subnet) ? "Ready" : "Failed!");
```

(continues on next page)

(continued from previous page)

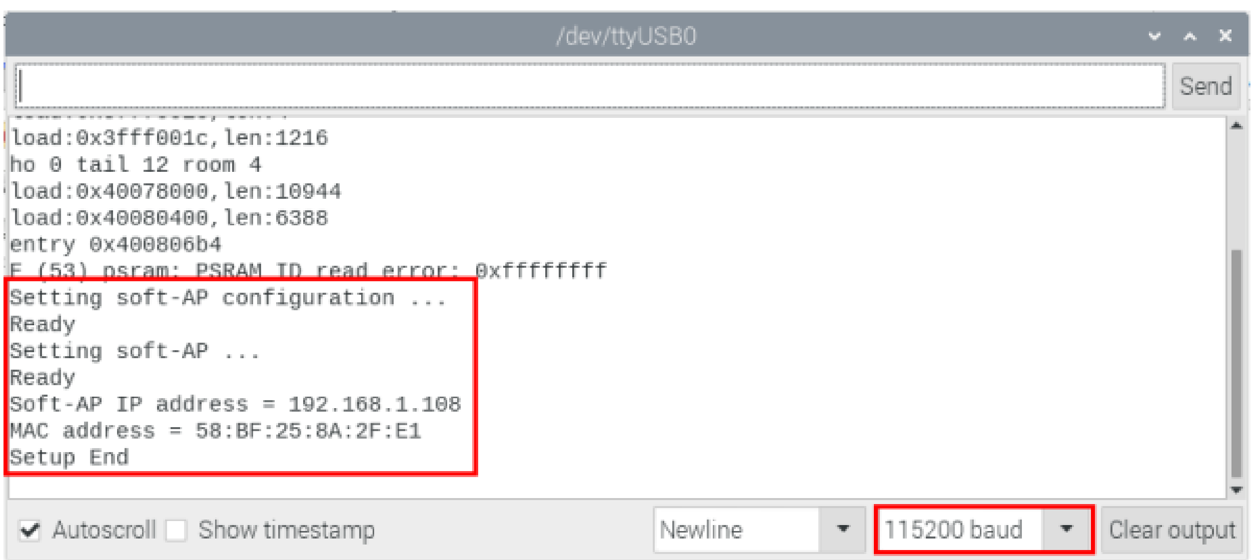
```
  Serial.println("Setting soft-AP ... ");
  boolean result = WiFi.softAP(ssid_AP, password_AP);
  if(result){
    Serial.println("Ready");
    Serial.println(String("Soft-AP IP address = ") + WiFi.softAPIP().toString());
    Serial.println(String("MAC address = ") + WiFi.softAPmacAddress().c_str());
  }else{
    Serial.println("Failed!");
  }
  Serial.println("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



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.

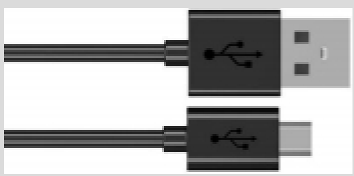
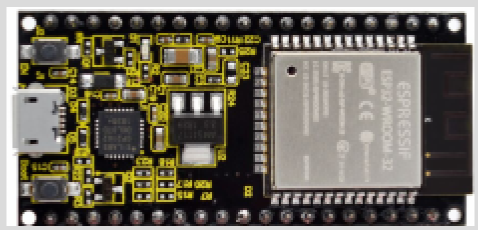
media/3e0ad895bea7f5100cc02a415adcace7.png

9.41 Project 38WiFi Station+AP Mode

9.41.1 Introduction

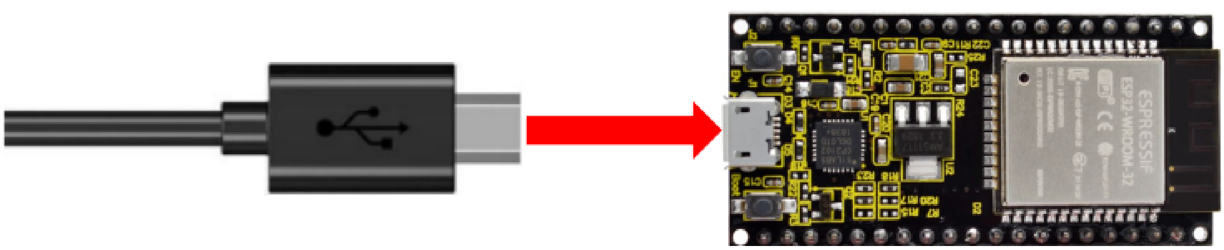
ESP32 has three different WiFi operating modes : Station modeAP mode and AP+Station mode. All WiFi programming projects must be configured with WiFi operating mode before using WiFi, otherwise WiFi cannot be used. In this project, we will learn ESP32's WiFi Station+AP mode.

9.41.2 Components

	
USB Cable*1	ESP32*1

9.41.3 Wiring

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



9.41.4 Component knowledge

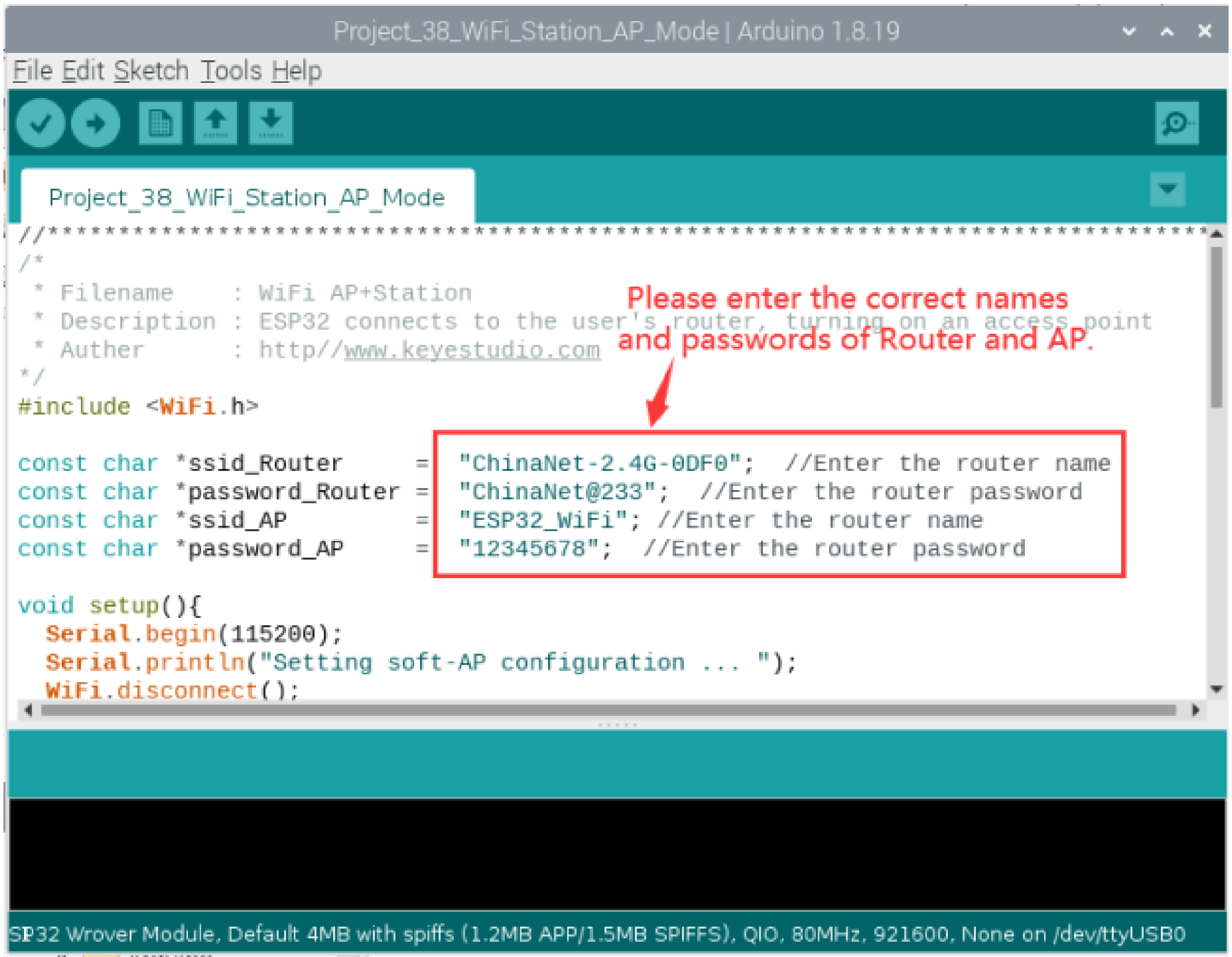
AP+Station mode: In addition to AP mode and Station mode, ESP32 can also use AP mode and Station mode at the same time. This mode contains the functions of the previous two modes. Turn on ESP32's Station mode, connect it to the router network, and it can communicate with the Internet via the router. 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.




```
//**********************************************************************************
/*
 * 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);
  Serial.println("Setting soft-AP ... ");
  boolean result = WiFi.softAP(ssid_AP, password_AP);
  if(result){
    Serial.println("Ready");
```

(continues on next page)

(continued from previous page)

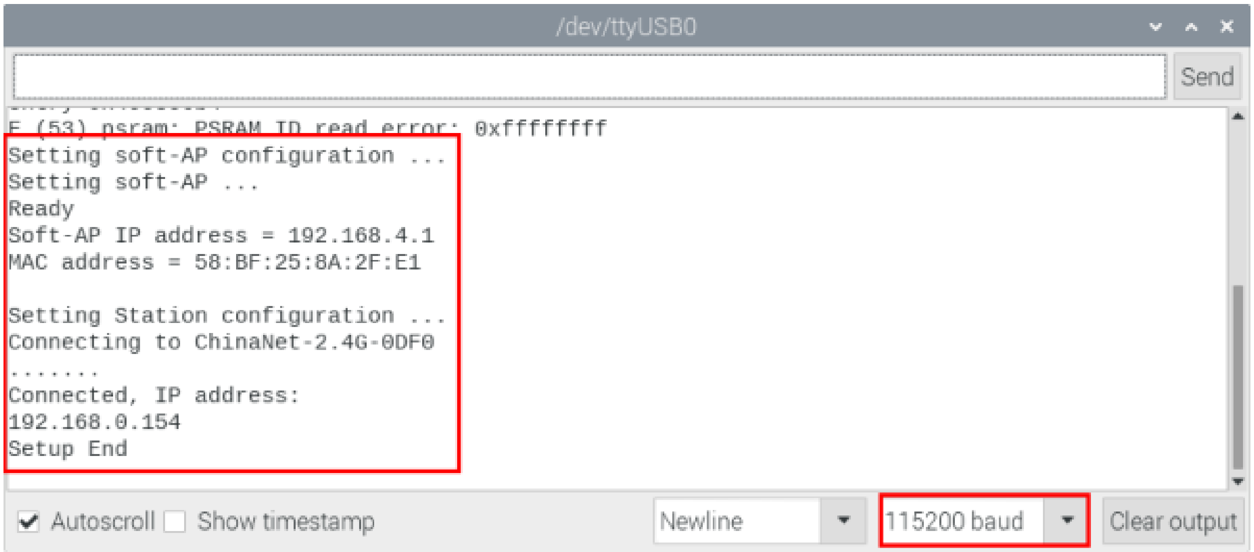
```
    Serial.println(String("Soft-AP IP address = ") + WiFi.softAPIP().toString());
    Serial.println(String("MAC address = ") + WiFi.softAPmacAddress().c_str());
  }else{
    Serial.println("Failed!");
  }

  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



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.

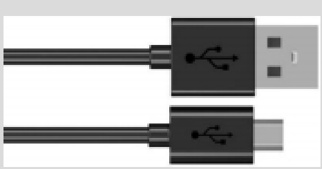
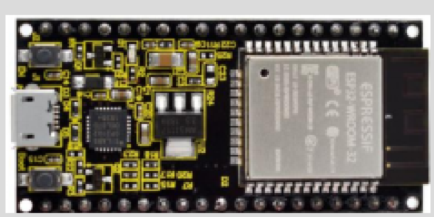
media/3e0ad895bea7f5100cc02a415adcace7.png

9.42 Project 39: WiFi Test

9.42.1 Introduction

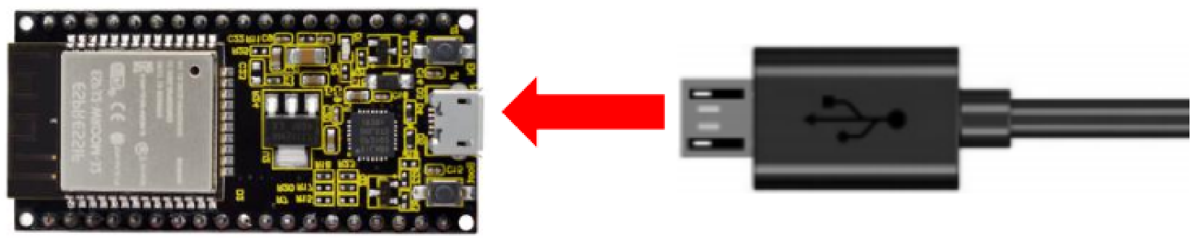
In this experiment, we first use the WiFi station mode of ESP32 to read the IP address of WiFi, and then connect WiFi through app to read the characters sent by each function button on App.

9.42.2 Components

		
USB Cable*1	ESP32*1	Smartphone/Tablet *1

9.42.3 Wiring

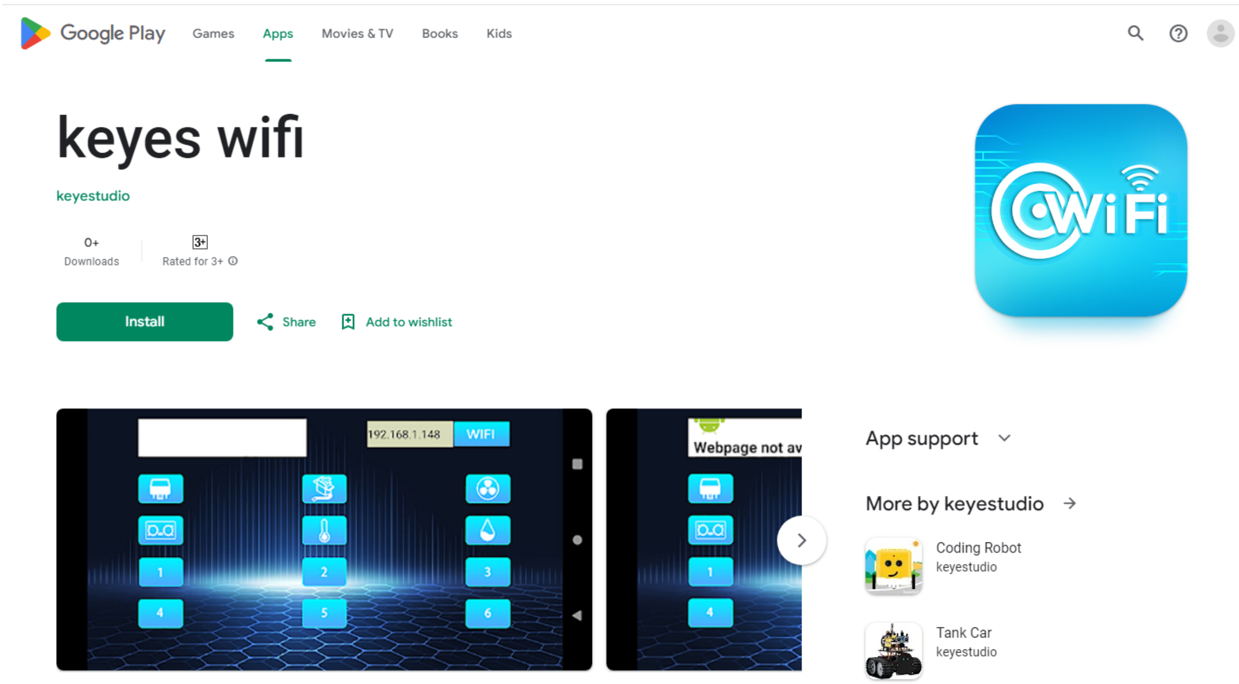
Connect the ESP32 to the USB port on your raspberry pi using a USB cable.



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/d620452a9d6188cb3946269510df5ae0.png media/b311329042f5bbd2880841127b91ebf8.png

media/7c5cfc935371c8e2ab30e999775d5f8f.png

media/d48c065ebaf1c5ca652eb72b15d3e596.png

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“ media/962a57f92b78eea1f0e3e81 ”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: ");
```

(continues on next page)

(continued from previous page)

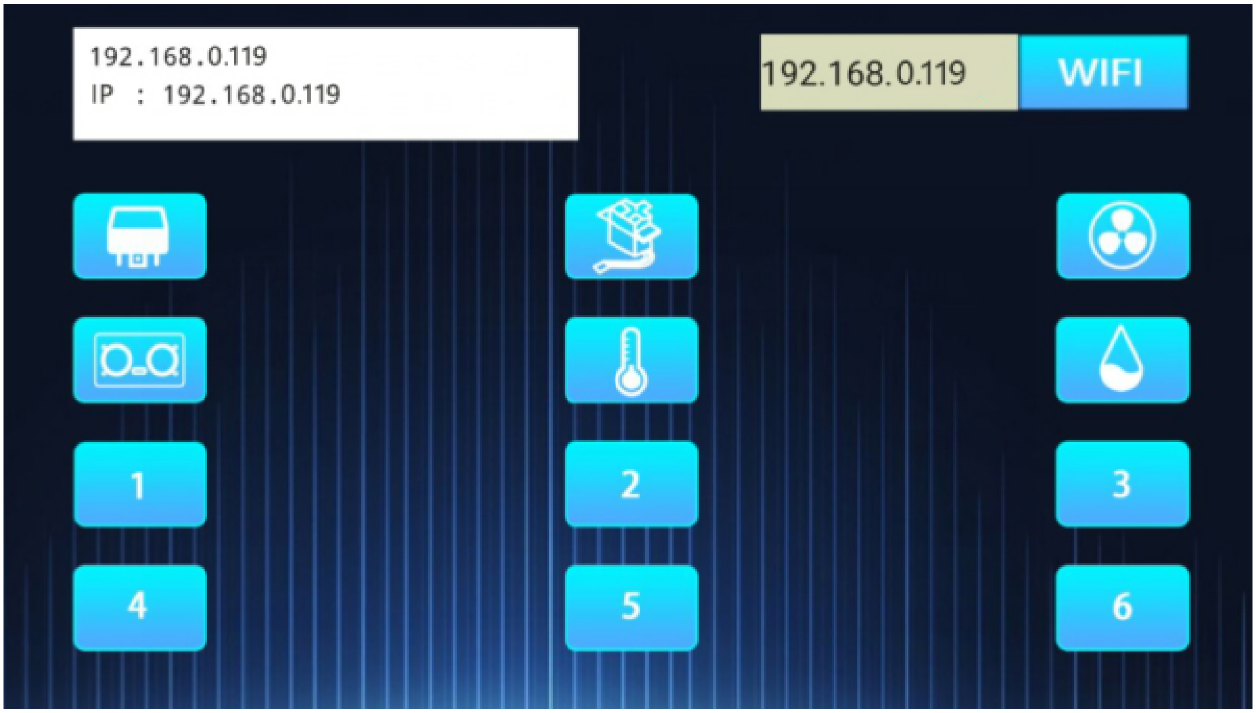
```
    Serial.println(req);
    return;
  }
  req = req.substring(addr_start + 1, addr_end);
  item=req;
  Serial.println(item);
  String s;
  if (req == "/")
  {
    IPAddress ip = WiFi.localIP();
    String ipStr = String(ip[0]) + '.' + String(ip[1]) + '.' + String(ip[2]) + '.' + String(ip[3]);
    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);
  }
  //client.print(s);
  client.stop();
}
//*************************************************************************************
```

Special attention: you need to change the user's Wifi name and Wifi password in the experiment code media/855512855d37d87fc8decadc09f968ec3.png 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 click media/d09c4a31563f04a42d451e7bc1a5fb8a.png, and release the Boot button media/dc77bfcf5851c8f43aab6cbe7cec7920.png after the percentage of uploading progress appears)open the serial monitor and set baud rate to 115200.

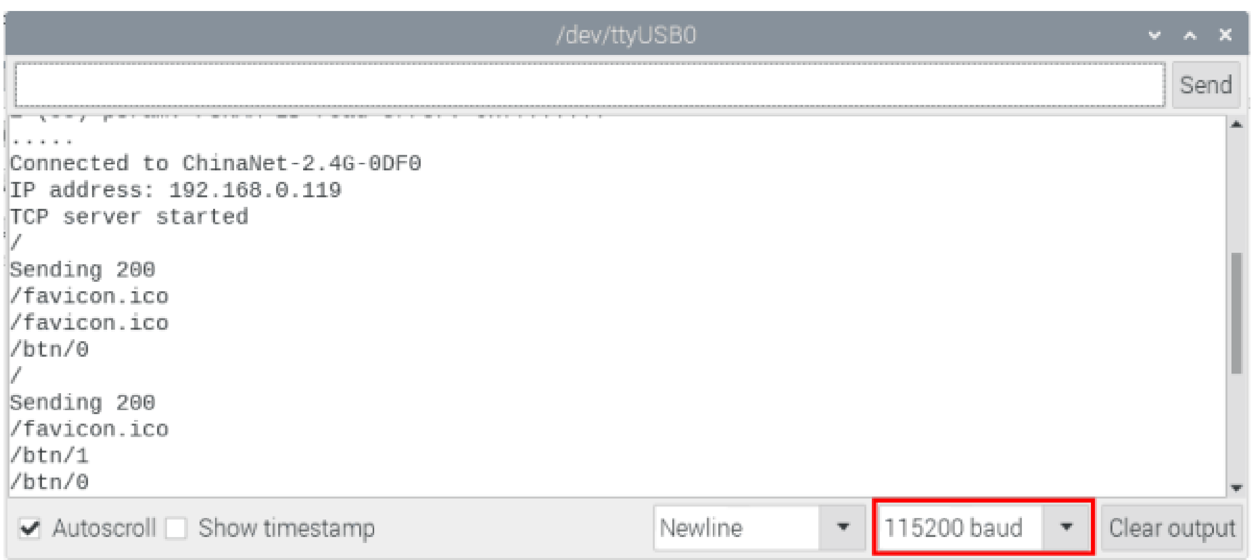
You need to press the reset button on the ESP32 mainboard first.In this way, the serial port monitor prints the detected WiFi IP address, then open the WiFi APP and enter the detected WiFi IP address in the text box in front of the WiFi button (for example, the IP address shown by the serial port monitor 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

)

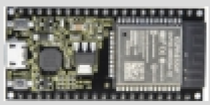
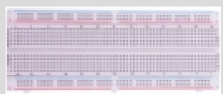
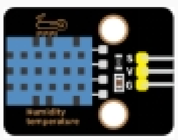
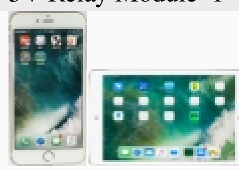
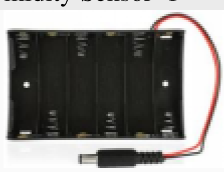
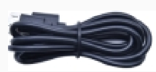
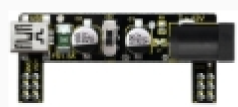
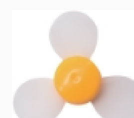


9.43 Project 40WiFi Smart Home

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

				
ESP32*1	Breadboard*1	130 Motor*1	5V Relay Module*1	Servo*1
				
Temperature and Humidity Sensor*1	HC-SR04 Ultrasonic Sensor*1	M-F Dupont Wires	Smartphone/Tablet*1	Jumper Wires
				
Battery Holder*1	Battery(self-provided)*6	USB Cable*1	Keyestudio breadboard special power module*1	Fan*1

9.43.3 Wiring diagram

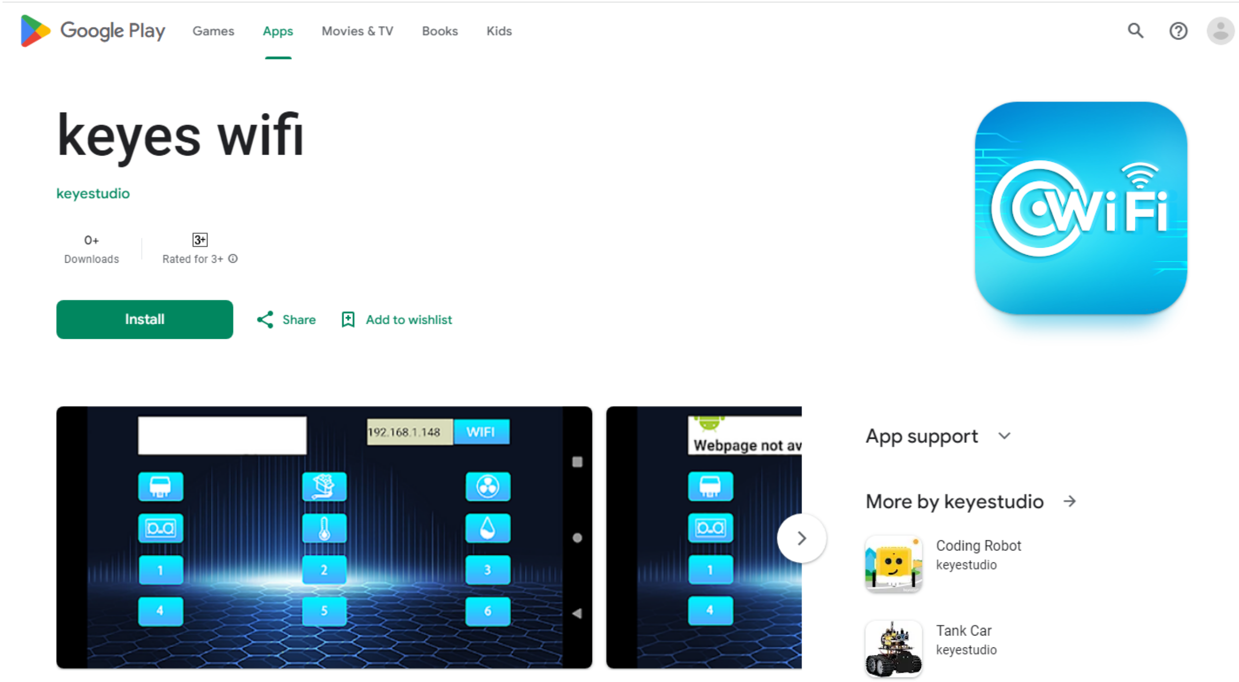
Relay Module	ESP32	Temperature and Humidity Sensor	ESP32
G	G	G	G
V	5V	V	3V3
S	IO32	S	IO15
Ultrasonic Sensor	**ESP32**	**130 Motor**	**ESP32**
Vcc	5V	G	G
Trig	IO14	V	5V
Echo	IO27	IN+	IO19
Gnd	G	IN-	IO18
Servo	**ESP32**		
Red line	5V		
Brown line	G		
Orange line	IO4		

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



Installation steps

media/d620452a9d6188cb3946269510df5ae0.png

media/b311329042f5bbd2880841127b91ebf8.png

media/7c5cfc935371c8e2ab30e999775d5f8f.png

media/d48c065ebaf1c5ca652eb72b15d3e596.png 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“ media/962a57f92b78eea1f0e3e814 ”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.

9.43.5 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”**.

```
//**********************************************************************************
/*
 * Filename    : WiFi Smart Home.
 * Description : WiFi APP controls Multiple sensors/modules work to achieve the effect of WiFi smart home.
 * Auther      : http//www.keyestudio.com
*/
#include <Arduino.h>
#include <WiFi.h>
#include <ESPmDNS.h>
#include <WiFiClient.h>
```

(continues on next page)

(continued from previous page)

```
#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);
```

(continues on next page)

(continued from previous page)

```
  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");
```

(continues on next page)

(continued from previous page)

```
    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;
  }
```

(continues on next page)

(continued from previous page)

```
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;
```

(continues on next page)

(continued from previous page)

```
      }
  }
  else if(req == "/btn/11")
  {
    Serial.write('l');
    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 click media/d09c4a31563f04a42d451e7bc1a5fb8a.png, and release the Boot button media/dc77bfcf5851c8f43aab6cbe7cec7920.png 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,(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

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:

1) Click media/5b9754cb6ec4f995c9eada1da89a8969.png button, the relay will be opened, the APP will display media/505b00b0e23f6498c5d51d5d775c8fc and the indicator lights up on the module. Click media/5b9754cb6ec4f995c9eada1da89a8969.png again, the relay will be closed, the APP will display

media/deb54a77cdcc87d7569e8b8e46de129f.png and the indicator on the module is off.

2. Click media/c54f78d819d4e6a8310eaeb79ff66910.png buttonthe servo rotates 180°the APP will display media/c54f78d819d4e6a8310eaeb79ff66910.png againthe APP will display media/dee12bee3866542bfe5d70a539f79f0b.png the servo rotates 0°.

3) Click media/5490abf5b2f8a1d9cea3055da07c251c.png buttonthe motorwith small fan bladesrotatesthe APP will display media/5490abf5b2f8a1d9cea3055da07c251c.png againclose the motorthe APP will display media/de6da02ede6d63344546173d36bf5371.png

4) Click media/95bfbe879d2391e4e48dcae085abe5a6.png buttonthe ultrasonic sensor detects the distance, put an object in front of the ultrasonic sensor, the APP will display media/e431e1b9c95bed37b053ae9617f93676.png different distances show different numbers, the distance between the object and the ultrasonic sensor is 6cmclick media/95bfbe879d2391e4e48dcae085abe5a6.png again, turn off the sensor, the APP will display

media/b1df35af68601022e54b7e575b0a07c7.png.

5) Click media/08c8a35841b31fa4b5327fb7b23a7af5.png buttonthe temperature and humidity sensor measures the temperature in the environment, the APP will display media/0aafd0af43f80b692eca3d6732b551b2.png different temperatures show different temperature valuesthe ambient temperature is 30 ° C, click media/08c8a35841b31fa4b5327fb7b23a7af5.png again, turn off the sensorthe APP will display media/82887a1385bc7411ecbdc41f6.

6) Click media/d8e3463ab2f644b3300cdeaa2a68e4c2.png buttonthe temperature and humidity sensor measures the humidity in the environment,the APP will display media/320b86323631e095db330a22ca97bad7.png different humiditys show different humiditys values, the ambient humidity is 55%click media/d8e3463ab2f644b3300cdeaa2a68e4c2.png againturn off the sensor, the APP will display media/adc18d06e626af067286da9040c.