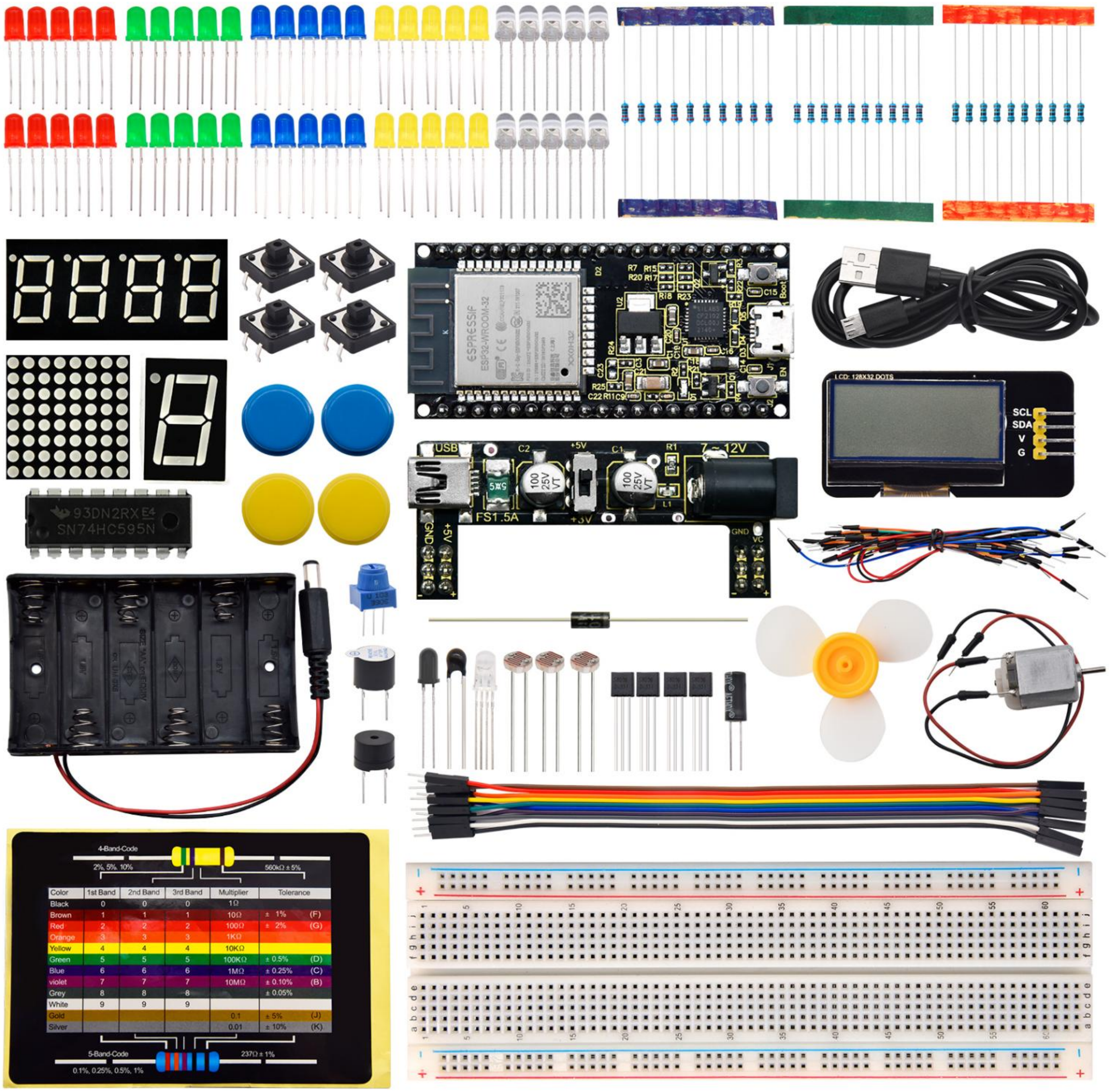
ESP32 Learning Kit

keyestudio WiKi

Dec 06, 2023

KEYESTUDIO ESP32 LEARNING KIT BASIC EDITION



CHAPTER

ONE

1.DESCRIPTION

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 Basic Edition mainly contains some common electronic components/sensors/modules, a ESP32 mainboard and bread wires are also included.

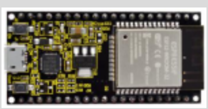
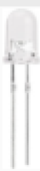
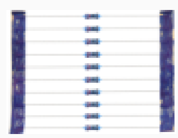
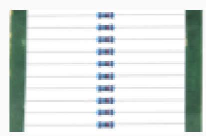
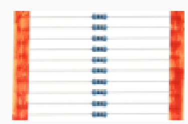
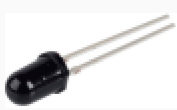
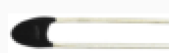
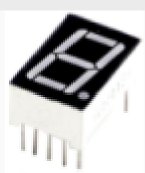
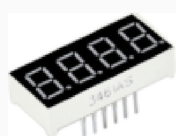
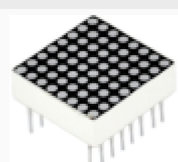
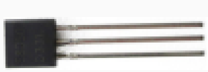
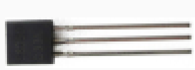
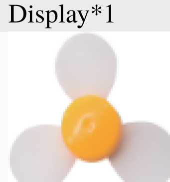
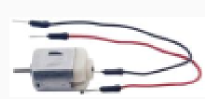
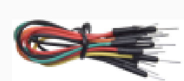
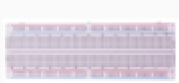
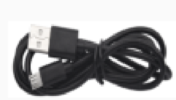
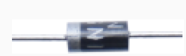
As many as 74 project tutorials are provided, which contain detailed wiring diagrams, components knowledge, and fascinating project code. Each project is produced using Thonny for Windows, Arduino IDE for Windows, and Arduino IDE for Raspberry Pi. It's easy to get started.

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

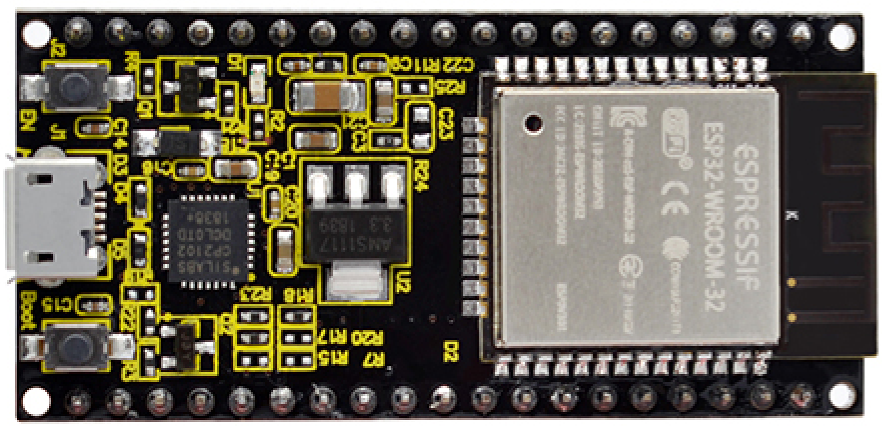
2.KIT LIST

				
ESP32 Mainboard*1	Blue LED*10	Red LED*10	Yellow LED*10	Green LED*10
				
White LED*10	RGB*1	220Resistor*10	10KResistor*10	1KResistor*10
				
10K Potentiometer*1	Active Buzzer*1	Passive Buzzer*1	Button Switch*4	Tilt Switch*1
				
Photoresistor*3	Flame Sensor*1	10K Thermistor*1	Yellow Cap*2	Blue Cap*2
				
IC 74HC595N *1	1-Digit Tube Display*1	4-Digit Tube Display*1	8*8 Dot Matrix Display *1	LCD_128X32_DOT *1
				
S8050 Triode *2	S8550 Triode *2	Fan*1	Dc Motor*1	Breadboard Wire*30
				

CHAPTER

THREE

3.KEYESTUDIO ESP32 CORE BOARD



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.

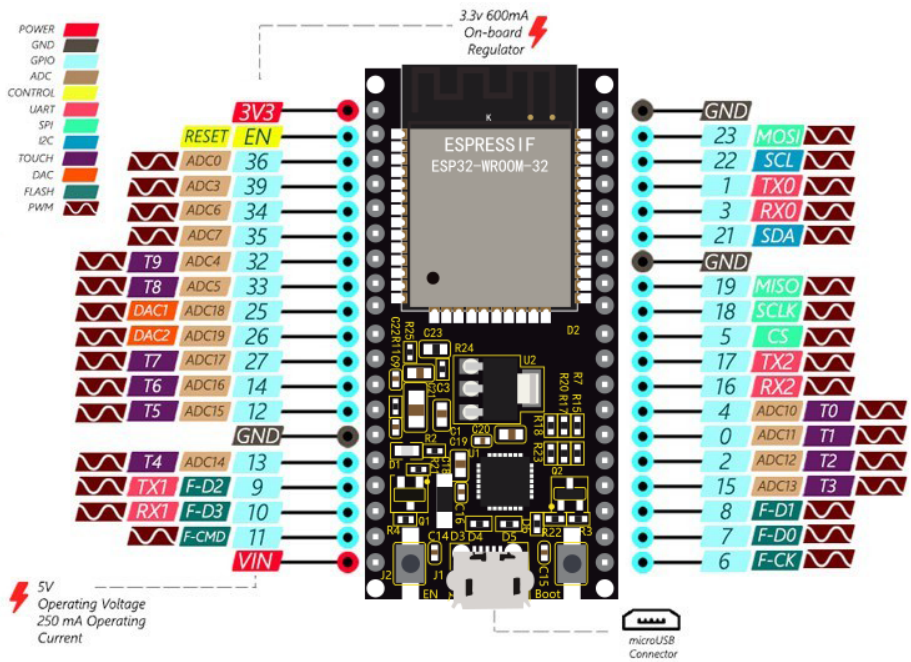
The ESP-WROOM-32 module is the industry's leading integrated WiFi + Bluetooth solution with less than 10 external components. It integrates antenna switches, RF balun, power amplifiers, low noise amplifiers, filters as well as power management modules. At the same time, it also integrates TSMC's low-power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

Specifications

- Microcontroller: ESP-WROOM-32 Module
- USB to serial port chip: CP2102-GMR
- Working voltage: DC 5V
- Working current80mAAverage
- Current supply500mAMinimum
- Working temperature range : -40°C ~ +85°C
- WiFi modeStation/SoftAP/SoftAP+Station/P2P
- WiFi protocol802.11b/g/n/e/i802.11nspeed up to 150 Mbps
- WiFi frequency range2.4 GHz ~ 2.5 GHz

- Bluetooth protocolconform to Bluetooth v4.2 BR/EDR and BLE Standard
- Dimensions55*26*13mm
- Weight9.3g

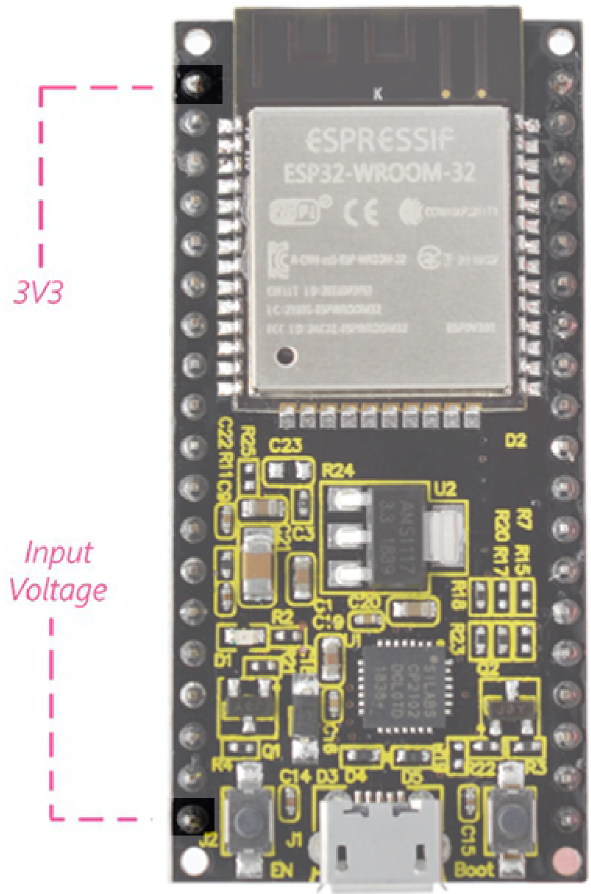
Pin out



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.

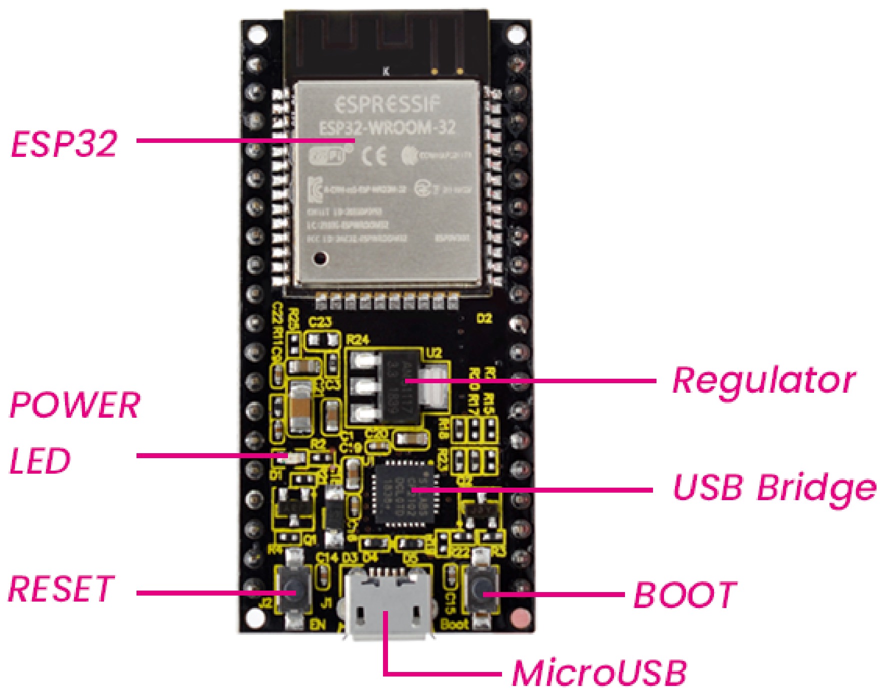
SPI: There are two SPI interfaces on the module, which can be used to connect the display screen, SD/microSD memory card module as well as external flash memory, etc.

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 motors, LED lights and color changes for some other sensorsfor example: color sensor, etc.

Components



CHAPTER

FOUR

GETTING STARTED WITH ARDUINO

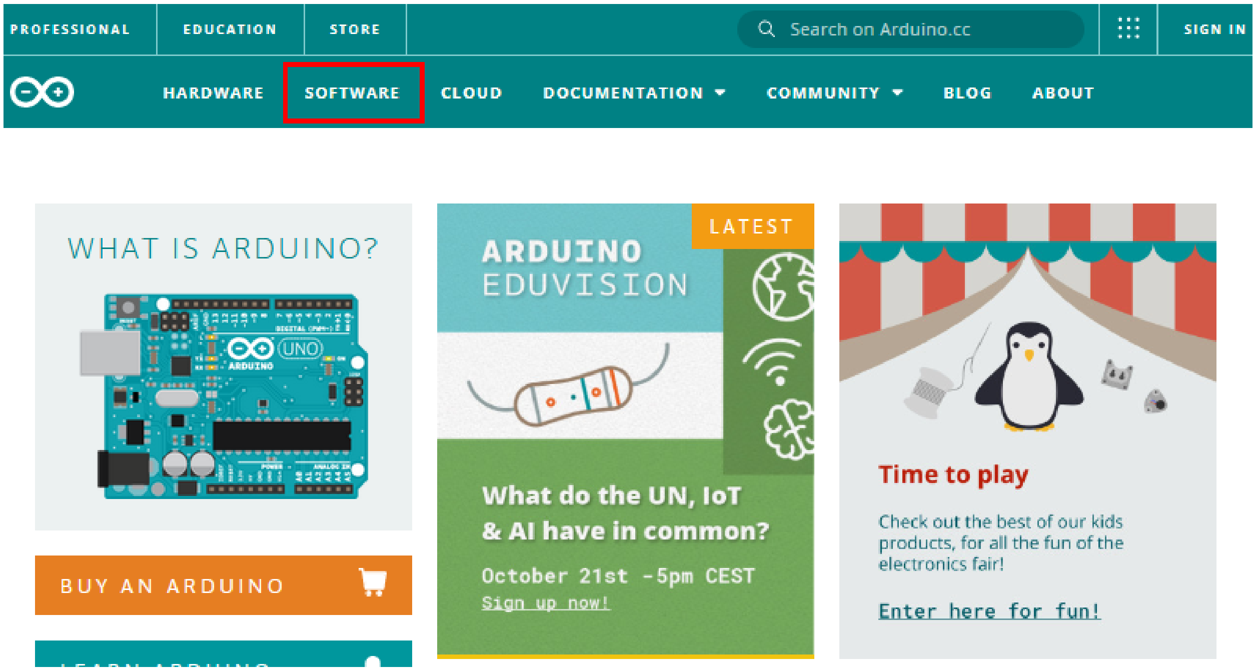
Development Environment Configuration

Click on the link to enter the development environment setup tutorial*Development Environment Configuration-Windows*

4.1 Windows System

4.1.1 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

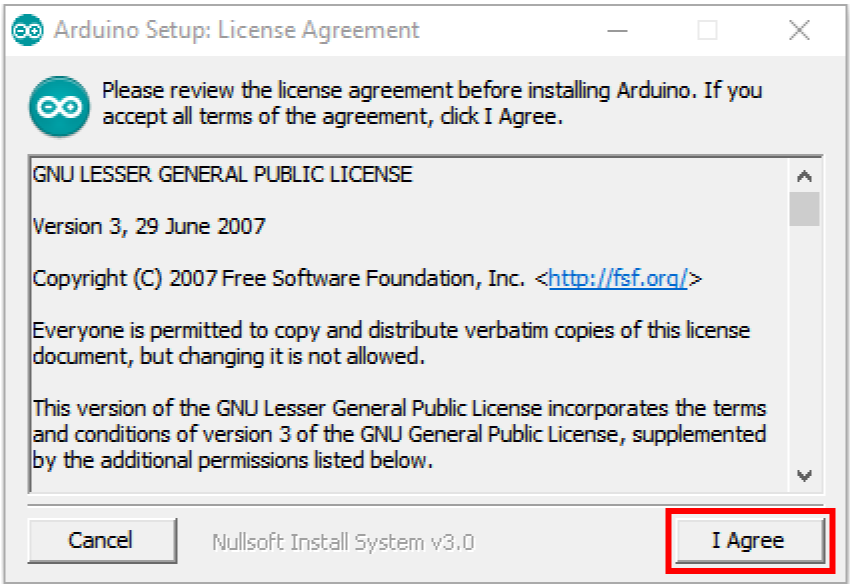


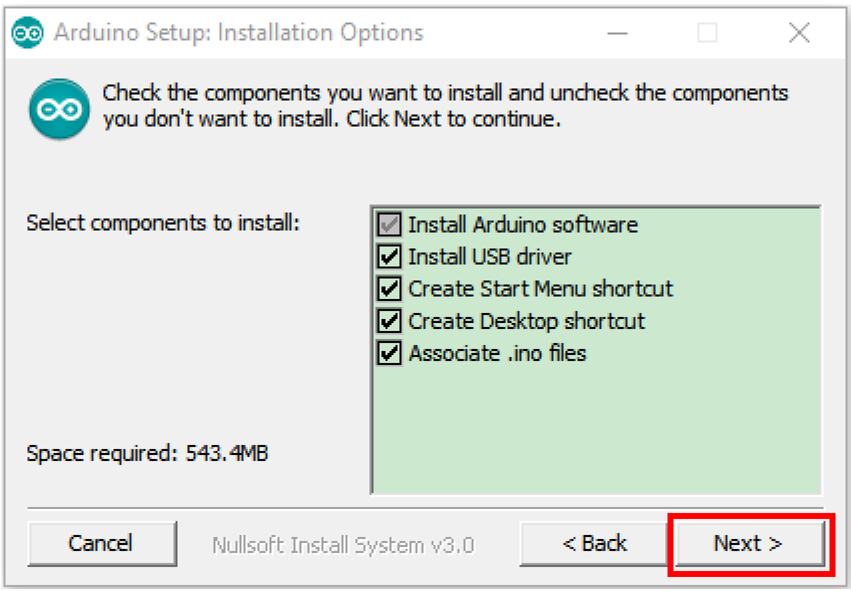
(2) Then, select and download the corresponding installer for your operating system. If you are a Windows user, please select "Windows Installer" to download to install the driver correctly.

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.

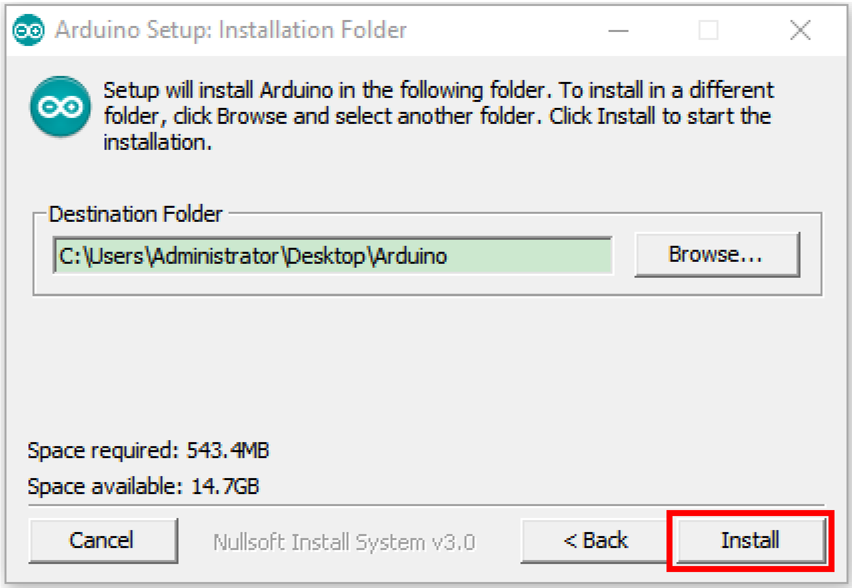
In general, you can click **JUST DOWNLOAD** to download it.

(3) After 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.

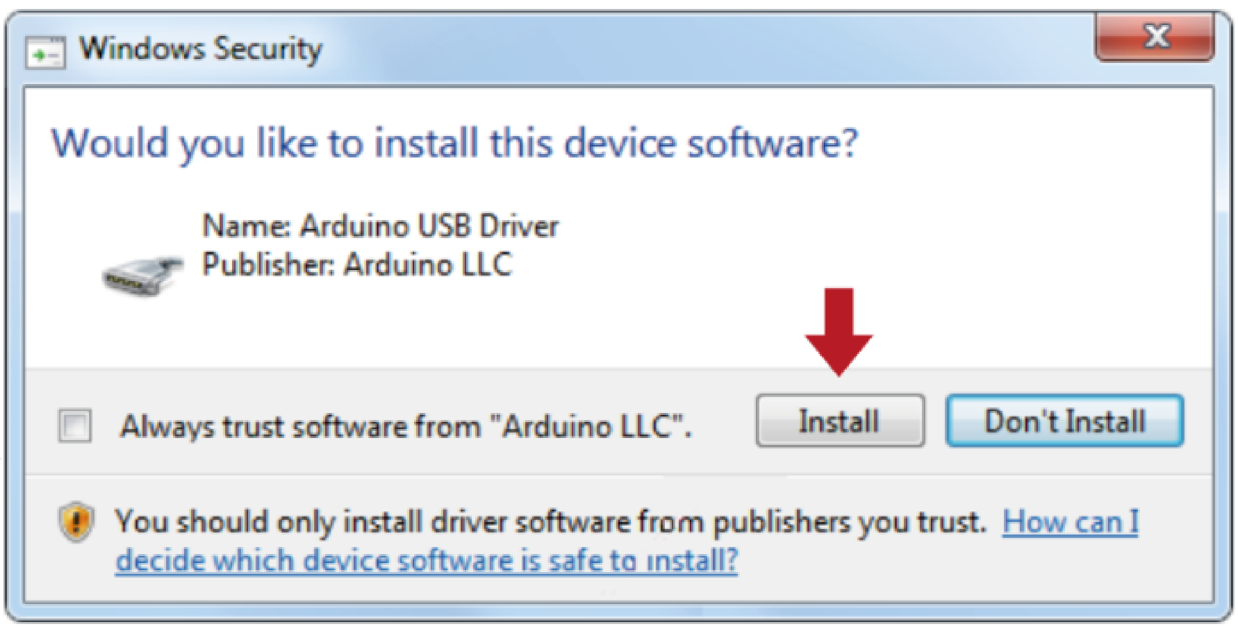




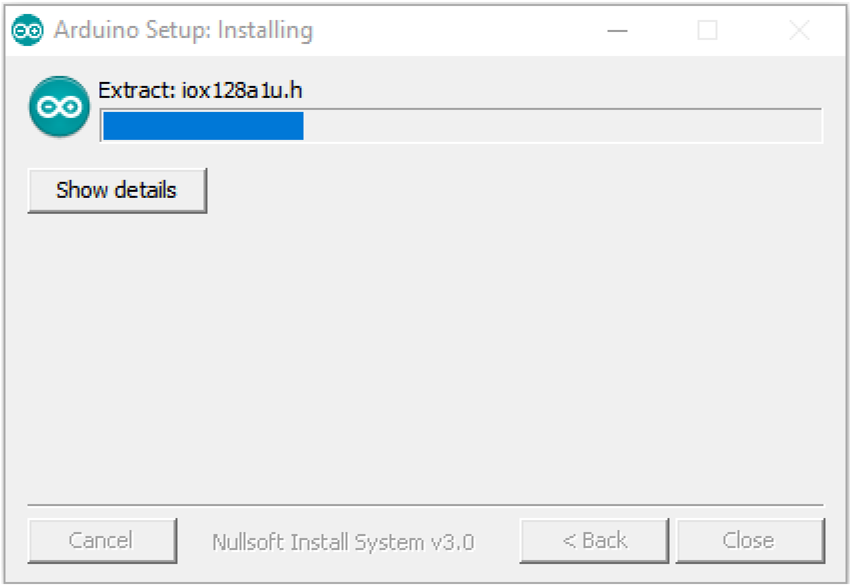
4Select the installation directory (we recommend keeping the default directory), and then click **Install**.



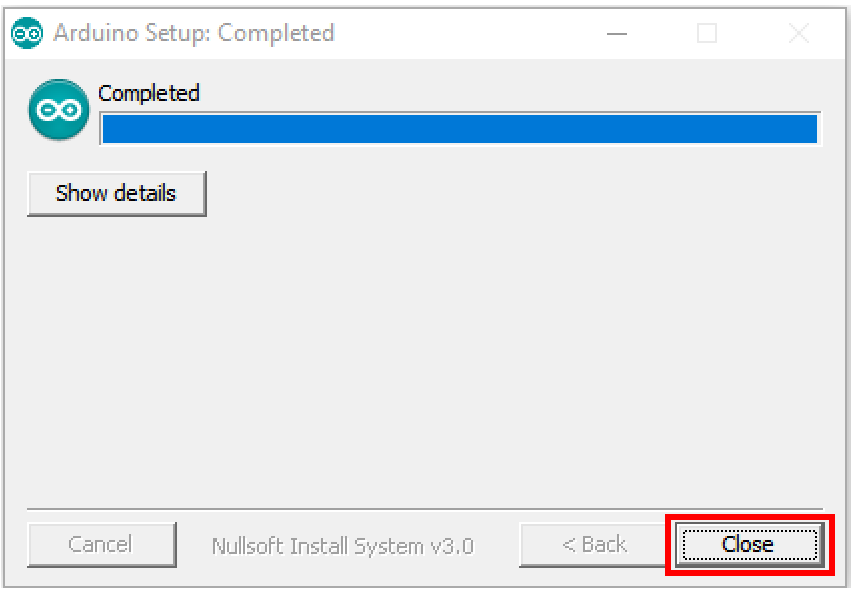
5Select Install if the following screen appears.



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 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 can not communicate with the 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.

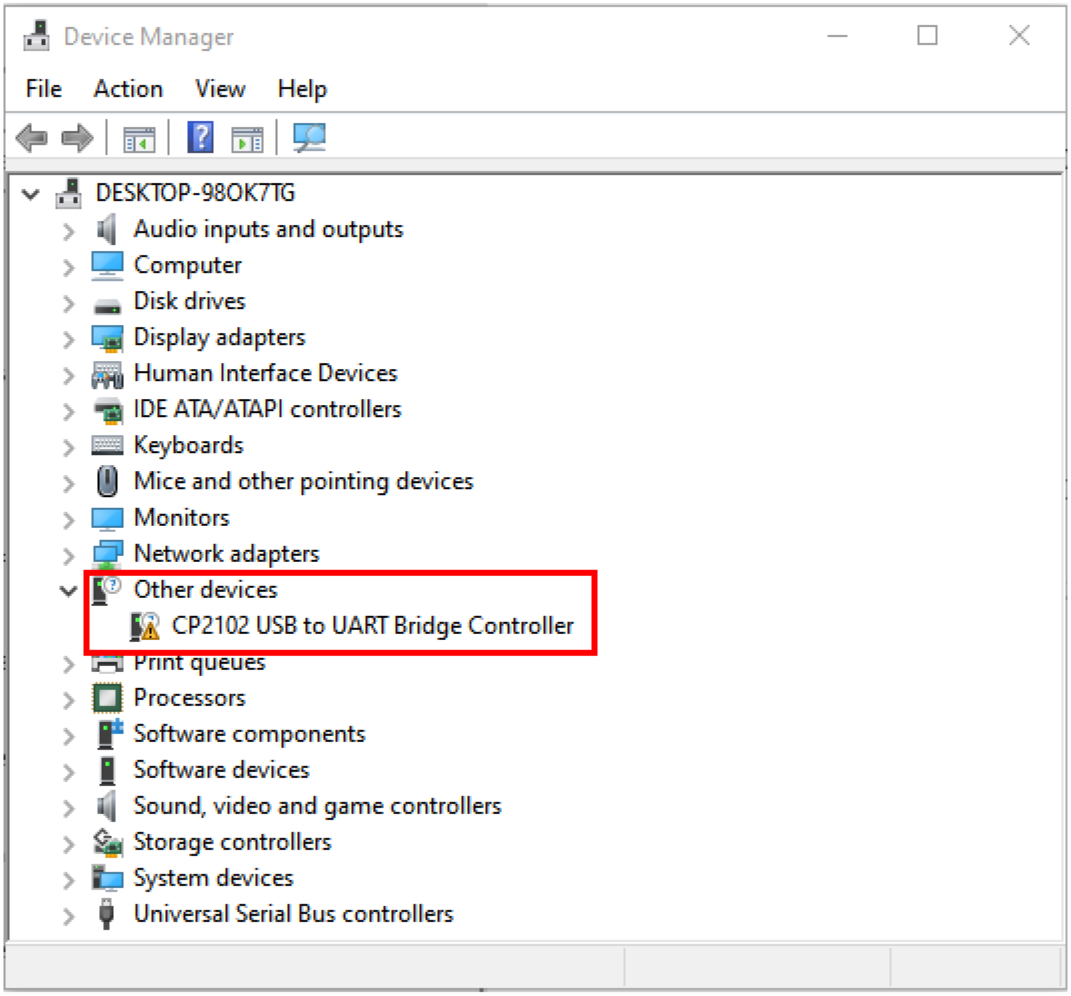
Note:

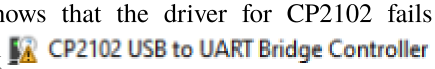
1. Please make sure that your IDE is updated to 1.8.0 or newer version

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

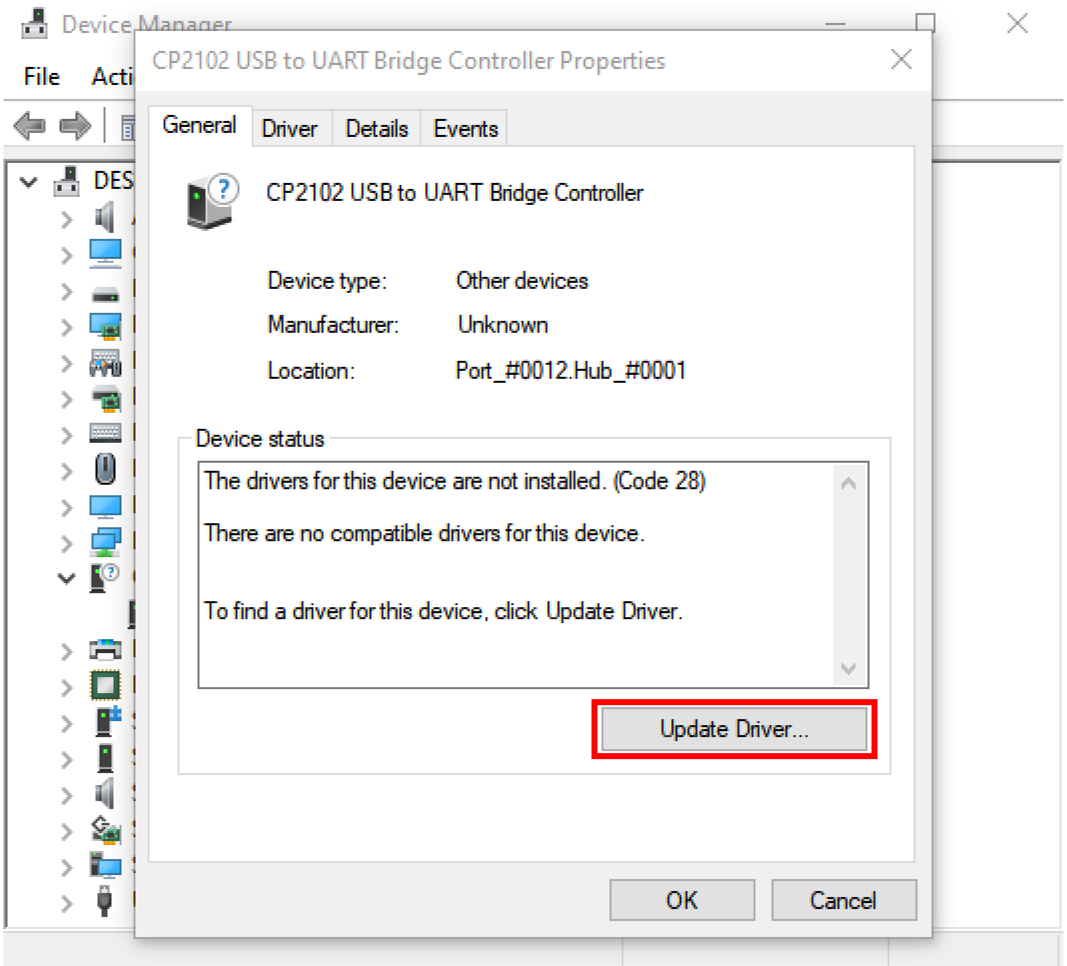
Link to download the driver of CP2102: CP2102-Driver-File-Windows

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.

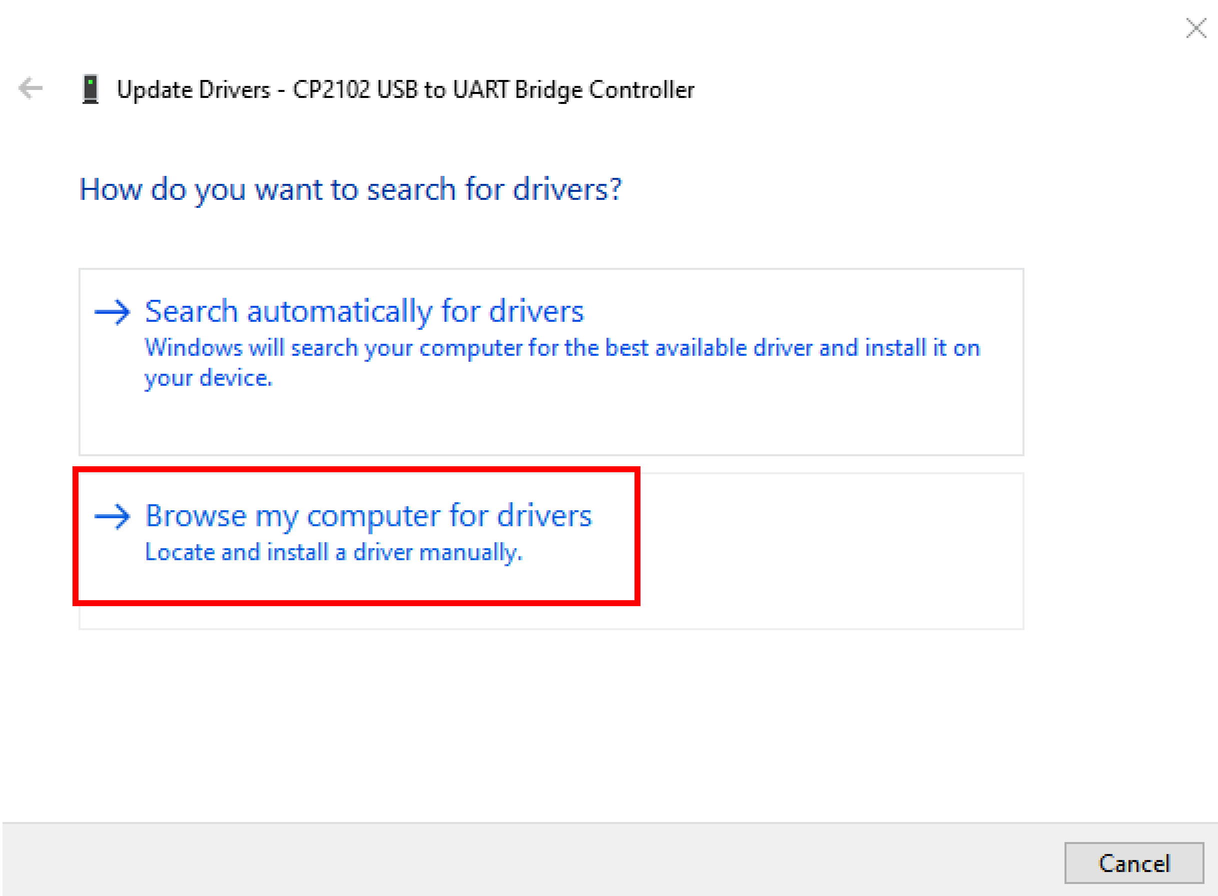


It shows that the driver for CP2102 fails to be installed successfully if there is a yellow mark. Double-click 

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

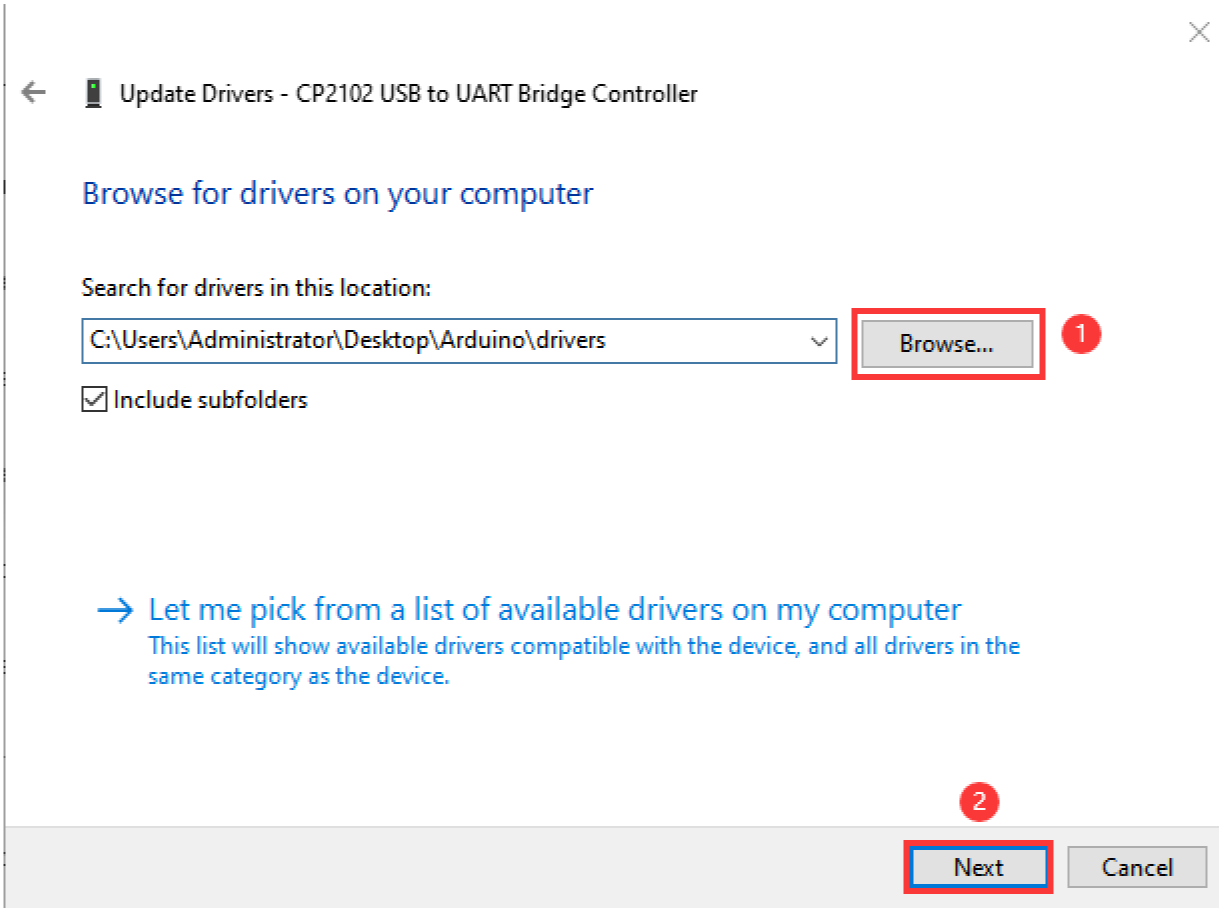


Click "**Browse my computer for drivers**"to find the Arduino software we installed or downloaded.

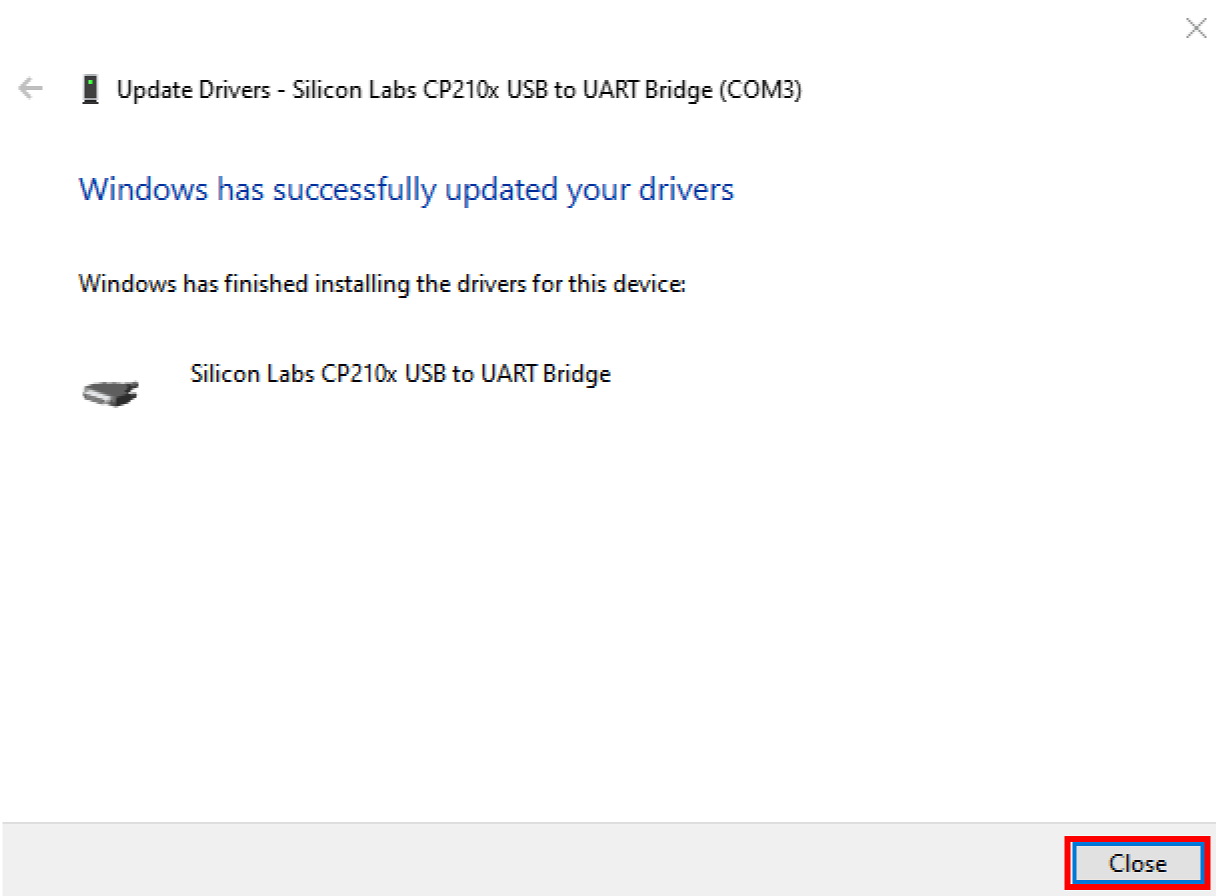


There is a **drivers** folder in Arduino software installed package (Arduino), open the driver folder and you can see the driver of CP210X series chips.

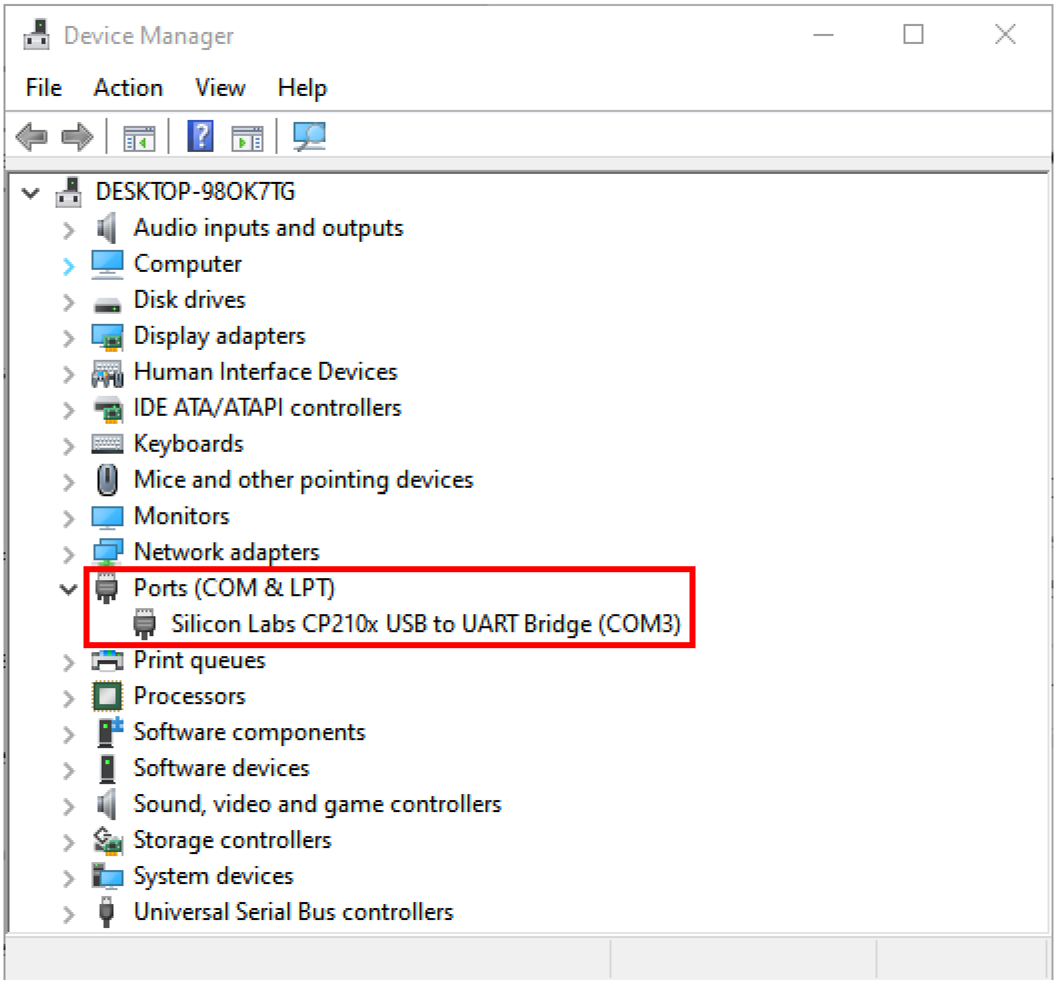
Click“**Browse…**”, then find the drivers folder, or you could enter“driver”to search in rectangular box, then click“**Next**”,



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.



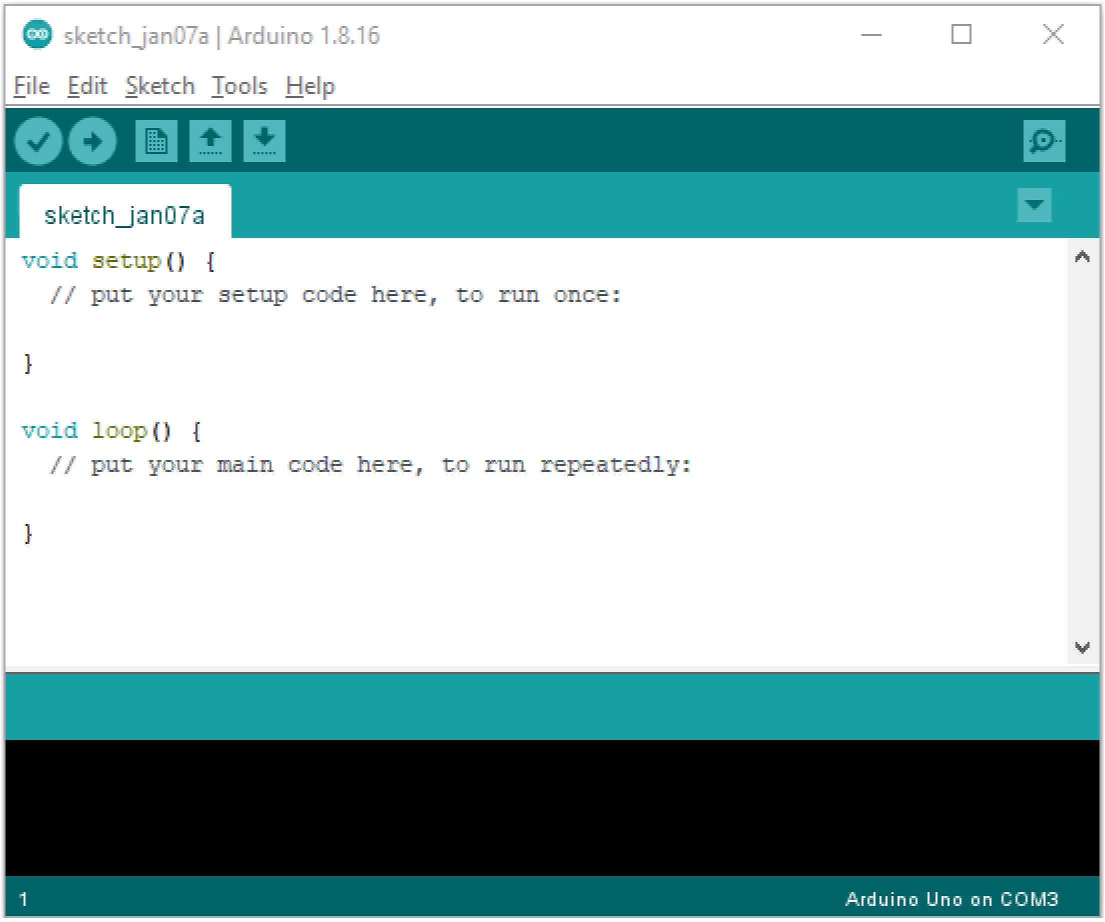
4.1.3 Install the ESP32 on Arduino IDE

The installation process for ESP32 is almost the same as that for ESP8266. If you are to install ESP32 on an Arduino IDE, follow these steps

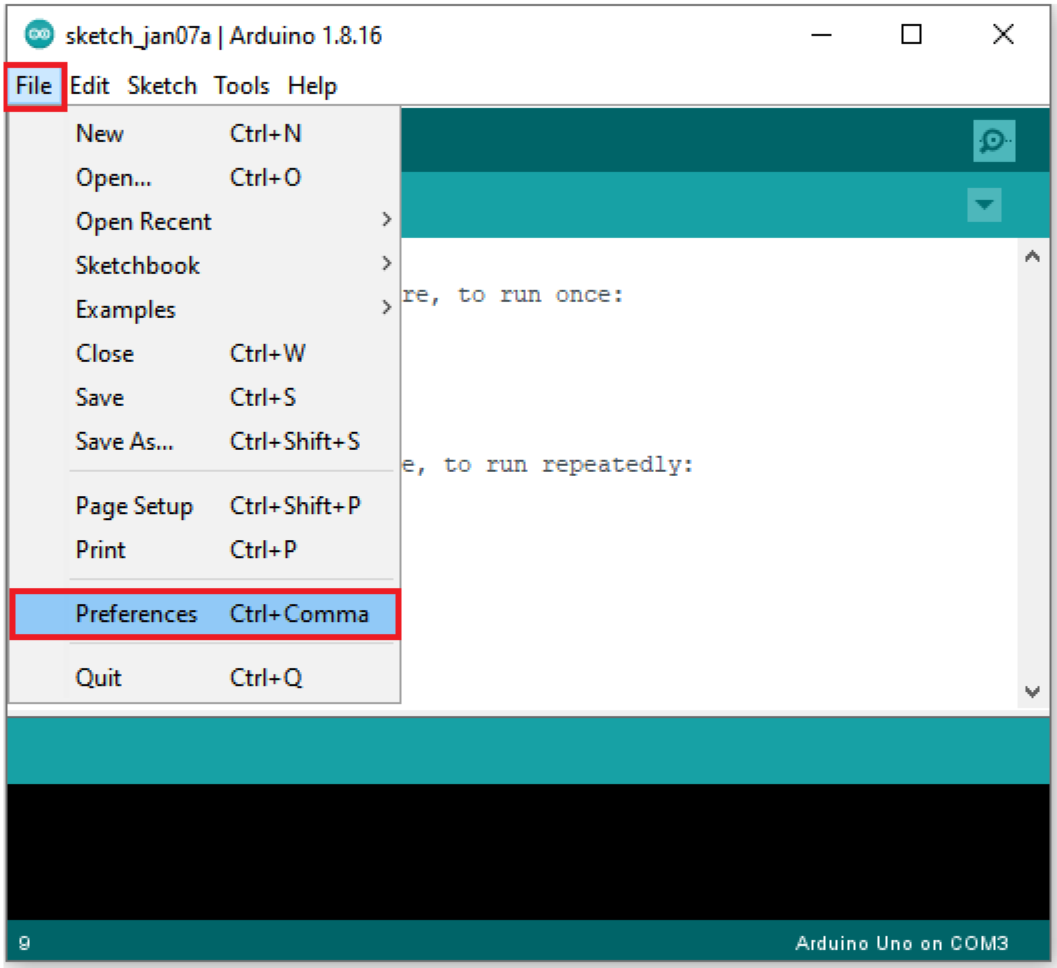
Noteyou need to download Arduino IDE 1.8.5 or advanced version to install the ESP32.

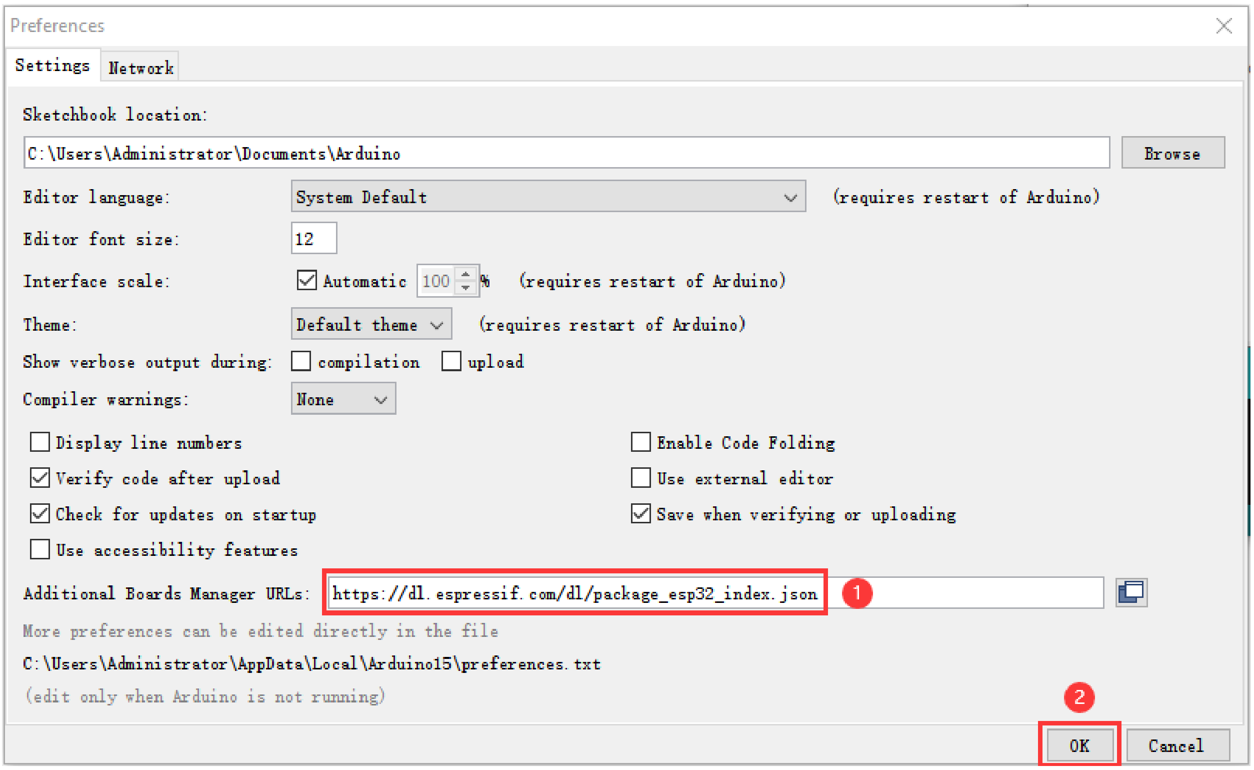
1) Click the icon 

to open the Arduino IDE

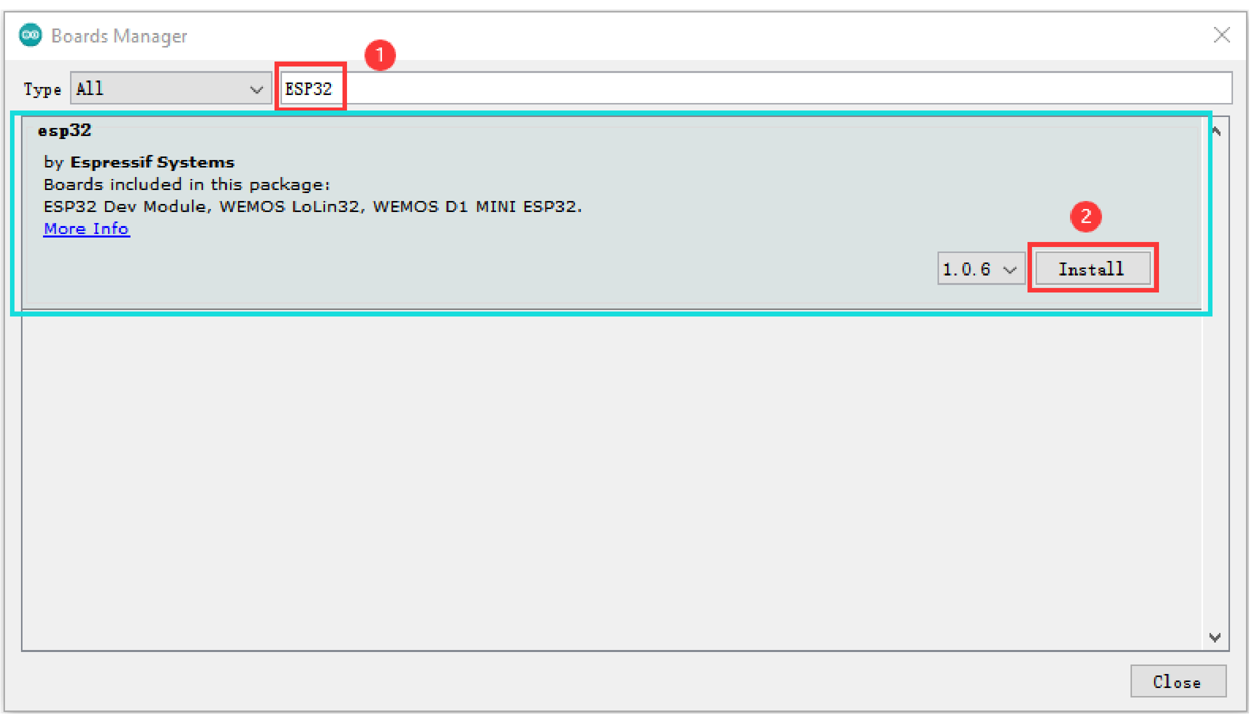


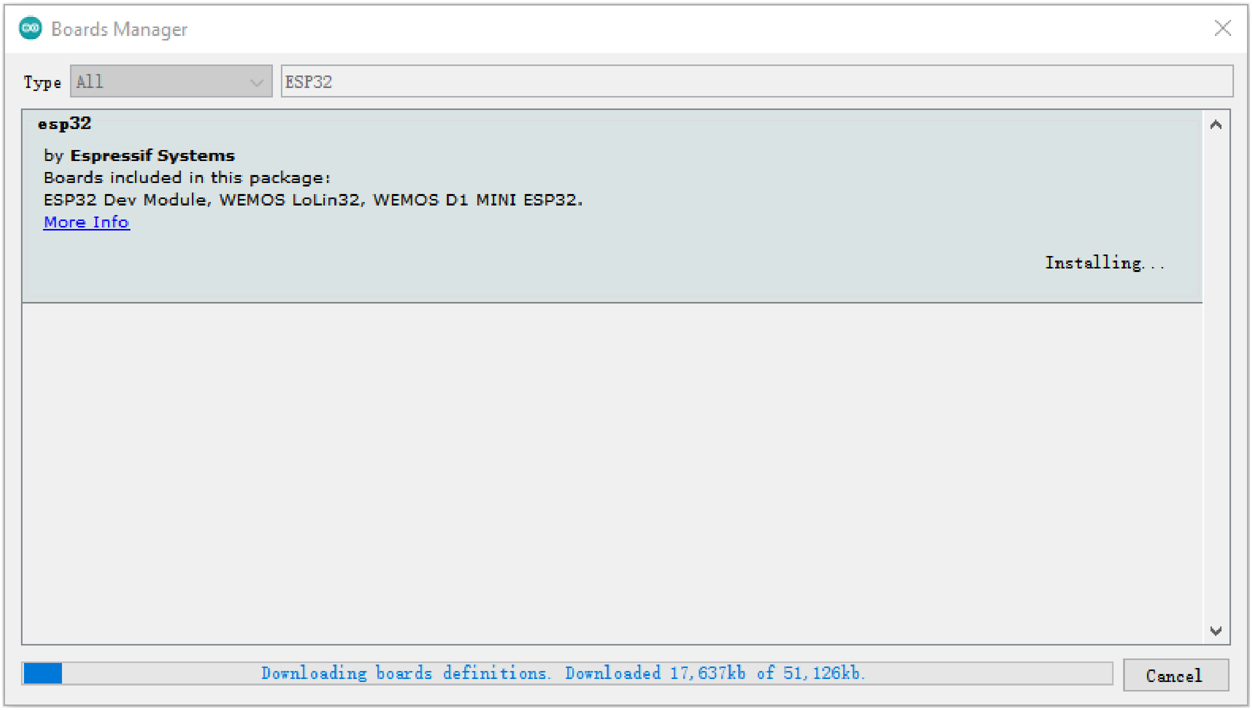
2Click“**File**” →**“Preferences”**copy the website addresshttps://dl.espressif.com/dl/package_esp32_index.json in the“**Additional Boards Manager URLs:**”and then click“**OK**” to save the address.


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.

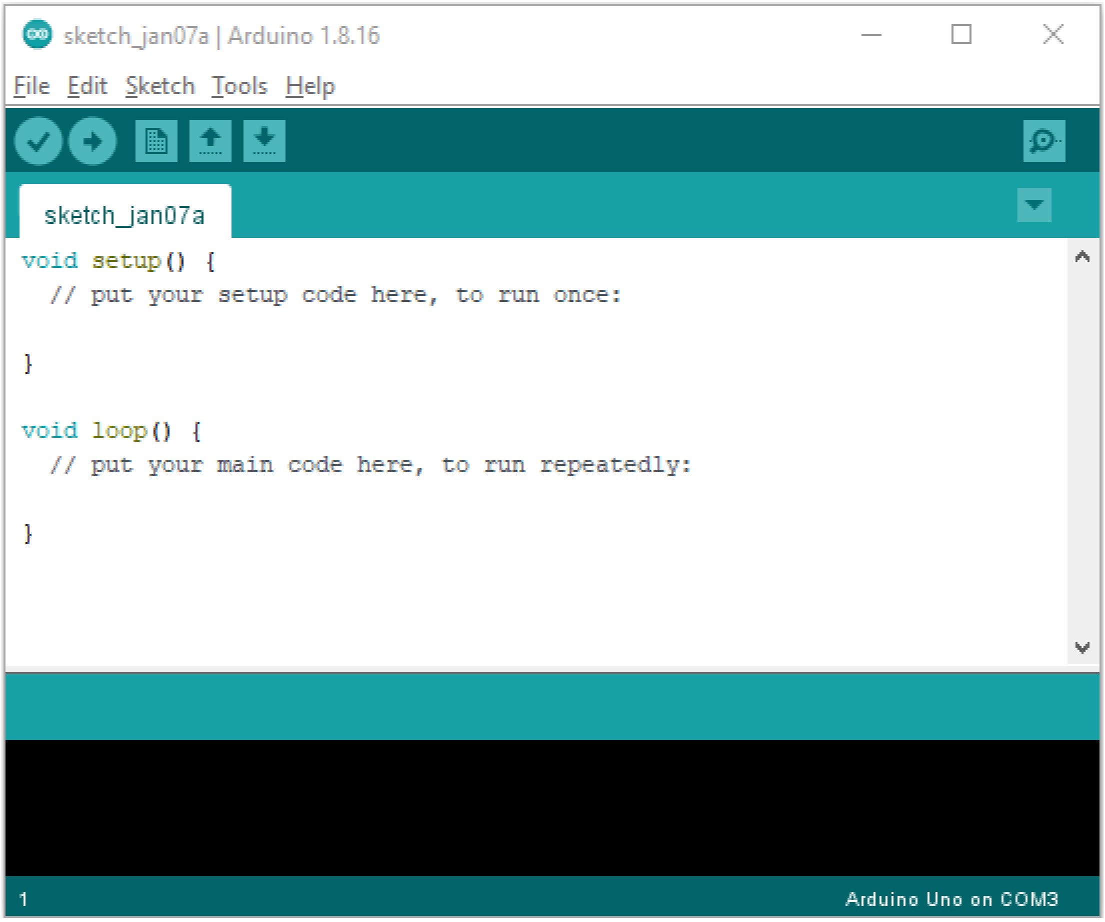




4. After successful installation, click“**Close**”to Close the page

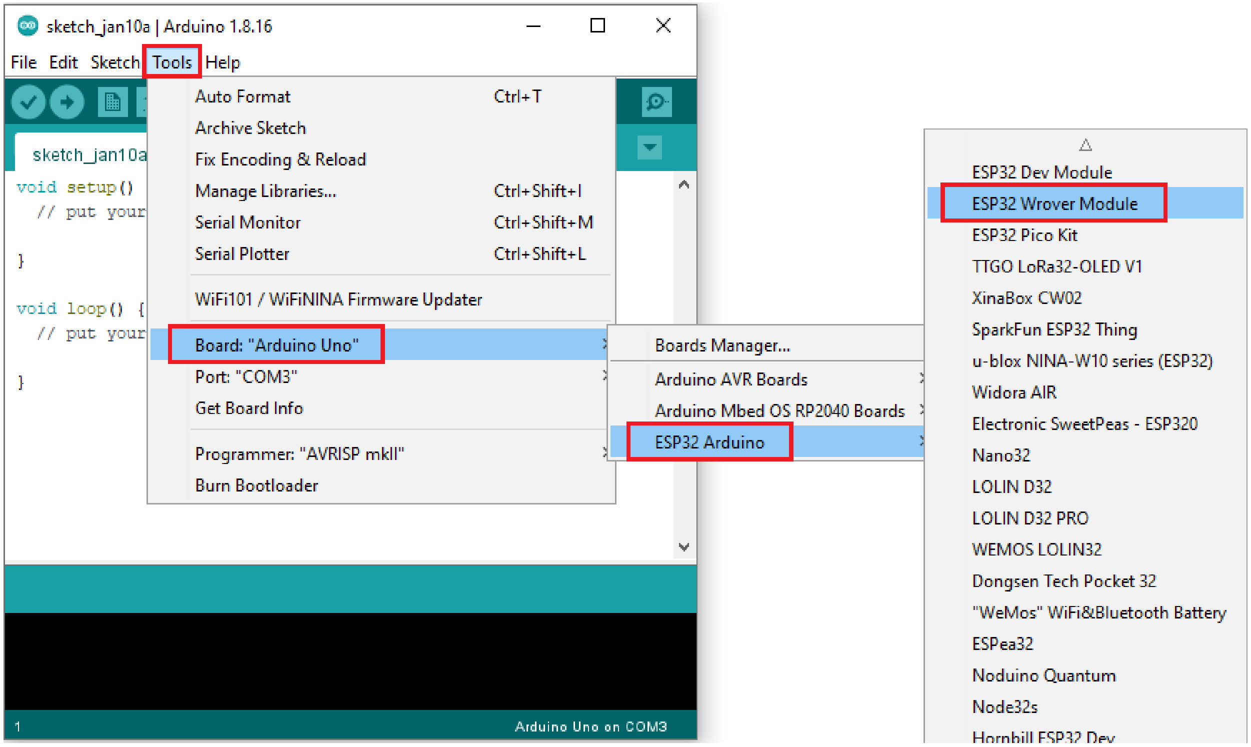
4.1.4 Arduino IDE Setting:

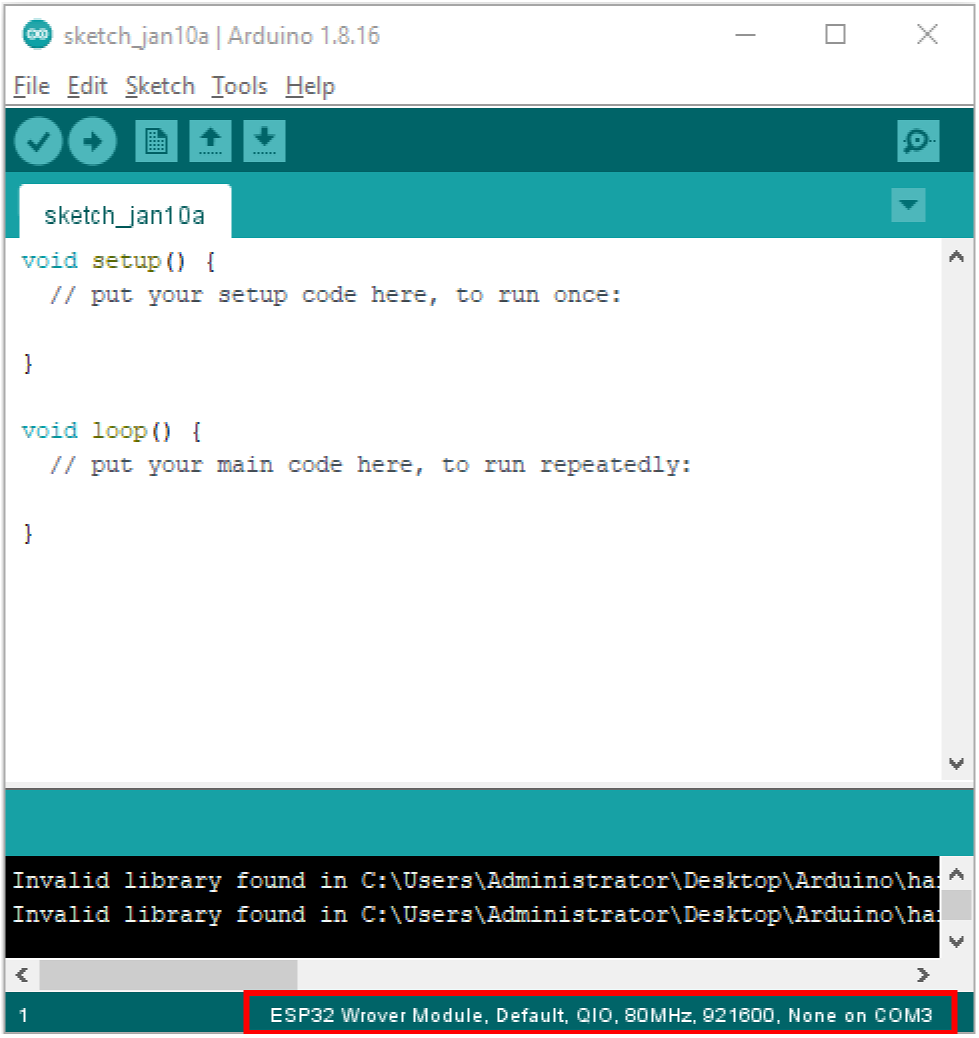
1Click the icon  to open the Arduino IDE.



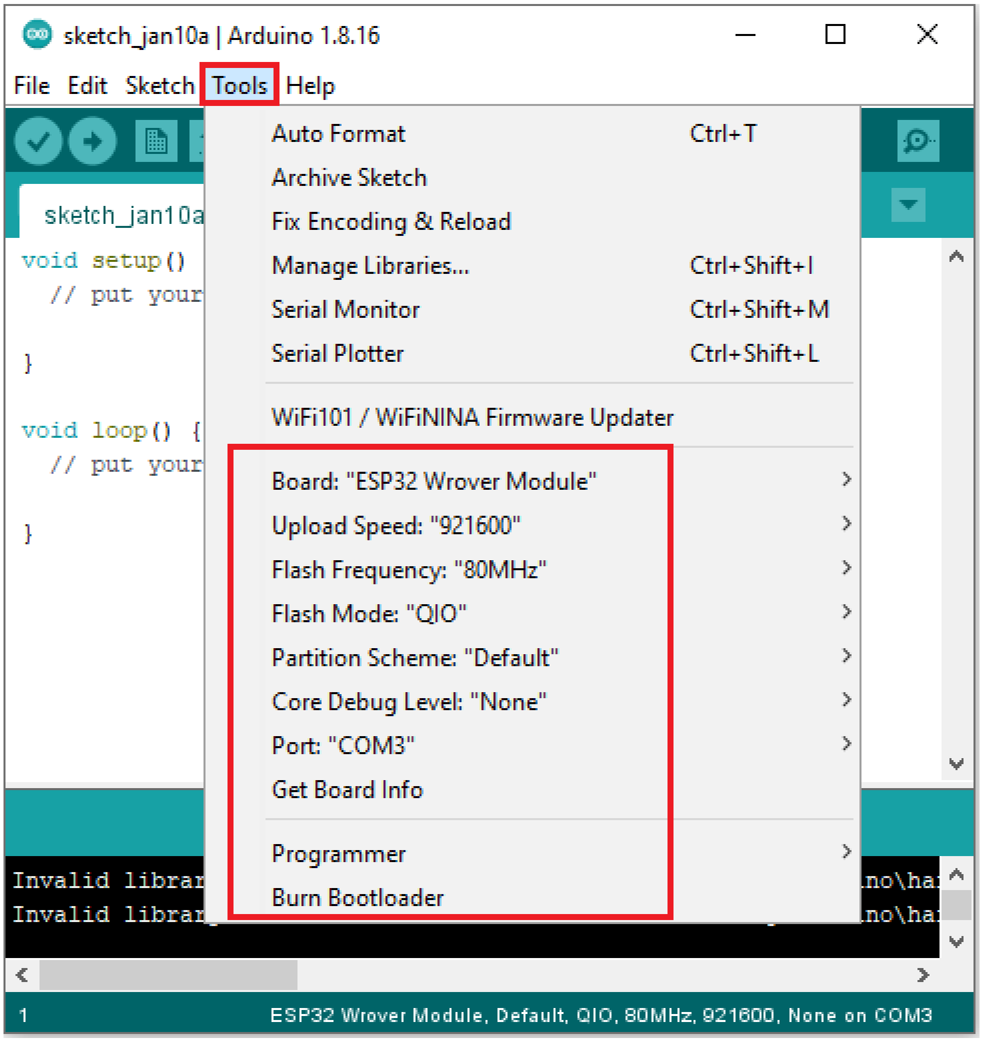
2When downloading the code to the board, you must select the correct name of Arduino board that matches the board connected to your computer, then click"**Tools**"→"**Board:**". As shown below ;

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

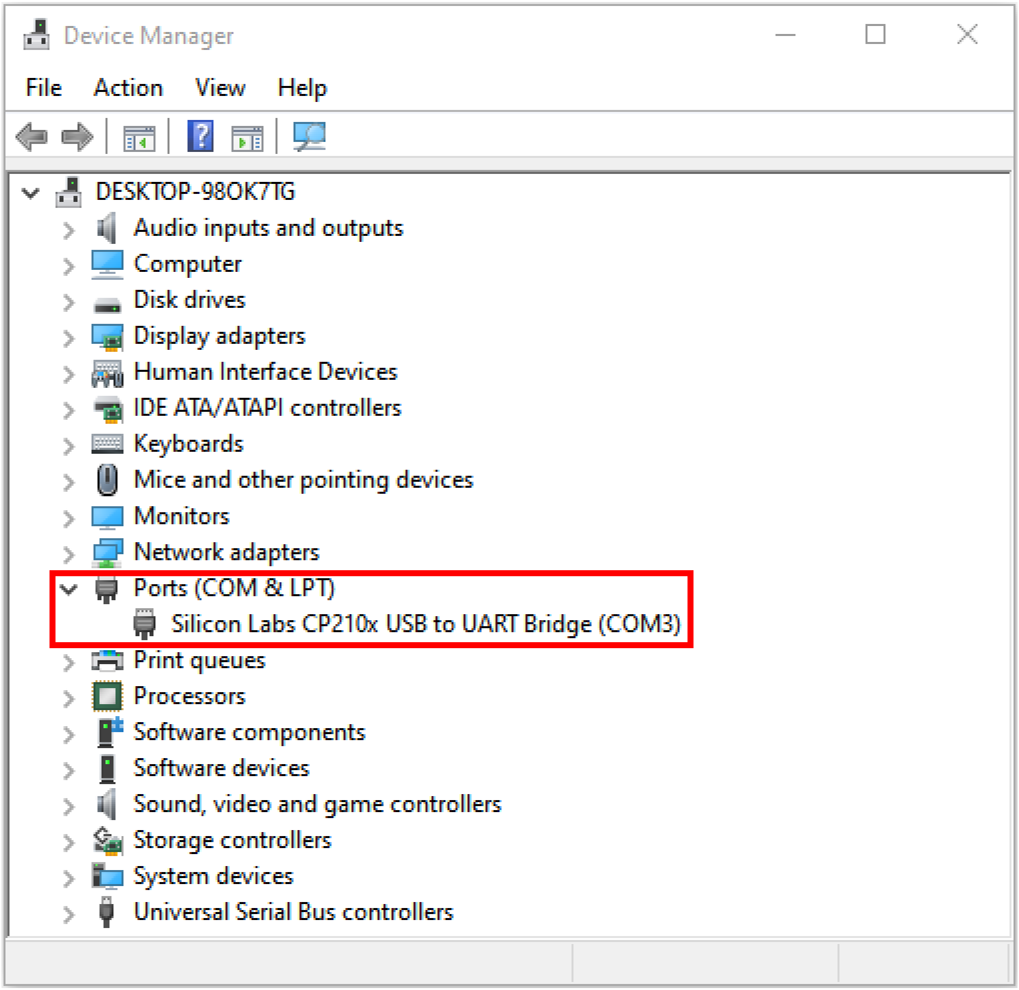

sketch_jan10a | Arduino 1.8.16
File Edit Sketch Tools Help
sketch_jan10a
void setup()
// put your
}
void loop() {
// put your
}
Auto Format Ctrl+T
Archive Sketch
Fix Encoding & Reload
Manage Libraries... Ctrl+Shift+I
Serial Monitor Ctrl+Shift+M
Serial Plotter Ctrl+Shift+L
WiFi101 / WiFiNINA Firmware 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
1
Arduino Uno on COM3

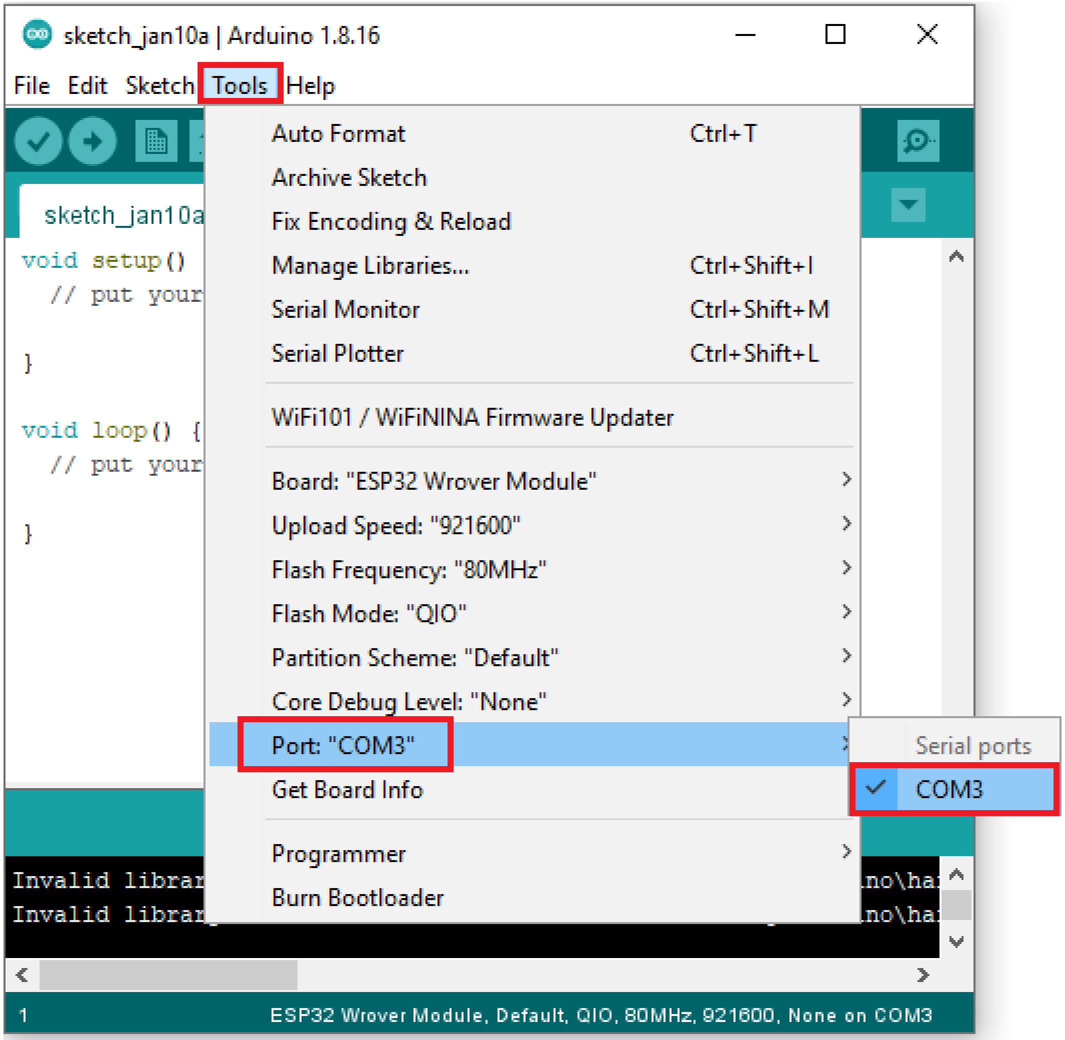


Set the board type as follows:

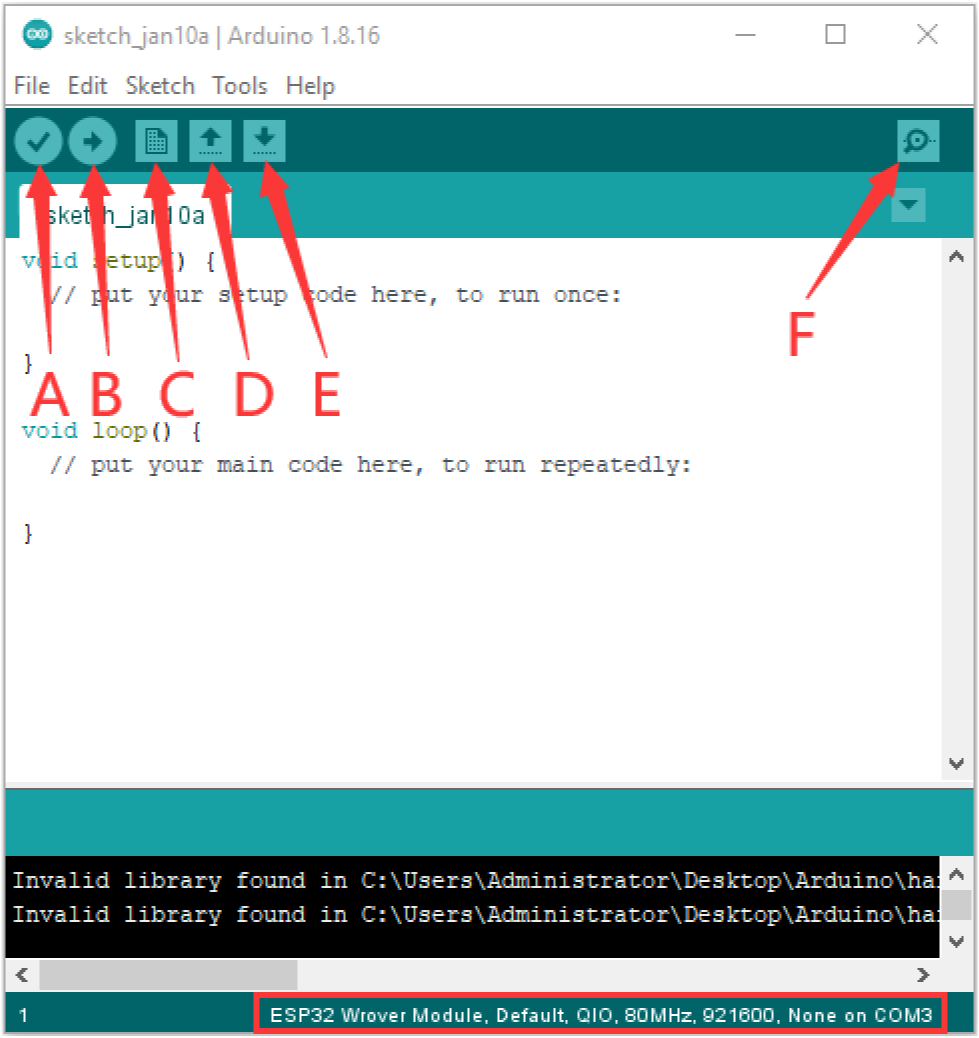


Then select the correct COM port (the corresponding COM port can be seen after the driver is installed successfully).


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.

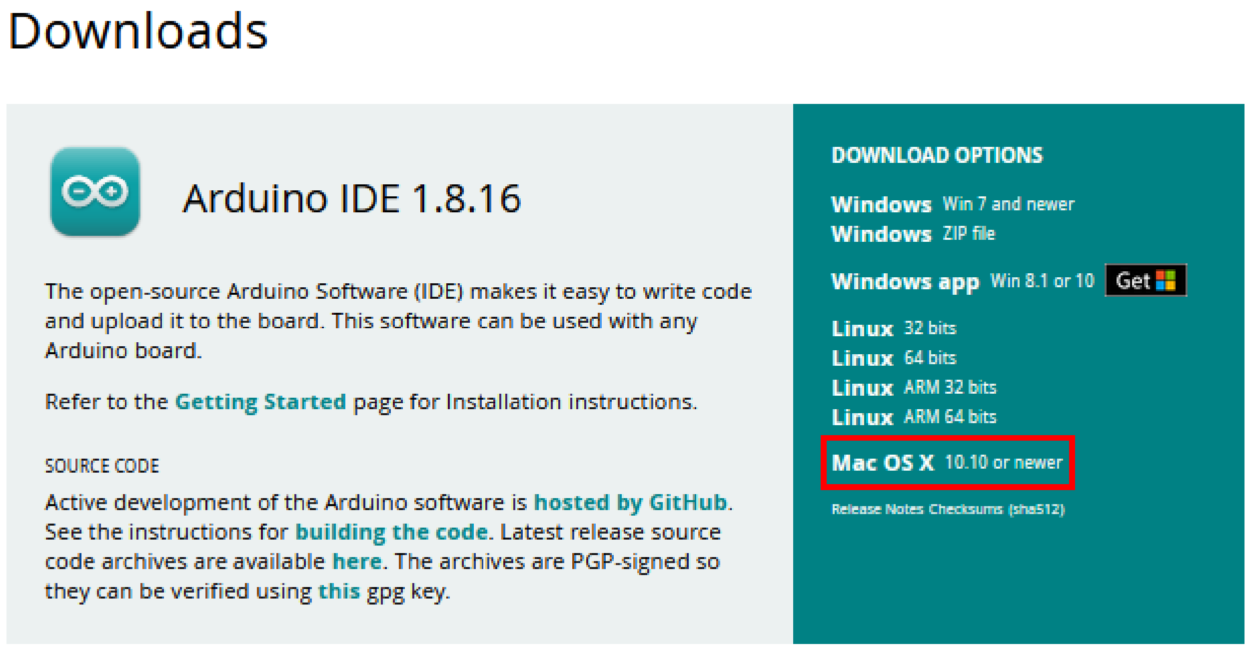
F- Used to send the serial data received from board to the serial monitor.

4.2 Arduino MacOS

4.2.1 Development Environment Configuration–Mac OS

Click on the link to enter the development environment setup tutorial*Development Environment Configuration-MacOS*

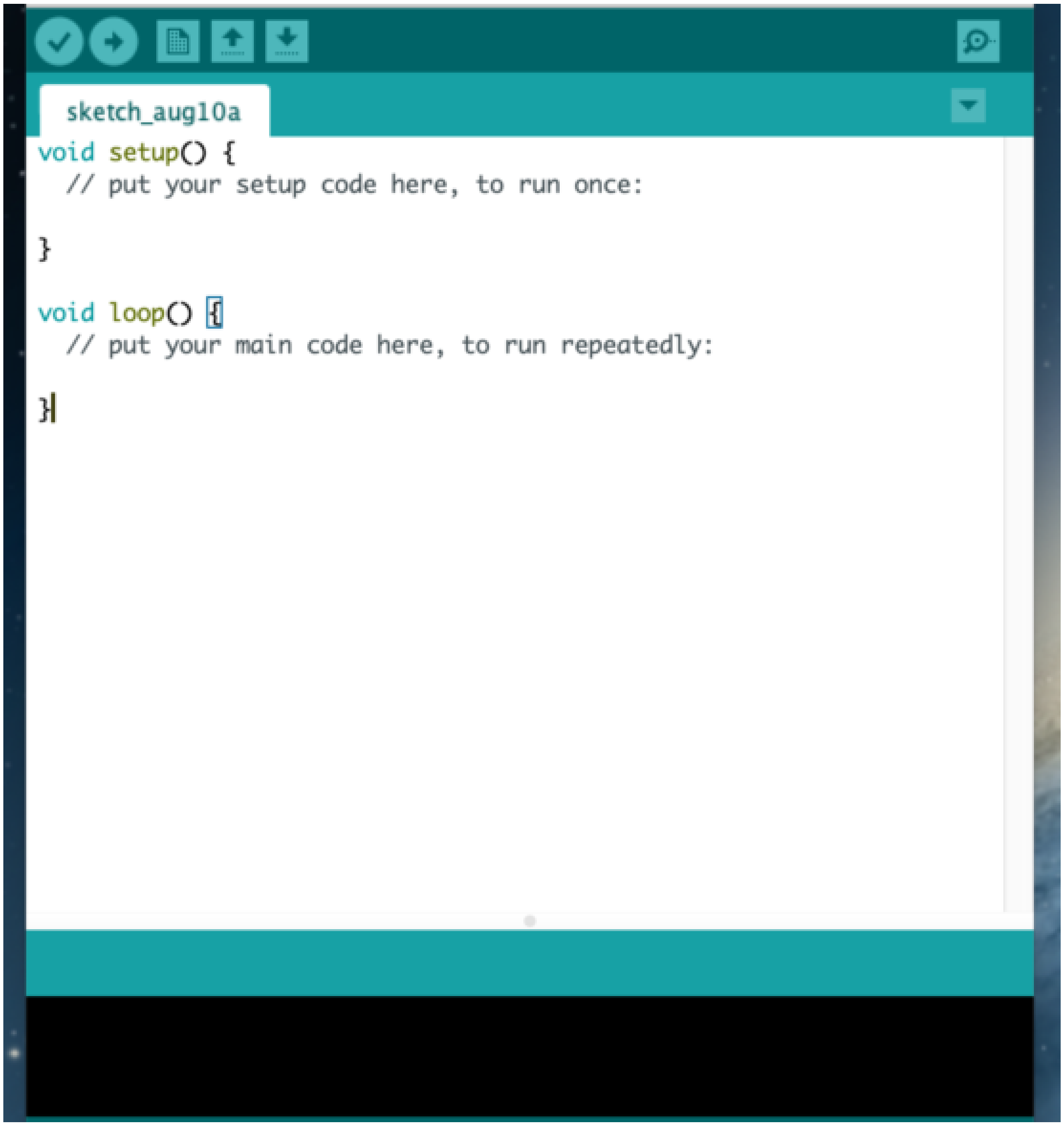
4.2.2 Download Arduino IDE:



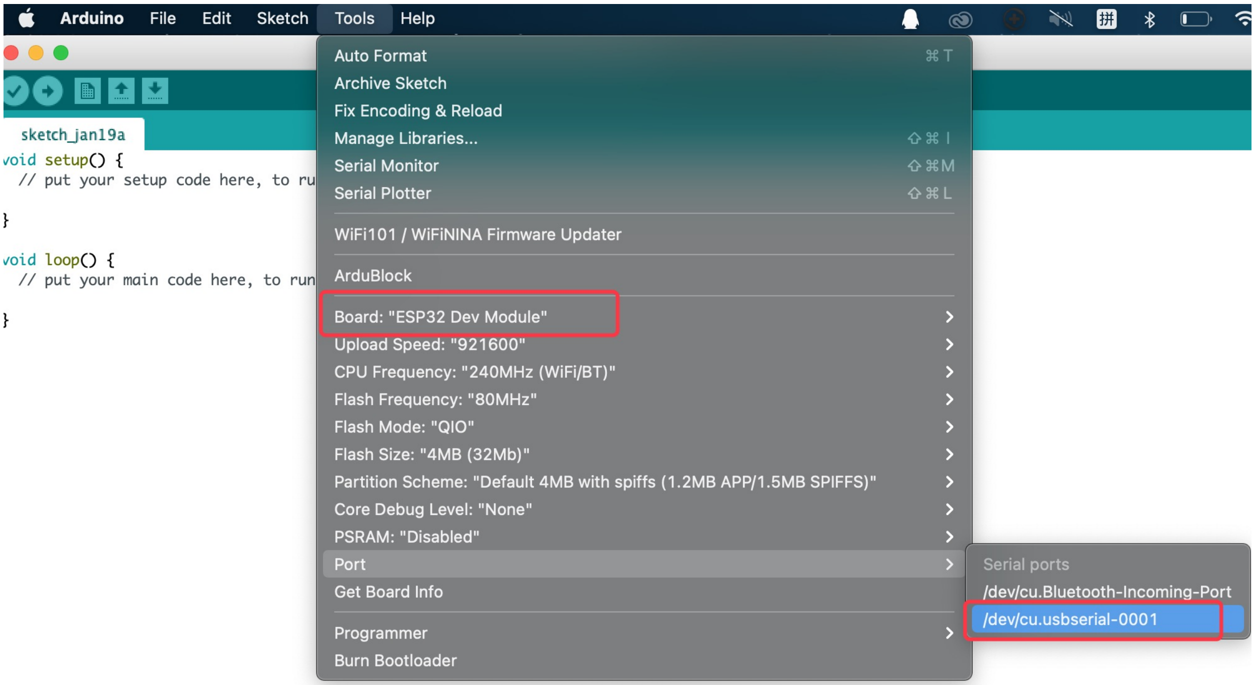
4.2.3 How to install the CP2102 driver

(Note: If you haven't installed the driver installed, please do the following.)

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



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

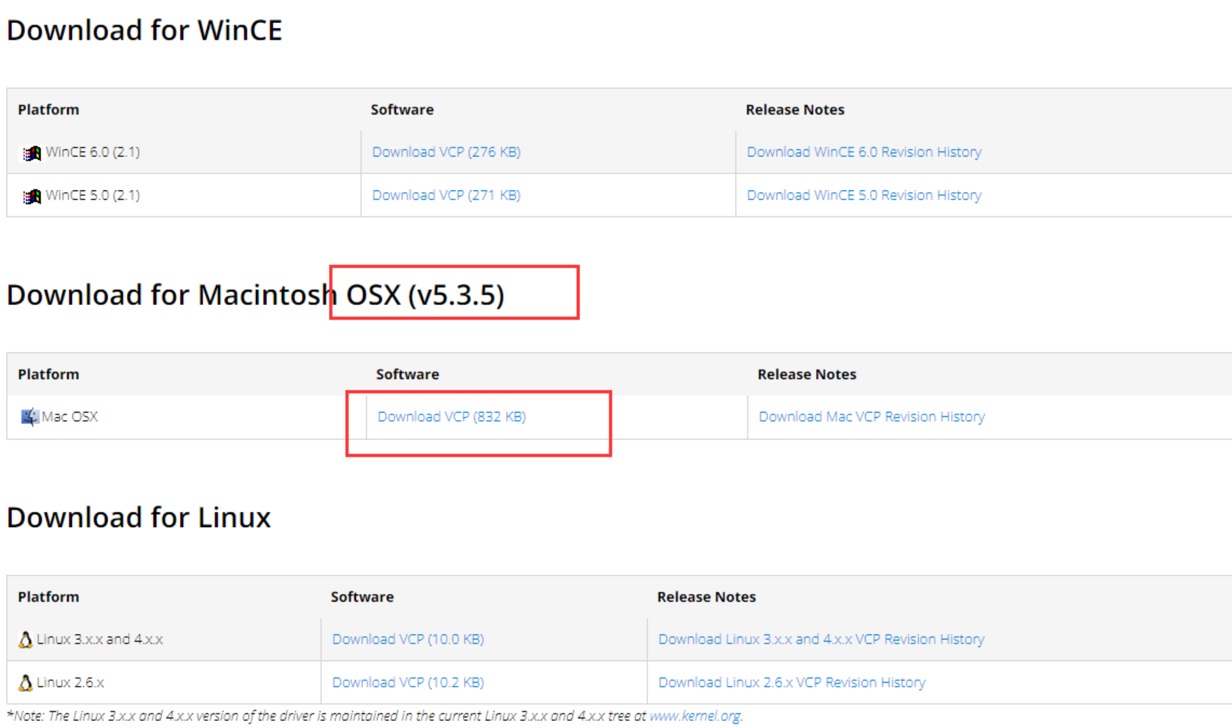


Click  to upload the test code

Note: If the the upload fails, follow the steps below to install the CP2102 driver. Perform step (2) to (16).

2Download link for CP2102CP2102-Driver-File-MAC.zip

3Download MacOS version



Download for WinCE

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

Download for Macintosh OSX (v5.3.5)

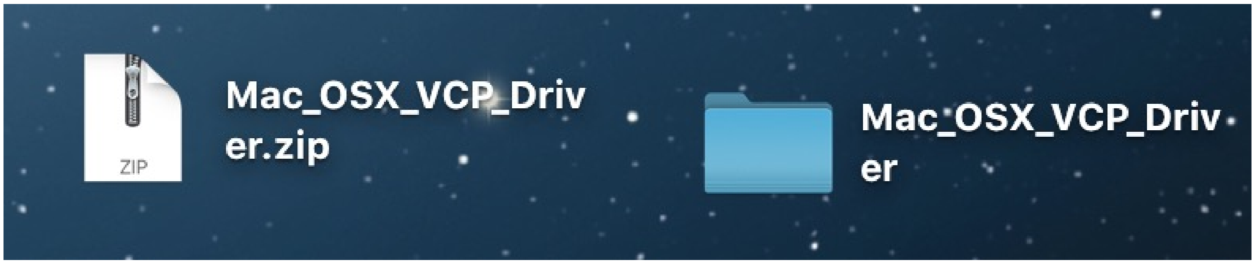
Platform	Software	Release Notes
Mac OSX	Download VCP (832 KB)	Download Mac VCP Revision History

Download for Linux

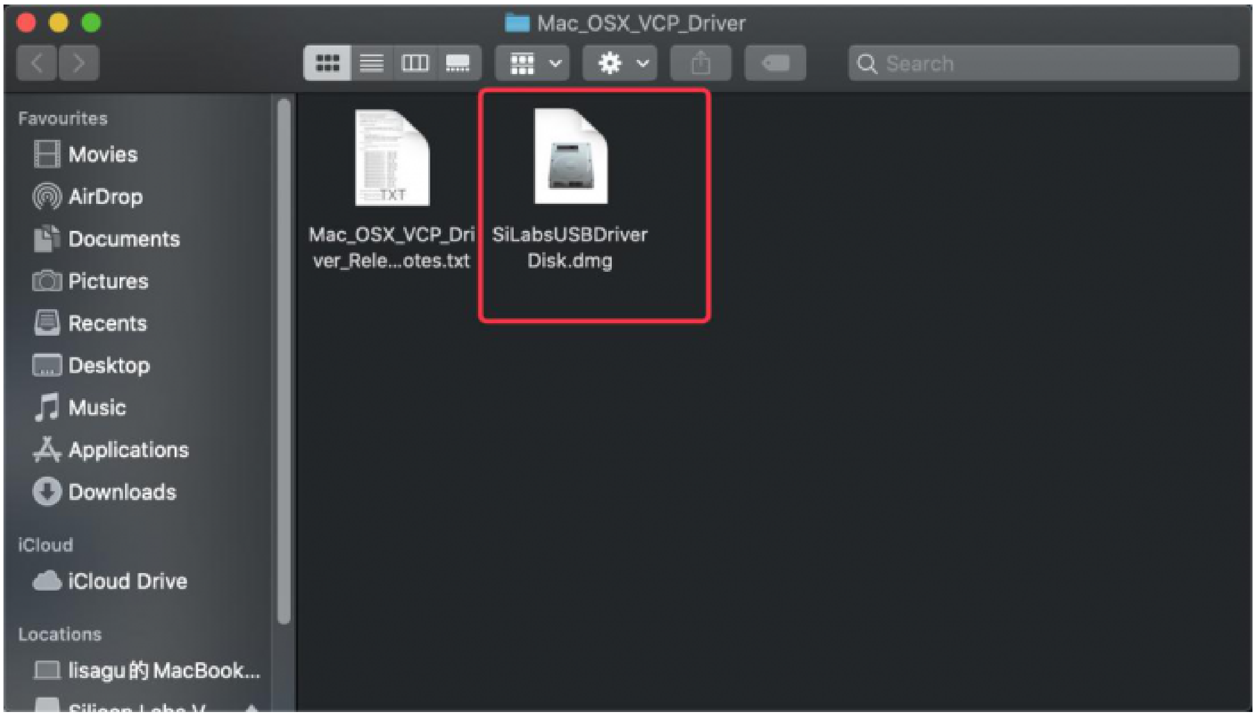
Platform	Software	Release Notes
Linux 3.x.x and 4.x.x	Download VCP (10.0 KB)	Download Linux 3.x.x and 4.x.x VCP Revision History
Linux 2.6.x	Download VCP (10.2 KB)	Download Linux 2.6.x VCP Revision History

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

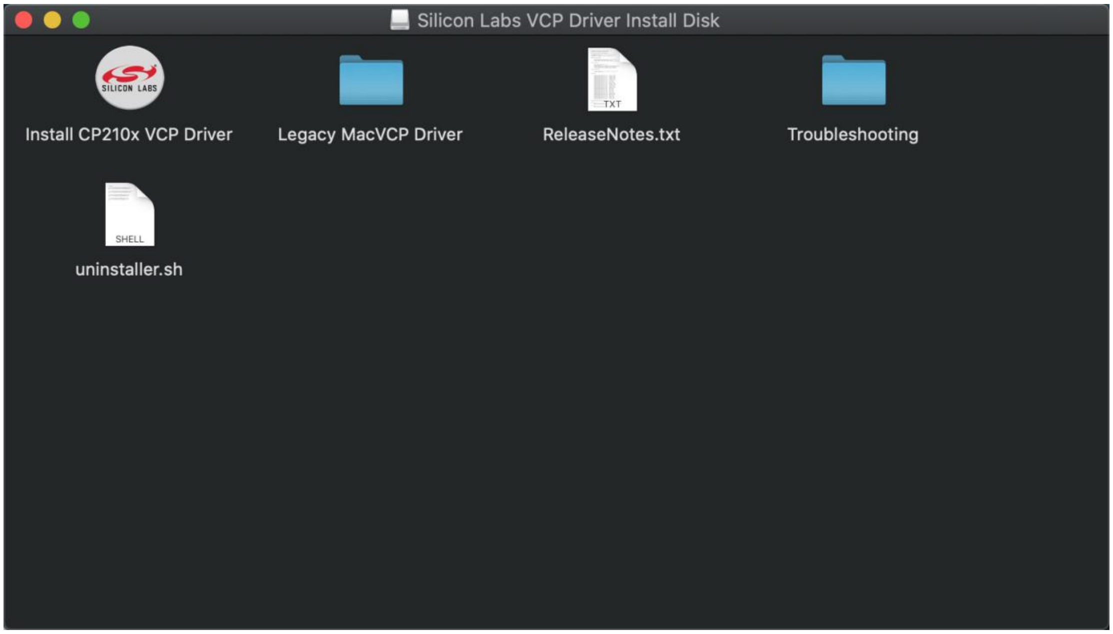
4Unzip the downloaded package



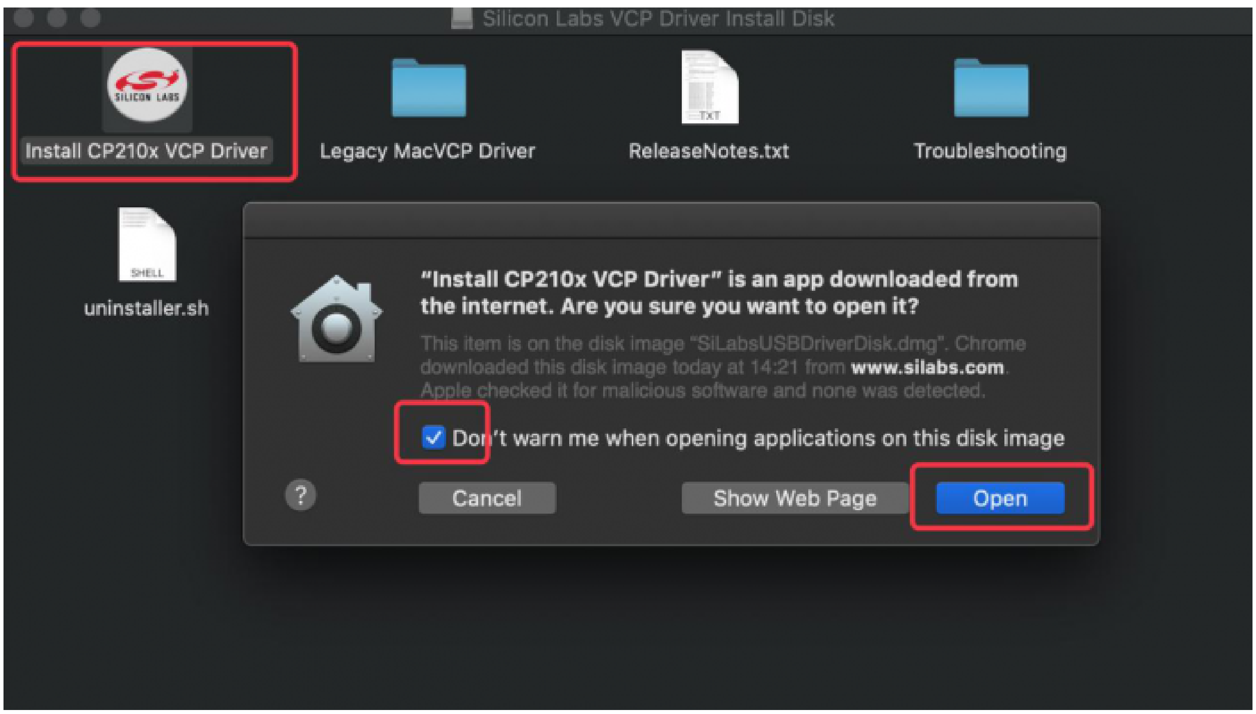
5Open the folder and double-click“SiLabsUSBDriverDisk.dmg”file



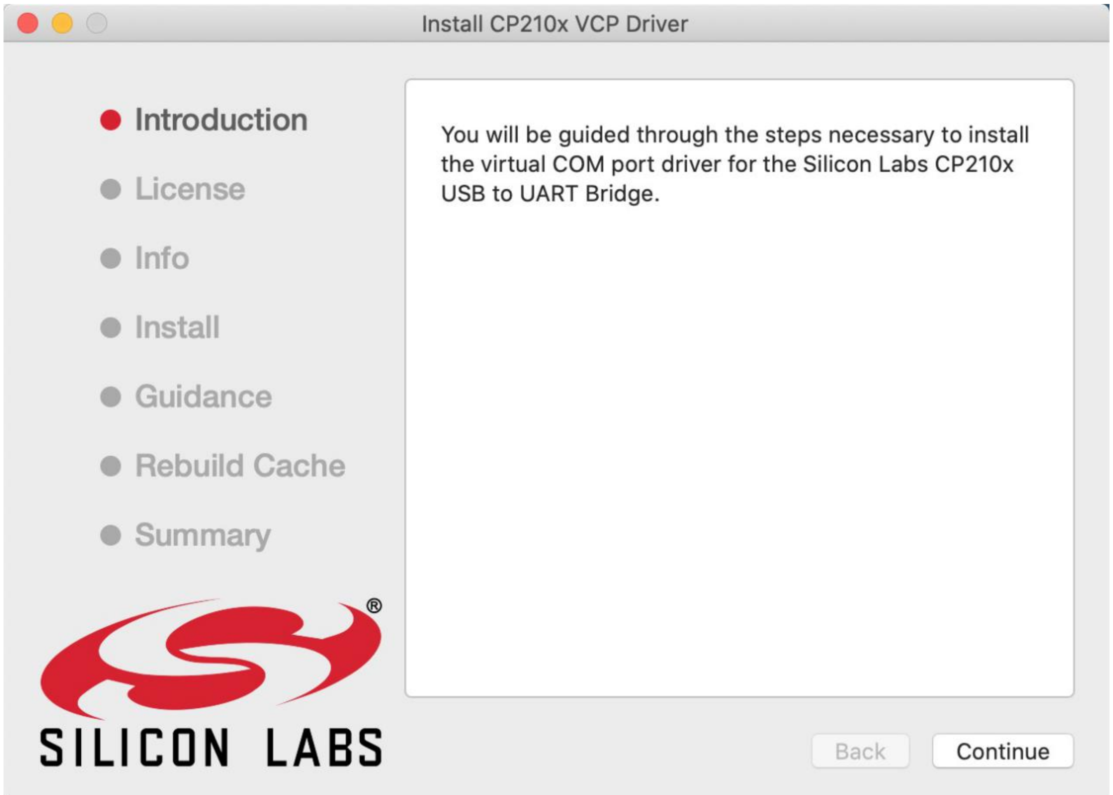
Then you can see the following file



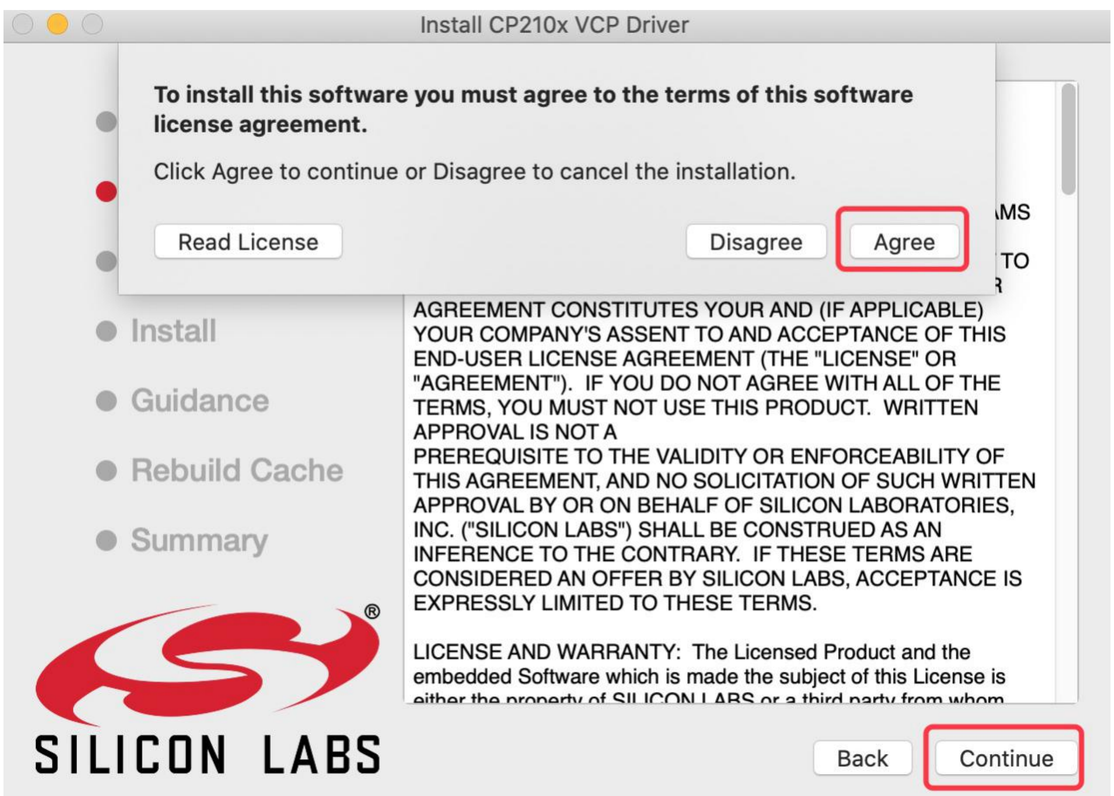
Double-click**"Install CP210x VCP Driver"tap**"**Don't warn me when opening application on this disk image**"and click"**Open**"



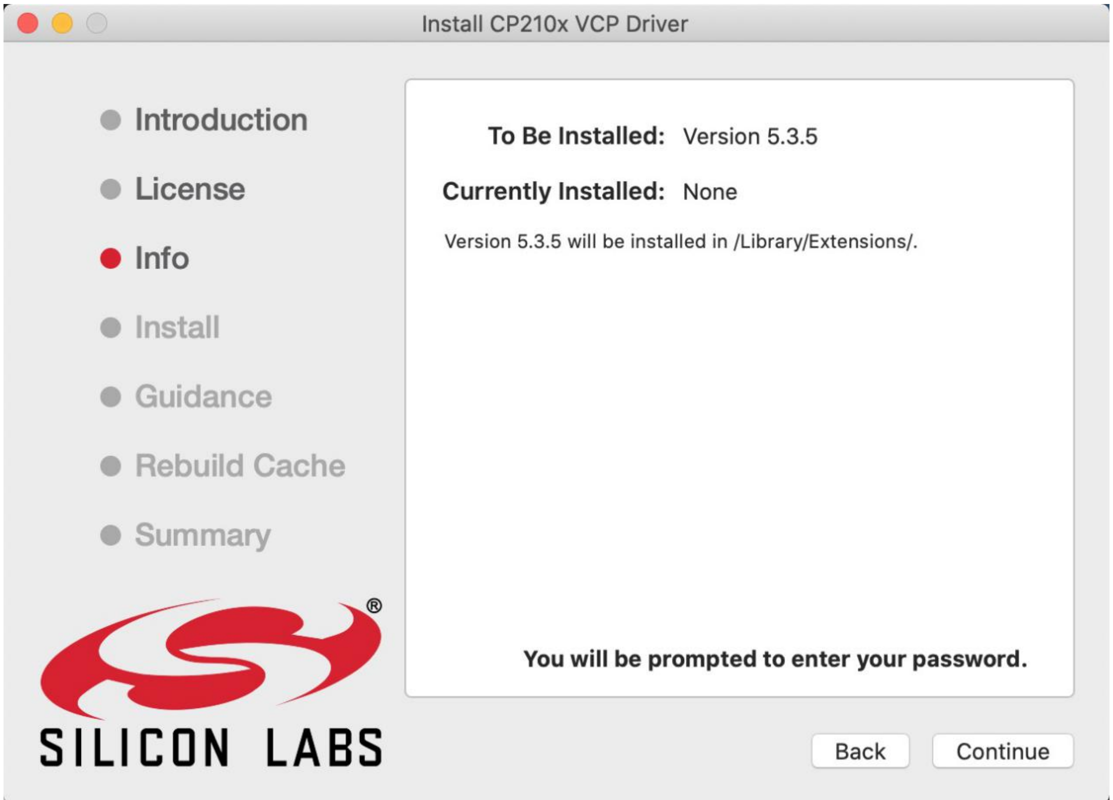
7Click"**Continue**"

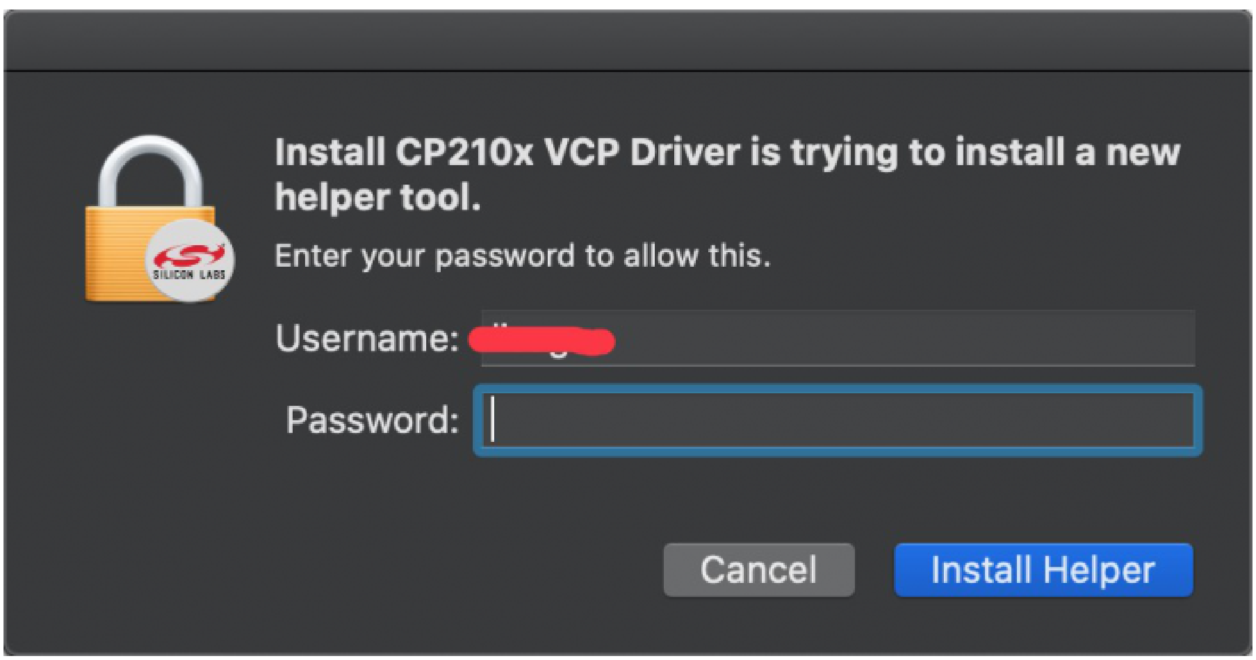


8Click“**Agree**”then tap“**Continue**”

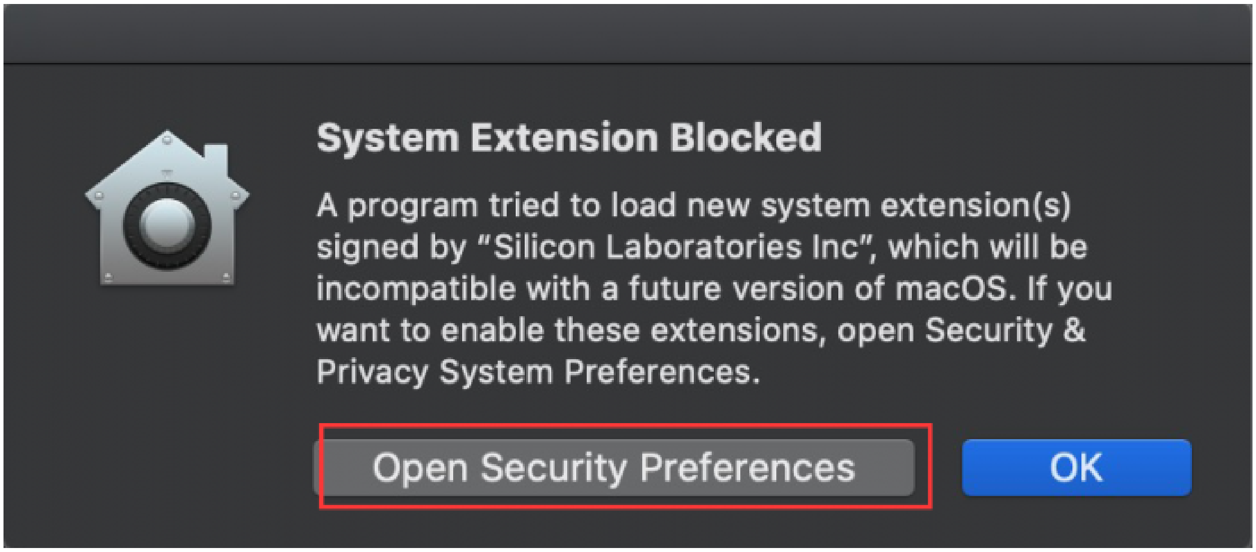


9Click“**Continue**”then input your user password

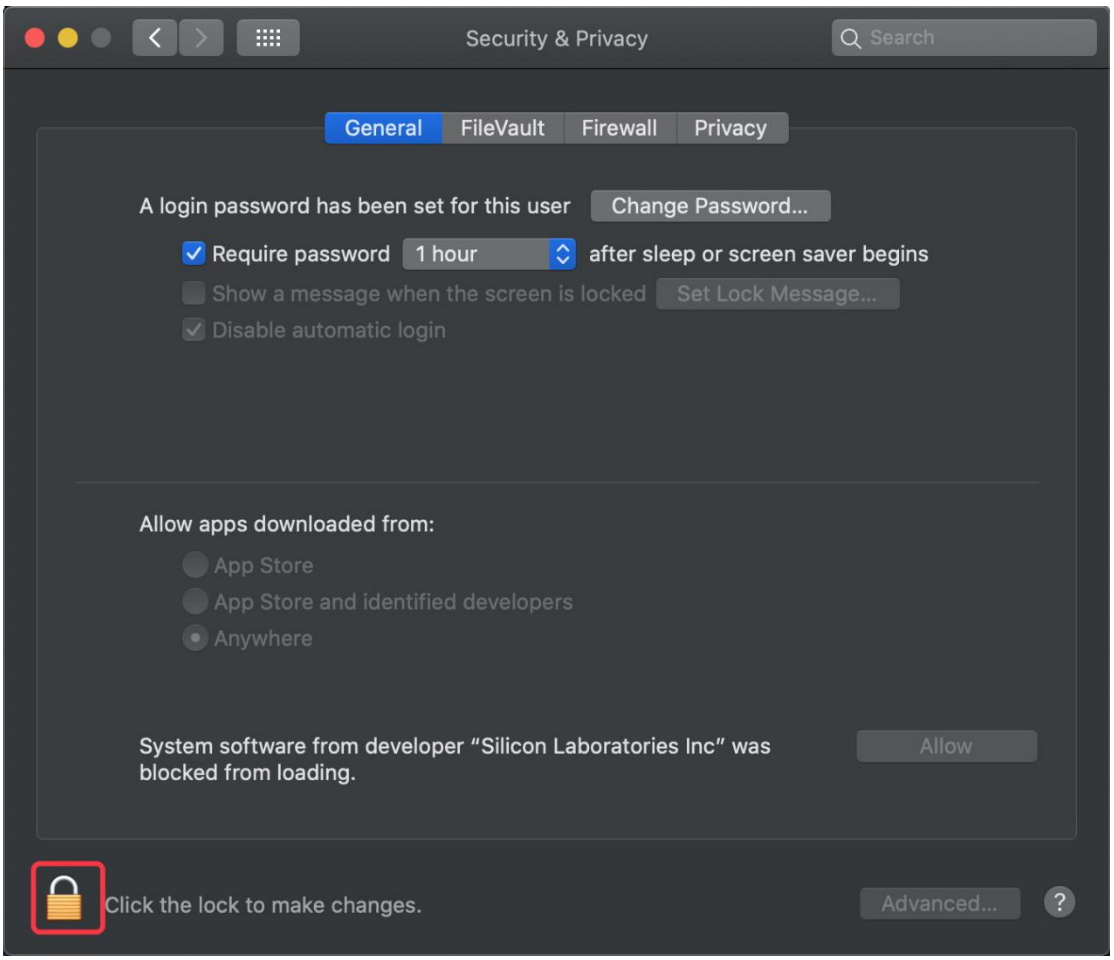


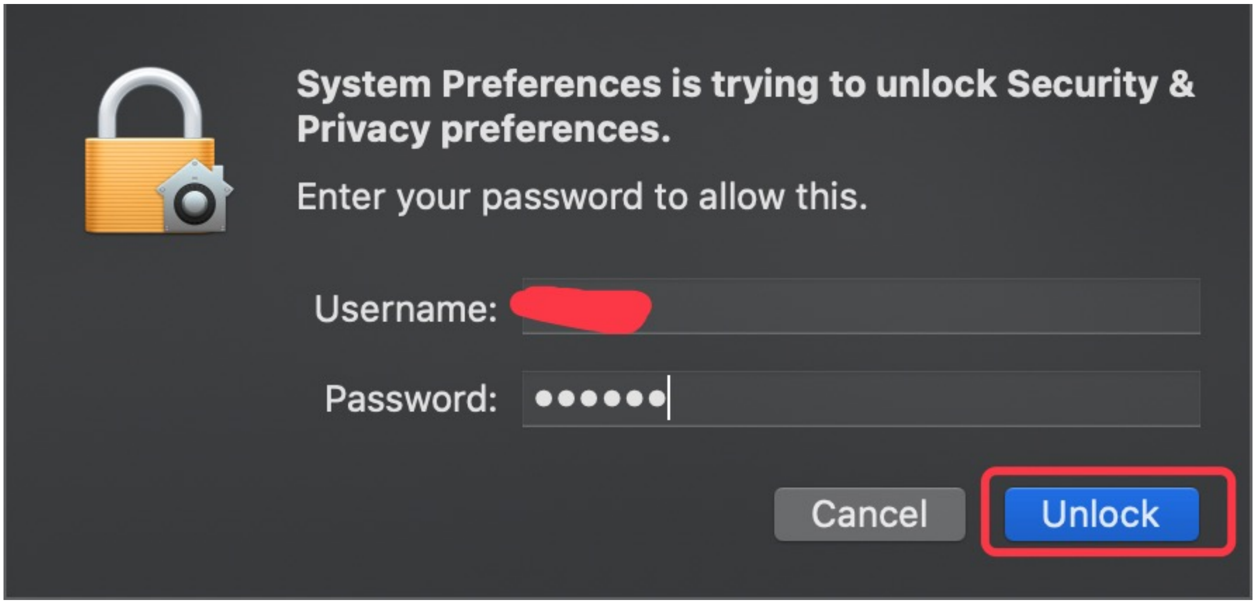


10. Select“**Select Open Security Preferences**”

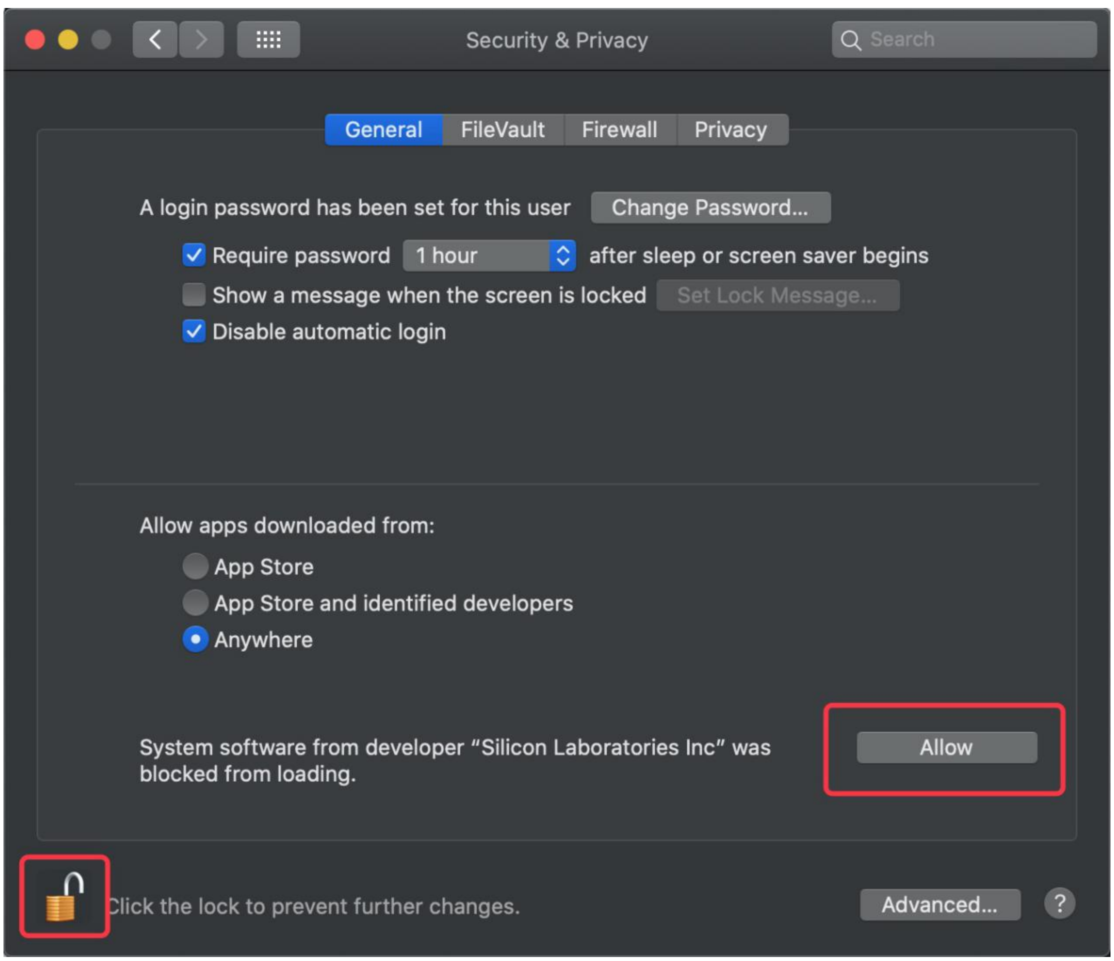


11Click on security lock and enter your user password to authorize.

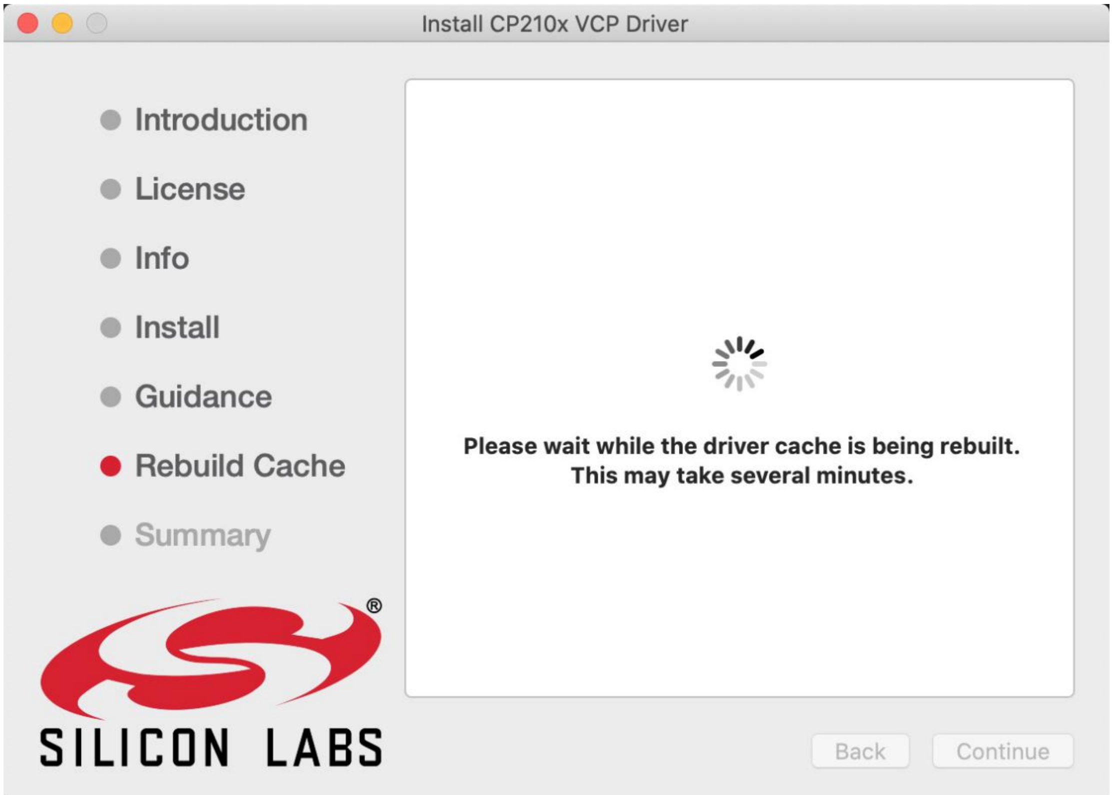




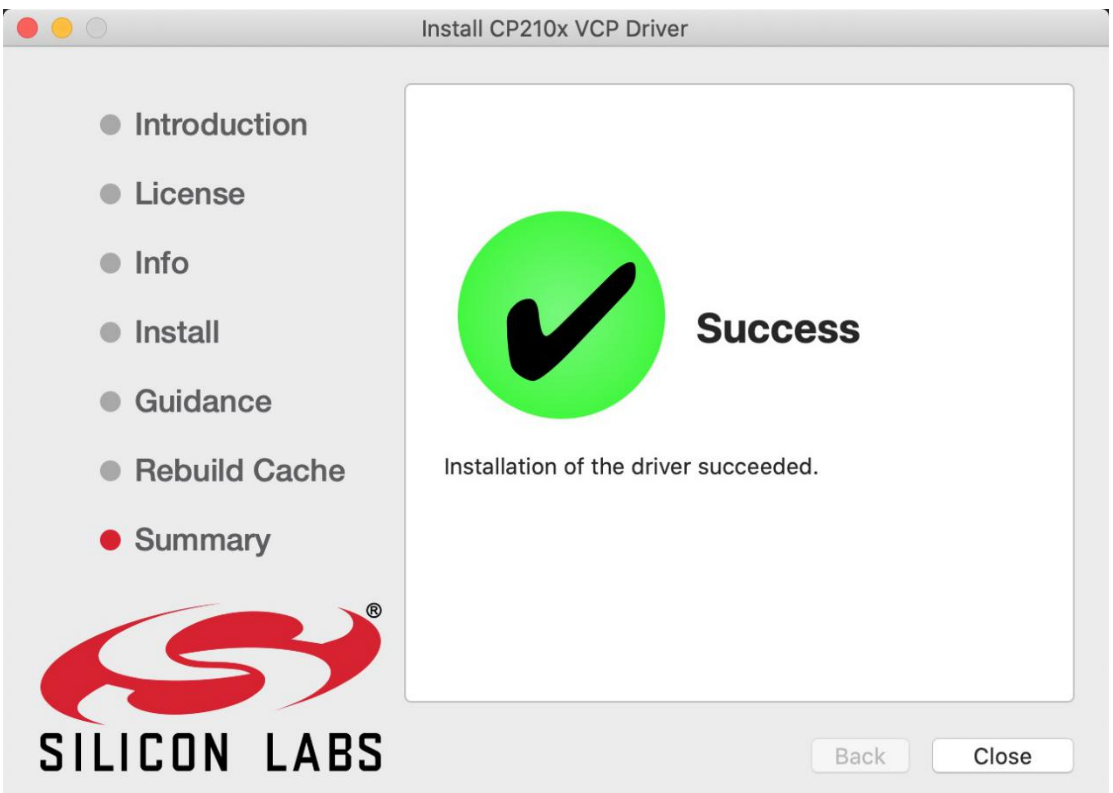
11) When you see that the lock is opened, click "Allow".



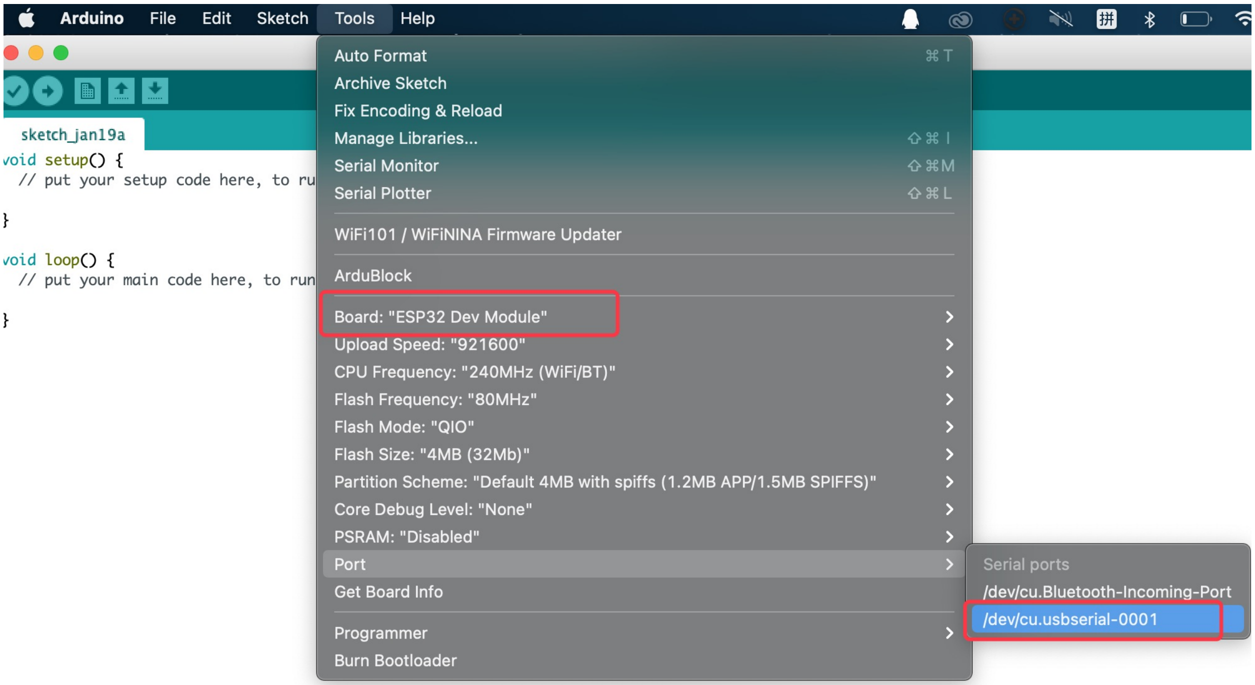
Return to the installation interface and wait for the installation as prompted.



14The installation is successful



15Open arduinoIDEclick“**Tools**”and tap“**ESP32 Dev Module**” and“**/dev/cu.usbserial-0001**”.



16Click  to upload the program, and you can see the program burned successfully

CHAPTER FIVE

ARDUINO TUTORIAL

5.1 Download Arduino code and library files

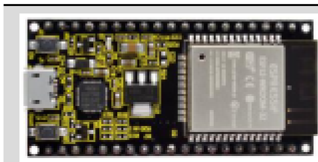
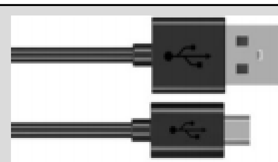
Click on the link to download Arduino code files and library files:Arduino-Codes-and-Libraries

5.2 Project 01: Hello World

5.2.1 1.Introduction

For ESP32 beginners, we'll start with some simple things. In this project, you just need an ESP32 mainboard, a USB cable and a 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.

5.2.2 2.Components

	
ESP32*1	USB Cable*1

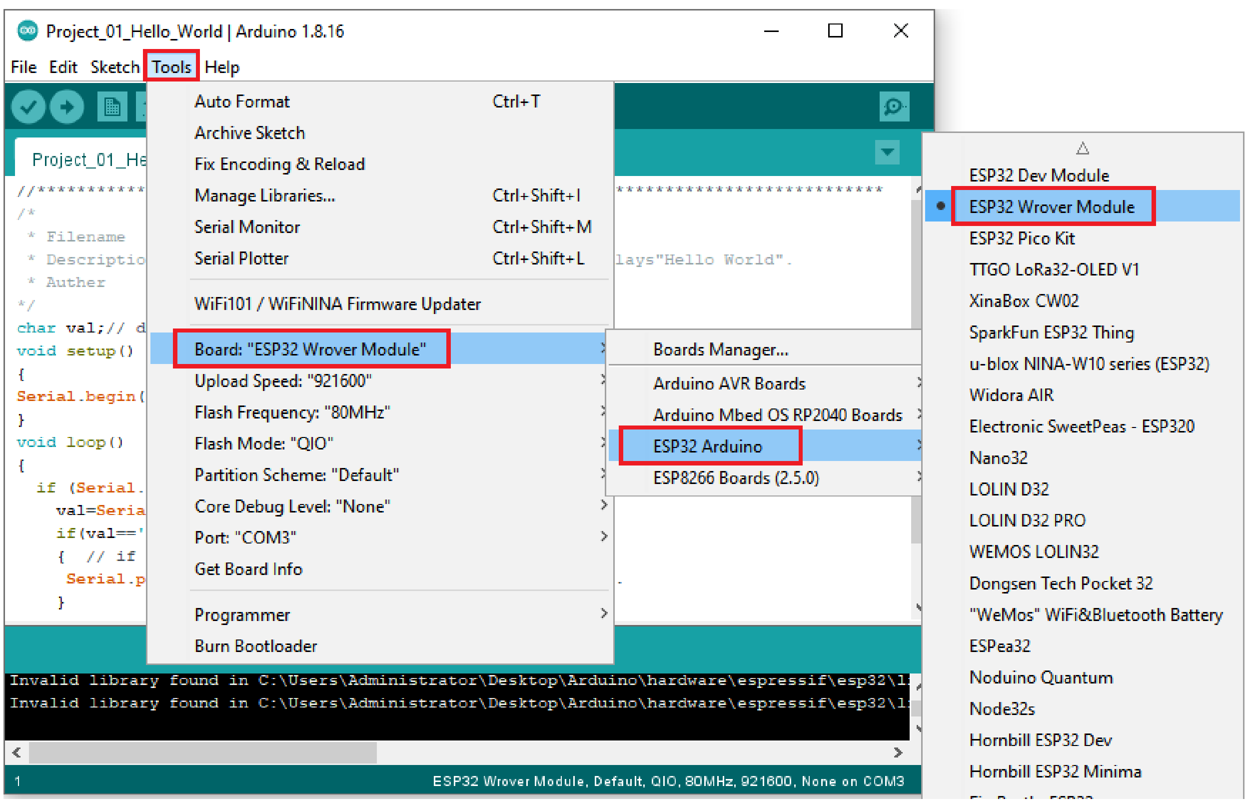
5.2.3 3.Wiring

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

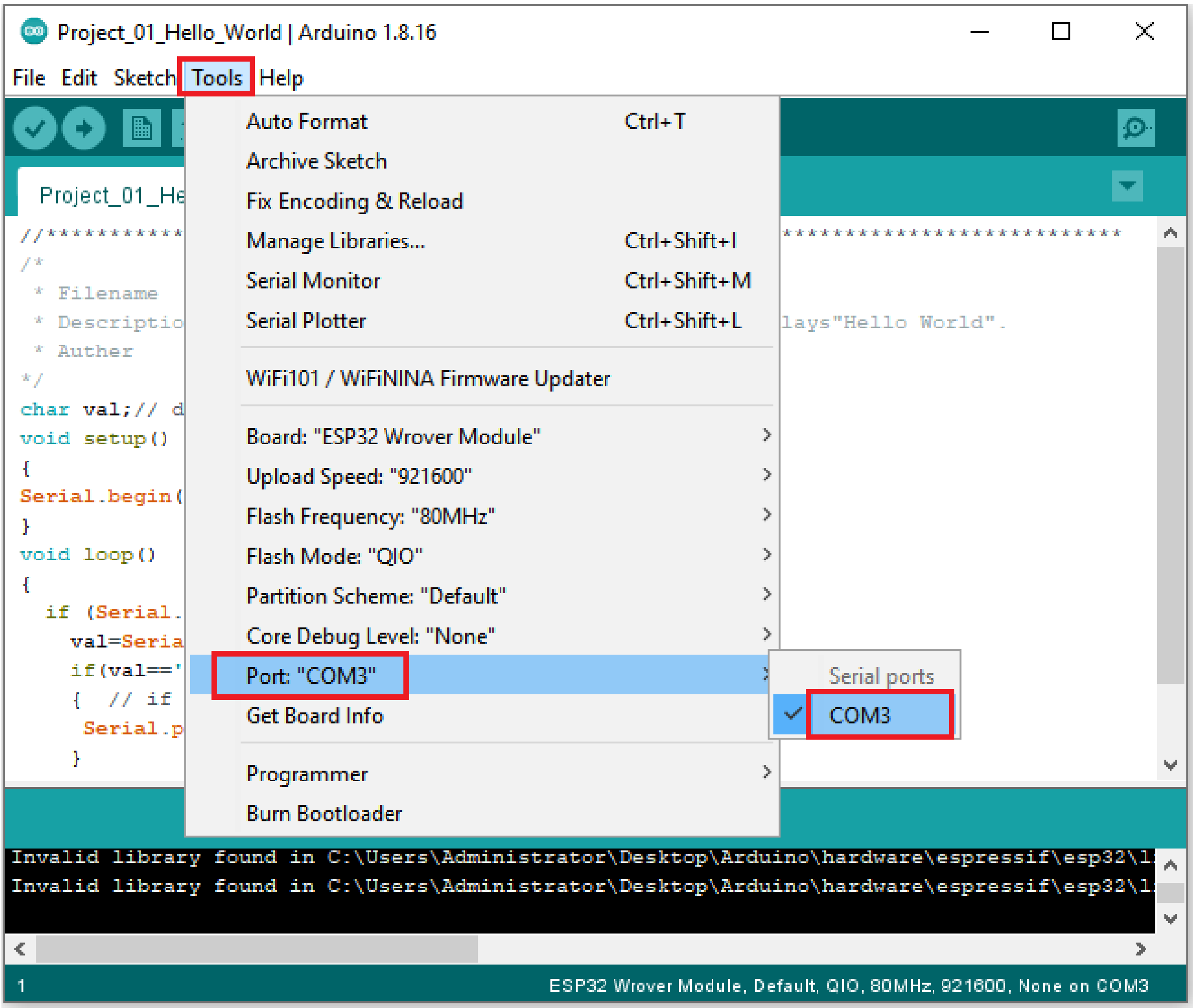
5.2.4 4.Test Code

```
//**********************************************************************************
/*
 * 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 !”.
    }
  }
}
//**********************************************************************************
```

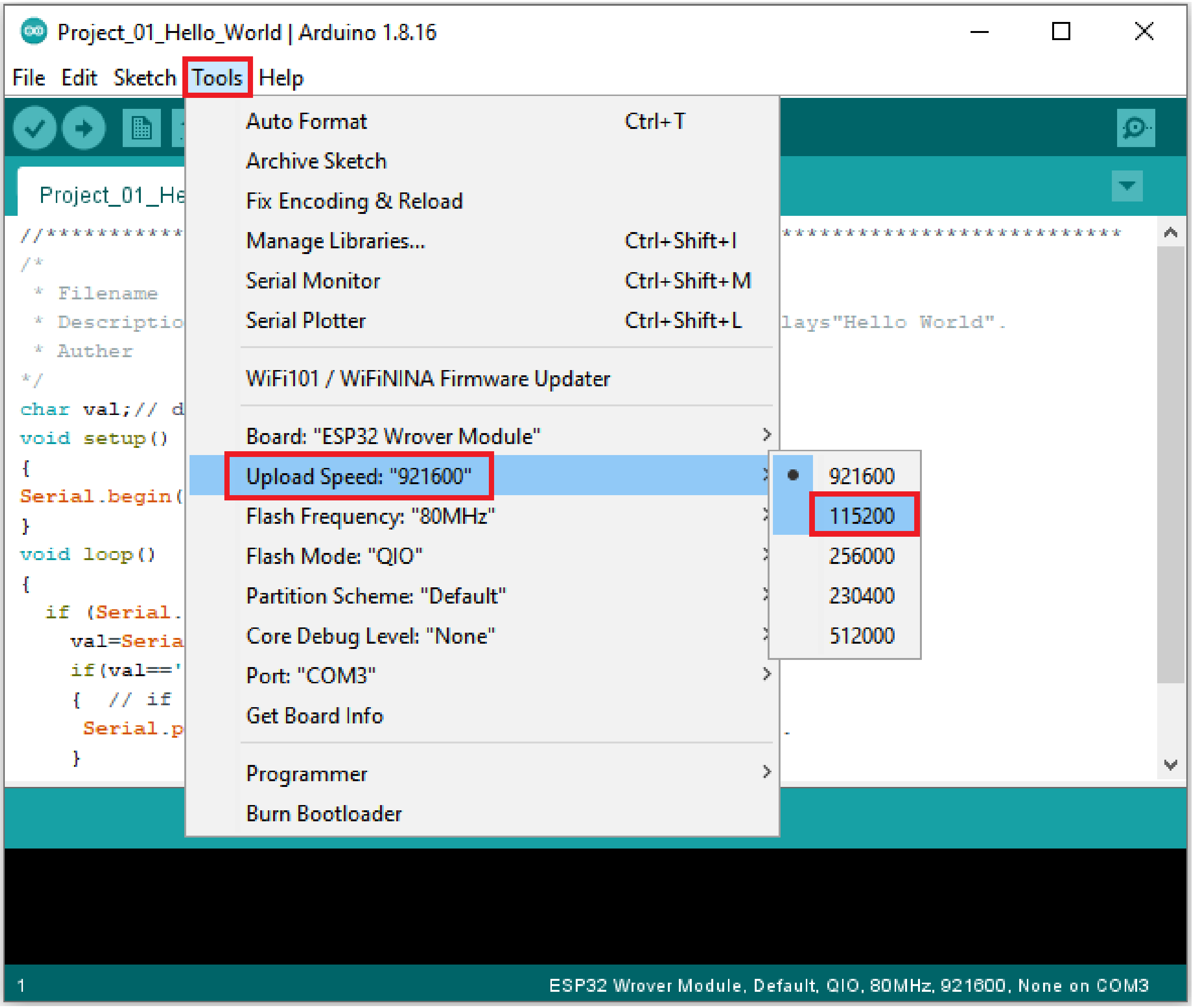
Before uploading the project code to ESP32click“Tools”→“Board” and select“ESP32 Wrover Module”.



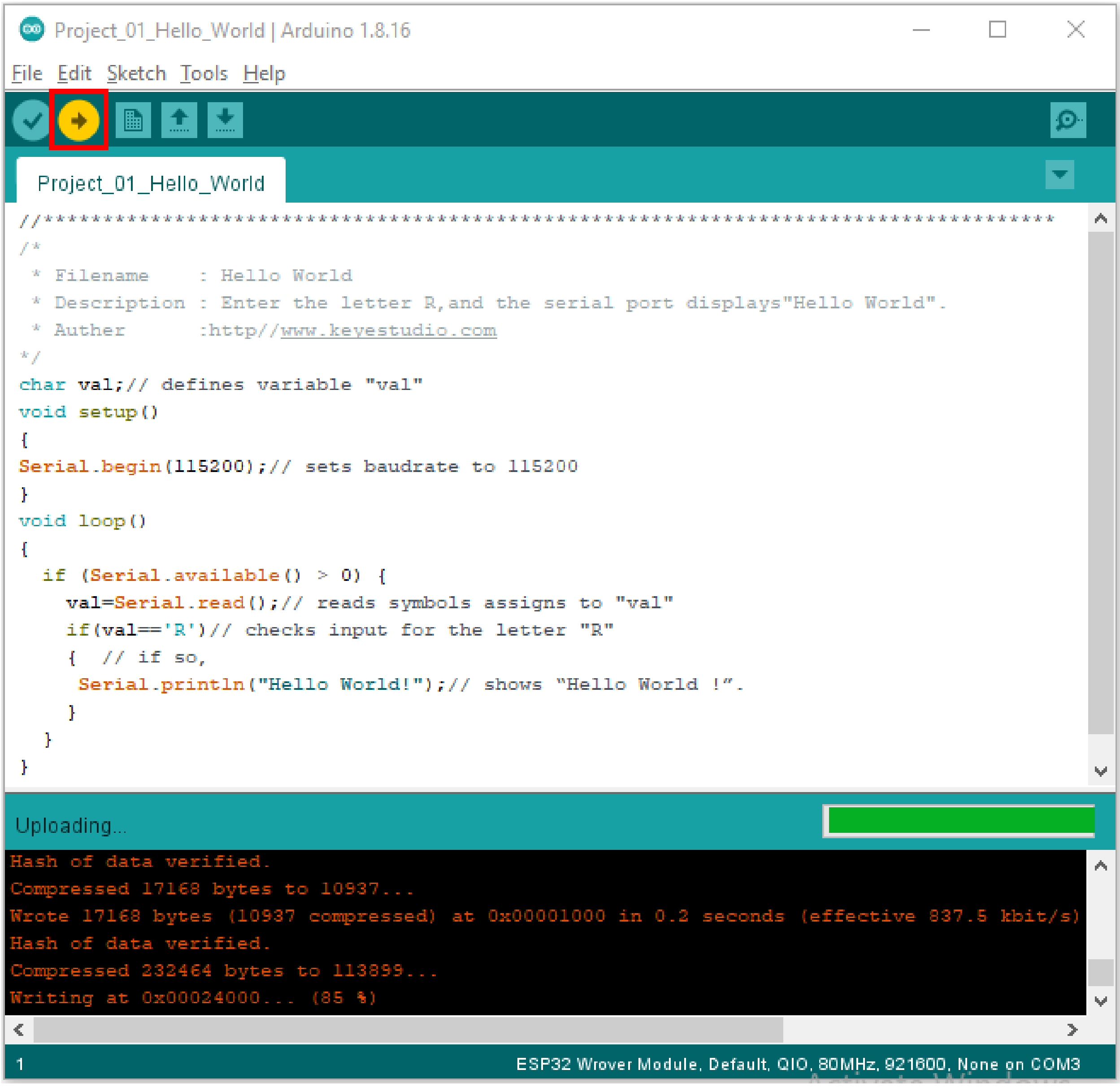
Select the serial port.



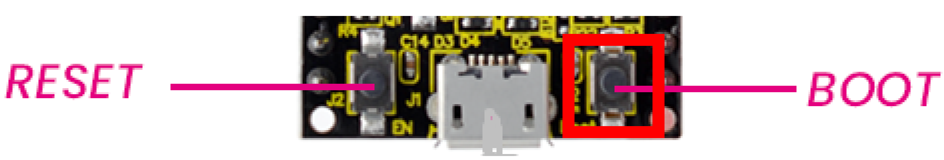
Note: For macOS users, if the uploading fails, please set the baud rate to 115200 before clicking .

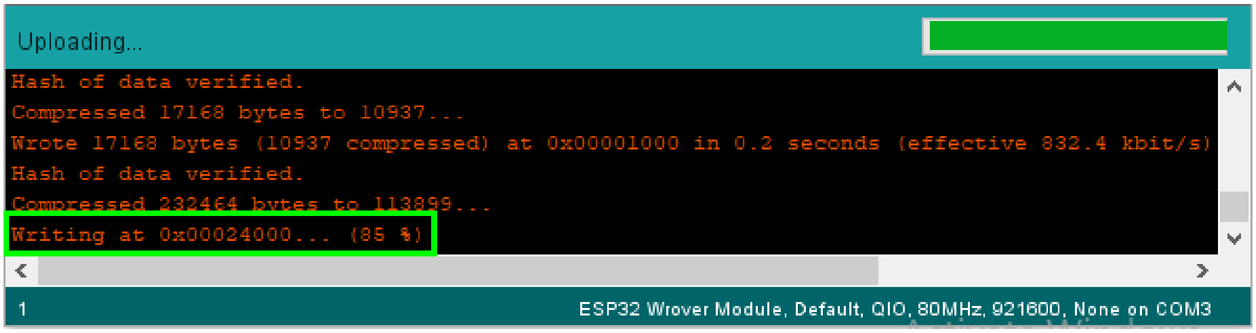


Click  to download the code to ESP32.



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:

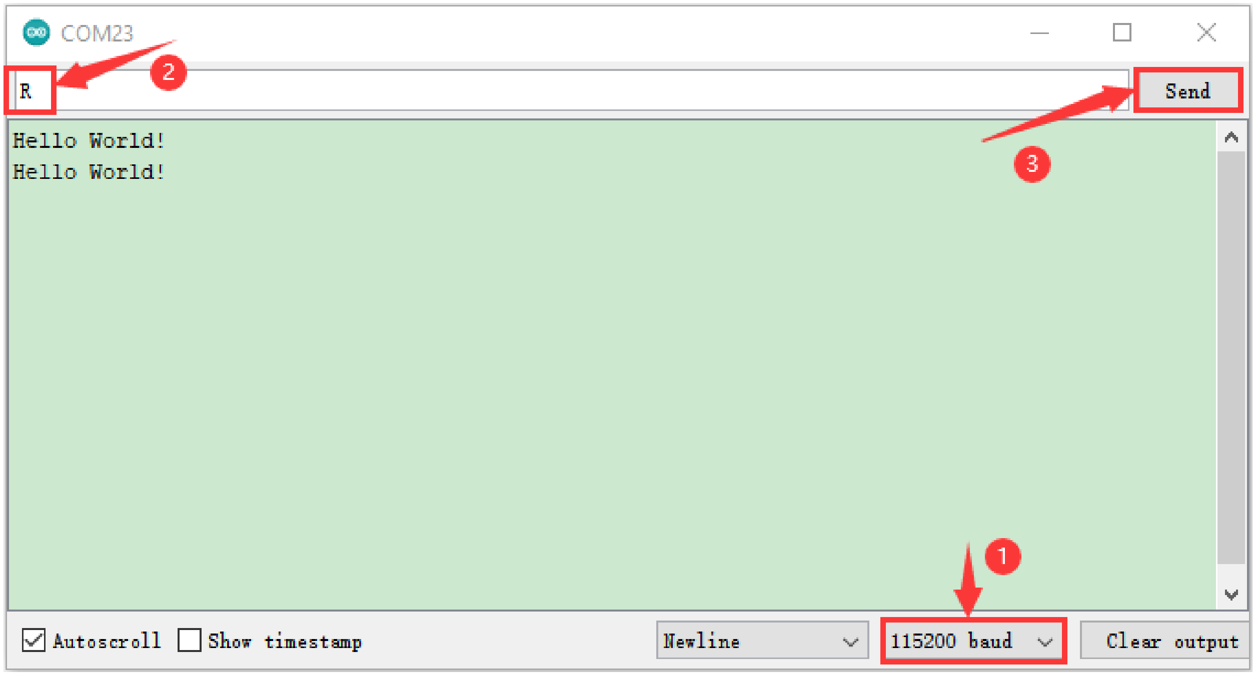




The code is uploaded successfully

5.2.5 5.Test Result

After the 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!".

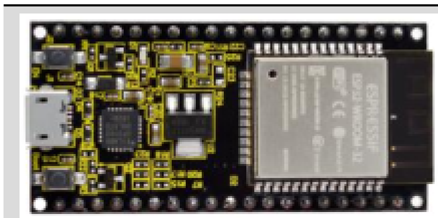
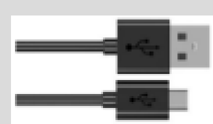
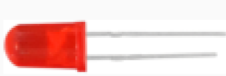
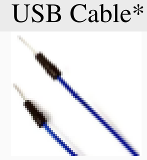


5.3 Project 02: Turn On LED

5.3.1 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.

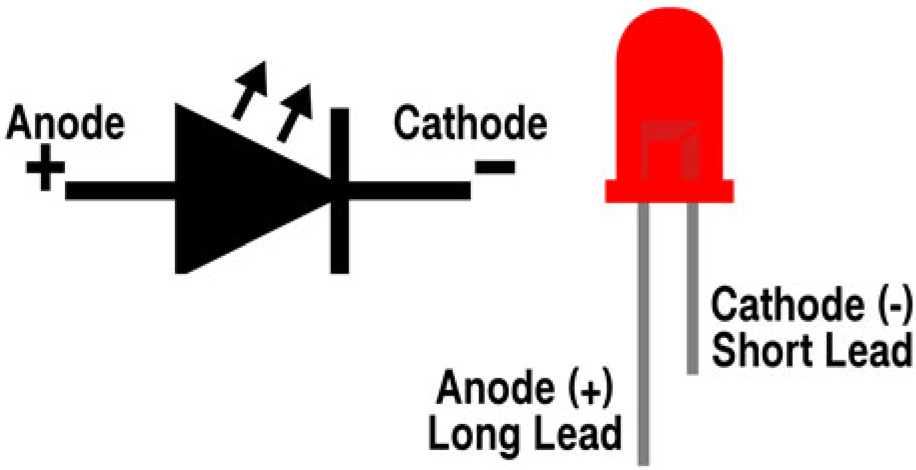
5.3.2 2.Components

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

5.3.3 3.Component Knowledge

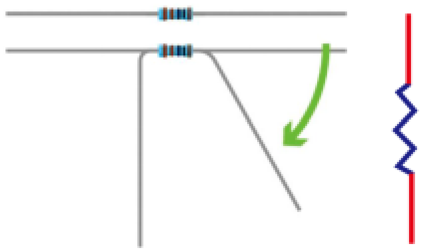
1LED:

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



2Five-band resistor

A resistor is an electronic component in a circuit that restricts or regulates the flow current to 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.

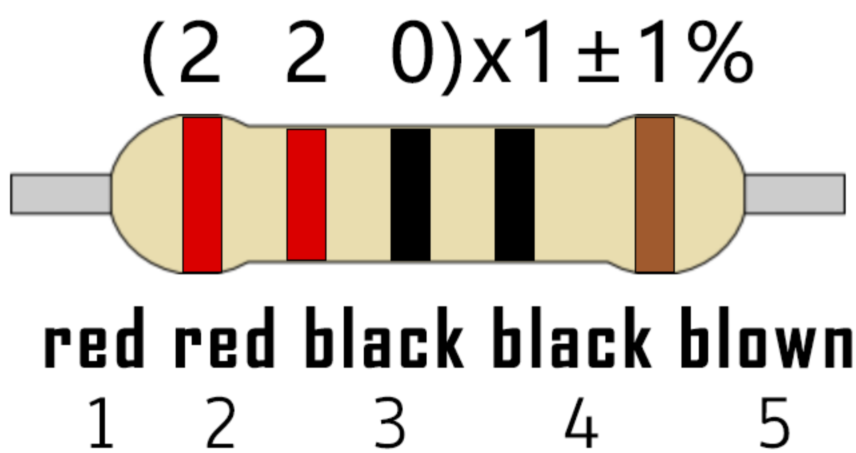
-Color 4 – Multiplier.

-Color 5 – Tolerance.

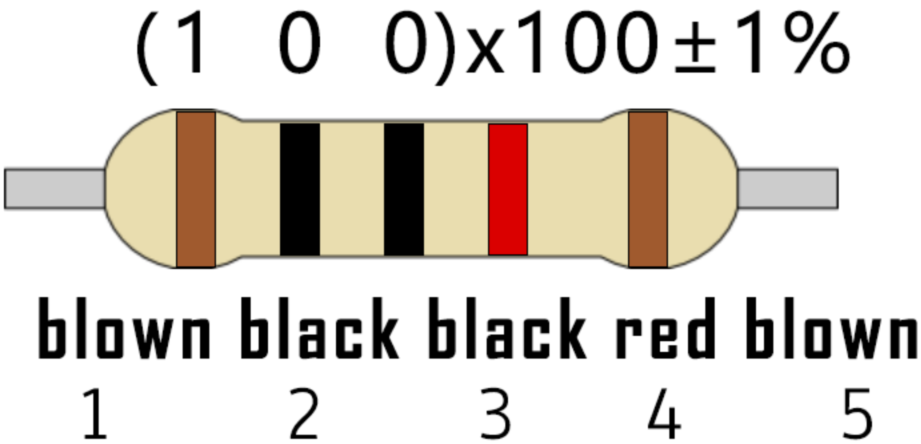
	1st Digit	2nd Digit	3rd Digit	Multiplier	Tolerance
Black		0	0	x1	
Brown	1	1	1	x10	± 1%
Red	2	2	2	x100	± 2%
Orange	3	3	3	x1K	± 3%
Yellow	4	4	4	x10K	± 4%
Green	5	5	5	x100K	± 0.5%
Blue	6	6	6	x1M	± 0.25%
Violet	7	7	7	x10M	± 0.10%
Grey	8	8	8	x100M	± 0.05%
White	9	9	9	x1G	
Gold				÷ 10	± 5%
Silver				÷ 100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

In this kit, we provide three five-band resistors with different resistance values. We three five-band resistors as an example.

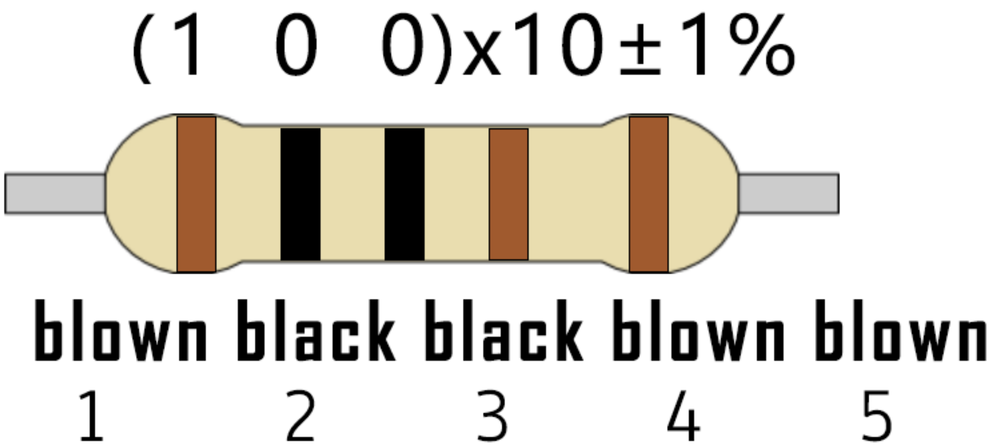
220 Resistor*10



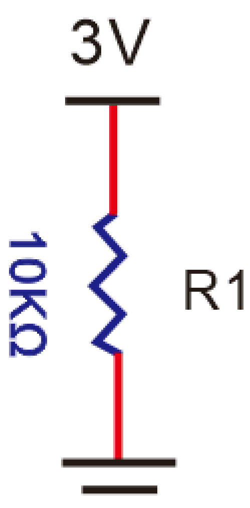
10K Resistor*10



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.

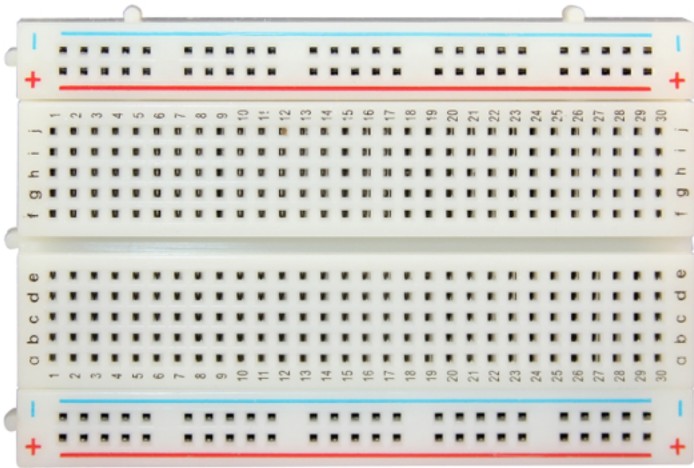


Don't connect a low resistance directly to the two poles of the power supply, which will cause excessive current to

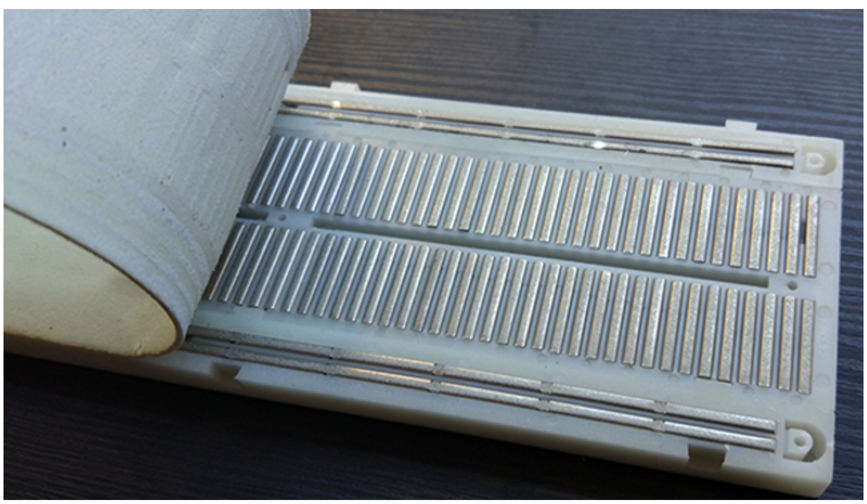
damage the electronic components. Resistors do not have positive and negative poles.

3Bread board

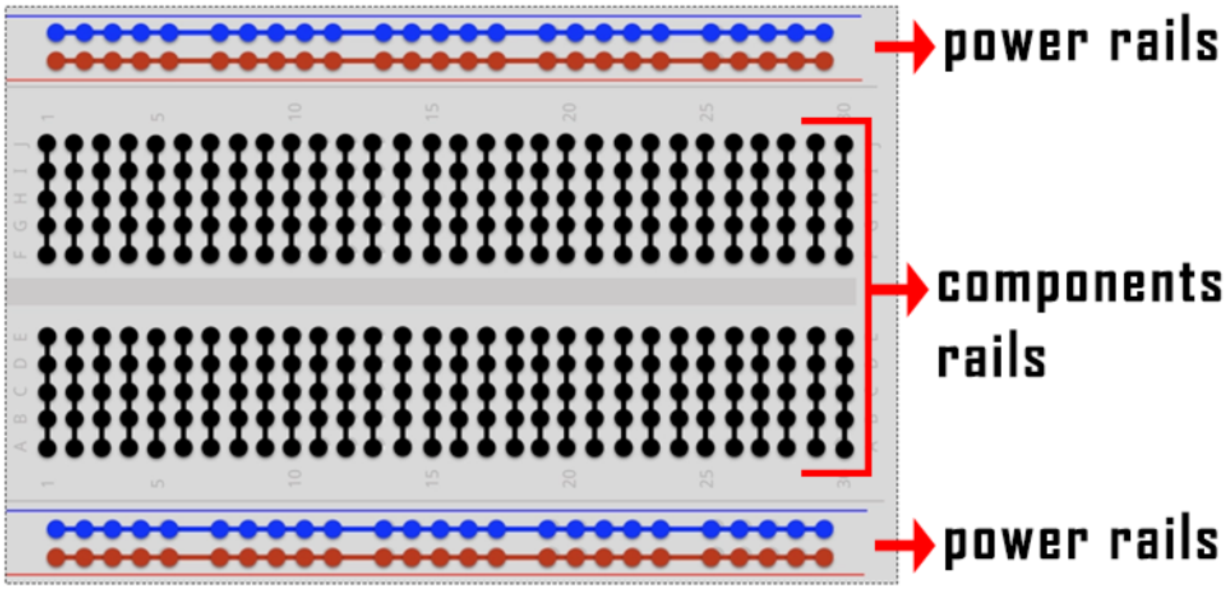
Breadboards are used to build and test circuits quickly before completing any circuit design. There are many holes in the breadboard that can be inserted into circuit components such as integrated circuits and resistors. A typical breadboard is shown below



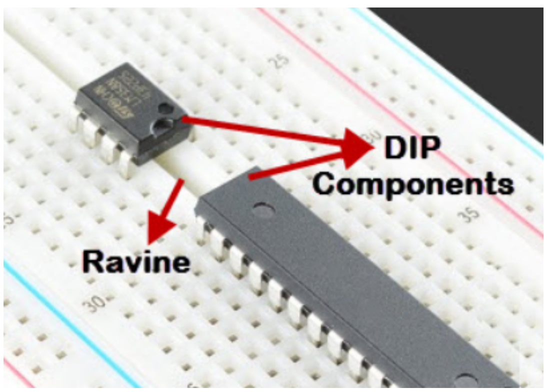
The breadboard has strips of metal , which run underneath the board and connect the holes on the top of the board. The metal strips are laid out as shown below. Note that the top and bottom rows of holes are connected horizontallywhile the remaining holes are connected vertically.

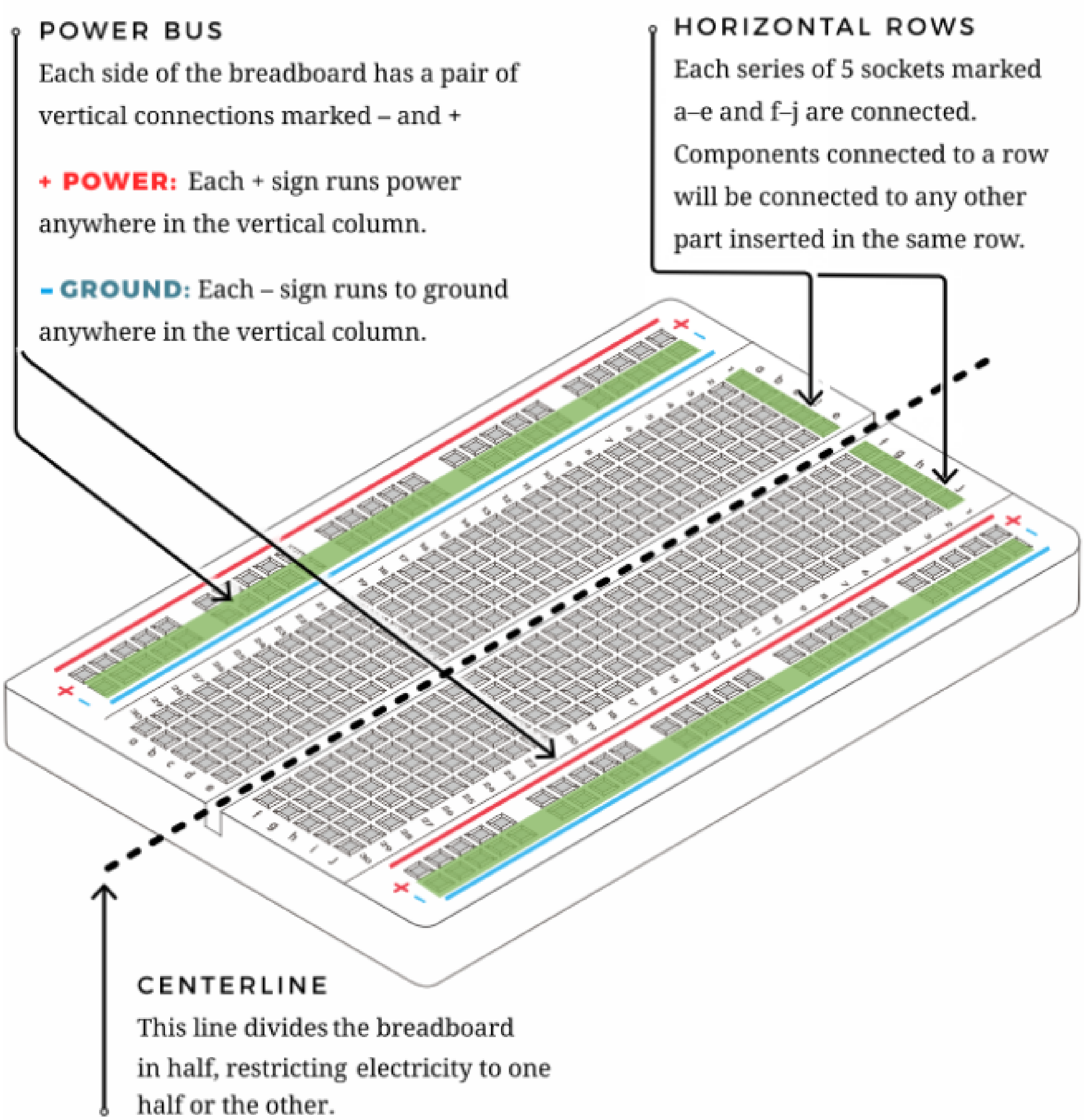


The first two rows (top) and the last two rows (bottom) of the breadboard are used for the positive pole (+) and negative pole (-) of the power supply respectively. The conductive layout of the breadboard is shown in the figure below:



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:




POWER BUS
Each side of the breadboard has a pair of vertical connections marked – and +
+ POWER: Each + sign runs power anywhere in the vertical column.
– GROUND: Each – sign runs to ground anywhere in the vertical column.
HORIZONTAL ROWS
Each series of 5 sockets marked a–e and f–j are connected. Components connected to a row will be connected to any other part inserted in the same row.
CENTERLINE
This line divides the breadboard in half, restricting electricity to one half or the other.

4Power Supply

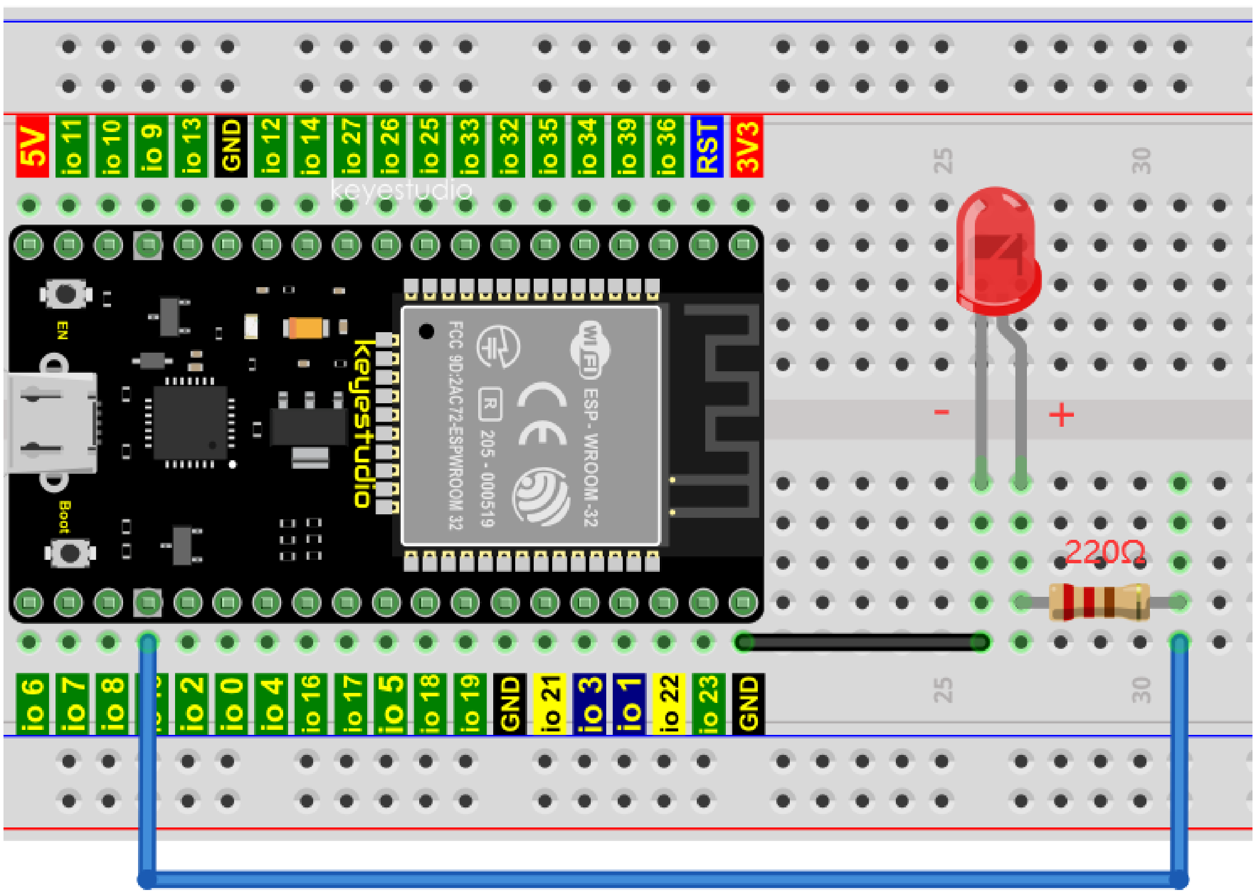
The ESP32 needs 3.3V-5V power supply. In this project, we will connect the ESP32 to the computer via an USB cable.

5.3.4 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 correctly, connect the ESP32 to your computer via a USB cable.

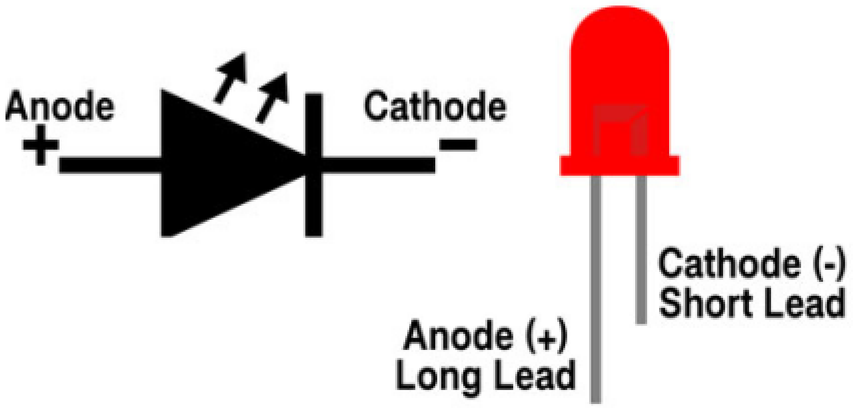
Note: Avoid any possible short circuits (especially connecting 3.3V and GND)!

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

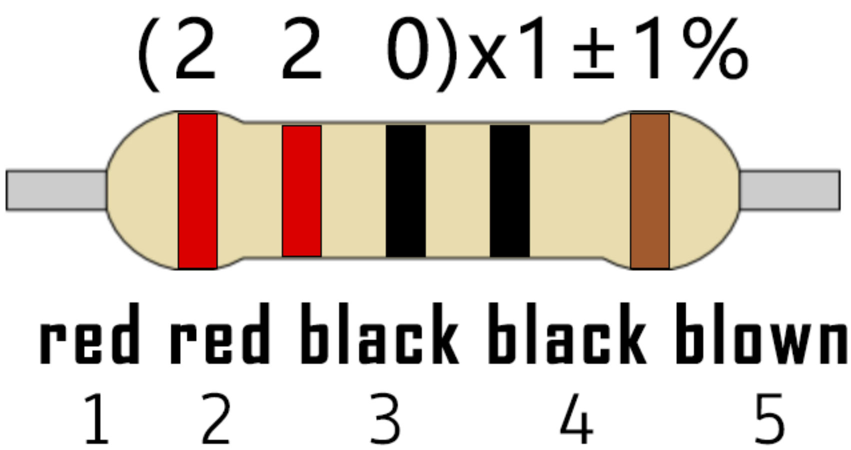


Note:

How to connect a LED



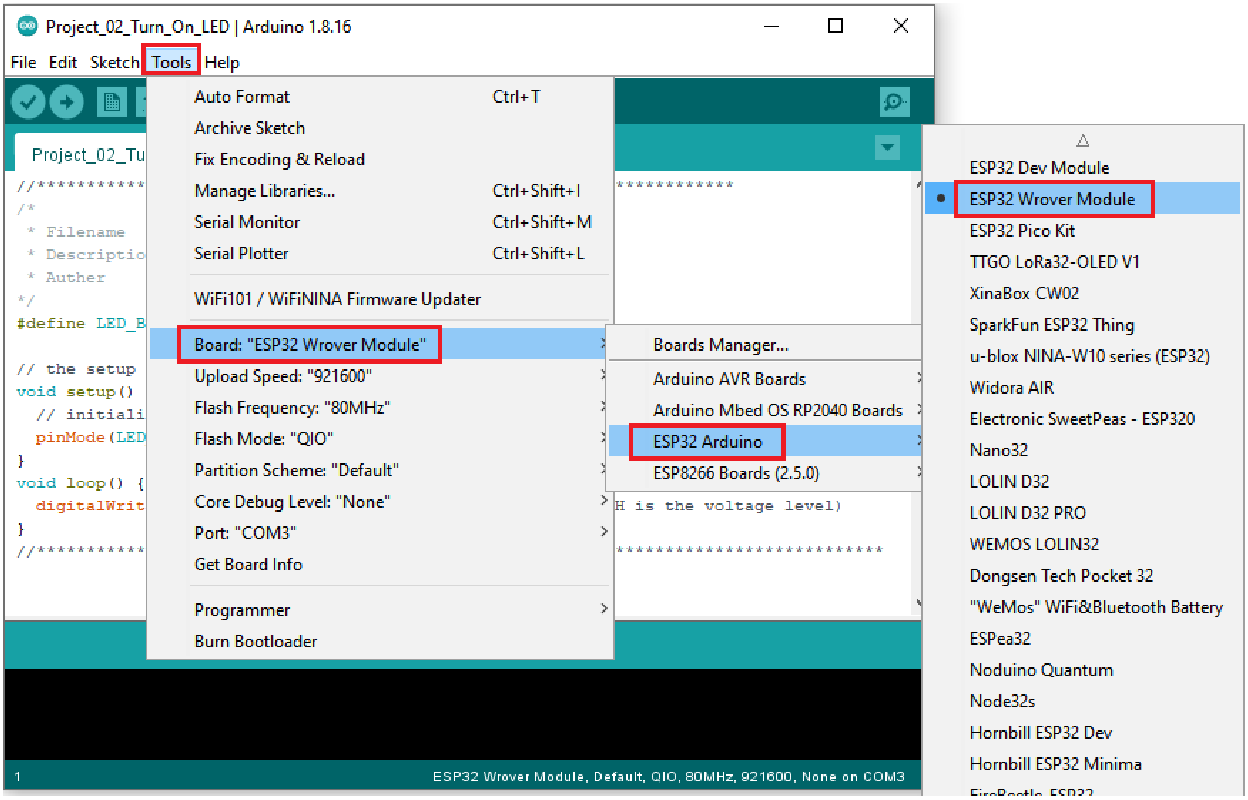
How to identify the 220 Five-band resistor



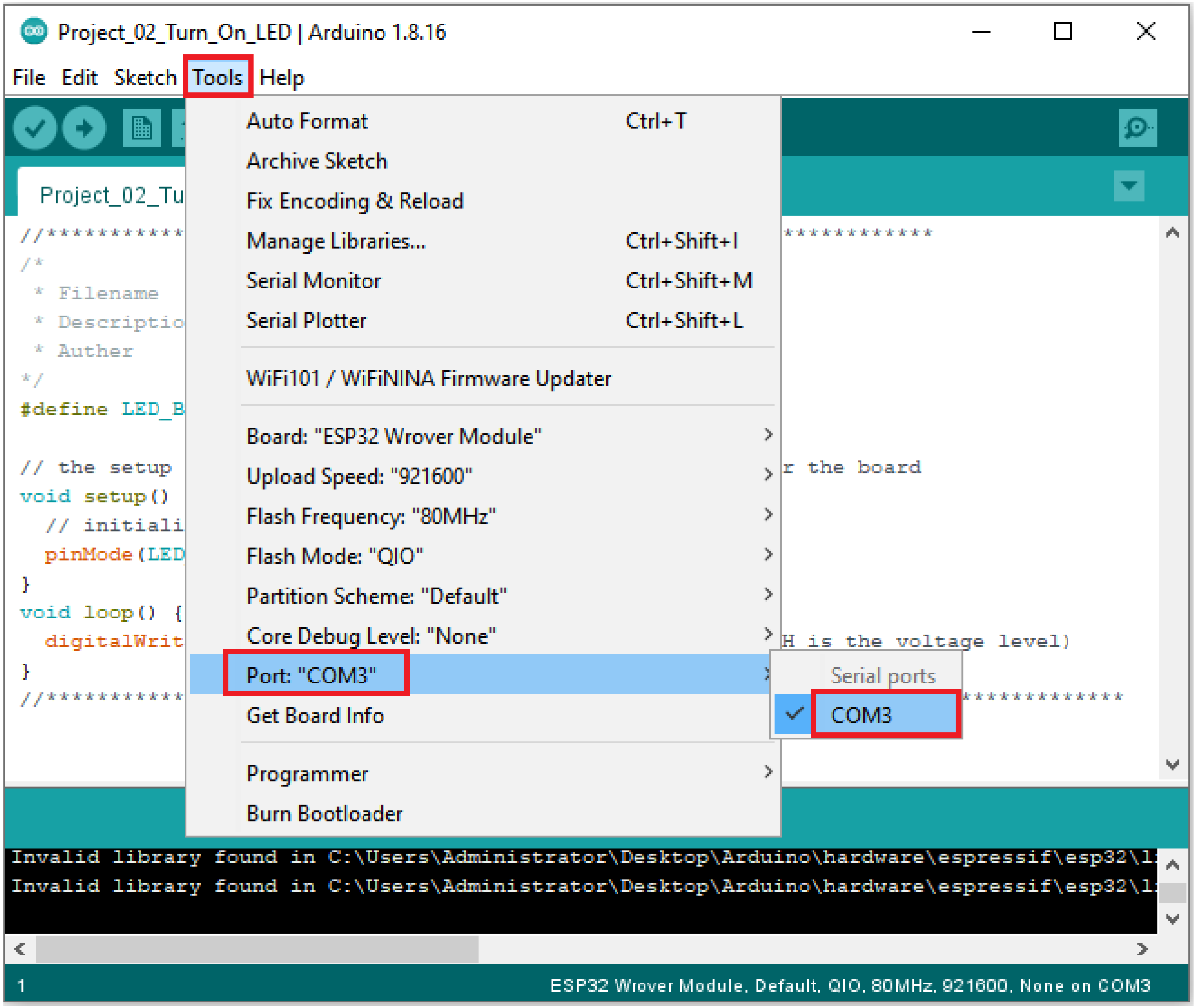
5.3.5 5.Test Code

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

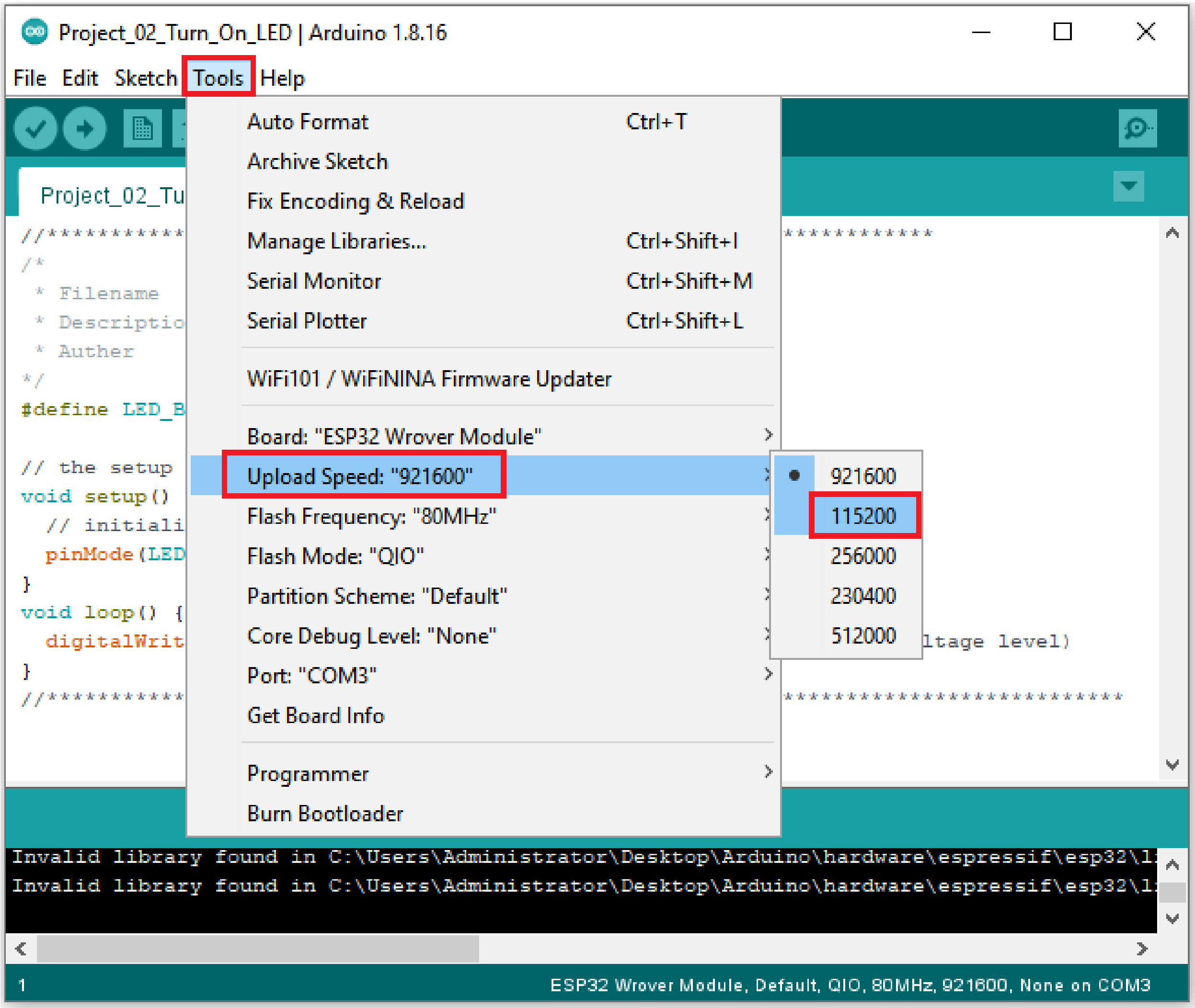
Before uploading the code to ESP32click"Tools"→"Board" and select"ESP32 Wrover Module".



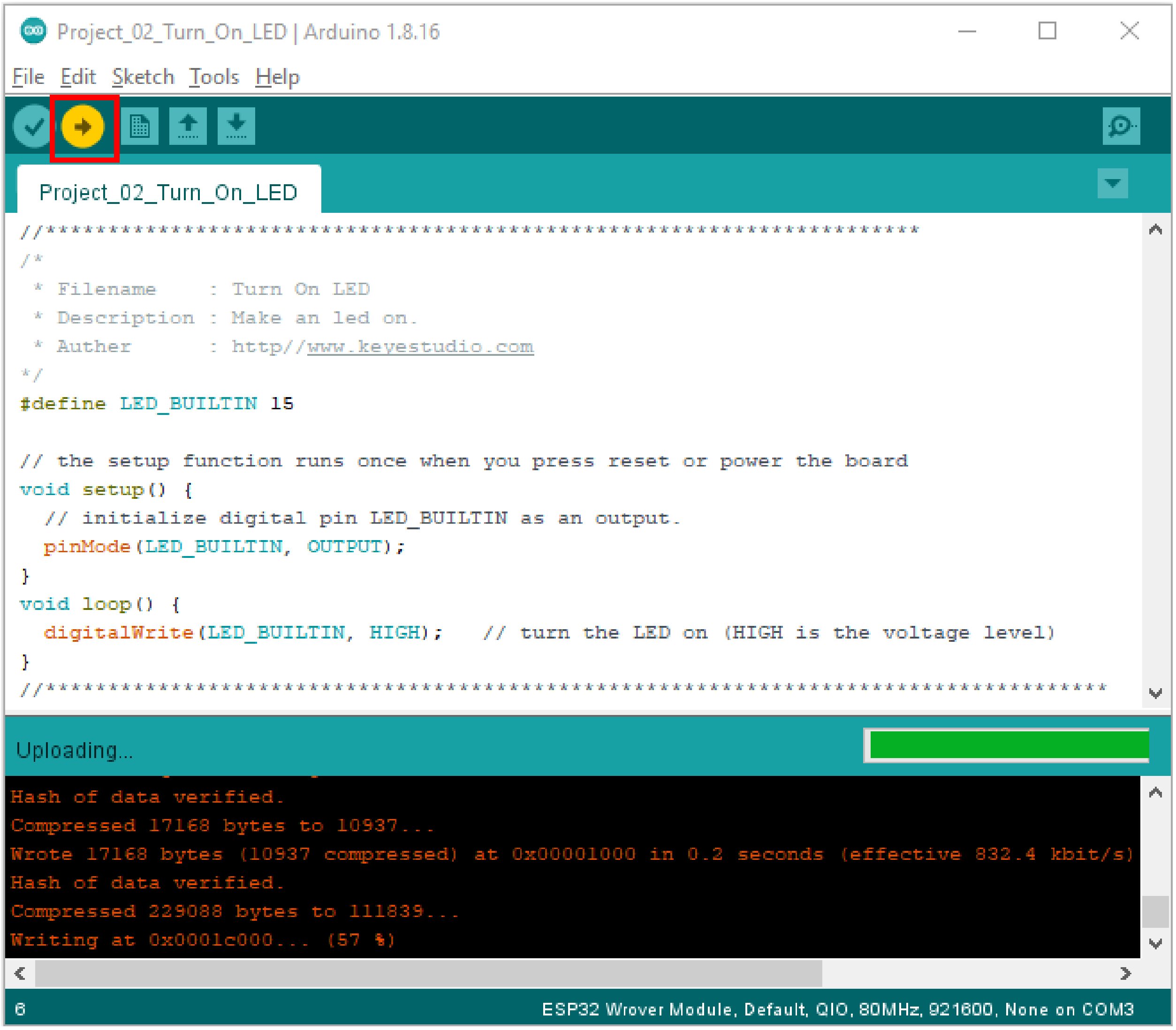
Select the serial port.



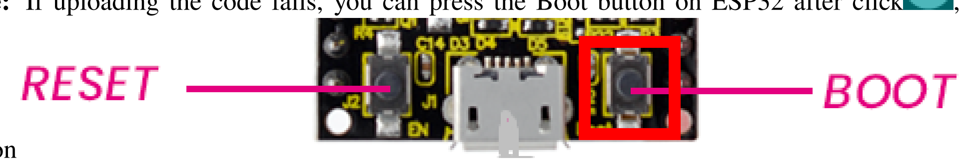
Note: For macOS users, if the uploading fails, please set the baud rate to 115200 before clicking .



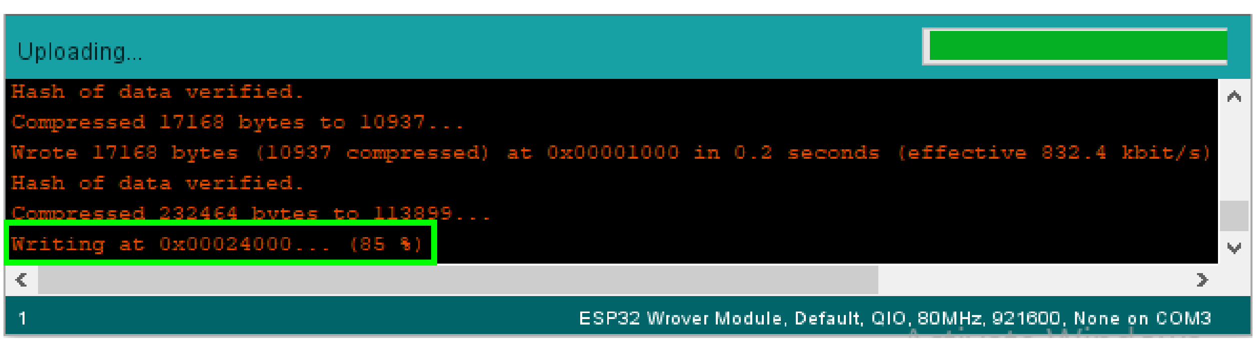
Click  to download the code to ESP32.



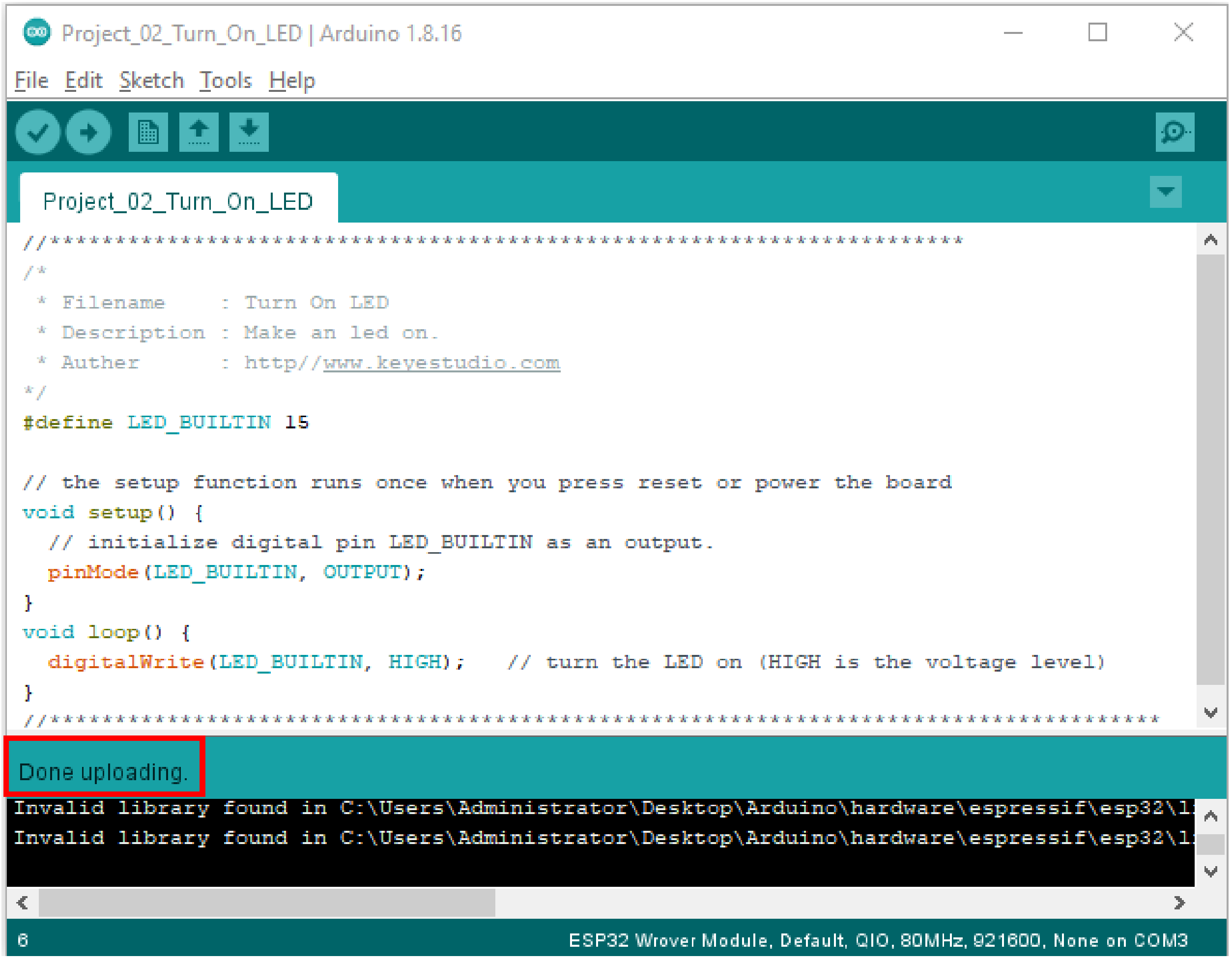
Note: If uploading the code fails, you can press the Boot button on ESP32 after click , and release the Boot button



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

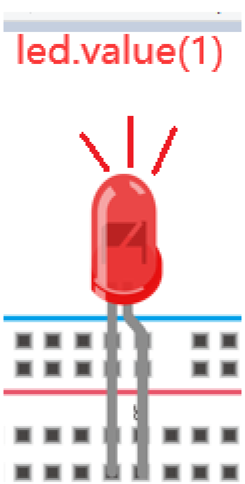


The code is uploaded successfully



5.3.6 6.Test Result

After uploading the code successfully, power up with a USB cable and the LED will lit up.

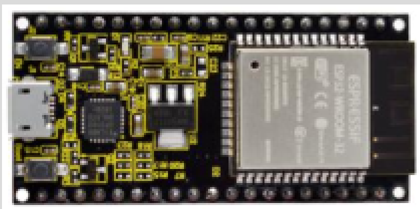
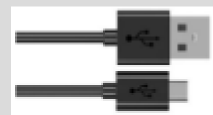
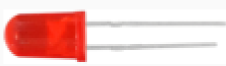
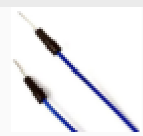


5.4 Project 03LED Flashing

5.4.1 1.Introduction

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

5.4.2 2.Components

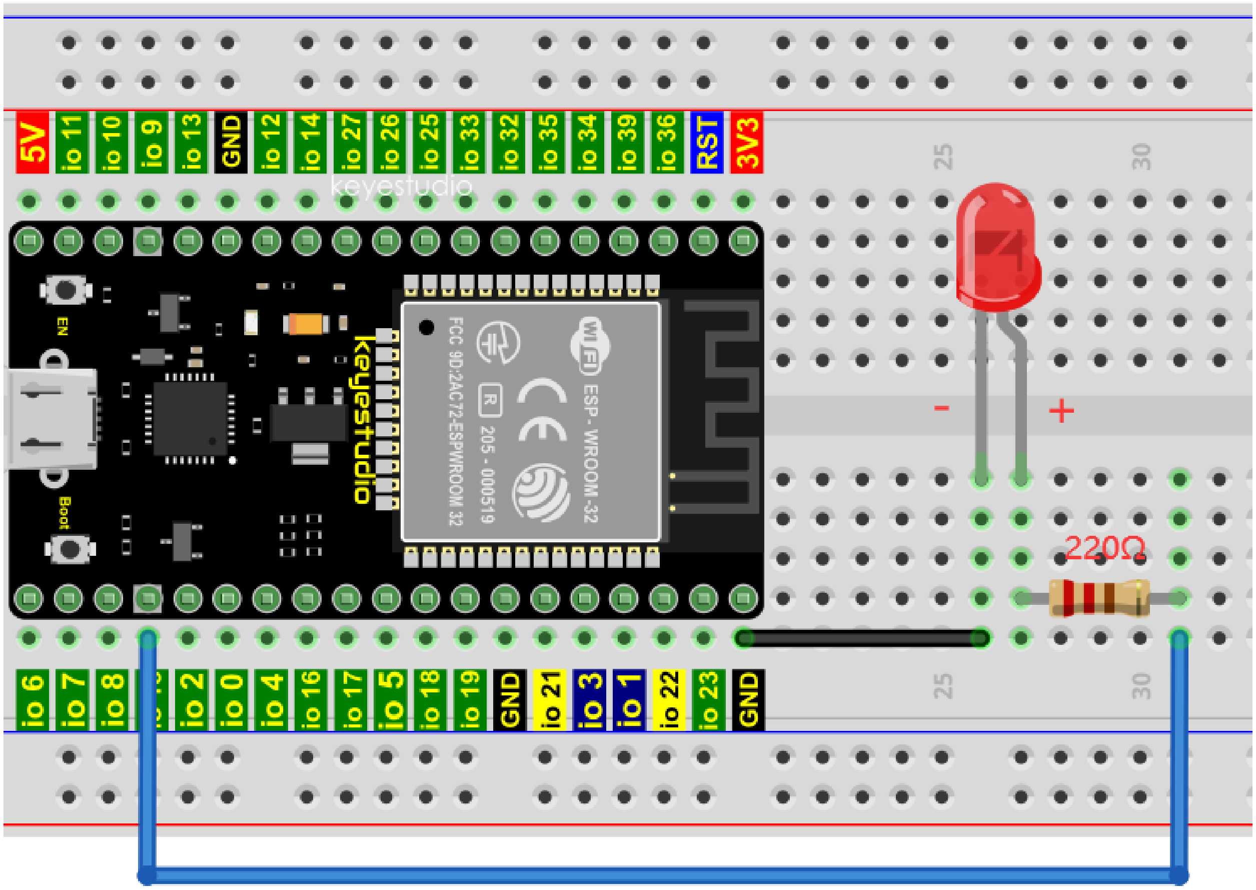
		
ESP32*1	Breadboard*1	USB Cable*1
		
Red LED*1	220 Resistor*1	Jumper Wire*2

5.4.3 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 correctly, connect the ESP32 to your computer via a USB cable.

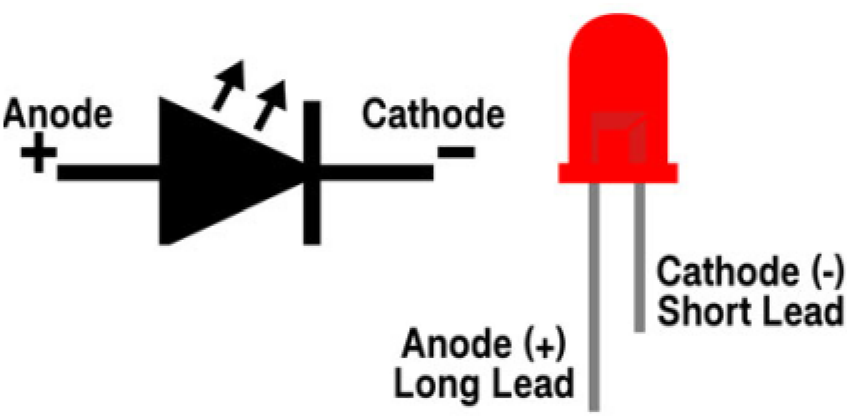
Note: Avoid any possible short circuits (especially connecting 3.3V and GND)!

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

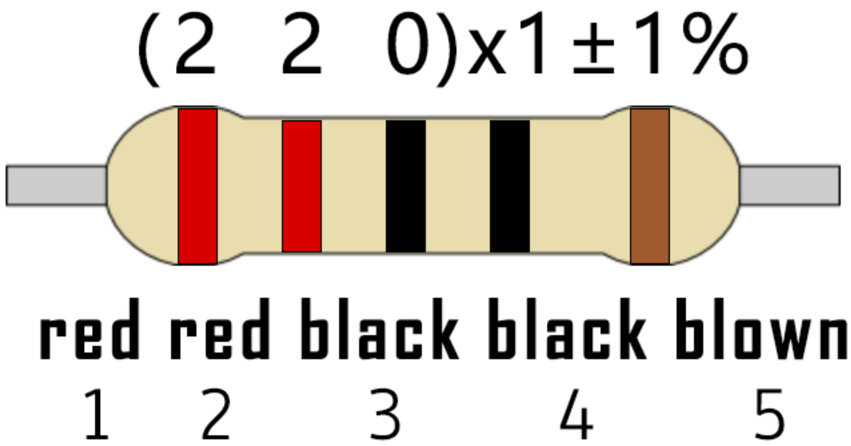


Note:

How to connect a LED



How to identify the 220 Five-band resistor

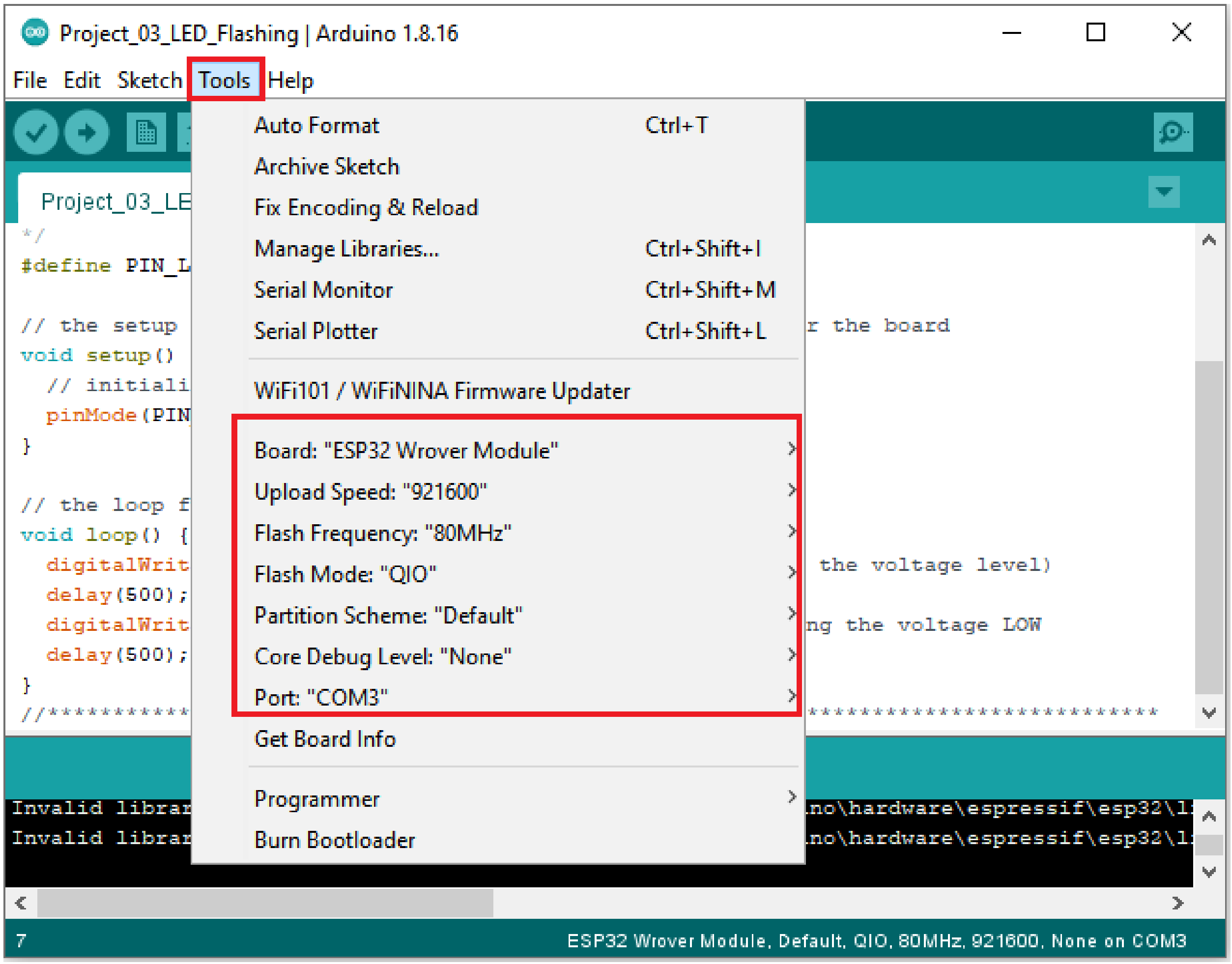


5.4.4 4.Test Code

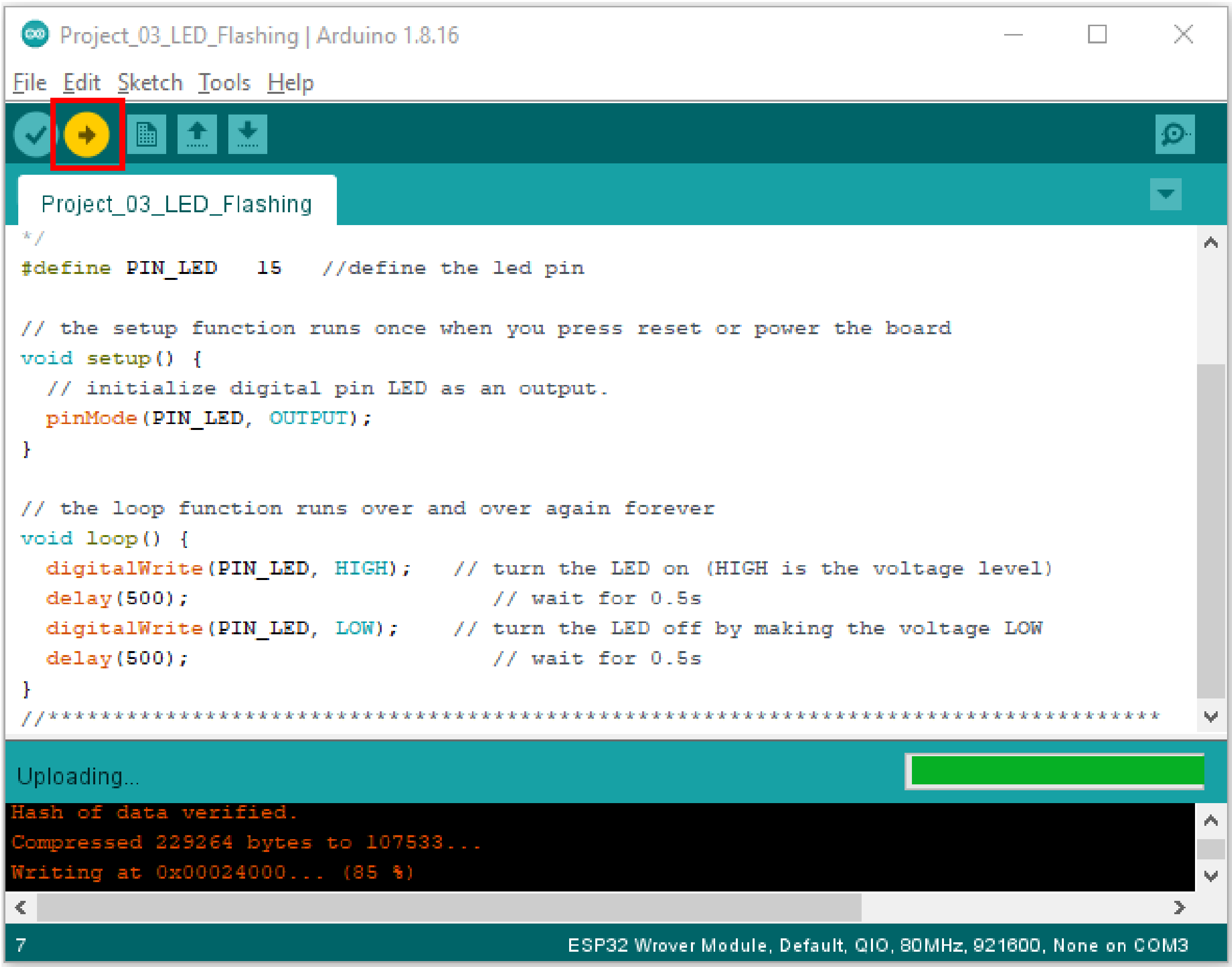
```
//**********************************************************************
/*
 * Filename    : External LED flashing
 * Description : Make an led blinking.
 * Auther      : http//www.keyestudio.com
*/
#define PIN_LED   15   //define the led pin

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

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

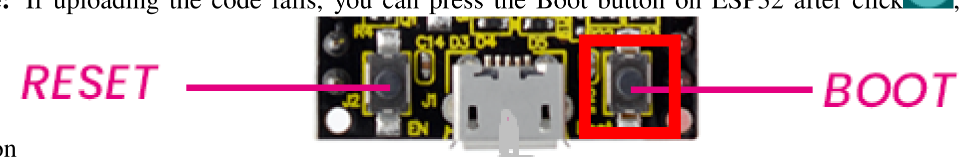
Click “**Tools**” to confirm the board type and port, as shown below:



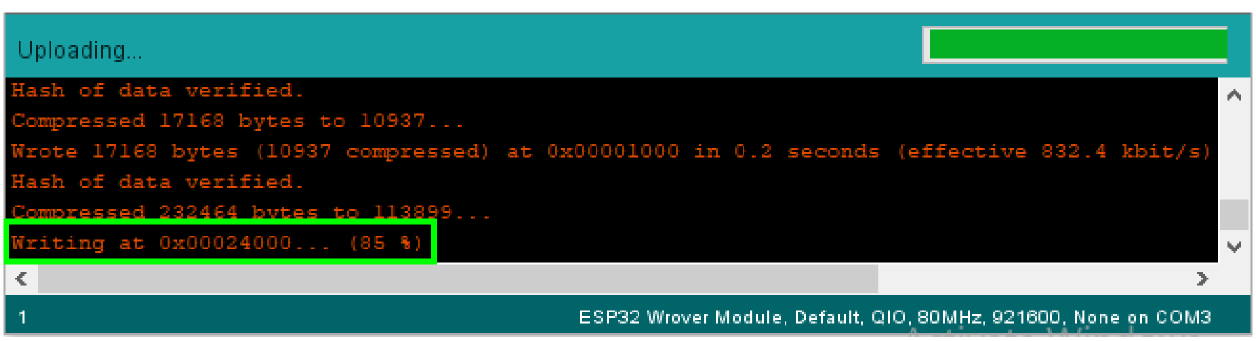
Click  to download the code to ESP32.



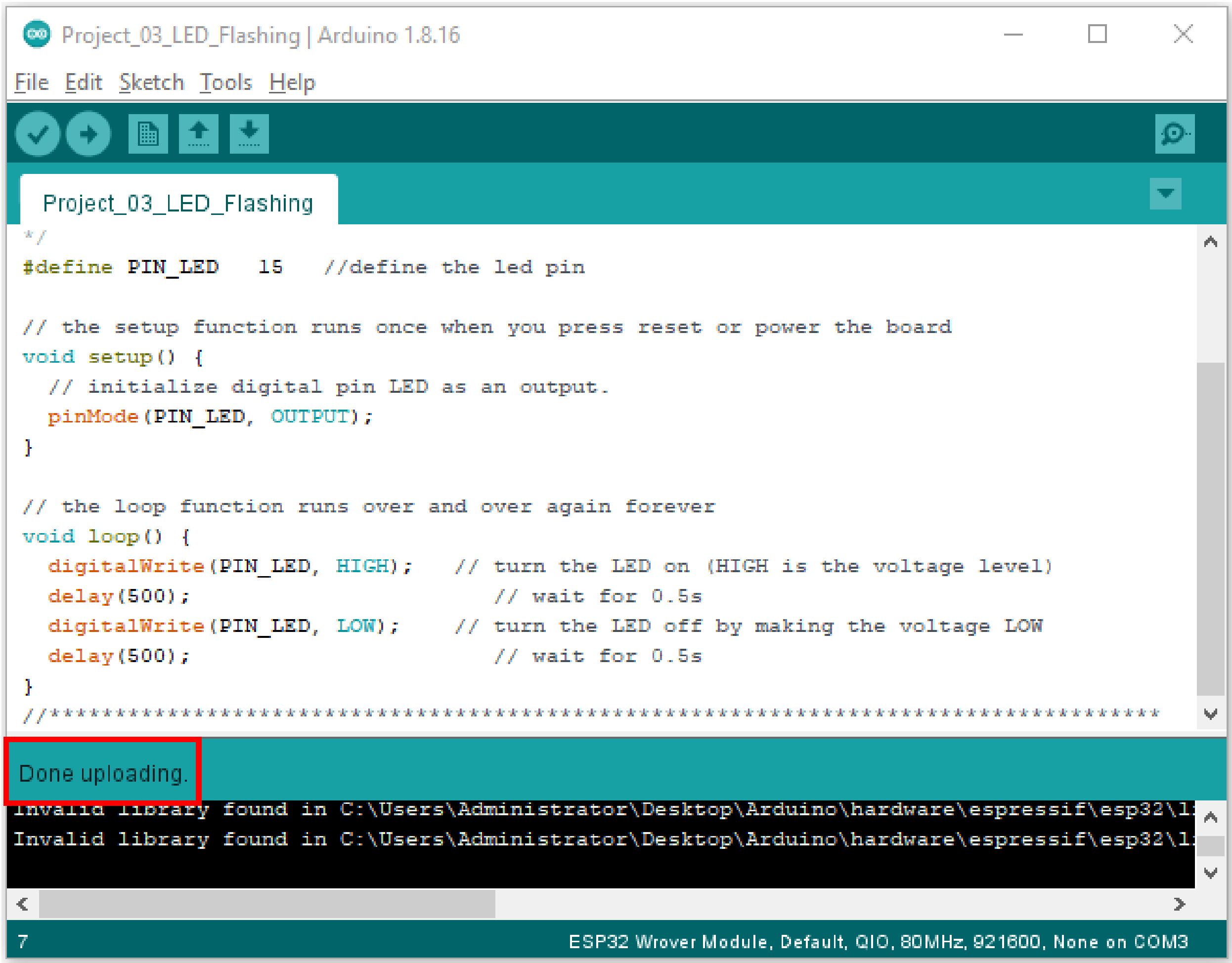
Note: If uploading the code fails, you can press the Boot button on ESP32 after click , and release the Boot button



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

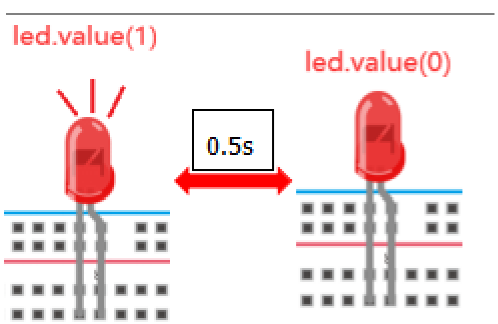


The code is uploaded successfully



5.4.5 5.Test Result

After uploading the code successfully, power up with a USB cable and the LED will start flashing.



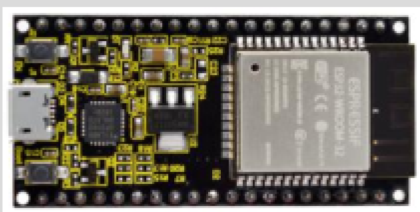
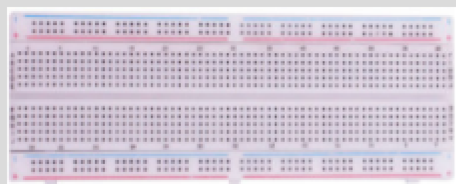
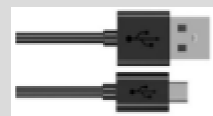
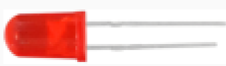
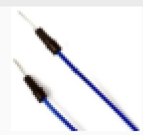
5.5 Project 04: Breathing Led

5.5.1 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". However, how to control the brightness of a LED? We will use ESP32's PWM to achieve this target.

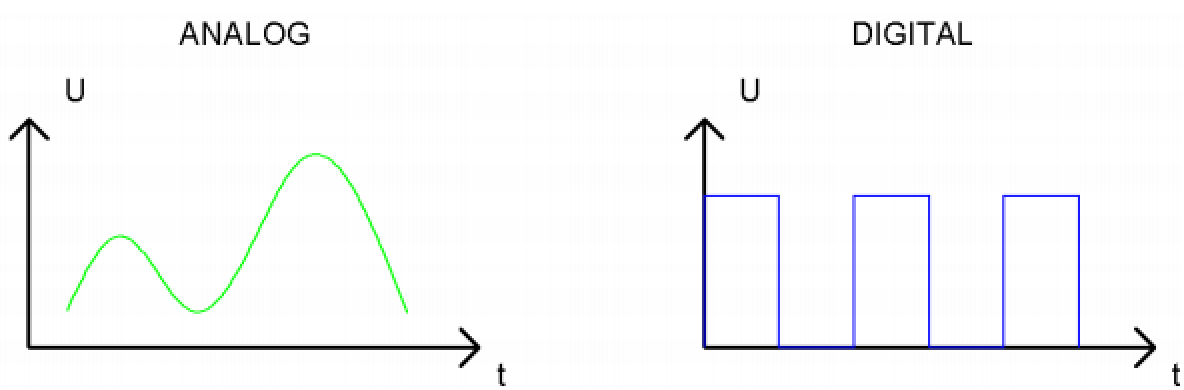
5.5.2 2.Components

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

5.5.3 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 continuously changes and could not 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 be seen more easily when compared, as shown below:



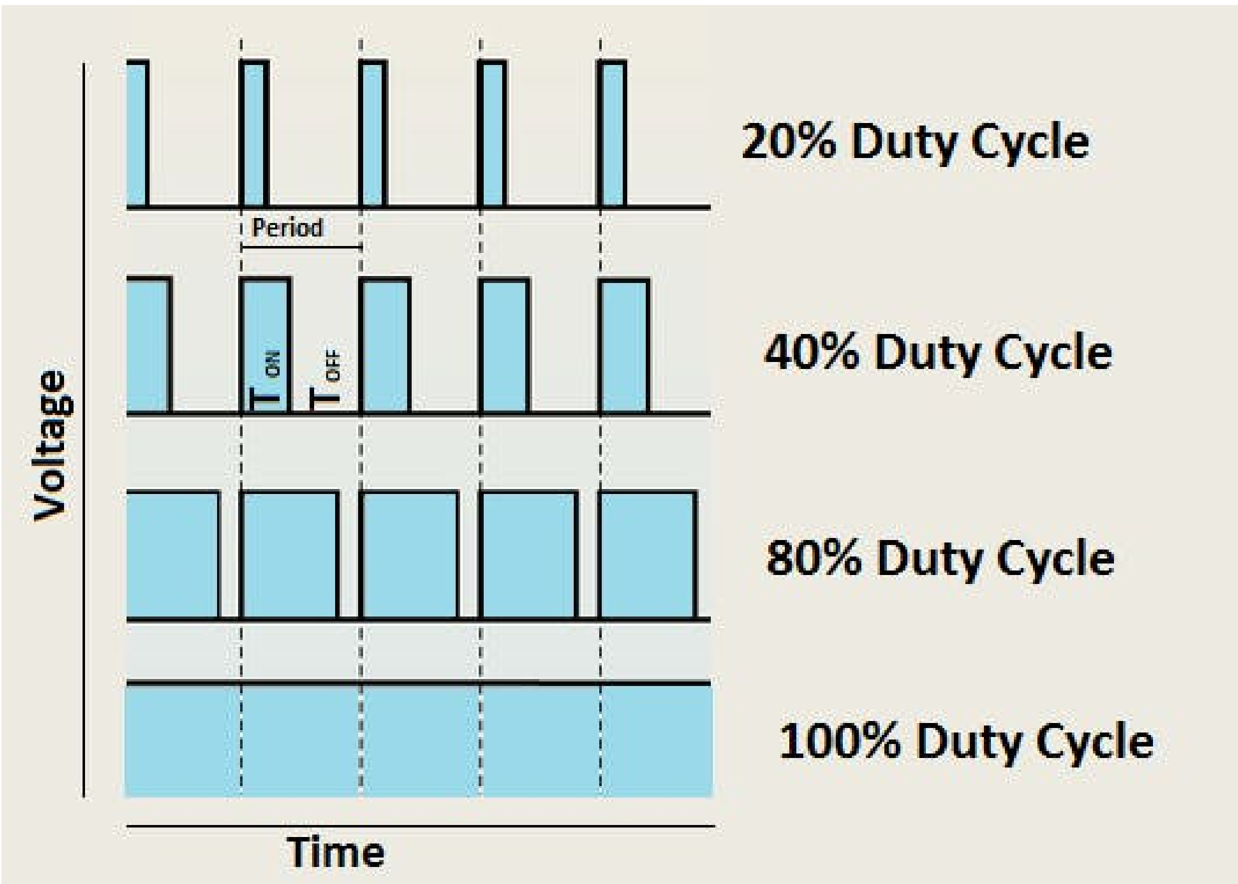
In practical application, we often use binary as the digital signal, that is a series of 0's and 1's. Since a binary signal only has two values (0 or 1), it has great stability and reliability. Lastly, both analog and digital signals can be converted into each other.

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 is 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-3.3V(high level is 3.3V) corresponding to the pulse width 0%-100%:



The longer the PWM duty cycle is, the higher the output power will be. Therefore, 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. Then we can control the output power of the LED and other output modules to achieve different effects.

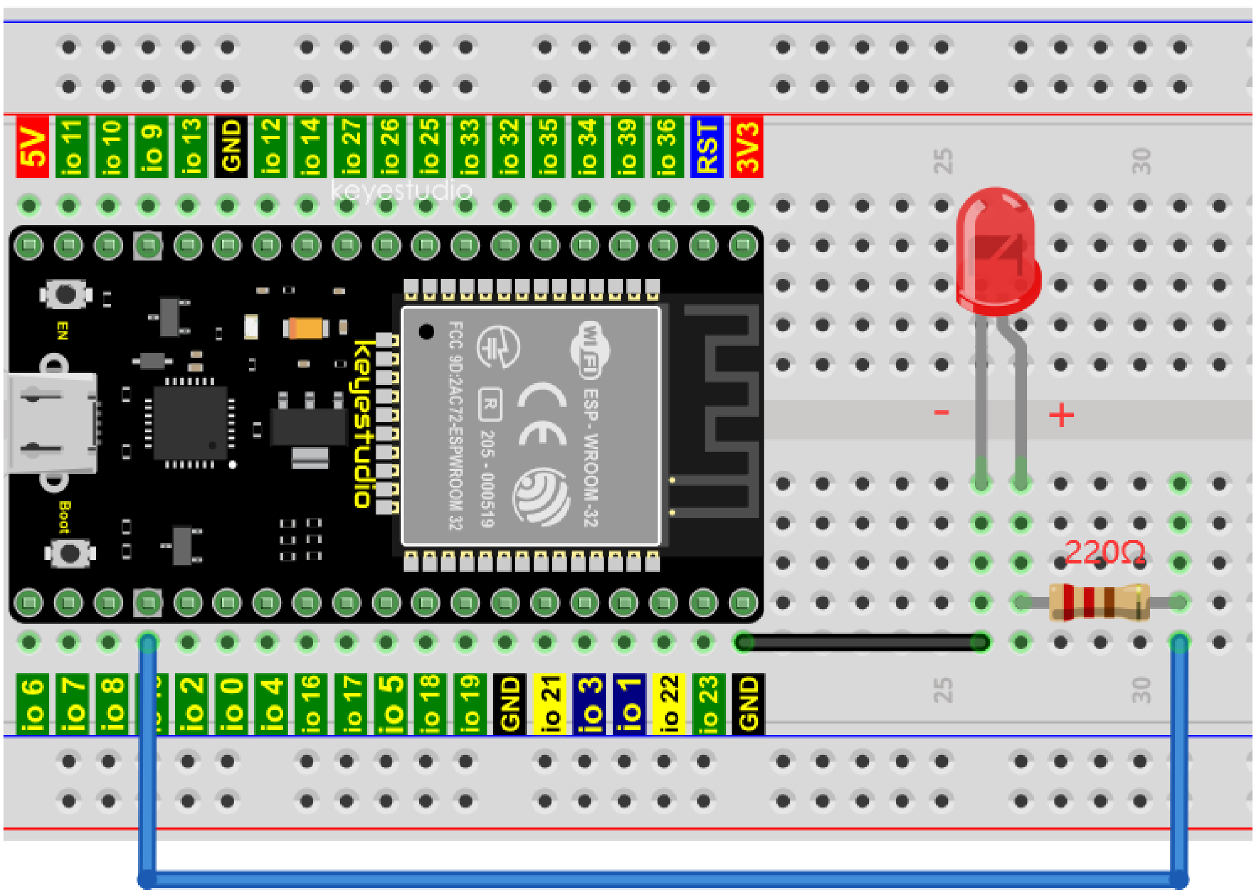
ESP32 and PWM:

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.

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

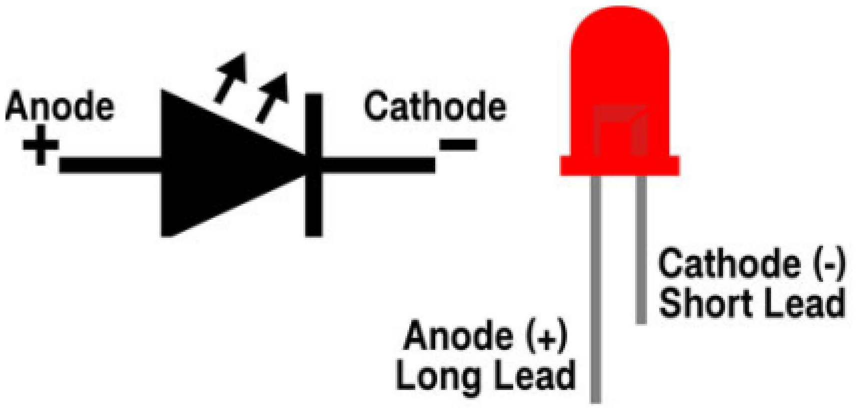
For example, generate a PWM with an 8-bit precision (28=256. Values range from 0 to 255) with a maximum frequency of 80,000,000/255 = 312,500Hz.

5.5.4 4.Wiring Diagram

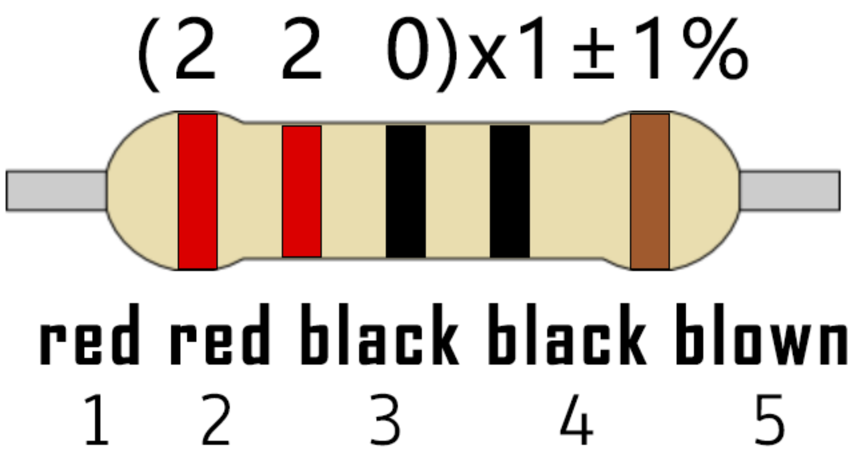


Note:

How to connect a LED



How to identify the 220 Five-band resistor



5.5.5 5.Test Code

The design of this project makes the GPIO15 output PWM, and the pulse width gradually increases from 0% to 100%, and then gradually decreases from 100% to 0%.

```
//**********************************************************************
/*
 * 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() {
```

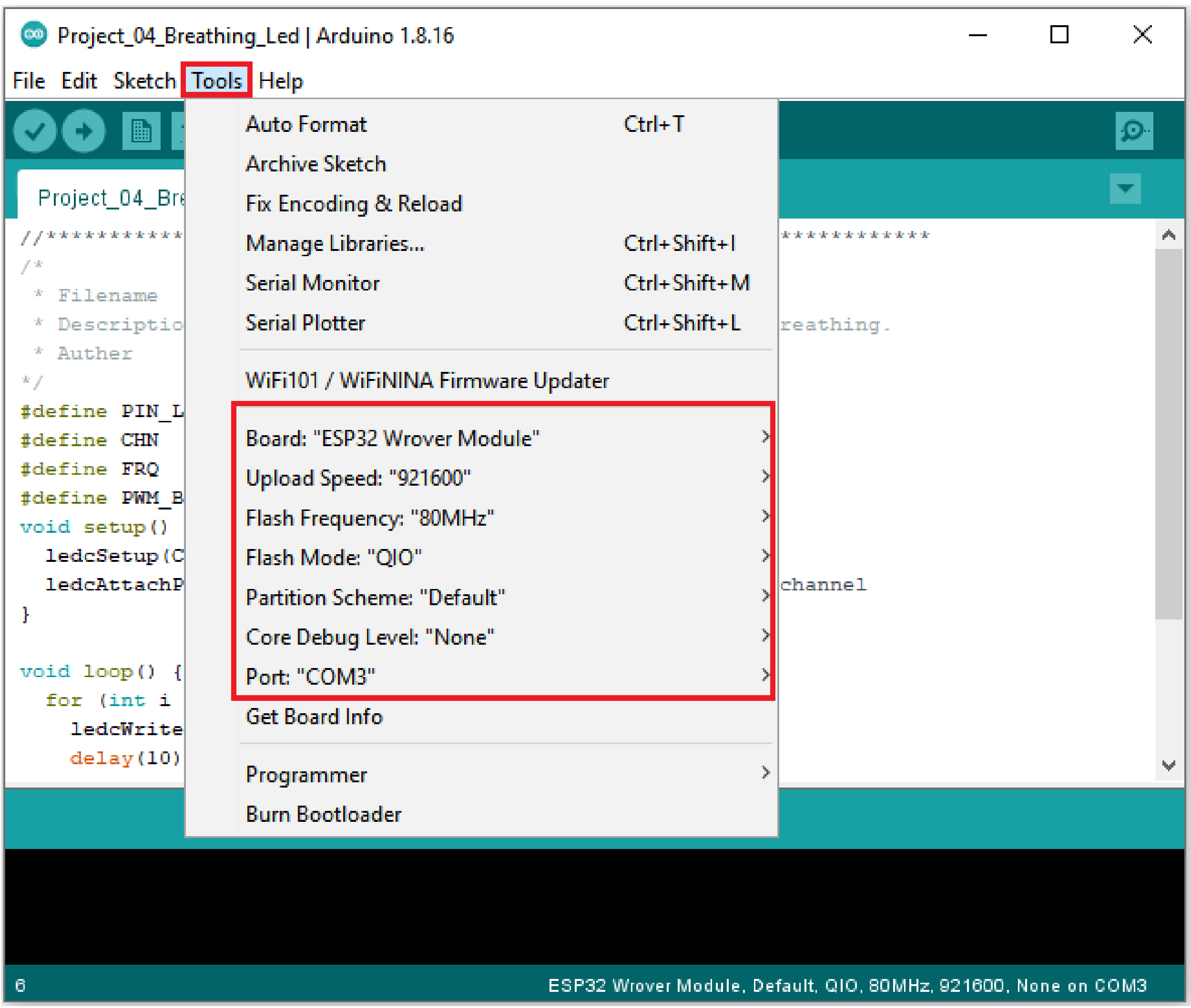
(continues on next page)

(continued from previous page)

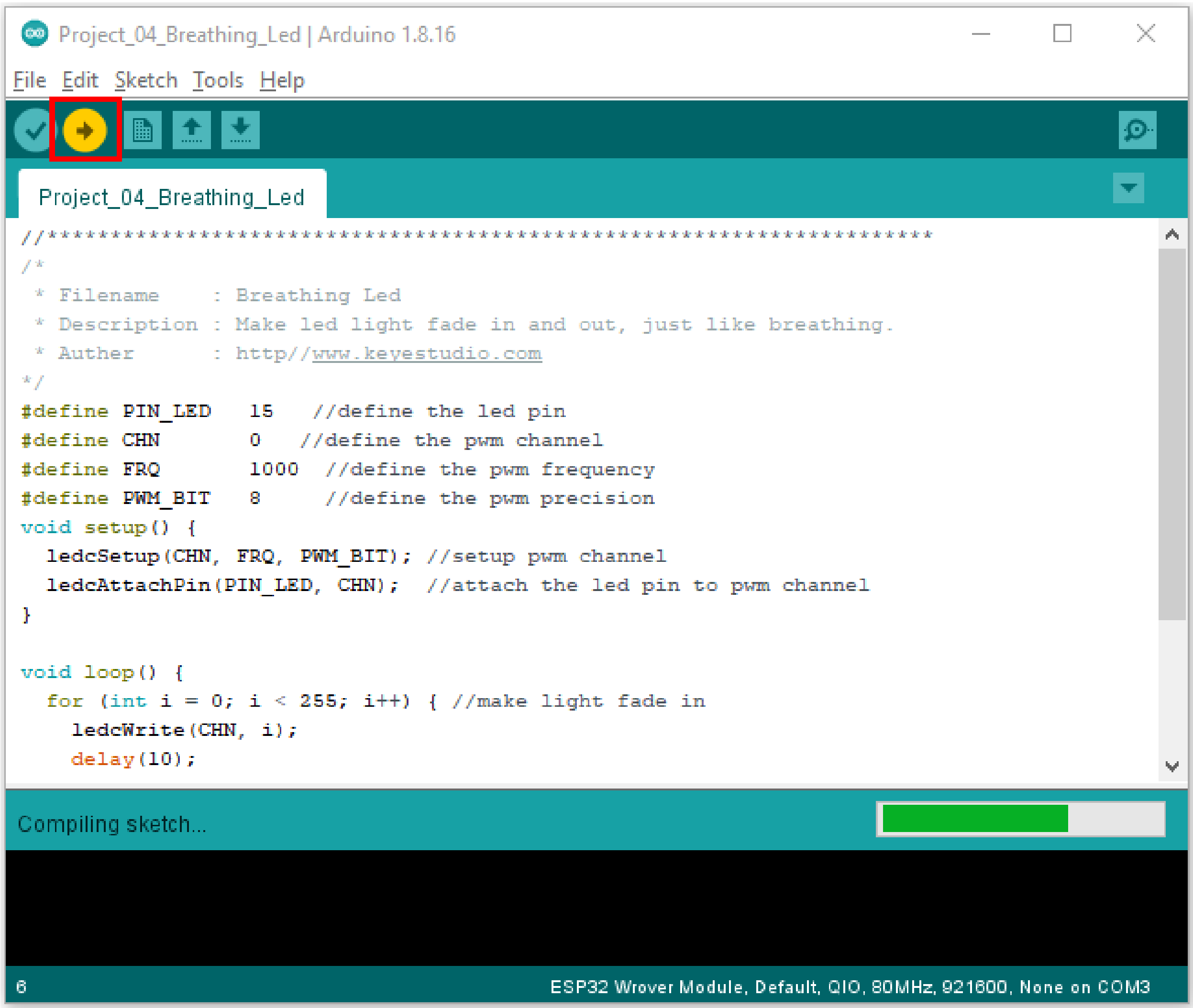
```
  for (int i = 0; i < 255; i++) { //make light fade in
    ledcWrite(CHN, i);
    delay(10);
  }
  for (int i = 255; i > -1; i--) {  //make light fade out
    ledcWrite(CHN, i);
    delay(10);
  }
}
//*************************************************************************************
```

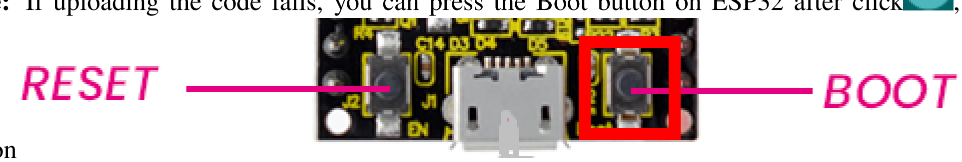
Before uploading Project Code to ESP32, please check the configuration of Arduino IDE.

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

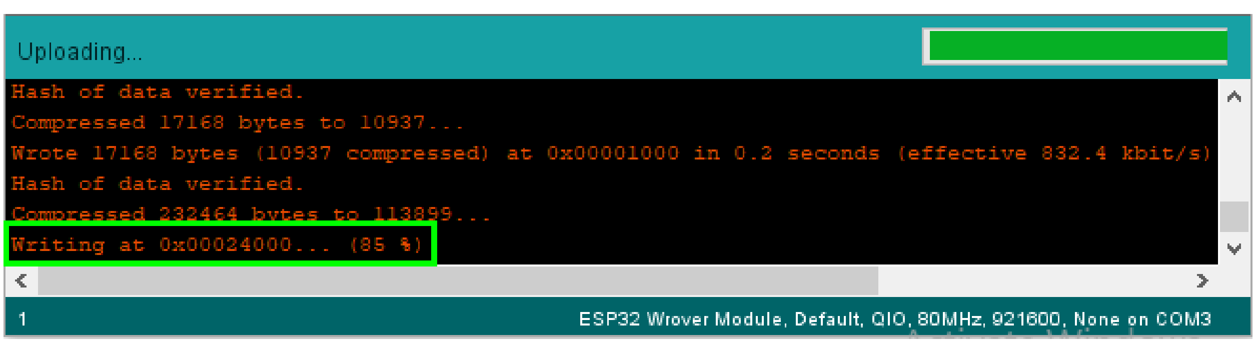


Click  to download the code to ESP32.

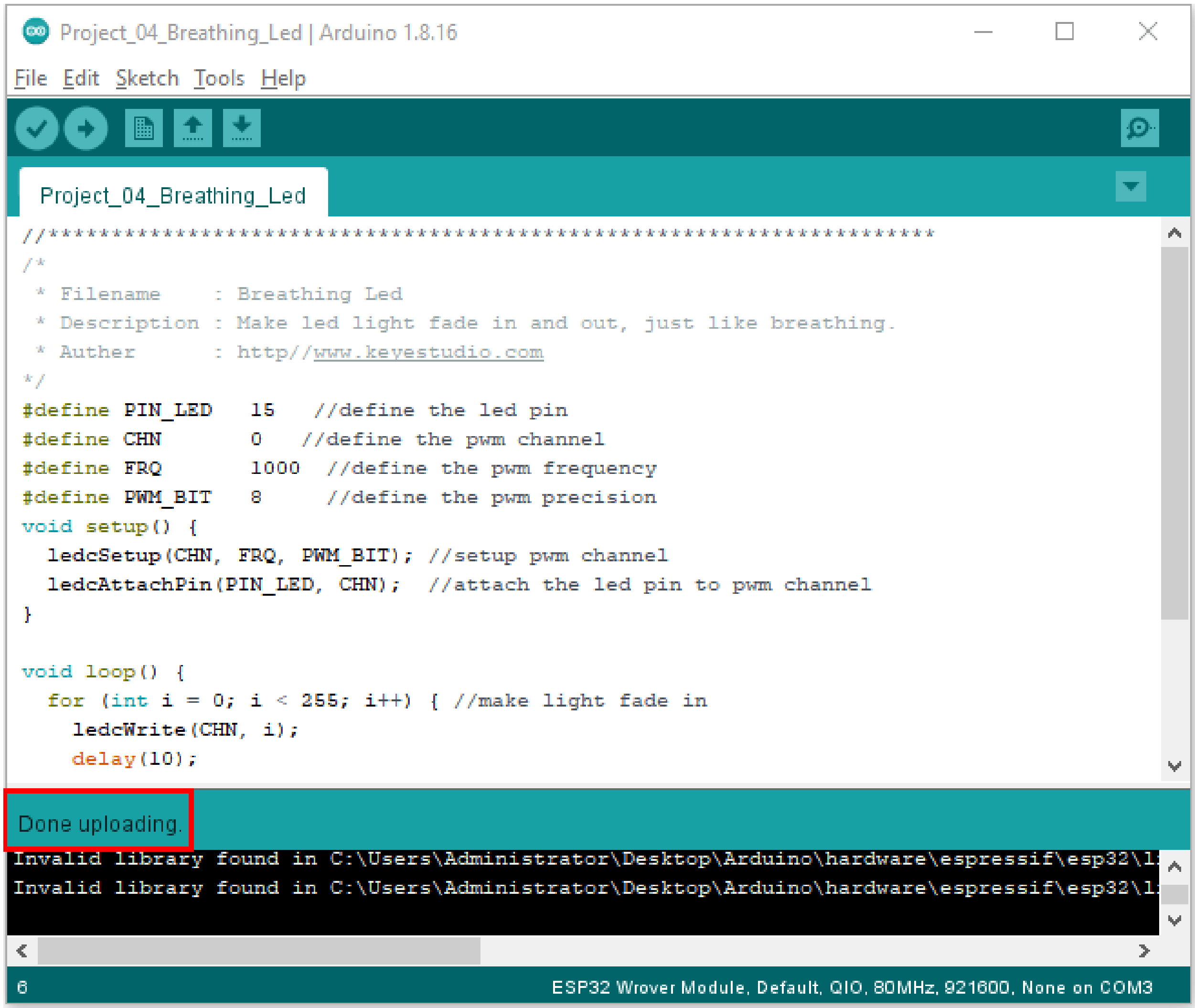


Note: If uploading the code fails, you can press the Boot button on ESP32 after click , and release the Boot button 

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



The code is uploaded successfully!



5.5.6 6.Test Result

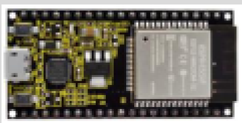
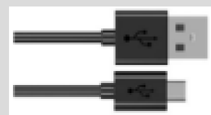
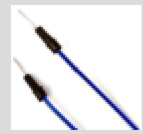
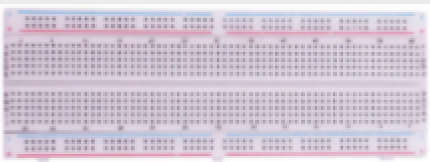
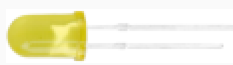
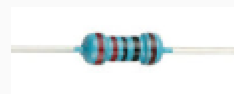
After uploading the code successfully, power up with a USB cable and the LED is turned from ON to OFF and then back from OFF to ON gradually like breathing.

5.6 Project 05Traffic Lights

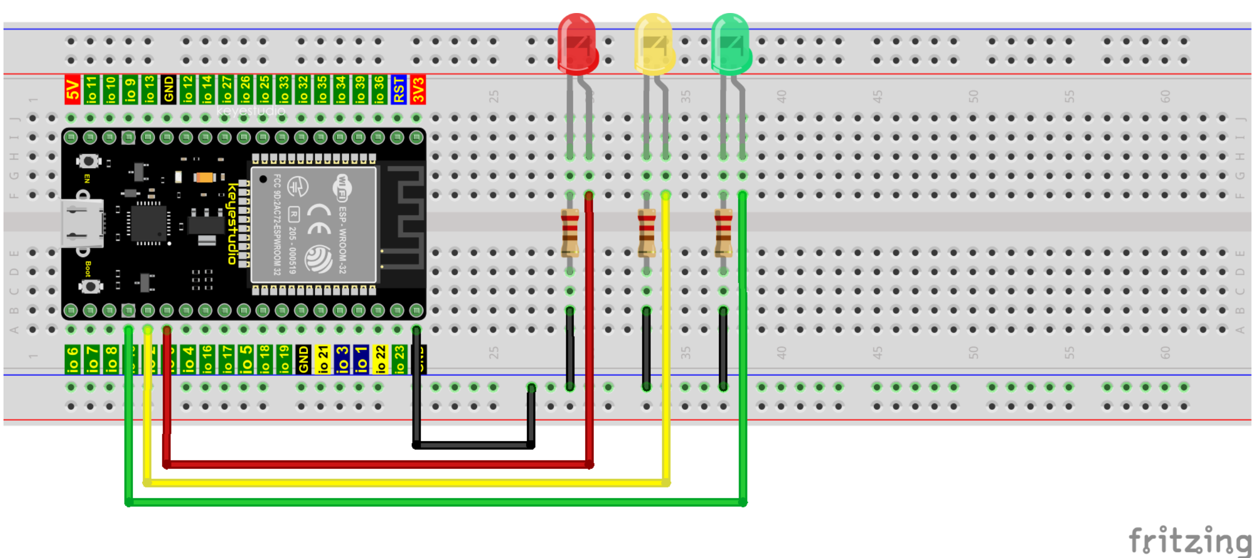
5.6.1 1.Introduction

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.

5.6.2 2.Components

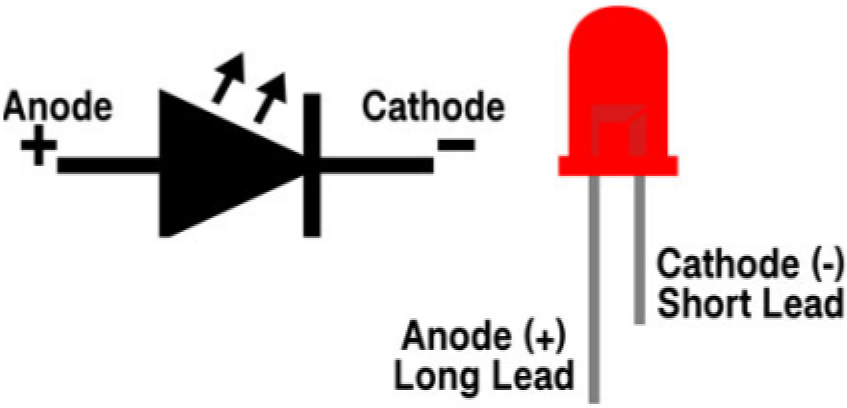
			
ESP32*1	Red LED*1	USB Cable*1	Jumper Wires
			
Bread board*1	Yellow LED*1	Green LED*1	220 Resistor*3

5.6.3 3.Wiring Diagram

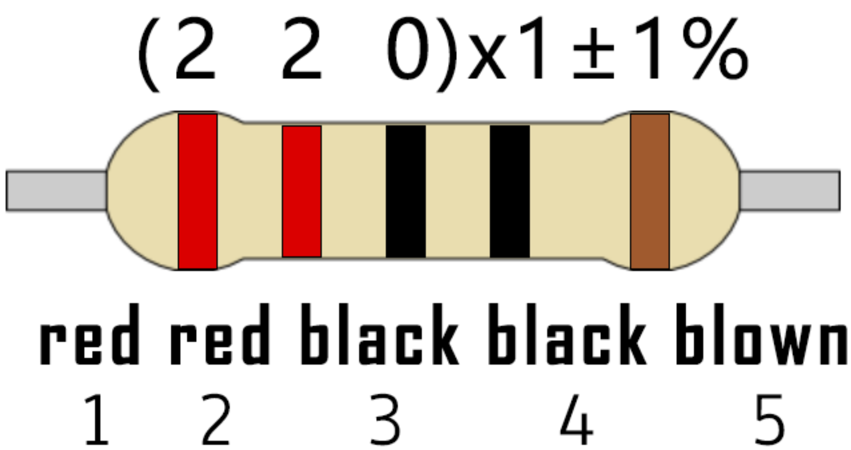


Note:

How to connect a LED



How to identify the 220 Five-band resistor



5.6.4 4.Test Code

```
//**********************************************************************
/*
 * 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_GREEN, HIGH);// turns on the green led
```

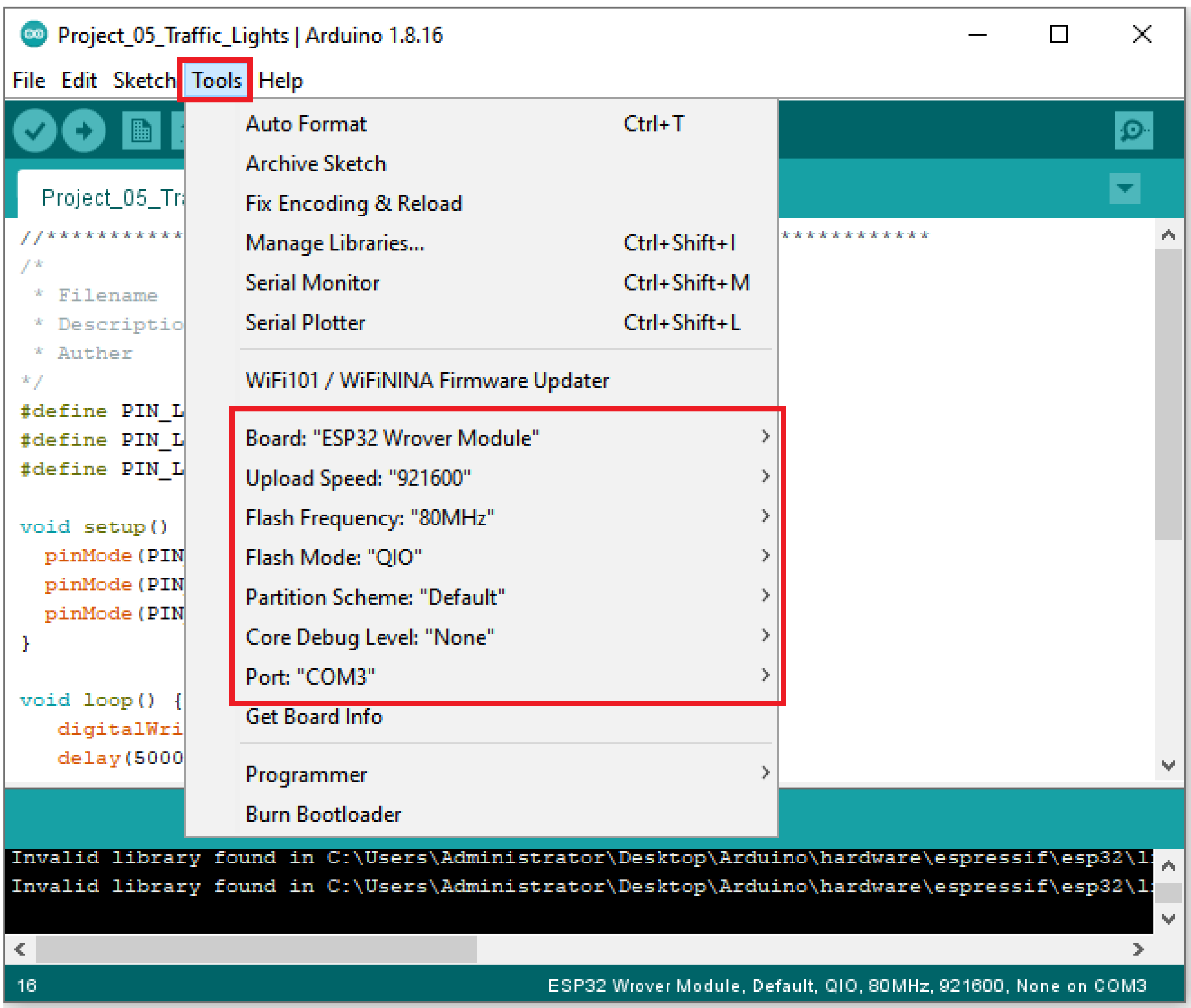
(continues on next page)

(continued from previous page)

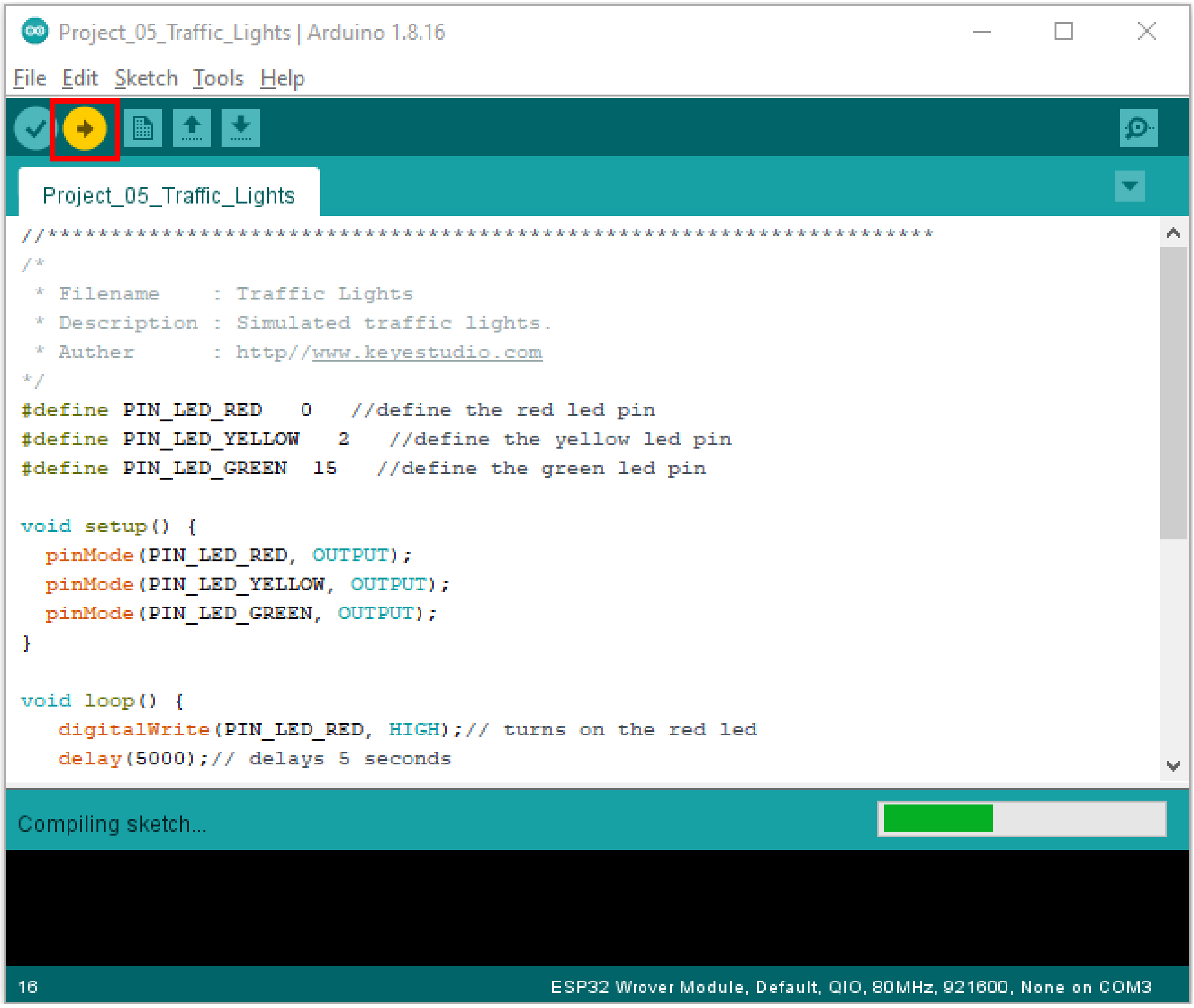
```
    delay(5000);// delays 5 seconds
    digitalWrite(PIN_LED_GREEN, LOW); // turns off the green led
    for(int i=0;i<3;i++)// flashes 3 times.
{
    delay(500);// delays 0.5 second
    digitalWrite(PIN_LED_YELLOW, HIGH);// turns on the yellow led
    delay(500);// delays 0.5 second
    digitalWrite(PIN_LED_YELLOW, LOW);// turns off the yellow led
}
    delay(500);// delays 0.5 second
    digitalWrite(PIN_LED_RED, HIGH);// turns on the red led
    delay(5000);// delays 5 second
    digitalWrite(PIN_LED_RED, LOW);// turns off the red led
}
//*************************************************************************************
```

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

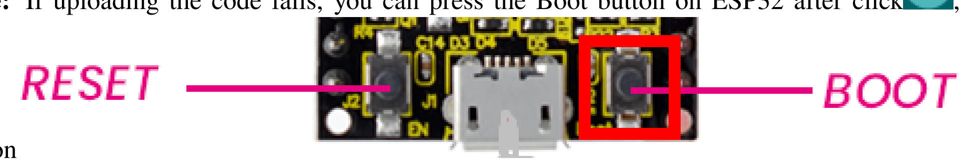
Click "**Tools**" to confirm the board type and port as shown below:



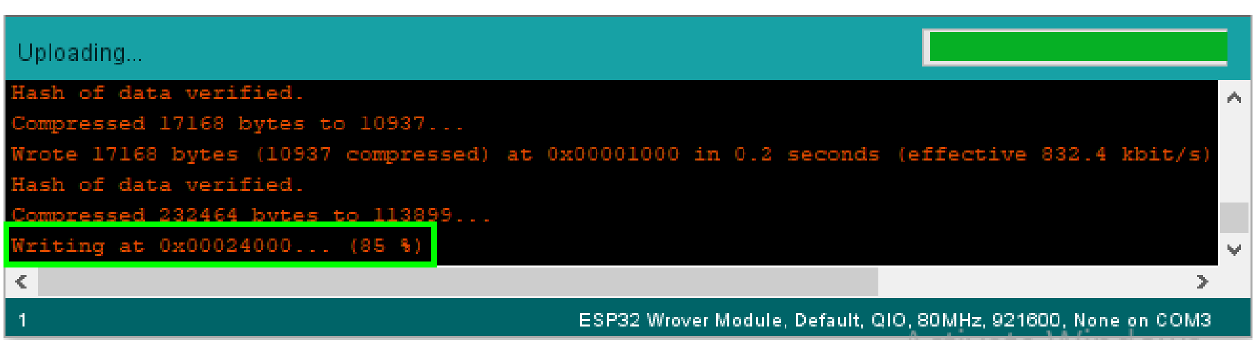
Click  to download the project code to ESP32.



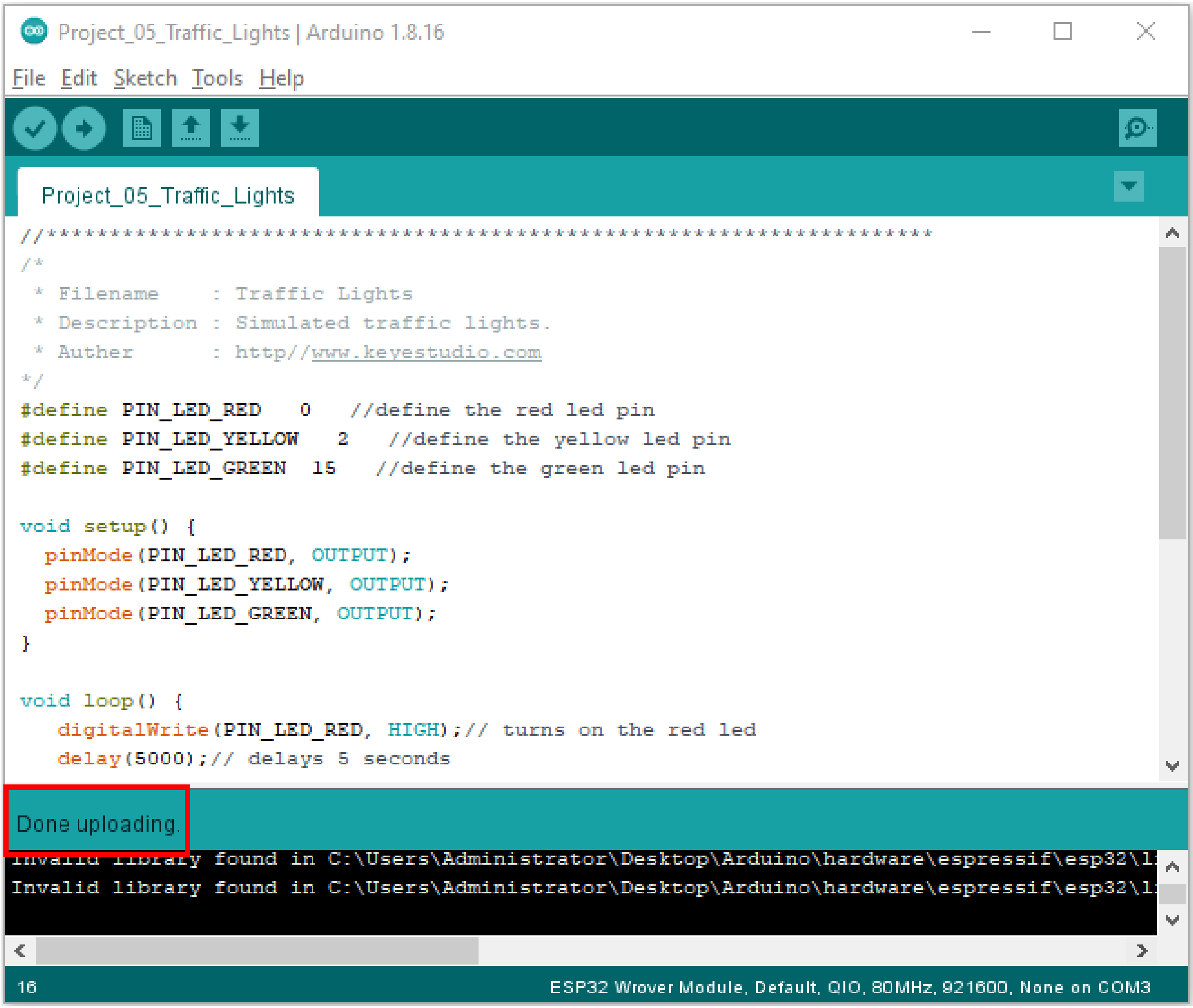
Note: If uploading the code fails, you can press the Boot button on ESP32 after click , and release the Boot button



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



The code is uploaded successfully



5.6.5 5.Test Result

After uploading the code successfully, power up with a USB cable and what you'll see are below:

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

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

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

Repeat step 1 to 3 above.

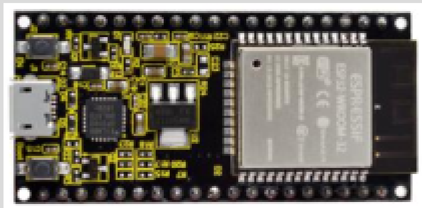
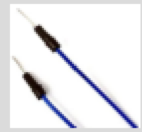
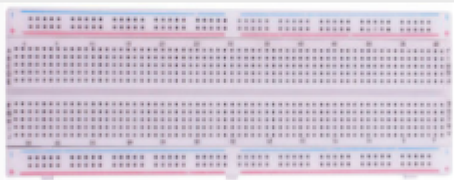
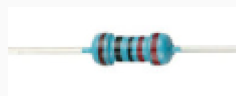
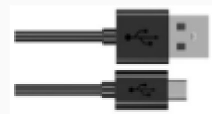
5.7 Project 06: RGB LED

5.7.1 1.Introduction

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

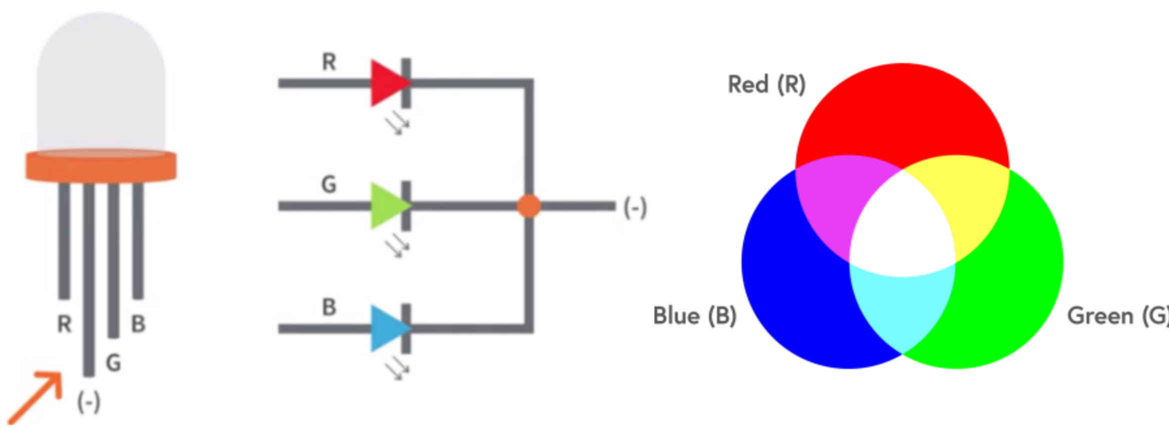
In this project, we will introduce the RGB and show you how to use ESP32 to control the RGB to emit different color lights .RGB is pretty basic, but it's also a great way to learn the fundamentals of electronics and coding.

5.7.2 2.Components

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

5.7.3 3.Component Knowledge

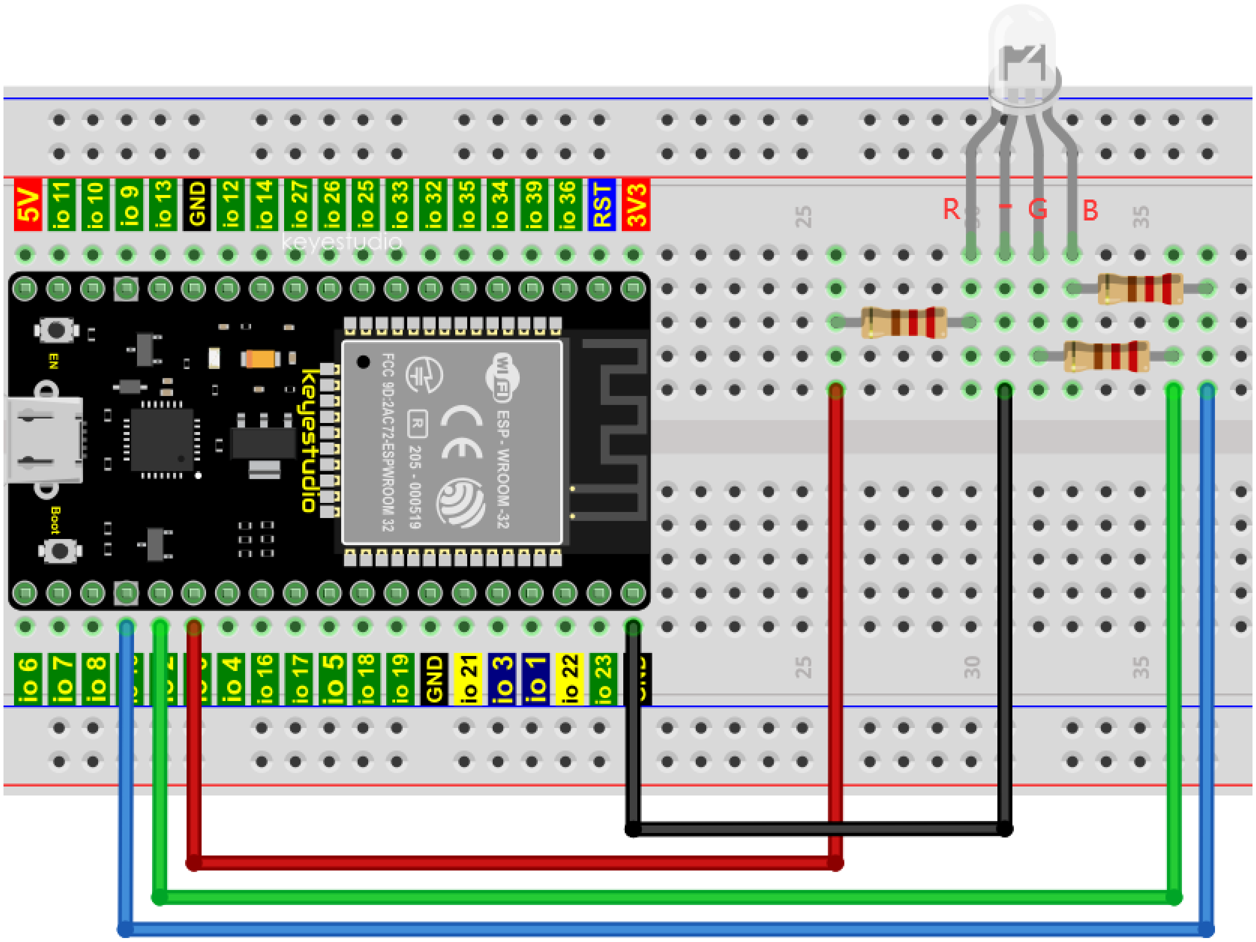
Most monitors adopt the RGB color standard, and all colors on a computer screen are a mixture of red, green and blue in varying proportions.



This RGB LED has 4 pins, with 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.

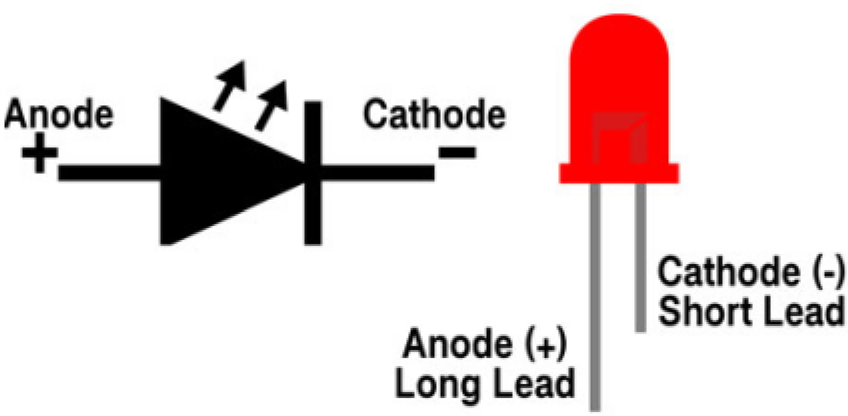
If we use three 10-bit PWM to control the RGB, in theory, we can create 2 10*210*210 =1,073,741,824(1billion) colors through different combinations.

5.7.4 4.Wiring Diagram

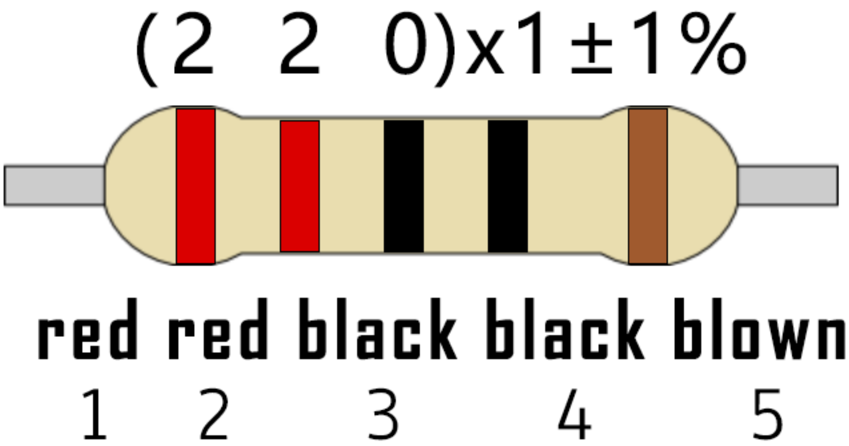


Note:

How to connect a LED



How to identify the 220 Five-band resistor



5.7.5 5.Test Code

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

5.7.6 6.Test Result

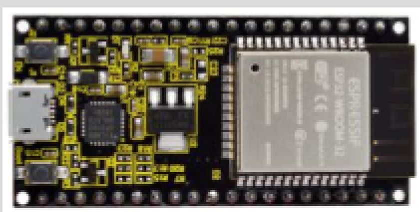
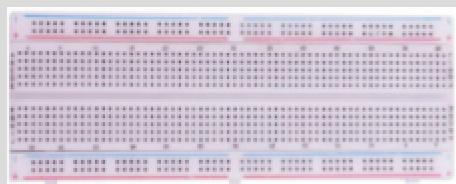
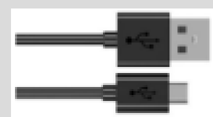
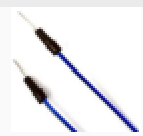
Compile and upload the code to ESP32, after the code is uploaded successfully, power up with a USB cable and you will see that the RGB LED starts to display random colors.

5.8 Project 07: Flowing Water Light

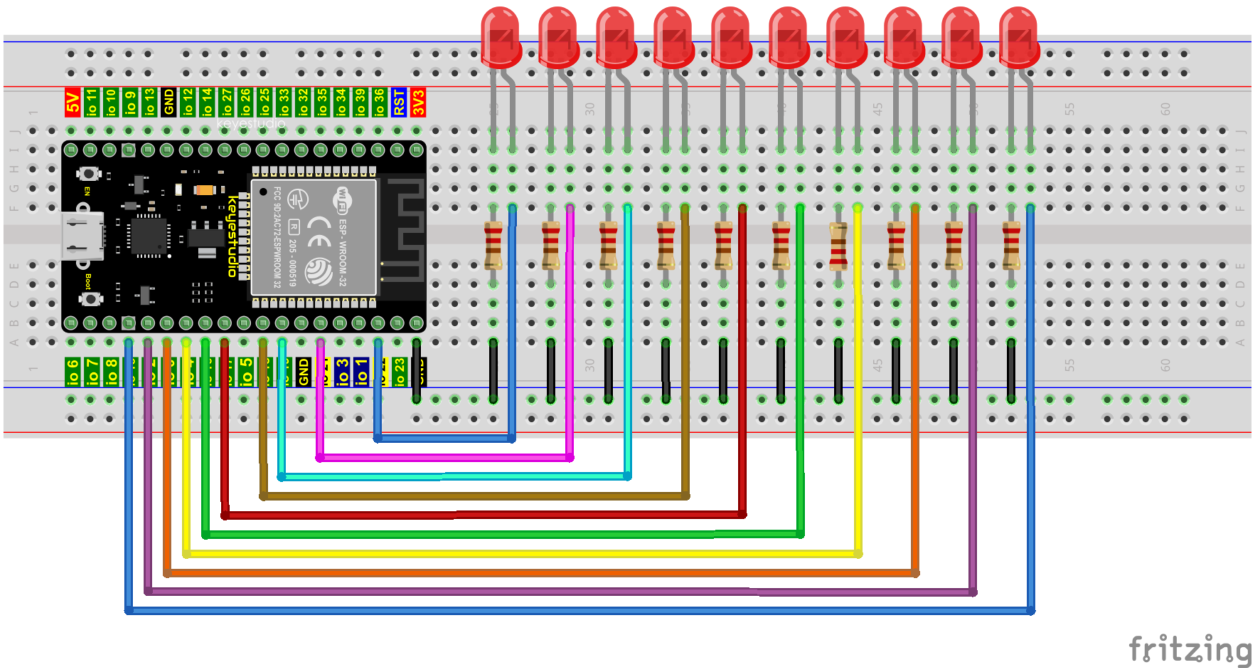
5.8.1 1.Introduction

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.

5.8.2 2.Components

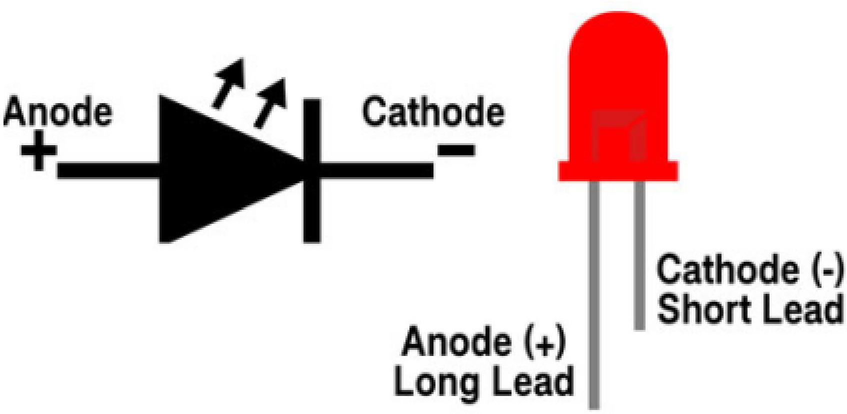
		
ESP32*1	Breadboard*1	USB Cable*1
		
Red LED*1	220 Resistor*1	Jumper Wire*2

5.8.3 3.Wiring Diagram

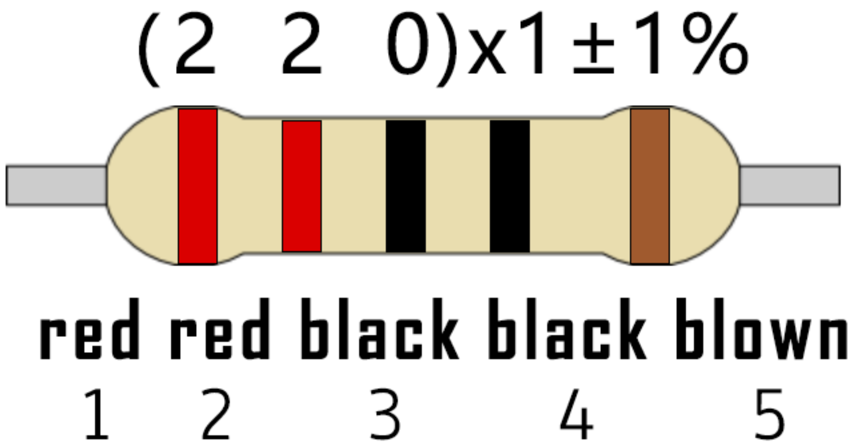


Note:

How to connect a LED



How to identify the 220 Five-band resistor



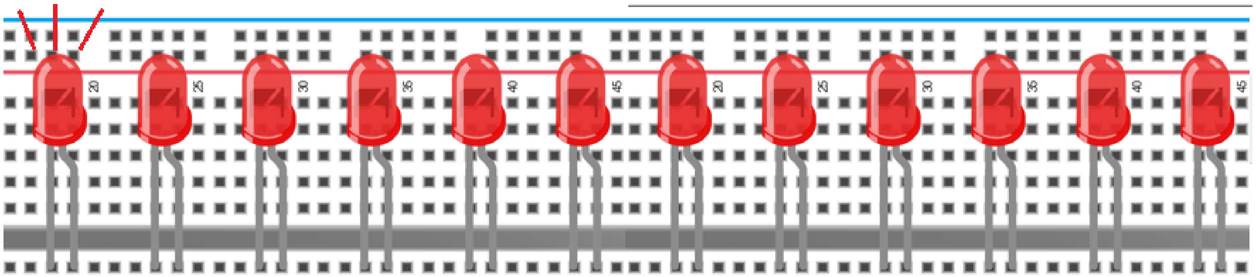
5.8.4 4.Test Code

This project is designed to make a flowing water lamp. 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.

```
//**********************************************************************
/*
 * Filename    : Flowing Water Light
 * Description : Using ten leds to demonstrate flowing lamp.
 * Auther      : http//www.keyestudio.com
*/
byte ledPins[] = {22, 21, 19, 18, 17, 16, 4, 0, 2, 15};
int ledCounts;

void setup() {
  ledCounts = sizeof(ledPins);
  for (int i = 0; i < ledCounts; i++) {
    pinMode(ledPins[i], OUTPUT);
  }
}

void loop() {
  for (int i = 0; i < ledCounts; i++) {
    digitalWrite(ledPins[i], HIGH);
    delay(100);
    digitalWrite(ledPins[i], LOW);
  }
  for (int i = ledCounts - 1; i > -1; i--) {
    digitalWrite(ledPins[i], HIGH);
    delay(100);
    digitalWrite(ledPins[i], LOW);
  }
}
//**********************************************************************
```

5.8.5 5.Test 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.



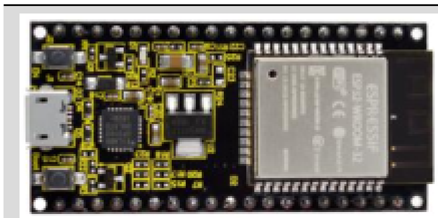
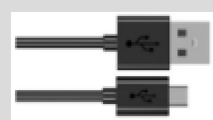
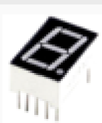
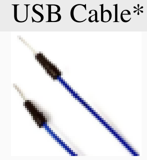
5.9 Project 081-Digit Digital Tube

5.9.1 1.Introduction

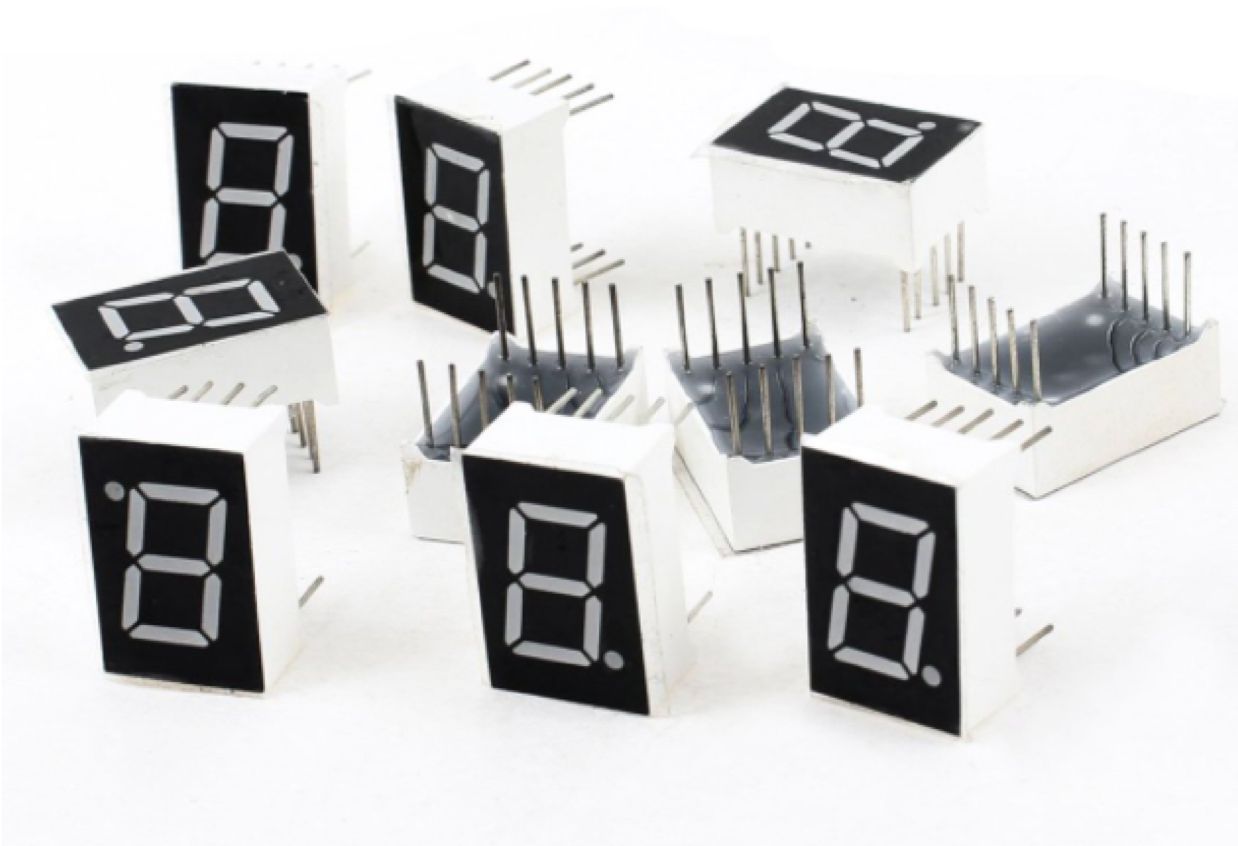
A 1-Digit 7-Segment Display is an electronic display device that displays decimal numbers. It is widely used in digital clocks, electronic meters, basic calculators and other electronic devices that display digital information.

Though 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.

5.9.2 2.Components

		
ESP32*1	Breadboard*1	USB Cable*1
		
1-Digit 7-Segment Display*1	220 Resistor*8	Jumper Wire*2

5.9.3 3.Component Knowledge

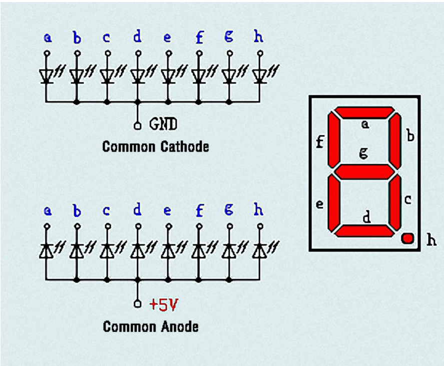


1-Digit 7-Segment Display principle:

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

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

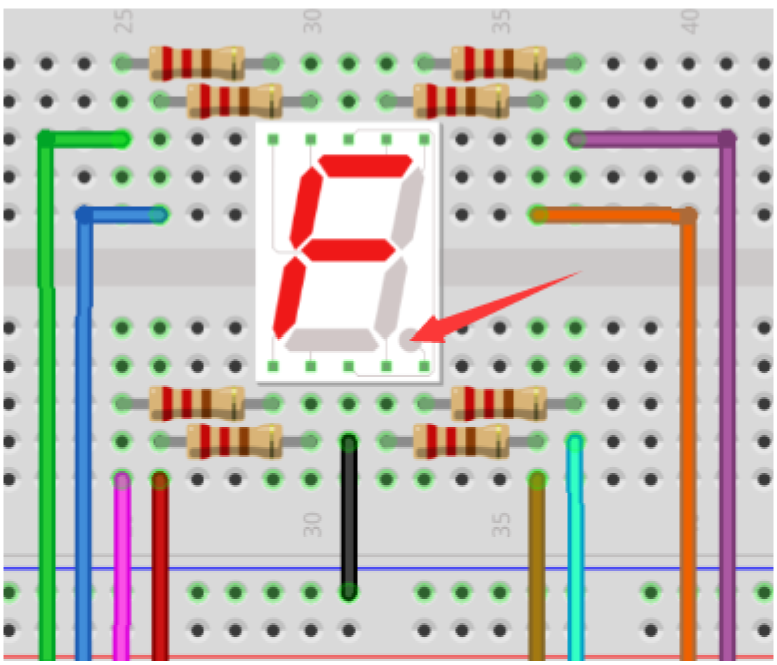
In the common anode 7-segment display, the LED anodes (positive electrodes) of all segments are connected together, so you should connect the common anode to"+5V". To light up a segmented LED, you can set its associated pin to"LOW".

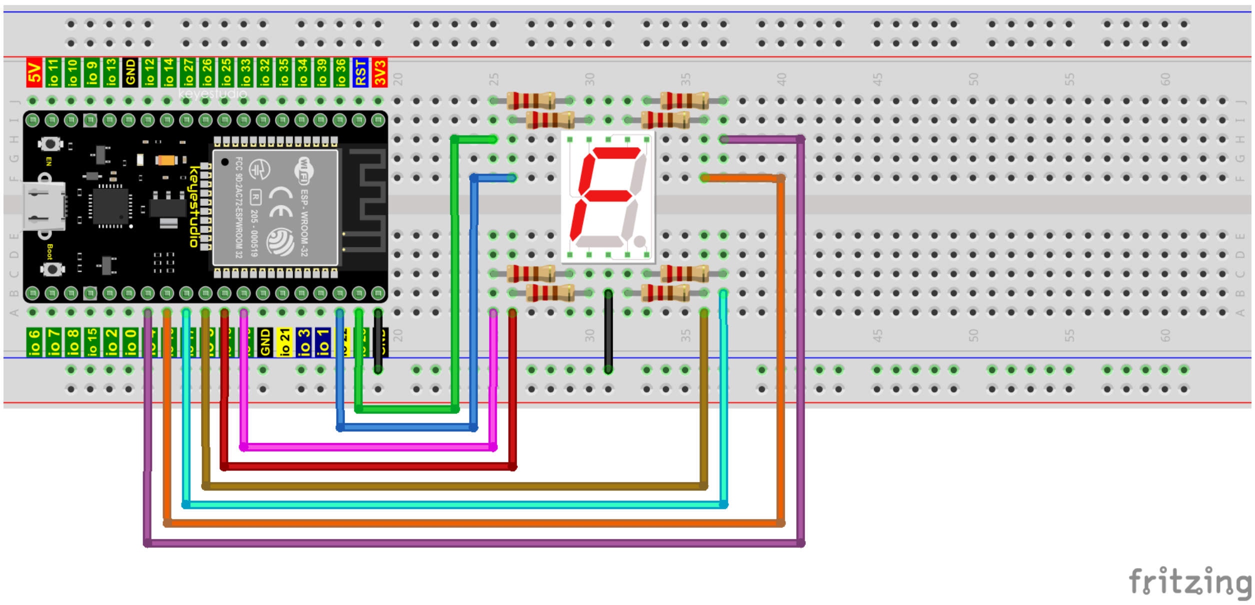


Each part of the digital tube is composed of an LED. So when you use it, you also need to use a current limiting resistor. Otherwise, the LED will be damaged. In this experiment, we use an ordinary common cathode one-digit digital tube. As we mentioned above, you should connect the common cathode to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

5.9.4 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.





5.9.5 5.Test 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.

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

(continues on next page)

(continued from previous page)

```
{
digitalWrite(a,LOW);
digitalWrite(b,HIGH);
digitalWrite(c,HIGH);
digitalWrite(d,LOW);
digitalWrite(e,LOW);
digitalWrite(f,LOW);
digitalWrite(g,LOW);
digitalWrite(dp,LOW);
}
void digital_2(void) // displays number 2
{
digitalWrite(a,HIGH);
digitalWrite(b,HIGH);
digitalWrite(c,LOW);
digitalWrite(d,HIGH);
digitalWrite(e,HIGH);
digitalWrite(f,LOW);
digitalWrite(g,HIGH);
digitalWrite(dp,LOW);
}
void digital_3(void) // displays number 3
{
digitalWrite(a,HIGH);
digitalWrite(b,HIGH);
digitalWrite(c,HIGH);
digitalWrite(d,HIGH);
digitalWrite(f,LOW);
digitalWrite(e,LOW);
digitalWrite(dp,LOW);
digitalWrite(g,HIGH);
}
void digital_4(void) // displays number 4
{
digitalWrite(a,LOW);
digitalWrite(b,HIGH);
digitalWrite(c,HIGH);
digitalWrite(d,LOW);
digitalWrite(e,LOW);
digitalWrite(f,HIGH);
digitalWrite(g,HIGH);
digitalWrite(dp,LOW);
}
void digital_5(void) // displays number 5
{
digitalWrite(a,HIGH);
digitalWrite(b,LOW);
digitalWrite(c,HIGH);
digitalWrite(d,HIGH);
digitalWrite(e,LOW);
digitalWrite(f,HIGH);
digitalWrite(g,HIGH);
```

(continues on next page)

(continued from previous page)

```
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);
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);
```

(continues on next page)

(continued from previous page)

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

5.9.6 6.Test Result

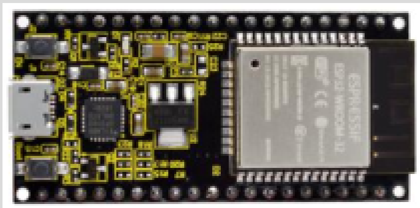
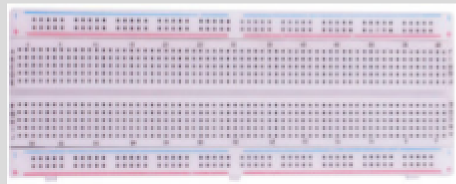
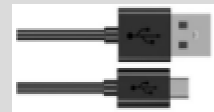
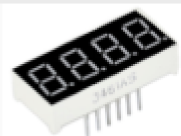
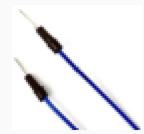
Compile and upload the code to ESP32, after the code is uploaded successfully, power up with a USB cable and you will see that the 1-Digit 7-Segment Display will display numbers from 9 to 0.

5.10 Project 094-Digit Digital Tube

5.10.1 1.Introduction

A 4-digit 7-segment display is a very practical display device and it is used for devices such as electronic 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 to control 4-digit 7-segment display to display digits.

5.10.2 2.Components

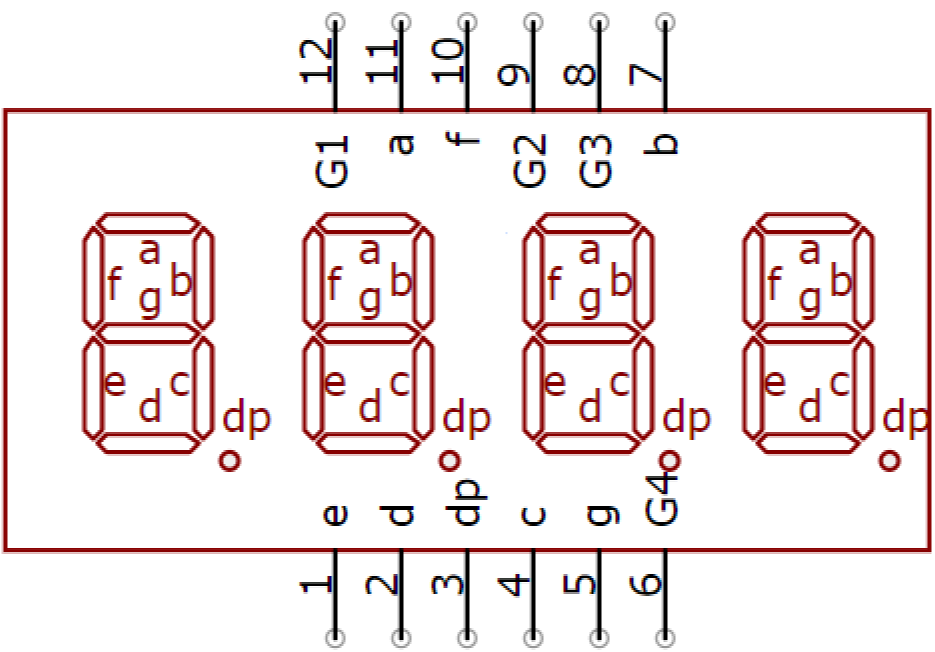
		
ESP32*1	Breadboard*1	USB Cable*1
		
4-digit 7-segment display Module*1	220 Resistor*8	Jumper Wire*2

5.10.3 3.Component Knowledge

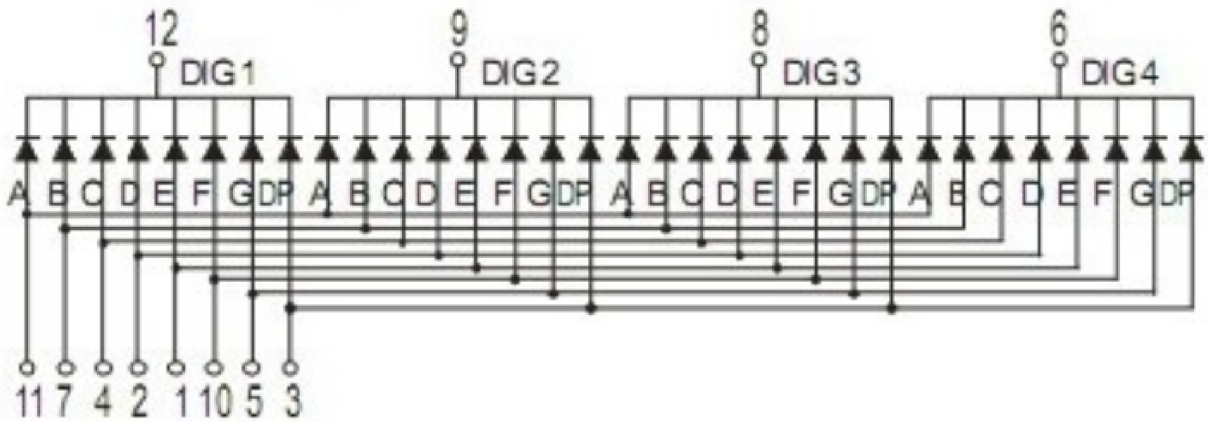
4-digit 7-segment display

It is a device with common cathode and anode, its display principle is similar to the 1-Digit digital tube display. They all have eight GPIO ports to control the digital tube display, that is 8 leds. However, here is 4-digit, so it needs four GPIO ports to control the bit selection end. Our 4 - digit digital tube is common cathode.

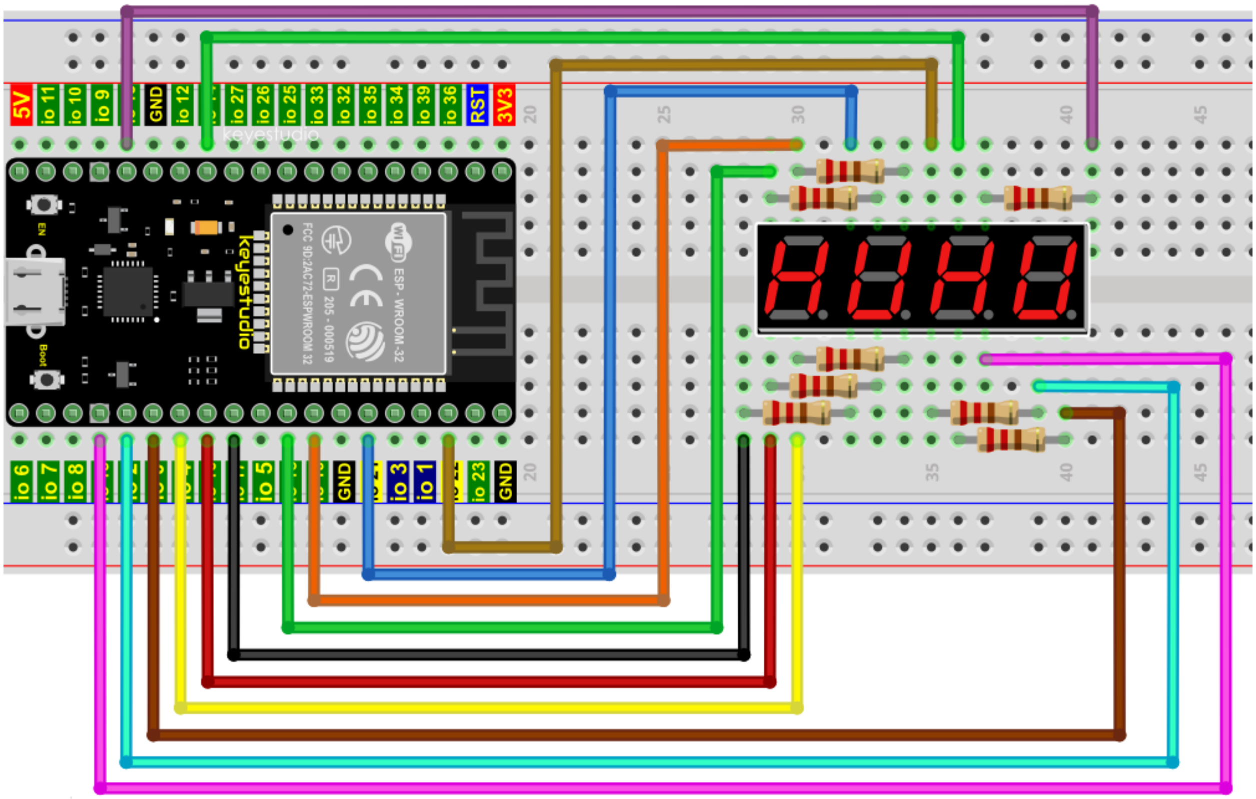
The following figure shows the pin diagram of the 4-digit digital tube. G1, G2, G3 and G4 are the control pins.



Schematic Diagram



5.10.4 4.Wiring Diagram



5.10.5 5.Test Code

```
//**********************************************************************
/*
 * Filename    : 4-Digit Digital Tube
 * Description : Four Digit Tube displays numbers from 0 to 9999.
 * Auther      : http//www.keyestudio.com
*/
#define d_a 18   //Define nixie tube a to pin 18
#define d_b 13
#define d_c 2
#define d_d 16
#define d_e 17
#define d_f 19
#define d_g 0
#define d_dp 4

#define G1 21   //Define the first set of nixtube G1 to pin 21
#define G2 22
#define G3 14
#define G4 15

//Nixie tube 0-F code value
```

(continues on next page)

(continued from previous page)

```
unsigned char num[17][8] =
{
 //a  b  c  d  e  f  g  dp
  {1, 1, 1, 1, 1, 1, 0, 0},     //0
  {0, 1, 1, 0, 0, 0, 0, 0},     //1
  {1, 1, 0, 1, 1, 0, 1, 0},     //2
  {1, 1, 1, 1, 0, 0, 1, 0},     //3
  {0, 1, 1, 0, 0, 1, 1, 0},     //4
  {1, 0, 1, 1, 0, 1, 1, 0},     //5
  {1, 0, 1, 1, 1, 1, 1, 0},     //6
  {1, 1, 1, 0, 0, 0, 0, 0},     //7
  {1, 1, 1, 1, 1, 1, 1, 0},     //8
  {1, 1, 1, 1, 0, 1, 1, 0},     //9
  {1, 1, 1, 0, 1, 1, 1, 1},     //A
  {1, 1, 1, 1, 1, 1, 1, 1},     //B
  {1, 0, 0, 1, 1, 1, 0, 1},     //C
  {1, 1, 1, 1, 1, 1, 0, 1},     //D
  {1, 0, 0, 1, 1, 1, 1, 1},     //E
  {1, 0, 0, 0, 1, 1, 1, 1},     //F
  {0, 0, 0, 0, 0, 0, 0, 1},     //.
};

void setup()
{
  pinMode(d_a,OUTPUT);    //Set to output pin
  pinMode(d_b,OUTPUT);
  pinMode(d_c,OUTPUT);
  pinMode(d_d,OUTPUT);
  pinMode(d_e,OUTPUT);
  pinMode(d_f,OUTPUT);
  pinMode(d_g,OUTPUT);
  pinMode(d_dp,OUTPUT);

  pinMode(G1,OUTPUT);
  pinMode(G2,OUTPUT);
  pinMode(G3,OUTPUT);
  pinMode(G4,OUTPUT);
}

void loop()
{

  //Start counting from 0 and gradually increase by 1 to 9999, repeating.
  for(int l = 0;l < 10;l++ )
  {
    for(int k = 0; k < 10;k++)
    {
      for(int j = 0; j < 10; j++)
      {
        for(int i = 0;i < 10;i++)
        {
          //125 flashes a second equals one second.
```

(continues on next page)

(continued from previous page)

```
          //1000/8=125
          for(int q = 0;q<125;q++)
          {
            Display(1,l);//The first nixie tube shows the value of L.
            delay(2);
            Display(2,k);
            delay(2);
            Display(3,j);
            delay(2);
            Display(4,i);
            delay(2);
          }

        }
      }
    }
  }
}

//Display functions: g ranges from 1 to 4,num ranges from 0 to 9
void Display(unsigned char g,unsigned char n)
{
  digitalWrite(d_a,LOW);      //Remove the light
  digitalWrite(d_b,LOW);
  digitalWrite(d_c,LOW);
  digitalWrite(d_d,LOW);
  digitalWrite(d_e,LOW);
  digitalWrite(d_f,LOW);
  digitalWrite(d_g,LOW);
  digitalWrite(d_dp,LOW);

  switch(g)           //Gate a choice
  {
    case 1:
      digitalWrite(G1,LOW);   //Choose the first digit
      digitalWrite(G2,HIGH);
      digitalWrite(G3,HIGH);
      digitalWrite(G4,HIGH);
      break;
    case 2:
      digitalWrite(G1,HIGH);
      digitalWrite(G2,LOW);   //Choose the second bit
      digitalWrite(G3,HIGH);
      digitalWrite(G4,HIGH);
      break;
    case 3:
      digitalWrite(G1,HIGH);
      digitalWrite(G2,HIGH);
      digitalWrite(G3,LOW);   //Choose the third bit
      digitalWrite(G4,HIGH);
      break;
    case 4:
```

(continues on next page)

(continued from previous page)

```
      digitalWrite(G1,HIGH);
      digitalWrite(G2,HIGH);
      digitalWrite(G3,HIGH);
      digitalWrite(G4,LOW);   //Choose the fourth bit
      break;
    default:break;
  }

  digitalWrite(d_a,num[n][0]);      //a Queries the code value table
  digitalWrite(d_b,num[n][1]);
  digitalWrite(d_c,num[n][2]);
  digitalWrite(d_d,num[n][3]);
  digitalWrite(d_e,num[n][4]);
  digitalWrite(d_f,num[n][5]);
  digitalWrite(d_g,num[n][6]);
  digitalWrite(d_dp,num[n][7]);
}
//*********************************************************************
```

5.10.6 6.Test 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 4-digit 7-segment display displays 0-9999and repeat these actions in an infinite loop.

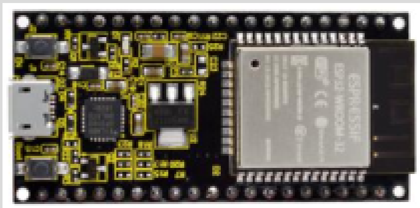
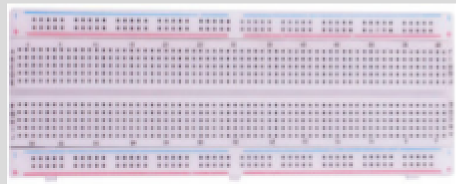
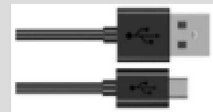
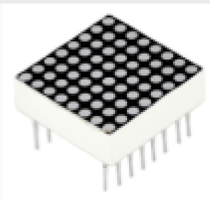
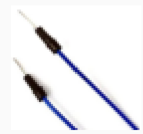
5.11 Project 108×8 Dot-matrix Display

5.11.1 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 to control 8x8 LED dot matrix to display digits.

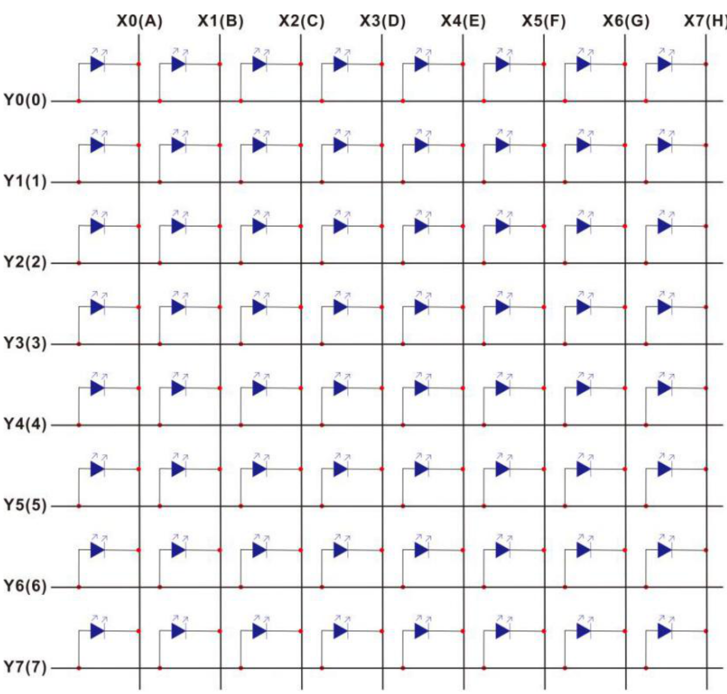
5.11.2 2.Components

		
ESP32*1	Breadboard*1	USB Cable*1
		
8*8 dot matrix module *1	220 Resistor*8	Jumper Wire*2

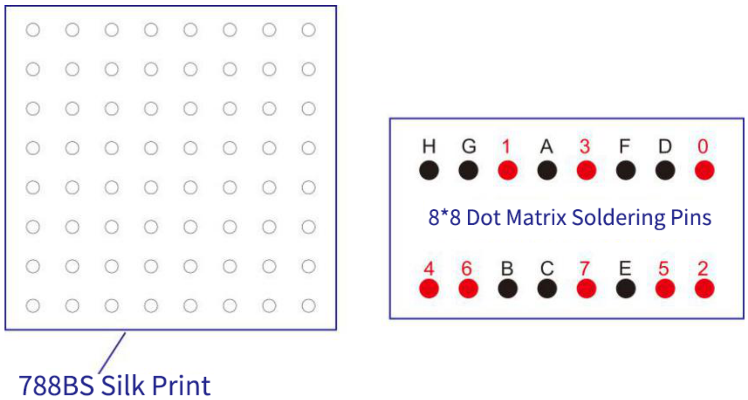
5.11.3 3.Component Knowledge

8*8 dot matrix module

The 8*8 dot matrix is composed of 64 LEDs, including row common anode and row common cathode. Our module is row common anode, that is, each row has a line connecting the positive pole of the LED, and the column is connecting the negative pole of the LED lamp, as shown in the following figure :

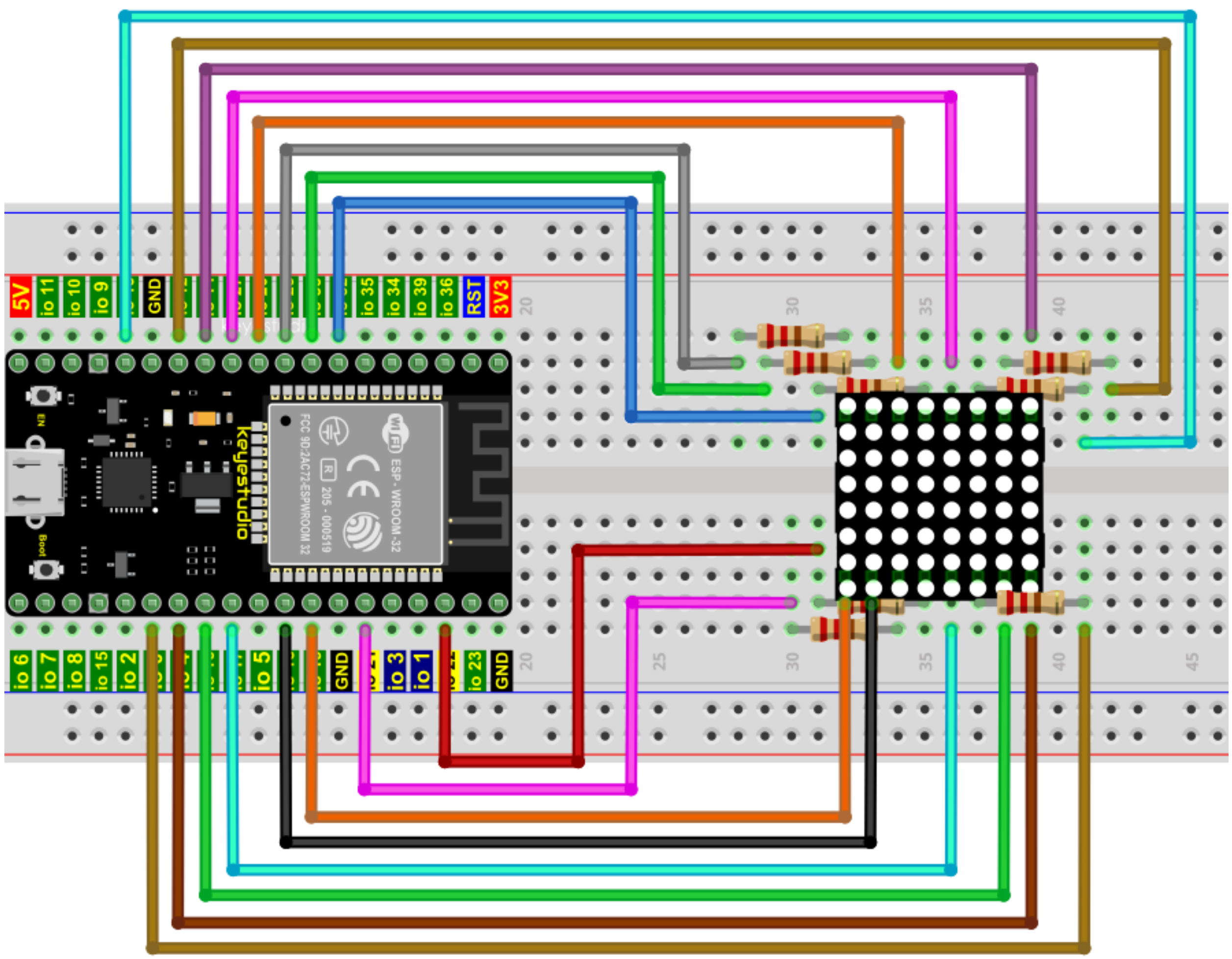


8*8 Dot Matrix LED Equivalent Circuit



8*8 Dot Matrix Outlook and Pinouts

5.11.4 4.Wiring Diagram



5.11.5 5.Test Code

```
//**********************************************************************************
/*
 * Filename    : 8×8 Dot-matrix Display
 * Description : 8×8 Dot-matrix displays numbers from 0 to 9.
 * Auther      : http//www.keyestudio.com
*/
int R[] = {14,26,4,27,19,16,18,17};
int C[] = {32,21,22,12,0,13,33,25};

unsigned char data_0[8][8] =
{
{0,0,1,1,1,0,0,0},
{0,1,0,0,0,1,0,0},
{0,1,0,0,0,1,0,0},
{0,1,0,0,0,1,0,0},
{0,1,0,0,0,1,0,0},
{0,1,0,0,0,1,0,0},
{0,1,0,0,0,1,0,0},
{0,0,1,1,1,0,0,0}
};

unsigned char data_1[8][8] =
{
{0,0,0,0,1,0,0,0},
{0,0,0,1,1,0,0,0},
{0,0,0,0,1,0,0,0},
{0,0,0,0,1,0,0,0},
{0,0,0,0,1,0,0,0},
{0,0,0,0,1,0,0,0},
{0,0,0,0,1,0,0,0},
{0,0,0,1,1,1,0,0}
};

unsigned char data_2[8][8] =
{
{0,0,1,1,1,0,0,0},
{0,1,0,0,0,1,0,0},
{0,0,0,0,0,1,0,0},
{0,0,0,0,1,0,0,0},
{0,0,0,1,0,0,0,0},
{0,0,1,0,0,0,0,0},
{0,1,1,1,1,1,0,0},
{0,0,0,0,0,0,0,0}
};

unsigned char data_3[8][8] =
{
{0,0,1,1,1,1,0,0},
{0,0,0,0,0,1,0,0},
{0,0,0,0,0,1,0,0},
{0,0,1,1,1,1,0,0},
```

(continues on next page)

(continued from previous page)

```
{0,0,0,0,0,1,0,0},
{0,0,0,0,0,1,0,0},
{0,0,1,1,1,1,0,0},
{0,0,0,0,0,0,0,0}
};

unsigned char data_4[8][8] =
{
{0,1,0,0,0,0,0,0},
{0,1,0,0,1,0,0,0},
{0,1,0,0,1,0,0,0},
{0,1,1,1,1,1,1,0},
{0,0,0,0,1,0,0,0},
{0,0,0,0,1,0,0,0},
{0,0,0,0,1,0,0,0},
{0,0,0,0,0,0,0,0}
};

unsigned char data_5[8][8] =
{
{0,1,0,0,0,0,0,0},
{0,1,1,1,1,1,0,0},
{0,1,0,0,0,0,0,0},
{0,1,1,1,1,1,0,0},
{0,0,0,0,0,1,0,0},
{0,0,0,0,0,1,0,0},
{0,1,1,1,1,1,0,0},
{0,0,0,0,0,0,0,0}
};

unsigned char data_6[8][8] =
{
{0,1,1,1,1,1,0,0},
{0,1,0,0,0,0,0,0},
{0,1,0,0,0,0,0,0},
{0,1,1,1,1,1,0,0},
{0,1,0,0,0,1,0,0},
{0,1,0,0,0,1,0,0},
{0,1,1,1,1,1,0,0},
{0,0,0,0,0,0,0,0}
};

unsigned char data_7[8][8] =
{
{0,0,0,0,0,0,0,0},
{0,1,1,1,1,1,0,0},
{0,0,0,0,0,1,0,0},
{0,0,0,0,1,0,0,0},
{0,0,0,1,0,0,0,0},
{0,0,1,0,0,0,0,0},
{0,1,0,0,0,0,0,0},
{0,0,0,0,0,0,0,0}
```

(continues on next page)

(continued from previous page)

```
};

unsigned char data_8[8][8] =
{
{0,1,1,1,1,1,0,0},
{0,1,0,0,0,1,0,0},
{0,1,0,0,0,1,0,0},
{0,1,1,1,1,1,0,0},
{0,1,0,0,0,1,0,0},
{0,1,0,0,0,1,0,0},
{0,1,1,1,1,1,0,0},
{0,0,0,0,0,0,0,0}
};

unsigned char data_9[8][8] =
{
{0,1,1,1,1,1,0,0},
{0,1,0,0,0,1,0,0},
{0,1,0,0,0,1,0,0},
{0,1,1,1,1,1,0,0},
{0,0,0,0,0,1,0,0},
{0,0,0,0,0,1,0,0},
{0,1,1,1,1,1,0,0},
{0,0,0,0,0,0,0,0}
};

void Display(unsigned char dat[8][8])
{
for(int c = 0; c<8;c++)
{
digitalWrite(C[c],LOW);
for(int r = 0;r<8;r++)
{
digitalWrite(R[r],dat[r][c]);
}
delay(1);
Clear();
}
}

void Clear()
{
for(int i = 0;i<8;i++)
{
digitalWrite(R[i],LOW);
digitalWrite(C[i],HIGH);
}
}

void setup(){
  for(int i = 0;i<8;i++)
  {
```

(continues on next page)

(continued from previous page)

```
    pinMode(R[i],OUTPUT);
    pinMode(C[i],OUTPUT);
  }

}

void loop(){
  for (int i = 1; i <= 100; i = i + (1)) {
    Display(data_0);
  }
  for (int i = 1; i <= 100; i = i + (1)) {
    Display(data_1);
  }
  for (int i = 1; i <= 100; i = i + (1)) {
    Display(data_2);
  }
  for (int i = 1; i <= 100; i = i + (1)) {
    Display(data_3);
  }
  for (int i = 1; i <= 100; i = i + (1)) {
    Display(data_4);
  }
  for (int i = 1; i <= 100; i = i + (1)) {
    Display(data_5);
  }
  for (int i = 1; i <= 100; i = i + (1)) {
    Display(data_6);
  }
  for (int i = 1; i <= 100; i = i + (1)) {
    Display(data_7);
  }
  for (int i = 1; i <= 100; i = i + (1)) {
    Display(data_8);
  }
  for (int i = 1; i <= 100; i = i + (1)) {
    Display(data_9);
  }
}
//**********************************************************************************
```

5.11.6 6.Test 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 displays the numbers 0~9 respectively.

5.12 Project 1174HC595N Control 8 LEDs

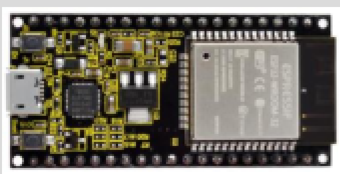
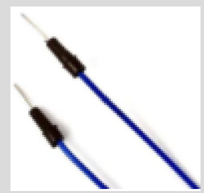
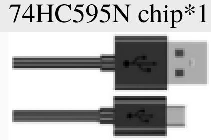
5.12.1 1.Introduction

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 an ESP32a 74HC595 chip and LEDs to make a flowing water light to understand the function of the 74HC595 chip.

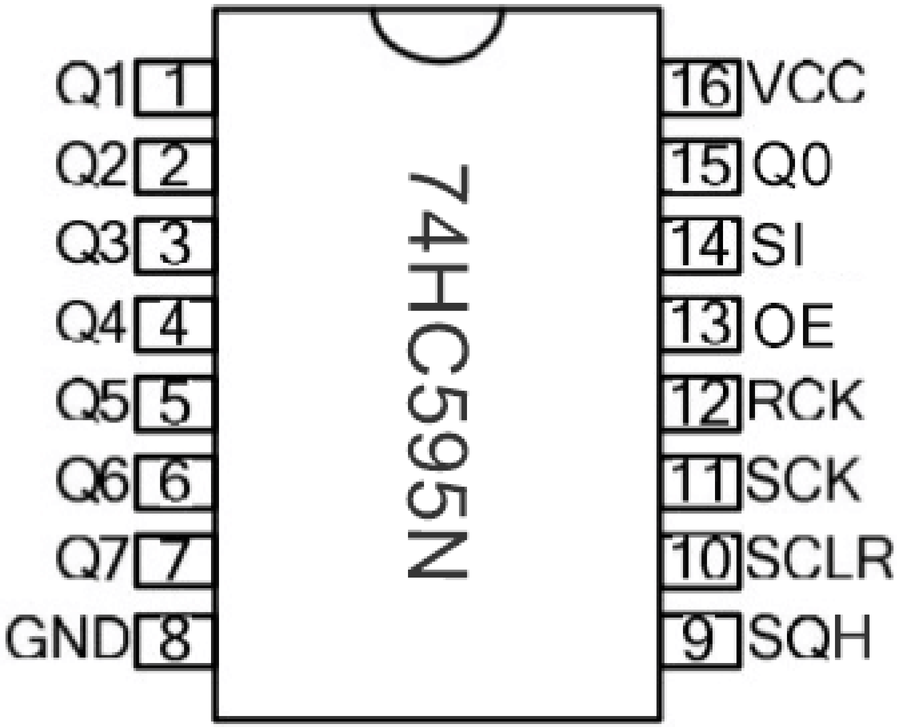
5.12.2 2.Components

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

5.12.3 3.Component Knowledge

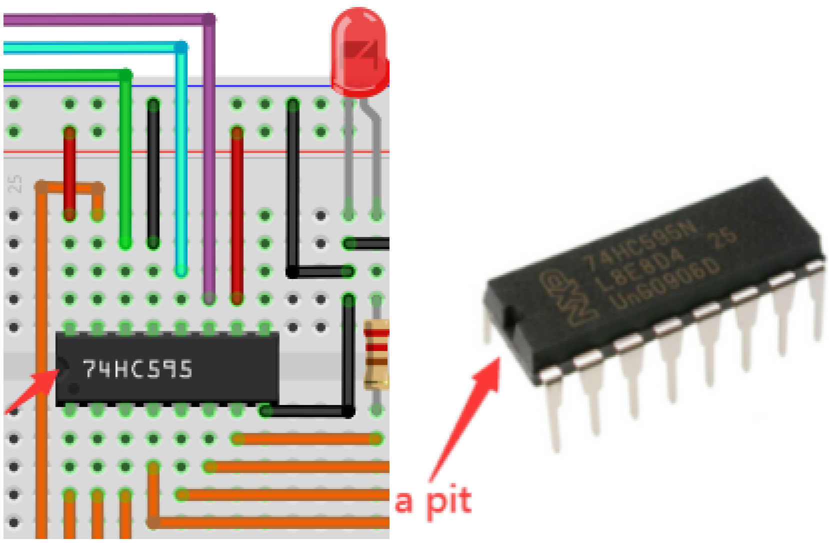
74HC595N Chip:

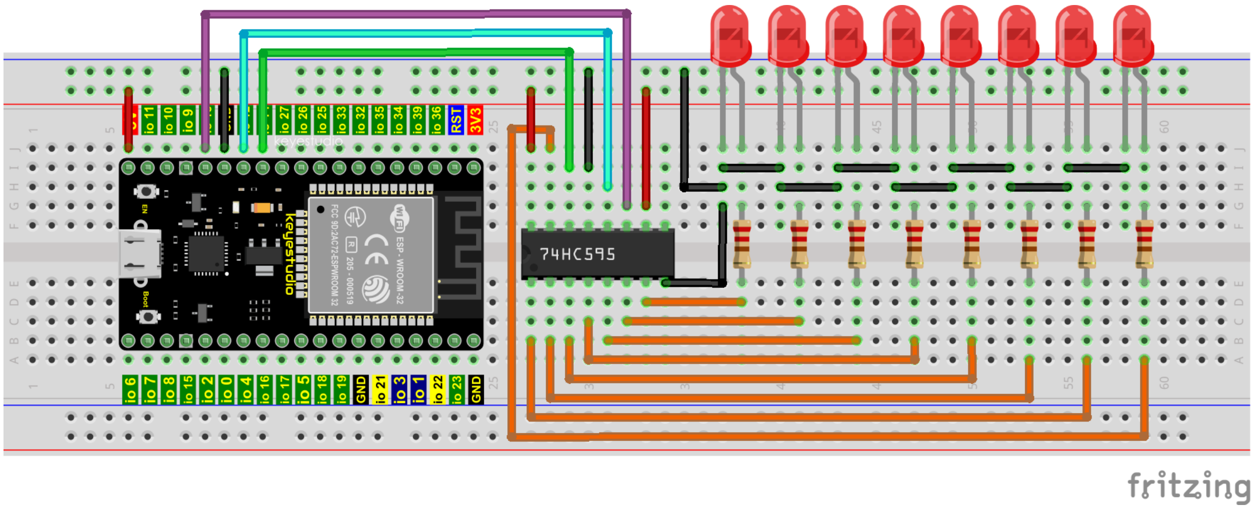
The 74HC595 chip is used to convert serial data into parallel data. A 74HC595 chip can convert the serial data of one byte into 8 bits, and send its corresponding level to each of the 8 ports correspondingly. With this characteristic, the 74HC595 chip can be used to expand the IO ports of an ESP32. At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



The ports of the 74HC595 chip are described as follows

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





5.12.4 5.Test Code

```
//**********************************************************************
/*
 * 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;
```

(continues on next page)

(continued from previous page)

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

5.12.5 6.Test 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.

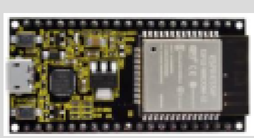
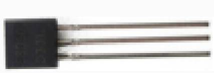
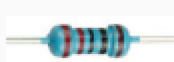
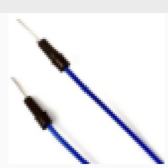
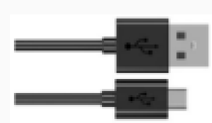
5.13 Project 12Active Buzzer

5.13.1 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.

5.13.2 2.Components

			
ESP32*1	Breadboard*1	Active buzzer*1	
			
NPN Transistor(S8050)*1	1k Resistor*1	Jumper Wires	USB Cable*1

5.13.3 3.Component Knowledge

Active buzzer:

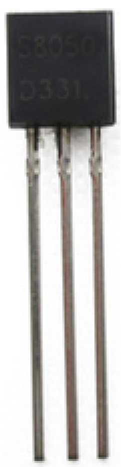
Active buzzer inside has a simple oscillator circuit, which can convert constant direct current into a certain frequency pulse signal. Once active buzzer receives a high level, it will produce sound.

Passive buzzer is an internal without vibration source integrated electronic buzzer, it must be driven by 2k to 5k square wave, rather than a DC signal.

The two buzzers are very similar in appearance, but one buzzer with a green circuit board is a passive buzzer, while the other buzzer with black tape is an active buzzer.

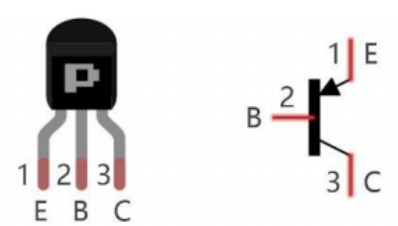
Passive buzzers don't have positive polarity, but active buzzers have. As shown below:

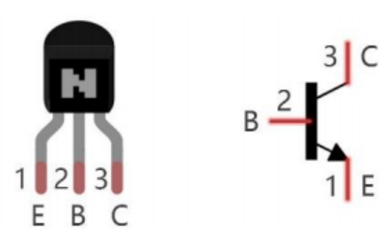
Transistor:



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

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





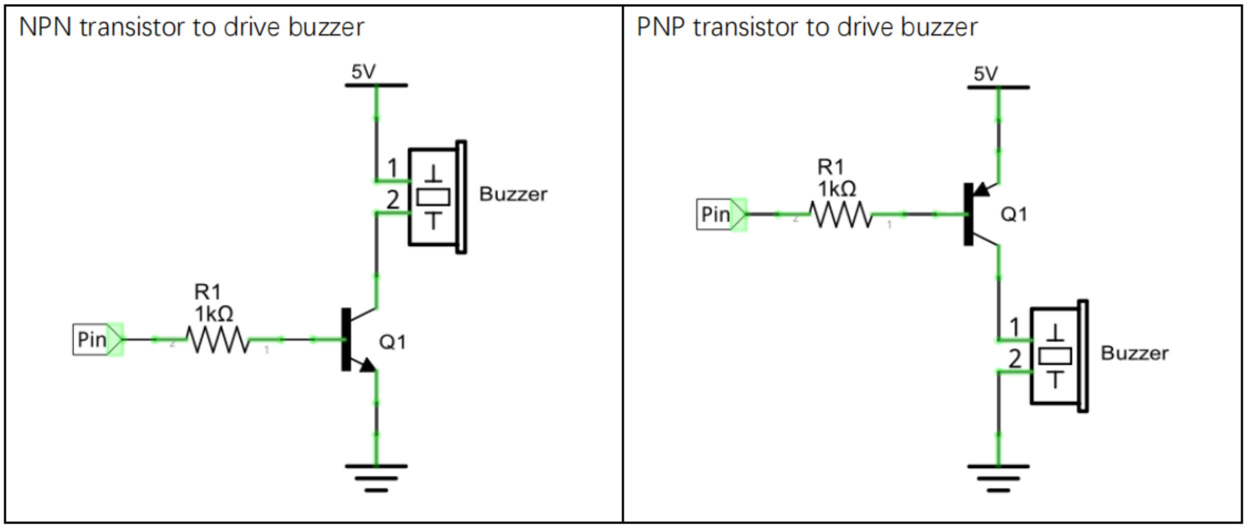
PNP transistor NPN transistor

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

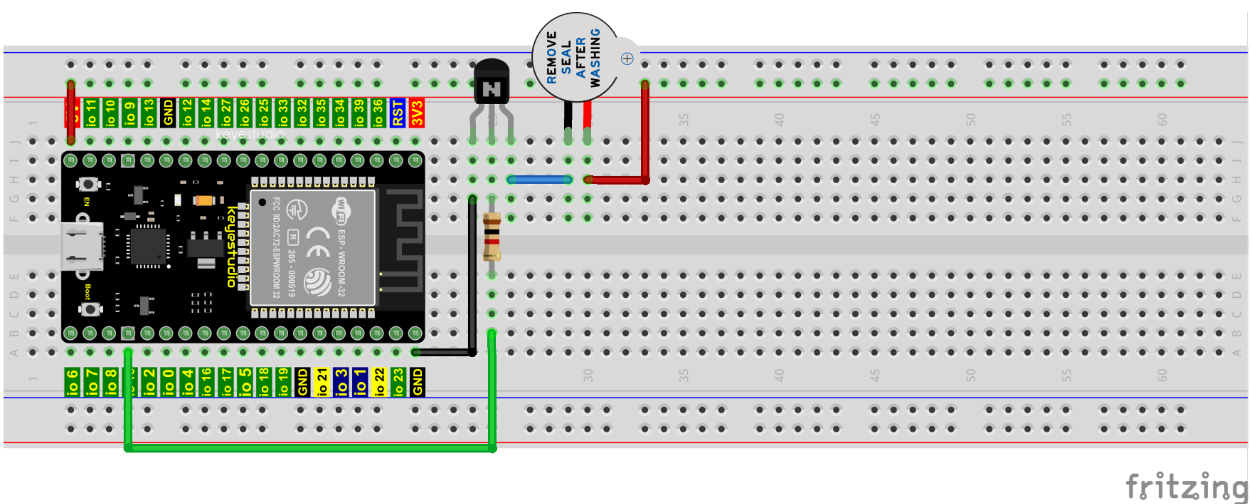
Based on the transistor's characteristics, it is often used as a switch in digital circuits. As micro-controller's capacity to output current is very weak, we will use transistor to amplify current and drive large-current components.

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.



5.13.4 4.Wiring Diagram



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

5.13.5 5.Test Code

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

5.13.6 6.Test Result

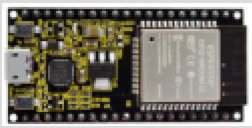
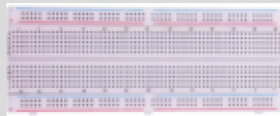
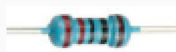
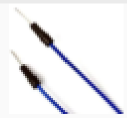
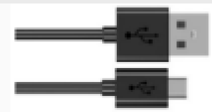
Compile and upload the code to ESP32, after the code is uploaded successfully, power up with a USB cable and you will see that the active buzzer beeps.

5.14 Project 13Passive Buzzer

5.14.1 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.

5.14.2 2.Components

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

5.14.3 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.

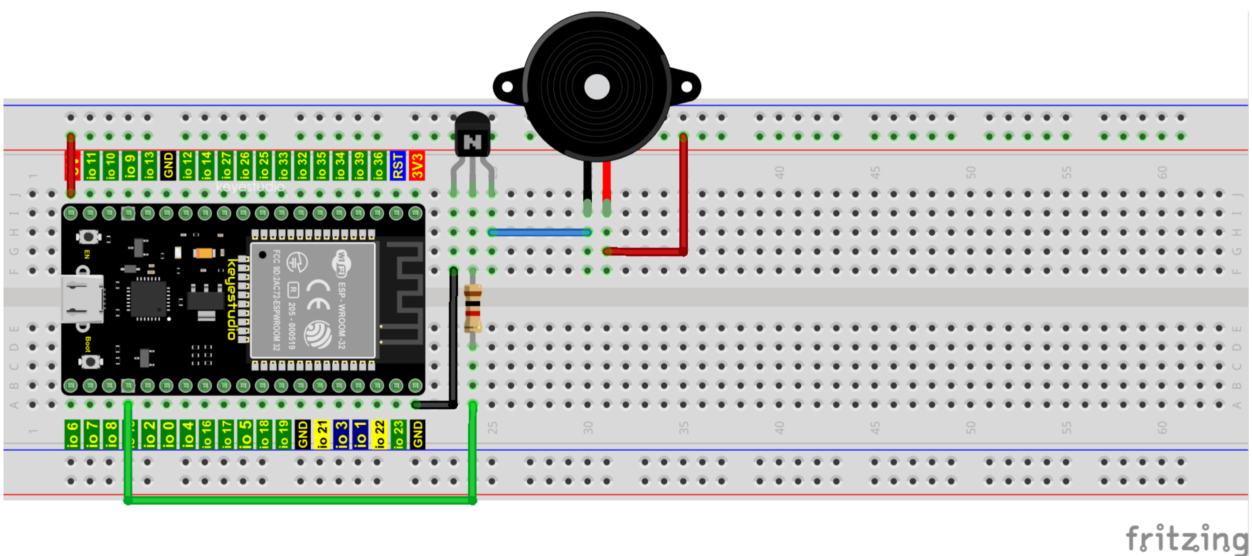
The two buzzers are very similar in appearance, but one buzzer with a green circuit board is a passive buzzer and the other buzzer with black tape is an active buzzer.

Passive buzzers cannot distinguish between positive polarity while active buzzers can.

Transistor:

Please refer to Project 12.

5.14.4 4.Wiring Diagram



5.14.5 5.Test Code

```
//**********************************************************************
/*
 * Filename    : Passive Buzzer
 * Description : Passive Buzzer sounds the alarm.
 * Auther      : http//www.keyestudio.com
*/
#define LEDC_CHANNEL_0 0
```

(continues on next page)

(continued from previous page)

```
// 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);

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

5.14.6 6.Test Result

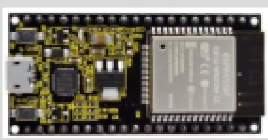
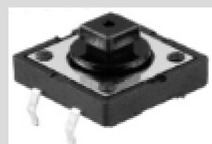
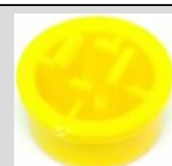
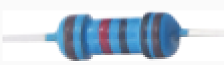
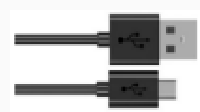
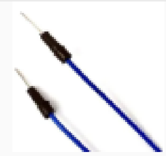
Compile and upload the code to ESP32, after the code is uploaded successfully, power up with a USB cable and you will see that the passive buzzer plays music.

5.15 Project 14: Mini Table Lamp

5.15.1 1.Introduction

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

5.15.2 2.Components

				
ESP32*1	Breadboard*1	Button*1	Button Cap*1	
				
10K Resistor*1	Red LED*1	22 Resistor*1	USB Cable*1	Jumper Wires

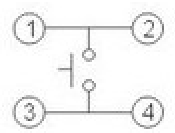
5.15.3 3.Component Knowledge

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

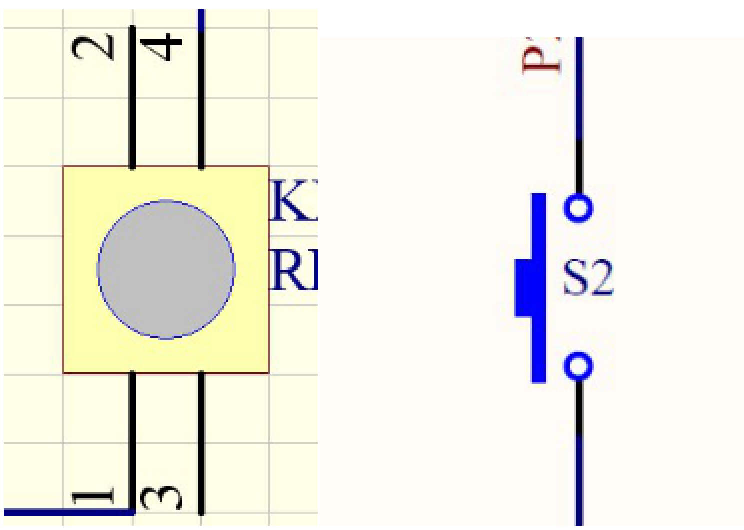


.

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:

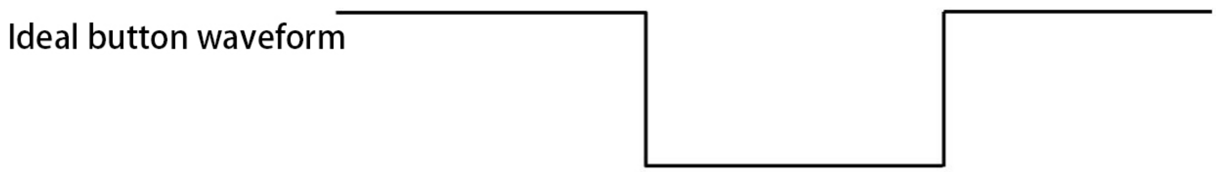


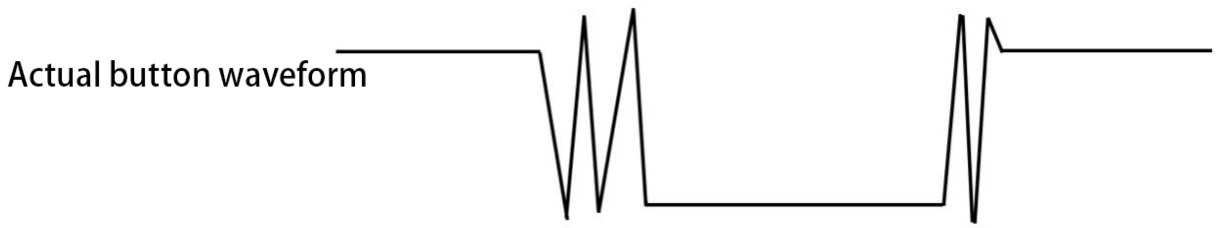
What is button [shake](javascript:;)?

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.



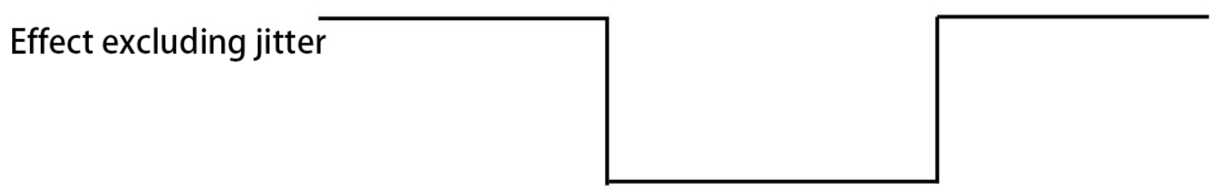


How to eliminate the [shake](javascript:;)?

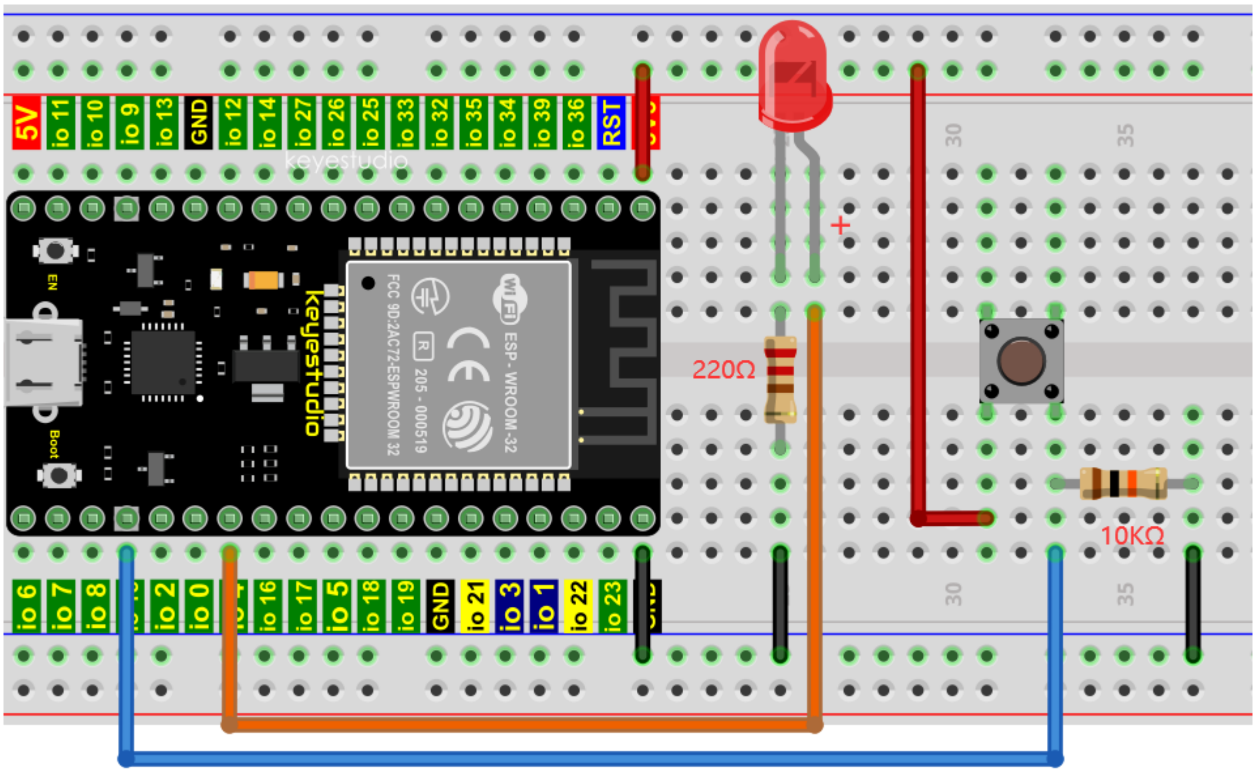
There are two common methods, namely fix [shake](javascript:;) in the software and hardware. We only discuss the [shake](javascript:;) removal in the software.

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.

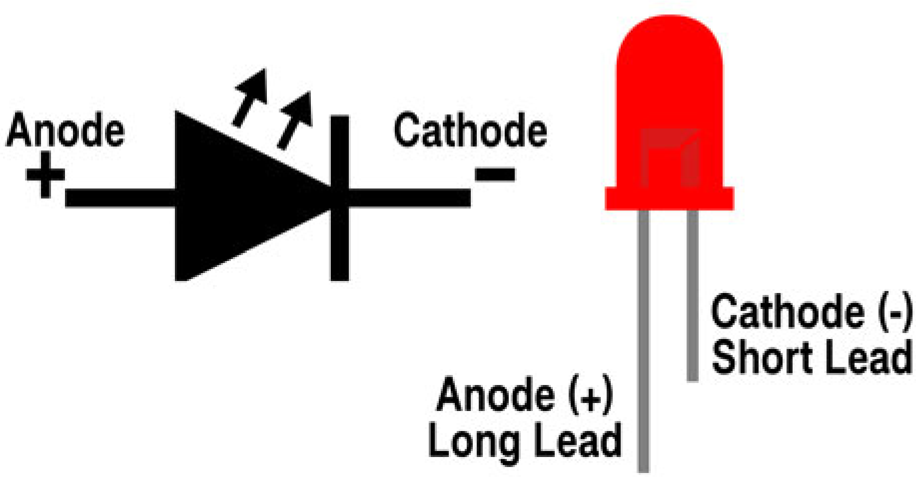


5.15.4 4.Wiring Diagram

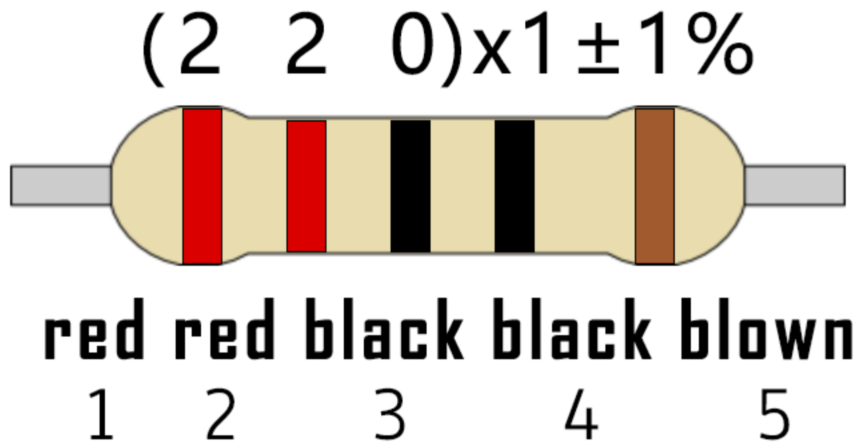


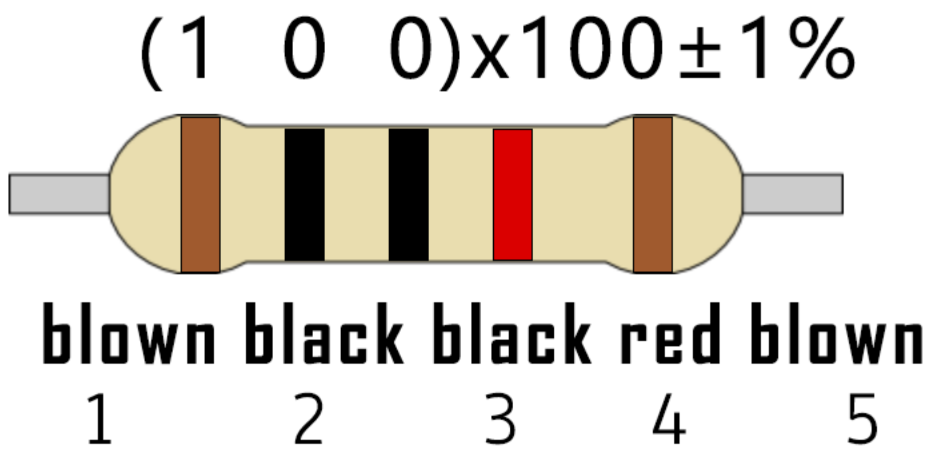
Note:

How to connect the LED



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





5.15.5 5.Test code

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

(continues on next page)

(continued from previous page)

```
    delay(20);
    if (digitalRead(PIN_BUTTON) == LOW) {
      reverseGPIO(PIN_LED);
    }
    while (digitalRead(PIN_BUTTON) == LOW);
  }
}

void reverseGPIO(int pin) {
  ledState = !ledState;
  digitalWrite(pin, ledState);
}
//************************************************************************
```

5.15.6 6.Test 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

5.16.1 1.Introduction

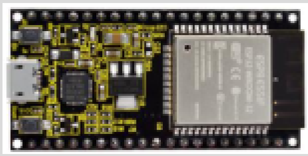
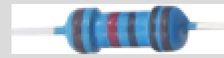
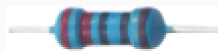
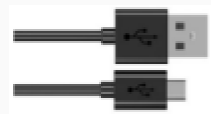
The ancients without electronic clock, so the hourglass are invented to measure time. The hourglass has a large capacity on both sides, and which is filled with fine sand on one side.

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 to make an electronic hourglass.

5.16.2 2.Components

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

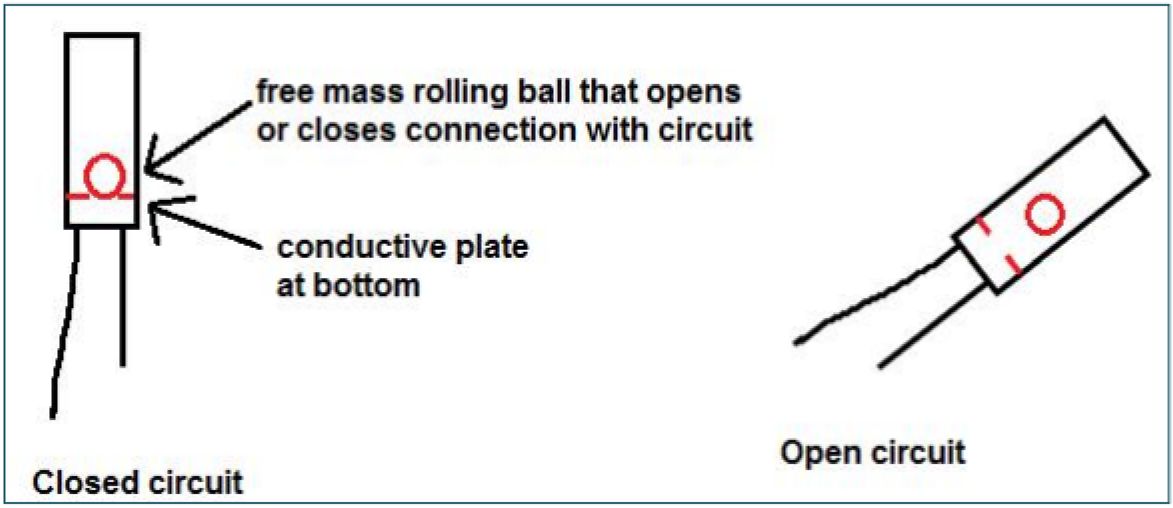
5.16.3 3.Component Knowledge

Tilt switch is also called digital switch. Inside is a metal ball that can roll.

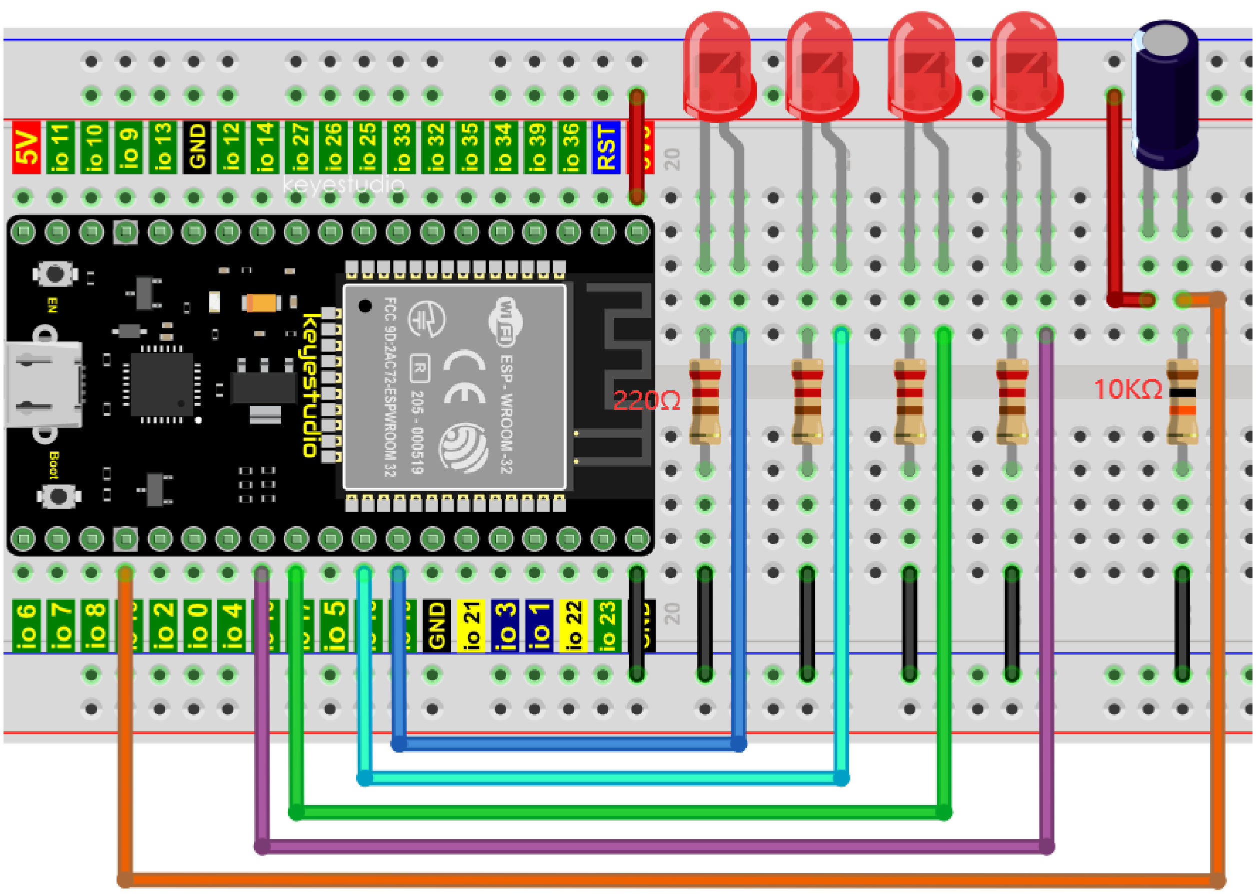
The principle of rolling the metal ball to contact with the conductive plate at the bottom, which is used to control the on and off of the circuit. When it is a rolling ball tilt sensing switch with single directional trigger, the tilt sensor is tilted toward the trigger end (two gold-plated pin ends), 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:

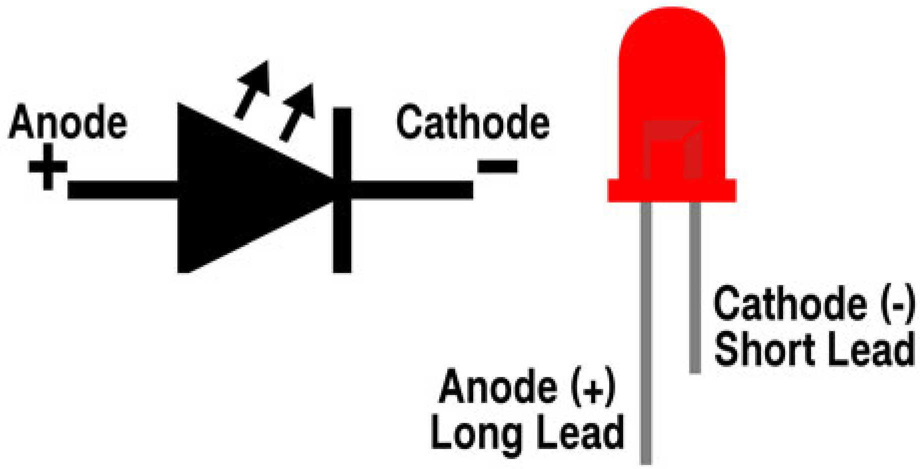


5.16.4 4.Wiring Diagram

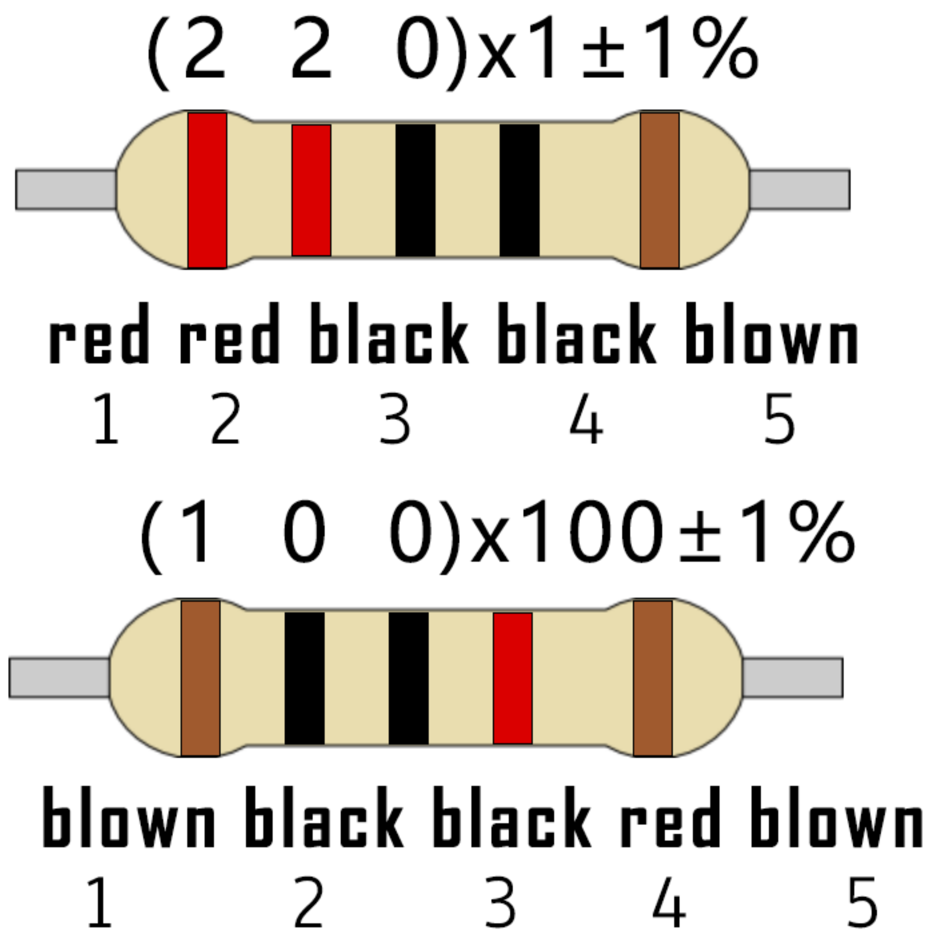


Note:

How to connect the LED



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



5.16.5 5.Test Code

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

5.16.6 6.Test 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 16: I2C 128×32 LCD

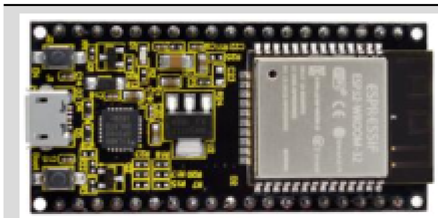
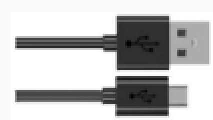
5.17.1 1.Introduction

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.

5.17.2 2.Components

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

5.17.3 3.Component Knowledge

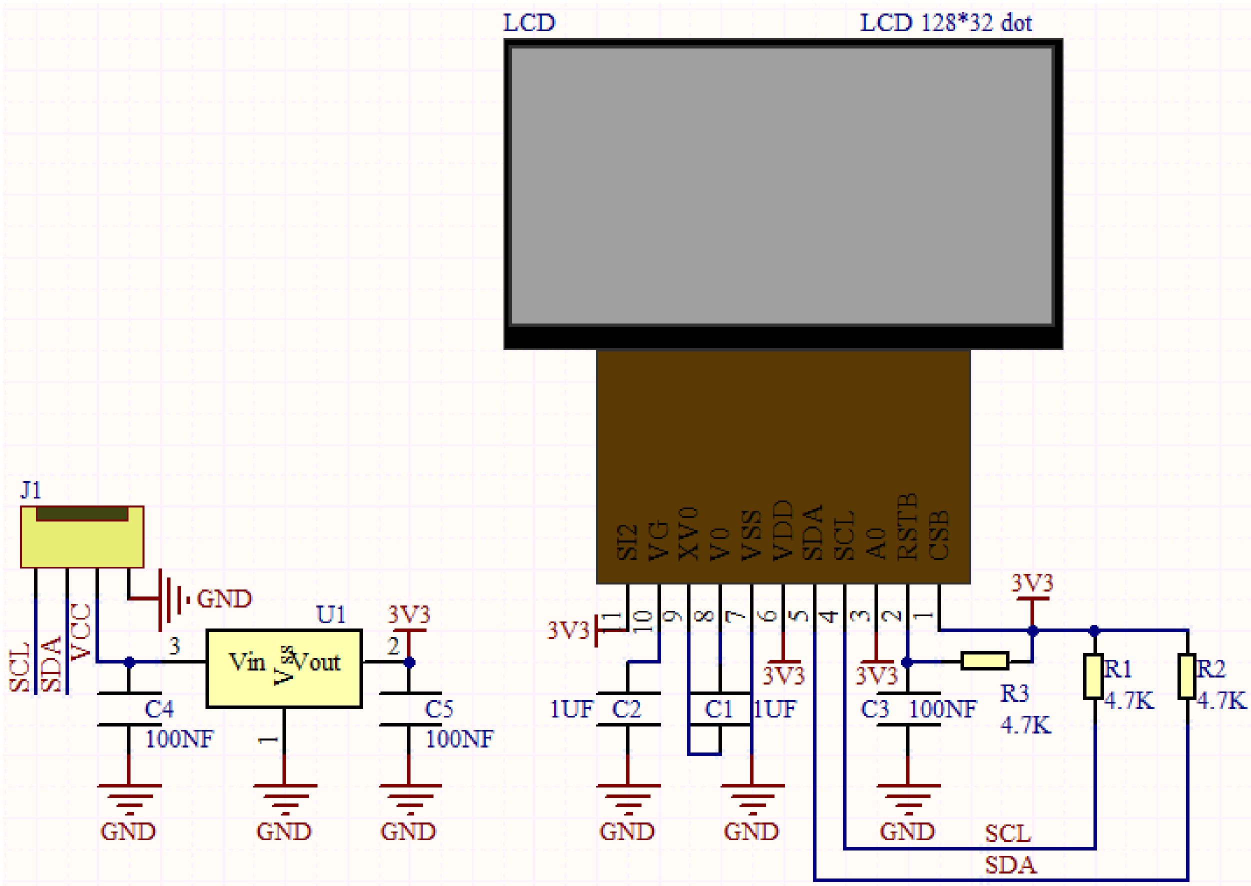
LCD_128X32_DOT:

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.

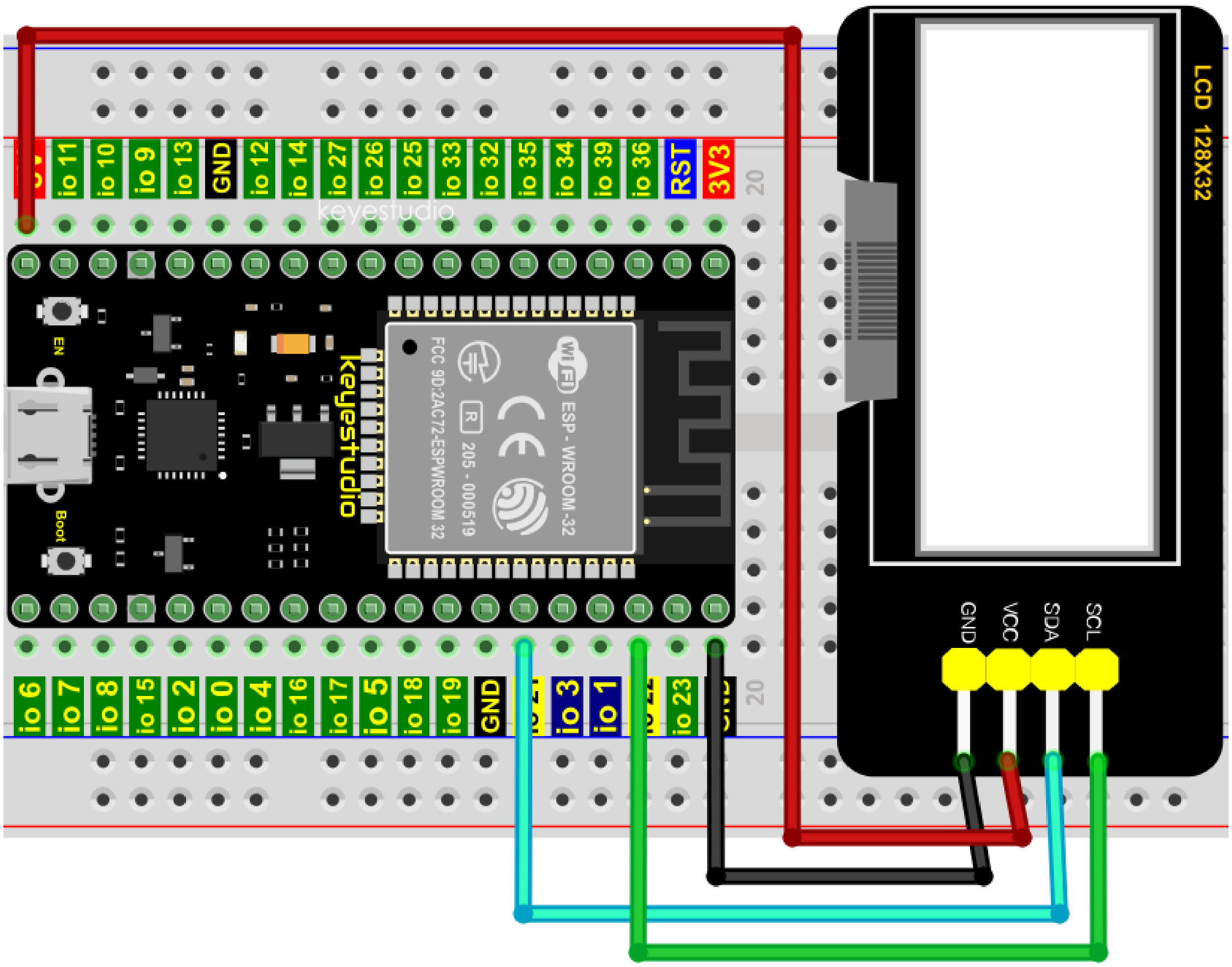
Schematic diagram of LCD_128X32_DOT



Features:

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

5.17.4 4.Wiring Diagram



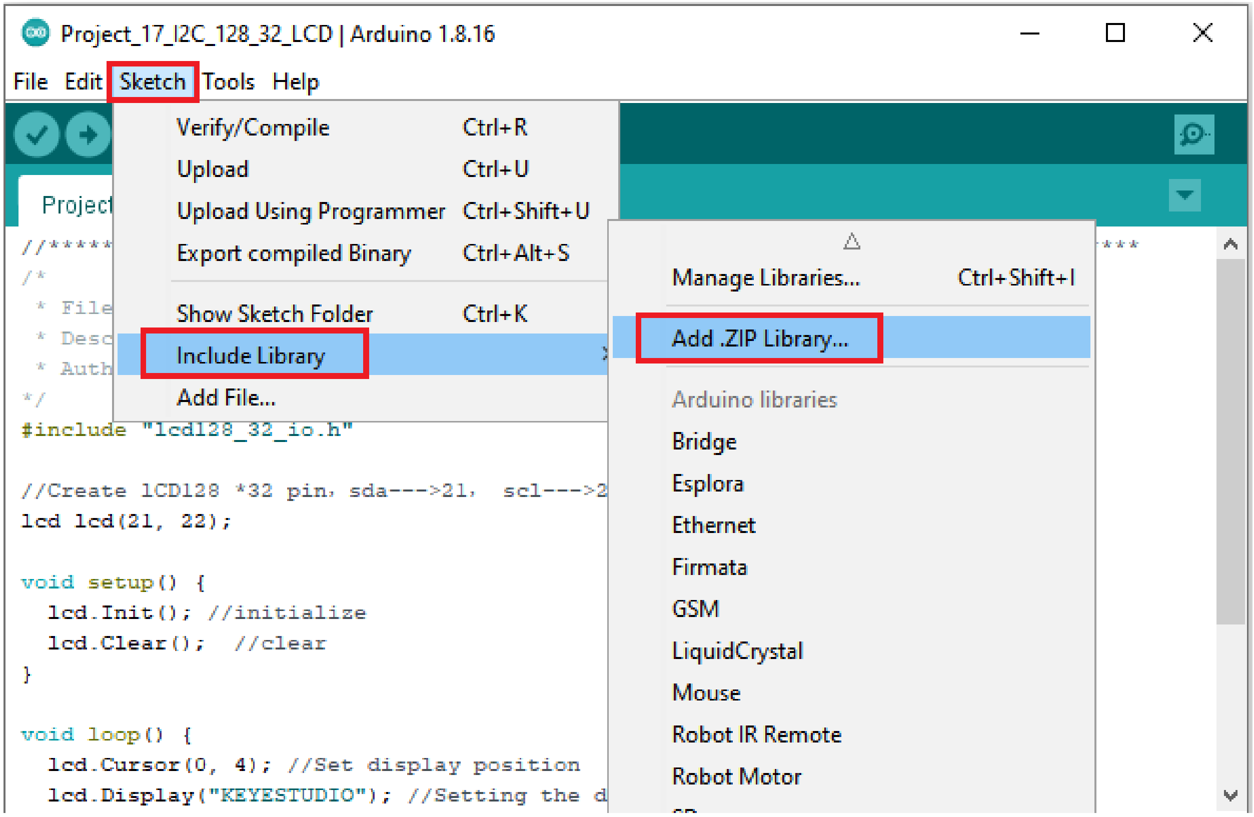
5.17.5 5.Adding the lcd128_32_io library

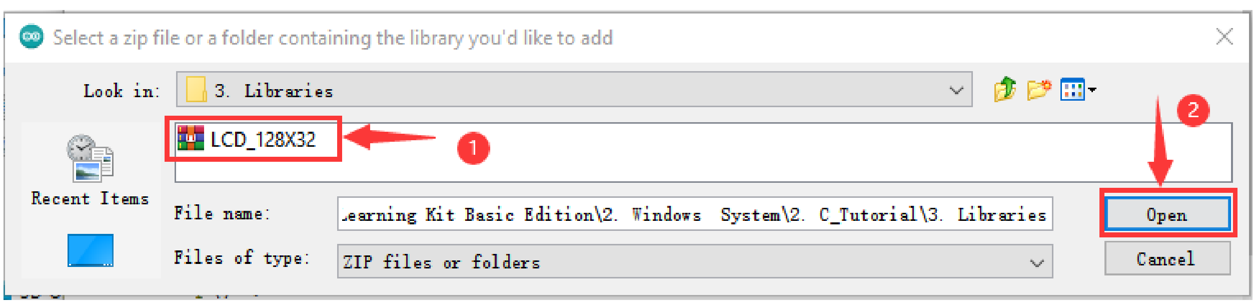
This code uses a library named "**lcd128_32_io**", if you haven't installed it yet, please do so before learning. The steps to add third-party libraries are as follows:

Open the Arduino IDEclick "Sketch"→"Include Library"→"Add .ZIP Library…".

Click on the link to download the library fileArduino C "lcd128_32_io.h" Librarie

Select the **LCD_128X32.ZIP** file and then click"Open".





5.17.6 6.Test Code

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

(continues on next page)

(continued from previous page)

```
}

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("%^&(){}:;'|?,.~\\[]");
}
//*************************************************************************************
```

5.17.7 7.Test Result

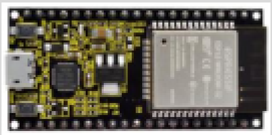
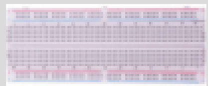
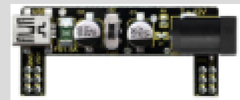
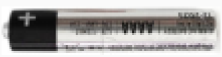
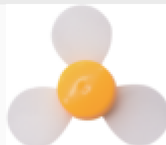
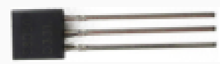
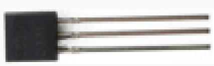
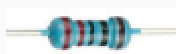
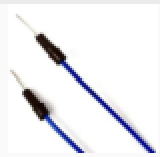
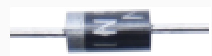
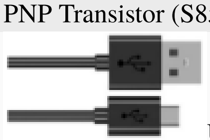
Compile and upload the code to ESP32, after the code is uploaded successfully, power up with a USB cable and you will see that the 128X32LCD module display will show"KEYESTUDIO"at the first line"ABCDEFGHIJKLMNOPQR"will be displayed at the second line"123456789±*/<>=$@"will be shown at the third line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

5.18 Project 17Small Fan

5.18.1 1.Introduction

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

5.18.2 2.Components

			
ESP32*1	Breadboard*1	6 AA Battery Holder*1	Breadboard Power Module*1
			
AA Battery(Self-prepared)*6	Fan*1	DC Motor*1	NPN Transistor (S8050)*1
			
PNP Transistor (S8550)*1	1K Resistor*1	Jumper Wire	Diode*1
 USB Cable*1			

5.18.3 3.Component Knowledge

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.

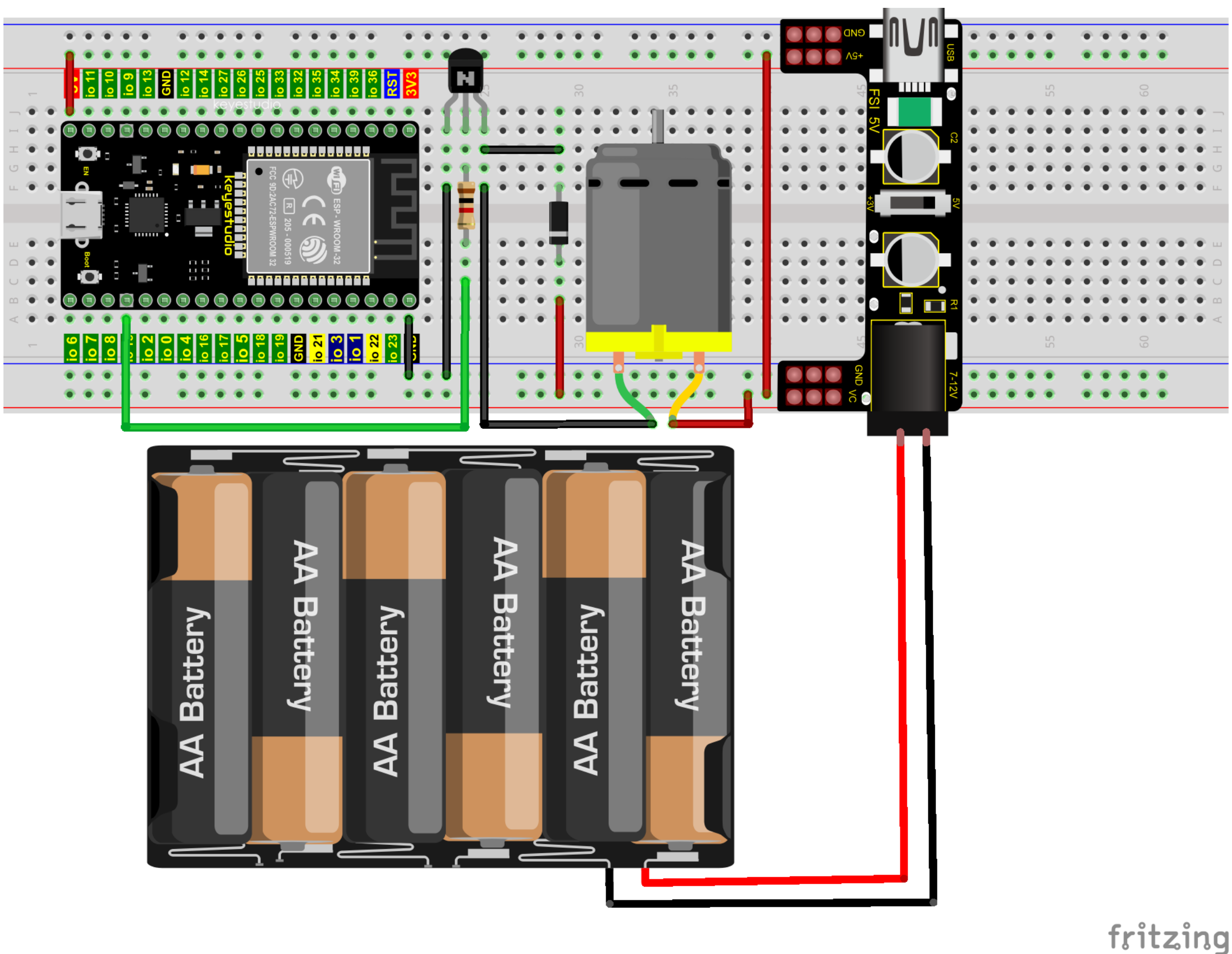
The output voltage is constant for the DC5V, and another way is powered by DC 7-12V, output controlled by the slide switch, respectively for DC5V and DC3.3V.

If the other power supply is DC 7-12v, 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 7-12V power supply terminal of the other power supply is DC3.3V.

Specification

- Applied to MB102 breadboard
- Input voltageDC 7-12V 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

5.18.4 4.Wiring Diagram 1



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

5.18.5 5. Test Code

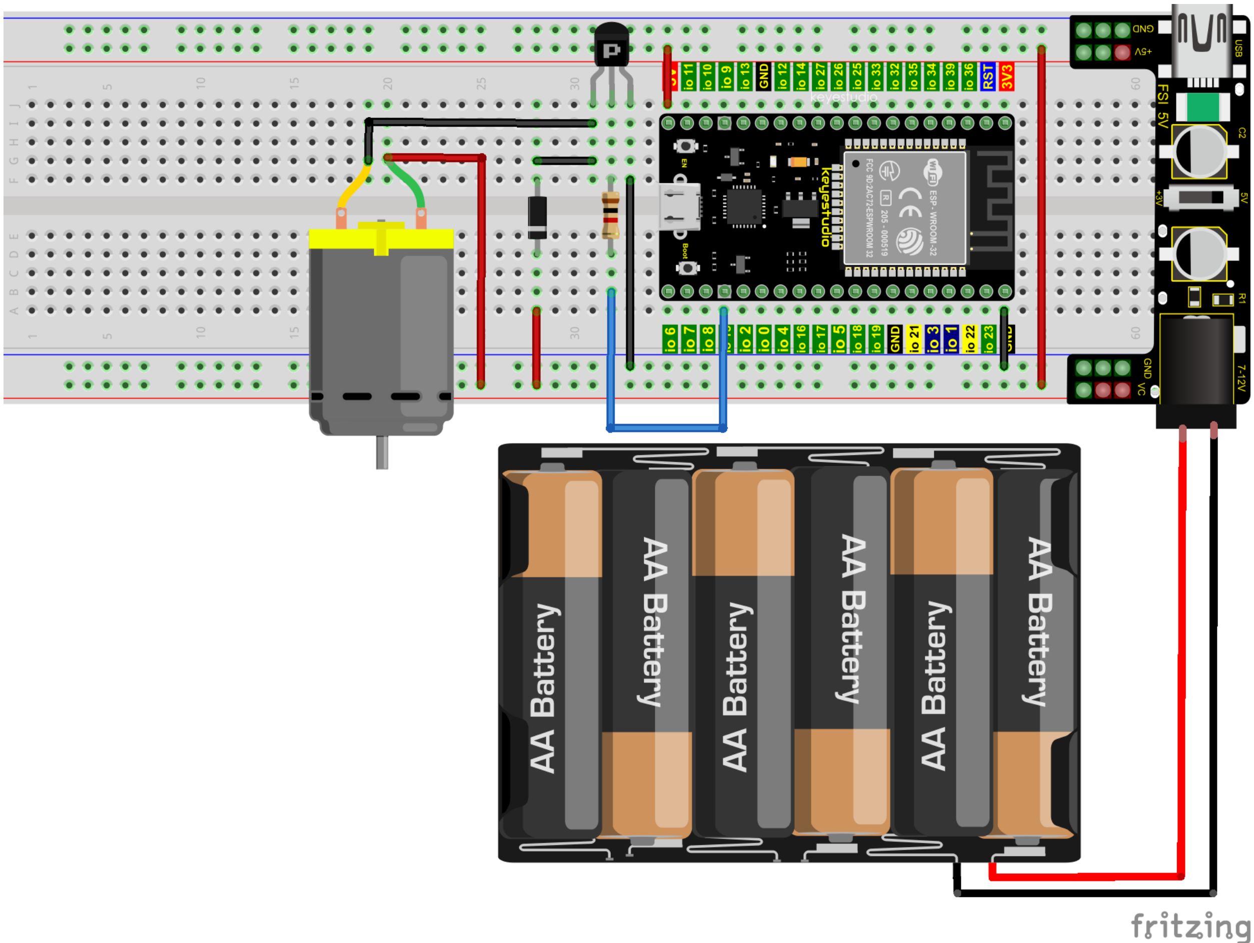
```
//**********************************************************************
/*
 * Filename    : Small_Fan
 * Description : S8050 triode drives the motor working
 * Auther      : http//www.keyestudio.com
*/

void setup() {

  pinMode(15, OUTPUT); // Initialize pin 15 as output.
}

void loop() {
  digitalWrite(15, HIGH);   // Turn on the motor (HIGH means HIGH level)
  delay(4000);              // Delay 4 seconds
  digitalWrite(15, LOW);    // Reduce the voltage and turn off the motor
  delay(2000);              // Delay 2 seconds
}
//************************************************************************************
```

5.18.6 6.Test Result 1

Upload the code to the ESP32 and power up. The motor rotates for 4s, stops for 2s, in loop way.

5.18.7 7.Wiring Diagram 2

We use the S8550 PNP transistor to control the motor.



Note: wire up and connect a fan on the motor.

5.18.8 8.Test Code 2

```
//**********************************************************************
/*
 * Filename    : Small_Fan
 * Description : S8550 triode drives the motor working
 * Auther      : http//www.keyestudio.com
*/

void setup() {

  pinMode(15, OUTPUT); // Initialize pin 15 as output.
}

void loop() {
  digitalWrite(15, LOW);   // Turn on the motor (LOW means LOW level)
  delay(4000);              // Delay 4 seconds
  digitalWrite(15, HIGH);    // Raise the voltage and turn off the motor
  delay(2000);              // Delay 2 seconds
}
```

(continues on next page)

(continued from previous page)

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

5.18.9 9.Test Result 2

Upload the code to the ESP32 and power up. The motor rotates for 4s, stops for 2s, in loop way.

5.19 Project 18Dimming Light

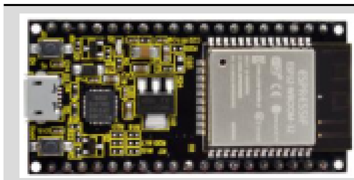
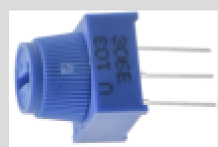
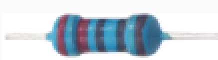
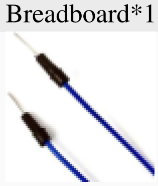
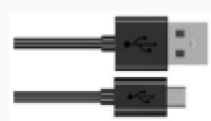
5.19.1 1.Introduction

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.

5.19.2 2.Components

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

5.19.3 3.Component Knowledge

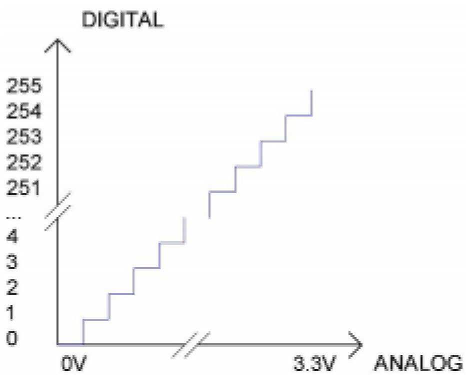
Adjustable potentiometer:

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:

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

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

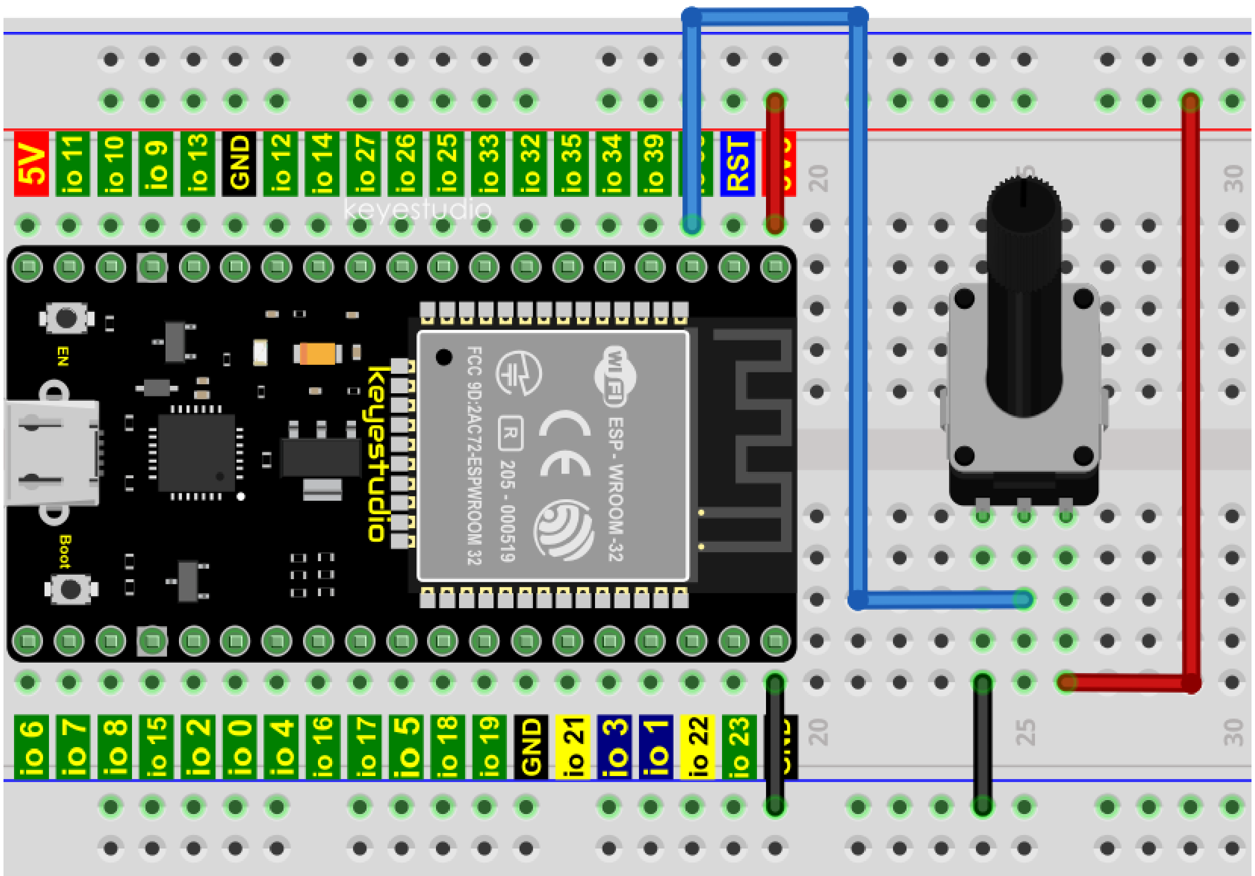
ESP32 has 16 pins can be used to measure analog signals. GPIO pin sequence number and analog pin definition are shown in the following table

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.

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




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

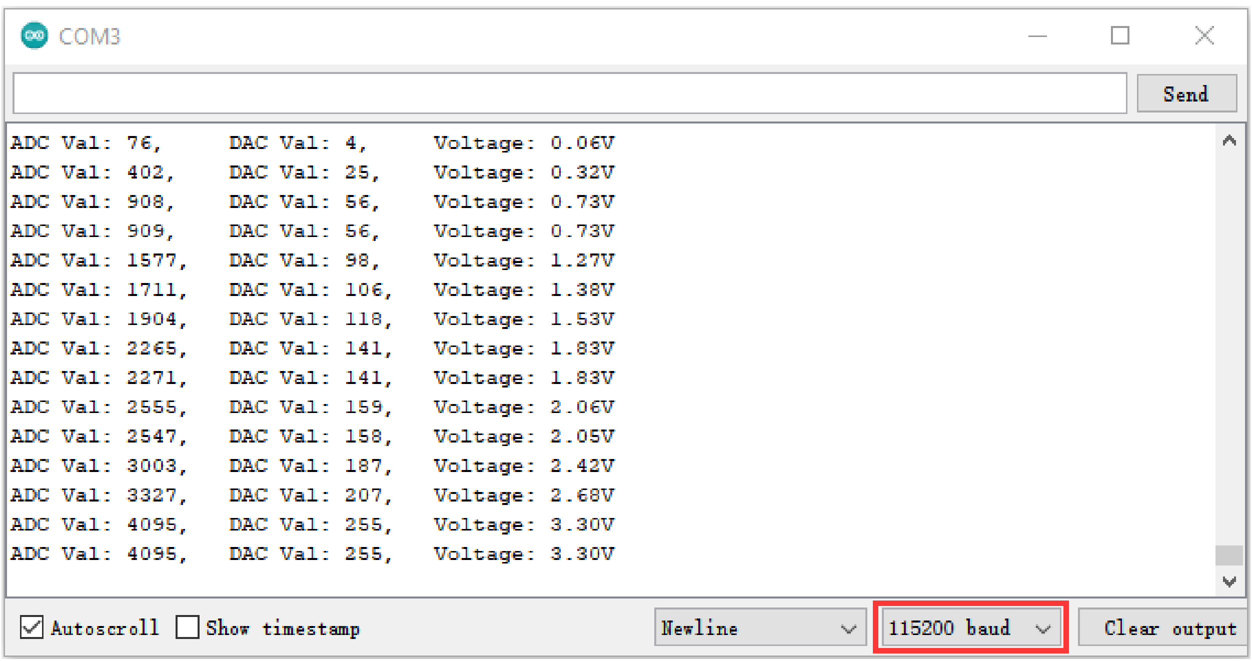
(continues on next page)

(continued from previous page)

```
//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 monitor window will print out the ADC value, DAC value and voltage value of the potentiometer.

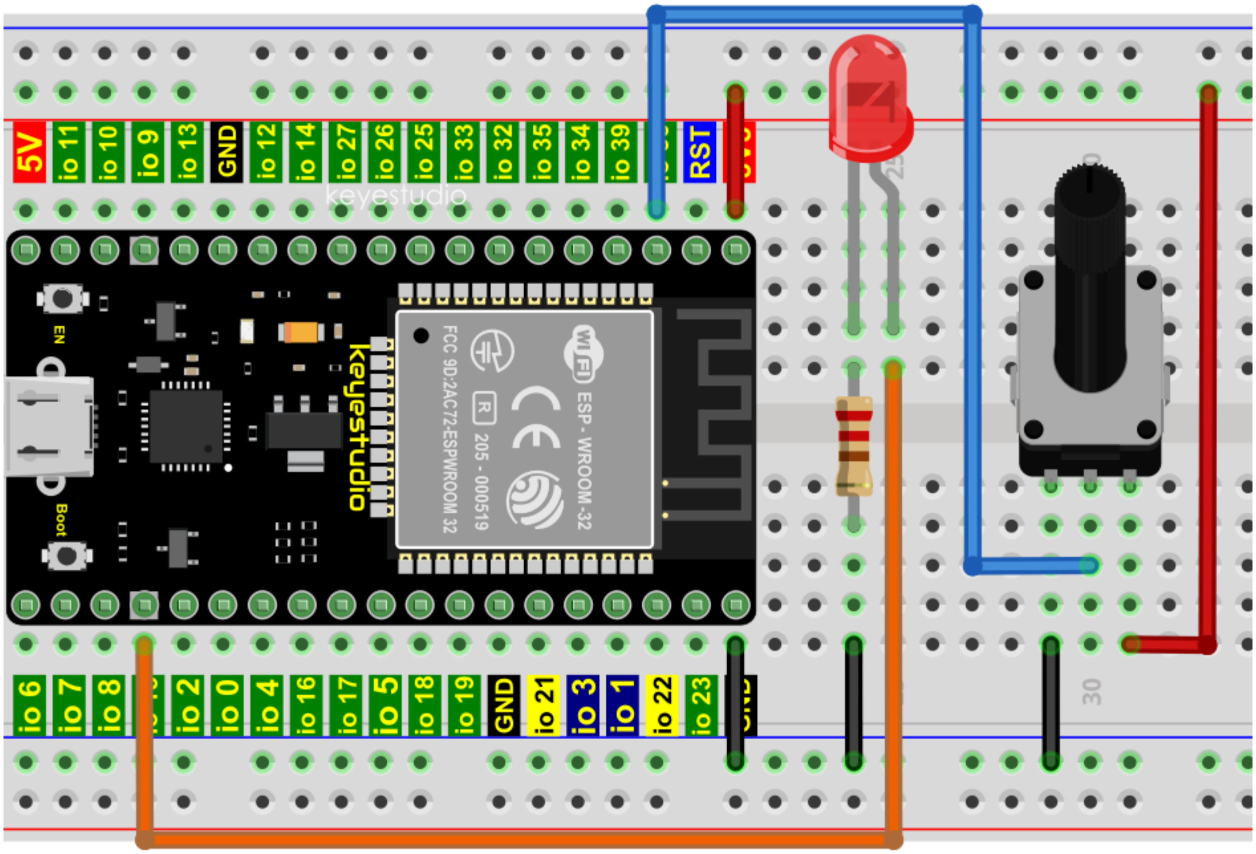
When turning the potentiometer handle, the ADC value, DAC value and voltage value will change. As shown below:



5.19.4 4.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 follow:



5.19.5 5.Test Code

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

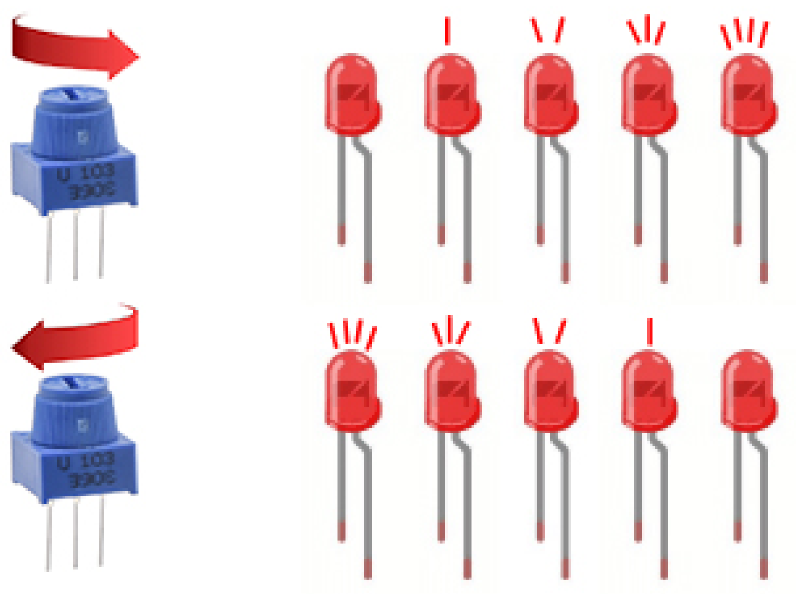
(continues on next page)

(continued from previous page)

```
  int pwmVal = adcVal;        // adcVal re-map to pwmVal
  ledcWrite(CHAN, pwmVal);    // set the pulse width.
  delay(10);
}
//*************************************************************************************
```

5.19.6 6.Test 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.



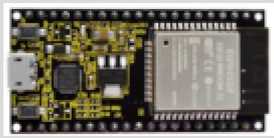
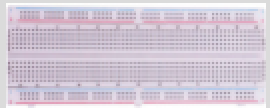
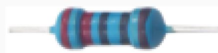
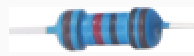
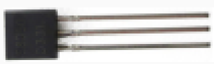
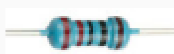
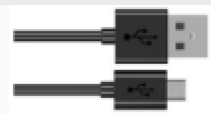
5.20 Project 19Flame Alarm

5.20.1 1.Introduction

Fire is a terrible disaster and fire alarm systems are very useful in houses, commercial 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.

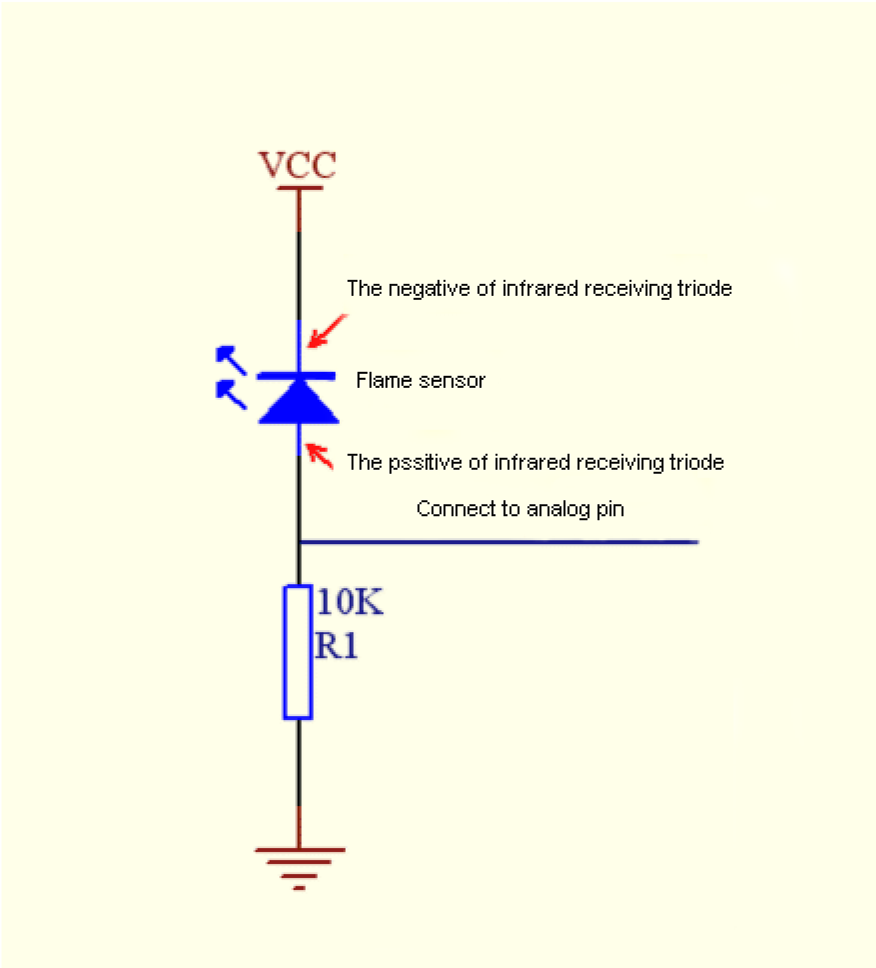
5.20.2 2.Components

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

5.20.3 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

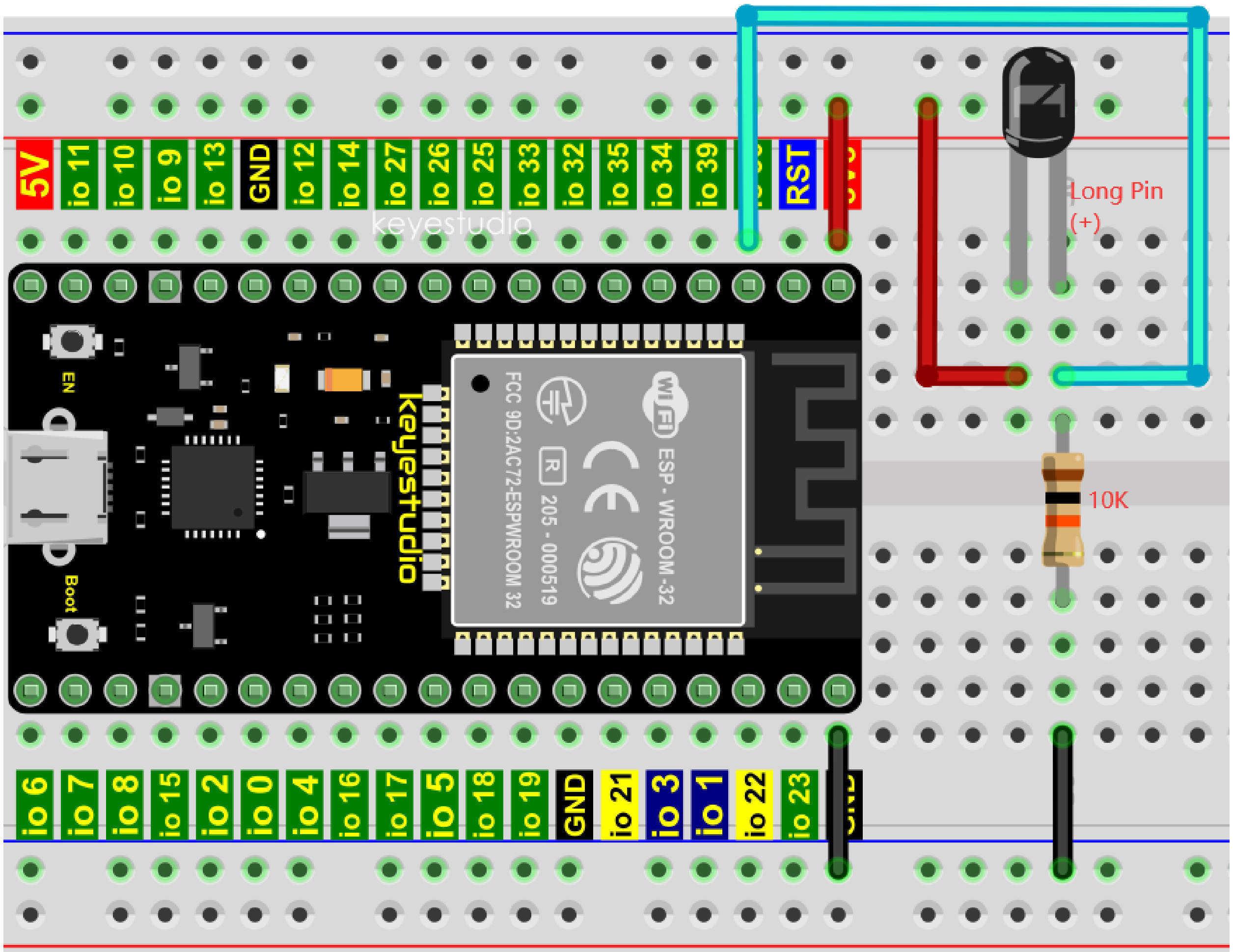


Note:

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.

5.20.4 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




```
//**********************************************************************************
/*
 * 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;
```

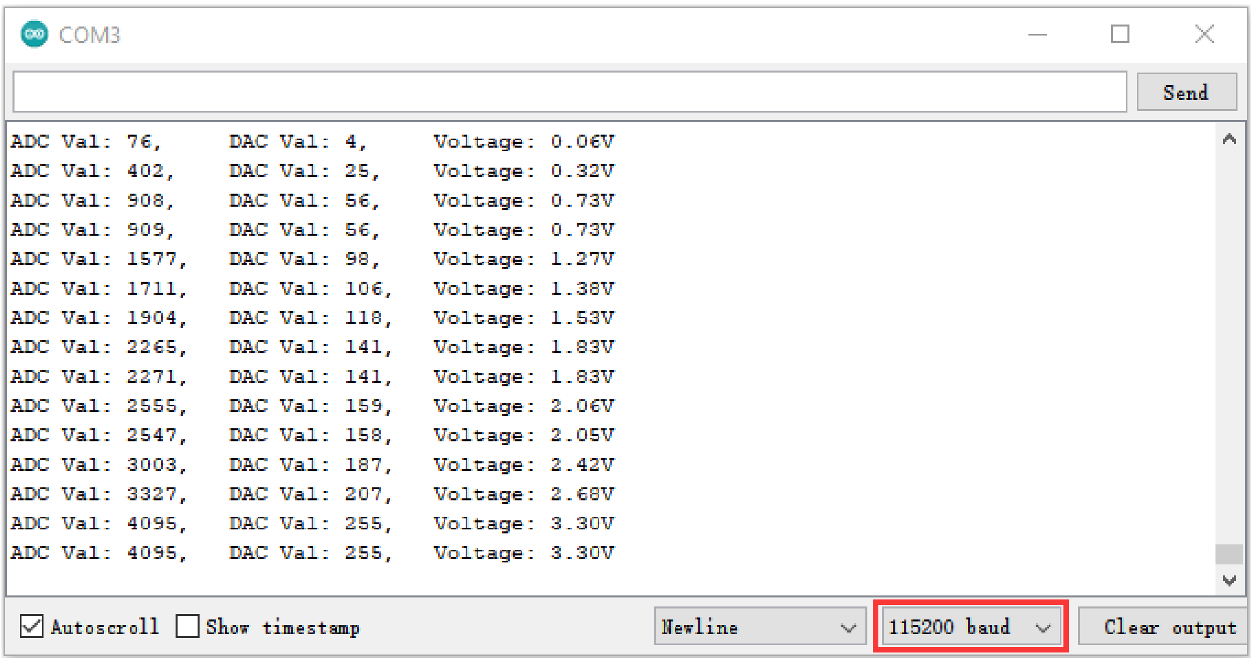
(continues on next page)

(continued from previous page)

```
  Serial.printf("ADC Val: %d, \t DAC Val: %d, \t Voltage: %.2fV\n", adcVal, dacVal,␣
↪voltage);
  delay(200);
}
//********************************************************************************
```

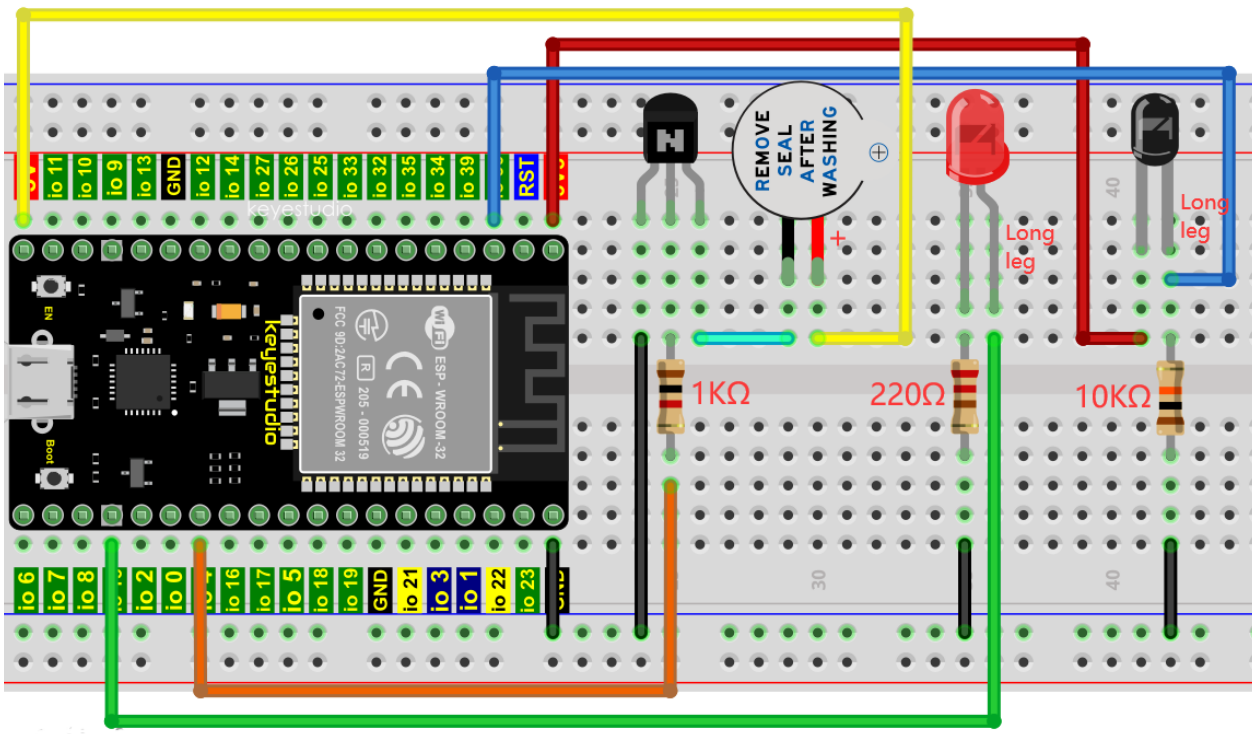
Compile and upload the code to ESP32, after the code is uploaded successfully, power up with a USB cable, open the serial monitor and set the baud rate to 115200. You will see that the serial monitor window will print out the ADC value, DAC value and voltage value of the flame sensor.

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.



5.20.5 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.



5.20.6 6.Test Code

```
value = get_value()
if value >500:
    buzzer.value(1)
```

Note the threshold of 500 in the code can be reset as required)

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

(continues on next page)

(continued from previous page)

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

5.20.7 7.Test 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.

5.21 Project 20Night Lamp

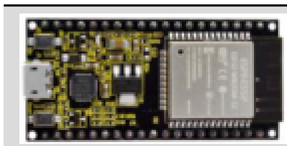
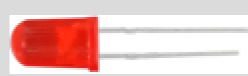
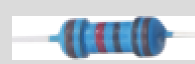
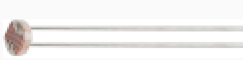
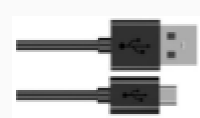
5.21.1 1.Introduction

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.

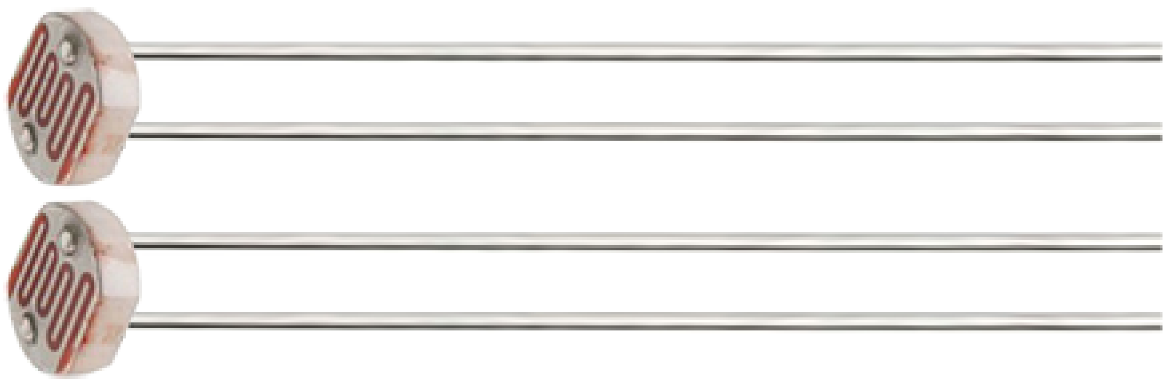
Why? In fact, this make use of a photosensitive element that senses the intensity of external ambient light. When the outdoor brightness decreases at night, the street lights will turn on automatically; In the daytime, the street lights will automatically turn off.

The principle of which is very simple, In this Project, we use a ESP32 to control a LED to achieve the effect of the street light.

5.21.2 2.Components

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

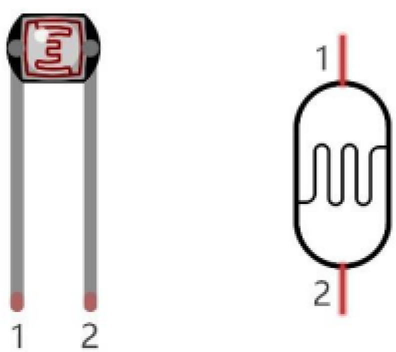
5.21.3 3.Component Knowledge



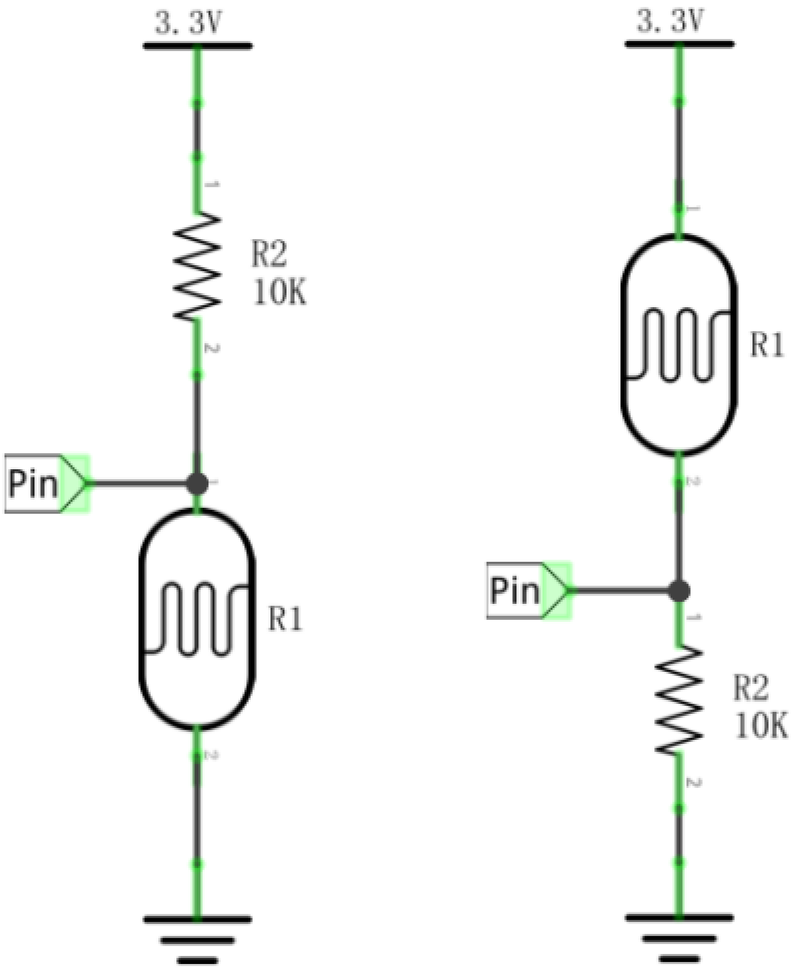
Photoresistor :

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

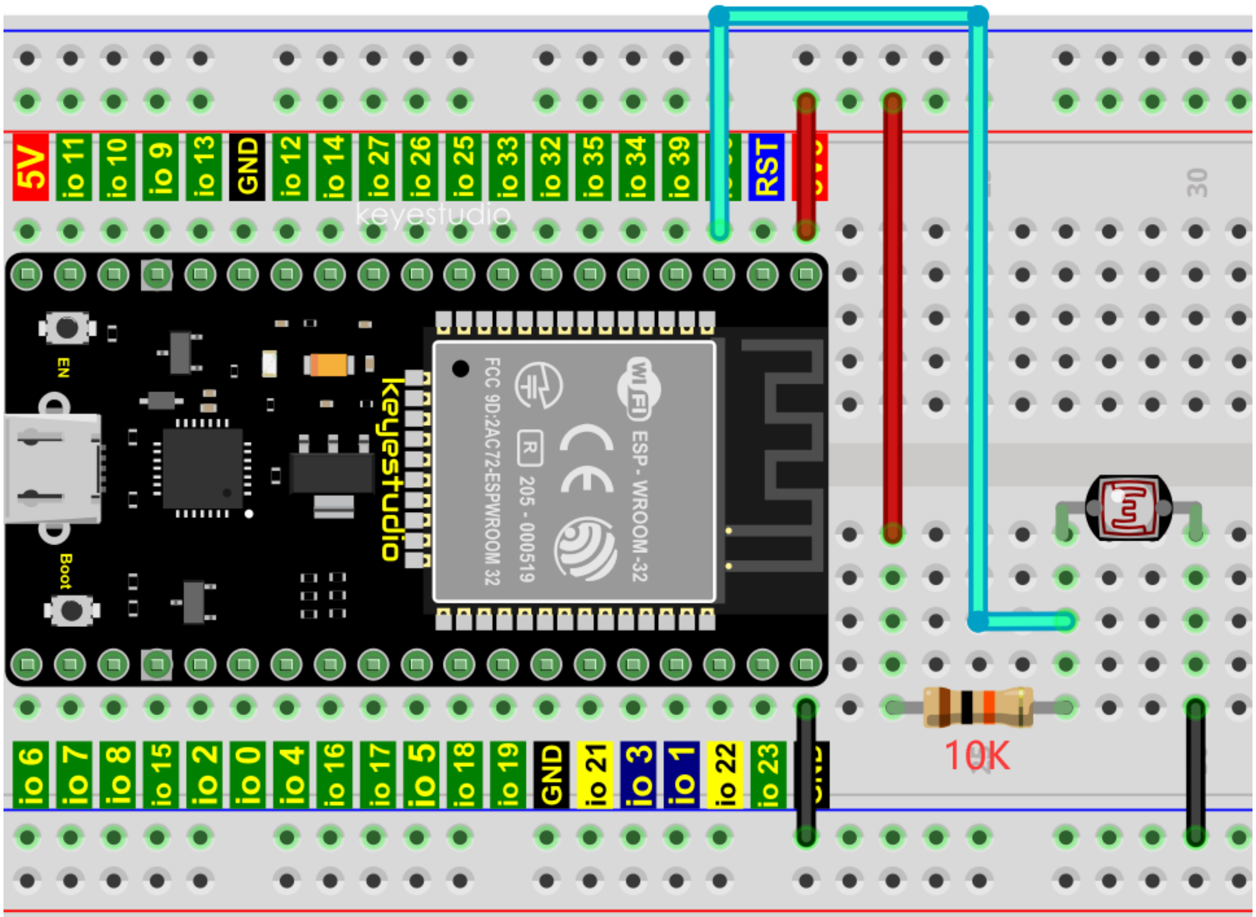


In the circuit above, when the resistance of the photoresistor changes due to the change of light intensity, the voltage between the photoresistor and resistance R2 will also change.

Thus, the intensity of light can be obtained by measuring this voltage.

5.21.4 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




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

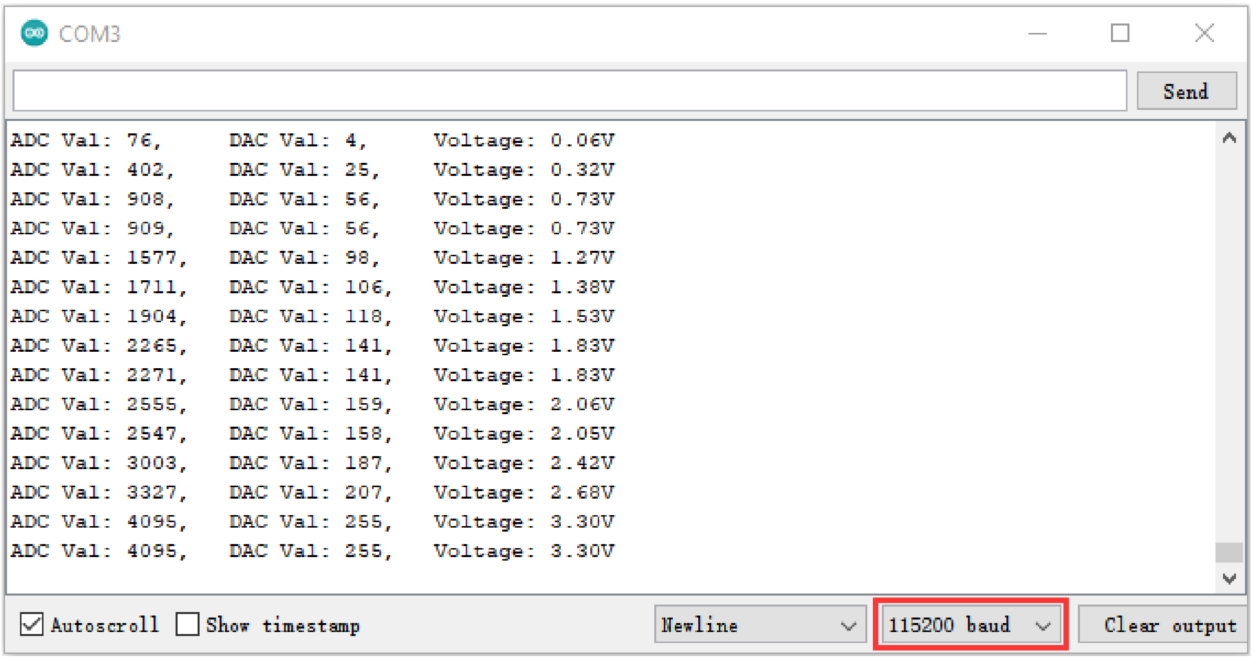
(continues on next page)

(continued from previous page)

```
  delay(200);
}
//*********************************************************************************
```

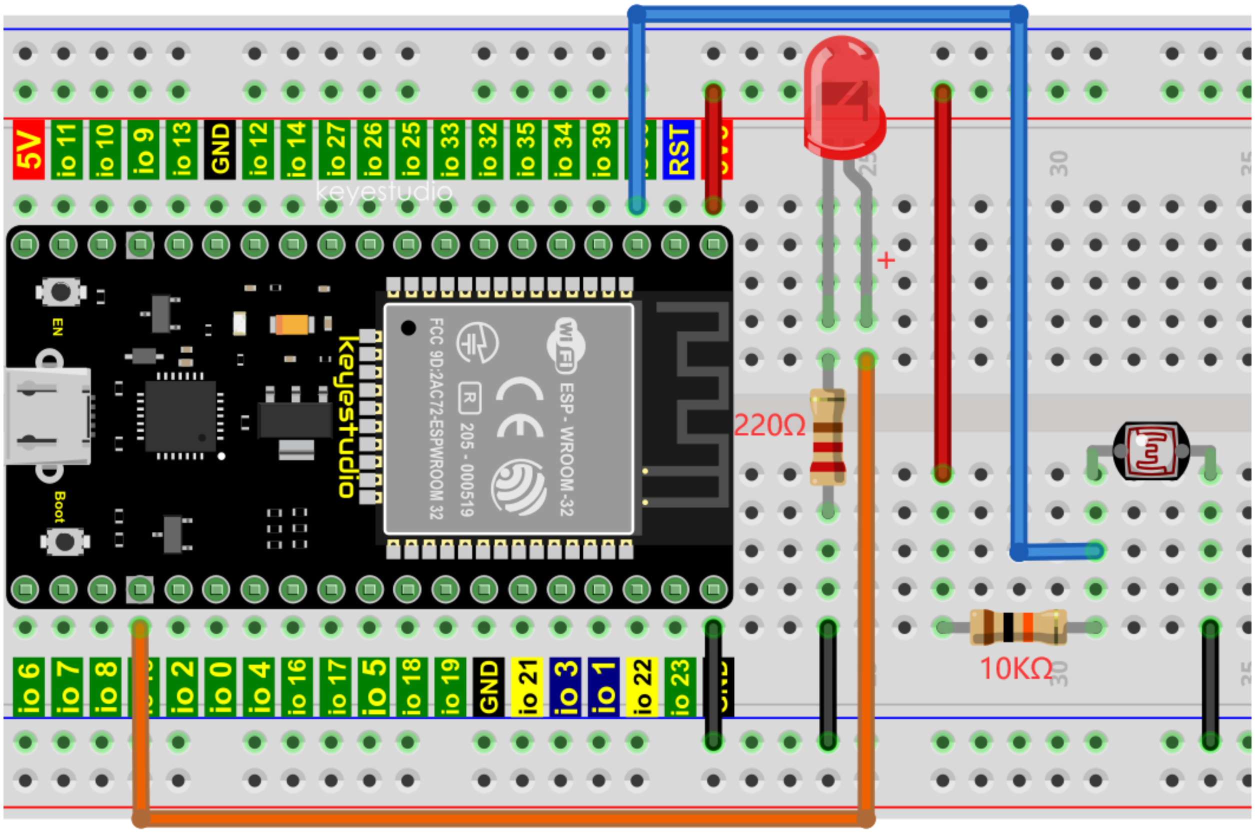
Compile and upload the code to ESP32, after the code is uploaded successfully, power up with a USB cable, open the serial monitor and set the baud rate to 115200.

You will see that the serial 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 decrease gradually.



5.21.5 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.



5.21.6 6.Test Code

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

(continues on next page)

(continued from previous page)

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

5.21.7 7.Test 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 contrary, the LED will be dim.

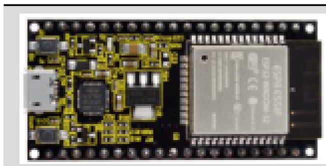
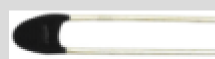
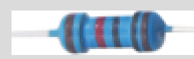
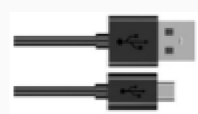
5.22 Project 21: Temperature Instrument

5.22.1 1.Introduction

Thermistor is a kind of resistor whose resistance depends on temperature changes, which is widely used in gardening, home alarm systems and other devices.

Therefore, we can use the features to make a temperature instrument.

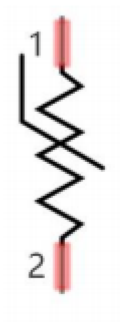
5.22.2 2.Components

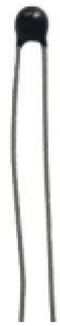
			
ESP32*1	Breadboard*1	Thermistor*1	10KResistor*1
			
M-F Dupont Wires	LCD 128X32 DOT*1	Jumper Wires	USB Cable*1

5.22.3 3.Component Knowledge

Thermistor: It 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 to detect temperature intensity. The thermistor and its electronic symbol are shown below:





The relationship between resistance and temperature of the 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;

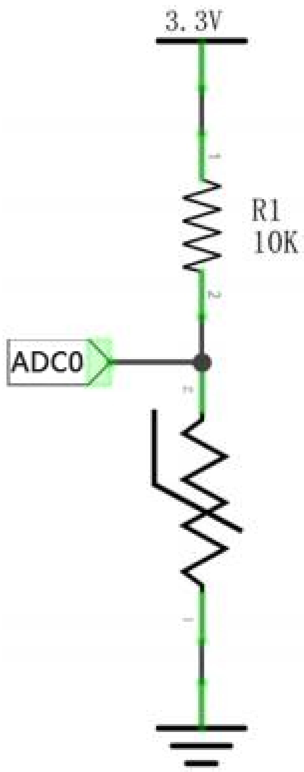
EXP[n] is nth power of e;

B is temperature index;

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

Parameters: B=3950, R=10k, T1=25.

The circuit connection method of the thermistor is similar to the photoresistor, as shown below



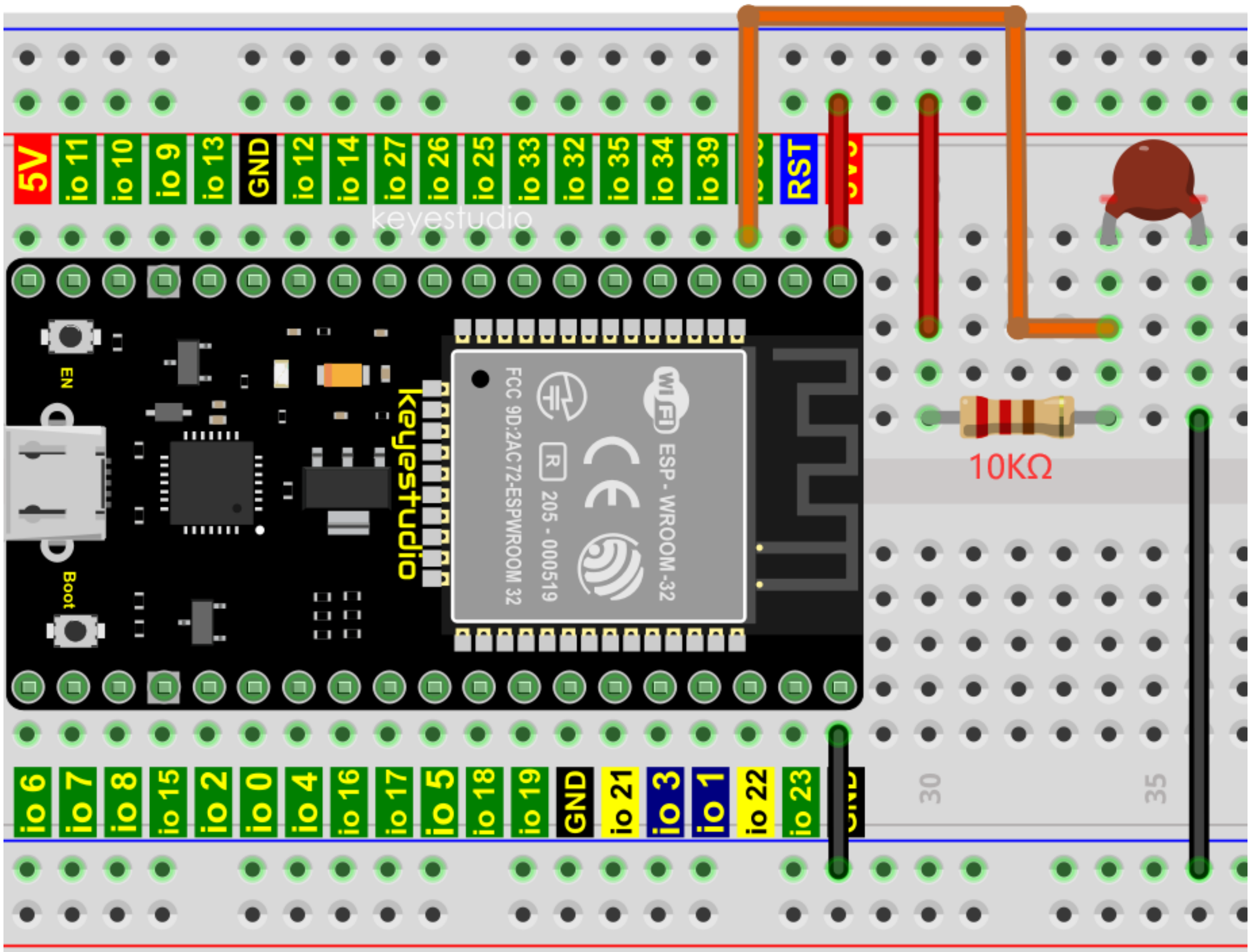
We can use the value measured by the ADC converter to obtain the resistance of thermistor, and then we can use the formula to obtain the temperature value.

Therefore, the temperature formula can be derived as:

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

5.22.4 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 wirings according to the wiring diagram below




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

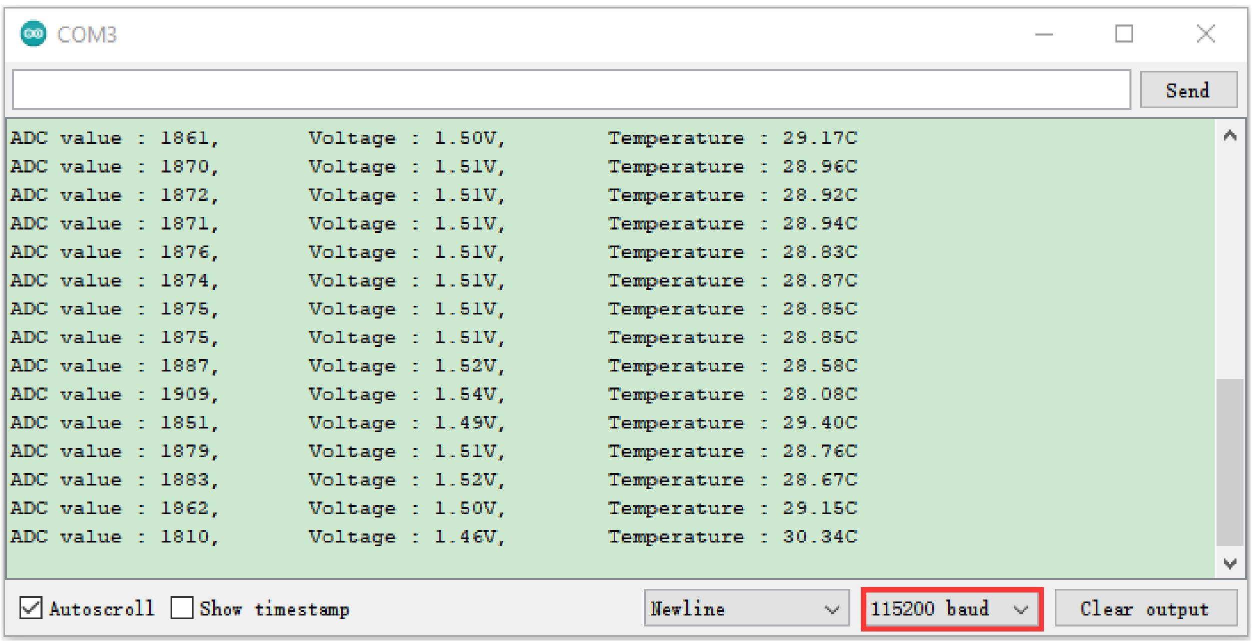
(continues on next page)

(continued from previous page)

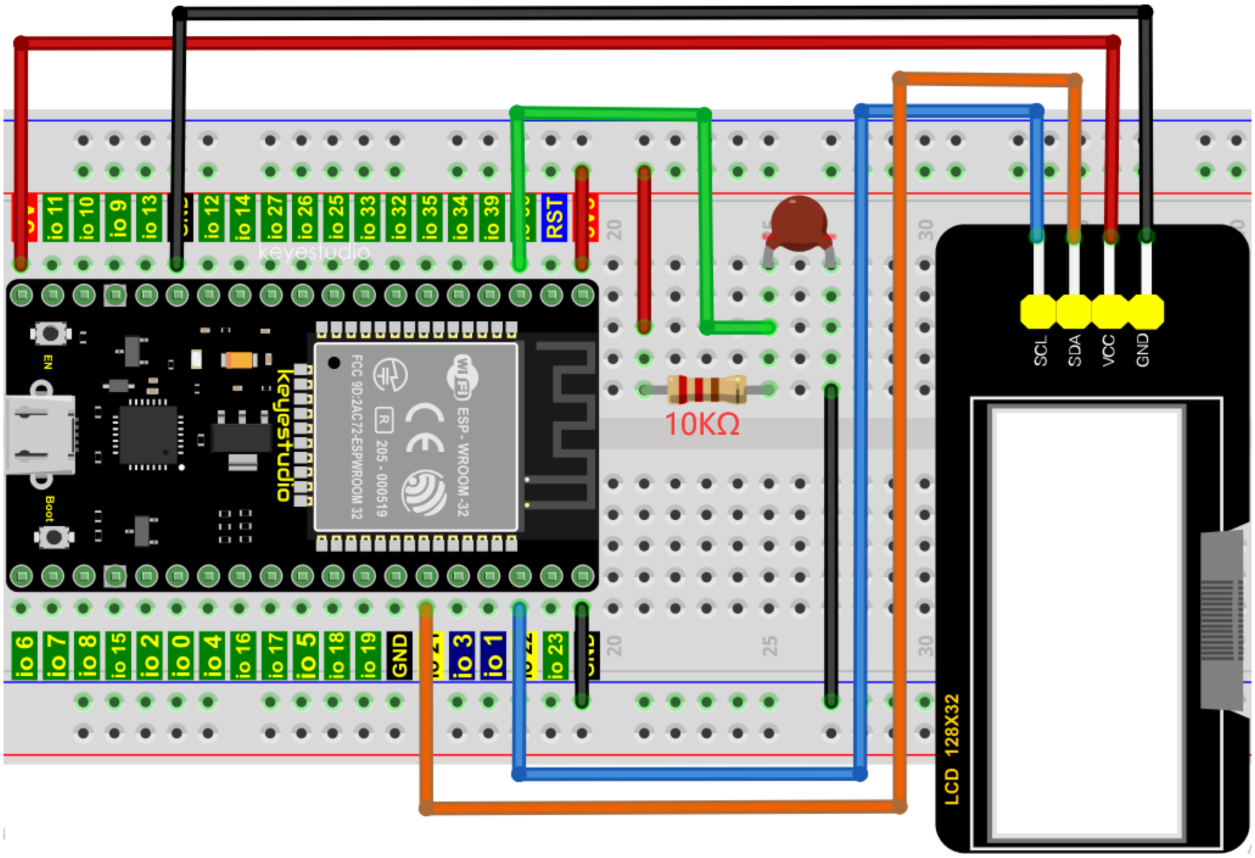
```
temperature (Kelvin)
  double tempC = tempK - 273.15;                              //calculate
temperature (Celsius)
  Serial.printf("ADC value : %d,\tVoltage : %.2fV, \tTemperature : %.2fC\n", adcValue,
voltage, tempC);
  delay(1000);
}
//*****************************************************************************
```

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 monitor 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 increasing.



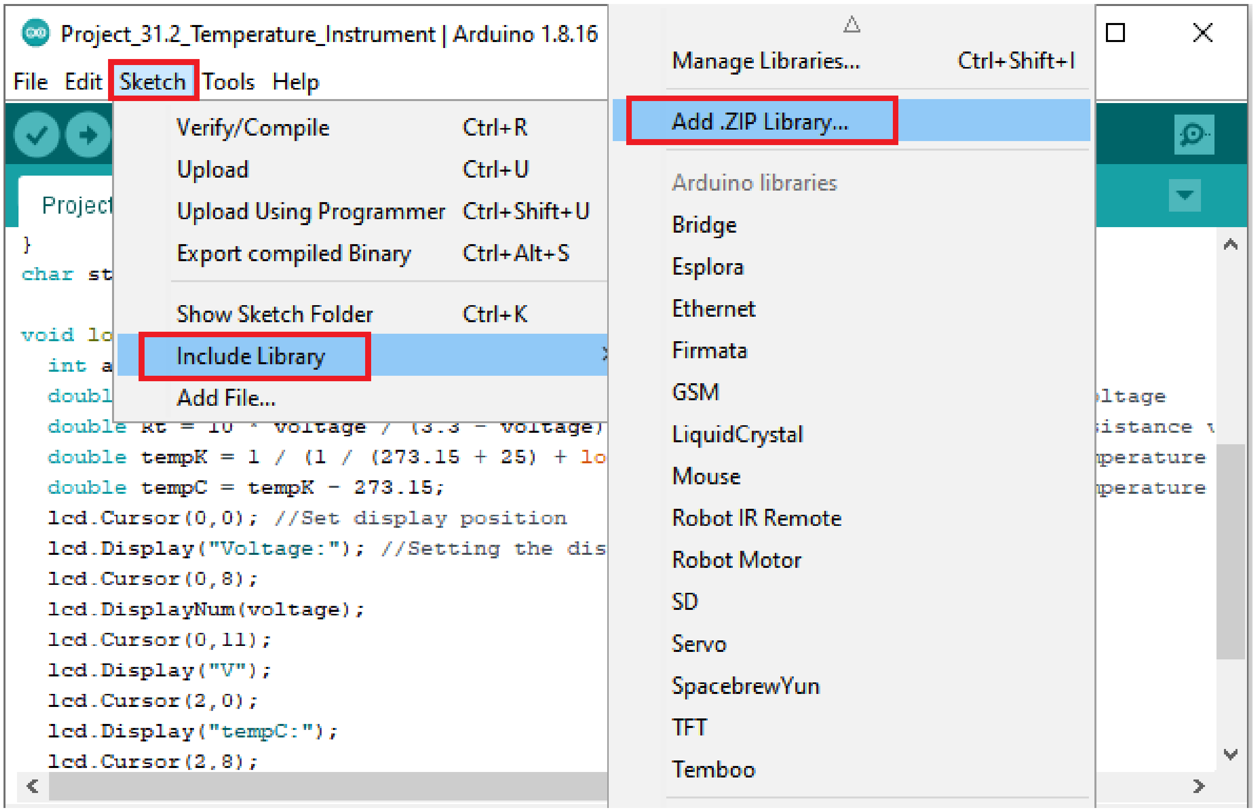
5.22.5 5.Wiring diagram of the temperature instrument

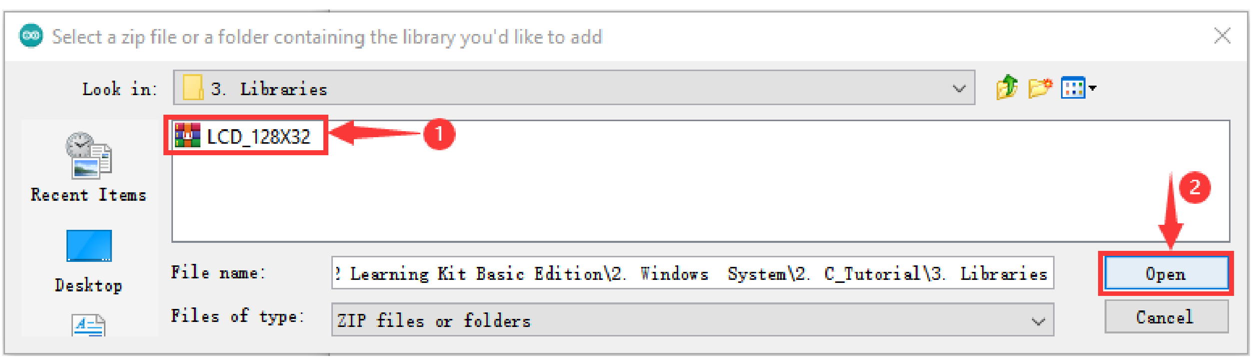


5.22.6 6.Adding the lcd128_32_io library

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\LCD_128X32.ZIP",** which locates in this directory. Select the **LCD_128X32.ZIP** file and then click "Open".





5.22.7 7.Test Code

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

(continues on next page)

(continued from previous page)

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

5.22.8 8.Test Result

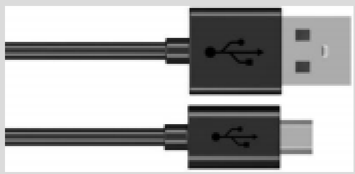
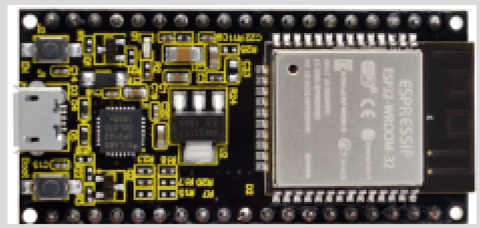
Compile and upload the code to ESP32, after the code is uploaded successfully, power up with a USB cable and you will see that the LCD 128X32 DOT displays the voltage value of the thermistor and the temperature value in the current environment.

5.23 Project 22Bluetooth

This chapter mainly introduces how to make simple data transmission through Bluetooth of ESP22 and mobile phones. Project 22.1 is classic Bluetooth while project 22.2 is Bluetooth control LED.

5.23.1 Project 22.1Classic Bluetooth

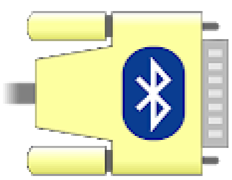
1.Components

	
USB Cable*1	ESP22*1

In this tutorial we need to use a Bluetooth APP called serial Bluetooth terminal to assist in the experiment.

Download link: https://www.appsapk.com/serial-Bluetooth-terminal/.

Here is its *sign*



2.Component Knowledge

Bluetooth is a short-distance communication system that can be divided into two types, namely low power Bluetooth (BLE) and classic Bluetooth. There are two modes for simple data transfer: master mode and slave mode.

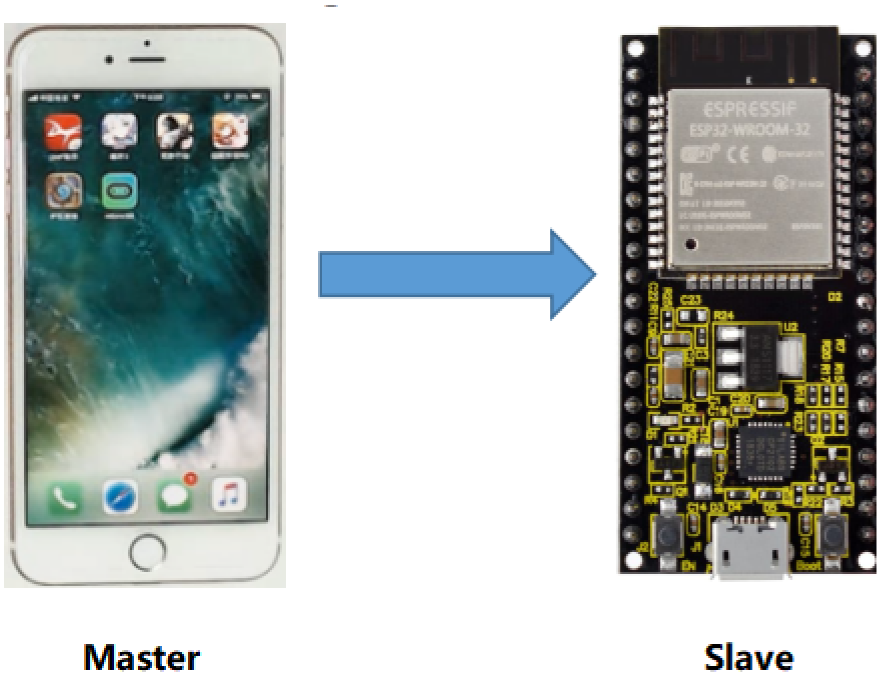
Master Mode:

In this mode, work is done on the master device and can be connected to the slave device. When the device initiates a connection request in the main mode, information such as the address and pairing password of other Bluetooth devices are required. Once paired, you can connect directly to them.

Slave Mode:

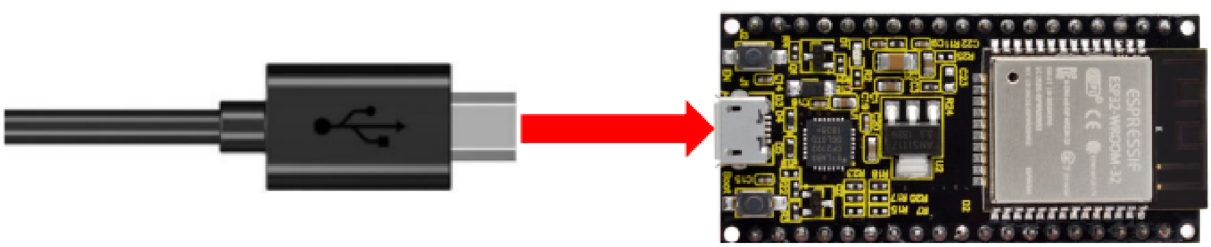
A Bluetooth module in the slave mode can only accept connection requests from the host, but cannot initiate connection requests. After being connected to a host device, it can send and receive data through the host device.

Bluetooth devices can interact with each other, when they interact, the Bluetooth device in the main mode searches for nearby devices. While a connection is established, they can exchange data. For example, when a mobile phone exchanges data with ESP22, the mobile phone is usually in master mode and the ESP22 is in slave mode.

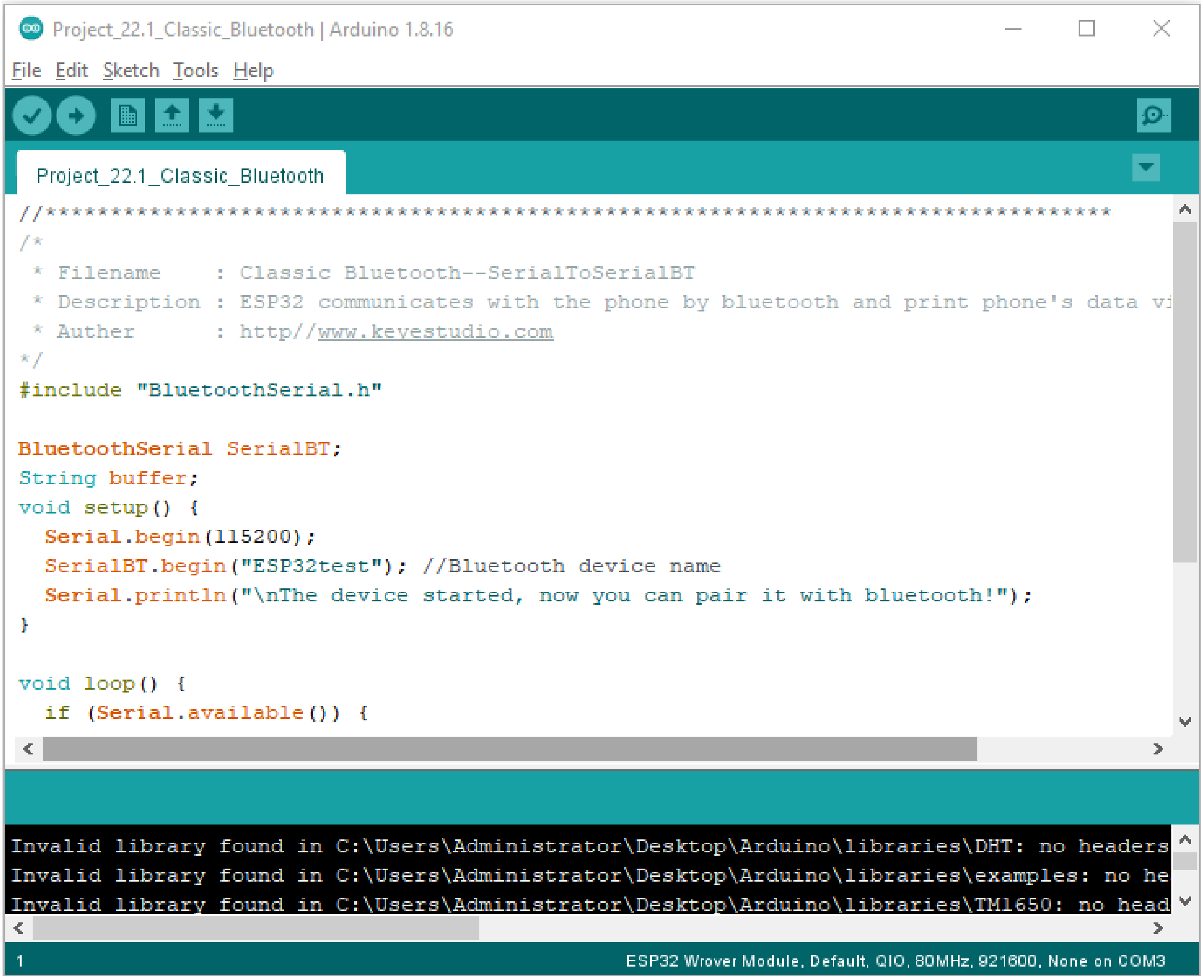


3.Wiring Diagram

We can use a USB cable to connect ESP22 mainboard to the USB port on a computer.



4.Test Code




```
//**********************************************************************************
/*
 * 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() {
```

(continues on next page)

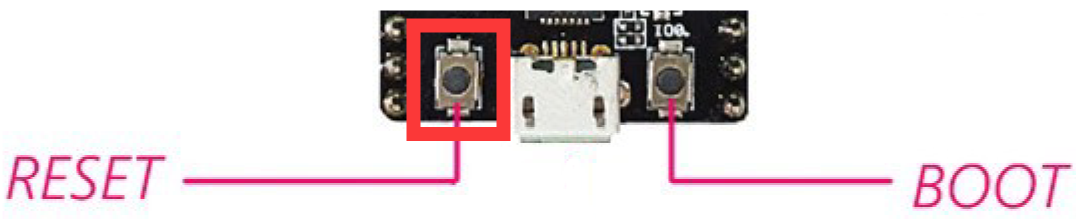
(continued from previous page)

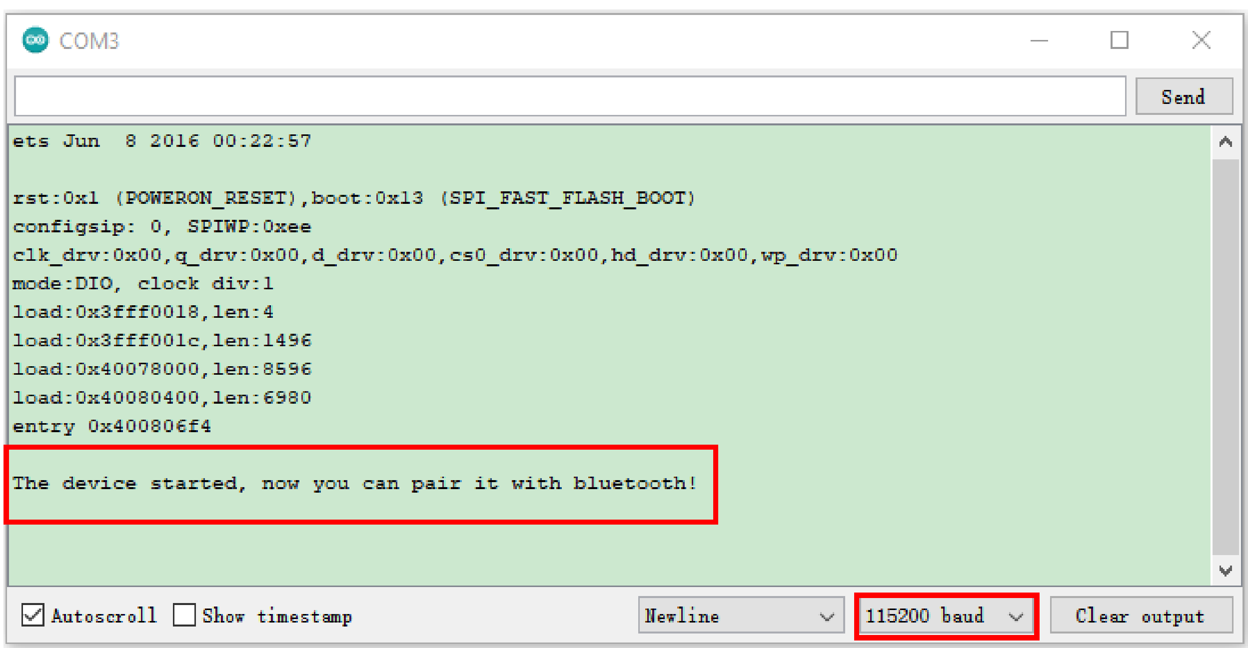
```
  if (Serial.available()) {
    SerialBT.write(Serial.read());
  }
  if (SerialBT.available()) {
    Serial.write(SerialBT.read());
  }
  delay(20);
}
//*********************************************************************************
```

6.Test Result

Compile and upload the code to the ESP22. After uploading successfully, we will use a USB cable to power on. Open the serial monitor and set the baud rate to **115200**.

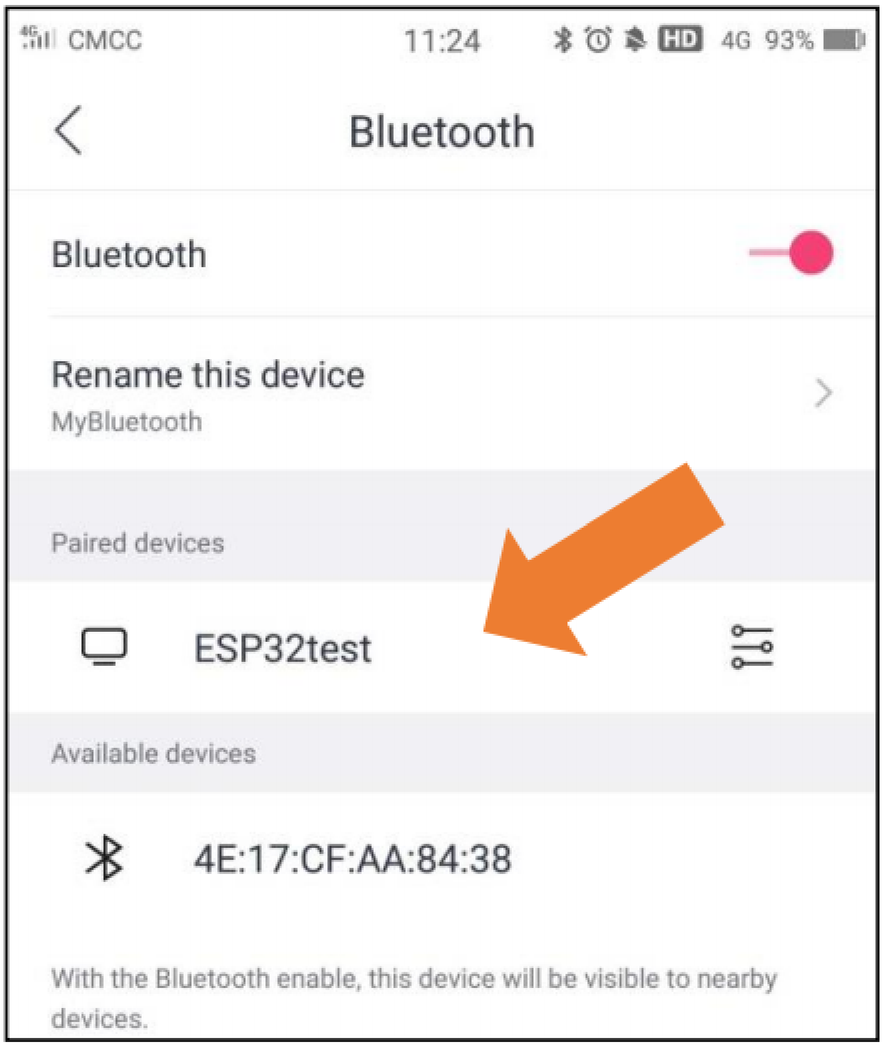
When you see the serial monitor prints out the character string as below, it indicates that the Bluetooth of ESP22 is ready and waiting for connection with a phone. (If open the serial monitor and set the baud rate to 115200, the information is not displayed, please press the RESET button of the ESP22)



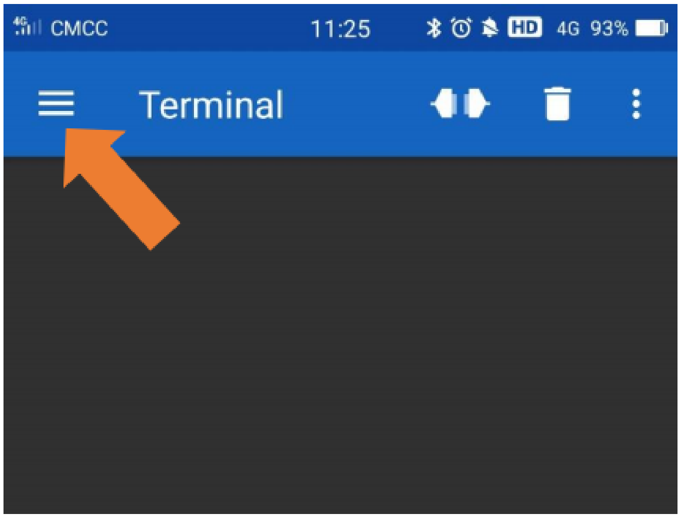


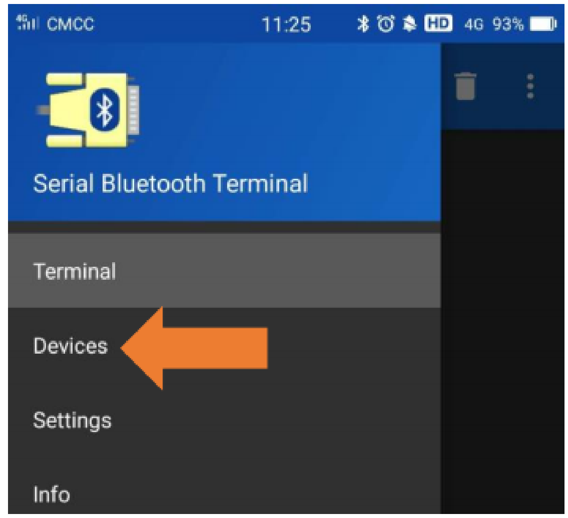
Make sure that the Bluetooth of your phone has been turned on and “Serial Bluetooth Terminal” has been installed.

Click"Search"search for the nearby Bluetooth and select to connect the"ESP22 test".

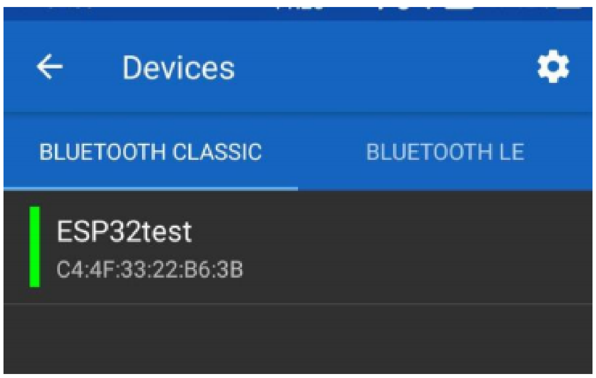


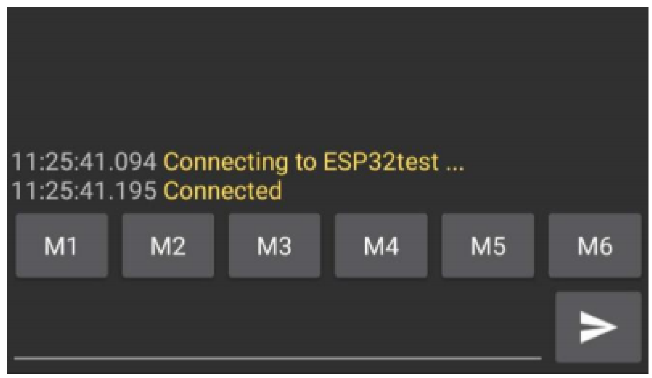
Turn on software APP, click the left of the terminal. Select "**Devices**" .





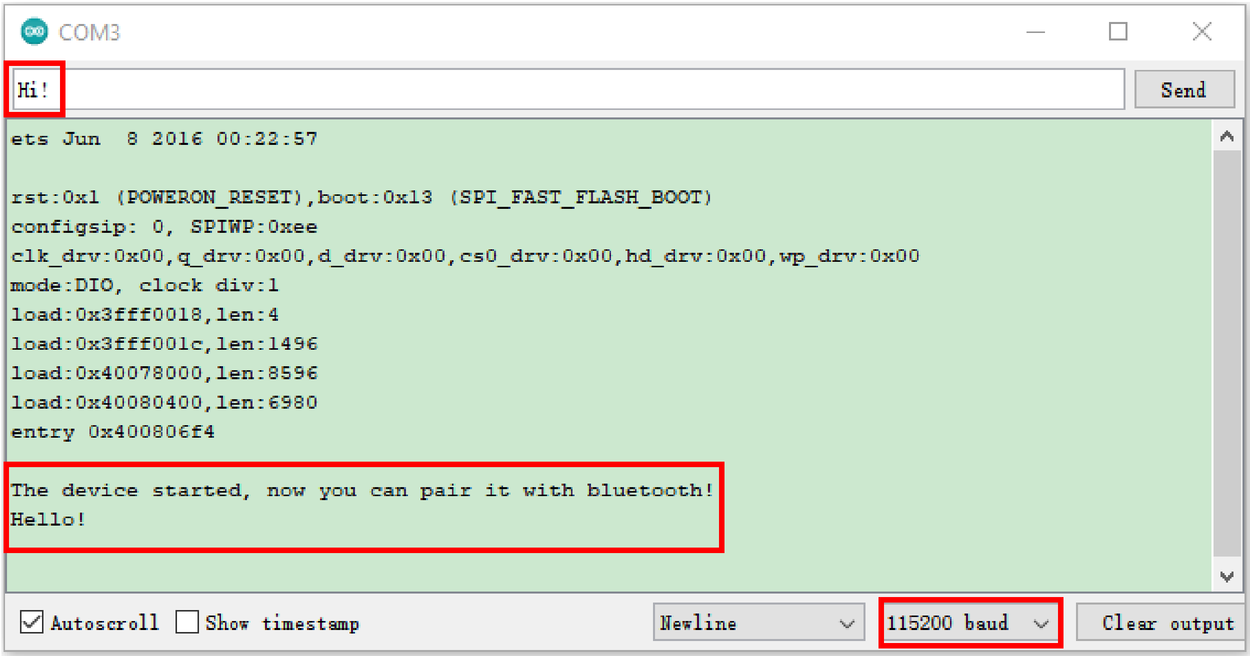
Select ESP22test in classic Bluetooth mode, and a successful connecting prompt will appear as shown below.

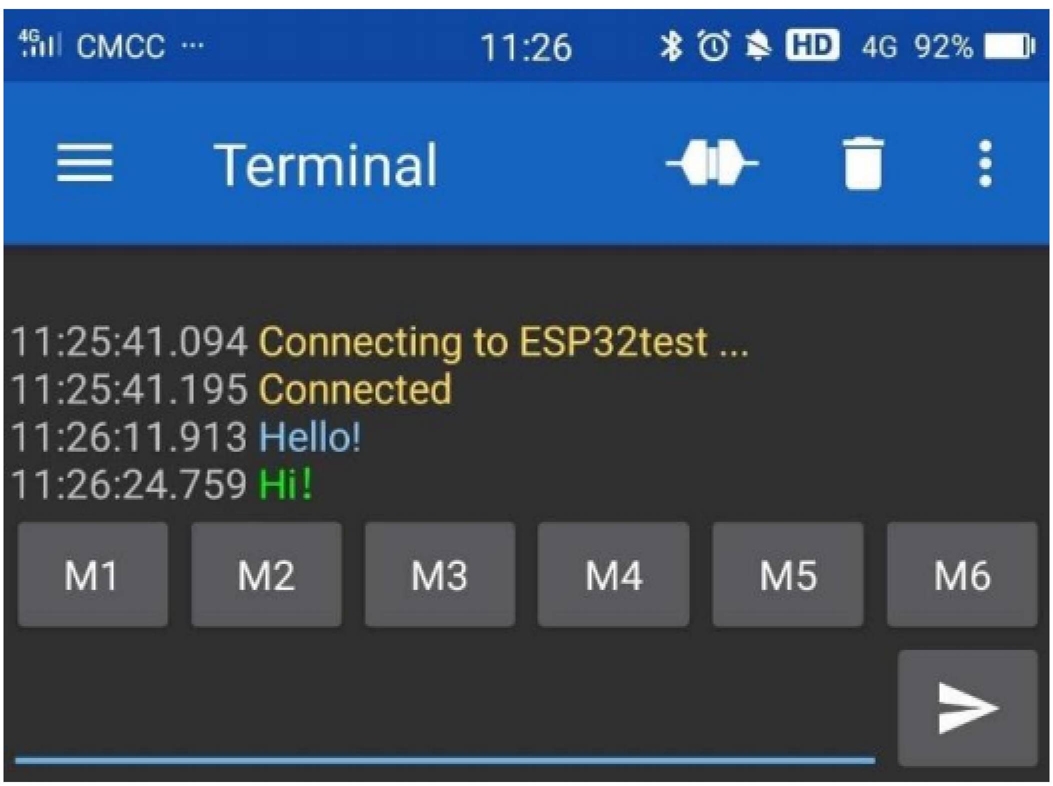




Data can be transferred between your phone and a computer via ESP22 now.

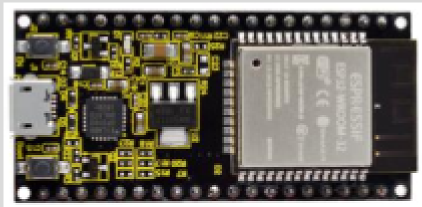
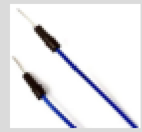
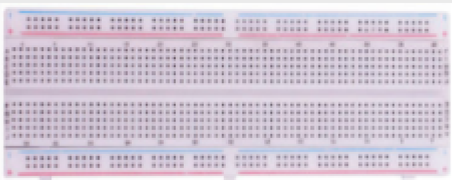
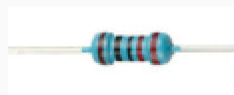
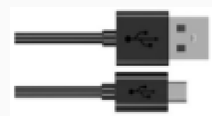
Send "Hello", When the computer receives it, which will reply with "Hi!".


COM3
Hi!
Send
ets Jun 8 2016 00:22:57
rst:0x1 (POWERON_RESET),boot:0x13 (SPI_FAST_FLASH_BOOT)
configsip: 0, SPIWP:0xee
clk_drv:0x00,q_drv:0x00,d_drv:0x00,cs0_drv:0x00,hd_drv:0x00,wp_drv:0x00
mode:DIO, clock div:1
load:0x3fff0018,len:4
load:0x3fff001c,len:1496
load:0x40078000,len:8596
load:0x40080400,len:6980
entry 0x400806f4
The device started, now you can pair it with bluetooth!
Hello!
Autoscroll
Show timestamp
Newline
115200 baud
Clear output

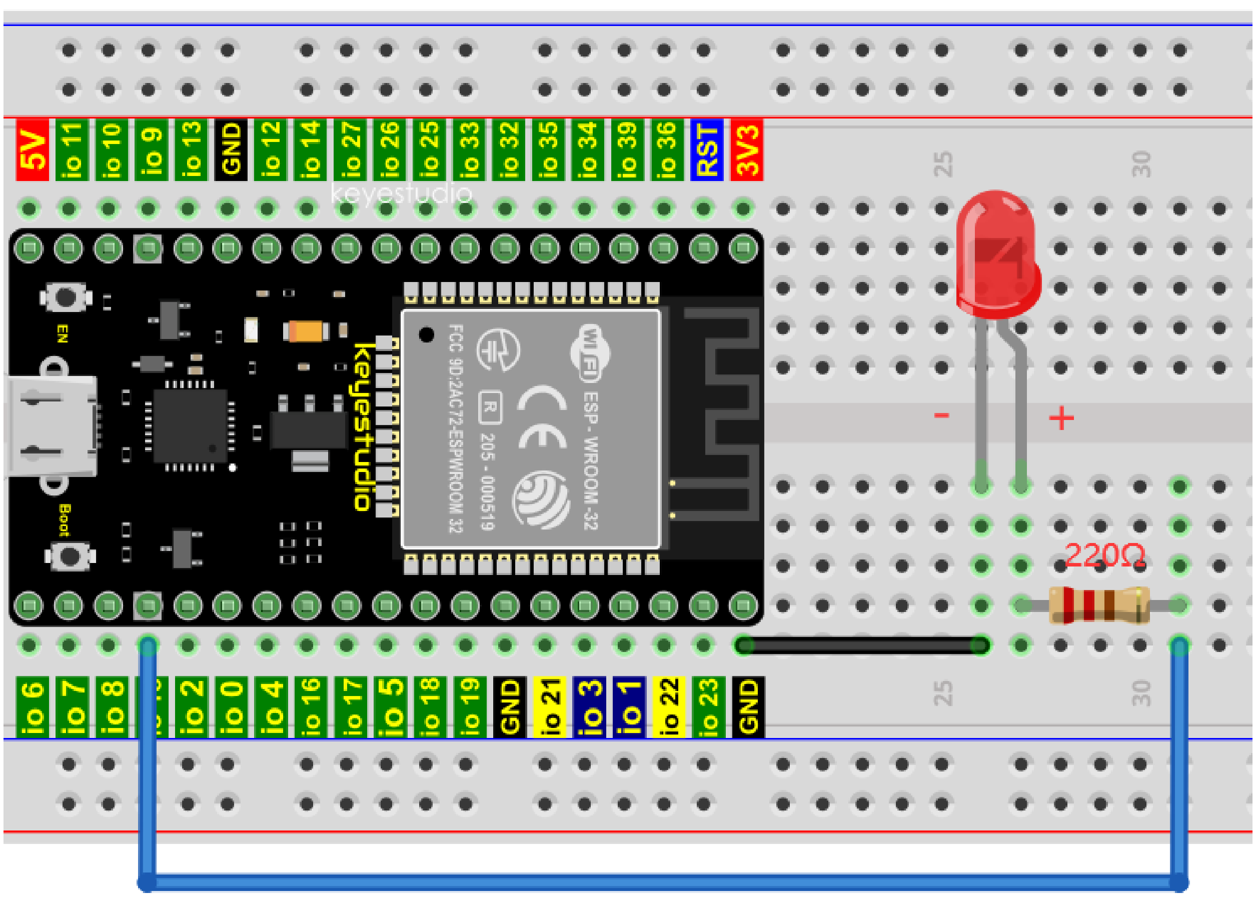

CMCC
11:26
HD
4G 92%
Terminal
11:25:41.094 Connecting to ESP32test ...
11:25:41.195 Connected
11:26:11.913 Hello!
11:26:24.759 Hi!
M1
M2
M3
M4
M5
M6

5.23.2 Project 22.2Bluetooth Control LED

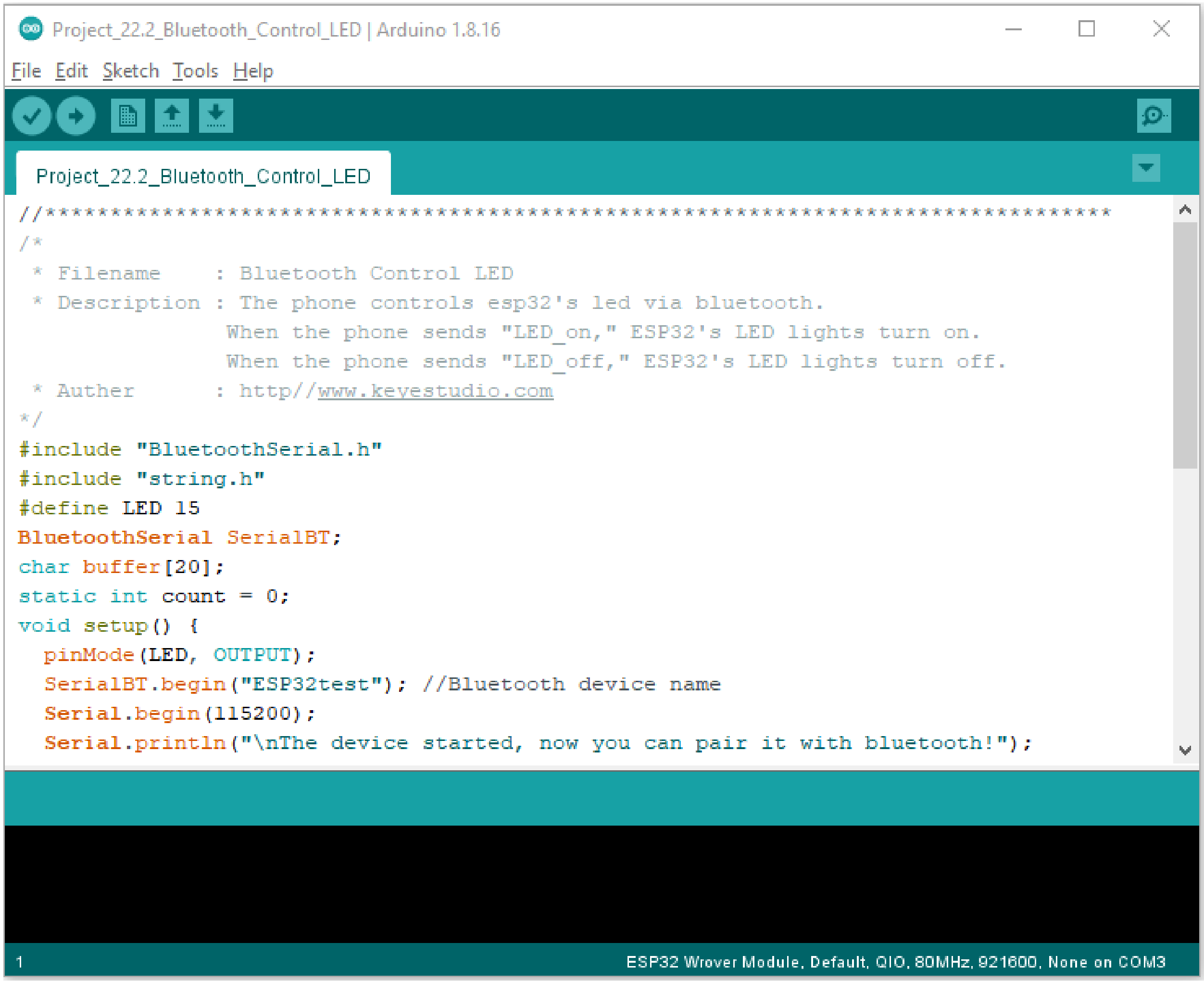
1.Components

		
ESP22*1	Red LED*1	Jumper Wires
		
Breadboard*1	220Resistor*1	USB Cable*1

2.Wiring Diagram



3.Test Code




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

(continues on next page)

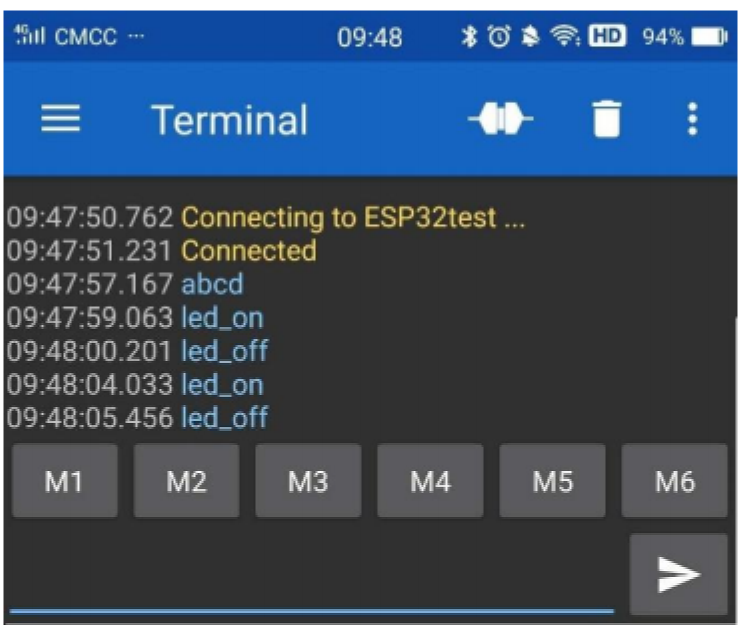
(continued from previous page)

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

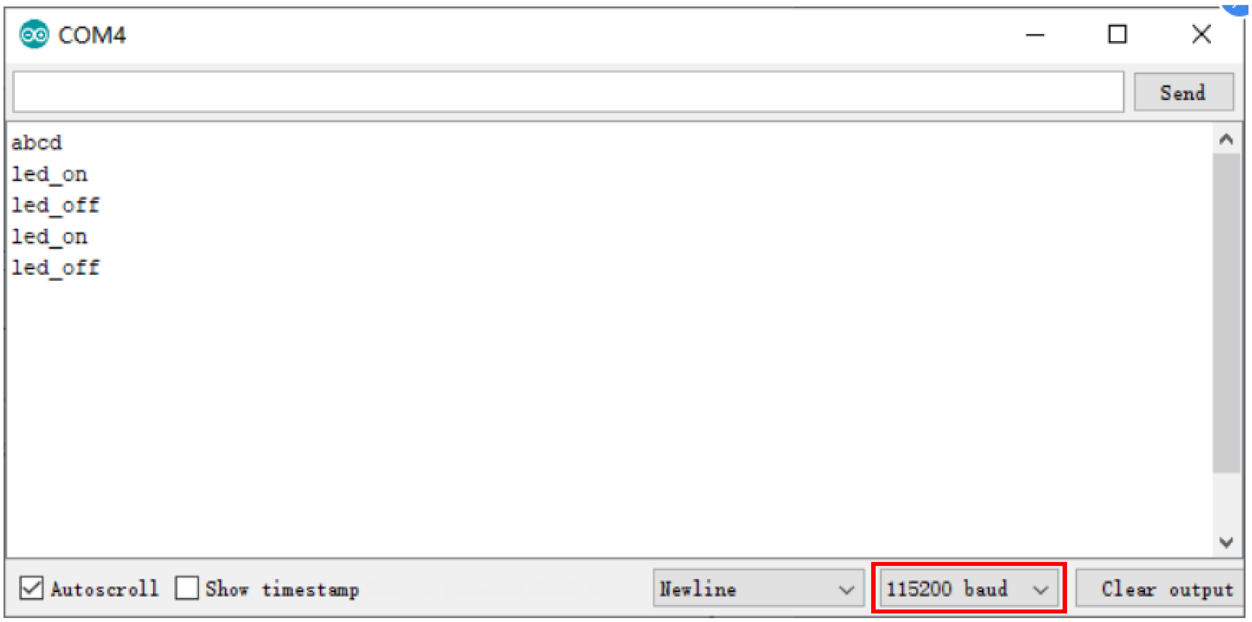
4.Test Result

Compile and upload the code to the ESP22. After uploading successfully, we will use a USB cable to power on. The APP operation is the same as the project 22.1. To make the external LED on and off, simply change the sending content to "led_on" and "led_off".

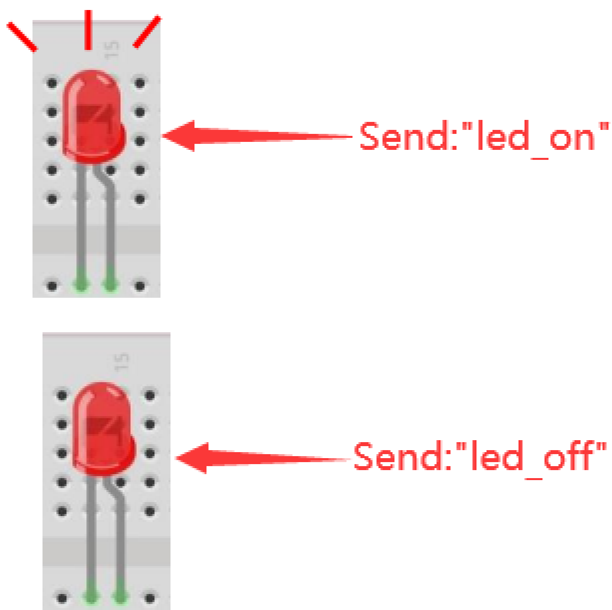
Moving the APP to send data:



The serial monitor will display as follows:



LED Circumstance



Note: If the sent content is not "led-on 'or" led-off ", the status of the LED will not change. If the LED is on, it remains on when irrelevant content is received; Conversely, if the LED is off, it continues to be off when irrelevant content is received.

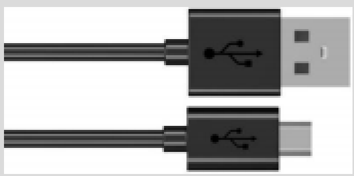
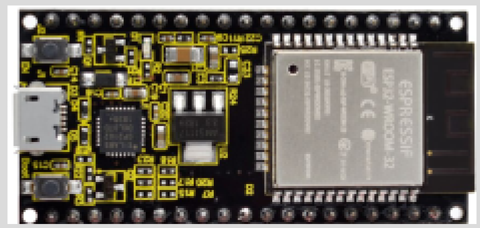
5.24 Project 23WiFi Station Mode

5.24.1 1.Introduction

ESP32 has three different WiFi operating modes: Station mode, AP mode and AP+Station mode. All WiFi programming projects must be configured with WiFi operating mode before using, otherwise WiFi cannot be used.

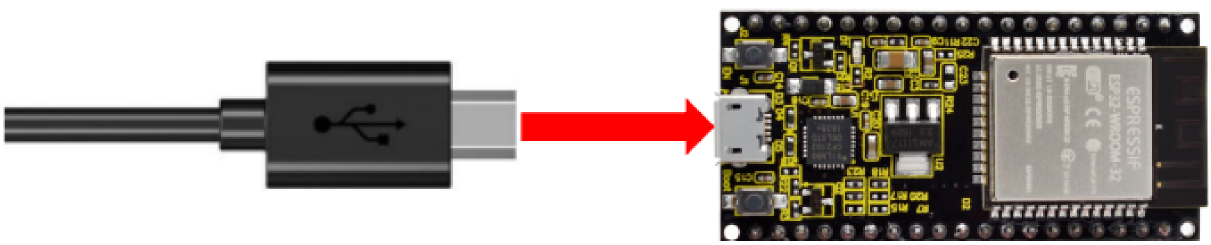
In this project, we are going to learn the WiFi Station mode of the ESP32.

5.24.2 2.Components

	
USB Cable*1	ESP32*1

5.24.3 3.Wiring Diagram

Plug the ESP32 to the USB port of your PC

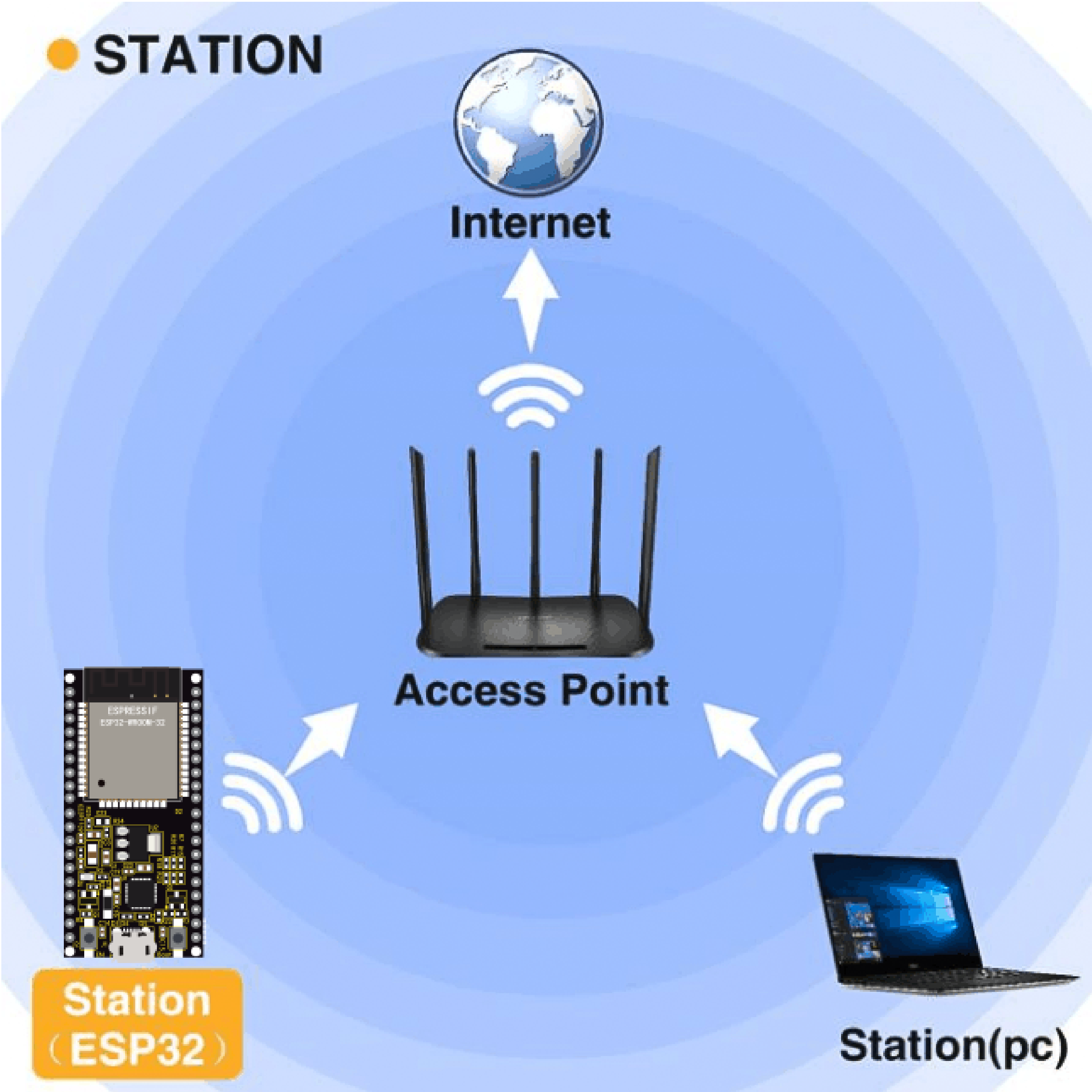


5.24.4 4.Component Knowledge

Station mode

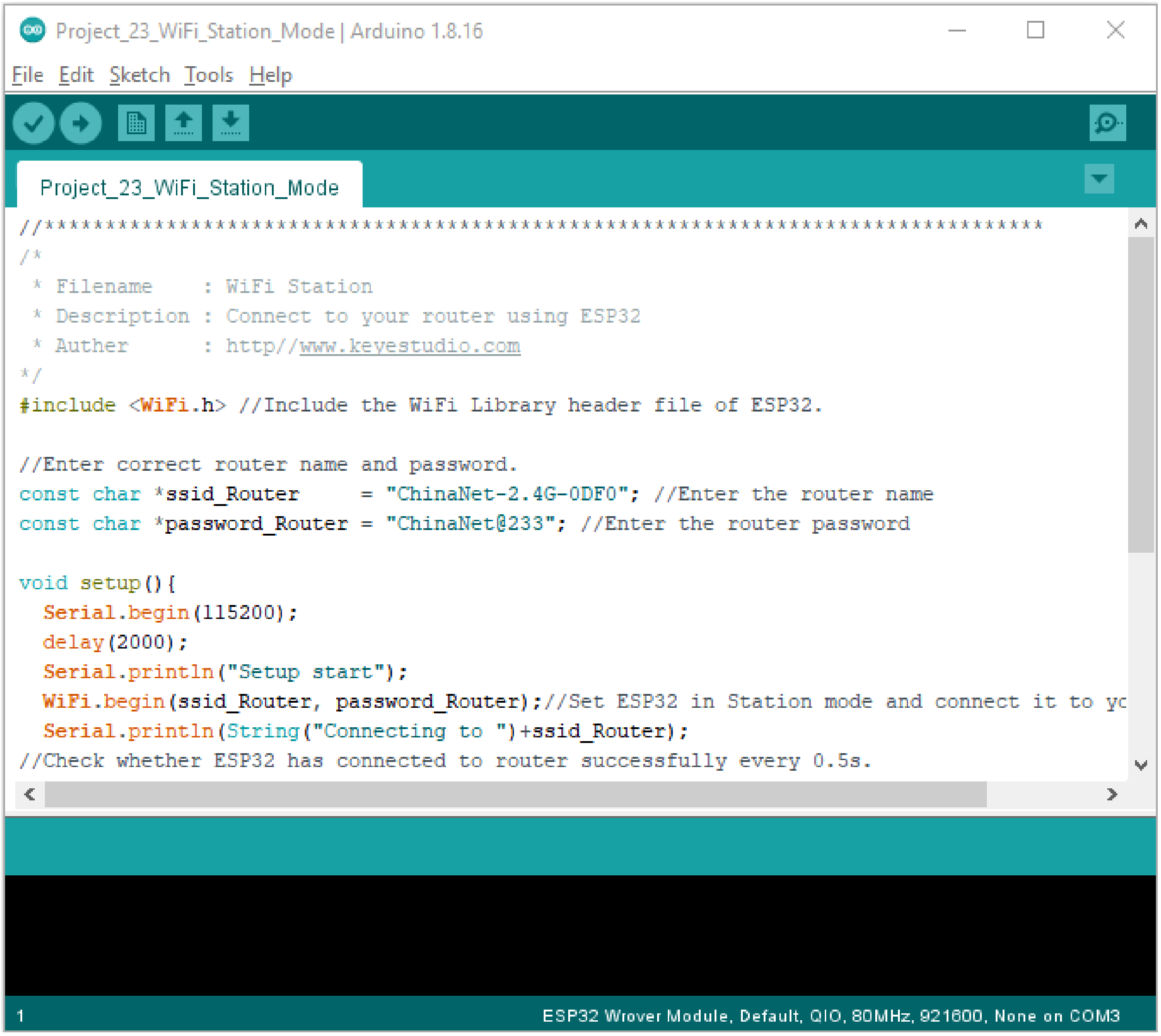
When setting Station mode, the ESP32 is taken as a WiFi client. It can connect to the router network and communicate with other devices on the router via a WiFi connection.

As shown in the figure below, the PC and the router have been connected. If the ESP32 wants to communicate with the PC, the PC and the router need to be connected.

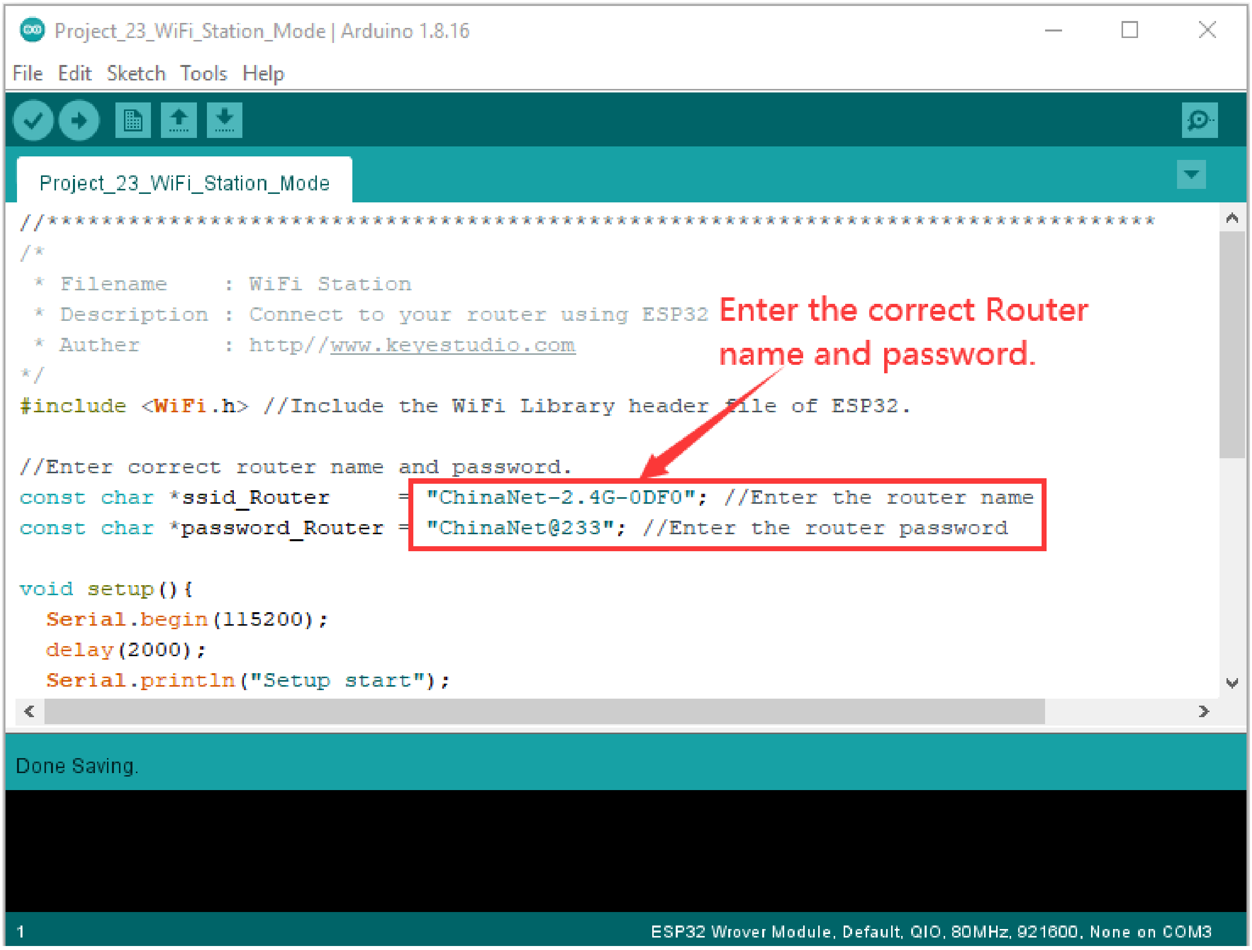


STATION
Internet
Access Point
Station
（ESP32）
Station(pc)

5.24.5 5.Test Code



Since WiFi names and passwords vary from place to place, thereby users need to enter the correct WiFi names and passwords in the box shown below before the program code runs.




```
//*************************************************************************************
/*
 * 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);
    Serial.print(".");
```

(continues on next page)

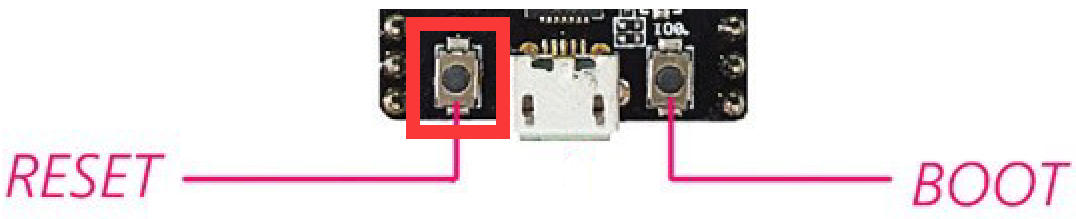
(continued from previous page)

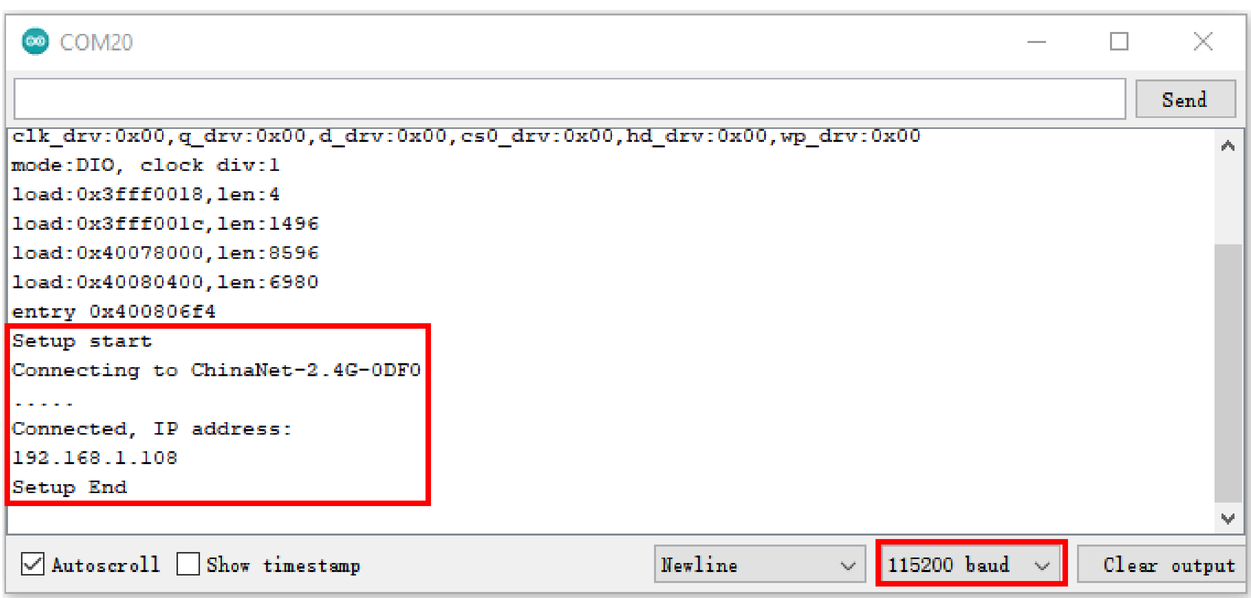
```
  }
  Serial.println("\nConnected, IP address: ");
  Serial.println(WiFi.localIP());//Serial monitor prints out the IP address assigned to␣
↪ESP32.
  Serial.println("Setup End");
}

void loop() {
}
//****************************************************************************
```

5.24.6 6.Test 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, then monitor 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)



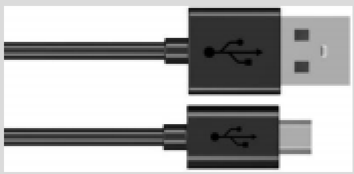
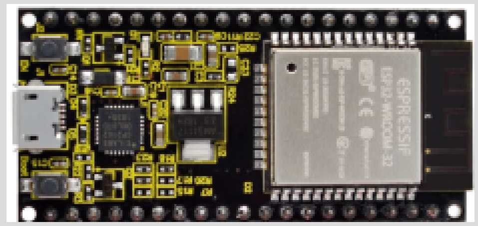


5.25 Project 24WiFi AP Mode

5.25.1 1.Introduction

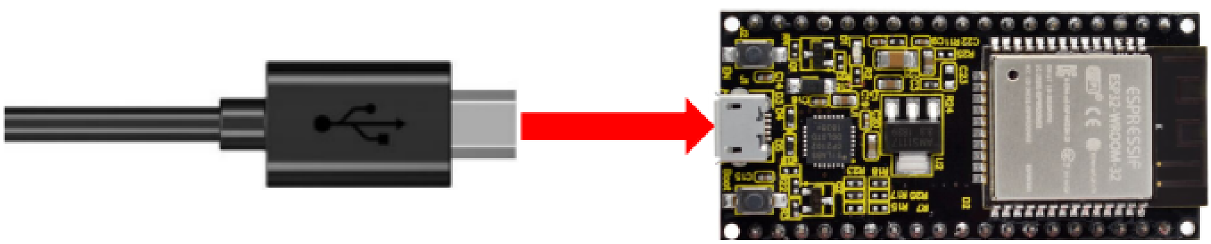
In this project, we are going to learn the WiFi AP mode of the ESP32.

5.25.2 2.Components

	
USB Cable*1	ESP32*1

5.25.3 3.Wiring Diagram

Plug the ESP32 mainboard to the USB port of your PC



5.25.4 4.Component Knowledge

AP Mode:

When setting AP mode, a hotspot network will be created, waiting for other WiFi devices to connect. As shown below;

Take the ESP32 as the hotspot, if a phone or PC needs to communicate with the ESP32, it must be connected to the ESP32's hotspot.

Communication is only possible after a connection is established via the ESP32.


AP
AP
（ESP32）
Station
(mobile phone)
Station
(pc)

5.25.5 5.Test Code

Before running the code , you can make any changes to the ESP32 AP name and password in the box as shown below, but in a default circumstance, it doesn't need to modify.


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

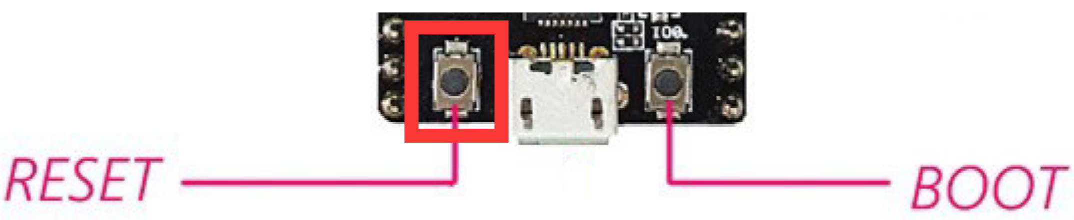
(continued from previous page)

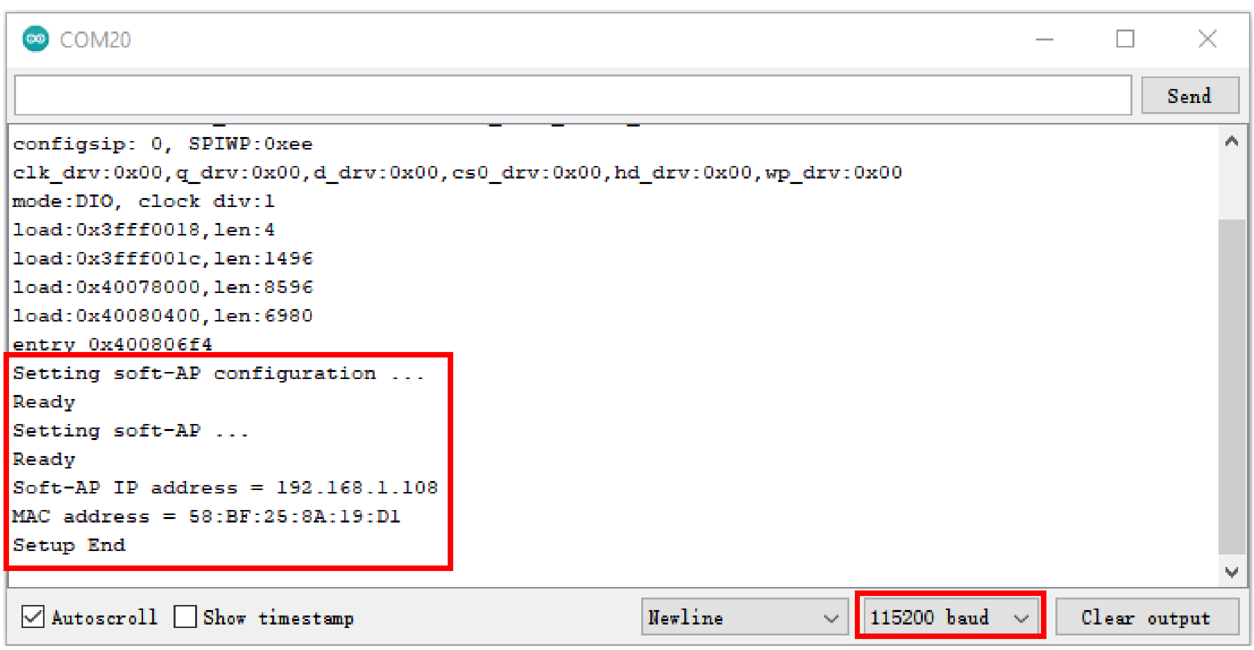
```
  Serial.println("Setting soft-AP ... ");
  boolean result = WiFi.softAP(ssid_AP, password_AP);
  if(result){
    Serial.println("Ready");
    Serial.println(String("Soft-AP IP address = ") + WiFi.softAPIP().toString());
    Serial.println(String("MAC address = ") + WiFi.softAPmacAddress().c_str());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

void loop() {
}
//*************************************************************************************
```

5.25.6 6.Test Result

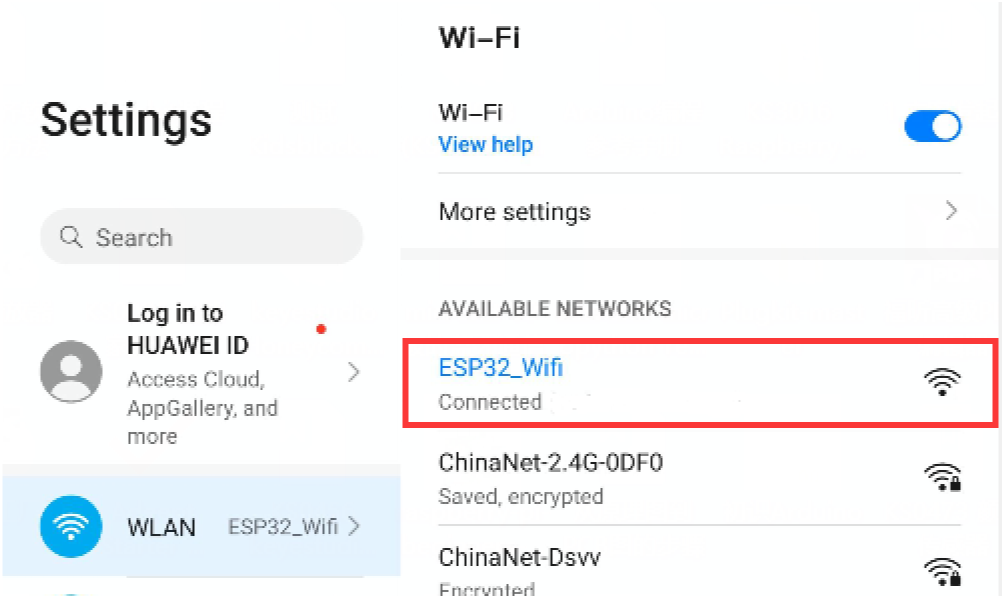
Enter the ESP32 AP name and password correctly, compile and upload the code to ESP32, open the serial monitor and set the baud rate to **115200**, then monitor will display as follows:

(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 printed information of the serial monitor, turn on the WiFi scanning function of the mobile phone, you can see the ssid_AP on ESP32, which is dubbed "ESP32_Wifi" in this code. You can connect to it either by typing the password "12245678" or by modifying the code to change its AP name and password.

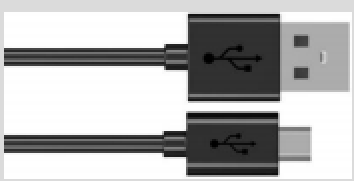
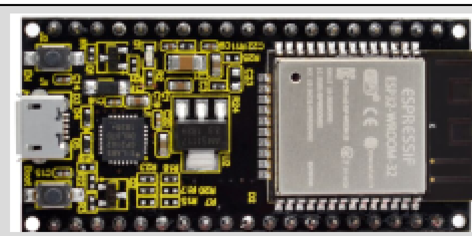


5.26 Project 25WiFi Station+AP Mode

5.26.1 1.Introduction

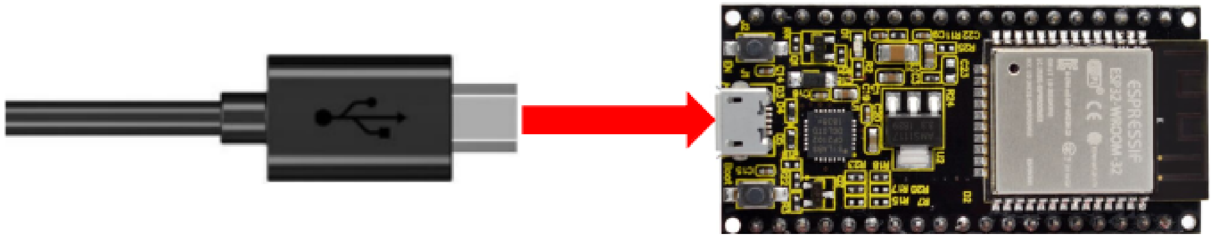
In this project, we are going to learn the AP+Station mode of the ESP32.

5.26.2 2.Components

	
USB Cable*1	ESP32*1

5.26.3 3.Wiring Diagram

Plug the ESP32 mainboard to the USB port of your PC



5.26.4 4.Component Knowledge

AP+Station mode:

In addition to the AP mode and the Station mode, **AP+Station mode** can be used at the same time. Turn on the Station mode of the ESP32, connect it to the router network, and it can communicate with the Internet through the router. Then turn on the AP mode to create a hotspot network.

Other WiFi devices can be connected to the router network or the hotspot network to communicate with the ESP32.

5.26.5 5.Test Code

Before running the code, you need to modify the ssid_Router, password_Router, ssid_AP and password_AP, as shown in the box below:


```
//*************************************************************************************
/*
 * 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");
    Serial.println(String("Soft-AP IP address = ") + WiFi.softAPIP().toString());
```

(continues on next page)

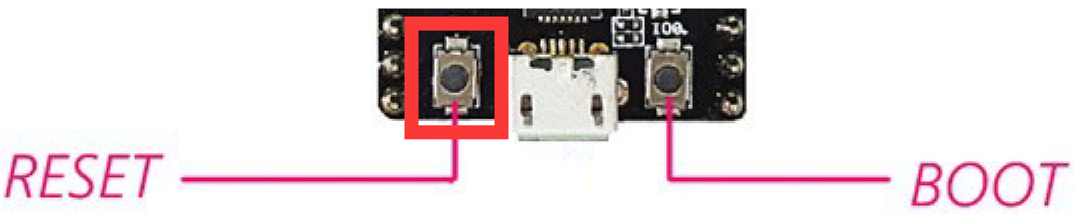
(continued from previous page)

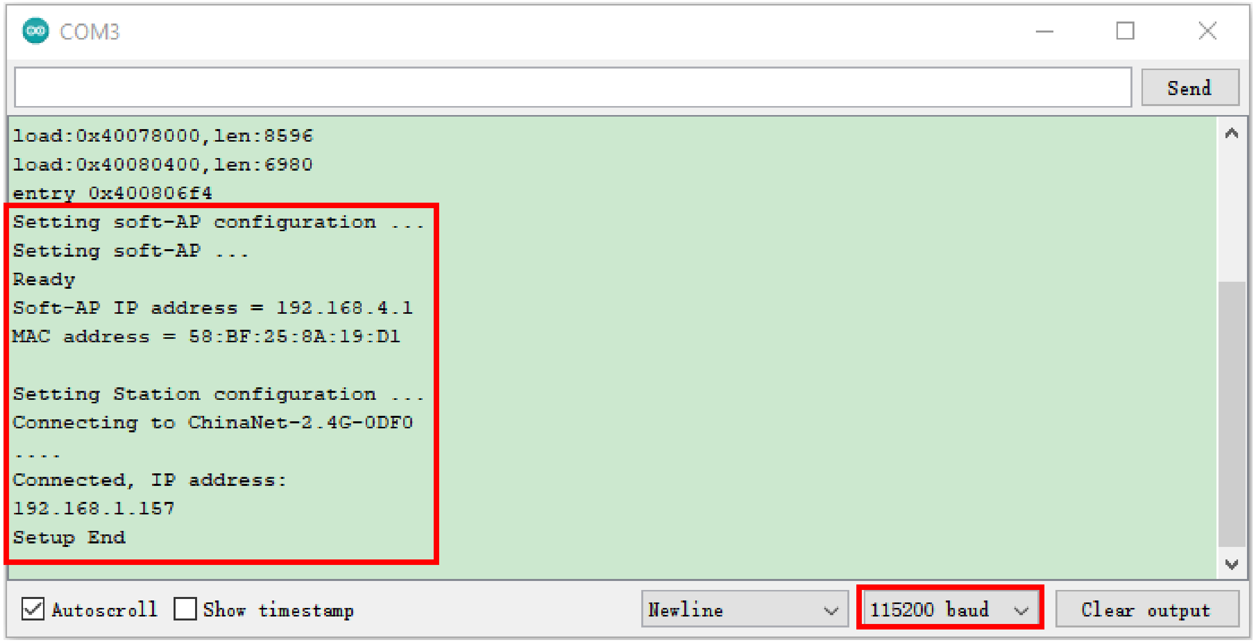
```
    Serial.println(String("MAC address = ") + WiFi.softAPmacAddress().c_str());
  }else{
    Serial.println("Failed!");
  }

  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.Test Result

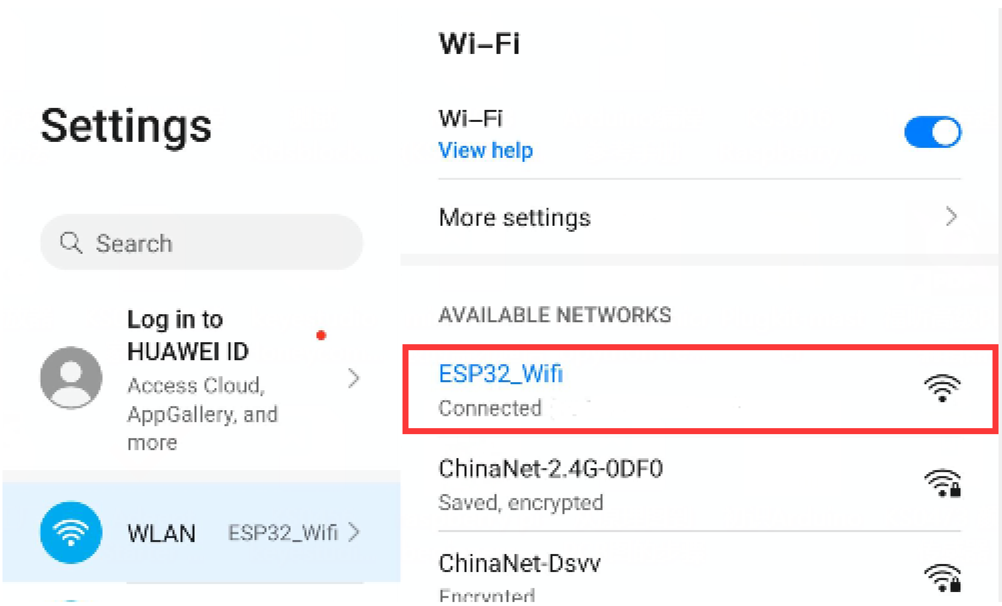
Ensure that the code in the program has been modified correctly, compile and upload the code to the ESP32. After uploading successfullywe will use a USB cable to power on.

Open the serial monitor and set the baud rate to 115200, then monitor 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)





Open the WiFi scanning function of the mobile phone, you can see the ssid_AP.



CHAPTER

SIX

GETTING STARTED WITH PYTHON

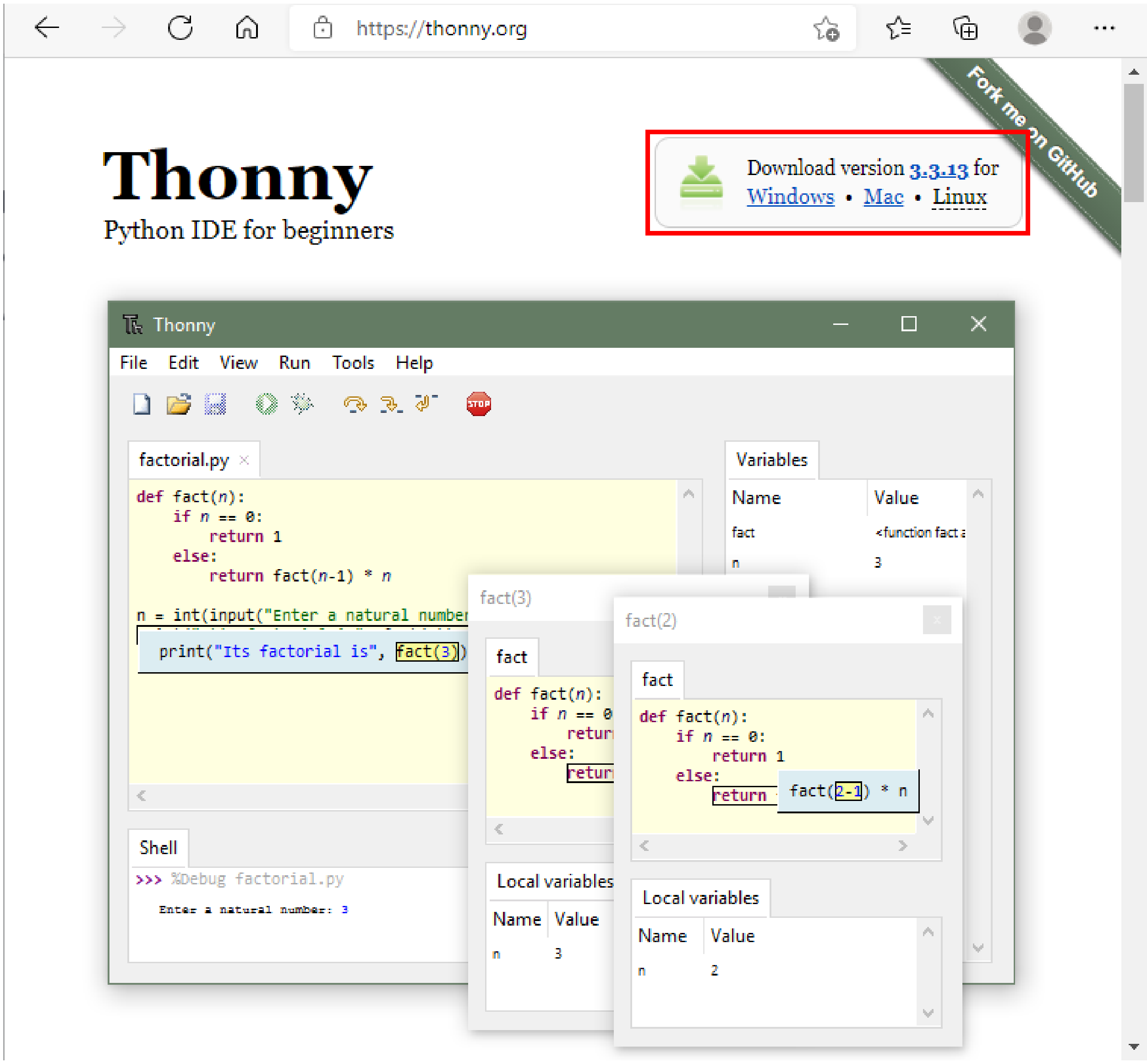
1.Install Thonny

Thonny is a free and open source software platform with small size, simple interface, simple operation and rich functions. It is a Python IDE suitable for beginners. In this tutorial, we use this IDE to develop a ESP32. Thonny supports multiple operating systems including Windows, Mac OS, Linux.

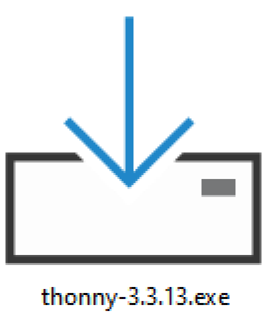
2.Download Thonny

1) Enter the websitehttps://thonny.org to download the latest version of Thonny.

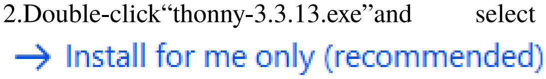
(2)Thonny open-source code libraryhttps://github.com/thonny/thonny.

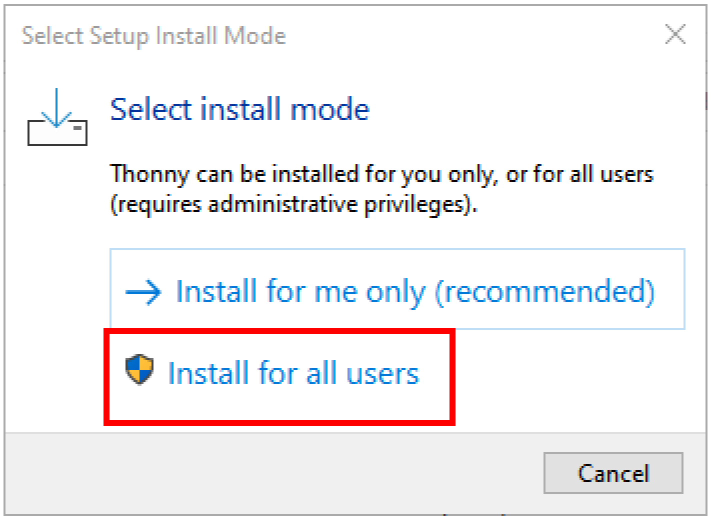


1. The downloaded Thonny icon is as follow:

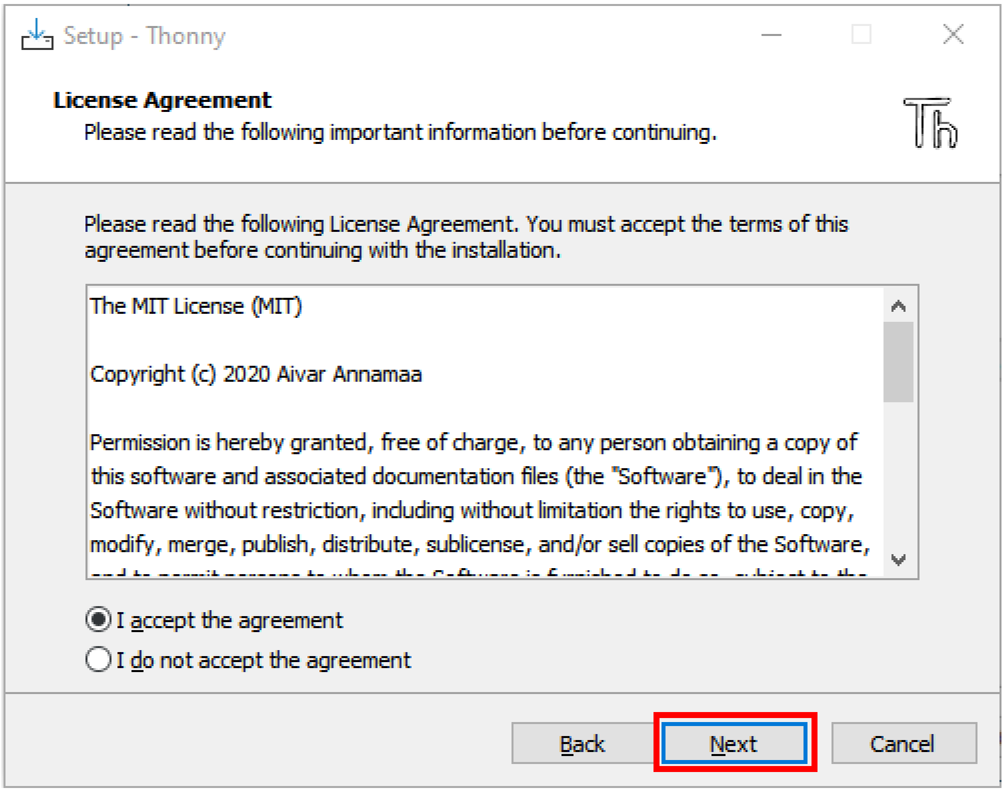


2.Double-click"thonny-3.3.13.exe"and select install mode. You can choose

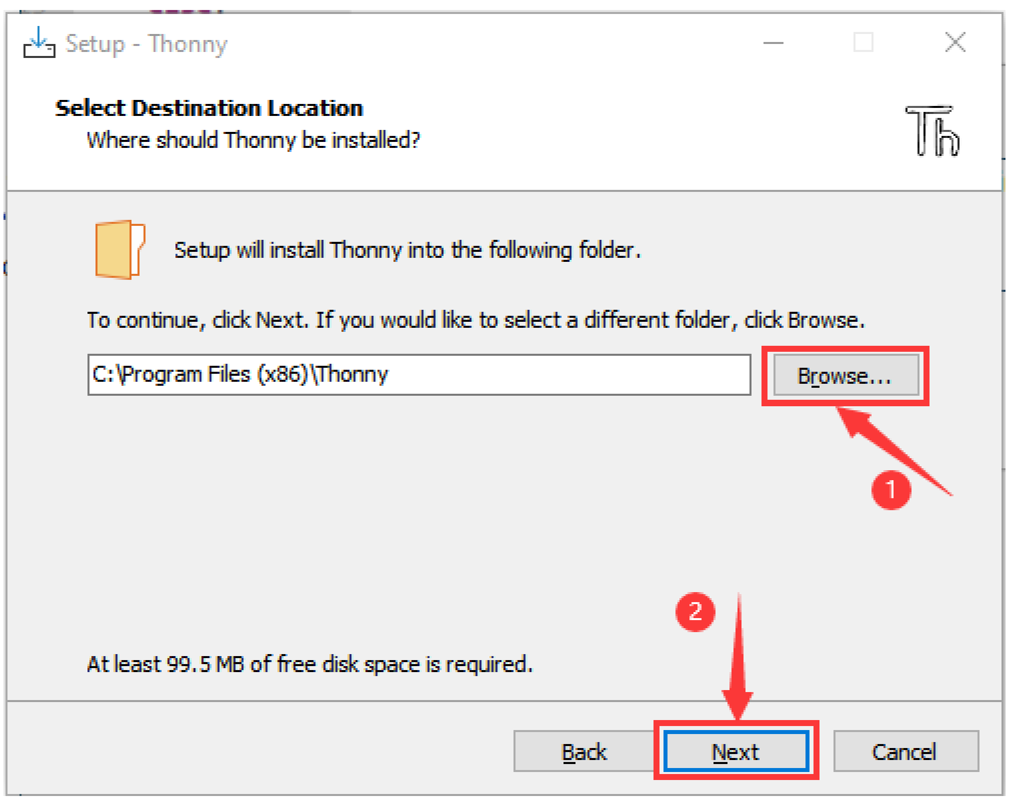


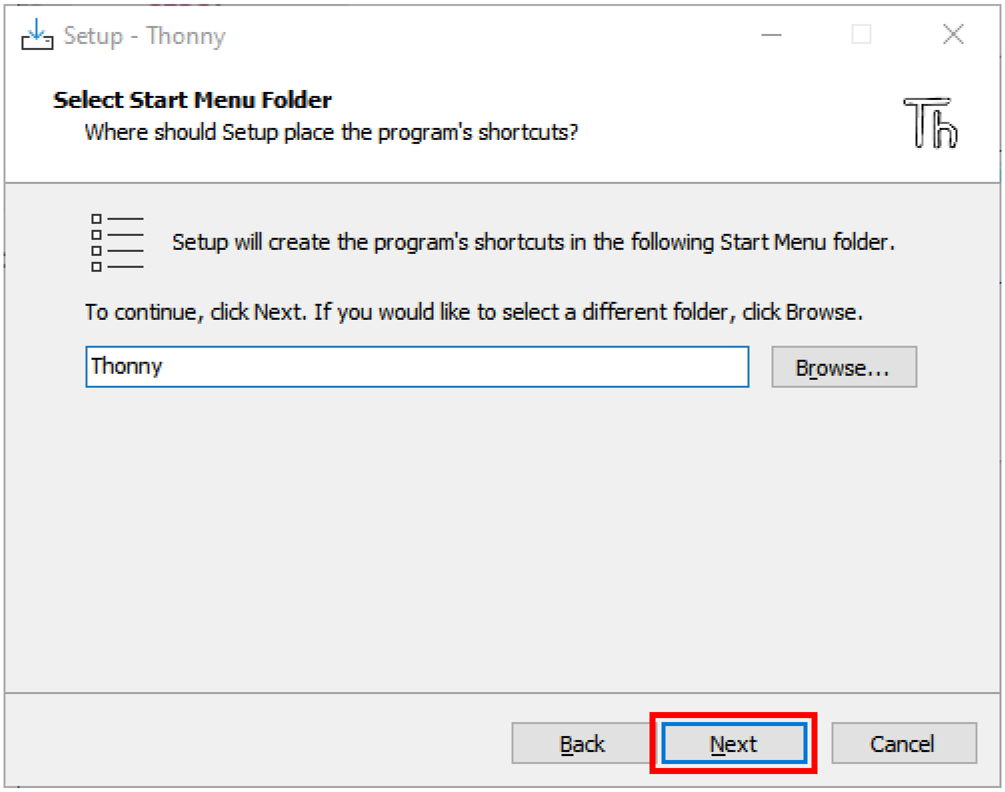


3.You can also keep selecting **Next** to finish the installation.

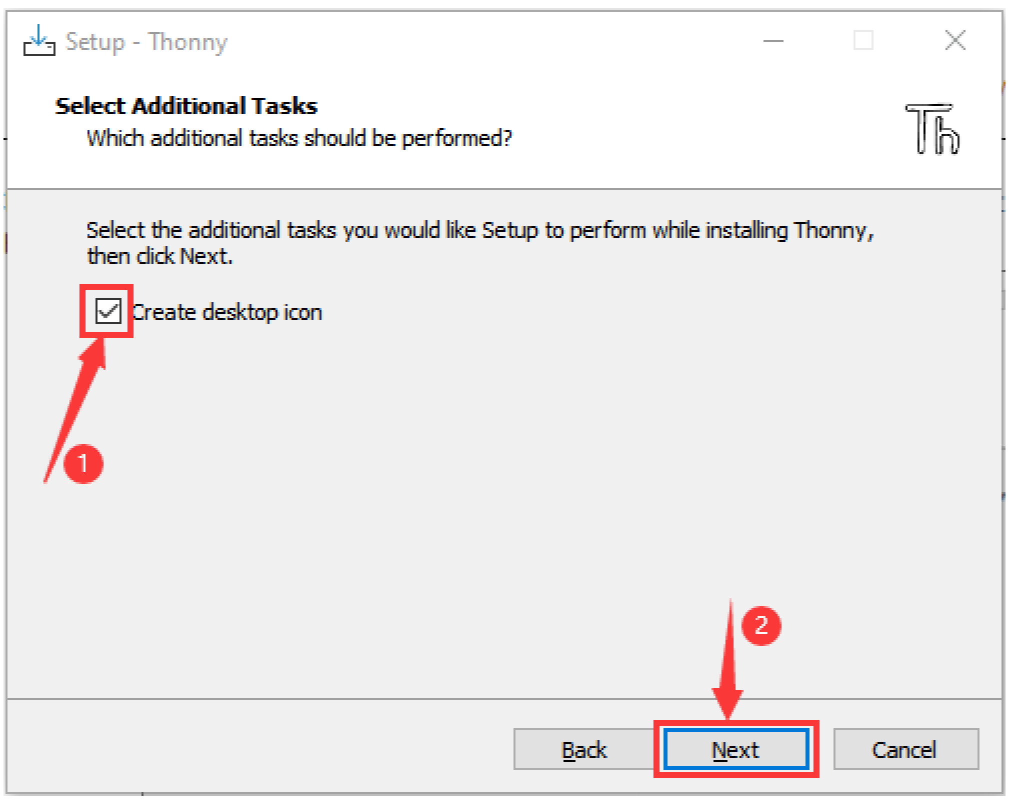


4.If you want to change the route of installing Thonnyjust click"**Browse...**"to select a new route and click **OK**.

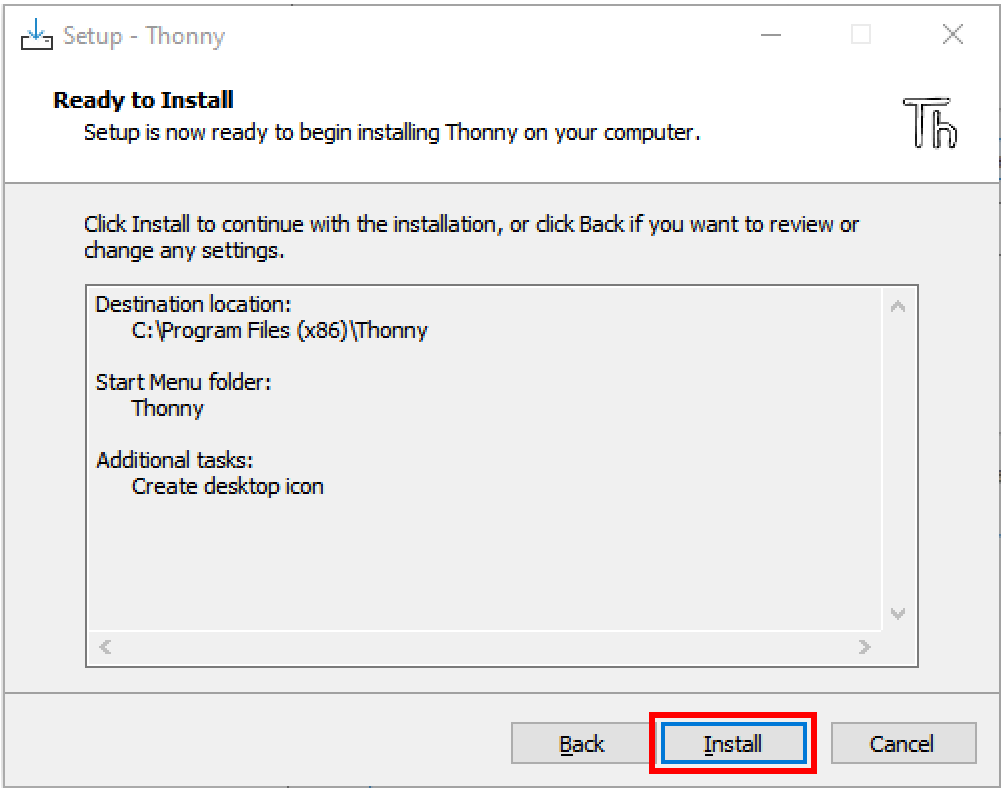




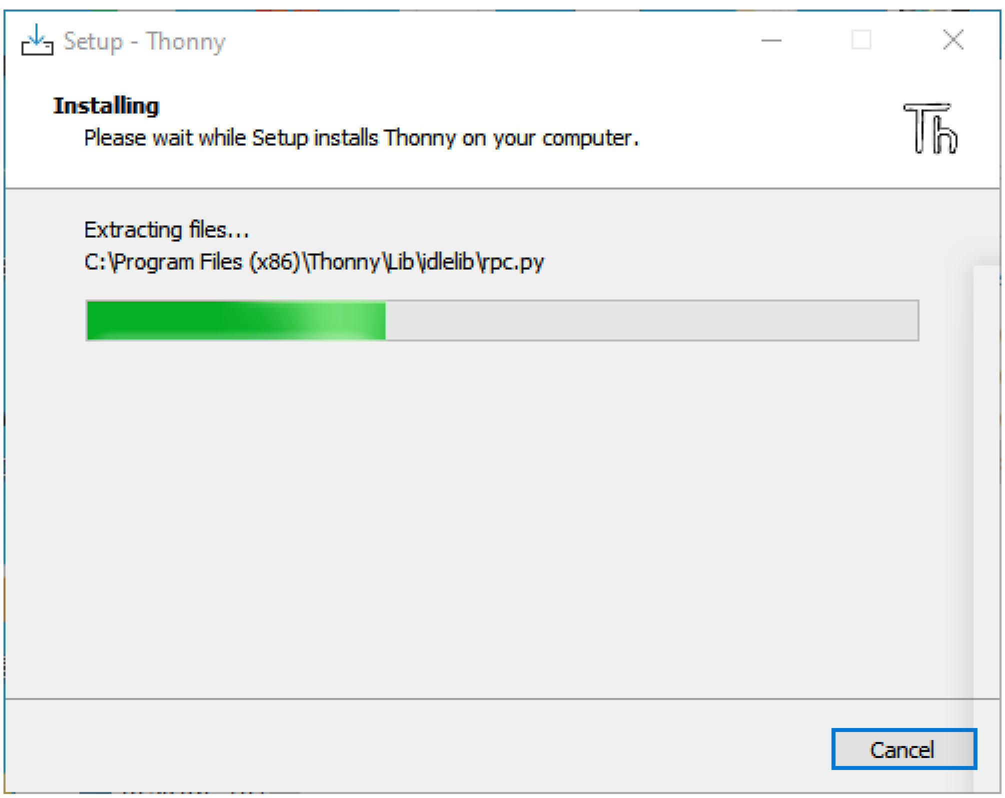
5.Click **Create desktop icon,** you will view Thonny on your desktop.



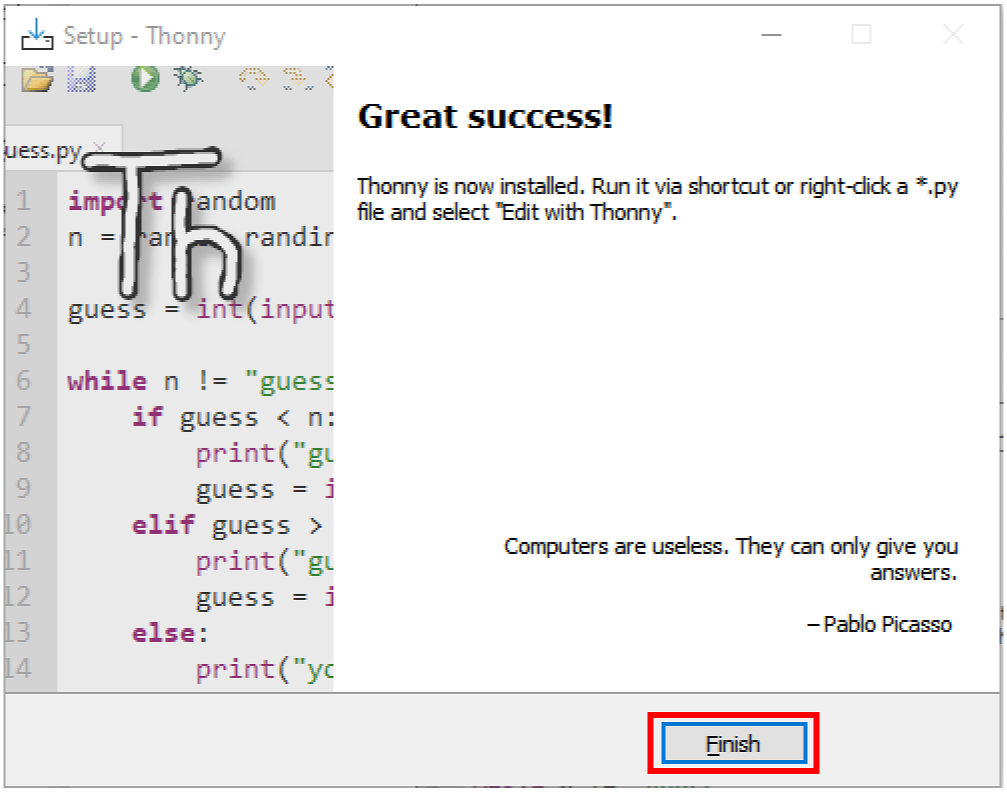
6.Click"**Install**"



7.Wait for a while but don't click **Cancel

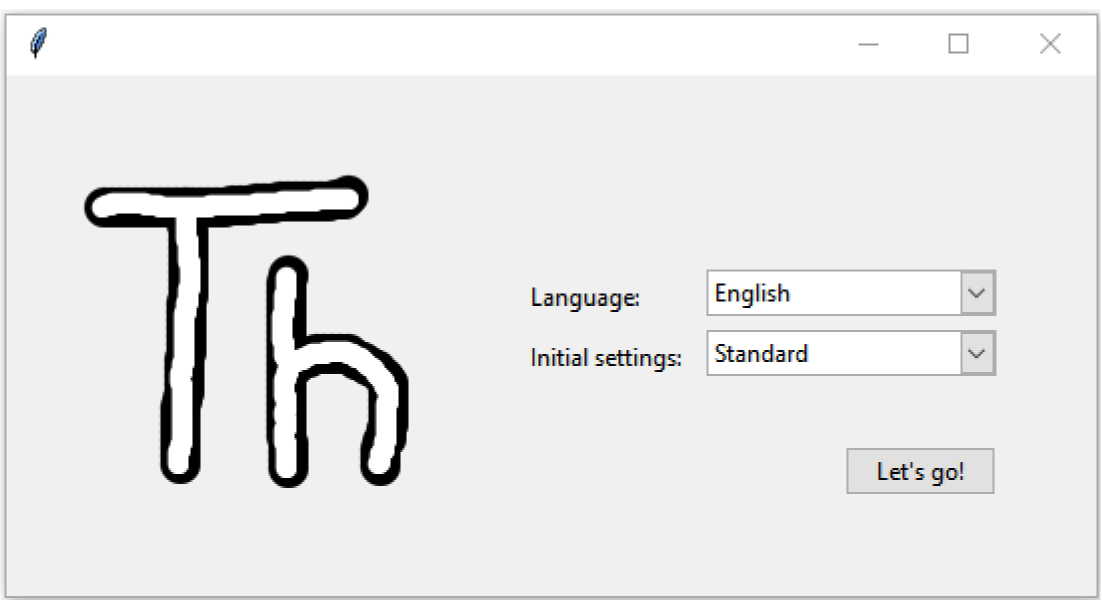


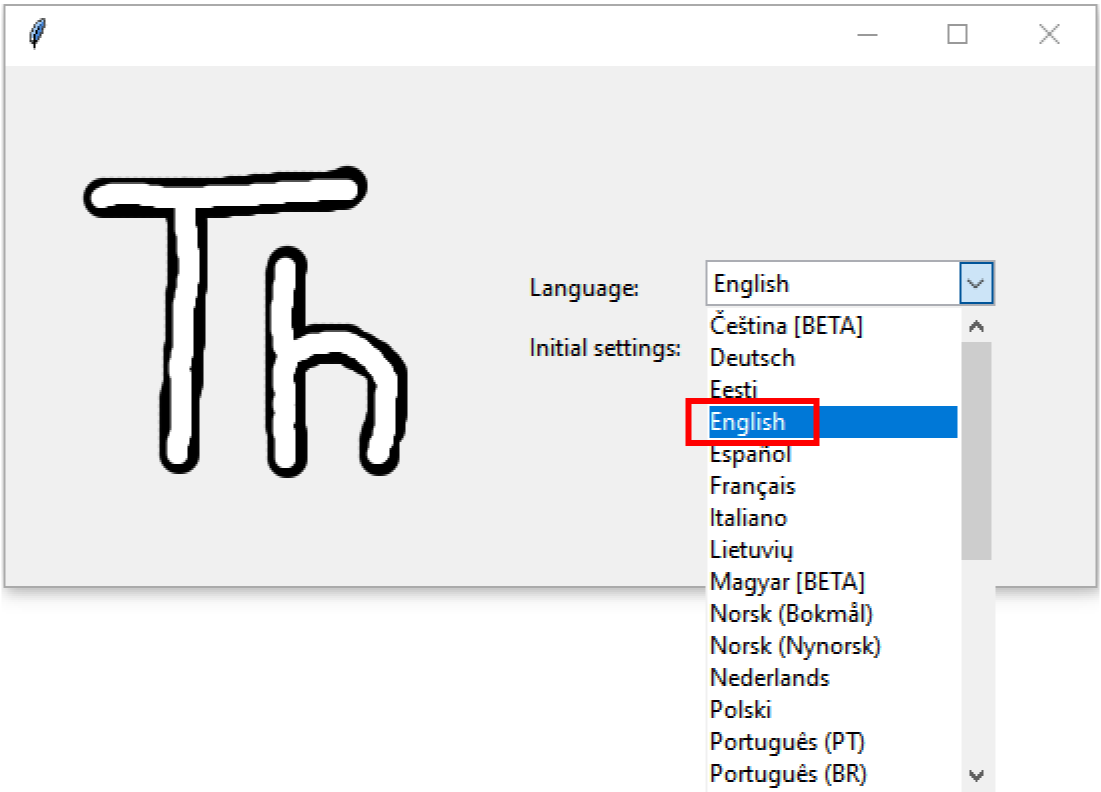
G. Click**"Finish"**

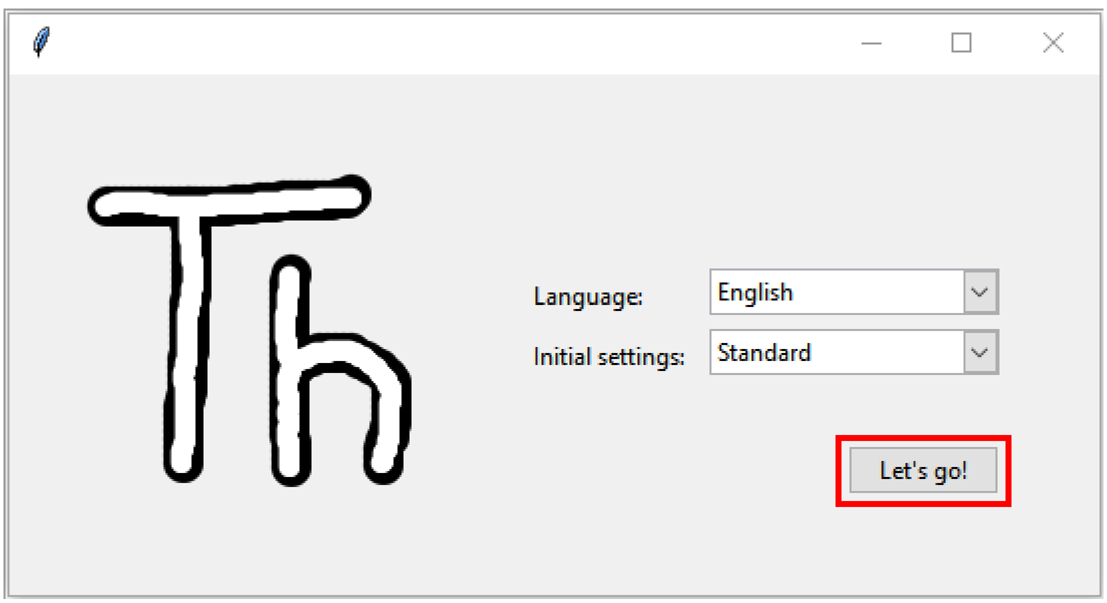


2.Basic Setting

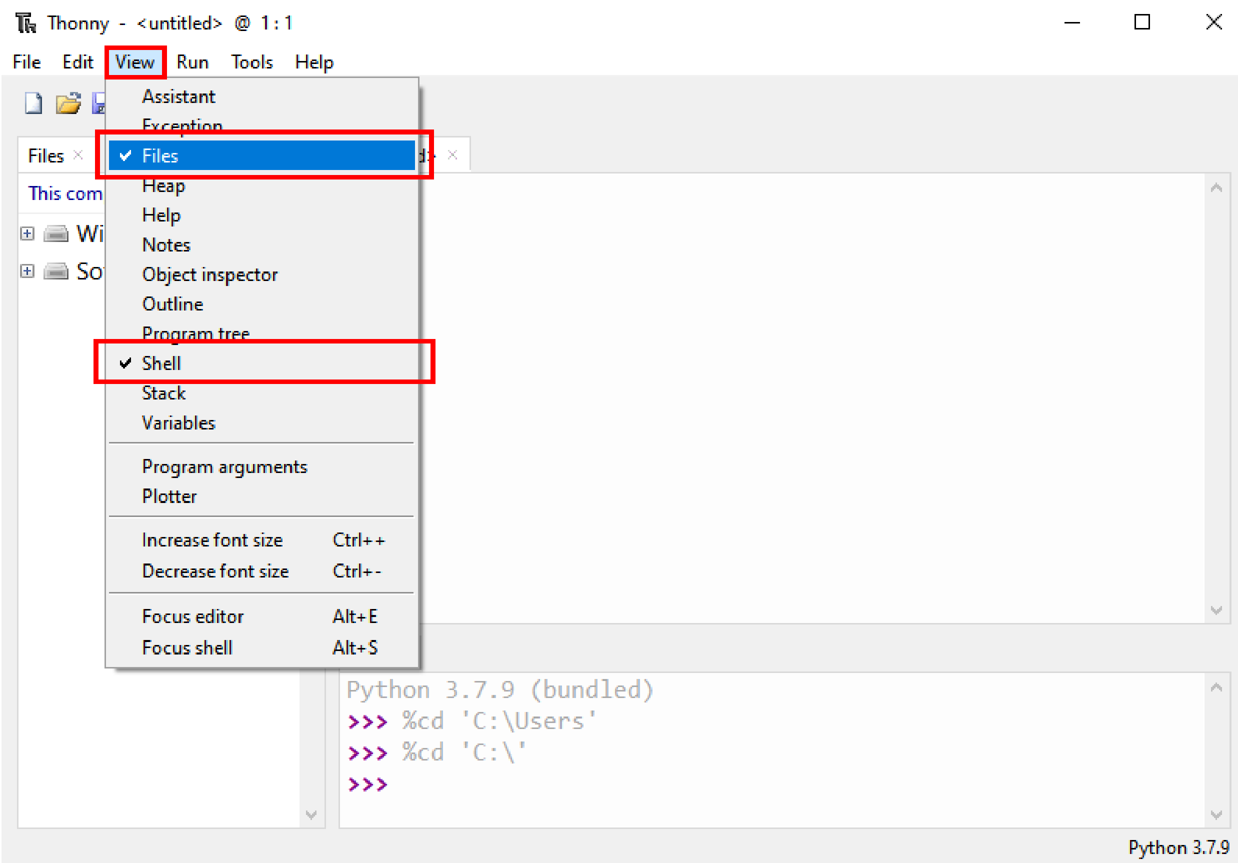
Double-click Thonny, choose language and initial settings and click **Let's go**

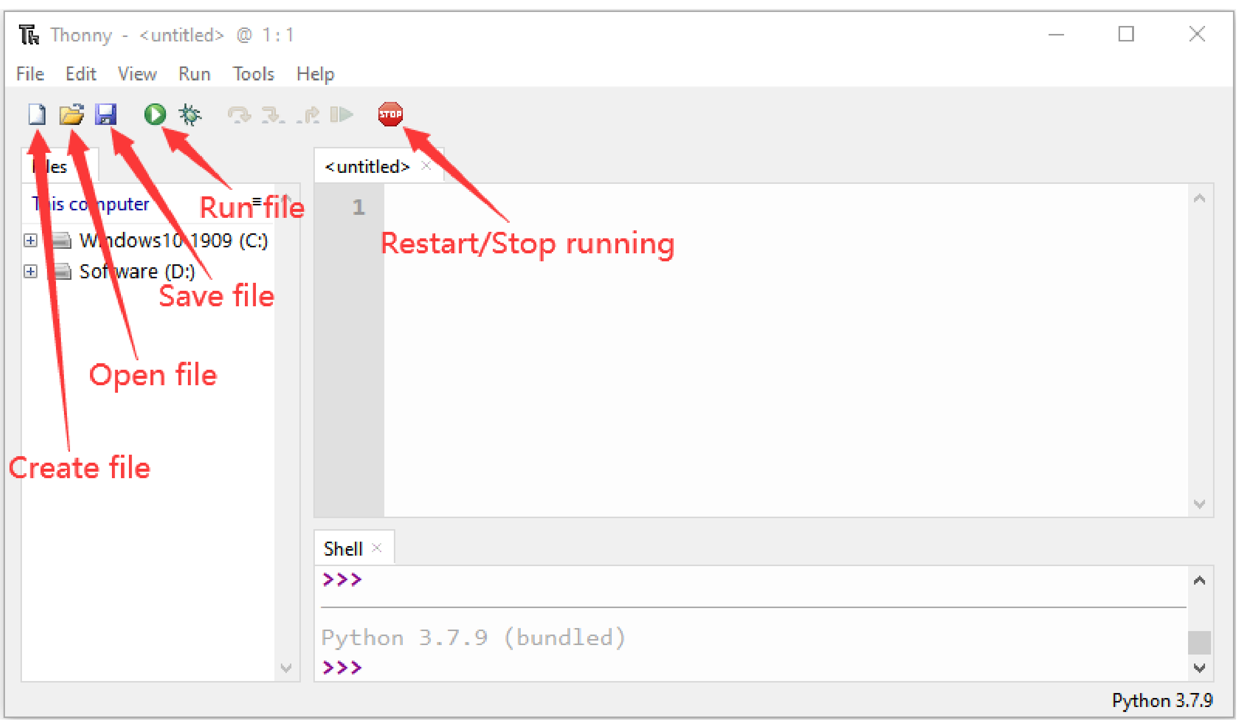


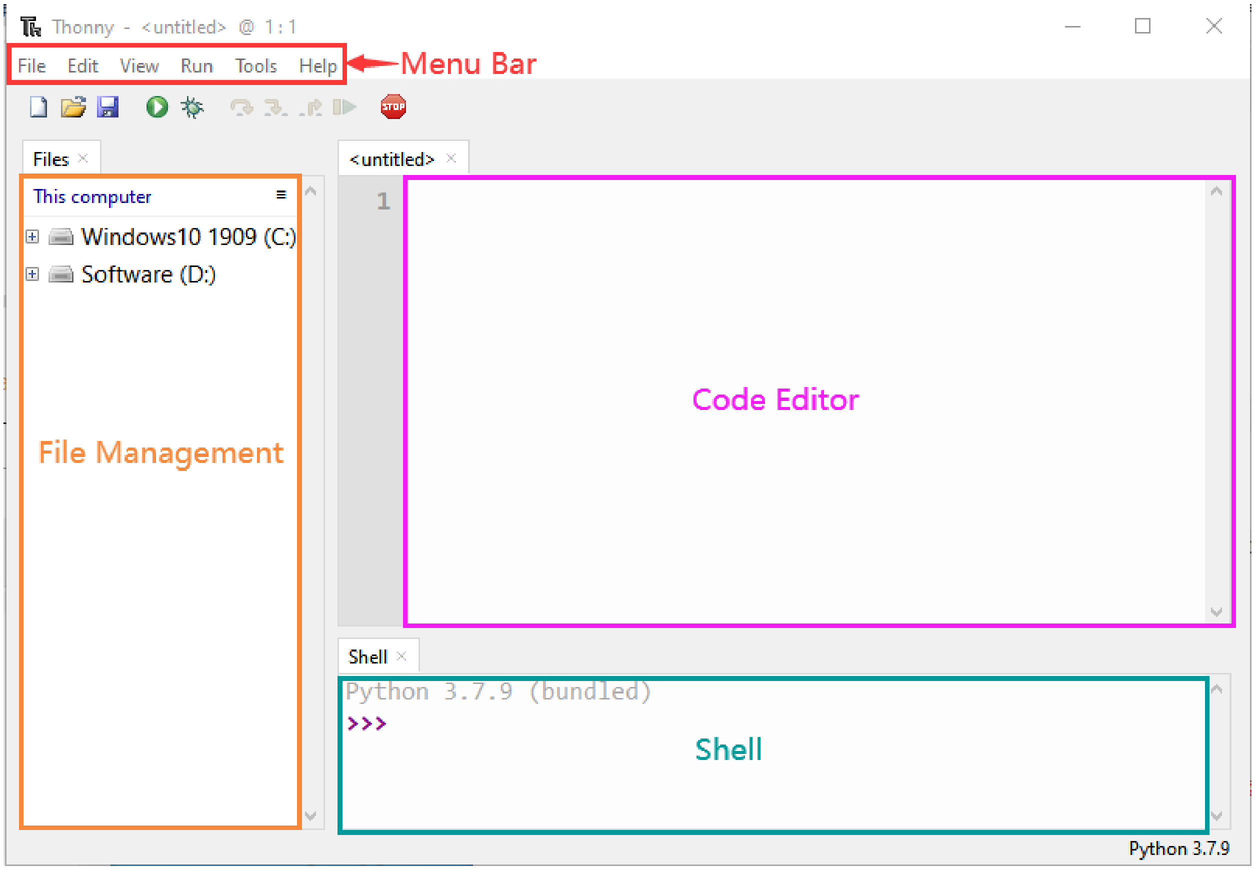




Click"**View**"→"**File**"and"**Shell**"


Thonny - <untitled> @ 1 : 1
File Edit View Run Tools Help
Assistant
Exception
Files
Heap
Help
Notes
Object inspector
Outline
Program tree
Shell
Stack
Variables
Program arguments
Plotter
Increase font size Ctrl++
Decrease font size Ctrl+-
Focus editor Alt+E
Focus shell Alt+S
Python 3.7.9 (bundled)
>>> %cd 'C:\Users'
>>> %cd 'C:\'
>>>
Python 3.7.9


Thonny - <untitled> @ 1 : 1
File Edit View Run Tools Help
Run file
Restart/Stop running
Save file
Open file
Create file
<untitled>
Shell
>>>
Python 3.7.9 (bundled)
>>>
Python 3.7.9



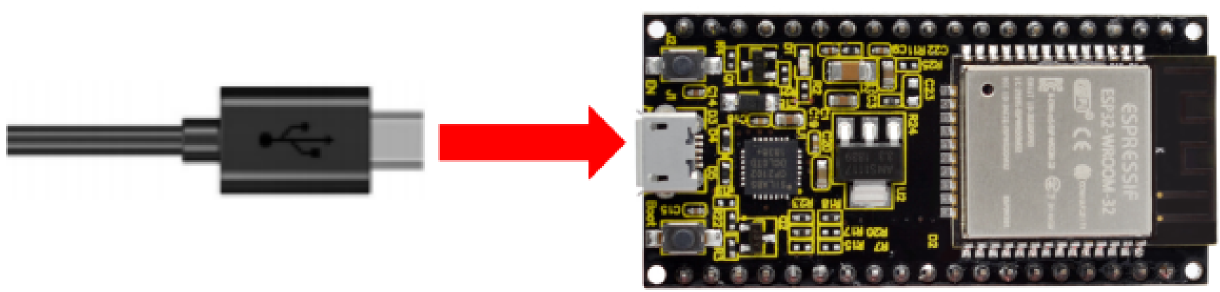
Install the CP2102 driver

Before using the Thonny, we need to install the CP2102 driver in the computer.

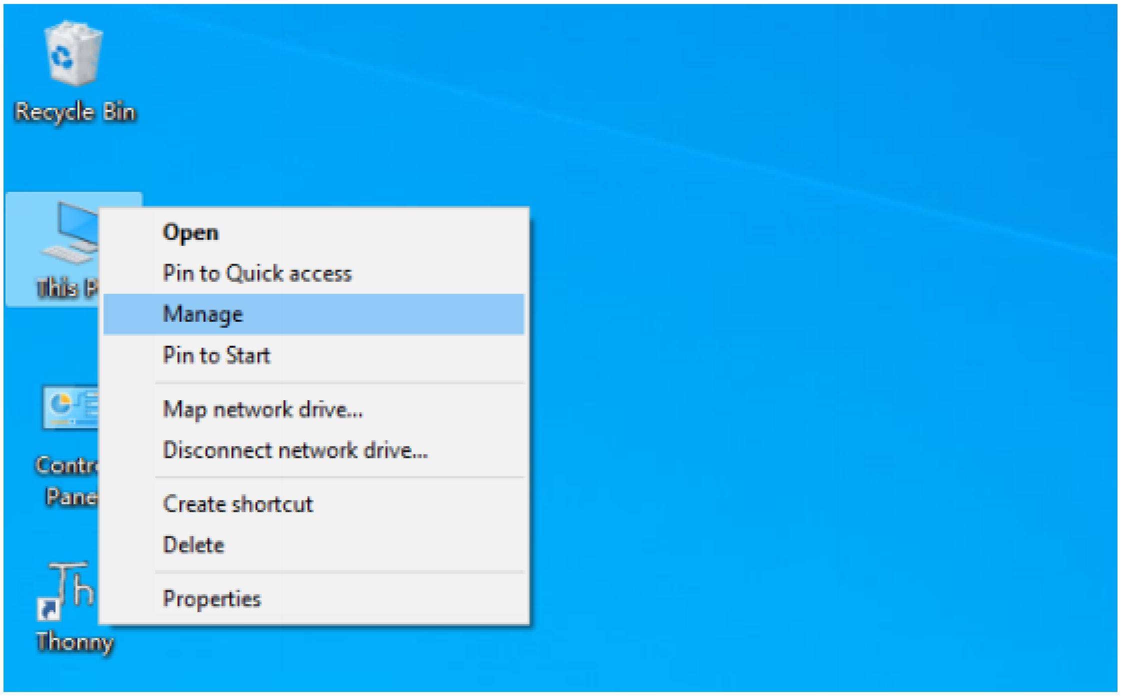
Windows system

Check if the CP2102 driver has been installed

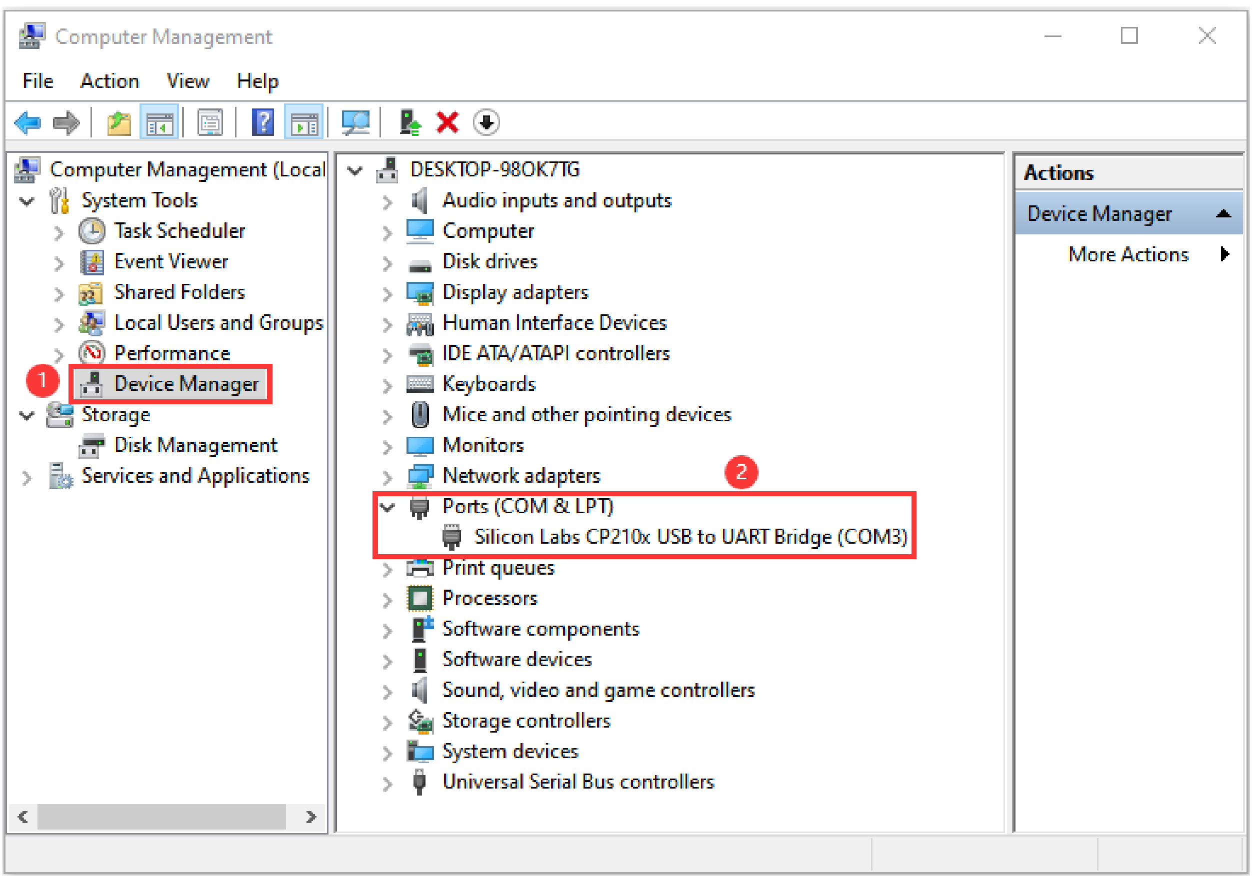
1. Interface the ESP32 with your PC with a USB cable



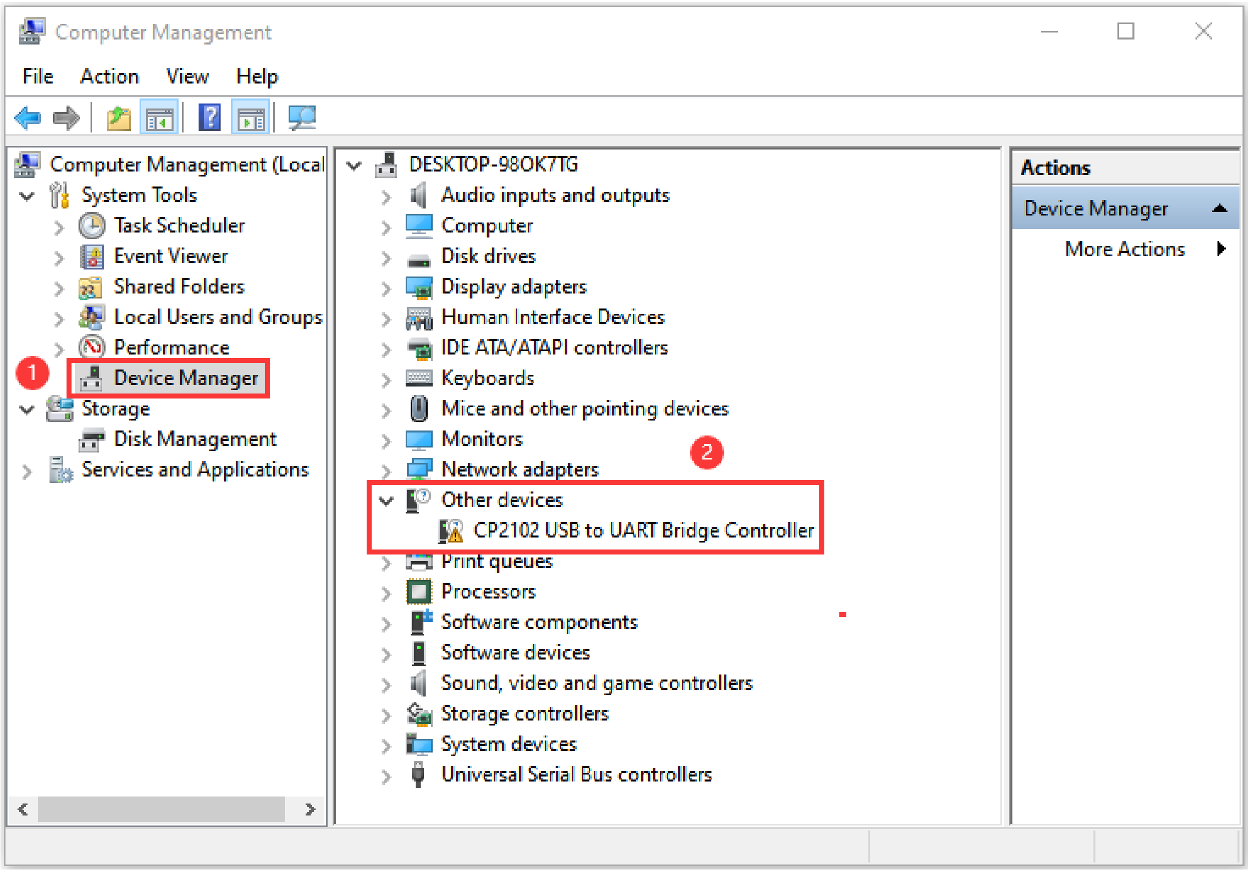
2. Click"This PC"and right-click Manage"



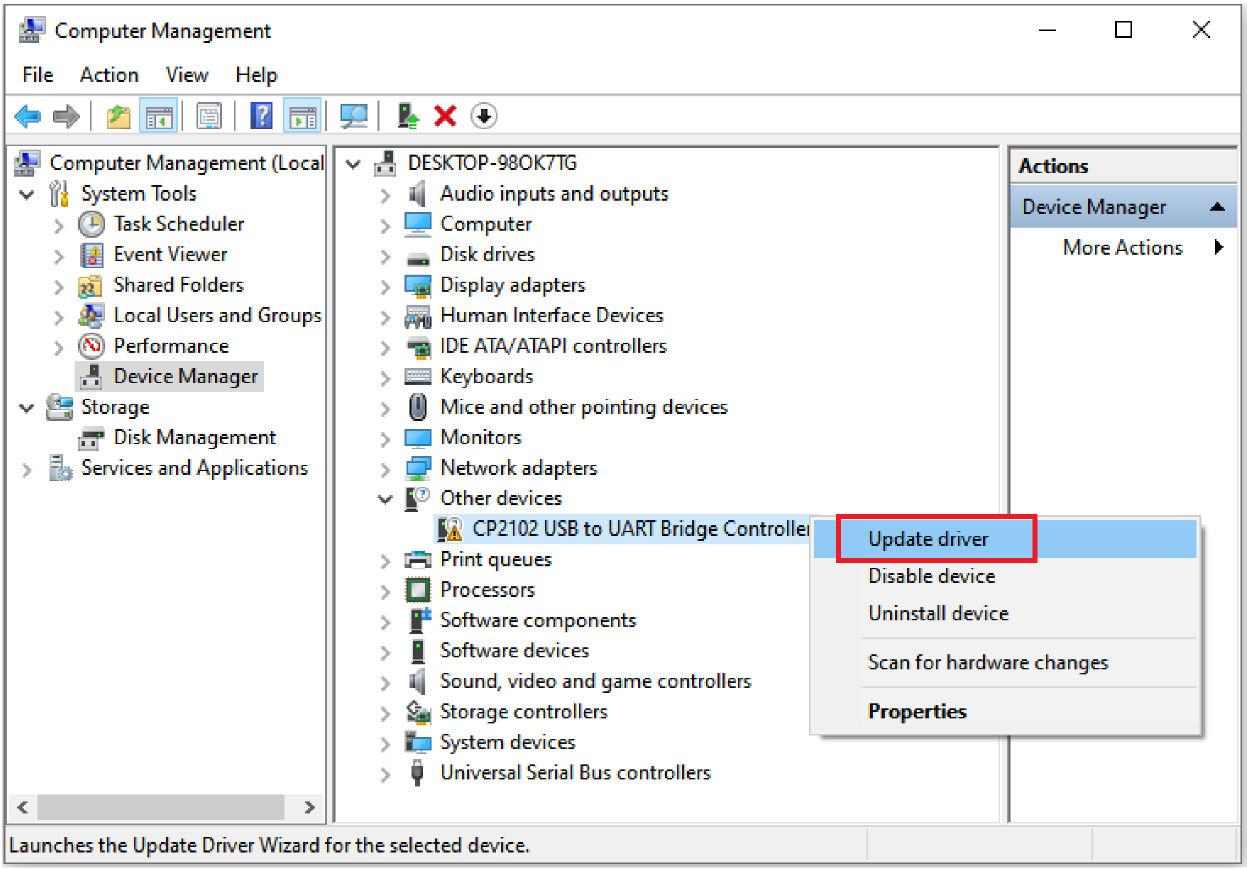
3.Click"Device Manager", if the CP2102 driver has been installedSilicon Labs CP210x USB to UART Bridge(COMx) will be shown.



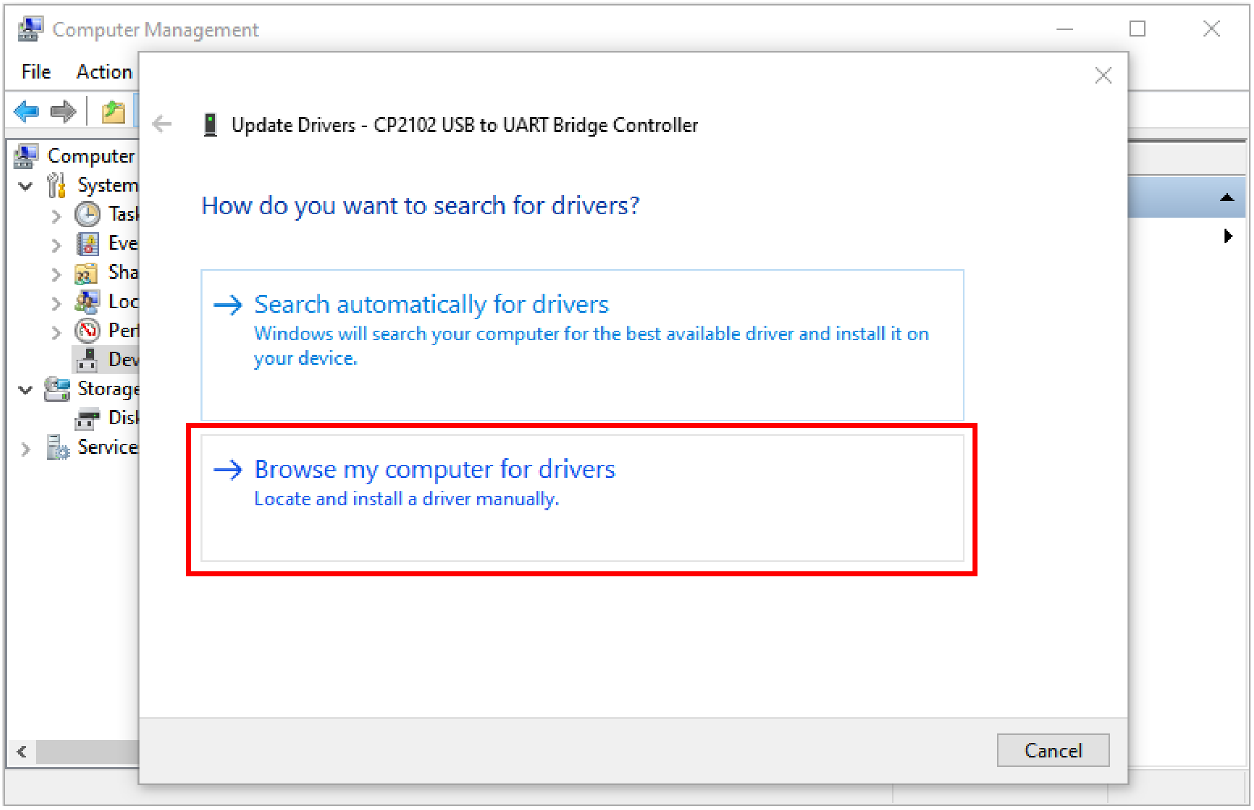
If the CP2102 has not been installed



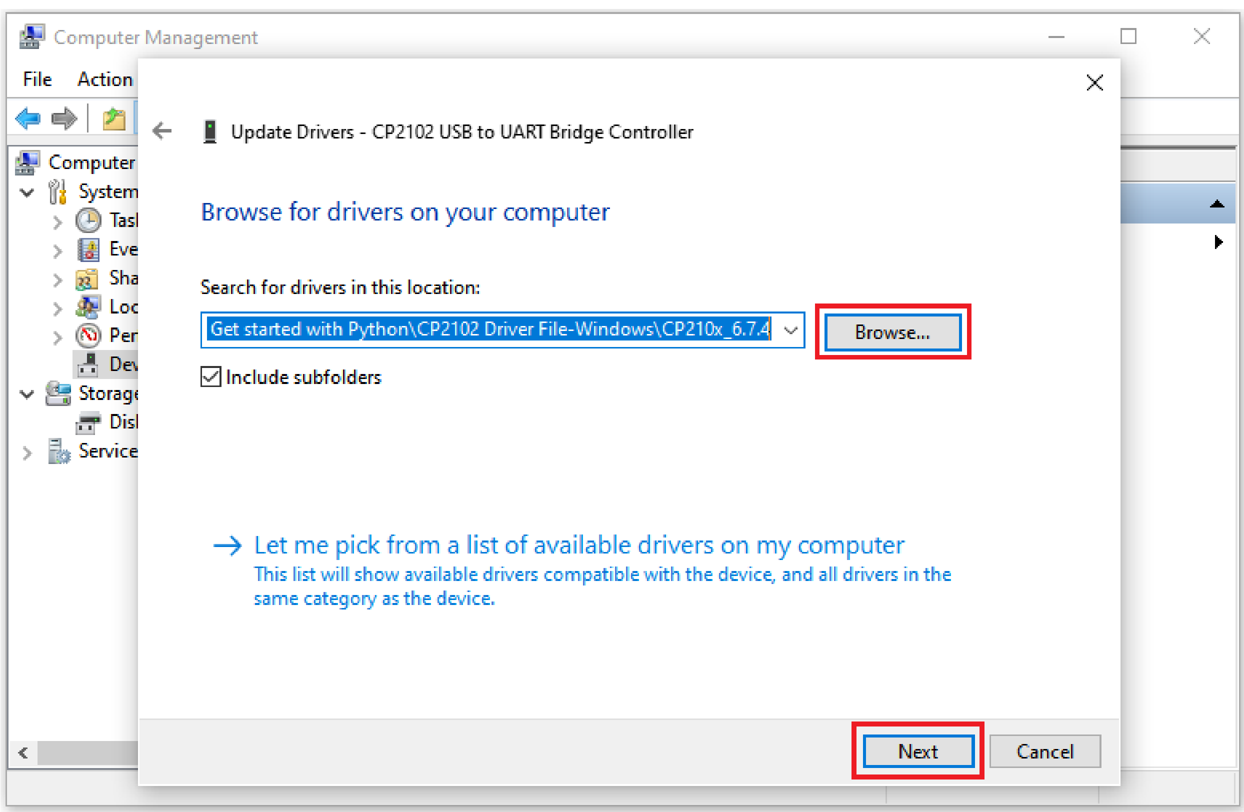
Click"CP2102USB to UART Bridge Controller"and Update driver".



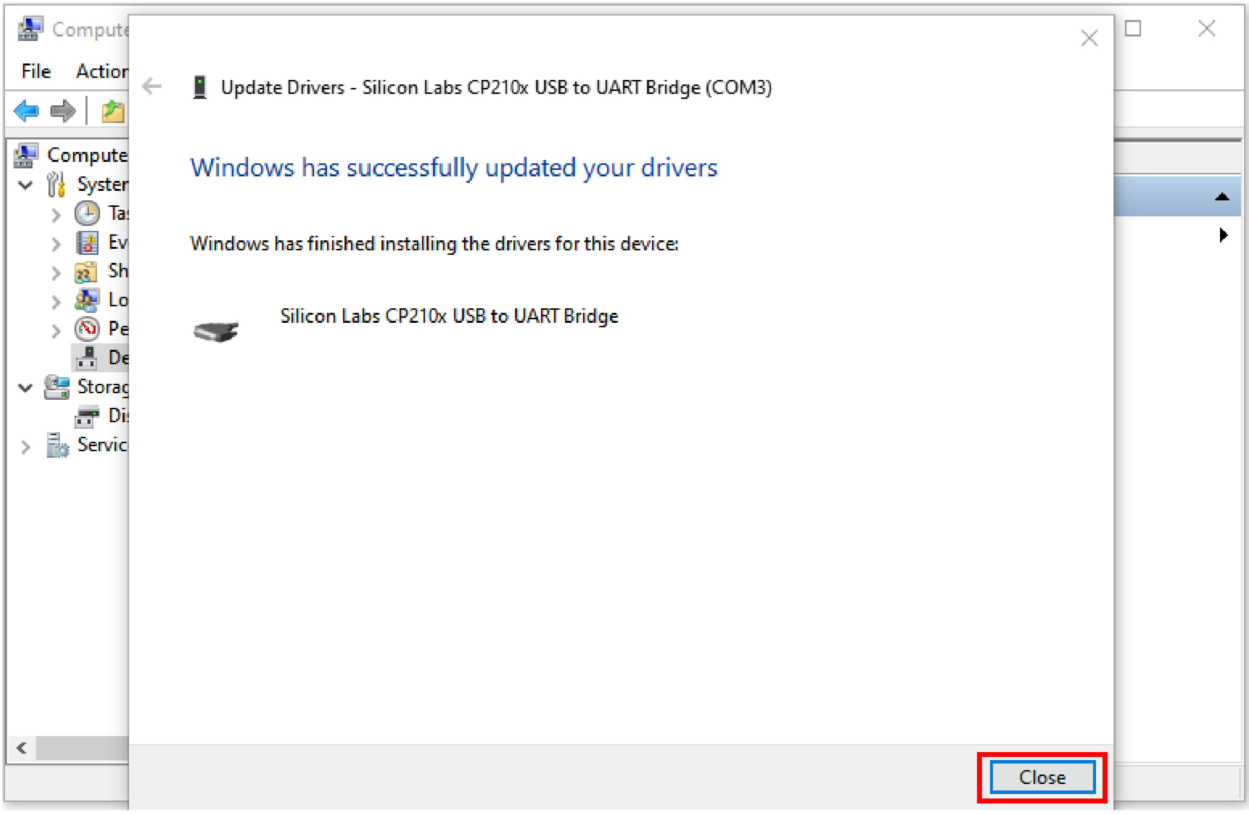
Click“Browse my computer for drivers ”.

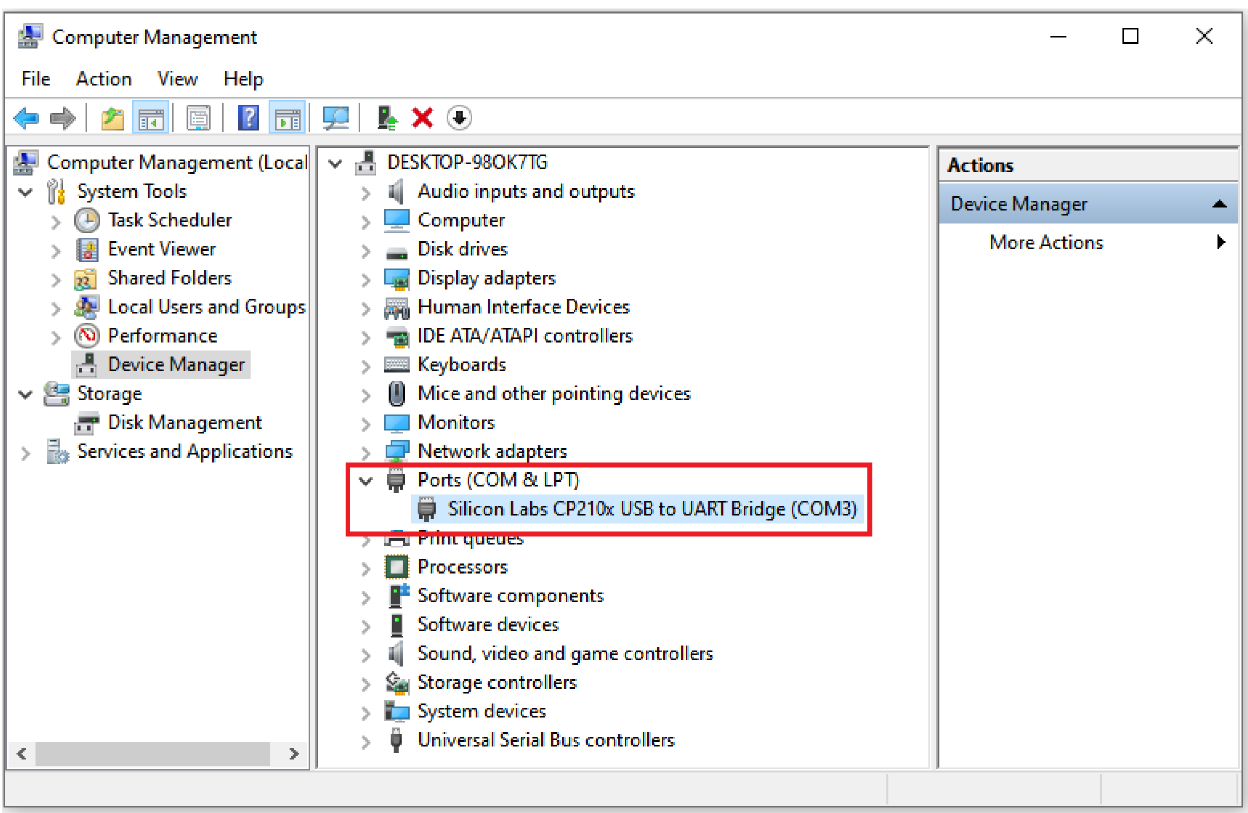


Click Browse… to choose CP210x_6.7.4 ("4. Python Tutorial\1.Development Environment Configuration\CP2102 Driver File-Windows") and click Next



The CP2102 driver will be installed

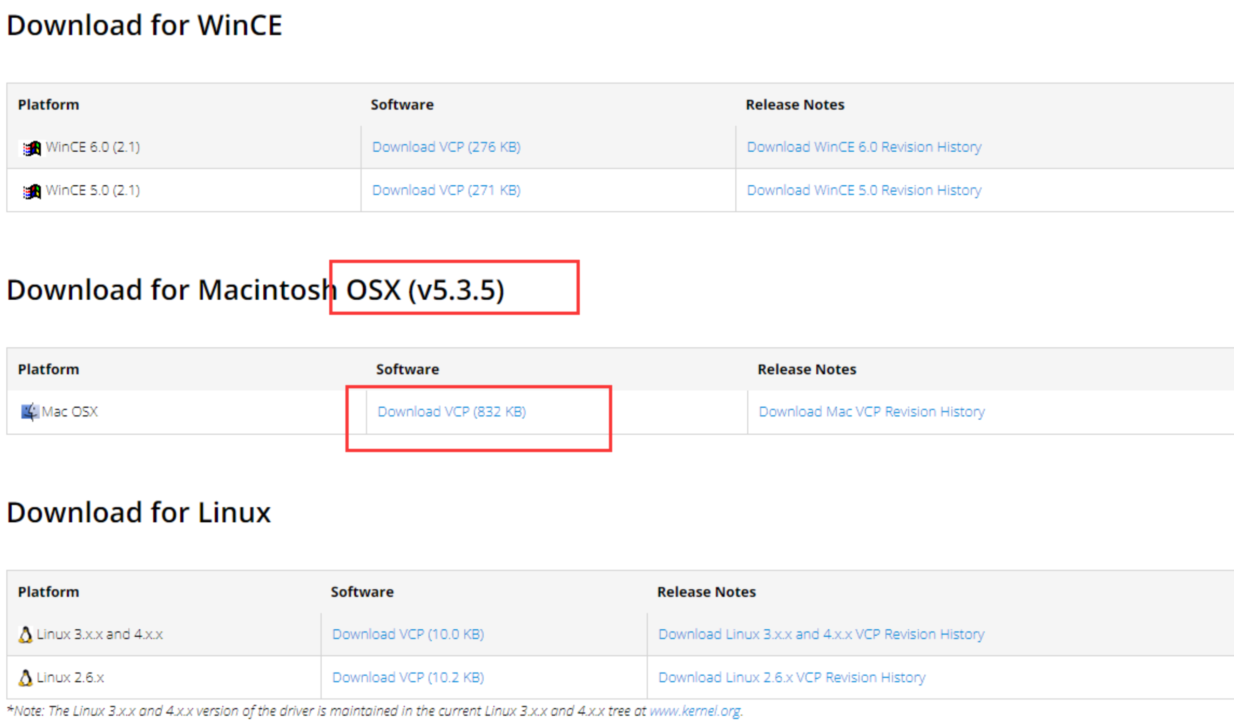




MAC System

Download link for CP2102CP2102-Driver-File-MAC.zip

Download MacOS version



Download for WinCE

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

Download for Macintosh OSX (v5.3.5)

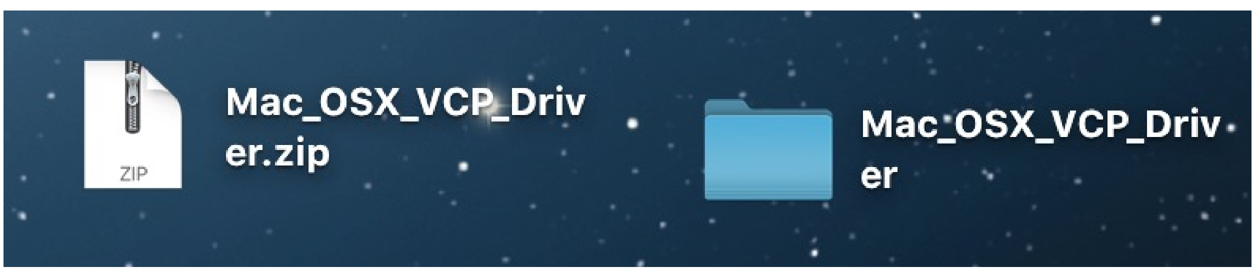
Platform	Software	Release Notes
Mac OSX	Download VCP (832 KB)	Download Mac VCP Revision History

Download for Linux

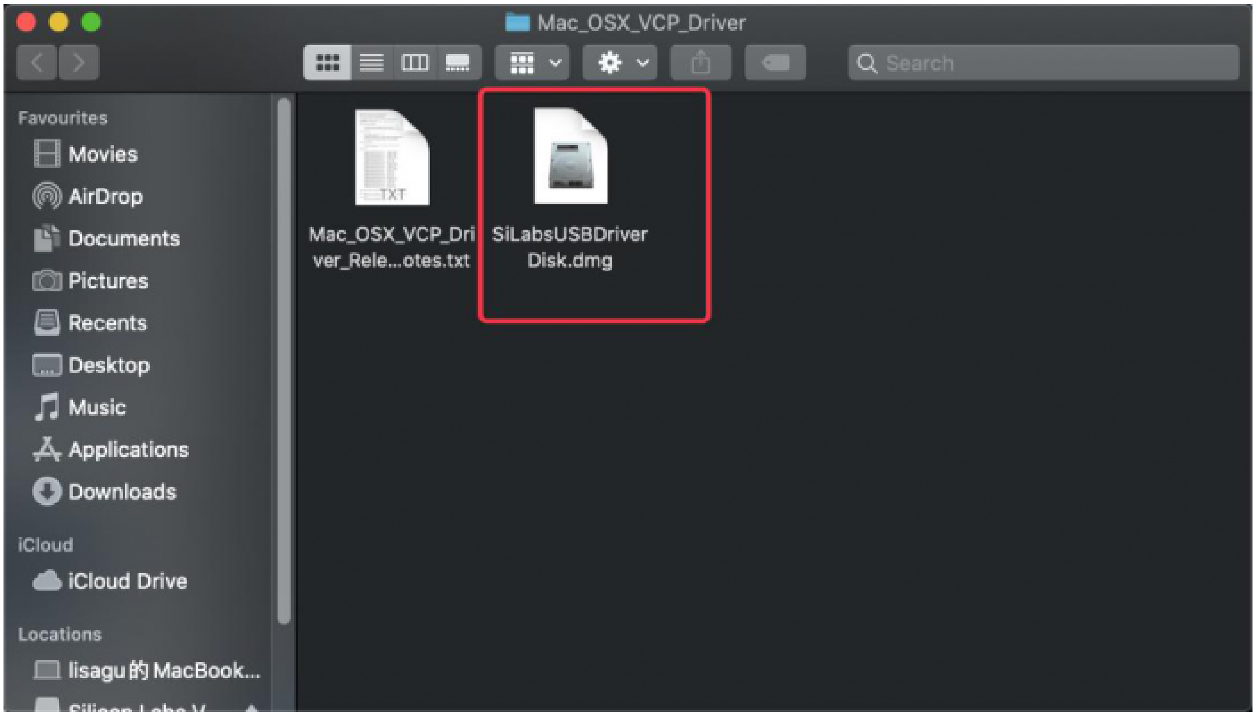
Platform	Software	Release Notes
Linux 3.x.x and 4.x.x	Download VCP (10.0 KB)	Download Linux 3.x.x and 4.x.x VCP Revision History
Linux 2.6.x	Download VCP (10.2 KB)	Download Linux 2.6.x VCP Revision History

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

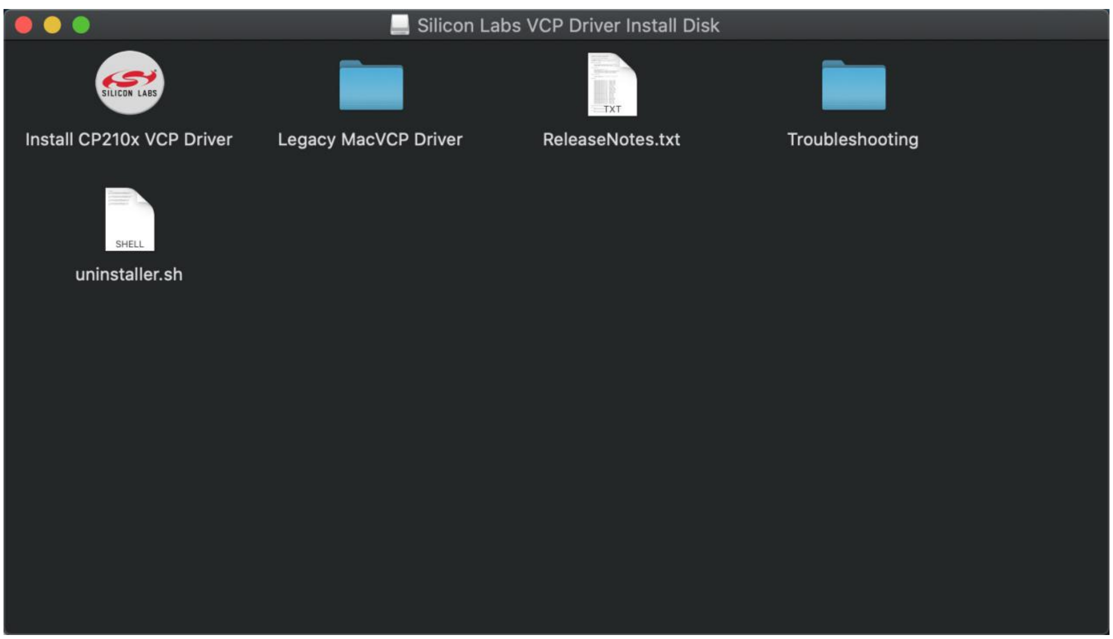
Unzip the downloaded package



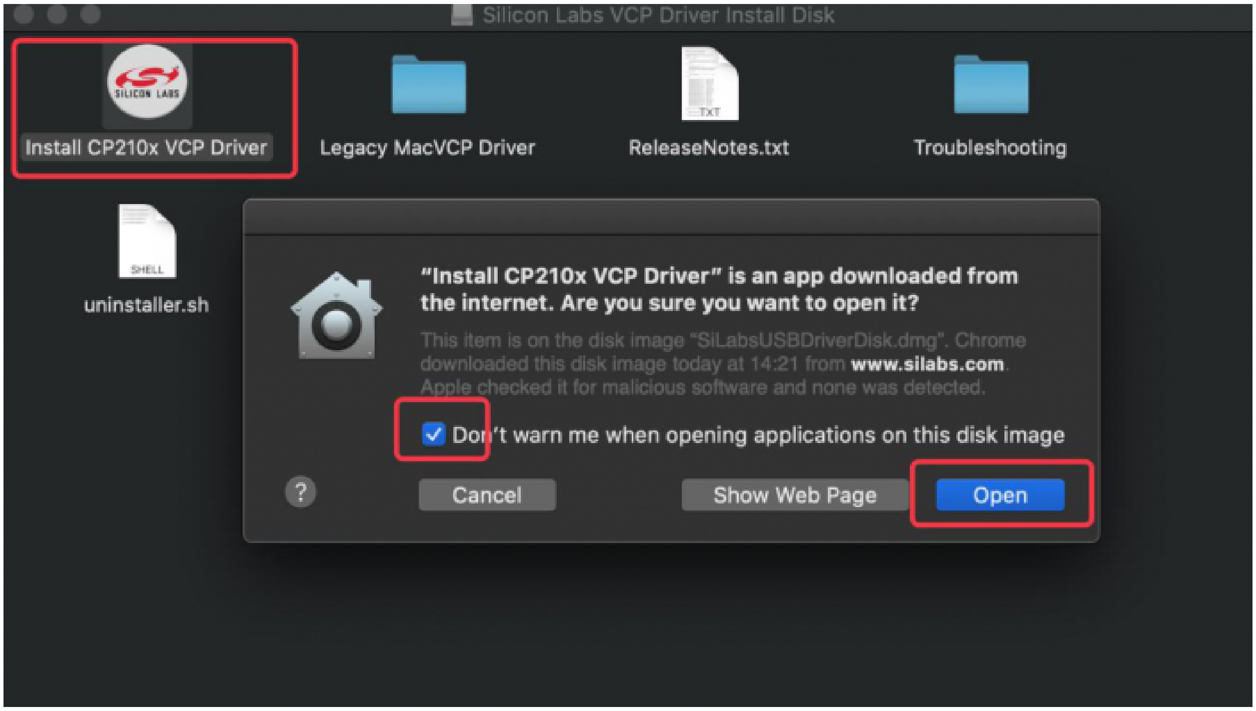
Open the folder and double-click"SiLabsUSBDriverDisk.dmg"file



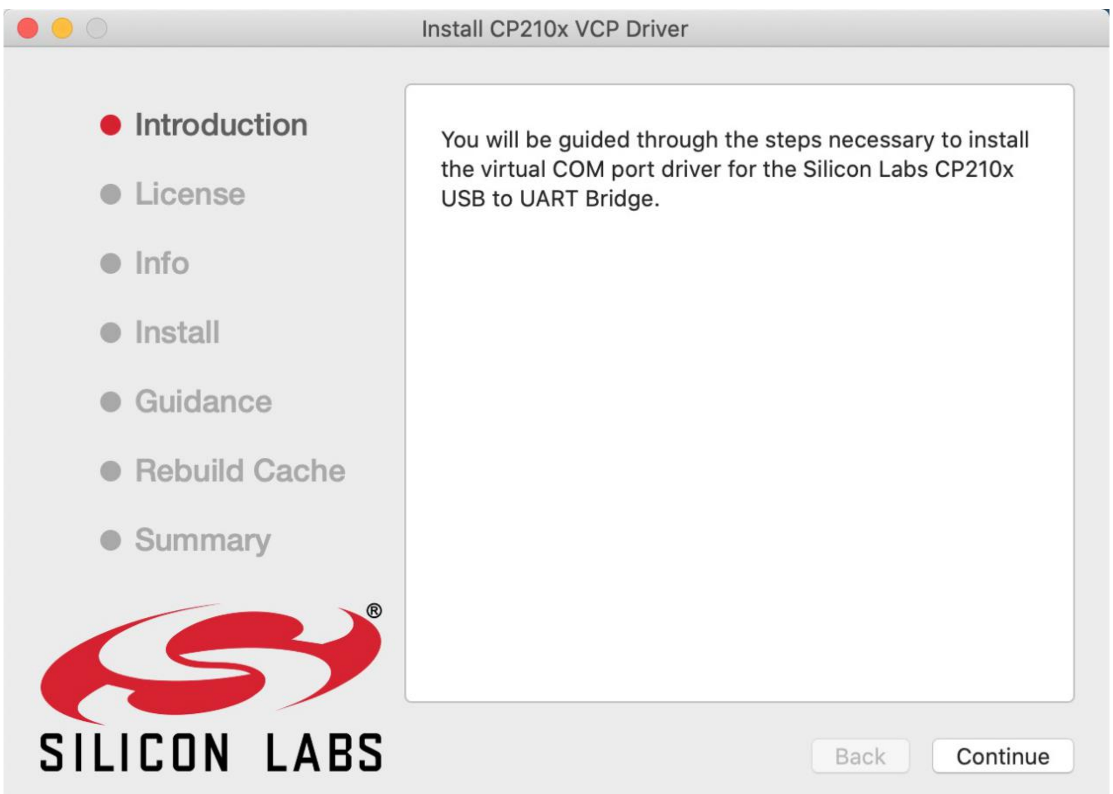
Then you can see the following file



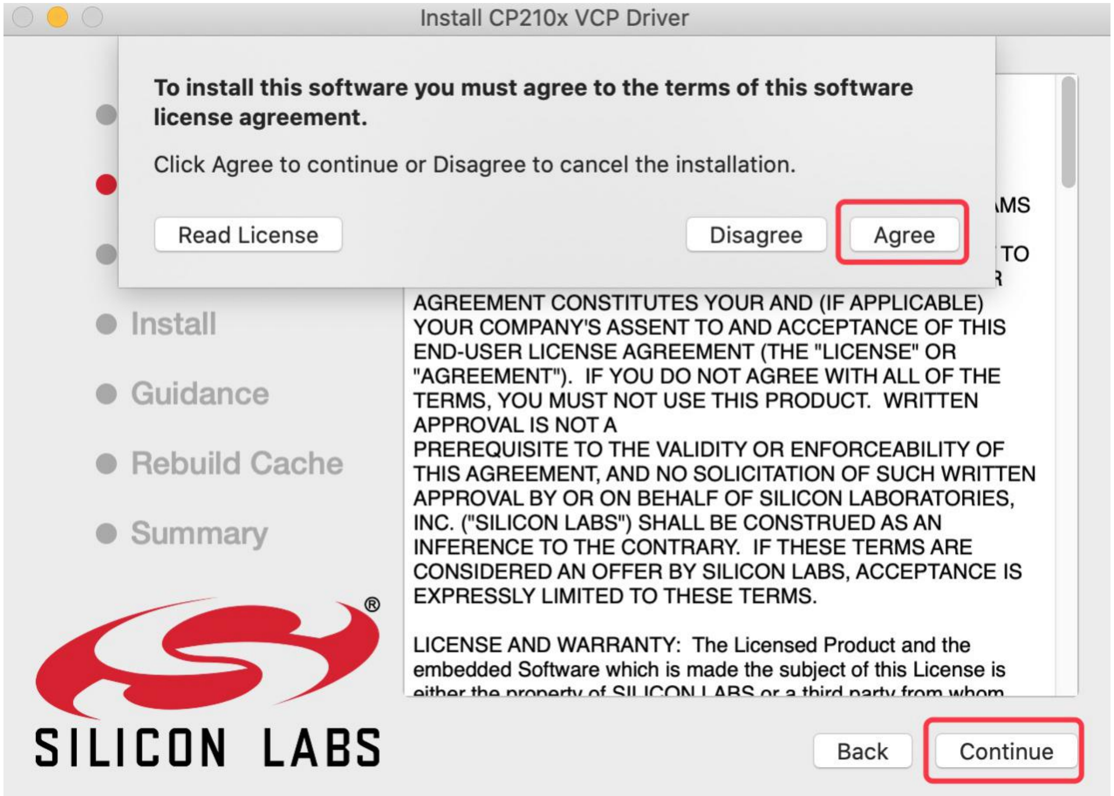
Double-click**"Install CP210x VCP Driver"tap**"**Don't warn me when opening application on this disk image**"and click"**Open**"



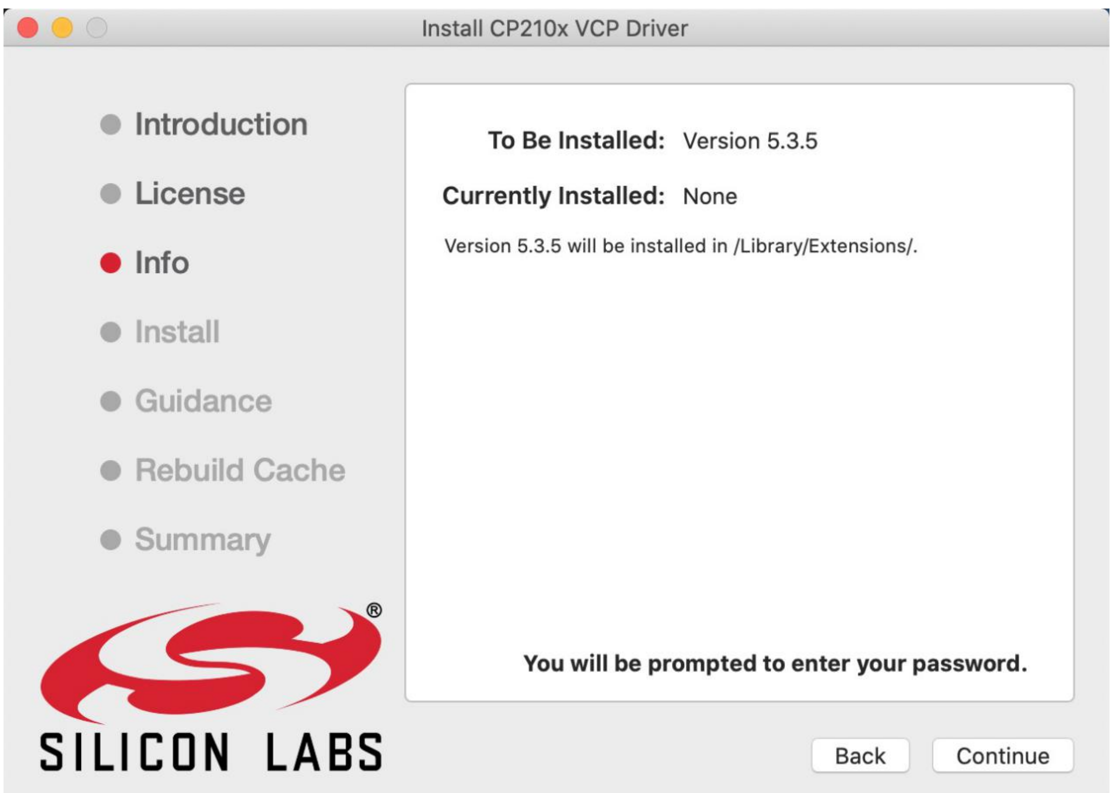
Click"**Continue**"

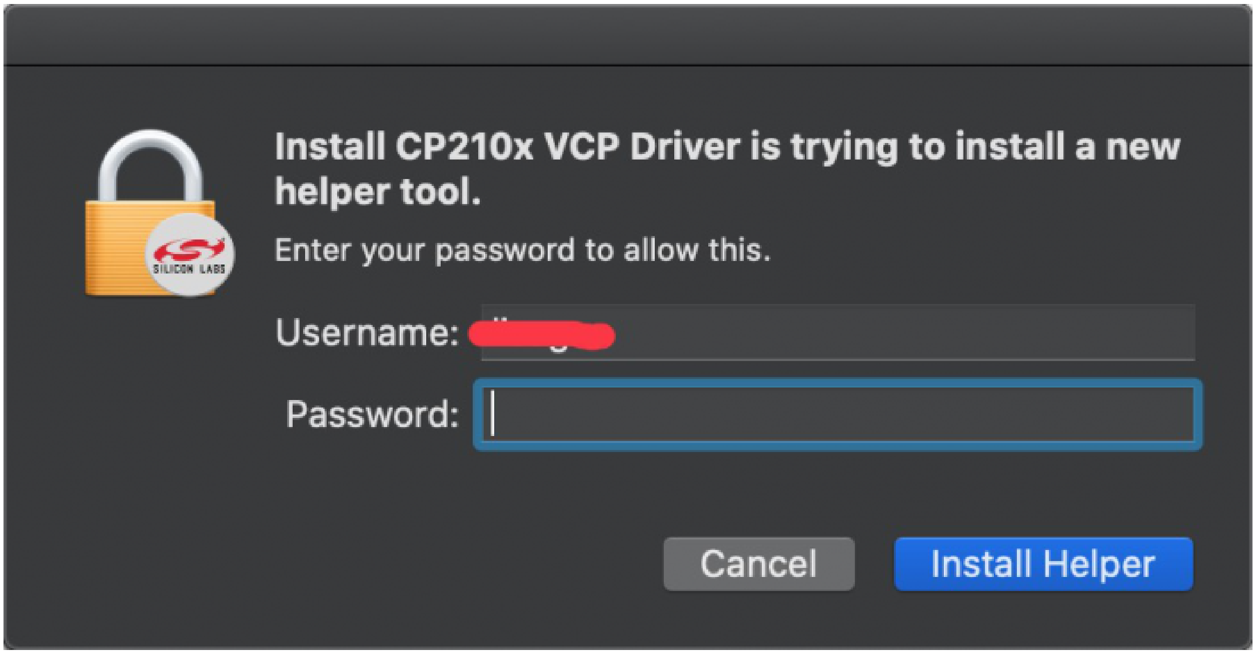


Click"**Agree**"then tap"**Continue**"

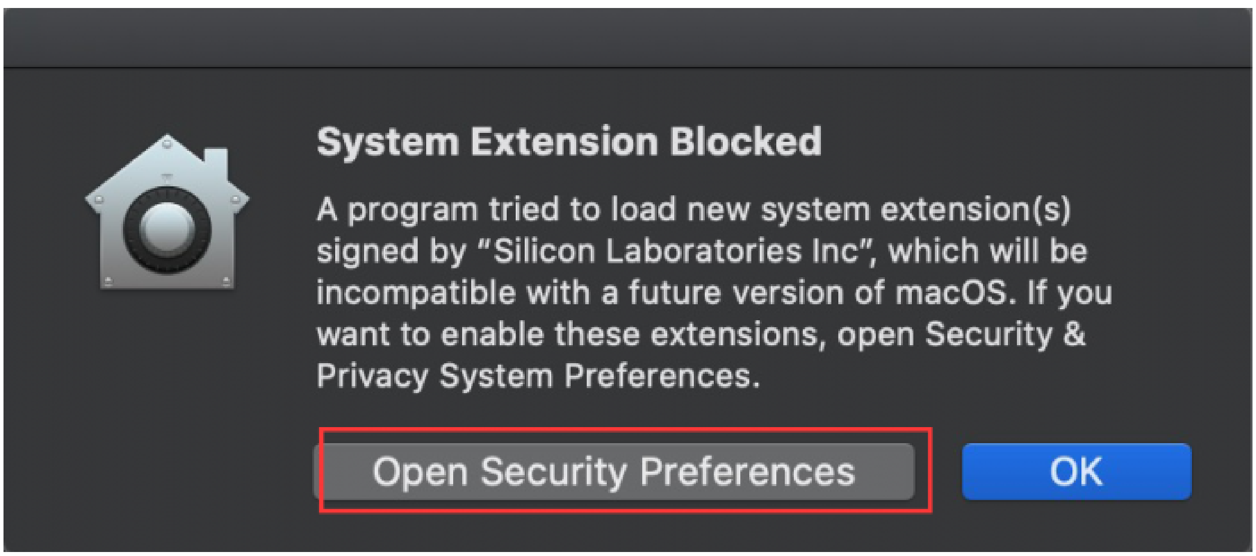


Click“**Continue**”then input your user password

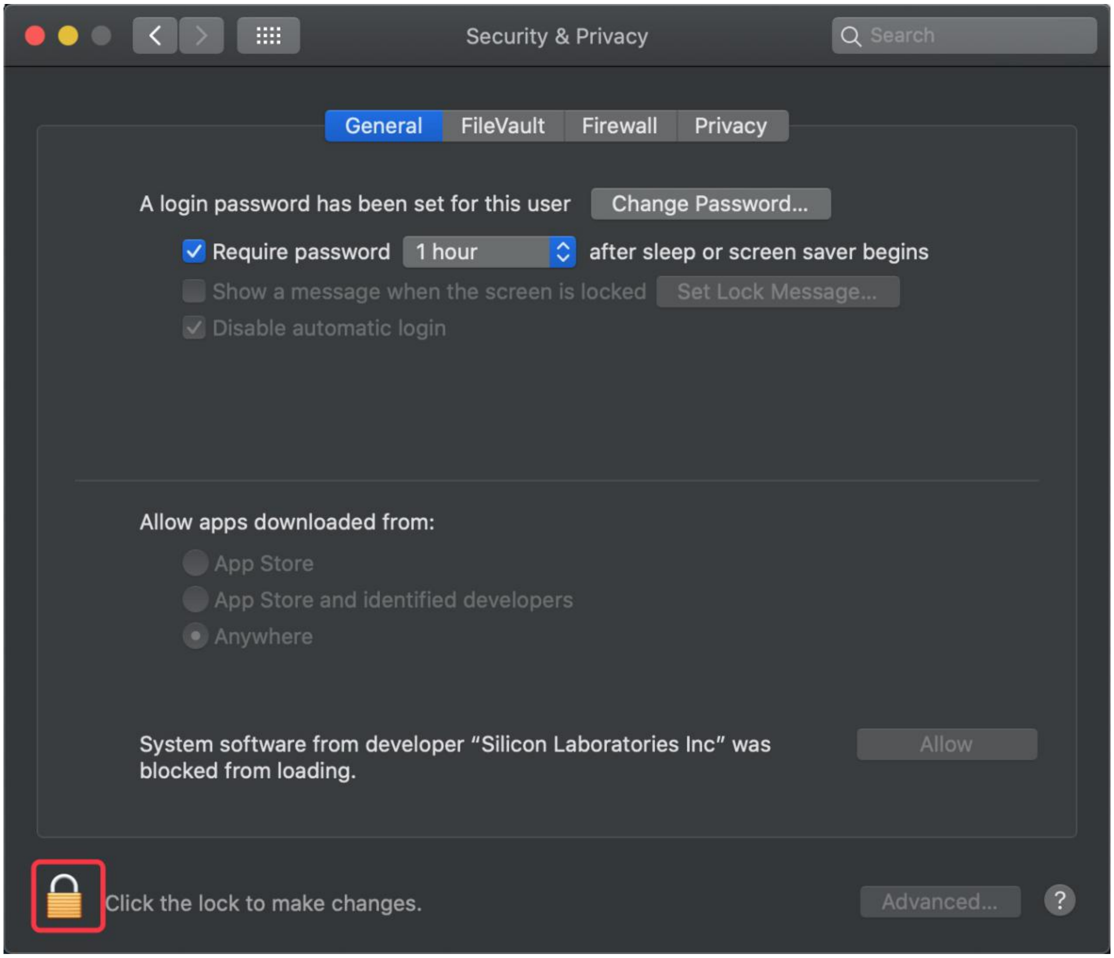


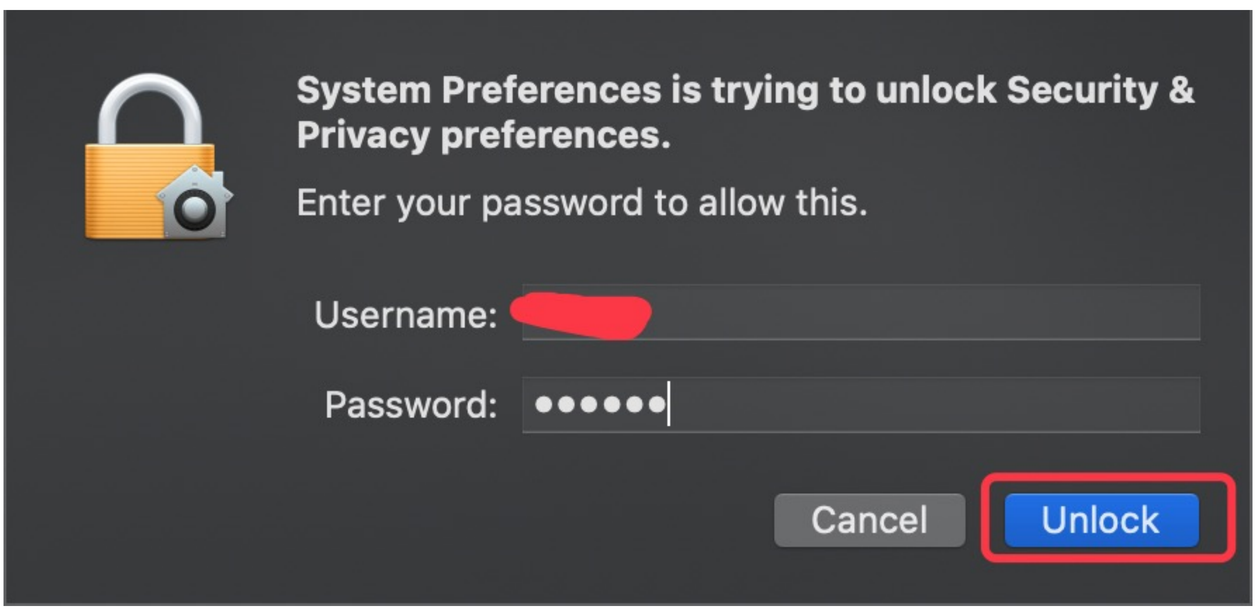


Select“**Select Open Security Preferences**”

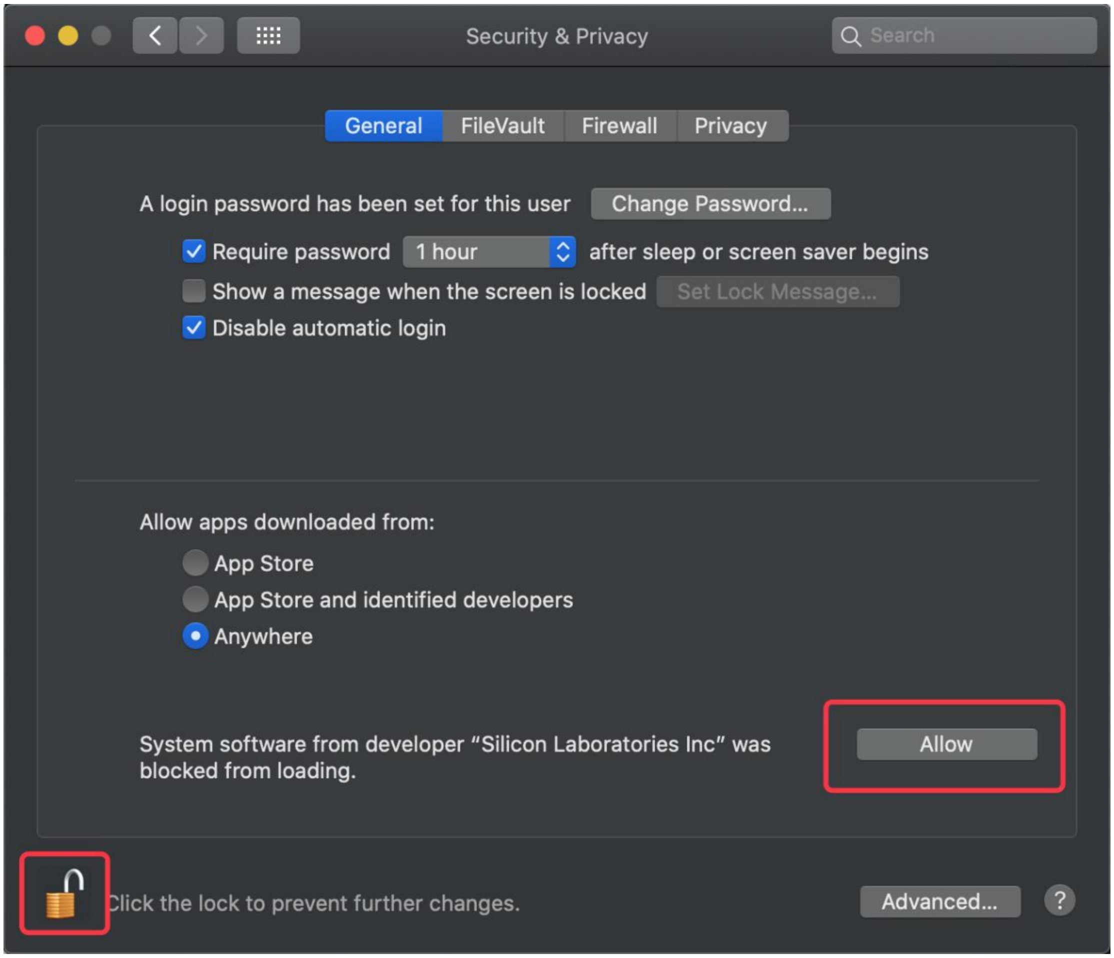


Click on security lock and enter your user password to authorize.

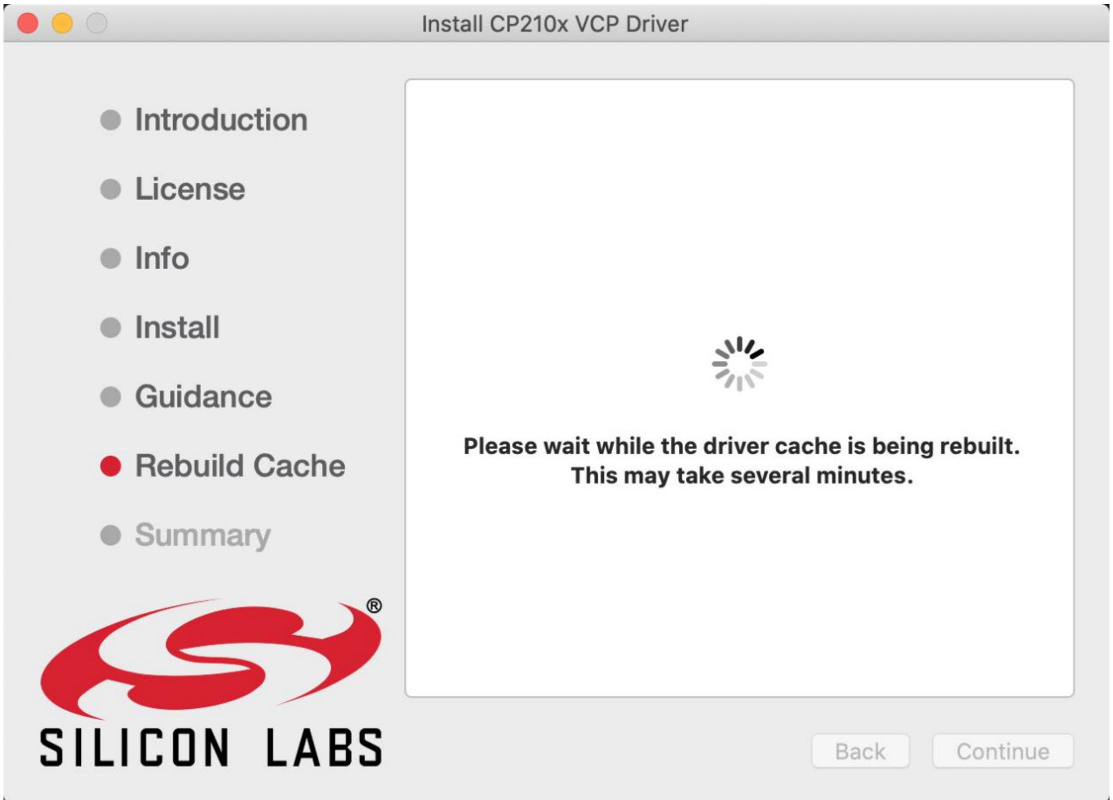




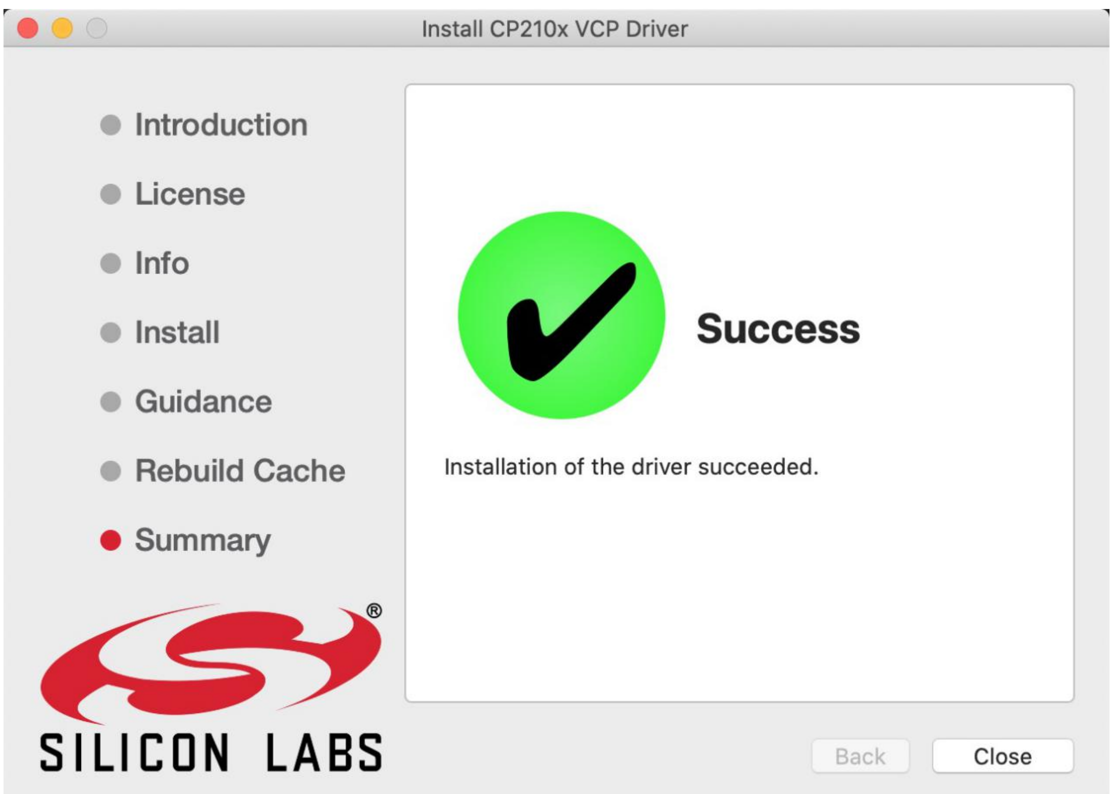
When you see that the lock is opened, click “Allow”.



Return to the installation interface and wait for the installation as prompted.



The installation is successful



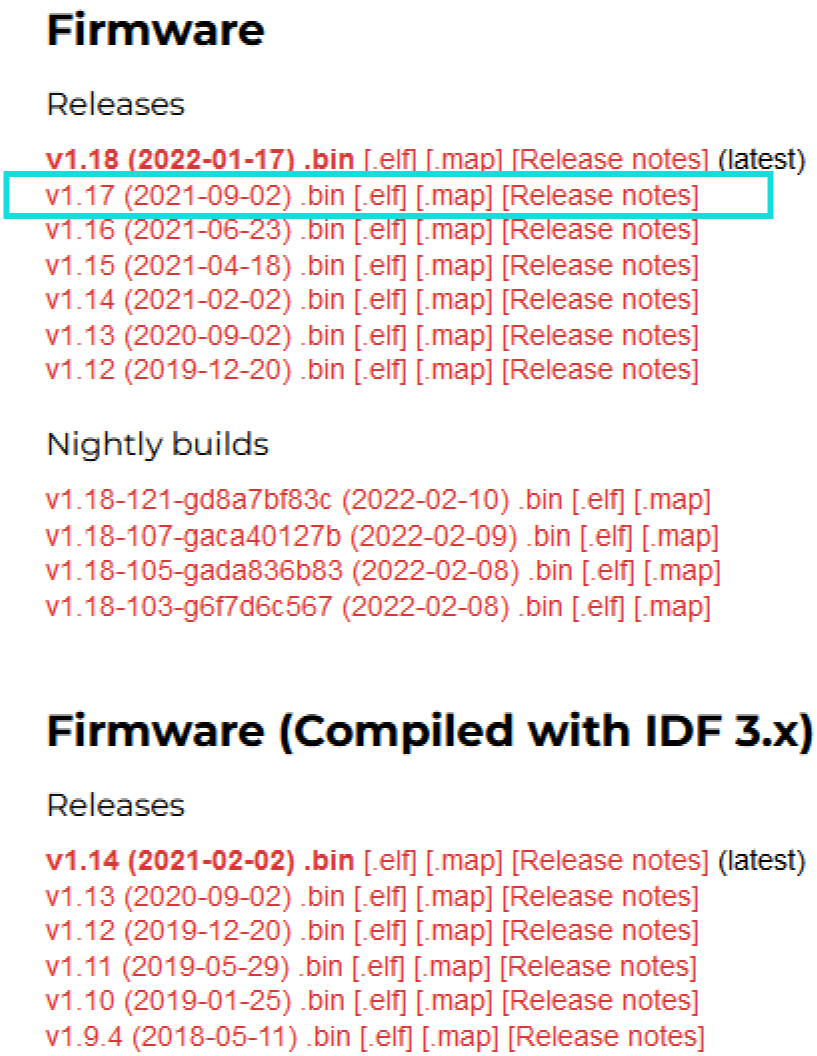
3.Burn Micropython firmware

To run a Python program on the ESP32 board, we need to burn the firmware to the ESP32 board first.

Download Micropython firmware

microPython websitehttp://micropython.org/

ESP32 firmwarehttps://micropython.org/download/esp32/

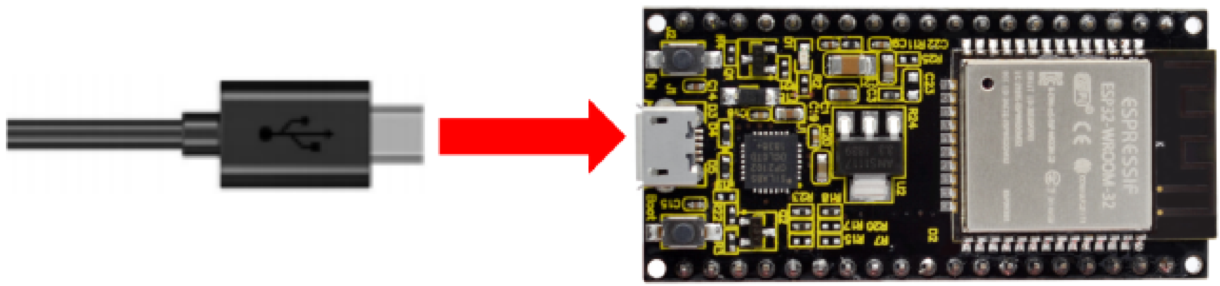


The firmware we use**esp32-20210902-v1.17.bin

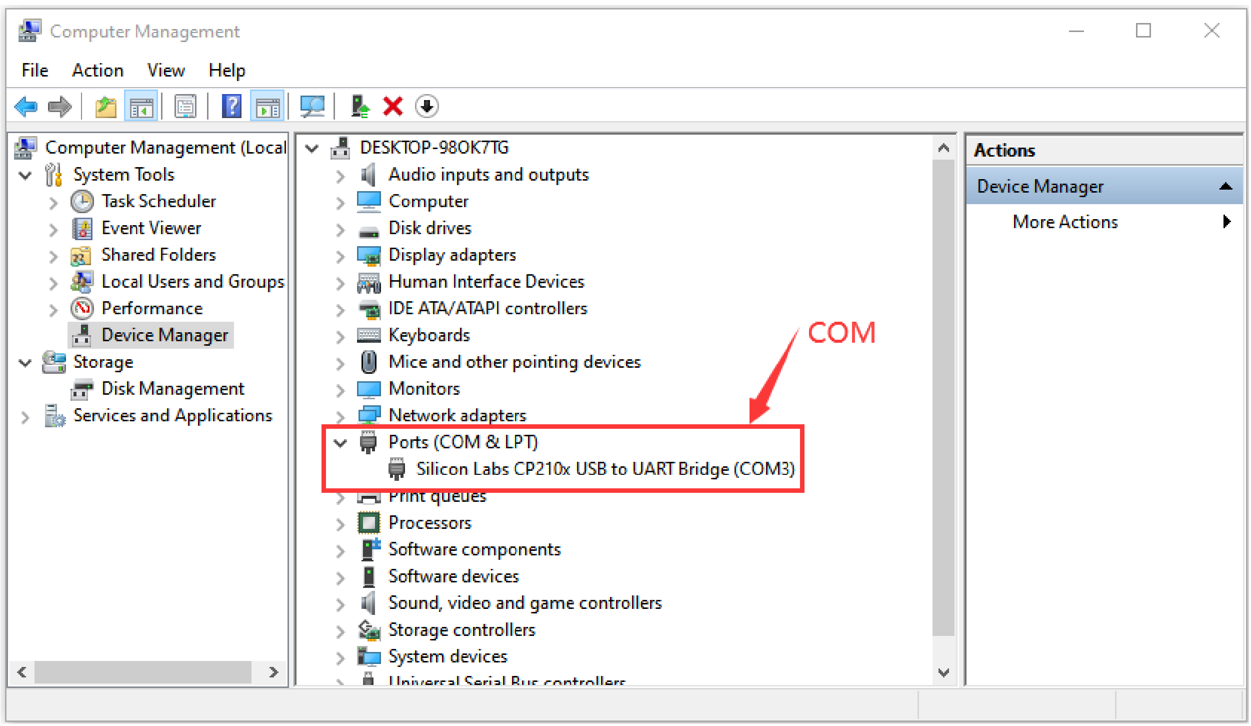
FirmwareDownload Python Firmware

Burn the Micropython firmware

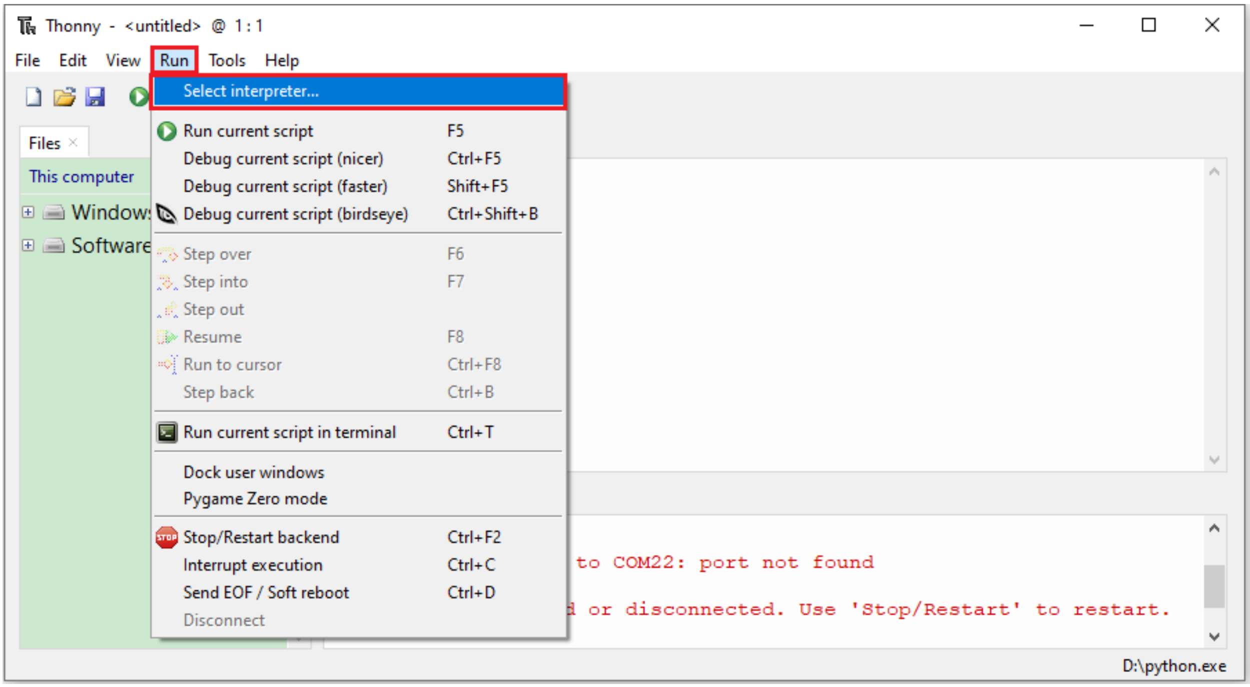
Connect the ESP32 to your PC with a USB cable



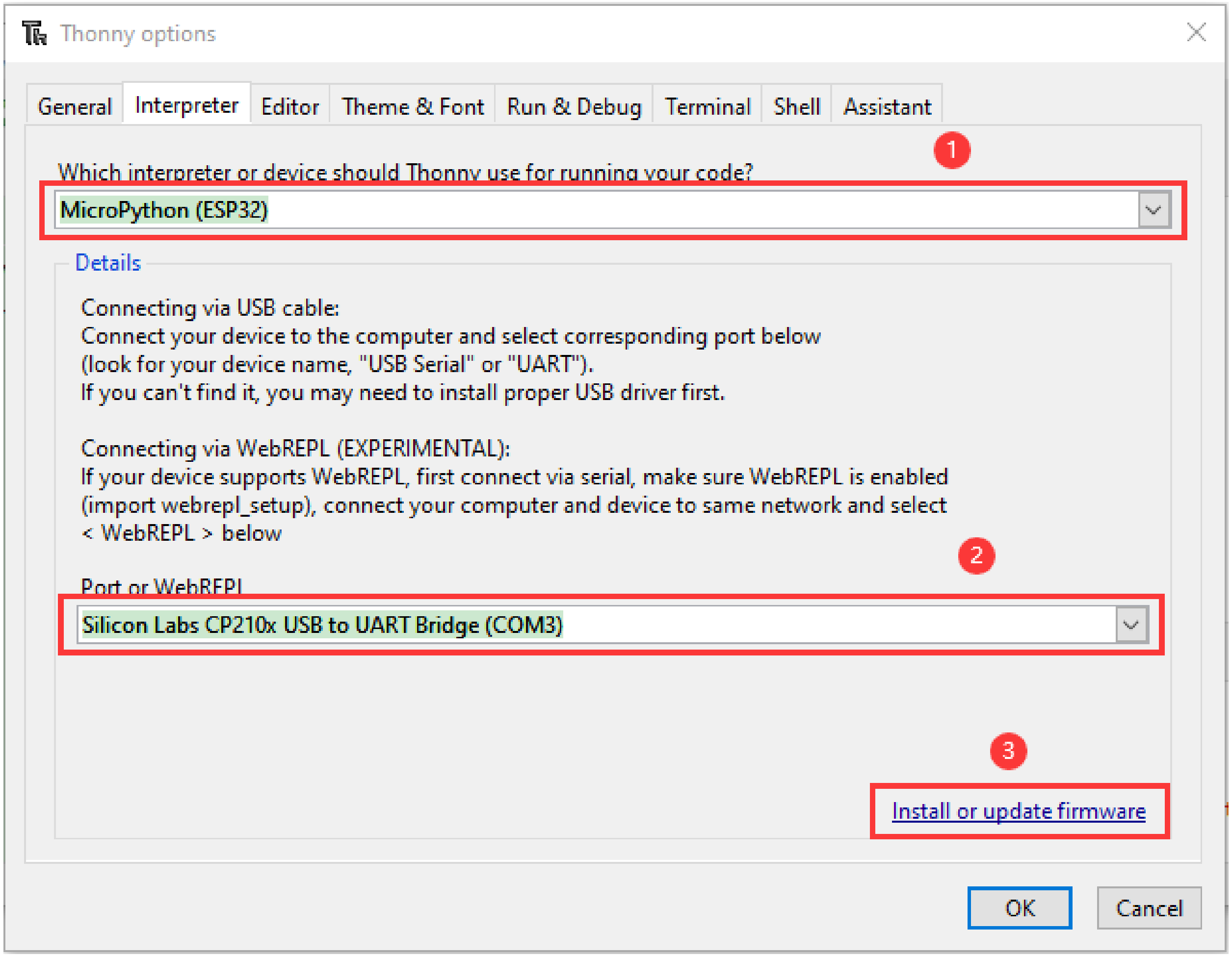
Make sure the driver has been installed successfully and the COM port can be identified correctly. Open Device Manager and expand "Ports".



Open Thonnyclick"run"and"Select interpreter. . . "



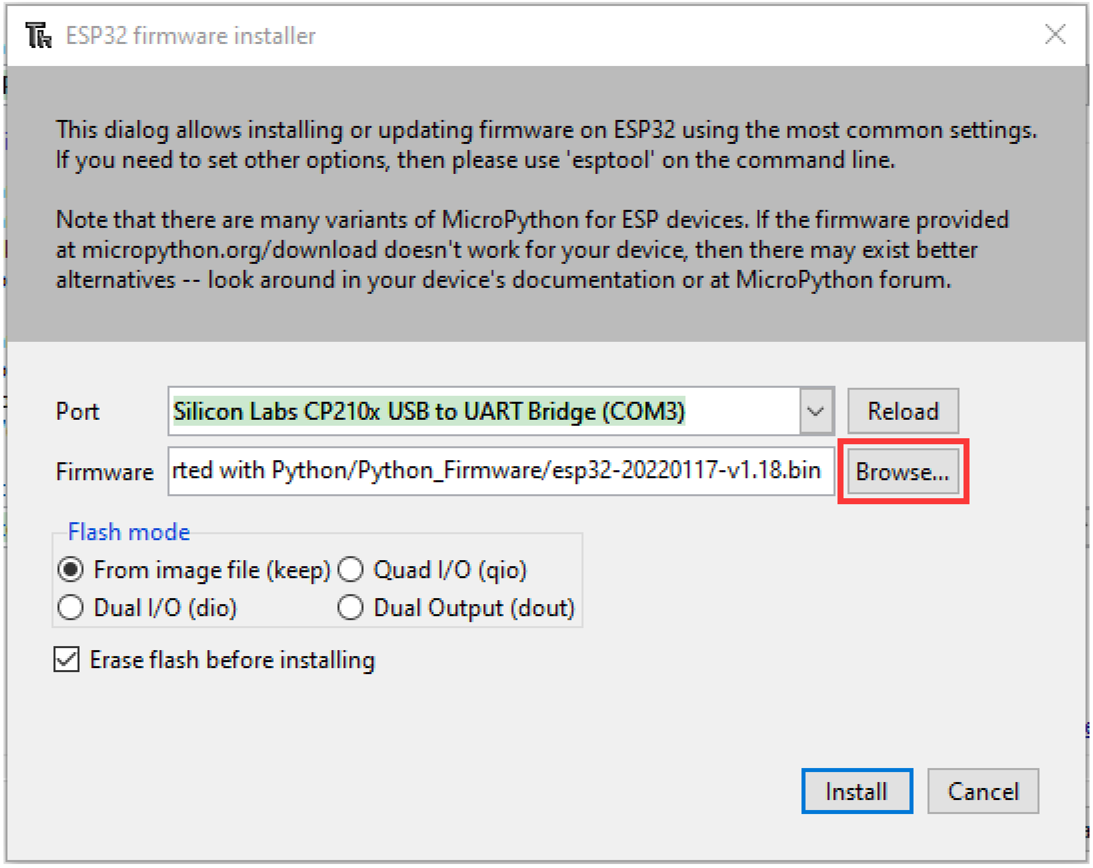
Select Micropython (ESP32) and Silicon Labs CP210x USB to UART Bridge(COM3) and click “Install or update firmware”.

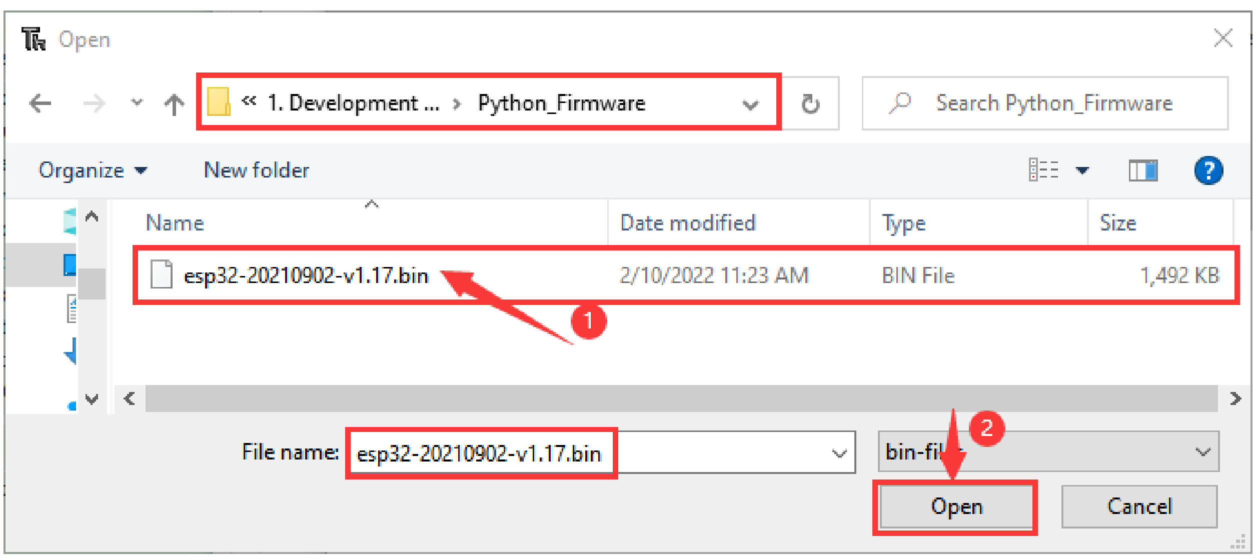


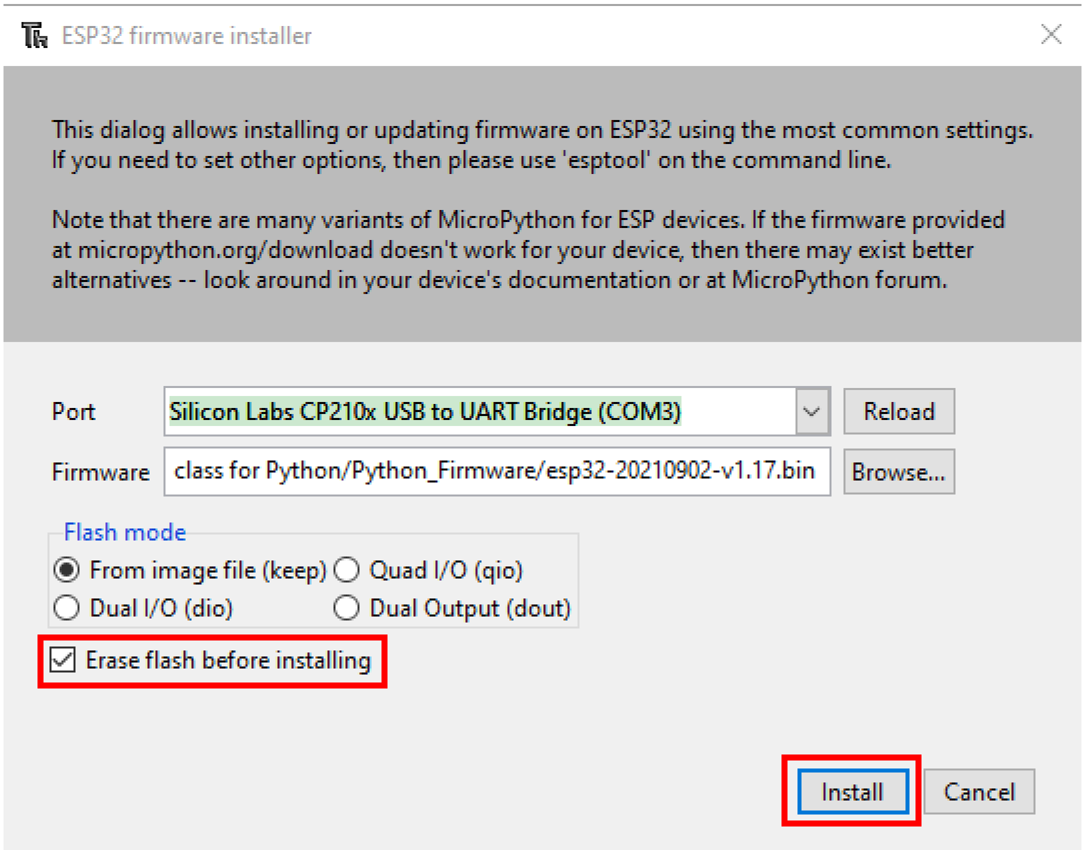
Select"Silicon Labs CP210x USB to UART Bridge(COM3)"click " Browse…"and choose the firmware **esp32-20210902-v1.17.bin.** Check"Erase flash before installing"and"Flash mode"then click"Install".

If you haven't downloaded the firmware, please click on the link to downloadDownload Python Firmware

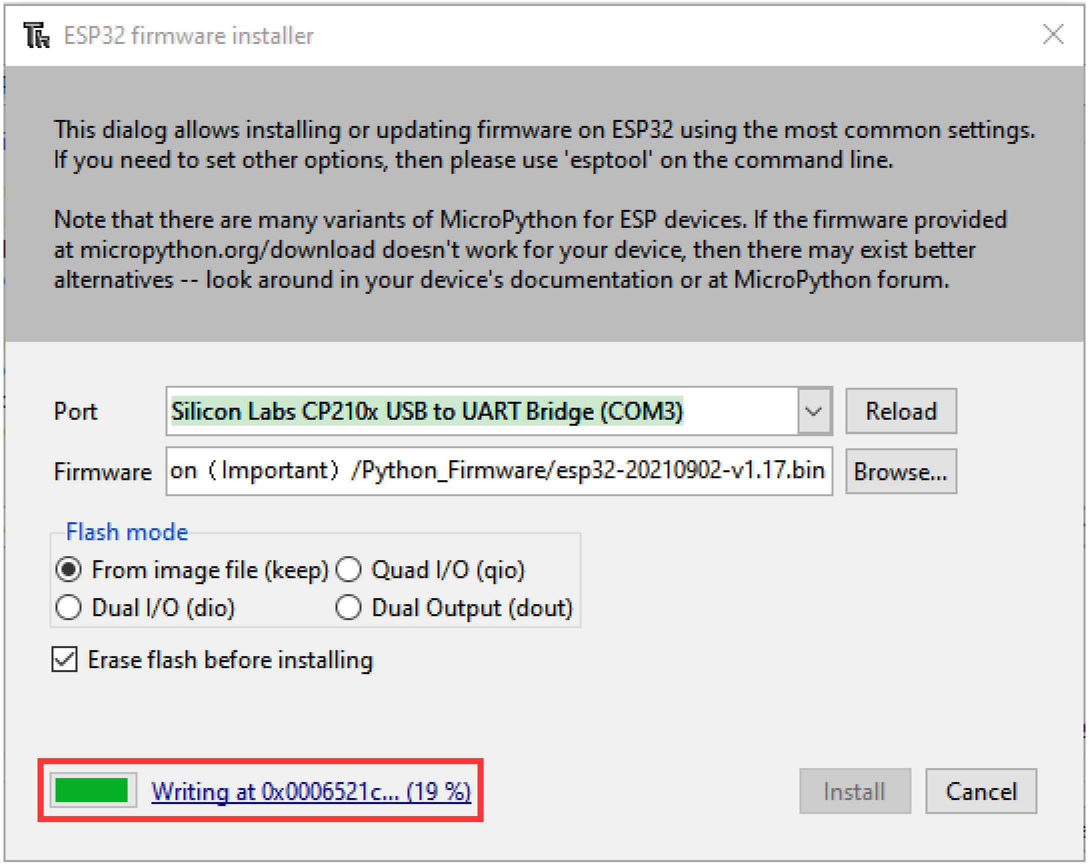
(NoteIf you fail to install the firmwarepress the Boot button on the ESP32 board and click"Install"

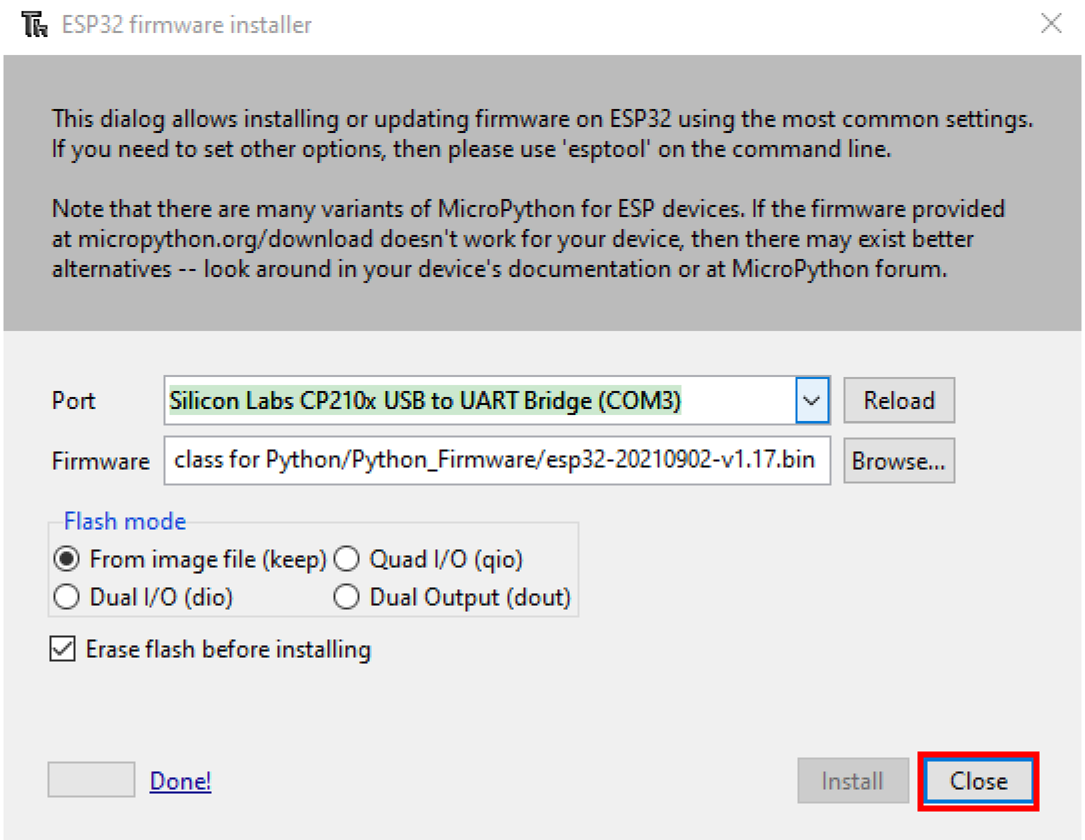

ESP32 firmware installer
This dialog allows installing or updating firmware on ESP32 using the most common settings.
If you need to set other options, then please use 'esptool' on the command line.
Note that there are many variants of MicroPython for ESP devices. If the firmware provided at micropython.org/download 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

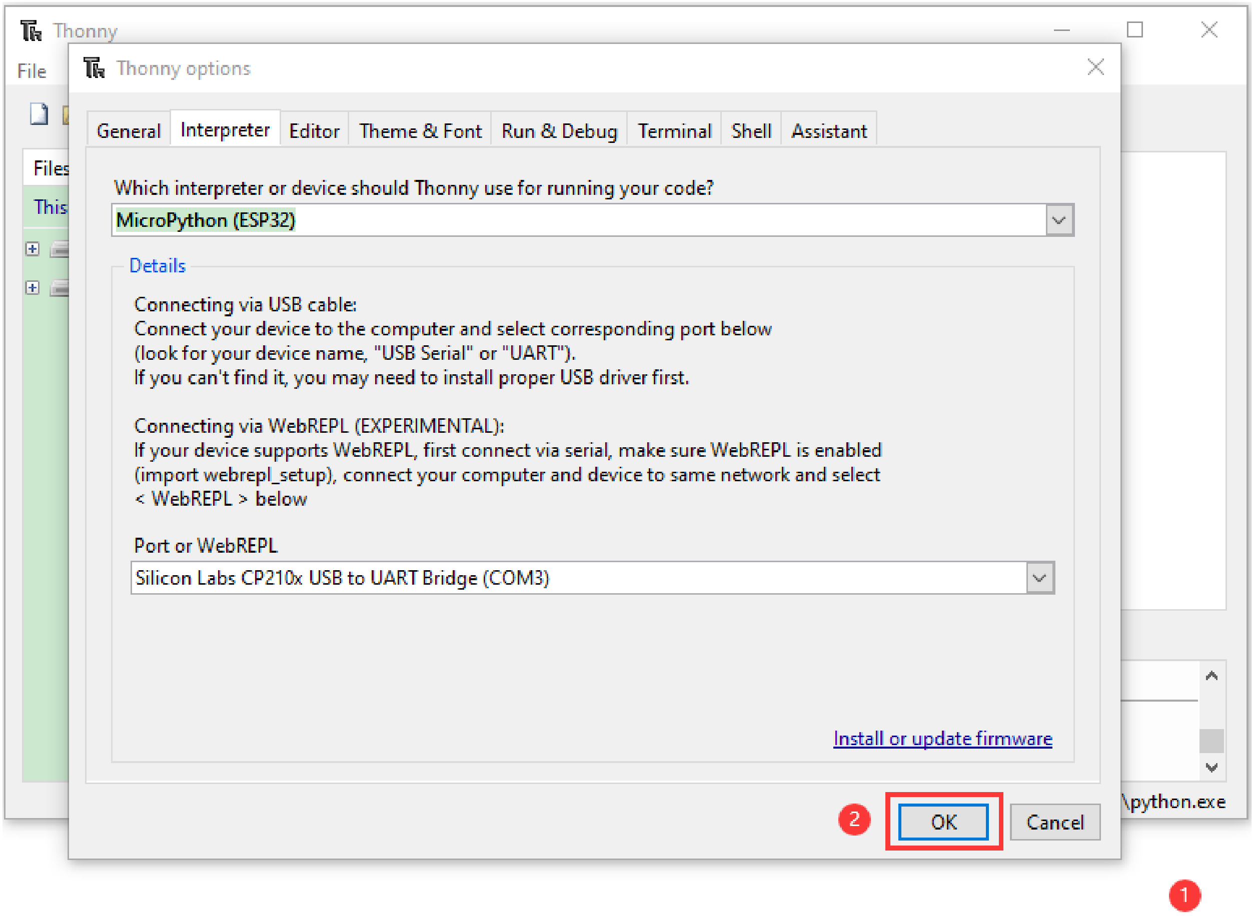

Open
« 1. Development ... › Python_Firmware
Search Python_Firmware
Organize
New folder
Name
Date modified
Type
Size
esp32-20210902-v1.17.bin
2/10/2022 11:23 AM
BIN File
1,492 KB
1
File name:
esp32-20210902-v1.17.bin
bin-fil
2
Open
Cancel



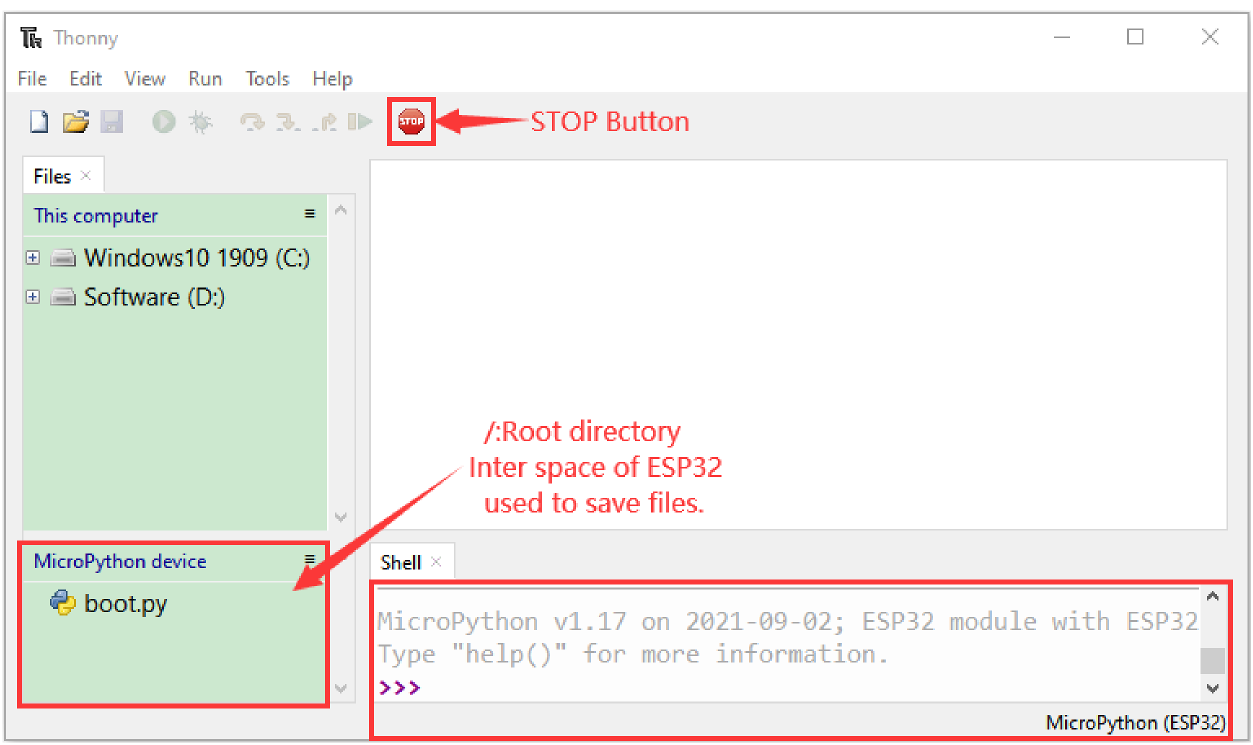
Then click Close and OK


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
class for Python/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
Done!
Install
Close



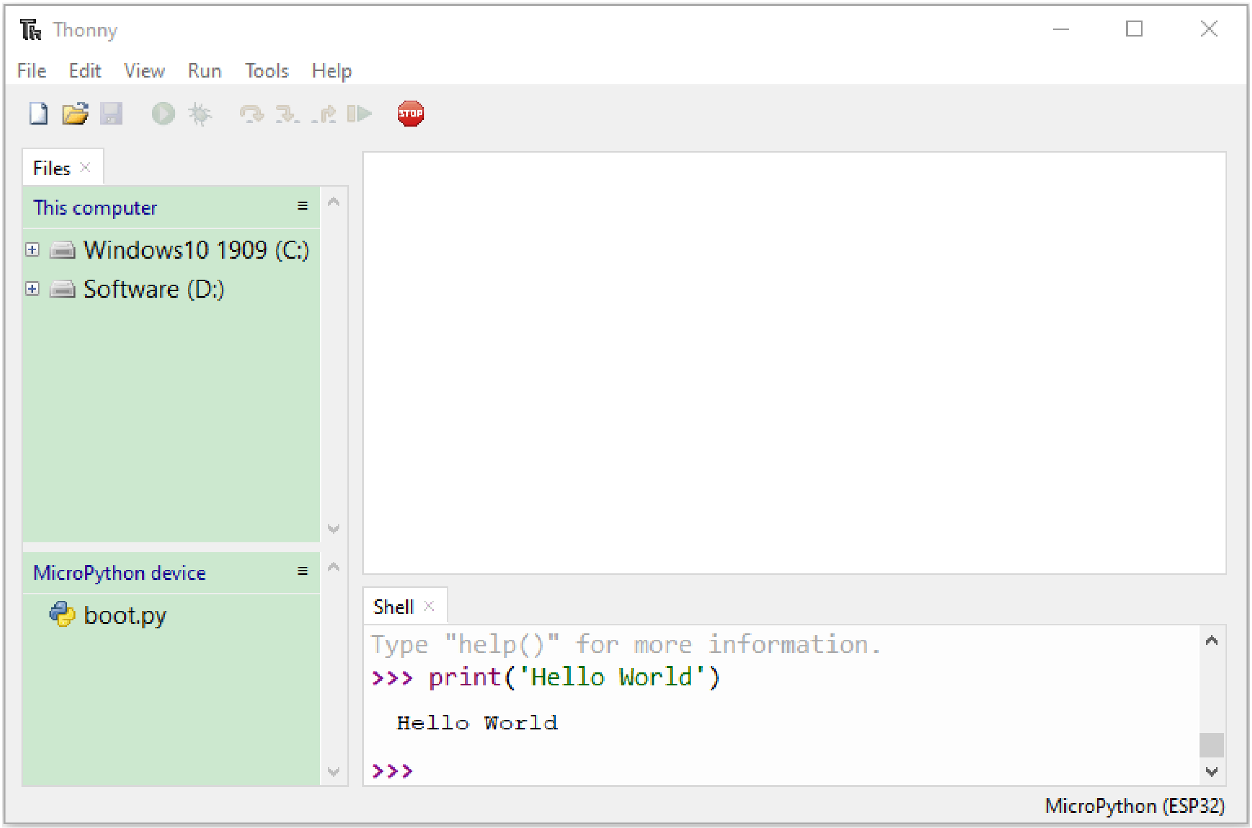
Turn off all windows and turn to the main page and click "STOP.



Test Code

Test the Shell commander

Input print('hello world') in the"Shell"and press **Enter**

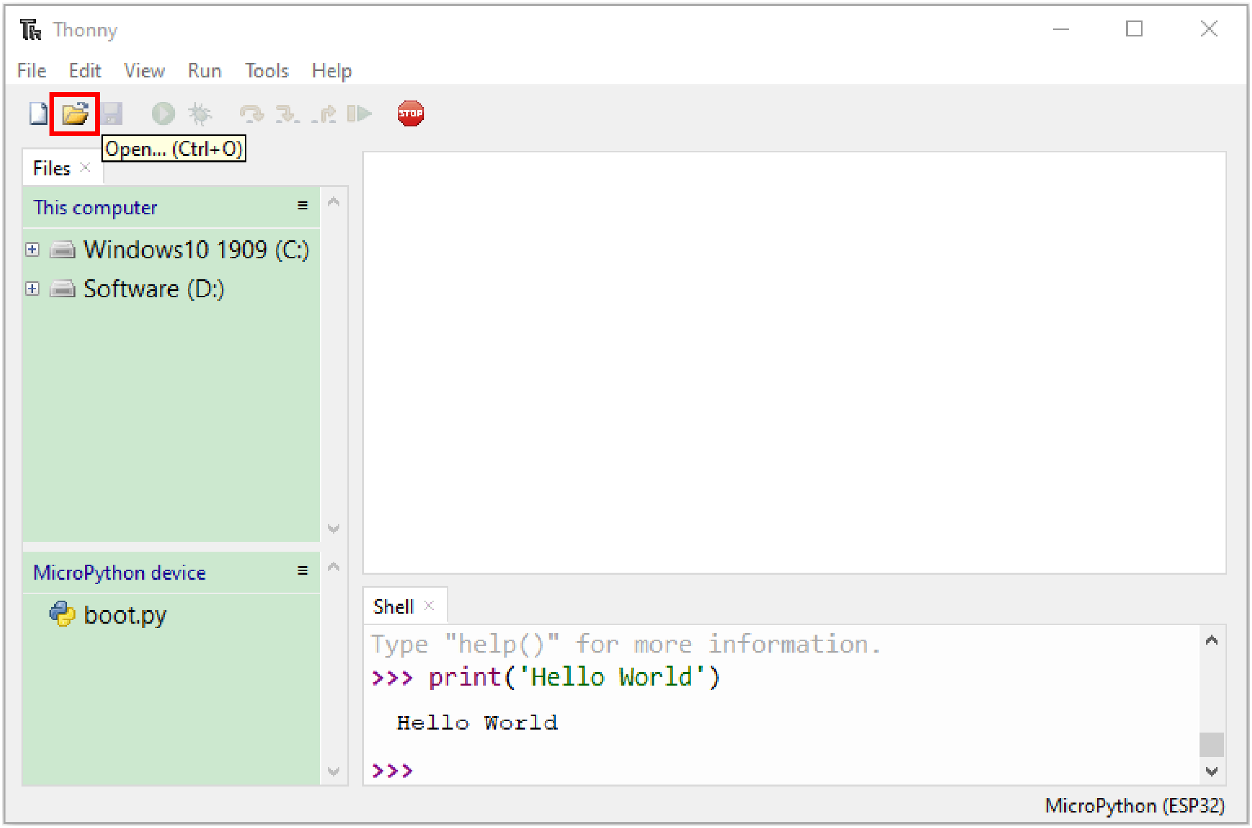


Run the test code(online)

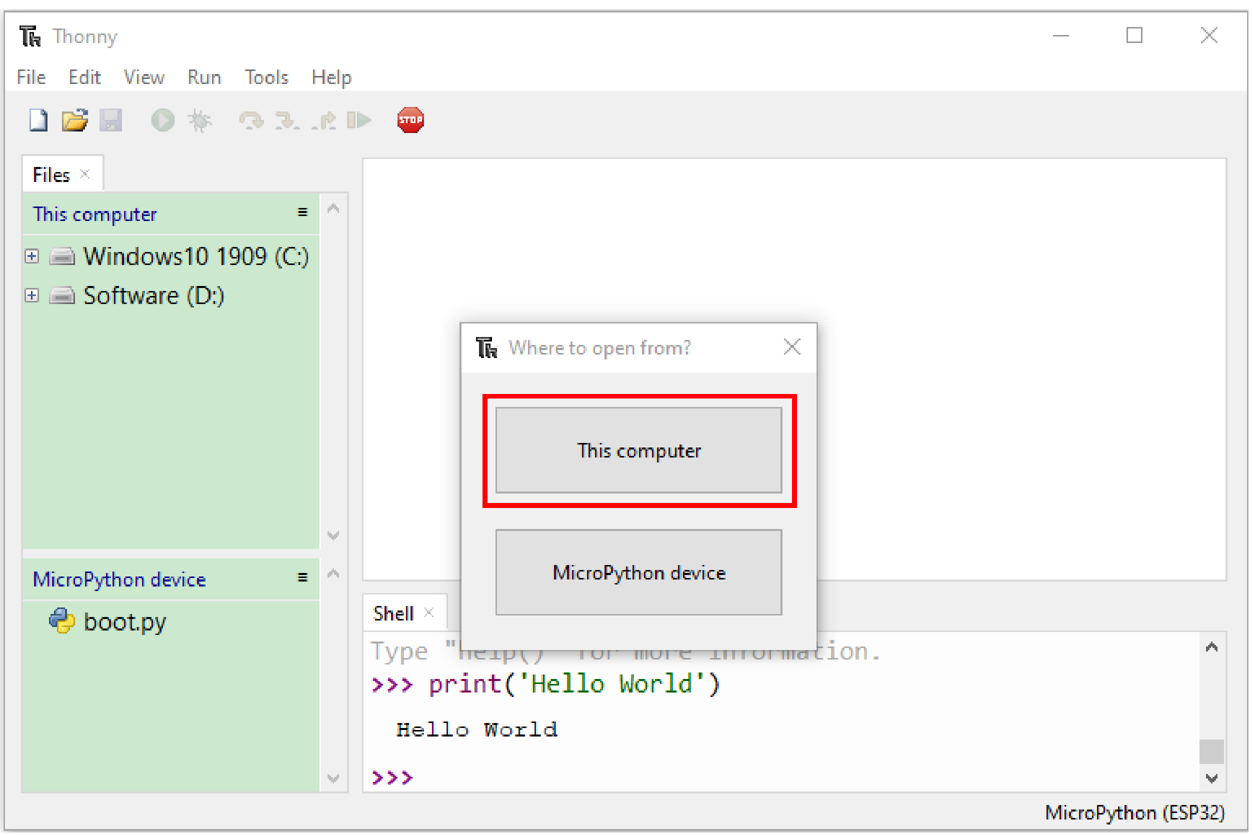
Connect the ESP32 to your PC. Users can program and debug programs with Thonny.

Open Thonny and click Open.

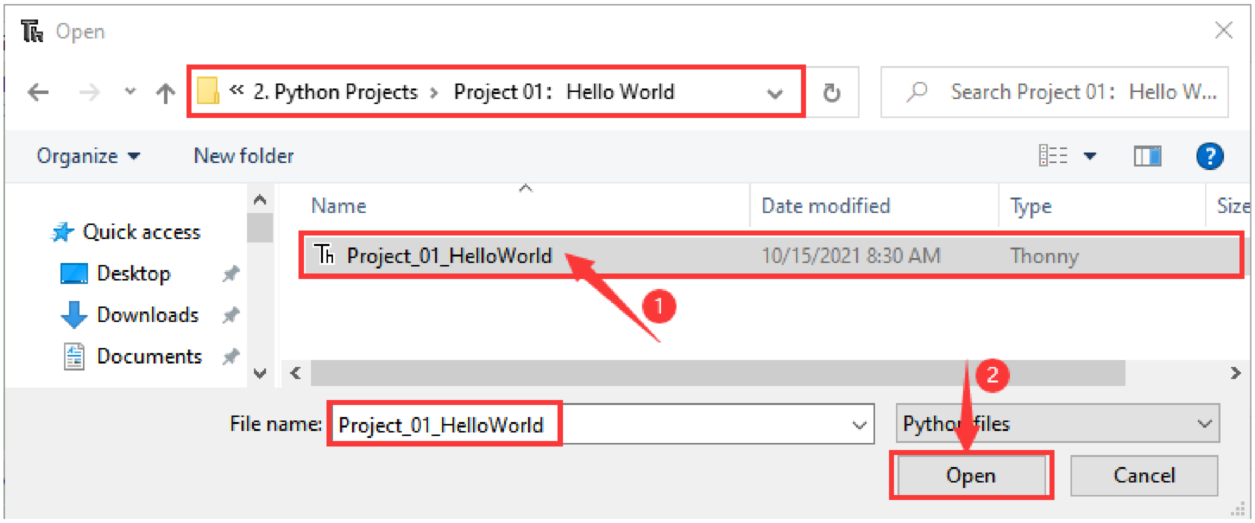
If you haven't downloaded the code file yet, please click on the link to download it:Download Python Codes



When a new window pops up, click“This computer”



Select the file“Project_01_HelloWorld.py”



Click ,“Hello World”will be printed in the“Shell”monitor.

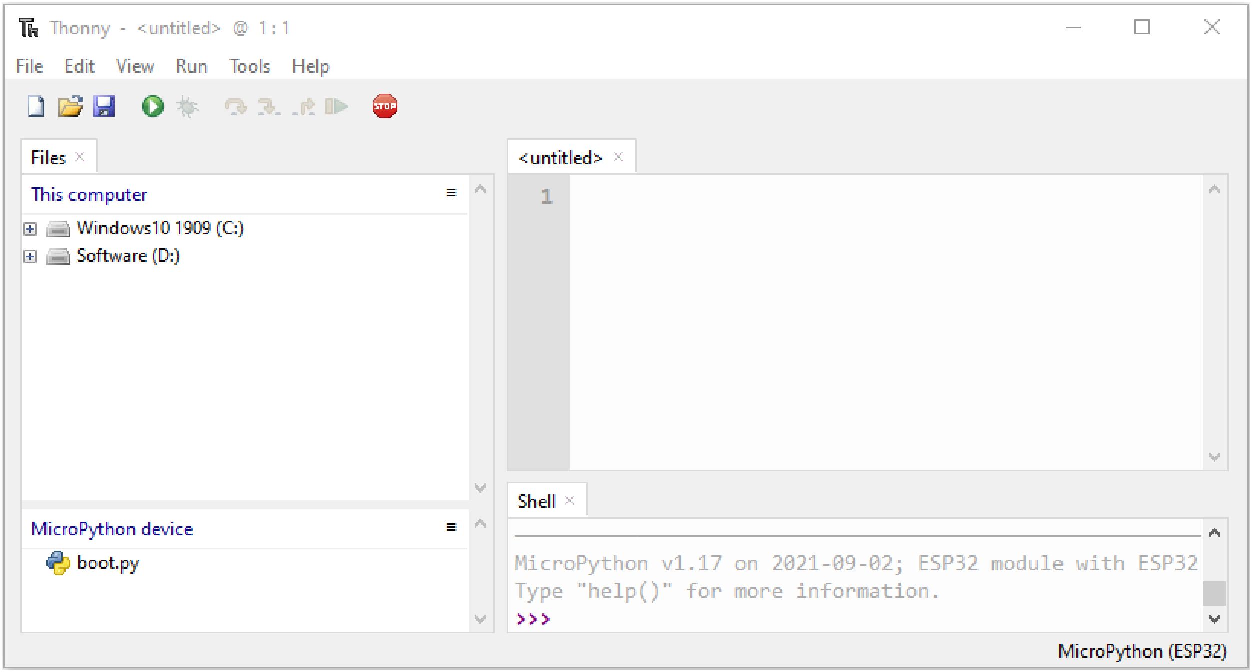
Note: Press the reset button to reboot

Run the test code(offline)

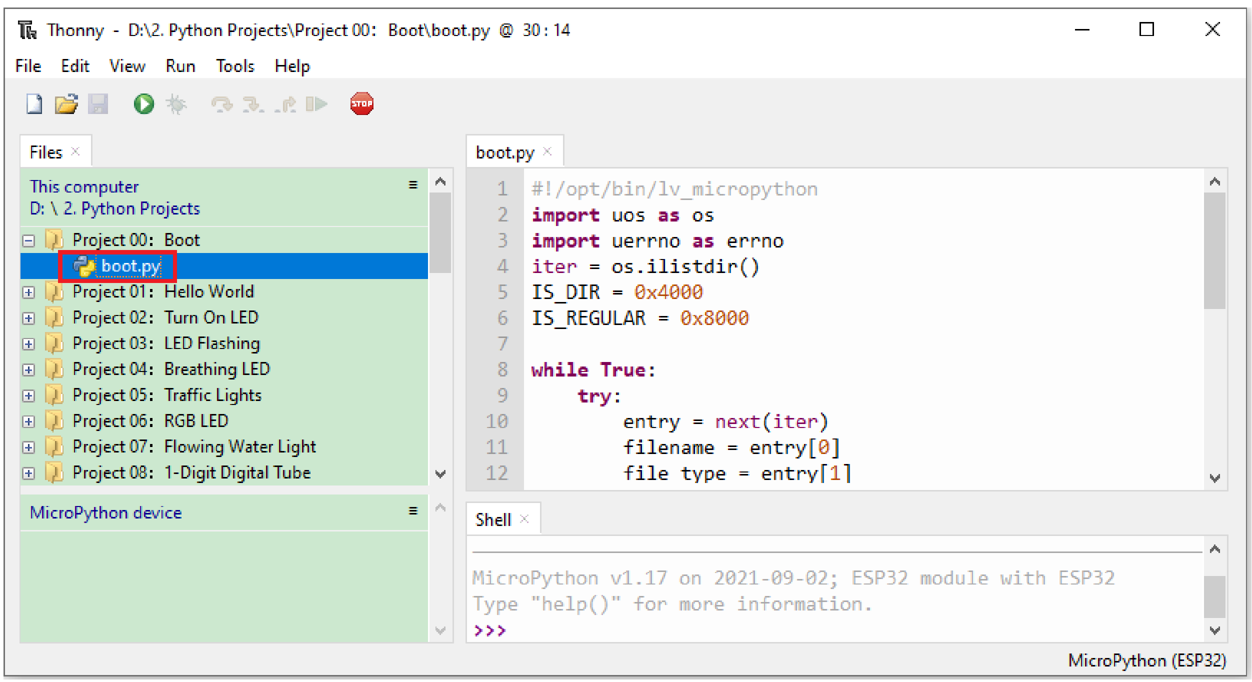
After rebooting the ESP32, run the boot.py file under the root directory first then run your code file.

So, we need to add a guide program to run the code of users.

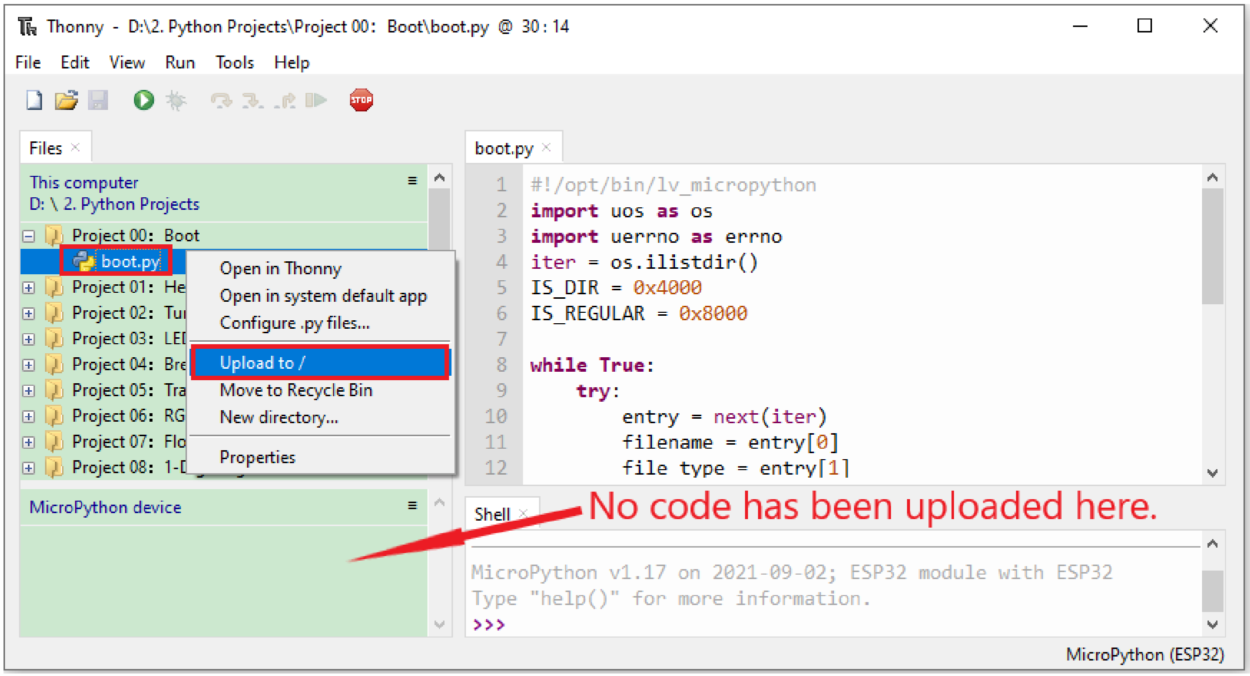
Move the file“4. Python Tutorial\2. Python Projects” to the disk(D)the route is“D:/2. Python Projects”, then open the “Thonny”.



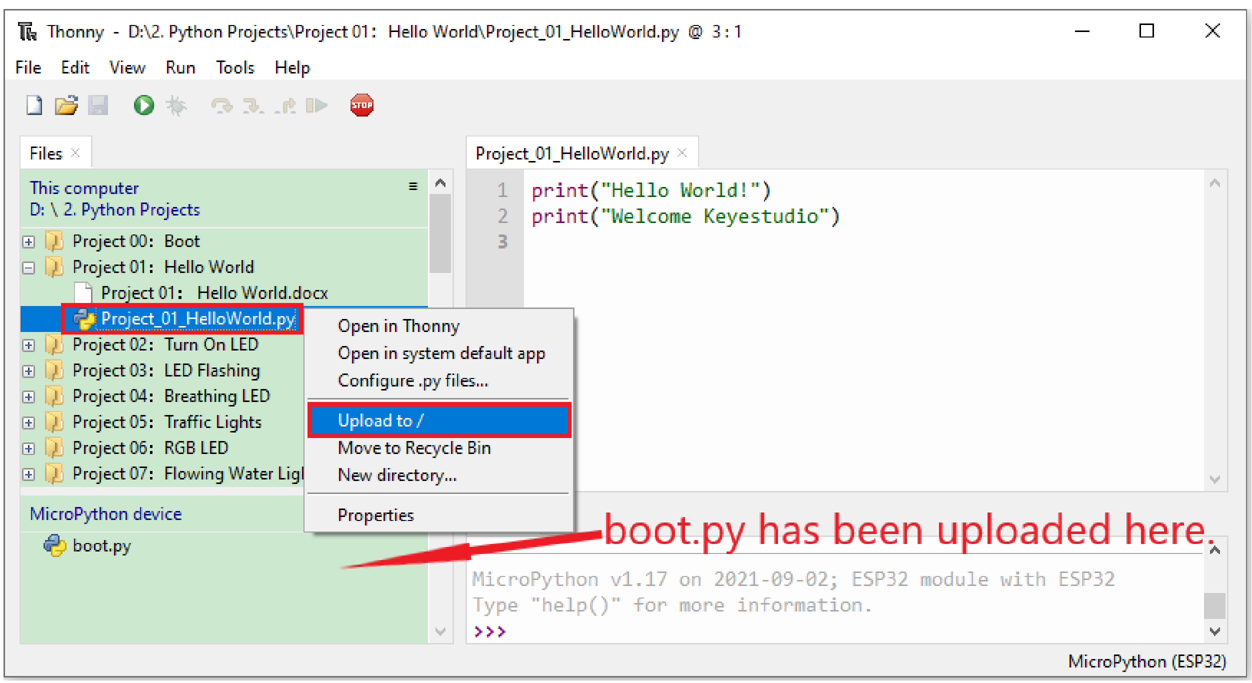
Click project 00. Boot.Boot and double-click boot.py, then the code under MicroPython device can run offline.



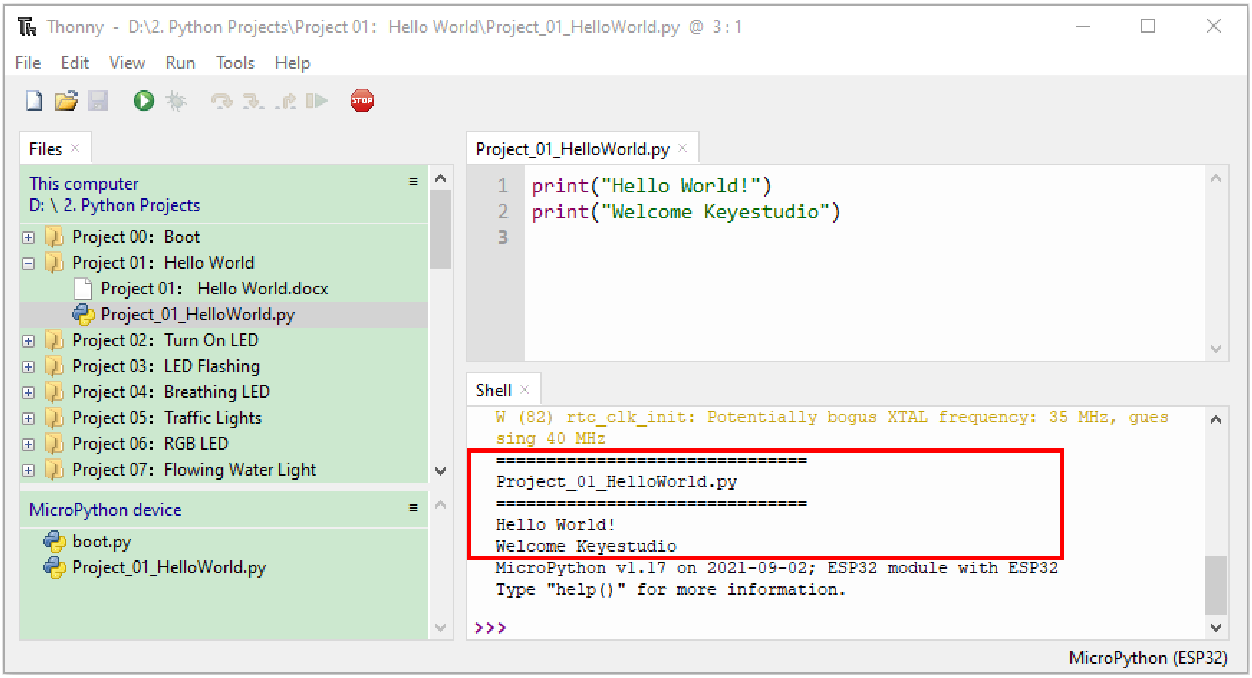
If you want to run the code offline, you nee to upload boot.py and program code to MicroPython device, then press the ESP32's reset button. We will take the project 00 and project 01 as an example. Select boot.py and right-click Upload to /.



Similarly, upload the project_01_Helloworld. py file to the "MicroPython Device".

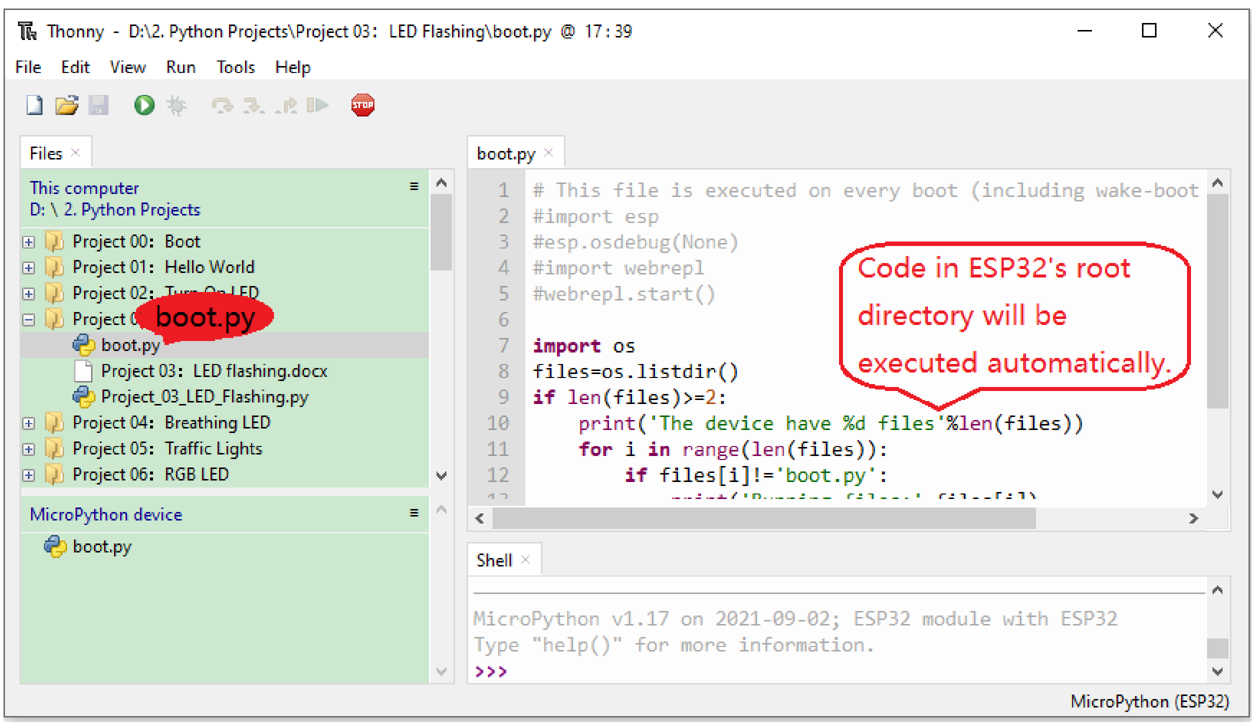


Press the Reset button, you will view code running in the Shell monitor

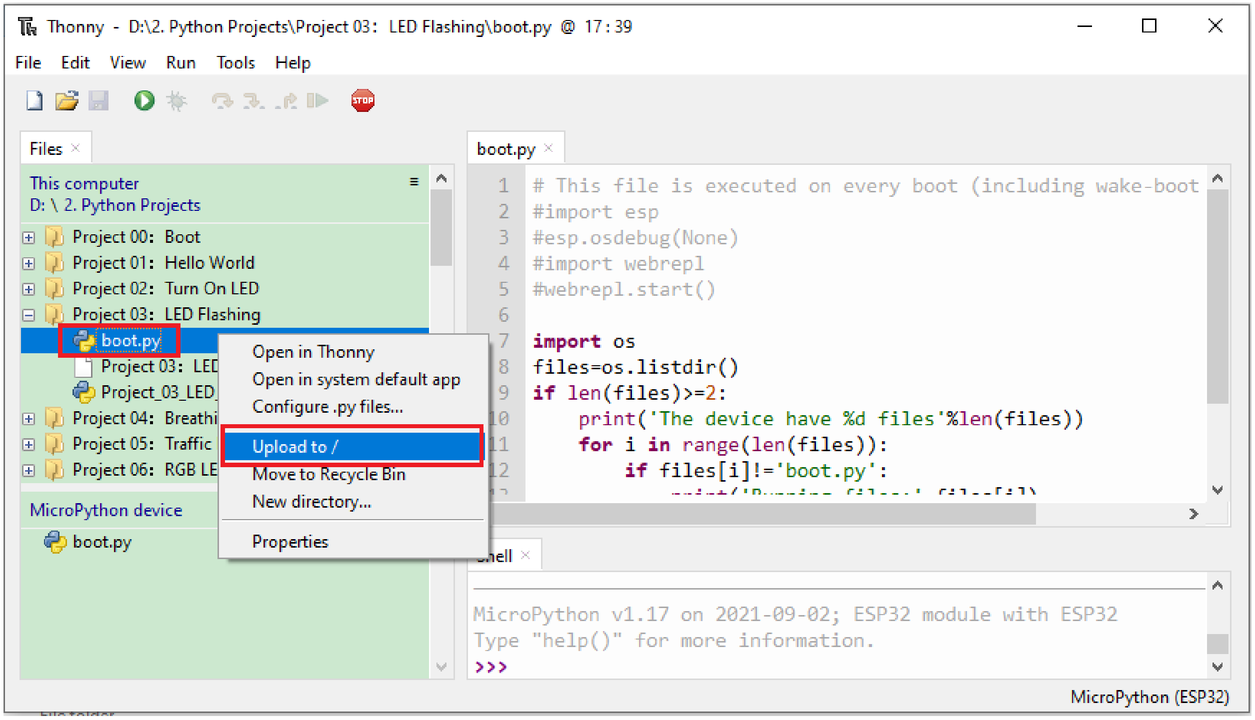


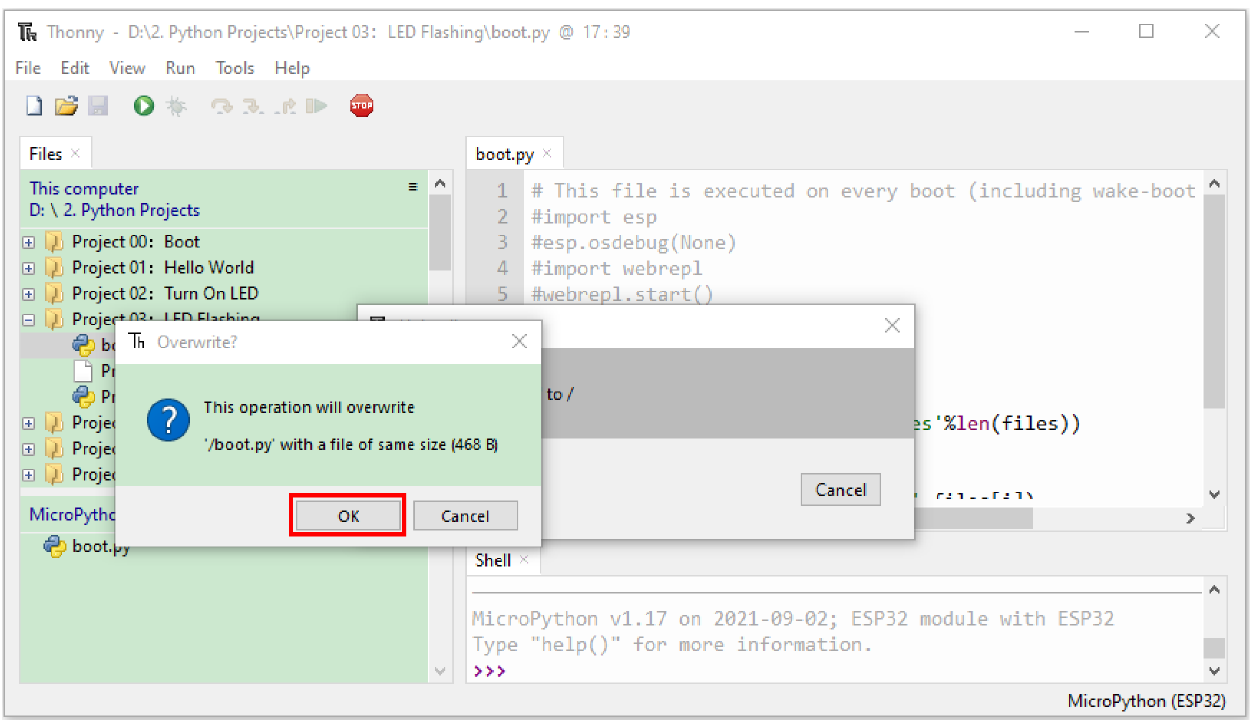
Upload the code to the ESP32

We take the boot.py as an example. If we add a boot.py in each code directory, reboot the ESP32, the boot.py will run first.



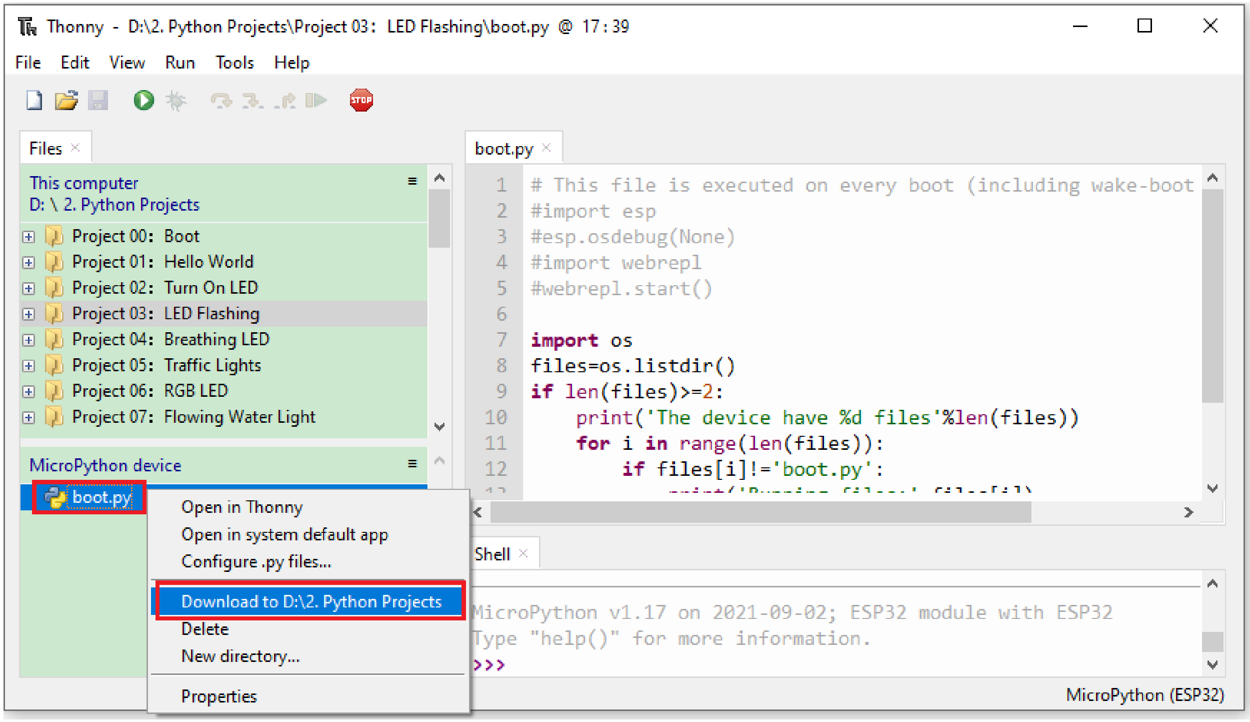
Select "boot.py"in the file Project 03LED Flashing, right-click to select"Upload to /". Then the code will be uploaded to the root directory of the ESP32 and click OK.





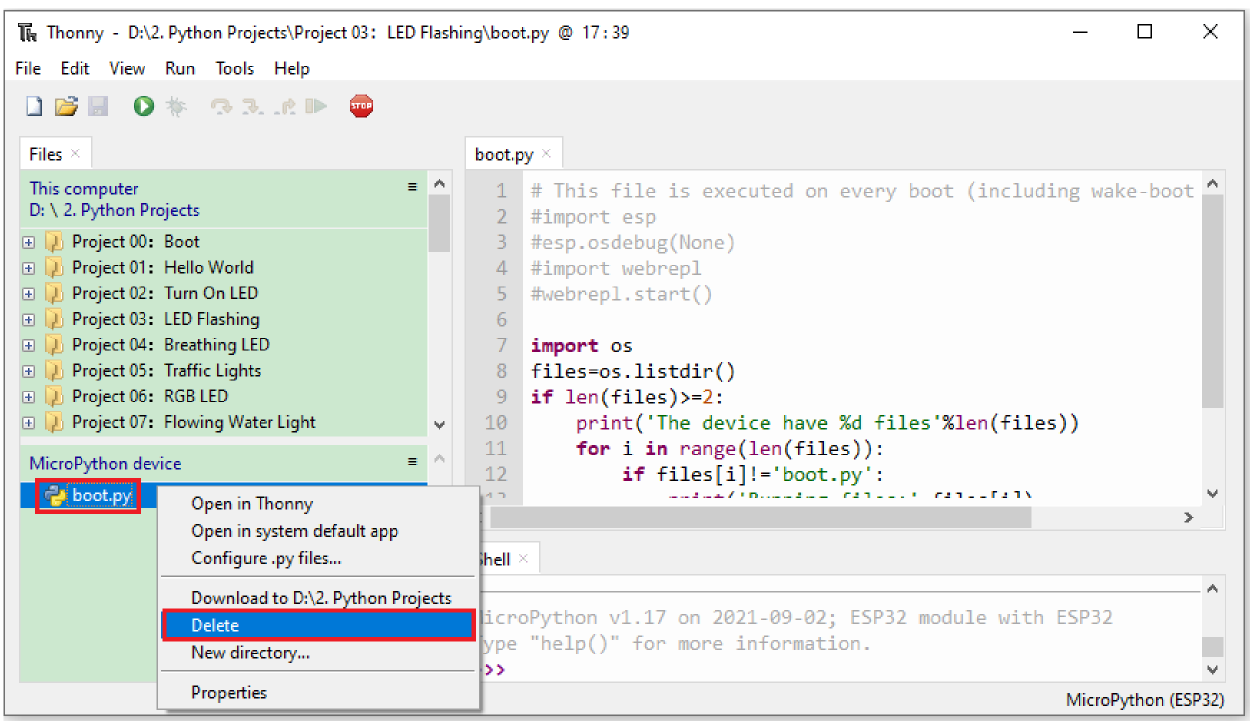
Download the code to your PC:

MicroPython device<boot.py, then right-click Download to…

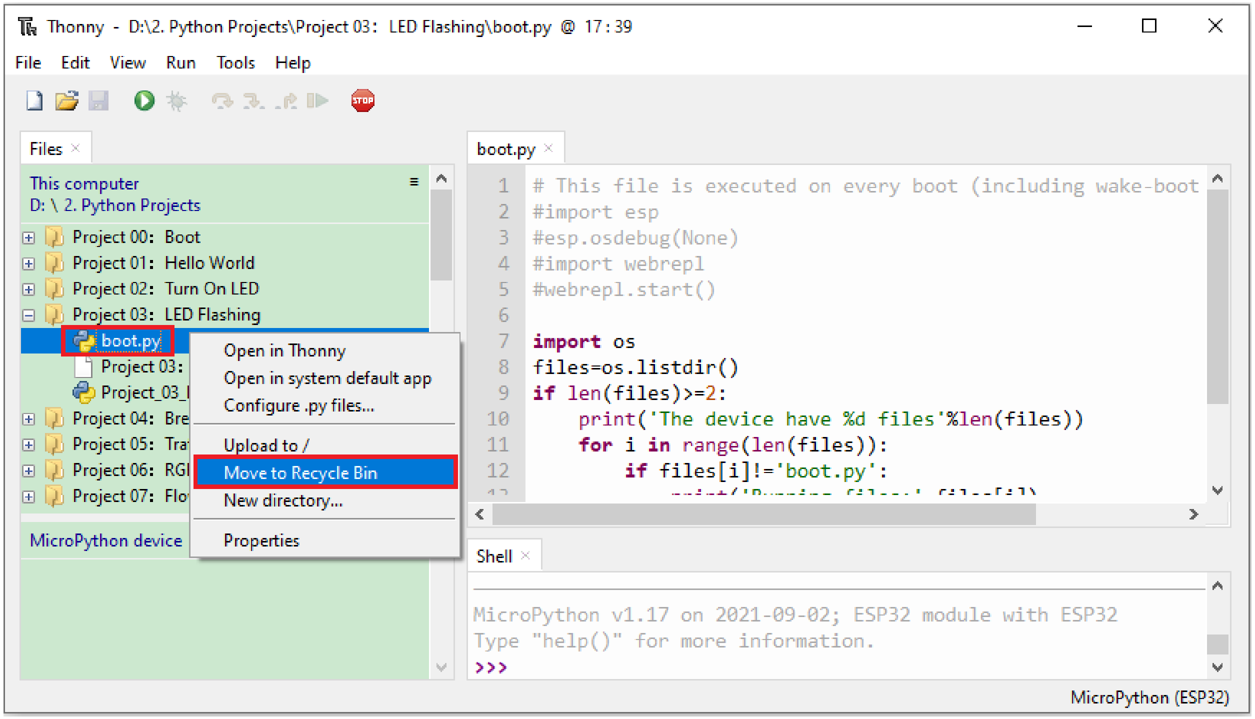


Delete files of the ESP32

For example, click“boot.py”in the MicroPython device and right-click Delete.

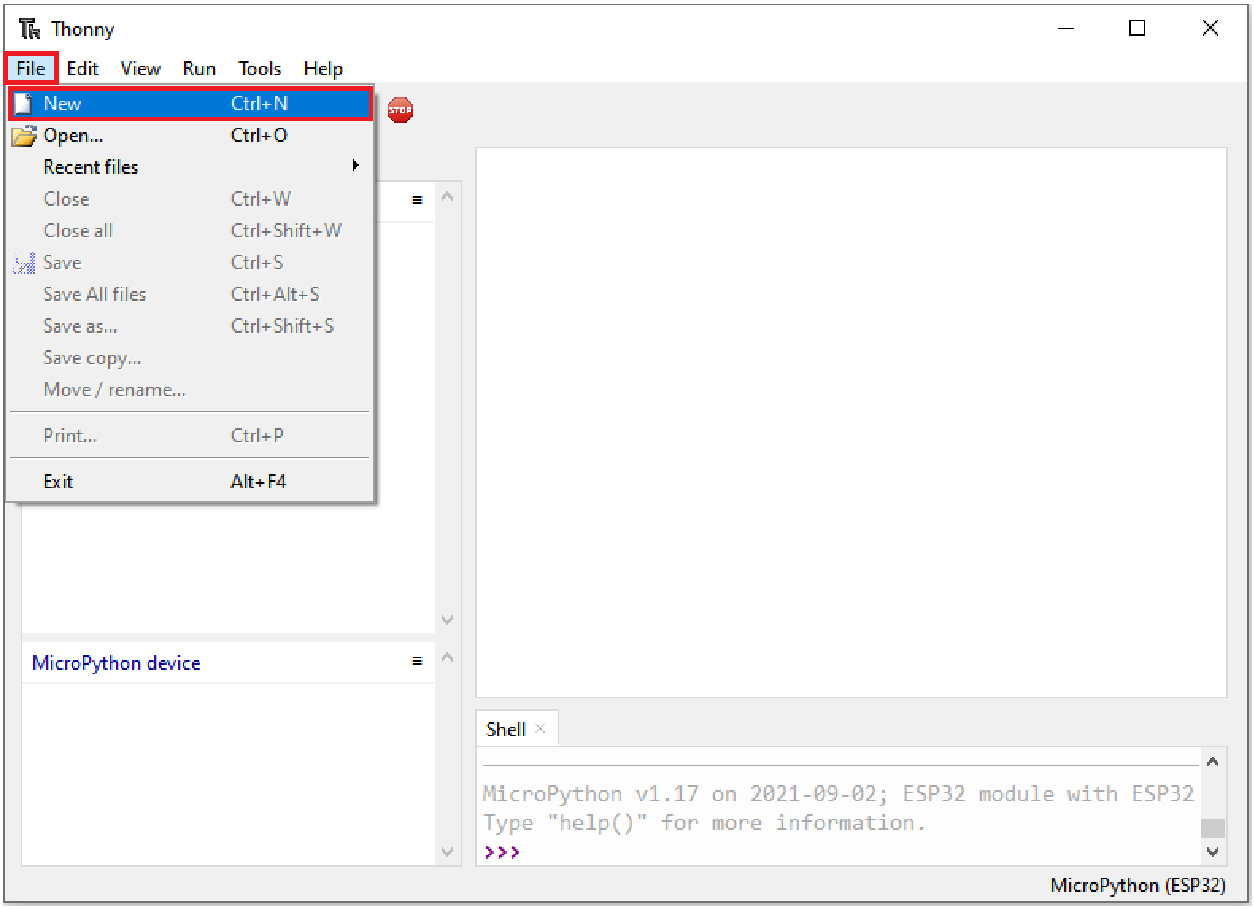


Select boot.py in the Project 03LED Flashing folder, right-click Move to Recycle Bin to delete it.

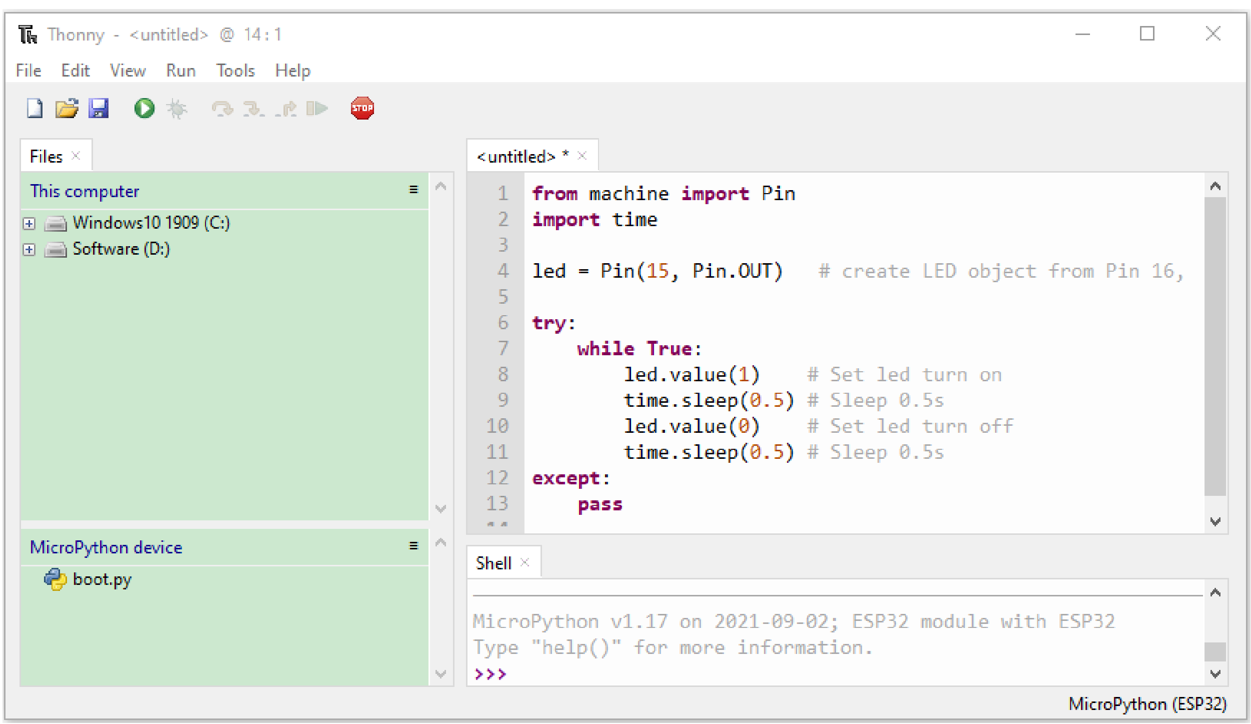


Create and save code

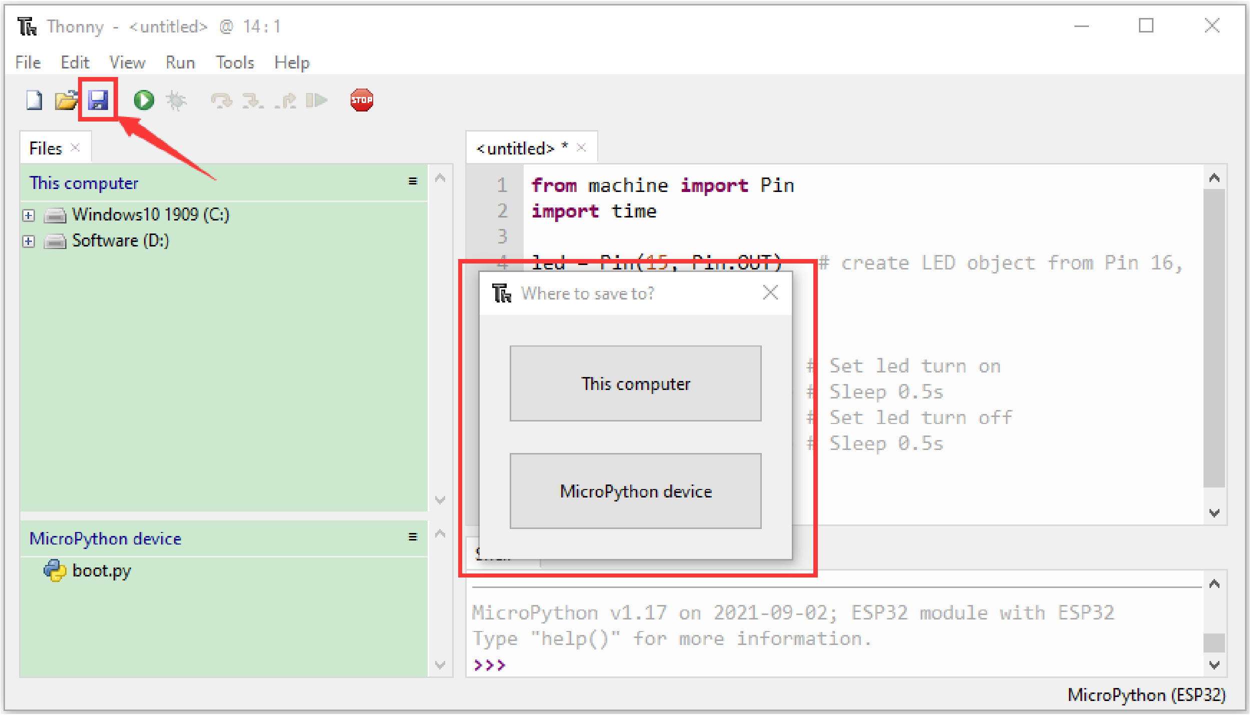
Click“File”→“New”to create and edit code.



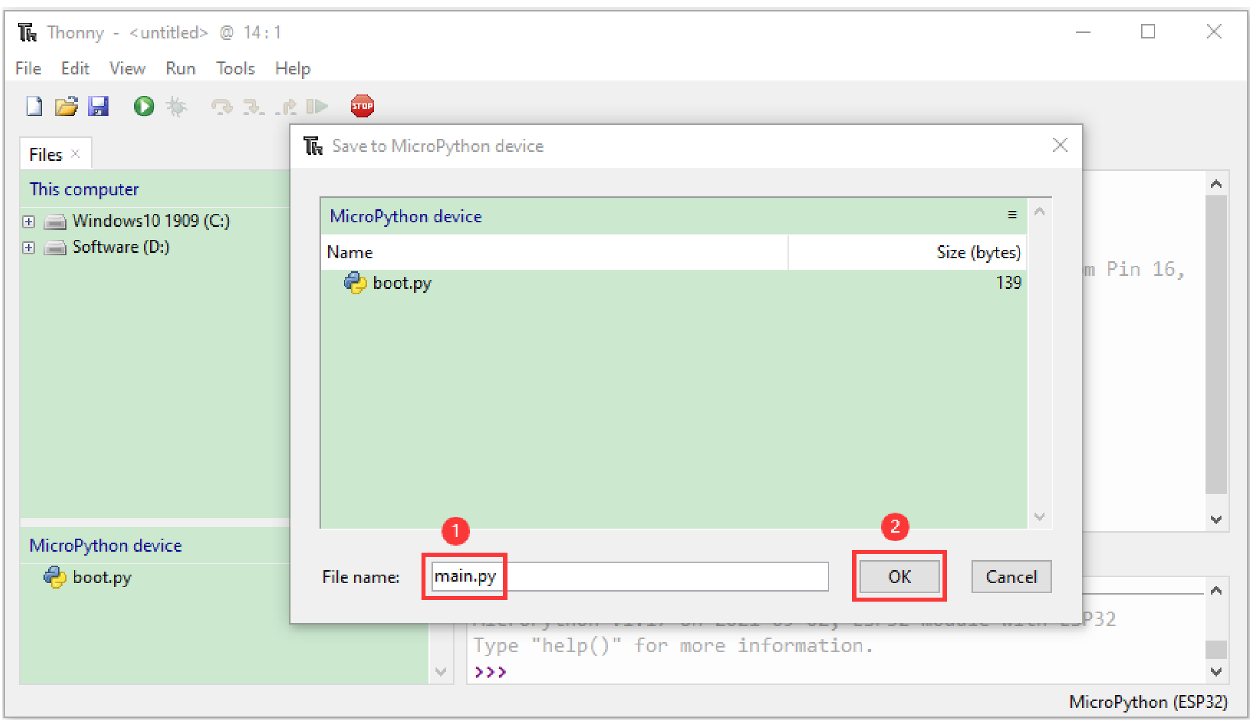
Enter the code in the new file. We take the Project_03_LED_Flashing.py as an example.



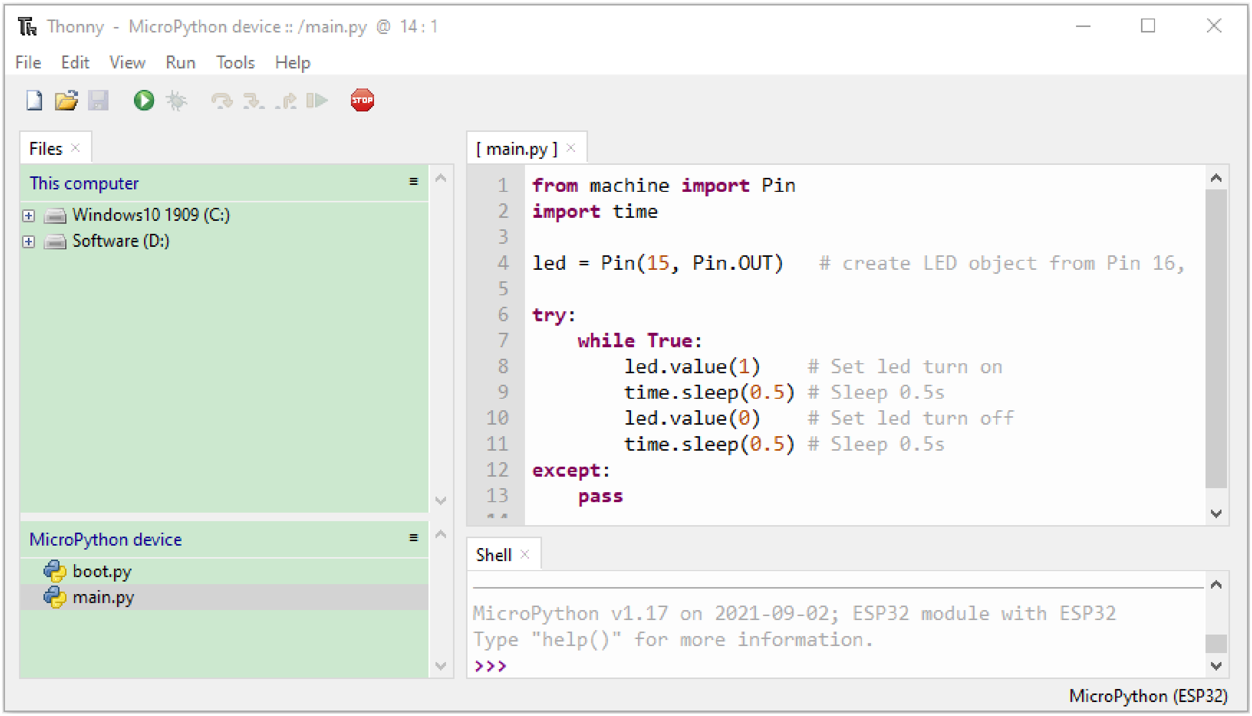
Click  to save the code to your PC or the ESP32.



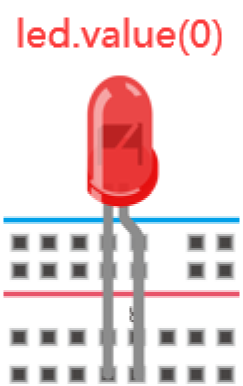
Select MicroPython device and enter main.py in the new page and click OK.



Then the code will be uploaded to the ESP32.



Disconnect the USB cable and connect it, you can see the effect of the LED flashing continuously in the circuit on a cycle.



PYTHON TUTORIAL

7.1 Download Python code files

Click on the link to download the Python code fileDownload Python Codes

7.2 Development Environment Configuration

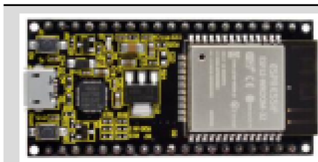
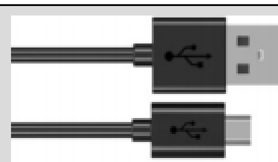
Click on the link to enter the development environment setup tutorial:*Pythin Development Environment Configuration*

7.3 Project 01: Hello World

1. Overview

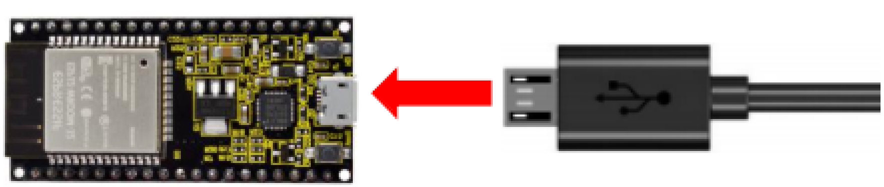
For ESP32 beginners, we will start with some simple things. In this project, you only need a ESP32 mainboard, a USB cable and Raspberry Pi to complete the "Hello World!" project, which is a test of communication between the ESP32 mainboard and the Raspberr y Pi as well as a primary project.

2. Components

	
ESP32*1	USB Cable*1

3. Wiring Diagram

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



Running code online

To run the ESP32 online, you need to connect the ESP32 to the computer, which allows you to compile or debug programs using Thonny software.

Advantages:

1. You can use the Thonny software to compile or debug programs.

2. Through the "Shell" window, you can view error messages and output results generated during the running of the program as well as query related function information online to help improve the program.

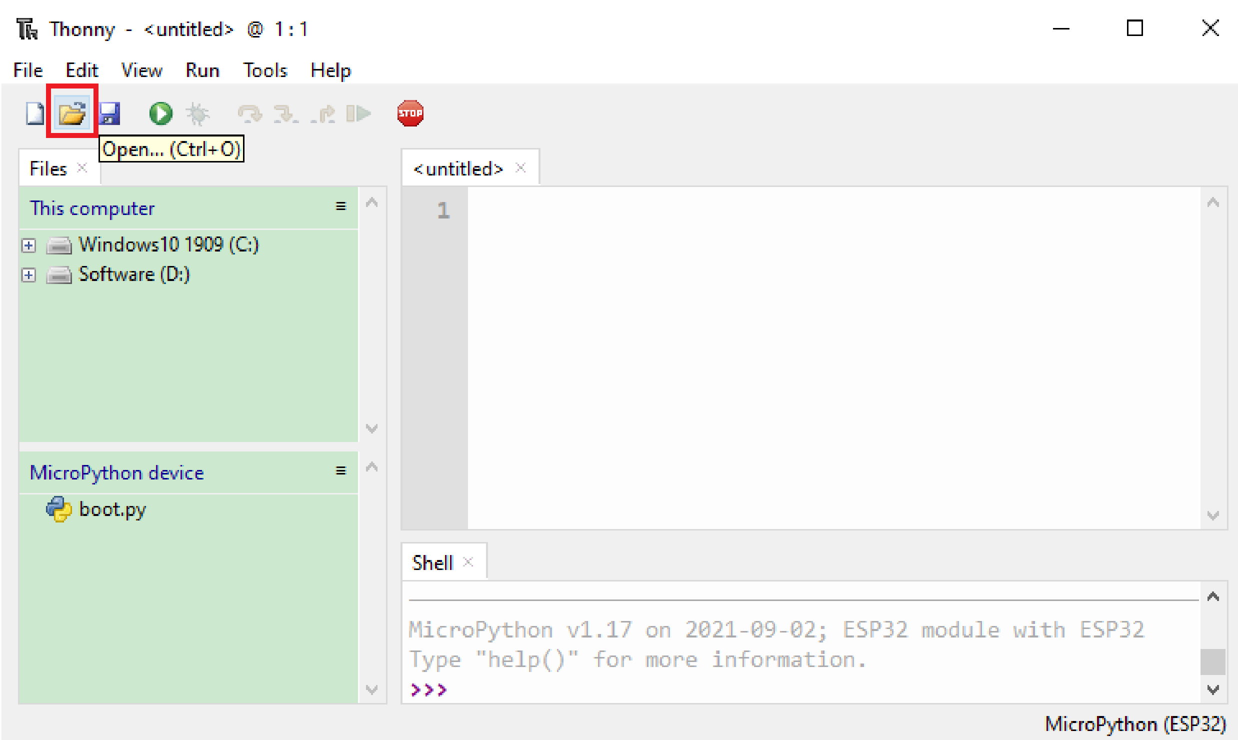
Disadvantages:

1. To run the ESP32 online, you must connect the ESP32 to a computer and run it with the Thonny software.

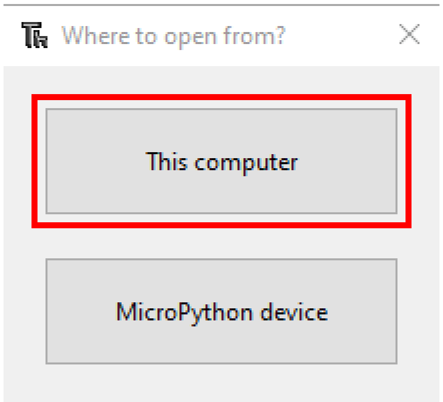
2. If the ESP32 is disconnected from the computer , when they reconnect, the program won't run again.

Basic Operation:

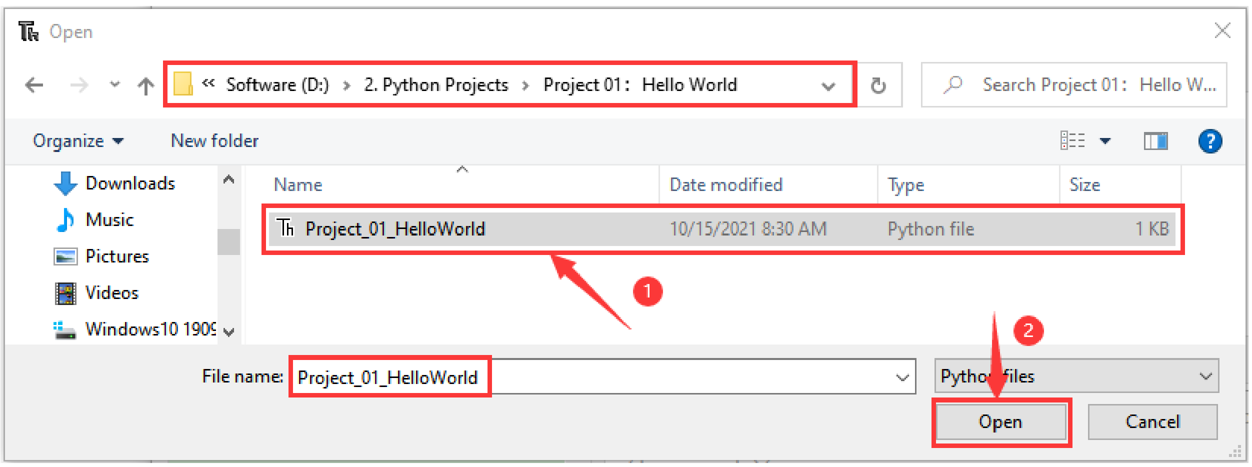
1. Open Thonny and click"Open…".



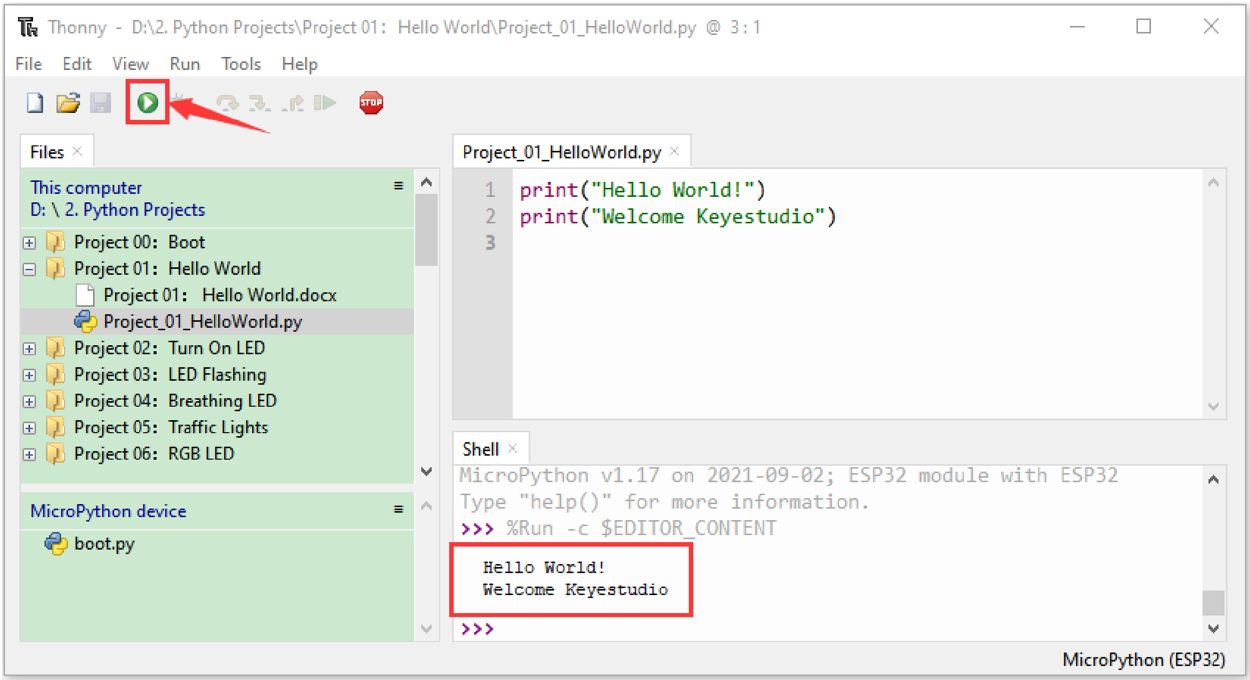
2. Click"This computer"in the new pop-up window.



In the new dialog boxselect“Project_01_HelloWorld.py”,click“Open”. (If you haven’t downloaded the code file, please click on the link to download it:Download Python Codes)

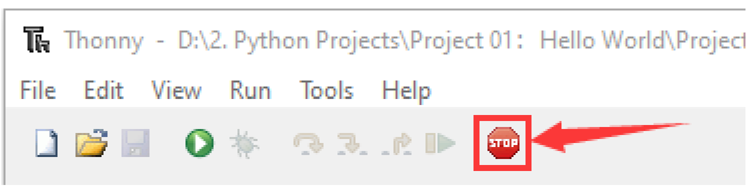


3. Click “Run current script”to execute the program“Hello World!”, “Welcome Keyestudio” , which will be printed in the“Shell”window.



Exit running online

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



5. Test Code

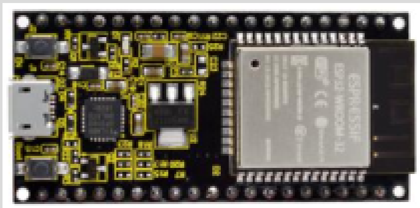
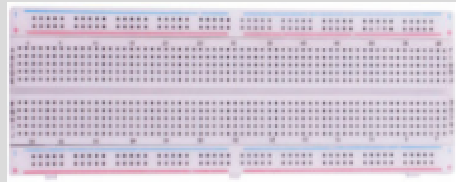
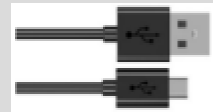
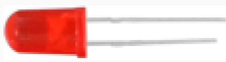
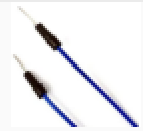
```
print("Hello World!")
print("Welcome Keyestudio")
```

7.4 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.

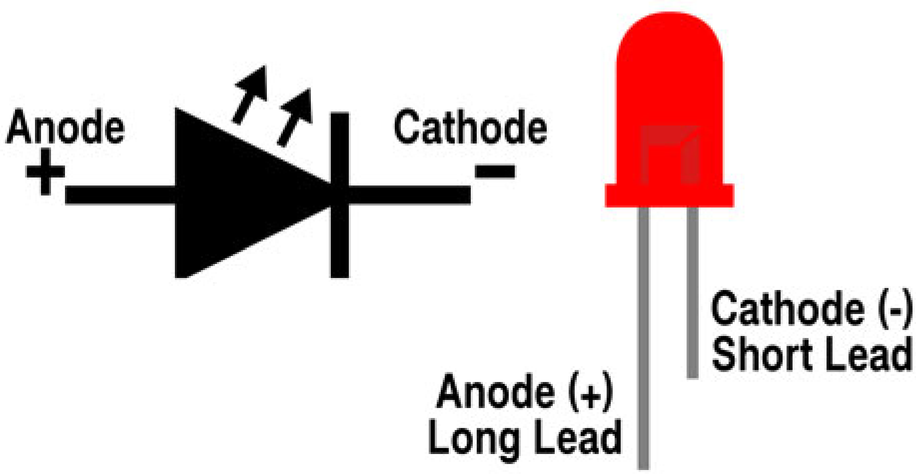
2. Components

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

3. Component Knowledge

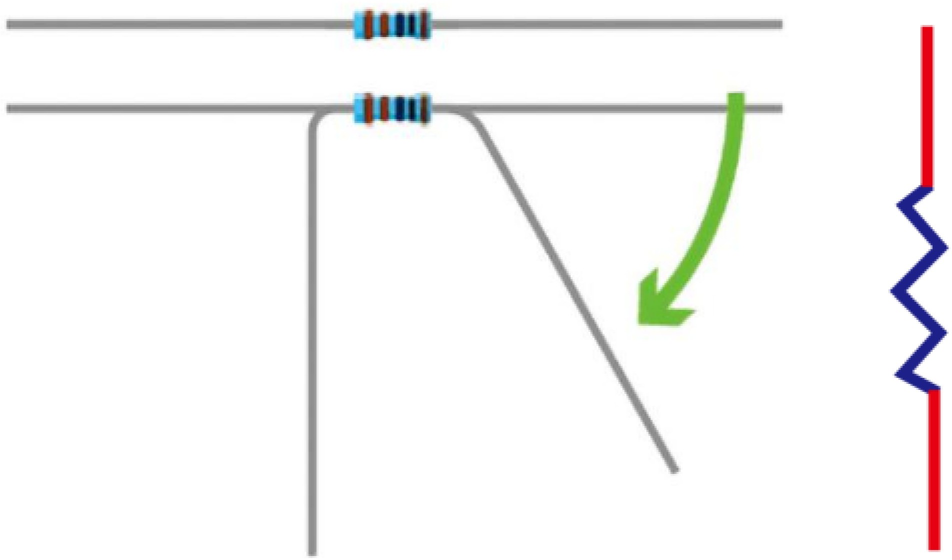
1LED:

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-band resistor

A resistor is an electronic component in a circuit that restricts or regulates the flow current to 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= 10007)



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.

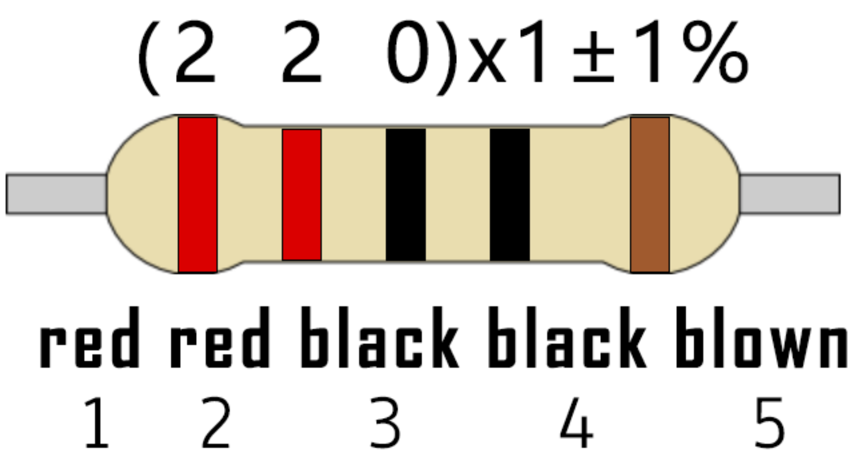
-Color 4 – Multiplier.

-Color 5 – Tolerance.

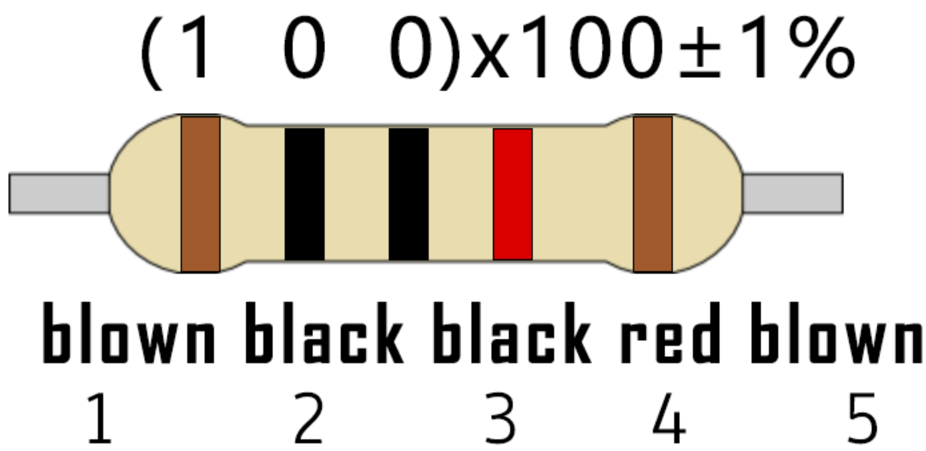
	1st Digit	2nd Digit	3rd Digit	Multiplier	Tolerance
Black		0	0	x1	
Brown	1	1	1	x10	± 1%
Red	2	2	2	x100	± 2%
Orange	3	3	3	x1K	± 3%
Yellow	4	4	4	x10K	± 4%
Green	5	5	5	x100K	± 0.5%
Blue	6	6	6	x1M	± 0.25%
Violet	7	7	7	x10M	± 0.10%
Grey	8	8	8	x100M	± 0.05%
White	9	9	9	x1G	
Gold				÷ 10	± 5%
Silver				÷ 100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

In this kit, we provide three five-band resistors with different resistance values. We three five-band resistors as an example.

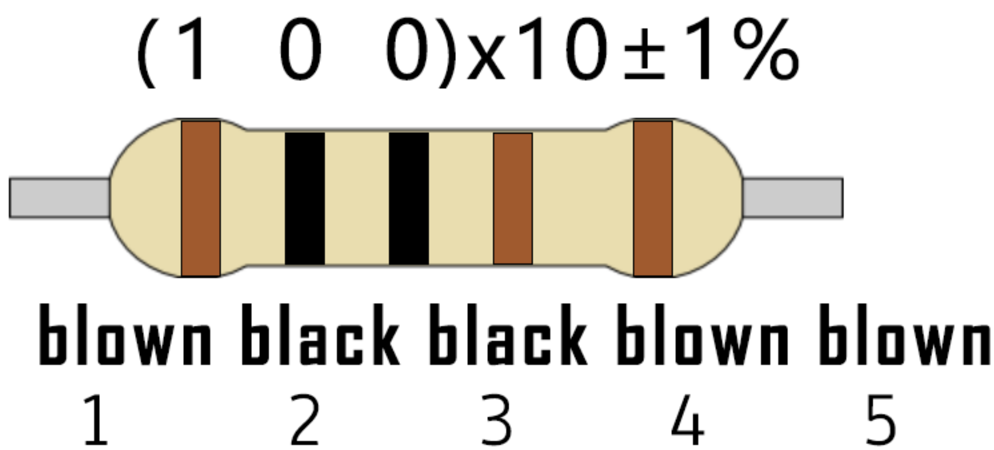
220 Resistor*10



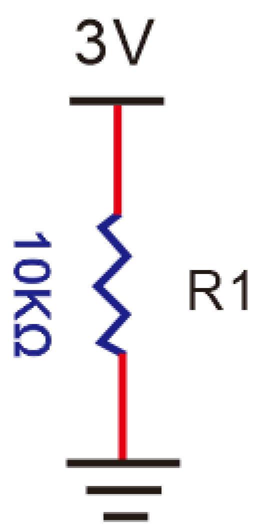
10K Resistor*10



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.

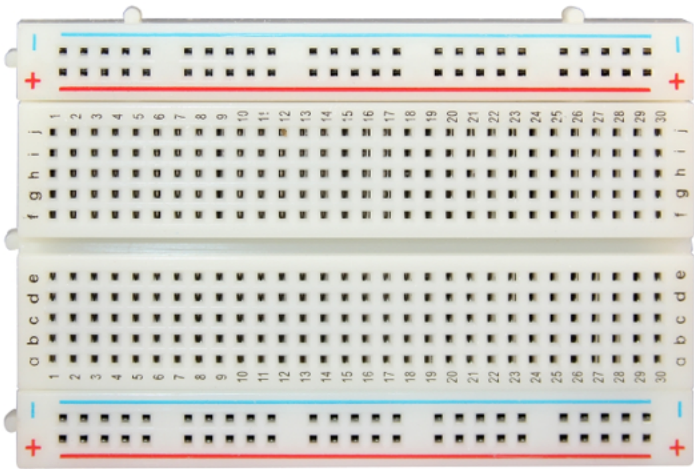


Don't connect a low resistance directly to the two poles of the power supply, which will cause excessive current to

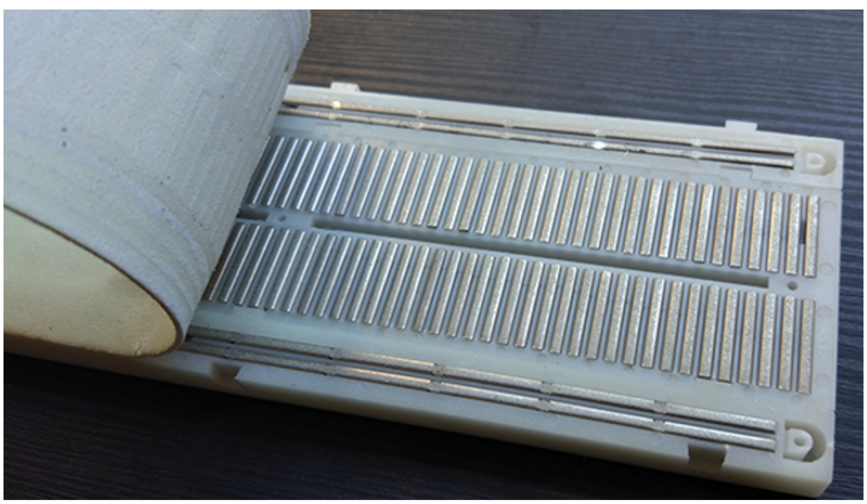
damage the electronic components. Resistors do not have positive and negative poles.

3Bread board

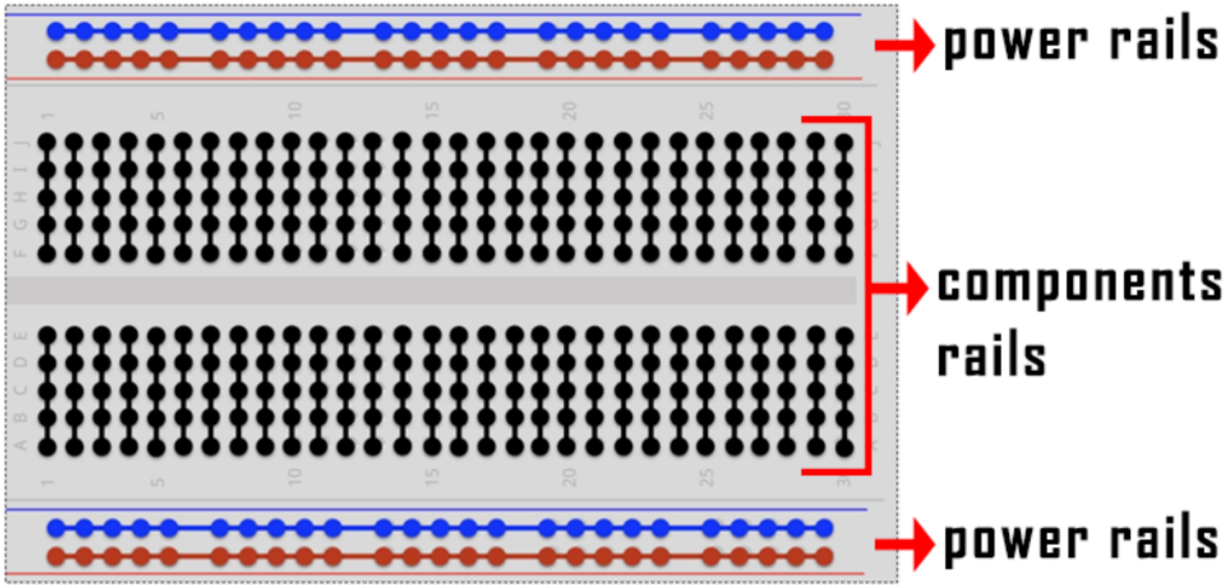
Breadboards are used to build and test circuits quickly before completing any circuit design. There are many holes in the breadboard that can be inserted into circuit components such as integrated circuits and resistors. A typical breadboard is shown below



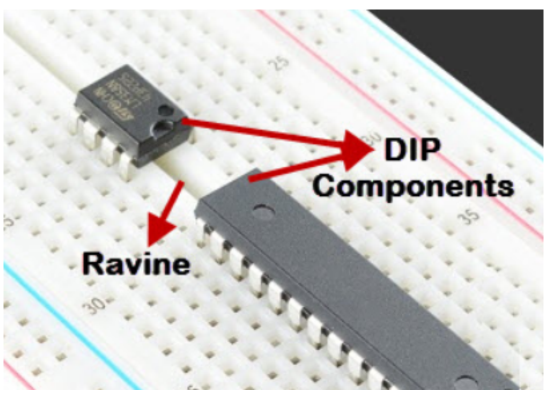
The breadboard has strips of metal , which run underneath the board and connect the holes on the top of the board. The metal strips are laid out as shown below. Note that the top and bottom rows of holes are connected horizontallywhile the remaining holes are connected vertically.

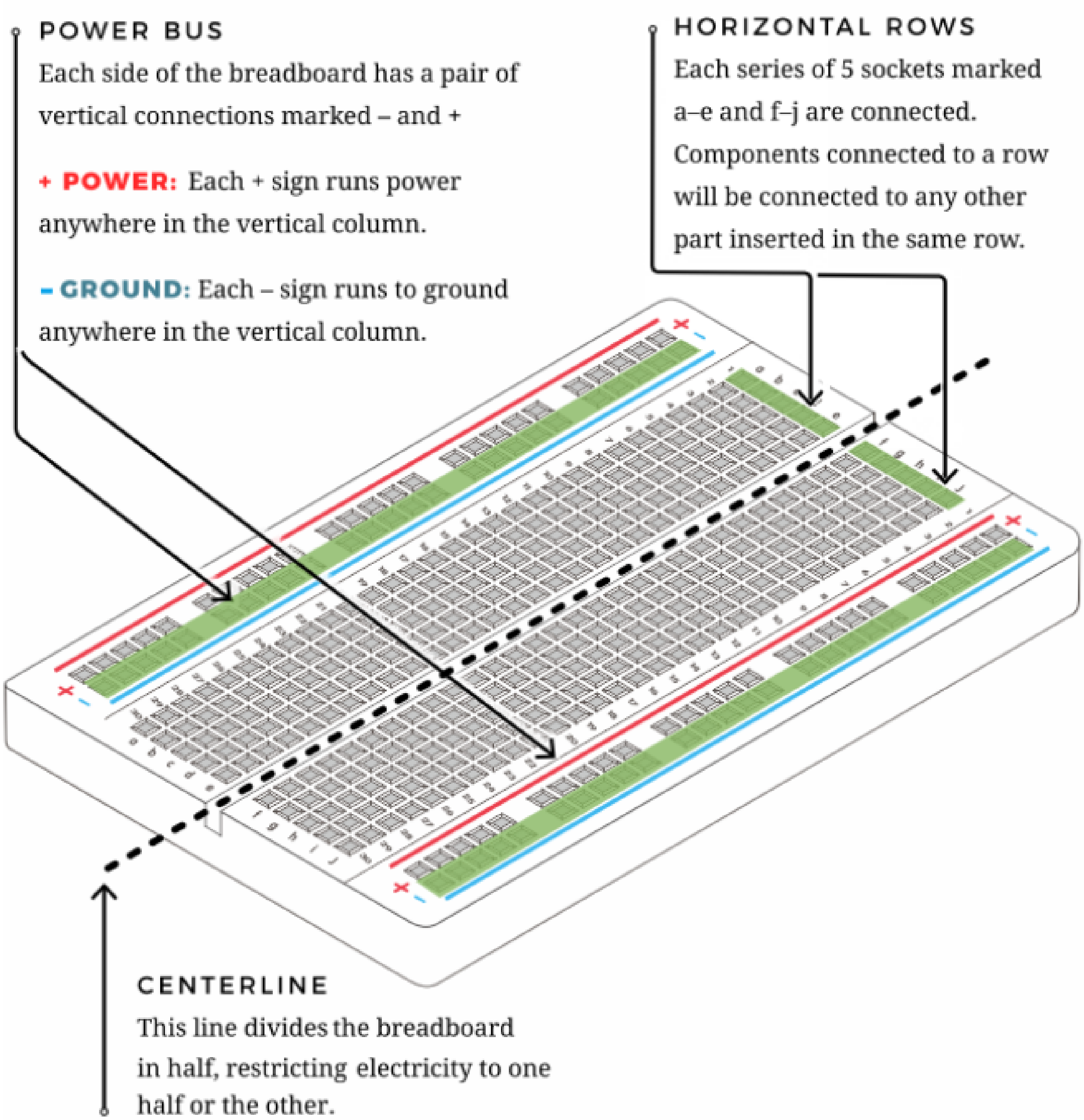


The first two rows (top) and the last two rows (bottom) of the breadboard are used for the positive pole (+) and negative pole (-) of the power supply respectively. The conductive layout of the breadboard is shown in the figure below:



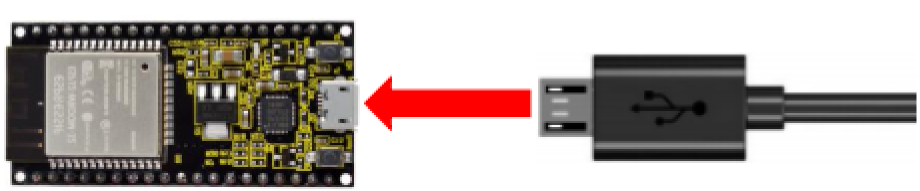
When we connect DIP (Dual In-line Packages) components, such as integrated circuits, microcontrollers, chips and so on, we can see that a groove in the middle isolates the middle part, so the top and bottom of the groove is not connected. DIP components can be connected as shown in the following diagram:





4) **Power Supply**

The ESP32 needs 3.3V-5V power supply. In this project, we will connect the ESP32 to the computer via an USB cable.

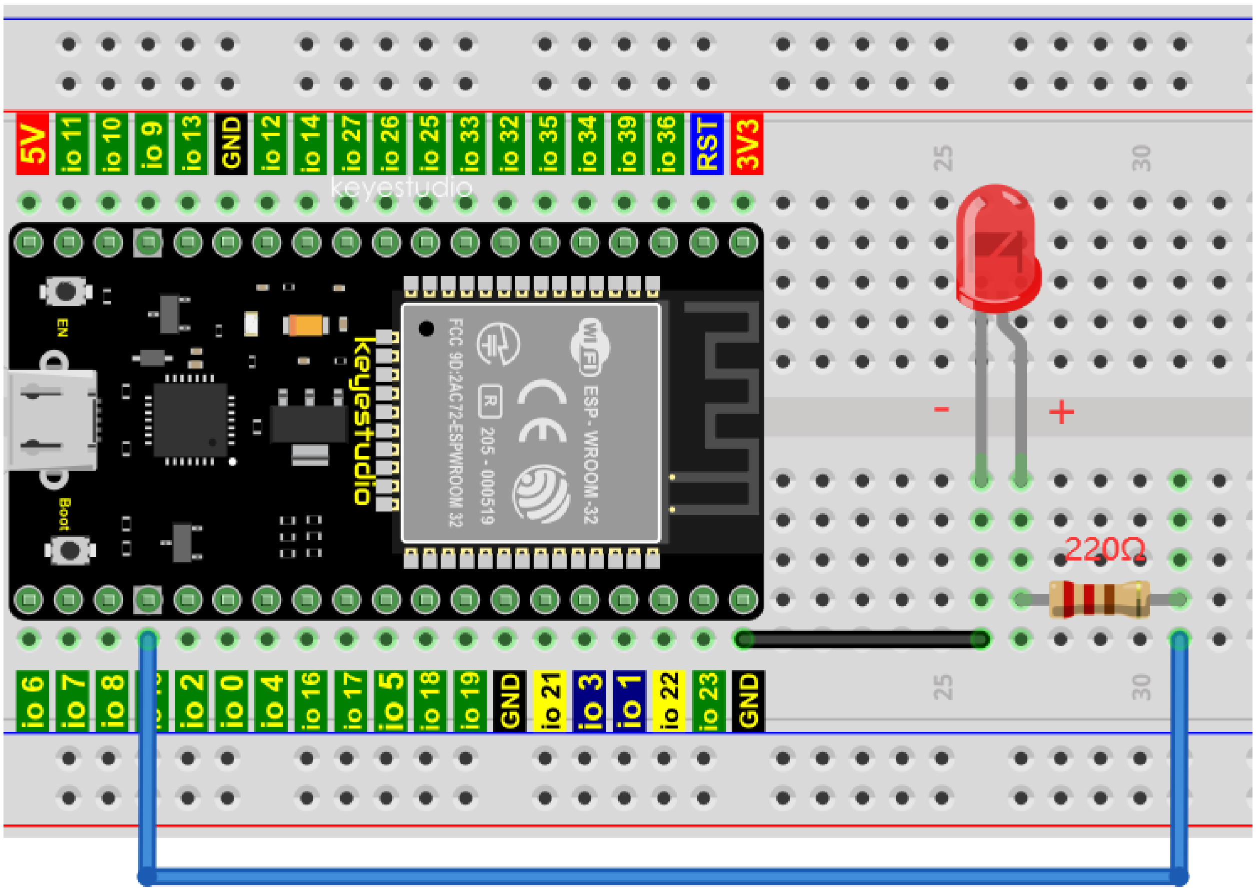


4.Wiring Diagram

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

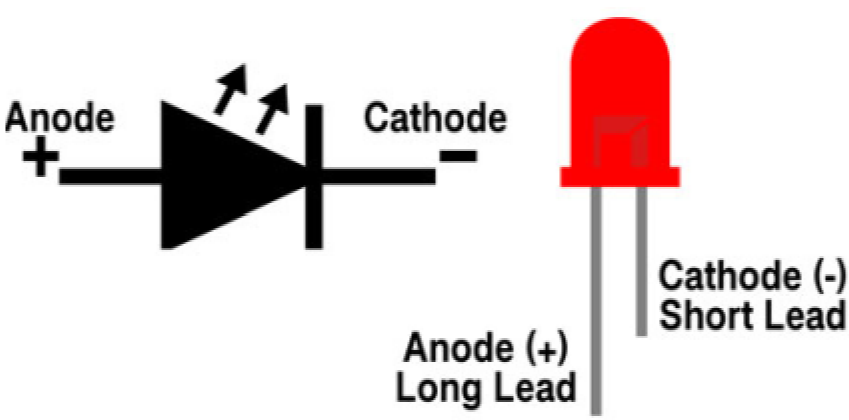
Note: Avoid any possible short circuits (especially connecting 3.3V and GND)!

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

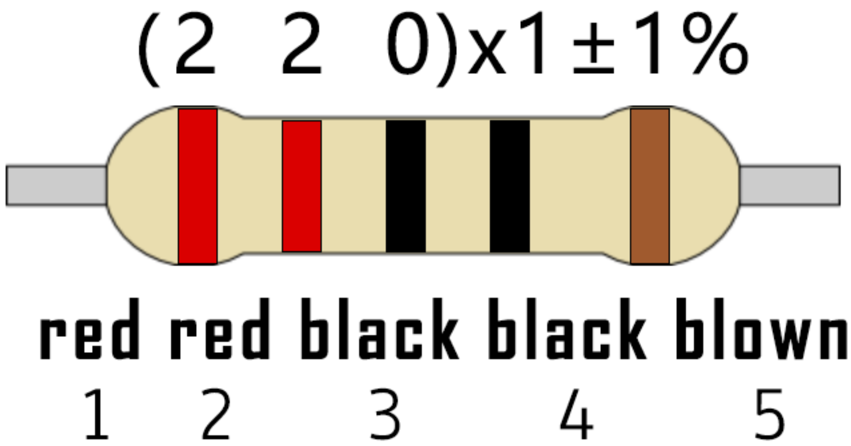


Note:

How to connect a LED

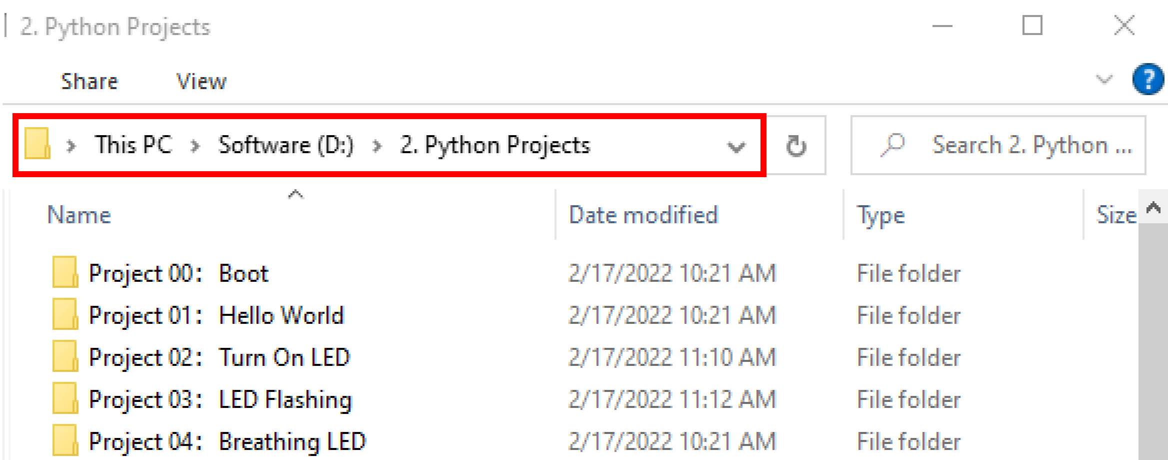


How to identify the 220 Five-band resistor



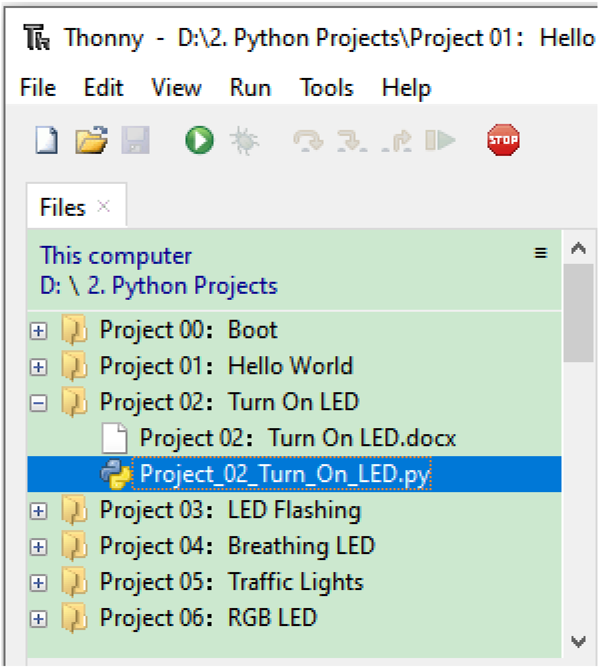
5. Test Code

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

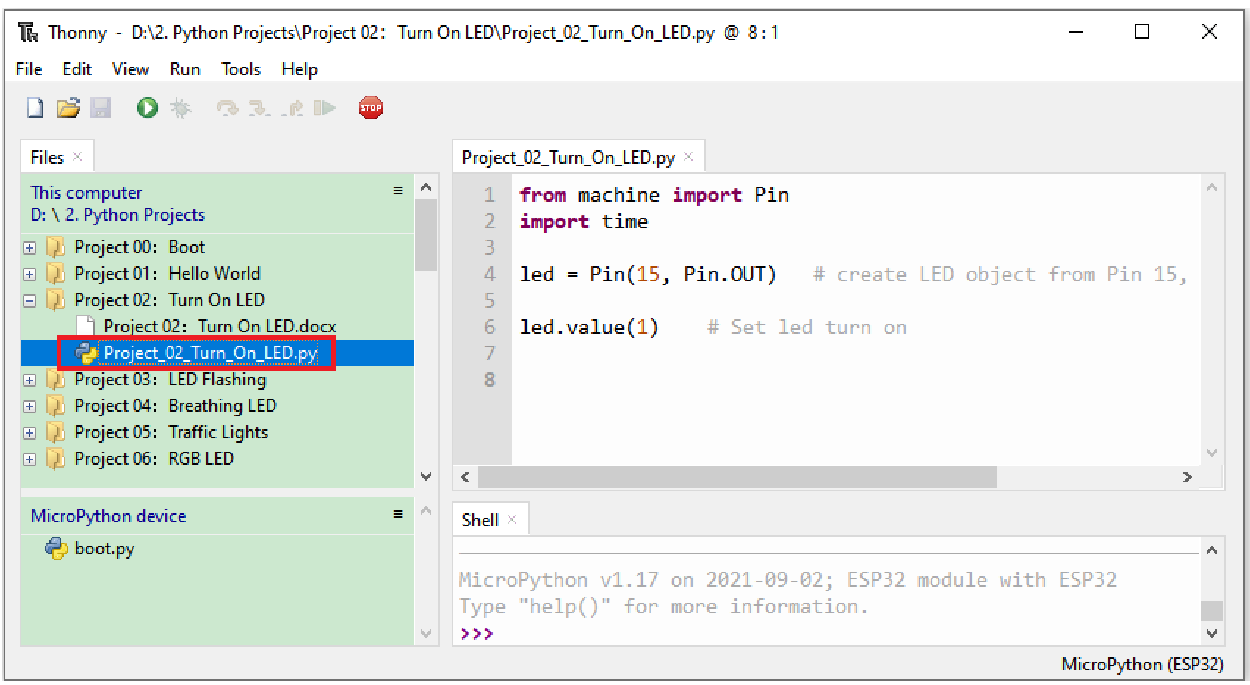


Exit running online

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

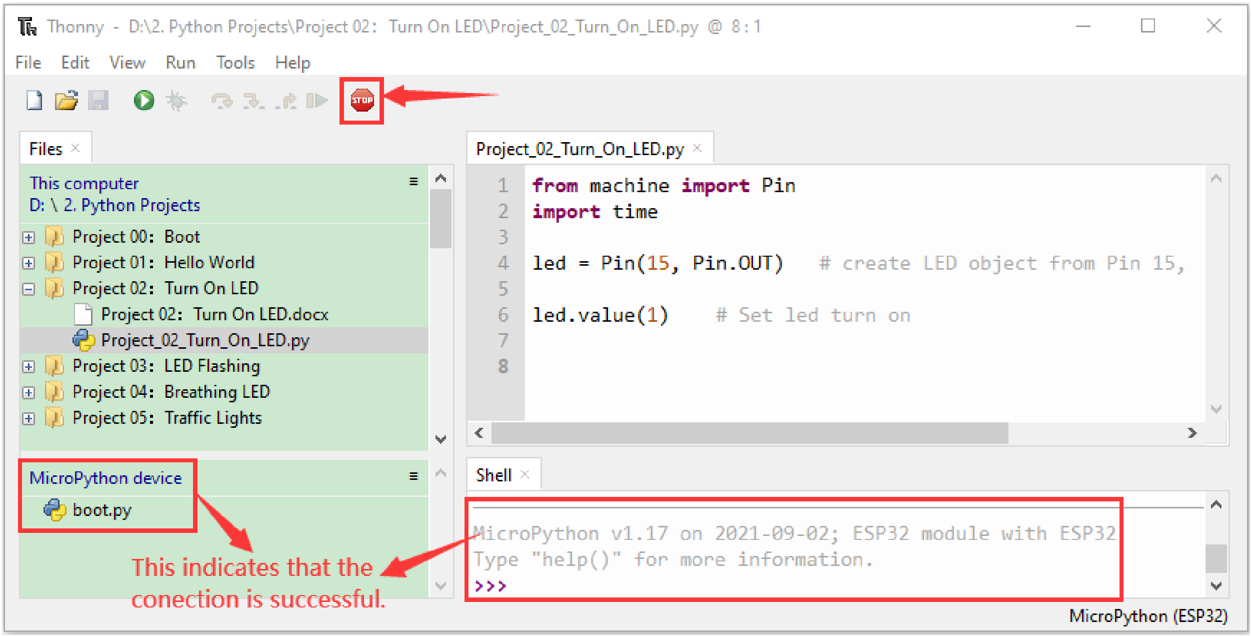


Click"Project 02Turn On LED", double-click"Project_02_Turn_On_LED.py"to open it, as shown below;

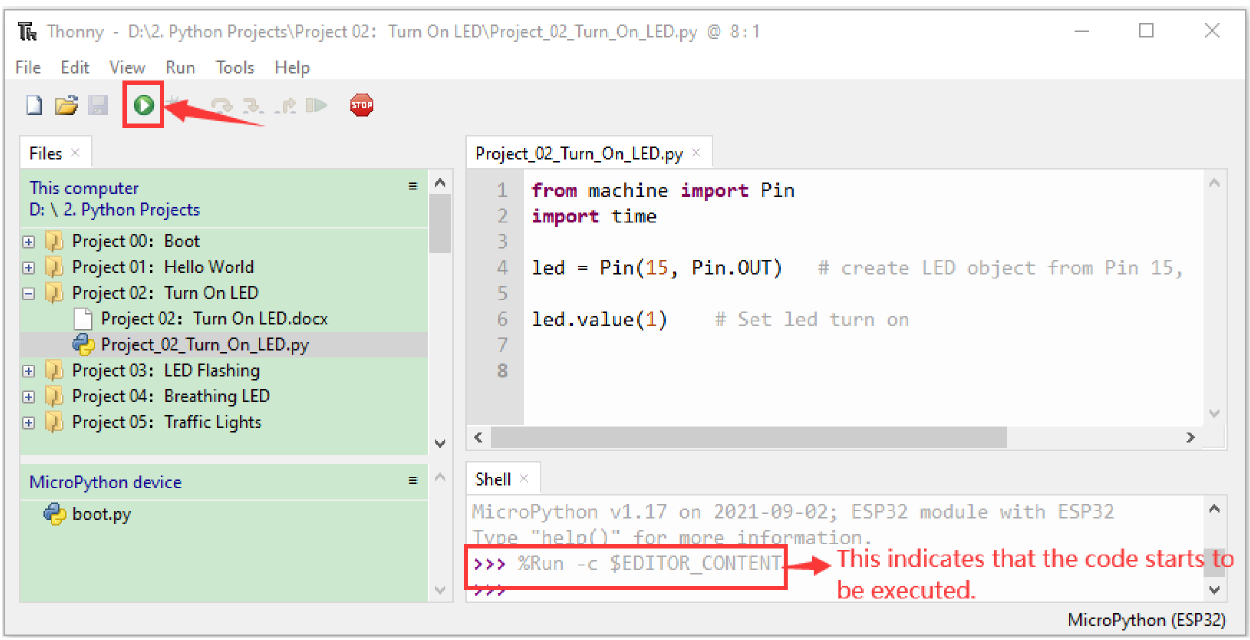


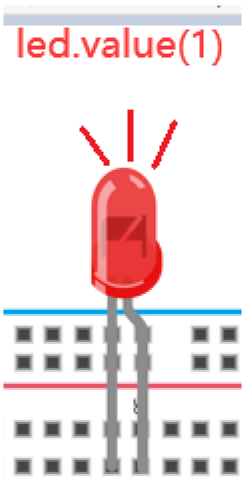

```
from machine import Pin
import time
led = Pin(15, Pin.OUT)    # create LED object from Pin 15, Set Pin 15 to output
led.value(1)     # Set led turn on
```

Connect the ESP32 to your PC. Click "Stop/Restart backend"then go to the Shell window to check.

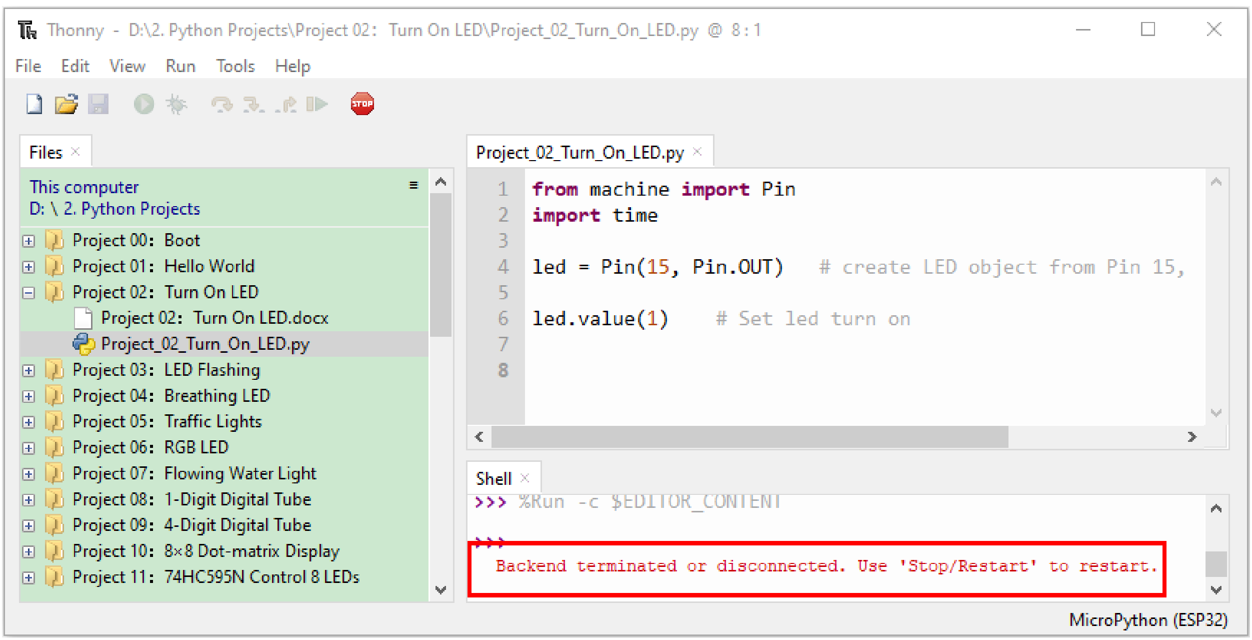


Click "Run current script"the code starts to be executed and the LED in the circuit lit up. Press "Ctrl+C" or click "Stop/Restart backend" to exit the program.



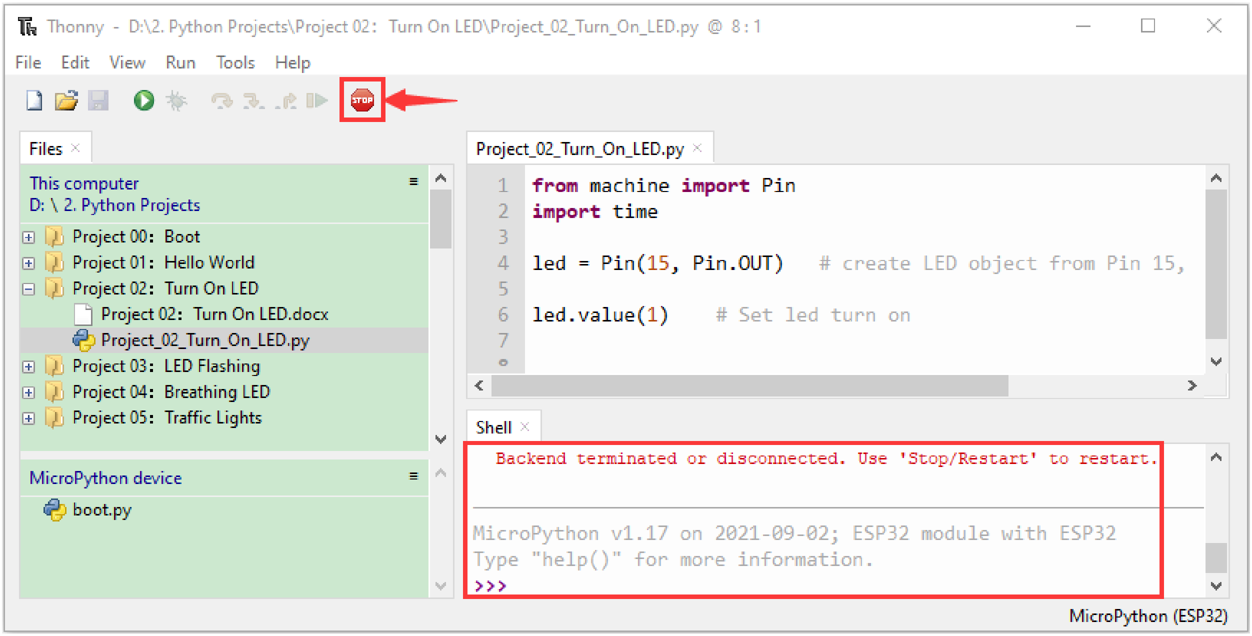


Note: This is the code running online. If you disconnect USB cable and power up the ESP32 or press its reset button, LED is not bright and the following messages will be displayed in the "**Shell**" window of Thonny:

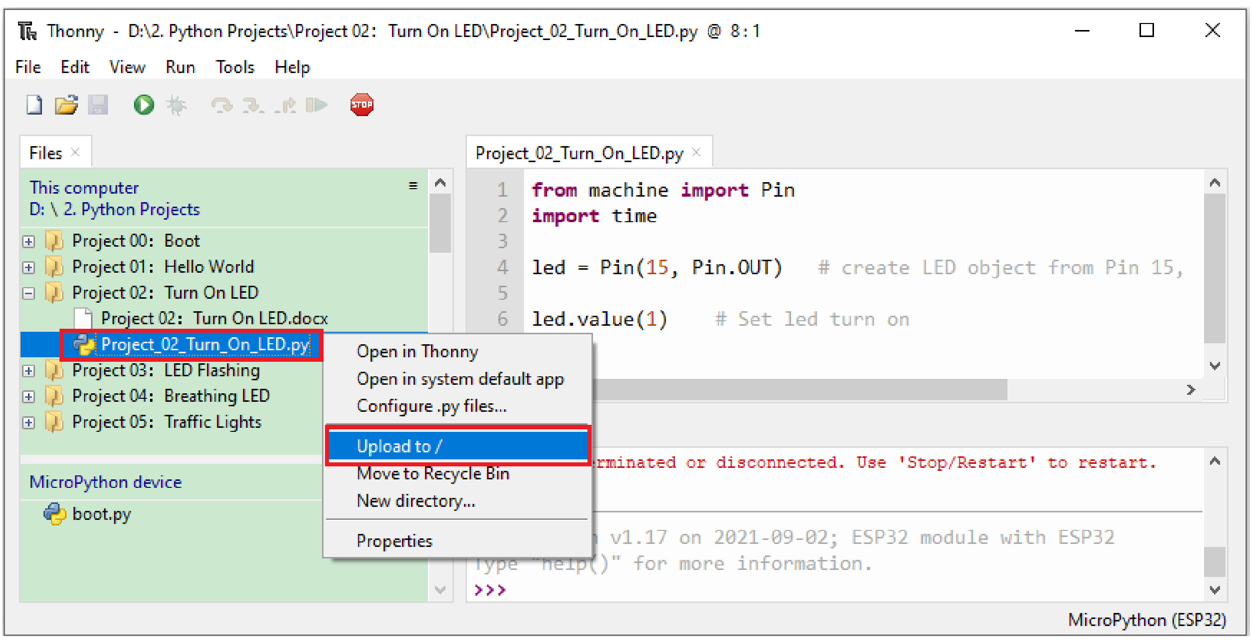


Code running offlineUpload the code to ESP32

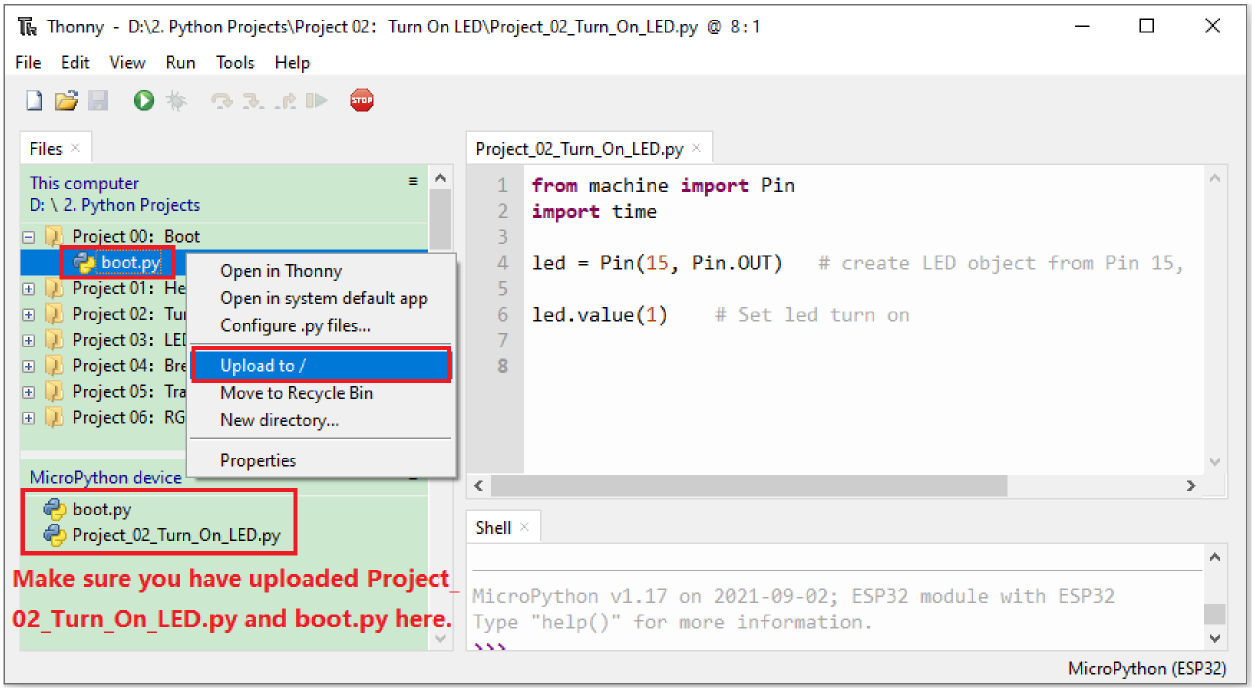
Make sure the ESP32 has been connected to the computer, click "Stop/Restart backend".



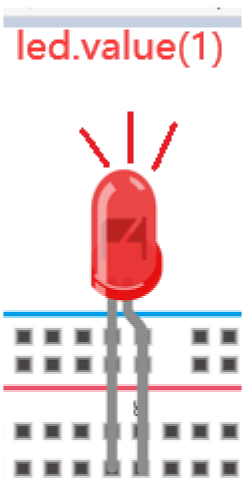
As shown below, right-click the file"Project_02_Turn_On_LED.py"select "**Upload to /**"to upload the code to ESP32.



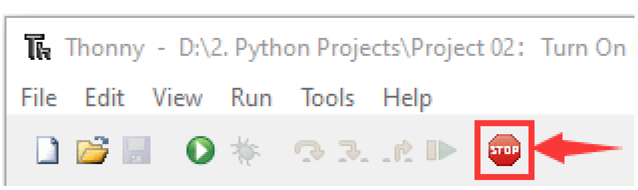
Upload"boot.py"in the same way.



Press the reset button of ESP32 and you can see LED is ON .



NoteCodes here is run offline. If you want to stop running offline and enter“**Shell**”, just click “Stop/Restart backend”in Thonny.

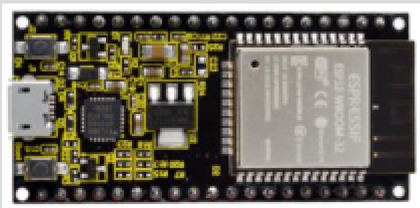
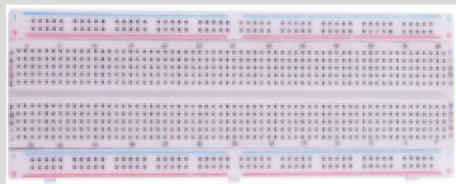
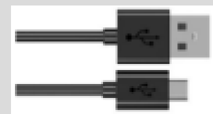
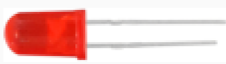
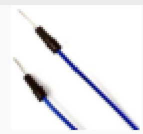


7.5 Project 03LED Flashing

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.

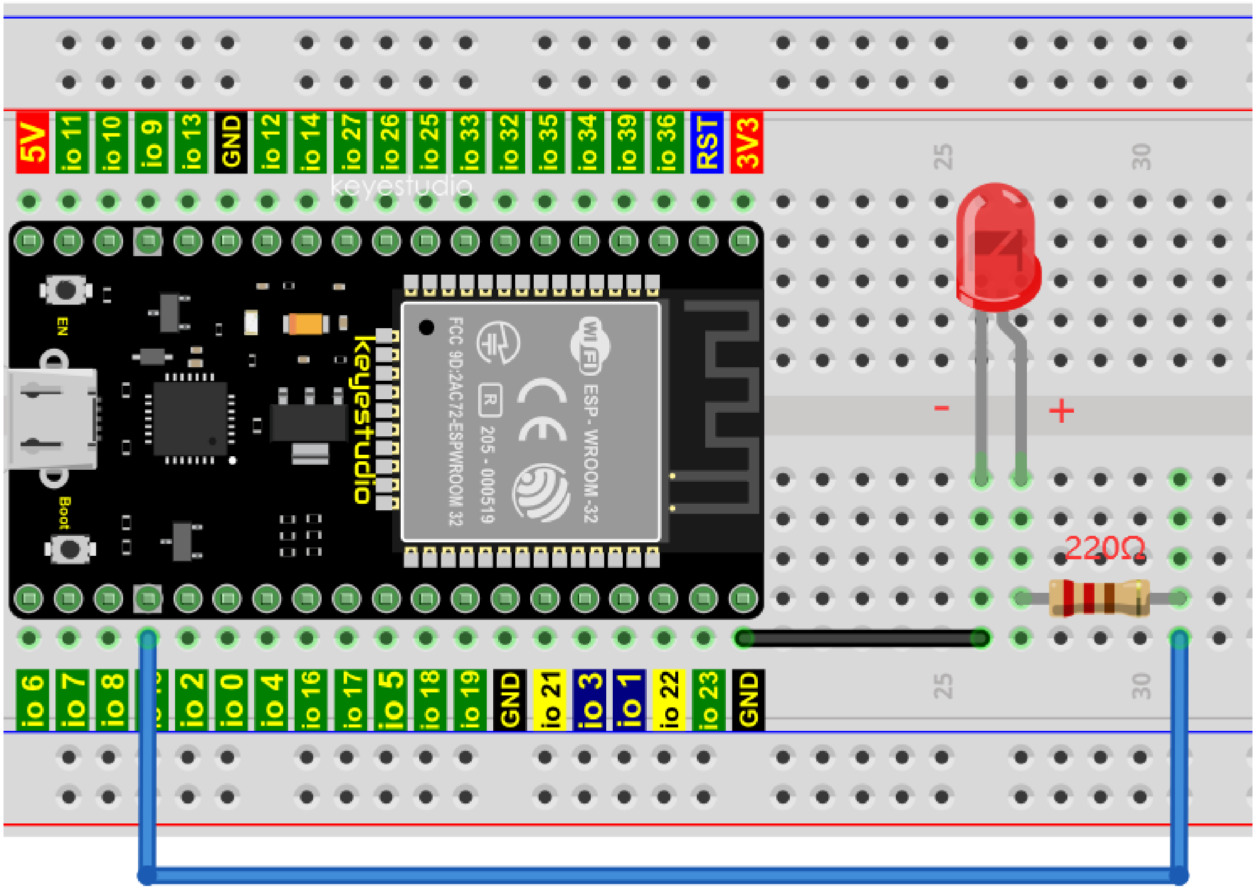
2.Components

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

3.Wiring diagram

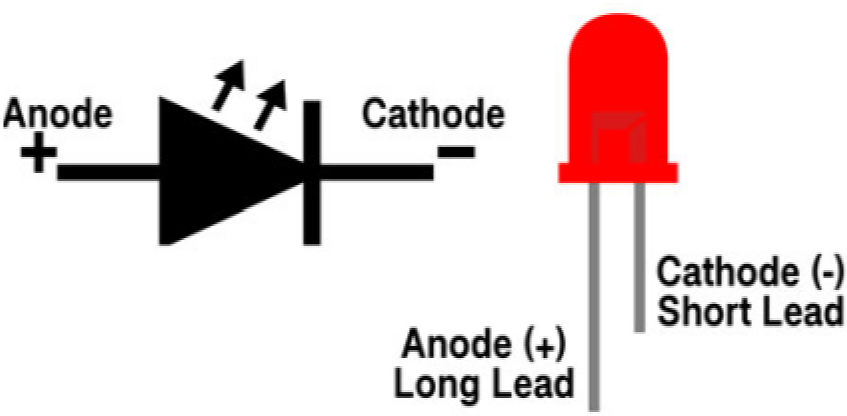
First, disconnect all power from the ESP32. Then build the circuit according to the wiring diagram. After the circuit is built and verified correct, connect the ESP32 to your computer using a USB cable.

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

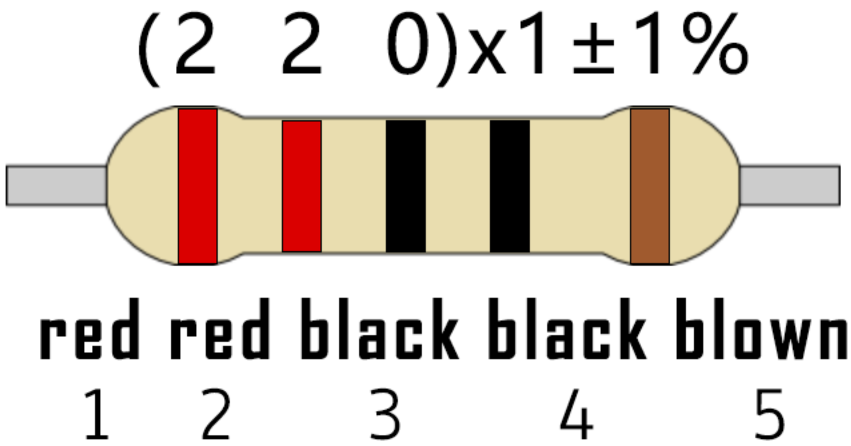


Note:

How to connect a LED



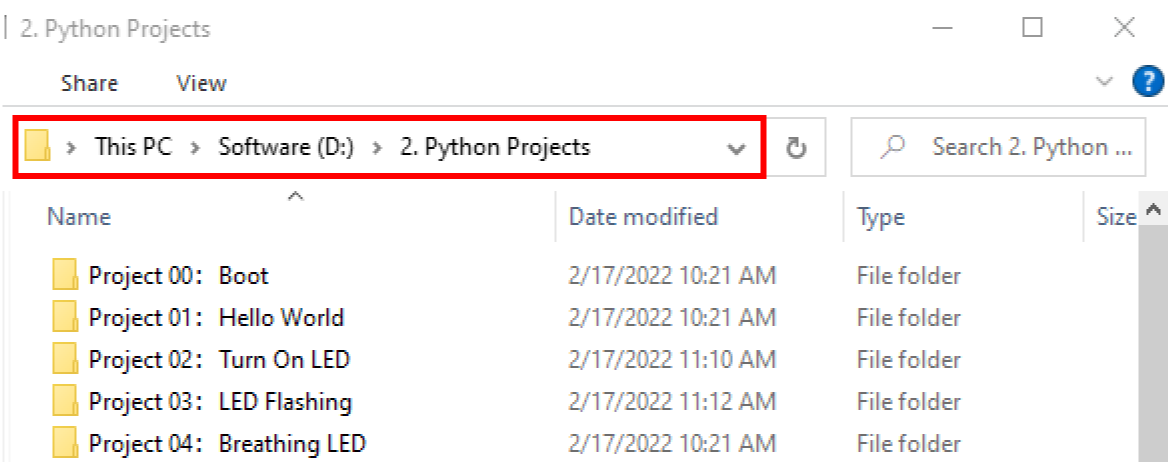
How to identify the 220 Five-color ring resistor



4.Project code

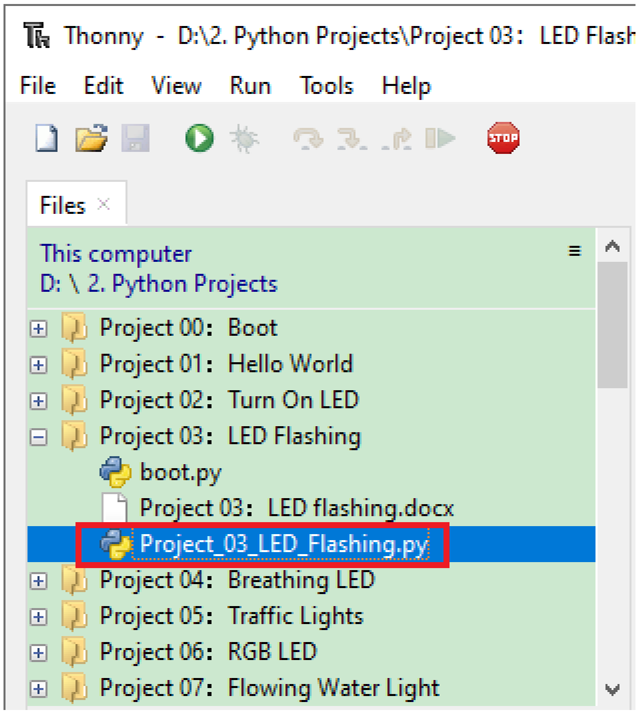
Codes used in this tutorial are saved in "2. Python Projects".

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

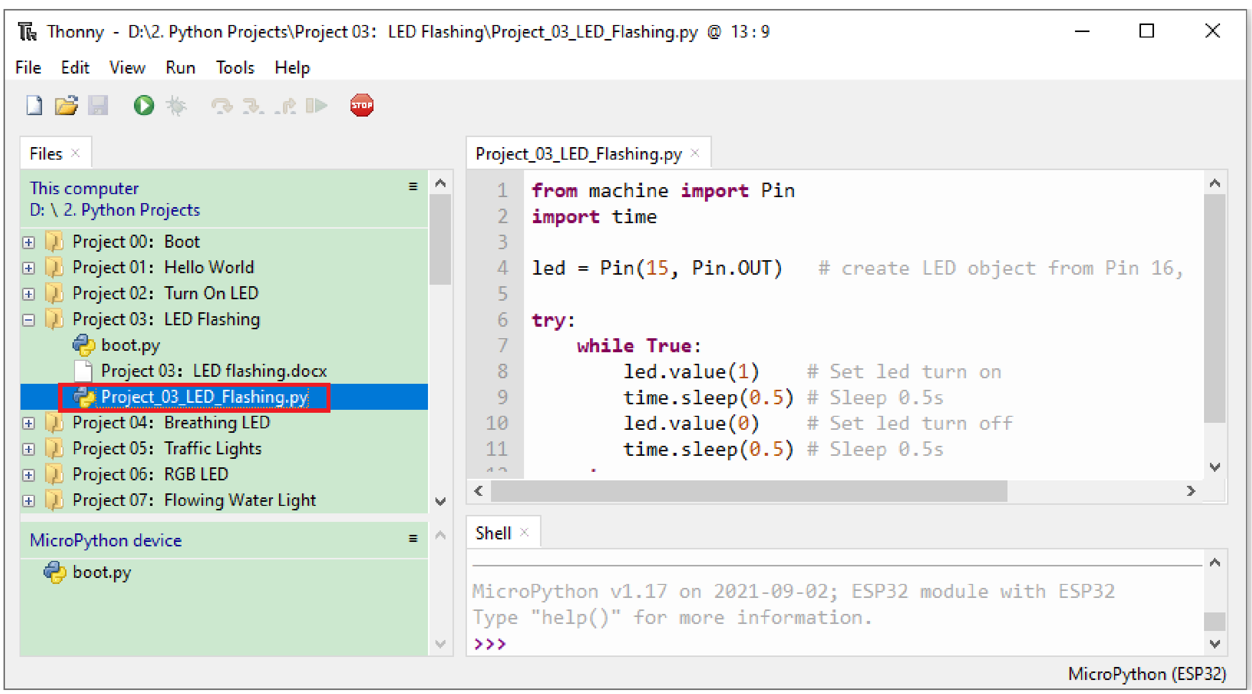


Code running online:

Open"Thonny"click"This computer"→"D:"→"2. Python Projects"→"Project 03LED Flashing".



Expand folder"Project 03: LED Flashing"and double left-click "Project_03_LED_Flashing.py" to open it. As shown in the illustration below



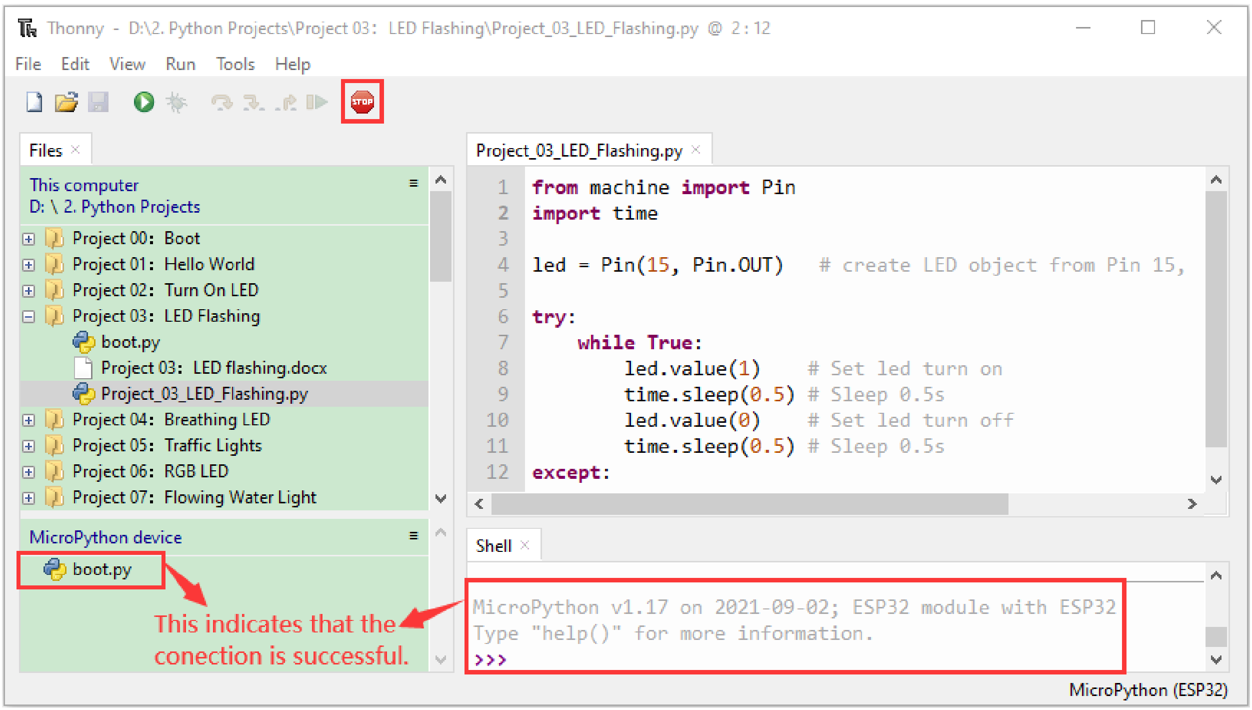

```
from machine import Pin
import time

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

try:
```

(continues on next page)

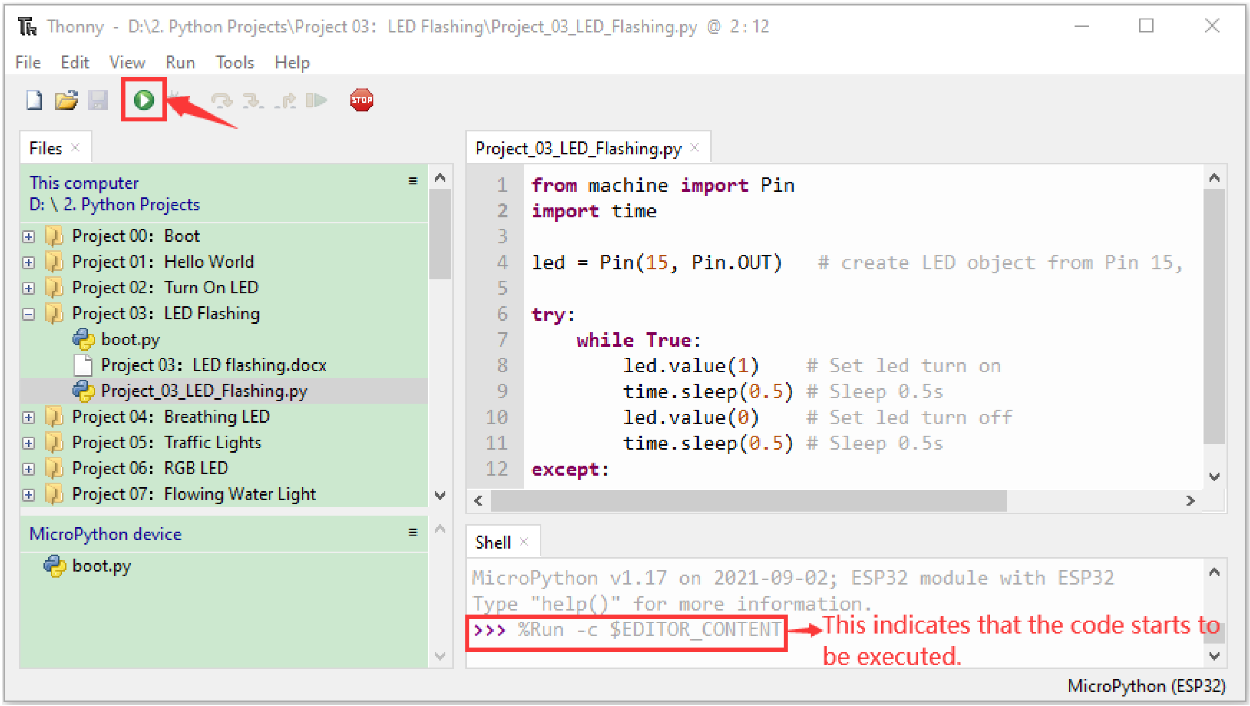
(continued from previous page)

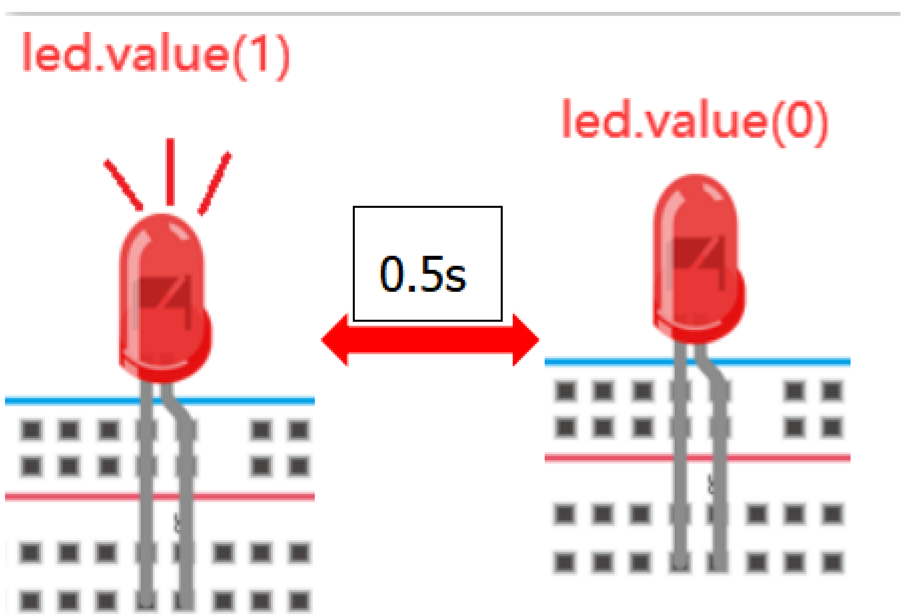
```
    while True:
        led.value(1)     # Set led turn on
        time.sleep(0.5) # Sleep 0.5s
        led.value(0)     # Set led turn off
        time.sleep(0.5) # Sleep 0.5s
except:
    pass
```

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

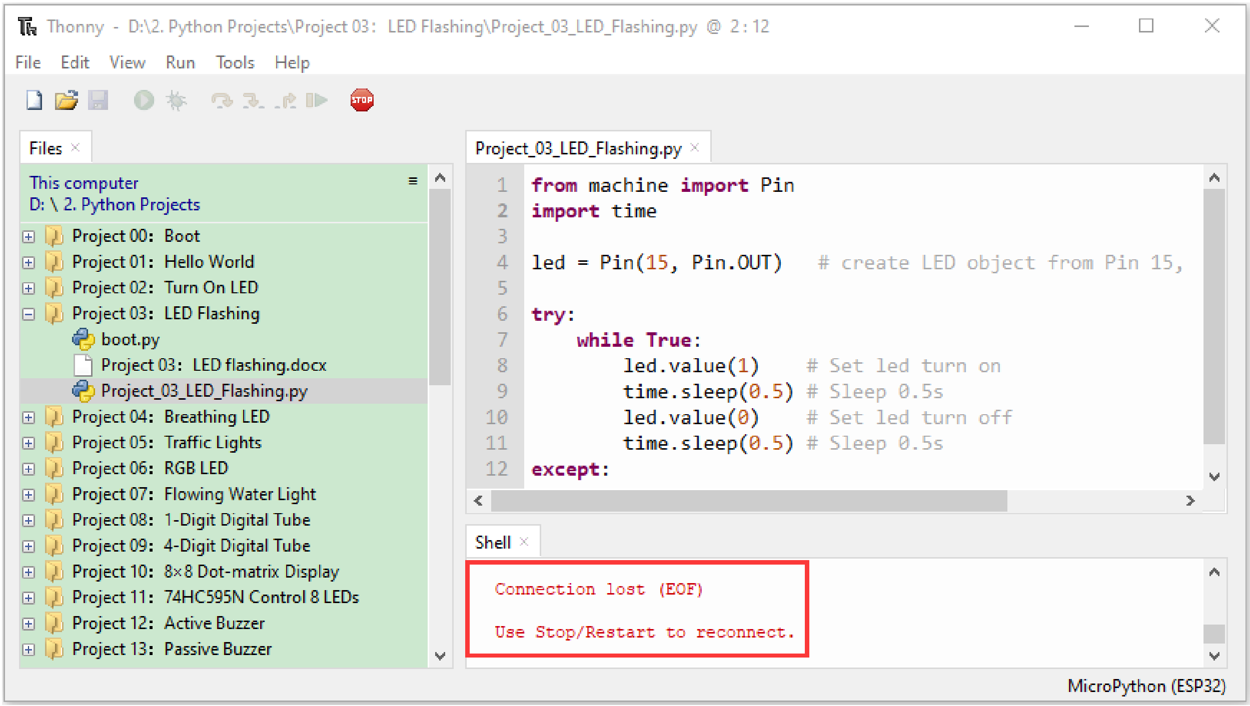


Click "Run current script"the code starts to be executed and you can see the LED flash. Press"Ctrl+C"or click "Stop/Restart backend"to exit the program.



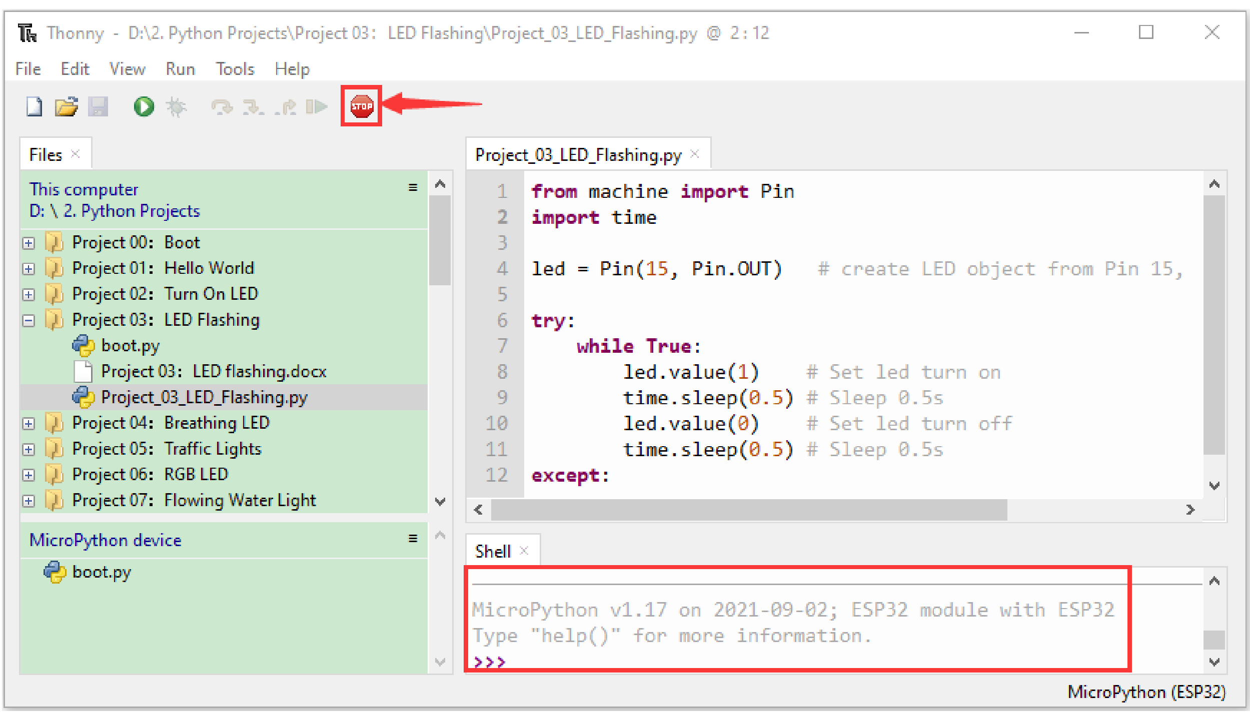


Note: This is the code running online. If you disconnect USB cable and power up the ESP32 or press its reset button, the LED in the circuit will stop flashing and the following messages will be displayed in the "Shell"

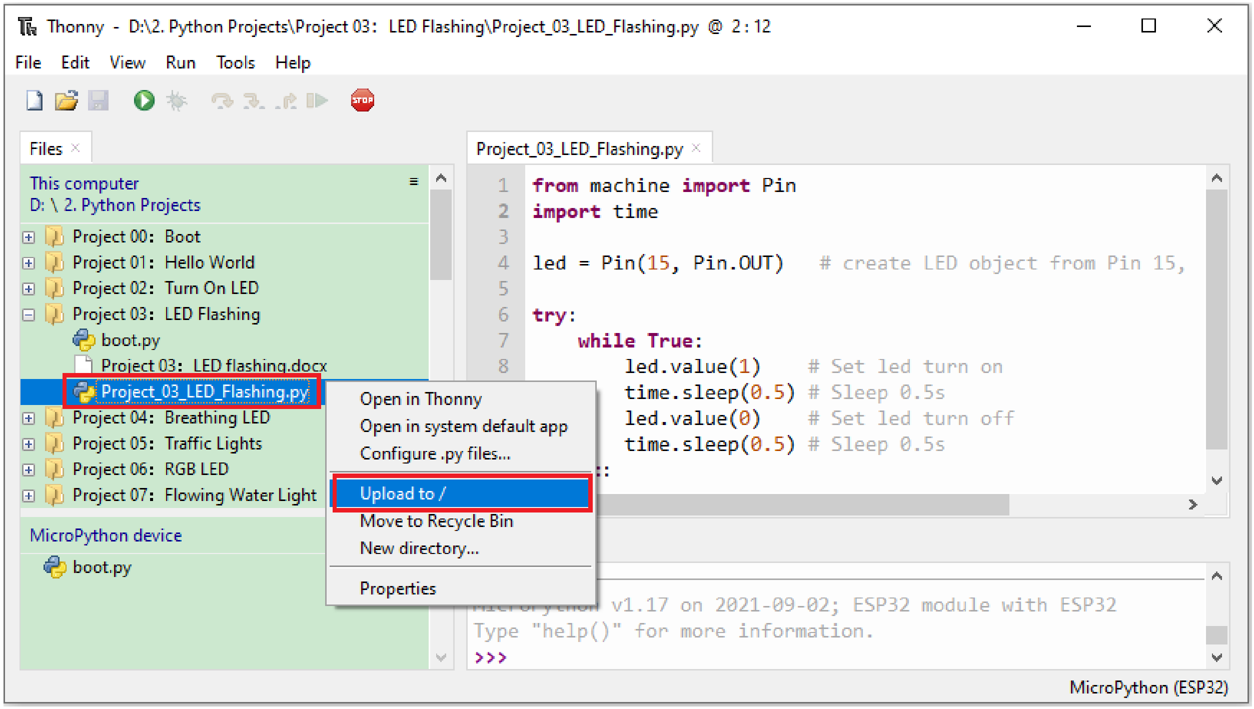


Code running offlineUpload the code to ESP32

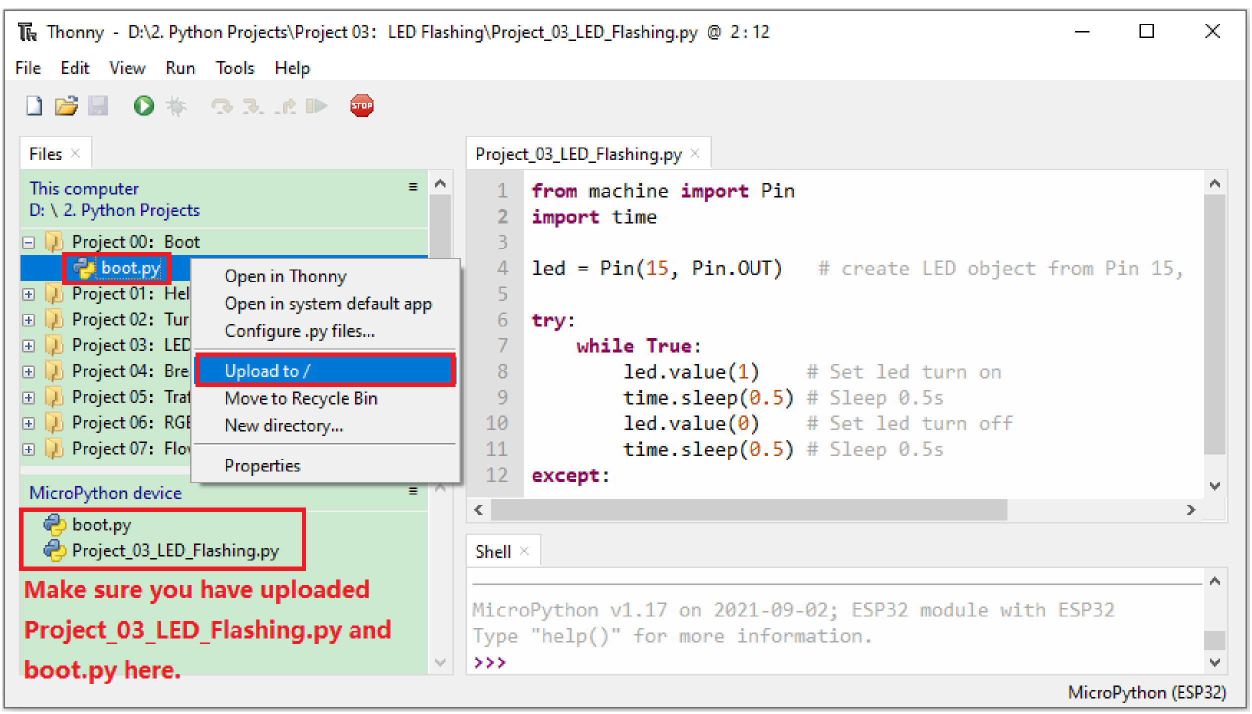
Make sure the ESP32 has been connected to the computer, click “Stop/Restart backend”.



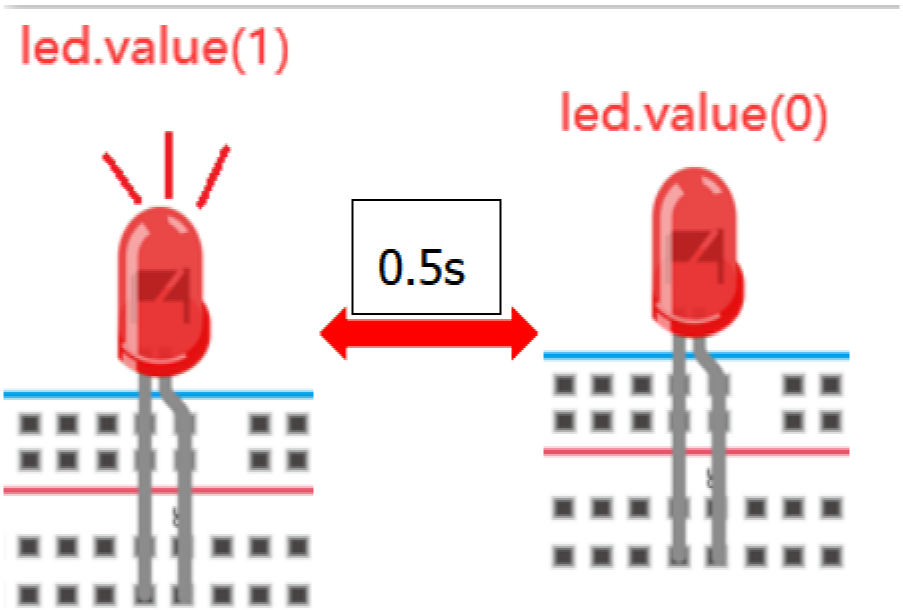
As shown below, right-click the file“Project_03_LED_Flashing.py”select “**Upload to /**”to upload the code to ESP32.



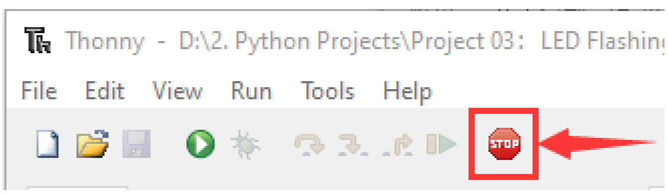
Upload“boot.py”in the same way.



Press the reset button of ESP32 and you can see the LED flash



NoteCodes here is run offline. If you want to stop running offline and enter"**Shell**", just click "Stop/Restart backend"in Thonny.



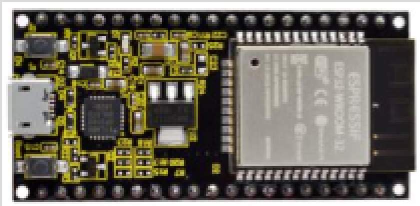
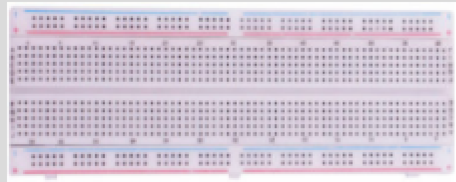
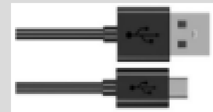
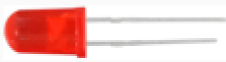
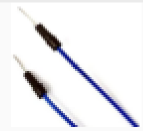
7.6 Project 04: Breathing Led

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.

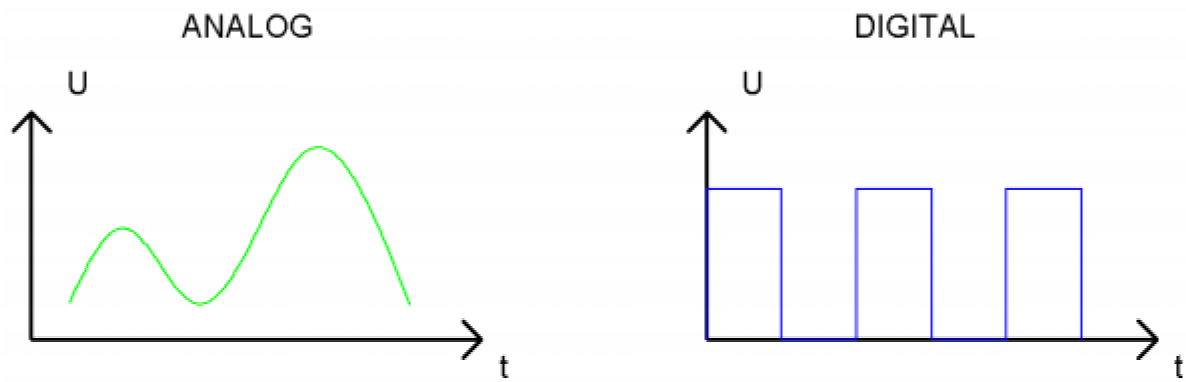
2.Components

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

3.Component knowledge

Analog & Digital:

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



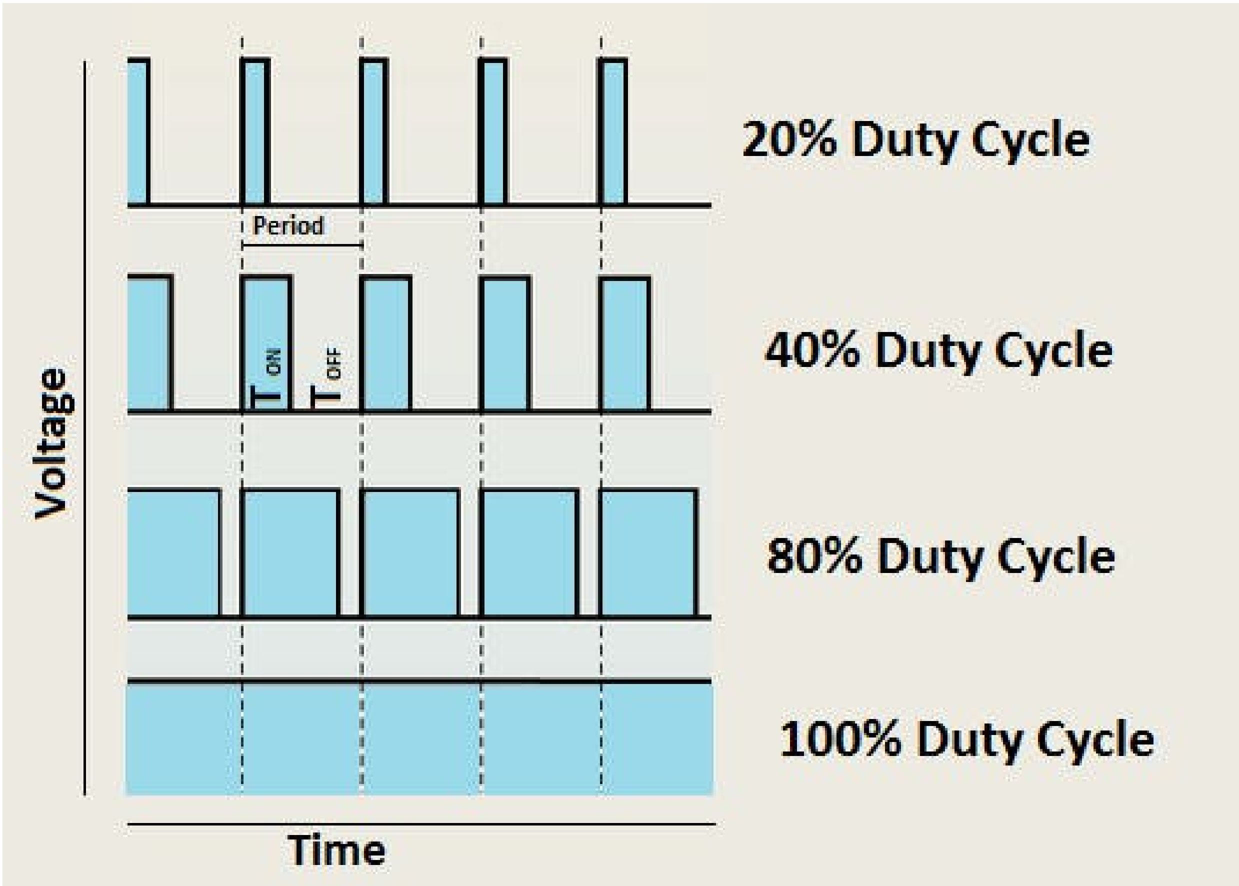
In practical application, we often use binary as the digital signal, that is a series of 0's and 1's. Since a binary signal only has two values (0 or 1), it has great stability and reliability. Lastly, both analog and digital signals can be converted into the other.

PWM

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

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

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

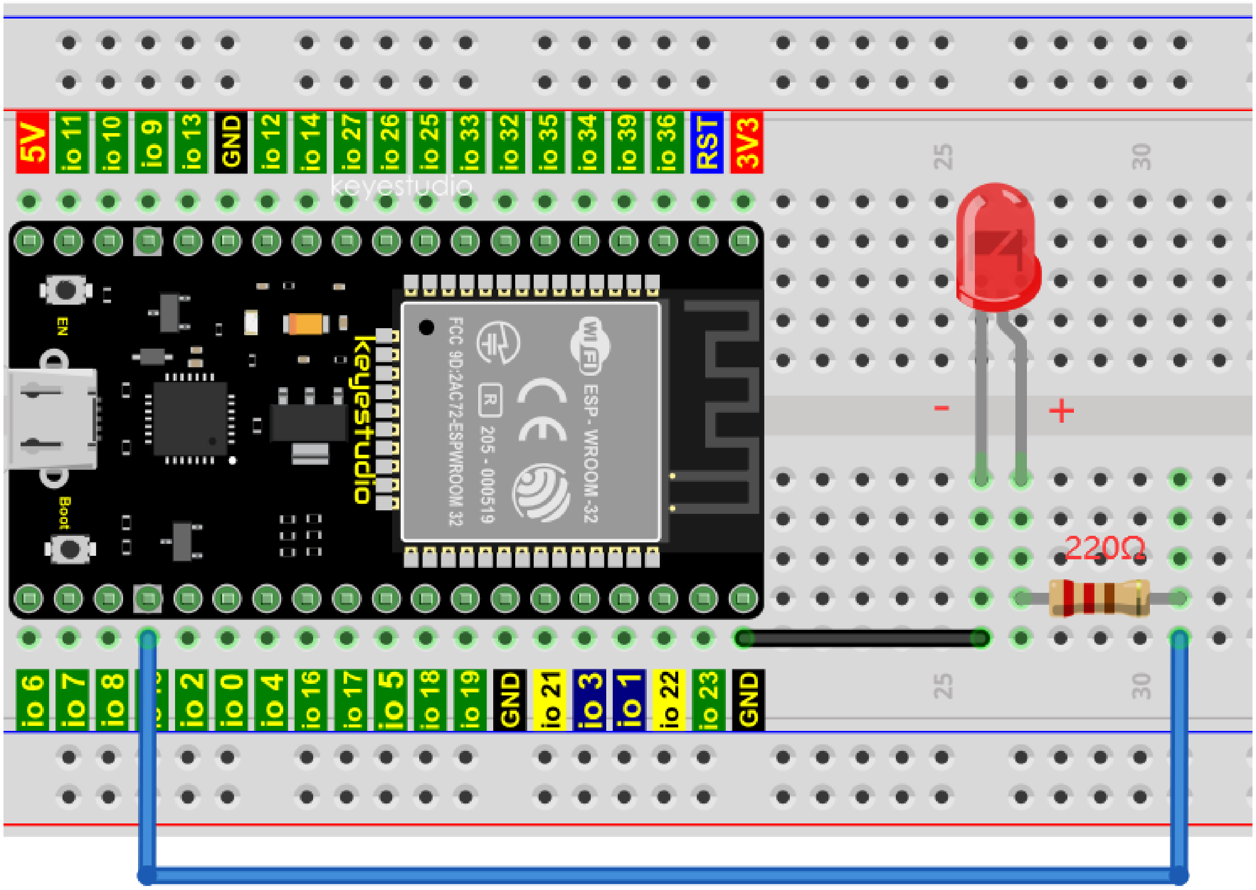


The longer the PWM duty cycle is, the higher the output power will be. Now that we understand this relationship, we can use PWM to control the brightness of an LED or the speed of DC motor and so on. It is evident from the above that PWM is not real analog, and the effective value of the voltage is equivalent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

ESP32 and PWM:

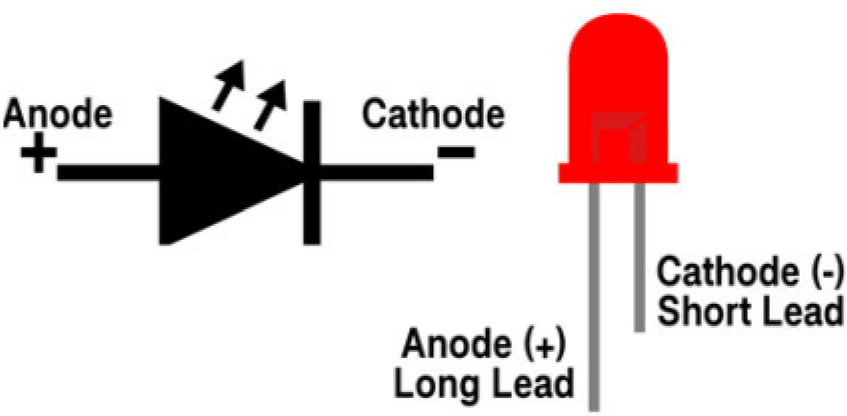
The ESP32 PWM controller has 8 independent channels, each of which can independently control frequency, duty cycle, and even accuracy. Unlike traditional PWM pins, the PWM output pins of ESP32 are configurable and they can be configured to PWM.

4.Wiring diagram

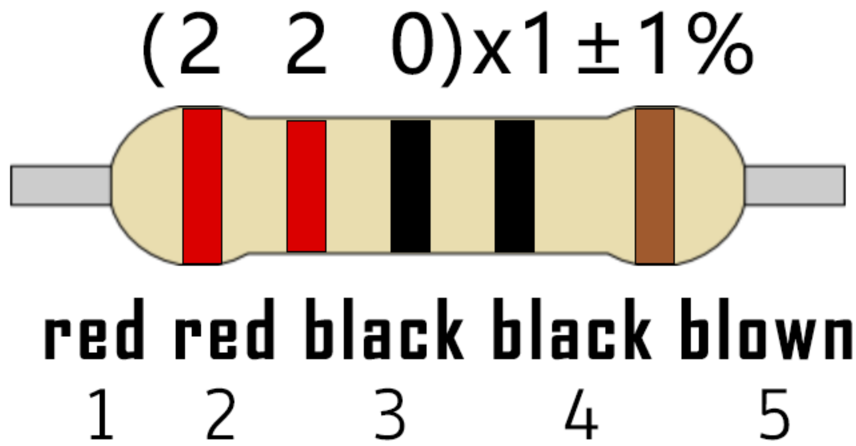


Note:

How to connect a LED



How to identify the 220 Five-color ring resistor

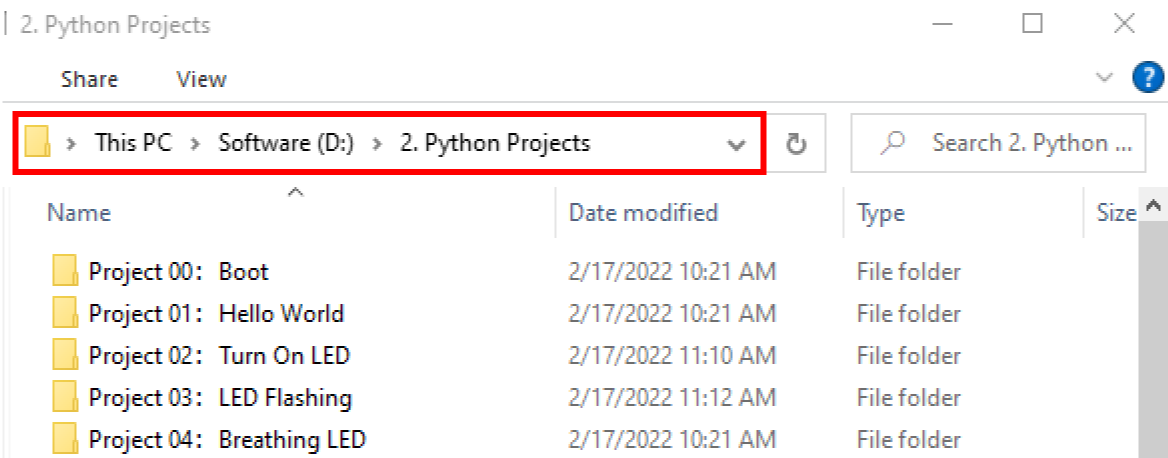


5.Project code

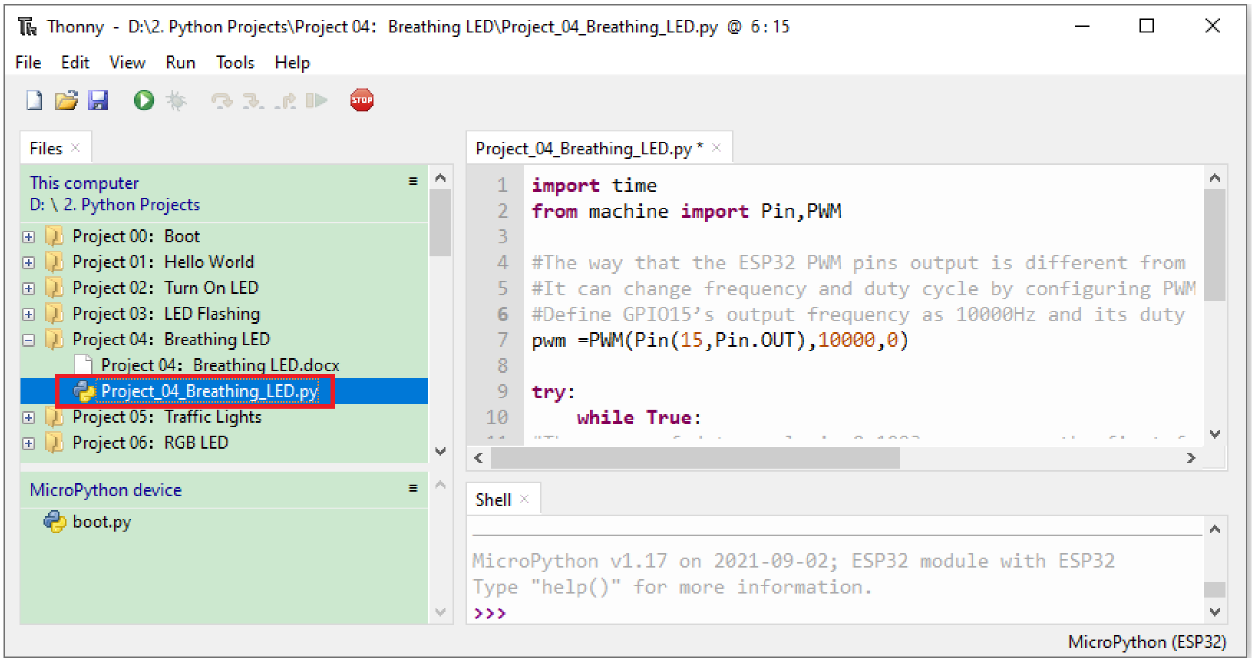
The design of this project makes the GP15 output PWM, and the pulse width gradually increases from 0% to 100%, and then gradually decreases from 100% to 0%.

Codes used in this tutorial are saved in“2. Python Projects”.

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



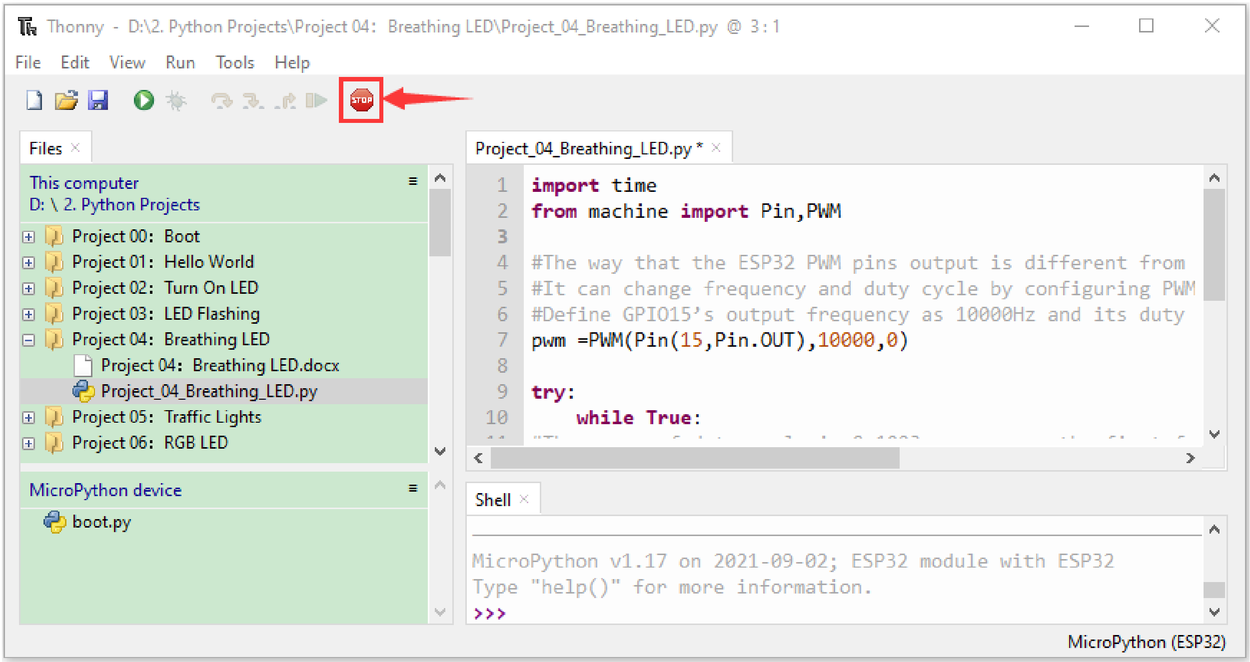
Open“Thonny”click“This computer”→“D:”→“2. Python Projects”→”Project 04Breathing Led”, and double left-click “Project_04_Breathing_LED.py”.



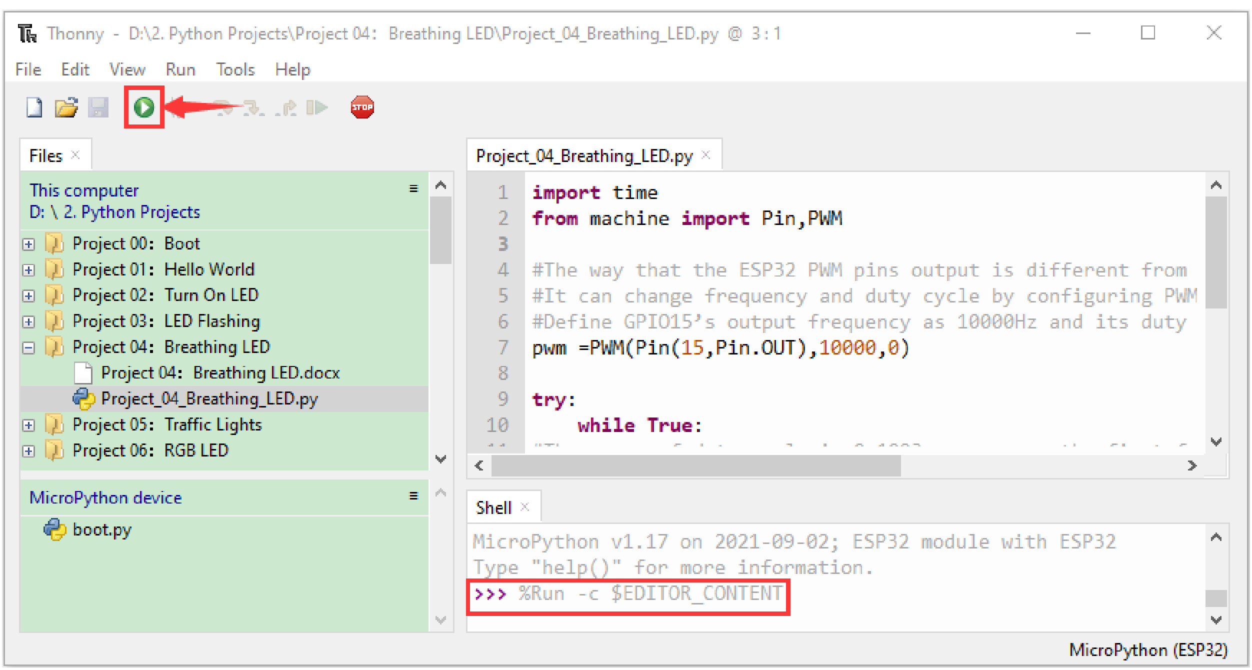

```python
import time
from machine import Pin,PWM

#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()
```

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.

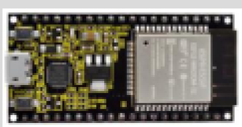
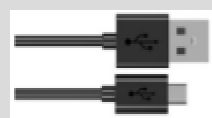
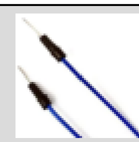
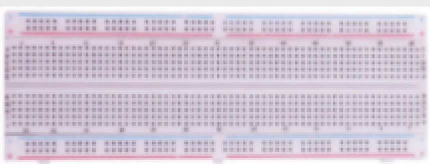
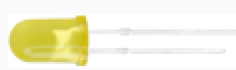
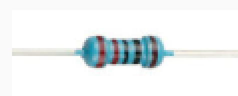


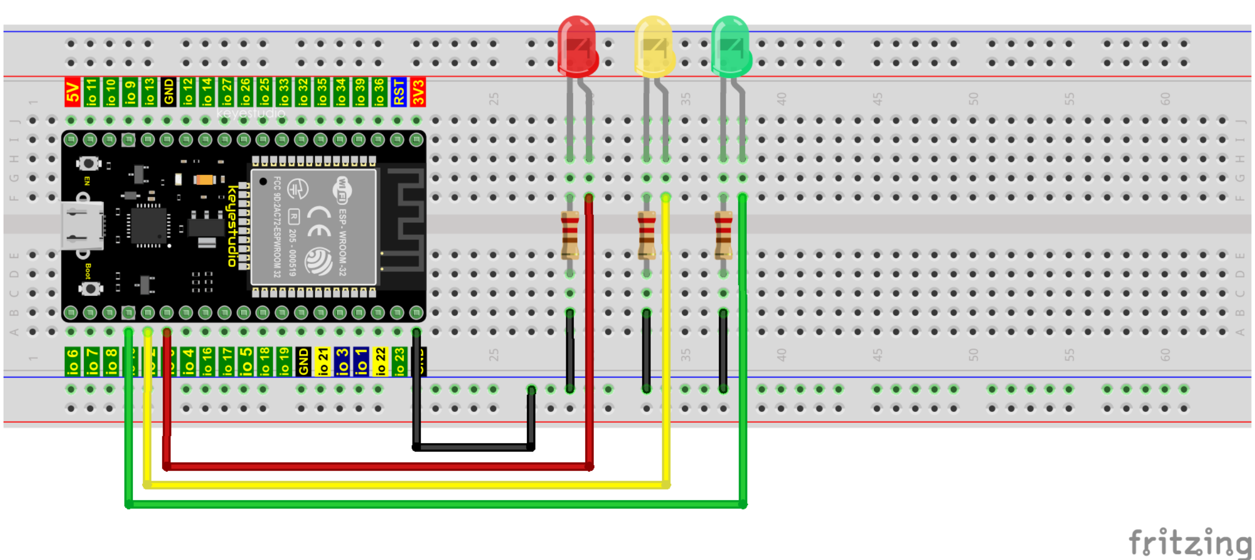
7.7 Project 05Traffic Lights

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.

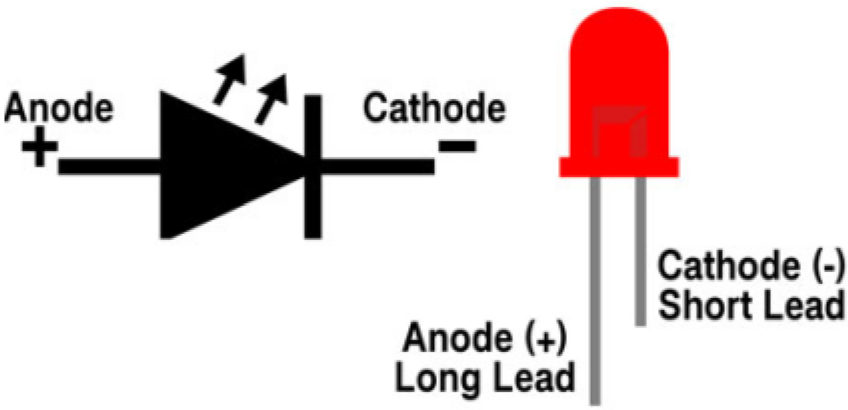
2.Components

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

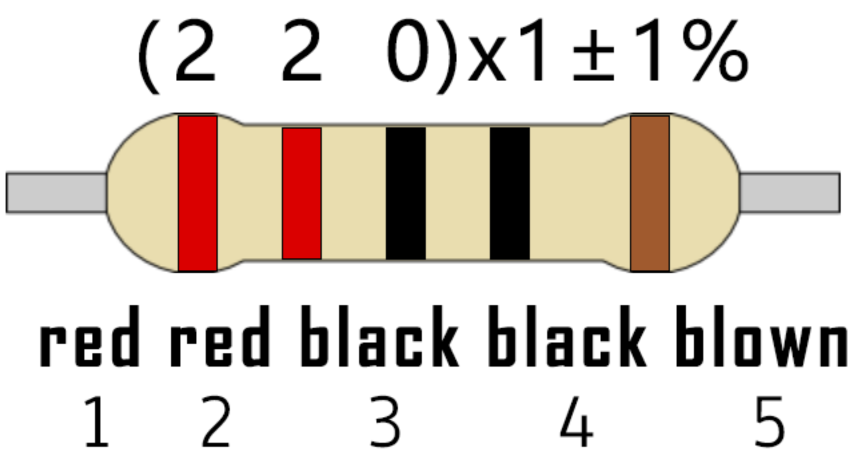


Note:

How to connect a LED



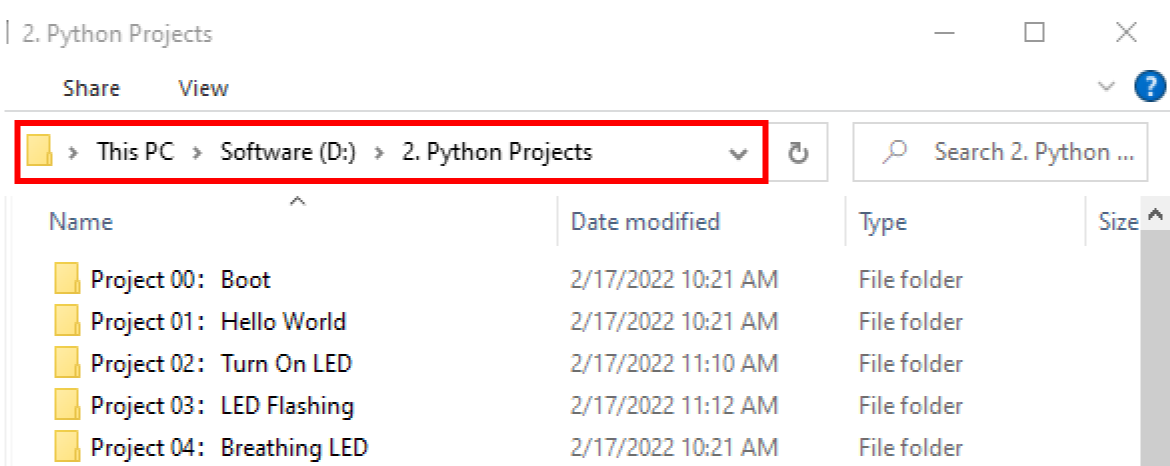
How to identify the 220 Five-color ring resistor



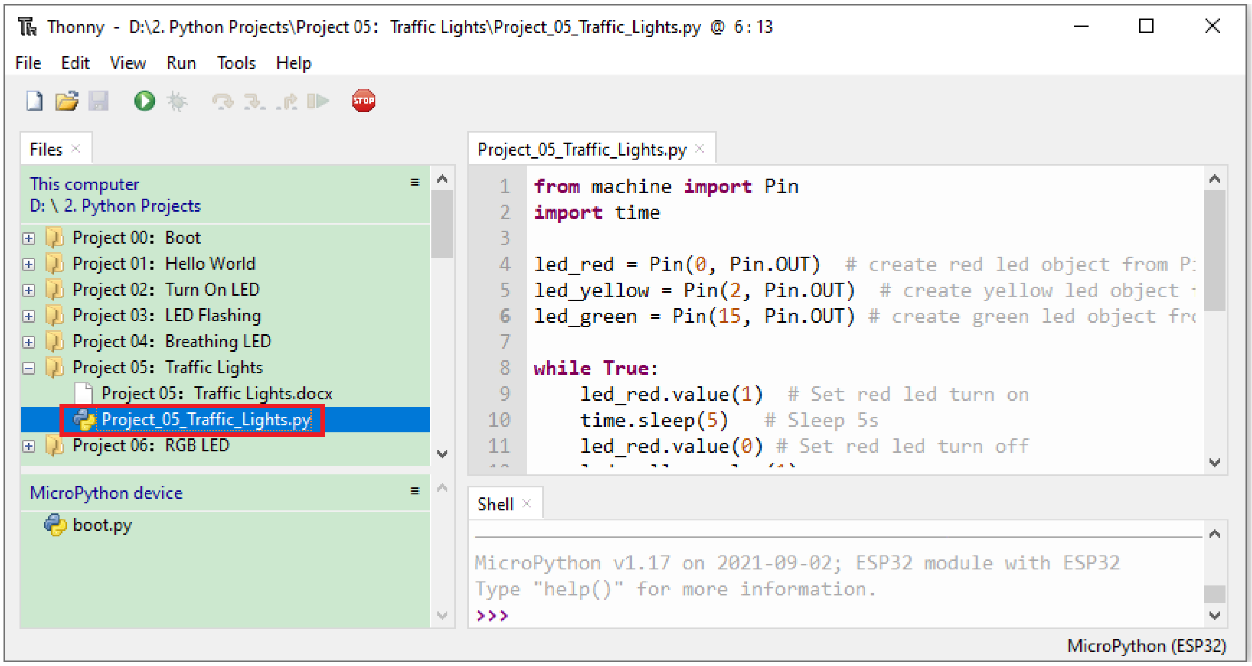
4.Project code

Codes used in this tutorial are saved in"2. Python Projects".

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



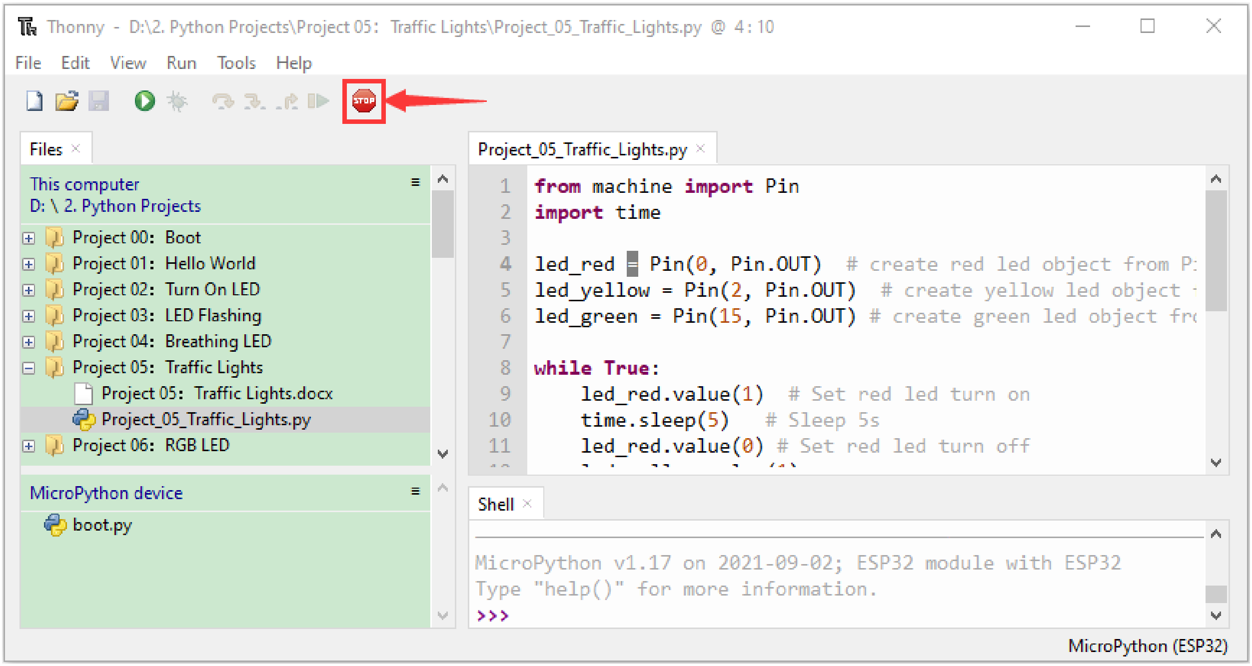
Open"Thonny"click"This computer"→"D:"→"2. Python Projects"→"Project 05Traffic Lights". and double left-click "Project_05_Traffic_Lights.py".




```
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)
```

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.

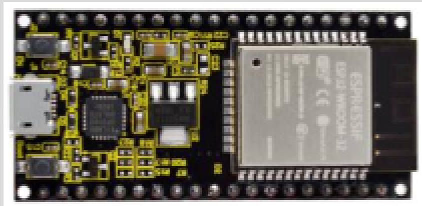
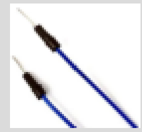
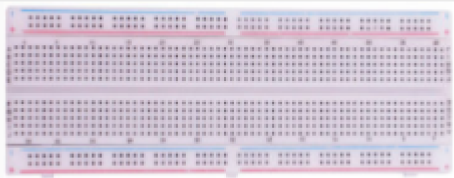
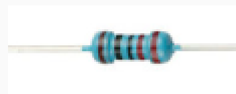
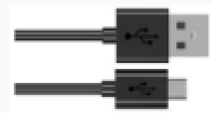
7.8 Project 06: RGB LED

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.

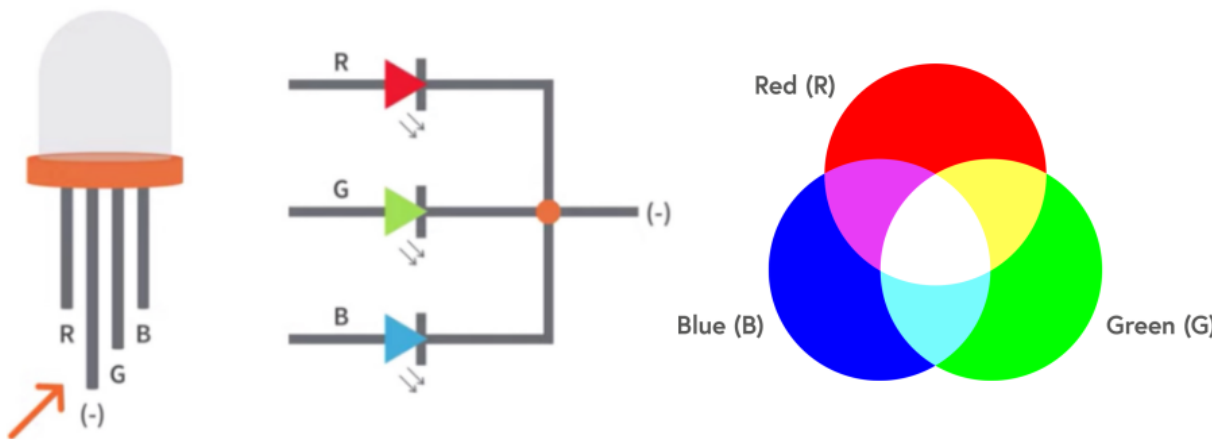
In this project, we will introduce the RGB and show you how to use ESP32 to control the RGB to emit different color light. RGB is pretty basic, but it's also a great way to learn the fundamentals of electronics and coding.

2.Components

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

3.Component knowledge

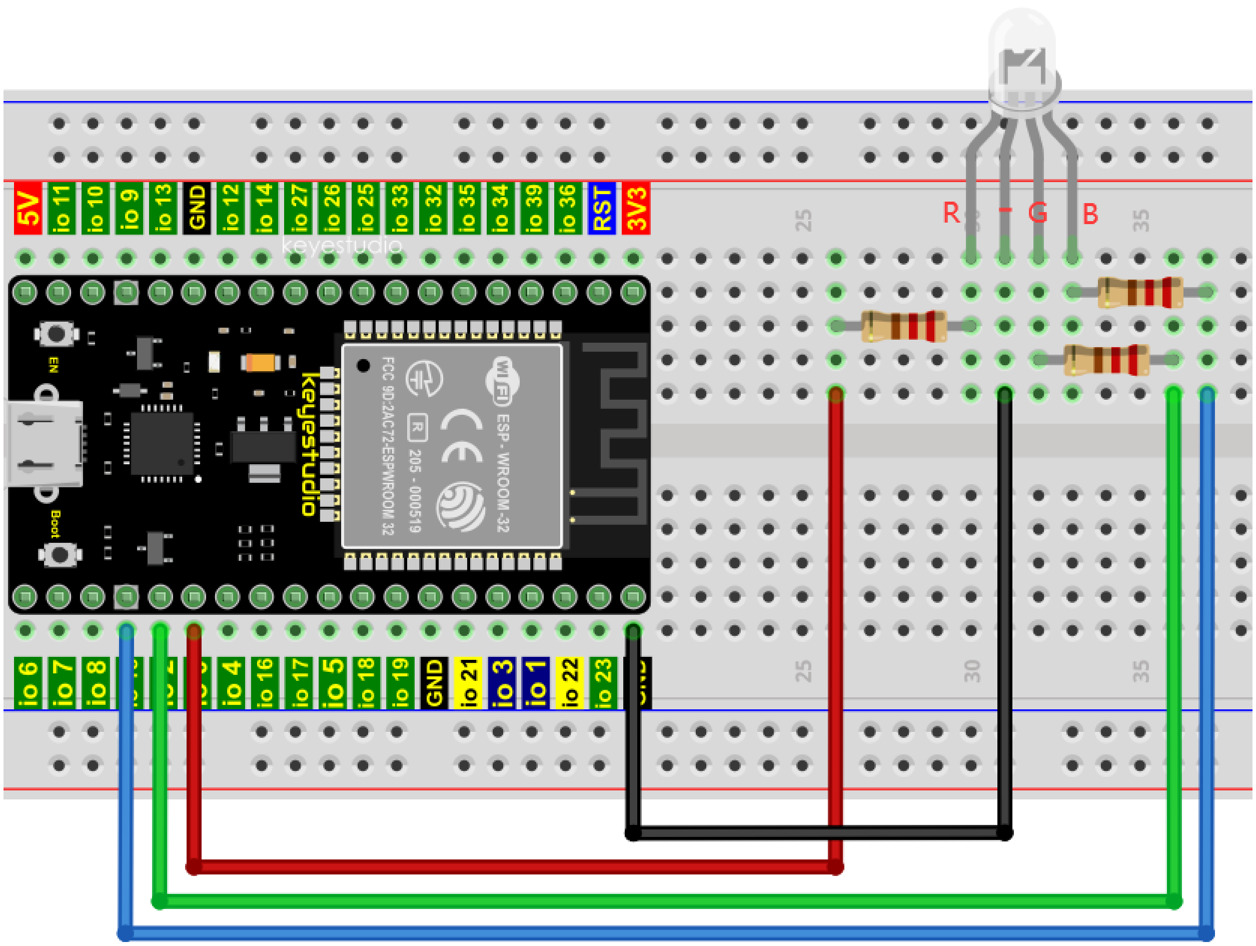
Most monitors adopt the RGB color standard, and all colors on a computer screen are a mixture of red, green and blue in varying proportions.



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.

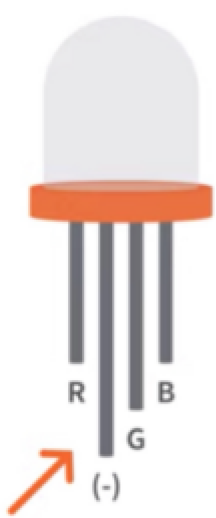
If we use three 10-bit PWM to control the RGB, in theory, we can create 2 10*210*210 = 1,073,741,824(1 billion) colors through different combinations.

4.Wiring diagram

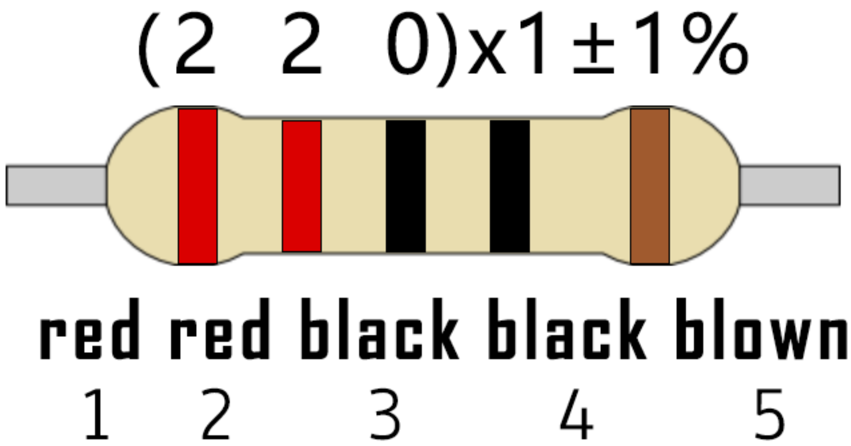


Note

The longest pin (common cathode) of the RGB LED is connected to GND.



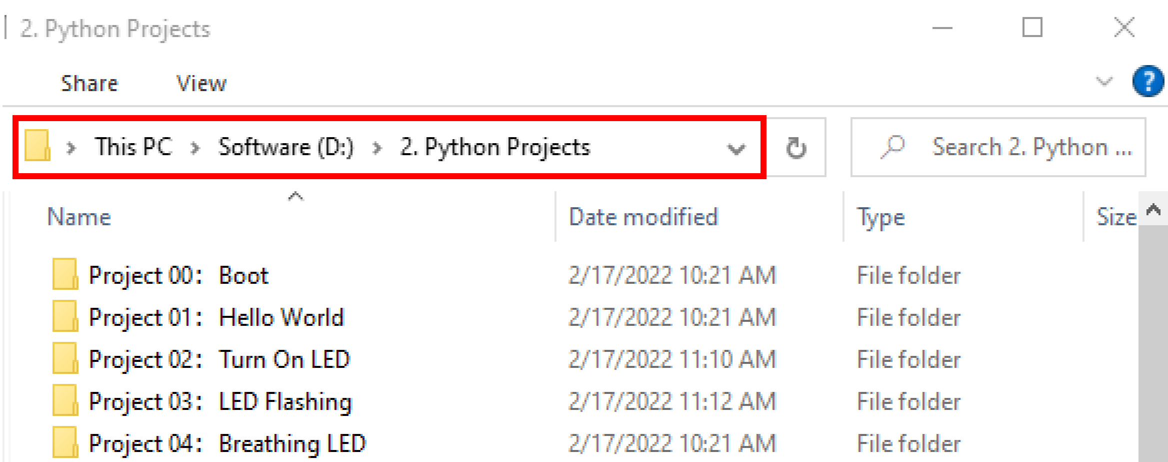
How to identify the 220 Five-color ring resistor



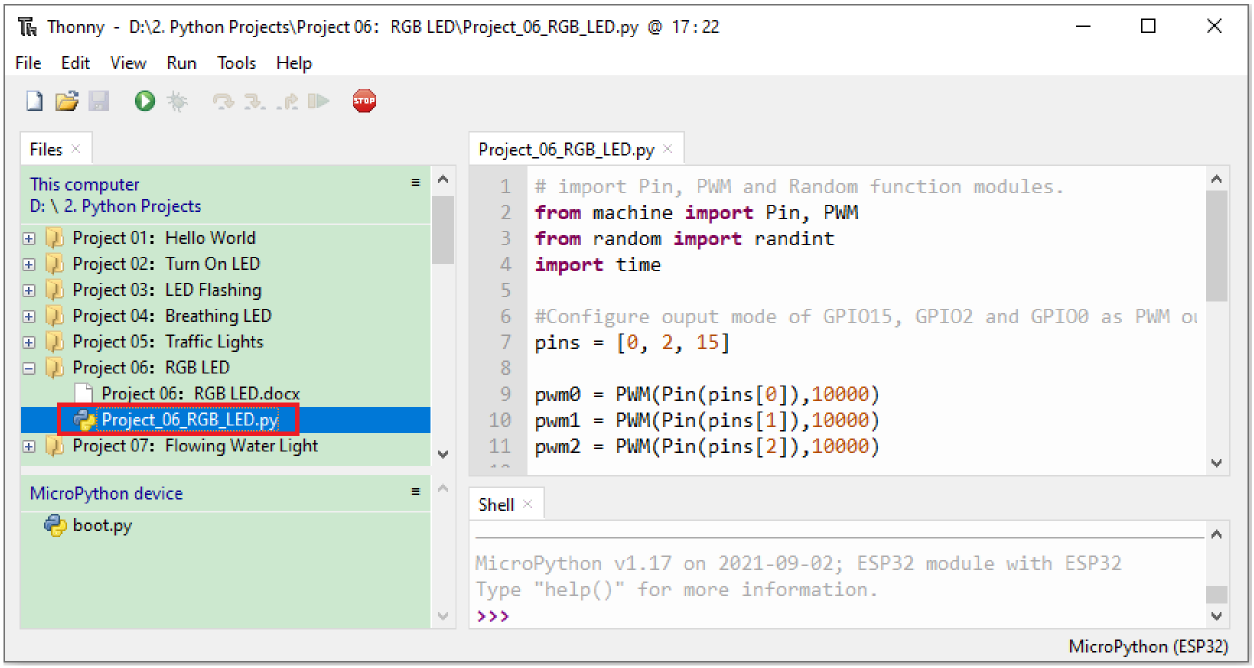
5.Project code

Codes used in this tutorial are saved in“2. Python Projects”.

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



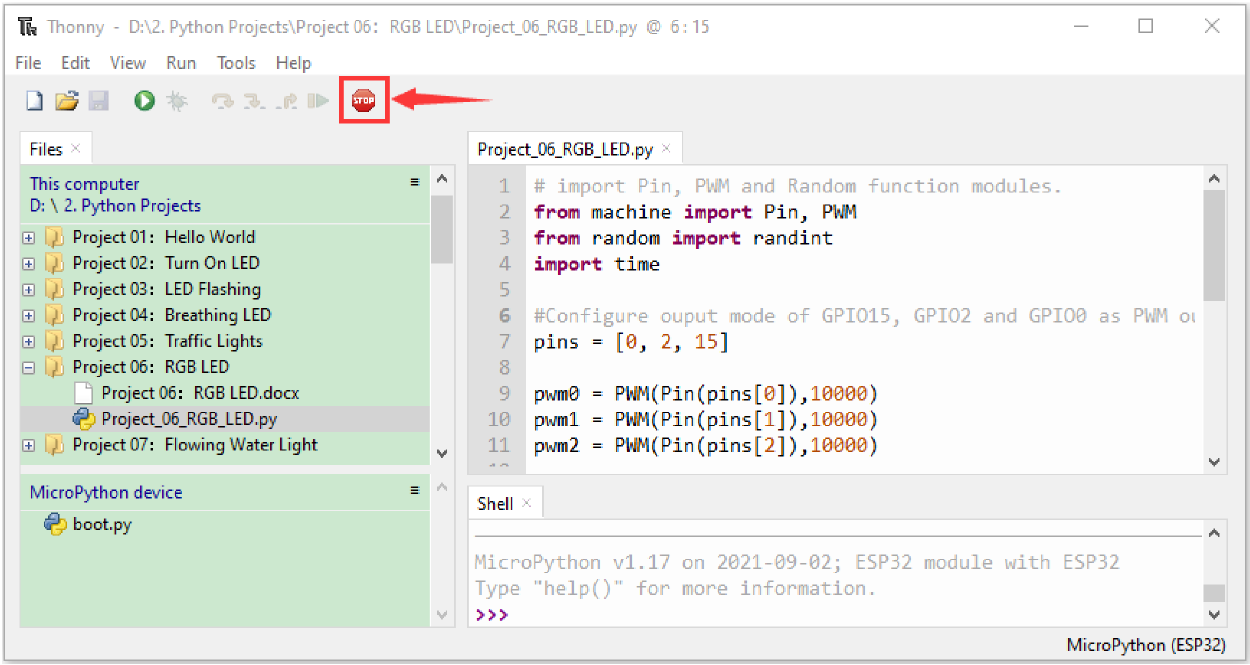
Open“Thonny”click“This computer”→“D:”→“2. Python Projects”→“Project 06RGB LED”, and double left-click “Project_06_RGB_LED.py”.



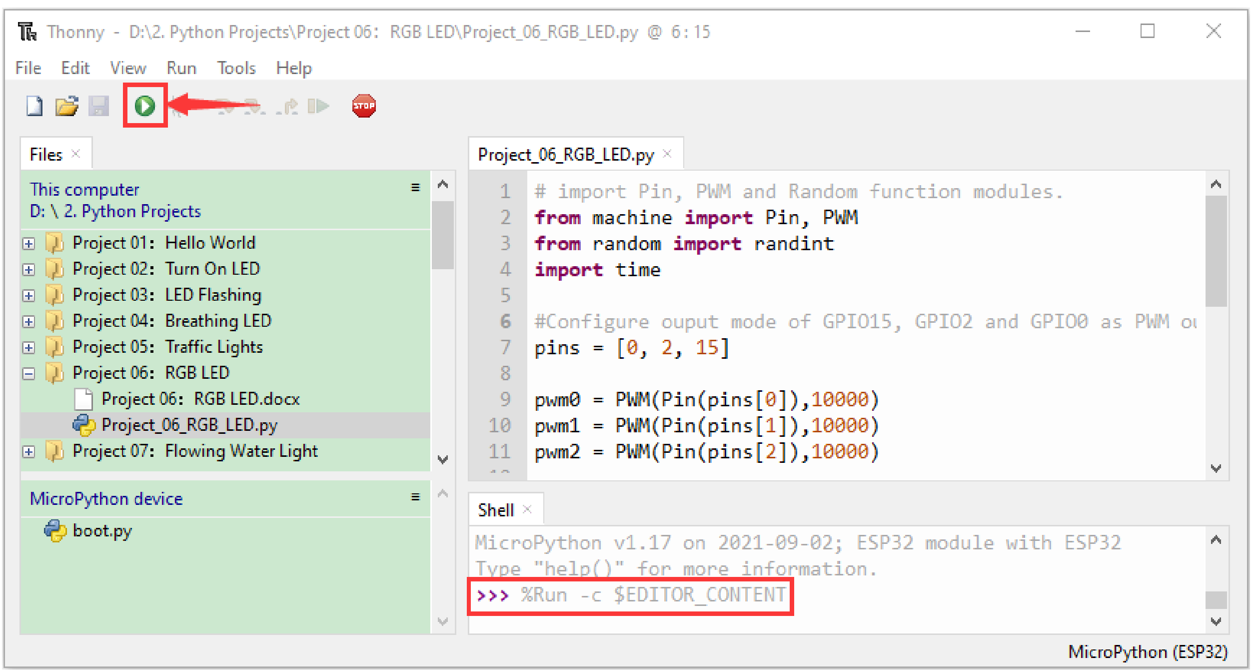

```python
# import Pin, PWM and Random function modules.
from machine import Pin, PWM
from random import randint
import time

#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()
```

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

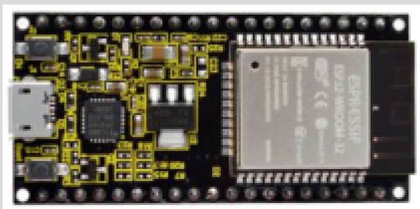
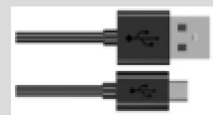
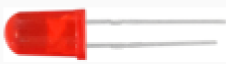
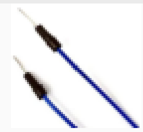


7.9 Project 07: Flowing Water Light

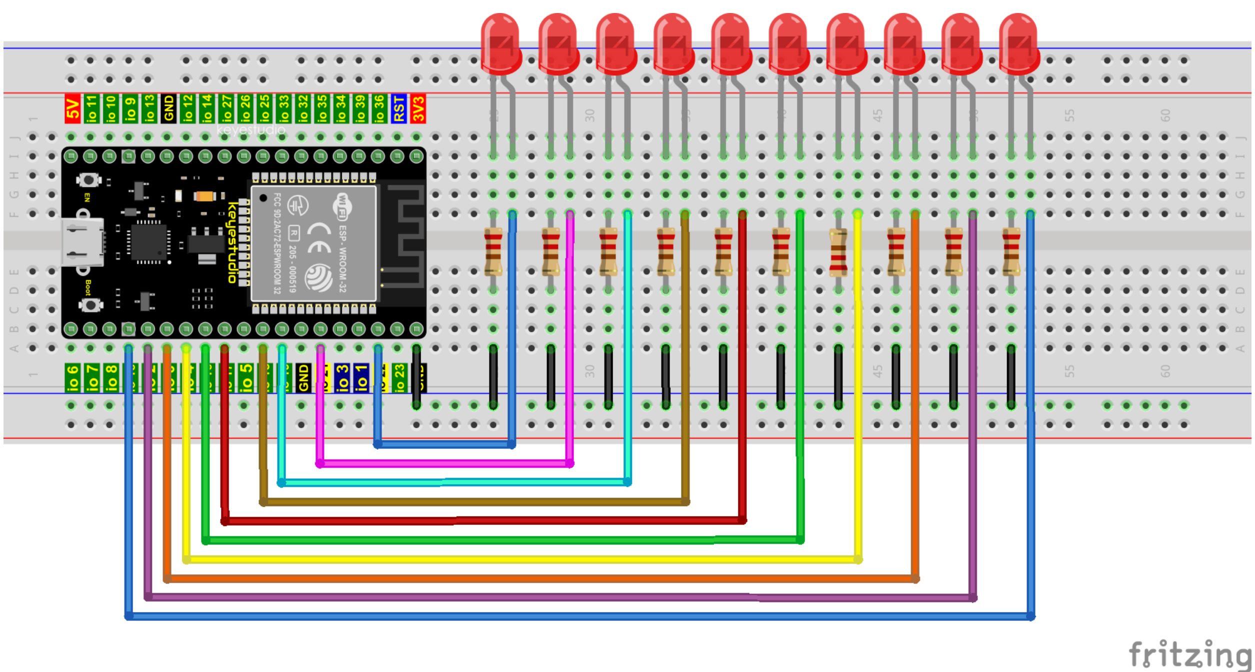
1.Introduction

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

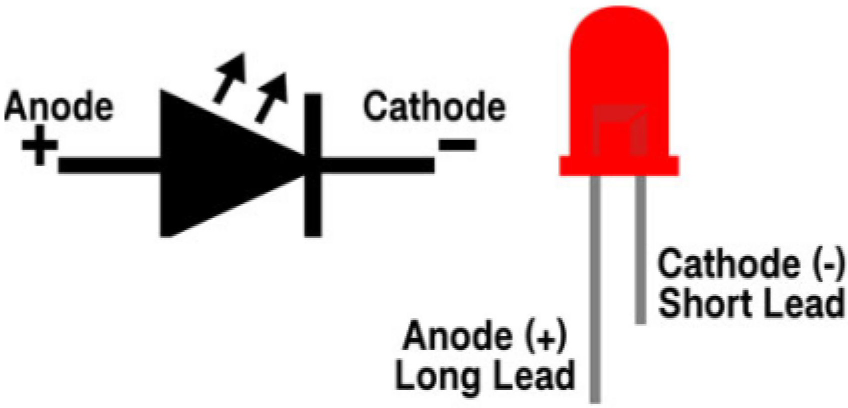
		
ESP32*1	Breadboard*1	USB Cable*1
		
Red LED*1	220 Resistor*1	Jumper Wire*2

3.Wiring diagram :

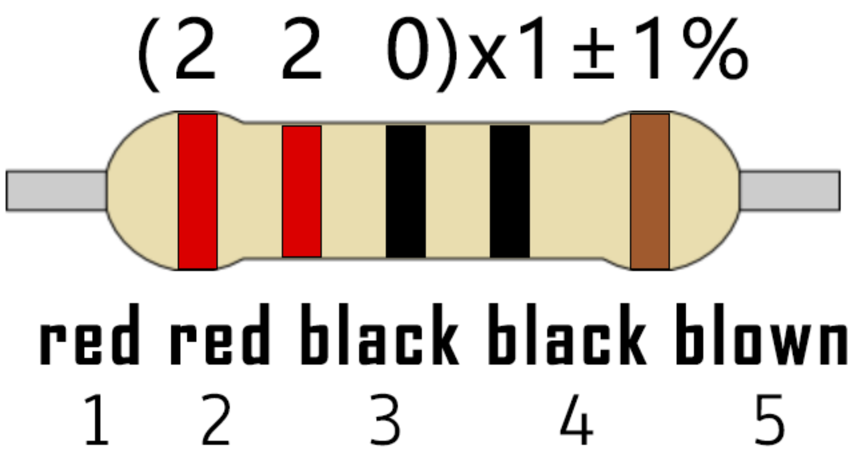


Note:

How to connect a LED



How to identify the 220 Five-color ring resistor

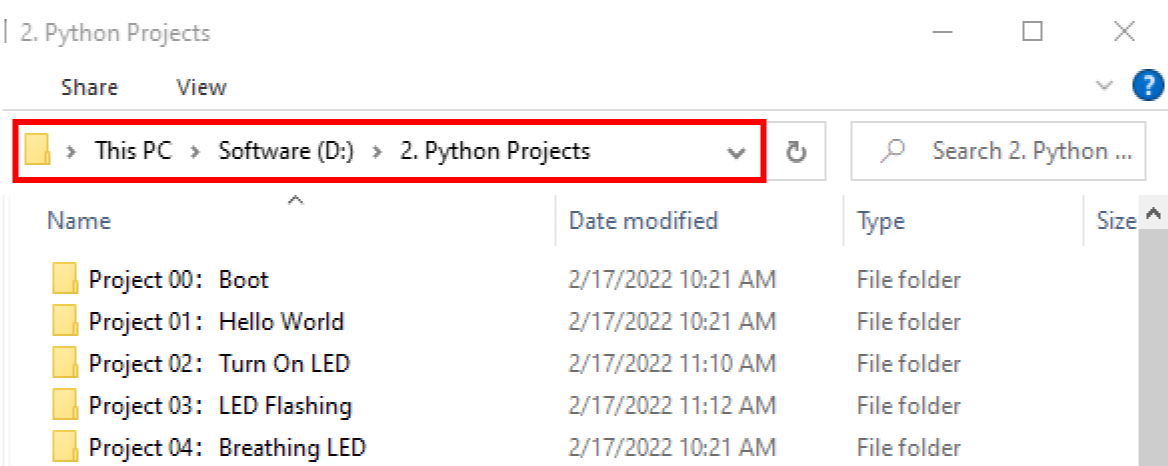


4.Project code

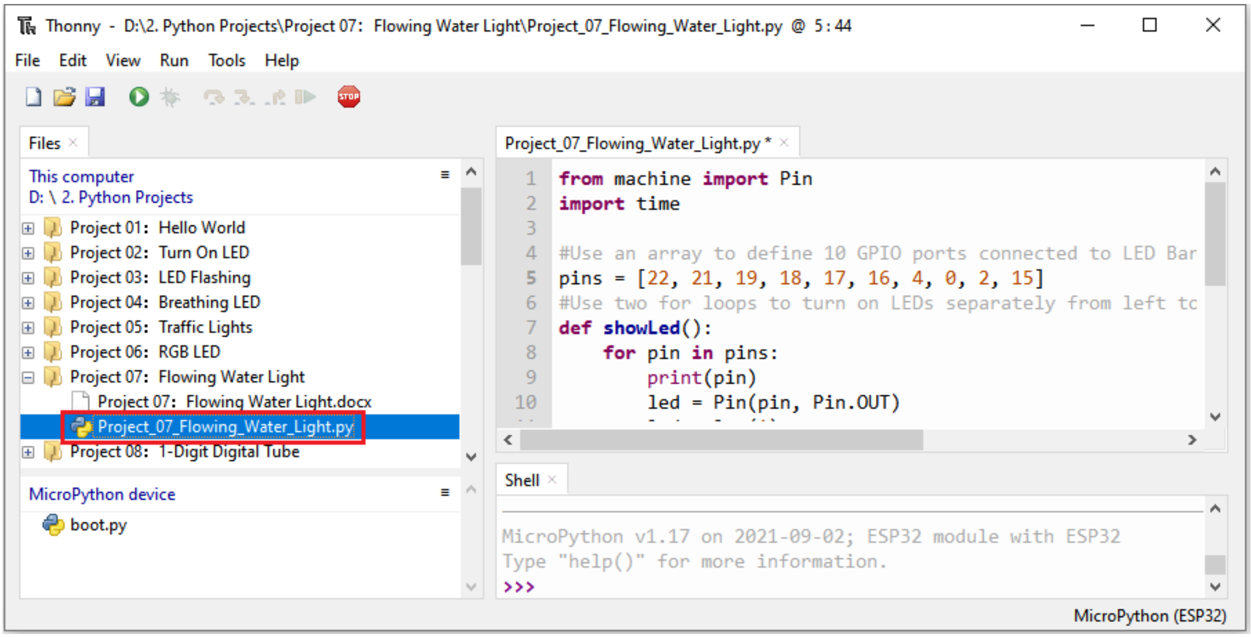
This project is designed to make a flowing water lamp. Which are these actions: First turn LED #1 ON, then turn it OFF. Then turn LED #2 ON, and then turn it OFF… and repeat the same to all 10 LEDs until the last LED is turns OFF. This process is repeated to achieve the“movements” of flowing water.

Codes used in this tutorial are saved in“2. Python Projects”.

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



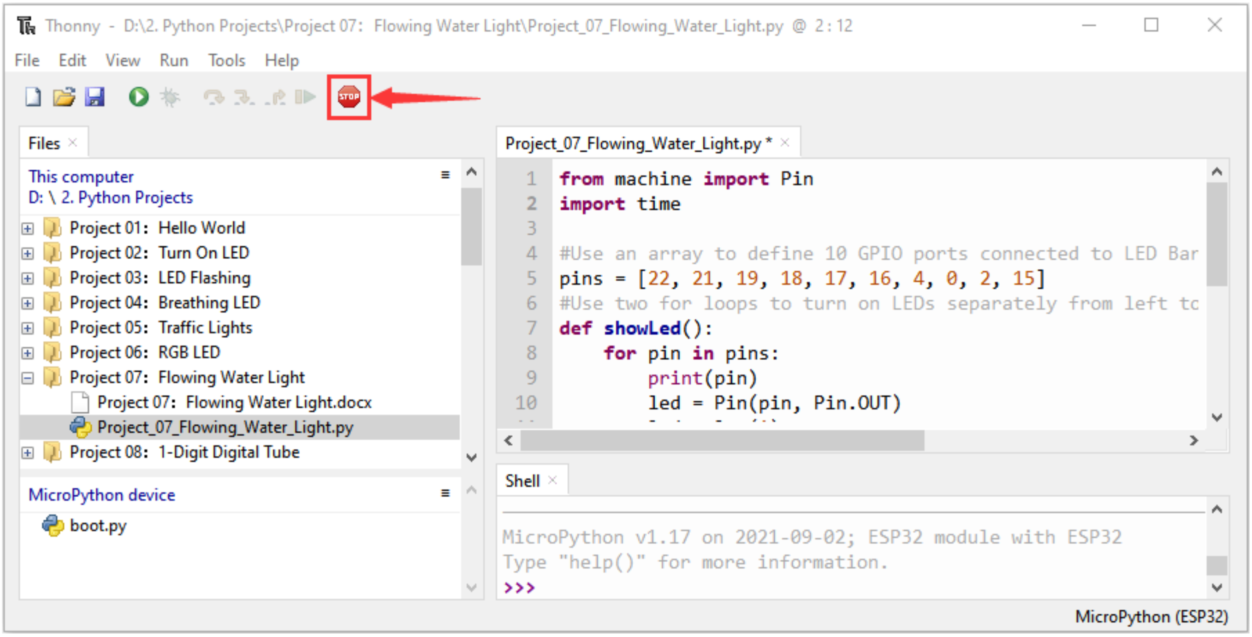
Open“Thonny”click“This computer”→“D:”→“2. Python Projects”→“Project 07Flowing Water Light”, and double left-click “Project_07_Flowing_Water_Light.py”.



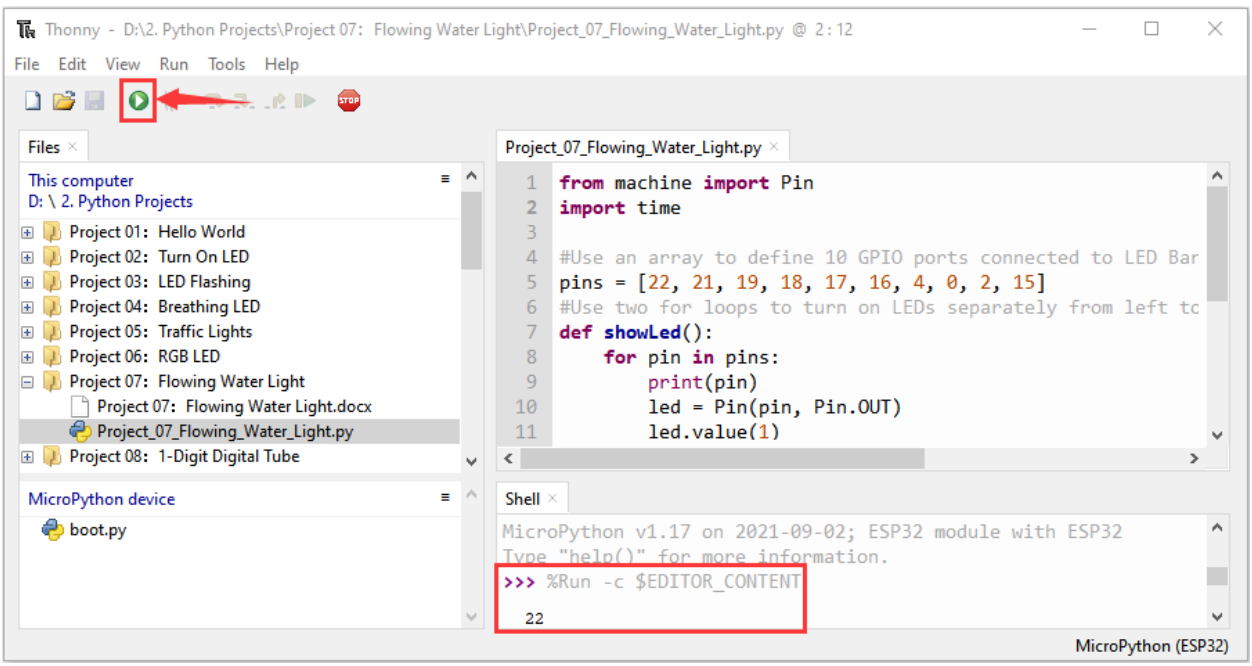

```
from machine import Pin
import time

#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()
```

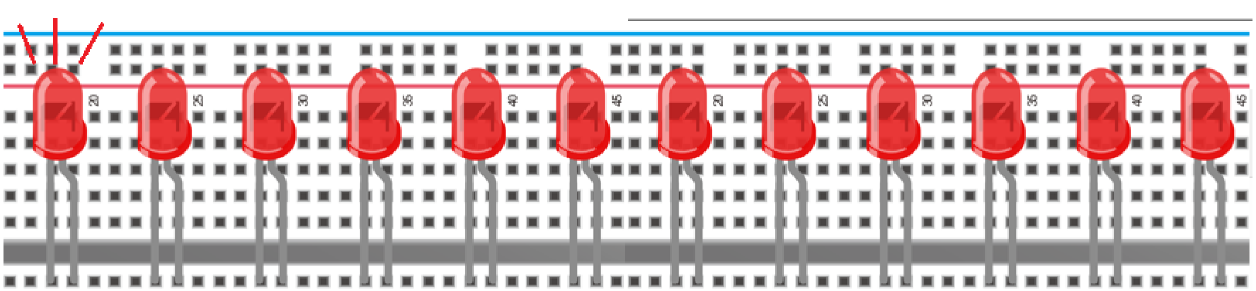
5.Project result

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.



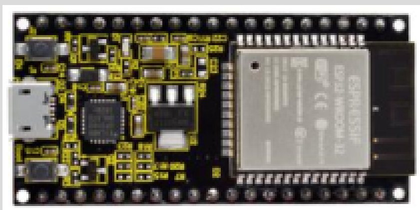
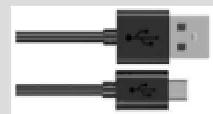
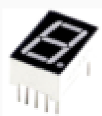
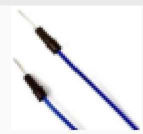


7.10 Project 081-Digit Digital Tube

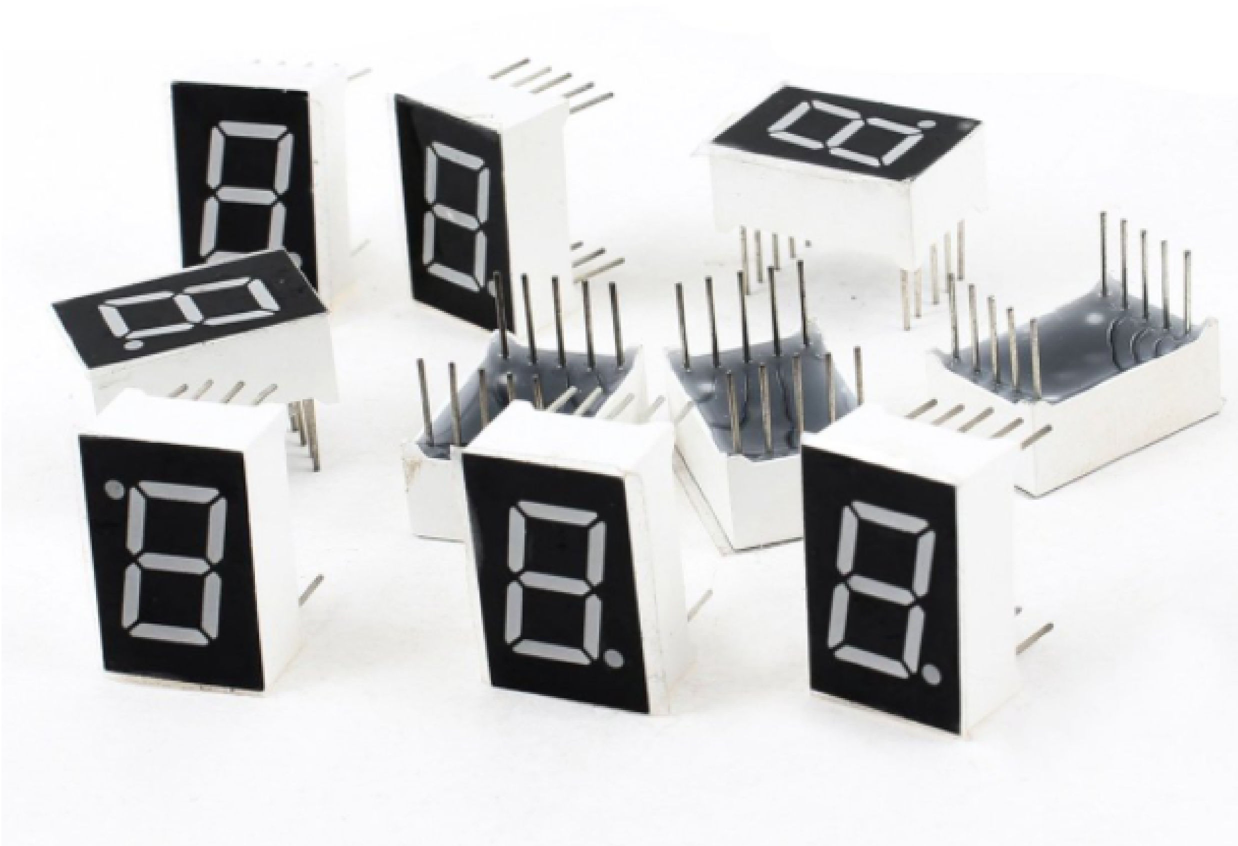
1.Introduction

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. Even though 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 to display numbers.

2.Components

		
ESP32*1	Breadboard*1	USB Cable*1
		
1-Digit 7-Segment Display*1	220 Resistor*8	Jumper Wire*2

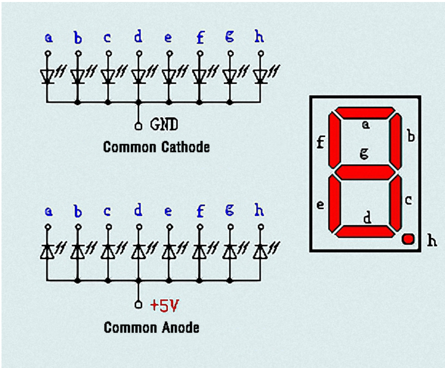
3.Component knowledge



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".

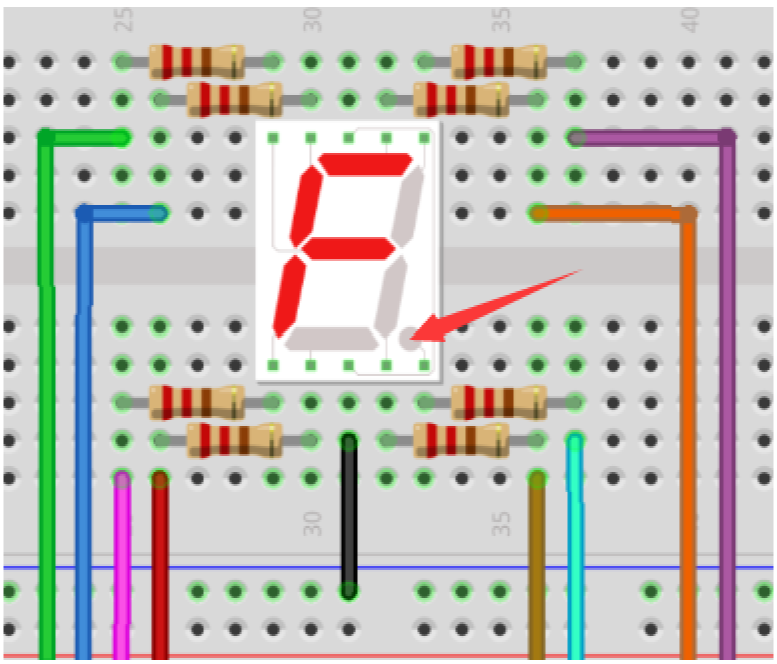
In the common anode 7-segment display, the LED anodes (positive electrodes) of all segments are connected together, so you should connect the common anode to"+5V". To light up a segmented LED, you can set its associated pin to"LOW".

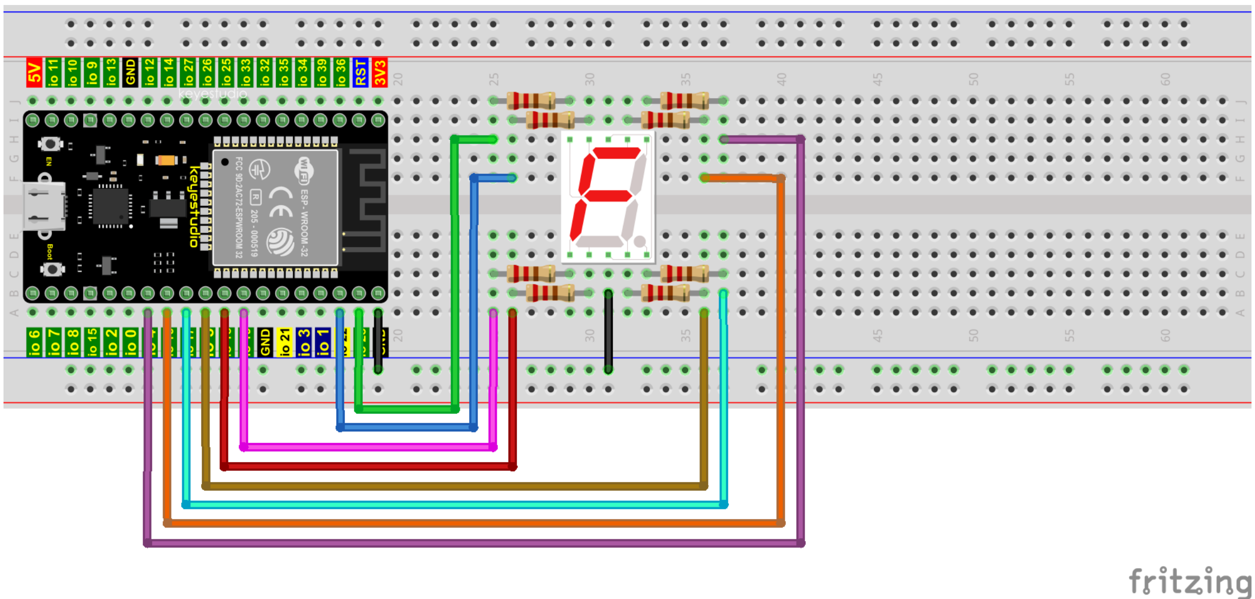


Each part of the digital tube is composed of an LED. So when you use it, you also need to use a current limiting resistor. Otherwise, the LED will be damaged. In this experiment, we use an ordinary common cathode one-digit digital tube. As we mentioned above, you should connect the common cathode to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

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.

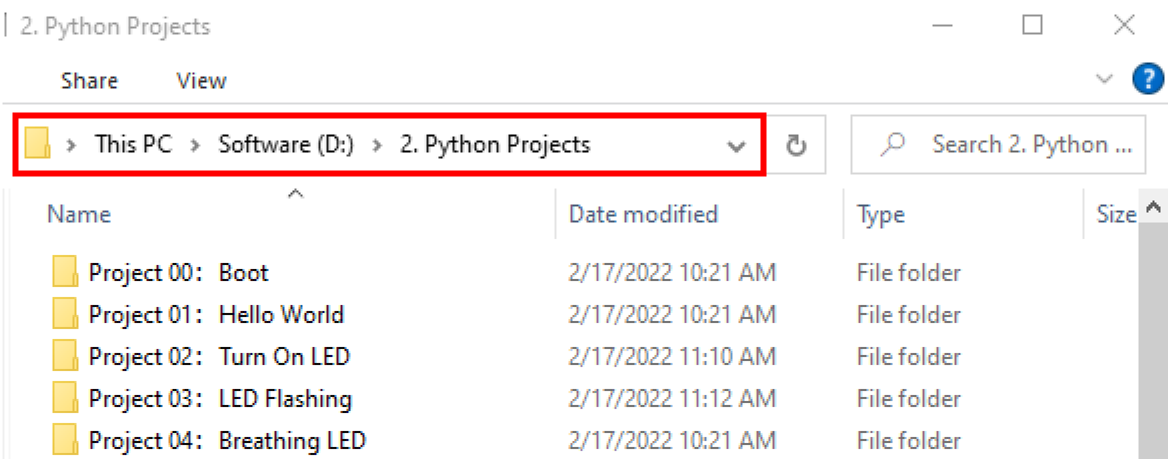




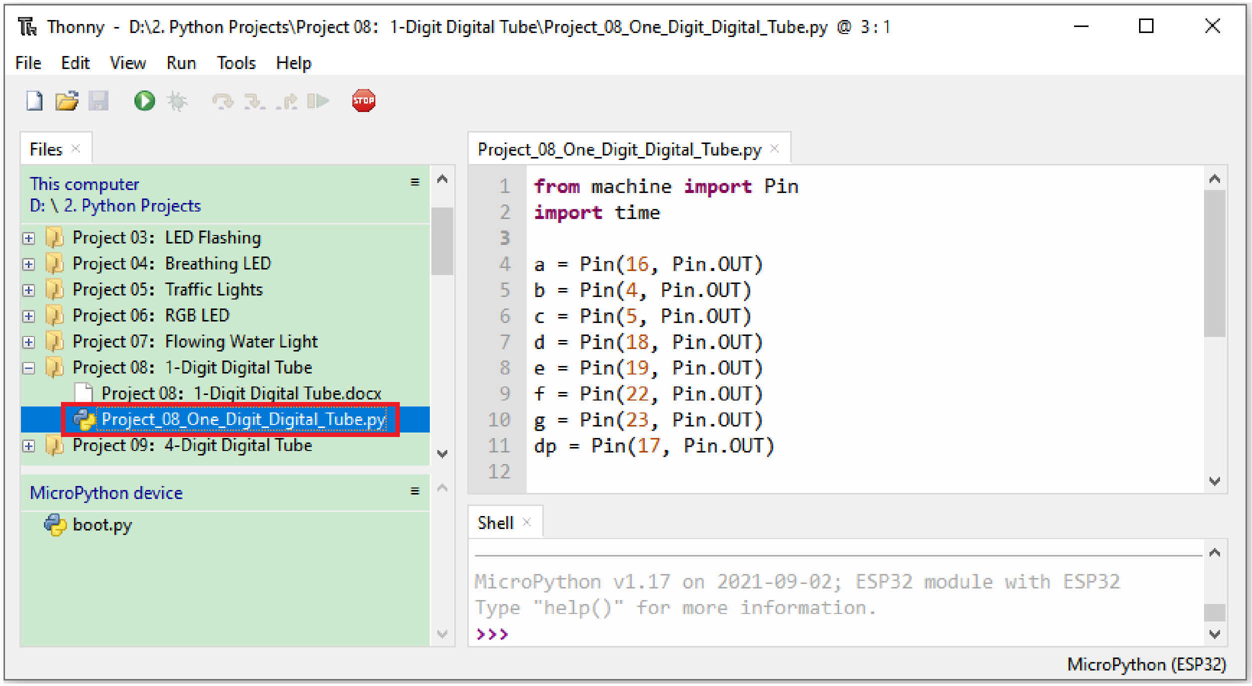
5.Project code

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.

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



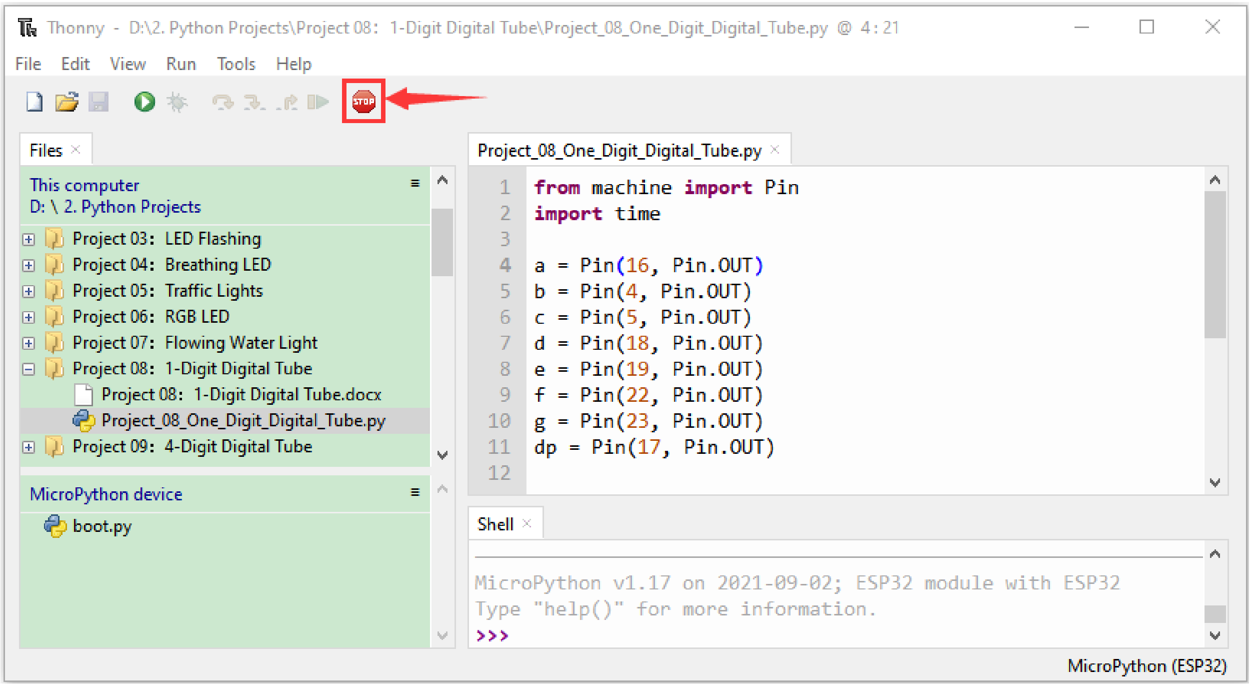
Open“Thonny”click“This computer”→“D:”→“2. Python Projects”→“Project 08: 1-Digit Digital Tube”, and double left-click “Project_08_One_Digit_Digital_Tube.py”.



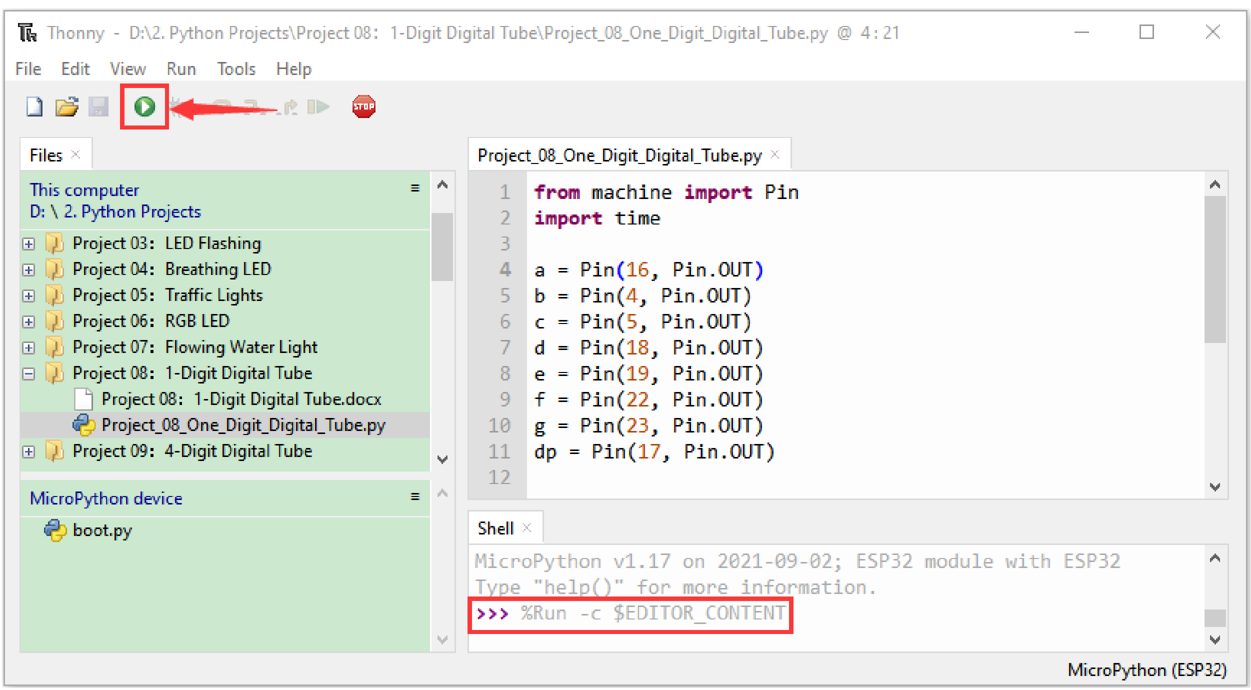

```
from machine import Pin
import time

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

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.

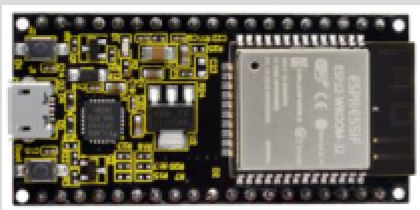
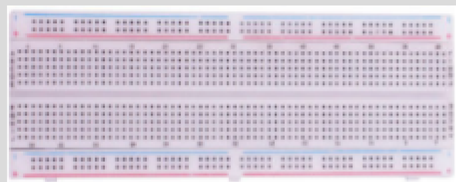
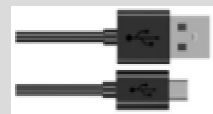
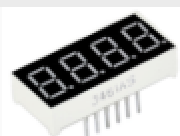
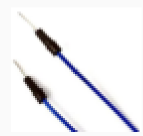


7.11 Project 094-digit Digital Tube

1.Introduction

The 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 the 4 Digit 7-segment display. In this project, we use ESP32 to control the 4-digit 7-segment display to display digits.

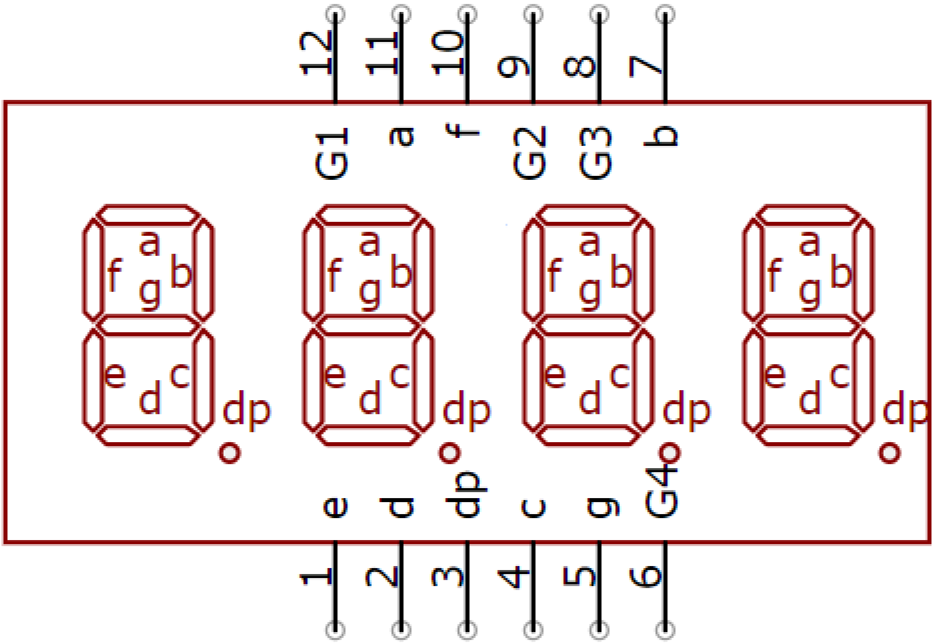
2.Components

		
ESP32*1	Breadboard*1	USB Cable*1
		
4-digit 7-segment display Module*1	220 Resistor*8	Jumper Wire*2

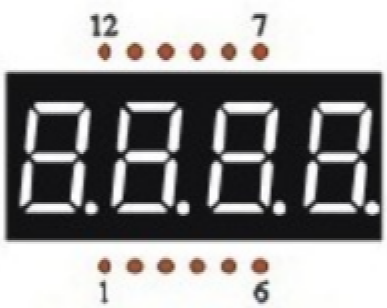
3.Component Knowledge

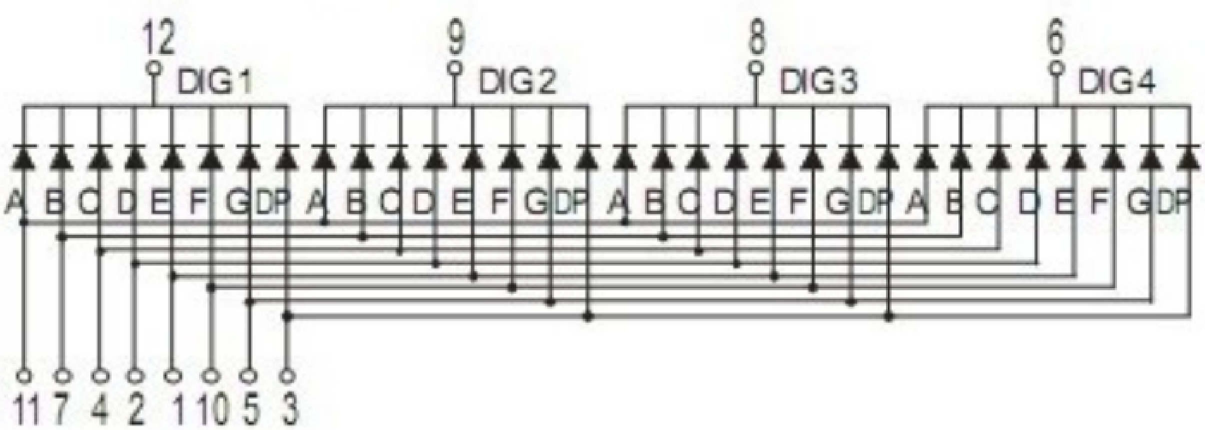
4-digit 7-segment display It is a device with common cathode and anode, its display principle is similar to the 1-Digit digital tube display. Both of them have eight GPIO ports to control the digital tube display, that is 8 leds. However, here is 4-digit, so it needs four GPIO ports to control the bit selection end. Our 4 - digit digital tube is common cathode.

The following figure shows the pin diagram of the 4-digit digital tube. G1, G2, G3 and G4 are the control pins.

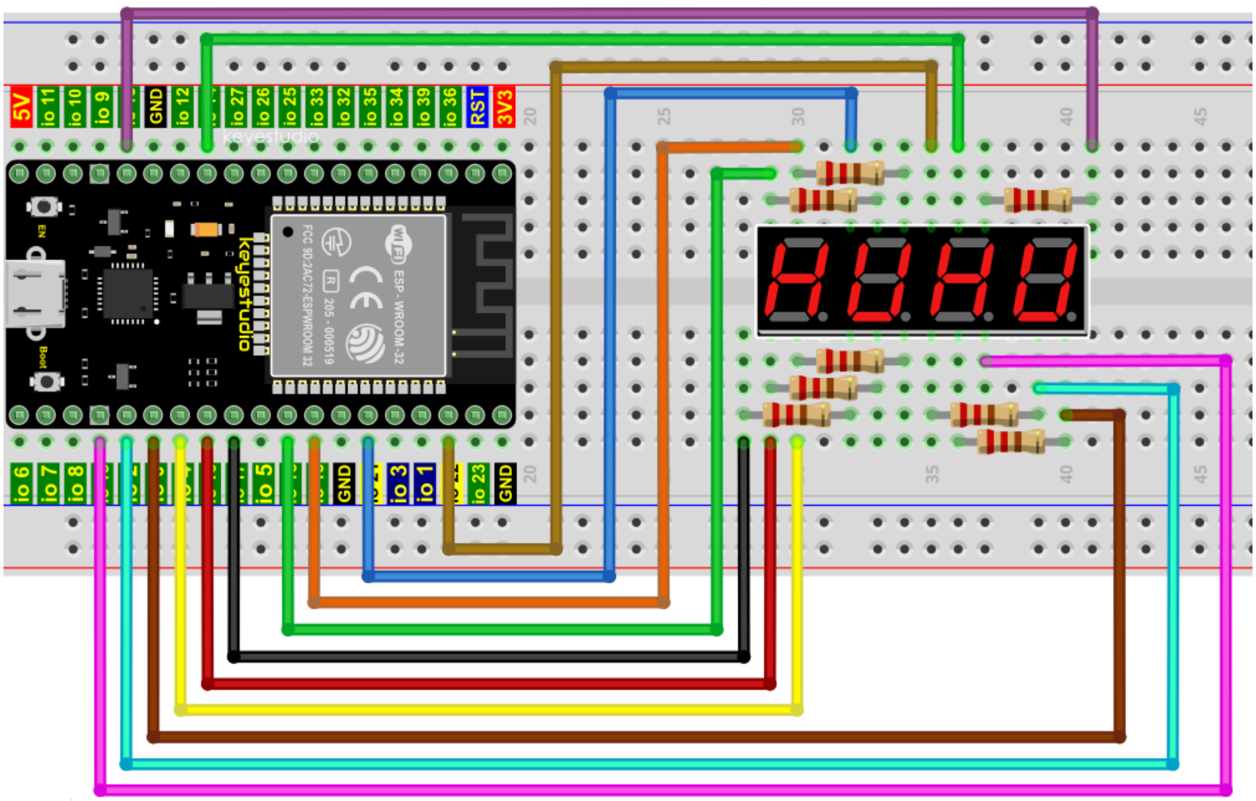


Schematic Diagram



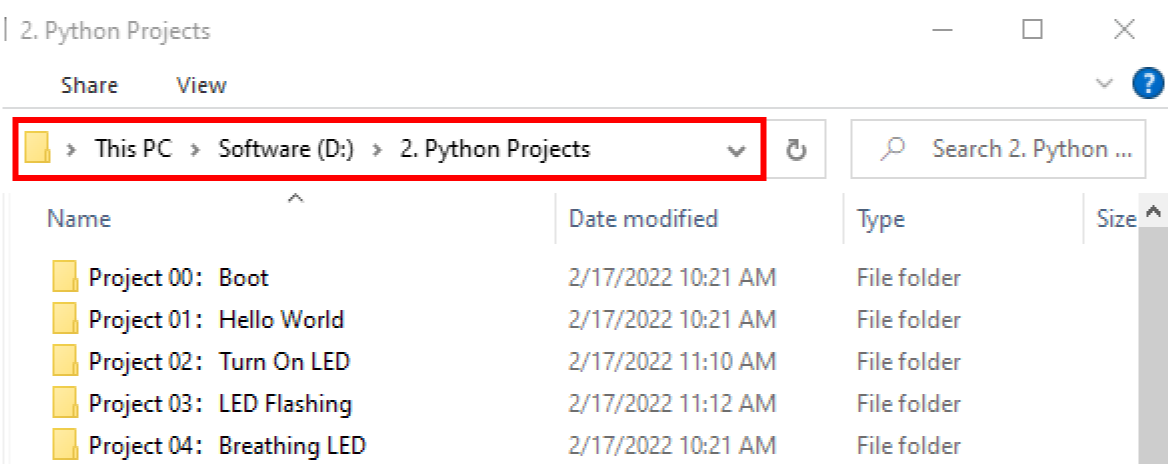


4.Wiring Diagram

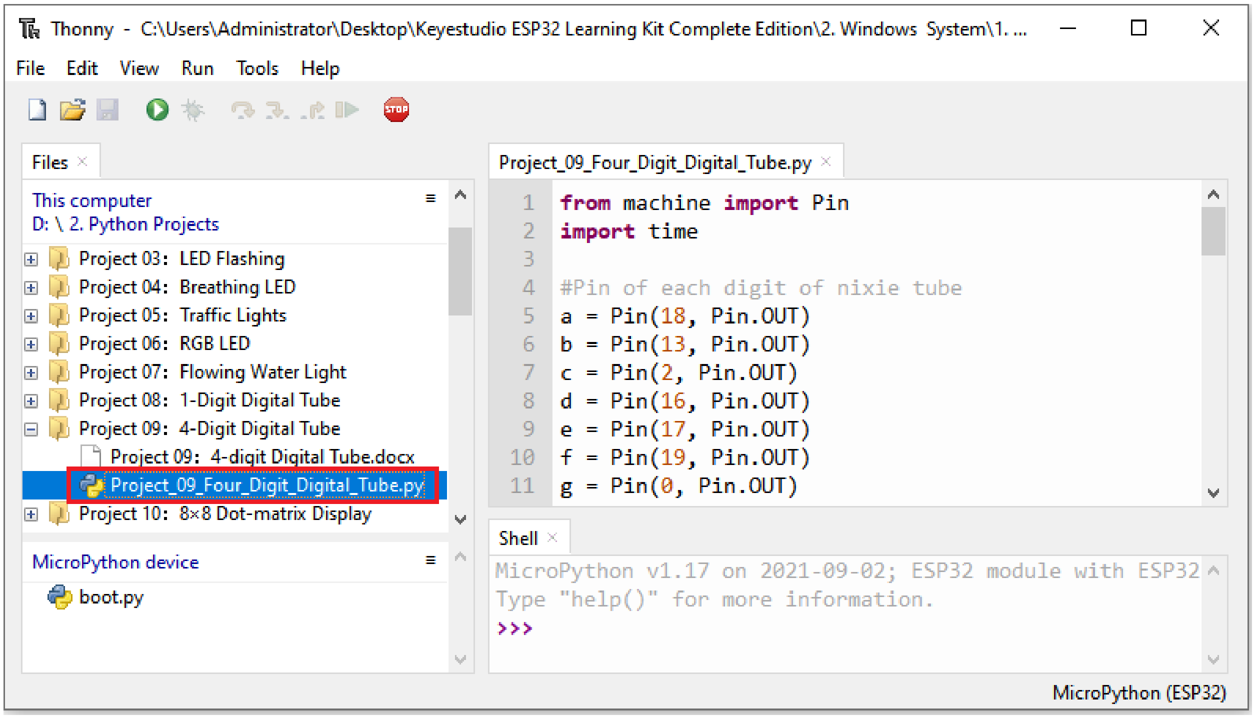


5.Test Code

Codes used in this tutorial are saved in“2. Python Projects”. If you haven’t downloaded the code file yet, 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”, then double left-click “Project_09_Four_Digit_Digital_Tube.py”.




```
from machine import Pin
import time

#Pin of each digit of nixie tube
a = Pin(18, Pin.OUT)
b = Pin(13, Pin.OUT)
c = Pin(2, Pin.OUT)
d = Pin(16, Pin.OUT)
e = Pin(17, Pin.OUT)
f = Pin(19, Pin.OUT)
g = Pin(0, Pin.OUT)
dp = Pin(4, Pin.OUT)

G1 = Pin(21, Pin.OUT)
G2 = Pin(22, Pin.OUT)
G3 = Pin(14, Pin.OUT)
G4 = Pin(15, Pin.OUT)

#digital tube a to dp corresponding development board pins
d_Pins=[Pin(i,Pin.OUT)  for i in [18,13,2,16,17,19,0,4]]
#Pin corresponding to digital tube segment G1, G2, G3, and G4
w_Pins=[Pin(i,Pin.OUT)  for i in [21,22,14,15]]

number={
    '0':
    [1,1,1,1,1,1,0,0],#0
    '1':
    [0,1,1,0,0,0,0,0],#1
    '2':
    [1,1,0,1,1,0,1,0],#2
```

(continues on next page)

(continued from previous page)

```
    '3':
    [1,1,1,1,0,0,1,0],#3
    '4':
    [0,1,1,0,0,1,1,0],#4
    '5':
    [1,0,1,1,0,1,1,0],#5
    '6':
    [1,0,1,1,1,1,1,0],#6
    '7':
    [1,1,1,0,0,0,0,0],#7
    '8':
    [1,1,1,1,1,1,1,0],#8
    '9':
    [1,1,1,1,0,1,1,0],#9
}

def display(num,dp):
    global number
    count=0
    for pin in d_Pins:#displays the value of num
        pin.value(number[num][count])
        count+=1
    if dp==1:
        d_Pins[7].value(0)
def clear():
    for i in w_Pins:
        i.value(0)
    for i in d_Pins:
        i.value(1)
def showData(num):
   #the hundreds, thousands, ones, and fractional values of a numeric value
    d_num=num
    location=d_num.find('.')
    if location>0:
        d_num=d_num.replace('.','')
        while len(d_num)<4:
            d_num='0'+d_num
        for i in range(0,4):
            time.sleep(2)
            clear()
            w_Pins[3-i].value(1)
            if i==location-1:
                display(d_num[i],1)
            else:
                display(d_num[i],0)
    if location<0:
        for i in range(0,4):
            time.sleep(2)
            clear()
            w_Pins[3-i].value(1)
            display(d_num[i],0)
while True:
```

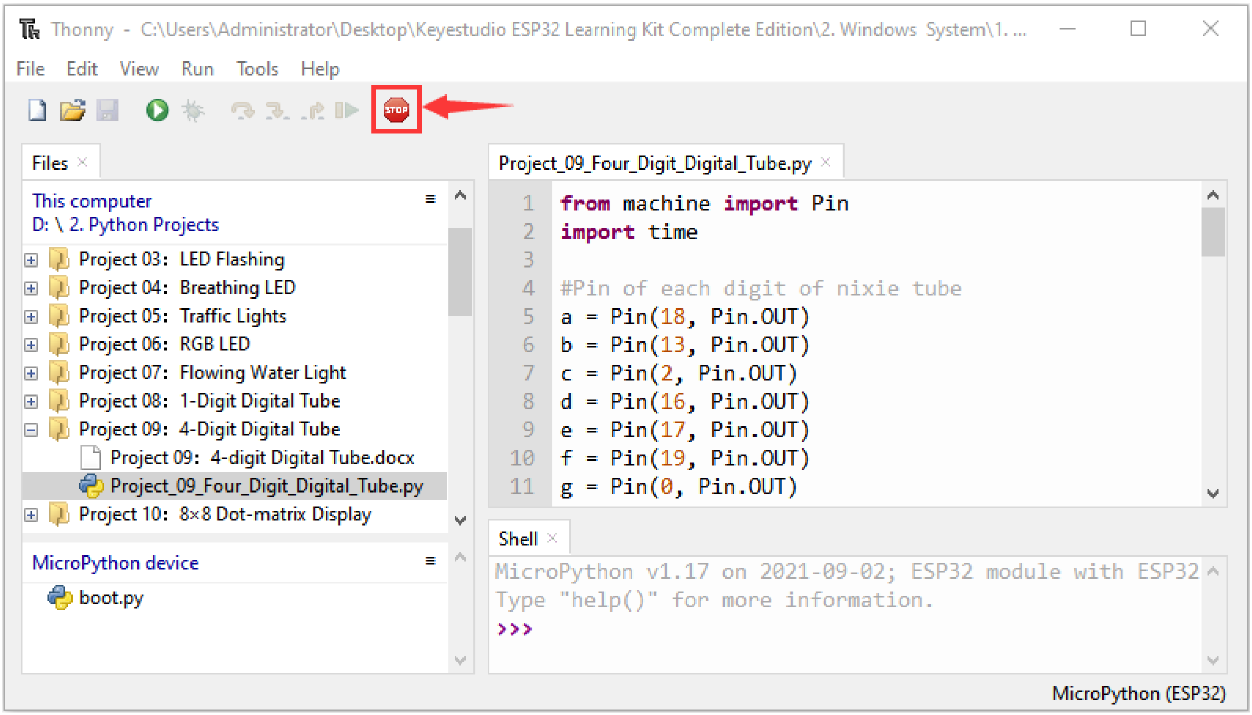
(continues on next page)

(continued from previous page)

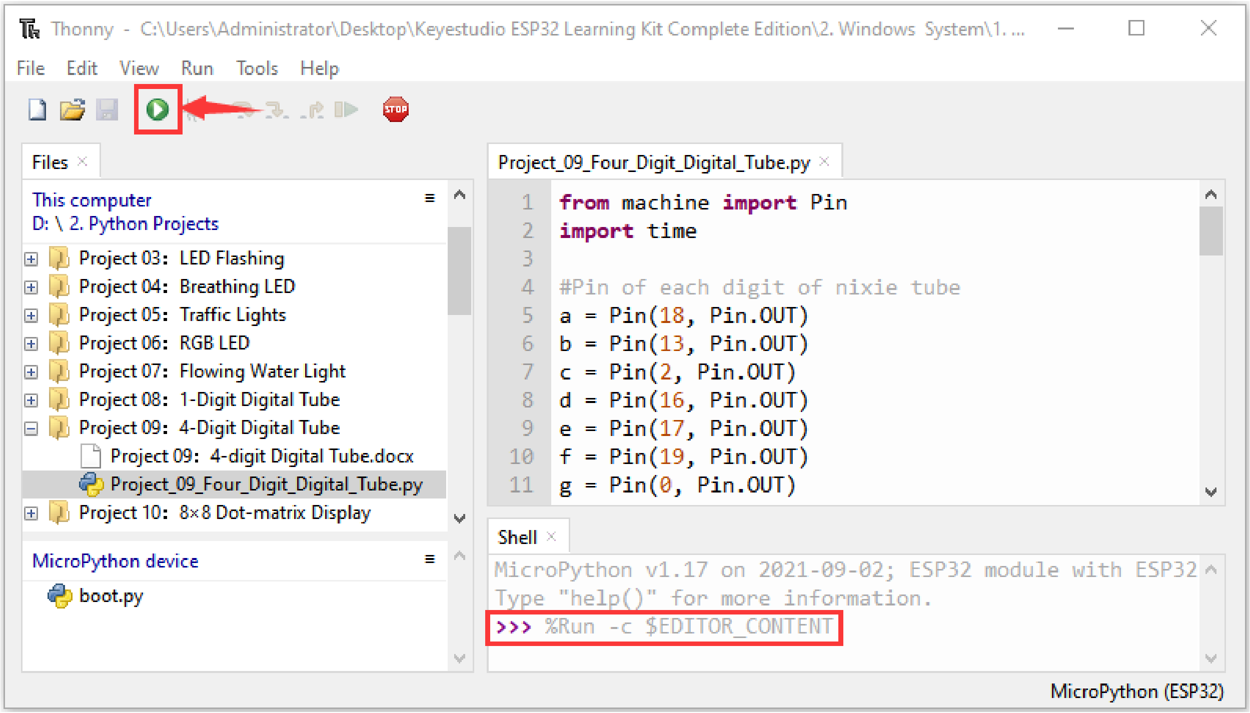
```
num='9016'
showData(num)
```

6.Test Result

Make sure the ESP32 has been connected to the computer, then 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 digitsand repeat these actions in an infinite loop. Press “Ctrl+C” or click “Stop/Restart backend” to exit the program.

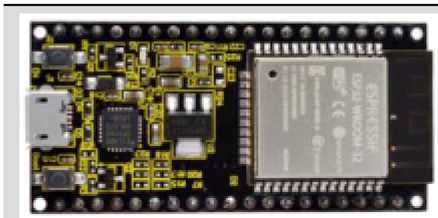
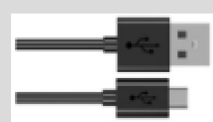
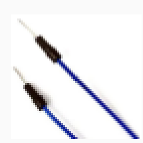


7.12 Project 108×8 Dot-matrix Display

1.Introduction

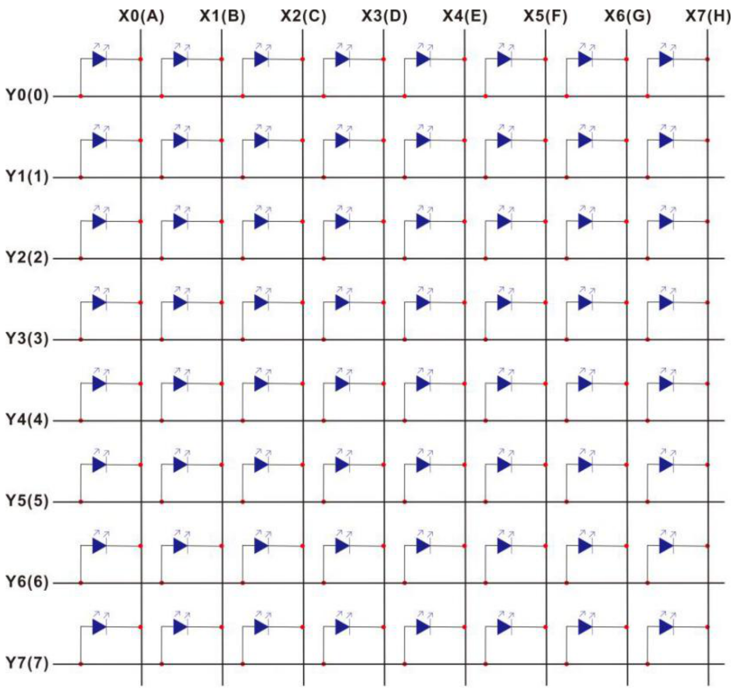
Dot matrix display is an electronic digital display device that can display information on machine, clocks, public transport departure indicators and many other devices. In this project, we will use ESP32 to control 8x8 LED dot matrix in a way that lights it up.

2.Components

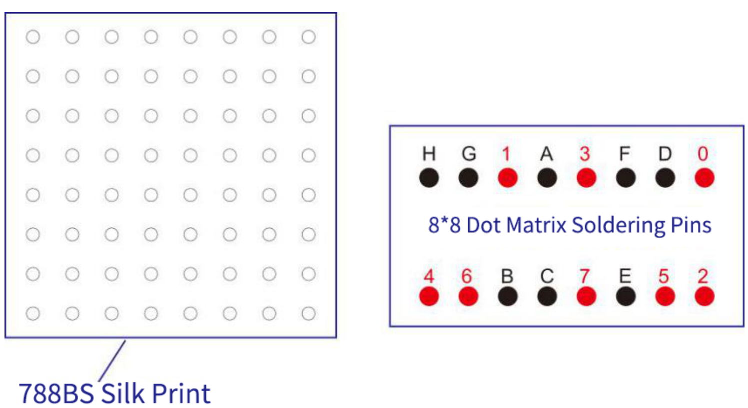
		
ESP32*1	Breadboard*1	USB Cable*1
		
8*8 dot matrix module *1	220 Resistor*8	Jumper Wire*2

3.Component Knowledge

8*8 dot matrix module The 8*8 dot matrix is composed of 64 LEDs, including row common anode and row common cathode. Our module is row common anode, each row has a line connecting the positive pole of the LED, and the column is connecting the negative pole of the LED lamp, as shown in the following figure :

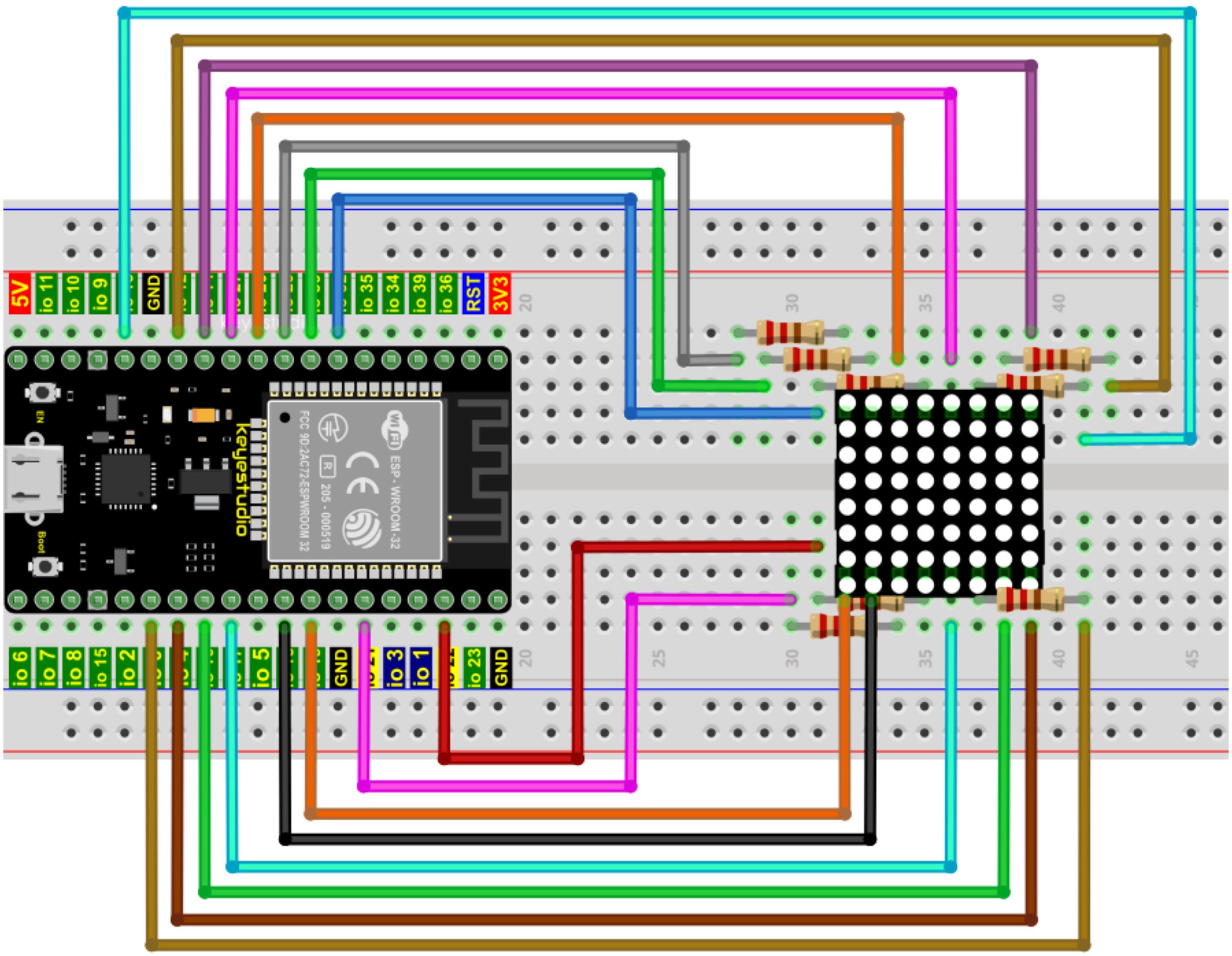


8*8 Dot Matrix LED Equivalent Circuit



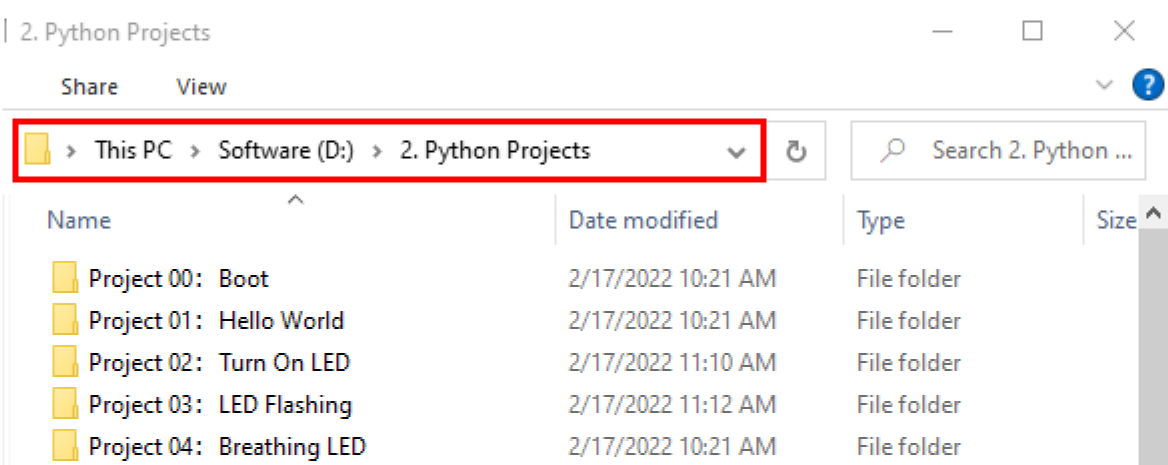
8*8 Dot Matrix Outlook and Pinouts

4.Wiring Diagram

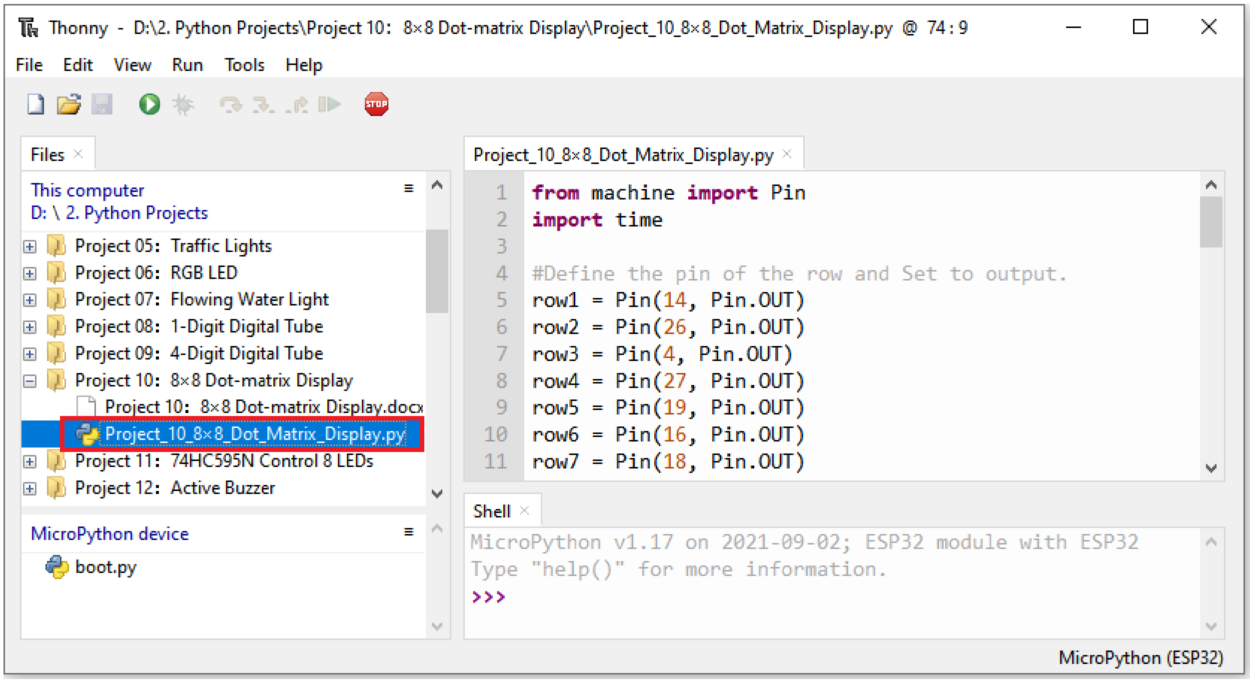


5.Test Code

The code used in this tutorial is saved in“2. Python Projects”. If you haven’t downloaded the code file yet, 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”, then double left-click“Project_10_8×8_Dot_Matrix_Display.py”.




```python
from machine import Pin
import time

#Define the pin of the row and Set to output.
row1 = Pin(14, Pin.OUT)
row2 = Pin(26, Pin.OUT)
row3 = Pin(4, Pin.OUT)
row4 = Pin(27, Pin.OUT)
row5 = Pin(19, Pin.OUT)
row6 = Pin(16, Pin.OUT)
row7 = Pin(18, Pin.OUT)
row8 = Pin(17, Pin.OUT)
#Define the pins of the column and Set to output
col1 = Pin(32, Pin.OUT)
col2 = Pin(21, Pin.OUT)
col3 = Pin(22, Pin.OUT)
col4 = Pin(12, Pin.OUT)
col5 = Pin(0, Pin.OUT)
col6 = Pin(13, Pin.OUT)
col7 = Pin(33, Pin.OUT)
col8 = Pin(25, Pin.OUT)

#Sets the pin of the column to low level
col1.value(0)
col2.value(0)
col3.value(0)
col4.value(0)
col5.value(0)
col6.value(0)
col7.value(0)
col8.value(0)
```

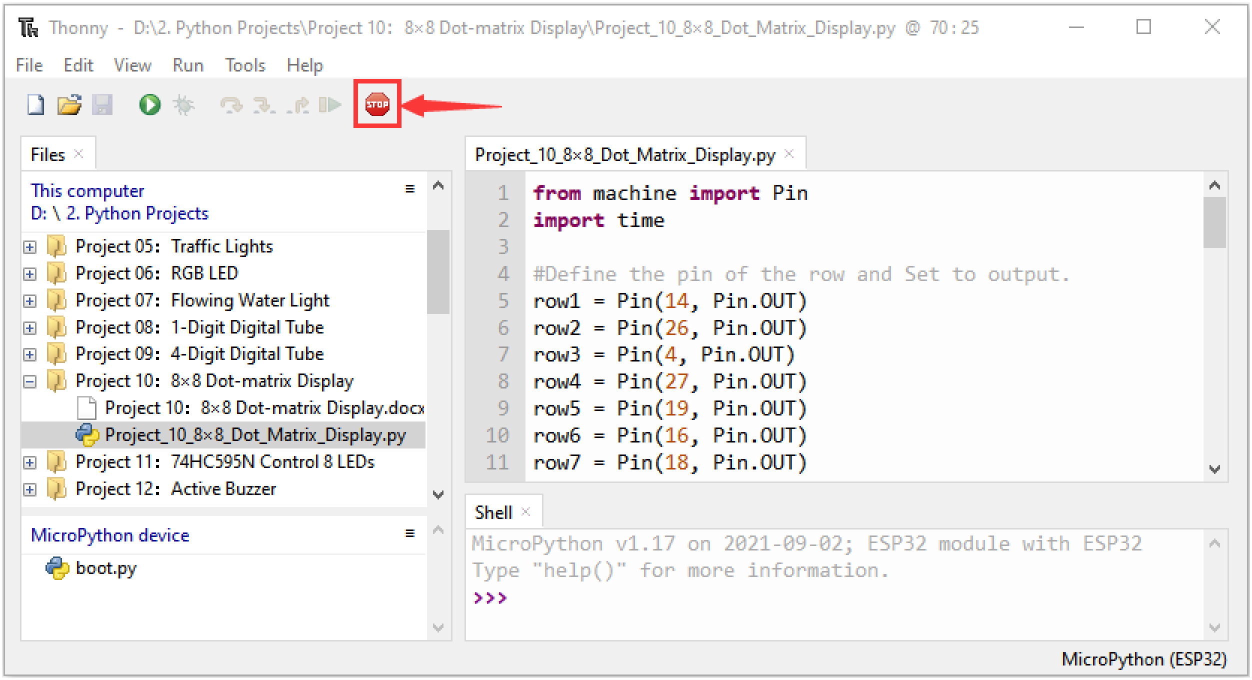
(continues on next page)

(continued from previous page)

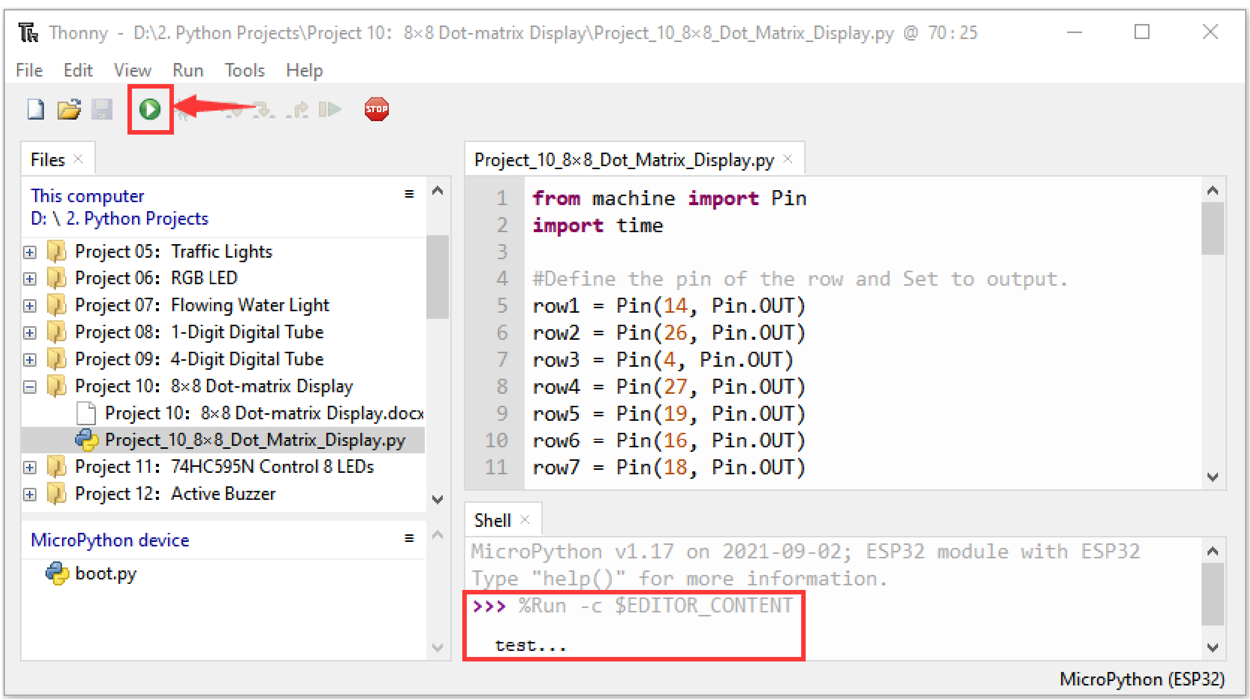
```
#Since the column of the lattice has been set to low level,
#the corresponding row of the lattice will light up when the pin of the row is at high level
def Row(d):
    if(d ==1):
        row1.value(1)  #Light the first line
    if(d ==2):
        row2.value(1)  #Light the second line
    if(d ==3):
        row3.value(1)
    if(d ==4):
        row4.value(1)
    if(d ==5):
        row5.value(1)
    if(d ==6):
        row6.value(1)
    if(d ==7):
        row7.value(1)
    if(d ==8):
        row8.value(1)

#Close the lattice
def off():
    row1.value(0)
    row2.value(0)
    row3.value(0)
    row4.value(0)
    row5.value(0)
    row6.value(0)
    row7.value(0)
    row8.value(0)

try:
    print("test...")
    while True:
         for num in range(1,10):  #Light the lattice line by line
             Row(num)
             if(num == 9):  #Because the lattice has only 8 rows, and I'm limiting it here, is equal to 9
                off()      #Close the lattice
             time.sleep(0.2)

except:
    pass
```

6.Test Result

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



Click “Run current script”, the code starts to be executed and you’ll see that the 8*8 dot matrix gradually lights up. Press“Ctrl+C” or click “Stop/Restart backend”to exit the program.



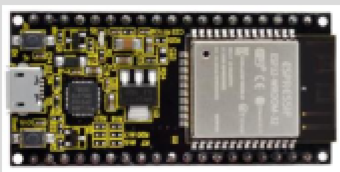
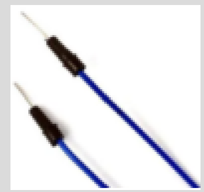
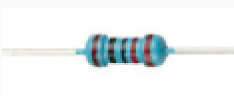
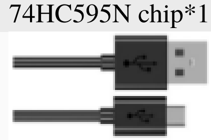
7.13 Project 1174HC595N Control 8 LEDs

1.Introduction

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

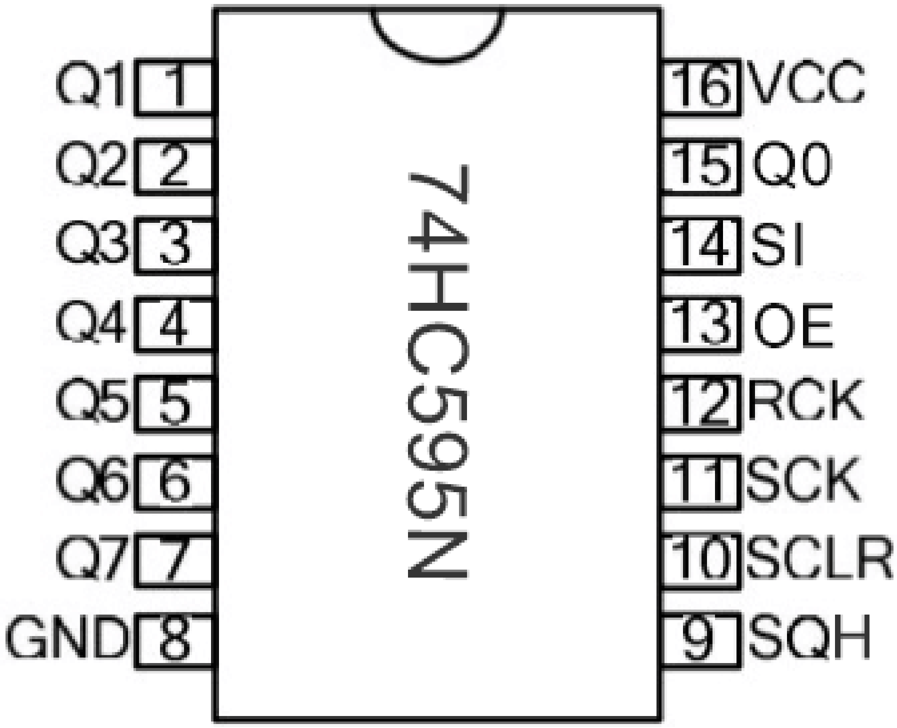
With only 32 IO ports on ESP32, how do we light up a lot of leds? Sometimes it is possible to run out of pins on the ESP32, and you need to extend it with the shift register.You can use the 74HC595N chip to control 8 outputs at a time, taking up only a few pins on your microcontroller. In addition, you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

2.Components

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

3.Component knowledge

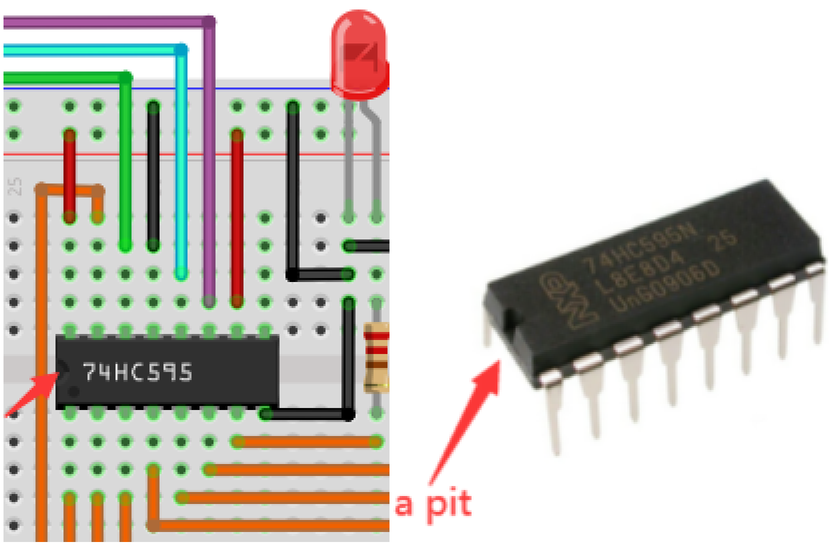
74HC595N Chip: The 74HC595 chip is used to convert serial data into parallel data. A 74HC595 chip can convert the serial data of one byte into 8 bits, and send its corresponding level to each of the 8 ports correspondingly. With this characteristic, the 74HC595 chip can be used to expand the IO ports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

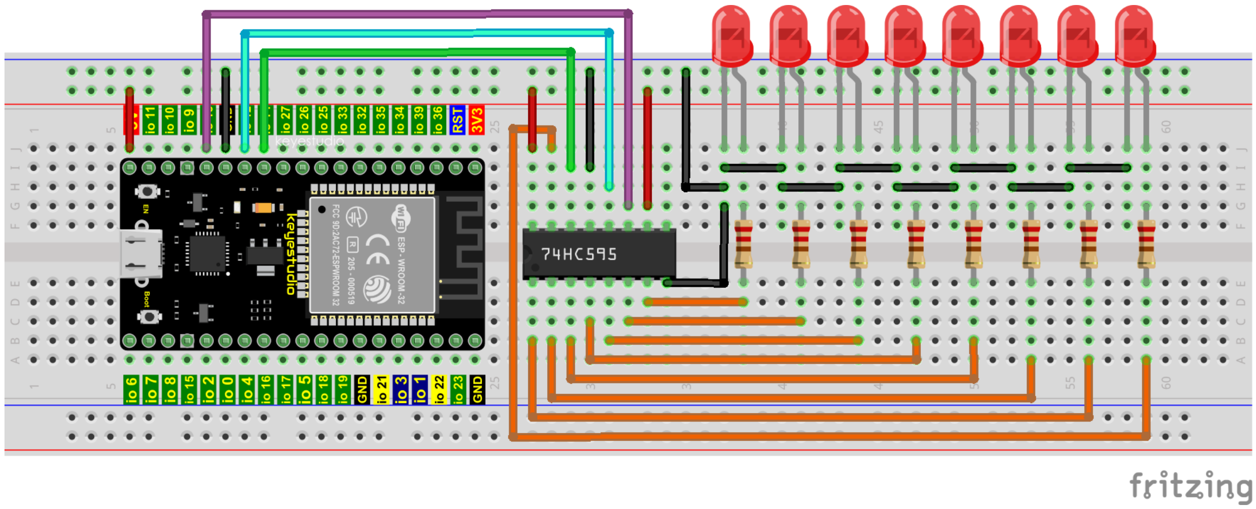


The ports of the 74HC595 chip are described as follows

4.Wiring diagram

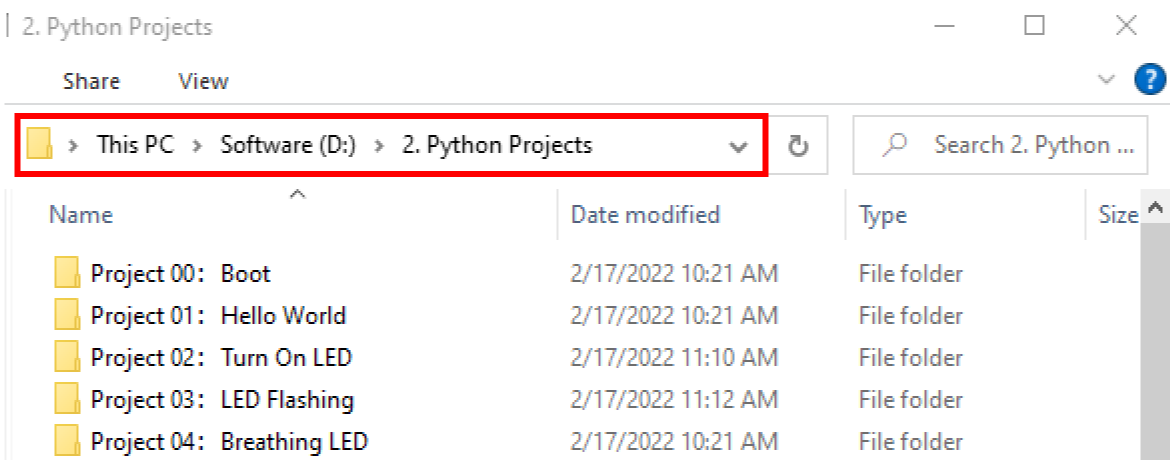
Note: Note the orientation in which the 74HC595N chip is inserted.





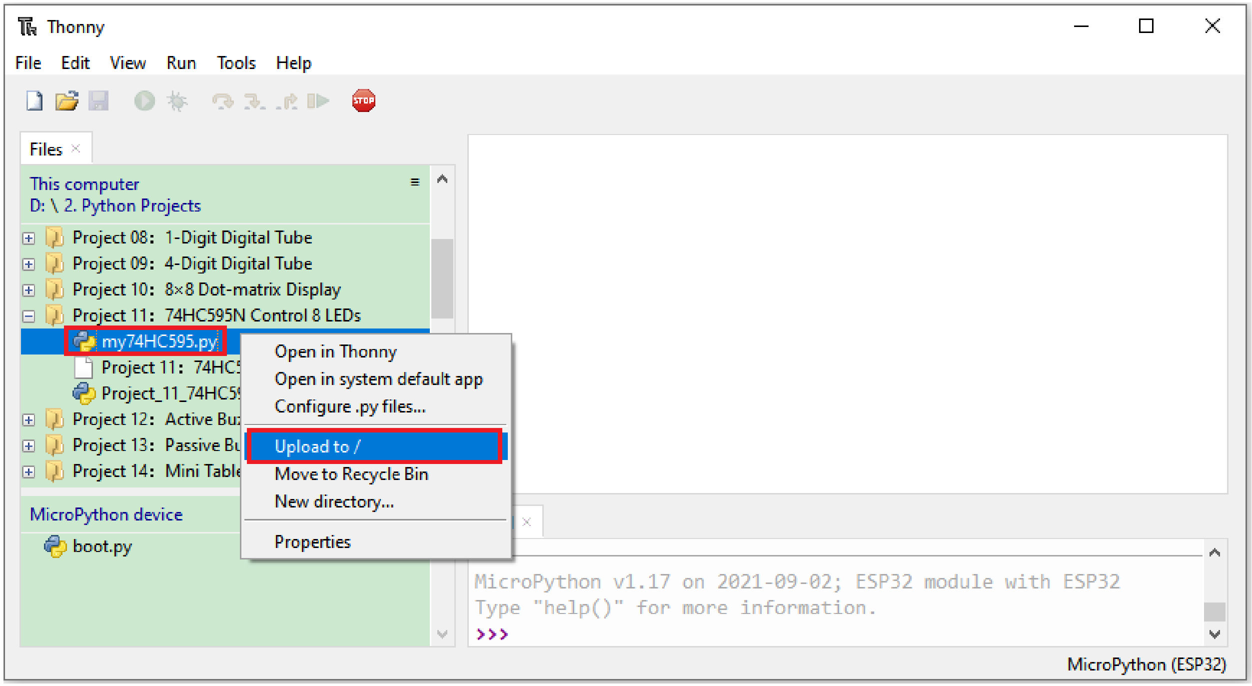
5.Project code

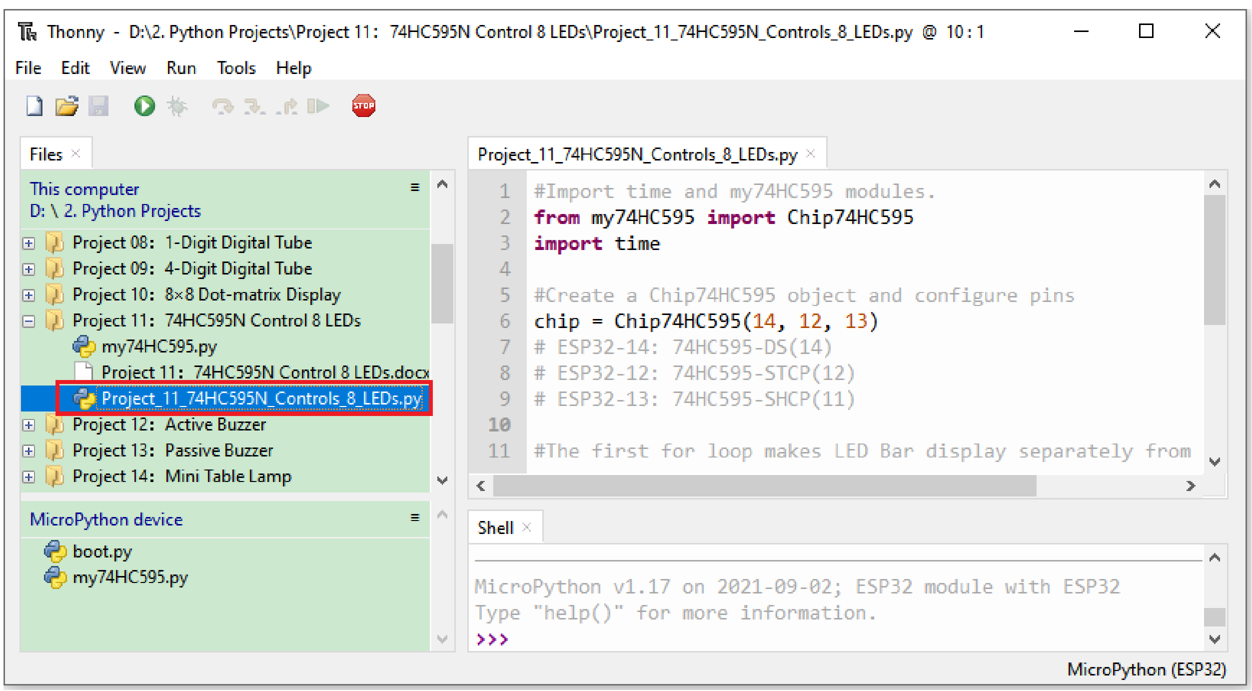
Codes used in this tutorial are saved in“2. Python Projects”. If you haven’t downloaded the code file yet, please click on the link to download it:Download Python Codes



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”.





```
#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)
```

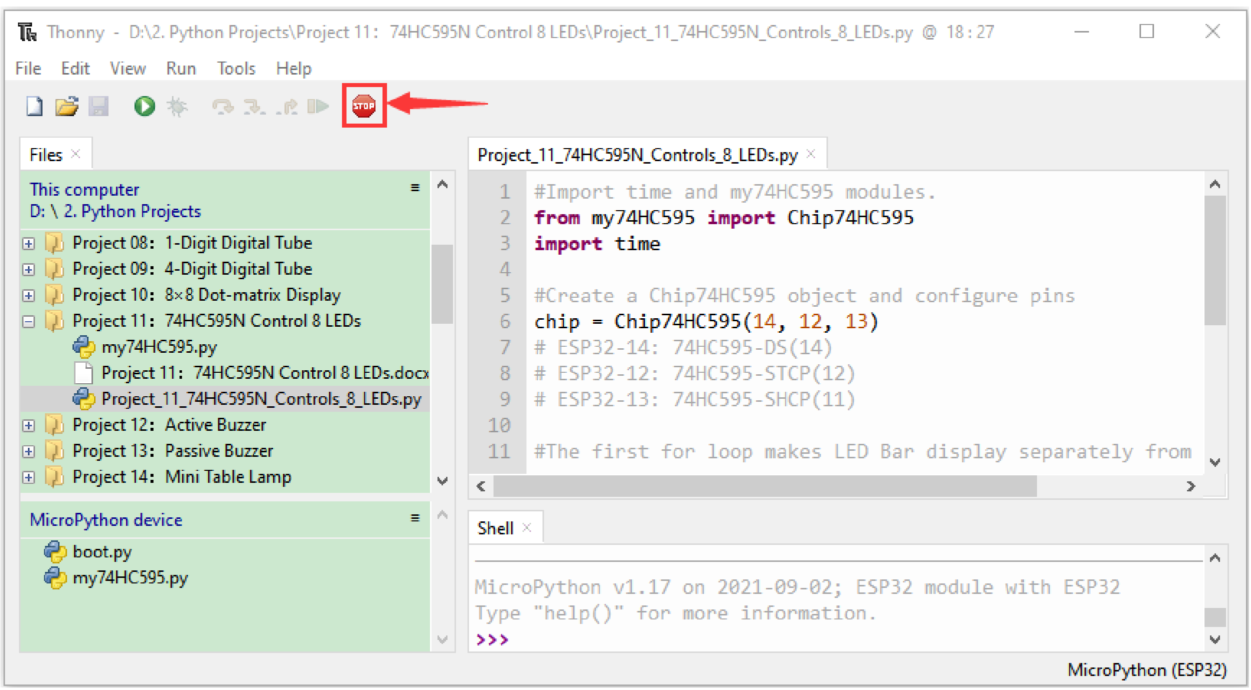
(continues on next page)

(continued from previous page)

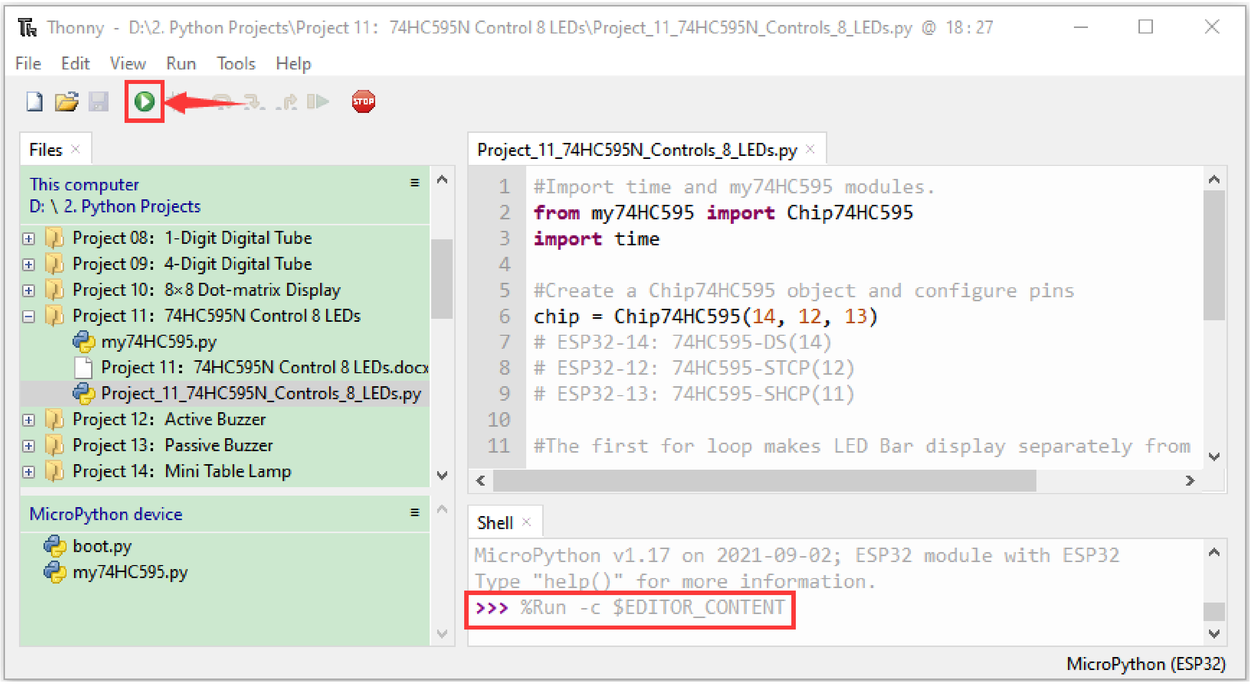
```
#The first for loop makes LED Bar display separately from left to right
#while the second for loop make it display separately from right to left.
while True:
    x = 0x01
    for count in range(8):
        chip.shiftOut(1, x)
        x = x<<1;
        time.sleep_ms(300)
    x = 0x01
    for count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

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.

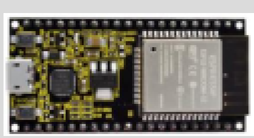
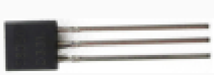
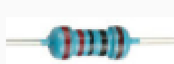
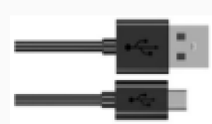


7.14 Project 12Active Buzzer

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.

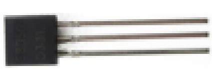
2.Components

			
ESP32*1	Breadboard*1	Active buzzer*1	
			
NPN Transistor(S8050)*1	1k Resistor*1	Jumper Wires	USB Cable*1

3.Component knowledge

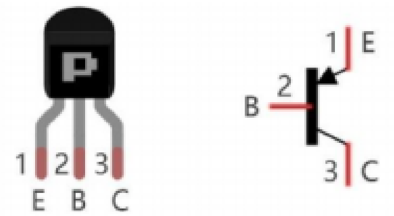
Active buzzer: Active buzzer inside has a simple oscillator circuit, which can convert constant direct current into a certain frequency pulse signal. Once active buzzer receives a high level, it will produce sound. Passive buzzer is an internal without vibration source integrated electronic buzzer, it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

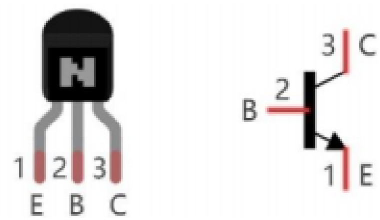
Transistor:



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.



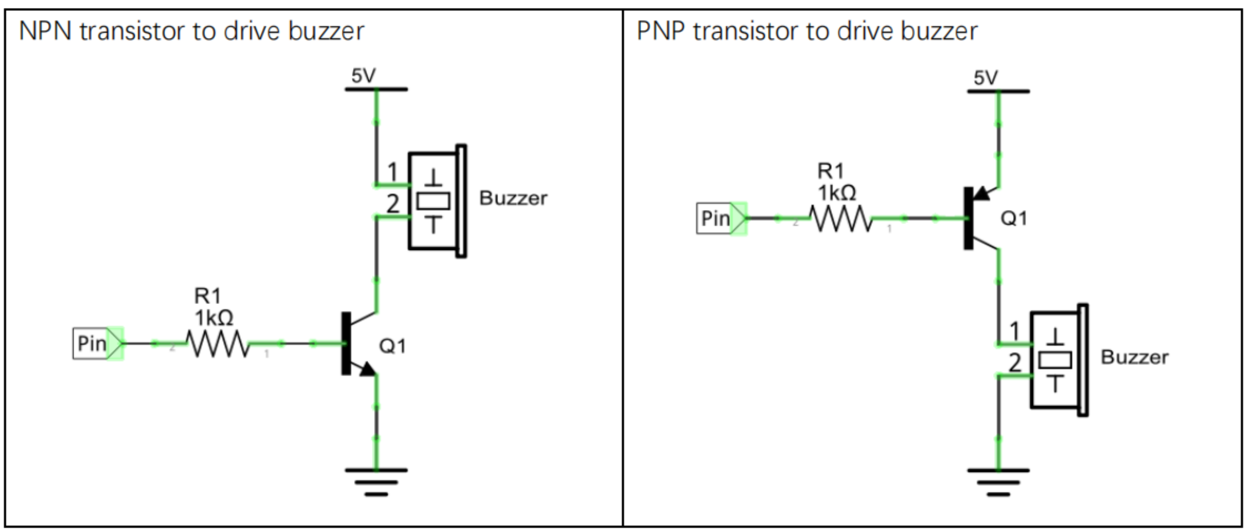


In our kit, the PNP transistor is marked with 8550, and the NPN transistor is marked with 8050.

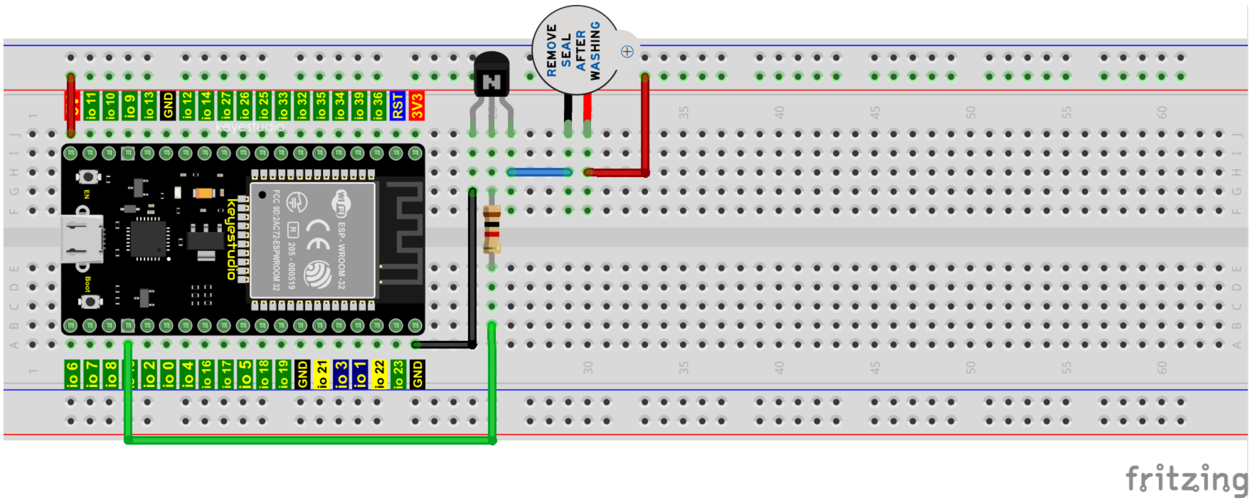
Based on the transistor's characteristics, it is often used as a switch in digital circuits. As micro-controller's capacity to output current is very weak, we will use transistor to amplify current and drive large-current components.

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.



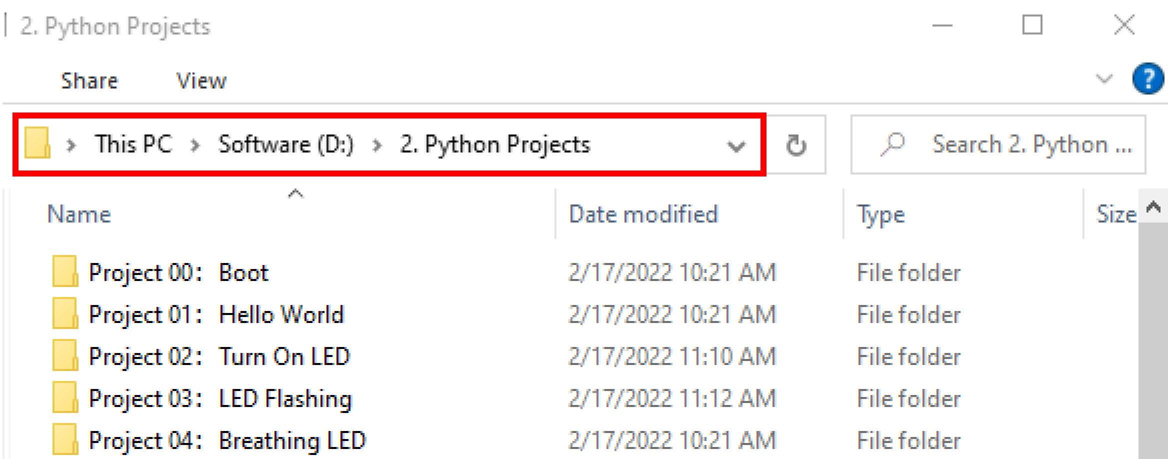
4.Wiring diagram



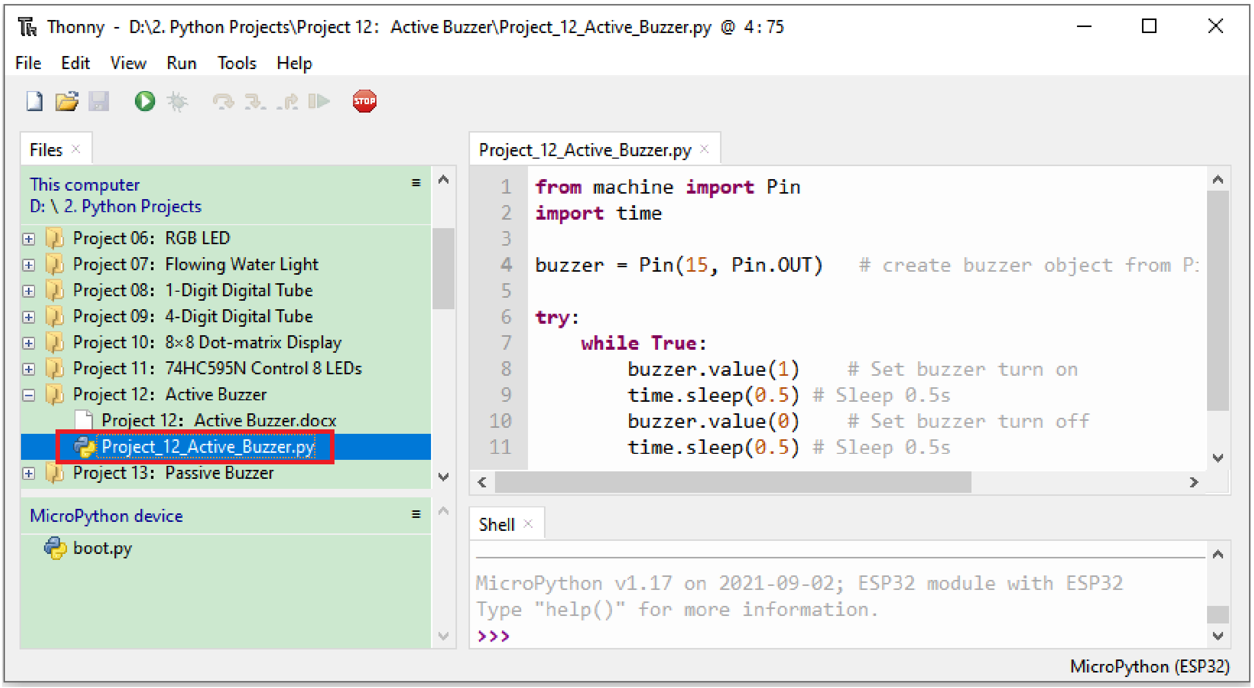
Note: The buzzer power supply in this circuit is 5V. On a 3.3V power supply, the buzzer can work, but will reduce the loudness.

5.Project code

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



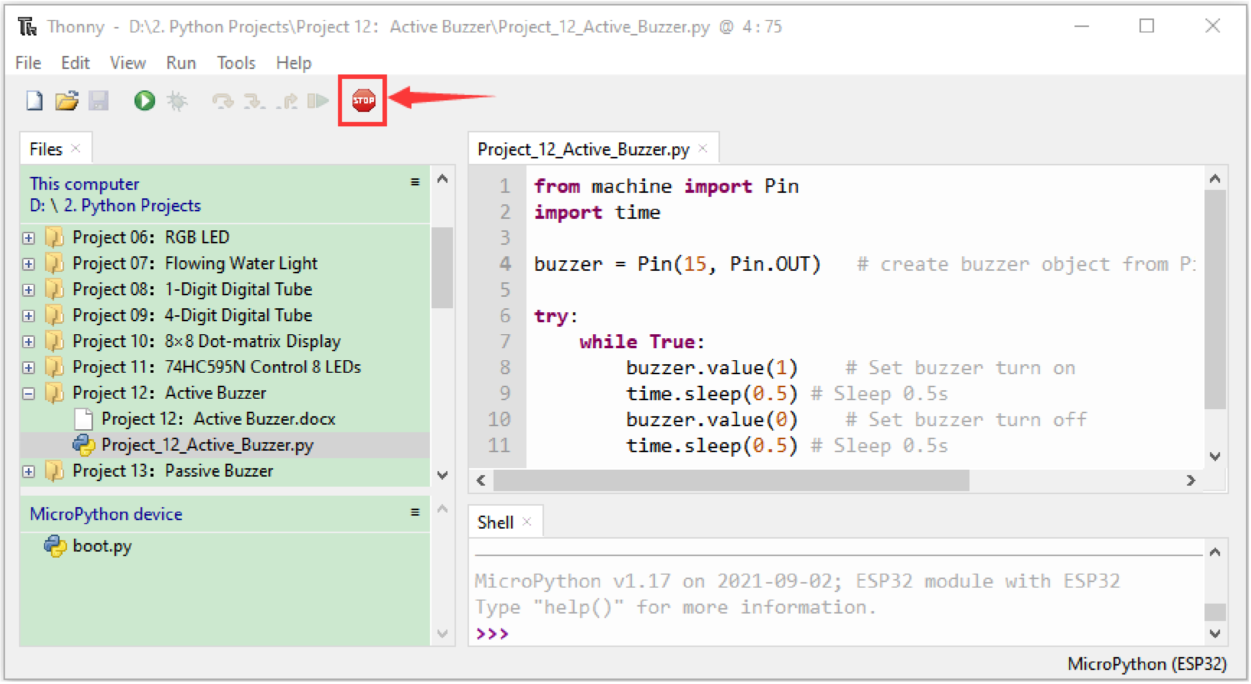
Open“Thonny”click“This computer”→“D:”→“2. Python Projects”→”Project 12: Active Buzzer”, and then double left-click “Project_12_Active_Buzzer.py”.



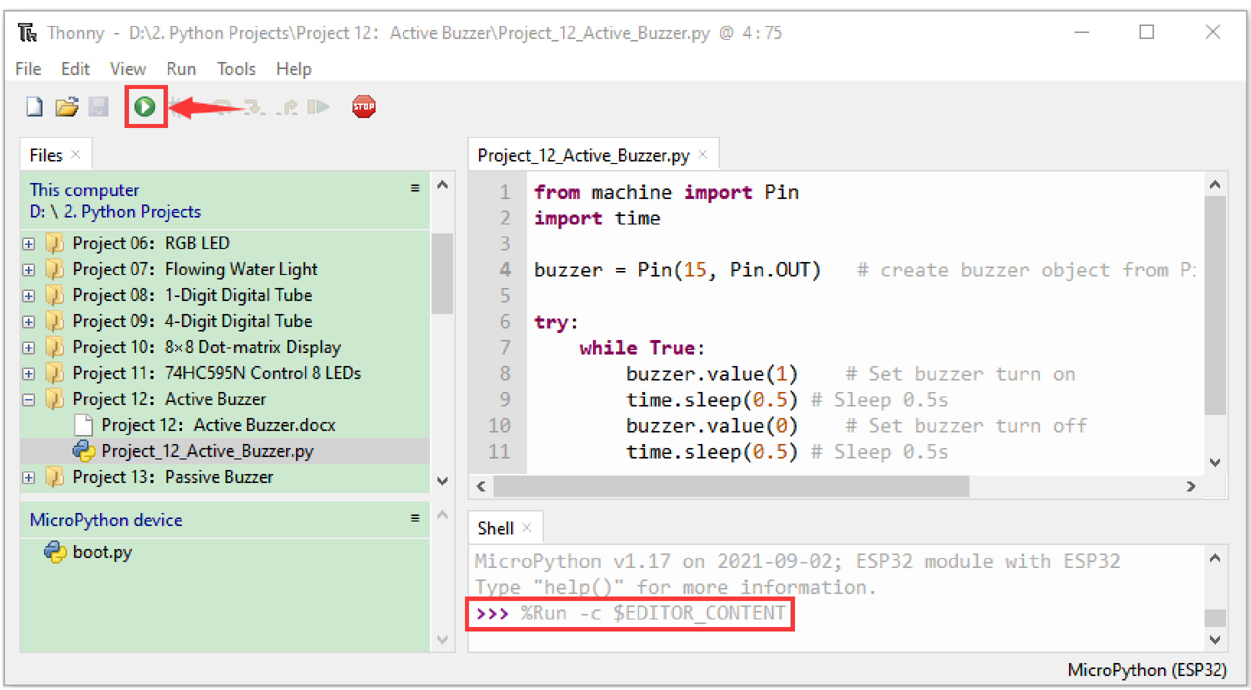

```python
from machine import Pin
import time

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
```

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.

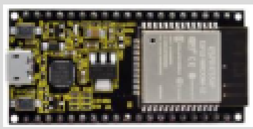
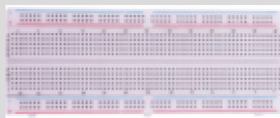
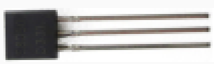
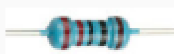
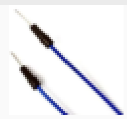
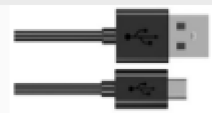


7.15 Project 13Passive Buzzer

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

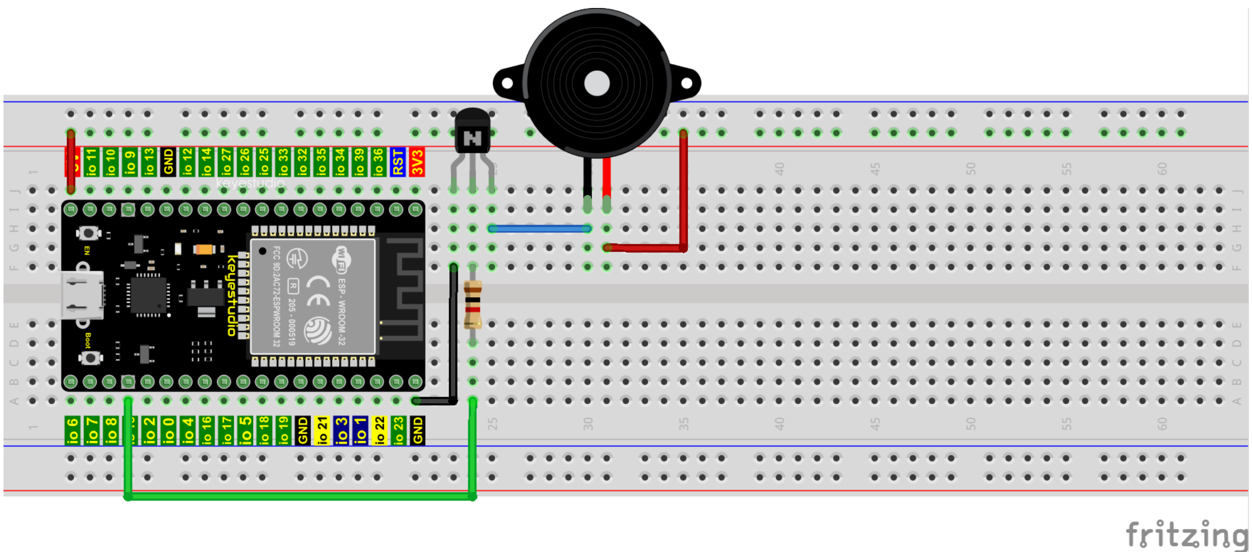
			
ESP32*1	Breadboard*1	Passive Buzzer *1	
			
NPN Transistor(S8050)*1	1kResistor*1	Jumper Wires	USB Cable*1

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.

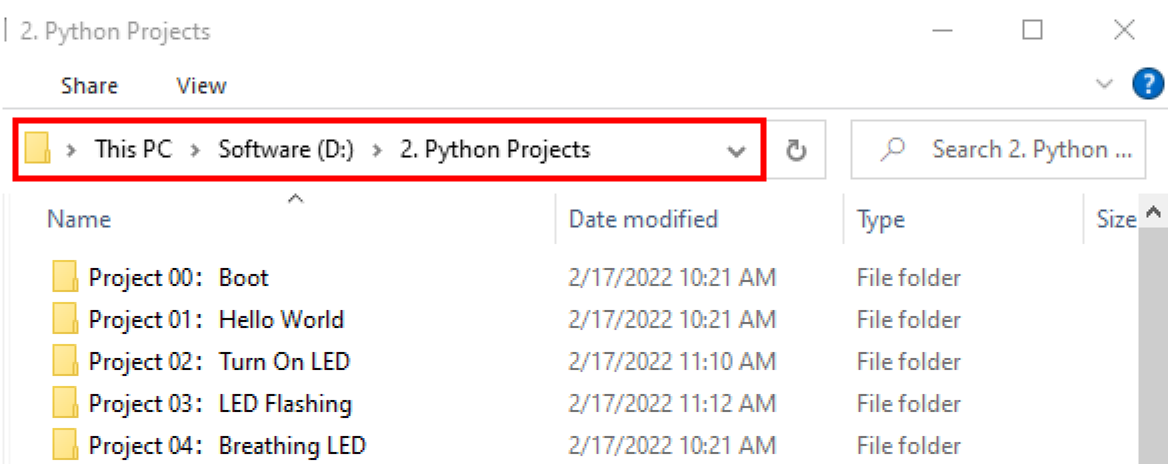
Transistor: Please refer to Project 12.

4.Wiring diagram:

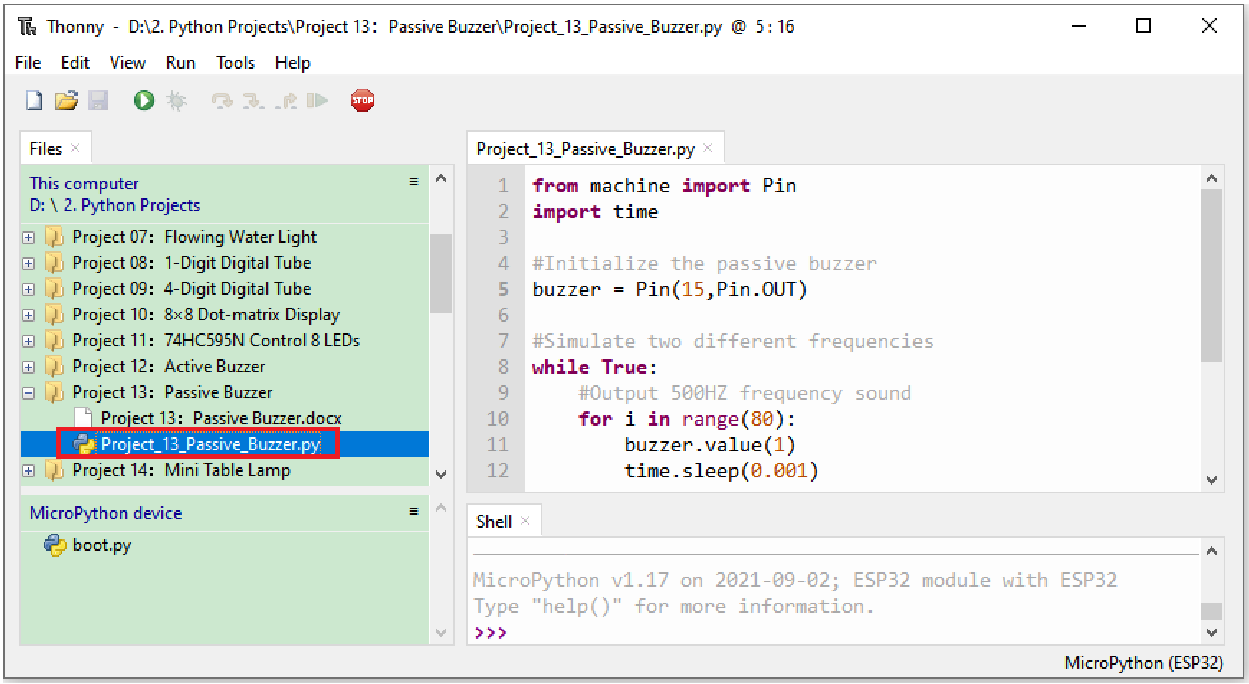


5.Project code

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



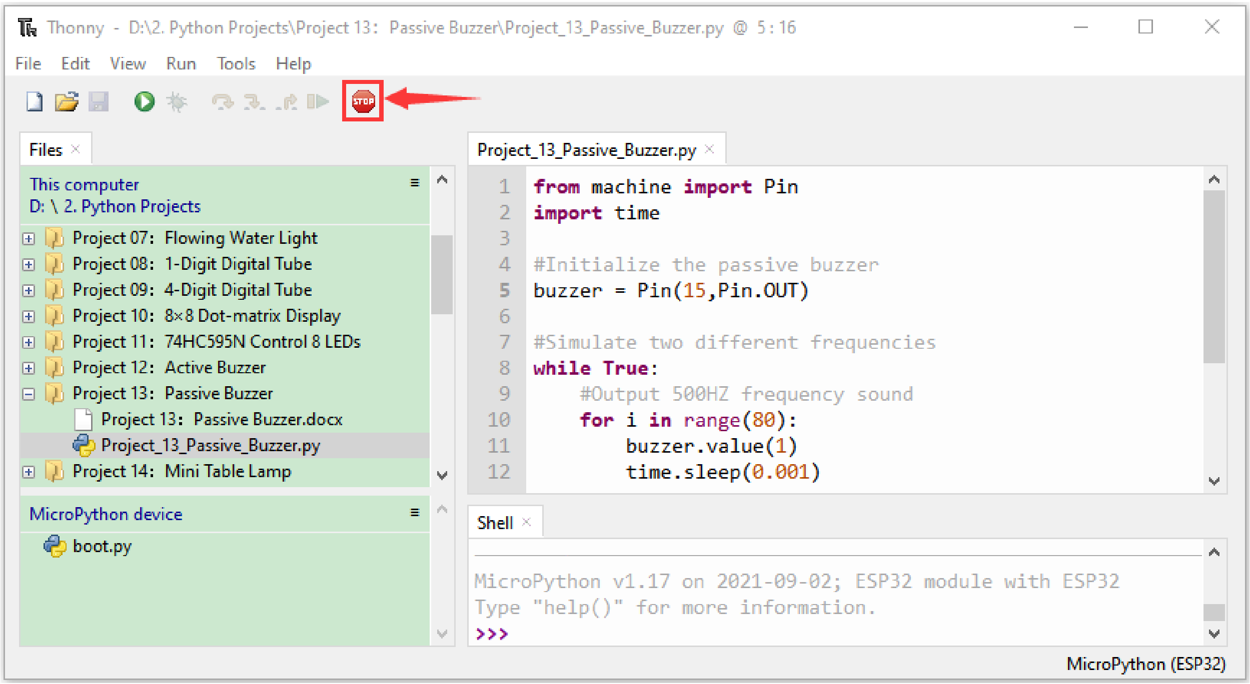
Open“Thonny”click“This computer”→“D:”→“2. Python Projects”→”Project 13: Passive Buzzer”, and then double left-click “Project_13_Passive_Buzzer.py”.



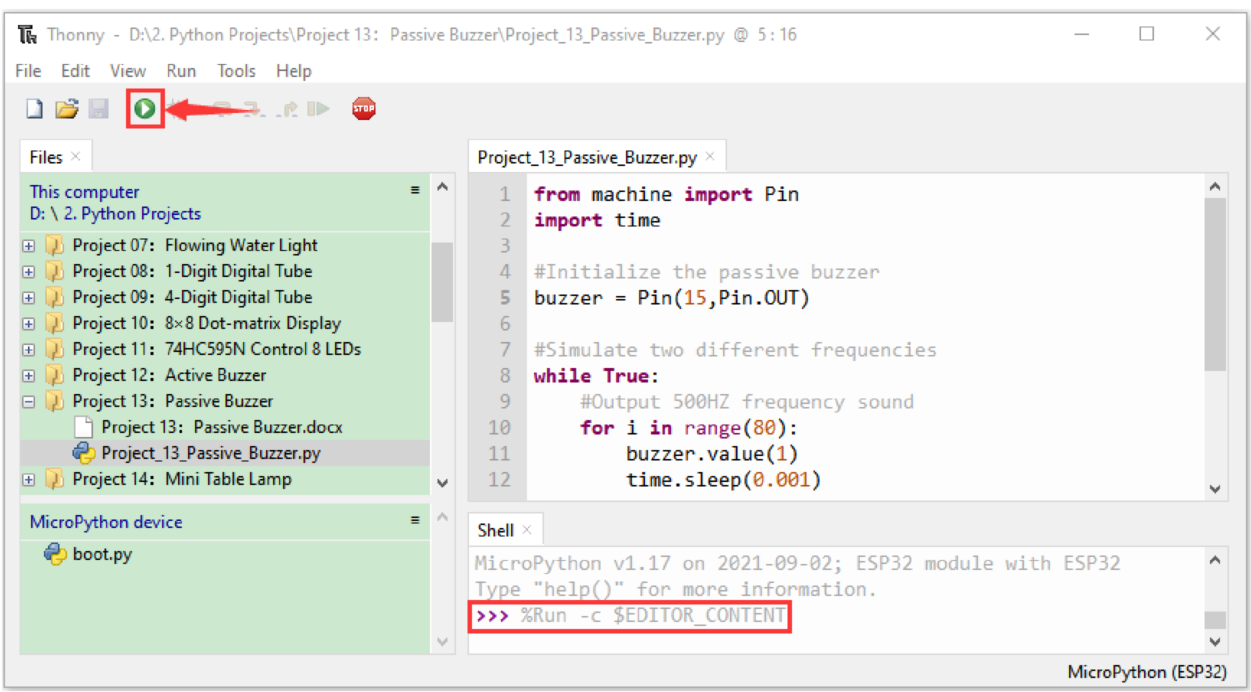

```python
from machine import Pin
import time

#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)
        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)
```

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.

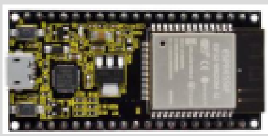
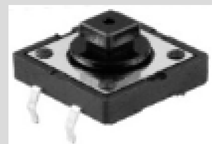
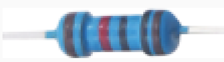
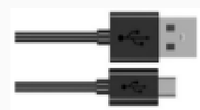
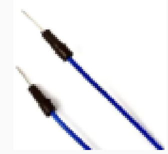


7.16 Project 14: Mini Table Lamp

1.Introduction

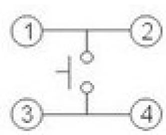
Do you know that the ESP32 can light up an LED when you press a button? In this project, we will use ESP32a button switch and an LED to make a mini table lamp.

2.Components

				
ESP32*1	Breadboard*1	Button*1	Button Cap*1	
				
10K Resistor*1	Red LED*1	22 Resistor*1	USB Cable*1	Jumper Wires

3.Component knowledge

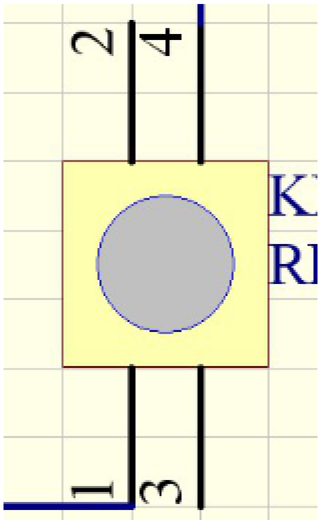
Button: A button can control the circuit on and off, the button is plugged into a circuit, the circuit is disconnected when the button is not pressed. The circuit works when you press the button, but breaks again when you release it. Why does it only work when you press it? It starts from the internal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

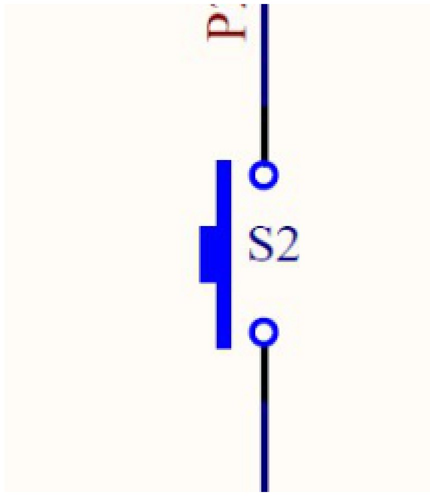
The internal structure of the button is shown in the figure 

. 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:

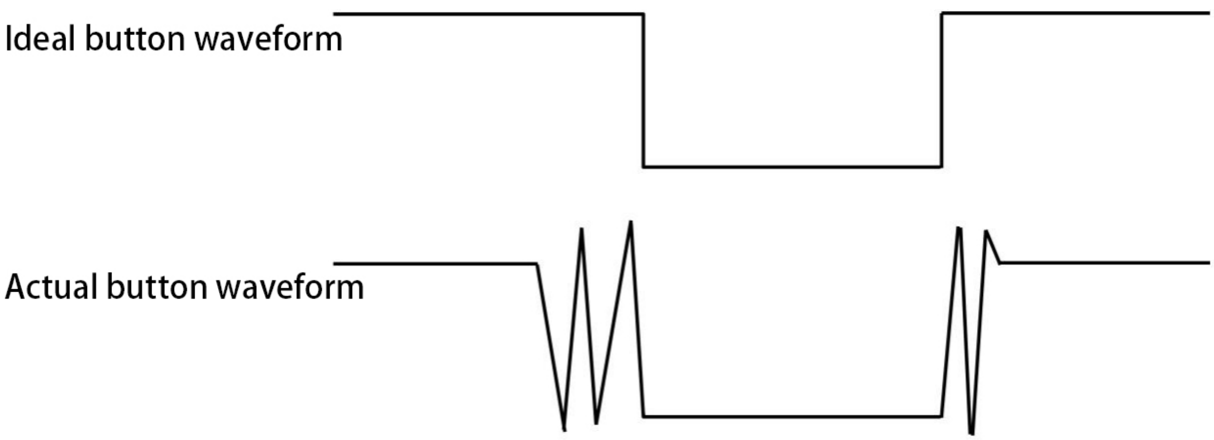




What is button shake javascript?

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.

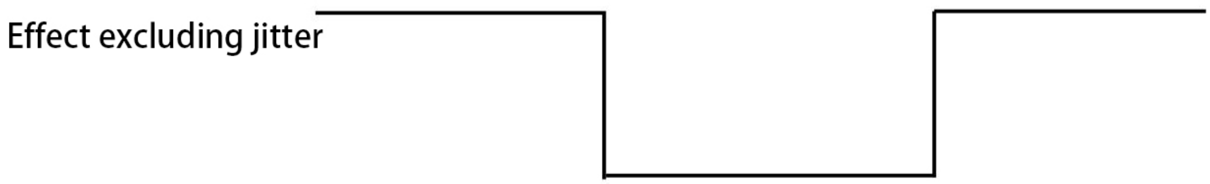


How to eliminate the [shake](javascript:;)?

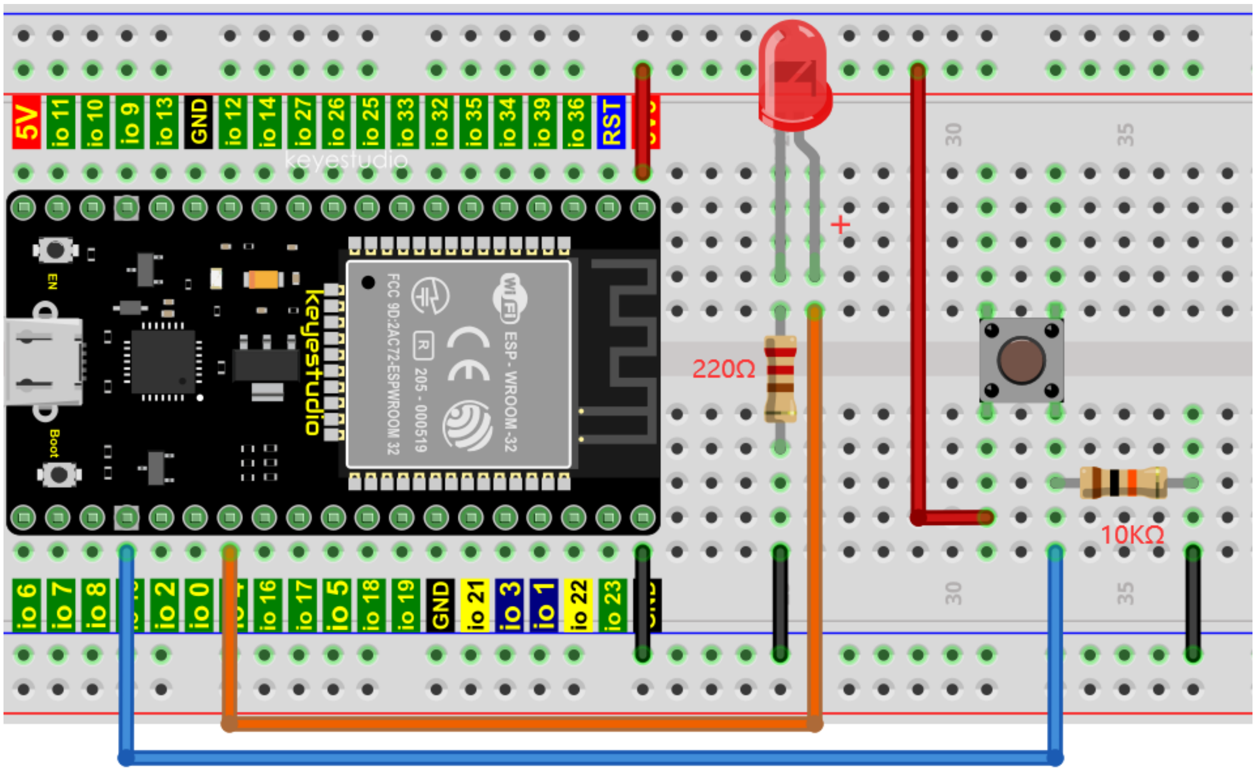
There are two common methods, namely fix [shake](javascript:;) in the software and hardware. We only discuss the [shake](javascript:;) removal in the software.

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.

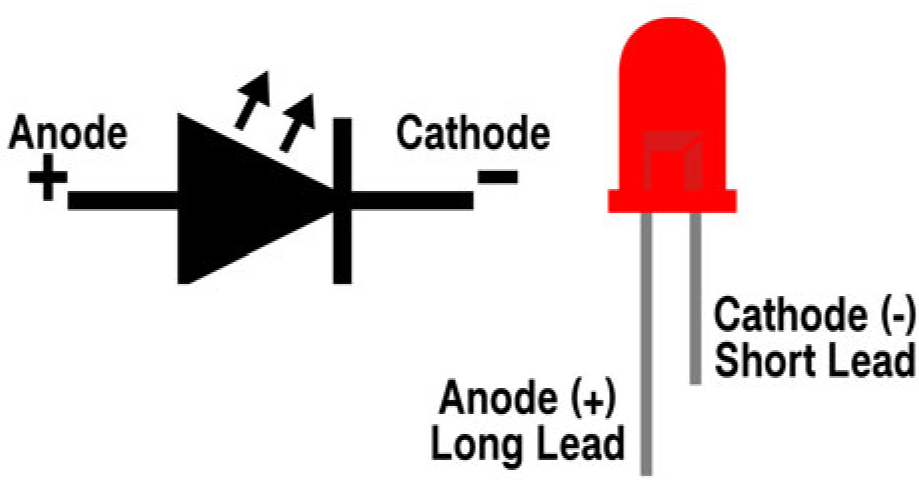


4.Wiring Diagram

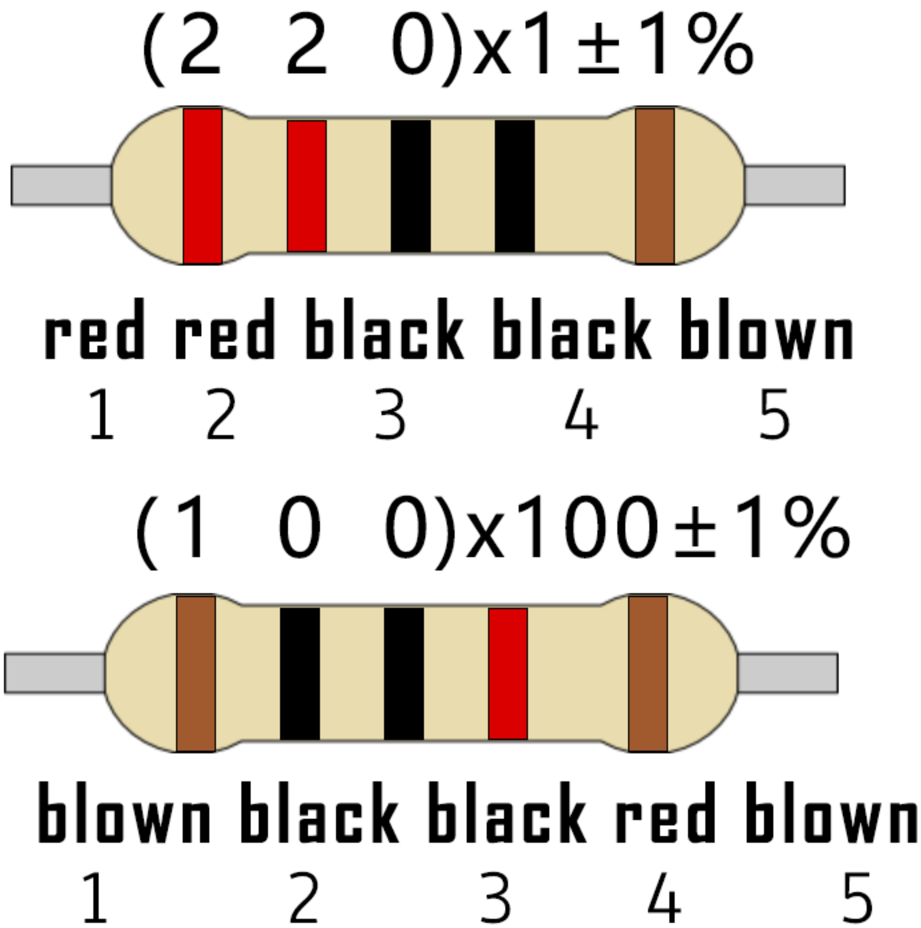


Note:

How to connect the LED

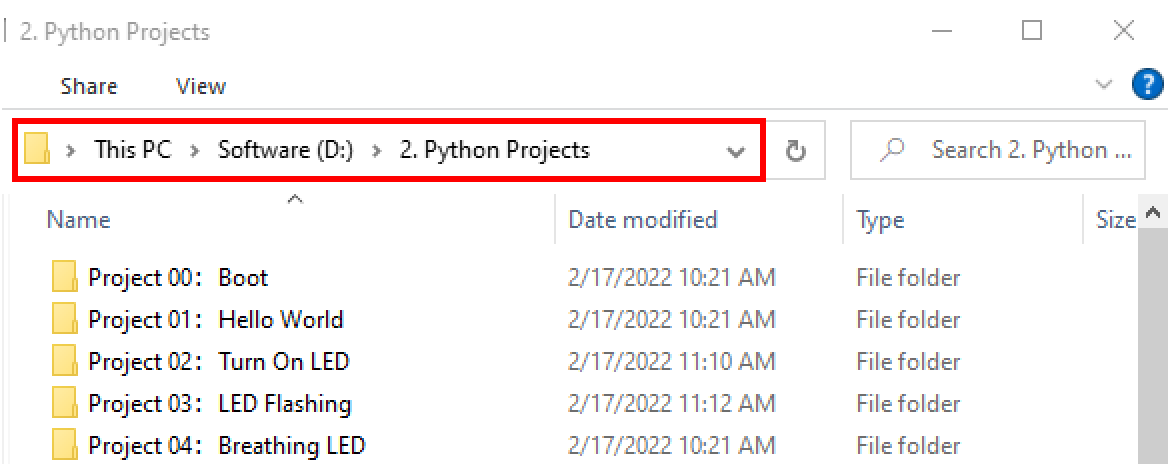


How to identify the 220 5-band resistor and 10K 5-band resistor

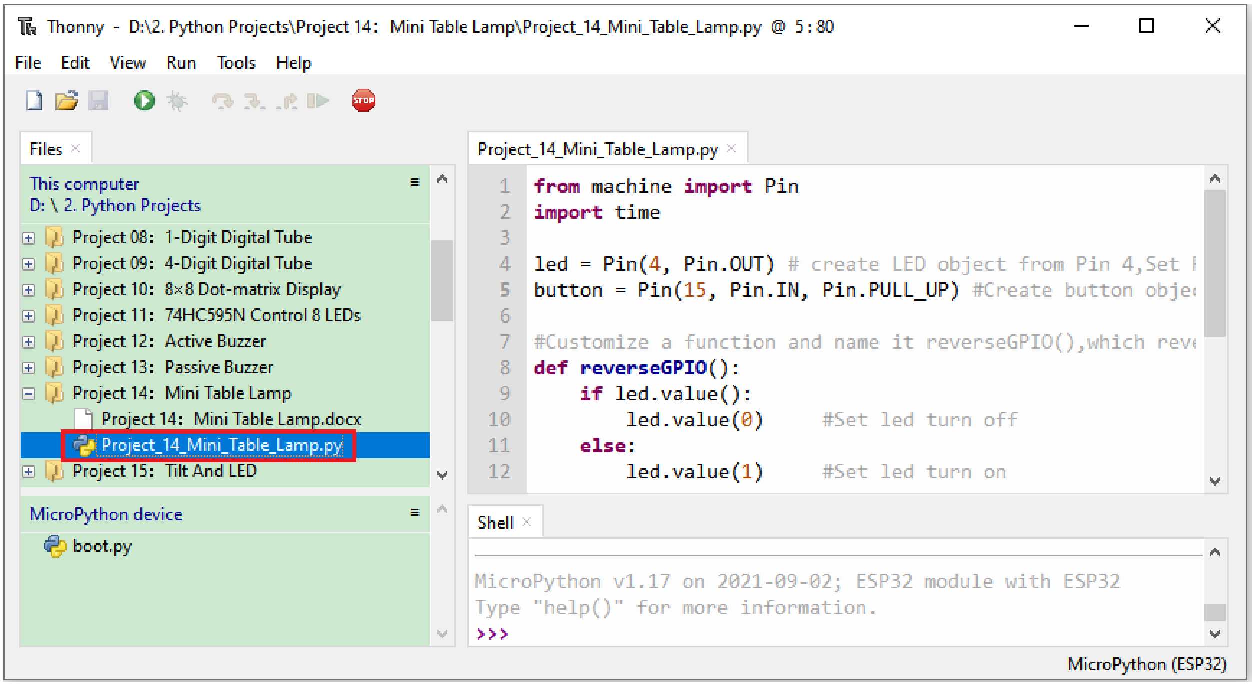


5.Project code

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



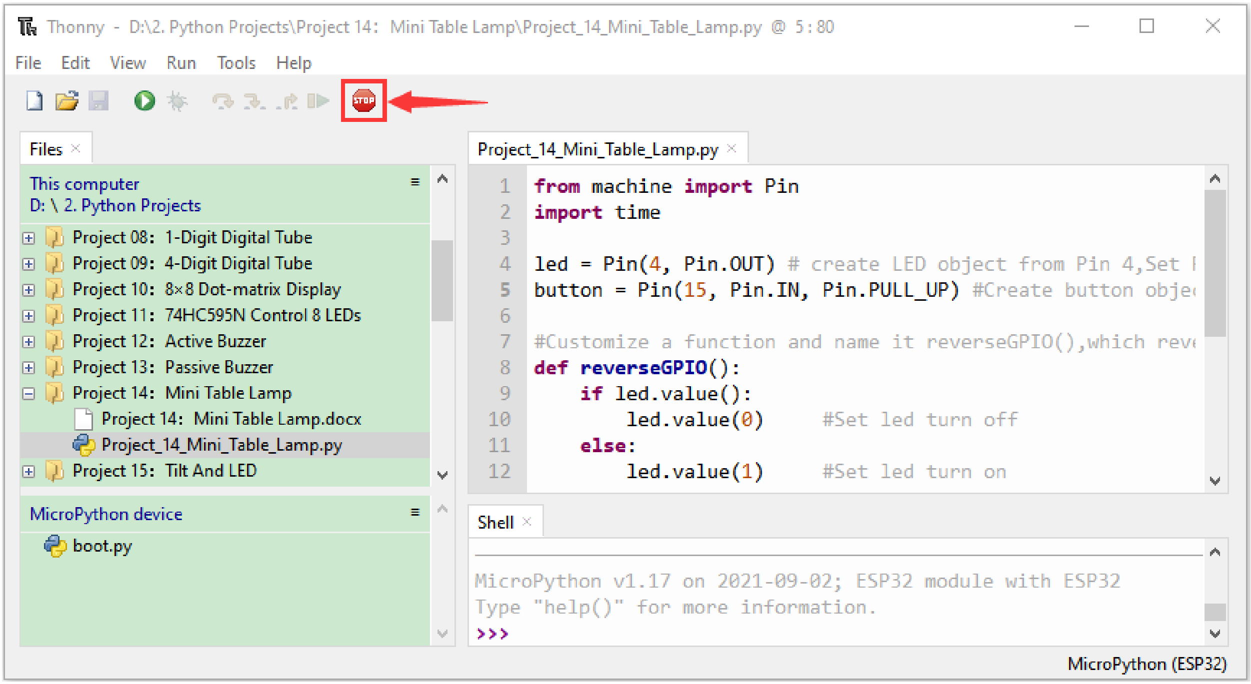
Open“Thonny”click“This computer”→“D:”→“2. Python Projects”→”Project 14: Mini Table Lamp”, and then double left-click “Project_14_Mini_Table_Lamp.py”.



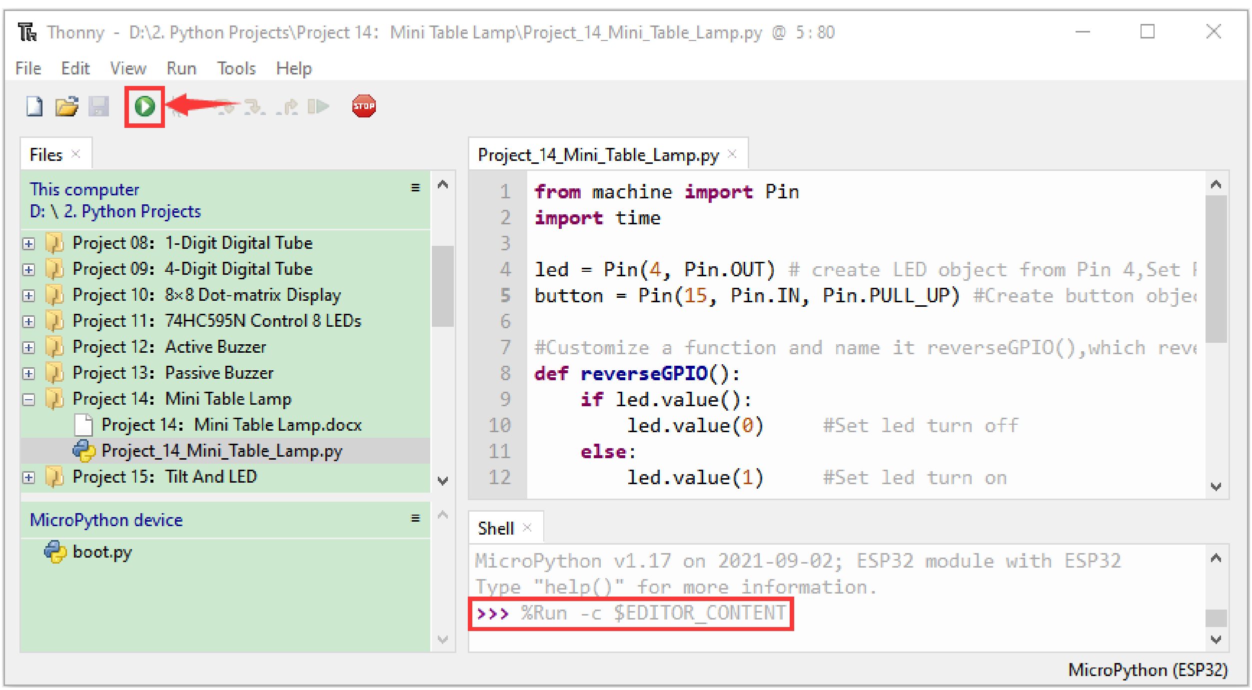

```python
from machine import Pin
import time

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
```

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.



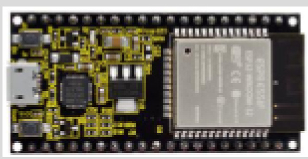
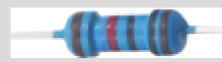
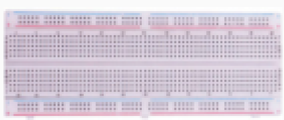
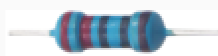
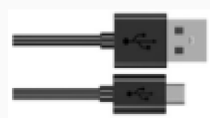
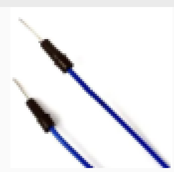
7.17 Project 15Tilt And LED

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

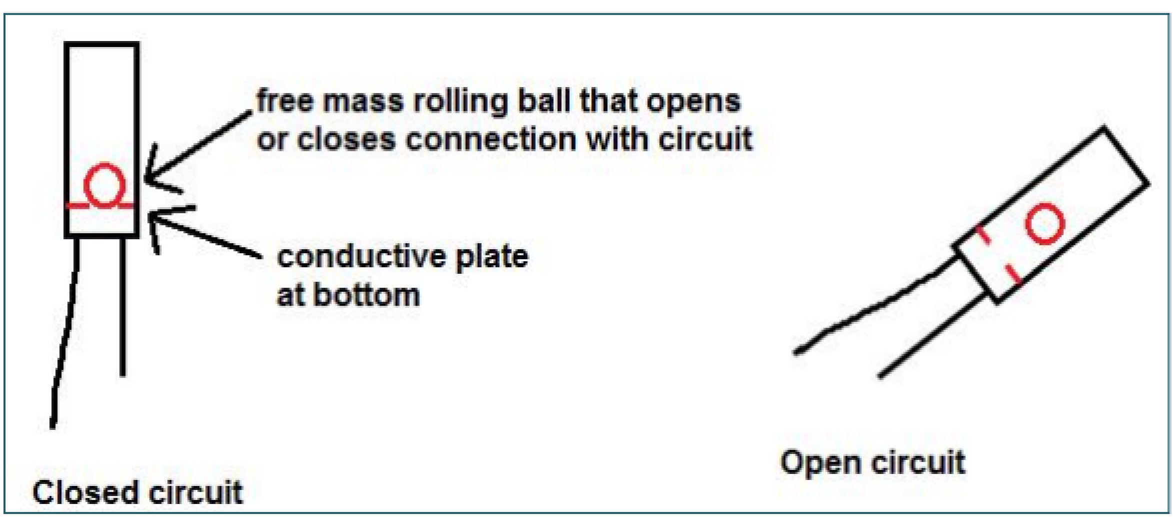
			
ESP32*1	Tilt Switch*1	Red LED*4	10K Resistor*1
			
Breadboard*1	220 Resistor*4	USB Cable*1	Jumper Wires

3.Component knowledge

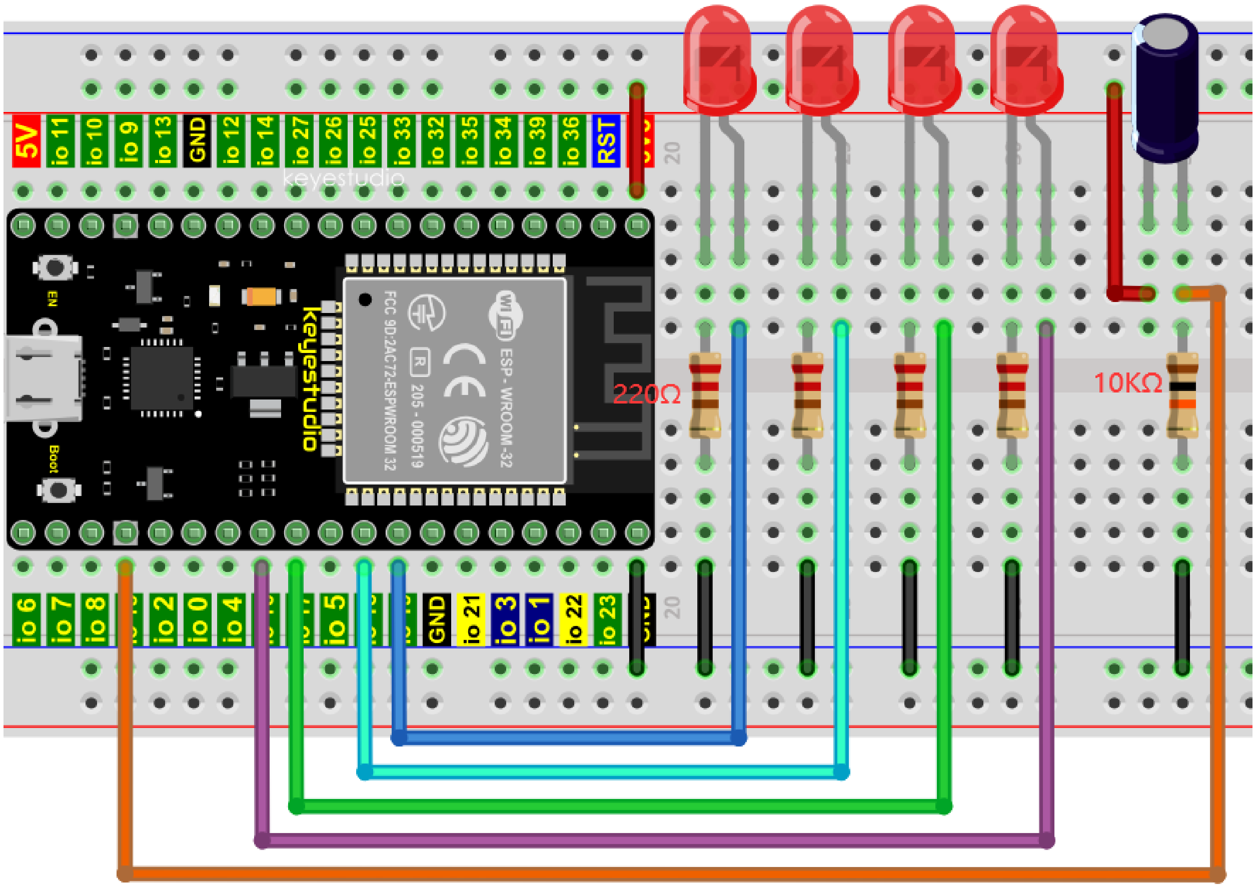
Tilt switch is also called digital switch. Inside is a metal ball that can roll. The principle of rolling the metal ball to contact with the conductive plate at the bottom, which is used to control the on and off of the circuit. When it is a rolling ball tilt sensing switch with single directional trigger,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

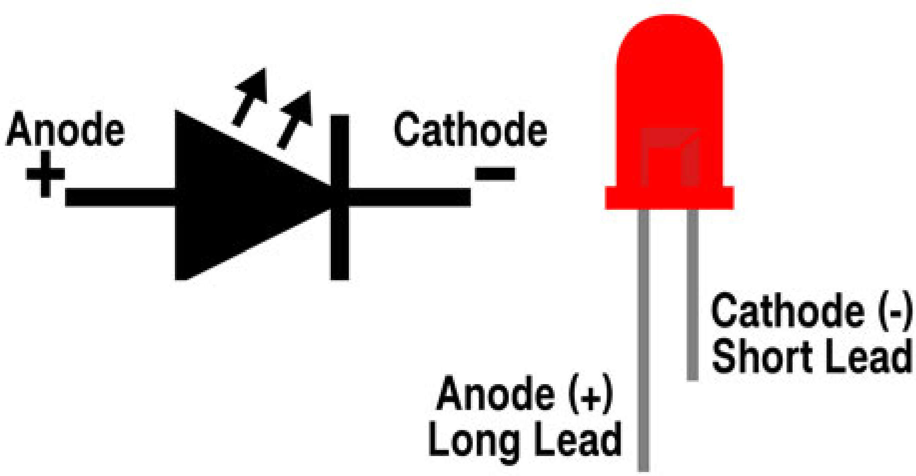


4.Wiring Diagram

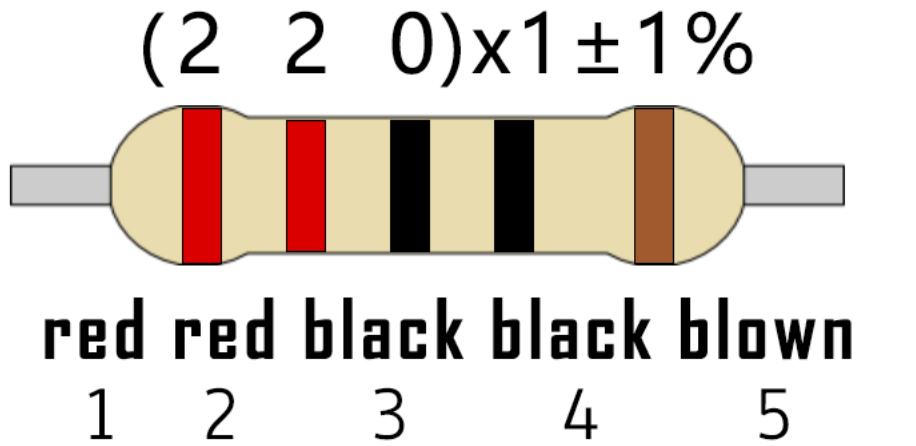


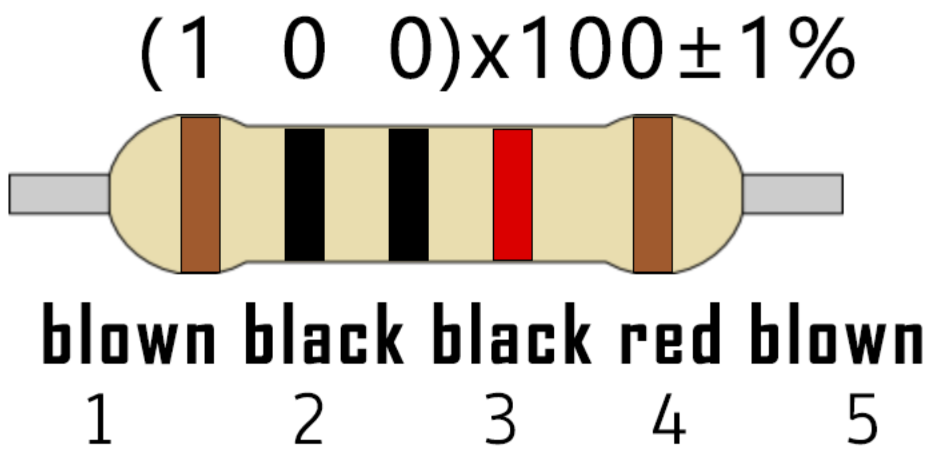
Note:

How to connect the LED



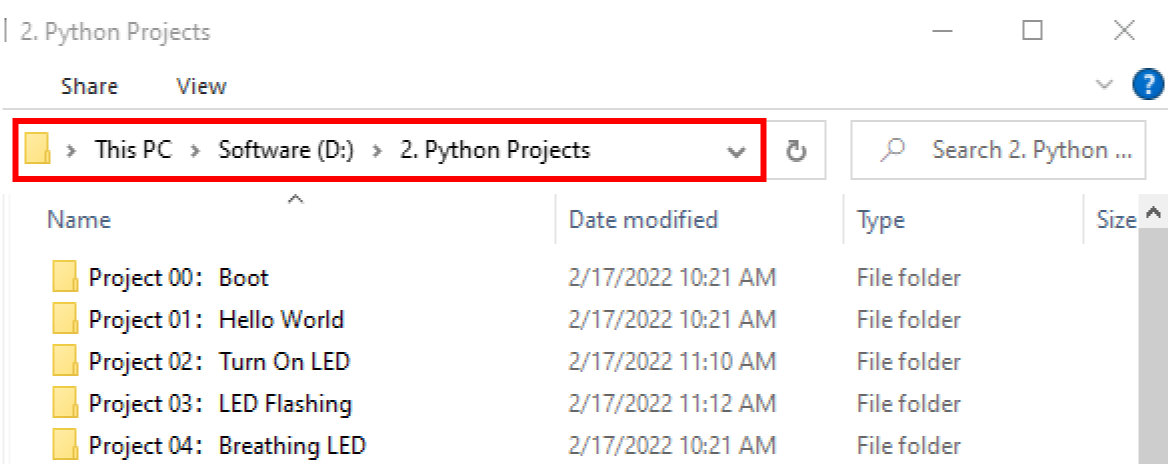
How to identify the 220 5-band resistor and 10K 5-band resistor



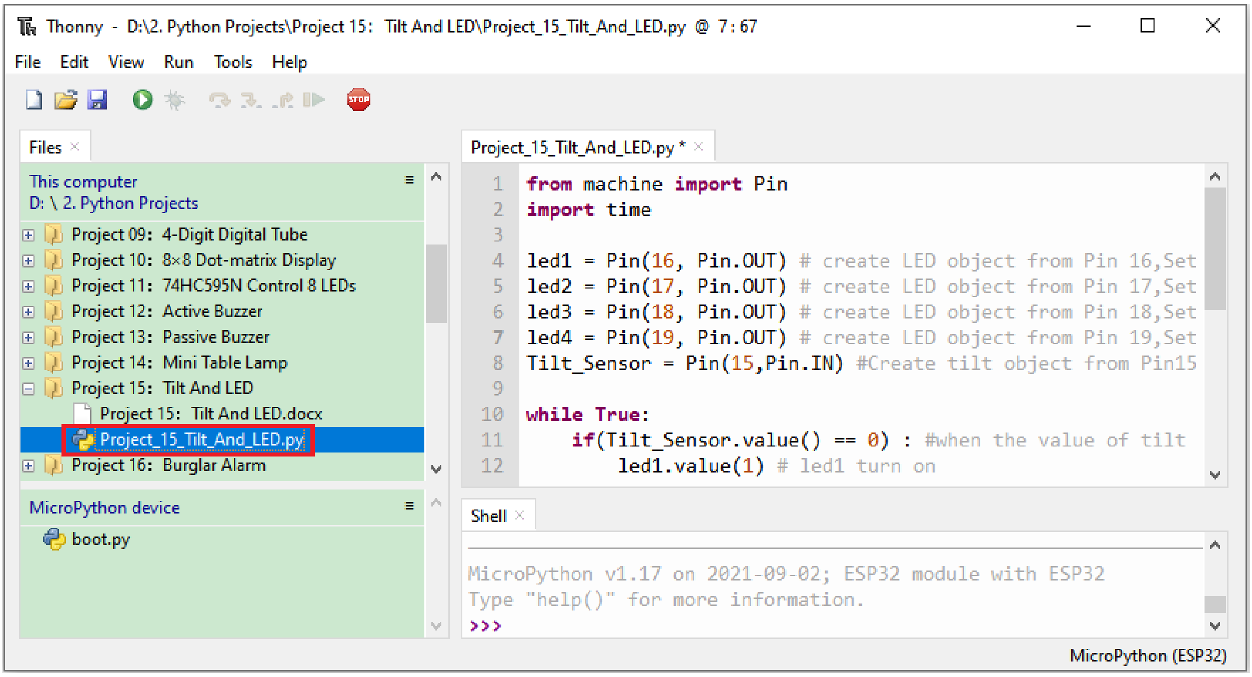


5.Project code

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



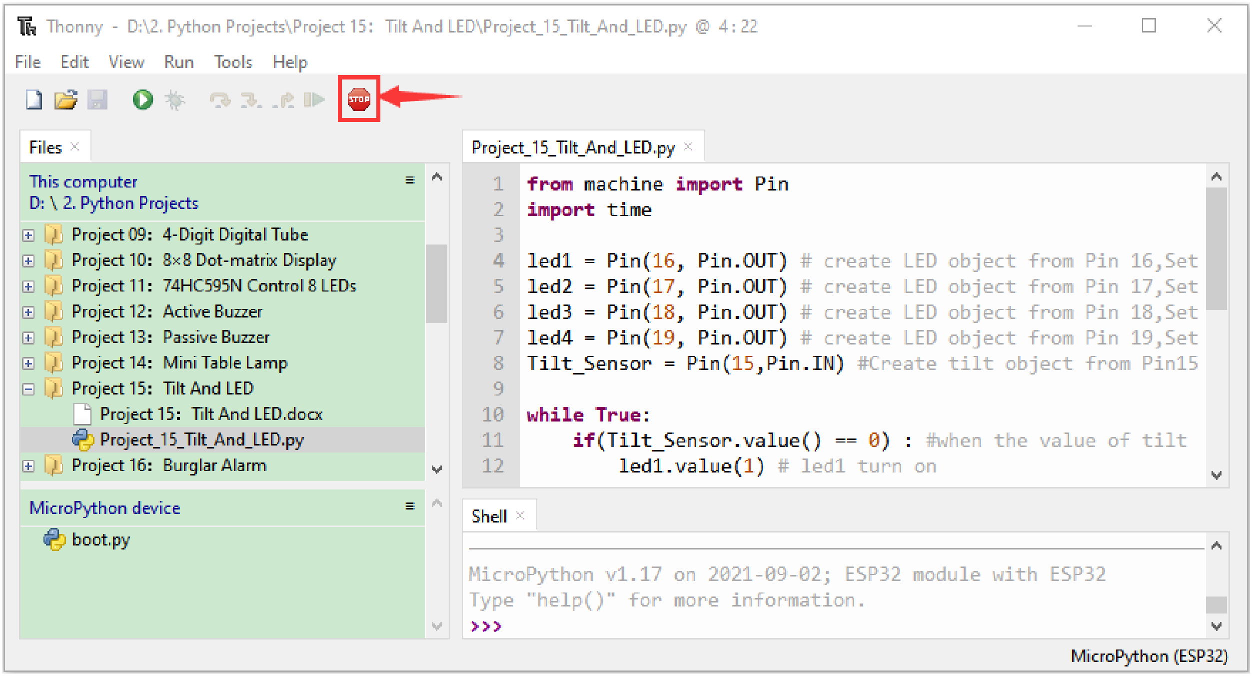
Open“Thonny”click“This computer”→“D:”→“2. Python Projects”→“Project 15: Tilt And LED”, and then double left-click “Project_15_Tilt_And_LED.py”.



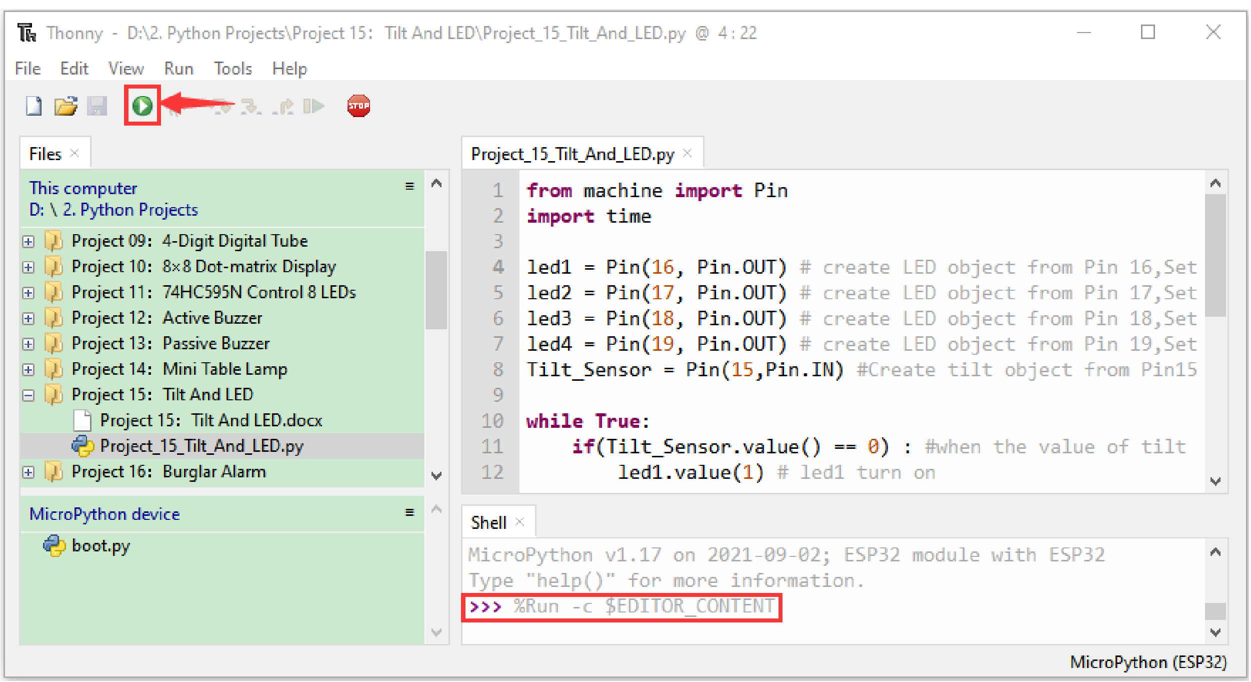

```
from machine import Pin
import time

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

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
```

6.Project result

Make sure the ESP32 has been connected to the computer, click STOP "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.

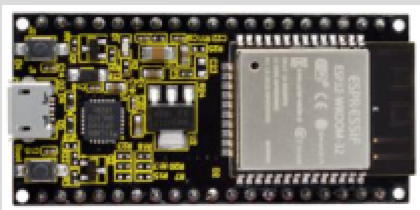
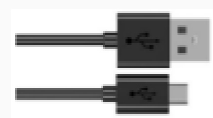


7.18 Project 16: I2C 128×32 LCD

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.

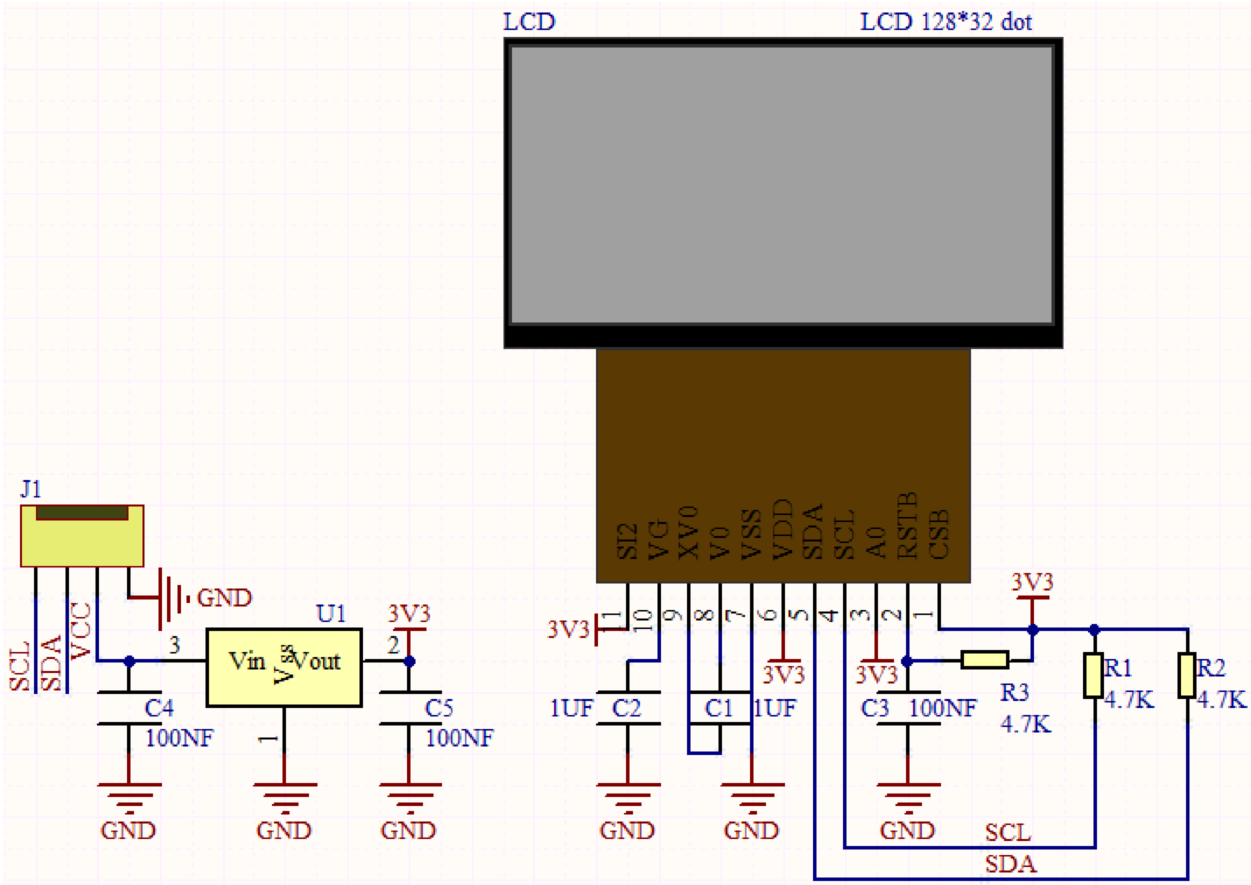
2.Components

		
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:

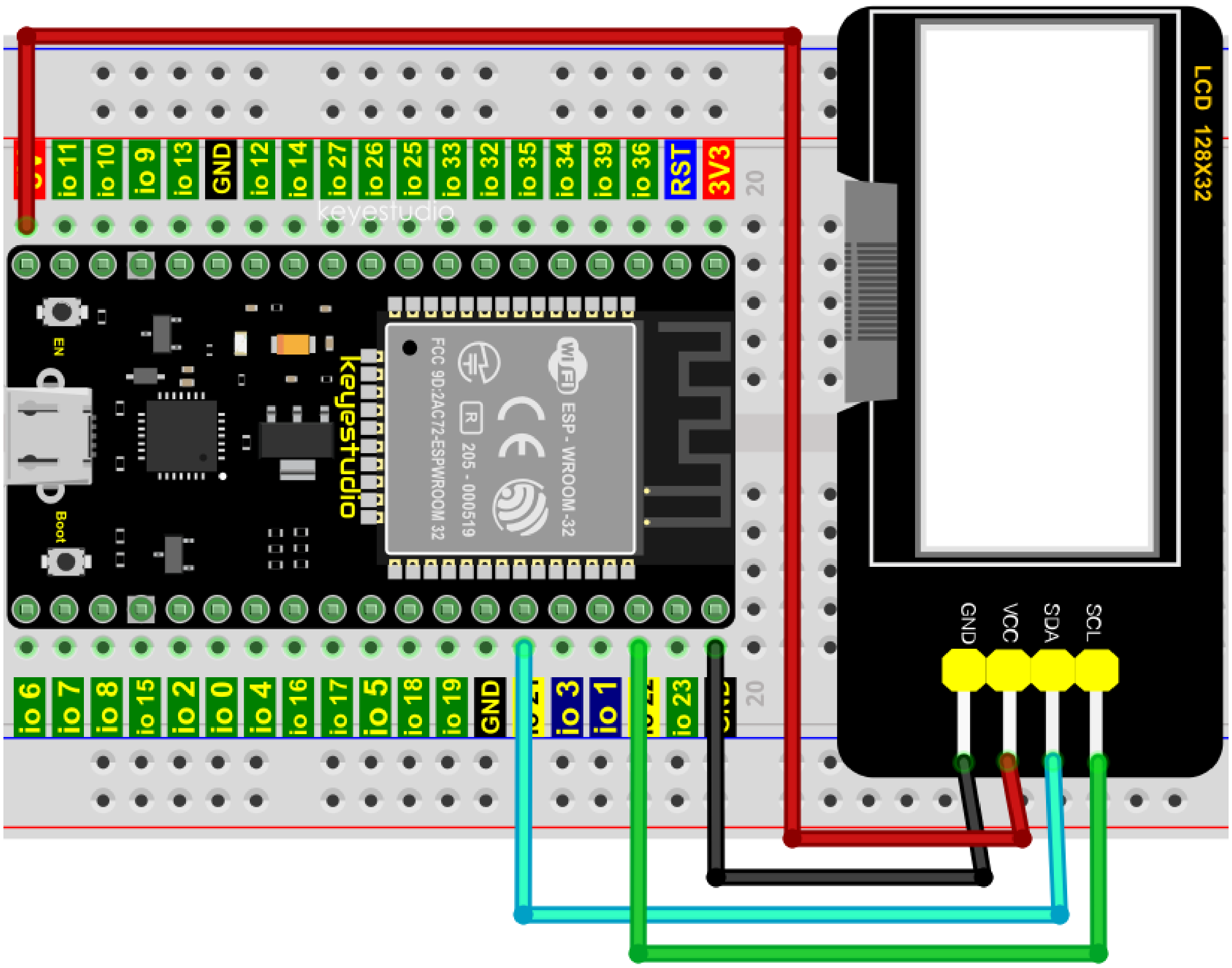
Pixel: 128*32 character

Operating voltage(chip)4.5V to 5.5V

Operating current100mA (5.0V)

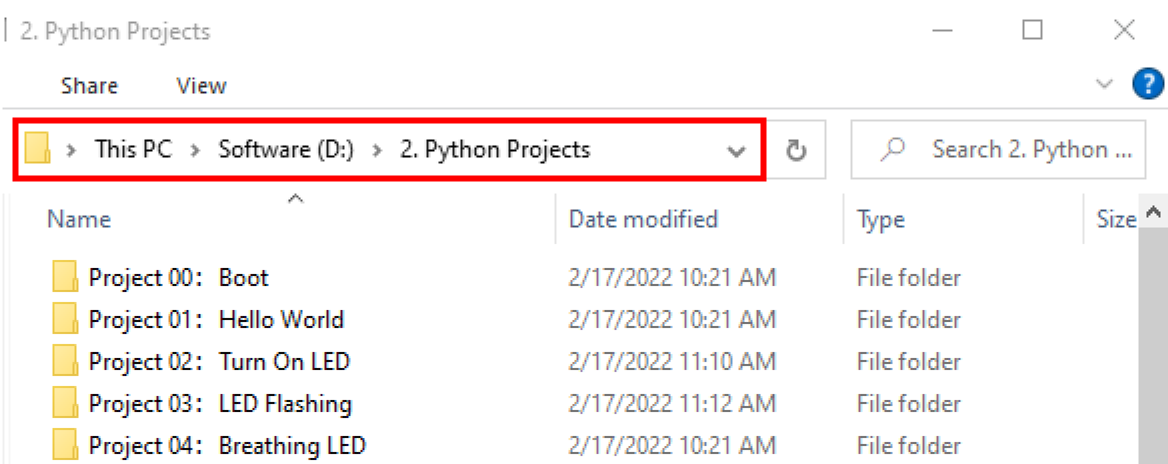
Optimal operating voltage(module):5.0V

4.Wiring Diagram

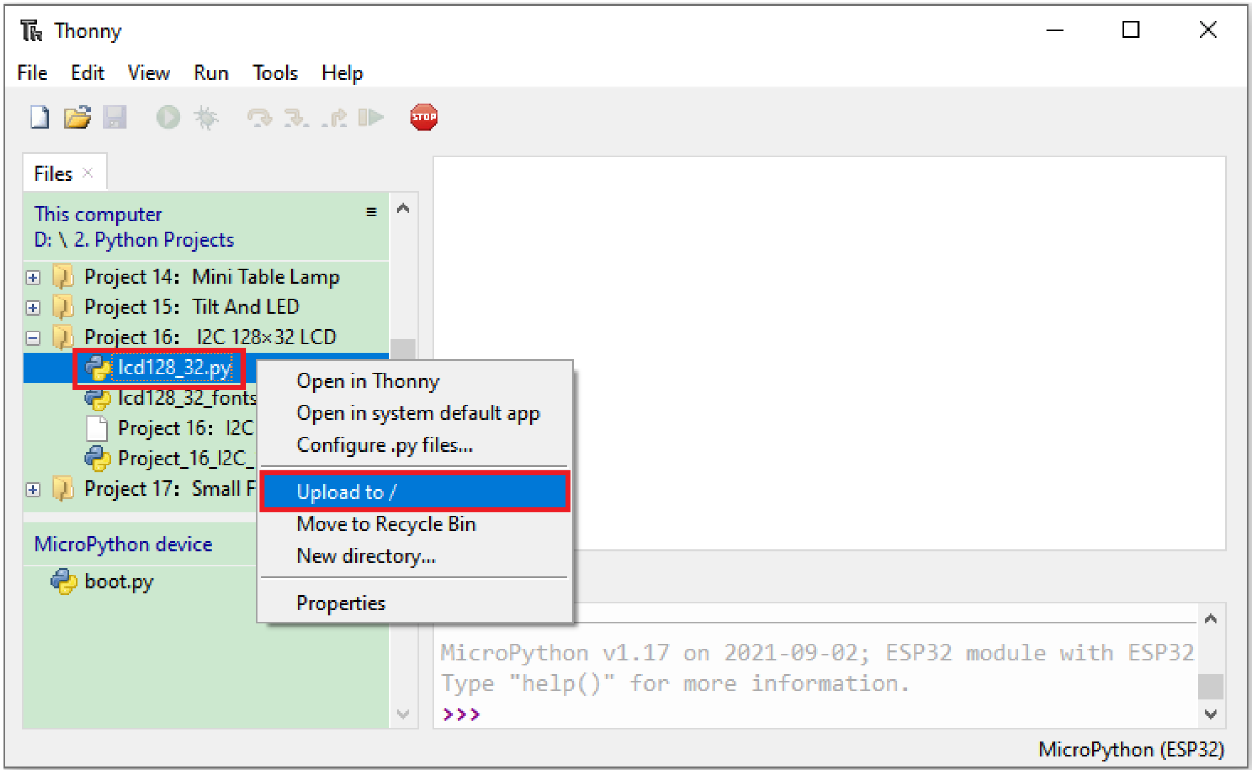


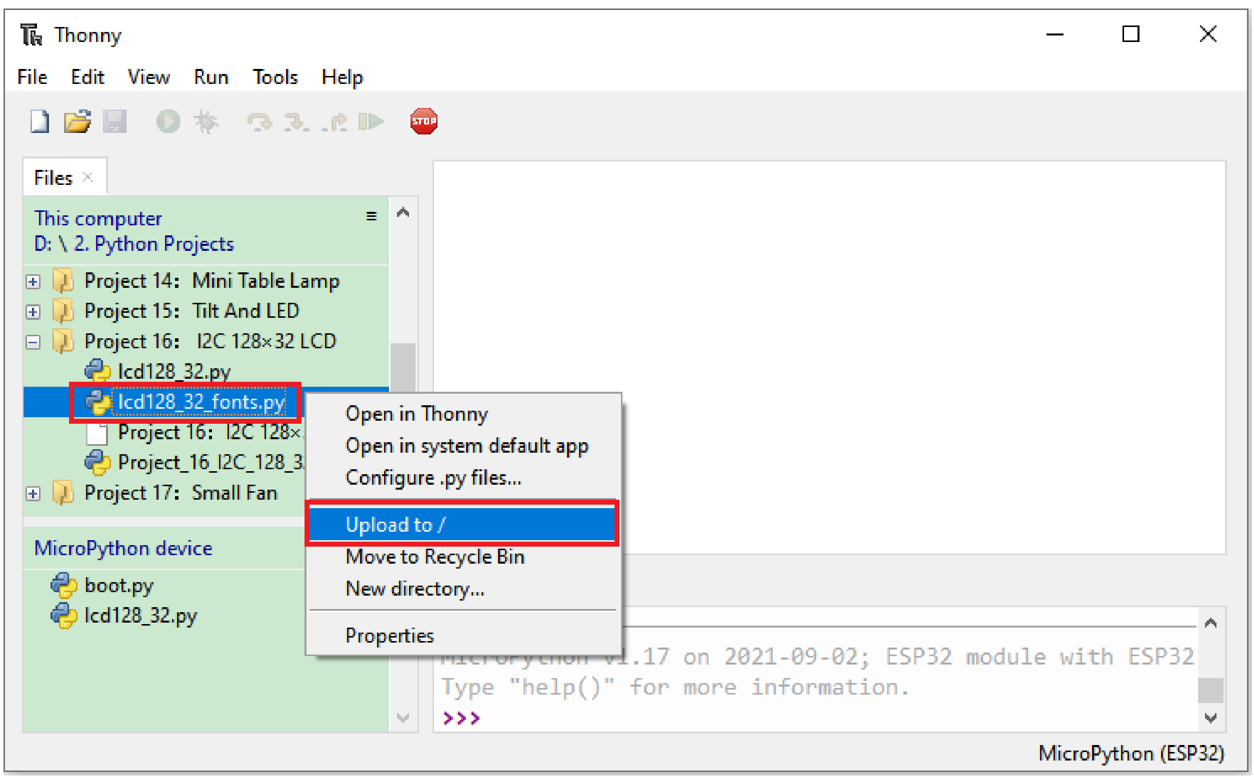
5.Project code

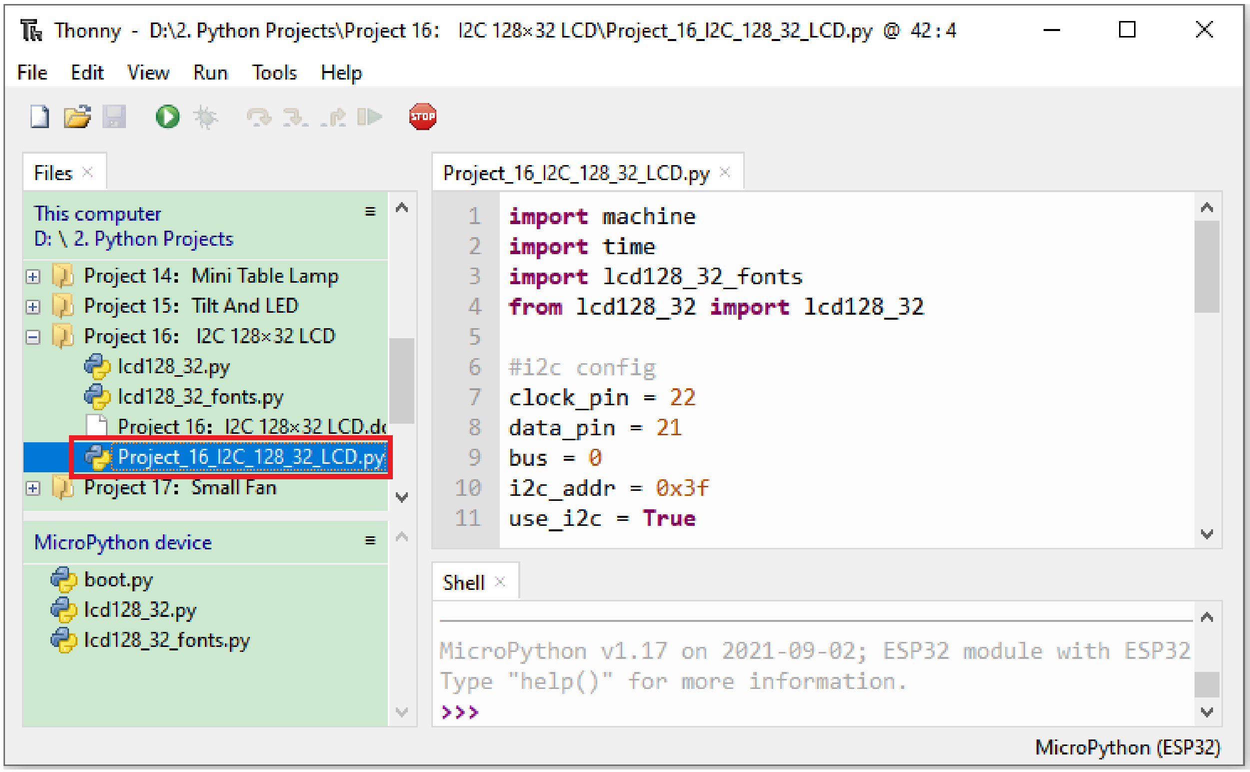
Codes used in this tutorial are saved in“2. Python Projects”. If you haven’t downloaded the code file yet, please click on the link to download it:Download Python Codes



Open“Thonny”click“This computer”→“D:”→“2. Python Projects”→“Project 16: I2C 128×32 LCD”. 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 then double left-click“Project_16_I2C_128_32_LCD.py”.


Thonny
File Edit View Run Tools Help
Files
This computer
D: \ 2. Python Projects
Project 14: Mini Table Lamp
Project 15: Tilt And LED
Project 16: I2C 128×32 LCD
lcd128_32.py
lcd128_32_fonts
Project 16: I2C
Project_16_I2C_
Project 17: Small F
MicroPython device
boot.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 (ESP32)


Thonny
File Edit View Run Tools Help
Files
This computer
D: \ 2. Python Projects
Project 14: Mini Table Lamp
Project 15: Tilt And LED
Project 16: I2C 128×32 LCD
lcd128_32.py
lcd128_32_fonts.py
Project 16: I2C 128×
Project_16_I2C_128_3
Project 17: Small Fan
MicroPython device
boot.py
lcd128_32.py
Open in Thonny
Open in system default app
Configure .py files...
Upload to /
Move to Recycle Bin
New directory...
Properties
.17 on 2021-09-02; ESP32 module with ESP32
Type "help()" for more information.
>>>
MicroPython (ESP32)




```
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)
```

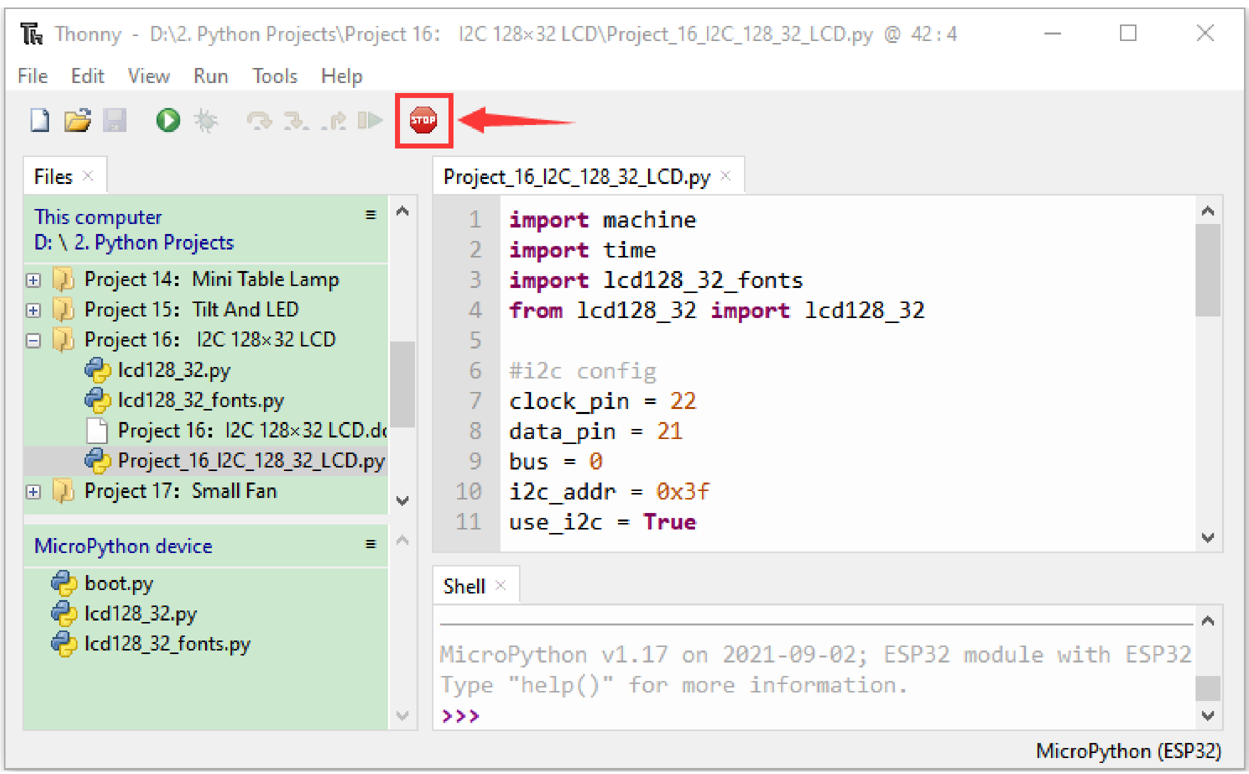
(continues on next page)

(continued from previous page)

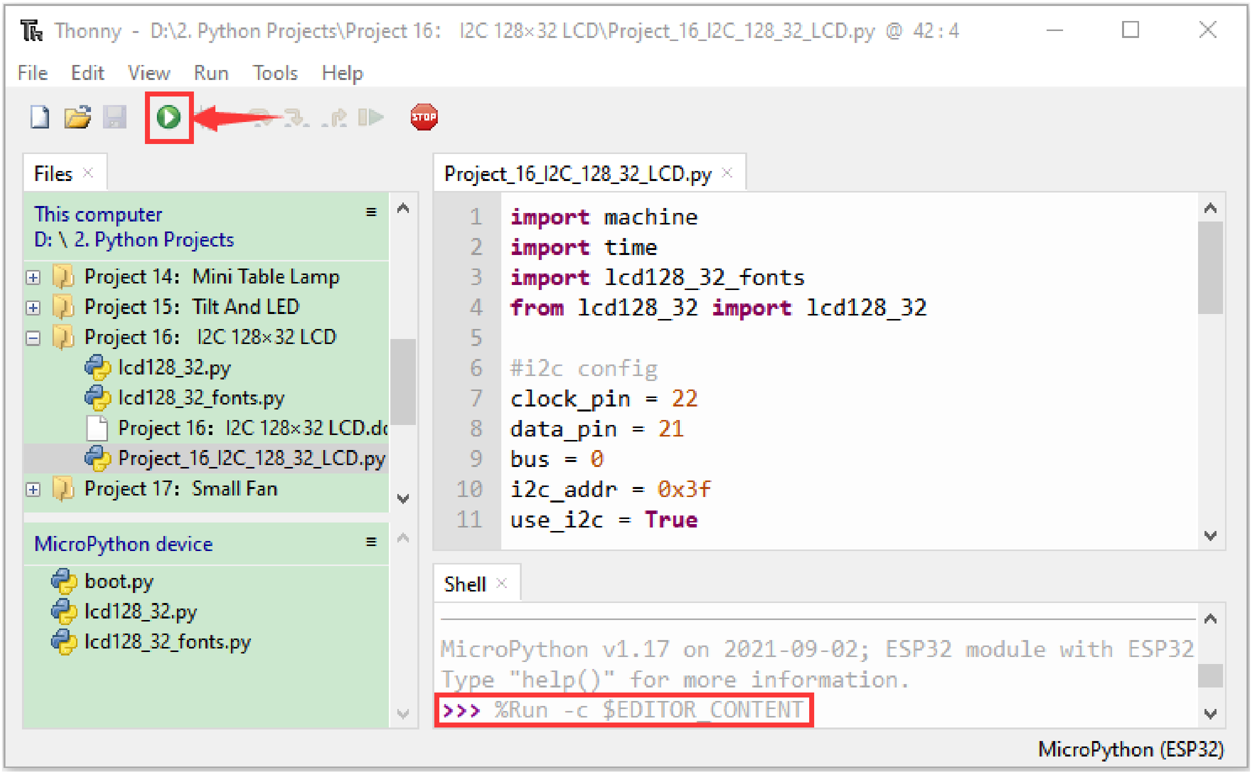
```
lcd.Display("KEYESTUDIO")
lcd.Cursor(1, 0)
lcd.Display("ABCDEFGHIJKLMNOPQR")
lcd.Cursor(2, 0)
lcd.Display("123456789+-*/<>=$@")
lcd.Cursor(3, 0)
lcd.Display("%^&(){}:;'|?,.~\\[]")
"""
while True:
    scan_for_devices()
    time.sleep(0.5)
"""
```

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

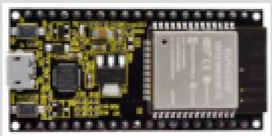
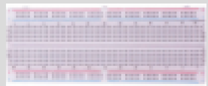
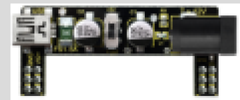
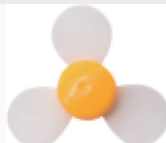
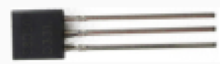
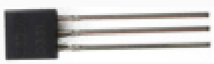
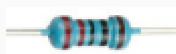
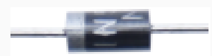
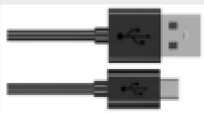


7.19 Project 17Small Fan

1.Introduction

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

2.Components

			
ESP32*1	Breadboard*1	6 AA Battery Holder*1	Breadboard Power Module*1
			
AA Battery(Self-prepared)*6	Fan*1	DC Motor*1	NPN Transistor (S8050)*1
			
PNP Transistor (S8550)*1	1K Resistor*1	Jumper Wire	Diode*1
 USB Cable*1			

Keyestudio Breadboard Power Supply Module

Introduction:

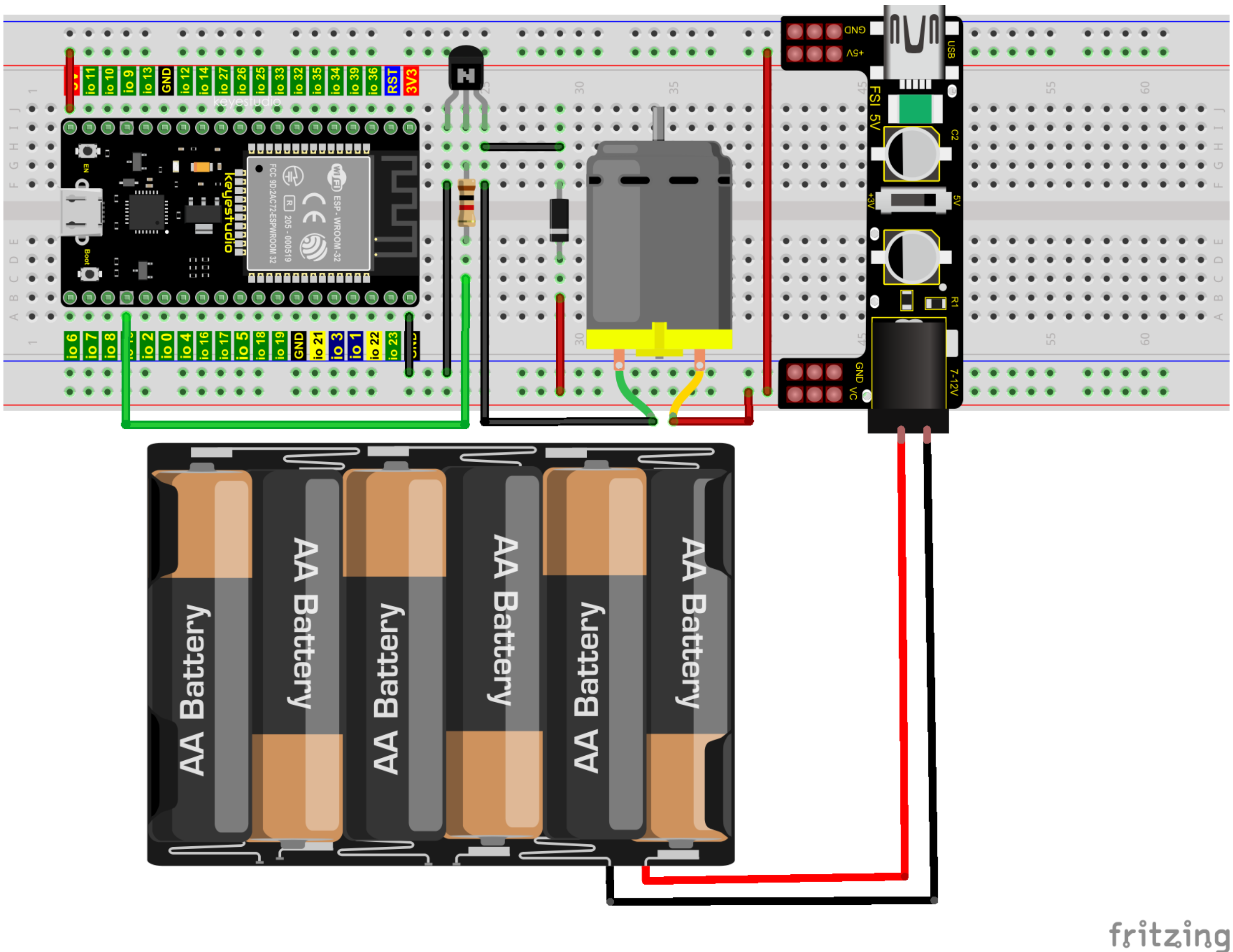
This breadboard power supply module is compatible with 5V and 3.3V, which can be applied to MB102 breadboard. The module contains two channels of independent control, powered by the USB all the way.

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 DC 5V, and the output voltage of the DC 6.5-12V power supply terminal of the other power supply is DC3.3V.

3.Wiring Diagram 1

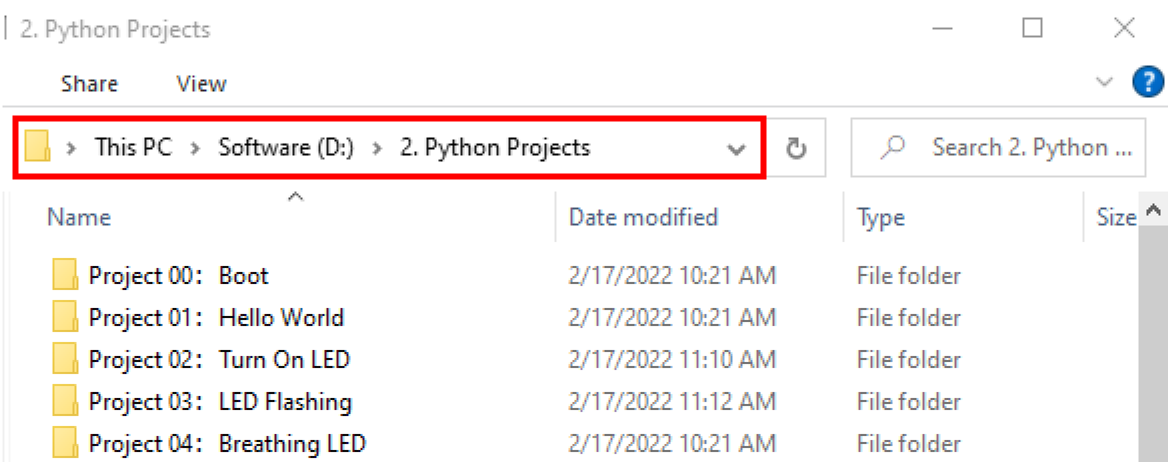
We use the S8050NPN transistor) to control the motor



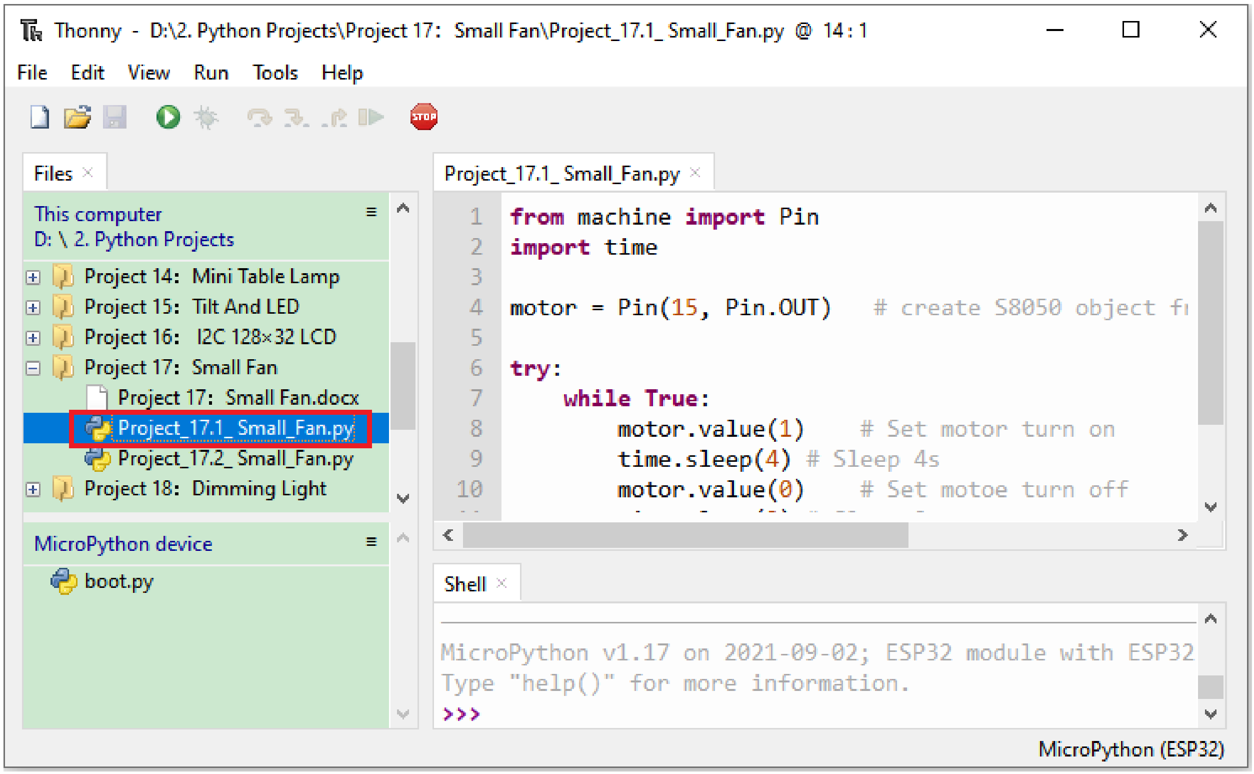
Wire up first, then connect a fan at the DC motor

5.Test Code 1

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



Open“Thonny”click“This computer”→“D:”→“2. Python Projects”→“Project 17: Small Fan”, and then double left-click“Project_17.1_ Small_Fan.py”.

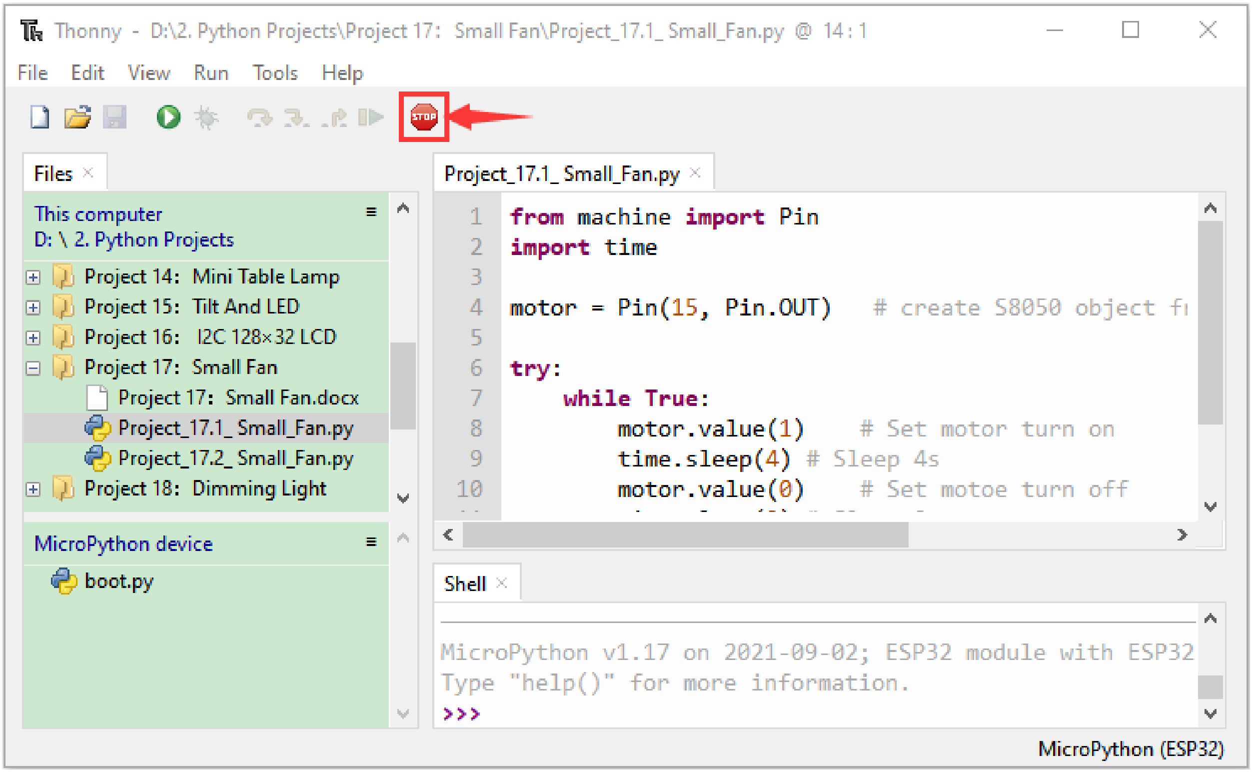



```
from machine import Pin
import time

motor = Pin(15, Pin.OUT)   # create S8050 object from Pin 15, Set Pin 15 to output

try:
    while True:
        motor.value(1)     # Set motor turn on
        time.sleep(4) # Sleep 4s
        motor.value(0)     # Set motoe turn off
        time.sleep(2) # Sleep 2s
except:
    pass
```

Ensure the ESP32 is connecteed to the computer and tap "Stop/Restart backend".

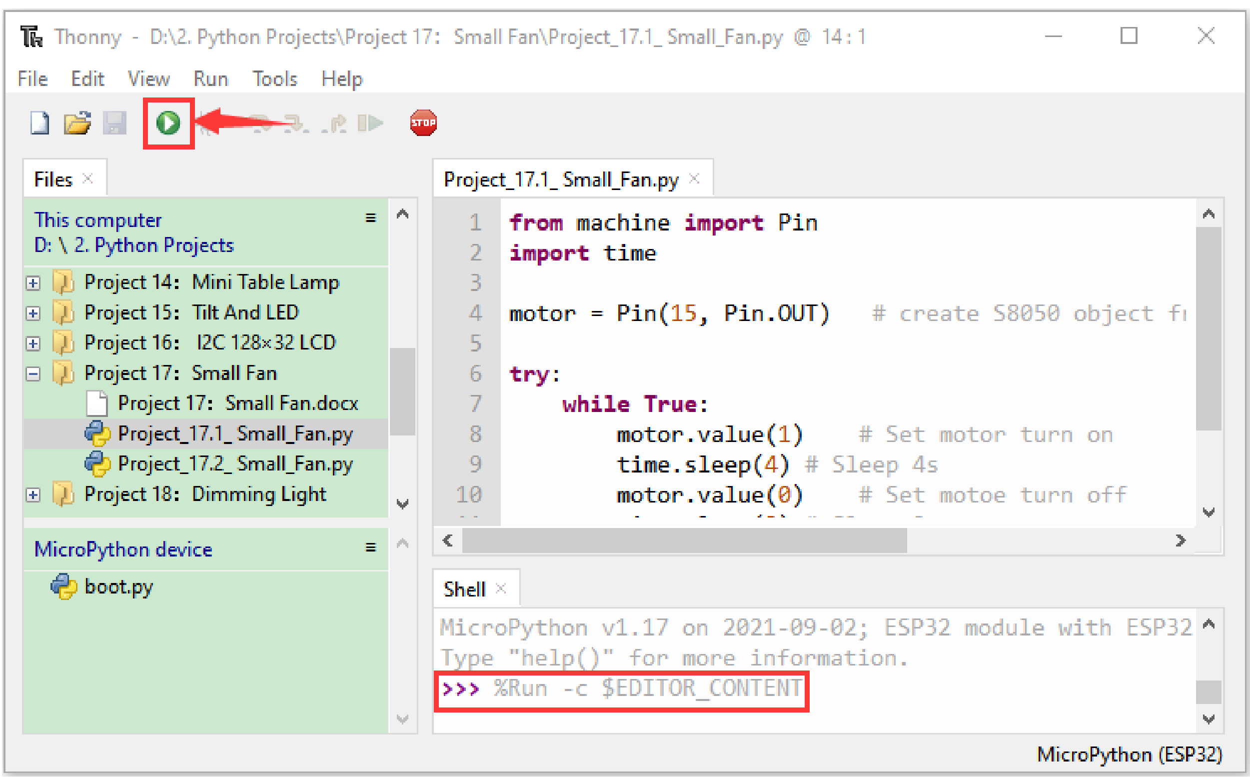
6.Project result

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



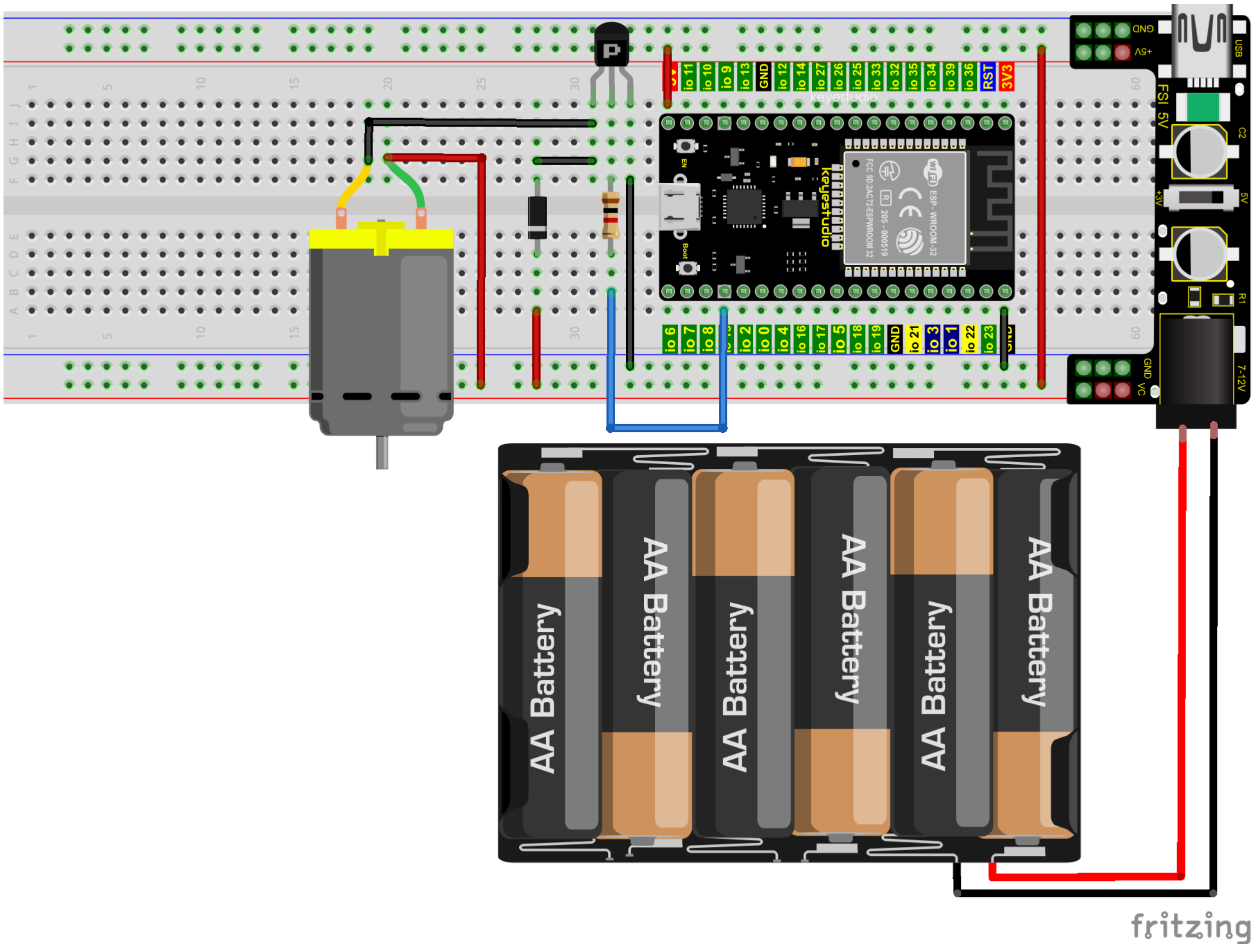
Power up and click "Run current script", the code starts to be executed and you'll see that the small fan turn for 4s and stop for 2s.

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



7.Wiring Diagram 2

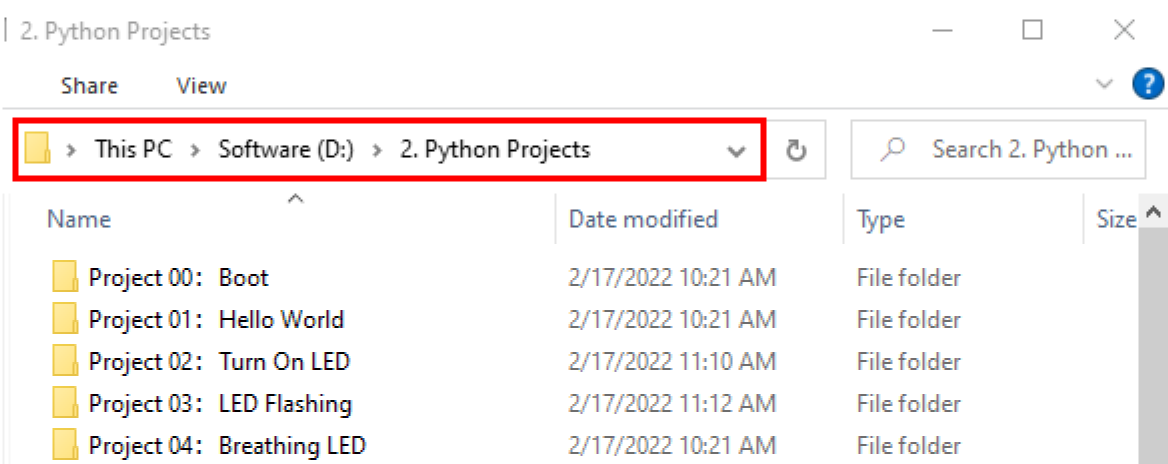
We use the S8050PNP transistor) to control the motor



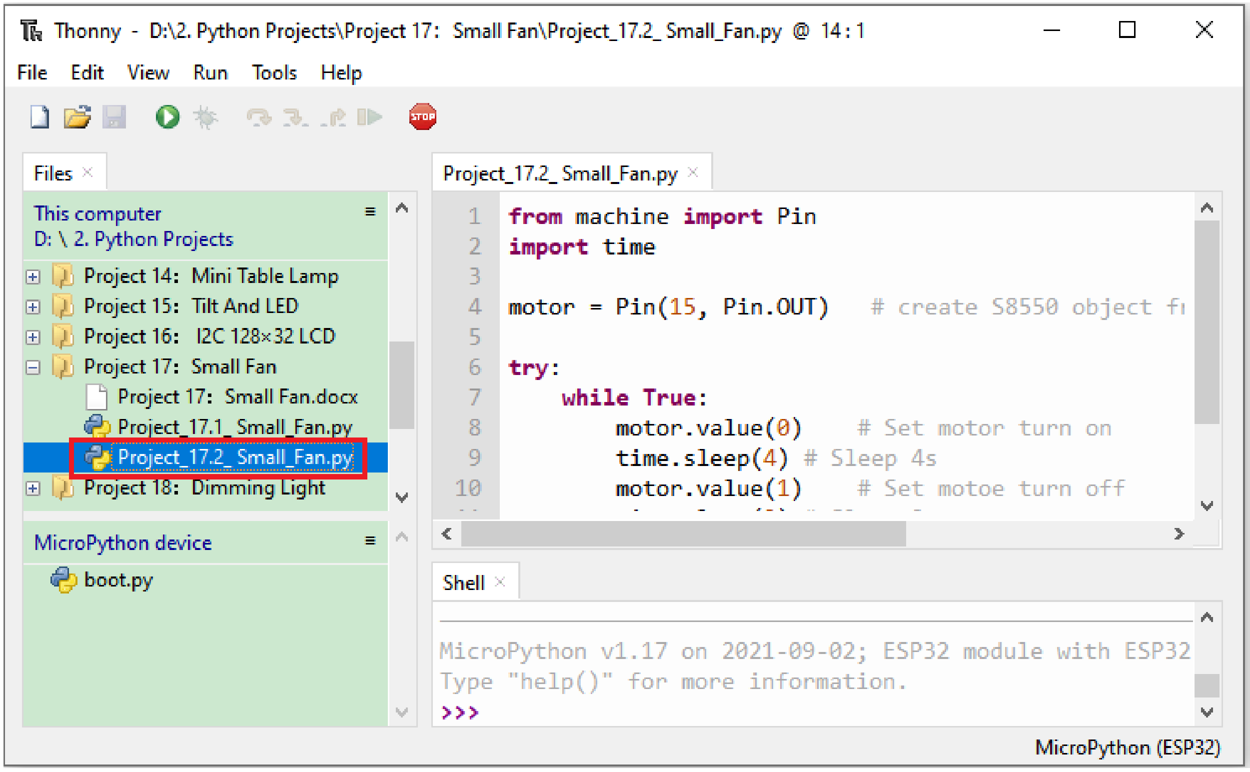
Wire up first, then connect a fan at the DC motor

8.Test Code 2

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



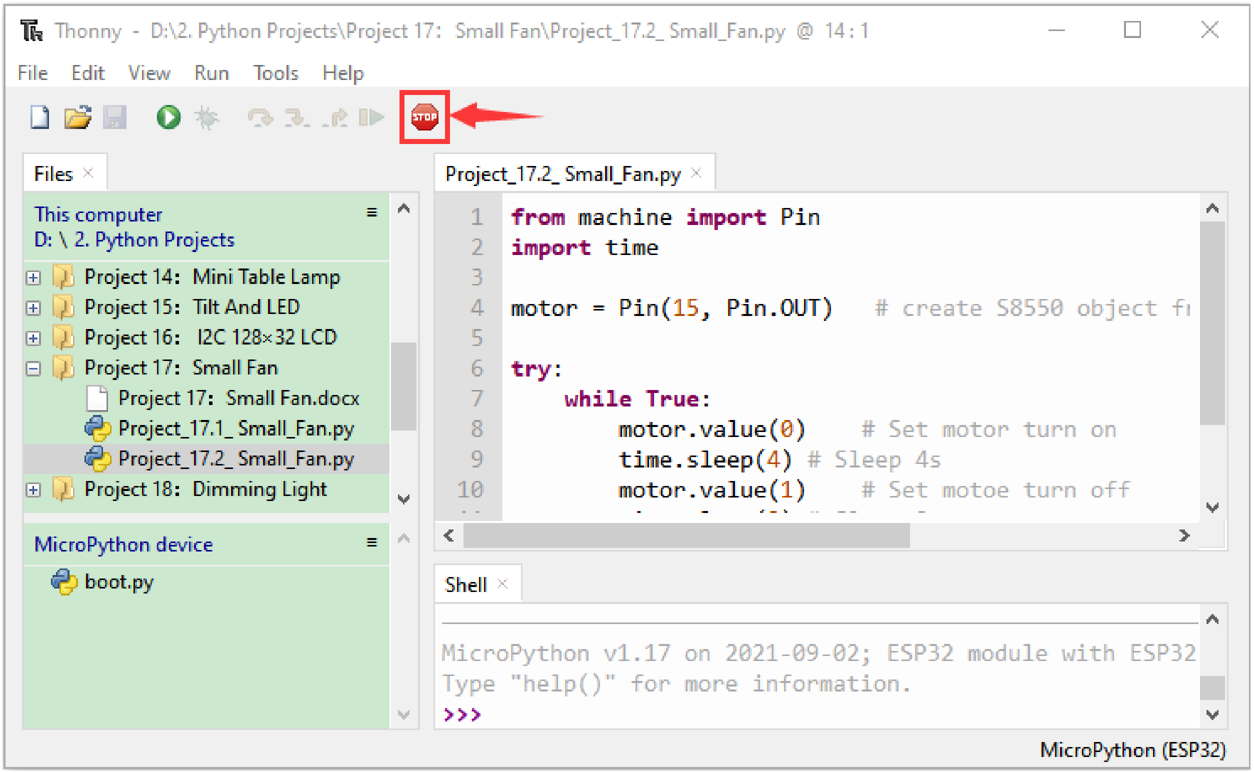
Open“Thonny”click“This computer”→“D:”→“2. Python Projects”→“Project 17Small Fan”, and then double left-click“Project_17.2_ Small_Fan.py”.



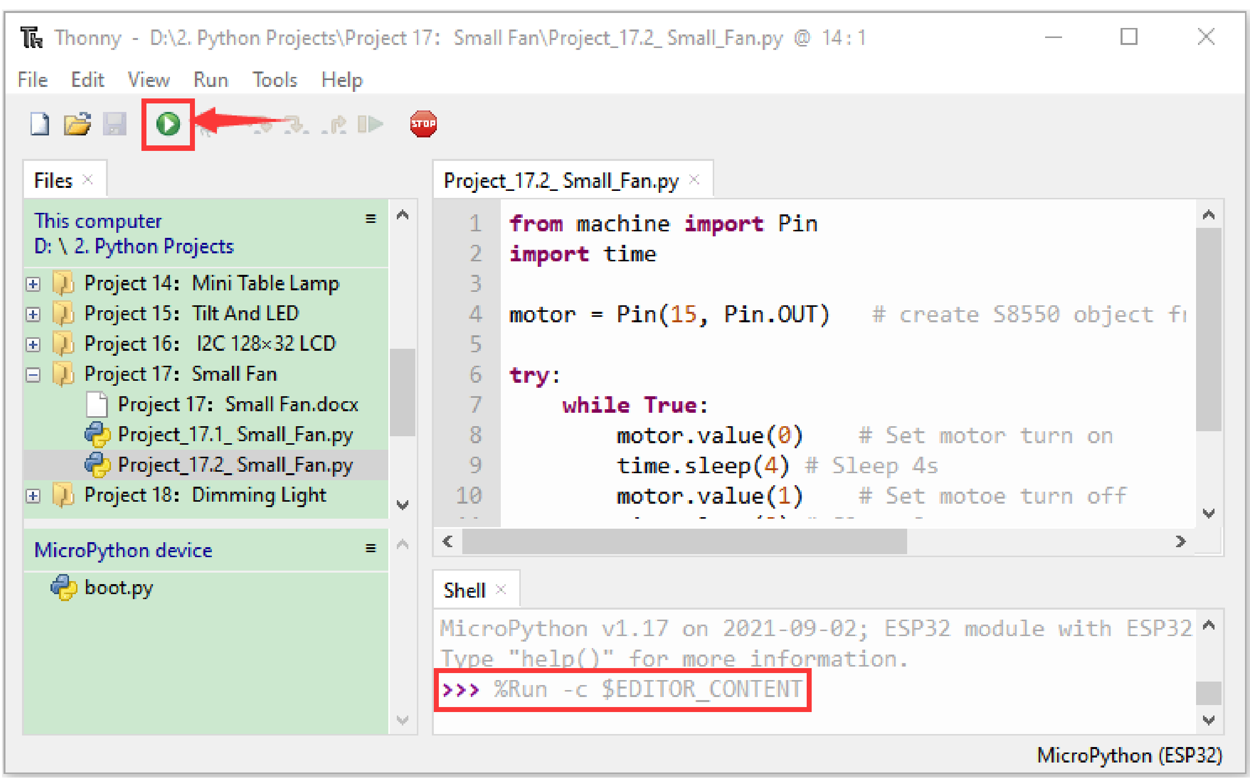

```python
from machine import Pin
import time

motor = Pin(15, Pin.OUT)   # create S8550 object from Pin 15, Set Pin 15 to output

try:
    while True:
        motor.value(0)     # Set motor turn on
        time.sleep(4) # Sleep 4s
        motor.value(1)     # Set motoe turn off
        time.sleep(2) # Sleep 2s
except:
    pass
```

9.Test Result 2

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



Power up and click "Run current script", the code starts to be executed and you'll see that the small fan turn for 4s and stop for 2s. Press"Ctrl+C"or click "Stop/Restart backend"to exit the program.



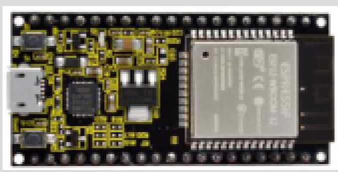
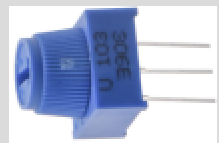
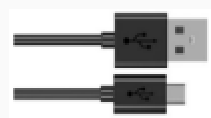
7.20 Project 18Dimming Light

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.

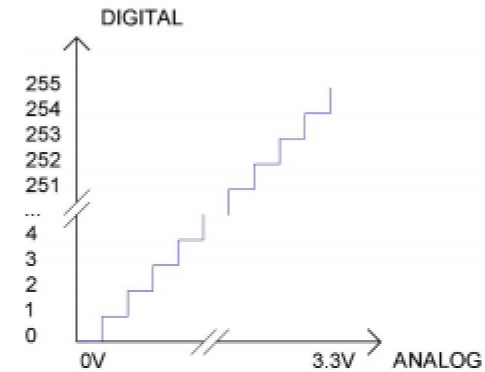
2.Components

			
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:

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

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:

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

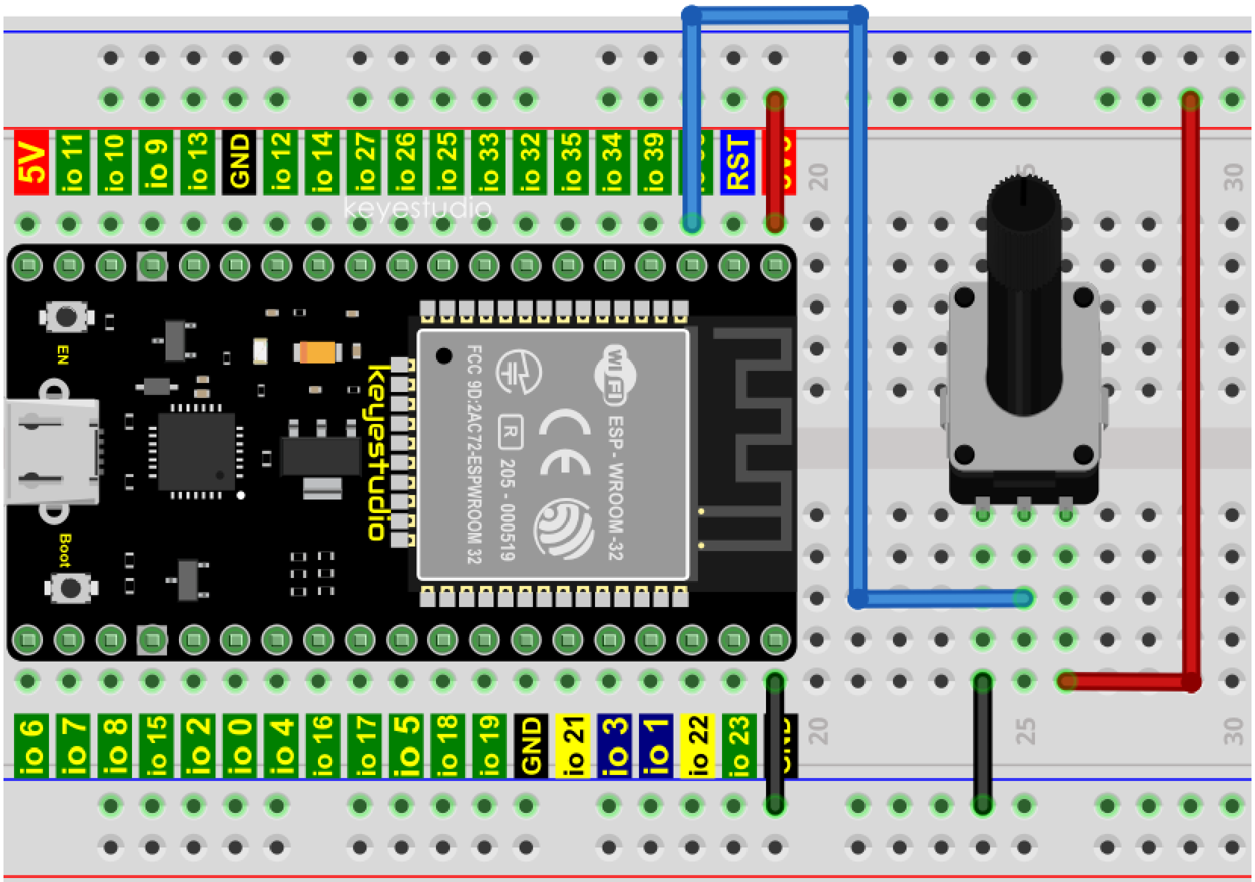
ADC on ESP32

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

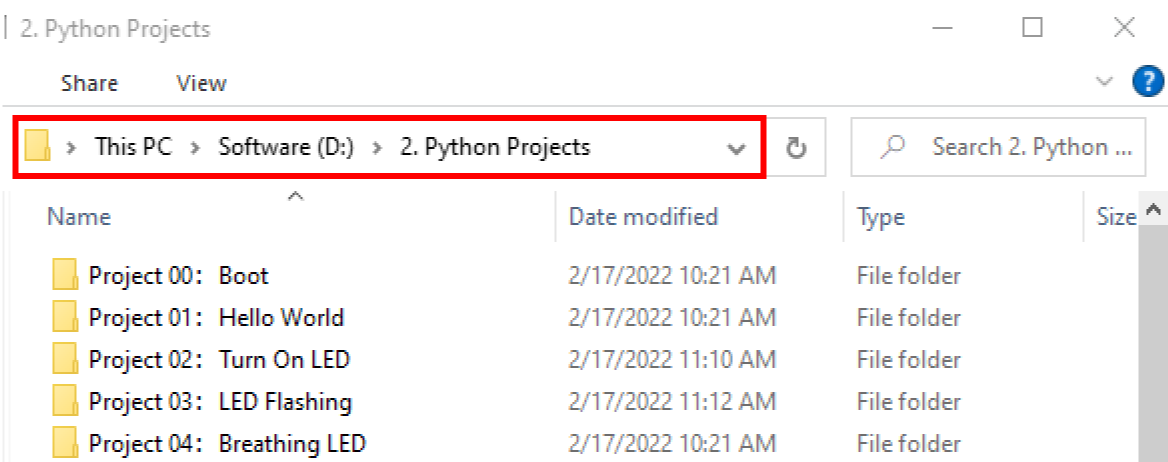
DAC on ESP32

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

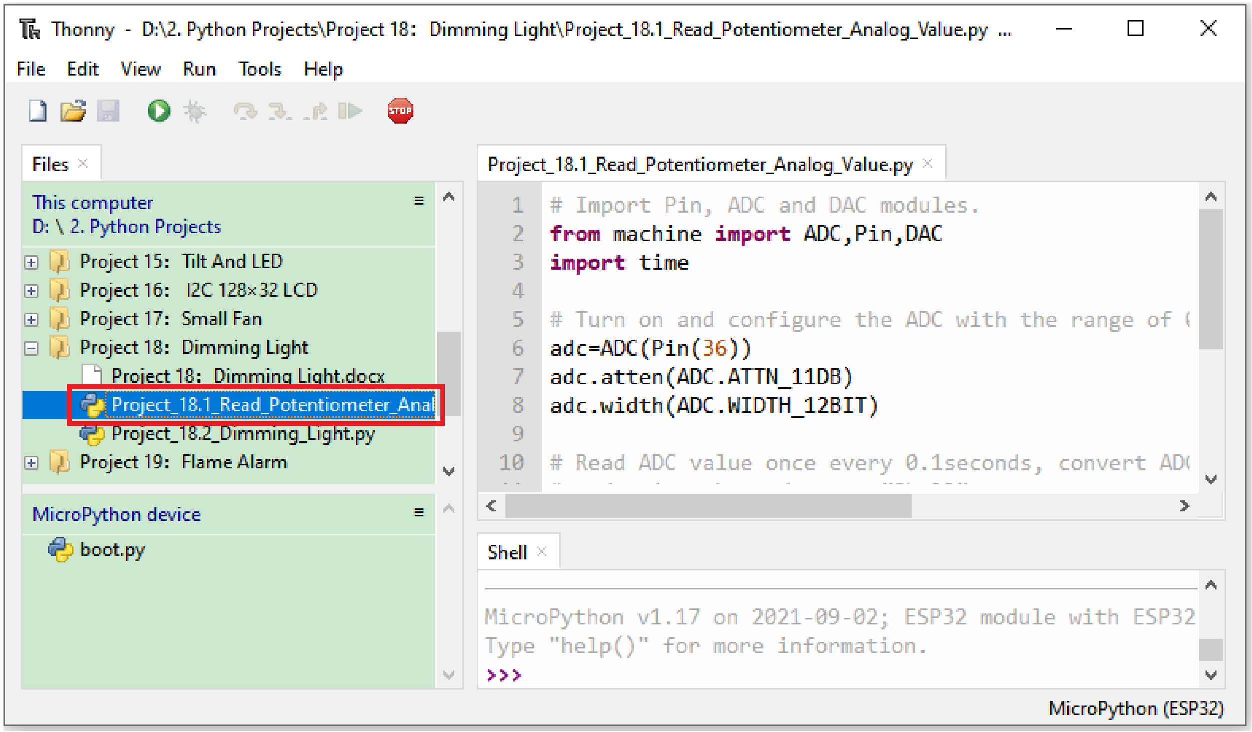
We connect the potentiometer to the analog IO port of ESP32 to read the ADC value, DAC value and voltage value of the potentiometer, please refer to the wiring diagram below



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

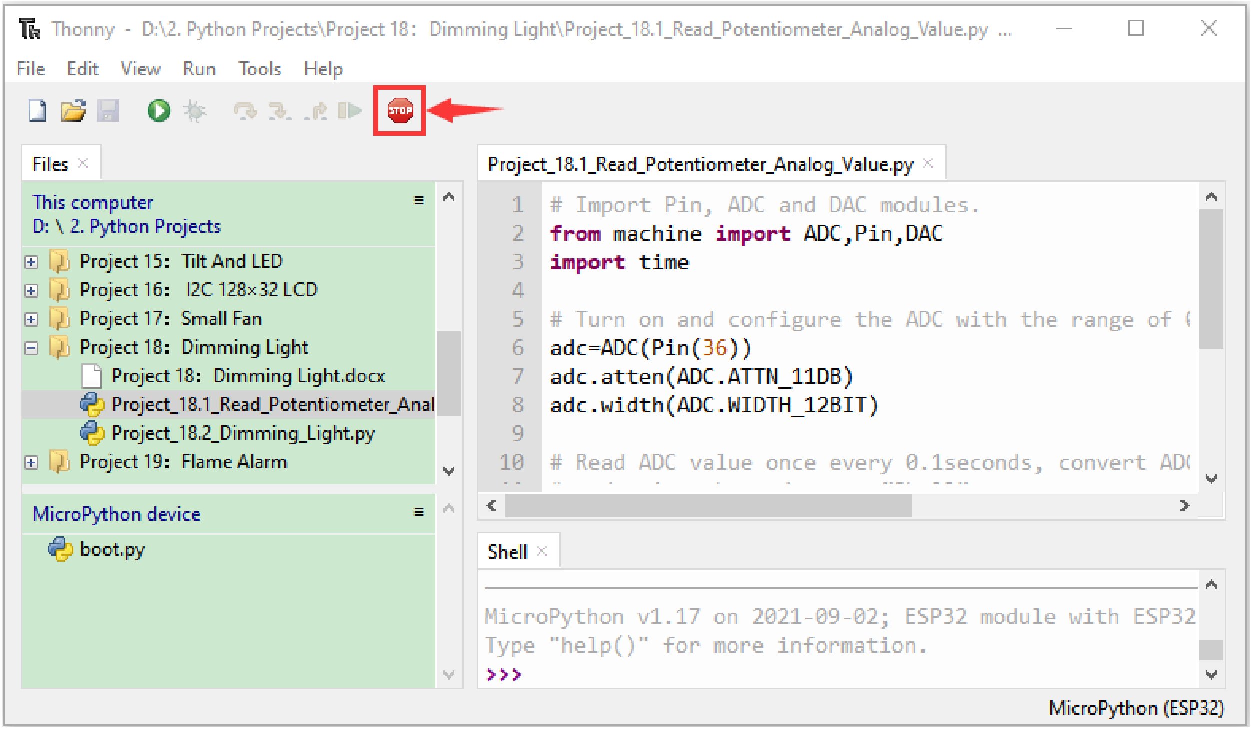


Open"Thonny"click"This computer"→"D:"→"2. Python Projects"→"Project 18Dimming Light"and then double left-click "Project_18.1_Read_Potentiometer_Analog_Value.py".

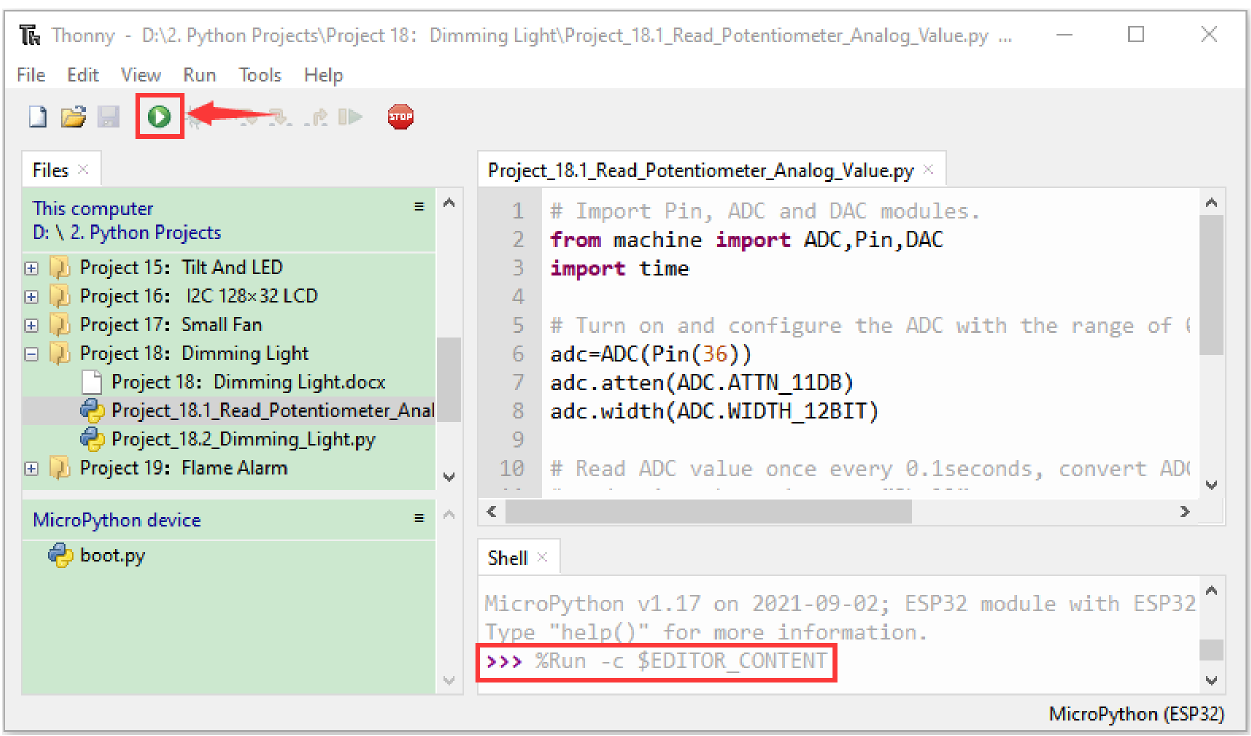



```python
# Import Pin, ADC and DAC modules.
from machine import ADC,Pin,DAC
import time

# 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 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.

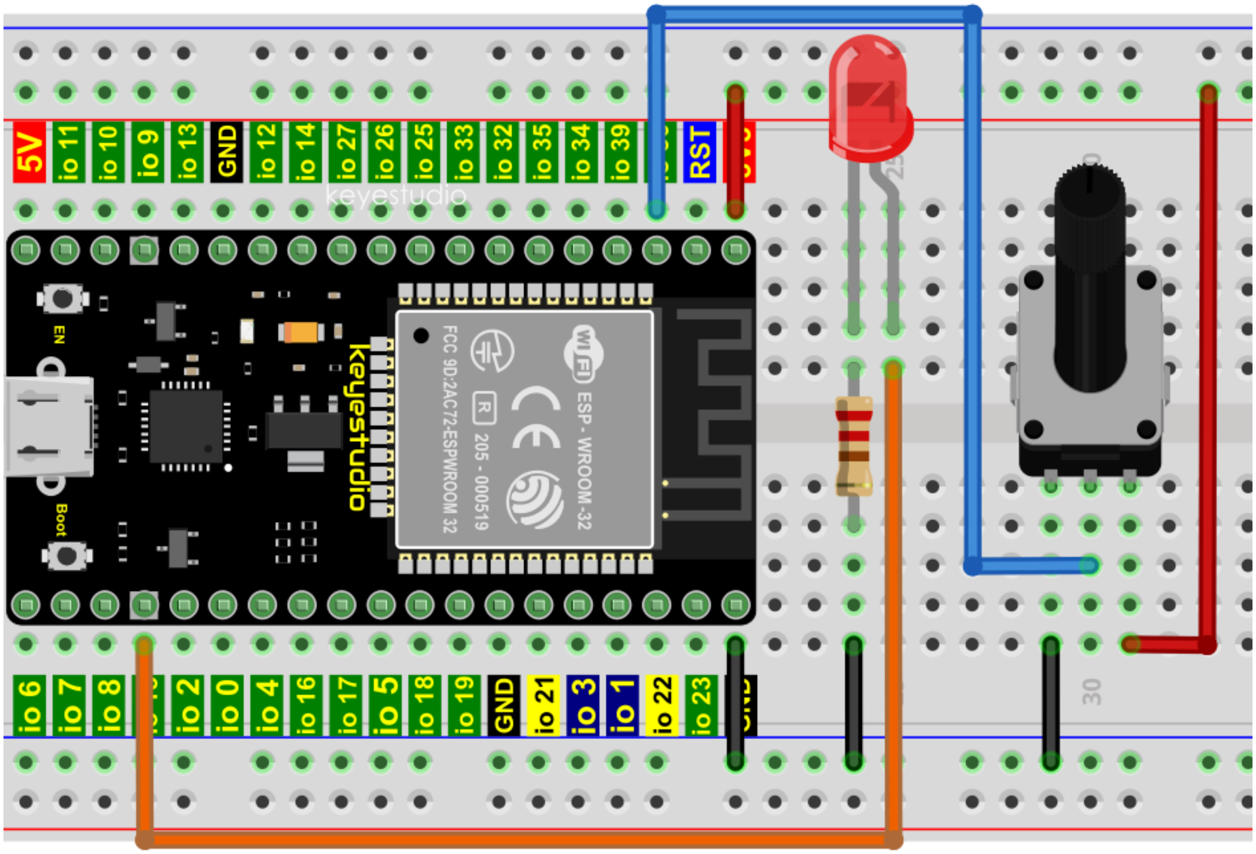


Shell

```
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
```

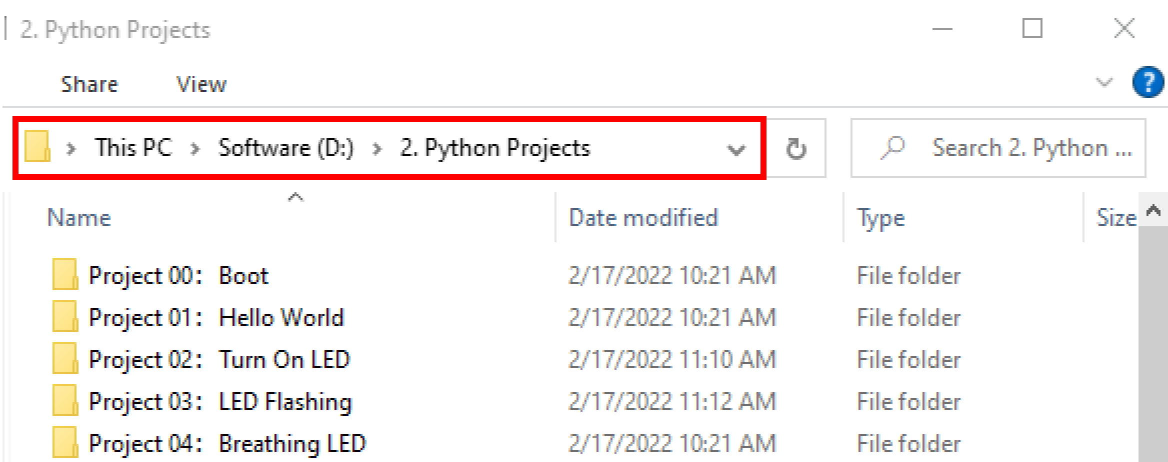
5.Wiring diagram of the dimming lamp

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

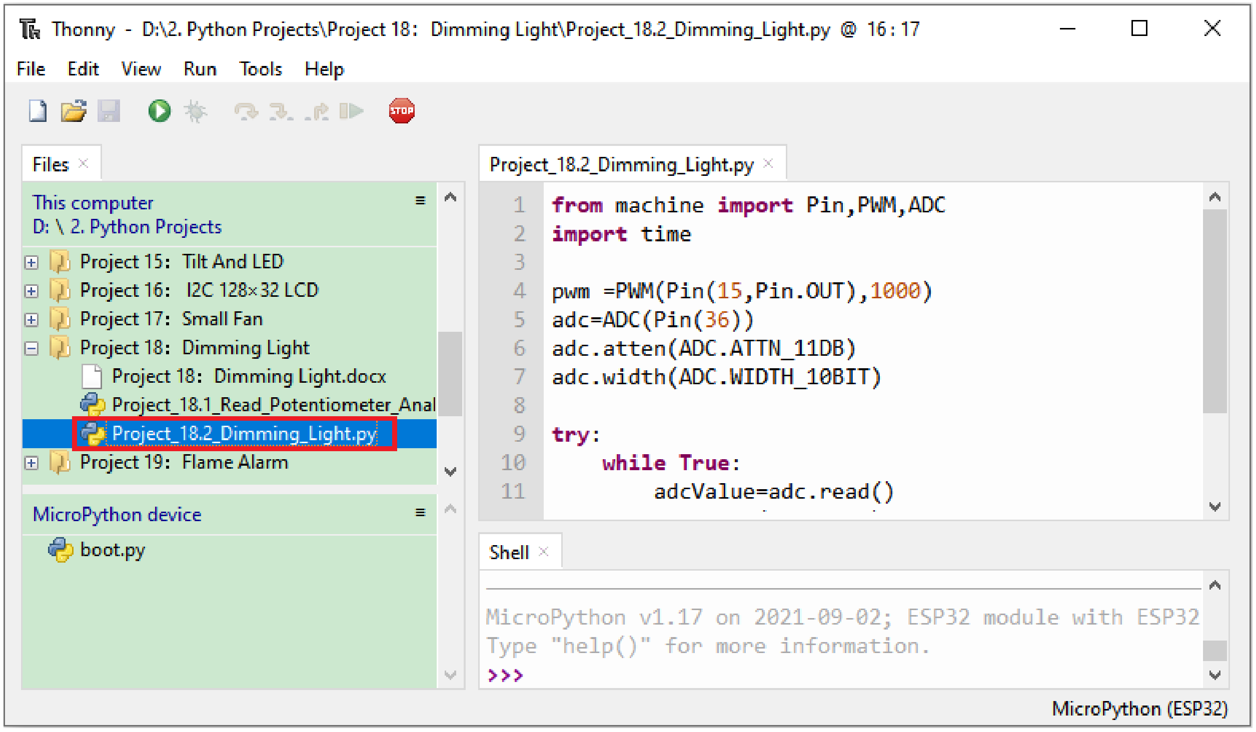


6.Project code

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



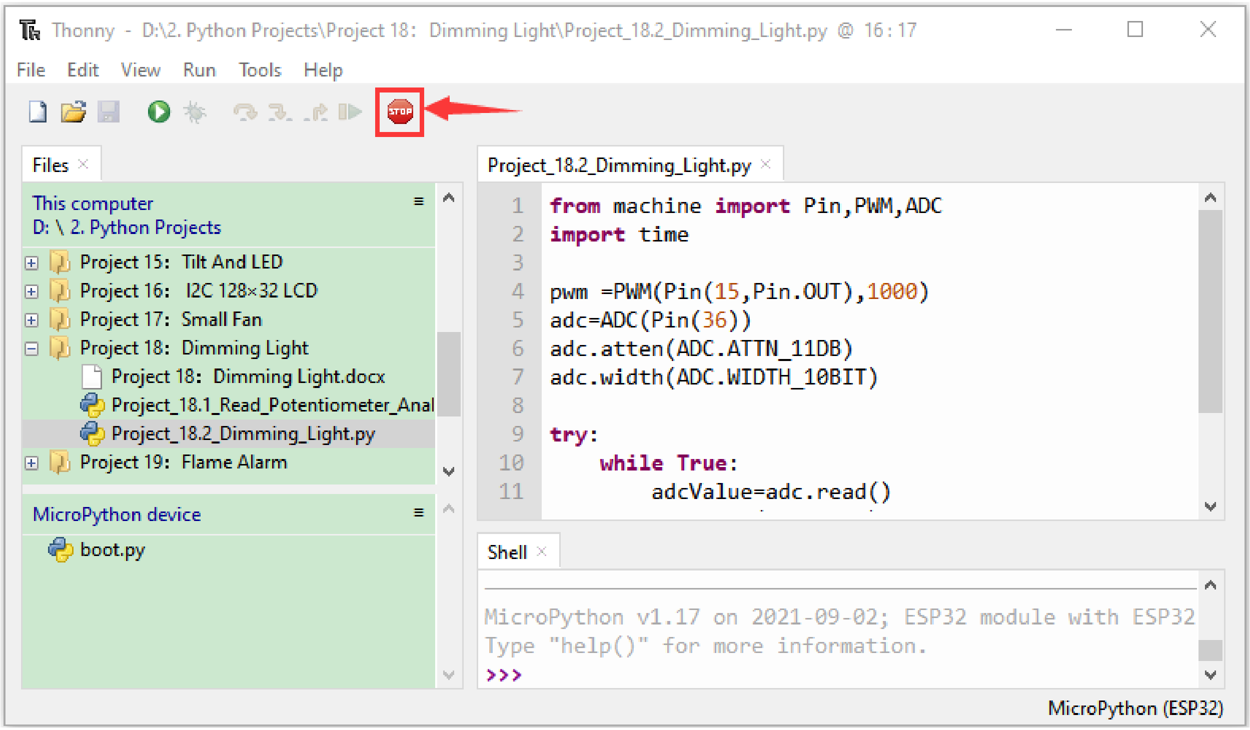
Open"Thonny"click"This computer"→"D:"→"2. Python Projects"→"Project 18: Dimming Light"and then double left-click"Project_18.2_Dimming_Light.py".



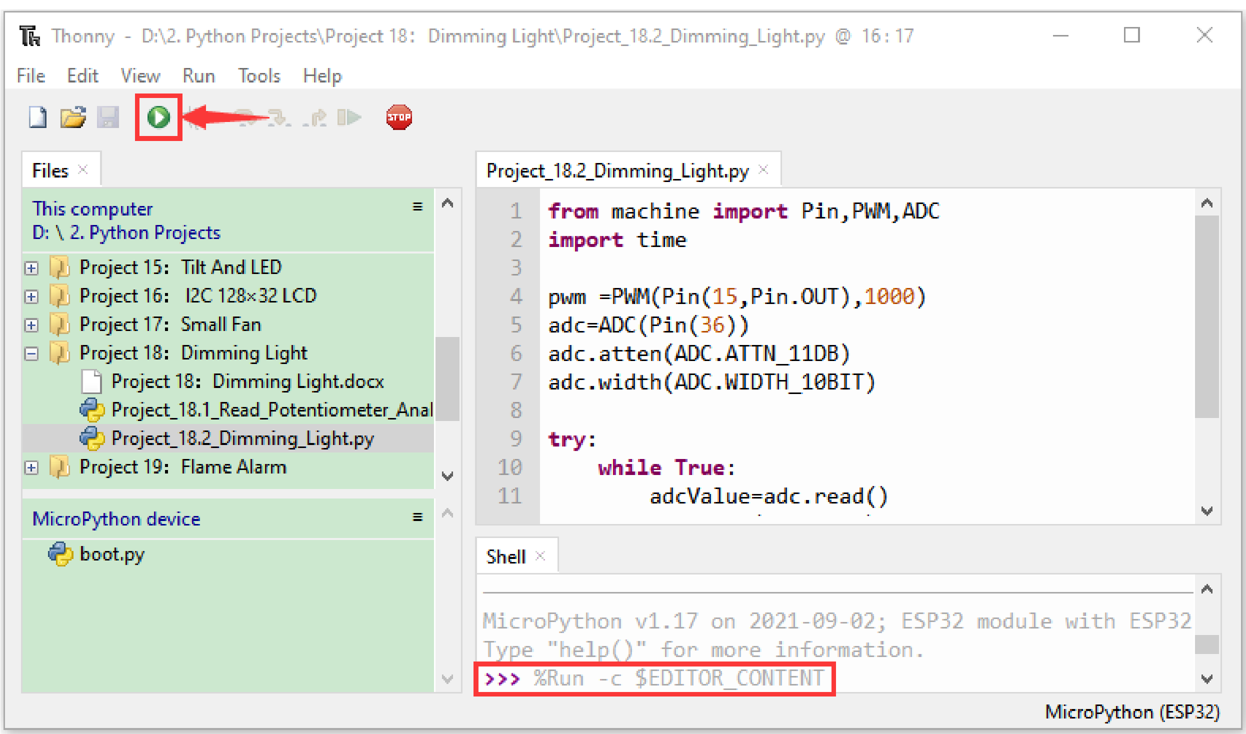

```
from machine import Pin,PWM,ADC
import time

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()
```

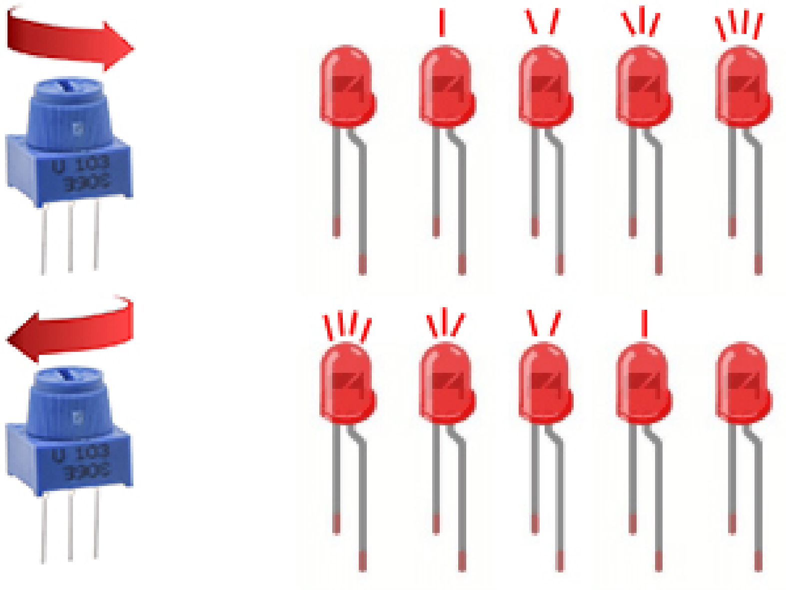
7.Project result

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.



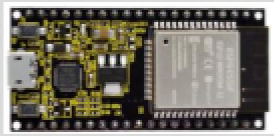
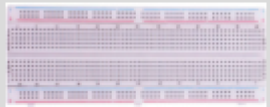
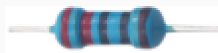
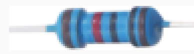
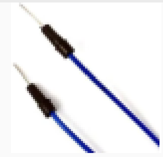
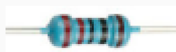
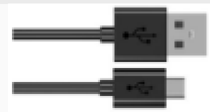


7.21 Project 19Flame Alarm

1.Introduction

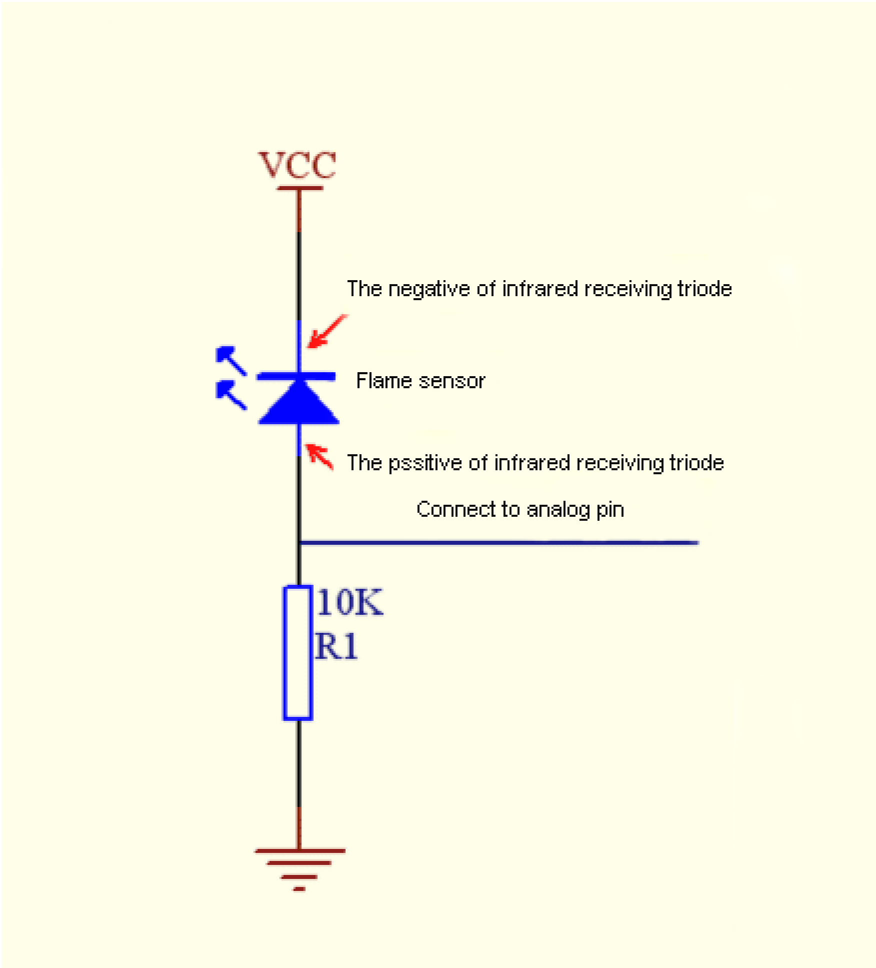
Fire is a terrible disaster and fire alarm systems are very useful in housescommercial buildings and factories. In this project, we will use ESP32 to control a flame sensor, a buzzer and a LED to simulate fire alarm devices. This is a meaningful maker activity.

2.Components

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

3.Component knowledge

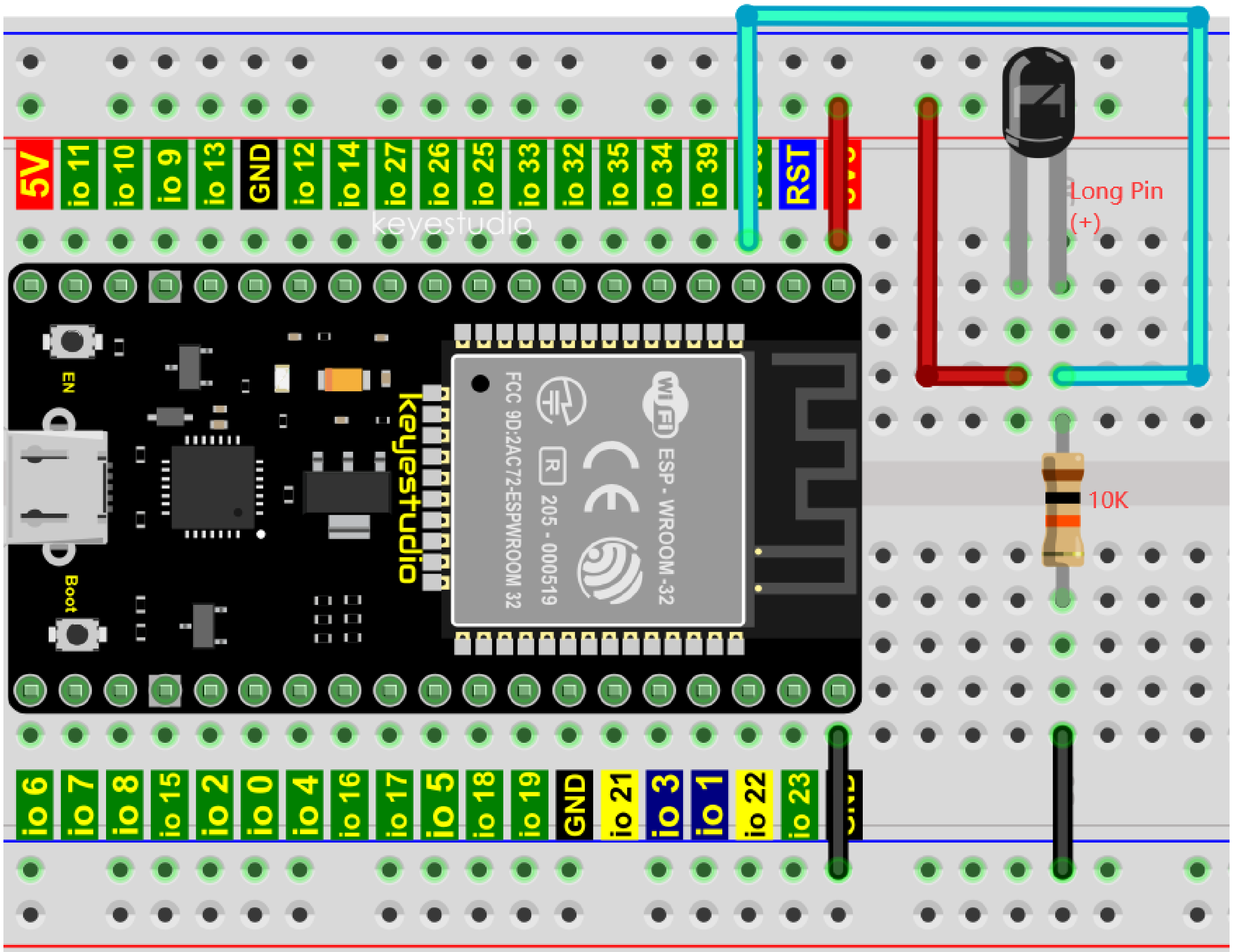
The flame emits a certain amount IR light that is invisible to the human eye, but our flame sensor can detect it and alert a microcontroller(such as ESP32) that a fire has been detected. It has a specially designed infrared receiver tube to detect the flame and then convert the flame brightness into 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



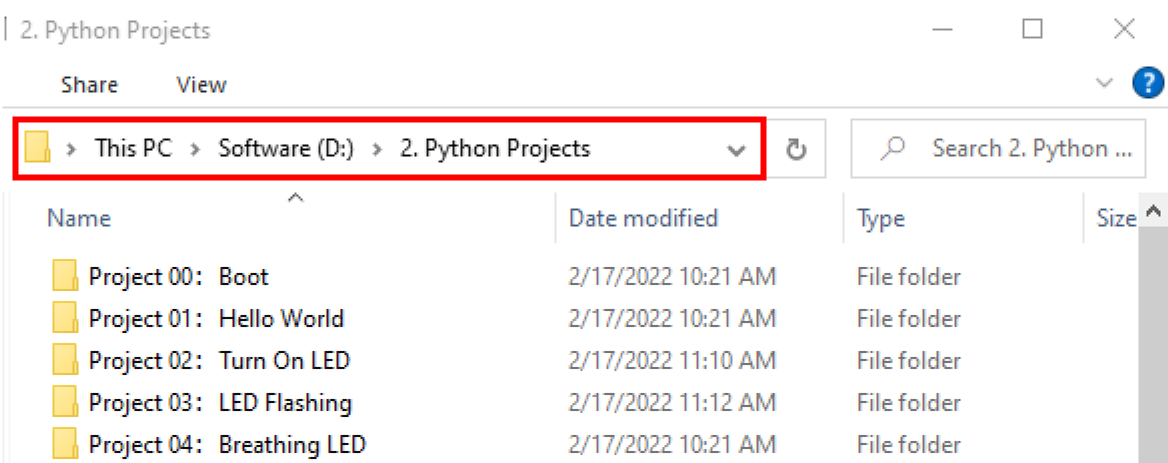
Note: Since vulnerable to radio frequency radiation and temperature changes, the flame sensor should be kept away from heat sources like radiators, heaters and air conditioners, as well as direct irradiation of sunlight, headlights and incandescent light.

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

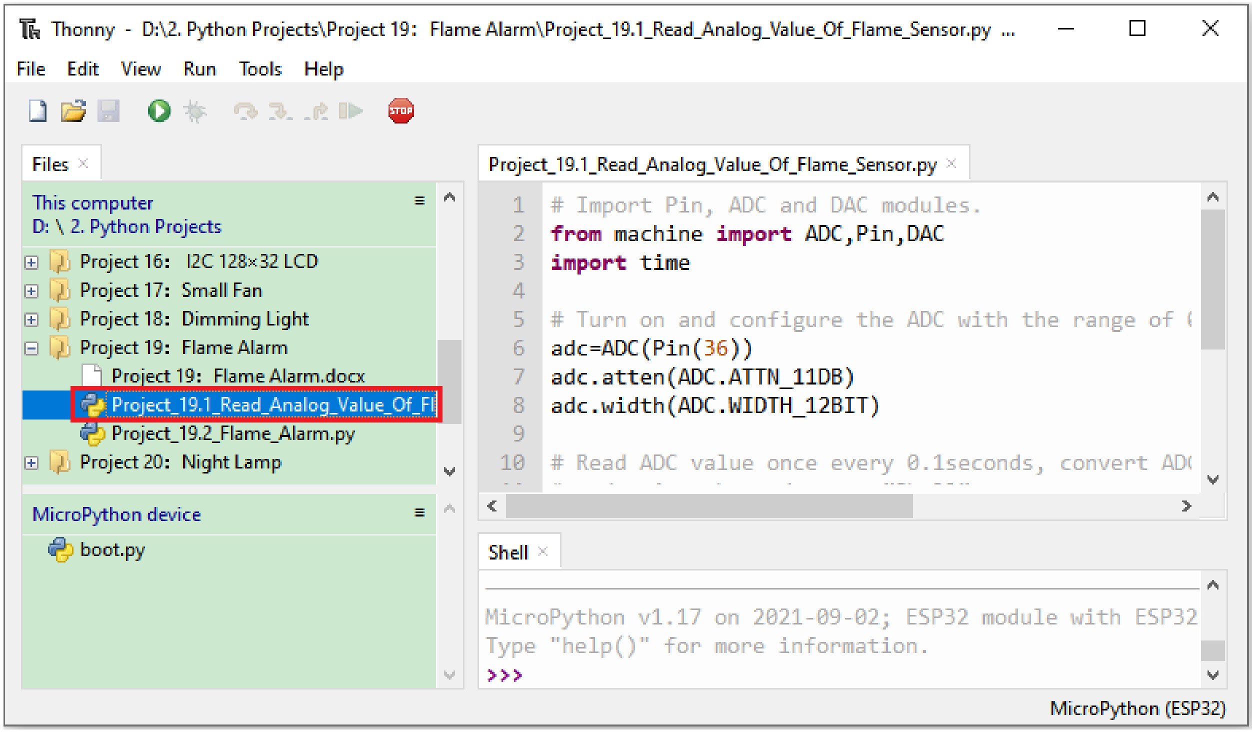
We first use a simple code to read the ADC value, DAC value and voltage value of the flame sensor and print them out. Please refer to the wiring diagram below



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

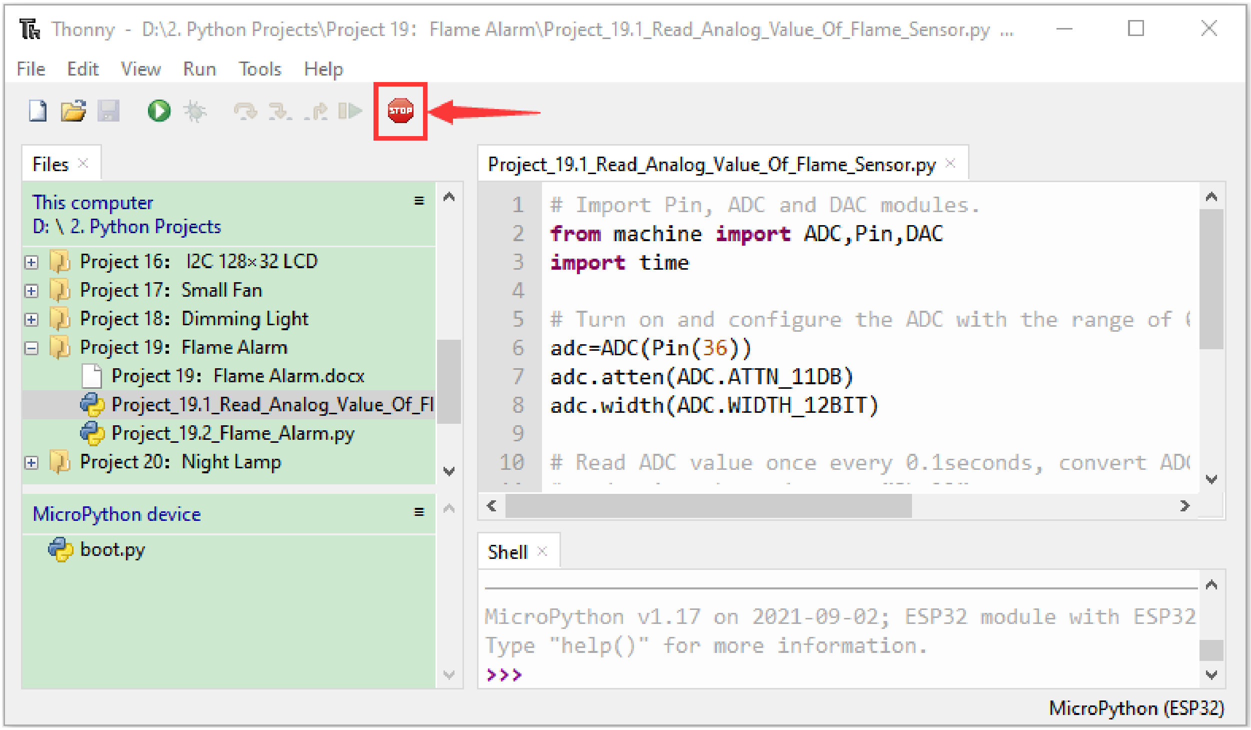


Open“Thonny”click“This computer”→“D:”→“2. Python Projects”→”Project 19: Flame Alarm”, and then double left-click “Project_19.1_Read_Analog_Value_Of_Flame_Sensor.py”.



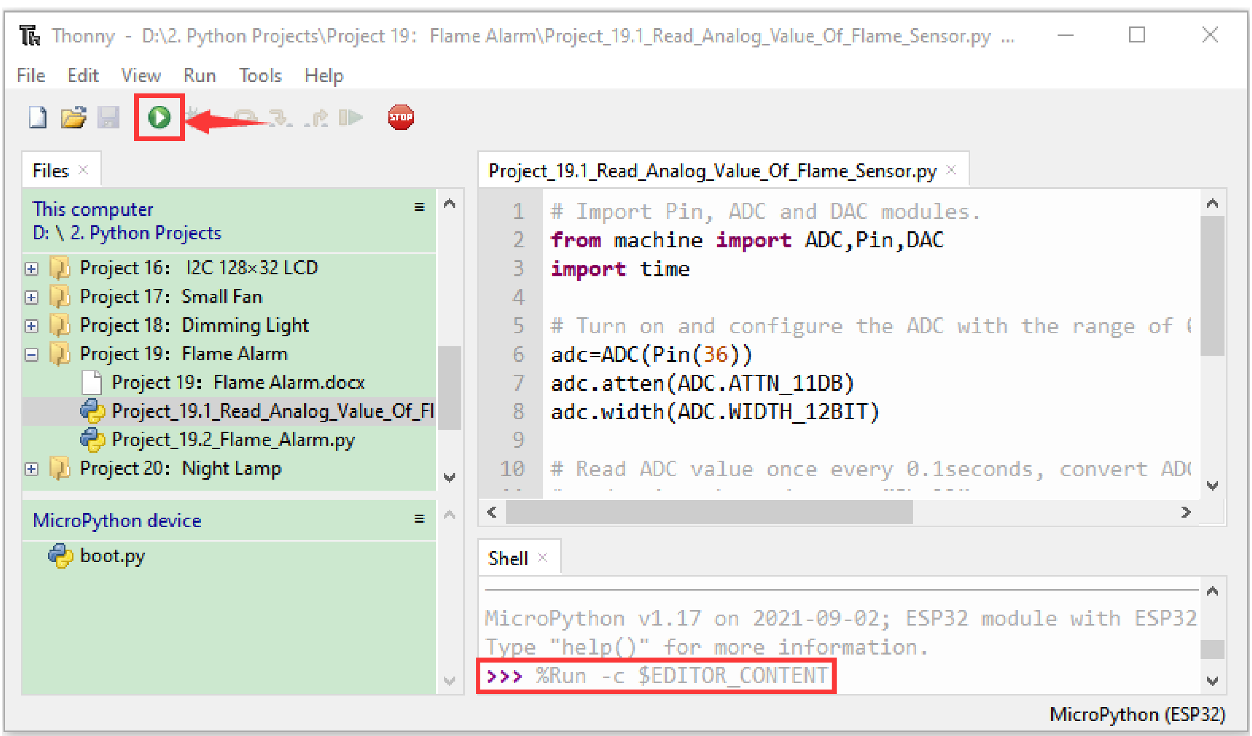

```
# Import Pin, ADC and DAC modules.
from machine import ADC,Pin,DAC
import time

# Turn on and configure the ADC with the range of 0-3.3V
adc=ADC(Pin(36))
adc.atten(ADC.ATTN_11DB)
adc.width(ADC.WIDTH_12BIT)

# Read ADC value once every 0.1seconds, convert ADC value to DAC value and output it,
# and print these data to “Shell”.
try:
    while True:
        adcVal=adc.read()
        dacVal=adcVal//16
        voltage = adcVal / 4095.0 * 3.3
        print("ADC Val:",adcVal,"DACVal:",dacVal,"Voltage:",voltage,"V")
        time.sleep(0.1)
except:
    pass
```

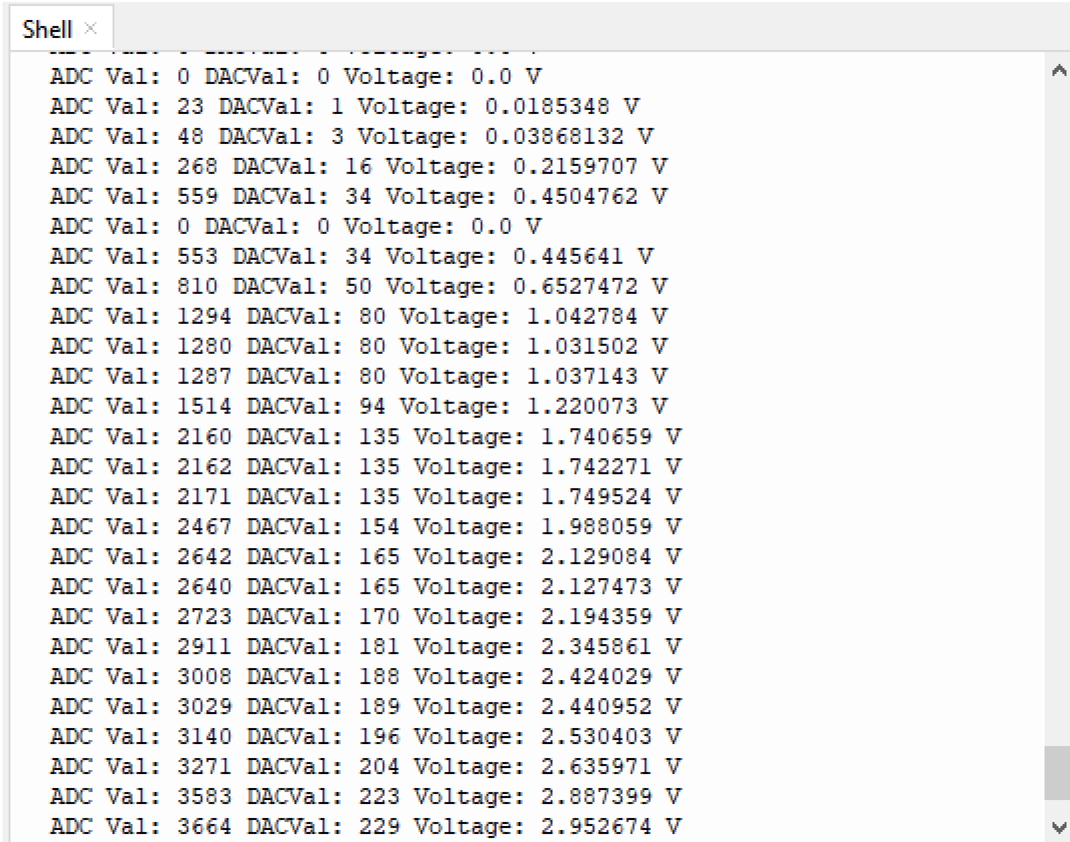
Make sure the ESP32 has been connected to the computer, click “Stop/Restart backend” .



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.

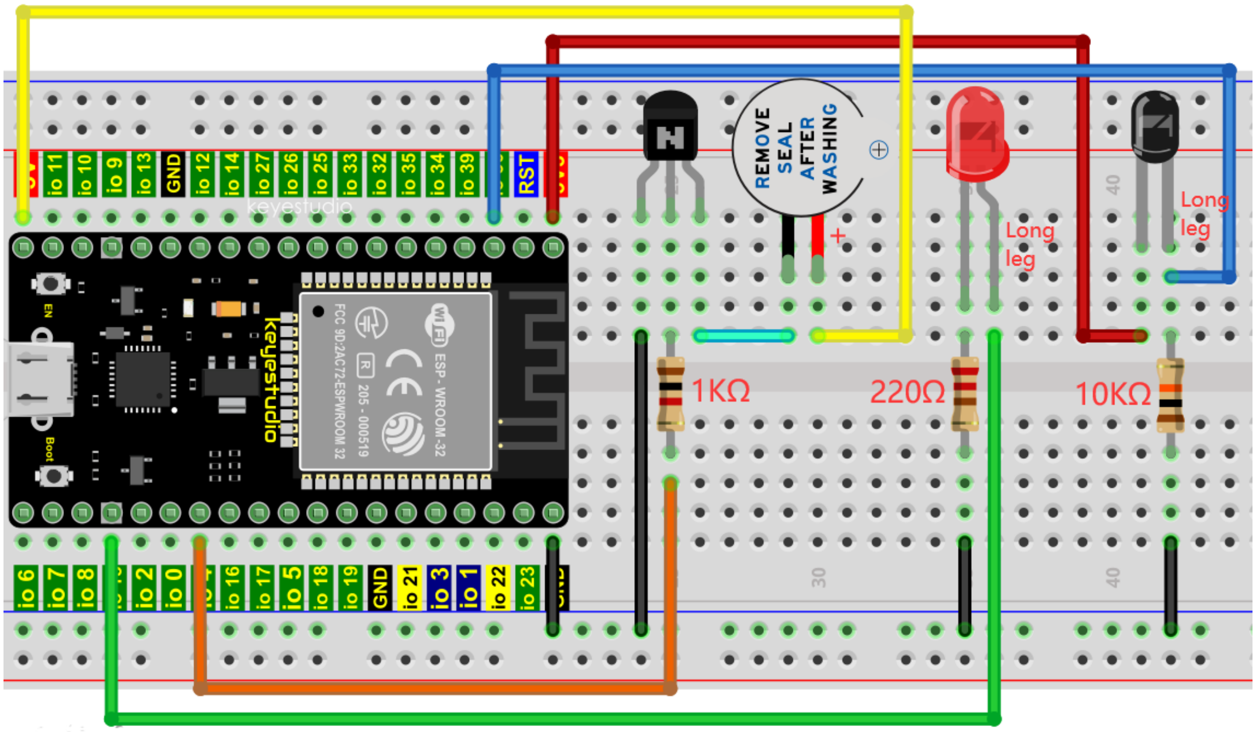




```
Shell
ADC Val: 0 DACVal: 0 Voltage: 0.0 V
ADC Val: 23 DACVal: 1 Voltage: 0.0185348 V
ADC Val: 48 DACVal: 3 Voltage: 0.03868132 V
ADC Val: 268 DACVal: 16 Voltage: 0.2159707 V
ADC Val: 559 DACVal: 34 Voltage: 0.4504762 V
ADC Val: 0 DACVal: 0 Voltage: 0.0 V
ADC Val: 553 DACVal: 34 Voltage: 0.445641 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
```

5.Wiring diagram of the flame alarm

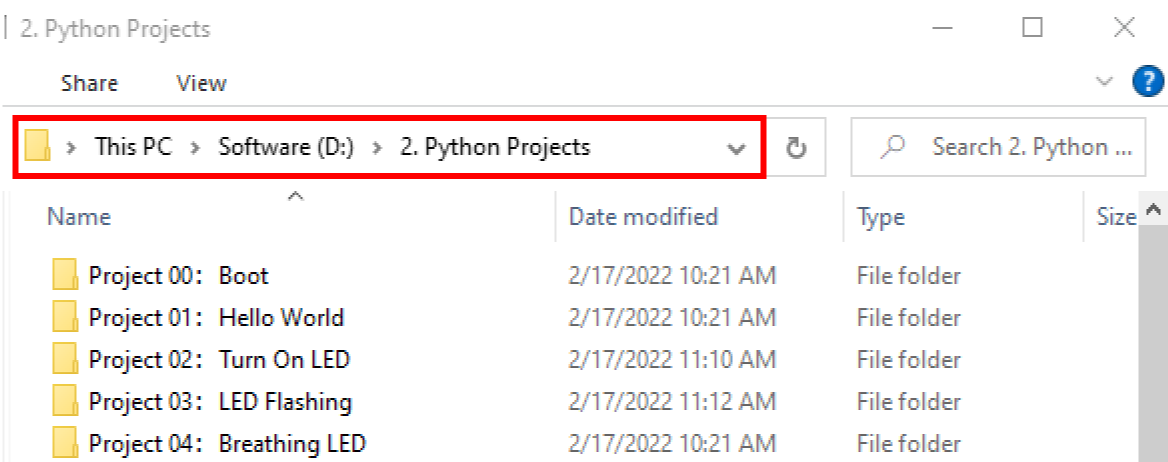
Next, we will use a flame sensor, a buzzer, and a LED to make an interesting project, that is flame alarm. When flame is detected, the LED flashes and the buzzer alarms.



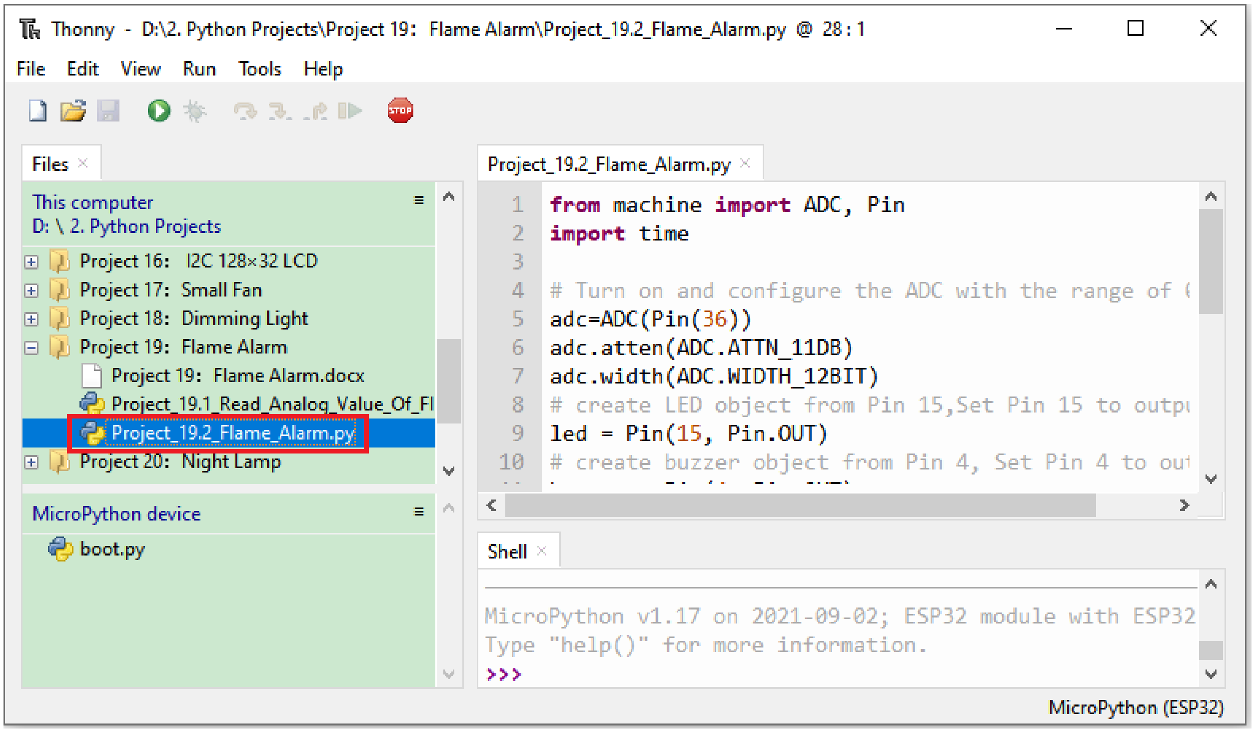

```
value = get_value()
if value >500:
    buzzer.value(1)
```

6.Project code:Note 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 yet, please click on the link to download it:Download Python Codes



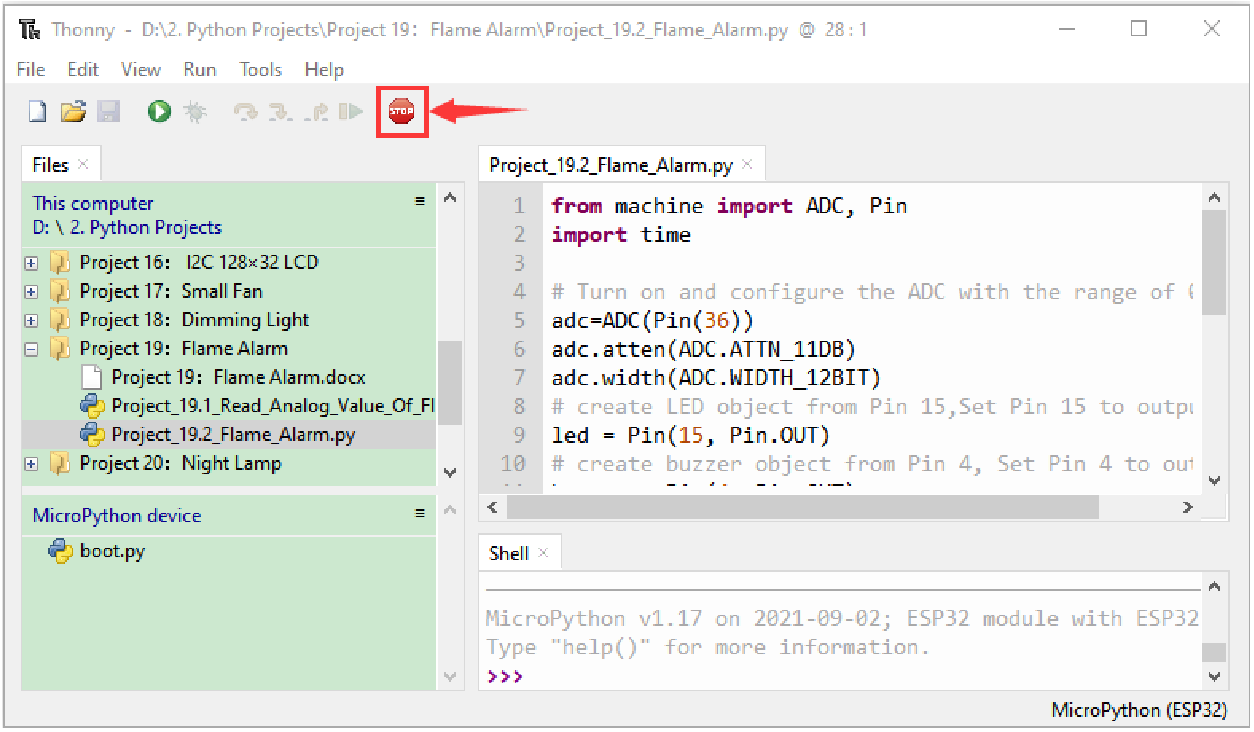
Open“Thonny”click“This computer”→“D:”→“2. Python Projects”→“Project 19: Flame Alarm”, and then double left-click“Project 19Flame Alarm”.



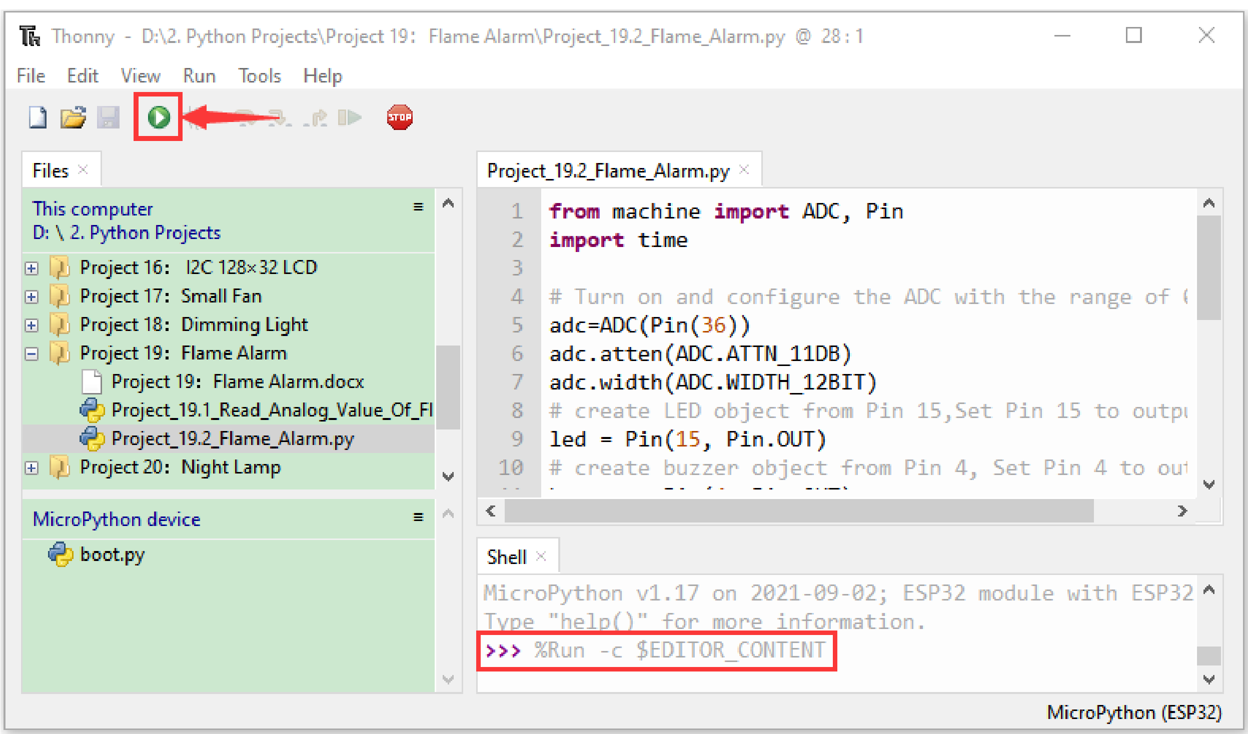

```python
from machine import ADC, Pin
import time

# 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)
# 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.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.

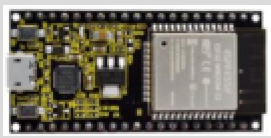
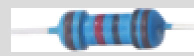
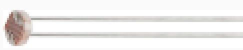
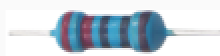
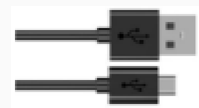


7.22 Project 20Night Lamp

1.Introduction

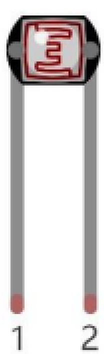
Sensors or components are ubiquitous in our daily life. For example, some public street lamps will automatically turn on at night and turn off during the day. Why? In fact, this make use of a photosensitive element that senses the intensity of external ambient light. When the outdoor brightness decreases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

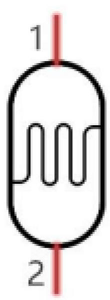
2.Components

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

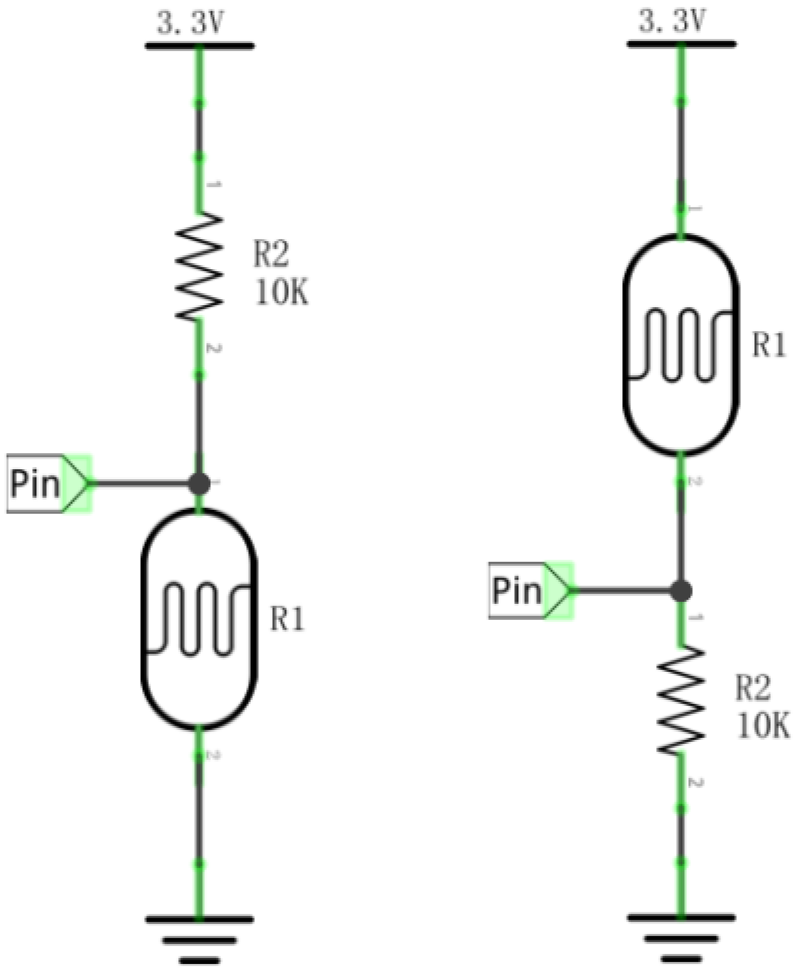
3.Component knowledge

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





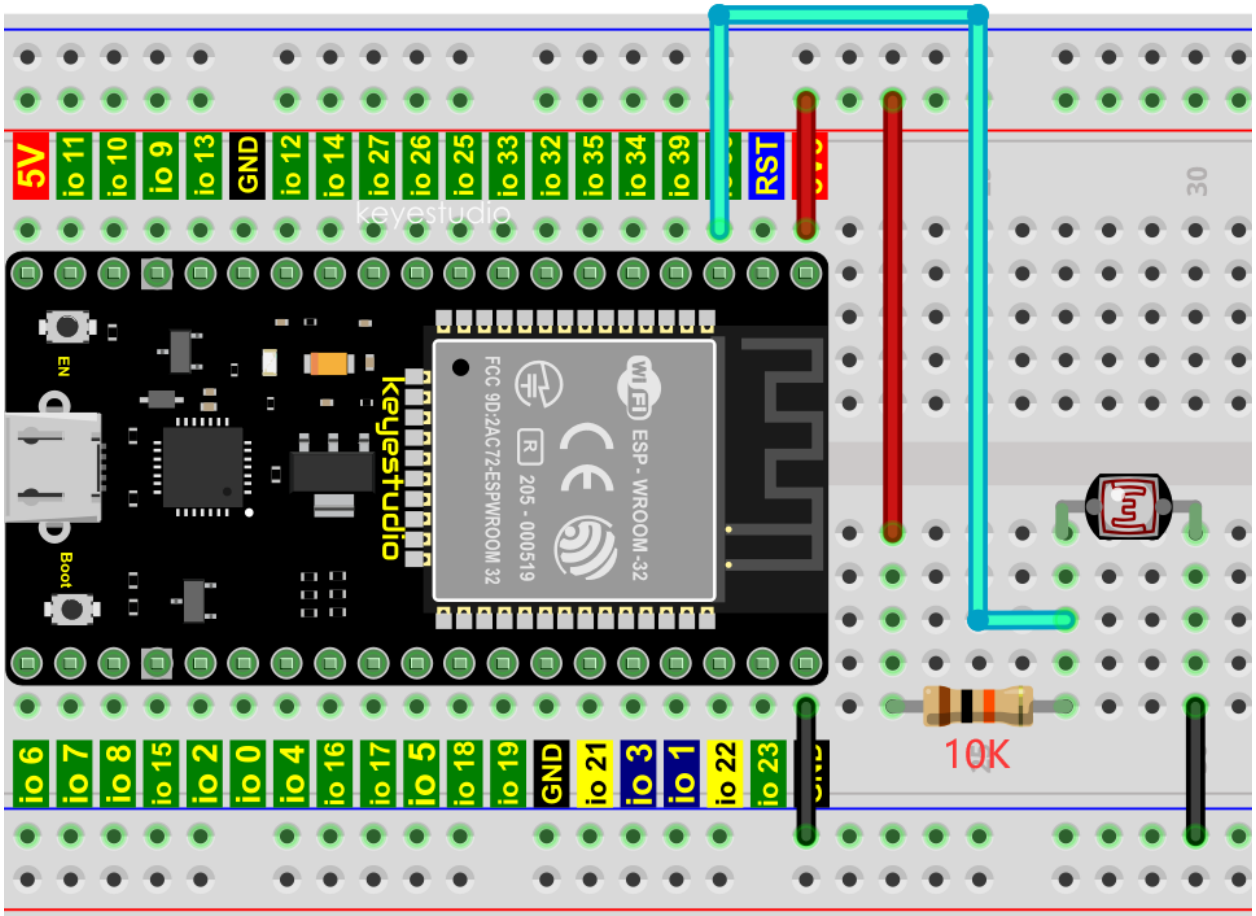
The following circuit is used to detect changes in resistance values of photoresistors



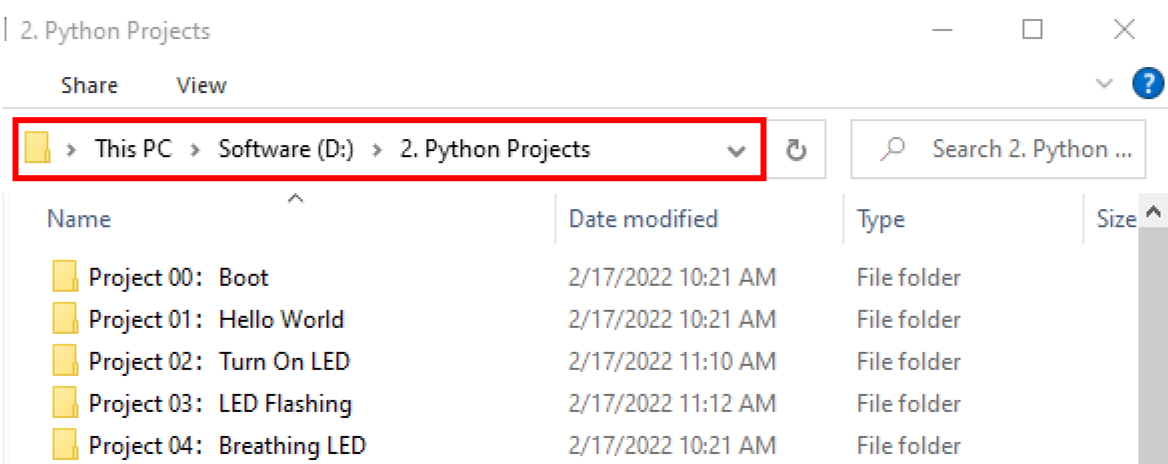
In the circuit above, when the resistance of the photoresistor changes due to the change of light intensity, the voltage between the photoresistor and resistance R2 will also change. Thus, the intensity of light can be obtained by measuring this voltage.

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

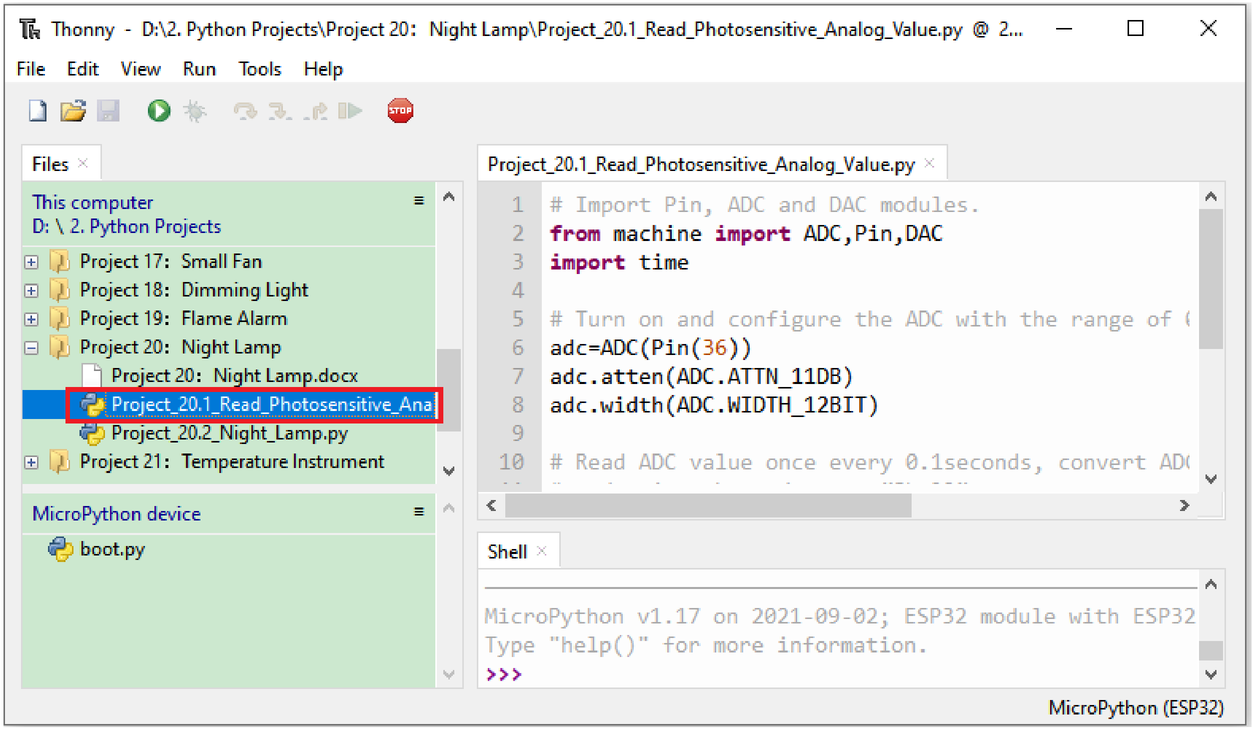
We first use a simple code to read the ADC value, DAC value and voltage value of the photoresistor and print them out. Please refer to the following wiring diagram



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

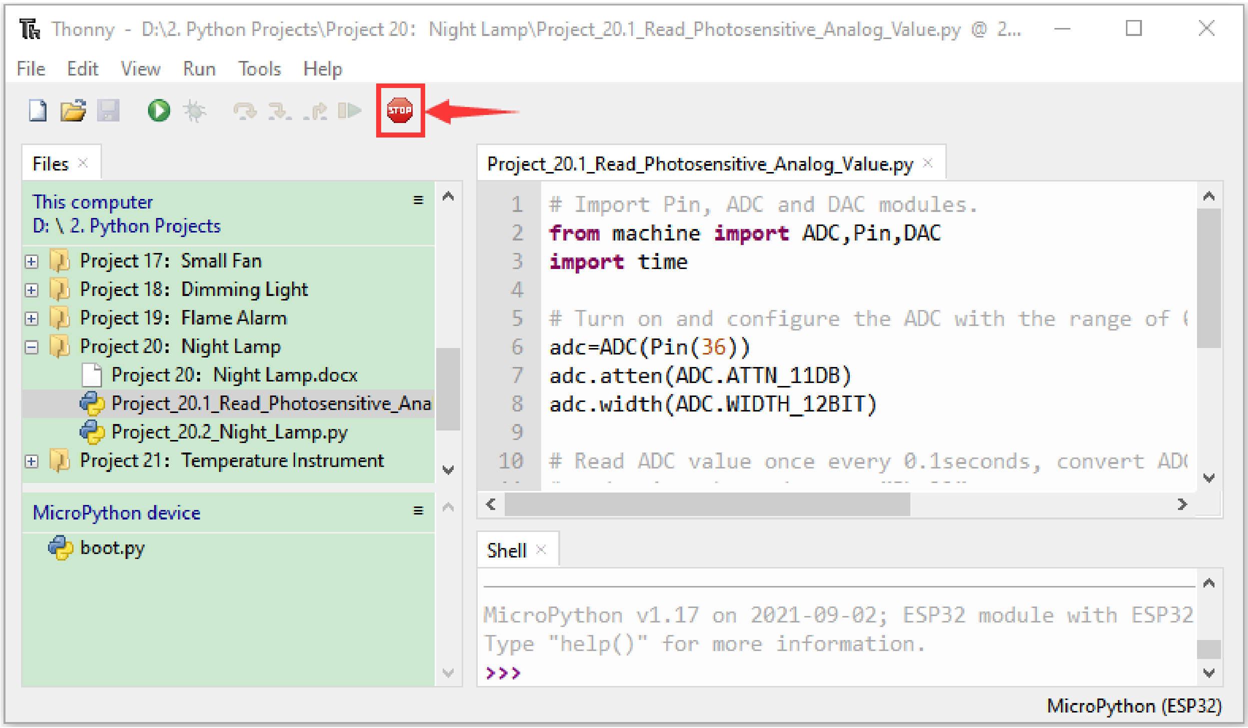


Open“Thonny”click“This computer”→“D:”→“2. Python Projects”→”Project 20: Night Lamp”and then double left-click “Project_20.1_Read_Photosensitive_Analog_Value.py””.

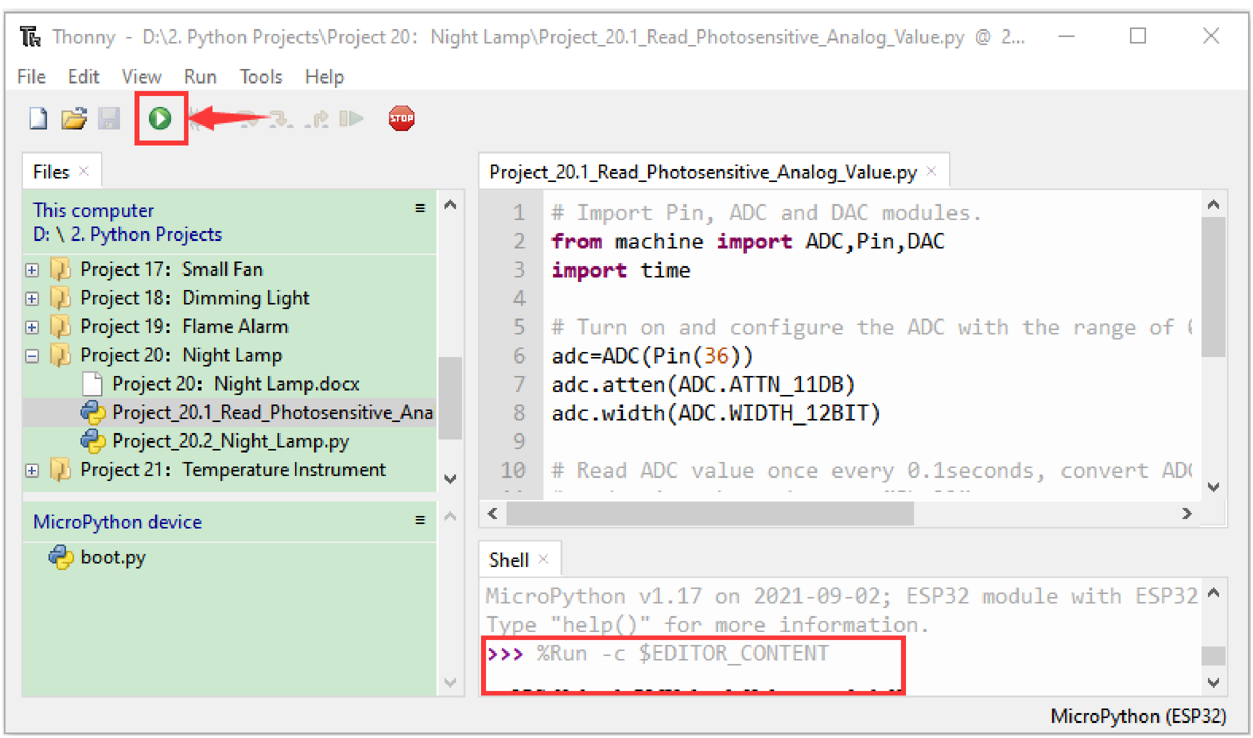



```
# Import Pin, ADC and DAC modules.
from machine import ADC,Pin,DAC
import time

# 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” .



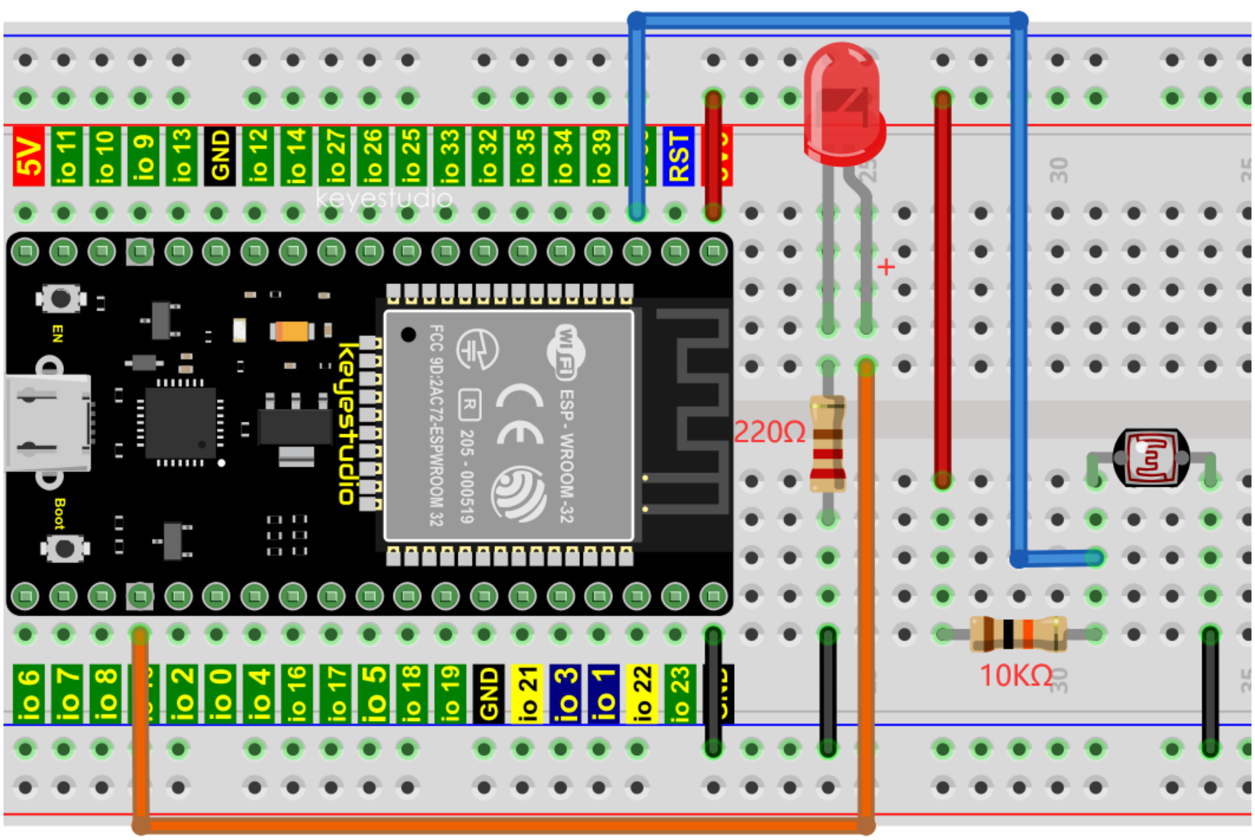
Click “Run current script”, the code starts to be executed and you’ll see that the “Shell” window of Thonny IDE will print the ADC valueDAC value and voltage value of the photoresistor. When the light intensity around the photoresistor is gradually reduced, the ADC valueDAC value and voltage value will 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
```

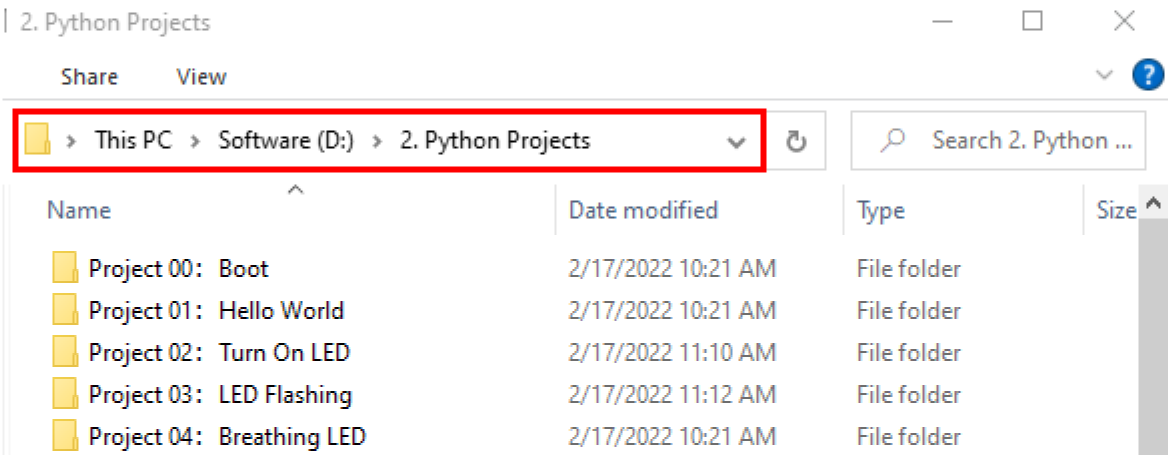
5.Wiring diagram of the light-controlled lamp

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.

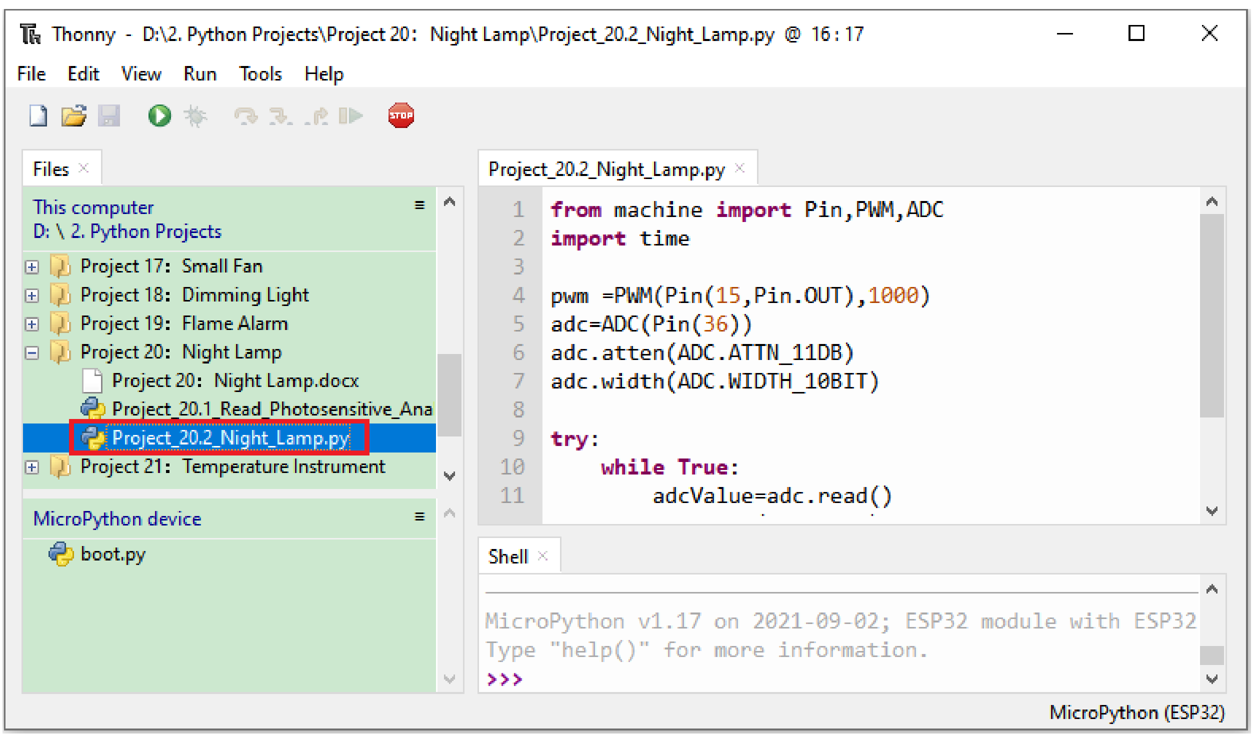


6.Project code

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



Open“Thonny”click“This computer”→“D:”→“2. Python Projects”→”Project 20: Night Lamp”and then double left-click “Project_20.2_Night_Lamp.py”.




```python
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)
```

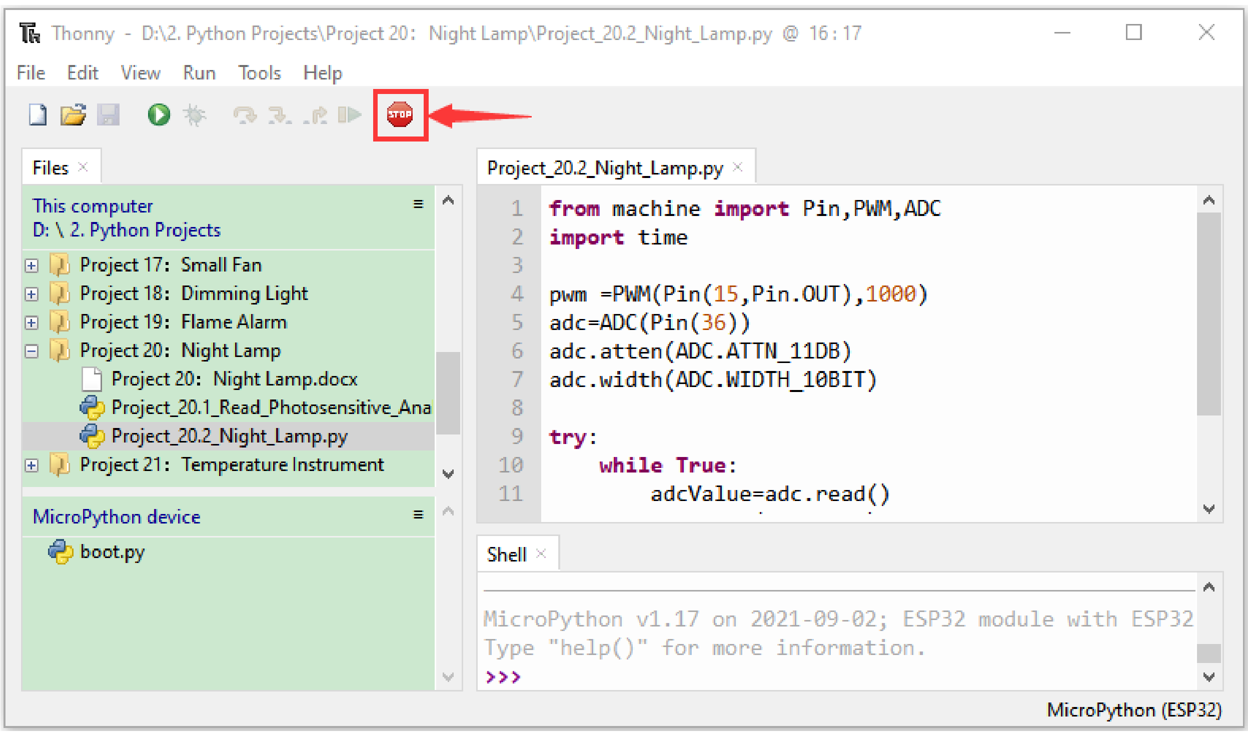
(continues on next page)

(continued from previous page)

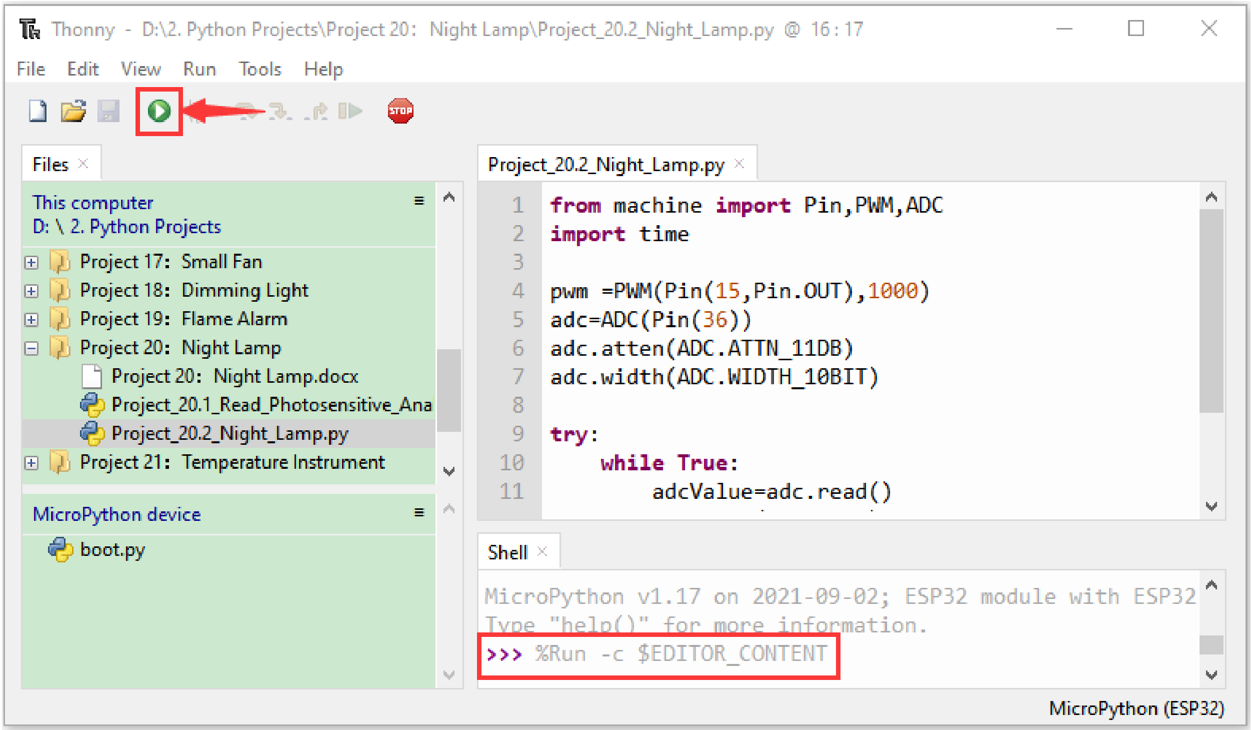
```
try:
    while True:
        adcValue=adc.read()
        pwm.duty(adcValue)
        print(adc.read())
        time.sleep_ms(100)
except:
    pwm.deinit()
```

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 contrary, the LED will be dim. Press"Ctrl+C"or click "Stop/Restart backend"to exit the program.

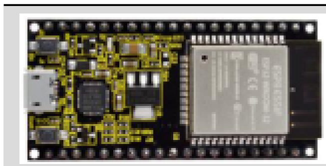
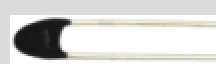
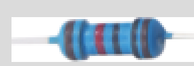
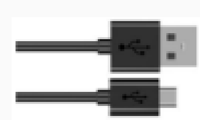


7.23 Project 21Temperature Instrument

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;

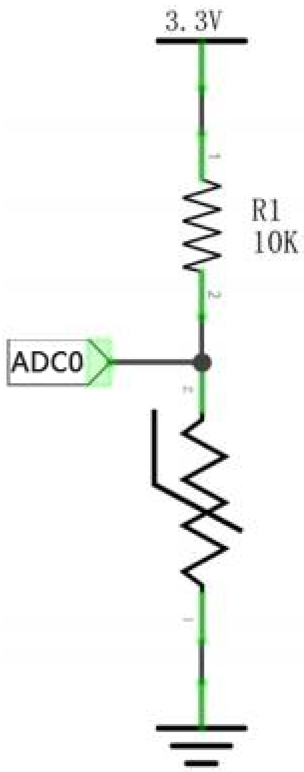
EXP[n] is nth power of e;

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



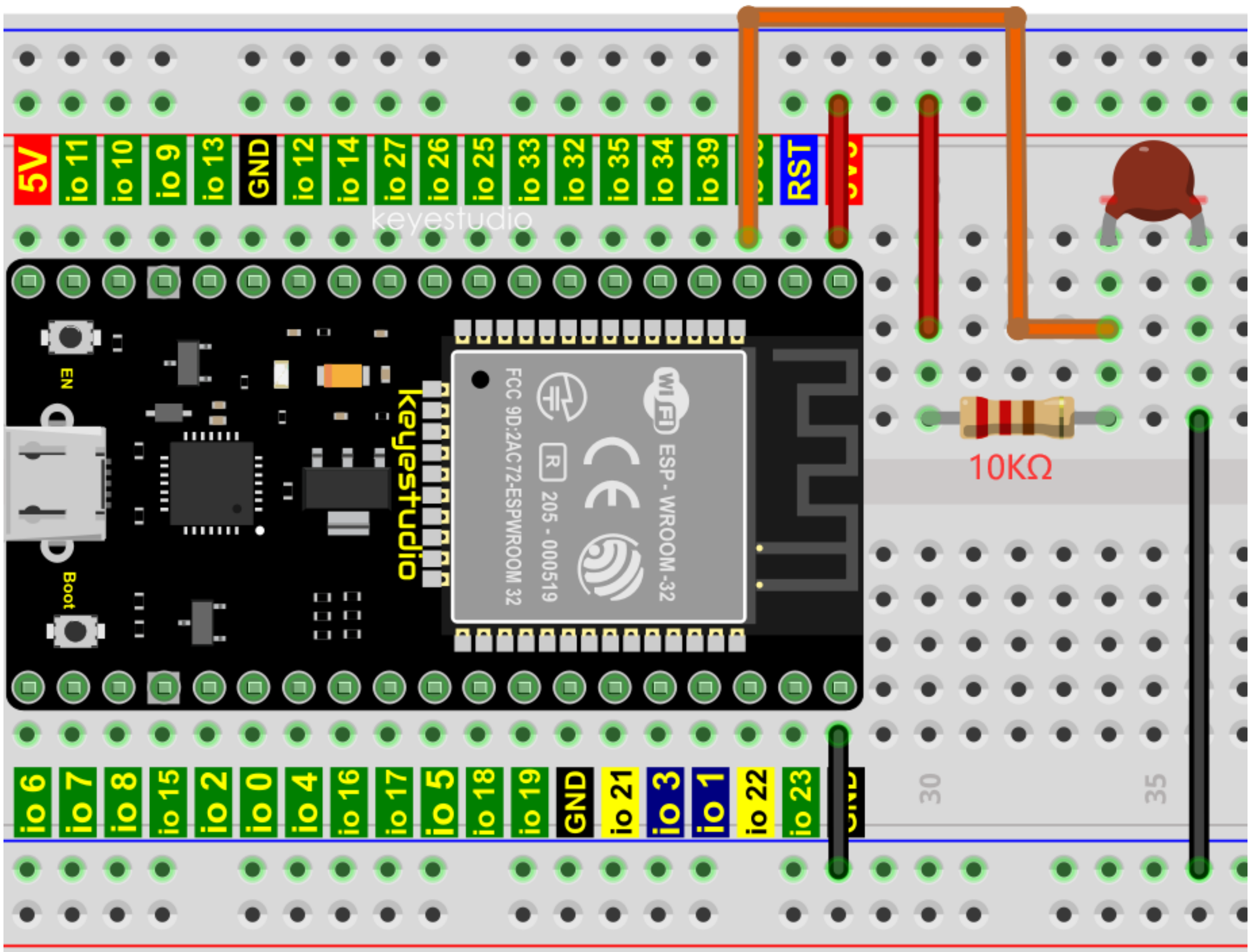
We can use the value measured by the ADC converter to obtain the resistance value of Thermistor, and then we can use the formula to obtain the temperature value.

Therefore, the temperature formula can be derived as:

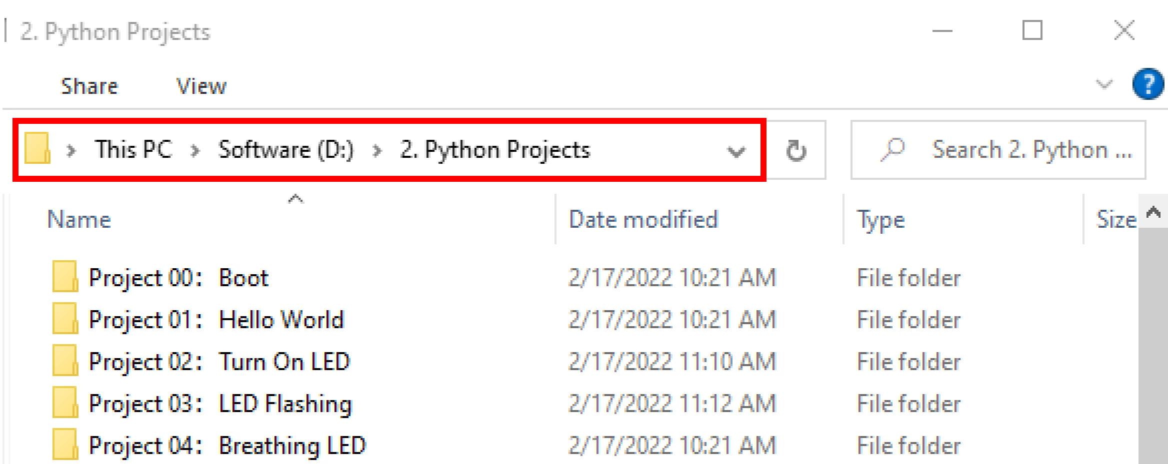
$$T2 = 1/\left(\frac{1}{T1} + \ln\left(\frac{Rt}{R}\right)/B\right)$$

4.Read the value of the Thermistor

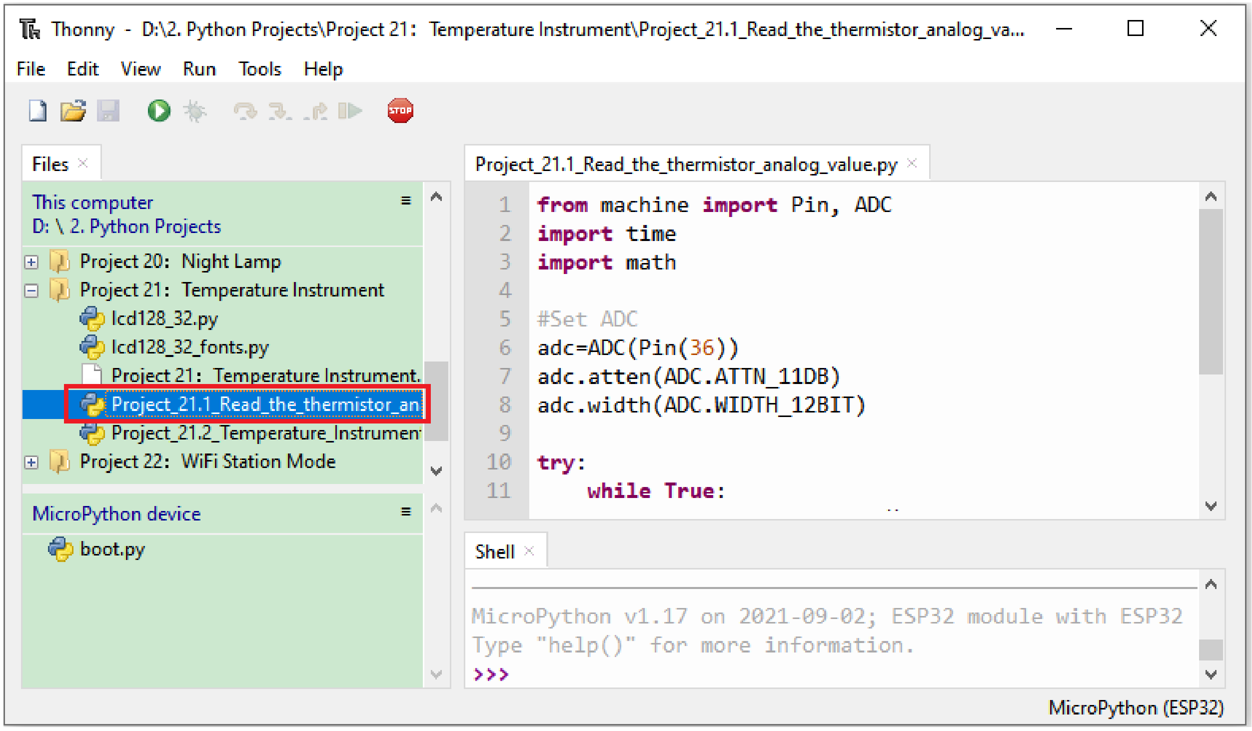
First we will learn the thermistor to read the current ADC value, voltage value and temperature value and print them out. Please connect the wires according to the wiring diagram below



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

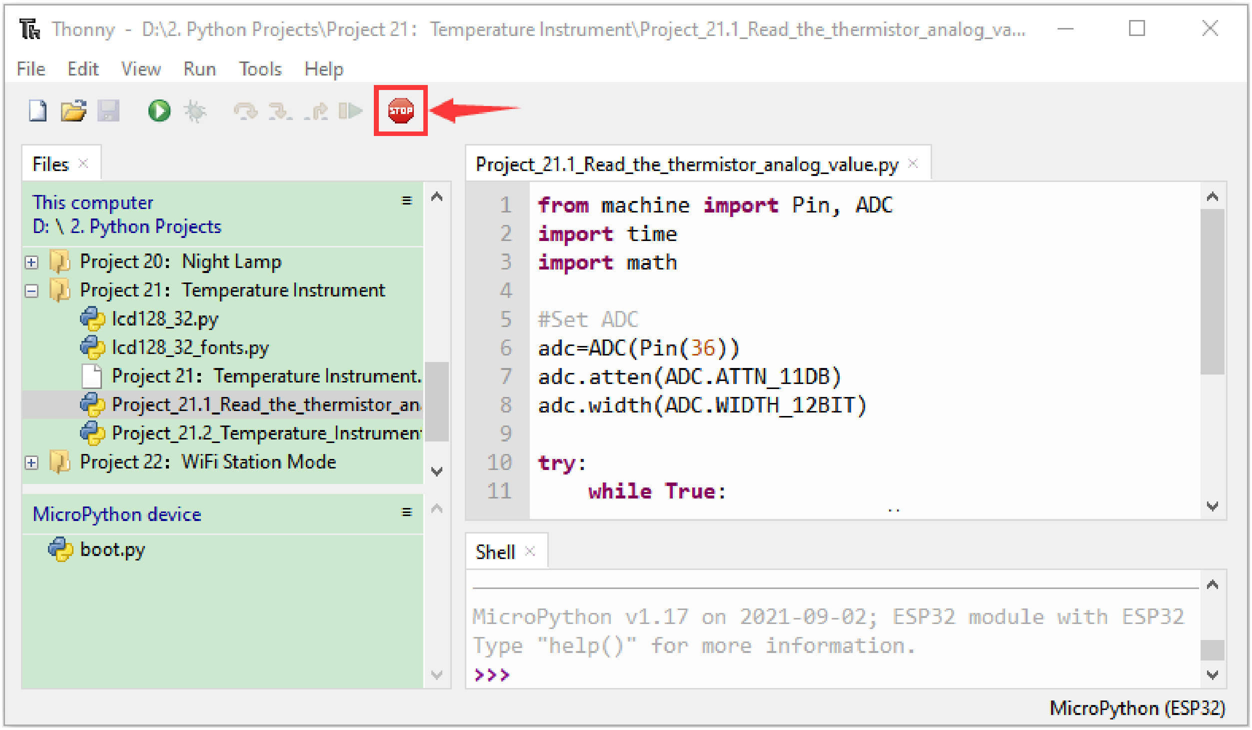


Open"Thonny"click"This computer"→"D:"→"2. Python Projects"→"Project 21: Temperature Instrument, and then double left-click "Project_21.1_Read_the_thermistor_analog_value.py".



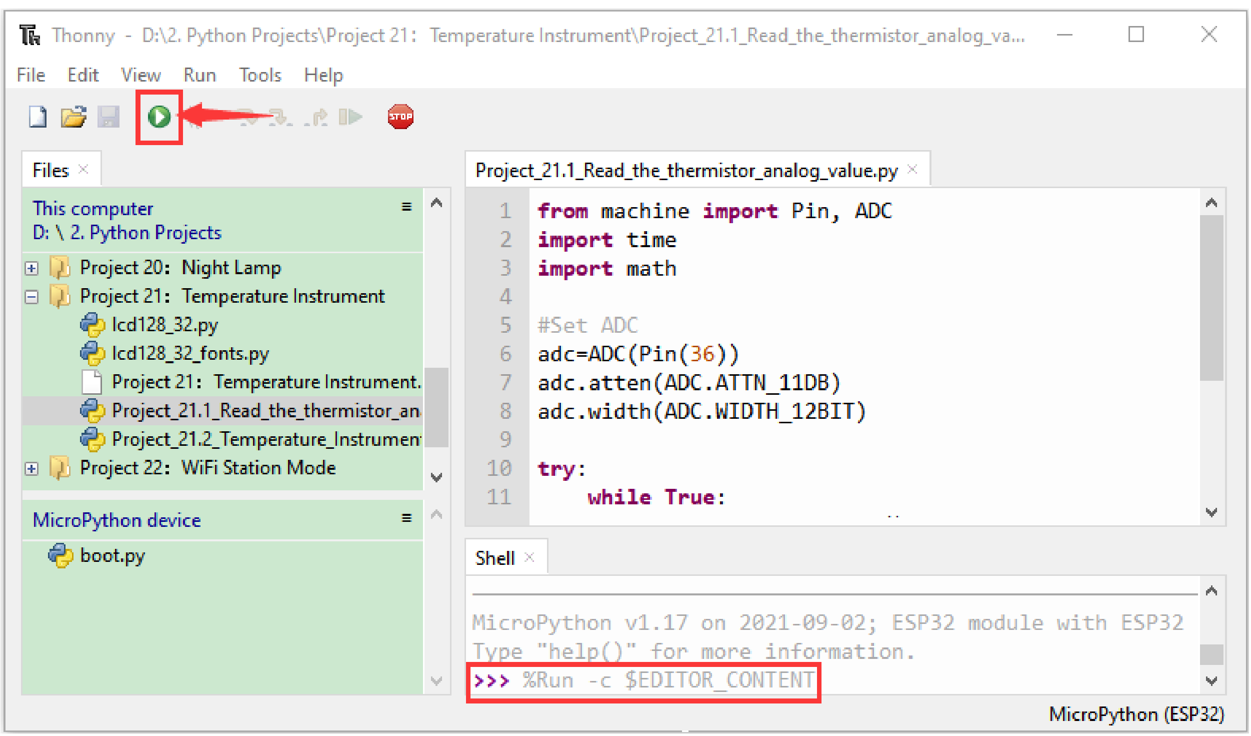

```
from machine import Pin, ADC
import time
import math

#Set ADC
adc=ADC(Pin(36))
adc.atten(ADC.ATTN_11DB)
adc.width(ADC.WIDTH_12BIT)

try:
    while True:
        adcValue = adc.read()
        voltage = adcValue / 4095 * 3.3
        Rt = 10 * voltage / (3.3-voltage)
        tempK = (1 / (1 / (273.15+25) + (math.log(Rt/10)) / 3950))
        tempC = (tempK - 273.15)
        print("ADC value:",adcValue,"  Voltage:",voltage,"V","  Temperature: ",tempC,"C
↪");
        time.sleep(1)
except:
    pass
```

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



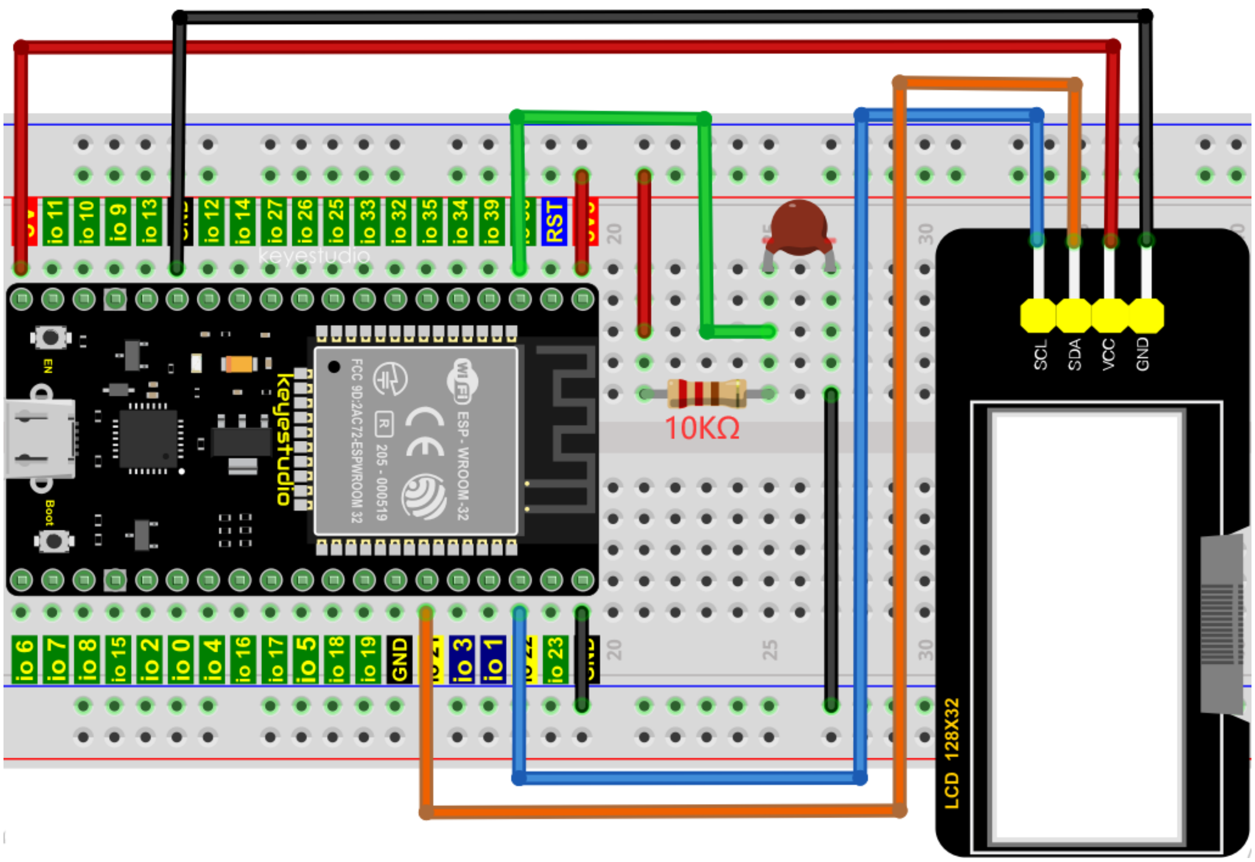
Click “Run current script”, the code starts to be executed and you’ll see that the “Shell” window of Thonny IDE will continuously display the thermistor’s current ADC valuevoltage value and temperature value. Try pinching the thermistor with your index finger and thumb (don’t touch wires) for a while,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

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



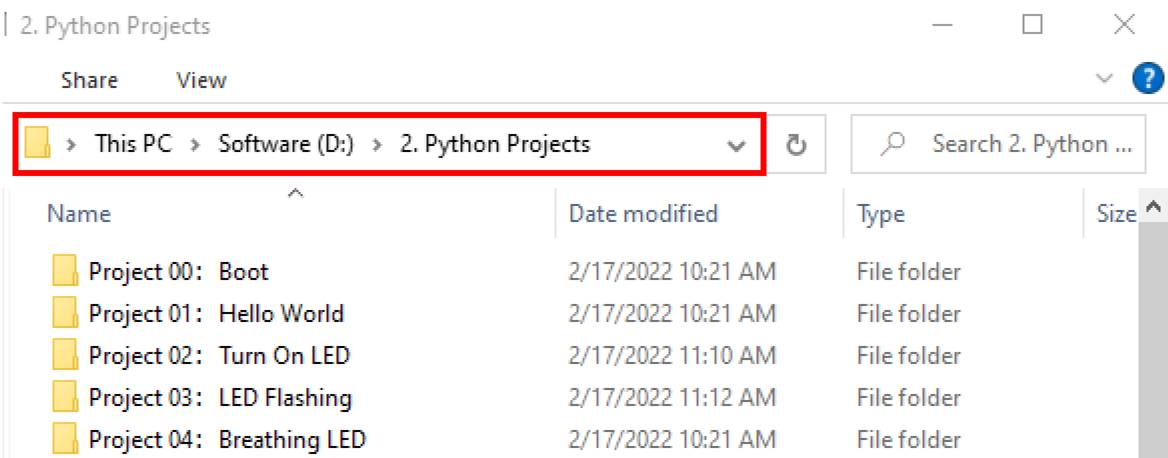
```
Shell ×
>>> %Run -c $EDITOR_CONTENT

  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
```

5.Wiring diagram of the temperature instrument

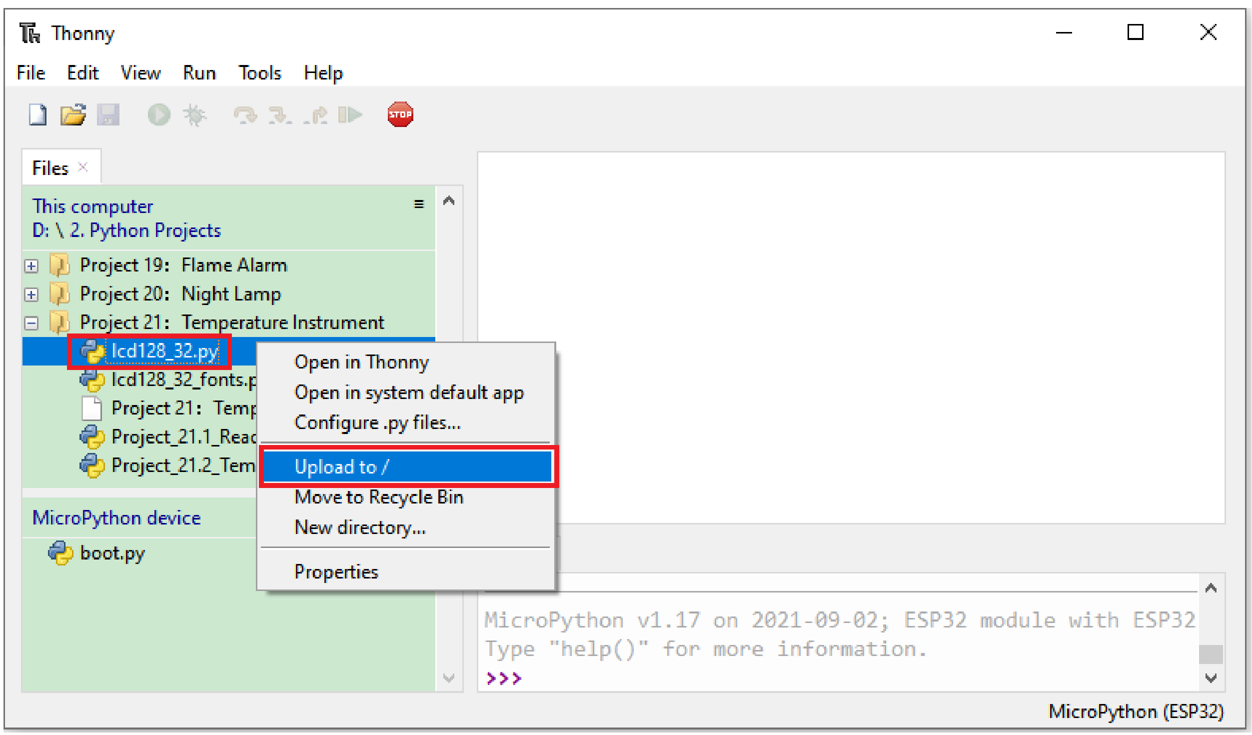


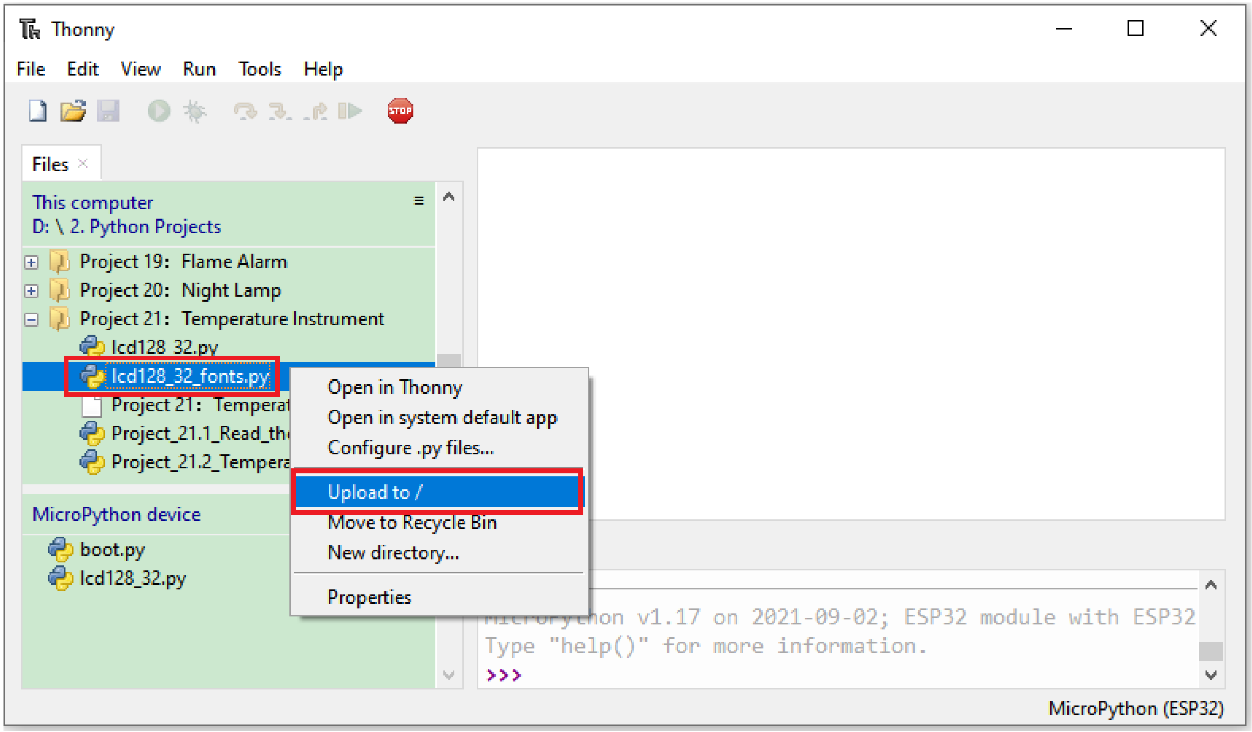
6.Project code

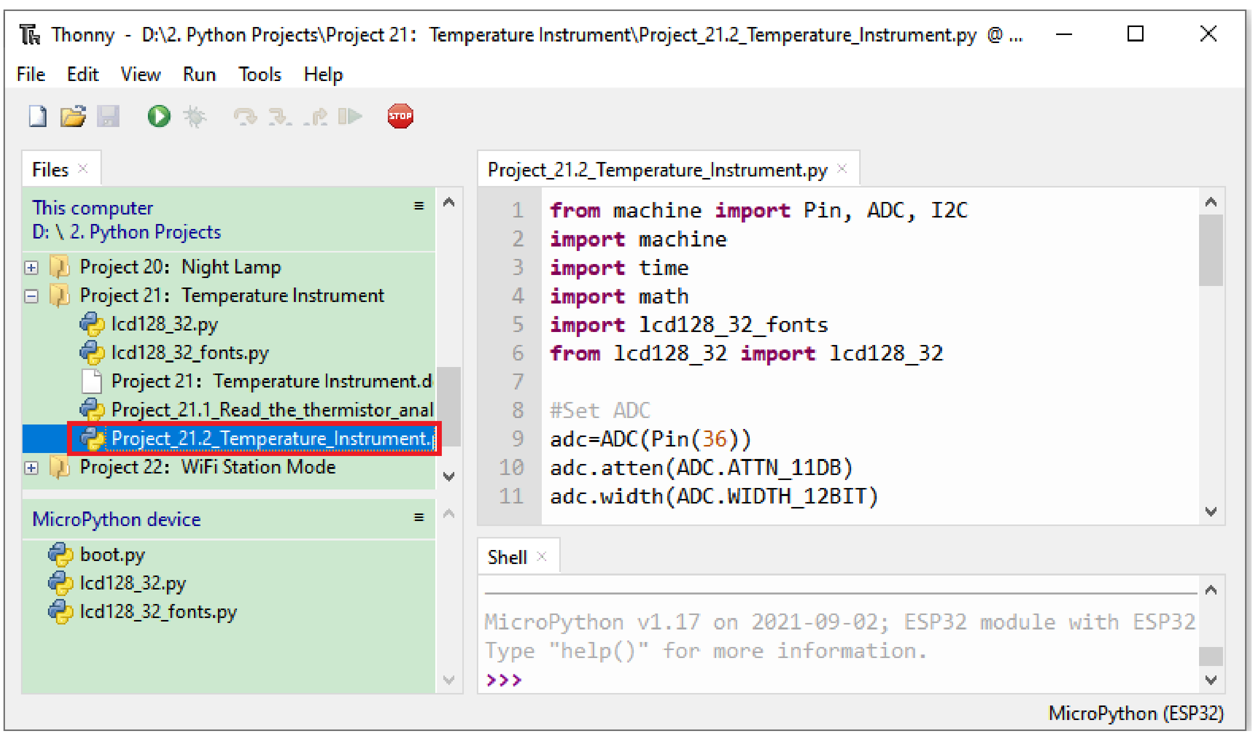
Codes used in this tutorial are saved in“2. Python Projects”. If you haven’t downloaded the code file yet, please click on the link to download it:Download Python Codes



Open“Thonny”click“This computer”→“D:”→“2. Python Projects”→“Project 21: Temperature Instrument”. Select“lcd128_32.py”and “lcd128_32_fonts.py”right-click your mouse to select“Upload to/”wait for“lcd128_32.py”an “lcd128_32_fonts.py”to be uploaded to ESP32and double left-click“Project_21.2_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
```

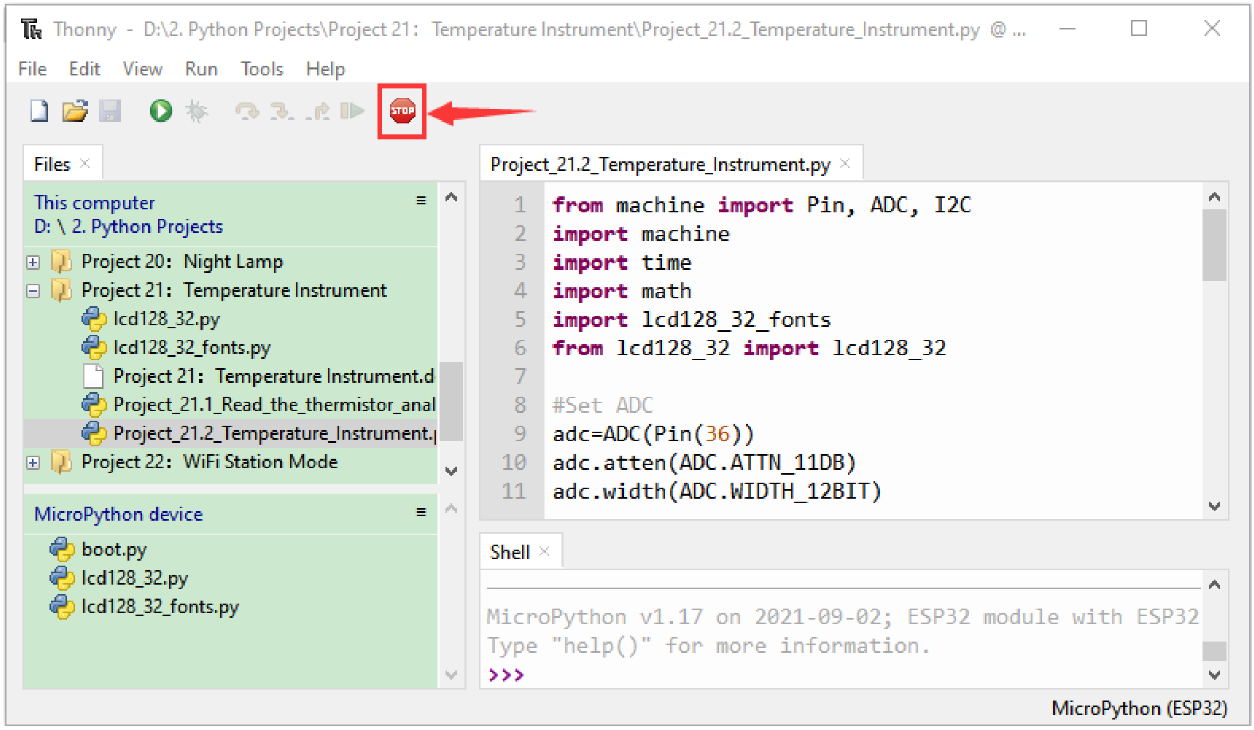
(continues on next page)

(continued from previous page)

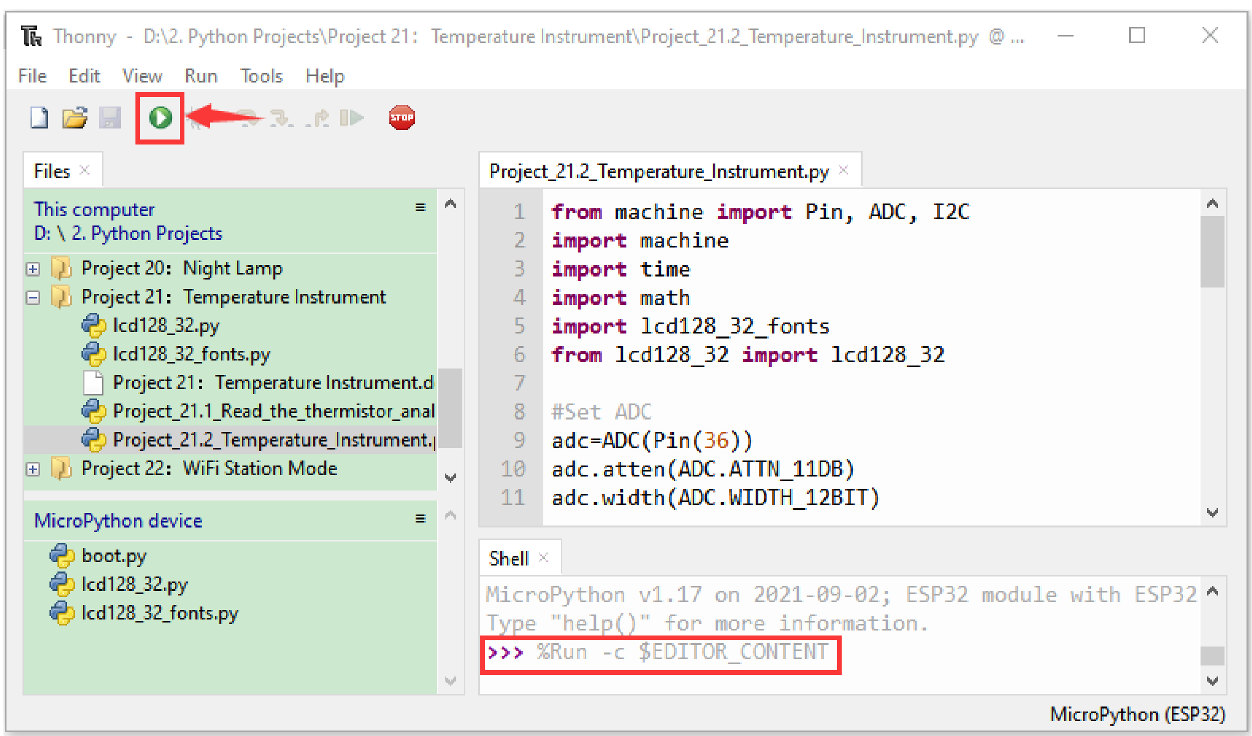
```
#Set ADC
adc=ADC(Pin(36))
adc.atten(ADC.ATTN_11DB)
adc.width(ADC.WIDTH_12BIT)

#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.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.

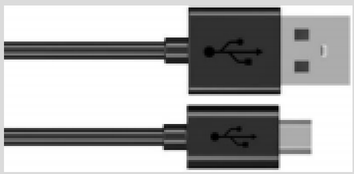
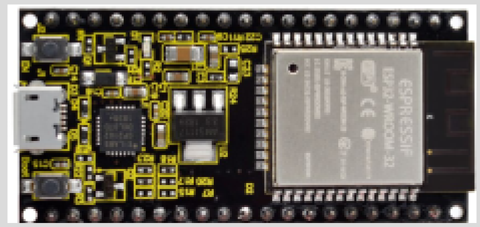


7.24 Project 22WiFi Station Mode

1.Introduction

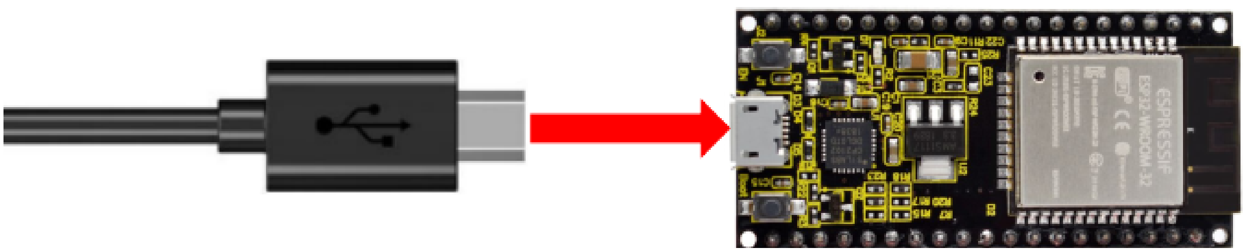
ESP32 has three different WiFi operating modes : Station modeAP mode and AP+Station mode. All WiFi programming projects must be configured with WiFi operating mode before using WiFi, otherwise WiFi cannot be used. In this project, we will learn about ESP32's WiFi Station mode.

2.Components

	
USB Cable*1	ESP22*1

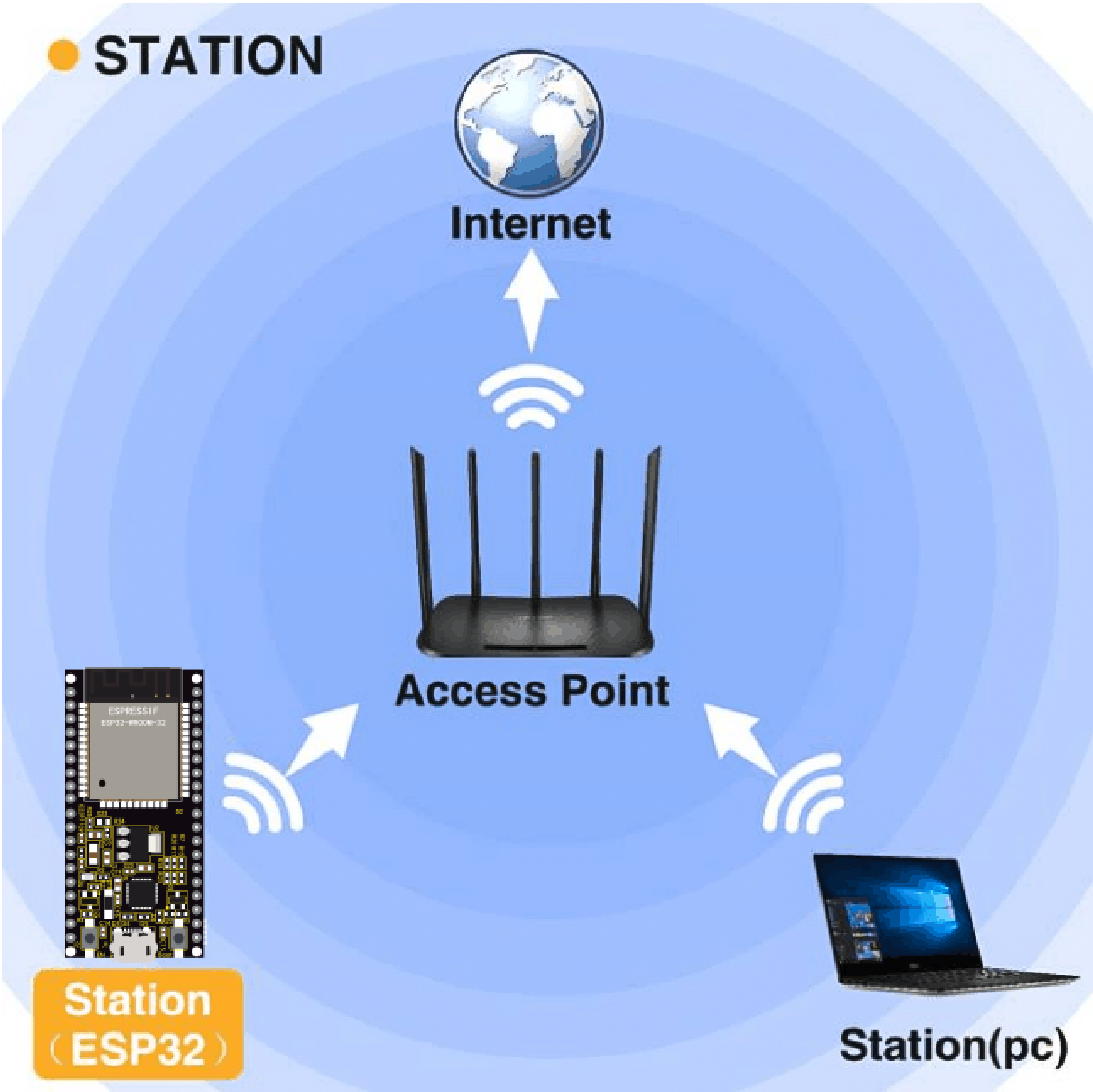
3.Project wiring

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



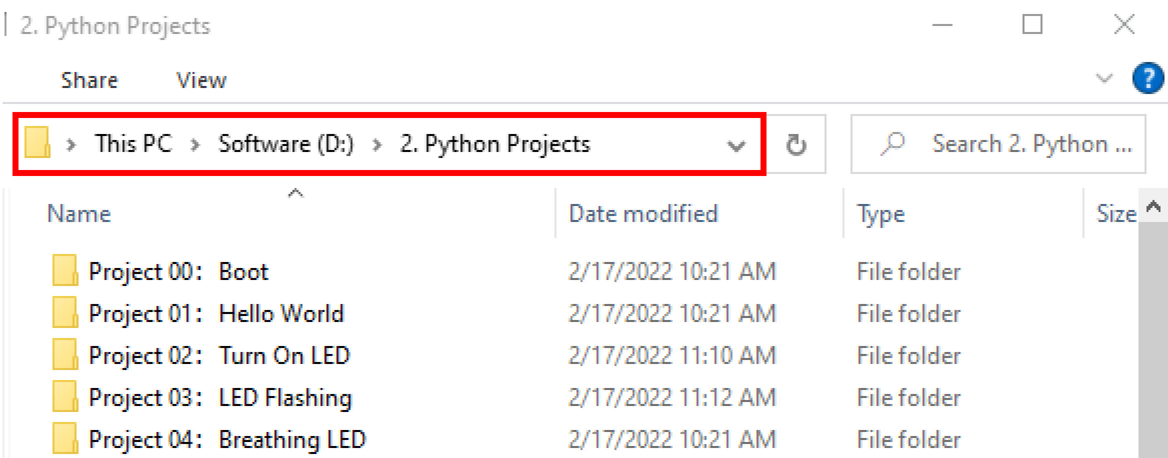
4.Component knowledge

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

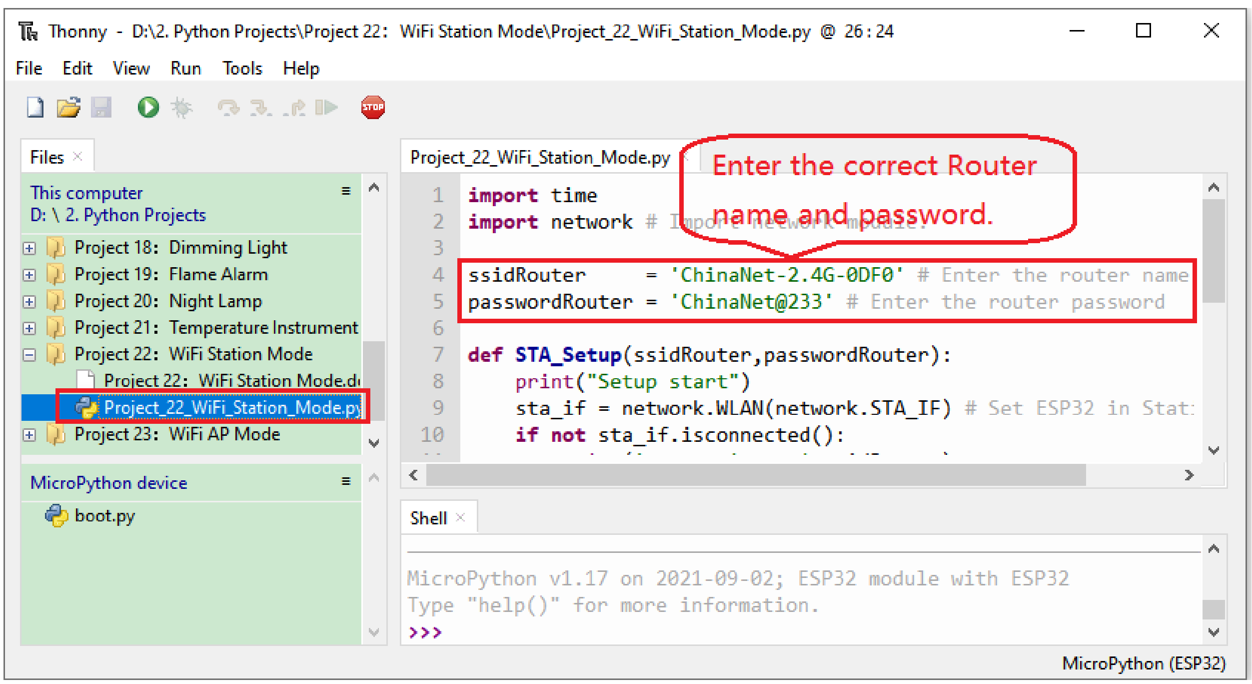


5.Project code

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



Open“Thonny”click“This computer”→“D:”→“2. Python Projects”→“Project 22WiFi Station Mode”and double left-click “Project_22_WiFi_Station_Mode.py”.




```python
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.
    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)
```

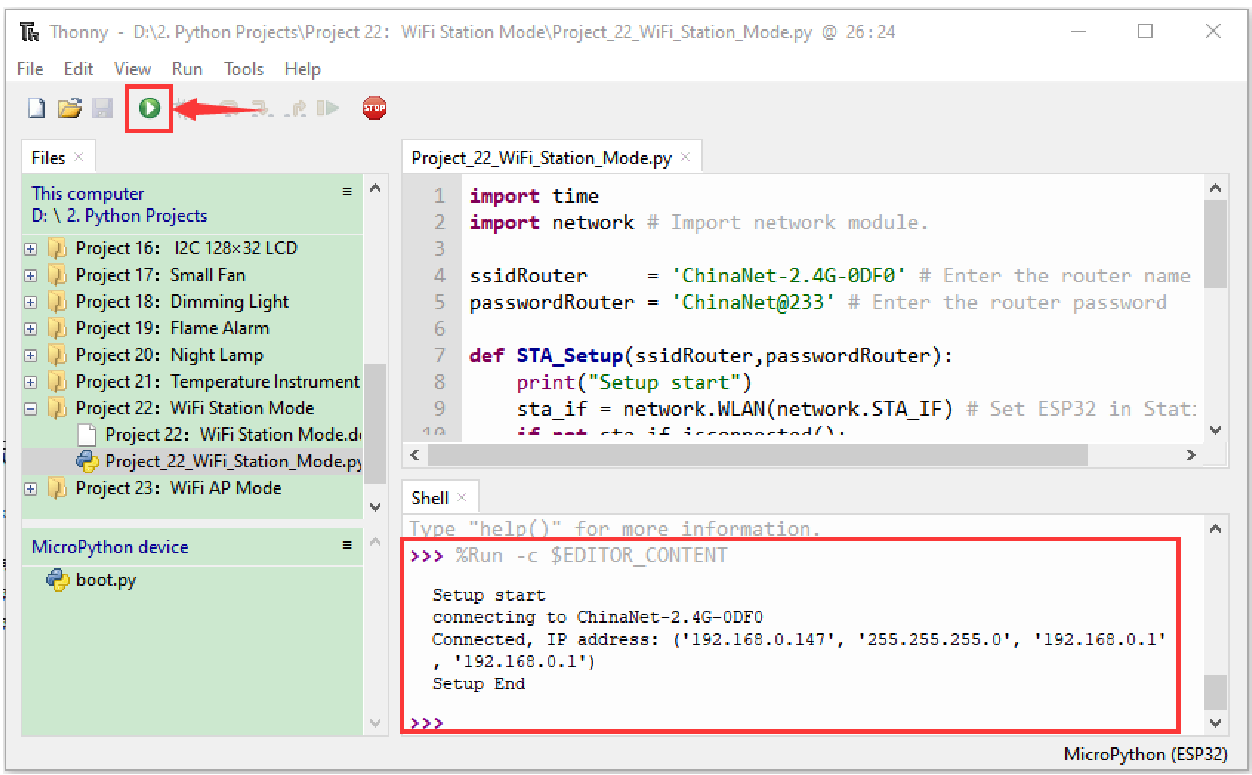
(continues on next page)

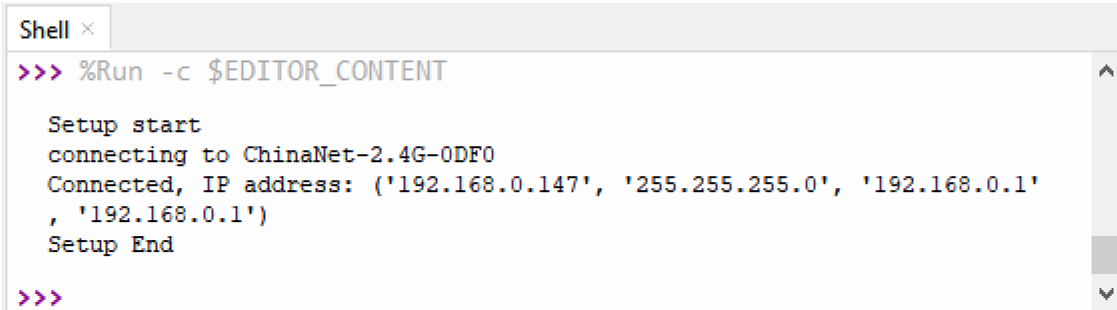
(continued from previous page)

```
        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()
```

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.





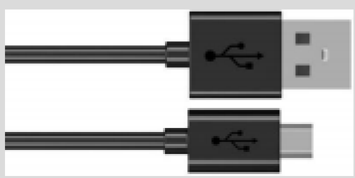
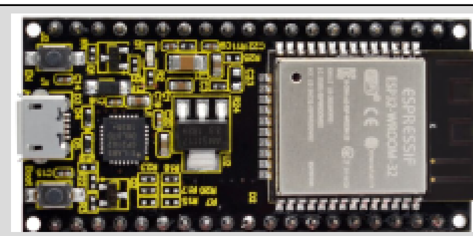
```
Shell
>>> %Run -c $EDITOR_CONTENT

  Setup start
  connecting to ChinaNet-2.4G-0DF0
  Connected, IP address: ('192.168.0.147', '255.255.255.0', '192.168.0.1'
  , '192.168.0.1')
  Setup End

>>>
```

7.25 Project 23WiFi AP Mode

1.Introduction

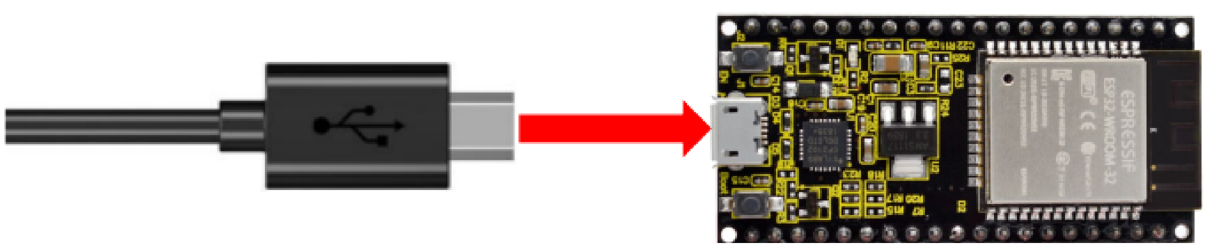
ESP32 has three different WiFi operating modes : Station modeAP mode and AP+Station mode. All WiFi programming projects must be configured with WiFi operating mode before using WiFi, otherwise WiFi cannot be used. In this project, we will learn about ESP32's WiFi AP mode.

2.Components

	
USB Cable*1	ESP22*1

3.Project wiring

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

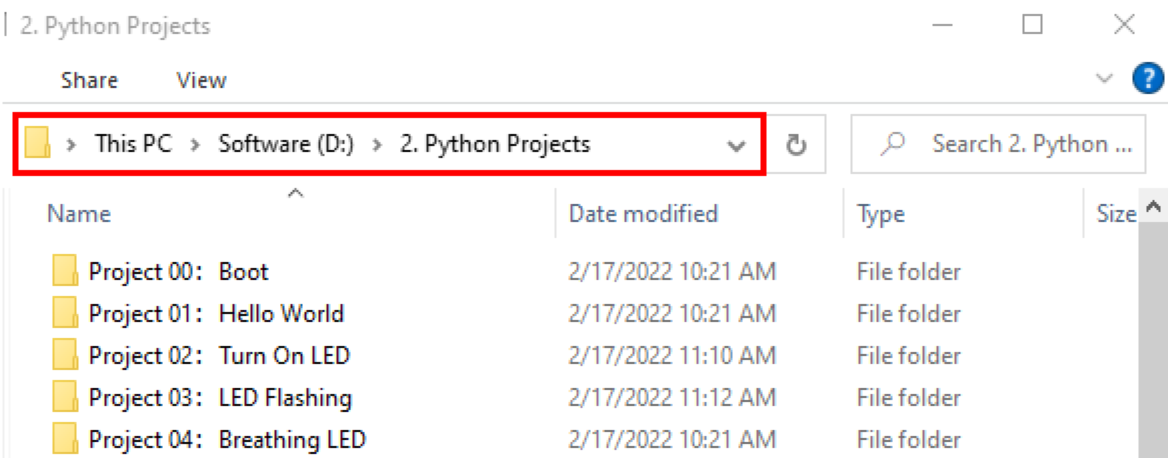


4.Component knowledge

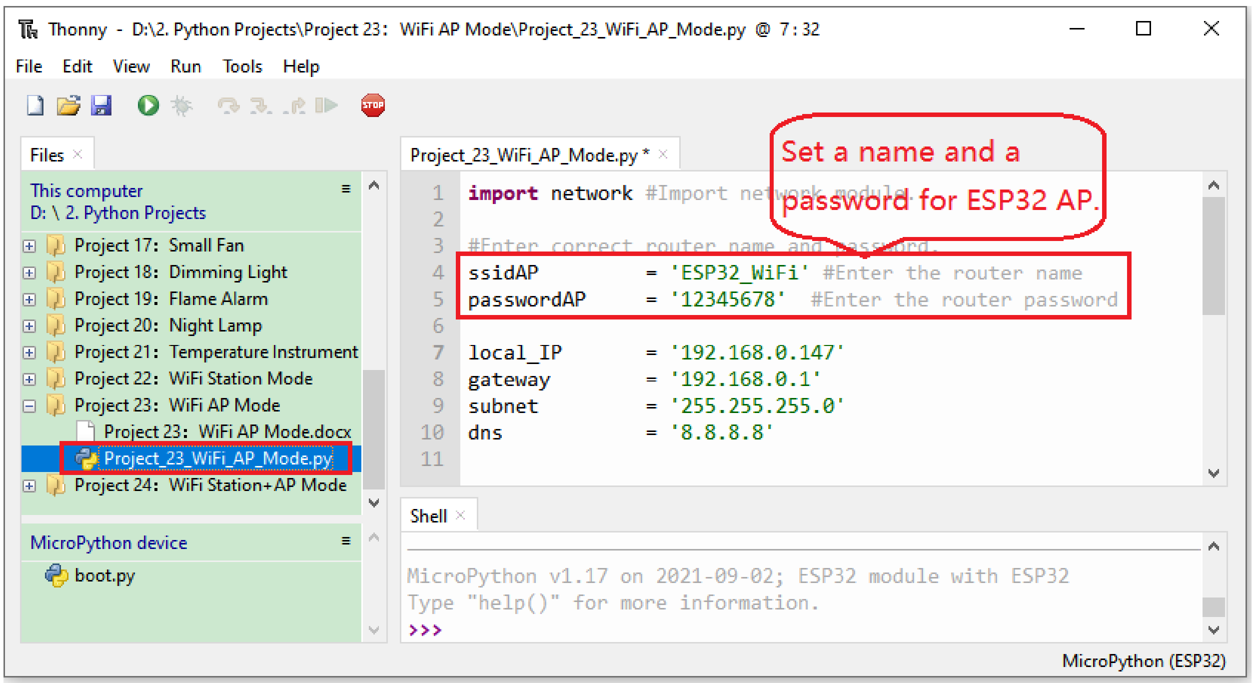
AP mode : When ESP32 selects AP mode, it creates a hotspot network that is separated from the Internet and waits for other WiFi devices to connect. As shown in the figure below, ESP32 is used as a hotspot. If a mobile phone or PC wants to communicate with ESP32, it must be connected to the hotspot of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

5.Project code

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



Open“Thonny”click“This computer”→“D:”→“2. Python Projects”→“Project 23WiFi AP Mode”, and double left-click “Project_23_WiFi_AP_Mode.py”.




```python
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       = '192.168.0.147'
gateway        = '192.168.0.1'
subnet         = '255.255.255.0'
dns            = '8.8.8.8'

#Set ESP32 in AP mode.
ap_if = network.WLAN(network.AP_IF)
```

(continues on next page)

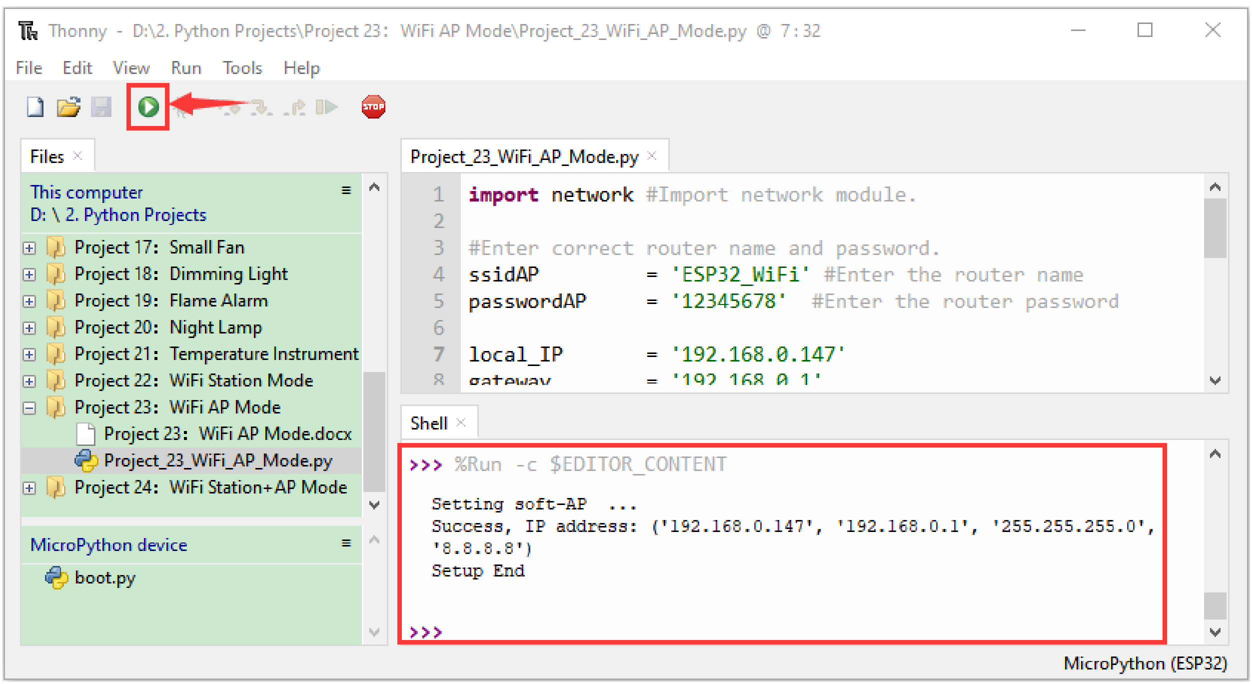
(continued from previous page)

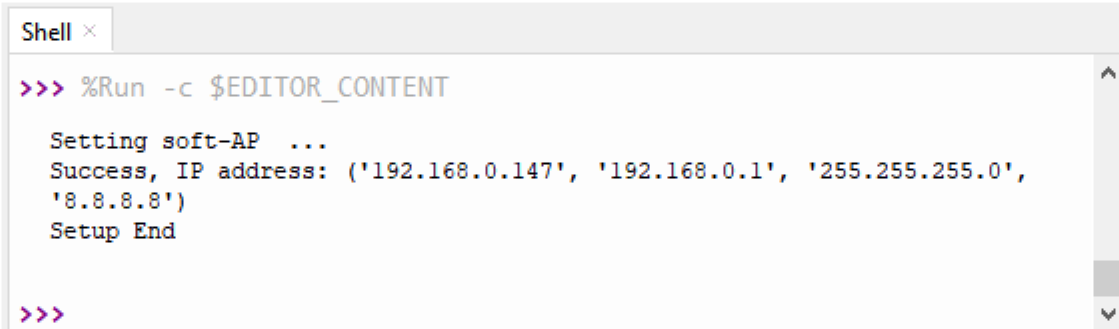
```
def AP_Setup(ssidAP,passwordAP):
    ap_if.ifconfig([local_IP,gateway,subnet,dns])
    print("Setting soft-AP  ... ")
    ap_if.config(essid=ssidAP,authmode=network.AUTH_WPA_WPA2_PSK, password=passwordAP)
    ap_if.active(True)
    print('Success, IP address:', ap_if.ifconfig())
    print("Setup End\n")

try:
    AP_Setup(ssidAP,passwordAP)
except:
    print("Failed, please disconnect the power and restart the operation.")
    ap_if.disconnect()
```

6.Project result

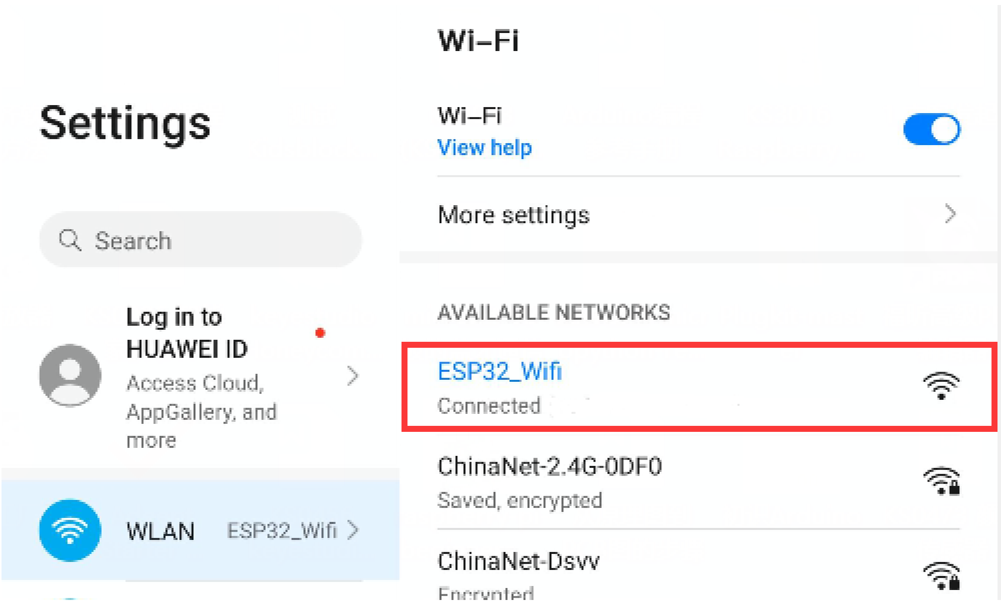
Before the code runs, you can make any changes to the AP name and password for ESP32 in the box as shown in the illustration above. Of course, you can leave it alone by default.

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 ode.

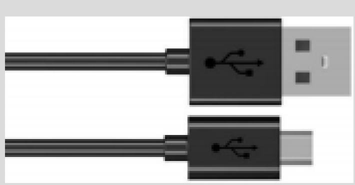
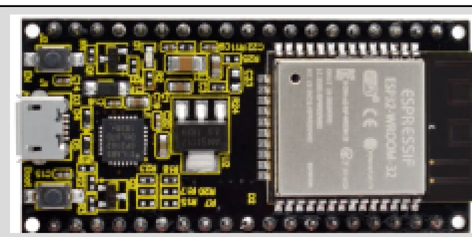


7.26 Project 24WiFi Station+AP Mode

1.Introduction

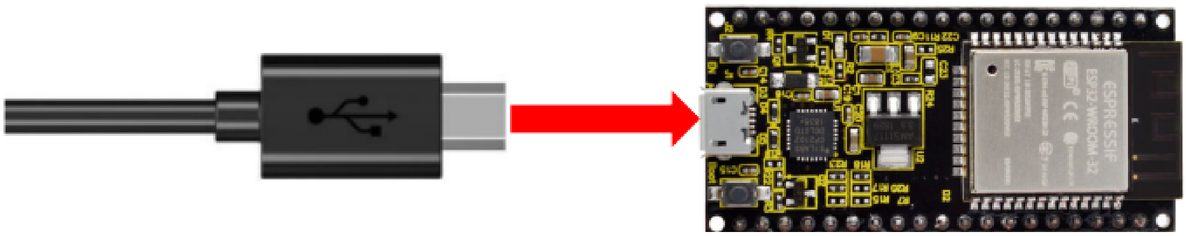
ESP32 has three different WiFi operating modes : Station modeAP mode and AP+Station mode. All WiFi programming projects must be configured with WiFi operating mode before using WiFi, otherwise WiFi cannot be used. In this project, we will learn ESP32's WiFi Station+AP mode.

2.Components

	
USB Cable*1	ESP22*1

3.Project wiring

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

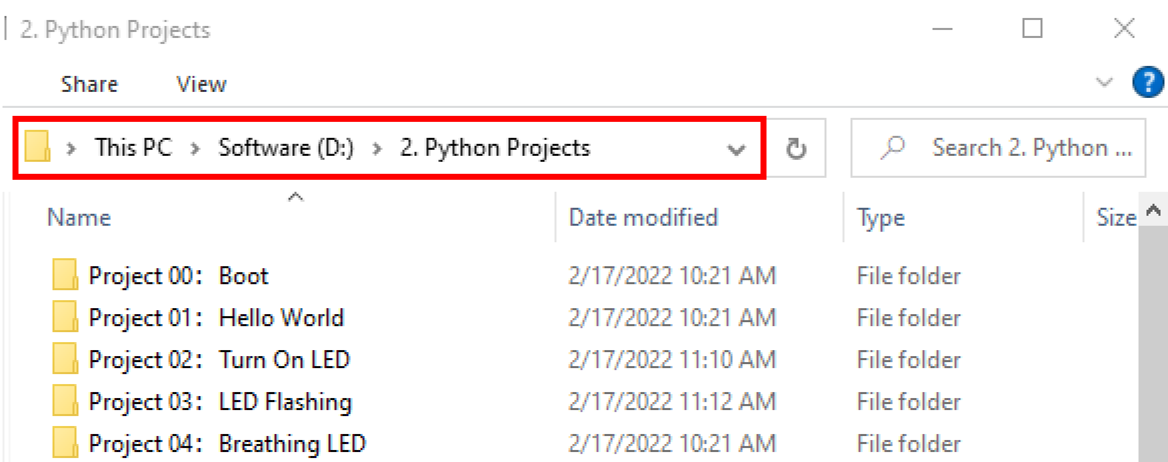


4.Component knowledge

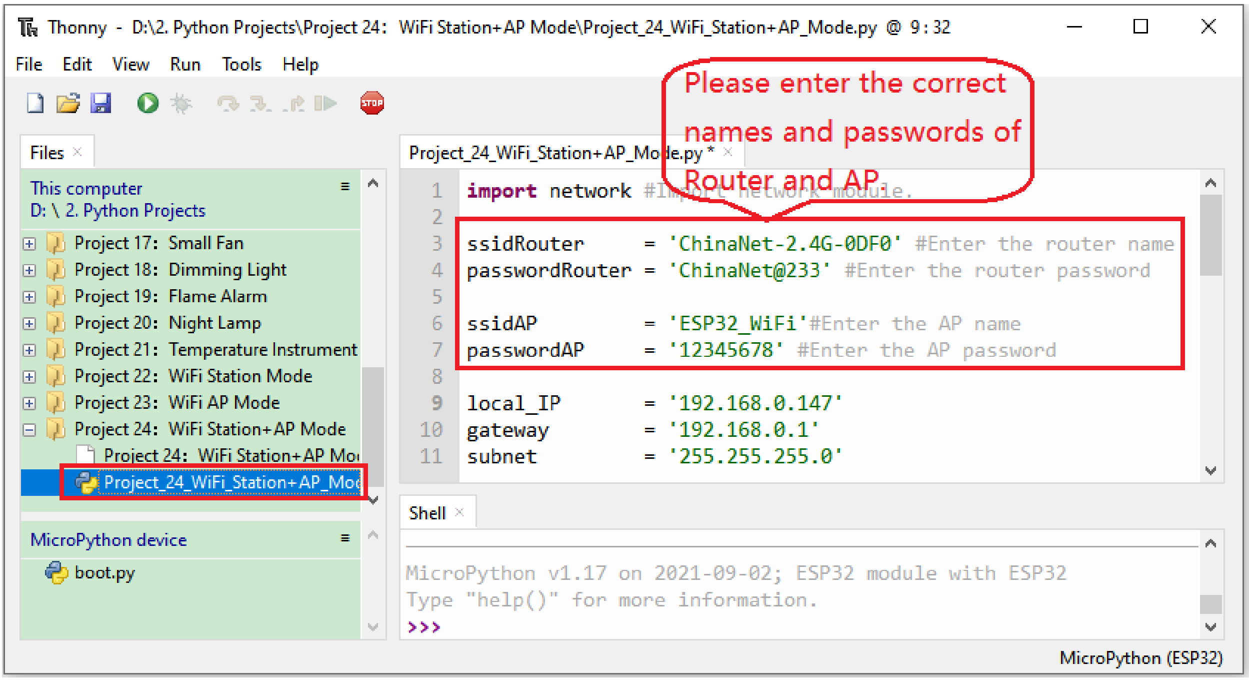
AP+Station mode: In addition to AP mode and Station mode, ESP32 can also use AP mode and Station mode at the same time. This mode contains the functions of the previous two modes. Turn on ESP32's Station mode, connect it to the router network, and it can communicate with the Internet via the router.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

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



Open"Thonny"click"This computer"→"D:"→"2. Python Projects"→"Project 24WiFi Station+AP Mode"and double left-click "Project_24_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       = '192.168.0.147'
gateway        = '192.168.0.1'
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)

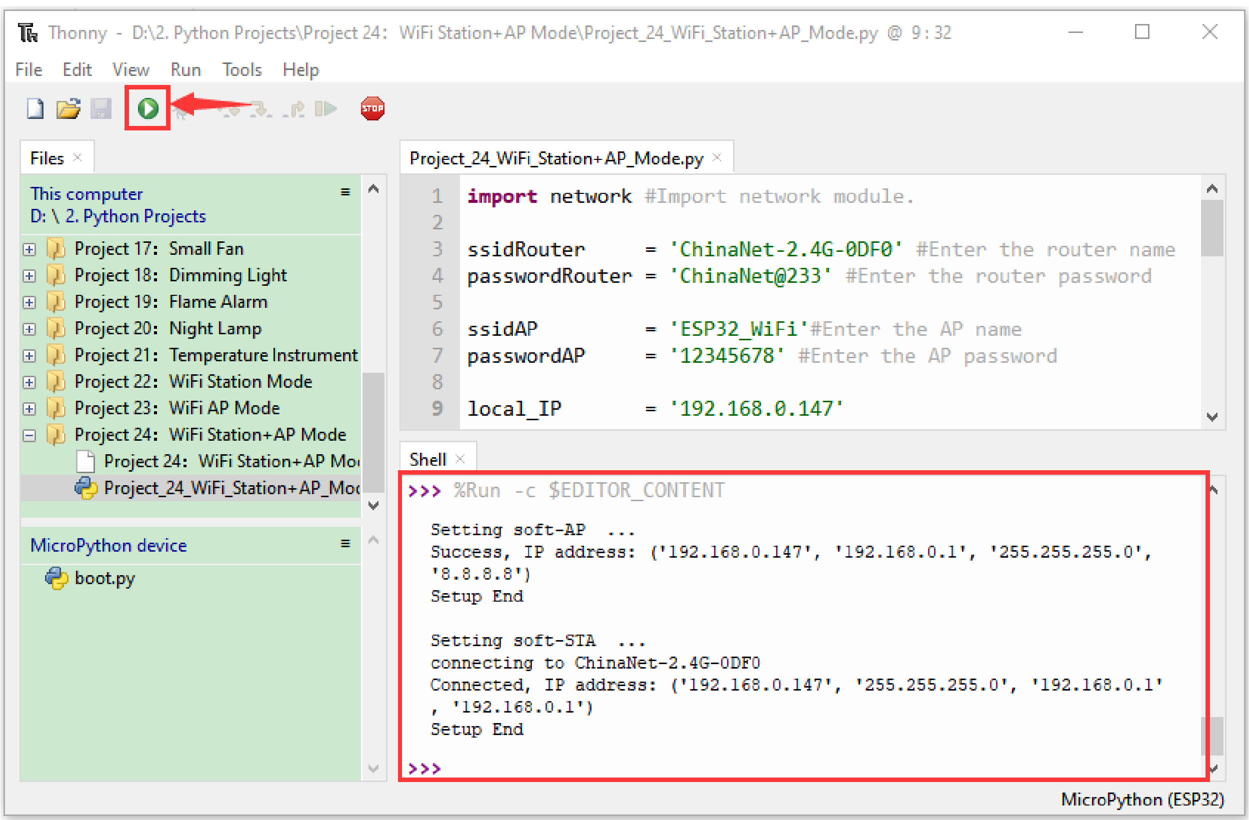
(continued from previous page)

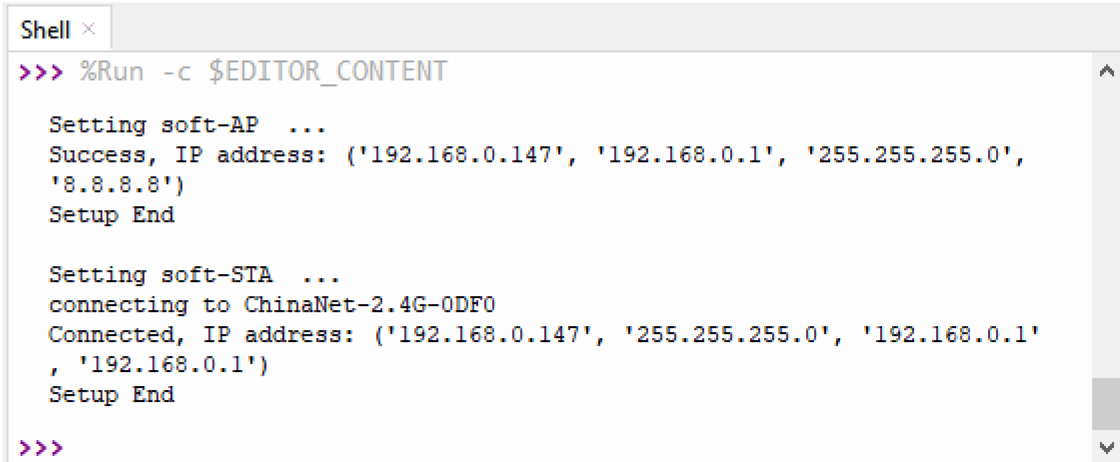
```
    ap_if.active(True)
    print('Success, IP address:', ap_if.ifconfig())
    print("Setup End\n")

try:
    AP_Setup(ssidAP,passwordAP)
    STA_Setup(ssidRouter,passwordRouter)
except:
    sta_if.disconnect()
    ap_if.idsconnect()
```

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.

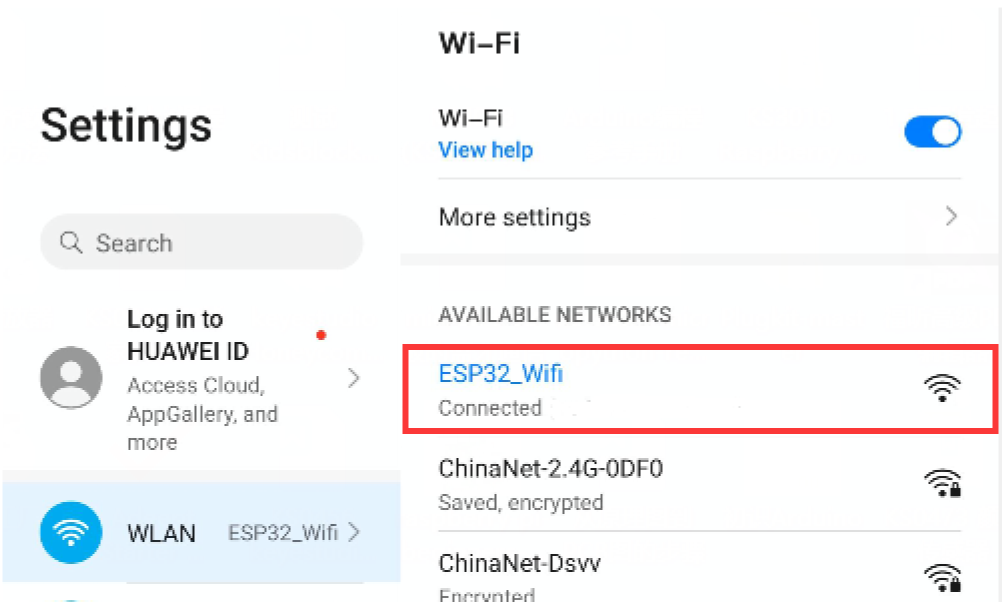
After making sure that the code is modified correctly, click "Run current script" the code starts to be executed and the"Shell"window of Thonny IDE will display as follows:





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

  Setting soft-AP  ...
  Success, IP address: ('192.168.0.147', '192.168.0.1', '255.255.255.0',
  '8.8.8.8')
  Setup End

  Setting soft-STA  ...
  connecting to ChinaNet-2.4G-0DF0
  Connected, IP address: ('192.168.0.147', '255.255.255.0', '192.168.0.1'
  , '192.168.0.1')
  Setup End

>>>
```

Turn on the WiFi scanning function of your phone, and you can see the ssidAP on ESP32.



s

CHAPTER

EIGHT

GETTING STARTED WITH C LANGUAGE(RASPBERRY PI)

Raspberry Pi is a card computer whose official system is Raspberry Pi OS, which can be installed on other systems, such as: ubuntu, Windows IoT. Raspberry Pi can be used as a personal server, a router camera monitoring and recognition, as well as voice interaction by connecting a camera and a voice interactive assistant.

Furthermore, Raspberry Pi leads out 40Pin pins that can be connected to various sensors and control LEDs, motors, etc, making it can be used to create a robot.

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

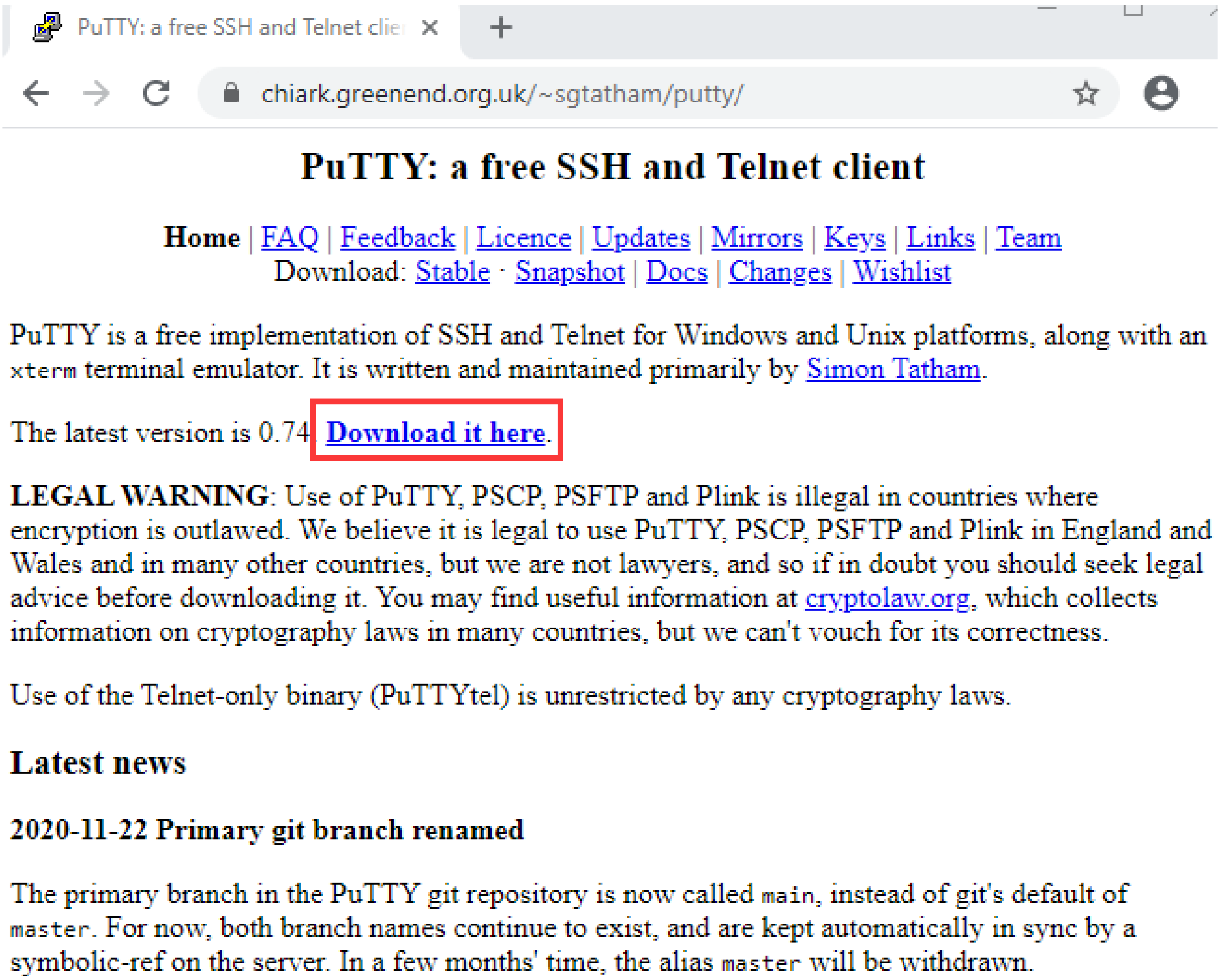
4Computer and other parts

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**.

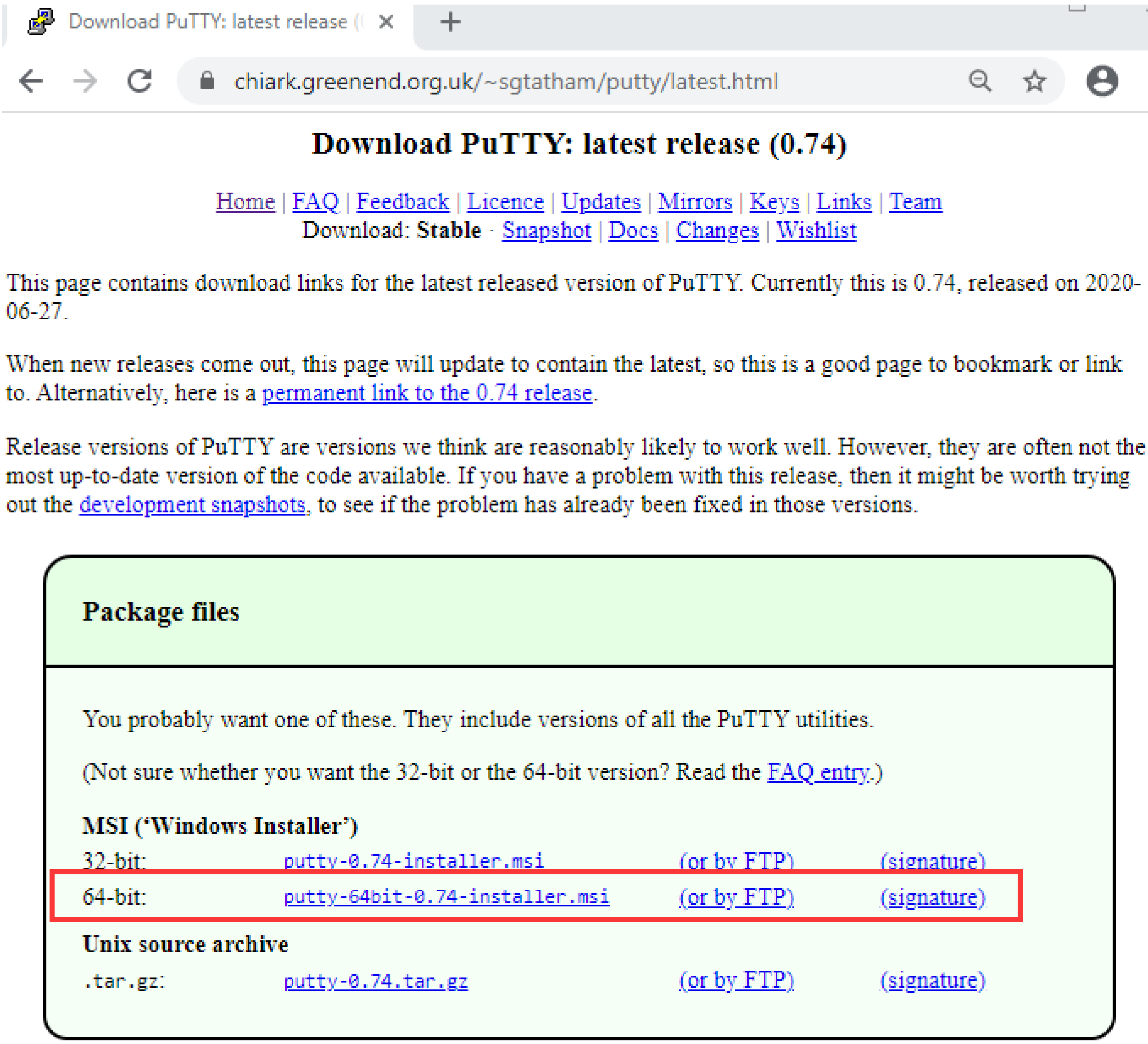
LEGAL WARNING: Use of PuTTY, PSCP, PSFTP and Plink is illegal in countries where encryption is outlawed. We believe it is legal to use PuTTY, PSCP, PSFTP and Plink in England and Wales and in many other countries, but we are not lawyers, and so if in doubt you should seek legal advice before downloading 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.

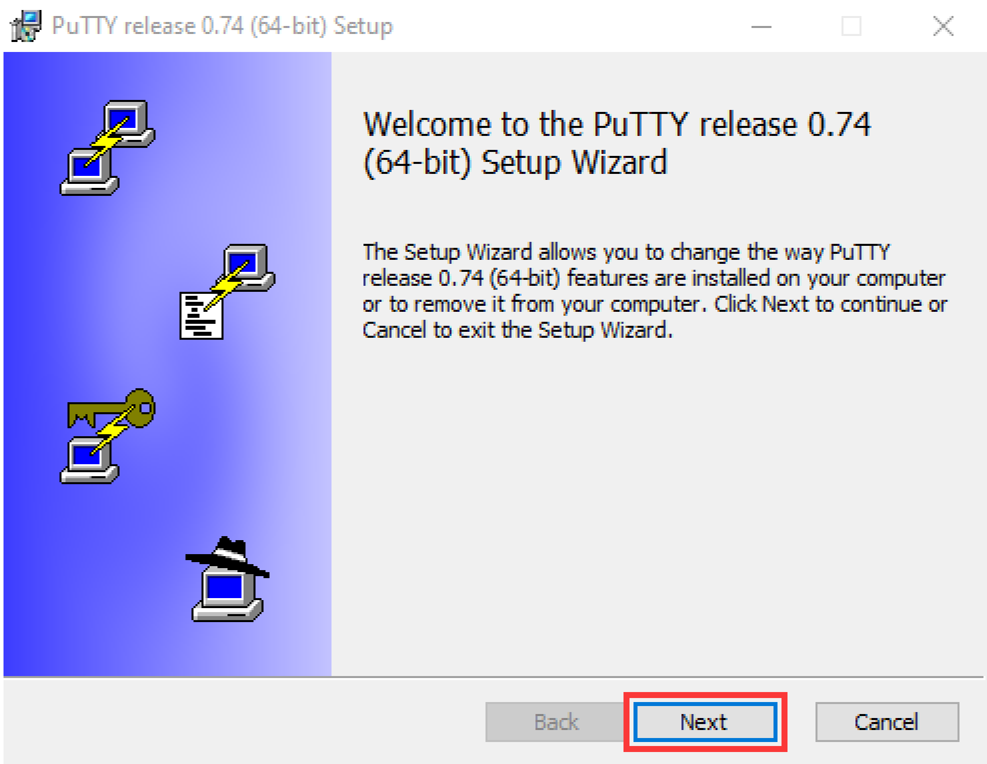
Latest news

2020-11-22 Primary git branch renamed

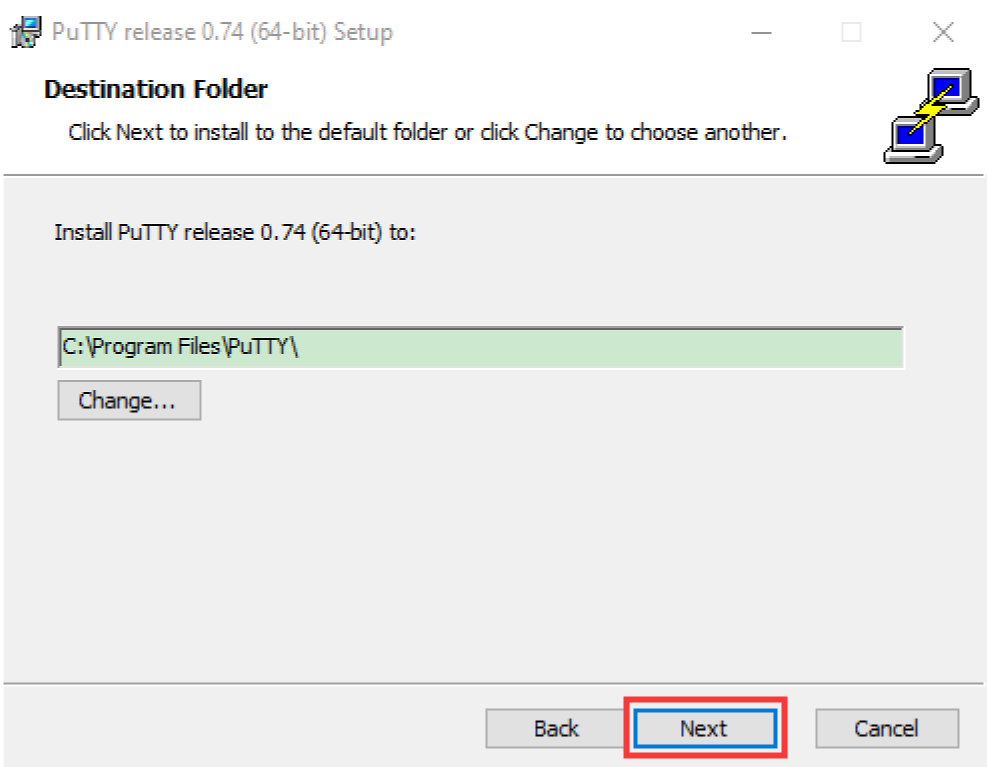
The primary branch in the PuTTY git repository is now called `main`, instead of git's default of `master`. For now, both branch names continue to exist, and are kept automatically in sync by a symbolic-ref on the server. In a few months' time, the alias `master` will be withdrawn.



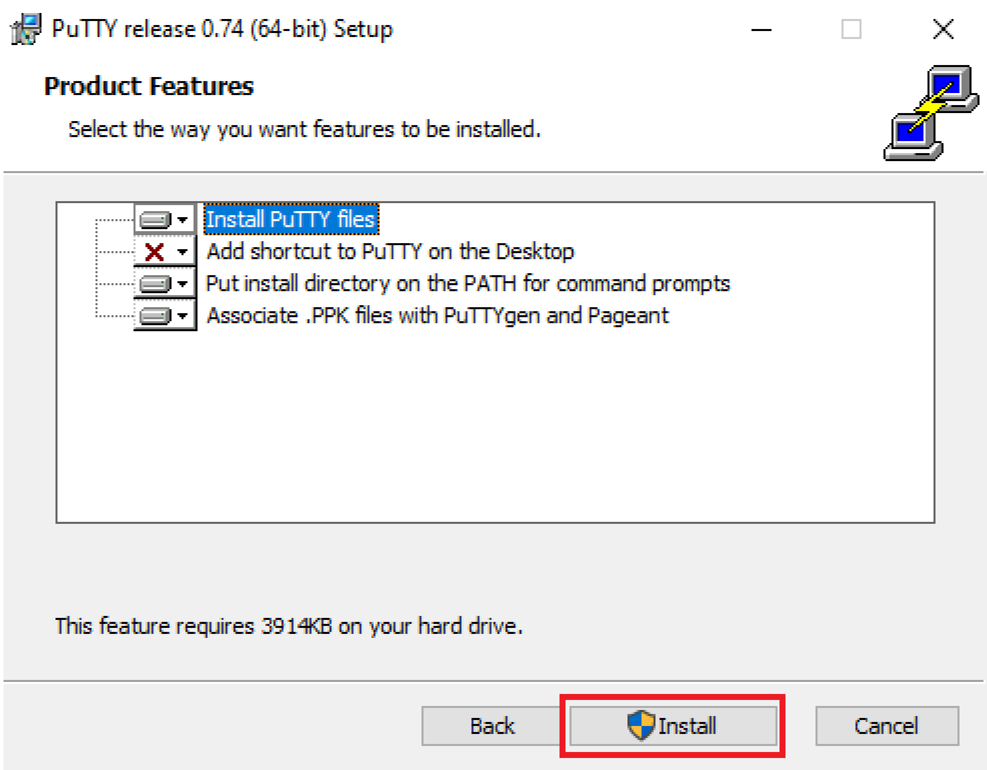
After downloading the package file putty-64bit-0.74-installer , double-click it and tap"Next".



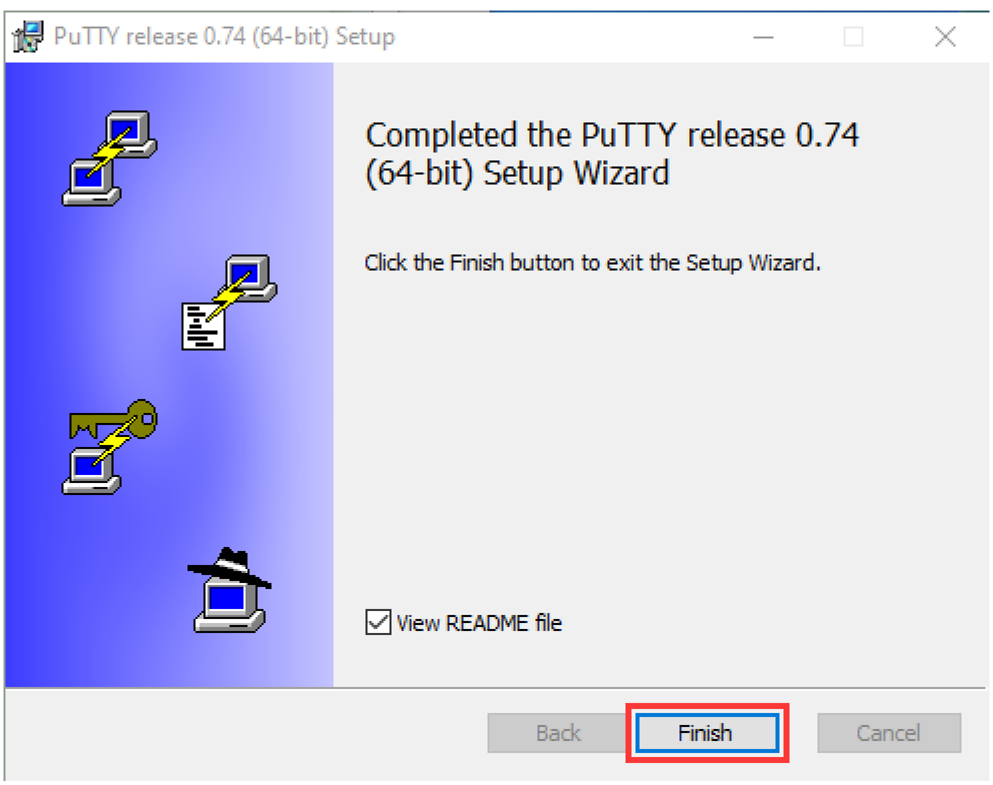
Click "Next".



Select "Install Putty files", and click "Install"



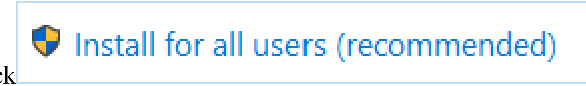
After a few seconds, the installation is complete, click "Finish".



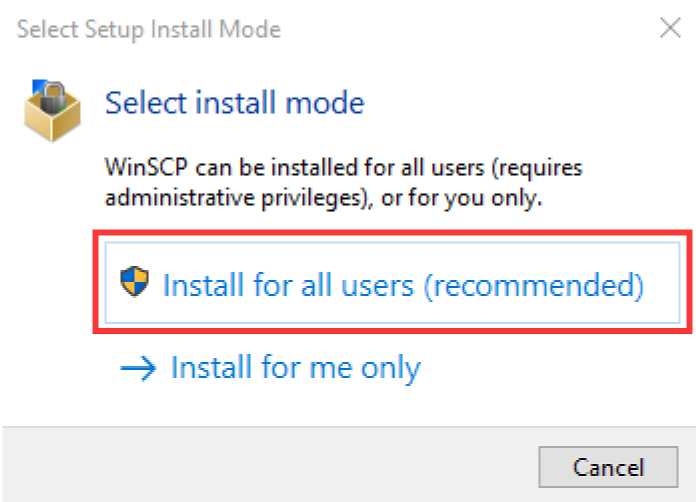
Remote Login software -WinSCP

Download link: https://winscp.net/eng/download.php

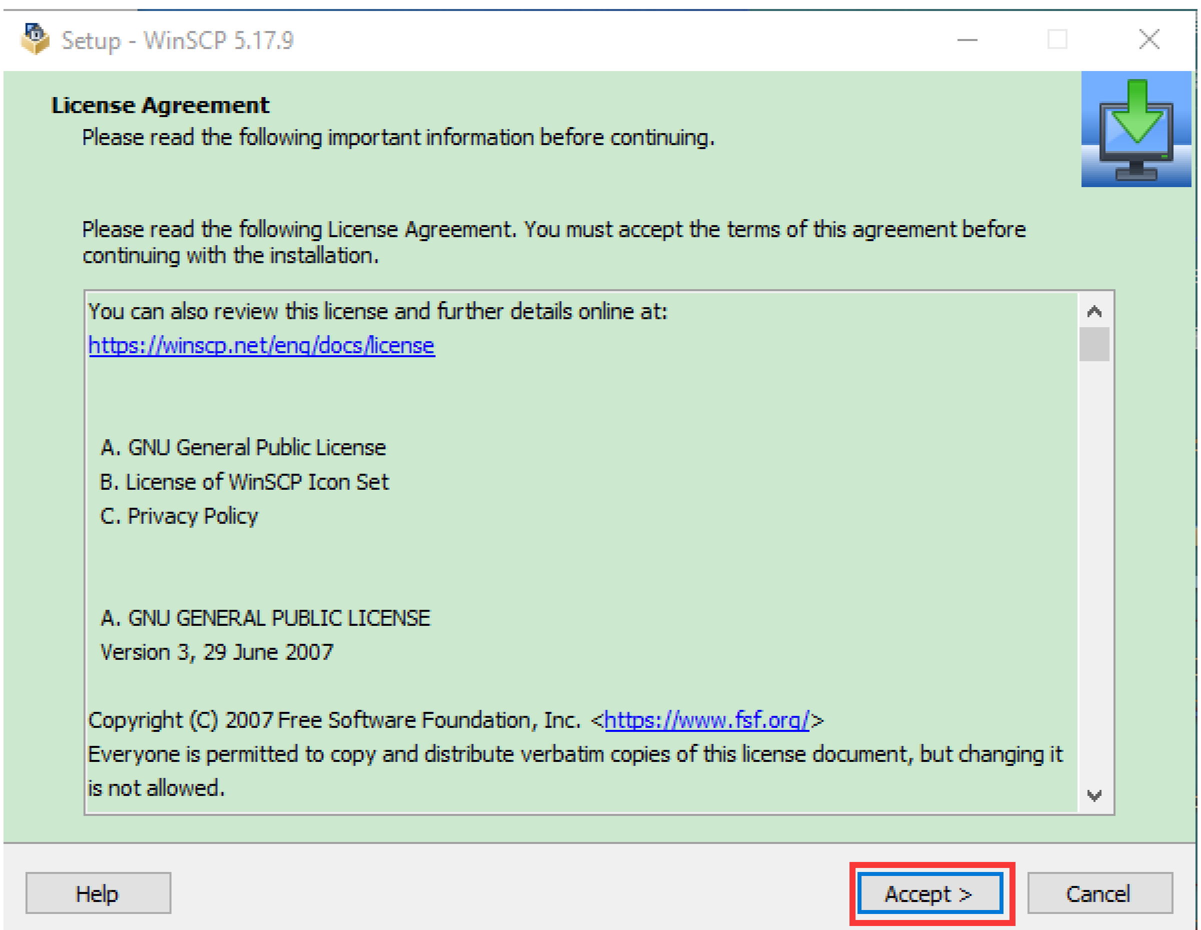
After downloading the WinSCP software file 

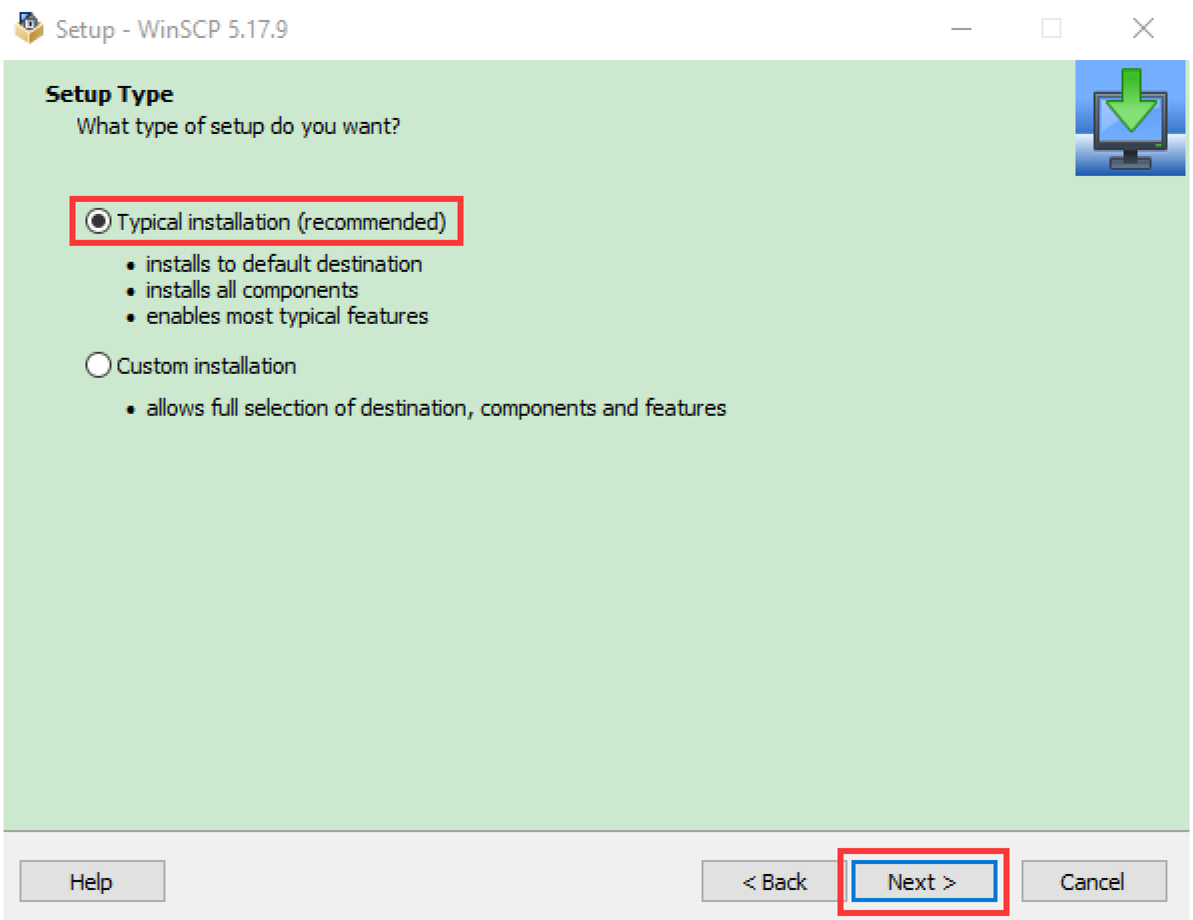
, double-click it then click 

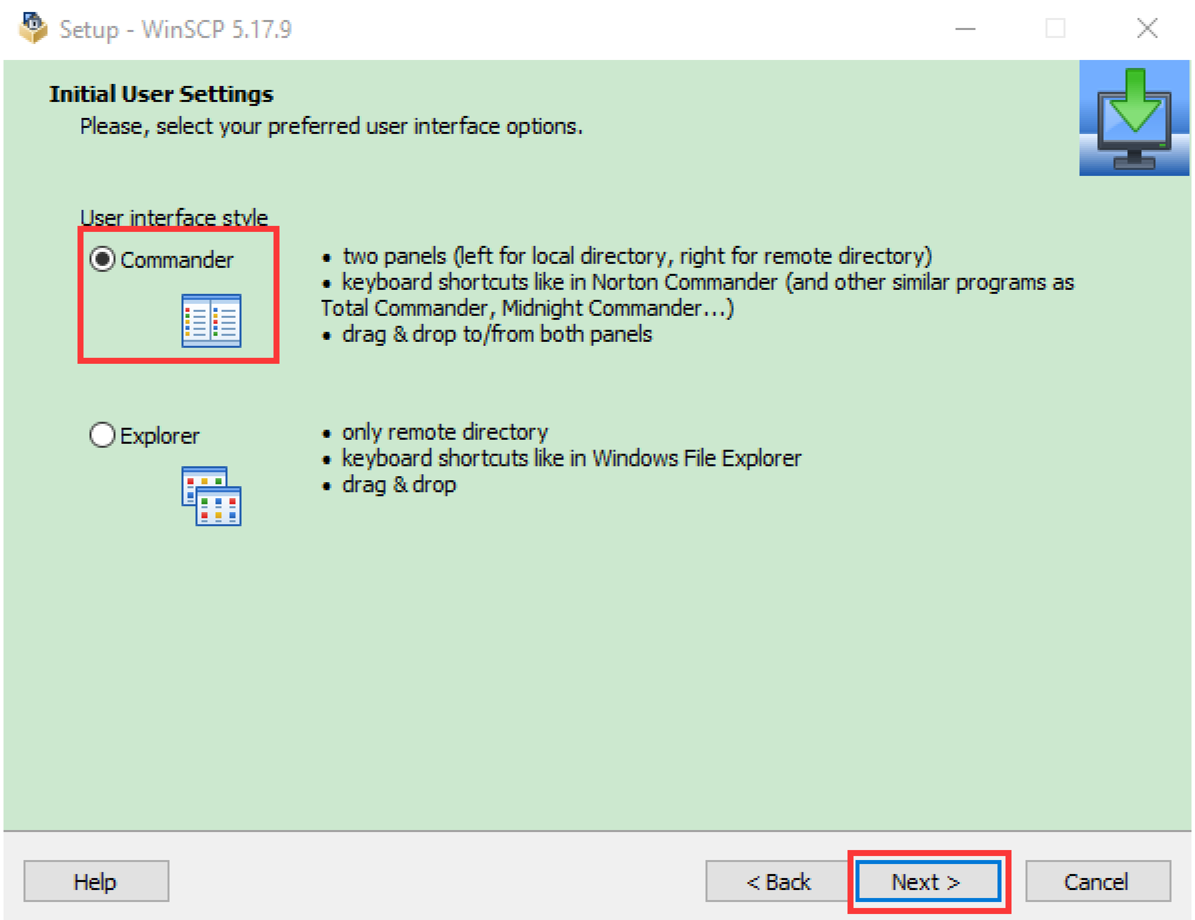
.

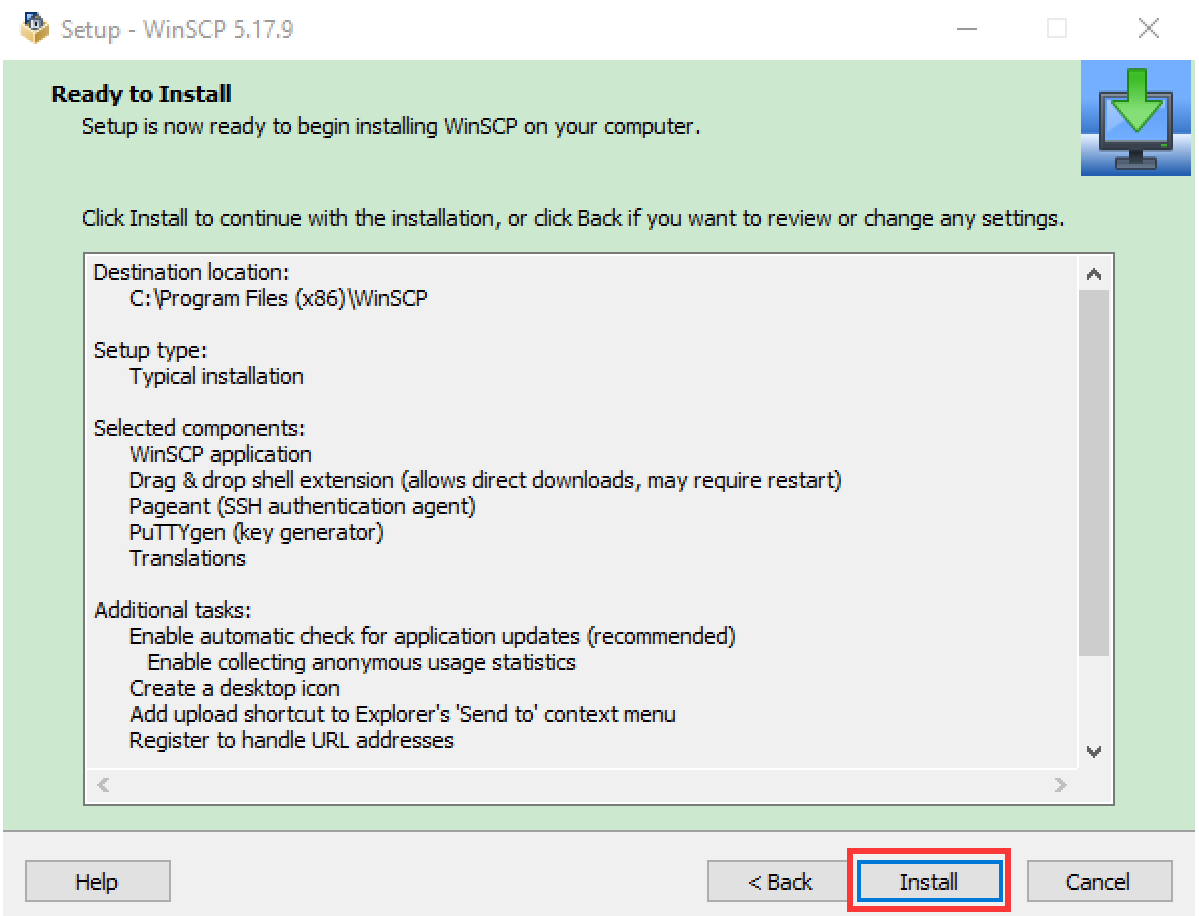


Click"Accept"then select the appropriate option and click"Next", then click"Install".




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

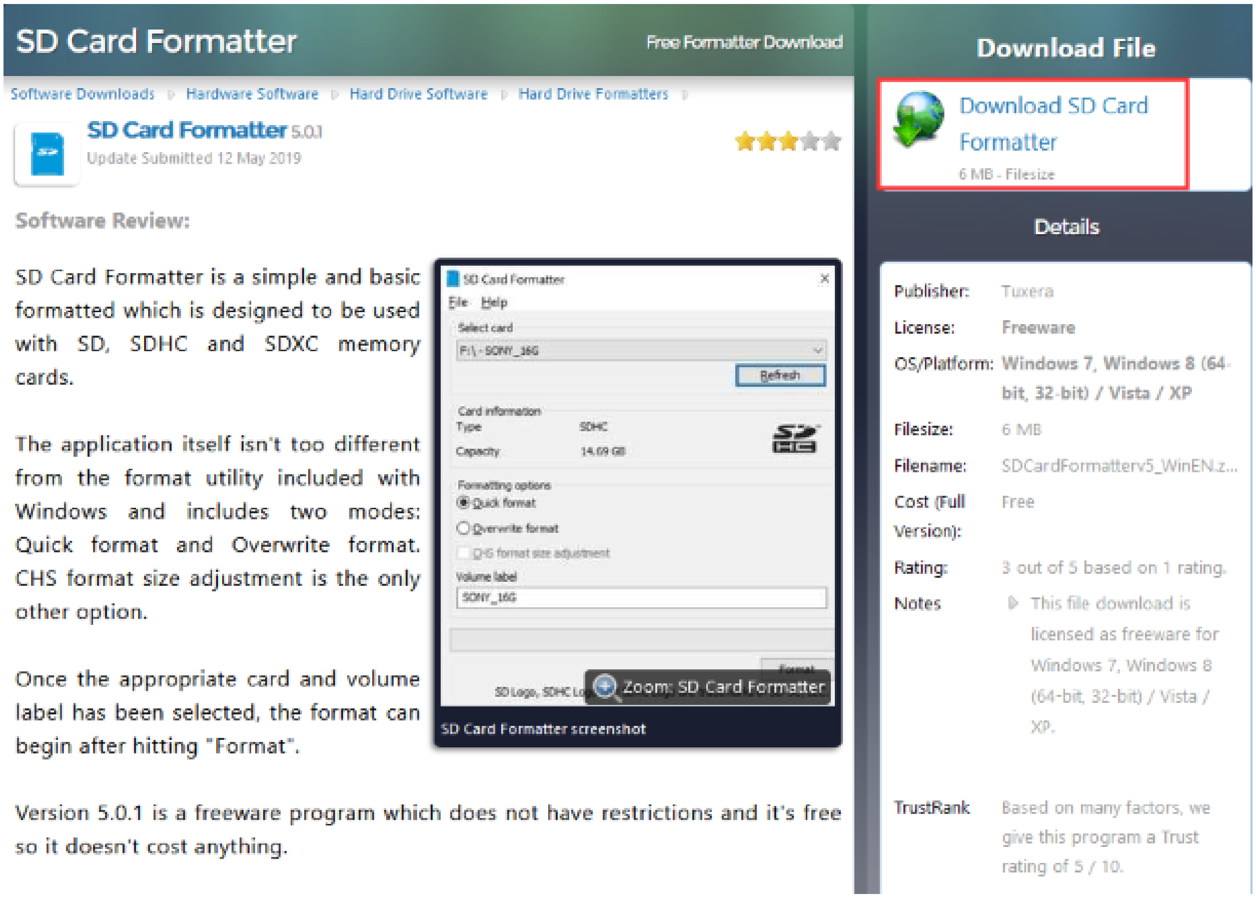


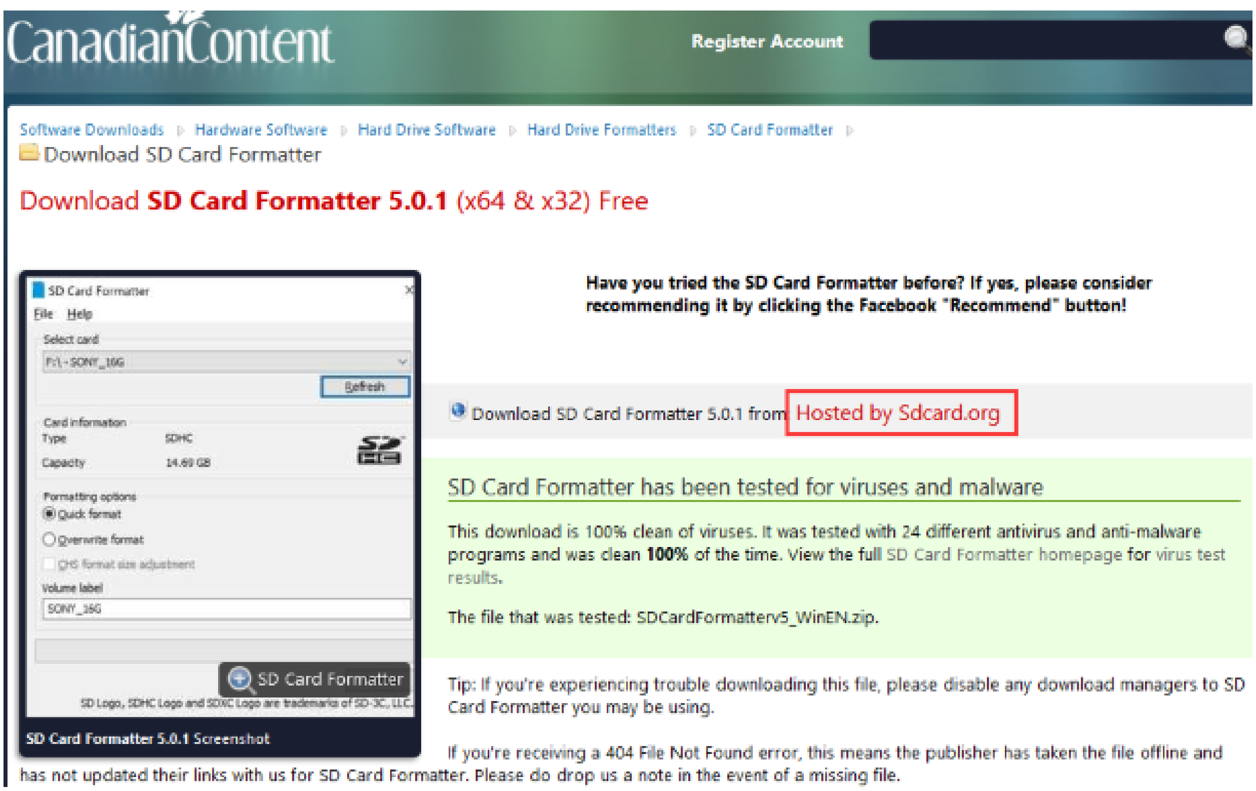
After a few seconds, the installation is complete, click "Finish".

Format TFT card tool– SD Card Formatter

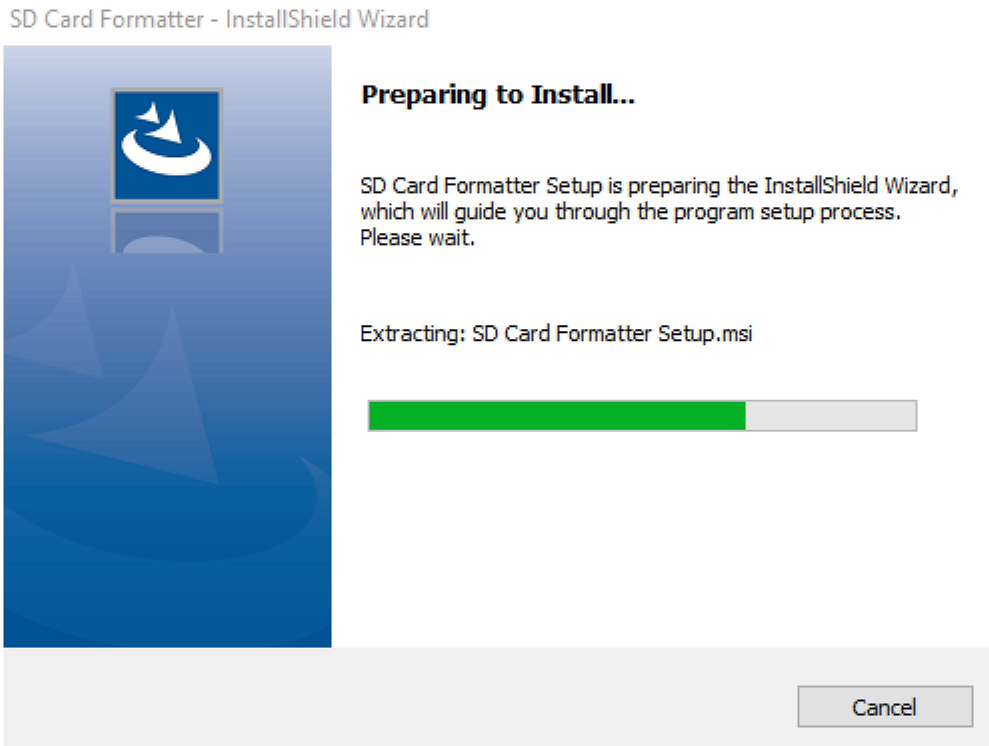
Download link

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

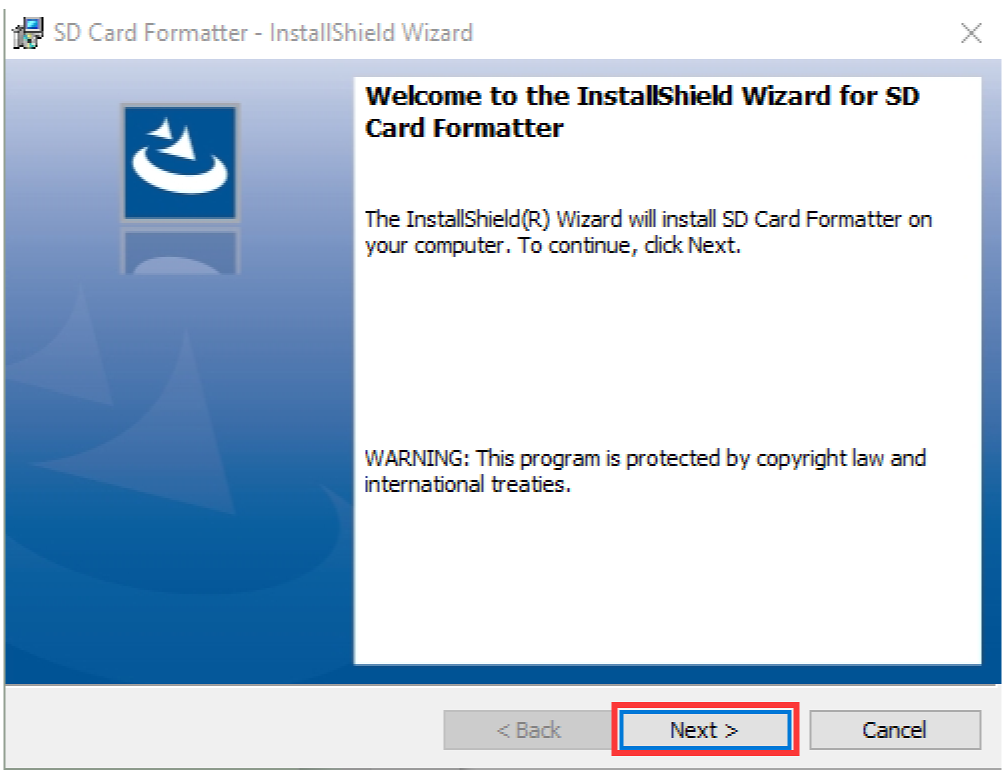

SD Card Formatter
Free Formatter Download
Software Downloads ▷ Hardware Software ▷ Hard Drive Software ▷ Hard Drive Formatters ▷
SD Card Formatter 5.0.1
Update Submitted 12 May 2019
Software Review:
SD Card Formatter is a simple and basic formatted which is designed to be used with SD, SDHC and SDXC 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.

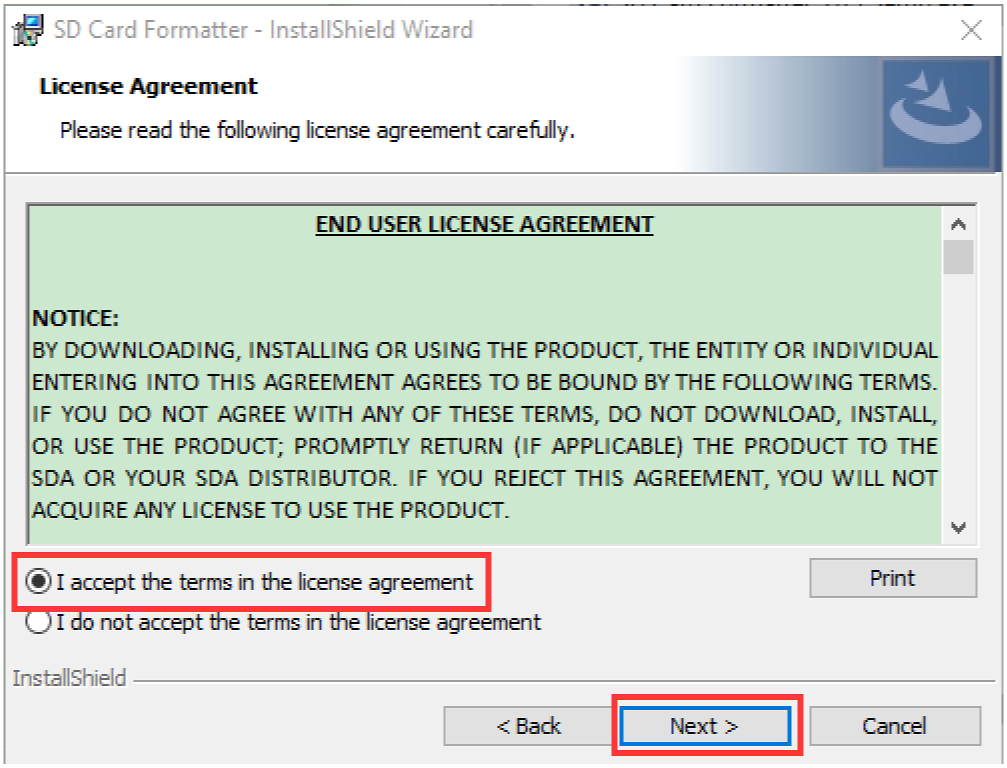

CanadianContent
Register Account
Software Downloads ▷ Hardware Software ▷ Hard Drive Software ▷ Hard Drive Formatters ▷ SD Card Formatter ▷
Download SD Card Formatter
Download SD Card Formatter 5.0.1 (x64 & x32) Free
Have you tried the SD Card Formatter before? If yes, please consider recommending it 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.

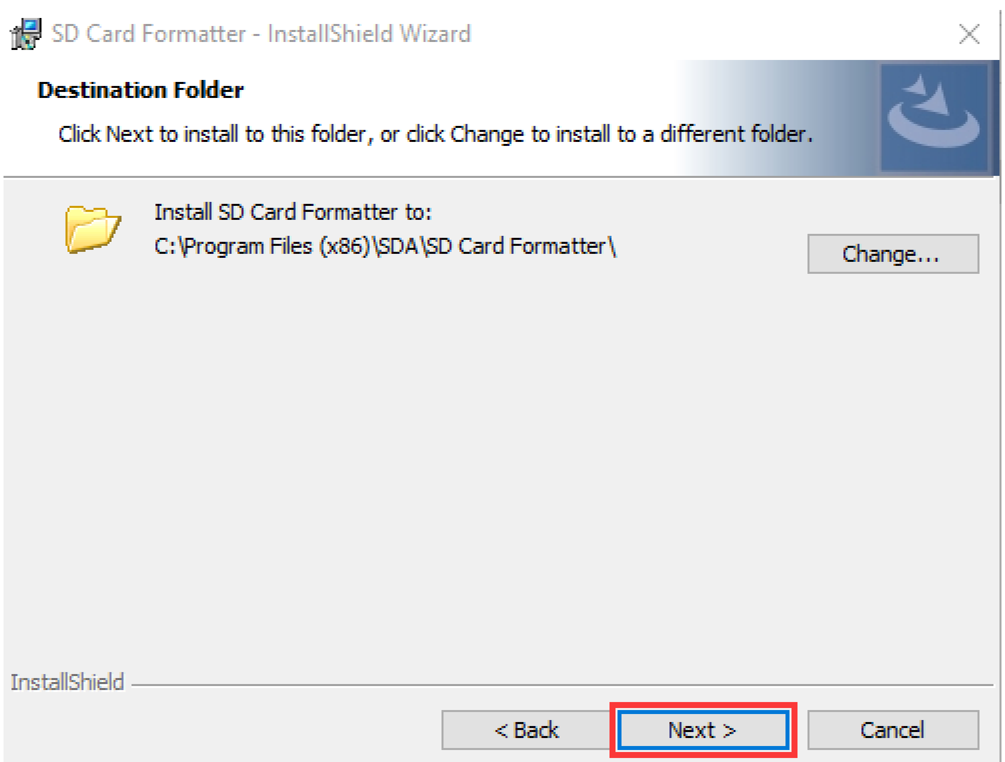


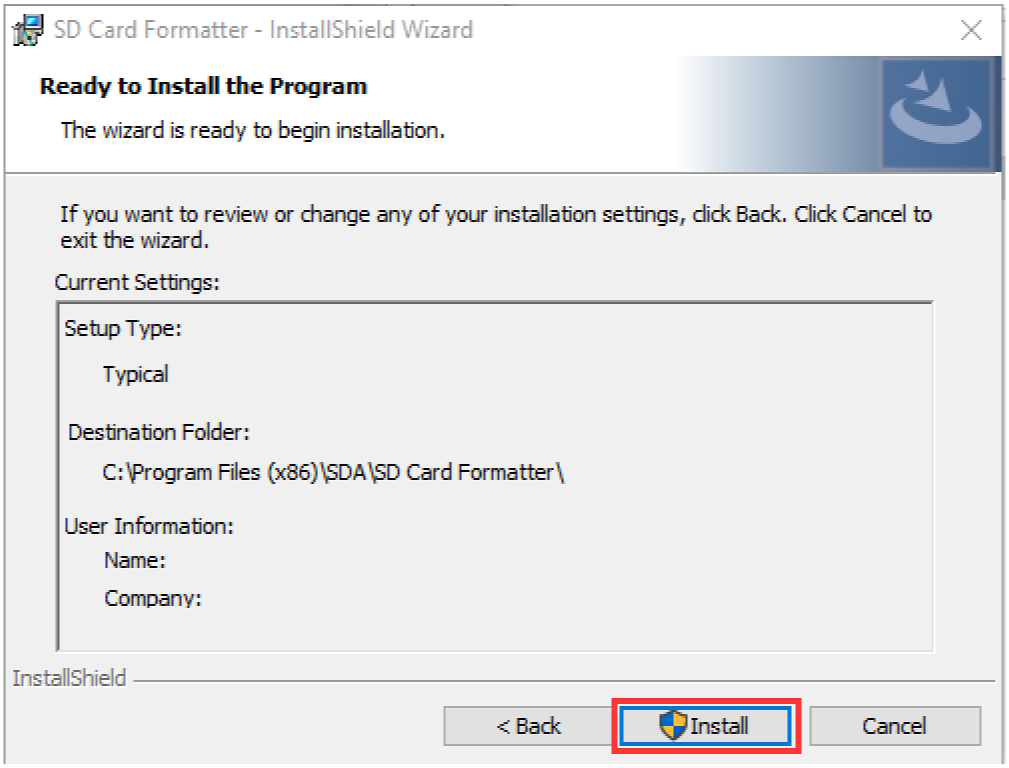
Click "Next"select " I accept the terms in the license agreement " and click "Next".



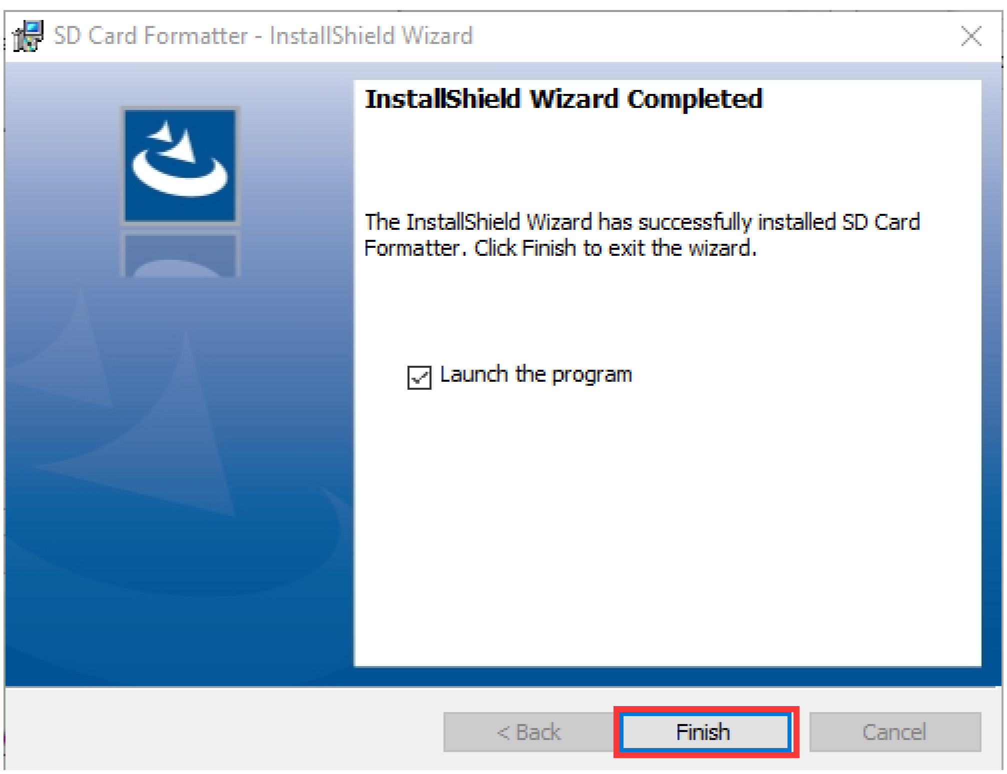


Click “Next” again, and then click “Install”.



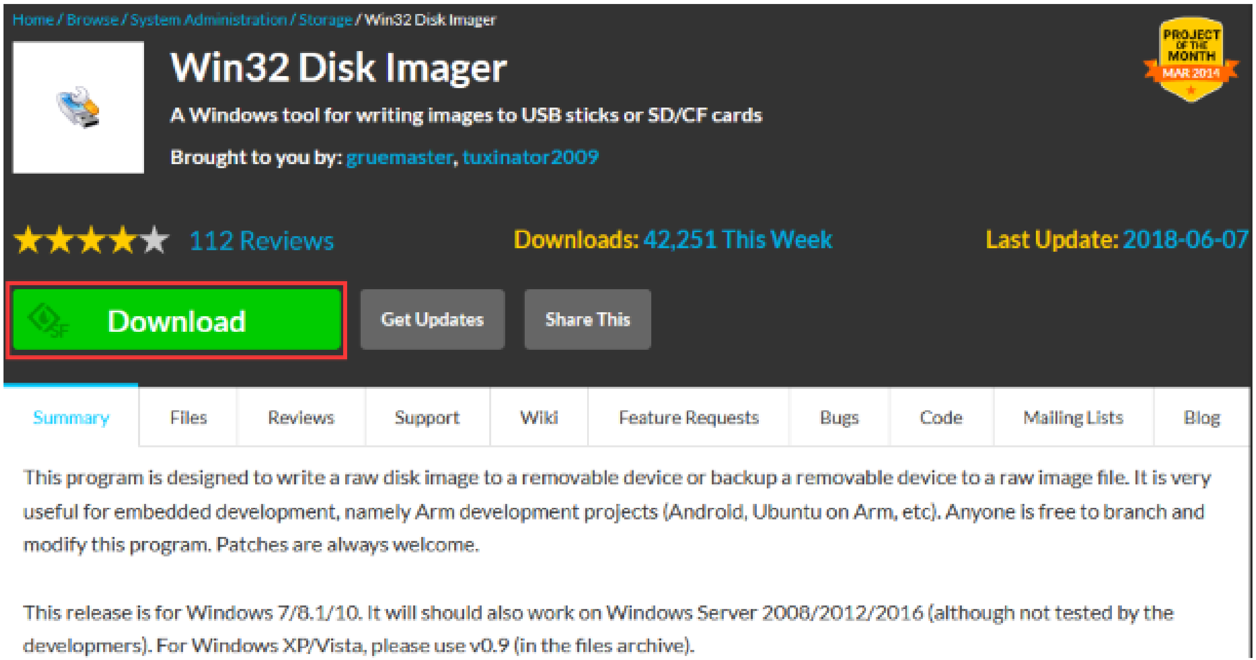


After a few seconds, the installation is complete, click "Finish".

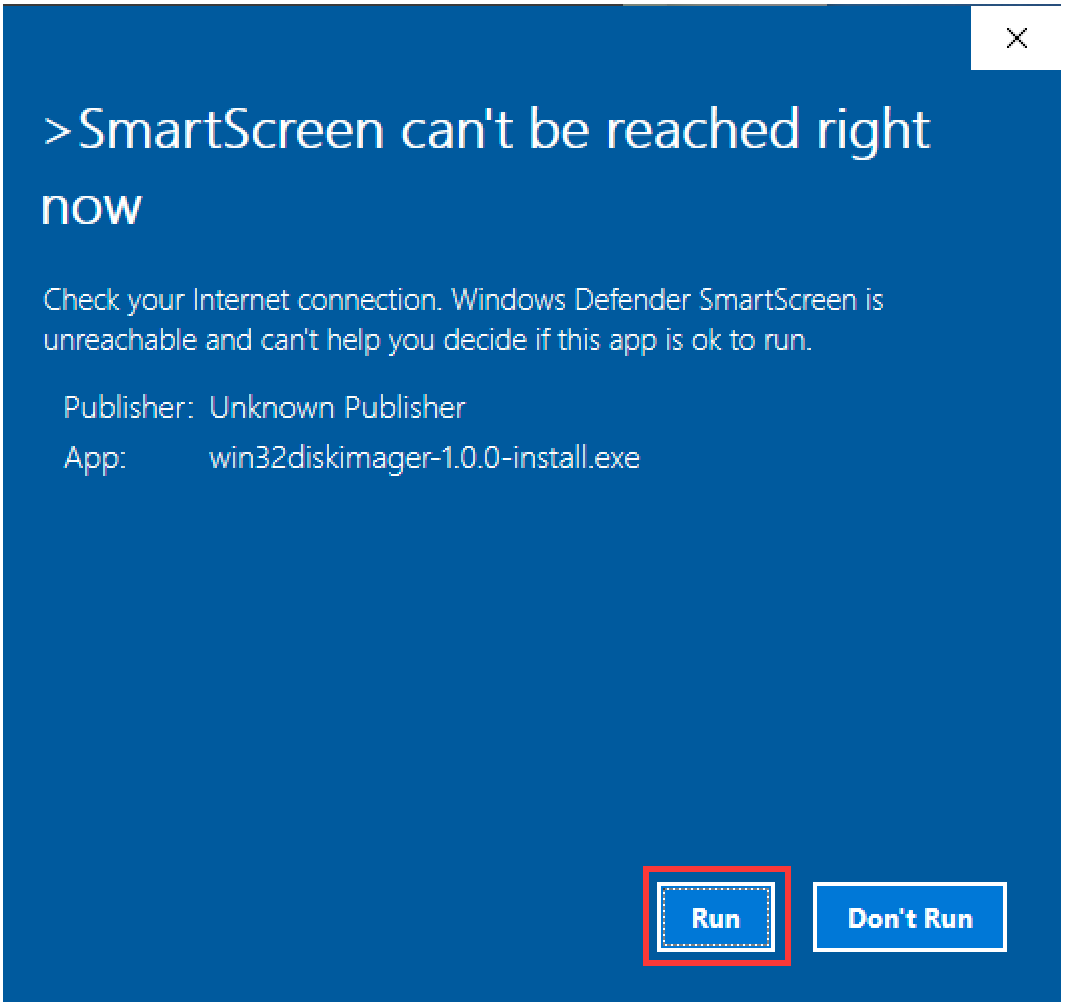


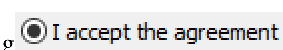
4Burn mirror system software tool— Win32DiskImager

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

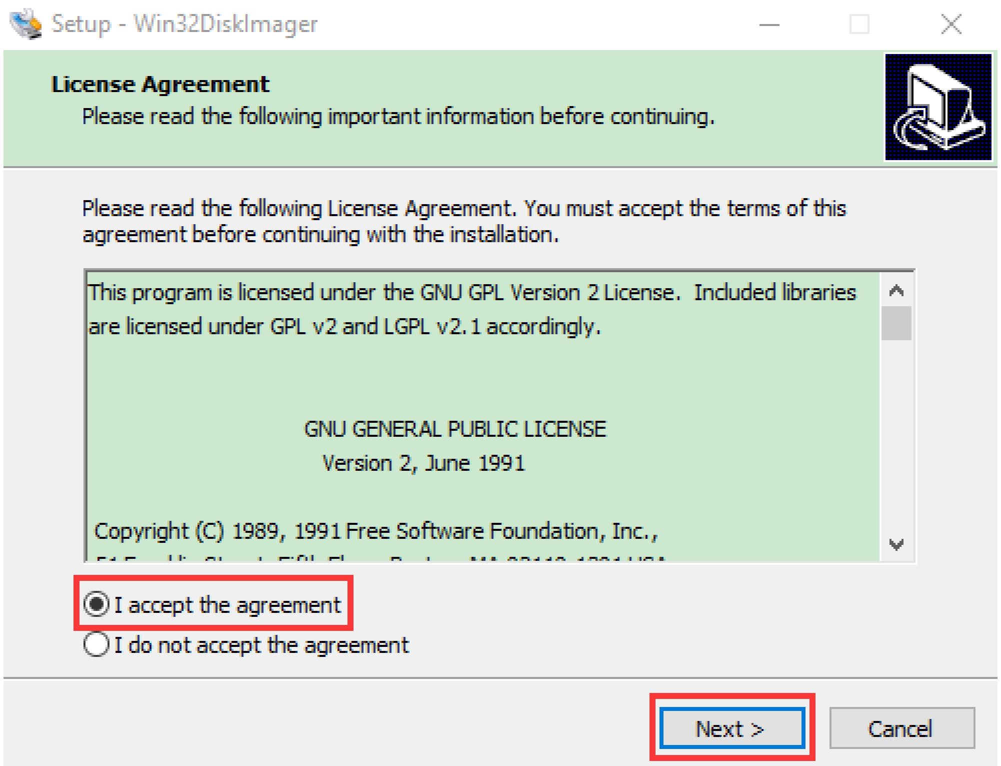


After downloading the software file win32diskimager-1.0.0-install.exe double-click it and then click "Run".

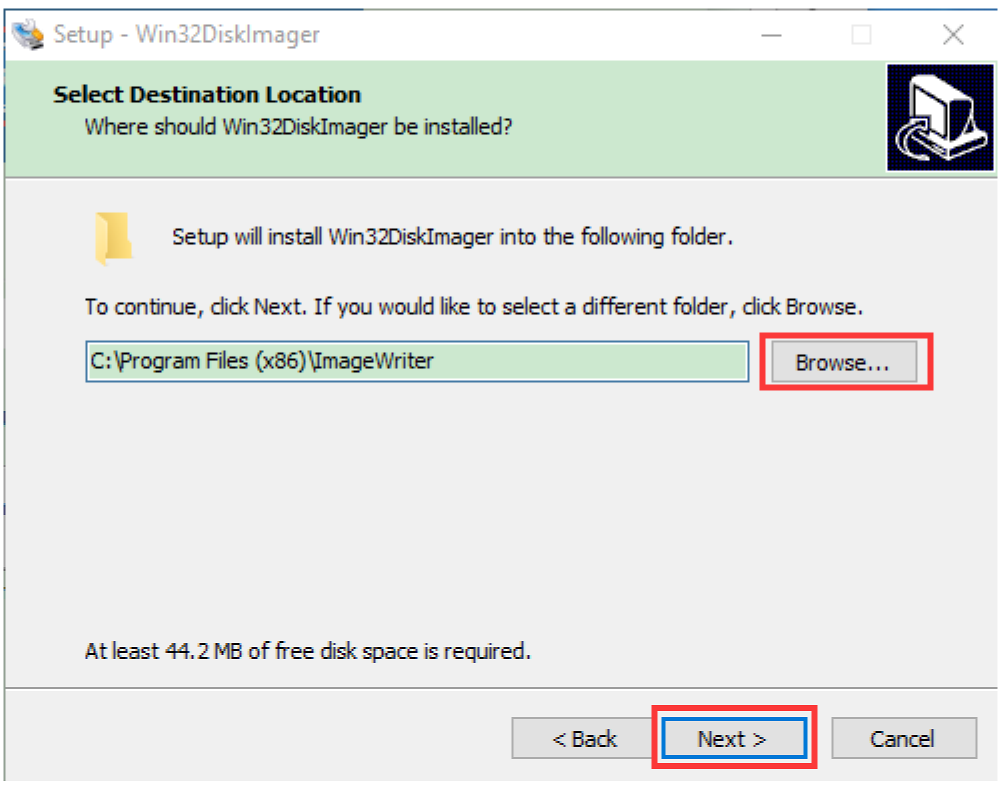


After selecting 

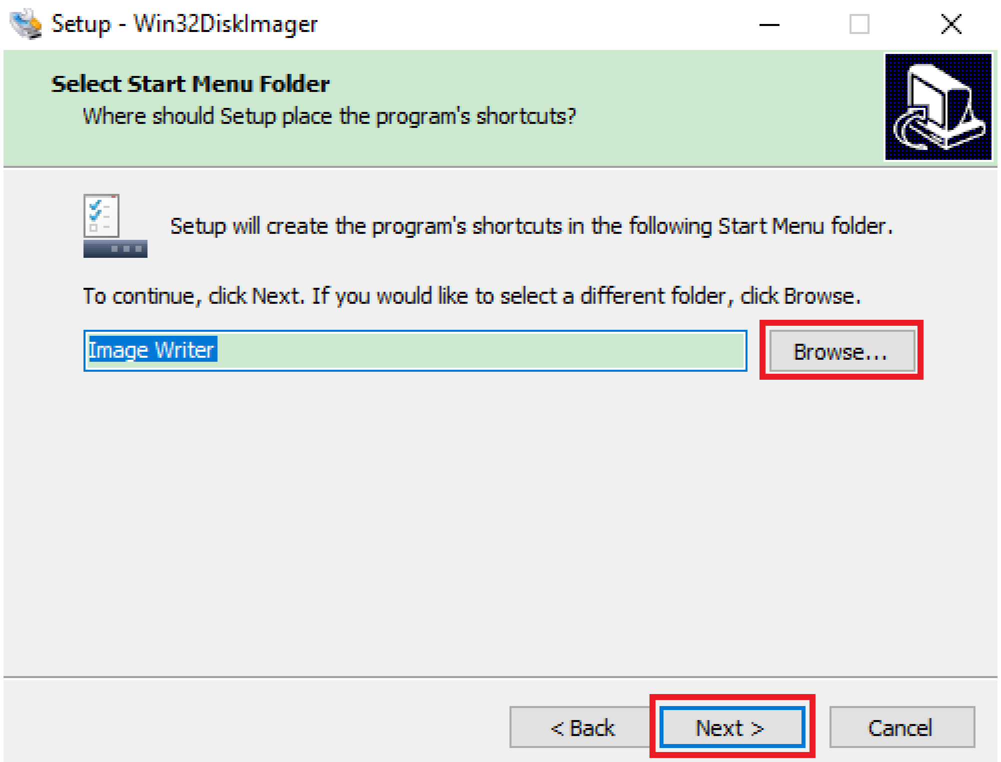
and click "Next".



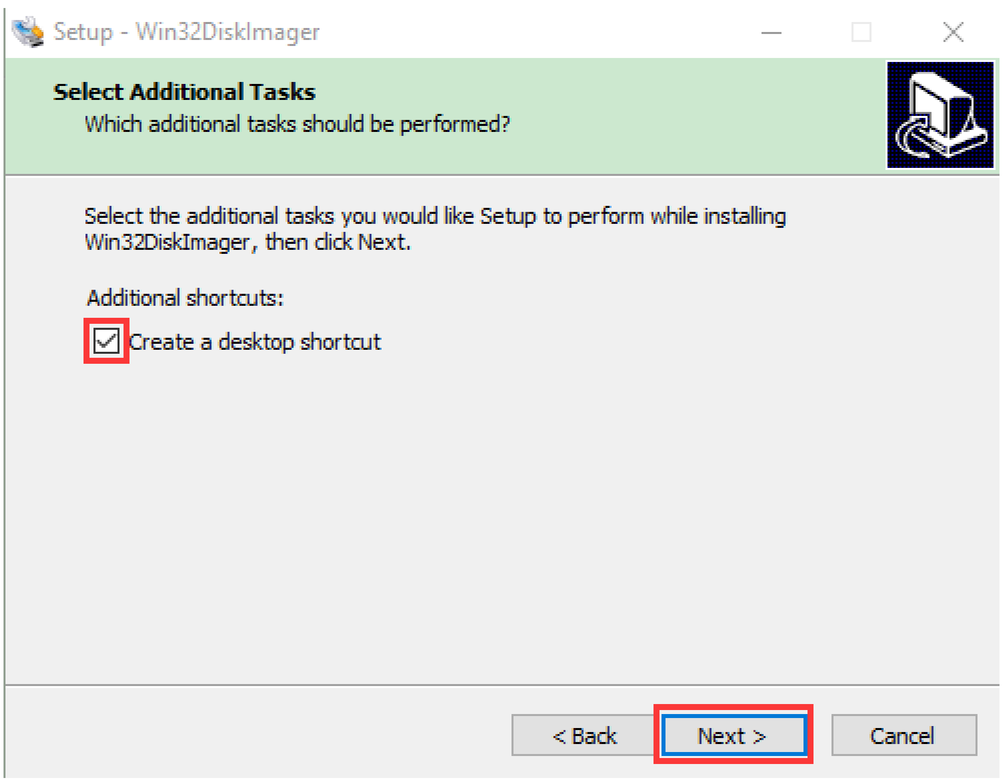
Click "Browse…"select the location where Win32DiskImager is installed and click "Next".

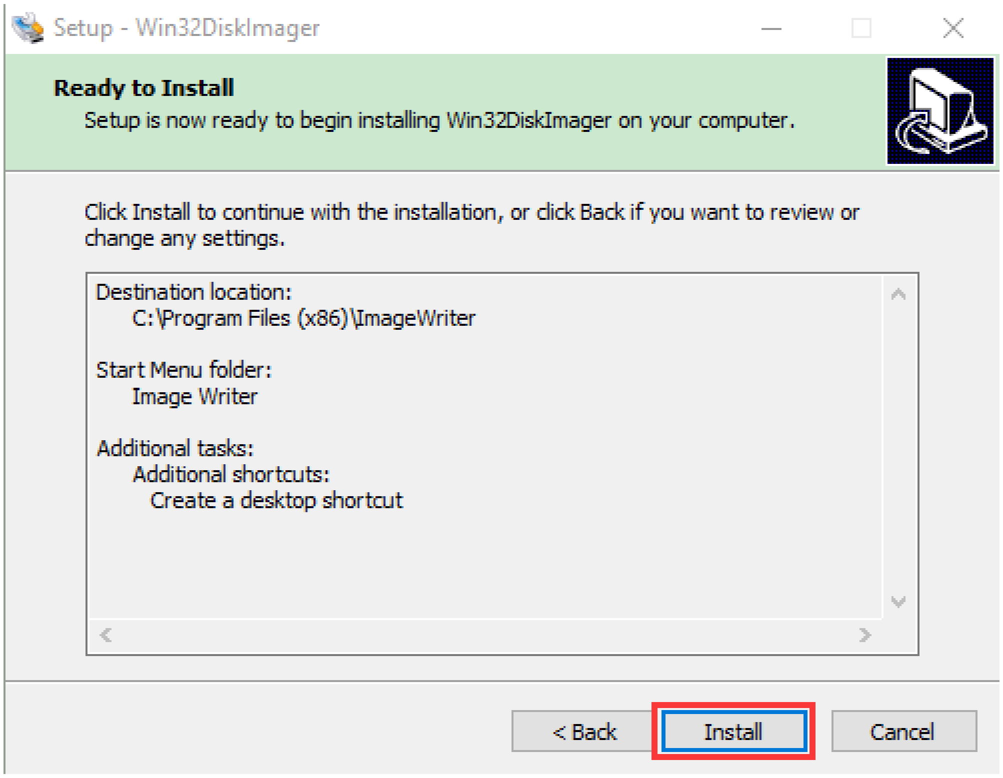


1. Click "Browse…"select the location where Win32DiskImager is installed and click"Next".

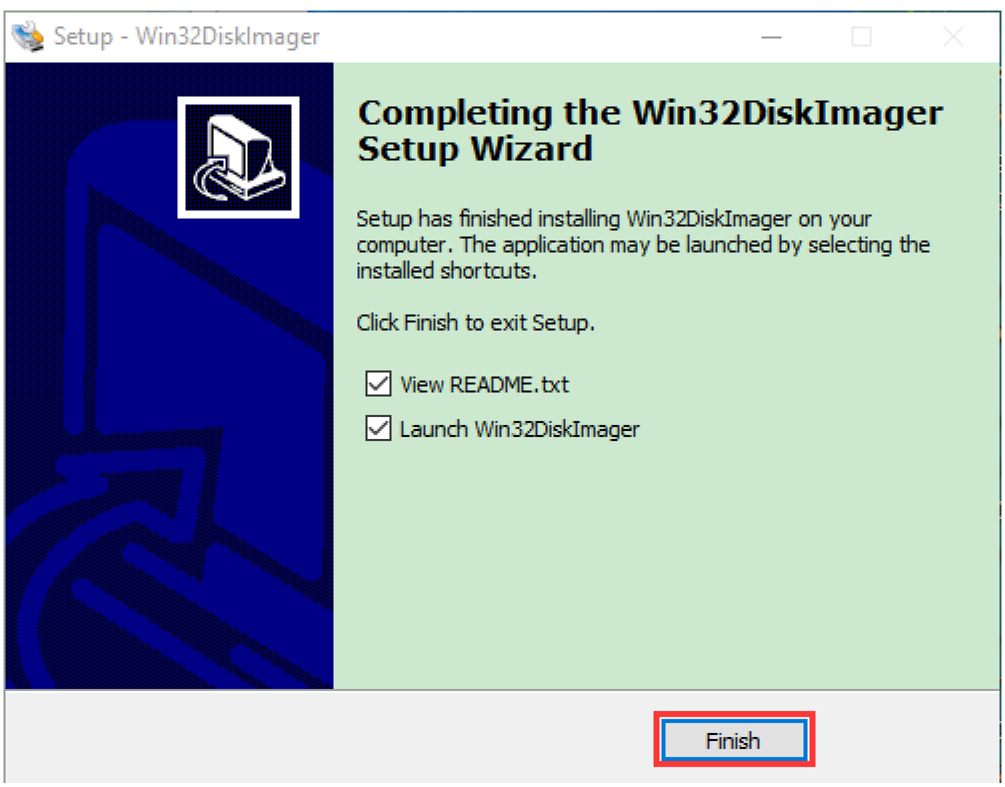


2. Select Create a desktop shortcut and click"Next"and then click"Install".





d. 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

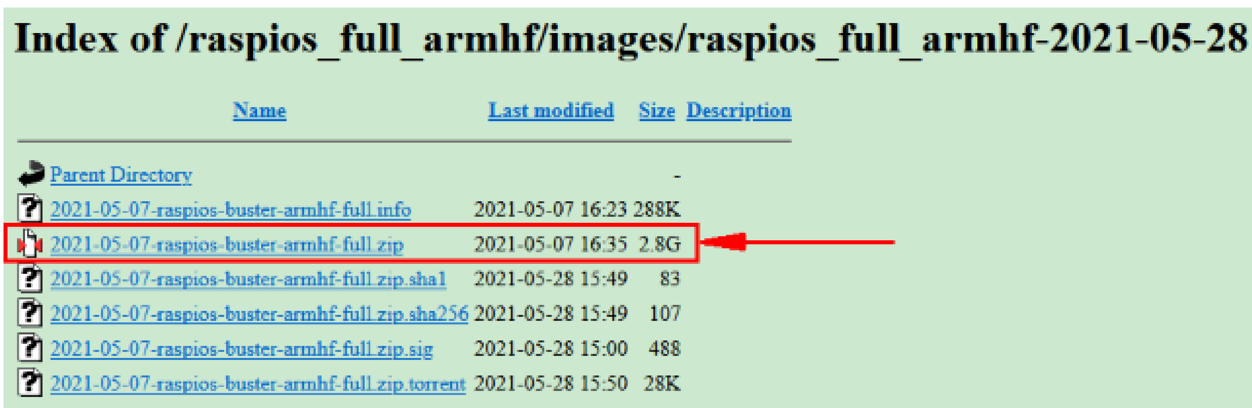
https://www.raspberrypi.org/downloads/raspberry-pi-os/

Download link for the old version

- Raspbian
- https://downloads.raspberrypi.org/raspbian/images/
- Raspbian full
- https://downloads.raspberrypi.org/raspbian_full/images/
- Raspbian lite
- https://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.)

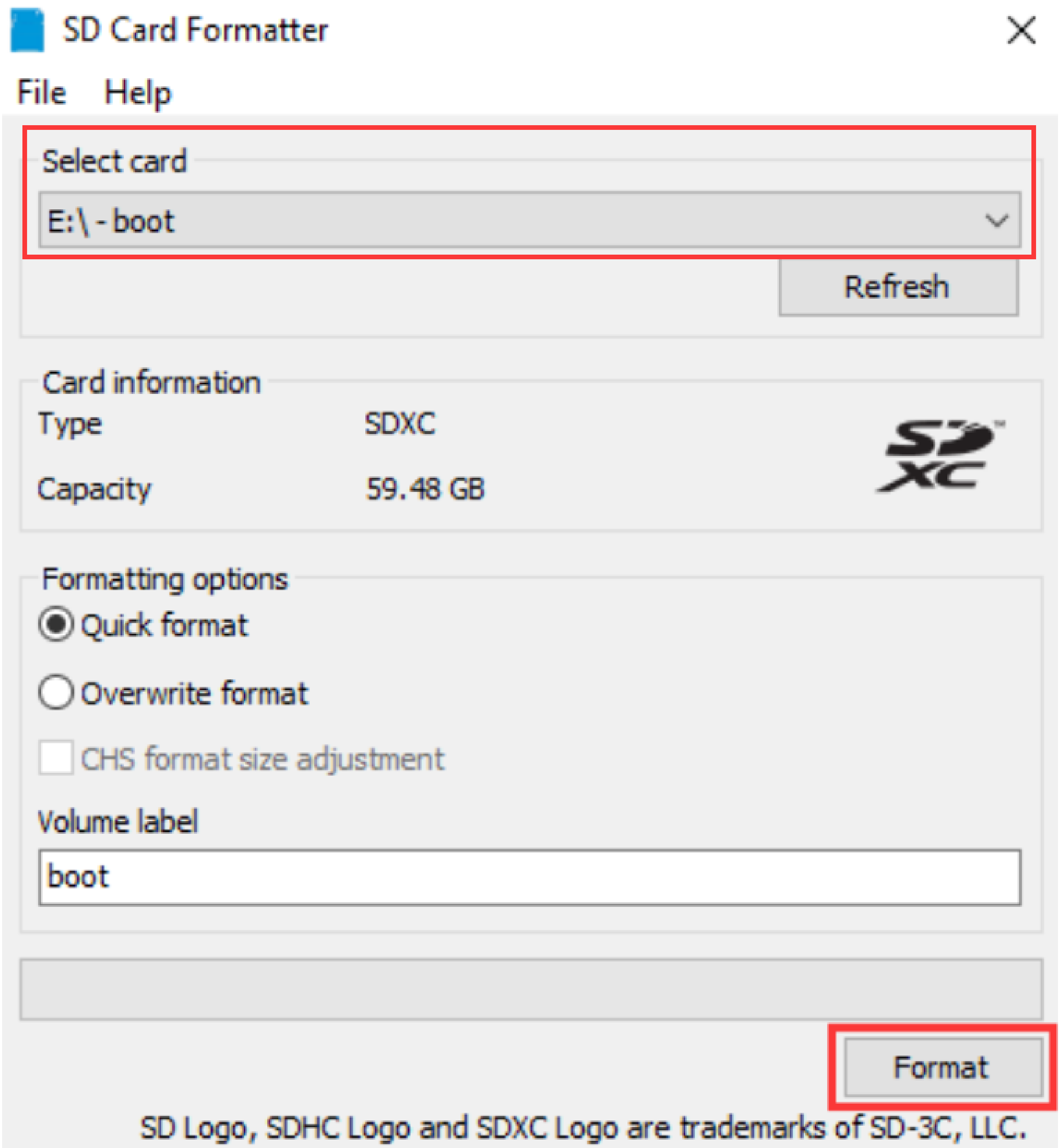
https://downloads.raspberrypi.org/raspios_full_armhf/images/raspios_full_armhf-2021-05-28/

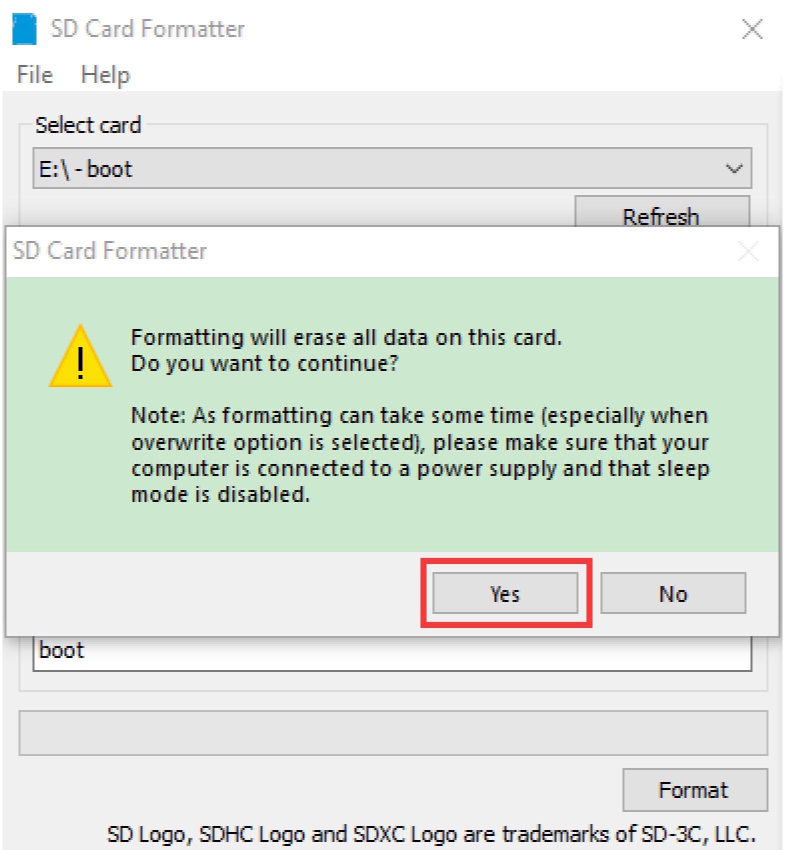


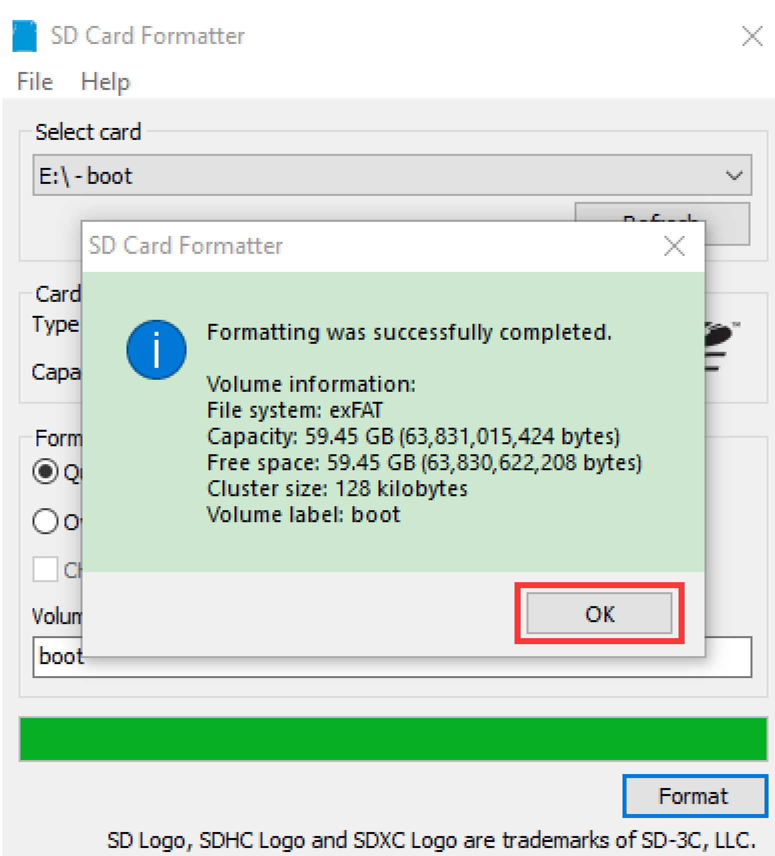
8.1.2 2. Install Raspberry Pi OS system on Raspberry Pi 4B:

2.1.Connect the TFT memory card to 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

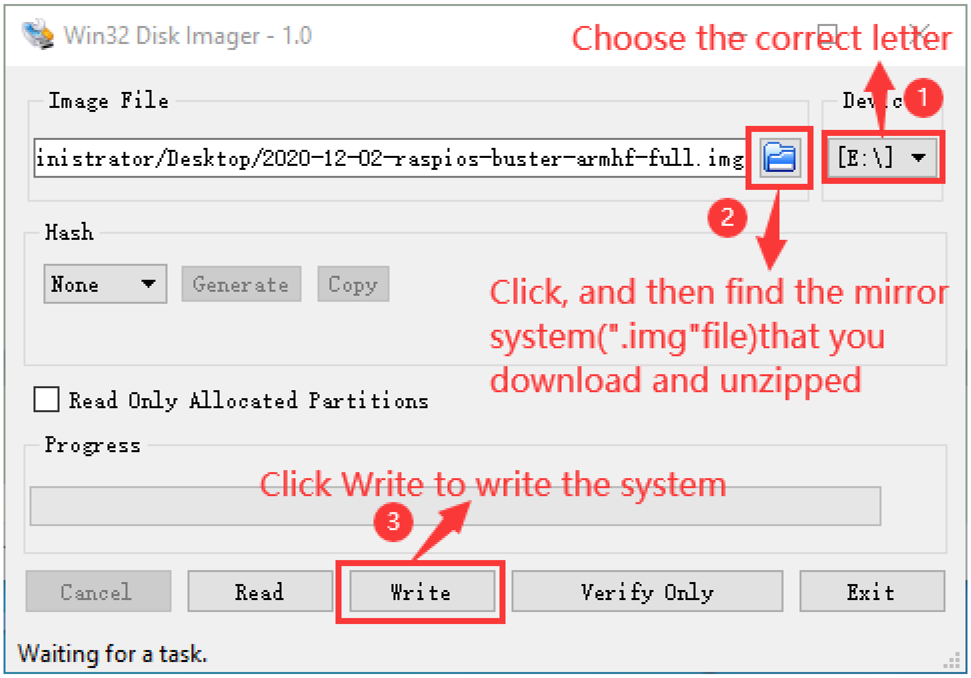


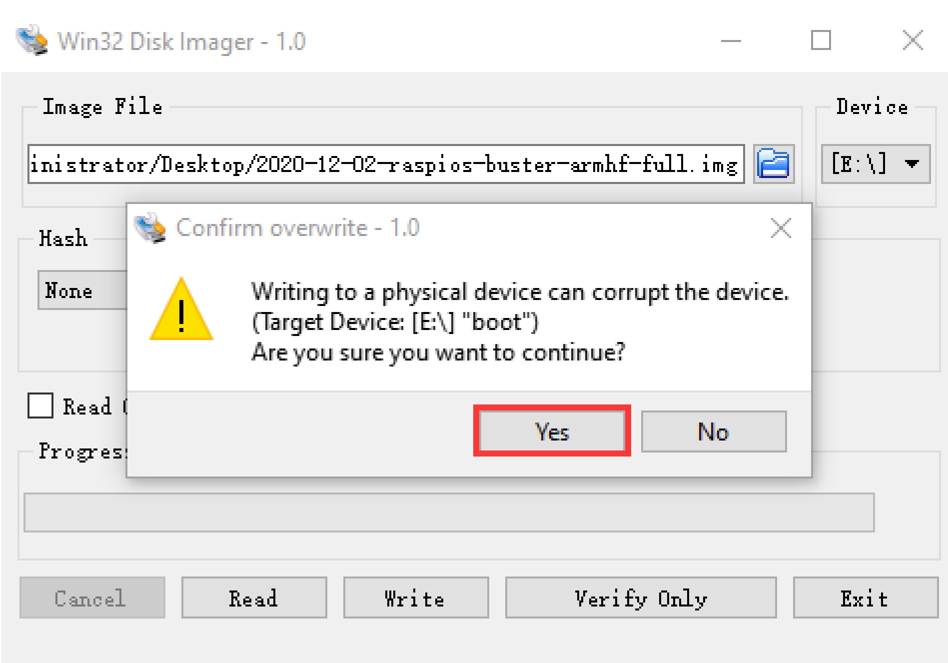

SD Card Formatter
File Help
Select card
E:\ - boot
Refresh
SD Card Formatter
Formatting will erase all data on this card.
Do you want to continue?
Note: As formatting can take some time (especially when overwrite option is selected), please make sure that your computer is connected to a power supply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

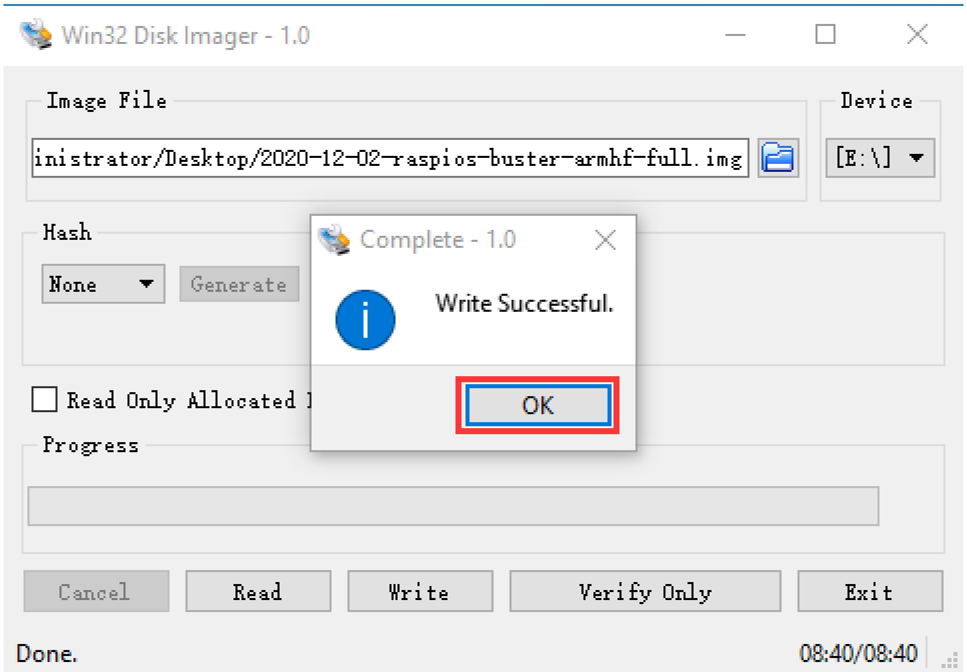

SD Card Formatter
File Help
Select card
E:\ - boot
SD Card Formatter
Formatting was successfully completed.
Volume information:
File system: exFAT
Capacity: 59.45 GB (63,831,015,424 bytes)
Free space: 59.45 GB (63,830,622,208 bytes)
Cluster size: 128 kilobytes
Volume label: boot
OK
boot
Format
SD Logo,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

2.3.Burn System:

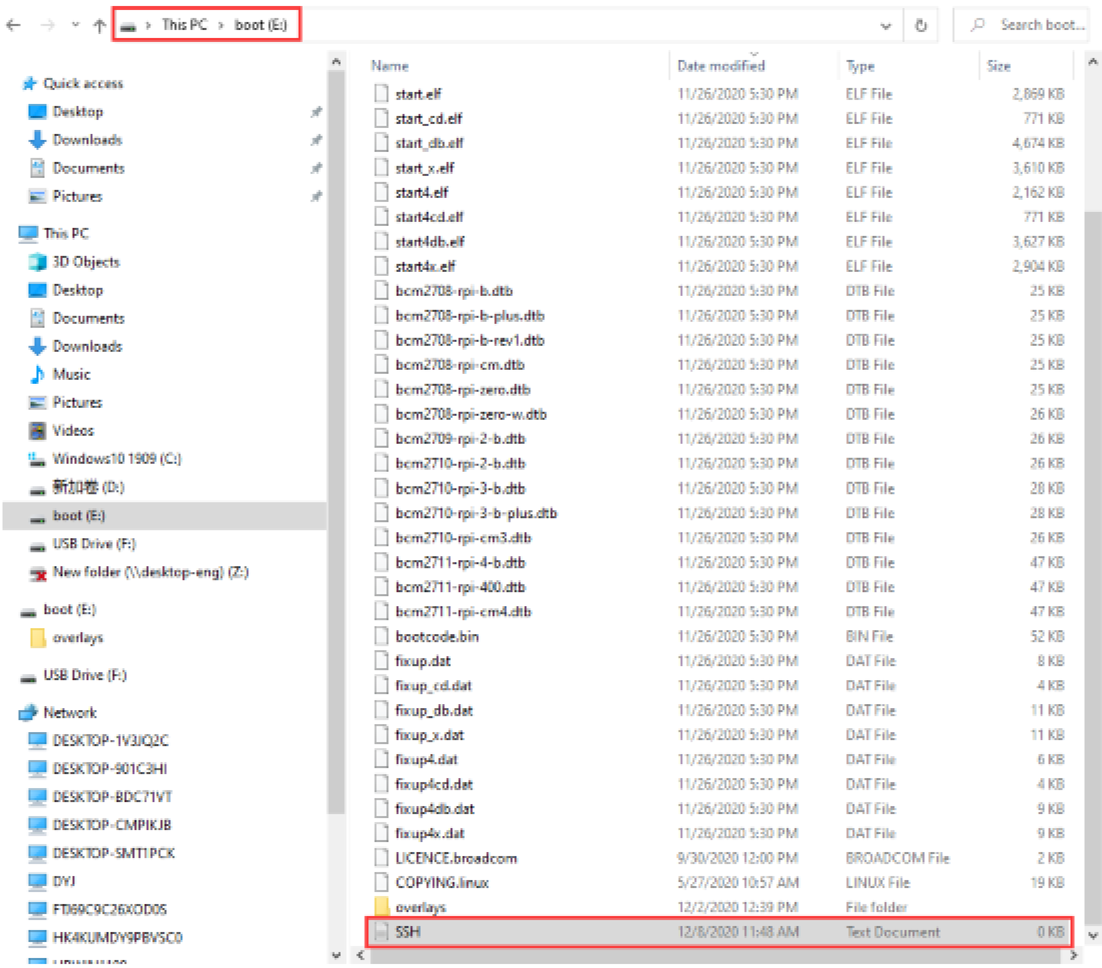
Use **Win32DiskImager** to burn the official **Raspberry Pi OS** mirror to the TFT memory card.







After the mirror system is burned, don't pull out the card reader, use a notepad to create a file named **SSH** and delete**.txt**, then copy it to the boot directory of the TFT card, so that you can open the SSH login function, as shown in the following figure:



Pull out the card reader.

2.4.Log in system:

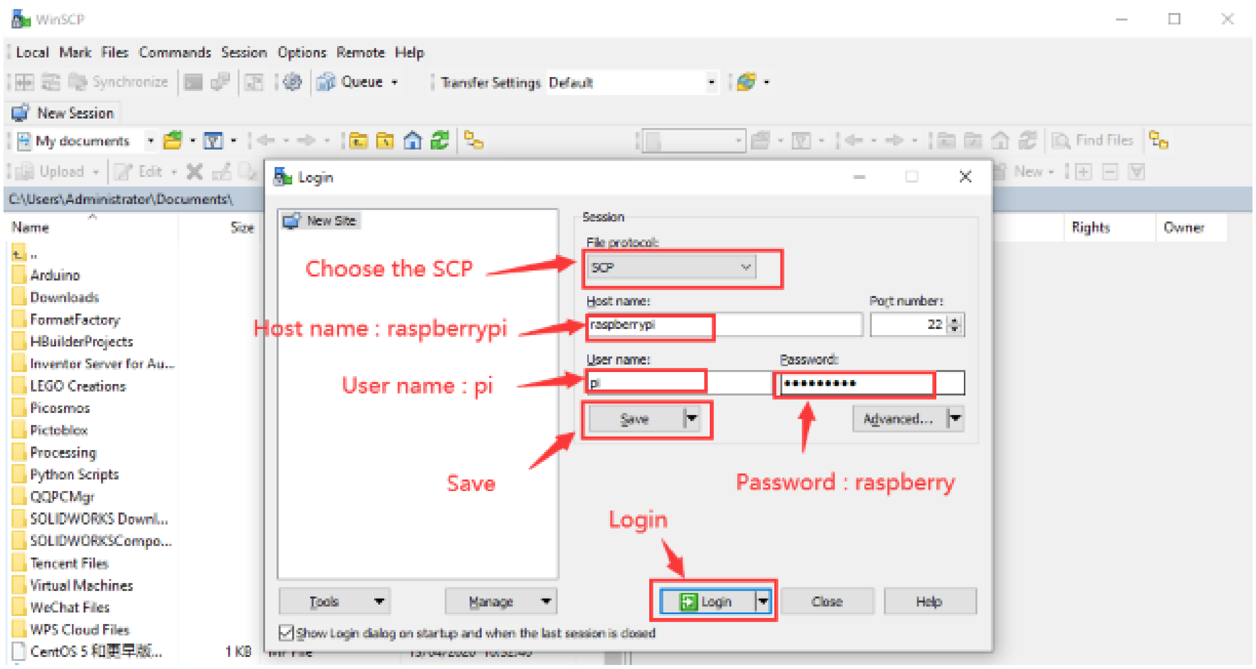
The following operations require raspberry to share the same LOCAL area network with the PC.

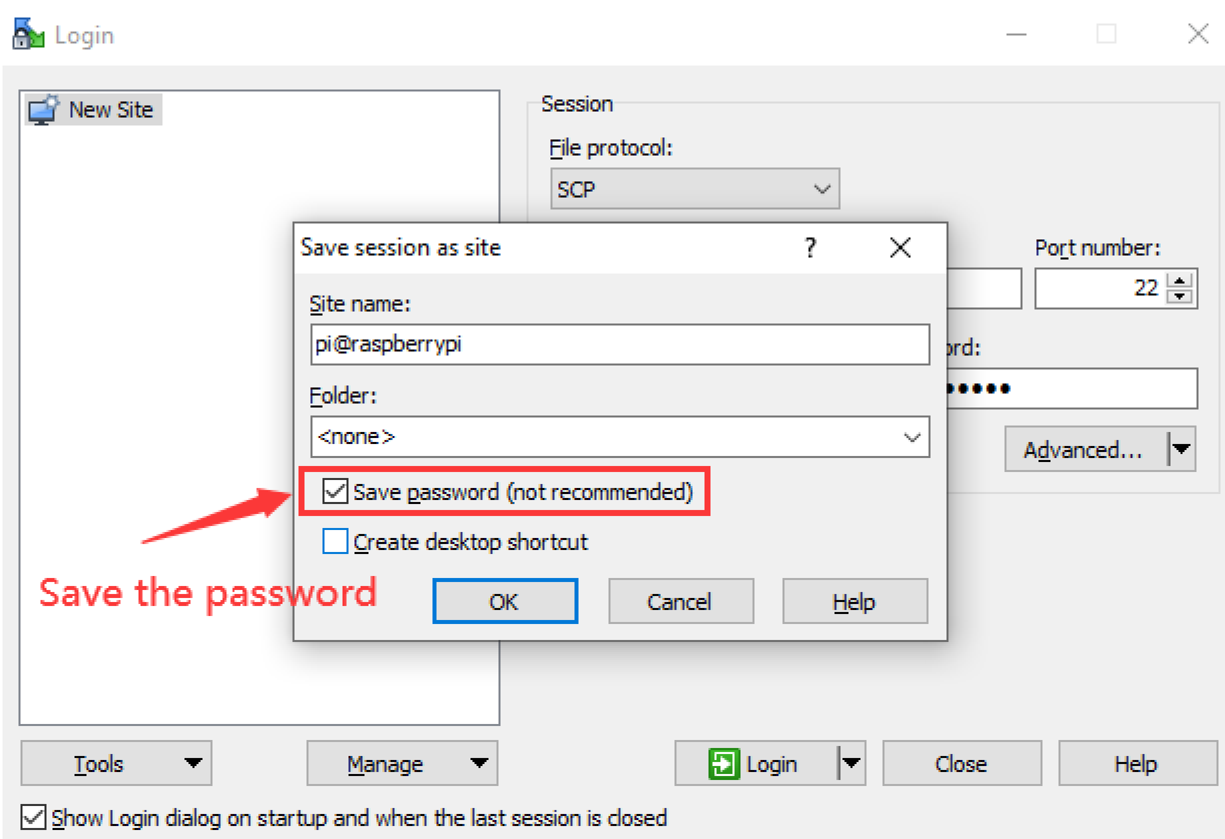
Insert the burned TFT memory card into the Raspberry Pi, connect internet cables and plug in power. If there is a screen and a HDMI cable of Raspberry Pi, connect the screen, and you can see the Raspberry Pi OS startup screen. If there is not a HDMI cable of Raspberry Pi, you can enter the desktop of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

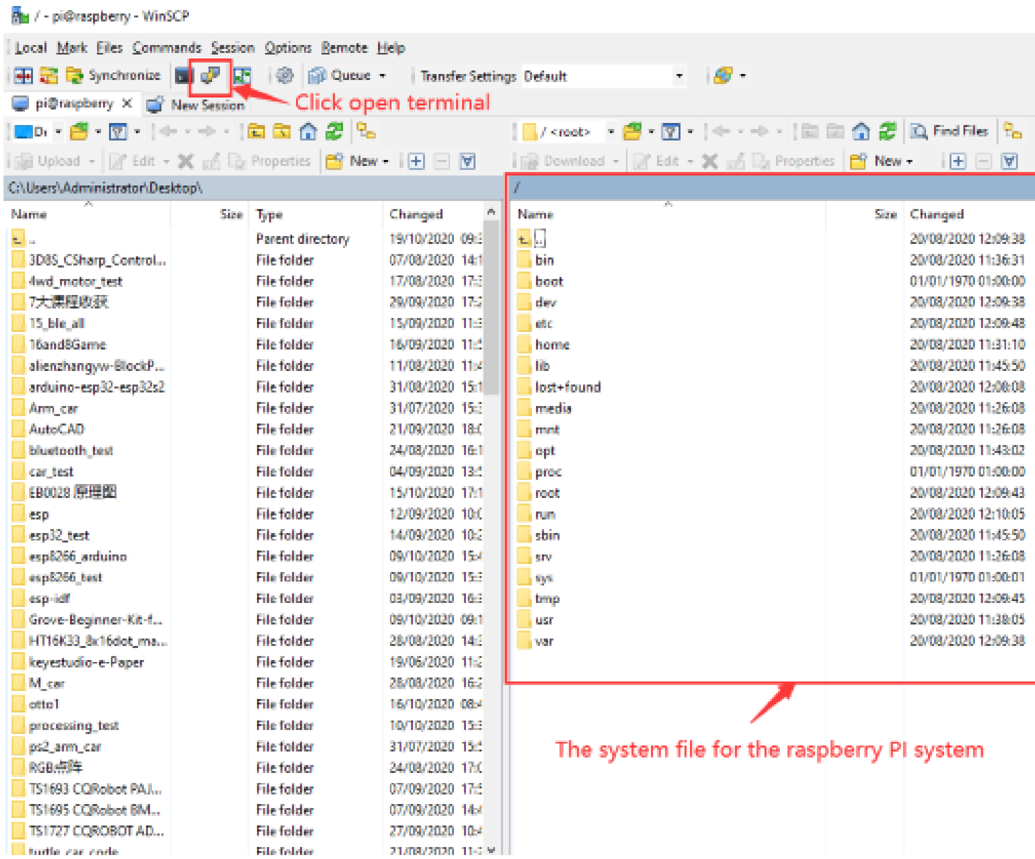
Remote login

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

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

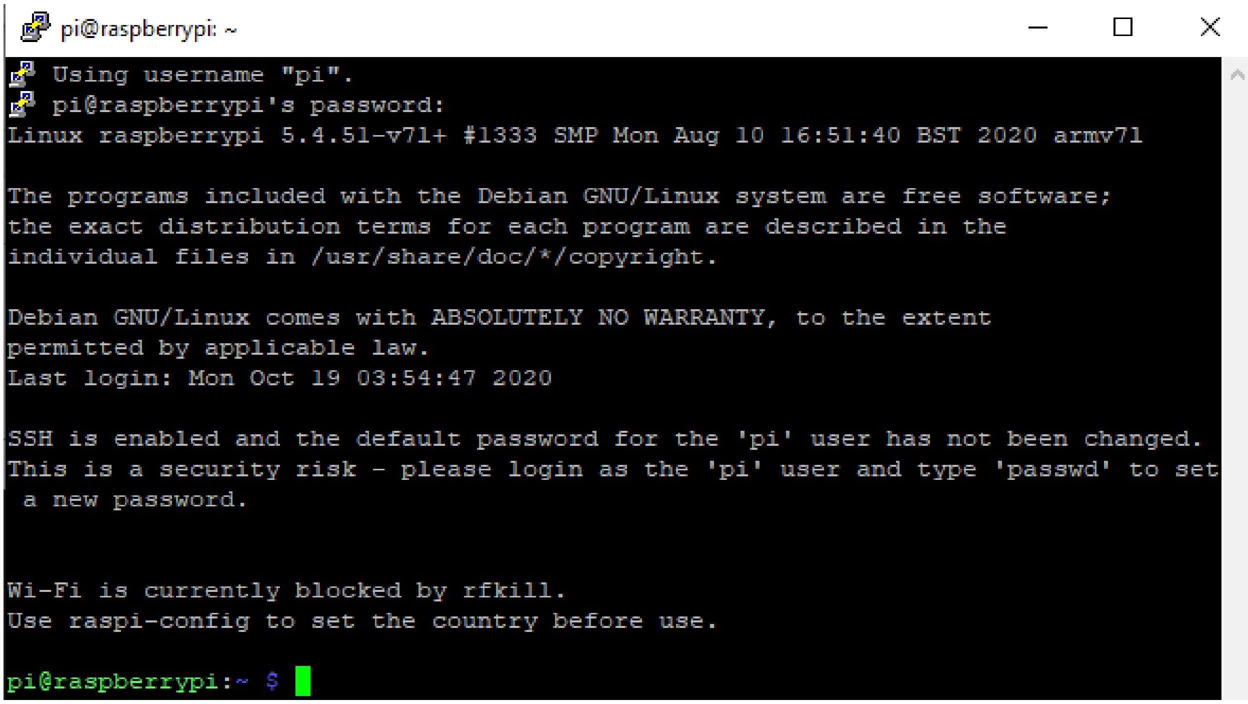

WinSCP
Local Mark Files Commands Session Options Remote Help
Login
New Site
Session
File protocol:
SCP
Host name:
raspberrypi
Port number:
22
User name:
pi
Password:
Save
Advanced...
Choose the SCP
Host name : raspberrypi
User name : pi
Save
Password : raspberry
Login
Tools
Manage
Login
Close
Help
Show Login dialog on startup and when the last session is closed


Login
New Site
Session
File protocol:
SCP
Port number:
22
Advanced...
Save session as site
Site name:
pi@raspberrypi
Folder:
<none>
Save password (not recommended)
Create desktop shortcut
OK
Cancel
Help
Save the password
Tools
Manage
Login
Close
Help
Show Login dialog on startup and when the last session is closed

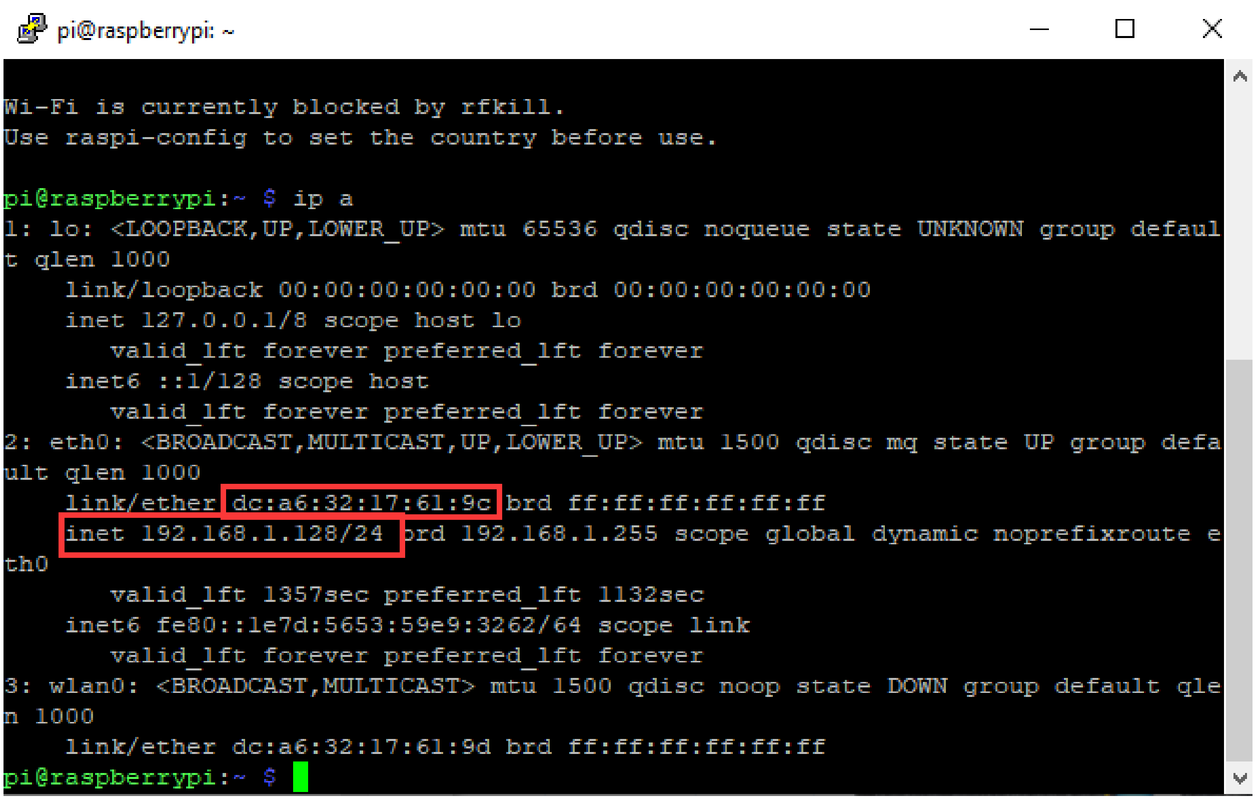


View the ip address and mac address

Click to open terminal and input the password: raspberry, and tap"Enter"on keyboard.



After successfully 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.

Fix 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

```
pi@raspberrypi:~ $ su root
Password:
root@raspberrypi:/home/pi# echo -e '
> auto eth0
> iface eth0 inet static
>     #Change IP address
>     address 192.168.1.99
>     netmask 255.255.255.0
>     gateway 192.168.1.1
>     network   192.168.1.0
>     broadcast 192.168.1.255
>     dns-domain 119.29.29.29
>     dns-nameservers 119.29.29.29
>     metric 0
> mtu 1492
> '>/etc/network/interfaces.d/eth0
root@raspberrypi:/home/pi#
```

Reboot the system to activate the configuration file.

Input the restart command in the terminal: sudo reboot

You could log in via fixed IP afterwards.

Check IP to ensure 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:~ $
```

Log 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

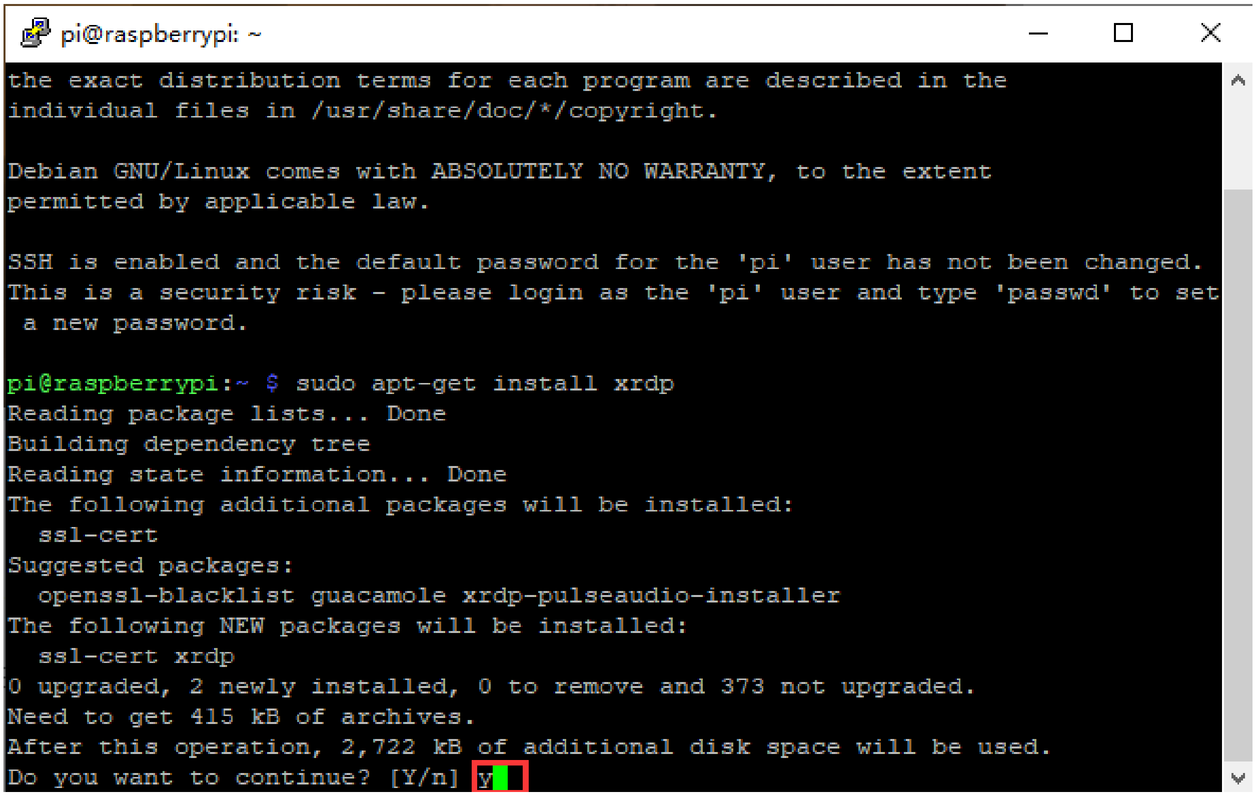
Installation commands:

Switch to root User: su root

Installation commands: apt-get install xrdp

Enter y and tap“**Enter**”keyboard…

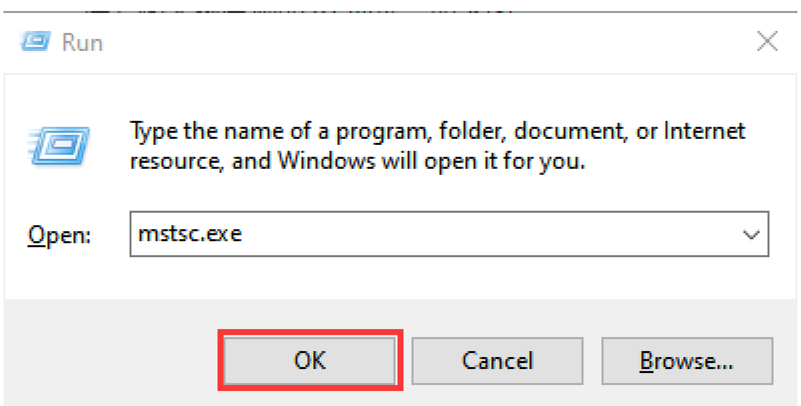
As shown below:



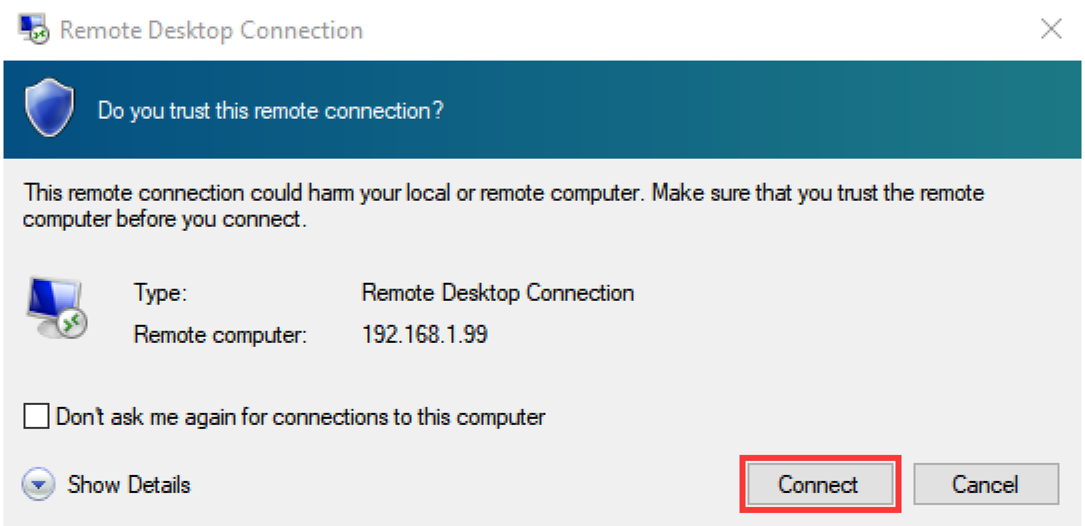
Open the remote desktop connection on Windows

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

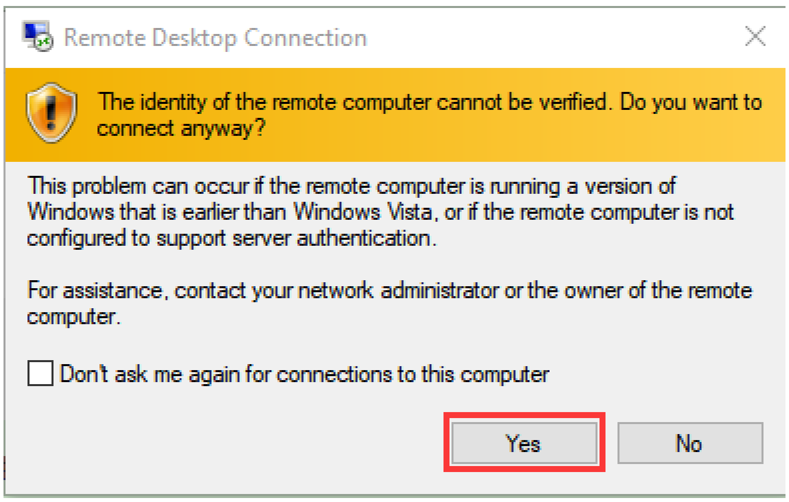
As shown below:



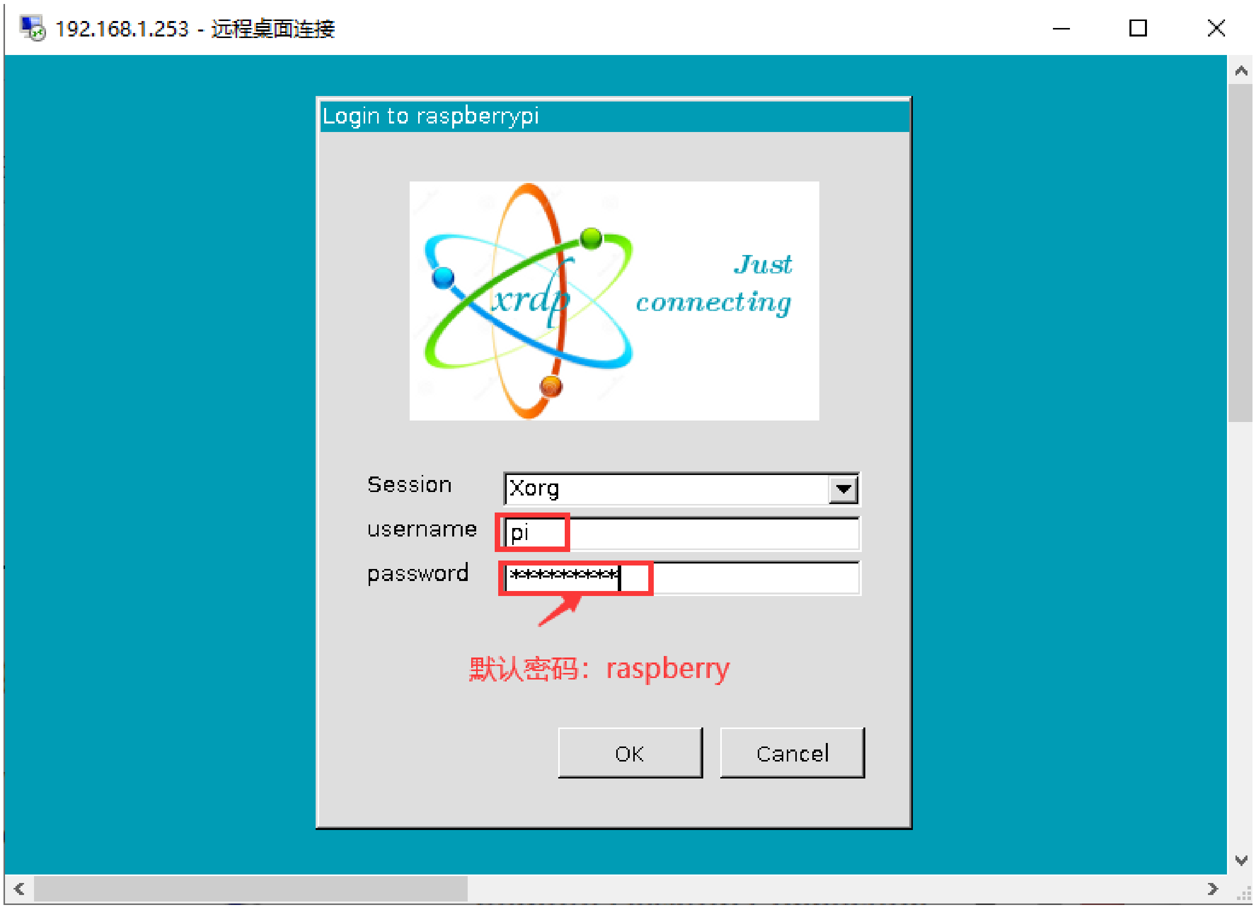
Enter the IP address of the Raspberry Pi, as shown below. Click“Connect” and then click “Connect”again. 192.168.1.99 is the ip address we use, you could change it into your IP address.



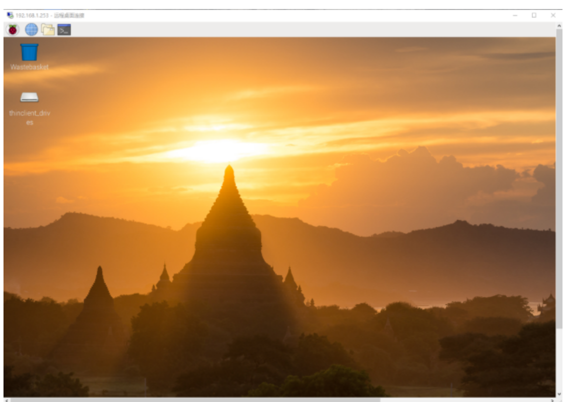
A prompt will appear and you can click“Yes”.



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.

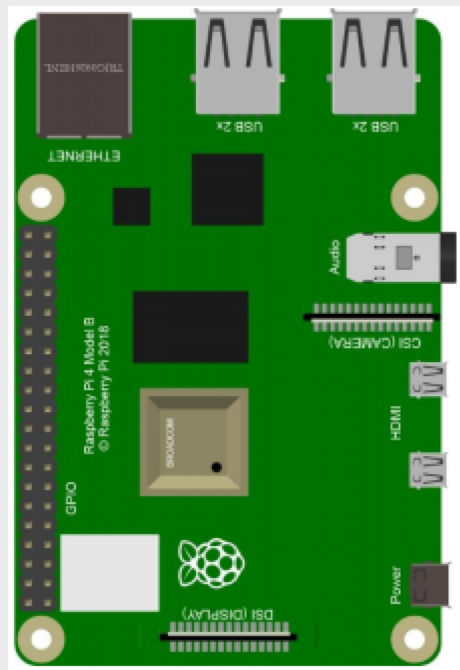
8.2 Preparation of C language control basic hardware:

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.

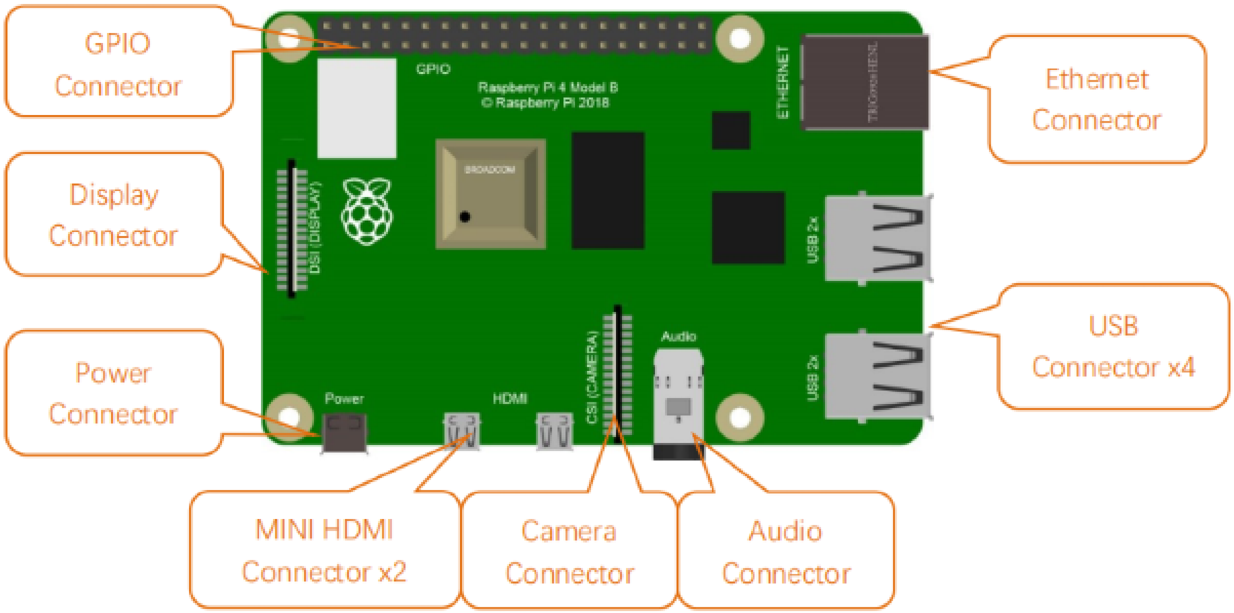
8.2.1 (1)Description of basic raspberry pi accessories

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
	

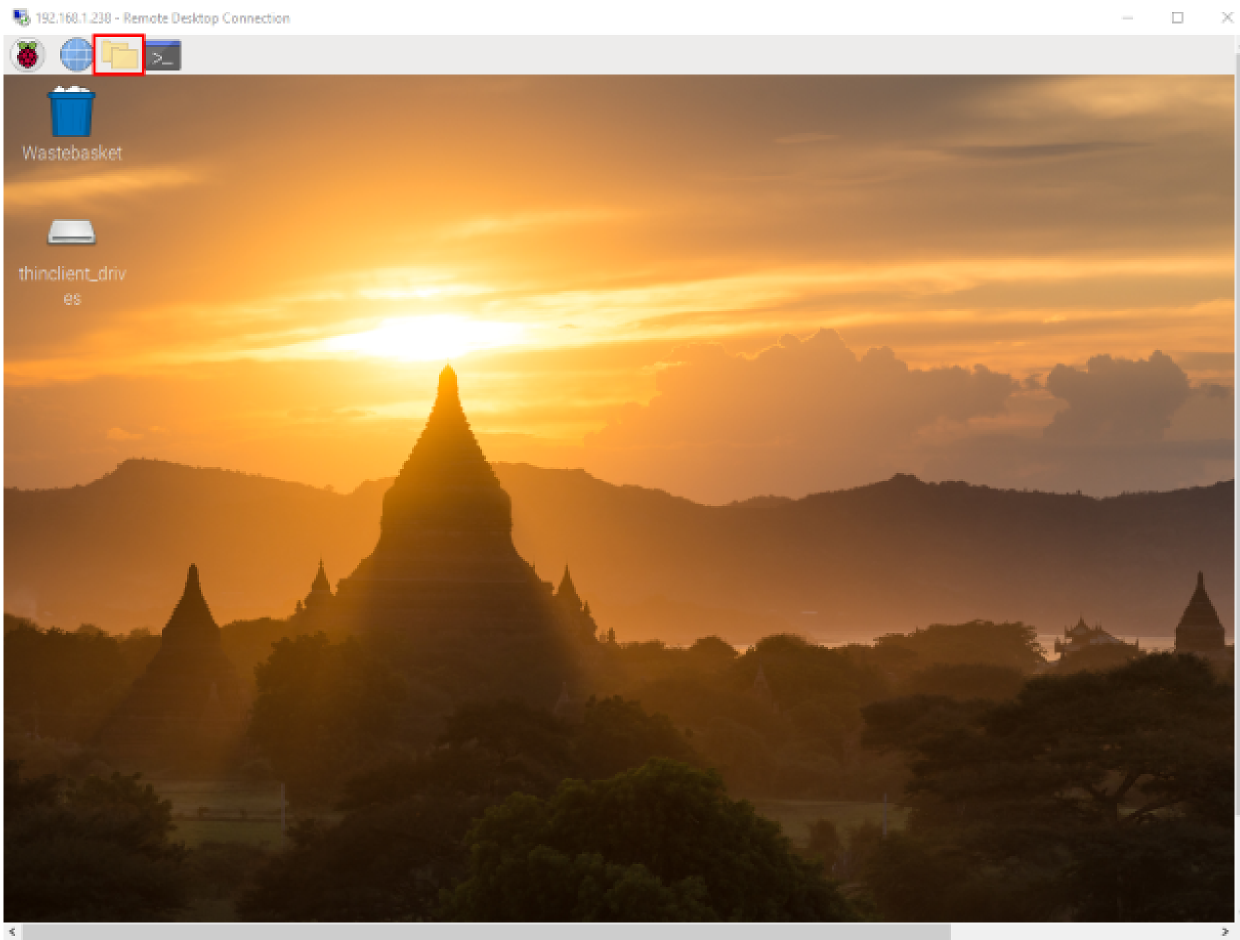
Hardware Interfaces

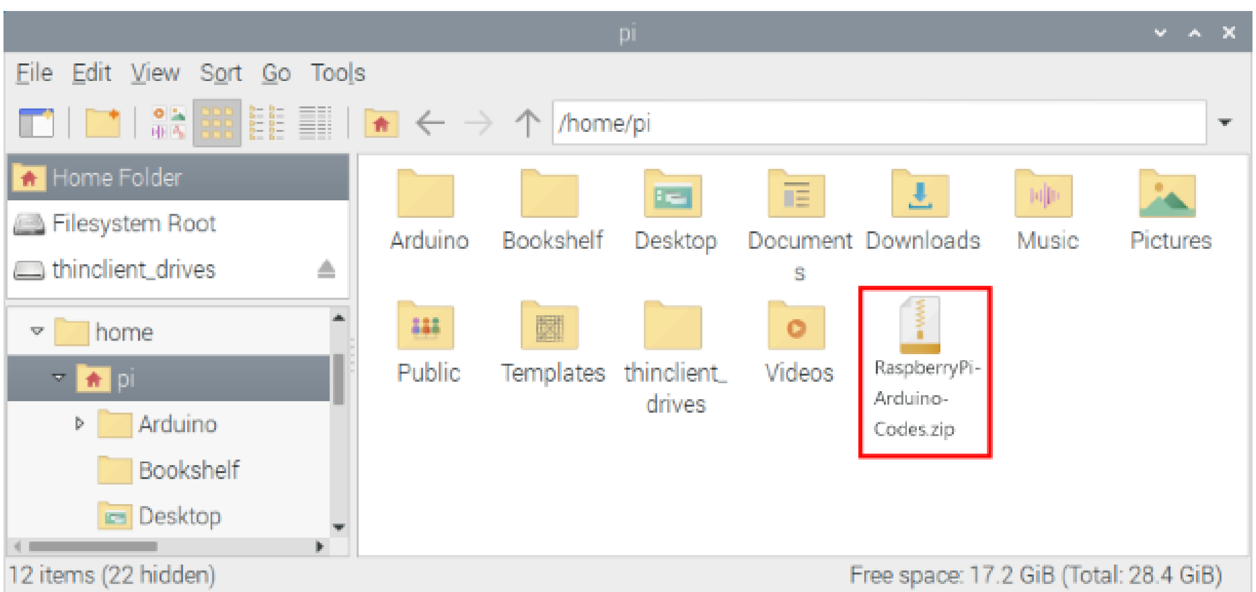


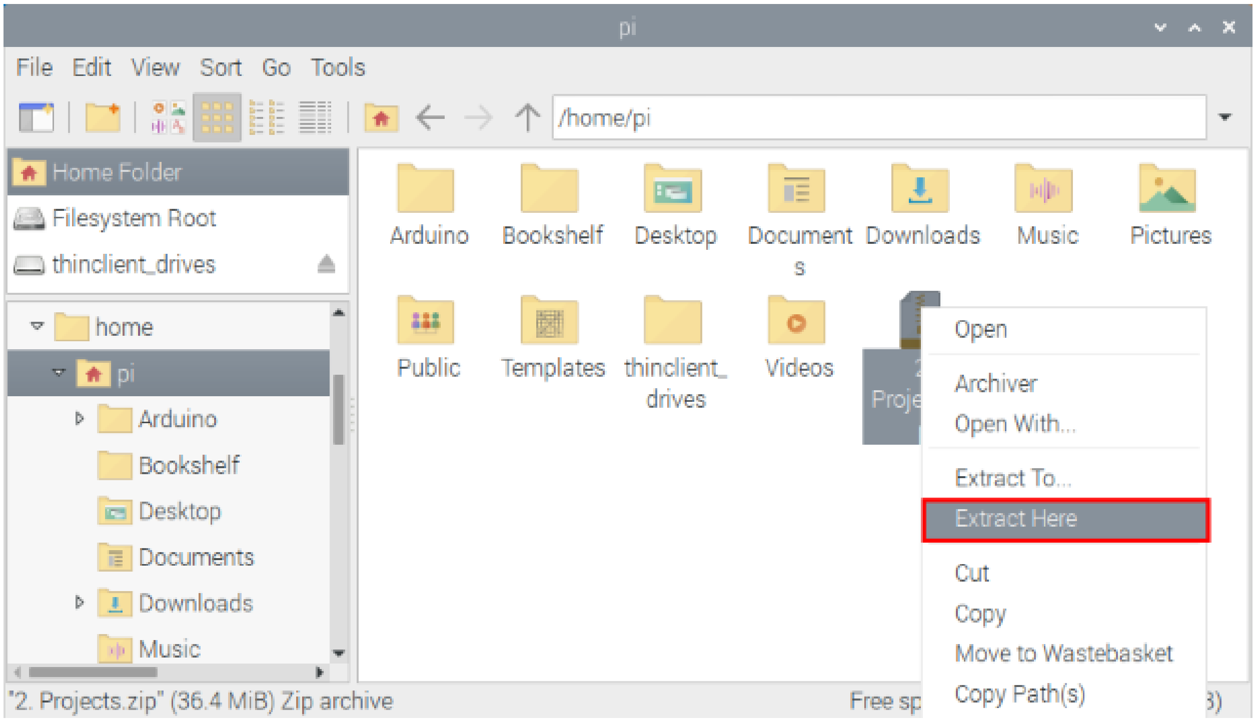
8.2.2 Raspberry Pi +ESP32 main board + breadboard +USB cable, as shown below

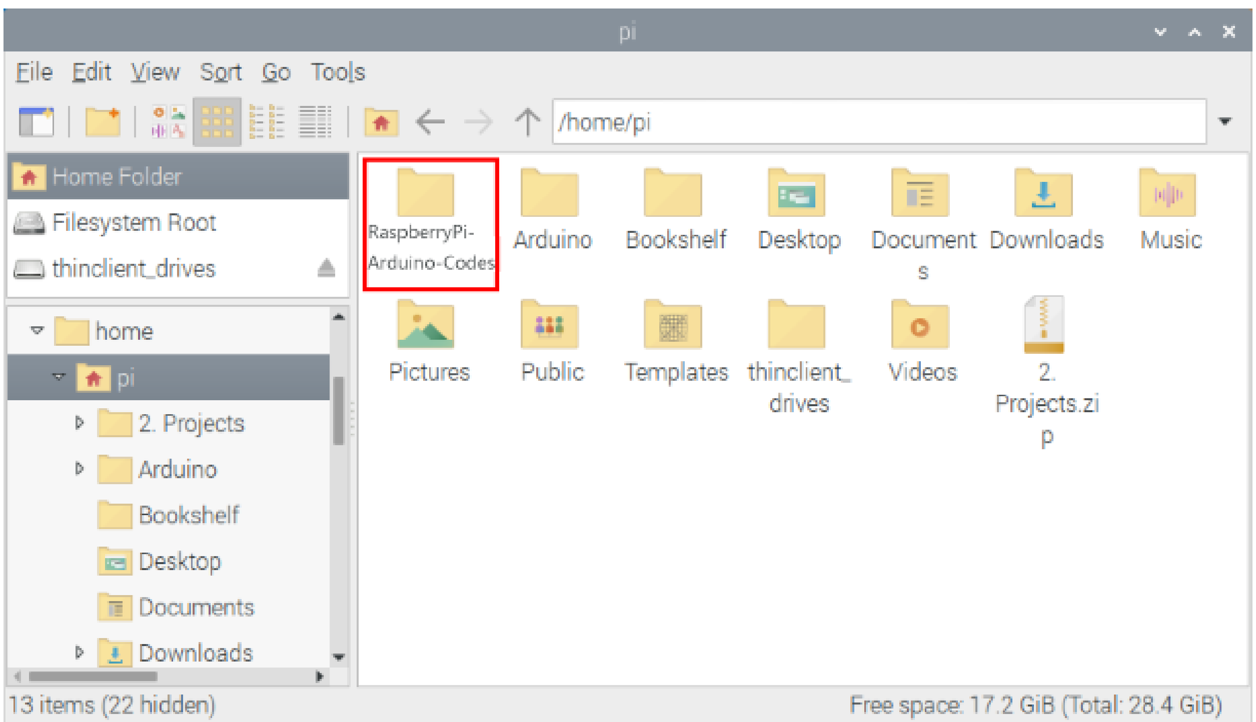
(3)Copy Example Code Folder to Raspberry Pi

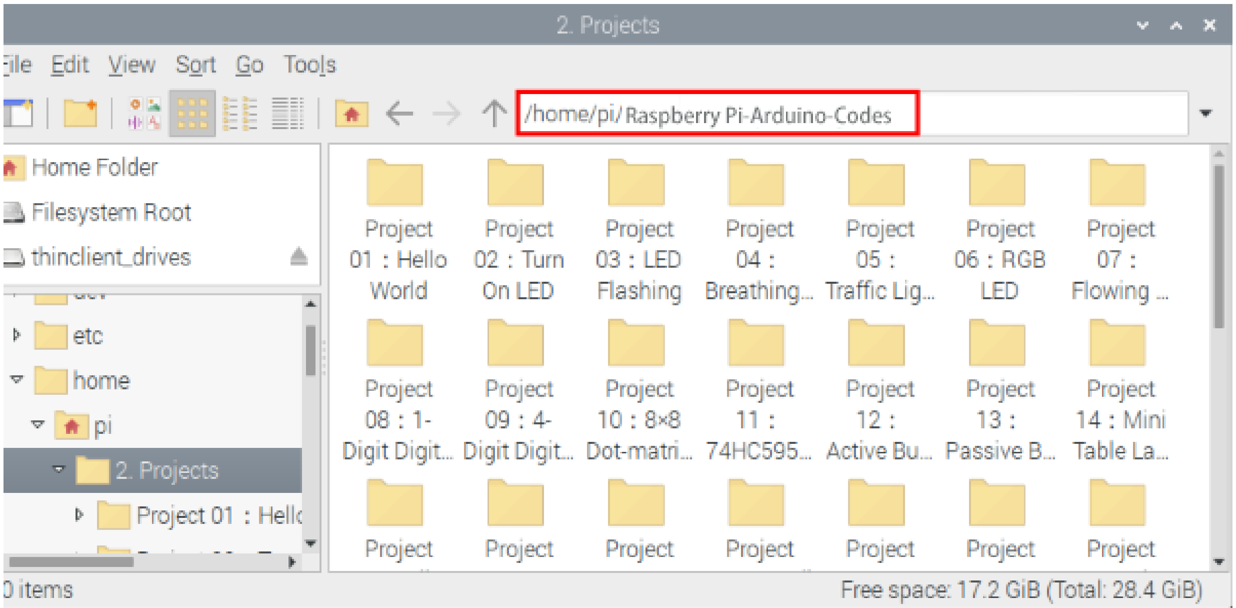
Place example code folder to the pi folder of Raspberry Pi. Just copy and paste the **2. Projects.zip** file (the default is **ZIP** file)that we provided into user pi and unzip it, as shown below:


192.168.1.238 - Remote Desktop Connection
Wastebasket
thinclient_drives


pi
File Edit View Sort Go Tools
/home/pi
Home Folder
Filesystem Root
thinclient_drives
home
pi
Arduino
Bookshelf
Desktop
Arduino
Bookshelf
Desktop
Documents
Downloads
Music
Pictures
Public
Templates
thinclient_drives
Videos
RaspberryPi-Arduino-Codes.zip
12 items (22 hidden)
Free space: 17.2 GiB (Total: 28.4 GiB)


pi
File Edit View Sort Go Tools
/home/pi
Home Folder
Filesystem Root
thinclient_drives
home
pi
Arduino
Bookshelf
Desktop
Documents
Downloads
Music
Arduino
Bookshelf
Desktop
Documents
Downloads
Music
Pictures
Public
Templates
thinclient_drives
Videos
Open
Archiver
Open With...
Extract To...
Extract Here
Cut
Copy
Move to Wastebasket
Copy Path(s)
"2. Projects.zip" (36.4 MiB) Zip archive

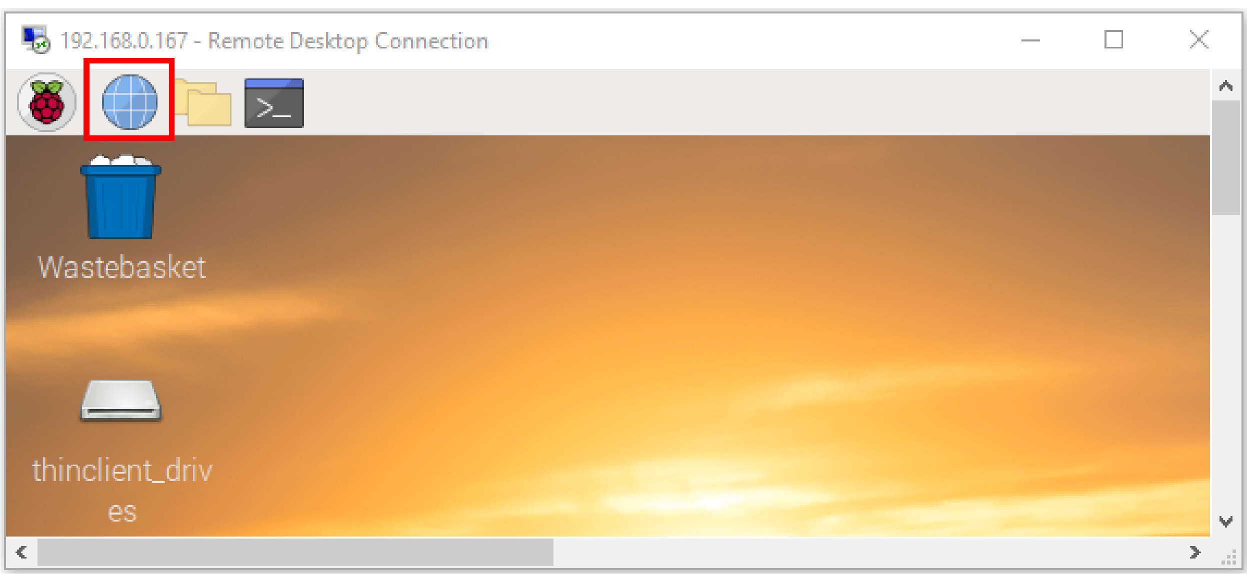

pi
File Edit View Sort Go Tools
/home/pi
Home Folder
Filesystem Root
thinclient_drives
home
pi
2. Projects
Arduino
Bookshelf
Desktop
Documents
Downloads
RaspberryPi-Arduino-Codes
Arduino
Bookshelf
Desktop
Documents
Downloads
Music
Pictures
Public
Templates
thinclient_drives
Videos
2. Projects.zip
13 items (22 hidden)
Free space: 17.2 GiB (Total: 28.4 GiB)



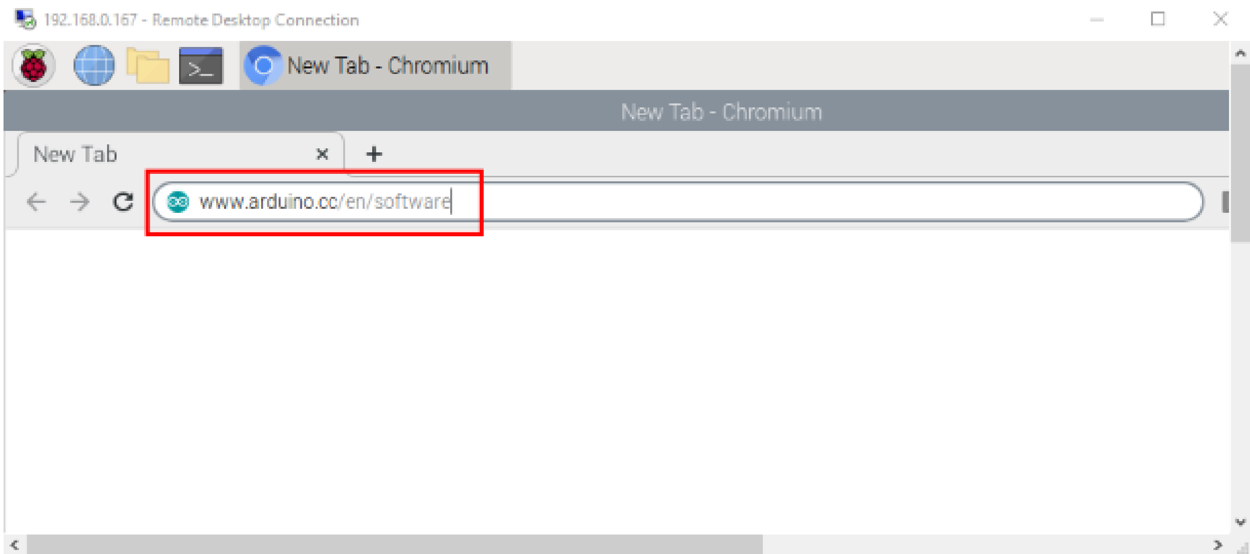
Linux SystemRaspberry Pi

8.2.3 (2) Download and install Arduino IDE

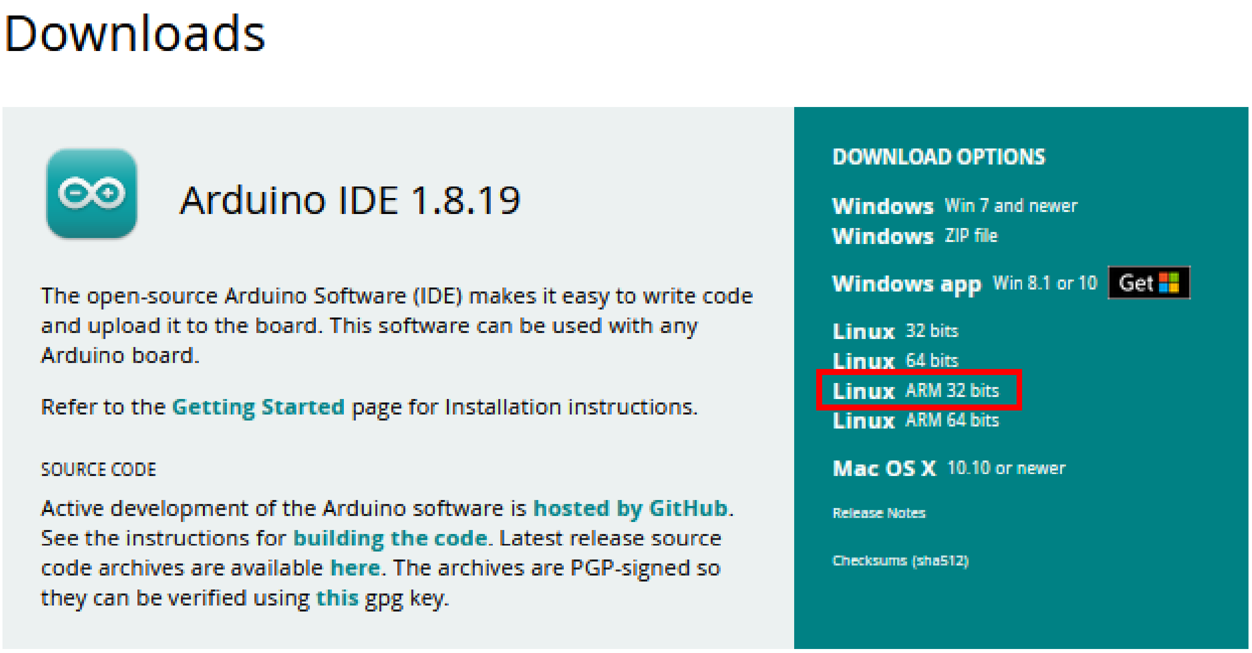
1First, click on Raspberry Pi's browser.



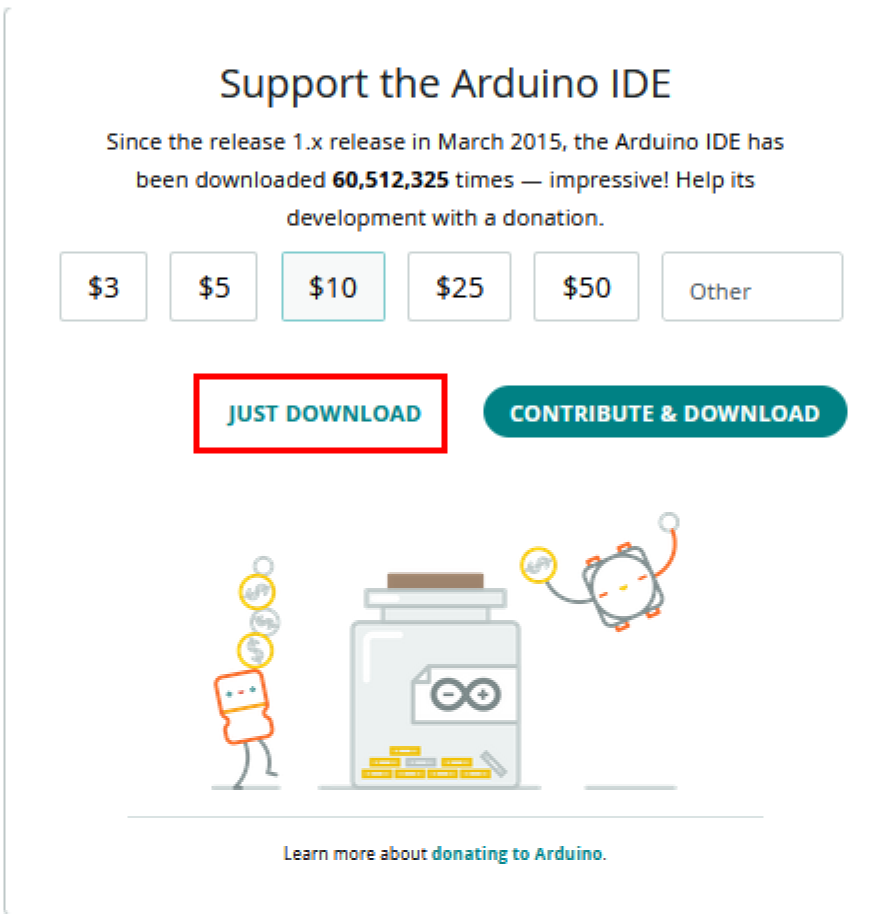
2. Enter the Official Arduino website in your browser: www.arduino.cc/en/software , as shown below:



There 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".

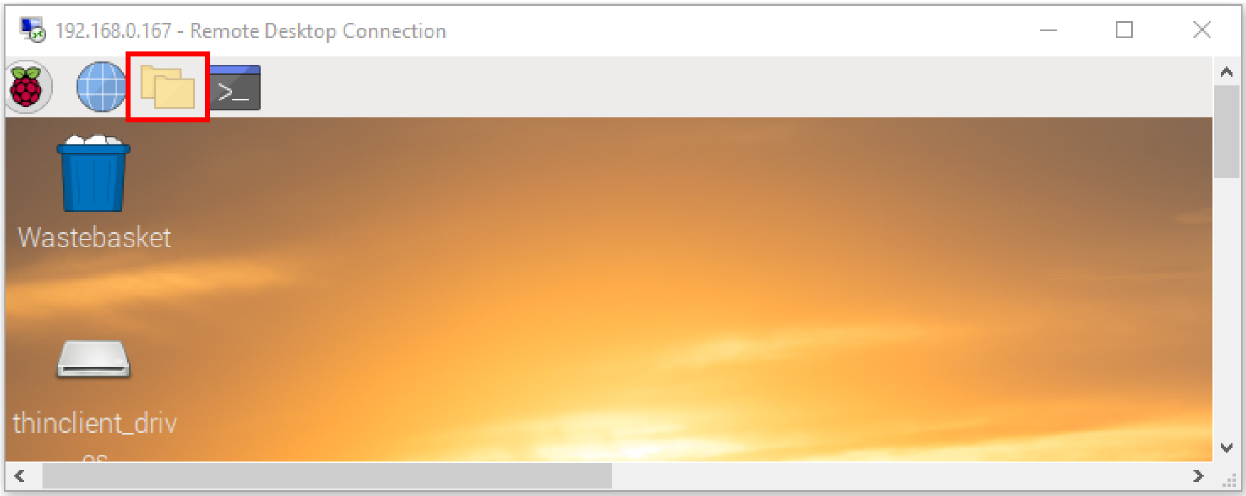


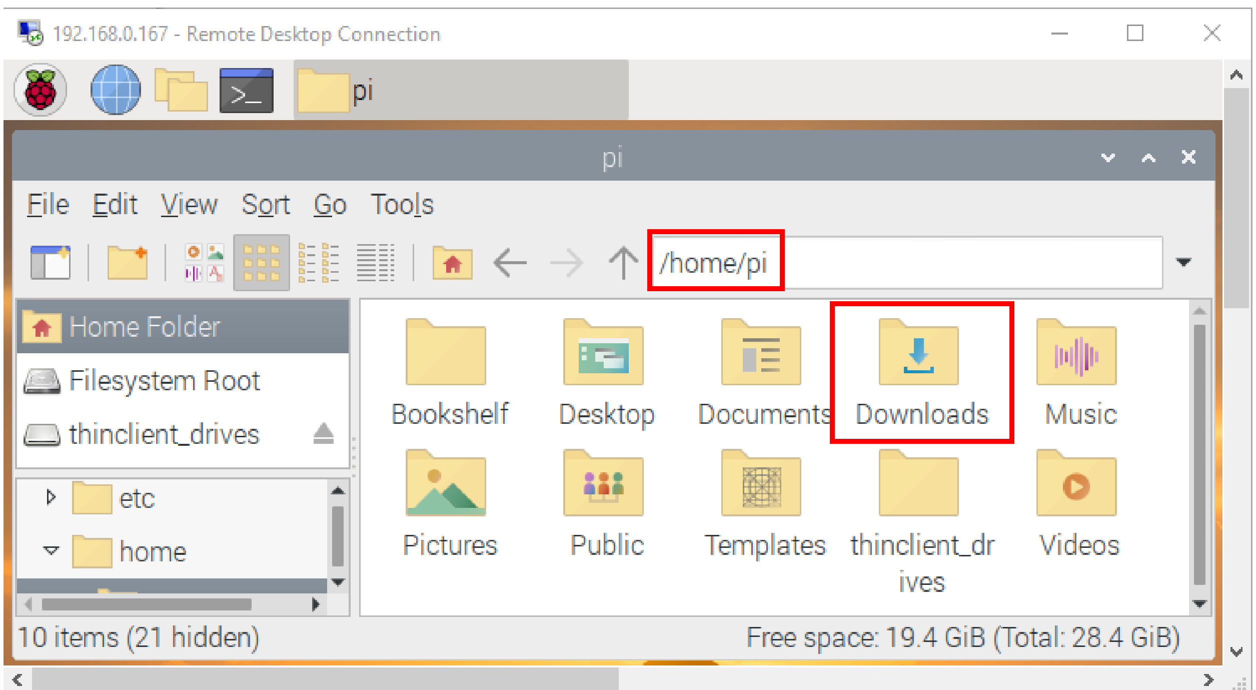
Just click **JUST DOWNLOAD** to download

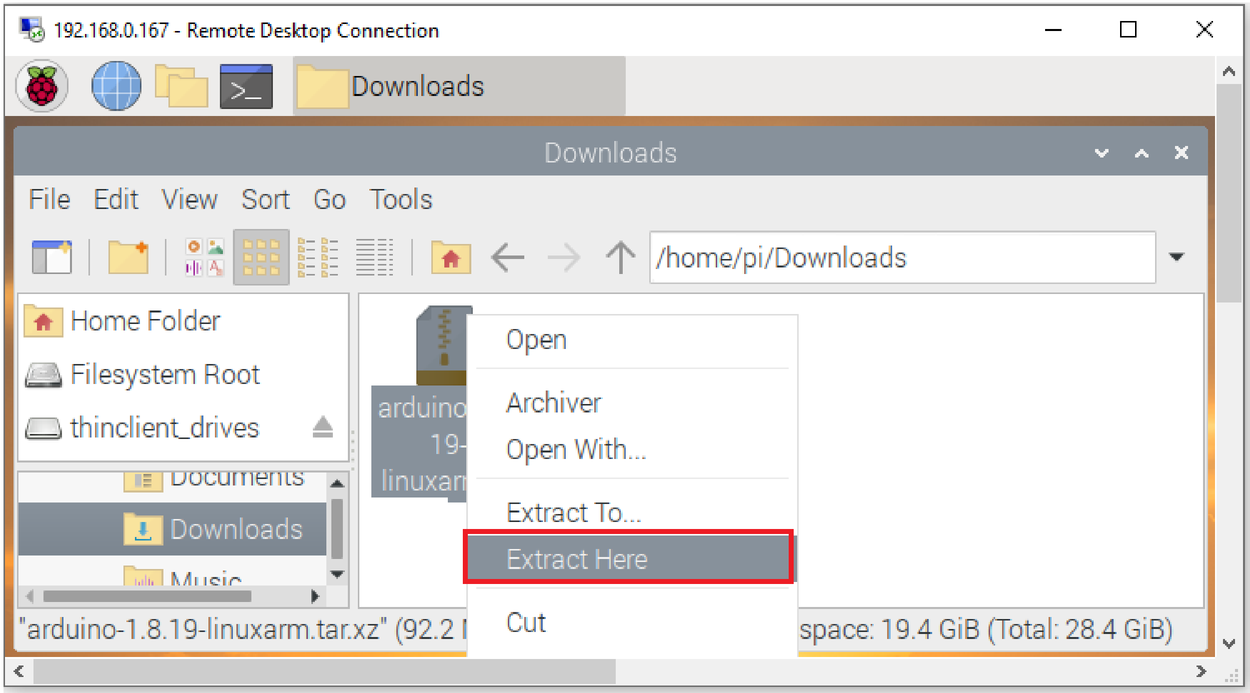


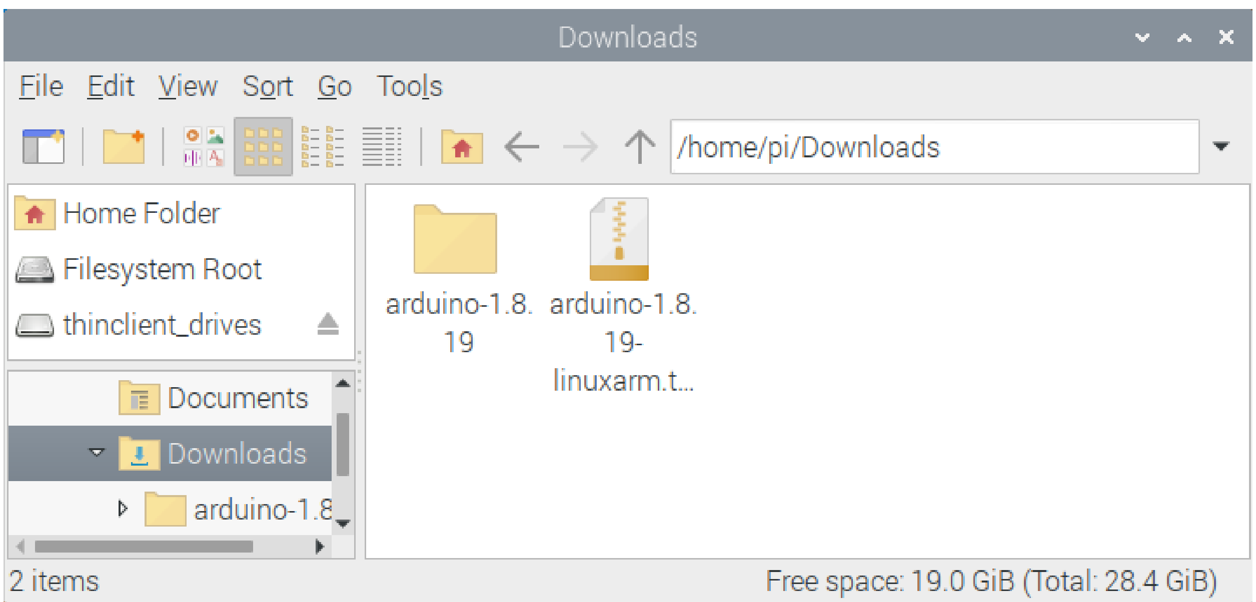
After a few seconds, the lastest Arduino IDEArduino 1.8.19 versionzip file can be directly downloaded.

(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.

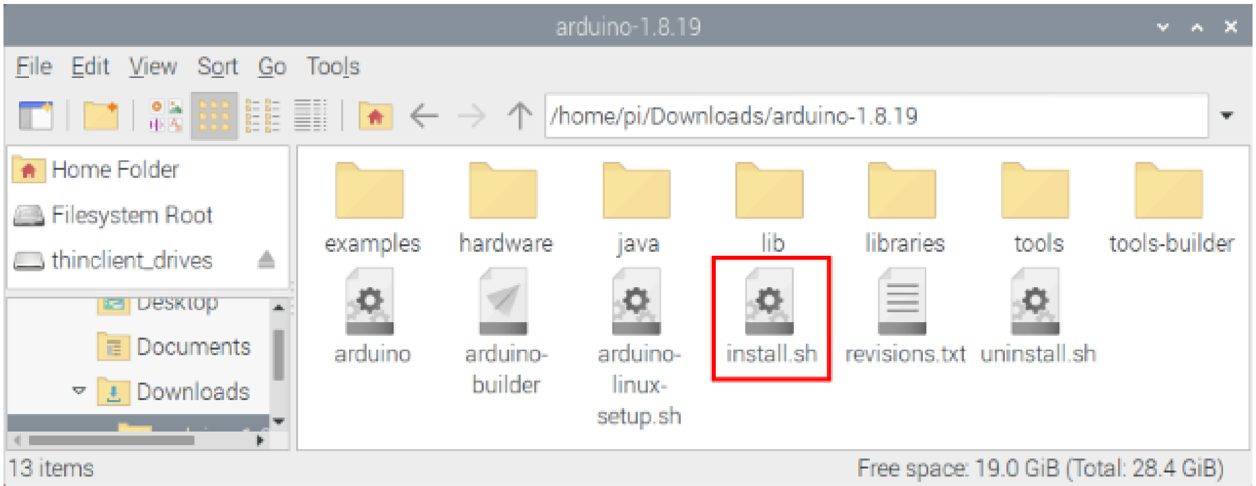


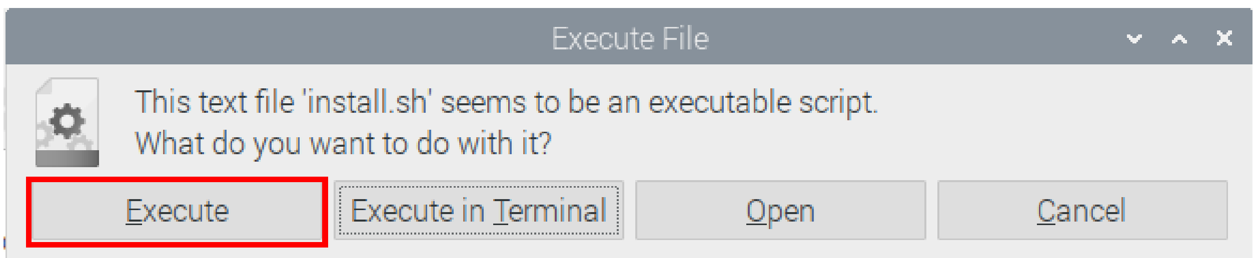


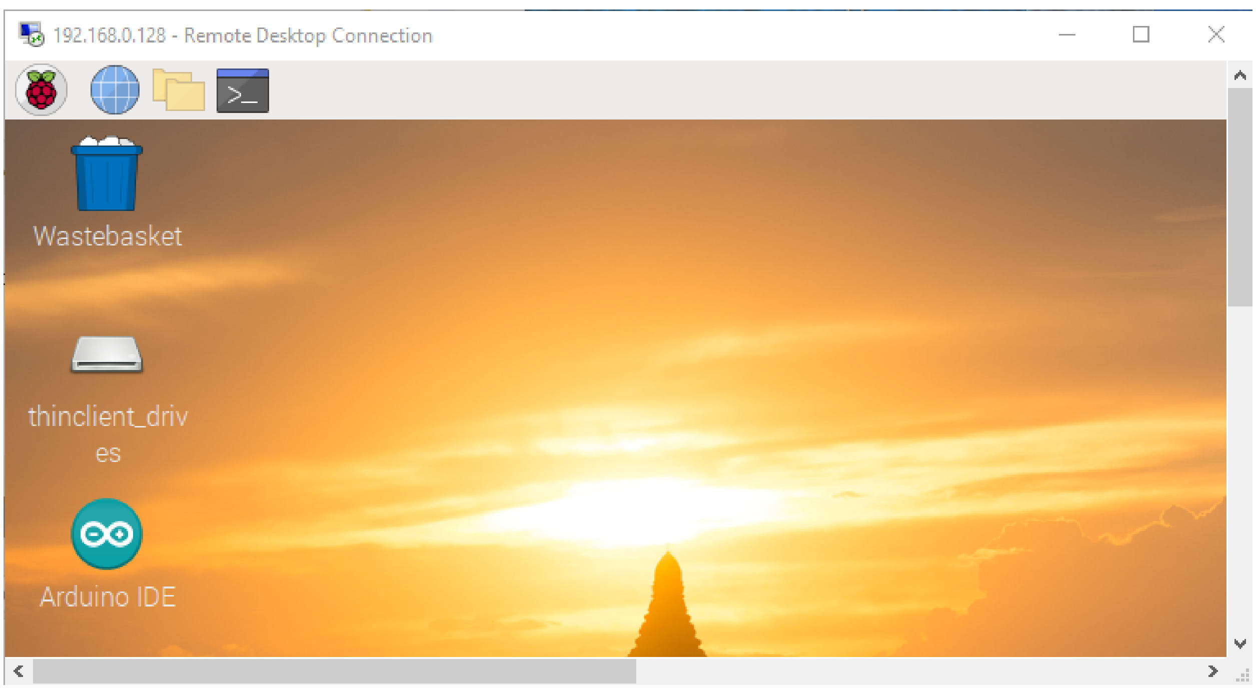




5Click install.sh file and tap itclick“Execute”in the dialog that appears to install the Arduino IDE. Once installed, an Arduino software shortcut is generated in the desktop.



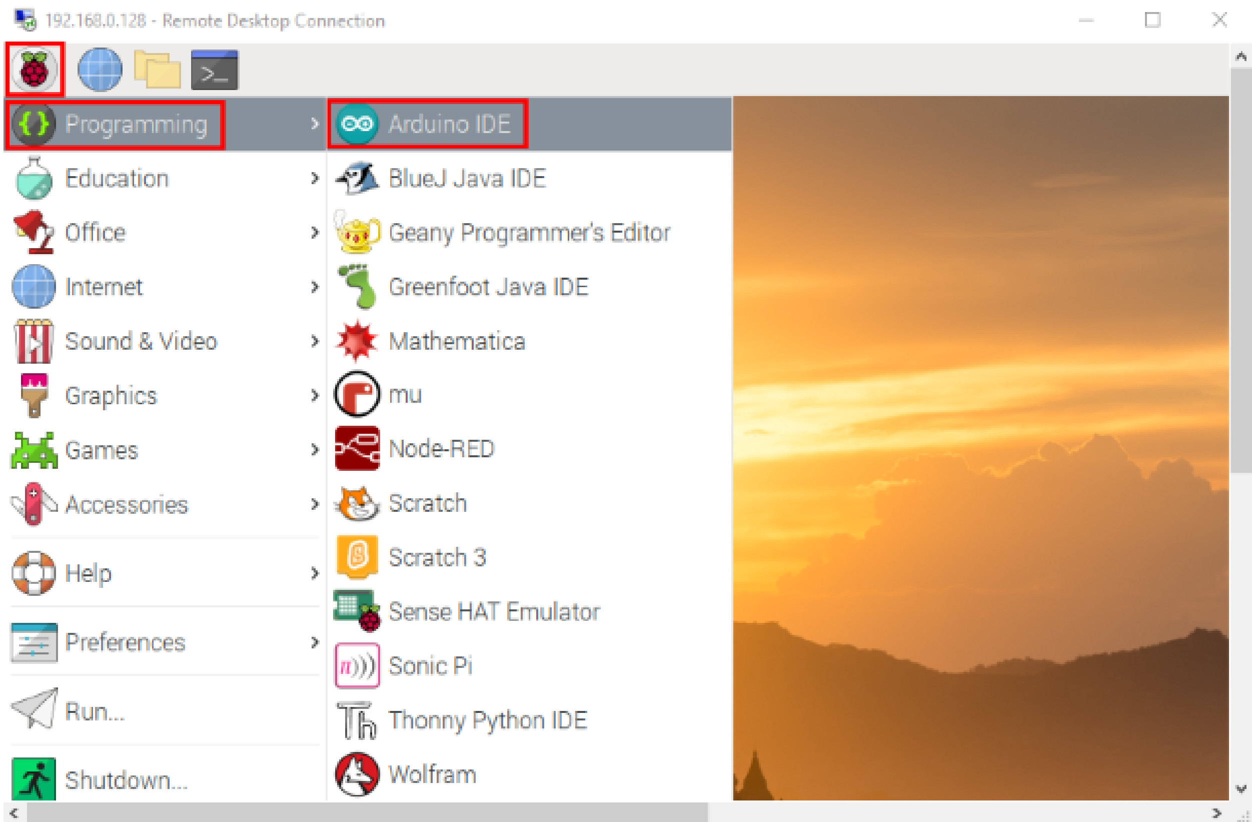


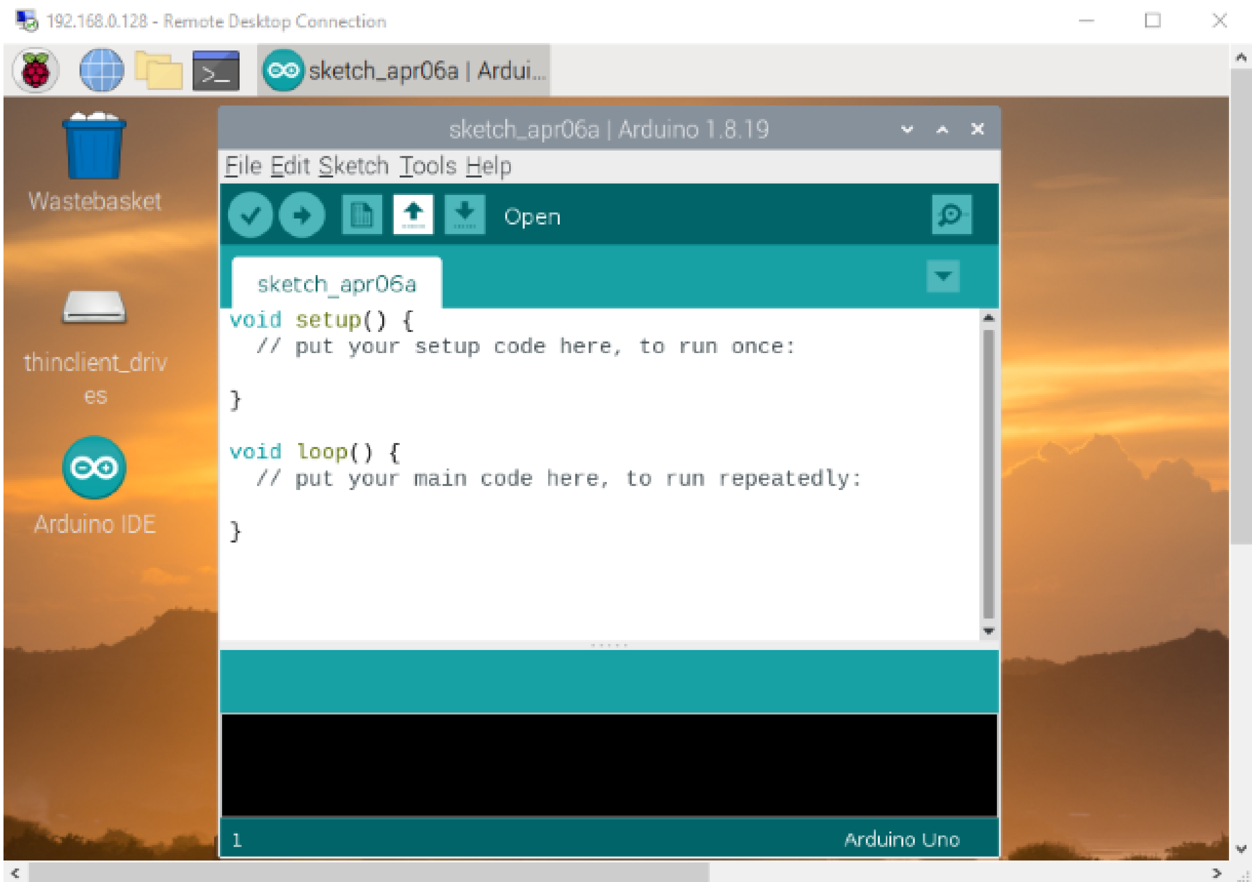


6Click 

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



8.2.4 (3) Install the ESP32 on Arduino IDE

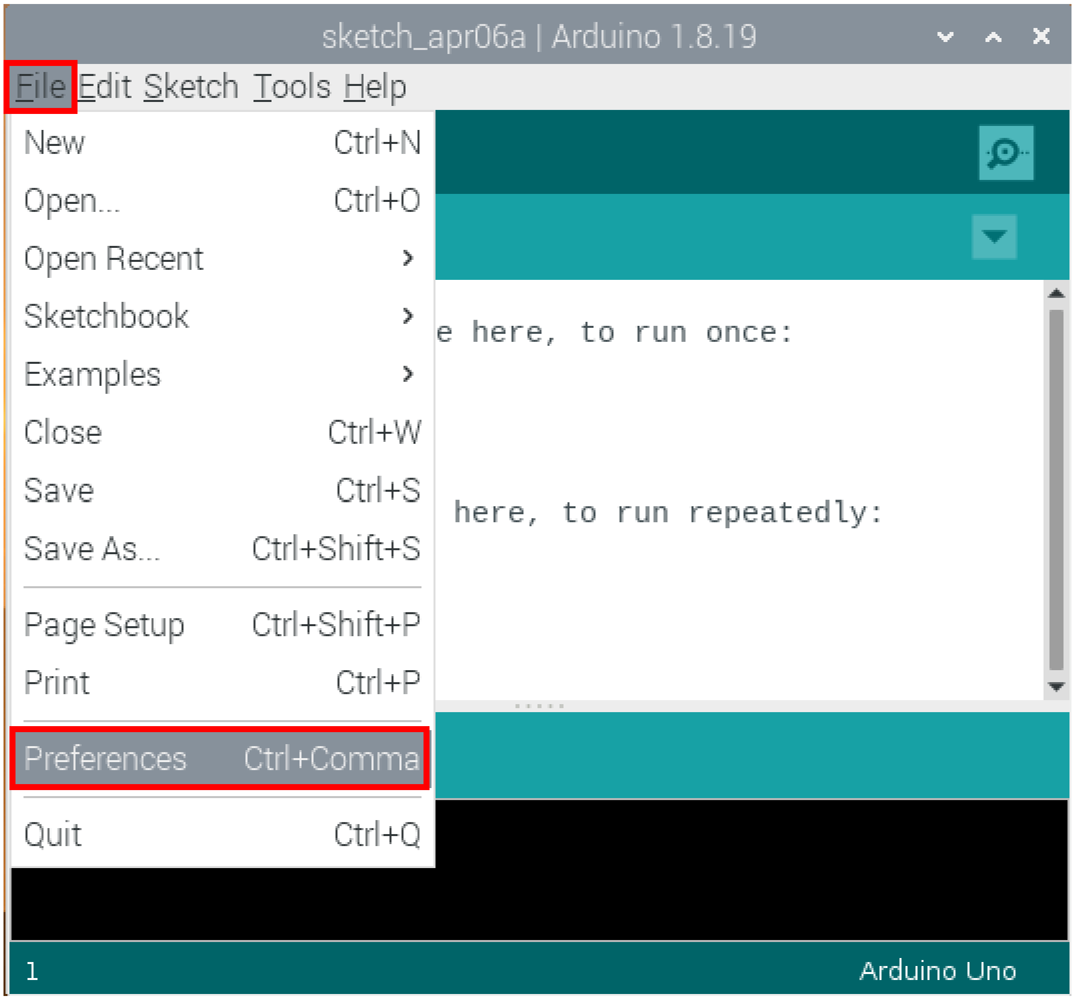
Note: You need to download Arduino IDE 1.8.5 or advanced version to install the ESP32.

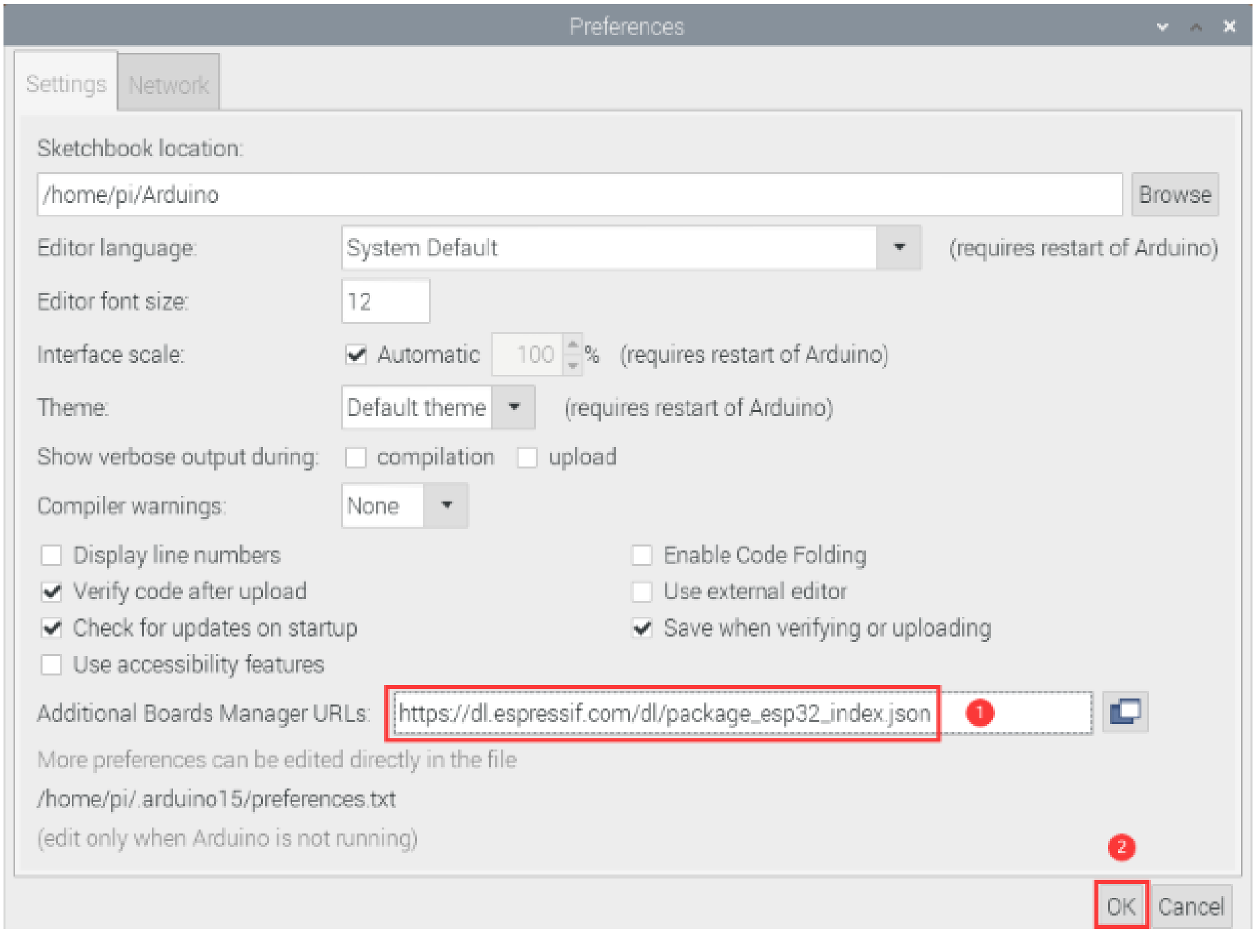
1) Click 

and click 

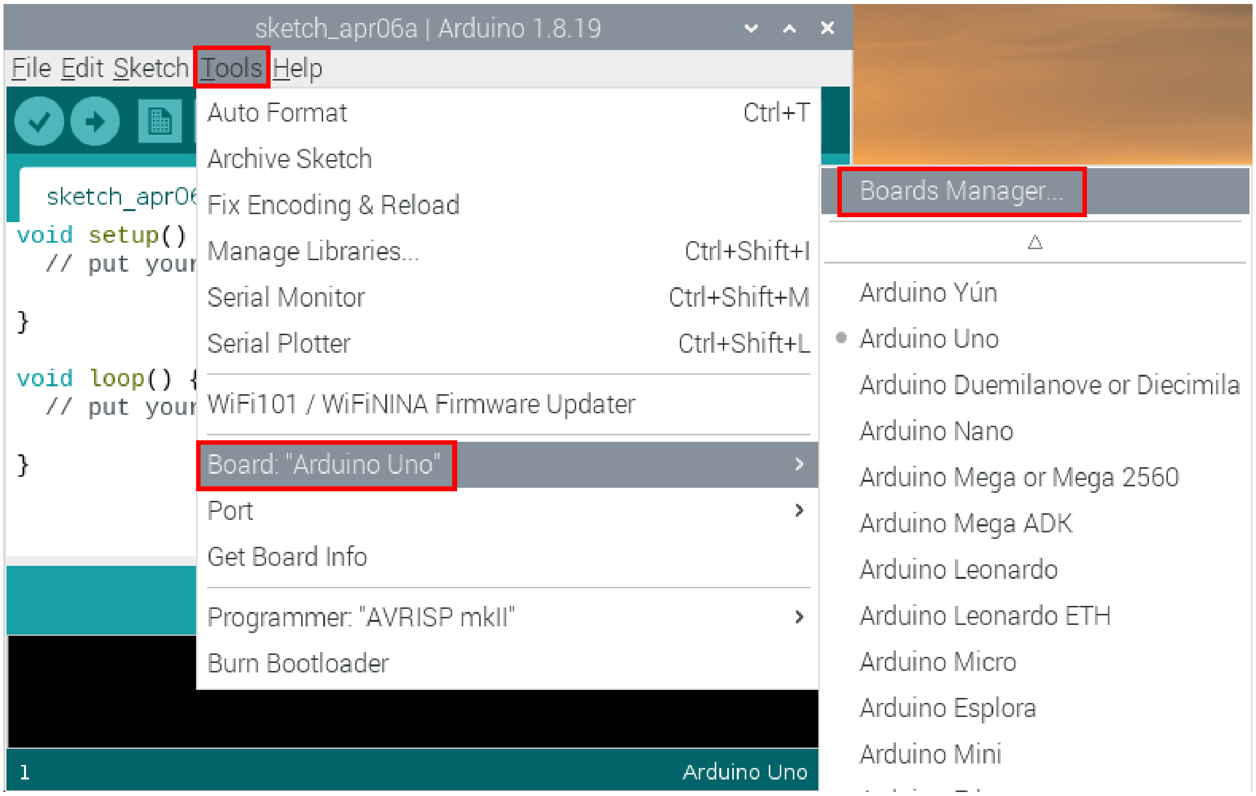
to open the Arduino IDE.

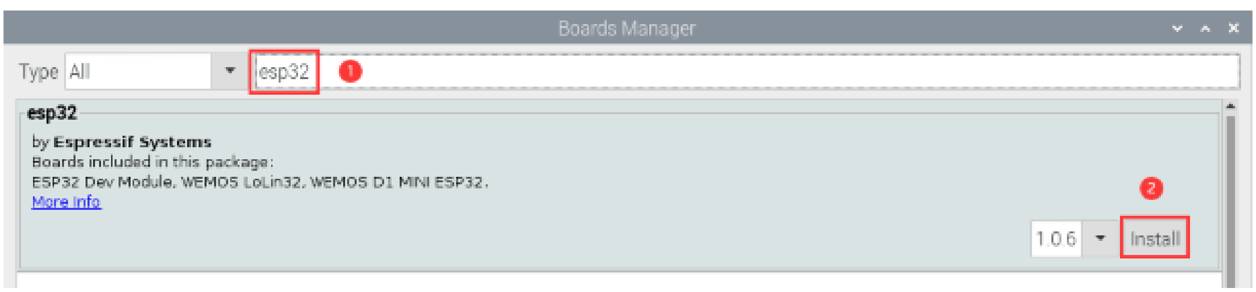
2) Click **"File"**→**"Preferences"**, copy the website address https://dl.espressif.com/dl/package_esp32_index.json in the "**Additional Boards Manager URLs:**" and click "**OK**" to save the address.

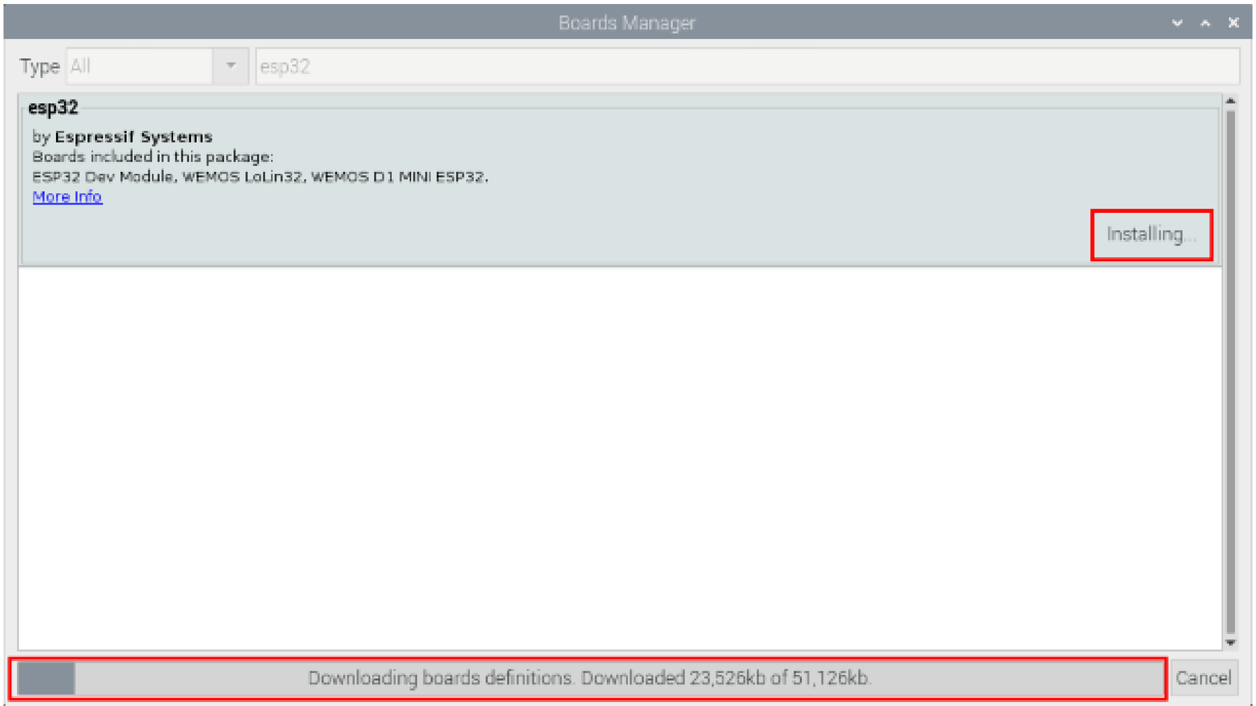

sketch_apr06a | Arduino 1.8.19
File Edit Sketch Tools Help
New Ctrl+N
Open... Ctrl+O
Open Recent
Sketchbook
Examples
Close Ctrl+W
Save Ctrl+S
Save As... Ctrl+Shift+S
Page Setup Ctrl+Shift+P
Print Ctrl+P
Preferences Ctrl+Comma
Quit Ctrl+Q
e here, to run once:
here, to run repeatedly:
1
Arduino Uno



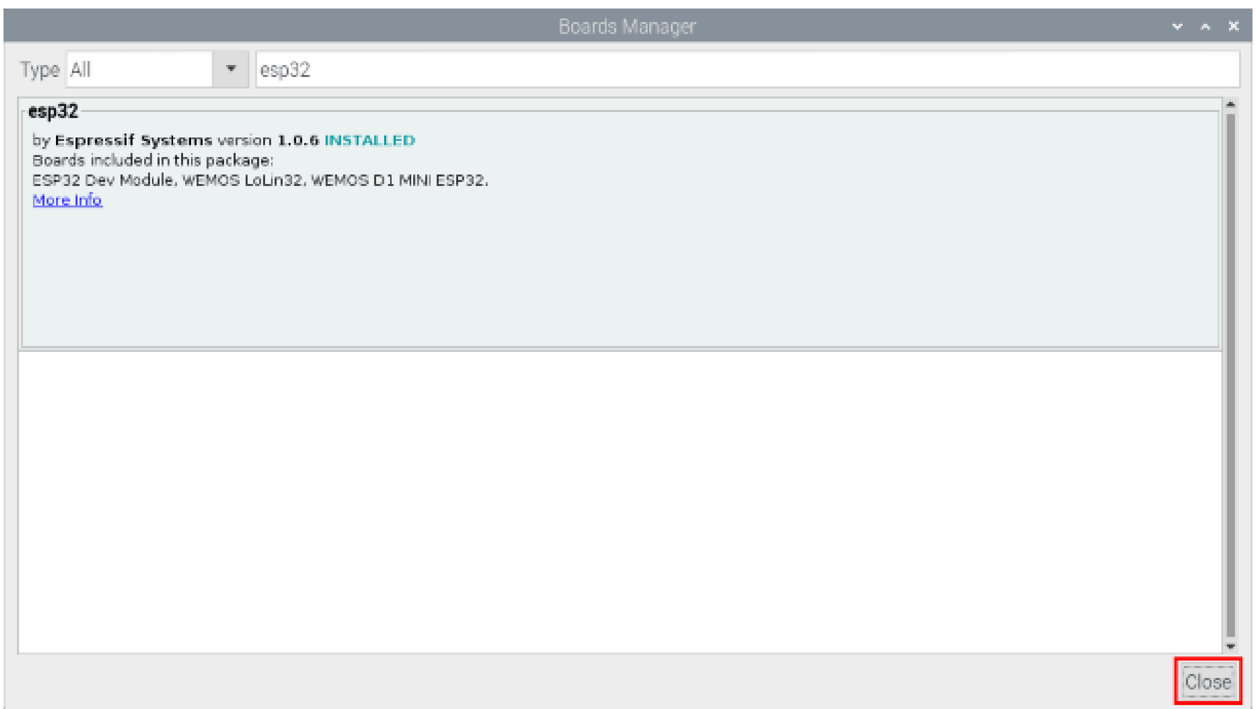
3. Click "**Tools**"→"**Board:**", then click "**Boards Manager...**" to enter "**Boards Manager**" page . Enter "**esp32**" as follows and select the latest version to Install. The installation package is not large, click "**Install**" to start the installation.


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
void setup()
// put your
}
void loop() {
// put your
}
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



4. After a while, the ESP32 installation package is installed. Then click "**Close**".

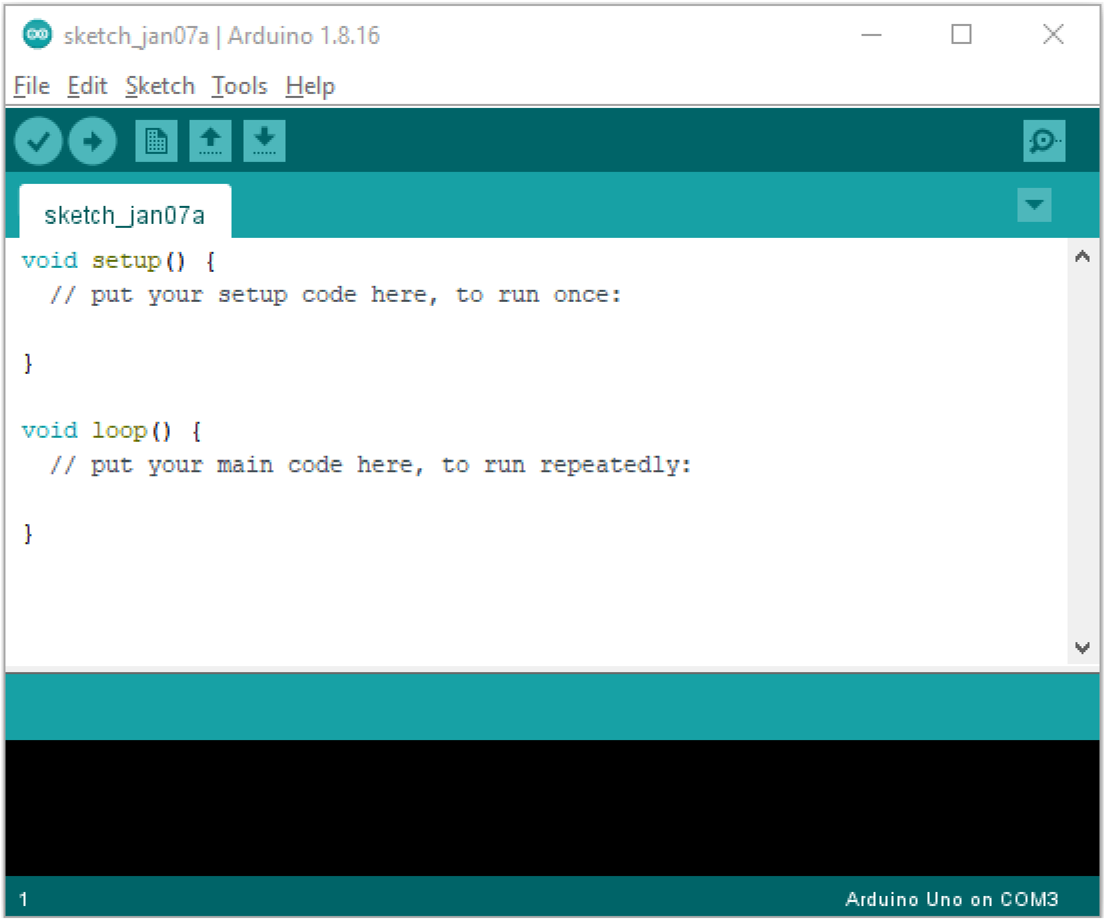


8.2.5 3. Arduino IDE Settings and Toolbars:

Click 

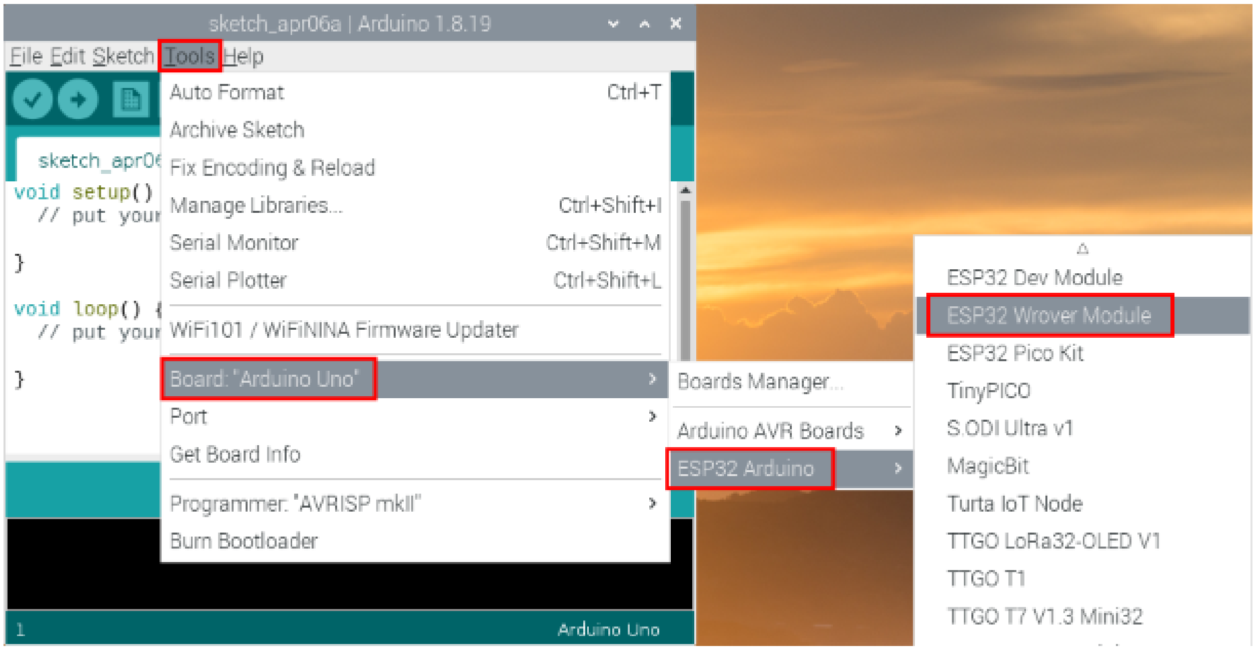
and click 

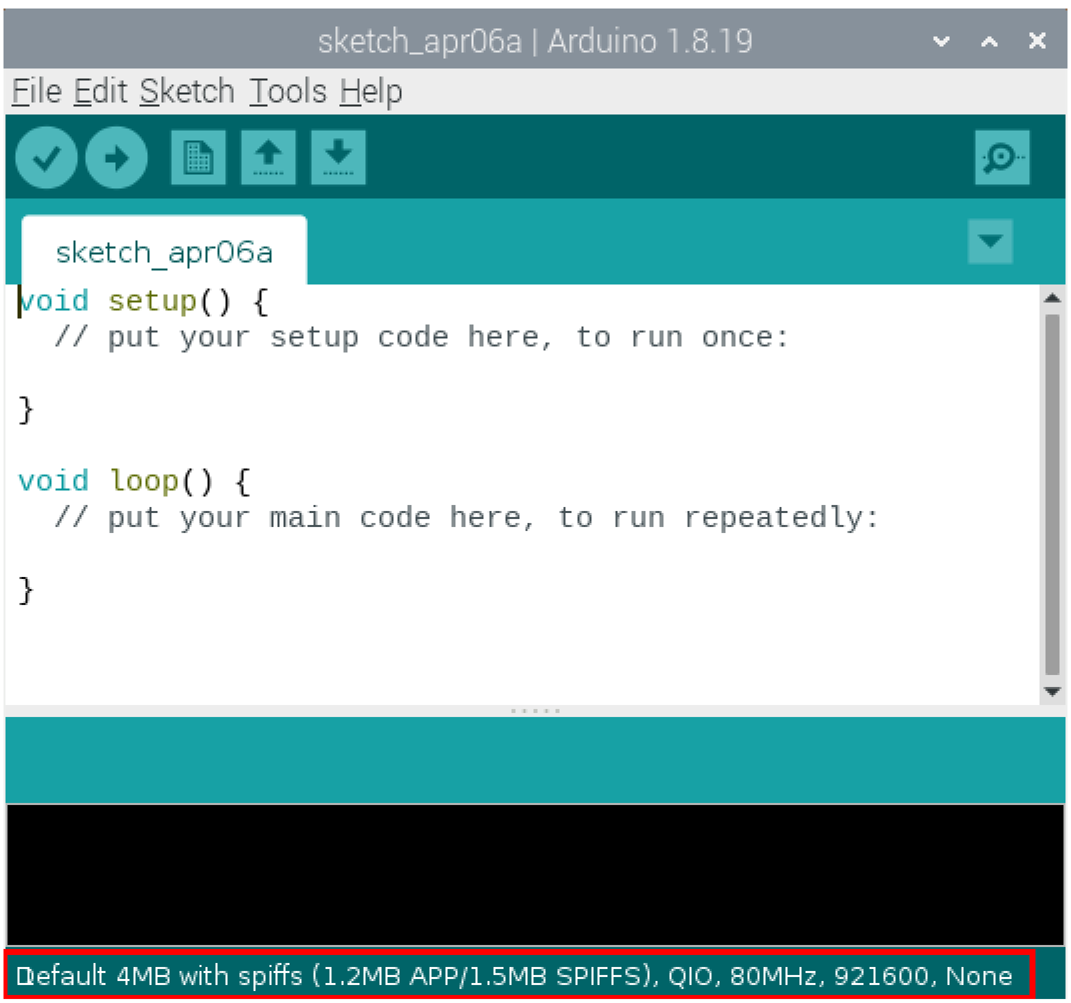
to open the Arduino IDE.



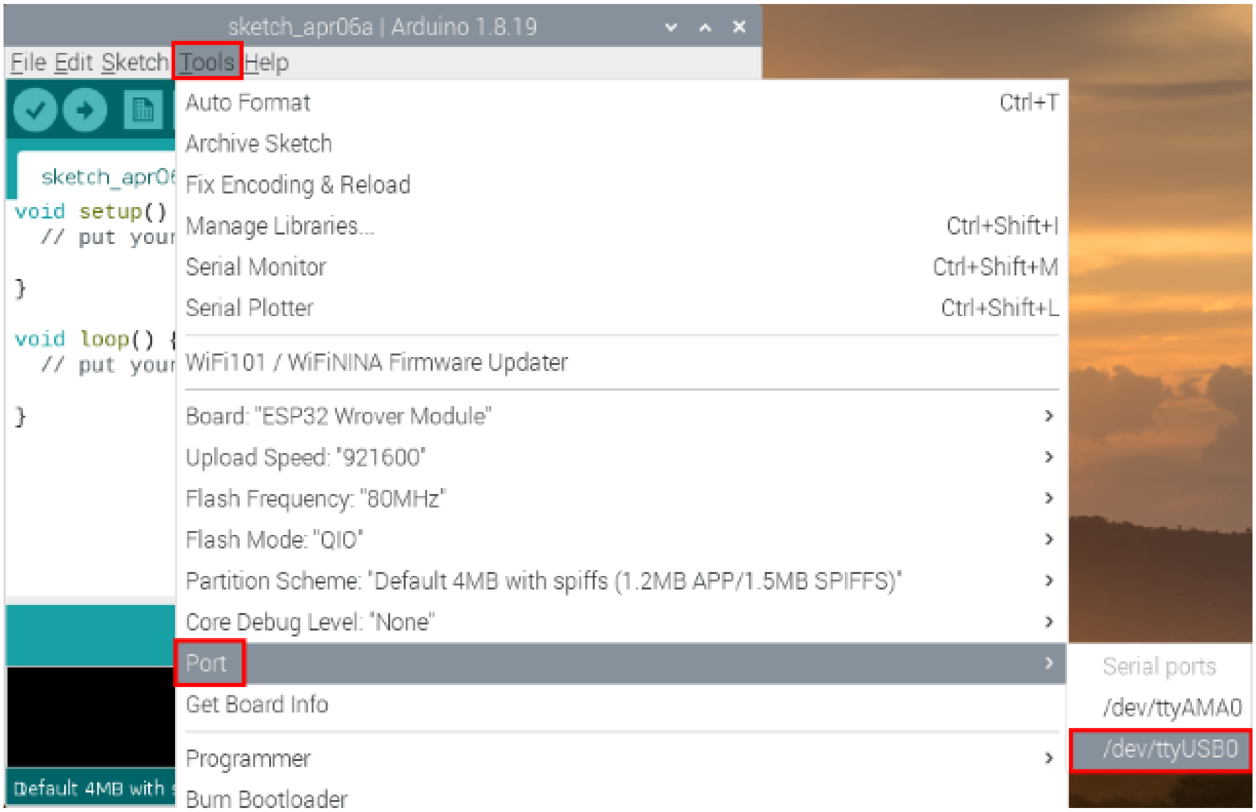
When downloading the code to the board, you must select the correct name of Arduino board that matches the board connected to the Raspberry Pi, click“**Tools**”→“**Board:**”, as shown below ;

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

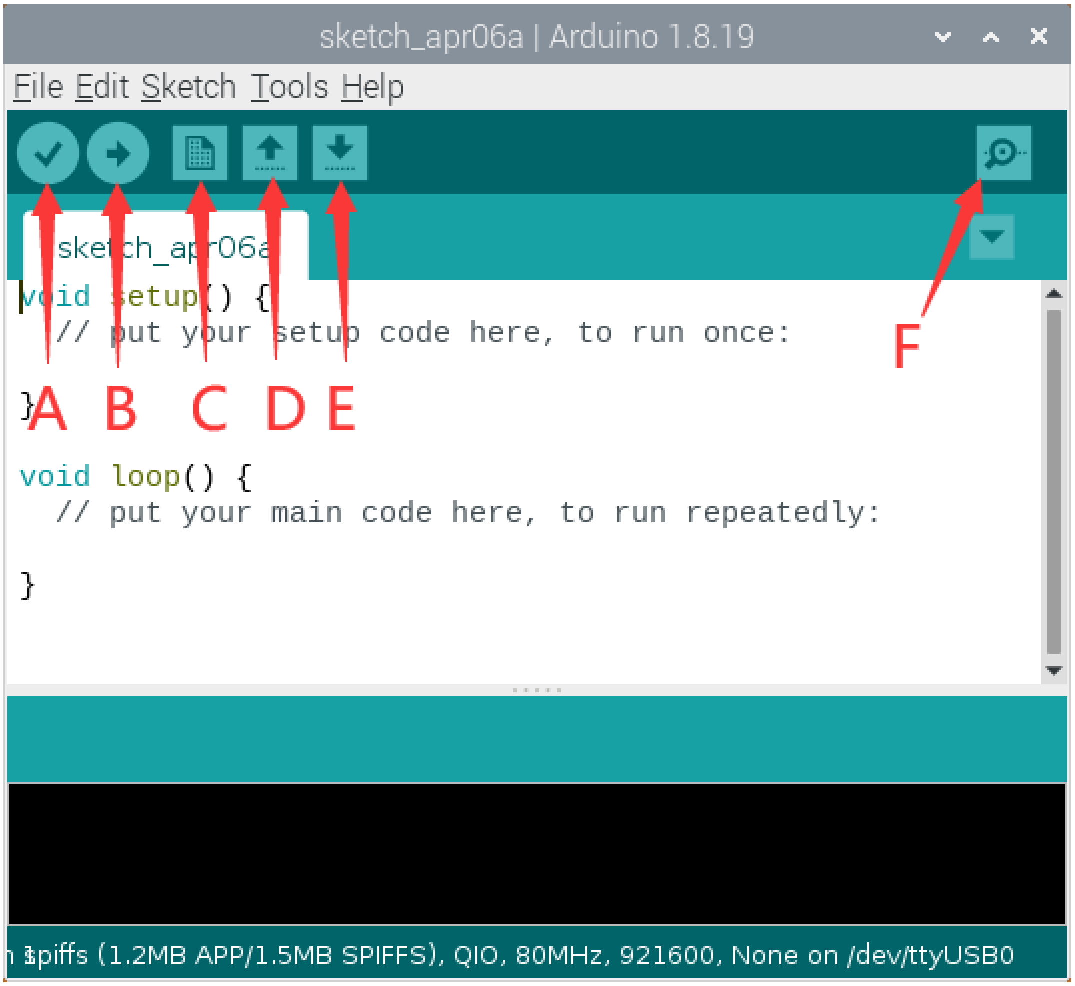




Then select the correct COM port (After connecting the ESP32 mainboard to the Raspberry Pi via a USB cable, you can see the corresponding COM port).



Before uploading the code to the ESP32 mainboard, we have to demonstrate the function of each symbol.



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

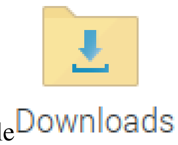
8.3.1 What are Libraries ?

Librariesare a collection of code that make it easy for you to connect sensors, displays and modules, etc.

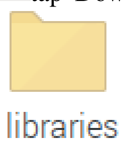
For example, the built-in LiquidCrystal library helps talk to LCD displays. There are hundreds of additional libraries available on the Internet for download.

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

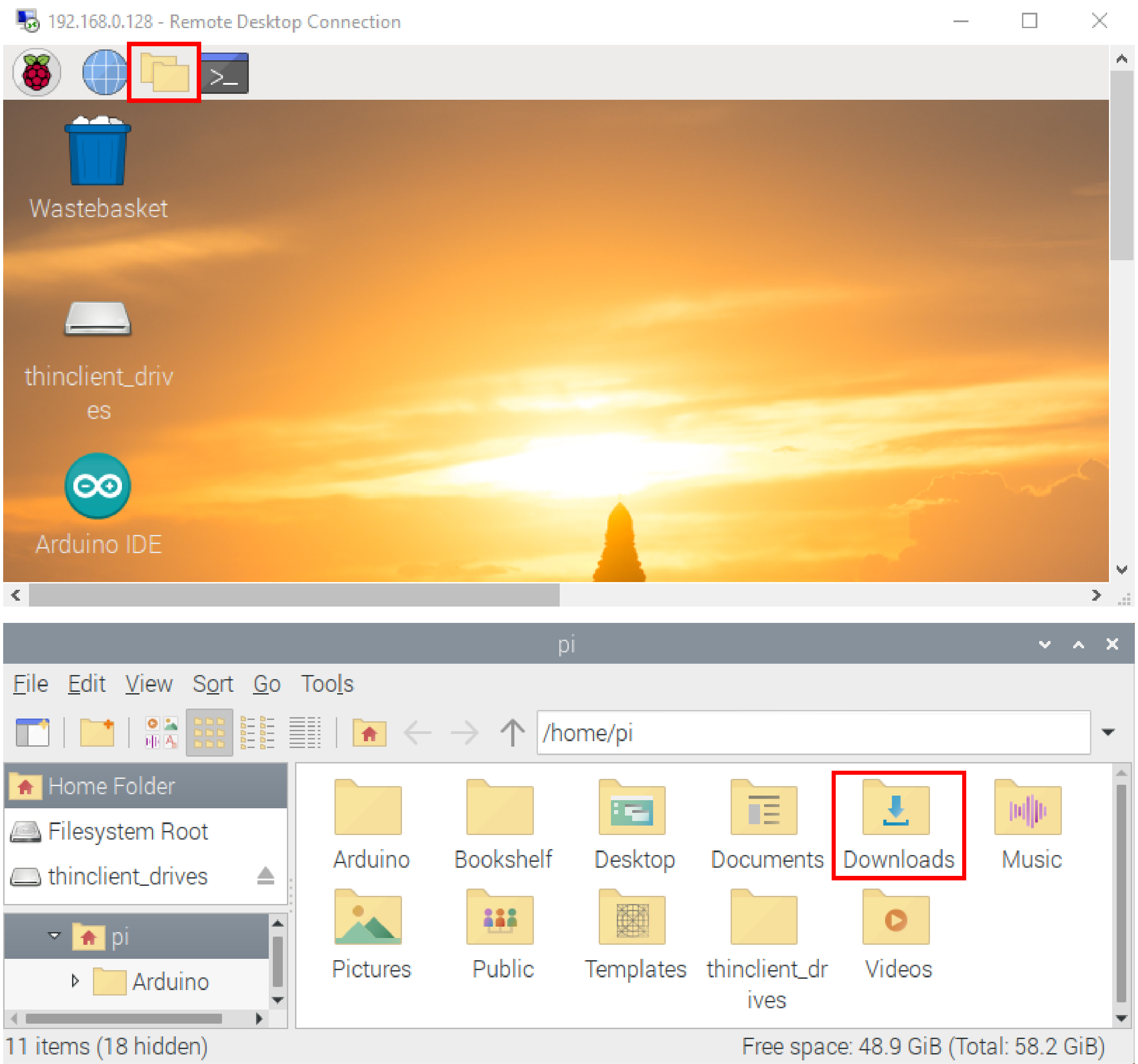
8.3.2 How to Install a Library ?

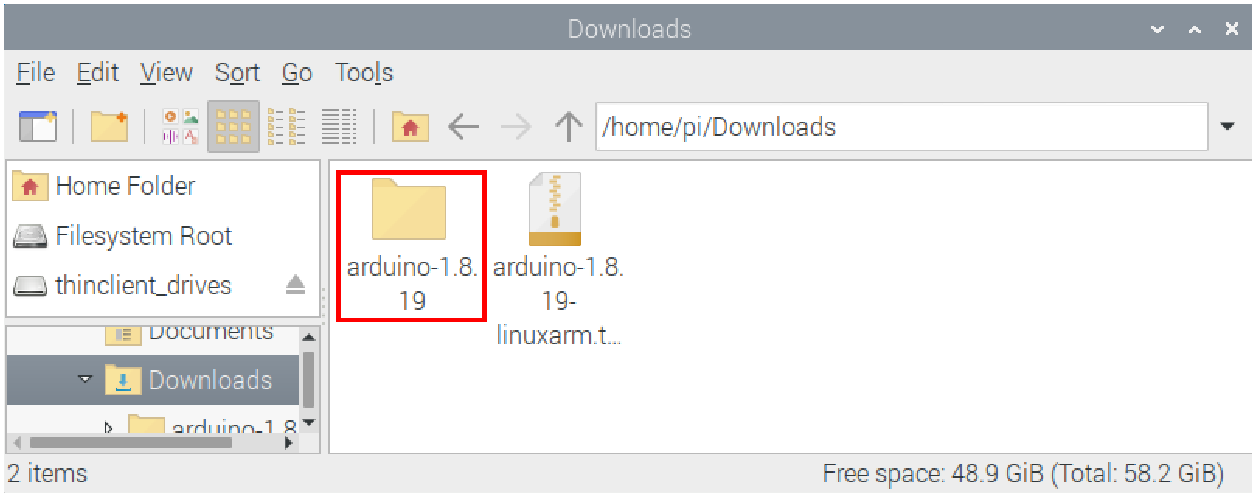
Step 1: Click  tap“Downloads”file 

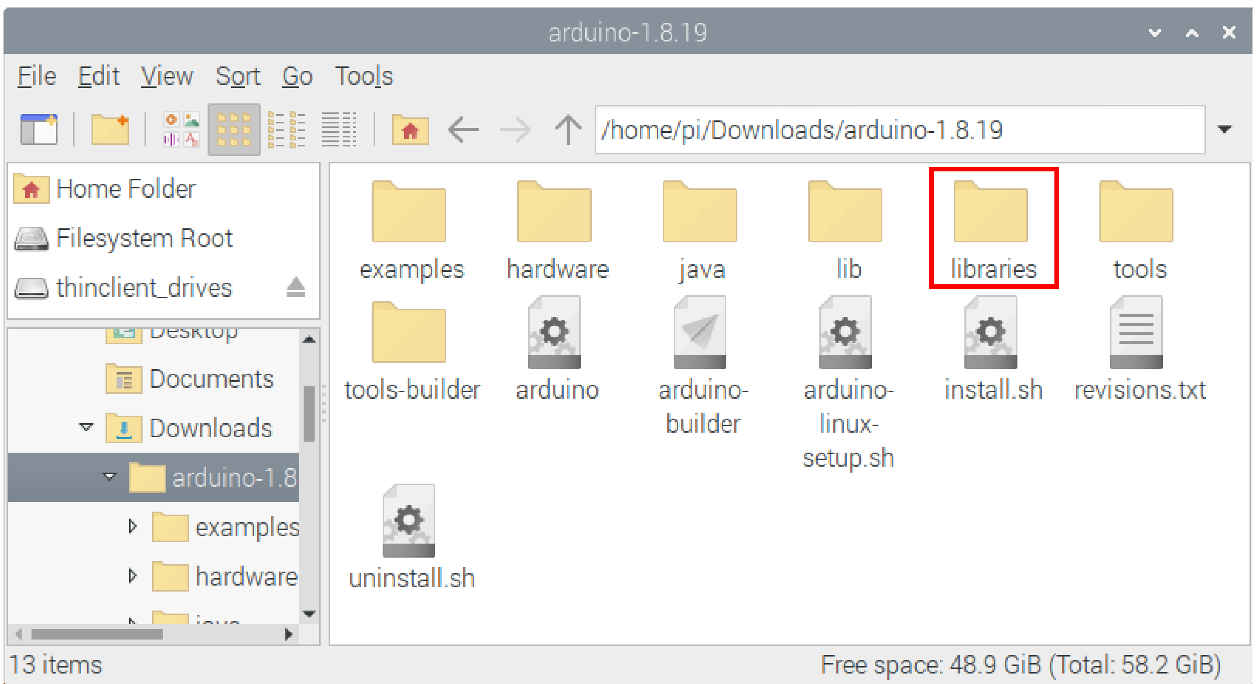
and click“arduino-1.8.19”file 

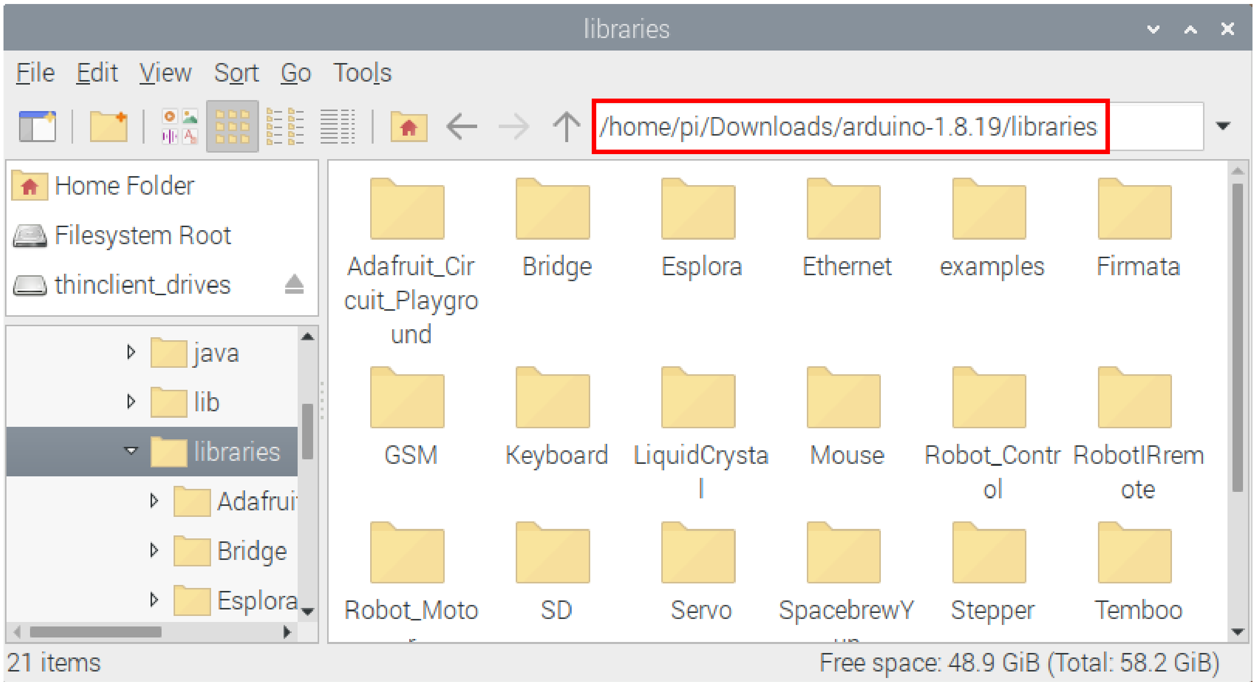
then find and click “libraries” file 

from the “arduino-1.8.19” file.



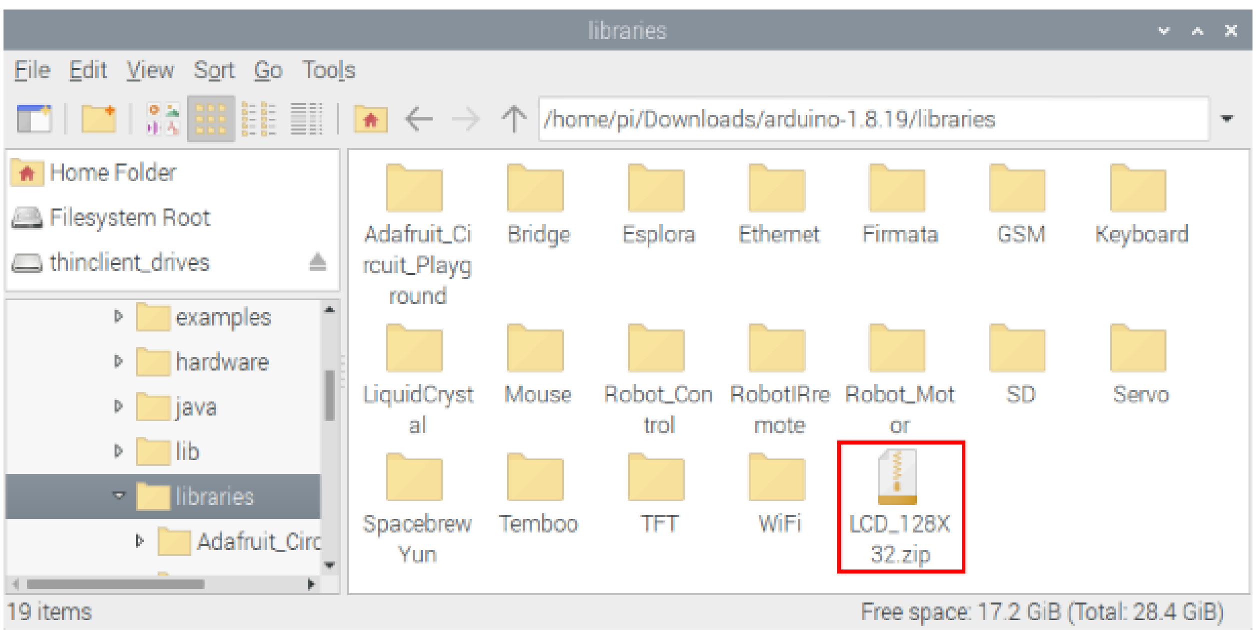

Downloads
File Edit View Sort Go Tools
/home/pi/Downloads
Home Folder
Filesystem Root
thinclient_drives
Documents
Downloads
arduino-1.8.19
arduino-1.8.19-linuxarm.t...
2 items
Free space: 48.9 GiB (Total: 58.2 GiB)

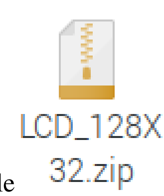

arduino-1.8.19
File Edit View Sort Go Tools
/home/pi/Downloads/arduino-1.8.19
Home Folder
Filesystem Root
thinclient_drives
Documents
Downloads
arduino-1.8
examples
hardware
examples
hardware
java
lib
libraries
tools
tools-builder
arduino
arduino-builder
arduino-linux-setup.sh
install.sh
revisions.txt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



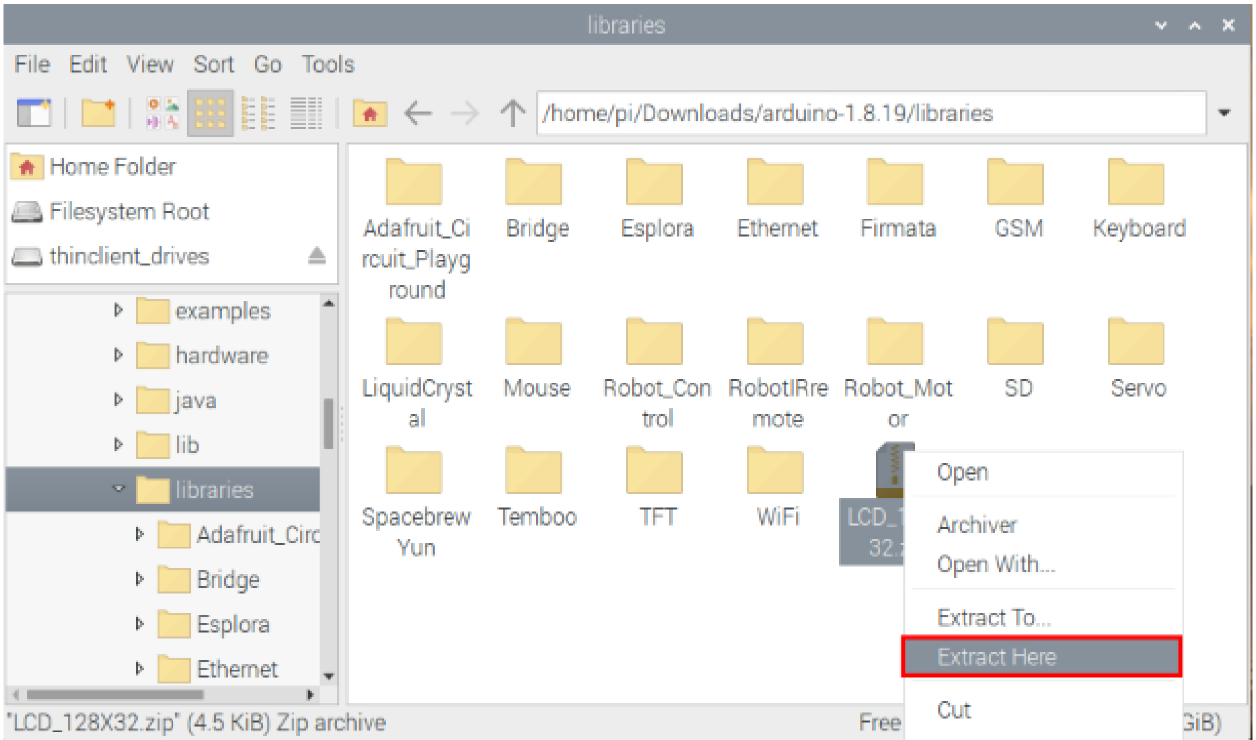
Step 2 : Copy and paste the Arduino C library ZIP file (the default is ZIP file) from the provided Arduino Libraries folder into the libraries file opened in the first stepthe route is/home/pi/Downloads/arduino-1.8.19/libraries.

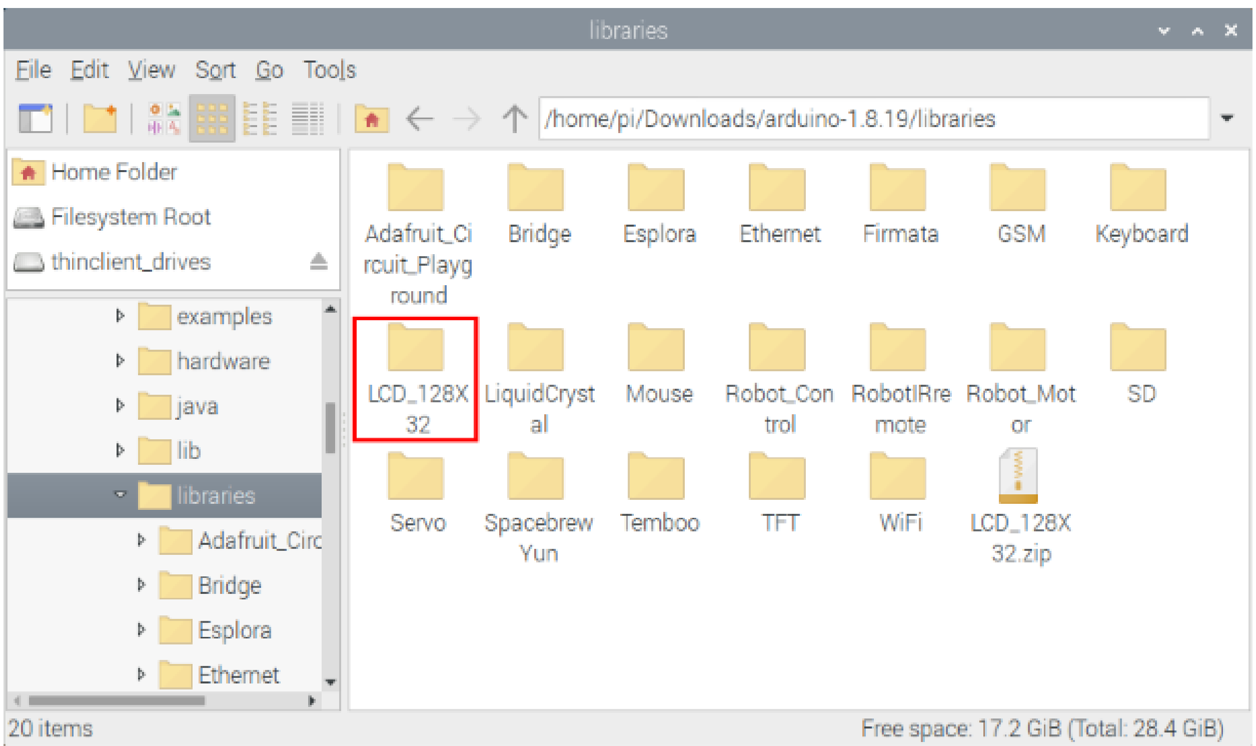
Click on the link to download the library fileArduino C Librarie





Step 3: Unzip the Arduino C package in the libraries folderfor exampleright-click"LCD_128X32.zip"file select and tap "Extract Here"to unzip the "LCD_128X32.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
examples
hardware
java
lib
libraries
Adafruit_Circ
Bridge
Esplora
Ethernet
Adafruit_Circuit_Playground
Bridge
Esplora
Ethernet
Firmata
GSM
Keyboard
LiquidCrystal
Mouse
Robot_Control
RobotIRremote
Robot_Motor
SD
Servo
SpacebrewYun
Temboo
TFT
WiFi
Open
Archiver
Open With...
Extract To...
Extract Here
Cut
"LCD_128X32.zip" (4.5 KiB) Zip archive
Free


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
Esplora
Ethernet
Adafruit_Circuit_Playground
Bridge
Esplora
Ethernet
Firmata
GSM
Keyboard
LCD_128X32
LiquidCrystal
Mouse
Robot_Control
RobotIRremote
Robot_Motor
SD
Servo
SpacebrewYun
Temboo
TFT
WiFi
LCD_128X32.zip
20 items
Free space: 17.2 GiB (Total: 28.4 GiB)

CHAPTER

NINE

C LANGUAGE (RASPBERRY PI) TUTORIAL

Development Environment Configuration

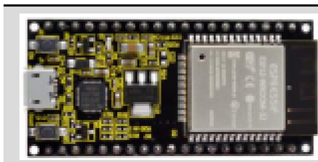
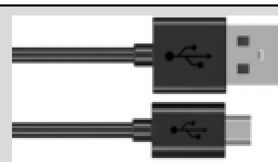
RaspberryPi——Arduino Development Environment Configuration*RaspberryPi——Arduino*

9.1 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 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.

2. Components

	
ESP32*1	USB Cable*1

3. Wiring Diagram

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

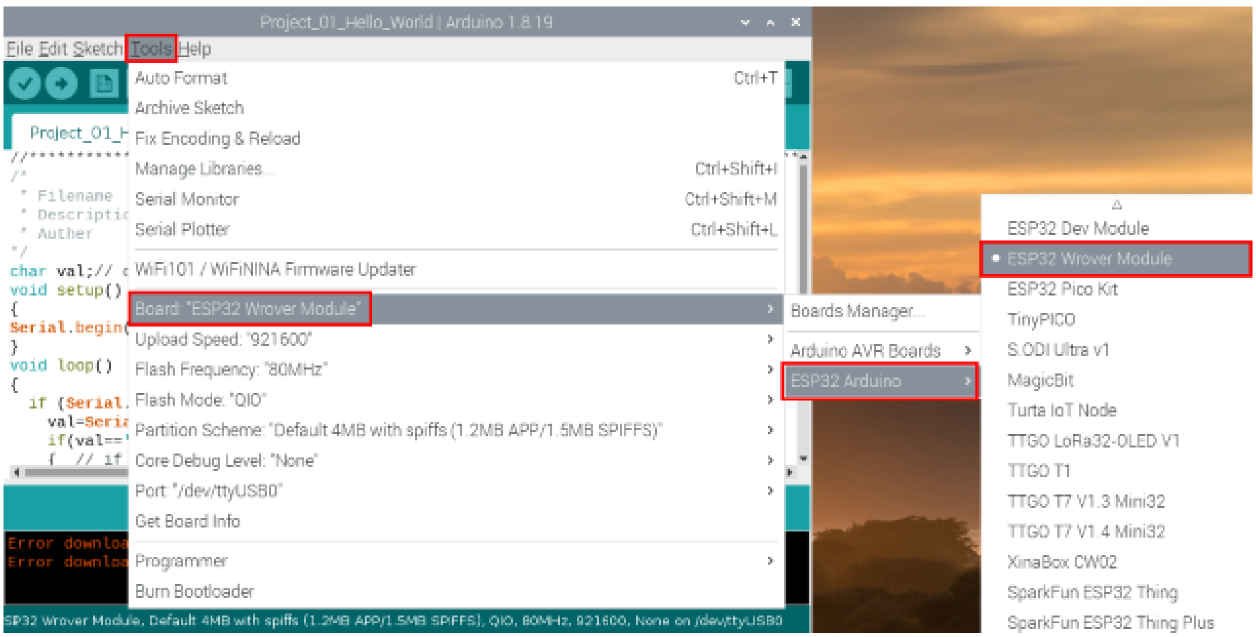
```
//**************************************************************************************
/*

 * 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()
```

(continues on next page)

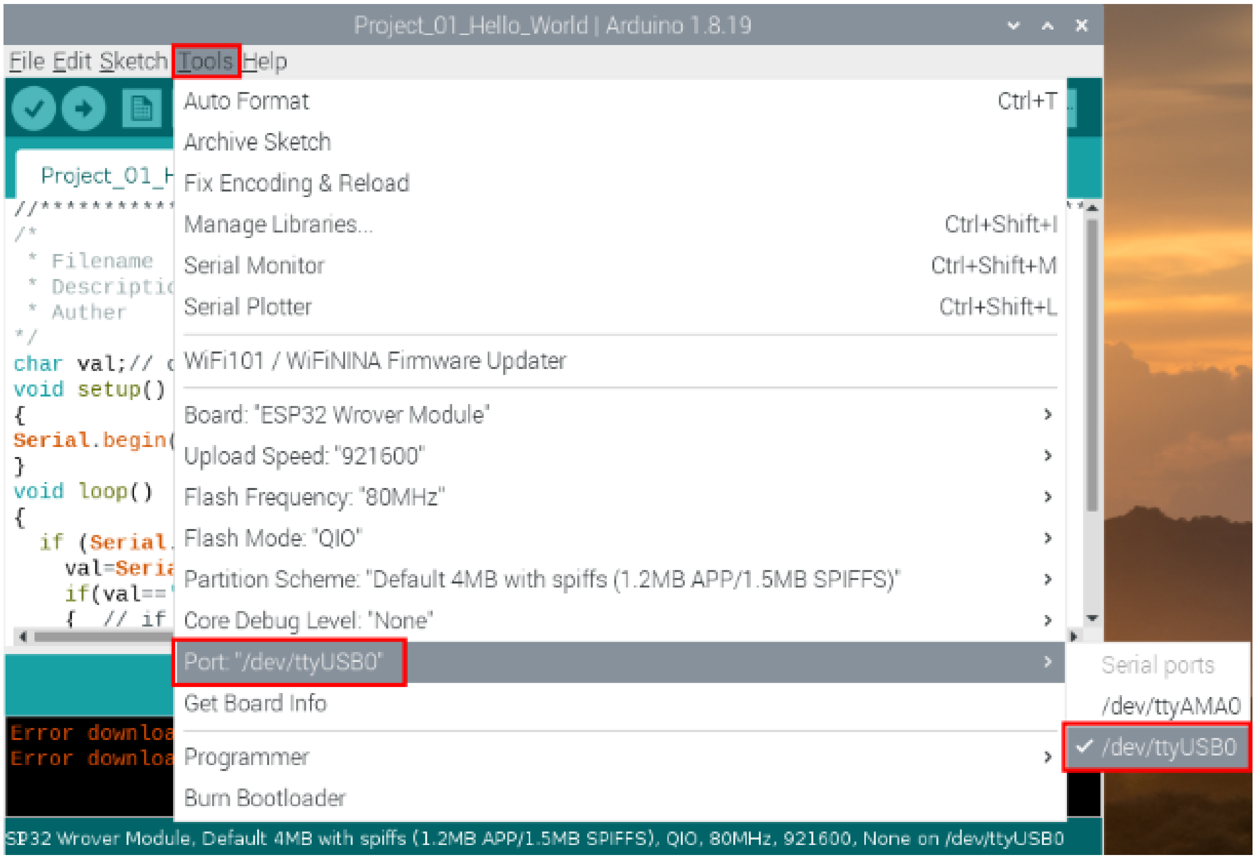
(continued from previous page)

```
{
  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 !”.
 }
  }
}
//
↪*********************************************************************************
```

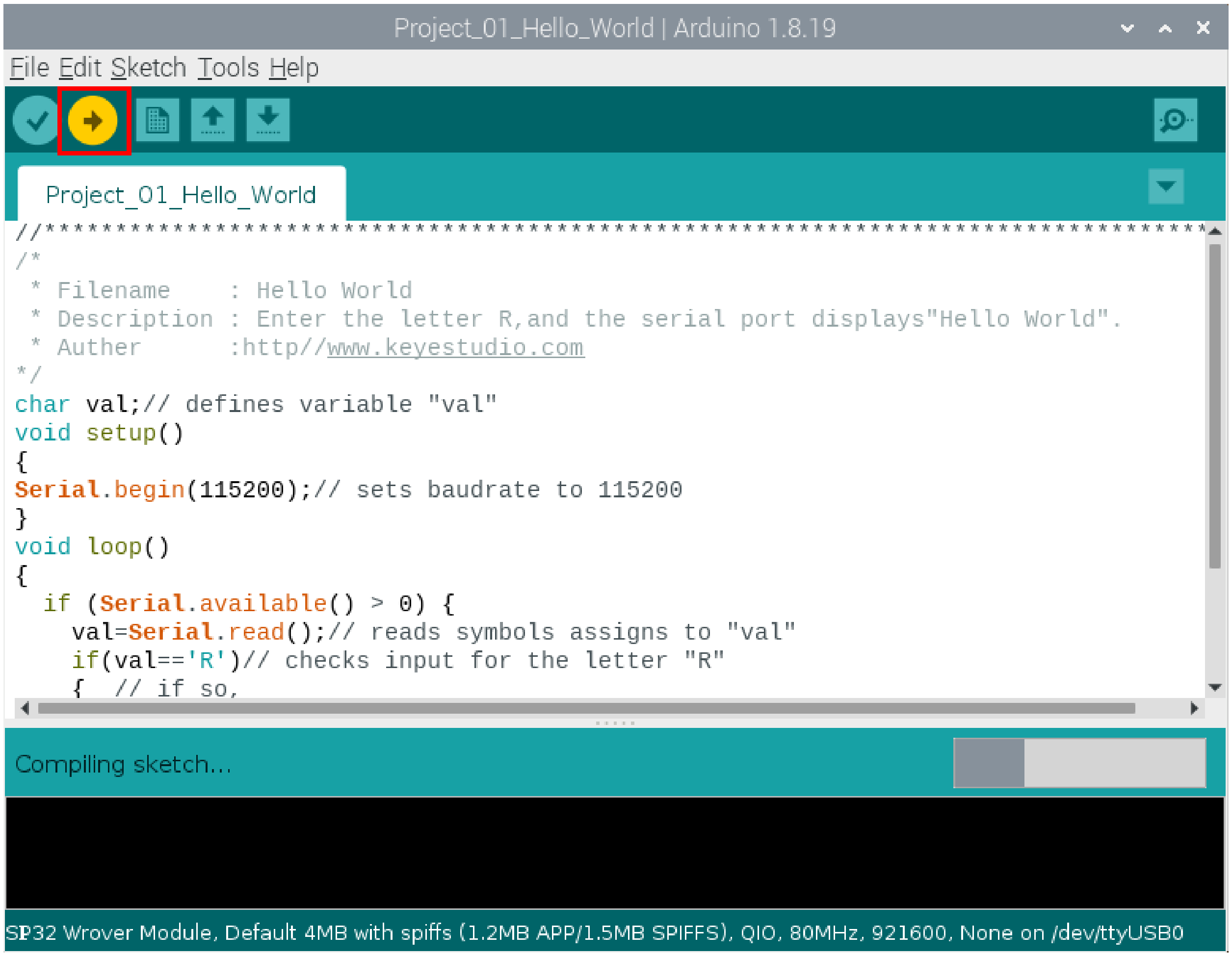
Before uploading the project code to ESP32click “Tools”→“Board” and select“ESP32 Wrover Module”.



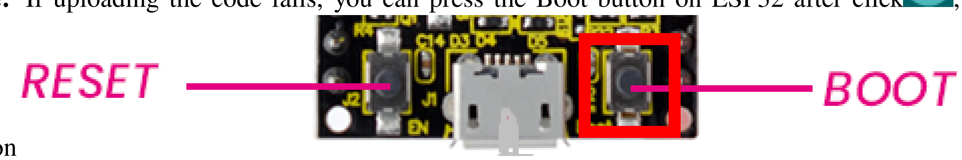
Select the serial port.



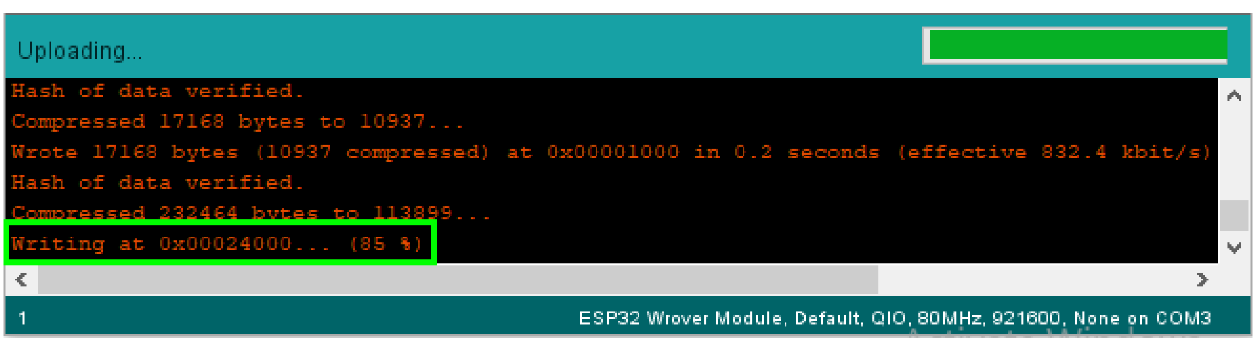
Click  to download the code to ESP32.



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:



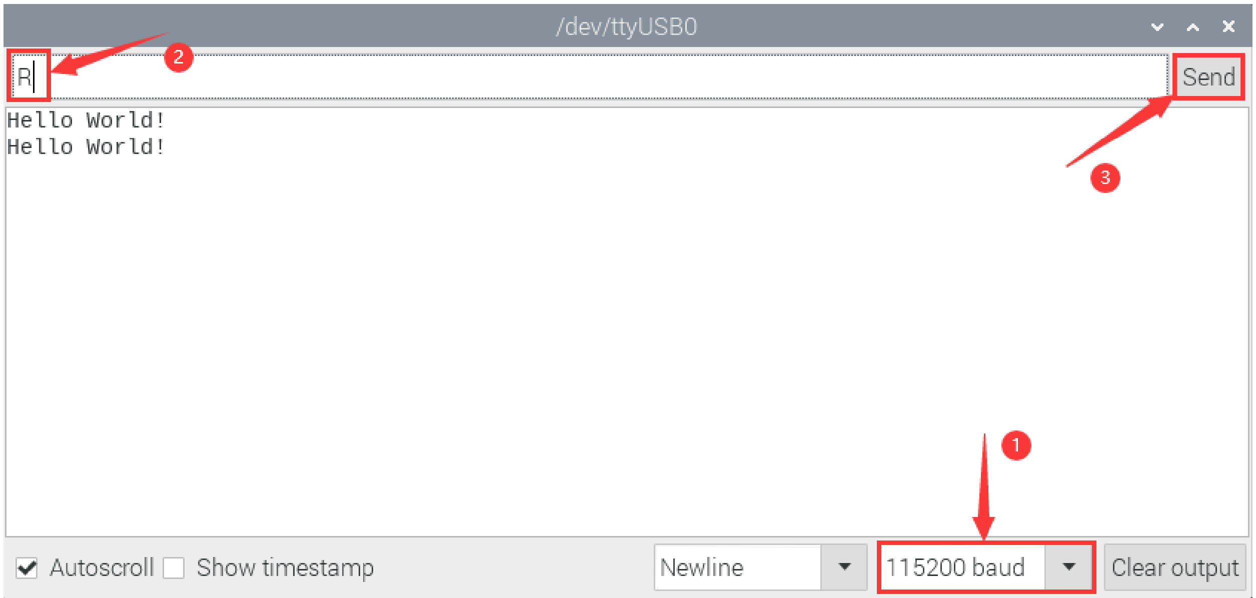
The Project code is uploaded successfully

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!”.

(**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 main board and repeat the above operation.)

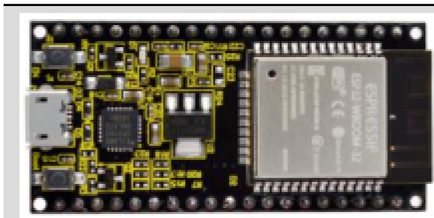
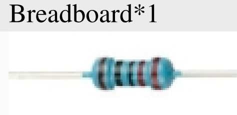
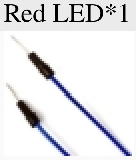
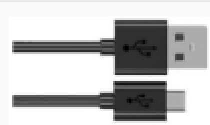


9.2 Project 02: Turn On LED

1.Introduction

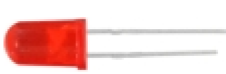
In this project, we will show you how to light up the LED. We use the ESP32's digital pin to turn on the LED so that the LED is lit up.

2.Components

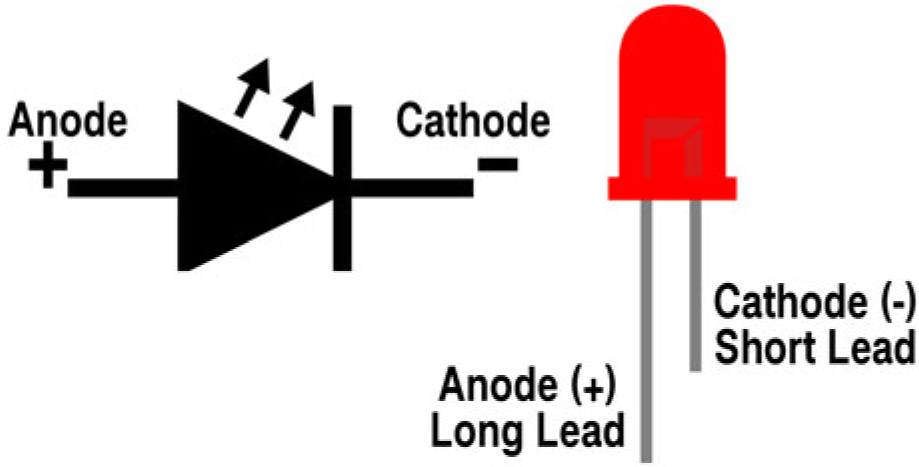
	
ESP32*1	Breadboard*1
	
Red LED*1	220 Resistor*1
	
Jumper Wire*2	USB Cable*1

3.Component knowledge

1LED:

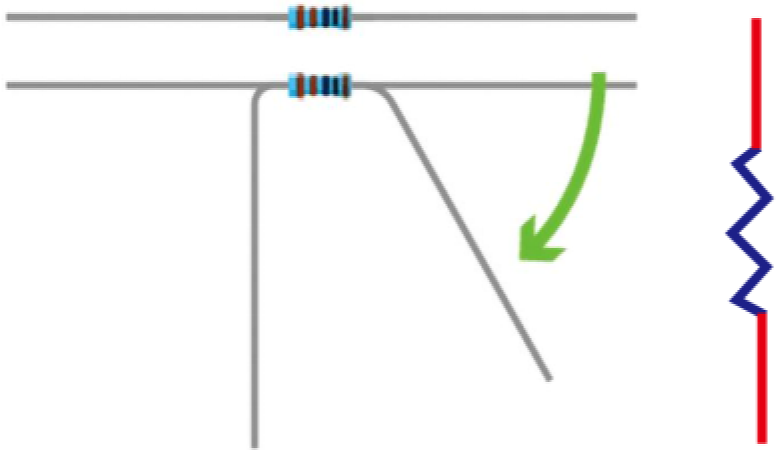


The LED is a semiconductor known as "light-emitting diode" ,which is an electronic device made from semiconducting materials(silicon, selenium, germanium, etc.). It has an anode and a cathode, the short lead is cathode, which connects to GND; the long lead is anode, which connects to 3.3V or 5V.



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.

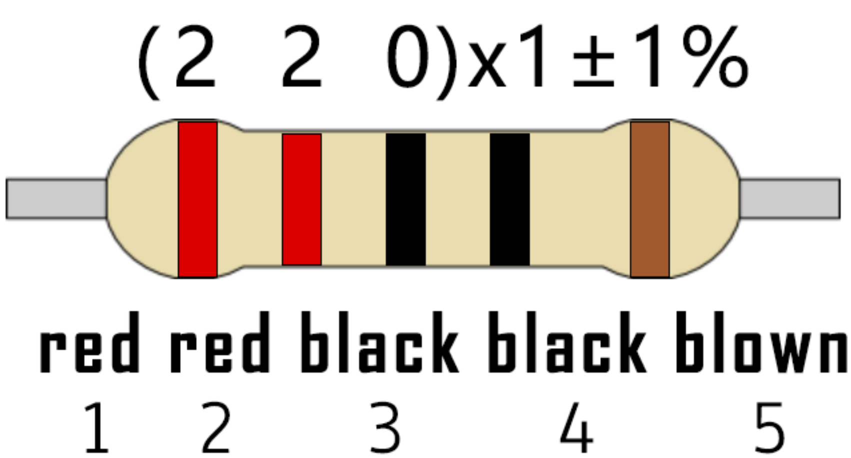
-Color 4 – Multiplier.

-Color 5 – Tolerance.

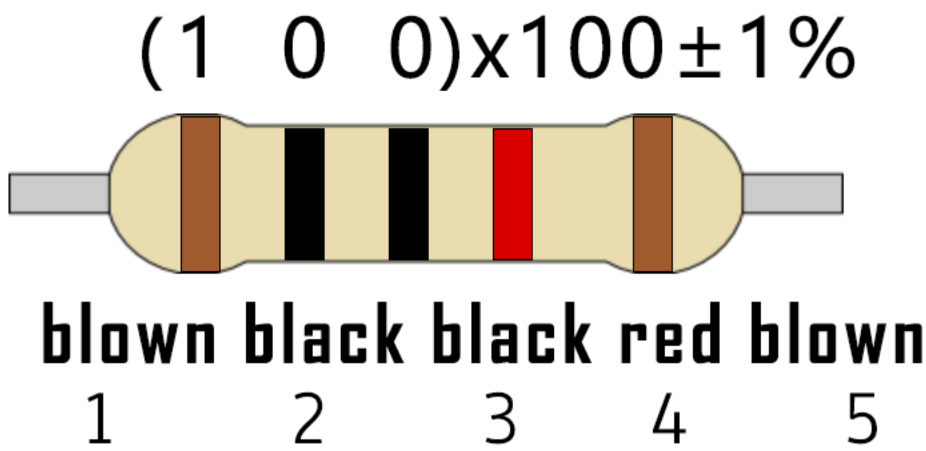
	1st Digit	2nd Digit	3rd Digit	Multiplier	Tolerance
Black		0	0	x1	
Brown	1	1	1	x10	± 1%
Red	2	2	2	x100	± 2%
Orange	3	3	3	x1K	± 3%
Yellow	4	4	4	x10K	± 4%
Green	5	5	5	x100K	± 0.5%
Blue	6	6	6	x1M	± 0.25%
Violet	7	7	7	x10M	± 0.10%
Grey	8	8	8	x100M	± 0.05%
White	9	9	9	x1G	
Gold				÷ 10	± 5%
Silver				÷ 100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

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

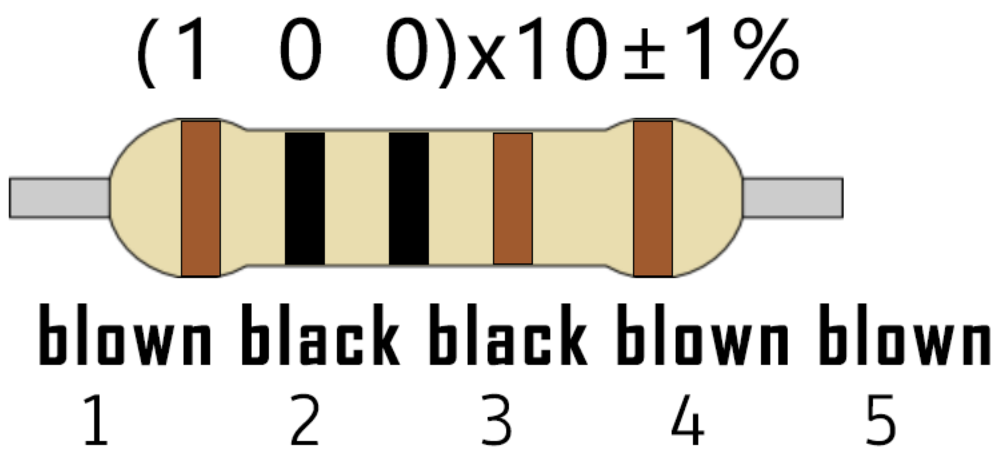
220 Resistor*10



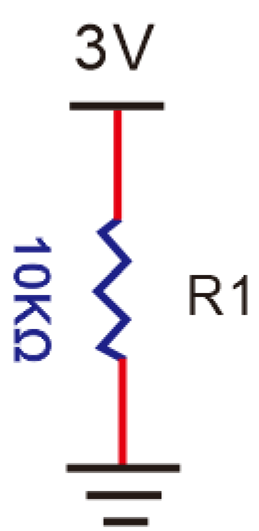
10K Resistor*10



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.

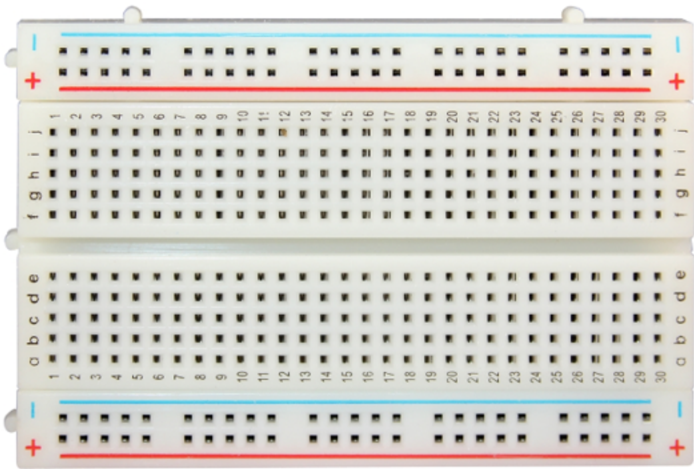


Don't connect a low resistance directly to the two poles of the power supply. as this will cause excessive current to

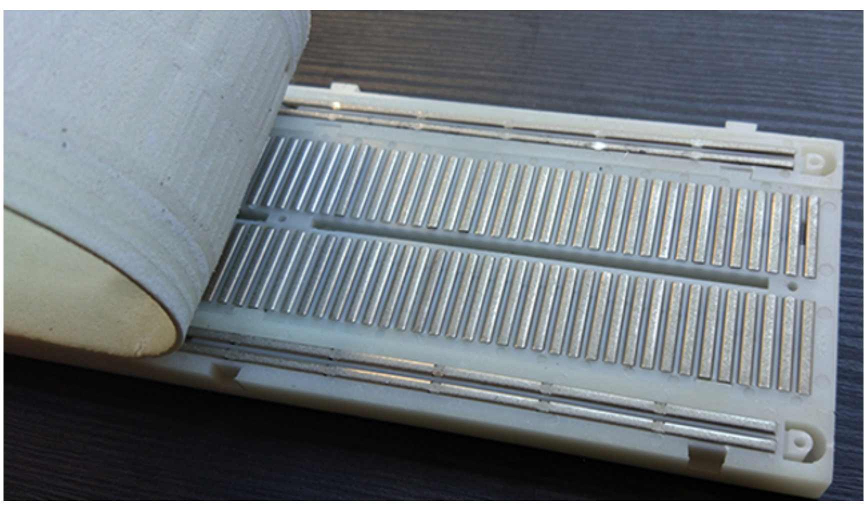
damage the electronic components. Resistors do not have positive and negative poles.

3Bread board

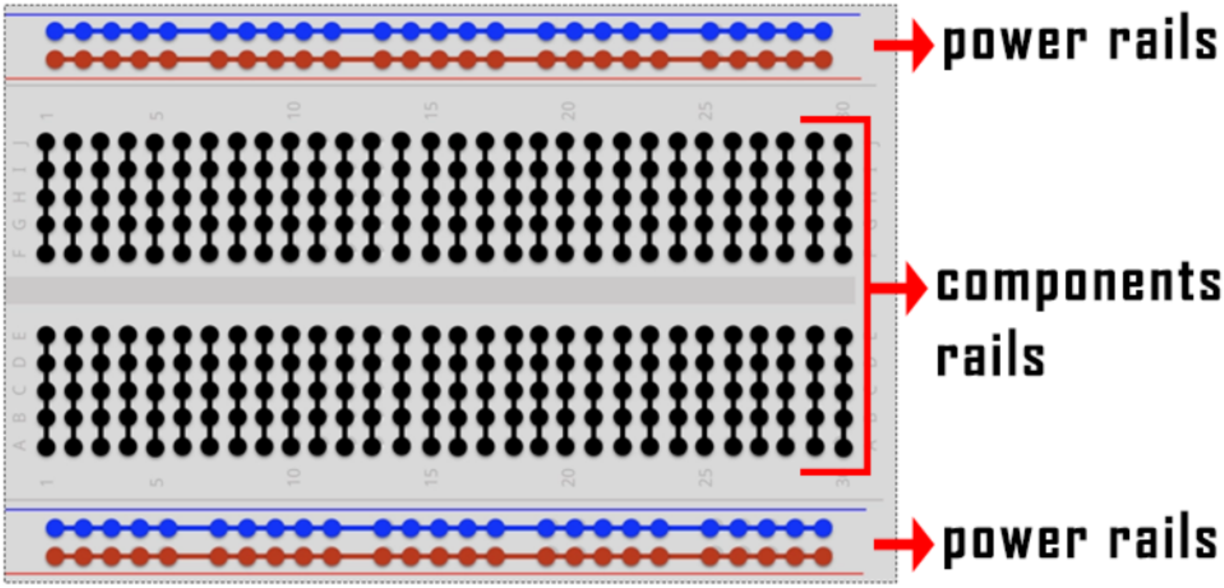
Breadboards are used to build and test circuits quickly before completing any circuit design. There are many holes in the breadboard that can be inserted into circuit components such as integrated circuits and resistors. A typical breadboard is shown below



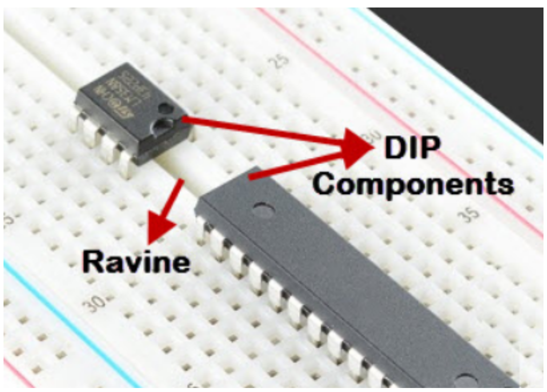
The breadboard has strips of metal , which run underneath the board and connect the holes on the top of the board. The metal strips are laid out as shown below. Note that the top and bottom rows of holes are connected horizontallywhile the remaining holes are connected vertically.

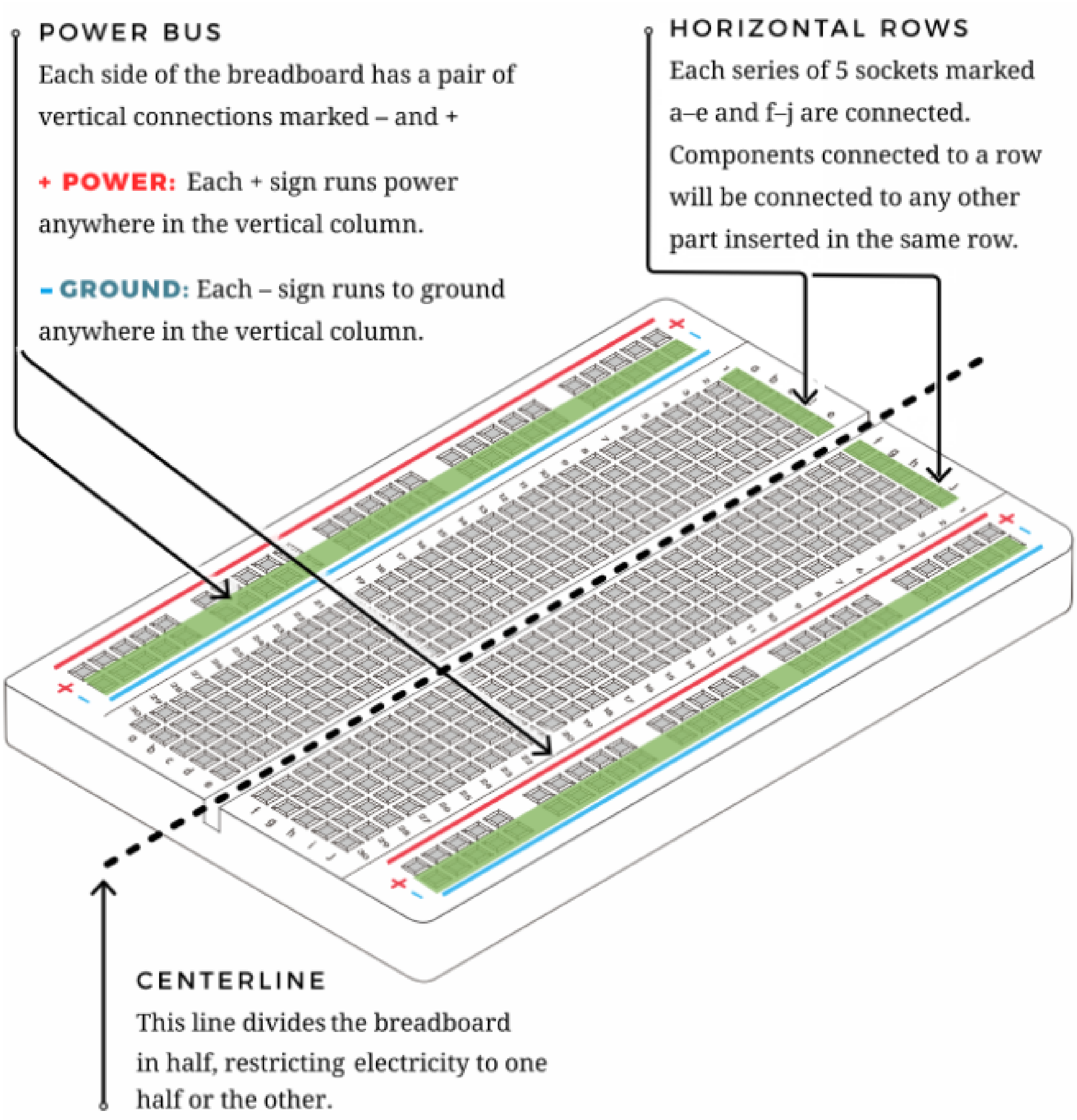


The first two rows (top) and the last two rows (bottom) of the breadboard are used for the positive pole (+) and negative pole (-) of the power supply respectively. The conductive layout of the breadboard is shown in the figure below:



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:





4Power Supply

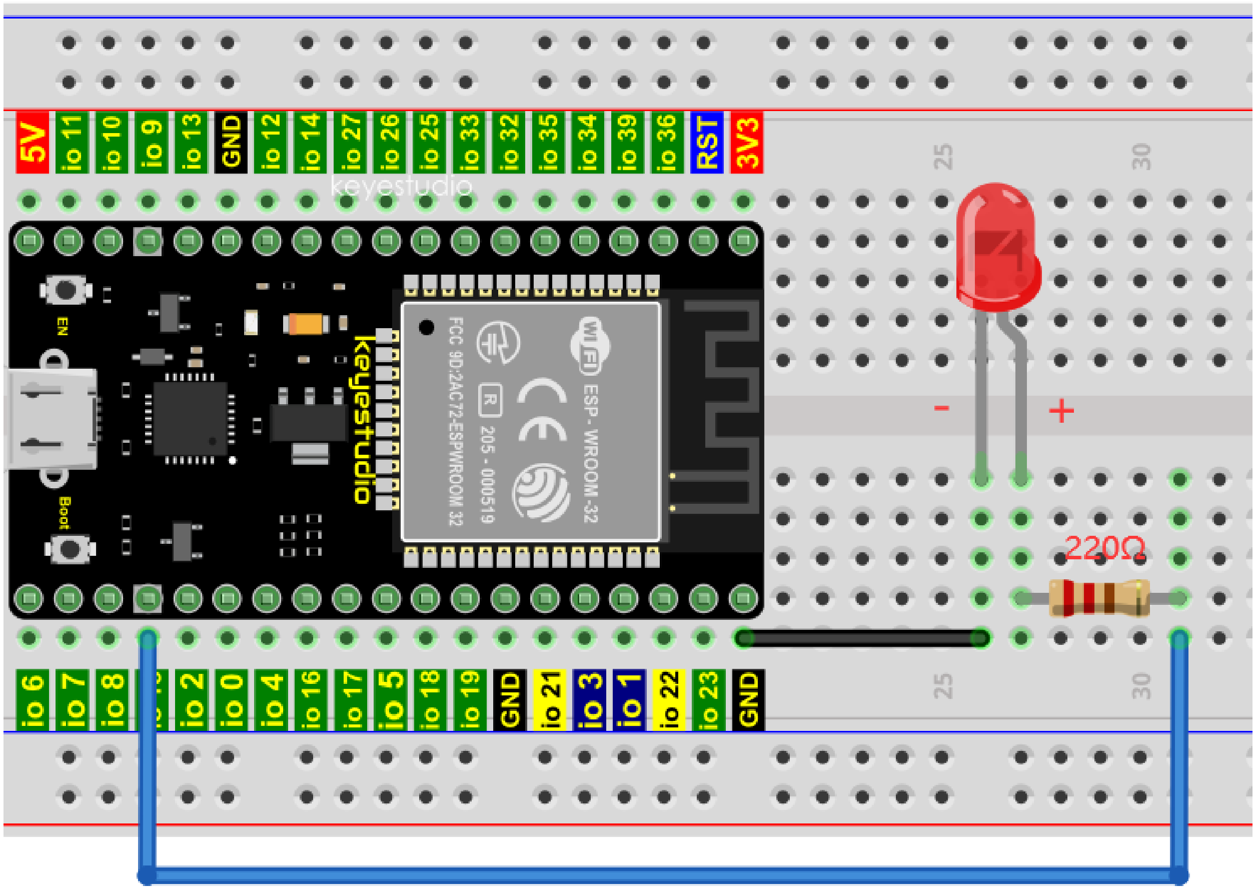
In this project, we connected the ESP32 to the Raspberry Pi by using USB cable.

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 the Raspberry Pi by using a USB cable.

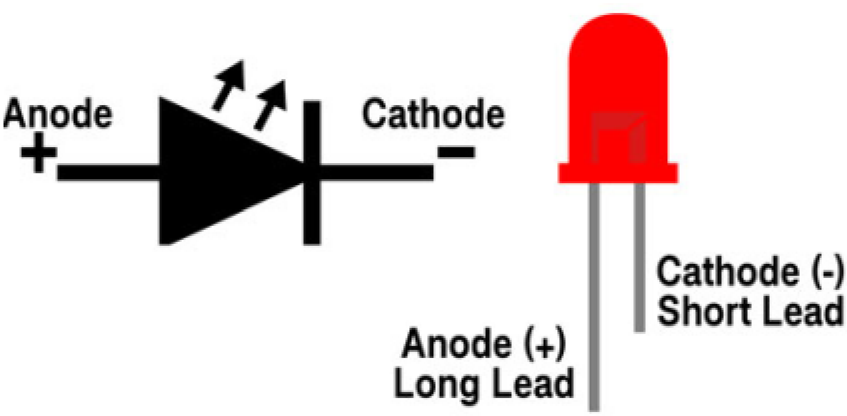
Note: Avoid any possible short circuits (especially connecting 3.3V and GND)!

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

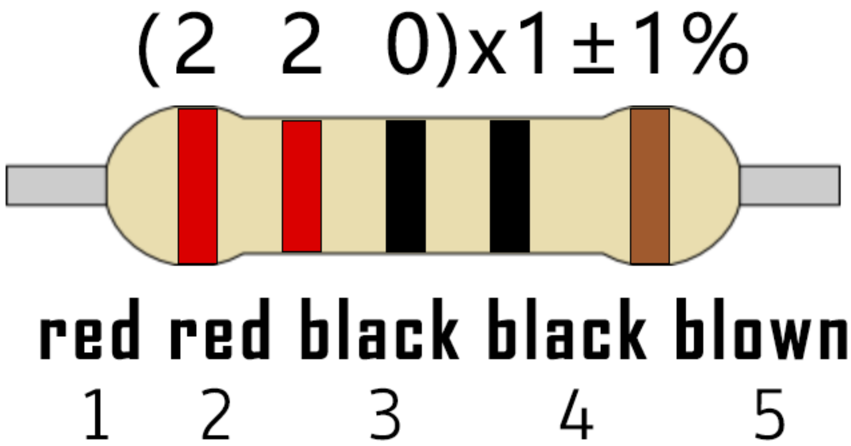


Note:

How to connect a LED



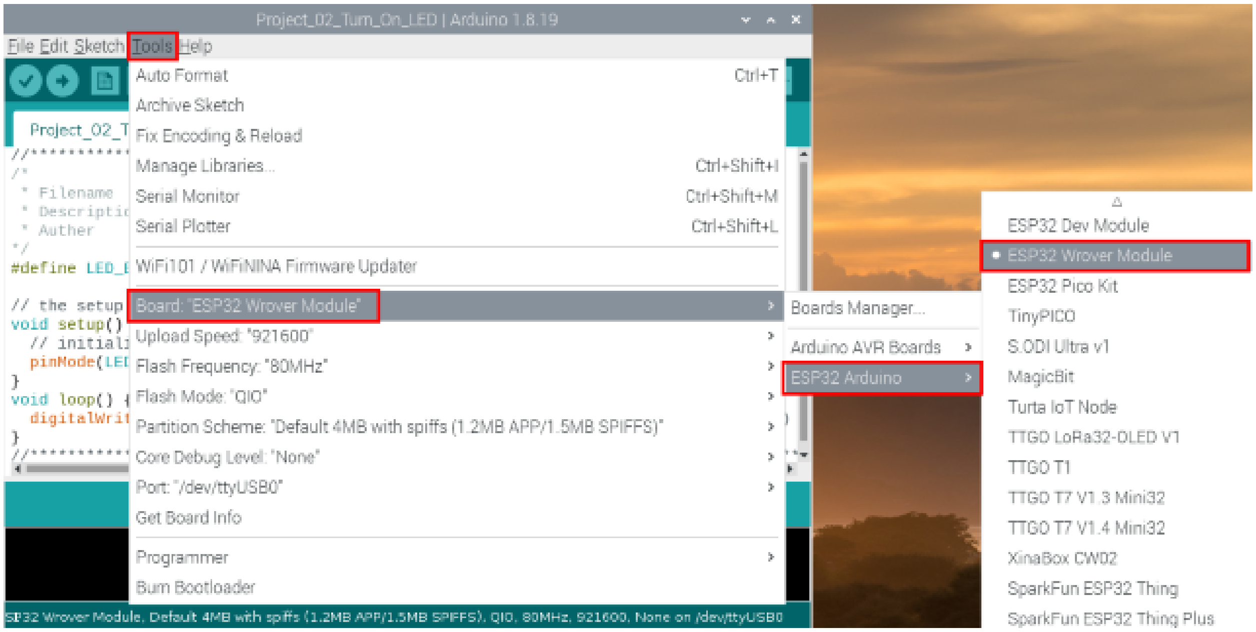
How to identify the 220 Five-color ring resistor



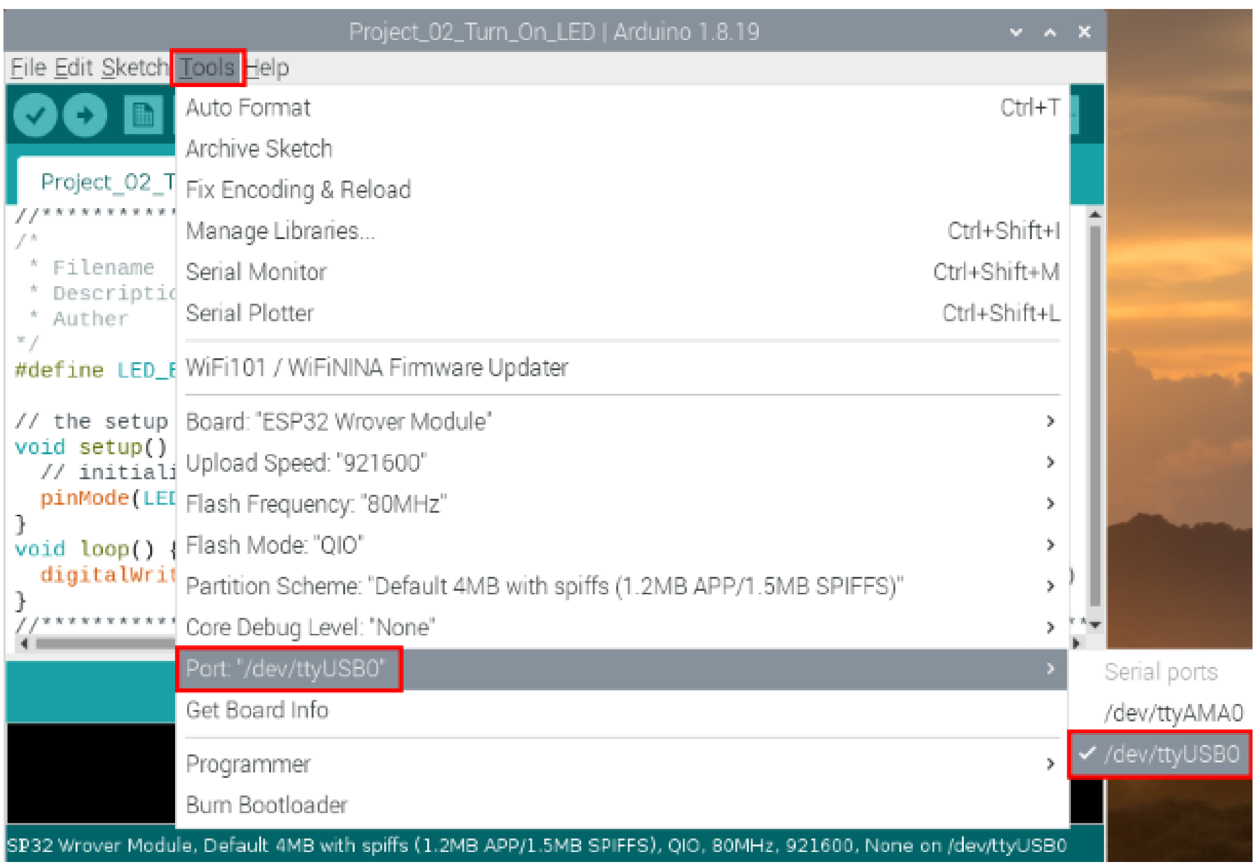
5.Project code

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

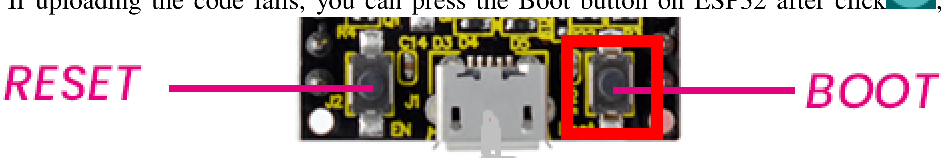
Before uploading the project code to ESP32click “Tools”→“Board” and select“ESP32 Wrover Module”.



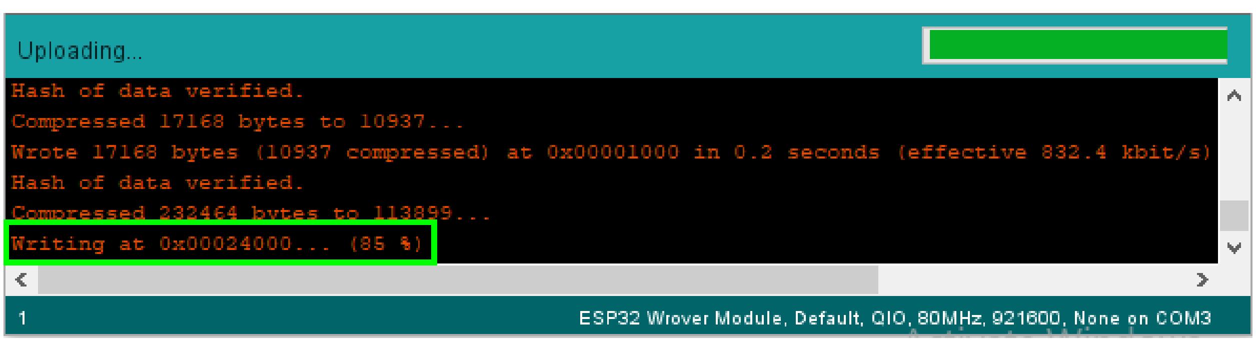
Select the serial port.



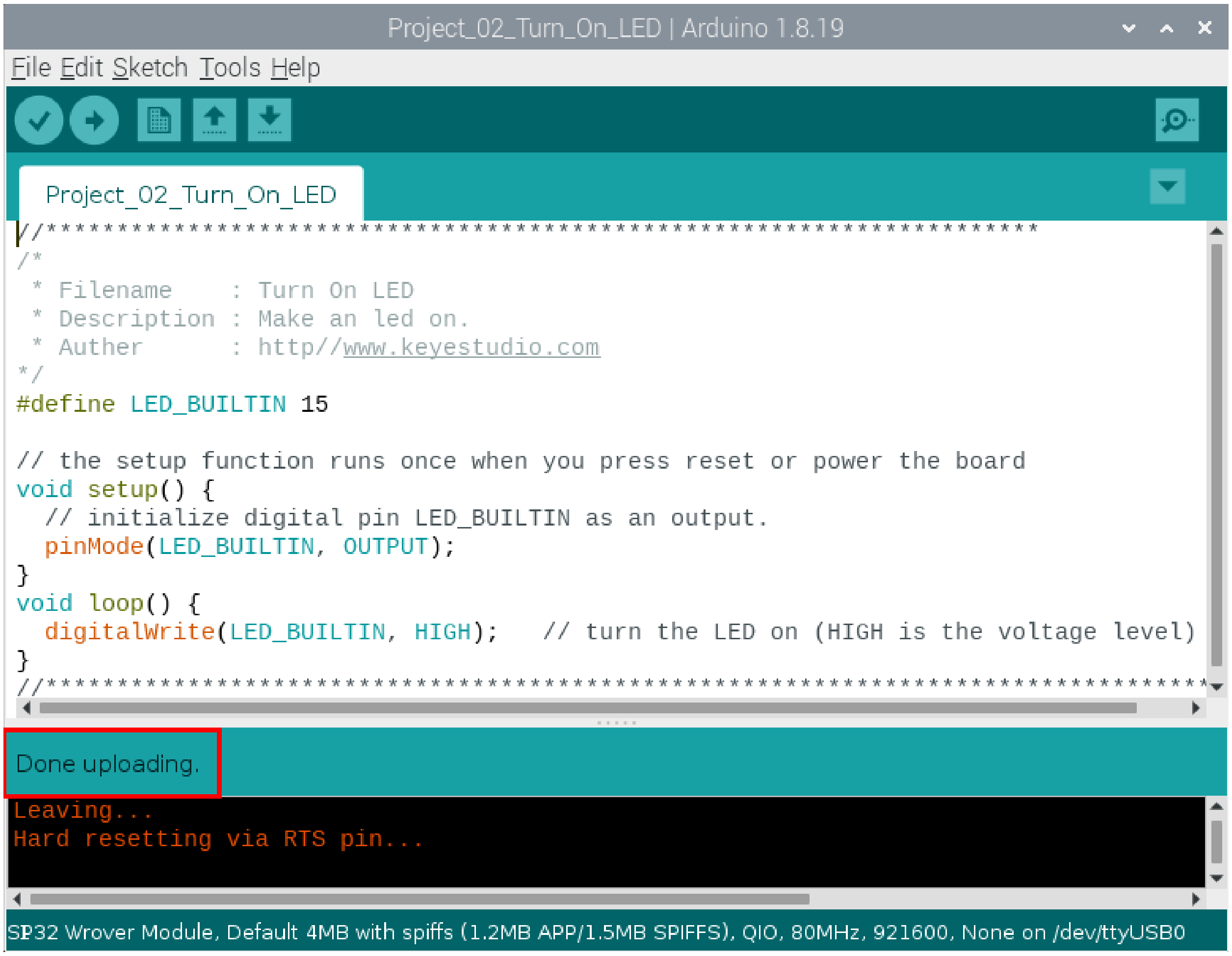
Click  to download the code to ESP32.

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:

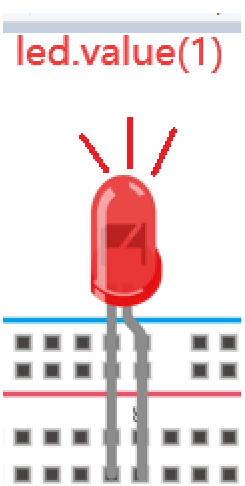


The Project code is uploaded successfully



6.Project result

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

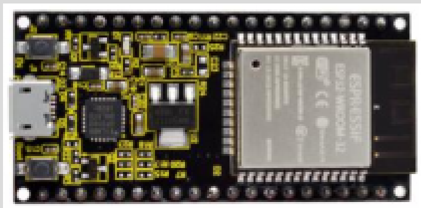
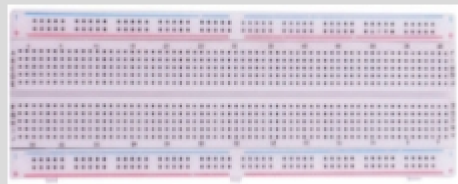
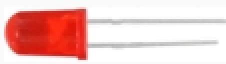
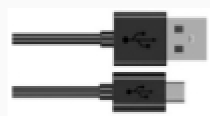


9.3 Project 03LED Flashing

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

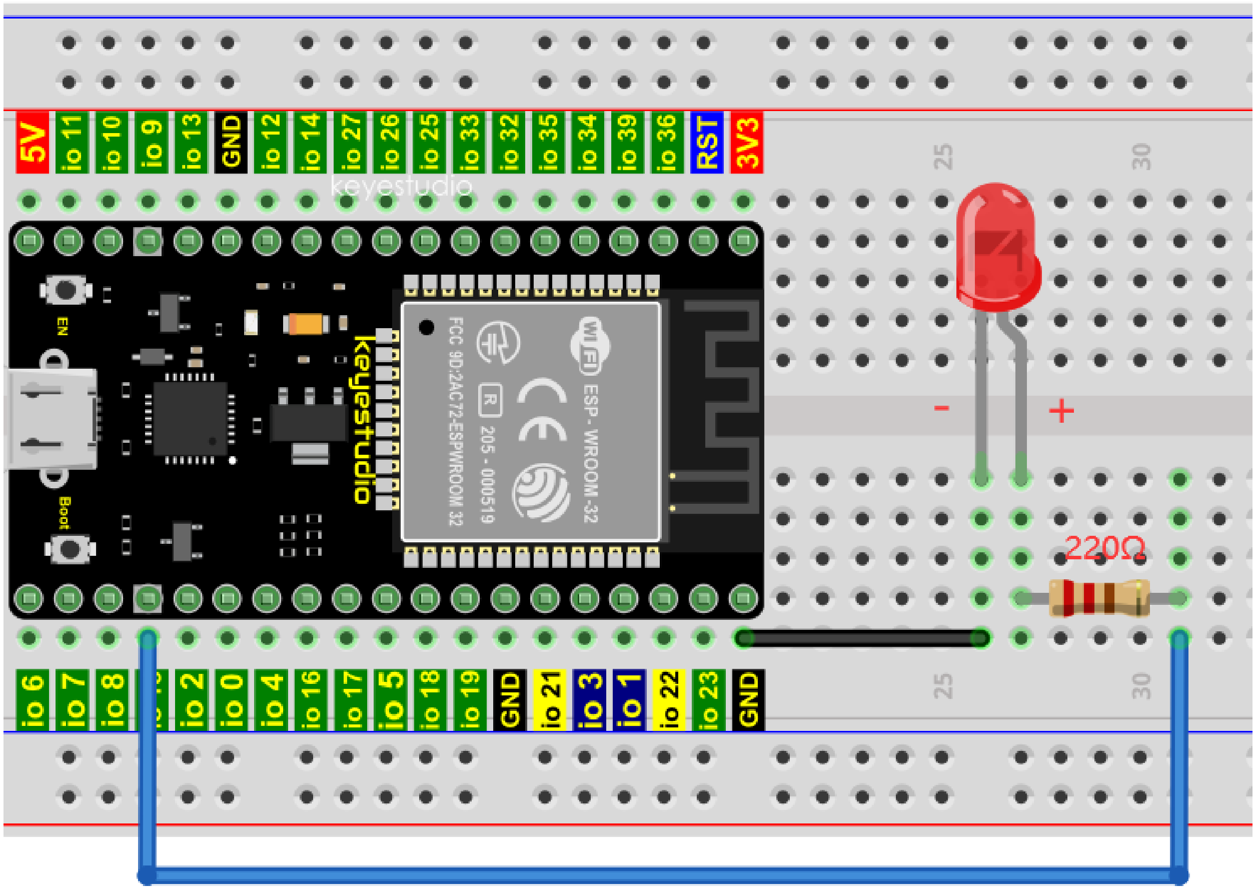
	
ESP32*1	Breadboard*1
	
Red LED*1	220 Resistor*1
	
Jumper Wire*2	USB Cable*1

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.

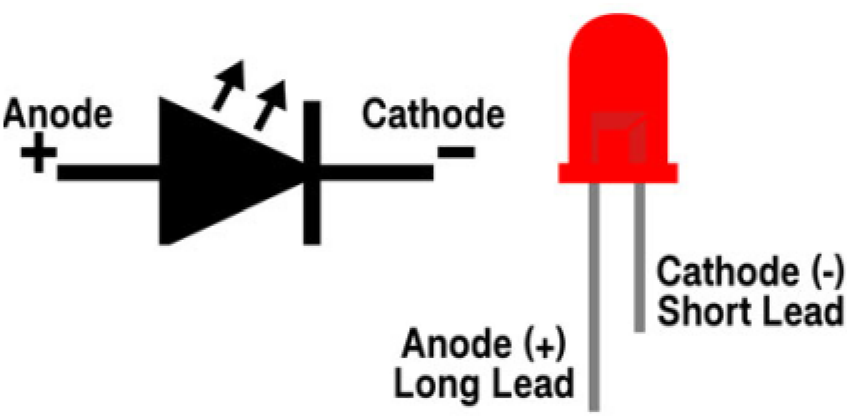
Note: Avoid any possible short circuits (especially connecting 3.3V and GND)!

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

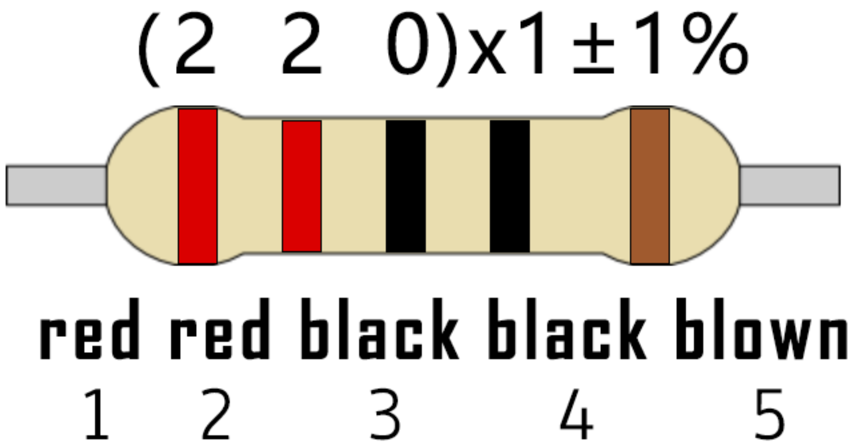


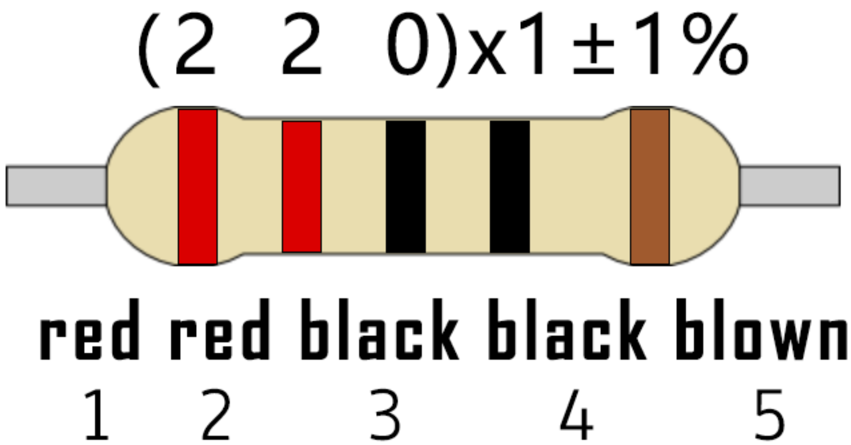
Note:

How to connect a LED



How to identify the 220 Five-color ring resistor





4.Test Code

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

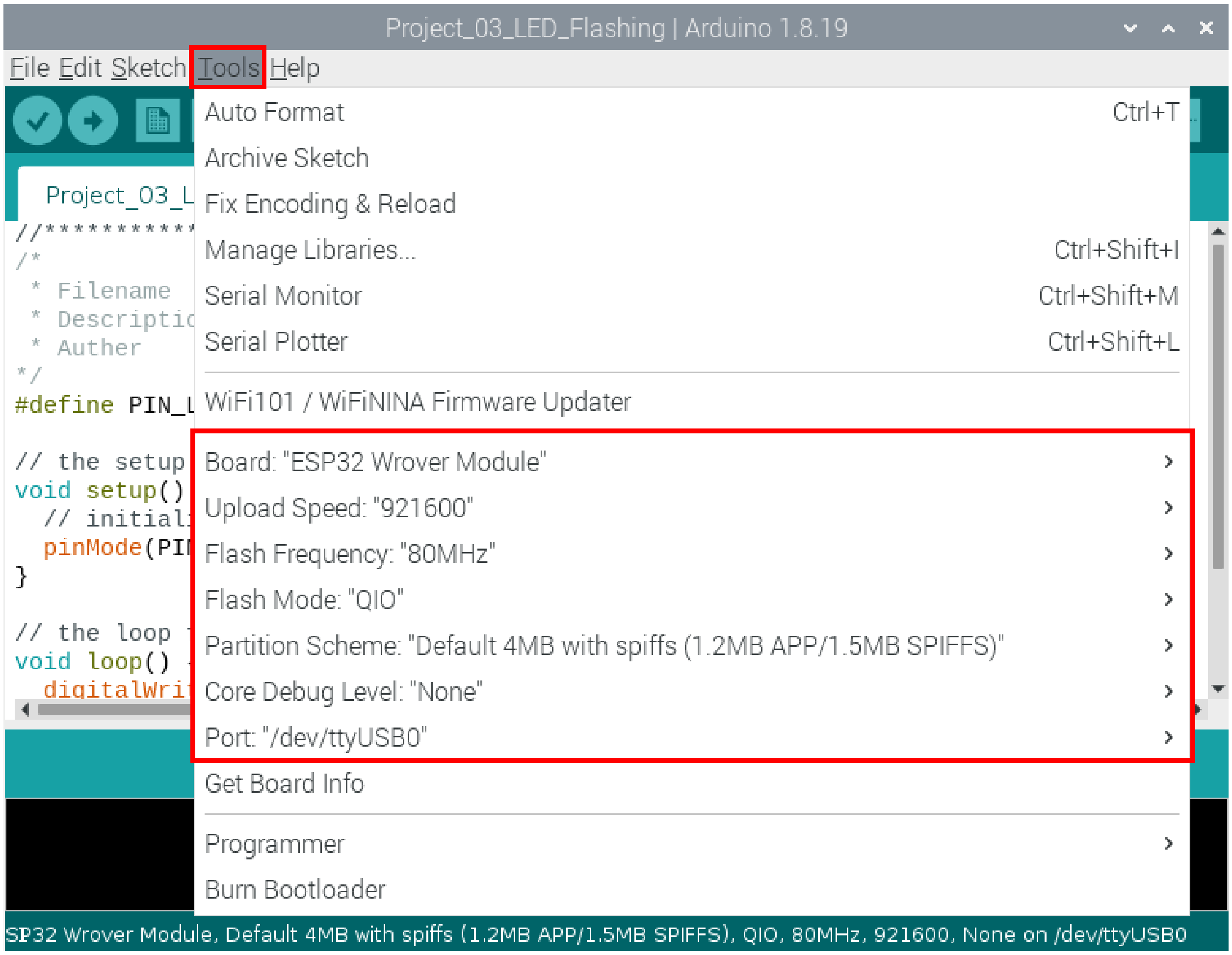
(continues on next page)

(continued from previous page)

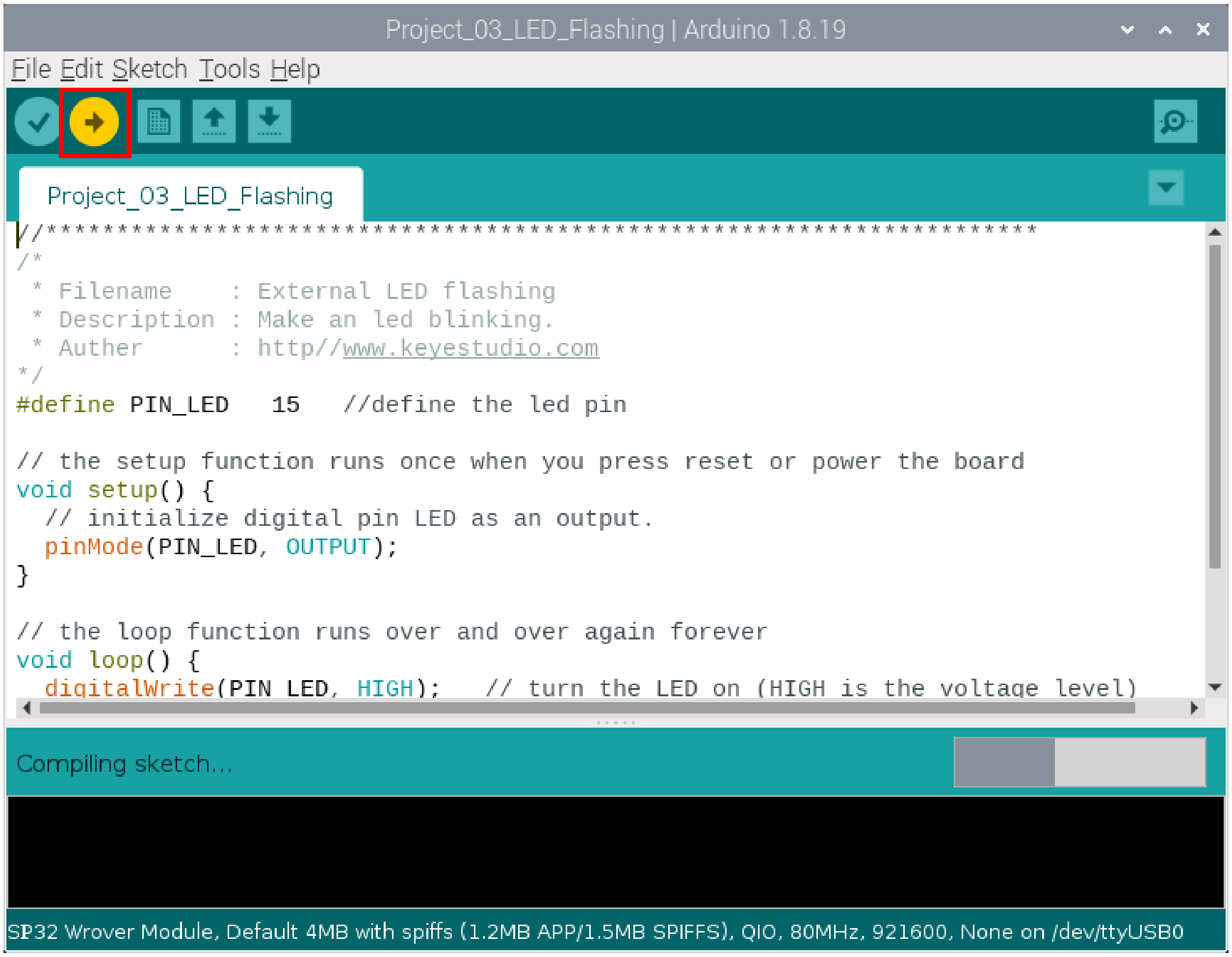
```
}
//*****************************************************************************************
```

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

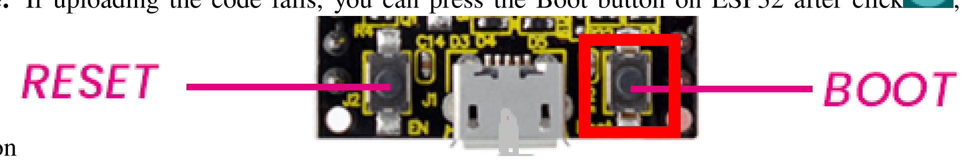
Click "**Tools**" to confirm the board type and port as shown below:

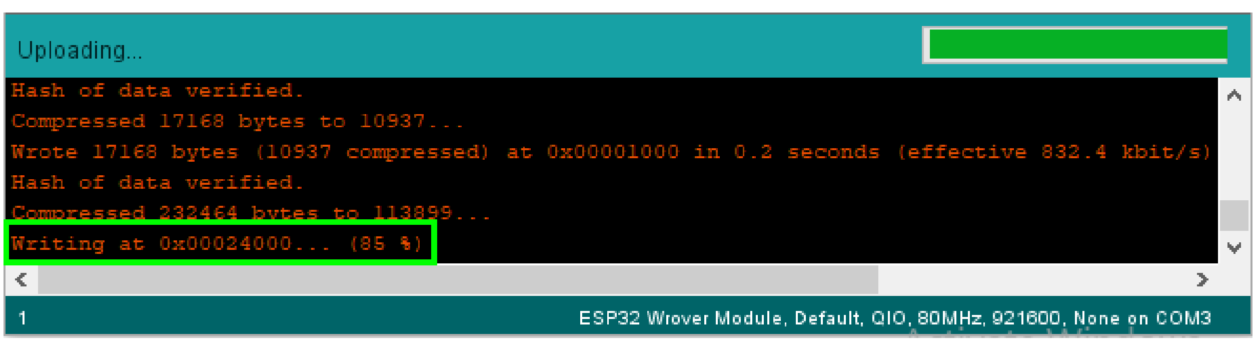


Click  to download the project code to ESP32.

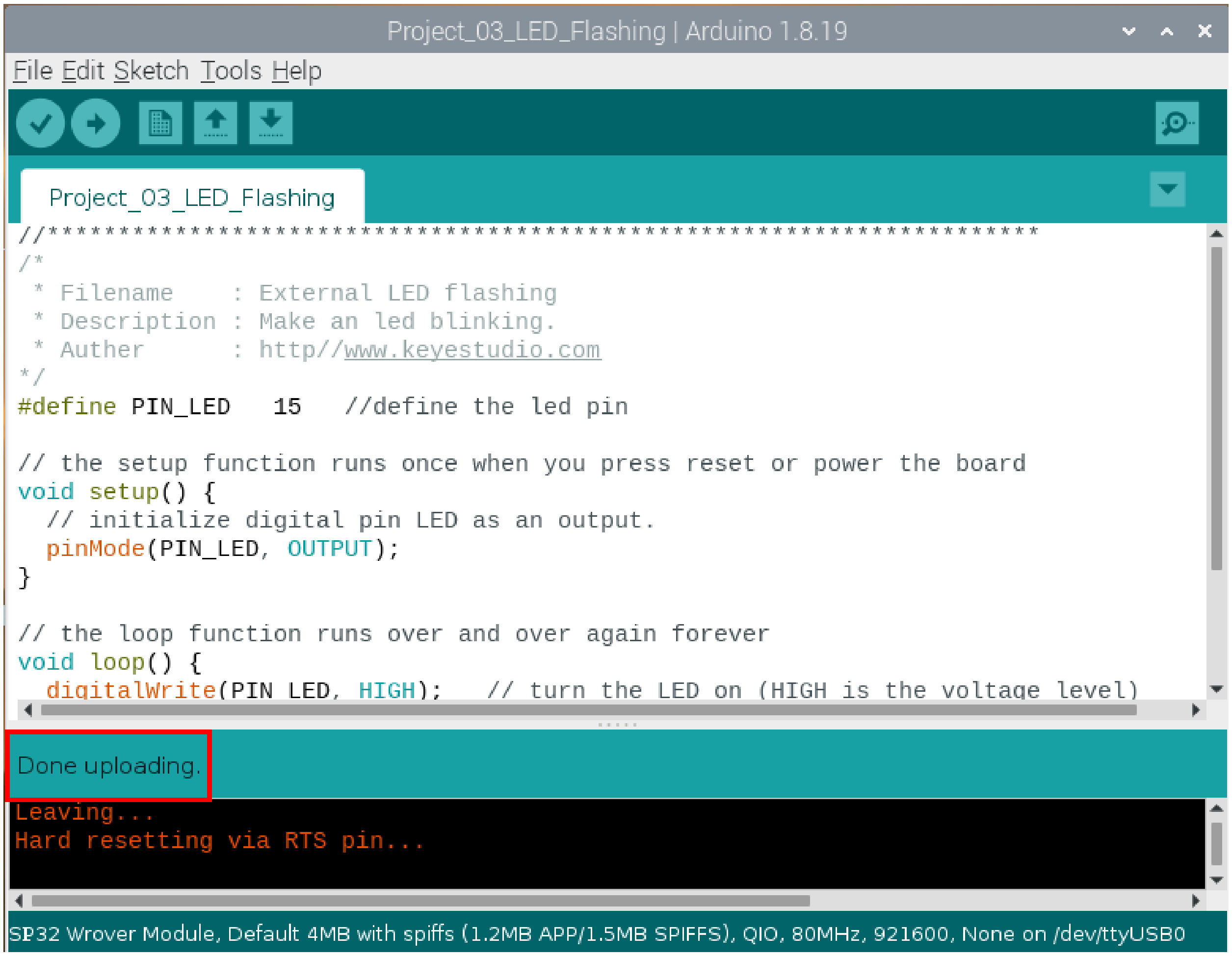


Note: If uploading the code fails, you can press the Boot button on ESP32 after click , and release the Boot button after the percentage of uploading progress appears, as shown below:



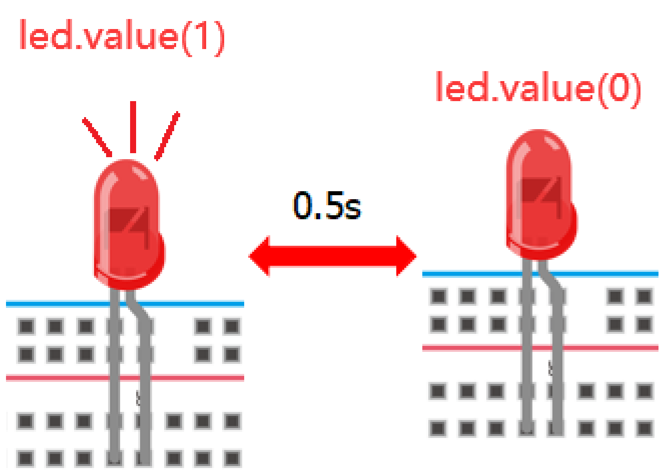


The Project code is uploaded successfully



5.Project result

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



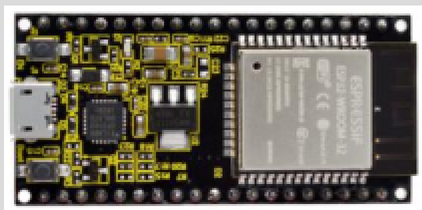
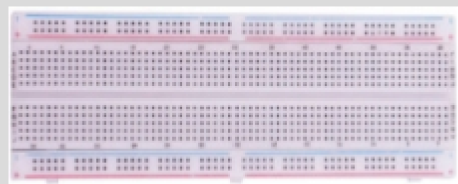
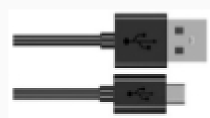
9.4 Project 04: Breathing Led

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.

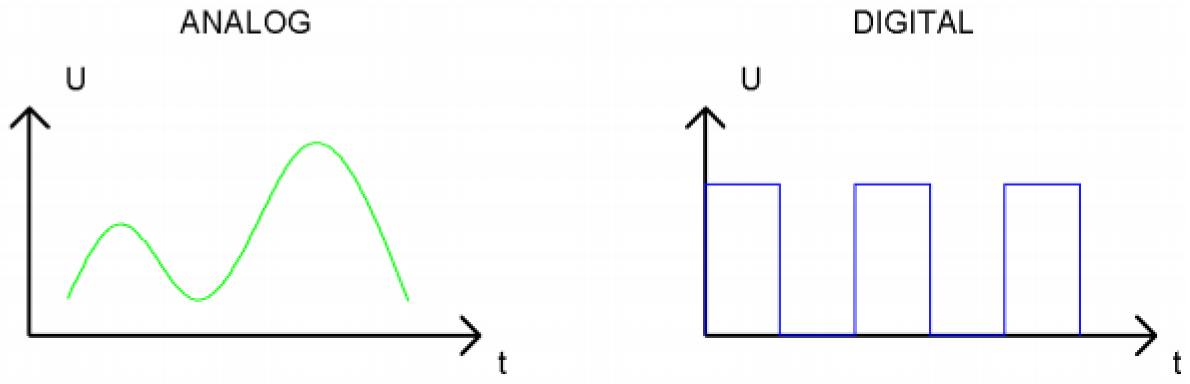
2.Components

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

2.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.



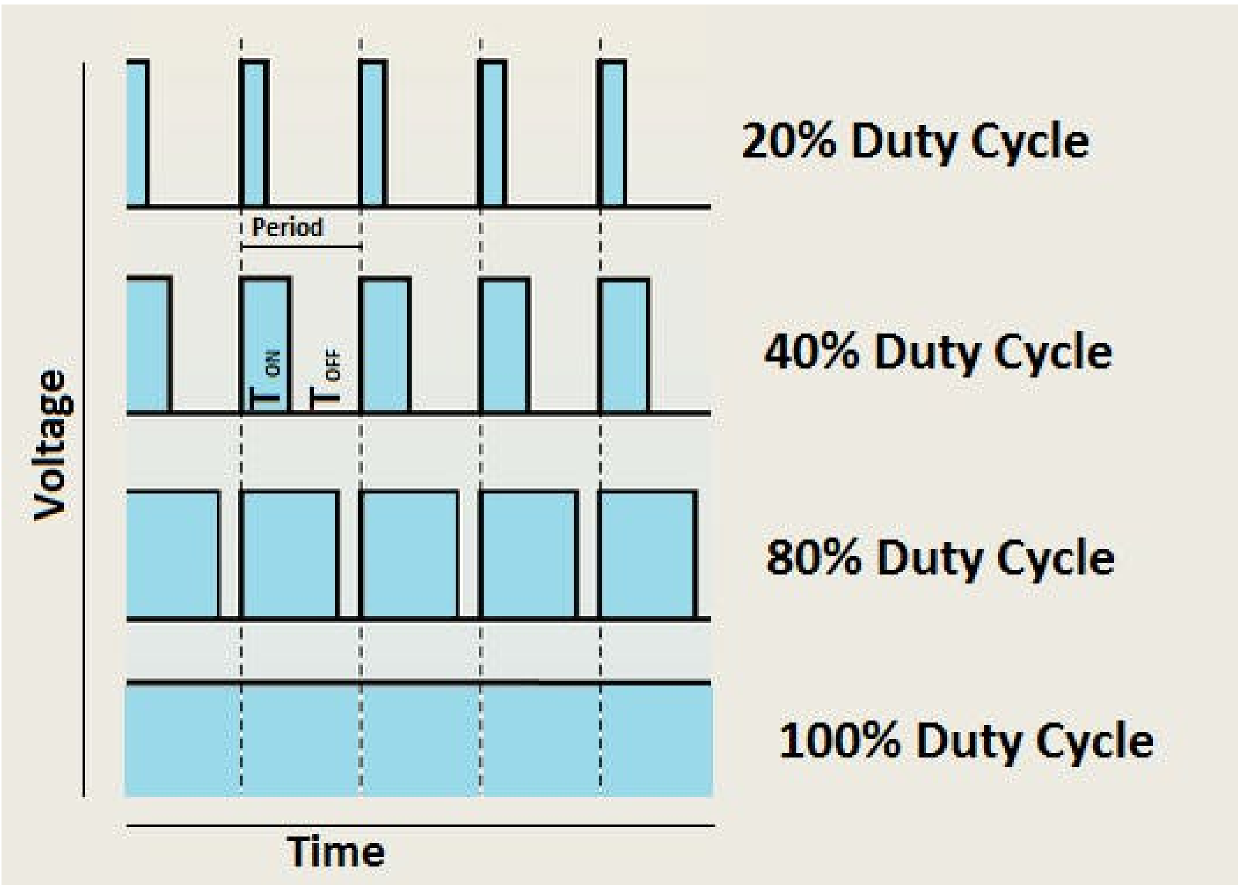
In practical application, we often use binary as the digital signal, that is a series of 0's and 1's. Since a binary signal only has two values (0 or 1), it has great stability and reliability. Lastly, both analog and digital signals can be converted into the other.

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:

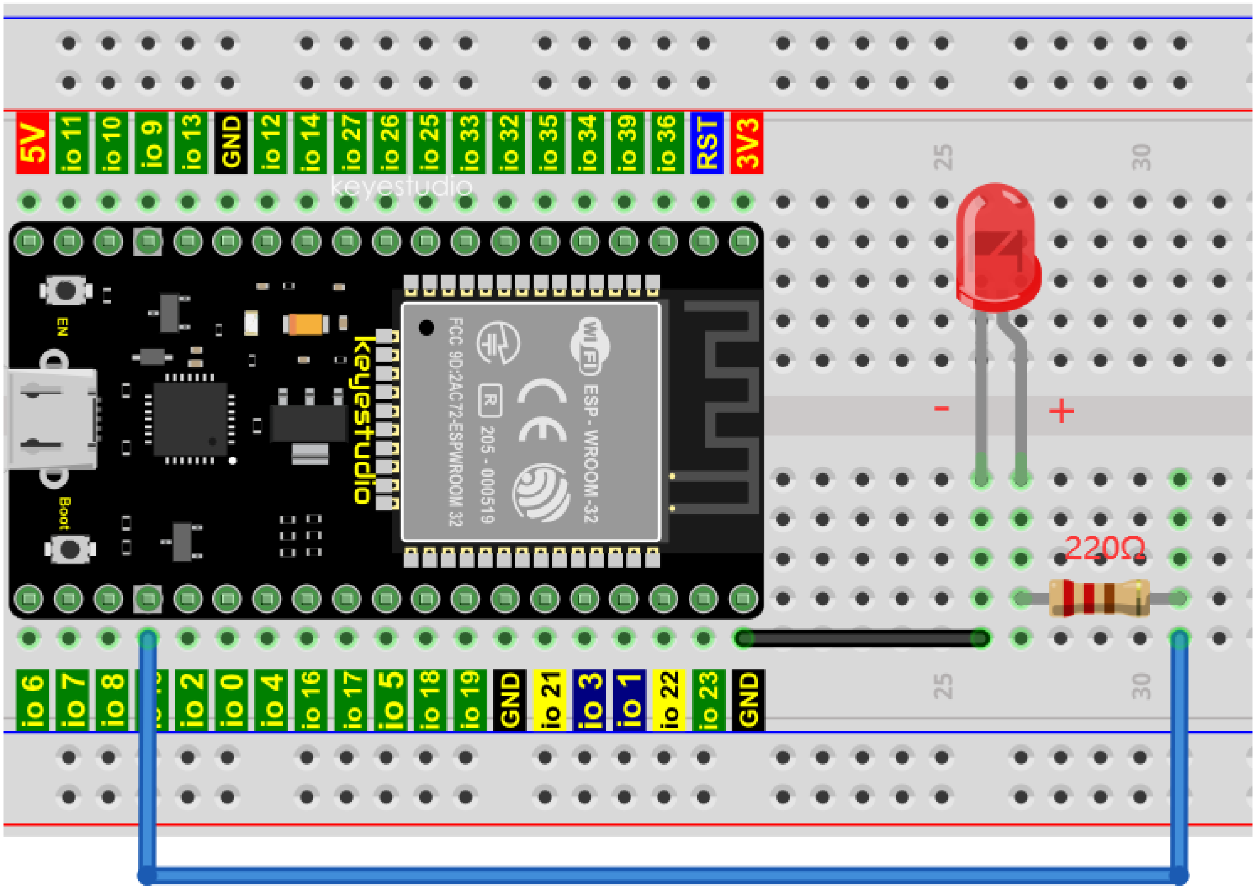
On ESP32, the LEDC(PWM) controller has 16 separate channels, each of which can independently control frequency, duty cycle, and even accuracy. Unlike traditional PWM pins, the PWM output pins of ESP32 are configurable, with one or more PWM output pins per channel. The relationship between the maximum

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

$$Freq_{max} = \frac{80{,}000{,}000}{1 \ll bit}$$

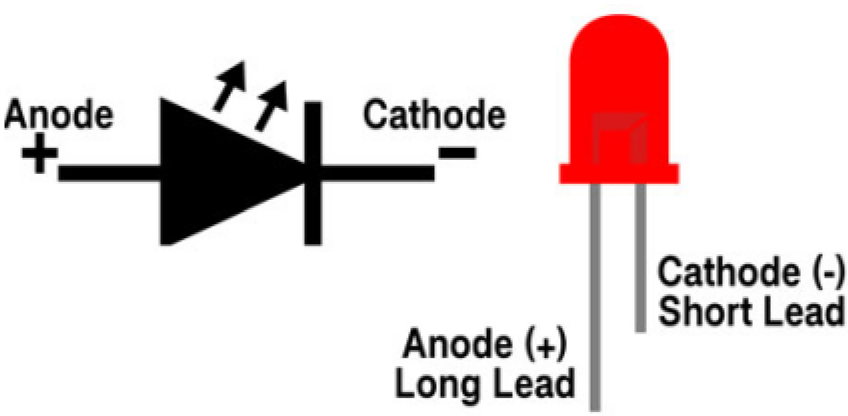
For example, generate a PWM with an 8-bit precision (28=256. Values range from 0 to 255) with a maximum frequency of 80,000,000/255 =312,500Hz.

3.Wiring diagram

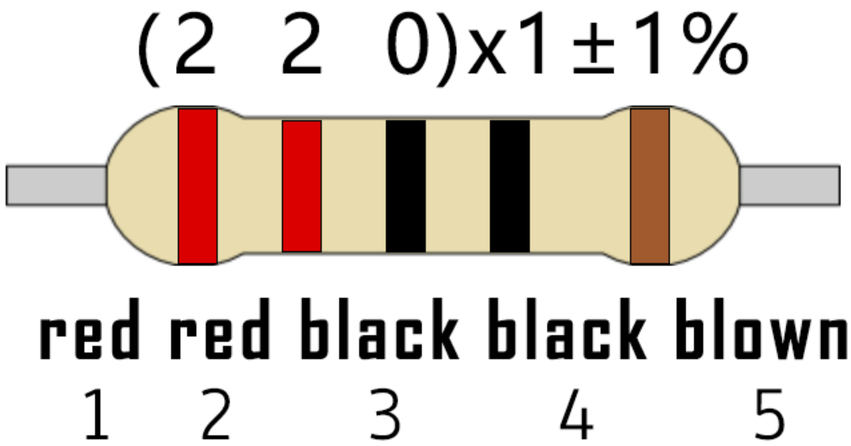


Note:

How to connect a LED



How to identify the 220 Five-color ring resistor



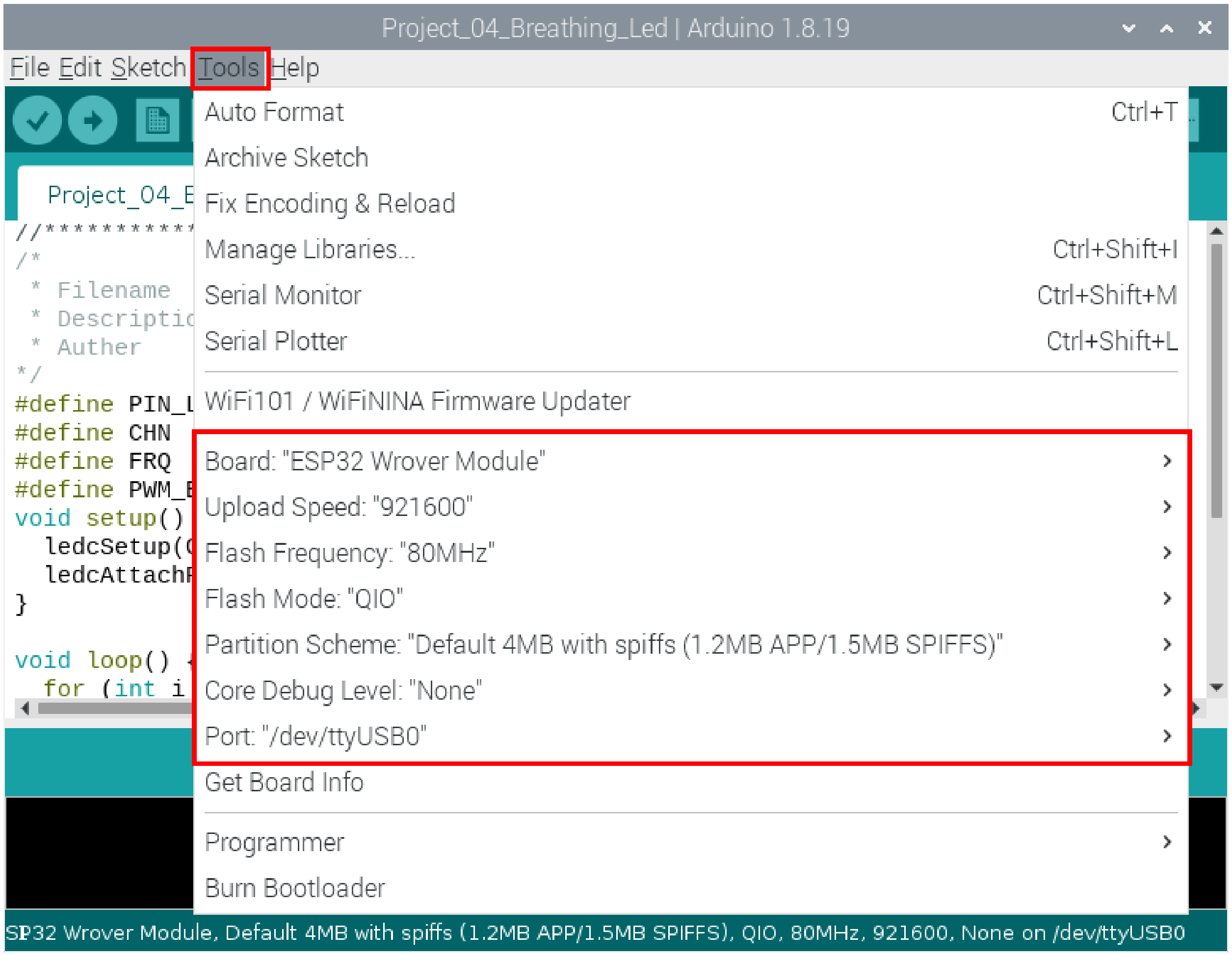
4.Project code

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%.

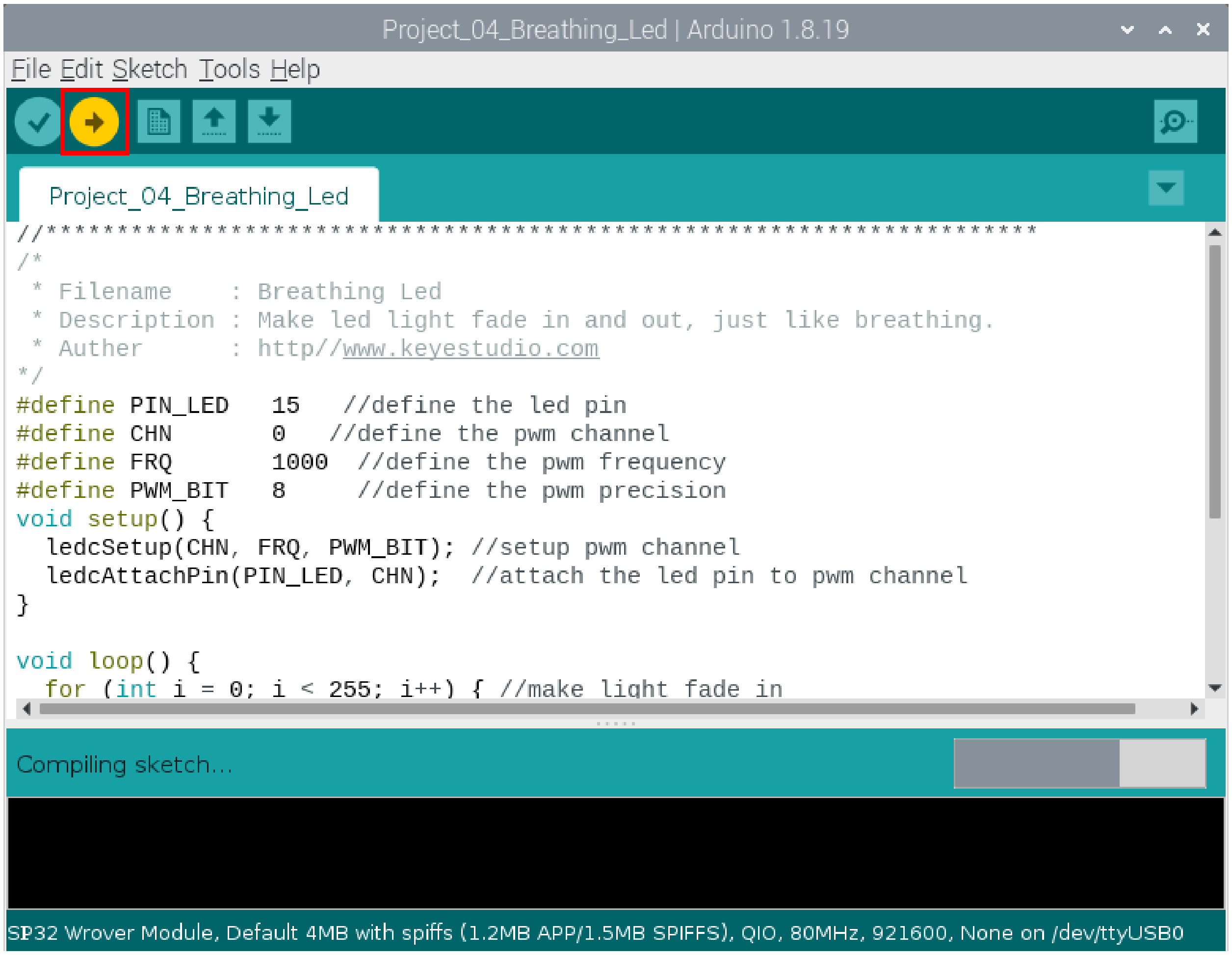
```
//**********************************************************************
/*
 * Filename    : Breathing Led
 * Description : Make led light fade in and out, just like breathing.
 * Auther      : http//www.keyestudio.com
*/
#define PIN_LED   15   //define the led pin
#define CHN       0   //define 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.

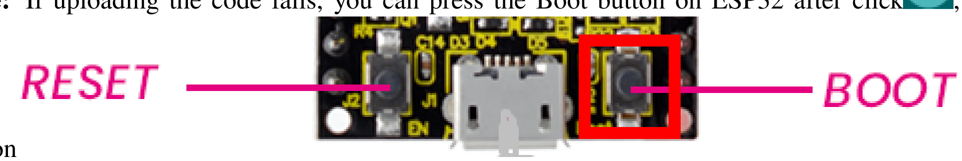
Click "**Tools**" to confirm the board type and port as shown below:



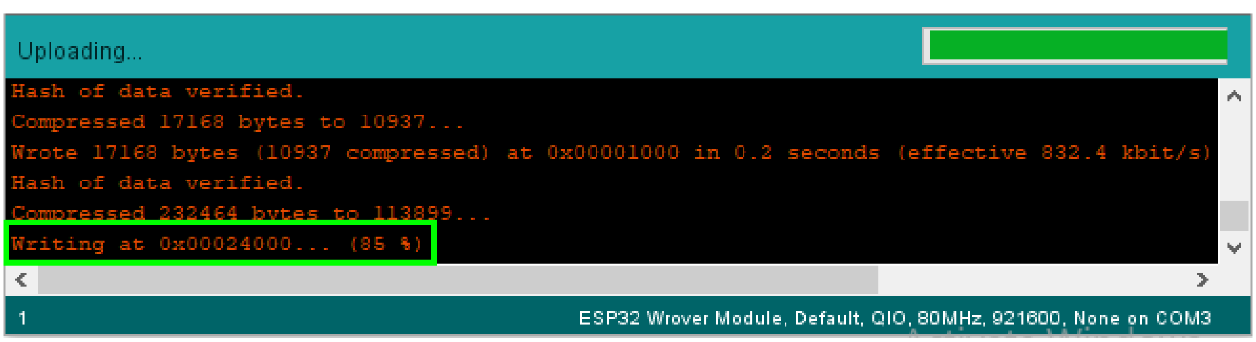
Click  to download the project code to ESP32.



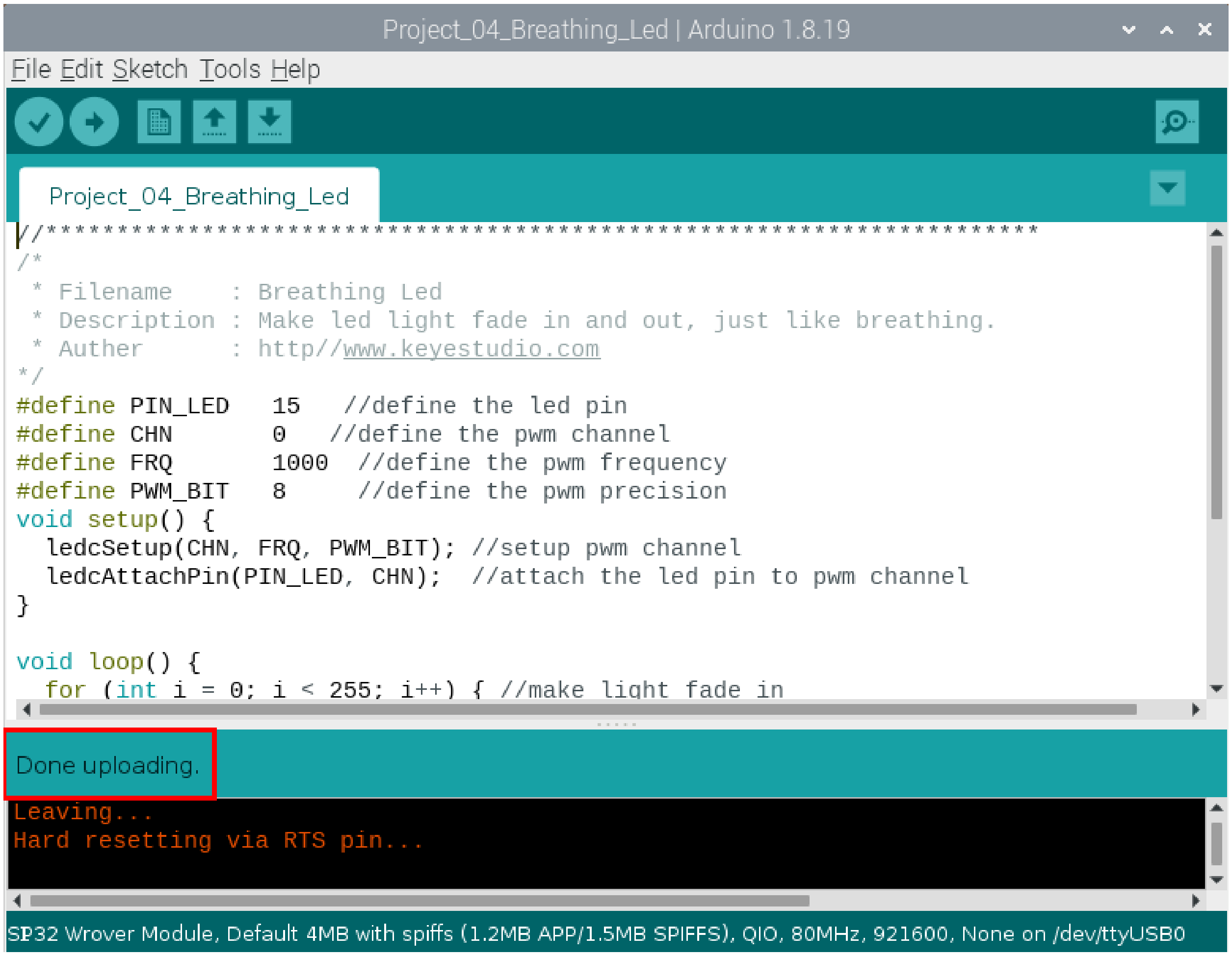
Note: If uploading the code fails, you can press the Boot button on ESP32 after click , and release the Boot button



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



The Project code is uploaded successfully



5.Project result

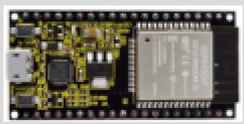
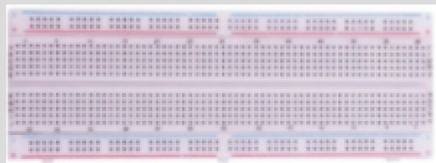
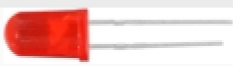
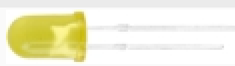
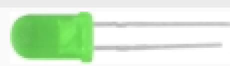
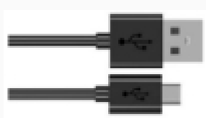
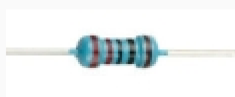
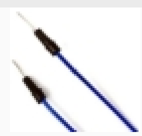
After the project code was uploaded successfully, power up with a USB cable and the LED is turned from ON to OFF and then back from OFF to ON gradually like breathing.

9.5 Project 05Traffic Lights

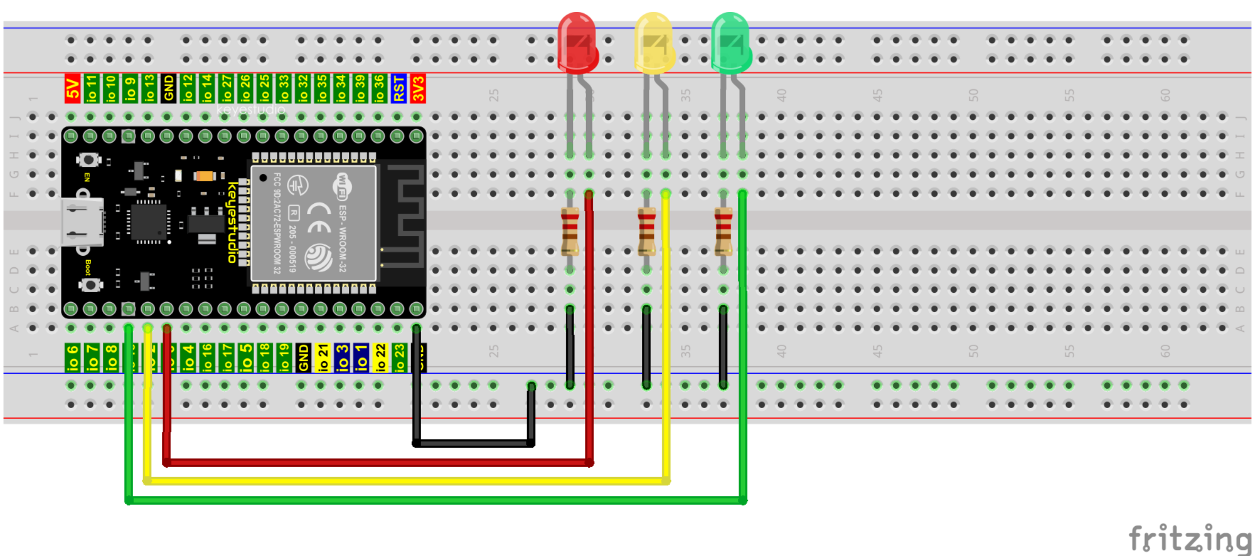
1.Introduction

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

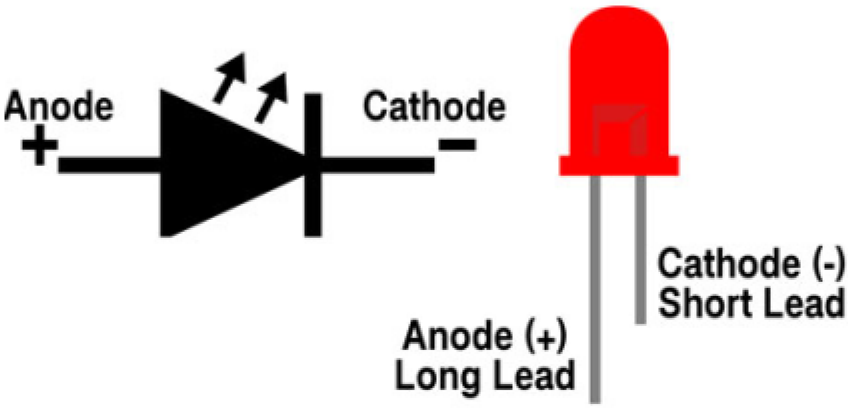
		
ESP32*1	Bread board*1	
		
Red LED*1	Yellow LED*1	Green LED*1
		
USB Cable*1	220 Resistor*3	Jumper Wires

3.Wiring diagram

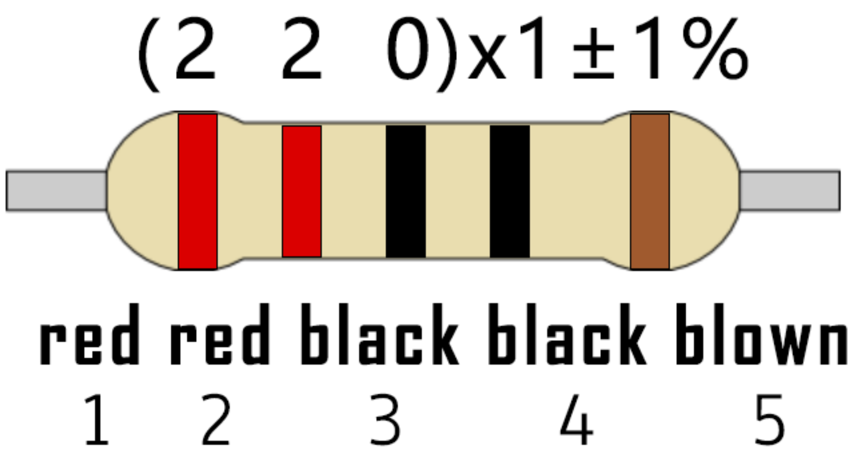


Note:

How to connect a LED



How to identify the 220 Five-color ring resistor



4.Test Code

```
//**********************************************************************
/*
 * 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_GREEN, HIGH);// turns on the green led
   delay(5000);// delays 5 seconds
   digitalWrite(PIN_LED_GREEN, LOW); // turns off the green led
```

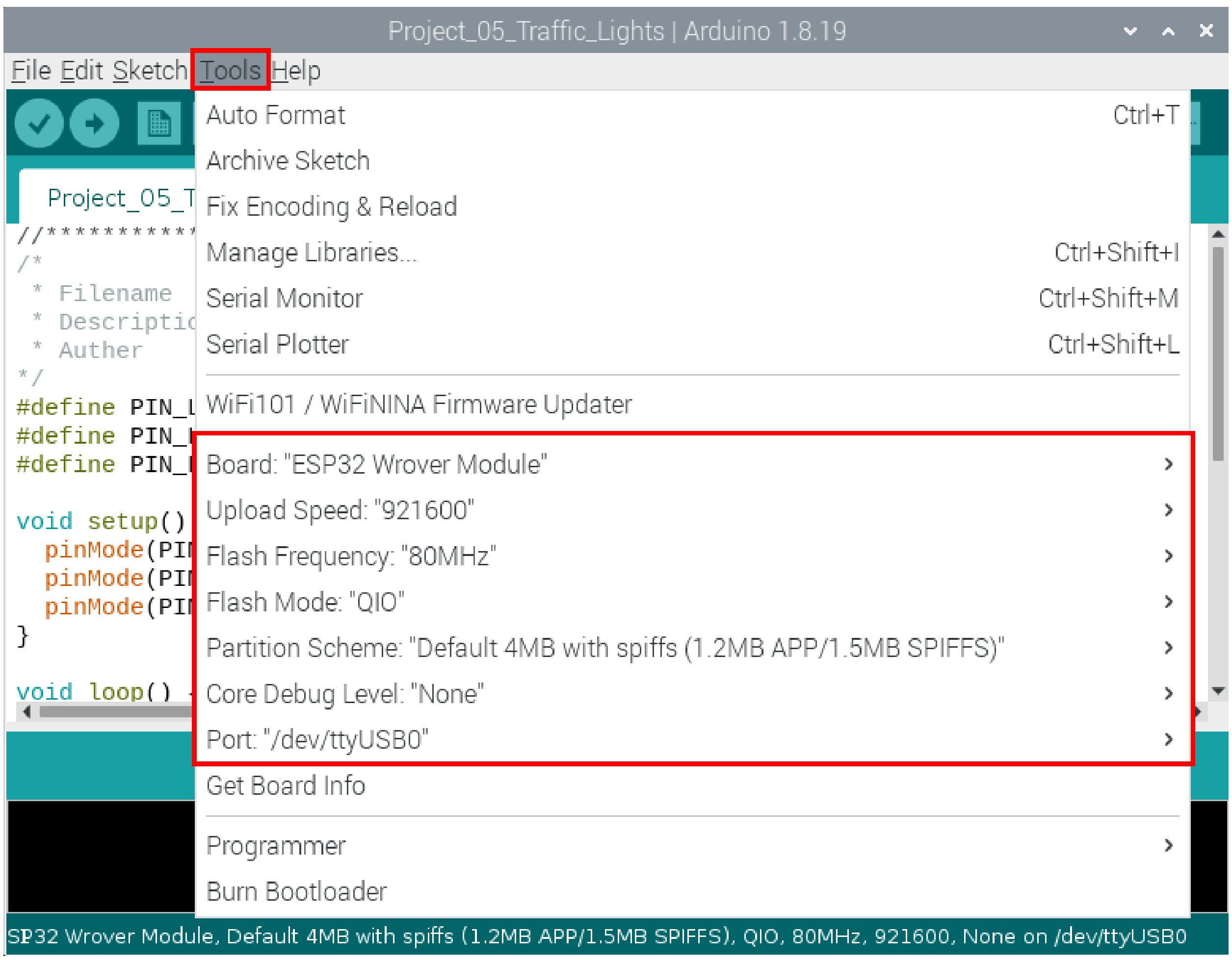
(continues on next page)

(continued from previous page)

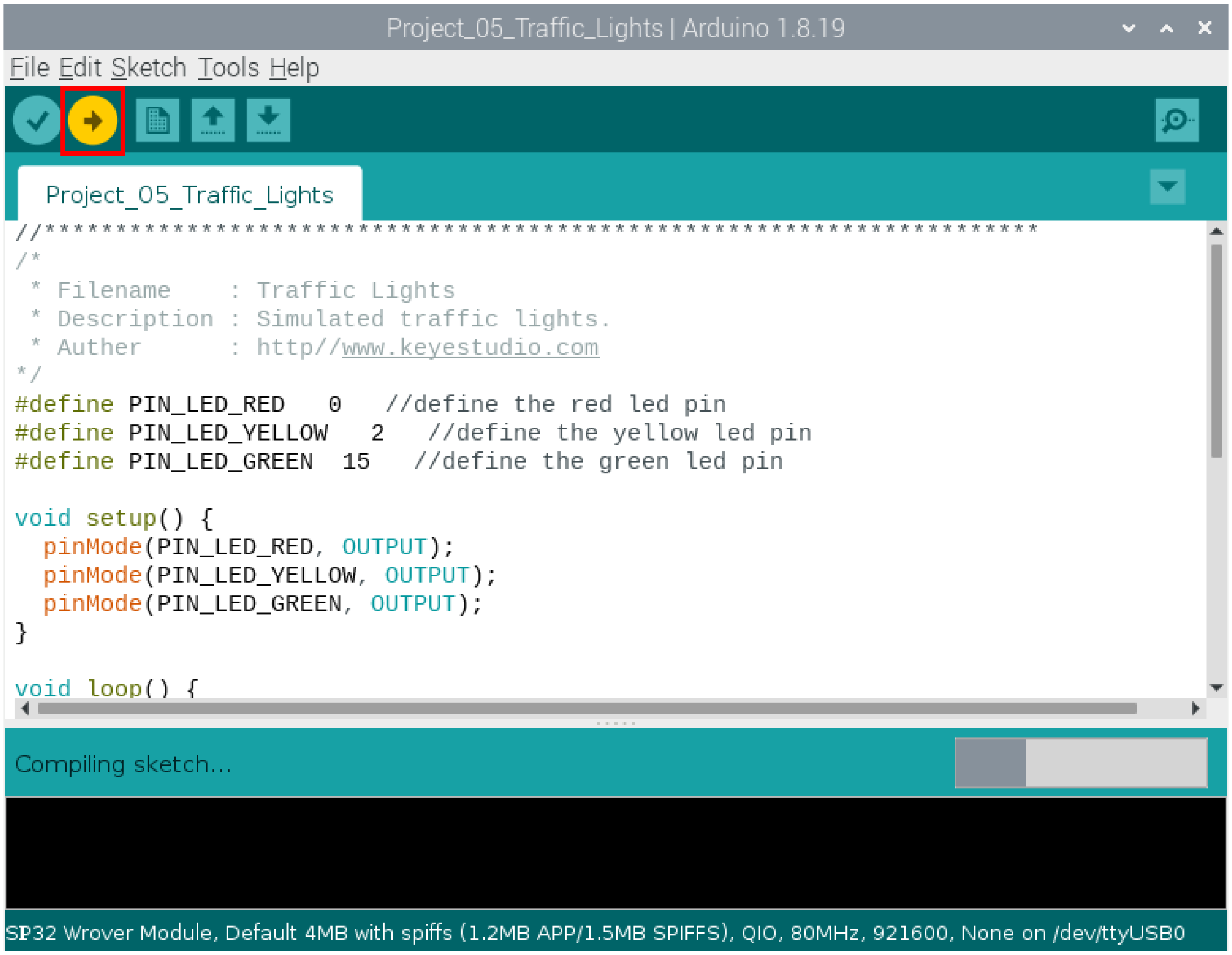
```
    for(int i=0;i<3;i++)// flashes 3 times.
{
    delay(500);// delays 0.5 second
    digitalWrite(PIN_LED_YELLOW, HIGH);// turns on the yellow led
    delay(500);// delays 0.5 second
    digitalWrite(PIN_LED_YELLOW, LOW);// turns off the yellow led
}
    delay(500);// delays 0.5 second
    digitalWrite(PIN_LED_RED, HIGH);// turns on the red led
    delay(5000);// delays 5 second
    digitalWrite(PIN_LED_RED, LOW);// turns off the red led
}
//*************************************************************************************
```

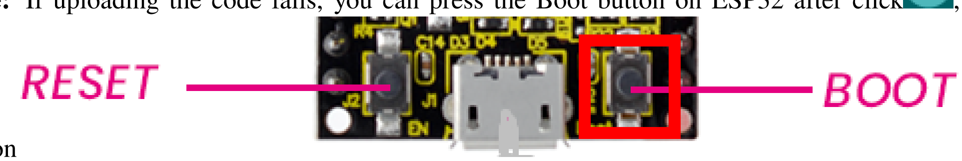
Before uploading Project Code to ESP32, please check the configuration of Arduino IDE.

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

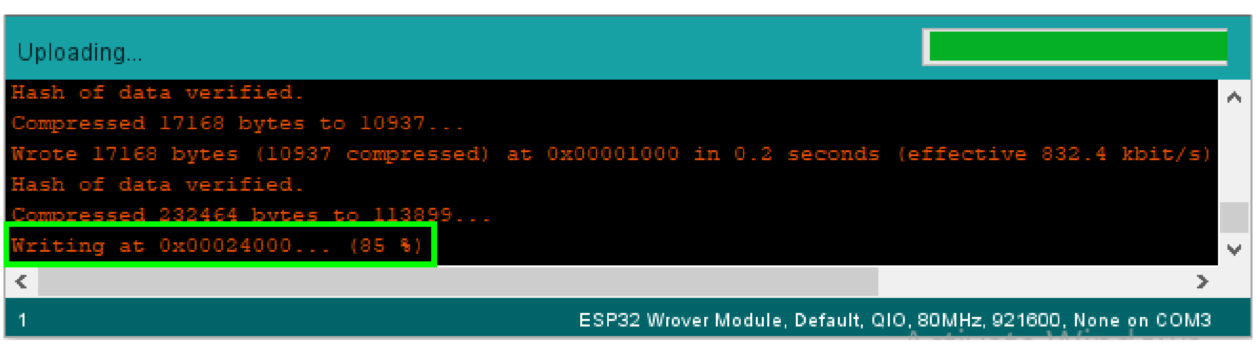


Click  to download the project code to ESP32.

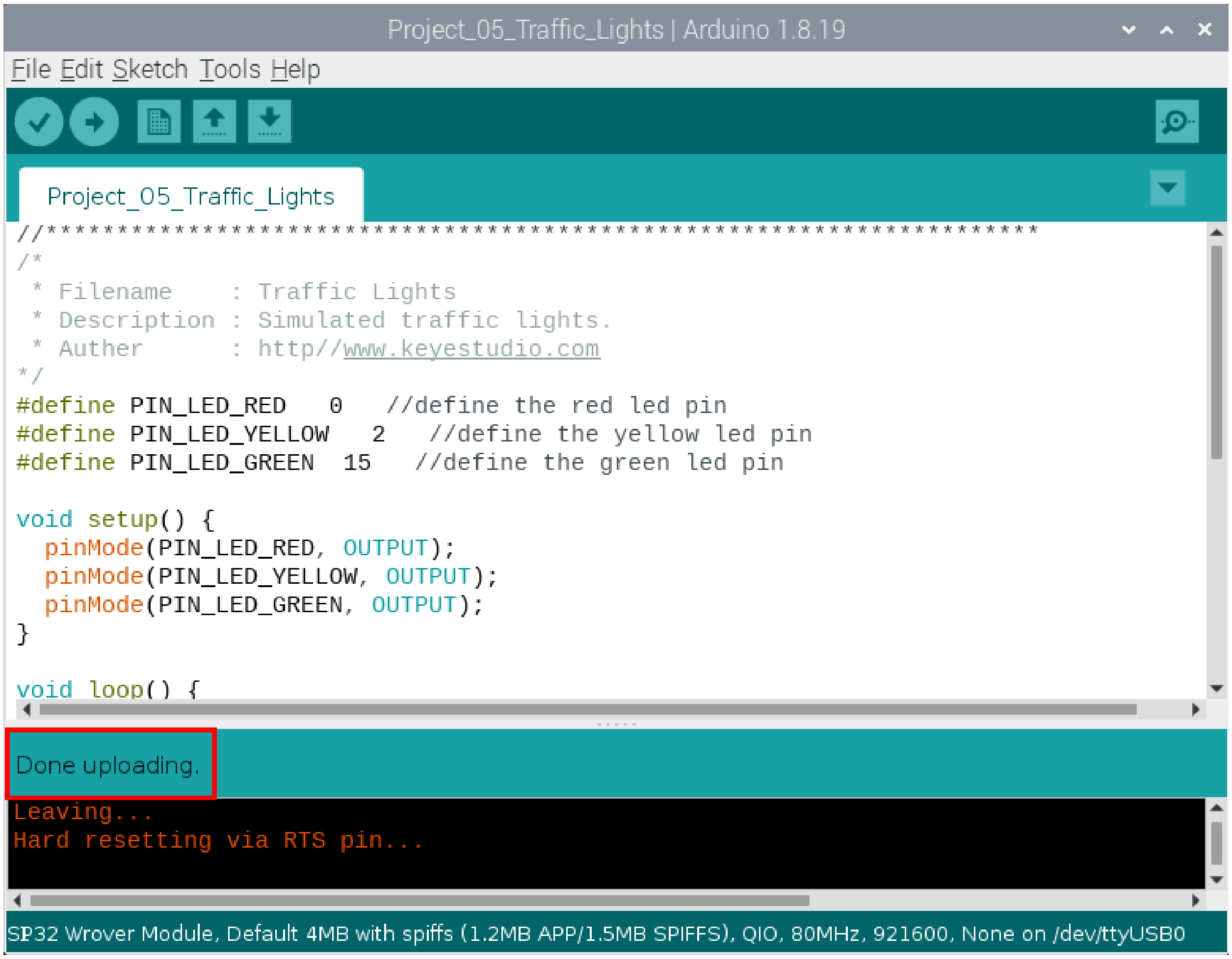


Note: If uploading the code fails, you can press the Boot button on ESP32 after click , and release the Boot button 

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.

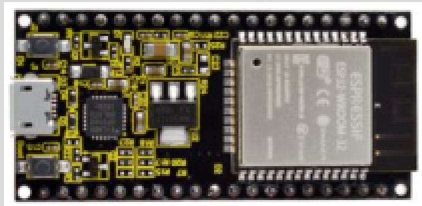
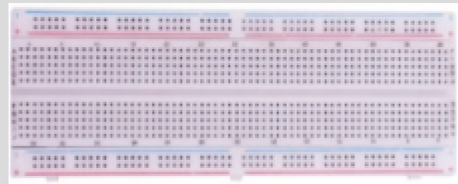
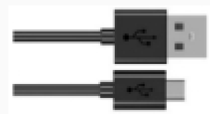
9.6 Project 06: RGB LED

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.

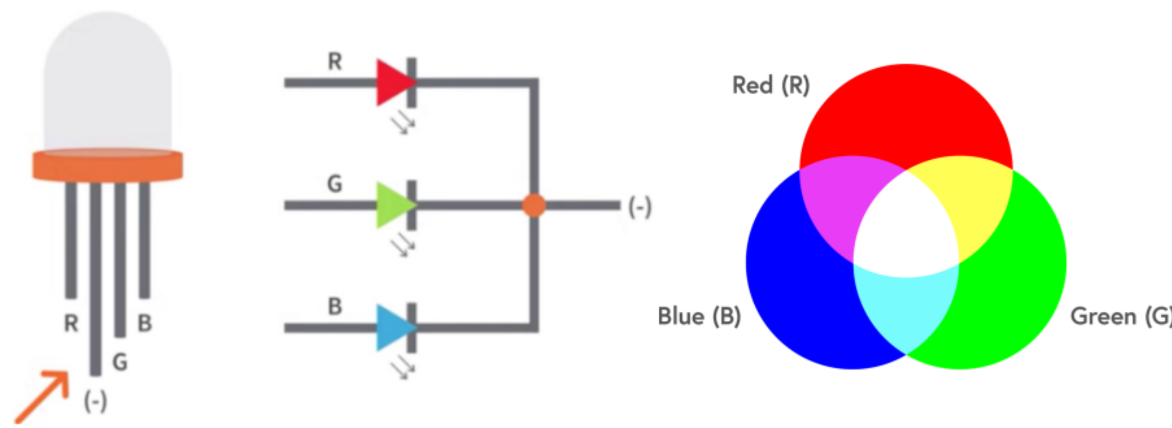
In this project, we will introduce the RGB and show you how to use ESP32 to control the RGB to emit different color light .RGB is pretty basic, but it's also a great way to learn the fundamentals of electronics and coding.

2.Components

	
ESP32*1	Breadboard*1
	
RGB*1	220 Resistor*1
	
jump wires	USB cable *1

3.Component knowledge

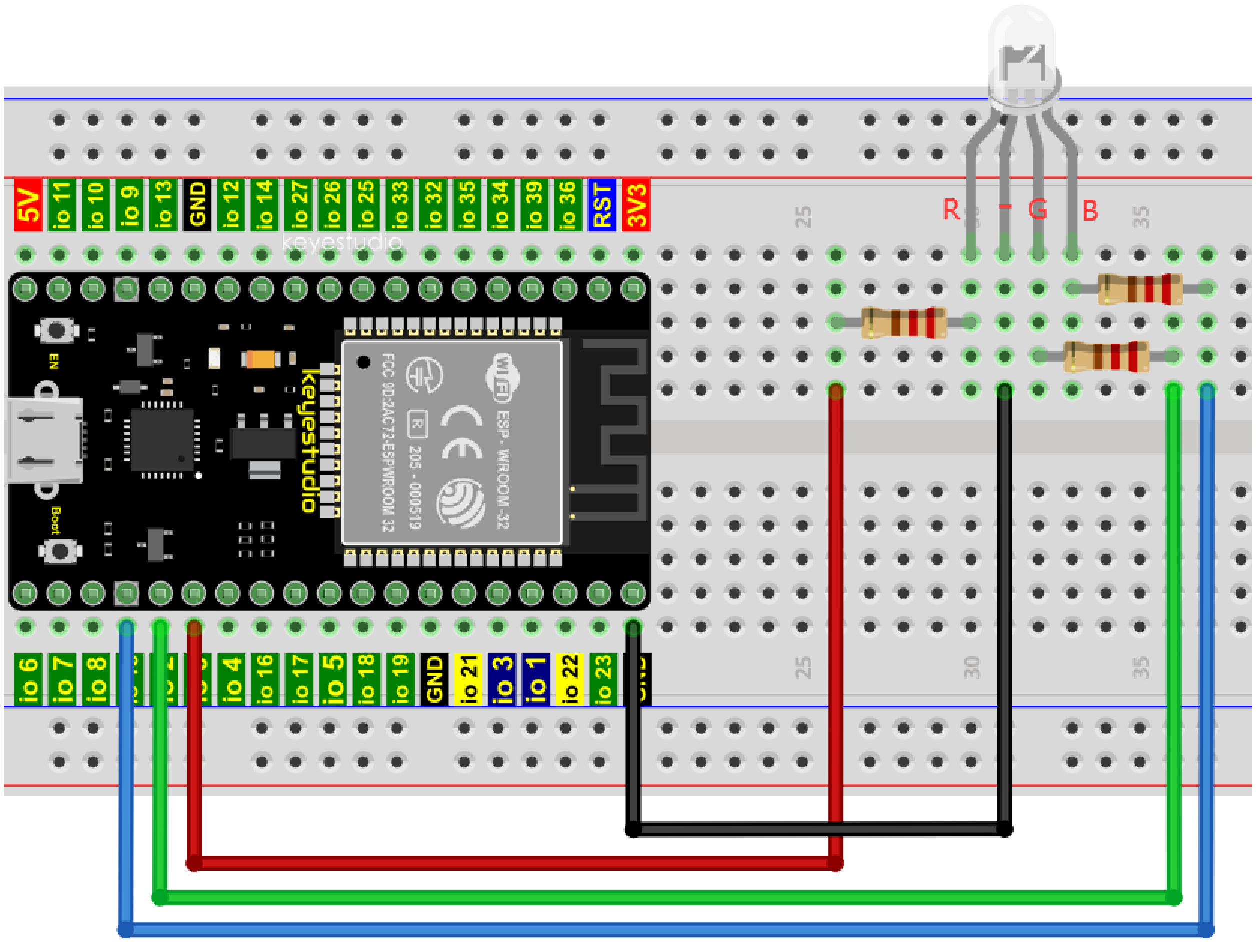
Most monitors adopt the RGB color standard, and all colors on a computer screen are a mixture of red, green and blue in varying proportions.



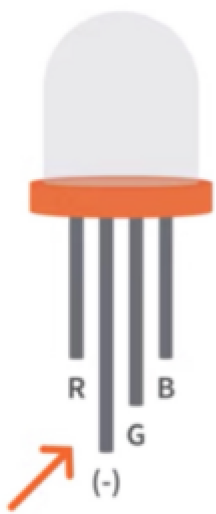
This RGB LED has 4 pins, each color (red, green, blue) and a common cathode,To change its brightness, we can use the PWM of the ESP32 pins, which can give different duty cycle signals to the RGB to produce different colors of light.

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

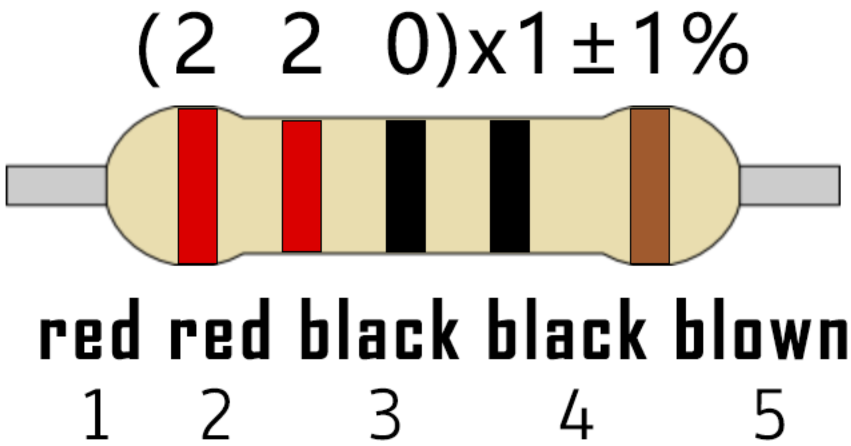
4.Wiring diagram



Notice: The longest pin (common cathode) of the RGB LED is connected to GND.



How to identify the 220 Five-color ring resistor



5.Project code

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

6.Project result

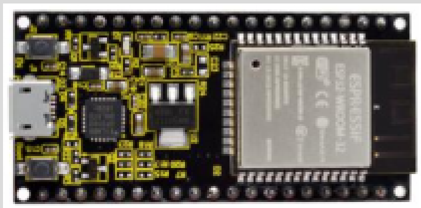
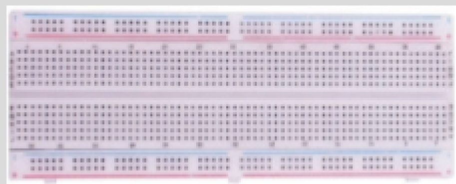
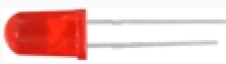
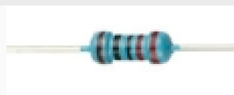
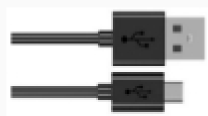
Compile and upload the code to ESP32, after the code is uploaded successfully, power up with a USB cable and you will see that the RGB LED starts to display random colors.

9.7 Project 07: Flowing Water Light

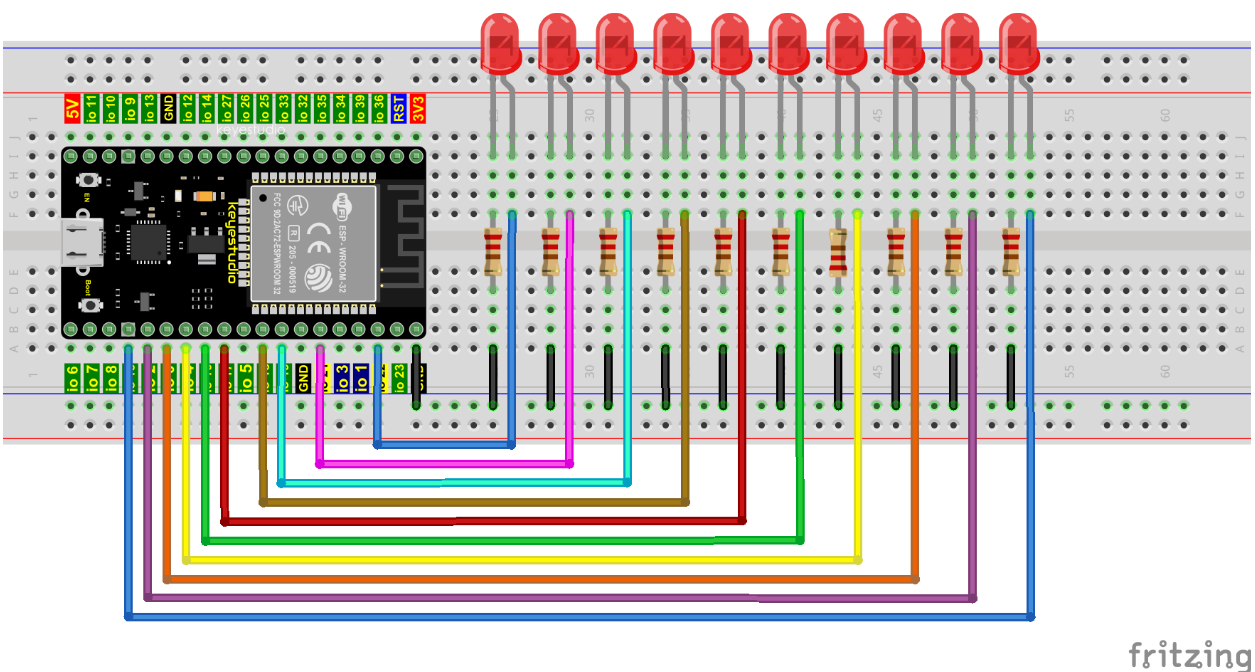
1.Introduction

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

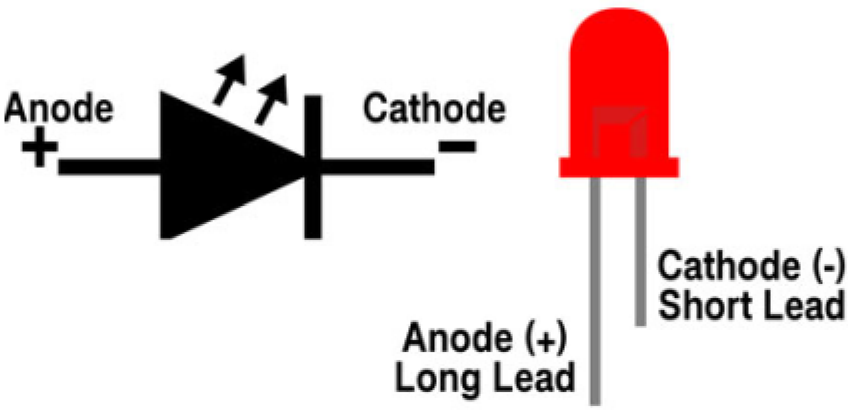
	
ESP32*1	Breadboard*1
	
Red LED*1	220 Resistor*1
	
Jumper Wires	USB Cable*1

3.Wiring diagram:

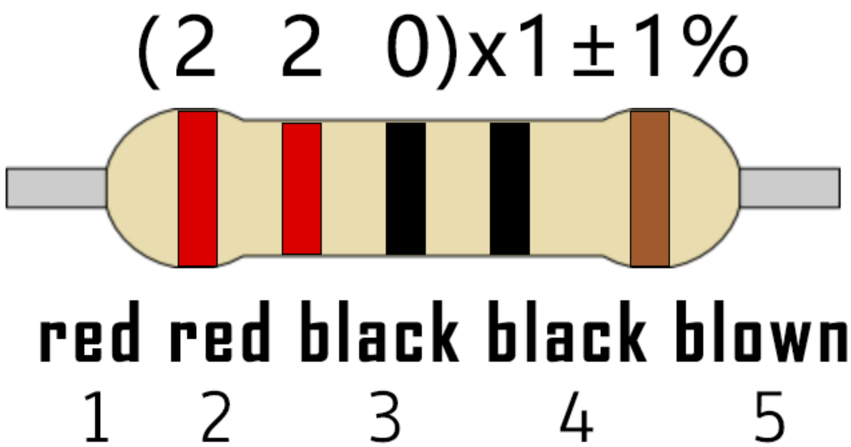


Note:

How to connect a LED



How to identify the 220 Five-color ring resistor



4. Test 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.

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

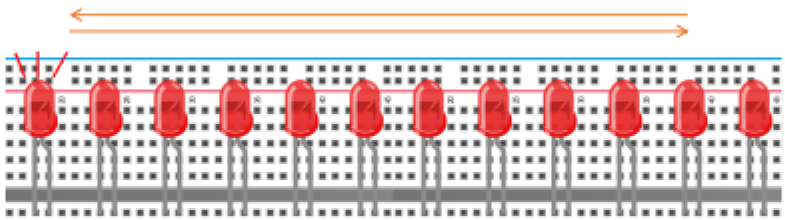
(continues on next page)

(continued from previous page)

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

4.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.

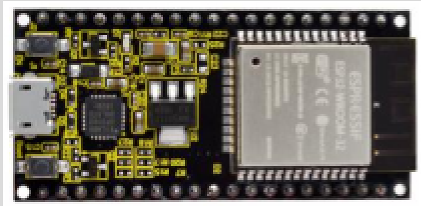
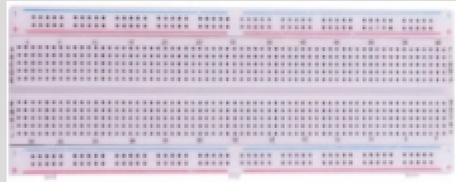
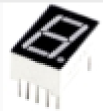
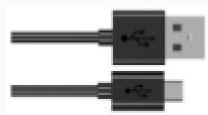


9.8 Project 081-Digit Digital Tube

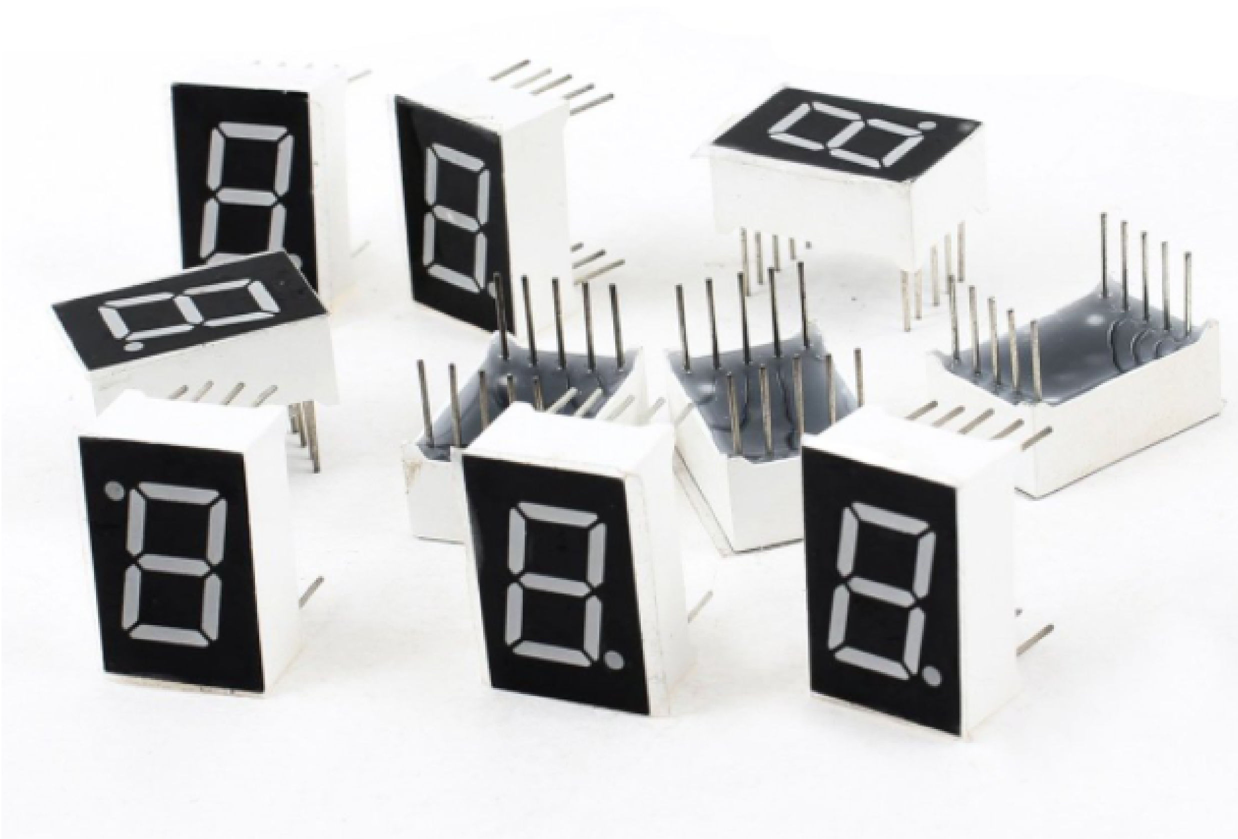
1.Introduction

The 1-Digit 7-Segment display is an device that displays decimal numbers, which is widely used in digital clocks, electronic meters, basic calculators and other electronic devices that display digital information. In this project, we will use ESP32 to control 1-Digit 7-segment display to display numbers.

2.Components

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

3.Component Knowledge

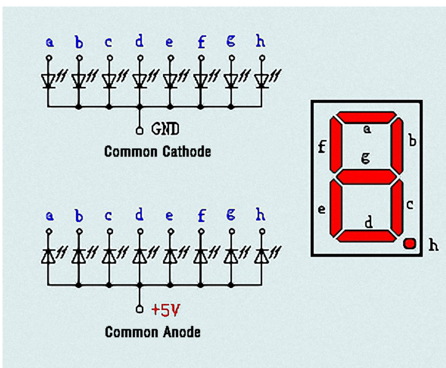


1-Digit 7-Segment Display: It is a semiconductor light emitting device, and its basic unit is a light-emitting diode (LED). The digital tube display can be divided into 7-segment display and 8-segment display according to the number of segments.

The 8-segment display has one more LED unit than the 7-segment display(used for decimal point display). Each segment of the 7-segment display is a separate LED. According to the connection mode of the LED unit, the digital tube can be divided into a common anode digital tube and a common cathode digital tube.

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

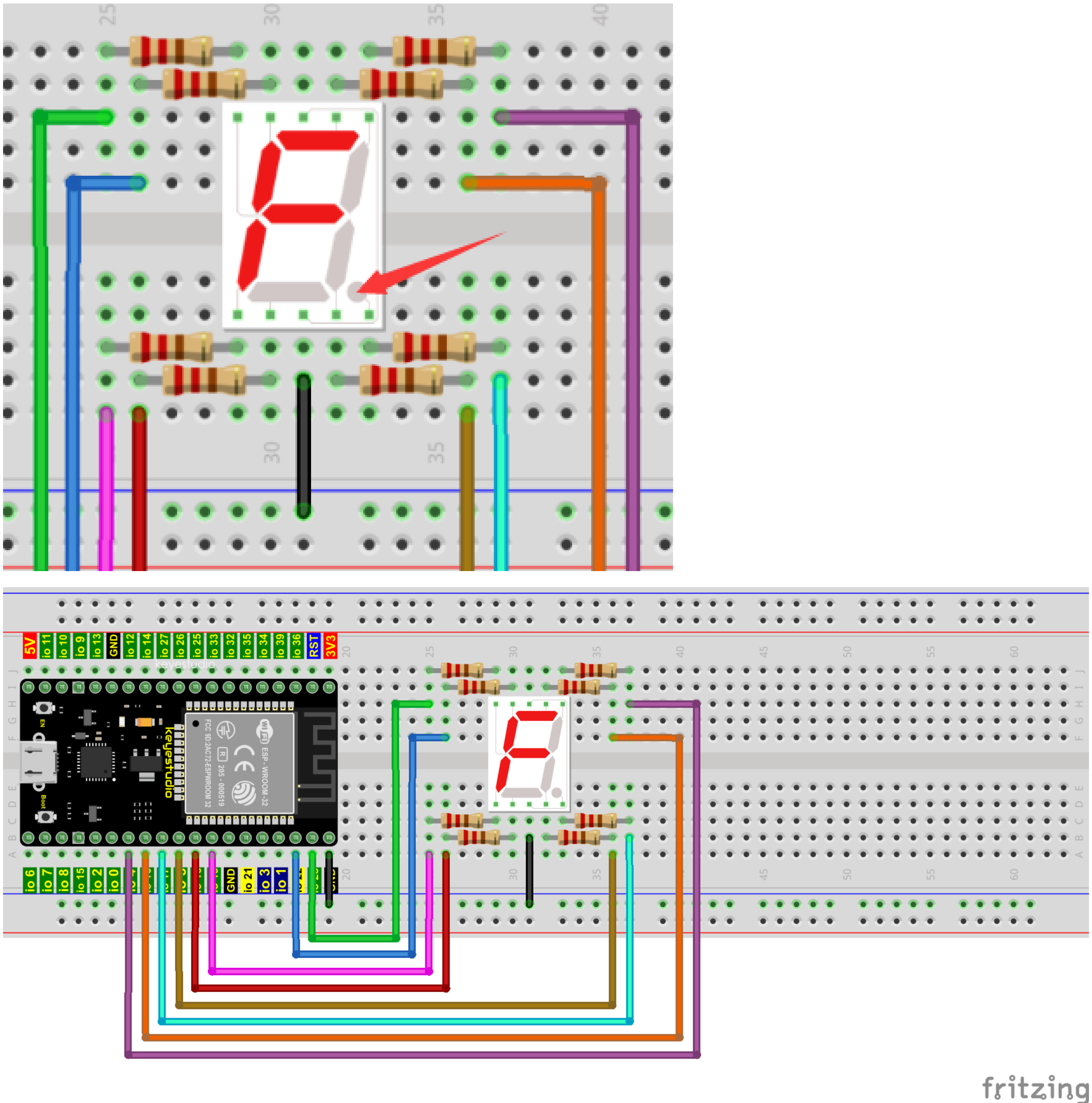
In the common anode 7-segment display, the LED anodes (positive pole) of all segments are connected together, so you should connect the common anode to"+5V". If you are about to light up a segmented LED, you can set its associated pin to"LOW".



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 will use an ordinary common cathode one-digit digital tube. As we mentioned above, you should connect the common cathode to GND. If you are about 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.Test 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.

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

(continues on next page)

(continued from previous page)

```
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);
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);
}
```

(continues on next page)

(continued from previous page)

```
void digital_4(void) // displays number 4
{
digitalWrite(a,LOW);
digitalWrite(b,HIGH);
digitalWrite(c,HIGH);
digitalWrite(d,LOW);
digitalWrite(e,LOW);
digitalWrite(f,HIGH);
digitalWrite(g,HIGH);
digitalWrite(dp,LOW);
}
void digital_5(void) // displays number 5
{
digitalWrite(a,HIGH);
digitalWrite(b,LOW);
digitalWrite(c,HIGH);
digitalWrite(d,HIGH);
digitalWrite(e,LOW);
digitalWrite(f,HIGH);
digitalWrite(g,HIGH);
digitalWrite(dp,LOW);
}
void digital_6(void) // displays number 6
{
digitalWrite(a,HIGH);
digitalWrite(b,LOW);
digitalWrite(c,HIGH);
digitalWrite(d,HIGH);
digitalWrite(e,HIGH);
digitalWrite(f,HIGH);
digitalWrite(g,HIGH);
digitalWrite(dp,LOW);
}
void digital_7(void) // displays number 7
{
digitalWrite(a,HIGH);
digitalWrite(b,HIGH);
digitalWrite(c,HIGH);
digitalWrite(d,LOW);
digitalWrite(e,LOW);
digitalWrite(f,LOW);
digitalWrite(g,LOW);
digitalWrite(dp,LOW);
}
void digital_8(void) // displays number 8
{
digitalWrite(a,HIGH);
digitalWrite(b,HIGH);
digitalWrite(c,HIGH);
digitalWrite(d,HIGH);
digitalWrite(e,HIGH);
digitalWrite(f,HIGH);
```

(continues on next page)

(continued from previous page)

```
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
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.Test Result

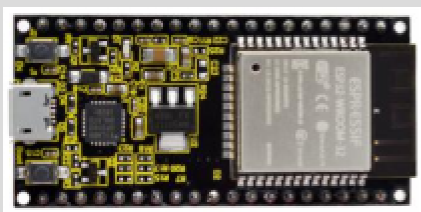
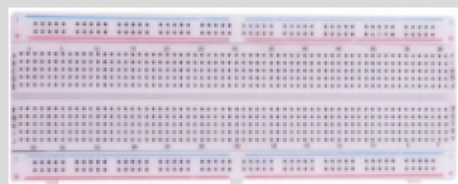
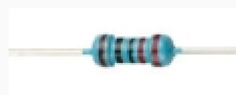
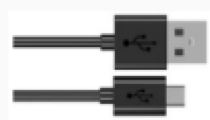
Compile and upload the code to ESP32, after the code is uploaded successfully, power up with a USB cable and you will see that the 1-Digit 7-Segment display will display numbers from 9 to 0.

9.9 Project 094-Digit Digital Tube

1.Introduction

The 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 the 4 Digit 7-segment display. In this project, we use ESP32 to control the 4-digit 7-segment display to display digits.

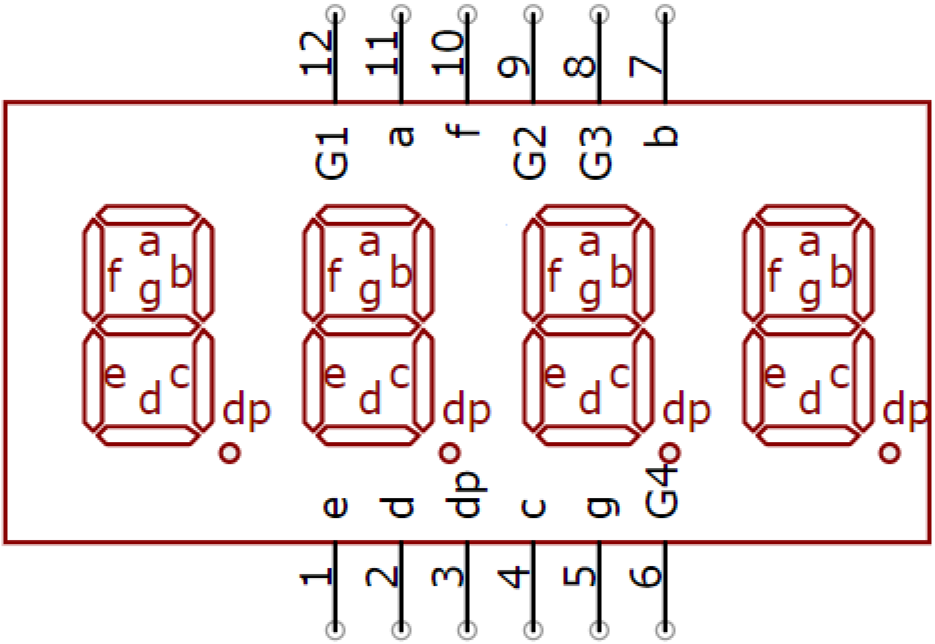
2.Components

	
ESP32*1	Breadboard*1
	
4-digit 7-segment display Module*1	220 Resistor*8
	
Jumper Wires	USB Cable*1

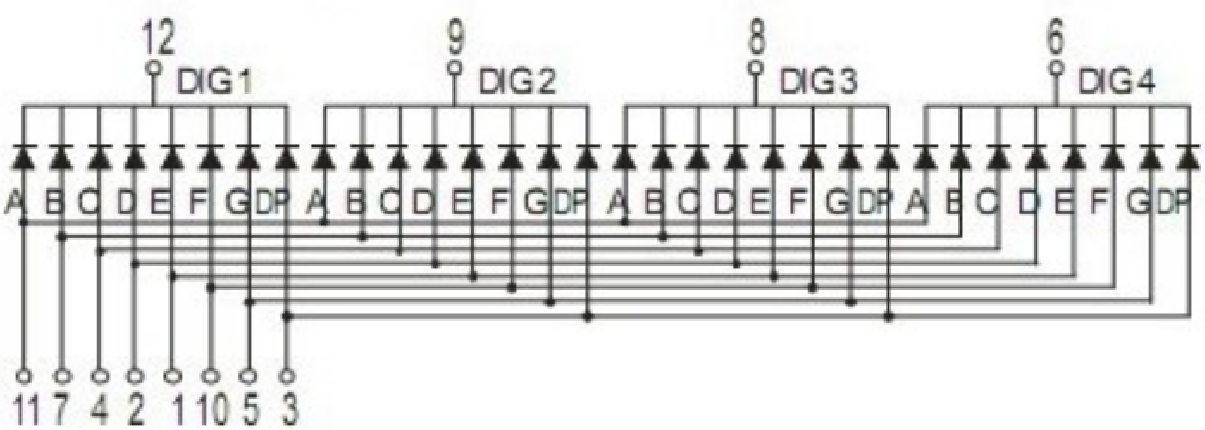
3.Component Knowledge

4-digit 7-segment display It is a device with common cathode and anode, its display principle is similar to the 1-Digit digital tube display. Both of them have eight GPIO ports to control the digital tube display, that is 8 leds. However, here is 4-digit, so it needs four GPIO ports to control the bit selection end. Our 4 - digit digital tube is common cathode.

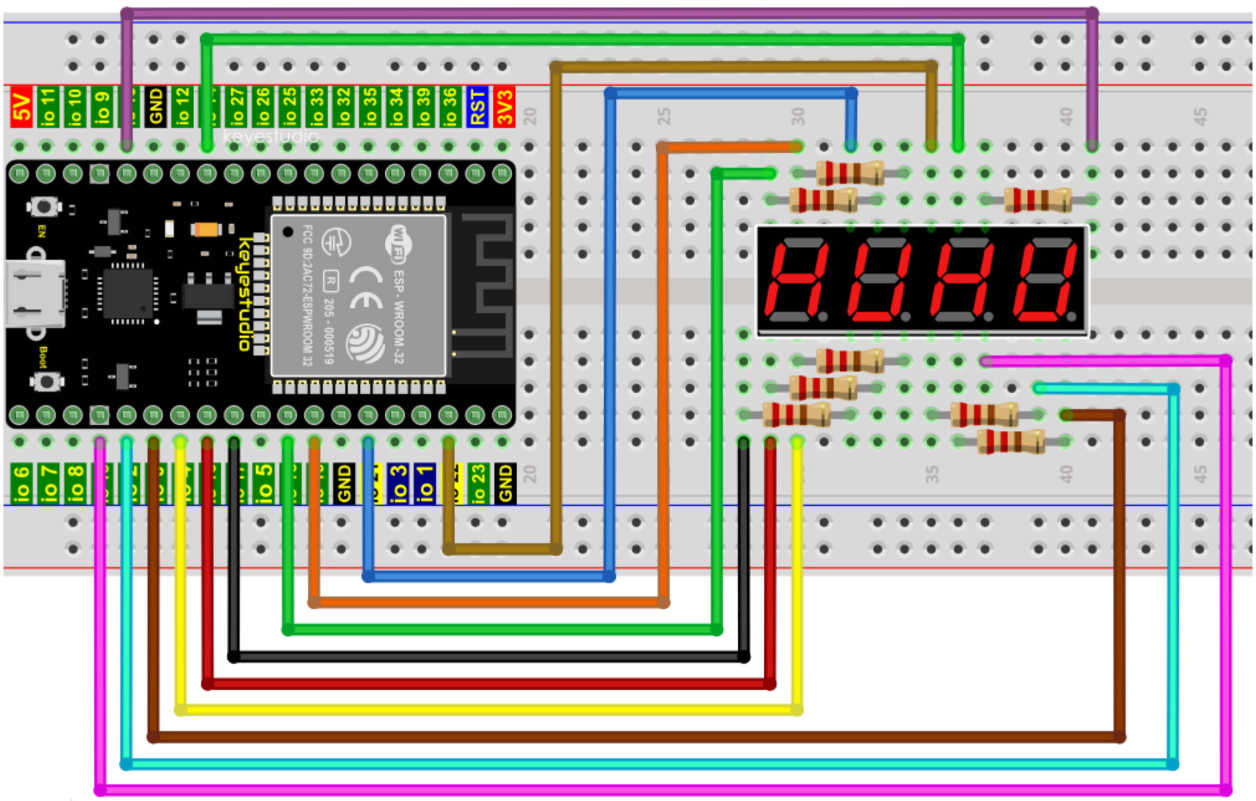
The following figure shows the pin diagram of the 4-digit digital tube. G1, G2, G3 and G4 are the control pins.



Schematic Diagram



4.Wiring Diagram



5.Test Code

```
//**********************************************************************
/*
 * Filename    : 4-Digit Digital Tube
 * Description : Four Digit Tube displays numbers from 0 to 9999.
 * Auther      : http//www.keyestudio.com
*/
#define d_a 18   //Define nixie tube a to pin 18
#define d_b 13
#define d_c 2
#define d_d 16
#define d_e 17
#define d_f 19
#define d_g 0
#define d_dp 4

#define G1 21   //Define the first set of nixtube G1 to pin 21
#define G2 22
#define G3 14
#define G4 15

//Nixie tube 0-F code value
unsigned char num[17][8] =
{
 //a  b  c  d  e  f  g  dp
  {1, 1, 1, 1, 1, 1, 0, 0},     //0
  {0, 1, 1, 0, 0, 0, 0, 0},     //1
```

(continues on next page)

(continued from previous page)

```
  {1, 1, 0, 1, 1, 0, 1, 0},     //2
  {1, 1, 1, 1, 0, 0, 1, 0},     //3
  {0, 1, 1, 0, 0, 1, 1, 0},     //4
  {1, 0, 1, 1, 0, 1, 1, 0},     //5
  {1, 0, 1, 1, 1, 1, 1, 0},     //6
  {1, 1, 1, 0, 0, 0, 0, 0},     //7
  {1, 1, 1, 1, 1, 1, 1, 0},     //8
  {1, 1, 1, 1, 0, 1, 1, 0},     //9
  {1, 1, 1, 0, 1, 1, 1, 1},     //A
  {1, 1, 1, 1, 1, 1, 1, 1},     //B
  {1, 0, 0, 1, 1, 1, 0, 1},     //C
  {1, 1, 1, 1, 1, 1, 0, 1},     //D
  {1, 0, 0, 1, 1, 1, 1, 1},     //E
  {1, 0, 0, 0, 1, 1, 1, 1},     //F
  {0, 0, 0, 0, 0, 0, 0, 1},     //.
};

void setup()
{
  pinMode(d_a,OUTPUT);    //Set to output pin
  pinMode(d_b,OUTPUT);
  pinMode(d_c,OUTPUT);
  pinMode(d_d,OUTPUT);
  pinMode(d_e,OUTPUT);
  pinMode(d_f,OUTPUT);
  pinMode(d_g,OUTPUT);
  pinMode(d_dp,OUTPUT);

  pinMode(G1,OUTPUT);
  pinMode(G2,OUTPUT);
  pinMode(G3,OUTPUT);
  pinMode(G4,OUTPUT);
}

void loop()
{

  //Start counting from 0 and gradually increase by 1 to 9999, repeating.
  for(int l = 0;l < 10;l++ )
  {
    for(int k = 0; k < 10;k++)
    {
      for(int j = 0; j < 10; j++)
      {
        for(int i = 0;i < 10;i++)
        {
          //125 flashes a second equals one second.
          //1000/8=125
          for(int q = 0;q<125;q++)
          {
            Display(1,l);//The first nixie tube shows the value of L.
            delay(2);
```

(continues on next page)

(continued from previous page)

```
            Display(2,k);
            delay(2);
            Display(3,j);
            delay(2);
            Display(4,i);
            delay(2);
          }

        }
      }
    }
  }
}
//Display functions: g ranges from 1 to 4,num ranges from 0 to 9
void Display(unsigned char g,unsigned char n)
{
  digitalWrite(d_a,LOW);      //Remove the light
  digitalWrite(d_b,LOW);
  digitalWrite(d_c,LOW);
  digitalWrite(d_d,LOW);
  digitalWrite(d_e,LOW);
  digitalWrite(d_f,LOW);
  digitalWrite(d_g,LOW);
  digitalWrite(d_dp,LOW);

  switch(g)           //Gate a choice
  {
    case 1:
      digitalWrite(G1,LOW);   //Choose the first digit
      digitalWrite(G2,HIGH);
      digitalWrite(G3,HIGH);
      digitalWrite(G4,HIGH);
      break;
    case 2:
      digitalWrite(G1,HIGH);
      digitalWrite(G2,LOW);   //Choose the second bit
      digitalWrite(G3,HIGH);
      digitalWrite(G4,HIGH);
      break;
    case 3:
      digitalWrite(G1,HIGH);
      digitalWrite(G2,HIGH);
      digitalWrite(G3,LOW);   //Choose the third bit
      digitalWrite(G4,HIGH);
      break;
    case 4:
      digitalWrite(G1,HIGH);
      digitalWrite(G2,HIGH);
      digitalWrite(G3,HIGH);
      digitalWrite(G4,LOW);   //Choose the fourth bit
      break;
    default:break;
```

(continues on next page)

(continued from previous page)

```
  }

  digitalWrite(d_a,num[n][0]);      //a Queries the code value table
  digitalWrite(d_b,num[n][1]);
  digitalWrite(d_c,num[n][2]);
  digitalWrite(d_d,num[n][3]);
  digitalWrite(d_e,num[n][4]);
  digitalWrite(d_f,num[n][5]);
  digitalWrite(d_g,num[n][6]);
  digitalWrite(d_dp,num[n][7]);
}
//**************************************************************************
```

6.Test Result

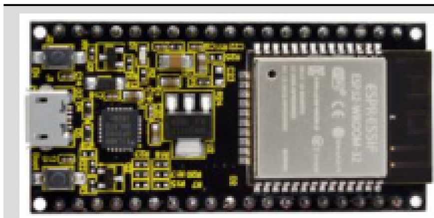
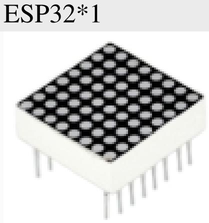
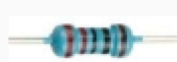
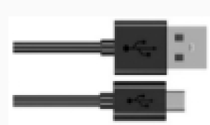
Compile and upload the code to ESP32, after the code is uploaded successfully, power up with a USB cable and you will see that the 4-digit 7-segment display displays 0-9999and repeat these actions in an infinite loop.

9.10 Project 108×8 Dot-matrix Display

1.Introduction

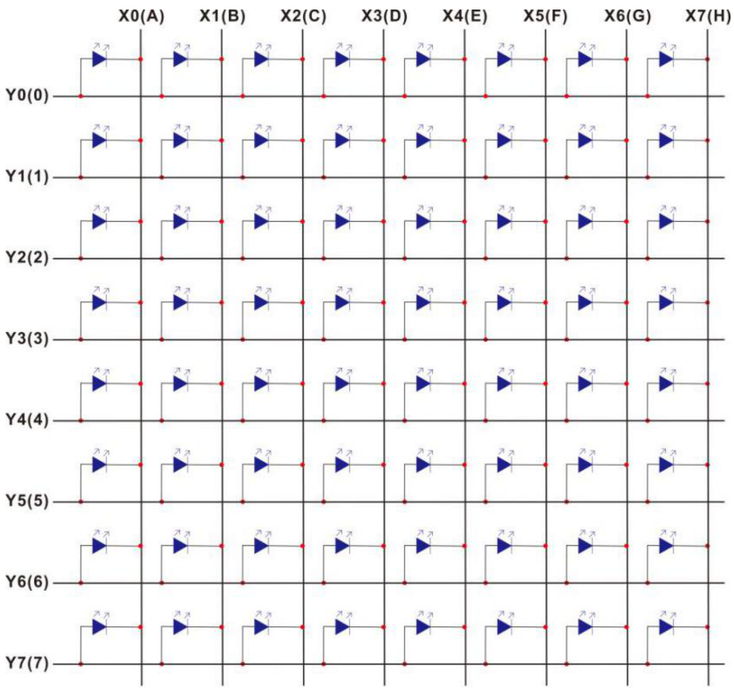
Dot matrix display is an electronic digital display device that can display information on machine, clocks, public transport departure indicators and many other devices. In this project, we will use ESP32 to control 8x8 LED dot matrix to display digits.

2.Components

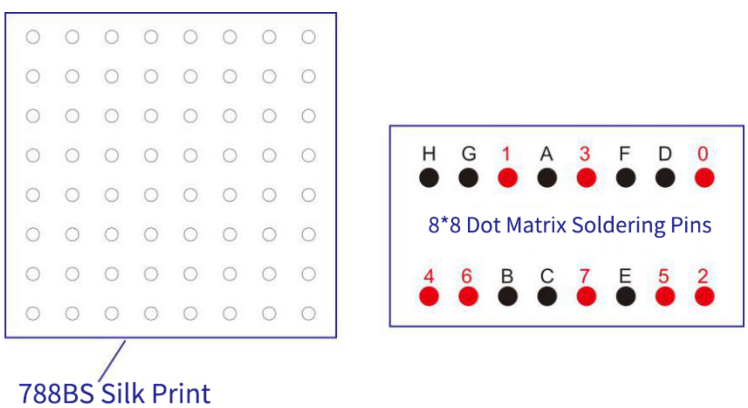
	
ESP32*1	Breadboard*1
	
8*8 dot matrix module *1	220 Resistor*8
	
Jumper Wires	USB Cable*1

3.Component Knowledge

8*8 dot matrix moduleThe 8*8 dot matrix is composed of 64 LEDs, including row common anode and row common cathode. Our module is row common anode, each row has a line connecting the positive pole of the LED, and the column is connecting the negative pole of the LED lamp, as shown in the following figure :

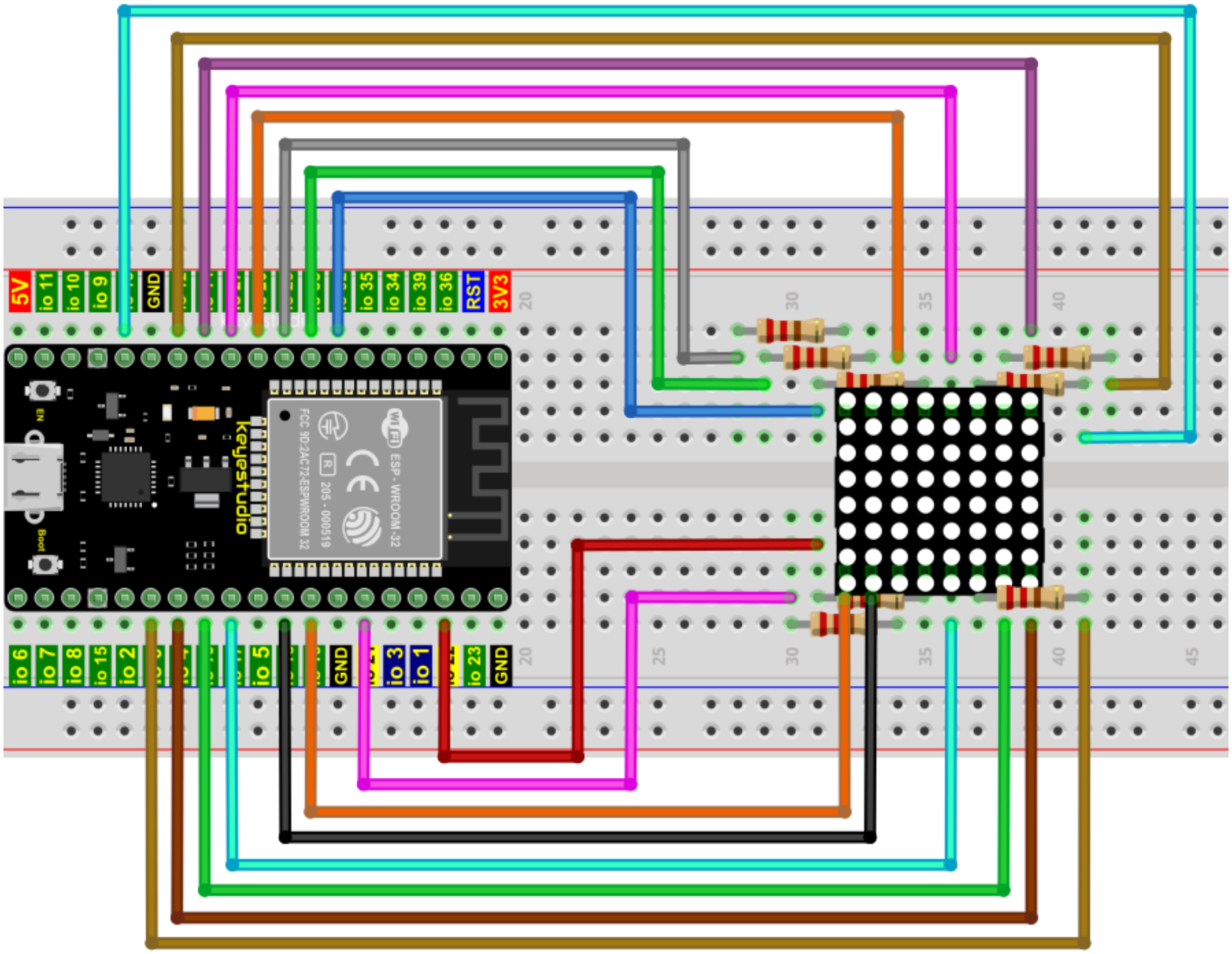


8*8 Dot Matrix LED Equivalent Circuit



8*8 Dot Matrix Outlook and Pinouts

4.Wiring Diagram



5.Test Code

```
//**********************************************************************************
/*
 * Filename    : 8×8 Dot-matrix Display
 * Description : 8×8 Dot-matrix displays numbers from 0 to 9.
 * Auther      : http//www.keyestudio.com
*/
int R[] = {14,26,4,27,19,16,18,17};
int C[] = {32,21,22,12,0,13,33,25};

unsigned char data_0[8][8] =
{
{0,0,1,1,1,0,0,0},
{0,1,0,0,0,1,0,0},
{0,1,0,0,0,1,0,0},
{0,1,0,0,0,1,0,0},
{0,1,0,0,0,1,0,0},
{0,1,0,0,0,1,0,0},
{0,1,0,0,0,1,0,0},
{0,0,1,1,1,0,0,0}
};
```

(continues on next page)

(continued from previous page)

```
unsigned char data_1[8][8] =
{
{0,0,0,0,1,0,0,0},
{0,0,0,1,1,0,0,0},
{0,0,0,0,1,0,0,0},
{0,0,0,0,1,0,0,0},
{0,0,0,0,1,0,0,0},
{0,0,0,0,1,0,0,0},
{0,0,0,0,1,0,0,0},
{0,0,0,1,1,1,0,0}
};

unsigned char data_2[8][8] =
{
{0,0,1,1,1,0,0,0},
{0,1,0,0,0,1,0,0},
{0,0,0,0,0,1,0,0},
{0,0,0,0,1,0,0,0},
{0,0,0,1,0,0,0,0},
{0,0,1,0,0,0,0,0},
{0,1,1,1,1,1,0,0},
{0,0,0,0,0,0,0,0}
};

unsigned char data_3[8][8] =
{
{0,0,1,1,1,1,0,0},
{0,0,0,0,0,1,0,0},
{0,0,0,0,0,1,0,0},
{0,0,1,1,1,1,0,0},
{0,0,0,0,0,1,0,0},
{0,0,0,0,0,1,0,0},
{0,0,1,1,1,1,0,0},
{0,0,0,0,0,0,0,0}
};

unsigned char data_4[8][8] =
{
{0,1,0,0,0,0,0,0},
{0,1,0,0,1,0,0,0},
{0,1,0,0,1,0,0,0},
{0,1,1,1,1,1,1,0},
{0,0,0,0,1,0,0,0},
{0,0,0,0,1,0,0,0},
{0,0,0,0,1,0,0,0},
{0,0,0,0,0,0,0,0}
};

unsigned char data_5[8][8] =
{
{0,1,0,0,0,0,0,0},
```

(continues on next page)

(continued from previous page)

```
{0,1,1,1,1,1,0,0},
{0,1,0,0,0,0,0,0},
{0,1,1,1,1,1,0,0},
{0,0,0,0,0,1,0,0},
{0,0,0,0,0,1,0,0},
{0,1,1,1,1,1,0,0},
{0,0,0,0,0,0,0,0}
};

unsigned char data_6[8][8] =
{
{0,1,1,1,1,1,0,0},
{0,1,0,0,0,0,0,0},
{0,1,0,0,0,0,0,0},
{0,1,1,1,1,1,0,0},
{0,1,0,0,0,1,0,0},
{0,1,0,0,0,1,0,0},
{0,1,1,1,1,1,0,0},
{0,0,0,0,0,0,0,0}
};

unsigned char data_7[8][8] =
{
{0,0,0,0,0,0,0,0},
{0,1,1,1,1,1,0,0},
{0,0,0,0,0,1,0,0},
{0,0,0,0,1,0,0,0},
{0,0,0,1,0,0,0,0},
{0,0,1,0,0,0,0,0},
{0,1,0,0,0,0,0,0},
{0,0,0,0,0,0,0,0}
};

unsigned char data_8[8][8] =
{
{0,1,1,1,1,1,0,0},
{0,1,0,0,0,1,0,0},
{0,1,0,0,0,1,0,0},
{0,1,1,1,1,1,0,0},
{0,1,0,0,0,1,0,0},
{0,1,0,0,0,1,0,0},
{0,1,1,1,1,1,0,0},
{0,0,0,0,0,0,0,0}
};

unsigned char data_9[8][8] =
{
{0,1,1,1,1,1,0,0},
{0,1,0,0,0,1,0,0},
{0,1,0,0,0,1,0,0},
{0,1,1,1,1,1,0,0},
{0,0,0,0,0,1,0,0},
```

(continues on next page)

(continued from previous page)

```
{0,0,0,0,0,1,0,0},
{0,1,1,1,1,1,0,0},
{0,0,0,0,0,0,0,0}
};

void Display(unsigned char dat[8][8])
{
for(int c = 0; c<8;c++)
{
digitalWrite(C[c],LOW);
for(int r = 0;r<8;r++)
{
digitalWrite(R[r],dat[r][c]);
}
delay(1);
Clear();
}
}

void Clear()
{
for(int i = 0;i<8;i++)
{
digitalWrite(R[i],LOW);
digitalWrite(C[i],HIGH);
}
}

void setup(){
  for(int i = 0;i<8;i++)
  {
    pinMode(R[i],OUTPUT);
    pinMode(C[i],OUTPUT);
  }

}

void loop(){
  for (int i = 1; i <= 100; i = i + (1)) {
    Display(data_0);
  }
  for (int i = 1; i <= 100; i = i + (1)) {
    Display(data_1);
  }
  for (int i = 1; i <= 100; i = i + (1)) {
    Display(data_2);
  }
  for (int i = 1; i <= 100; i = i + (1)) {
    Display(data_3);
  }
  for (int i = 1; i <= 100; i = i + (1)) {
    Display(data_4);
```

(continues on next page)

(continued from previous page)

```
  }
  for (int i = 1; i <= 100; i = i + (1)) {
    Display(data_5);
  }
  for (int i = 1; i <= 100; i = i + (1)) {
    Display(data_6);
  }
  for (int i = 1; i <= 100; i = i + (1)) {
    Display(data_7);
  }
  for (int i = 1; i <= 100; i = i + (1)) {
    Display(data_8);
  }
  for (int i = 1; i <= 100; i = i + (1)) {
    Display(data_9);
  }
}
//*************************************************************************************
```

6.Test 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 displays the numbers 0~9 respectively.

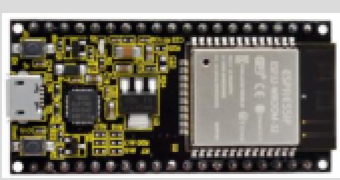
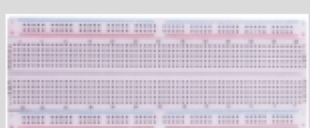
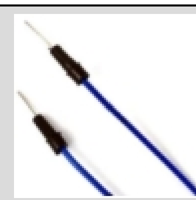
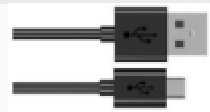
9.11 Project 1174HC595N Control 8 LEDs

1.Introduction

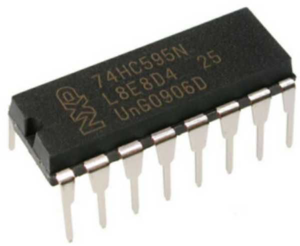
In previous projects, we learned how to light up an LED. With only 32 IO ports on ESP32, how do we light up a lot of leds? Sometimes it is possible to run out of pins on the ESP32, and you need to extend it with the shift register. You can use the 74HC595N chip to control 8 outputs at a time, taking 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 a ESP32a 74HC595 chip and LEDs to make a flowing water light to understand the function of the 74HC595 chip.

2.Components

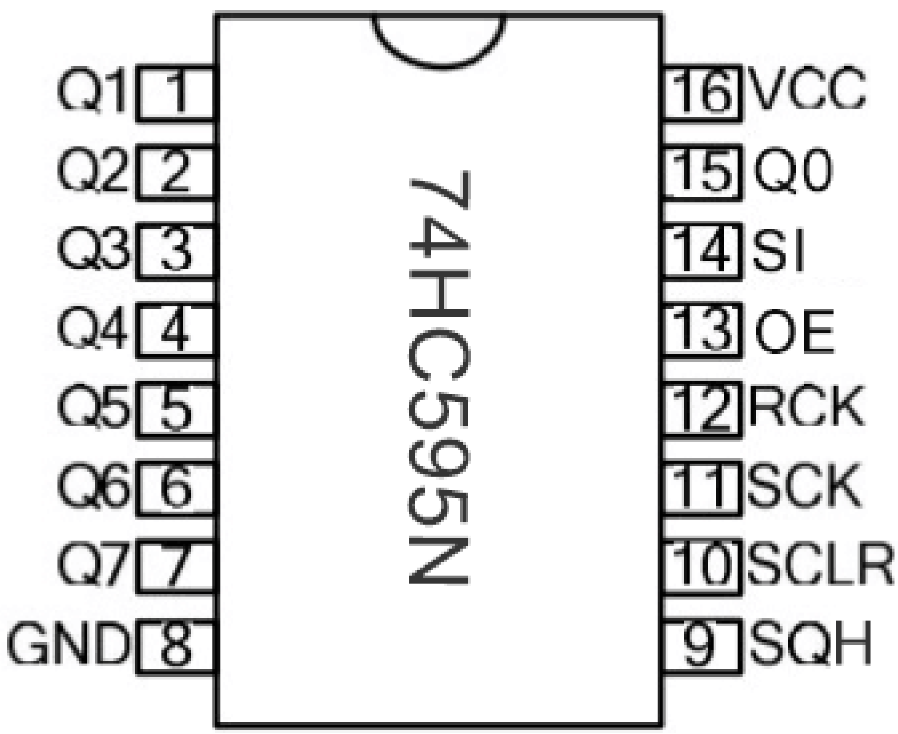
			
ESP32*1	Breadboard*1	74HC595N chip*1	Jumper Wires
			
220 Resistor*8	Red LED*8	USB Cable*1	

3.Component Knowledge



74HC595N Chip: To put it simply, 74HC595N chip is a combination of 8-digit shifting register, memorizer and equipped with tri-state output. The shift register and the memorizer are synchronized to different clocks, and the data is input on the rising edge of the shift register clock SCK and goes into the memory register on the rising edge of the memory register clock RCK.

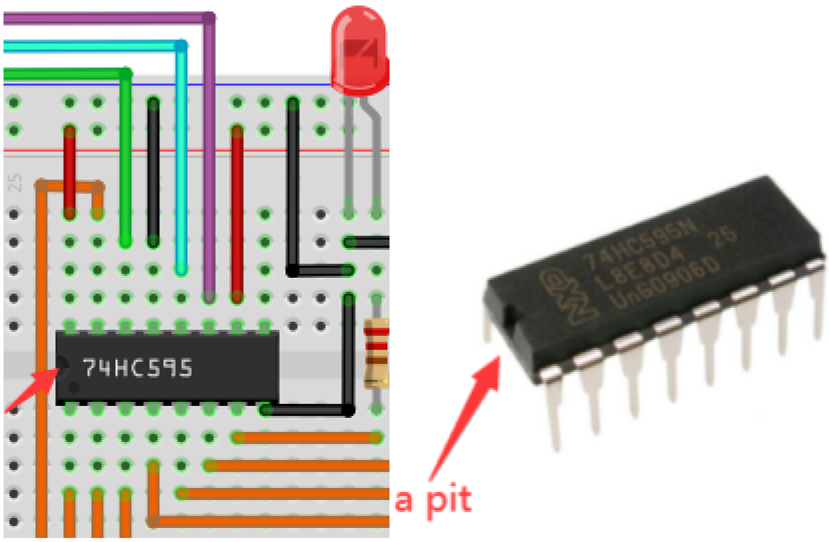
If the two clocks are connected together, the shift register is always one pulse earlier than the storage register. The shift register has a serial shift input (SI) and a serial output (SQH) for cascading. The 8-bit shift register can be reset asynchronously (low-level reset), and the storage register has an 8-bit Three-state parallel bus output, when the output enable (OE) is enabled (active low), the storage register is output to the 74HC595N pin (bus).

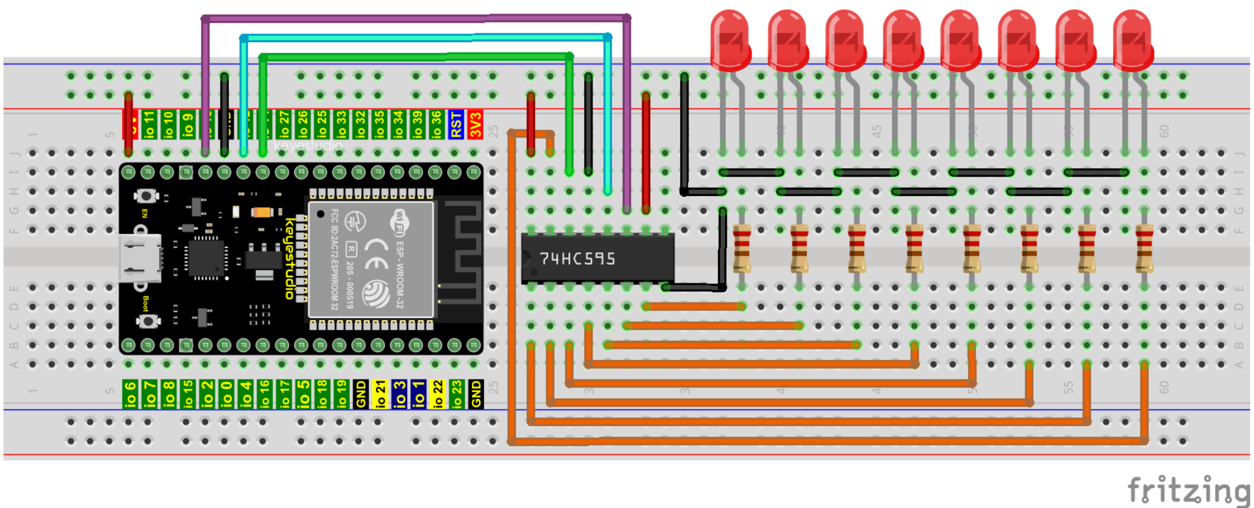


Pins

4.Wiring Diagram

Note: Pay attention to the direction in which the 74HC595N chip is inserted.





5.Test Code

```
//**********************************************************************
/*
 * 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.
```

(continues on next page)

(continued from previous page)

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

6.Test Result

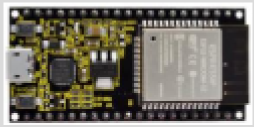
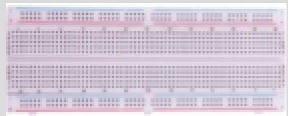
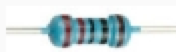
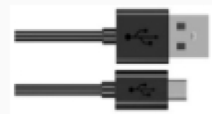
Compile and upload the code to ESP32, after the code is uploaded successfully, power up with a USB cable and you will see that the 8 LEDs start flashing in flowing water mode.

9.12 Project 12Active Buzzer

1.Introduction

Active buzzer is a sound component that is widely used as a sound component for computers, printers, alarms, electronic toys and phones, timers 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

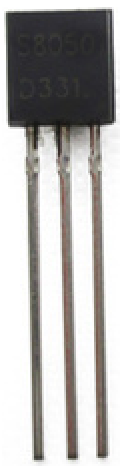
			
ESP32*1	Breadboard*1	Active buzzer*1	
			
NPN Transistor(S8050)*1	1k Resistor*1	Jumper Wires	USB Cable*1

3.Component Knowledge

The 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 sound. The passive buzzer is an integrated electronic buzzer with no internal vibration source. It must be driven by 2K to 5K square wave instead of a DC signal.

The appearance of the two buzzers is very similar, but passive buzzers come with a green circuit board, and active buzzers come with a black tape. Passive buzzers don't have positive pole, but active buzzers have. As shown below:

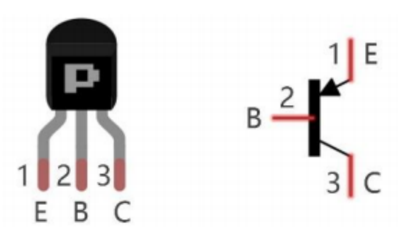
Transistor:

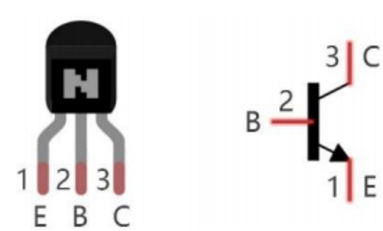


Since the buzzer requires such large current that GPIO of ESP32 output capability cannot meet the requirement, a transistor of NPN type is needed here to amplify the current.

Transistor, the full name: semiconductor transistor, is a semiconductor device that controls current. Transistor can be used to amplify weak signal, or work as a switch. It has three electrodes(PINs): base (b), collector © and emitter (e).

When there is current passing between "be", "ce" , which 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" , which 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,





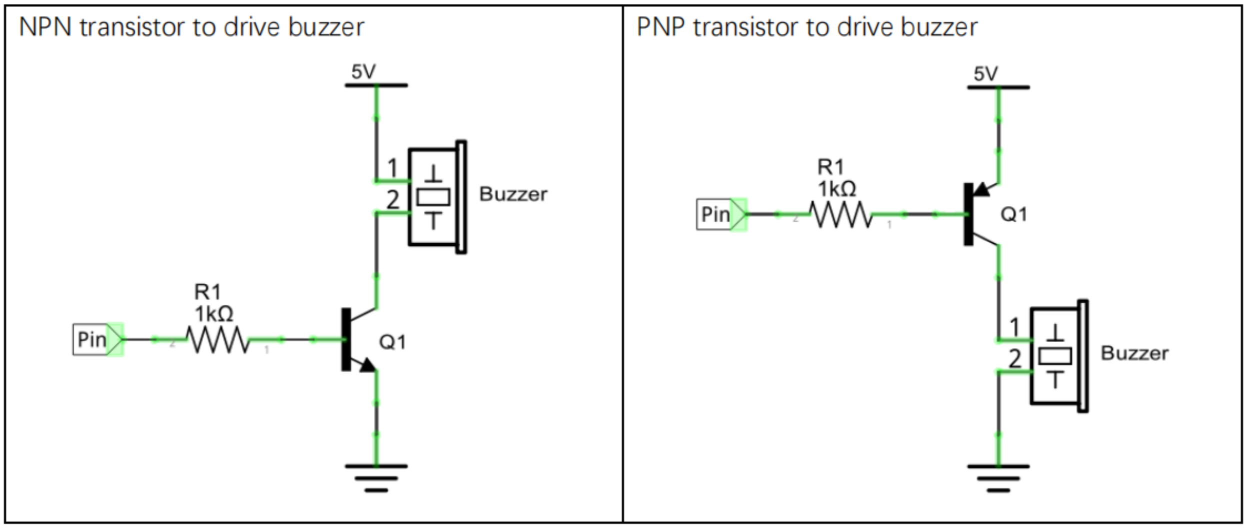
PNP transistor NPN transistor

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

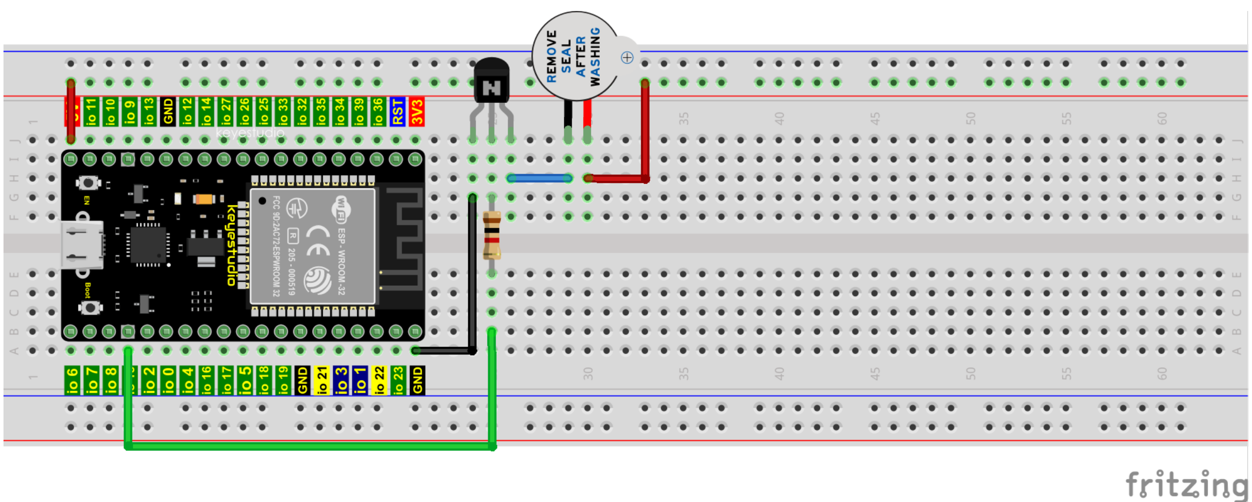
Based on the transistor's characteristics, it is often used as a switch in digital circuits. As micro-controller's capacity to output current is very weak, we will use a transistor to amplify current and drive large-current components.

When using the NPN transistor to drive a 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 the PNP transistor to drive a 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 it will reduce the loudness.

5.Test Code

```
//**********************************************************************
/*
 * Filename    : Active Buzzer
 * Description : Active buzzer beeps.
```

(continues on next page)

(continued from previous page)

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

6.Test Result

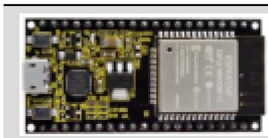
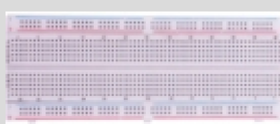
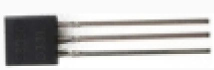
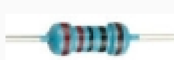
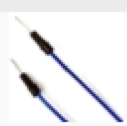
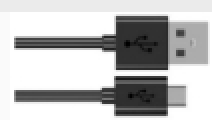
Compile and upload the code to ESP32, after the code is uploaded successfully, power up with a USB cable and you will see that the active buzzer beeps.

9.13 Project 13Passive Buzzer

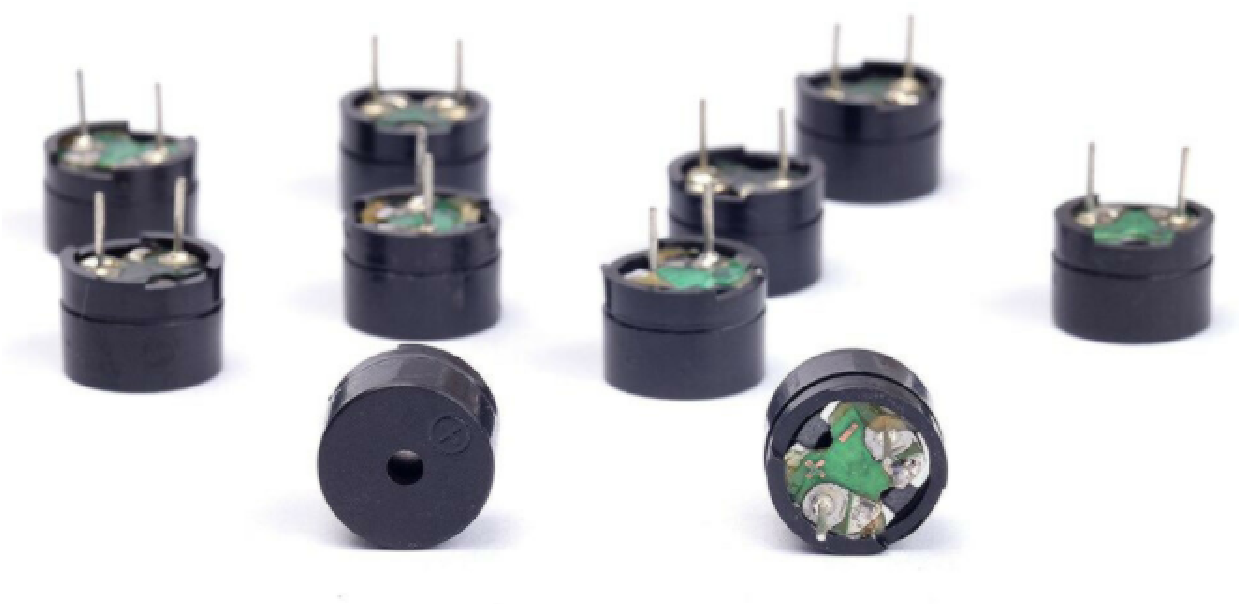
1.Introduction

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

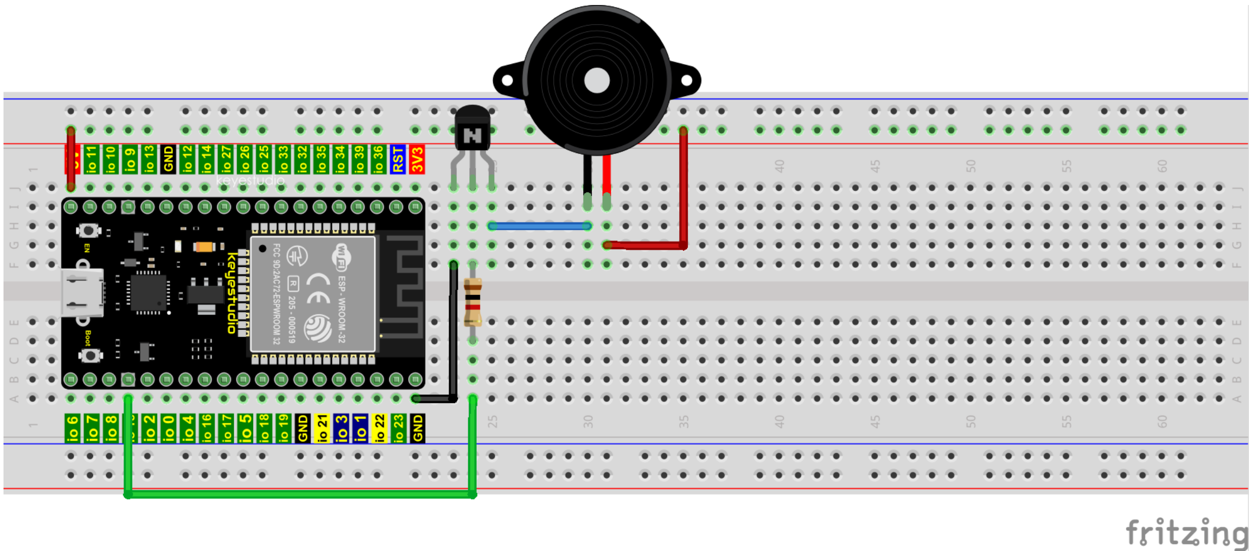
3.Component Knowledge



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.Test Code

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

6.Test Result

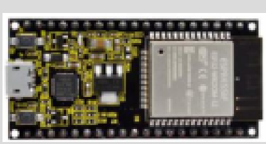
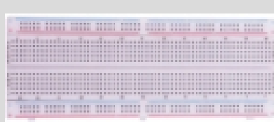
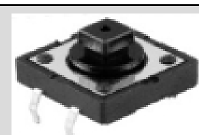
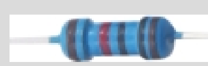
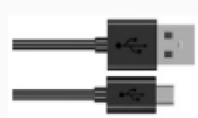
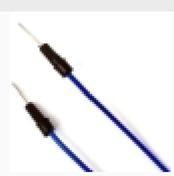
Compile and upload the code to ESP32, after the code is uploaded successfully, power up with a USB cable and you will see that the passive buzzer plays music.

9.14 Project 14: Mini Table Lamp

1. Introduction

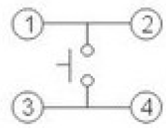
Do you know that the ESP32 can light up an LED when you press a button? In this project, we will use a ESP32, a button switch and a LED to make a mini table lamp.

2. Components

				
ESP32*1	Breadboard*1	Button*1	10K Resistor*1	
				
Red LED*1	22 Resistor*1	USB Cable*1	Jumper Wires	Button Cap*1

3. Component Knowledge

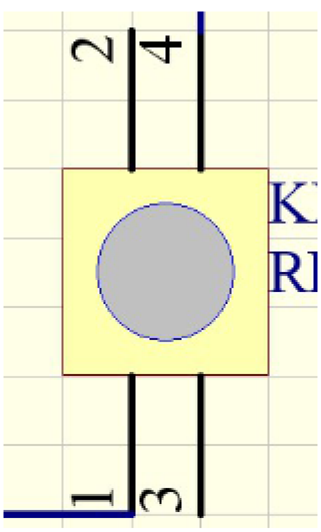
Button: A button can control the circuit on and off, the button is plugged into a circuit, the circuit is disconnected when the button is not pressed. The circuit works when you press the button, but breaks again when you release it. Why does it only work when you press it? It starts from the internal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

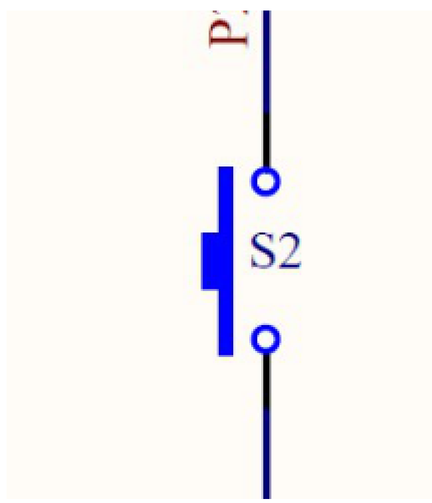
The internal structure of the button is shown in the figure 

. 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:

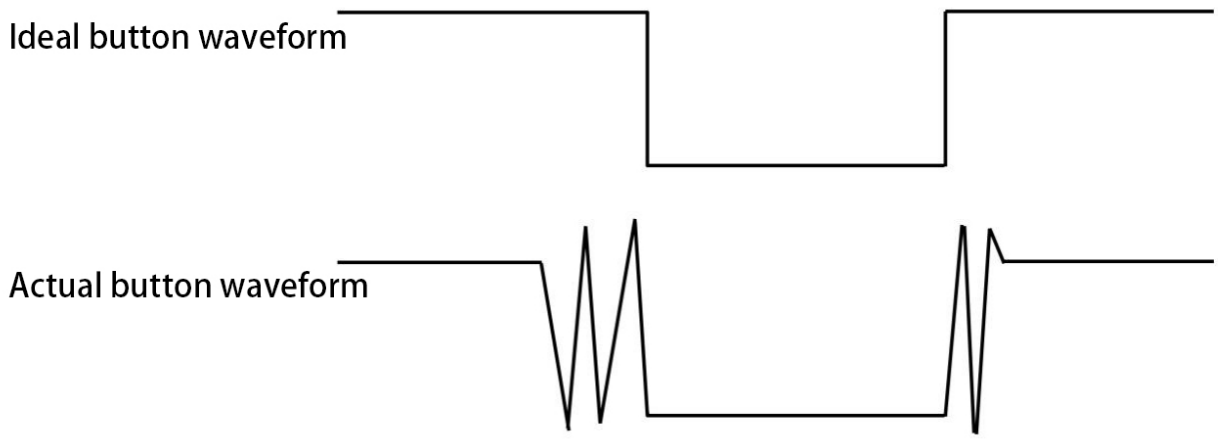




What is button shake?

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.

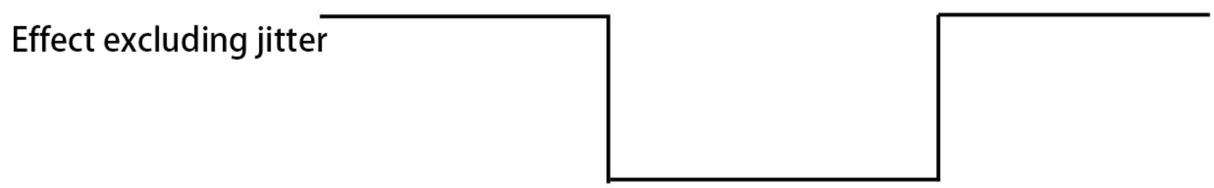


How to eliminate the shake?

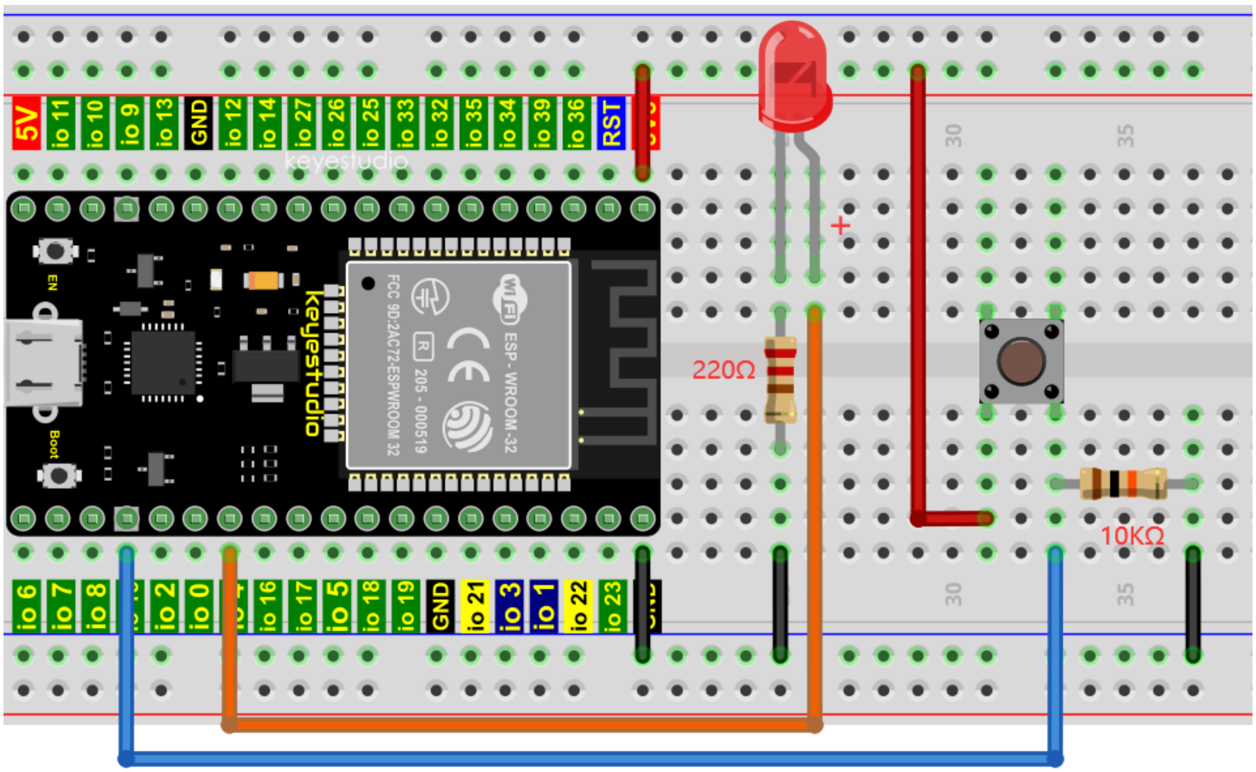
There are two common methods, namely fix shake in the software and hardware. We only discuss the shake removal in the software.

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.

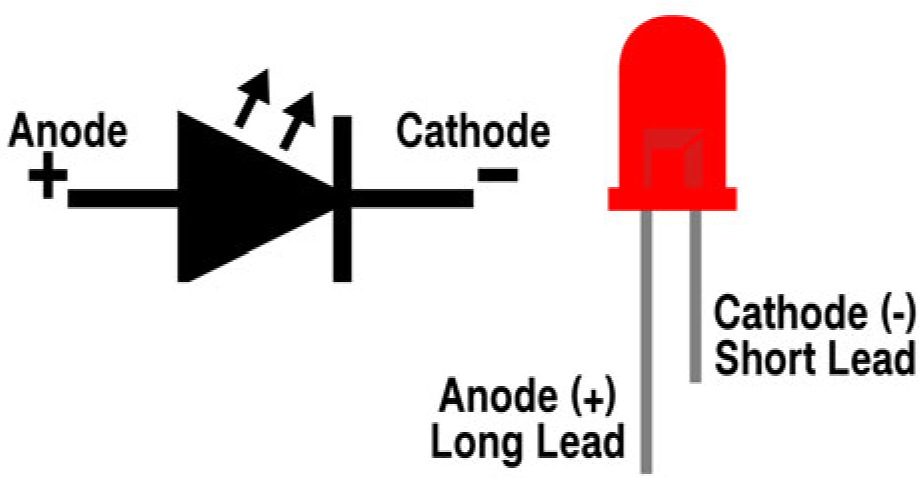


4. Wiring Diagram

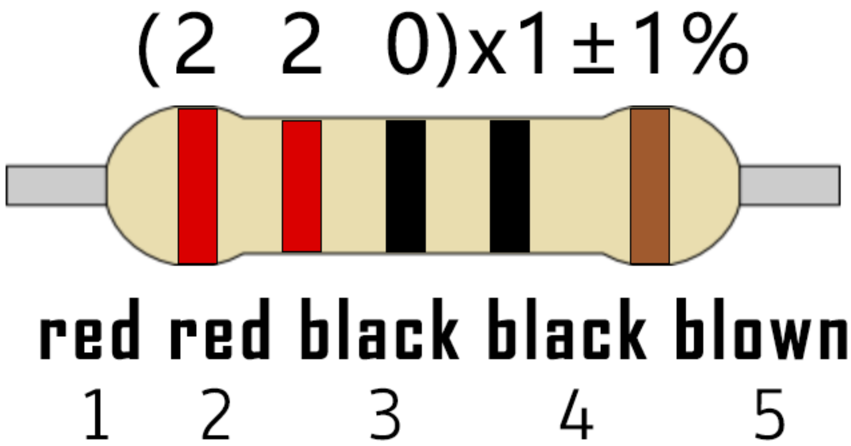


Note:

How to connect the LED



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



5. Test code

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

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

6. Test 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?

9.15 Project 15Tilt And LED

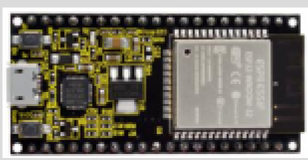
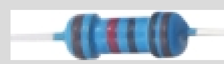
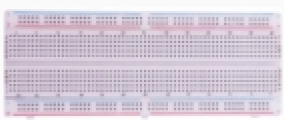
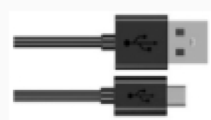
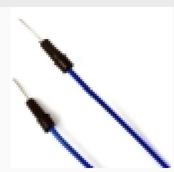
1. Introduction

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 to make an electronic hourglass.

2. Components

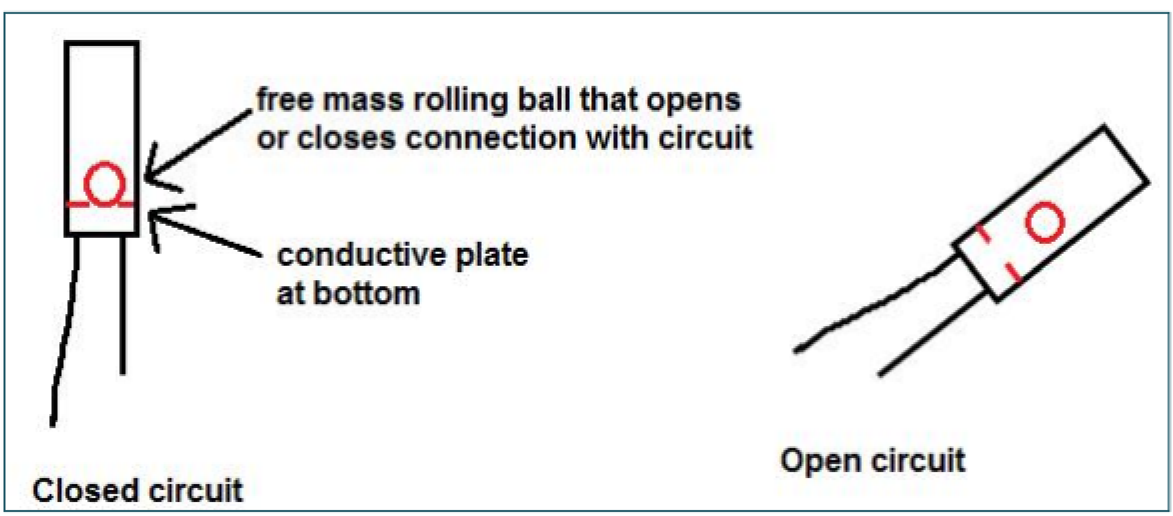
			
ESP32*1	Tilt Switch*1	Red LED*4	10K Resistor*1
			
Breadboard*1	220 Resistor*4	USB Cable*1	Jumper Wires

3. Component Knowledge

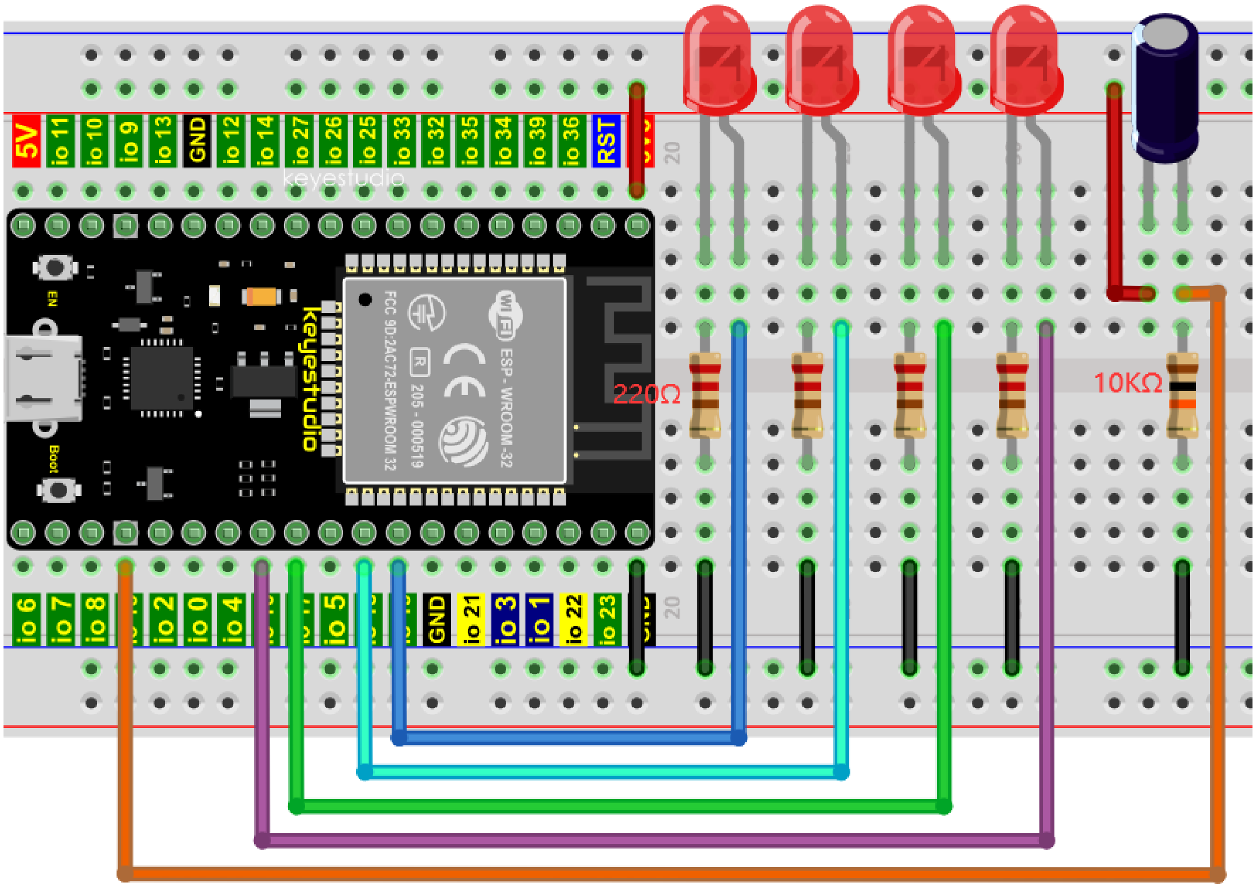
Tilt switch is also called digital switch. Inside is a metal ball that can roll. The principle of rolling the metal ball to contact with the conductive plate at the bottom, which is used to control the on and off of the circuit. When it is a rolling ball tilt sensing switch with single directional trigger,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

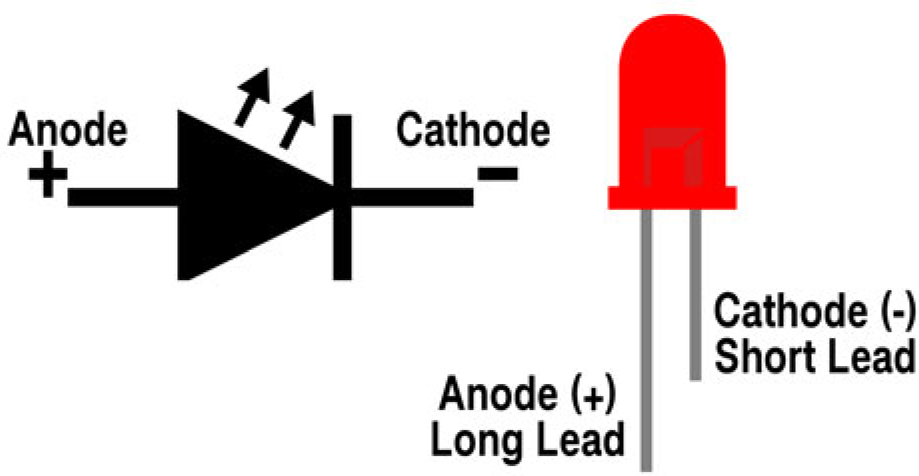


4. Wiring Diagram

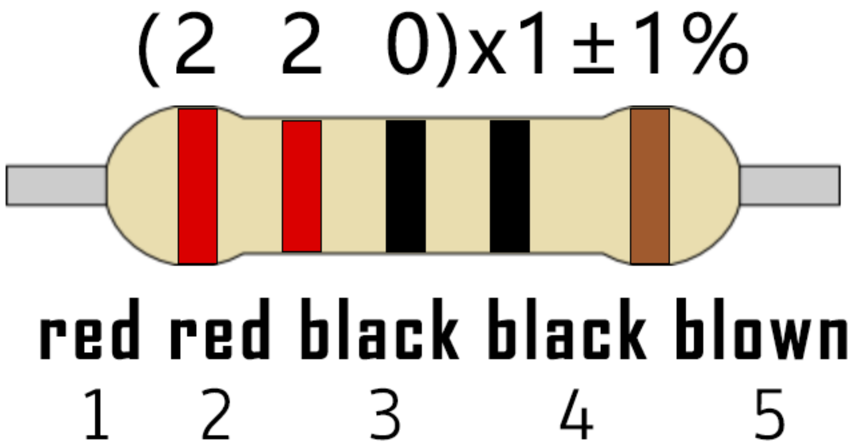


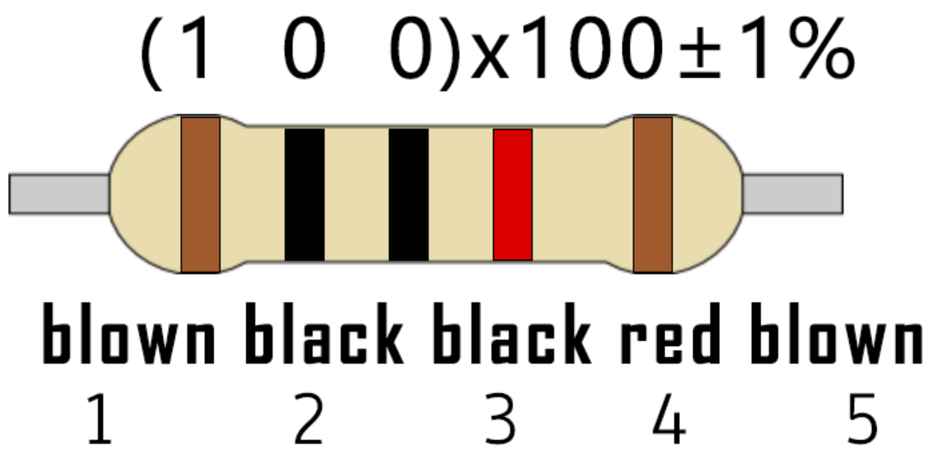
Note:

How to connect the LED



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





5. Test Code

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

(continues on next page)

(continued from previous page)

```
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. Test 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.

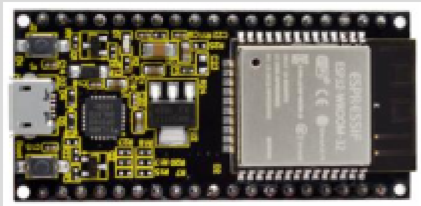
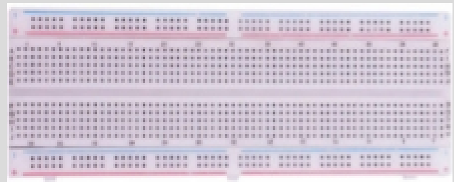
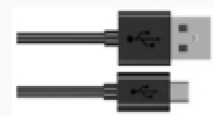
9.16 Project 16: I2C 128×32 LCD

1. Introduction

In everyday life, we can do a host of experiments with the display module and also DIY a broad menu of small objects. For example, you can make a temperature meter with a temperature sensor and a display, or make a distance meter with an ultrasonic module and a 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 show various English words, common symbols and numbers.

2. Components

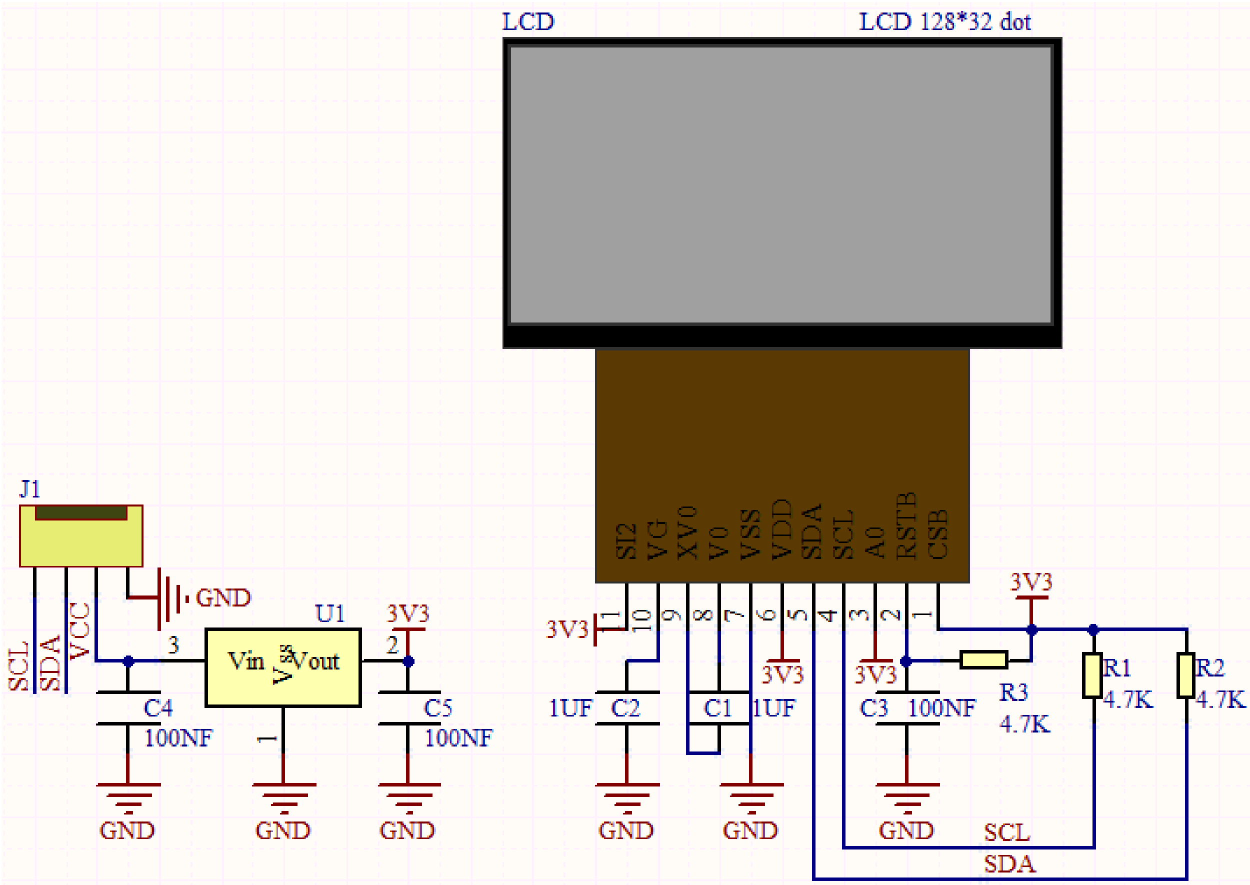
		
ESP32*1	Breadboard*1	
		
LCD_128X32_DOT*1	M-F Dupont Wires	USB Cable*1

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:

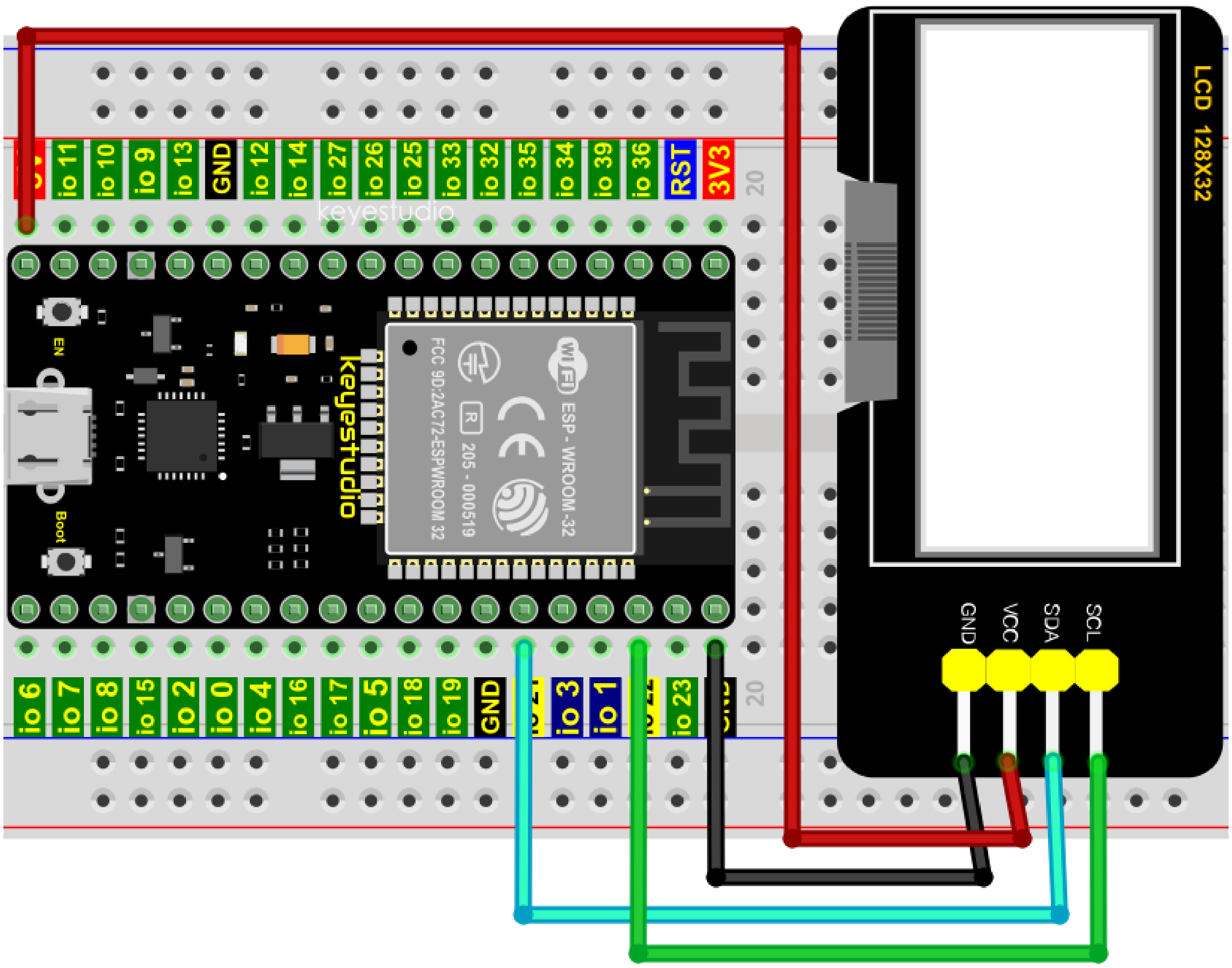
Pixel: 128*32 character

Operating voltage(chip)4.5V to 5.5V

Operating current100mA (5.0V)

Optimal operating voltage(module):5.0V

4. Wiring Diagram



5. Adding the lcd128_32_io library

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:

Click on the link to download the library file:Arduino C "lcd128_32_io.h" Librarie

6. Test Code

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

 * 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
```

(continues on next page)

(continued from previous page)

```
  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("%^&(){}:;'|?,.~\\[]");
}
//*********************************************************************************
```

6. Test Result

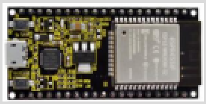
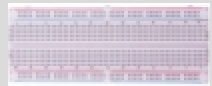
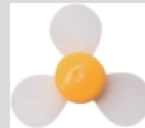
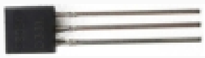
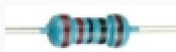
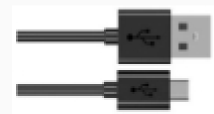
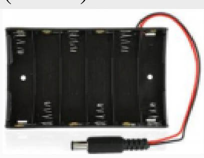
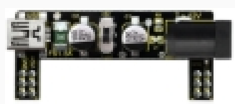
Compile and upload the code to ESP32, after the code is uploaded successfully, power up with a USB cable and you will see that the 128X32LCD module display will show"KEYESTUDIO"at the first line"ABCDEFGHIJKLMNOPQR"will be displayed at the second line"123456789±*/<>=$@"will be shown at the third line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

9.17 Project 17Small Fan

1. Introduction

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

2. Components

				
ESP32*1	DC Motor*1	Breadboard*1	Fan*1	NPN Transistor (S8050)*1
				
PNP Transistor (S8550)*1	1K Resistor*1	Jumper Wire	Diode*1	USB Cable*1
				
6 AA Battery Holder*1	Keyestudio Breadboard Power Module*1	AA Battery(Self-prepared)*6		

3. Component Knowledge:

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 DC 5V , and the output voltage of the DC 6.5-12V power supply terminal of the other power supply is DC 3.3V.

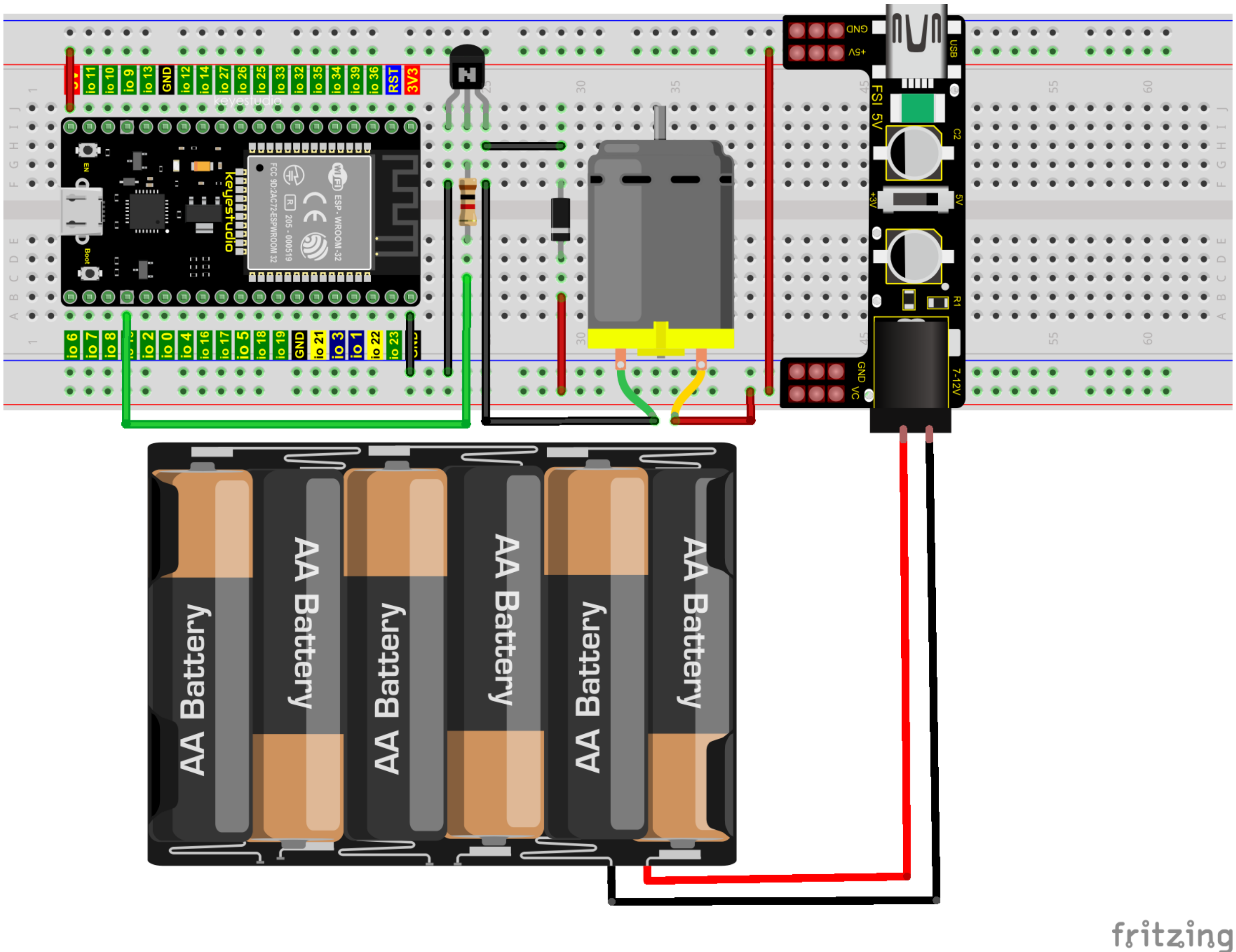
Specification:

- 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 1

We use the S8050NPN transistor) to control the motor



Wire up first, then connect a fan at the DC motor

5. Test Code 1

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

 * Filename    : Small_Fan
 * Description : S8050 triode drives the motor working
 * Auther      : http//www.keyestudio.com
   */

void setup() {

  pinMode(15, OUTPUT); // Initialize pin 15 as output.
}

void loop() {
  digitalWrite(15, HIGH);   // Turn on the motor (HIGH means HIGH level)
  delay(4000);              // Delay 4 seconds
```

(continues on next page)

(continued from previous page)

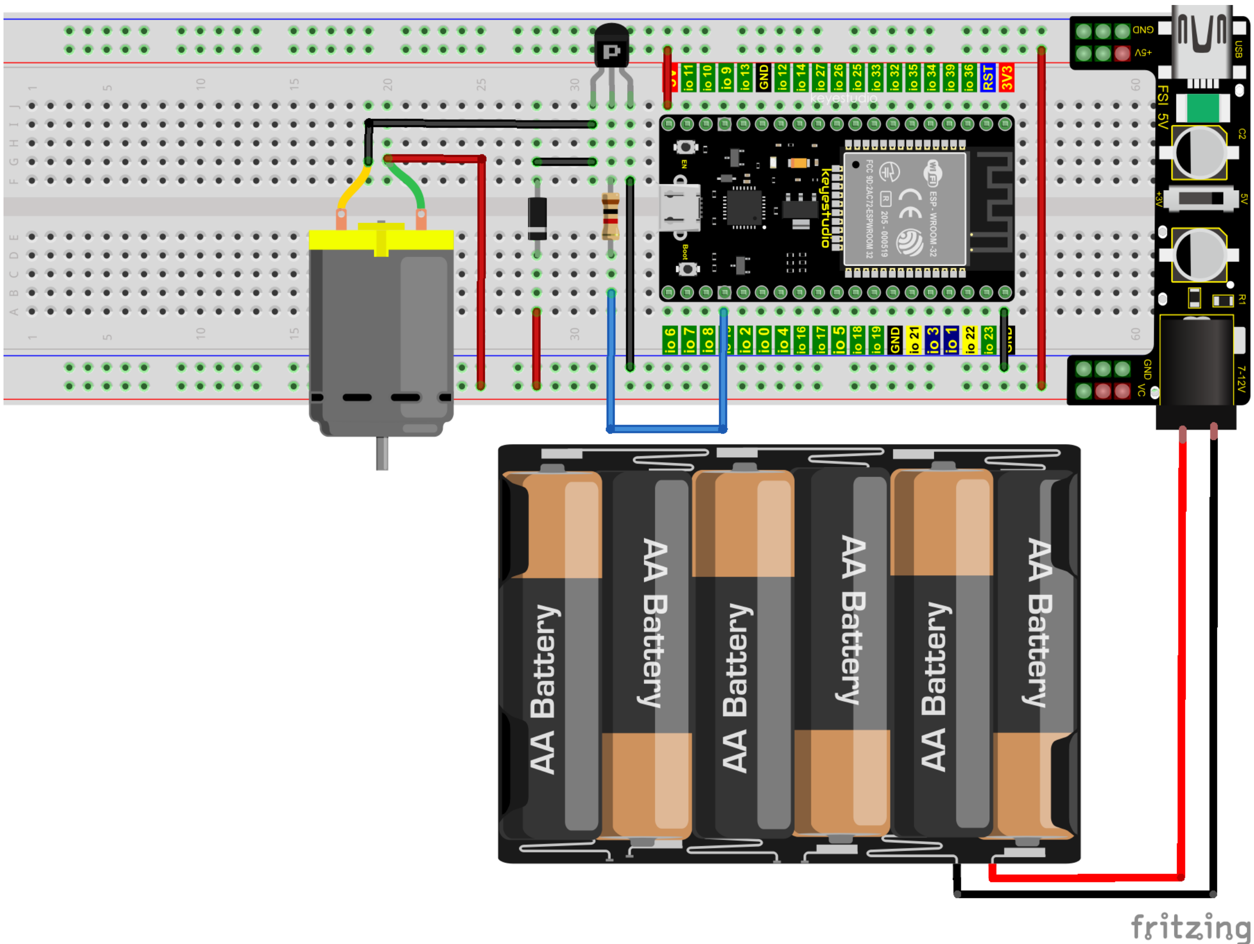
```
  digitalWrite(15, LOW);     // Reduce the voltage and turn off the motor
  delay(2000);               // Delay 2 seconds
}
//*************************************************************************************
```

6. Test Result 1

Upload the code to the ESP32 and power up. You will view the motor rotate for 4s, stop for 2s, in a loop way

7. Wiring Diagram 2

We use the S8050PNP transistor) to control the motor



Wire up first, then connect a fan at the DC motor

8. Test Code 2

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

 * Filename    : Small_Fan
 * Description : S8550 triode drives the motor working
 * Auther      : http//www.keyestudio.com
   */
```

(continues on next page)

(continued from previous page)

```
void setup() {

  pinMode(15, OUTPUT); // Initialize pin 15 as output.
}

void loop() {
  digitalWrite(15, LOW);   // Turn on the motor (LOW means LOW level)
  delay(4000);              // Delay 4 seconds
  digitalWrite(15, HIGH);    // Raise the voltage and turn off the motor
  delay(2000);              // Delay 2 seconds
}
//*************************************************************************************
```

9. Test Result 2

Upload the code to the ESP32 and power up. You will view the motor rotate for 4s, stop for 2s, in a loop way

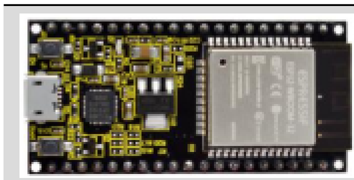
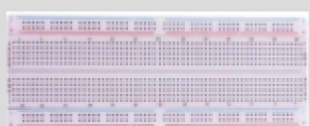
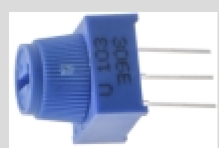
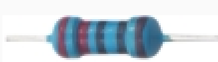
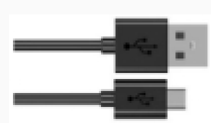
9.18 Project 18: Dimming Light

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.

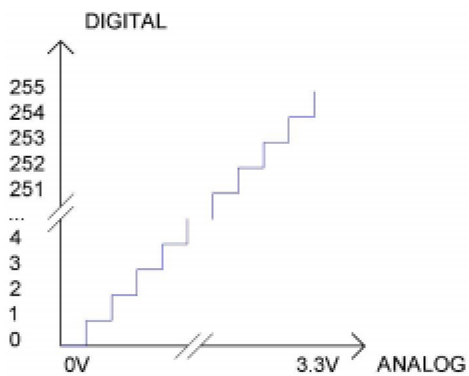
2. Components

			
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:

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

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

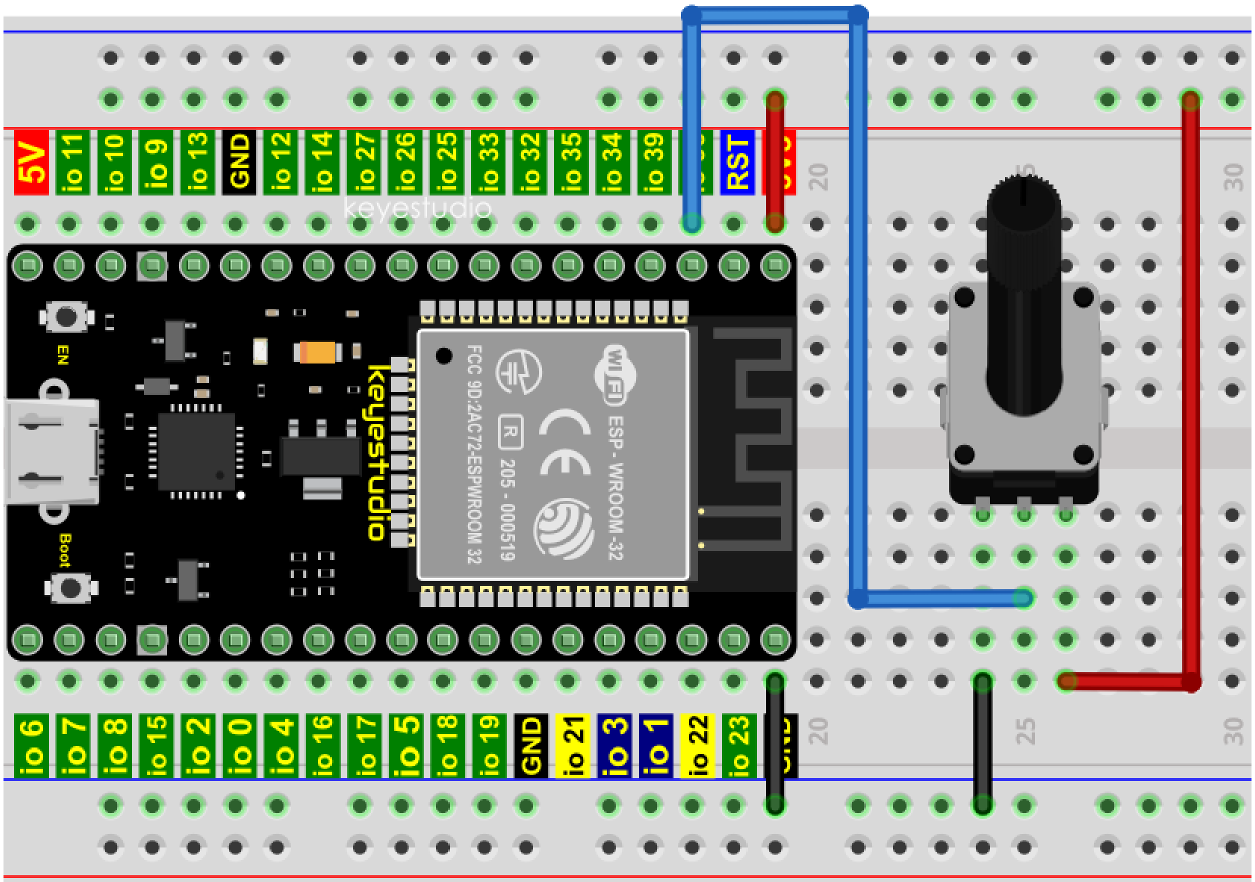
ESP32 has two 8-bit digital analog converters to be connected to GPIO25 and GPIO26 pins, respectively, and it is immutable. As shown in the following table

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.

4. Read the values of the potentiometer

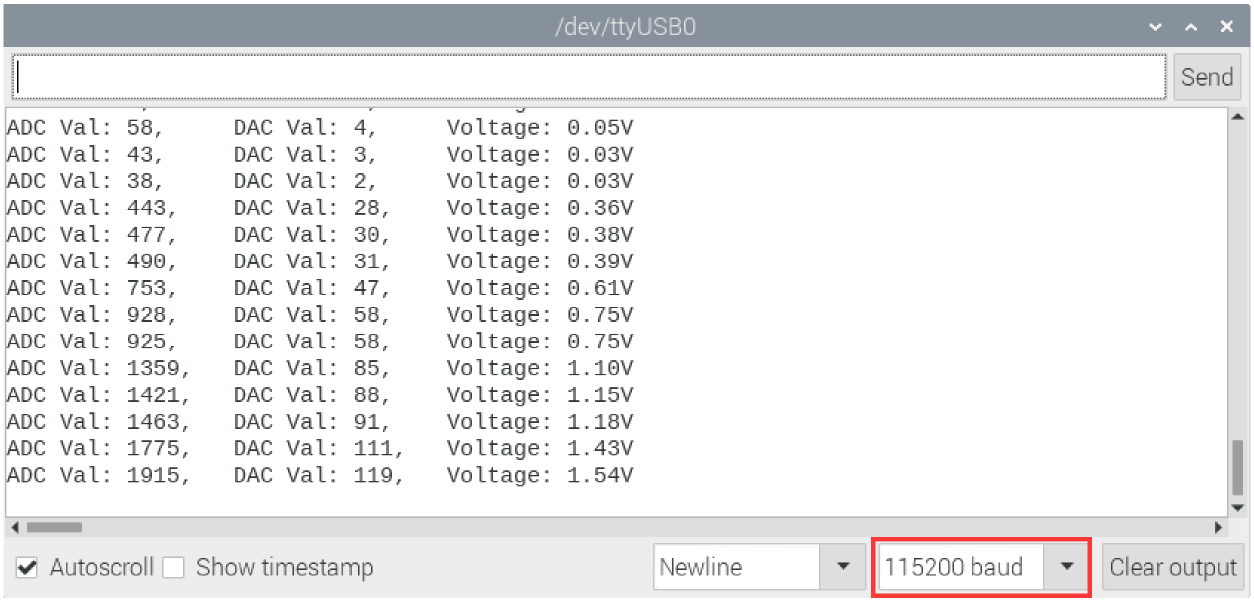
We connect the potentiometer to the analog IO port of ESP32 to read the ADC value, DAC value and voltage value of the potentiometer, please refer to the wiring diagram below




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

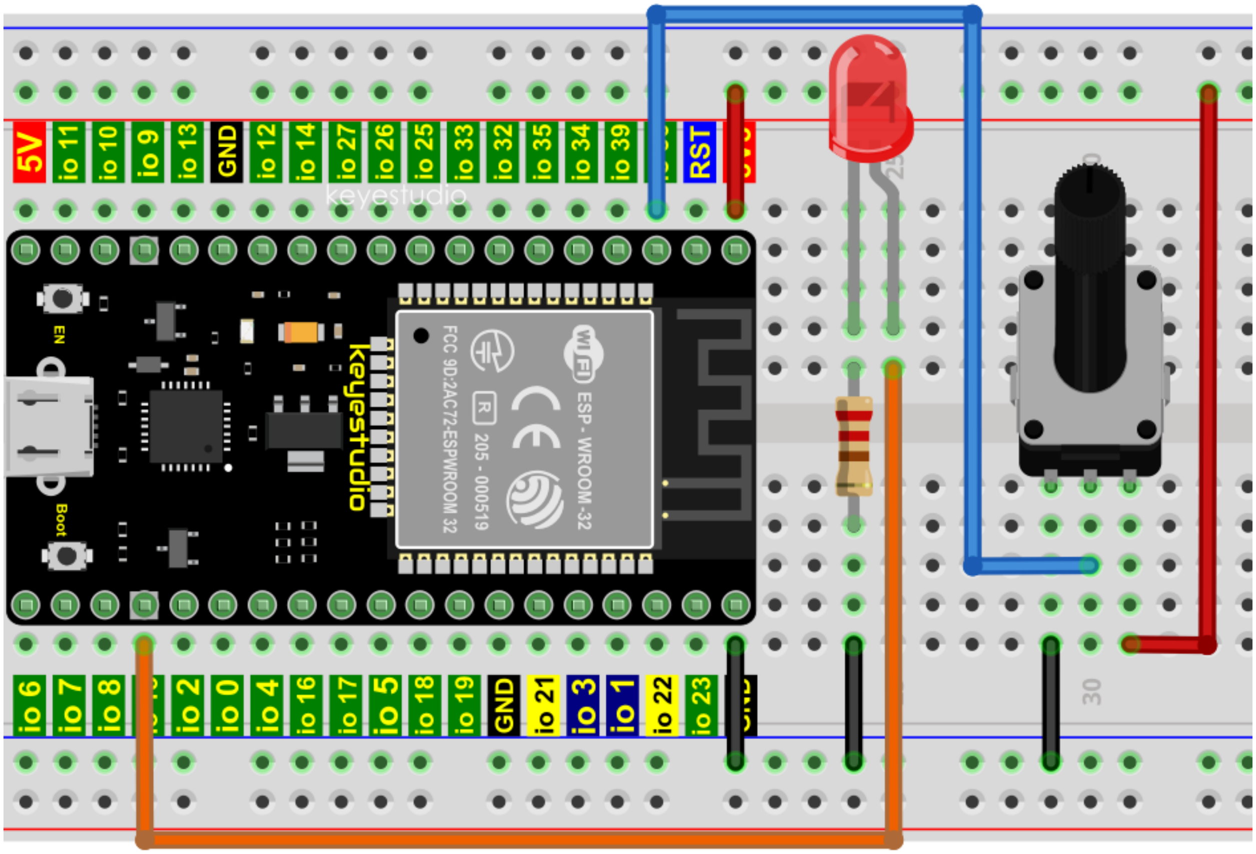
Compile and upload the code to ESP32, after the code is uploaded successfully, power up with a USB cable, open the serial monitor and set the baud rate to 115200 and press the reset button first.

You will see that the serial 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 follow:

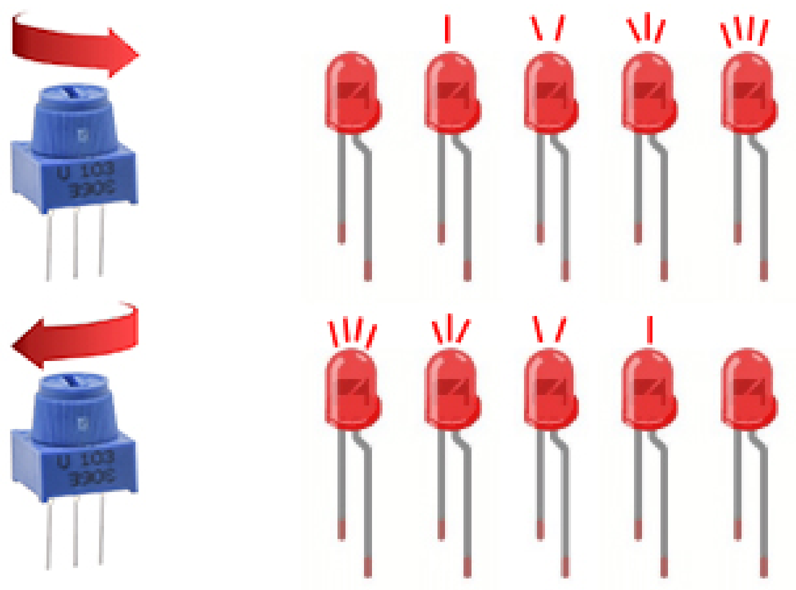


6. Test Code

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

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

6. Test 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.

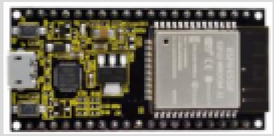
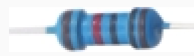
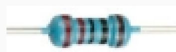
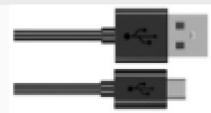


9.19 Project 19Flame Alarm

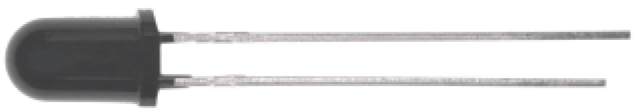
1. Introduction

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. Component

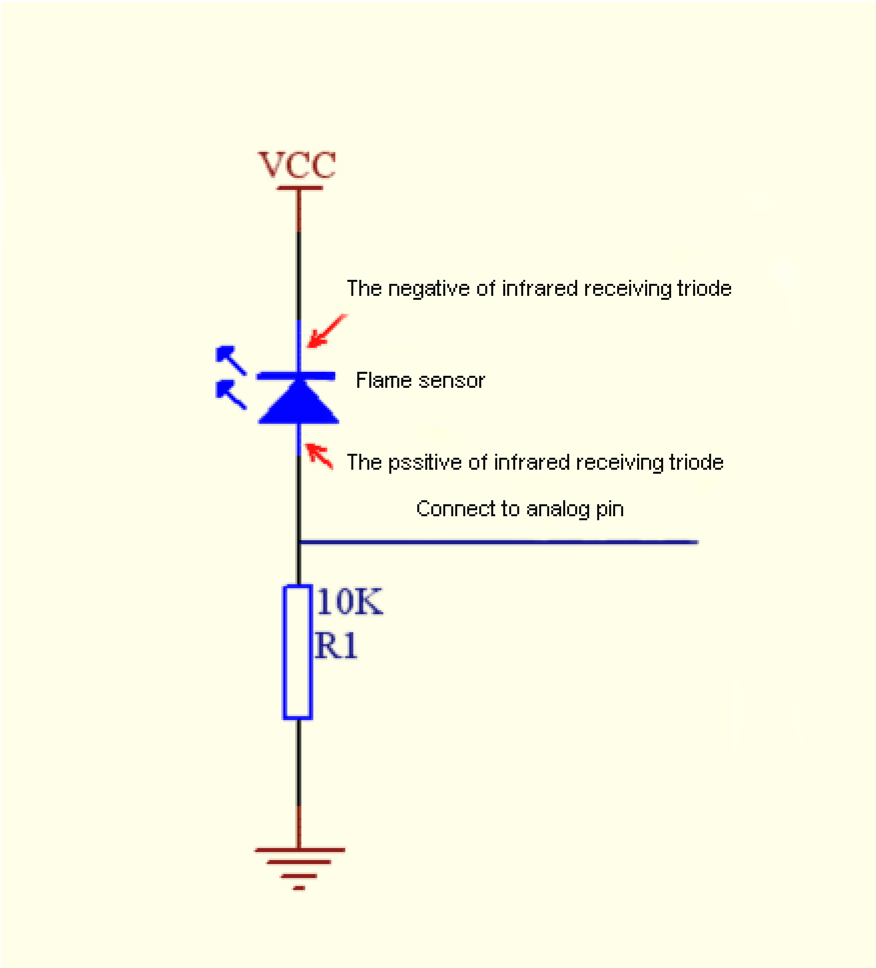
			
ESP32*1	Breadboard*1	Red LED*1	Active Buzzer*1
			
Flame Sensor*1	220 Resistor*1	10KResistor*1	Jumper Wires
			
NPN Transistor(S8050)*1	1k Resistor*1	USB Cable*1	

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.

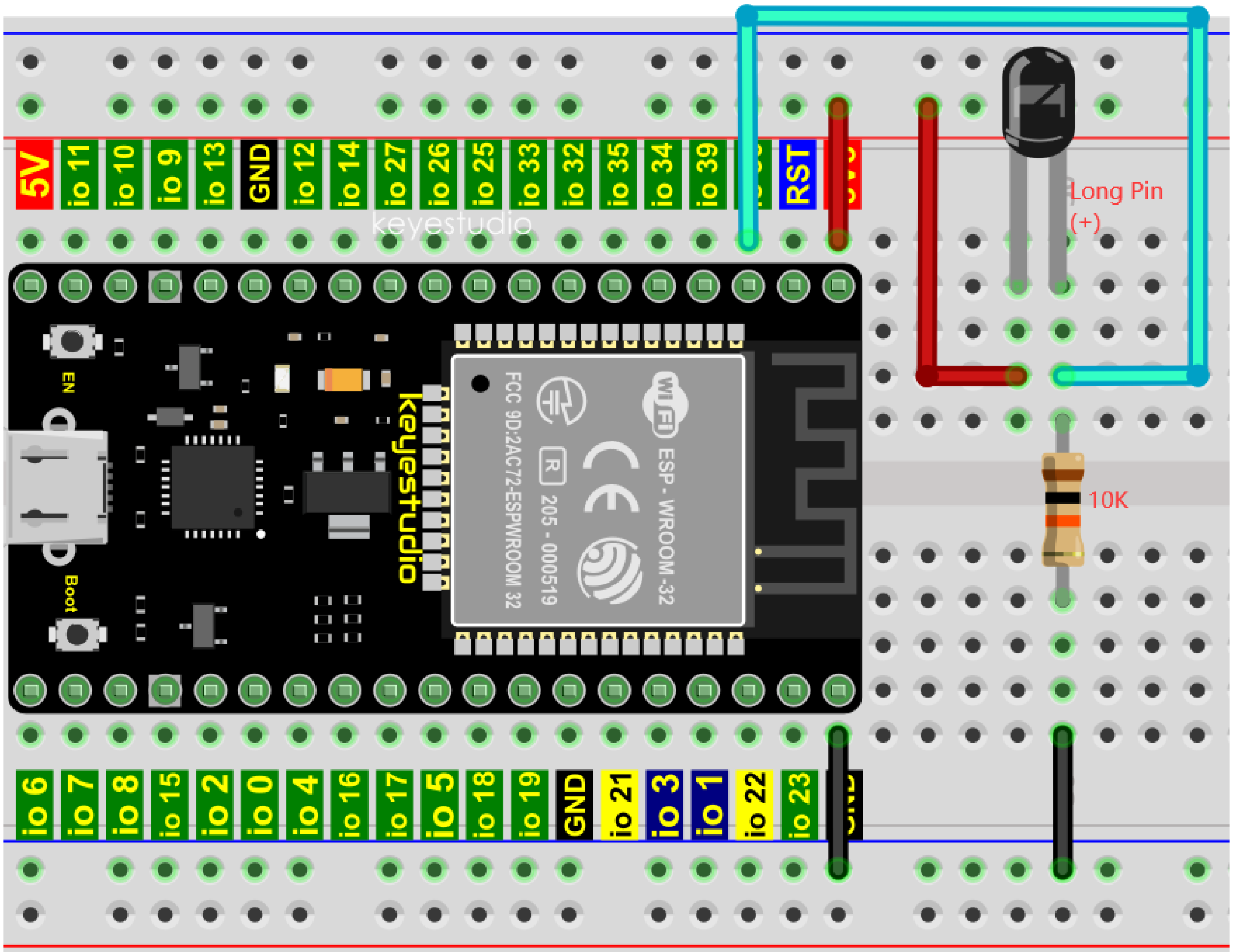
The short pin of the receiving triode is negative pole and the other long pin is positive pole. We should connect the short pin (negative) to 5V and the long pin (positive) to the analog pin, a resistor and GND. As shown in the figure below



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 values 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




```
//**********************************************************************************
/*
 * 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,␣
```

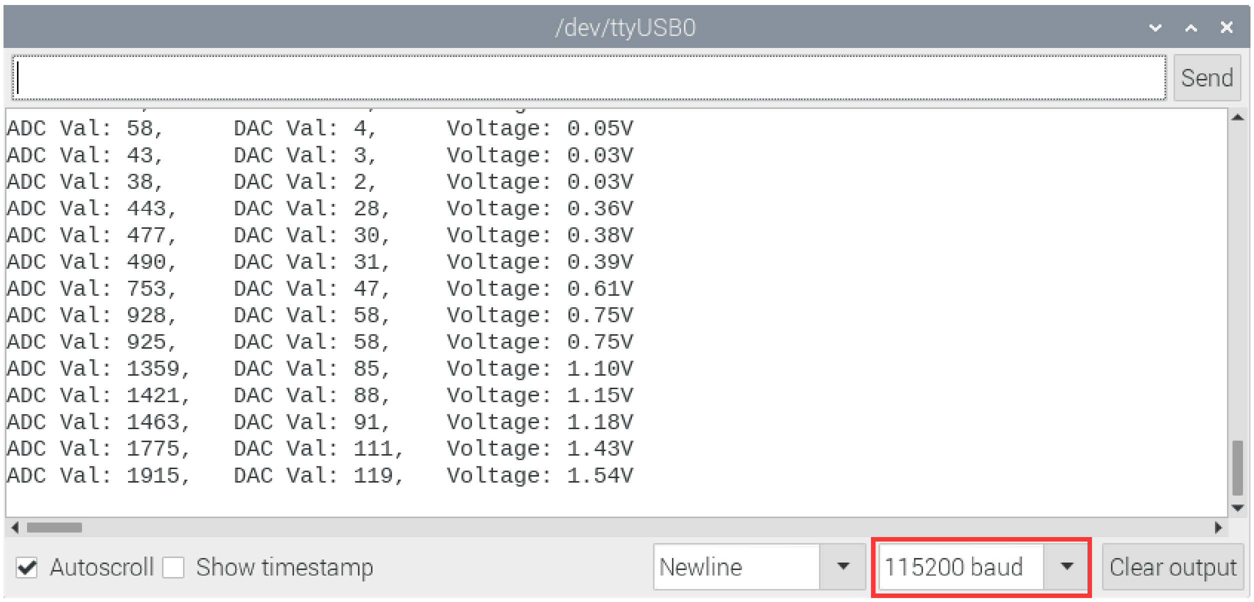
(continues on next page)

(continued from previous page)

```
voltage);
  delay(200);
}
//*************************************************************************************
```

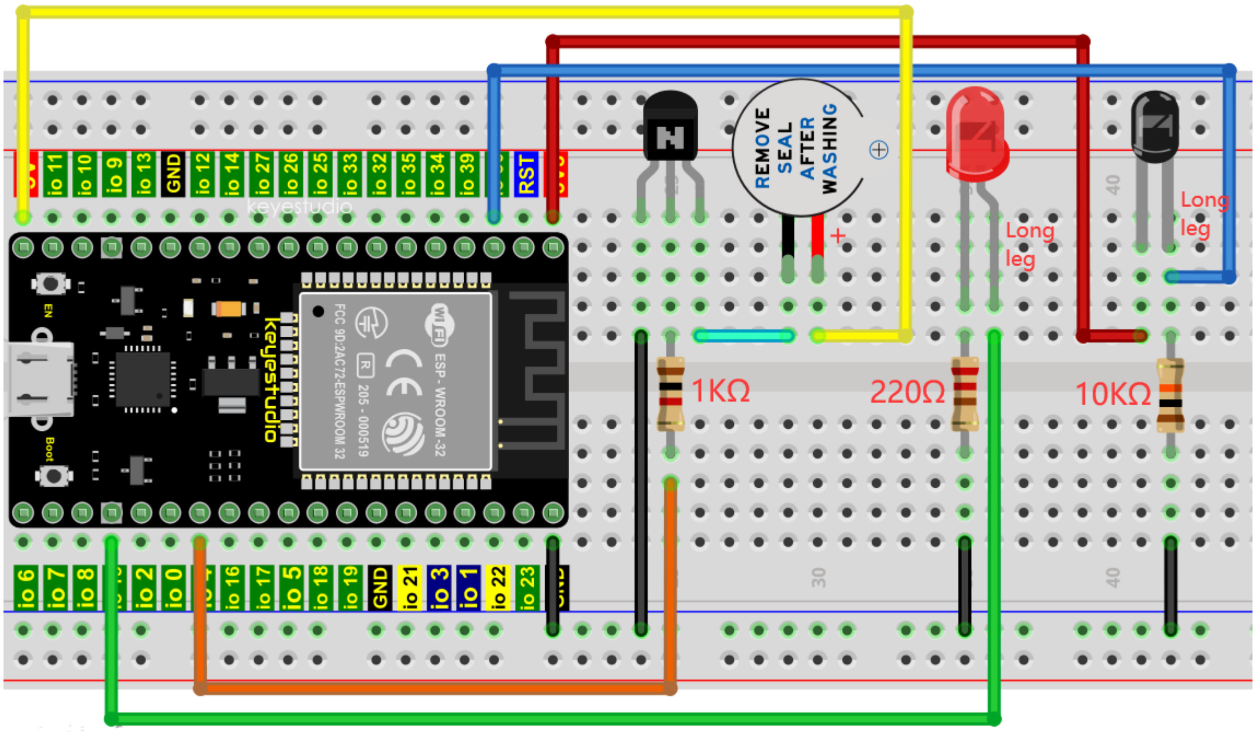
Compile and upload the code to ESP32, after the code is uploaded successfully, power up with a USB cable, open the serial monitor and set the baud rate to 115200 and press the reset button first.

You will see that the serial 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.



6. Test Code

Note: `if adcVal >500:` the threshold of 500 in the code can be reset as required)

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

 * 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
```

(continues on next page)

(continued from previous page)

```
  }
 else
 {
    digitalWrite(PIN_LED, LOW);  //turn off LED
    digitalWrite (PIN_BUZZER, LOW); //turn off buzzer
  }
}
//*********************************************************************************
```

7. Test 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.

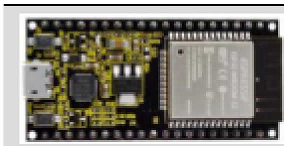
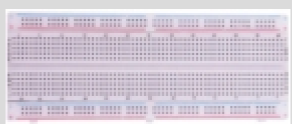
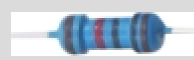
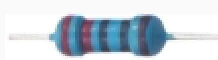
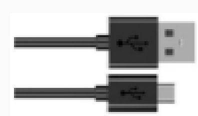
9.20 Project 20: Night Lamp

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. 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.

In this project, we use a ESP32 to control a LED to achieve the effect of the street light.

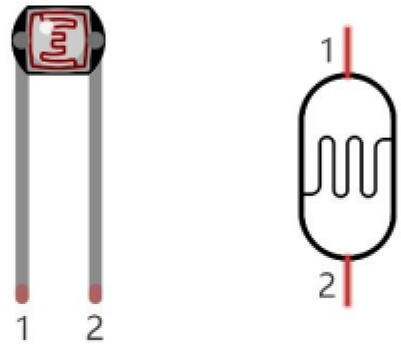
2. Components

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

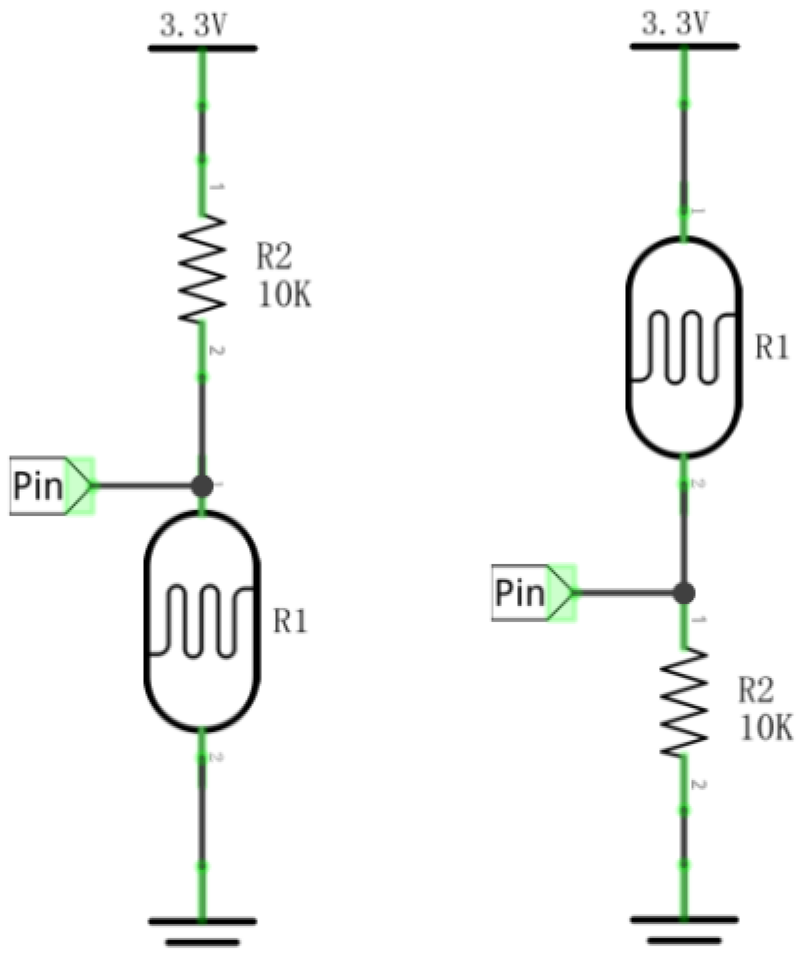
3. Component Knowledge

docs/media/9e553e75b6f976f33438171eb2f2e775-1699344906400-78.png

Photoresistor : It is a kind of photosensitive resistor, 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 photoresistor to detect the light intensity. Photoresistor and its electronic symbol are as follows



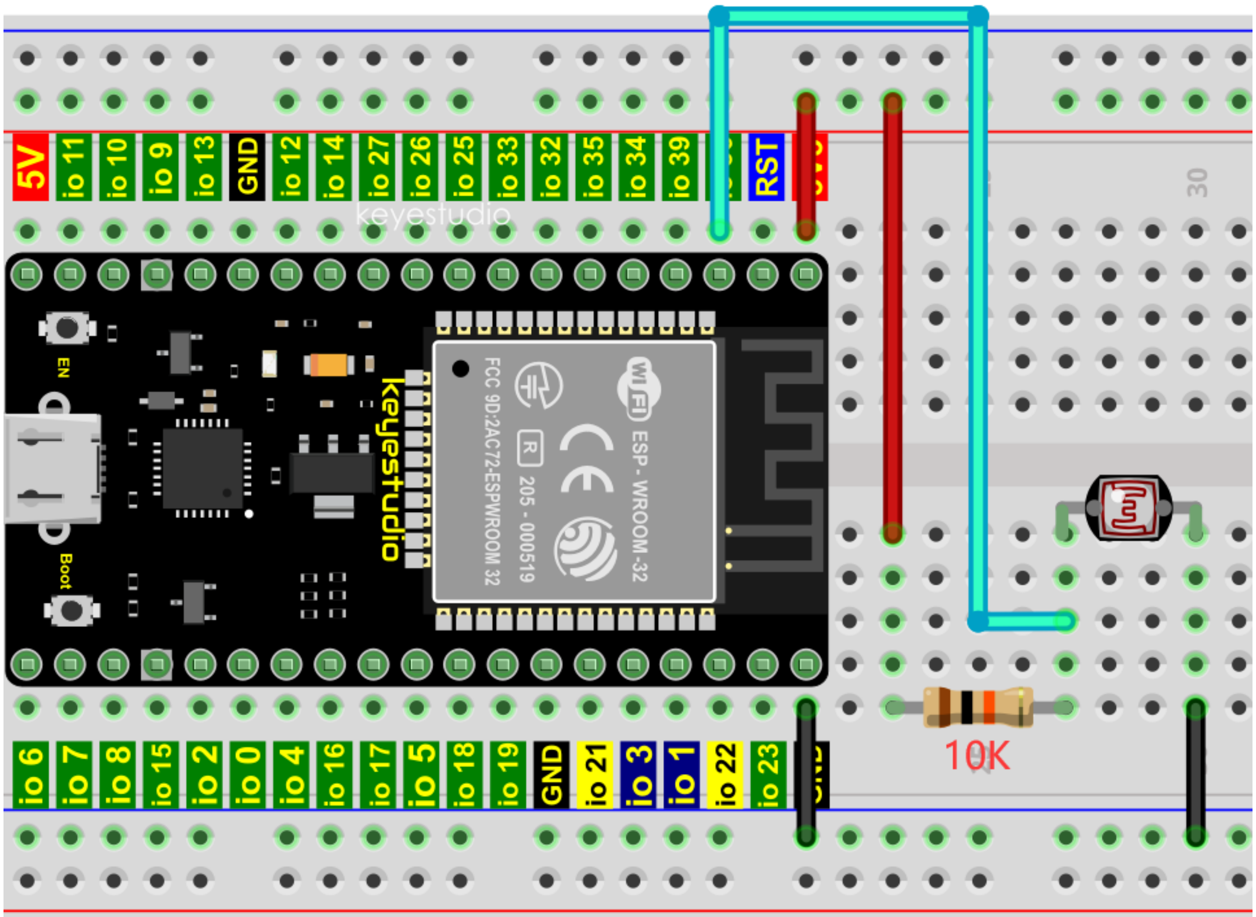
The following circuit is used to detect changes in resistance values of photoresistors



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

4. Read the values 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



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

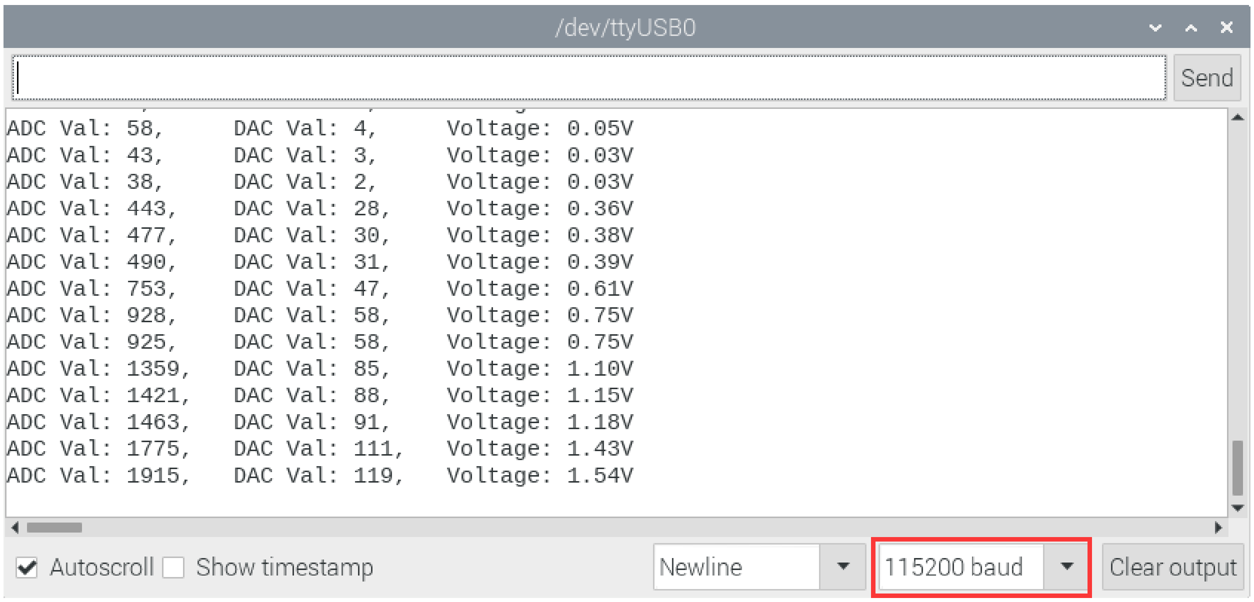
(continues on next page)

(continued from previous page)

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

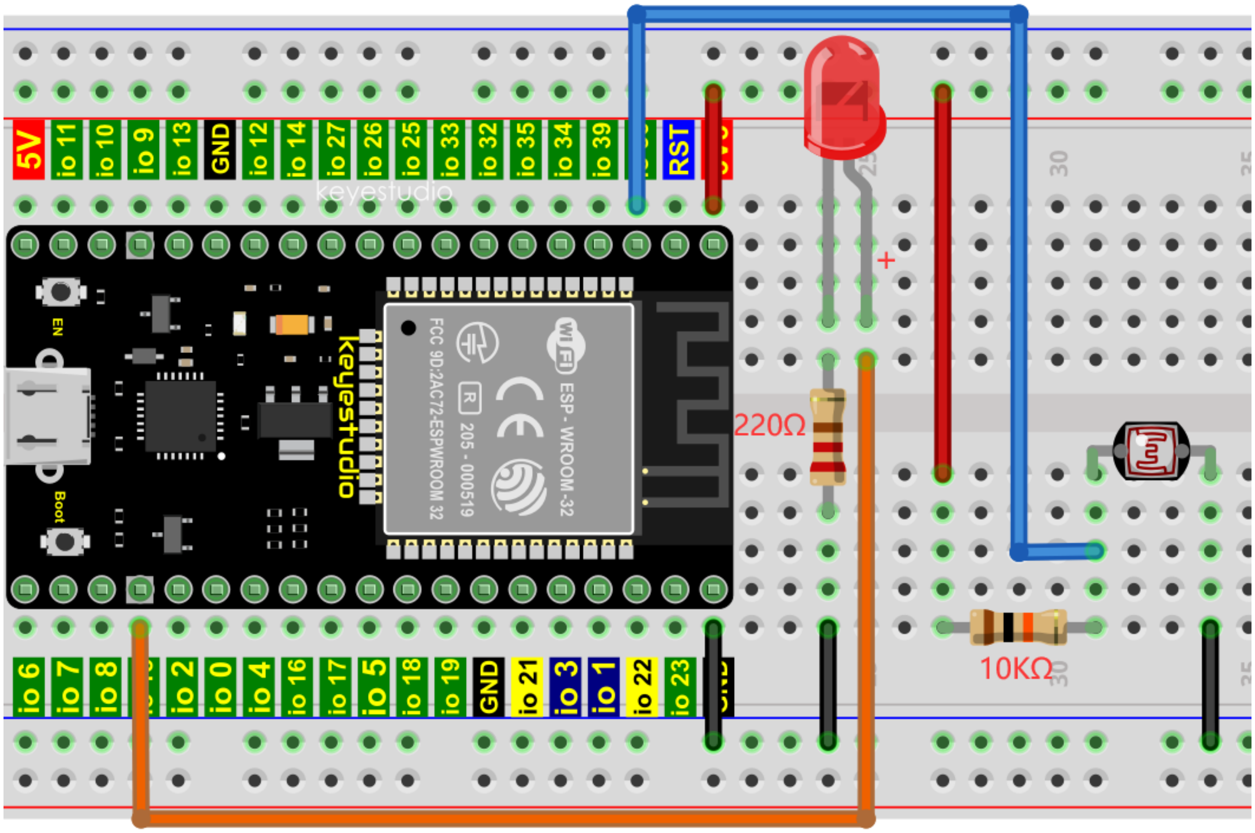
Compile and upload the code to ESP32, after the code is uploaded successfully, power up with a USB cable, open the serial monitor and set the baud rate to 115200 and press the reset button first.

You will see that the serial 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 decrease gradually.



5. Wiring diagram of the light-controlled lamp

Now we will make a light controlled lamp. The principle is the same as the small dimming lamp , that is, the ESP32 takes the analog values of the sensor, and then adjusts the brightness of the LED.



6. Test Code

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

 * 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. Test Result

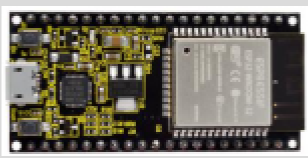
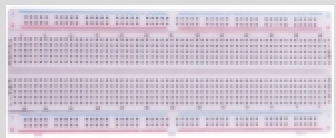
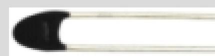
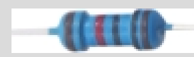
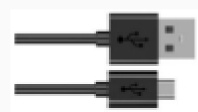
Compile and upload the code to ESP32, after the code is uploaded successfully, power up with a USB cable and you will see that when the intensity of light around the photoresistor is reduced, the LED will be bright, on the contrary, the LED will be dim.

9.21 Project 21Temperature Instrument

1.Introduction

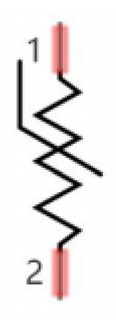
Thermistor is a kind of resistor whose resistance depends on temperature changes, which is widely used in gardening, home alarm systems and other devices. Therefore, we can use the features to make a temperature instrument.

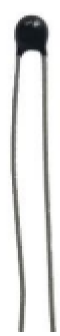
2.Components

			
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: It 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 to detect temperature intensity. The thermistor and its electronic symbol are shown below:





The relationship between resistance and temperature of the thermistor is

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

Rt is the thermistor resistance under T2 temperature;

R is the nominal resistance of thermistor under T1 temperature;

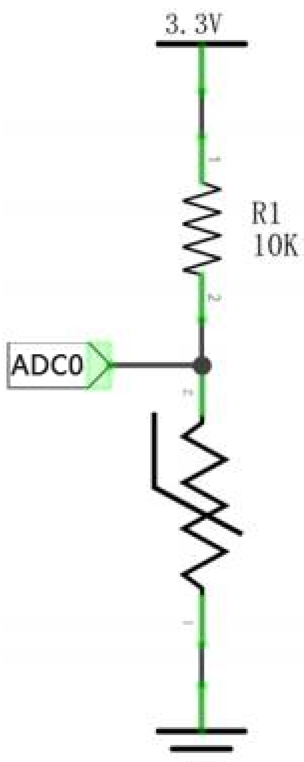
EXP[n] is nth power of e;

B is temperature index;

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

Parameters : B=3950, R=10k, T1=25.

The circuit connection method of the thermistor is similar to the photoresistor, as shown below



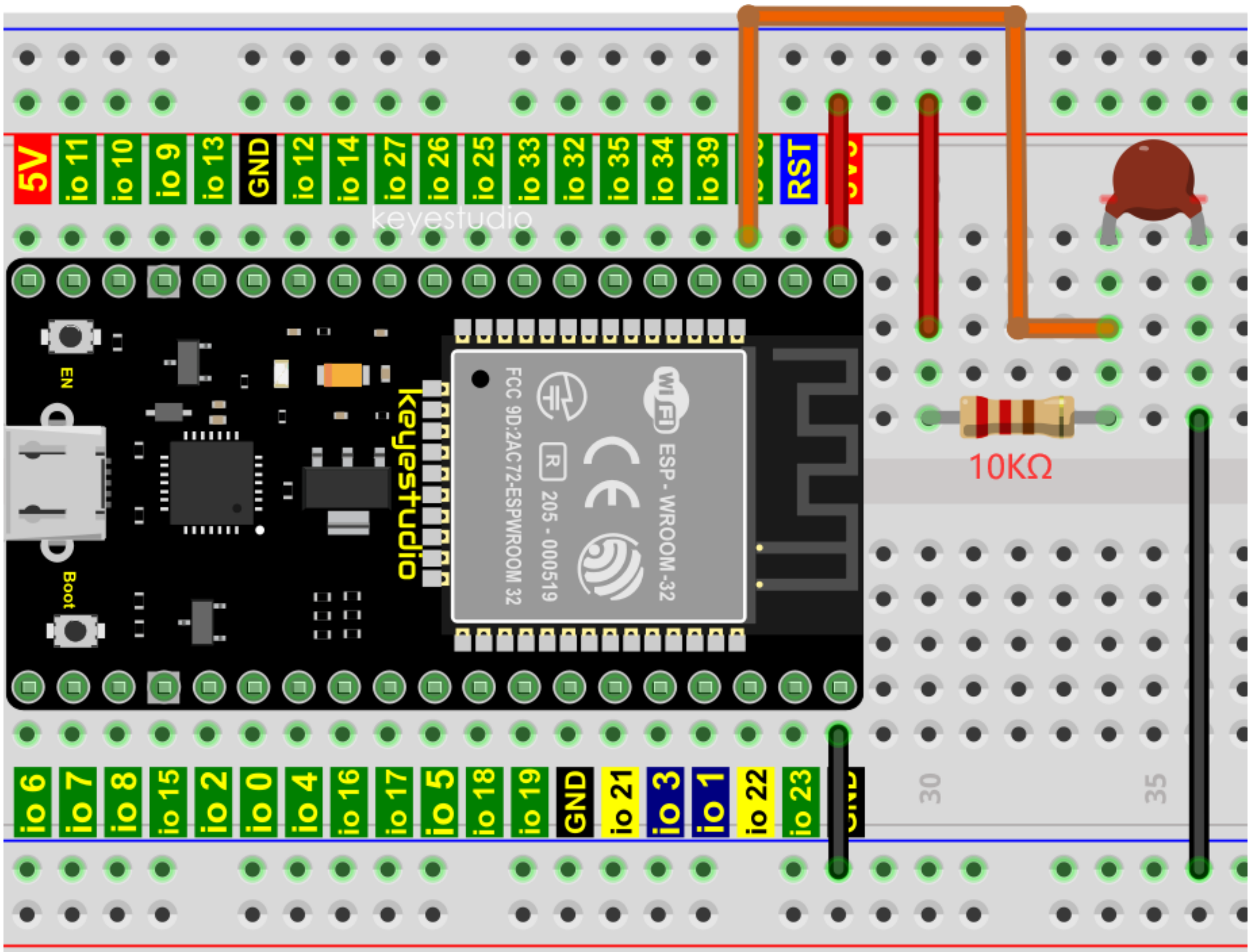
We can use the value measured by the ADC converter to obtain the resistance of thermistor, and then we can use the formula to obtain the temperature value.

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 reading the current ADC value, voltage value and temperature value and print them out. Please connect the wirings according to the wiring diagram below




```
//**********************************************************************************
/*
 * Filename    : Thermomter
 * Description : Making a thermometer by thermistor.
 * Auther      : http//www.keyestudio.com
*/
```

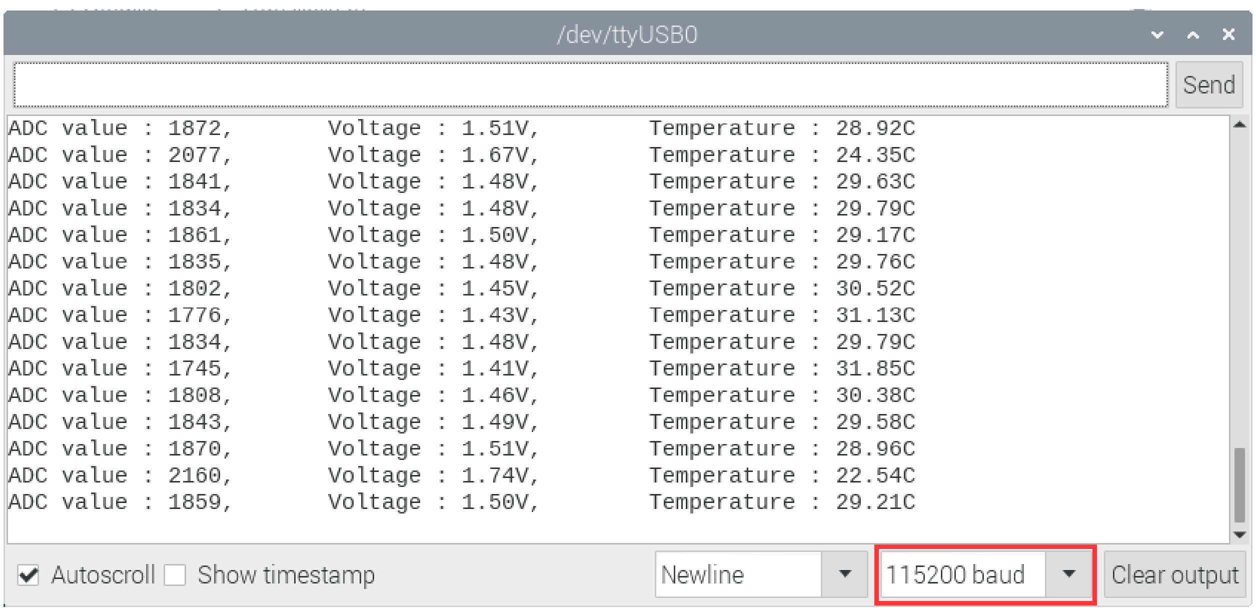
(continues on next page)

(continued from previous page)

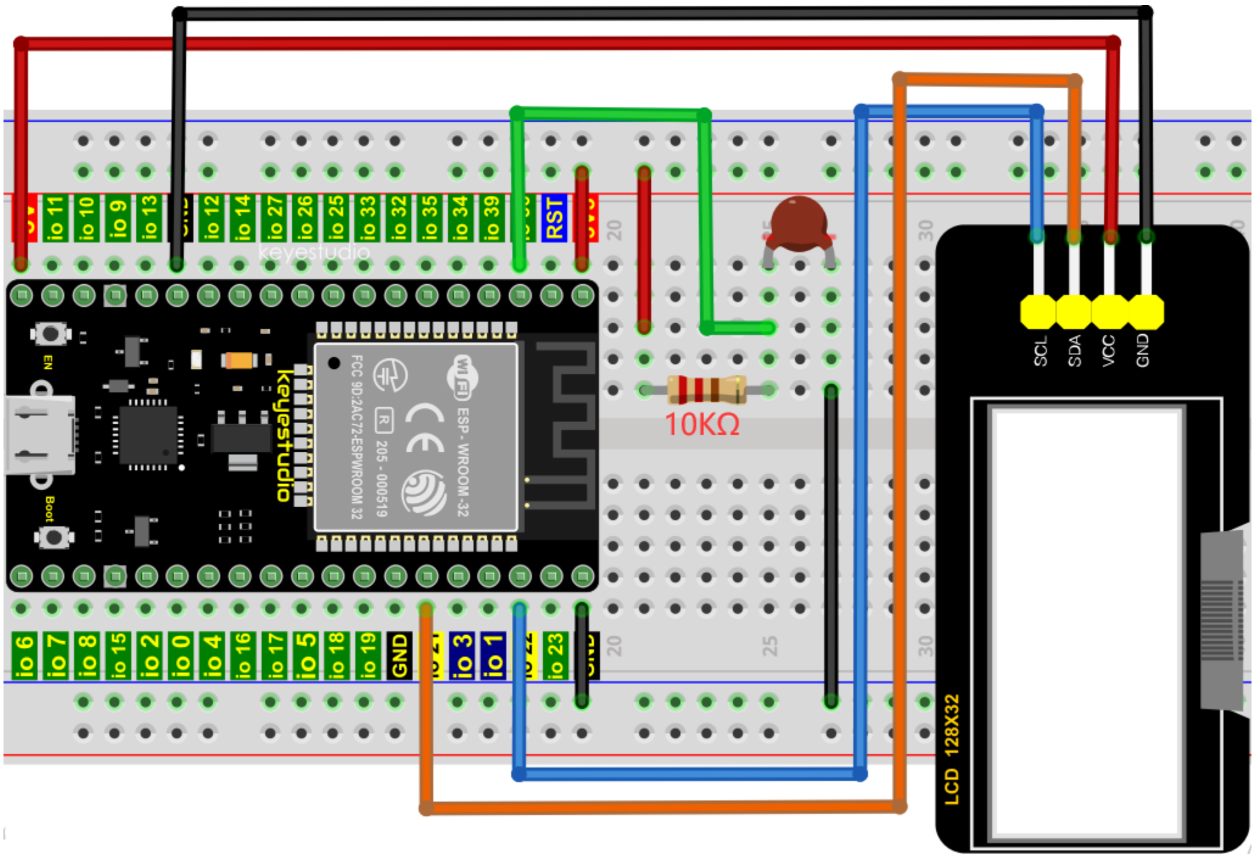
```
#define PIN_ANALOG_IN   36
void setup() {
  Serial.begin(115200);
}

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

Upload the code to the ESP32, power up with a USB cable, open serial monitor and set baud rate to 115200. Press the rest button of the ESP32 board,then you will view ADC value, voltage value and temperature value of the thermistor displayed.

Pinch the thermistor a while, the temperature value will increase.

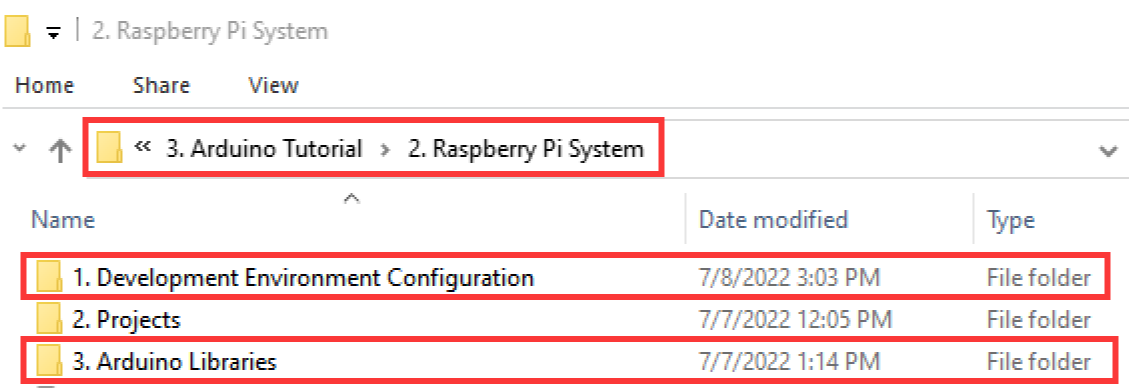


5.Wiring diagram of the temperature instrument



6.Adding the lcd128_32_io library

Please add the **lcd128_32_io** library first



7.Test Code

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

(continues on next page)

(continued from previous page)

```
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.Test Result

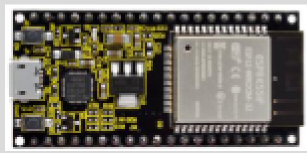
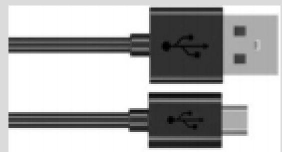
Compile and upload the code to ESP32, after the code is uploaded successfully, power up with a USB cable and you will see that the LCD 128X32 DOT displays the voltage value of the thermistor and the temperature value in the current environment.

9.22 Project 22Bluetooth

This chapter mainly introduces how to make simple data transmission through Bluetooth of ESP32 and mobile phones. Project 22.1 is classic Bluetooth while project 22.2 is Bluetooth control LED.

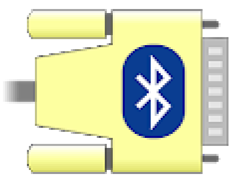
9.22.1 Project 22.1Classic Bluetooth

1.Components

	
ESP32*1	USB Cable*1

In this tutorial we need to use a Bluetooth APP called serial Bluetooth terminal to assist in the experiment.

Download link: https://www.appsapk.com/serial-bluetooth-terminal/ .



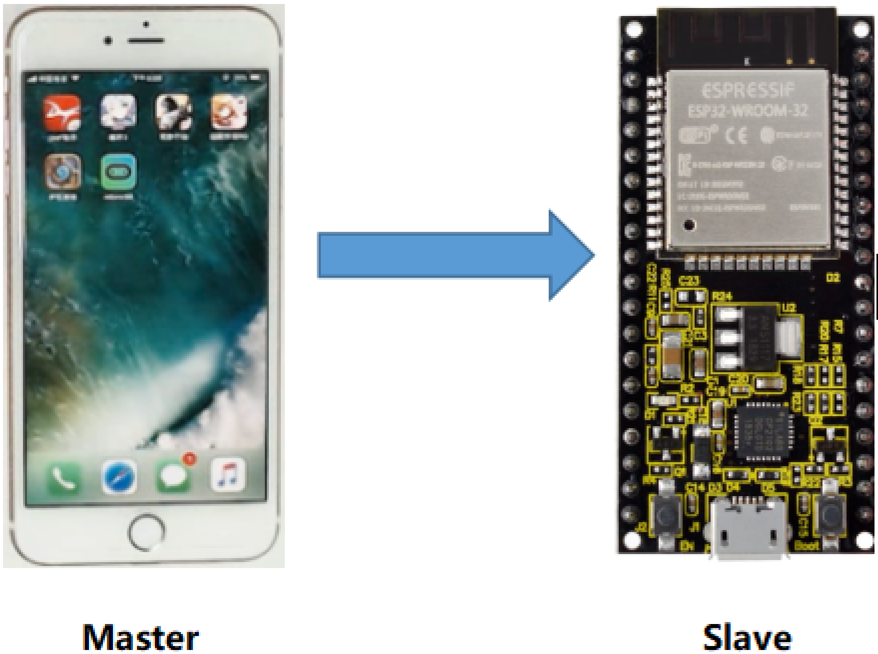
2.Component Knowledge

Bluetooth is a short-distance communication system that can be divided into two types, namely low power bluetooth (BLE) and classic bluetooth. There are two modes for simple data transfer: master mode and slave mode.

Master Mode: In this mode, work is done on the master device and can be connected to the slave device. When the device initiates a connection request in the main mode, information such as the address and pairing password of other bluetooth devices are required. Once paired, you can connect directly to them.

Slave Mode: A bluetooth module in the slave mode can only accept connection requests from the host, but cannot initiate connection requests. After being connected to a host device, it can send and receive data through the host device.

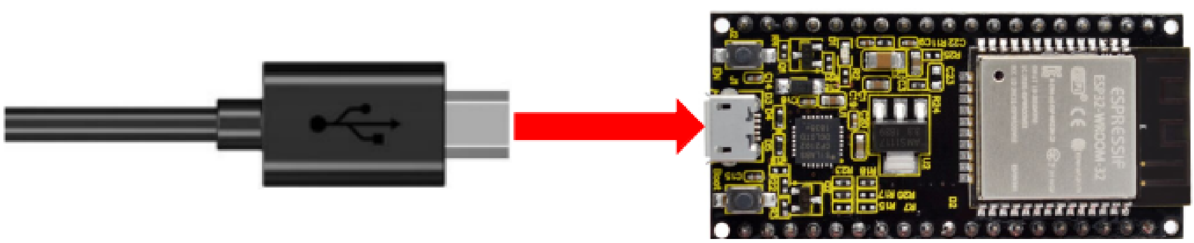
Bluetooth devices can interact with each other, when they interact, the bluetooth device in the main mode searches for nearby devices. While a connection is established, they can exchange data. For example, when a mobile phone exchanges data with ESP32, the mobile phone is usually in master mode and the ESP32 is in slave mode.



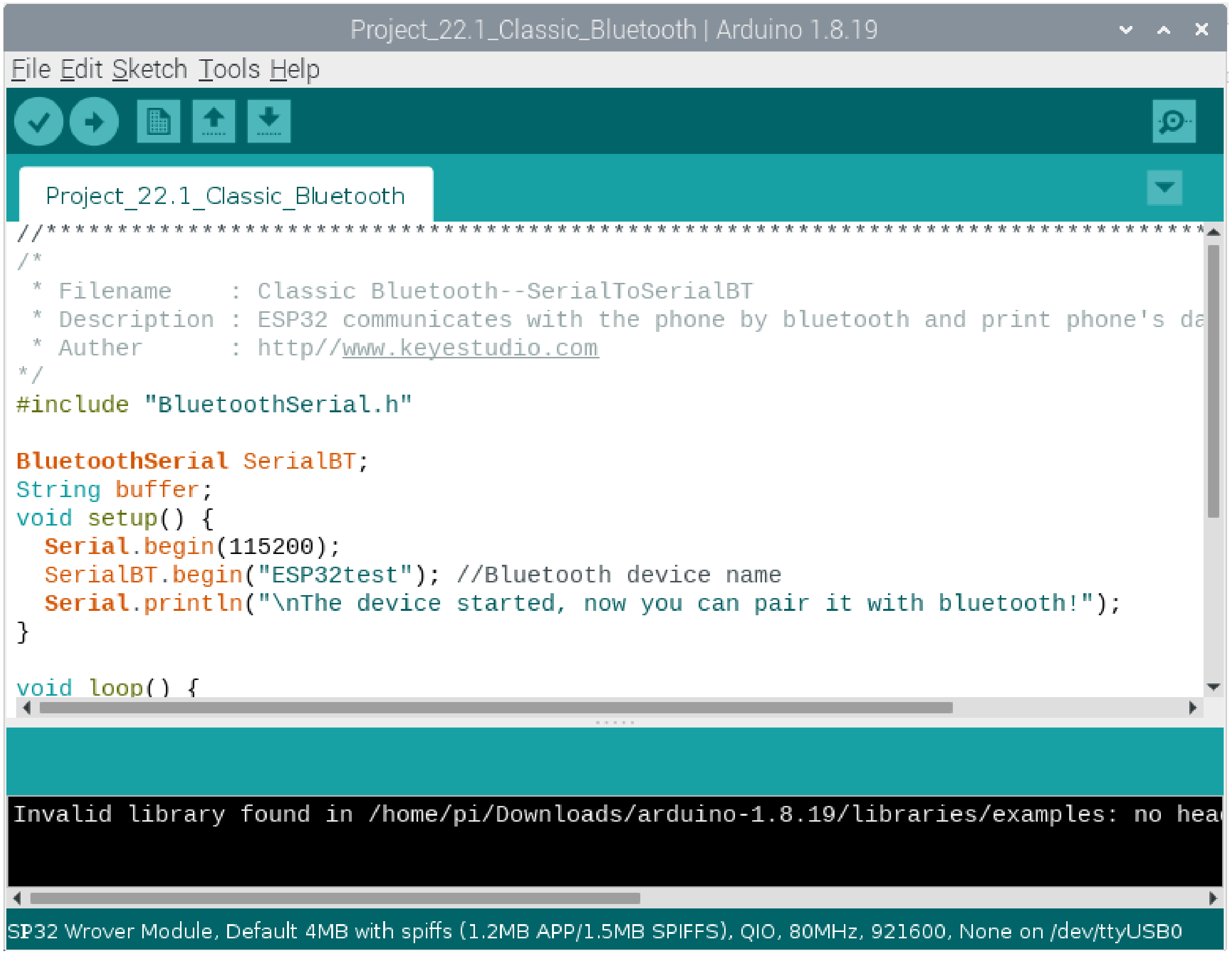
Master Slave

3.Wiring Diagram

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



4.Test Code




```
//**********************************************************************************
/*
 * 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()) {
```

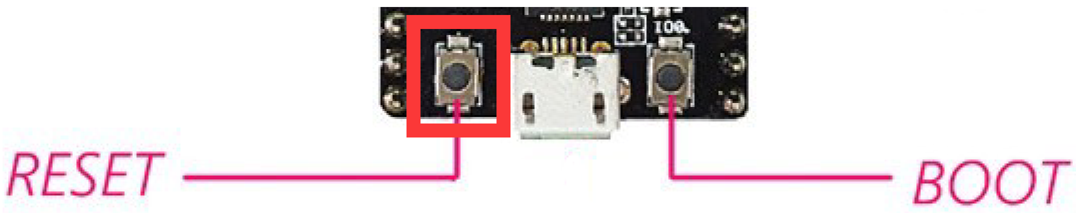
(continues on next page)

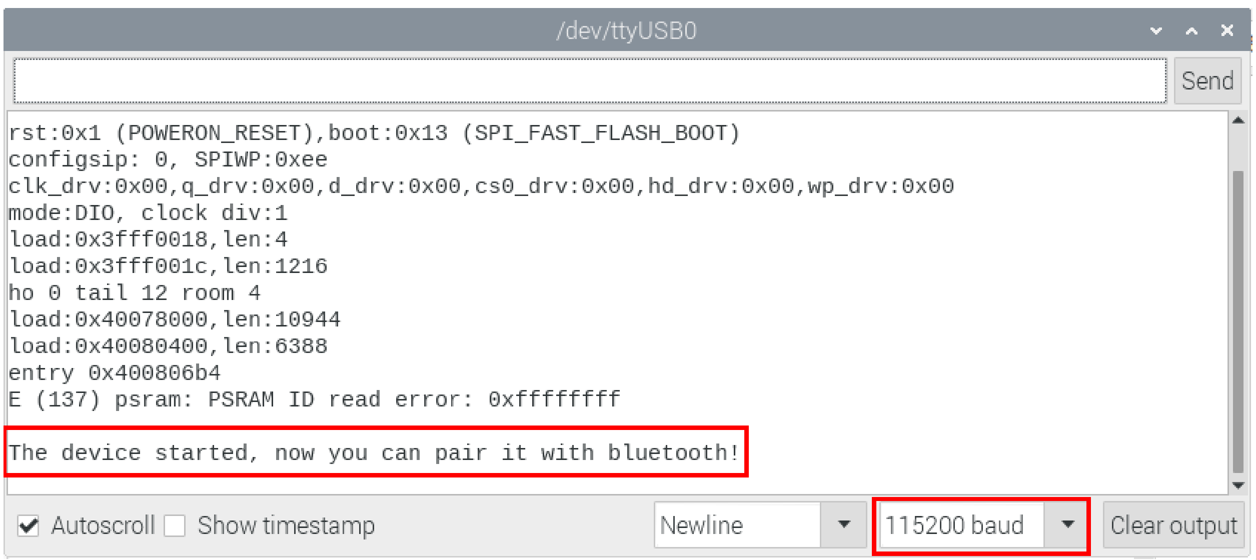
(continued from previous page)

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

5.Test Result

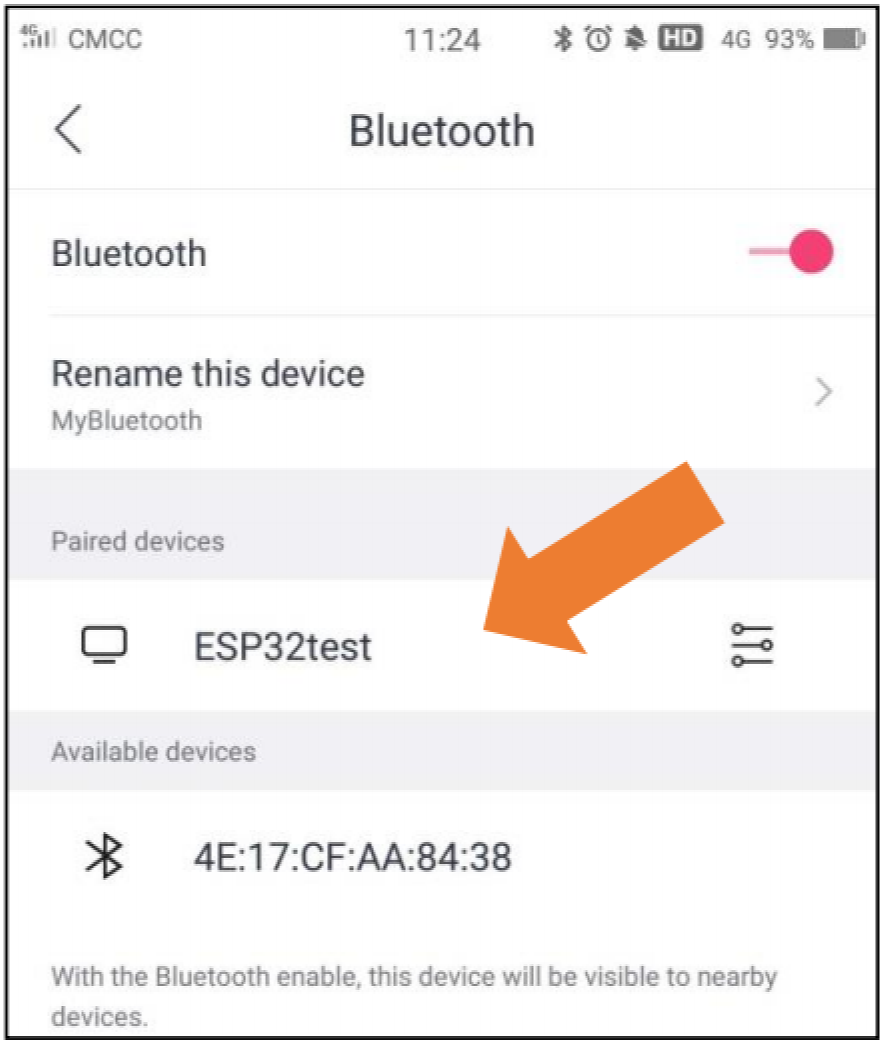
Compile and upload the code to the ESP32. After uploading successfullywe will use a USB cable to power on. Open the serial monitor and set the baud rate to **115200 then press the reset button first**. When you see the serial monitor prints out the character string as below, it indicates that the Bluetooth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



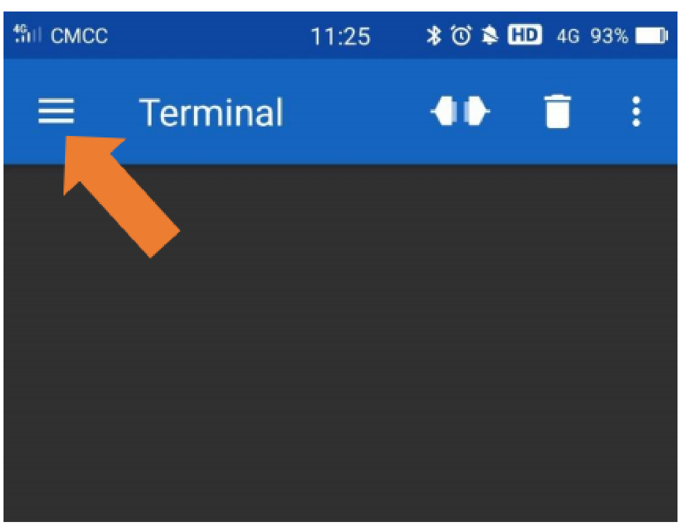


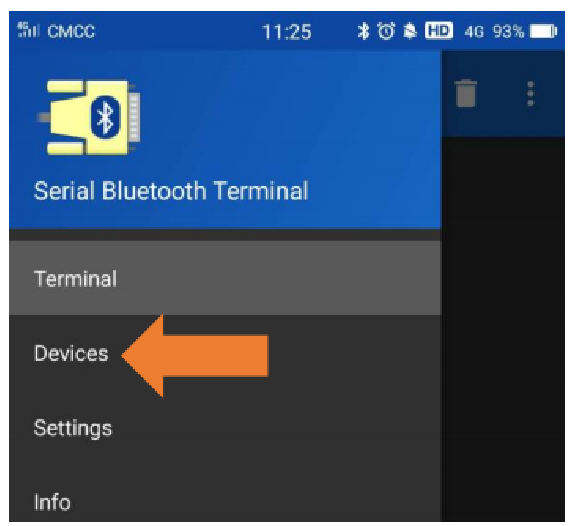
Make sure that the Bluetooth of your phone has been turned on and “Serial Bluetooth Terminal”has been installed.

Click“Search”search for the nearby Bluetooth and select to connect the“ESP32 test”.

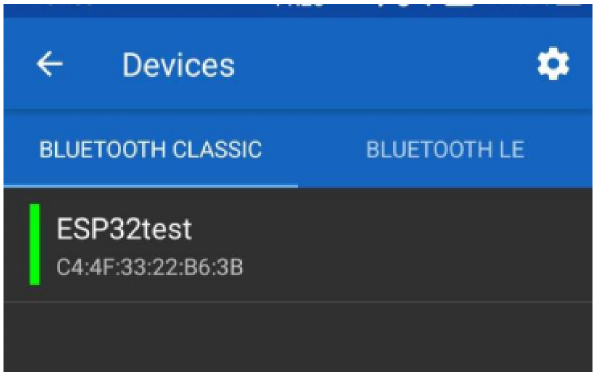


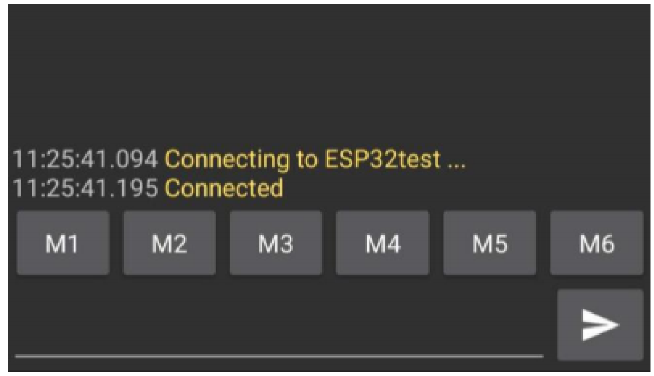
Turn on software APP, click the left of the terminal. Select “**Devices**” .





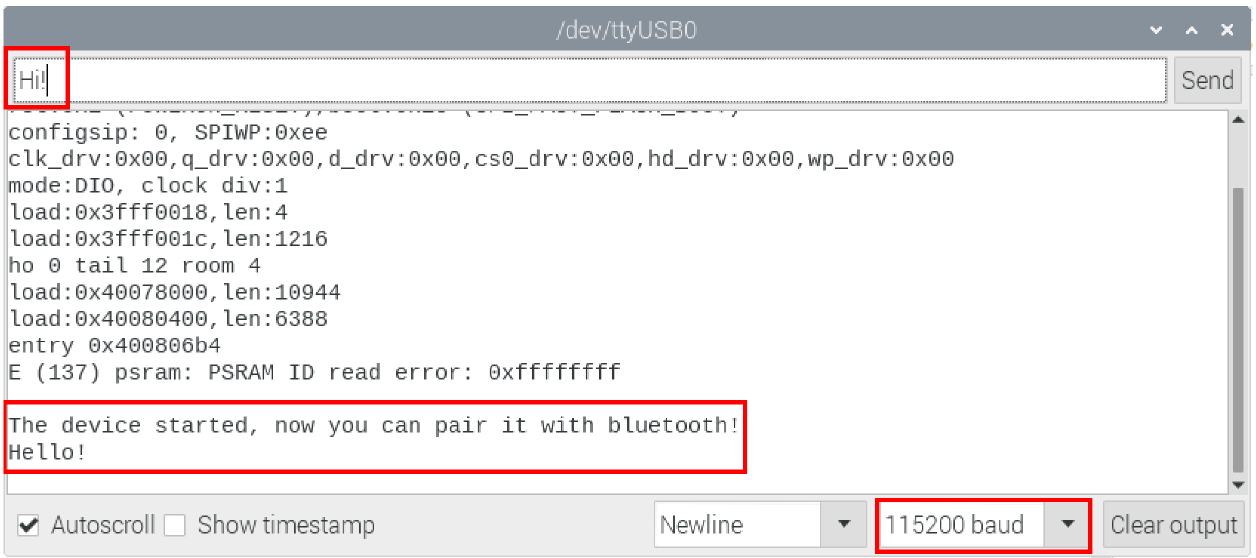
Select ESP32test in classic Bluetooth mode, and a successful connecting prompt will appear as shown below.

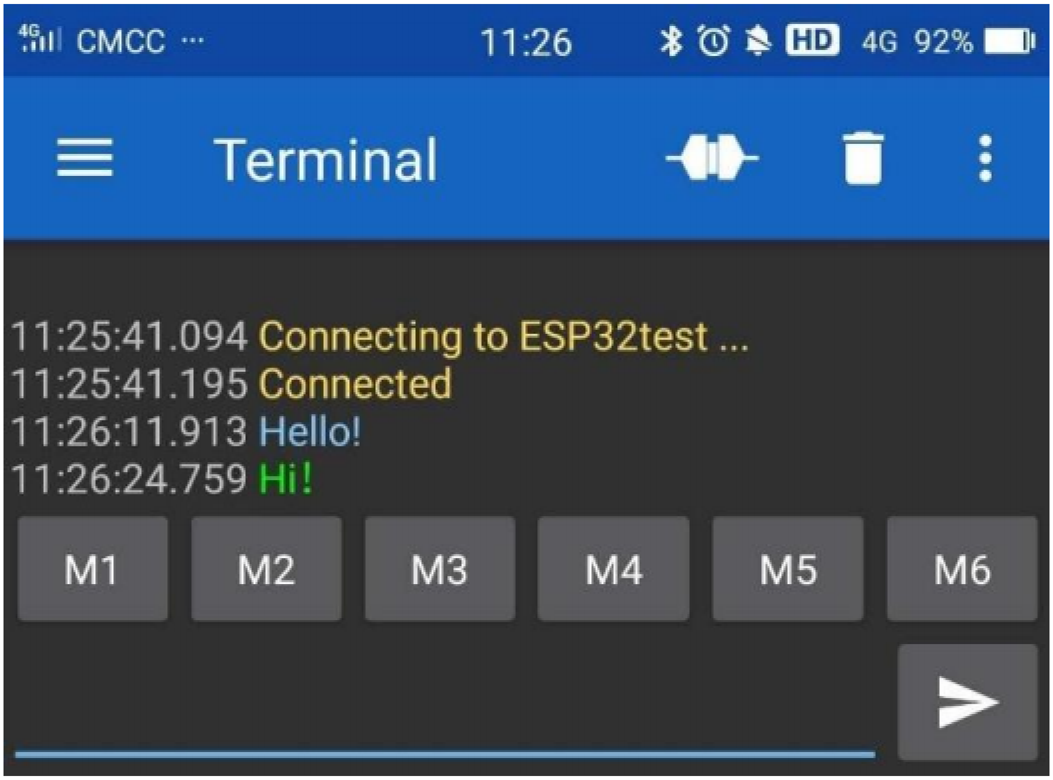




Data can be transferred between your phone and the Raspberry Pi via ESP32 now.

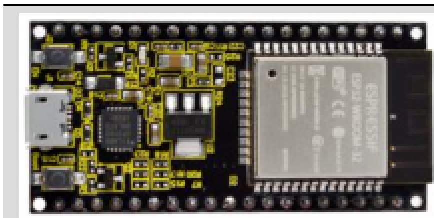
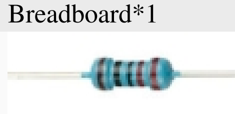
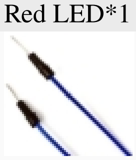
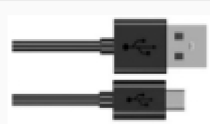
Send"Hello!", When the Raspberry Pi receives it, which will reply with "Hi!".



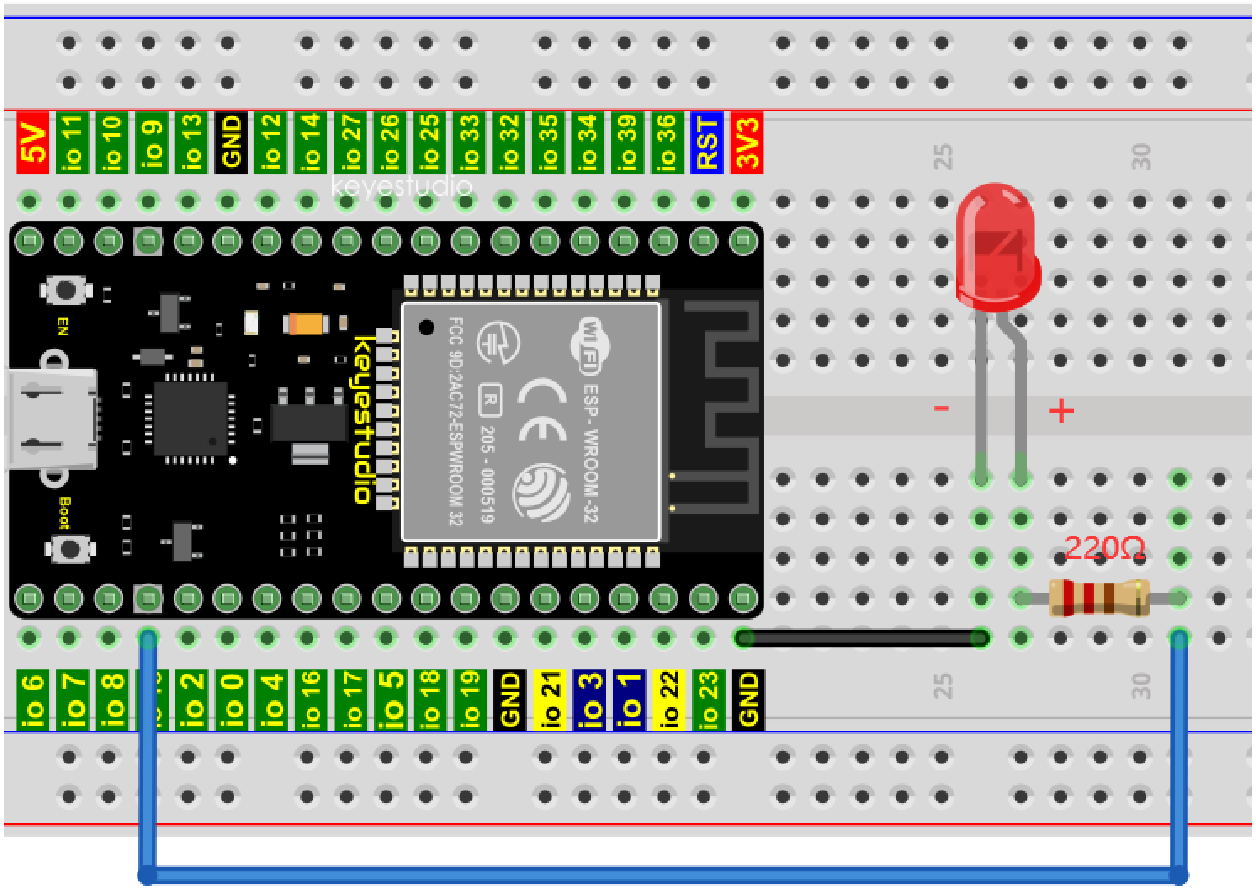


9.22.2 Project 22.2Bluetooth Control LED

1.Components

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

2.Wiring Diagram


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
EN
Boot
ESP-WROOM-32
205-000519
FCC 9D:2AC72-ESPWROOM 32
25
30
-
+
220Ω
io 6
io 7
io 8
io 2
io 0
io 4
io 16
io 17
io 5
io 18
io 19
GND
io 21
io 3
io 1
io 22
io 23
GND

3.Test Code


```
//**************************************************************************************
/*
 * 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!");
}

void loop() {
```

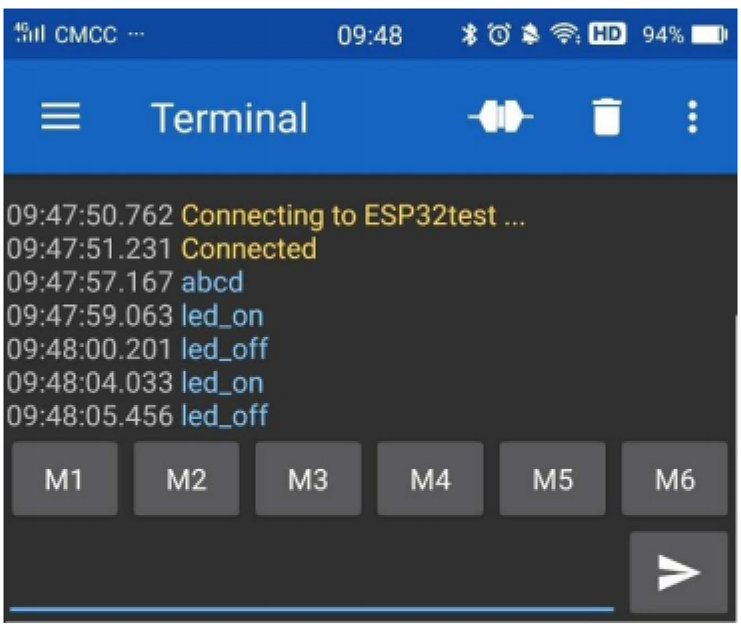
(continues on next page)

(continued from previous page)

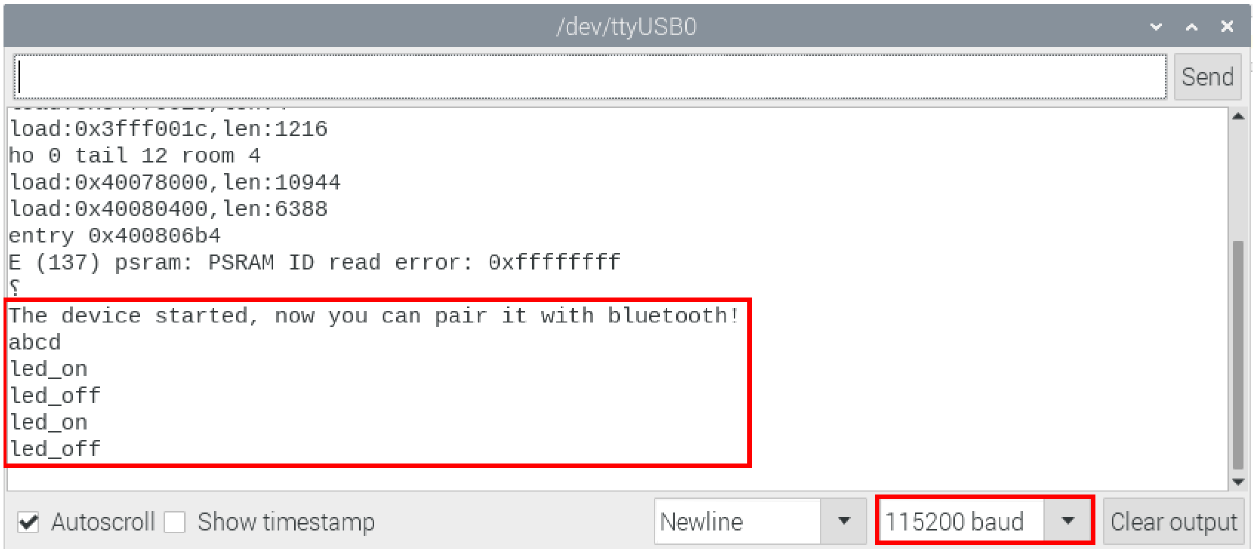
```
  while(SerialBT.available())
  {
    buffer[count] = SerialBT.read();
    count++;
  }
  if(count>0){
    Serial.print(buffer);
    if(strncmp(buffer,"led_on",6)==0){
      digitalWrite(LED,HIGH);
    }
    if(strncmp(buffer,"led_off",7)==0){
      digitalWrite(LED,LOW);
    }
    count=0;
    memset(buffer,0,20);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

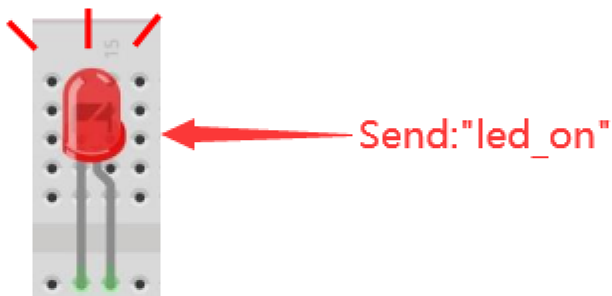
4.Test Result

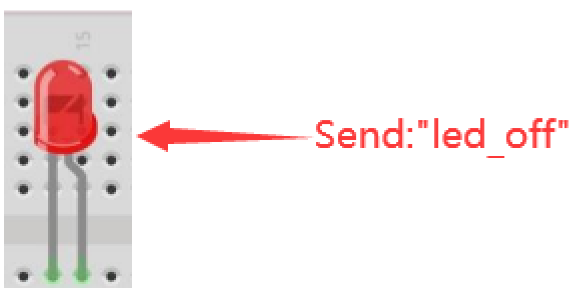
Compile and upload the code to the ESP32. The APP operation is the same as the project 22.1. To make the external LED on and off, simply change the sending content to "led_on" and "led_off". Moving the APP to send data:



The serial monitor will display as follows:







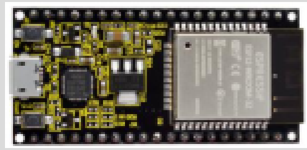
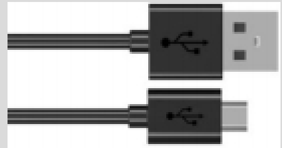
Note: If the "led-on 'or" led-off " is not sent, the status of the LED will not change. If the LED is on, it remains on when irrelevant content is received; Conversely, if the LED is off, it continues to be off when irrelevant content is received.

9.23 Project 23WiFi Station Mode

1.Introduction

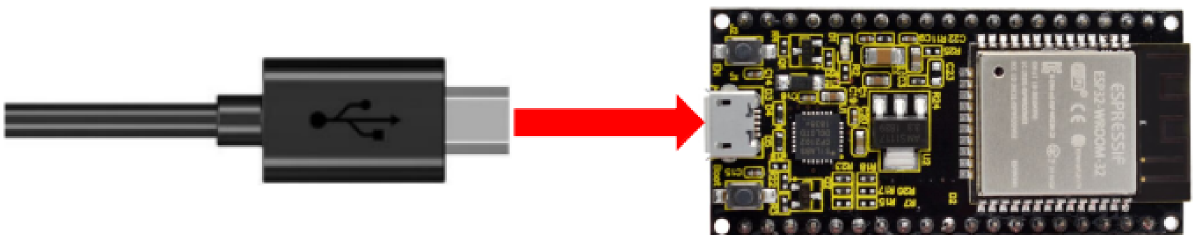
ESP32 has three different WiFi operating modes : Station modeAP mode and AP+Station mode. All WiFi programming projects must be configured with WiFi operating mode before using, otherwise WiFi cannot be used. In this project, we are going to learn the WiFi Station mode of the ESP32.

2.Components

	
ESP32*1	USB Cable*1

3.Wiring Diagram

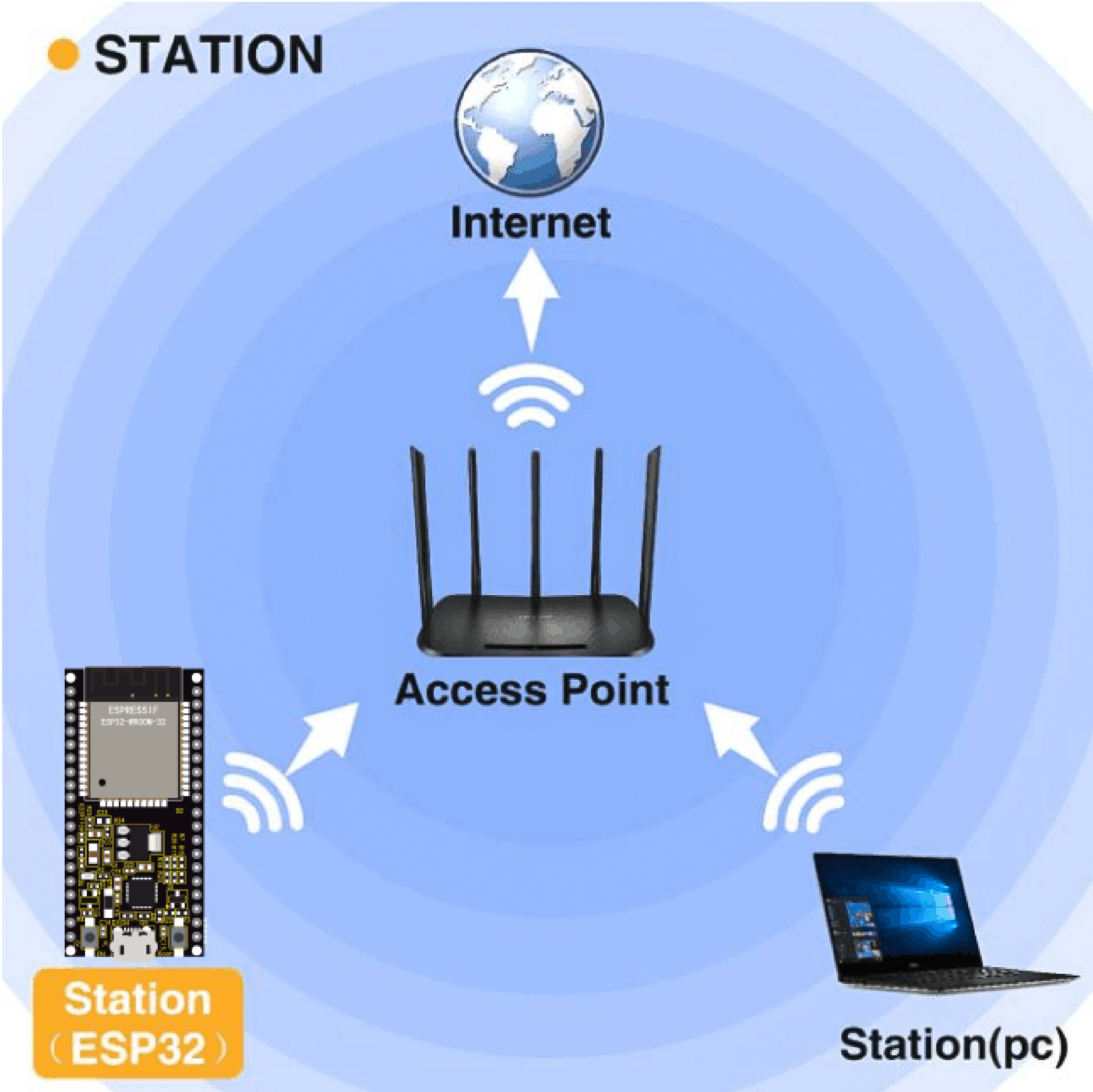
Plug the ESP32 to the USB port of the Raspberry Pi



4.Component Knowledge

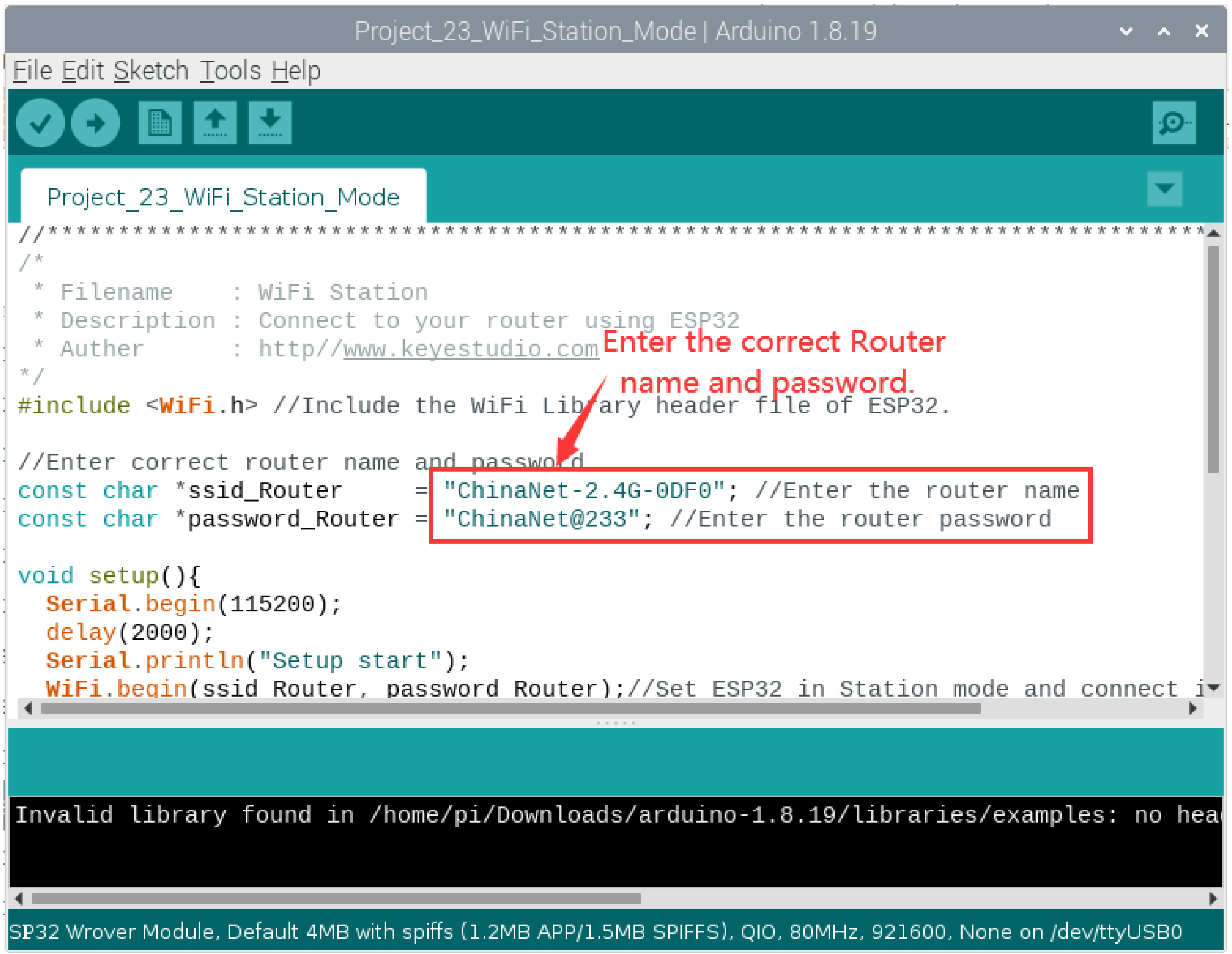
Station mode

When setting Station mode, the ESP32 is taken as a WiFi client. It can connect to the router network and communicate with other devices on the router via a WiFi connection. As shown in the figure below, the PC and the router have been connected. If the ESP32 wants to communicate with the PC, the PC and the router need to be connected.


STATION
Internet
Access Point
ESPRESSIF
Station
(ESP32)
Station(pc)

5.Test Code

Since WiFi names and passwords vary from place to place, thereby users need to enter the correct WiFi names and passwords in the box shown below before running the program code.

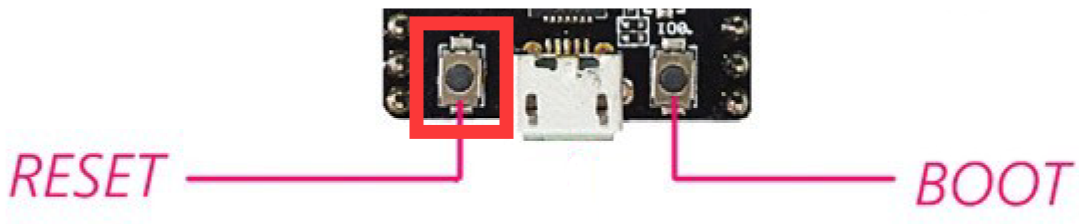



```
//**************************************************************************************
/*
 * Filename    : WiFi Station
 * Description : Connect to your router using ESP32
 * Auther      : http//www.keyestudio.com
*/
#include <WiFi.h> //Include the WiFi Library header file of ESP32.

//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)

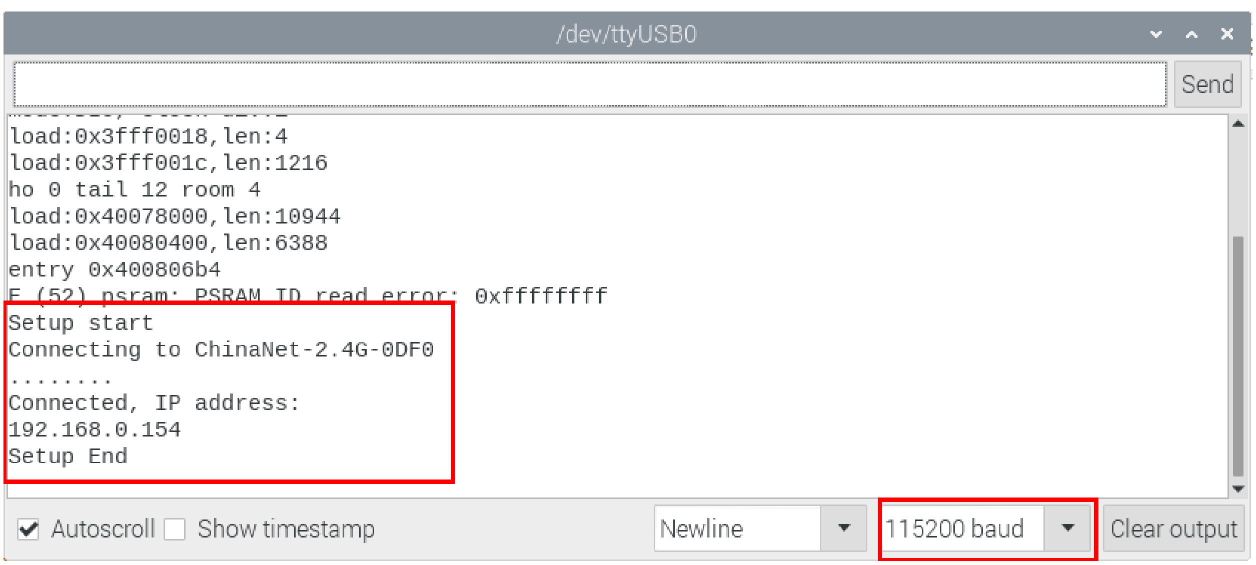
(continued from previous page)

```
    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() {
}
//*************************************************************************************
```



6.Test 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 then press the reset button first. When ESP32 successfully connects to"ssid_Router", serial monitor will print out the IP address, then monitor will display as follows: (If open the serial monitor and set the baud rate to 115200 and the information is not displayed, please press the RESET button of the ESP32).

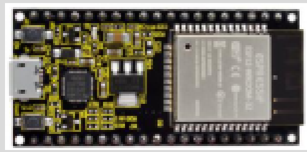
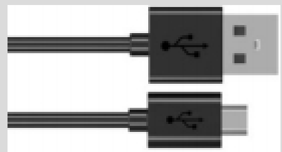


9.24 Project 24WiFi AP Mode

1.Introduction

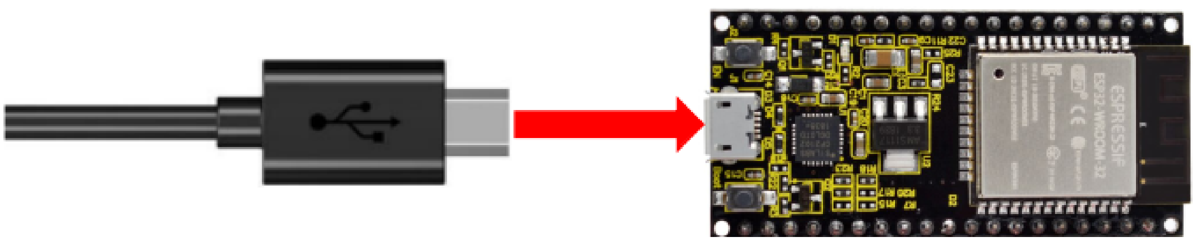
In this project, we are going to learn the WiFi AP mode of the ESP32.

2.Components

	
ESP32*1	USB Cable*1

3.Wiring Diagram

Plug the ESP32 mainboard to the USB port of the Raspberry Pi

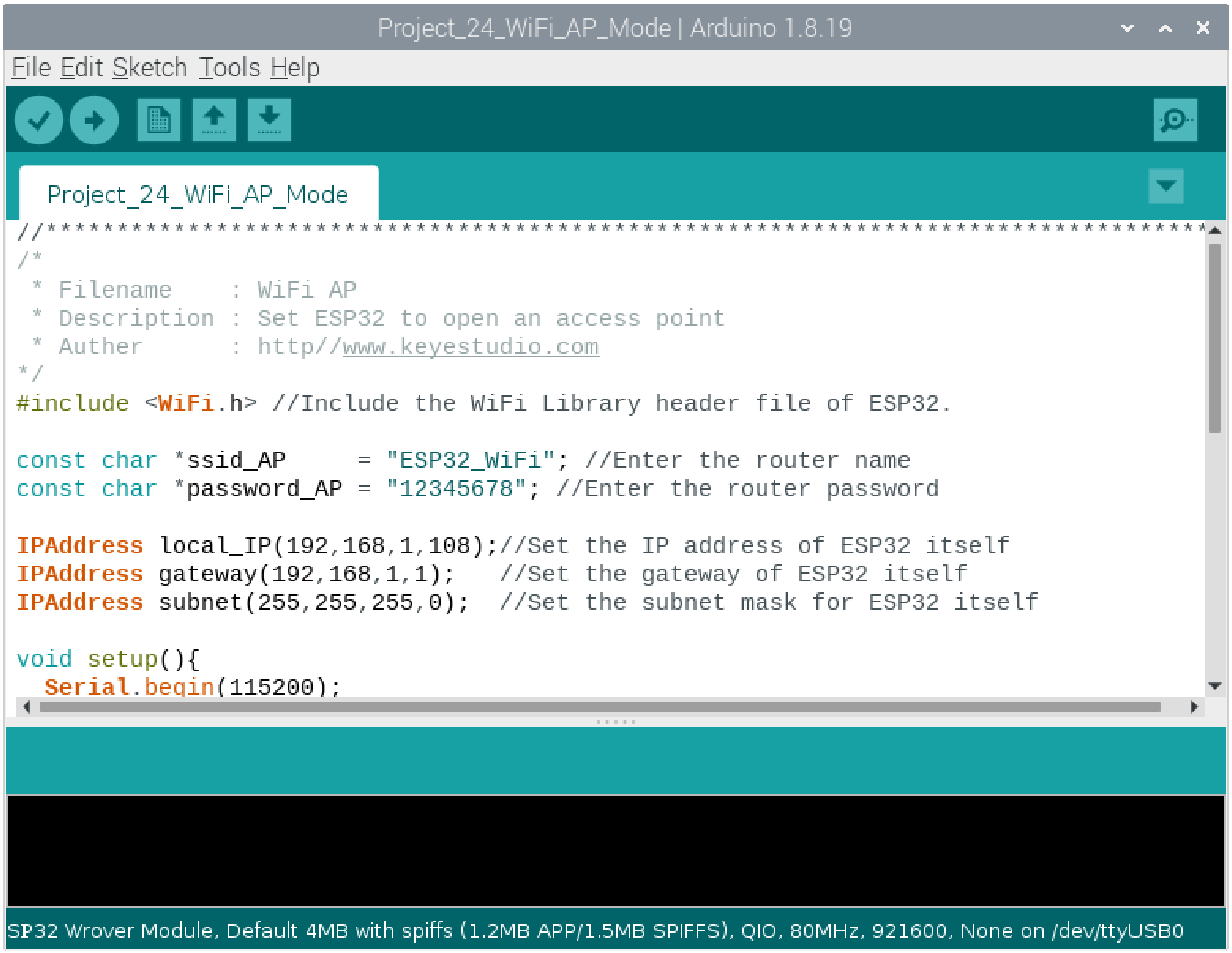


4.Component Knowledge

AP Mode:

When setting AP mode, a hotspot network will be created, waiting for other WiFi devices to connect. As shown below;

5.Test Code



Before running the code , you can make any changes to the ESP32 AP name and password in the box as shown below, but in a default circumstance, it doesn't need to modify.


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

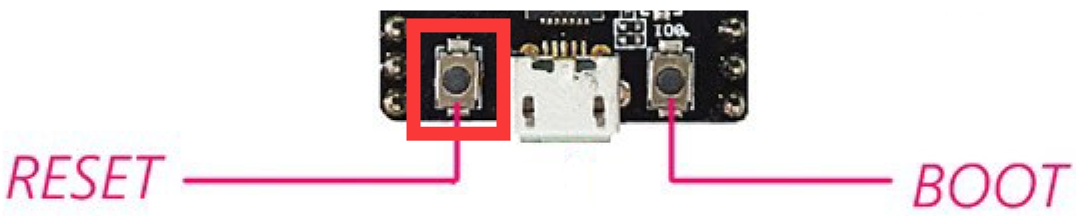
(continued from previous page)

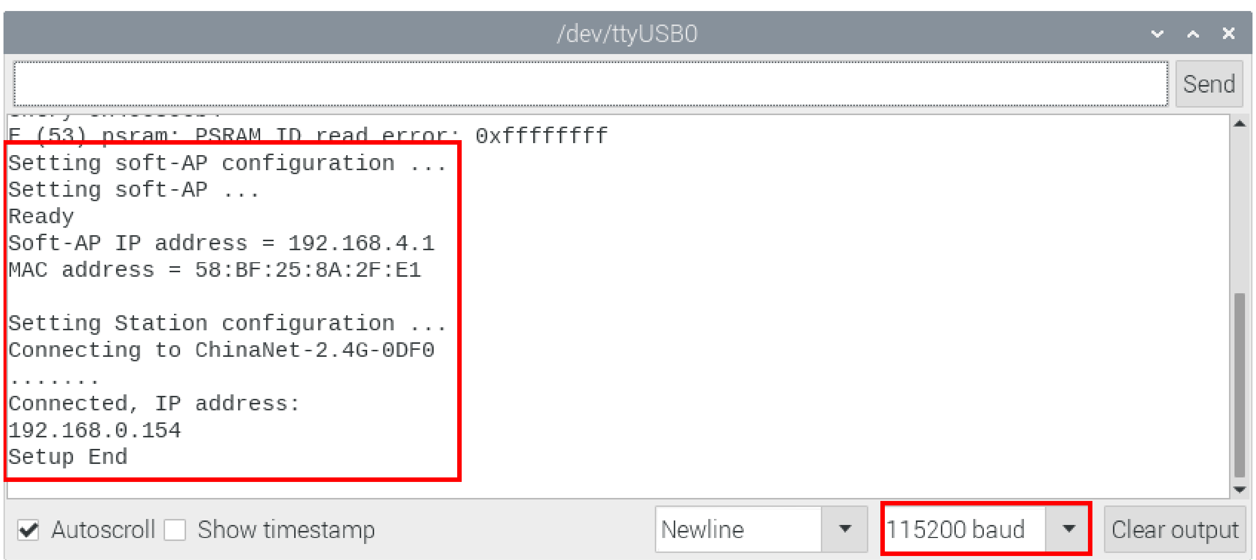
```
  Serial.println("Setting soft-AP ... ");
  boolean result = WiFi.softAP(ssid_AP, password_AP);
  if(result){
    Serial.println("Ready");
    Serial.println(String("Soft-AP IP address = ") + WiFi.softAPIP().toString());
    Serial.println(String("MAC address = ") + WiFi.softAPmacAddress().c_str());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

void loop() {
}
//*************************************************************************************
```

6.Test Result

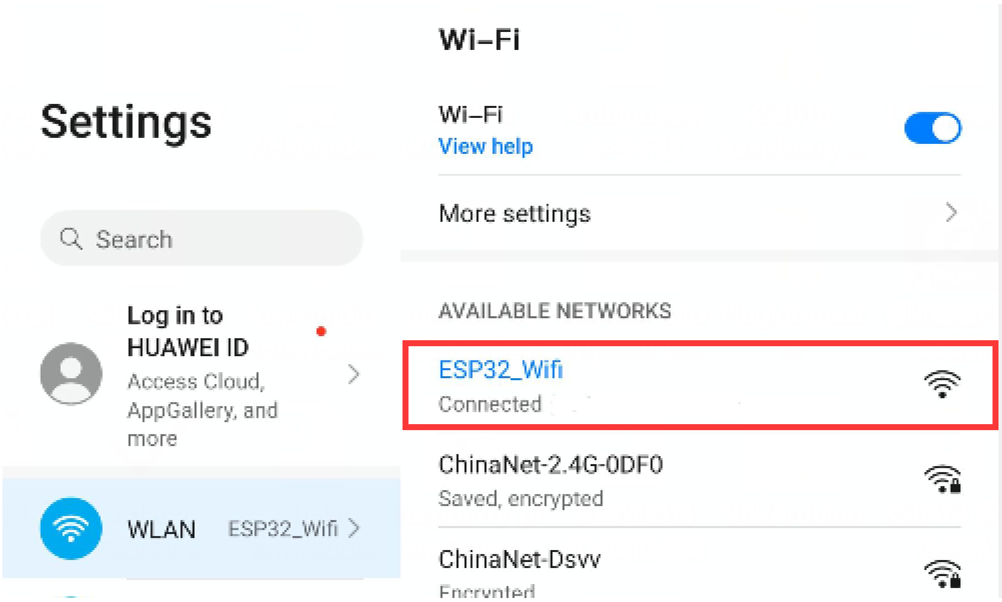
Enter the ESP32 AP name and password correctly, compile and upload the code to ESP32, open the serial monitor and set the baud rate to **115200 and press the reset button first**, then monitor will display as follows:

(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 printed information of the serial monitor, turn on the WiFi scanning function of the mobile phone, you can see the ssid_AP on ESP32, which is dubbed “ESP32_Wifi” in this code. You can connect to it either by typing the password “12345678” or by modifying the code to change its AP name and password.

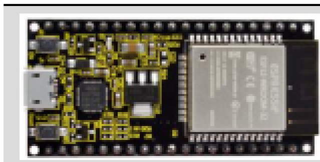
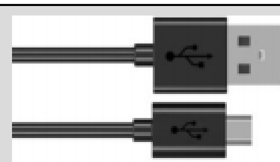


9.25 Project 25WiFi Station+AP Mode

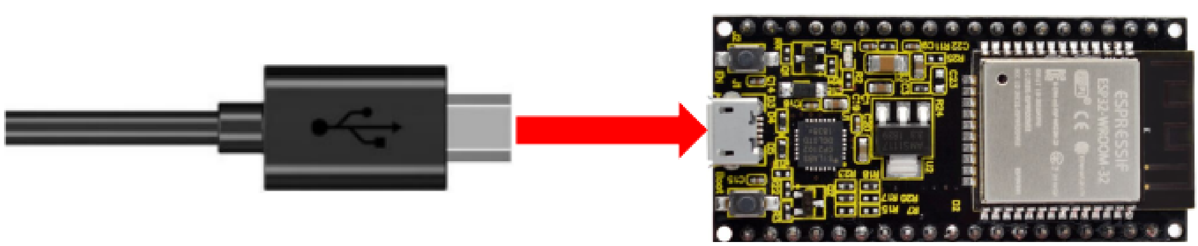
1.Introduction

In this project, we are going to learn the AP+Station mode of the ESP32.

2.Components

	
ESP32*1	USB Cable*1

3.Wiring Diagram



Plug the ESP32 mainboard to the USB port of the Raspberry Pi

4.Component Knowledge

AP+Station mode:

In addition to the AP mode and the Station mode, **AP+Station mode** can be used at the same time. Turn on the Station mode of the ESP32, connect it to the router network, and it can communicate with the Internet through the router. Then turn on the AP mode to create a hotspot network. Other WiFi devices can be connected to the router network or the hotspot network to communicate with the ESP32.

5.Test Code

Before running the code, you need to modify the ssid_Router, password_Routerssid_AP and password_AP, as shown in the box below:


```
//**************************************************************************
/*
 * 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");
```

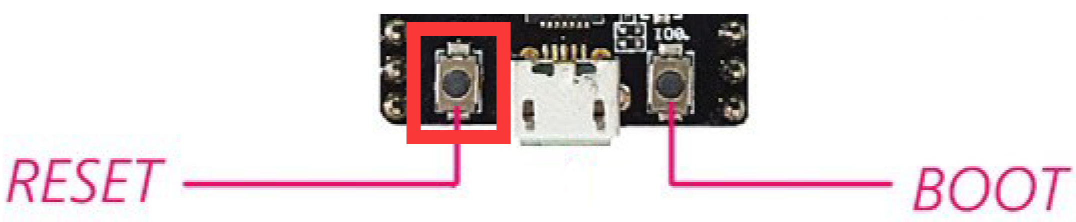
(continues on next page)

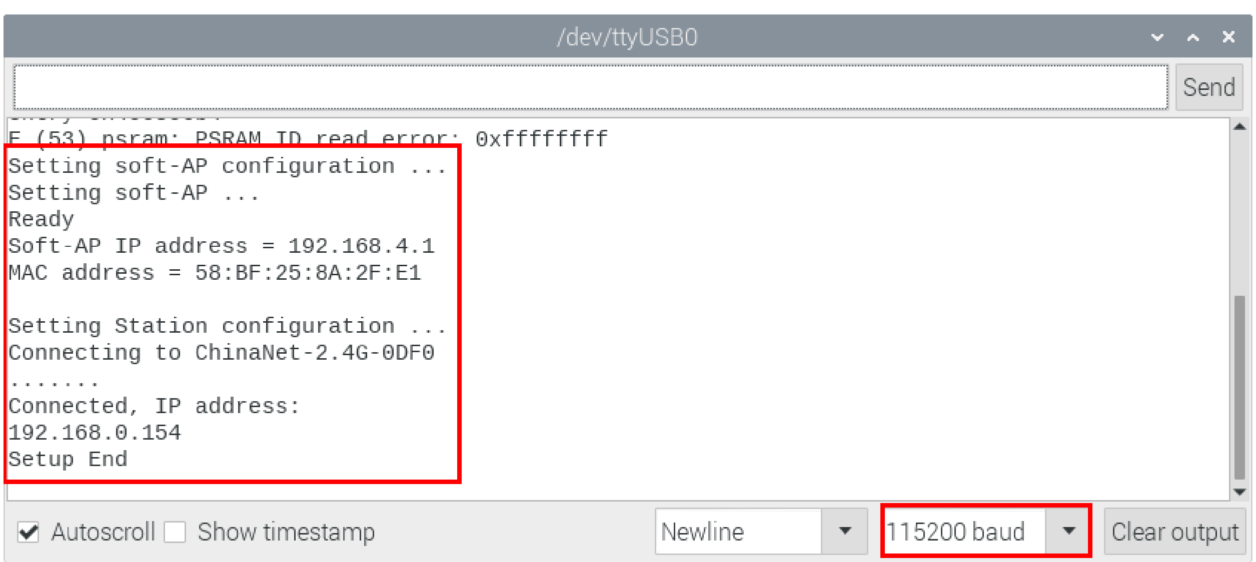
(continued from previous page)

```
    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() {
}
//*************************************************************************************
```

6.Test Result

Ensure that the code in the program has been modified correctly, 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 and press the reset button, then monitor will display as follows: (If open the serial monitor and set the baud rate to 115200 and the information is not displayed, please press the RESET button of the ESP32)





Open the WiFi scanning function of the mobile phone, you can see the ssid_AP.