Important Information

Thank you for choosing Freenove products!

Getting Started

First, please read the **Start Here.pdf** document in the unzipped folder you created.
If you have not yet downloaded the zip file, associated with this kit, please do so now and unzip it.

Get Support and Offer Input

Freenove provides free and responsive product and technical support, including but not limited to:

- Product quality issues
- Product use and build issues
- Questions regarding the technology employed in our products for learning and education
- Your input and opinions are always welcome
- We also encourage your ideas and suggestions for new products and product improvements

For any of the above, you may send us an email to:

support@freenove.com

Safety and Precautions

Please follow the following safety precautions when using or storing this product:

- Keep this product out of the reach of children under 6 years old.
- This product should be **used only when there is adult supervision present** as young children lack necessary judgment regarding safety and the consequences of product misuse.
- This product contains small parts and parts, which are sharp. This product contains electrically conductive parts. **Use caution with electrically conductive parts near or around power supplies, batteries and powered (live) circuits.**
- When the product is turned ON, activated or tested, some parts will move or rotate. **To avoid injuries to hands and fingers keep them away from any moving parts!**
- It is possible that an improperly connected or shorted circuit may cause overheating. **Should this happen, immediately disconnect the power supply or remove the batteries and do not touch anything until it cools down!** When everything is safe and cool, review the product tutorial to identify the cause.
- Only operate the product in accordance with the instructions and guidelines of this tutorial, otherwise parts may be damaged or you could be injured.
- Store the product in a cool dry place and avoid exposing the product to direct sunlight.
- After use, always turn the power OFF and remove or unplug the batteries before storing.

About Freenove

Freenove provides open source electronic products and services worldwide.

Freenove is committed to assist customers in their education of robotics, programming and electronic circuits so that they may transform their creative ideas into prototypes and new and innovative products. To this end, our services include but are not limited to:

- Educational and Entertaining Project Kits for Robots, Smart Cars and Drones
- Educational Kits to Learn Robotic Software Systems for Arduino, Raspberry Pi and micro: bit
- Electronic Component Assortments, Electronic Modules and Specialized Tools
- **Product Development and Customization Services**

You can find more about Freenove and get our latest news and updates through our website:

http://www.freenove.com

sale@freenove.com

Copyright

FREENOVE™

Contents

Important Information 1
Contents 1
Preface 4
ESP32-S3 WROOM 5
Extension board of the ESP32-S3 WROOM 7
CH343 (Importance) 8
Programming Software 19
Environment Configuration 22
Notes for GPIO 25
Chapter 0 LED 28
Project 0.1 Blink 28
Chapter 1 LED 37
Project 1.1 Blink 37
Chapter 2 Button & LED 44
Project 2.1 Button & LED 44
Project 2.2 MINI table lamp 49
Chapter 3 LED Bar 52
Project 3.1 Flowing Light 52
Chapter 4 Analog & PWM 57
Project 4.1 Breathing LED 57
Project 4.2 Meteor Flowing Light 63
Chapter 5 RGB LED 68
Project 5.1 Random Color Light 68
Project 5.2 Gradient Color Light 73
Chapter 6 Buzzer 75
Project 6.1 Doorbell 75
Project 6.2 Alertor 81
Chapter 7 Serial Communication 84
Project 7.1 Serial Print 84
Project 7.2 Serial Read and Write 88

Chapter 8 AD Converter 90
Project 8.1 Read the Voltage of Potentiometer 90
Chapter 9 Touch Sensor 97
Project 9.1 Read Touch Sensor 97
Project 9.2 Touch Lamp 102
Chapter 10 Potentiometer & LED 107
Project 10.1 Soft Light 107
Project 10.2 Soft Colorful Light 110
Chapter 11 Photoresistor & LED 114
Project 11.1 NightLamp 114
Chapter 12 Thermistor 119
Project 12.1 Thermometer 119
Chapter 13 Joystick 124
Project 13.1 Joystick 124
Chapter 14 74HC595 & LED Bar Graph 129
Project 14.1 Flowing Water Light 129
Chapter 15 74HC595 & 7-Segment Display 135
Project 15.1 7-Segment Display. 135
Chapter 16 Relay & Motor 142
Project 16.1 Control Motor with Potentiometer 142
Chapter 17 Servo 150
Project 17.1 Servo Sweep 150
Project 17.2 Servo Knop 156
Chapter 18 LCD1602 160
Project 18.1 LCD1602 160
Chapter 19 Ultrasonic Ranging 168
Project 19.1 Ultrasonic Ranging 168
Project 19.2 Ultrasonic Ranging 174
Chapter 20 Bluetooth 178
Project 20.1 Bluetooth Low Energy Data Passthrough 178
Project 20.2 Bluetooth Control LED 190

Chapter 21 Read and Write the SDcard 198
Project 21.1 SDMMC Test 198
Chapter 22 Play SD card music 210
Project 22.1 SDMMC Music 210
Chapter 23 WiFi Working Modes 217
Project 23.1 Station mode 217
Project 23.2 AP mode 222
Project 23.3 AP+Station mode 227
Chapter 24 TCP/IP 231
Project 24.1 As Client 231
Project 24.2 As Server 243
Chapter 25 Camera Web Server 249
Project 25.1 Camera Web Server 249
Project 25.2 Video Web Server 258
Project 25.3 Camera and SDcard 264
Chapter 26 Camera Tcp Server 273
Project 26.1 Camera Tcp Server 273
What's next? 291
End of the Tutorial 291

Preface

ESP32-S3 is a micro control unit with integrated Wi-Fi launched by Espressif, which features strong properties and integrates rich peripherals. It can be designed and studied as an ordinary Single Chip Micyoco(SCM) chip, or connected to the Internet and used as an Internet of Things device.

ESP32-S3 can be developed using the Arduino platform, which will definitely make it easier for people who have learned arduino to master. Moreover, the code of ESP32-S3 is completely open-source, so beginners can quickly learn how to develop and design IOT smart household products including smart curtains, fans, lamps and clocks.

Generally, ESP32-S3 projects consist of code and circuits. Don't worry even if you've never learned code and circuits, because we will gradually introduce the basic knowledge of C programming language and electronic circuits, from easy to difficult. Our products contain all the electronic components and modules needed to complete these projects. It's especially suitable for beginners.

We divide each project into four parts, namely Component List, Component Knowledge, Circuit and Code. Component List helps you to prepare material for the experiment more quickly. Component Knowledge allows you to quickly understand new electronic modules or components, while Circuit helps you understand the operating principle of the circuit. And Code allows you to easily master the use of SEP32 and accessory kit. After finishing all the projects in this tutorial, you can also use these components and modules to make products such as smart household, smart cars and robots to transform your creative ideas into prototypes and new and innovative products.

In addition, if you have any difficulties or questions with this tutorial or toolkit, feel free to ask for our quick and free technical support through **support@freenove.com**

ESP32-S3 WROOM

ESP32-S3-WROOM-1 has launched a total of two antenna packages, PCB on-board antenna and IPEX antenna respectively. The PCB on-board antenna is an integrated antenna in the chip module itself, so it is convenient to carry and design. The IPEX antenna is a metal antenna derived from the integrated antenna of the chip module itself, which is used to enhance the signal of the module.

PCB on-board antenna	IPEX antenna
	

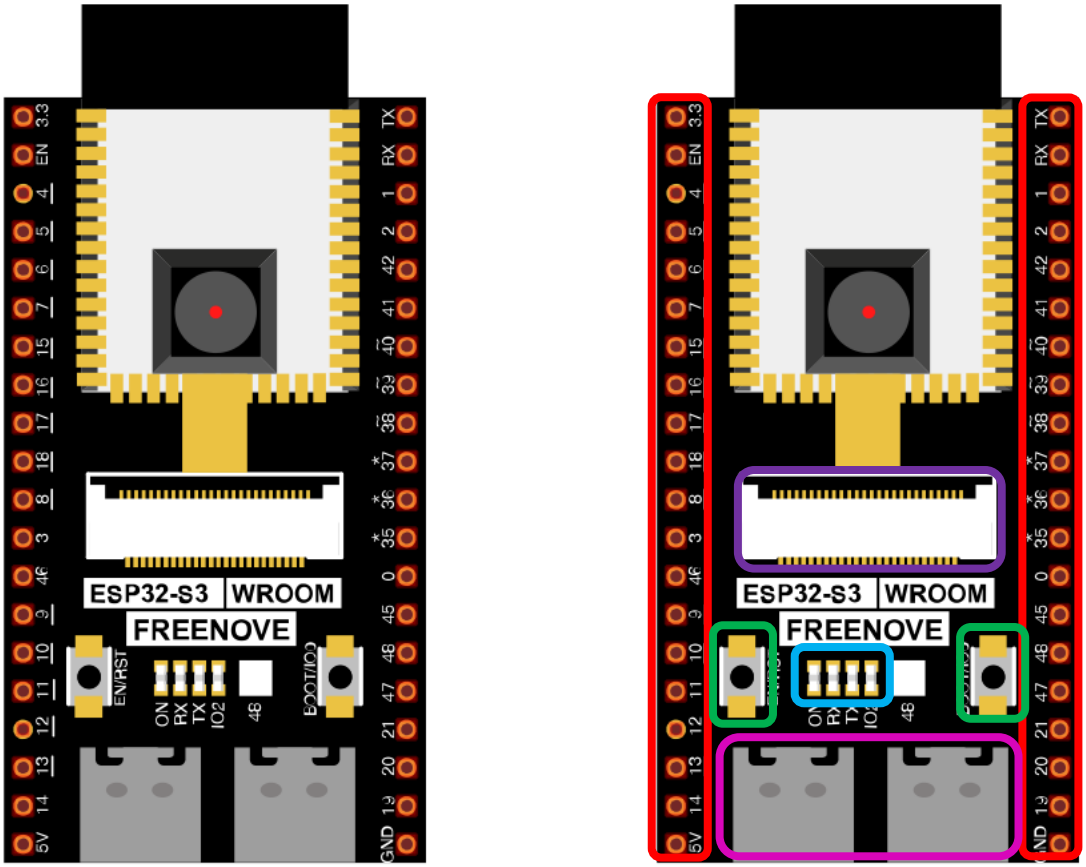
In this tutorial, the ESP32-S3 WROOM is designed based on the PCB on-board antenna-packaged ESP32-S3-WROOM-1 module.

Currently, we have two development board options available: the N8R8(**8MB Flash + 8MB PSRAM**) version with a black PCB and the N16R8(**16MB Flash + 8MB PSRAM**) version with a red PCB.The only difference between these two boards lies in their built-in Flash storage capacity, while all other features, functionalities, and peripheral circuits remain identical.

(Note: The N16R8 version is currently sold as a single-board kit only.)

ESP32-S3 WROOM N16R8	ESP32-S3 WROOM N8R8
	

The hardware interfaces of ESP32-S3 WROOM are distributed as follows:



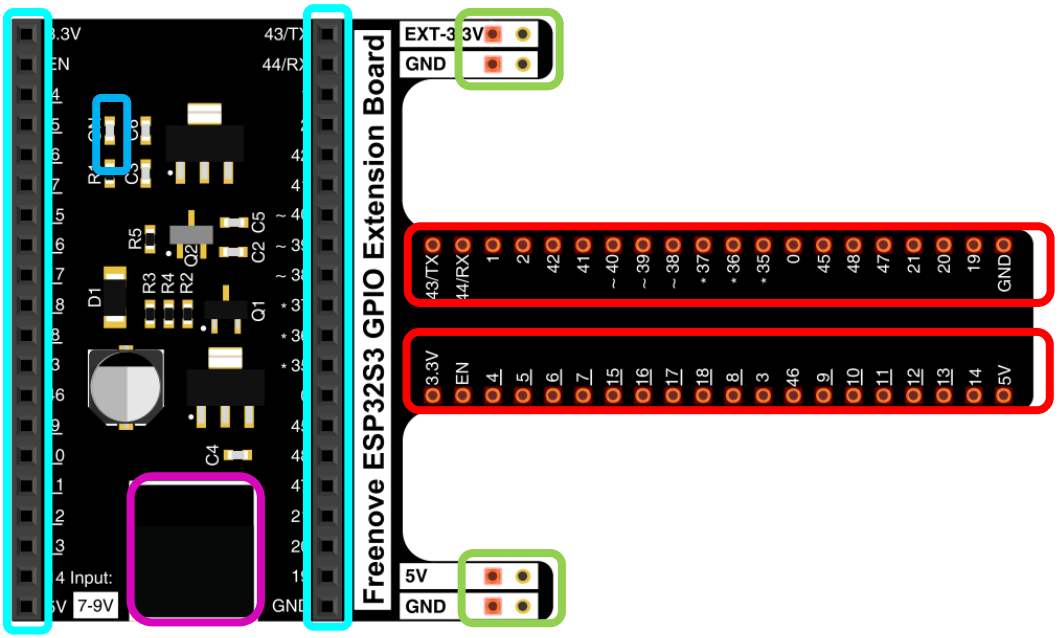
Compare the left and right images. We've boxed off the resources on the ESP32-S3 WROOM in different colors to facilitate your understanding of the ESP32-S3 WROOM.

Box color	Corresponding resources introduction
	GPIO pin
	LED indicator
	Camera interface
	Reset button, Boot mode selection button
	USB port

For more information, please visit: https://www.espressif.com.cn/sites/default/files/documentation/esp32-s3-wroom-1_wroom-1u_datasheet_en.pdf.

Extension board of the ESP32-S3 WROOM

And we also design an extension board, so that you can use the ESP32-S3 more easily in accordance with the circuit diagram provided. The followings are their photos.
The hardware interfaces of ESP32-S3 WROOM are distributed as follows:



We've boxed off the resources on the ESP32-S3 WROOM in different colors to facilitate your understanding of the ESP32-S3 WROOM.

Box color	Corresponding resources introduction
	GPIO pin
	LED indicator
	GPIO interface of development board
	power supplied by the extension board
	External power supply

In ESP32-S3, GPIO is an interface to control peripheral circuit.

In the following projects, we only use USB cable to power ESP32-S3 WROOM by default.

In the whole tutorial, we don't use T extension to power ESP32-S3 WROOM. So 5V and 3.3V (including EXT 3.3V) on the extension board are provided by ESP32-S3 WROOM.

We can also use DC jack of extension board to power ESP32-S3 WROOM. In this way, 5v and EXT 3.3v on extension board are provided by external power resource.

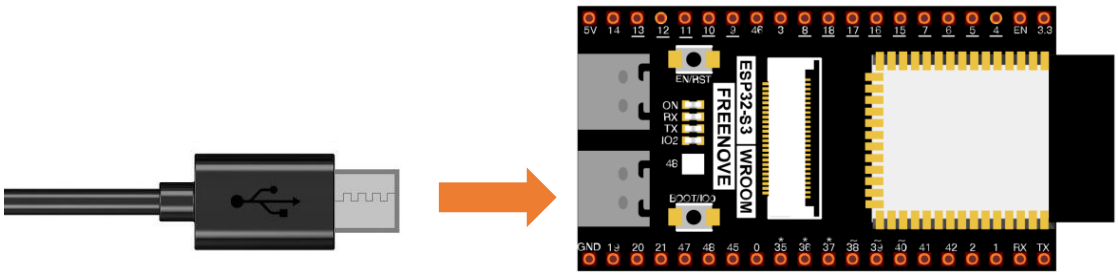
CH343 (Importance)

ESP32-S3 WROOM uses CH343 to download codes. So before using it, we need to install CH343 driver in our computers.

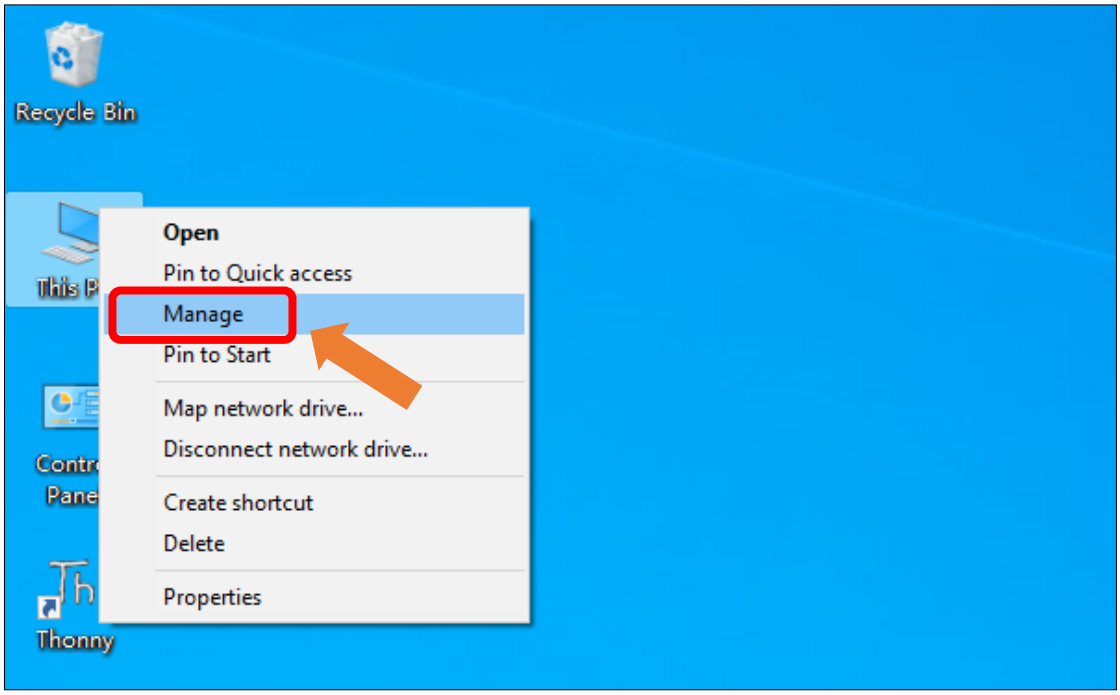
Windows

Check whether CH343 has been installed

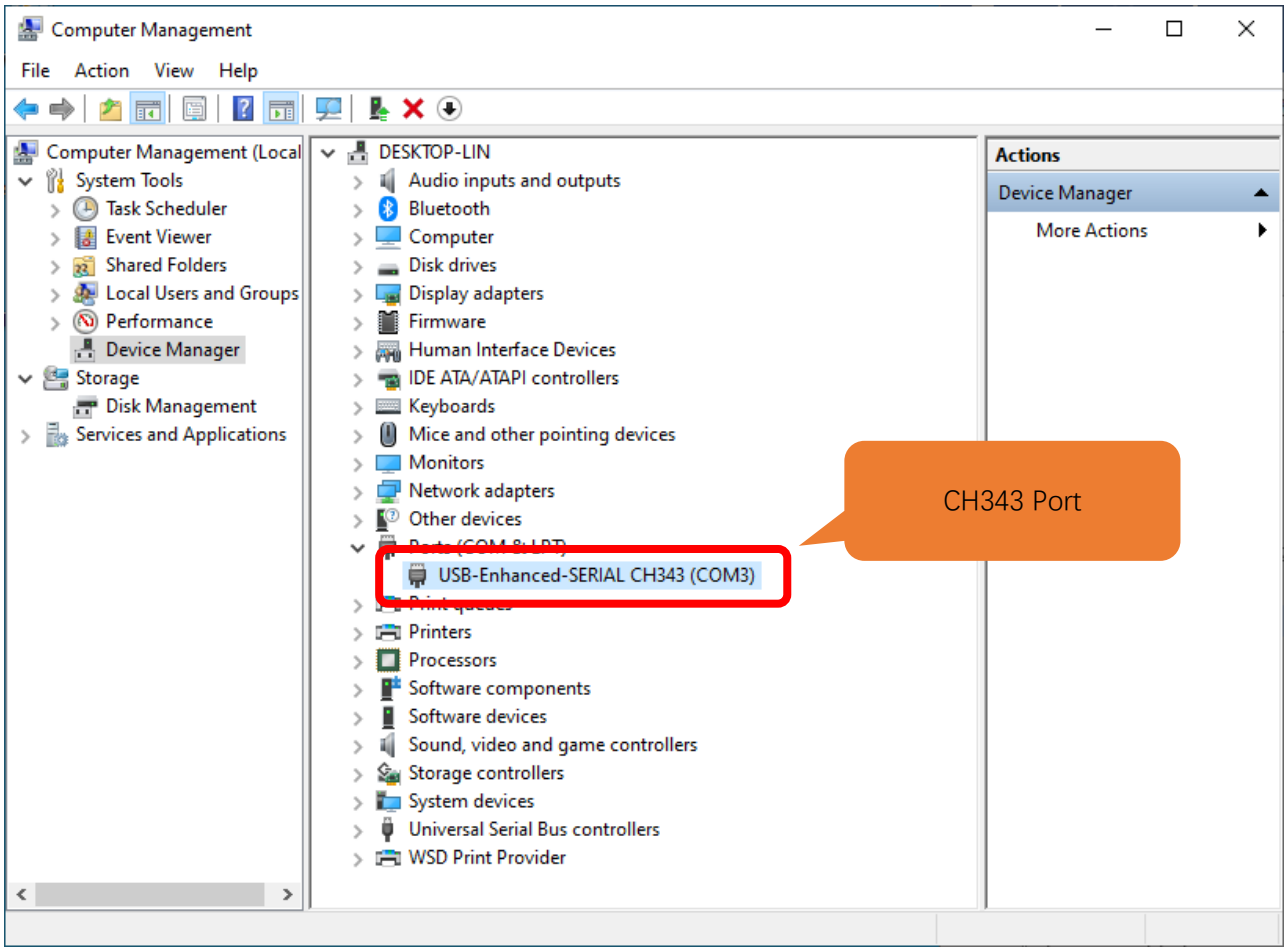
1. Connect your computer and ESP32-S3 WROOM with a USB cable.



2. Turn to the main interface of your computer, select “This PC” and right-click to select “Manage”.

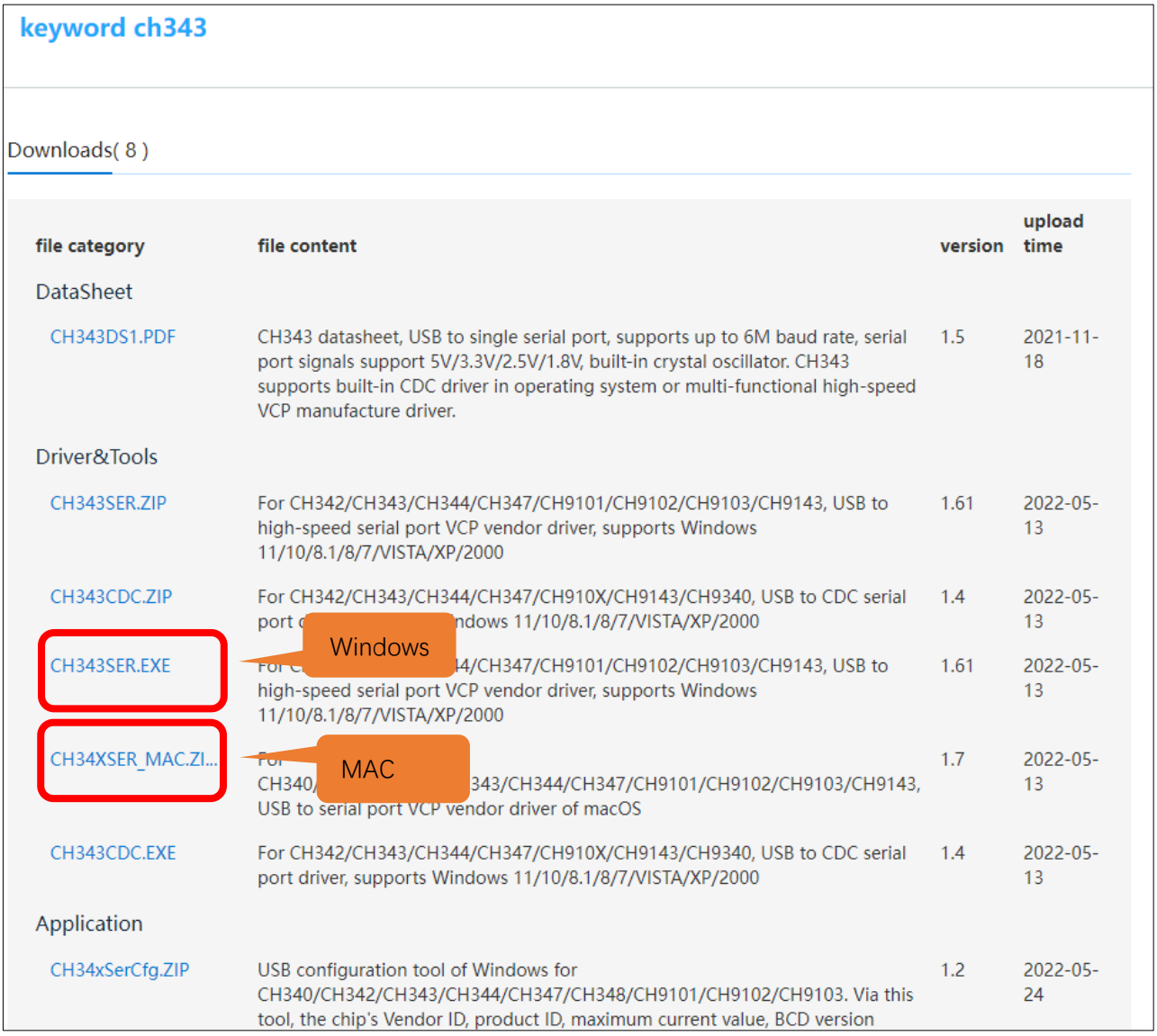


3. Click "Device Manager". If your computer has installed CH343, you can see"USB-Enhances-SERIAL CH343 (COMx)". And you can click here to move to the next step.

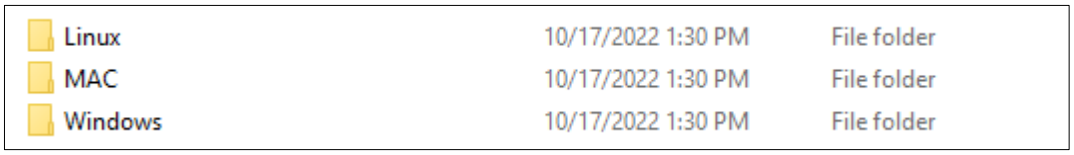


Installing CH343

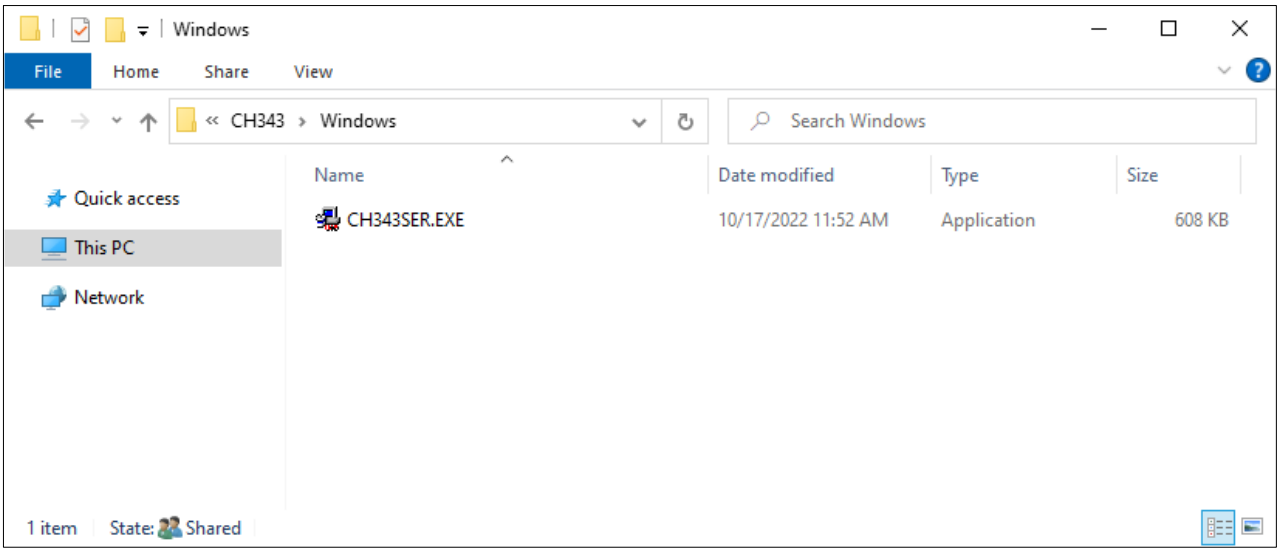
1. First, download CH343 driver, click http://www.wch-ic.com/search?t=all&q=ch343 to download the appropriate one based on your operating system.



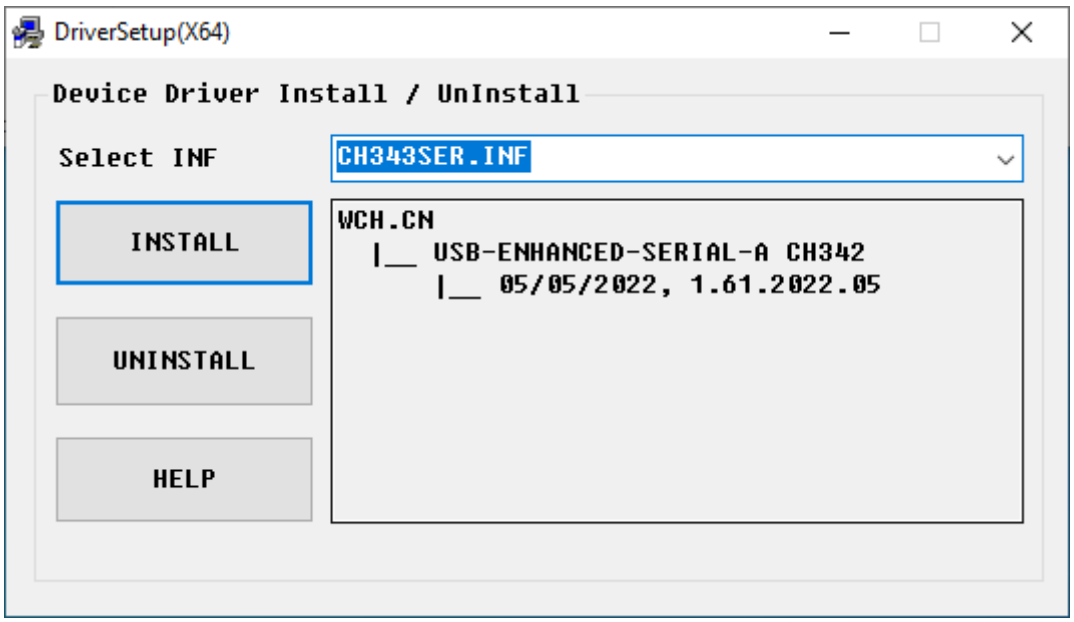
If you would not like to download the installation package, you can open **"Freenove_Super_Starter_Kit_for_ESP32_S3/CH343"**, we have prepared the installation package.



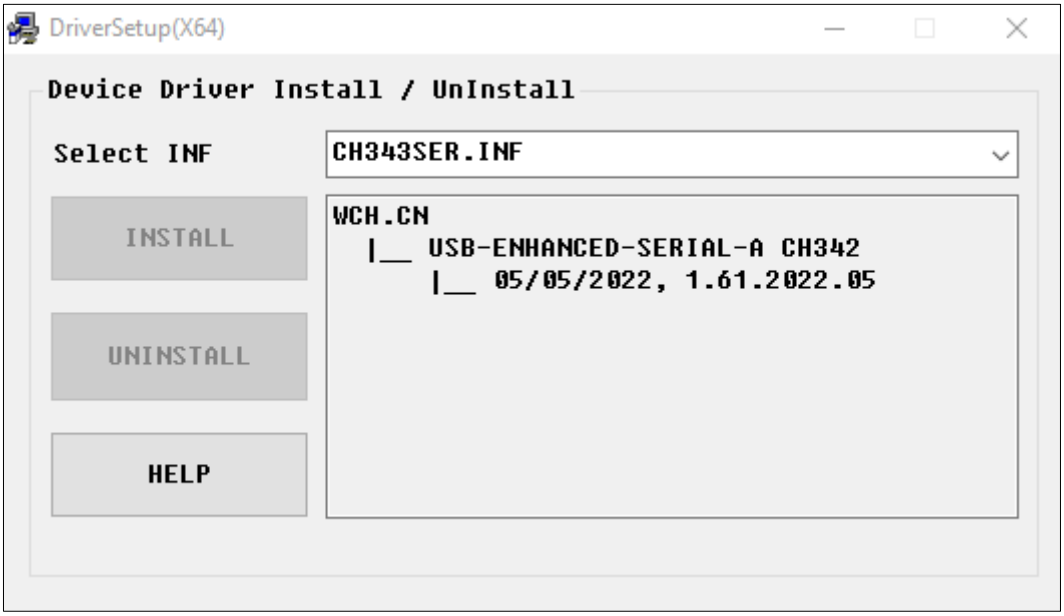
2. Open the folder "**Freenove_Super_Starter_Kit_for_ESP32_S3/CH343/Windows/**"



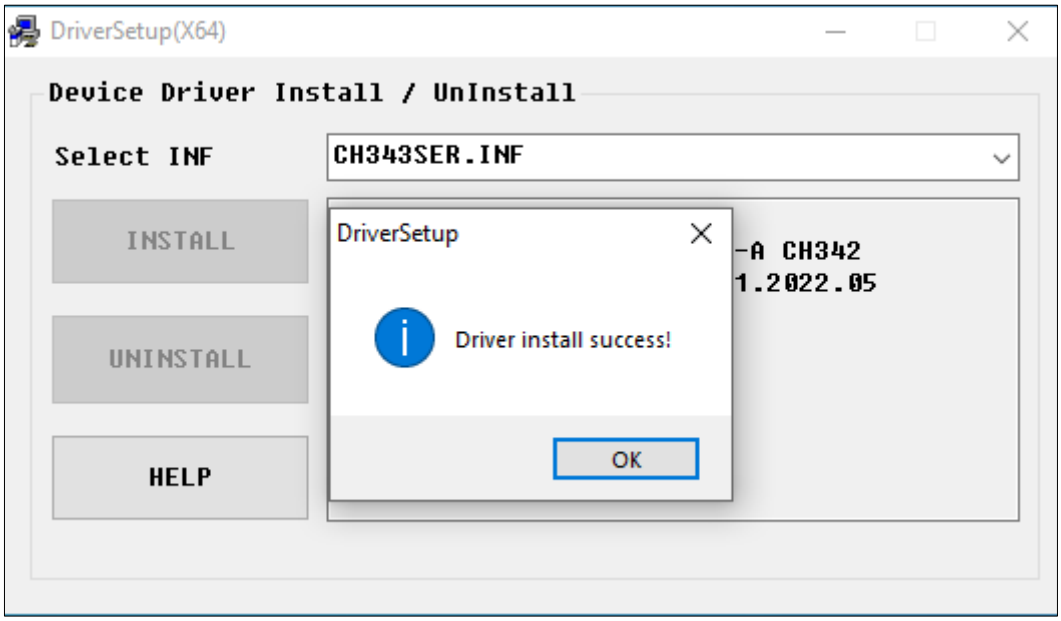
3. Double click "**CH343SER.EXE**".



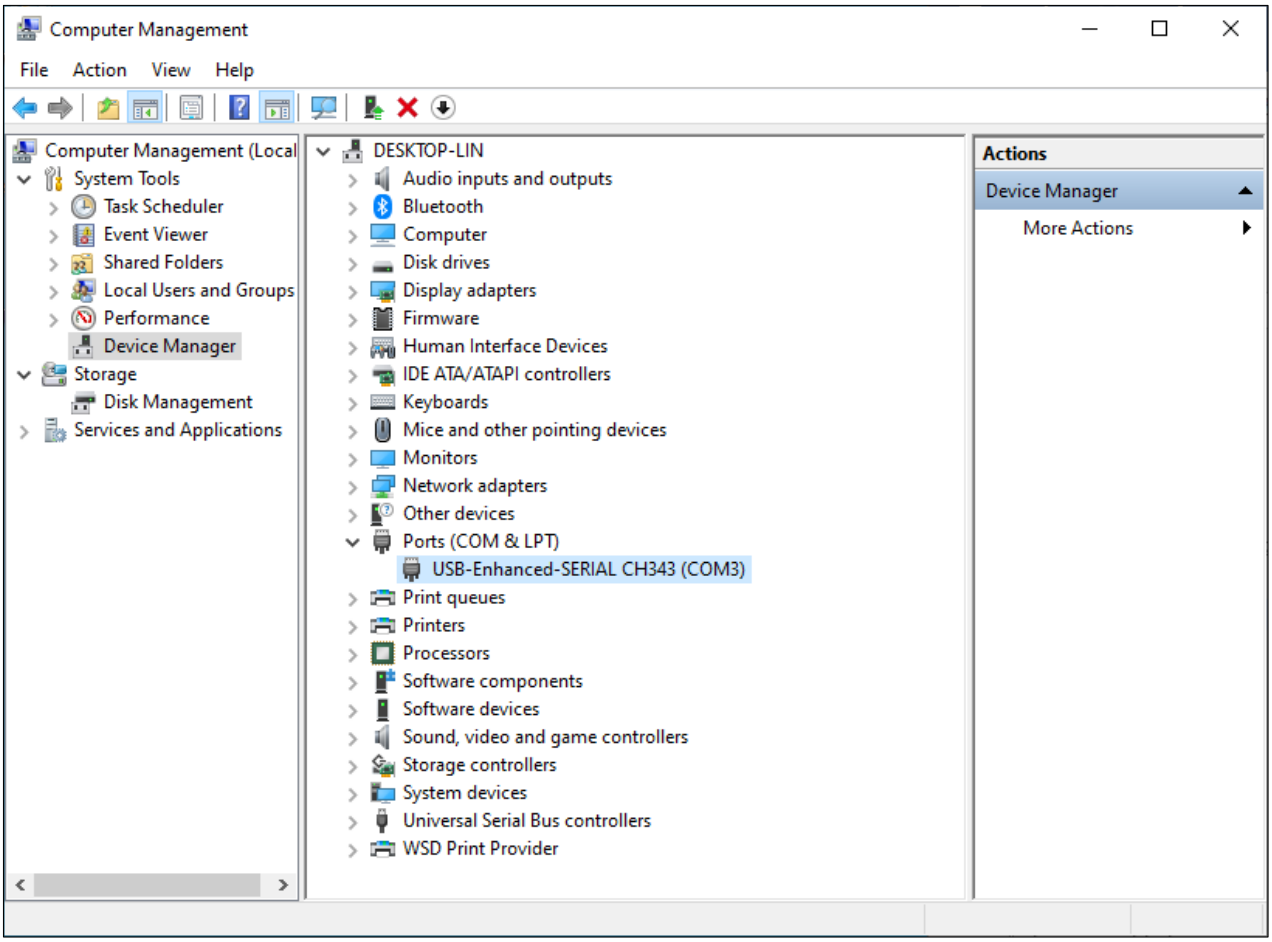
4. Click “INSTALL” and wait for the installation to complete.



5. Install successfully. Close all interfaces.



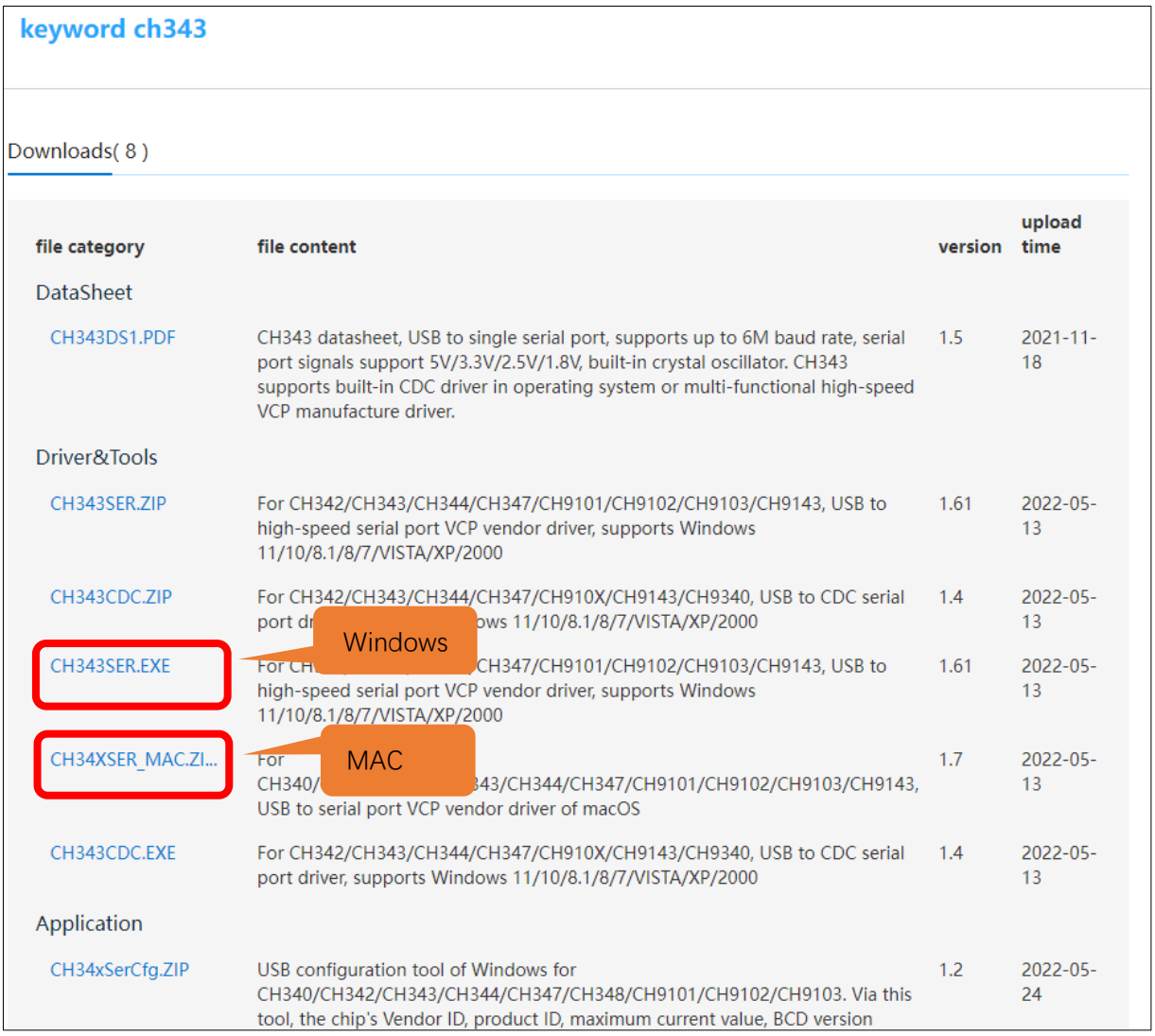
6. When ESP32-S3 WROOM is connected to computer, select “This PC”, right-click to select “Manage” and click “Device Manager” in the newly pop-up dialog box, and you can see the following interface.



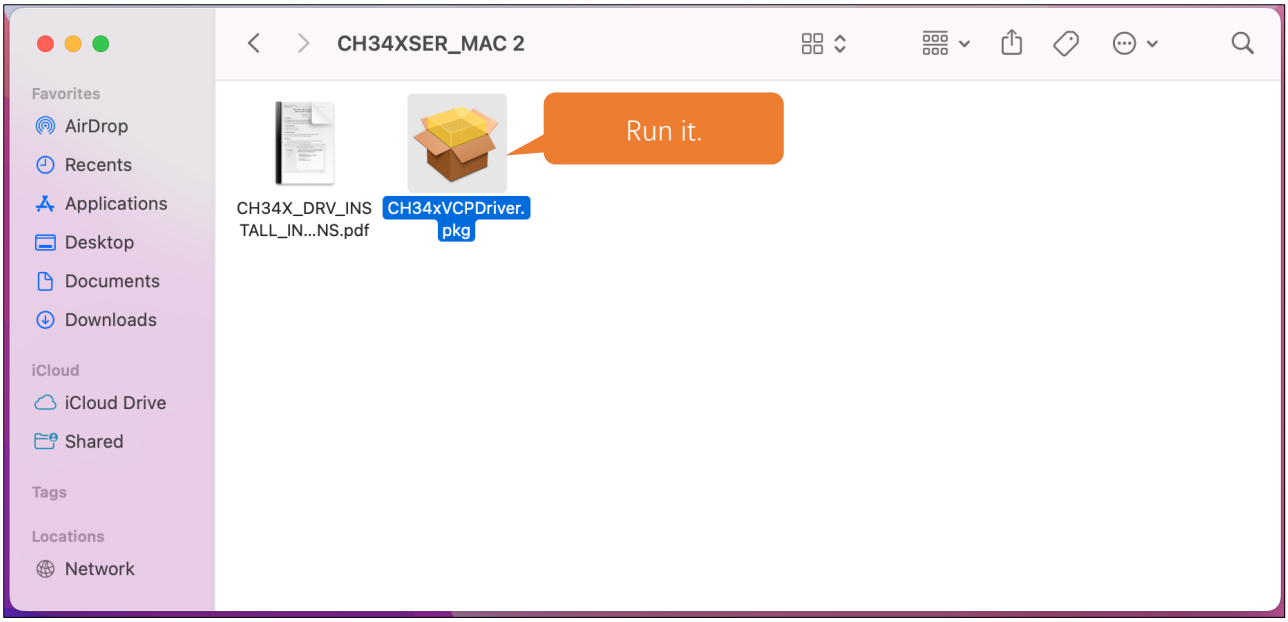
7. So far, CH343 has been installed successfully. Close all dialog boxes.

MAC

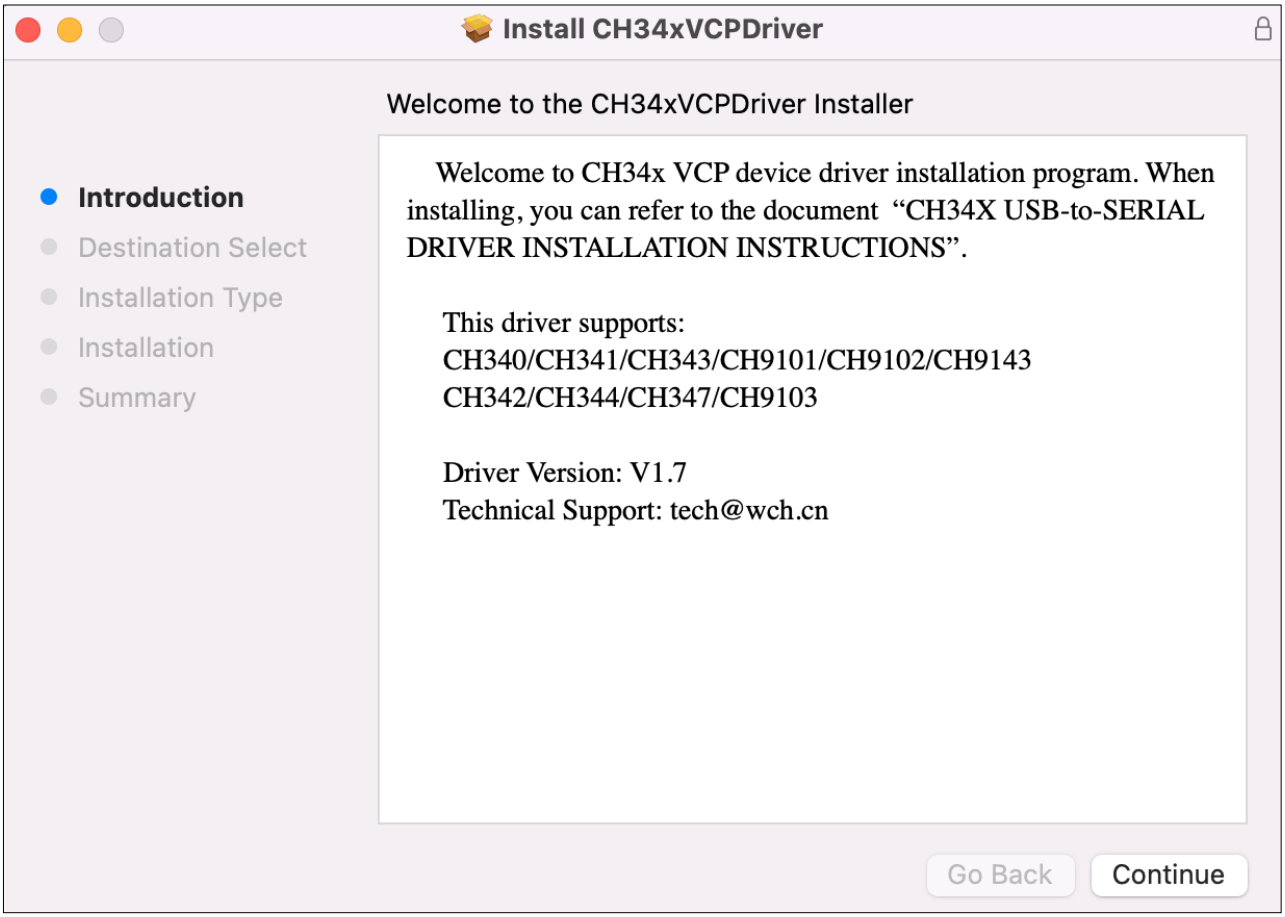
First, download CH343 driver, click http://www.wch-ic.com/search?t=all&q=ch343 to download the appropriate one based on your operating system.



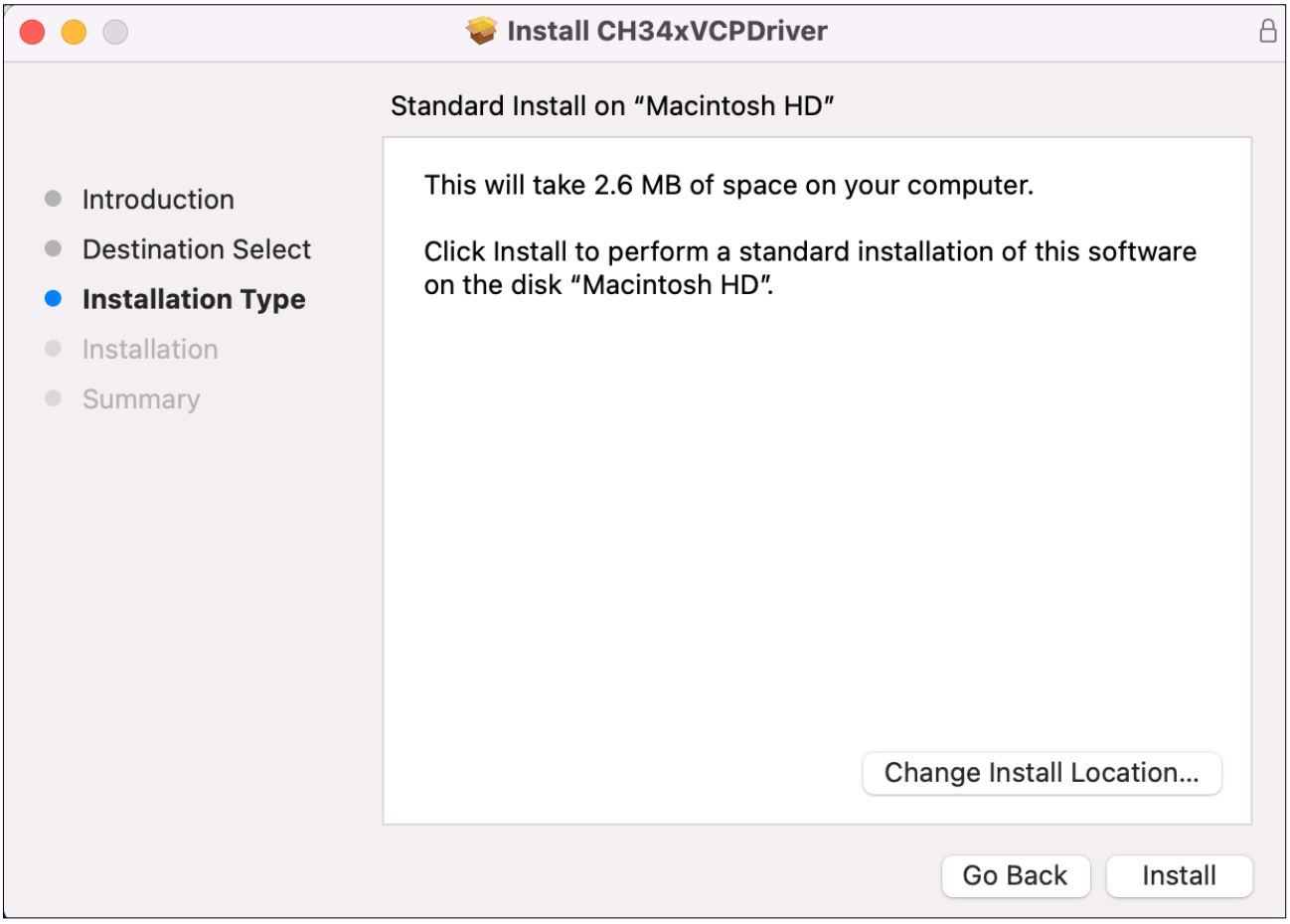
If you would not like to download the installation package, you can open **"Freenove_Super_Starter_Kit_for_ESP32_S3/CH343"**, we have prepared the installation package. Second, open the folder "**Freenove_Super_Starter_Kit_for_ESP32_S3/CH343/MAC/**"



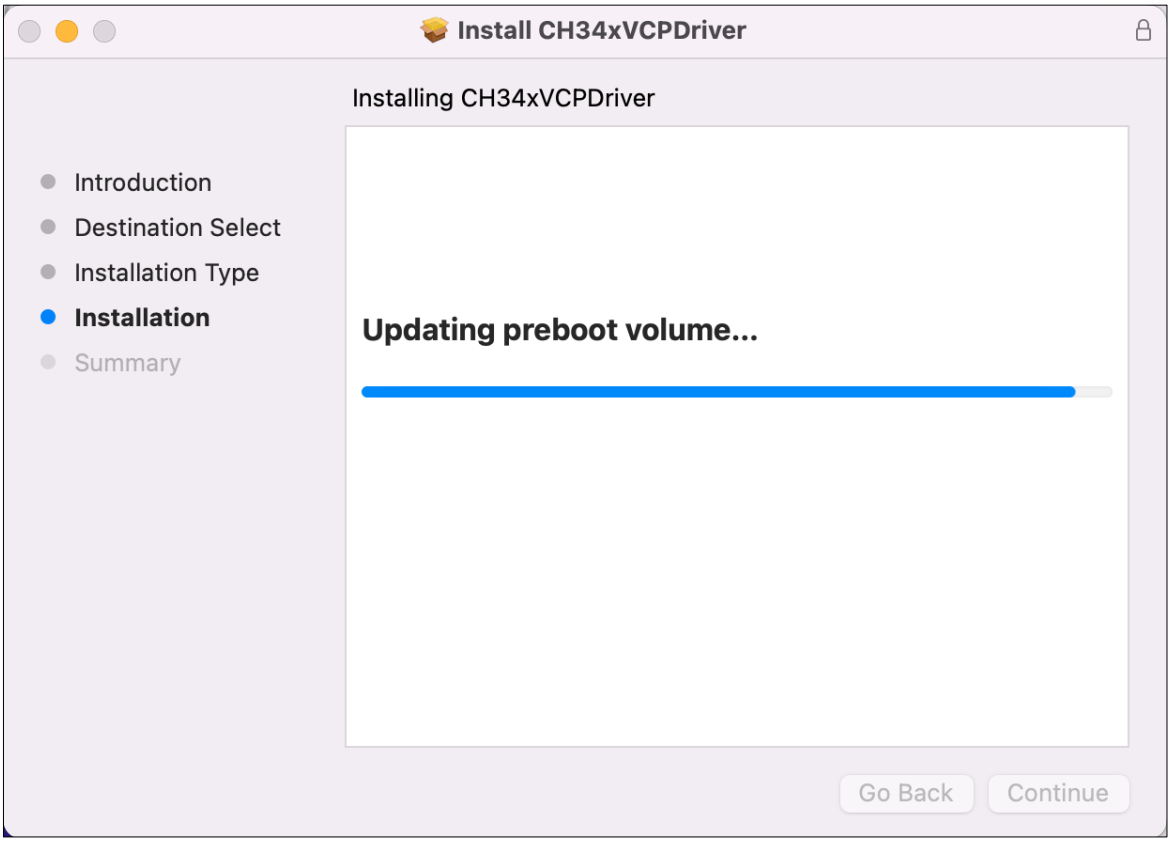
Third, click Continue.



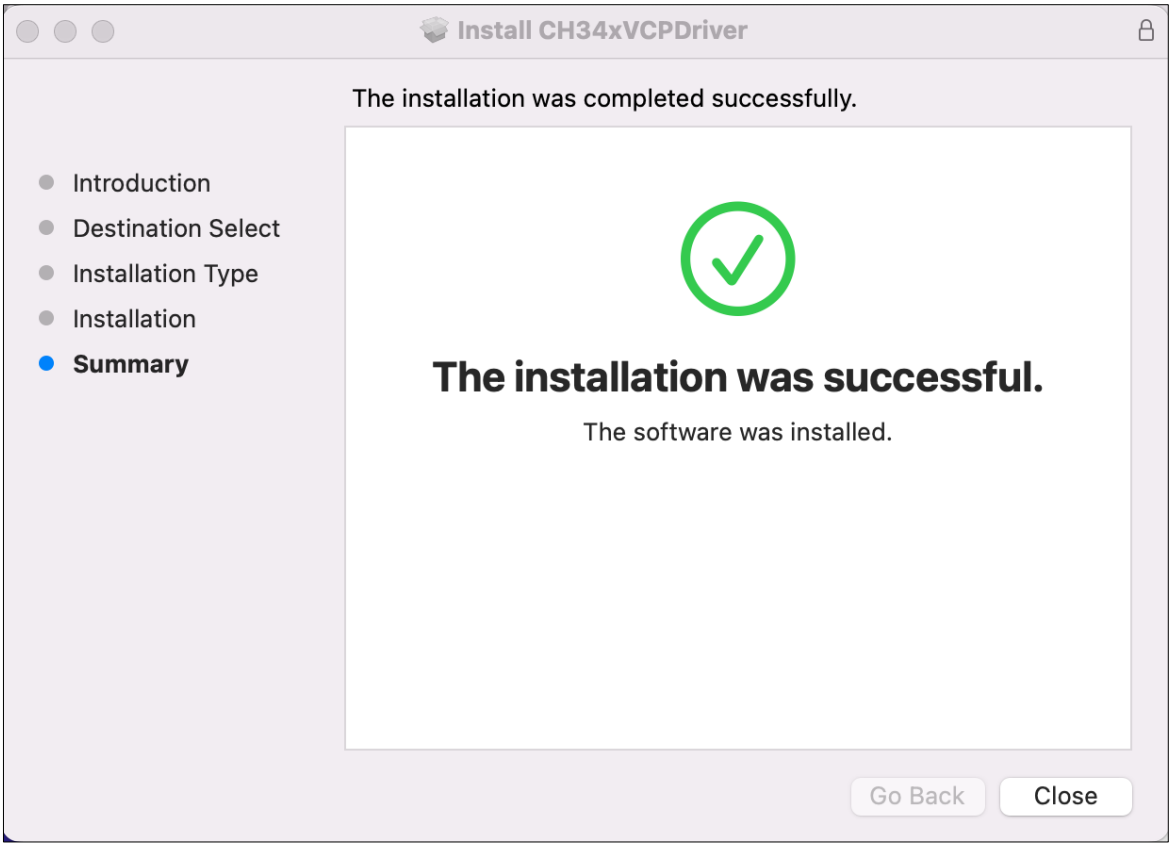
Fourth, click Install.



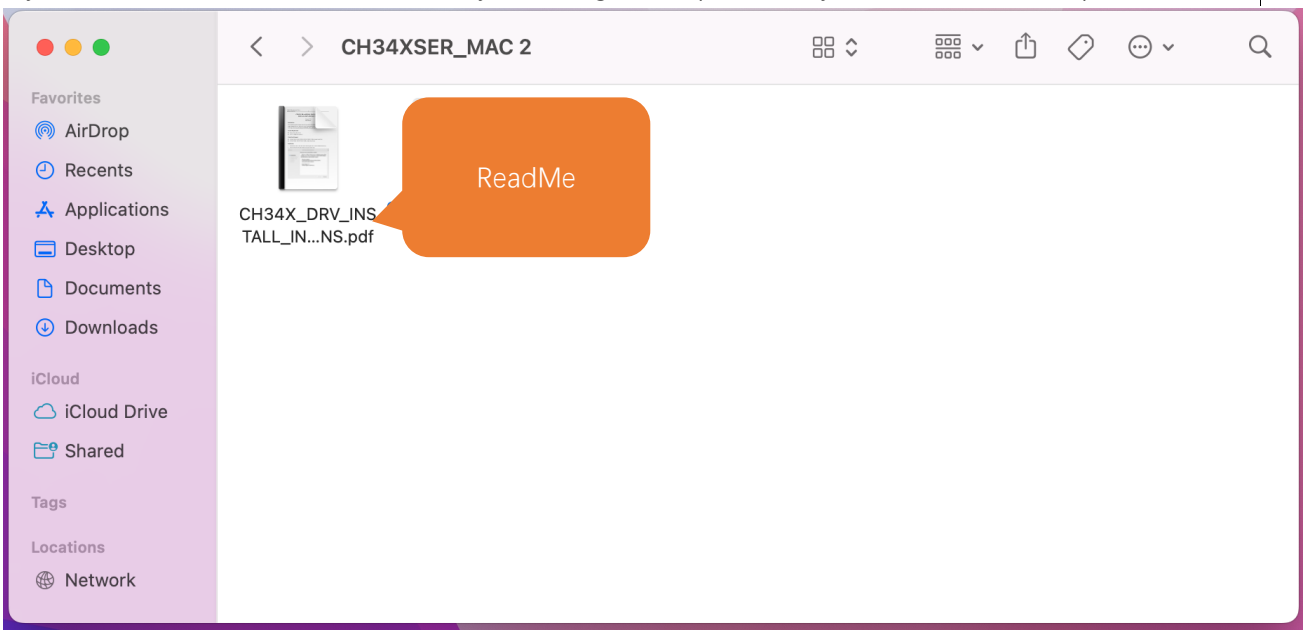
Then, waiting Finsh.



Finally, restart your PC.

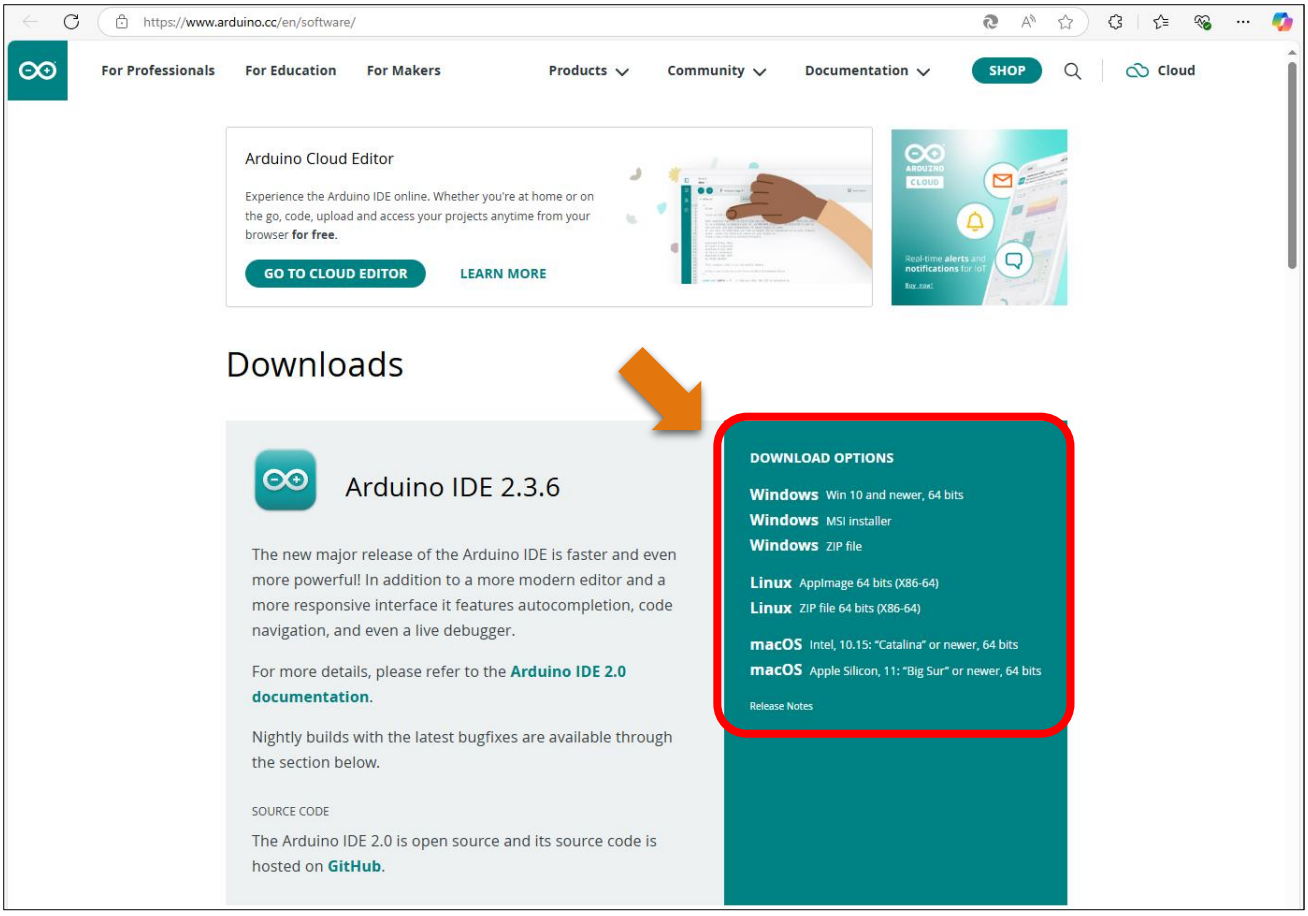


If you still haven't installed the CH340 by following the steps above, you can view readme.pdf to install it.



Programming Software

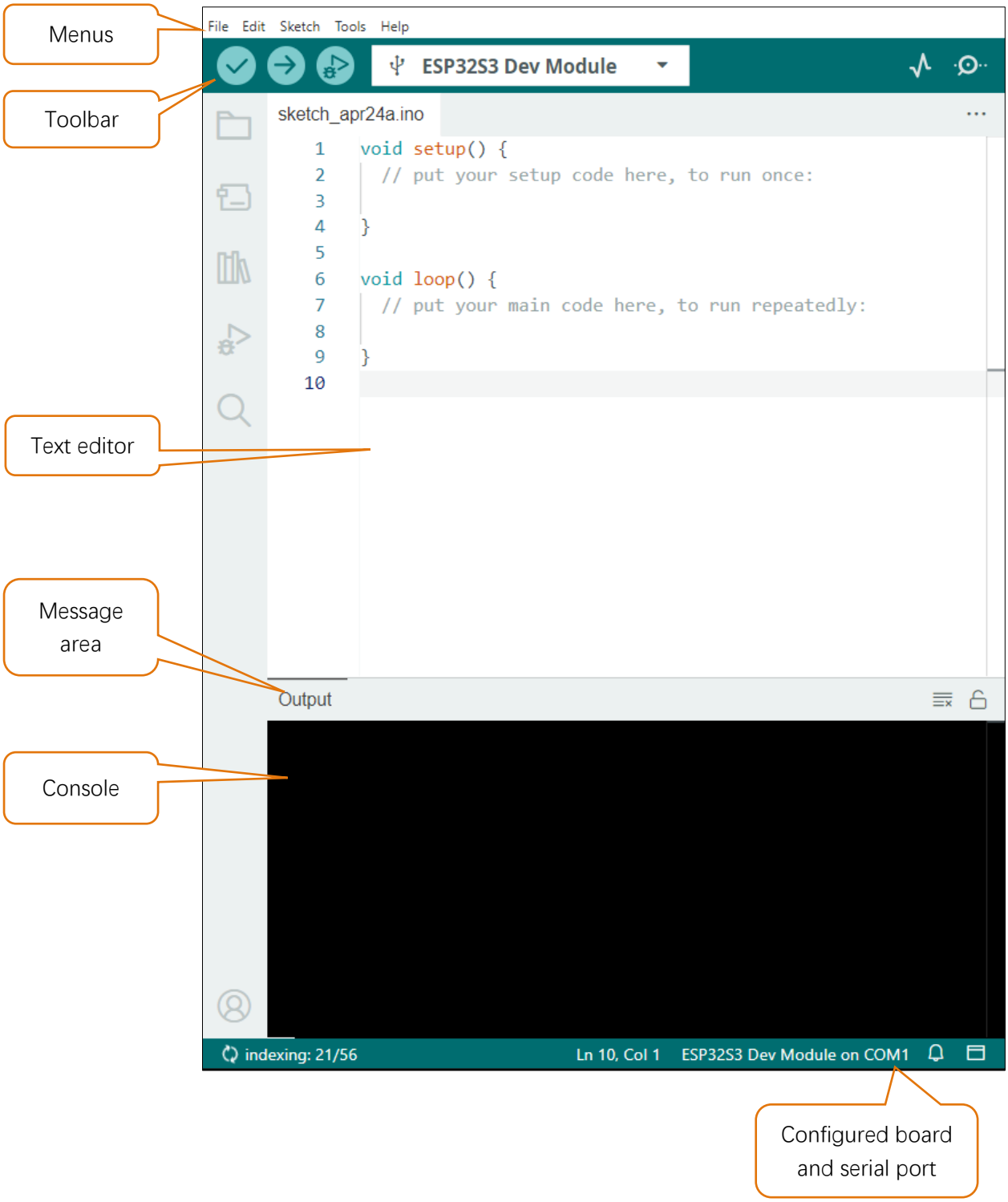
Arduino Software (IDE) is used to write and upload the code for Arduino Board.
First, install Arduino Software (IDE): visit https://www.arduino.cc/en/software/ to enter the download page. Select and download corresponding installer according to your operating system. If you are a windows user, please select the "Windows Installer" to download to install the driver correctly.



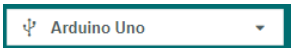
After the download completes, run the installer. For Windows users, there may pop up an installation dialog box of driver during the installation process. When it popes up, please allow the installation.

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

The interface of Arduino Software is as follows:



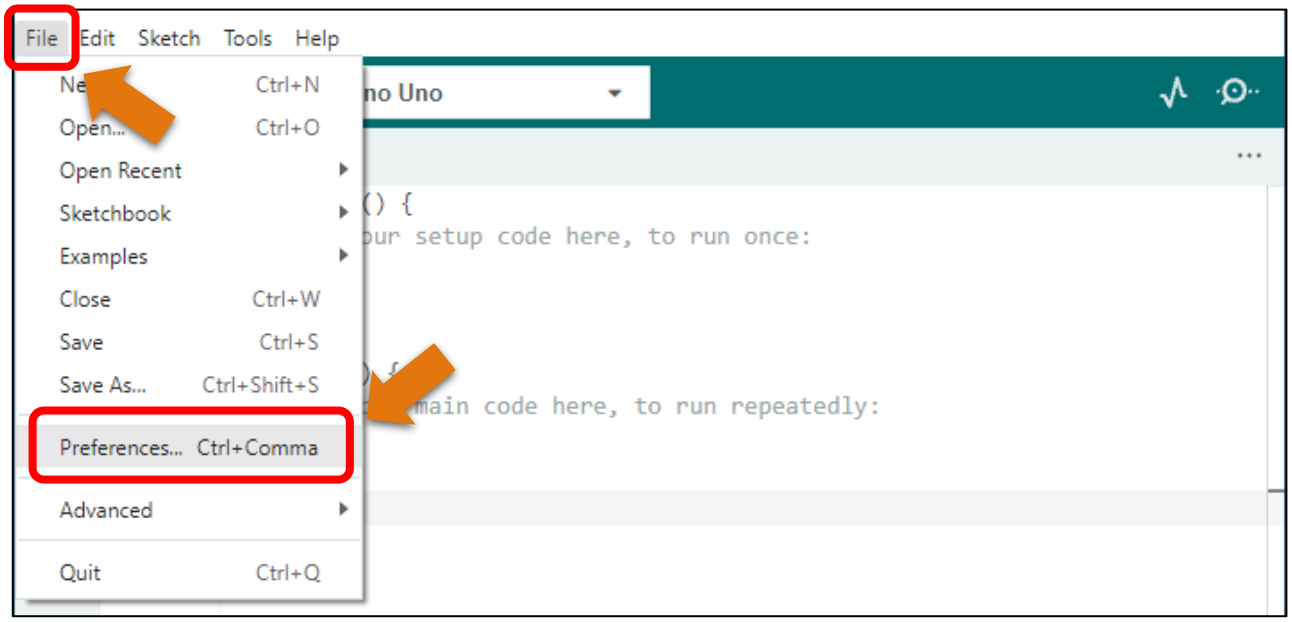
Programs written with Arduino Software (IDE) are called **sketches**. These sketches are written in the text editor and saved with the file extension**.ino**. The editor has features for cutting/pasting and searching/replacing text. The message area gives feedback while saving and exporting and also displays errors. The console displays text output by the Arduino Software (IDE), including complete error messages and other information. The bottom right-hand corner of the window displays the configured board and serial port. The toolbar buttons allow you to verify and upload programs, create, open, and save sketches, and open the serial monitor.

	Verify Check your code for compile errors .
	Upload Compile your code and upload them to the configured board.
	Debug Debug code running on the board. (Some development boards do not support this function)
	Development board selection Configure the support package and upload port of the development board.
	Serial Plotter Receive serial port data and plot it in a discounted graph.
	Serial Monitor Open the serial monitor.

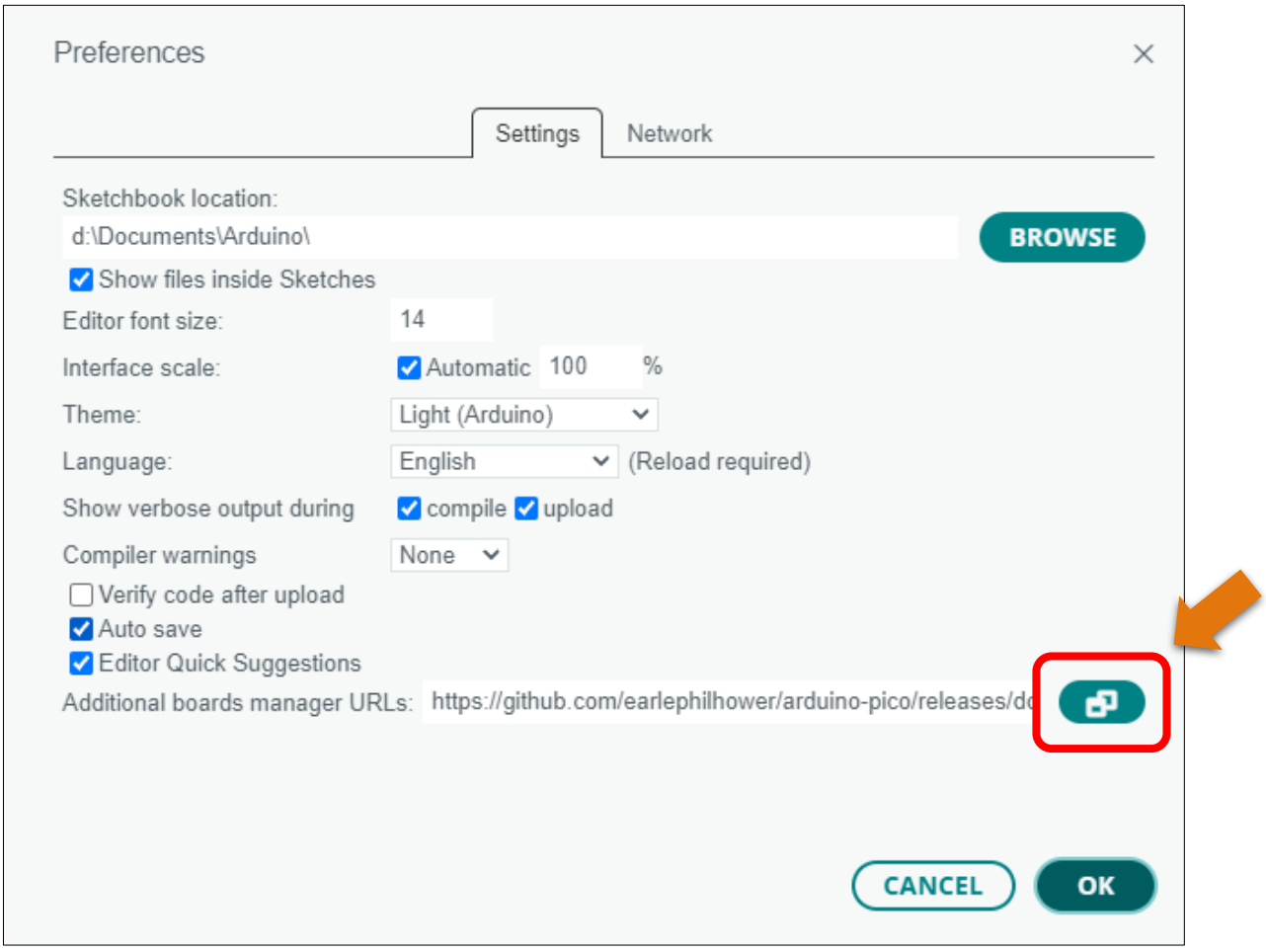
Additional commands are found within the five menus: File, Edit, Sketch, Tools, Help. The menus are context sensitive, which means only those items relevant to the work currently being carried out are available.

Environment Configuration

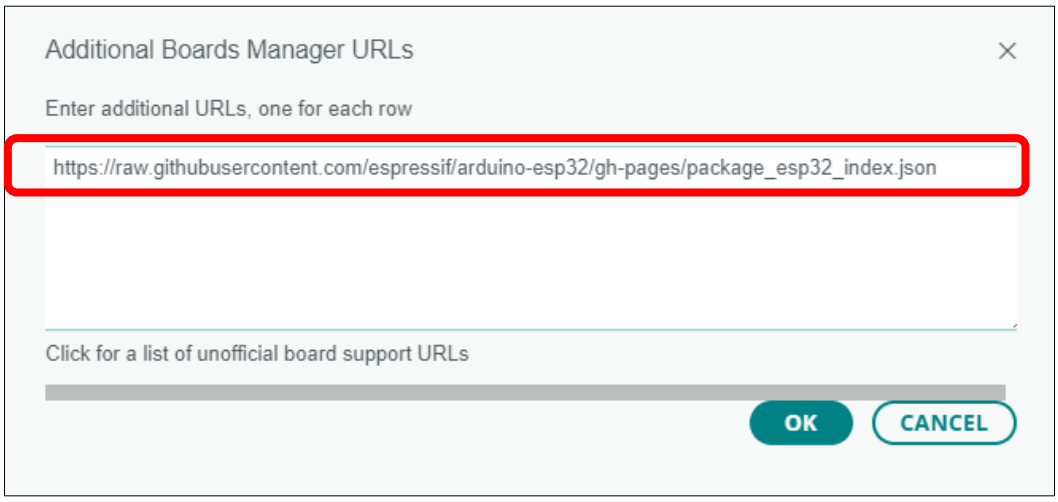
First, open the software platform arduino, and then click File in Menus and select Preferences.



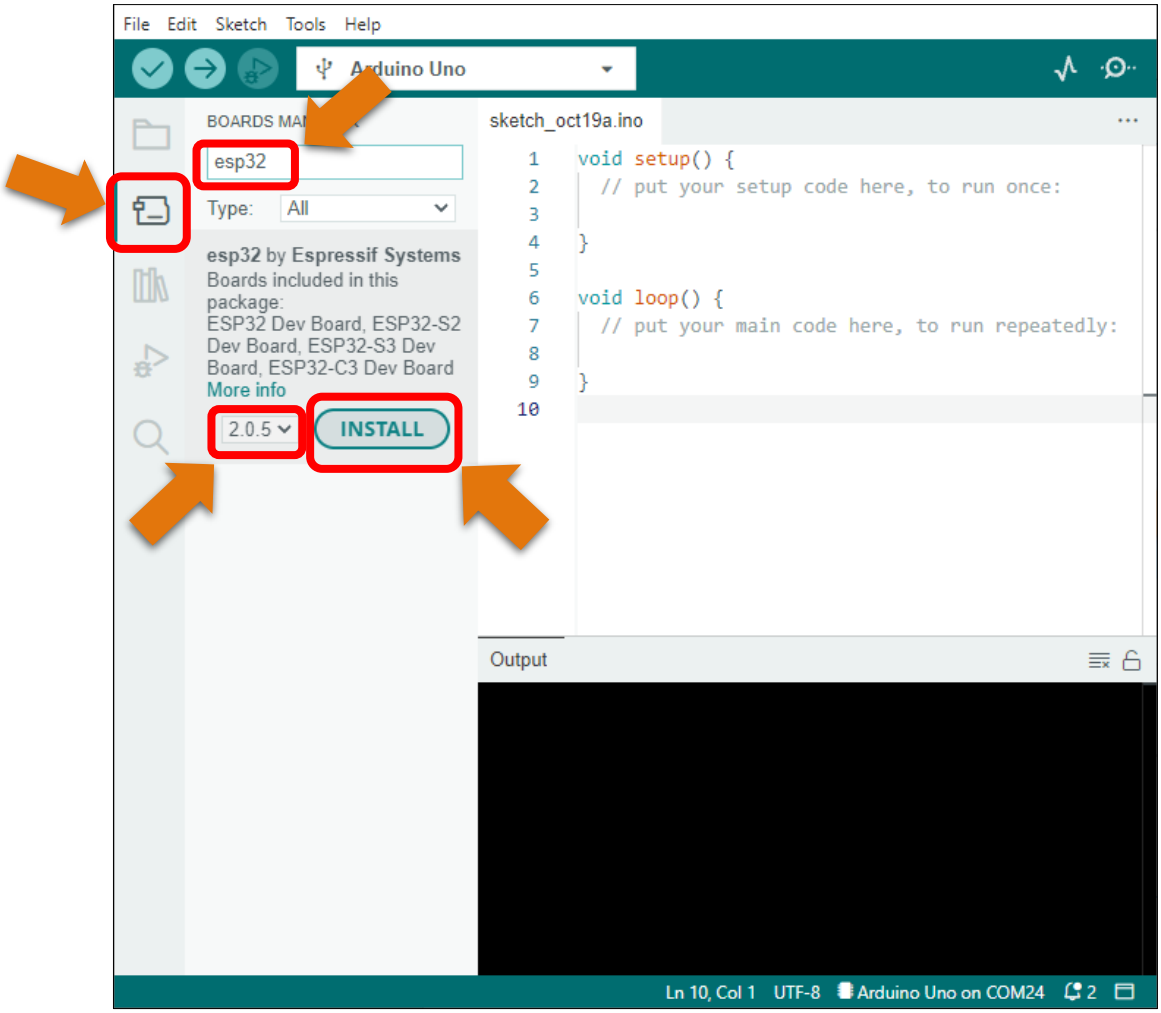
Second, click on the symbol behind "Additional Boards Manager URLs"



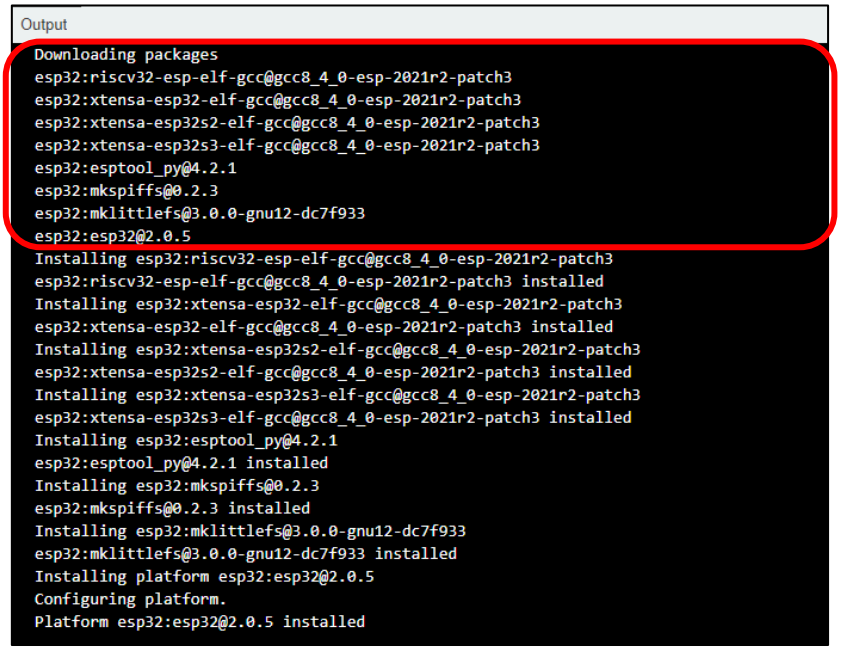
Third, fill in https://raw.githubusercontent.com/espressif/arduino-esp32/gh-pages/package_esp32_index.json in the new window, click OK, and click OK on the Preferences window again.



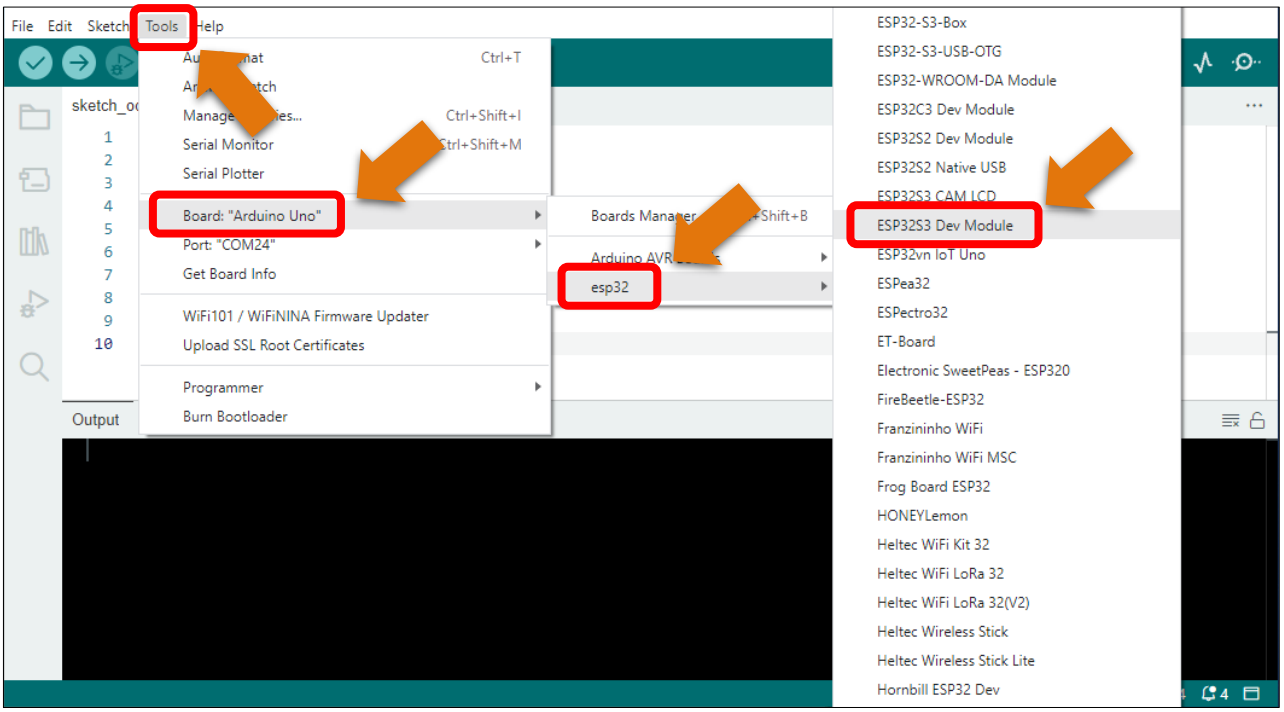
Fourth, click "Boards Manager". Enter “esp32” in Boards manager and select 2.0.5， Then click “INSTALL”.



Arduinowill download these files automaticly. Wait for the installation to complete.



When finishing installation, click Tools in the Menus again and select Board: "Arduino Uno", and then you can see information of ESP32. click "ESP32-S3 Dev Module" so that the ESP32-S3 programming development environment is configured.



Notes for GPIO

Strapping Pin

There are four Strapping pins for ESP32-S3：GPIO0、GPIO45、GPIO46、GPIO3。
With the release of the chip's system reset (power-on reset, RTC watchdog reset, undervoltage reset), the strapping pins sample the level and store it in the latch as "0" or "1" ", and keep it until the chip is powered off or turned off.
Each Strapping pin is connecting to internal pull-up/pull-down. Connecting to high-impedance external circuit or without an external connection, a strapping pin's default value of input level will be determined by internal weak pull-up/pull-down. To change the value of the Strapping, users can apply an external pull-down/pull-up resistor, or use the GPIO of the host MCU to control the level of the strapping pin when the ESP32-S3's power on reset is released.
When releasing the reset, the strapping pin has the same function as a normal pin.
The followings are default configurations of these four strapping pins at power-on and their functions under the corresponding configuration.

VDD_SPI Voltage			
Pin	Default	3.3 V	1.8 V
GPIO45	Pull-down	0	1
Booting Mode [1]			
Pin	Default	SPI Boot	Download Boot
GPIO0	Pull-up	1	0
GPIO46	Pull-down	Don't care	0
Enabling/Disabling ROM Messages Print During Booting [2]			
Pin	Default	Enabled	Disabled
GPIO46	Pull-down	See the 2nd note	See the 2nd note
JTAG Signal Selection			
Pin	Default	EFUSE_DIS_USB_JTAG = 0, EFUSE_DIS_PAD_JTAG = 0, EFUSE_STRAP_JTAG_SEL=1	
GPIO3	N/A	0: JTAG signal from on-chip JTAG pins 1: JTAG signal from USB Serial/JTAG controller	

Note:

1. The strapping combination of GPIO46 = 1 and GPIO0 = 0 is invalid and will trigger unexpected behavior.
2. By default, the ROM boot messages are printed over UART0 (U0TXD pin) and USB Serial/JTAG controller together. The ROM code printing can be disabled through configuration register and eFuse. For detailed information, please refer to Chapter Chip Boot Control in *ESP32-S3 Technical Reference Manual*.

If you have any difficulties or questions with this tutorial or toolkit, feel free to ask for our quick and free technical support through support@freenove.com at any time.
or check: https://www.espressif.com/sites/default/files/documentation/esp32-s3-wroom-1_wroom-1u_datasheet_en.pdf

Any concerns? ✉ support@freenove.com

PSRAM Pin

The module on the ESP32-S3-WROOM board uses the ESP32-S3R8 chip with 8MB of external Flash. When we use the OPI PSRAM, please note that the GPIO35-GPIO37 on the ESP32-S3-WROOM board will not be available for other purposes. When OPI PSRAM is not used, GPIO35-GPIO37 on the board can be used as normal GPIO.

ESP32-S3R8 / ESP32-S3R8V	In-package PSRAM (8 MB, Octal SPI)
SPICLK	CLK
SPICS1	CE#
SPID	DQ0
SPIQ	DQ1
SPIWP	DQ2
SPIHD	DQ3
GPIO33	DQ4
GPIO34	DQ5
GPIO35	DQ6
GPIO36	DQ7
GPIO37	DQS/DM

SDcard Pin

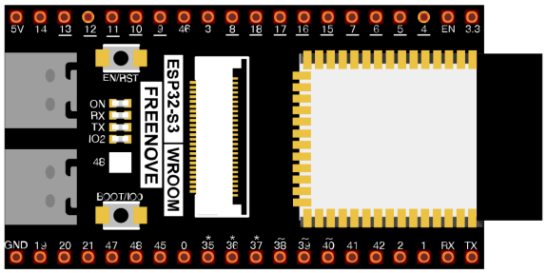
An SDcard slot is integrated on the back of the ESP32-S3-WROOM board. We can use GPIO38-GPIO40 of ESP32-S3-WROOM to drive SD card.
The SDcard of ESP32-S3-WROOM uses SDMMC, a 1-bit bus driving method, which has been integrated in the Arduino IDE, and we can call the "SD_MMC.h" library to drive it. For details, see the SDcard chapter in this tutorial.

USB Pin

In Micropython, GPIO19 and GPIO20 are used for the USB function of ESP32S3, so they cannot be used as other functions!

Cam Pin

When using the camera of our ESP32-S3 WROOM, please check the pins of it. Pins with underlined numbers are used by the camera function, if you want to use other functions besides it, please avoid using them.



CAM_Pin	GPIO_pin
SIOD	GPIO4
SIOC	GPIO5
CSI_VYSNC	GPIO6
CSI_HREF	GPIO7
CSI_Y9	GPIO16
XCLK	GPIO15
CSI_Y8	GPIO17
CSI_Y7	GPIO18
CSI_PCLK	GPIO13
CSI_Y6	GPIO12
CSI_Y2	GPIO11
CSI_Y5	GPIO10
CSI_Y3	GPIO9
CSI_Y4	GPIO8

If you have any questions about the information of GPIO, you can click here to go back to ESP32-S3 WROOM to view specific information about GPIO.
or check: https://www.espressif.com/sites/default/files/documentation/esp32-s3_datasheet_en.pdf.

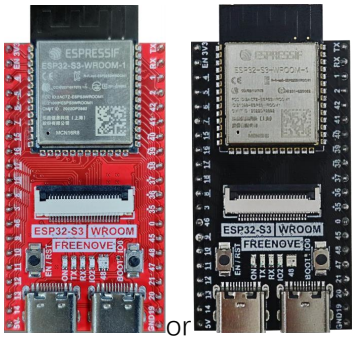
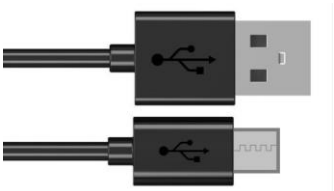
Chapter 0 LED

This chapter is the Start Point in the journey to build and explore ESP32-S3 WROOM electronic projects. We will start with simple “Blink” project.

Project 0.1 Blink

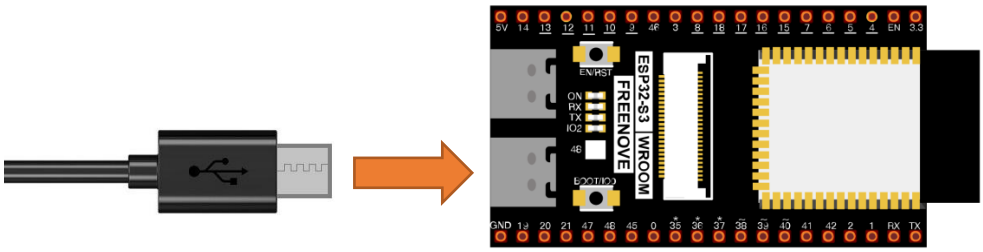
In this project, we will use ESP32-S3 WROOM to control blinking a common LED.

Component List

ESP32-S3(N16R8)/ESP32-S3(N8R8) 	USB cable x1 

Power

ESP32-S3 WROOM needs 5v power supply. In this tutorial, we need connect ESP32-S3 WROOM to computer via USB cable to power it and program it. We can also use other 5v power source to power it.



In the following projects, we only use USB cable to power ESP32-S3 WROOM by default.

In the whole tutorial, we don’t use T extension to power ESP32-S3 WROOM. So 5V and 3.3V (includeing EXT 3.3V) on the extension board are provided by ESP32-S3 WROOM.

We can also use DC jack of extension board to power ESP32-S3 WROOM. In this way, 5v and EXT 3.3v on extension board are provided by external power resource.

Sketch

According to the circuit, when the GPIO2 of ESP32-S3 WROOM output level is high, the LED turns ON. Conversely, when the GPIO2 ESP32-S3 WROOM output level is low, the LED turns OFF. Therefore, we can let GPIO2 circularly output high and low level to make the LED blink.

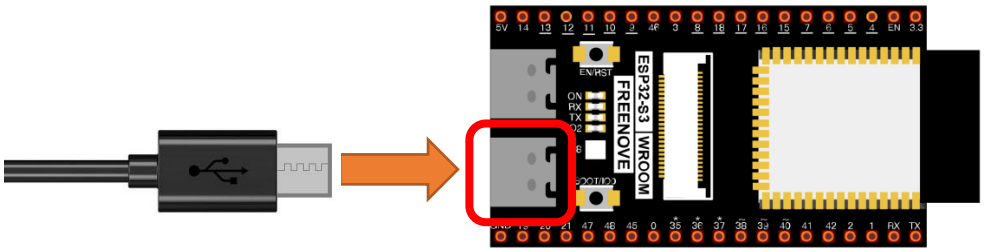
Upload the following Sketch:

Freenove_Super_Starter_Kit_for_ESP32_S3\Sketches\Sketch_01.1_Blink.

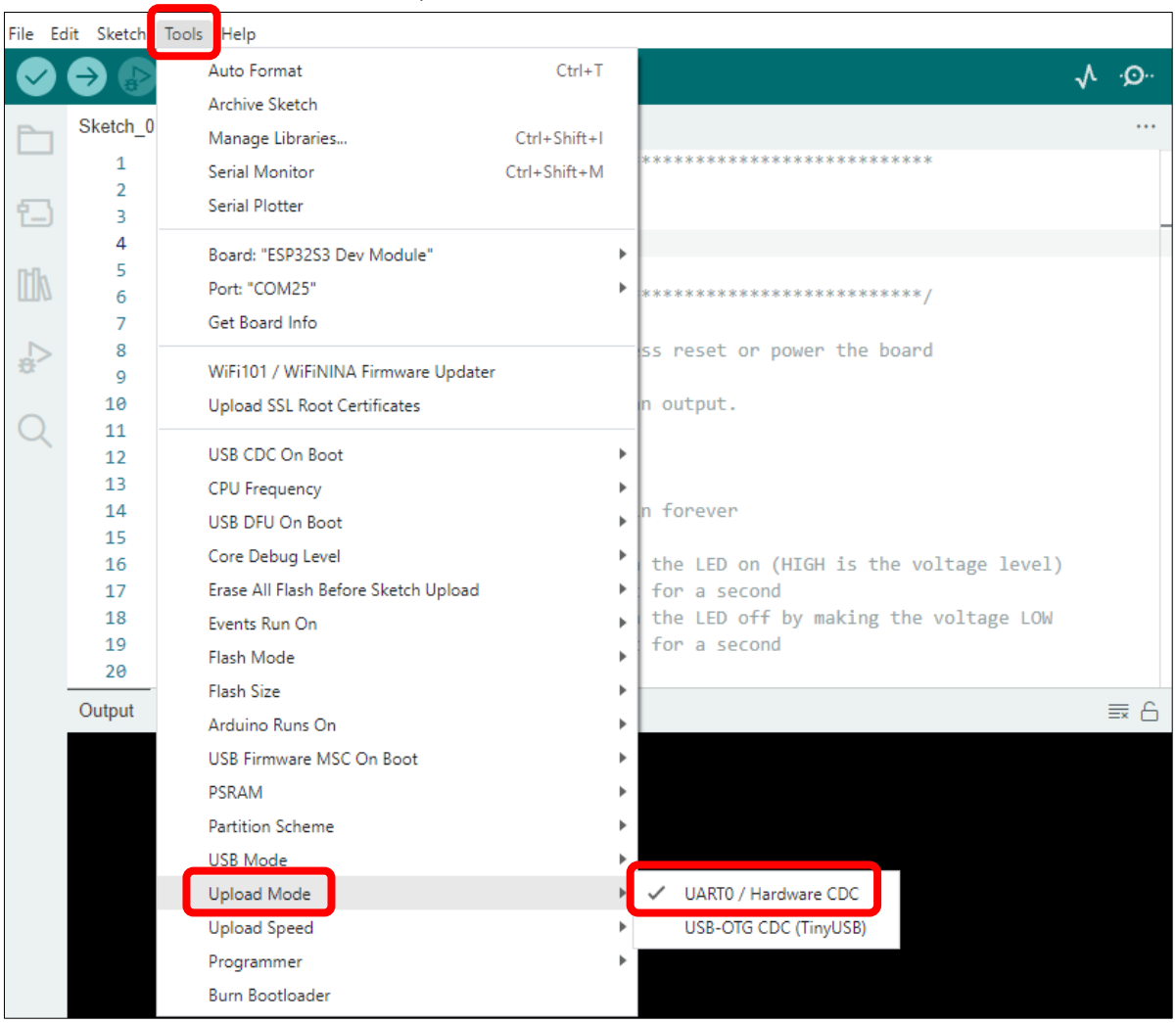
Next we will introduce two ways to upload code to ESP32-S3 WROOM.

Option 1:

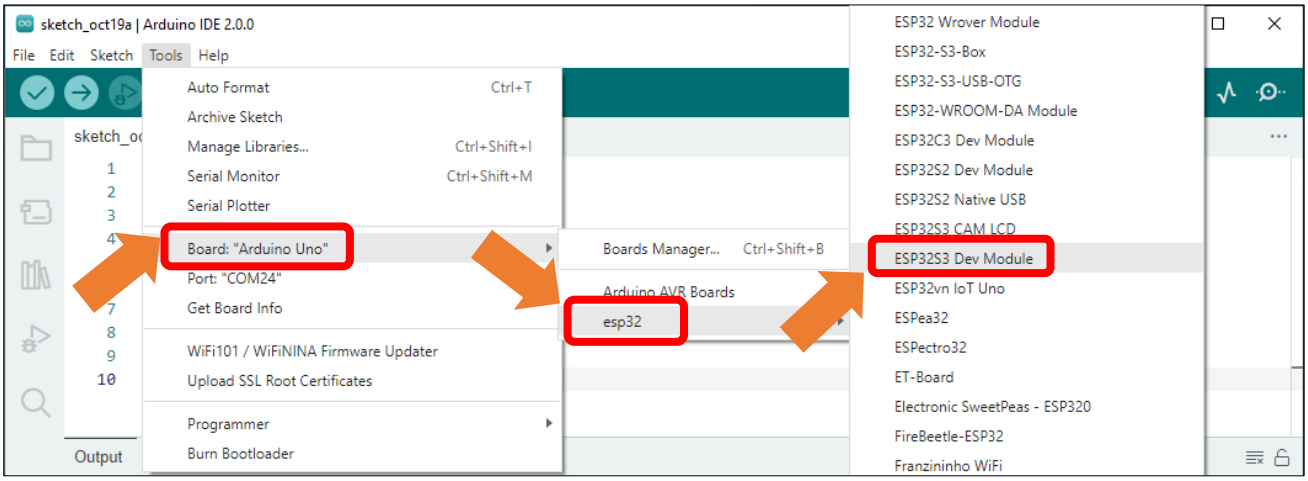
Connect ESP32-S3 WROOM to computer.



Open Arduino IDE 2.0.0. Click Tools->Upload Mode. Select UART0 / Hardware CDC.

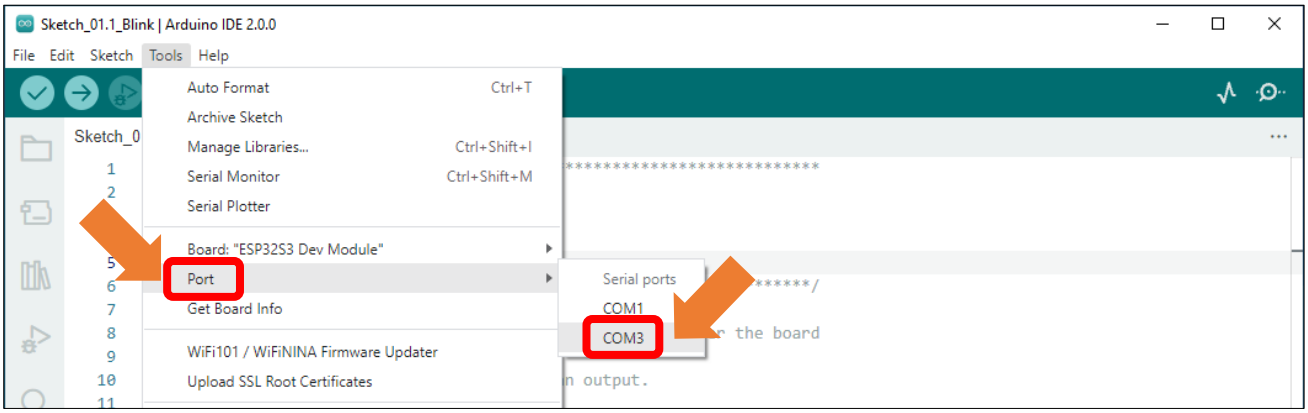


Before uploading the code, click "**Tools**", "**Board**" and select "**ESP32S3 Dev Module**".

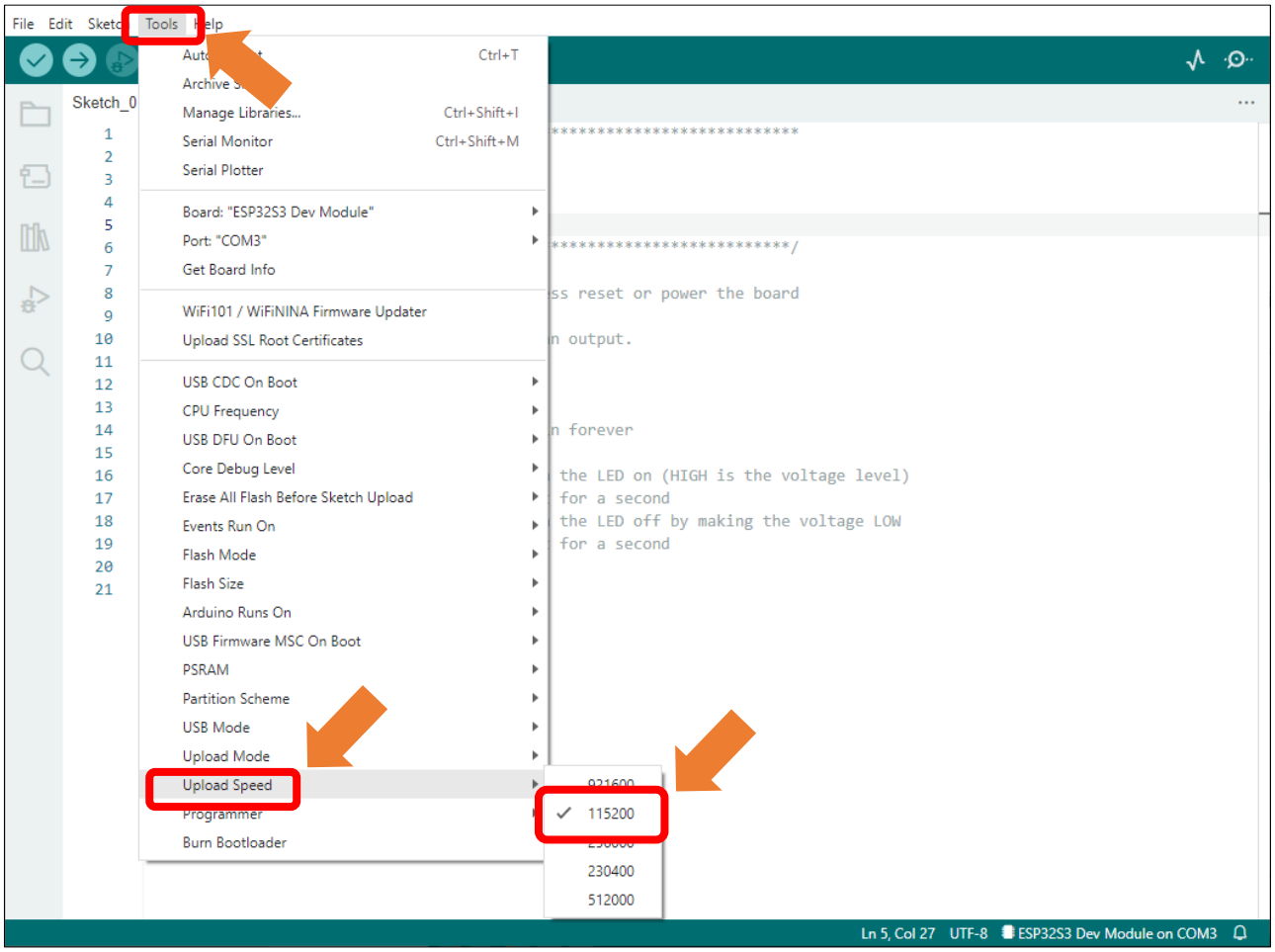


Select the serial port.

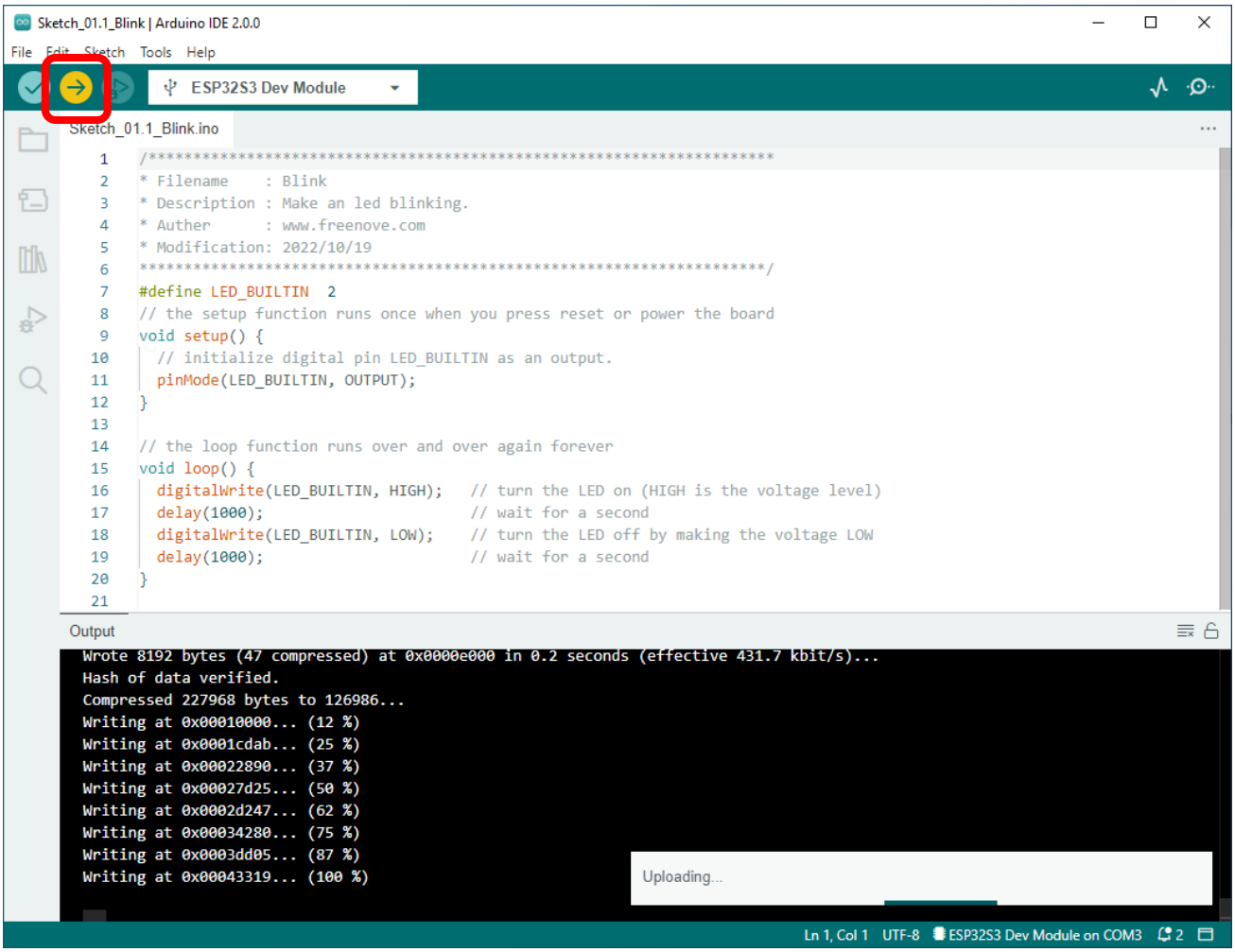
Note that the computer port number of each user may be different. Please select the correct serial port according to your computer. Taking the window system as an example, my computer recognizes that the communication interface of the ESP32-S3-WROOM is COM3, so I select COM3.



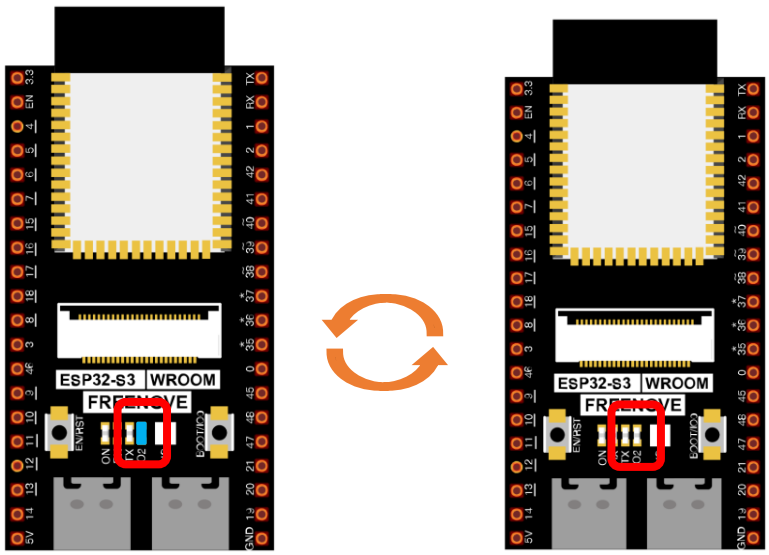
Note: For macOS users, if the uploading fails, please set the baud rate to 115200 before clicking "Upload Using Programmer".



Click the Upload button and it will compile and upload the Sketch to the ESP32-S3-WROOM.



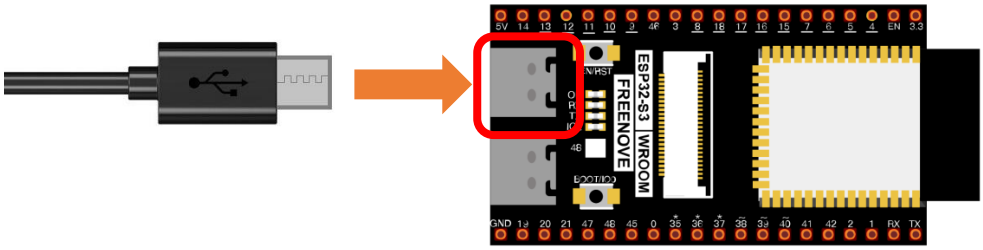
Wait for the Sketch upload to complete, and observe the ESP32-S3-WROOM. You can see that the blue LED (IO2) on the board flashes cyclically.



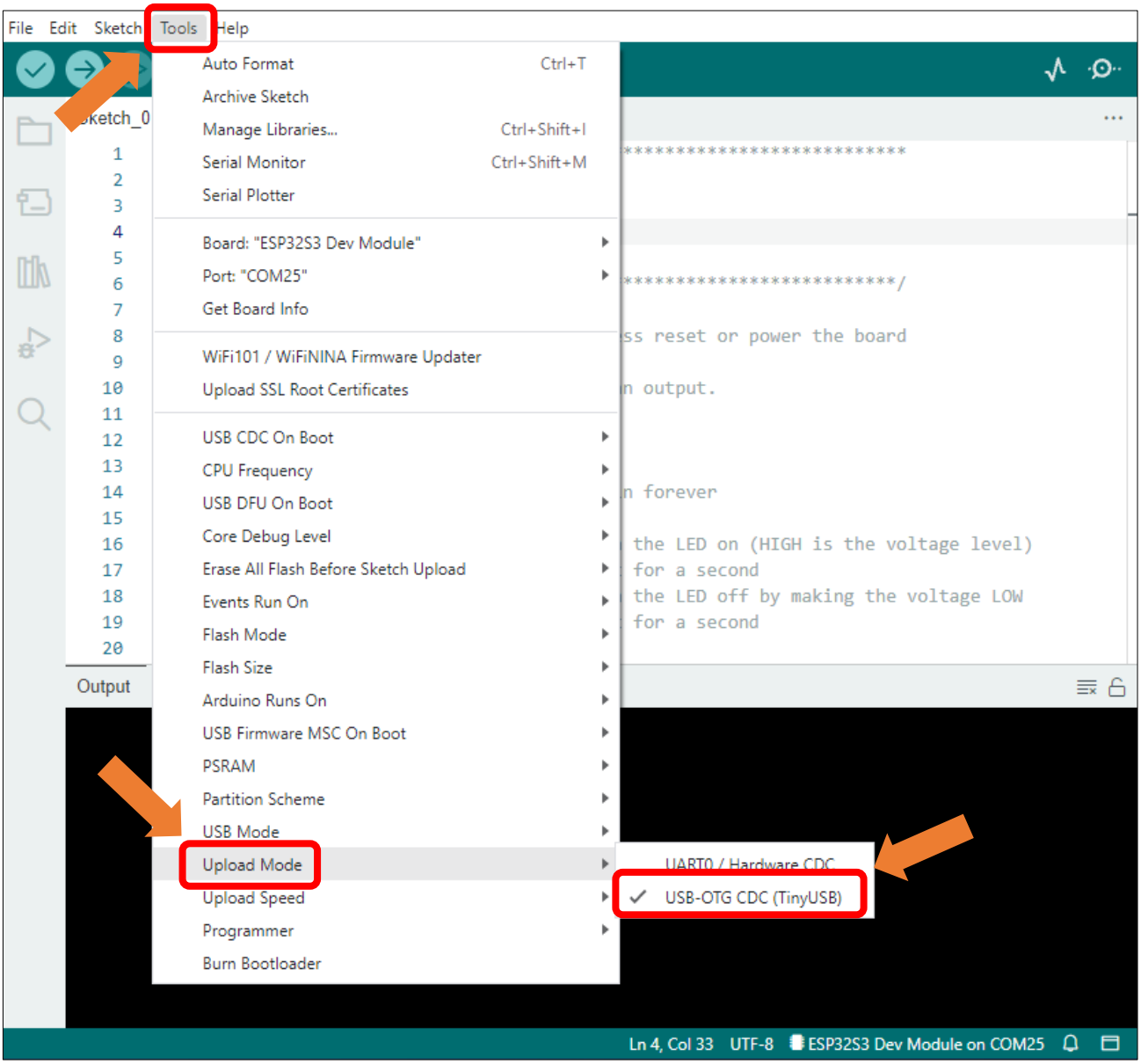
If you have any concerns, please contact us via: support@freenove.com.

Option 2:

Connect ESP32-S3 WROOM to computer.

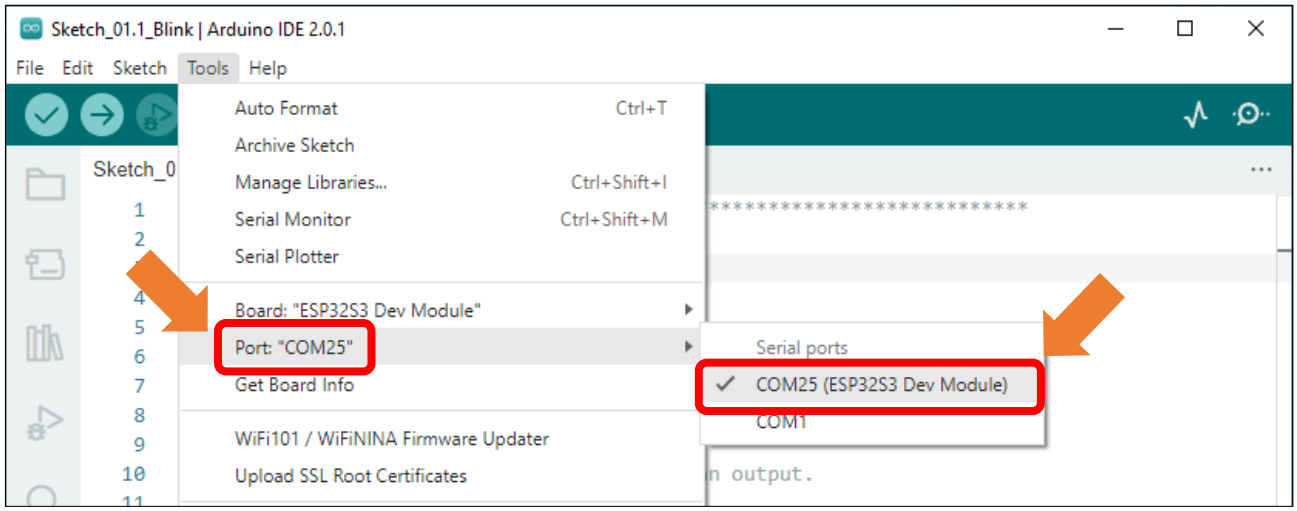


Open Arduino IDE 2.0.0. Click Tools->Upload Mode. Select USB-OTG CDC(TinyUSB).

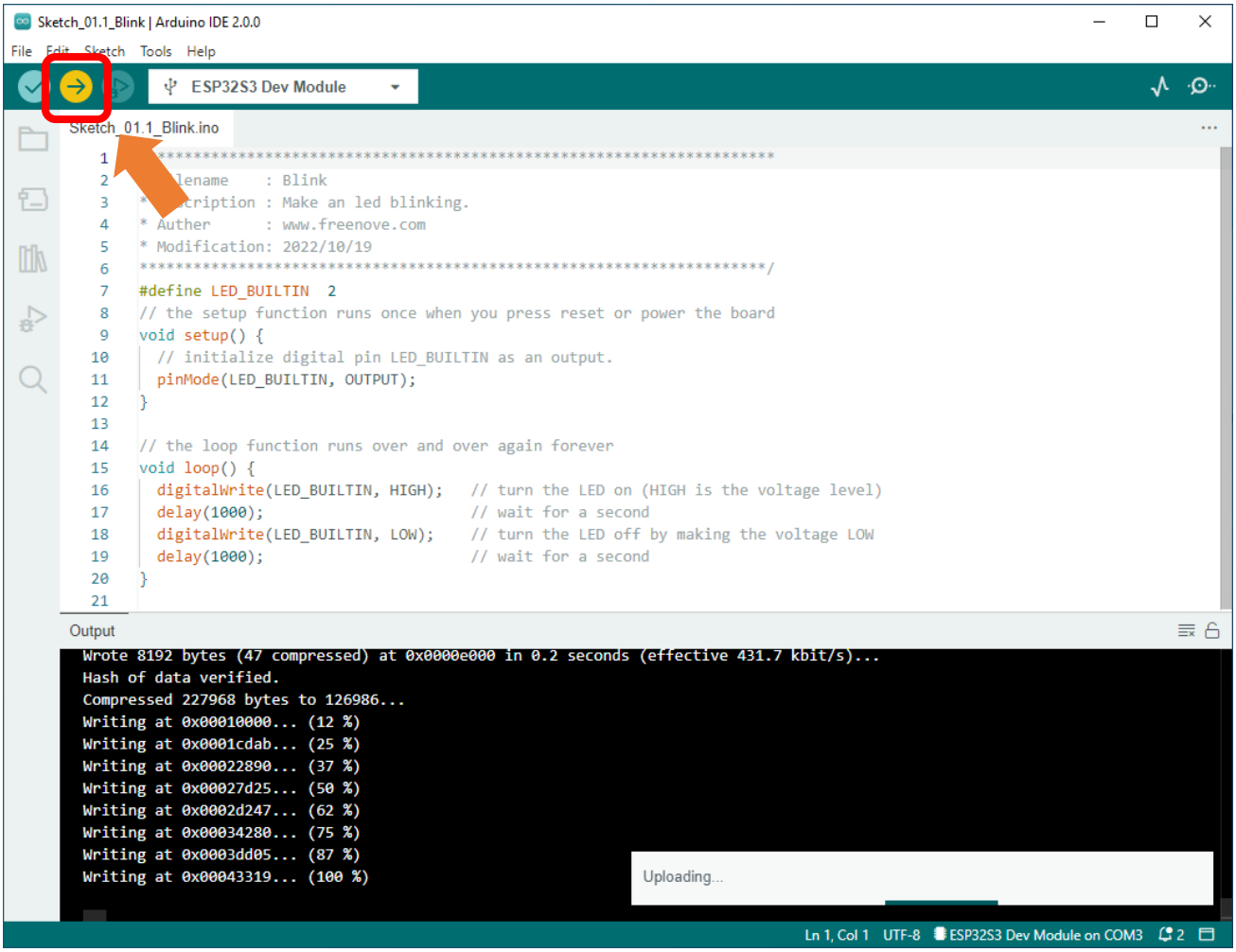


Select the serial port.

Note that the computer port number of each user may be different. Please select the correct serial port according to your computer. Taking the window system as an example, my computer recognizes that the communication interface of the ESP32-S3-WROOM is COM25, so I select COM25.

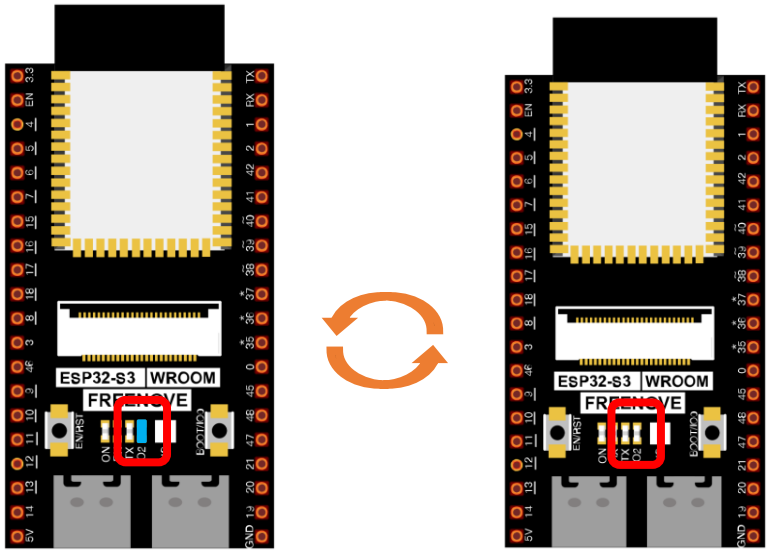


Click the Upload button and it will compile and upload the Sketch to the ESP32-S3-WROOM.



Wait for the Sketch upload to complete, and observe the ESP32-S3-WROOM. You can see that the blue

LED (IO2) on the board flashes cyclically.

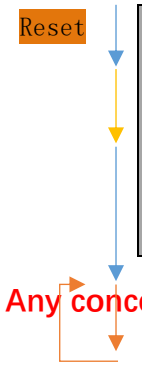


Sketch_01.1_Blink

The following is the program code:

```
1  #define LED_BUILTIN 2
2  // the setup function runs once when you press reset or power the board
3  void setup() {
4    // initialize digital pin LED_BUILTIN as an output.
5    pinMode(LED_BUILTIN, OUTPUT);
6  }
7
8  // the loop function runs over and over again forever
9  void loop() {
10   digitalWrite(LED_BUILTIN, HIGH);   // turn the LED on (HIGH is the voltage level)
11   delay(1000);                       // wait for a second
12   digitalWrite(LED_BUILTIN, LOW);    // turn the LED off by making the voltage LOW
13   delay(1000);                       // wait for a second
14 }
```

The Arduino IDE code usually contains two basic functions: void setup() and void loop().
After the board is reset, the setup() function will be executed firstly, and then the loop() function.
setup() function is generally used to write code to initialize the hardware. And loop() function is used to write code to achieve certain functions. loop() function is executed repeatedly. When the execution reaches the end of loop(), it will jump to the beginning of loop() to run again.




```
1  // the setup function runs once when you press reset or power the board
2  void setup() {
…    …
5  }
6
7  // the loop function runs over and over again forever
```

```
8   void loop() {
...   ...
13  }
```

Reset

Reset operation will lead the code to be executed from the beginning. Switching on the power, finishing uploading the code and pressing the reset button will trigger reset operation.

In the circuit, ESP32-S3 WROOM's GPIO2 is connected to the LED, so the LED pin is defined as 2.

```
1   #define LED_BUILTIN 2
```

This means that after this line of code, all LED_BUILTIN will be treated as 2.

In the setup () function, first, we set the LED_BUILTIN as output mode, which can make the port output high level or low level.

```
4   // initialize digital pin LED_BUILTIN as an output.
5   pinMode(LED_BUILTIN, OUTPUT);
```

Then, in the loop () function, set the LED_BUILTIN to output high level to make LED light up.

```
10  digitalWrite(LED_BUILTIN, HIGH);  // turn the LED on (HIGH is the voltage level)
```

Wait for 1000ms, that is 1s. Delay () function is used to make control board wait for a moment before executing the next statement. The parameter indicates the number of milliseconds to wait for.

```
11  delay(1000);                      // wait for a second
```

Then set the LED_BUILTIN to output low level, and LED light off. One second later, the execution of loop () function will be completed.

```
12  digitalWrite(LED_BUILTIN, LOW);   // turn the LED off by making the voltage LOW
13  delay(1000);                      // wait for a second
```

The loop() function is constantly being executed, so LED will keep blinking.

Reference

void pinMode(int pin, int mode);

Configures the specified pin to behave either as an input or an output.

Parameters

pin: the pin number to set the mode of.

mode: INPUT, OUTPUT, INPUT_PULLDOWM, or INPUT_PULLUP.

void digitalWrite (int pin, int value);

Writes the value HIGH or LOW (1 or 0) to the given pin which must have been previously set as an output.

For more related functions, please refer to https://www.arduino.cc/reference/en/

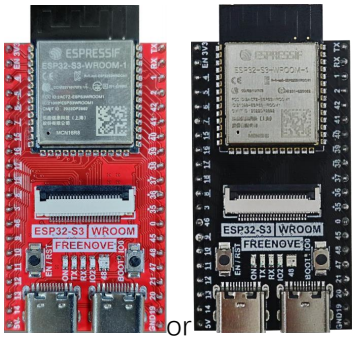
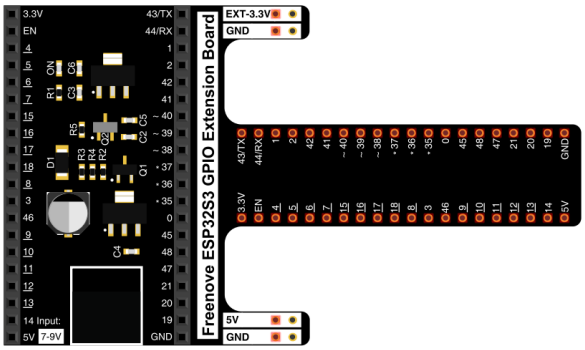
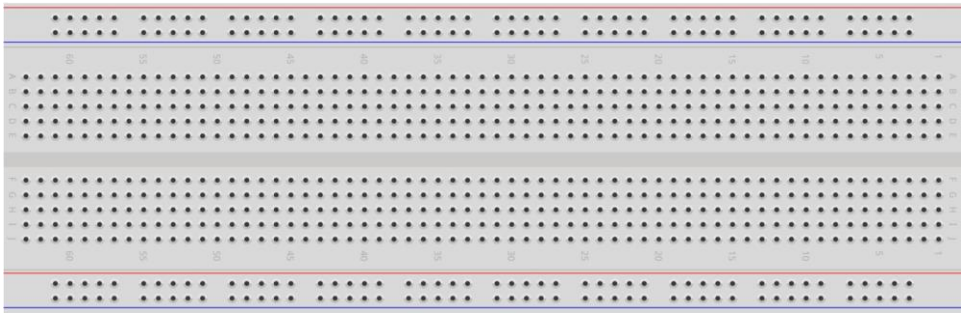
Chapter 1 LED

This chapter is the Start Point in the journey to build and explore ESP32-S3 WROOM electronic projects. We will start with simple “Blink” project.

Project 1.1 Blink

In this project, we will use ESP32-S3 WROOM to control blinking a common LED.

Component List

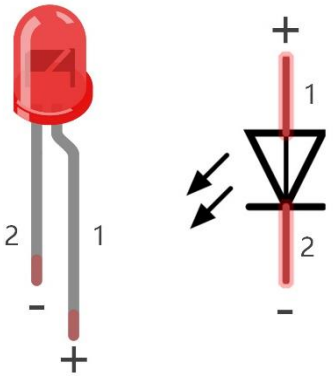
<table>
<tr><td>ESP32-S3(N16R8)/ESP32-S3(N8R8)
 
or</td><td>GPIO Extension Board x1
  </td></tr>
<tr><td colspan="2">Breadboard x1
  </td></tr>
<tr><td>LED x1
  </td><td>Resistor 220Ω x1
  </td><td>Jumper M/M x2
  </td></tr>
</table>

Component knowledge

LED

A LED is a type of diode. All diodes only work if current is flowing in the correct direction and have two poles. A LED will only work (light up) if the longer pin (+) of LED is connected to the positive output from a power source and the shorter pin is connected to the negative (-). Negative output is also referred to as Ground (GND). This type of component is known as "diodes" (think One-Way Street).

All common 2 lead diodes are the same in this respect. Diodes work only if the voltage of its positive electrode is higher than its negative electrode and there is a narrow range of operating voltage for most all common diodes of 1.9 and 3.4V. If you use much more than 3.3V the LED will be damaged and burn out.



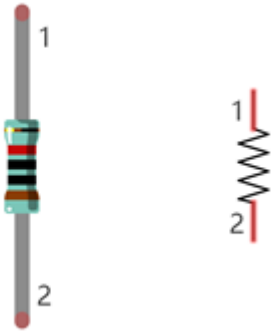
LED	Voltage	Maximum current	Recommended current
Red	1.9-2.2V	20mA	10mA
Green	2.9-3.4V	10mA	5mA
Blue	2.9-3.4V	10mA	5mA
Volt ampere characteristics conform to diode			

Note: LEDs cannot be directly connected to a power supply, which usually ends in a damaged component. A resistor with a specified resistance value must be connected in series to the LED you plan to use.

Resistor

Resistors use Ohms (Ω) as the unit of measurement of their resistance (R). 1MΩ=1000kΩ, 1kΩ=1000Ω.

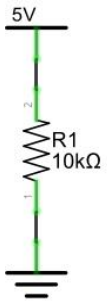
A resistor is a passive electrical component that limits or regulates the flow of current in an electronic circuit. On the left, we see a physical representation of a resistor, and the right is the symbol used to represent the presence of a resistor in a circuit diagram or schematic.



The bands of color on a resistor is a shorthand code used to identify its resistance value. For more details of resistor color codes, please refer to the appendix of this tutorial.

With a fixed voltage, there will be less current output with greater resistance added to the circuit. The relationship between Current, Voltage and Resistance can be expressed by this formula: I=V/R known as Ohm's Law where I = Current, V = Voltage and R = Resistance. Knowing the values of any two of these allows you to solve the value of the third.

In the following diagram, the current through R1 is: I=U/R=5V/10kΩ=0.0005A=0.5mA.

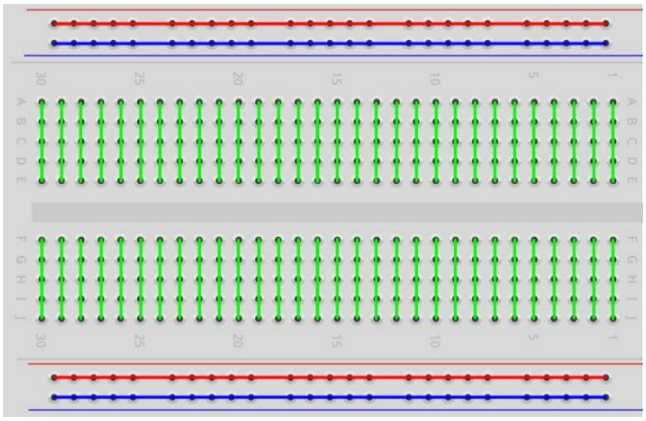


WARNING: Never connect the two poles of a power supply with anything of low resistance value (i.e. a metal object or bare wire) this is a Short and results in high current that may damage the power supply and electronic components.

Note: Unlike LEDs and diodes, resistors have no poles and re non-polar (it does not matter which direction you insert them into a circuit, it will work the same)

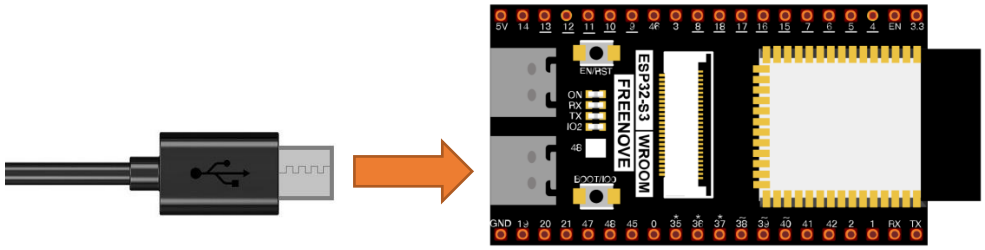
Breadboard

Here we have a small breadboard as an example of how the rows of holes (sockets) are electrically attached. The left picture shows the way to connect pins. The right picture shows the practical internal structure.



Power

ESP32-S3 WROOM needs 5v power supply. In this tutorial, we need connect ESP32-S3 WROOM to computer via USB cable to power it and program it. We can also use other 5v power source to power it.



In the following projects, we only use USB cable to power ESP32-S3 WROOM by default.

In the whole tutorial, we don't use T extension to power ESP32-S3 WROOM. So 5V and 3.3V (includeing EXT 3.3V) on the extension board are provided by ESP32-S3 WROOM.

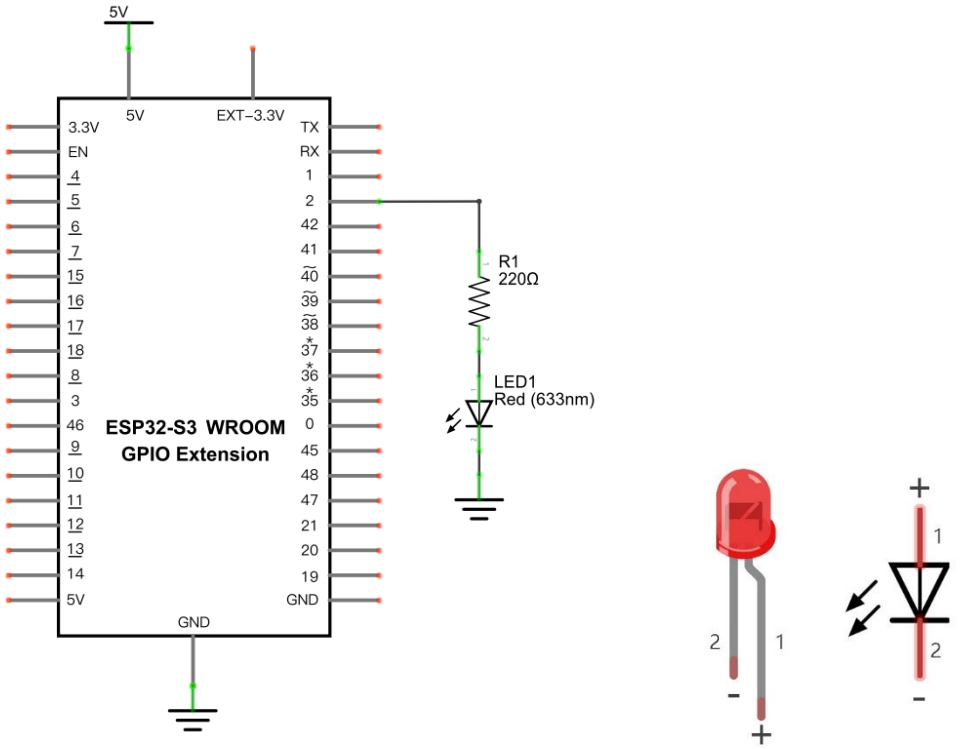
We can also use DC jack of extension board to power ESP32-S3 WROOM.In this way, 5v and EXT 3.3v on extension board are provided by external power resource.

Circuit

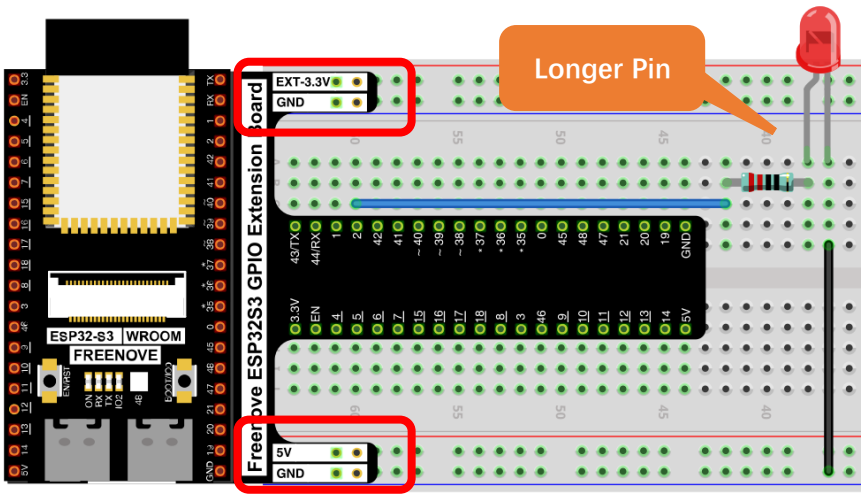
First, disconnect all power from the ESP32-S3 WROOM. Then build the circuit according to the circuit and hardware diagrams. After the circuit is built and verified correct, connect the PC to ESP32-S3 WROOM.

CAUTION: Avoid any possible short circuits (especially connecting 5V or GND, 3.3V and GND)! WARNING: A short circuit can cause high current in your circuit, generate excessive component heat and cause permanent damage to your hardware!

Schematic diagram



Hardware connection. **If you need any support, please contact us via: support@freenove.com**



Don't rotate ESP32-S3 WROOM 180° for connection.

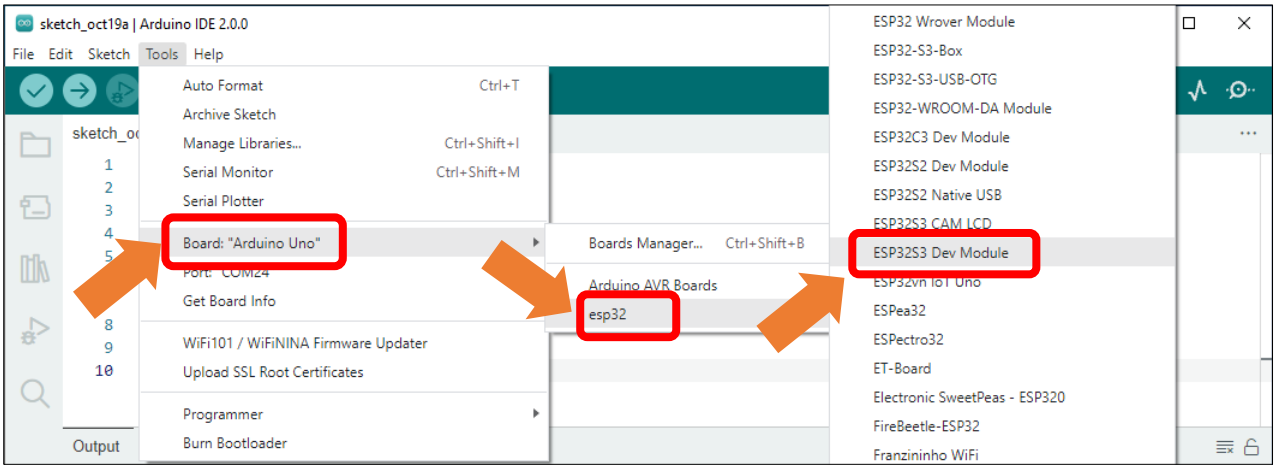
Sketch

According to the circuit, when the GPIO2 of ESP32-S3 WROOM output level is high, the LED turns ON. Conversely, when the GPIO2 ESP32-S3 WROOM output level is low, the LED turns OFF. Therefore, we can let GPIO2 circularly output high and low level to make the LED blink.

Upload the following Sketch:

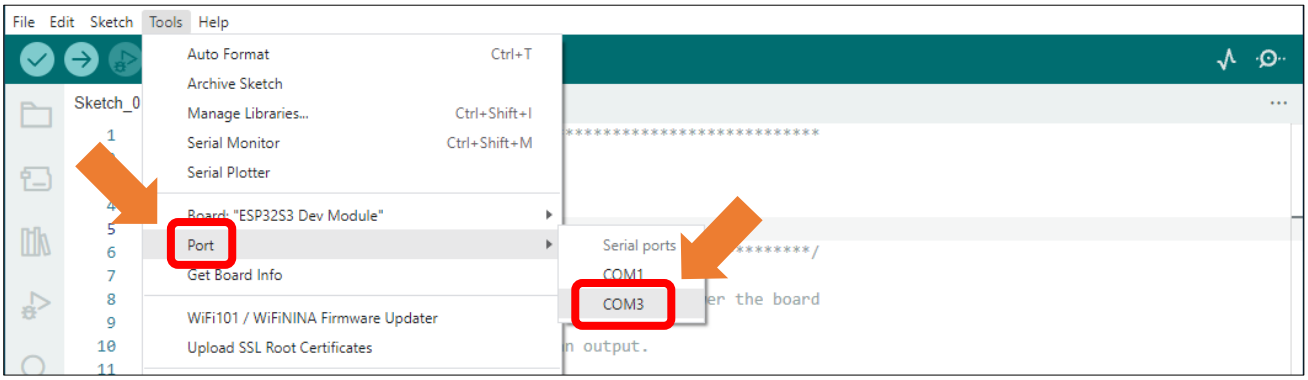
Freenove_Super_Starter_Kit_for_ESP32_S3\Sketches\Sketch_01.1_Blink.

Before uploading the code, click "**Tools**", "**Board**" and select "**ESP32S3 Dev Module** ".

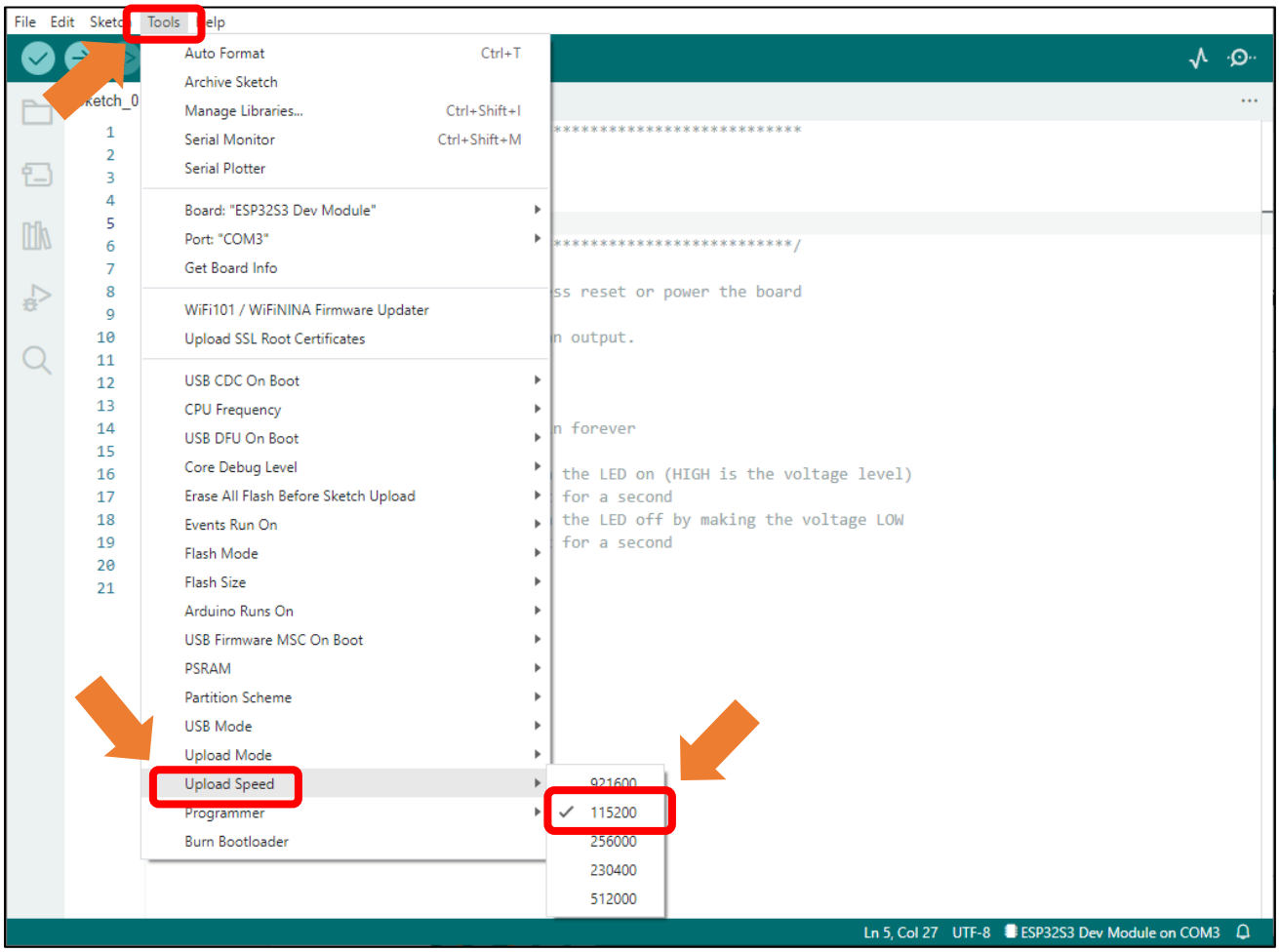


Select the serial port.

Note that the computer port number of each user may be different. Please select the correct serial port according to your computer. Taking the window system as an example, my computer recognizes that the communication interface of the ESP32-S3-WROOM is COM3, so I select COM3.

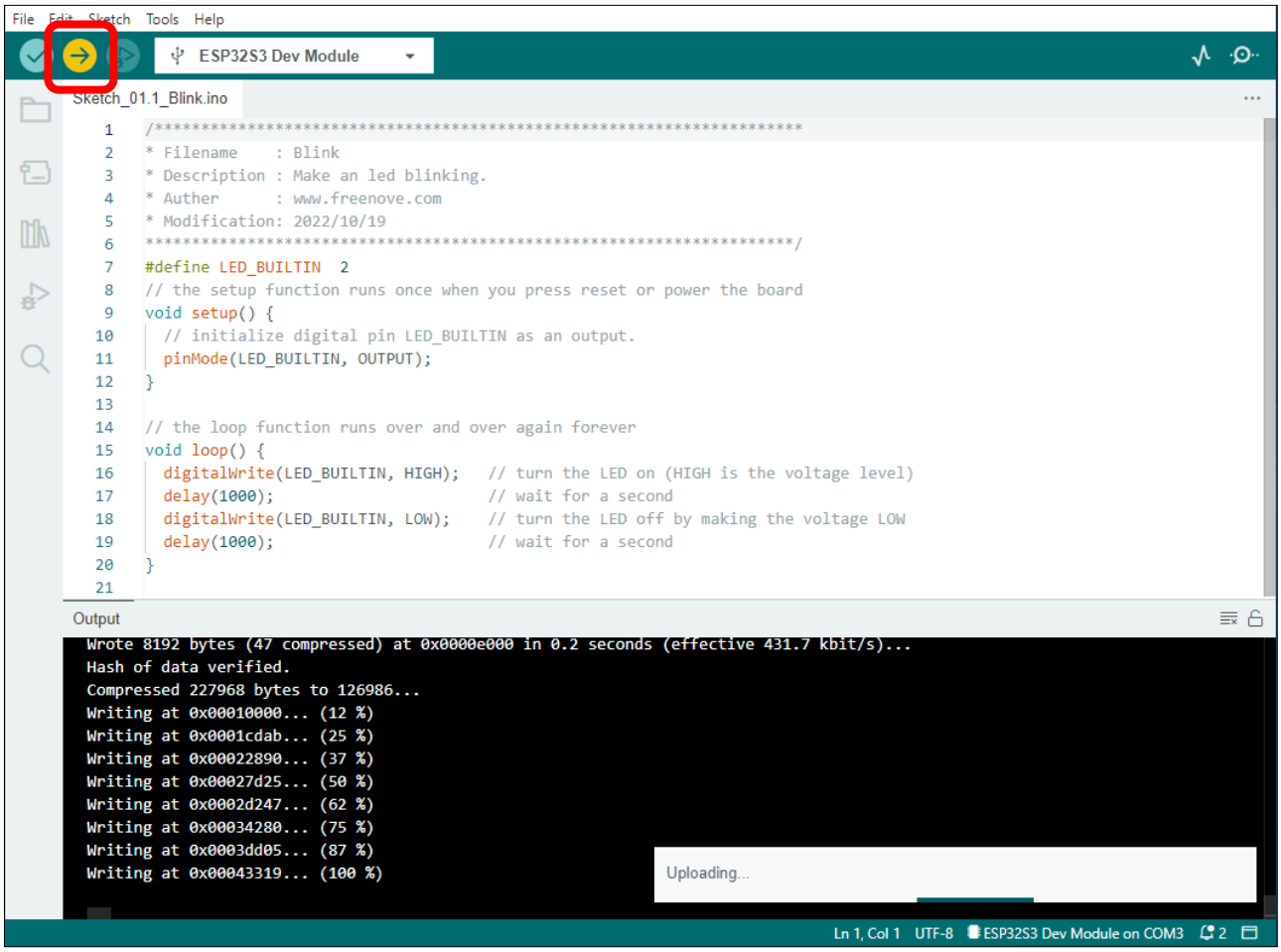


Note: For macOS users, if the uploading fails, please set the baud rate to 115200 before clicking "Upload Using Programmer".

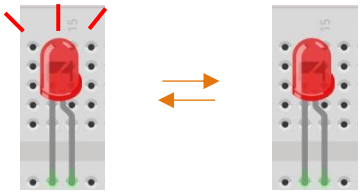


Sketch_01.1_Blink

Click the Upload button and it will compile and upload the Sketch to the ESP32-S3-WROOM.



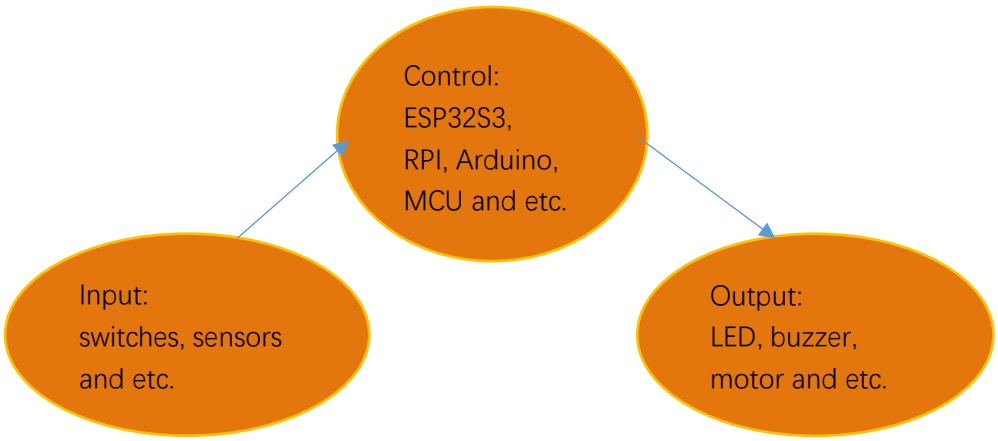
Wait for the Sketch upload to complete, and observe the ESP32-S3 WROOM. You can see that the LED on breadboard flashes cyclically.



If you have any concerns, please contact us via: support@freenove.com

Chapter 2 Button & LED

Usually, there are three essential parts in a complete automatic control device: INPUT, OUTPUT, and CONTROL. In last section, the LED module was the output part and ESP32-S3 was the control part. In practical applications, we not only make LEDs flash, but also make a device sense the surrounding environment, receive instructions and then take the appropriate action such as LEDs light up, make a buzzer turn ON and so on.

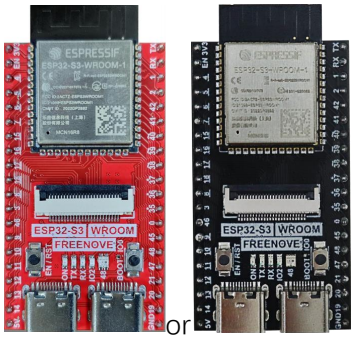
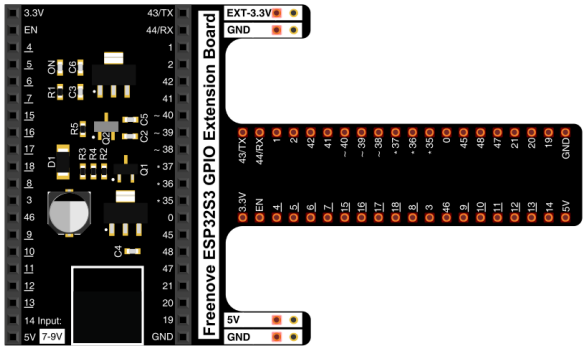
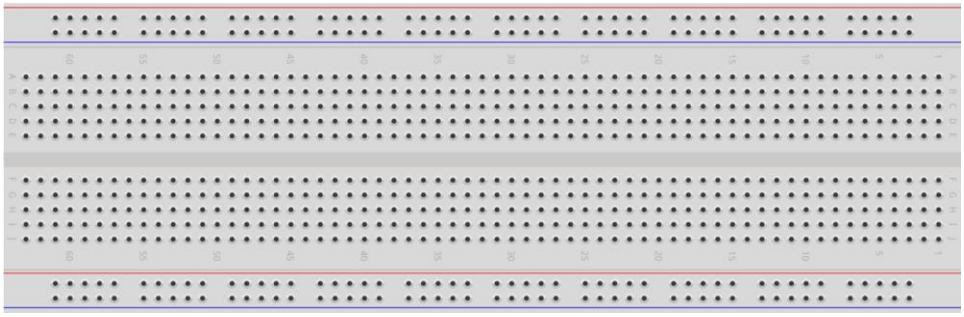


Next, we will build a simple control system to control a LED through a push button switch.

Project 2.1 Button & LED

In the project, we will control the LED state through a Push Button Switch. When the button is pressed, our LED will turn ON, and when it is released, the LED will turn OFF.

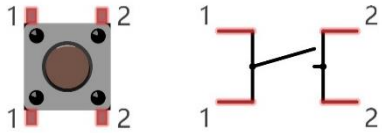
Component List

ESP32-S3(N16R8)/ESP32-S3(N8R8) 	GPIO Extension Board x1 			
Breadboard x1 				
Jumper M/M x4 	LED x1 	Resistor 220Ω x1 	Resistor 10kΩ x2 	Push button x1 

Component knowledge

Push button

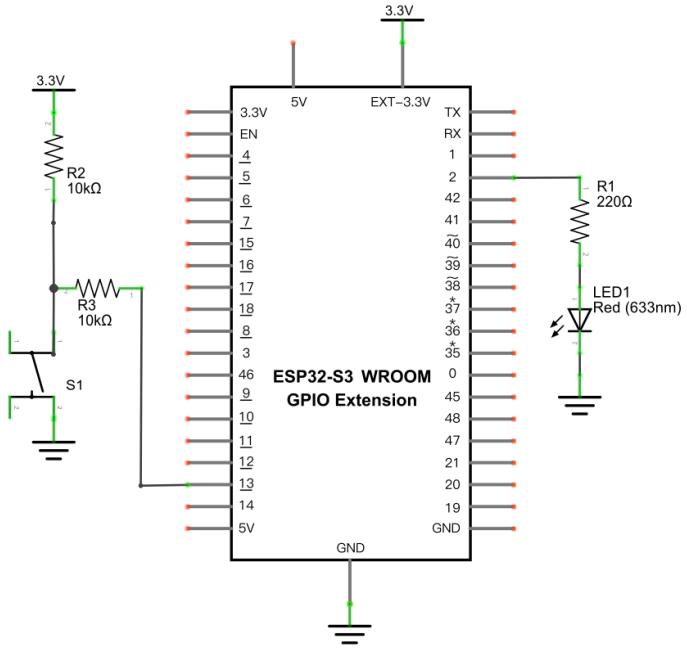
This type of push button switch has 4 pins (2 Pole Switch). Two pins on the left are connected, and both left and right sides are the same per the illustration:



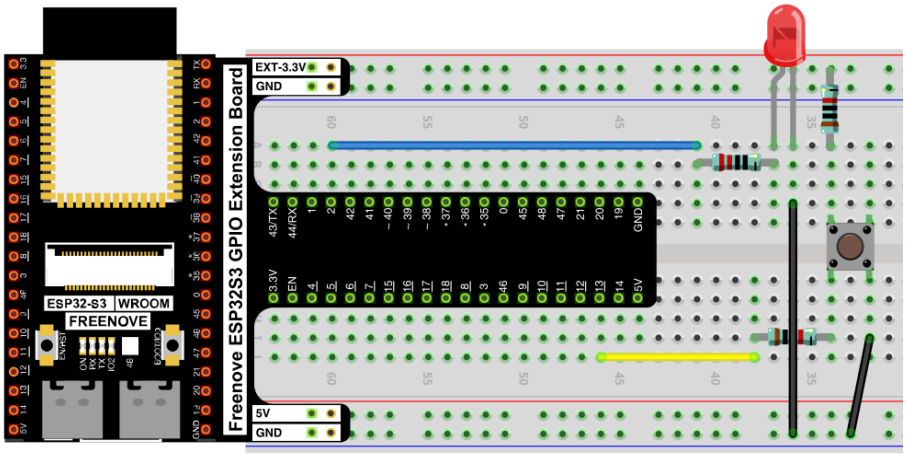
When the button on the switch is pressed, the circuit is completed (your project is powered ON).

Circuit

Schematic diagram



Hardware connection. If you need any support, please feel free to contact us via: support@freenove.com

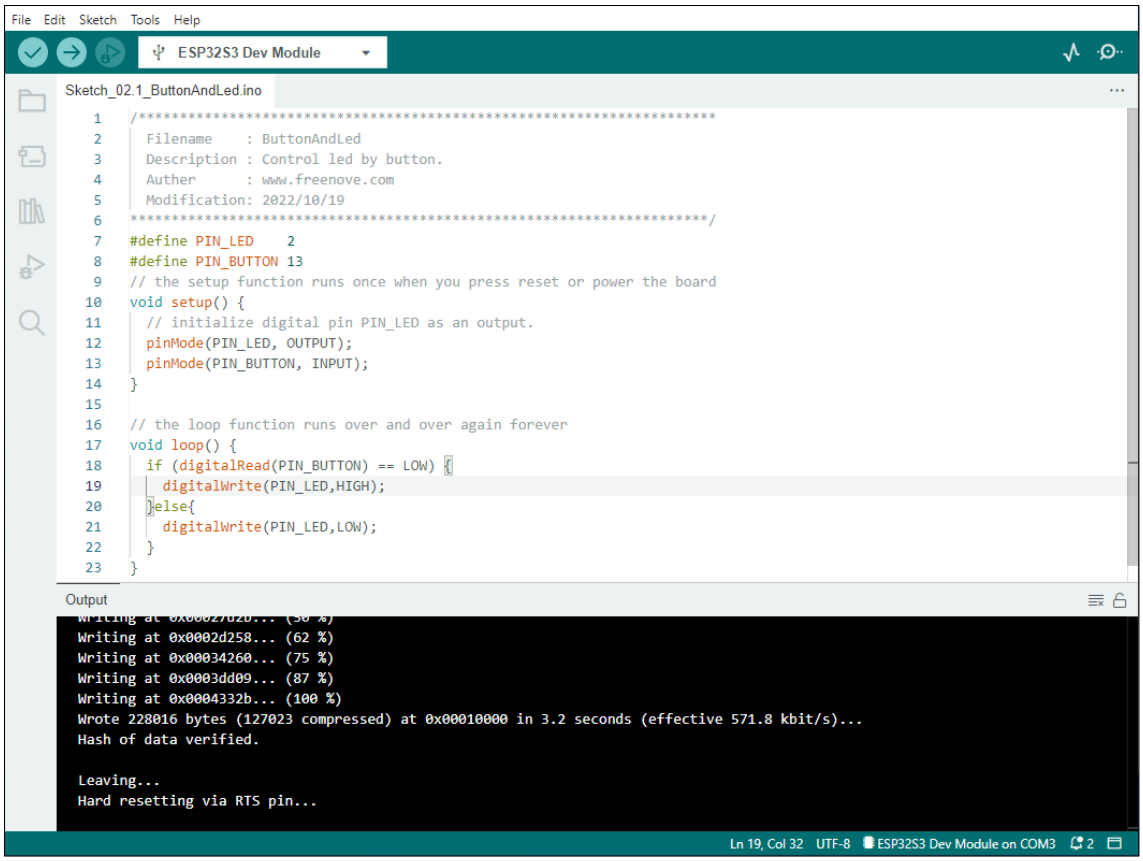


Sketch

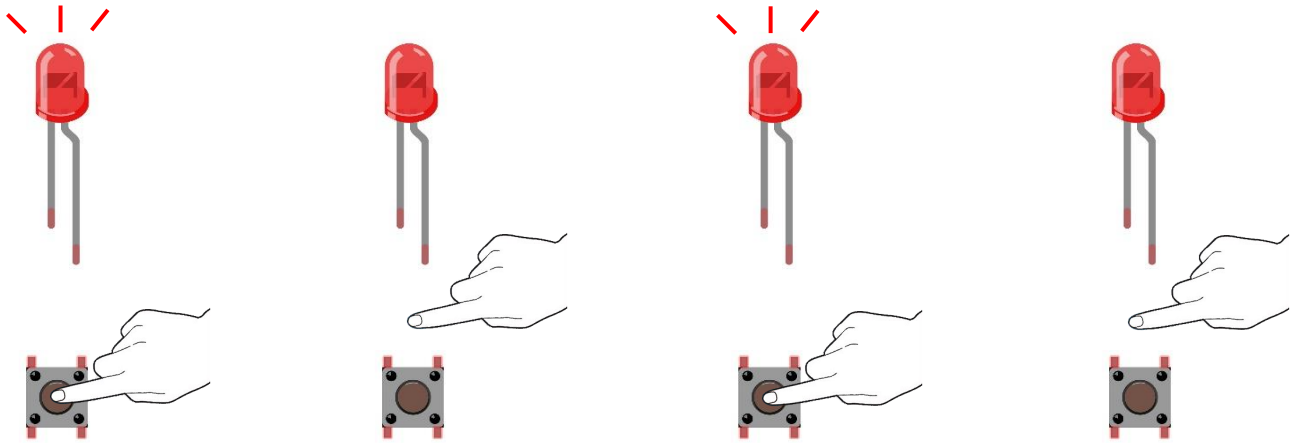
This project is designed for learning how to use push button switch to control a LED. We first need to read the state of switch, and then determine whether to turn the LED ON in accordance to the state of the switch. Upload following sketch:

Freenove_Super_Starter_Kit_for_ESP32_S3\Sketches\Sketch_02.1_ButtonAndLed.

Sketch_02.1_ButtonAndLed



Download the code to ESP32-S3 WROOM, then press the key, the LED turns ON, release the switch, the LED turns OFF.



If you have any concerns, please contact us via: support@freenove.com

The following is the program code:

```
#define PIN_LED    2
#define PIN_BUTTON 13
// the setup function runs once when you press reset or power the board
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) {
    digitalWrite(PIN_LED,HIGH);
  }else{
    digitalWrite(PIN_LED,LOW);
  }
}
```

In the circuit connection, LED and button are connected with GPIO2 and GPIO13 respectively, so define ledPin and buttonPin as 2 and 13 respectively.

```
#define PIN_LED    2
#define PIN_BUTTON 13
```

In the while cycle of main function, use digitalRead(buttonPin) to determine the state of button. When the button is pressed, the function returns low level, the result of “if” is true, and then turn on LED. Otherwise, turn off LED.

```
void loop() {
  if (digitalRead(PIN_BUTTON) == LOW) {
    digitalWrite(PIN_LED,HIGH);
  }else{
    digitalWrite(PIN_LED,LOW);
  }
}
```

Reference

int digitalRead (int pin);

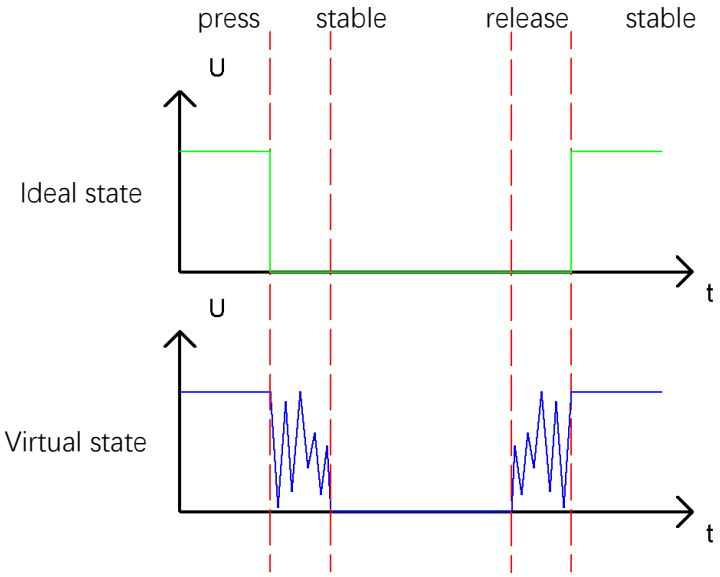
This function returns the value read at the given pin. It will be “HIGH” or “LOW”(1 or 0) depending on the logic level at the pin.

Project 2.2 MINI table lamp

We will also use a push button switch, LED and ESP32-S3 to make a MINI table lamp but this will function differently: Press the button, the LED will turn ON, and pressing the button again, the LED turns OFF. The ON switch action is no longer momentary (like a door bell) but remains ON without needing to continually press on the Button Switch.
First, let us learn something about the push button switch.

Debounce for Push Button

The moment when a push button switch is pressed, it will not change from one state to another state immediately. Due to tiny mechanical vibrations, there will be a short period of continuous buffeting before it completely reaches another state too fast for humans to detect but not for computer microcontrollers. The same is true when the push button switch is released. This unwanted phenomenon is known as “bounce”.



Therefore, if we can directly detect the state of the push button switch, there are multiple pressing and releasing actions in one pressing cycle. This buffeting will mislead the high-speed operation of the microcontroller to cause many false decisions. Therefore, we need to eliminate the impact of buffeting. Our solution: to judge the state of the button multiple times. Only when the button state is stable (consistent) over a period of time, can it indicate that the button is actually in the ON state (being pressed).
This project needs the same components and circuits as we used in the previous section.

Sketch

Sketch_02.2_Tablelamp

File Edit Sketch Tools Help

ESP32S3 Dev Module

Sketch_02.2_TableLamp.ino

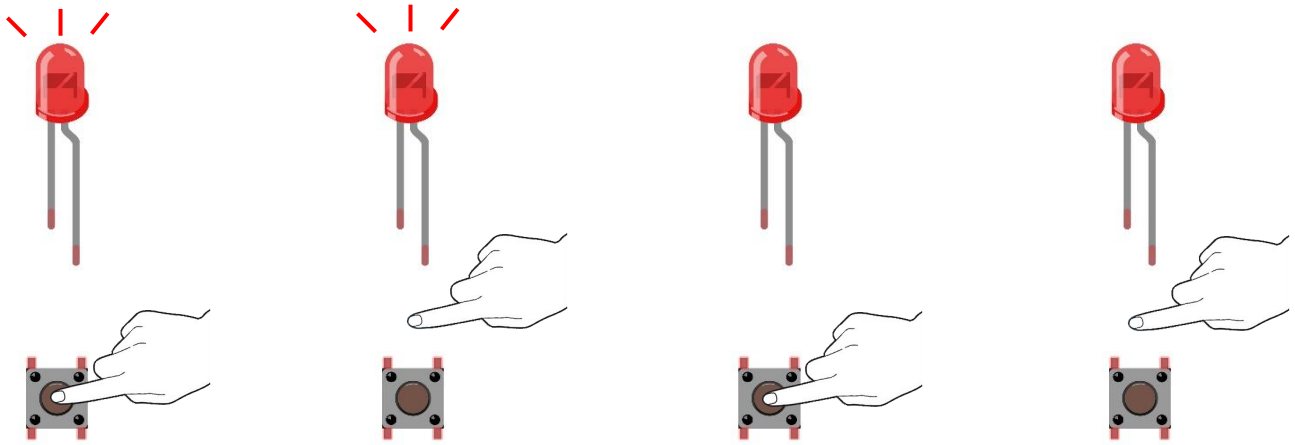
```
#define PIN_LED    2
#define PIN_BUTTON 13
// the setup function runs once when you press reset or power the board
void setup() {
  // initialize digital pin PIN_LED as an output.
  pinMode(PIN_LED, OUTPUT);
  pinMode(PIN_BUTTON, INPUT);
}

// 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);
    delay(20);
    //while (digitalRead(PIN_BUTTON) == LOW);
  }
}

void reverseGPIO(int pin) {
  digitalWrite(pin, !digitalRead(pin));
}

```

Output

```
Writing at 0x0002d27e... (62 %)
Writing at 0x00034258... (75 %)
Writing at 0x0003dd27... (87 %)
Writing at 0x0004335b... (100 %)
Wrote 228112 bytes (127069 compressed) at 0x00010000 in 3.2 seconds (effective 571.4 kbit/s)...
Hash of data verified.

Leaving...
Hard resetting via RTS pin...
```

Ln 5, Col 27 UTF-8 ESP32S3 Dev Module on COM3 2

Download the code to the ESP32-S3 WROOM, press the button, the LED turns ON, and press the button again, the LED turns OFF.



If you have any concerns, please contact us via: support@freenove.com

The following is the program code:

```
#define PIN_LED    2
#define PIN_BUTTON 13
// the setup function runs once when you press reset or power the board
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);
    delay(20);
    while (digitalRead(PIN_BUTTON) == LOW);
  }
}

void reverseGPIO(int pin) {
  digitalWrite(pin, ! digitalRead(pin));
}
```

When judging the push button state, if it is detected as "pressed down", wait for a certain time to detect again to eliminate the effect of bounce. When confirmed, flip the LED on and off. Then it starts to wait for the pressed button to be released, and waits for a certain time to eliminate the effect of bounce after it is released.

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

The subfunction reverseGPIO() means reading the state value of the specified pin, taking the value back and writing it to the pin again to achieve the function of flipping the output state of the pin.

```
void reverseGPIO(int pin) {
  digitalWrite(pin, ! digitalRead(pin));
}
```

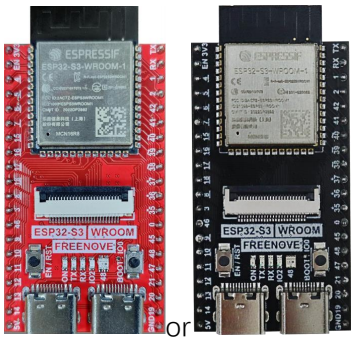
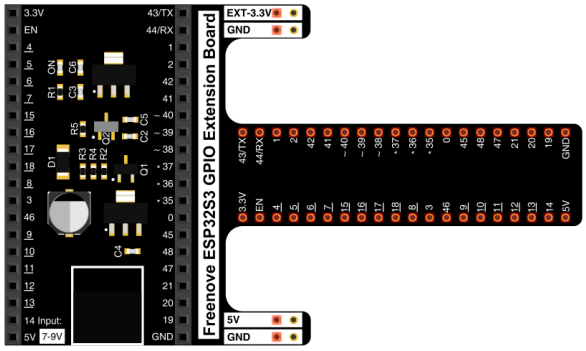
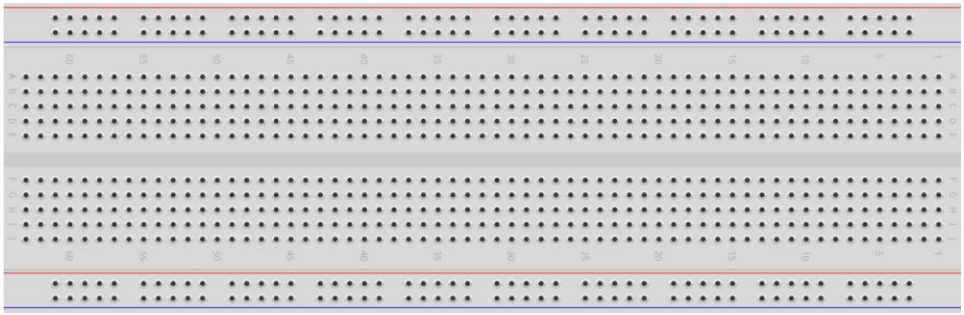
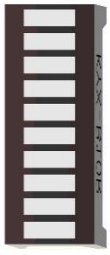
Chapter 3 LED Bar

We have learned how to control a LED blinking, next we will learn how to control a number of LEDs.

Project 3.1 Flowing Light

In this project, we use a number of LEDs to make a flowing light.

Component List

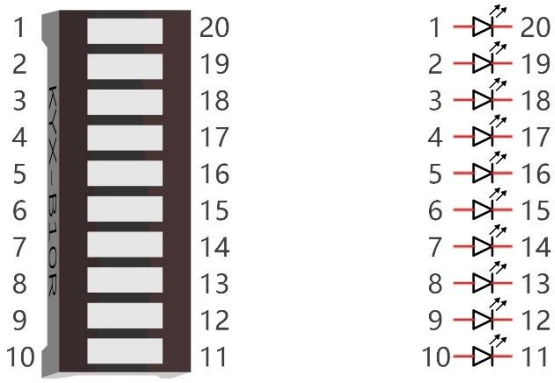
<table>
<tr><td>ESP32-S3(N16R8)/ESP32-S3(N8R8)
 
or</td><td>GPIO Extension Board x1
  </td></tr>
<tr><td colspan="2">Breadboard x1
  </td></tr>
<tr><td>Jumper M/M x10
  </td><td>LED bar graph x1
  </td><td>Resistor 220Ω x10
  </td></tr>
</table>

Component knowledge

Let's learn about the basic features of these components to use and understand them better.

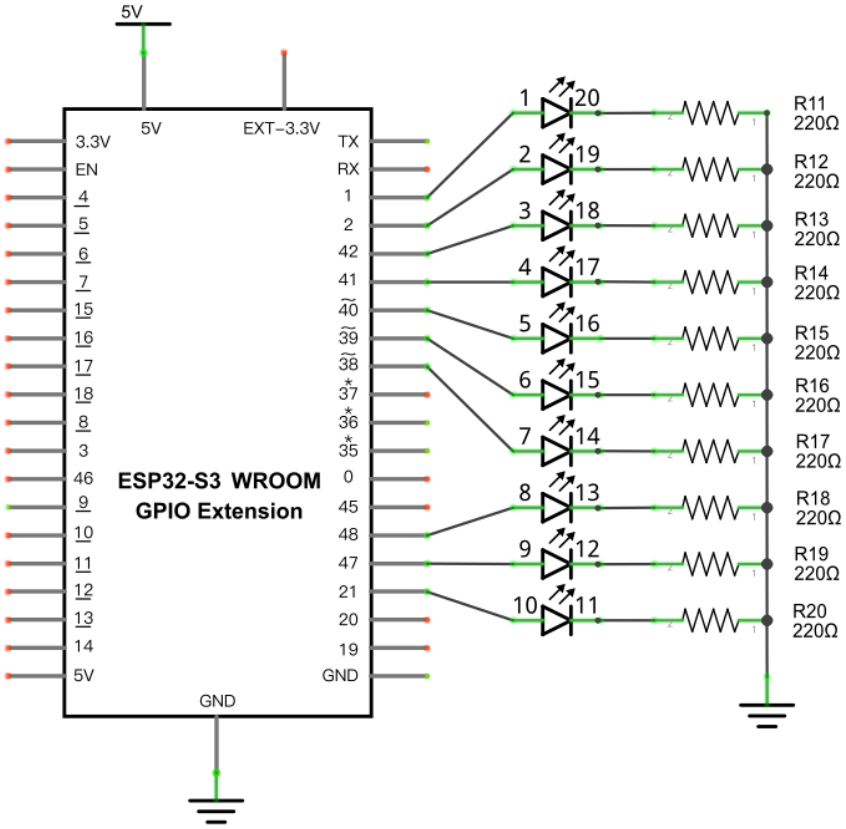
LED bar

A LED bar graph has 10 LEDs integrated into one compact component. The two rows of pins at its bottom are paired to identify each LED like the single LED used earlier.

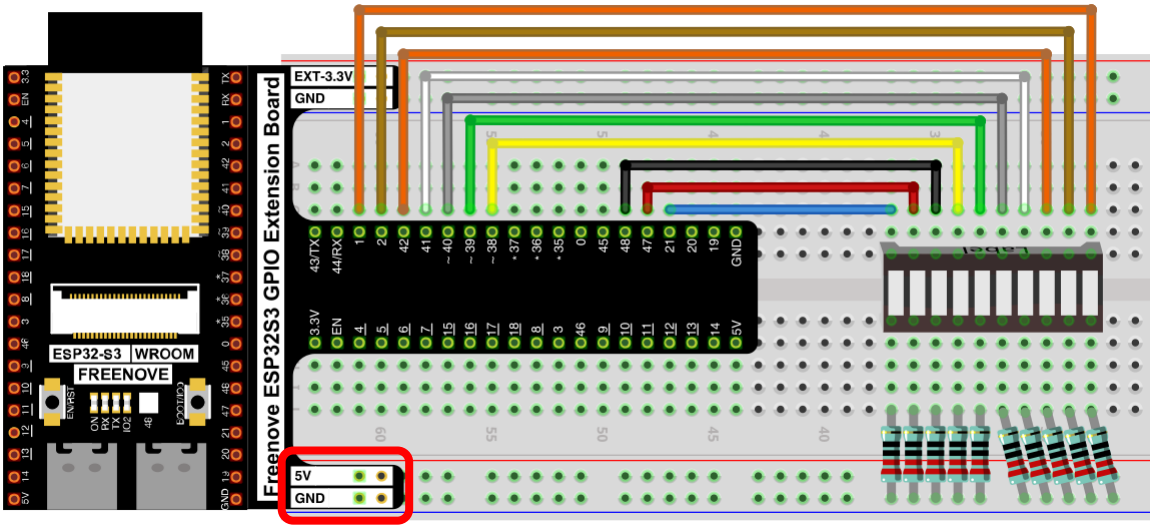


Circuit

Schematic diagram



Hardware connection. If you need any support, please feel free to contact us via: support@freenove.com



If LED bar does not work, try to rotate it for 180°. The label is random.

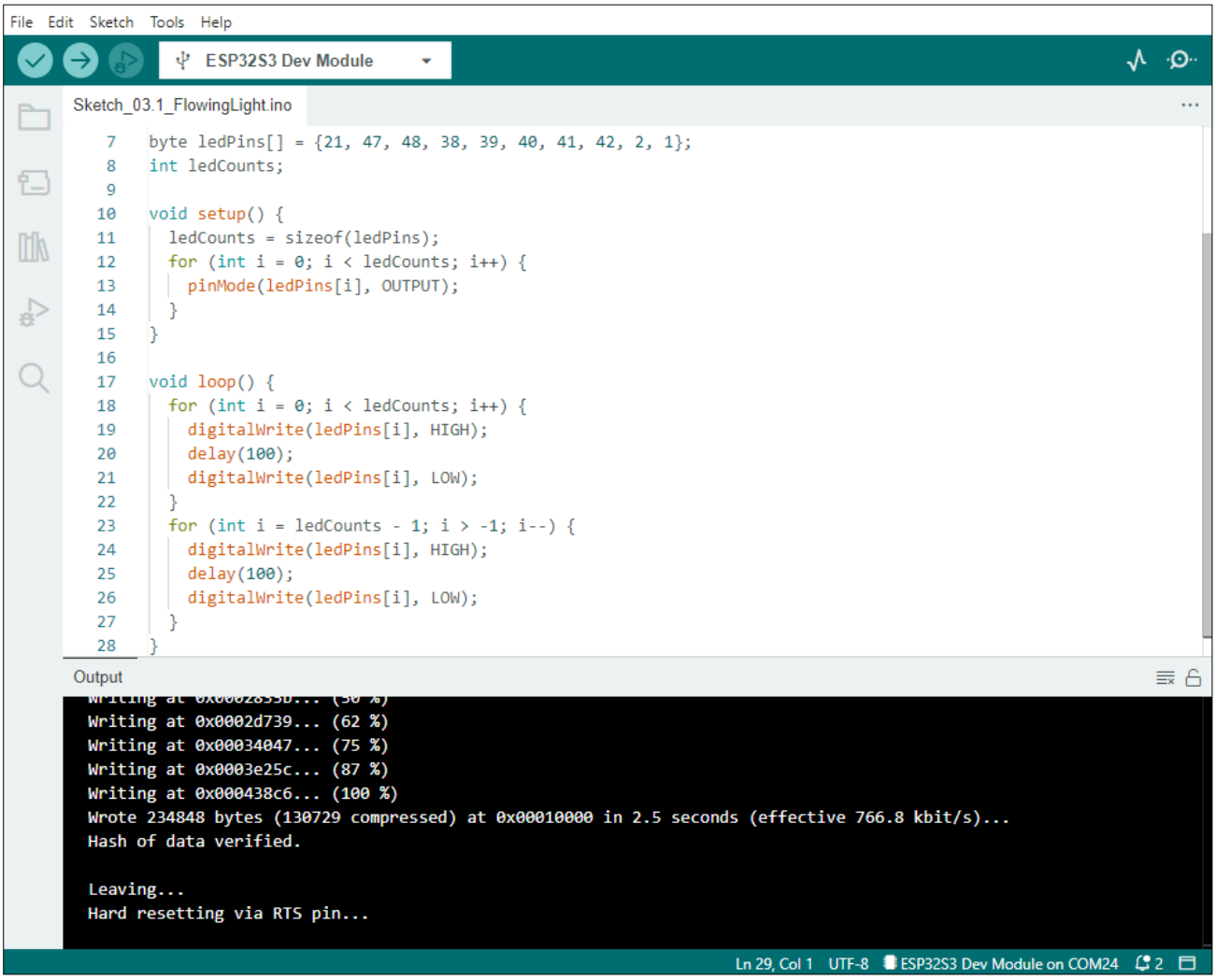
Sketch

This project is designed to make a flowing water lamp. Which are these actions: First turn LED1 ON, then turn it OFF. Then turn LED2 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.

Upload following sketch:

Freenove_Super_Starter_Kit_for_ESP32_S3\Sketches\Sketch_03.1_FlowingLight.

Sketch_03.1_FlowingLight



Download the code to ESP32-S3 WROOM and LED bar graph will light up from left to right and from right to left.

If you have any concerns, please contact us via: support@freenove.com

The following is the program code:

```
byte ledPins[] = {21, 47, 48, 38, 39, 40, 41, 42, 2, 1};
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);
  }
}
```

Use an array to define 10 GPIO ports connected to LED bar graph for easier operation.

```
byte ledPins[] = {21, 47, 48, 38, 39, 40, 41, 42, 2, 1};
```

In setup(), use sizeof() to get the number of array, which is the number of LEDs, then configure the GPIO port to output mode.

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

Then, in loop(), use two “for” loop to realize flowing water light from left to right and from right to left.

```
  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);
  }
```

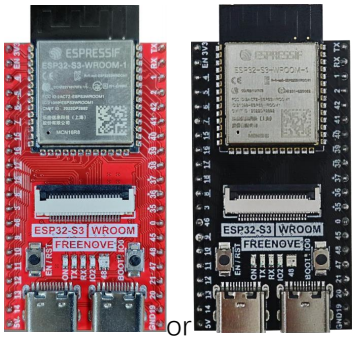
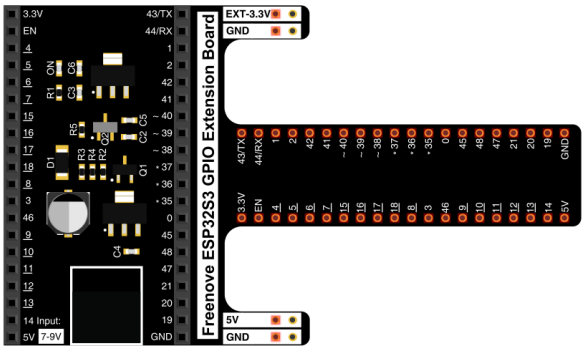
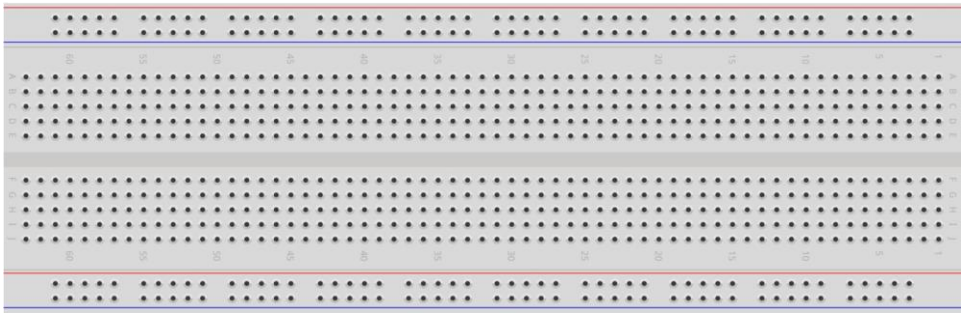
Chapter 4 Analog & PWM

In previous study, we have known that one button has two states: pressed and released, and LED has light-on/off state, then how to enter a middle state? How to output an intermediate state to let LED "semi bright"? That's what we're going to learn.
First, let's learn how to control the brightness of a LED.

Project 4.1 Breathing LED

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 PWM to achieve this target.

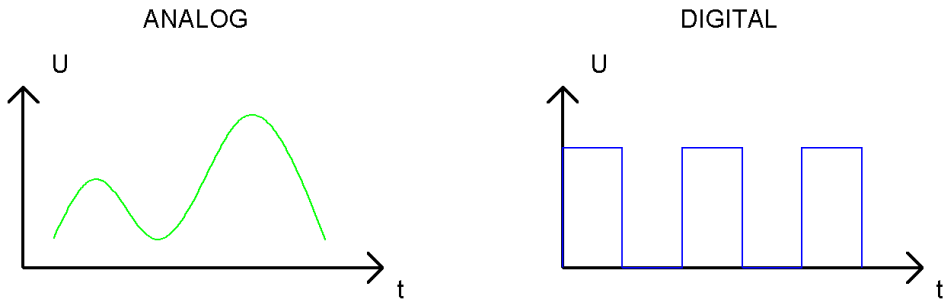
Component List

<table>
<tr><td colspan="2">ESP32-S3(N16R8)/ESP32-S3(N8R8)
 
or</td><td>GPIO Extension Board x1
  </td></tr>
<tr><td colspan="3">Breadboard x1
  </td></tr>
<tr><td>LED x1
  </td><td>Resistor 220Ω x1
  </td><td>Jumper M/M x2
  </td></tr>
</table>

Related 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.



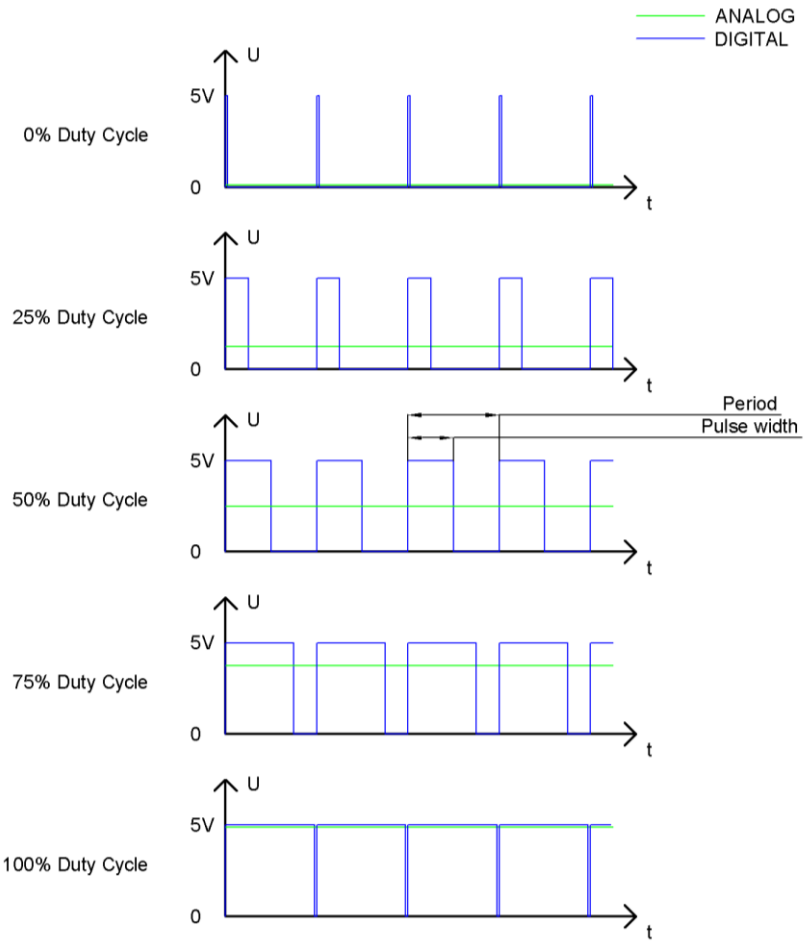
In practical application, we often use binary as the digital signal, that is a series of 0's and 1's. Since a binary signal only has two values (0 or 1), it has great stability and reliability. Lastly, both analog and digital signals can be converted into the other.

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 outputs 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-5V (high level is 5V) 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 a 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. Therefore, we can control the output power of the LED and other output modules to achieve different effects.

ESP32-S3 and PWM

On ESP32-S3, the LEDC(PWM) controller has 8 separate channels, each of which can independently control frequency, duty cycle, and even accuracy. Unlike traditional PWM pins, the PWM output pins of ESP32-S3 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.

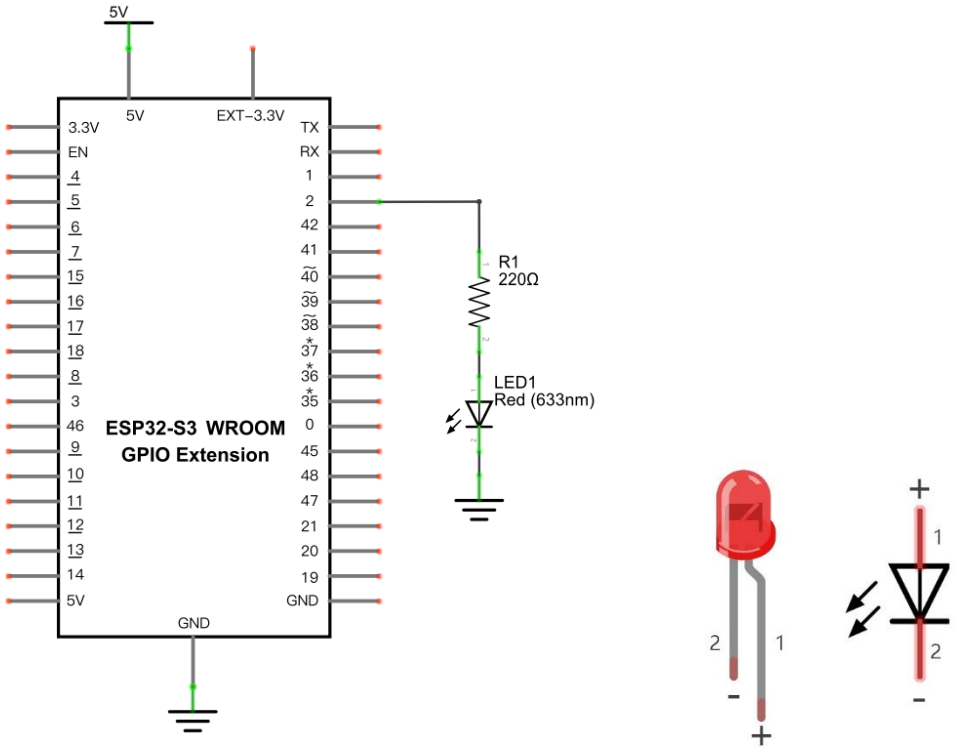
$$Freq_{max} = \frac{80{,}000{,}000}{1 \ll bit}$$

For example, generate a PWM with an 8-bit precision (2^8=256. Values range from 0 to 255) with a maximum frequency of 80,000,000/256 = 312,500Hz.)

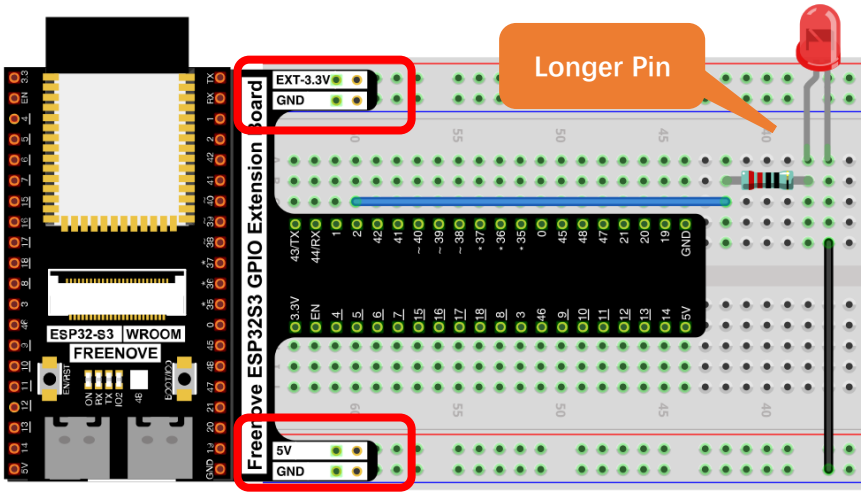
Circuit

This circuit is the same as the one in engineering Blink.

Schematic diagram



Hardware connection. **If you need any support, please contact us via: support@freenove.com**

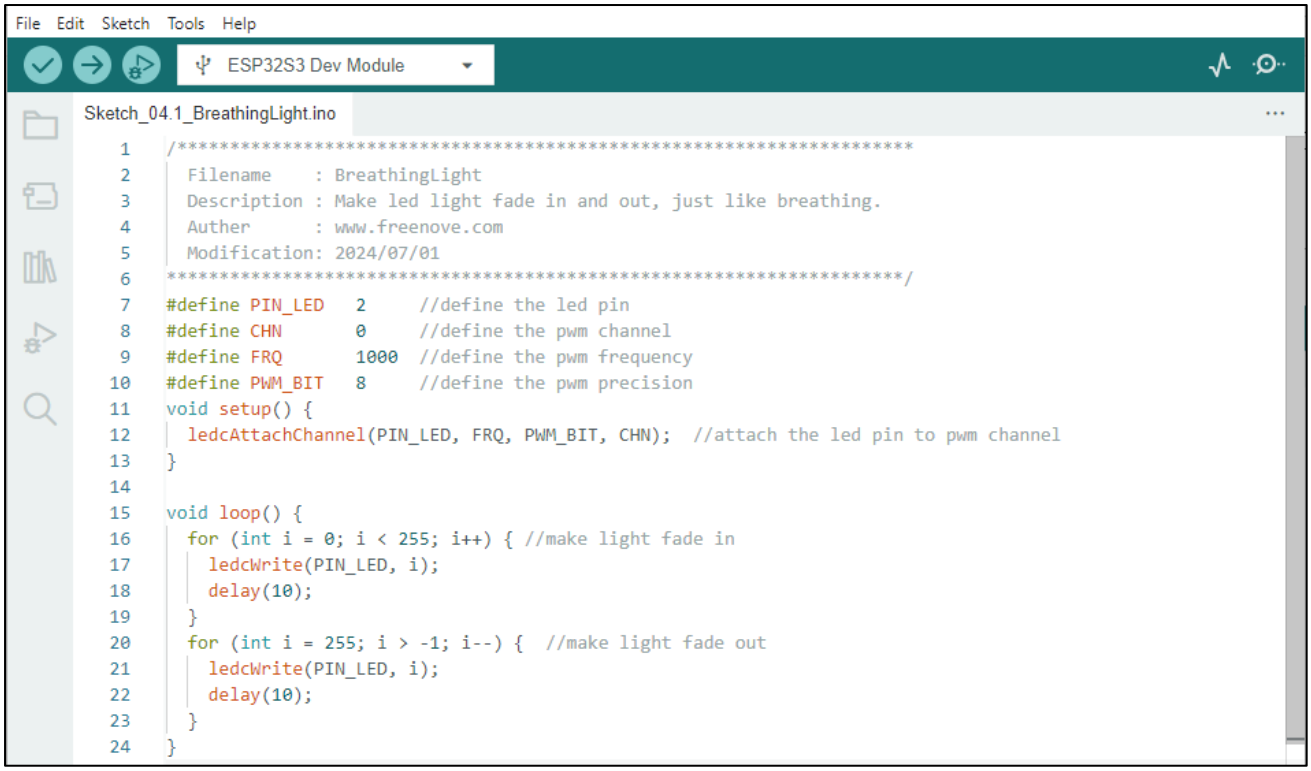


Don't rotate ESP32-S3 WROOM 180° for connection.

Sketch

This project is designed to make PWM output GPIO2 with pulse width increasing from 0% to 100%, and then reducing from 100% to 0% gradually.

Sketch_04.1_BreathingLight



Download the code to ESP32-S3 WROOM, and you'll see that LED is turned from on to off and then from off to on gradually like breathing.

The following is the program code:

```
1  #define PIN_LED   2     //define the led pin
2  #define CHN       0     //define the pwm channel
3  #define FRQ       1000  //define the pwm frequency
4  #define PWM_BIT   8     //define the pwm precision
5  void setup() {
```

```
6     ledcAttachChannel(PIN_LED, FRQ, PWM_BIT, CHN);  //attach the led pin to pwm channel
7   }
8
9   void loop() {
10    for (int i = 0; i < 255; i++) { //make light fade in
11      ledcWrite(PIN_LED, i);
12      delay(10);
13    }
14    for (int i = 255; i > -1; i--) {  //make light fade out
15      ledcWrite(PIN_LED, i);
16      delay(10);
17    }
18  }
```

The PWM pin output mode of ESP32-S3 is not the same as the traditional controller. It controls each parameter of PWM by controlling the PWM channel. Any number of GPIO can be connected with the PWM channel to output PWM. In setup(), you first configure a PWM channel and set the frequency and precision. Then the GPIO is associated with the PWM channel.

```
6   ledcAttachChannel(PIN_LED, FRQ, PWM_BIT, CHN);  //attach the led pin to pwm channel
```

In the loop(), There are two "for" loops. The first makes the ledPin output PWM from 0% to 100% and the second makes the ledPin output PWM from 100% to 0%. This allows the LED to gradually light and extinguish。

```
11    for (int i = 0; i < 255; i++) {   //make light fade in
12      ledcWrite(PIN_LED, i);
13      delay(10);
14    }
15    for (int i = 255; i > -1; i--) {  //make light fade out
16      ledcWrite(PIN_LED, i);
17      delay(10);
18    }
```

You can also adjust the rate of the state change of LED by changing the parameters of the delay() function in the "for" loop.

```
void ledcAttachChannel (uint8_t pin, double freq, uint8_t bit_num, uint8_t channel);
void ledcDetachPin(uint8_t pin);
```

Set the frequency and accuracy of a PWM channel.

Parameters

chan: channel index. Value range :0-7

freq: frequency, it could be a decimal.

bit_num: precision of values.

channel: Bind/unbind a GPIO to a PWM channel.

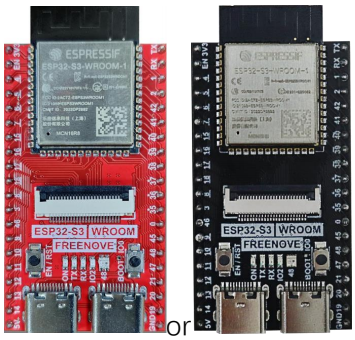
```
void ledcWrite(uint8_t channel, uint32_t duty);
```

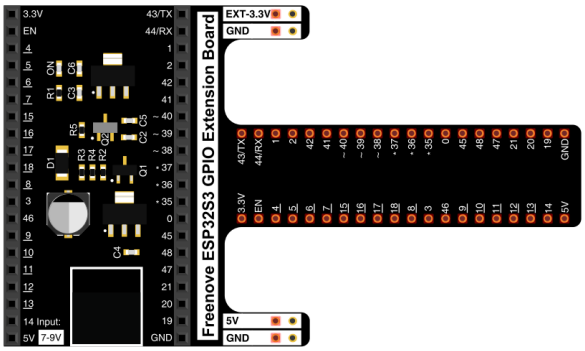
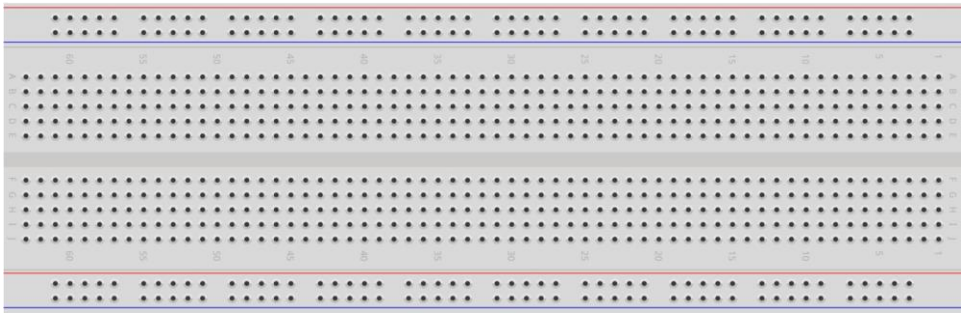
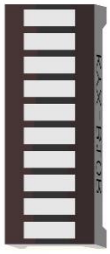
Writes the pulse width value to a PWM channel.

Project 4.2 Meteor Flowing Light

After learning about PWM, we can use it to control LED bar graph and realize a cooler flowing light. The component list, circuit, and hardware are exactly consistent with the project Flowing Light.

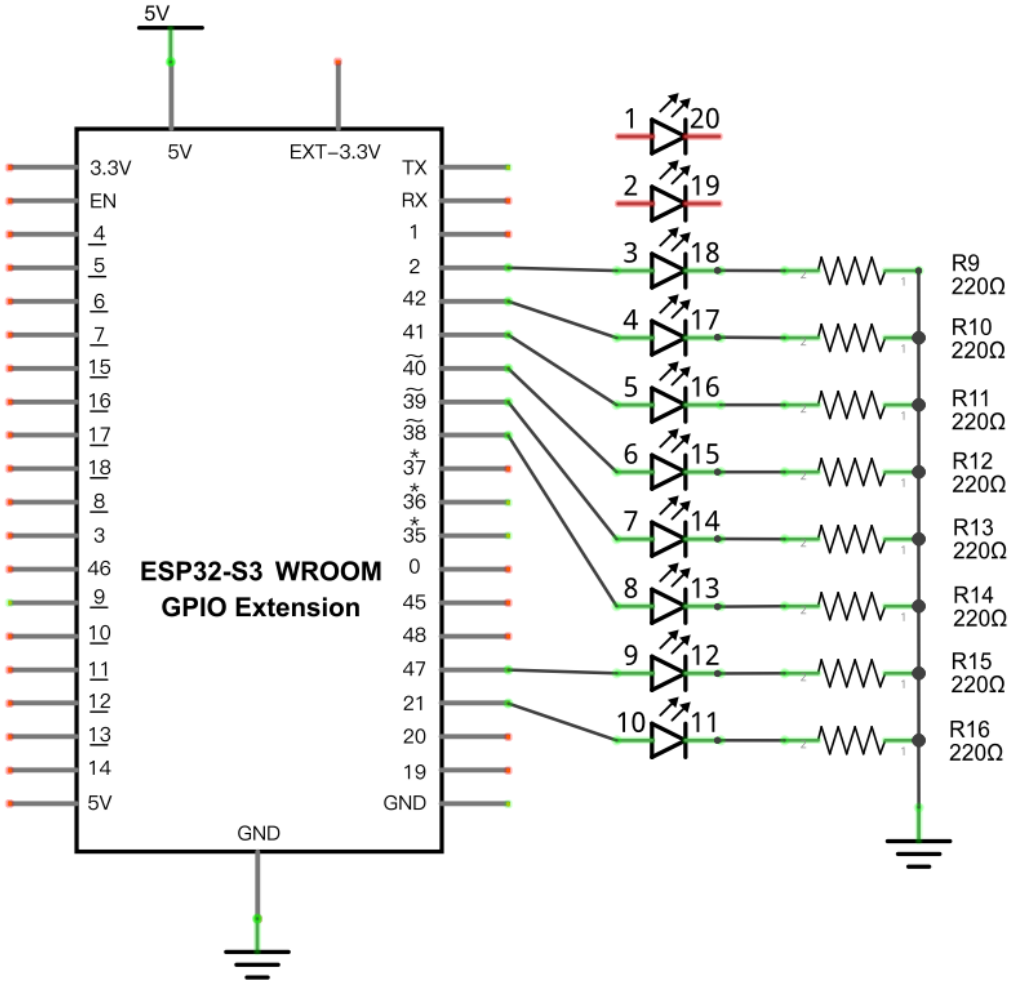
Component List

<table>
<tr><td>ESP32-S3(N16R8)/ESP32-S3(N8R8)
 

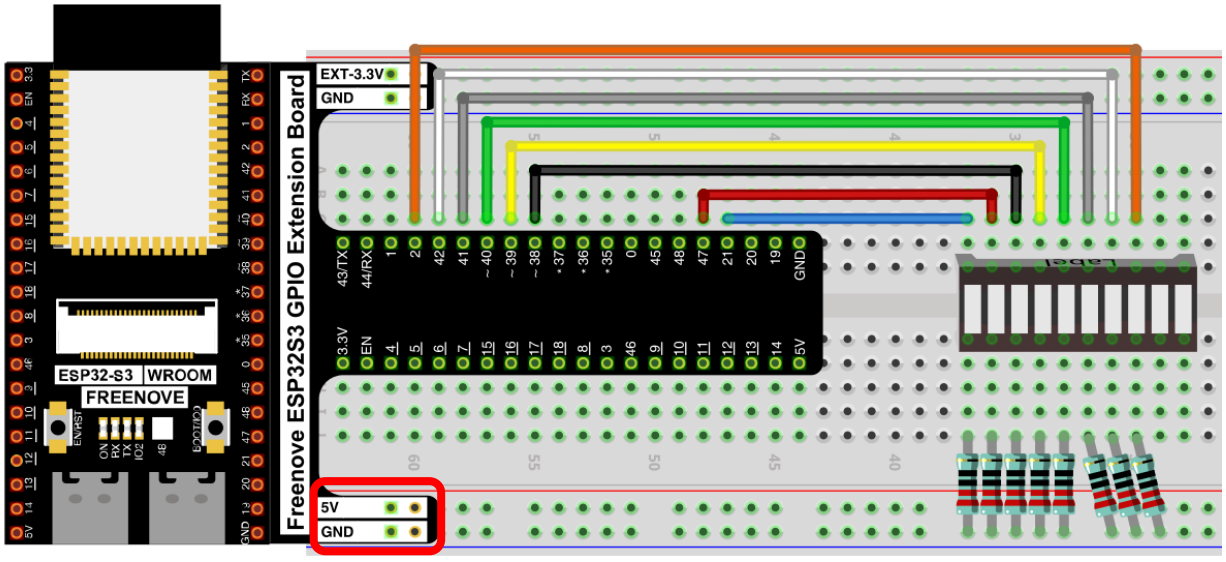
or</td><td>GPIO Extension Board x1
 
</td></tr>
<tr><td colspan="2">Breadboard x1
  </td></tr>
<tr><td>Jumper M/M x8
  </td><td>LED bar graph x1
  </td><td>Resistor 220Ω x8
  </td></tr>
</table>

Circuit

Schematic diagram



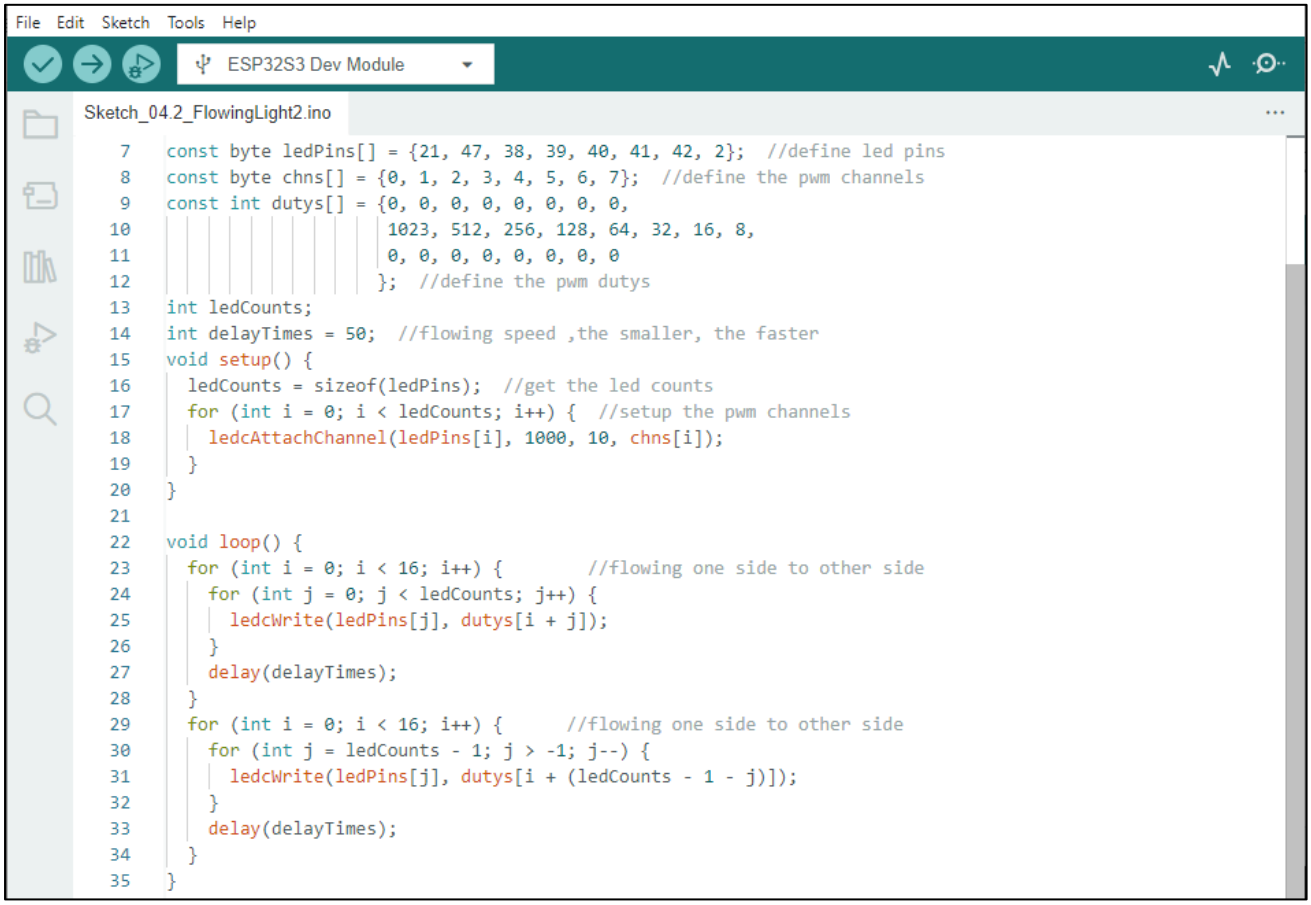
Hardware connection. If you need any support, please feel free to contact us via: support@freenove.com



Sketch

Meteor flowing light will be implemented with PWM.

Sketch_04.2_FlowingLight2



Download the code to ESP32-S3 WROOM, and LED bar graph will gradually light up and out from left to right, then light up and out from right to left.

The following is the program code:

```
1  const byte ledPins[] = {21, 47, 38, 39, 40, 41, 42, 2};    //define led pins
2  const byte chns[] = {0, 1, 2, 3, 4, 5, 6, 7};              //define the pwm channels
3  const int dutys[] = {0, 0, 0, 0, 0, 0, 0, 0,
4                       1023, 512, 256, 128, 64, 32, 16, 8,
5                       0, 0, 0, 0, 0, 0, 0, 0
6                      };      //define the pwm dutys
7  int ledCounts;              //led counts
8  int delayTimes = 50;        //flowing speed ,the smaller, the faster
9  void setup() {
10   ledCounts = sizeof(ledPins);    //get the led counts
11   for (int i = 0; i < ledCounts; i++) {   //setup the pwm channels
12     ledcAttachChannel(ledPins[i], 1000, 10, chns[i]);
13   }
```

```
}

void loop() {
  for (int i = 0; i < 16; i++) {        //flowing one side to other side
    for (int j = 0; j < ledCounts; j++) {
      ledcWrite(ledPins[j], dutys[i + j]);
    }
    delay(delayTimes);
  }
  for (int i = 0; i < 16; i++) {        //flowing one side to other side
    for (int j = ledCounts - 1; j > -1; j--) {
      ledcWrite(ledPins[j], dutys[i + (ledCounts - 1 - j)]);
    }
    delay(delayTimes);
  }
}
```

First we defined 8 GPIO, 8 PWM channels, and 24 pulse width values.

```
const byte ledPins[] = {21, 47, 38, 39, 40, 41, 42, 2};    //define led pins
const byte chns[] = {0, 1, 2, 3, 4, 5, 6, 7};              //define the pwm channels
const int dutys[] = {0, 0, 0, 0, 0, 0, 0, 0,
                     1023, 512, 256, 128, 64, 32, 16, 8,
                     0, 0, 0, 0, 0, 0, 0, 0
                    };      //define the pwm dutys
```

In setup(), set the frequency of 8 PWM channels to 1000Hz, the accuracy to 10bits, and the maximum pulse width to 1023. Attach GPIO to these PWM channels.

```
  for (int i = 0; i < ledCounts; i++) {   //setup the pwm channels
    ledcAttachChannel(ledPins[i], 1000, 10, chns[i]);
  }
```

In loop(), a nested for loop is used to control the pulse width of the PWM, and LED bar graph moves one grid after each 1 is added in the first for loop, gradually changing according to the values in the array duties. As shown in the table below, the value of the second row is the value in the array duties, and the 8 green squares in each row below represent the 8 LEDs on the LED bar graph. Every 1 is added to I, the value of the LED bar graph will move to the right by one grid, and when it reaches the end, it will move from the end to the starting point, achieving the desired effect.

	0	1	2	3	4	5	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
d i	0	0	0	0	0	0	0	0	1023	512	256	128	64	32	16	8	0	0	0	0	0	0	0	0
0																								
1																								
2																								
3																								
...																								
13																								
14																								
15																								

In the code, two nested for loops are used to achieve this effect.

```
17    for (int i = 0; i < 16; i++) {        //flowing one side to other side
18      for (int j = 0; j < ledCounts; j++) {
19        ledcWrite(ledPins[j], dutys[i + j]);
20      }
21      delay(delayTimes);
22    }
23    for (int i = 0; i < 16; i++) {       //flowing from one side to the other
24      for (int j = ledCounts - 1; j > -1; j--) {
25        ledcWrite(ledPins[j], dutys[i + (ledCounts - 1 - j)]);
26      }
27      delay(delayTimes);
28    }
```

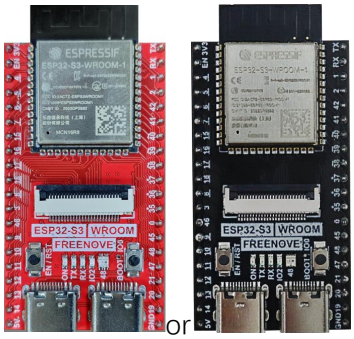
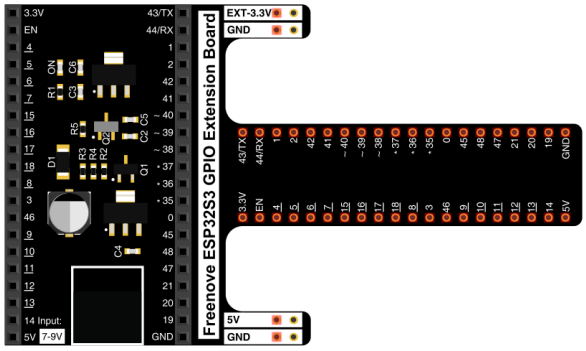
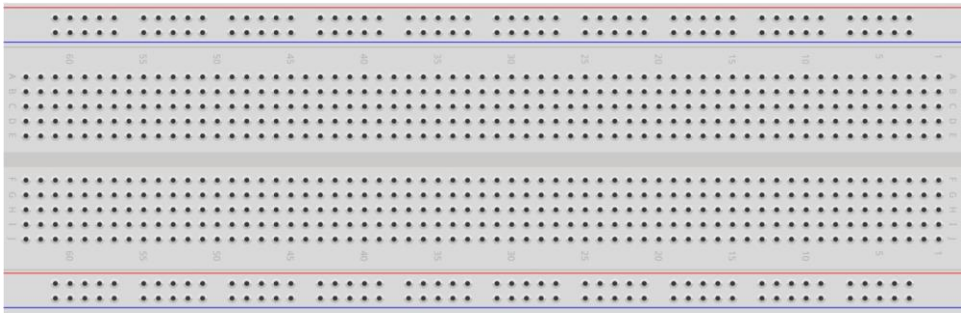
Chapter 5 RGB LED

In this chapter, we will learn how to control a RGB LED. It can emit different colors of light. Next, we will use RGB LED to make a multicolored light.

Project 5.1 Random Color Light

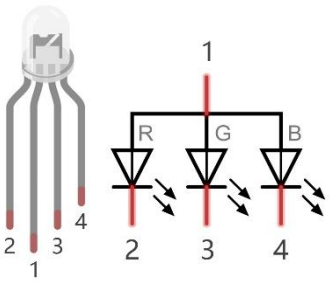
In this project, we will make a multicolored LED. And we can control RGB LED to switch different colors automatically.

Component List

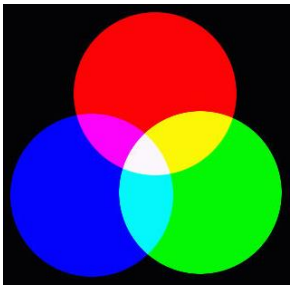
ESP32-S3(N16R8)/ESP32-S3(N8R8) 	GPIO Extension Board x1 
Breadboard x1 	
RGBLED x1 	Resistor 220Ω x3 
Jumper M/M x4 	

Related knowledge

RGB LED has integrated 3 LEDs that can respectively emit red, green and blue light. And it has 4 pins. The long pin (1) is the common port, that is, 3 LED 's positive or negative port. The RGB LED with common positive port and its symbol is shown below. We can make RGB LED emit various colors of light by controlling these 3 LEDs to emit light with different brightness,



Red, green, and blue are known as three primary colors. When you combine these three primary-color lights with different brightness, it can produce almost all kinds of visible lights. Computer screens, single pixel of cell phone screen, neon, and etc. are working under this principle.

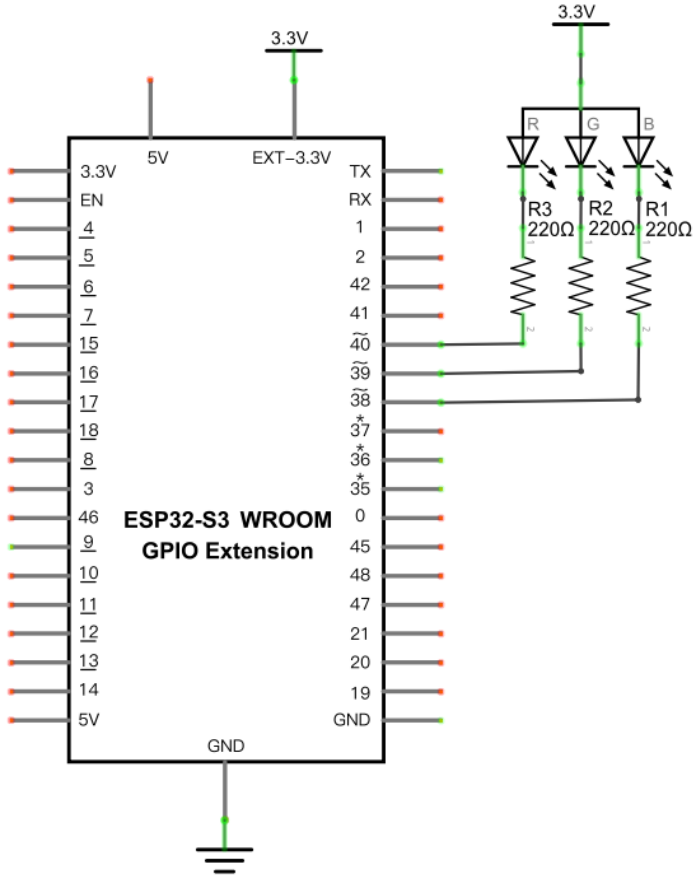


RGB

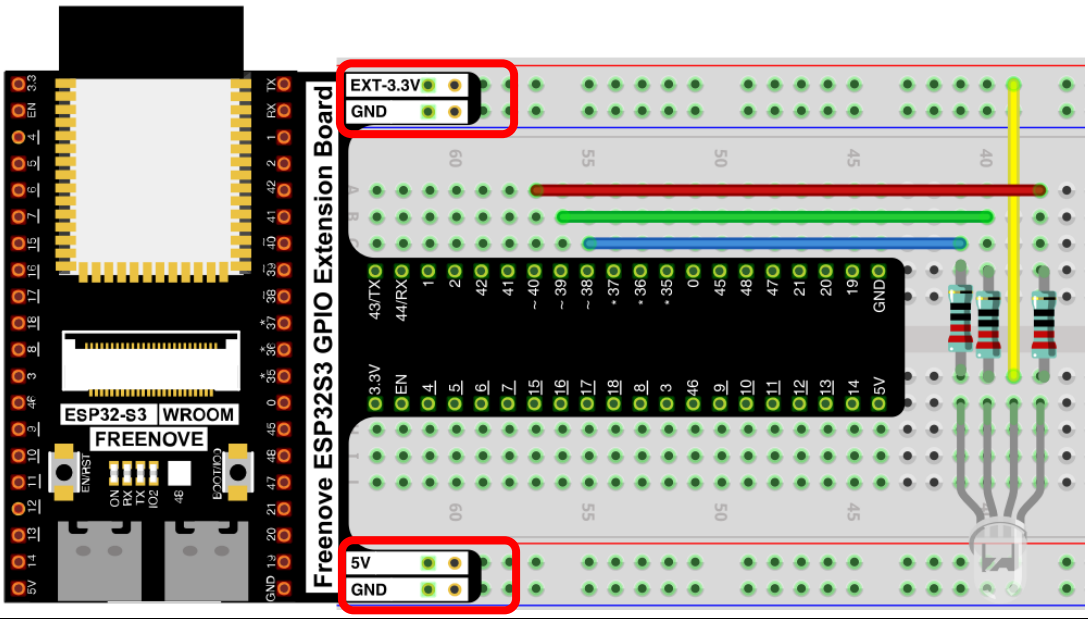
If we use three 8-bit PWMs to control the RGB LED, in theory, we can create $2^8*2^8*2^8$=16777216 (16 million) colors through different combinations.

Circuit

Schematic diagram



Hardware connection. If you need any support, please feel free to contact us via: support@freenove.com



Sketch

We need to create three PWM channels and use random duty cycle to make random RGB LED color.

Sketch_05.1_ColorfulLight

With the code downloaded to ESP32-S3 WROOM, RGB LED begins to display random colors.

If you have any concerns, please contact us via: support@freenove.com

The following is the program code:

```
1  const byte ledPins[] = {38, 39, 40};    //define red, green, blue led pins
2  const byte chns[] = {0, 1, 2};          //define the pwm channels
3  int red, green, blue;
4  void setup() {
5    for (int i = 0; i < 3; i++) {         //setup the pwm channels,1KHz,8bit
6      ledcAttachChannel(ledPins[i], 1000, 8, chns[i]);
7    }
8  }
9
10 void loop() {
11   red = random(0, 256);
12   green = random(0, 256);
13   blue = random(0, 256);
14   setColor(red, green, blue);
```

Any concerns? ✉ support@freenove.com

```
15    delay(200);
16  }
17
18  void setColor(byte r, byte g, byte b) {
19    ledcWrite(ledPins[0], 255 - r); //Common anode LED, low level to turn on the led.
20    ledcWrite(ledPins[1], 255 - g);
21    ledcWrite(ledPins[2], 255 - b);
22  }
```

Define the PWM channel and associate it with the pin connected to RGB LED, and define the variable to hold the color value and initialize it in setup().

```
1   const byte ledPins[] = {38, 39, 40};    //define red, green, blue led pins
2   const byte chns[] = {0, 1, 2};          //define the pwm channels
3   int red, green, blue;
4   void setup() {
5     for (int i = 0; i < 3; i++) {         //setup the pwm channels,1KHz,8bit
6       ledcAttachChannel(ledPins[i], 1000, 8, chns[i]);
7     }
8   }
```

In setColor(), this function controls the output color of RGB LED by the given color value. Because the circuit uses a common anode, the LED lights up when the GPIO outputs low power. Therefore, in PWM, low level is the active level, so 255 minus the given value is necessary.

```
18  void setColor(byte r, byte g, byte b) {
19    ledcWrite(ledPins[0], 255 - r); //Common anode LED, low level to turn on the led.
20    ledcWrite(ledPins[1], 255 - g);
21    ledcWrite(ledPins[2], 255 - b);
22  }
```

In loop(), get three random Numbers and set them as color values.

```
11    red = random(0, 256);
12    green = random(0, 256);
13    blue = random(0, 256);
14    setColor(red, green, blue);
15    delay(200);
```

The related function of software PWM can be described as follows:

long random(min, max);
This function will return a random number(min --- max-1).

Project 5.2 Gradient Color Light

In the previous project, we have mastered the usage of RGB LED, but the random display of colors is rather stiff. This project will realize a fashionable light with soft color changes.

Component list and the circuit are exactly the same as the random color light.

Using a color model, the color changes from 0 to 255 as shown below.

In this code, the color model will be implemented and RGB LED will change colors along the model.

Sketch_05.2_SoftColorfulLight

The following is the program code:

```
const byte ledPins[] = {38, 39, 40};  //define led pins
const byte chns[] = {0, 1, 2};        //define the pwm channels

void setup() {
  for (int i = 0; i < 3; i++) {       //setup the pwm channels
    ledcAttachChannel(ledPins[i], 1000, 8, chns[i]);
  }
}

void loop() {
  for (int i = 0; i < 256; i++) {
    setColor(wheel(i));
    delay(20);
  }
}

void setColor(long rgb) {
  ledcWrite(ledPins[0], 255 - (rgb >> 16) & 0xFF);
  ledcWrite(ledPins[1], 255 - (rgb >> 8) & 0xFF);
  ledcWrite(ledPins[2], 255 - (rgb >> 0) & 0xFF);
}

long wheel(int pos) {
  long WheelPos = pos % 0xff;
  if (WheelPos < 85) {
    return ((255 - WheelPos * 3) << 16) | ((WheelPos * 3) << 8);
  } else if (WheelPos < 170) {
```

```
28      WheelPos -= 85;
29      return (((255 - WheelPos * 3) << 8) | (WheelPos * 3));
30    } else {
31      WheelPos -= 170;
32      return ((WheelPos * 3) << 16 | (255 - WheelPos * 3));
33    }
34  }
```

In setColor(), a variable represents the value of RGB, and a hexadecimal representation of color is a common representation, such as 0xAABBCC, where AA represents the red value, BB represents the green value, and CC represents the blue value. The use of a variable can make the transmission of parameters more convenient, in the split, only a simple operation can take out the value of each color channel

```
18  void setColor(long rgb) {
19    ledcWrite(ledPins[0], 255 - (rgb >> 16) & 0xFF);
20    ledcWrite(ledPins[1], 255 - (rgb >> 8) & 0xFF);
21    ledcWrite(ledPins[2], 255 - (rgb >> 0) & 0xFF);
22  }
```

The wheel() function is the color selection method for the color model introduced earlier. The **pos** parameter ranges from 0 to 255 and outputs a color value in hexadecimal.

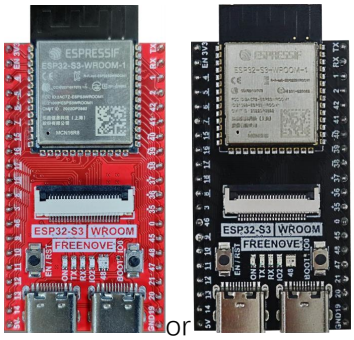
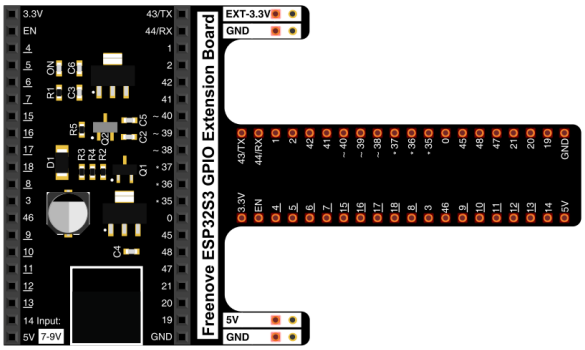
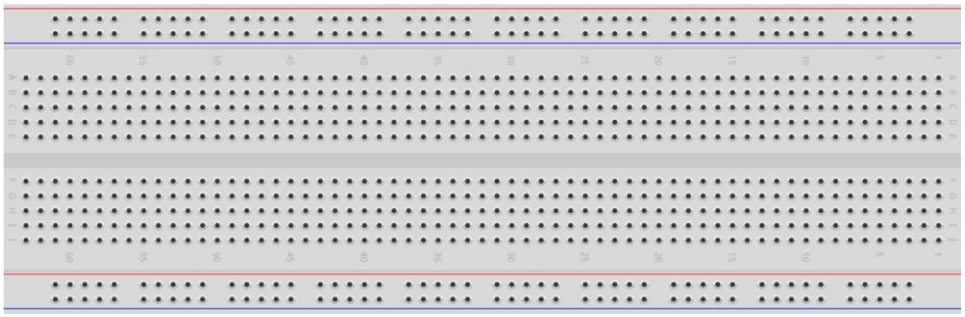
Chapter 6 Buzzer

In this chapter, we will learn about buzzers that can make sounds.

Project 6.1 Doorbell

We will make this kind of doorbell: when the button is pressed, the buzzer sounds; and when the button is released, the buzzer stops sounding.

Component List

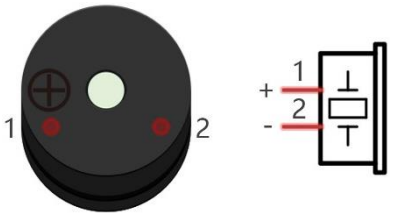
<table>
<tr><td colspan="3">ESP32-S3(N16R8)/ESP32-S3(N8R8)
 
or</td><td colspan="2">GPIO Extension Board x1
  </td></tr>
<tr><td colspan="5">Breadboard x1
  </td></tr>
<tr><td colspan="5">Jumper M/M x6
  </td></tr>
<tr><td>NPN transistorx1
(S8050)
  </td><td>Active buzzer x1
  </td><td>Push button x1
  </td><td>Resistor 1kΩ x1
  </td><td>Resistor 10kΩ x2
  </td></tr>
</table>

Component knowledge

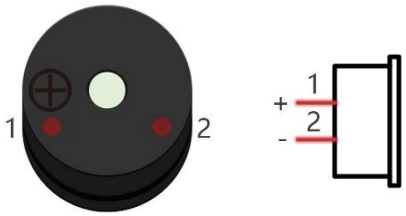
Buzzer

Buzzer is a sounding component, which is widely used in electronic devices such as calculator, electronic warning clock and alarm. Buzzer has two types: active and passive. Active buzzer has oscillator inside, which will sound as long as it is supplied with power. Passive buzzer requires external oscillator signal (generally use PWM with different frequency) to make a sound.

Active buzzer



Passive buzzer



Active buzzer is easy to use. Generally, it can only make a specific frequency of sound. Passive buzzer requires an external circuit to make a sound, but it can be controlled to make a sound with different frequency. The resonant frequency of the passive buzzer is 2kHz, which means the passive buzzer is loudest when its resonant frequency is 2kHz.
Next, we will use an active buzzer to make a doorbell and a passive buzzer to make an alarm.

How to identify active and passive buzzer?

1. Usually, there is a label on the surface of active buzzer covering the vocal hole, but this is not an absolute judgment method.
2. Active buzzers are more complex than passive buzzers in their manufacture. There are many circuits and crystal oscillator elements inside active buzzers; all of this is usually protected with a waterproof coating (and a housing) exposing only its pins from the underside. On the other hand, passive buzzers do not have protective coatings on their underside. From the pin holes viewing of a passive buzzer, you can see the circuit board, coils, and a permanent magnet (all or any combination of these components depending on the model.

Active buzzer

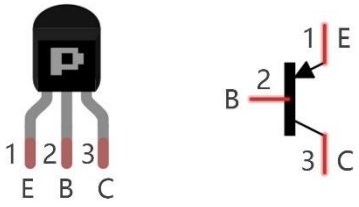
Passive buzzer

Transistor

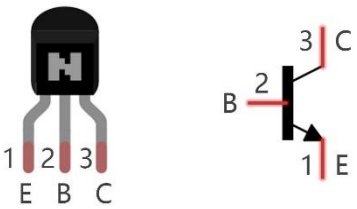
Because the buzzer requires such large current that GPIO of ESP32-S3 output capability cannot meet the requirement, a transistor of NPN type is needed here to amplify the current.

Transistor, the full name: semiconductor transistor, is a semiconductor device that controls current. Transistor can be used to amplify weak signal, or works as a switch. It has three electrodes(PINs): base (b), collector (c) 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

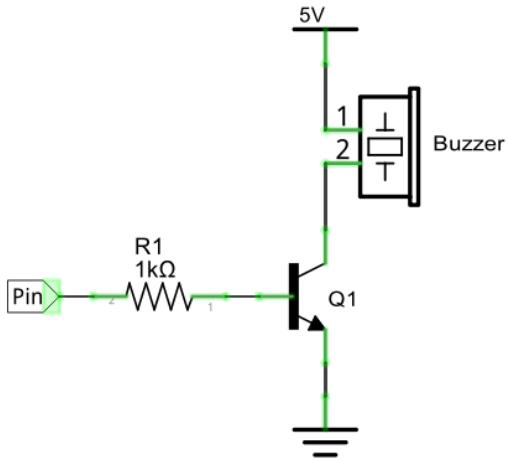
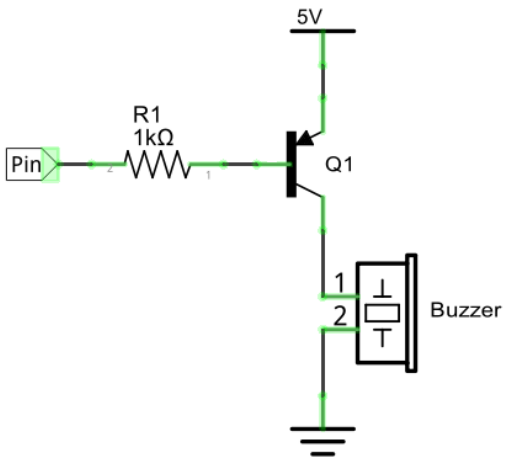


In our kit, the PNP transistor is marked with 8550, and the NPN transistor is marked with 8050.

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.

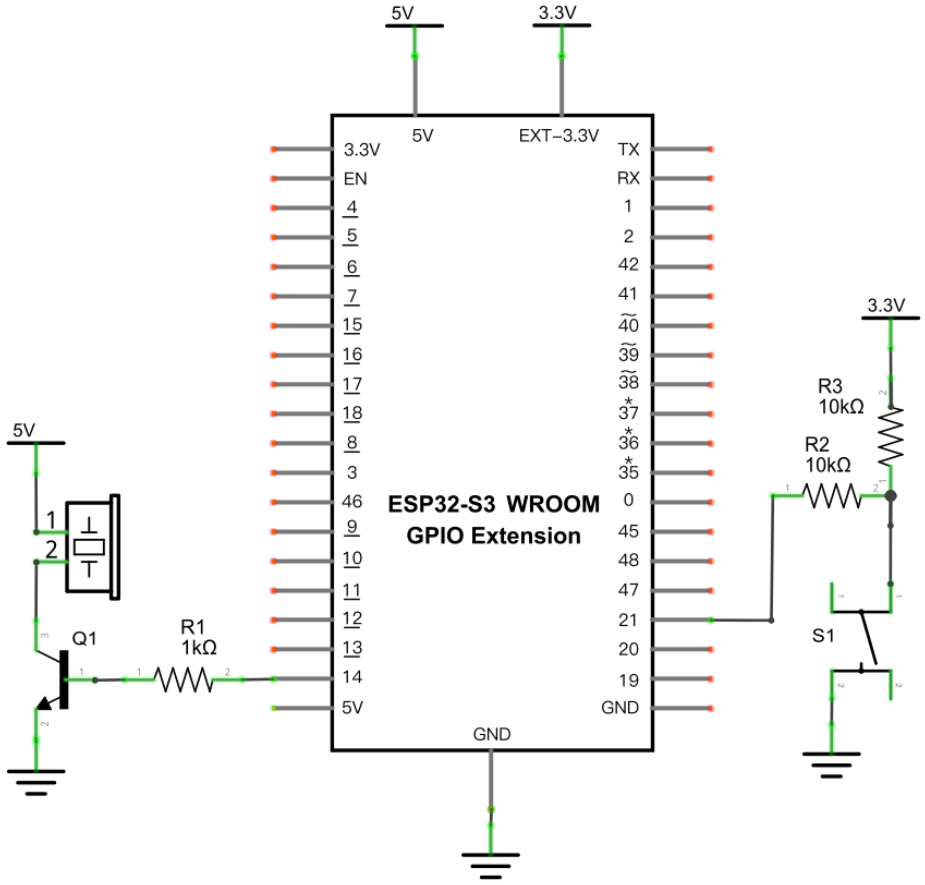
When use NPN transistor to drive buzzer, we often adopt the following method. If GPIO outputs high level, current will flow through R1, the transistor will get conducted, and the buzzer will sound. If GPIO outputs low level, no current flows through R1, the transistor will not be conducted, and buzzer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

When use 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.

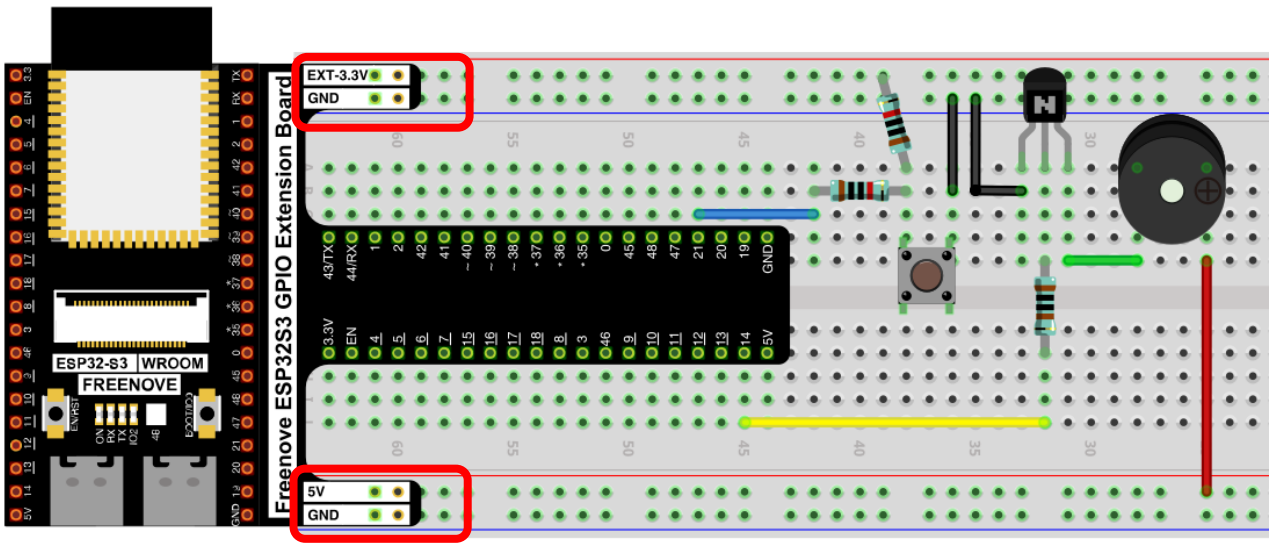
NPN transistor to drive buzzer	PNP transistor to drive buzzer
	

Circuit

Schematic diagram



Hardware connection. If you need any support, please feel free to contact us via: support@freenove.com

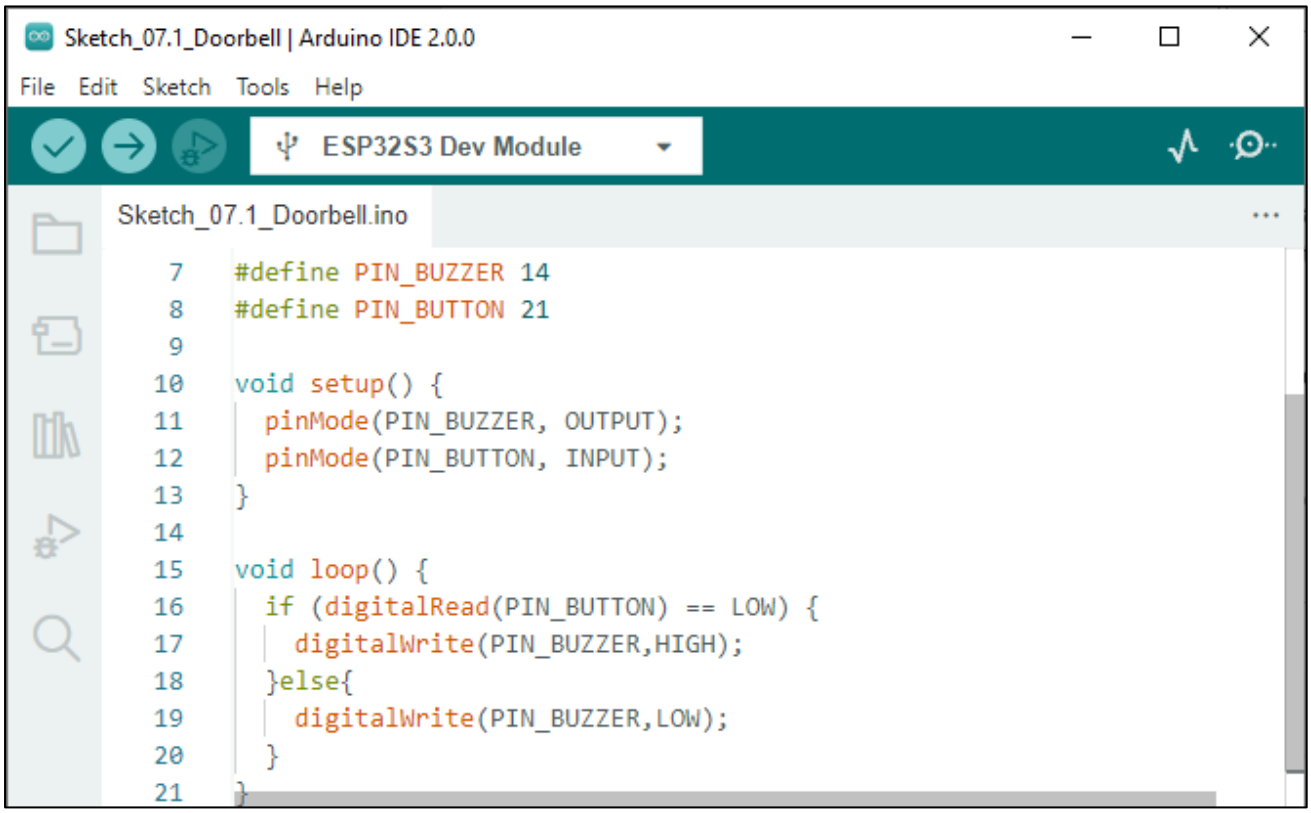


Note: in this circuit, the power supply for buzzer is 5V, and pull-up resistor of the button connected to the power 3.3V. The buzzer can work when connected to power 3.3V, but it will reduce the loudness.

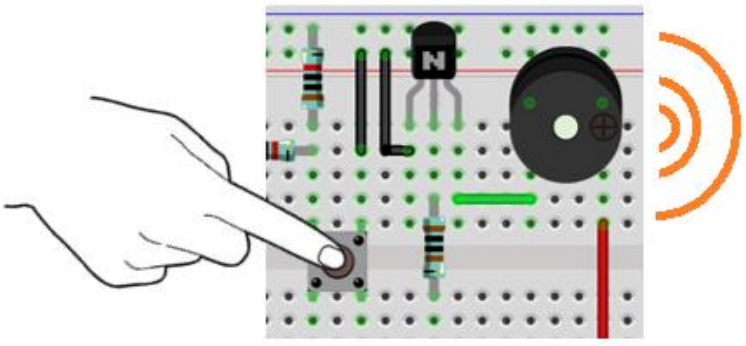
Sketch

In this project, a buzzer will be controlled by a push button switch. When the button switch is pressed, the buzzer sounds and when the button is released, the buzzer stops. It is analogous to our earlier project that controlled a LED ON and OFF.

Sketch_06.1_Doorbell



Download the code to ESP32-S3 WROOM, press the push button switch and the buzzer will sound. Release the push button switch and the buzzer will stop.



The following is the program code:

```
1  #define PIN_BUZZER 14
2  #define PIN_BUTTON 21
3
4  void setup() {
```

```
5     pinMode(PIN_BUZZER, OUTPUT);
6     pinMode(PIN_BUTTON, INPUT);
7   }
8
9   void loop() {
10    if (digitalRead(PIN_BUTTON) == LOW) {
11      digitalWrite(PIN_BUZZER,HIGH);
12    }else{
13      digitalWrite(PIN_BUZZER,LOW);
14    }
15  }
```

The code is logically the same as using button to control LED.

Project 6.2 Alertor

Next, we will use a passive buzzer to make an alarm.
Component list and the circuit is similar to the last section. In the Doorbell circuit only the **active buzzer** needs to be **replaced** with a **passive buzzer.**

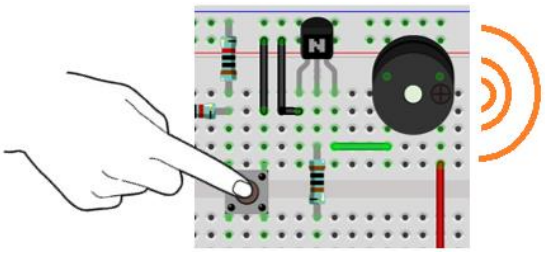
Sketch

In this project, the buzzer alarm is controlled by the button. Press the button, then buzzer sounds. If you release the button, the buzzer will stop sounding. It is logically the same as using button to control LED, but in the control method, passive buzzer requires PWM of certain frequency to sound.
Sketch_06.2_Alertor

File Edit Sketch Tools Help
ESP32S3 Dev Module
Sketch_07.2_Aleror.ino

```
#define PIN_BUZZER 14
#define PIN_BUTTON 21
#define CHN        0   //define the pwm channel

void setup() {
  pinMode(PIN_BUTTON, INPUT);
  pinMode(PIN_BUZZER, OUTPUT);
  ledcAttachChannel(PIN_BUZZER, 1, 10, CHN);  //attach the led pin to pwm channel
  ledcWriteTone(PIN_BUZZER, 2000);        //Sound at 2KHz for 0.3 seconds
  delay(300);
}

void loop() {
  if (digitalRead(PIN_BUTTON) == LOW) {
    alert();
  } else {
    ledcWriteTone(PIN_BUZZER, 0);
  }
}

void alert() {
  float sinVal;         // Define a variable to save sine value
  int toneVal;          // Define a variable to save sound frequency
  for (int x = 0; x < 360; x += 10) {     // X from 0 degree->360 degree
    sinVal = sin(x * (PI / 180));       // Calculate the sine of x
    toneVal = 2000 + sinVal * 500;      // Calculate sound frequency according to the sine of x
    ledcWriteTone(PIN_BUZZER, toneVal);
    delay(10);
  }
}
```

Download the code to ESP32-S3 WROOM, press the button, then alarm sounds. And when the button is released, the alarm will stop sounding.



The following is the program code:

```
#define PIN_BUZZER 14
#define PIN_BUTTON 21
#define CHN        0   //define the pwm channel

void setup() {
  pinMode(PIN_BUTTON, INPUT);
  pinMode(PIN_BUZZER, OUTPUT);
  ledcAttachChannel(PIN_BUZZER, 0, 10, CHN);  //attach the led pin to pwm channel
  ledcWriteTone(PIN_BUZZER, 2000);        //Sound at 2KHz for 0.3 seconds
  delay(300);
}

void loop() {
  if (digitalRead(PIN_BUTTON) == LOW) {
    alert();
  } else {
    ledcWriteTone(PIN_BUZZER, 0);
  }
}

void alert() {
  float sinVal;         // Define a variable to save sine value
  int toneVal;          // Define a variable to save sound frequency
  for (int x = 0; x < 360; x += 10) {     // X from 0 degree->360 degree
    sinVal = sin(x * (PI / 180));       // Calculate the sine of x
    toneVal = 2000 + sinVal * 500;      //Calculate sound frequency according to the sine of x
    ledcWriteTone(PIN_BUZZER, toneVal);
    delay(10);
  }
}
```

The code is the same as the active buzzer logically, but the way to control the buzzer is different. Passive buzzer requires PWM of certain frequency to control, so you need to create a PWM channel through ledcAttachChannel(). Here ledcWriteTone() is designed to generating square wave with variable frequency and duty cycle fixed to 50%, which is a better choice for controlling the buzzer.

```
8    ledcAttachChannel(PIN_BUZZER, 0, 10, CHN);  //attach the led pin to pwm channel
9    ledcWriteTone(PIN_BUZZER, 2000);         //Sound at 2KHz for 0.3 seconds
```

In the while cycle of main function, when the button is pressed, subfunction alert() will be called and the alertor will issue a warning sound. The frequency curve of the alarm is based on the sine curve. We need to calculate the sine value from 0 to 360 degree and multiply a certain value (here is 500) and plus the resonant frequency of buzzer.

```
21  void alert() {
22    float sinVal;         // Define a variable to save sine value
23    int toneVal;          // Define a variable to save sound frequency
24    for (int x = 0; x < 360; x += 10) {     // X from 0 degree->360 degree
25      sinVal = sin(x * (PI / 180));       // Calculate the sine of x
26      toneVal = 2000 + sinVal * 500;      //Calculate sound frequency according to the sine of x
27      ledcWriteTone(PIN_BUZZER, toneVal);
28      delay(10);
29    }
30  }
```

If you want to close the buzzer, just set PWM frequency of the buzzer pin to 0.

```
17  ledcWriteTone(PIN_BUZZER, 0);
```

Reference

```
double ledcWriteTone(uint8_t channel, double freq);
```

This updates the tone frequency value on the given channel.

This function has some bugs in the current version (V1.0.4): when the call interval is less than 20ms, the resulting PWM will have an exception. We will get in touch with the authorities to solve this problem and give solutions in the following two projects.

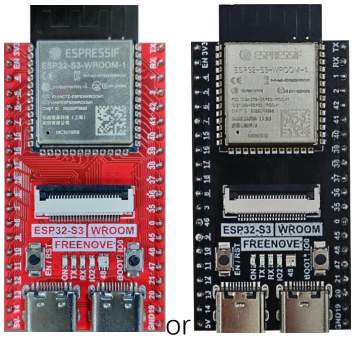
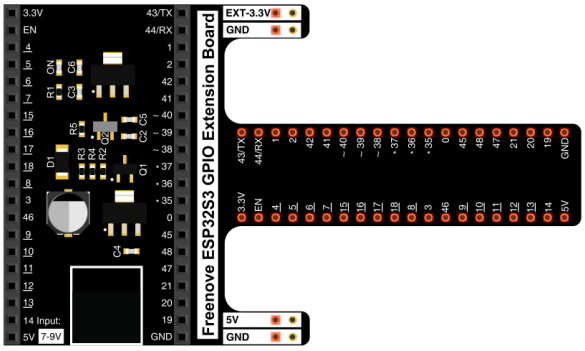
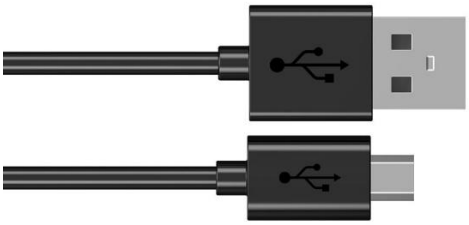
Chapter 7 Serial Communication

Serial Communication is a means of communication between different devices/devices. This section describes ESP32-S3's Serial Communication.

Project 7.1 Serial Print

This project uses ESP32-S3's serial communicator to send data to the computer and print it on the serial monitor.

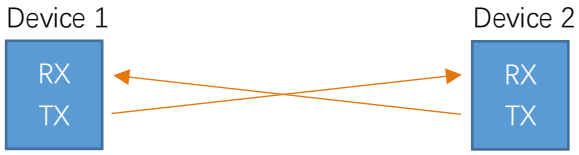
Component List

ESP32-S3(N16R8)/ESP32-S3(N8R8)  or	GPIO Extension Board x1 
Micro USB Wire x1 	

Related knowledge

Serial communication

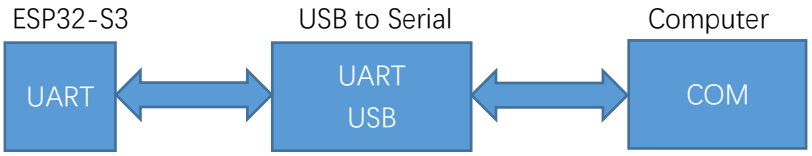
Serial communication generally refers to the Universal Asynchronous Receiver/Transmitter (UART), which is commonly used in electronic circuit communication. It has two communication lines, one is responsible for sending data (TX line) and the other for receiving data (RX line). The serial communication connections of two devices is as follows:



Before serial communication starts, the baud rate of both sides must be the same. Communication between devices can work only if the same baud rate is used. The baud rates commonly used is 9600 and 115200.

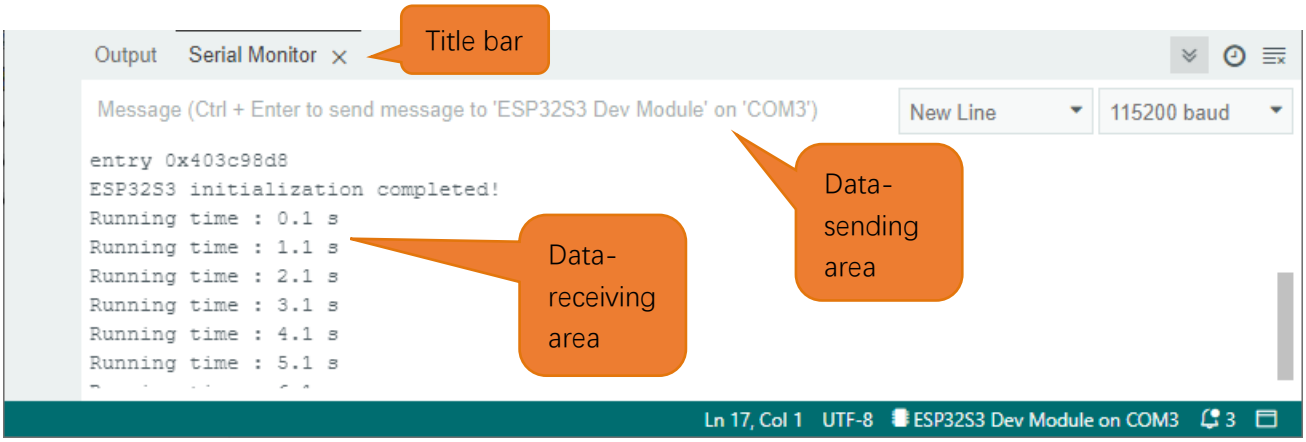
Serial port on ESP32-S3

Freenove ESP32-S3 has integrated USB to serial transfer, so it could communicate with computer connecting to USB cable.



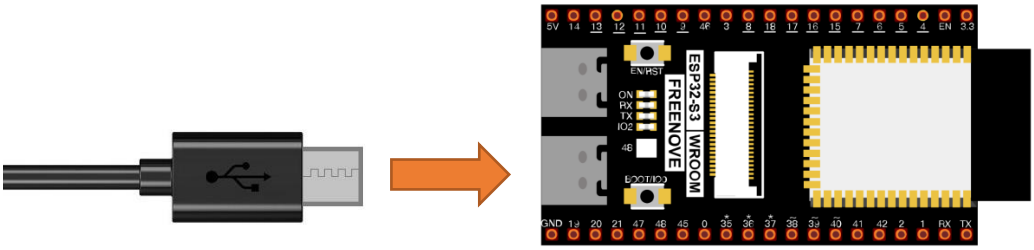
Arduino Software also uploads code to Freenove ESP32-S3 through the serial connection.
Your computer identifies serial devices connecting to it as COMx. We can use the Serial Monitor window of Arduino Software to communicate with Freenove ESP32-S3, connect Freenove ESP32-S3 to computer through the USB cable, choose the correct device, and then click the Serial Monitor icon to open the Serial Monitor window.

Interface of serial monitor window is as follows. If you can't open it, make sure Freenove ESP32-S3 has been connected to the computer, and choose the right serial port in the menu bar "Tools".



Circuit

Connect Freenove ESP32-S3 to the computer with USB cable.



Sketch

Sketch_07.1_SerialPrinter

```
/**********************************************************************
  Filename    : SerialPrinter
  Description : Use UART send some data to PC, and show them on serial monitor.
  Auther      : www.freenove.com
  Modification: 2022/10/20
**********************************************************************/

void setup() {
  Serial.begin(115200);
  Serial.println("ESP32S3 initialization completed!");
}

void loop() {
  Serial.printf("Running time : %.1f s\n", millis() / 1000.0f);
  delay(1000);
}
```

Download the code to ESP32-S3 WROOM, open the serial port monitor, set the baud rate to 115200, and press the reset button. As shown in the following figure:

```
load:0x403cc700,len:0x2a3c
entry 0x403c98d8
ESP32S3 initialization completed!
Running time : 0.1 s
Running time : 1.1 s
Running time : 2.1 s
Running time : 3.1 s
Running time : 4.1 s
```

As shown in the image above, "ESP32-S3 initialization completed! " The previous is the printing message

when the system is started. The user program is then printed at a baud rate of 115200.

The following is the program code:

```
1  void setup() {
2    Serial.begin(115200);
3    Serial.println("ESP32S3 initialization completed! ");
4  }
5
6  void loop() {
7    Serial.printf("Running time : %.1f s\n", millis() / 1000.0f);
8    delay(1000);
9  }
```

Reference

```
void begin(unsigned long baud, uint32_t config=SERIAL_8N1, int8_t rxPin=-1,
           int8_t txPin=-1, bool invert=false, unsigned long timeout_ms = 20000UL);
```

Initializes the serial port. Parameter baud is baud rate, other parameters generally use the default value.

```
size_t println( arg );
```

Print to the serial port and wrap. The parameter **arg** can be a number, a character, a string, an array of characters, etc.

```
size_t printf(const char * format, ...)  __attribute__ ((format (printf, 2, 3)));
```

Print formatted content to the serial port in the same way as print in standard C.

```
unsigned long millis();
```

Returns the number of milliseconds since the current system was booted.

Project 7.2 Serial Read and Write

From last section, we use serial port on Freenove ESP32-S3 to send data to a computer, now we will use that to receive data from computer.

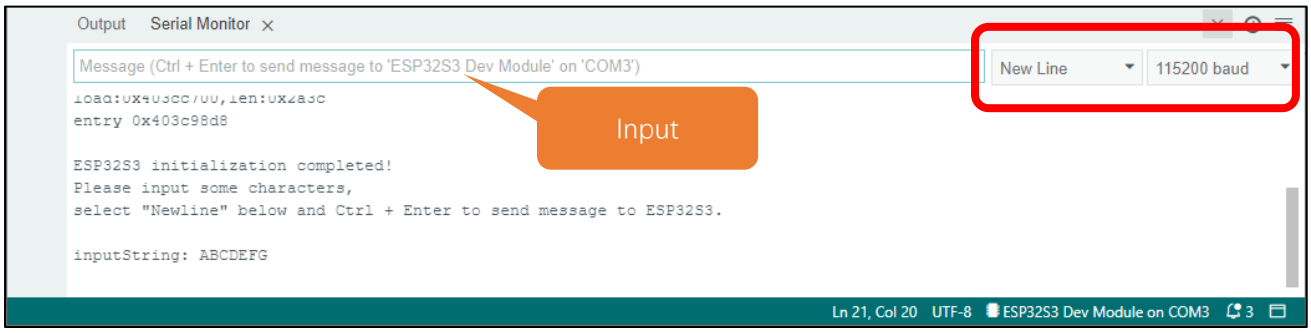
Component and circuit are the same as in the previous project.

Sketch

Sketch_07.2_SerialRW

Download the code to ESP32-S3 WROOM, open the serial monitor, and set the top right corner to **Newline**, **115200**. As shown in the following figure:

Then type characters like 'ABCDEFG' into the data sent at the top, and press Ctrl+Enter to send the message.



The following is the program code:

```
String inputString = "";      //a String to hold incoming data
bool stringComplete = false;  // whether the string is complete

void setup() {
  Serial.begin(115200);
  Serial.println(String("\nESP32S3 initialization completed! \r\n")
                 + String("Please input some characters,\r\n")
                 + String("select \"Newline\" below and Ctrl + Enter to send message to
ESP32S3. \r\n"));
}

void loop() {
  if (Serial.available()) {         // judge whether data has been received
    char inChar = Serial.read();          // read one character
    inputString += inChar;
    if (inChar == '\n') {
      stringComplete = true;
    }
  }
  if (stringComplete) {
    Serial.printf("inputString: %s \n", inputString);
    inputString = "";
    stringComplete = false;
  }
}
```

In loop(), determine whether the serial port has data, if so, read and save the data, and if the newline character is read, print out all the data that has been read.

Reference

String();
Constructs an instance of the String class. For more information, please visit https://www.arduino.cc/reference/en/language/variables/data-types/stringobject/
int available(void);
Get the number of bytes (characters) available for reading from the serial port. This is data that's already arrived and stored in the serial receive buffer.
Serial.read();
Reads incoming serial data.

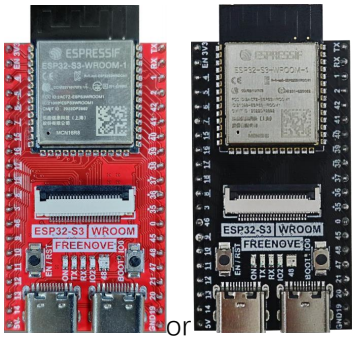
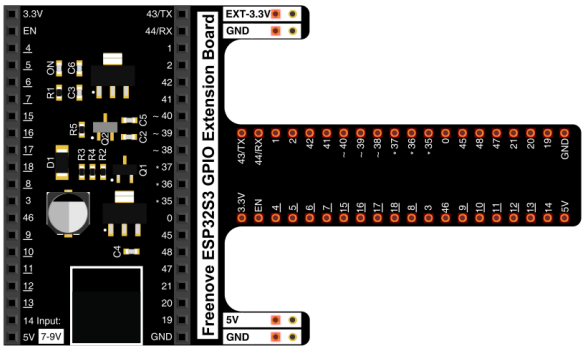
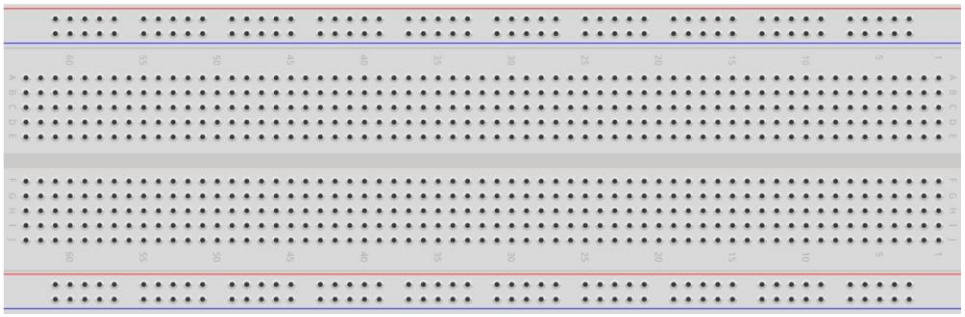
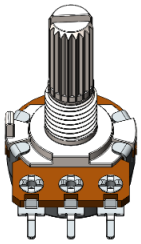
Chapter 8 AD Converter

In this chapter, we will learn how to use ESP32-S3 to read analog signals.

Project 8.1 Read the Voltage of Potentiometer

In this project, we will use the ADC function of ESP32-S3 to read the voltage value of the potentiometer and print it out through the serial monitor.

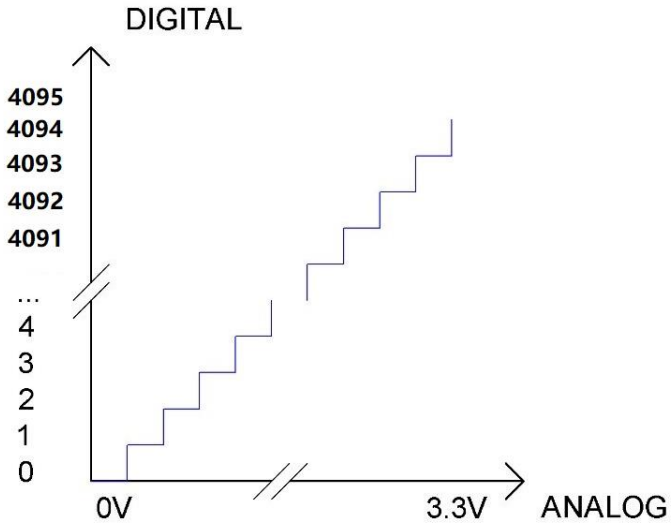
Component List

ESP32-S3(N16R8)/ESP32-S3(N8R8)  or	GPIO Extension Board x1 
Breadboard x1 	
Rotary potentiometer x1 	Jumper M/M x3 

Related knowledge

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-S3 is 12 bits, that means the resolution is 2^12=4096, and it represents a range (at 3.3V) will be divided equally to 4096 parts. The rage of analog values corresponds to ADC values. So the more bits the ADC has, the denser the partition of analog will be and the greater the precision of the resulting conversion.



Subsection 1: the analog in rang of 0V---3.3/4095 V corresponds to digital 0;
Subsection 2: the analog in rang of 3.3/4095 V---2*3.3 /4095V corresponds to digital 1;
...
The following analog will be divided accordingly.
The conversion formula is as follows:

$$ADC\ Value = \frac{\text{Analog Voltage}}{3.3} * 4095$$

ADC on ESP32-S3

ESP32-S3 has two digital analog converters with successive approximations of 12-bit accuracy, and a total of 20 pins can be used to measure analog signals. GPIO pin sequence number and analog pin definition are shown in the following table.

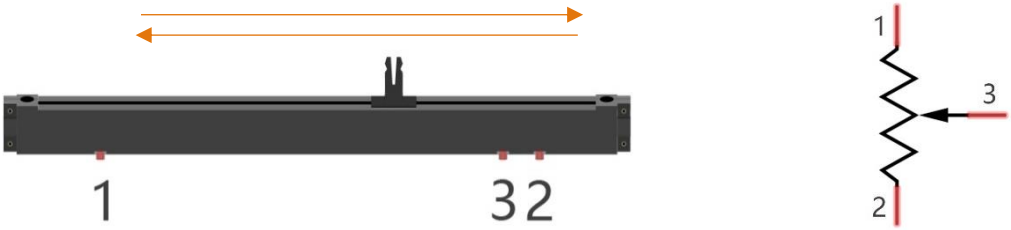
Pin number in Arduino	GPIO number	ADC channel
A0	GPIO 1	**ADC1_CH0**
A1	GPIO 2	**ADC1_CH1**
A2	GPIO 3	**ADC1_CH2**
A3	GPIO 4	**ADC1_CH3**
A4	GPIO 5	**ADC1_CH4**
A5	GPIO 6	**ADC1_CH5**
A6	GPIO 7	**ADC1_CH6**
A7	GPIO 8	**ADC1_CH7**
A8	GPIO 9	**ADC1_CH8**
A9	GPIO 10	**ADC1_CH9**
A10	GPIO 11	ADC2_CH0
A11	GPIO 12	ADC2_CH1
A12	GPIO 13	ADC2_CH2
A13	GPIO 14	ADC2_CH3
A14	GPIO 15	ADC2_CH4
A15	GPIO 16	ADC2_CH5
A16	GPIO 17	ADC2_CH6
A17	GPIO 18	ADC2_CH7
A18	GPIO 19	ADC2_CH8
A19	GPIO 20	ADC2_CH9

The analog pin number is also defined in ESP32-S3's code base. For example, you can replace GPIO1 with A0 in the code.

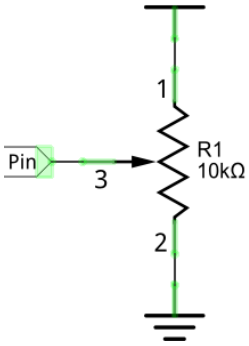
Component knowledge

Potentiometer

A potentiometer is a three-terminal resistor. Unlike the resistors that we have used thus far in our project which have a fixed resistance value, the resistance value of a potentiometer can be adjusted. A potentiometer is often made up by a resistive substance (a wire or carbon element) and movable contact brush. When the brush moves along the resistor element, there will be a change in the resistance of the potentiometer's output side (3) (or change in the voltage of the circuit that is a part). The illustration below represents a linear sliding potentiometer and its electronic symbol on the right.

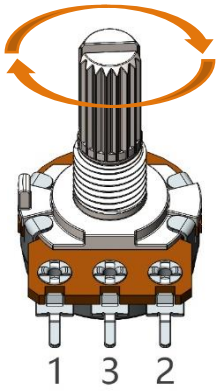


What between potentiometer pin 1 and pin 2 is the resistor body, and pins 3 is connected to brush. When brush moves from pin 1 to pin 2, the resistance between pin 1 and pin 3 will increase up to body resistance linearly, and the resistance between pin 2 and pin 3 will decrease down to 0 linearly.
In the circuit. The both sides of resistance body are often connected to the positive and negative electrode of the power. When you slide the brush pin 3, you can get a certain voltage in the range of the power supply.



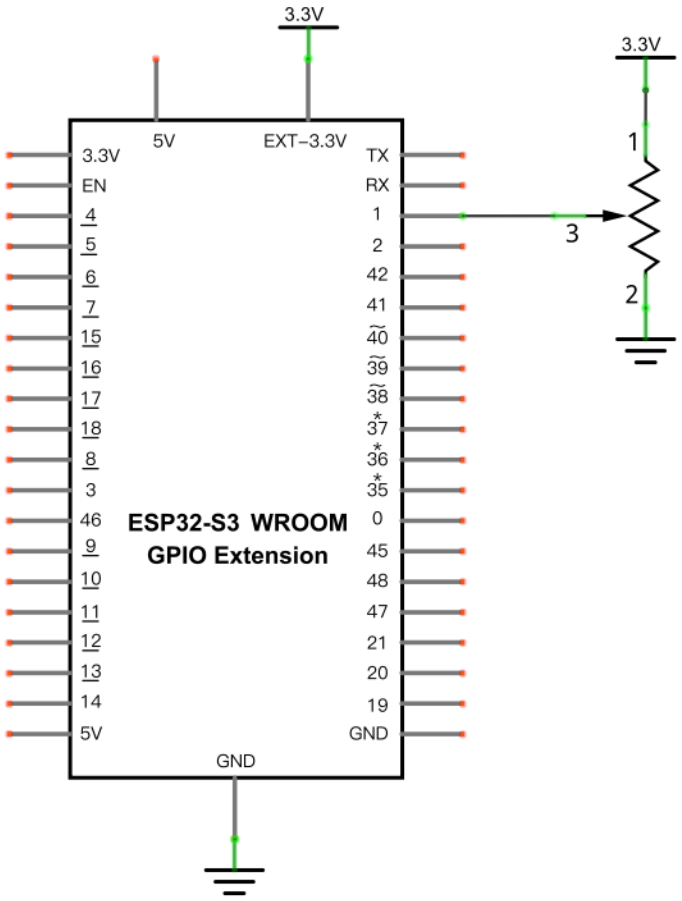
Rotary potentiometer

Rotary potentiometer and linear potentiometer have similar function; their only difference is: the resistance is adjusted by rotating the potentiometer.

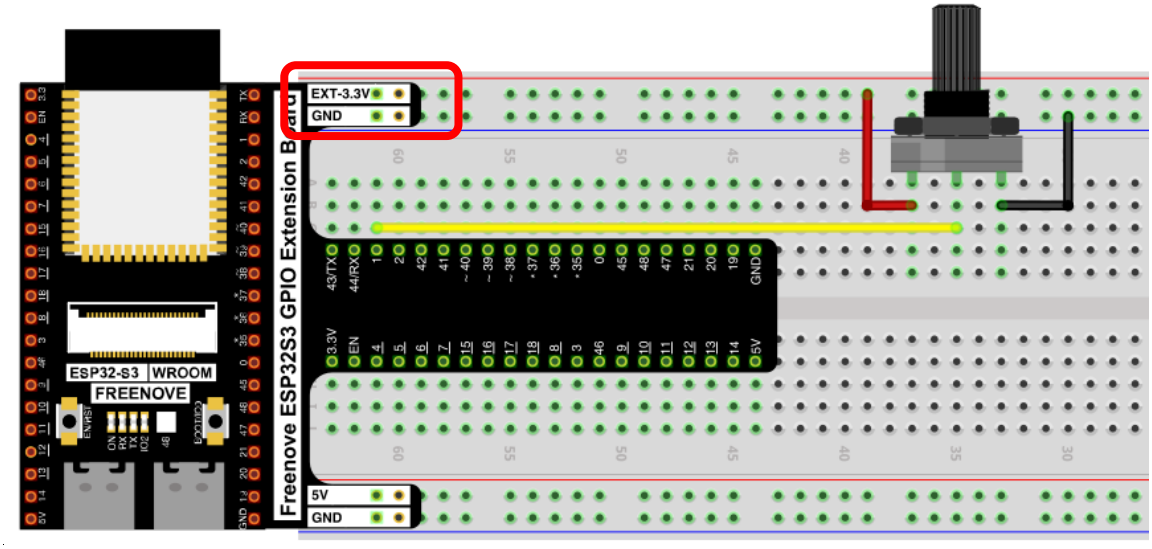


Circuit

Schematic diagram



Hardware connection. If you need any support, please feel free to contact us via: support@freenove.com



Sketch

Sketch_08.1_ADC

File Edit Sketch Tools Help

ESP32S3 Dev Module

Sketch_09.1_ADC.ino

```
/**********************************************************************
  Filename    : ADC_DAC
  Description : Basic usage of ADC and DAC for esp32.
  Auther      : www.freenove.com
  Modification: 2022/10/20
**********************************************************************/
#define PIN_ANALOG_IN  1
void setup() {
  Serial.begin(115200);
}

void loop() {
  int adcVal = analogRead(PIN_ANALOG_IN);
  double voltage = adcVal / 4095.0 * 3.3;
  Serial.printf("ADC Val: %d, \t Voltage: %.2fV\r\n", adcVal, voltage);
  delay(200);
}
```

Download the code to ESP32-S3 WROOM, open the serial monitor, and set the baud rate to 115200. As shown in the following figure.

Message (Ctrl + Enter to send message to 'ESP32S3 Dev Module' on 'COM3')

New Line 115200 baud

```
ADC Val: 0,     Voltage: 0.00V
ADC Val: 155,   Voltage: 0.12V
ADC Val: 613,   Voltage: 0.49V
ADC Val: 1303,  Voltage: 1.05V
ADC Val: 2297,  Voltage: 1.85V
ADC Val: 3307,  Voltage: 2.66V
ADC Val: 4095,  Voltage: 3.30V
ADC Val: 4095,  Voltage: 3.30V
ADC Val: 4095,  Voltage: 3.30V
ADC Val: 4095,  Voltage: 3.30V
ADC Val: 3107,  Voltage: 2.50V
ADC Val: 2341,  Voltage: 1.89V
ADC Val: 1727,  Voltage: 1.39V
ADC Val: 820,   Voltage: 0.66V
```

indexing: 1/48 Ln 18, Col 1 UTF-8 ESP32S3 Dev Module on COM3 2

As shown in the picture above, as long as the handle of the potentiometer is rotated, the serial monitor will print out the ADC value, as well as the voltage value of the potentiometer.

The following is the code:

```
#define PIN_ANALOG_IN  1
void setup() {
  Serial.begin(115200);
}

void loop() {
  int adcVal = analogRead(PIN_ANALOG_IN);
  double voltage = adcVal / 4095.0 * 3.3;
  Serial.printf("ADC Val: %d, \t Voltage: %.2fV\n", adcVal, voltage);
  delay(200);
}
```

In loop(), use the analogRead() function to obtain the input ADC value of the potentiometer, calculate the voltage value of the potentiometer according to the formula in the previous knowledge point, and print it out through the serial port.

```
  int adcVal = analogRead(PIN_ANALOG_IN);
  double voltage = adcVal / 4095.0 * 3.3;
  Serial.printf("ADC Val: %d, \t Voltage: %.2fV\n", adcVal, voltage);
```

Reference

uint16_t analogRead(*uint8_t* pin);

Reads the value from the specified analog pin. Return the analog reading on the pin. (0-4095 for 12 bits).

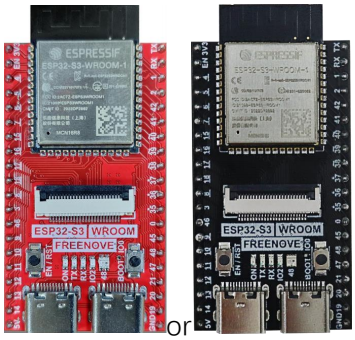
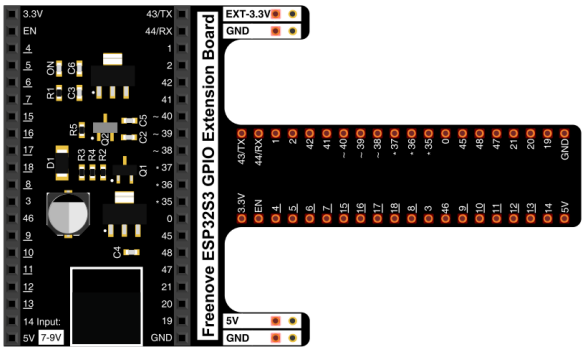
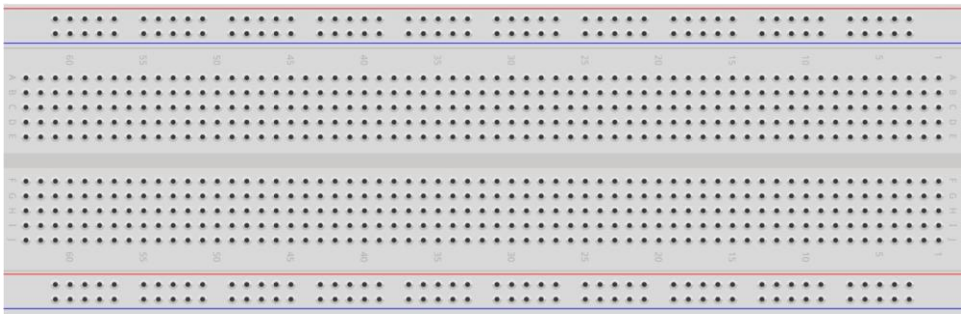
Chapter 9 Touch Sensor

ESP32-S3 offers up to 14 capacitive touch GPIO, and as you can see from the previous section, mechanical switches are prone to jitter that must be eliminated when used, which is not the case with ESP32-S3's built-in touch sensor. In addition, on the service life, the touch switch also has advantages that mechanical switch is completely incomparable.

Project 9.1 Read Touch Sensor

This project reads the value of the touch sensor and prints it out.

Component List

ESP32-S3(N16R8)/ESP32-S3(N8R8) 	GPIO Extension Board x1 
Breadboard x1 	
Jumper M/M x1 	

Related knowledge

Touch sensor

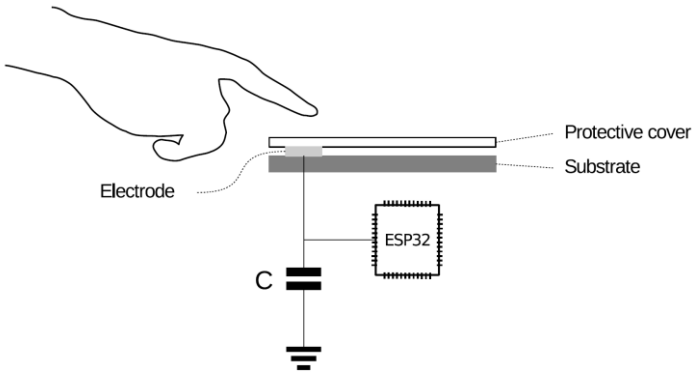
ESP32-S3's touch sensor supports up to 14 GPIO channels as capacitive touch pins. Each pin can be used separately as an independent touch switch or be combined to produce multiple touch points. The following table is a list of available touch pins on ESP32-S3.

Name of touch sensing signal	GPIO number
T1	GPIO1
T2	GPIO2
T3	GPIO3
T4	GPIO4
T5	GPIO5
T6	GPIO6
T7	GPIO7
T8	GPIO8
T9	GPIO9
T10	GPIO10
T11	GPIO11
T12	GPIO12
T13	GPIO13
T14	GPIO14

The touch pin number is already defined in ESP32-S3's code base. For example, in the code, you can use T1 to represent GPIO1.

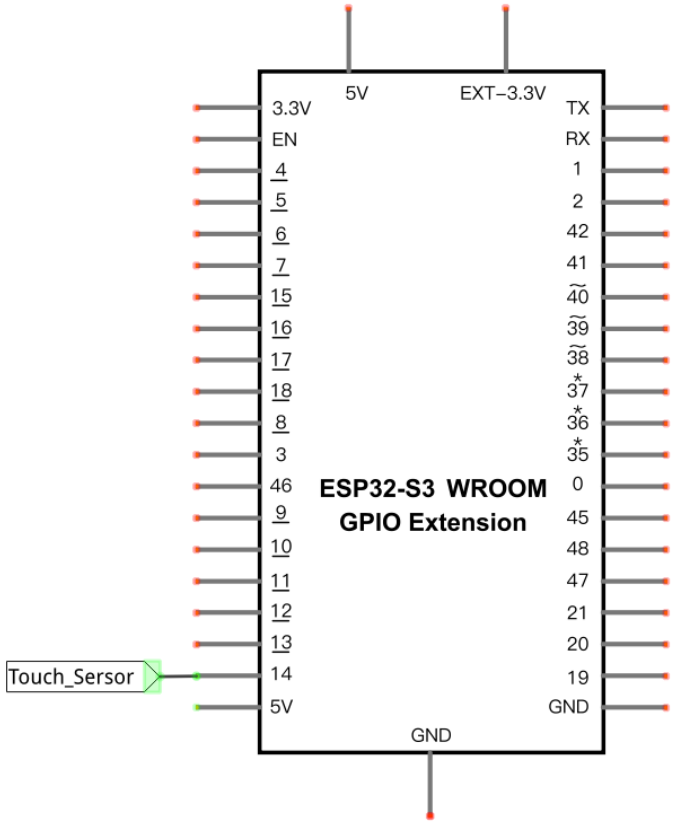
The electrical signals generated by touch are analog data, which are converted by an internal ADC converter. You may have noticed that all touch pins have ADC functionality.

The hardware connection method is shown in the following figure.

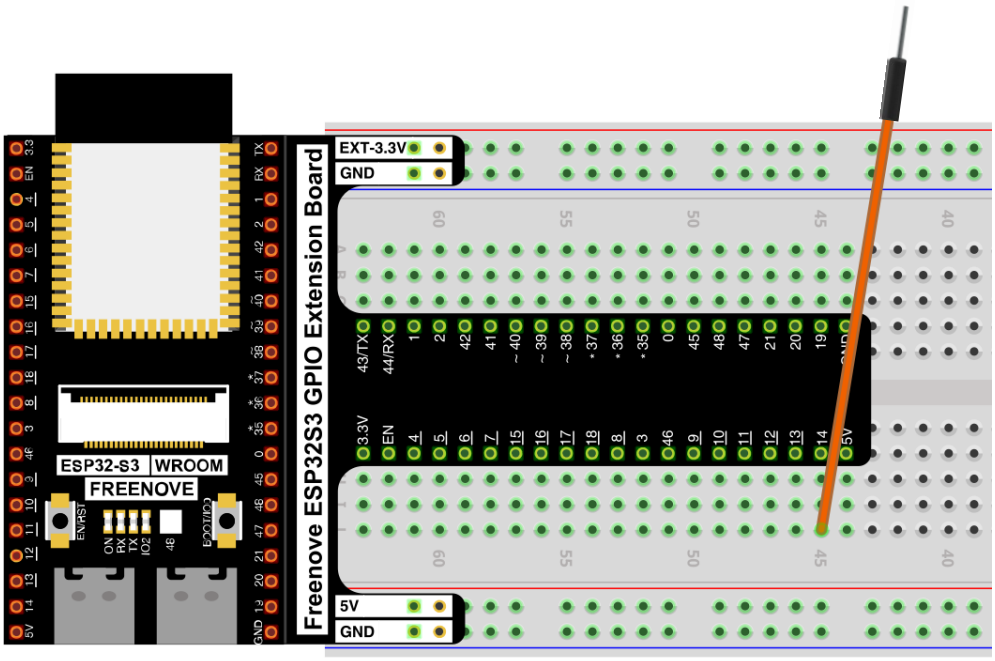


Circuit

Schematic diagram

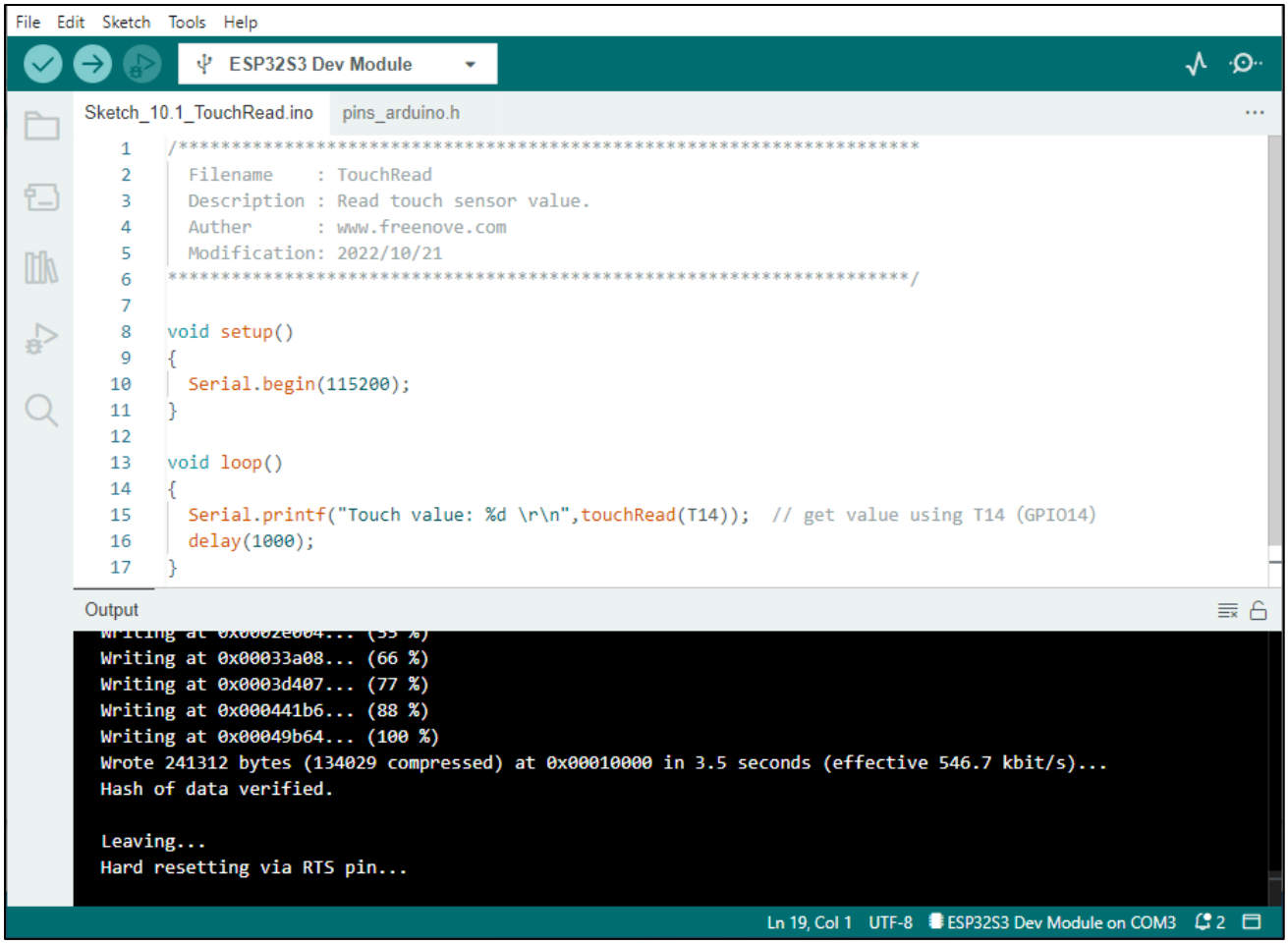


Hardware connection. If you need any support, please feel free to contact us via: support@freenove.com

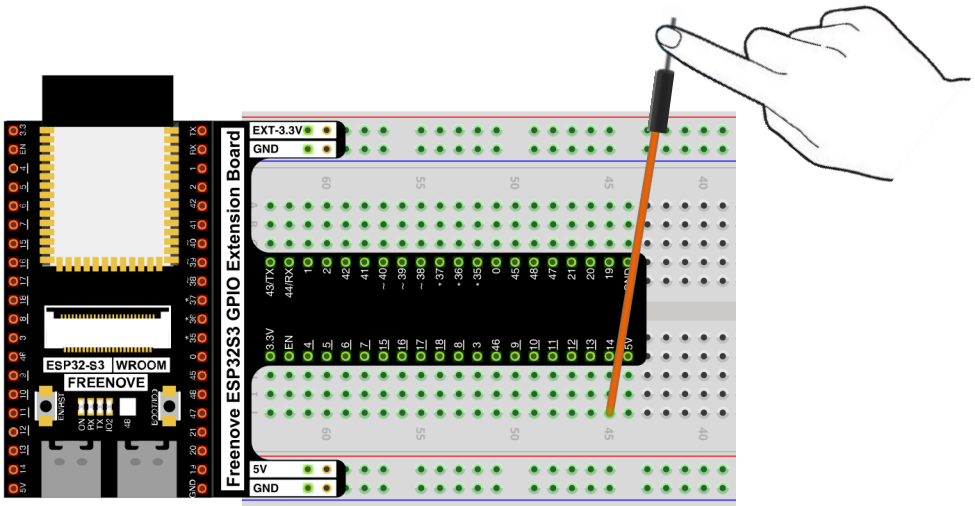


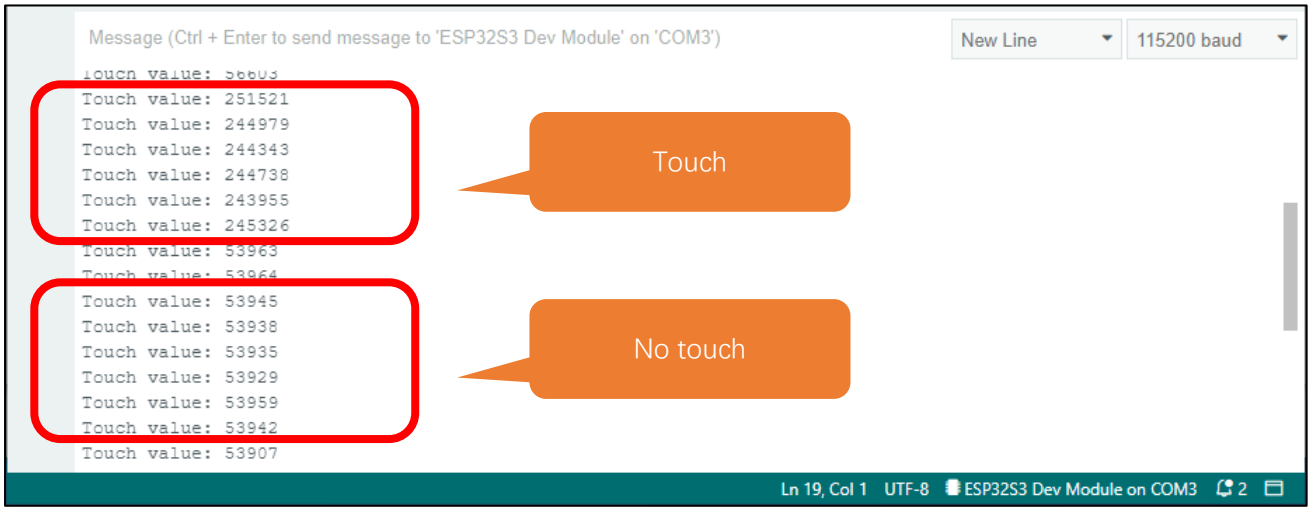
Sketch

Sketch_09.1_TouchRead



Download the code to ESP32-S3 WROOM, open the serial monitor, and set the baud rate to 115200. Touch jumper with hand. As shown in the following figure,





Reference

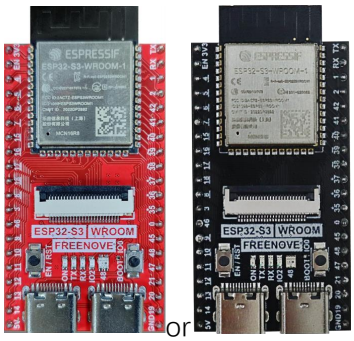
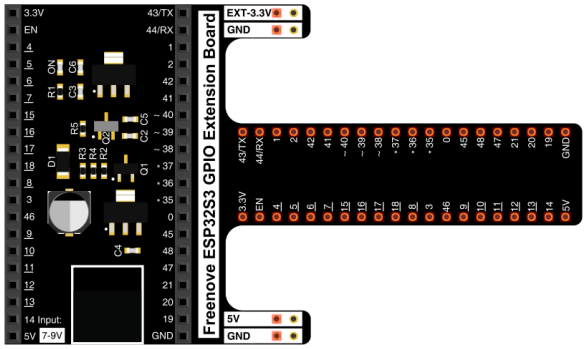
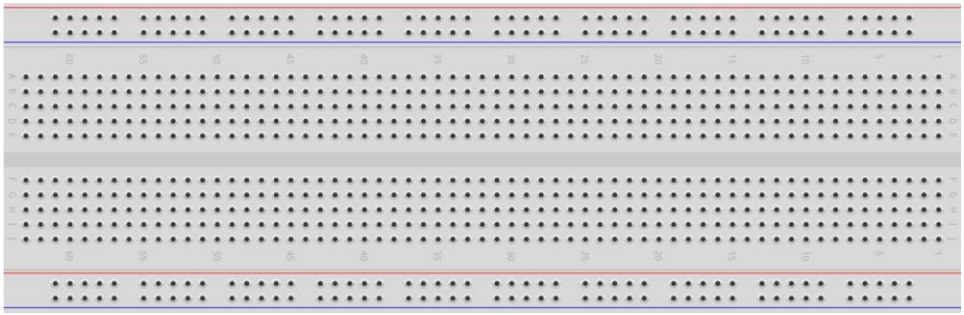
```
uint16_t touchRead(uint8_t pin);
```

Read touch sensor value. (values close to 0 mean touch detected)

Project 9.2 Touch Lamp

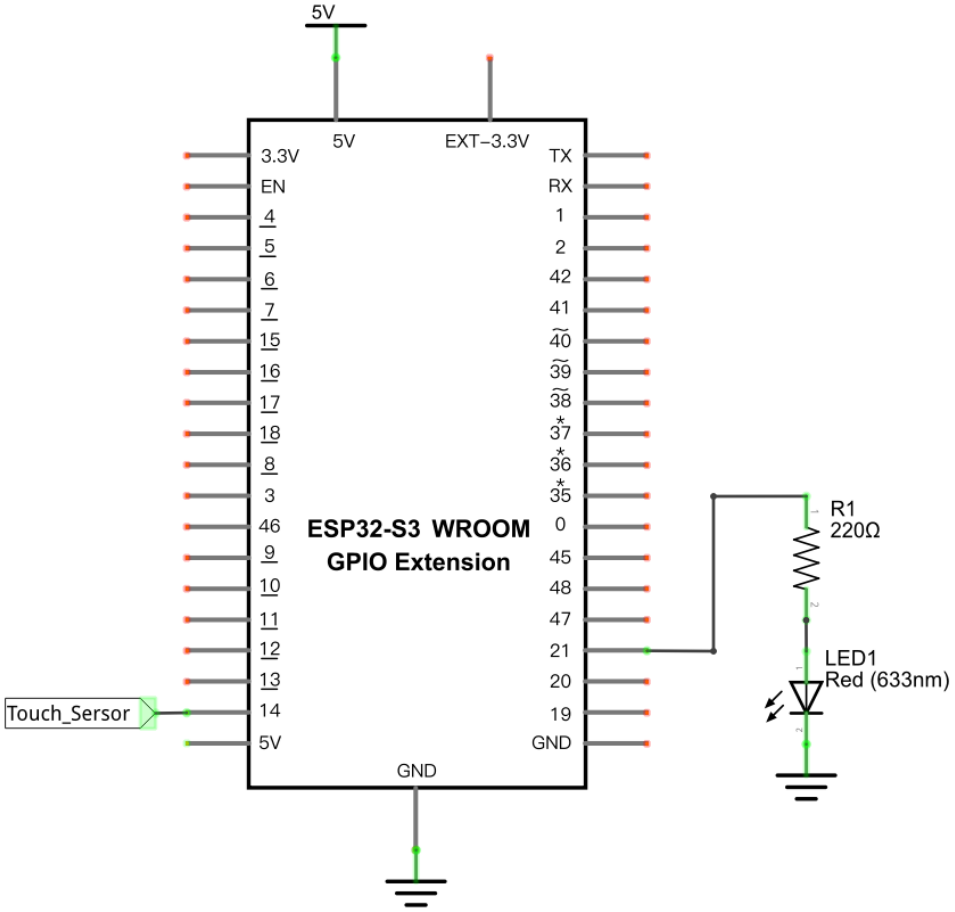
In this project, we will use ESP32-S3's touch sensor to create a touch switch lamp.

Component List

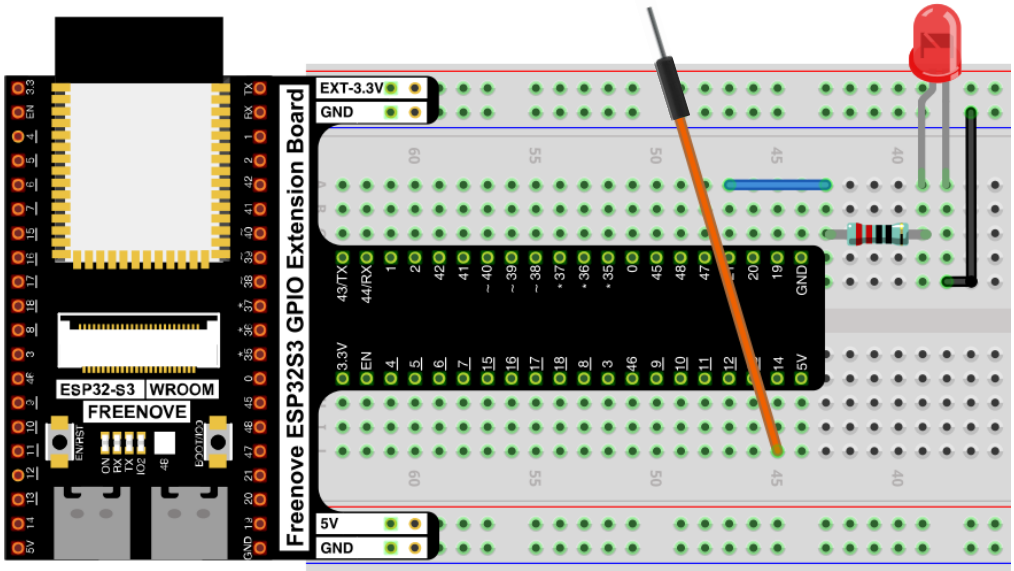
<table>
<tr><td>ESP32-S3(N16R8)/ESP32-S3(N8R8)
  </td><td>GPIO Extension Board x1
  </td></tr>
<tr><td colspan="2">Breadboard x1
  </td></tr>
<tr><td>Jumper M/M x3
  </td><td>LED x1
  </td><td>Resistor 220Ω x1
  </td></tr>
</table>

Circuit

Schematic diagram



Hardware connection. If you need any support, please feel free to contact us via: support@freenove.com

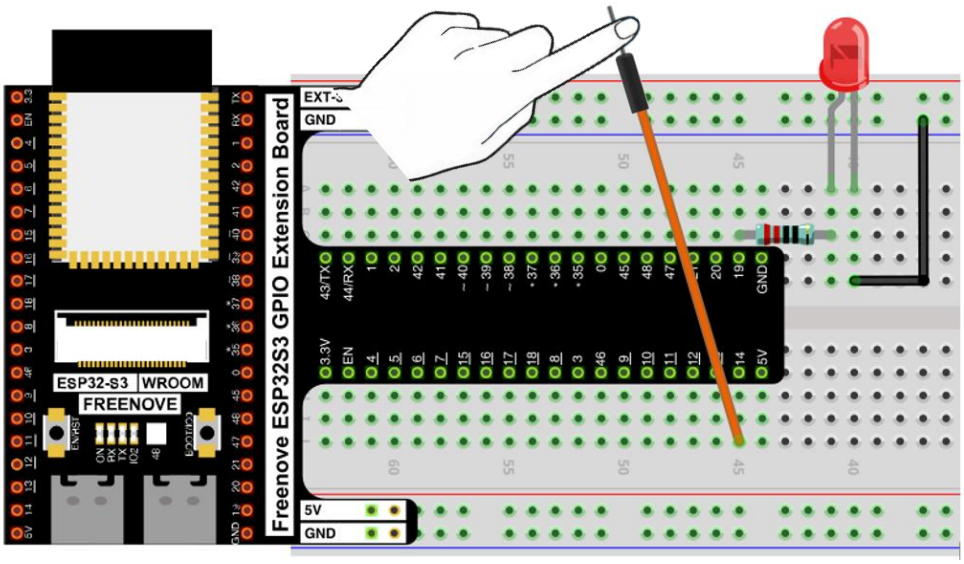


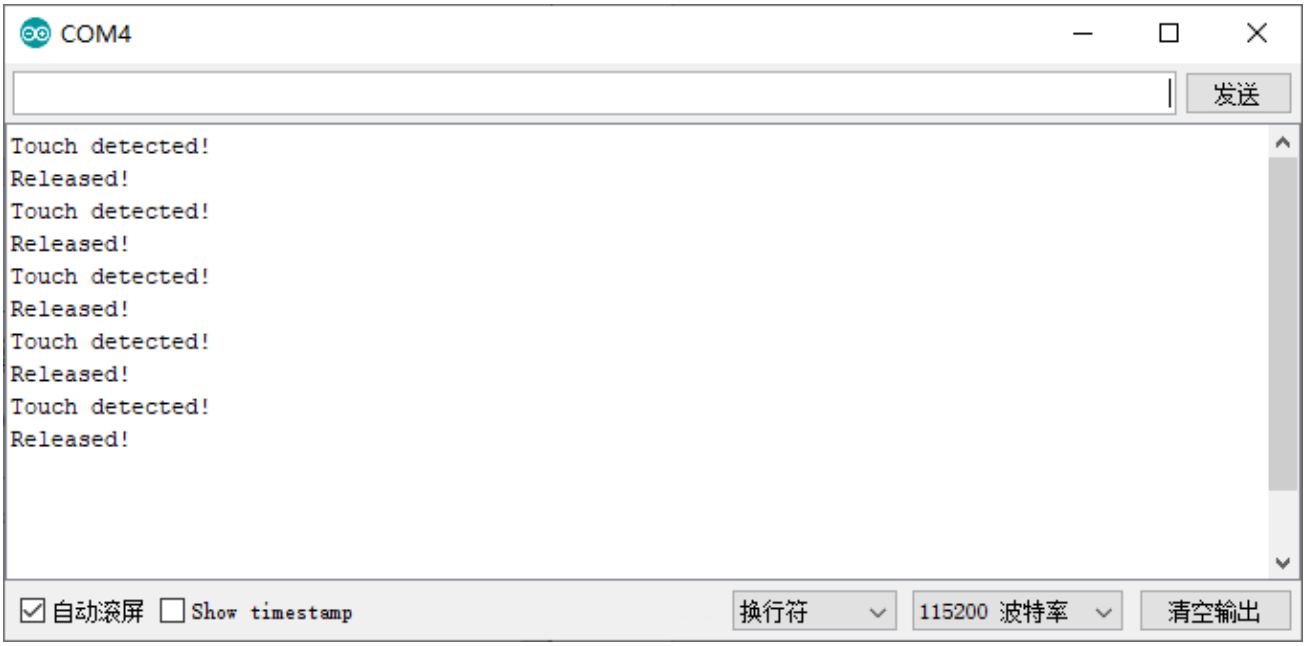
Sketch

Sketch_09.2_TouchLamp

```
#define PIN_LED     21
#define PRESS_VAL   200000    //Set a threshold to judge touch
#define RELEASE_VAL 60000     //Set a threshold to judge release

bool isProcessed = false;
void setup() {
  Serial.begin(115200);
  pinMode(PIN_LED, OUTPUT);
}
void loop() {
  if (touchRead(T14) > PRESS_VAL) {
    if (!isProcessed) {
      isProcessed = true;
      Serial.println("Touch detected! ");
      reverseGPIO(PIN_LED);
    }
  }
  if (touchRead(T14) < RELEASE_VAL) {
    if (isProcessed) {
      isProcessed = false;
      Serial.println("Released! ");
    }
  }
}
```

Download the code to ESP32-S3 WROOM, open the serial monitor, and set the baud rate to 115200. Touch jumper with hand. As shown in the following figure,





With a touch pad, the state of the LED changes with each touch, and the detection state of the touch sensor is printed in the serial monitor.

The following is the program code:

```
#define PIN_LED      21
#define PRESS_VAL    200000 //Set a threshold to judge touch
#define RELEASE_VAL 60000  //Set a threshold to judge release

bool isProcessed = false;
void setup() {
  Serial.begin(115200);
  pinMode(PIN_LED, OUTPUT);
}
void loop() {
  if (touchRead(T14) > PRESS_VAL) {
    if (! isProcessed) {
      isProcessed = true;
      Serial.println("Touch detected!  ");
      reverseGPIO(PIN_LED);
    }
  }

  if (touchRead(T14) < RELEASE_VAL) {
    if (isProcessed) {
      isProcessed = false;
      Serial.println("Released!  ");
    }
  }
```

```
}

void reverseGPIO(int pin) {
  digitalWrite(pin, ! digitalRead(pin));
}
```

Due to different operating environments, the return value of the function touchRead() may not be the same or similar. Therefore, with the help of Project 9.1, we can know the return values of touchRead() in different states, and based on these return values, we can set a valid threshold range for the touch function.

For example, when touchRead() returns a value greater than 200000, we consider the touch function to be triggered by a human. Similarly, when the return value of touchRead() is less than 60000, we consider that the touch function has not been triggered by someone. Note that the threshold range here can be modified by users according to their own conditions

```
#define PRESS_VAL   200000 //Set a threshold to judge touch
#define RELEASE_VAL 60000  //Set a threshold to judge release
```

In loop(), first determine whether the touch was detected. If yes, print some messages, flip the state of the LED, and set the flag bit **isProcessed** to true to avoid repeating the program after the touch was successful.

```
  if (touchRead(T14) > PRESS_VAL) {
    if (! isProcessed) {
      isProcessed = true;
      Serial.println("Touch detected!  ");
      reverseGPIO(PIN_LED);
    }
  }
```

It then determines if the touch key is released, and if so, prints some messages and sets the **isProcessed** to false to avoid repeating the process after the touch release and to prepare for the next touch probe.

```
  if (touchRead(T14) < RELEASE_VAL) {
    if (isProcessed) {
      isProcessed = false;
      Serial.println("Released!  ");
    }
  }
```

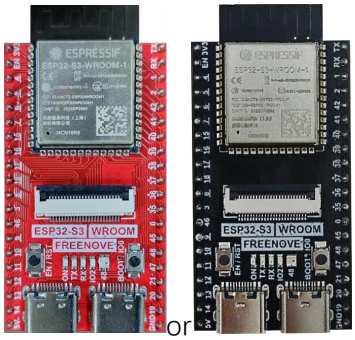
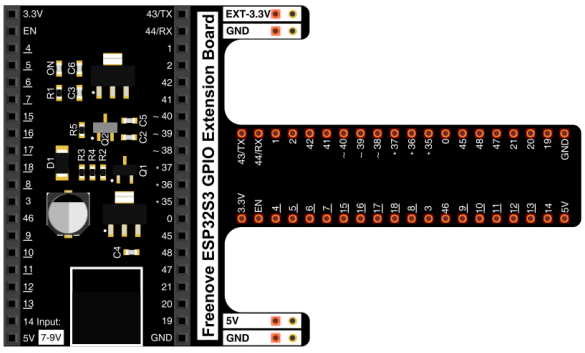
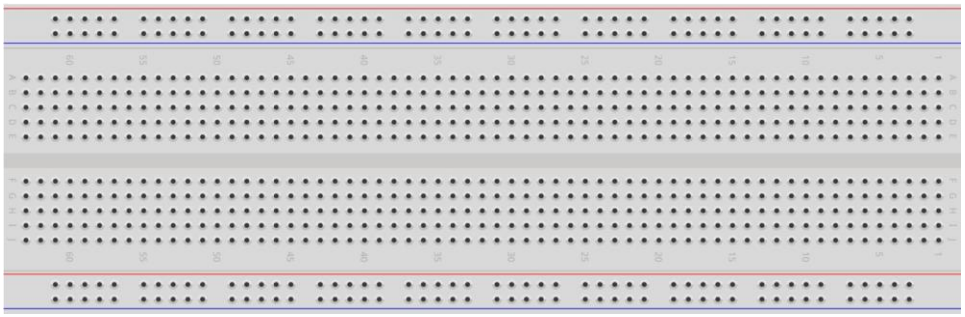
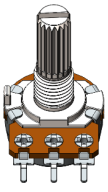
Chapter 10 Potentiometer & LED

Earlier we have learned the use of ADC and PWM. In this chapter, we will learn how to use a potentiometer to control the brightness of an LED.

Project 10.1 Soft Light

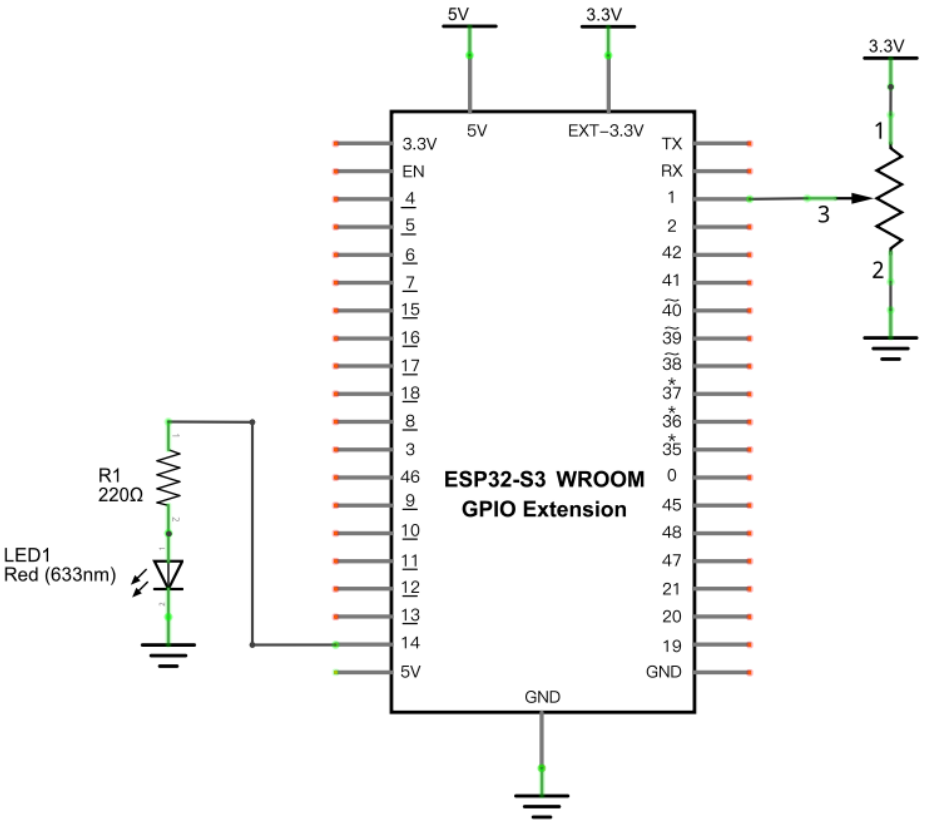
In this project, we will make a soft light. We will use an ADC Module to read ADC values of a potentiometer and map it to duty cycle of the PWM used to control the brightness of a LED. Then you can change the brightness of a LED by adjusting the potentiometer.

Component List

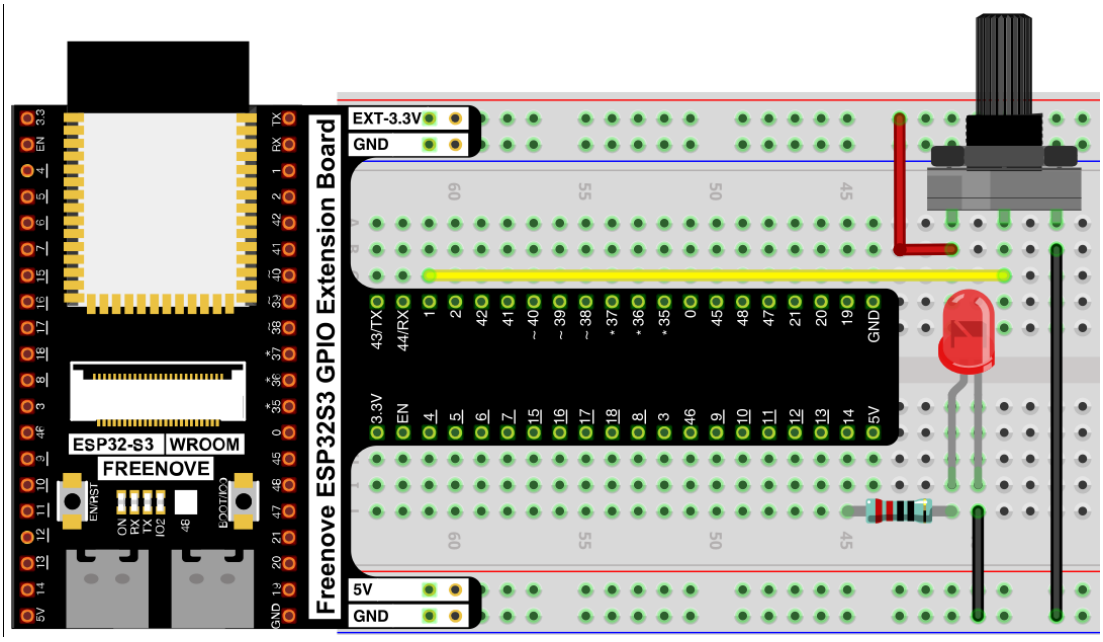
<table>
<tr><td>ESP32-S3(N16R8)/ESP32-S3(N8R8)
  </td><td>GPIO Extension Board x1
  </td></tr>
<tr><td colspan="2">Breadboard x1
  </td></tr>
<tr><td>Rotary potentiometer x1
  </td><td>Resistor 220Ω x1
  </td><td>LED x1
  </td><td>Jumper M/M x5
</td></tr>
</table>

Circuit

Schematic diagram

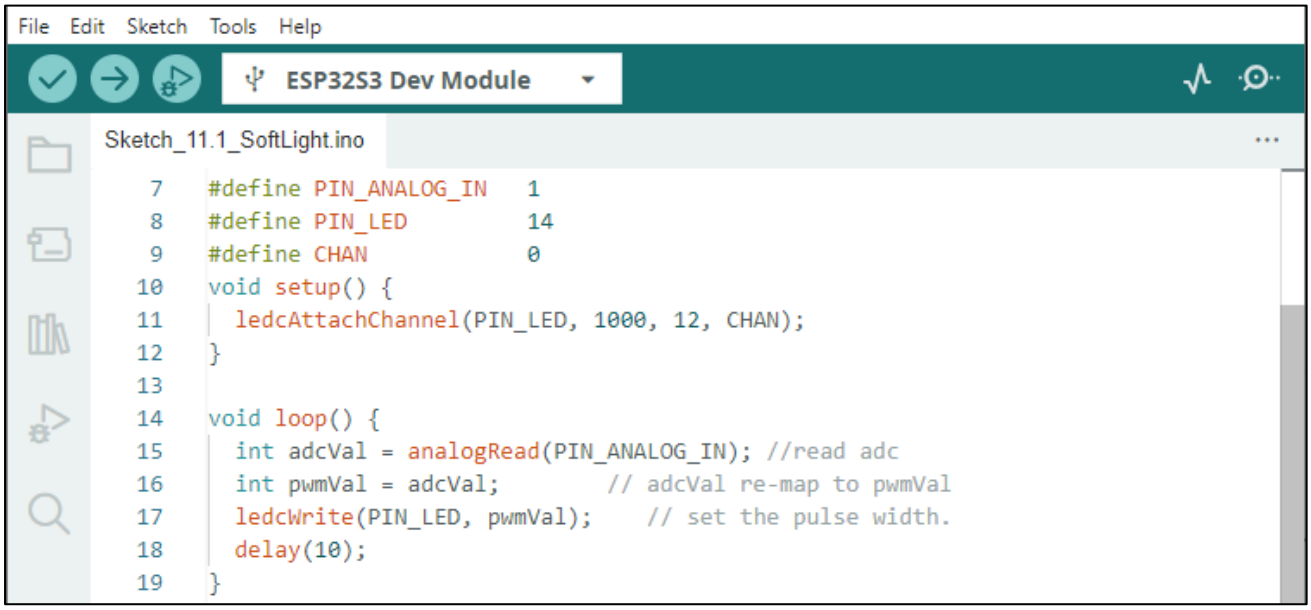


Hardware connection. If you need any support, please feel free to contact us via: support@freenove.com



Sketch

Sketch_10.1_Softlight



Download the code to ESP32-S3 WROOM, by turning the adjustable resistor to change the input voltage of GPIO19, ESP32-S3 changes the output voltage of GPIO14 according to this voltage value, thus changing the brightness of the LED.

The following is the code:

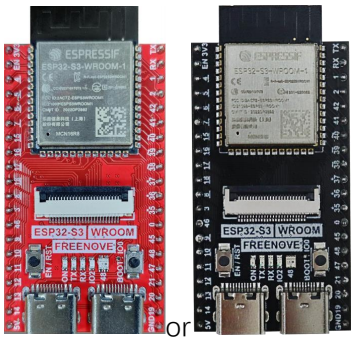
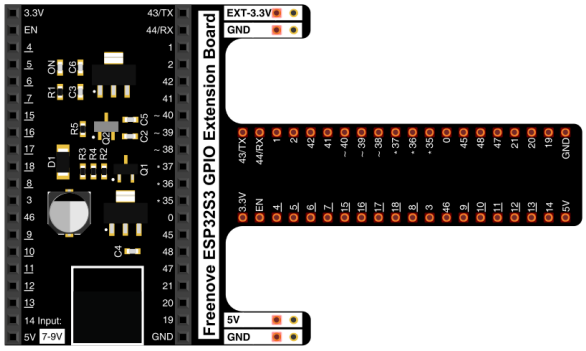
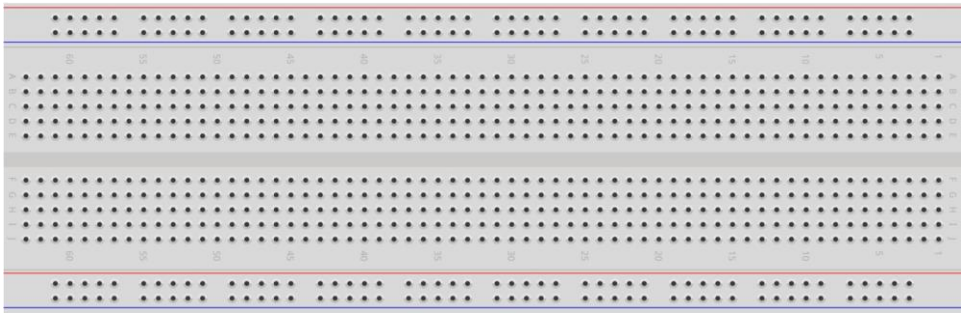
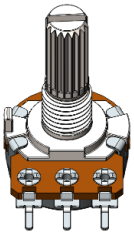
```
#define PIN_ANALOG_IN   1
#define PIN_LED         14
#define CHAN            0
void setup() {
  ledcAttachChannel(PIN_LED, 1000, 12, CHAN);
}

void loop() {
  int adcVal = analogRead(PIN_ANALOG_IN); //read adc
  int pwmVal = adcVal;          // adcVal re-map to pwmVal
  ledcWrite(PIN_LED, pwmVal);   // set the pulse width.
  delay(10);
}
```

In the code, read the ADC value of potentiometer and map it to the duty cycle of PWM to control LED brightness.

Project 10.2 Soft Colorful Light

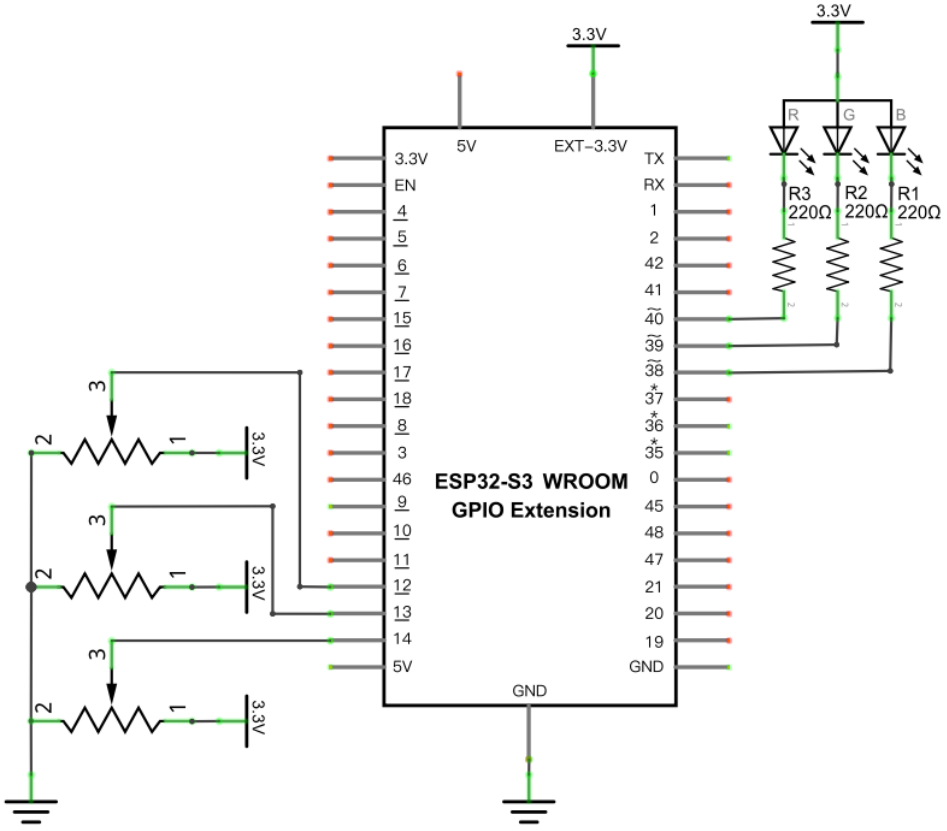
In this project, 3 potentiometers are used to control the RGB LED and in principle it is the same as the Soft Light project. Namely, read the voltage value of the potentiometer and then convert it to PWM used to control LED brightness. Difference is that the original project only controlled one LED, but this project required (3) RGB LEDs.

Component List

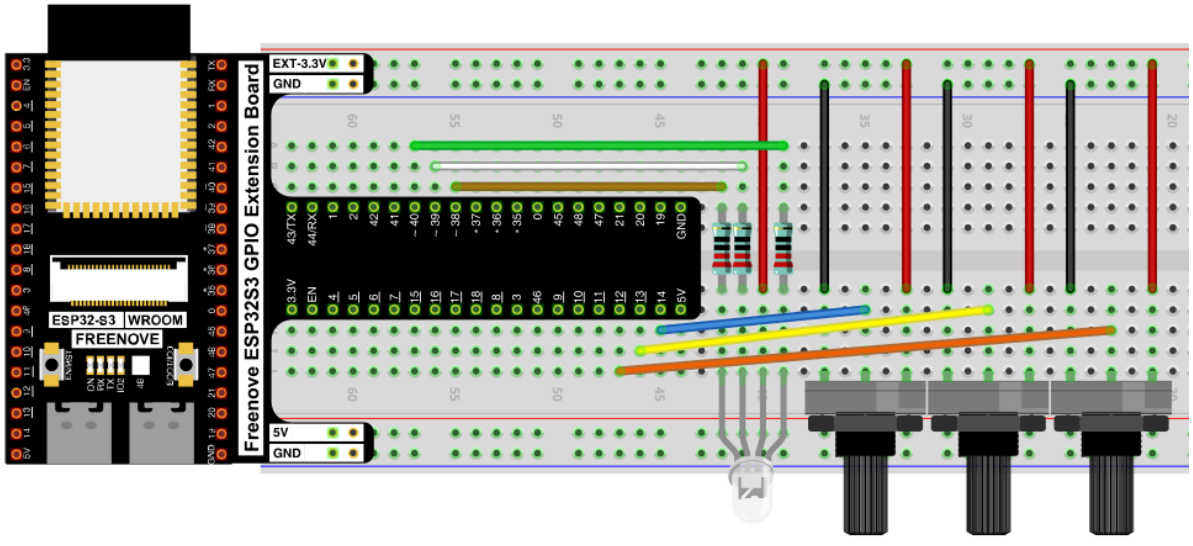
<table>
<tr><td>ESP32-S3(N16R8)/ESP32-S3(N8R8)
 
or</td><td>GPIO Extension Board x1
  </td></tr>
<tr><td colspan="2">Breadboard x1
  </td></tr>
<tr><td>Rotary potentiometer x3
  </td><td>Resistor 220Ω x3
  </td><td>RGBLED x1
  </td><td>Jumper M/M x13
  </td></tr>
</table>

Circuit

Schematic diagram

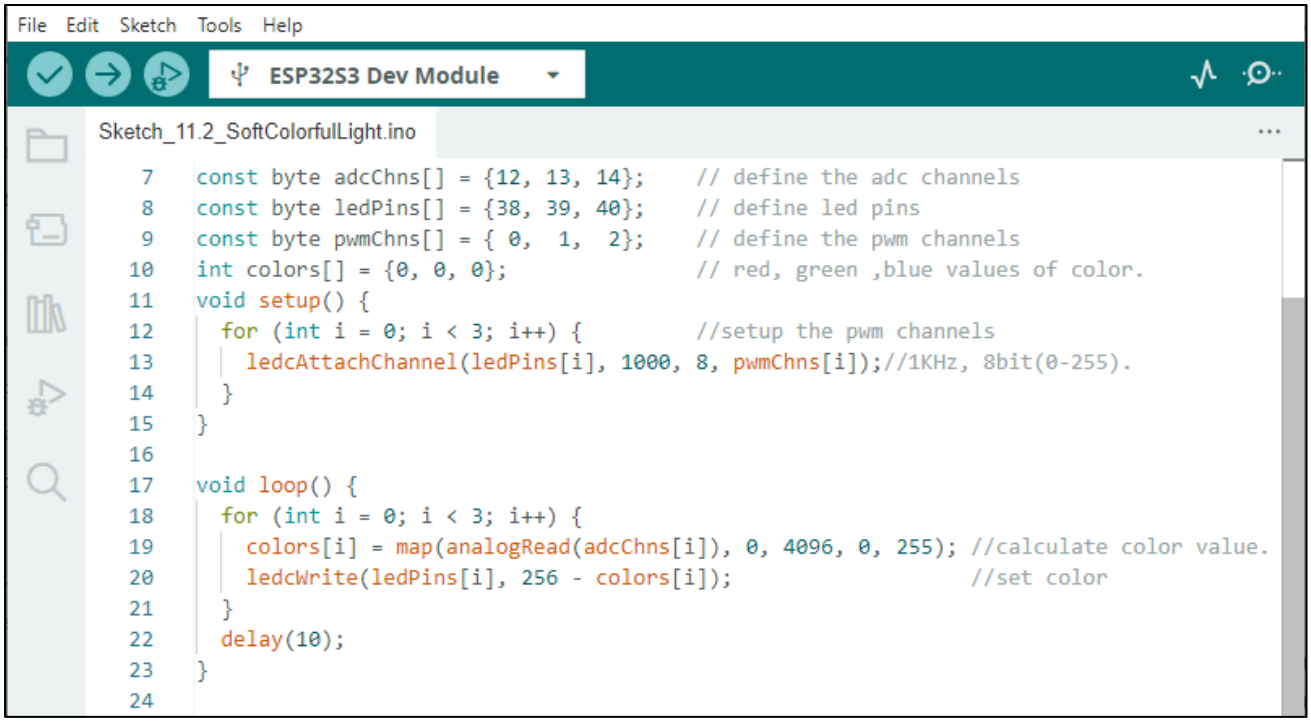


Hardware connection. If you need any support, please feel free to contact us via: support@freenove.com



Sketch

Sketch_10.2_SoftColorfulLight



Download the code to ESP32-S3 WROOM, rotate one of the potentiometers, then the color of RGB LED will change.

If you have any concerns, please contact us via: support@freenove.com

The following is the program code:

```
const byte adcChns[] = {12, 13, 14};      // define the adc channels
const byte ledPins[] = {38, 39, 40};      //define led pins
const byte chns[] =    { 0,  1,  2};      //define the pwm channels
int colors[] = {0, 0, 0};                 // red, green,blue values of color.
void setup() {
  for (int i = 0; i < 3; i++) {           //setup the pwm channels
    ledcAttachChannel(ledPins[i], 1000, 8, chns[i]);  //1KHz, 8bit(0-255).
  }
}

void loop() {
  for (int i = 0; i < 3; i++) {
    colors[i] = map(analogRead(adcChns[i]), 0, 4096, 0, 255); //calculate color
value.
    ledcWrite(ledPins[i], 256 - colors[i]);                   //set color
  }
  delay(10);
}
```

In the code you can read the ADC values of the 3 potentiometers and map it into a PWM duty cycle to control the 3 LED elements to vary the color of their respective RGB LED.

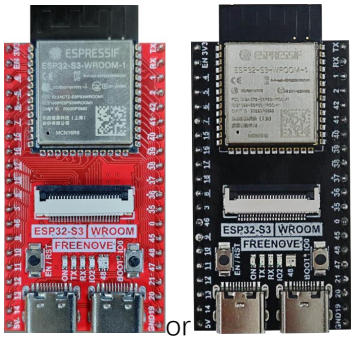
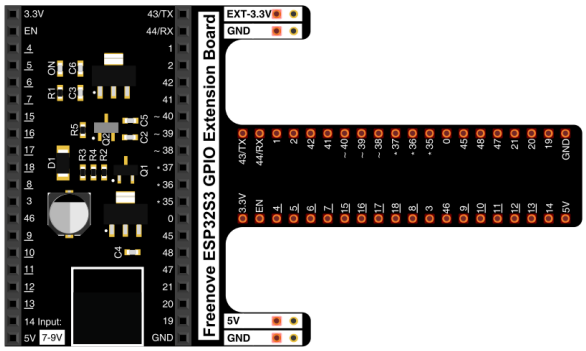
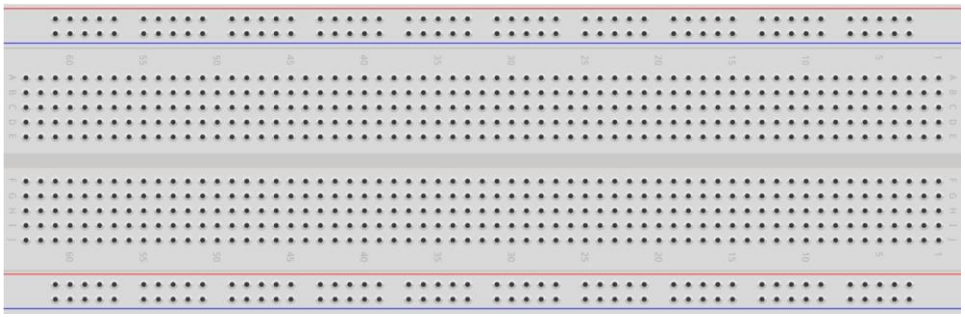
Chapter 11 Photoresistor & LED

In this chapter, we will learn how to use a photoresistor.

Project 11.1 NightLamp

A photoresistor is very sensitive to the amount of light present. We can take advantage of the characteristic to make a nightlight with the following function: when the ambient light is less (darker environment) the LED will automatically become brighter to compensate and when the ambient light is greater (brighter environment) the LED will automatically dim to compensate.

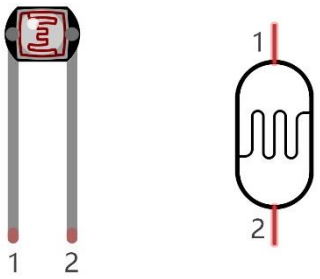
Component List

<table>
<tr><td colspan="2">ESP32-S3(N16R8)/ESP32-S3(N8R8)
 
or</td><td colspan="2">GPIO Extension Board x1
  </td></tr>
<tr><td colspan="4">Breadboard x1
  </td></tr>
<tr><td>Photoresistor x1
  </td><td colspan="2">Resistor
220Ω x1
 
10KΩ x1
  </td><td>LED x1
  </td><td>Jumper M/M x4
  </td></tr>
</table>

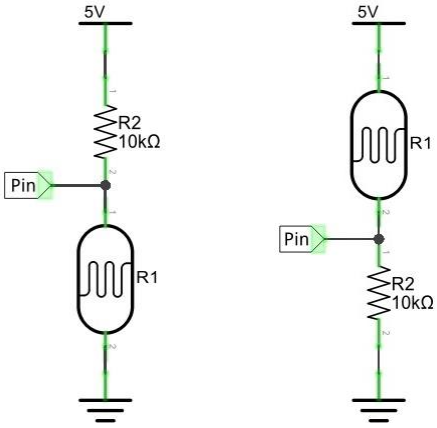
Component knowledge

Photoresistor

A photoresistor is simply a light sensitive resistor. It is an active component that decreases resistance with respect to receiving luminosity (light) on the component's light sensitive surface. A photoresistor's resistance value will change in proportion to the ambient light detected. With this characteristic, we can use a photoresistor to detect light intensity. The photoresistor and its electronic symbol are as follows.



The circuit below is used to detect the change of a photoresistor's resistance value:

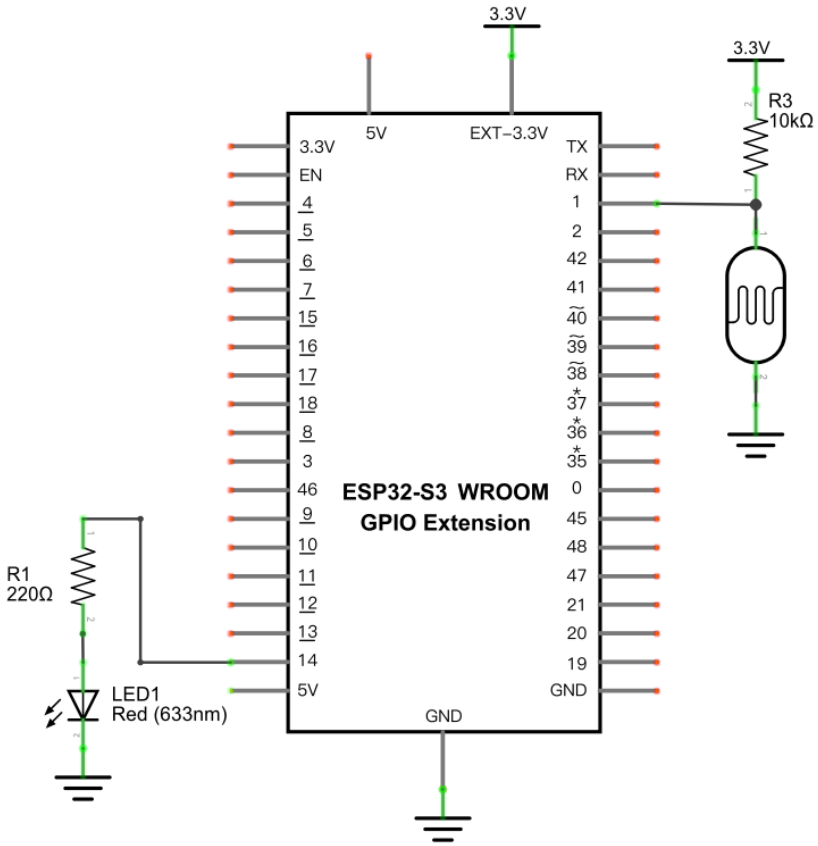


In the above circuit, when a photoresistor's resistance vale changes due to a change in light intensity, the voltage between the photoresistor and resistor R1 will also change. Therefore, the intensity of the light can be obtained by measuring this voltage.

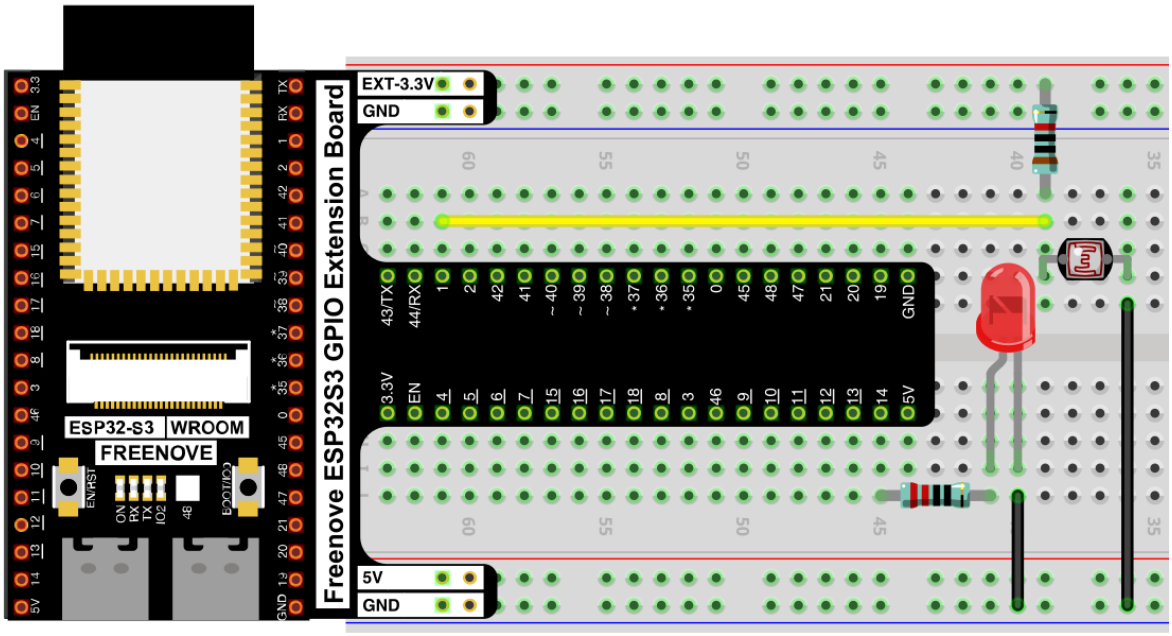
Circuit

The circuit of this project is similar to project Soft Light. The only difference is that the input signal is changed from a potentiometer to a combination of a photoresistor and a resistor.

Schematic diagram



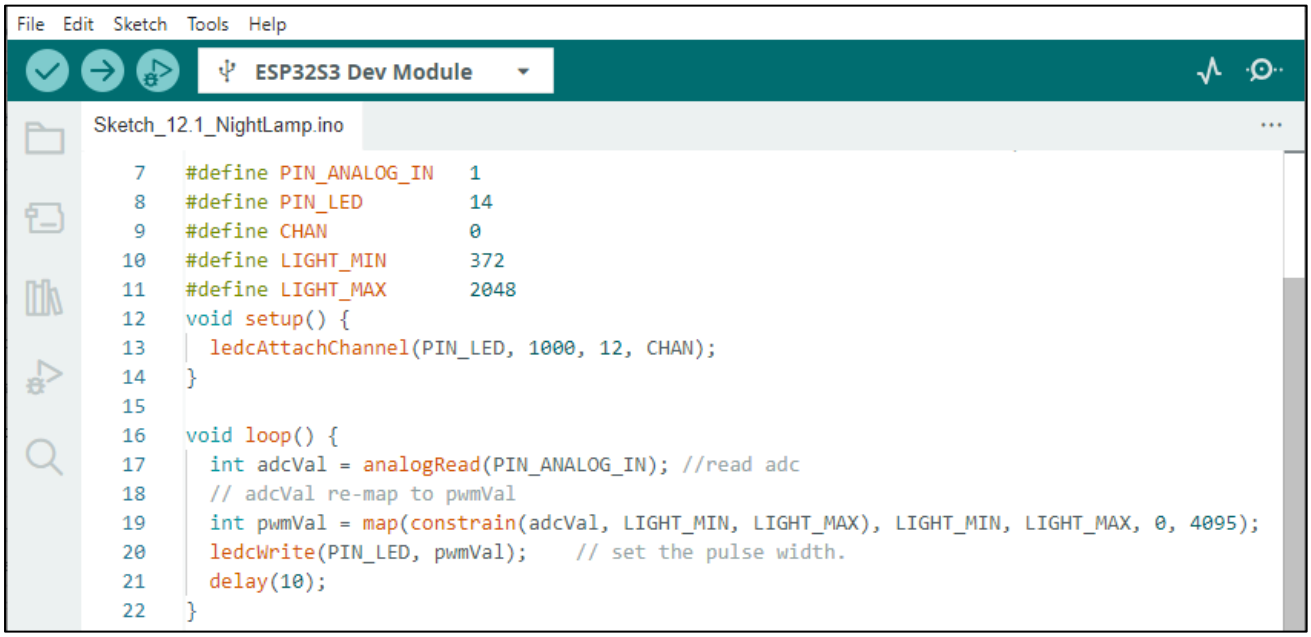
Hardware connection. If you need any support, please feel free to contact us via: support@freenove.com



Sketch

The circuit used is similar to the project Soft Light. The only difference is that the input signal of the pin of ADC changes from a potentiometer to a combination of a photoresistor and a resistor.

Sketch_11.1_Nightlamp




```
#define PIN_ANALOG_IN   1
#define PIN_LED         14
#define CHAN            0
#define LIGHT_MIN       372
#define LIGHT_MAX       2048
void setup() {
  ledcAttachChannel(PIN_LED, 1000, 12, CHAN);
}

void loop() {
  int adcVal = analogRead(PIN_ANALOG_IN); //read adc
  // adcVal re-map to pwmVal
  int pwmVal = map(constrain(adcVal, LIGHT_MIN, LIGHT_MAX), LIGHT_MIN, LIGHT_MAX, 0, 4095);
  ledcWrite(PIN_LED, pwmVal);    // set the pulse width.
  delay(10);
}
```

Download the code to ESP32-S3 WROOM, if you cover the photoresistor or increase the light shining on it, the brightness of the LED changes accordingly.

If you have any concerns, please contact us via: support@freenove.com

The following is the program code:

```
#define PIN_ANALOG_IN   1
#define PIN_LED          14
#define CHAN            0
#define LIGHT_MIN        372
#define LIGHT_MAX       2048
void setup() {
  ledcAttachChannel(PIN_LED, 1000, 12, CHAN);
}

void loop() {
  int adcVal = analogRead(PIN_ANALOG_IN); //read adc
  // adcVal re-map to pwmVal
  int pwmVal = map(constrain(adcVal, LIGHT_MIN, LIGHT_MAX), LIGHT_MIN, LIGHT_MAX, 0, 4095);
  ledcWrite(PIN_LED, pwmVal);    // set the pulse width.
  delay(10);
}
```

Reference

constrain(amt,low,high)

```
#define constrain(amt,low,high) ((amt)<(low)? (low):((amt)>(high)? (high):(amt)))
```

Constrain the value amt between low and high.

long map(long value,long fromLow,long fromHigh,long toLow,long toHigh);

Re-maps a number from one range to another. That is, a value of fromLow would get mapped to toLow, a value of fromHigh to toHigh, values in-between to values in-between, etc.

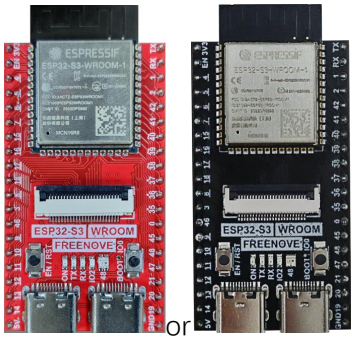
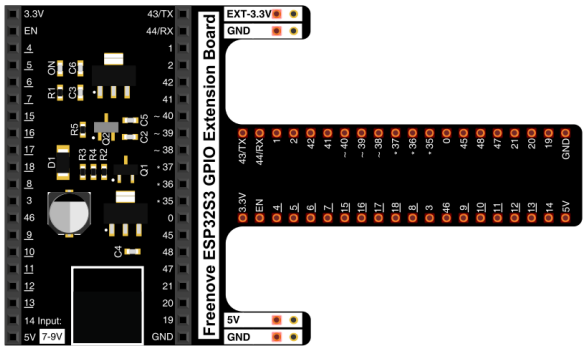
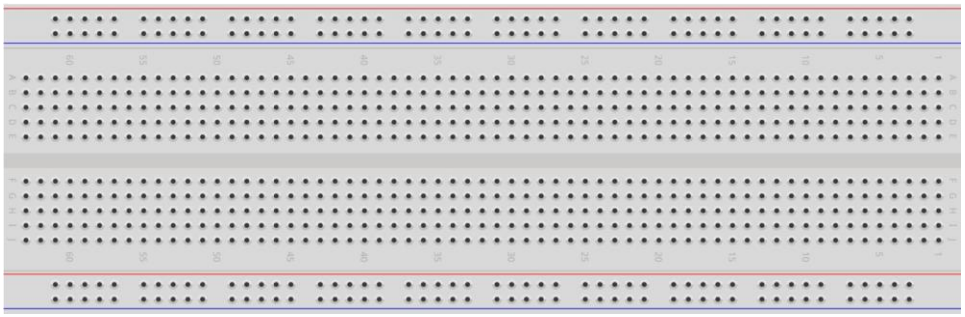
Chapter 12 Thermistor

In this chapter, we will learn about thermistors which are another kind of resistor

Project 12.1 Thermometer

A thermistor is a type of resistor whose resistance value is dependent on temperature and changes in temperature. Therefore, we can take advantage of this characteristic to make a thermometer.

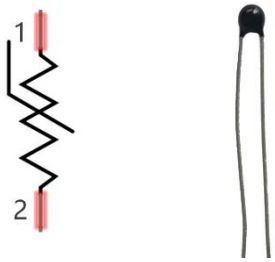
Component List

<table>
<tr><td colspan="2">ESP32-S3(N16R8)/ESP32-S3(N8R8)
 
or</td><td>GPIO Extension Board x1
  </td></tr>
<tr><td colspan="3">Breadboard x1
  </td></tr>
<tr><td>Thermistor x1
  </td><td>Resistor 10kΩ x1
  </td><td>Jumper M/M x3
  </td></tr>
</table>

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:

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

Where:

Rt is the thermistor resistance under T2 temperature;

R is the nominal resistance of thermistor under T1 temperature;

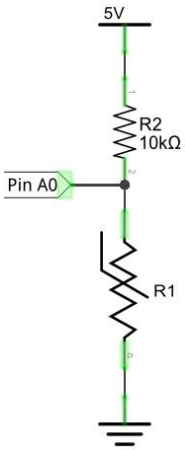
EXP[n] is nth power of E;

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.

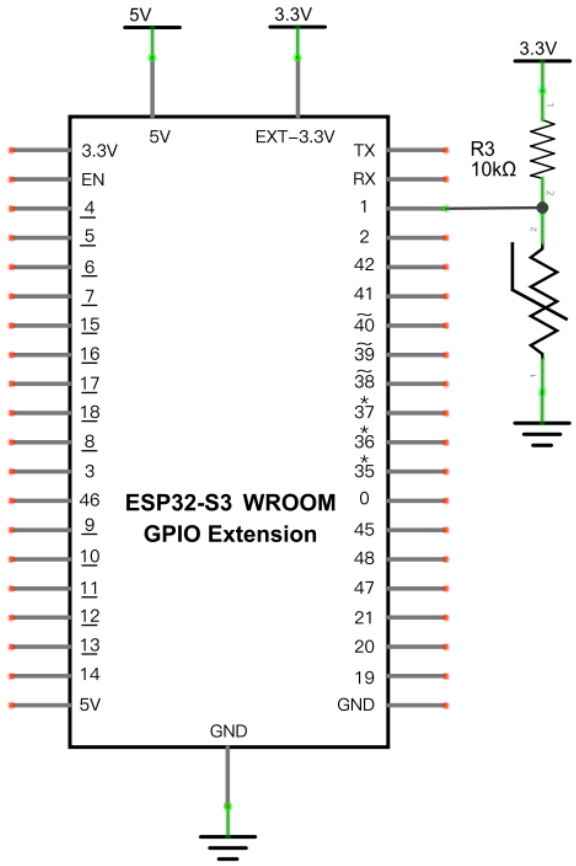
Therefore, the temperature formula can be derived as:

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

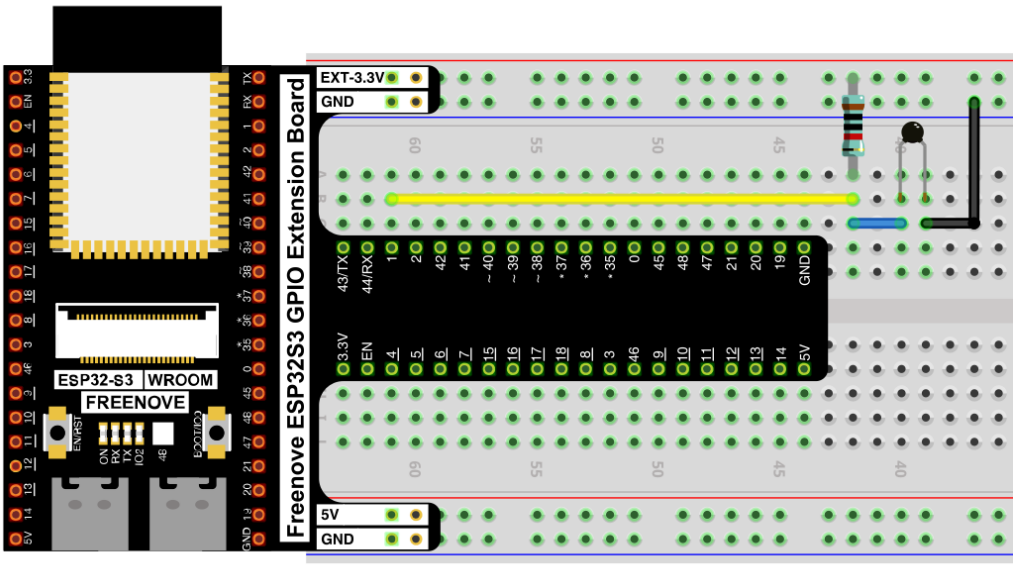
Circuit

The circuit of this project is similar to the one in the last chapter. The only difference is that the photoresistor is replaced by the thermistor.

Schematic diagram

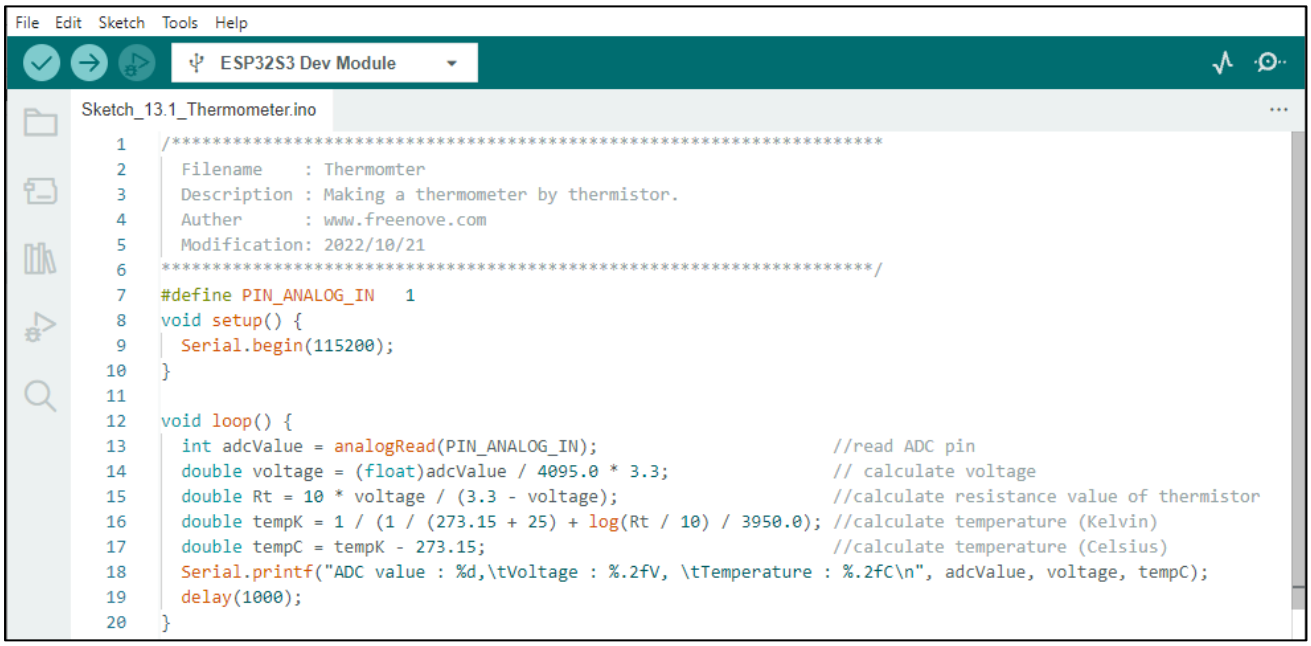


Hardware connection. If you need any support, please feel free to contact us via: support@freenove.com



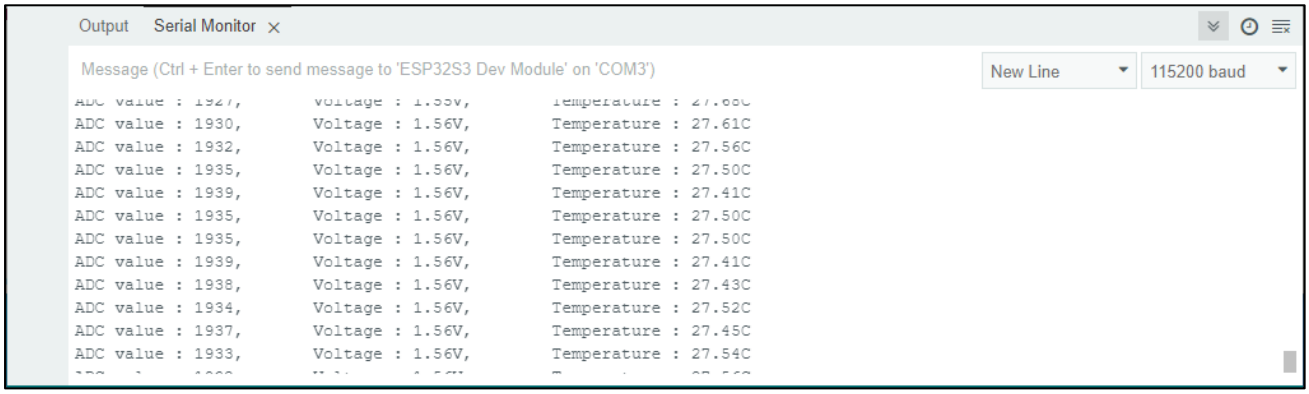
Sketch

Sketch_12.1_Thermometer




```
/**********************************************************************
  Filename    : Thermomter
  Description : Making a thermometer by thermistor.
  Auther      : www.freenove.com
  Modification: 2022/10/21
**********************************************************************/
#define PIN_ANALOG_IN   1
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);
}
```

Download the code to ESP32-S3 WROOM, the terminal window will display the current ADC value, voltage value and temperature value. Try to "pinch" the thermistor (without touching the leads) with your index finger and thumb for a brief time, you should see that the temperature value increases.



If you have any concerns, please contact us via: support@freenove.com

The following is the code:

```
#define PIN_ANALOG_IN   1
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);
}
```

In the code, GPIO1 is connected to the thermistor circuit. ESP32-S3 reads the ADC value of GPIO1, calculates the voltage and resistance value of the thermistor according to Ohm's law, and finally calculates the temperature value perceived by the thermistor according to the formula.

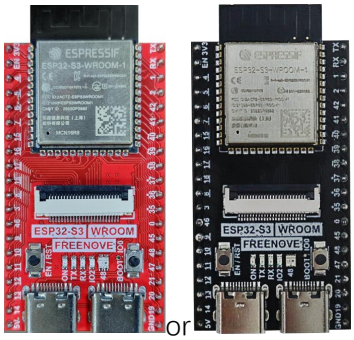
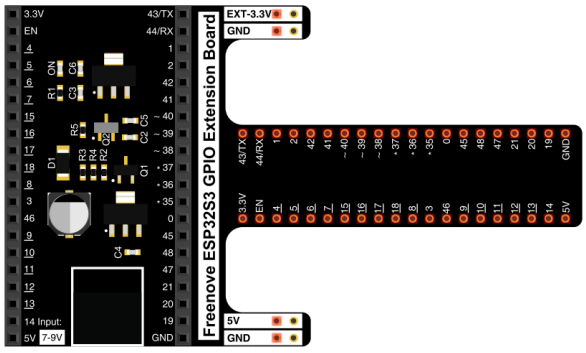
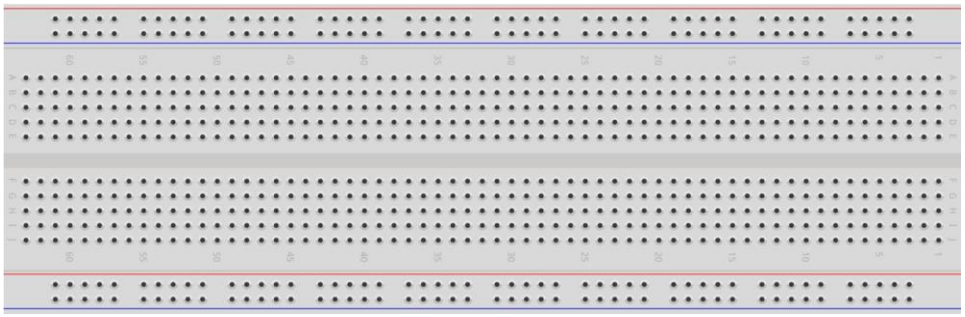
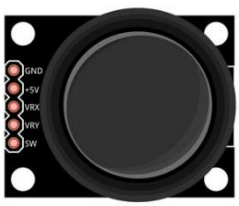
Chapter 13 Joystick

In the previous chapter, we have learned how to use rotary potentiometer. Now, let's learn a new electronic module joystick which working on the same principle as rotary potentiometer.

Project 13.1 Joystick

In this project, we will read the output data of a joystick and display it to the Terminal screen.

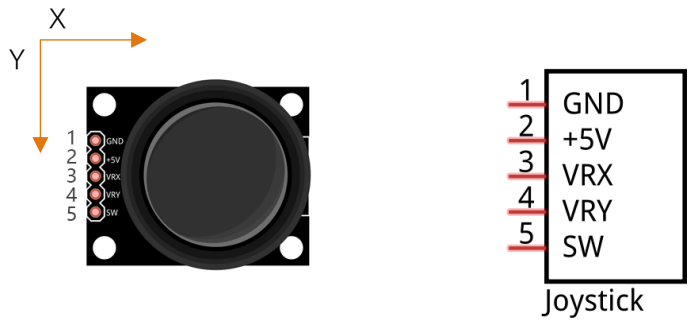
Component List

ESP32-S3(N16R8)/ESP32-S3(N8R8)  	GPIO Extension Board x1 
Breadboard x1 	
Joystick x1  	Jumper F/M x5 

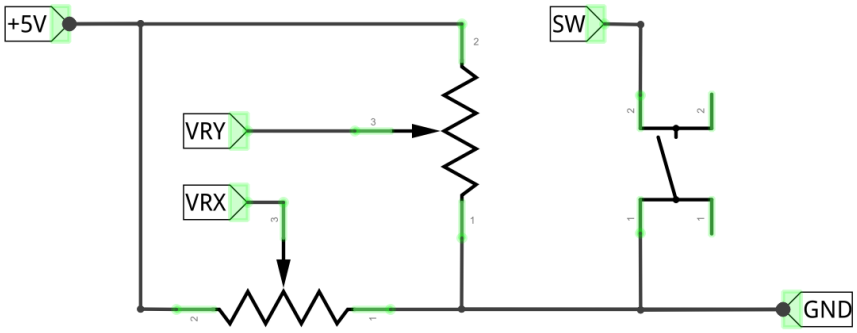
Component knowledge

Joystick

A joystick is a kind of input sensor used with your fingers. You should be familiar with this concept already as they are widely used in gamepads and remote controls. It can receive input on two axes (Y and or X) at the same time (usually used to control direction on a two dimensional plane). And it also has a third direction capability by pressing down (Z axis/direction).



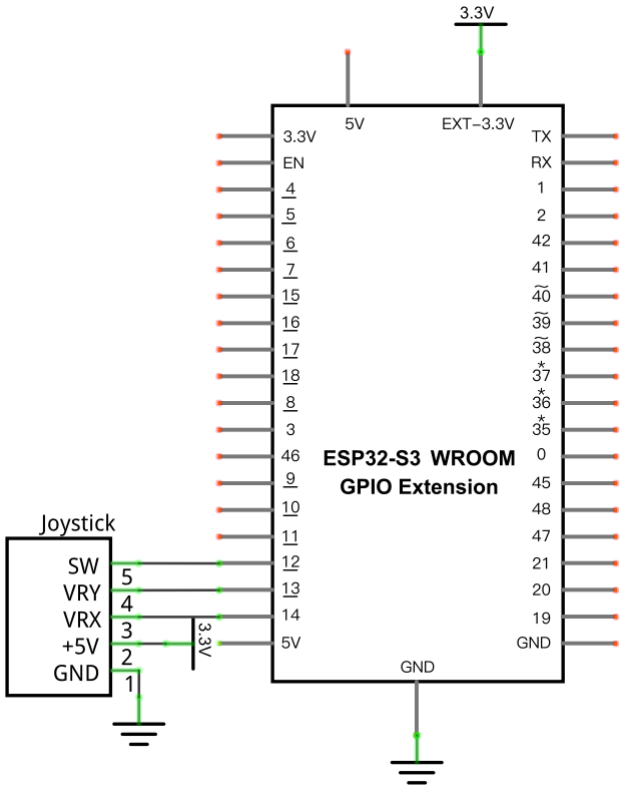
This is accomplished by incorporating two rotary potentiometers inside the joystick Module at 90 degrees of each other, placed in such a manner as to detect shifts in direction in two directions simultaneously and with a push button switch in the “vertical” axis, which can detect when a User presses on the Joystick.



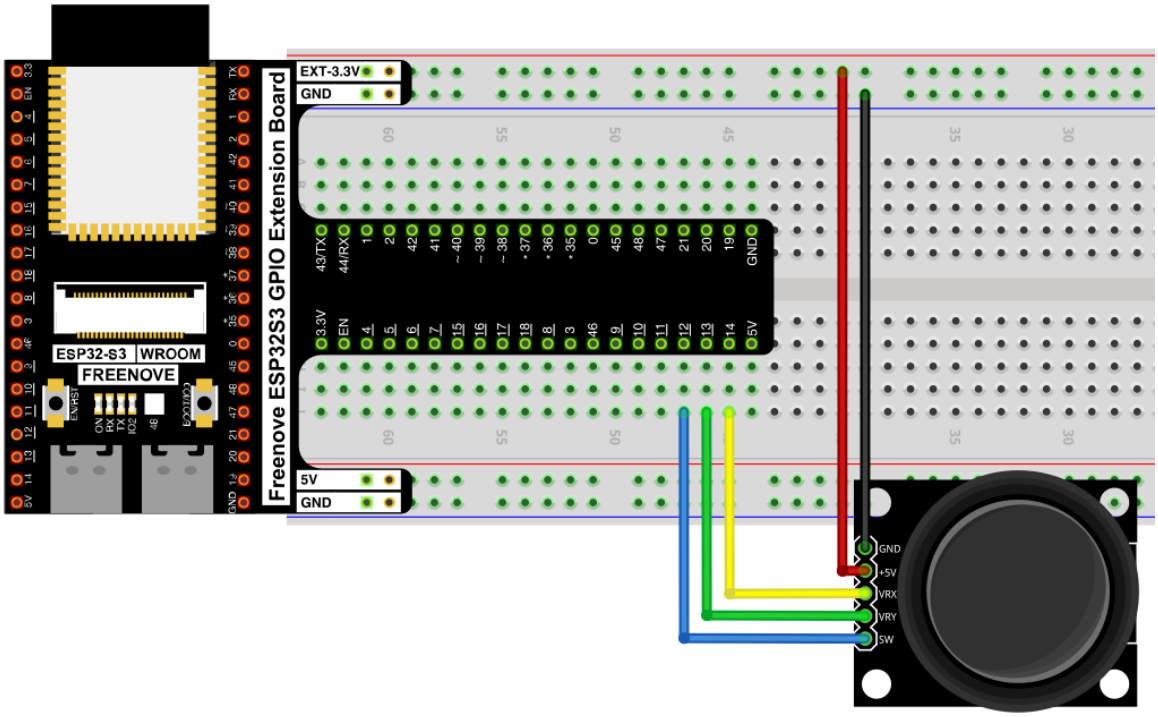
When the joystick data is read, there are some differences between the axes: data of X and Y axes is analog, which needs to use the ADC. The data of the Z axis is digital, so you can directly use the GPIO to read this data or you have the option to use the ADC to read this.

Circuit

Schematic diagram



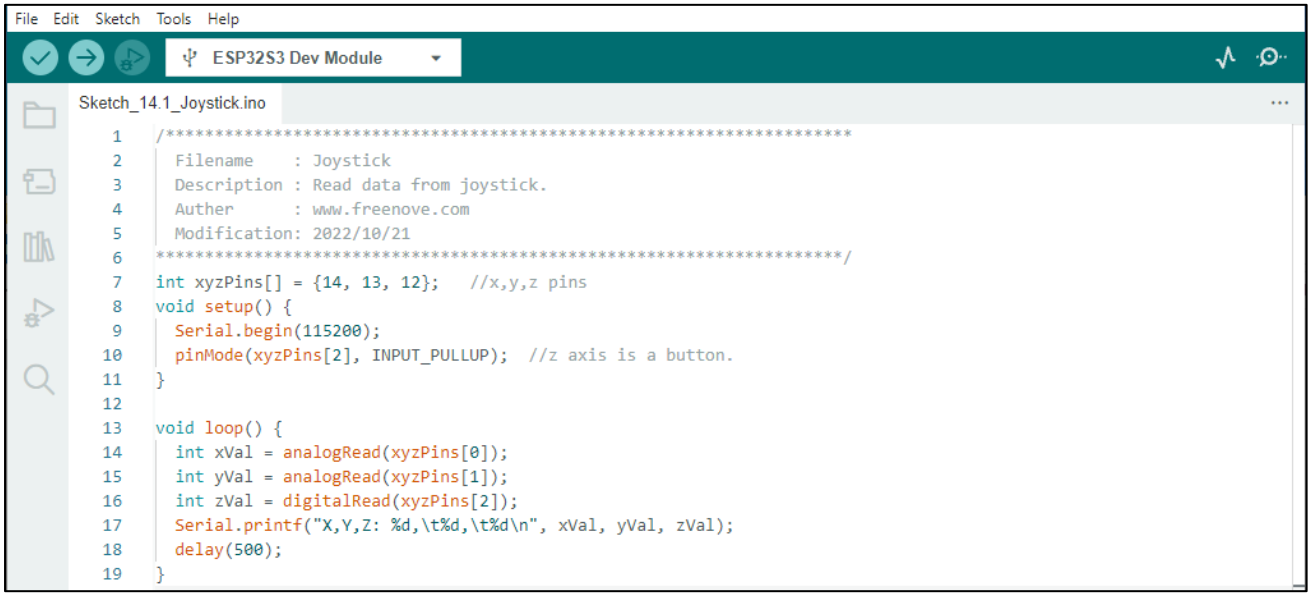
Hardware connection. If you need any support, please feel free to contact us via: support@freenove.com



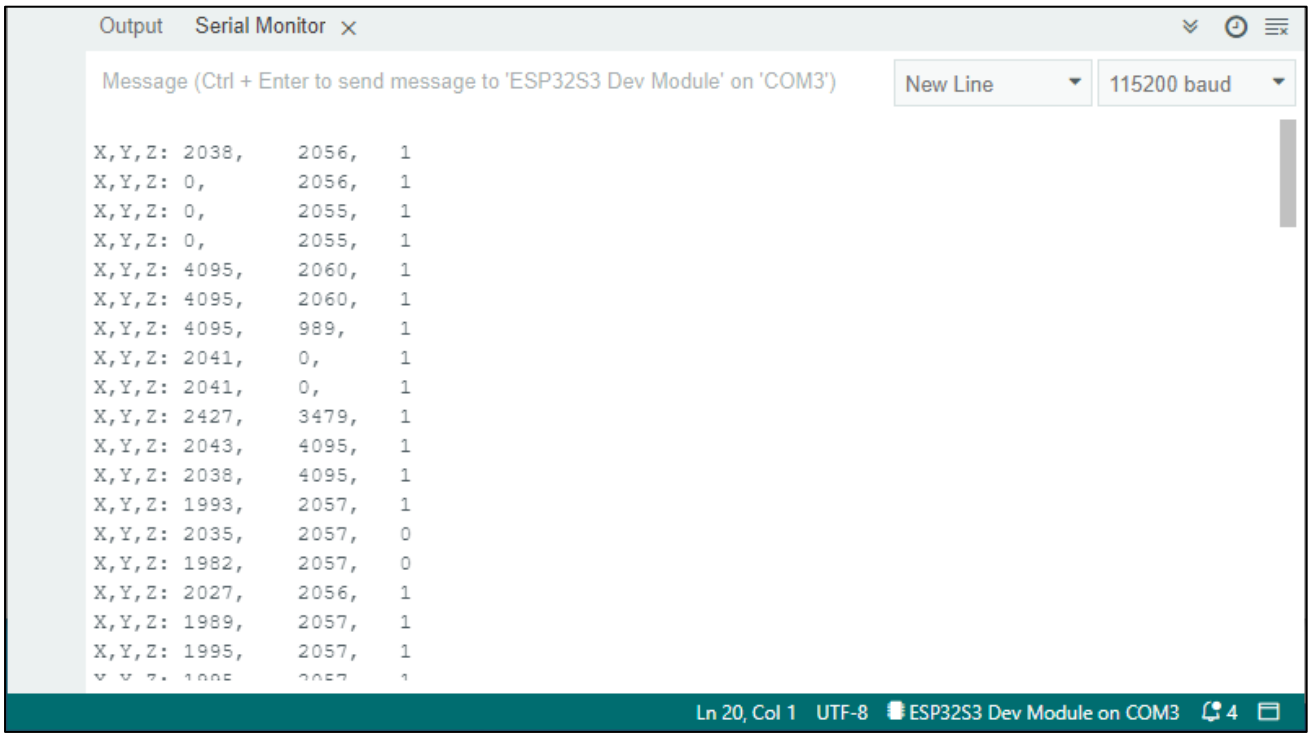
Sketch

In this project's code, we will read the ADC values of X and Y axes of the joystick, and read digital quality of the Z axis, then display these out in terminal.

Sketch_13.1_Joystick



Download the code to ESP32-S3 WROOM, open the serial port monitor, the baud rate is 115200, as shown in the figure below, shift (moving) the joystick or pressing it down will make the data change.



The following is the code:

```
int xyzPins[] = {14, 13, 12};   //x,y,z pins
void setup() {
  Serial.begin(115200);
  pinMode(xyzPins[2], INPUT_PULLUP);  //z axis is a button.
}

void loop() {
  int xVal = analogRead(xyzPins[0]);
  int yVal = analogRead(xyzPins[1]);
  int zVal = digitalRead(xyzPins[2]);
  Serial.printf("X,Y,Z: %d,\t%d,\t%d\n", xVal, yVal, zVal);
  delay(500);
}
```

In the code, configure xyzPins[2] to pull-up input mode. In loop(), use analogRead () to read the value of axes X and Y and use digitalRead () to read the value of axis Z, then display them.

```
  int xVal = analogRead(xyzPins[0]);
  int yVal = analogRead(xyzPins[1]);
  int zVal = digitalRead(xyzPins[2]);
  Serial.printf("X,Y,Z: %d,\t%d,\t%d\n", xVal, yVal, zVal);
  delay(500);
```

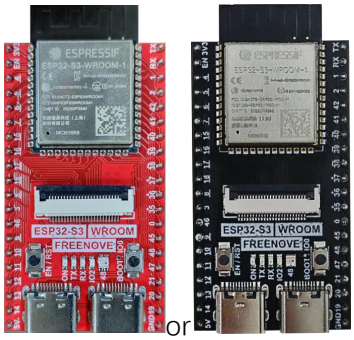
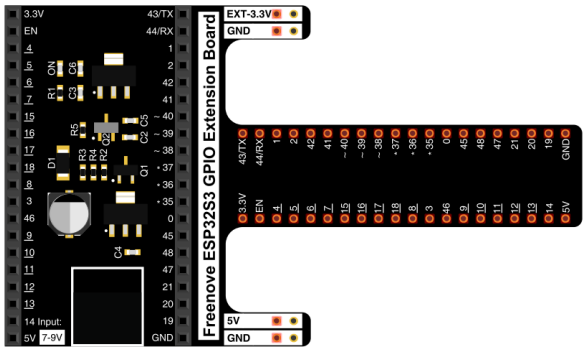
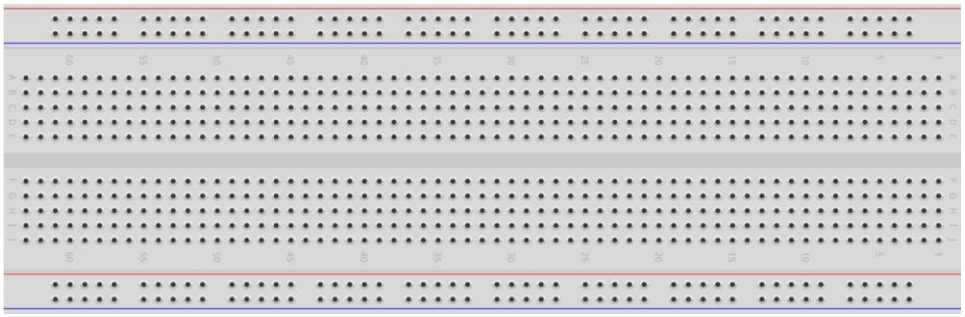
Chapter 14 74HC595 & LED Bar Graph

We have used LED bar graph to make a flowing water light, in which 10 GPIO ports of ESP32-S3 is occupied. More GPIO ports mean that more peripherals can be connected to ESP32-S3, so GPIO resource is very precious. Can we make flowing water light with less GPIO? In this chapter, we will learn a component, 74HC595, which can achieve the target.

Project 14.1 Flowing Water Light

Now let's learn how to use the 74HC595 IC chip to make a flowing water light using less GPIO.

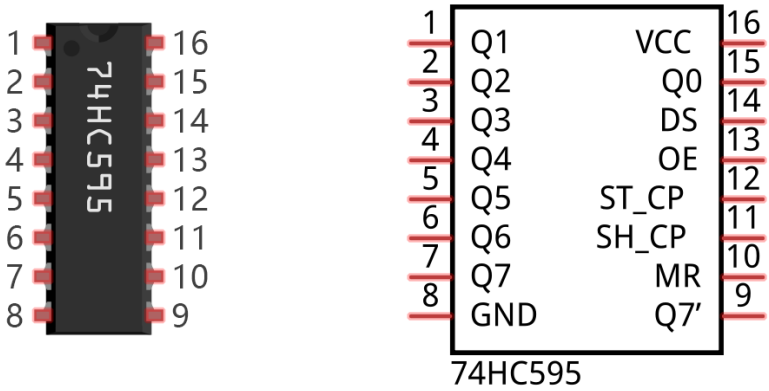
Component List

<table>
<tr><td colspan="2">ESP32-S3(N16R8)/ESP32-S3(N8R8)
 
or</td><td colspan="2">GPIO Extension Board x1
  </td></tr>
<tr><td colspan="4">Breadboard x1
  </td></tr>
<tr><td>74HC595 x1
  </td><td>LED Bar Graph x1
  </td><td>Resistor 220Ω x8
  </td><td>Jumper M/M x15
  </td></tr>
</table>

Related knowledge

74HC595

A 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 a ESP32-S3. At least 3 ports are required to control the 8 ports of the 74HC595 chip.



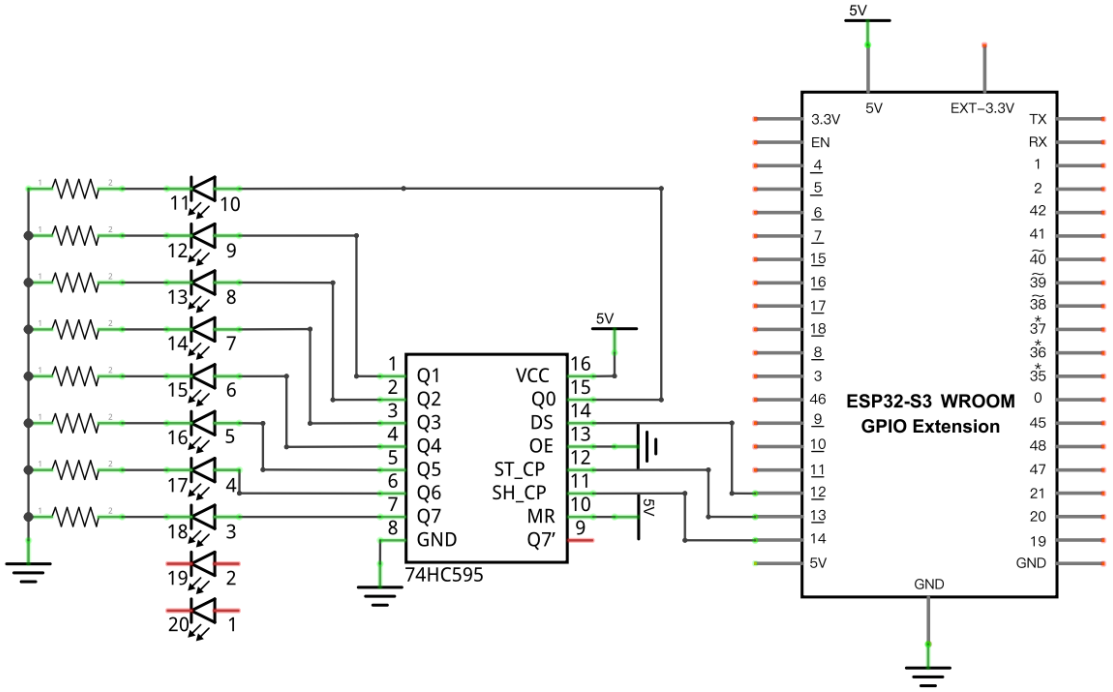
The ports of the 74HC595 chip are described as follows:

Pin name	GPIO number	Description
Q0-Q7	15, 1-7	Parallel data output
VCC	16	The positive electrode of power supply, the voltage is 2~6V
GND	8	The negative electrode of power supply
DS	14	Serial data Input
OE	13	Enable output, When this pin is in high level, Q0-Q7 is in high resistance state When this pin is in low level, Q0-Q7 is in output mode
ST_CP	12	Parallel Update Output: when its electrical level is rising, it will update the parallel data output.
SH_CP	11	Serial shift clock: when its electrical level is rising, serial data input register will do a shift.
MR	10	Remove shift register: When this pin is in low level, the content in shift register will be cleared.
Q7'	9	Serial data output: it can be connected to more 74HC595 in series.

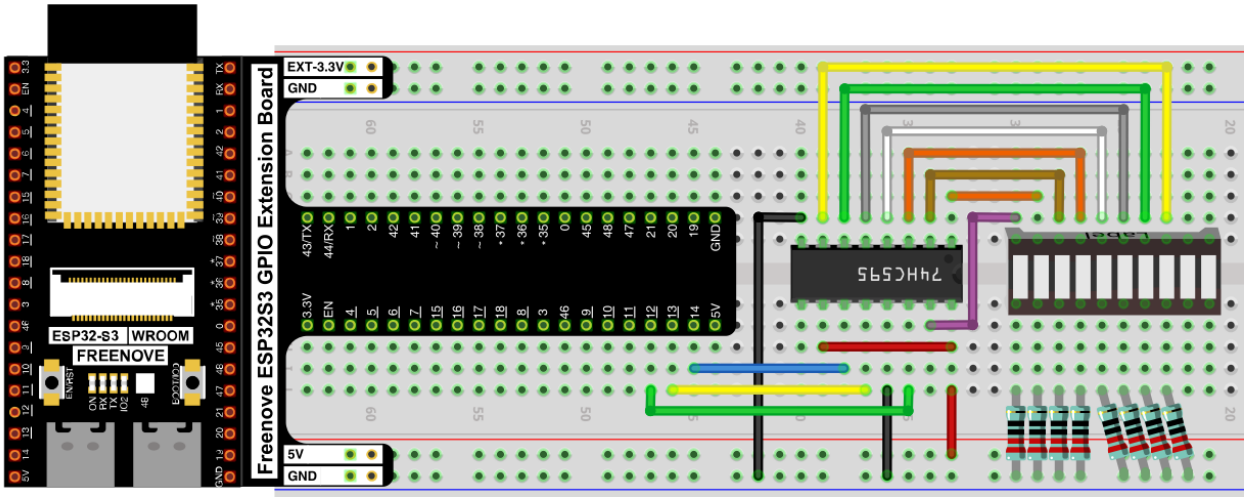
For more detail, please refer to the datasheet on the 74HC595 chip.

Circuit

Schematic diagram



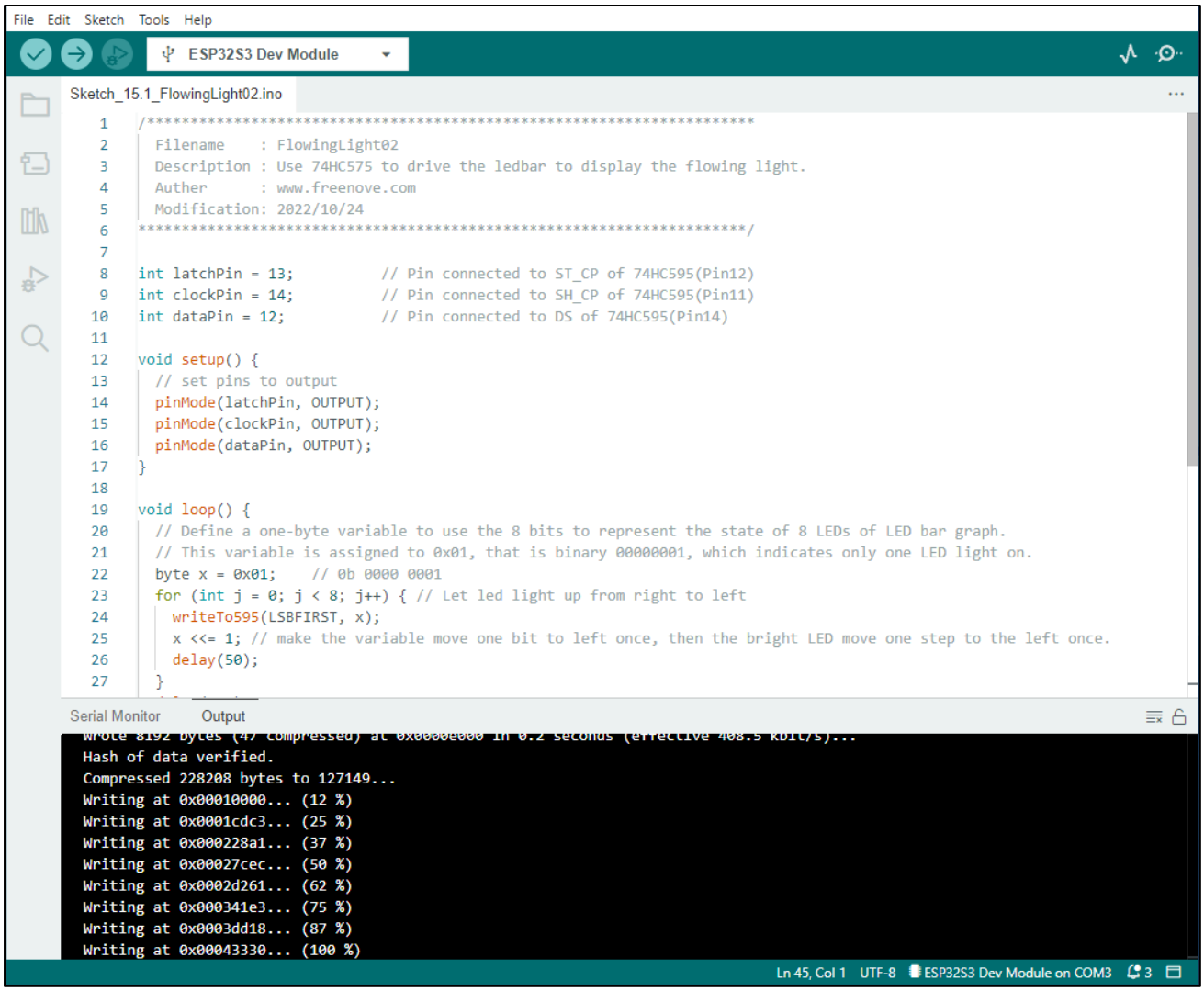
Hardware connection. If you need any support, please feel free to contact us via: support@freenove.com



Sketch

In this project, we will make a flowing water light with a 74HC595 chip to learn about its functions.

Sketch_14.1_FlowingLight2



Download the code to ESP32-S3 WROOM. You will see that LED bar graph starts with the flowing water pattern flashing from left to right and then back from right to left.

If you have any concerns, please contact us via: support@freenove.com

The following is the program code:

```
int latchPin = 13;          // Pin connected to ST_CP of 74HC595(Pin12)
int clockPin = 14;          // Pin connected to SH_CP of 74HC595(Pin11)
int dataPin = 12;           // Pin connected to DS of 74HC595(Pin14)

void setup() {
  // set pins to output
  pinMode(latchPin, OUTPUT);
  pinMode(clockPin, OUTPUT);
```

```
9   pinMode(dataPin, OUTPUT);
10  }
11
12  void loop() {
13    // Define a one-byte variable to use the 8 bits to represent the state of 8 LEDs of LED bar
14  graph.
15    // This variable is assigned to 0x01, that is binary 00000001, which indicates only one LED
16  light on.
17    byte x = 0x01;    // 0b 0000 0001
18    for (int j = 0; j < 8; j++) { // Let led light up from right to left
19      writeTo595(LSBFIRST, x);
20      x <<= 1; // make the variable move one bit to left once, then the bright LED move one step
21  to the left once.
22      delay(50);
23    }
24    delay(1000);
25    x = 0x80;       //0b 1000 0000
26    for (int j = 0; j < 8; j++) { // Let led light up from left to right
27      writeTo595(LSBFIRST, x);
28      x >>= 1;
29      delay(50);
30    }
31    delay(1000);
32  }
33  void writeTo595(int order, byte _data ) {
34    // Output low level to latchPin
35    digitalWrite(latchPin, LOW);
36    // Send serial data to 74HC595
37    shiftOut(dataPin, clockPin, order, _data);
38    // Output high level to latchPin, and 74HC595 will update the data to the parallel output
39  port.
40    digitalWrite(latchPin, HIGH);
41  }
```

In the code, we configure three pins to control the 74HC595 chip and define a one-byte variable to control the state of the 8 LEDs (in the LED bar graph Module) through the 8 bits of the variable. The LEDs light ON when the corresponding bit is 1. If the variable is assigned to 0x01, that is 00000001 in binary, there will be only one LED ON.

```
17  x=0x01;
```

In the loop(), use “for” loop to send x to 74HC595 output pin to control the LED. In "for" loop, x will shift one bit to the LEFT in one cycle, then when data of x is sent to 74HC595, the LED that is turned ON will move one bit to the LEFT once.

```
18    for (int j = 0; j < 8; j++) { // Let led light up from right to left
```

```
19      writeTo595(LSBFIRST, x);
20      x <<= 1;
21      delay(50);
22    }
```

In second “for” loop, the situation is the same. The difference is that x is shift from 0x80 to the RIGHT in order.

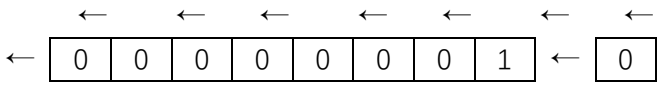
The subfunction writeTo595() is used to write data to 74HC595 and immediately output on the port of 74HC595.

Reference

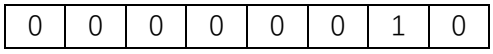
<< operator

"<<" is the left shift operator, which can make all bits of 1 byte shift by several bits to the left (high) direction and add 0 on the right (low). For example, shift binary 00000001 by 1 bit to left:

```
byte x = 1 << 1;
```

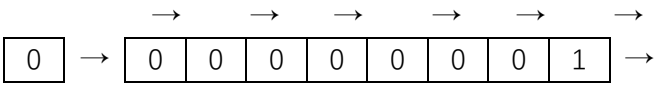


The result of x is 2（binary 00000010）.

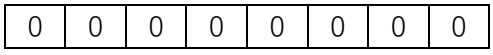


There is another similar operator" >>". For example, shift binary 00000001 by 1 bit to right:

```
byte x = 1 >> 1;
```



The result of x is 0（00000000）.



X <<= 1 is equivalent to x = x << 1 and x >>= 1 is equivalent to x = x >> 1

void shiftOut(uint8_t dataPin, uint8_t clockPin, uint8_t bitOrder, uint8_t val);

This is used to shift an 8-bit data value in with the data appearing on the dataPin and the clock being sent out on the clockPin. Order is as above. The data is sampled after the cPin goes high. (So clockPin high, sample data, clockPin low, repeat for 8 bits) The 8-bit value is returned by the function.

Parameters

dataPin: the pin on which to output each bit. Allowed data types: int.

clockPin: the pin to toggle once the dataPin has been set to the correct value. Allowed data types: int.

bitOrder: which order to shift out the bits; either MSBFIRST or LSBFIRST. (Most Significant Bit First, or, Least Significant Bit First).

value: the data to shift out. Allowed data types: byte.

For more details about shift function, please refer to:

https://www.arduino.cc/reference/en/language/functions/advanced-io/shiftout/

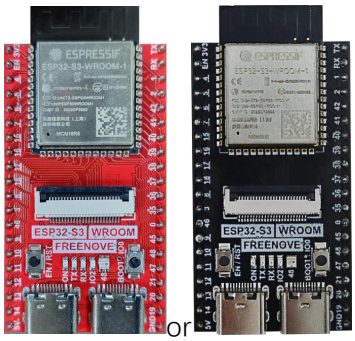
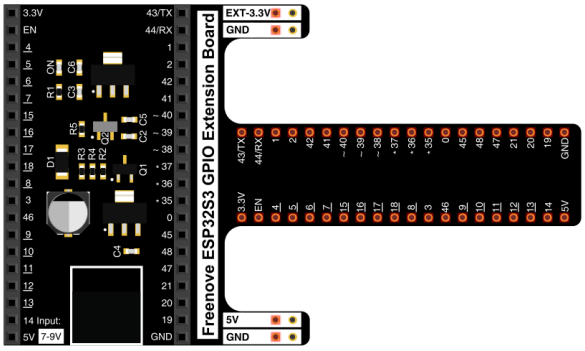
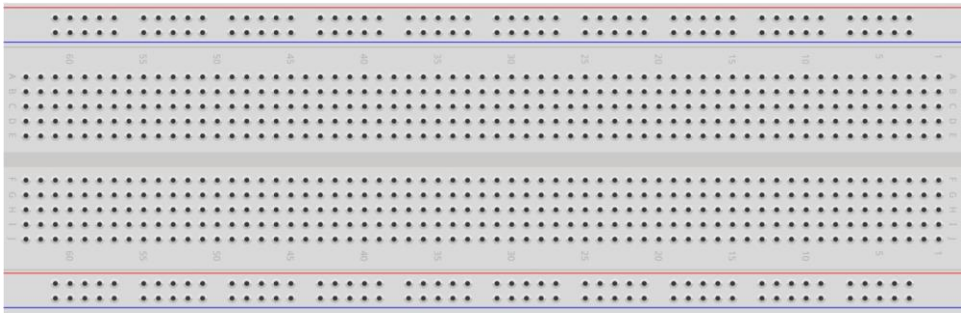
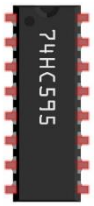
Chapter 15 74HC595 & 7-Segment Display.

In this chapter, we will introduce the 7-Segment Display.

Project 15.1 7-Segment Display.

We will use 74HC595 to control 7-segment display and make it display hexadecimal character "0-F".

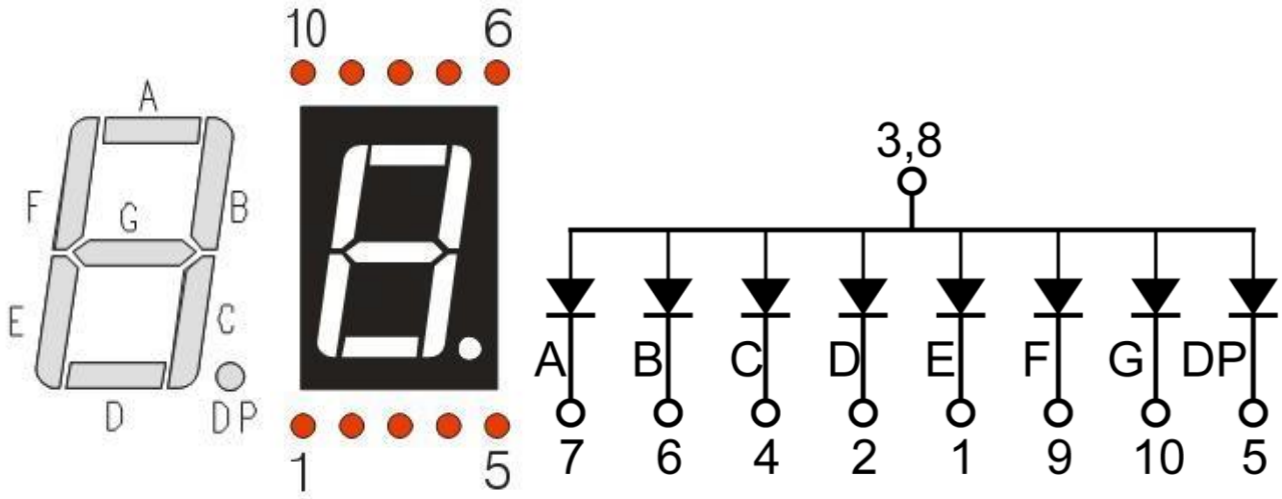
Component List

<table>
<tr><td colspan="2">ESP32-S3(N16R8)/ESP32-S3(N8R8)
 
or</td><td colspan="2">GPIO Extension Board x1
  </td></tr>
<tr><td colspan="4">Breadboard x1
  </td></tr>
<tr><td>74HC595 x1
  </td><td>7-segment display x1
  </td><td>Resistor 220Ω x8
  </td><td>Jumper M/M
  </td></tr>
</table>

Component knowledge

7-segment display

A 7-segment display is a digital electronic display device. There is a figure "8" and a decimal point represented, which consists of 8 LEDs. The LEDs have a common anode and individual cathodes. Its internal structure and pin designation diagram is shown below:



As we can see in the above circuit diagram, we can control the state of each LED separately. Also, by combining LEDs with different states of ON and OFF, we can display different characters (Numbers and Letters). For example, to display a “0”: we need to turn ON LED segments A, B, C, D, E and F, and turn OFF LED segments G and DP.

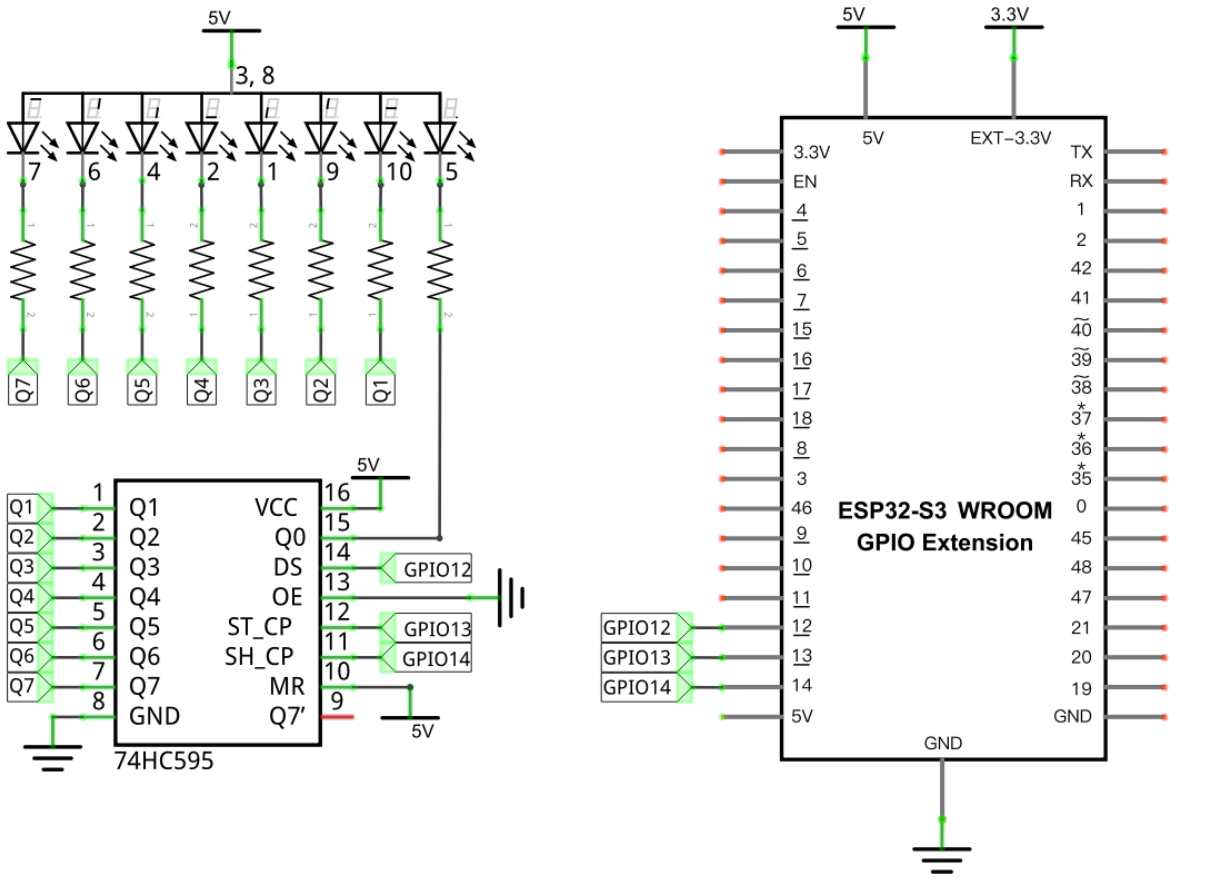
In this project, we will use a 7-Segment Display with a common anode. Therefore, when there is an input low level to a LED segment the LED will turn ON. Defining segment “A” as the lowest level and segment “DP” as the highest level, from high to low would look like this: “DP”, “G”, “F”, “E”, “D”, “C”, “B”, “A”. Character "0" corresponds to the code: 1100 0000b=0xc0.

For detailed code values, please refer to the following table (common anode).

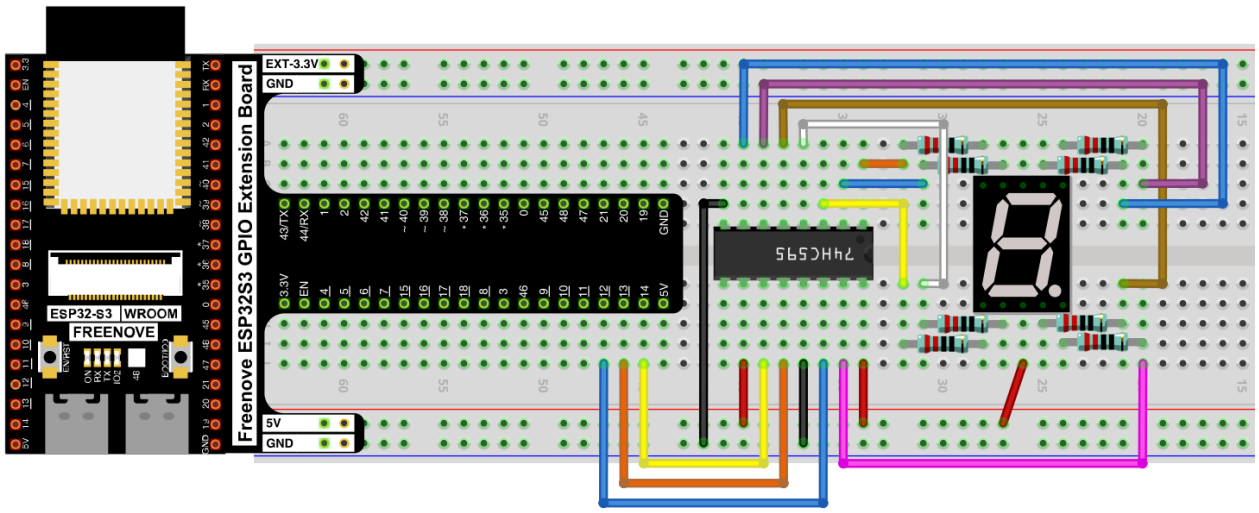
CHAR	DP	G	F	E	D	C	B	A	Hex	ASCII
0	1	1	0	0	0	0	0	0	0xc0	1100 0000
1	1	1	1	1	1	0	0	1	0xf9	1111 1001
2	1	0	1	0	0	1	0	0	0xa4	1010 0100
3	1	0	1	1	0	0	0	0	0xb0	1011 0000
4	1	0	0	1	1	0	0	1	0x99	1001 1001
5	1	0	0	1	0	0	1	0	0x92	1001 0010
6	1	0	0	0	0	0	1	0	0x82	1000 0010
7	1	1	1	1	1	0	0	0	0xf8	1111 1000
8	1	0	0	0	0	0	0	0	0x80	1000 0000
9	1	0	0	1	0	0	0	0	0x90	1001 0000
A	1	0	0	0	1	0	0	0	0x88	1000 1000
B	1	0	0	0	0	0	1	1	0x83	1000 0011
C	1	1	0	0	0	1	1	0	0xc6	1100 0110
D	1	0	1	0	0	0	0	1	0xa1	1010 0001
E	1	0	0	0	0	1	1	0	0x86	1000 0110
F	1	0	0	0	1	1	1	0	0x8e	1000 1110

Circuit

Schematic diagram



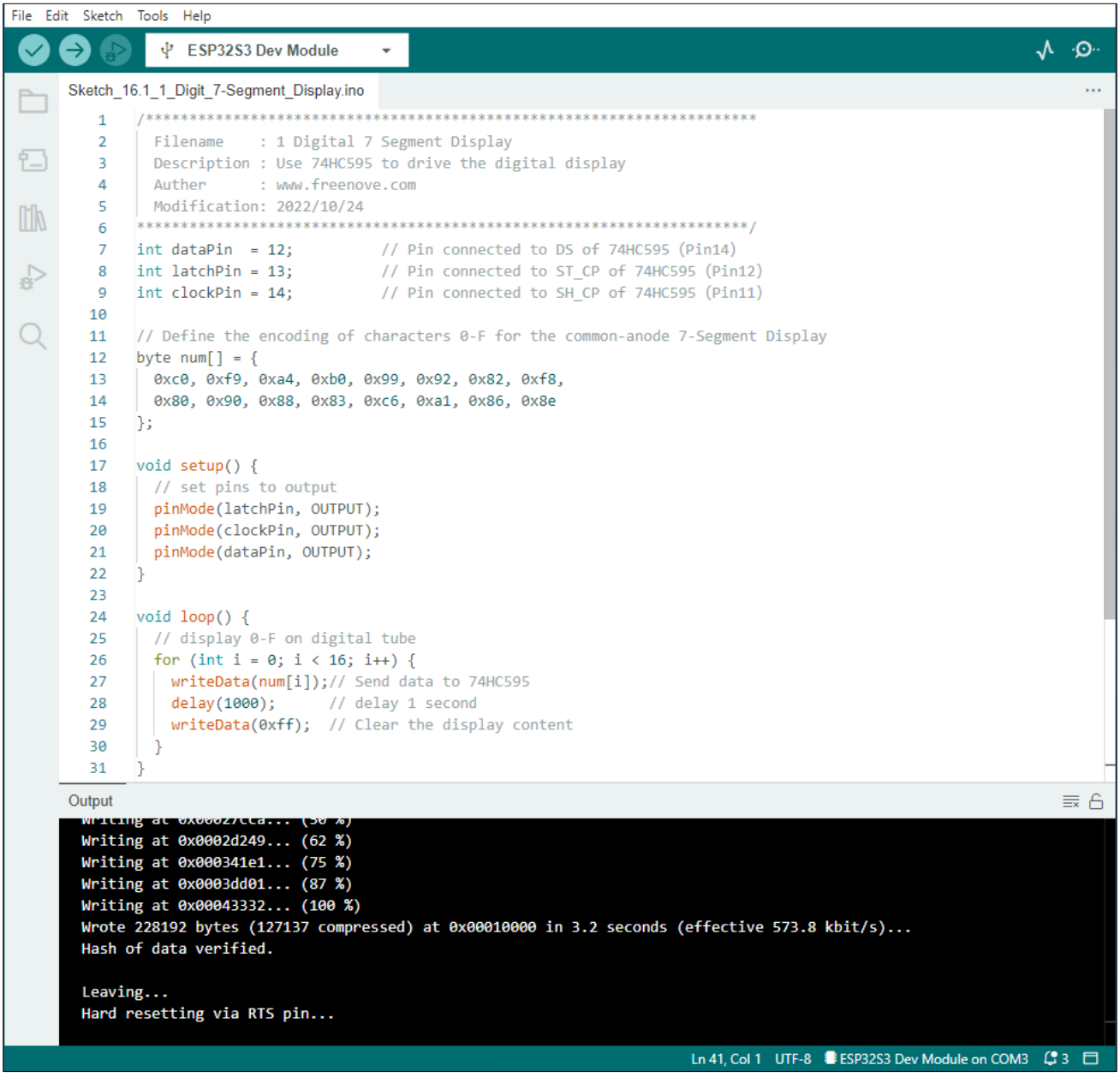
Hardware connection. If you need any support, please feel free to contact us via: support@freenove.com



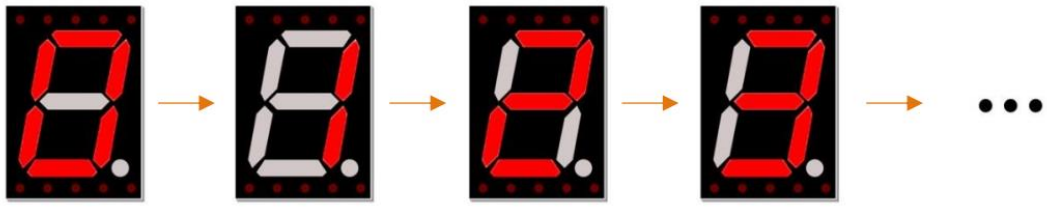
Sketch

In this section, the 74HC595 is used in the same way as in the previous section, but with different values transferred. We can learn how to master the digital display by sending the coded value of "0" - "F".

Sketch_15.1_7_Segment_Display



Verify and upload the code, and you'll see a 1-bit, 7-segment display displaying 0-f in a loop.



The following is the program code:

```
int dataPin  = 12;          // Pin connected to DS of 74HC595 (Pin14)
int latchPin = 13;          // Pin connected to ST_CP of 74HC595 (Pin12)
int clockPin = 14;          // Pin connected to SH_CP of 74HC595 (Pin11)
// Define the encoding of characters 0-F for the common-anode 7-Segment Display
byte num[] = {
  0xc0, 0xf9, 0xa4, 0xb0, 0x99, 0x92, 0x82, 0xf8,
  0x80, 0x90, 0x88, 0x83, 0xc6, 0xa1, 0x86, 0x8e
};

void setup() {
  // set pins to output
  pinMode(latchPin, OUTPUT);
  pinMode(clockPin, OUTPUT);
  pinMode(dataPin, OUTPUT);
}

void loop() {
  // display 0-F on digital tube
  for (int i = 0; i < 16; i++) {
    writeData(num[i]);// Send data to 74HC595
    delay(1000);      // delay 1 second
    writeData(0xff);  // Clear the display content
  }
}

void writeData(int value) {
  // Make latchPin output low level
  digitalWrite(latchPin, LOW);
  // Send serial data to 74HC595
  shiftOut(dataPin, clockPin, LSBFIRST, value);
  // Make latchPin output high level
  digitalWrite(latchPin, HIGH);
}
```

First, put encoding of “0”- “F” into the array.

```
// Define the encoding of characters 0-F for the common-anode 7-Segment Display
byte num[] = {
  0xc0, 0xf9, 0xa4, 0xb0, 0x99, 0x92, 0x82, 0xf8,
  0x80, 0x90, 0x88, 0x83, 0xc6, 0xa1, 0x86, 0x8e
};
```

Then, in the loop, we transfer the member of the “num” to 74HC595 by calling the writeData function, so that the digital tube displays what we want. After each display, “0xff” is used to eliminate the previous effect and prepare for the next display.

```
void loop() {
  // display 0-F on digital tube
  for (int i = 0; i < 16; i++) {
    writeData(num[i]);// Send data to 74HC595
    delay(1000);      // delay 1 second
    writeData(0xff);  // Clear the display content
  }
}
```

In the shiftOut() function, whether to use LSBFIRST or MSBFIRST as the parameter depends on the physical situation.

```
void writeData(int value) {
  // Make latchPin output low level
  digitalWrite(latchPin, LOW);
  // Send serial data to 74HC595
  shiftOut(dataPin, clockPin, LSBFIRST, value);
  // Make latchPin output high level, then 74HC595 will update data to parallel output
  digitalWrite(latchPin, HIGH);
}
```

If you want to display the decimal point, make the highest bit of each array become 0, which can be implemented easily by num[i]&0x7f.

```
shiftOut(dataPin,clockPin, LSBFIRST, value & 0x7f);
```

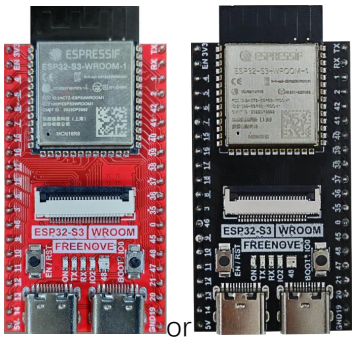
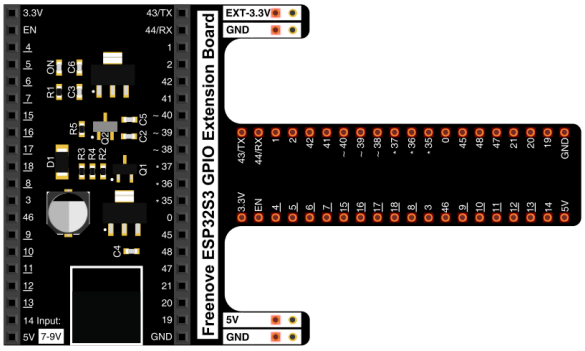
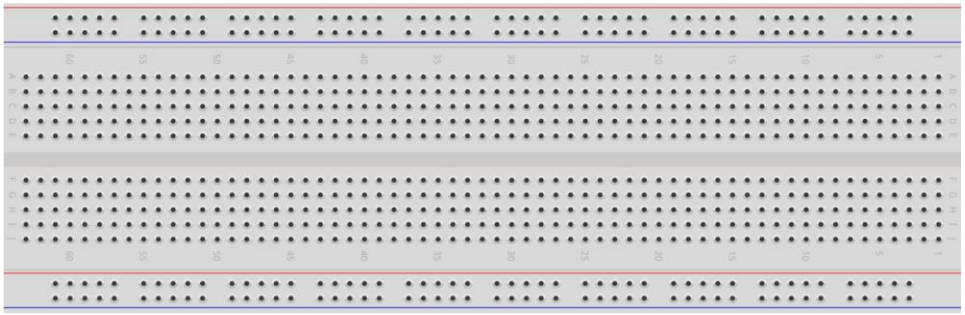
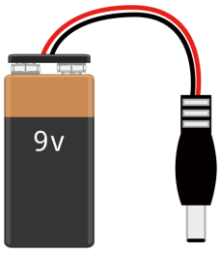
Chapter 16 Relay & Motor

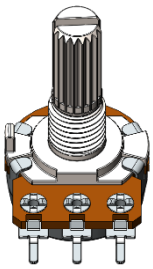
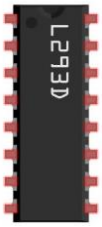
In this chapter, we will learn a kind of special switch module, relay module.

Project 16.1 Control Motor with Potentiometer

Control the direction and speed of the motor with a potentiometer.

Component List

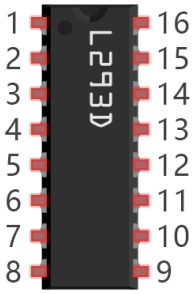
ESP32-S3(N16R8)/ESP32-S3(N8R8) 	GPIO Extension Board x1 
Breadboard x1 	
Jumper M/M 	9V battery (prepared by yourself) & battery line 

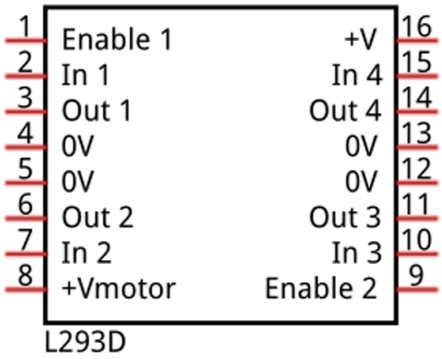
Rotary potentiometer x1	Motor x1	L293D
		

Component knowledge

L293D

L293D is an IC chip (Integrated Circuit Chip) with a 4-channel motor drive. You can drive a unidirectional DC motor with 4 ports or a bi-directional DC motor with 2 ports or a stepper motor (stepper motors are covered later in this Tutorial).



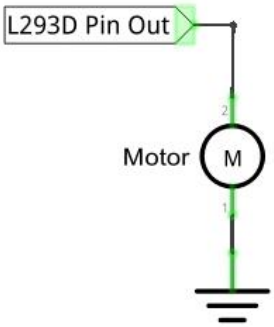


Port description of L293D module is as follows:

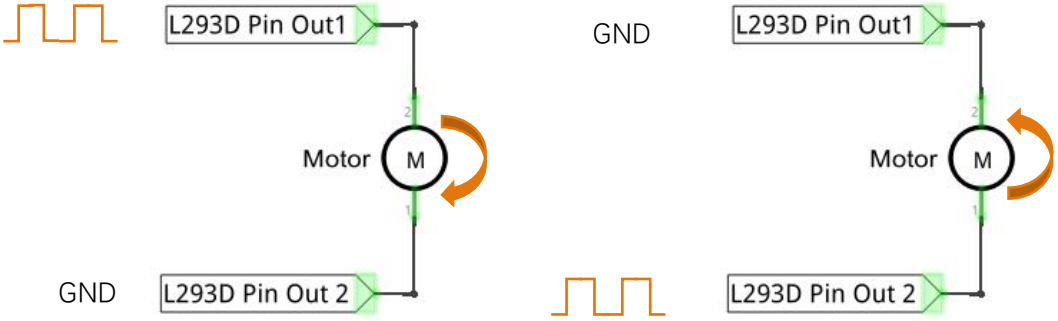
Pin name	Pin number	Description
In x	2, 7, 10, 15	Channel x digital signal input pin
Out x	3, 6, 11, 14	Channel x output pin, input high or low level according to In x pin, get connected to +Vmotor or 0V
Enable1	1	Channel 1 and channel 2 enable pin, high level enable
Enable2	9	Channel 3 and channel 4 enable pin, high level enable
0V	4, 5, 12, 13	Power cathode (GND)
+V	16	Positive electrode (VCC) of power supply, supply voltage 3.0~36V
+Vmotor	8	Positive electrode of load power supply, provide power supply for the Out pin x, the supply voltage is +V~36V

For more detail, please refer to the datasheet for this IC Chip.

When using L293D to drive DC motor, there are usually two connection options.
The following connection option uses one channel of the L239D, which can control motor speed through the PWM, However the motor then can only rotate in one direction.



The following connection uses two channels of the L239D: one channel outputs the PWM wave, and the other channel connects to GND, therefore you can control the speed of the motor. When these two channel signals are exchanged, not only controls the speed of motor, but also can control the steering of the motor.

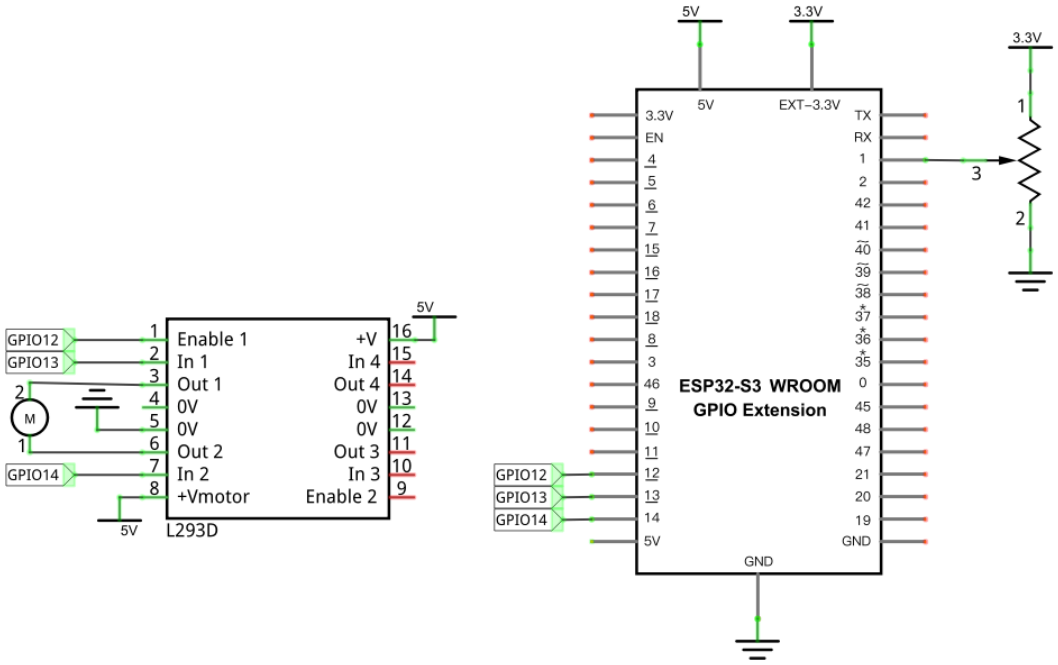


In practical use the motor is usually connected to channel 1 and 2 by outputting different levels to in1 and in2 to control the rotational direction of the motor, and output to the PWM wave to Enable1 port to control the motor's rotational speed. If the motor is connected to channel 3 and 4 by outputting different levels to in3 and in4 to control the motor's rotation direction, and output to the PWM wave to Enable2 pin to control the motor's rotational speed.

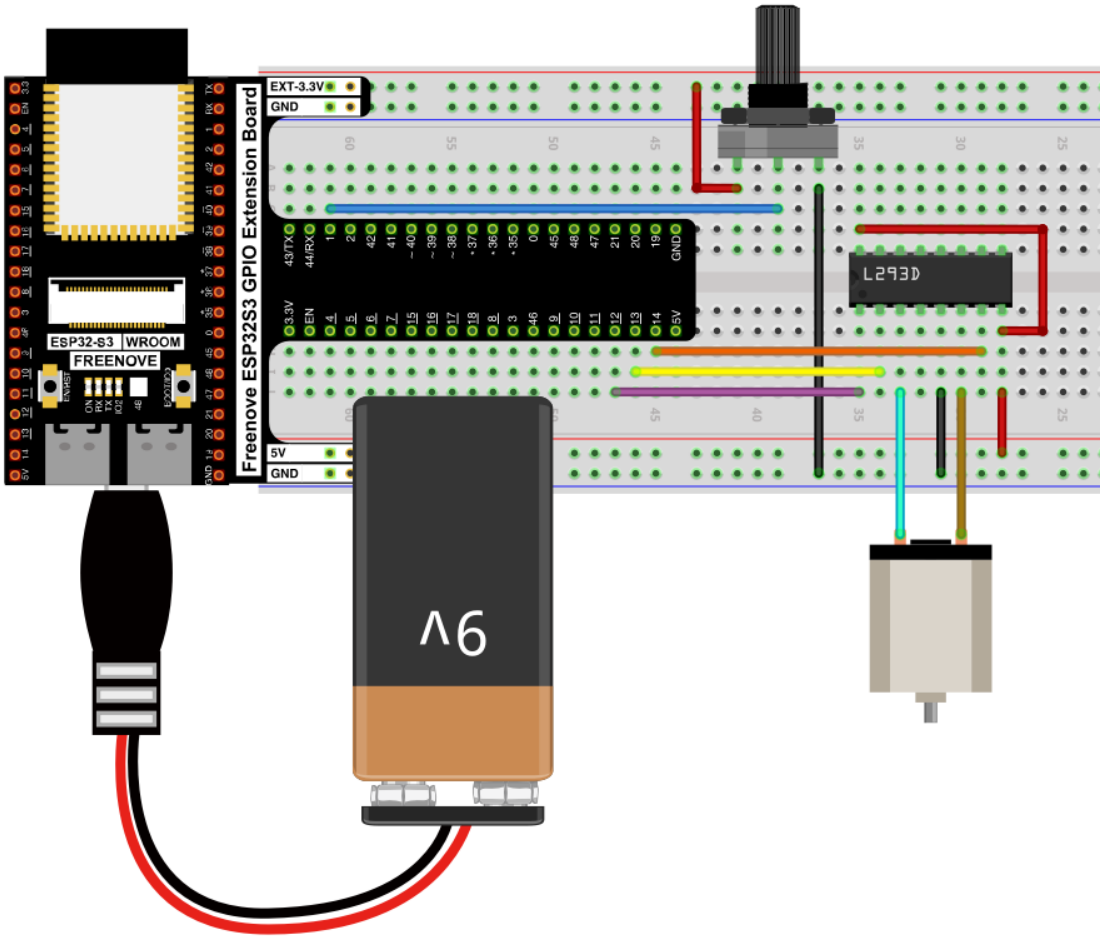
Circuit

Use caution when connecting this circuit, because the DC motor is a high-power component, do not use the power provided by the ESP32-S3 to power the motor directly, which may cause permanent damage to your ESP32-S3! The logic circuit can be powered by the ESP32-S3 power or an external power supply, which should share a common ground with ESP32-S3.

Schematic diagram



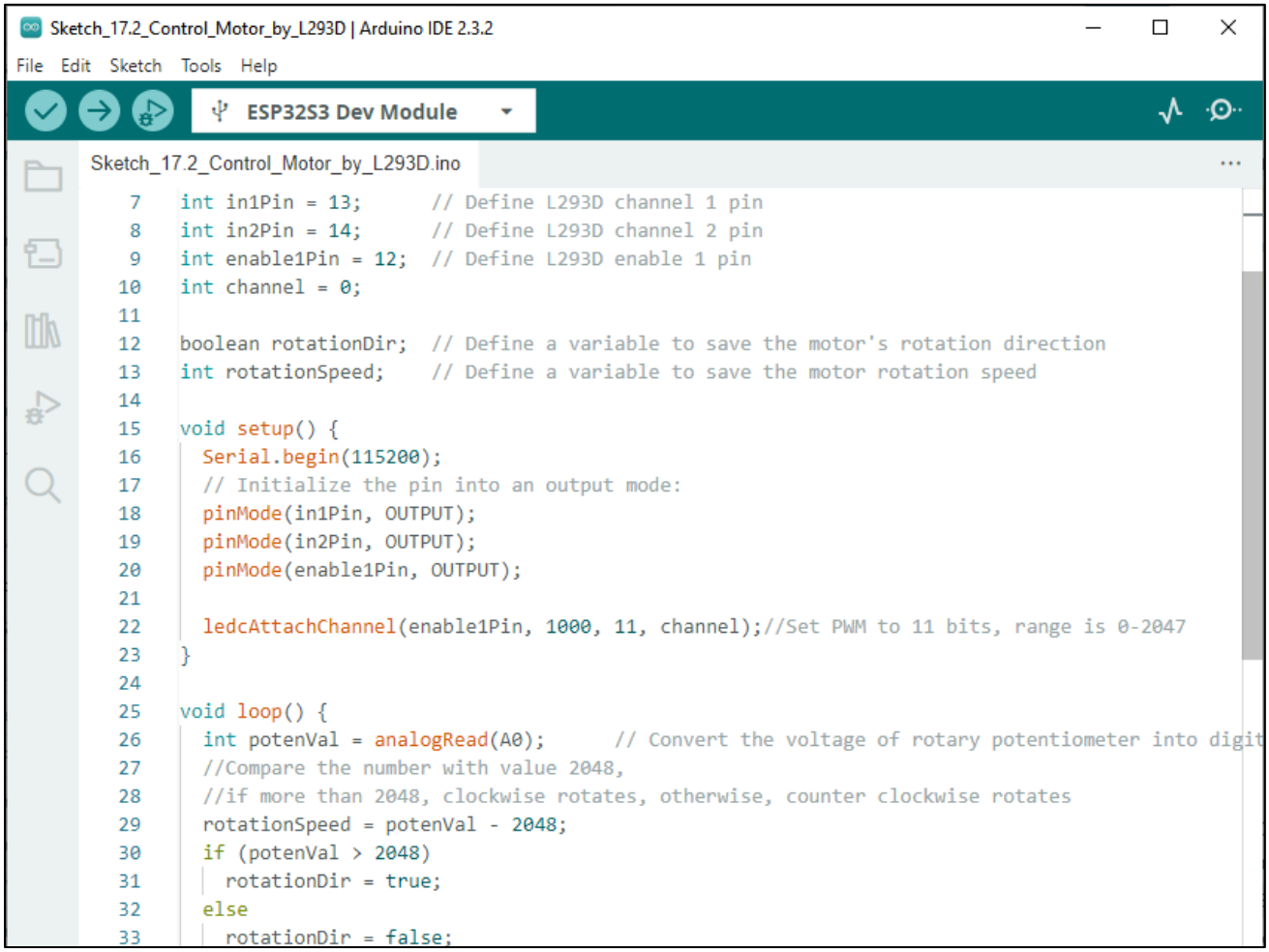
Hardware connection. If you need any support, please feel free to contact us via: support@freenove.com



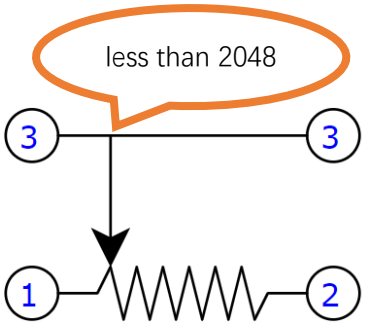
Note: the motor circuit uses A large current, about 0.2-0.3A without load.We recommend that you use a 9V battery to power the extension board.

Sketch

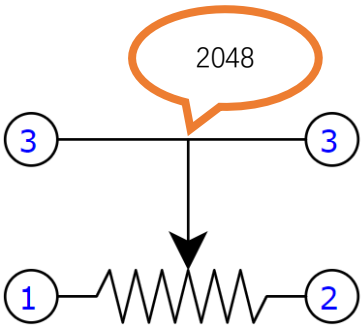
Sketch_16.1_Control_Motor_by_L293D



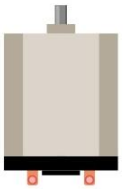
Download code to ESP32-S3 WROOM, rotate the potentiometer in one direction and the motor speeds up slowly in one direction. And then rotate the potentiometer in the other direction and the motor will slow down to stop. And then rotate it in an inverse direction to accelerate the motor.

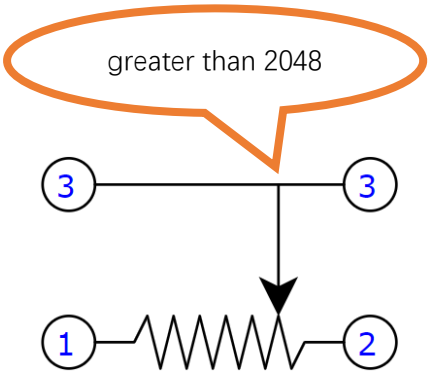

less than 2048
3
3
1
2


1 3 2


2048
3
3
1
2


1 3 2




greater than 2048
3
3
1
2


1 3 2

The following is the sketch:

```
int in1Pin = 13;      // Define L293D channel 1 pin
int in2Pin = 14;      // Define L293D channel 2 pin
int enable1Pin = 12;  // Define L293D enable 1 pin
int channel = 0;

boolean rotationDir;  // Define a variable to save the motor's rotation direction
int rotationSpeed;    // Define a variable to save the motor rotation speed

void setup() {
  // Initialize the pin into an output mode:
  pinMode(in1Pin, OUTPUT);
  pinMode(in2Pin, OUTPUT);
  pinMode(enable1Pin, OUTPUT);
  ledcAttachChannel(enable1Pin, 1000, 11, channel);//Set PWM to 11 bits, range is 0-2047
}

void loop() {
  int potenVal = analogRead(A0);  // Convert the voltage of rotary potentiometer into digital
  rotationSpeed = potenVal - 2048;
  if (potenVal > 2048)
    rotationDir = true;
  else
    rotationDir = false;
  // Calculate the motor speed
  rotationSpeed = abs(potenVal - 2048);
  //Control the steering and speed of the motor
  Serial.println(rotationSpeed);
  delay(100);
  driveMotor(rotationDir, constrain(rotationSpeed,0,2048));
}

void driveMotor(boolean dir, int spd) {
  if (dir) {// Control motor rotation direction
    digitalWrite(in1Pin, HIGH);
    digitalWrite(in2Pin, LOW);
  }
  else {
    digitalWrite(in1Pin, LOW);
    digitalWrite(in2Pin, HIGH);
  }
  ledcWrite(enable1Pin, spd);// Control motor rotation speed
}
```

The ADC of ESP32-S3 has a 12-bit accuracy, corresponding to a range from 0 to 4095. In this program, set the number 2048 as the midpoint. If the value of ADC is less than 2048, make the motor rotate in one direction. If the value of ADC is greater than 2048, make the motor rotate in the other direction. Subtract 2048 from the ADC value and take the absolute value and use this result as the speed of the motor.

```
20    int potenVal = analogRead(A0);// Convert the voltage of rotary potentiometer into digital
21    rotationSpeed = potenVal - 2048;
22    if (potenVal > 2048)
23      rotationDir = true;
24    else
25      rotationDir = false;
26    // Calculate the motor speed
27    rotationSpeed = abs(potenVal - 2048);
28    //Control the steering and speed of the motor
29    driveMotor(rotationDir, constrain(rotationSpeed,0,2048));
30  }
```

Set the accuracy of the PWM to 11 bits and range from 0 to 2047 to control the rotation speed of the motor.

```
15  ledcSetup(channel,1000,11);   //Set PWM to 11 bits, range is 0-2047
```

Function driveMotor is used to control the rotation direction and speed of the motor. The **dir** represents direction while **spd** refers to speed.

```
34  void driveMotor(boolean dir, int spd) {
35    // Control motor rotation direction
36    if (rotationDir) {
37      digitalWrite(in1Pin, HIGH);
38      digitalWrite(in2Pin, LOW);
39    }
40    else {
41      digitalWrite(in1Pin, LOW);
42      digitalWrite(in2Pin, HIGH);
43    }
44    // Control motor rotation speed
45    ledcWrite(enable1Pin, spd);
46  }
```

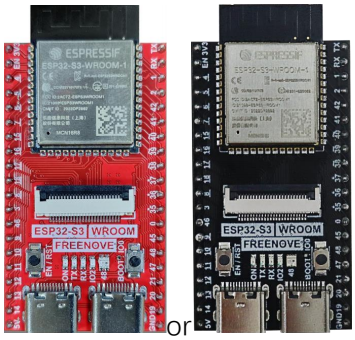
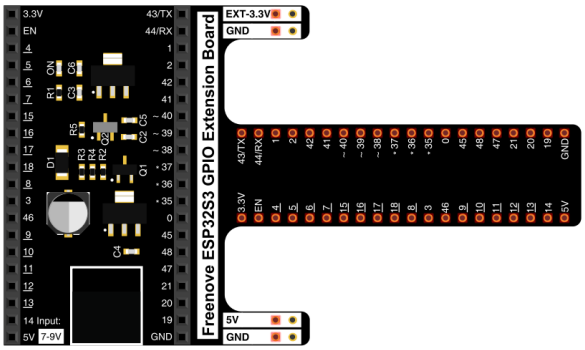
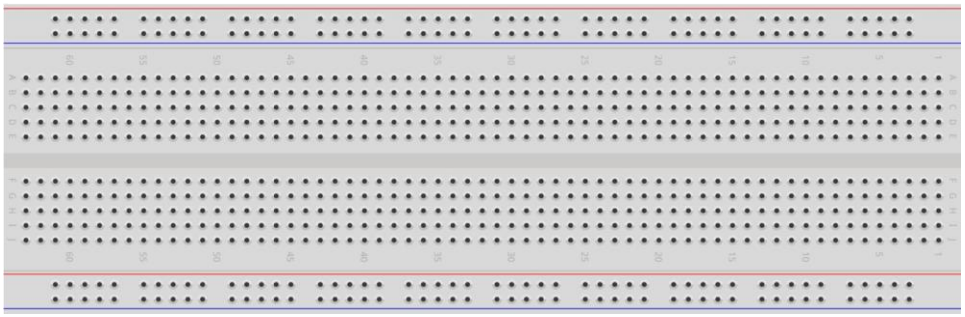
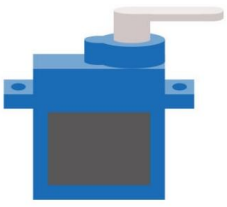
Chapter 17 Servo

Previously, we learned how to control the speed and rotational direction of a motor. In this chapter, we will learn about servos which are a rotary actuator type motor that can be controlled to rotate to specific angles.

Project 17.1 Servo Sweep

First, we need to learn how to make a servo rotate.

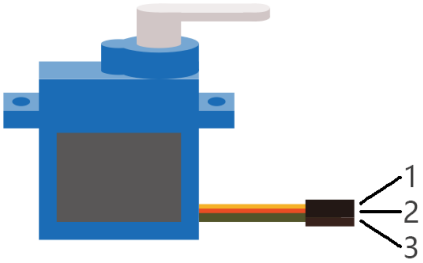
Component List

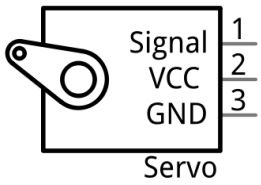
ESP32-S3(N16R8)/ESP32-S3(N8R8) 	GPIO Extension Board x1 
Breadboard x1 	
Servo x1 	Jumper M/M x3 

Component knowledge

Servo

Servo is a compact package which consists of a DC motor, a set of reduction gears to provide torque, a sensor and control circuit board. Most servos only have a 180-degree range of motion via their "horn". Servos can output higher torque than a simple DC motor alone and they are widely used to control motion in model cars, model airplanes, robots, etc. Servos have three wire leads which usually terminate to a male or female 3-pin plug. Two leads are for electric power: positive (2-VCC, Red wire), negative (3-GND, Brown wire), and the signal line (1-Signal, Orange wire), as represented in the Servo provided in your Kit.





We will use a 50Hz PWM signal with a duty cycle in a certain range to drive the Servo. The lasting time of 0.5ms-2.5ms of PWM single cycle high level corresponds to the servo angle 0 degrees - 180 degree linearly. Part of the corresponding values are as follows:

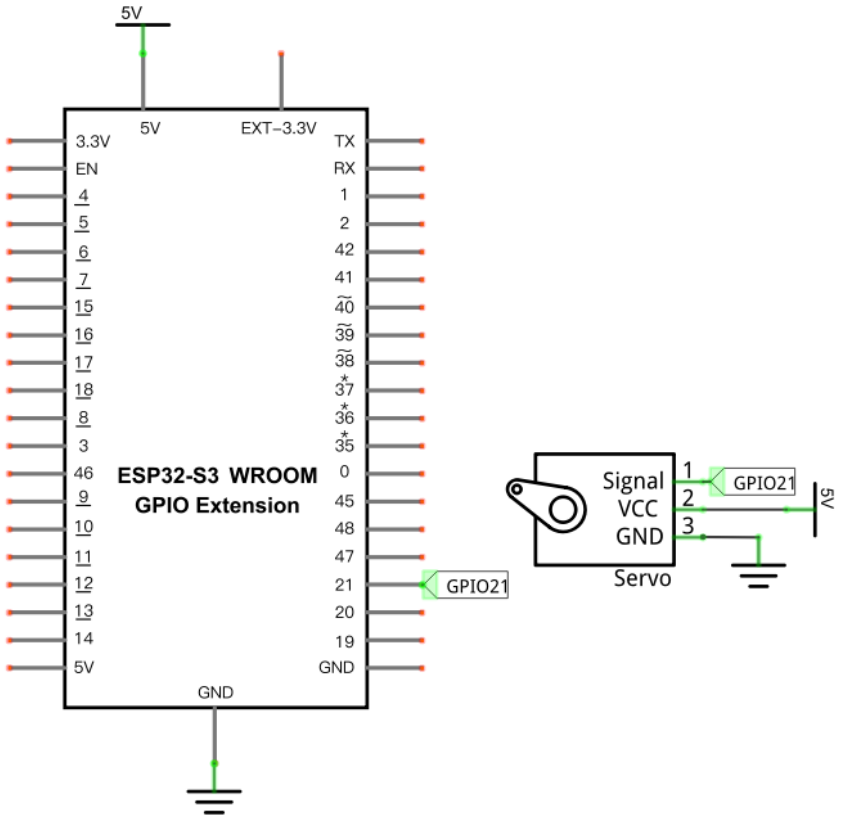
High level time	Servo angle
0.5ms	0 degree
1ms	45 degree
1.5ms	0 degree
2ms	45 degree
2.5ms	180 degree

When you change the servo signal value, the servo will rotate to the designated angle.

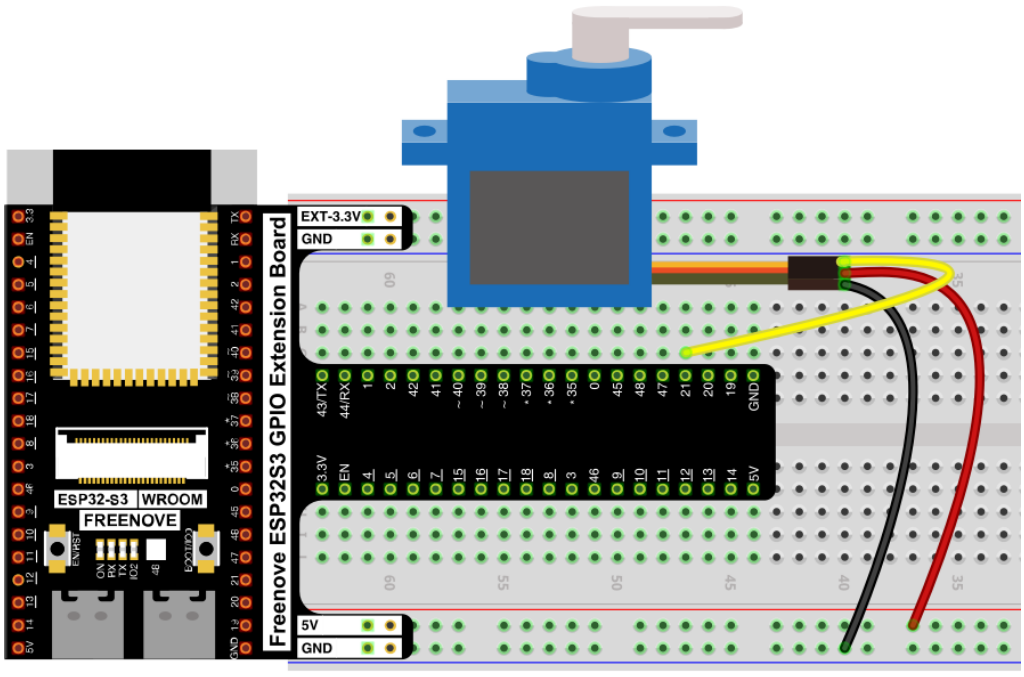
Circuit

Use caution when supplying power to the servo, it should be 5V. Make sure you do not make any errors when connecting the servo to the power supply.

Schematic diagram



Hardware connection. If you need any support, please feel free to contact us via: support@freenove.com



Sketch

Sketch_17.1_Servo_Sweep

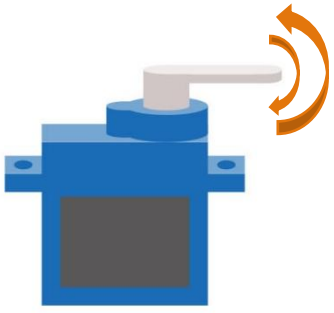
File Edit Sketch Tools Help

ESP32S3 Dev Module

Sketch_18.1_Servo_Sweep.ino

```
/**********************************************************************
  Filename    : Servo Sweep
  Description : Control the servo motor for sweeping
  Auther      : www.freenove.com
  Modification: 2022/10/25
**********************************************************************/
#define SERVO_PIN 21  //define the pwm pin
#define SERVO_CHN 0   //define the pwm channel
#define SERVO_FRQ 50  //define the pwm frequency
#define SERVO_BIT 12  //define the pwm precision

void servo_set_pin(int pin);
void servo_set_angle(int angle);

void setup() {
  servo_set_pin(SERVO_PIN);
}

void loop() {
  for (int i = 0; i < 180; i++) {  //make light fade in
    servo_set_angle(i);
    delay(10);
  }
  for (int i = 180; i > 0; i--) {  //make light fade out
    servo_set_angle(i);
    delay(10);
  }
```

Compile and upload the code to ESP32-S3 WROOM, the servo will rotate from 0 degrees to 180 degrees and then reverse the direction to make it rotate from 180 degrees to 0 degrees and repeat these actions in an endless loop.



The following is the program code:

```
#define SERVO_PIN 21   //define the pwm pin
#define SERVO_CHN 0    //define the pwm channel
#define SERVO_FRQ 50   //define the pwm frequency
#define SERVO_BIT 12   //define the pwm precision
void servo_set_pin(int pin);
void servo_set_angle(int angle);

void setup() {
  servo_set_pin(SERVO_PIN);
}

void loop() {
  for (int i = 0; i < 180; i++) {  //make light fade in
    servo_set_angle(i);
    delay(10);
  }
  for (int i = 180; i > 0; i--) {  //make light fade out
    servo_set_angle(i);
    delay(10);
  }
}

void servo_set_pin(int pin) {
  ledcAttachChannel(pin, SERVO_FRQ, SERVO_BIT, SERVO_CHN);
}

void servo_set_angle(int angle) {
  if (angle > 180 || angle < 0)
    return;
  long pwm_value = map(angle, 0, 180, 103, 512);
  ledcWrite(SERVO_PIN, pwm_value);
}
```

Define the pins controlling Servo and the frequency and duty cycle of the signal.

```
#define SERVO_PIN 21   //define the pwm pin
#define SERVO_CHN 0    //define the pwm channel
#define SERVO_FRQ 50   //define the pwm frequency
#define SERVO_BIT 12   //define the pwm precision
```

Initialize Servo pin. Here, PWM control mode is used to control Servo motor.

```
void servo_set_pin(int pin) {
  ledcAttachChannel(pin, SERVO_FRQ, SERVO_BIT, SERVO_CHN);
}
```

Write a function to control the rotation angle of Servo. The angle range is 0-180 degrees.

```
27  void servo_set_angle(int angle) {
28    if (angle > 180 || angle < 0)
29      return;
30    long pwm_value = map(angle, 0, 180, 103, 512);
31    ledcWrite(SERVO_PIN, pwm_value);
32  }
```

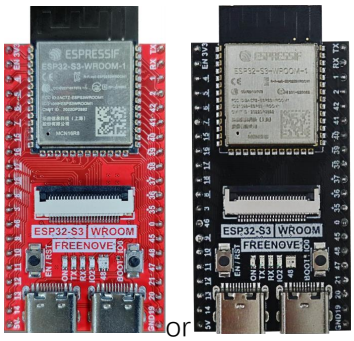
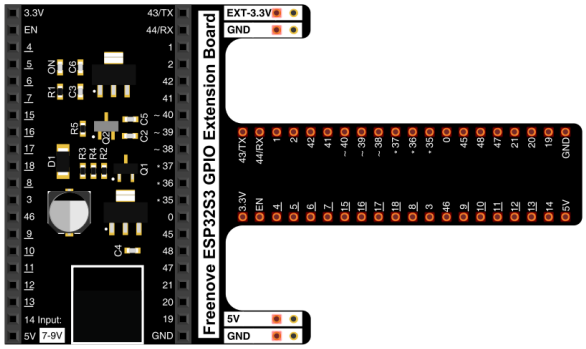
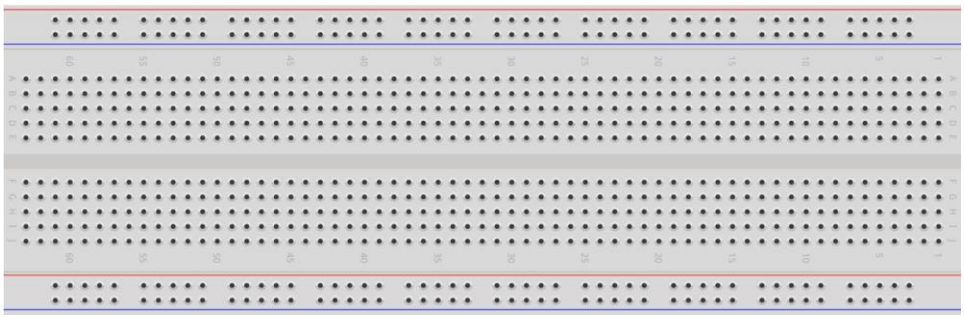
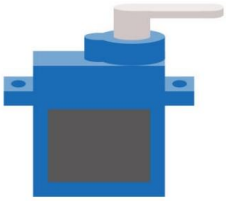
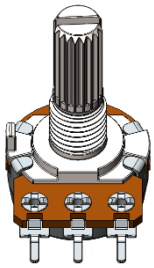
Control the steering gear to rotate from 0 ° to 180 °, and then rotate from 180 ° to 0 °, and keep rotating circularly.

```
13    for (int i = 0; i < 180; i++) {  //make light fade in
14      servo_set_angle(i);
15      delay(10);
16    }
17    for (int i = 180; i > 0; i--) {  //make light fade out
18      servo_set_angle(i);
19      delay(10);
20    }
```

Project 17.2 Servo Knop

Use a potentiometer to control the servo motor to rotate at any angle.

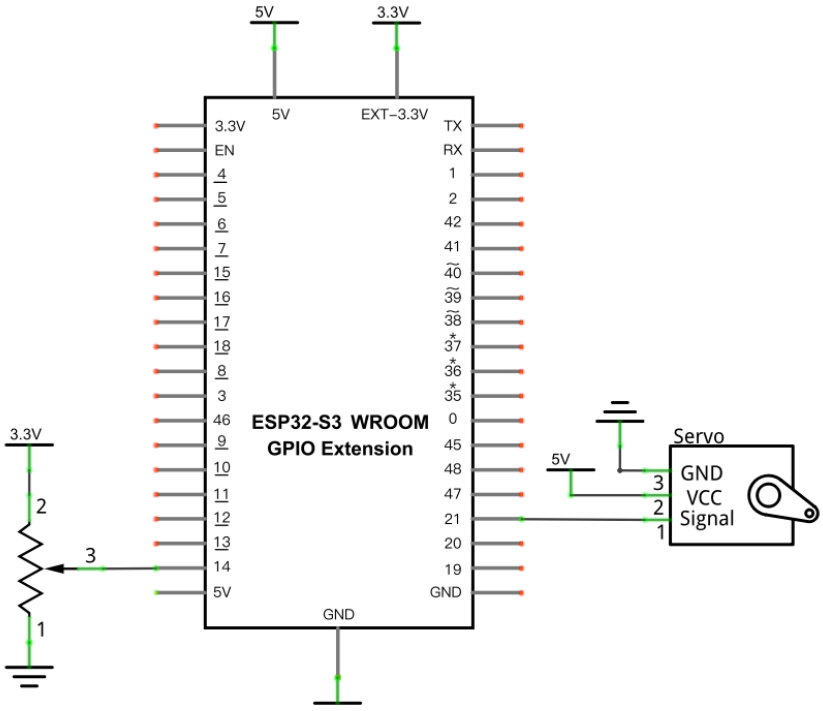
Component List

<table>
<tr><td>ESP32-S3(N16R8)/ESP32-S3(N8R8)
  </td><td colspan="2">GPIO Extension Board x1
  </td></tr>
<tr><td colspan="3">Breadboard x1
  </td></tr>
<tr><td>Servo x1
  </td><td>Jumper M/M x6
  </td><td>Rotary potentiometer x1
  </td></tr>
</table>

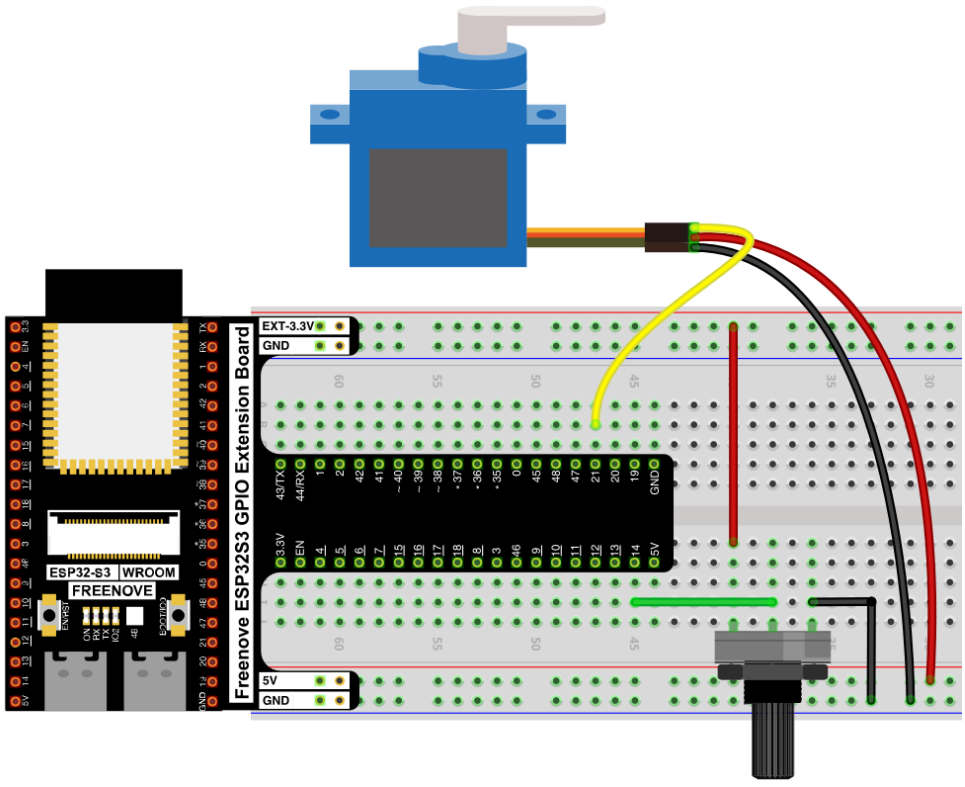
Circuit

Use caution when supplying power to the servo, it should be 5V. Make sure you do not make any errors when connecting the servo to the power supply.

Schematic diagram



Hardware connection. If you need any support, please feel free to contact us via: support@freenove.com



Sketch

Sketch_17.2_Servo_Sweep

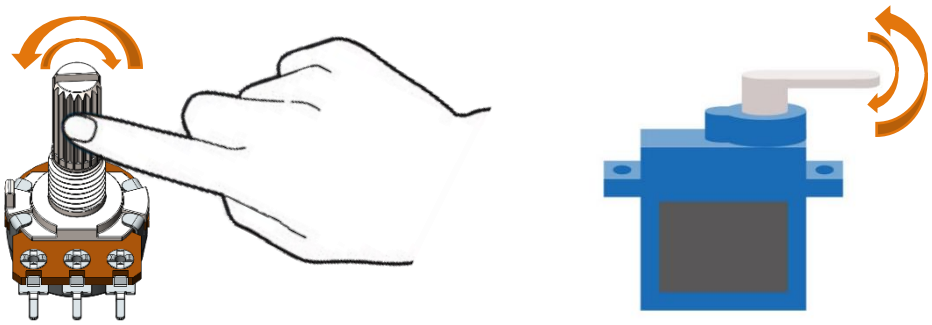
File Edit Sketch Tools Help

ESP32S3 Dev Module

Sketch_18.2_Control_Servo_by_Potentiometer.ino

```
/**********************************************************************
  Filename    : Control Servo by Potentiometer
  Description : Use potentiometer to control the rotation of servo motor.
  Auther      : www.freenove.com
  Auther      : www.freenove.com
  Modification: 2022/10/25
**********************************************************************/
#define SERVO_PIN 21  //define the pwm pin
#define SERVO_CHN 0   //define the pwm channel
#define SERVO_FRQ 50  //define the pwm frequency
#define SERVO_BIT 12  //define the pwm precision
#define ADC_PIN   14  //define the adc pin

void servo_set_pin(int pin);
void servo_set_angle(int angle);

void setup() {
  servo_set_pin(SERVO_PIN);
  Serial.begin(115200);
}

void loop() {
  // read the value of the potentiometer (value between 0 and 4095)
  int potVal = analogRead(ADC_PIN);
  Serial.printf("potVal_1: %d\t",potVal);
```

Compile and upload the code to ESP32-S3 WROOM, twist the potentiometer back and forth, and the servo motor rotates accordingly.



The following is the program code:

```
#define SERVO_PIN 21  //define the pwm pin
#define SERVO_CHN 0   //define the pwm channel
#define SERVO_FRQ 50  //define the pwm frequency
#define SERVO_BIT 12  //define the pwm precision
#define ADC_PIN   14  //define the adc pin

void servo_set_pin(int pin);
void servo_set_angle(int angle);

void setup() {
  servo_set_pin(SERVO_PIN);
  Serial.begin(115200);
}

void loop() {
  // read the value of the potentiometer (value between 0 and 4095)
  int potVal = analogRead(ADC_PIN);
  Serial.printf("potVal_1: %d\t",potVal);
  // scale it to use it with the servo (value between 0 and 180)
  potVal = map(potVal, 0, 4095, 0, 180);
  // set the servo position according to the scaled value
  servo_set_angle(potVal);
  Serial.printf("potVal_2: %d\r\n",potVal);
  delay(15);// wait for the servo to get there
}

void servo_set_pin(int pin) {
  ledcAttachChannel(pin, SERVO_FRQ, SERVO_BIT, SERVO_CHN);
}

void servo_set_angle(int angle) {
  if (angle > 180 || angle < 0)
    return;
  long pwm_value = map(angle, 0, 180, 103, 512);
  ledcWrite(SERVO_PIN, pwm_value);
}
```

In this experiment, we obtain the ADC value of the potentiometer and store it in potVal. Use map function to convert it into corresponding angle value and we can control the motor to rotate to a specified angle, and print the value via serial.

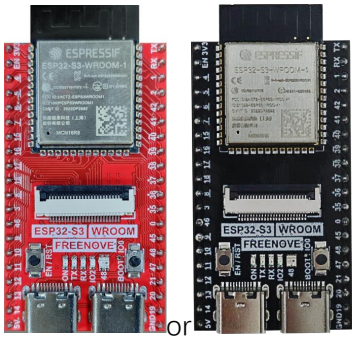
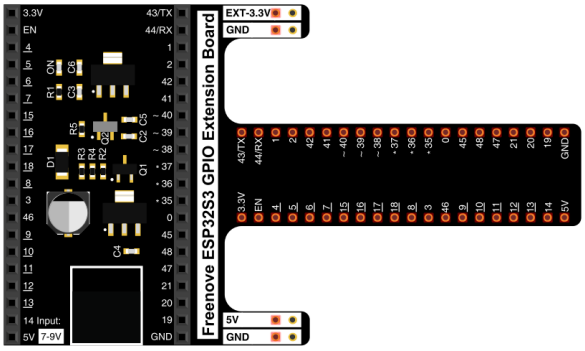
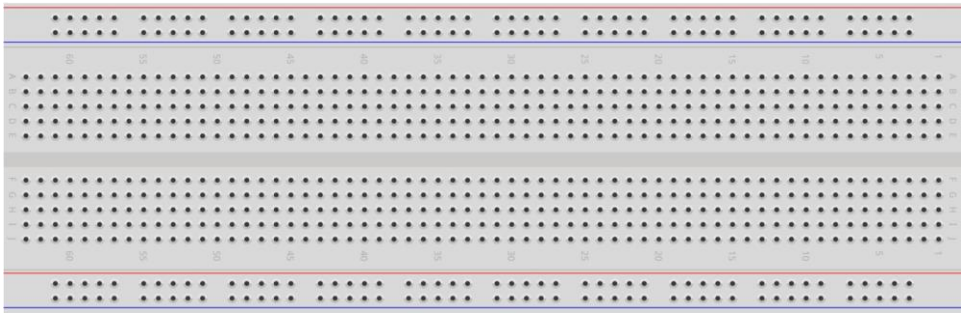
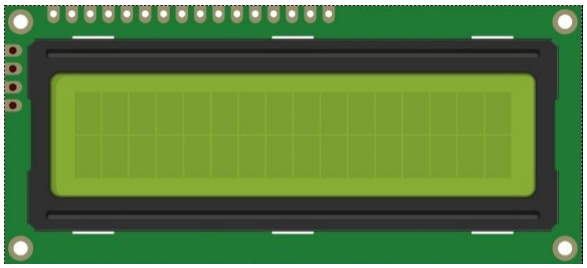
Chapter 18 LCD1602

In this chapter, we will learn about the LCD1602 Display Screen

Project 18.1 LCD1602

In this section we learn how to use LCD1602 to display something.

Component List

ESP32-S3(N16R8)/ESP32-S3(N8R8) 	GPIO Extension Board x1 
Breadboard x1 	
LCD1602 Module x1 	Jumper F/M x4 

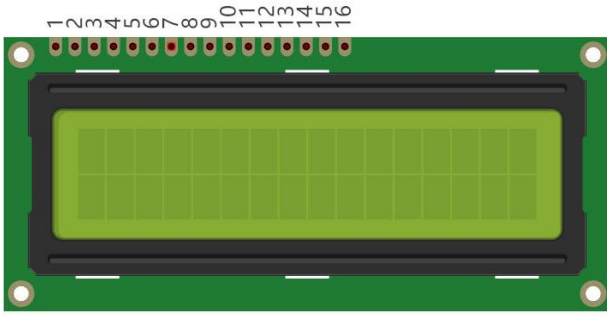
Component knowledge

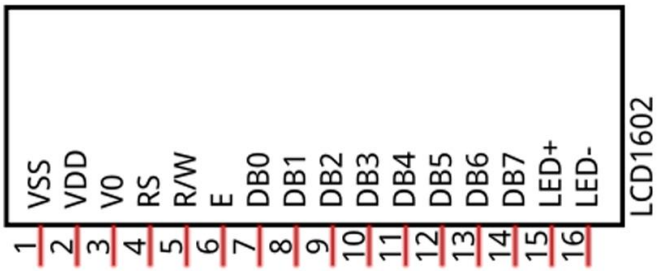
I2C communication

I2C (Inter-Integrated Circuit) is a two-wire serial communication mode, which can be used for the connection of micro controllers and their peripheral equipment. Devices using I2C communication must be connected to the serial data (SDA) line, and serial clock (SCL) line (called I2C bus). Each device has a unique address and can be used as a transmitter or receiver to communicate with devices connected to the bus.

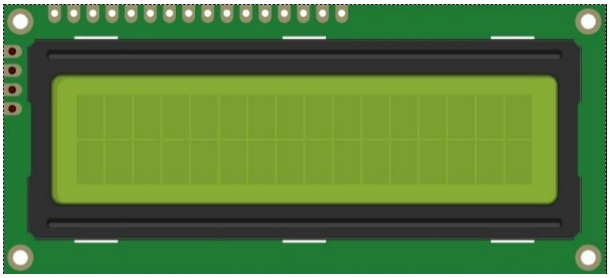
LCD1602 communication'

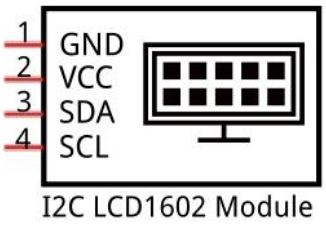
The LCD1602 display screen can display 2 lines of characters in 16 columns. It is capable of displaying numbers, letters, symbols, ASCII code and so on. As shown below is a monochrome LCD1602 display screen along with its circuit pin diagram





I2C LCD1602 display screen integrates a I2C interface, which connects the serial-input & parallel-output module to the LCD1602 display screen. This allows us to only use 4 lines to the operate the LCD1602.

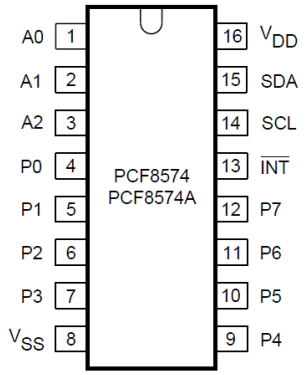




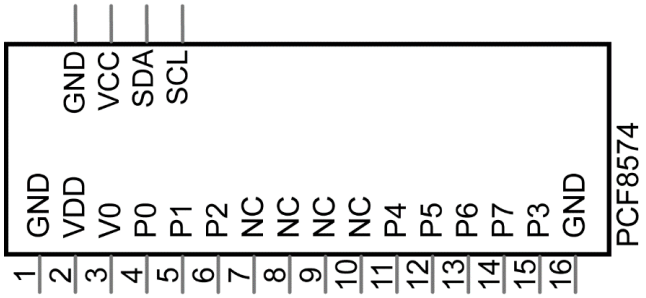
The serial-to-parallel IC chip used in this module is PCF8574T (PCF8574AT), and its default I2C address is 0x27(0x3F).

Below is the PCF8574 pin schematic diagram and the block pin diagram:

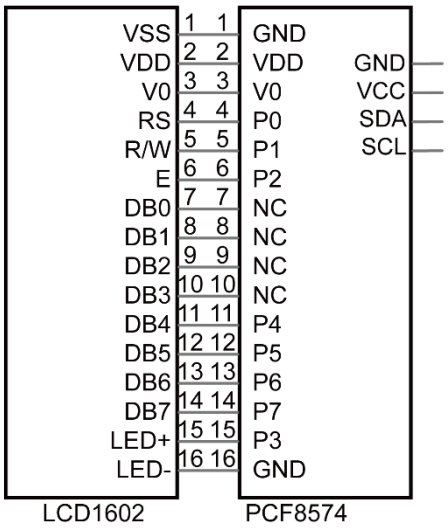
PCF8574 chip pin diagram:



PCF8574 module pin diagram



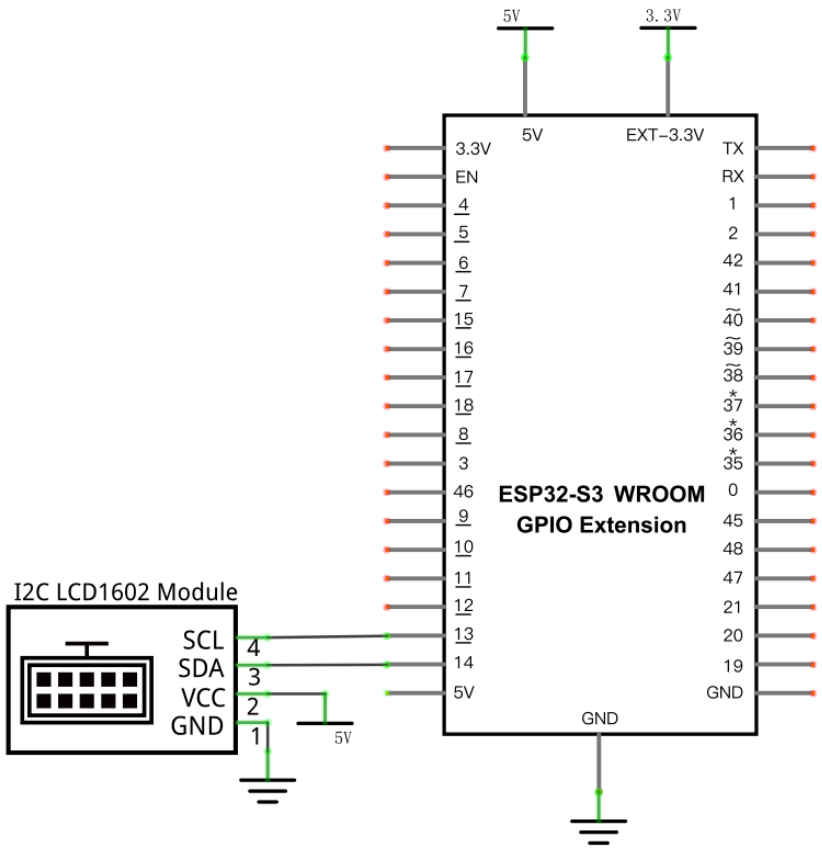
PCF8574 module pin and LCD1602 pin are corresponding to each other and connected with each other:



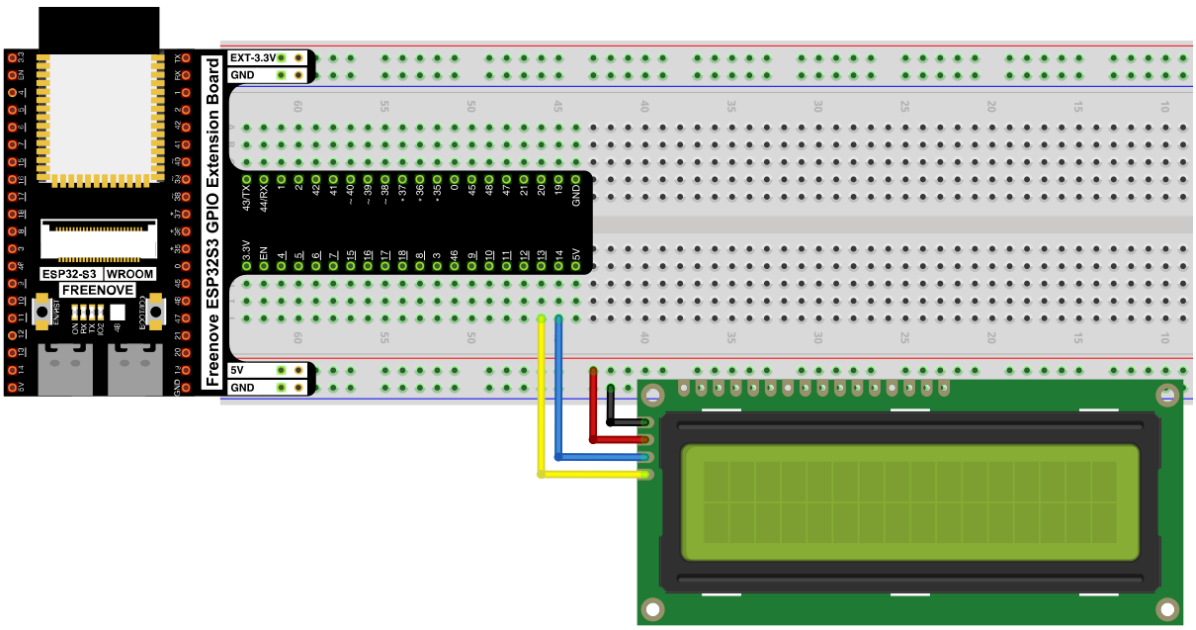
So we only need 4 pins to control the 16 pins of the LCD1602 display screen through the I2C interface.
In this project, we will use the I2C LCD1602 to display some static characters and dynamic variables.

Circuit

Schematic diagram



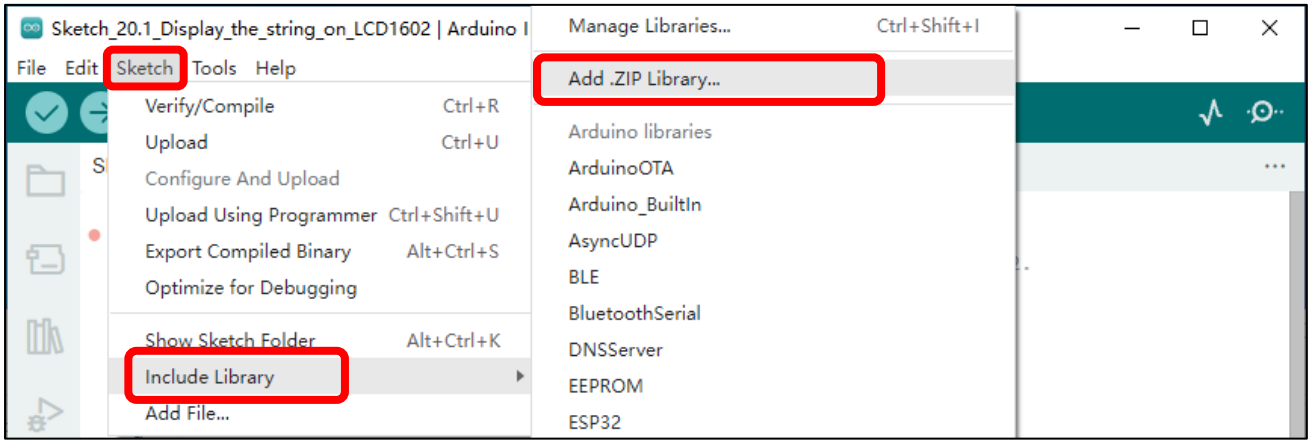
Hardware connection. If you need any support, please feel free to contact us via: support@freenove.com

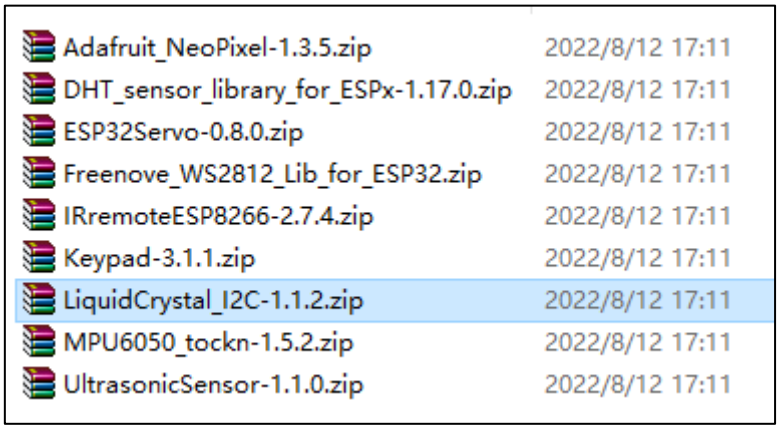


Sketch

How to install the library

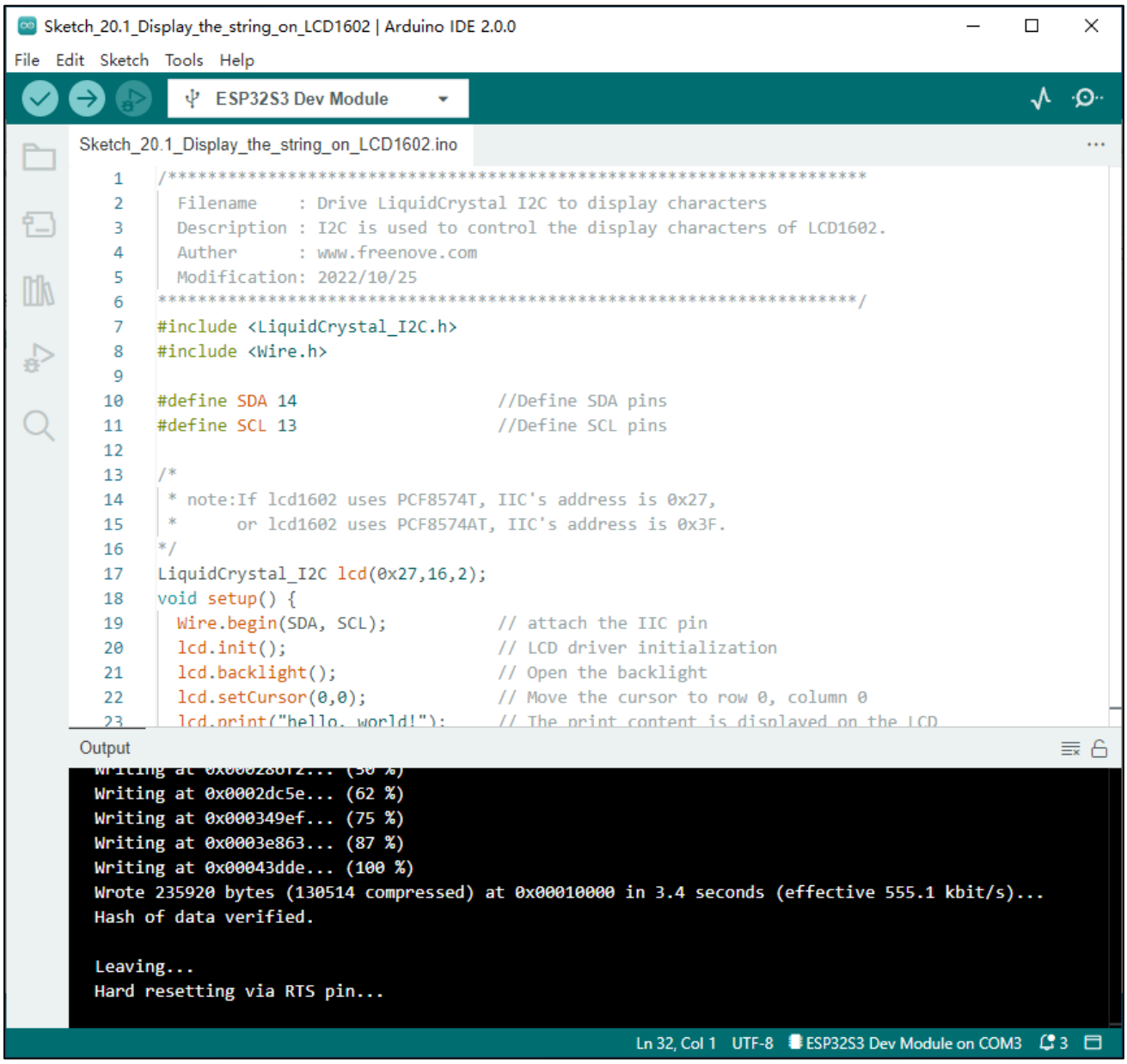
We use the third party library LiquidCrystal I2C. If you haven't installed it yet, please do so before learning.
The steps to add third-party Libraries are as follows:
open arduino->Sketch->Include library-> Add .zip Library....
Select "**Freenove_Super_Starter_Kit_for_ESP32_S3\C\Libraries\LiquidCrystal_I2C.zip**" for installation.





Use I2C LCD 1602 to display characters and variables.

Sketch_18.1_Display_the_string_on_LCD1602



Compile and upload the code to ESP32-S3 WROOM and the LCD1602 displays characters.

The following is the program code:

```
#include <LiquidCrystal_I2C.h>
#include <Wire.h>

#define SDA 14                    //Define SDA pins
#define SCL 13                    //Define SCL pins

/*
 * note:If lcd1602 uses PCF8574T, IIC's address is 0x27,
 *      or lcd1602 uses PCF8574AT, IIC's address is 0x3F.
*/
LiquidCrystal_I2C lcd(0x27,16,2);
void setup() {
  Wire.begin(SDA, SCL);           // attach the IIC pin
  if (!i2CAddrTest(0x27)) {
    lcd = LiquidCrystal_I2C(0x3F, 16, 2);
  }
  lcd.init();                     // LCD driver initialization
  lcd.backlight();                // Open the backlight
  lcd.setCursor(0,0);             // Move the cursor to row 0, column 0
  lcd.print("hello, world!");     // The print content is displayed on the LCD
}

void loop() {
  lcd.setCursor(0,1);             // Move the cursor to row 1, column 0
  lcd.print("Counter:");          // The count is displayed every second
  lcd.print(millis() / 1000);
  delay(1000);
}

bool i2CAddrTest(uint8_t addr) {
  Wire.beginTransmission(addr);
  if (Wire.endTransmission() == 0) {
    return true;
  }
  return false;
}
```

Include header file of Liquid Crystal Display (LCD)1602 and I2C.

```
#include <LiquidCrystal_I2C.h>
#include <Wire.h>
```

Instantiate the I2C LCD1602 screen. It should be noted here that if your LCD driver chip uses PCF8574T, set the I2C address to 0x27, and if uses PCF8574AT, set the I2C address to 0x3F.

```
  Wire.begin(SDA, SCL);           // attach the IIC pin
  if (!i2CAddrTest(0x27)) {
```

```
15      lcd = LiquidCrystal_I2C(0x3F, 16, 2);
16    }
```

Initialize LCD1602 and turn on the backlight of LCD.

```
17    lcd.init();                       // LCD driver initialization
18    lcd.backlight();                  // Turn on the backlight
```

Move the cursor to the first row, first column, and then display the character.

```
19    lcd.setCursor(0,0);               // Move the cursor to row 0, column 0
20    lcd.print("hello, world! ");       // The print content is displayed on the LCD
```

Print the number on the second line of LCD1602.

```
23  void loop(){
24    lcd.setCursor(0,1);               // Move the cursor to row 1, column 0
25    lcd.print("Counter:");            // The count is displayed every second
26    lcd.print(millis() / 1000);
27    delay(1000);
28  }
```

Check whether the I2C address is responded by a device.

```
30  bool i2CAddrTest(uint8_t addr) {
31    Wire.beginTransmission(addr);
32    if (Wire.endTransmission() == 0) {
33      return true;
34    }
35    return false;
36  }
```

Reference

class LiquidCrystal

The LiquidCrystal class can manipulate common LCD screens. The first step is defining an object of LiquidCrystal, for example:

`LiquidCrystal_I2C lcd(0x27,16,2);`

Instantiate the Lcd1602 and set the I2C address to 0x27, with 16 columns per row and 2 rows per column.

`init();`

Initializes the Lcd1602's device

`backlight();`

Turn on Lcd1602's backlight.

`setCursor(column,row);`

Sets the screen's column and row.

column: The range is 0 to 15.

row: The range is 0 to 1.

`print(String);`

Print the character string on Lcd1602

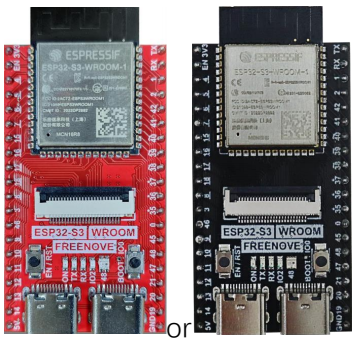
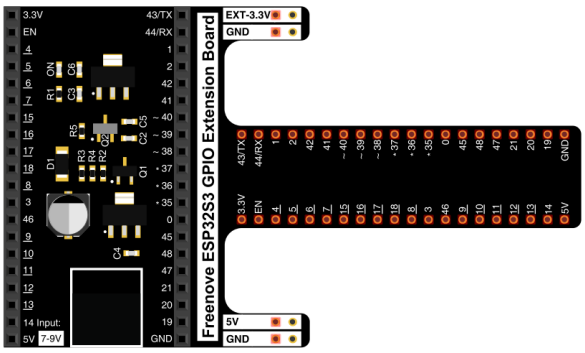
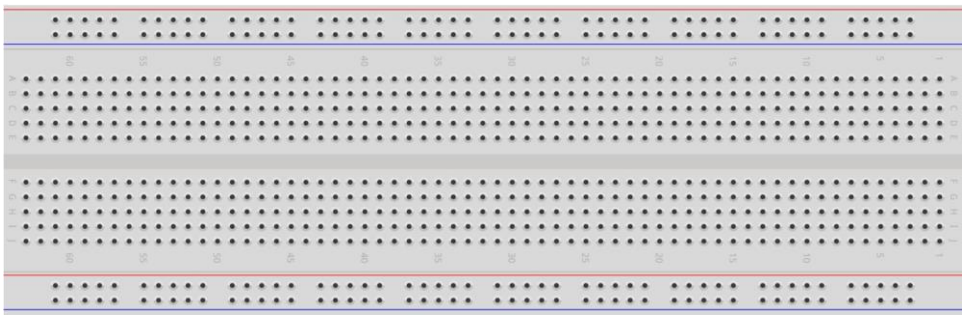
Chapter 19 Ultrasonic Ranging

In this chapter, we learn a module which use ultrasonic to measure distance, HC SR04.

Project 19.1 Ultrasonic Ranging

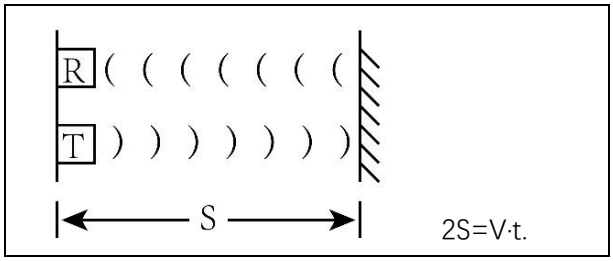
In this project, we use ultrasonic ranging module to measure distance, and print out the data in the terminal.

Component List

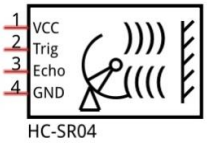
<table>
<tr><td>ESP32-S3(N16R8)/ESP32-S3(N8R8)
 
or</td><td>GPIO Extension Board x1
  </td></tr>
<tr><td colspan="2">Breadboard x1
  </td></tr>
<tr><td>Jumper F/M x4
  </td><td>HC SR04 x1
  </td></tr>
</table>

Component Knowledge

The ultrasonic ranging module uses the principle that ultrasonic waves will be sent back when encounter obstacles. We can measure the distance by counting the time interval between sending and receiving of the ultrasonic waves, and the time difference is the total time of the ultrasonic wave's journey from being transmitted to being received. Because the speed of sound in air is a constant, about v=340m/s, we can calculate the distance between the ultrasonic ranging module and the obstacle: s=vt/2.



The HC-SR04 ultrasonic ranging module integrates both an ultrasonic transmitter and a receiver. The transmitter is used to convert electrical signals (electrical energy) into high frequency (beyond human hearing) sound waves (mechanical energy) and the function of the receiver is opposite of this. The picture and the diagram of the HC SR04 ultrasonic ranging module are shown below:



Pin description:

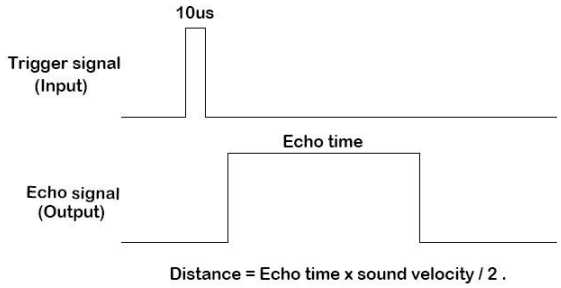
Pin	Description
VCC	power supply pin
Trig	trigger pin
Echo	Echo pin
GND	GND

Technical specs:

Working voltage: 5V Working current: 12mA

Minimum measured distance: 2cm Maximum measured distance: 200cm

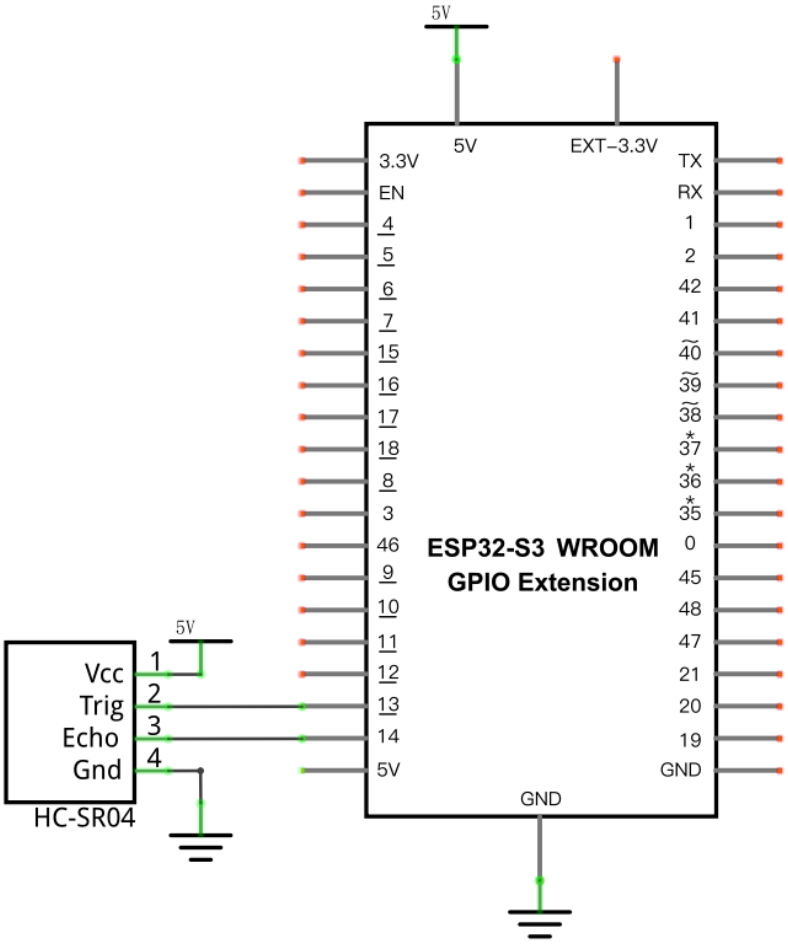
Instructions for use: output a high-level pulse in Trig pin lasting for least 10us, the module begins to transmit ultrasonic waves. At the same time, the Echo pin is pulled up. When the module receives the returned ultrasonic waves from encountering an obstacle, the Echo pin will be pulled down. The duration of high level in the Echo pin is the total time of the ultrasonic wave from transmitting to receiving, s=vt/2.



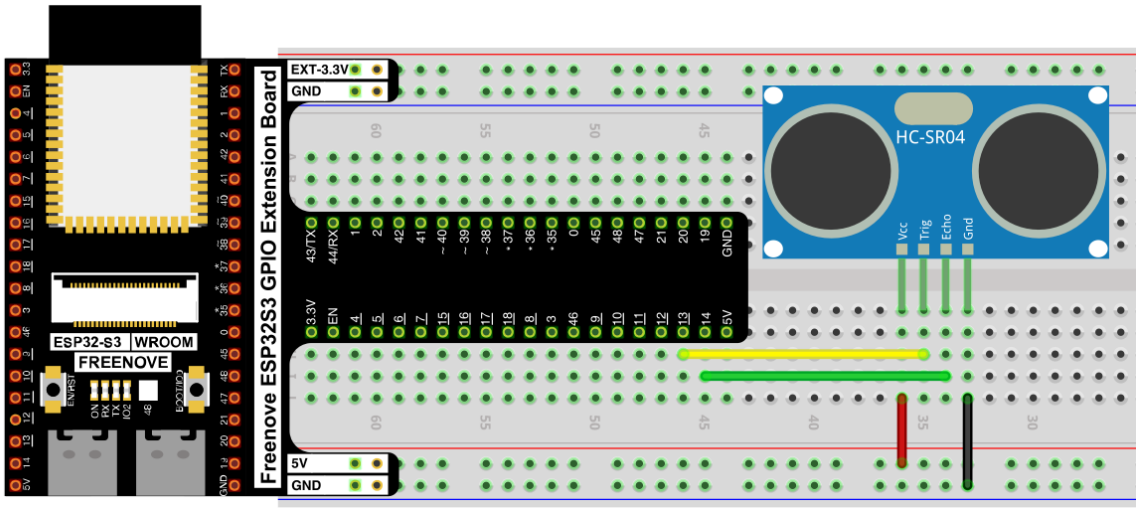
Circuit

Note that the voltage of ultrasonic module is 5V in the circuit.

Schematic diagram

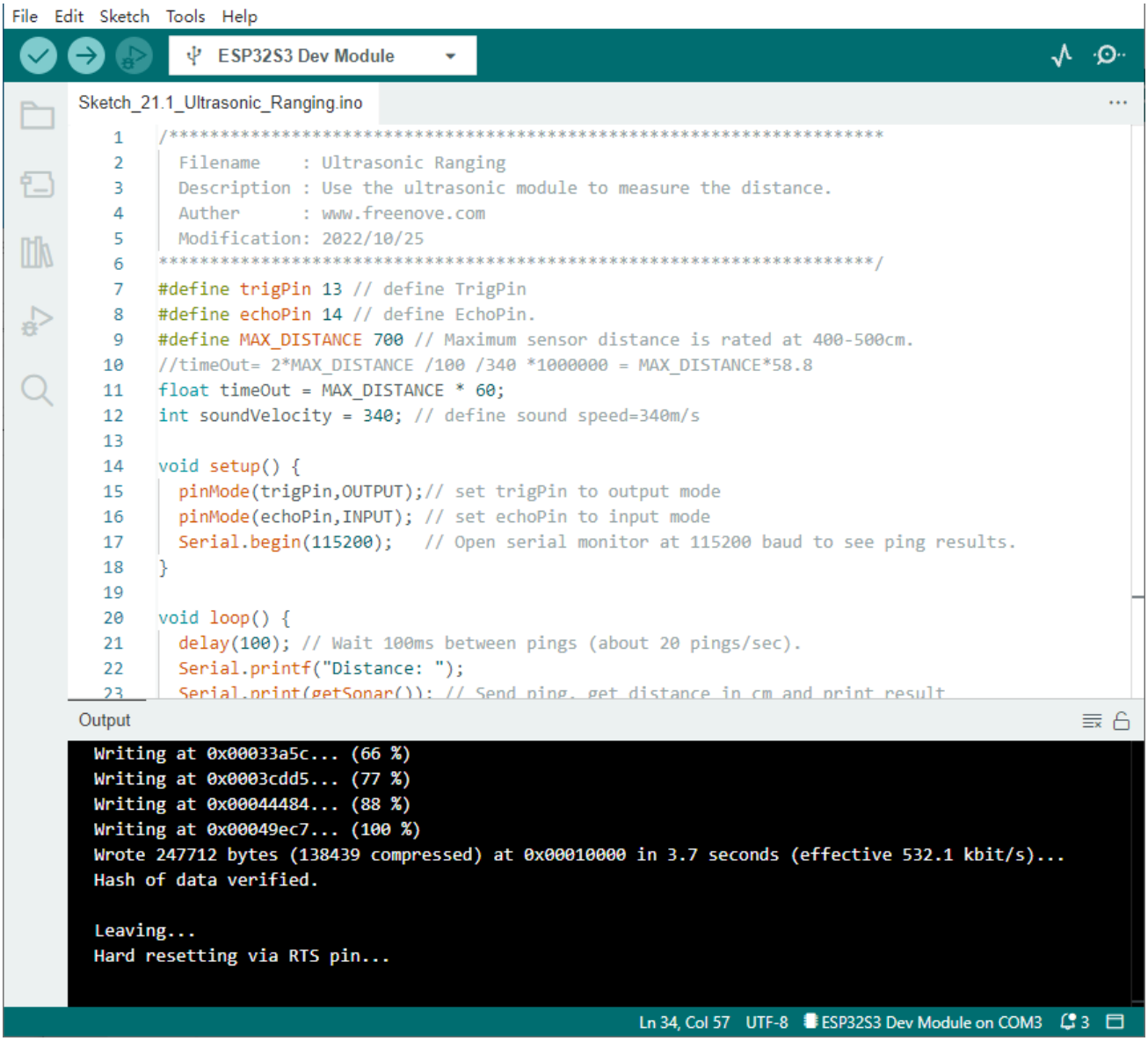


Hardware connection. If you need any support, please feel free to contact us via: support@freenove.com



Sketch

Sketch_19.1_Ultrasonic_Ranging



Download the code to ESP32-S3 WROOM, open the serial port monitor, set the baud rate to 115200 and you can use it to measure the distance between the ultrasonic module and the object. As shown in the following figure:

The following is the program code:

```
#define trigPin 13 // define trigPin
#define echoPin 14 // define echoPin.
#define MAX_DISTANCE 700 // Maximum sensor distance is rated at 400-500cm.
//timeOut= 2*MAX_DISTANCE /100 /340 *1000000 = MAX_DISTANCE*58.8
float timeOut = MAX_DISTANCE * 60;
int soundVelocity = 340; // define sound speed=340m/s

void setup() {
  pinMode(trigPin,OUTPUT);// set trigPin to output mode
  pinMode(echoPin,INPUT); // set echoPin to input mode
  Serial.begin(115200);   // Open serial monitor at 115200 baud to see ping results.
}

void loop() {
  delay(100); // Wait 100ms between pings (about 20 pings/sec).
  Serial.printf("Distance: ");
  Serial.print(getSonar()); // Send ping, get distance in cm and print result
  Serial.println("cm");
}

float getSonar() {
  unsigned long pingTime;
  float distance;
  // make trigPin output high level lasting for 10us to trigger HC_SR04
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);
  // Wait HC-SR04 returning to the high level and measure out this waiting time
  pingTime = pulseIn(echoPin, HIGH, timeOut);
  // calculate the distance according to the time
```

```
31    distance = (float)pingTime * soundVelocity / 2 / 10000;
32    return distance; // return the distance value
33  }
```

First, define the pins and the maximum measurement distance.

```
1   #define trigPin 13 // define trigPin
2   #define echoPin 14 // define echoPin.
3   #define MAX_DISTANCE 700        //define the maximum measured distance
```

If the module does not return high level, we cannot wait for this forever, so we need to calculate the time period for the maximum distance, that is, time Out. timeOut= 2*MAX_DISTANCE/100/340*1000000. The result of the constant part in this formula is approximately 58.8.

```
5   float timeOut = MAX_DISTANCE * 60;
```

Subfunction getSonar () function is used to start the ultrasonic module to begin measuring, and return the measured distance in cm units. In this function, first let trigPin send 10us high level to start the ultrasonic module. Then use pulseIn () to read the ultrasonic module and return the duration time of high level. Finally, the measured distance according to the time is calculated.

```
21  float getSonar() {
22    unsigned long pingTime;
23    float distance;
24    // make trigPin output high level lasting for 10μs to triger HC_SR04?
25    digitalWrite(trigPin, HIGH);
26    delayMicroseconds(10);
27    digitalWrite(trigPin, LOW);
28    // Wait HC-SR04 returning to the high level and measure out this waitting time
29    pingTime = pulseIn(echoPin, HIGH, timeOut);
30    // calculate the distance according to the time
31    distance = (float)pingTime * soundVelocity / 2 / 10000;
32    return distance; // return the distance value
33  }
```

Lastly, in loop() function, get the measurement distance and display it continually.

```
14  void loop() {
15    delay(100); // Wait 100ms between pings (about 20 pings/sec).
16    Serial.printf("Distance: ");
17    Serial.print(getSonar()); // Send ping, get distance in cm and print result
18    Serial.println("cm");
19  }
```

About function **pulseIn**():

int pulseIn(int pin, int level, int timeout);

pin: the number of the Arduino pin on which you want to read the pulse. Allowed data types: int.
value: type of pulse to read: either HIGH or LOW. Allowed data types: int.
timeout (optional): the number of microseconds to wait for the pulse to start; default is one second.

Project 19.2 Ultrasonic Ranging

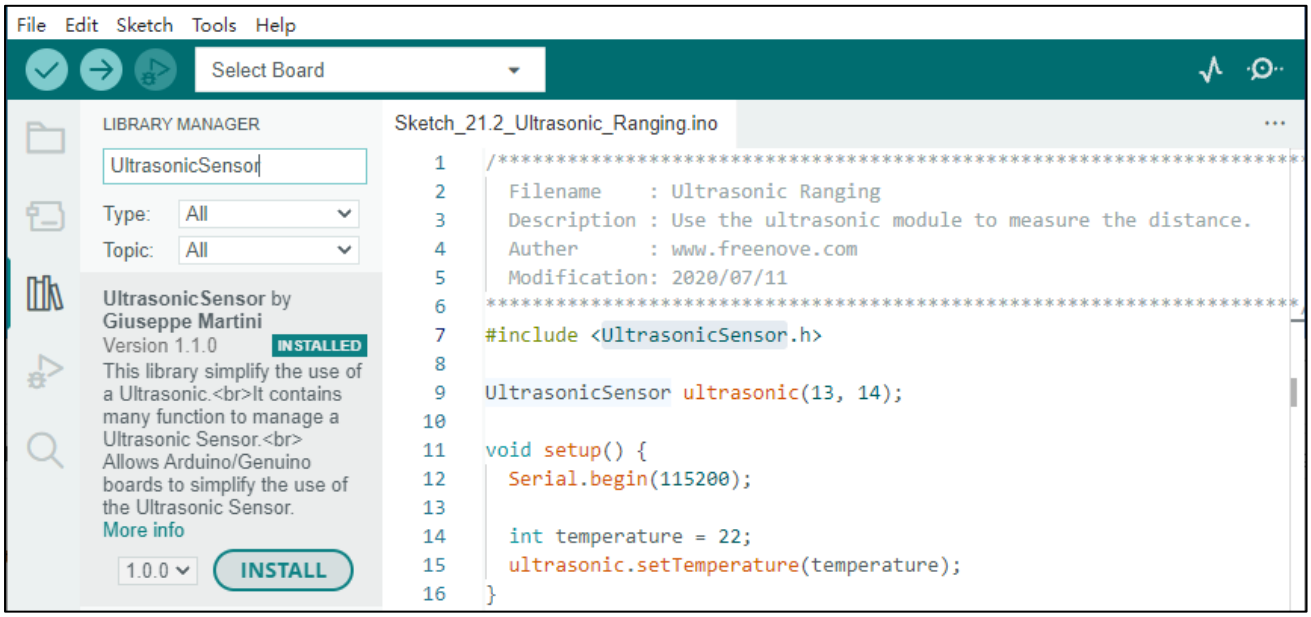
Component List and Circuit

Component List and Circuit are the same as the previous section.

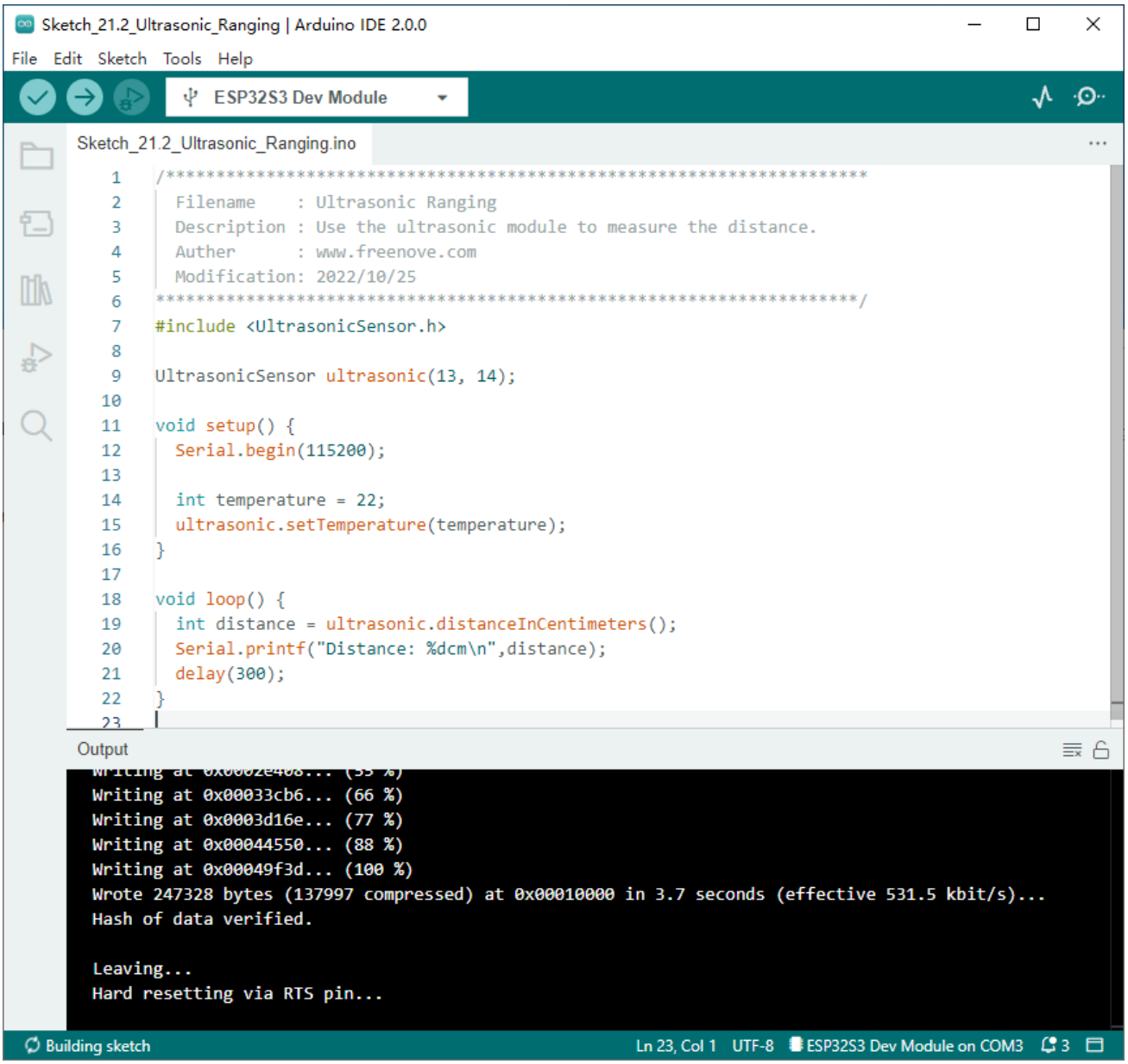
Sketch

How to install the library

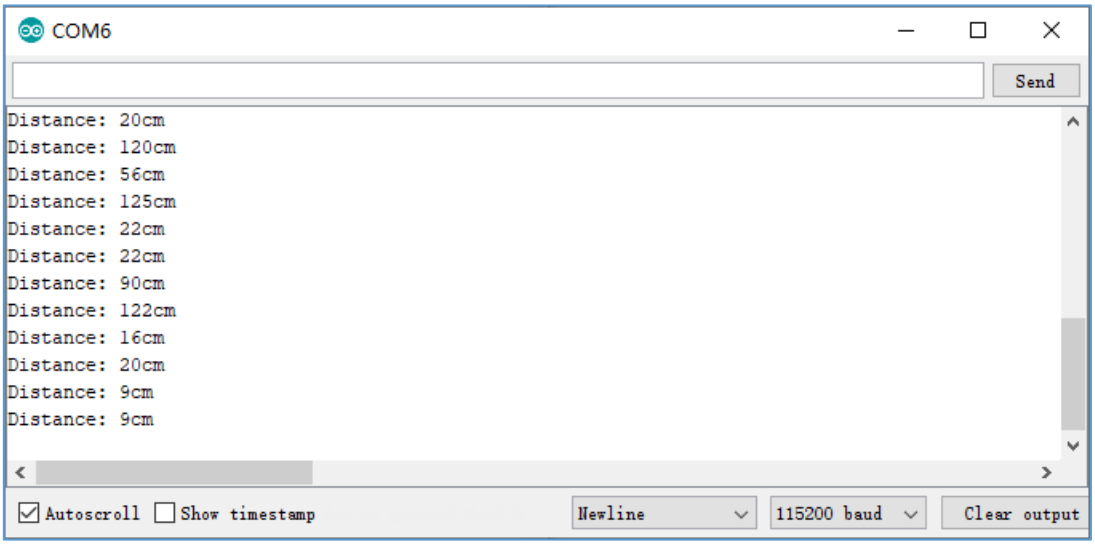
We use the third party library UltrasonicSensor. If you haven't installed it yet, please do so before learning.
The steps to add third-party Libraries are as follows: open arduino->Sketch->Include library-> Manage libraries.
Enter "UltrasonicSensor" in the search bar and select "UltrasonicSensor" for installation.
Refer to the following operations:



Sketch_19.2_Ultrasonic_Ranging



Download the code to ESP32-S3 WROOM, open the serial port monitor, set the baud rate to 115200. Use the ultrasonic module to measure distance. As shown in the following figure:



The following is the program code:

```
#include <UltrasonicSensor.h>
//Attach the trigger and echo pins to pins 13 and 14 of esp32
UltrasonicSensor ultrasonic(13, 14);

void setup() {
  Serial.begin(115200);
  //set the speed of sound propagation according to the temperature to reduce errors
  int temperature = 22; //Setting ambient temperature
  ultrasonic.setTemperature(temperature);
}

void loop() {
  int distance = ultrasonic.distanceInCentimeters();
  Serial.printf("Distance: %dcm\n",distance);
  delay(300);
}
```

First, add UltrasonicSensor library.

```
#include <UltrasonicSensor.h>
```

Define an ultrasonic object and associate the pins.

```
UltrasonicSensor ultrasonic(13, 14);
```

Set the ambient temperature to make the module measure more accurately.

```
ultrasonic.setTemperature(temperature);
```

Use the distanceInCentimeters function to get the distance measured by the ultrasound and print it out through the serial port.

```
void loop() {
  int distance = ultrasonic.distanceInCentimeters();
  Serial.printf("Distance: %dcm\n",distance);
  delay(300);
}
```

Reference

class UltrasonicSensor

class UltrasonicSensor must be instantiated when used, that is, define an object of Servo type, for example:

UltrasonicSensor ultrasonic(13, 14);

setTemperature(value): The speed of sound propagation is different at different temperatures. In order to get more accurate data, this function needs to be called. **value** is the temperature value of the current environment.

`distanceInCentimeters()`: The ultrasonic distance acquisition function returns the value in centimeters.

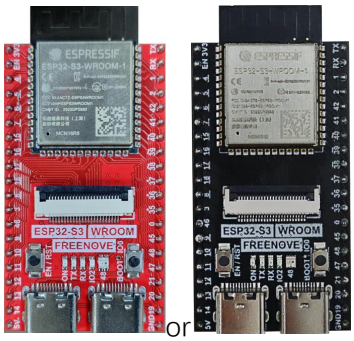
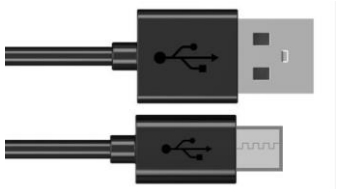
distanceInMillimeters(): The ultrasonic distance acquisition function returns the value in millimeter.

Chapter 20 Bluetooth

This chapter mainly introduces how to make simple data transmission through Bluetooth of ESP32-S3 WROOM and mobile phones.

Project 20.1 Bluetooth Low Energy Data Passthrough

Component List

ESP32-S3(N16R8)/ESP32-S3(N8R8)  or	USB cable x1 

Component knowledge

ESP32-S3's integrated Bluetooth function Bluetooth is a short-distance communication system, which can be divided into two types, namely Bluetooth Low Energy(BLE) and Classic Bluetooth. There are two modes for simple data transmission: master mode and slave mode.

Master mode

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

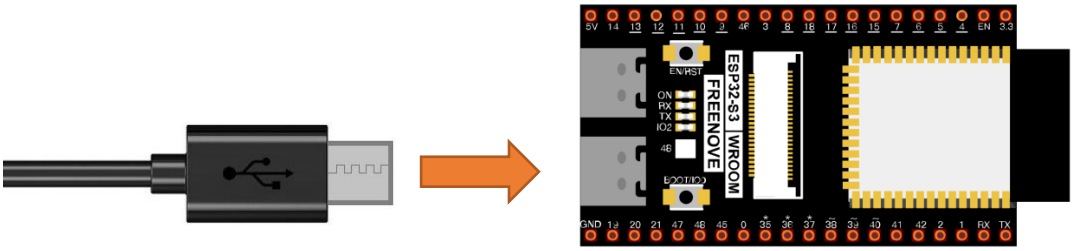
Slave mode

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

Bluetooth devices can make data interaction with each other, as one is in master mode and the other in slave mode. When they are making data interaction, the Bluetooth device in master mode searches and selects devices nearby to connect to. When establishing connection, they can exchange data. When mobile phones exchange data with ESP32-S3, they are usually in master mode and ESP32-S3 in slave mode.

Circuit

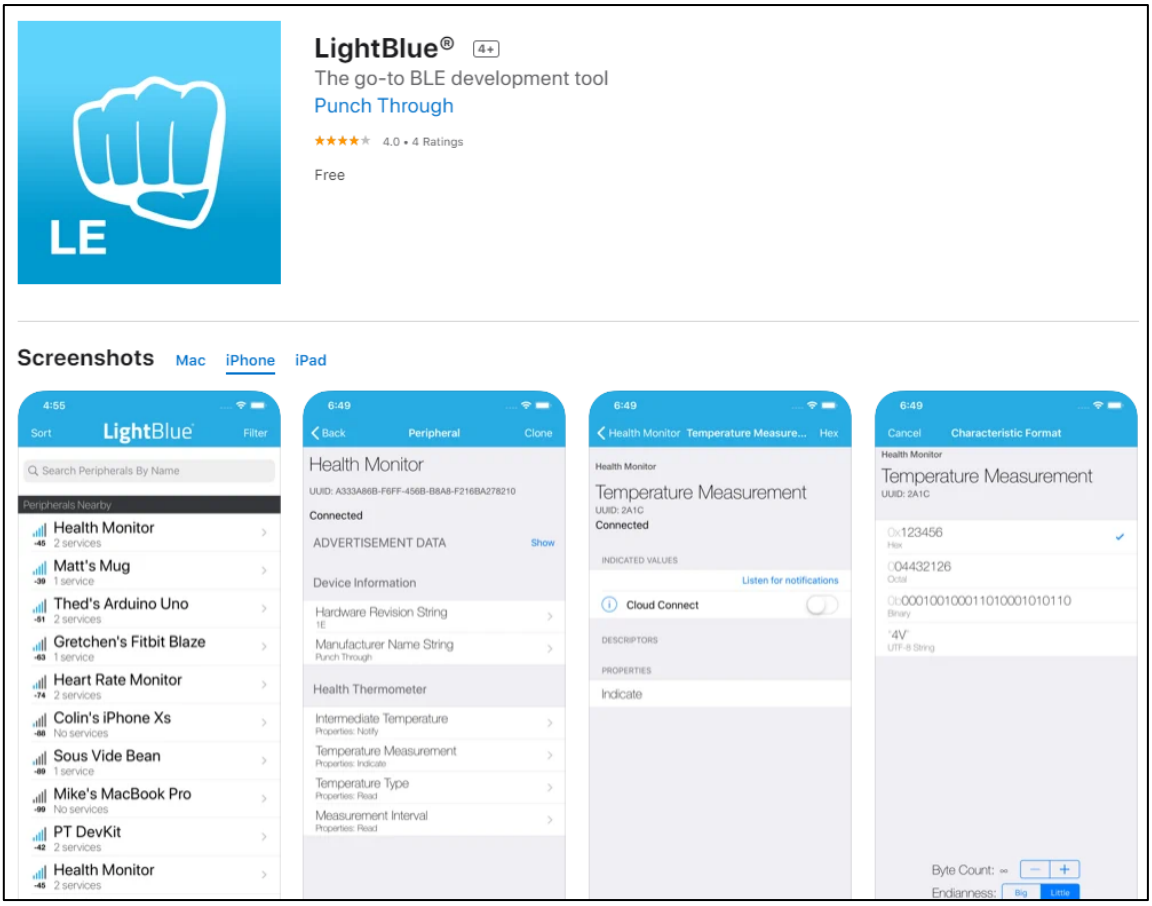
Connect Freenove ESP32-S3 to the computer using the USB cable.



Sketch

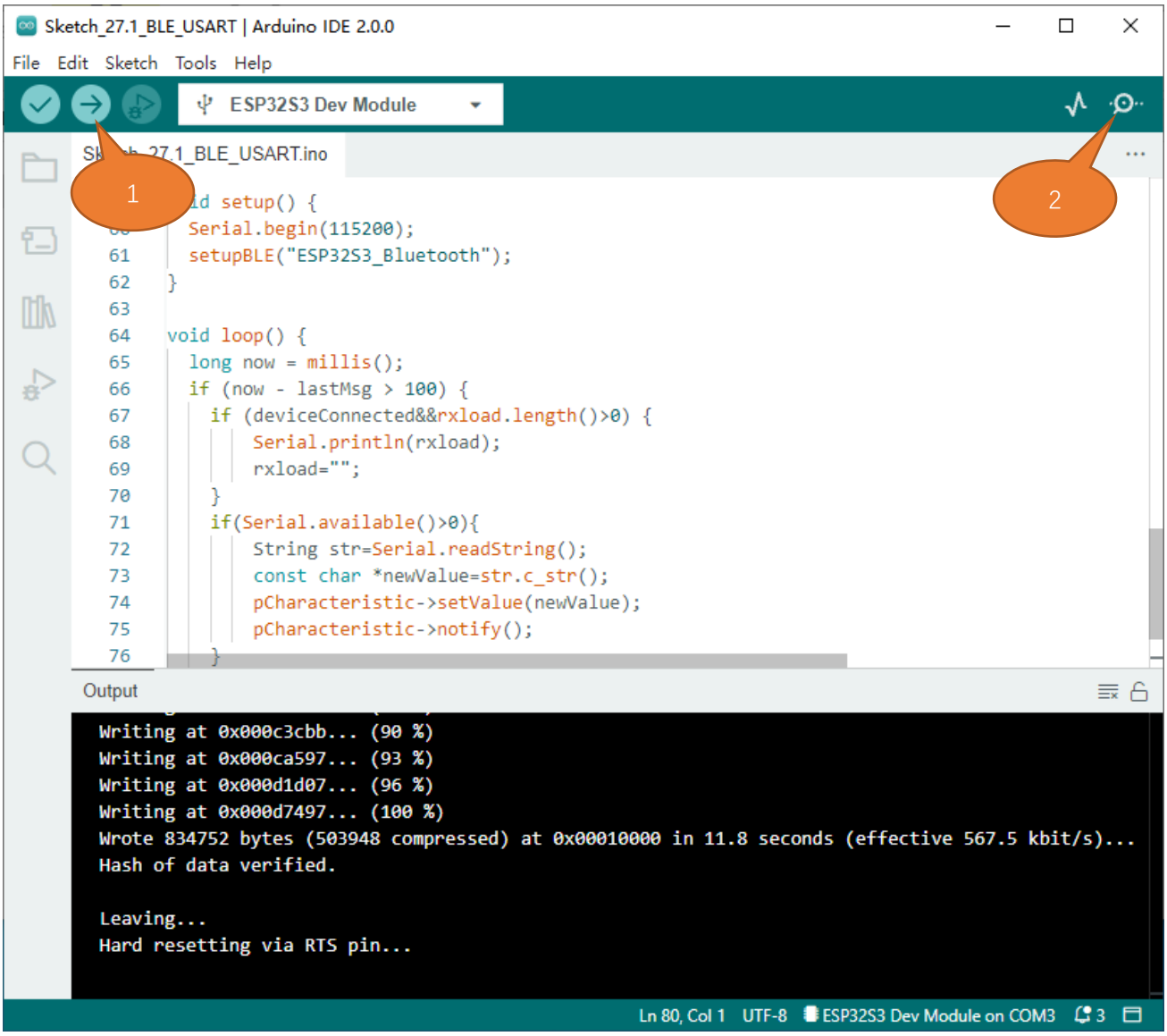
Lightblue

If you can't install Serial Bluetooth on your phone, try LightBlue.If you do not have this software installed on your phone, you can refer to this link:
https://apps.apple.com/us/app/lightblue/id557428110#?platform=iphone.



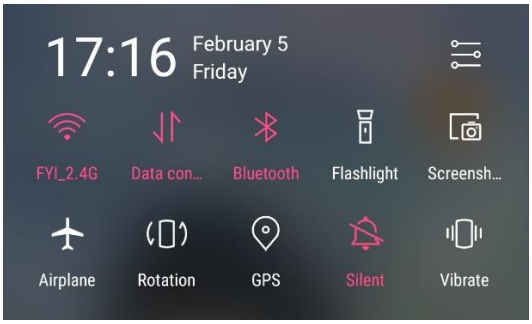
Step1. Upload the code of Project 20.1 to ESP32-S3.

Step2. Click on serial monitor.

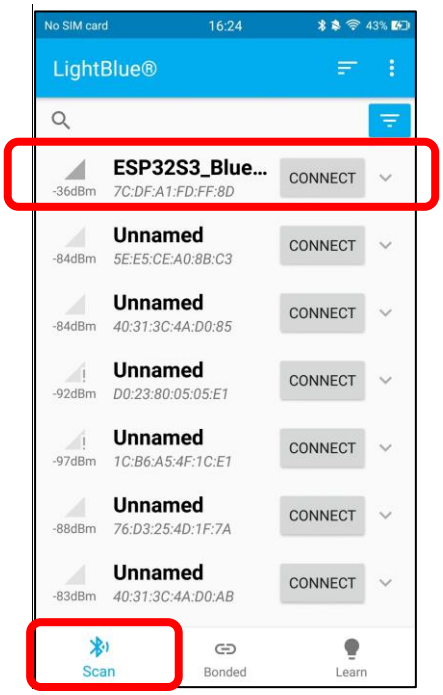


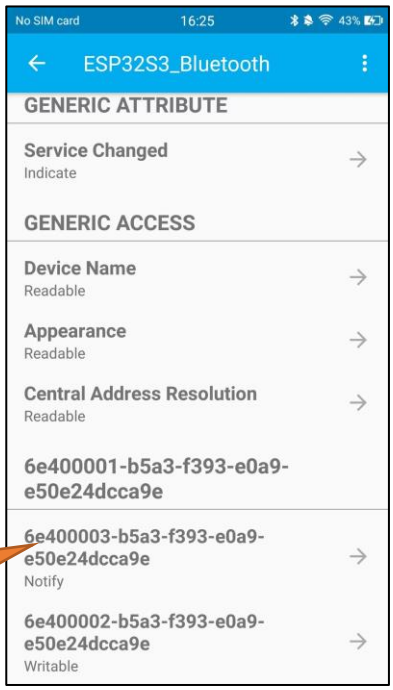
Step3. Set baud rate to 115200.

Turn ON Bluetooth on your phone, and open the Lightblue APP.



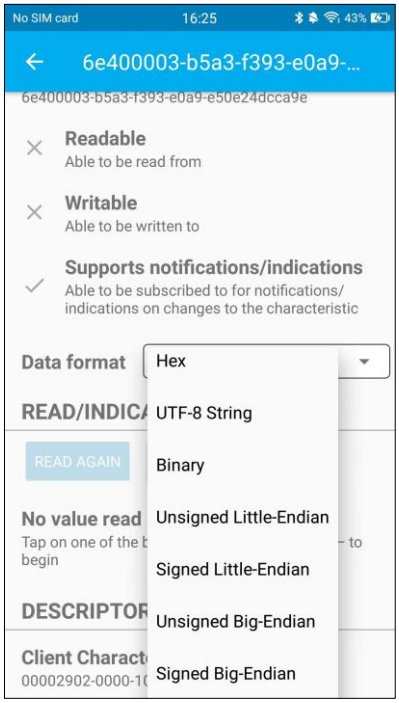
In the Scan page, swipe down to refresh the name of Bluetooth that the phone searches for. Click ESP32S3_Bluetooth.

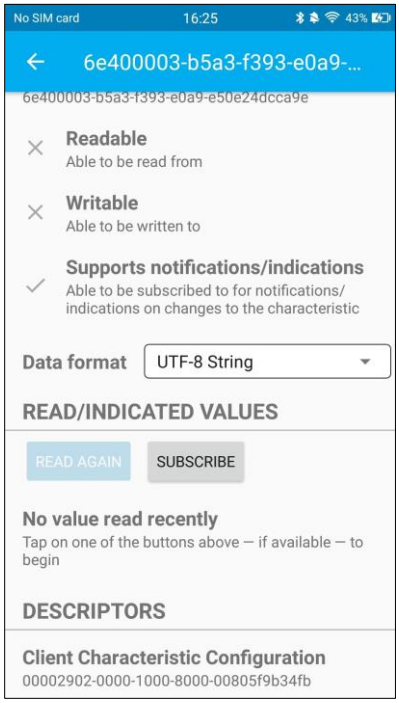




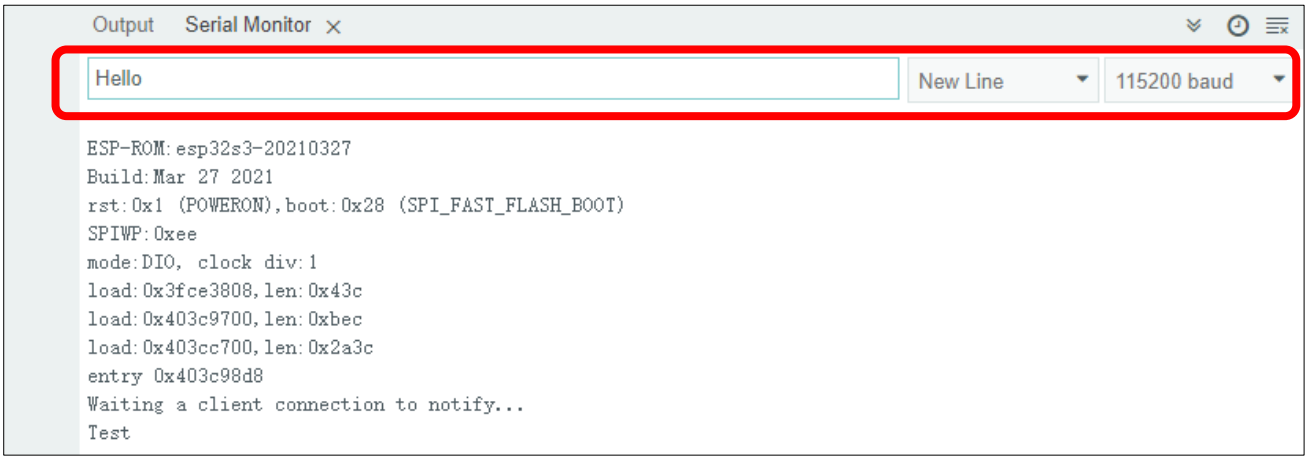
Receive

Click "Receive". Select the appropriate Data format in the box to the right of Data Format. For example, HEX for hexadecimal, utf-string for character, Binary for Binary, etc. Then click SUBSCRIBE.

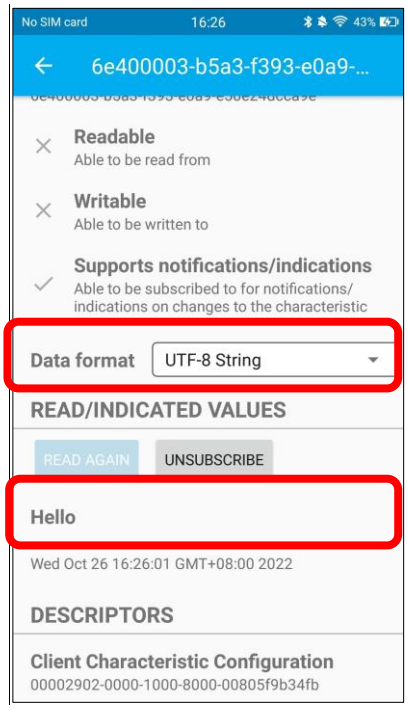




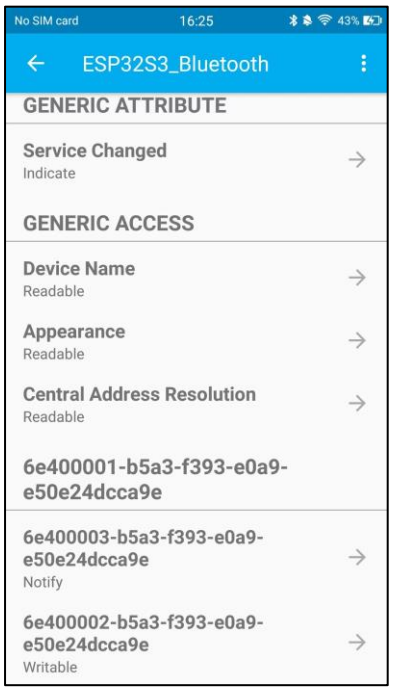
Back to the serial monitor on your computer. You can type anything in the left border of Send, and then click Send.

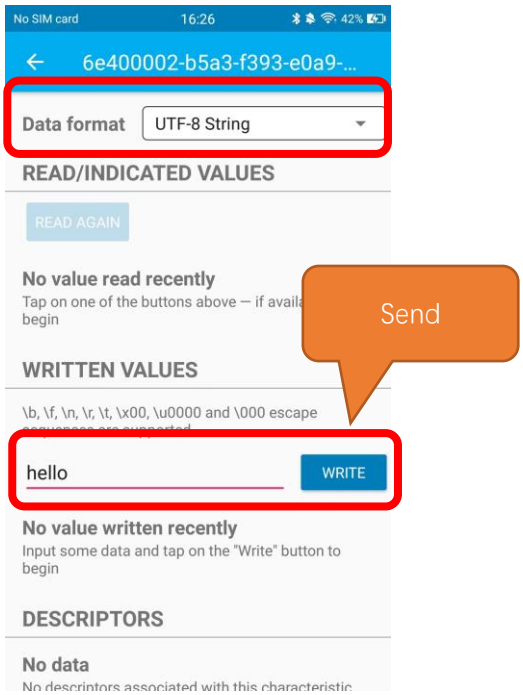


And then you can see the mobile Bluetooth has received the message.

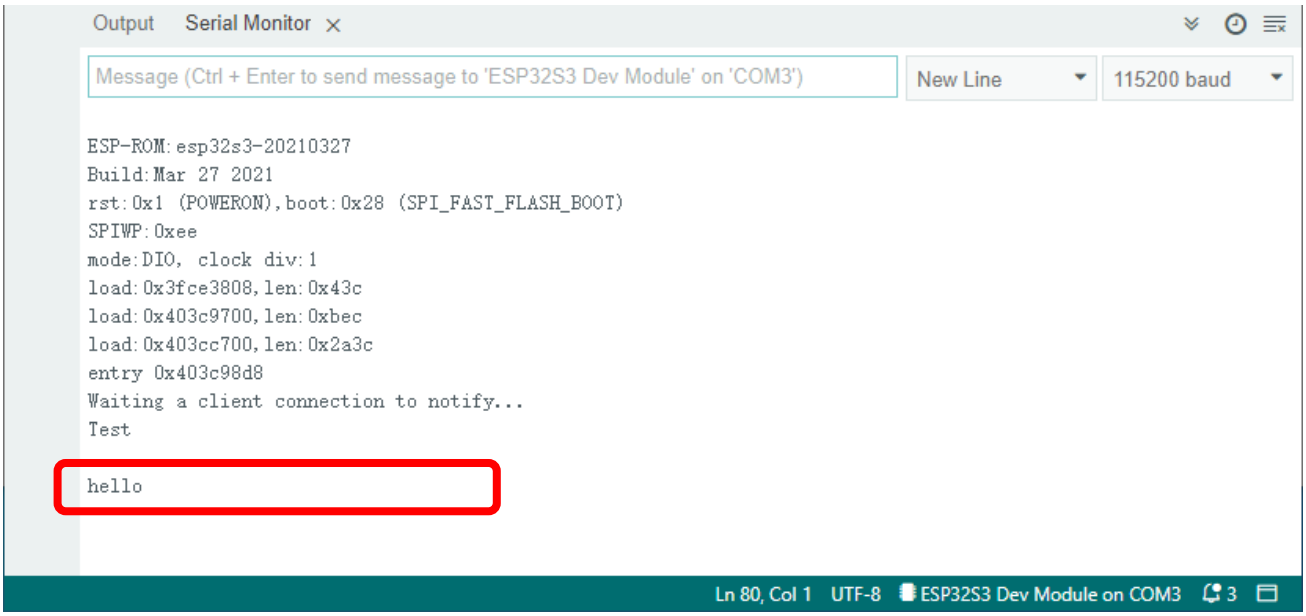


Similarly, you can select "Send" on your phone. Set Data format, and then enter anything in the sending box and click Write to send.





And the computer will receive the message from the mobile Bluetooth.


Output
Serial Monitor
Message (Ctrl + Enter to send message to 'ESP32S3 Dev Module' on 'COM3')
New Line
115200 baud
ESP-ROM:esp32s3-20210327
Build:Mar 27 2021
rst:0x1 (POWERON),boot:0x28 (SPI_FAST_FLASH_BOOT)
SPIWP:0xee
mode:DIO, clock div:1
load:0x3fce3808,len:0x43c
load:0x403c9700,len:0xbec
load:0x403cc700,len:0x2a3c
entry 0x403c98d8
Waiting a client connection to notify...
Test
hello
Ln 80, Col 1
UTF-8
ESP32S3 Dev Module on COM3
3

And now data can be transferred between your mobile phone and computer via ESP32-S3 WROOM.
The following is the program code:

```
#include <BLEDevice.h>
#include <BLEServer.h>
#include <BLEUtils.h>
#include <BLE2902.h>

BLECharacteristic *pCharacteristic;
bool deviceConnected = false;
uint8_t txValue = 0;
long lastMsg = 0;
String rxload="Test\n";

#define SERVICE_UUID           "6E400001-B5A3-F393-E0A9-E50E24DCCA9E"
#define CHARACTERISTIC_UUID_RX "6E400002-B5A3-F393-E0A9-E50E24DCCA9E"
#define CHARACTERISTIC_UUID_TX "6E400003-B5A3-F393-E0A9-E50E24DCCA9E"

class MyServerCallbacks: public BLEServerCallbacks {
    void onConnect(BLEServer* pServer) {
      deviceConnected = true;
    };
    void onDisconnect(BLEServer* pServer) {
      deviceConnected = false;
    }
};

class MyCallbacks: public BLECharacteristicCallbacks {
    void onWrite(BLECharacteristic *pCharacteristic) {
      String rxValue = pCharacteristic->getValue();
      if (rxValue.length() > 0) {
        rxload="";
        for (int i = 0; i < rxValue.length(); i++){
          rxload +=(char)rxValue[i];
        }
      }
    }
};

void setupBLE(String BLEName){
  const char *ble_name=BLEName.c_str();
  BLEDevice::init(ble_name);
  BLEServer *pServer = BLEDevice::createServer();
  pServer->setCallbacks(new MyServerCallbacks());
  BLEService *pService = pServer->createService(SERVICE_UUID);
```

```
43    pCharacteristic=
      pService->createCharacteristic(CHARACTERISTIC_UUID_TX,BLECharacteristic::PROPERTY_NOTIFY);
44    pCharacteristic->addDescriptor(new BLE2902());
45    BLECharacteristic *pCharacteristic =
      pService->createCharacteristic(CHARACTERISTIC_UUID_RX,BLECharacteristic::PROPERTY_WRITE);
46    pCharacteristic->setCallbacks(new MyCallbacks());
47    pService->start();
48    pServer->getAdvertising()->start();
49    Serial.println("Waiting a client connection to notify...");
50  }
51
52  void setup() {
53    Serial.begin(115200);
54    setupBLE("ESP32S3_Bluetooth");
55  }
56
57  void loop() {
58    long now = millis();
59    if (now - lastMsg > 1000) {
60      if (deviceConnected&&rxload.length()>0) {
61          Serial.println(rxload);
62          rxload="";
63      }
64      if(Serial.available()>0){
65          String str=Serial.readString();
66          const char *newValue=str.c_str();
67          pCharacteristic->setValue(newValue);
68          pCharacteristic->notify();
69      }
70      lastMsg = now;
71    }
72  }
```

Define the specified UUID number for BLE vendor.

```
12  #define SERVICE_UUID           "6E400001-B5A3-F393-E0A9-E50E24DCCA9E"
13  #define CHARACTERISTIC_UUID_RX "6E400002-B5A3-F393-E0A9-E50E24DCCA9E"
14  #define CHARACTERISTIC_UUID_TX "6E400003-B5A3-F393-E0A9-E50E24DCCA9E"
```

Write a Callback function for BLE server to manage connection of BLE.

```
16 class MyServerCallbacks: public BLEServerCallbacks {
17     void onConnect(BLEServer* pServer) {
18       deviceConnected = true;
19     };
20     void onDisconnect(BLEServer* pServer) {
21       deviceConnected = false;
22     }
23 };
```

Write Callback function with BLE features. When it is called, as the mobile terminal send data to ESP32-S3, it will store them into reload.

```
25 class MyCallbacks: public BLECharacteristicCallbacks {
26     void onWrite(BLECharacteristic *pCharacteristic) {
27       String rxValue = pCharacteristic->getValue();
28       if (rxValue.length() > 0) {
29         rxload="";
30         for (int i = 0; i < rxValue.length(); i++){
31           rxload +=(char)rxValue[i];
32         }
33       }
34     }
35 };
```

Initialize the BLE function and name it.

```
54 setupBLE("ESP32S3_Bluetooth");
```

When the mobile phone send data to ESP32-S3 via BLE Bluetooth, it will print them out with serial port; When the serial port of ESP32-S3 receive data, it will send them to mobile via BLE Bluetooth.

```
58   long now = millis();
59   if (now - lastMsg > 1000) {
60     if (deviceConnected&&rxload.length()>0) {
61         Serial.println(rxload);
62         rxload="";
63     }
64     if(Serial.available()>0){
65         String str=Serial.readString();
66         const char *newValue=str.c_str();
67         pCharacteristic->setValue(newValue);
68         pCharacteristic->notify();
69     }
70     lastMsg = now;
71   }
```

The design for creating the BLE server is:

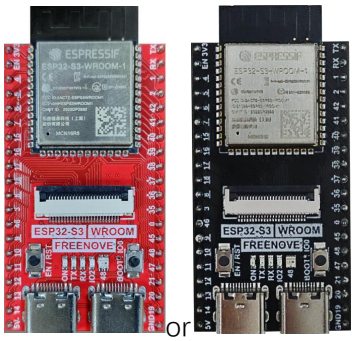
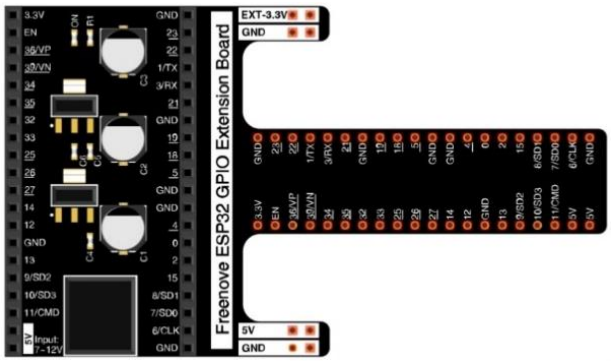
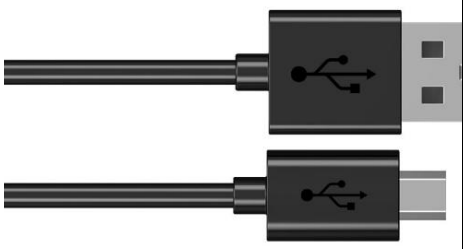
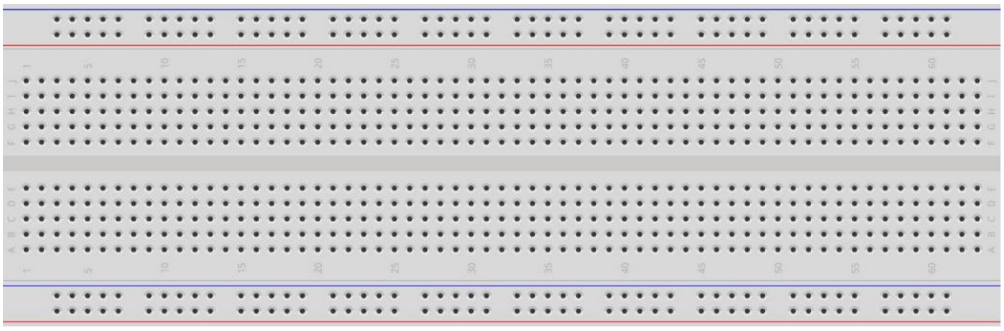
1. Create a BLE Server
2. Create a BLE Service
3. Create a BLE Characteristic on the Service
4. Create a BLE Descriptor on the characteristic
5. Start the service.
6. Start advertising.

```
37  void setupBLE(String BLEName){
38    const char *ble_name=BLEName.c_str();
39    BLEDevice::init(ble_name);
40    BLEServer *pServer = BLEDevice::createServer();
41    pServer->setCallbacks(new MyServerCallbacks());
42    BLEService *pService = pServer->createService(SERVICE_UUID);
43    pCharacteristic=
      pService->createCharacteristic(CHARACTERISTIC_UUID_TX,BLECharacteristic::PROPERTY_NOTIFY);
44    pCharacteristic->addDescriptor(new BLE2902());
45    BLECharacteristic *pCharacteristic =
      pService->createCharacteristic(CHARACTERISTIC_UUID_RX,BLECharacteristic::PROPERTY_WRITE);
46    pCharacteristic->setCallbacks(new MyCallbacks());
      pService->start();
47    pServer->getAdvertising()->start();
48    Serial.println("Waiting a client connection to notify...");
49  }
```

Project 20.2 Bluetooth Control LED

In this section, we will control the LED with Bluetooth.

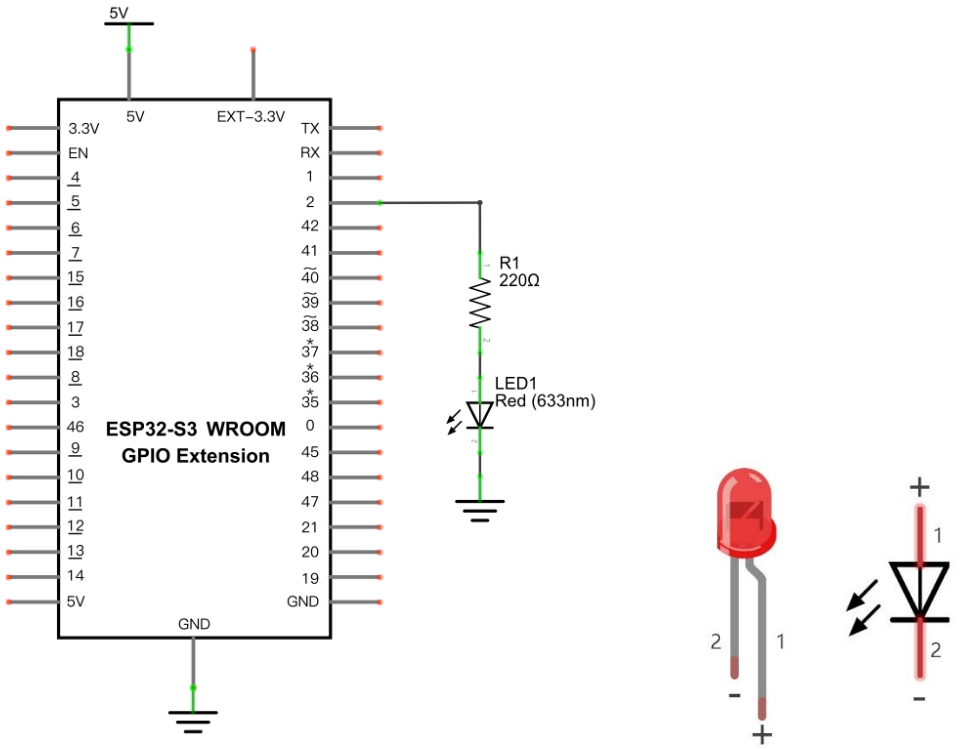
Component List

<table>
<tr><td>ESP32-S3(N16R8)/ESP32-S3(N8R8)
  </td><td colspan="3">GPIO Extension Board x1
  </td></tr>
<tr><td>Micro USB Wire x1
  </td><td>LED x1
  </td><td>Resistor 220Ω x1
  </td><td>Jumper M/M x2
  </td></tr>
<tr><td colspan="4">Breadboard x1
  </td></tr>
</table>

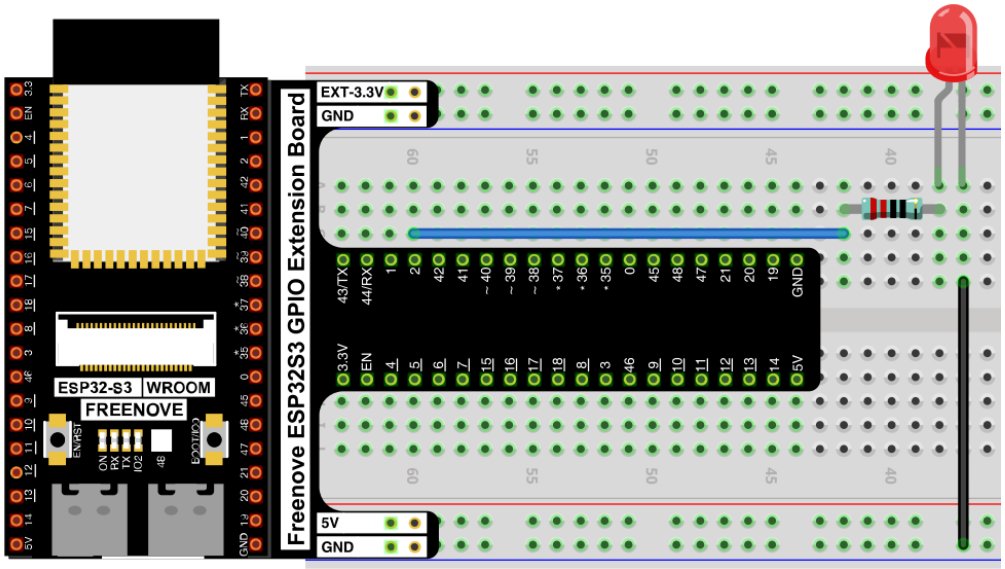
Circuit

Connect Freenove ESP32-S3 to the computer using a USB cable.

Schematic diagram

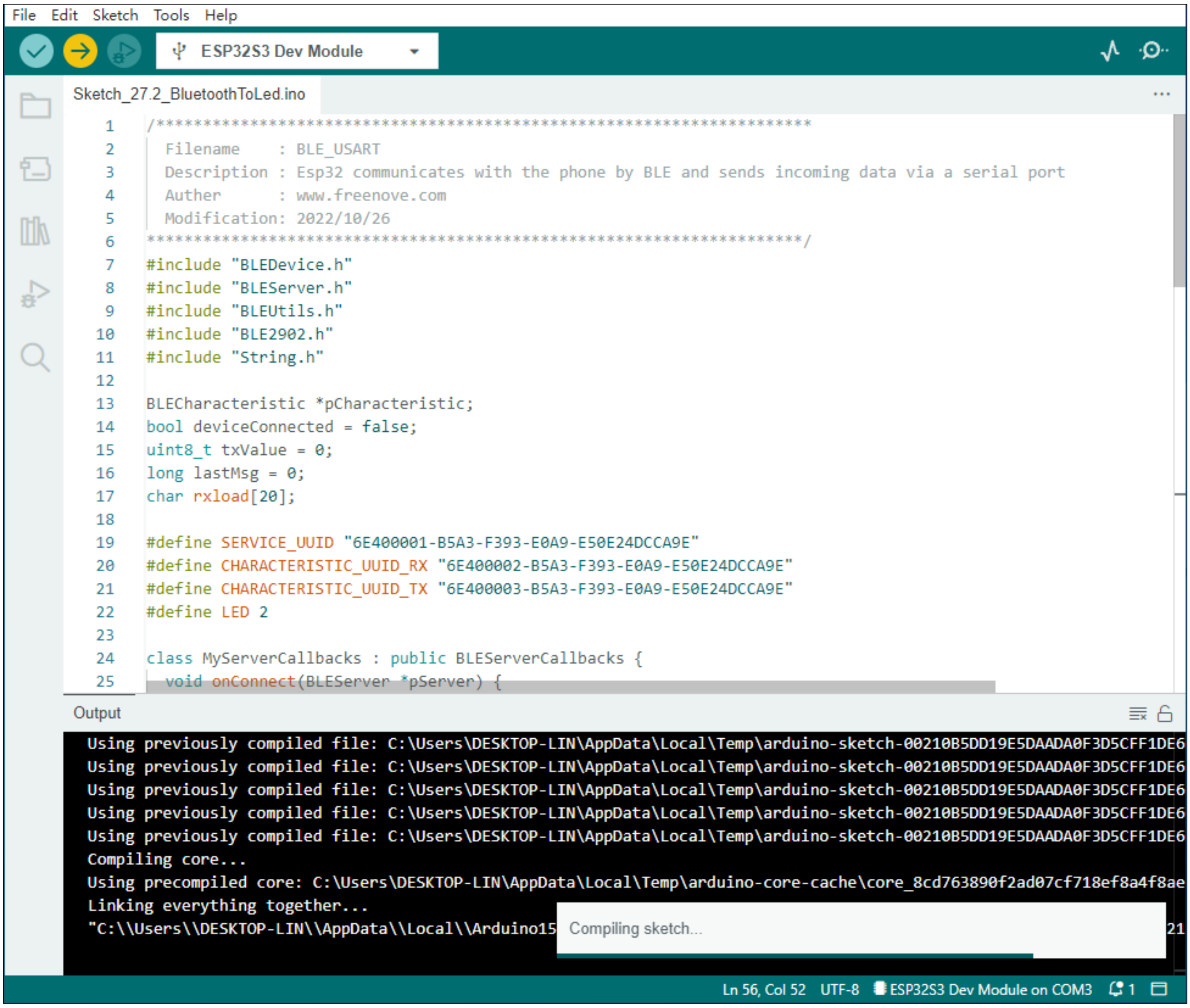


Hardware connection. **If you need any support, please contact us via: support@freenove.com**



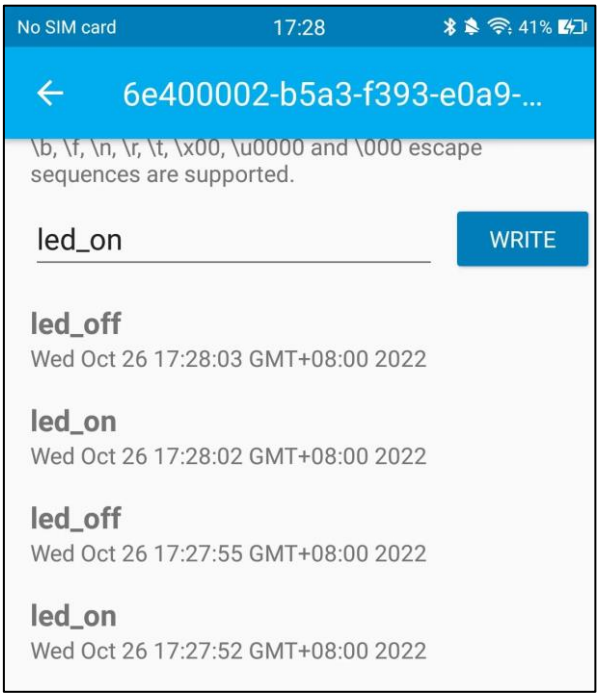
Sketch

Sketch_20.2_Bluetooth_Control_LED



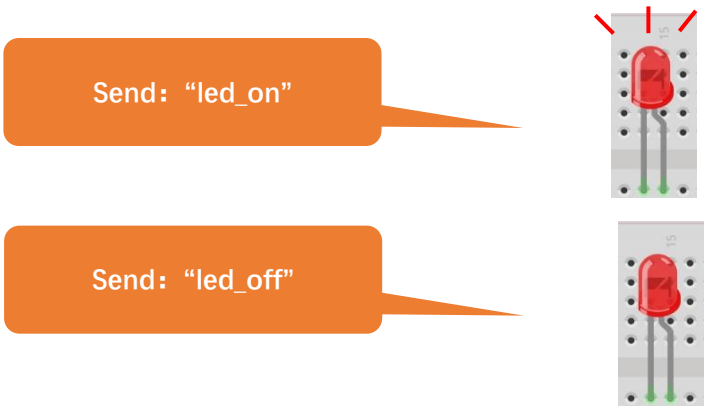
Compile and upload code to **ESP32S3_Blueooth**. The operation of the APP is the same as 27.1, you only need to change the sending content to "**led_on**" and "**led_off**" to operate LEDs on the ESP32-S3 WROOM.

Data sent from mobile APP:



Display on the serial port of the computer:

The phenomenon of LED



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

The following is the program code:

```
#include "BLEDevice.h"
#include "BLEServer.h"
#include "BLEUtils.h"
#include "BLE2902.h"
#include "String.h"

BLECharacteristic *pCharacteristic;
bool deviceConnected = false;
uint8_t txValue = 0;
long lastMsg = 0;
char rxload[20];

#define SERVICE_UUID "6E400001-B5A3-F393-E0A9-E50E24DCCA9E"
#define CHARACTERISTIC_UUID_RX "6E400002-B5A3-F393-E0A9-E50E24DCCA9E"
#define CHARACTERISTIC_UUID_TX "6E400003-B5A3-F393-E0A9-E50E24DCCA9E"
#define LED 2

class MyServerCallbacks : public BLEServerCallbacks {
  void onConnect(BLEServer *pServer) {
    deviceConnected = true;
  };
  void onDisconnect(BLEServer *pServer) {
    deviceConnected = false;
  }
};

class MyCallbacks : public BLECharacteristicCallbacks {
  void onWrite(BLECharacteristic *pCharacteristic) {
    String rxValue = pCharacteristic->getValue();
    if (rxValue.length() > 0) {
      for (int i = 0; i < 20; i++) {
        rxload[i] = 0;
      }
      for (int i = 0; i < rxValue.length(); i++) {
        rxload[i] = (char)rxValue[i];
      }
    }
  }
};

void setupBLE(String BLEName) {
  const char *ble_name = BLEName.c_str();
  BLEDevice::init(ble_name);
```

```
  BLEServer *pServer = BLEDevice::createServer();
  pServer->setCallbacks(new MyServerCallbacks());
  BLEService *pService = pServer->createService(SERVICE_UUID);
  pCharacteristic = pService->createCharacteristic(CHARACTERISTIC_UUID_TX,
BLECharacteristic::PROPERTY_NOTIFY);
  pCharacteristic->addDescriptor(new BLE2902());
  BLECharacteristic *pCharacteristic = pService->createCharacteristic(CHARACTERISTIC_UUID_RX,
BLECharacteristic::PROPERTY_WRITE);
  pCharacteristic->setCallbacks(new MyCallbacks());
  pService->start();
  pServer->getAdvertising()->start();
  Serial.println("Waiting a client connection to notify...");
}

void setup() {
  pinMode(LED, OUTPUT);
  setupBLE("ESP32S3_Bluetooth");
  Serial.begin(115200);
  Serial.println("\nThe device started, now you can pair it with Bluetooth!");
}

void loop() {
  long now = millis();
  if (now - lastMsg > 100) {
    if (deviceConnected && strlen(rxload) > 0) {
      if (strncmp(rxload, "led_on", 6) == 0) {
        digitalWrite(LED, HIGH);
      }
      if (strncmp(rxload, "led_off", 7) == 0) {
        digitalWrite(LED, LOW);
      }
      Serial.println(rxload);
      memset(rxload,0,sizeof(rxload));
    }
    lastMsg = now;
  }
}
```

Use character string to handle function header file.

```
#include "string.h"
```

Define a character array to save data from Bluetooth.

```
char rxload[20];
```

Initialize the BLE Bluetooth and name it as "ESP32-S3"

```
  setupBLE("ESP32S3_Bluetooth");
```

Write a Callback function for BLE server to manage connection of BLE.

```
class MyServerCallbacks: public BLEServerCallbacks {
    void onConnect(BLEServer* pServer) {
      deviceConnected = true;
    };
    void onDisconnect(BLEServer* pServer) {
      deviceConnected = false;
    }
};
```

Write Callback function with BLE features. When it is called, as the mobile terminal send data to ESP32-S3, it will store them into reload.

```
      String rxValue = pCharacteristic->getValue();
      if (rxValue.length() > 0) {
        rxload="";
        for (int i = 0; i < rxValue.length(); i++){
          rxload +=(char)rxValue[i];
        }
      }
```

Compare the content in buffer array with "led_on" and "led_off" to see whether they are the same. If yes, execute the corresponding operation.

```
if (deviceConnected && strlen(rxload) > 0) {
      if (strncmp(rxload, "led_on", 6) == 0) {
        digitalWrite(LED, HIGH);
      }
      if (strncmp(rxload, "led_off", 7) == 0) {
        digitalWrite(LED, LOW);
      }
      Serial.println(rxload);
    }
```

After comparing the content of array, to ensure successful transmission next time, please empty the array.

```
      Serial.println(rxload);
      memset(rxload,0,sizeof(rxload));
```

Reference

strncmp() functions are often used for string comparisons, which are accurate and stable.

int strncmp(const char *str1, const char *str2, size_t n)
str1: the first string to be compared
str2: the second string to be compared
n: the biggest string to be compared
Return value: if stir1>str2, then return value>0.
If return value is 0, then the contents of str1 and str2 are the same.
If str1< str2, then return value<0.

Function memset is mainly used to clean and initialize the memory of array

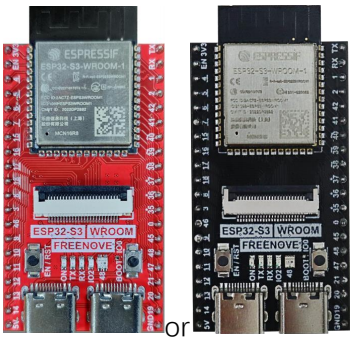
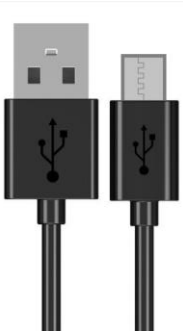
void memset(void *s, int c, unsigned long n)
Function memset() is to set the content of a certain internal storage as specified value.
*s: the initial address of the content to clear out.
c:to be replaced as specified value
n: the number of byte to be replaced

Chapter 21 Read and Write the SDcard

An SDcard slot is integrated on the back of the ESP32-S3 WROOM. In this chapter we learn how to use ESP32-S3 to read and write SDcard.

Project 21.1 SDMMC Test

Component List

ESP32-S3(N16R8)/ESP32-S3(N8R8)  	USB cable x1 
SDcard reader x1 (random color)  (Not a USB flash drive.)	SDcard x1 

Component knowledge

SD card read and write method

ESP32-S3 has two ways to use SD card, one is to use the SPI interface to access the SD card, and the other is to use the SDMMC interface to access the SD card. SPI mode uses 4 IOs to access SD card. The SDMMC has one-bit bus mode and four-bit bus mode. In one-bit bus mode, SDMMC use 3 IOs to access SD card. In four-bit bus mode, SDMMC uses 6 IOs to access the SD card.

The above three methods can all be used to access the SD card, the difference is that the access speed is different.

In the four-bit bus mode of SDMMC, the reading and writing speed of accessing the SD card is the fastest. In the one-bit bus mode of SDMMC, the access speed is about 80% of the four-bit bus mode. The access speed of SPI is the slowest, which is about 50% of the four-bit bus mode of SDMMC.

Usually, we recommend using the one-bit bus mode to access the SD card, because in this mode, we only need to use the least pin IO to access the SD card with good performance and speed.

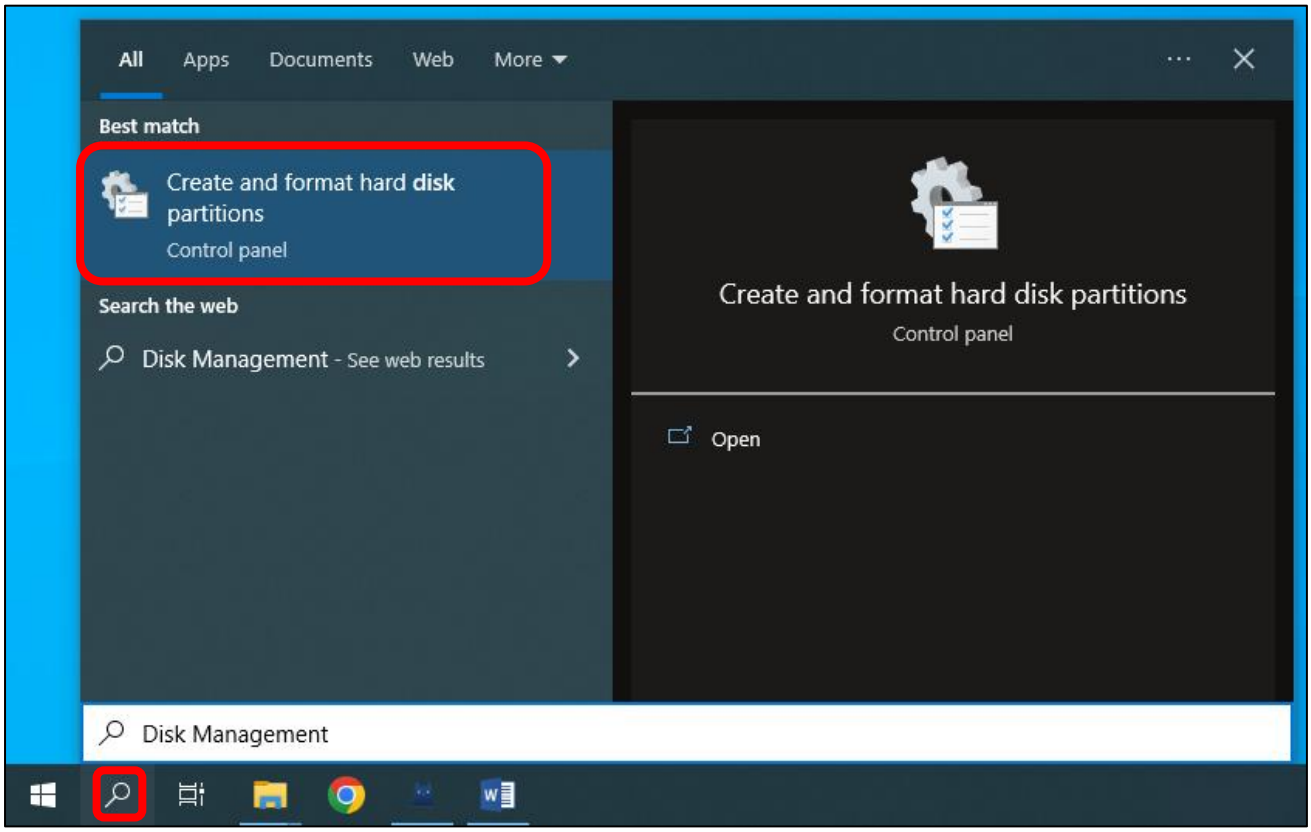
Format SD card

Before starting the tutorial, we need to create a drive letter for the blank SD card and format it. This step requires a card reader and SD card. Please prepare them in advance. Below we will guide you to do it on different computer systems. You can choose the guide that matches your computer.

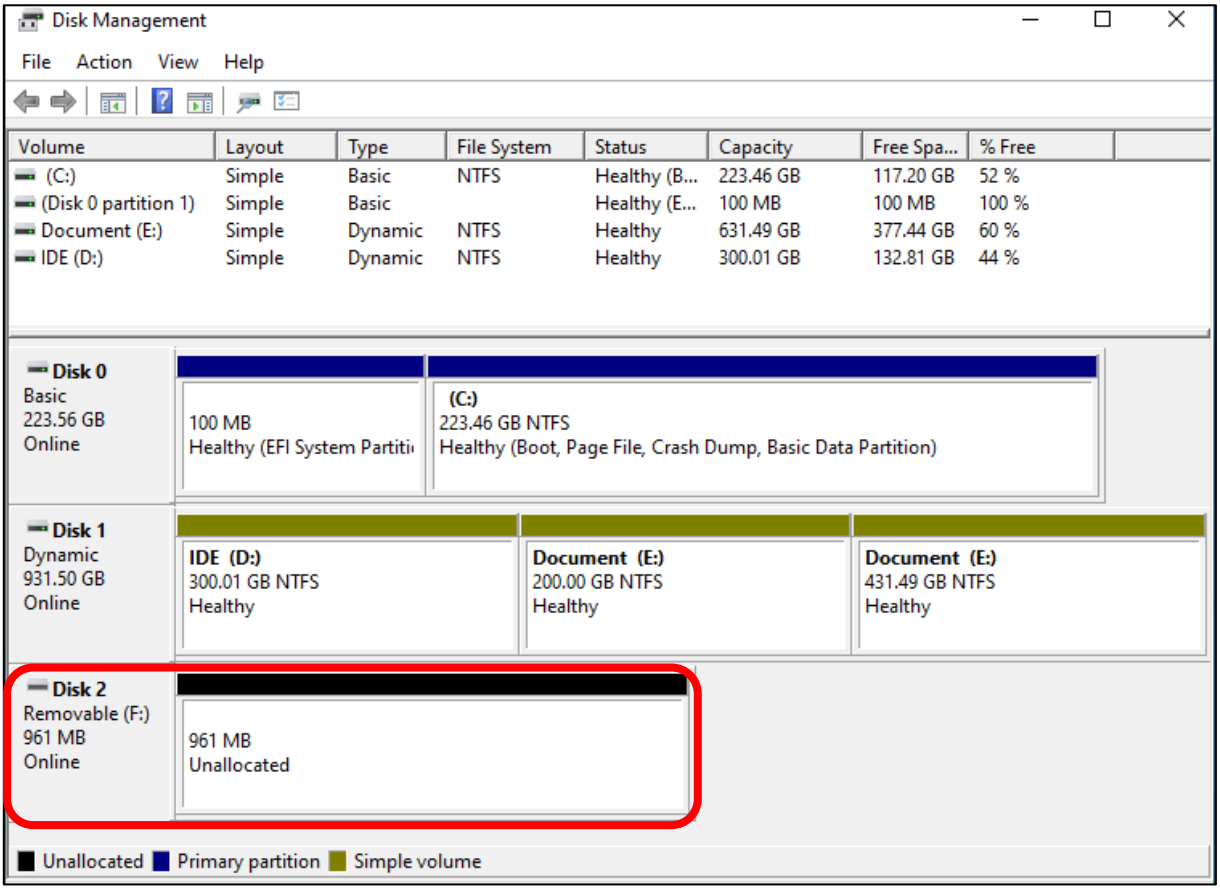
Windows

Insert the SD card into the card reader, then insert the card reader into the computer.

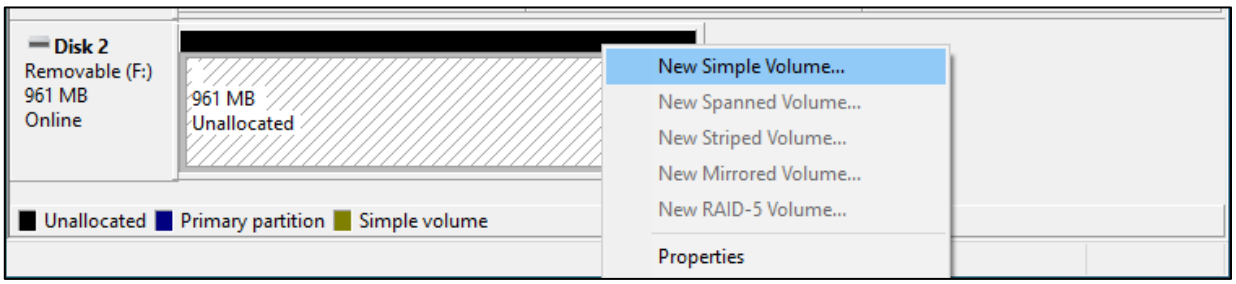
In the Windows search box, enter "Disk Management" and select "Create and format hard disk partitions".



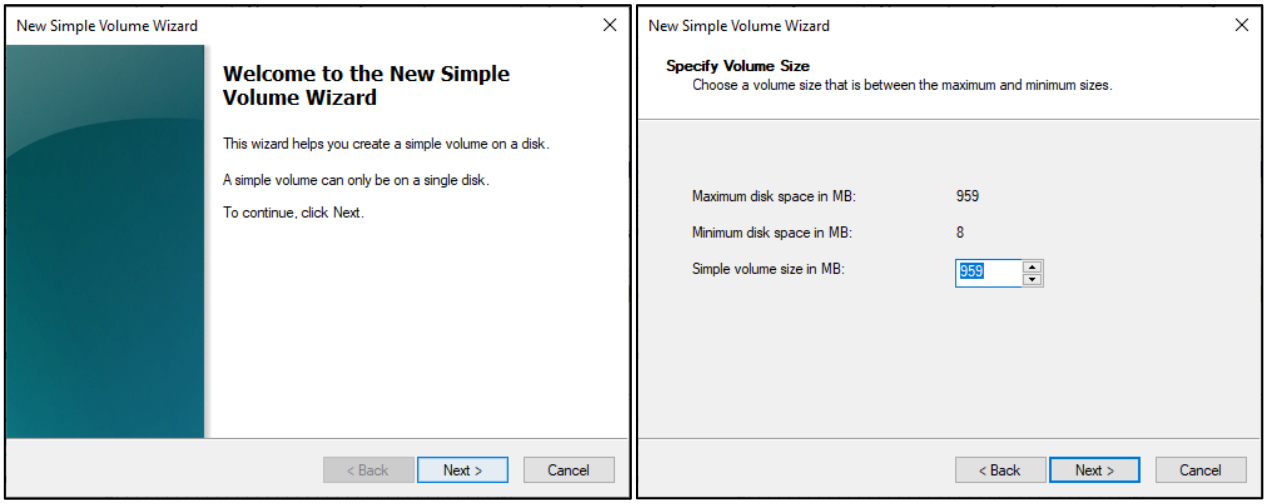
In the new pop-up window, find an unallocated volume close to 1G in size.



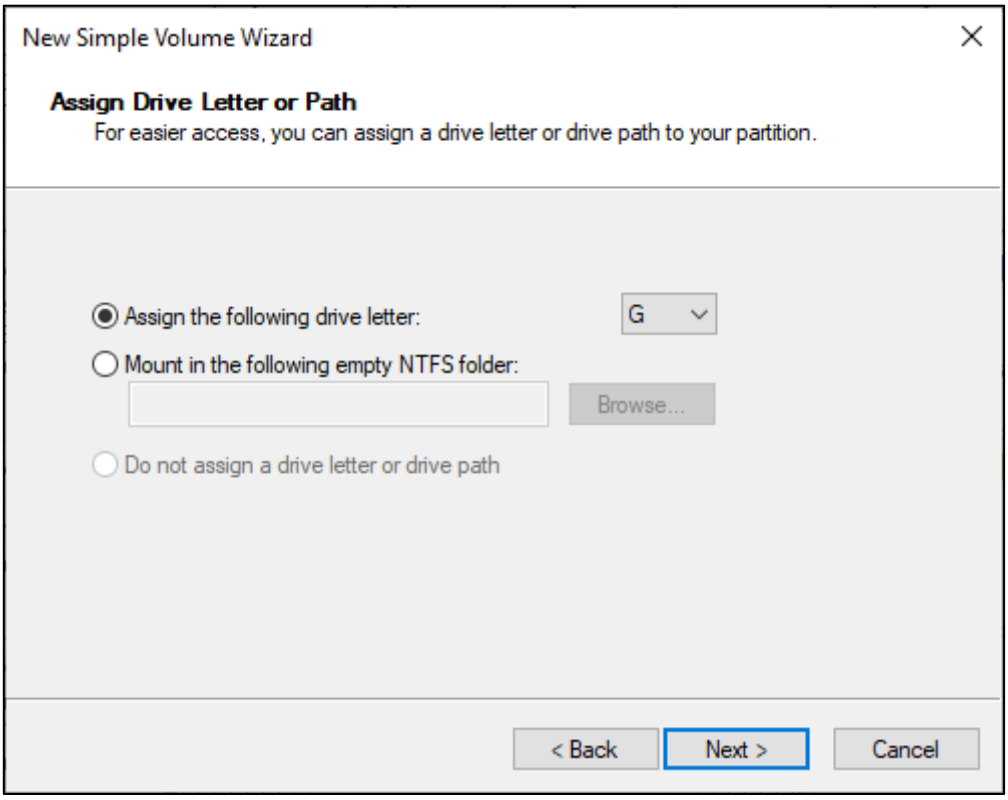
Click to select the volume, right-click and select "New Simple Volume".



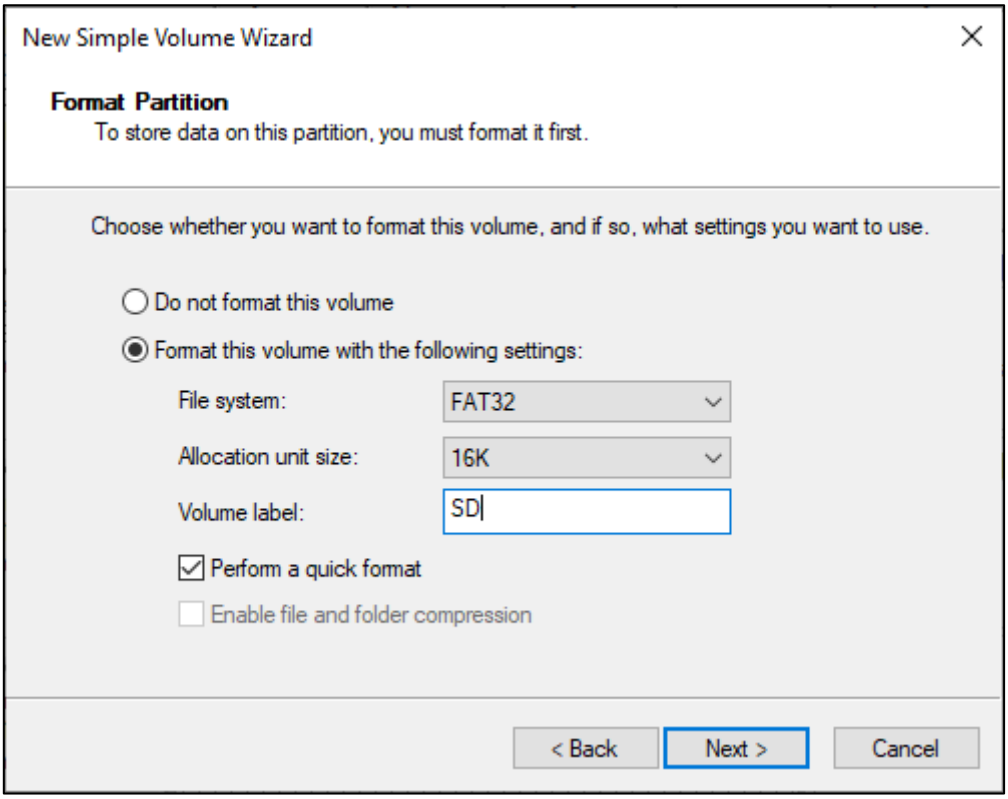
Click Next.



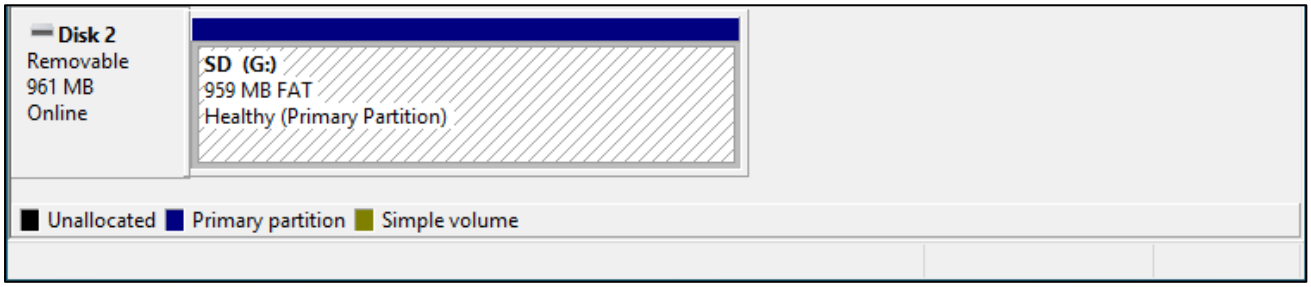
You can choose the drive letter on the right, or you can choose the default. By default, just click Next.



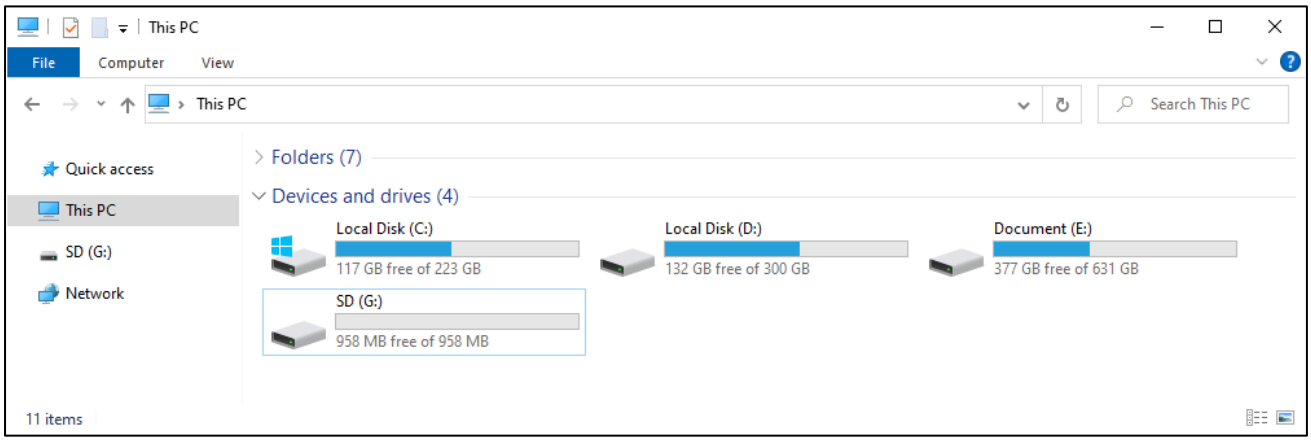
File system is FAT(or FAT32). The Allocation unit size is 16K, and the Volume label can be set to any name. After setting, click Next.



Click Finish. Wait for the SD card initialization to complete.

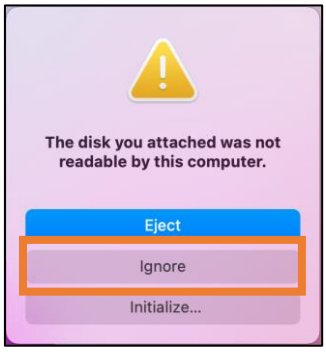


At this point, you can see the SD card in This PC.

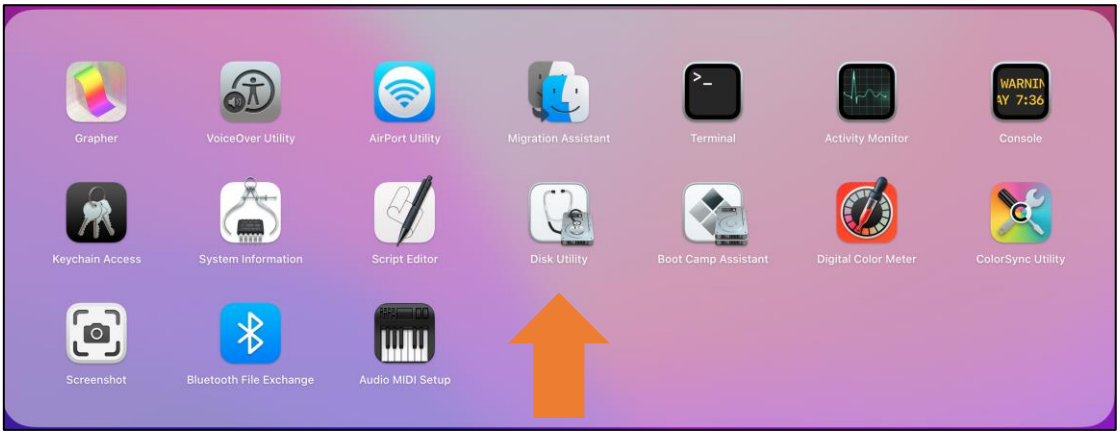


MAC

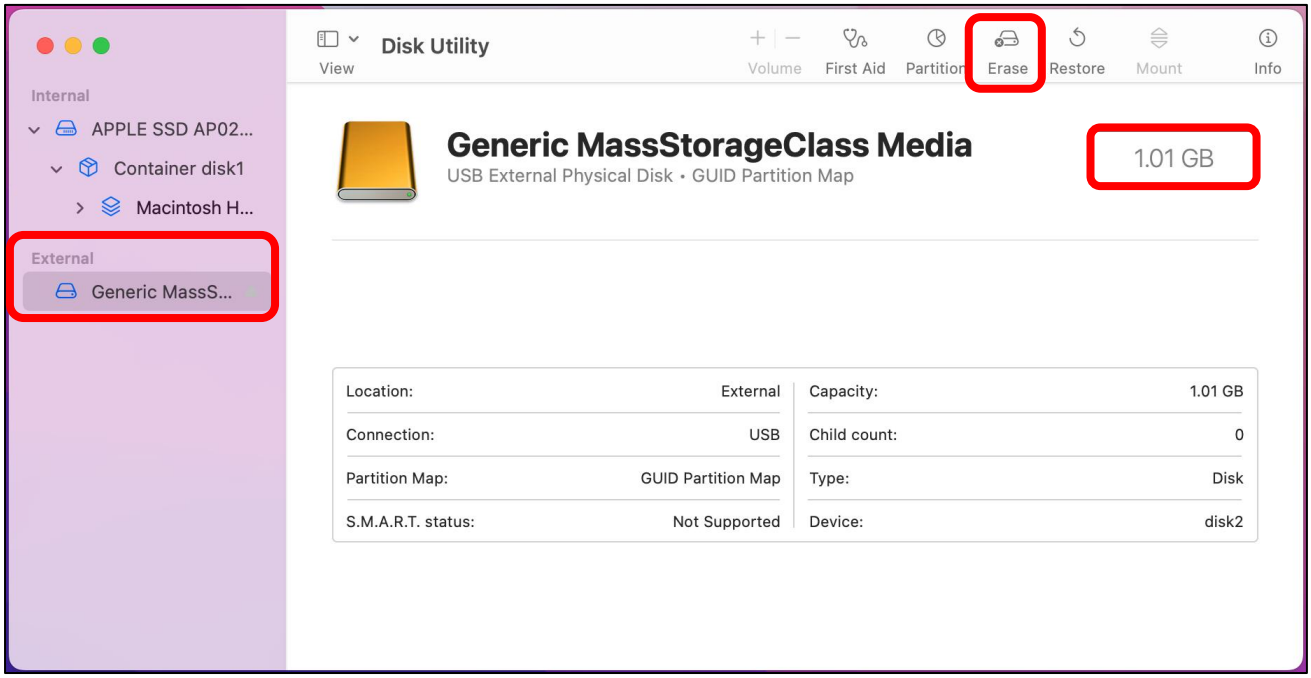
Insert the SD card into the card reader, then insert the card reader into the computer. Some computers will prompt the following information, please click to ignore it.



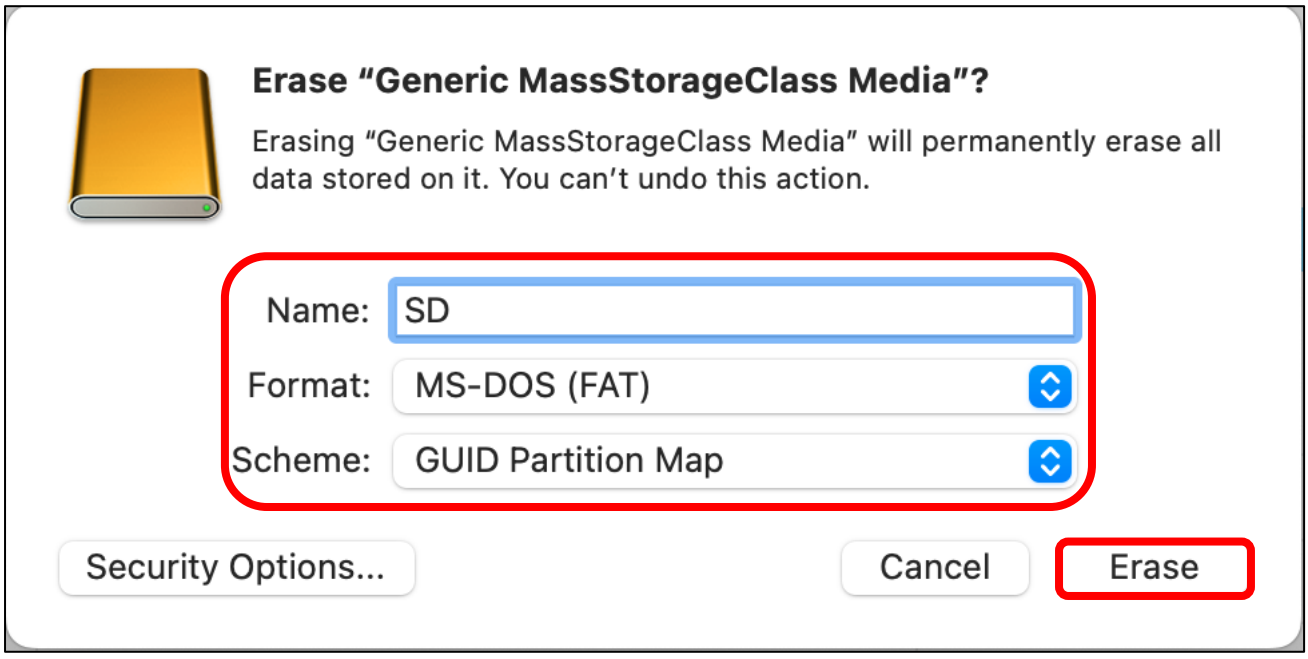
Find "Disk Utility" in the MAC system and click to open it.



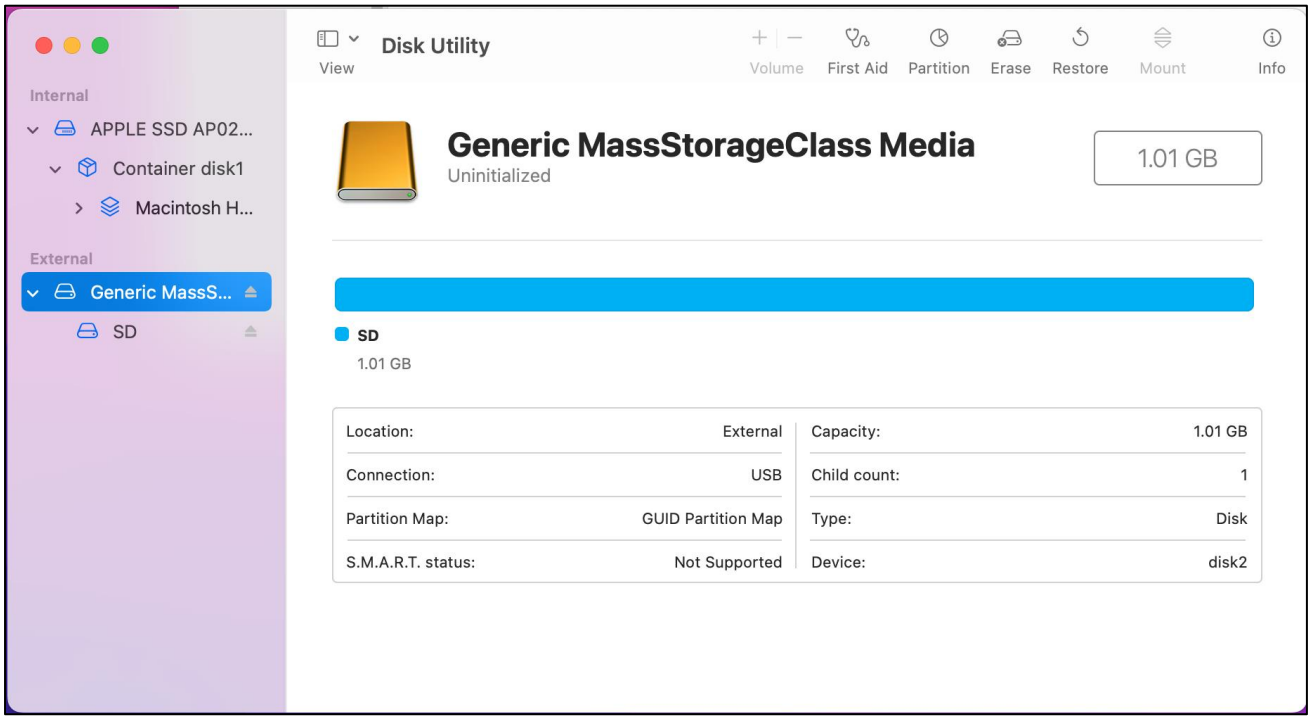
Select "Generic MassStorageClass Media", note that its size is about 1G. Please do not choose wrong item. Click "Erase".



Select the configuration as shown in the figure below, and then click "Erase".

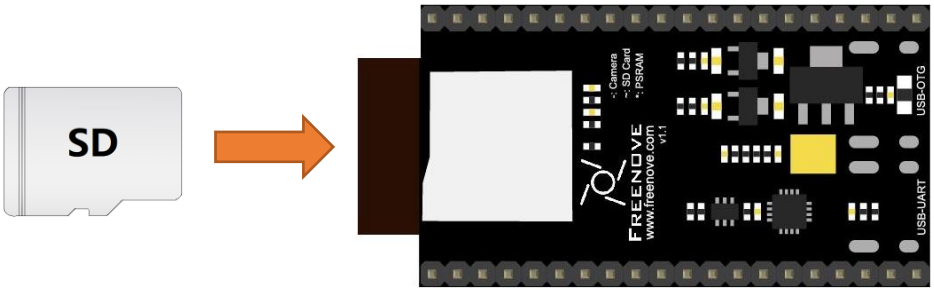


Wait for the formatting to complete. When finished, it will look like the picture below. At this point, you can see a new disk on the desktop named "SD".

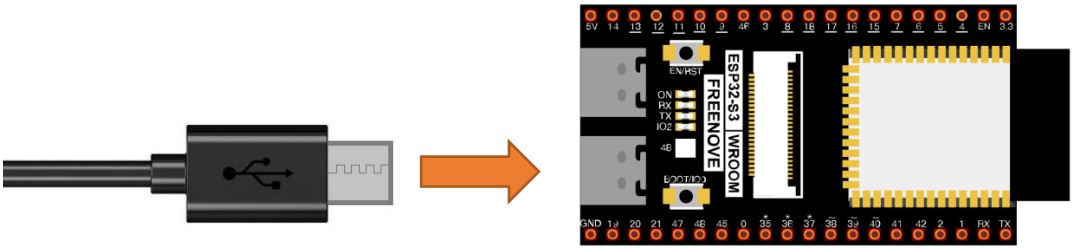


Circuit

Before connecting the USB cable, insert the SD card into the SD card slot on the back of the ESP32-S3.



Connect Freenove ESP32-S3 to the computer using the USB cable.



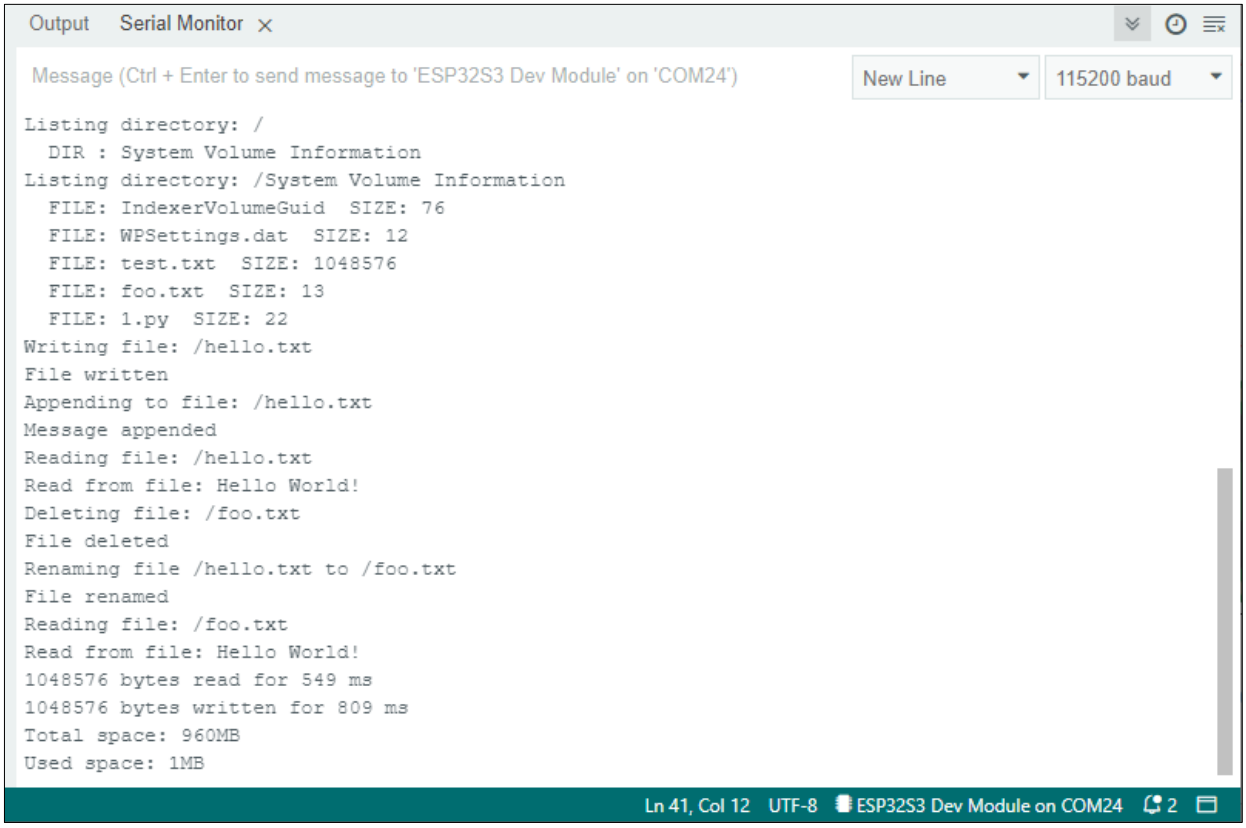
Sketch

Sketch_21.1_SDMMC_Test


```
168  void setup(){
169      Serial.begin(115200);
170      SD_MMC.setPins(SD_MMC_CLK, SD_MMC_CMD, SD_MMC_D0);
171      if (!SD_MMC.begin("/sdcard", true, true, SDMMC_FREQ_DEFAULT, 5)) {
172        Serial.println("Card Mount Failed");
173        return;
174      }
175      uint8_t cardType = SD_MMC.cardType();
176
177      if(cardType == CARD_NONE){
178          Serial.println("No SD_MMC card attached");
179          return;
180      }
181
182      Serial.print("SD_MMC Card Type: ");
183      if(cardType == CARD_MMC){
184          Serial.println("MMC");
185      } else if(cardType == CARD_SD){
186          Serial.println("SDSC");
187      } else if(cardType == CARD_SDHC){
188          Serial.println("SDHC");
189      } else {
190          Serial.println("UNKNOWN");
191      }
192
193      uint64_t cardSize = SD_MMC.cardSize() / (1024 * 1024);
194      Serial.printf("SD_MMC Card Size: %lluMB\n", cardSize);
```

Compile and upload the code to ESP32-S3-WROOM, open the serial monitor, and press the RST button on the board.

You can see the printout as shown below.



The following is the program code:

```
#include "sd_read_write.h"
#include "SD_MMC.h"

#define SD_MMC_CMD 38 //Please do not modify it.
#define SD_MMC_CLK 39 //Please do not modify it.
#define SD_MMC_D0  40 //Please do not modify it.

void setup(){
    Serial.begin(115200);
    SD_MMC.setPins(SD_MMC_CLK, SD_MMC_CMD, SD_MMC_D0);
    if (!SD_MMC.begin("/sdcard", true, true, SDMMC_FREQ_DEFAULT, 5)) {
      Serial.println("Card Mount Failed");
      return;
    }
    uint8_t cardType = SD_MMC.cardType();
    if(cardType == CARD_NONE){
        Serial.println("No SD_MMC card attached");
        return;
    }
    Serial.print("SD_MMC Card Type: ");
    if(cardType == CARD_MMC){
        Serial.println("MMC");
    } else if(cardType == CARD_SD){
        Serial.println("SDSC");
    } else if(cardType == CARD_SDHC){
        Serial.println("SDHC");
    } else {
        Serial.println("UNKNOWN");
    }

    uint64_t cardSize = SD_MMC.cardSize() / (1024 * 1024);
    Serial.printf("SD_MMC Card Size: %lluMB\n", cardSize);

    listDir(SD_MMC, "/", 0);

    createDir(SD_MMC, "/mydir");
    listDir(SD_MMC, "/", 0);

    removeDir(SD_MMC, "/mydir");
    listDir(SD_MMC, "/", 2);

    writeFile(SD_MMC, "/hello.txt", "Hello ");
    appendFile(SD_MMC, "/hello.txt", "World!\n");
```

```
44      readFile(SD_MMC, "/hello.txt");
45
46      deleteFile(SD_MMC, "/foo.txt");
47      renameFile(SD_MMC, "/hello.txt", "/foo.txt");
48      readFile(SD_MMC, "/foo.txt");
49
50      testFileIO(SD_MMC, "/test.txt");
51
52      Serial.printf("Total space: %lluMB\r\n", SD_MMC.totalBytes() / (1024 * 1024));
53      Serial.printf("Used space: %lluMB\r\n", SD_MMC.usedBytes() / (1024 * 1024));
54  }
55
56  void loop(){
57    delay(10000);
58  }
```

Add the SD card drive header file.

```
1  #include "sd_read_write.h"
2  #include "SD_MMC.h"
```

Defines the drive pins of the SD card. Please do not modify it. Because these pins are fixed.

```
4  #define SD_MMC_CMD 38 //Please do not modify it.
5  #define SD_MMC_CLK 39 //Please do not modify it.
6  #define SD_MMC_D0  40 //Please do not modify it.
```

Initialize the serial port function. Sets the drive pin for SDMMC one-bit bus mode.

```
9       Serial.begin(115200);
10      SD_MMC.setPins(SD_MMC_CLK, SD_MMC_CMD, SD_MMC_D0);
```

Set the mount point of the SD card, set SDMMC to one-bit bus mode, and set the read and write speed to 20MHz.

```
11      if (!SD_MMC.begin("/sdcard", true, true, SDMMC_FREQ_DEFAULT, 5)) {
12        Serial.println("Card Mount Failed");
13        return;
14      }
```

Get the type of SD card and print it out through the serial port.

```
15      uint8_t cardType = SD_MMC.cardType();
16      if(cardType == CARD_NONE){
17          Serial.println("No SD_MMC card attached");
18          return;
19      }
20      Serial.print("SD_MMC Card Type: ");
21      if(cardType == CARD_MMC){
22          Serial.println("MMC");
23      } else if(cardType == CARD_SD){
24          Serial.println("SDSC");
25      } else if(cardType == CARD_SDHC){
26          Serial.println("SDHC");
```

```
27      } else {
28          Serial.println("UNKNOWN");
29      }
```

Call the listDir() function to read the folder and file names in the SD card, and print them out through the serial port. This function can be found in "sd_read_write.cpp".

```
34  listDir(SD_MMC, "/", 0);
```

Call createDir() to create a folder, and call removeDir() to delete a folder.

```
36  createDir(SD_MMC, "/mydir");
39  removeDir(SD_MMC, "/mydir");
```

Call writeFile() to write any content to the txt file. If there is no such file, create this file first.

Call appendFile() to append any content to txt.

Call readFile() to read the content in txt and print it via the serial port.

```
42  writeFile(SD_MMC, "/hello.txt", "Hello ");
43  appendFile(SD_MMC, "/hello.txt", "World!\n");
44  readFile(SD_MMC, "/hello.txt");
```

Call deleteFile() to delete a specified file.

Call renameFile() to copy a file and rename it.

```
46      deleteFile(SD_MMC, "/foo.txt");
47      renameFile(SD_MMC, "/hello.txt", "/foo.txt");
```

Call the testFileIO() function to test the time it takes to read 512 bytes and the time it takes to write 2048*512 bytes of data.

```
50  testFileIO(SD_MMC, "/test.txt");
```

Print the total size and used size of the SD card via the serial port.

```
52      Serial.printf("Total space: %lluMB\r\n", SD_MMC.totalBytes() / (1024 * 1024));
53      Serial.printf("Used space: %lluMB\r\n", SD_MMC.usedBytes() / (1024 * 1024));
```

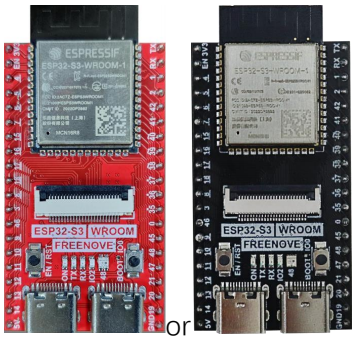
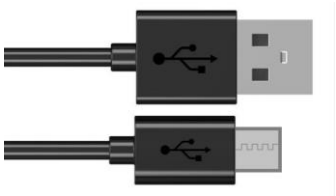
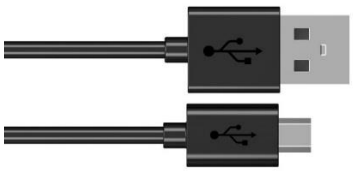
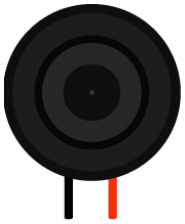
Chapter 22 Play SD card music

In the previous study, we have learned how to use the SD card, and then we will learn to play the music in the SD card.

Project 22.1 SDMMC Music

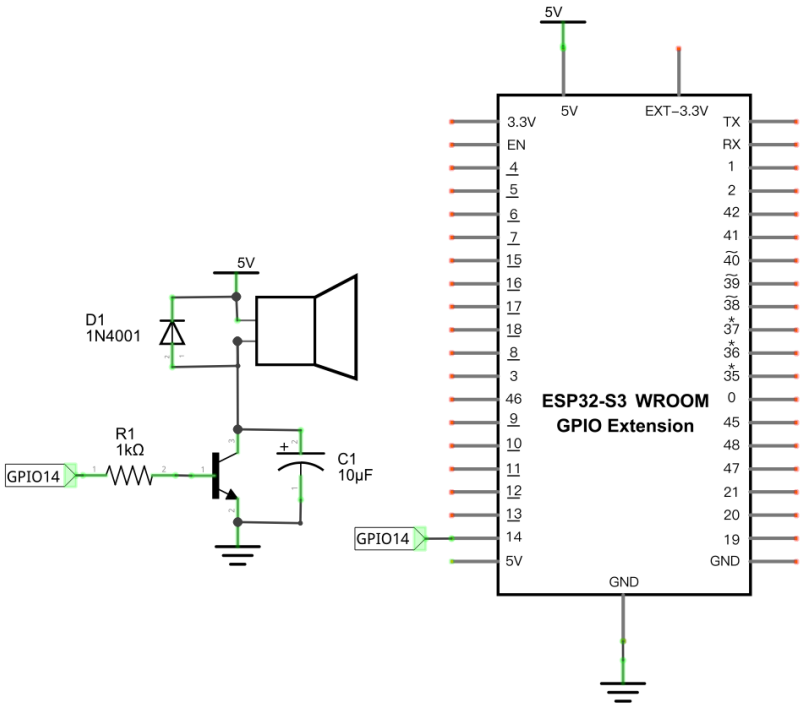
In this project, we will read an mp3 file from an SD card, decode it through ESP32-S3, and use a speaker to play it.

Component List

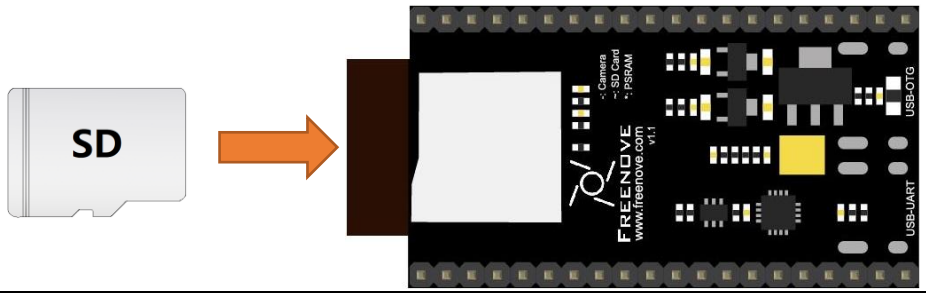
ESP32-S3(N16R8)/ESP32-S3(N8R8)  or	USB cable x1 	SDcard x1 
Micro USB Wire x1 	NPN transistorx1 (S8050) 	Speaker 
Diode x1 	Resistor 1kΩ x1 	Capacitor 10uF x1 
Jumper F/M x4 Jumper F/F x2 	SDcard reader x1 (random color)  (Not a USB flash drive.)	

Circuit

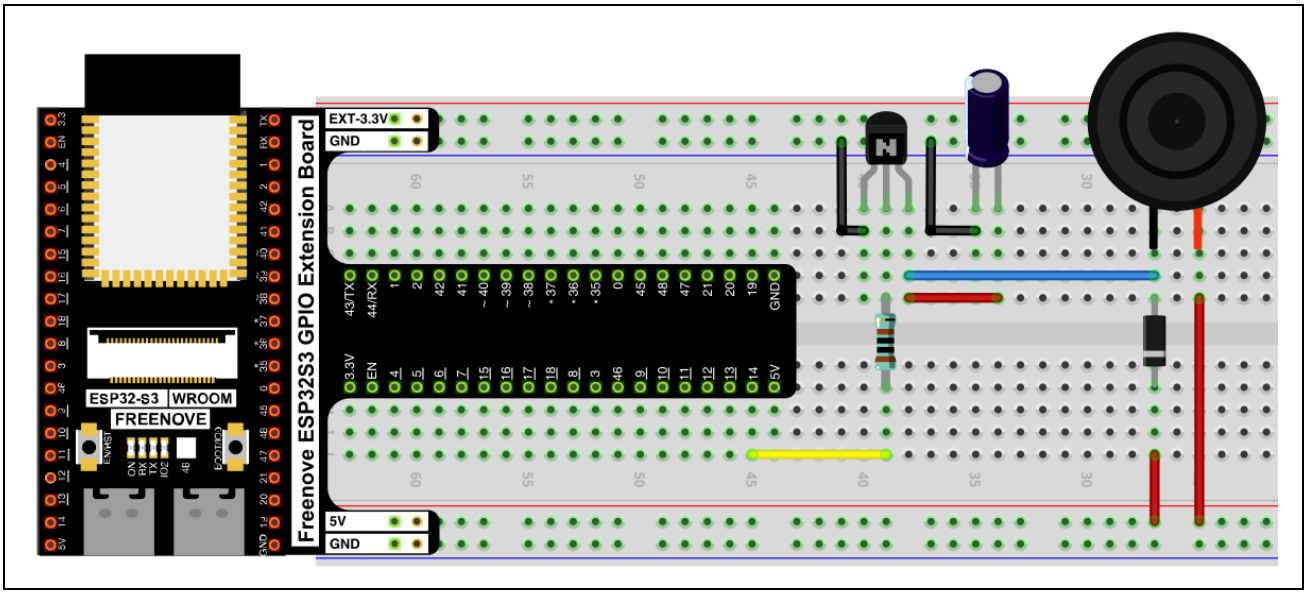
Schematic diagram



Please note that before connecting the USB cable, please put the music into the SD card and insert the SD card into the card slot on the back of the ESP32-S3.



Hardware connection. If you need any support, please feel free to contact us via: support@freenove.com

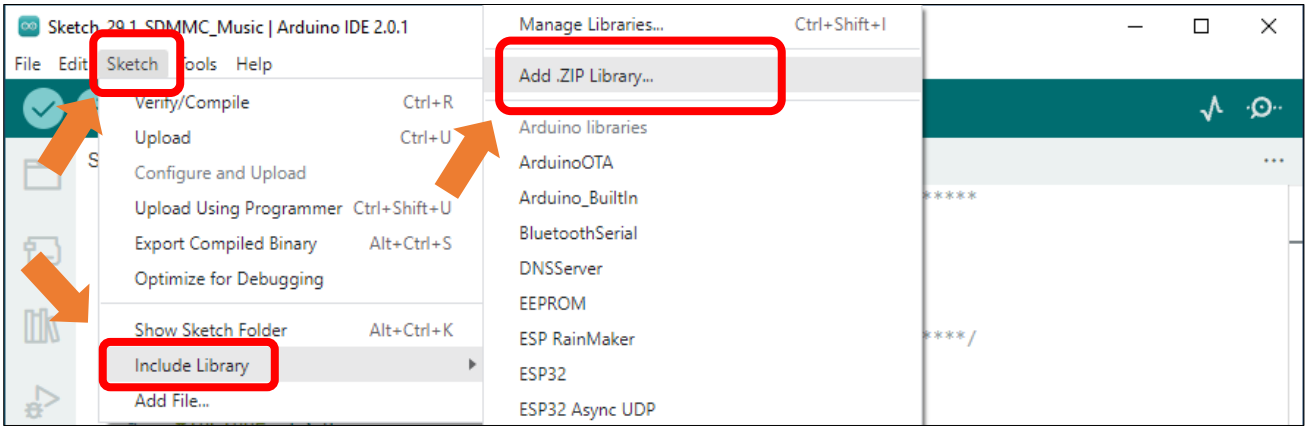


Sketch

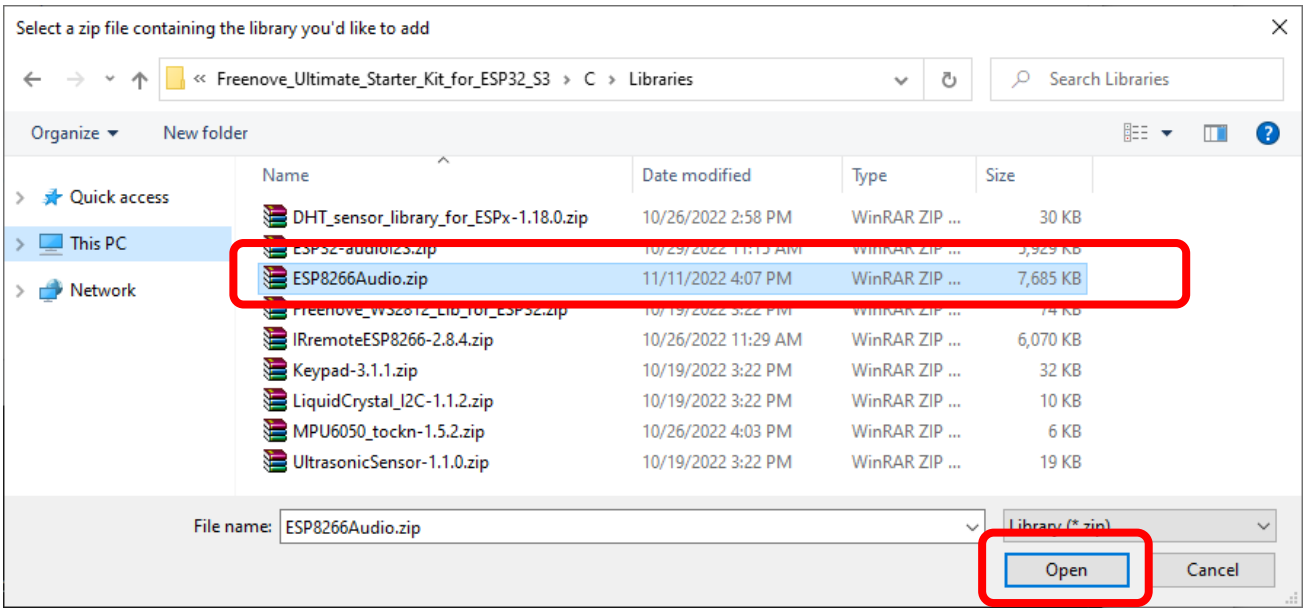
How to install the library

In this project, we will use the ESP8266Audio.zip library to decode the audio files in the SD card, and then output the audio signal through GPIO. If you have not installed this library, please follow the steps below to install it.

Open arduino->Sketch->Include library-> Add .ZIP Library.



In the new pop-up window, select "**Freenove_Super_Starter_Kit_for_ESP32_S3\C\Libraries\ESP8266Audio.zip**". Then click "Open".

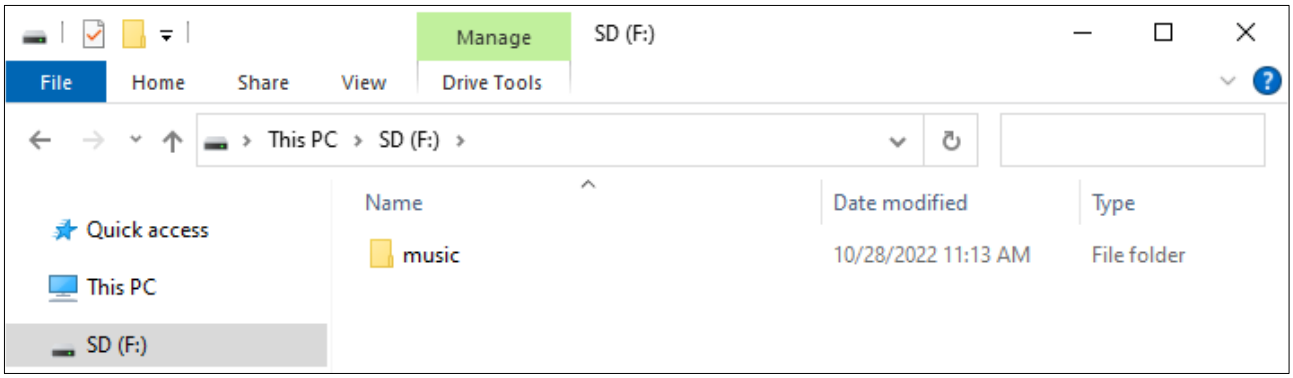


Sketch_22.1_PlayMP3FromSD

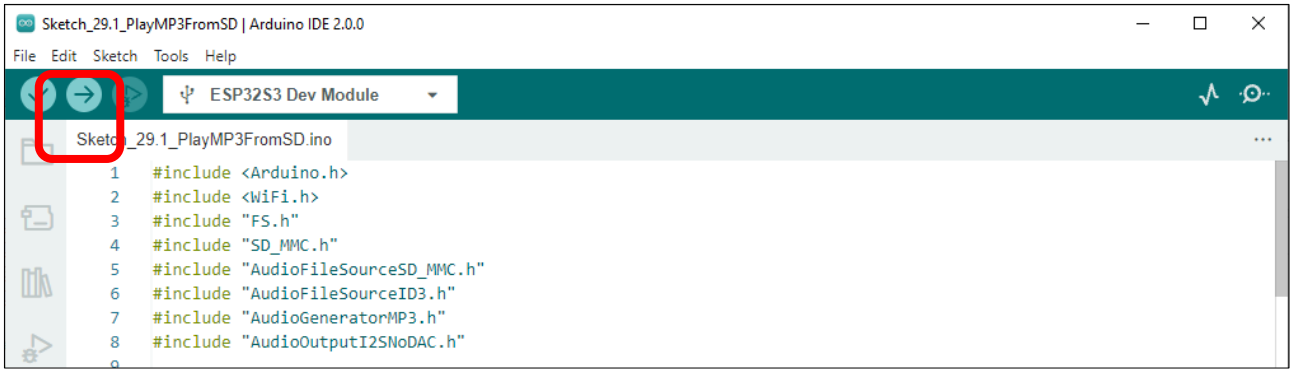
We placed a folder called "music" in:

Freenove_Super_Starter_Kit_for_ESP32_S3\Sketches\Sketch_22.1_PlayMP3FromSD

User needs to copy this folder to SD card.



Click upload.



Compile and upload the code to the ESP32-S3 WROOM and open the serial monitor. ESP32-S3 takes a few seconds to initialize the program. When you see the message below, it means that ESP32-S3 has started parsing the mp3 in sd and started playing music through Pin.

```
Message (Ctrl + Enter to send message to 'ESP32S3 Dev Module'

SPIWP:0xee
mode:DIO, clock div:1
load:0x3fce3808,len:0x43c
load:0x403c9700,len:0xbec
load:0x403cc700,len:0x2a3c
entry 0x403c98d8
Sample MP3 playback begins...
+0 0x3fce2c34
ID3 callback for: Year = ''
ID3 callback for: eof = 'id3'
```

The following is the program code:

```
#include <Arduino.h>
#include <WiFi.h>
#include "FS.h"
#include "SD_MMC.h"
#include "AudioFileSourceSD_MMC.h"
#include "AudioFileSourceID3.h"
#include "AudioGeneratorMP3.h"
#include "AudioOutputI2SNoDAC.h"

#define SD_MMC_CMD 38 //Please do not modify it.
#define SD_MMC_CLK 39 //Please do not modify it.
#define SD_MMC_D0  40 //Please do not modify it.

AudioGeneratorMP3 *mp3;
AudioFileSourceID3 *id3;
AudioOutputI2SNoDAC *out;
AudioFileSourceSD_MMC *file = NULL;

// Called when a metadata event occurs (i.e. an ID3 tag, an ICY block, etc.
void MDCallback(void *cbData, const char *type, bool isUnicode, const char *string)
{
  (void)cbData;
  Serial.printf("ID3 callback for: %s = '", type);

  if (isUnicode) {
    string += 2;
  }

  while (*string) {
    char a = *(string++);
    if (isUnicode) {
      string++;
    }
    Serial.printf("%c", a);
```

```
35    }
36    Serial.printf("'\n");
37    Serial.flush();
38  }
39
40  void setup()
41  {
42    WiFi.mode(WIFI_OFF);
43    Serial.begin(115200);
44    delay(1000);
45    SD_MMC.setPins(SD_MMC_CLK, SD_MMC_CMD, SD_MMC_D0);
46    if (!SD_MMC.begin("/sdcard", true, true, SDMMC_FREQ_DEFAULT, 5)) {
47        Serial.println("Card Mount Failed");
48        return;
49    }
50    Serial.printf("Sample MP3 playback begins...\n");
51
52    audioLogger = &Serial;
53    file = new AudioFileSourceSD_MMC("/music/01.mp3");
54    id3 = new AudioFileSourceID3(file);
55    id3->RegisterMetadataCB(MDCallback, (void*)"ID3TAG");
56    out = new AudioOutputI2SNoDAC();
57    out->SetPinout(12,13,14);//Set the audio output pin, Only 14 were used
58    out->SetGain(0.3);//Setting the Volume
59    mp3 = new AudioGeneratorMP3();
60    mp3->begin(id3, out);
61  }
62
63  void loop()
64  {
65    if (mp3->isRunning()) {
66      if (!mp3->loop()) mp3->stop();
67    } else {
68      Serial.printf("MP3 done\n");
69      delay(1000);
70    }
71  }
```

Add music decoding header files and SD card drive files.

```
#include <Arduino.h>
#include <WiFi.h>
#include "FS.h"
#include "SD_MMC.h"
#include "AudioFileSourceSD_MMC.h"
#include "AudioFileSourceID3.h"
#include "AudioGeneratorMP3.h"
#include "AudioOutputI2SNoDAC.h"
```

Define the drive pins for SD card. Note that the SD card driver pins cannot be modified.

```
#define SD_MMC_CMD 38 //Please do not modify it.
#define SD_MMC_CLK 39 //Please do not modify it.
#define SD_MMC_D0  40 //Please do not modify it.
```

Apply for audio decoding class object.

```
AudioGeneratorMP3 *mp3;
AudioFileSourceID3 *id3;
AudioOutputI2SNoDAC *out;
AudioFileSourceSD_MMC *file = NULL;
```

Set the audio file source and associate it with the decoder. Initialize the audio output pin and set the volume to 2.

```
  audioLogger = &Serial;
  file = new AudioFileSourceSD_MMC("/music/01.mp3");
  id3 = new AudioFileSourceID3(file);
  id3->RegisterMetadataCB(MDCallback, (void*)"ID3TAG");
  out = new AudioOutputI2SNoDAC();
  out->SetPinout(12,13,14);//Set the audio output pin, Only 14 were used
  out->SetGain(2);//Setting the Volume(0-3.9)
  mp3 = new AudioGeneratorMP3();
  mp3->begin(id3, out);
```

Determine whether the mp3 player is finished. If it is playing, continue playing. If it is finished, print a message.

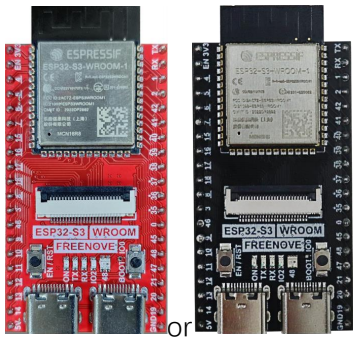
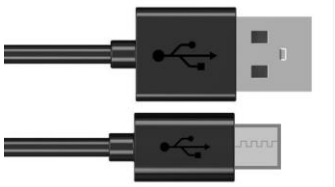
```
  if (mp3->isRunning()) {
    if (!mp3->loop()) mp3->stop();
  } else {
    Serial.printf("MP3 done\n");
    delay(1000);
  }
```

Chapter 23 WiFi Working Modes

In this chapter, we'll focus on the WiFi infrastructure for ESP32-S3 WROOM.
ESP32-S3 WROOM has 3 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 WiFi, otherwise WiFi cannot be used.

Project 23.1 Station mode

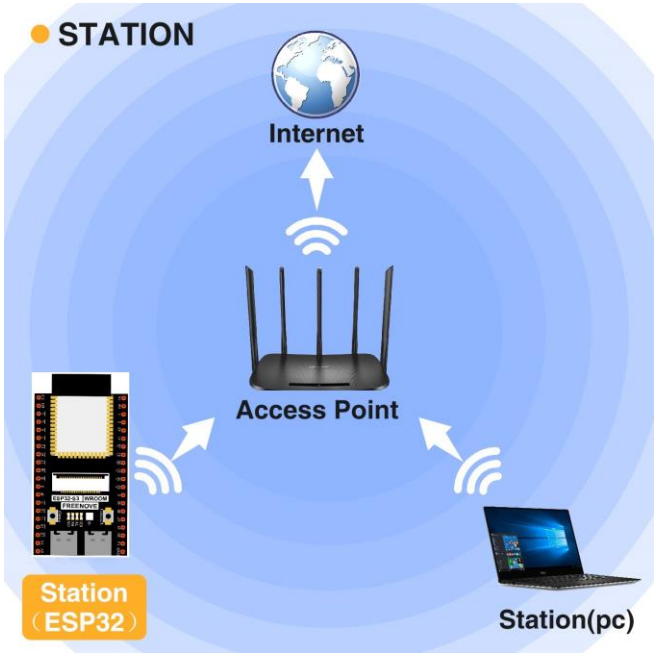
Component List

ESP32-S3(N16R8)/ESP32-S3(N8R8)  or	USB cable x1 

Component knowledge

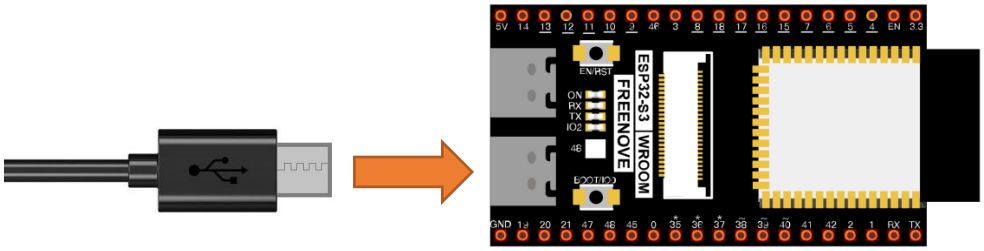
Station mode

When ESP32-S3 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-S3 wants to communicate with the PC, it needs to be connected to the router.



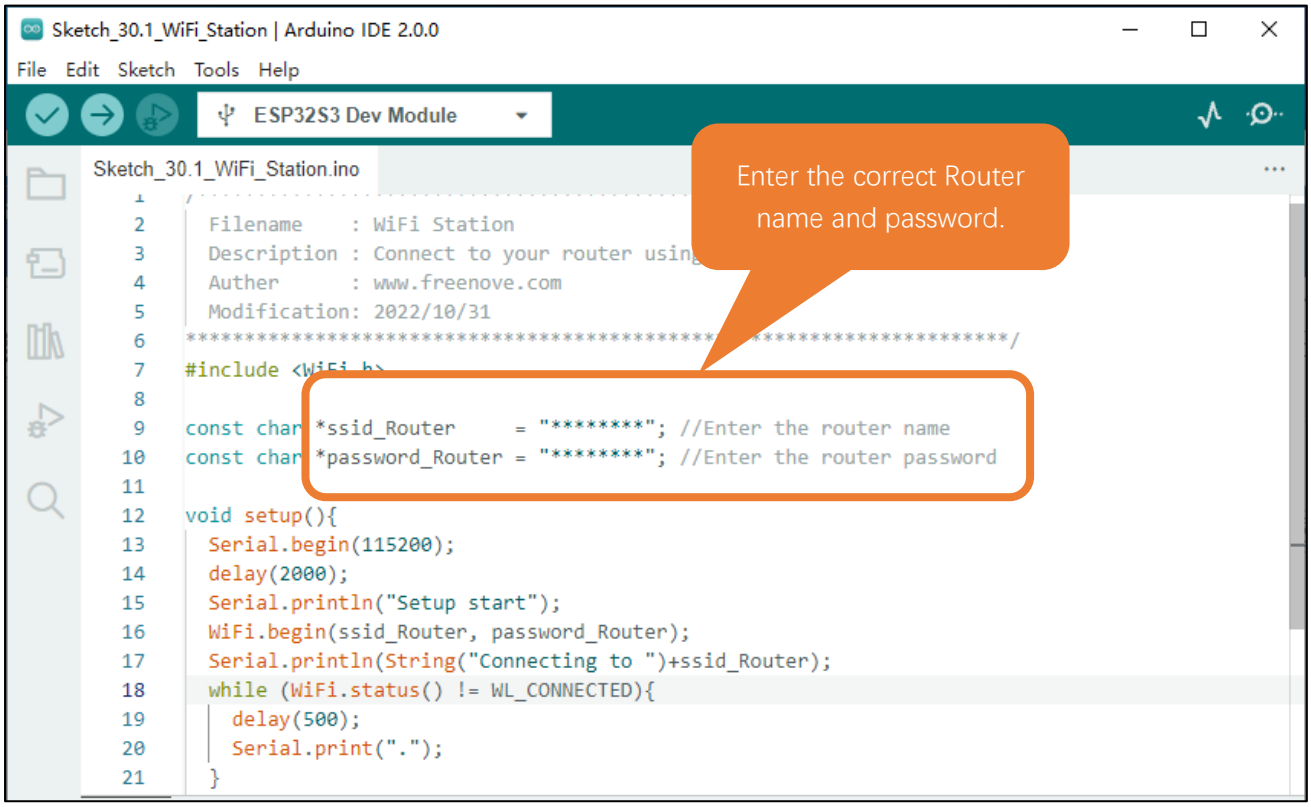
Circuit

Connect Freenove ESP32-S3 to the computer using the USB cable.

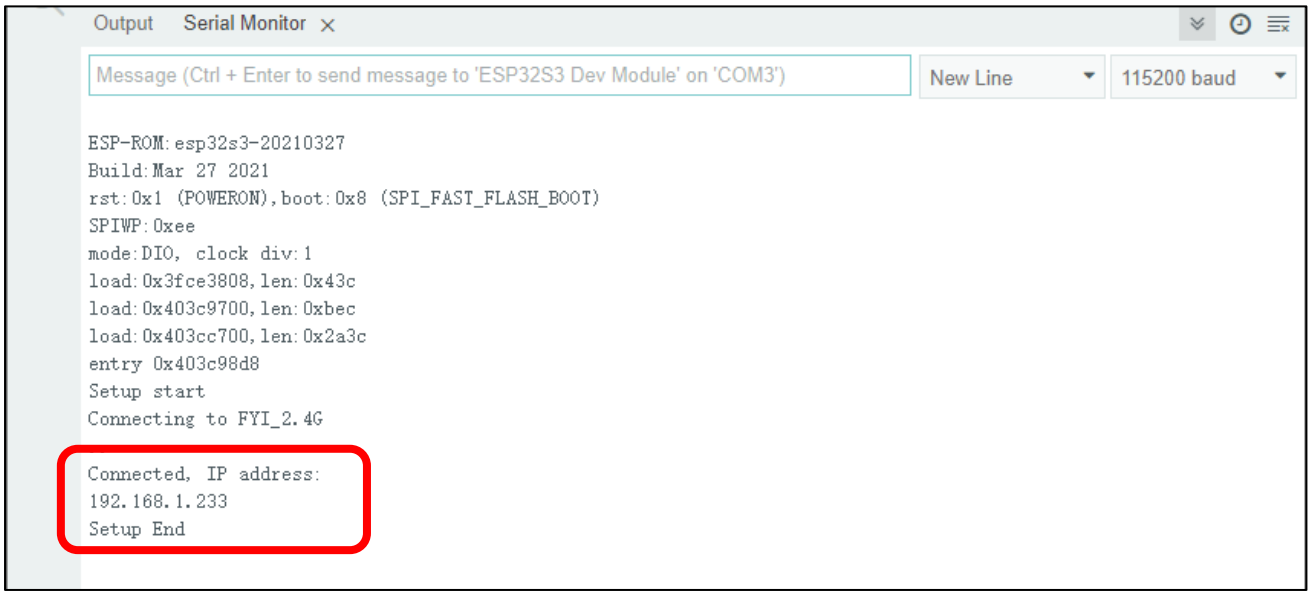


Sketch

Sketch_23.1_Station_mode



Because the names and passwords of routers in various places are different, before the Sketch 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, compile and upload codes to ESP32-S3 WROOM, open serial monitor and set baud rate to 115200. And then it will display as follows:



When ESP32-S3 WROOM successfully connects to "ssid_Router", serial monitor will print out the IP address assigned to ESP32-S3 WROOM by the router.

The following is the program code:

```
#include <WiFi.h>

const char *ssid_Router     = "********"; //Enter the router name
const char *password_Router = "********"; //Enter the router password

void setup(){
  Serial.begin(115200);
  delay(2000);
  Serial.println("Setup start");
  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() {
}
```

Include the WiFi Library header file of ESP32-S3.

```
#include <WiFi.h>
```

Enter correct router name and password.

```
const char *ssid_Router     = "********"; //Enter the router name
const char *password_Router = "********"; //Enter the router password
```

Set ESP32-S3 in Station mode and connect it to your router.

```
  WiFi.begin(ssid_Router, password_Router);
```

Check whether ESP32-S3 has connected to router successfully every 0.5s.

```
  while (WiFi.status() != WL_CONNECTED){
    delay(500);
    Serial.print(".");
  }
```

Serial monitor prints out the IP address assigned to ESP32-S3 WROOM

```
  Serial.println(WiFi.localIP());
```

Reference

Class Station

Every time when using WiFi, you need to include header file "WiFi.h.".

begin(ssid, password,channel, bssid, connect): ESP32-S3 is used as Station to connect hotspot.

ssid: WiFi hotspot name

password: WiFi hotspot password
channel: WiFi hotspot channel number; communicating through specified channel; optional parameter
bssid: mac address of WiFi hotspot, optional parameter
connect: blloean optional parameter, defaulting to true. If set as false, then ESP32-S3 won't connect WiFi.
config(local_ip, gateway, subnet, dns1, dns2): set static local IP address.
    local_ip: station fixed IP address.
    subnet: subnet mask
    dns1,dns2: optional parameter. define IP address of domain name server
status: obtain the connection status of WiFI
local IP(): obtian IP address in Station mode
disconnect(): disconnect wifi
setAutoConnect(boolen): set automatic connection Every time ESP32-S3 is power on, it will connect WiFi aitomatically.
setAutoReconnect(boolen): set automatic reconnection Every time ESP32-S3 disconnects WiFi, it will reconnect to WiFi automatically.

Project 23.2 AP mode

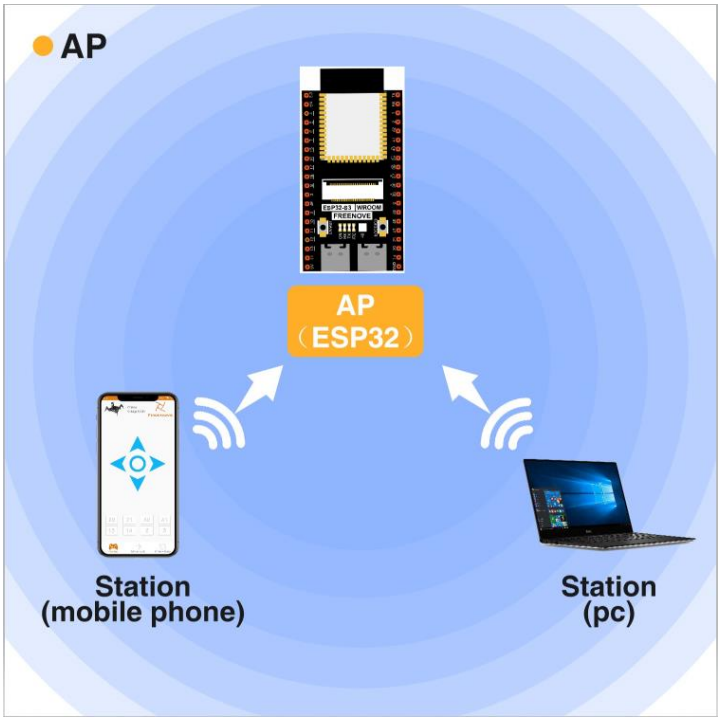
Component List & Circuit

Component List & Circuit are the same as in Project 23.1.

Component knowledge

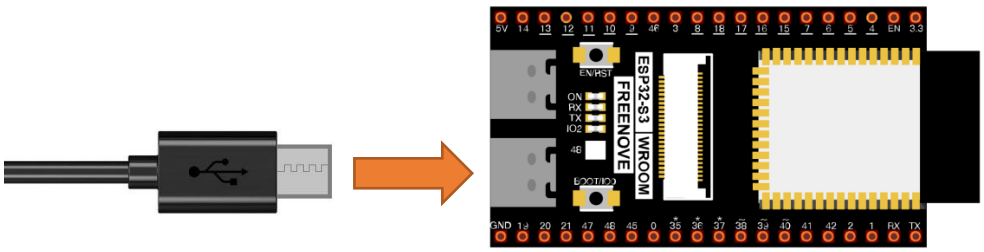
AP mode

When ESP32-S3 selects AP mode, it creates a hotspot network that is separate from the Internet and waits for other WiFi devices to connect. As shown in the figure below, ESP32-S3 is used as a hotspot. If a mobile phone or PC wants to communicate with ESP32-S3, it must be connected to the hotspot of ESP32-S3. Only after a connection is established with ESP32-S3 can they communicate.

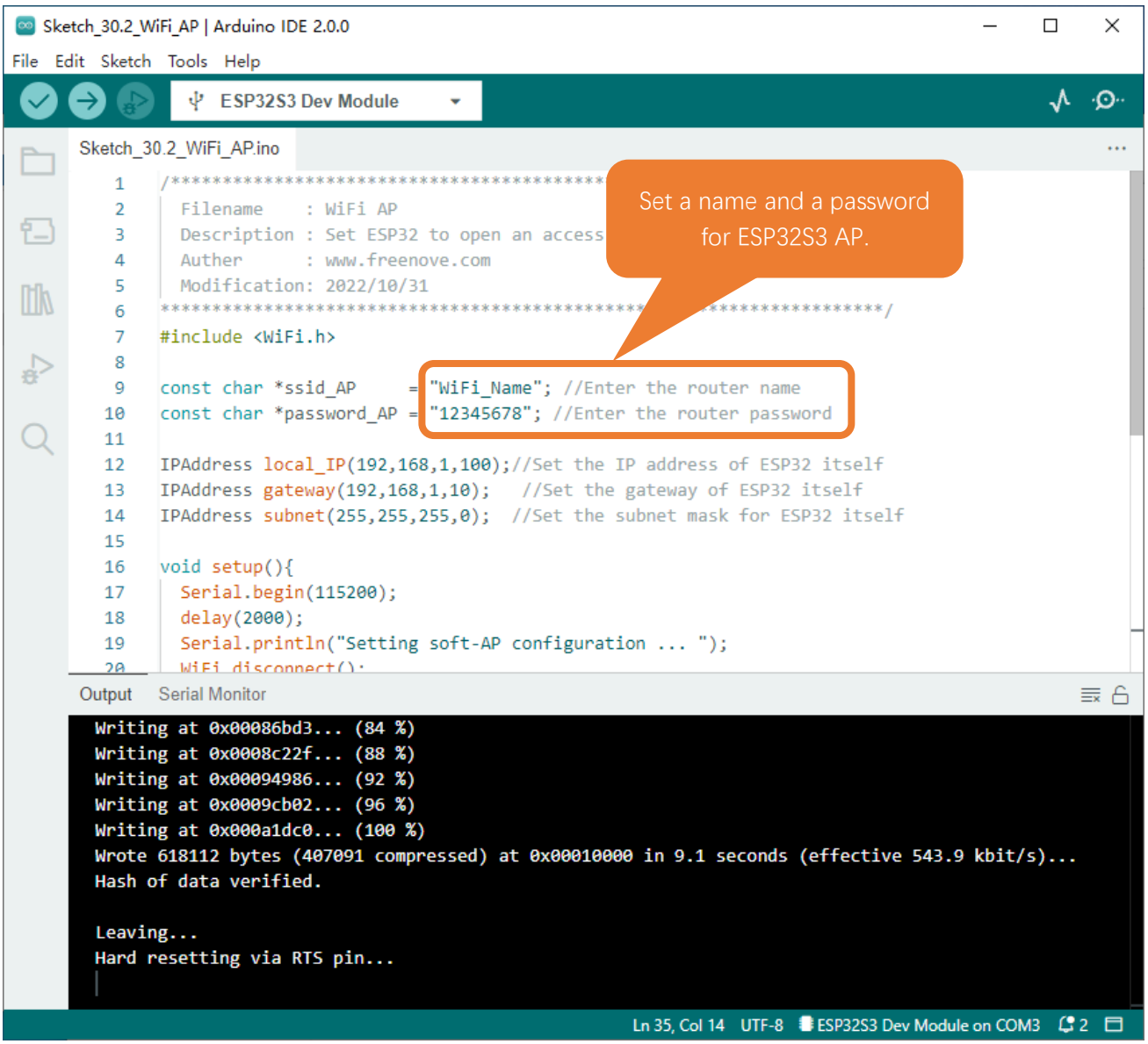


Circuit

Connect Freenove ESP32-S3 to the computer using the USB cable.



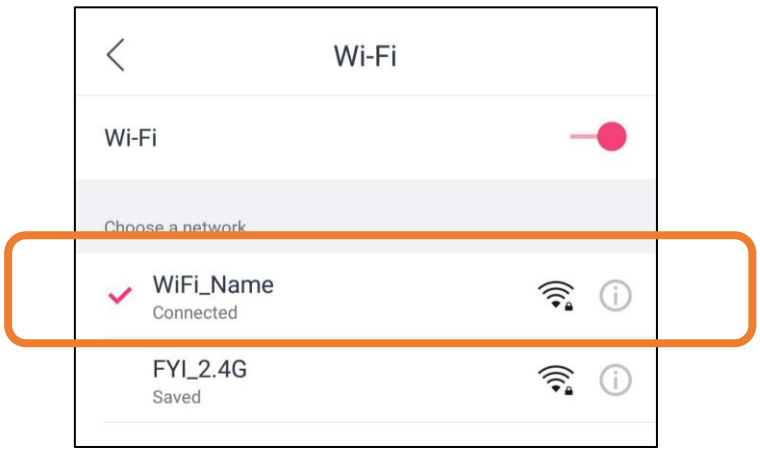
Sketch



Before the Sketch runs, you can make any changes to the AP name and password for ESP32-S3 in the box as shown in the illustration above. Of course, you can leave it alone by default.

Compile and upload codes to ESP32-S3 WROOM, open the serial monitor and set the baud rate to 115200. And then it will display as follows.

When observing the print information of the serial monitor, turn on the WiFi scanning function of your phone, and you can see the ssid_AP on ESP32-S3, which is called "WiFi_Name" in this Sketch. You can enter the password "12345678" to connect it or change its AP name and password by modifying Sketch.



Sketch_23.2_AP_mode

The following is the program code:

```
#include <WiFi.h>

const char *ssid_AP     = "WiFi_Name"; //Enter the router name
const char *password_AP = "12345678"; //Enter the router password

IPAddress local_IP(192,168,1,100);//Set the IP address of ESP32-S3 itself
IPAddress gateway(192,168,1,10);   //Set the gateway of ESP32-S3 itself
IPAddress subnet(255,255,255,0);  //Set the subnet mask for ESP32-S3 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!");
  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() {
}
```

Include WiFi Library header file of ESP32-S3.

```
#include <WiFi.h>
```

Enter correct AP name and password.

```
const char *ssid_AP     = "WiFi_Name"; //Enter the router name
const char *password_AP = "12345678"; //Enter the router password
```

Set ESP32-S3 in AP mode.

```
WiFi.mode(WIFI_AP);
```

Configure IP address, gateway and subnet mask for ESP32-S3.

```
WiFi.softAPConfig(local_IP, gateway, subnet)
```

Turn on an AP in ESP32-S3, whose name is set by ssid_AP and password is set by password_AP.

```
WiFi.softAP(ssid_AP, password_AP);
```

Check whether the AP is turned on successfully. If yes, print out IP and MAC address of AP established by ESP32-S3. If no, print out the failure prompt.

```
19  if(result){
20      Serial.println("Ready");
21      Serial.println(String("Soft-AP IP address = ") + WiFi.softAPIP().toString());
22      Serial.println(String("MAC address = ") + WiFi.softAPmacAddress().c_str());
23    }else{
24      Serial.println("Failed!");
25    }
26    Serial.println("Setup End");
```

Reference

Class AP

Every time when using WiFi, you need to include header file "WiFi.h.".

softAP(ssid, password, channel, ssid_hidden, max_connection):

ssid: WiFi hotspot name

password: WiFi hotspot password

channel: Number of WiFi connection channels, range 1-13. The default is 1.

ssid_hidden: Whether to hide WiFi name from scanning by other devices. The default is not hide.

max_connection: Maximum number of WiFi connected devices. The range is 1-4. The default is 4.

softAPConfig(local_ip, gateway, subnet): set static local IP address.

local_ip: station fixed IP address.

Gateway: gateway IP address

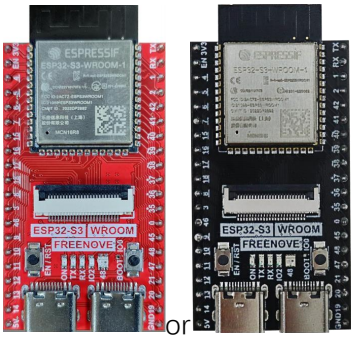
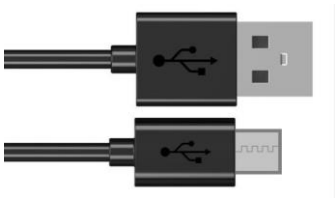
subnet: subnet mask

softAP(): obtian IP address in AP mode

softAPdisconnect (): disconnect AP mode.

Project 23.3 AP+Station mode

Component List

ESP32-S3(N16R8)/ESP32-S3(N8R8) 	USB cable x1 

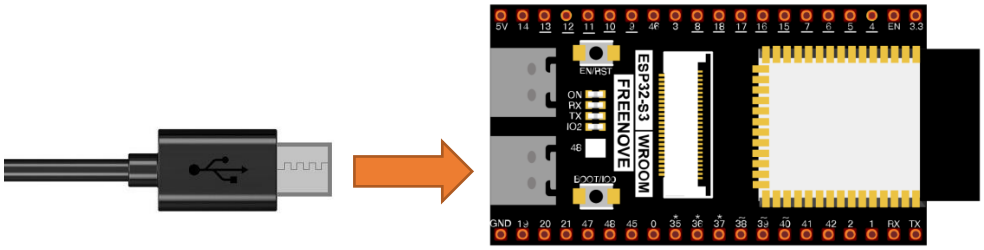
Component knowledge

AP+Station mode

In addition to AP mode and station mode, ESP32-S3 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-S3'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-S3.

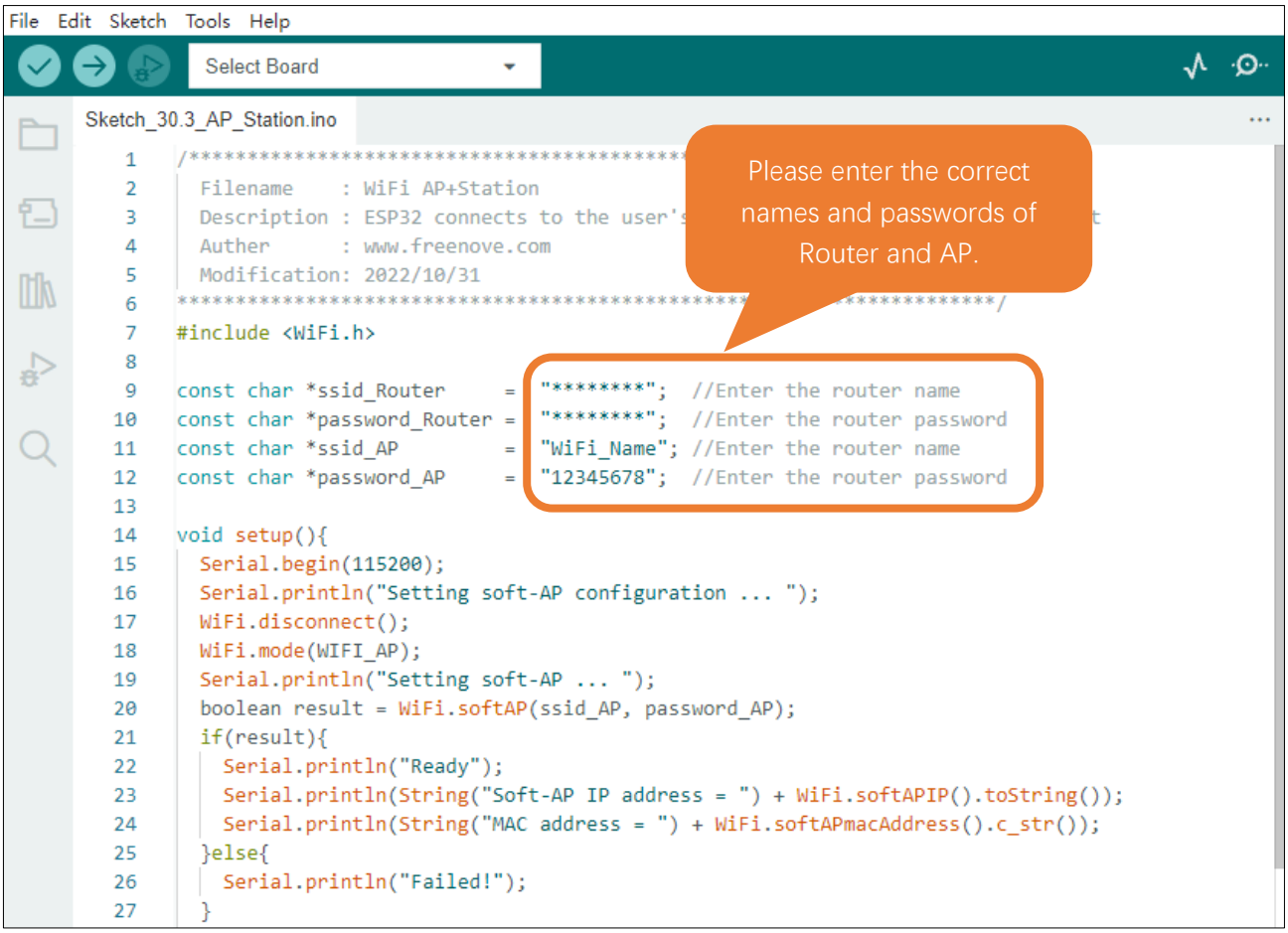
Circuit

Connect Freenove ESP32-S3 to the computer using the USB cable.



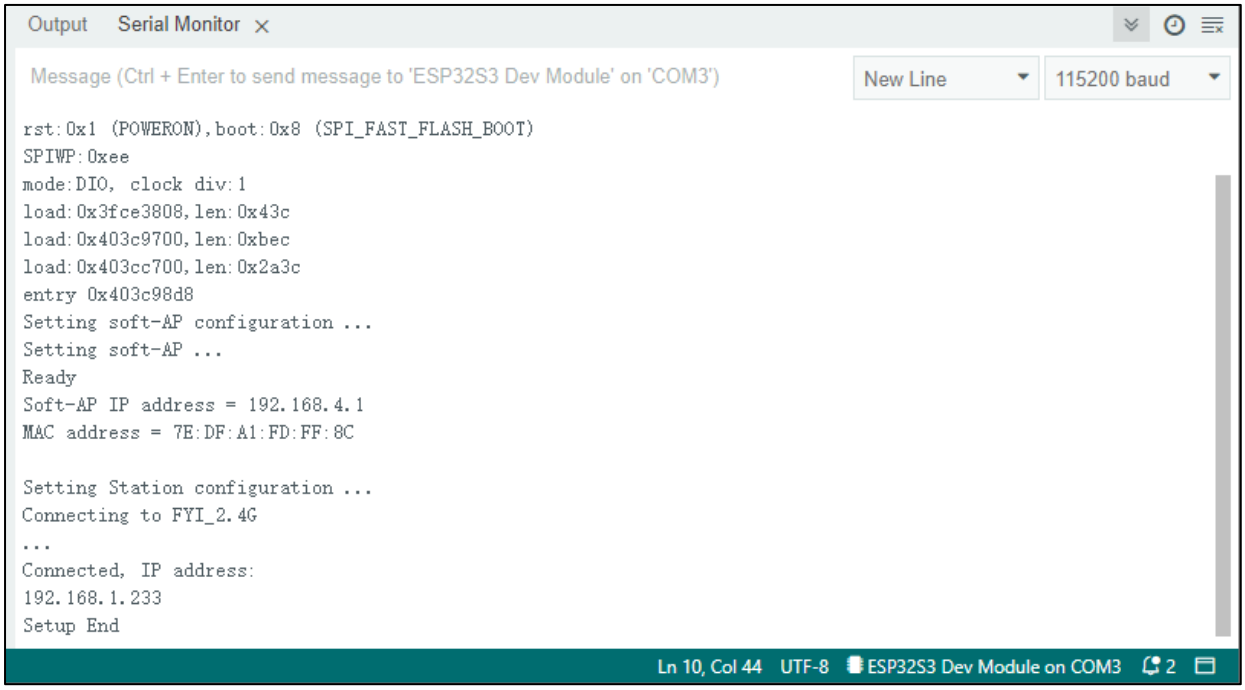
Sketch

Sketch_23.3_AP_Station_mode

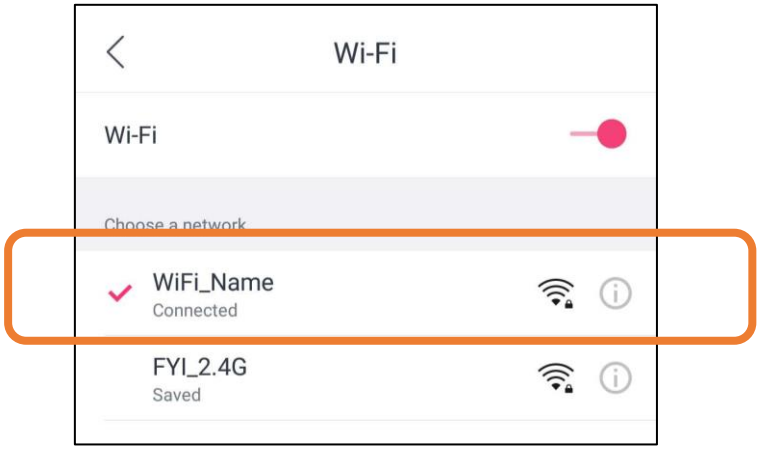


It is analogous to Project 23.1 and Project 23.2. Before running the Sketch, you need to modify ssid_Router, password_Router, ssid_AP and password_AP shown in the box of the illustration above.

After making sure that Sketch is modified correctly, compile and upload codes to ESP32-S3 WROOM, open serial monitor and set baud rate to 115200. And then it will display as follows:



When observing the print information of the serial monitor, turn on the WiFi scanning function of your phone, and you can see the ssid_AP on ESP32-S3.



The following is the program code:

```
#include <WiFi.h>

const char *ssid_Router     = "********";  //Enter the router name
const char *password_Router = "********";  //Enter the router password
const char *ssid_AP         = "WiFi_Name"; //Enter the AP name
const char *password_AP     = "12345678";  //Enter the AP password

void setup(){
  Serial.begin(115200);
  Serial.println("Setting soft-AP configuration ... ");
```

```
11    WiFi.disconnect();
12    WiFi.mode(WIFI_AP);
13    Serial.println("Setting soft-AP ... ");
14    boolean result = WiFi.softAP(ssid_AP, password_AP);
15    if(result){
16      Serial.println("Ready");
17      Serial.println(String("Soft-AP IP address = ") + WiFi.softAPIP().toString());
18      Serial.println(String("MAC address = ") + WiFi.softAPmacAddress().c_str());
19    }else{
20      Serial.println("Failed!");
21    }
22
23    Serial.println("\nSetting Station configuration ... ");
24    WiFi.begin(ssid_Router, password_Router);
25    Serial.println(String("Connecting to ")+ ssid_Router);
26    while (WiFi.status() != WL_CONNECTED){
27      delay(500);
28      Serial.print(".");
29    }
30    Serial.println("\nConnected, IP address: ");
31    Serial.println(WiFi.localIP());
32    Serial.println("Setup End");
33  }
34
35  void loop() {
36  }
```

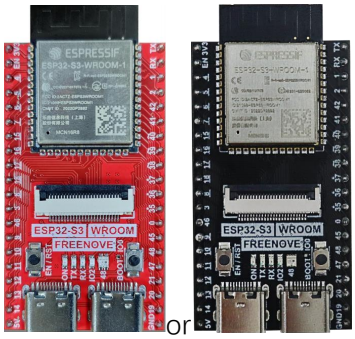
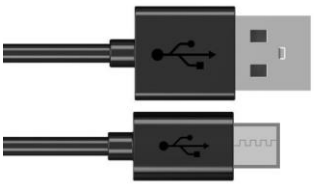
Chapter 24 TCP/IP

In this chapter, we will introduce how ESP32-S3 implements network communications based on TCP/IP protocol. There are two roles in TCP/IP communication, namely Server and Client, which will be implemented respectively with two projects in this chapter.

Project 24.1 As Client

In this section, ESP32-S3 is used as Client to connect Server on the same LAN and communicate with it.

Component List

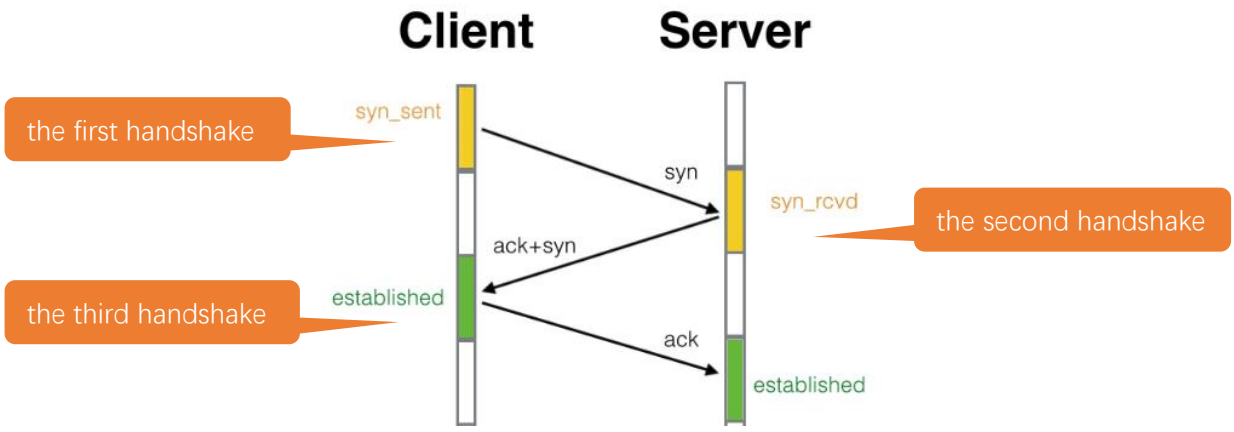
ESP32-S3(N16R8)/ESP32-S3(N8R8) 	USB cable x1 

Component knowledge

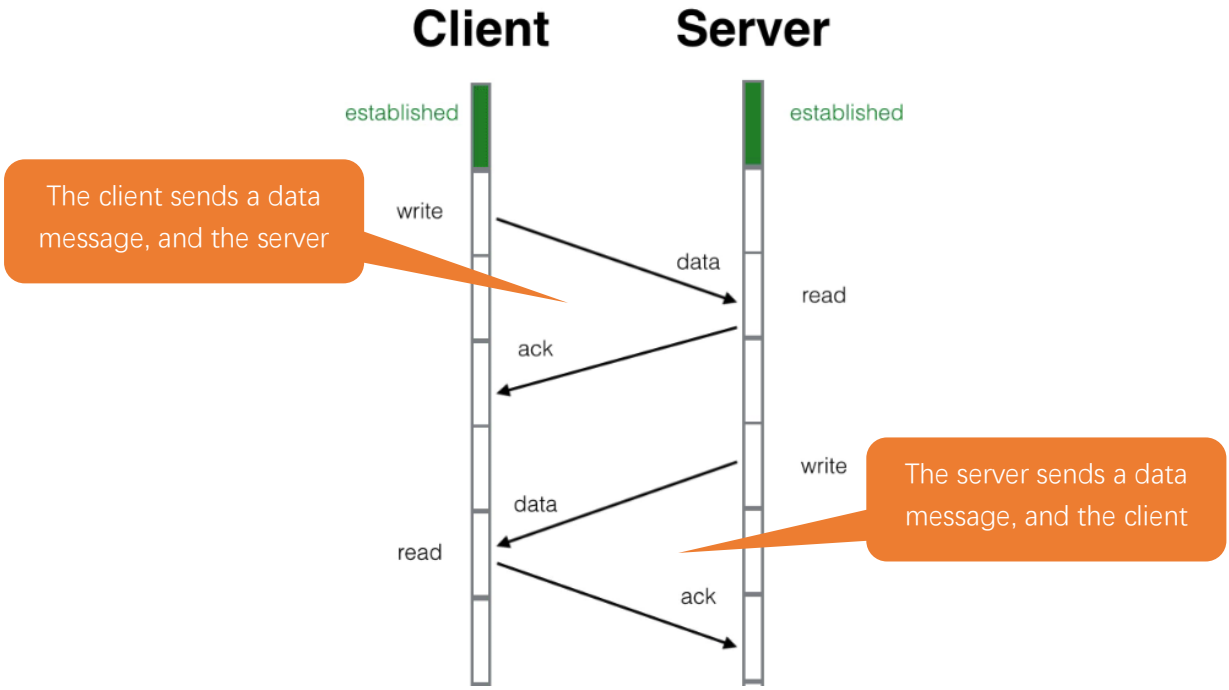
TCP connection

Before transmitting data, TCP needs to establish a logical connection between the sending end and the receiving end. It provides reliable and error-free data transmission between the two computers. In the TCP connection, the client and the server must be clarified. The client sends a connection request to the server, and each time such a request is proposed, a "three-times handshake" is required.

Three-times handshake: In the TCP protocol, during the preparation phase of sending data, the client and the server interact three times to ensure the reliability of the connection, which is called "three-times handshake".
The first handshake, the client sends a connection request to the server and waits for the server to confirm.
The second handshake, the server sends a response back to the client informing that it has received the connection request.
The third handshake, the client sends a confirmation message to the server again to confirm the connection.



TCP is a connection-oriented, low-level transmission control protocol. After TCP establishes a connection, the client and server can send and receive messages to each other, and the connection will always exist as long as the client or server does not initiate disconnection. Each time one party sends a message, the other party will reply with an ack signal.



Install Processing

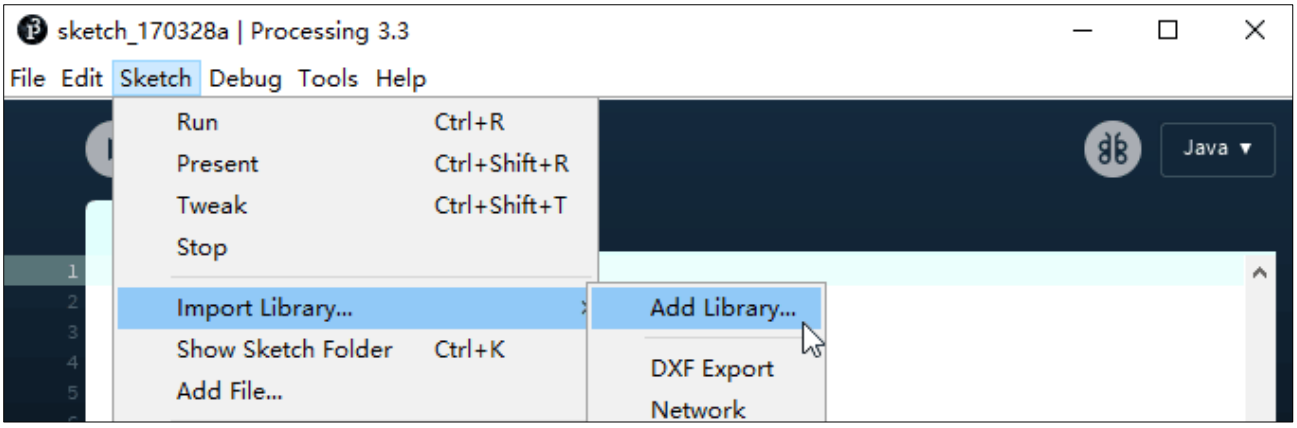
In this tutorial, we use Processing to build a simple TCP/IP communication platform.

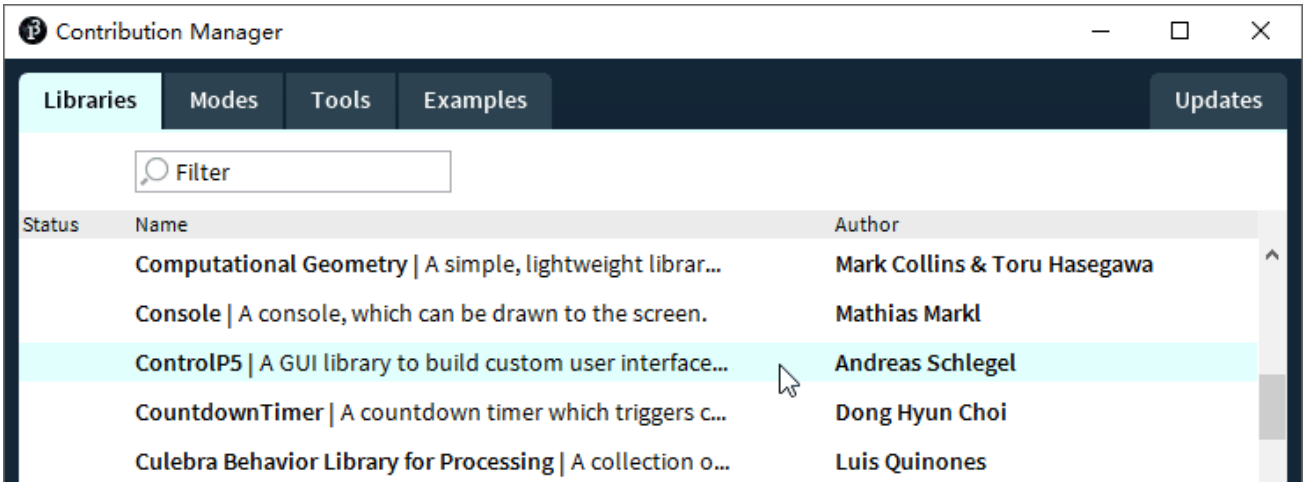
If you've not installed Processing, you can download it by clicking https://processing.org/download/. You can choose an appropriate version to download according to your PC system.

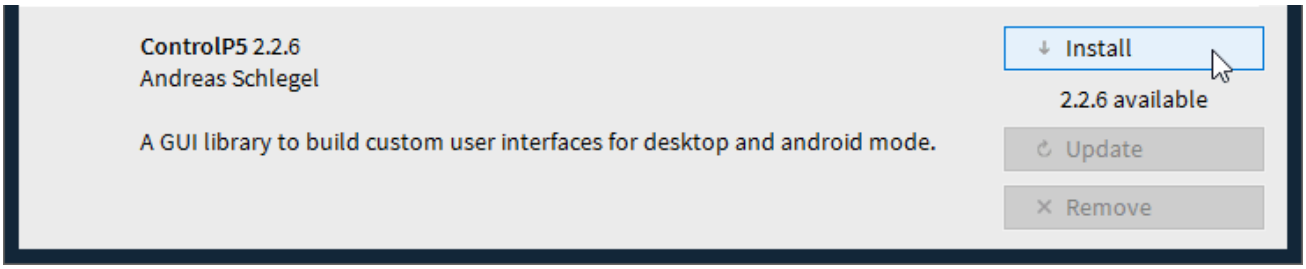
Unzip the downloaded file to your computer. Click "processing.exe" as the figure below to run this software.

Use Server mode for communication

Install ControlP5.

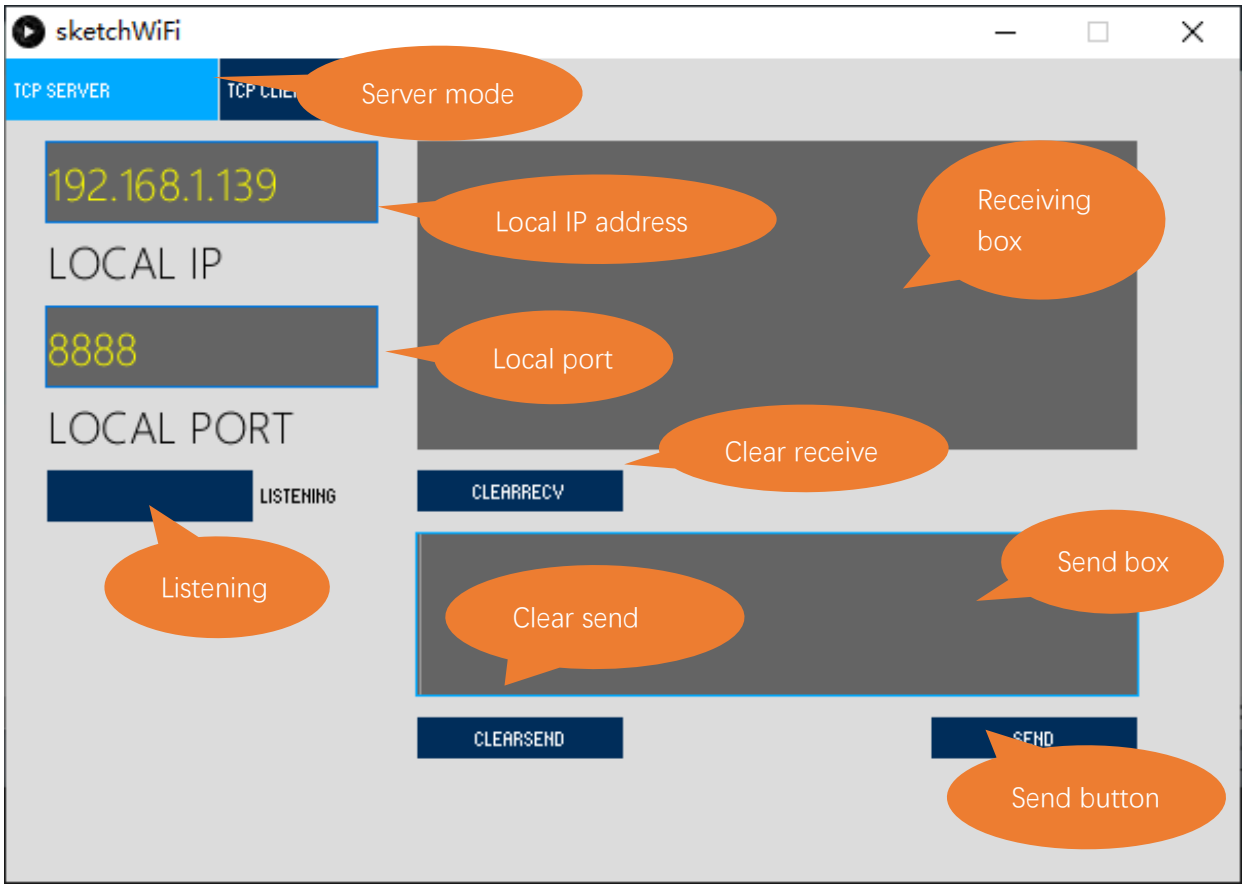






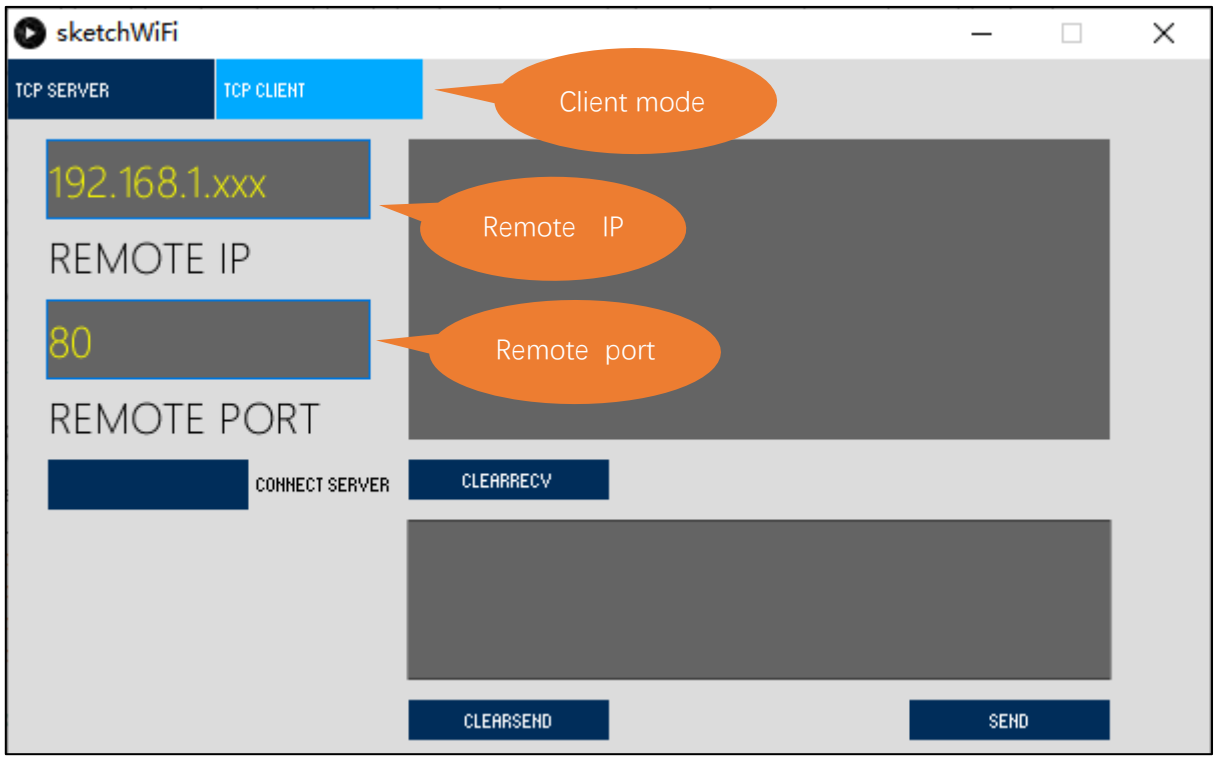
Open the "**Freenove_Super_Starter_Kit_for_ESP32_S3\Sketches\Sketches\Sketch_24.1_WiFiClient\sketchWiFi\sketchWiFi.pde**", and click "Run".

The new pop-up interface is as follows. If ESP32-S3 is used as client, select TCP SERVER mode for sketchWiFi.



When sketchWiFi selects TCP SERVER mode, ESP32-S3 Sketch needs to be changed according to sketchWiFi's displaying of LOCAL IP or LOCAL PORT.

If ESP32-S3 serves as server, select TCP CLIENT mode for sketchWiFi.



When sketchWiFi selects TCP CLIENT mode, the LOCAL IP and LOCAL PORT of sketchWiFi need to be changed according to the IP address and port number printed by the serial monitor.

Mode selection: select **Server mode/Client mode**.

IP address: In server mode, this option does not need to be filled in, and the computer will automatically obtain the IP address.

In client mode, fill in the remote IP address to be connected.

Port number: In server mode, fill in a port number for client devices to make an access connection.

In client mode, fill in port number given by the Server devices to make an access connection.

Start button: In server mode, push the button, then the computer will serve as server and open a port number for client to make access connection. During this period, the computer will keep monitoring.

In client mode, before pushing the button, please make sure the server is on, remote IP address and remote port number is correct; push the button, and the computer will make access connection to the remote port number of the remote IP as a client.

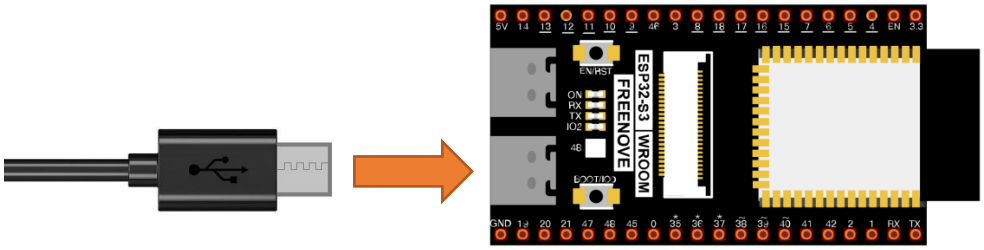
clear receive: clear out the content in the receiving text box

clear send: clear out the content in the sending text box

Sending button: push the sending button, the computer will send the content in the text box to others.

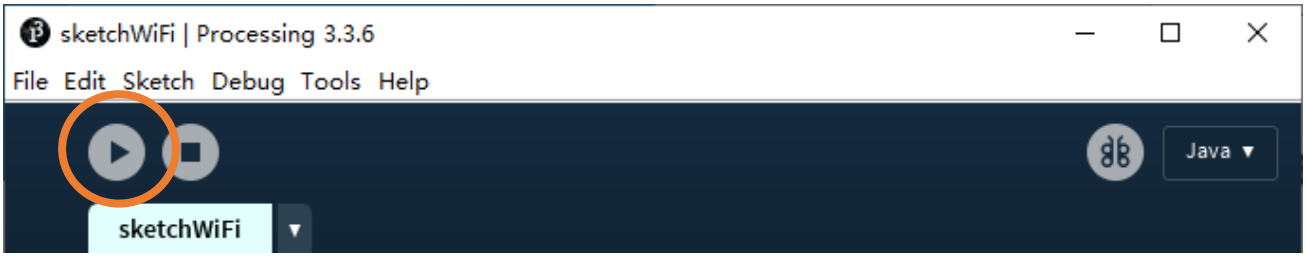
Circuit

Connect Freenove ESP32-S3 to the computer using the USB cable.

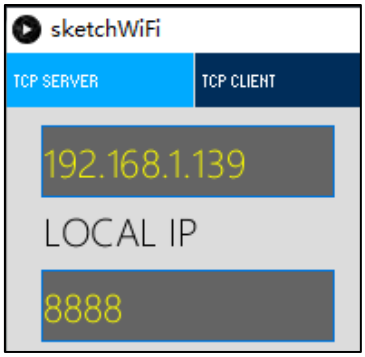


Sketch

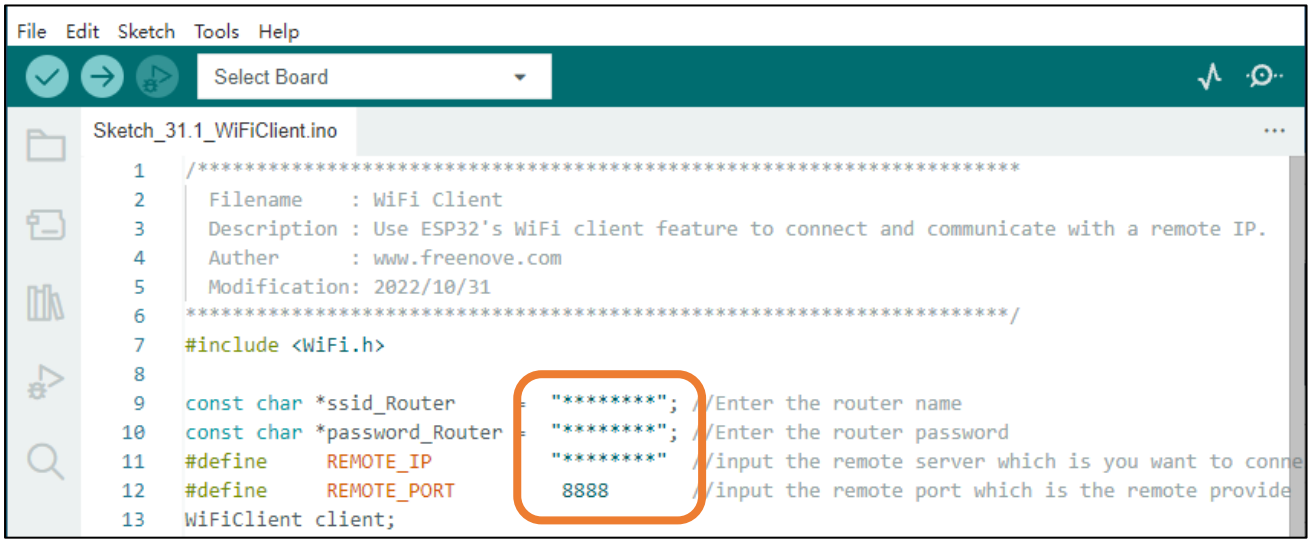
Before running the Sketch, please open “sketchWiFi.pde.” first, and click “Run”.



The newly pop up window will use the computer’s IP address by default and open a data monitor port.

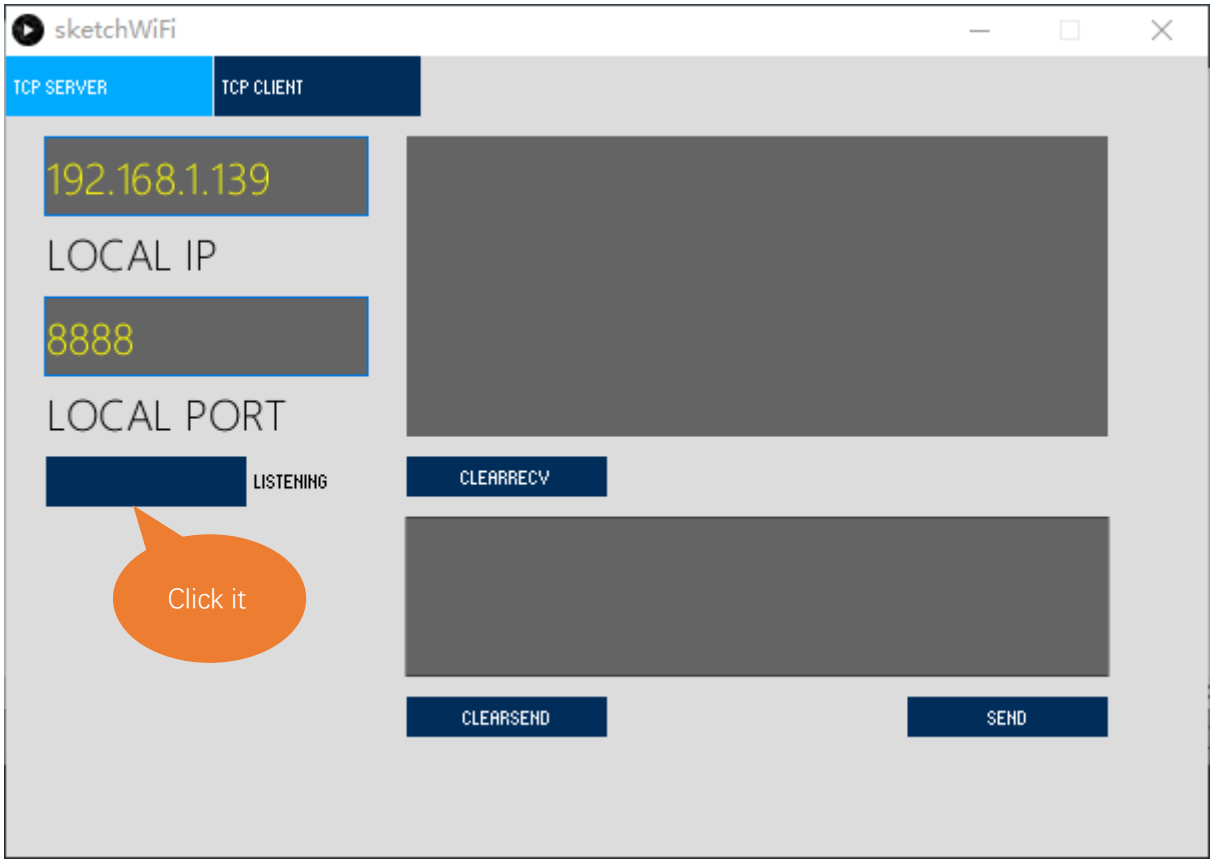


Next, open Sketch_24.1_WiFiClient.ino. Before running it, please change the following information based on "LOCAL IP" and "LOCAL PORT" in the figure above.

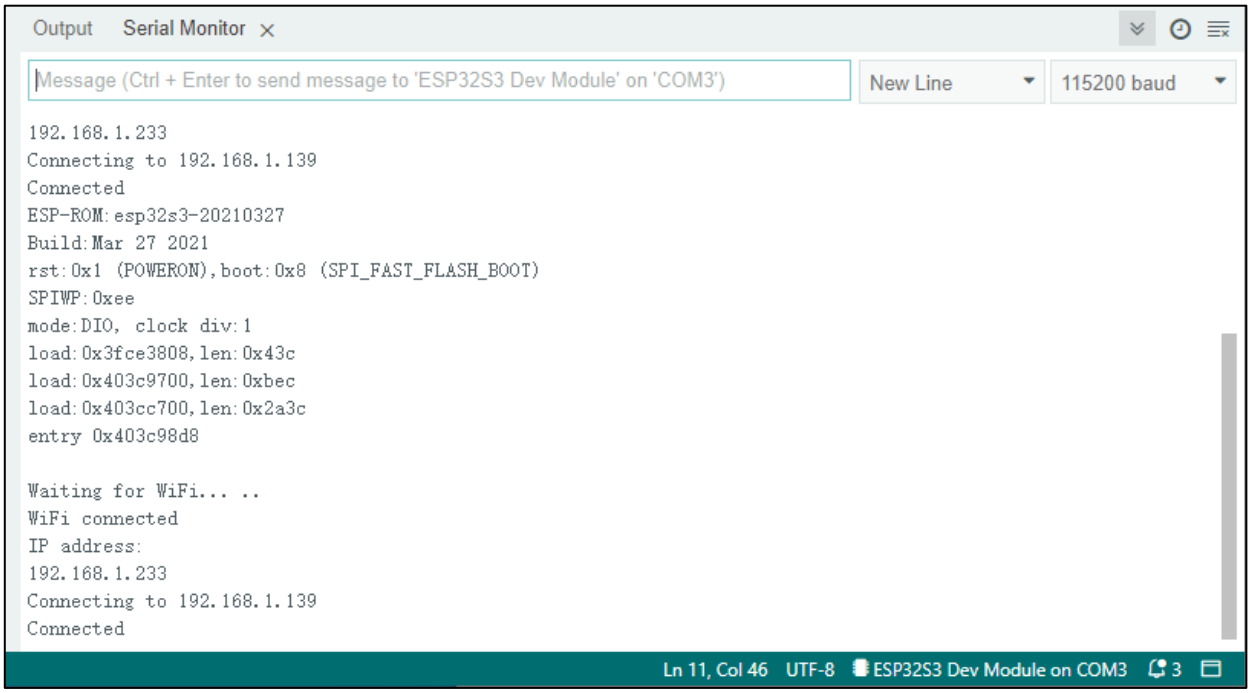


REMOTE_IP needs to be filled in according to the interface of sketchWiFi.pde. Taking this tutorial as an example, its REMOTE_IP is "192.168.1.133". Generally, by default, the ports do not need to change its value.

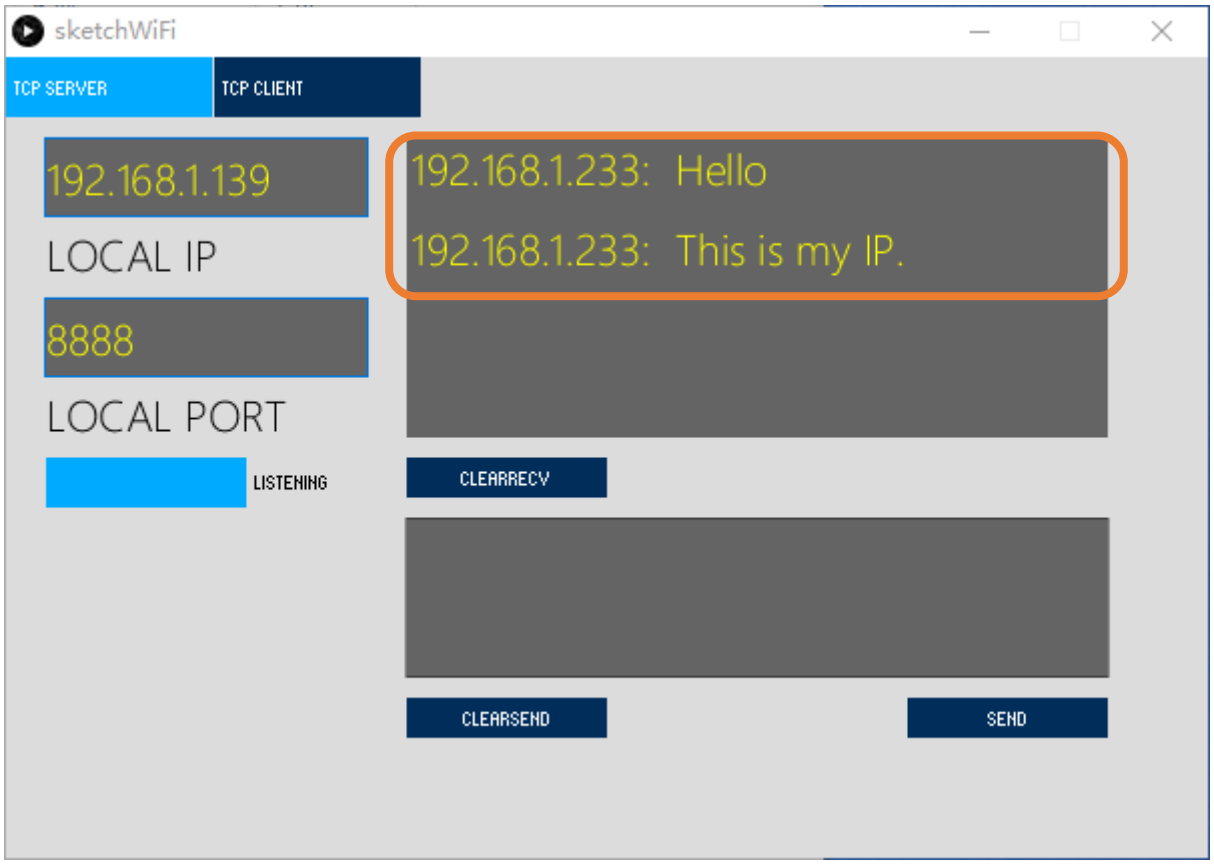
Click LISTENING, turn on TCP SERVER's data listening function and wait for ESP32-S3 to connect.



Compile and upload code to ESP32-S3 WROOM, open the serial monitor and set the baud rate to 115200. ESP32-S3 connects router, obtains IP address and sends access request to server IP address on the same LAN till the connection is successful. When connect successfully, ESP32-S3 can send messages to server.



ESP32-S3 connects with TCP SERVER, and TCP SERVER receives messages from ESP32-S3, as shown in the figure below.



Sketch_24.1_As_Client

The following is the program code:

```
#include <WiFi.h>

const char *ssid_Router     =  "********"; //Enter the router name
const char *password_Router =  "********"; //Enter the router password
#define     REMOTE_IP       "********"  //input the remote server which is you want to connect
#define     REMOTE_PORT      8888       //input the remote port which is the remote provide
WiFiClient client;

void setup() {
  Serial.begin(115200);
  delay(10);

  WiFi.begin(ssid_Router, password_Router);
  Serial.print("\nWaiting for WiFi... ");
  while (WiFi.status() != WL_CONNECTED) {
    Serial.print(".");
    delay(500);
  }
  Serial.println("");
  Serial.println("WiFi connected");
  Serial.println("IP address: ");
  Serial.println(WiFi.localIP());
  delay(500);

  Serial.print("Connecting to ");
  Serial.println(REMOTE_IP);

  while (!client.connect(REMOTE_IP, REMOTE_PORT)) {
    Serial.println("Connection failed.");
    Serial.println("Waiting a moment before retrying...");
  }
  Serial.println("Connected");
  client.print("Hello\n");
  client.print("This is my IP.\n");
}

void loop() {
  if (client.available() > 0) {
    delay(20);
    //read back one line from the server
    String line = client.readString();
    Serial.println(REMOTE_IP + String(":") + line);
```

```
42    }
43    if (Serial.available() > 0) {
44      delay(20);
45      String line = Serial.readString();
46      client.print(line);
47    }
48    if (client.connected () == 0) {
49      client.stop();
50      WiFi.disconnect();
51    }
52  }
```

Add WiFi function header file.

```
1   #include <WiFi.h>
```

Enter the actual router name, password, remote server IP address, and port number.

```
3   const char *ssid_Router     =  "********"; //Enter the router name
4   const char *password_Router =  "********"; //Enter the router password
5   #define     REMOTE_IP       "********"  //input the remote server which is you want to connect
6   #define     REMOTE_PORT      8888       //input the remote port which is the remote provide
```

Apply for the method class of WiFiClient.

```
7   WiFiClient client;
```

Connect specified WiFi until it is successful. If the name and password of WiFi are correct but it still fails to connect, please push the reset key.

```
13    WiFi.begin(ssid_Router, password_Router);
14    Serial.print("\nWaiting for WiFi... ");
15    while (WiFi.status() ! = WL_CONNECTED) {
16      Serial.print(".");
17      delay(500);
18    }
```

Send connection request to remote server until connect successfully. When connect successfully, print out the connecting prompt on the serial monitor and send messages to remote server.

```
28    while (!client.connect(REMOTE_IP, REMOTE_PORT)) {//Connect to Server
29      Serial.println("Connection failed.");
30      Serial.println("Waiting a moment before retrying...");
31    }
32    Serial.println("Connected");
33    client.print("Hello\n");
```

When ESP32-S3 receive messages from servers, it will print them out via serial port; Users can also send messages to servers from serial port.

```
37  if (client.available() > 0) {
38      delay(20);
39      //read back one line from the server
40      String line = client.readString();
41      Serial.println(REMOTE_IP + String(":") + line);
42    }
```

```
43    if (Serial.available() > 0) {
44      delay(20);
45      String line = Serial.readString();
46      client.print(line);
47    }
```

If the server is disconnected, turn off WiFi of ESP32-S3.

```
48    if (client.connected () == false) {
49      client.stop();
50      WiFi.disconnect();
51    }
```

Reference

Class Client

Every time when using Client, you need to include header file "WiFi.h."

connect(ip, port, timeout)/connect(*host, port, timeout): establish a TCP connection.

ip, *host: ip address of target server

port: port number of target server

timeout: connection timeout

connected(): judge whether client is connecting. If return value is 1, then connect successfully; If return value is 0, then fail to connect.

stop(): stop tcp connection

print(): send data to server connecting to client

available(): return to the number of bytes readable in receive buffer, if no, return to 0 or -1.

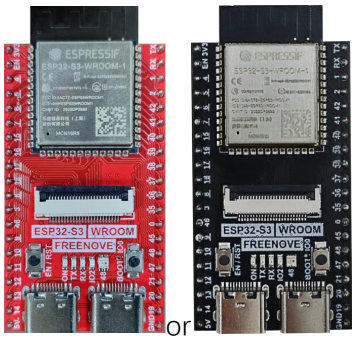
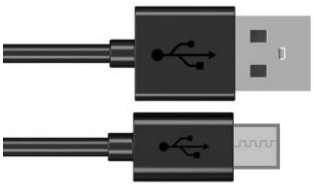
read(): read one byte of data in receive buffer

readString(): read string in receive buffer

Project 24.2 As Server

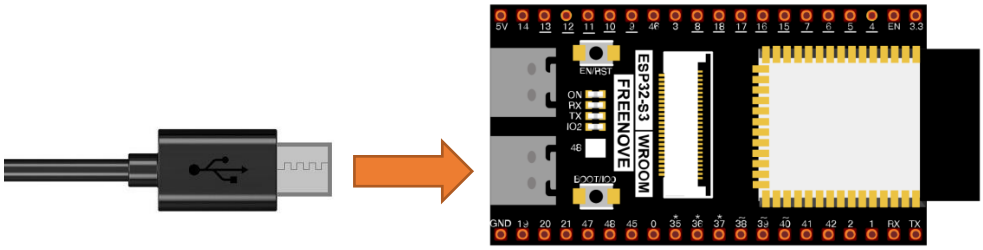
In this section, ESP32-S3 is used as a server to wait for the connection and communication of client on the same LAN.

Component List

ESP32-S3(N16R8)/ESP32-S3(N8R8)  or	USB cable x1 

Circuit

Connect Freenove ESP32-S3 to the computer using a USB cable.



Sketch

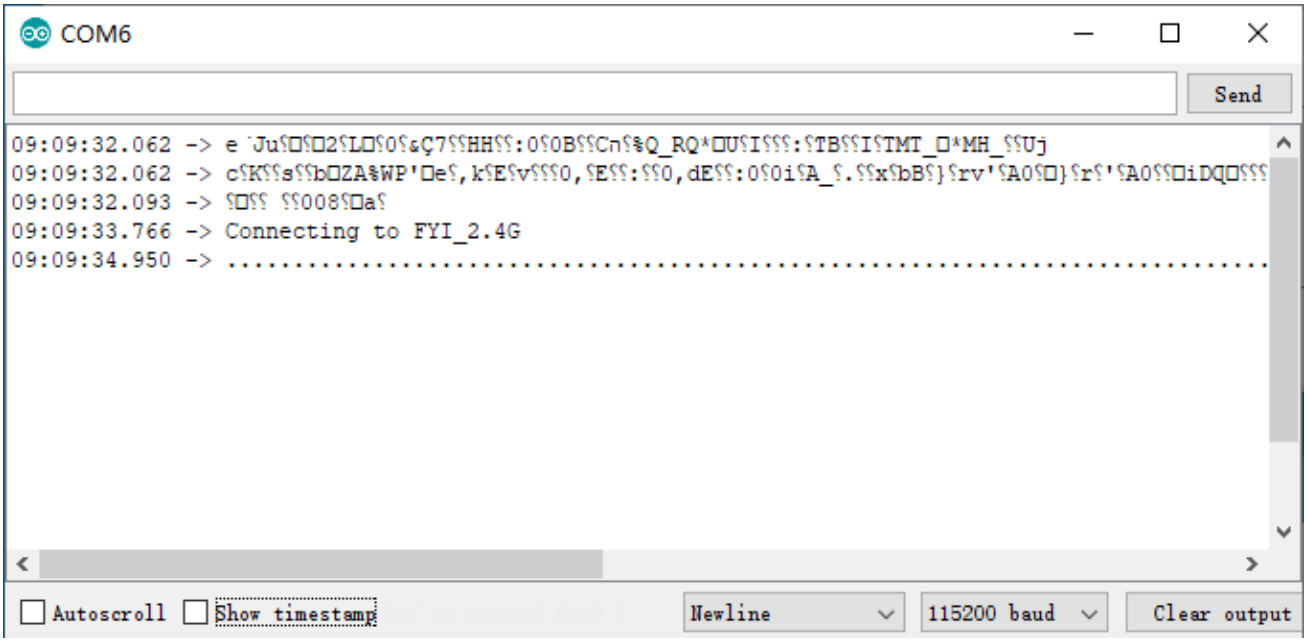
Before running Sketch, please modify the contents of the box below first.

Sketch_24.2_As_Server

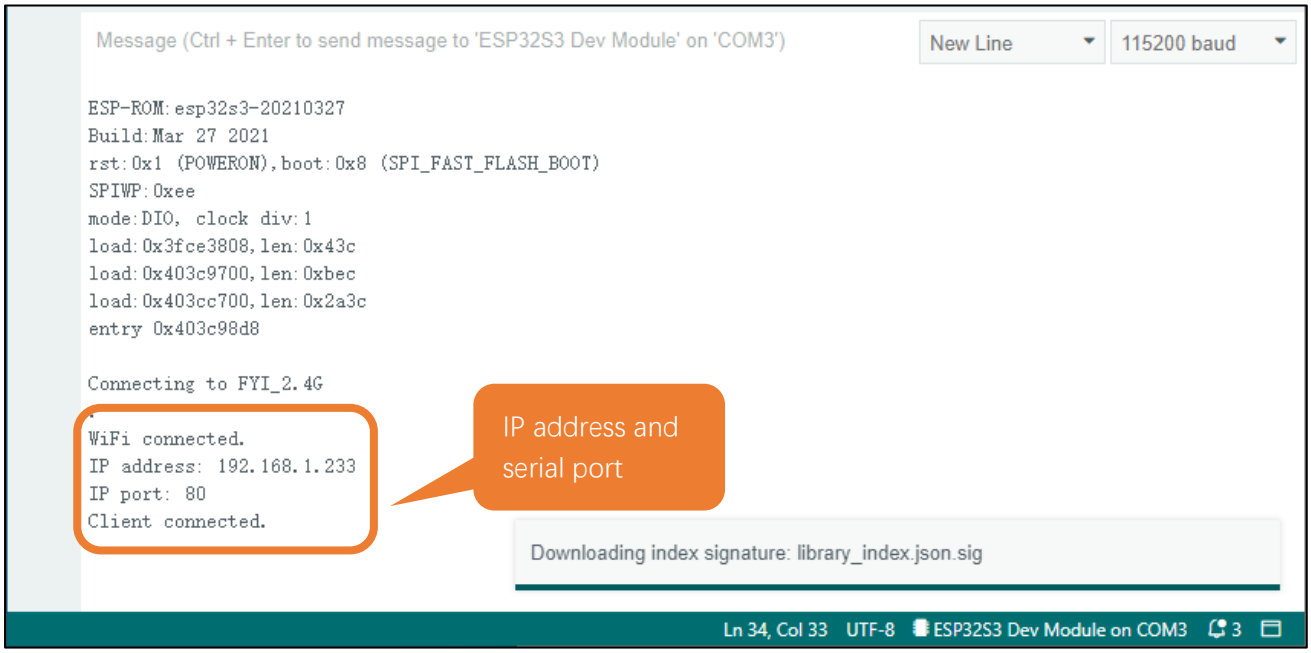
```
#include <WiFi.h>

#define port 80
const char *ssid_Router     = "********";  //input your wifi name
const char *password_Router = "********";  //input your wifi passwords
WiFiServer  server(port);

void setup()
{
    Serial.begin(115200);
    Serial.printf("\nConnecting to ");
    Serial.println(ssid_Router);
    WiFi.disconnect();
```

Compile and upload code to ESP32-S3 WROOM board, open the serial monitor and set the baud rate to 115200. Turn on server mode for ESP32-S3, waiting for the connection of other devices on the same LAN. Once a device connects to server successfully, they can send messages to each other.

If the ESP32-S3 fails to connect to router, press the reset button as shown below and wait for ESP32-S3 to run again.



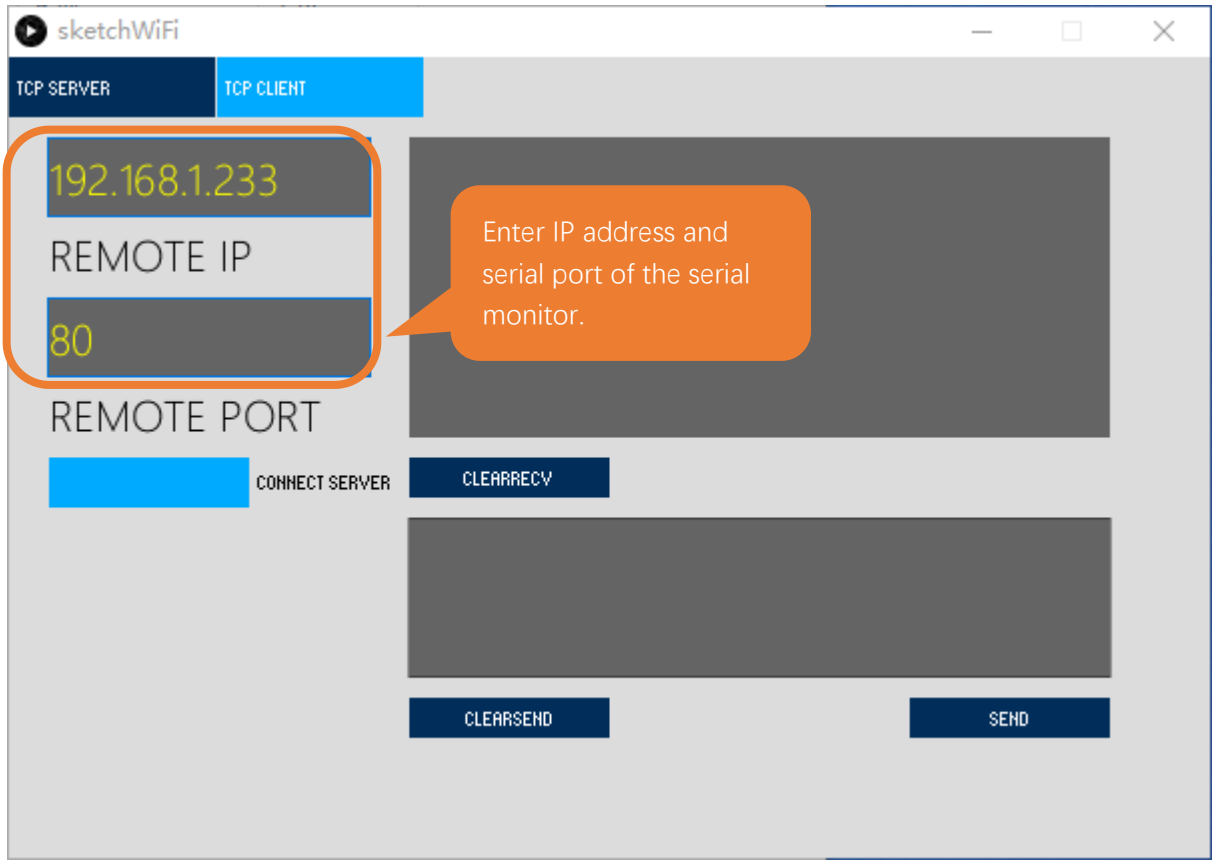
Serial Monitor



Processing:

Open the "**Freenove_Super_Starter_Kit_for_ESP32_S3\Sketches\Sketches\Sketch_24.2_WiFiServer\sketchWiFi\sketchWiFi.pde**".

Based on the messages printed by the serial monitor, enter correct IP address and serial port in Processing to establish connection and make communication.



The following is the program code:

```
#include <WiFi.h>

#define port 80
const char *ssid_Router      = "********";  //input your wifi name
const char *password_Router  = "********";  //input your wifi passwords
WiFiServer  server(port);

void setup()
{
    Serial.begin(115200);
    Serial.printf("\nConnecting to ");
    Serial.println(ssid_Router);
    WiFi.disconnect();
    WiFi.begin(ssid_Router, password_Router);
    delay(1000);
    while (WiFi.status() != WL_CONNECTED) {
        delay(500);
        Serial.print(".");
    }
    Serial.println("");
    Serial.println("WiFi connected.");
    Serial.print("IP address: ");
    Serial.println(WiFi.localIP());
    Serial.printf("IP port: %d\n",port);
    server.begin(port);
    WiFi.setAutoReconnect(true);
}

void loop(){
 WiFiClient client = server.accept();                // listen for incoming clients
  if (client) {                                      // if you get a client
    Serial.println("Client connected.");
    while (client.connected()) {                     // loop while the client's connected
      if (client.available()) {                      // if there's bytes to read from the
client
        Serial.println(client.readStringUntil('\n')); // print it out the serial monitor
        while(client.read()>0);                      // clear the wifi receive area cache
      }
      if(Serial.available()){                        // if there's bytes to read from the
serial monitor
        client.print(Serial.readStringUntil('\n')); // print it out the client.
        while(Serial.read()>0);                      // clear the wifi receive area cache
      }
```

```
42      }
43      client.stop();                                  // stop the client connecting.
44      Serial.println("Client Disconnected.");
45    }
46  }
```

Apply for method class of WiFiServer.

```
6   WiFiServer server(port);          //Apply for a Server object whose port number is 80
```

Connect specified WiFi until it is successful. If the name and password of WiFi are correct but it still fails to connect, please push the reset key.

```
13      WiFi.disconnect();
14      WiFi.begin(ssid_Router, password_Router);
15      delay(1000);
16      while (WiFi.status() != WL_CONNECTED) {
17          delay(500);
18          Serial.print(".");
19      }
20      Serial.println("");
21      Serial.println("WiFi connected.");
```

Print out the IP address and port number of ESP32-S3.

```
22      Serial.print("IP address: ");
23      Serial.println(WiFi.localIP());                  //print out IP address of ESP32-S3
24      Serial.printf("IP port: %d\n",port);             //Print out ESP32-S3's port number
```

Turn on server mode of ESP32-S3, turn on automatic reconnection.

```
25      server.begin();                                  //Turn ON ESP32-S3 as Server mode
26      WiFi.setAutoReconnect(true);
```

When ESP32-S3 receive messages from servers, it will print them out via serial port; Users can also send messages to servers from serial port.

```
34      if (client.available()) {                       // if there's bytes to read from the
        client
35        Serial.println(client.readStringUntil('\n')); // print it out the serial monitor
36        while(client.read()>0);                       // clear the wifi receive area cache
37      }
38      if(Serial.available()){                         // if there's bytes to read from the
        serial monitor
39        client.print(Serial.readStringUntil('\n')); // print it out the client.
40        while(Serial.read()>0);                       // clear the wifi receive area cache
41      }
```

Reference

Class Server

Every time use Server functionality, we need to include header file"WiFi.h".

WiFiServer(uint16_t port=80, uint8_t max_clients=4): create a TCP Server.

port: ports of Server; range from 0 to 65535 with the default number as 80.

max_clients: maximum number of clients with default number as 4.

begin(port): start the TCP Server.

port: ports of Server; range from 0 to 65535 with the default number as 0.

setNoDelay(bool nodelay): whether to turn off the delay sending functionality.

nodelay: true stands for forbidden Nagle algorithm.

close(): close tcp connection.

stop(): stop tcp connection.

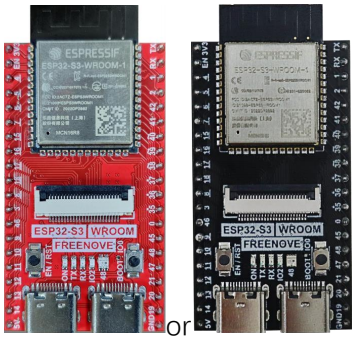
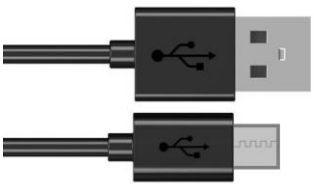
Chapter 25 Camera Web Server

In this section, we'll use ESP32-S3's video function as an example to study.

Project 25.1 Camera Web Server

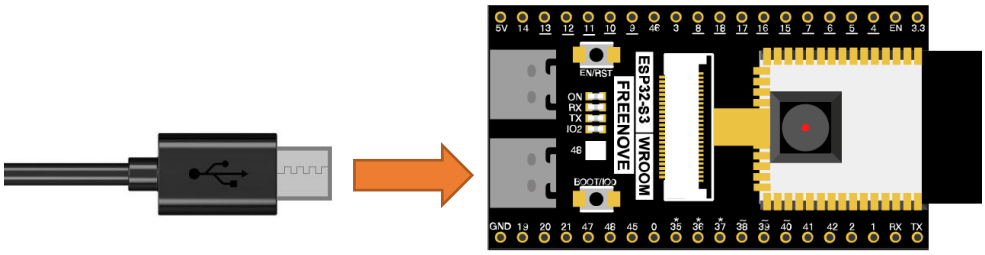
Connect ESP32-S3 using USB and check its IP address through serial monitor. Use web page to access IP address to obtain video and image data.

Component List

ESP32-S3(N16R8)/ESP32-S3(N8R8) 	USB cable x1 

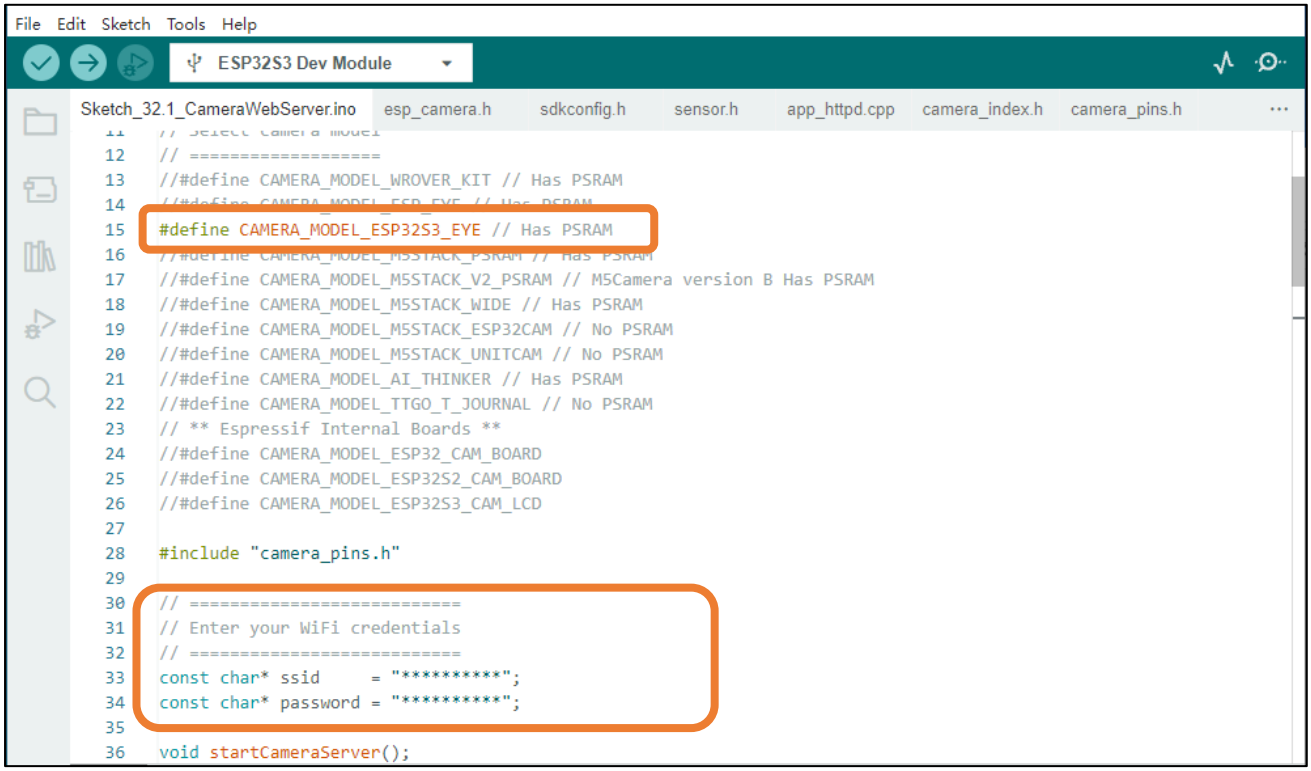
Circuit

Connect Freenove ESP32-S3 to the computer using the USB cable.



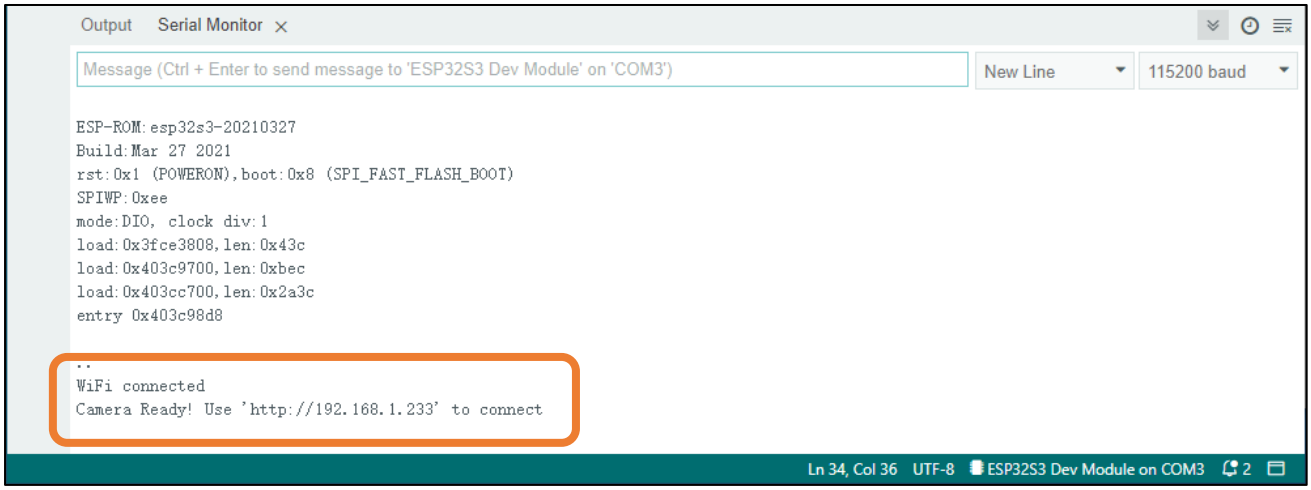
Sketch

Sketch_25.1_As_CameraWebServer



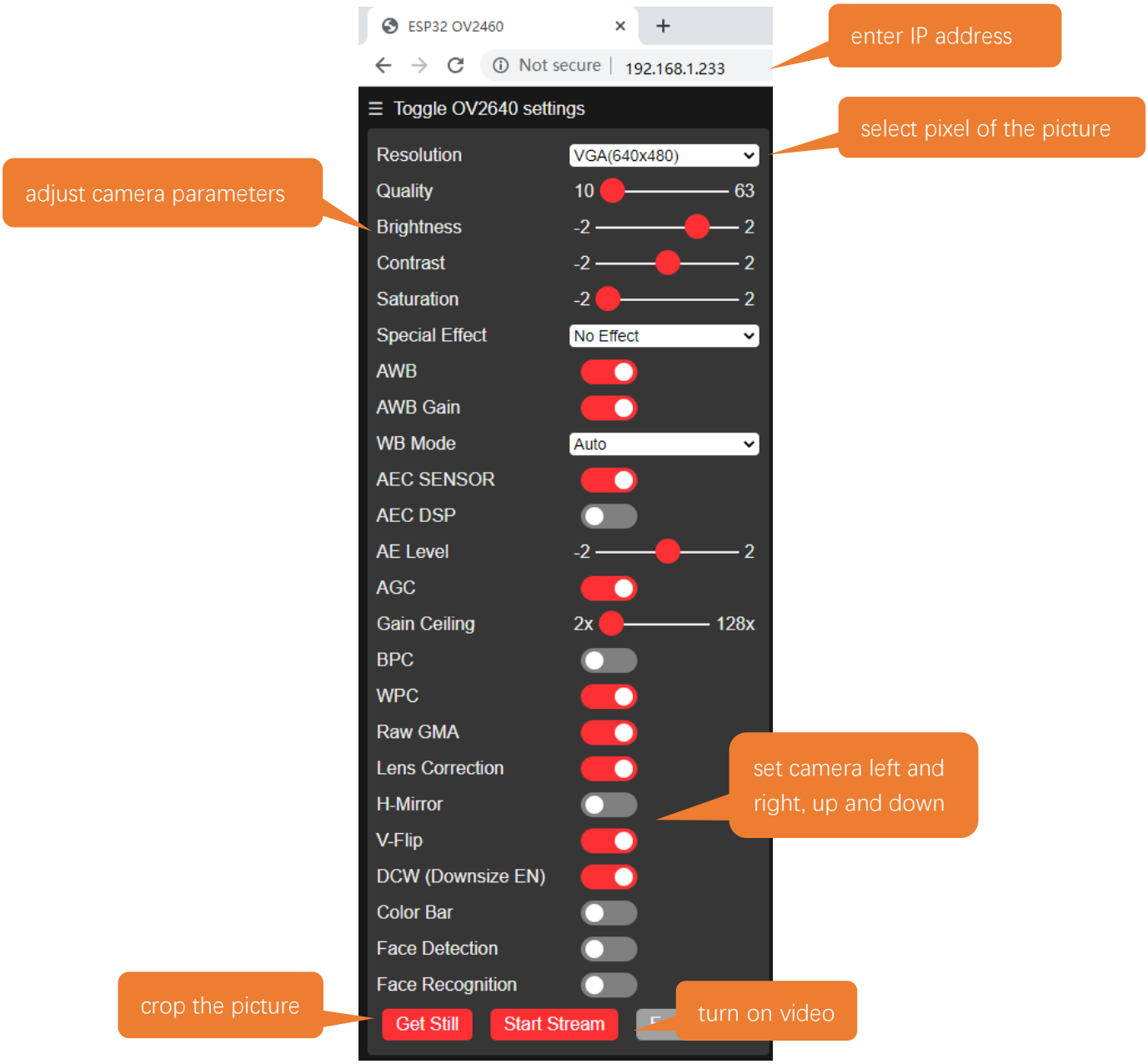
Before running the program, please modify your router's name and password in the box shown in the illustration above to make sure that your Sketch can compile and work successfully.

Compile and upload codes to ESP32-S3, open the serial monitor and set the baud rate to 115200, and the serial monitor will print out a network link address.

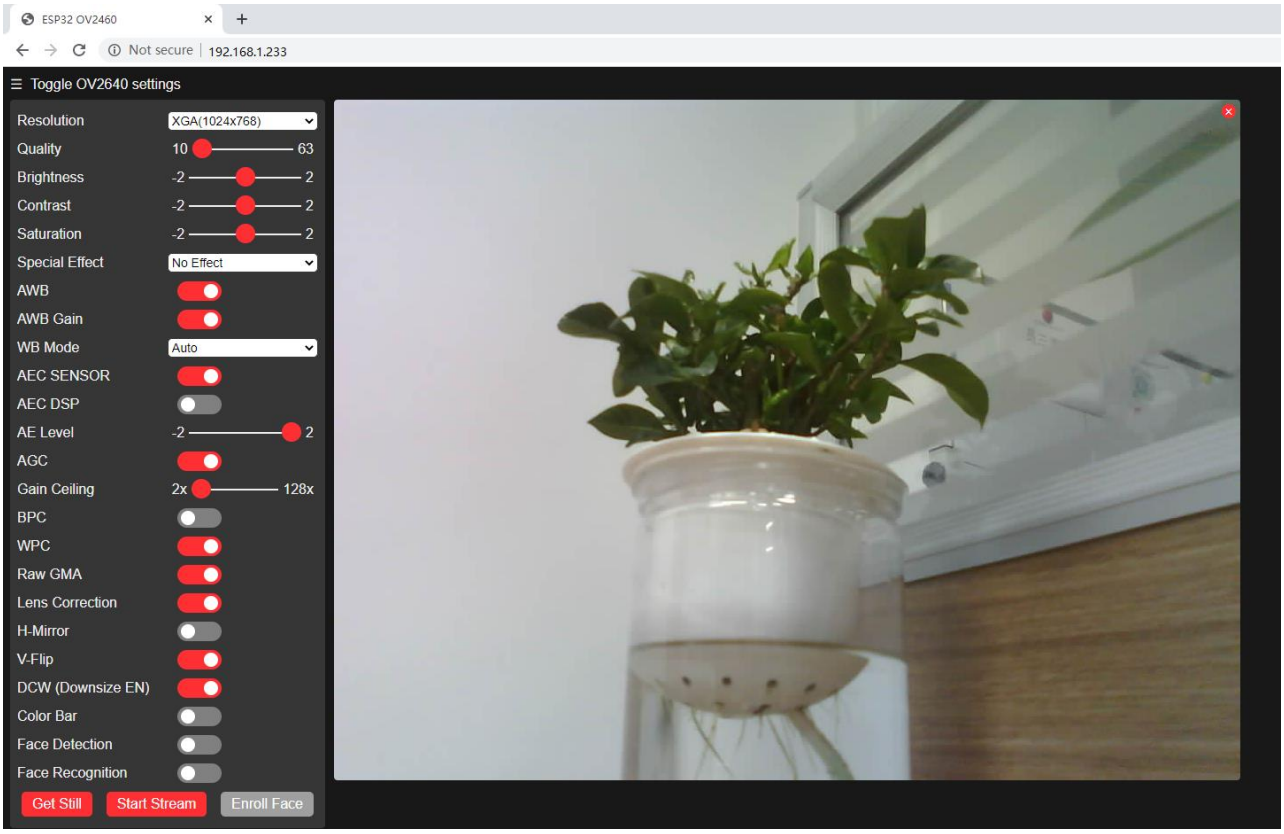


If your ESP32-S3 has been in the process of connecting to router, but the information above has not been printed out, please re-check whether the router name and password have been entered correctly and press the reset key on ESP32-S3 WROOM to wait for a successful connection prompt.
Open a web browser, enter the IP address printed by the serial monitor in the address bar, and access it.

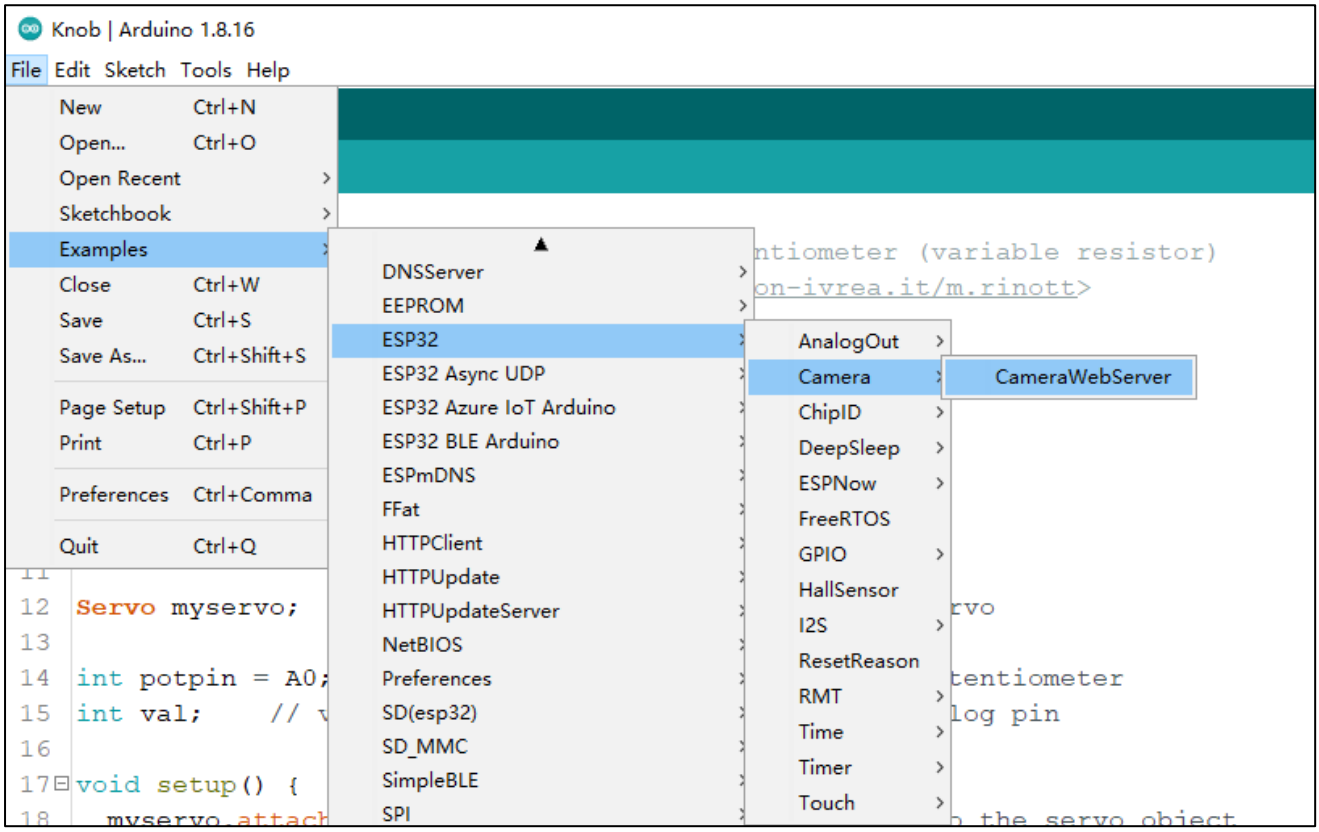
Taking the Google browser as an example, here's what the browser prints out after successful access to ESP32-S3's IP.



Click on Start Stream. The effect is shown in the image below.



Note: If sketch compilation fails due to ESP32-S3 support package, follow the steps of the image to open the CameraWebServer. This sketch is the same as described in the tutorial above.



The following is the main program code. You need include other code files in the same folder when write your own code.

```
#include "esp_camera.h"
#include <WiFi.h>

// ===================
// Select camera model
// ===================
//#define CAMERA_MODEL_WROVER_KIT // Has PSRAM
//#define CAMERA_MODEL_ESP_EYE // Has PSRAM
#define CAMERA_MODEL_ESP32S3_EYE // Has PSRAM
//#define CAMERA_MODEL_M5STACK_PSRAM // Has PSRAM
//#define CAMERA_MODEL_M5STACK_V2_PSRAM // M5Camera version B Has PSRAM
//#define CAMERA_MODEL_M5STACK_WIDE // Has PSRAM
//#define CAMERA_MODEL_M5STACK_ESP32CAM // No PSRAM
//#define CAMERA_MODEL_M5STACK_UNITCAM // No PSRAM
//#define CAMERA_MODEL_AI_THINKER // Has PSRAM
//#define CAMERA_MODEL_TTGO_T_JOURNAL // No PSRAM
// ** Espressif Internal Boards **
//#define CAMERA_MODEL_ESP32_CAM_BOARD
//#define CAMERA_MODEL_ESP32S2_CAM_BOARD
//#define CAMERA_MODEL_ESP32S3_CAM_LCD

#include "camera_pins.h"

// ===========================
// Enter your WiFi credentials
// ===========================
const char* ssid     = "********";
const char* password = "********";

void startCameraServer();

void setup() {
  Serial.begin(115200);
  Serial.setDebugOutput(true);
  Serial.println();

  camera_config_t config;
  config.ledc_channel = LEDC_CHANNEL_0;
  config.ledc_timer = LEDC_TIMER_0;
  config.pin_d0 = Y2_GPIO_NUM;
  config.pin_d1 = Y3_GPIO_NUM;
  config.pin_d2 = Y4_GPIO_NUM;
```

```
  config.pin_d3 = Y5_GPIO_NUM;
  config.pin_d4 = Y6_GPIO_NUM;
  config.pin_d5 = Y7_GPIO_NUM;
  config.pin_d6 = Y8_GPIO_NUM;
  config.pin_d7 = Y9_GPIO_NUM;
  config.pin_xclk = XCLK_GPIO_NUM;
  config.pin_pclk = PCLK_GPIO_NUM;
  config.pin_vsync = VSYNC_GPIO_NUM;
  config.pin_href = HREF_GPIO_NUM;
  config.pin_sccb_sda = SIOD_GPIO_NUM;
  config.pin_sccb_scl = SIOC_GPIO_NUM;
  config.pin_pwdn = PWDN_GPIO_NUM;
  config.pin_reset = RESET_GPIO_NUM;
  config.xclk_freq_hz = 20000000;
  config.frame_size = FRAMESIZE_UXGA;
  config.pixel_format = PIXFORMAT_JPEG; // for streaming
  config.grab_mode = CAMERA_GRAB_WHEN_EMPTY;
  config.fb_location = CAMERA_FB_IN_PSRAM;
  config.jpeg_quality = 12;
  config.fb_count = 1;

  // if PSRAM IC present, init with UXGA resolution and higher JPEG quality
  // for larger pre-allocated frame buffer.
  if(psramFound()){
    config.jpeg_quality = 10;
    config.fb_count = 2;
    config.grab_mode = CAMERA_GRAB_LATEST;
  } else {
    // Limit the frame size when PSRAM is not available
    config.frame_size = FRAMESIZE_SVGA;
    config.fb_location = CAMERA_FB_IN_DRAM;
  }

  // camera init
  esp_err_t err = esp_camera_init(&config);
  if (err != ESP_OK) {
    Serial.printf("Camera init failed with error 0x%x", err);
    return;
  }

  sensor_t * s = esp_camera_sensor_get();
  // initial sensors are flipped vertically and colors are a bit saturated
  s->set_vflip(s, 1); // flip it back
  s->set_brightness(s, 1); // up the brightness just a bit
```

```
87    s->set_saturation(s, -1); // lower the saturation
88
89    WiFi.begin(ssid, password);
90    WiFi.setSleep(false);
91
92    while (WiFi.status() != WL_CONNECTED) {
93      delay(500);
94      Serial.print(".");
95    }
96    while (WiFi.STA.hasIP() != true) {
97      delay(500);
98      Serial.print(".");
99    }
100
101   Serial.println("");
102   Serial.println("WiFi connected");
103
104   startCameraServer();
105
106   Serial.print("Camera Ready! Use 'http://");
107   Serial.print(WiFi.localIP());
108   Serial.println("' to connect");
109 }
110
111 void loop() {
112   // Do nothing. Everything is done in another task by the web server
113   delay(10000);
114 }
```

Add procedure files and API interface files related to ESP32-S3 camera.

```
1   #include "esp_camera.h"
2   #include <WiFi.h>
…
9   #define CAMERA_MODEL_ESP32S3_EYE // Has PSRAM
…
11  #include "camera_pins.h"
```

Enter the name and password of the router

```
13  const char *ssid     = "********";  //input your wifi name
14  const char *password = "********";  //input your wifi passwords
```

Initialize serial port, set baud rate to 115200; open the debug and output function of the serial.

```
21    Serial.begin(115200);
22    Serial.setDebugOutput(true);
23    Serial.println();
```

Configure parameters including interface pins of the camera. Note: It is generally not recommended to change them.

```
37    camera_config_t config;
38    config.ledc_channel = LEDC_CHANNEL_0;
39    config.ledc_timer = LEDC_TIMER_0;
40    config.pin_d0 = Y2_GPIO_NUM;
41    config.pin_d1 = Y3_GPIO_NUM;
42    config.pin_d2 = Y4_GPIO_NUM;
43    config.pin_d3 = Y5_GPIO_NUM;
44    config.pin_d4 = Y6_GPIO_NUM;
45    config.pin_d5 = Y7_GPIO_NUM;
46    config.pin_d6 = Y8_GPIO_NUM;
47    config.pin_d7 = Y9_GPIO_NUM;
48    config.pin_xclk = XCLK_GPIO_NUM;
49    config.pin_pclk = PCLK_GPIO_NUM;
50    config.pin_vsync = VSYNC_GPIO_NUM;
51    config.pin_href = HREF_GPIO_NUM;
52    config.pin_sccb_sda = SIOD_GPIO_NUM;
53    config.pin_sccb_scl = SIOC_GPIO_NUM;
54    config.pin_pwdn = PWDN_GPIO_NUM;
55    config.pin_reset = RESET_GPIO_NUM;
56    config.xclk_freq_hz = 20000000;
57    config.frame_size = FRAMESIZE_UXGA;
58    config.pixel_format = PIXFORMAT_JPEG; // for streaming
59    config.grab_mode = CAMERA_GRAB_WHEN_EMPTY;
60    config.fb_location = CAMERA_FB_IN_PSRAM;
61    config.jpeg_quality = 12;
62    config.fb_count = 1;
```

ESP32-S3 connects to the router and prints a successful connection prompt. If it has not been successfully connected, press the reset key on the ESP32-S3 WROOM.

```
89    WiFi.begin(ssid, password);
90    WiFi.setSleep(false);
91
92    while (WiFi.status() != WL_CONNECTED) {
93      delay(500);
94      Serial.print(".");
95    }
96    while (WiFi.STA.hasIP() != true) {
97      delay(500);
98      Serial.print(".");
99    }
100
101   Serial.println("");
102   Serial.println("WiFi connected");
```

Open the video streams server function of the camera and print its IP address via serial port.

```
99    startCameraServer();
100
101   Serial.print("Camera Ready! Use 'http://");
102   Serial.print(WiFi.localIP());
103   Serial.println("' to connect");
```

Configure the display image information of the camera.

The set_vflip() function sets whether the image is flipped 180°, with 0 for no flip and 1 for flip 180°.

The set_brightness() function sets the brightness of the image, with values ranging from -2 to 2.

The set_saturation() function sets the color saturation of the image, with values ranging from -2 to 2.

```
36    sensor_t * s = esp_camera_sensor_get();
37    s->set_vflip(s, 1);        //flip it back
38    s->set_brightness(s, 1);   //up the blightness just a bit
39    s->set_saturation(s, -1);  //lower the saturation
```

Modify the resolution and sharpness of the images captured by the camera. The sharpness ranges from 10 to 63, and the smaller the number, the sharper the picture. The larger the number, the blurrier the picture. Please refer to the table below.

```
26    config.frame_size = FRAMESIZE_VGA;
27    config.jpeg_quality = 10;
```

Reference

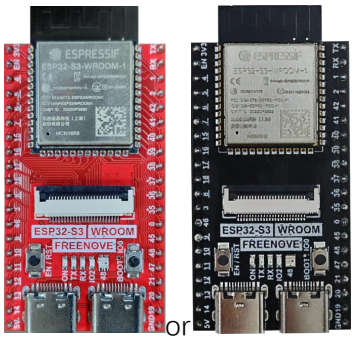
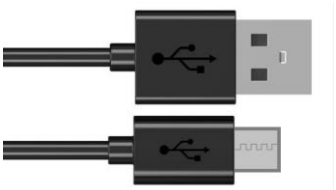
Image resolution	Sharpness	Image resolution	Sharpness
FRAMESIZE_96x96	96x96	FRAMESIZE_HVGA	480x320
FRAMESIZE_QQVGA	160x120	FRAMESIZE_VGA	640x480
FRAMESIZE_QCIF	176x144	FRAMESIZE_SVGA	800x600
FRAMESIZE_HQVGA	240x176	FRAMESIZE_XGA	1024x768
FRAMESIZE_240x240	240x240	FRAMESIZE_HD	1280x720
FRAMESIZE_QVGA	320x240	FRAMESIZE_SXGA	1280x1024
FRAMESIZE_CIF	400x296	FRAMESIZE_UXGA	1600x1200

We recommend that the resolution not exceed VGA(640x480).

Project 25.2 Video Web Server

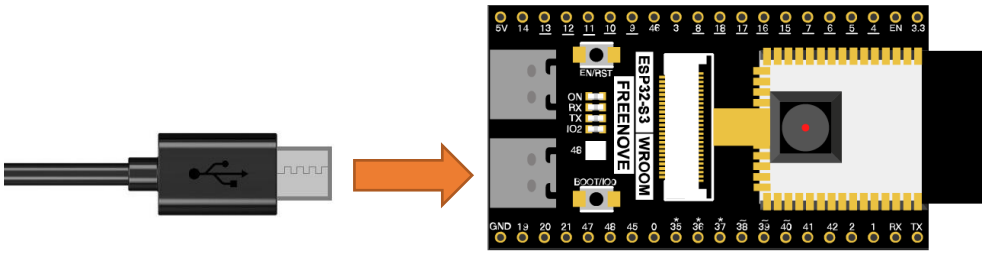
Connect to ESP32-S3 using USB and view its IP address through a serial monitor. Access IP addresses through web pages to obtain real-time video data.

Component List

ESP32-S3(N16R8)/ESP32-S3(N8R8) 	USB cable x1 	SDcard x1 

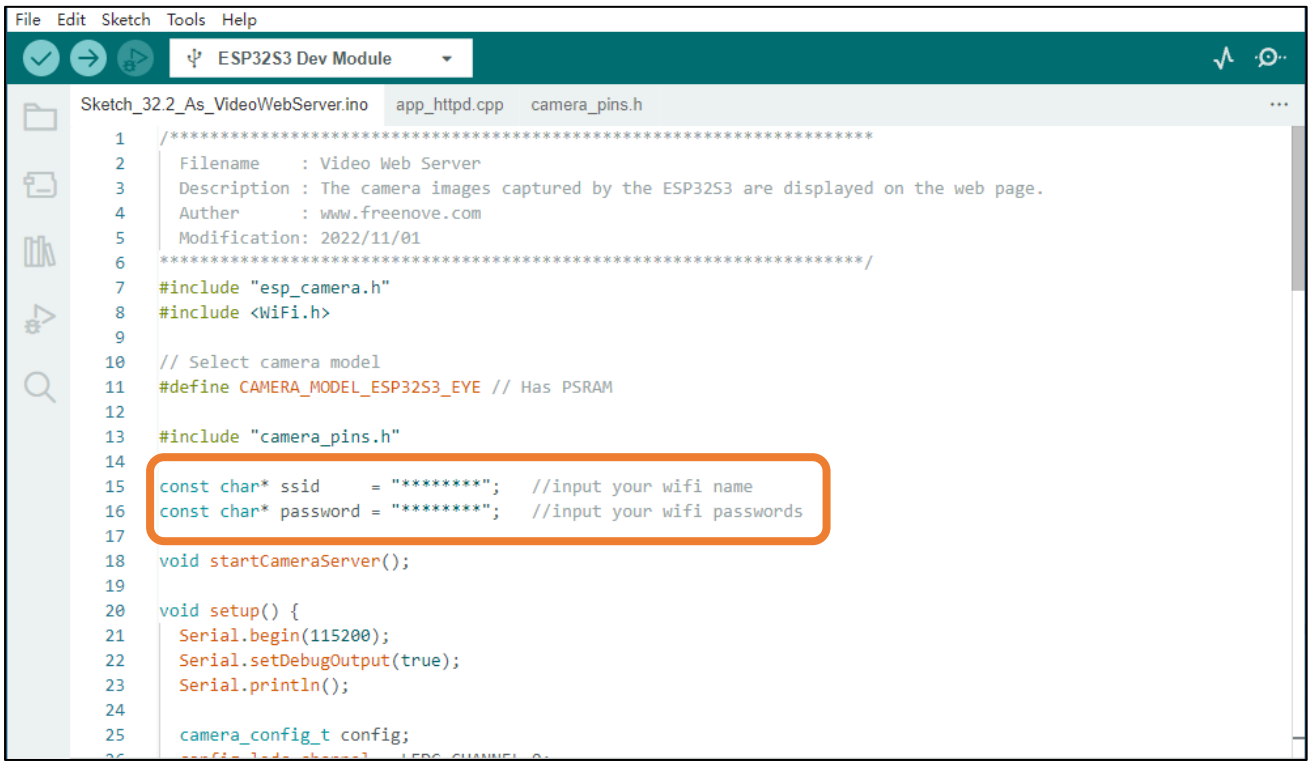
Circuit

Connect Freenove ESP32-S3 to the computer using the USB cable.



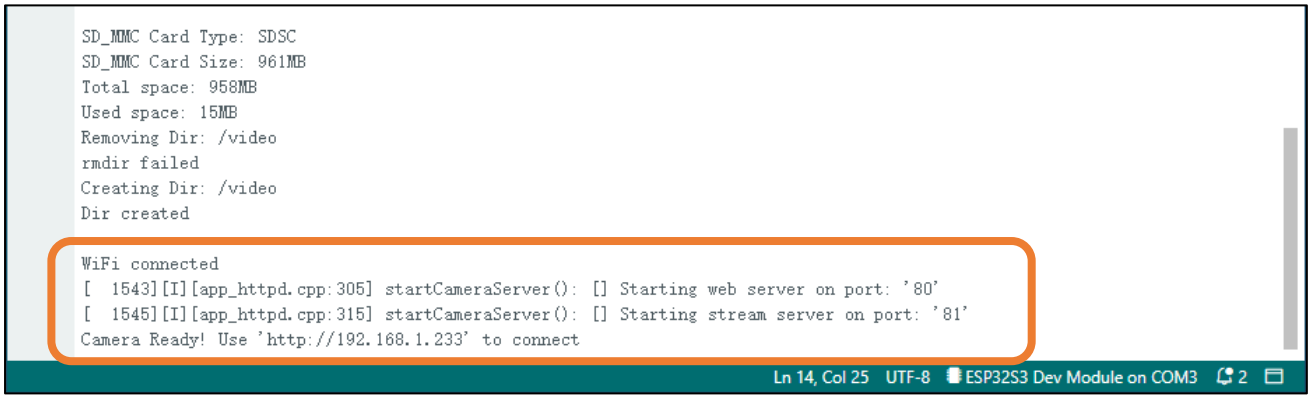
Sketch

Sketch_25.2_As_VideoWebServer



Before running the program, please modify your router's name and password in the box shown in the illustration above to make sure that your Sketch can compile and work successfully.

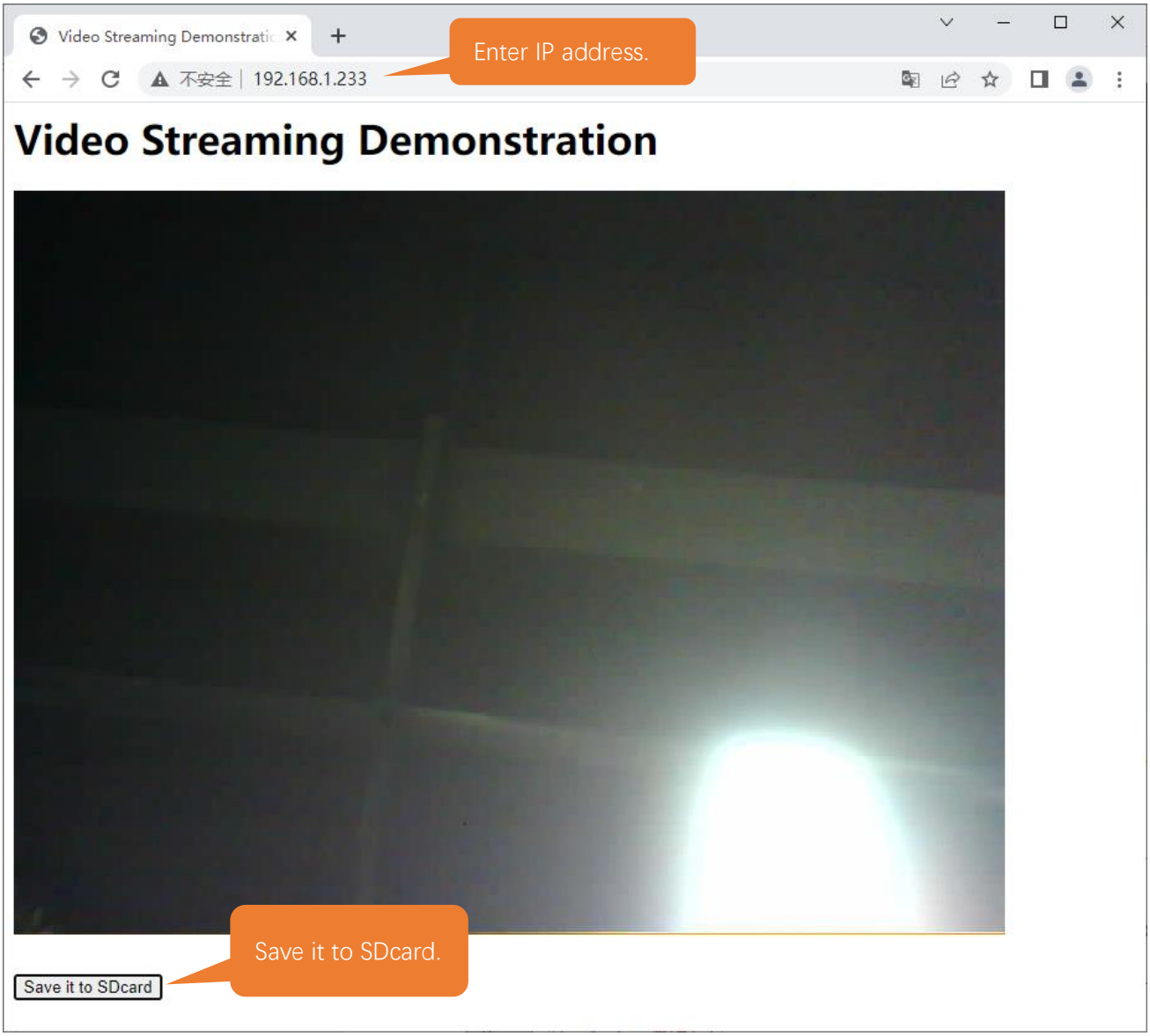
Compile and upload codes to ESP32-S3, open the serial monitor and set the baud rate to 115200, and the serial monitor will print out a network link address.



If your ESP32-S3 has been in the process of connecting to router, but the information above has not been printed out, please re-check whether the router name and password have been entered correctly and press the reset key on ESP32-S3 WROOM to wait for a successful connection prompt.

Open a web browser, enter the IP address printed by the serial monitor in the address bar, and access it. Taking the Google browser as an example, here's what the browser prints out after successful access to ESP32-S3's IP.

The effect is shown in the image below.



The following is the main program code. You need include other code files in the same folder when write your own code.

```
#include "esp_camera.h"
#include <WiFi.h>

// Select camera model
#define CAMERA_MODEL_ESP32S3_EYE // Has PSRAM
#include "camera_pins.h"

```

```
8  const char* ssid     = "********";   //input your wifi name
9  const char* password = "********";   //input your wifi passwords
10 void startCameraServer();
11
12 void setup() {
13   Serial.begin(115200);
14   Serial.setDebugOutput(true);
15   Serial.println();
16
17   camera_config_t config;
18   config.ledc_channel = LEDC_CHANNEL_0;
19   config.ledc_timer = LEDC_TIMER_0;
20   config.pin_d0 = Y2_GPIO_NUM;
21   config.pin_d1 = Y3_GPIO_NUM;
22   config.pin_d2 = Y4_GPIO_NUM;
23   config.pin_d3 = Y5_GPIO_NUM;
24   config.pin_d4 = Y6_GPIO_NUM;
25   config.pin_d5 = Y7_GPIO_NUM;
26   config.pin_d6 = Y8_GPIO_NUM;
27   config.pin_d7 = Y9_GPIO_NUM;
28   config.pin_xclk = XCLK_GPIO_NUM;
29   config.pin_pclk = PCLK_GPIO_NUM;
30   config.pin_vsync = VSYNC_GPIO_NUM;
31   config.pin_href = HREF_GPIO_NUM;
32   config.pin_sccb_sda = SIOD_GPIO_NUM;
33   config.pin_sccb_scl = SIOC_GPIO_NUM;
34   config.pin_pwdn = PWDN_GPIO_NUM;
35   config.pin_reset = RESET_GPIO_NUM;
36   config.xclk_freq_hz = 20000000;
37   config.frame_size = FRAMESIZE_UXGA;
38   config.pixel_format = PIXFORMAT_JPEG; // for streaming
39   config.grab_mode = CAMERA_GRAB_WHEN_EMPTY;
40   config.fb_location = CAMERA_FB_IN_PSRAM;
41   config.jpeg_quality = 12;
42   config.fb_count = 1;
43
44   // if PSRAM IC present, init with UXGA resolution and higher JPEG quality
45   // for larger pre-allocated frame buffer.
46   if(psramFound()){
47     config.jpeg_quality = 10;
48     config.fb_count = 2;
49     config.grab_mode = CAMERA_GRAB_LATEST;
50   } else {
51     // Limit the frame size when PSRAM is not available
```

```
    config.frame_size = FRAMESIZE_SVGA;
    config.fb_location = CAMERA_FB_IN_DRAM;
  }

  // camera init
  esp_err_t err = esp_camera_init(&config);
  if (err != ESP_OK) {
    Serial.printf("Camera init failed with error 0x%x", err);
    return;
  }

  sensor_t * s = esp_camera_sensor_get();
  // initial sensors are flipped vertically and colors are a bit saturated
  s->set_vflip(s, 1); // flip it back
  s->set_brightness(s, 1); // up the brightness just a bit
  s->set_saturation(s, 0); // lower the saturation

  WiFi.begin(ssid, password);

  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
  Serial.println("");
  Serial.println("WiFi connected");

  startCameraServer();

  Serial.print("Camera Ready! Use 'http://");
  Serial.print(WiFi.localIP());
  Serial.println("' to connect");
}

void loop() {
  // put your main code here, to run repeatedly:
  delay(10000);
}
```

Configure parameters including interface pins of the camera. Note: It is generally not recommended to change them.

```
17    camera_config_t config;
18    config.ledc_channel = LEDC_CHANNEL_0;
19    config.ledc_timer = LEDC_TIMER_0;
20    config.pin_d0 = Y2_GPIO_NUM;
21    config.pin_d1 = Y3_GPIO_NUM;
22    config.pin_d2 = Y4_GPIO_NUM;
23    config.pin_d3 = Y5_GPIO_NUM;
24    config.pin_d4 = Y6_GPIO_NUM;
25    config.pin_d5 = Y7_GPIO_NUM;
26    config.pin_d6 = Y8_GPIO_NUM;
27    config.pin_d7 = Y9_GPIO_NUM;
28    config.pin_xclk = XCLK_GPIO_NUM;
29    config.pin_pclk = PCLK_GPIO_NUM;
30    config.pin_vsync = VSYNC_GPIO_NUM;
31    config.pin_href = HREF_GPIO_NUM;
32    config.pin_sccb_sda = SIOD_GPIO_NUM;
33    config.pin_sccb_scl = SIOC_GPIO_NUM;
34    config.pin_pwdn = PWDN_GPIO_NUM;
35    config.pin_reset = RESET_GPIO_NUM;
36    config.xclk_freq_hz = 20000000;
37    config.frame_size = FRAMESIZE_UXGA;
38    config.pixel_format = PIXFORMAT_JPEG; // for streaming
39    config.grab_mode = CAMERA_GRAB_WHEN_EMPTY;
40    config.fb_location = CAMERA_FB_IN_PSRAM;
41    config.jpeg_quality = 12;
42    config.fb_count = 1;
```

ESP32-S3 connects to the router and prints a successful connection prompt. If it has not been successfully connected, press the reset key on the ESP32-S3 WROOM.

```
69    WiFi.begin(ssid, password);
70
71    while (WiFi.status() != WL_CONNECTED) {
72      delay(500);
73      Serial.print(".");
74    }
75    Serial.println("");
76    Serial.println("WiFi connected");
```

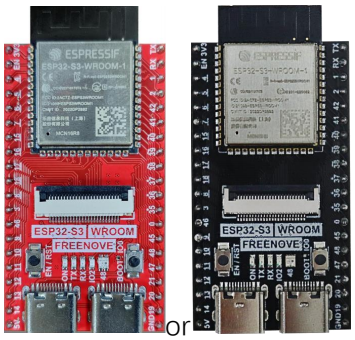
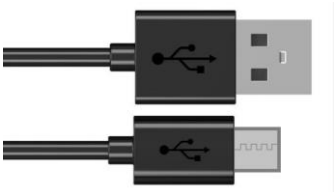
Open the video streams server function of the camera and print its IP address via serial port.

```
82    startCameraServer();
79
80    Serial.print("Camera Ready! Use 'http://");
81    Serial.print(WiFi.localIP());
82    Serial.println("' to connect");
```

Project 25.3 Camera and SDcard

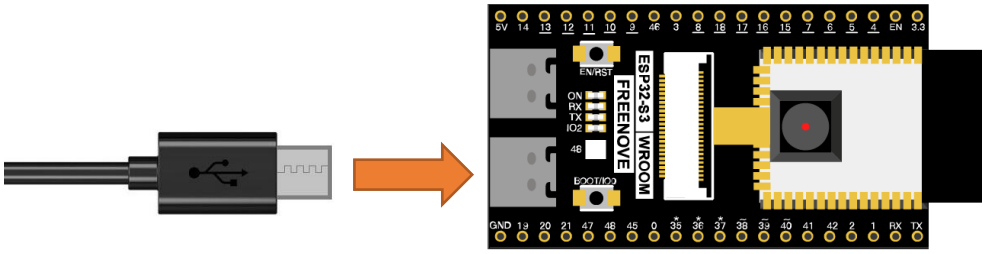
In this chapter, we continue to use the camera and SD card. We will use the onboard button as the shutter. When the button is pressed, the ESP32-S3 takes a photo and stores the photo in the SD folder.

Component List

ESP32-S3(N16R8)/ESP32-S3(N8R8)  or	USB cable x1 	SDcard x1 

Circuit

Connect Freenove ESP32-S3 to the computer using the USB cable.



Sketch

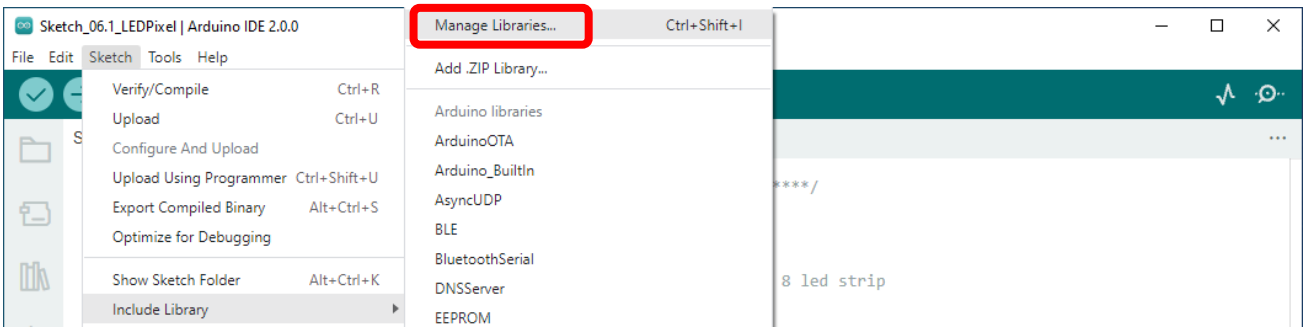
This code uses a library named "Freenove_WS2812_Lib_for_ESP32", if you have not installed it, please do so first.

Library is an important feature of the open source world, and we know that Arduino is an open source platform that everyone can contribute to. Libraries are generally licensed under the LGPL, which means you can use them for free to apply to your creations.

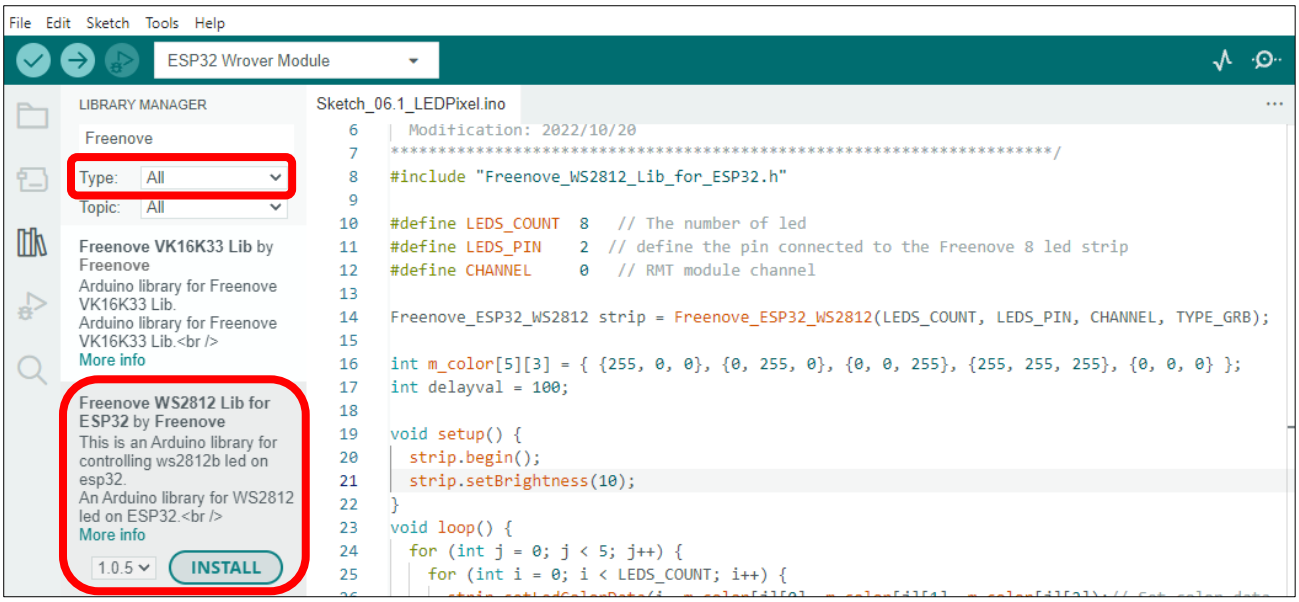
How to install the library

There are two ways to add libraries.

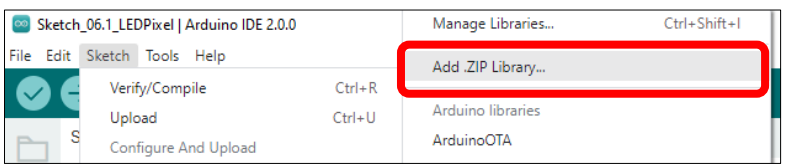
The first way, open the Arduino IDE, click Sketch → Include Library → Manager Libraries.



In the pop-up window, Library Manager, search for the name of the Library, “Freenove WS2812 Lib for ESP32”. Then click Install.



The second way, open Arduino IDE, click Sketch→Include Library→Add .ZIP Library, In the pop-up window, find the file named “./Libraries/Freenove_WS2812_Lib_for_ESP32.Zip” which locates in this directory, and click OPEN.



Sketch_25.3_Camera_SDcard

File Edit Sketch Tools Help

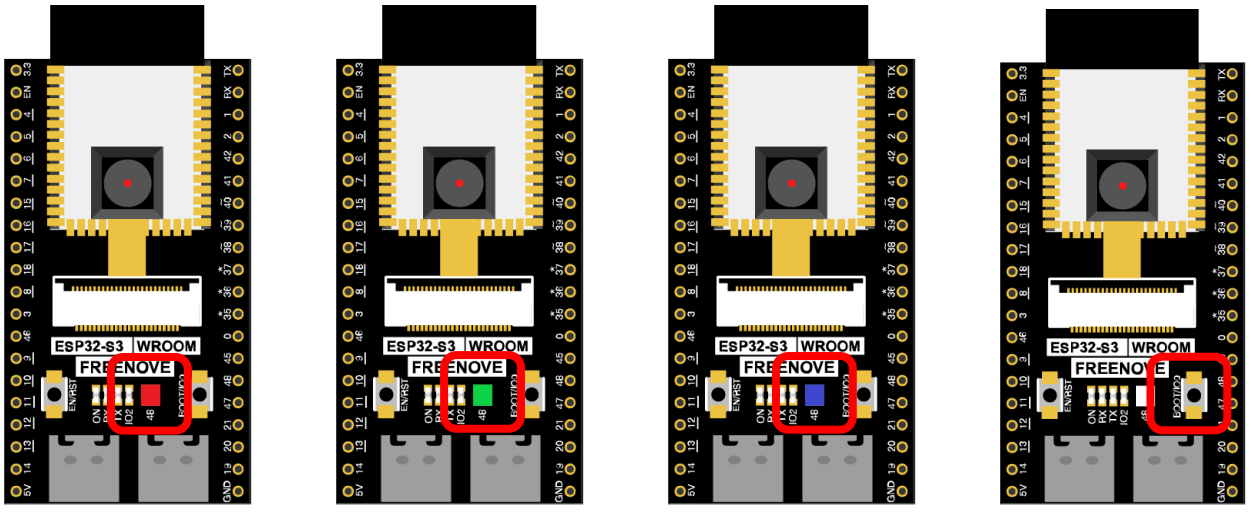
ESP32S3 Dev Module

Sketch_32.3_Camera_SDcard.ino | camera_pins.h | sd_read_write.cpp | sd_read_write.h | ws2812.cpp | ws2812.h

```
/*********************************************************************
  Filename    : Camera and SDcard
  Description : Use the onboard buttons to take photo and save them to an SD card.
  Auther      : www.freenove.com
  Modification: 2022/11/02
*********************************************************************/
#include "esp_camera.h"
#define CAMERA_MODEL_ESP32S3_EYE
#include "camera_pins.h"
#include "ws2812.h"
#include "sd_read_write.h"

#define BUTTON_PIN  0

void setup() {
  Serial.begin(115200);
  Serial.setDebugOutput(false);
  Serial.println();
  pinMode(BUTTON_PIN, INPUT_PULLUP);
  ws2812Init();
  sdmmcInit();
  //removeDir(SD_MMC, "/camera");
  createDir(SD_MMC, "/camera");
```

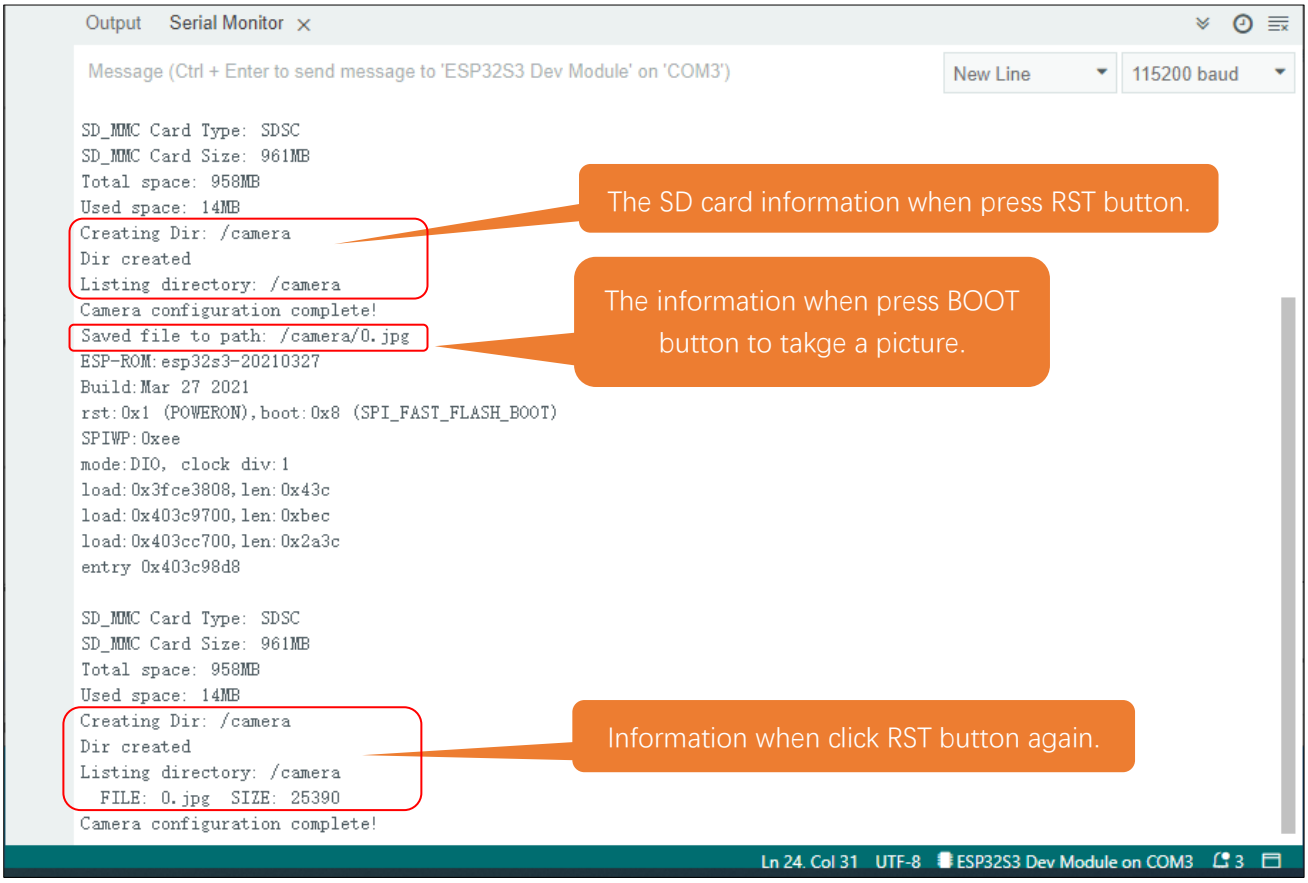
Compile and upload the code to the ESP32-S3.

If your camera is not installed properly, causing the camera to fail to initialize, or you have not inserted the SD card into the ESP32-S3 in advance, the on-board colored lights will turn on red as a reminder. If all is well, the onboard colored light will light up green. When the onboard BOOT button is pressed, the ESP32-S3 will capture the current camera image and save it in the "Camera" folder of the SD card. At the same time, the onboard LED lights up blue, and returns to green after taking a photo.



As shown in the image below, after uploading the code to the ESP32-S3, the ESP32-S3 will automatically create a folder named "camera" in the SD card. Every time the BOOT button is pressed, the on-board colored light turns on blue, and ESP32-S3 collects a photo information and stores it in the "camera" folder. Press the button once to take a photo.

When we press the RST button to reset the ESP32-S3, we can see that there are some photo files in the SD card folder. These photos you can read directly through the card reader.



The following is the main program code. You need include other code files in the same folder when write your own code.

```
#include "esp_camera.h"
#define CAMERA_MODEL_ESP32S3_EYE
#include "camera_pins.h"
#include "ws2812.h"
#include "sd_read_write.h"

#define BUTTON_PIN  0

void setup() {
  Serial.begin(115200);
  Serial.setDebugOutput(false);
  Serial.println();
  pinMode(BUTTON_PIN, INPUT_PULLUP);
  ws2812Init();
  sdmmcInit();
  //removeDir(SD_MMC, "/camera");
  createDir(SD_MMC, "/camera");
  listDir(SD_MMC, "/camera", 0);
  if(cameraSetup()==1){
    ws2812SetColor(2);
  }
  else{
    ws2812SetColor(1);
    return;
  }
}

void loop() {
  if(digitalRead(BUTTON_PIN)==LOW){
    delay(20);
    if(digitalRead(BUTTON_PIN)==LOW){
      ws2812SetColor(3);
      while(digitalRead(BUTTON_PIN)==LOW);
      camera_fb_t * fb = NULL;
      fb = esp_camera_fb_get();
      if (fb != NULL) {
       int photo_index = readFileNum(SD_MMC, "/camera");
        if(photo_index!=-1)
        {
          String path = "/camera/" + String(photo_index) +".jpg";
          writejpg(SD_MMC, path.c_str(), fb->buf, fb->len);
        }
```

```
43          esp_camera_fb_return(fb);
44        }
45        else {
46          Serial.println("Camera capture failed.");
47        }
48        ws2812SetColor(2);
49      }
50    }
51  }
52
53  int cameraSetup(void) {
54    camera_config_t config;
55    config.ledc_channel = LEDC_CHANNEL_0;
56    config.ledc_timer = LEDC_TIMER_0;
57    config.pin_d0 = Y2_GPIO_NUM;
58    config.pin_d1 = Y3_GPIO_NUM;
59    config.pin_d2 = Y4_GPIO_NUM;
60    config.pin_d3 = Y5_GPIO_NUM;
61    config.pin_d4 = Y6_GPIO_NUM;
62    config.pin_d5 = Y7_GPIO_NUM;
63    config.pin_d6 = Y8_GPIO_NUM;
64    config.pin_d7 = Y9_GPIO_NUM;
65    config.pin_xclk = XCLK_GPIO_NUM;
66    config.pin_pclk = PCLK_GPIO_NUM;
67    config.pin_vsync = VSYNC_GPIO_NUM;
68    config.pin_href = HREF_GPIO_NUM;
69    config.pin_sccb_sda = SIOD_GPIO_NUM;
70    config.pin_sccb_scl = SIOC_GPIO_NUM;
71    config.pin_pwdn = PWDN_GPIO_NUM;
72    config.pin_reset = RESET_GPIO_NUM;
73    config.xclk_freq_hz = 20000000;
74    config.frame_size = FRAMESIZE_UXGA;
75    config.pixel_format = PIXFORMAT_JPEG; // for streaming
76    config.grab_mode = CAMERA_GRAB_WHEN_EMPTY;
77    config.fb_location = CAMERA_FB_IN_PSRAM;
78    config.jpeg_quality = 12;
79    config.fb_count = 1;
80
81    // if PSRAM IC present, init with UXGA resolution and higher JPEG quality
82    // for larger pre-allocated frame buffer.
83    if(psramFound()){
84      config.jpeg_quality = 10;
85      config.fb_count = 2;
86      config.grab_mode = CAMERA_GRAB_LATEST;
```

```
87    } else {
88      // Limit the frame size when PSRAM is not available
89      config.frame_size = FRAMESIZE_SVGA;
90      config.fb_location = CAMERA_FB_IN_DRAM;
91    }
92
93    // camera init
94    esp_err_t err = esp_camera_init(&config);
95    if (err != ESP_OK) {
96      Serial.printf("Camera init failed with error 0x%x", err);
97      return 0;
98    }
99
100   sensor_t * s = esp_camera_sensor_get();
101   // initial sensors are flipped vertically and colors are a bit saturated
102   s->set_vflip(s, 1); // flip it back
103   s->set_brightness(s, 1); // up the brightness just a bit
104   s->set_saturation(s, 0); // lower the saturation
105
106   Serial.println("Camera configuration complete!");
107   return 1;
108 }
```

Configure camera parameters, including camera interface pins and other information. Altering them is generally not recommended. Returns 1 if the camera is initialized successfully, and returns 0 if it fails.

```
53  int cameraSetup(void) {
54    camera_config_t config;
55    config.ledc_channel = LEDC_CHANNEL_0;
56    config.ledc_timer = LEDC_TIMER_0;
57    config.pin_d0 = Y2_GPIO_NUM;
58    config.pin_d1 = Y3_GPIO_NUM;
59    config.pin_d2 = Y4_GPIO_NUM;
60    config.pin_d3 = Y5_GPIO_NUM;
61    config.pin_d4 = Y6_GPIO_NUM;
62    config.pin_d5 = Y7_GPIO_NUM;
63    config.pin_d6 = Y8_GPIO_NUM;
64    config.pin_d7 = Y9_GPIO_NUM;
65    config.pin_xclk = XCLK_GPIO_NUM;
66    config.pin_pclk = PCLK_GPIO_NUM;
67    config.pin_vsync = VSYNC_GPIO_NUM;
68    config.pin_href = HREF_GPIO_NUM;
69    config.pin_sccb_sda = SIOD_GPIO_NUM;
70    config.pin_sccb_scl = SIOC_GPIO_NUM;
71    config.pin_pwdn = PWDN_GPIO_NUM;
72    config.pin_reset = RESET_GPIO_NUM;
```

```
  config.xclk_freq_hz = 20000000;
  config.frame_size = FRAMESIZE_UXGA;
  config.pixel_format = PIXFORMAT_JPEG; // for streaming
  config.grab_mode = CAMERA_GRAB_WHEN_EMPTY;
  config.fb_location = CAMERA_FB_IN_PSRAM;
  config.jpeg_quality = 12;
  config.fb_count = 1;

  // if PSRAM IC present, init with UXGA resolution and higher JPEG quality
  // for larger pre-allocated frame buffer.
  if(psramFound()){
    config.jpeg_quality = 10;
    config.fb_count = 2;
    config.grab_mode = CAMERA_GRAB_LATEST;
  } else {
    // Limit the frame size when PSRAM is not available
    config.frame_size = FRAMESIZE_SVGA;
    config.fb_location = CAMERA_FB_IN_DRAM;
  }

  // camera init
  esp_err_t err = esp_camera_init(&config);
  if (err != ESP_OK) {
    Serial.printf("Camera init failed with error 0x%x", err);
    return 0;
  }

  sensor_t * s = esp_camera_sensor_get();
  // initial sensors are flipped vertically and colors are a bit saturated
  s->set_vflip(s, 1); // flip it back
  s->set_brightness(s, 1); // up the brightness just a bit
  s->set_saturation(s, 0); // lower the saturation

  Serial.println("Camera configuration complete!");
  return 1;
}
```

Initialize the serial port, buttons, lights and SD card.

```
10    Serial.begin(115200);
11    Serial.setDebugOutput(false);
12    Serial.println();
13    pinMode(BUTTON_PIN, INPUT_PULLUP);
14    ws2812Init();
15    sdmmcInit();
```

Call ws2812SetColor() to set the color of the LED. When the parameter is 0, the LED is turned off, when the parameter is 1, the red light is displayed, when the parameter is 2, the green light is displayed, and when the parameter is 3, the blue light is displayed.

```
20    ws2812SetColor(2);
```

Get the camera data once, then read the file number in the camera folder of the SD card, and create a new file based on this, write the camera data into it, and finally return the camera structure pointer. If the camera data cannot be obtained, the prompt information will be printed directly.

```
34        camera_fb_t * fb = NULL;
35        fb = esp_camera_fb_get();
36        if (fb != NULL) {
37         int photo_index = readFileNum(SD_MMC, "/camera");
38          if(photo_index!=-1)
39          {
40            String path = "/camera/" + String(photo_index) +".jpg";
41            writejpg(SD_MMC, path.c_str(), fb->buf, fb->len);
42          }
43          esp_camera_fb_return(fb);
44        }
45        else {
46          Serial.println("Camera capture failed.");
47        }
```

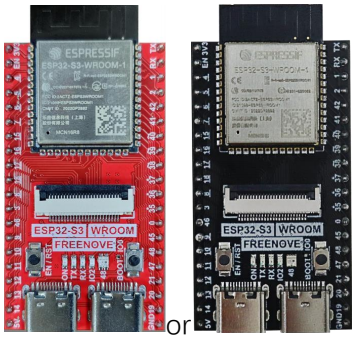
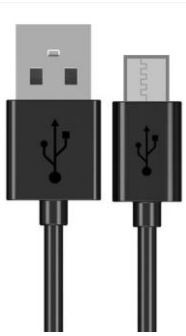
Chapter 26 Camera Tcp Server

In the previous section, we used web page to display the video data captured by ESP32-S3, and in this section, we will use a mobile phone to display it.

Project 26.1 Camera Tcp Server

Connect ESP32-S3 using USB and check its IP address through serial monitor. Use a mobile phone to obtain video and image data.

Component List

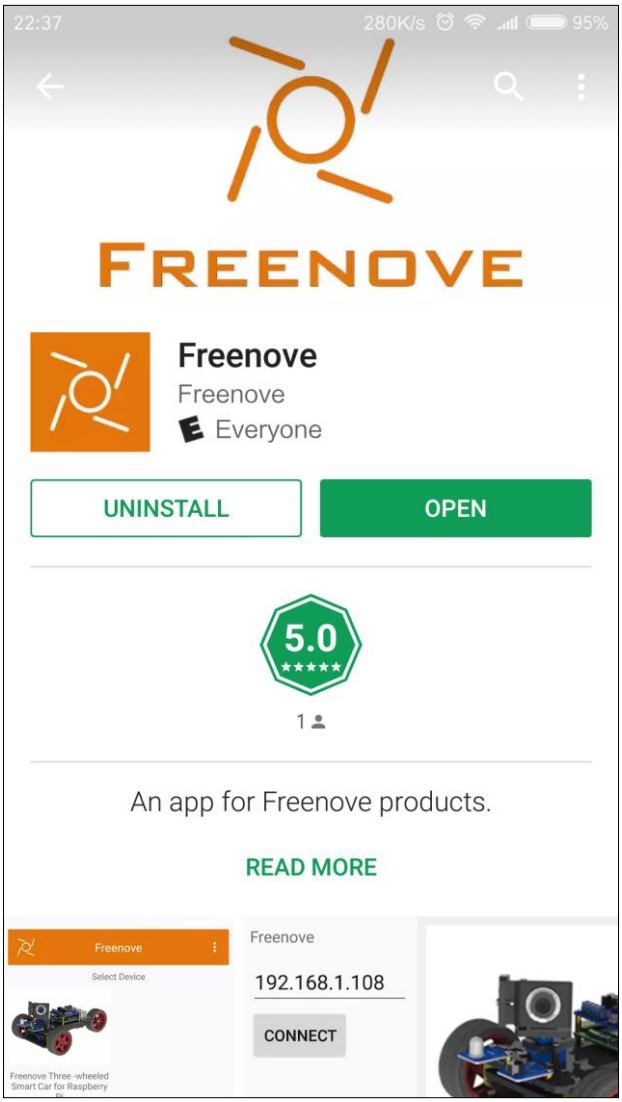
ESP32-S3(N16R8)/ESP32-S3(N8R8)  or	USB cable x1 

Install Freenove app

There are three ways to install app, you can choose any one.

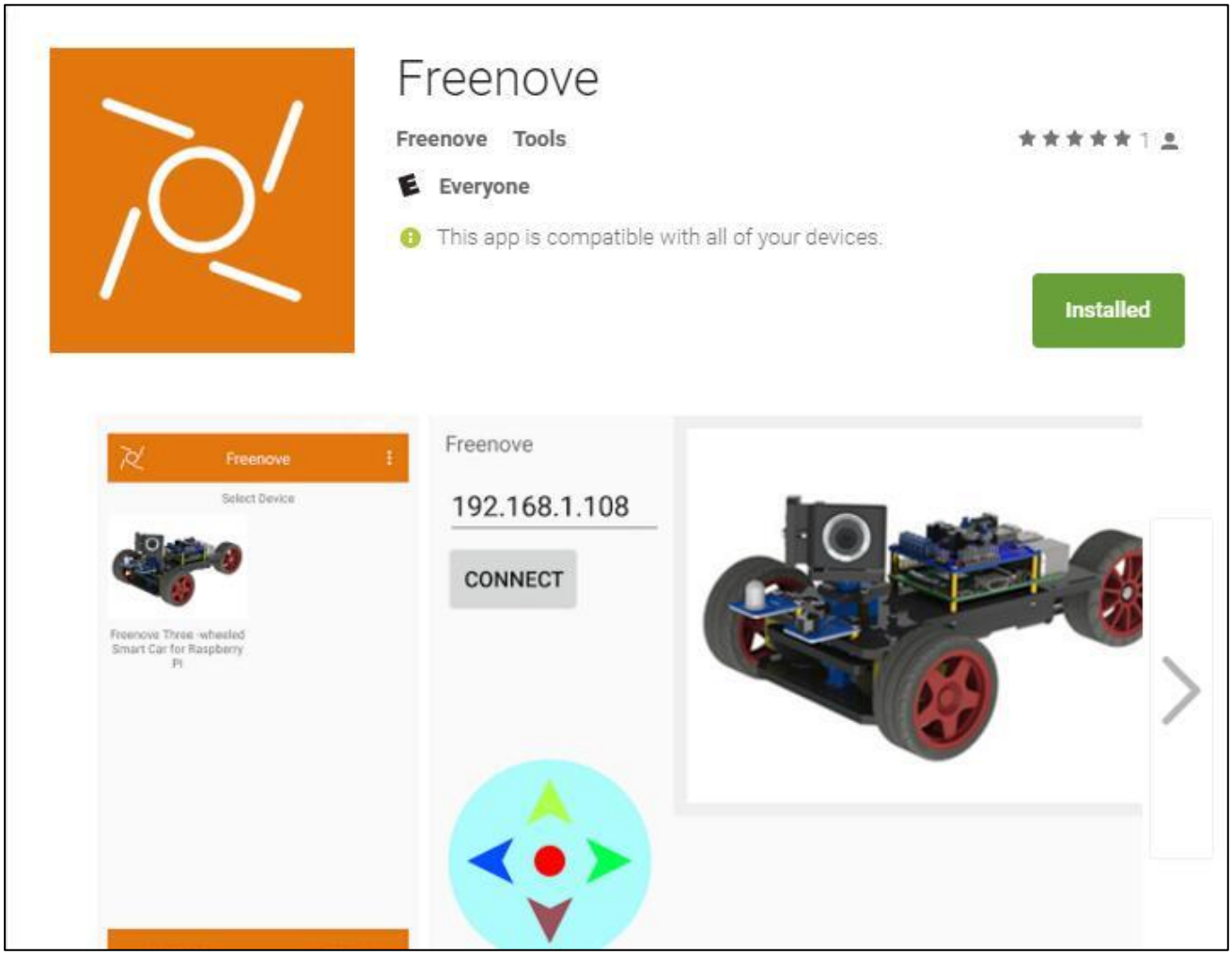
Method 1

Use Google play to search “Freenove”, download and install.


22:37
280K/s
95%
FREENOVE
Freenove
Freenove
Everyone
UNINSTALL
OPEN
5.0
1
An app for Freenove products.
READ MORE
Freenove
Select Device
Freenove Three-wheeled Smart Car for Raspberry
Freenove
192.168.1.108
CONNECT

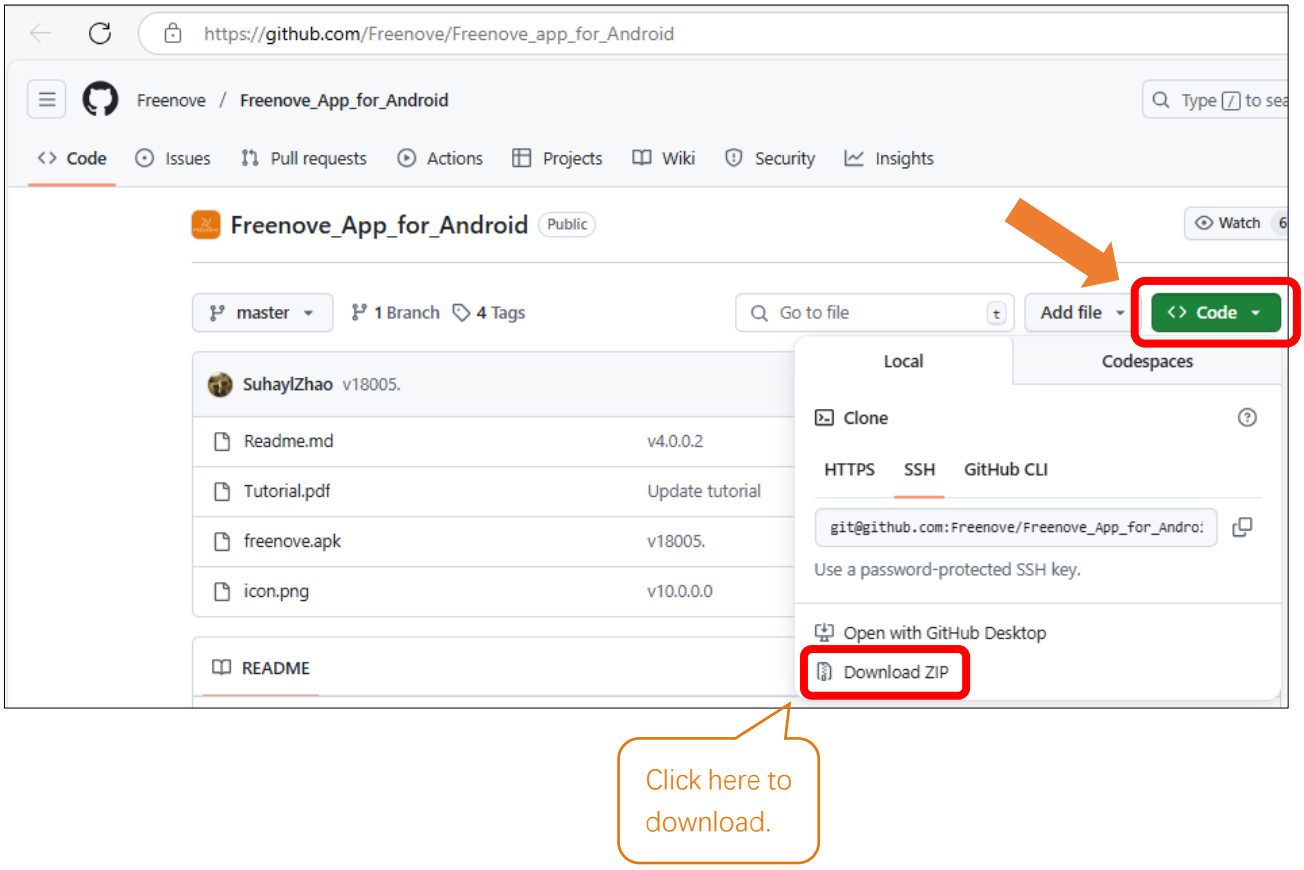
Method 2

Visit https://play.google.com/store/apps/details?id=com.freenove.suhayl.Freenove, and click install.



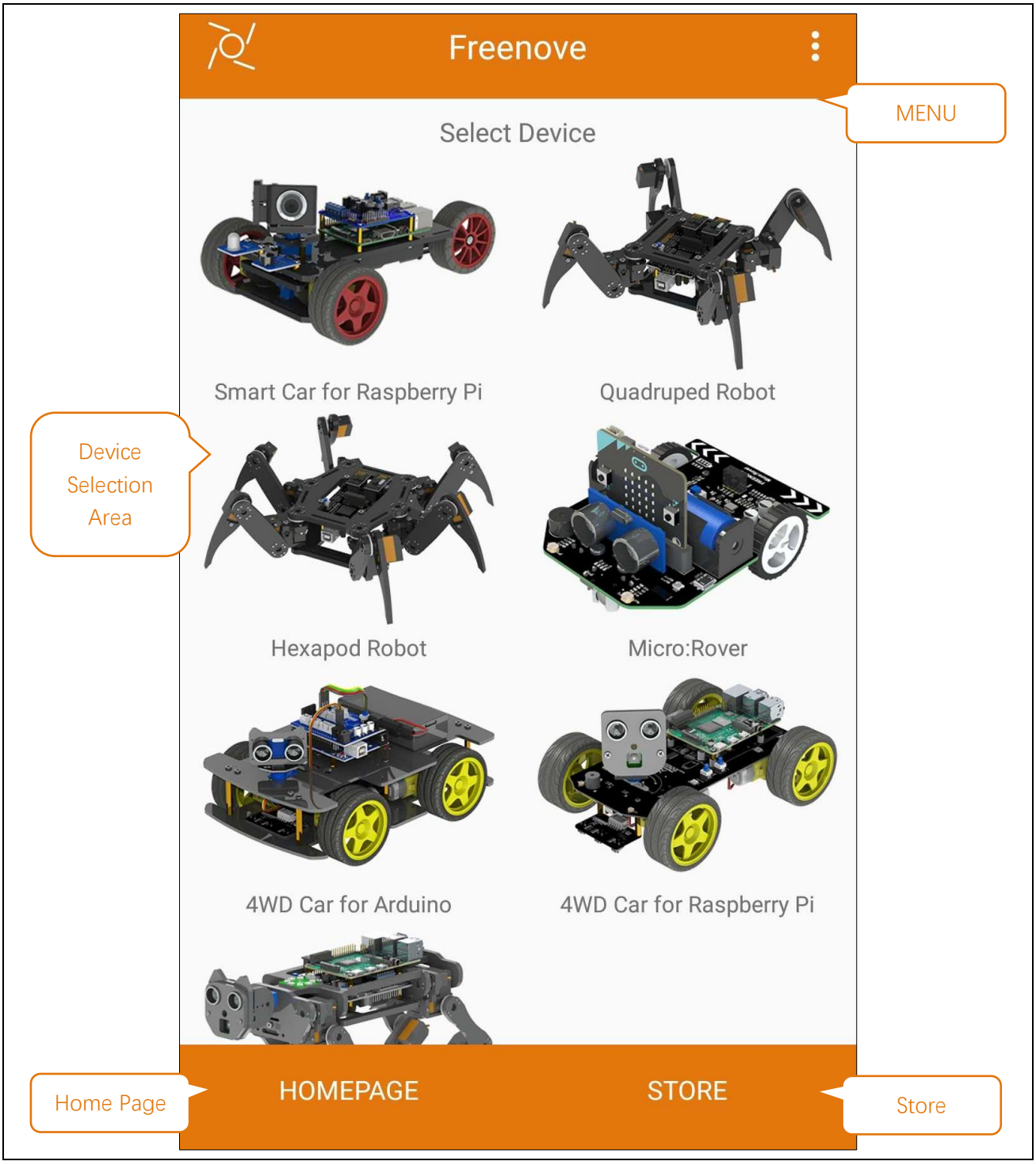
Method 3

Visit https://github.com/Freenove/Freenove_app_for_Android, download the files in this library, and install freenove.apk to your Android phone manually.



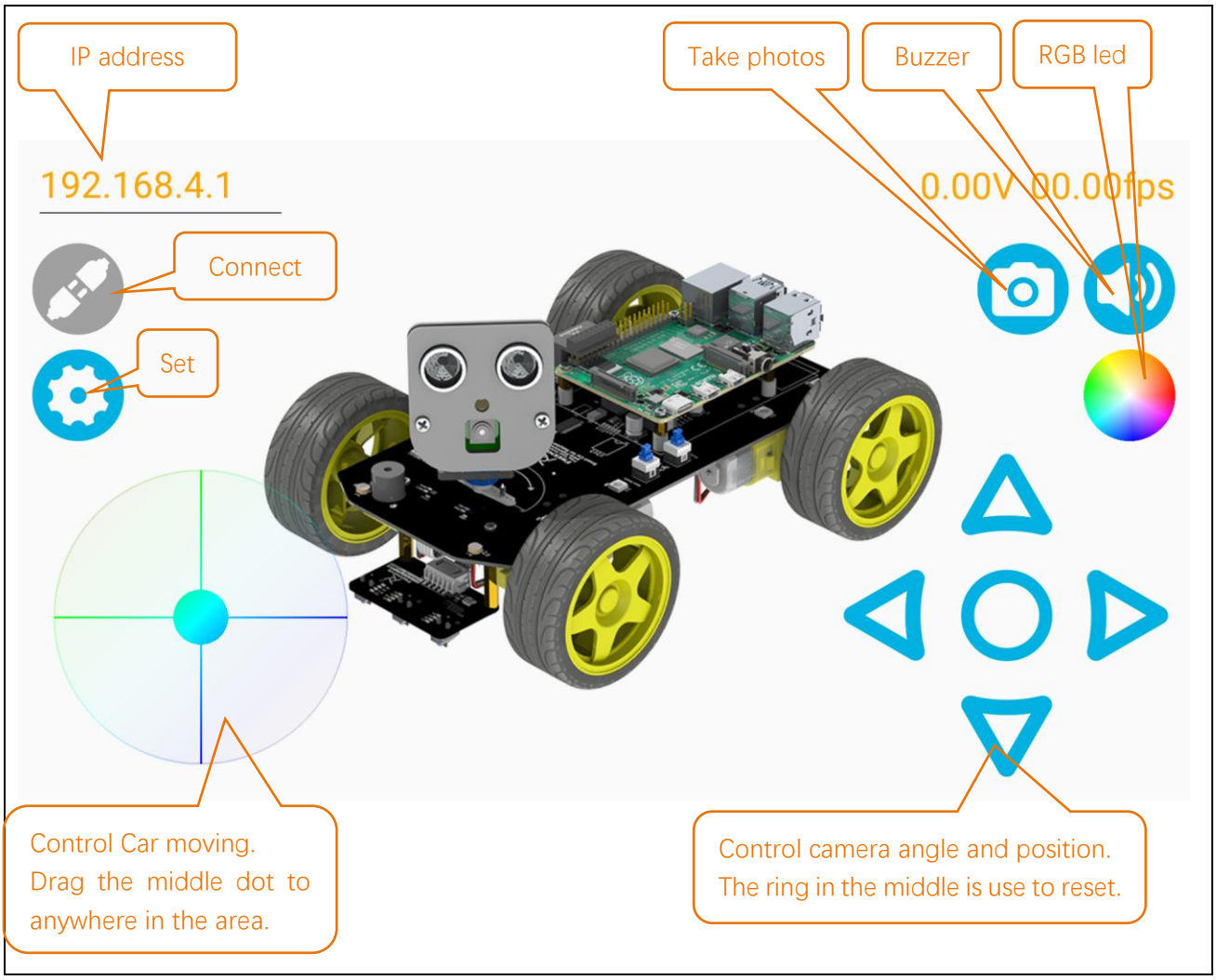
Menu

Open application “Freenove”, as shown below:



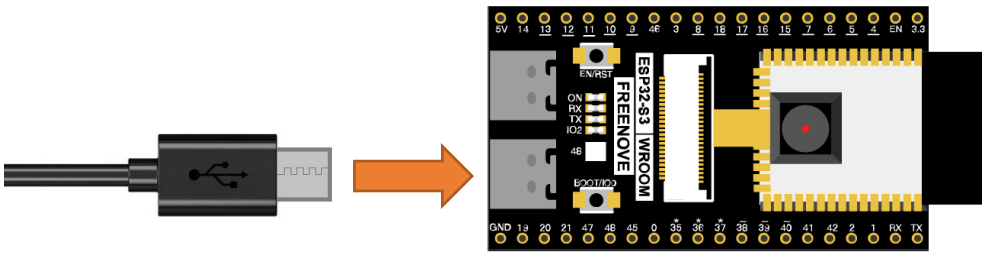
Freenove 4WD Car for Raspberry Pi

In this chapter, we use Freenove 4WD Car for Raspberry Pi, so it is necessary to understand the interface of this mode.



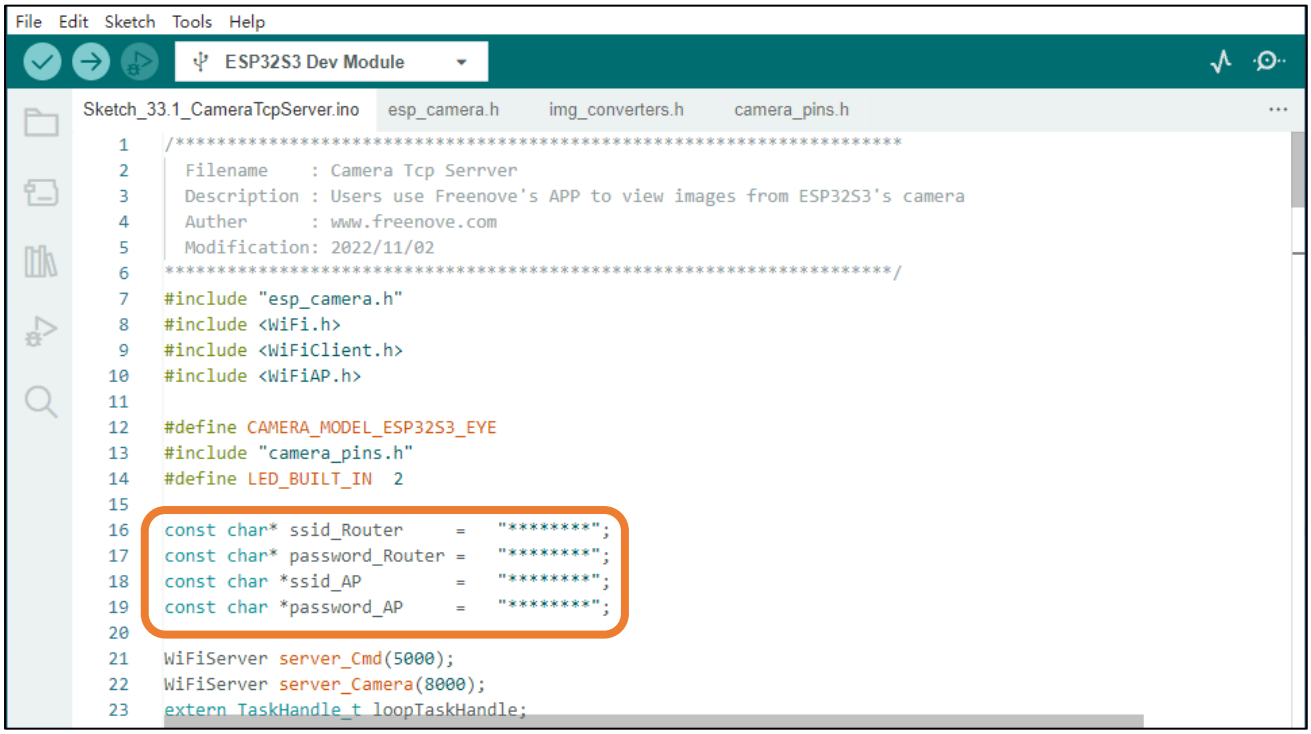
Circuit

Connect Freenove ESP32-S3 to the computer using the USB cable.



Sketch

After making sure the Tools is configured correctly, don't run Sketch. Due to WiFi, we need to modify Sketch a little bit based on physical situation.

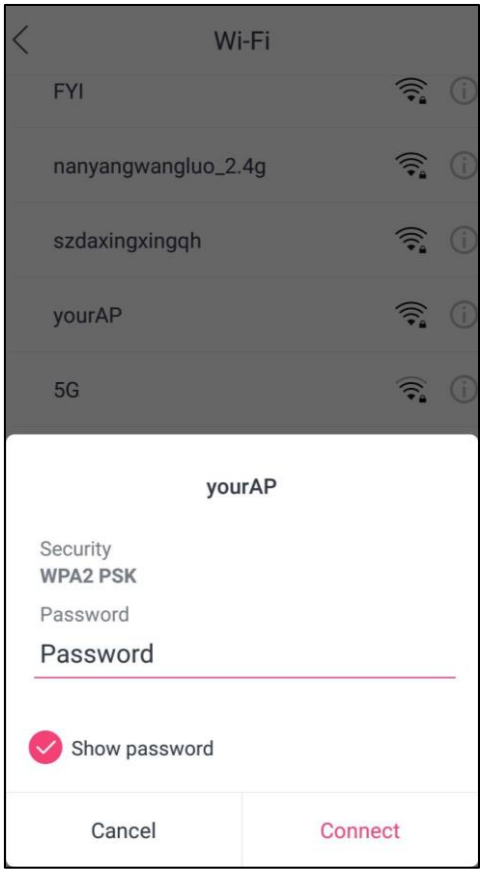


In the box in the figure above, ssid_Router and password_Router are the user's Router name and password, which need to be modified according to the actual name and password. ssid_AP and password_AP are name and password of a AP created by ESP32-S3, and they are freely set by the user. When all settings are correct, compile and upload the code to ESP32-S3, turn on the serial port monitor, and set the baud rate to 115200. The serial monitor will print out two IP addresses.

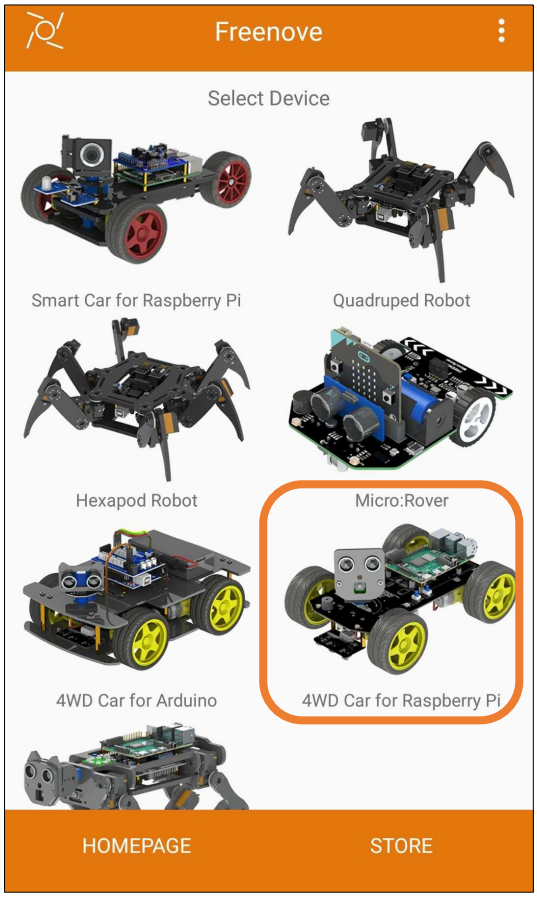
There are two methods for you to check camera data of ESP32-S3 via mobile phone APP.

Method 1:

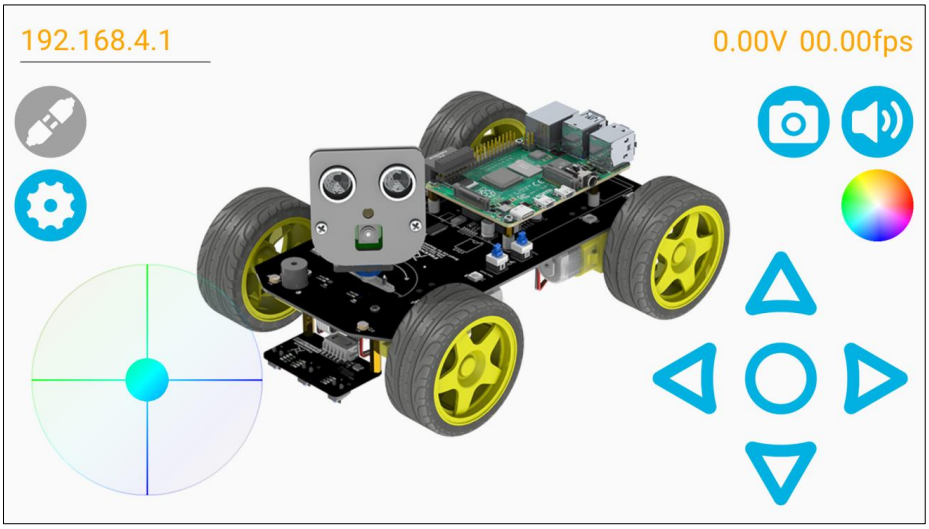
Using your phone's WiFi function, select the WiFi name represented by ssid_AP in Sketch and enter the password “password_AP” to connect.



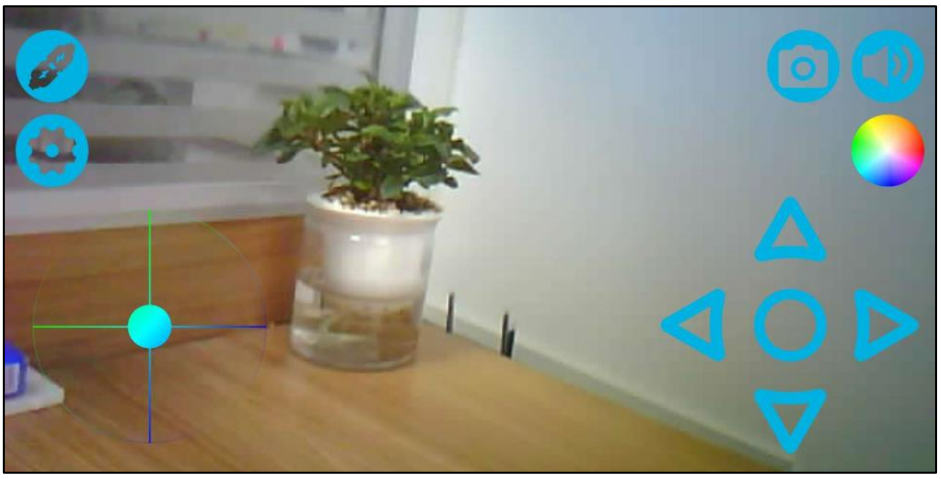
Next, open Freenove app and select 4WD Car for Raspberry Pi mode.



Enter the IP address printed by serial port in the new interface, which generally is “192.168.4.1”



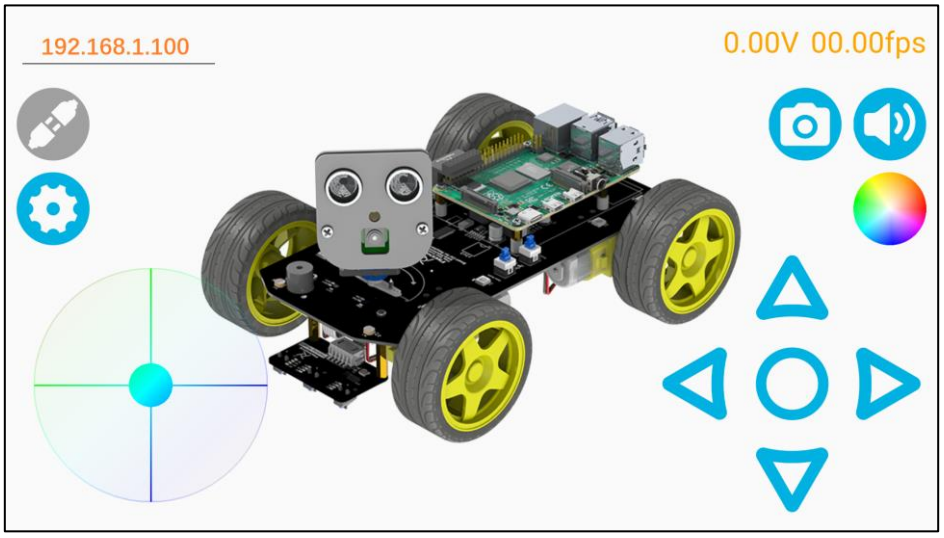
Click “Connect”.



Method 2:

Using your phone's WiFi function, select the router named ssid_Router and enter the password “ssid_password” to connect. And then open Freenove app and select 4WD Car for Raspberry Pi mode. The operation is similar to Method 1.

Enter the IP address printed by serial port in the new interface, which generally is not “192.168.4.1” but another one. The IP address in this example is “192.168.1.100”. After entering the IP address, click “Connect”.



The following is the main program code. You need include other code files in the same folder when write your own code.

Sketch_26.1_Camera_Tcp_Server

```
#include "esp_camera.h"
#include <WiFi.h>
#include <WiFiClient.h>
#include <WiFiAP.h>

#define CAMERA_MODEL_ESP32S3_EYE
```

```
#include "camera_pins.h"
#define LED_BUILT_IN  2

const char *ssid_Router     =   "********";
const char *password_Router =   "********";
const char *ssid_AP         =   "********";
const char *password_AP     =   "********";

WiFiServer server_Cmd(5000);
WiFiServer server_Camera(8000);
extern TaskHandle_t loopTaskHandle;

void setup() {
  Serial.begin(115200);
  Serial.setDebugOutput(false);
  Serial.println();
  pinMode(LED_BUILT_IN, OUTPUT);
  cameraSetup();

  WiFi.softAP(ssid_AP, password_AP);
  IPAddress myIP = WiFi.softAPIP();
  Serial.print("AP IP address: ");
  Serial.println(myIP);
  server_Camera.begin(8000);
  server_Cmd.begin(5000);
  ////////////////////////////////////////////////
  WiFi.begin(ssid_Router, password_Router);
  Serial.print("Connecting ");
  Serial.print(ssid_Router);
  while (WiFi.isConnected() != true) {
    delay(500);
    Serial.print(".");
  }
  while (WiFi.STA.hasIP() != true) {
    Serial.print(".");
    delay(500);
  }
  Serial.println("");
  Serial.println("WiFi connected");
  ////////////////////////////////////////////////
  Serial.print("Camera Ready! Use '");
  Serial.print(WiFi.softAPIP());
  Serial.print(" or ");
  Serial.print(WiFi.localIP());
```

```
  Serial.println("' to connect in Freenove app.");
  disableCoreOWDT();
  xTaskCreateUniversal(loopTask_Cmd, "loopTask_Cmd", 8192, NULL, 1, &loopTaskHandle,
0);//loopTask_Cmd uses core 0.
  xTaskCreateUniversal(loopTask_Blink, "loopTask_Blink", 8192, NULL, 1, &loopTaskHandle,
0);//loopTask_Blink uses core 0.
}
//task loop uses core 1.
void loop() {
  WiFiClient client = server_Camera.available();           // listen for incoming clients
  if (client) {                                            // if you get a client,
    Serial.println("Camera Server connected to a client.");// print a message out the serial
port
    String currentLine = "";     // make a String to hold incoming data from the client
    while (client.connected()) { // loop while the client's connected
      camera_fb_t * fb = NULL;
      while (client.connected()) {
        fb = esp_camera_fb_get();
        if (fb != NULL) {
          uint8_t slen[4];
          slen[0] = fb->len >> 0;
          slen[1] = fb->len >> 8;
          slen[2] = fb->len >> 16;
          slen[3] = fb->len >> 24;
          client.write(slen, 4);
          client.write(fb->buf, fb->len);
          esp_camera_fb_return(fb);
        }
        else {
          Serial.println("Camera Error");
        }
      }
    }
    // close the connection:
    client.stop();
    Serial.println("Camera Client Disconnected.");
  }
}

void loopTask_Cmd(void *pvParameters) {
  Serial.println("Task Cmd_Server is starting ... ");
  while (1) {
    WiFiClient client = server_Cmd.available(); // listen for incoming clients
```

```
92      if (client) {                               // if you get a client,
93        Serial.println("Command Server connected to a client.");// print a message out the
    serial port
94        String currentLine = "";          // make a String to hold incoming data from the client
95        while (client.connected()) {      // loop while the client's connected
96          if (client.available()) {       // if there's bytes to read from the client,
97            char c = client.read();       // read a byte, then
98            client.write(c);
99            Serial.write(c);              // print it out the serial monitor
100           if (c == '\n') {              // if the byte is a newline character
101             currentLine = "";
102           }
103           else {
104             currentLine += c;           // add it to the end of the currentLine
105           }
106         }
107       }
108       // close the connection:
109       client.stop();
110       Serial.println("Command Client Disconnected.");
111     }
112   }
113 }
114 void loopTask_Blink(void *pvParameters) {
115   Serial.println("Task Blink is starting ... ");
116   while (1) {
117     digitalWrite(LED_BUILT_IN, !digitalRead(LED_BUILT_IN));
118     delay(1000);
119   }
120 }
121
122 void cameraSetup() {
123   camera_config_t config;
124   config.ledc_channel = LEDC_CHANNEL_0;
125   config.ledc_timer = LEDC_TIMER_0;
126   config.pin_d0 = Y2_GPIO_NUM;
127   config.pin_d1 = Y3_GPIO_NUM;
128   config.pin_d2 = Y4_GPIO_NUM;
129   config.pin_d3 = Y5_GPIO_NUM;
130   config.pin_d4 = Y6_GPIO_NUM;
131   config.pin_d5 = Y7_GPIO_NUM;
132   config.pin_d6 = Y8_GPIO_NUM;
133   config.pin_d7 = Y9_GPIO_NUM;
134   config.pin_xclk = XCLK_GPIO_NUM;
```

```
135   config.pin_pclk = PCLK_GPIO_NUM;
136   config.pin_vsync = VSYNC_GPIO_NUM;
137   config.pin_href = HREF_GPIO_NUM;
138   config.pin_sccb_sda = SIOD_GPIO_NUM;
139   config.pin_sccb_scl = SIOC_GPIO_NUM;
140   config.pin_pwdn = PWDN_GPIO_NUM;
141   config.pin_reset = RESET_GPIO_NUM;
142   config.xclk_freq_hz = 20000000;
143   config.frame_size = FRAMESIZE_UXGA;
144   config.pixel_format = PIXFORMAT_JPEG; // for streaming
145   config.grab_mode = CAMERA_GRAB_WHEN_EMPTY;
146   config.fb_location = CAMERA_FB_IN_PSRAM;
147   config.jpeg_quality = 12;
148   config.fb_count = 1;
149
150   // if PSRAM IC present, init with UXGA resolution and higher JPEG quality
151   // for larger pre-allocated frame buffer.
152   if(psramFound()){
153     config.jpeg_quality = 10;
154     config.fb_count = 2;
155     config.grab_mode = CAMERA_GRAB_LATEST;
156   } else {
157     // Limit the frame size when PSRAM is not available
158     config.frame_size = FRAMESIZE_SVGA;
159     config.fb_location = CAMERA_FB_IN_DRAM;
160   }
161
162   // camera init
163   esp_err_t err = esp_camera_init(&config);
164   if (err != ESP_OK) {
165     Serial.printf("Camera init failed with error 0x%x", err);
166     return;
167   }
168
169   sensor_t * s = esp_camera_sensor_get();
170   // initial sensors are flipped vertically and colors are a bit saturated
171   s->set_vflip(s, 1); // flip it back
172   s->set_brightness(s, 1); // up the brightness just a bit
173   s->set_saturation(s, 0); // lower the saturation
174
175   Serial.println("Camera configuration complete!");
176 }
```

Include header files that drive camera and WiFi.

```
#include "esp_camera.h"
#include <WiFi.h>
#include <WiFiClient.h>
#include <WiFiAP.h>

#define CAMERA_MODEL_ESP32S3_EYE
#include "camera_pins.h"
```

Set name and password for router that ESP32-S3 needs to connect to. And set ESP32-S3 to open two servers, whose port are 8000 and 5000 respectively.

```
const char *ssid_Router     =   "********";
const char *password_Router =   "********";
const char *ssid_AP         =   "********";
const char *password_AP     =   "********";
```

Enable ESP32-S3's server function and set two monitor ports as 5000 and 8000. In general, the two port numbers do not require modifications.

```
WiFiServer server_Cmd(5000);
WiFiServer server_Camera(8000);
extern TaskHandle_t loopTaskHandle;
```

Initialize serial port, set baud rate to 115200; open the debug and output function of the serial.

```
  Serial.begin(115200);
  Serial.setDebugOutput(true);
  Serial.println();
```

Define a variable for camera interface and initialize it.

```
void cameraSetup() {
  camera_config_t config;
  config.ledc_channel = LEDC_CHANNEL_0;
  config.ledc_timer = LEDC_TIMER_0;
  config.pin_d0 = Y2_GPIO_NUM;
  config.pin_d1 = Y3_GPIO_NUM;
  config.pin_d2 = Y4_GPIO_NUM;
  config.pin_d3 = Y5_GPIO_NUM;
  config.pin_d4 = Y6_GPIO_NUM;
  config.pin_d5 = Y7_GPIO_NUM;
  config.pin_d6 = Y8_GPIO_NUM;
  config.pin_d7 = Y9_GPIO_NUM;
  config.pin_xclk = XCLK_GPIO_NUM;
  config.pin_pclk = PCLK_GPIO_NUM;
  config.pin_vsync = VSYNC_GPIO_NUM;
  config.pin_href = HREF_GPIO_NUM;
  config.pin_sccb_sda = SIOD_GPIO_NUM;
  config.pin_sccb_scl = SIOC_GPIO_NUM;
  config.pin_pwdn = PWDN_GPIO_NUM;
  config.pin_reset = RESET_GPIO_NUM;
```

```
139    config.xclk_freq_hz = 20000000;
140    config.frame_size = FRAMESIZE_UXGA;
141    config.pixel_format = PIXFORMAT_JPEG; // for streaming
142    config.grab_mode = CAMERA_GRAB_WHEN_EMPTY;
143    config.fb_location = CAMERA_FB_IN_PSRAM;
144    config.jpeg_quality = 12;
145    config.fb_count = 1;
146
147    // if PSRAM IC present, init with UXGA resolution and higher JPEG quality
148    // for larger pre-allocated frame buffer.
149    if(psramFound()){
150      config.jpeg_quality = 10;
151      config.fb_count = 2;
152      config.grab_mode = CAMERA_GRAB_LATEST;
153    } else {
154      // Limit the frame size when PSRAM is not available
155      config.frame_size = FRAMESIZE_SVGA;
156      config.fb_location = CAMERA_FB_IN_DRAM;
157    }
158
159    // camera init
160    esp_err_t err = esp_camera_init(&config);
161    if (err != ESP_OK) {
162      Serial.printf("Camera init failed with error 0x%x", err);
163      return;
164    }
165
166    sensor_t * s = esp_camera_sensor_get();
167    // initial sensors are flipped vertically and colors are a bit saturated
168    s->set_vflip(s, 1); // flip it back
169    s->set_brightness(s, 1); // up the brightness just a bit
170    s->set_saturation(s, 0); // lower the saturation
171
172    Serial.println("Camera configuration complete!");
173  }
```

Loop function will constantly send camera data obtained to mobile phone APP.

```
60        while (client.connected()) {
61          fb = esp_camera_fb_get();
62          if (fb != NULL) {
63            uint8_t slen[4];
64            slen[0] = fb->len >> 0;
65            slen[1] = fb->len >> 8;
66            slen[2] = fb->len >> 16;
```

```
          slen[3] = fb->len >> 24;
          client.write(slen, 4);
          client.write(fb->buf, fb->len);
          esp_camera_fb_return(fb);
        }
        else {
          Serial.println("Camera Error");
        }
      }
```

The loopTask_Cmd() function sends the received instruction back to the phone app and prints it out through a serial port.

```
void loopTask_Cmd(void *pvParameters) {
  Serial.println("Task Cmd_Server is starting ... ");
  while (1) {
    WiFiClient client = server_Cmd.available(); // listen for incoming clients
    if (client) {                               // if you get a client,
      Serial.println("Command Server connected to a client.");// print a message out the
serial port
      String currentLine = "";          // make a String to hold incoming data from the client
      while (client.connected()) {      // loop while the client's connected
        if (client.available()) {       // if there's bytes to read from the client,
          char c = client.read();       // read a byte, then
          client.write(c);
          Serial.write(c);              // print it out the serial monitor
          if (c == '\n') {              // if the byte is a newline character
            currentLine = "";
          }
          else {
            currentLine += c;           // add it to the end of the currentLine
          }
        }
      }
      // close the connection:
      client.stop();
      Serial.println("Command Client Disconnected.");
    }
  }
}
```

loopTask_ Blink()function will control the blinking of LED. When you see LED blinking, it indicates that ESP32-S3 has been configured and starts working.

```
112 void loopTask_Blink(void *pvParameters) {
113     Serial.println("Task Blink is starting ... ");
114     while (1) {
115         digitalWrite(LED_BUILT_IN, !digitalRead(LED_BUILT_IN));
116         delay(1000);
117     }
118 }
```

If you do not have a router near you, or if you are outdoors, you can annotate the following code, and then compile and upload it to ESP32-S3. And you can display the video images on your phone by Method 1.

```
32     ////////////////////////////////////////////////
33     WiFi.begin(ssid_Router, password_Router);
34     Serial.print("Connecting ");
35     Serial.print(ssid_Router);
36     while (WiFi.isConnected() != true) {
37       delay(500);
38       Serial.print(".");
39     }
40     while (WiFi.STA.hasIP() != true) {
41       Serial.print(".");
42       delay(500);
43     }
44     Serial.println("");
45     Serial.println("WiFi connected");
46     ////////////////////////////////////////////////
```

What's next?

Thanks for your reading. This tutorial is all over here. If you find any mistakes, omissions or you have other ideas and questions about contents of this tutorial or the kit and etc., please feel free to contact us:

support@freenove.com

We will check and correct it as soon as possible.

If you want learn more about ESP32-S3, you view our ultimate tutorial:
https://github.com/Freenove/Freenove_Ultimate_Starter_Kit_for_ESP32_S3/archive/master.zip

If you want to learn more about Arduino, Raspberry Pi, smart cars, robots and other interesting products in science and technology, please continue to focus on our website. We will continue to launch cost-effective, innovative and exciting products.

http://www.freenove.com/

End of the Tutorial

Thank you again for choosing Freenove products.