# 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....1
ESP32-S3 WROOM....2
CH343 (Importance)....4
Programming Software....16
Install Arduino ESP32 Package....19
Config ESP32S3 WROOM Lite....22
Notes for GPIO....23
Chapter 1 LED....25
Project 1.1 Blink....25
Chapter 2 WS2812....34
Project 2.1 WS2812....34
Chapter 3 Serial Communication....39
Project 3.1 Serial Print....39
Project 3.2 Serial Read and Write....43
Chapter 4 BLE....45
Project 4.1 Bluetooth Low Energy Data Passthrough....45
Project 4.2 Bluetooth Control LED....57
Chapter 5 WiFi Working Modes....63
Project 5.1 Station mode....63
Project 5.2 AP mode....67
Project 5.3 AP+Station mode....72
Chapter 6 TCP/IP....76
Project 6.1 As Client....76
Project 6.2 As Server....88
What's next?....93
End of the Tutorial....93

# 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**

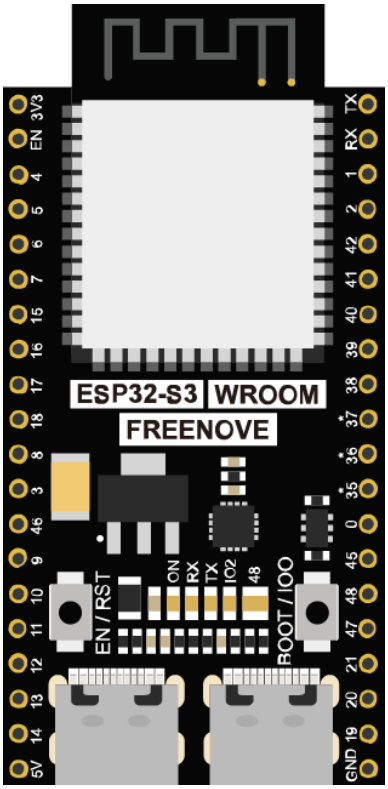
# ESP32-S3 WROOM

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.

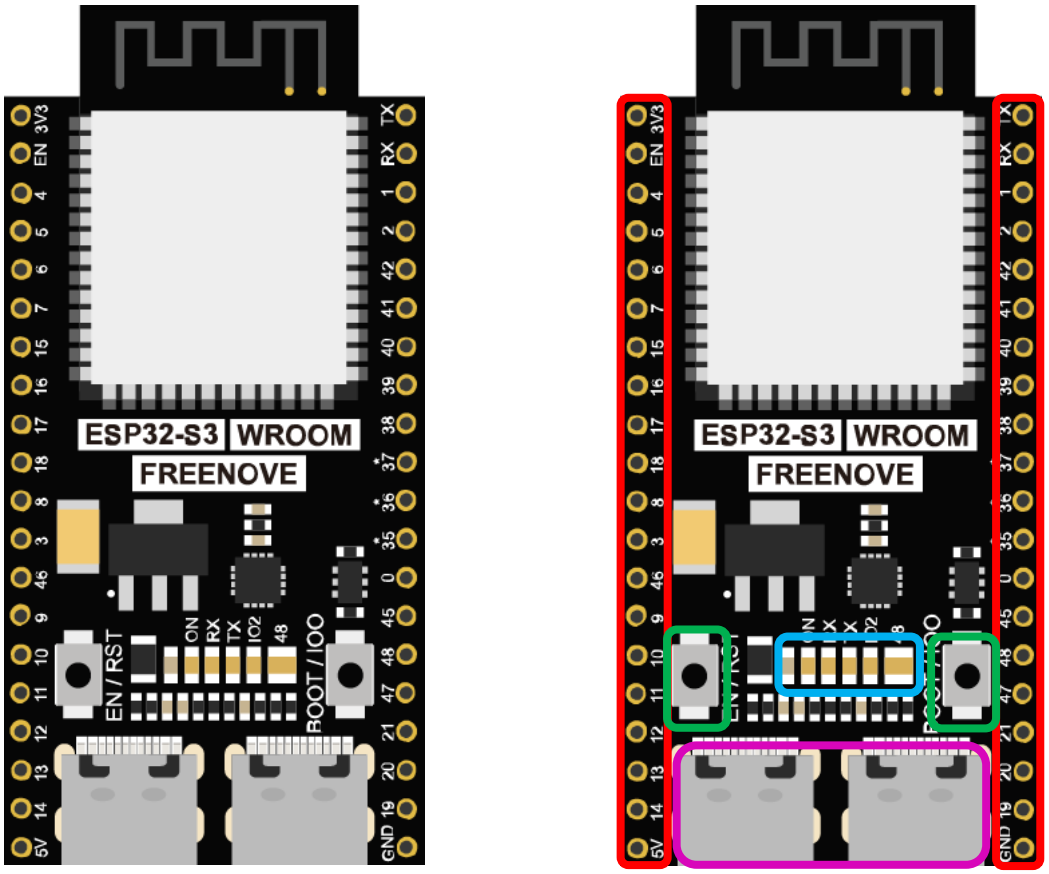
| PCB on-board antenna | IPEX antenna |
| --- | --- |
|  |  |

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

ESP32-S3 WROOM Lite



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** |
| | **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.

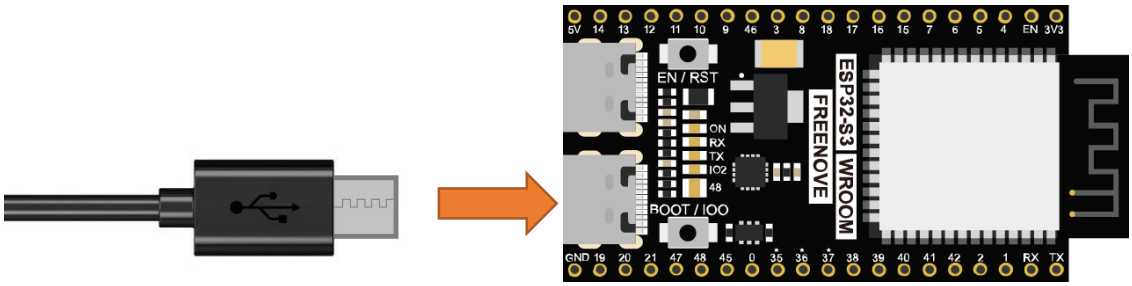
# CH343 (Importance)

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

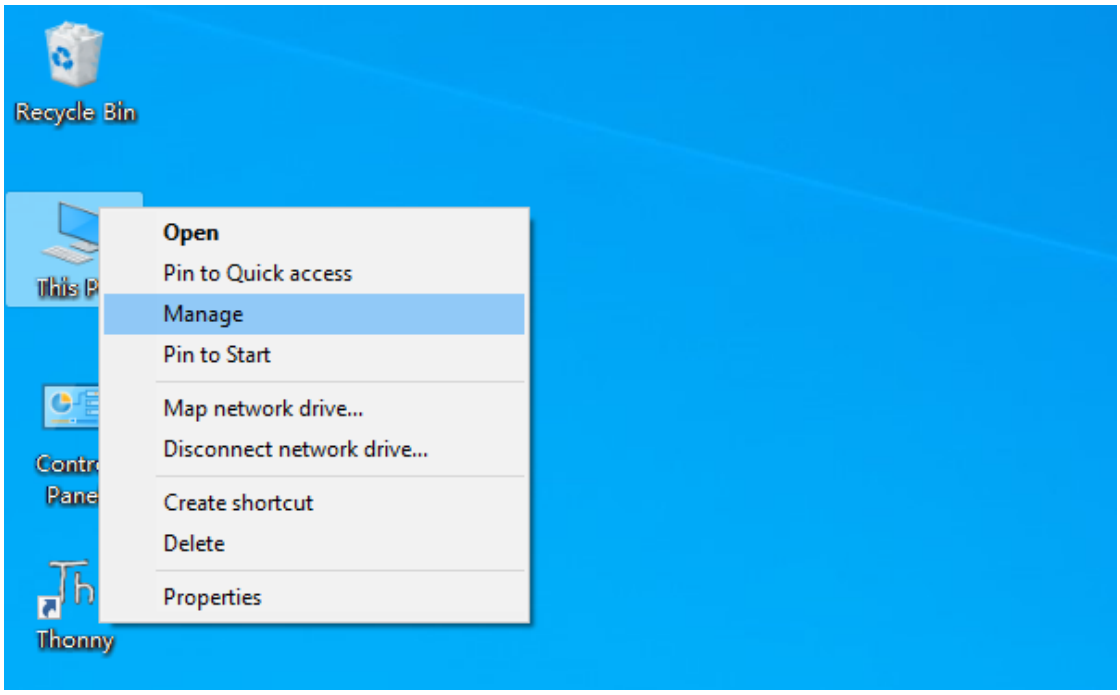
## Windows

### Check whether CH343 has been installed

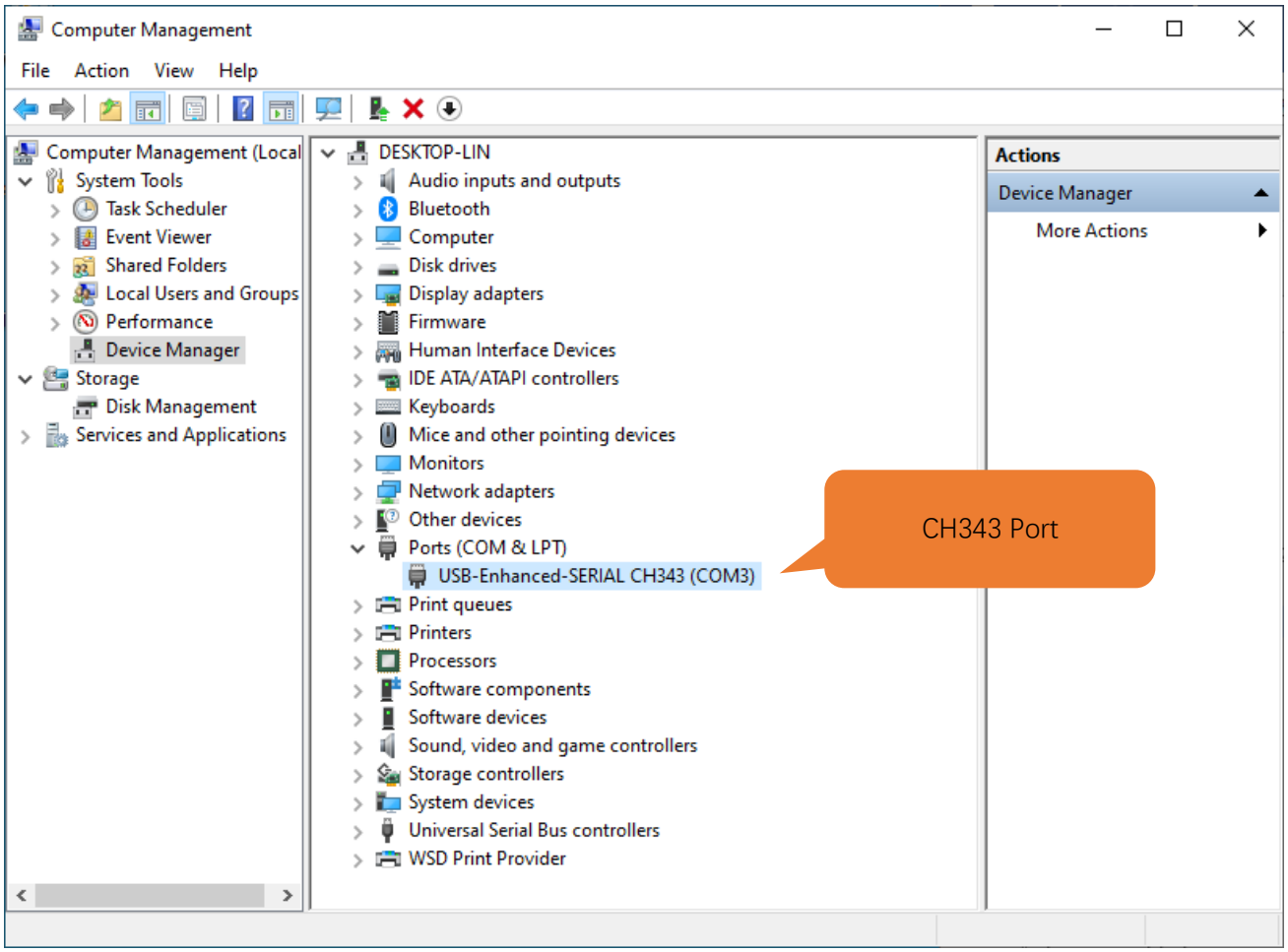
1. Connect your computer and ESP32-S3 WROOM with a Type C cable.




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

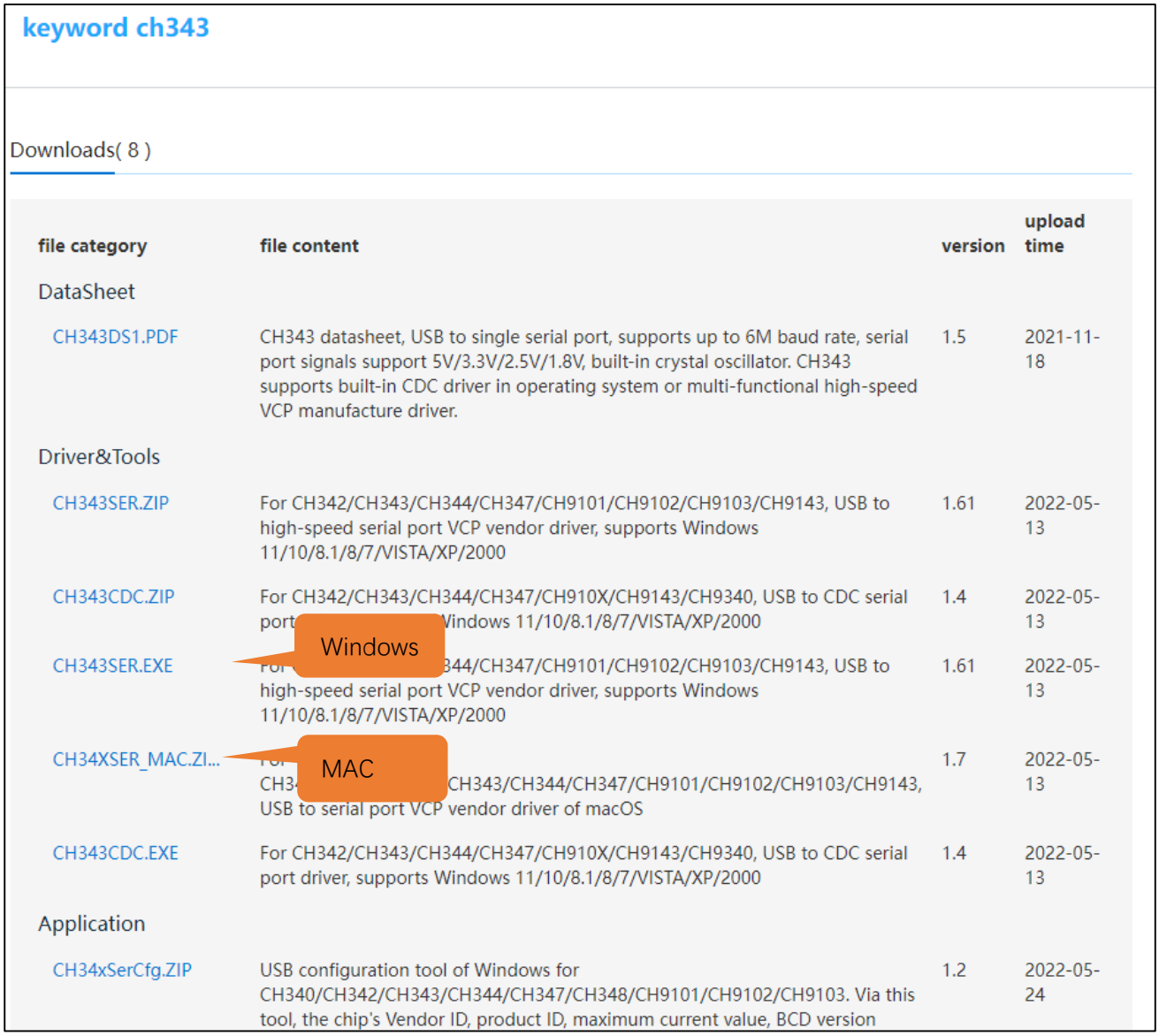


3. Click "Device Manager". If your computer has installed CH343, you can see"USB-Enhances-SERIAL CH343 (COMx)". And you can click here to move to the next step.



## 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.



keyword ch343

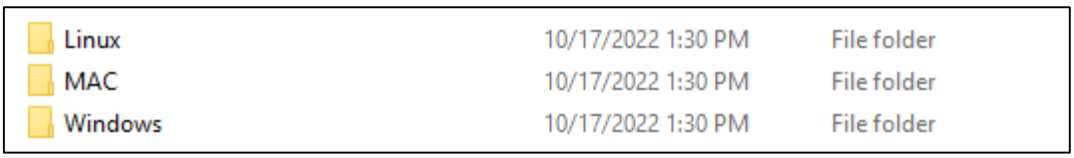
Downloads( 8 )

| file category | file content | version | upload time |
|---|---|---|---|
| DataSheet | | | |
| CH343DS1.PDF | CH343 datasheet, USB to single serial port, supports up to 6M baud rate, serial port signals support 5V/3.3V/2.5V/1.8V, built-in crystal oscillator. CH343 supports built-in CDC driver in operating system or multi-functional high-speed VCP manufacture driver. | 1.5 | 2021-11-18 |
| Driver&Tools | | | |
| CH343SER.ZIP | For CH342/CH343/CH344/CH347/CH9101/CH9102/CH9103/CH9143, USB to high-speed serial port VCP vendor driver, supports Windows 11/10/8.1/8/7/VISTA/XP/2000 | 1.61 | 2022-05-13 |
| CH343CDC.ZIP | For CH342/CH343/CH344/CH347/CH910X/CH9143/CH9340, USB to CDC serial port Windows 11/10/8.1/8/7/VISTA/XP/2000 | 1.4 | 2022-05-13 |
| CH343SER.EXE | For 344/CH347/CH9101/CH9102/CH9103/CH9143, USB to high-speed serial port VCP vendor driver, supports Windows 11/10/8.1/8/7/VISTA/XP/2000 | 1.61 | 2022-05-13 |
| CH34XSER_MAC.ZI... | For CH343/CH344/CH347/CH9101/CH9102/CH9103/CH9143, USB to serial port VCP vendor driver of macOS | 1.7 | 2022-05-13 |
| CH343CDC.EXE | For CH342/CH343/CH344/CH347/CH910X/CH9143/CH9340, USB to CDC serial port driver, supports Windows 11/10/8.1/8/7/VISTA/XP/2000 | 1.4 | 2022-05-13 |
| Application | | | |
| CH34xSerCfg.ZIP | USB configuration tool of Windows for CH340/CH342/CH343/CH344/CH347/CH348/CH9101/CH9102/CH9103. Via this tool, the chip's Vendor ID, product ID, maximum current value, BCD version | 1.2 | 2022-05-24 |

Windows

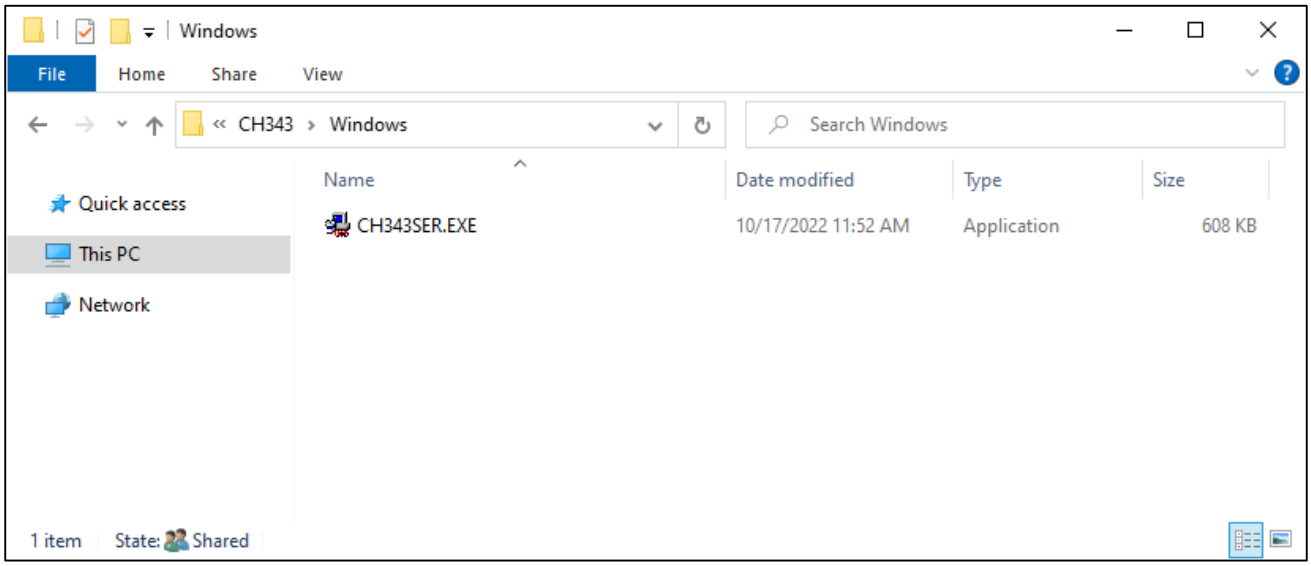
MAC

If you would not like to download the installation package, you can open **"Freenove_ESP32_S3_WROOM_Board_Lite/CH343"**, we have prepared the installation package.

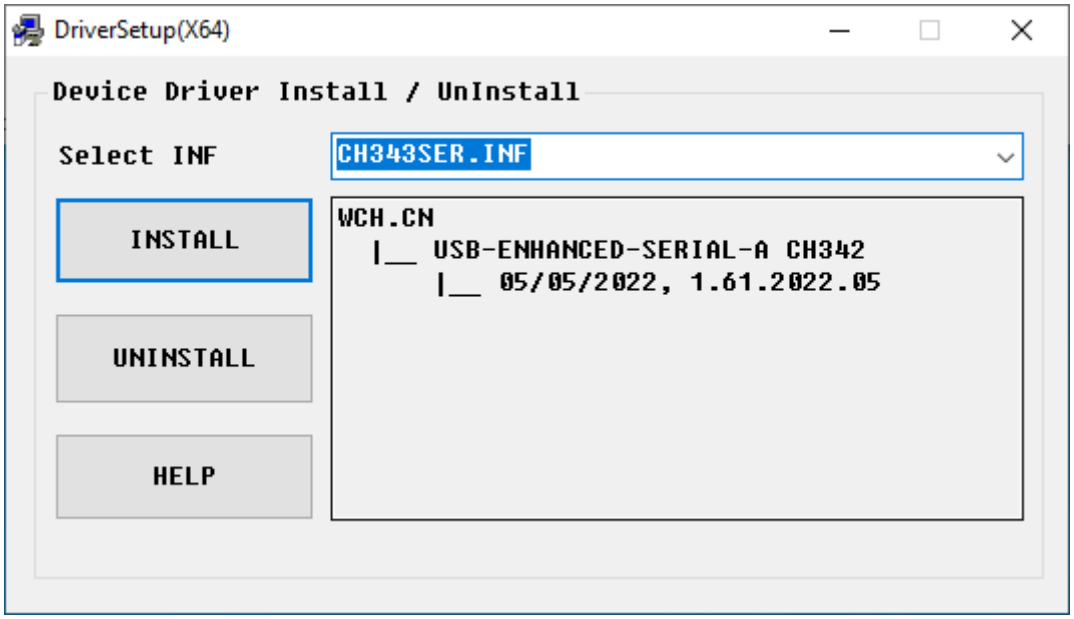


| | | |
|---|---|---|
| Linux | 10/17/2022 1:30 PM | File folder |
| MAC | 10/17/2022 1:30 PM | File folder |
| Windows | 10/17/2022 1:30 PM | File folder |

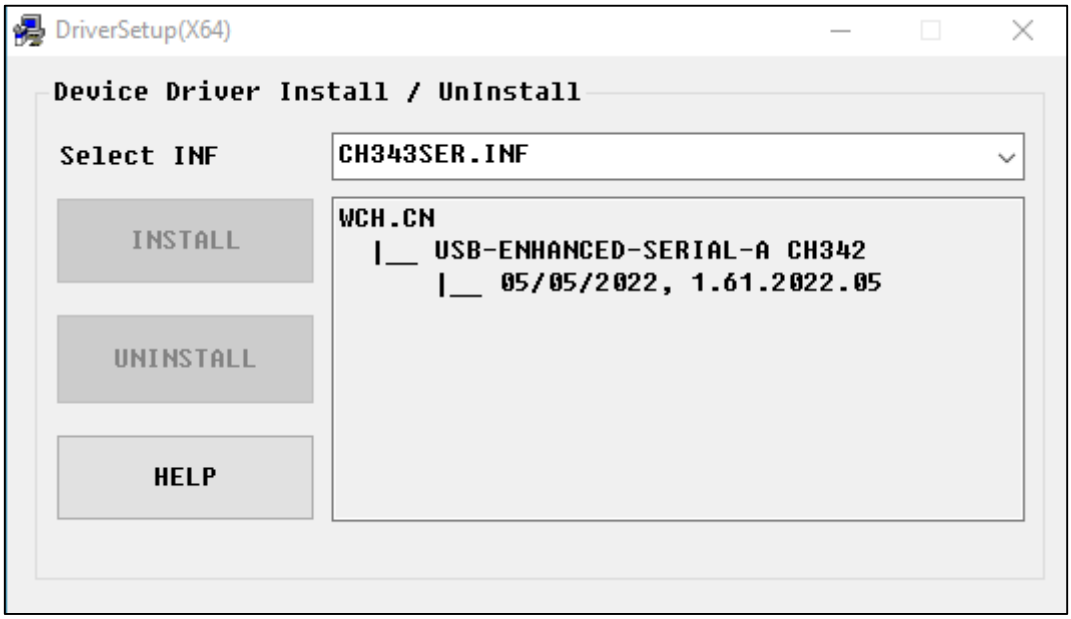
2. Open the folder "**Freenove_ESP32_S3_WROOM_Board_Lite/CH343/Windows/**"



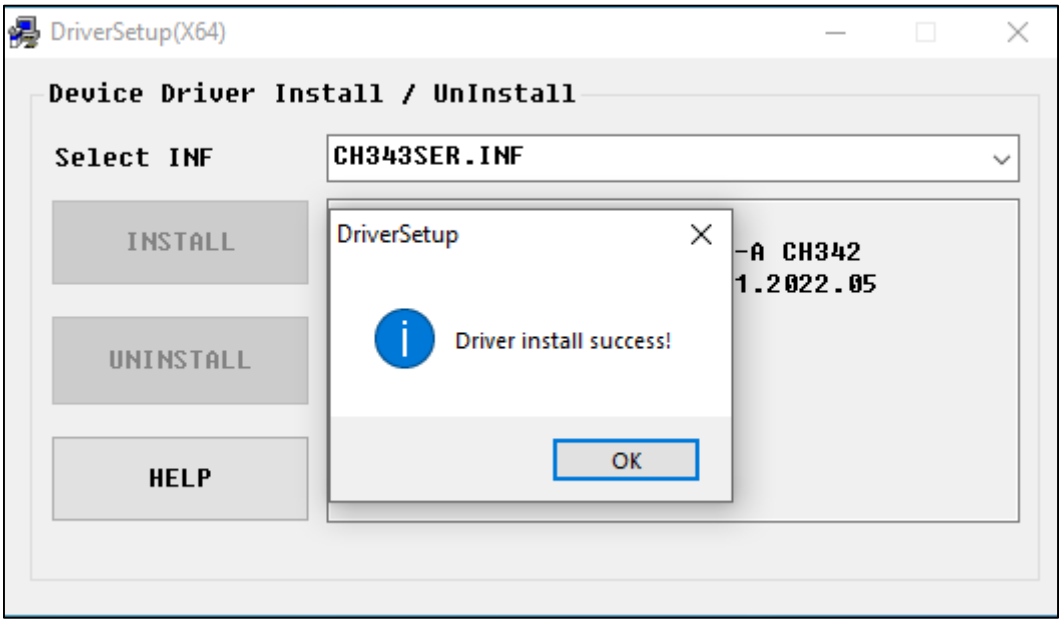

3. Double click "**CH343SER.EXE**".



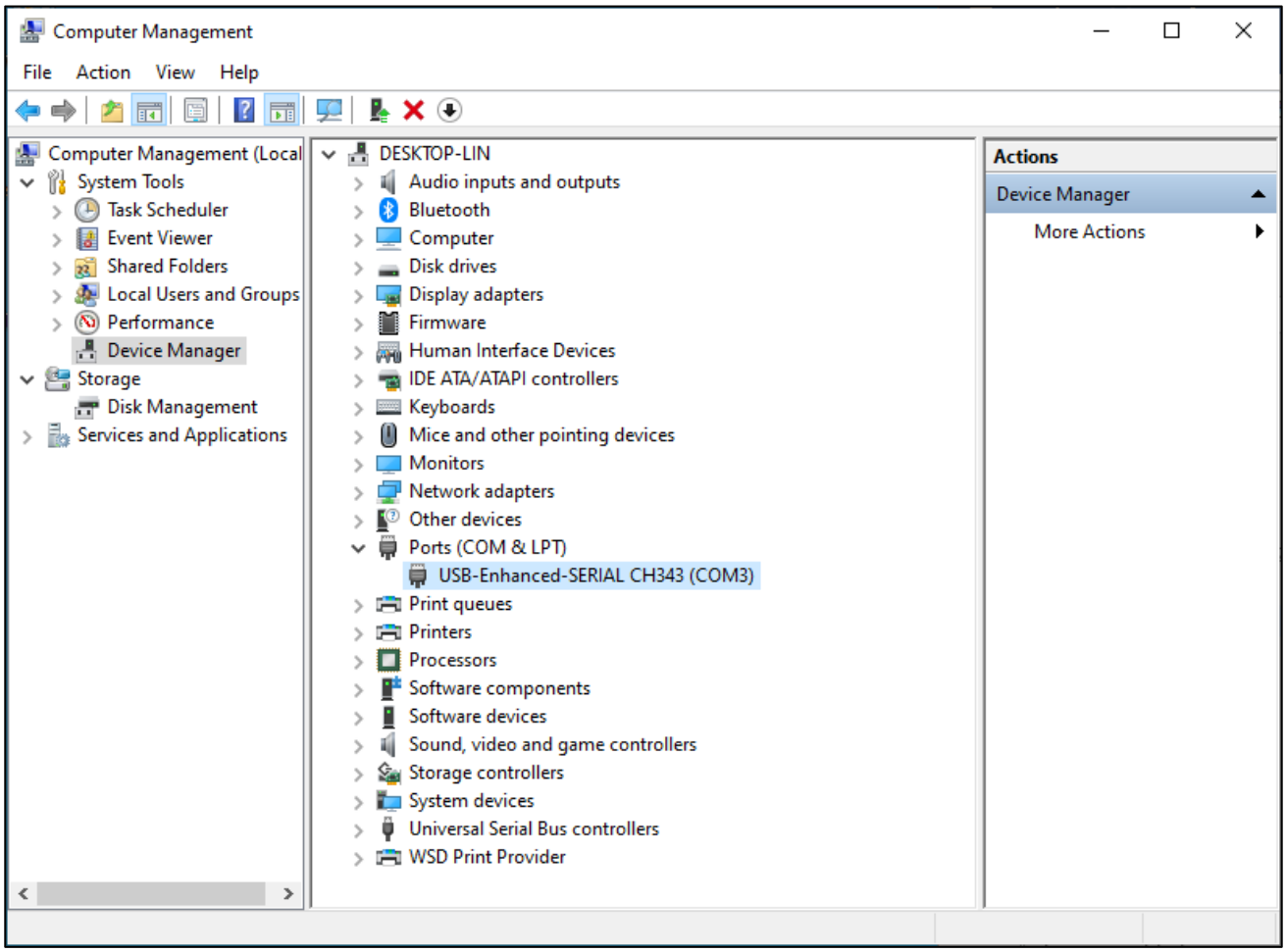
4. Click “INSTALL” and wait for the installation to complete.




5. Install successfully. Close all interfaces.



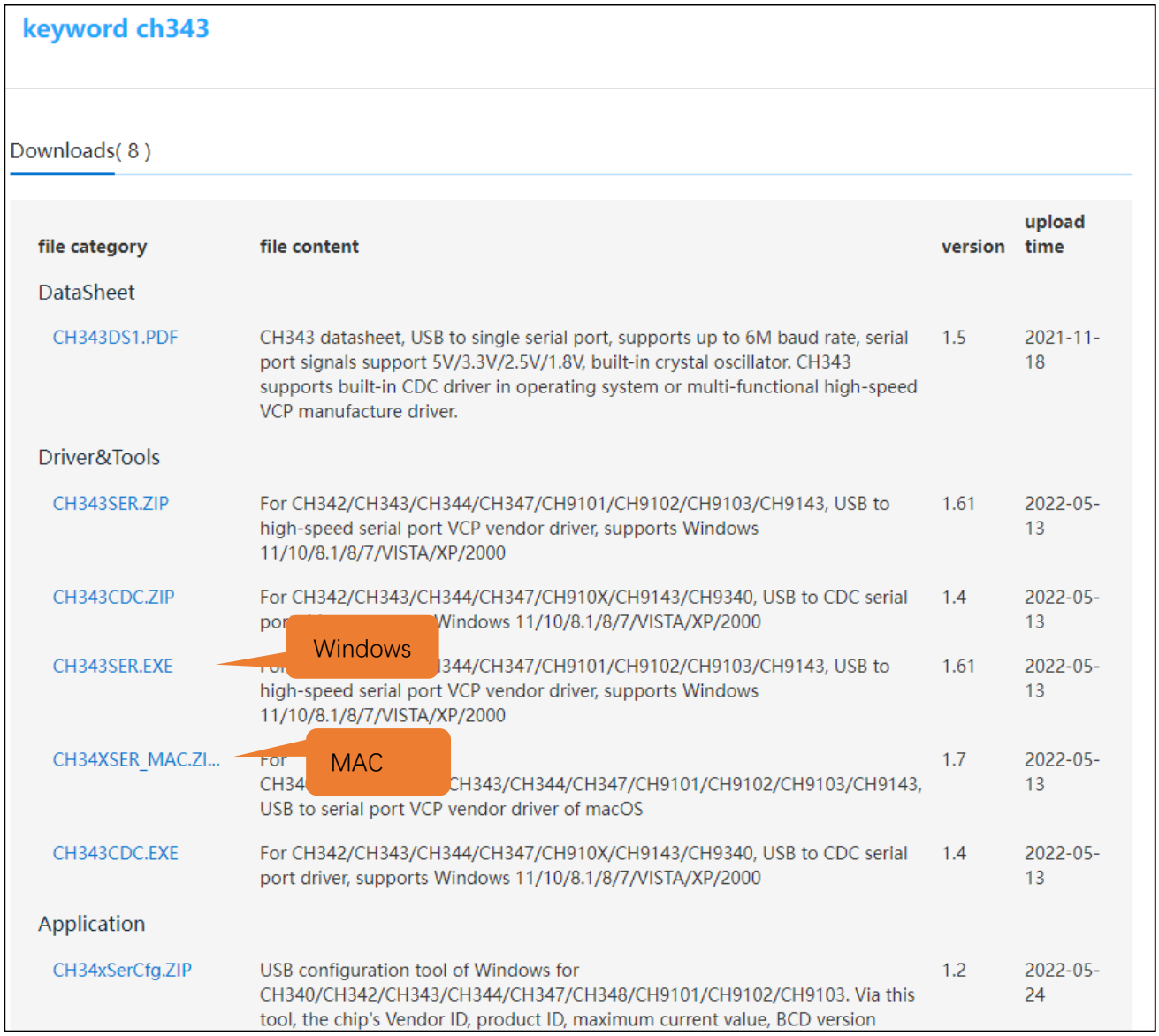
6. When ESP32-S3 WROOM is connected to computer, select “This PC”, right-click to select “Manage” and click “Device Manager” in the newly pop-up dialog box, and you can see the following interface.



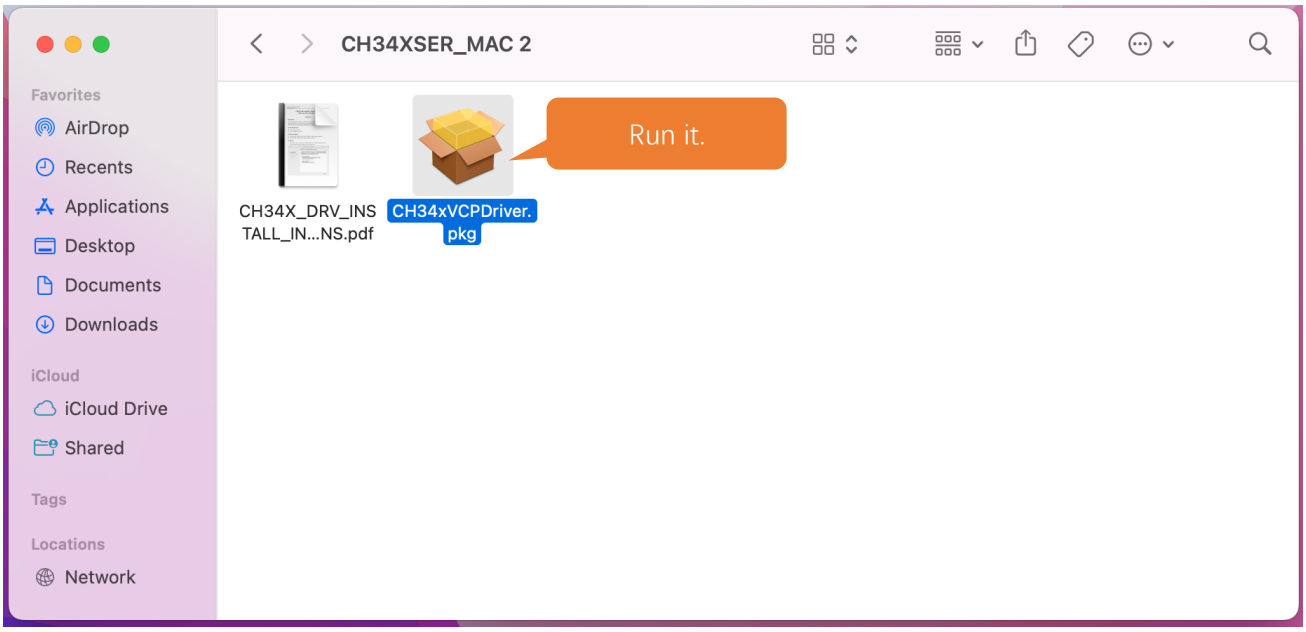

7. So far, CH343 has been installed successfully. Close all dialog boxes.

## MAC

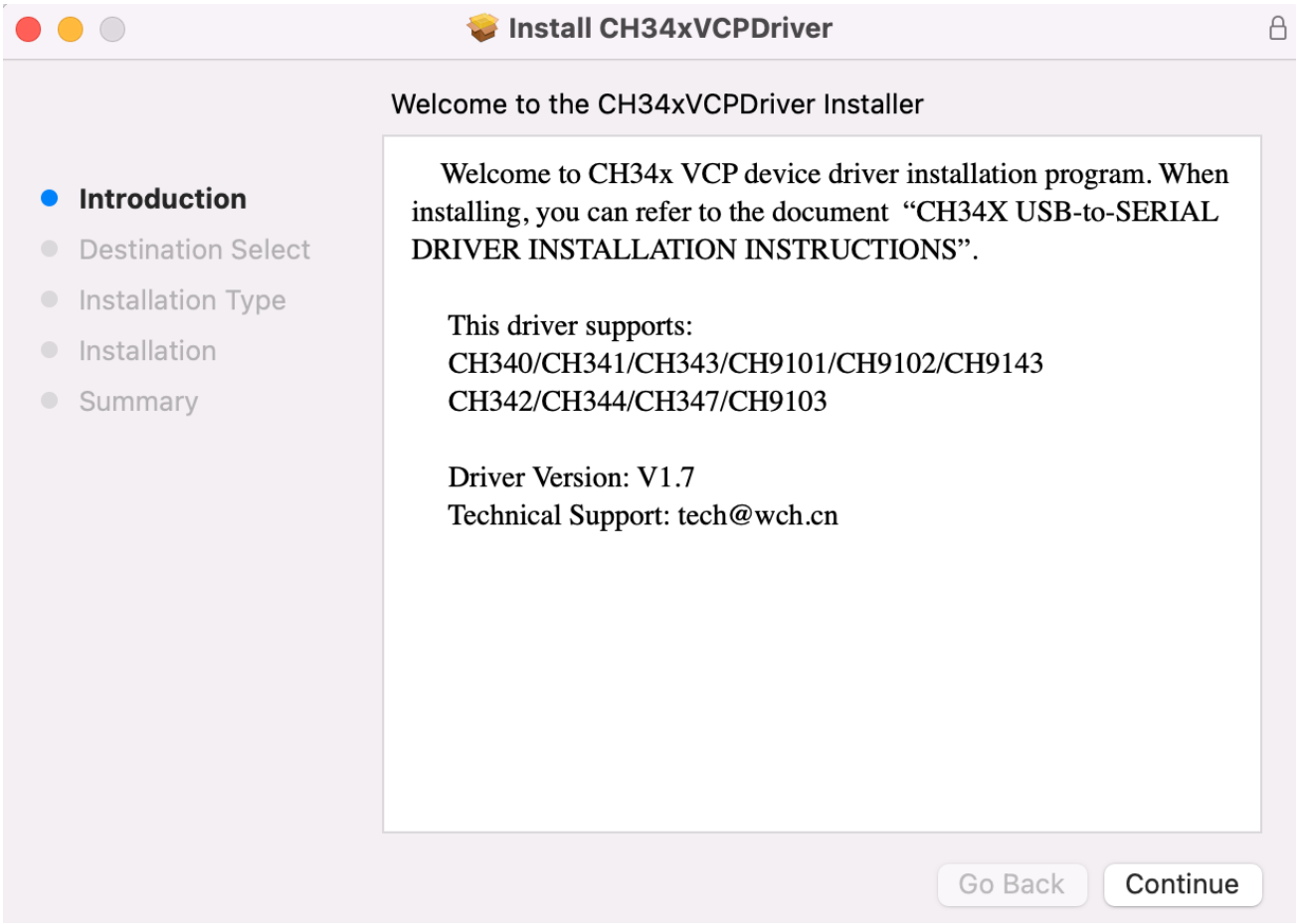
First, download CH343 driver, click http://www.wch-ic.com/search?t=all&q=ch343 to download the appropriate one based on your operating system.



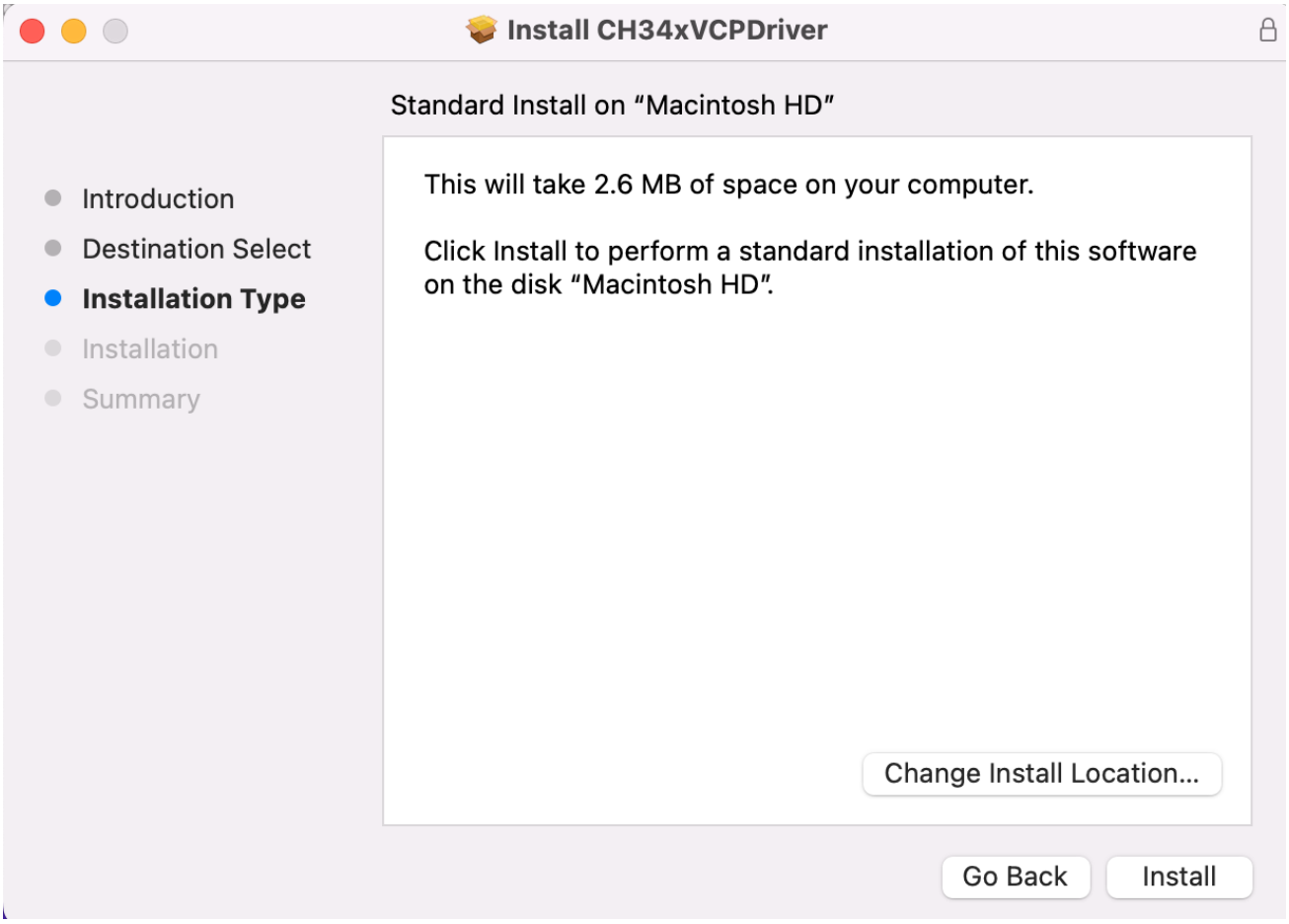

If you would not like to download the installation package, you can open **"Freenove_ESP32_S3_WROOM_Board_Lite/CH343"**, we have prepared the installation package.
Second, open the folder "**Freenove_ESP32_S3_WROOM_Board_Lite/CH343/MAC/**"



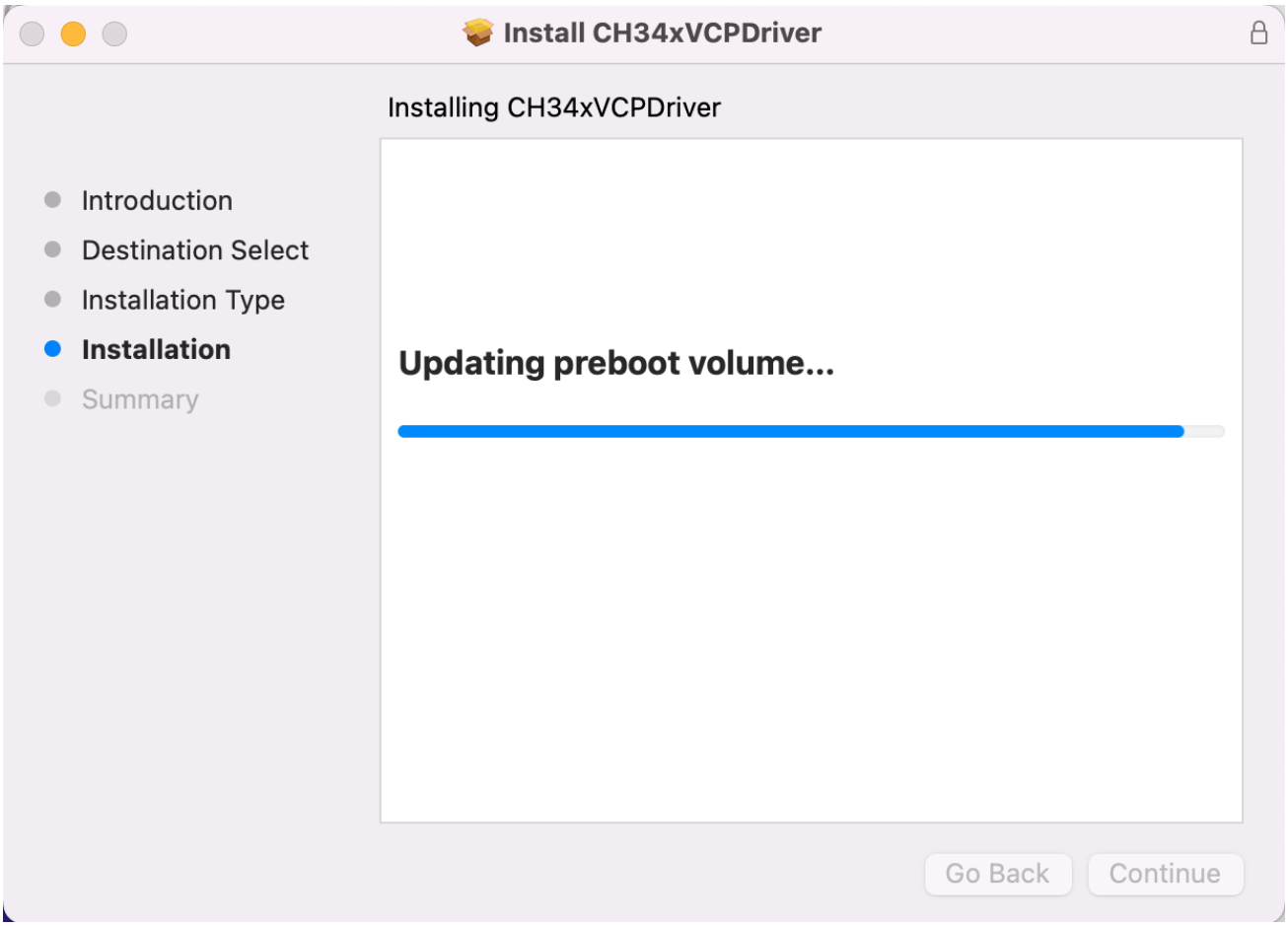

Third, click Continue.



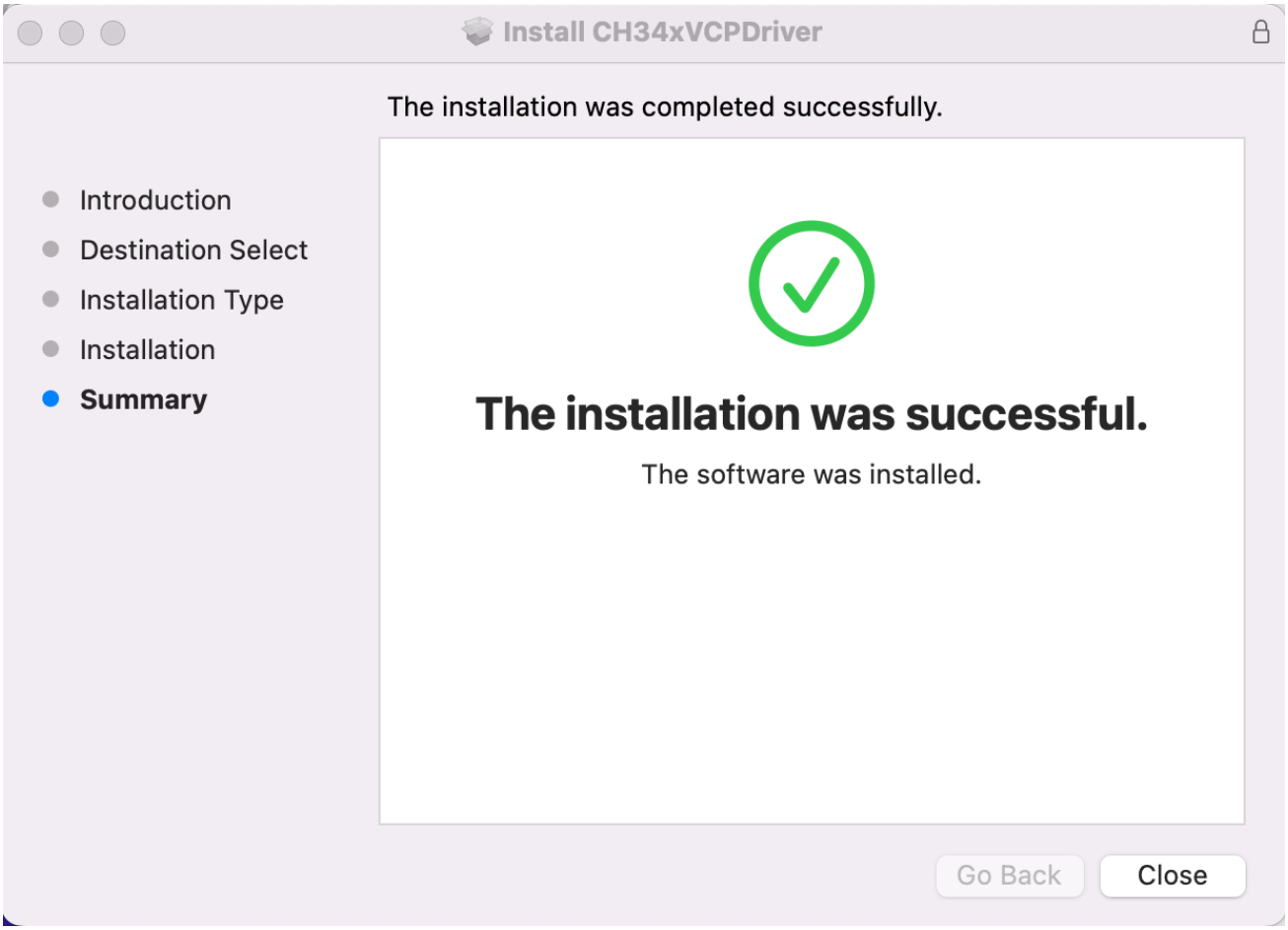

Fourth, click Install.


Install CH34xVCPDriver
Standard Install on "Macintosh HD"
Introduction
Destination Select
Installation Type
Installation
Summary
This will take 2.6 MB of space on your computer.
Click Install to perform a standard installation of this software on the disk "Macintosh HD".
Change Install Location...
Go Back
Install

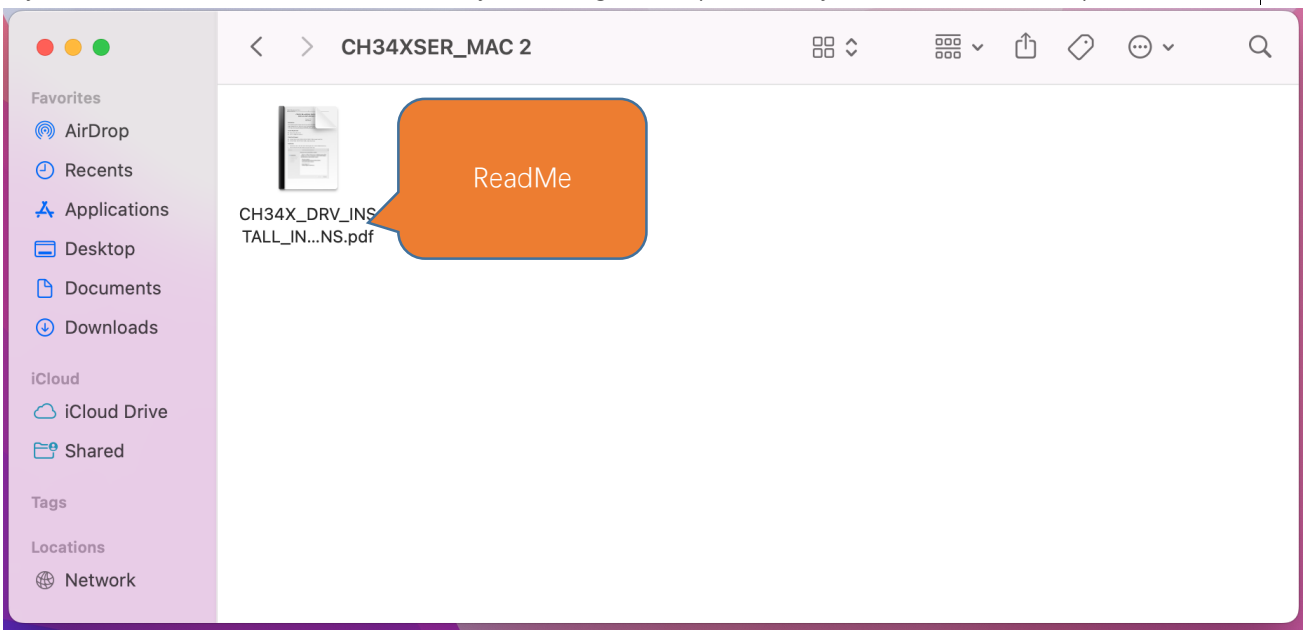
Then, waiting Finsh.




Finally, restart your PC.

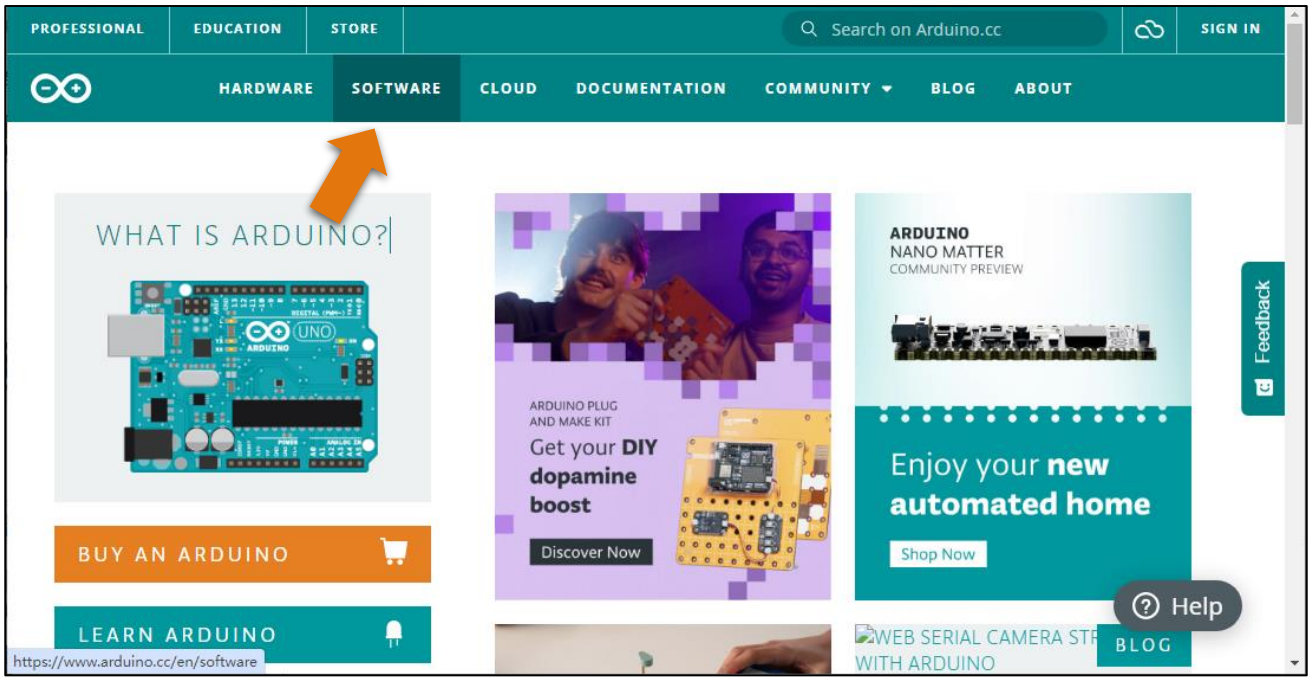

Install CH34xVCPDriver
The installation was completed successfully.
Introduction
Destination Select
Installation Type
Installation
Summary
The installation was successful.
The software was installed.
Go Back
Close

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

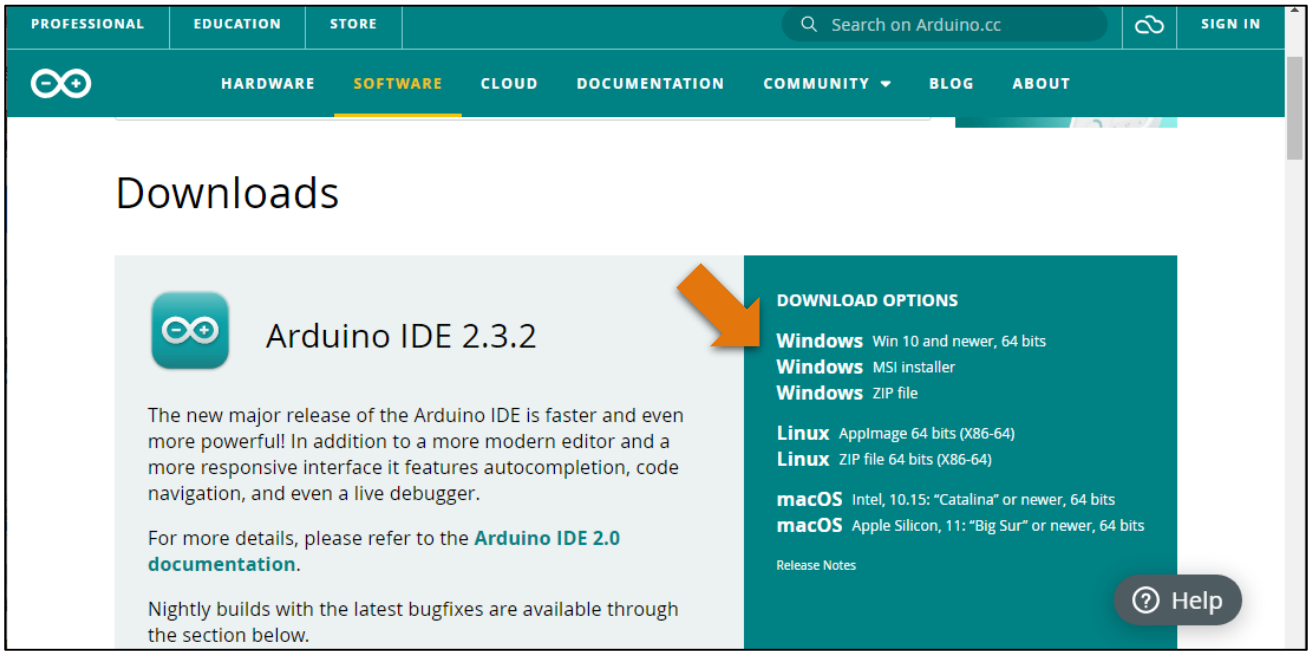


## Programming Software

Arduino Software (IDE) is used to write and upload the code for Arduino Board.
First, install Arduino Software (IDE): visit https://www.arduino.cc, click "Download" to enter the download page.

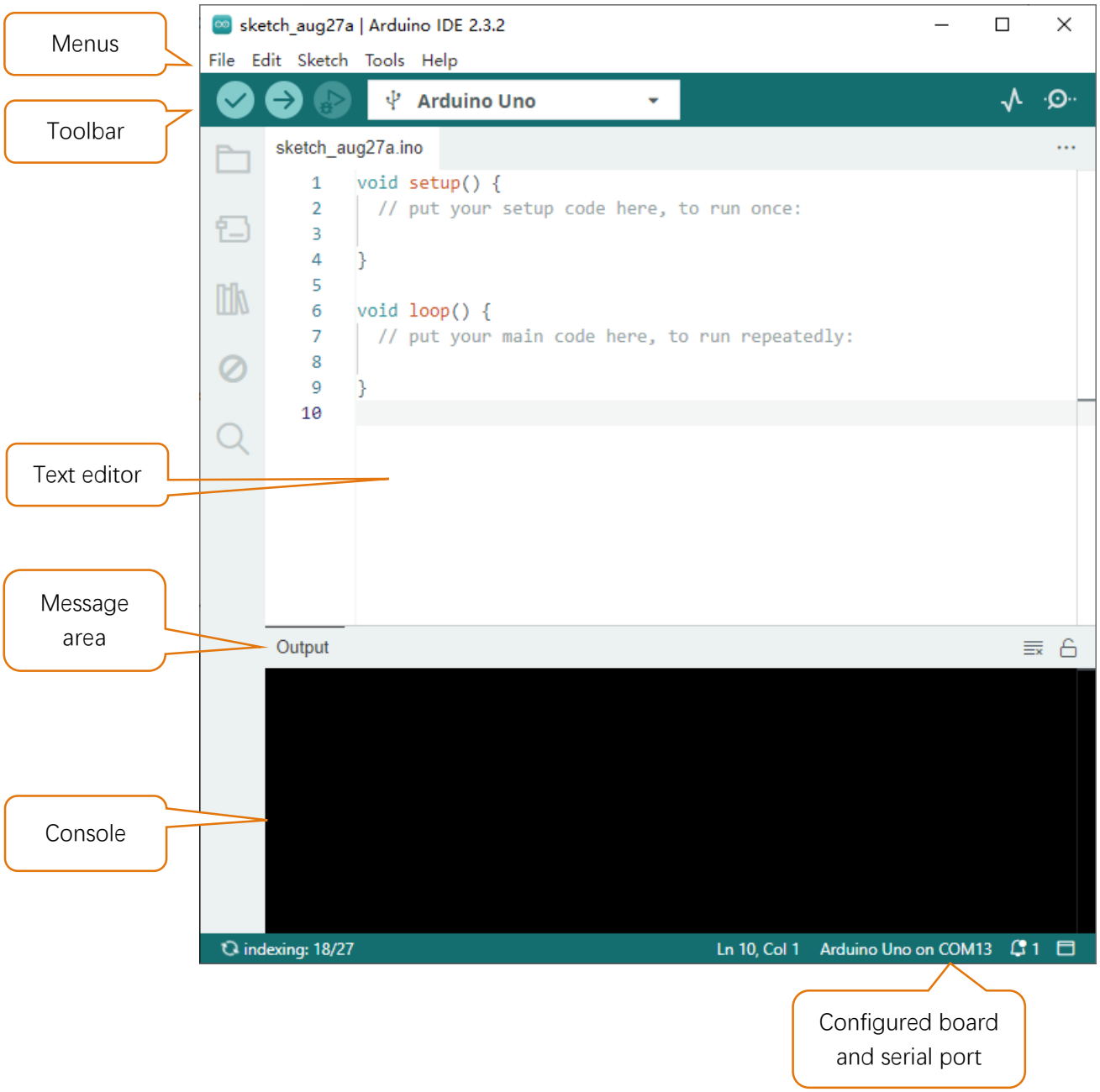



Select and download corresponding installer according to your operating system. If you are a windows user, please select the "Windows Installer" to download to install the driver correctly.




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:



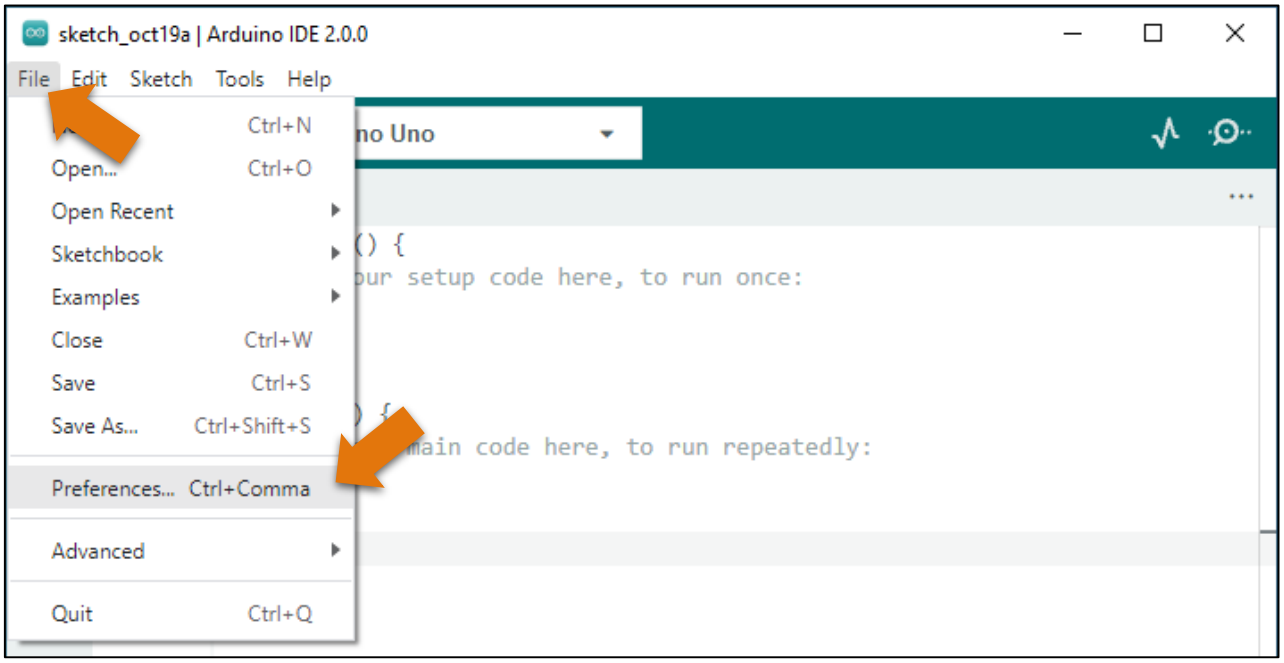
Programs written with Arduino Software (IDE) are called **sketches**. These sketches are written in the text editor and saved with the file extension**.ino**. The editor has features for cutting/pasting and searching/replacing text. The message area gives feedback while saving and exporting and also 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<br>Check your code for compile errors . |
|  | Upload<br>Compile your code and upload them to the configured board. |
|  | Debug<br>Debug code running on the board. (Some development boards do not support this function) |
|  | Development board selection<br>Configure the support package and upload port of the development board. |
|  | Serial Plotter<br>Receive serial port data and plot it in a discounted graph. |
|  | Serial Monitor<br>Open the serial monitor. |

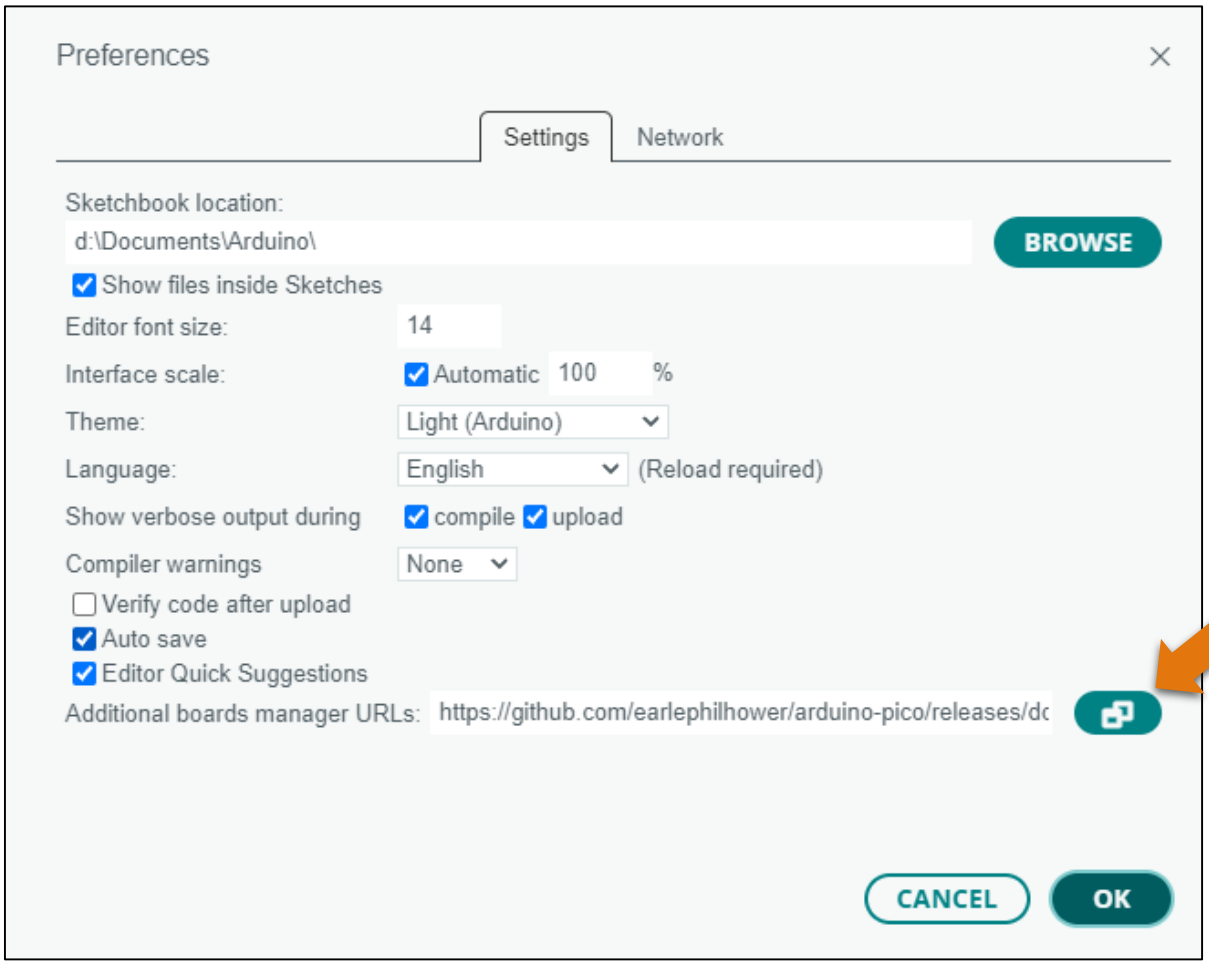
Additional commands are found within the five menus: File, Edit, Sketch, Tools, Help. The menus are context sensitive, which means only those items relevant to the work currently being carried out are available.

## Install Arduino ESP32 Package

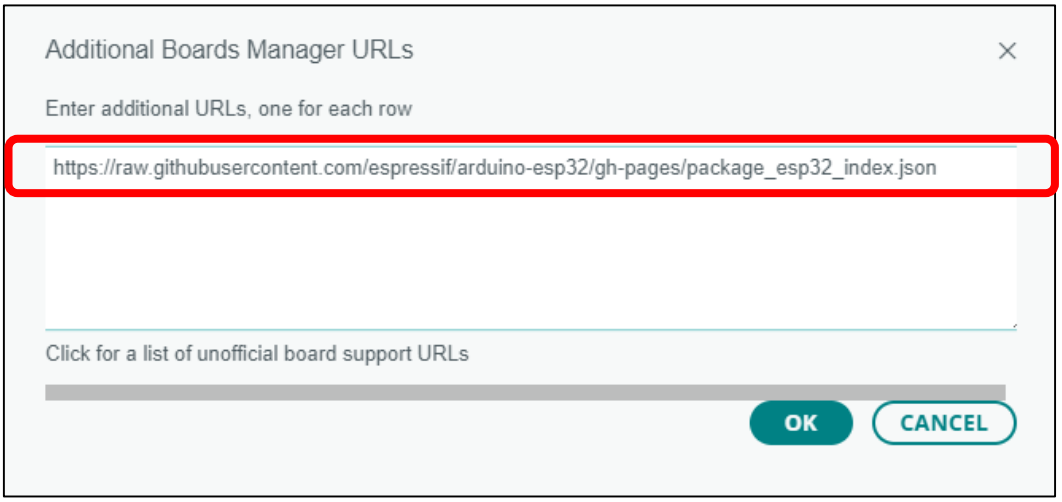
First, open the software platform arduino, and then click File in Menus and select Preferences.



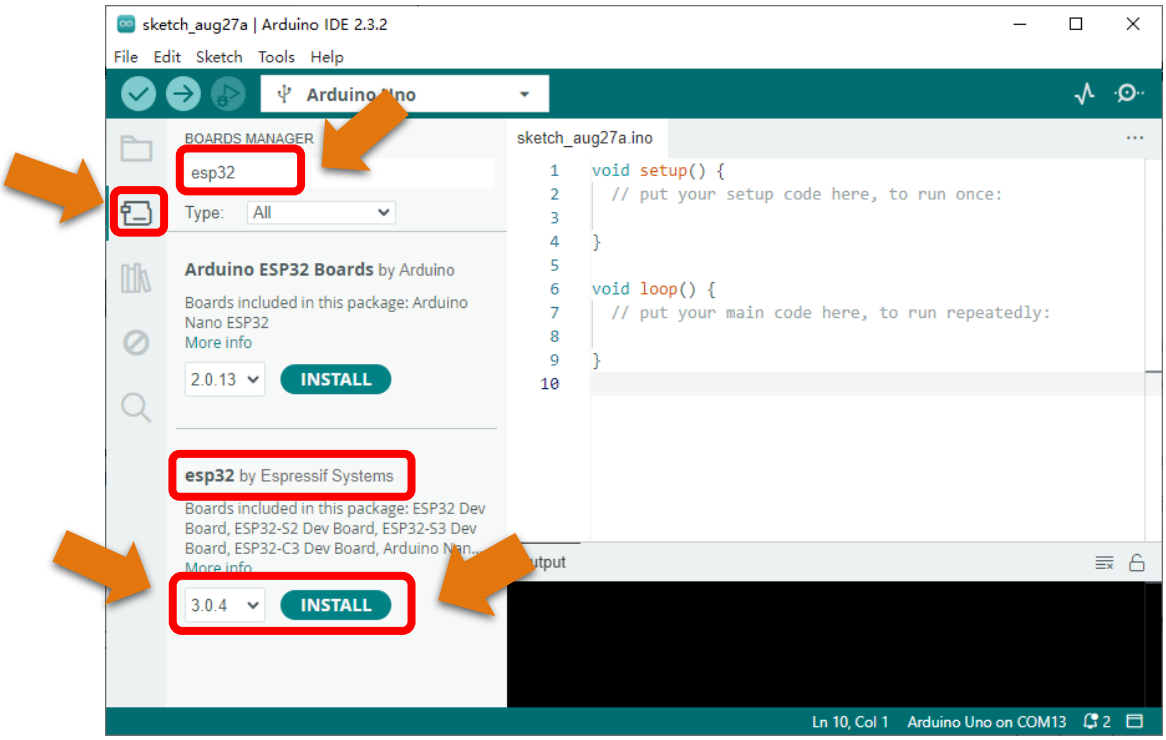

Second, click on the symbol behind "Additional Boards Manager URLs"



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.



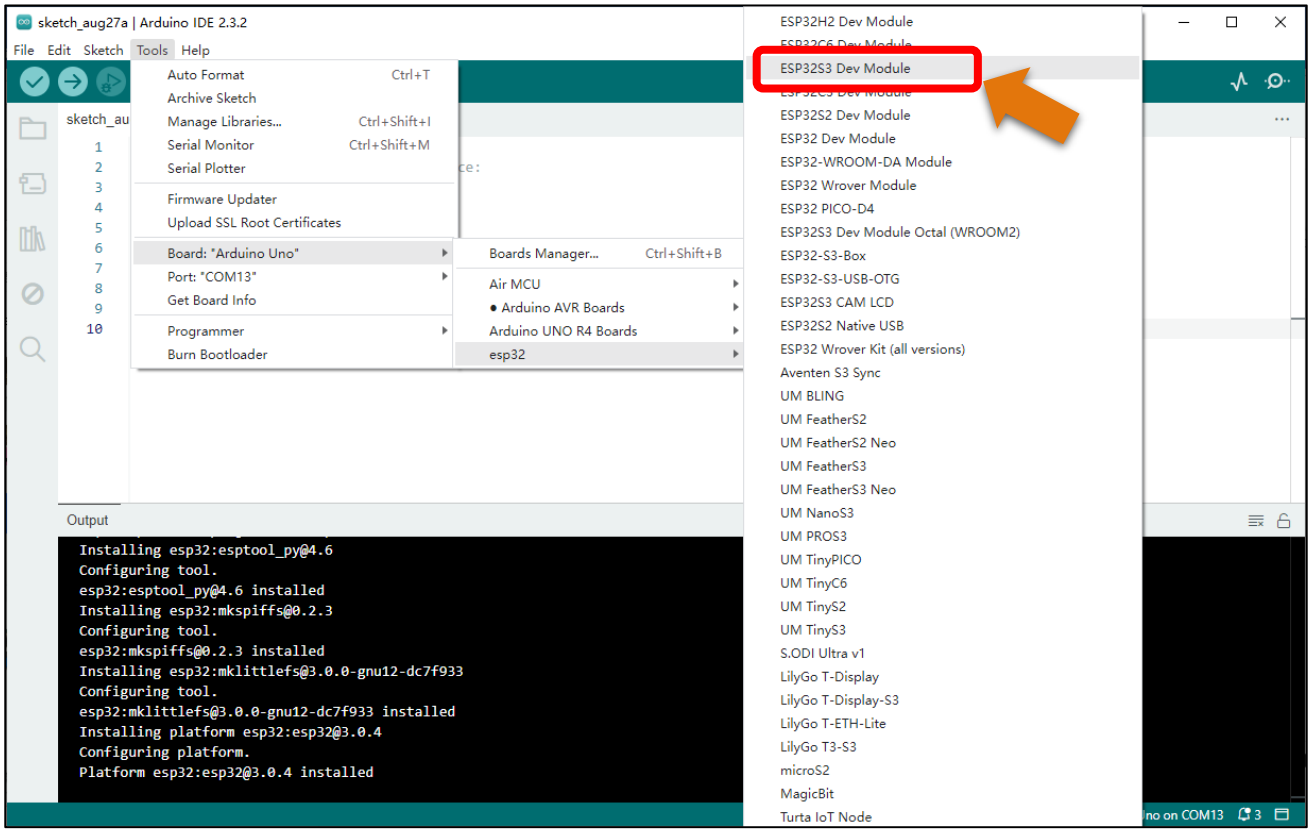

Fourth, click "Boards Manager". Enter “esp32” in Boards manager and select 3.0.4, Then click “INSTALL”.



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

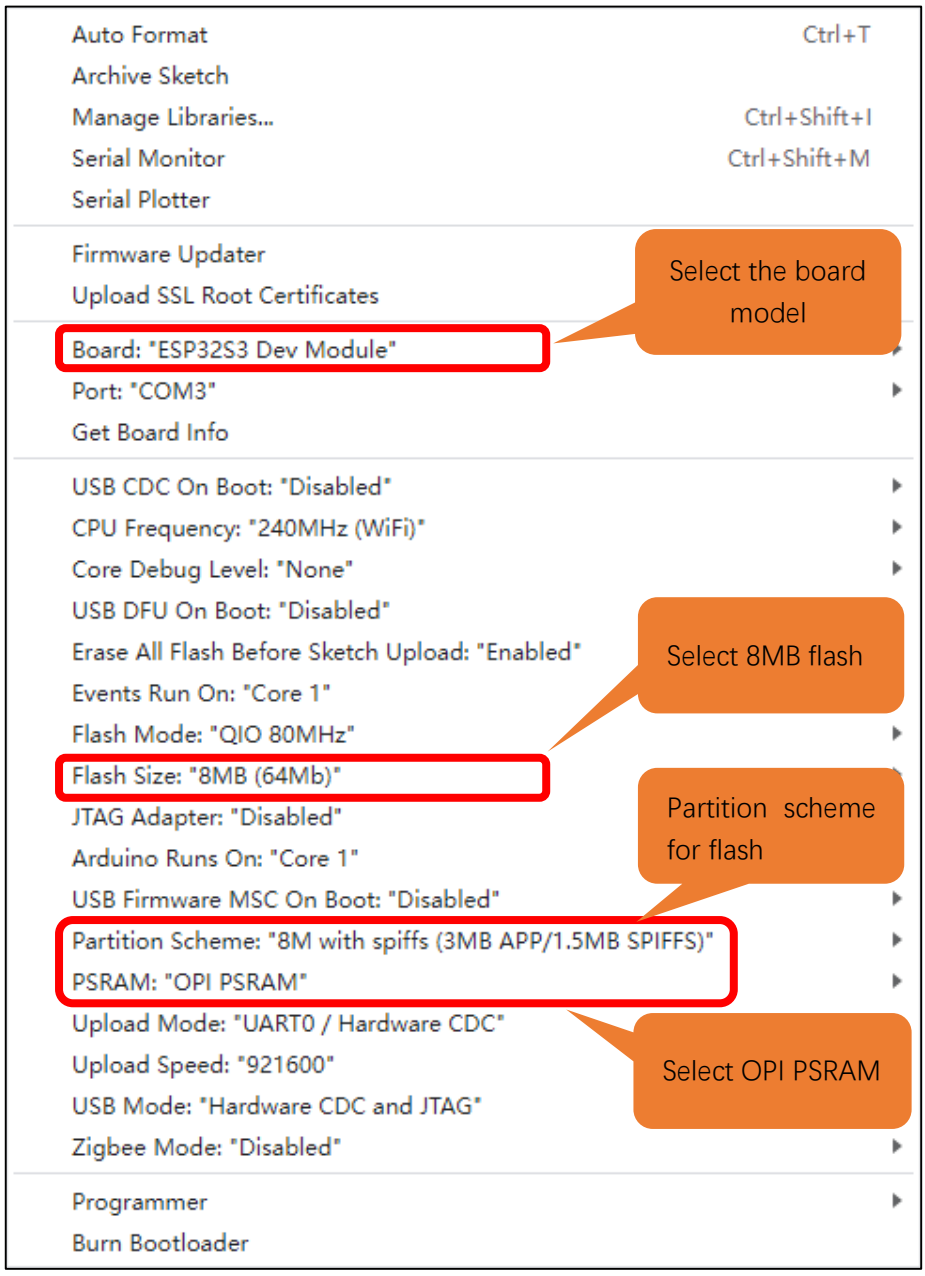
```
Installing esp32:esptool_py@4.6
Configuring tool.
esp32:esptool_py@4.6 installed
Installing esp32:mkspiffs@0.2.3
Configuring tool.
esp32:mkspiffs@0.2.3 installed
Installing esp32:mklittlefs@3.0.0-gnu12-dc7f933
Configuring tool.
esp32:mklittlefs@3.0.0-gnu12-dc7f933 installed
Installing platform esp32:esp32@3.0.4
Configuring platform.
Platform esp32:esp32@3.0.4 installed
```

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.



# Config ESP32S3 WROOM Lite

Please configure the ESP32S3 WROOM as follows when opening each Sketch.



# 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<br>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 |

## USB Pin

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

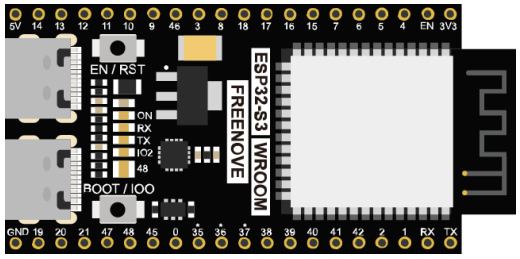
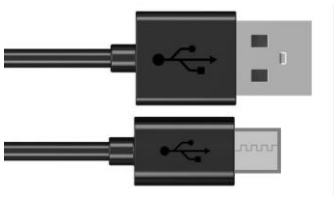
# Chapter 1 LED

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

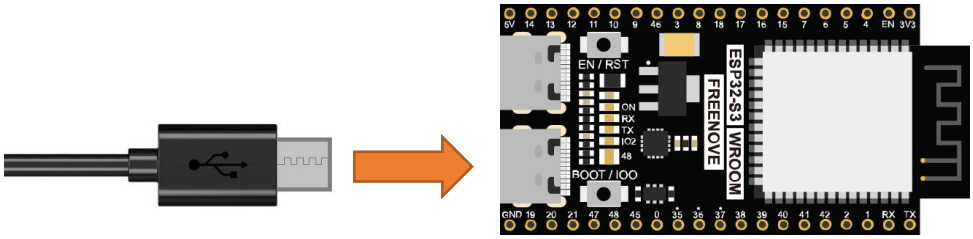
In this project, we will use ESP32-S3 WROOM to control blinking a common LED.

### Component List

| | |
|---|---|
| ESP32-S3 WROOM x1  | Type C Wire x1  |

#### Power

ESP32-S3 WROOM needs 5v power supply. In this tutorial, we need connect ESP32-S3 WROOM to computer via Type C 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 Type C cable to power ESP32-S3 WROOM by default.

## 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.

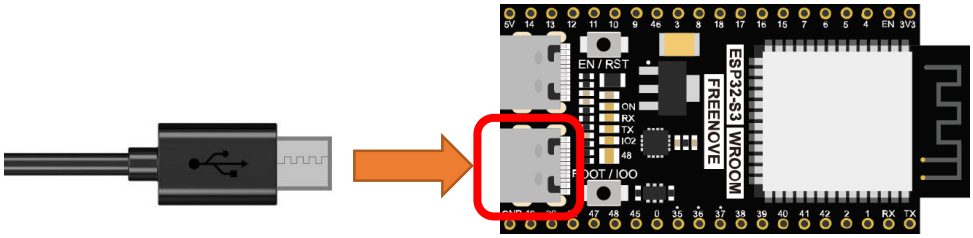
Upload the following Sketch:

**Freenove_ESP32_S3_WROOM_Board_Lite\Sketches\Sketch_01.1_Blink**.

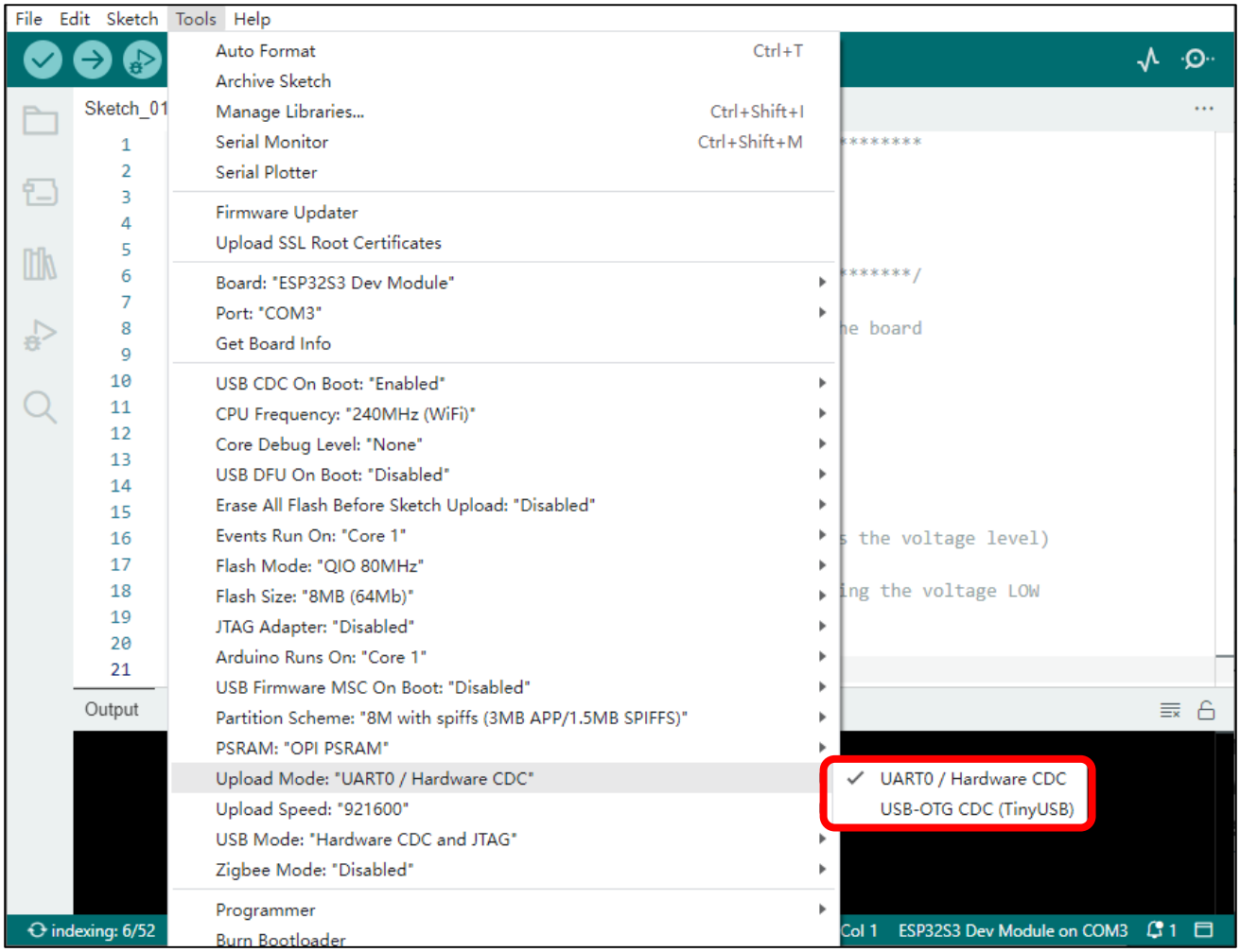
Next we will introduce two ways to upload code to ESP32-S3 WROOM.

### Option 1:

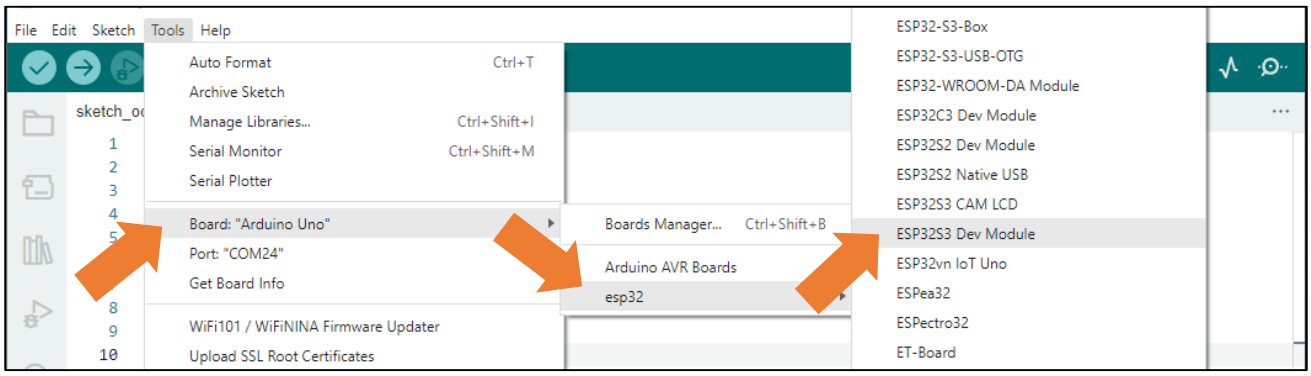
Connect ESP32-S3 WROOM to computer.




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

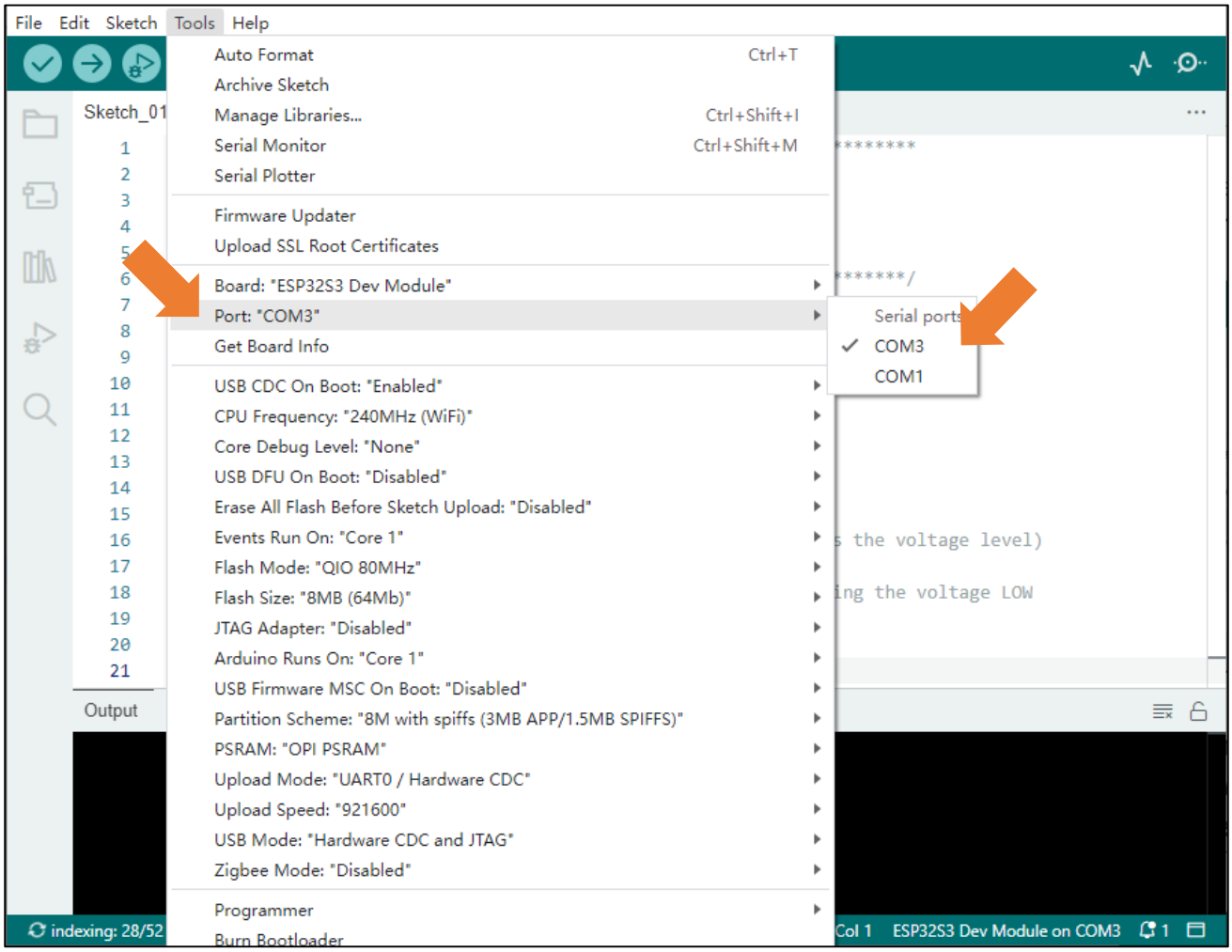


Before uploading the code, click "**Tools**", "**Board**" and select "**ESP32S3 Dev Module**".

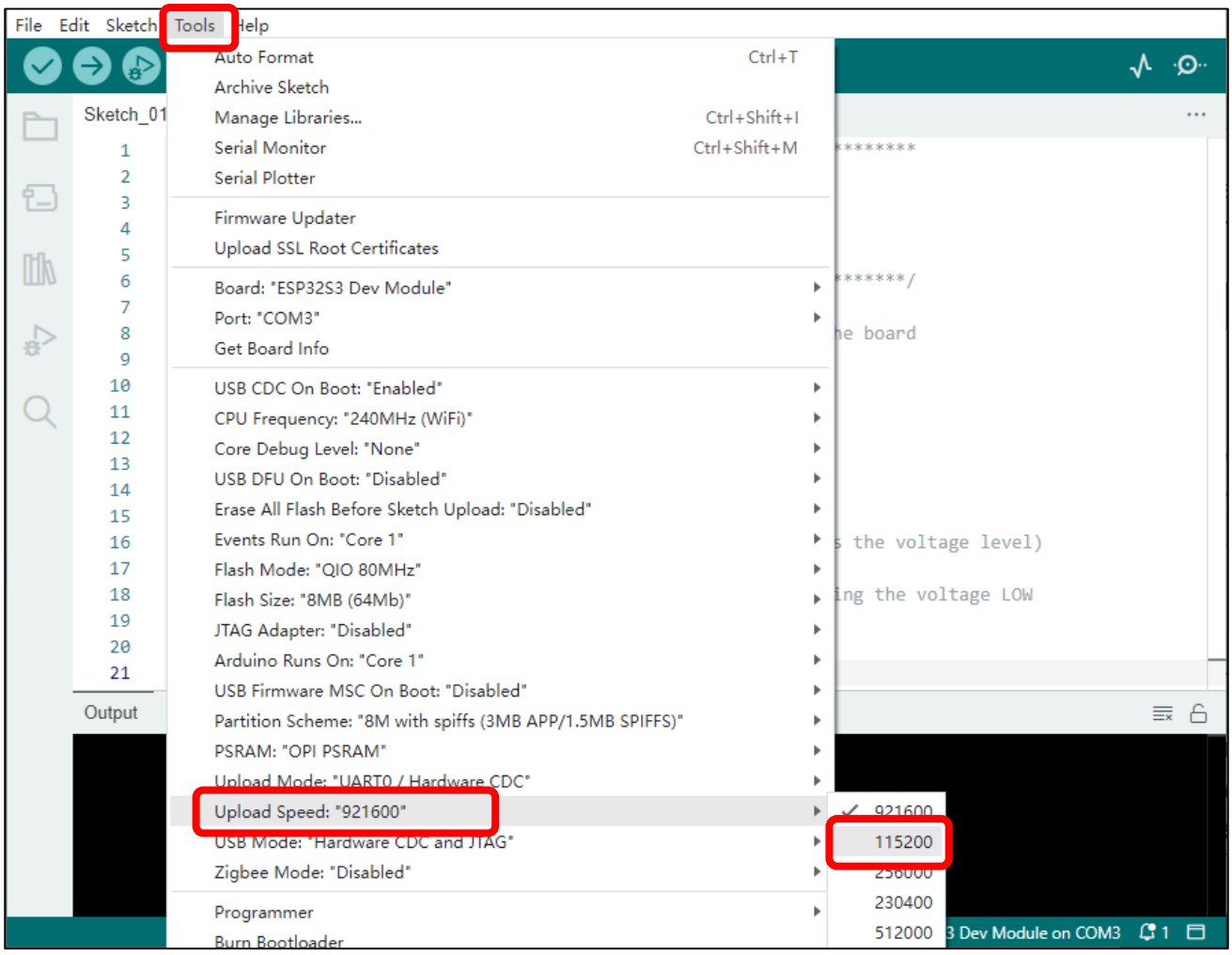



Select the serial port.

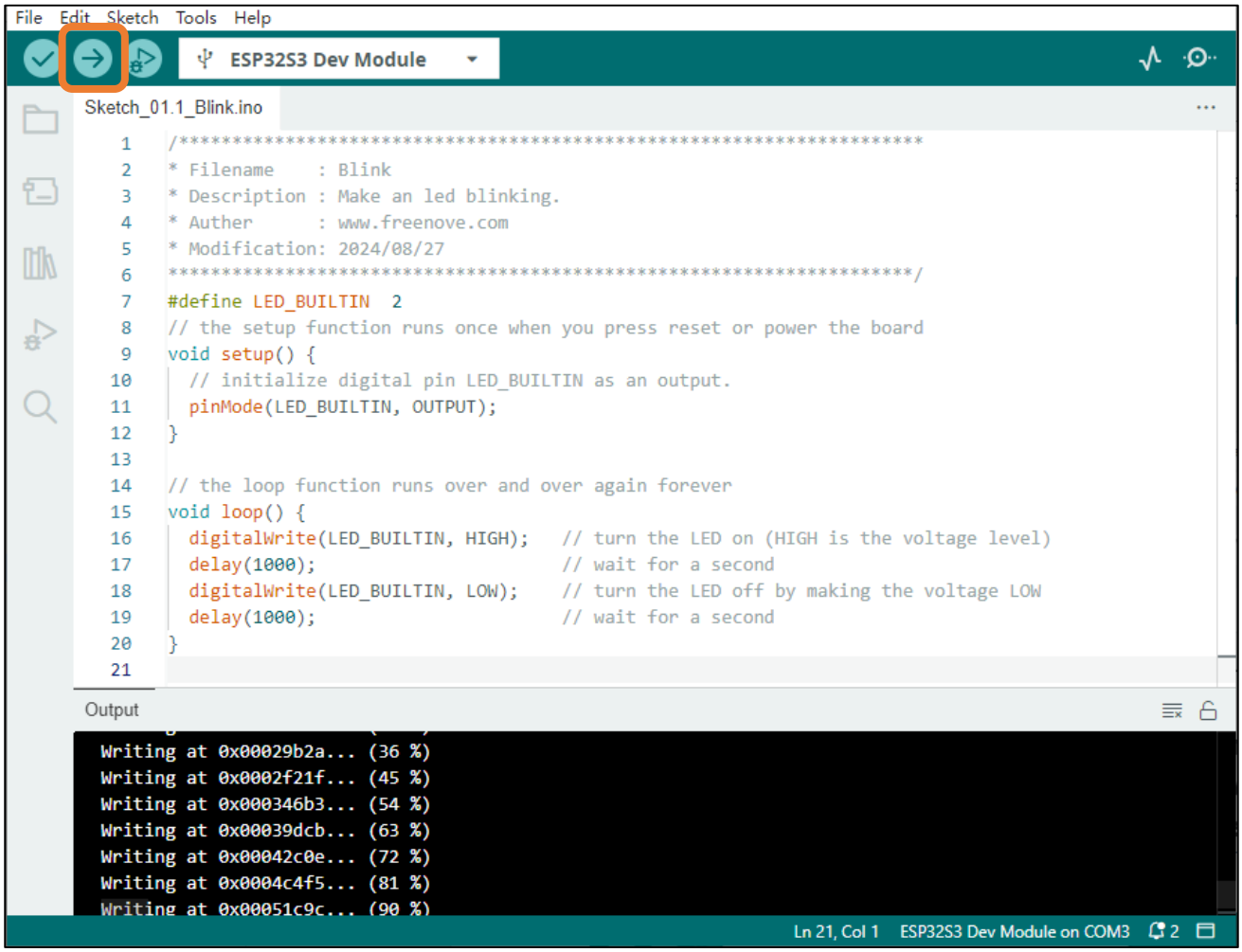
**Note that the computer port number of each user may be different. Please select the correct serial port according to your computer. Taking the window system as an example, my computer recognizes that the communication interface of the ESP32-S3-WROOM is COM3, so I select COM3.**



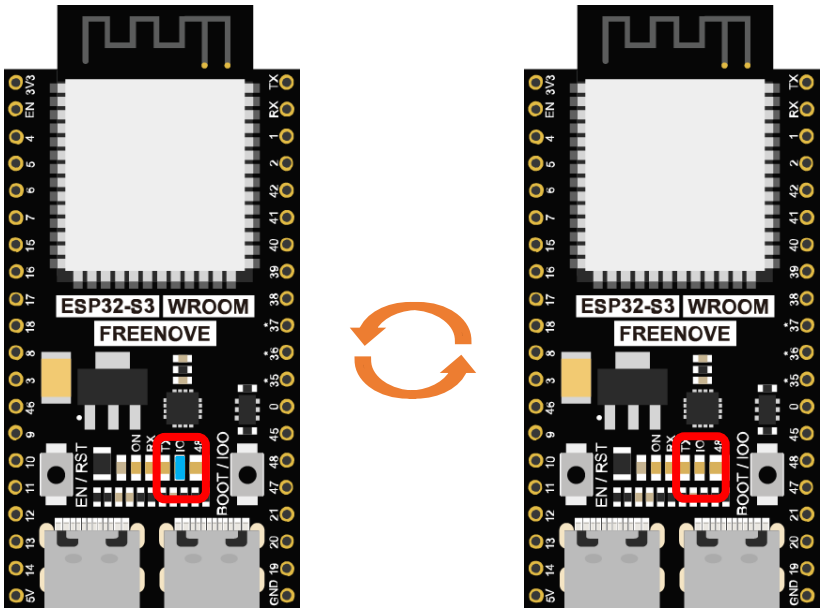
Note: For macOS users, if the uploading fails, please set the baud rate to 115200 before clicking "Upload Using Programmer".



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.

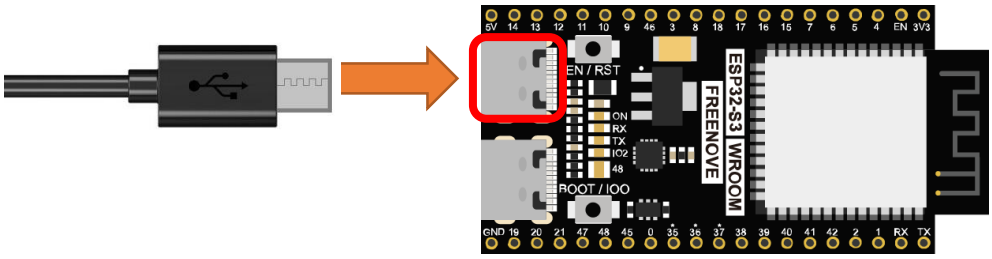



If you have any concerns, please contact us via: support@freenove.com.

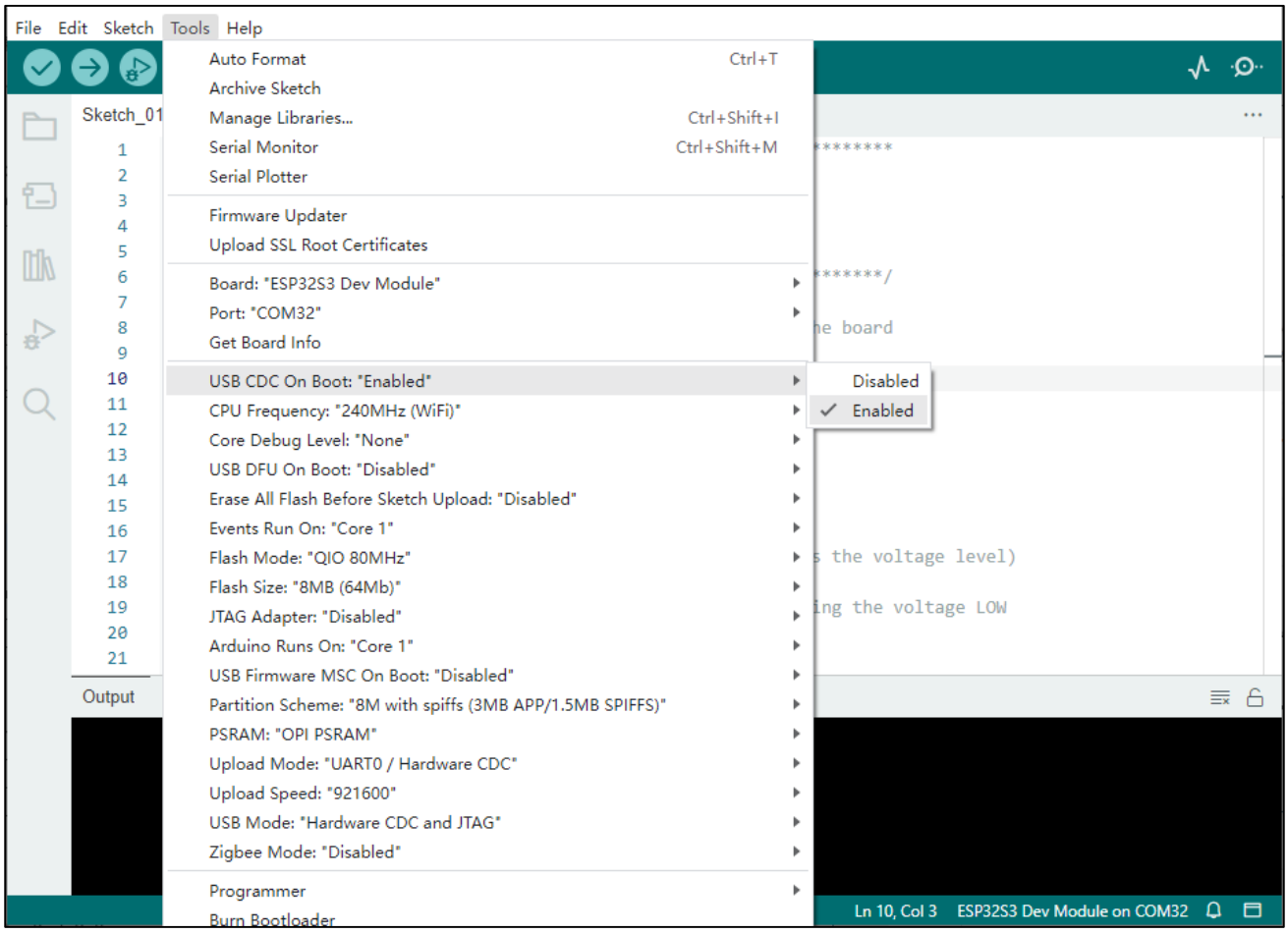
Any concerns? ✉ support@freenove.com

Option 2:

Connect ESP32-S3 WROOM to computer.

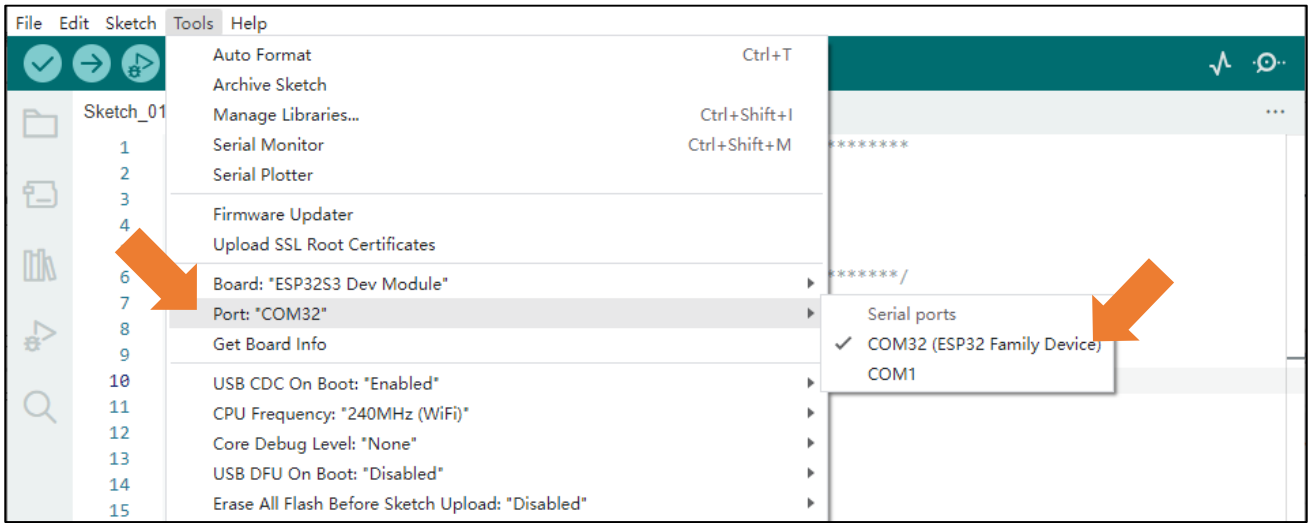



Open Arduino IDE. Click Tools->USB CDC On Boot: Enable.

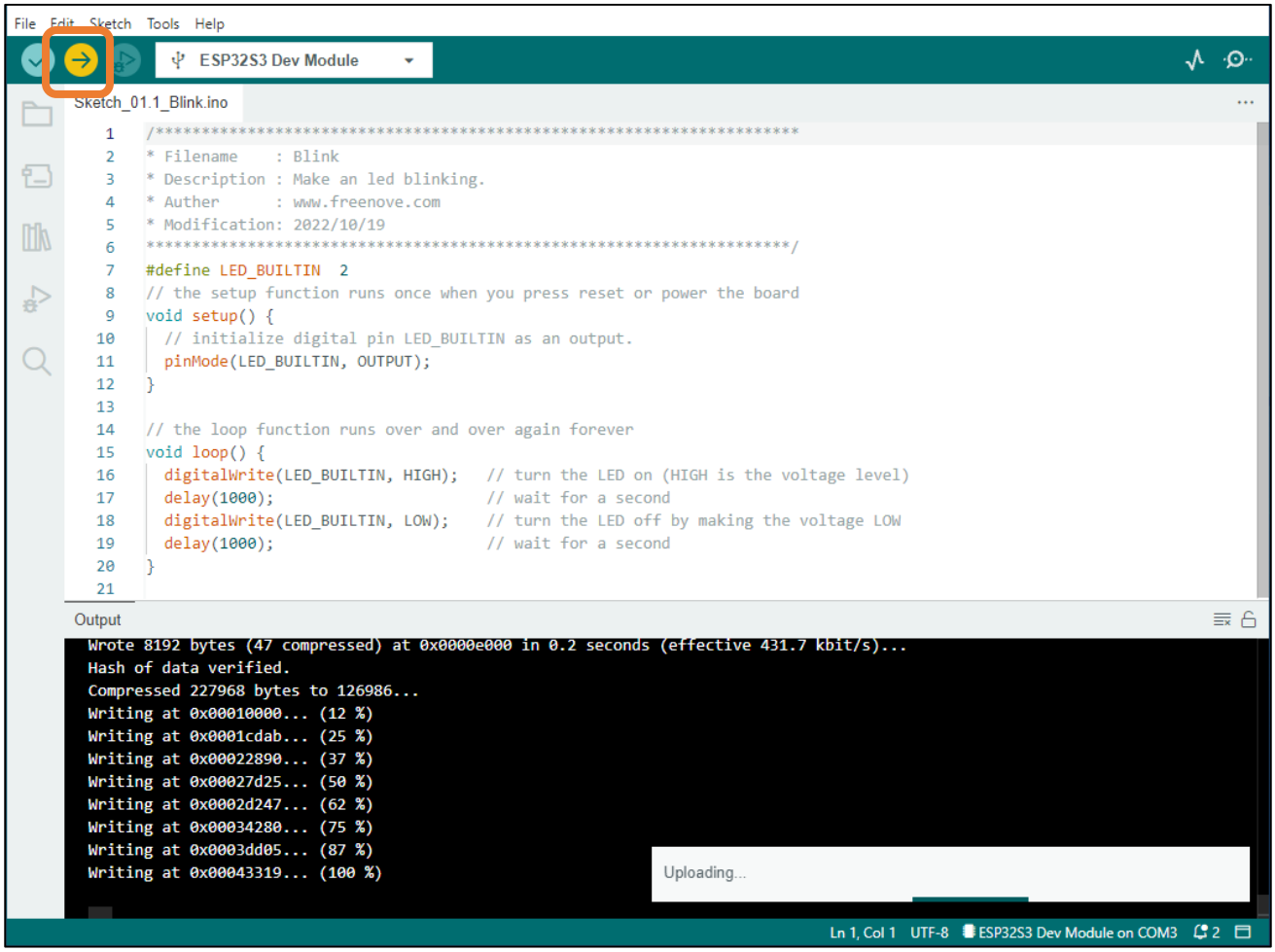


Select the serial port.

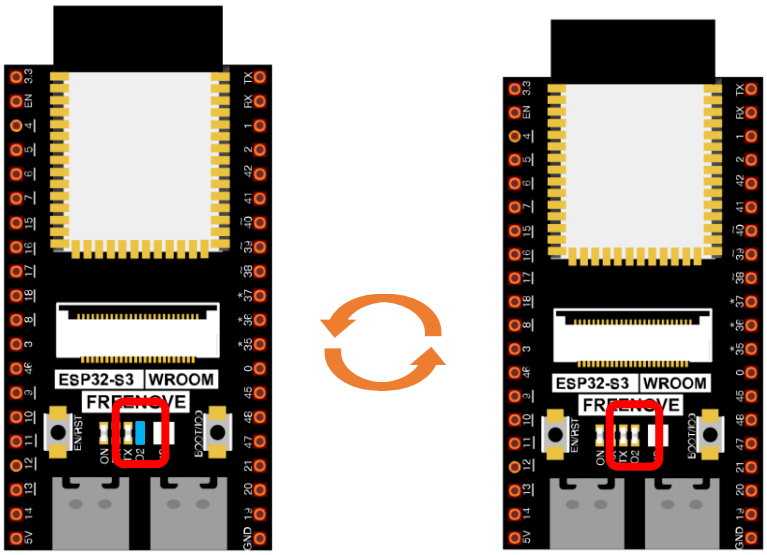
**Note that the computer port number of each user may be different. Please select the correct serial port according to your computer. Taking the window system as an example, my computer recognizes that the communication interface of the ESP32-S3-WROOM is COM25, so I select COM25.**




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:

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

// the loop function runs over and over again forever
void loop() {
  digitalWrite(LED_BUILTIN, HIGH);   // turn the LED on (HIGH is the voltage level)
  delay(1000);                       // wait for a second
  digitalWrite(LED_BUILTIN, LOW);    // turn the LED off by making the voltage LOW
  delay(1000);                       // wait for a second
}
```

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 2 WS2812

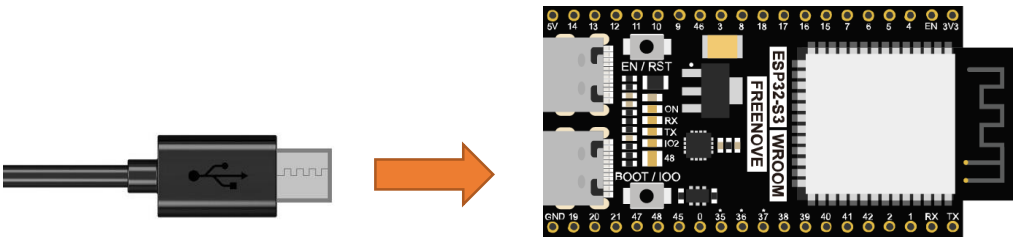
This chapter will help you learn to use a more convenient RGB LED lamp, which requires only one GPIO control and can be connected in infinite series in theory. Each LED can be controlled independently.

## Project 2.1 WS2812

Learn the basic usage of ws2812 and use it to flash red, green, blue and white.

### Circuit

Connect your computer and ESP32-S3 WROOM with a Type C cable.




### Sketch

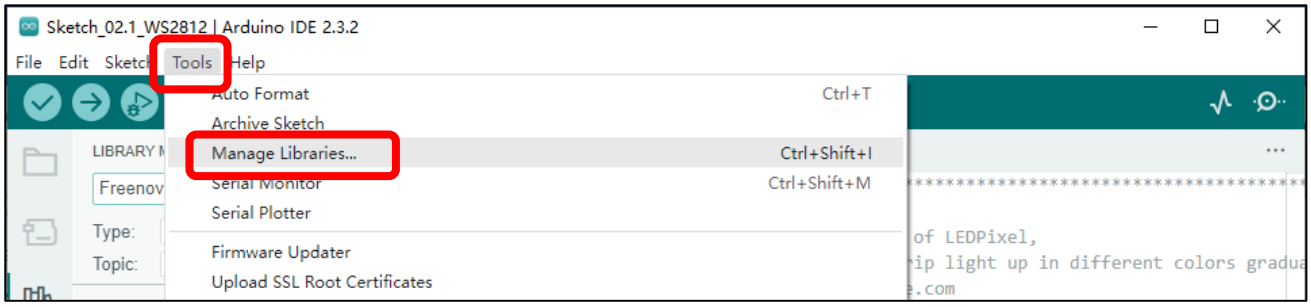
This code uses a library named "Freenove_WS2812_Lib_for_ESP32", if you have not installed it, please do so first.

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.

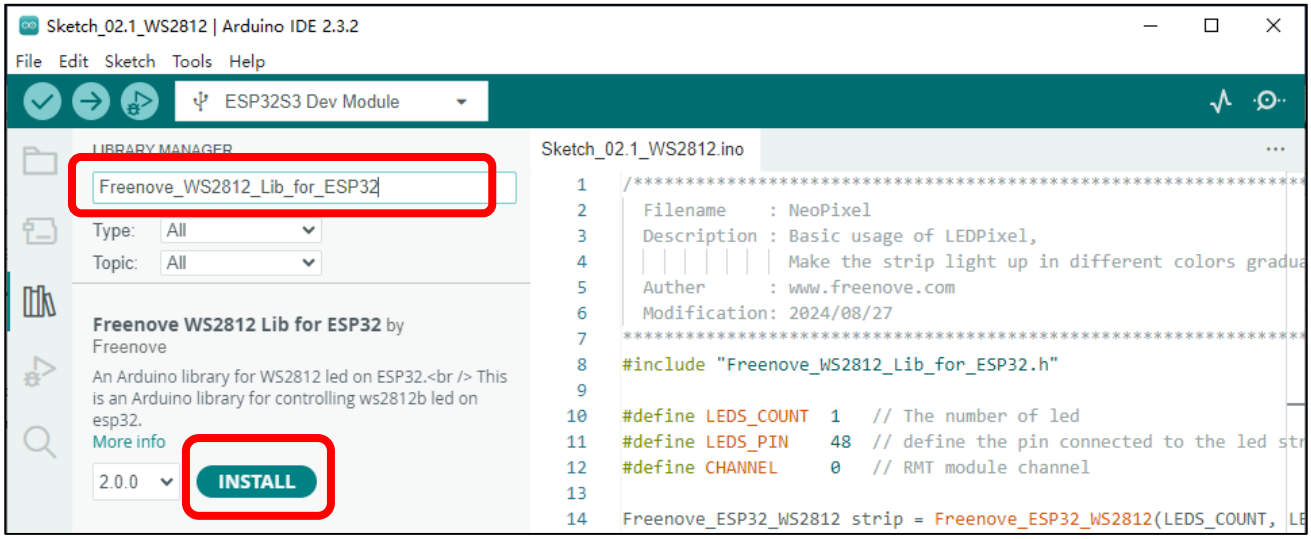
#### How to install the library

There are two ways to add libraries.

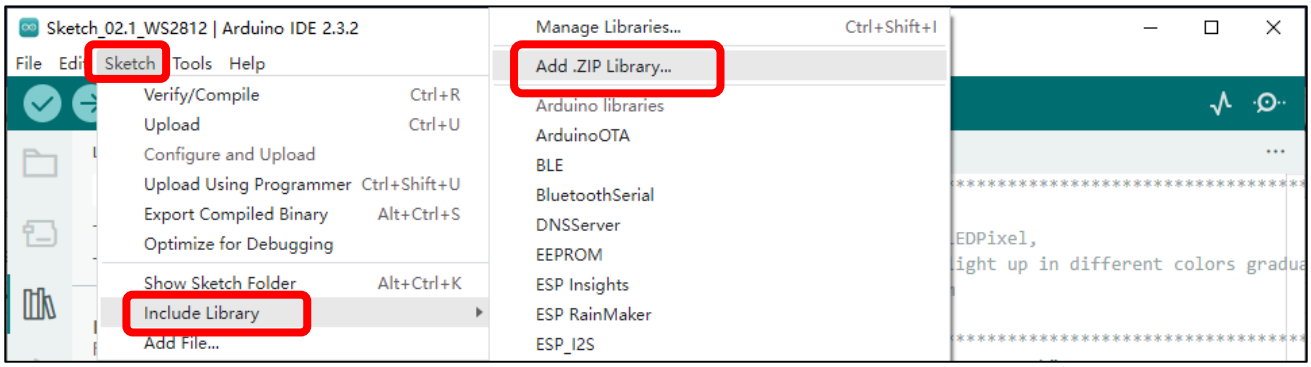
The first way, open the Arduino IDE, click →Manager Libraries.



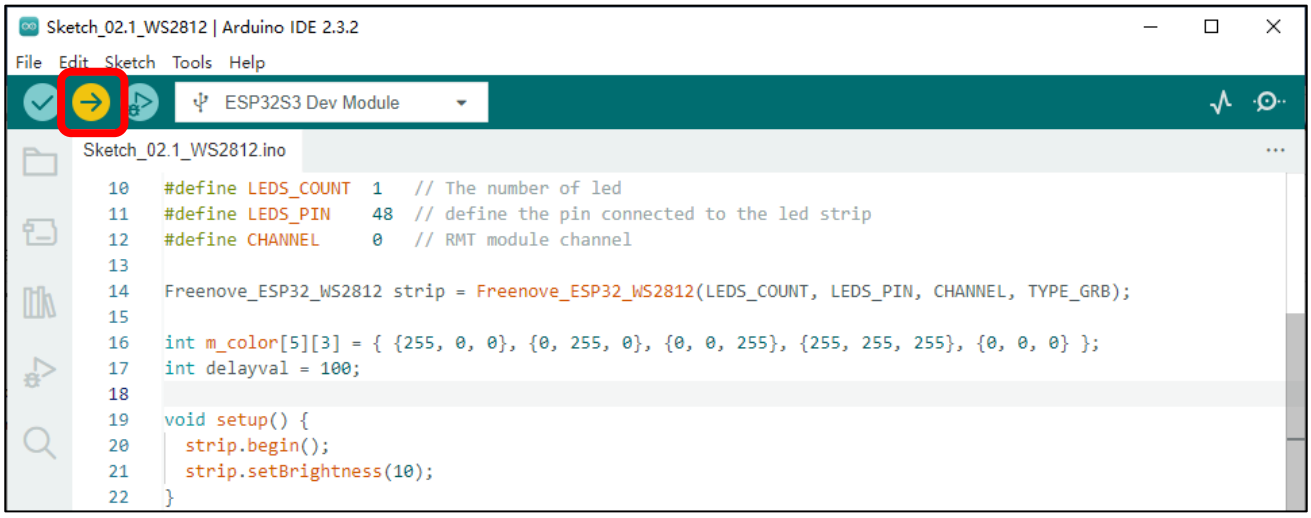
In the window, Library Manager, search for the name of the Library, "Freenove WS2812 Lib for ESP32". Then click Install.



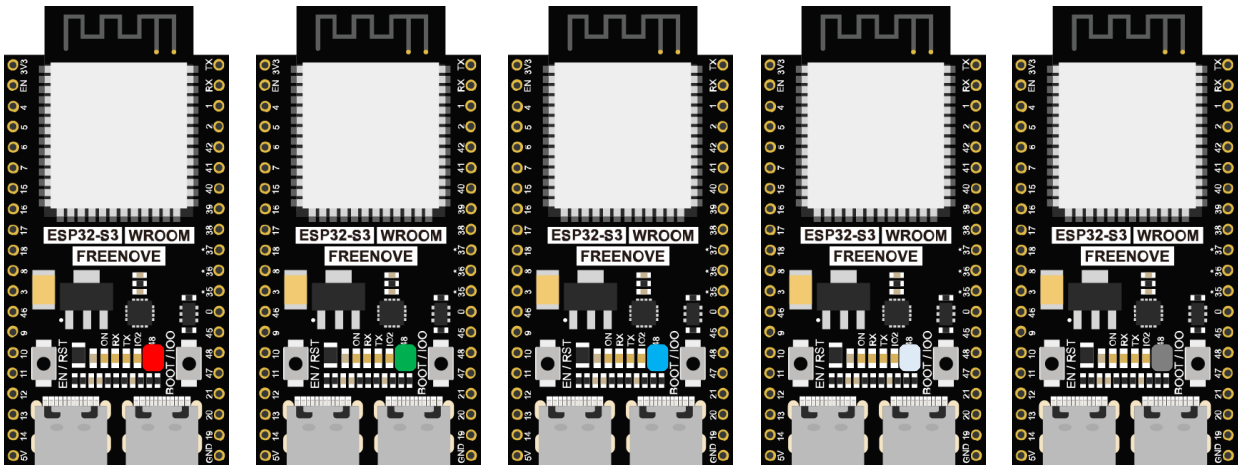

The second way, open Arduino IDE, click Sketch→Include Library→Add .ZIP Library, In the pop-up window, find the file named "./Libraries/Freenove_WS2812_Lib_for_ESP32.Zip" which locates in this directory, and click OPEN.




Sketch_02.1_WS2812




Download the code to ESP32-S3 WROOM and RGB LED begins to light up in red, green, blue, white and black.




The following is the program code:

```
#include "Freenove_WS2812_Lib_for_ESP32.h"

#define LEDS_COUNT  1   // The number of led
#define LEDS_PIN    16 // define the pin connected to the led strip
#define CHANNEL     0   // RMT channel

Freenove_ESP32_WS2812 strip = Freenove_ESP32_WS2812(LEDS_COUNT, LEDS_PIN, CHANNEL, TYPE_GRB);

u8 m_color[5][3] = { {255, 0, 0}, {0, 255, 0}, {0, 0, 255}, {255, 255, 255}, {0, 0, 0} };
int delayval = 100;

void setup() {
    strip.begin();
    strip.setBrightness(10);
}
void loop() {
    for (int j = 0; j < 5; j++) {
        for (int i = 0; i < LEDS_COUNT; i++) {
            strip.setLedColorData(i, m_color[j][0], m_color[j][1], m_color[j][2]);
            strip.show();
            delay(delayval);
        }
        delay(500);
    }
}
```

To use some libraries, first you need to include the library's header file.

```
1  #include "Freenove_WS2812_Lib_for_ESP32.h"
```

Define the pins connected to the ring, the number of LEDs on the ring, and RMT channel values。

```
3  #define LEDS_COUNT  1  // The number of led
4  #define LEDS_PIN    16 // define the pin connected to the led strip
5  #define CHANNEL     0  // RMT channel
```

Use the above parameters to create a ws2812 object strip.

```
7  Freenove_ESP32_WS2812 strip = Freenove_ESP32_WS2812(LEDS_COUNT, LEDS_PIN, CHANNEL, TYPE_GRB);
```

Define the color values to be used, as red, green, blue, white, and black.

```
9  u8 m_color[5][3] = { {255, 0, 0}, {0, 255, 0}, {0, 0, 255}, {255, 255, 255}, {0, 0, 0} };
```

Define a variable to set the time interval for each led to light up. The smaller the value is, the faster it will light up.

```
10 int delayval = 50;
```

Initialize strip() in setup() and set the brightness.

```
13   strip.begin();
14   strip.setBrightness(10);
```

In the loop(), there are two “for” loops, the internal for loop to light the LED one by one, and the external for loop to switch colors. strip.setLedColorData() is used to set the color, but it does not change immediately. Only when strip.show() is called will the color data be sent to the LED to change the color.

```
17     for (int j = 0; j < 5; j++) {
18         for (int i = 0; i < LEDS_COUNT; i++) {
19             strip.setLedColorData(i, m_color[j][0], m_color[j][1], m_color[j][2]);
20             strip.show();
21             delay(delayval);
22         }
23         delay(500);
24     }
```

Reference

**Freenove_ESP32_WS2812(u16 n = 8, u8 pin_gpio = 2, u8 chn = 0, LED_TYPE t = TYPE_GRB)**

Constructor to create a Ws2812 object.

Before each use of the constructor, please add “**#include "Freenove_WS2812_Lib_for_ESP32.h**”

**Parameters**

**n**: The number of led.

**pin_gpio**: A pin connected to an led.

**Chn:** RMT channel, which uses channel 0 by default, has a total of eight channels, 0-7. This means that you can use eight LEDPixel modules for the display at the same time, and these modules do not interfere with each other

**t**: Types of LED.

TYPE_RGB: The sequence of Ws2812 module loading color is red, green and blue.

TYPE_RBG: The sequence of Ws2812 module loading color is red, blue and green.

TYPE_GRB: The sequence of Ws2812 module loading color is green, red and blue.

TYPE_GBR: The sequence of Ws2812 module loading color is green, blue and red.

TYPE_BRG: The sequence of Ws2812 module loading color is blue, red and green.

TYPE_BGR: The sequence of Ws2812 module loading color is blue,g reen and red.

| |
|---|
| `void begin(void);` |
| Initialize the Ws2812 object |
| `void setLedColorData (u8 index, u8 r, u8 g, u8 b);`<br>`void setLedColorData (u8 index, u32 rgb);`<br>`void setLedColor (u8 index, u8 r, u8 g, u8 b);`<br>`void setLedColor (u8 index, u32 rgb);` |
| Set the color of led with order number n. |
| `void show(void);` |
| Send the color data to the led and display the set color immediately. |
| `void setBrightness(uint8_t);` |
| Set the brightness of the LED. |

If you want to learn more about this library, you can visit the following website:
https://github.com/Freenove/Freenove_WS2812_Lib_for_ESP32

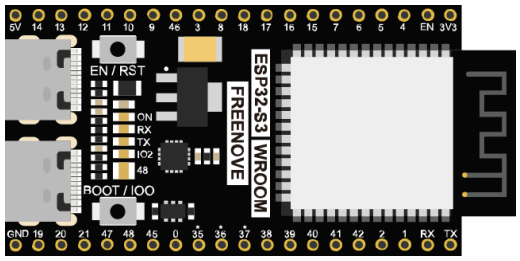
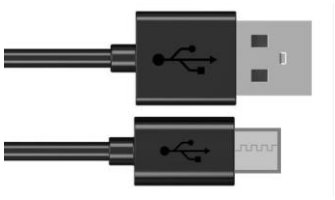
# Chapter 3 Serial Communication

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

## Project 3.1 Serial Print

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

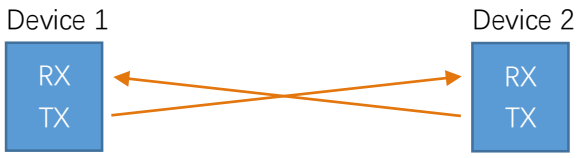
### Component List

| | |
|---|---|
| ESP32-S3 WROOM x1  | Type C Wire x1  |

### Related knowledge

#### Serial communication

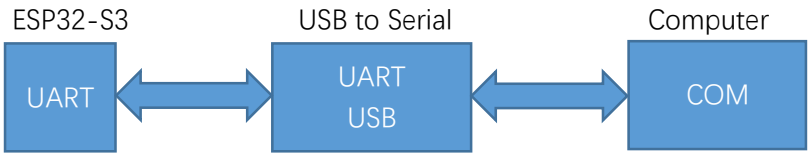
Serial communication generally refers to the Universal Asynchronous Receiver/Transmitter (UART), which is commonly used in electronic circuit communication. It has two communication lines, one is responsible for sending data (TX line) and the other for receiving data (RX line). The serial communication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

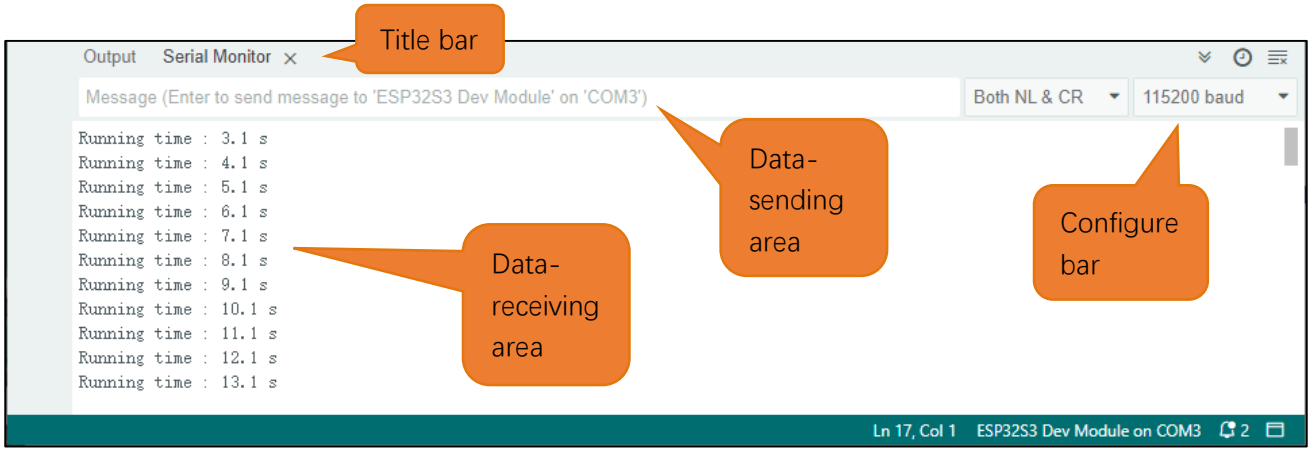
### Serial port on ESP32-S3

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



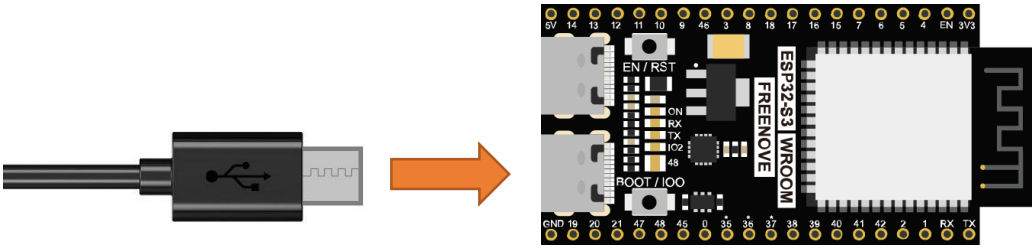

Arduino Software also uploads code to Freenove ESP32-S3 through the serial connection.
Your computer identifies serial devices connecting to it as COMx. We can use the Serial Monitor window of Arduino Software to communicate with Freenove ESP32-S3, connect Freenove ESP32-S3 to computer through the Type C 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".



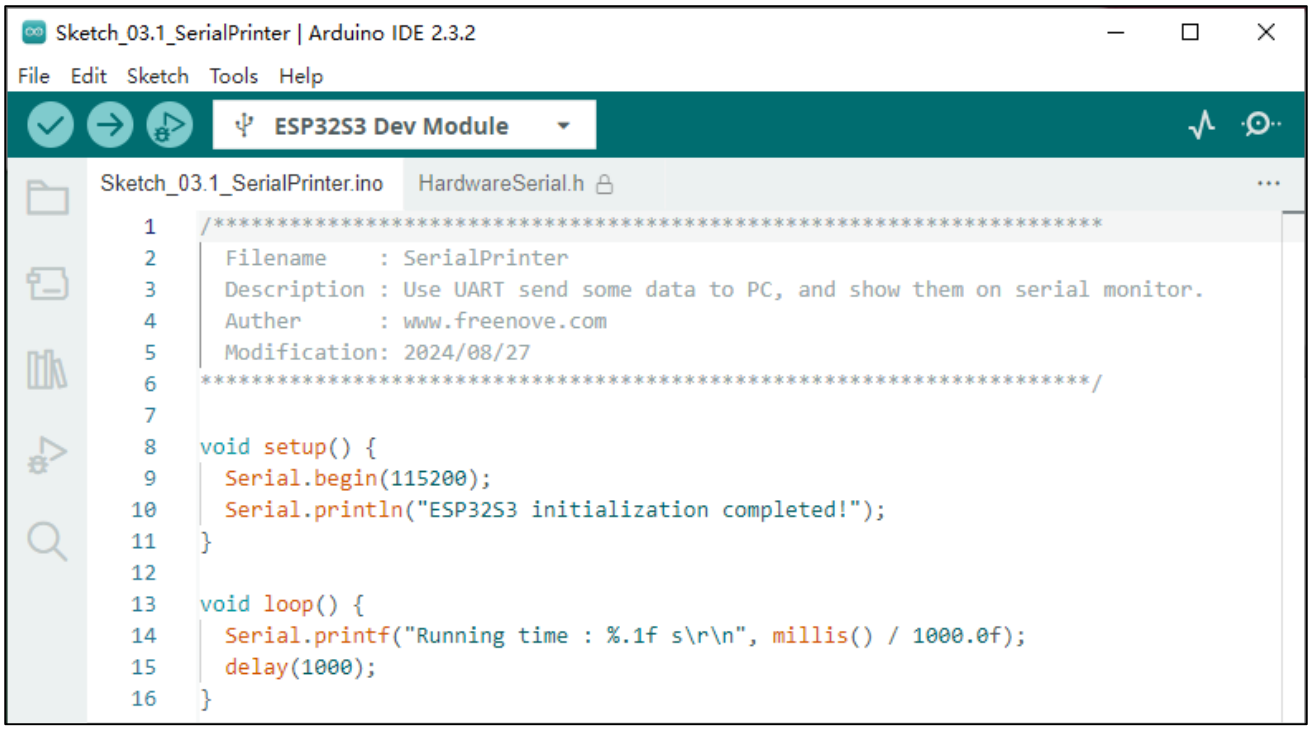

## Circuit

Connect Freenove ESP32-S3 to the computer with Type C cable.



## Sketch

### Sketch_03.1_SerialPrinter




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:


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.

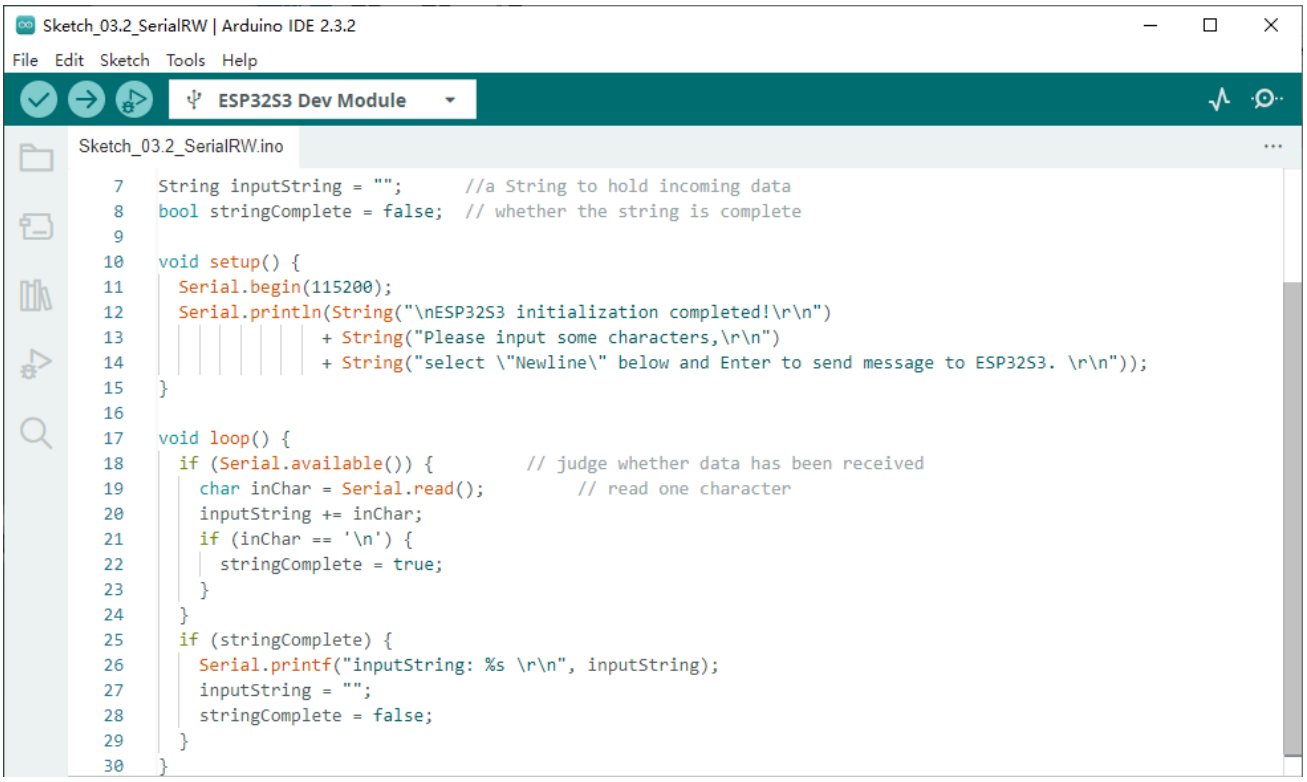
# Project 3.2 Serial Read and Write

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_03.2_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

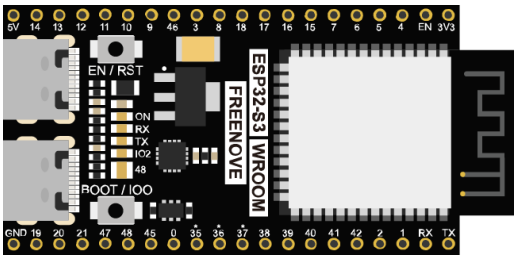
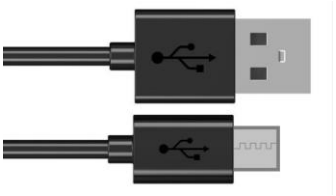
| String();<br>Constructs an instance of the String class.<br>For more information, please visit<br>https://www.arduino.cc/reference/en/language/variables/data-types/stringobject/ |
|---|
| int available(void);<br>Get the number of bytes (characters) available for reading from the serial port. This is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

# Chapter 4 BLE

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

## Project 4.1 Bluetooth Low Energy Data Passthrough

### Component List

| | |
|---|---|
| ESP32-S3 WROOM x1  | Type C Wire 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.

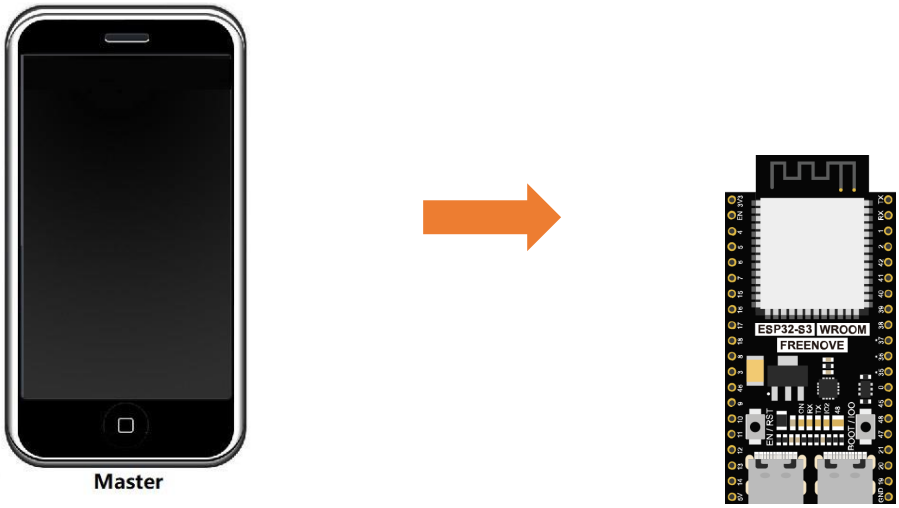
#### Master mode

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

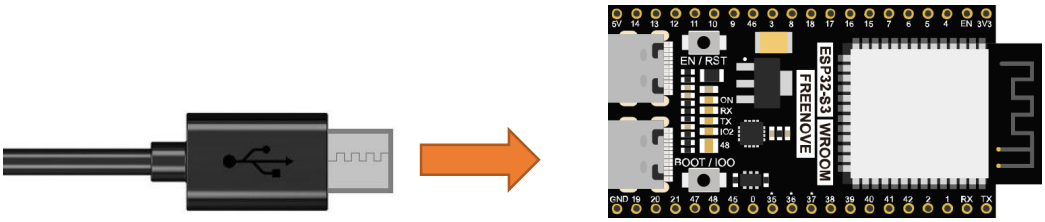
The Bluetooth module in slave mode can only accept connection request from a host computer, but cannot initiate a connection request. After connecting with a host device, it can send data to or receive from the host device.

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 Type C cable.

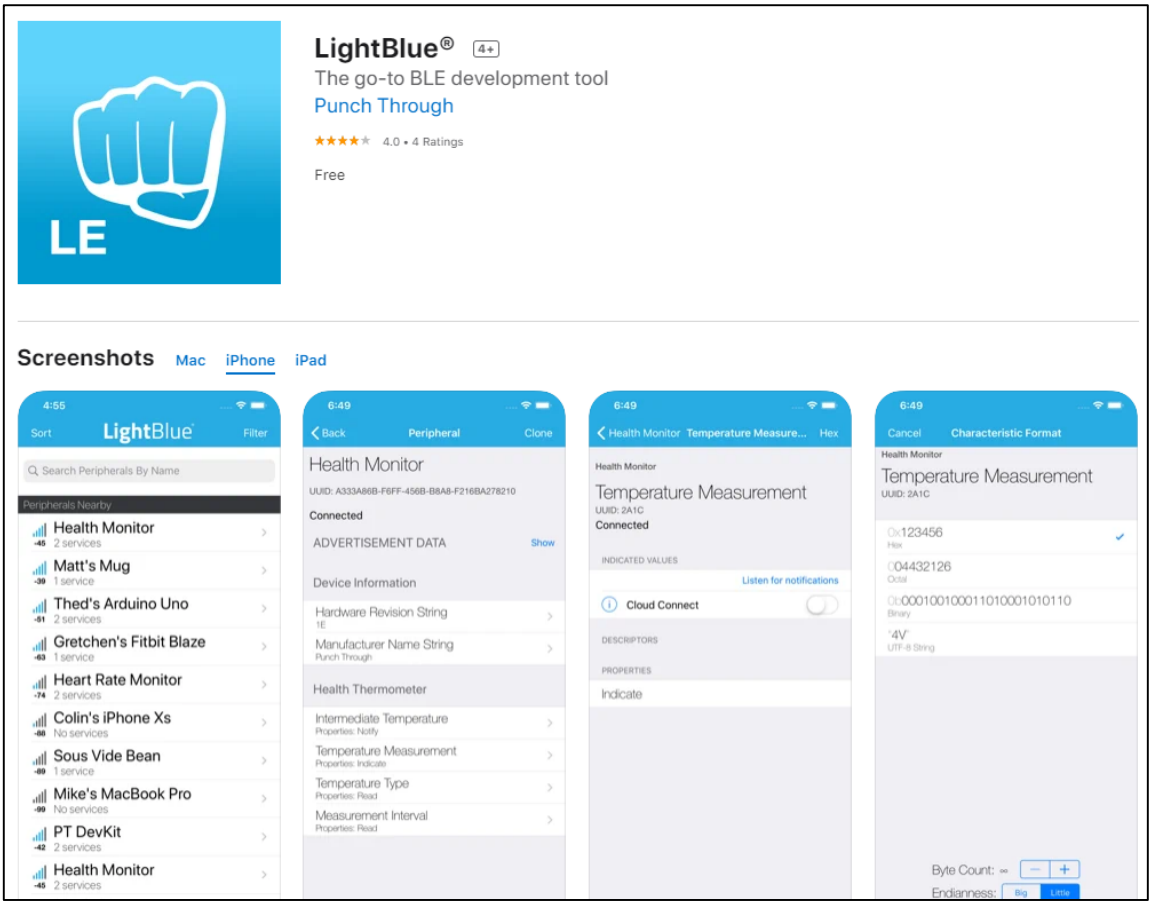


# Sketch

## Lightblue

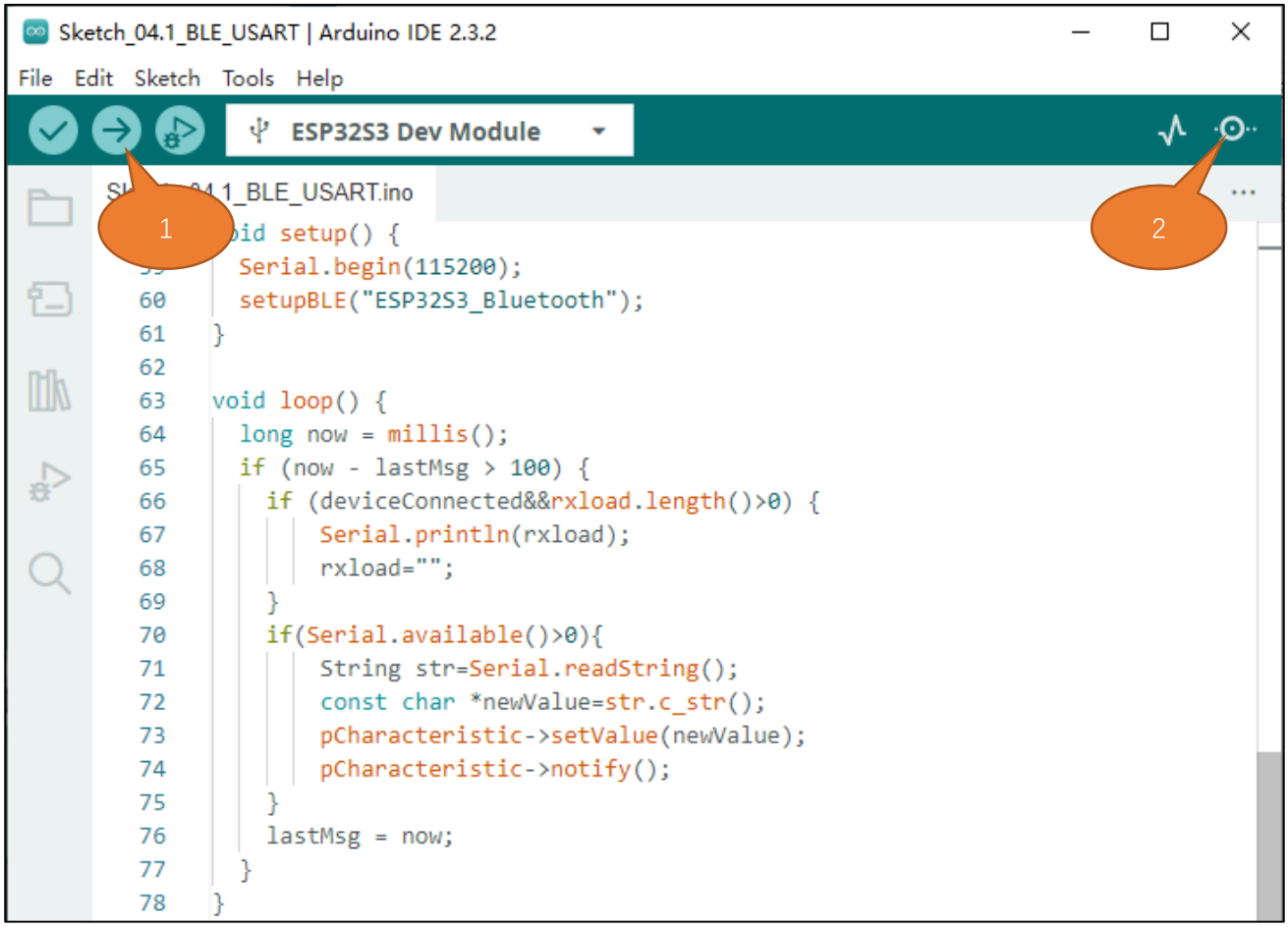
If you can't install Serial Bluetooth on your phone, try LightBlue.If you do not have this software installed on your phone, you can refer to this link:

https://apps.apple.com/us/app/lightblue/id557428110

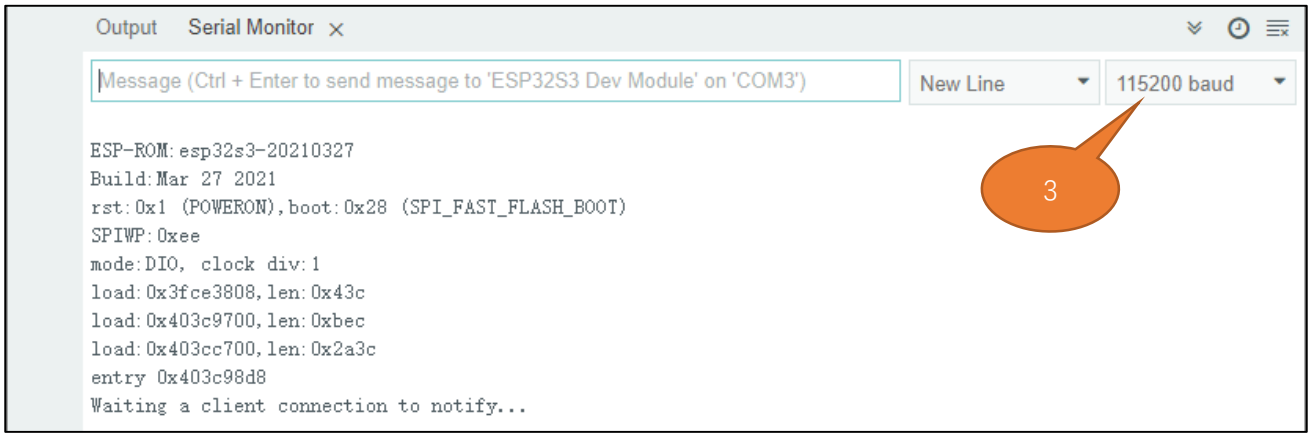


Step1. Upload the code of Project 4.1 to ESP32-S3.

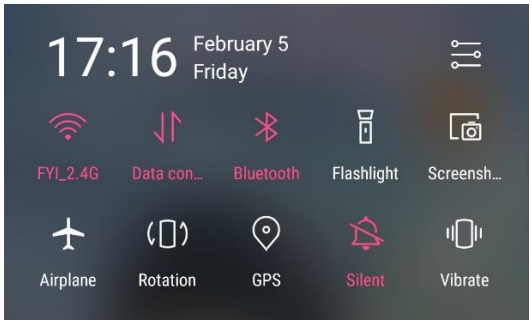
Step2. Click on serial monitor.




Step3. Set baud rate to 115200.

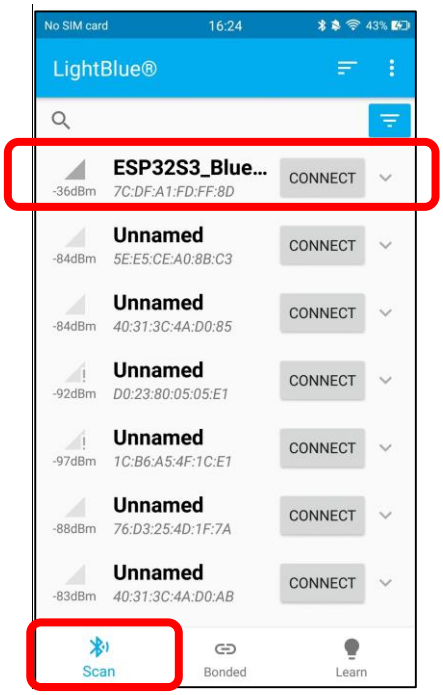


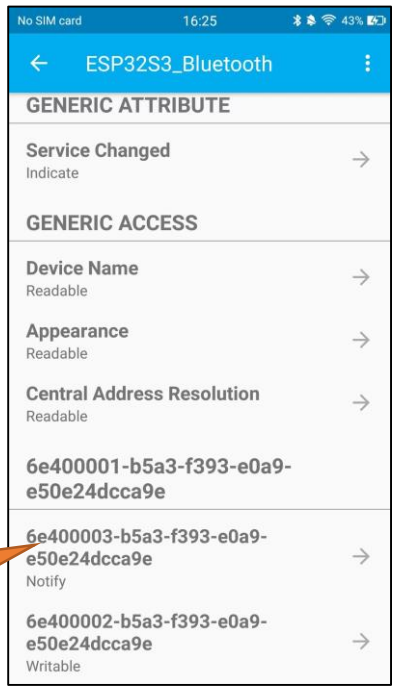
Turn ON Bluetooth on your phone, and open the Lightblue APP.





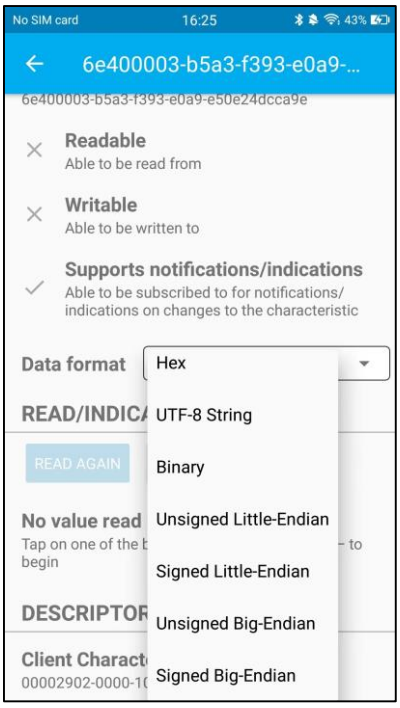
In the Scan page, swipe down to refresh the name of Bluetooth that the phone searches for. Click ESP32S3_Bluetooth.

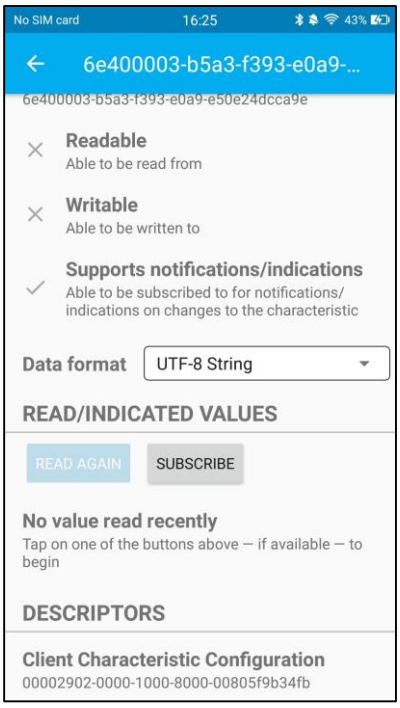


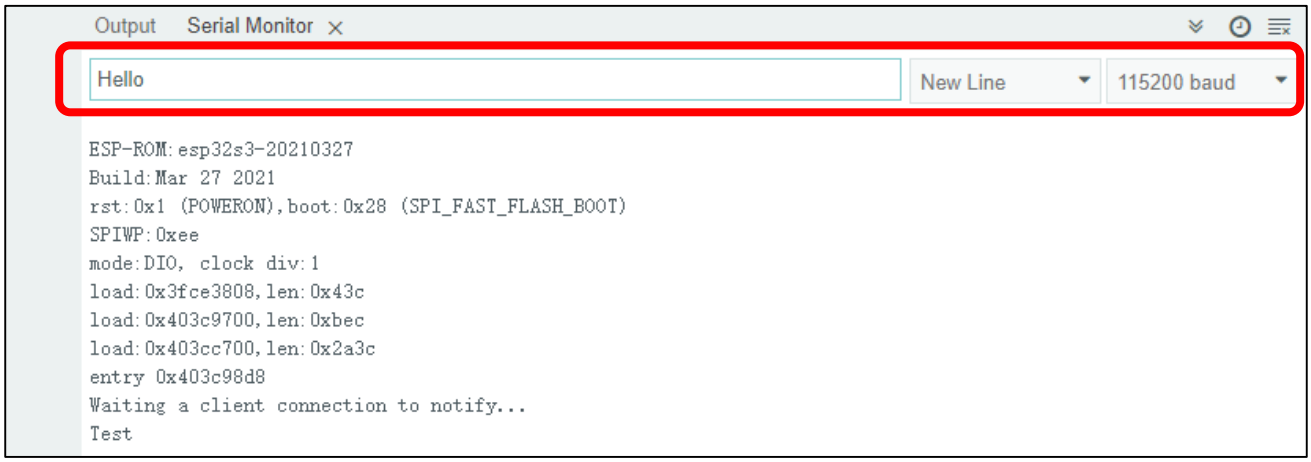

Receive

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.

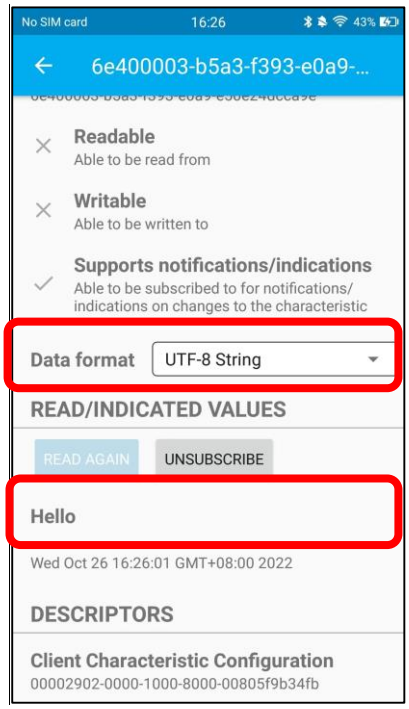


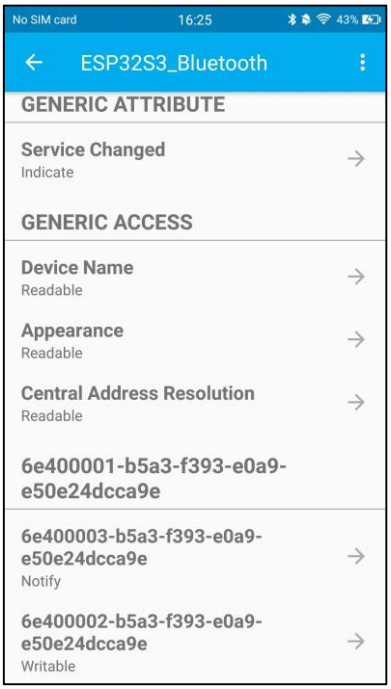

Back to the serial monitor on your computer. You can type anything in the left border of Send, and then click Send.

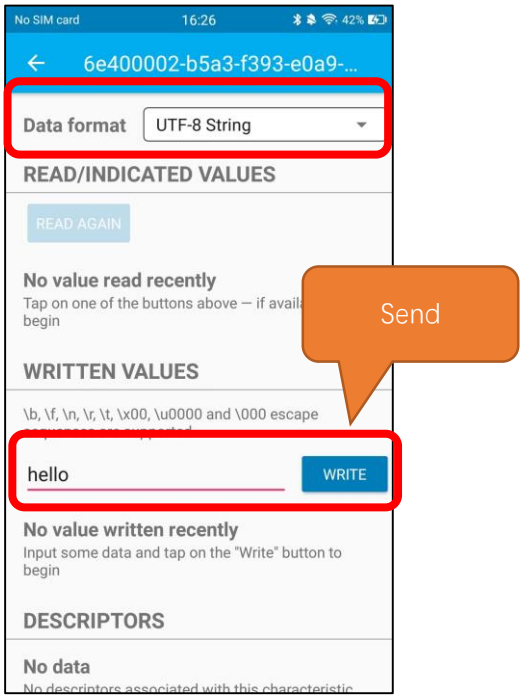


And then you can see the mobile Bluetooth has received the message.

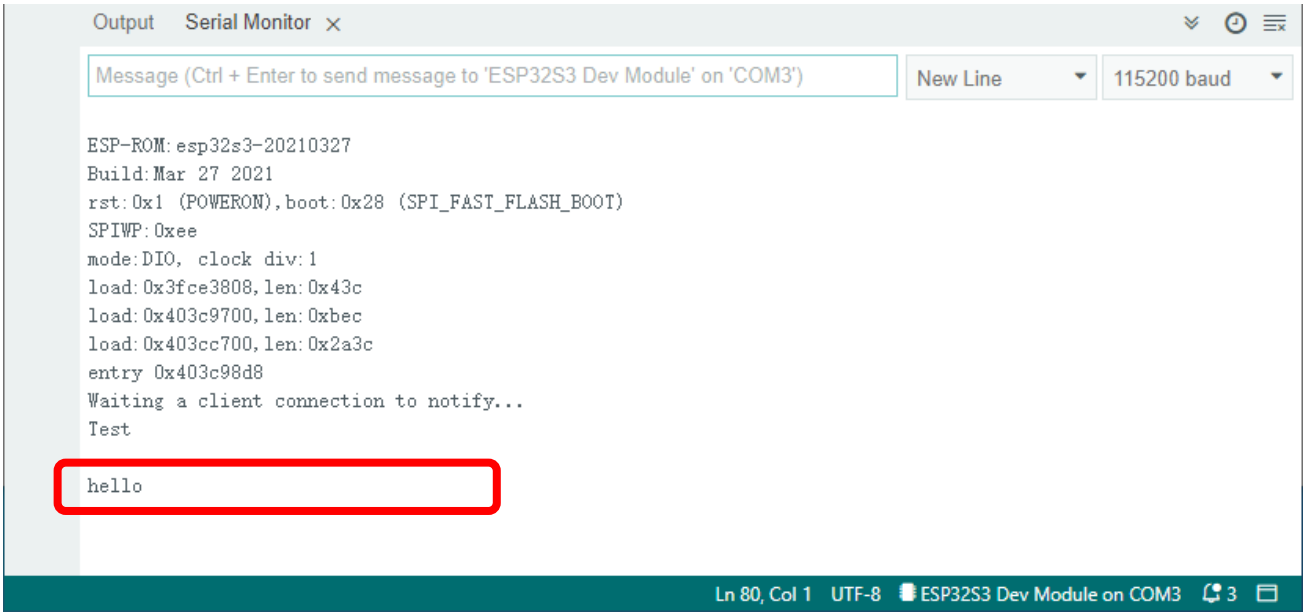



Similarly, you can select "Send" on your phone. Set Data format, and then enter anything in the sending box and click Write to send.



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



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:

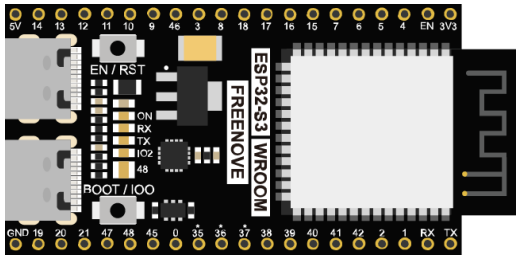
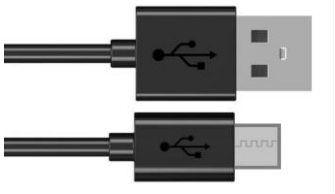
1. Create a BLE Server
2. Create a BLE Service
3. Create a BLE Characteristic on the Service
4. Create a BLE Descriptor on the characteristic
5. Start the service.
6. Start advertising.

```
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 4.2 Bluetooth Control LED

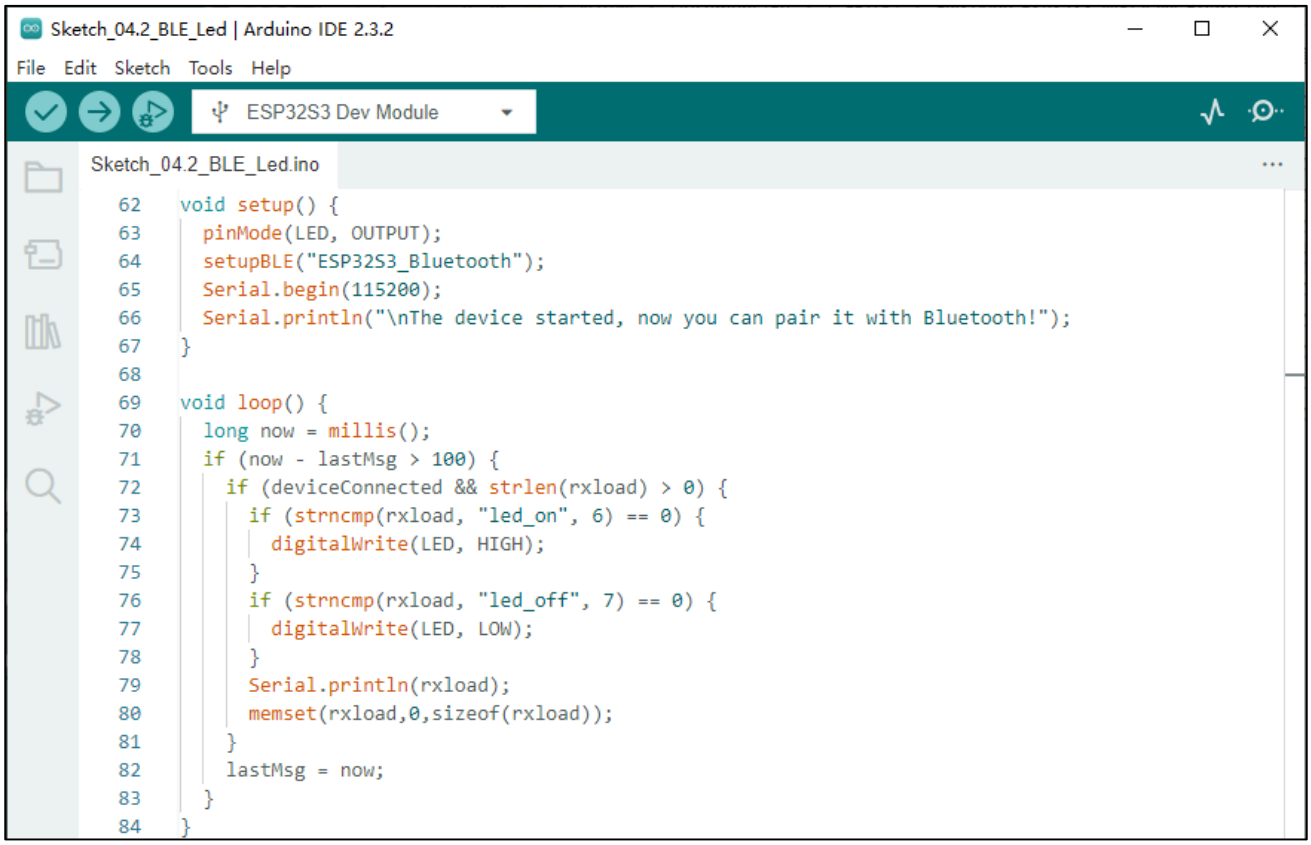
In this section, we will control the LED with Bluetooth.

## Component List

| | |
|---|---|
| ESP32-S3 WROOM x1  | Type C Wire x1  |

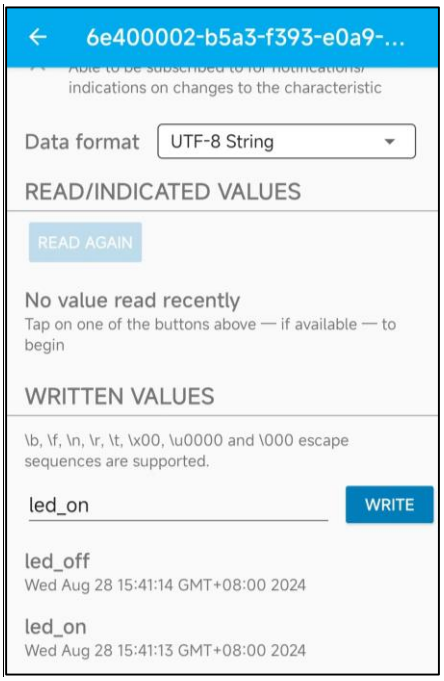
## Sketch

### Sketch_04.2_BLE_Led

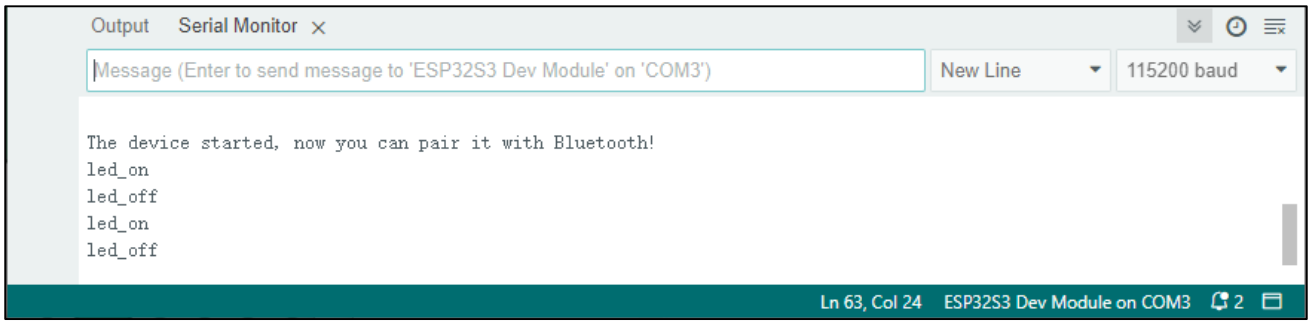



```
62  void setup() {
63    pinMode(LED, OUTPUT);
64    setupBLE("ESP32S3_Bluetooth");
65    Serial.begin(115200);
66    Serial.println("\nThe device started, now you can pair it with Bluetooth!");
67  }
68
69  void loop() {
70    long now = millis();
71    if (now - lastMsg > 100) {
72      if (deviceConnected && strlen(rxload) > 0) {
73        if (strncmp(rxload, "led_on", 6) == 0) {
74          digitalWrite(LED, HIGH);
75        }
76        if (strncmp(rxload, "led_off", 7) == 0) {
77          digitalWrite(LED, LOW);
78        }
79        Serial.println(rxload);
80        memset(rxload,0,sizeof(rxload));
81      }
82      lastMsg = now;
83    }
84  }
```

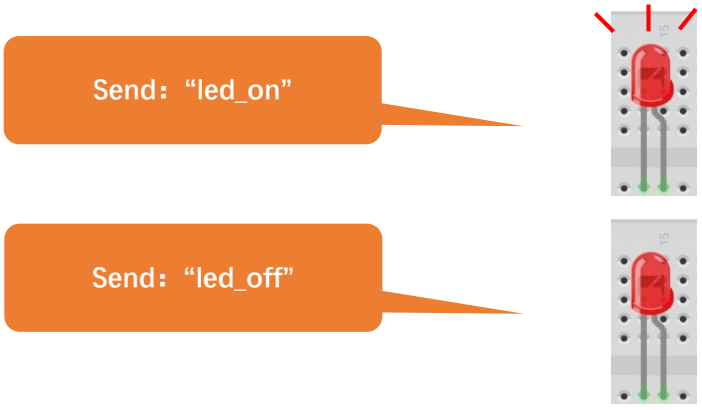
Compile and upload code to **ESP32S3_Blueooth**. The operation of the APP is the same as 27.1, you only need to change the sending content to "**led_on**" and "**led_off**" to operate LEDs on the ESP32-S3 WROOM.
Data sent from mobile APP:




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);
```

```
44    BLEServer *pServer = BLEDevice::createServer();
45    pServer->setCallbacks(new MyServerCallbacks());
46    BLEService *pService = pServer->createService(SERVICE_UUID);
47    pCharacteristic = pService->createCharacteristic(CHARACTERISTIC_UUID_TX,
      BLECharacteristic::PROPERTY_NOTIFY);
48    pCharacteristic->addDescriptor(new BLE2902());
49    BLECharacteristic *pCharacteristic = pService->createCharacteristic(CHARACTERISTIC_UUID_RX,
      BLECharacteristic::PROPERTY_WRITE);
50    pCharacteristic->setCallbacks(new MyCallbacks());
51    pService->start();
52    pServer->getAdvertising()->start();
53    Serial.println("Waiting a client connection to notify...");
54  }
55
56  void setup() {
57    pinMode(LED, OUTPUT);
58    setupBLE("ESP32S3_Bluetooth");
59    Serial.begin(115200);
60    Serial.println("\nThe device started, now you can pair it with Bluetooth!");
61  }
62
63  void loop() {
64    long now = millis();
65    if (now - lastMsg > 100) {
66      if (deviceConnected && strlen(rxload) > 0) {
67        if (strncmp(rxload, "led_on", 6) == 0) {
68          digitalWrite(LED, HIGH);
69        }
70        if (strncmp(rxload, "led_off", 7) == 0) {
71          digitalWrite(LED, LOW);
72        }
73        Serial.println(rxload);
74        memset(rxload,0,sizeof(rxload));
75      }
76      lastMsg = now;
77    }
78  }
```

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 "ESP32S3_Bluetooth".

```
  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

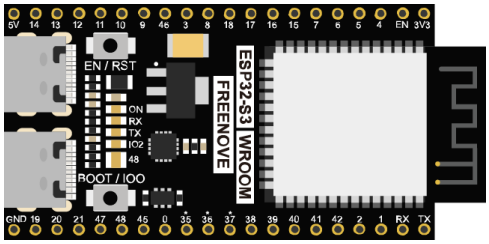
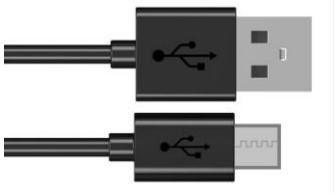
n: the number of byte to be replaced

# Chapter 5 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 5.1 Station mode

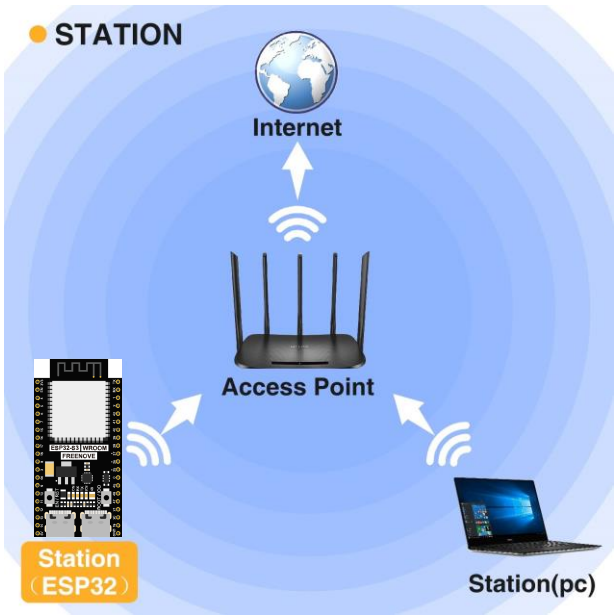
### Component List

| | |
|---|---|
| ESP32-S3 WROOM x1  | Type C Wire x1  |

### Component knowledge

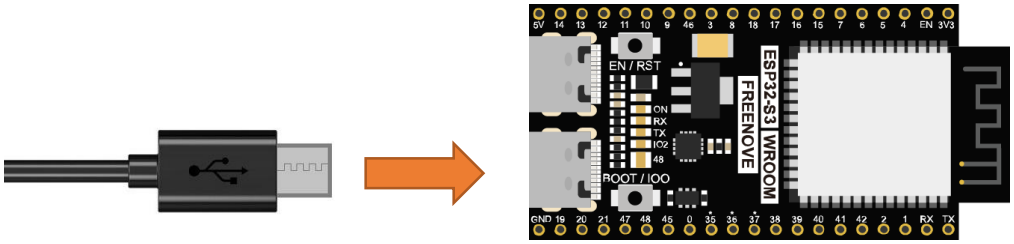
#### Station mode

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



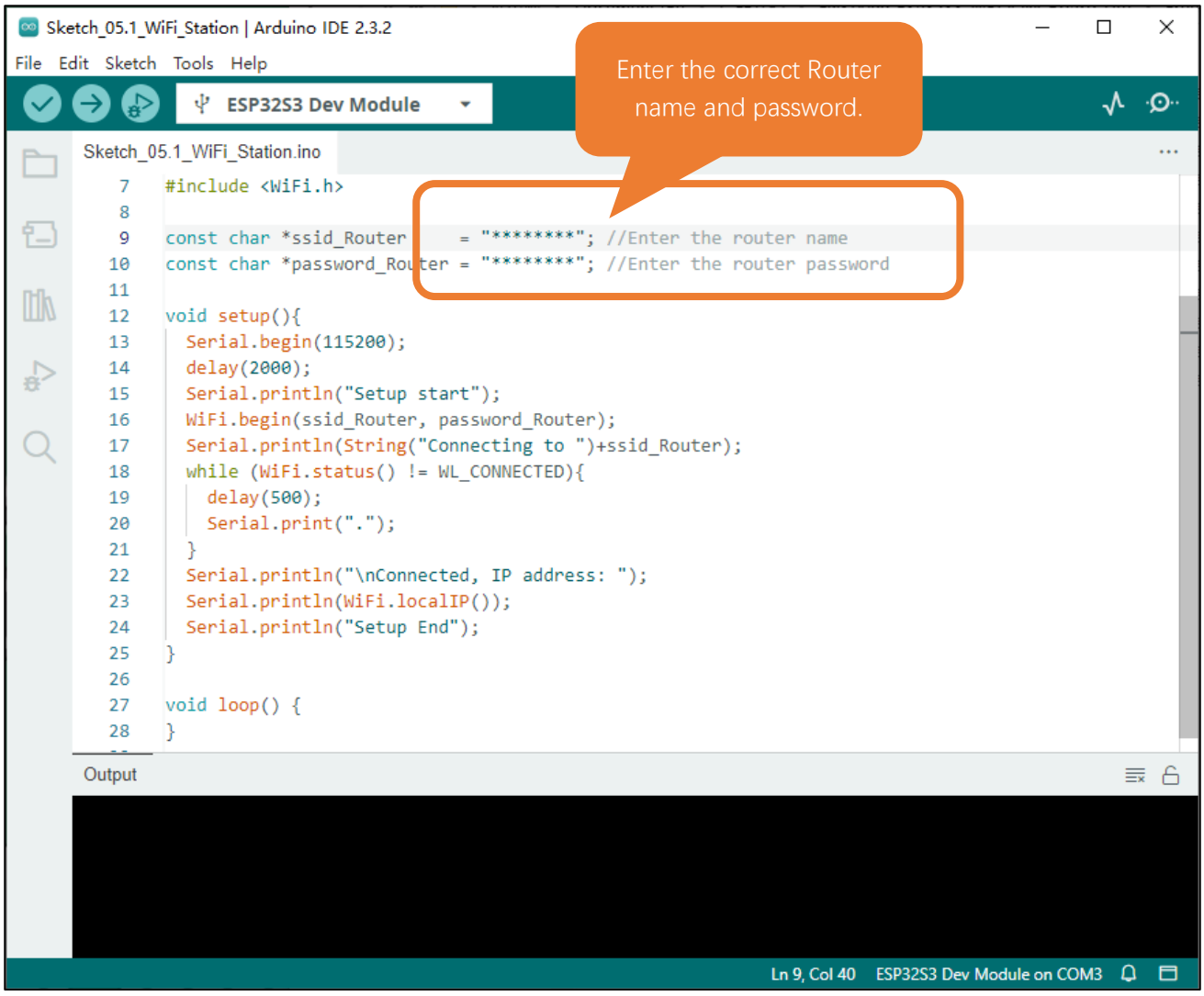
## Circuit

Connect Freenove ESP32-S3 WROOM to the computer using the Type C cable.



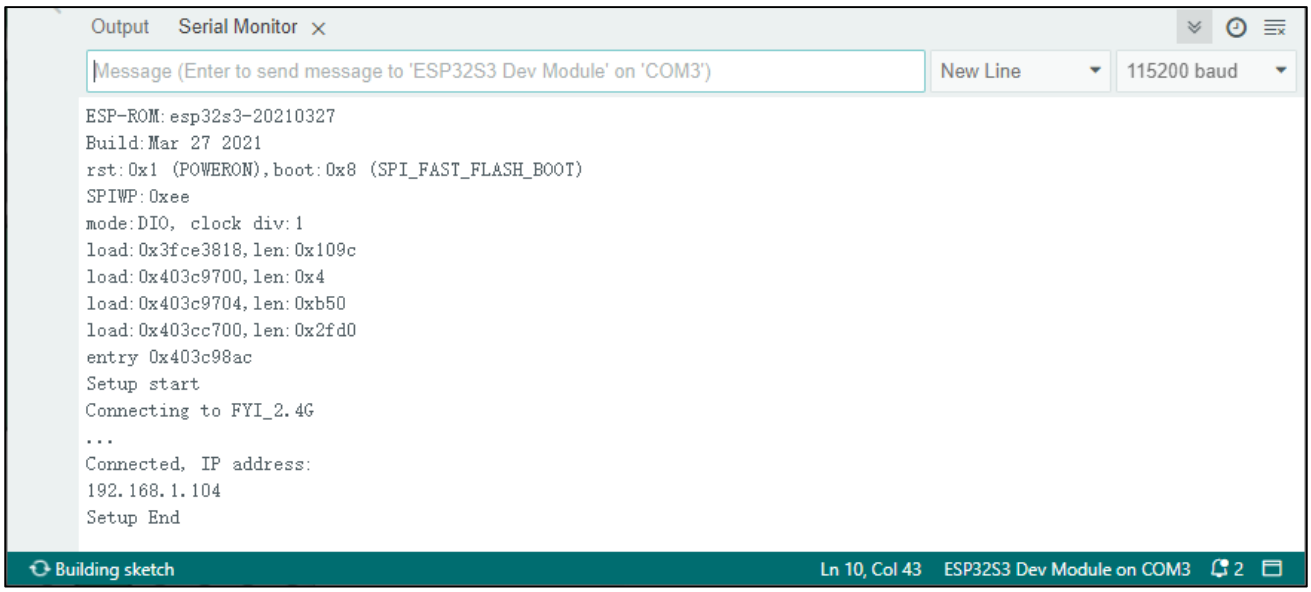

## Sketch

### Sketch_05.1_WiFi_Station




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.

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

Enter correct router name and password.

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

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

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

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

```
12   while (WiFi.status() != WL_CONNECTED){
13     delay(500);
14     Serial.print(".");
15   }
```

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

```
17   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 WROOM 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 WROOM 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

**setAutoReconnect(boolen)**: set automatic reconnection Every time ESP32-S3 WROOM disconnects WiFi, it will reconnect to WiFi automatically.

# Project 5.2 AP mode

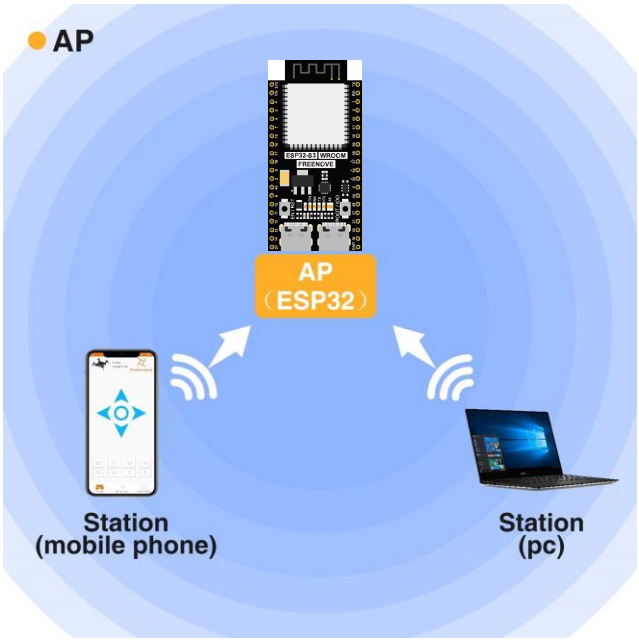
## Component List & Circuit

Component List & Circuit are the same as in Project 5.1.

## Component knowledge

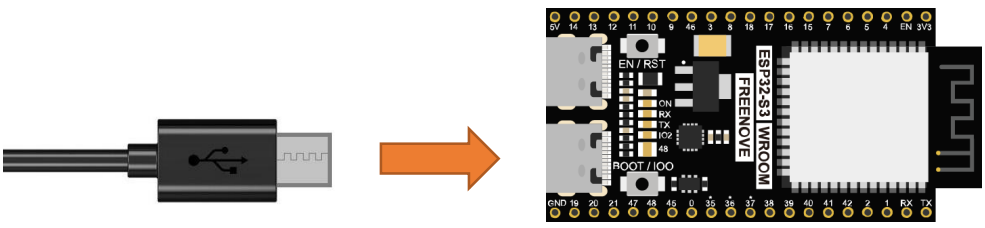
### AP mode

When ESP32-S3 WROOM 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 WROOM is used as a hotspot. If a mobile phone or PC wants to communicate with ESP32-S3 WROOM , it must be connected to the hotspot of ESP32-S3 WROOM . Only after a connection is established with ESP32-S3 WROOM can they communicate.

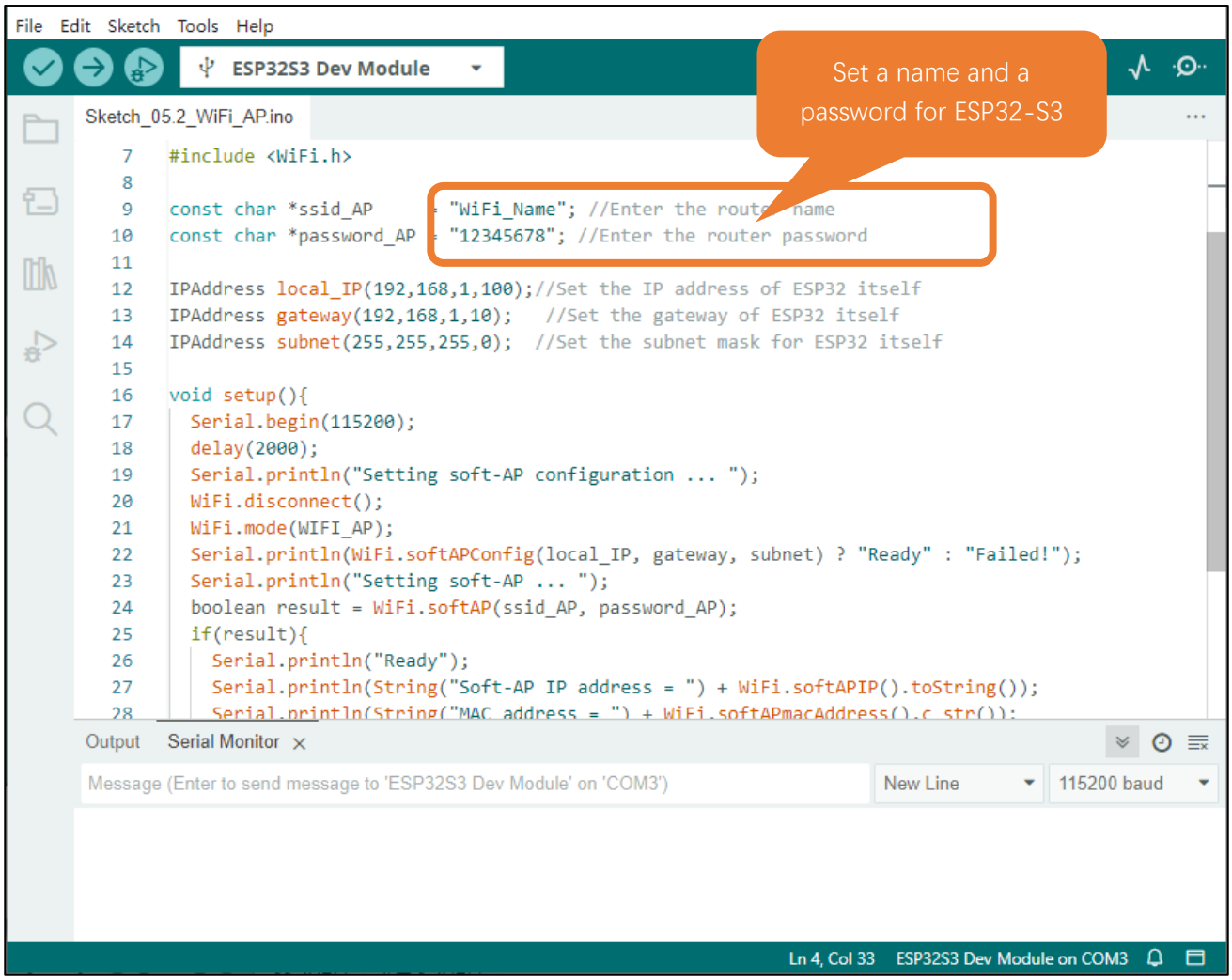



## Circuit

Connect Freenove ESP32-S3 WROOM to the computer using the Type C cable.

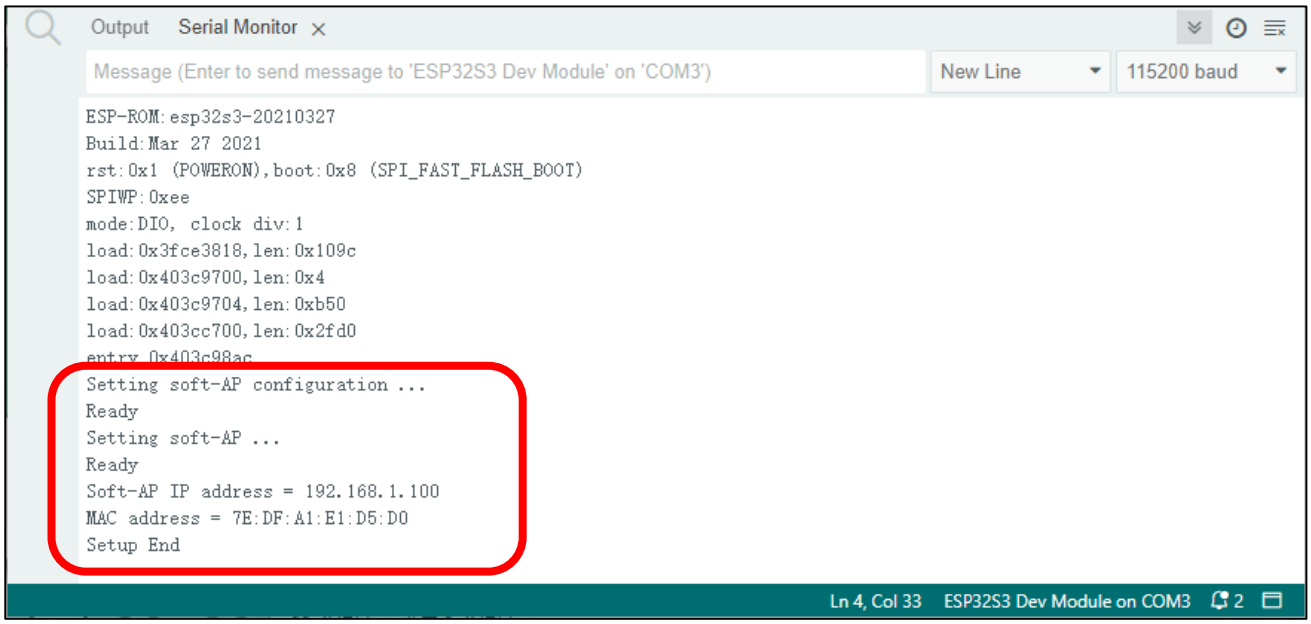


## Sketch

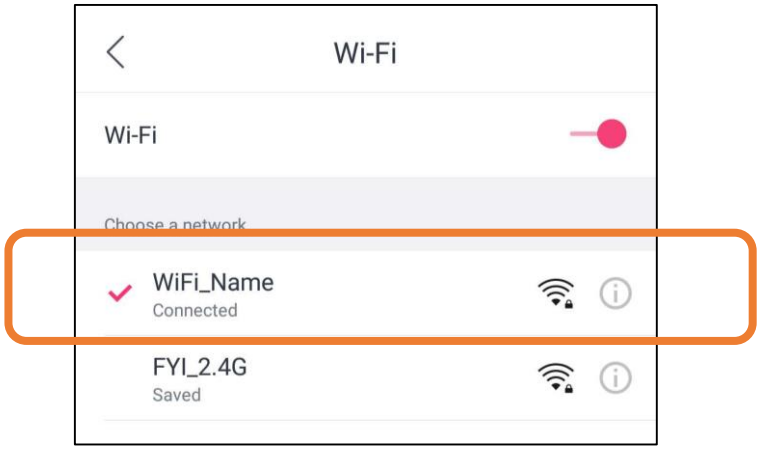



Before the Sketch runs, you can make any changes to the AP name and password for ESP32-S3 WROOM in the box as shown in the illustration above. Of course, you can leave it alone by default.

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 WROOM , 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_05.2_WiFi_AP

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 WROOM itself
IPAddress gateway(192,168,1,10);   //Set the gateway of ESP32-S3 WROOM itself
IPAddress subnet(255,255,255,0);   //Set the subnet mask for ESP32-S3 WROOM 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 WROOM .

```
#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 WROOM in AP mode.

```
WiFi.mode(WIFI_AP);
```

Configure IP address, gateway and subnet mask for ESP32-S3 WROOM .

```
WiFi.softAPConfig(local_IP, gateway, subnet)
```

Turn on an AP in ESP32-S3 WROOM , 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 WROOM . 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

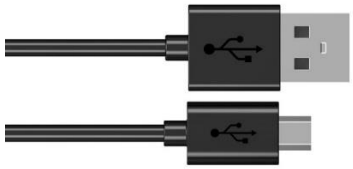
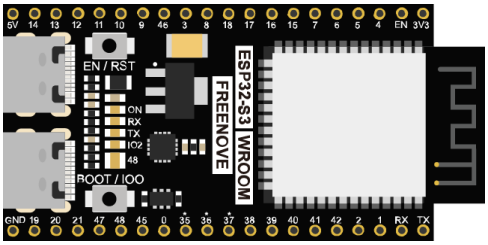
**subnet**: subnet mask

**softAP()**: obtian IP address in AP mode

**softAPdisconnect ()**: disconnect AP mode.

# Project 5.3 AP+Station mode

## Component List

| | |
|---|---|
| Type C Wire x1  | ESP32-S3 WROOM x1   |

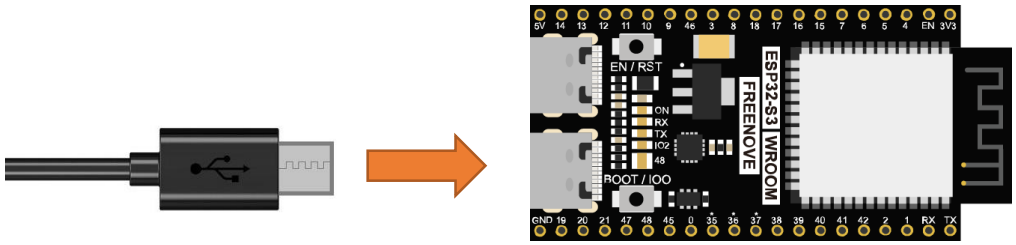
## Component knowledge

### AP+Station mode

In addition to AP mode and station mode, ESP32-S3 WROOM 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 WROOM '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 WROOM .

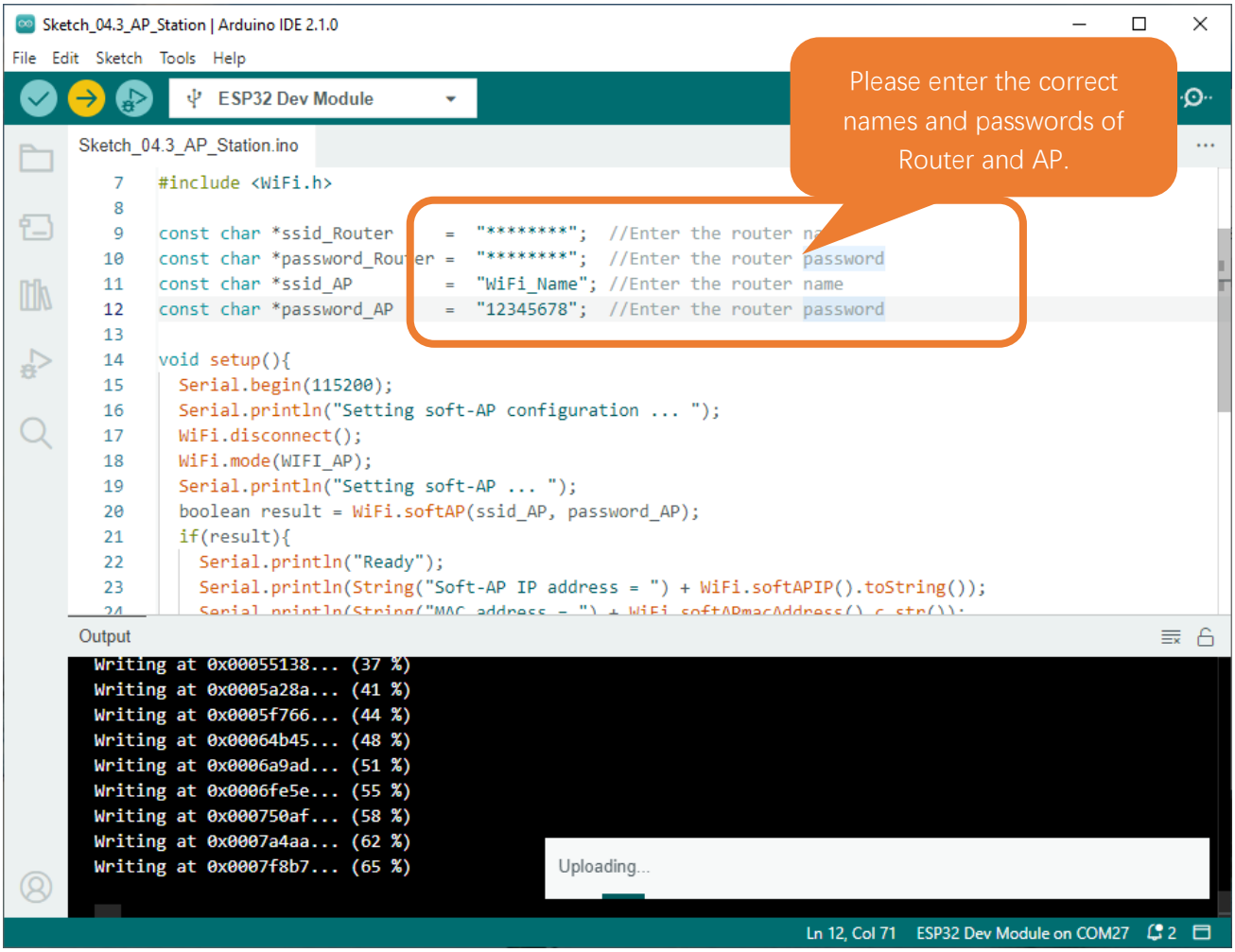
## Circuit

Connect Freenove ESP32-S3 WROOM to the computer using the Type C cable.




## Sketch

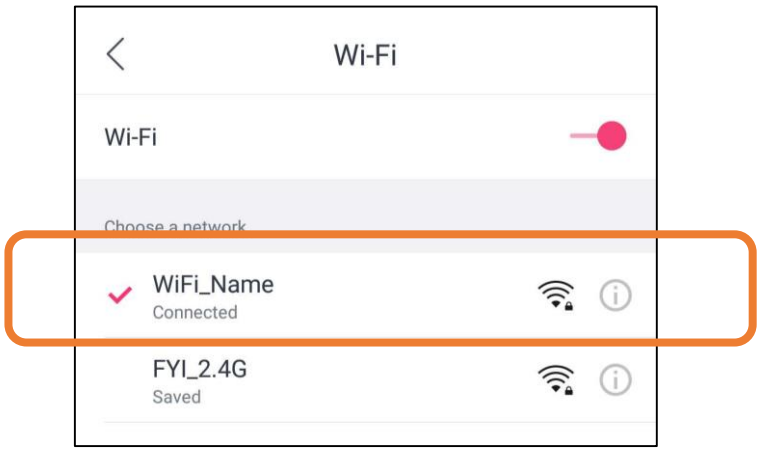
### Sketch_04.3_AP_Station_mode




It is analogous to Project 4.1 and Project 4.2. 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 WROOM .




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);
```

```
10    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  }
```

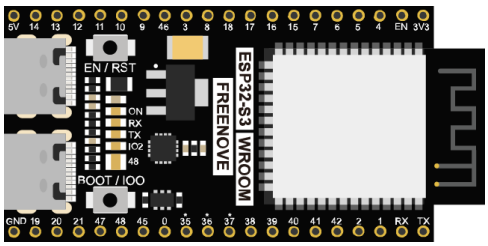
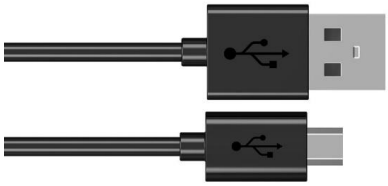
# Chapter 6 TCP/IP

In this chapter, we will introduce how ESP32-S3 WROOM 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 6.1 As Client

In this section, ESP32-S3 WROOM is used as Client to connect Server on the same LAN and communicate with it.

### Component List

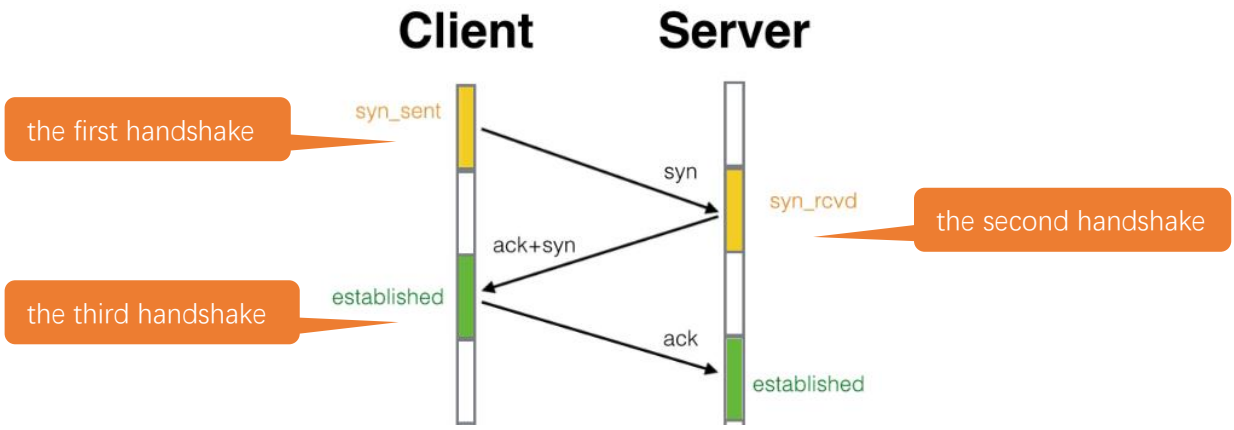
| | |
|---|---|
| ESP32-S3 WROOM x1  | Type C Wire x1  |

### Component knowledge

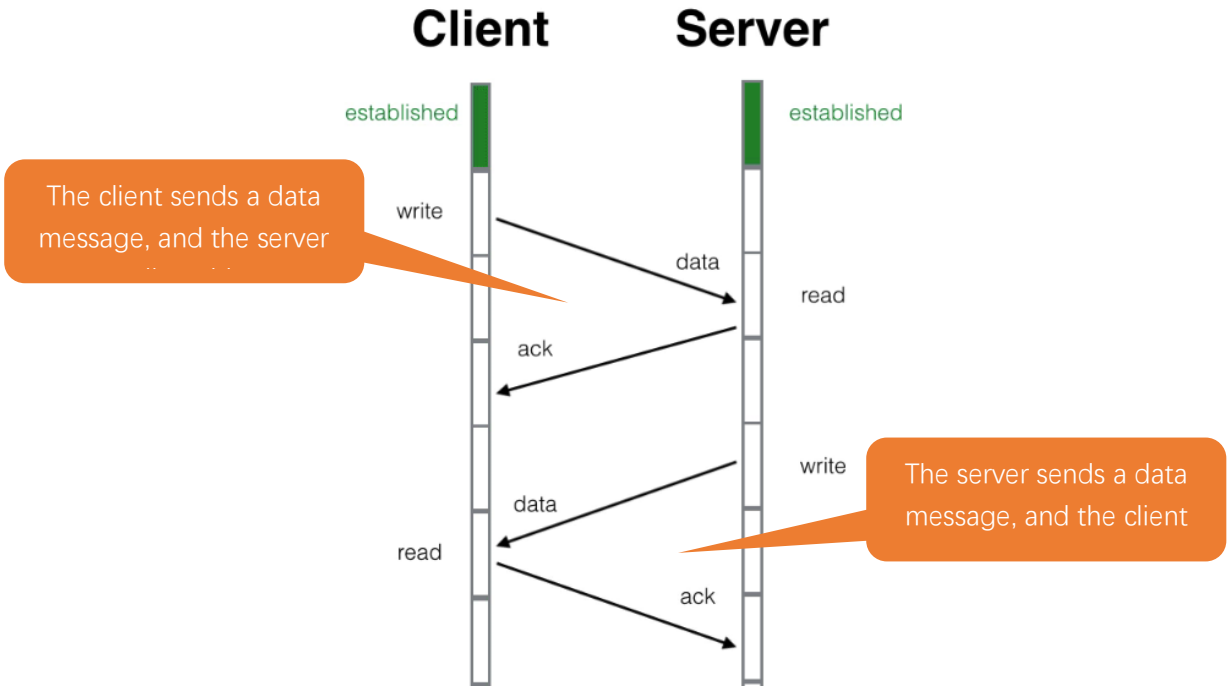
#### TCP connection

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.



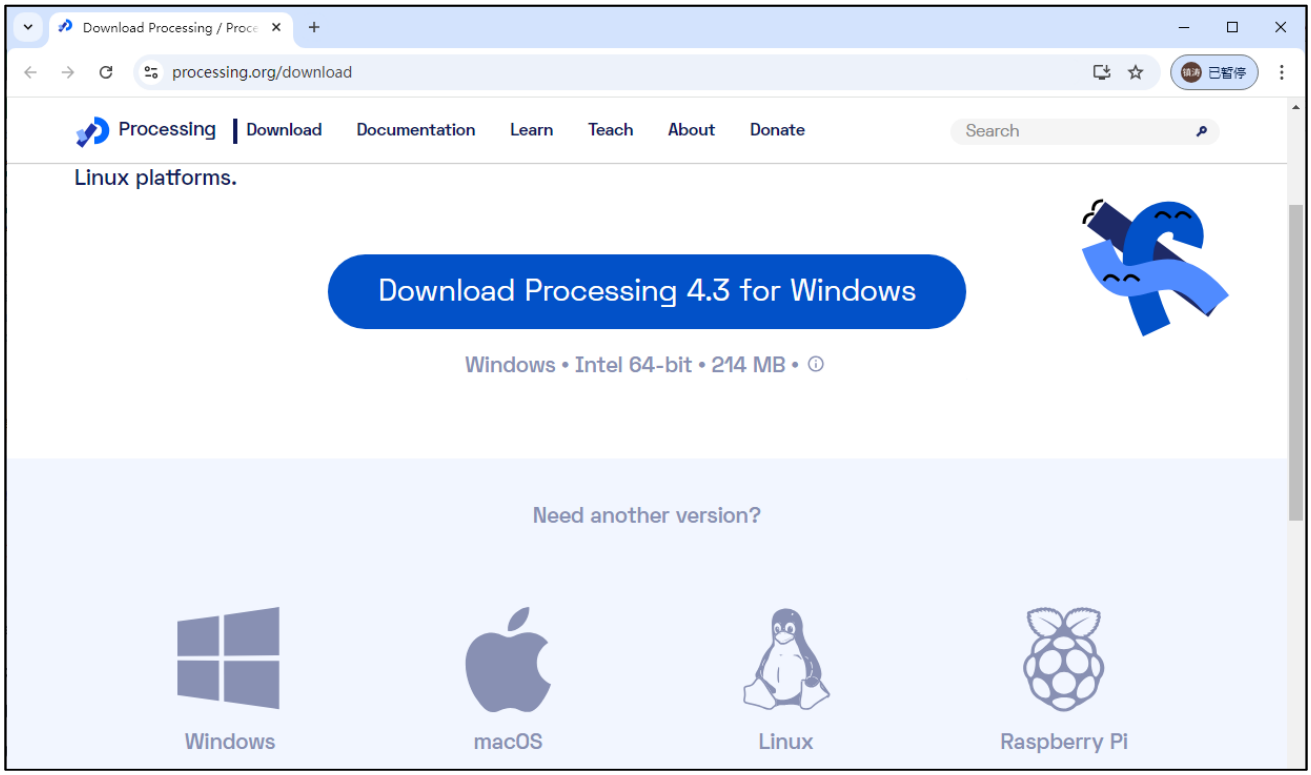

TCP is a connection-oriented, low-level transmission control protocol. After TCP establishes a connection, the client and server can send and receive messages to each other, and the connection will always exist as long as the client or server does not initiate disconnection. Each time one party sends a message, the other party will reply with an ack signal.



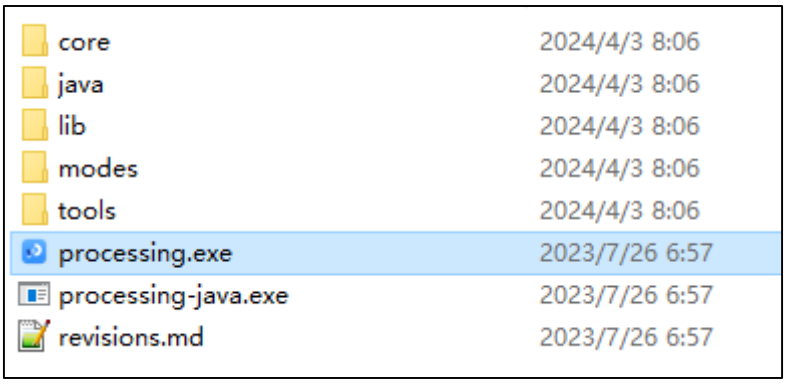
## Install Processing

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.

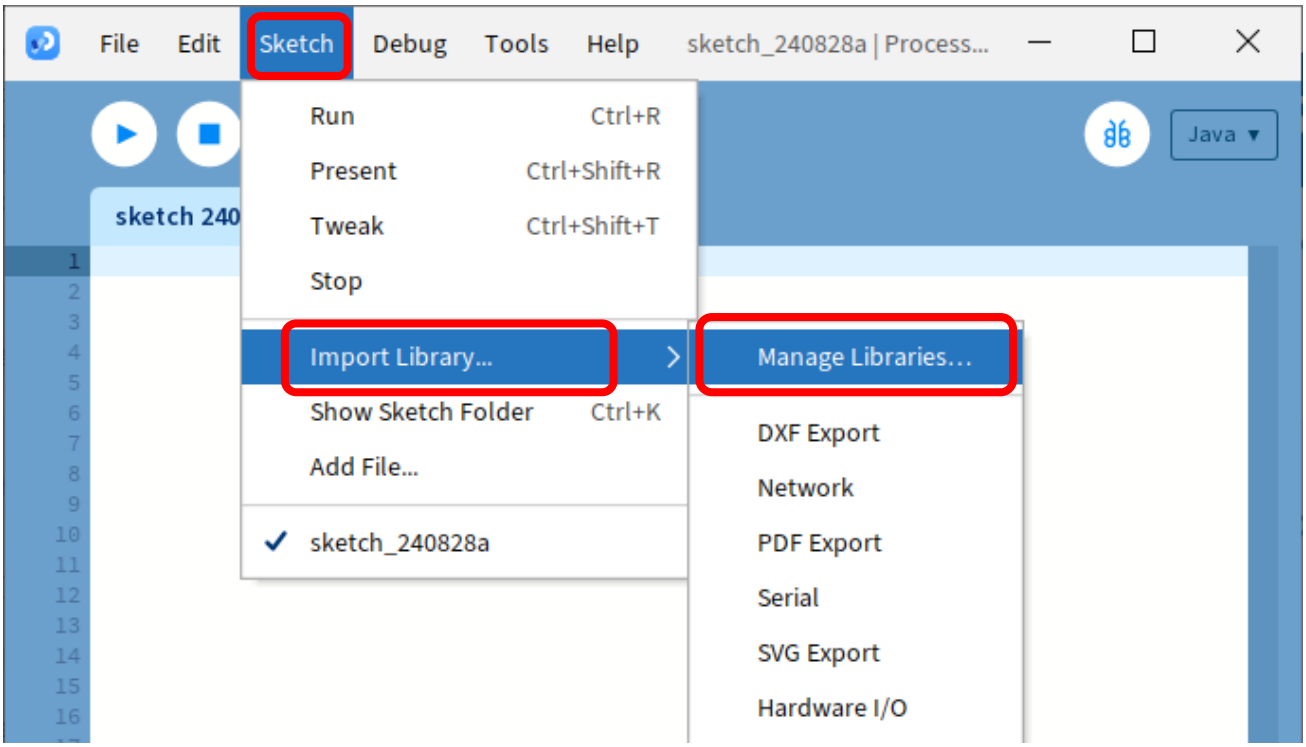


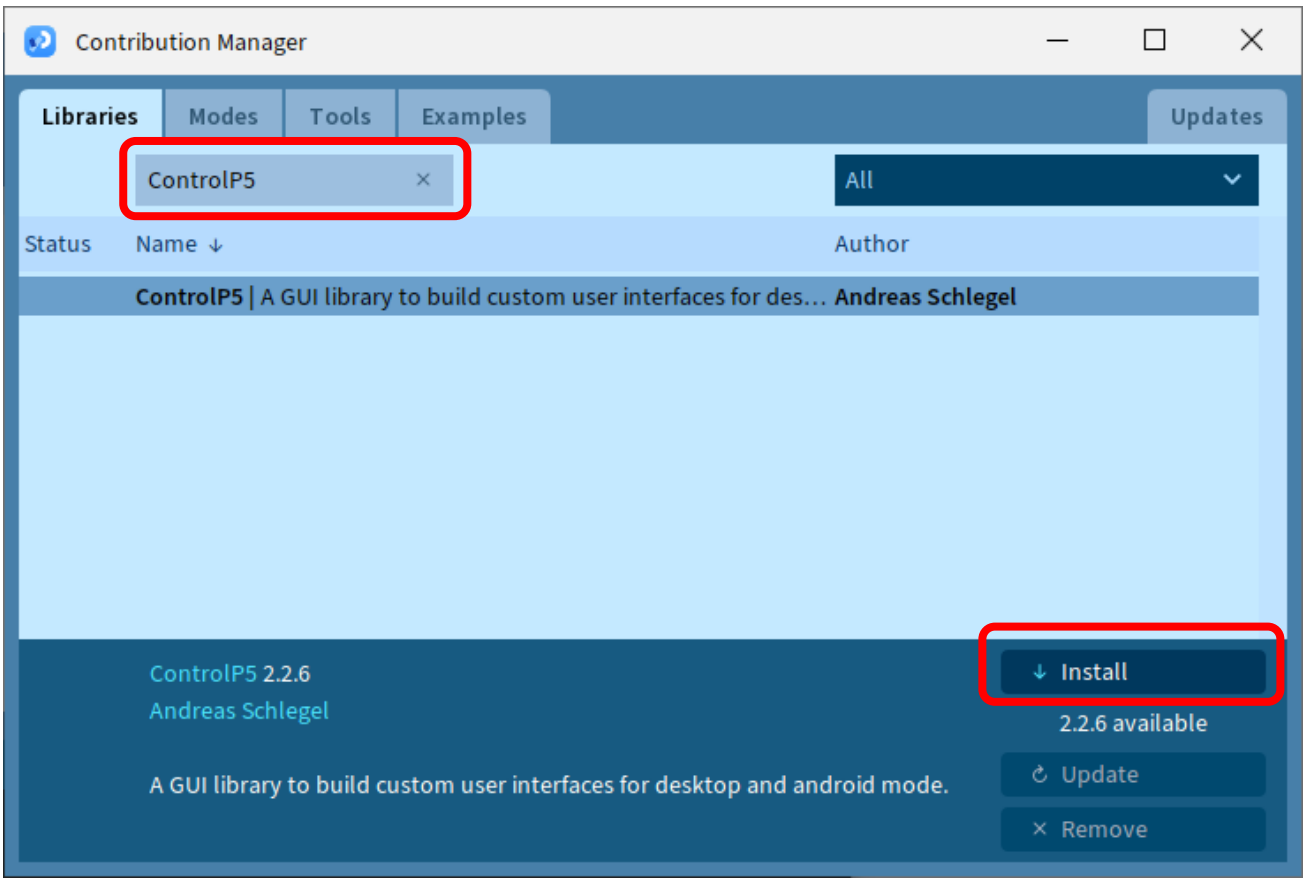

Unzip the downloaded file to your computer. Click "processing.exe" as the figure below to run this software.



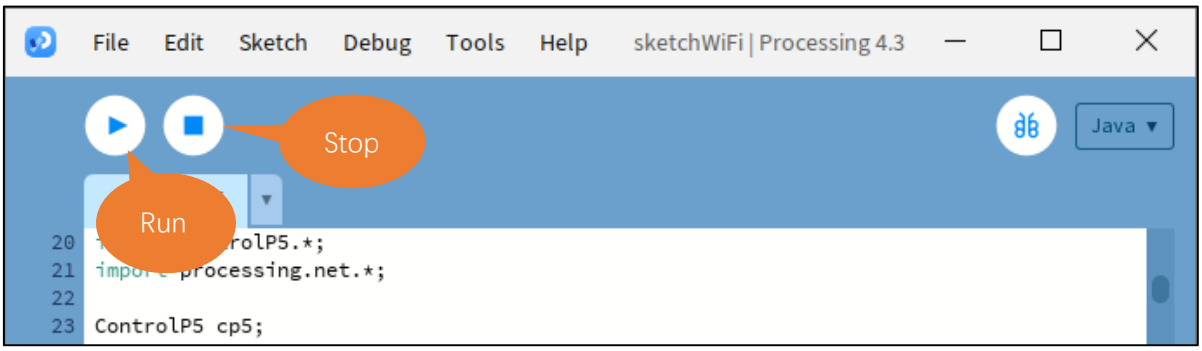
Use Server mode for communication

Install ControlP5.

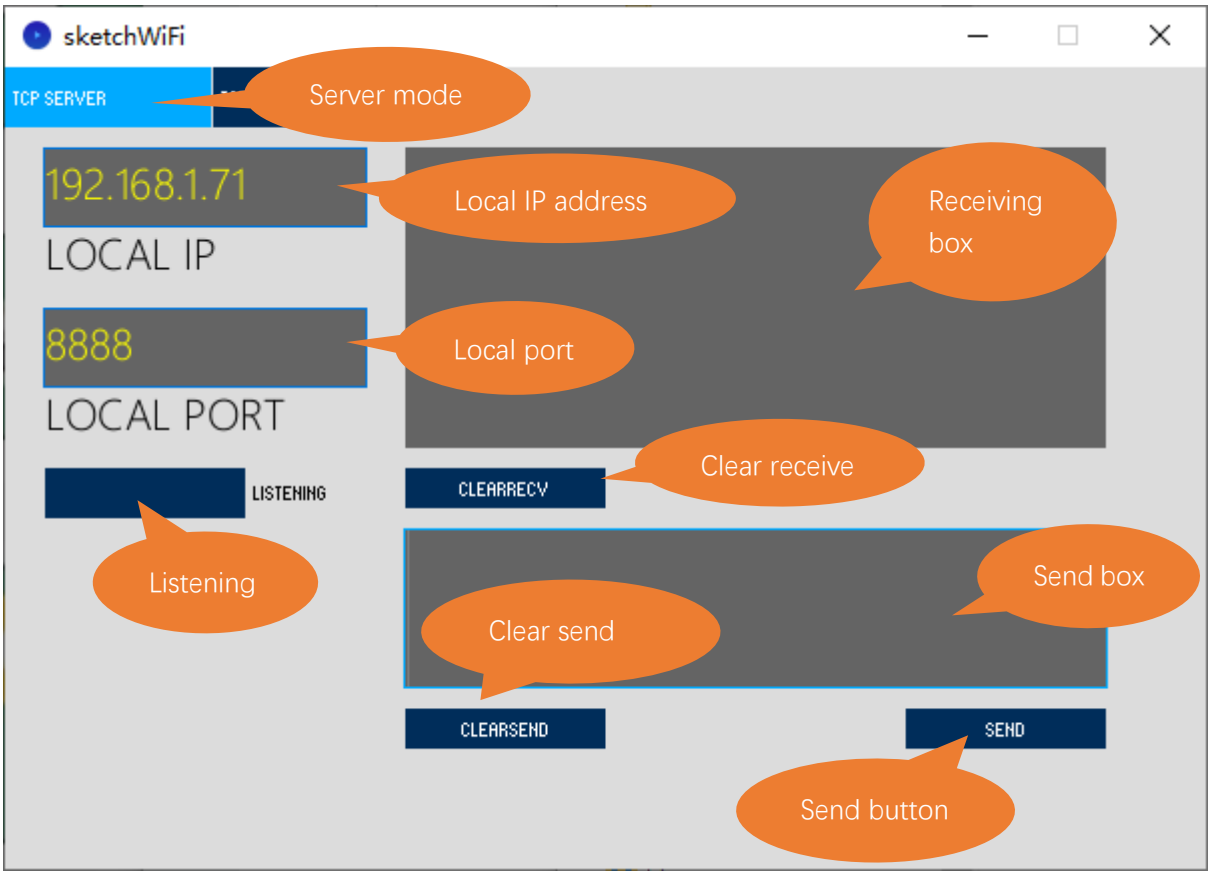


Open the "**Freenove_ESP32_S3_WROOM_Board_Lite\Sketches\Sketches\Sketch_06.1_WiFiClient\sketchWiFi\sketchWiFi.pde**", and click "Run".

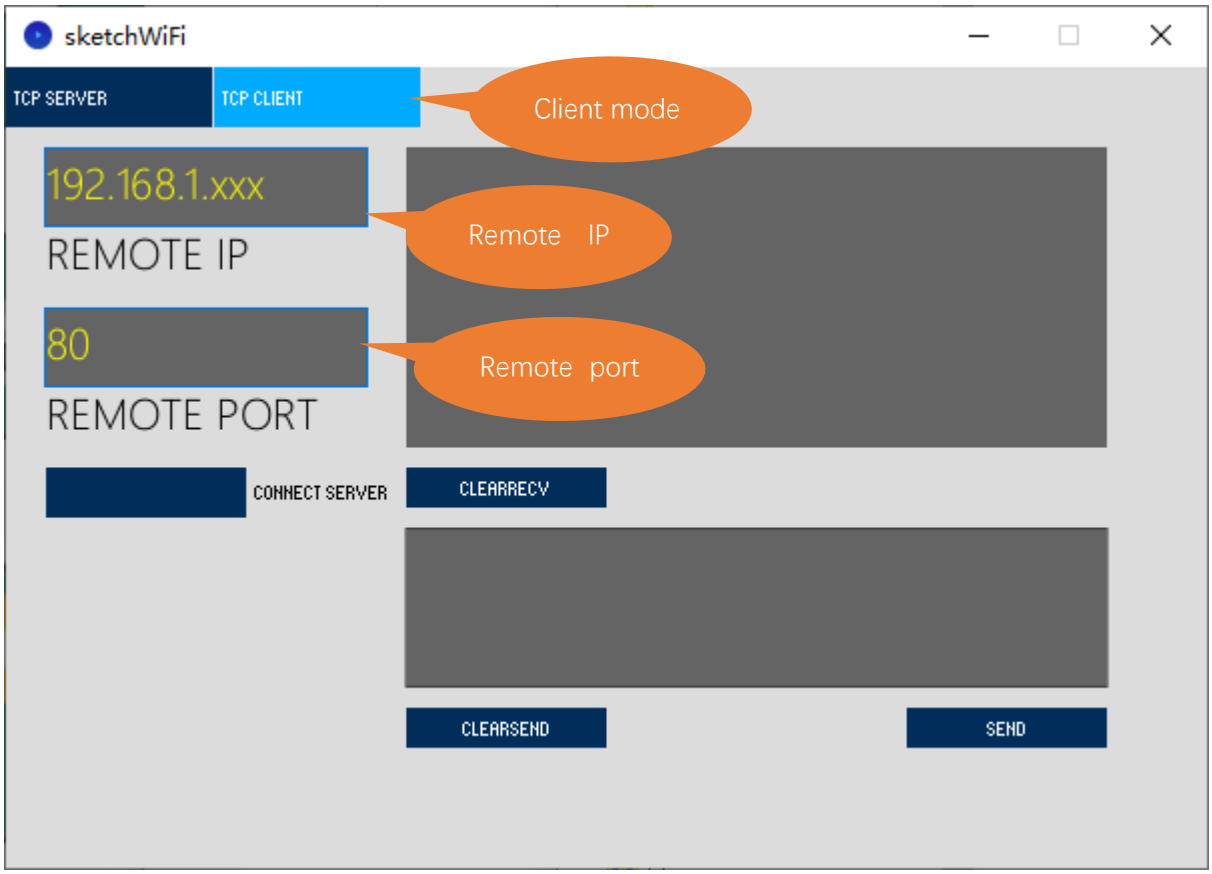



The new pop-up interface is as follows. If ESP32-S3 WROOM is used as client, select TCP SERVER mode for sketchWiFi.




When sketchWiFi selects TCP SERVER mode, ESP32-S3 WROOM Sketch needs to be changed according to sketchWiFi's displaying of LOCAL IP or LOCAL PORT.

If ESP32-S3 WROOM 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.

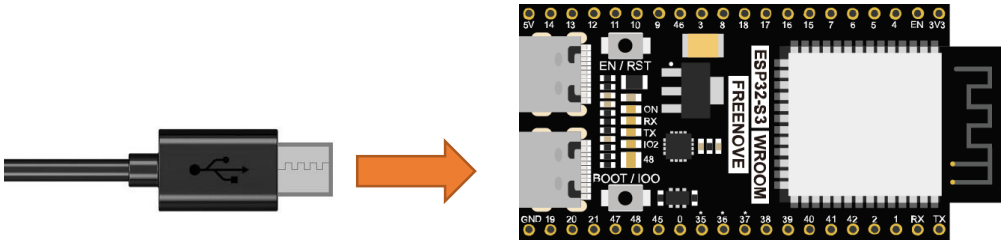
**clear receive**: clear out the content in the receiving text box

**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.

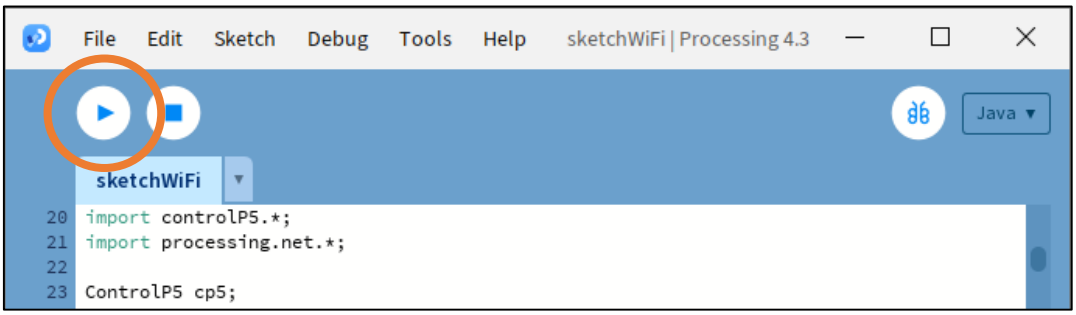
## Circuit

Connect Freenove ESP32-S3 WROOM to the computer using Type C cable.

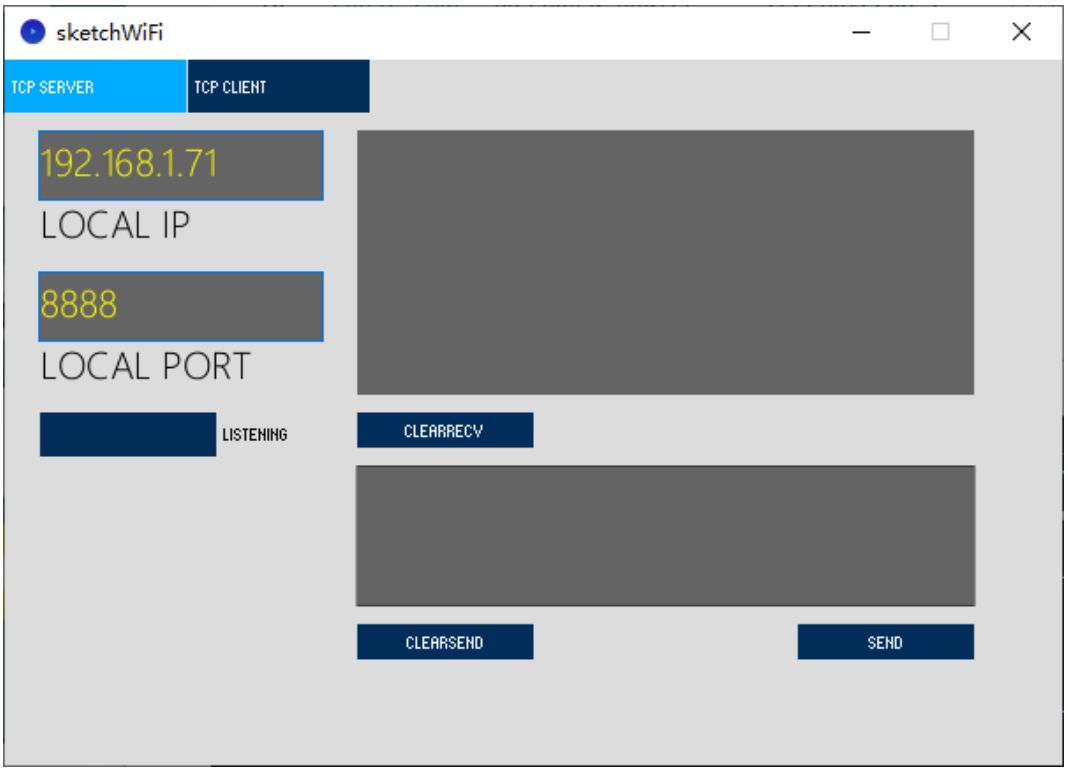



## Sketch

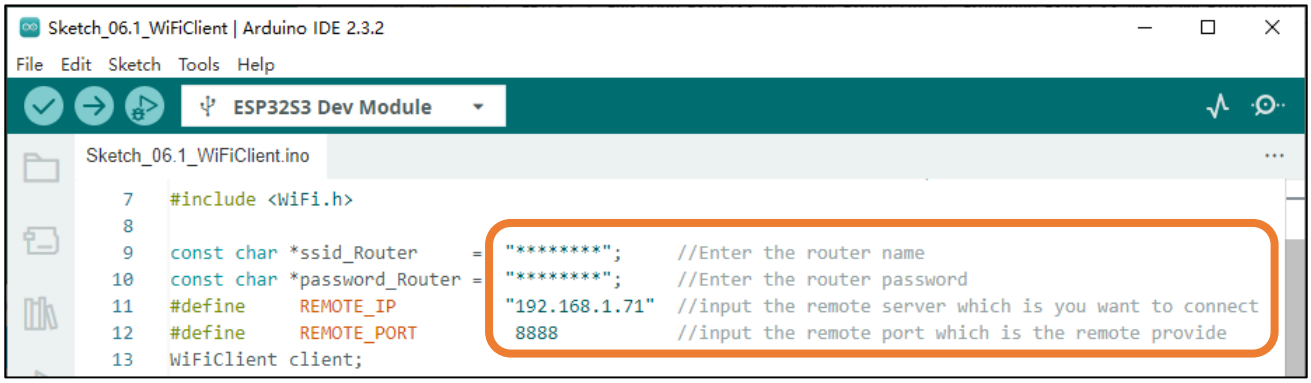
Before running the Sketch, please open "sketchWiFi.pde." first, and click "Run".




The newly pop up window will use the computer's IP address by default and open a data monitor port.

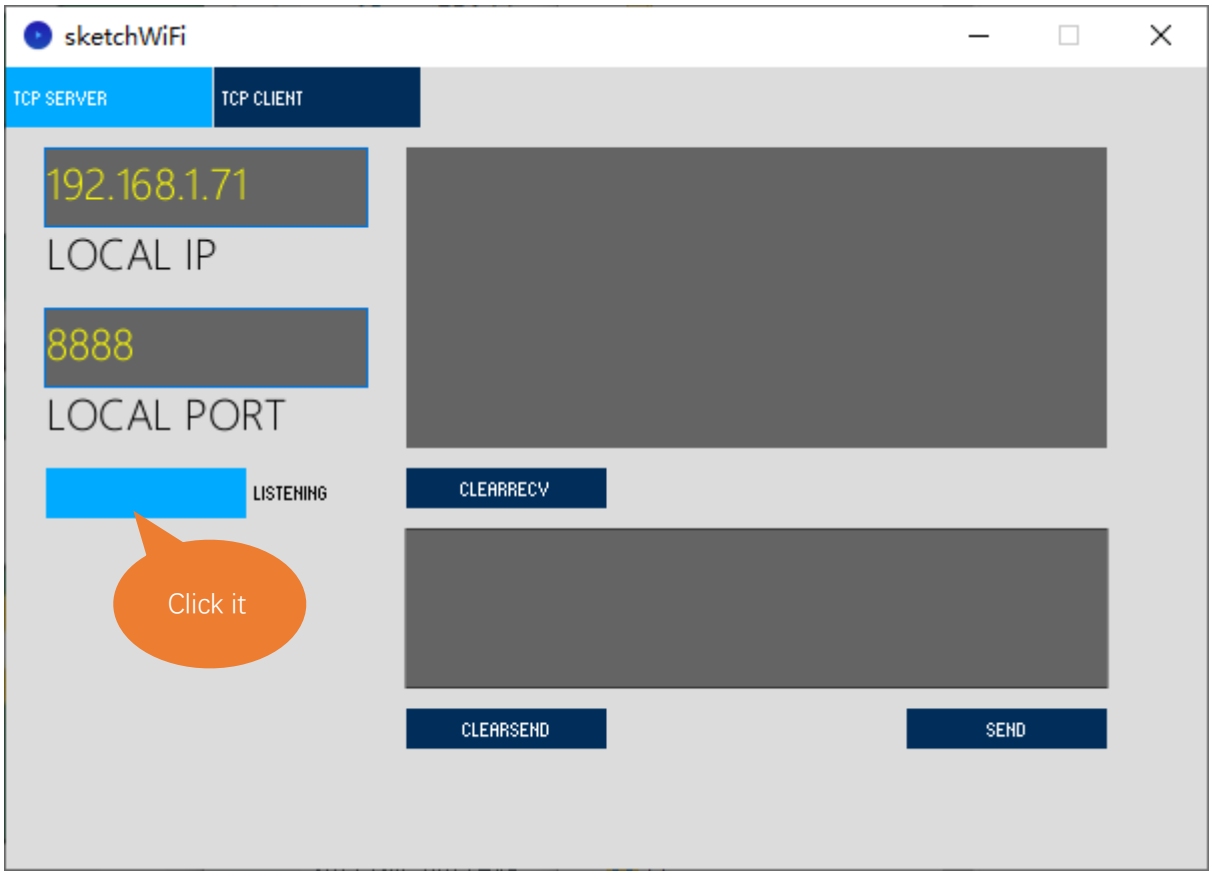


Next, open Sketch_06.1_WiFiClient.ino. Before running it, please change the following information based on "LOCAL IP" and "LOCAL PORT" in the figure above.

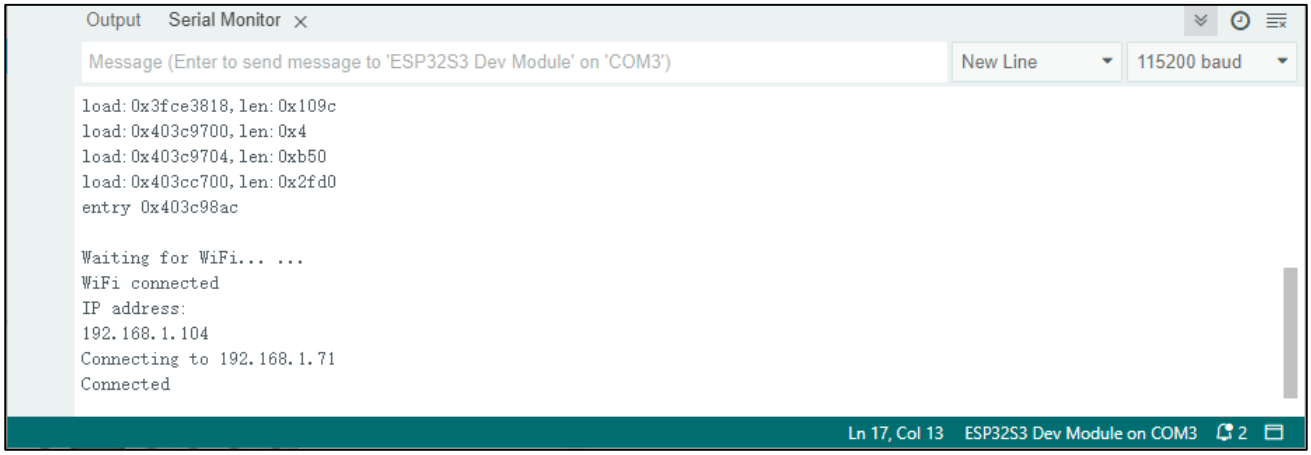



REMOTE_IP needs to be filled in according to the interface of sketchWiFi.pde. Taking this tutorial as an example, its REMOTE_IP is "192.168.1.71". Generally, by default, the ports do not need to change its value.

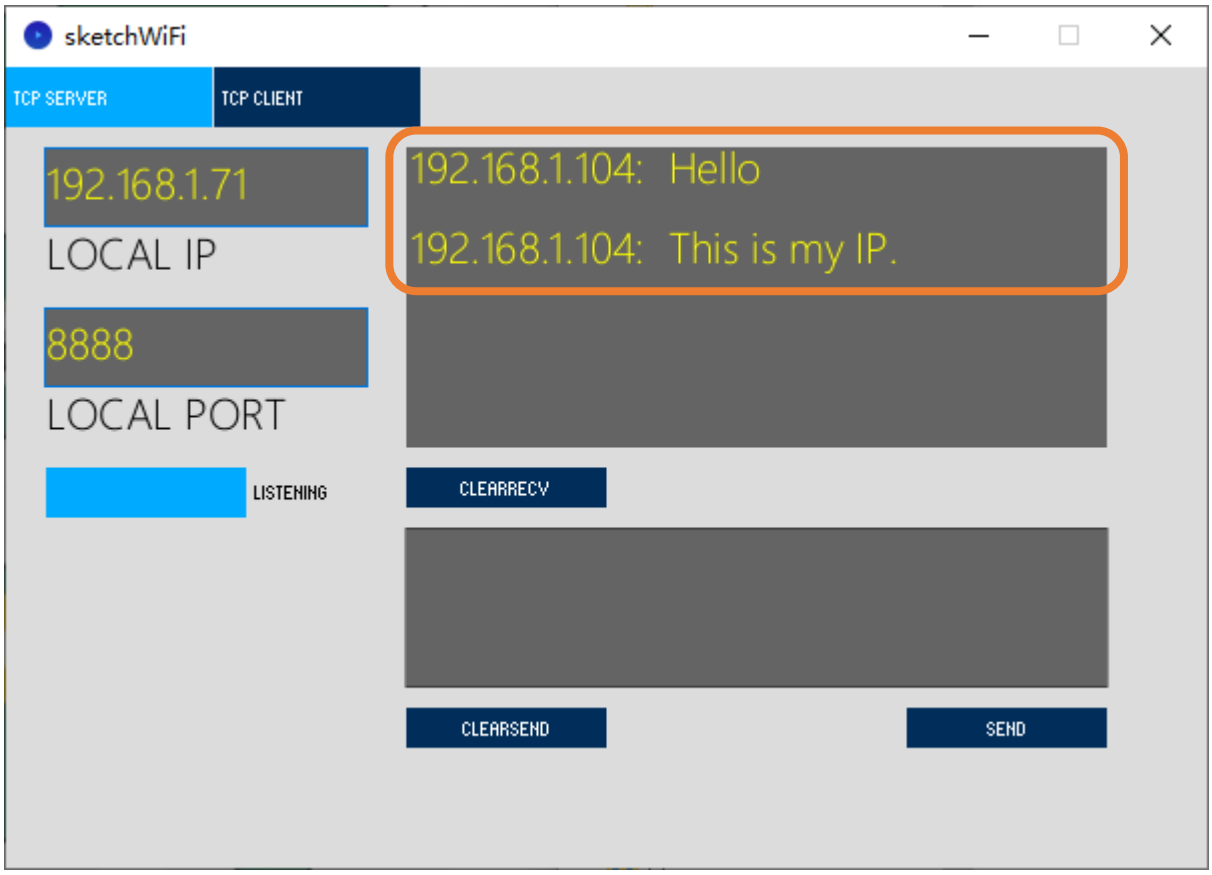
Click LISTENING, turn on TCP SERVER's data listening function and wait for ESP32-S3 WROOM to connect.




Compile and upload code to ESP32-S3 WROOM, open the serial monitor and set the baud rate to 115200. ESP32-S3 WROOM 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 WROOM can send messages to server.




ESP32-S3 WROOM connects with TCP SERVER, and TCP SERVER receives messages from ESP32, as shown in the figure below.



Sketch_06.1_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 WROOM 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 WROOM .

```
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.

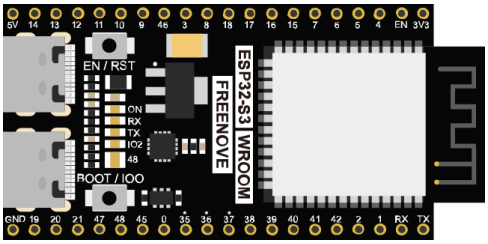
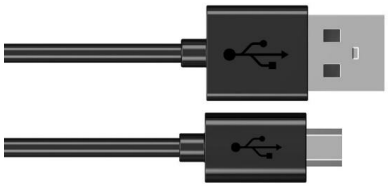
**read()**: read one byte of data in receive buffer

**readString()**: read string in receive buffer

# Project 6.2 As Server

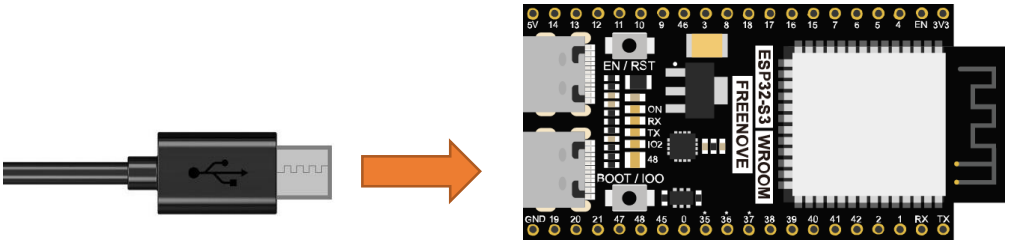
In this section, ESP32-S3 WROOM is used as a server to wait for the connection and communication of client on the same LAN.

## Component List

| | |
|---|---|
| ESP32-S3 WROOM x1  | Type C Wire x1  |

## Circuit

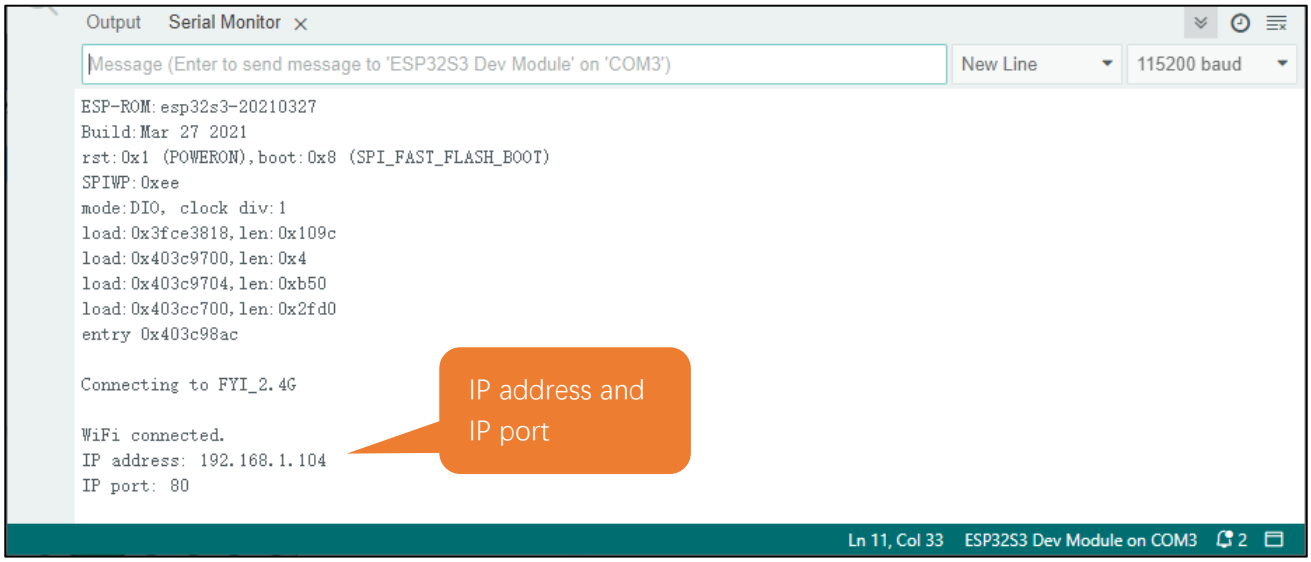
Connect Freenove ESP32-S3 WROOM to the computer using a Type C cable.



## Sketch

Before running Sketch, please modify the contents of the box below first.
Sketch_06.2_As_Server

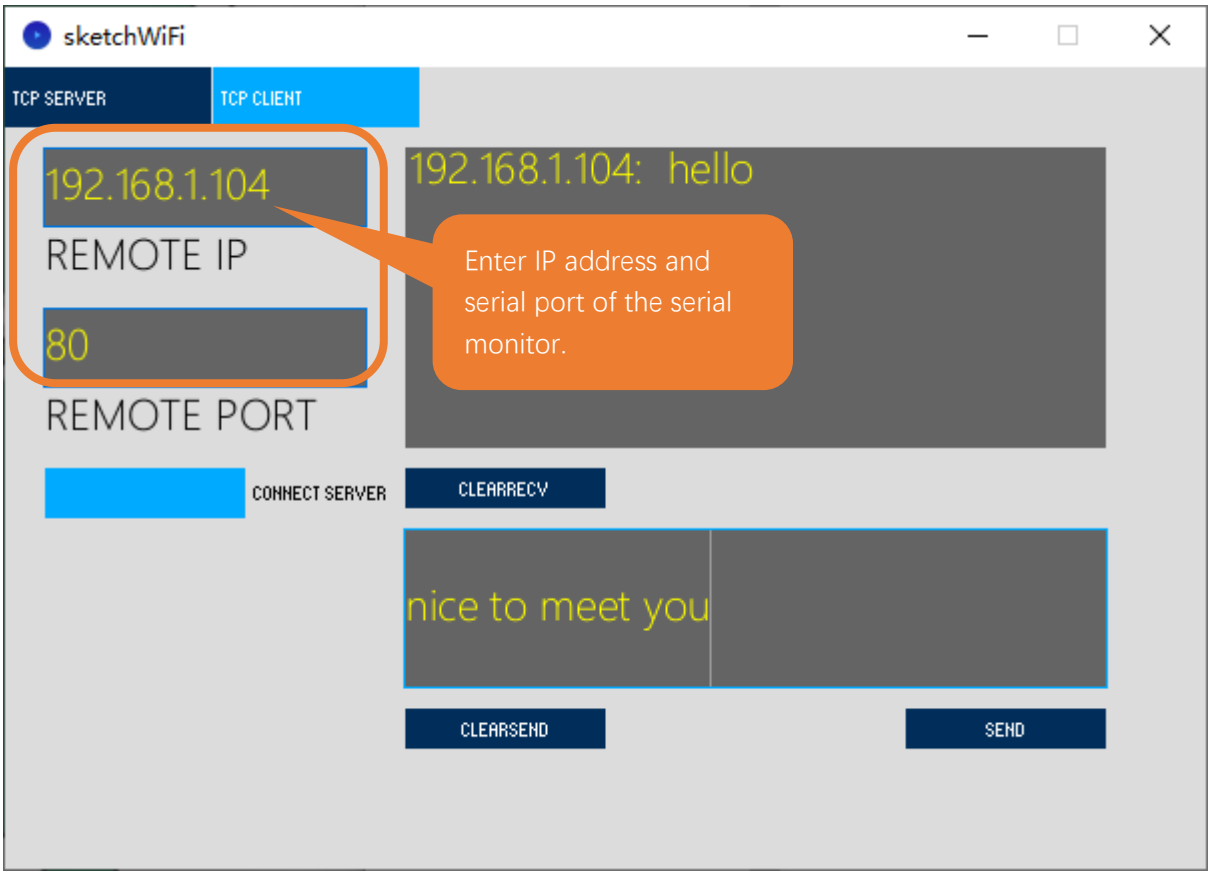

Compile and upload code to ESP32-S3 WROOM board, open the serial monitor and set the baud rate to 115200. Turn on server mode for ESP32-S3 WROOM , waiting for the connection of other devices on the same LAN. Once a device connects to server successfully, they can send messages to each other.
If the ESP32-S3 WROOM fails to connect to router, press the reset button as shown below and wait for ESP32-S3 WROOM to run again.



Processing:

Open the "**Freenove_ESP32_S3_WROOM_Board_Lite\Sketches\Sketches\Sketch_06.2_WiFiServer\sketchWiFi\sketchWiFi.pde**".

Based on the messages printed by the serial monitor, enter correct IP address and IP 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) {
```

```
17          delay(500);
18          Serial.print(".");
19      }
20      Serial.println("");
21      Serial.println("WiFi connected.");
22      Serial.print("IP address: ");
23      Serial.println(WiFi.localIP());
24      Serial.printf("IP port: %d\n",port);
25      server.begin(port);
26      WiFi.setAutoConnect(true);
27      WiFi.setAutoReconnect(true);
28  }
29
30  void loop(){
31   WiFiClient client = server.available();            // listen for incoming clients
32    if (client) {                                    // if you get a client
33      Serial.println("Client connected.");
34      while (client.connected()) {                   // loop while the client's connected
35        if (client.available()) {                    // if there's bytes to read from the
    client
36          Serial.println(client.readStringUntil('\n')); // print it out the serial monitor
37          while(client.read()>0);                    // clear the wifi receive area cache
38        }
39        if(Serial.available()){                      // if there's bytes to read from the
    serial monitor
40          client.print(Serial.readStringUntil('\n')); // print it out the client.
41          while(Serial.read()>0);                    // clear the wifi receive area cache
42        }
43      }
44      client.stop();                                 // stop the client connecting.
45      Serial.println("Client Disconnected.");
46    }
47  }
```

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 WROOM .

```
22      Serial.print("IP address: ");
23      Serial.println(WiFi.localIP());              //print out IP address of ESP32-S3 WROOM
24      Serial.printf("IP port: %d\n",port);         //Print out ESP32-S3 WROOM 's port number
```

Turn on server mode of ESP32-S3 WROOM , start automatic connection and turn on automatic reconnection.

```
25      server.begin();                              //Turn ON ESP32-S3 WROOM as Server mode
26      WiFi.setAutoConnect(true);
27      WiFi.setAutoReconnect(true);
```

When ESP32-S3 WROOM receive messages from servers, it will print them out via serial port; Users can also send messages to servers from serial port.

```
35      if (client.available()) {                    // if there's bytes to read from the
        client
36        Serial.println(client.readStringUntil('\n')); // print it out the serial monitor
37        while(client.read()>0);                    // clear the wifi receive area cache
38      }
39      if(Serial.available()){                      // if there's bytes to read from the
        serial monitor
40        client.print(Serial.readStringUntil('\n')); // print it out the client.
41        while(Serial.read()>0);                    // clear the wifi receive area cache
42      }
```

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.

# 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 WROOM , you view our ultimate tutorial:
https://github.com/Freenove/Freenove_ESP32_S3_WROOM_Board_Lite/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.