

# 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](mailto: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.

**Any concerns?** ✉ [support@freenove.com](mailto:support@freenove.com)

## 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](mailto:sale@freenove.com)

## Copyright

All the files, materials and instructional guides provided are released under [Creative Commons Attribution-NonCommercial-ShareAlike 3.0 Unported License](https://creativecommons.org/licenses/by-nc-sa/3.0/). A copy of this license can be found in the folder containing the Tutorial and software files associated with this product.



This means you can use these resource in your own derived works, in part or completely but **NOT for the intent or purpose of commercial use.**

Freenove brand and logo are copyright of Freenove Creative Technology Co., Ltd. and cannot be used without written permission.



Any concerns? [✉ support@freenove.com](mailto:support@freenove.com)

## Contents

Important Information.....	1
Contents.....	1
Preface.....	1
ESP32-WROVER .....	2
CH340 (Importance).....	5
Programming Software .....	15
Environment Configuration .....	18
Notes for GPIO.....	22
Chapter 1 LED .....	25
Project 1.1 Blink .....	25
Chapter 2 Bluetooth.....	30
Project 2.1 Bluetooth Passthrough.....	30
Project 2.2 Bluetooth Low Energy Data Passthrough.....	36
Chapter 3 Read and Write the Sdcard.....	48
Project 3.1 SDMMC Test.....	48
Chapter 4 WiFi Working Modes.....	59
Project 4.1 Station mode .....	59
Project 4.2 AP mode .....	63
Project 3.3 AP+Station mode .....	68
Chapter 5 TCP/IP.....	72
Project 5.1 As Client .....	72
Project 5.2 As Server .....	84
Chapter 6 Camera Web Server.....	90
Project 6.1 Camera Web Server .....	90
Project 6.2 Video Web Server.....	99
What's next? .....	105
End of the Tutorial.....	105



# Preface

ESP32 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 Microcontroller (SCM) chip, or connected to the Internet and used as an Internet of Things device.

ESP32 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 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.

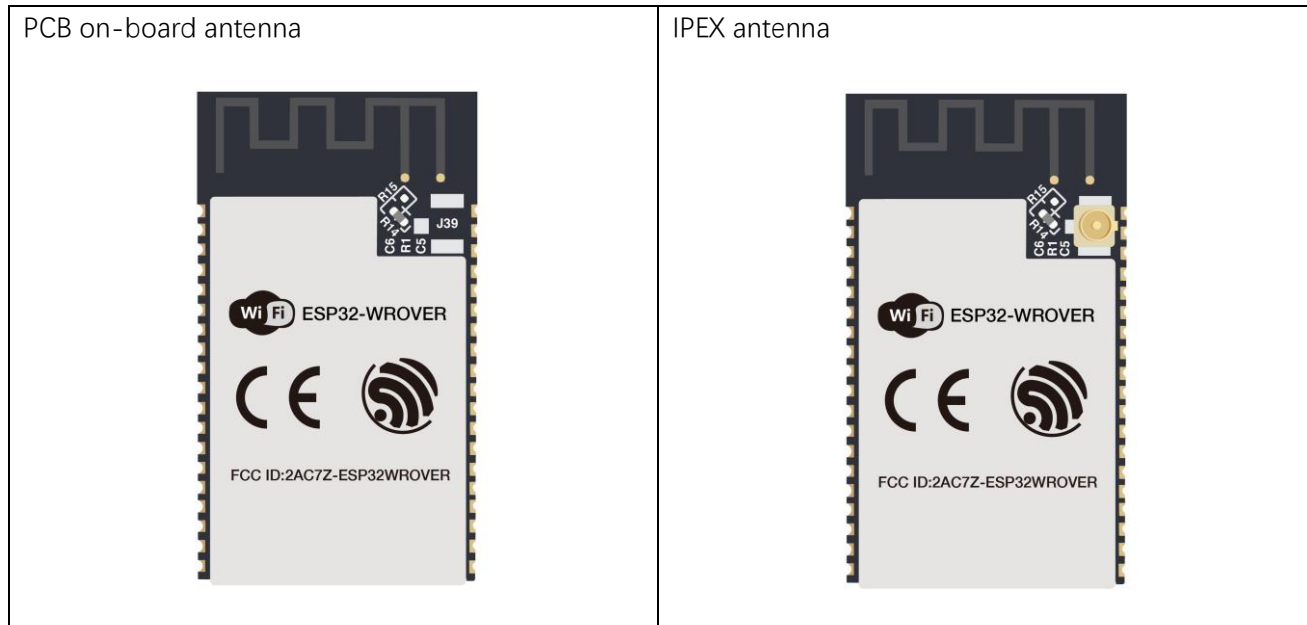
Generally, ESP32 projects consist of code and circuits. Don't worry even if you've never learned code and circuits, because we will gradually introduce the basic knowledge of C programming language and electronic circuits, from easy to difficult. Our products contain all the electronic components and 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 ESP32 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.

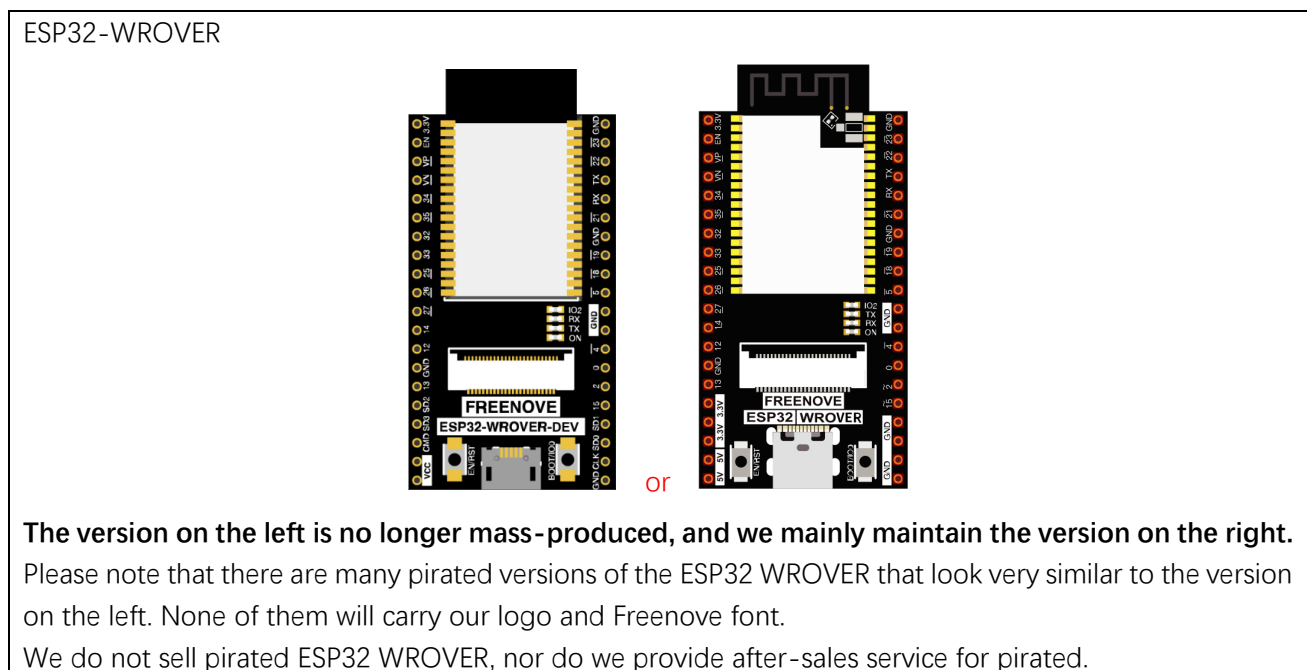
In addition, if you have any difficulties or questions with this tutorial or toolkit, feel free to ask for our quick and free technical support through [support@freenove.com](mailto:support@freenove.com)

## ESP32-WROVER

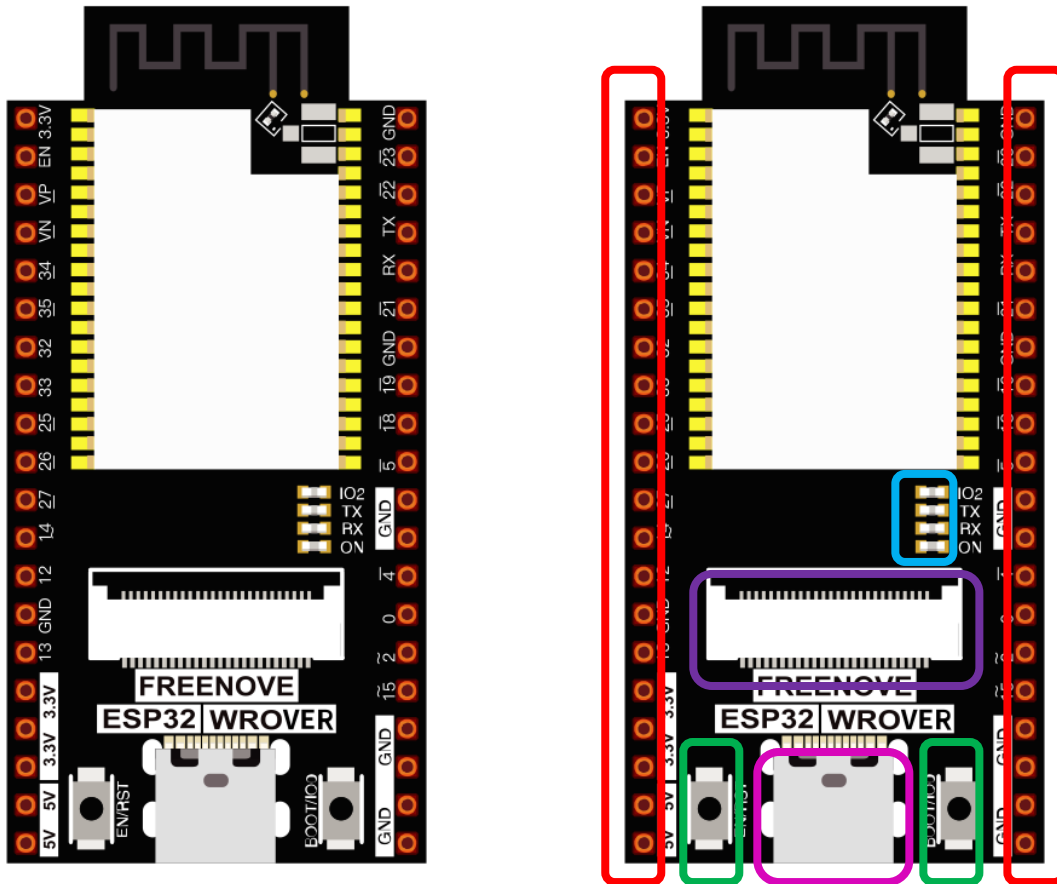
ESP32-WROVER has launched a total of two antenna packages, PCB on-board antenna and IPEX antenna respectively. The PCB on-board antenna is an integrated antenna in the chip module itself, so it is convenient to carry and design. The IPEX antenna is a metal antenna derived from the integrated antenna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








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



The hardware interfaces of ESP32-WROVER are distributed as follows:



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

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

Name	No.	Type	Function
GND	1	P	Ground
3V3	2	P	Power supply
EN	3	I	Module-enable signal. Active high.
SENSOR_VP	4	I	GPIO36, ADC1_CH0, RTC_GPIO0
SENSOR_VN	5	I	GPIO39, ADC1_CH3, RTC_GPIO3
IO34	6	I	GPIO34, ADC1_CH6, RTC_GPIO4
IO35	7	I	GPIO35, ADC1_CH7, RTC_GPIO5
IO32	8	I/O	GPIO32, XTAL_32K_P (32.768 kHz crystal oscillator input), ADC1_CH4, TOUCH9, RTC_GPIO9
IO33	9	I/O	GPIO33, XTAL_32K_N (32.768 kHz crystal oscillator output), ADC1_CH5, TOUCH8, RTC_GPIO8
IO25	10	I/O	GPIO25, DAC_1, ADC2_CH8, RTC_GPIO6, EMAC_RXD0
IO26	11	I/O	GPIO26, DAC_2, ADC2_CH9, RTC_GPIO7, EMAC_RXD1
IO27	12	I/O	GPIO27, ADC2_CH7, TOUCH7, RTC_GPIO17, EMAC_RX_DV
IO14	13	I/O	GPIO14, ADC2_CH6, TOUCH6, RTC_GPIO16, MTMS, HSPICLK, HS2_CLK, SD_CLK, EMAC_TXD2
IO12 <sup>1</sup>	14	I/O	GPIO12, ADC2_CH5, TOUCH5, RTC_GPIO15, MTDI, HSPIQ, HS2_DATA2, SD_DATA2, EMAC_TXD3
GND	15	P	Ground
IO13	16	I/O	GPIO13, ADC2_CH4, TOUCH4, RTC_GPIO14, MTCK, HSPID, HS2_DATA3, SD_DATA3, EMAC_RX_ER
SHD/SD2 <sup>2</sup>	17	I/O	GPIO9, SD_DATA2, SPIHD, HS1_DATA2, U1RXD
SWP/SD3 <sup>2</sup>	18	I/O	GPIO10, SD_DATA3, SPIWP, HS1_DATA3, U1TXD
SCS/CMD <sup>2</sup>	19	I/O	GPIO11, SD_CMD, SPICS0, HS1_CMD, U1RTS
SCK/CLK <sup>2</sup>	20	I/O	GPIO6, SD_CLK, SPICLK, HS1_CLK, U1CTS
SDO/SD0 <sup>2</sup>	21	I/O	GPIO7, SD_DATA0, SPIQ, HS1_DATA0, U2RTS
SDI/SD1 <sup>2</sup>	22	I/O	GPIO8, SD_DATA1, SPID, HS1_DATA1, U2CTS
IO15	23	I/O	GPIO15, ADC2_CH3, TOUCH3, MTDO, HSPICS0, RTC_GPIO13, HS2_CMD, SD_CMD, EMAC_RXD3
IO2	24	I/O	GPIO2, ADC2_CH2, TOUCH2, RTC_GPIO12, HSPWP, HS2_DATA0, SD_DATA0
IO0	25	I/O	GPIO0, ADC2_CH1, TOUCH1, RTC_GPIO11, CLK_OUT1, EMAC_TX_CLK
IO4	26	I/O	GPIO4, ADC2_CH0, TOUCH0, RTC_GPIO10, HSPHD, HS2_DATA1, SD_DATA1, EMAC_TX_ER
NC1	27	-	-
NC2	28	-	-
IO5	29	I/O	GPIO5, VSPICS0, HS1_DATA6, EMAC_RX_CLK
IO18	30	I/O	GPIO18, VSPICLK, HS1_DATA7
IO19	31	I/O	GPIO19, VSPIQ, U0CTS, EMAC_TXD0
NC	32	-	-
IO21	33	I/O	GPIO21, VSPHD, EMAC_TX_EN
RXD0	34	I/O	GPIO3, U0RXD, CLK_OUT2
TXD0	35	I/O	GPIO1, U0TXD, CLK_OUT3, EMAC_RXD2
IO22	36	I/O	GPIO22, VSPWP, U0RTS, EMAC_TXD1
IO23	37	I/O	GPIO23, VSPID, HS1_STROBE
GND	38	P	Ground

**Notice:**

1. GPIO12 is internally pulled high in the module and is not recommended for use as a touch pin.
2. Pins SCK/CLK, SDO/SD0, SDI/SD1, SHD/SD2, SWP/SD3 and SCS/CMD, namely, GPIO6 to GPIO11 are connected to the SPI flash integrated on the module and are not recommended for other uses.

For more information, please visit: [https://www.espressif.com/sites/default/files/documentation/esp32-wrover\\_datasheet\\_en.pdf](https://www.espressif.com/sites/default/files/documentation/esp32-wrover_datasheet_en.pdf)

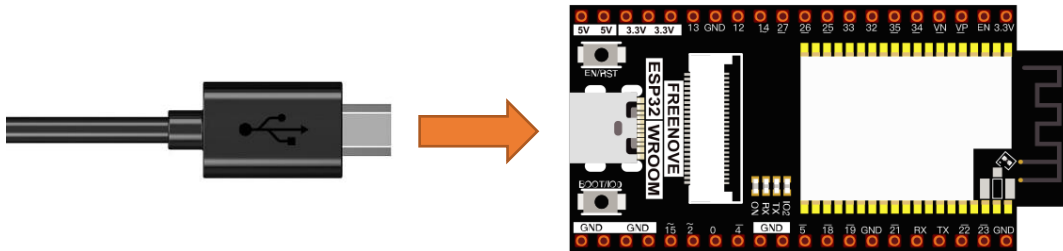
## CH340 (Importance)

ESP32 uses CH340 to download codes. So before using it, we need to install CH340 driver in our computers.

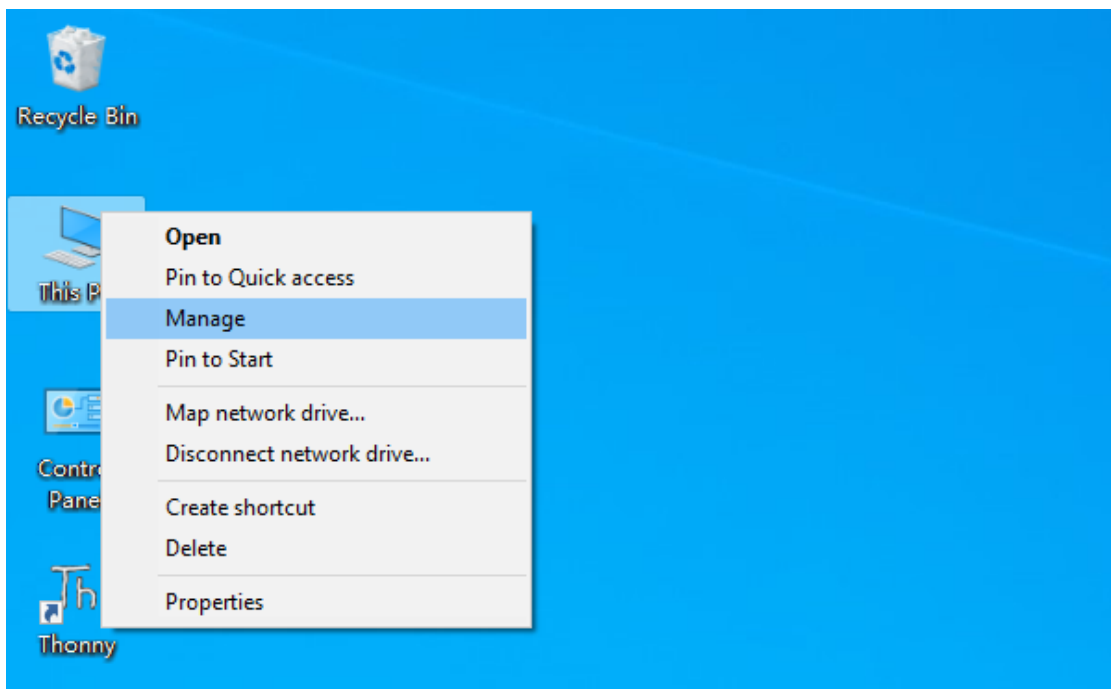
## Windows

Check whether CH340 has been installed

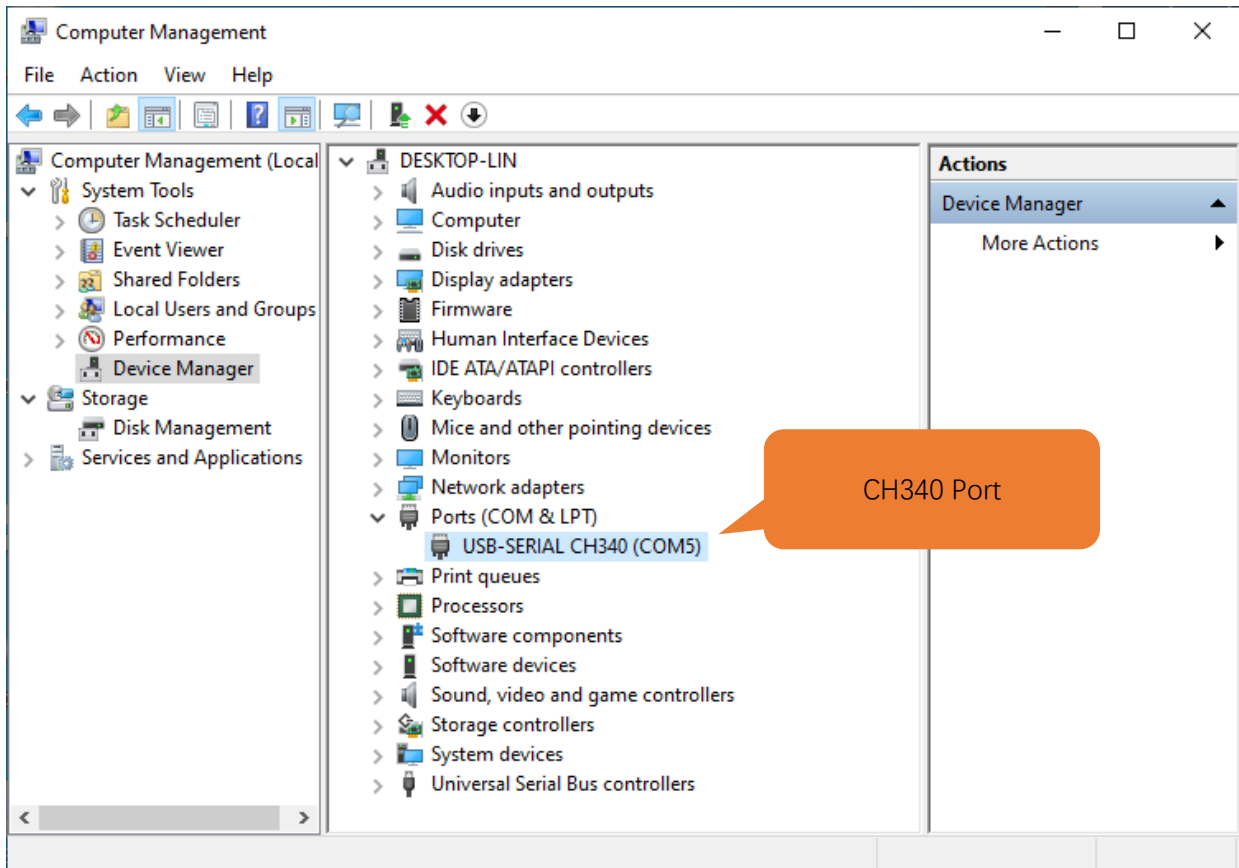
1. Connect your computer and ESP32 with a USB cable.



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



3. Click “Device Manager”. If your computer has installed CH340, you can see “USB-SERIAL CH340 (COMx)”. And you can click [here](#) to move to the next step.



## Installing CH340

1. First, download CH340 driver, click <http://www.wch-ic.com/search?q=CH340&t=downloads> to download the appropriate one based on your operating system.

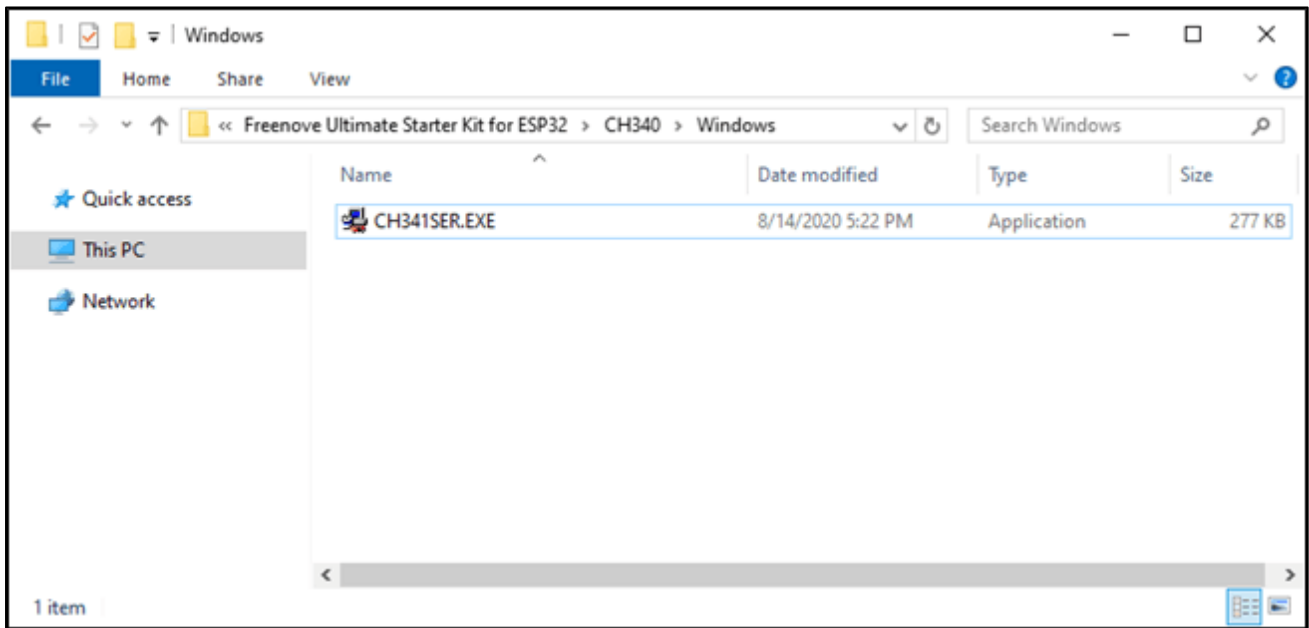
file category	file content	version	upload time
Driver&Tools			
CH341SER.EXE	CH340/CH341 USB to serial port Windows driver, supports 32/64-bit Windows 10/8.1/8/7/VISTA/XP, Server 2016/2012/2008/2003, 2000/ME/98	3.5	2019-03-18
CH341SER.ZIP	CH340/CH341 USB to serial port Windows driver, includes DLL dynamic library and non-standard baud rate settings and other instructions. Supports 32/64-bit Windows 10/8.1/8/7/VISTA/XP, Server 2016/2012/2008/2003, 2000/ME/98	3.5	2019-03-05
CH341SER_ANDROID...	CH340/CH341 USB to serial port Android free drive application library, for Android OS 3.1 and above version which supports USB Host mode already, no need to load Android kernel driver, no root privileges. Contains apk, lib library, file (Host Driver), App Demo Example (USB to UART Demo)	1.6	2019-04-19
CH341SER_LINUX...	CH340/CH341 USB to serial port LINUX driver	1.5	2018-03-18
CH341SER_MAC.ZIP	CH340/CH341 USB to serial port MAC OS driver	1.5	2018-07-05
Others			
PRODUCT_GUIDE.P...	Electronic selection of product selection manual, please refer to related product technical manual for more technical information.	1.4	2018-12-29
InstallNoteOn64...	Instructions for the driver after 18 years of August cannot be installed under some 64-bit WIN7 (English)	1.0	2019-01-10

If you would not like to download the installation package, you can open

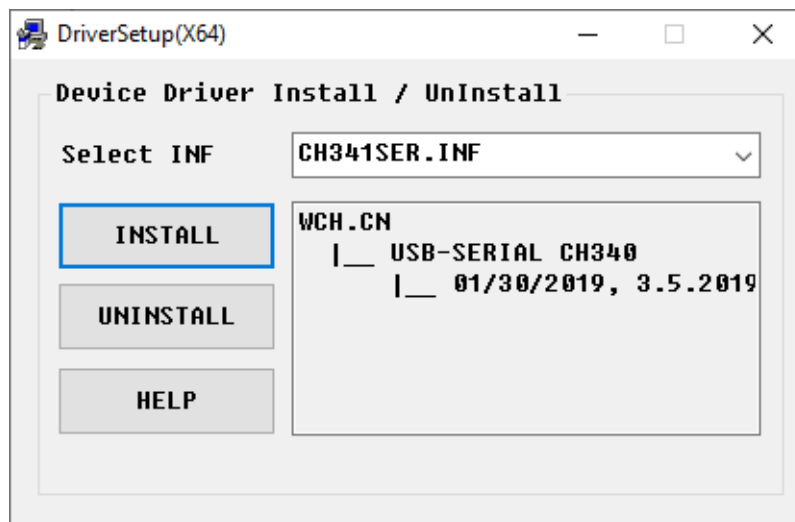
“Freenove\_ESP32\_WROVER\_Board/CH340”, we have prepared the installation package.

Name	Date modified	Type	Size
Linux	8/14/2020 5:24 PM	File folder	
MAC	8/14/2020 5:23 PM	File folder	
Windows	8/14/2020 5:23 PM	File folder	

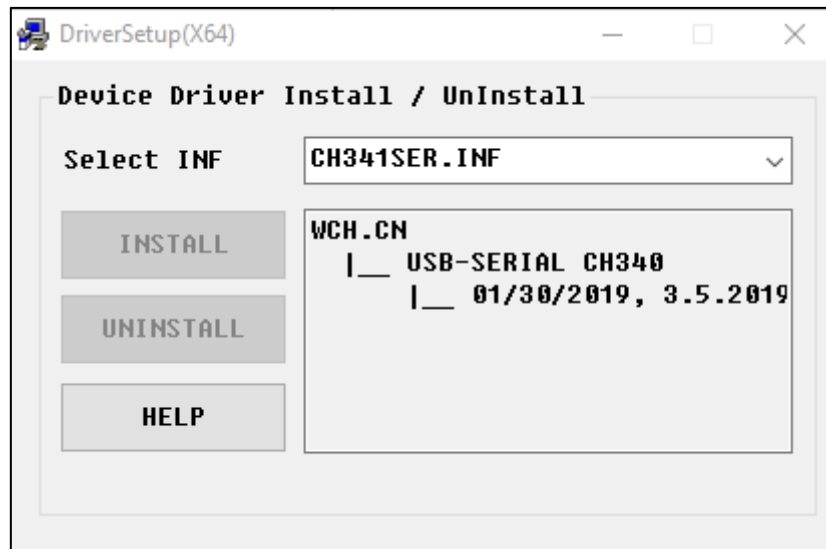
2. Open the folder "Freenove\_ESP32\_WROVER\_Board/CH340/Windows/"



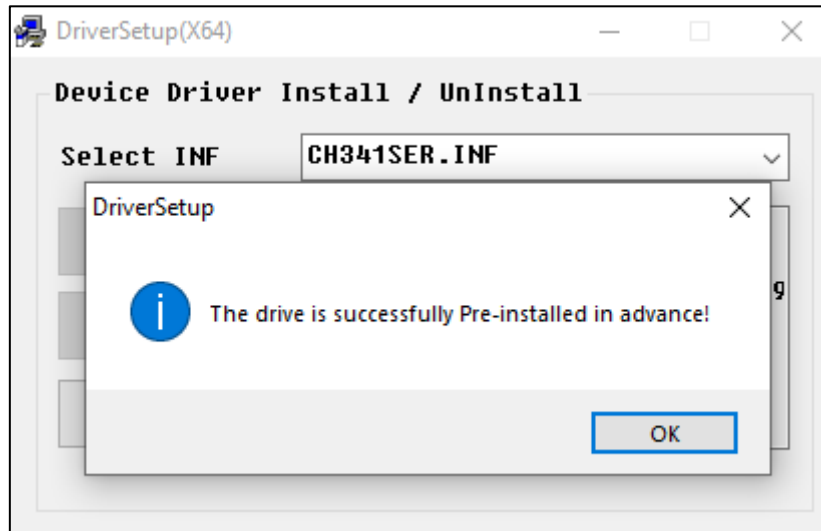
3. Double click "CH341SER.EXE".



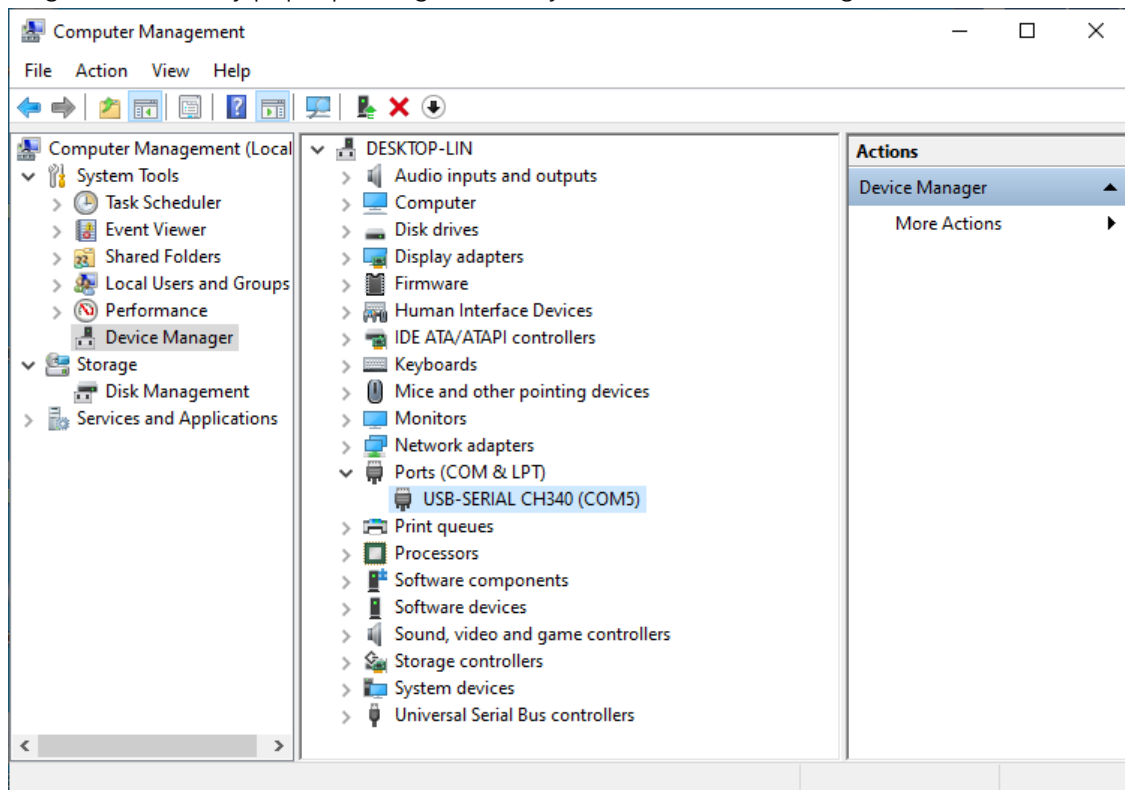
- Click "INSTALL" and wait for the installation to complete.



- Install successfully. Close all interfaces.



6. When ESP32 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, CH340 has been installed successfully. Close all dialog boxes.

## MAC

First, download CH340 driver, click <http://www.wch-ic.com/search?q=CH340&t=downloads> to download the appropriate one based on your operating system.

keyword ch340

Downloads( 7 )

file category	file content	version	upload time
Driver&Tools			
CH341SER.EXE	CH340/CH341 USB to serial port Windows driver, supports 32/64-bit Windows 10/8.1/8/7/VISTA/XP, Server 2016/2012/2008/2003, 2000/ME/98	3.5	2019-03-18
CH341SER.ZIP	CH340/CH341 USB to serial port Windows driver, includes DLL dynamic library and non-standard baud rate settings and other instructions. Supports 32/64-bit Windows 10/8.1/8/7/VISTA/XP, Server 2016/2012/2008/2003, 2000/ME/98	3.5	2019-03-05
CH341SER_ANDROID...	CH340/CH341 USB to serial port Android free drive application library, for Android OS 3.1 and above version which supports USB Host mode already, no need to load Android kernel driver, no root privileges. Contains apk, lib library file (Java Driver), App Demo Example, AT Demo SDK).	1.6	2019-04-19
CH341SER_LINUX...	CH340/CH341 USB to serial port LINUX driver	1.5	2018-03-18
CH341SER_MAC.ZI...	CH340/CH341 USB to serial port MAC OS driver	1.5	2018-07-05
Others			

Callouts: Windows, Linux, MAC

If you would not like to download the installation package, you can open “Freenove\_ESP32\_WROVER\_Board/CH340”, we have prepared the installation package. Second, open the folder “Freenove\_ESP32\_WROVER\_Board/CH340/MAC/”

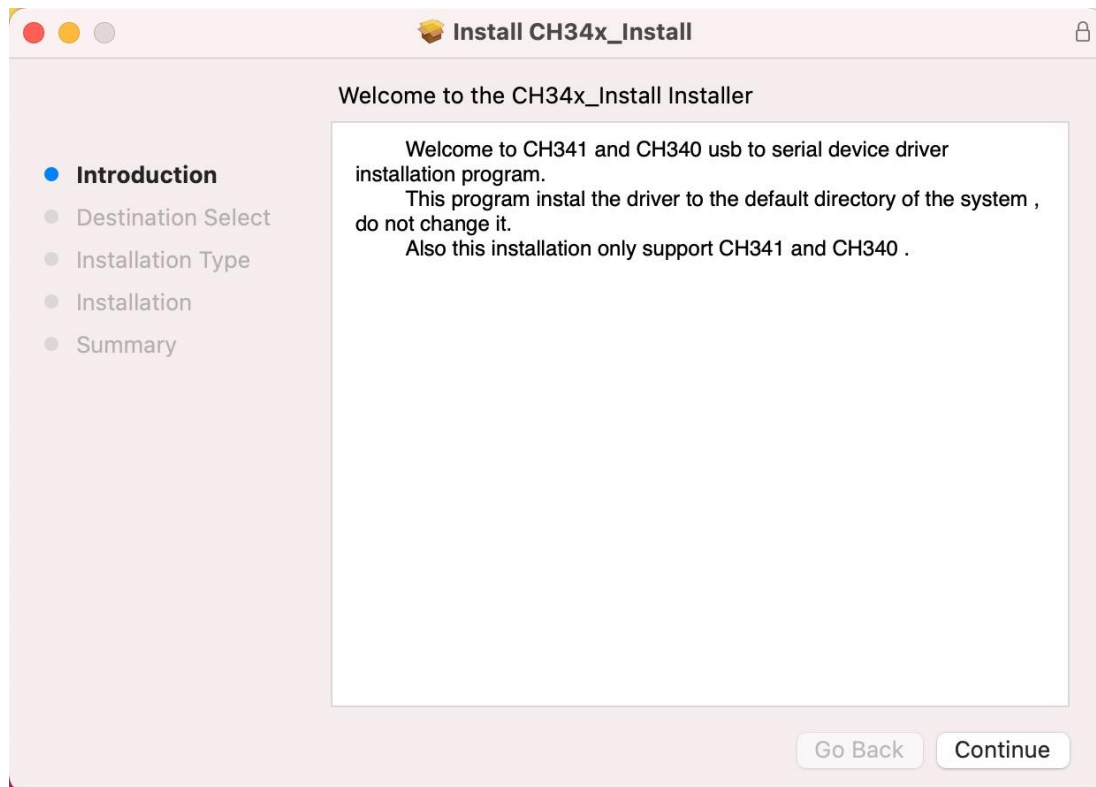
MAC

Name	Modified	Size	Kind
CH34x_Install_V1.5.pkg	2018 at 2:00 PM	26 KB	Installer package
ReadMe.pdf	Dec 8, 2016 at 4:09 PM	146 KB	Adobe...ocument

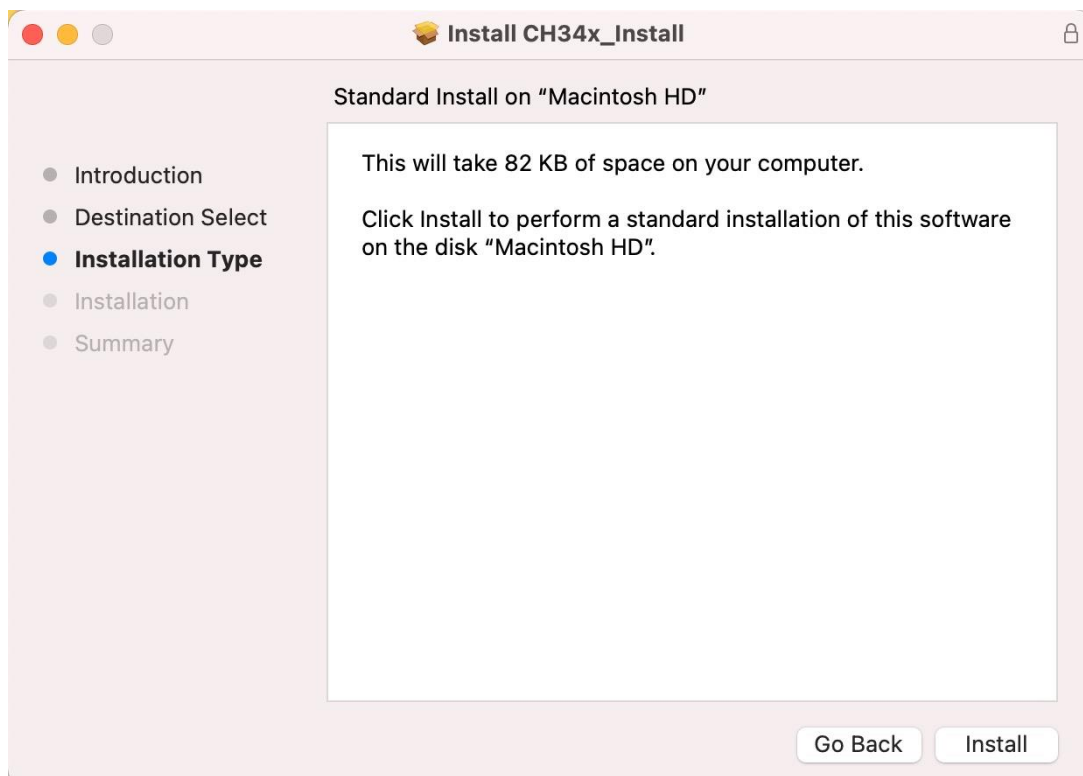
Callout: Run it.

Any concerns? [✉ support@freenove.com](mailto:support@freenove.com)

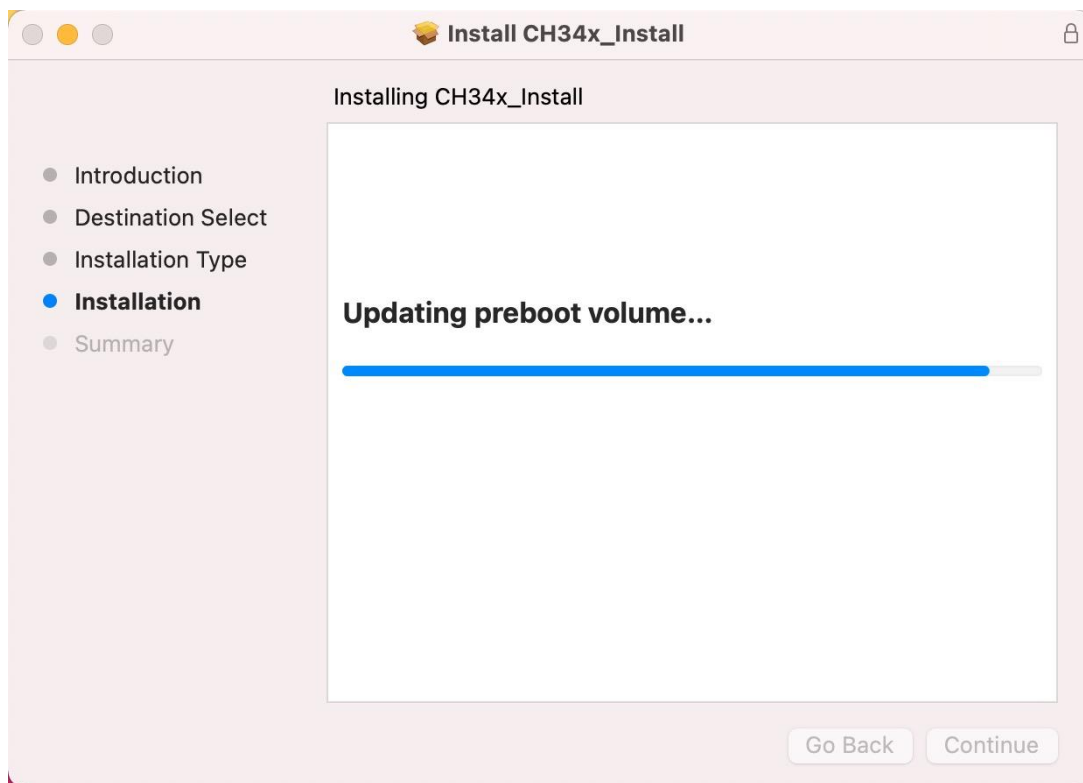
Third, click Continue.



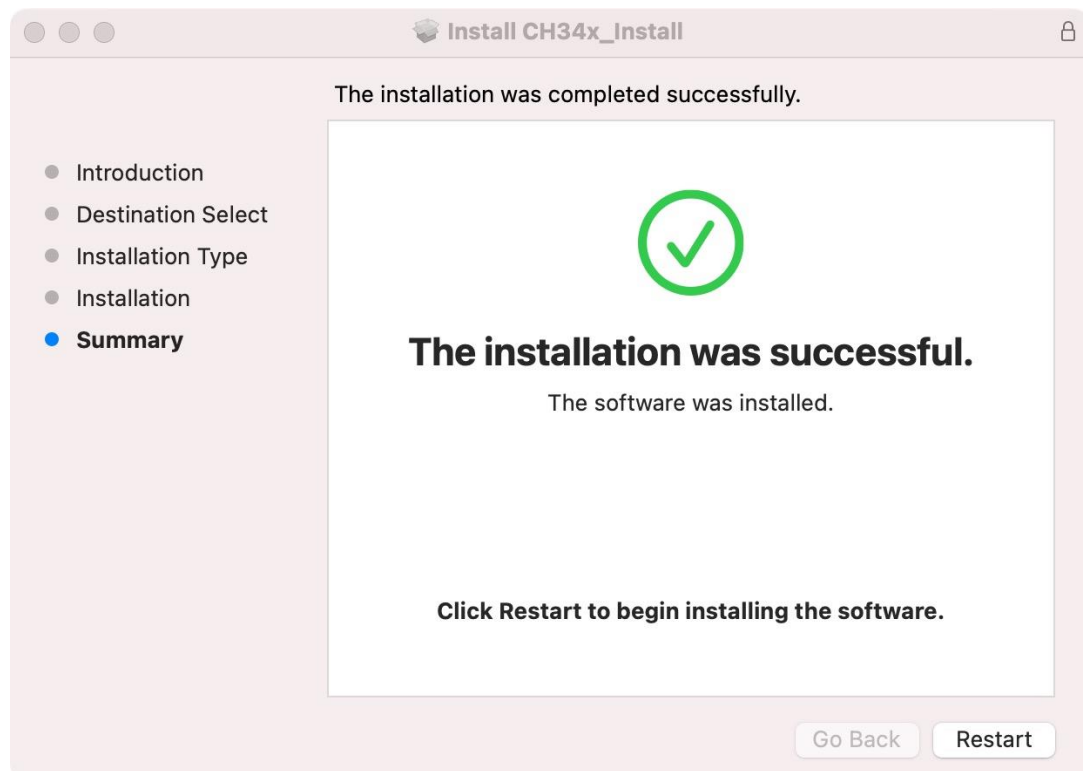
Fourth, click Install.



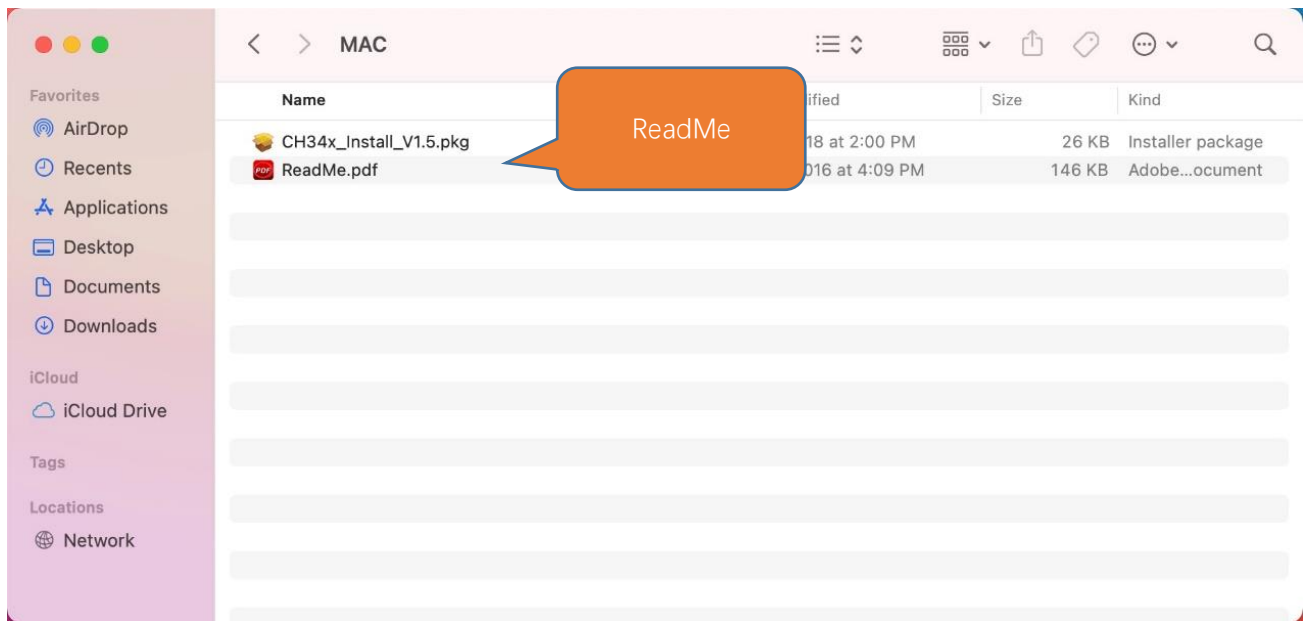
Then, waiting Finish.



Finally, restart your PC.



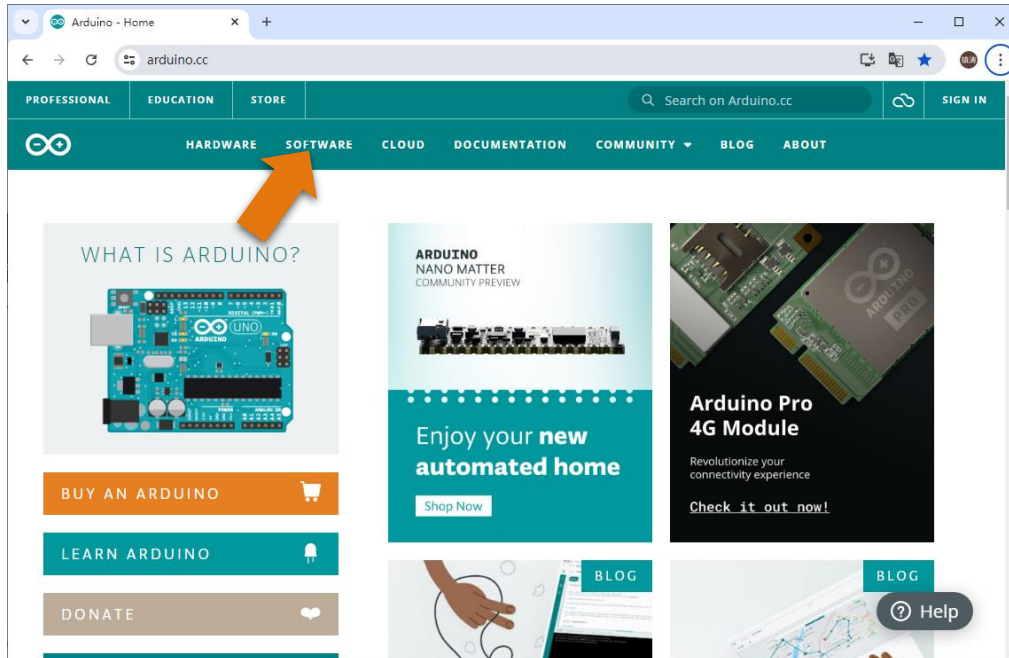
If you still haven't installed the CH340 by following the steps above, you can view readme.pdf to install it.



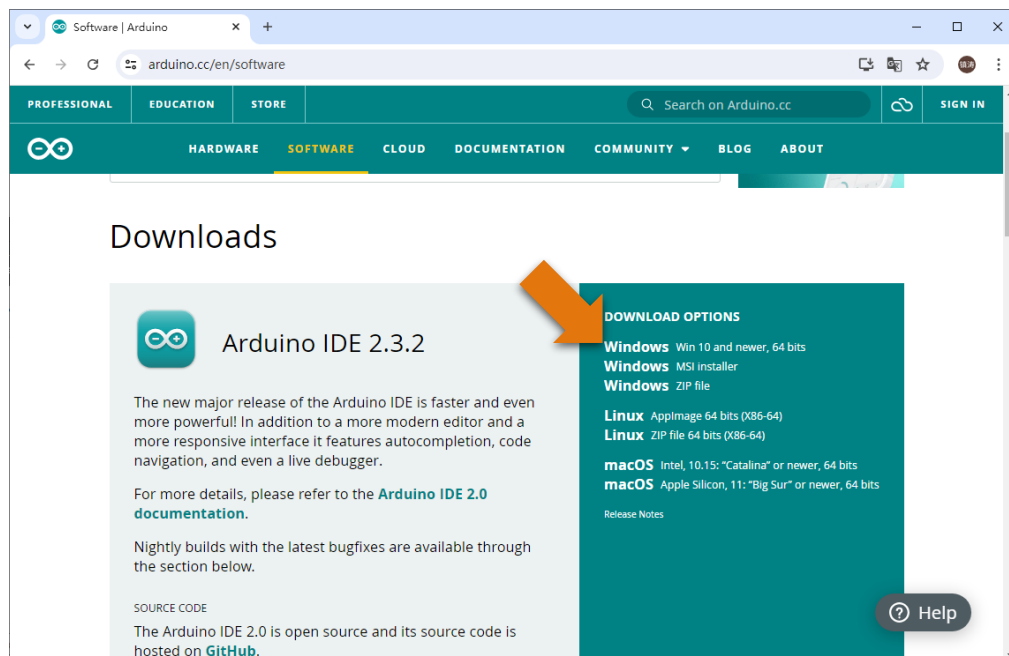
## Programming Software

Arduino Software (IDE) is used to write and upload the code for Arduino Board.

First, install Arduino Software (IDE): visit <https://www.arduino.cc>, 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.

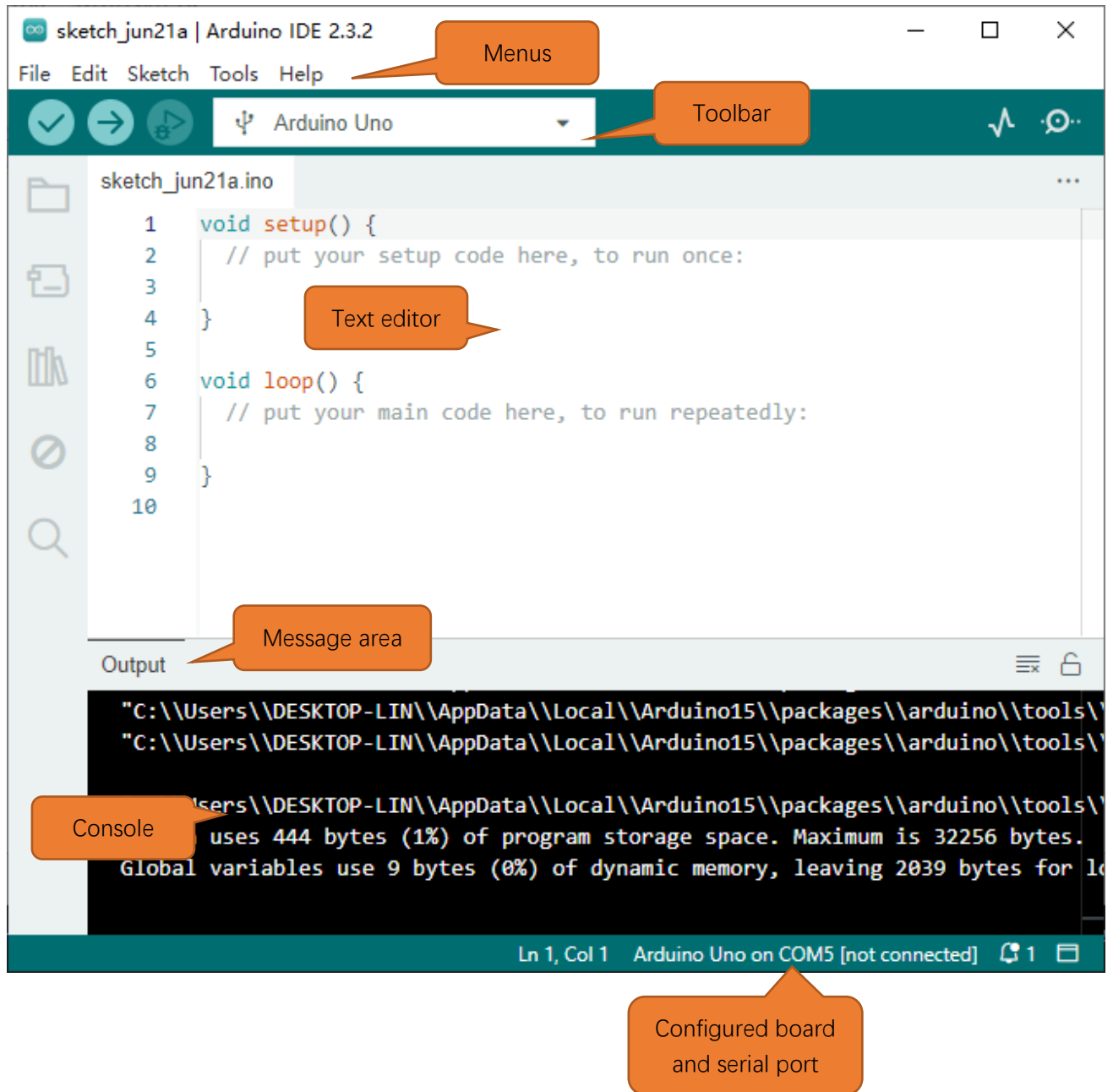


After the download completes, run the installer. For Windows users, there may pop up an installation dialog box of driver during the installation process. When it popes up, please allow the installation.

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  
Check your code for compile errors .



Upload  
Compile your code and upload them to the configured board.



Debug  
With a debugger or USB port, debug the code.



Serial Plotter  
Open the serial plotter.

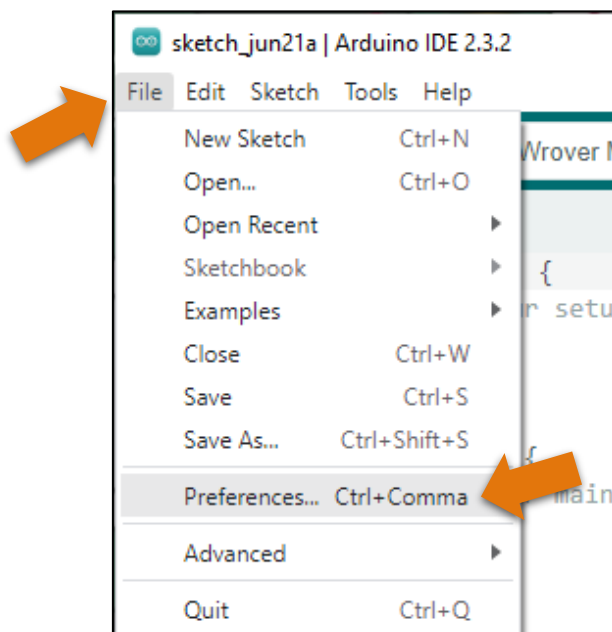


Serial Monitor  
Open the serial monitor.

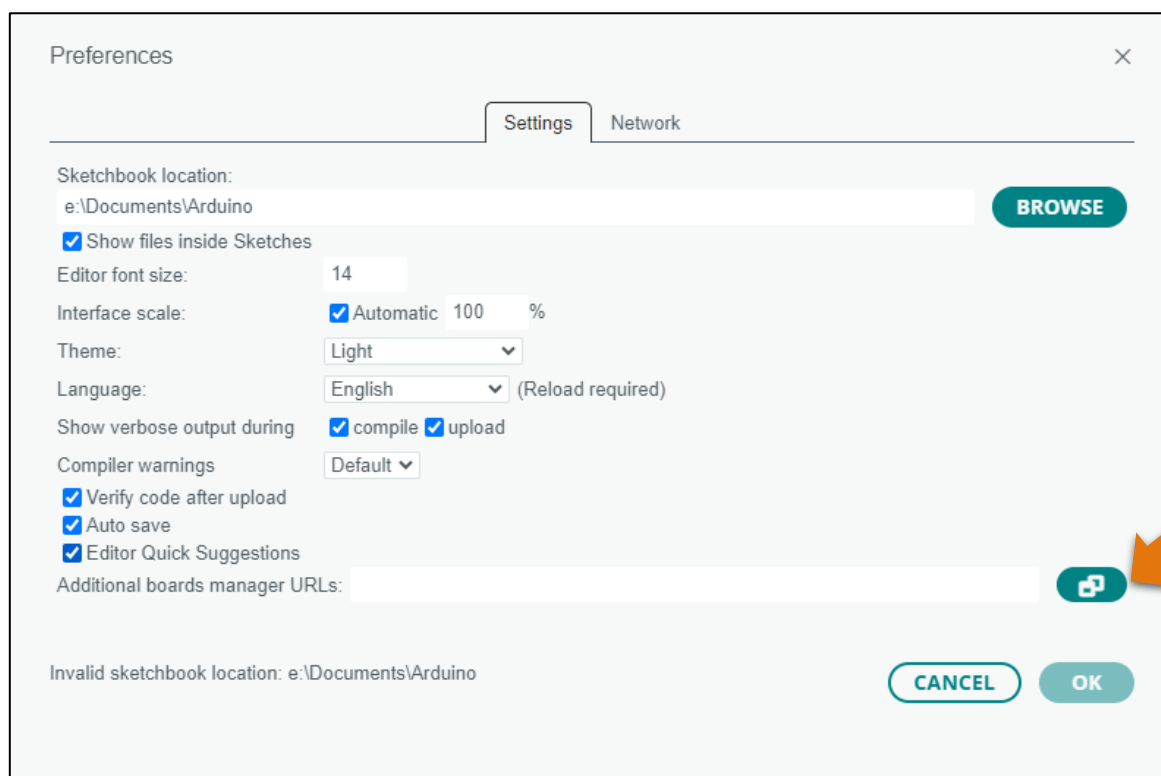
Additional commands are found within the five menus: File, Edit, Sketch, Tools, Help. The menus are context sensitive, which means only those items relevant to the work currently being carried out are available.

## Environment Configuration

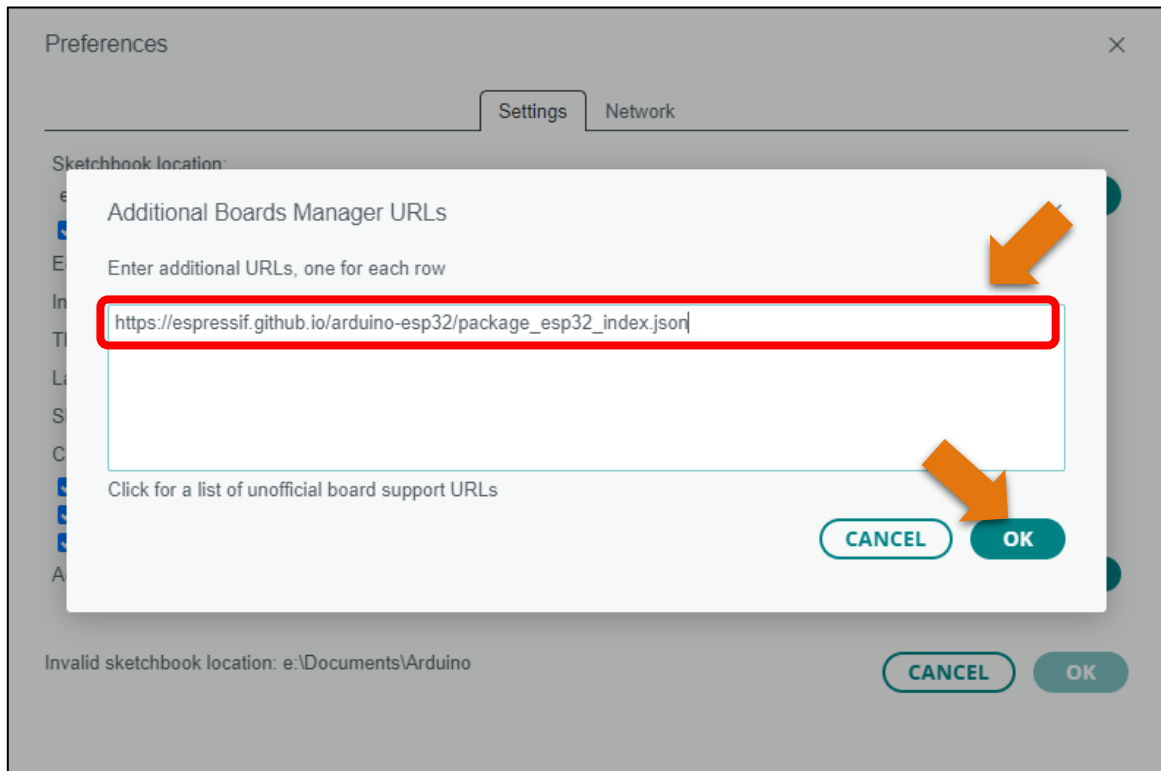
First, open the software platform arduino, and then click File in Menus and select Preferences.



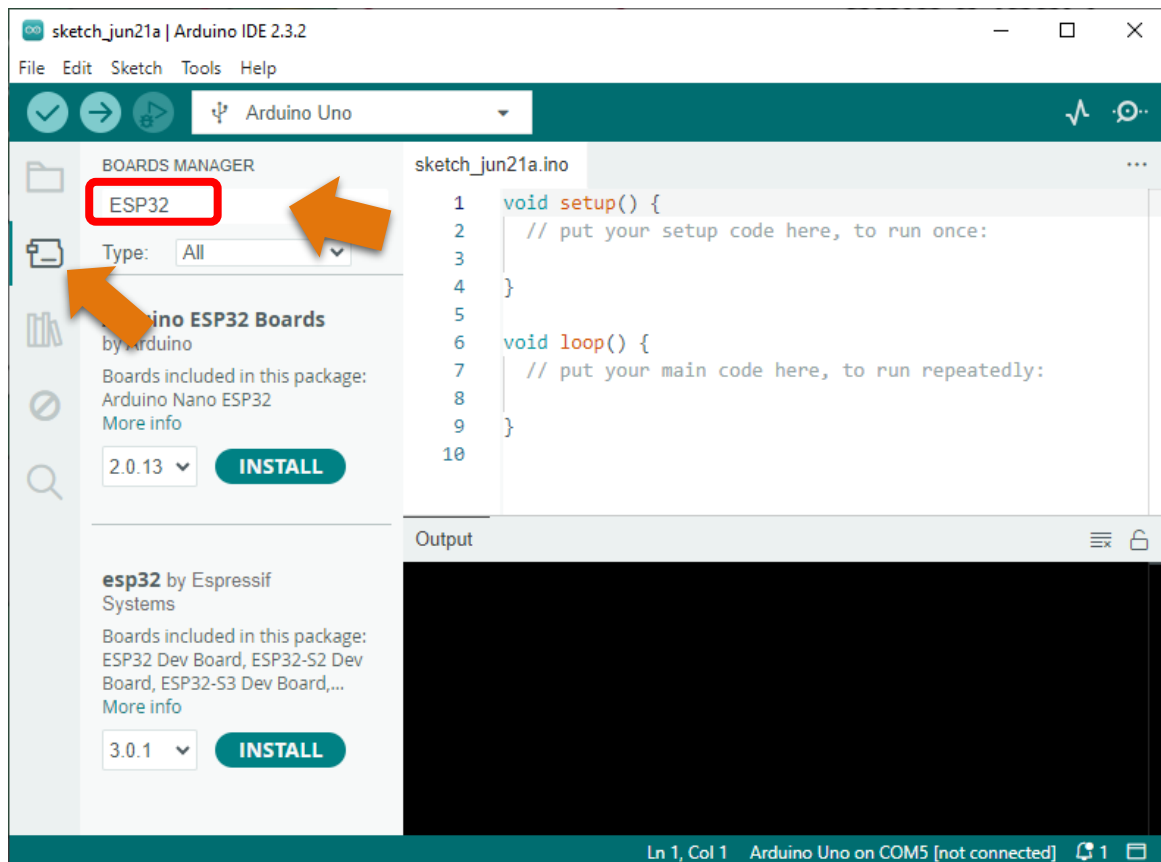
Second, click on the symbol behind "Additional Boards Manager URLs"



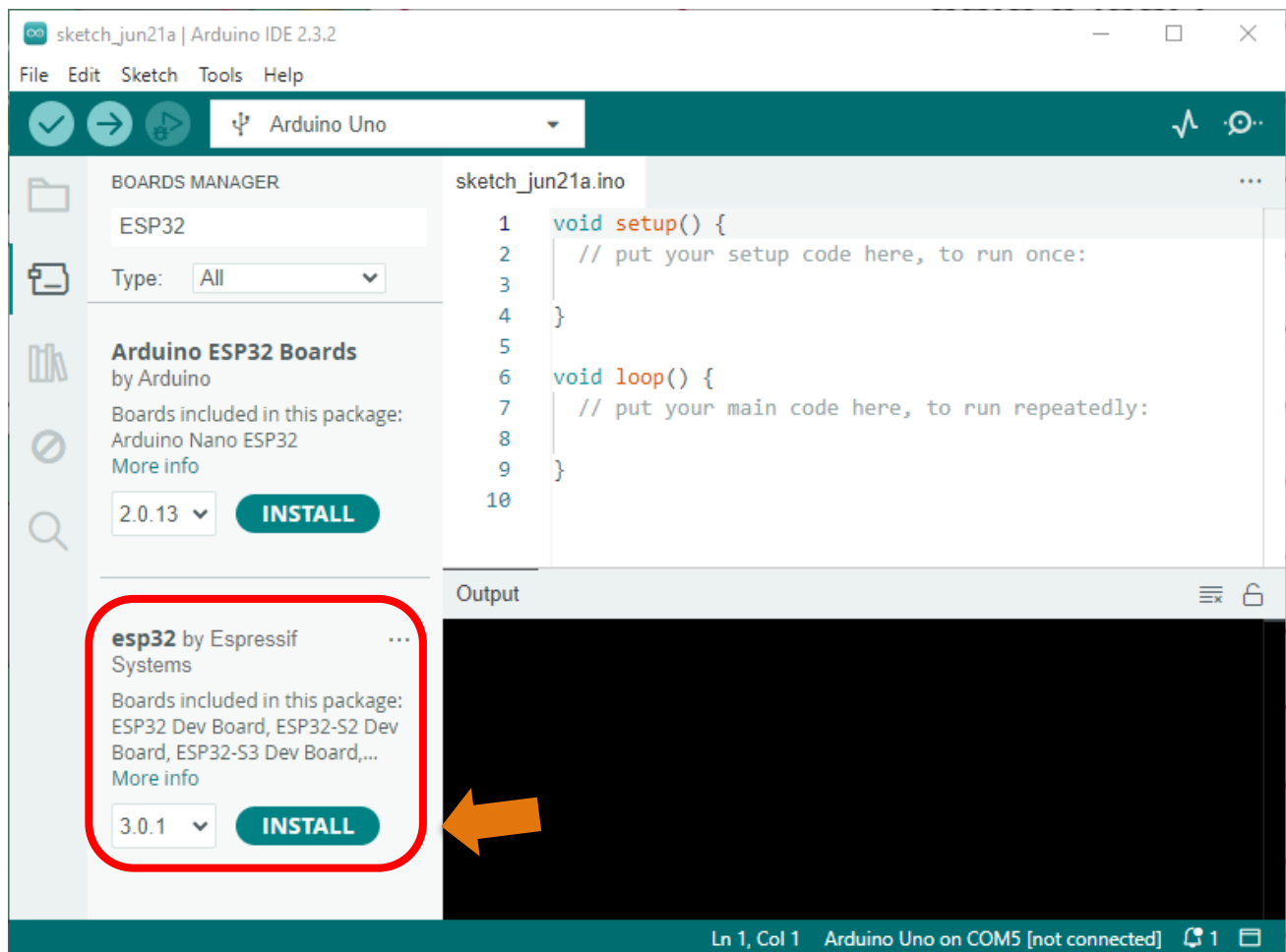
Third, fill in [https://espressif.github.io/arduino-esp32/package\\_esp32\\_index.json](https://espressif.github.io/arduino-esp32/package_esp32_index.json) in the new window, click OK, and click OK on the Preferences window again.



Fourth, click "BOARDS MANAGER" on the left and type "ESP32" in the search box.

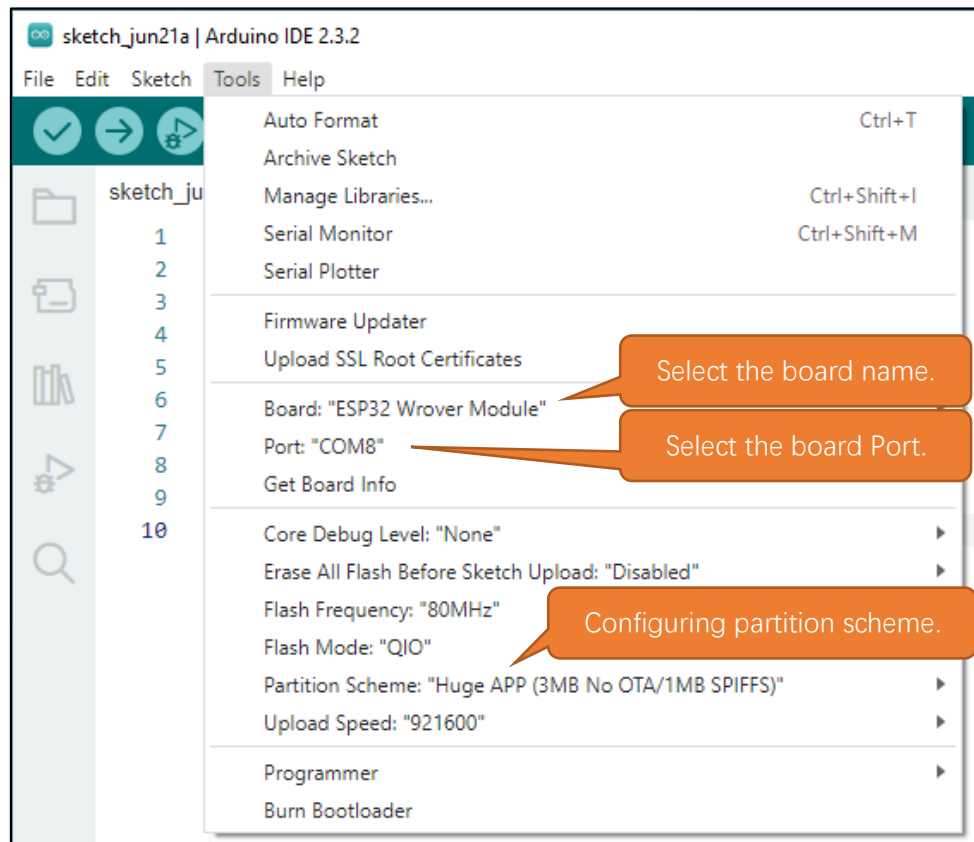
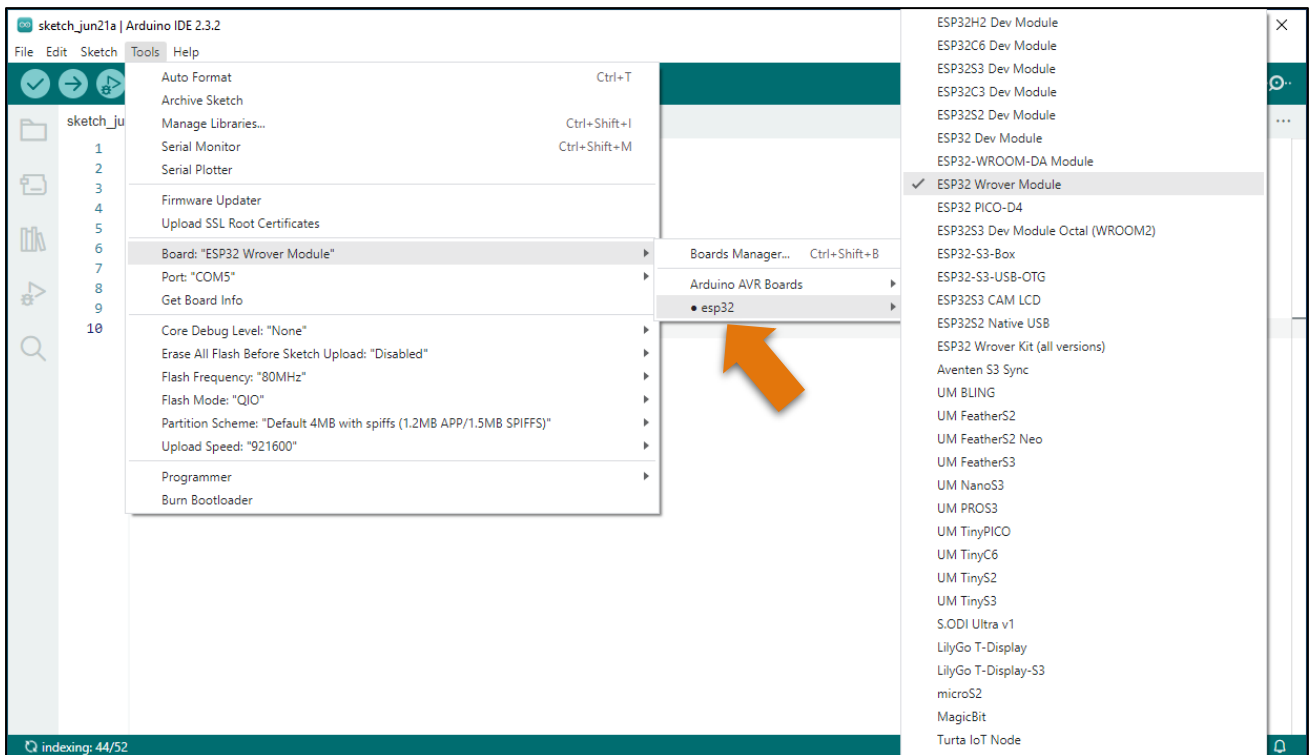


Fifth, select Espressif Systems' ESP32 and select version 3.0.x. Click "INSTALL" to install esp32.



Note: it takes a while to install the ESP32, make sure your network is stable.

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



## Notes for GPIO

### Strapping Pin

There are five Strapping pins for ESP32: MTDI、GPIO0、GPIO2、MTDO、GPIO5。

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'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 five strapping pins at power-on and their functions under the corresponding configuration.

Voltage of Internal LDO (VDD_SDIO)					
Pin	Default	3.3 V		1.8 V	
MTDI	Pull-down	0		1	
Bootling Mode					
Pin	Default	SPI Boot		Download Boot	
GPIO0	Pull-up	1		0	
GPIO2	Pull-down	Don't-care		0	
Enabling/Disabling Debugging Log Print over U0TXD During Bootling					
Pin	Default	U0TXD Active		U0TXD Silent	
MTDO	Pull-up	1		0	
Timing of SDIO Slave					
Pin	Default	Falling-edge Sampling Falling-edge Output	Falling-edge Sampling Rising-edge Output	Rising-edge Sampling Falling-edge Output	Rising-edge Sampling Rising-edge Output
MTDO	Pull-up	0	0	1	1
GPIO5	Pull-up	0	1	0	1

**Note:**

- Firmware can configure register bits to change the settings of "Voltage of Internal LDO (VDD\_SDIO)" and "Timing of SDIO Slave" after bootling.
- The MTDI is internally pulled high in the module, as the flash and SRAM in ESP32-WROVER only support a power voltage of 1.8 V (output by VDD\_SDIO).

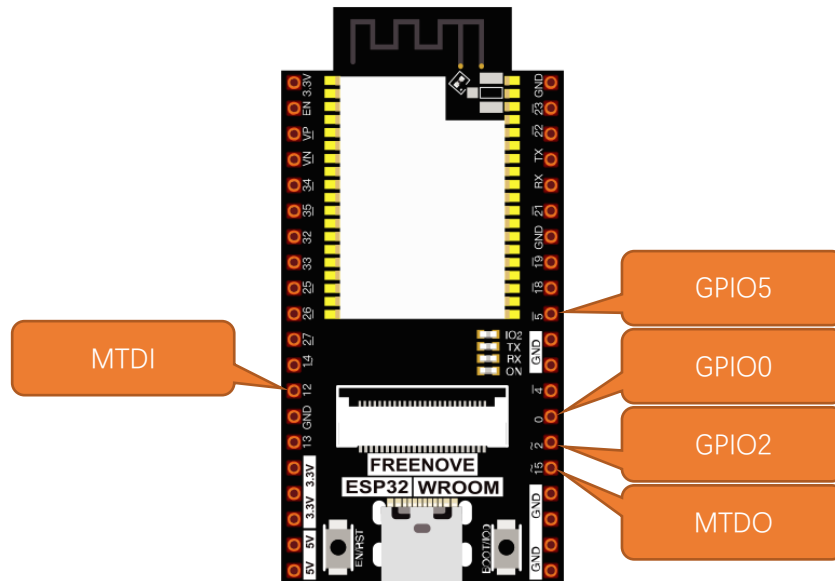
If you have any questions about the information of GPIO, you can click [here](#) to go back to ESP32-WROVER to view specific information about GPIO.

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-wrover\\_datasheet\\_en.pdf](https://www.espressif.com/sites/default/files/documentation/esp32-wrover_datasheet_en.pdf)

**Any concerns?** ✉ [support@freenove.com](mailto:support@freenove.com)

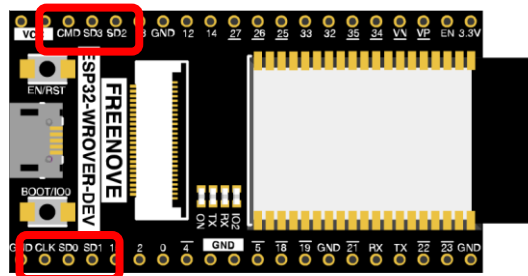
## Strapping Pin



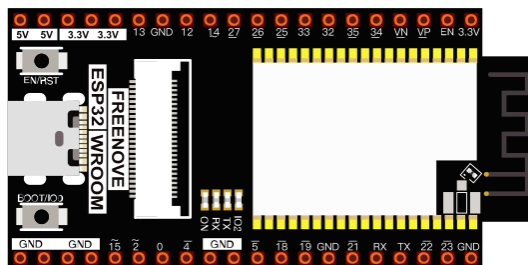
## Flash Pin

GPIO6-11 has been used to connect the integrated SPI flash on the module, and is used when GPIO 0 is power on and at high level. Flash is related to the operation of the whole chip, so the external pin GPIO6-11 cannot be used as an experimental pin for external circuits, otherwise it may cause errors in the operation of the program.

In older versions, the flash pin looks like the image below.



In the new release, we no longer introduce GPIO6-11.



GPIO16-17 has been used to connect the integrated PSRAM on the module.

Because of external pull-up, MTDI pin is not suggested to be used as a touch sensor. For details, please refer to Peripheral Interface and Sensor chapter in "ESP32 Data\_Sheet".

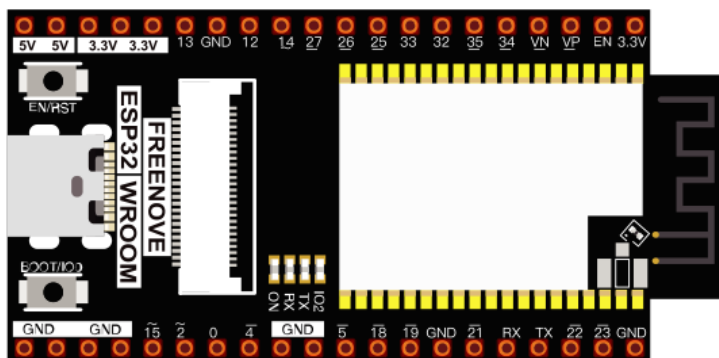
For more relevant information, please check:

[https://www.espressif.com/sites/default/files/documentation/esp32-wrover\\_datasheet\\_en.pdf](https://www.espressif.com/sites/default/files/documentation/esp32-wrover_datasheet_en.pdf).

## Cam Pin

When using the camera of our ESP32-WROVER, please check the pins of it.

**Pins with underlined numbers are used by the camera function**, if you want to use other functions besides it, please avoid using them.



CAM_Pin	GPIO_pin
I2C_SDA	GPIO26
I2C_SCL	GPIO27
CSI_VSYNC	GPIO25
CSI_HREF	GPIO23
CSI_Y9	GPIO35
XCLK	GPIO21
CSI_Y8	GPIO34
CSI_Y7	GPIO39
CSI_PCLK	GPIO22
CSI_Y6	GPIO36
CSI_Y2	GPIO4
CSI_Y5	GPIO19
CSI_Y3	GPIO5
CSI_Y4	GPIO18

If you have any questions about the information of GPIO, you can click [here](#) to go back to ESP32-WROVER to view specific information about GPIO.

or check: [https://www.espressif.com/sites/default/files/documentation/esp32-wrover\\_datasheet\\_en.pdf](https://www.espressif.com/sites/default/files/documentation/esp32-wrover_datasheet_en.pdf).

# Chapter 1 LED

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

## Project 1.1 Blink

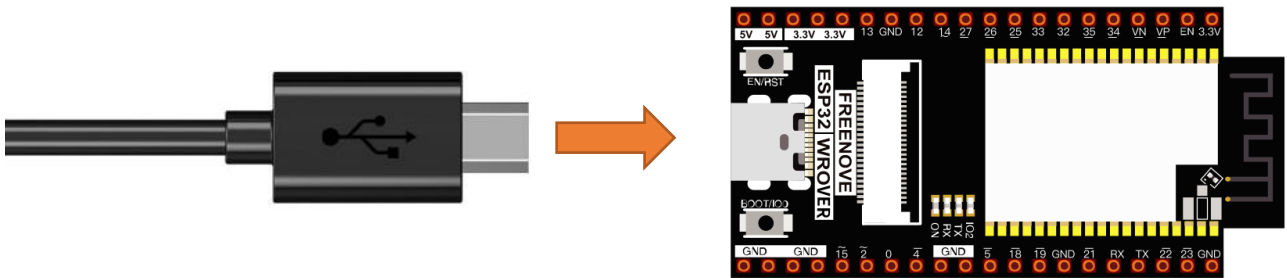
In this project, we will use ESP32 to control blinking a common LED.

## Component List

ESP32-WROVER x1	USB cable
-----------------	-----------

### Power

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



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

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

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

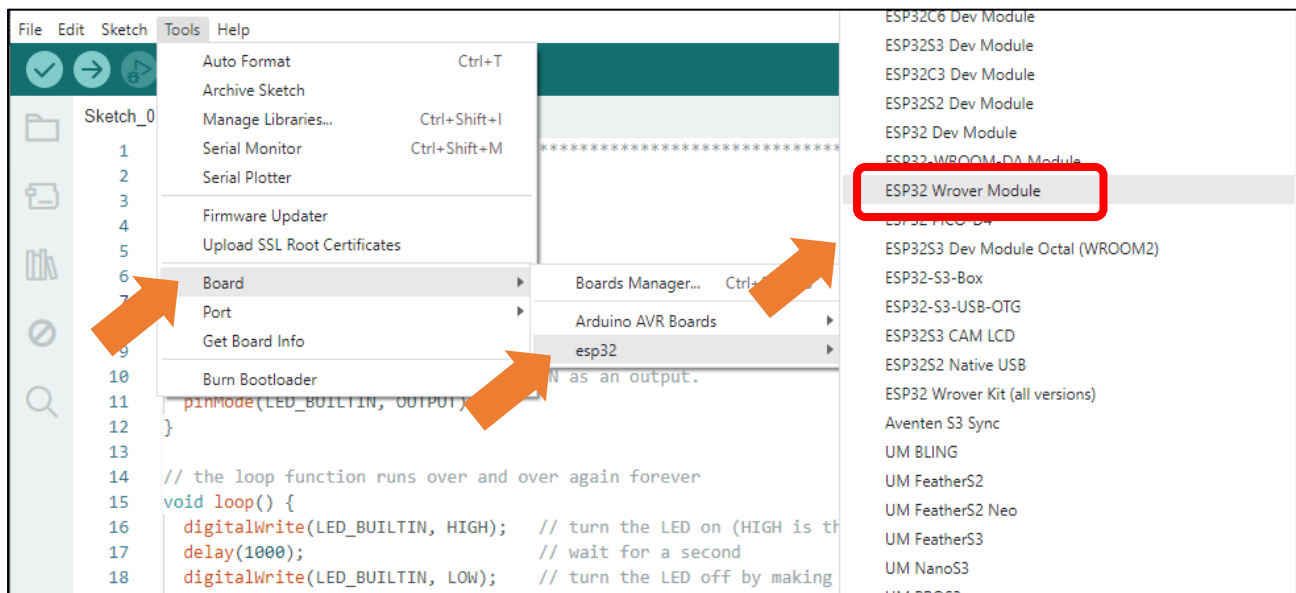
## Sketch

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

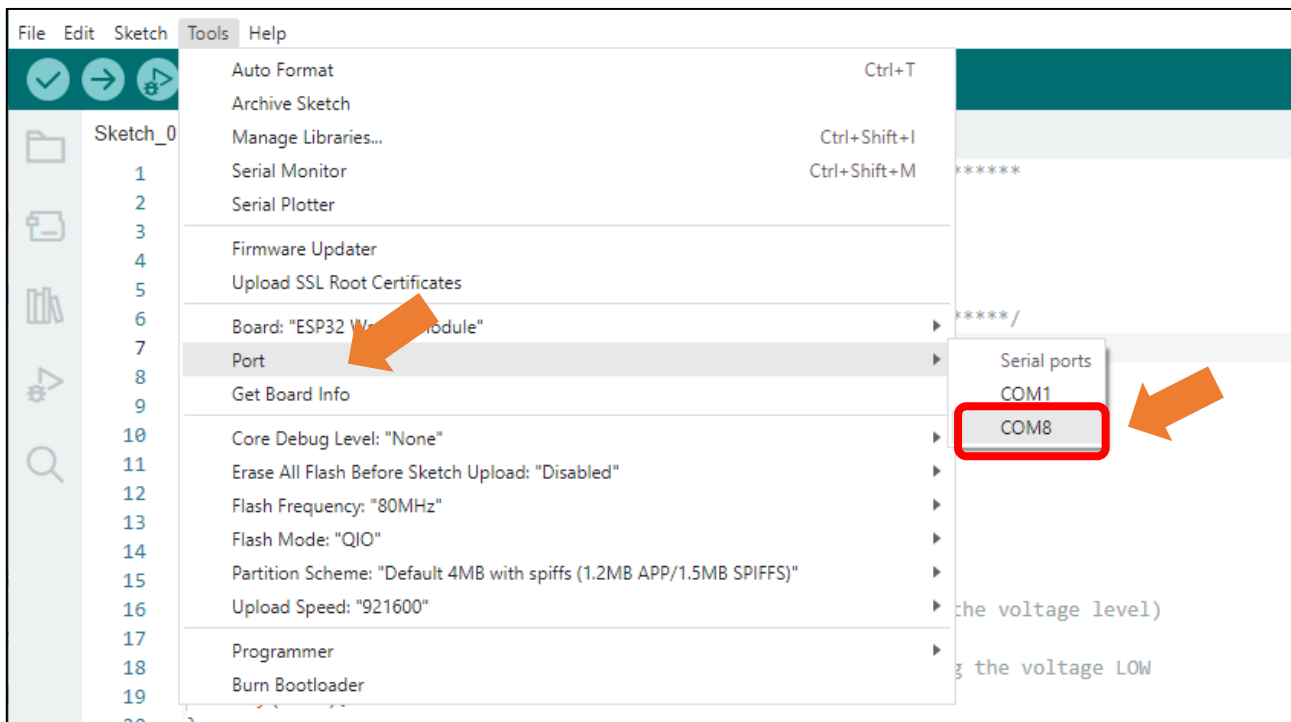
Upload the following Sketch:

**Freenove\_ESP32\_WROVER\_Board\Sketches\Sketch\_01.1\_Blink.**

Before uploading the code, click "Tools", "Board" and select "ESP32 Wrover Module".

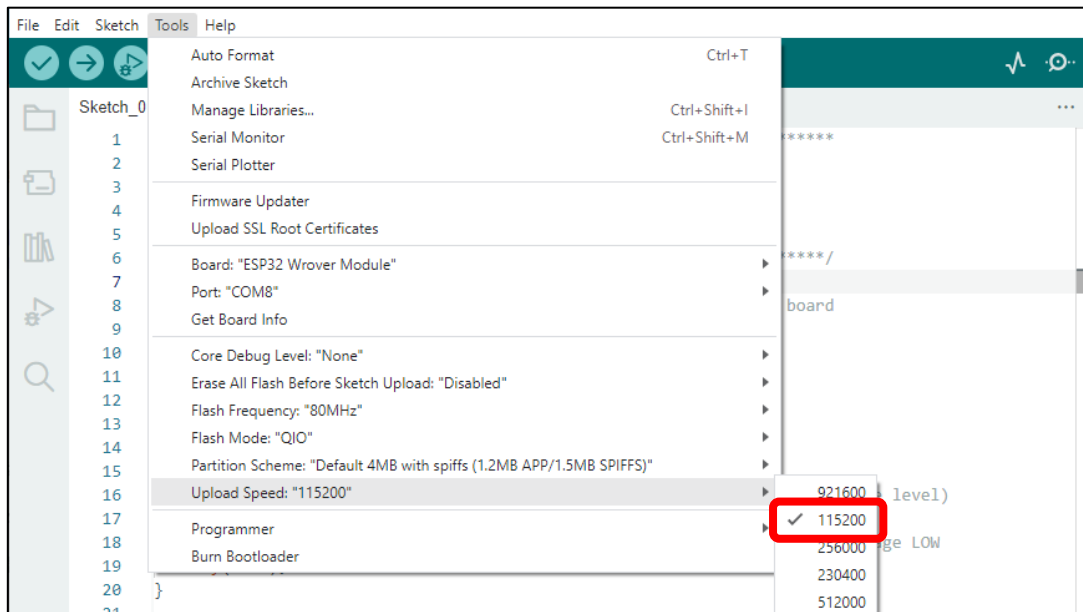


Select the serial port.

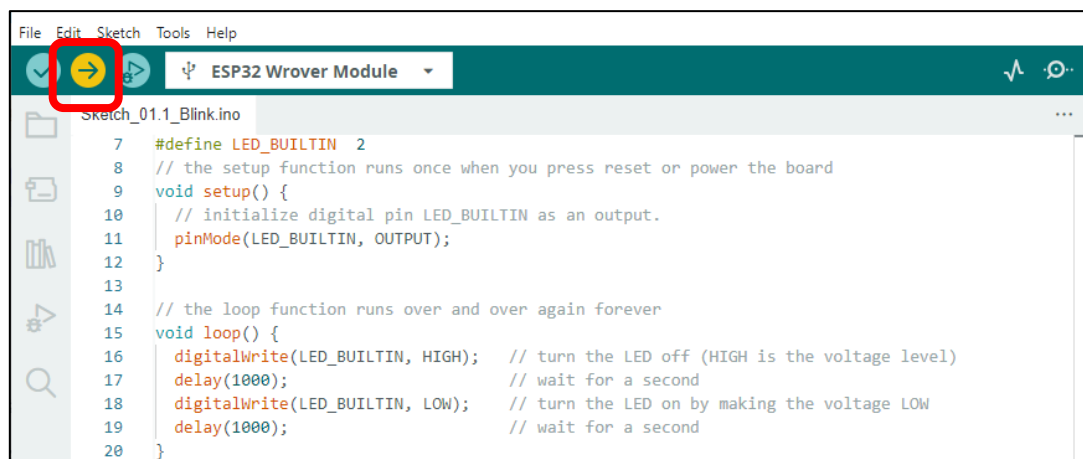


**Note:** For macOS users, if the uploading fails, please set the baud rate to 115200 before clicking

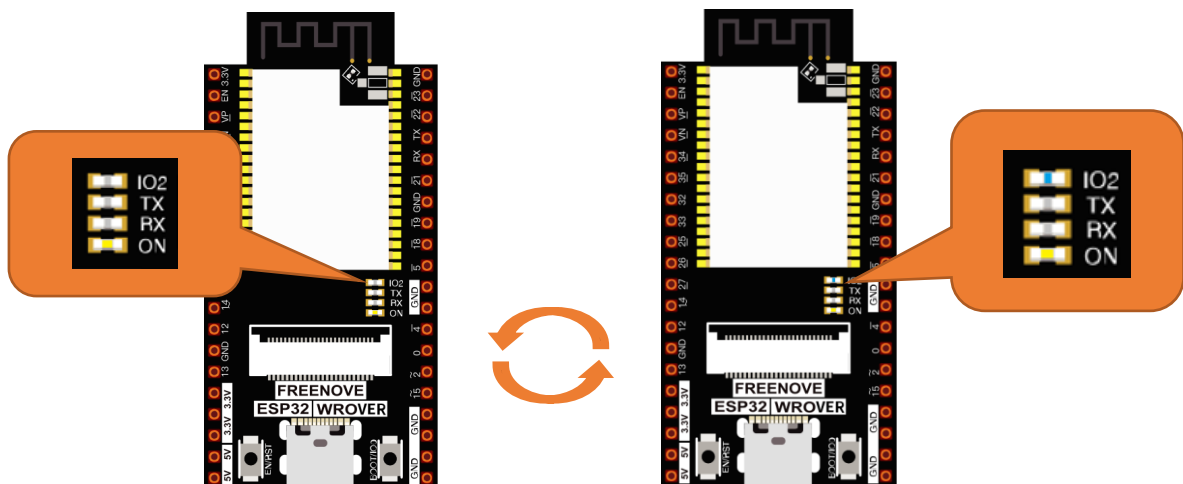
“Upload Speed”.



Sketch\_01.1\_Blink



Click “Upload”, Download the code to ESP32-WROVER and your LED in the circuit starts Blink.



If you have any concerns, please contact us via: [support@freenove.com](mailto:support@freenove.com)

Any concerns? [support@freenove.com](mailto:support@freenove.com)

The following is the program code:

```

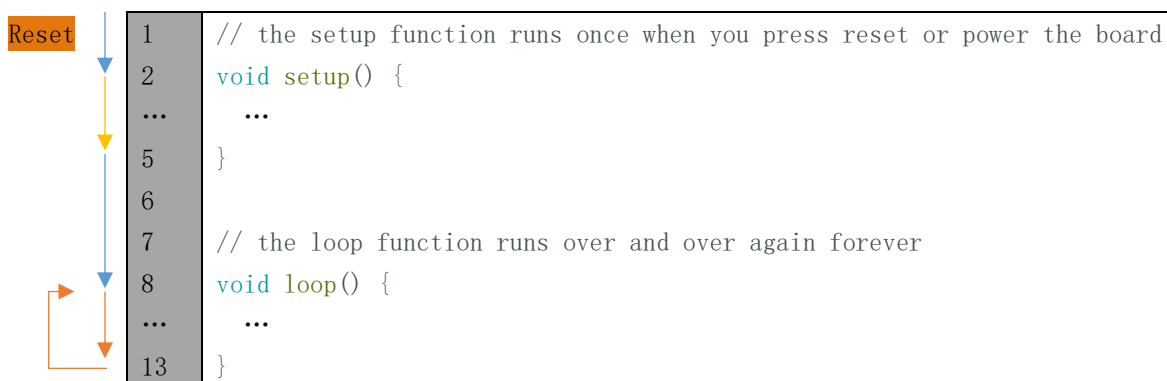
1  #define PIN_LED 2
2  // the setup function runs once when you press reset or power the board
3  void setup() {
4      // initialize digital pin LED_BUILTIN as an output.
5      pinMode(PIN_LED, OUTPUT);
6  }
7
8  // the loop function runs over and over again forever
9  void loop() {
10     digitalWrite(PIN_LED,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

```

The Arduino IDE code usually contains two basic functions: void setup() and void loop().

After the board is reset, the setup() function will be executed firstly, and then the loop() function.

setup() function is generally used to write code to initialize the hardware. And loop() function is used to write code to achieve certain functions. loop() function is executed repeatedly. When the execution reaches the end of loop(), it will jump to the beginning of loop() to run again.



#### 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-WROVER's GPIO2 is connected to the LED, so the LED pin is defined as 2.

```

1  #define PIN_LED 2

```

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

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

```

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

```

Then, in the loop () function, set the PIN\_LED to output high level to make LED light off.

```

10 digitalWrite(PIN_LED, HIGH); // turn the LED off (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 PIN\_LED to output low level, and LED light up. One second later, the execution of loop () function will be completed.

```
12    digitalWrite(PIN_LED, LOW); // turn the LED on 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\_PULLDOWN, 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/>

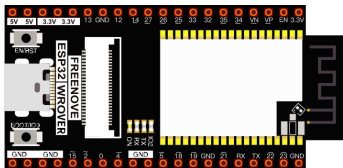
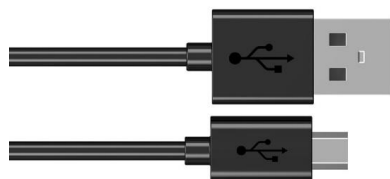
## Chapter 2 Bluetooth

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

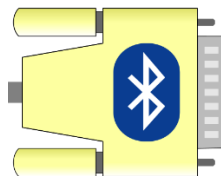
Project 2.1 is classic Bluetooth and Project 2.2 is low power Bluetooth. If you are an iPhone user, please start with Project 2.2.

### Project 2.1 Bluetooth Passthrough

#### Component List

ESP32-WROVER x1 	Micro USB Wire x1 
---	---

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



## Component knowledge

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

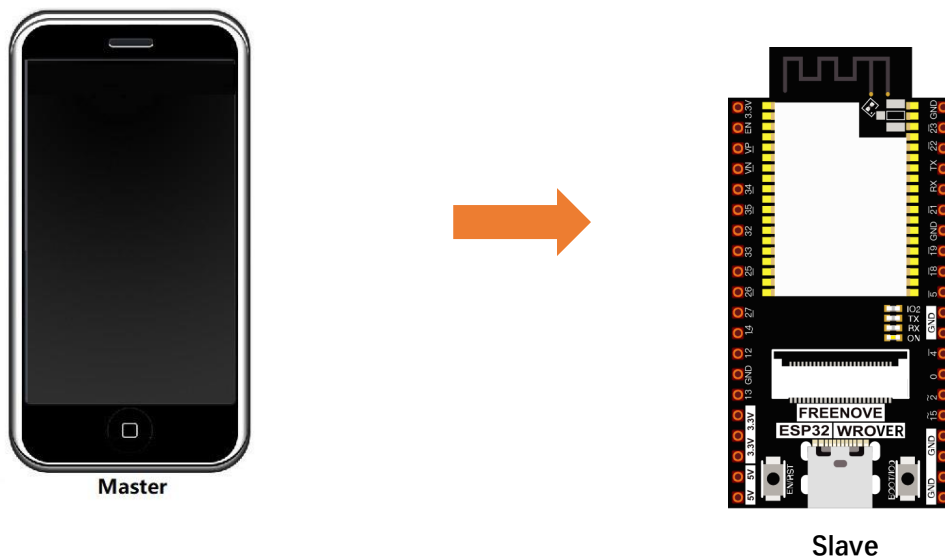
### Master mode

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

### Slave mode

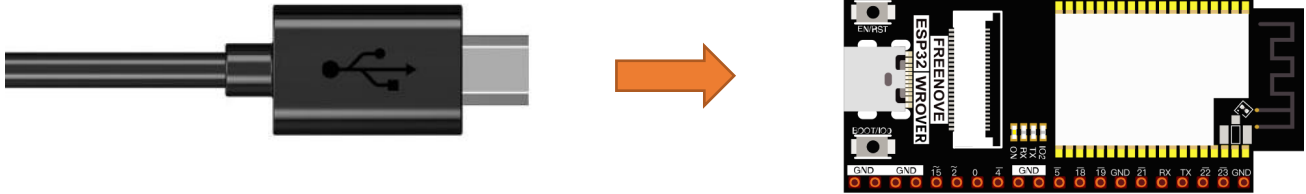
The Bluetooth module in slave mode can only accept connection request from a host computer, but cannot initiate a connection request. After connecting with a host device, it can send data to or receive from the host device.

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



## Circuit

Connect Freenove ESP32 to the computer using the USB cable.



## Sketch

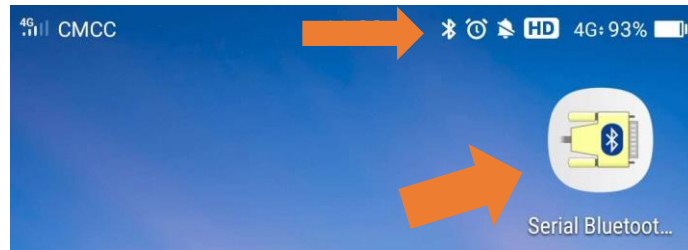
### Sketch\_02.1\_SerialToSerialBT

```
File Edit Sketch Tools Help
ESP32 Wrover Module
Sketch_27.1_SerialToSerialBT.ino
7  #include "BluetoothSerial.h"
8
9  BluetoothSerial SerialBT;
10 String buffer;
11 void setup() {
12     Serial.begin(115200);
13     SerialBT.begin("ESP32test"); //Bluetooth device name
14     Serial.println("\nThe device started, now you can pair it with bluetooth!");
15 }
16
17 void loop() {
18     if (Serial.available()) {
19         SerialBT.write(Serial.read());
20     }
21     if (SerialBT.available()) {
22         Serial.write(SerialBT.read());
23     }
24     delay(20);
25 }
```

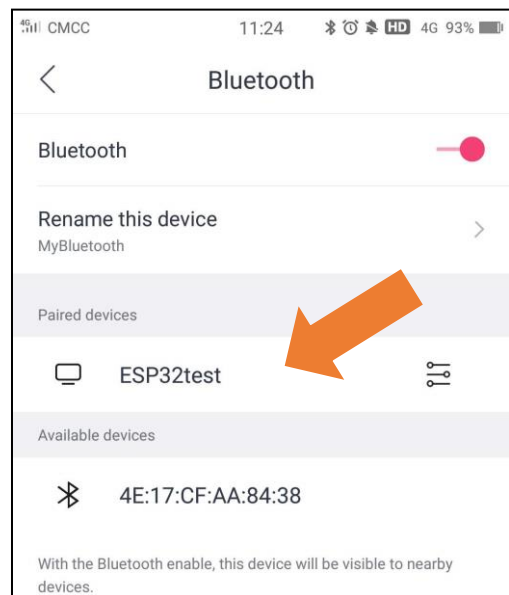
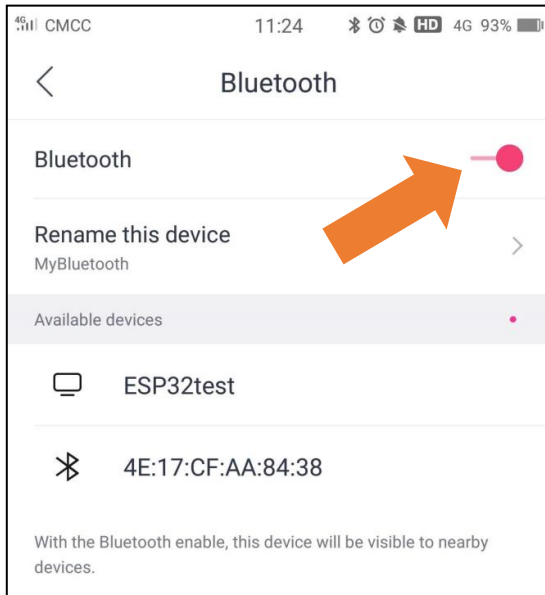
Compile and upload the code to the ESP32-WROVER, open the serial monitor, and set the baud rate to 115200. When you see the serial printing out the character string as below, it indicates that the Bluetooth of ESP32 is ready and waiting to connect with the mobile phone.

```
11:07:06.801 -> rst:0x1 (POWERON_RESET),boot:0x1b (SPI_FAST_FLASH_BOOT)
11:07:06.842 -> configsip: 0, SPIWP:0xee
11:07:06.842 -> clk_drv:0x00,q_drv:0x00,d_drv:0x00,cs0_drv:0x00,hd_drv:0x00,wp_drv:0x00
11:07:06.842 -> mode:DIO, clock div:1
11:07:06.843 -> load:0x3fff0030,len:1448
11:07:06.843 -> load:0x40078000,len:14844
11:07:06.843 -> ho 0 tail 12 room 4
11:07:06.843 -> load:0x40080400,len:4
11:07:06.843 -> load:0x40080404,len:3356
11:07:06.843 -> entry 0x4008059c
11:07:08.127 ->
11:07:08.127 -> The device started, now you can pair it with bluetooth!
```

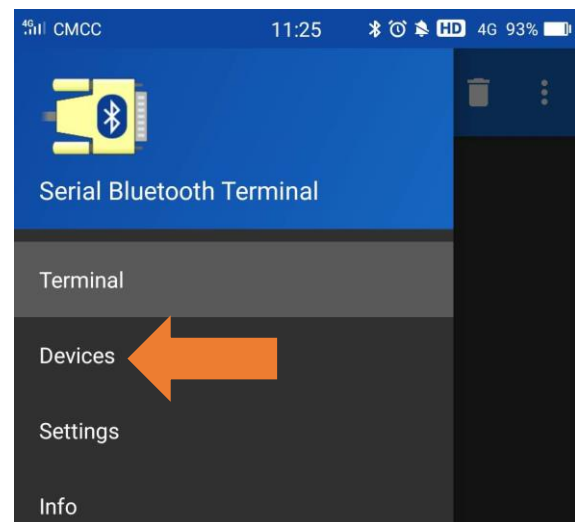
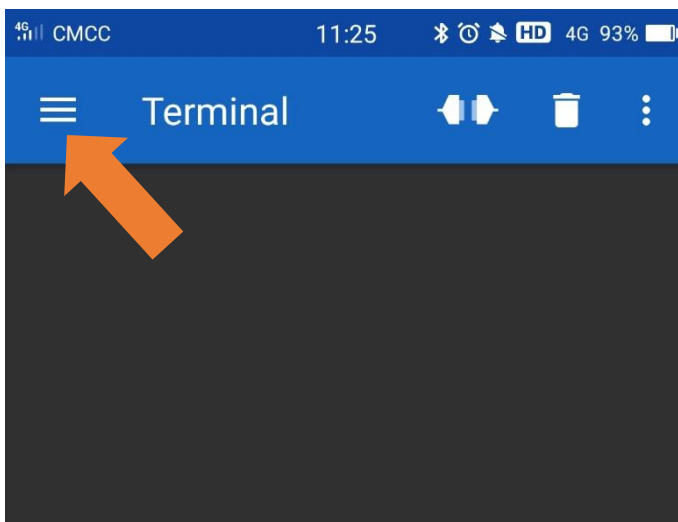
Make sure that the Bluetooth of your phone has been turned on and Serial Bluetooth Terminal has been installed.



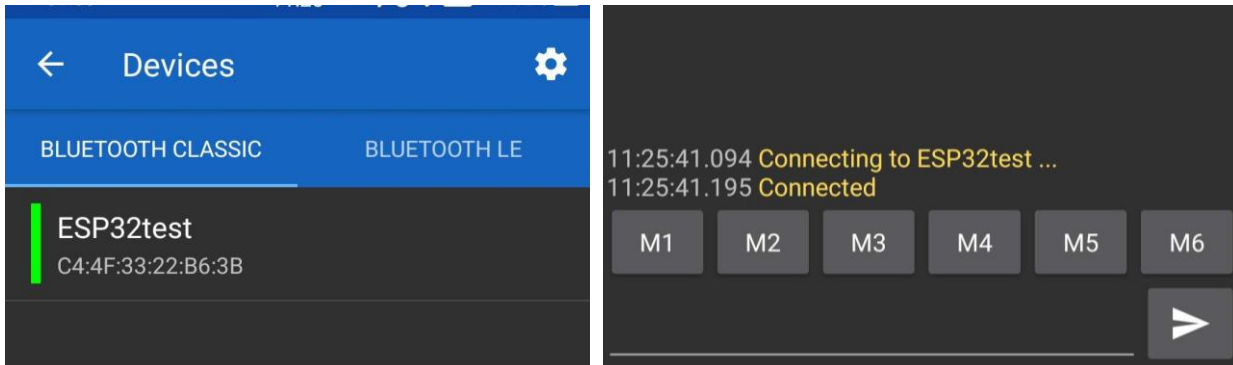
Click "Search" to search Bluetooth devices nearby and select "ESP32 test" to connect to.



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

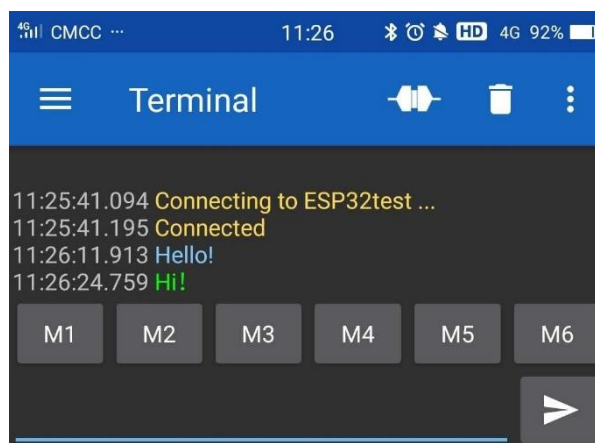
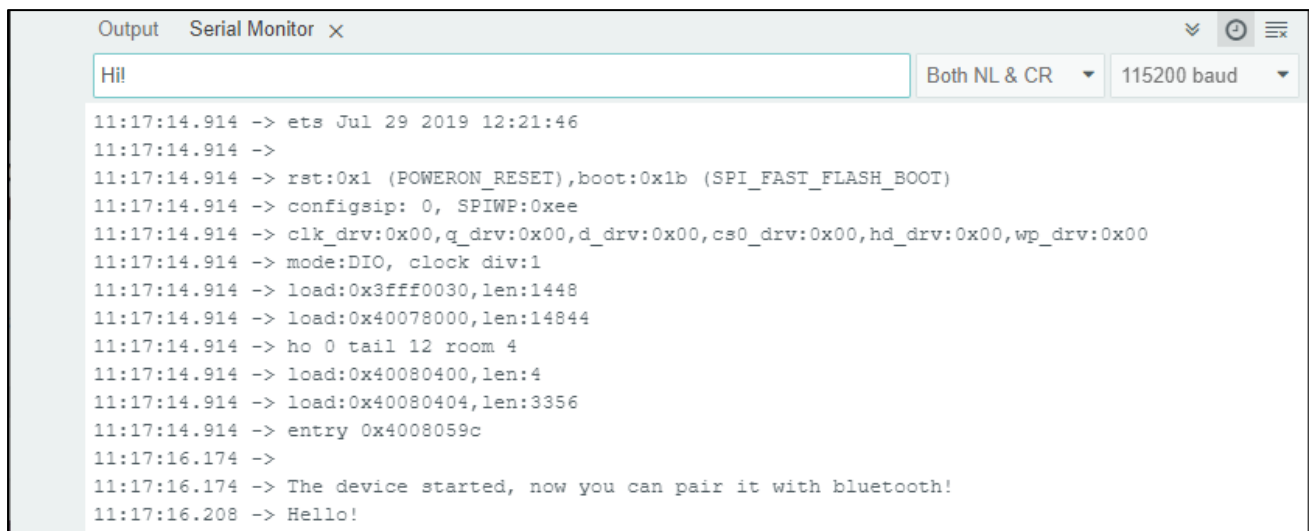


Select ESP32test in classic Bluetooth mode, and a successful connecting prompt will appear as shown on the right illustration.



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

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



## Reference

**Class BluetoothSerial**

This is a class library used to operate **BluetoothSerial**, which can directly read and set **BluetoothSerial**.

Here are some member functions:

**begin(localName, isMaster)**: Initialization function of the Bluetooth

name: name of Bluetooth module; Data type: String

**isMaster**: bool type, whether to set Bluetooth as Master. By default, it is false.

**available()**: acquire digits sent from the buffer, if not, return 0.

**read()**: read data from Bluetooth, data type of return value is int.

**readString()**: read data from Bluetooth, data type of return value is String.

**write(val)**: send an int data val to Bluetooth.

**write(str)**: send an String data str to Bluetooth.

**write(buf, len)**: Sends the first len data in the buf Array to Bluetooth.

**setPin(const char \*pin)**: set a four-digit Bluetooth pairing code. By default, it is 1234

**connect(remoteName)**: connect a Bluetooth named remoteName, data type: String

**connect(remoteAddress[])**: connect the physical address of Bluetooth, data type: uint8-t.

**disconnect()**: disconnect all Bluetooth devices.

**end()**: disconnect all Bluetooth devices and turn off the Bluetooth, release all occupied space

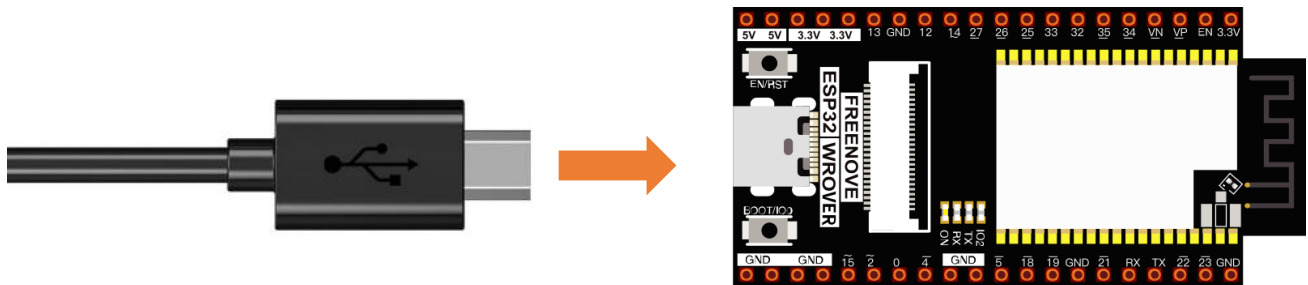
## Project 2.2 Bluetooth Low Energy Data Passthrough

### Component List

ESP32-WROVER x1	Micro USB Wire x1
-----------------	-------------------

### Circuit

Connect Freenove ESP32 to the computer using the USB cable.



## Sketch

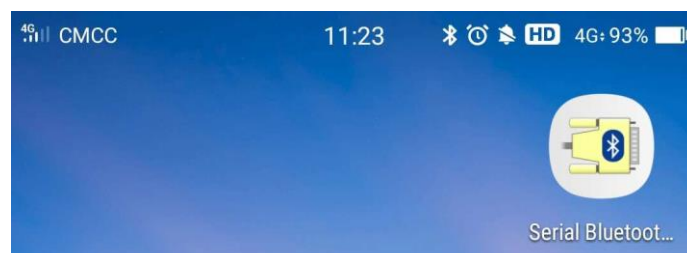
### Sketch\_02.2\_BLE



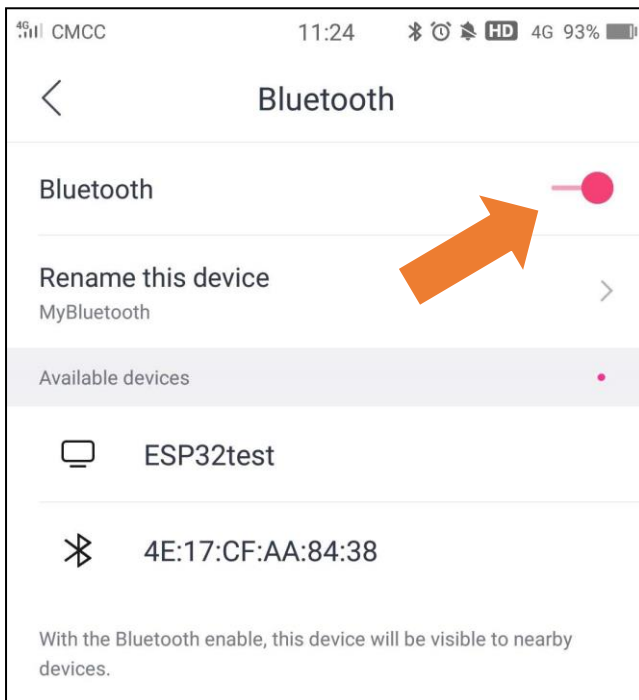
### Serial Bluetooth

Compile and upload code to ESP32, the operation is similar to the last section.

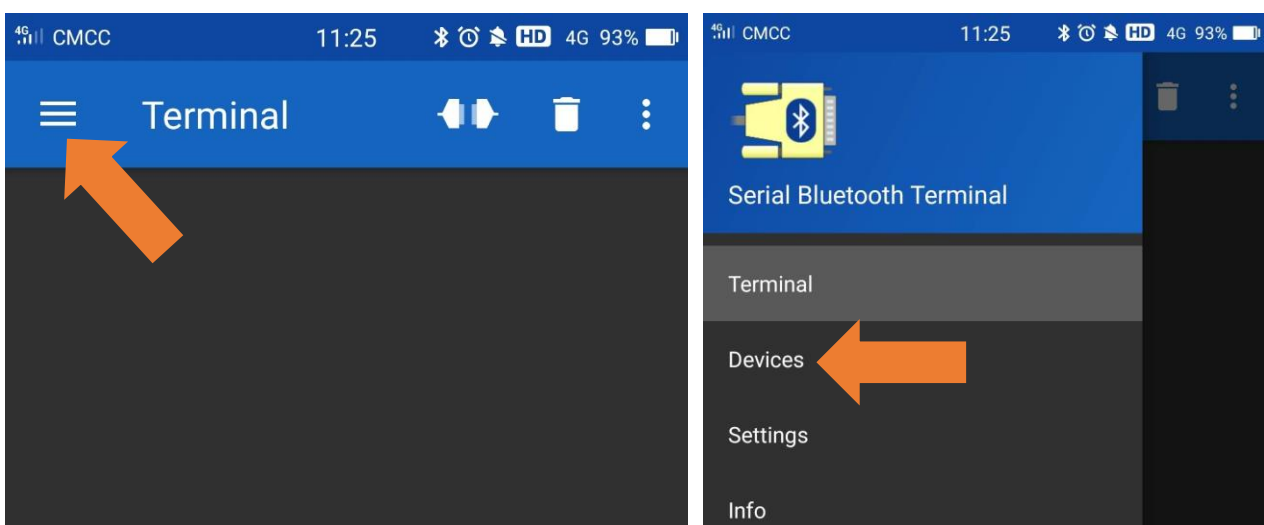
First, make sure you've turned on the mobile phone Bluetooth, and then open the software.



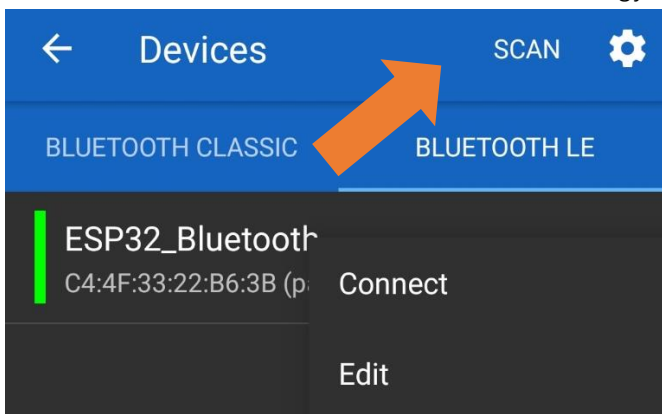
Click "Search" to search Bluetooth devices nearby and select "ESP32 test" to connect to.



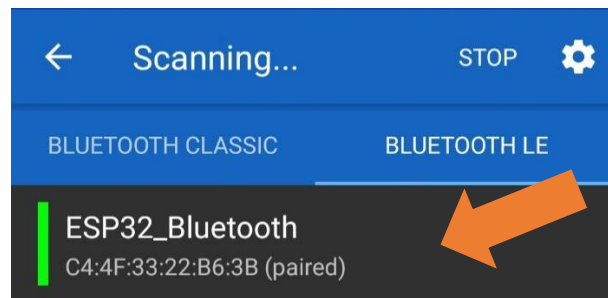
Turn on software APP, click the left of the terminal. Select "Devices"



Select BLUETOOTHLE, click SCAN to scan Low Energy Bluetooth devices nearby.



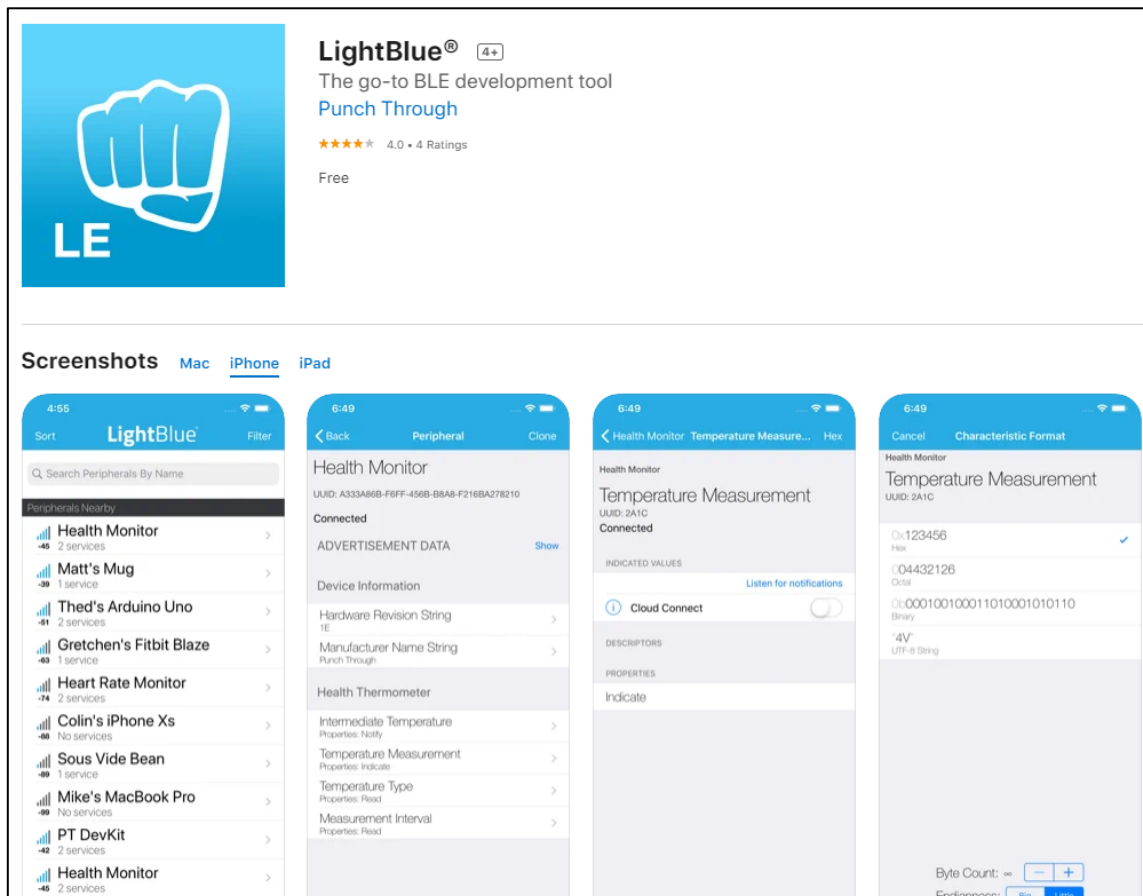
Select "ESP32-Bluetooth"



### Lightblue

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

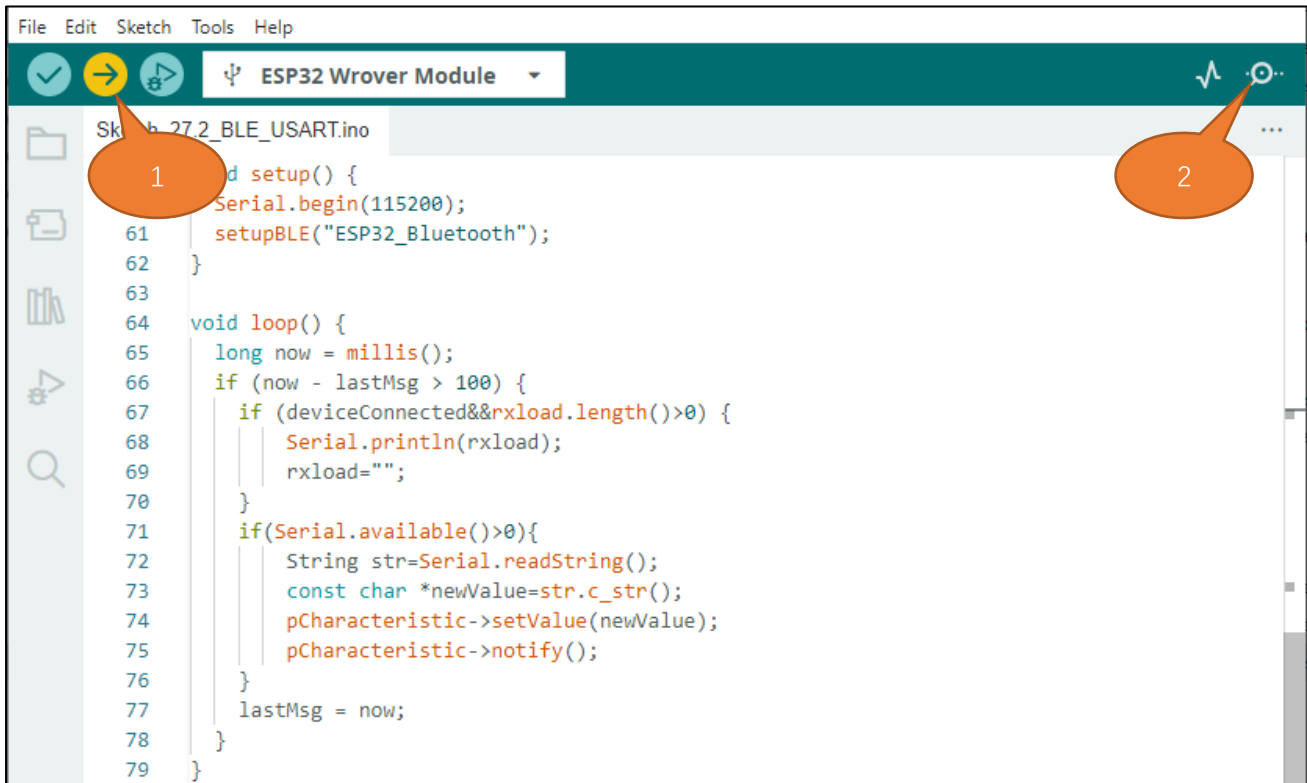
<https://apps.apple.com/us/app/lightblue/id557428110/?platform=iphone>.



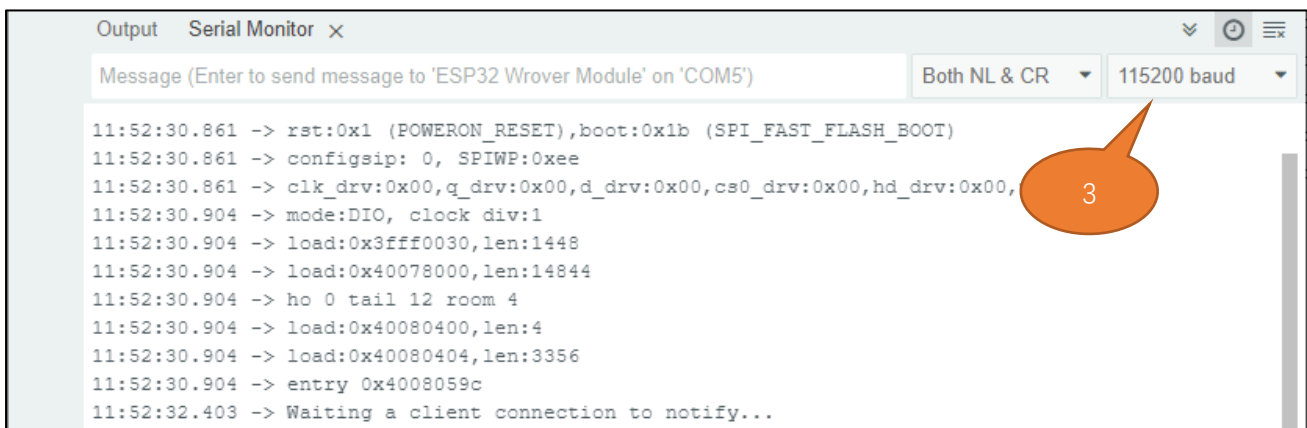
Any concerns? [✉ support@freenove.com](mailto:support@freenove.com)

Step1. Upload the code of Project27.2 to ESP32.

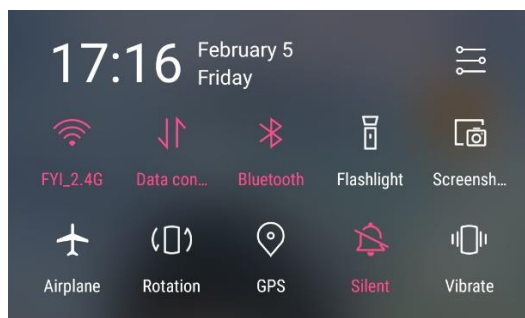
Step2. Click on serial monitor.



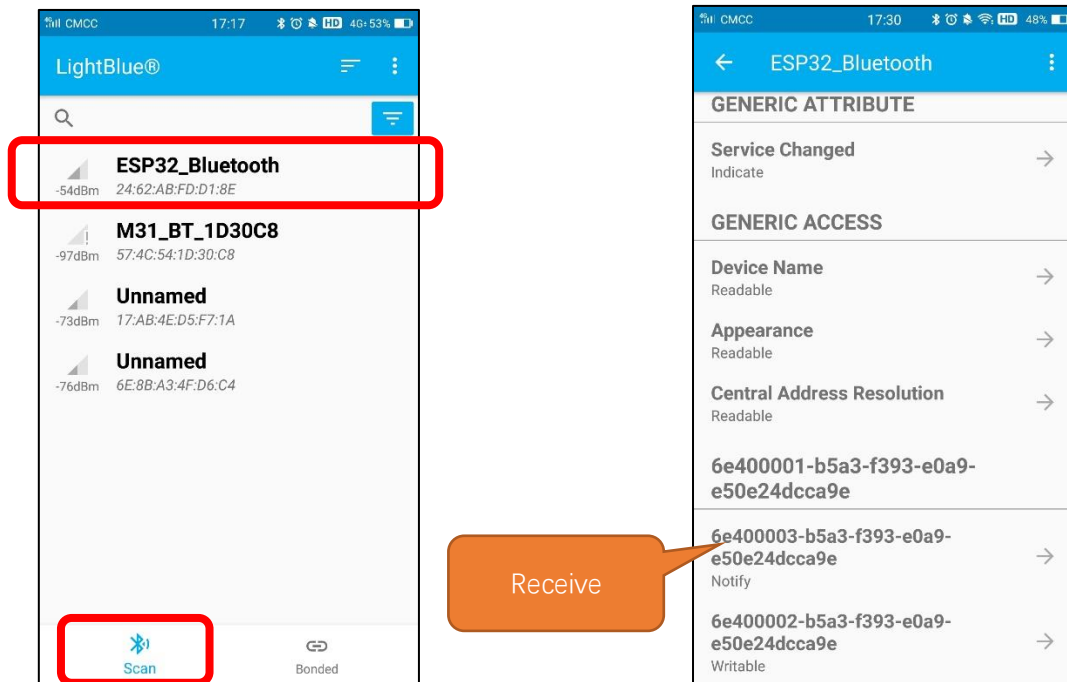
Step3. Set baud rate to 115200.



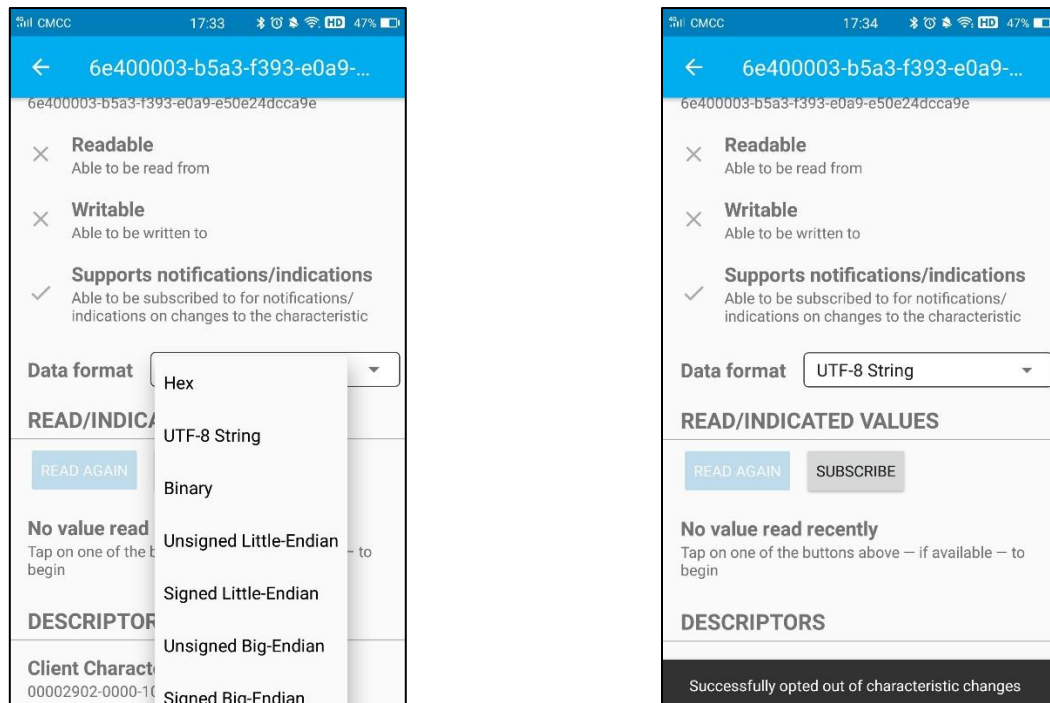
Turn ON Bluetooth on your phone, and open the Lightblue APP.



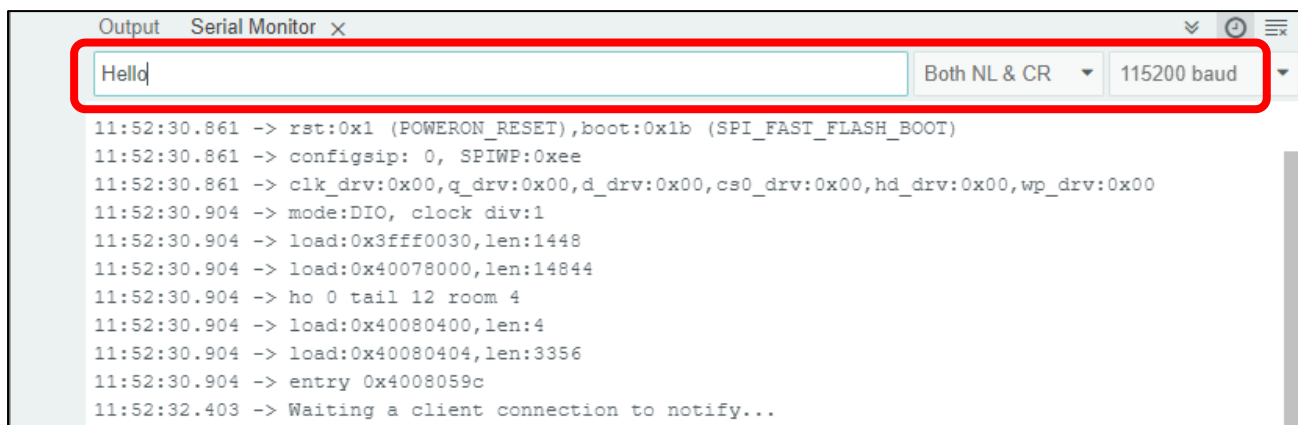
In the Scan page, swipe down to refresh the name of Bluetooth that the phone searches for. Click ESP32\_Bluetooth.



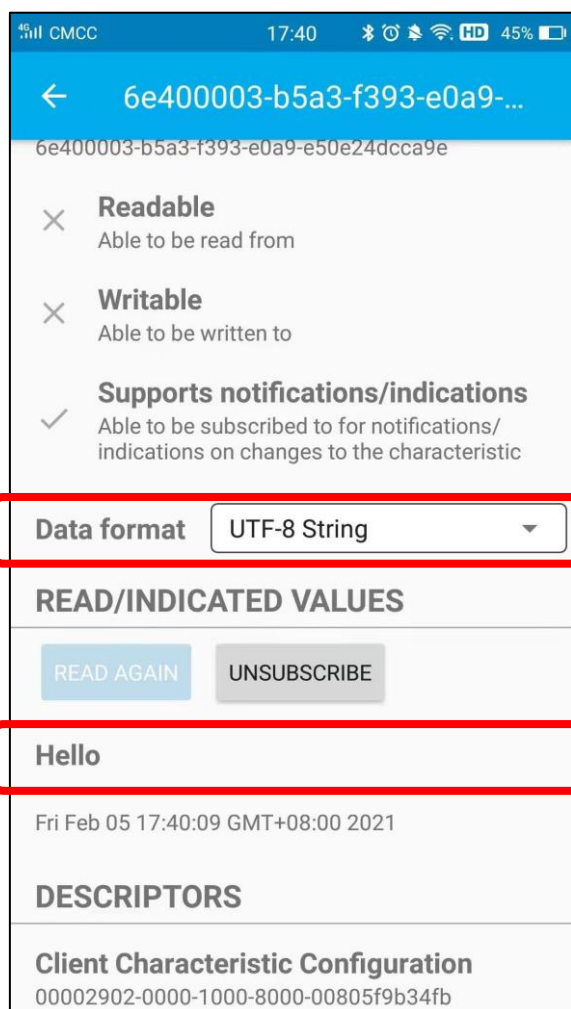
Click "Receive". Select the appropriate Data format in the box to the right of Data Format. For example, HEX for hexadecimal, utf-string for character, Binary for Binary, etc. Then click SUBSCRIBE.



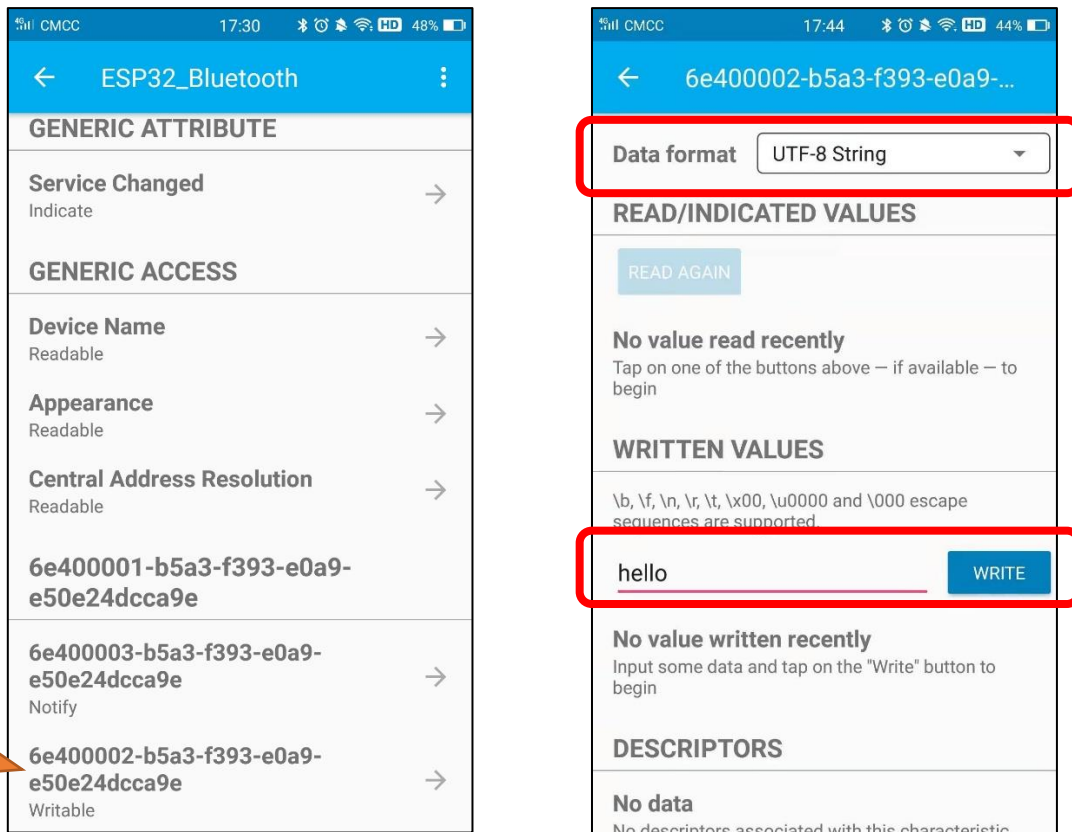
Back to the serial monitor on your computer. You can type anything in the left border of Send, and then click Send.



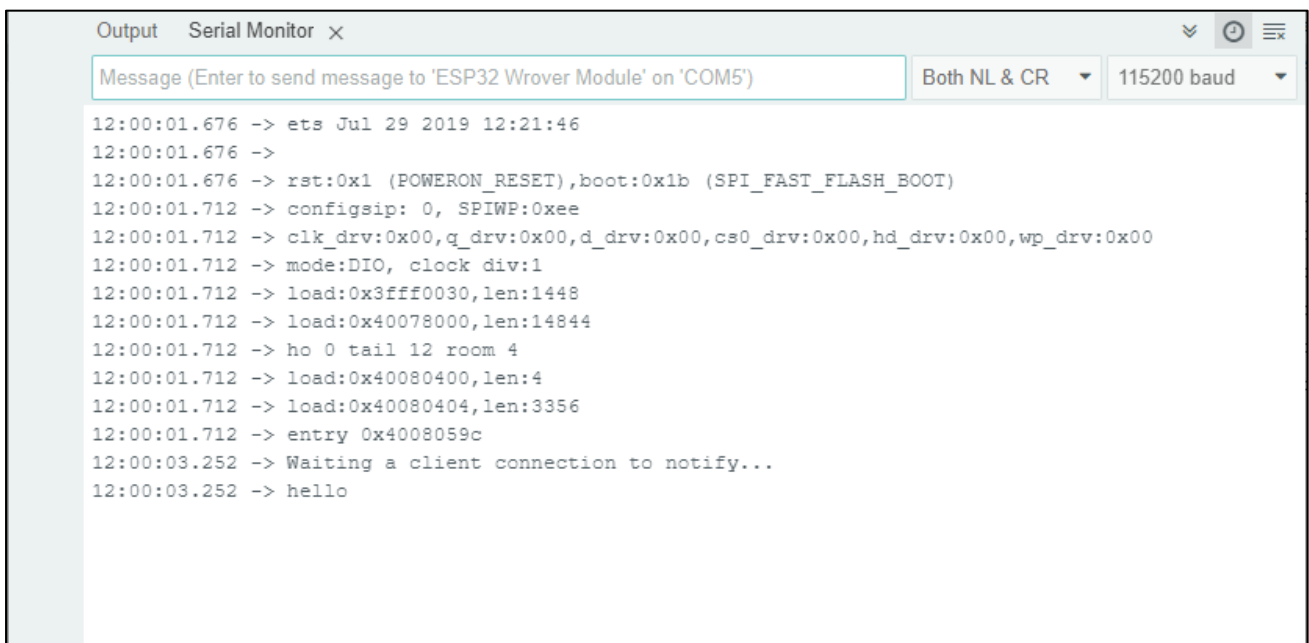
And then you can see the mobile Bluetooth has received the message.



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-WROVER.

The following is the program code:

```

1  #include <BLEDevice.h>
2  #include <BLEServer.h>
3  #include <BLEUtils.h>
4  #include <BLE2902.h>
5  #include <String.h>
6
7  BLECharacteristic *pCharacteristic;
8  bool deviceConnected = false;
9  uint8_t txValue = 0;
10 long lastMsg = 0;
11 String rxload="Test\n";
12
13 #define SERVICE_UUID           "6E400001-B5A3-F393-E0A9-E50E24DCCA9E"
14 #define CHARACTERISTIC_UUID_RX "6E400002-B5A3-F393-E0A9-E50E24DCCA9E"
15 #define CHARACTERISTIC_UUID_TX "6E400003-B5A3-F393-E0A9-E50E24DCCA9E"
16
17 class MyServerCallbacks: public BLEServerCallbacks {
18     void onConnect(BLEServer* pServer) {
19         deviceConnected = true;
20     };
21     void onDisconnect(BLEServer* pServer) {
22         deviceConnected = false;
23     }
24 };
25
26 class MyCallbacks: public BLECharacteristicCallbacks {
27     void onWrite(BLECharacteristic *pCharacteristic) {
28         std::string rxValue = pCharacteristic->getValue();
29         if (rxValue.length() > 0) {
30             rxload="";
31             for (int i = 0; i < rxValue.length(); i++){
32                 rxload +=(char)rxValue[i];
33             }
34         }
35     }
36 };
37
38 void setupBLE(String BLEName){
39     const char *ble_name=BLEName.c_str();
40     BLEDevice::init(ble_name);
41     BLEServer *pServer = BLEDevice::createServer();
42     pServer->setCallbacks(new MyServerCallbacks());

```

```

43  BLEService *pService = pServer->createService(SERVICE_UUID);
44  pCharacteristic=
pService->createCharacteristic(CHARACTERISTIC_UUID_TX, BLECharacteristic::PROPERTY_NOTIFY);
45  pCharacteristic->addDescriptor(new BLE2902());
46  BLECharacteristic *pCharacteristic =
pService->createCharacteristic(CHARACTERISTIC_UUID_RX, BLECharacteristic::PROPERTY_WRITE);
47  pCharacteristic->setCallbacks(new MyCallbacks());
48  pService->start();
49  pServer->getAdvertising()->start();
50  Serial.println("Waiting a client connection to notify...");
51  }
52
53  void setup() {
54    Serial.begin(9600);
55    setupBLE("ESP32_Bluetooth");
56  }
57
58  void loop() {
59    long now = millis();
60    if (now - lastMsg > 1000) {
61      if (deviceConnected&&rxload.length()>0) {
62        Serial.println(rxload);
63        rxload="";
64      }
65      if(Serial.available()>0){
66        String str=Serial.readString();
67        const char *newValue=str.c_str();
68        pCharacteristic->setValue(newValue);
69        pCharacteristic->notify();
70      }
71      lastMsg = now;
72    }
73  }

```

Define the specified UUID number for BLE vendor.

```

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

```

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

```

17 class MyServerCallbacks: public BLEServerCallbacks {
18     void onConnect(BLEServer* pServer) {
19         deviceConnected = true;
20     };
21     void onDisconnect(BLEServer* pServer) {
22         deviceConnected = false;
23     }
24 };

```

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

```

26 class MyCallbacks: public BLECharacteristicCallbacks {
27     void onWrite(BLECharacteristic *pCharacteristic) {
28         std::string rxValue = pCharacteristic->getValue();
29         if (rxValue.length() > 0) {
30             rxload="";
31             for (int i = 0; i < rxValue.length(); i++) {
32                 rxload +=(char)rxValue[i];
33             }
34         }
35     }
36 };

```

Initialize the BLE function and name it.

```

55 setupBLE("ESP32_Bluetooth");

```

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

```

59 long now = millis();
60 if (now - lastMsg > 1000) {
61     if (deviceConnected&&rxload.length()>0) {
62         Serial.println(rxload);
63         rxload="";
64     }
65     if(Serial.available()>0){
66         String str=Serial.readString();
67         const char *newValue=str.c_str();
68         pCharacteristic->setValue(newValue);
69         pCharacteristic->notify();
70     }
71     lastMsg = now;
72 }

```

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.

```
38 void setupBLE(String BLEName) {
39     const char *ble_name=BLEName.c_str();
40     BLEDevice::init(ble_name);
41     BLEServer *pServer = BLEDevice::createServer();
42     pServer->setCallbacks(new MyServerCallbacks());
43     BLEService *pService = pServer->createService(SERVICE_UUID);
44     pCharacteristic=
pService->createCharacteristic(CHARACTERISTIC_UUID_TX,BLECharacteristic::PROPERTY_NOTIFY);
45     pCharacteristic->addDescriptor(new BLE2902());
46     BLECharacteristic *pCharacteristic =
pService->createCharacteristic(CHARACTERISTIC_UUID_RX,BLECharacteristic::PROPERTY_WRITE);
47     pCharacteristic->setCallbacks(new MyCallbacks());
48     pService->start();
49     pServer->getAdvertising()->start();
50     Serial.println("Waiting a client connection to notify...");
51 }
```

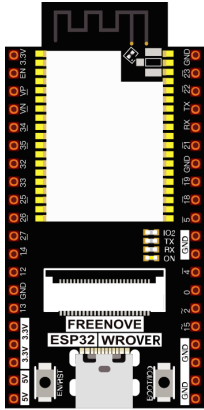
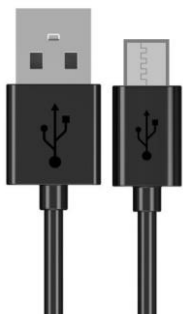


## Chapter 3 Read and Write the Sdcard

Note: The SD card chapter only applies to the ESP32 WROVER development board with an SD card slot on the back. If your ESP32 WROVER does not have an SD card slot on the back, please skip this chapter.

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

### Project 3.1 SDMMC Test

#### Component List

ESP32 WROVER x1 	USB cable x1 	SDcard reader x1 (random color)  (Not a USB flash drive.)	SDcard x1 
---	--	--	---

#### Component knowledge

##### SD card read and write method

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

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

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

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

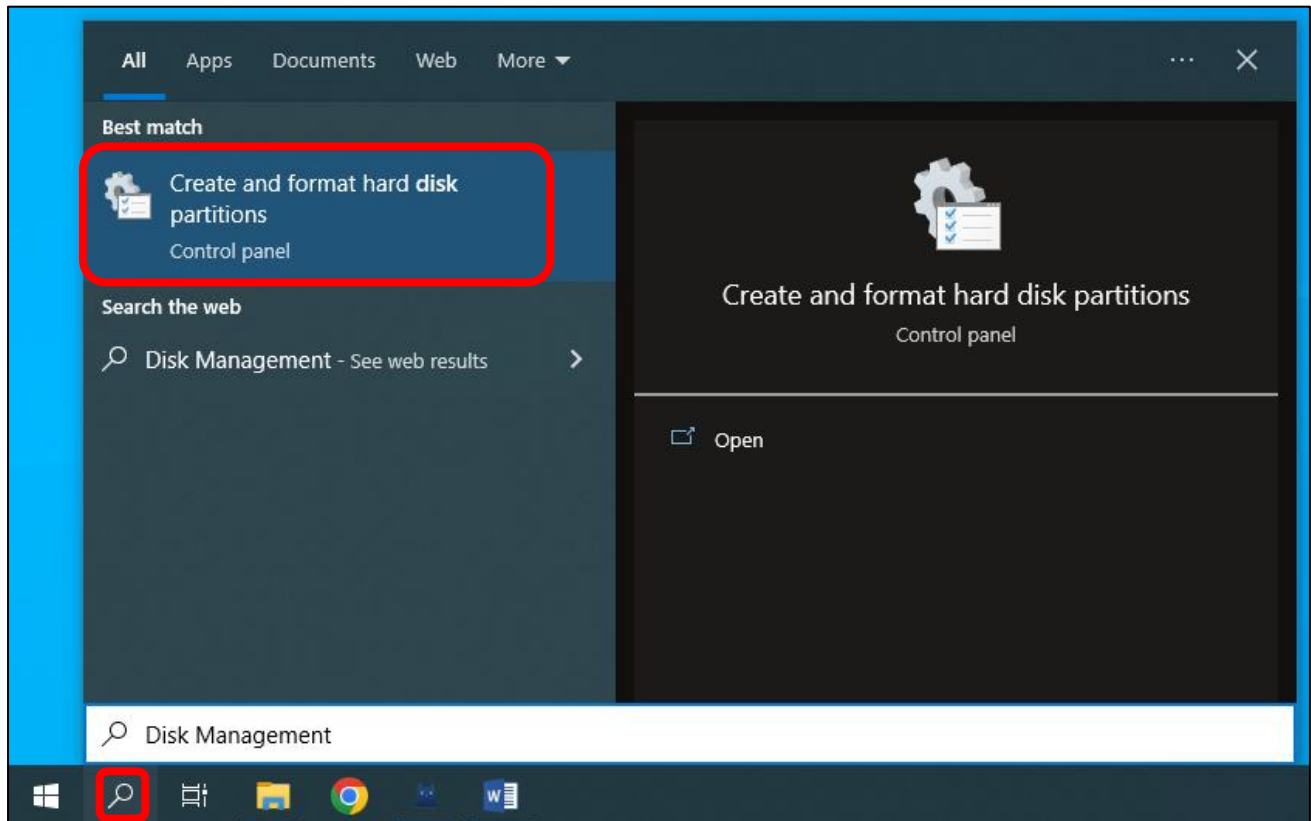
## Format SD card

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

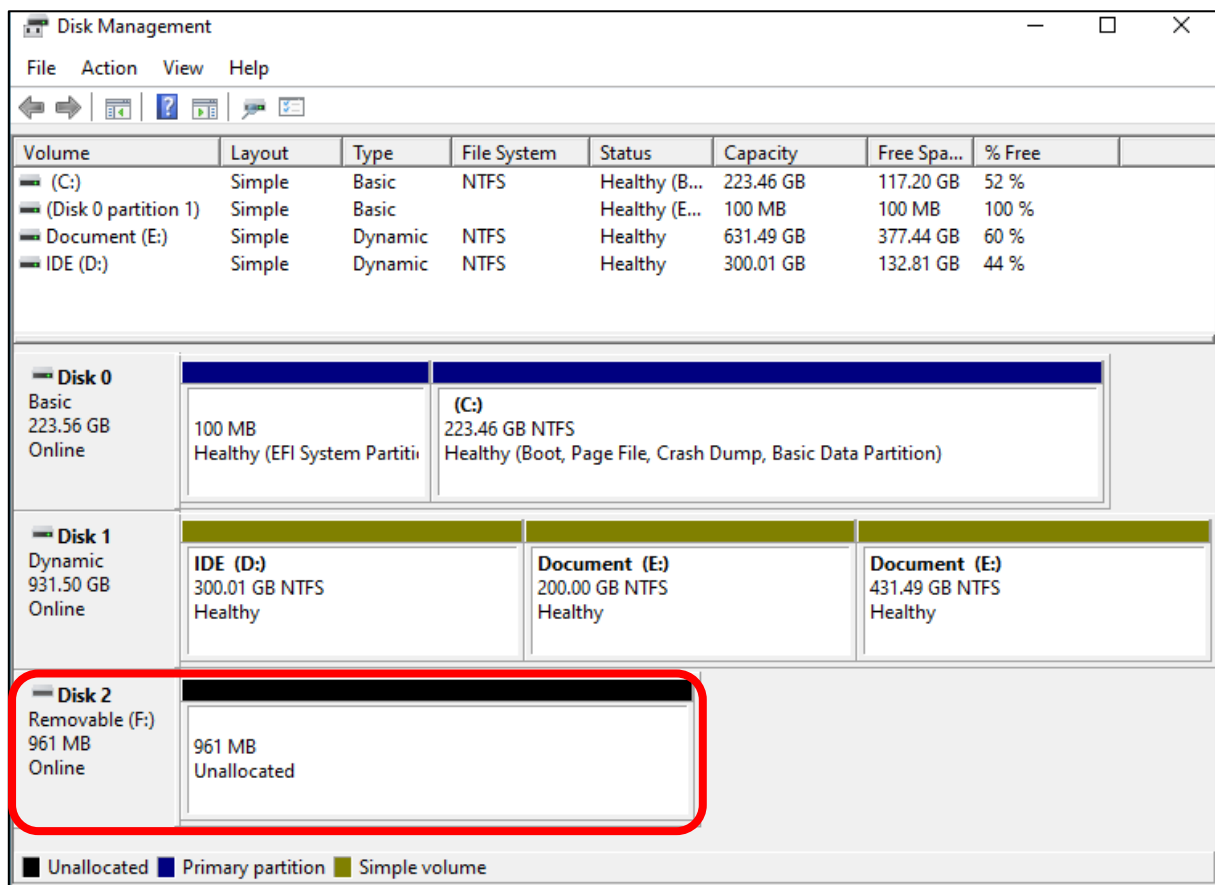
### Windows

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

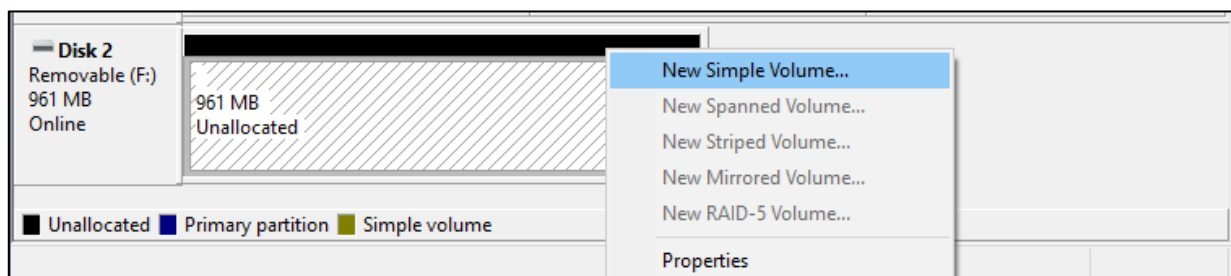
In the Windows search box, enter "Disk Management" and select "Create and format hard disk partitions".



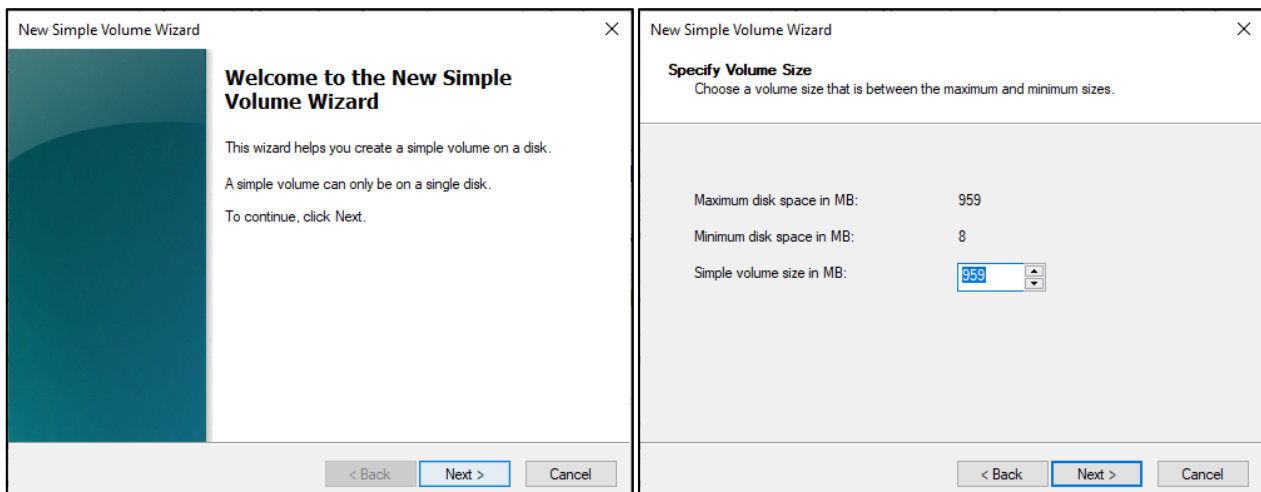
In the new pop-up window, find an unallocated volume close to 1G in size.



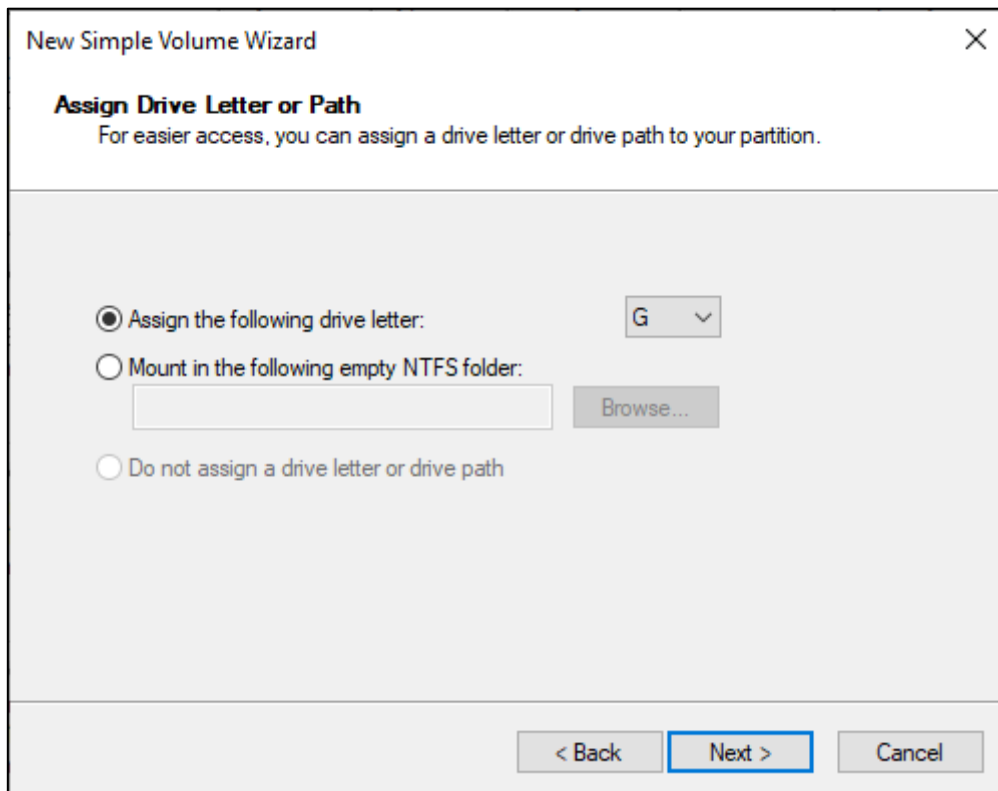
Click to select the volume, right-click and select "New Simple Volume".



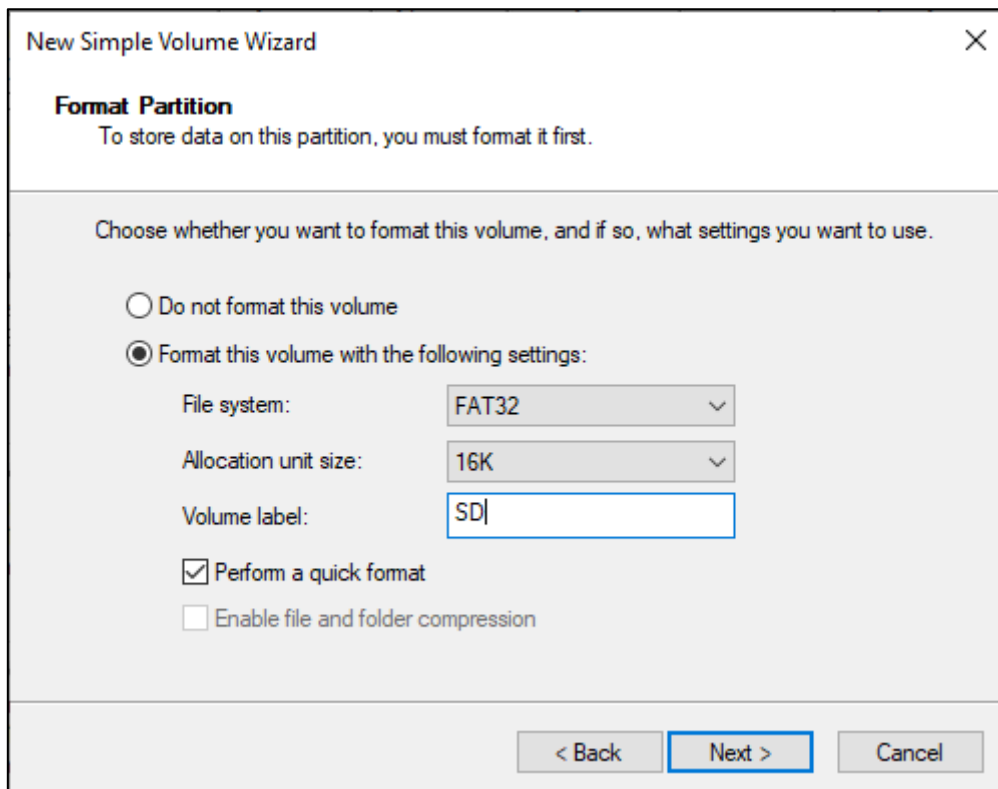
Click Next.



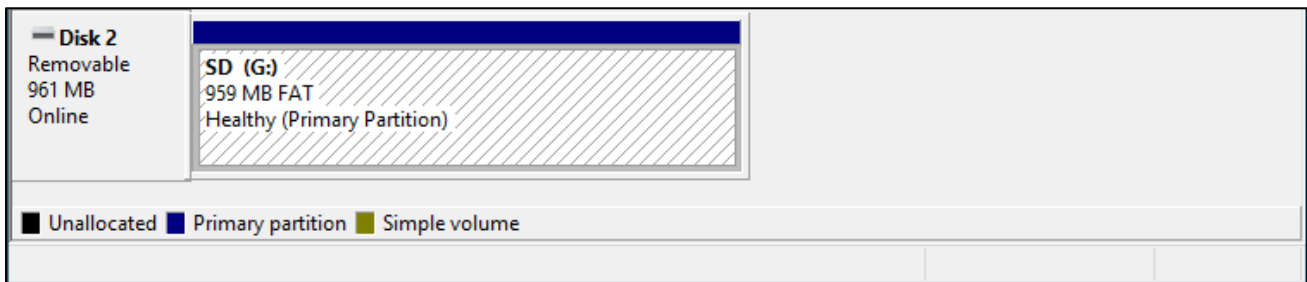
You can choose the drive letter on the right, or you can choose the default. By default, just click Next.



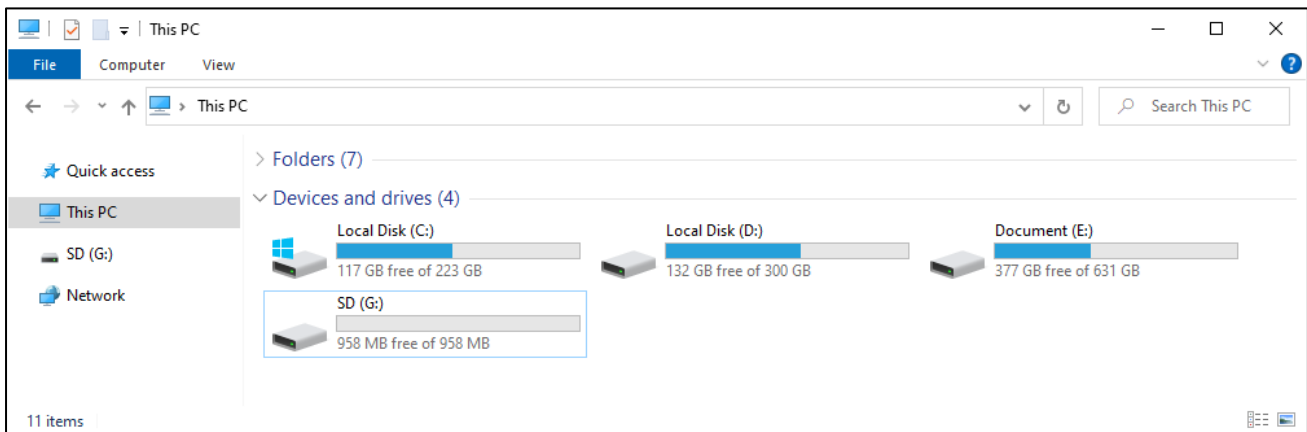
File system is FAT(or FAT32). The Allocation unit size is 16K, and the Volume label can be set to any name. After setting, click Next.



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



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

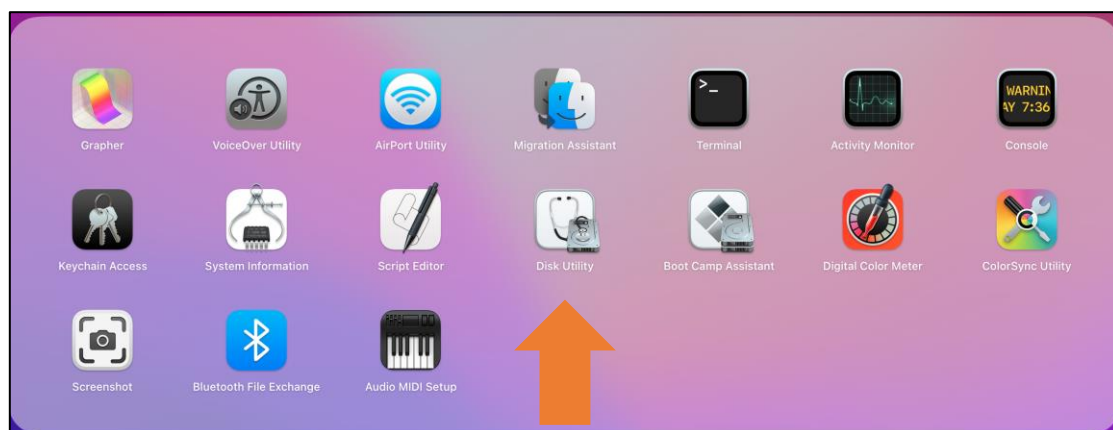


## MAC

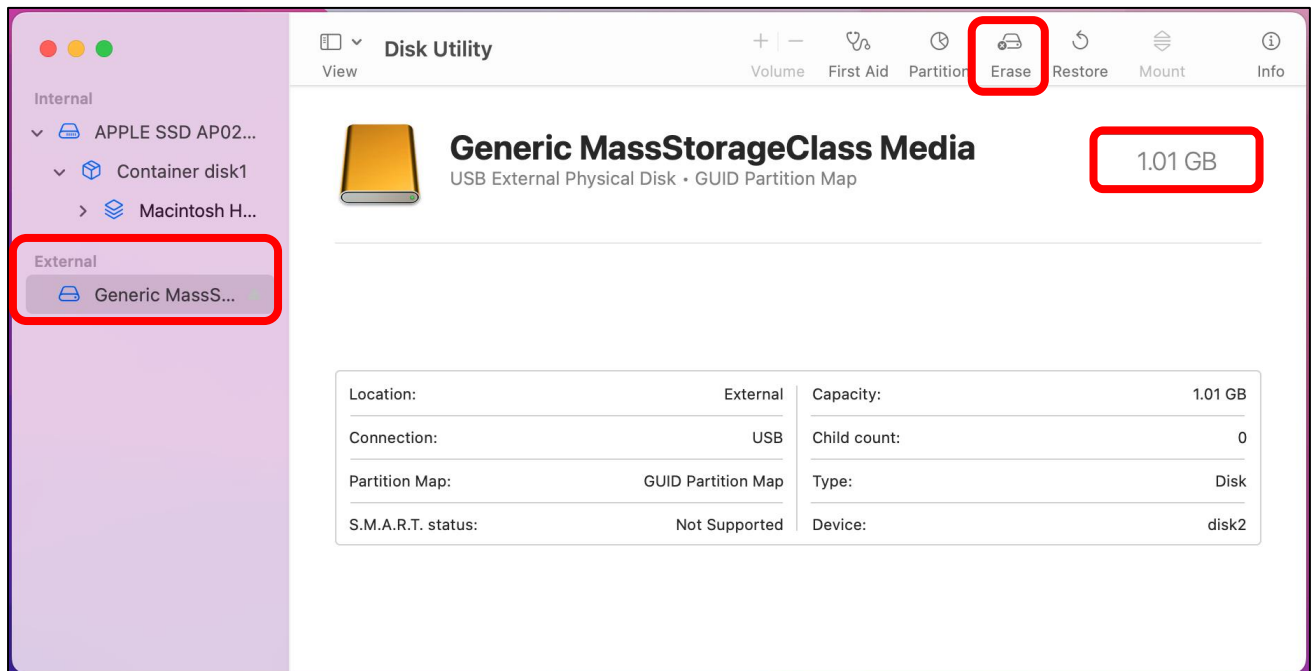
Insert the SD card into the card reader, then insert the card reader into the computer. Some computers will prompt the following information, please click to ignore it.



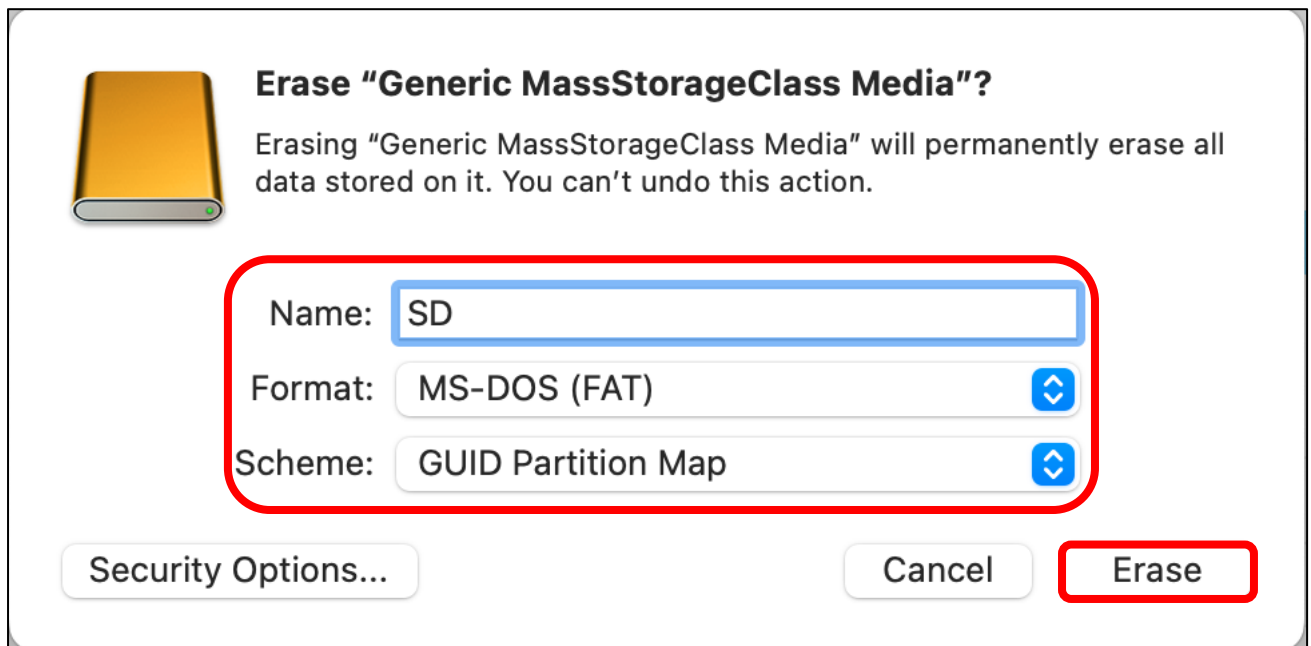
Find "Disk Utility" in the MAC system and click to open it.



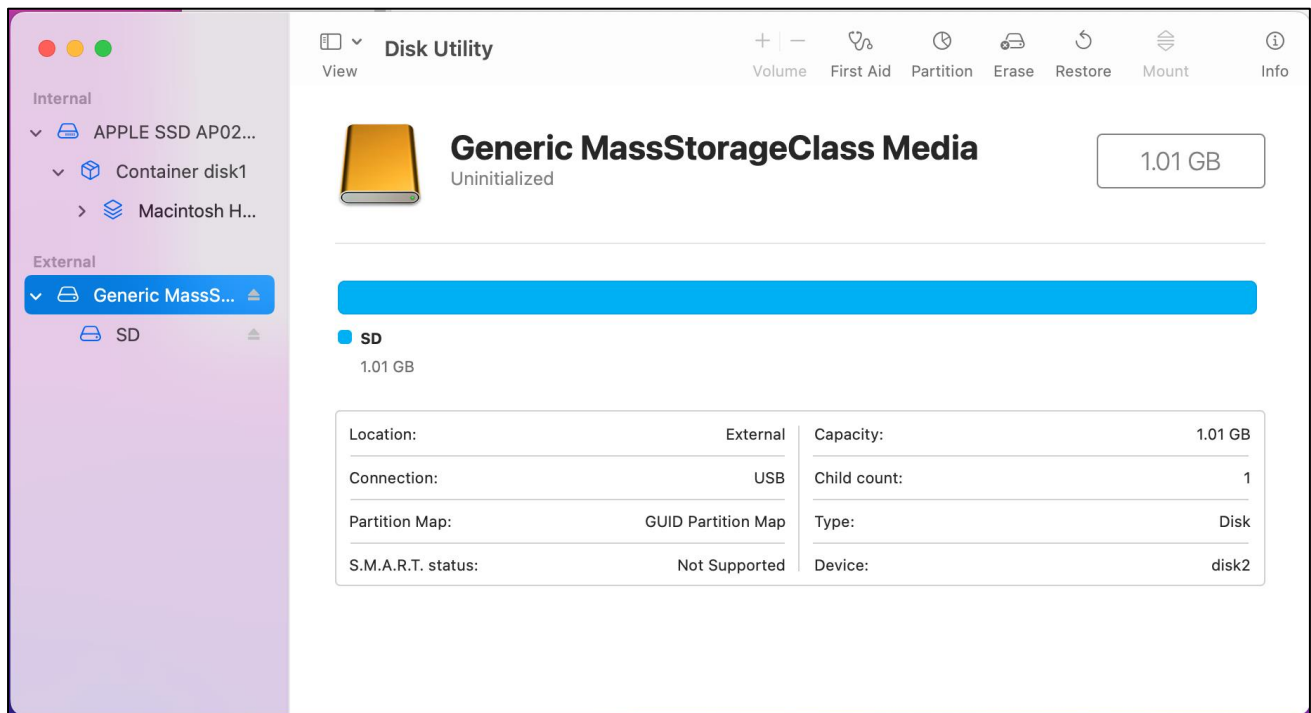
Select "Generic MassStorageClass Media", note that its size is about 1G. Please do not choose wrong item. Click "Erase".



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

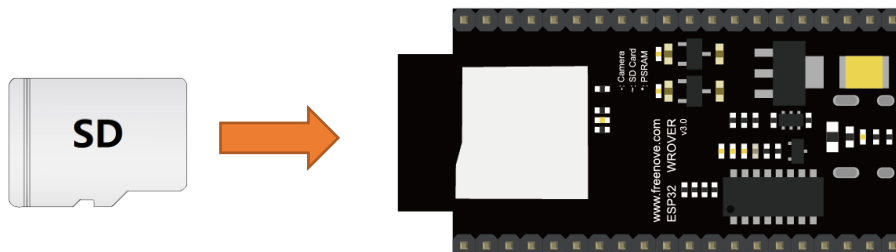


Wait for the formatting to complete. When finished, it will look like the picture below. At this point, you can see a new disk on the desktop named "SD".

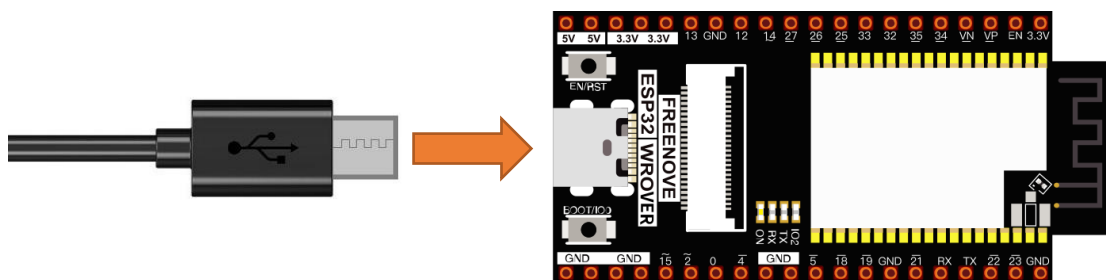


## Circuit

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

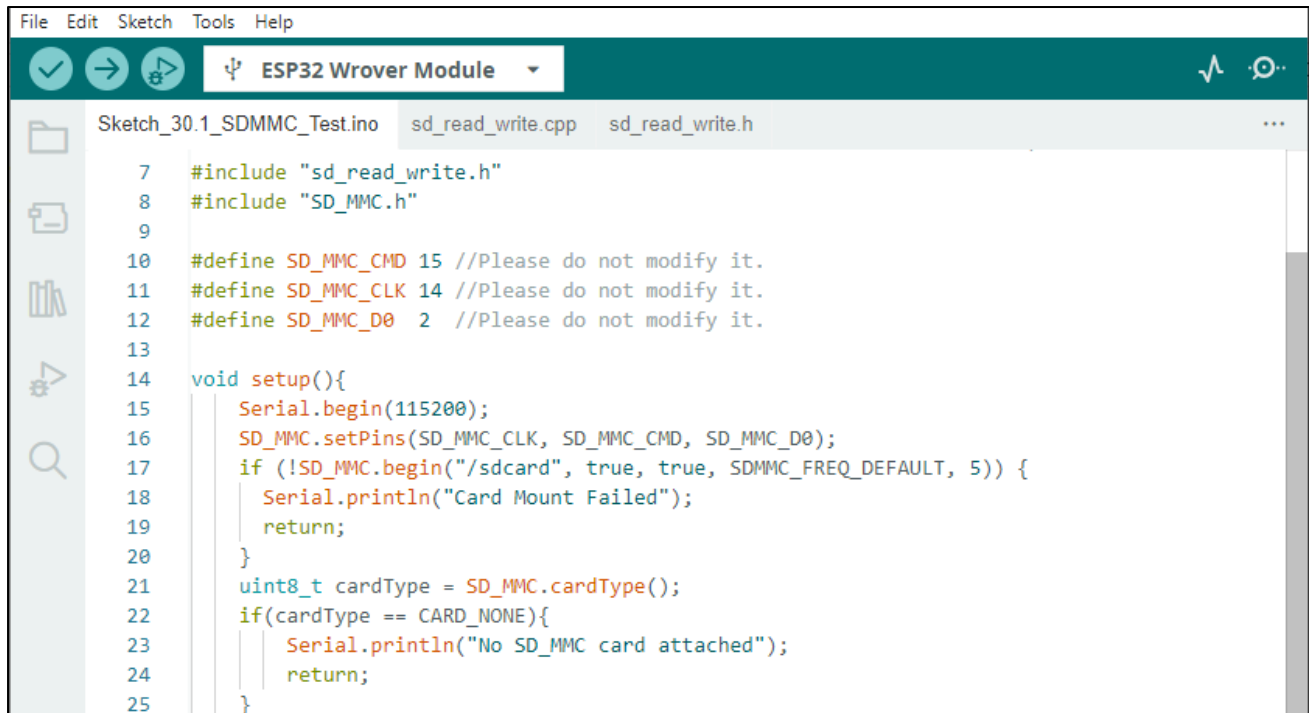


Connect Freenove ESP32 to the computer using the USB cable.



## Sketch

### Sketch\_03.1\_SDMMC\_Test



```
File Edit Sketch Tools Help
ESP32 Wrover Module

Sketch_03.1_SDMMC_Test.ino sd_read_write.cpp sd_read_write.h

7  #include "sd_read_write.h"
8  #include "SD_MMC.h"
9
10 #define SD_MMC_CMD 15 //Please do not modify it.
11 #define SD_MMC_CLK 14 //Please do not modify it.
12 #define SD_MMC_D0 2 //Please do not modify it.
13
14 void setup(){
15     Serial.begin(115200);
16     SD_MMC.setPins(SD_MMC_CLK, SD_MMC_CMD, SD_MMC_D0);
17     if (!SD_MMC.begin("/sdcard", true, true, SDMMC_FREQ_DEFAULT, 5)) {
18         Serial.println("Card Mount Failed");
19         return;
20     }
21     uint8_t cardType = SD_MMC.cardType();
22     if(cardType == CARD_NONE){
23         Serial.println("No SD_MMC card attached");
24         return;
25     }
}
```

Compile and upload the code to ESP32, open the serial monitor, and press the RST button on the board. You can see the printout as shown below.

```
14:47:21.289 -> Listing directory: /System Volume Information
14:47:21.289 -> FILE: WPSettings.dat SIZE: 12
14:47:21.289 -> FILE: IndexerVolumeGuid SIZE: 76
14:47:21.289 -> FILE: test.txt SIZE: 1048576
14:47:21.289 -> FILE: foo.txt SIZE: 13
14:47:21.289 -> DIR : music
14:47:21.289 -> Listing directory: /music
14:47:21.289 -> FILE: 01.mp3 SIZE: 3528199
14:47:21.289 -> FILE: Good Time.mp3 SIZE: 3291708
14:47:21.289 -> FILE: Jingle Bells.mp3 SIZE: 8004409
14:47:21.289 -> Writing file: /hello.txt
14:47:21.289 -> File written
14:47:21.416 -> Appending to file: /hello.txt
14:47:21.416 -> Message appended
14:47:21.461 -> Reading file: /hello.txt
14:47:21.461 -> Read from file: Hello World!
14:47:21.461 -> Deleting file: /foo.txt
14:47:21.502 -> File deleted
14:47:21.502 -> Renaming file /hello.txt to /foo.txt
14:47:21.535 -> File renamed
14:47:21.535 -> Reading file: /foo.txt
14:47:21.535 -> Read from file: Hello World!
14:47:22.054 -> 1048576 bytes read for 547 ms
14:47:22.758 -> 1048576 bytes written for 687 ms
14:47:22.804 -> Total space: 953MB
14:47:22.804 -> Used space: 15MB
```

The following is the program code:

```

1  #include "sd_read_write.h"
2  #include "SD_MMC.h"
3
4  #define SD_MMC_CMD 15 //Please do not modify it.
5  #define SD_MMC_CLK 14 //Please do not modify it.
6  #define SD_MMC_DO 2 //Please do not modify it.
7
8  void setup() {
9      Serial.begin(115200);
10     SD_MMC.setPins(SD_MMC_CLK, SD_MMC_CMD, SD_MMC_DO);
11     if (!SD_MMC.begin("/sdcard", true, true, SDMMC_FREQ_DEFAULT, 5)) {
12         Serial.println("Card Mount Failed");
13         return;
14     }
15     uint8_t cardType = SD_MMC.cardType();
16     if(cardType == CARD_NONE) {
17         Serial.println("No SD_MMC card attached");
18         return;
19     }
20     Serial.print("SD_MMC Card Type: ");
21     if(cardType == CARD_MMC) {
22         Serial.println("MMC");
23     } else if(cardType == CARD_SD) {
24         Serial.println("SDSC");
25     } else if(cardType == CARD_SDHC) {
26         Serial.println("SDHC");
27     } else {
28         Serial.println("UNKNOWN");
29     }
30
31     uint64_t cardSize = SD_MMC.cardSize() / (1024 * 1024);
32     Serial.printf("SD_MMC Card Size: %lluMB\n", cardSize);
33
34     listDir(SD_MMC, "/", 0);
35
36     createDir(SD_MMC, "/mydir");
37     listDir(SD_MMC, "/", 0);
38
39     removeDir(SD_MMC, "/mydir");
40     listDir(SD_MMC, "/", 2);
41
42     writeFile(SD_MMC, "/hello.txt", "Hello ");
43     appendFile(SD_MMC, "/hello.txt", "World!\n");

```

```

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

```

Add the SD card drive header file.

```

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

```

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

```

4  #define SD_MMC_CMD 15 //Please do not modify it.
5  #define SD_MMC_CLK 14 //Please do not modify it.
6  #define SD_MMC_D0 2 //Please do not modify it.

```

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

```

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

```

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

```

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

```

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

```

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

```

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

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

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

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

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

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

Call `appendFile()` to append any content to txt.

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

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

Call `deleteFile()` to delete a specified file.

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

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

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

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

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

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

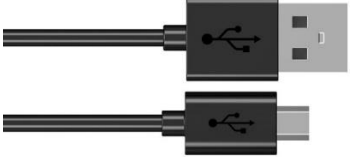
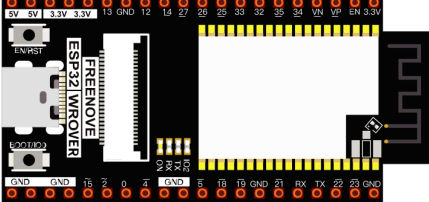
## Chapter 4 WiFi Working Modes

In this chapter, we'll focus on the WiFi infrastructure for ESP32-WROVER.

ESP32-WROVER 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 4.1 Station mode

#### Component List

Micro USB Wire x1	ESP32-WROVER x1
	

#### Component knowledge

##### Station mode

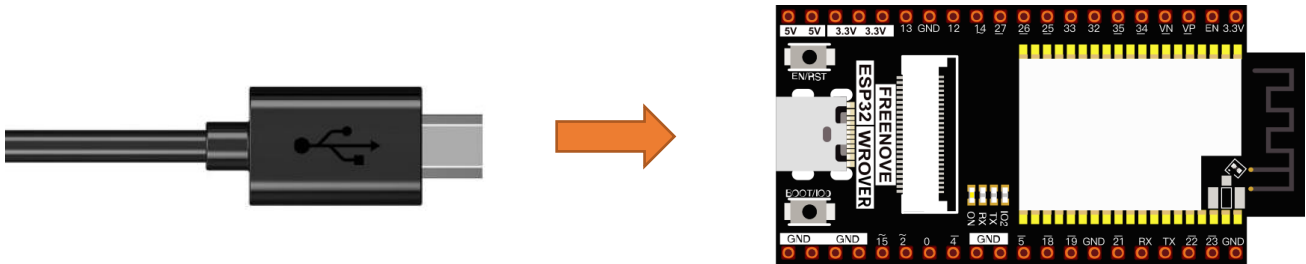
When ESP32 selects Station mode, it acts as a WiFi client. It can connect to the router network and communicate with other devices on the router via WiFi connection. As shown below, the PC is connected to the router, and if ESP32 wants to communicate with the PC, it needs to be connected to the router.



Any concerns? [✉ support@freenove.com](mailto:support@freenove.com)

## Circuit

Connect Freenove ESP32 to the computer using the USB cable.



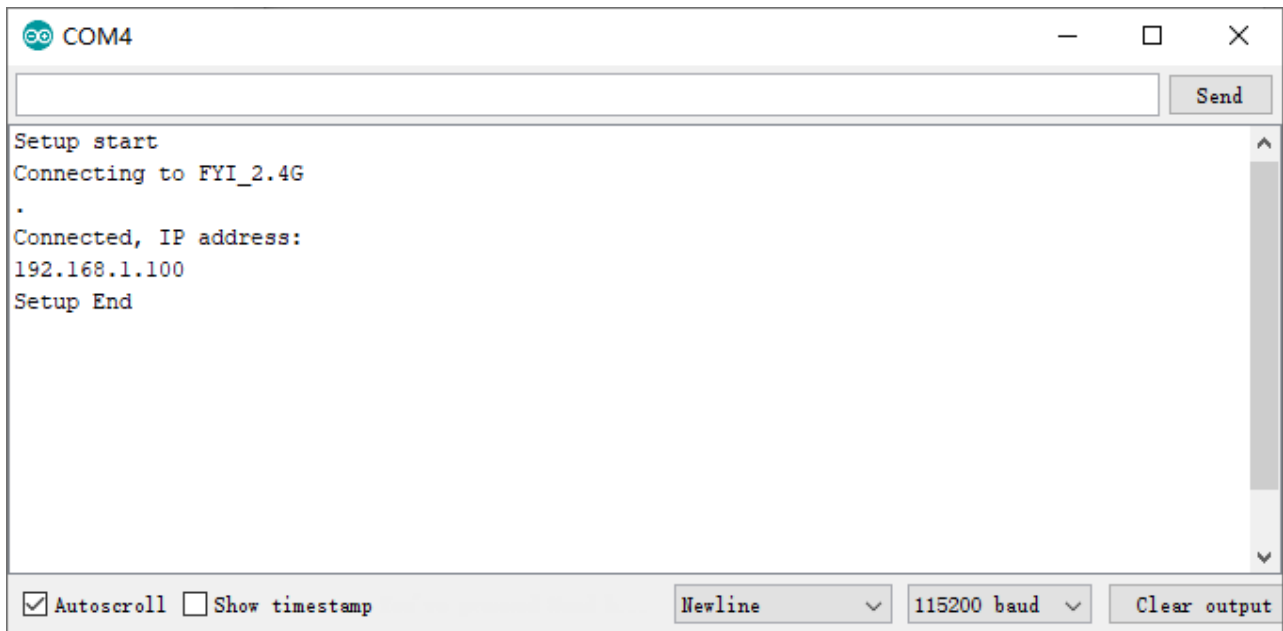
## Sketch

### Sketch\_04.1\_Station\_mode



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

After making sure the router name and password are entered correctly, compile and upload codes to ESP32-WROVER, open serial monitor and set baud rate to 115200. And then it will display as follows:



When ESP32-WROVER successfully connects to "ssid\_Router", serial monitor will print out the IP address assigned to ESP32-WROVER by the router.

The following is the program code:

```

1  #include <WiFi.h>
2
3  const char *ssid_Router    = "*****"; //Enter the router name
4  const char *password_Router = "*****"; //Enter the router password
5
6  void setup() {
7      Serial.begin(115200);
8      delay(2000);
9      Serial.println("Setup start");
10     WiFi.begin(ssid_Router, password_Router);
11     Serial.println(String("Connecting to ") + ssid_Router);
12     while (WiFi.status() != WL_CONNECTED) {
13         delay(500);
14         Serial.print(".");
15     }
16     Serial.println("\nConnected, IP address: ");
17     Serial.println(WiFi.localIP());
18     Serial.println("Setup End");
19 }
20
21 void loop() {
22 }

```

Include the WiFi Library header file of ESP32.

```

1  #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 in Station mode and connect it to your router.

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

Check whether ESP32 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-WROVER

```
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 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:** boolean optional parameter, defaulting to true. If set as false, then ESP32 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():** obtain IP address in Station mode

**disconnect():** disconnect wifi

**setAutoConnect(boolean):** set automatic connection Every time ESP32 is power on, it will connect WiFi automatically.

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

## Project 4.2 AP mode

### Component List & Circuit

Component List & Circuit are the same as in Section 30.1.

### Component knowledge

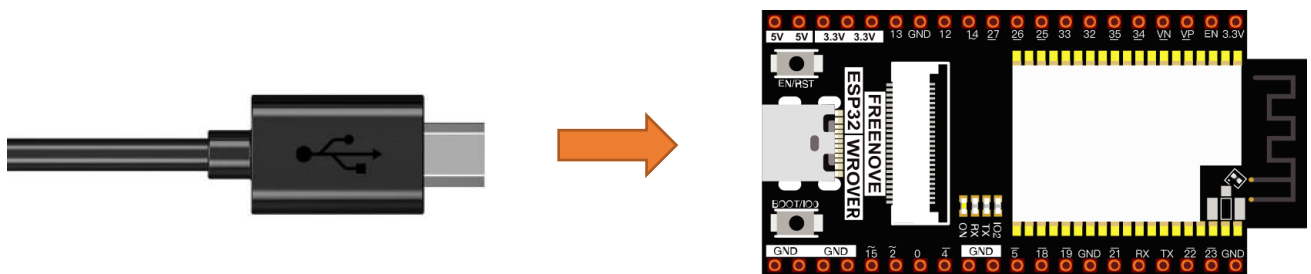
#### AP mode

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



### Circuit

Connect Freenove ESP32 to the computer using the USB cable.



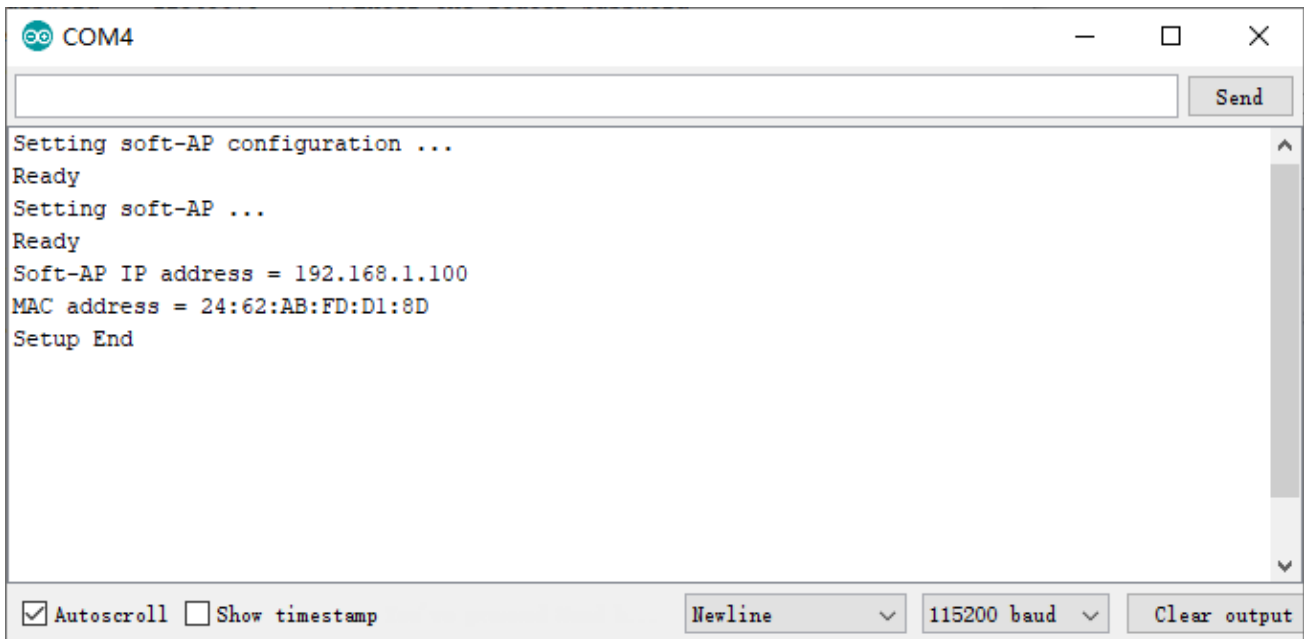
Any concerns? [✉ support@freenove.com](mailto:support@freenove.com)

## Sketch



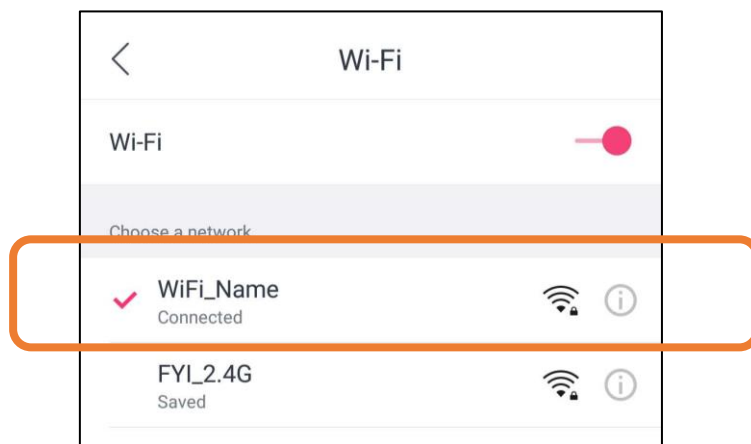
Before the Sketch runs, you can make any changes to the AP name and password for ESP32 in the box as shown in the illustration above. Of course, you can leave it alone by default.

Compile and upload codes to ESP32-WROVER, open the serial monitor and set the baud rate to 115200. And then it will display as follows.



```
COM4
Setting soft-AP configuration ...
Ready
Setting soft-AP ...
Ready
Soft-AP IP address = 192.168.1.100
MAC address = 24:62:AB:FD:D1:8D
Setup End
```

When observing the print information of the serial monitor, turn on the WiFi scanning function of your phone, and you can see the ssid\_AP on ESP32, which is called "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\_04.2\_AP\_mode

The following is the program code:

```

1  #include <WiFi.h>
2
3  const char *ssid_AP    = "WiFi_Name"; //Enter the router name
4  const char *password_AP = "12345678"; //Enter the router password
5
6  IPAddress local_IP(192, 168, 1, 100); //Set the IP address of ESP32 itself
7  IPAddress gateway(192, 168, 1, 10);   //Set the gateway of ESP32 itself
8  IPAddress subnet(255, 255, 255, 0);   //Set the subnet mask for ESP32 itself
9
10 void setup() {
11     Serial.begin(115200);
12     delay(2000);
13     Serial.println("Setting soft-AP configuration ... ");
14     WiFi.disconnect();
15     WiFi.mode(WIFI_AP);
16     Serial.println(WiFi.softAPConfig(local_IP, gateway, subnet) ? "Ready" : "Failed!");
17     Serial.println("Setting soft-AP ... ");
18     boolean result = WiFi.softAP(ssid_AP, password_AP);
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");
27 }
28
29 void loop() {
30 }

```

Include WiFi Library header file of ESP32.

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

Enter correct AP name and password.

```

3  const char *ssid_AP    = "WiFi_Name"; //Enter the router name
4  const char *password_AP = "12345678"; //Enter the router password

```

Set ESP32 in AP mode.

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

Configure IP address, gateway and subnet mask for ESP32.

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

Turn on an AP in ESP32, whose name is set by ssid\_AP and password is set by password\_AP.

```
18  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. 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


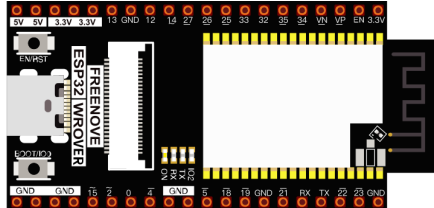
**subnet:** subnet mask

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

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

## Project 3.3 AP+Station mode

### Component List

<p>Micro USB Wire x1</p> 	<p>ESP32-WROVER x1</p> 
--	---

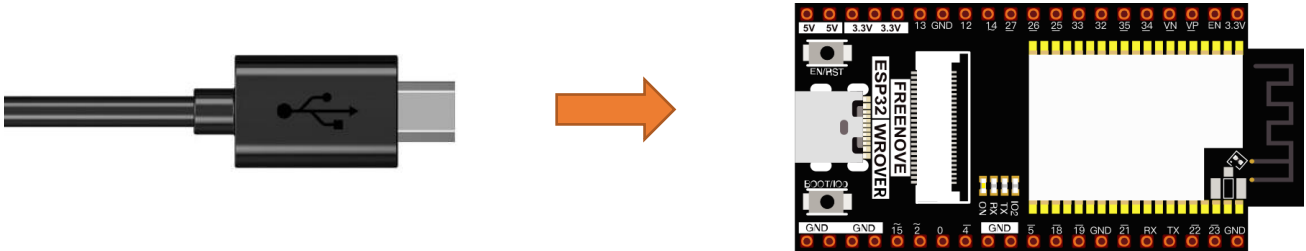
### Component knowledge

#### AP+Station mode

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

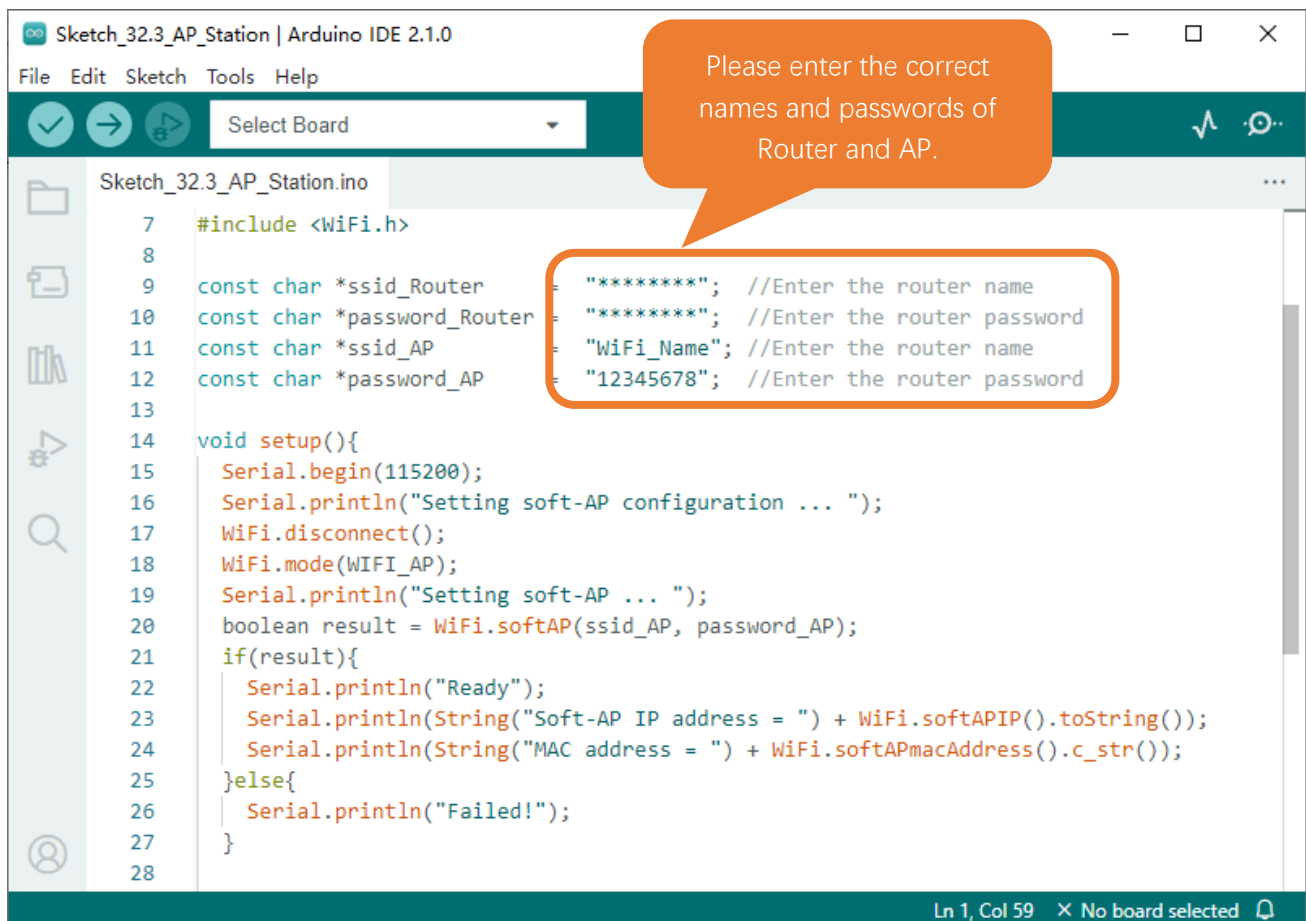
## Circuit

Connect Freenove ESP32 to the computer using the USB cable.



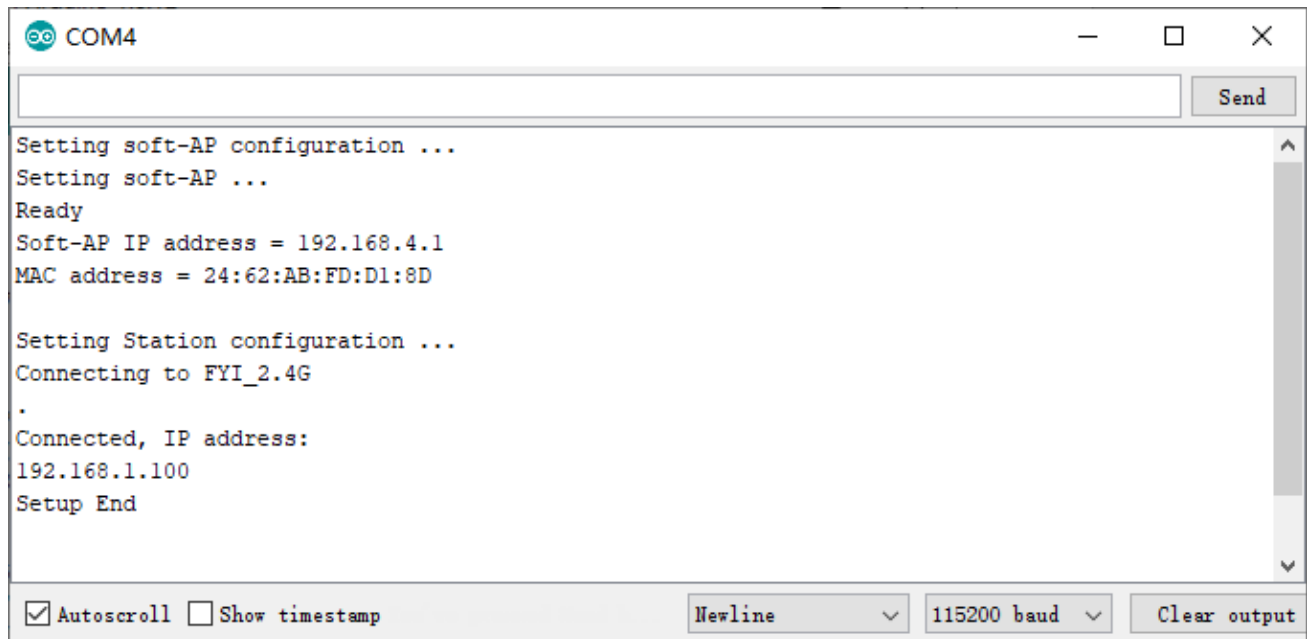
## Sketch

Sketch\_04.3\_AP\_Station\_mode

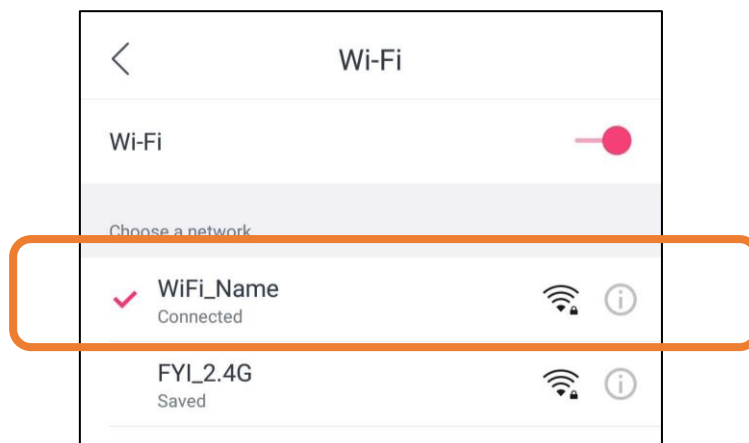


It is analogous to Project 4.1 and Project 4.2. Before running the Sketch, you need to modify ssid\_Router, password\_Router, ssid\_AP and password\_AP shown in the box of the illustration above.

After making sure that Sketch is modified correctly, compile and upload codes to ESP32-WROVER, 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.



The following is the program code:

```

1  #include <WiFi.h>
2
3  const char *ssid_Router    = "*****"; //Enter the router name
4  const char *password_Router = "*****"; //Enter the router password
5  const char *ssid_AP       = "WiFi_Name"; //Enter the AP name
6  const char *password_AP   = "12345678"; //Enter the AP password
7
8  void setup() {
9      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 }
```

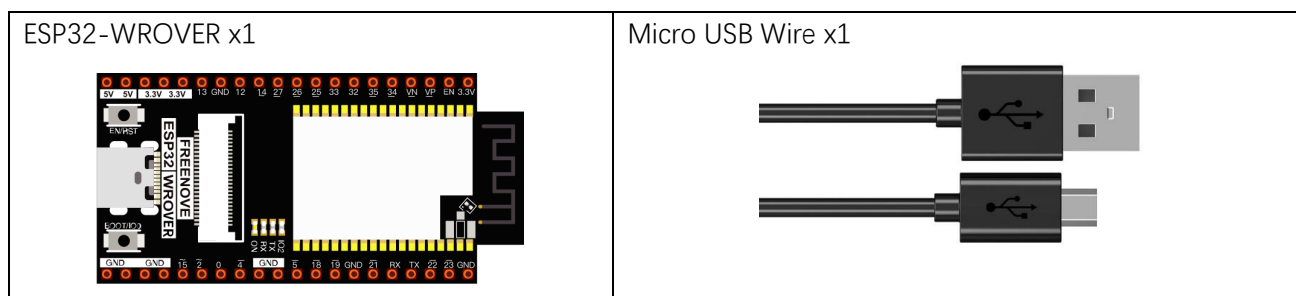
## Chapter 5 TCP/IP

In this chapter, we will introduce how ESP32 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 5.1 As Client

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

### Component List



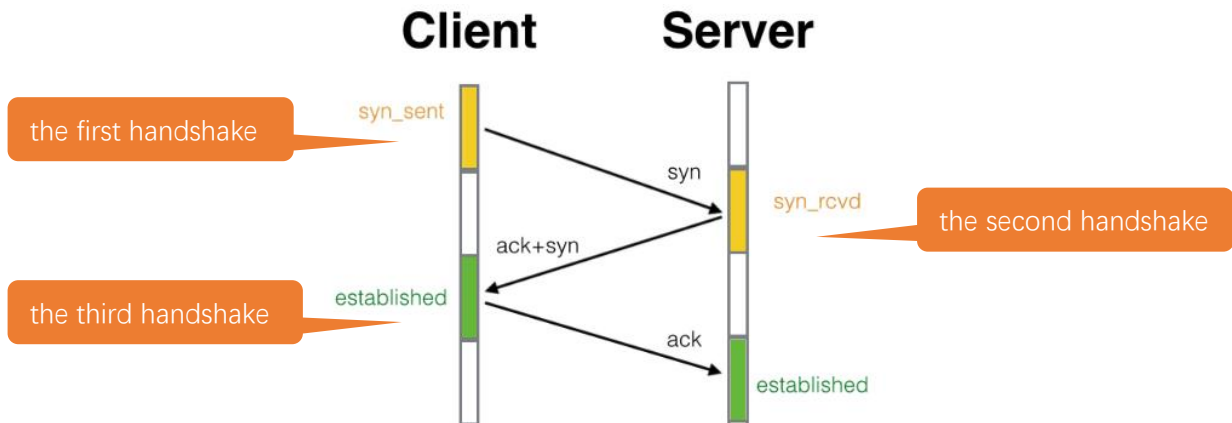
### Component knowledge

#### TCP connection

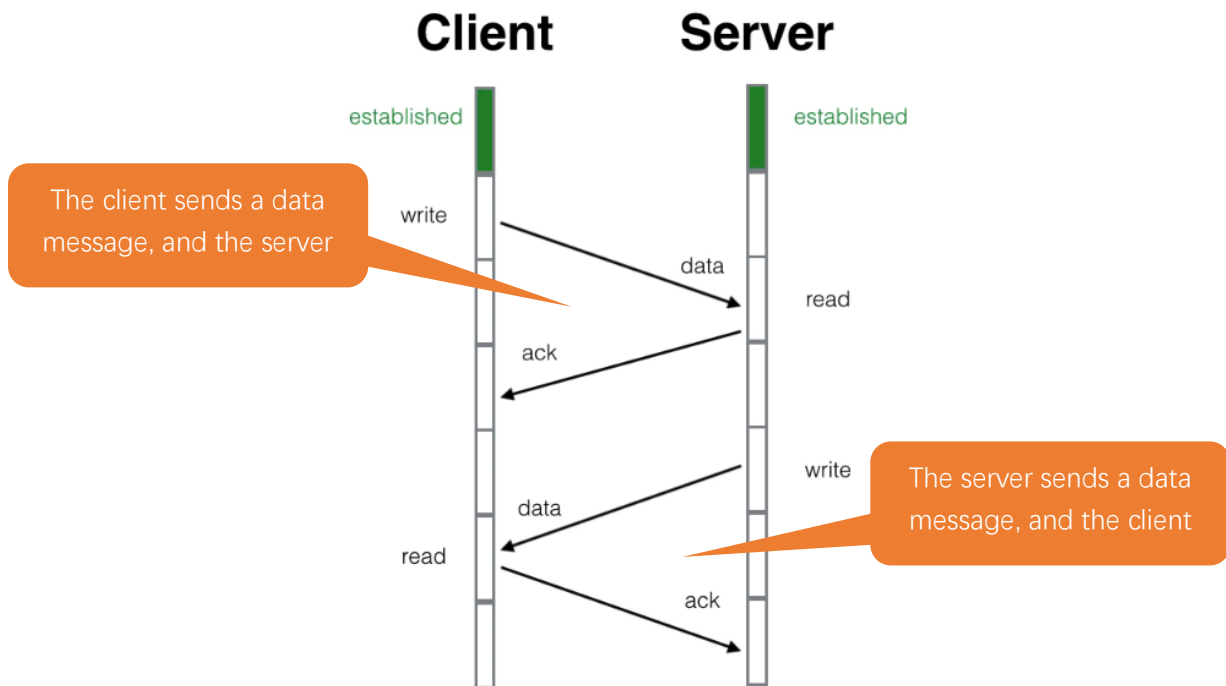
Before transmitting data, TCP needs to establish a logical connection between the sending end and the receiving end. It provides reliable and error-free data transmission between the two computers. In the TCP connection, the client and the server must be clarified. The client sends a connection request 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.

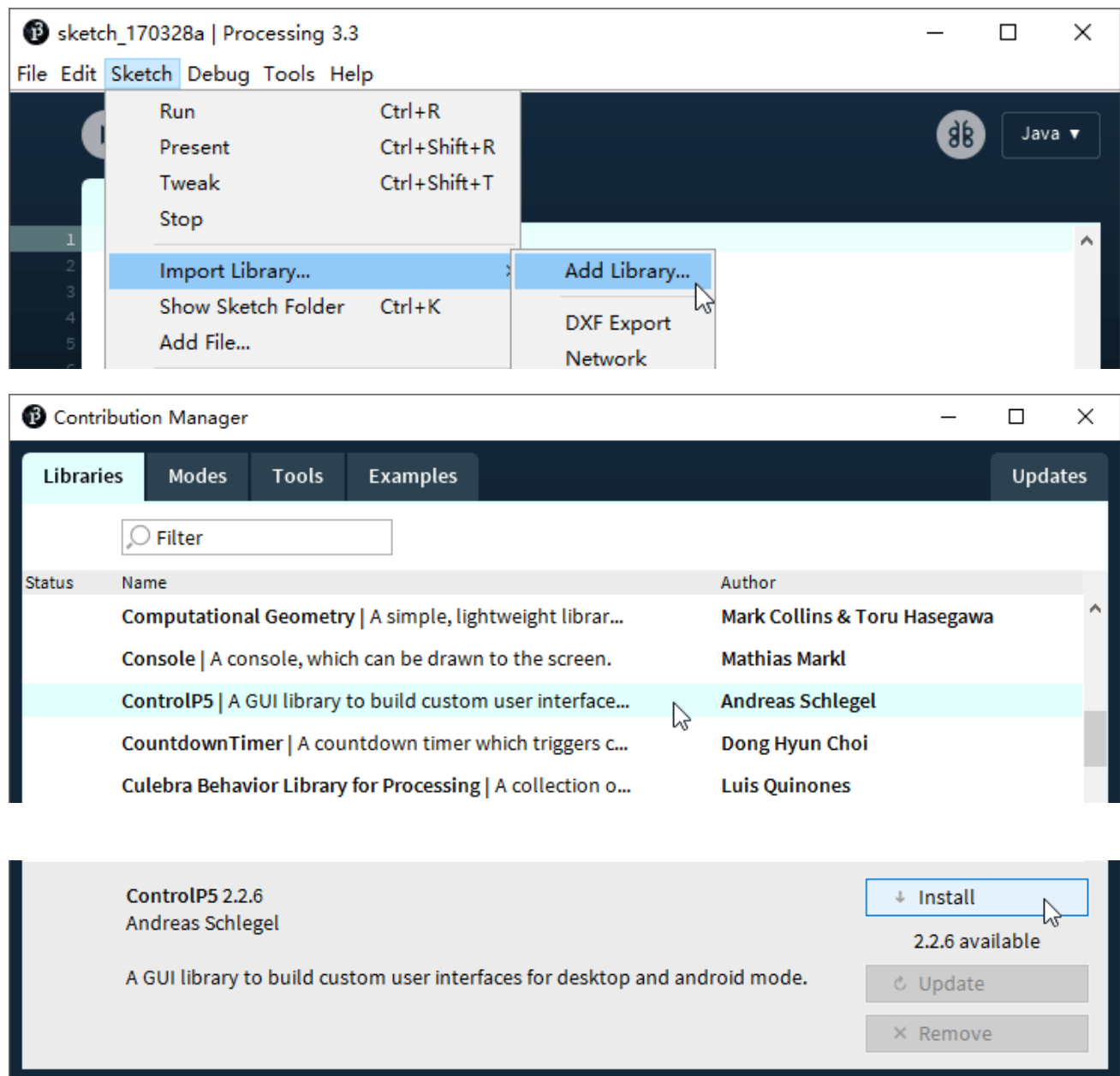


Unzip the downloaded file to your computer. Click "processing.exe" as the figure below to run this software.

 core	2020/1/17 12:16
 java	2020/1/17 12:17
 lib	2020/1/17 12:16
 modes	2020/1/17 12:16
 tools	2020/1/17 12:16
 processing.exe	2020/1/17 12:16
 processing-java.exe	2020/1/17 12:16
 revisions.txt	2020/1/17 12:16

Use Server mode for communication

Install ControlP5.



Any concerns? [✉ support@freenove.com](mailto:support@freenove.com)

Open the “Freenove\_ESP32\_WROVER\_Board\Sketches\Sketches\Sketch\_05.1\_WiFiClient\sketchWiFi\sketchWiFi.pde”, and click “Run”.

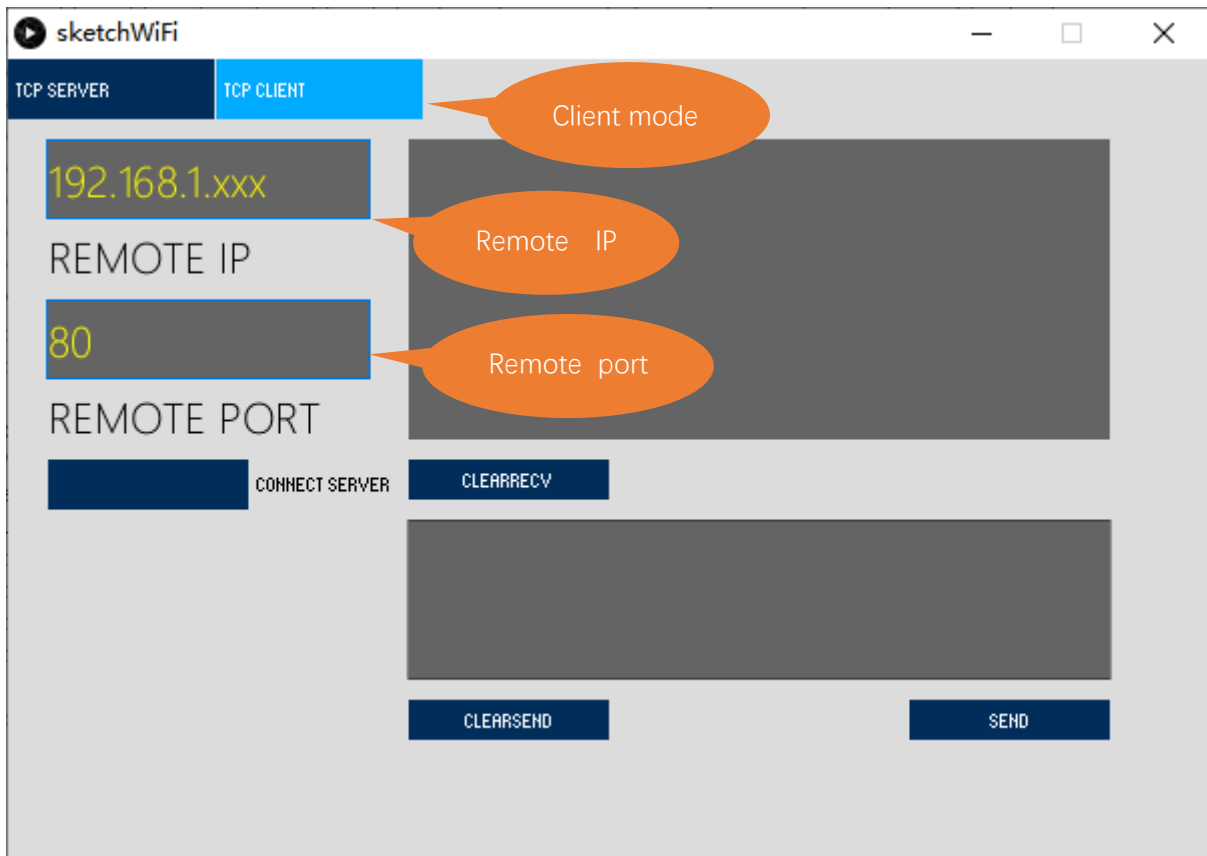


The new pop-up interface is as follows. If ESP32 is used as client, select TCP SERVER mode for sketchWiFi.



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

If ESP32 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.

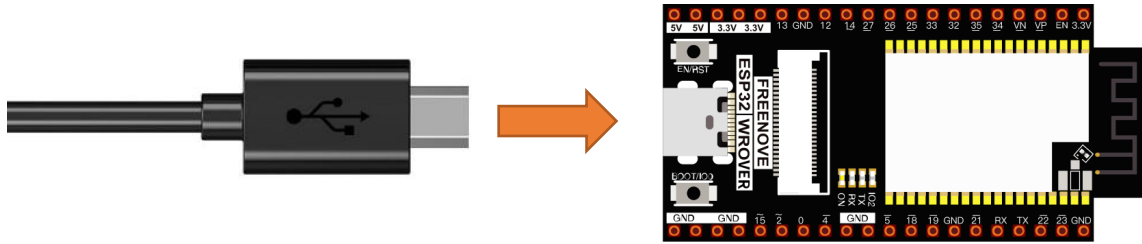
**clear receive:** clear out the content in the receiving text box

**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.

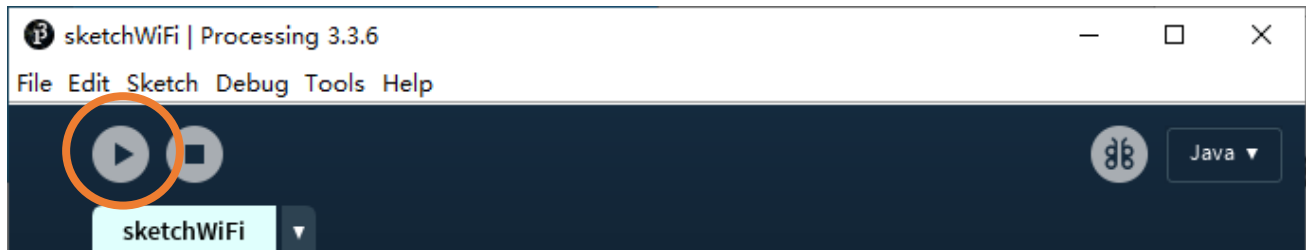
## Circuit

Connect Freenove ESP32 to the computer using USB cable.

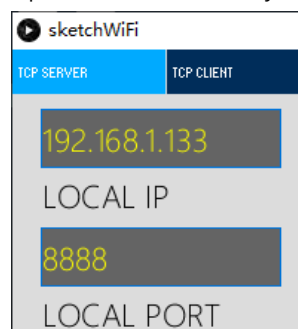


## Sketch

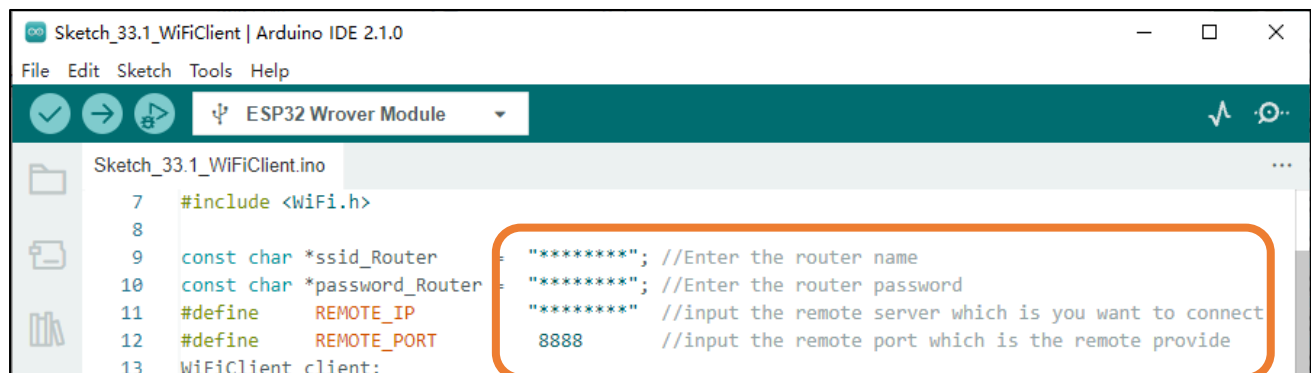
Before running the Sketch, please open "sketchWiFi.pde." first, and click "Run".



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

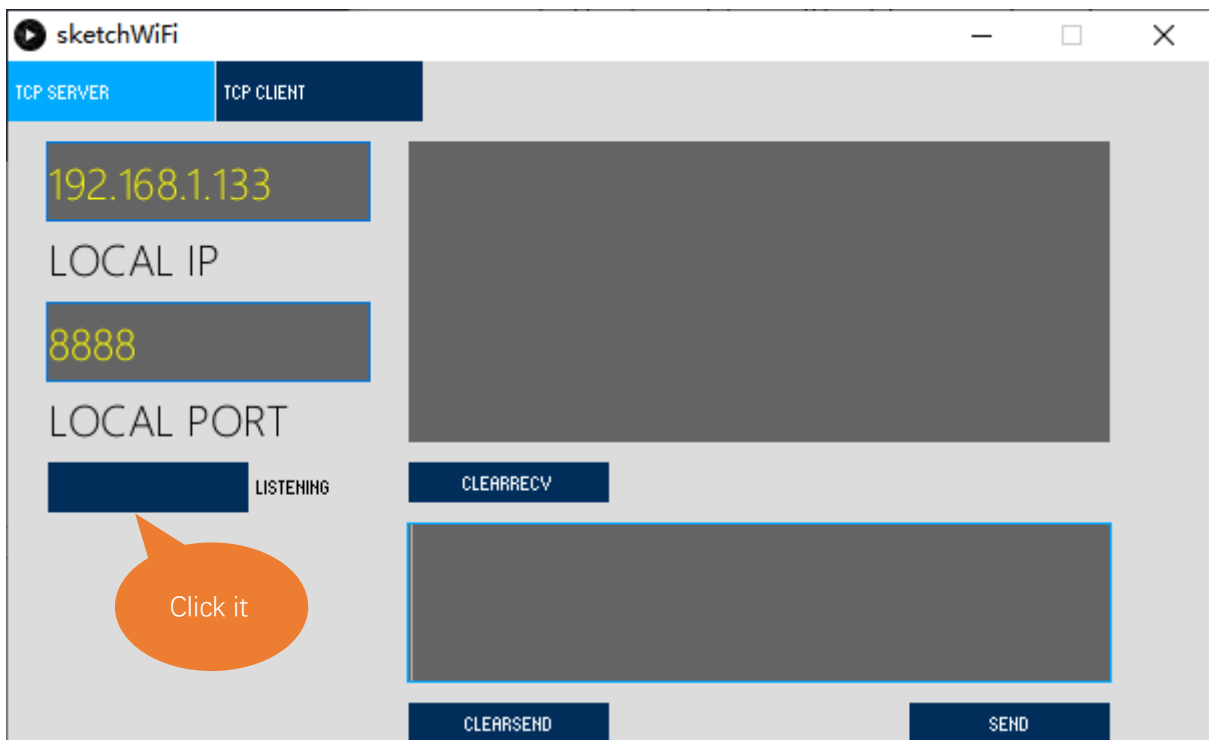


Next, open Sketch\_05.1\_WiFiClient.ino. Before running it, please change the following information based on "LOCAL IP" and "LOCAL PORT" in the figure above.



REMOTE\_IP needs to be filled in according to the interface of sketchWiFi.pde. Taking this tutorial as an example, its REMOTE\_IP is "192.168.1.133". Generally, by default, the ports do not need to change its value.

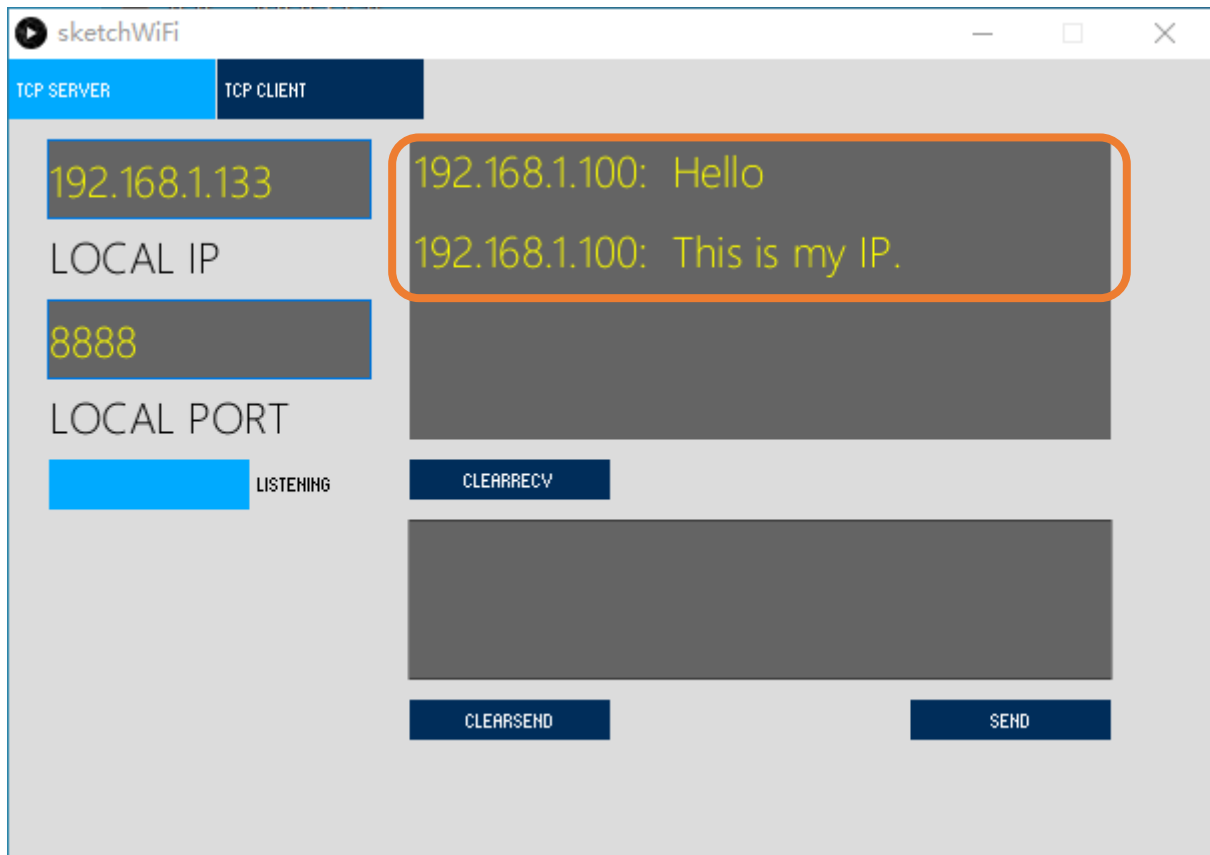
Click LISTENING, turn on TCP SERVER's data listening function and wait for ESP32 to connect.



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

```
Waiting for WiFi... .  
WiFi connected  
IP address:  
192.168.1.100  
Connecting to 192.168.1.133  
Connected
```

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



## Sketch\_05.1\_As\_Client

The following is the program code:

```

1  #include <WiFi.h>
2
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
7  WiFiClient client;
8
9  void setup() {
10     Serial.begin(115200);
11     delay(10);
12
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     }
19     Serial.println("");
20     Serial.println("WiFi connected");
21     Serial.println("IP address: ");
22     Serial.println(WiFi.localIP());
23     delay(500);
24
25     Serial.print("Connecting to ");
26     Serial.println(REMOTE_IP);
27
28     while (!client.connect(REMOTE_IP, REMOTE_PORT)) {
29         Serial.println("Connection failed.");
30         Serial.println("Waiting a moment before retrying...");
31     }
32     Serial.println("Connected");
33     client.print("Hello\n");
34     client.print("This is my IP.\n");
35
36     void loop() {
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     }
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 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.

```
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 5.2 As Server

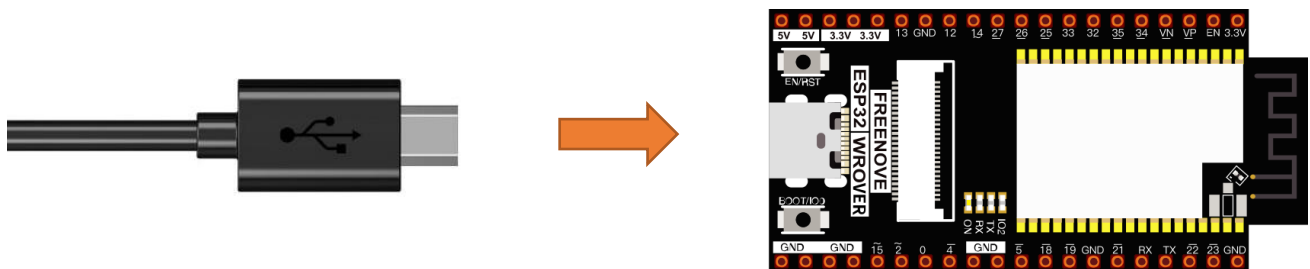
In this section, ESP32 is used as a server to wait for the connection and communication of client on the same LAN.

### Component List

ESP32-WROVER x1	Micro USB Wire x1
-----------------	-------------------

### Circuit

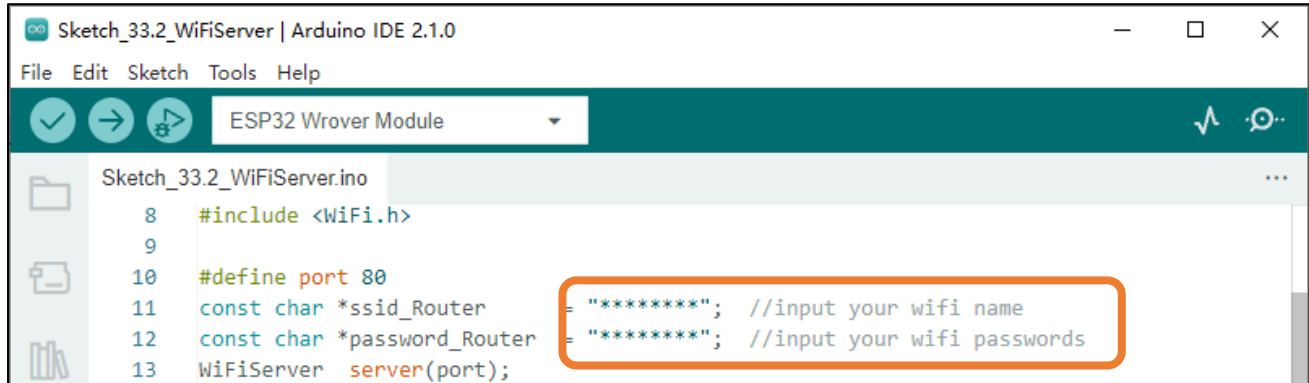
Connect Freenove ESP32 to the computer using a USB cable.



## Sketch

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

Sketch\_05.2\_As\_Server



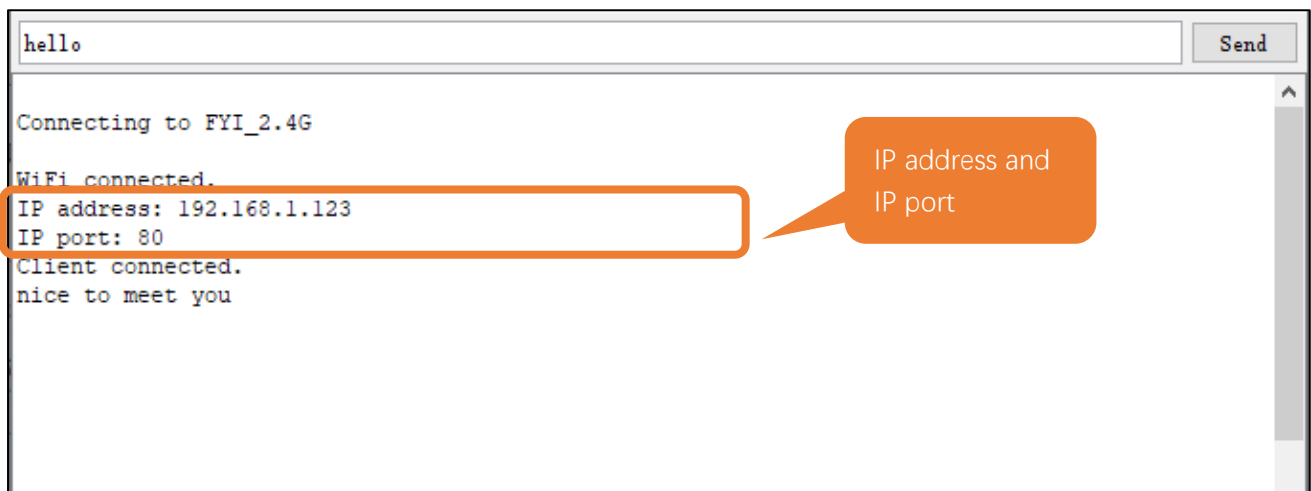
Compile and upload code to ESP32-WROVER board, open the serial monitor and set the baud rate to 115200. Turn on server mode for ESP32, waiting for the connection of other devices on the same LAN. Once a device connects to server successfully, they can send messages to each other.

If the ESP32 fails to connect to router, press the reset button as shown below and wait for ESP32 to run again.

```

09:09:32.062 -> e `Ju$Q2$LD$Q$&Q7$H$H$:0$0B$Cn$%Q_RQ*QUS$TMT_Q*MH_$Uj
09:09:32.062 -> c$K$fa$bQZA$WP'De$,k$E$V$0,$E$:0,dE$:0$0i$A_$. $x$bB$}$rv'$A0$Q}$r$'$A0$Q$DQ$Q$
09:09:32.093 -> $Q$ $008$Qa$
09:09:33.766 -> Connecting to FYI_2.4G
09:09:34.950 -> .....
  
```

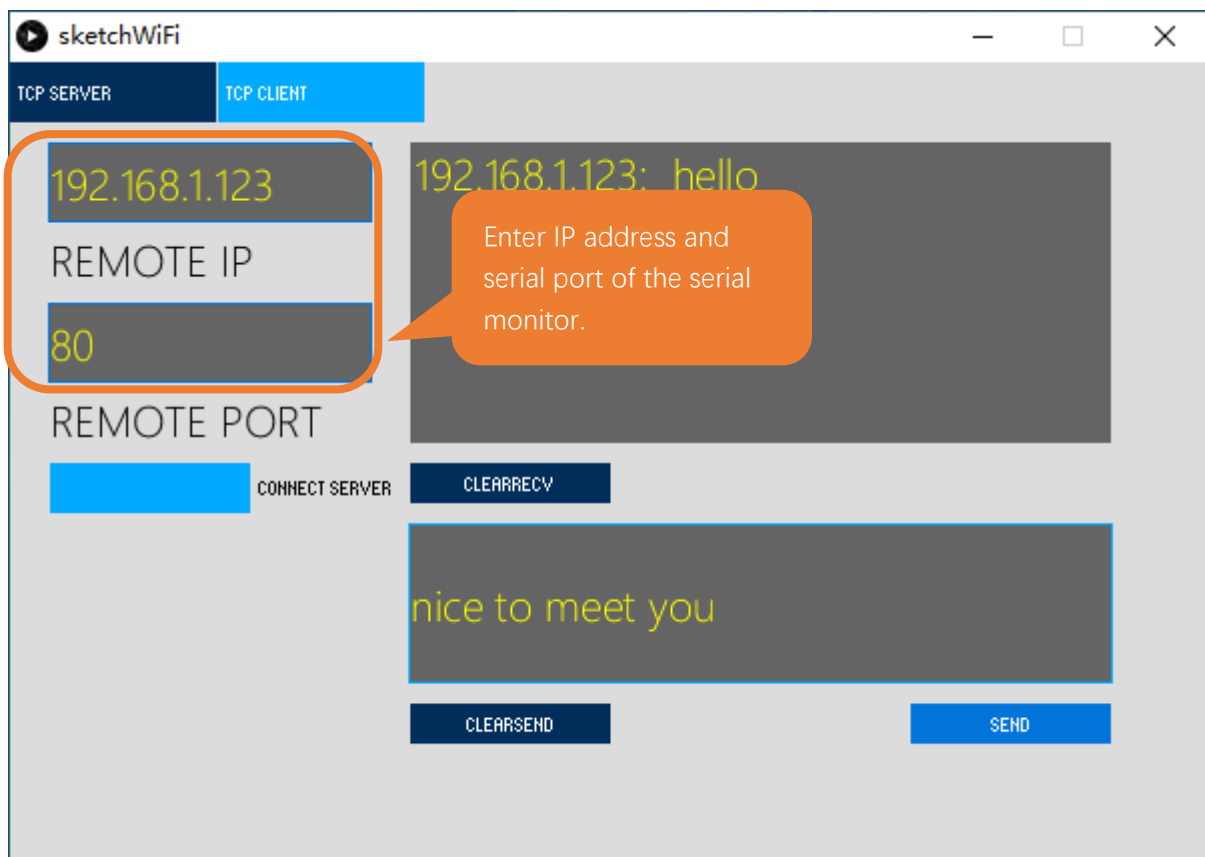
## Serial Monitor



Processing:

Open the "Freenove\_ESP32\_WROVER\_Board\Sketches\Sketches\Sketch\_05.2\_WiFiServer\sketchWiFi\sketchWiFi.pde".

Based on the messages printed by the serial monitor, enter correct IP address and serial port in Processing to establish connection and make communication.



The following is the program code:

```

1  #include <WiFi.h>
2
3  #define port 80
4  const char *ssid_Router      = "*****"; //input your wifi name
5  const char *password_Router = "*****"; //input your wifi passwords
6  WiFiServer server(port);
7
8  void setup()
9  {
10     Serial.begin(115200);
11     Serial.printf("\nConnecting to ");
12     Serial.println(ssid_Router);
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.");
22     Serial.print("IP address: ");
23     Serial.println(WiFi.localIP());
24     Serial.printf("IP port: %d\n",port);
25     server.begin(port);
26     WiFi.setAutoReconnect(true);
27 }
28
29 void loop() {
30     WiFiClient client = server.accept();           // listen for incoming clients
31     if (client) {                                   // if you get a client
32         Serial.println("Client connected.");
33         while (client.connected()) {                // loop while the client's connected
34             if (client.available()) {               // if there's bytes to read from the
client
35                 Serial.println(client.readStringUntil('\n')); // print it out the serial monitor
36                 while(client.read()>0);             // clear the wifi receive area cache
37             }
38             if(Serial.available()){                 // if there's bytes to read from the
serial monitor
39                 client.print(Serial.readStringUntil('\n')); // print it out the client.
40                 while(Serial.read()>0);             // clear the wifi receive area cache
41             }

```

```

42     }
43     client.stop(); // stop the client connecting.
44     Serial.println("Client Disconnected.");
45 }
46 }

```

Apply for method class of WiFiServer.

```

6 WiFiServer server(port); //Apply for a Server object whose port number is 80

```

Connect specified WiFi until it is successful. If the name and password of WiFi are correct but it still fails to connect, please push the reset key.

```

13 WiFi.disconnect();
14 WiFi.begin(ssid_Router, password_Router);
15 delay(1000);
16 while (WiFi.status() != WL_CONNECTED) {
17     delay(500);
18     Serial.print(".");
19 }
20 Serial.println("");
21 Serial.println("WiFi connected.");

```

Print out the IP address and port number of ESP32.

```

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

```

Turn on server mode of ESP32, turn on automatic reconnection.

```

25 server.begin(); //Turn ON ESP32 as Server mode
26 WiFi.setAutoReconnect(true);

```

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

```

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

```

## Reference

**Class Server**

Every time use Server functionality, we need to include header file "WiFi.h".

**WiFiServer(uint16\_t port=80, uint8\_t max\_clients=4):** create a TCP Server.

**port:** ports of Server; range from 0 to 65535 with the default number as 80.

**max\_clients:** maximum number of clients with default number as 4.

**begin(port):** start the TCP Server.

**port:** ports of Server; range from 0 to 65535 with the default number as 0.

**setNoDelay(bool nodelay):** whether to turn off the delay sending functionality.

**nodelay:** true stands for forbidden Nagle algorithm.

**close():** close tcp connection.

**stop():** stop tcp connection.

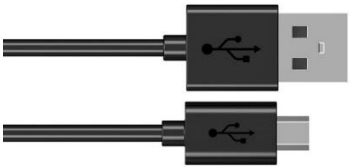
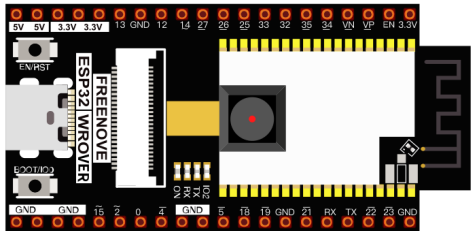
## Chapter 6 Camera Web Server

In this section, we'll use ESP32's video function as an example to study.

### Project 6.1 Camera Web Server

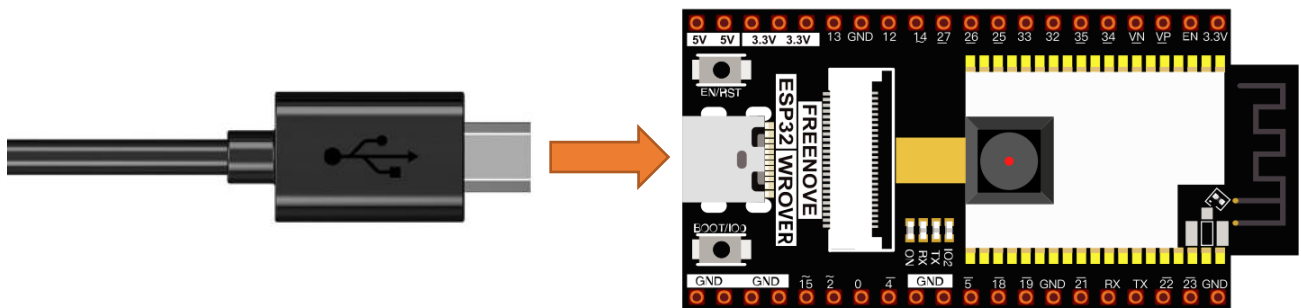
Connect ESP32 using USB and check its IP address through serial monitor. Use web page to access IP address to obtain video and image data.

### Component List

<p>Micro USB Wire x1</p> 	<p>ESP32-WROVER x1</p> 
--	--

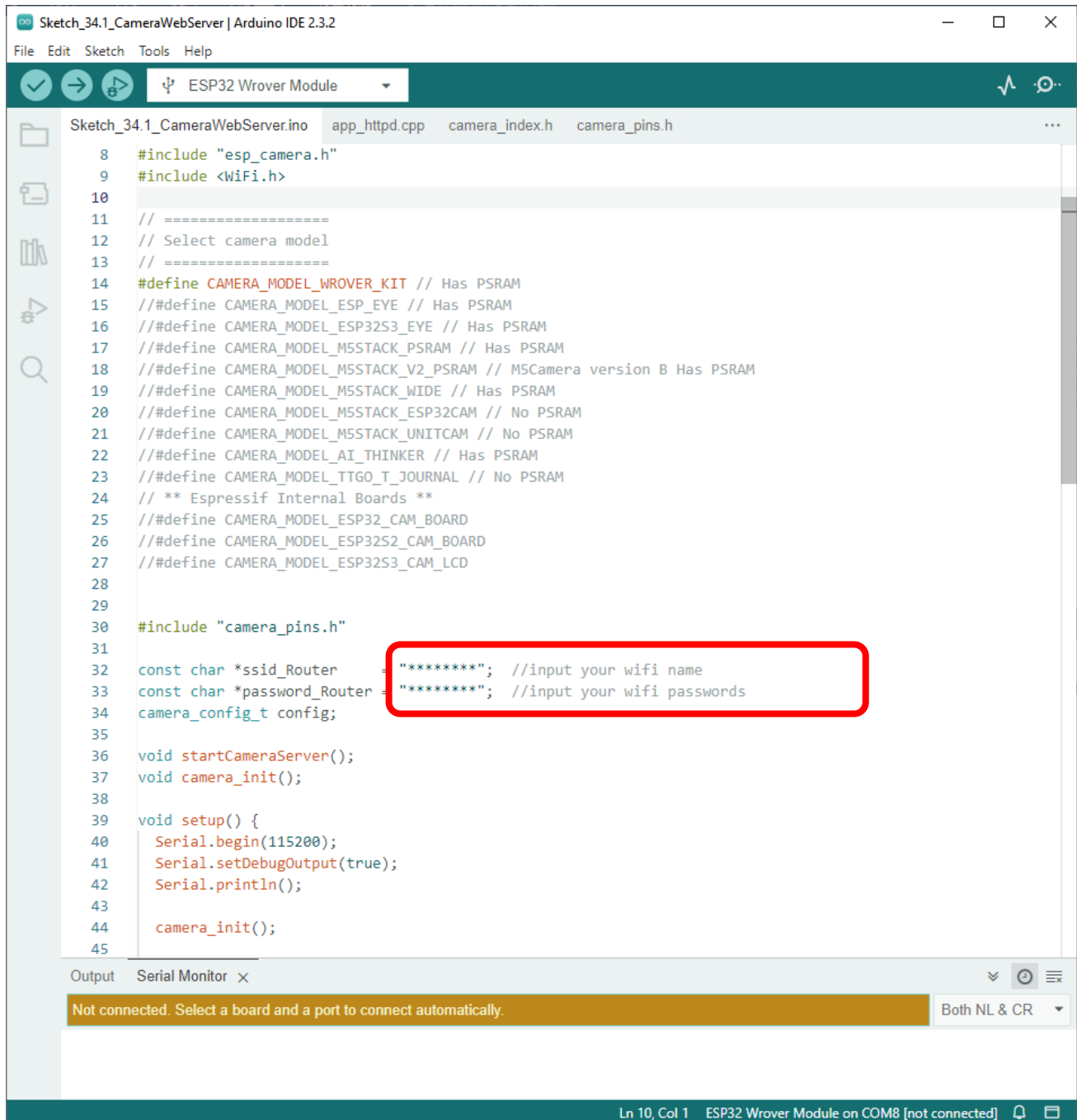
### Circuit

Connect Freenove ESP32 to the computer using USB cable.



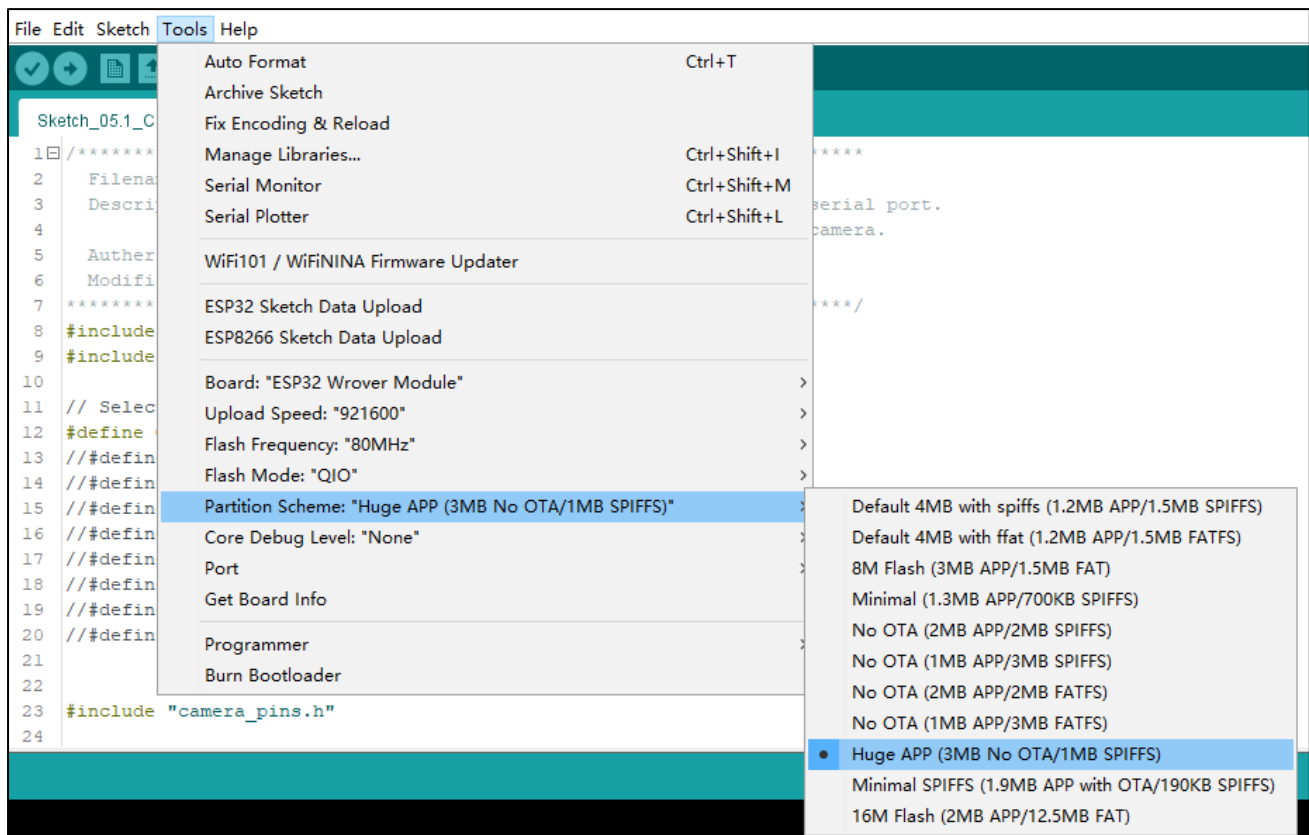
## Sketch

### Sketch\_06.1\_As\_CameraWebServer



```
Sketch_34.1_CameraWebServer | Arduino IDE 2.3.2
File Edit Sketch Tools Help
ESP32 Wrover Module
Sketch_34.1_CameraWebServer.ino app_httpd.cpp camera_index.h camera_pins.h
8 #include "esp_camera.h"
9 #include <WiFi.h>
10
11 // =====
12 // Select camera model
13 // =====
14 #define CAMERA_MODEL_WROVER_KIT // Has PSRAM
15 //#define CAMERA_MODEL_ESP_EYE // Has PSRAM
16 //#define CAMERA_MODEL_ESP32S3_EYE // Has PSRAM
17 //#define CAMERA_MODEL_M5STACK_PSRAM // Has PSRAM
18 //#define CAMERA_MODEL_M5STACK_V2_PSRAM // M5Camera version B Has PSRAM
19 //#define CAMERA_MODEL_M5STACK_WIDE // Has PSRAM
20 //#define CAMERA_MODEL_M5STACK_ESP32CAM // No PSRAM
21 //#define CAMERA_MODEL_M5STACK_UNITCAM // No PSRAM
22 //#define CAMERA_MODEL_AI_THINKER // Has PSRAM
23 //#define CAMERA_MODEL_TTGO_T_JOURNAL // No PSRAM
24 // ** Espressif Internal Boards **
25 //#define CAMERA_MODEL_ESP32_CAM_BOARD
26 //#define CAMERA_MODEL_ESP32S2_CAM_BOARD
27 //#define CAMERA_MODEL_ESP32S3_CAM_LCD
28
29
30 #include "camera_pins.h"
31
32 const char *ssid_Router = "*****"; //input your wifi name
33 const char *password_Router = "*****"; //input your wifi passwords
34 camera_config_t config;
35
36 void startCameraServer();
37 void camera_init();
38
39 void setup() {
40     Serial.begin(115200);
41     Serial.setDebugOutput(true);
42     Serial.println();
43
44     camera_init();
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000
1001
1002
1003
1004
1005
1006
1007
1008
1009
1010
1011
1012
1013
1014
1015
1016
1017
1018
1019
1020
1021
1022
1023
1024
1025
1026
1027
1028
1029
1030
1031
1032
1033
1034
1035
1036
1037
1038
1039
1040
1041
1042
1043
1044
1045
1046
1047
1048
1049
1050
1051
1052
1053
1054
1055
1056
1057
1058
1059
1060
1061
1062
1063
1064
1065
1066
1067
1068
1069
1070
1071
1072
1073
1074
1075
1076
1077
1078
1079
1080
1081
1082
1083
1084
1085
1086
1087
1088
1089
1090
1091
1092
1093
1094
1095
1096
1097
1098
1099
1100
1101
1102
1103
1104
1105
1106
1107
1108
1109
1110
1111
1112
1113
1114
1115
1116
1117
1118
1119
1120
1121
1122
1123
1124
1125
1126
1127
1128
1129
1130
1131
1132
1133
1134
1135
1136
1137
1138
1139
1140
1141
1142
1143
1144
1145
1146
1147
1148
1149
1150
1151
1152
1153
1154
1155
1156
1157
1158
1159
1160
1161
1162
1163
1164
1165
1166
1167
1168
1169
1170
1171
1172
1173
1174
1175
1176
1177
1178
1179
1180
1181
1182
1183
1184
1185
1186
1187
1188
1189
1190
1191
1192
1193
1194
1195
1196
1197
1198
1199
1200
1201
1202
1203
1204
1205
1206
1207
1208
1209
1210
1211
1212
1213
1214
1215
1216
1217
1218
1219
1220
1221
1222
1223
1224
1225
1226
1227
1228
1229
1230
1231
1232
1233
1234
1235
1236
1237
1238
1239
1240
1241
1242
1243
1244
1245
1246
1247
1248
1249
1250
1251
1252
1253
1254
1255
1256
1257
1258
1259
1260
1261
1262
1263
1264
1265
1266
1267
1268
1269
1270
1271
1272
1273
1274
1275
1276
1277
1278
1279
1280
1281
1282
1283
1284
1285
1286
1287
1288
1289
1290
1291
1292
1293
1294
1295
1296
1297
1298
1299
1300
1301
1302
1303
1304
1305
1306
1307
1308
1309
1310
1311
1312
1313
1314
1315
1316
1317
1318
1319
1320
1321
1322
1323
1324
1325
1326
1327
1328
1329
1330
1331
1332
1333
1334
1335
1336
1337
1338
1339
1340
1341
1342
1343
1344
1345
1346
1347
1348
1349
1350
1351
1352
1353
1354
1355
1356
1357
1358
1359
1360
1361
1362
1363
1364
1365
1366
1367
1368
1369
1370
1371
1372
1373
1374
1375
1376
1377
1378
1379
1380
1381
1382
1383
1384
1385
1386
1387
1388
1389
1390
1391
1392
1393
1394
1395
1396
1397
1398
1399
1400
1401
1402
1403
1404
1405
1406
1407
1408
1409
1410
1411
1412
1413
1414
1415
1416
1417
1418
1419
1420
1421
1422
1423
1424
1425
1426
1427
1428
1429
1430
1431
1432
1433
1434
1435
1436
1437
1438
1439
1440
1441
1442
1443
1444
1445
1446
1447
1448
1449
1450
1451
1452
1453
1454
1455
1456
1457
1458
1459
1460
1461
1462
1463
1464
1465
1466
1467
1468
1469
1470
1471
1472
1473
1474
1475
1476
1477
1478
1479
1480
1481
1482
1483
1484
1485
1486
1487
1488
1489
1490
1491
1492
1493
1494
1495
1496
1497
1498
1499
1500
1501
1502
1503
1504
1505
1506
1507
1508
1509
1510
1511
1512
1513
1514
1515
1516
1517
1518
1519
1520
1521
1522
1523
1524
1525
1526
1527
1528
1529
1530
1531
1532
1533
1534
1535
1536
1537
1538
1539
1540
1541
1542
1543
1544
1545
1546
1547
1548
1549
1550
1551
1552
1553
1554
1555
1556
1557
1558
1559
1560
1561
1562
1563
1564
1565
1566
1567
1568
1569
1570
1571
1572
1573
1574
1575
1576
1577
1578
1579
1580
1581
1582
1583
1584
1585
1586
1587
1588
1589
1590
1591
1592
1593
1594
1595
1596
1597
1598
1599
1600
1601
1602
1603
1604
1605
1606
1607
1608
1609
1610
1611
1612
1613
1614
1615
1616
1617
1618
1619
1620
1621
1622
1623
1624
1625
1626
1627
1628
1629
1630
1631
1632
1633
1634
1635
1636
1637
1638
1639
1640
1641
1642
1643
1644
1645
1646
1647
1648
1649
1650
1651
1652
1653
1654
1655
1656
1657
1658
1659
1660
1661
1662
1663
1664
1665
1666
1667
1668
1669
1670
1671
1672
1673
1674
1675
1676
1677
1678
1679
1680
1681
1682
1683
1684
1685
1686
1687
1688
1689
1690
1691
1692
1693
1694
1695
1696
1697
1698
1699
1700
1701
1702
1703
1704
1705
1706
1707
1708
1709
1710
1711
1712
1713
1714
1715
1716
1717
1718
1719
1720
1721
1722
1723
1724
1725
1726
1727
1728
1729
1730
1731
1732
1733
1734
1735
1736
1737
1738
1739
1740
1741
1742
1743
1744
1745
1746
1747
1748
1749
1750
1751
1752
1753
1754
1755
1756
1757
1758
1759
1760
1761
1762
1763
1764
1765
1766
1767
1768
1769
1770
1771
1772
1773
1774
1775
1776
1777
1778
1779
1780
1781
1782
1783
1784
1785
1786
1787
1788
1789
1790
1791
1792
1793
1794
1795
1796
1797
1798
1799
1800
1801
1802
1803
1804
1805
1806
1807
1808
1809
1810
1811
1812
1813
1814
1815
1816
1817
1818
1819
1820
1821
1822
1823
1824
1825
1826
1827
1828
1829
1830
1831
1832
1833
1834
1835
1836
1837
1838
1839
1840
1841
1842
1843
1844
1845
1846
1847
1848
1849
1850
1851
1852
1853
1854
1855
1856
1857
1858
1859
1860
1861
1862
1863
1864
1865
1866
1867
1868
1869
1870
1871
1872
1873
1874
1875
1876
1877
1878
1879
1880
1881
1882
1883
1884
1885
1886
1887
1888
1889
1890
1891
1892
1893
1894
1895
1896
1897
1898
1899
1900
1901
1902
1903
1904
1905
1906
1907
1908
1909
1910
1911
1912
1913
1914
1915
1916
1917
1918
1919
1920
1921
1922
1923
1924
1925
1926
1927
1928
1929
1930
1931
1932
1933
1934
1935
1936
1937
1938
1939
1940
1941
1942
1943
1944
1945
1946
1947
1948
1949
1950
1951
1952
1953
1954
1955
1956
1957
1958
1959
1960
1961
1962
1963
1964
1965
1966
1967
1968
1969
1970
1971
1972
1973
1974
1975
1976
1977
1978
1979
1980
1981
1982
1983
1984
1985
1986
1987
1988
1989
1990
1991
1992
1993
1994
1995
1996
1997
1998
1999
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2010
2011
2012
2013
2014
2015
2016
2017
2018
2019
2020
2021
2022
2023
2024
2025
2026
2027
2028
2029
2030
2031
2032
2033
2034
2035
2036
2037
2038
2039
2040
2041
2042
2043
2044
2045
2046
2047
2048
2049
2050
2051
2052
2053
2054
2055
2056
2057
2058
2059
2060
2061
2062
2063
2064
2065
2066
2067
2068
2069
2070
2071
2072
2073
2074
2075
2076
2077
2078
2079
2080
2081
2082
2083
2084
2085
2086
2087
2088
2089
2090
2091
2092
2093
2094
2095
2096
2097
2098
2099
2100
2101
2102
2103
2104
2105
2106
2107
2108
2109
2110
2111
2112
2113
2114
2115
2116
2117
2118
2119
2120
2121
2122
2123
2124
2125
2126
2127
2128
2129
2130
2131
2132
2133
2134
2135
2136
2137
2138
2139
2140
2141
2142
2143
2144
2145
2146
2147
2148
2149
2150
2151
2152
2153
2154
2155
2156
2157
2158
2159
2160
2161
2162
2163
2164
2165
2166
2167
2168
2169
2170
2171
2172
2173
2174
2175
2176
2177
2178
2179
2180
2181
2182
2183
2184
2185
2186
2187
2188
2189
2190
2191
2192
2193
2194
2195
2196
2197
2198
2199
2200
2201
2202
2203
2204
2205
2206
2207
2208
2209
2210
2211
2212
2213
2214
2215
2216
2217
2218
2219
2220
2221
2222
2223
2224
2225
2226
2227
2228
2229
2230
2231
2232
2233
2234
2235
2236
2237
2238
2239
2240
2241
2242
2243
2244
2245
2246
2247
2248
2249
2250
2251
2252
2253
2254
2255
2256
2257
2258
2259
2260
2261
2262
2263
2264
2265
2266
2267
2268
2269
2270
2271
2272
2273
2274
2275
2276
2277
2278
2279
2280
2281
2282
2283
2284
2285
2286
2287
2288
2289
2290
2291
2292
2293
2294
2295
2296
2297
2298
2299
2300
2301
2302
2303
2304
2305
2306
2307
2308
2309
2310
2311
2312
2313
2314
2315
2316
2317
2318
2319
2320
2321
2322
2323
2324
2325
2326
2327
2328
2329
2330
2331
2332
2333
2334
2335
2336
2337
2338
2339
2340
2341
2342
2343
2344
2345
2346
2347
2348
2349
2350
2351
2352
2353
2354
2355
2356
2357
2358
2359
2360
2361
2362
2363
2364
2365
2366
2367
2368
2369
2370
2371
2372
2373
2374
2375
2376
2377
2378
2379
2380
2381
2382
2383
2384
2385
2386
2387
2388
2389
2390
2391
2392
2393
2394
2395
2396
2397
2398
2399
2400
2401
2402
2403
2404
2405
2406
2407
2408
2409
2410
2411
2412
2413
2414
2415
2416
2417
2418
2419
2420
2421
2422
2423
2424
2425
2426
2427
2428
2429
2430
2431
2432
2433
2434
2435
2436
2437
2438
2439
2440
2441
2442
2443
2444
2445
2446
2447
2448
2449
2450
2451
2452
2453
2454
2455
2456
2457
2458
2459
2460
2461
2462
2463
2464
2465
2466
2467
2468
2469
2470
2471
2472
2473
2474
2475
2476
2477
2478
2479
2480
2481
2482
2483
2484
2485
2486
2487
2488
2489
2490
2491
2492
2493
2494
2495
2496
2497
2498
2499
2500
2501
2502
2503
2504
2505
2506
2507
2508
2509
2510
2511
2512
2513
2514
2515
2516
2517
2518
2519
2520
2521
2522
2523
2524
2525
2526
2527
2528
2529
2530
2531
2532
2533
2534
2535
2536
2537
2538
2539
2540
2541
2542
2543
2544
2545
2546
2547
2548
2549
2550
2551
2552
2553
2554
2555
2556
2557
2558
2559
2560
2561
2562
2563
2564
2565
2566
2567
2568
2569
2570
2571
2572
2573
2574
25
```

If your Arduino IDE prompts you that your sketch is out of your project's storage space, compile the code again as configured below.



Compile and upload codes to ESP32, open the serial monitor and set the baud rate to 115200, and the serial monitor will print out a network link address.

```
WiFi connected
Starting web server on port: '80'
Starting stream server on port: '81'
Camera Ready! Use 'http://192.168.1.100' to connect
```

If your ESP32 has been in the process of connecting to router, but the information above has not been printed out, please re-check whether the router name and password have been entered correctly and press the reset key on ESP32-WROVER to wait for a successful connection prompt.

Open a web browser, enter the IP address printed by the serial monitor in the address bar, and access it. Taking the Google browser as an example, here's what the browser prints out after successful access to ESP32's IP.

We recommend that the resolution not exceed VGA(640x480).

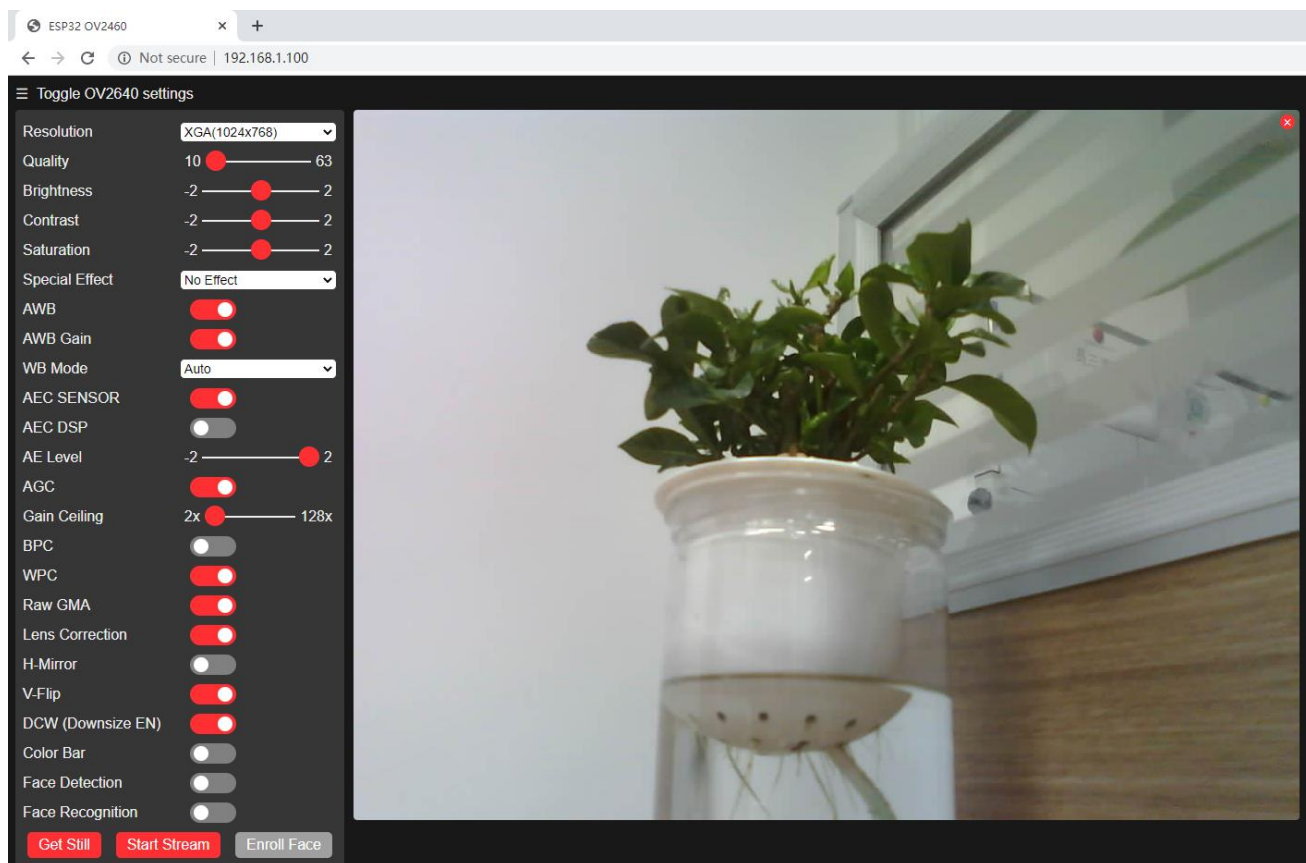
The screenshot shows a web browser window with the title 'ESP32 OV2640' and the address bar displaying '192.168.1.100'. The page content is titled 'Toggle OV2640 settings' and lists various camera parameters. Annotations with orange callouts point to specific features:

- enter IP address**: Points to the address bar.
- select pixel of the picture**: Points to the 'Resolution' dropdown menu, which is set to 'VGA(640x480)'.
- adjust camera parameters**: Points to the 'Quality' slider, which is set to 10.
- set camera left and right, up and down**: Points to the 'V-Flip' toggle switch, which is turned on.
- crop the picture**: Points to the 'Get Still' button.
- turn on video**: Points to the 'Start Stream' button.

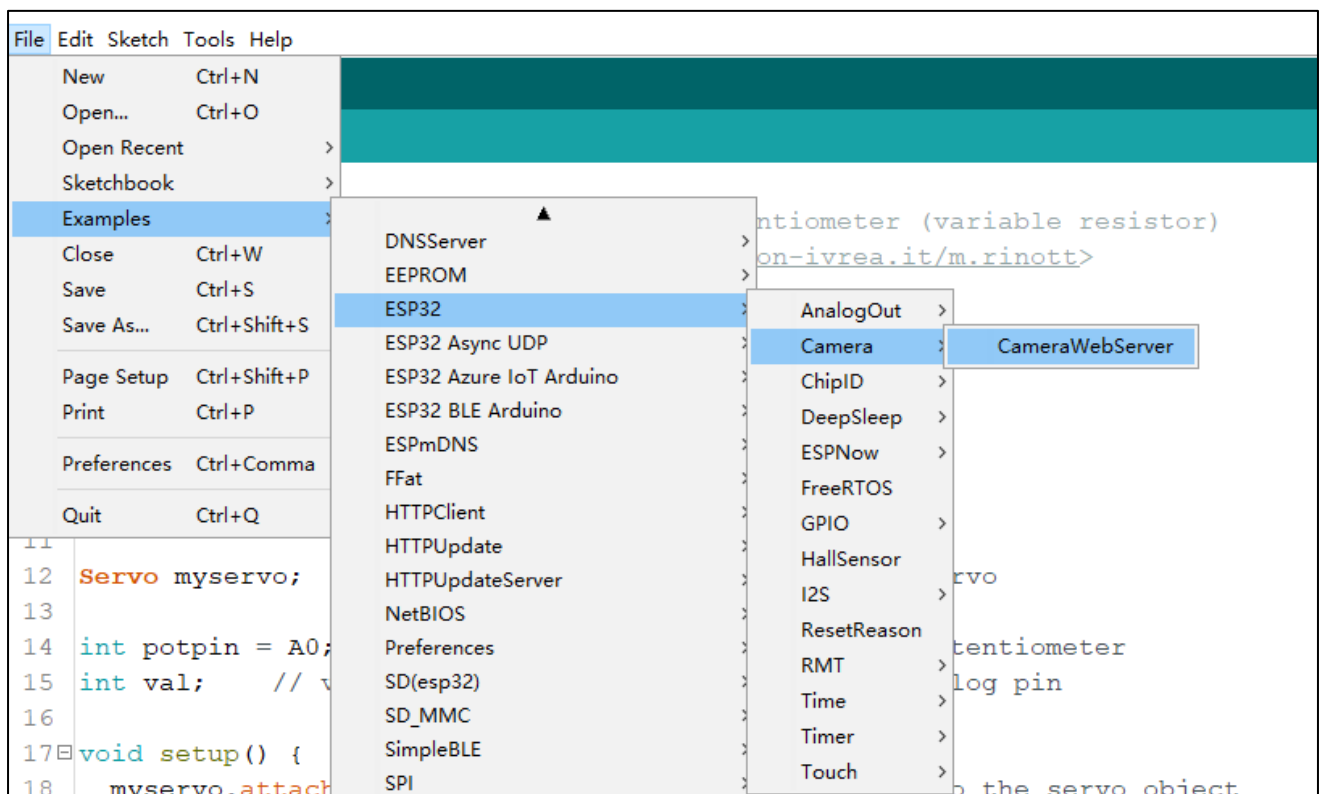
The settings list includes:

- Resolution: VGA(640x480)
- Quality: 10
- Brightness: -2 to 2
- Contrast: -2 to 2
- Saturation: -2 to 2
- Special Effect: No Effect
- AWB: On
- AWB Gain: On
- WB Mode: Auto
- AEC SENSOR: On
- AEC DSP: Off
- AE Level: -2 to 2
- AGC: On
- Gain Ceiling: 2x to 128x
- BPC: Off
- WPC: On
- Raw GMA: On
- Lens Correction: On
- H-Mirror: Off
- V-Flip: On
- DCW (Downsize EN): On
- Color Bar: Off
- Face Detection: Off
- Face Recognition: Off

Click on Start Stream. The effect is shown in the image below.



**Note:** If sketch compilation fails due to ESP32 support package, follow the steps of the image to open the CameraWebServer. This sketch is the same as described in the tutorial above.



The following is the main program code. You need include other code files in the same folder when write your own code.

```
1  #include "esp_camera.h"
2  #include <WiFi.h>
3
4  // Select camera model
5  #define CAMERA_MODEL_WROVER_KIT
6  //#define CAMERA_MODEL_ESP_EYE
7  //#define CAMERA_MODEL_M5STACK_PSRAM
8  //#define CAMERA_MODEL_M5STACK_WIDE
9  //#define CAMERA_MODEL_AI_THINKER
10
11 #include "camera_pins.h"
12
13 const char *ssid_Router    = "*****"; //input your wifi name
14 const char *password_Router = "*****"; //input your wifi passwords
15 camera_config_t config;
16
17 void startCameraServer();
18 void camera_init();
19
20 void setup() {
21     Serial.begin(115200);
22     Serial.setDebugOutput(true);
23     Serial.println();
24
25     camera_init();
26     config.frame_size = FRAMESIZE_VGA;
27     config.jpeg_quality = 10;
28
29     // camera init
30     esp_err_t err = esp_camera_init(&config);
31     if (err != ESP_OK) {
32         Serial.printf("Camera init failed with error 0x%x", err);
33         return;
34     }
35
36     sensor_t * s = esp_camera_sensor_get();
37     s->set_vflip(s, 1); //flip it back
38     s->set_brightness(s, 1); //up the blightness just a bit
39     s->set_saturation(s, -1); //lower the saturation
40
41     WiFi.begin(ssid_Router, password_Router);
42     while (WiFi.status() != WL_CONNECTED) {
```

```
43     delay(500);
44     Serial.print(".");
45 }
46 Serial.println("");
47 Serial.println("WiFi connected");
48
49 startCameraServer();
50
51 Serial.print("Camera Ready! Use 'http://");
52 Serial.print(WiFi.localIP());
53 Serial.println("' to connect");
54 }
55
56 void loop() {
57     ;
58 }
59
60 void camera_init(){
61     config.ledc_channel = LEDC_CHANNEL_0;
62     config.ledc_timer = LEDC_TIMER_0;
63     config.pin_d0 = Y2_GPIO_NUM;
64     config.pin_d1 = Y3_GPIO_NUM;
65     config.pin_d2 = Y4_GPIO_NUM;
66     config.pin_d3 = Y5_GPIO_NUM;
67     config.pin_d4 = Y6_GPIO_NUM;
68     config.pin_d5 = Y7_GPIO_NUM;
69     config.pin_d6 = Y8_GPIO_NUM;
70     config.pin_d7 = Y9_GPIO_NUM;
71     config.pin_xclk = XCLK_GPIO_NUM;
72     config.pin_pclk = PCLK_GPIO_NUM;
73     config.pin_vsync = VSYNC_GPIO_NUM;
74     config.pin_href = HREF_GPIO_NUM;
75     config.pin_sccb_sda = SIOD_GPIO_NUM;
76     config.pin_sccb_scl = SIOC_GPIO_NUM;
77     config.pin_pwdn = PWDN_GPIO_NUM;
78     config.pin_reset = RESET_GPIO_NUM;
79     config.xclk_freq_hz = 20000000;
80     config.pixel_format = PIXFORMAT_JPEG;
81     config.fb_count = 1;
82 }
```

Add procedure files and API interface files related to ESP32 camera.

```

1  #include "esp_camera.h"
2  #include <WiFi.h>
3
4  // Select camera model
5  #define CAMERA_MODEL_WROVER_KIT
6  //#define CAMERA_MODEL_ESP_EYE
7  //#define CAMERA_MODEL_M5STACK_PSRAM
8  //#define CAMERA_MODEL_M5STACK_WIDE
9  //#define CAMERA_MODEL_AI_THINKER
10
11 #include "camera_pins.h"

```

Enter the name and password of the router

```

13 const char *ssid_Router    = "*****"; //input your wifi name
14 const char *password_Router = "*****"; //input your wifi passwords

```

Initialize serial port, set baud rate to 115200; open the debug and output function of the serial.

```

21 Serial.begin(115200);
22 Serial.setDebugOutput(true);
23 Serial.println();

```

Configure parameters including interface pins of the camera. Note: It is generally not recommended to change them.

```

60 void camera_init(){
61     config.ledc_channel = LEDC_CHANNEL_0;
62     config.ledc_timer = LEDC_TIMER_0;
63     config.pin_d0 = Y2_GPIO_NUM;
64     config.pin_d1 = Y3_GPIO_NUM;
65     config.pin_d2 = Y4_GPIO_NUM;
66     config.pin_d3 = Y5_GPIO_NUM;
67     config.pin_d4 = Y6_GPIO_NUM;
68     config.pin_d5 = Y7_GPIO_NUM;
69     config.pin_d6 = Y8_GPIO_NUM;
70     config.pin_d7 = Y9_GPIO_NUM;
71     config.pin_xclk = XCLK_GPIO_NUM;
72     config.pin_pclk = PCLK_GPIO_NUM;
73     config.pin_vsync = VSYNC_GPIO_NUM;
74     config.pin_href = HREF_GPIO_NUM;
75     config.pin_sccb_sda = SIOD_GPIO_NUM;
76     config.pin_sccb_scl = SIOC_GPIO_NUM;
77     config.pin_pwdn = PWDN_GPIO_NUM;
78     config.pin_reset = RESET_GPIO_NUM;
79     config.xclk_freq_hz = 20000000;
80     config.pixel_format = PIXFORMAT_JPEG;
81     config.fb_count = 1;
82 }

```

ESP32 connects to the router and prints a successful connection prompt. If it has not been successfully connected, press the reset key on the ESP32-WROVER.

```

41  WiFi.begin(ssid_Router, password_Router);
42  while (WiFi.status() != WL_CONNECTED) {
43      delay(500);
44      Serial.print(".");
45  }
46  Serial.println("");
47  Serial.println("WiFi connected");

```

Open the video streams server function of the camera and print its IP address via serial port.

```

49  startCameraServer();
50
51  Serial.print("Camera Ready! Use 'http://");
52  Serial.print(WiFi.localIP());
53  Serial.println("' to connect");

```

Configure the display image information of the camera.

The `set_vflip()` function sets whether the image is flipped 180°, with 0 for no flip and 1 for flip 180°.

The `set_brightness()` function sets the brightness of the image, with values ranging from -2 to 2.

The `set_saturation()` function sets the color saturation of the image, with values ranging from -2 to 2.

```

36  sensor_t * s = esp_camera_sensor_get();
37  s->set_vflip(s, 1);           //flip it back
38  s->set_brightness(s, 1);      //up the blightness just a bit
39  s->set_saturation(s, -1);     //lower the saturation

```

Modify the resolution and sharpness of the images captured by the camera. The sharpness ranges from 10 to 63, and the smaller the number, the sharper the picture. The larger the number, the blurrier the picture. Please refer to the table below.

```

26  config.frame_size = FRAMESIZE_VGA;
27  config.jpeg_quality = 10;

```

#### Reference

Image resolution	Sharpness	Image resolution	Sharpness
FRAMESIZE_96x96	96x96	FRAMESIZE_HVGA	480x320
FRAMESIZE_QQVGA	160x120	FRAMESIZE_VGA	640x480
FRAMESIZE_QCIF	176x144	FRAMESIZE_SVGA	800x600
FRAMESIZE_HQVGA	240x176	FRAMESIZE_XGA	1024x768
FRAMESIZE_240x240	240x240	FRAMESIZE_HD	1280x720
FRAMESIZE_QVGA	320x240	FRAMESIZE_SXGA	1280x1024
FRAMESIZE_CIF	400x296	FRAMESIZE_UXGA	1600x1200

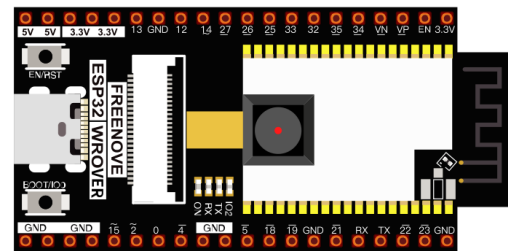
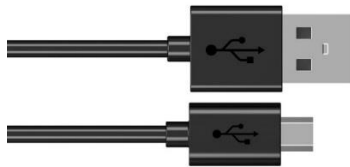
We recommend that the resolution not exceed VGA(640x480).

## Project 6.2 Video Web Server

Connect to ESP32 using USB and view its IP address through a serial monitor. Access IP addresses through web pages to obtain real-time video data.

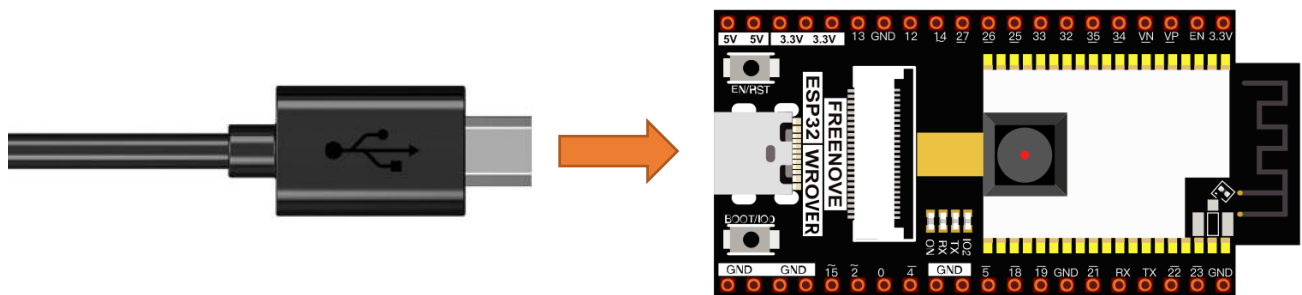
### Component List

Micro USB Wire x1	ESP32-WROVER x1
-------------------	-----------------



### Circuit

Connect Freenove ESP32 to the computer using USB cable.



## Sketch

### Sketch\_06.2\_As\_VideoWebServer



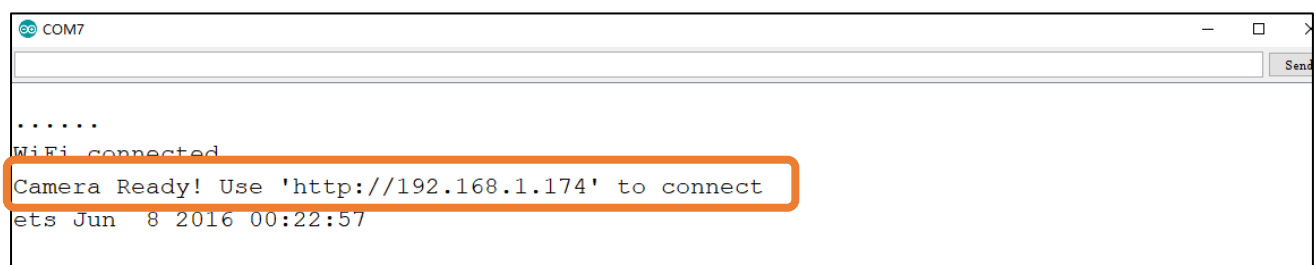
```
File Edit Sketch Tools Help
Sketch_32.2_As_VideoWebServer
11
12 #include "camera_pins.h"
13
14 const char* ssid = "*****"; //input your wifi name
15 const char* password = "*****"; //input your wifi passwords
16
17 void startCameraServer(),
18
19 void setup() {
20   Serial.begin(115200);
21   Serial.setDebugOutput(true);
22   Serial.println();
23
24   camera_config_t config;
25   config.ledc_channel = LEDC_CHANNEL_0;
26   config.ledc_timer = LEDC_TIMER_0;
27
28   //WiFi.begin(ssid, password);
29   //while(WiFi.status() != WL_CONNECTED) {
30   //  delay(5000);
31   //  Serial.print(".");
32   //}
33   //Serial.println("");
34   //Serial.print("IP address: ");
35   //Serial.println(WiFi.localIP());
36
37   Serial.println("Camera Ready! Use 'http://192.168.1.174' to connect");
38
39   //WiFi.disconnect();
40   //delay(1000);
41   //Serial.println("");
42   //Serial.println("Leaving...");
43   //Serial.println("Hard resetting via RTS pin...");
44 }
```

Done Saving.  
Write 3072 bytes (120 compressed) at 0x00000000 in 0.0 seconds (effective 615.7 Kbit/s)  
Hash of data verified.  
Leaving...  
Hard resetting via RTS pin...

31 ESP32 Wrover Module on COM7

Before running the program, please modify your router's name and password in the box shown in the illustration above to make sure that your Sketch can compile and work successfully.

Compile and upload codes to ESP32, open the serial monitor and set the baud rate to 115200, and the serial monitor will print out a network link address.

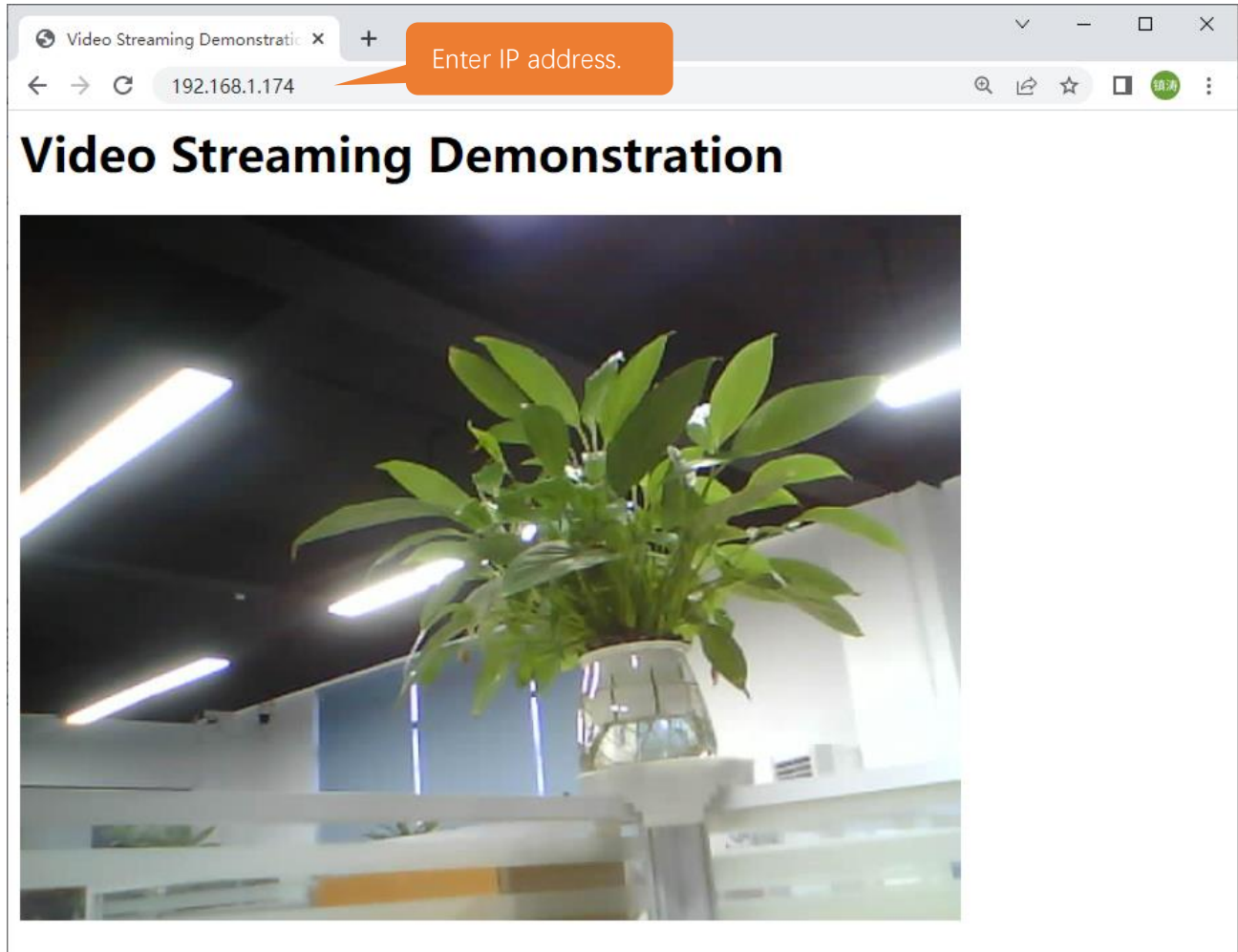


```
COM7
.....
WiFi connected
Camera Ready! Use 'http://192.168.1.174' to connect
ets Jun  8 2016 00:22:57
```

If your ESP32 has been in the process of connecting to router, but the information above has not been printed out, please re-check whether the router name and password have been entered correctly and press the reset key on ESP32-WROVER to wait for a successful connection prompt.

Open a web browser, enter the IP address printed by the serial monitor in the address bar, and access it. Taking the Google browser as an example, here's what the browser prints out after successful access to ESP32's IP.

The effect is shown in the image below.



The following is the main program code. You need include other code files in the same folder when write your own code.

```
1  #include "esp_camera.h"
2  #include <WiFi.h>
3  //
4  // WARNING!!! PSRAM IC required for UXGA resolution and high JPEG quality
5  //           Ensure ESP32 Wrover Module or other board with PSRAM is selected
6  //           Partial images will be transmitted if image exceeds buffer size
7  //
8
9  // Select camera model
10 #define CAMERA_MODEL_WROVER_KIT // Has PSRAM
11
```

Any concerns? ✉ [support@freenove.com](mailto:support@freenove.com)

```
12 #include "camera_pins.h"
13
14 const char* ssid      = "*****"; //input your wifi name
15 const char* password = "*****"; //input your wifi passwords
16
17 void startCameraServer();
18
19 void setup() {
20     Serial.begin(115200);
21     Serial.setDebugOutput(true);
22     Serial.println();
23
24     camera_config_t config;
25     config.ledc_channel = LEDC_CHANNEL_0;
26     config.ledc_timer = LEDC_TIMER_0;
27     config.pin_d0 = Y2_GPIO_NUM;
28     config.pin_d1 = Y3_GPIO_NUM;
29     config.pin_d2 = Y4_GPIO_NUM;
30     config.pin_d3 = Y5_GPIO_NUM;
31     config.pin_d4 = Y6_GPIO_NUM;
32     config.pin_d5 = Y7_GPIO_NUM;
33     config.pin_d6 = Y8_GPIO_NUM;
34     config.pin_d7 = Y9_GPIO_NUM;
35     config.pin_xclk = XCLK_GPIO_NUM;
36     config.pin_pclk = PCLK_GPIO_NUM;
37     config.pin_vsync = VSYNC_GPIO_NUM;
38     config.pin_href = HREF_GPIO_NUM;
39     config.pin_sccb_sda = SIOD_GPIO_NUM;
40     config.pin_sccb_scl = SIOC_GPIO_NUM;
41     config.pin_pwdn = PWDN_GPIO_NUM;
42     config.pin_reset = RESET_GPIO_NUM;
43     config.xclk_freq_hz = 20000000;
44     config.pixel_format = PIXFORMAT_JPEG;
45
46     // if PSRAM IC present, init with UXGA resolution and higher JPEG quality
47     //                               for larger pre-allocated frame buffer.
48     if(psramFound()) {
49         config.frame_size = FRAMESIZE_VGA;
50         config.jpeg_quality = 10;
51         config.fb_count = 2;
52     } else {
53         config.frame_size = FRAMESIZE_HVGA;
54         config.jpeg_quality = 12;
55         config.fb_count = 1;
```

```
56     }
57
58     // camera init
59     esp_err_t err = esp_camera_init(&config);
60     if (err != ESP_OK) {
61         Serial.printf("Camera init failed with error 0x%x", err);
62         return;
63     }
64
65     sensor_t * s = esp_camera_sensor_get();
66     // drop down frame size for higher initial frame rate
67     s->set_framesize(s, FRAMESIZE_QVGA);
68
69     WiFi.begin(ssid, password);
70
71     while (WiFi.status() != WL_CONNECTED) {
72         delay(500);
73         Serial.print(".");
74     }
75     Serial.println("");
76     Serial.println("WiFi connected");
77
78     startCameraServer();
79
80     Serial.print("Camera Ready! Use 'http://");
81     Serial.print(WiFi.localIP());
82     Serial.println("' to connect");
83 }
84
85 void loop() {
86     // put your main code here, to run repeatedly:
87     delay(10000);
88 }
```

Configure parameters including interface pins of the camera. Note: It is generally not recommended to change them.

```
24 camera_config_t config;
25 config.ledc_channel = LEDC_CHANNEL_0;
26 config.ledc_timer = LEDC_TIMER_0;
27 config.pin_d0 = Y2_GPIO_NUM;
28 config.pin_d1 = Y3_GPIO_NUM;
29 config.pin_d2 = Y4_GPIO_NUM;
30 config.pin_d3 = Y5_GPIO_NUM;
31 config.pin_d4 = Y6_GPIO_NUM;
```

```

32 config.pin_d5 = Y7_GPIO_NUM;
33 config.pin_d6 = Y8_GPIO_NUM;
34 config.pin_d7 = Y9_GPIO_NUM;
35 config.pin_xclk = XCLK_GPIO_NUM;
36 config.pin_pclk = PCLK_GPIO_NUM;
37 config.pin_vsync = VSYNC_GPIO_NUM;
38 config.pin_href = HREF_GPIO_NUM;
39 config.pin_sccb_sda = SIOD_GPIO_NUM;
40 config.pin_sccb_scl = SIOC_GPIO_NUM;
41 config.pin_pwn = PWDN_GPIO_NUM;
42 config.pin_reset = RESET_GPIO_NUM;
43 config.xclk_freq_hz = 20000000;
44 config.pixel_format = PIXFORMAT_JPEG;

```

ESP32 connects to the router and prints a successful connection prompt. If it has not been successfully connected, press the reset key on the ESP32-WROVER.

```

69 WiFi.begin(ssid, password);
70
71 while (WiFi.status() != WL_CONNECTED) {
72     delay(500);
73     Serial.print(".");
74 }
75 Serial.println("");
76 Serial.println("WiFi connected");

```

Open the video streams server function of the camera and print its IP address via serial port.

```

78 startCameraServer();
79
80 Serial.print("Camera Ready! Use 'http://");
81 Serial.print(WiFi.localIP());
82 Serial.println("' to connect");

```

## 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](mailto:support@freenove.com)

We will check and correct it as soon as possible.

If you want learn more about ESP32, you view our ultimate tutorial:

[https://github.com/Freenove/Freenove\\_ESP32\\_WROVER\\_Board/archive/master.zip](https://github.com/Freenove/Freenove_ESP32_WROVER_Board/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.