Important Information

Thank you for choosing Freenove products!

Getting Started

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

Get Support and Offer Input

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

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

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

support@freenove.com

Safety and Precautions

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

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

About Freenove

Freenove provides open source electronic products and services worldwide.

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

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

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

http://www.freenove.com

sale@freenove.com

Copyright

FREENOVE™

Contents

Preface

ESP8266 is a micro control unit with integrated Wi-Fi launched by Espressif, which features strong properties and integrates rich peripherals. It can be designed and studied as an ordinary Single Chip Micyoco(SCM) chip, or connected to the Internet and used as an Internet of Things device.

ESP8266 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 ESP8266 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, ESP8266 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.

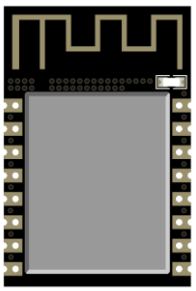
We divide each project into four parts, namely Component List, Component Knowledge, Circuit and Code. Component List helps you to prepare material for the experiment more quickly. Component Knowledge allows you to quickly understand new electronic modules or components, while Circuit helps you understand the operating principle of the circuit. And Code allows you to easily master the use of ESP8266 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**

ESP8266

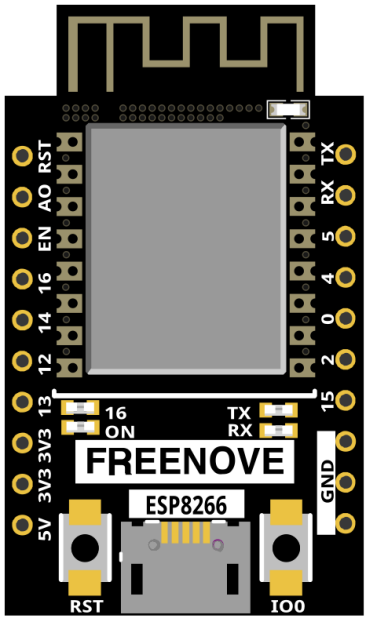
ESP8266 has PCB on-board antenna. The PCB on-board antenna is an integrated antenna in the chip module itself, so it is convenient to carry and design.

PCB on-board antenna

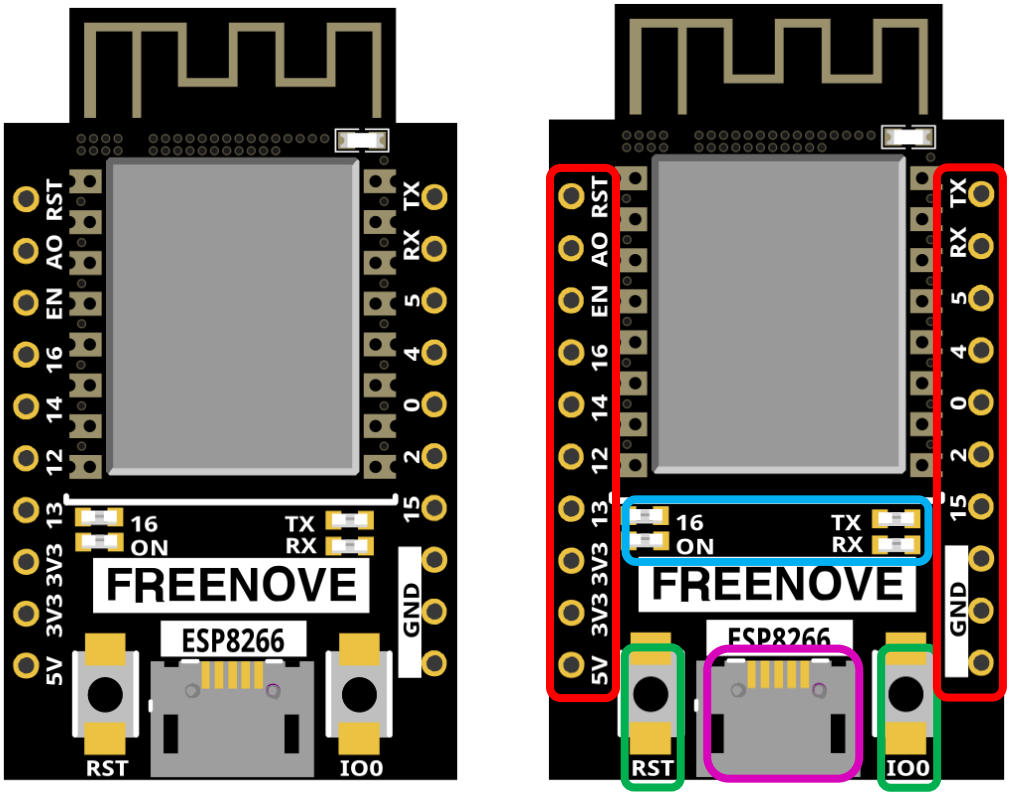


In this tutorial, the ESP8266 development board is designed based on the PCB on-board antenna-packaged ESP8266 module. The following tutorials will be based on the ESP8266 development board.

ESP8266 development board



The hardware interfaces of ESP8266 are distributed as follows:



Compare the left and right images. We've boxed off the resources on the ESP8266 in different colors to facilitate your understanding of the ESP8266 development board.

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

NO.	Pin Name	Functional Description
1	RST	Reset Pin, Active Low
2	ADC	AD conversion, Input voltage range 0~3.3V, the value range is 0~1024.
3	EN	Chip Enabled Pin, Active High
4	IO16	Connect with RST pin to wake up Deep Slee
5	IO14	GPIO14; HSPI_CLK
6	IO12	GPIO12; HSPI_MISO
7	IO13	GPIO13; HSPI_MOSI; UART0_CTS
8	VCC	Module power supply pin, Voltage 3.0V ~ 3.6V
9	GND	GND
10	IO15	GPIO15; MTDO; HSPICS; UART0
11	IO2	GPIO2; UART1_TXD
12	IO0	GPIO0; UART1_RXD
13	IO4	GPIO4
14	IO5	GPIO5;IR_R
15	RXD	UART0_RXD; GPIO3
16	TXD	UART0_TXD; GPIO1

Description of the ESP8266 series module boot mode:

Mode	CH_PD(EN)	RST	GPIO15	GPIO0	GPIO2	TXD0
Download mode	high	high	low	low	high	high
Running mode	high	high	low	high	high	high

Notes: Some of the pins inside the module have been pulled or pulled down.

For more information, please visit:

If you want to learn more about this, you can read the following files:

“Freenove_ESP8266_Board/Datasheet/esp-12s_datasheet_en.pdf”

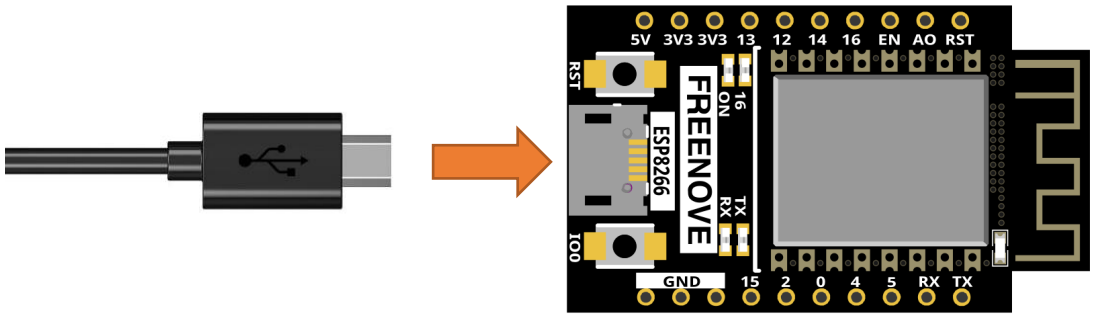
CH340

ESP8266 uses CH340 to download codes. So before using it, we need to install CH340 driver in our computers.

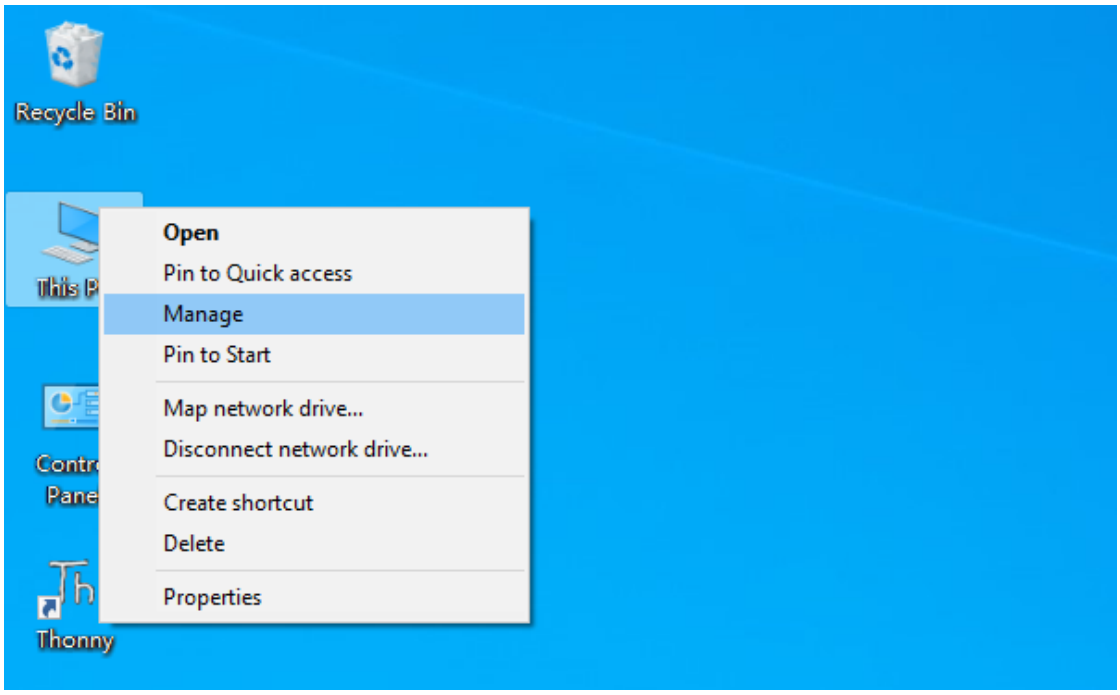
Windows

Check whether CH340 has been installed

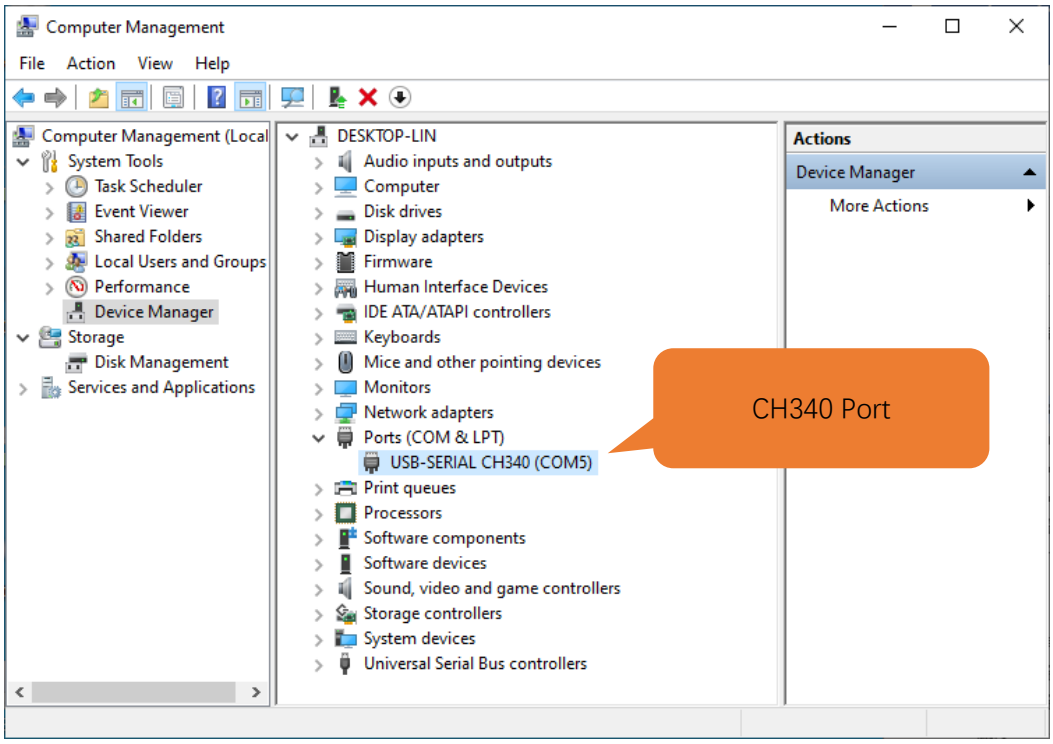
1. Connect your computer and ESP8266 with a USB cable.



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

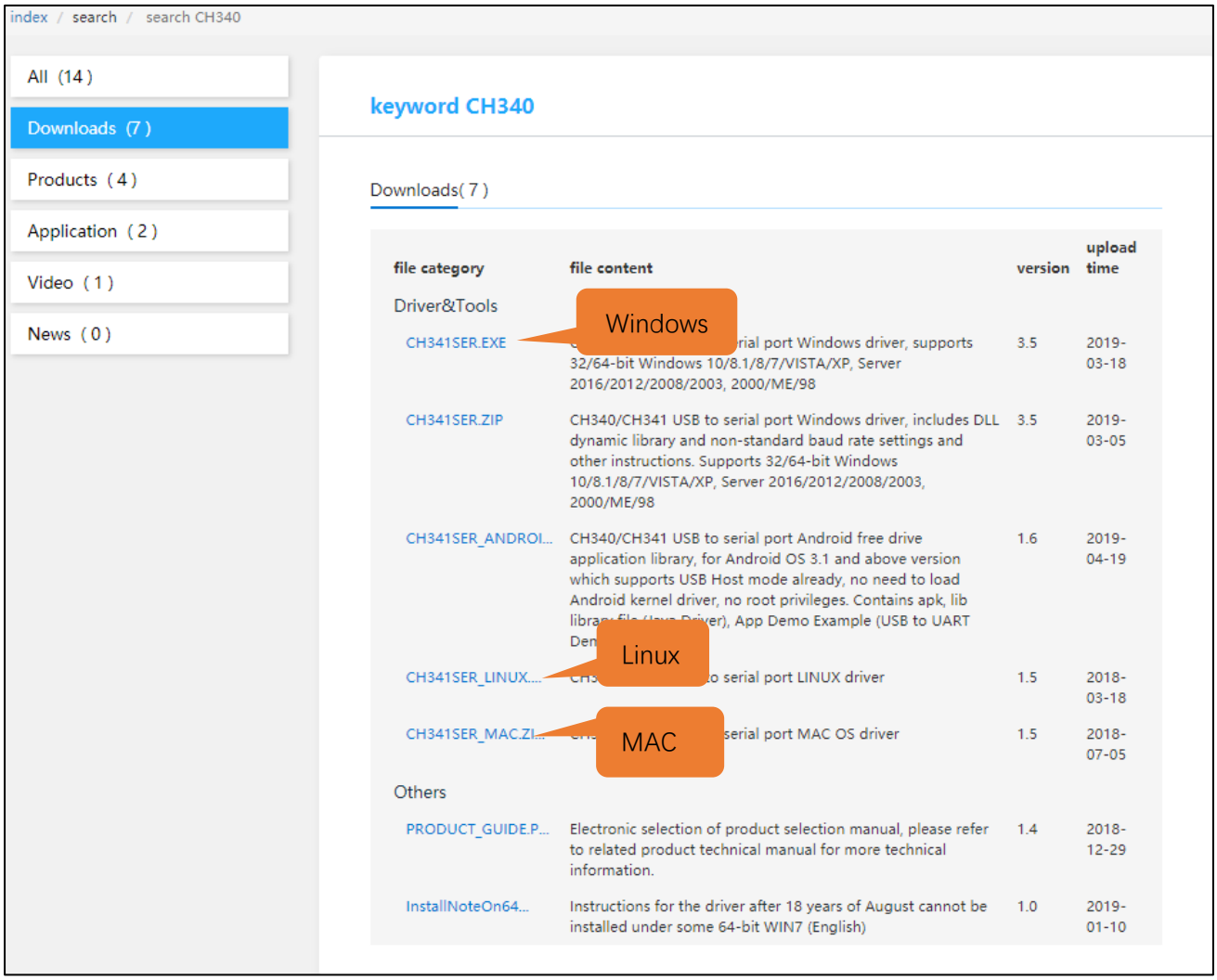


3. Click "Device Manager". If your computer has installed CH340, you can see"USB-SERIAL CH340 (COMx)". And you can click here to move to the next step.

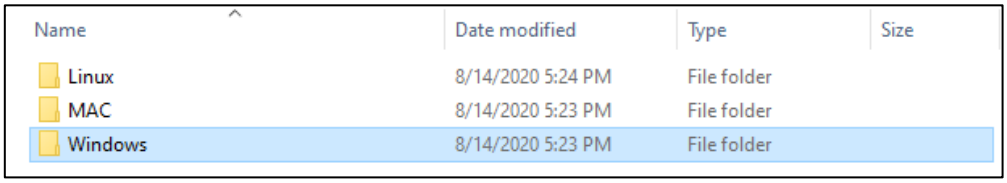


Installing CH340

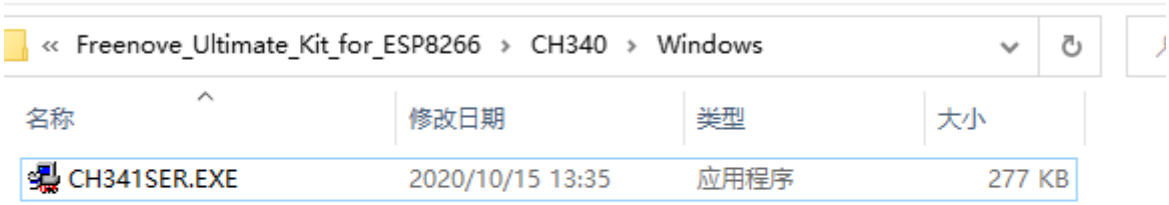
1. First, download CH340 driver, click http://www.wch-ic.com/search?q=CH340&t=downloads to download the appropriate one based on your operating system.



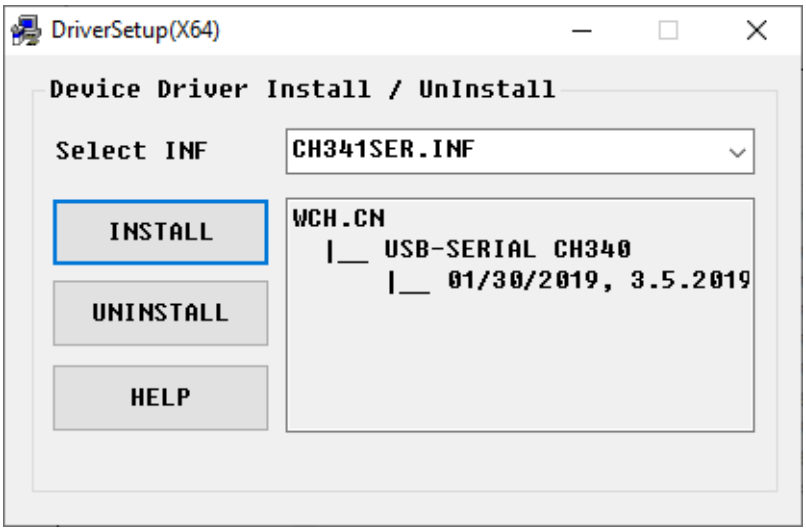
If you would not like to download the installation package, you can open **"Freenove_ESP8266_Board/CH340"**, we have prepared the installation package.



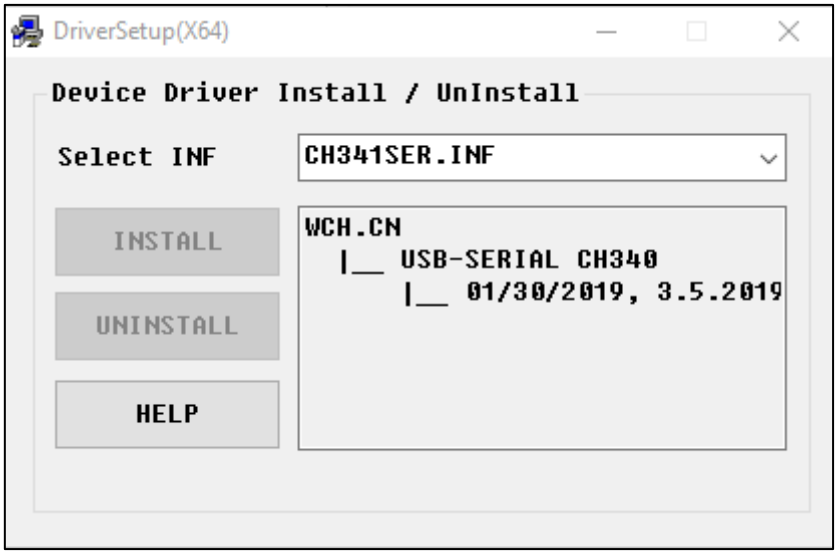
2. Open the folder "**Freenove_ESP8266_Board/CH340/Windows/**"



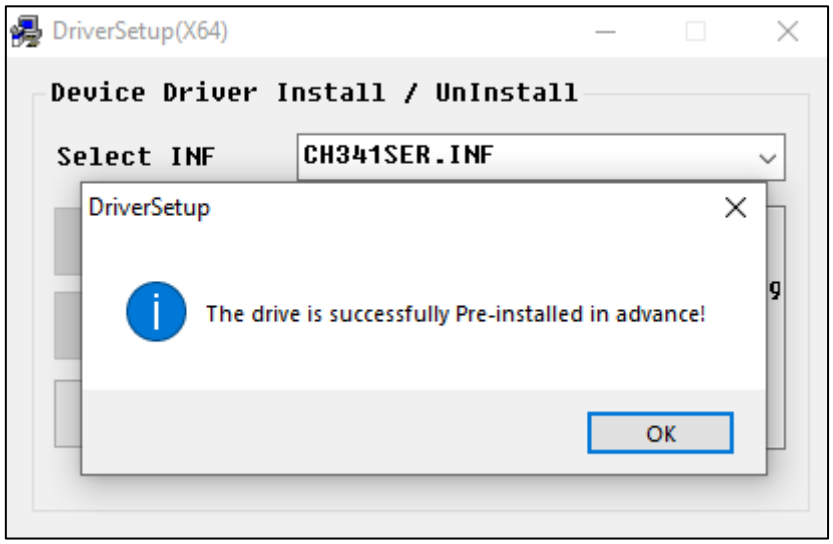
3. Double click "**CH341SER.EXE**".



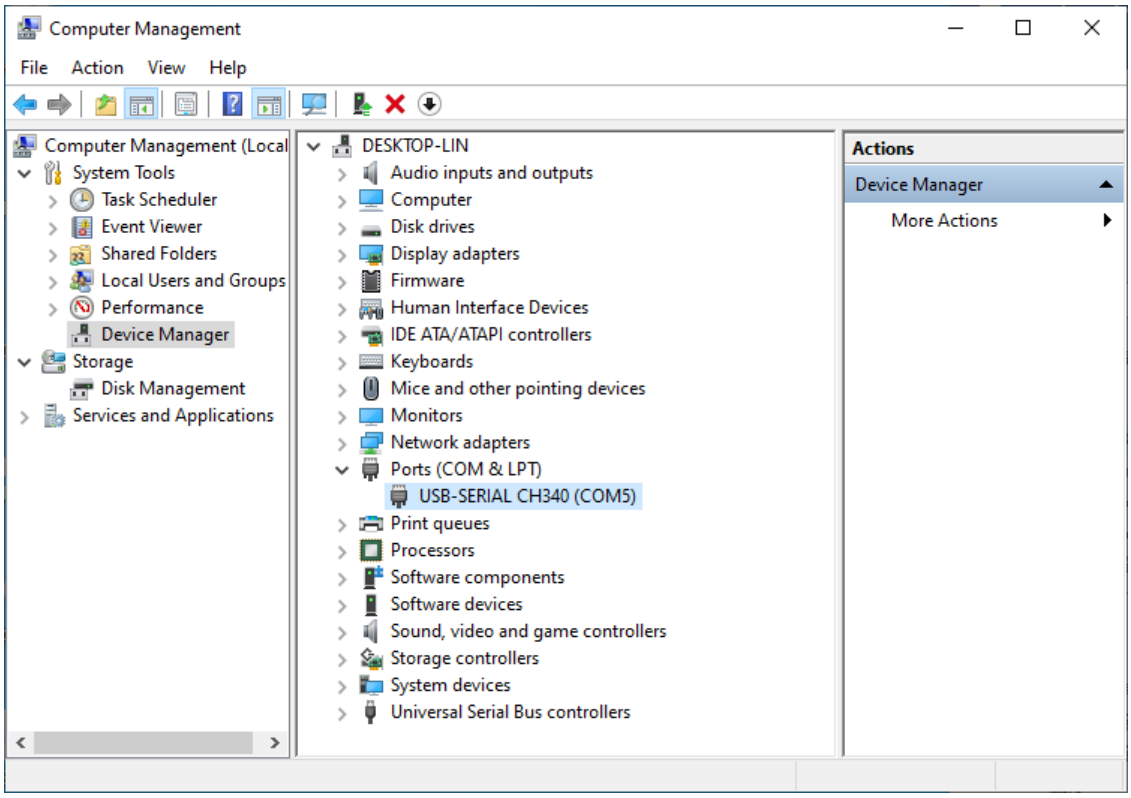
4. Click “INSTALL” and wait for the installation to complete.



5. Install successfully. Close all interfaces.



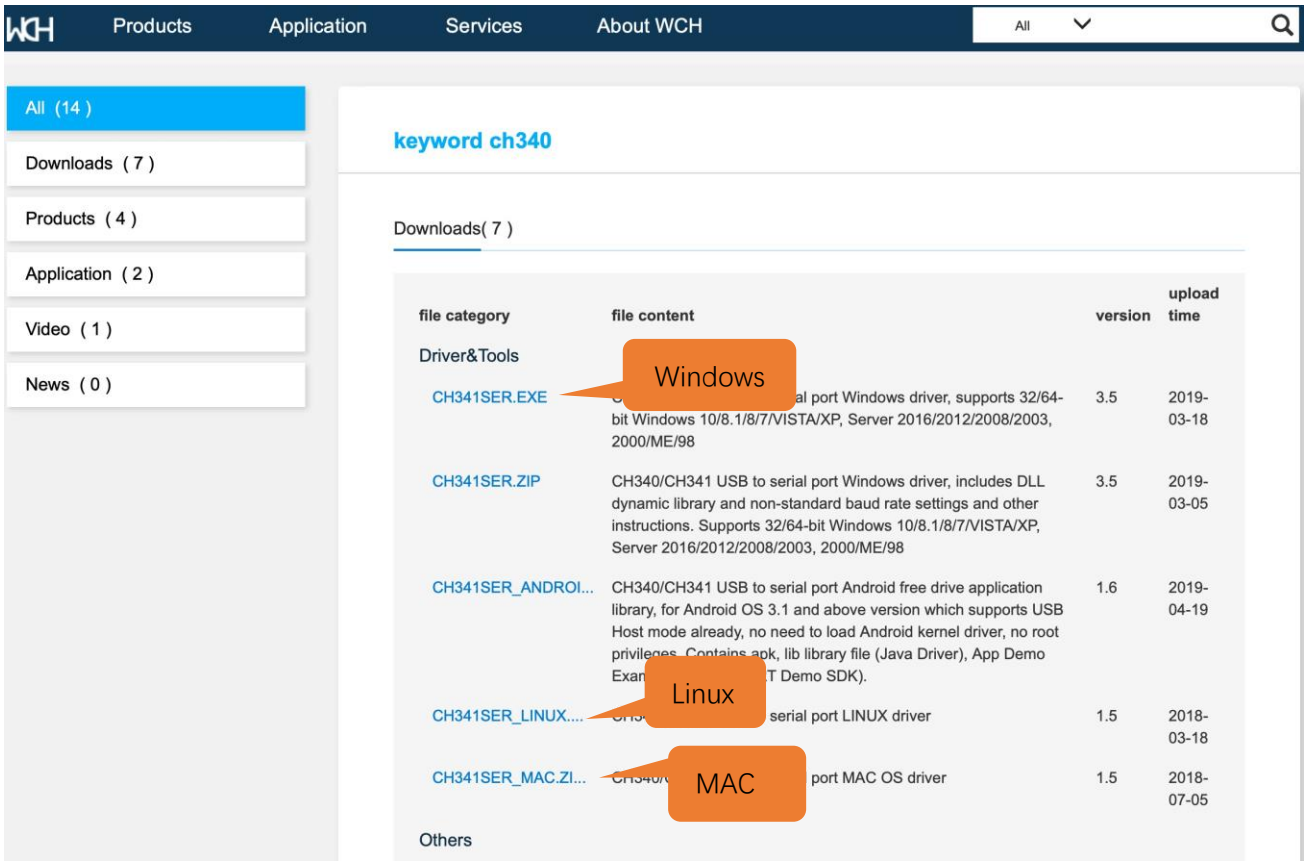
6. When ESP8266 is connected to computer, select “This PC”, right-click to select “Manage” and click “Device Manager” in the newly pop-up dialog box, and you can see the following interface.



7. So far, CH340 has been installed successfully. Close all dialog boxes.

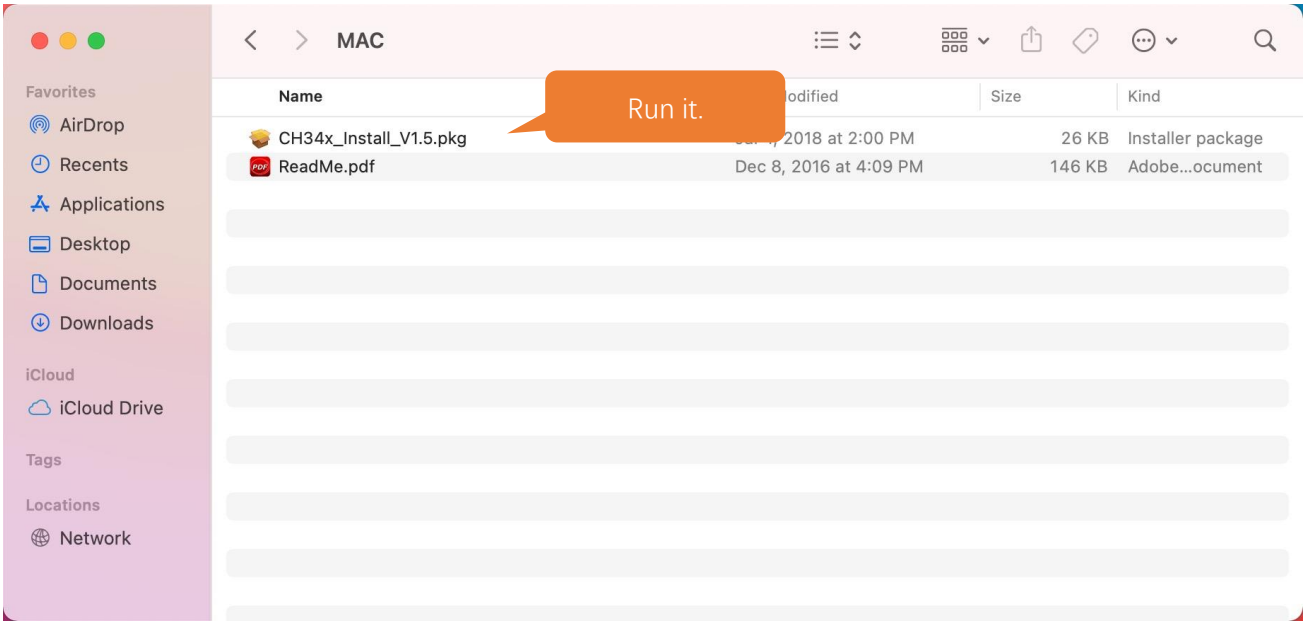
MAC

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.

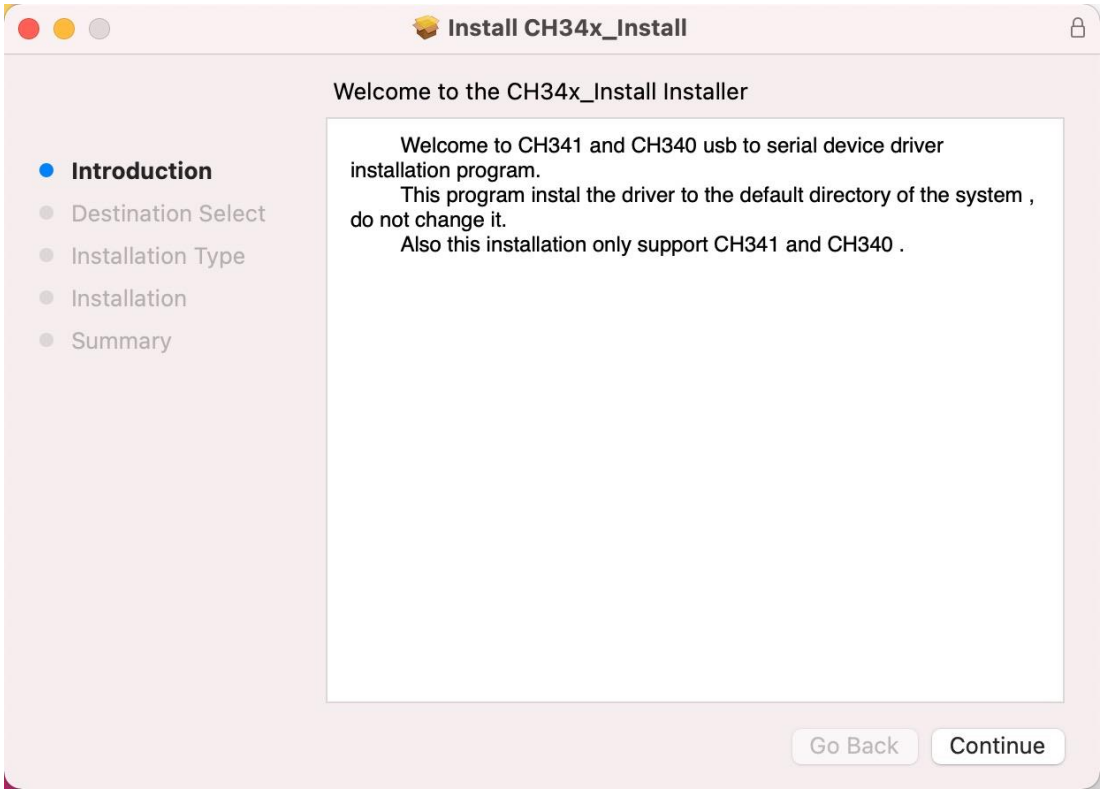


If you would not like to download the installation package, you can open **"Freenove_ESP8266_Board/CH340"**, we have prepared the installation package.

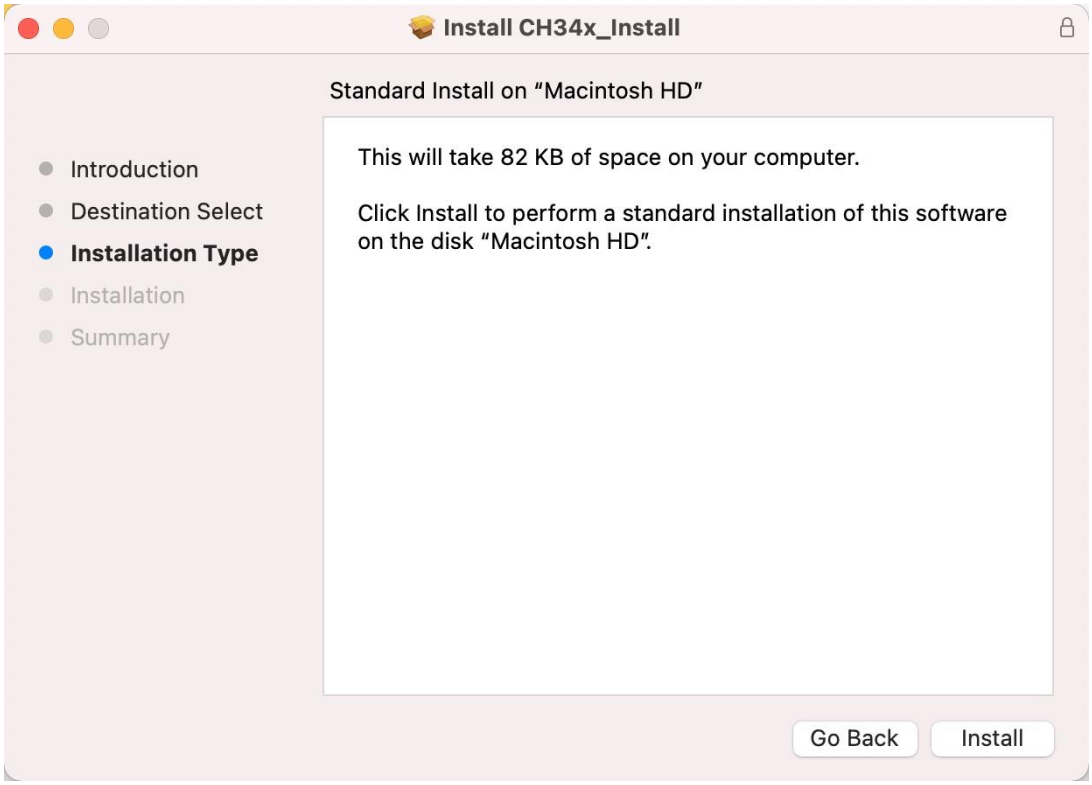
Second, open the folder "**Freenove_ESP8266_Board/CH340/MAC/**"



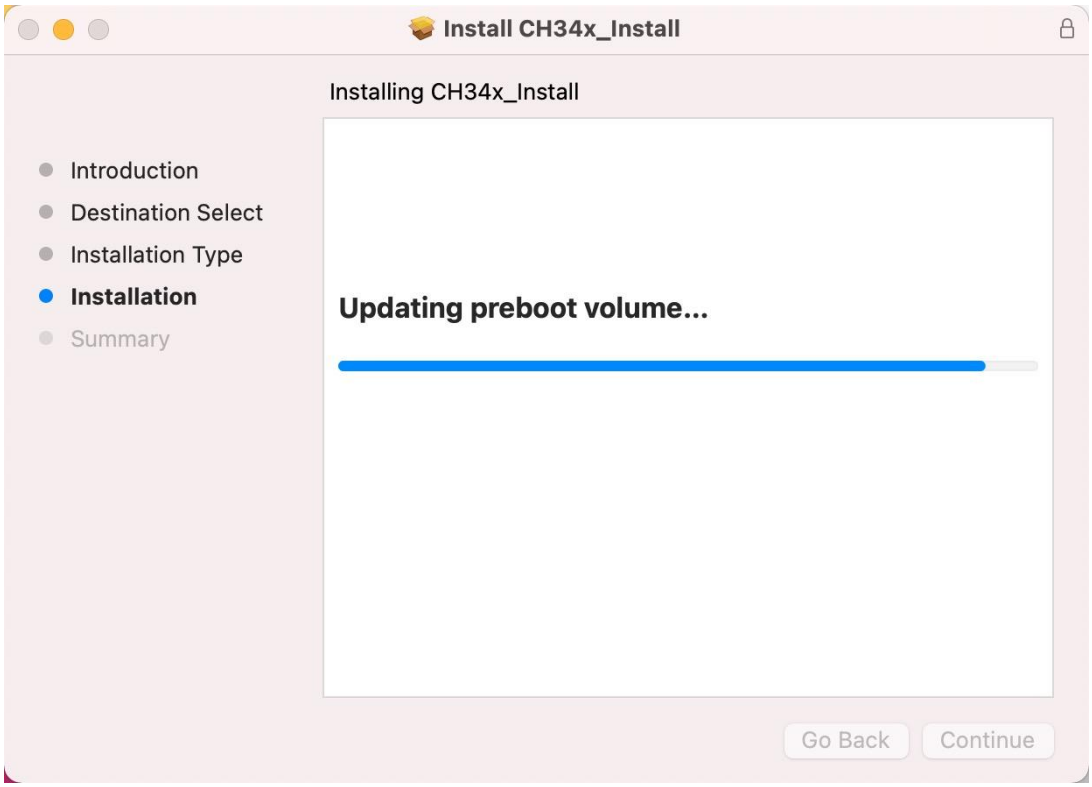
Third, click Continue.



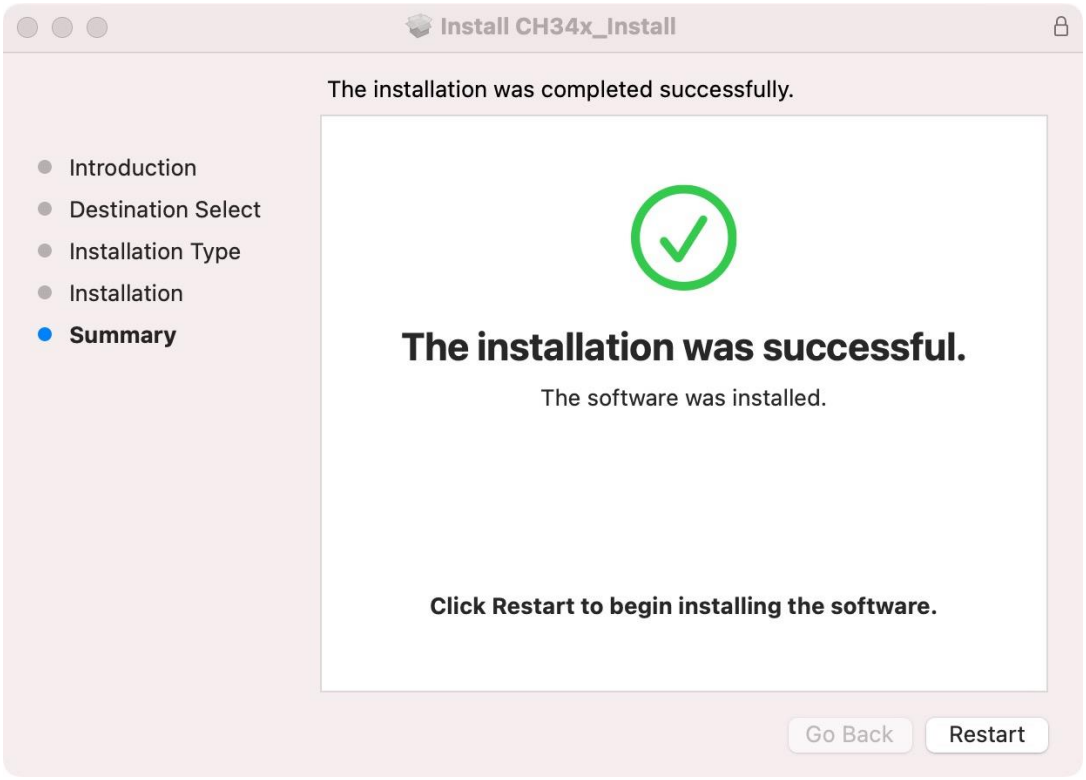
Fourth, click Install.



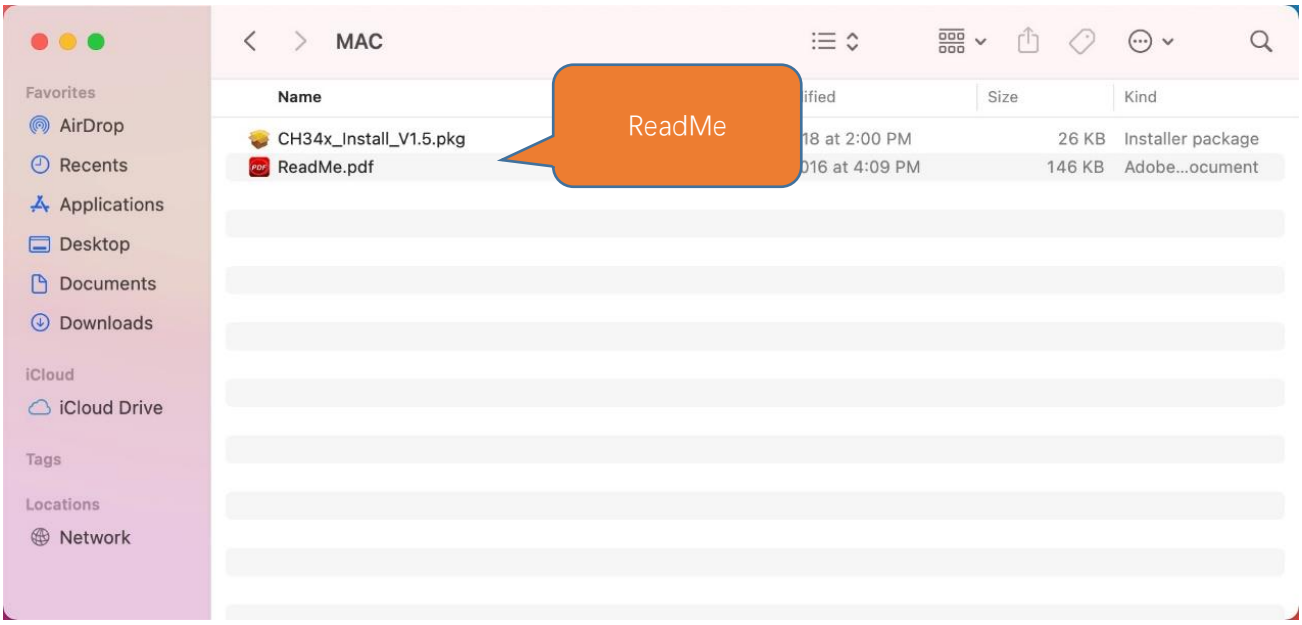
Then, waiting Finsh.



Finally, restart your PC.

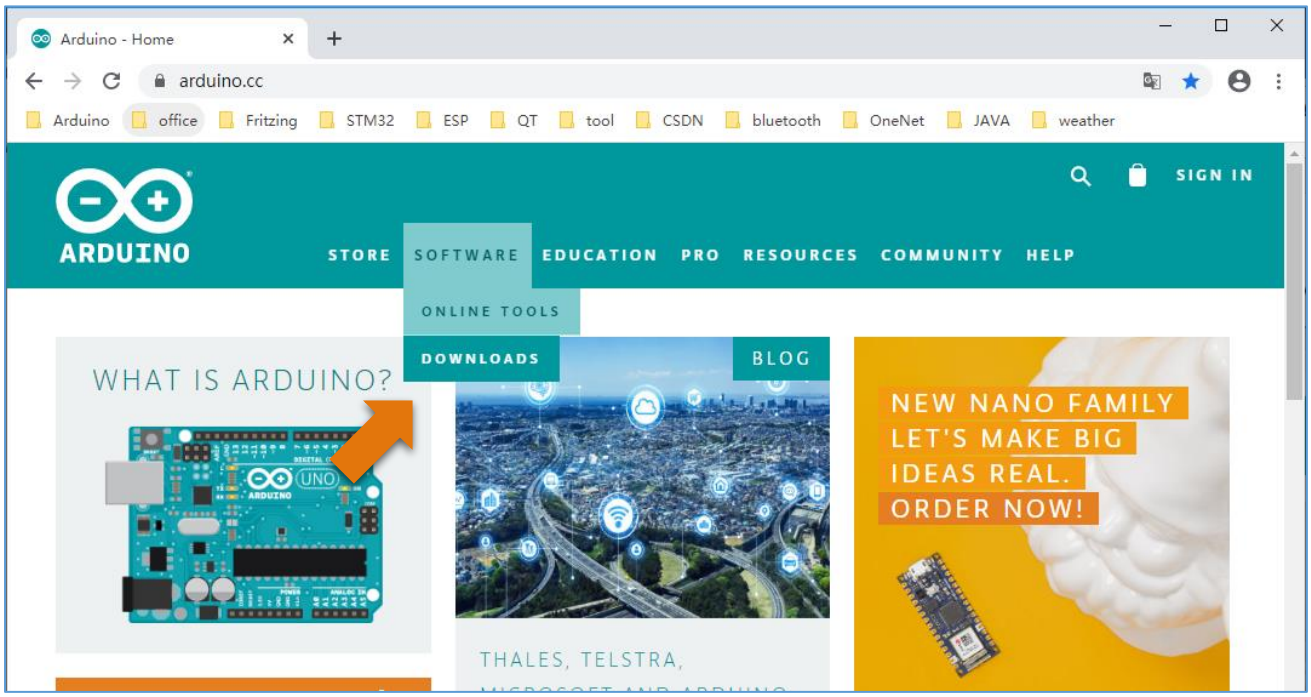


If you still haven't installed the CH340 by following the steps above, you can view readme.pdf to install it.

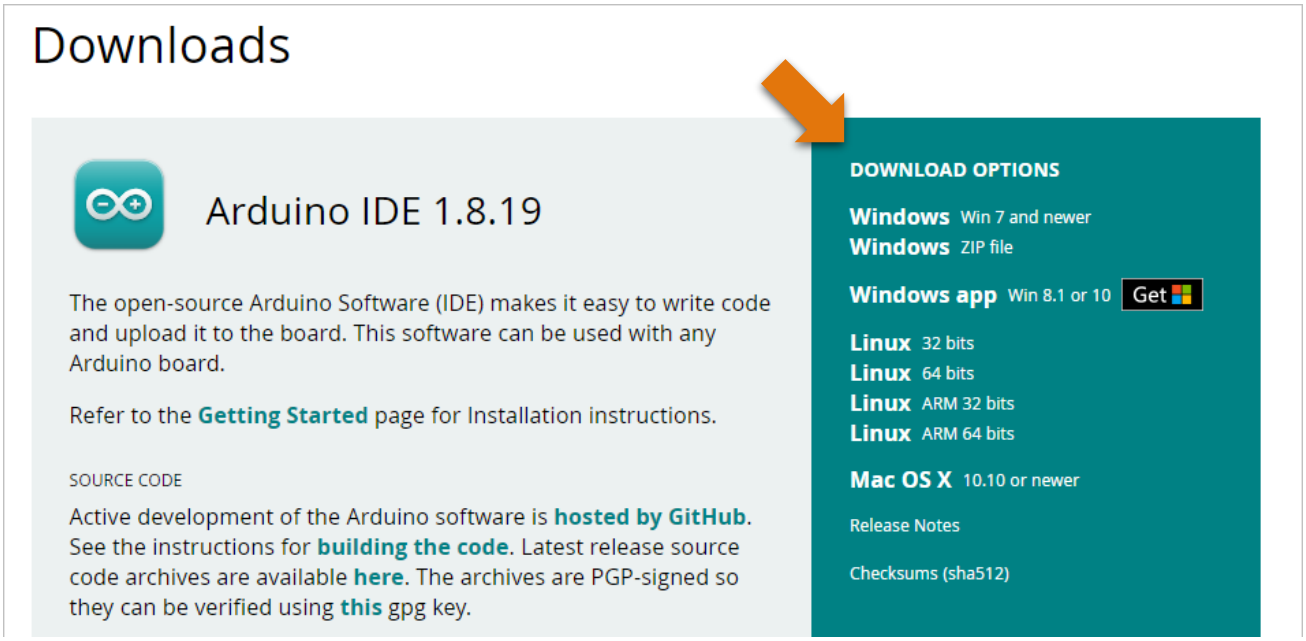


Programming Software

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

New
Create a new sketch.

Open
Present a menu of all the sketches in your sketchbook. Clicking one will open it within the current window and overwrite its content.

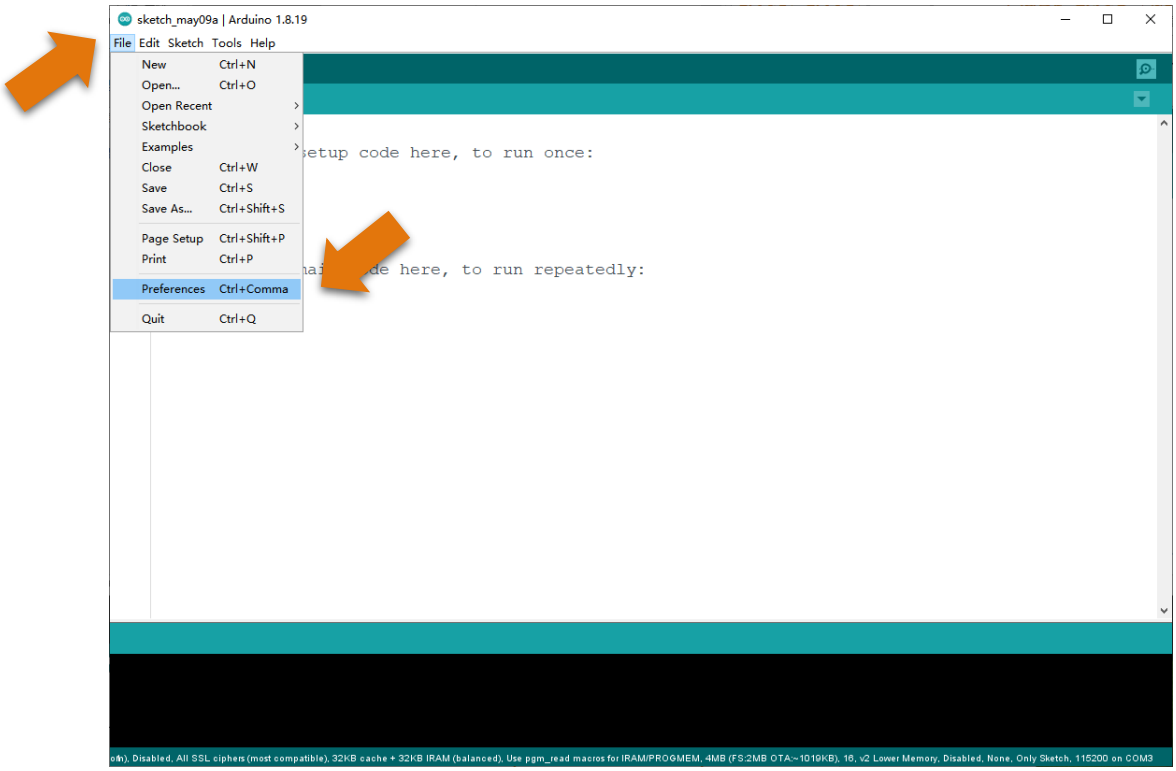
Save
Save your sketch.

Serial Monitor
Open the serial monitor.

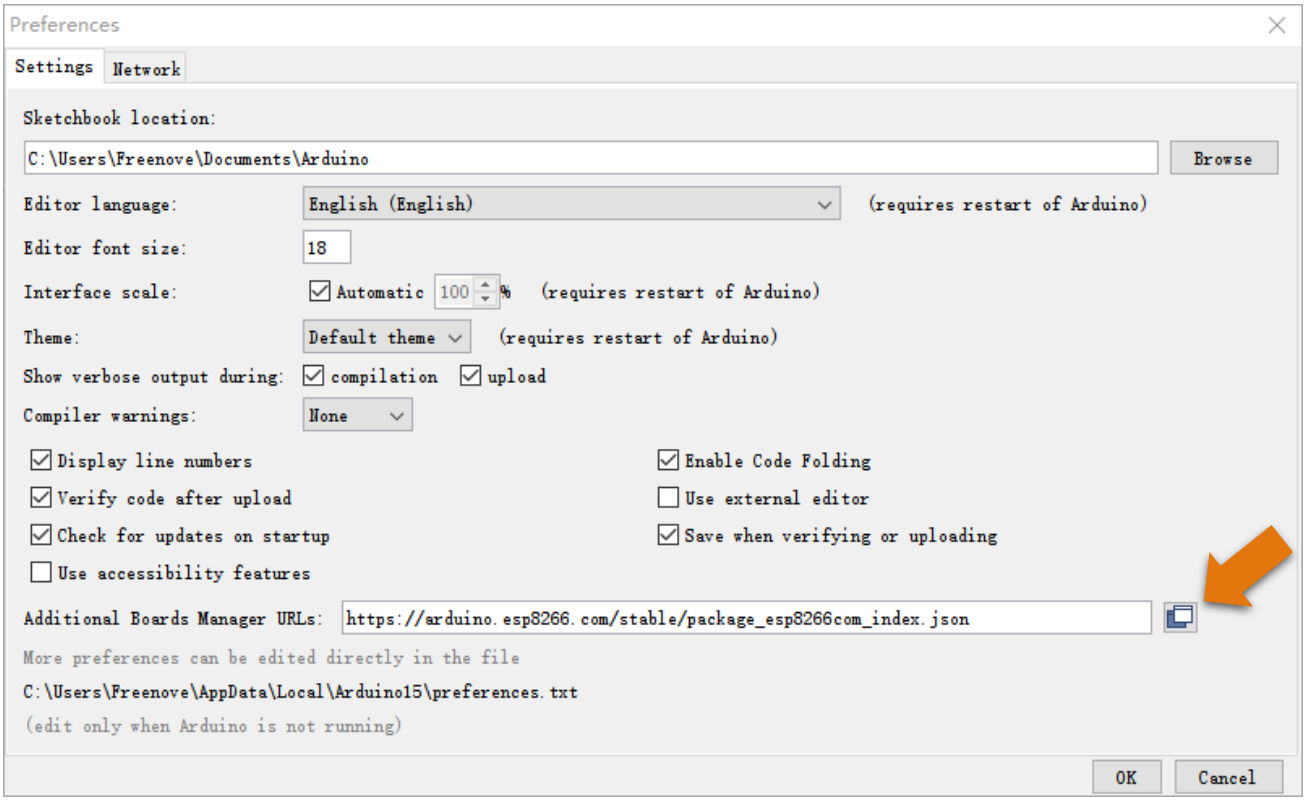
Additional commands are found within the five menus: File, Edit, Sketch, Tools, Help. The menus are context sensitive, which means only those items relevant to the work currently being carried out are available.

Environment Configuration

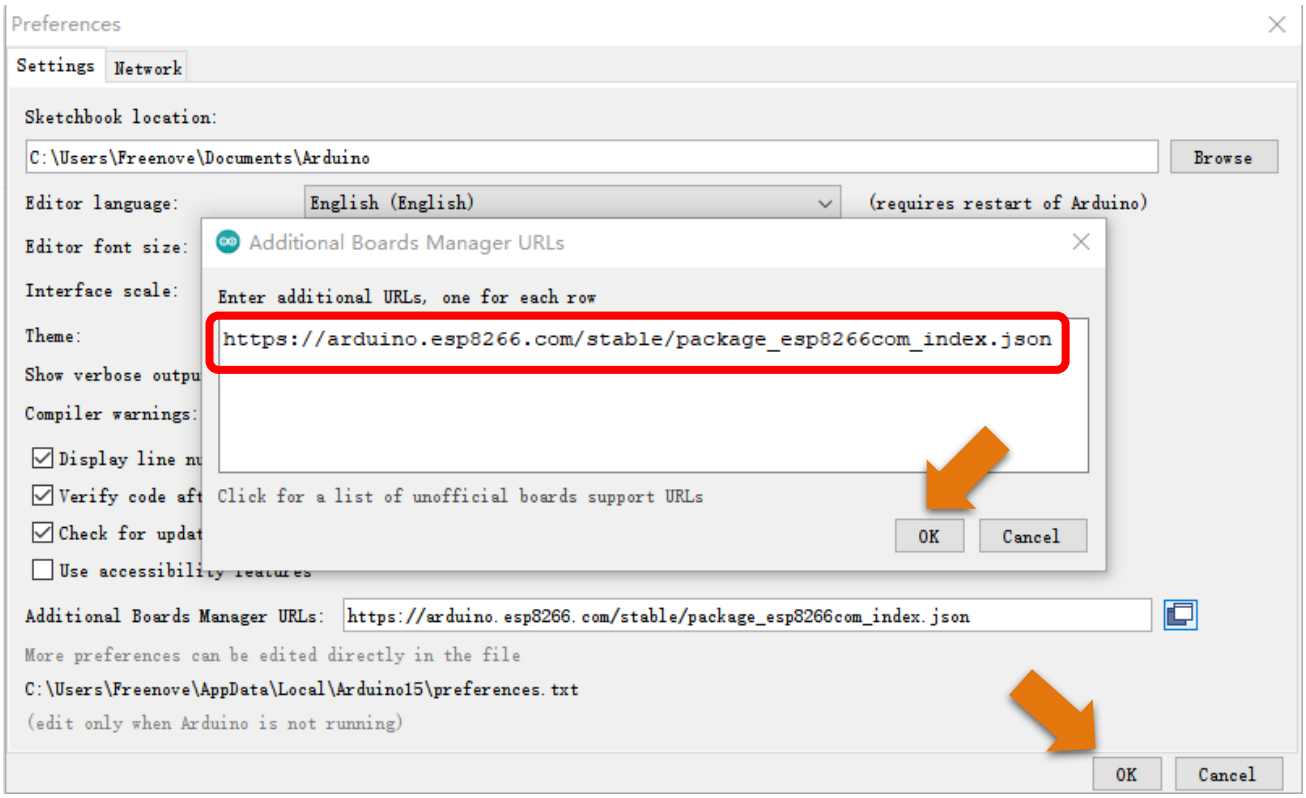
First, open the software platform arduino, and then click File in Menus and select Preferences.



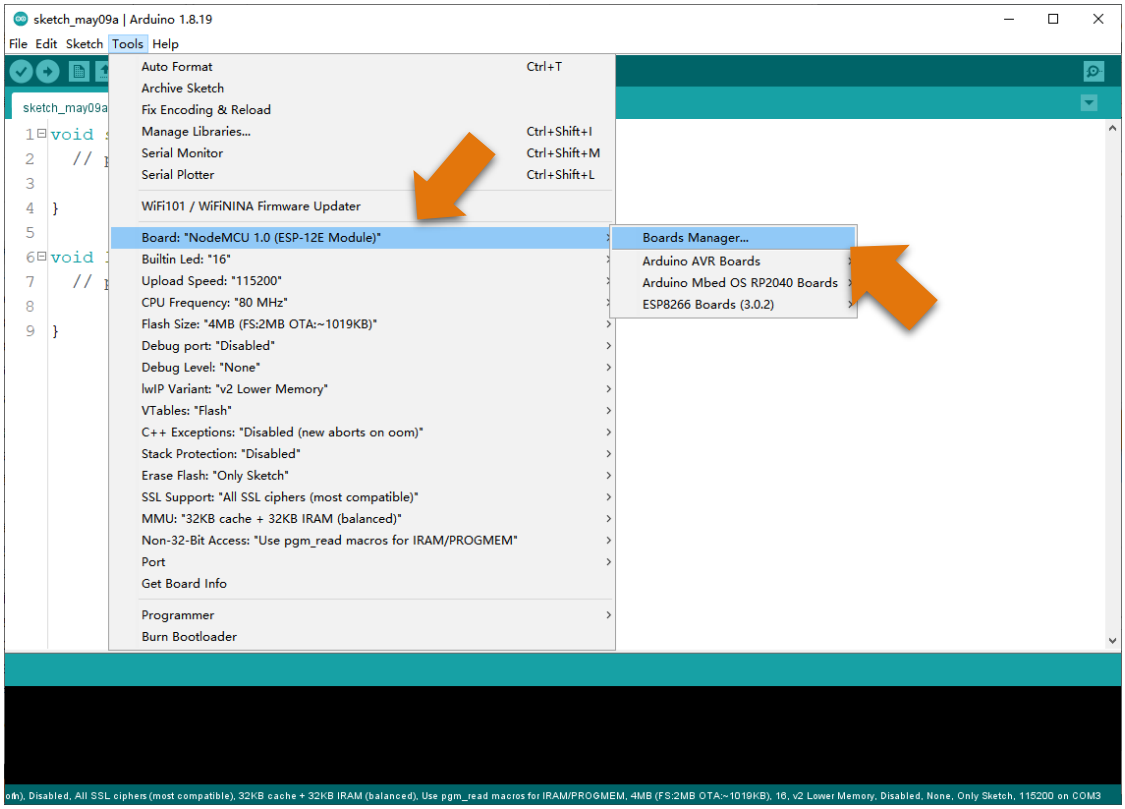
Second, click on the symbol behind "Additional Boards Manager URLs"



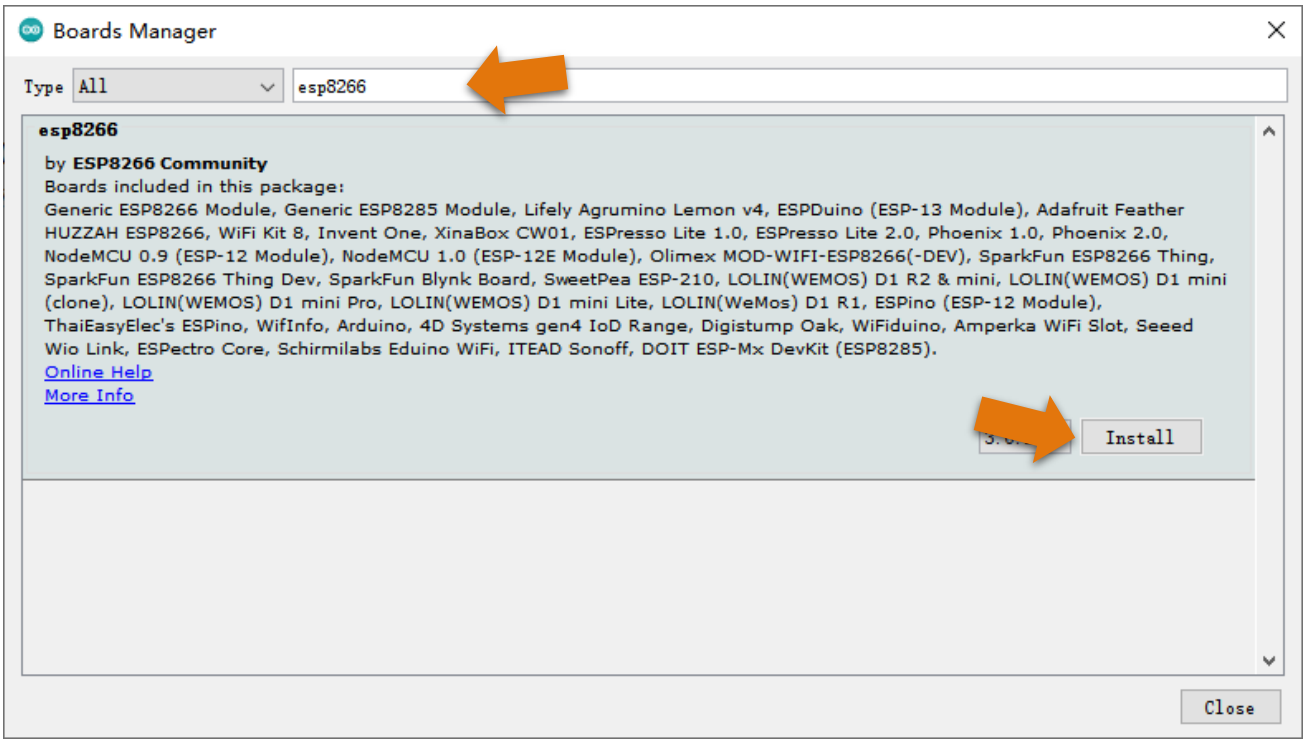
Third, fill in https://arduino.esp8266.com/stable/package_esp8266com_index.json in the new window, click OK, and click OK on the Preferences window again.



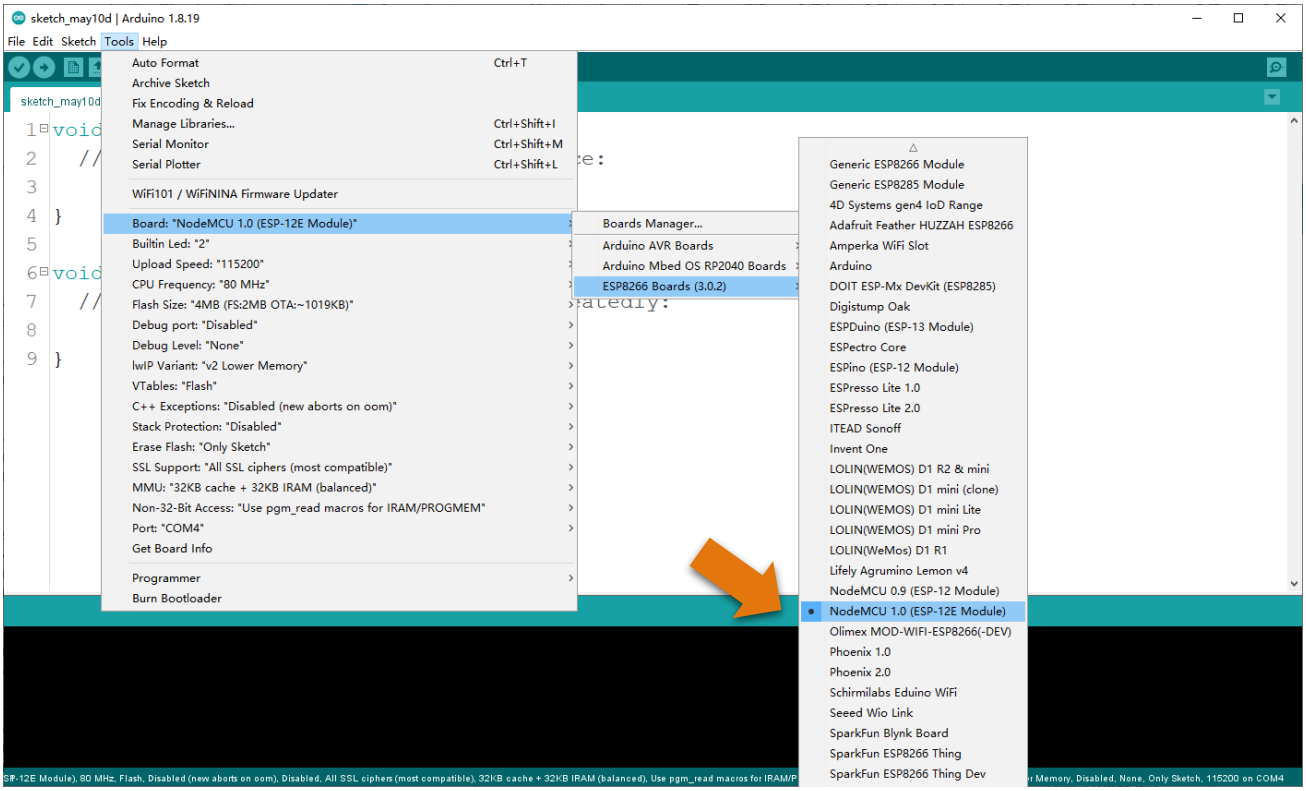
Fourth, click Tools in Menus, select Board:"ArduinoUno", and then select "Boards Manager".



Fifth, input "esp8266" in the window below, and press Enter. click "Install" to install.

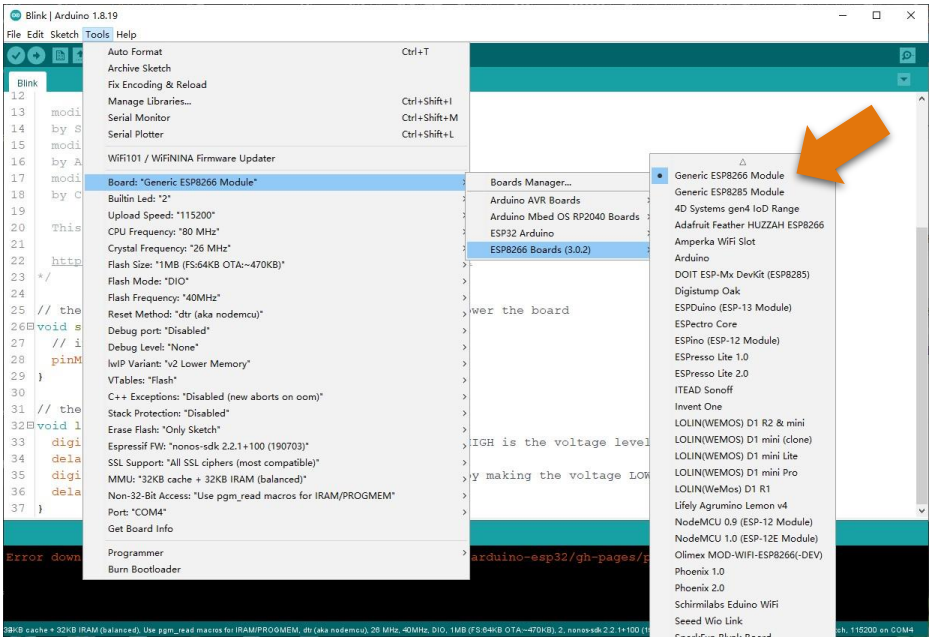


When finishing installation, click Tools in the Menus again and select Board: "NodeMCU 1.0(ESP-12E Module)", and then you can see information of ESP8266 click "NodeMCU 1.0(ESP-12E Module)" so that the ESP8266 programming development environment is configured.

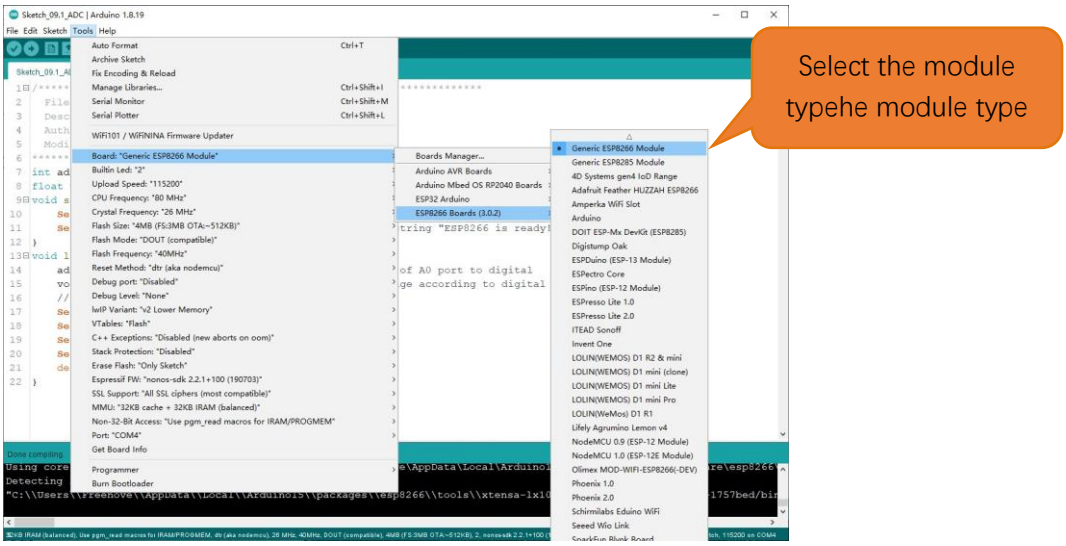


In our tutorial, we chose "NodeMCU 1.0(ESP-12E Module)" as the development board Module.This choice will facilitate learning and understanding of ESP8266.Of course, you can choose "Generic ESP8266 Module". Select "Generic ESP8266 Module" to apply to all Generic ESP8266 modules.Of course, this setup will have some more common configuration.

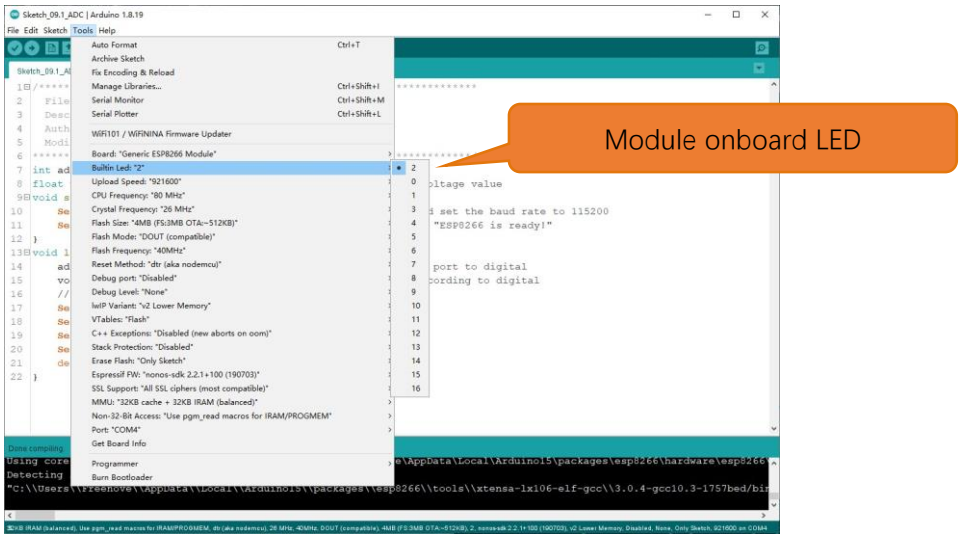
When you select "Generic ESP8266 Module", the interface is as follows:



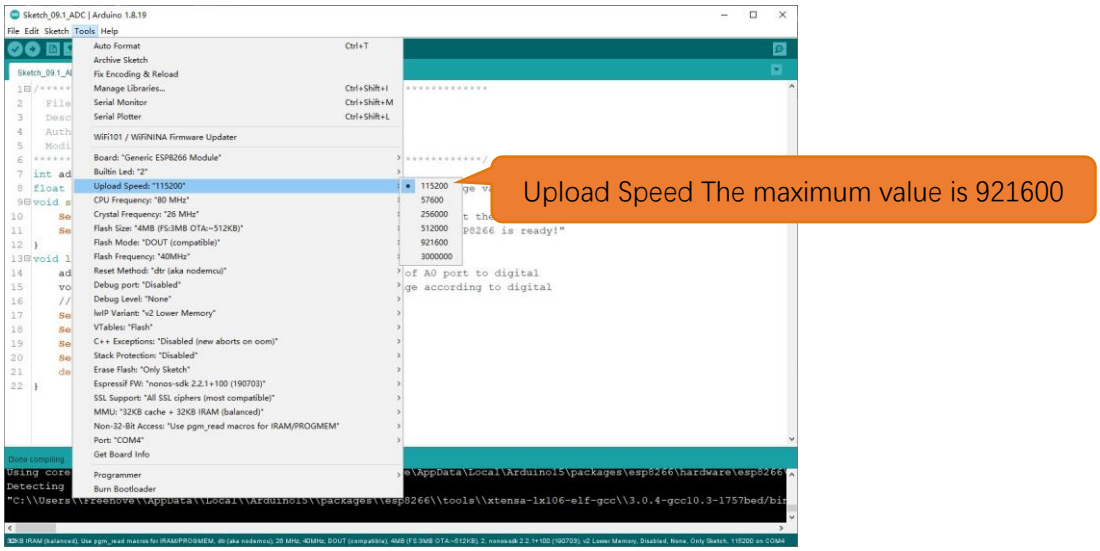
Select the module typehe module.Here you can choose the appropriate module based on your requirements.



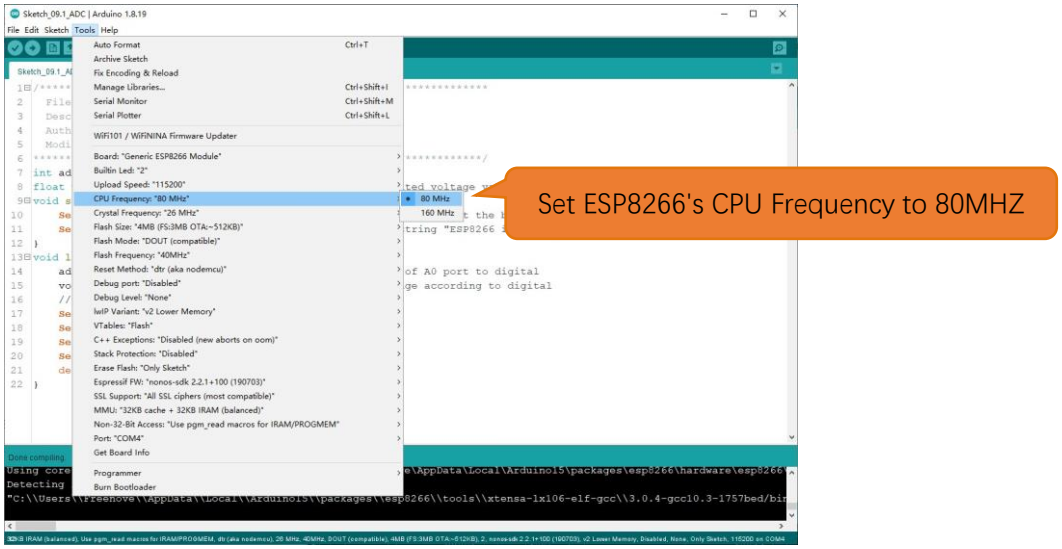
Module Onboard LED, in our ESP8266 development board, has an onboard LED of 2.



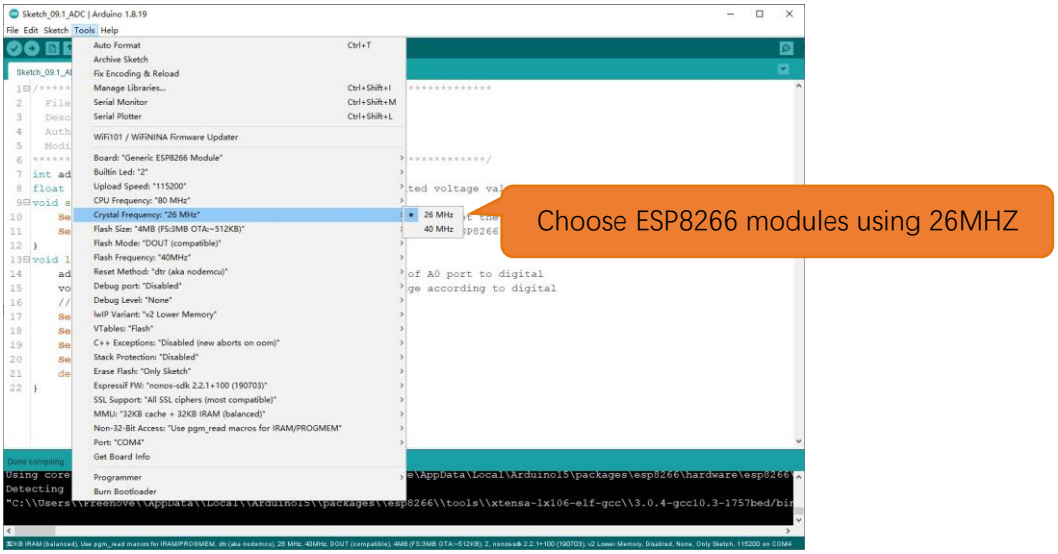
Upload Speed The maximum value is 921600.By default, Upload Speed is 115200.You can choose according to your needs.



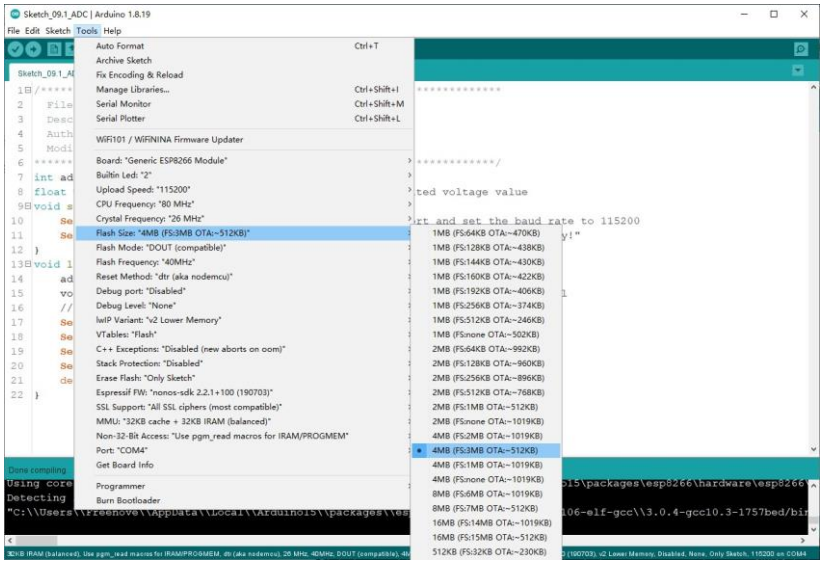
ESP8266's CPU frequency standard is 80MHz, which can be changed to 160MHz.



Most ESP8266 modules use 26 MHz crystals, but some have other values.



Choose the appropriate Flash size based on your ESP8266 module type. In our ESP8266 development board, we chose 4MB (FS: 3MB OTA: ~512KB).



Here we need to select Flash mode.On our ESP8266 development board, choose "DIO" mode or "DOUT" mode for better compatibility.If the ESP8266 module is abnormal, check whether the ESP8266 module works in the two modes.

Flash works in DOUT, DIO, QOUT, and QIO modes.

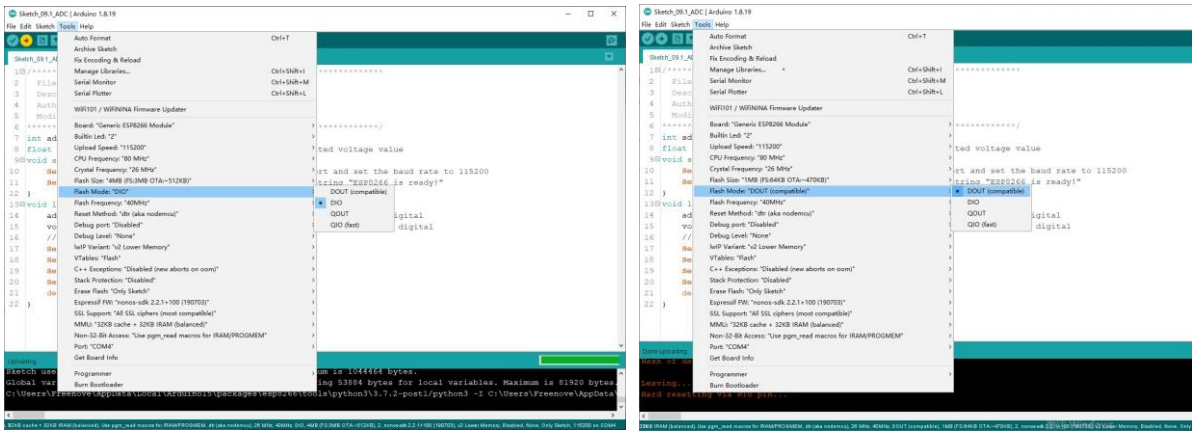
1.DOUT: Address is input in 1-line mode and data is output in 2-line mode.

2.DIO: Address is input in 2-line mode and data is output in 2-line mode.

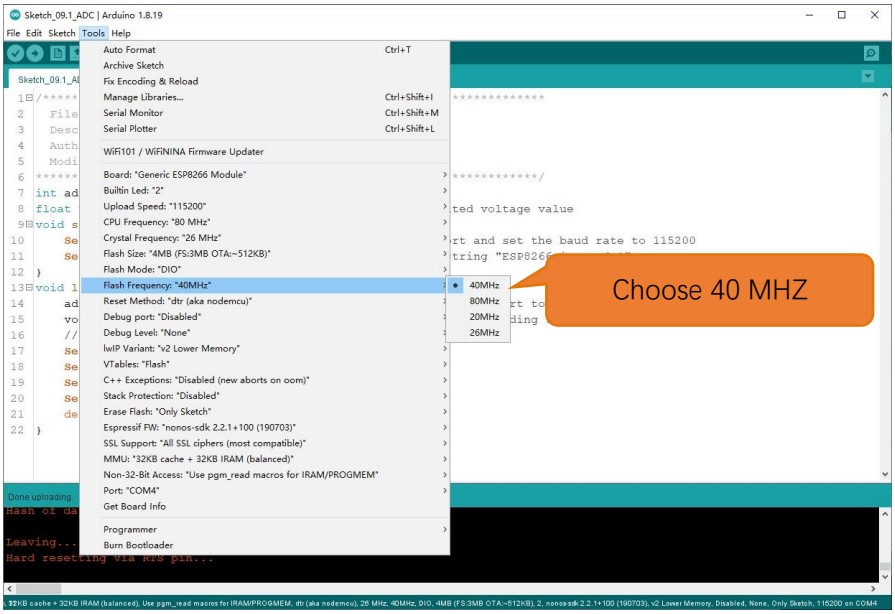
3.QOUT: Address is input in 1-line mode and data is output in 4-line mode.

4.QIO: Address is input in 4-line mode and data is output in 4-line mode.

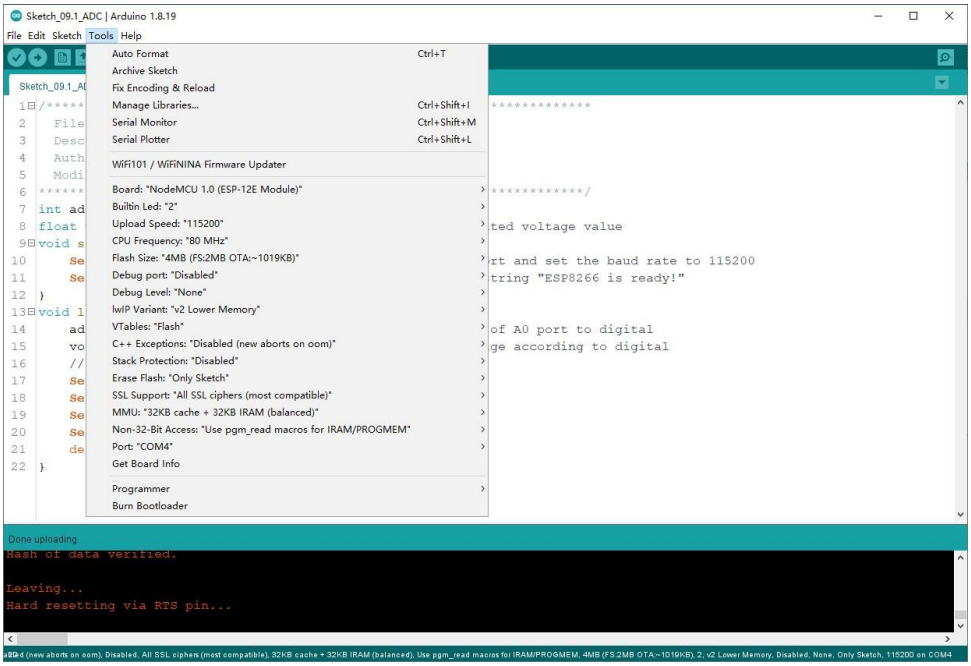
If you need to use the QIO mode, ensure that the Flash supports the QIO mode.



The flash chip connected to most chips operates at 40MHz clock speed, but you can try a lower value if the device fails to boot.The highest flash clock speed of 80MHz will provide the best performance, but can cause crashes if the flash or board design cannot achieve this speed.



If you select NodeMCU 1.0(ESP-12E Module), the following interface is displayed:



Here, you can see that this is similar to "Generic ESP8266 Module" in that the omitted parts are configured with default values.If you have problems working through this tutorial, try using the "Generic ESP8266 Module" configuration.

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

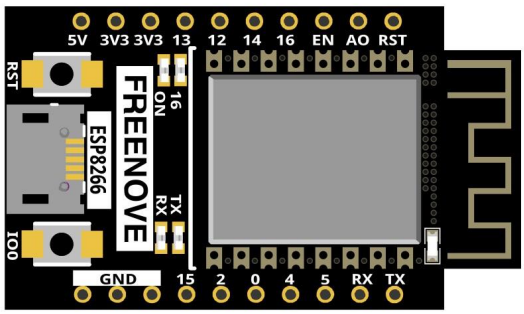
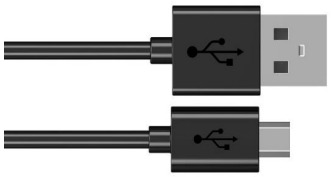
Chapter 1 LED

This chapter is the Start Point in the journey to build and explore ESP8266 electronic projects. We will start with simple “Blink” project.

Project 1.1 Blink

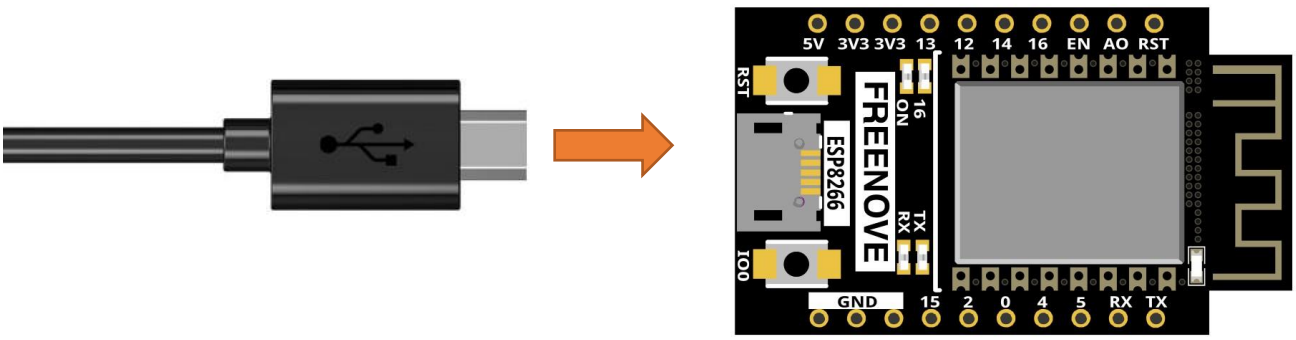
In this project, we will use ESP8266 to control blinking a common LED.

Component List

ESP8266 x1 	USB cable 

Power

ESP8266 needs 5v power supply. In this tutorial, we need connect ESP8266 development board 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 ESP8266 development board by default.

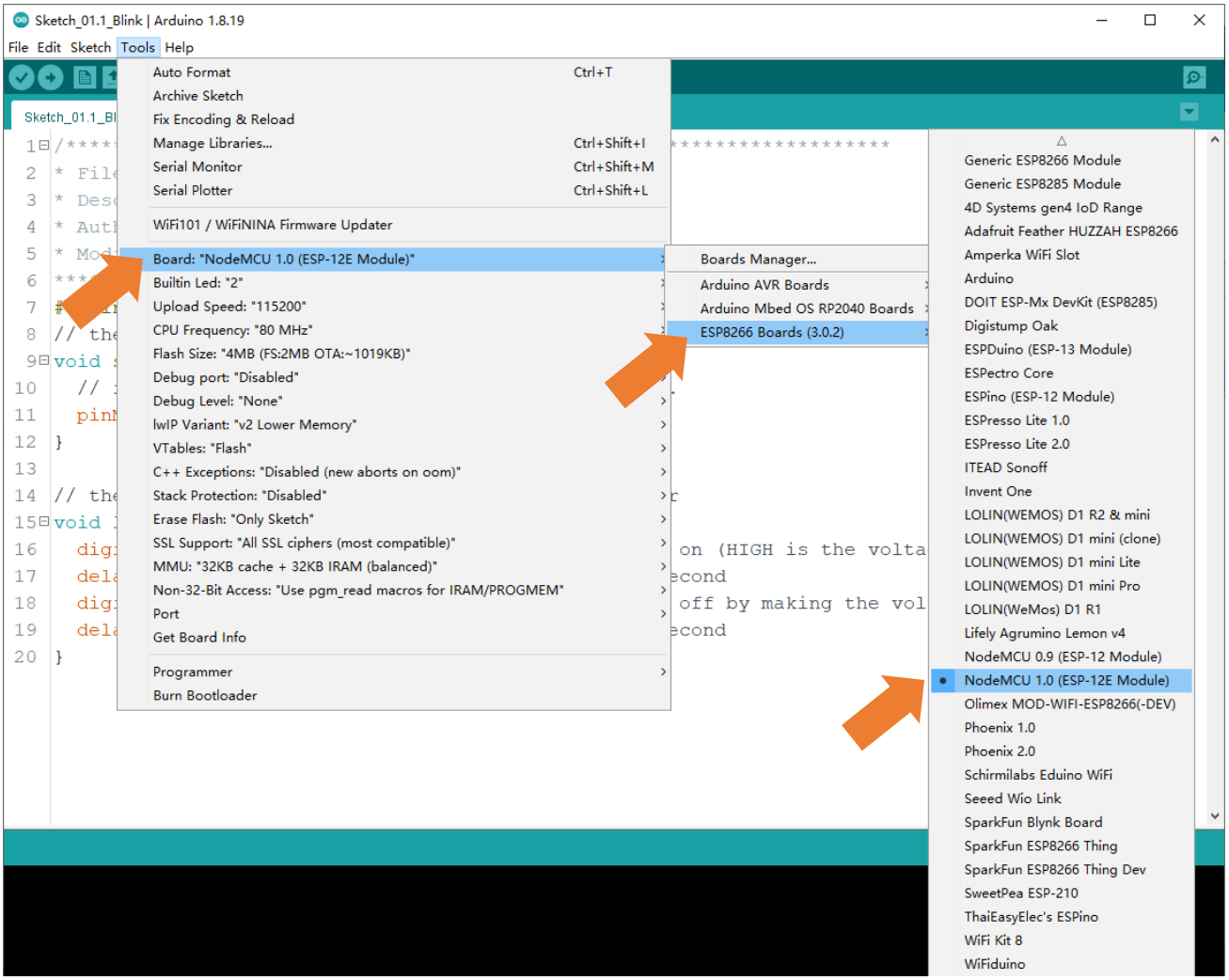
Sketch

According to the circuit, when the GPIO2 of ESP8266 output level is high, the LED turns ON. Conversely, when the GPIO2 ESP8266 output level is low, the LED turns OFF. Therefore, we can let GPIO2 circularly output high and low level to make the LED blink.

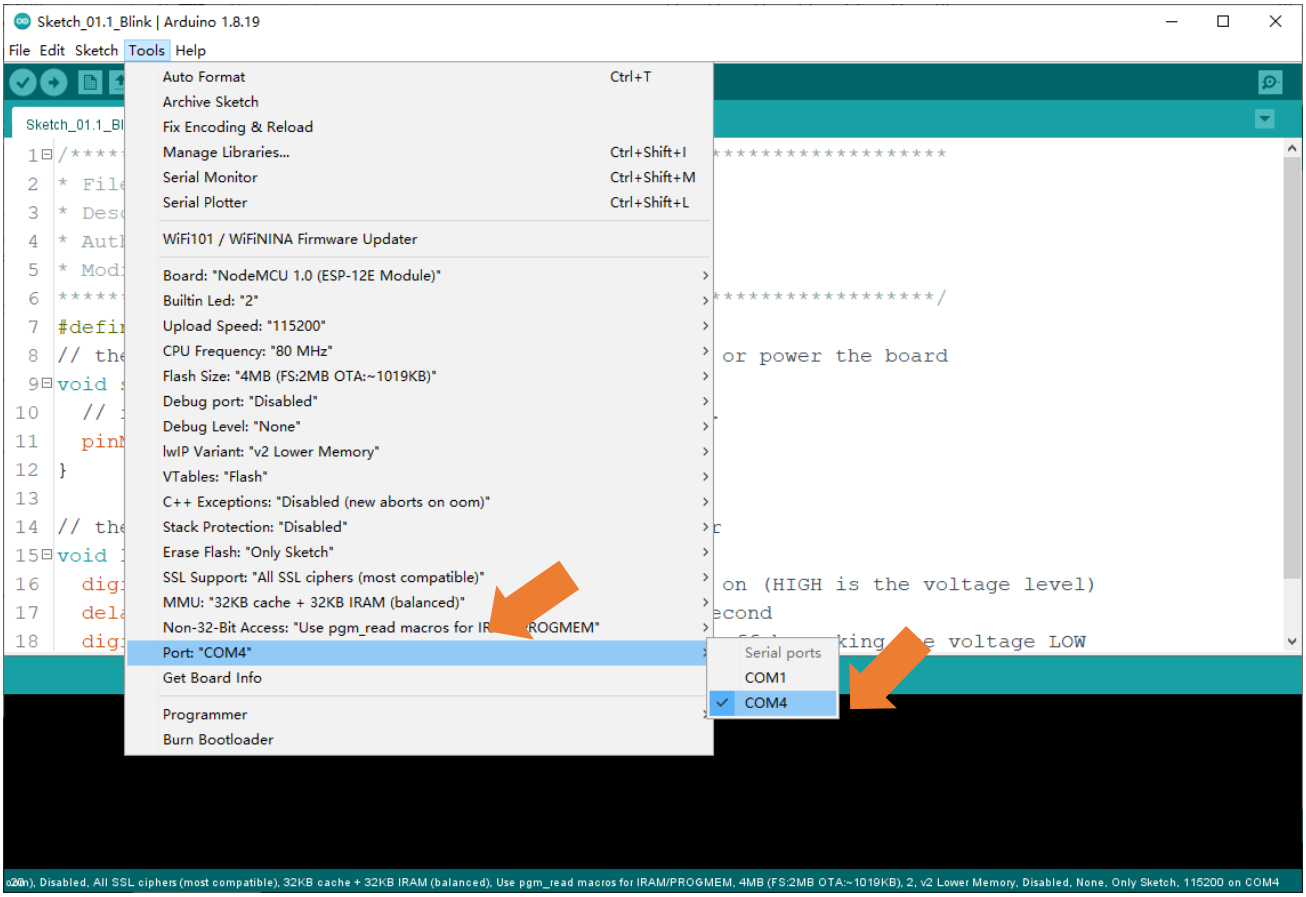
Upload the following Sketch:

Freenove_ESP8266_Board\C\Sketches\Sketch_01.1_Blink

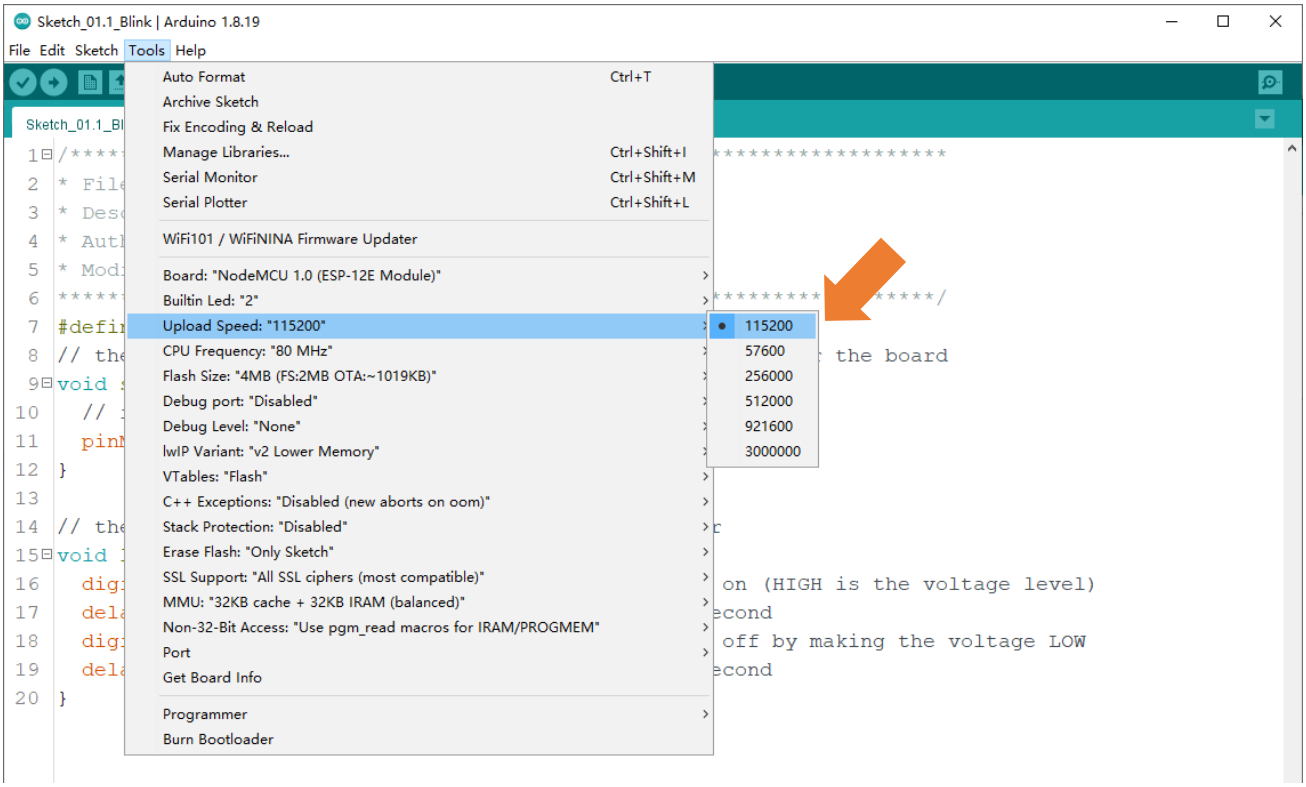
Before uploading the code, click "**Tools**", "**Board**" and select " **NodeMCU 1.0（ESP-12E Module）** ".



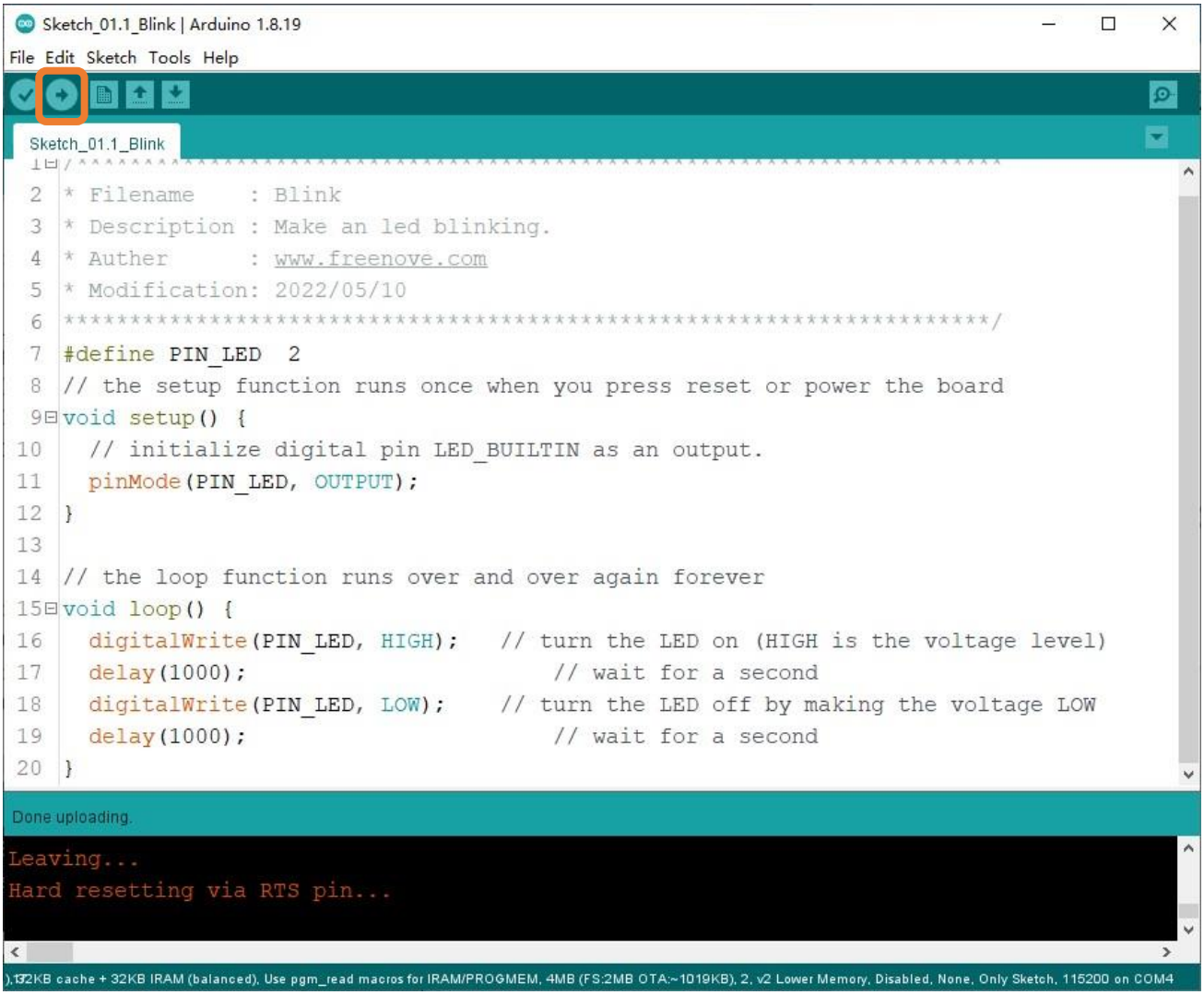
Select the serial port.



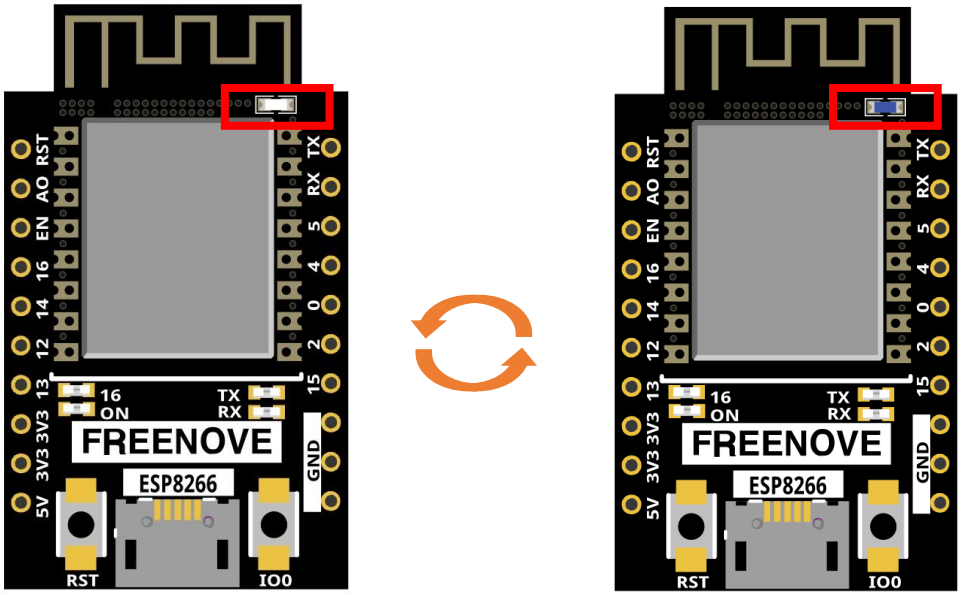
Note: For macOS users, if the uploading fails, please set the baud rate to 115200 before clicking "Upload Using Programmer".



Sketch_01.1_Blink



Click "Upload", Download the code to ESP8266 and your LED in the circuit starts Blink.



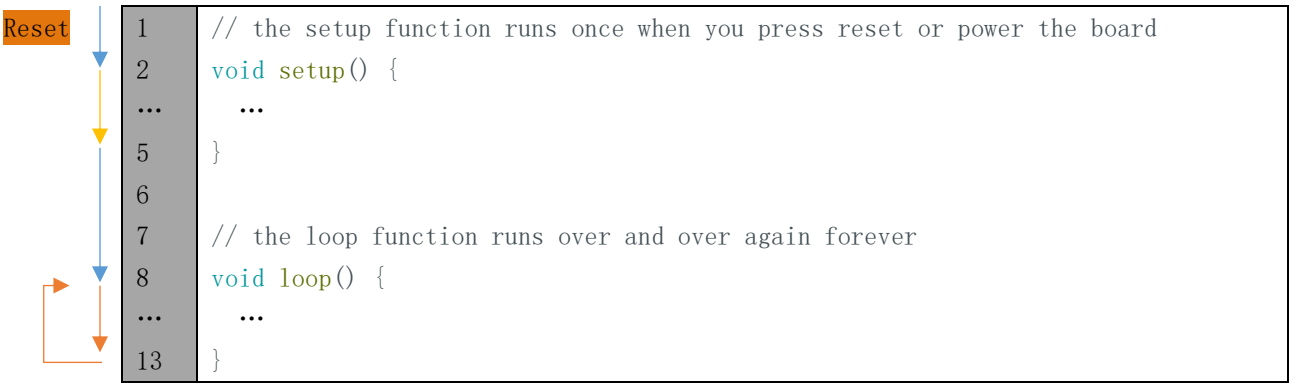
If you have any concerns, please contact us via: support@freenove.com

Any concerns? ✉ support@freenove.com

The following is the program code:

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

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

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



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, ESP8266's GPIO2 is connected to the LED, so the LED pin is defined as 2.

```
#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.

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

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

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

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

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

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

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

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

Reference

void pinMode(int pin, int mode);

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

Parameters

pin: the pin number to set the mode of.

mode: INPUT, OUTPUT, INPUT_PULLDOWM, or INPUT_PULLUP.

void digitalWrite (int pin, int value);

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

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

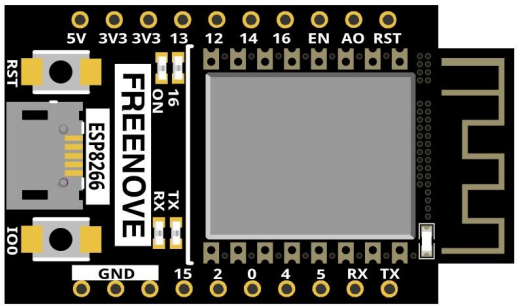
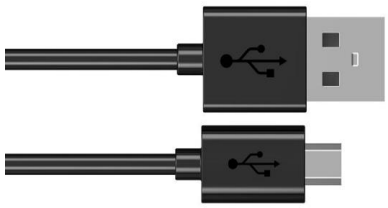
Chapter 2 Serial Communication

Serial Communication is a means of communication between different devices/devices. This section describes ESP8266's Serial Communication.

Project 2.1 Serial Print

This project uses ESP8266's serial communicator to send data to the computer and print it on the serial monitor.

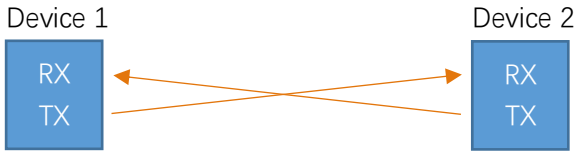
Component List

ESP8266 x1 	USB cable 

Related knowledge

Serial communication

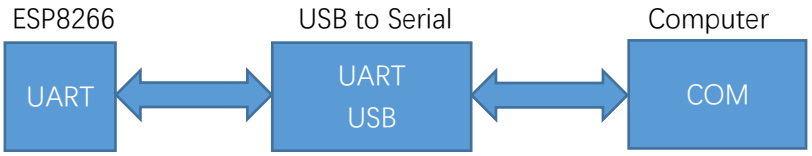
Serial communication generally refers to the Universal Asynchronous Receiver/Transmitter (UART), which is commonly used in electronic circuit communication. It has two communication lines, one is responsible for sending data (TX line) and the other for receiving data (RX line). The serial communication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



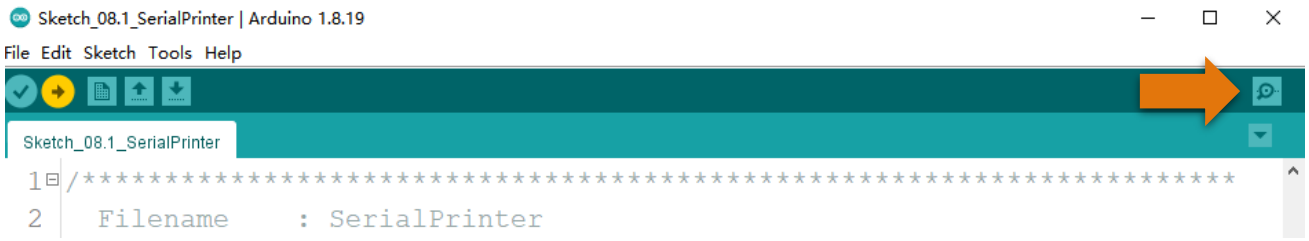
Before serial communication starts, the baud rate of both sides must be the same. Communication between devices can work only if the same baud rate is used. The baud rates commonly used is 9600 and 115200.

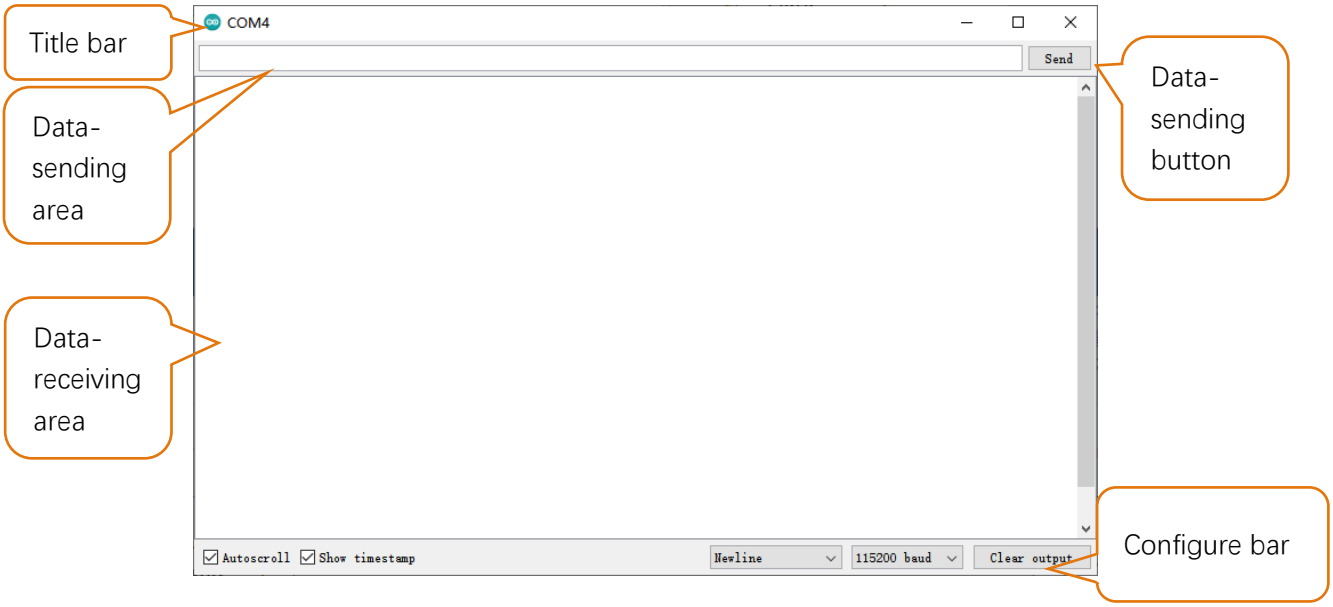
Serial port on ESP8266

Freenove ESP8266 has integrated USB to serial transfer, so it could communicate with computer connecting to USB cable.



Arduino Software also uploads code to Freenove ESP8266 through the serial connection.
Your computer identifies serial devices connecting to it as COMx. We can use the Serial Monitor window of Arduino Software to communicate with Freenove ESP8266, connect Freenove ESP8266 to computer through the USB cable, choose the correct device, and then click the Serial Monitor icon to open the Serial Monitor window.
Interface of serial monitor window is as follows. If you can't open it, make sure Freenove ESP8266 has been connected to the computer, and choose the right serial port in the menu bar "Tools".

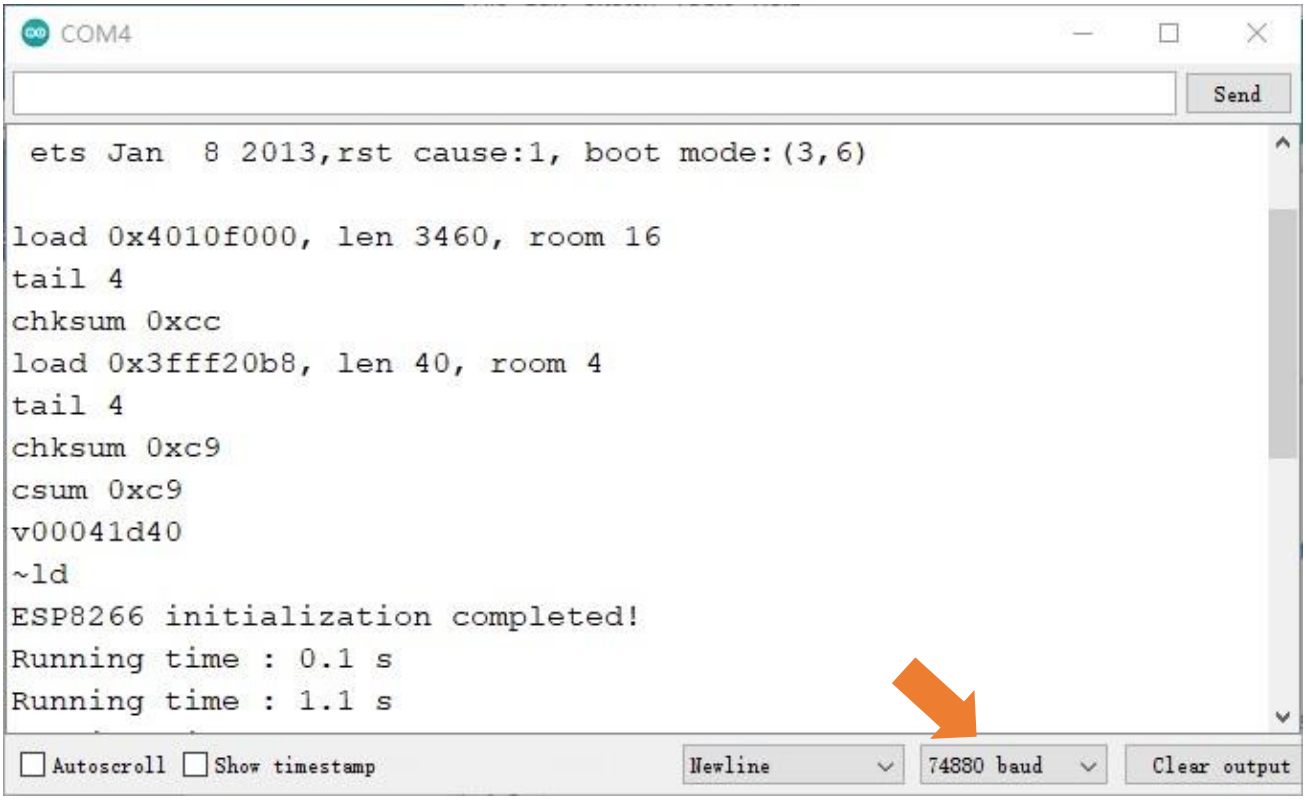




When the ESP8266 is powered on, the default baud rate is 74880. The default communication and serial port in the ESP8266 firmware is 115200. So if you set the serial port to 74880, this time can be displayed normally. If the baud rate is not 74880, some garbled characters will appear. Please do not worry.

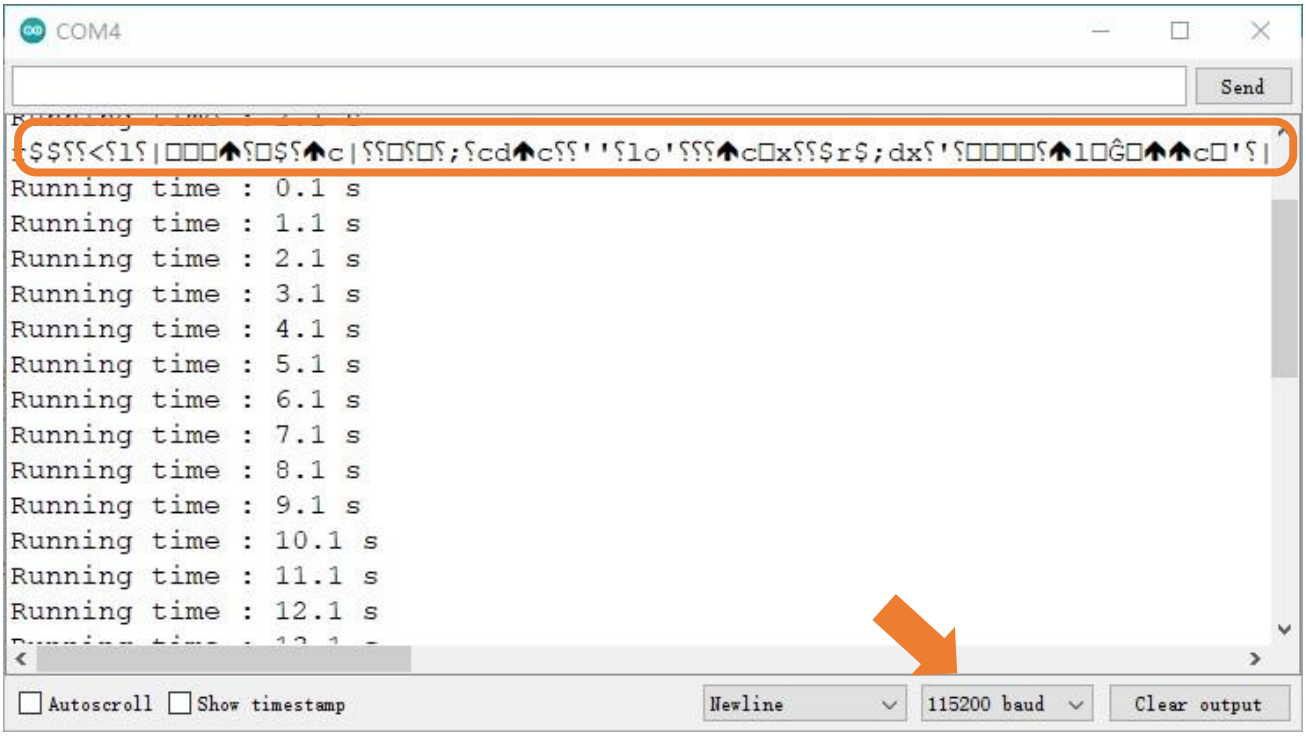
Here, we use The Arduino IDE serial port tool for output and display. The details are as follows:

```
void setup() {
    Serial.begin(74880);
    Serial.println("E 266 initialization completed!");
}

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



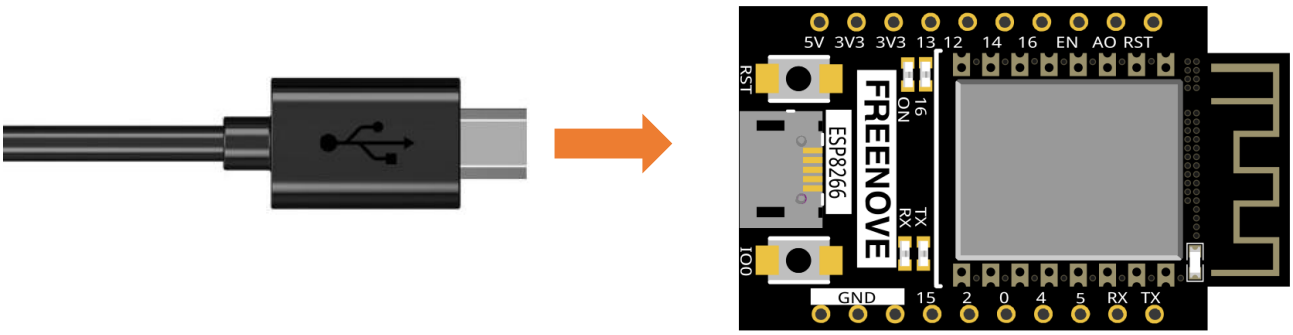
Garbled characters are displayed when you set the baud rate to another value as follows:

```
void setup() {
    Serial.begin(115200);
    Serial.println("8266 initialization completed!");
}

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



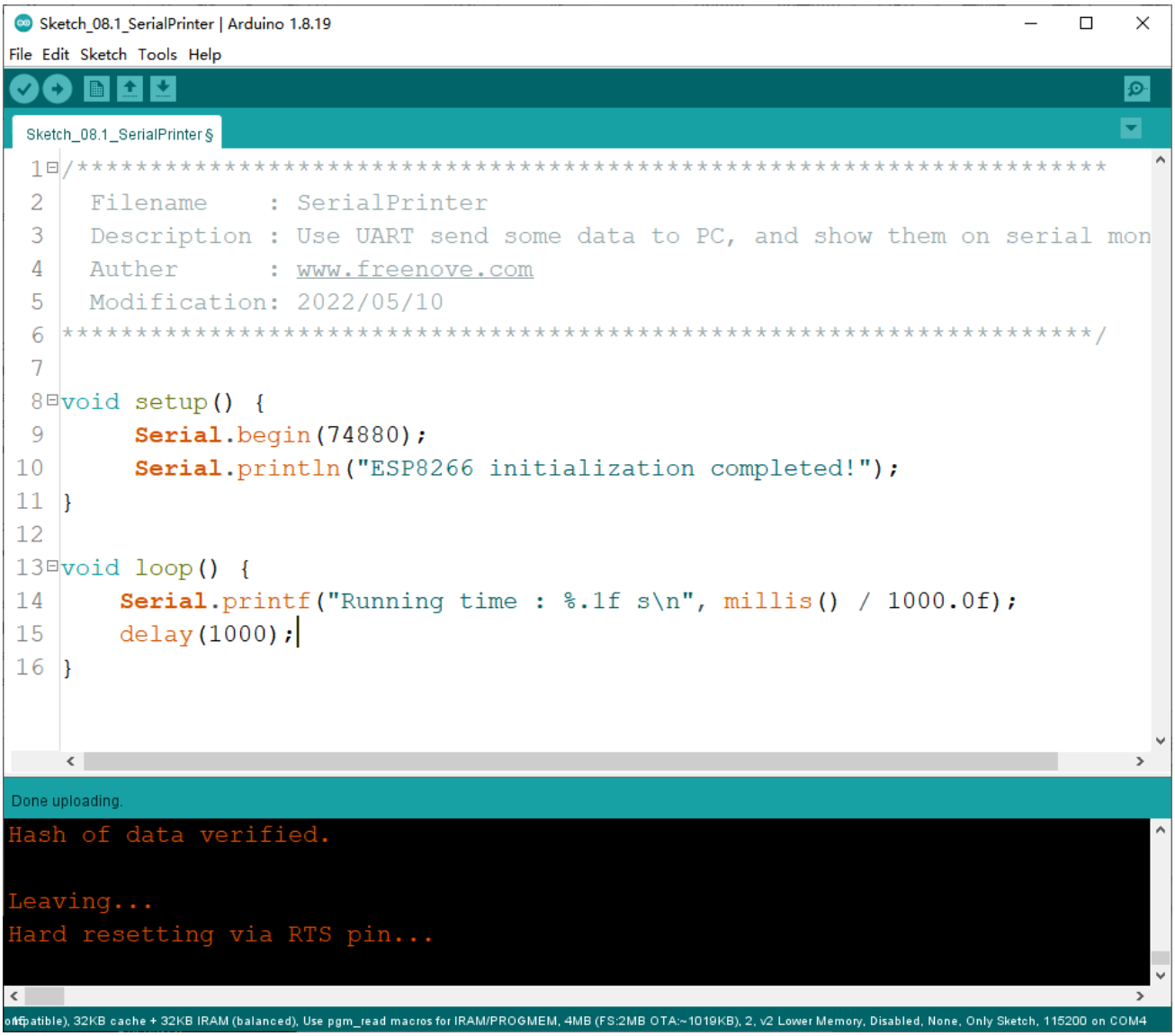
Circuit

Connect Freenove ESP8266 to the computer with USB cable.

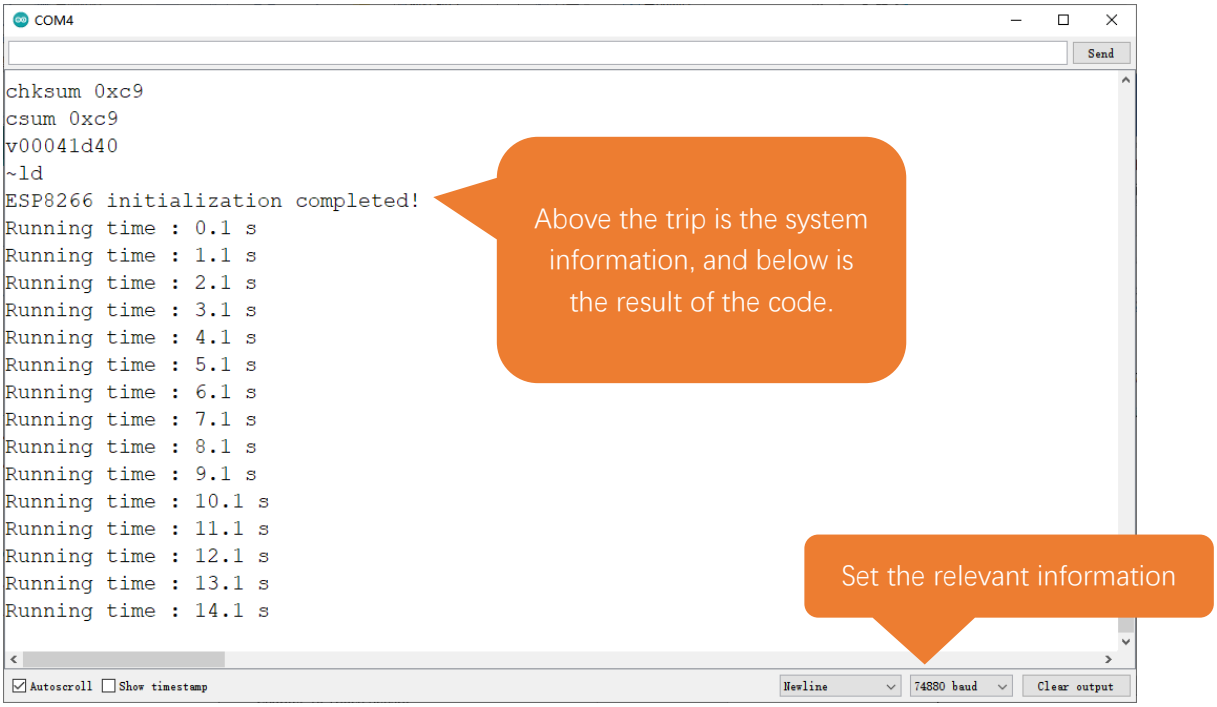


Sketch

Sketch_02.1_SerialPrinter



Download the code to ESP8266, open the serial port monitor, set the baud rate to 74880, and press the reset button. As shown in the following figure:



As shown in the image above, "ESP8266 initialization completed! " The previous is the printing message when the system is started, it uses the baud rate of 120,000, which is incorrect, so the garbled code is displayed. The user program is then printed at a baud rate of 74880.

For more information, click here.

The following is the program code:

```
1  void setup() {
2    Serial.begin(74880);
3    Serial.println("ESP8266 initialization completed! ");
4  }
5
6  void loop() {
7    Serial.printf("Running time : %.1f s\n", millis() / 1000.0f);
8    delay(1000);
9  }
```

Reference

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

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

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

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

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

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

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

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

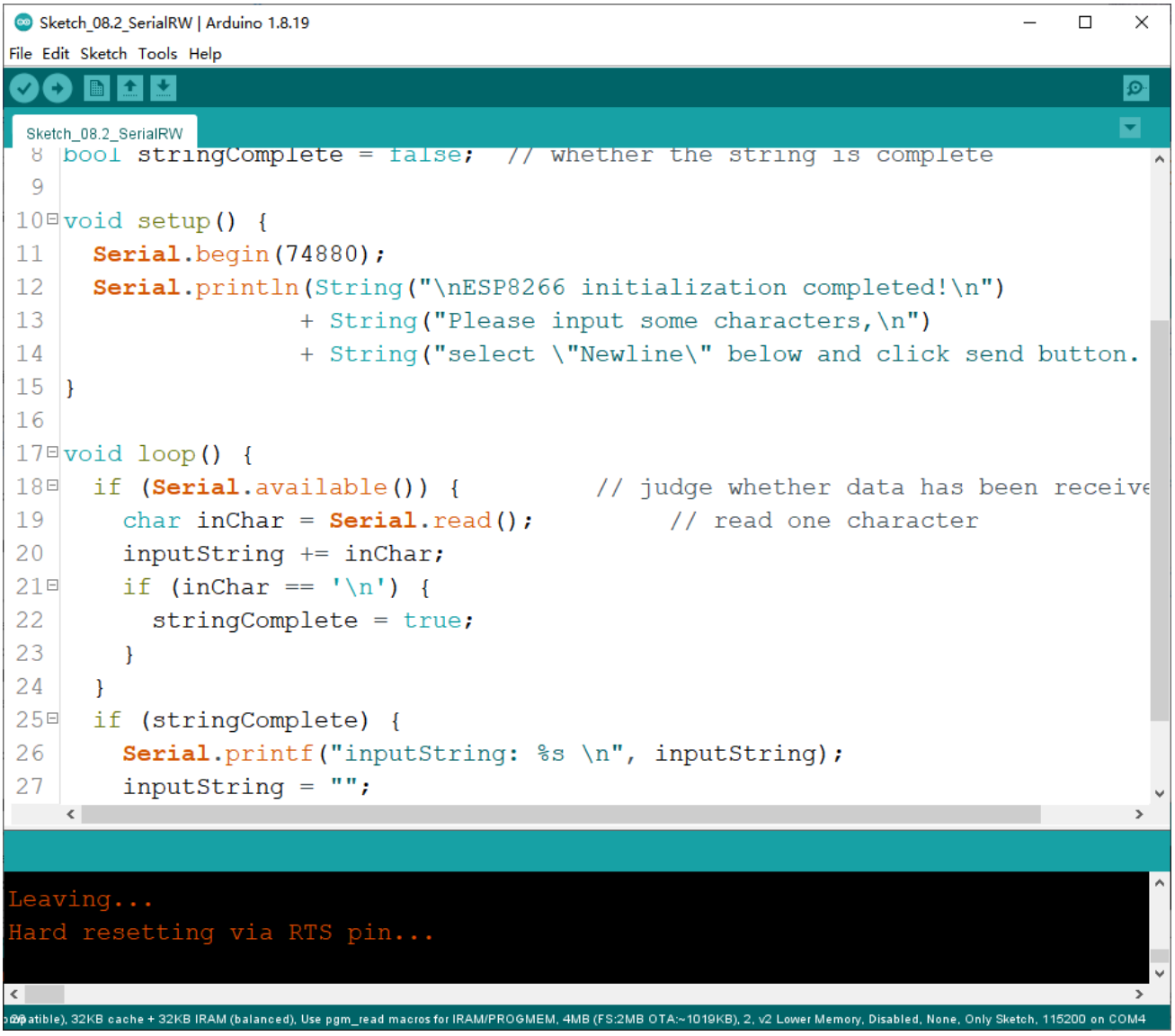
Project 2.2 Serial Read and Write

From last section, we use serial port on Freenove ESP8266 to send data to a computer, now we will use that to receive data from computer.

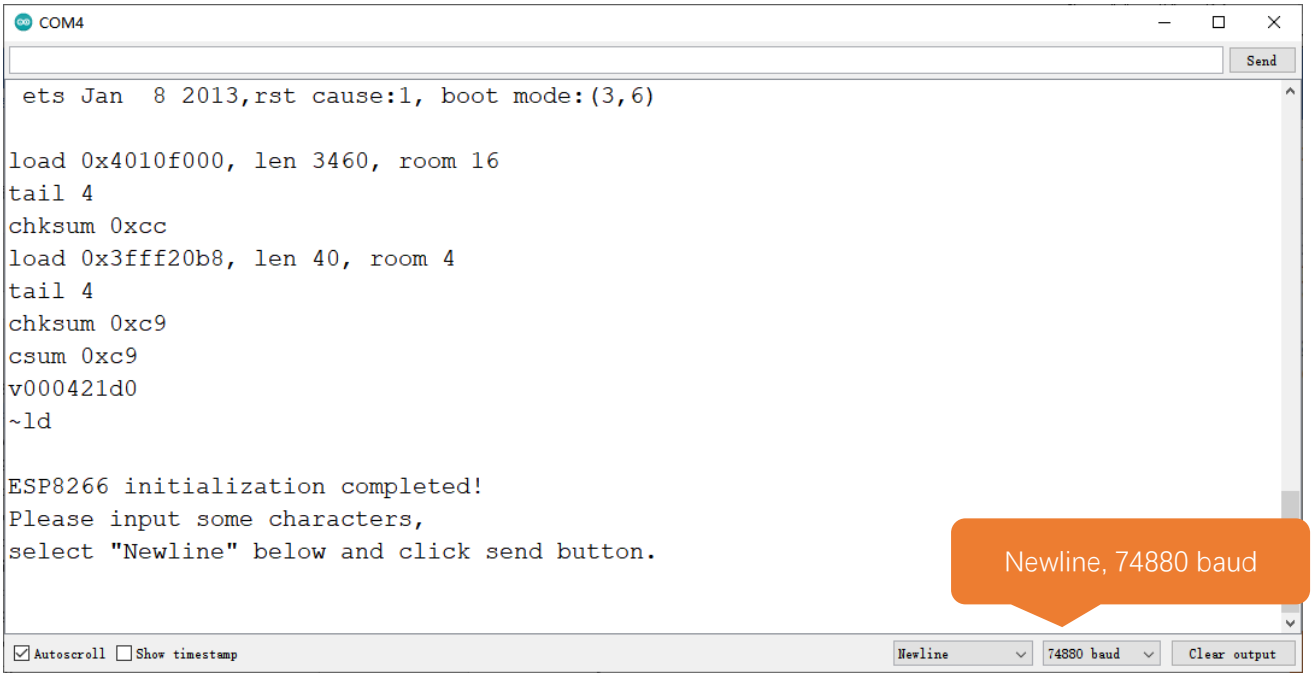
Component and circuit are the same as in the previous project.

Sketch

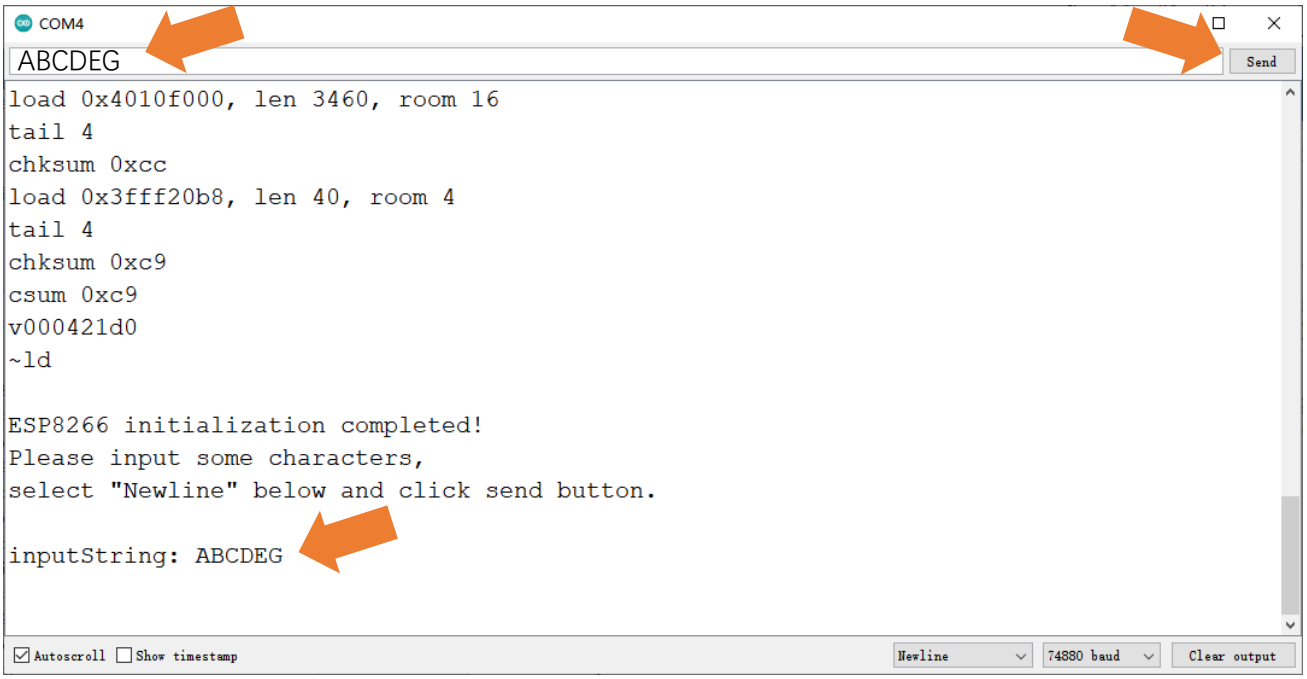
Sketch_02.2_SerialRW



Download the code to ESP8266, open the serial monitor, and set the bottom to Newline, 74880. As shown in the following figure:



Then type characters like 'ABCDEG' into the data sent at the top and click the Send button to print out the data ESP8266 receives.



The following is the program code:

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

void setup() {
  Serial.begin(74880);
  Serial.println(String("\nESP8266 initialization completed! \n")
                 + String("Please input some characters,\n")
                 + String("select \"Newline\" below and click send button. \n"));
}

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

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

Reference

`String();`

Constructs an instance of the String class.
For more information, please visit
https://www.arduino.cc/reference/en/language/variables/data-types/stringobject/

`int available(void);`

Get the number of bytes (characters) available for reading from the serial port. This is data that's already arrived and stored in the serial receive buffer.

`Serial.read();`

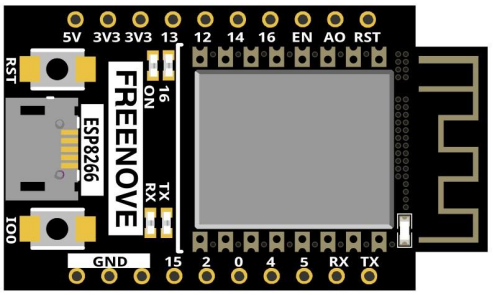
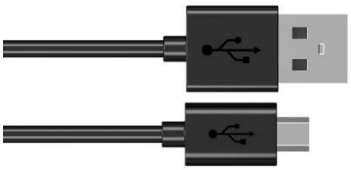
Reads incoming serial data.

Chapter 3 WiFi Working Modes

In this chapter, we'll focus on the WiFi infrastructure for ESP8266.
ESP8266 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 3.1 Station mode

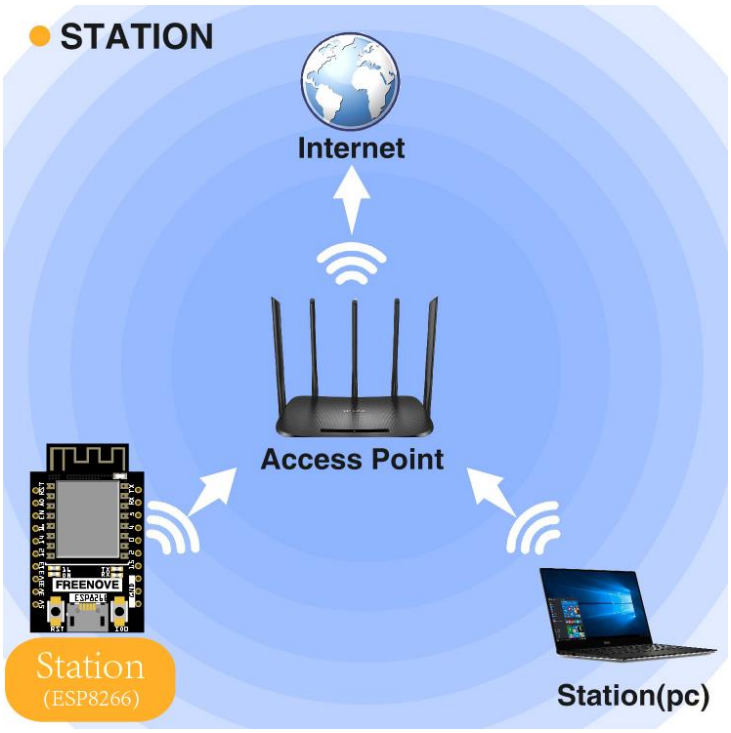
Component List

ESP8266 x1 	Micro USB Wire x1 

Component knowledge

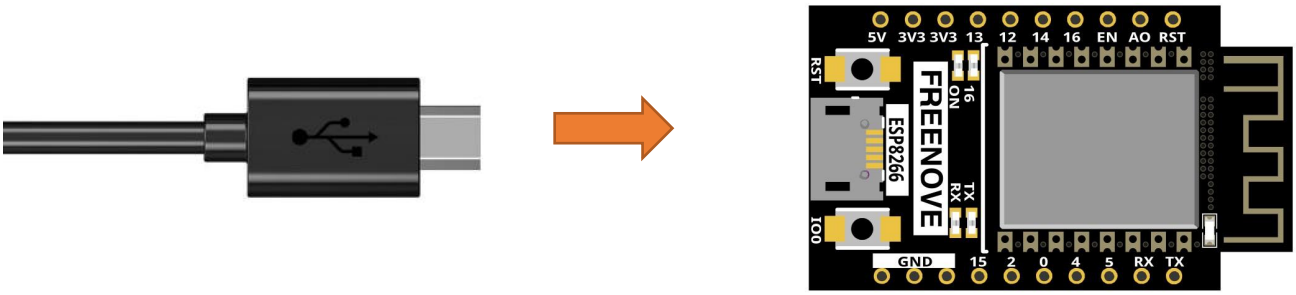
Station mode

When ESP8266 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 ESP8266 wants to communicate with the PC, it needs to be connected to the router.



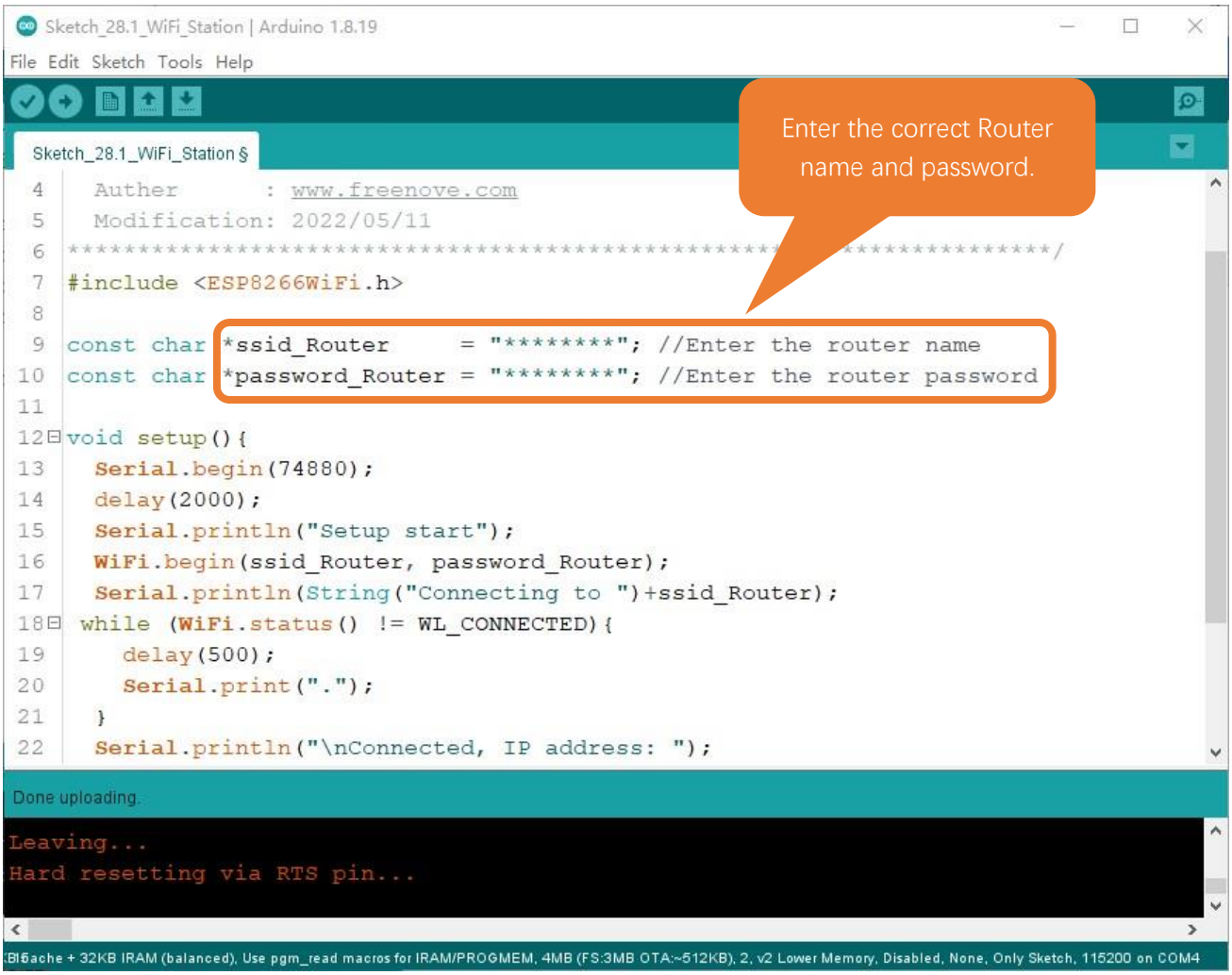
Circuit

Connect Freenove ESP8266 to the computer using the USB cable.



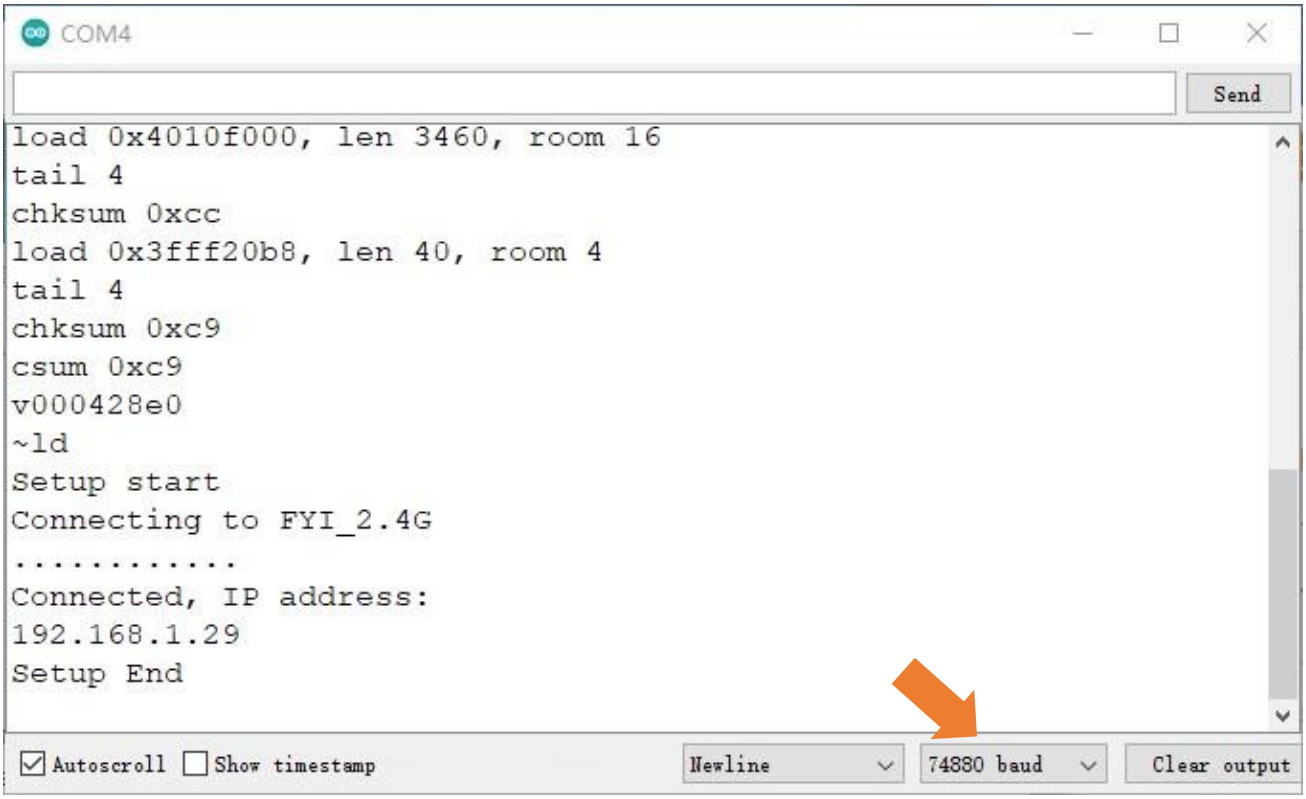
Sketch

Sketch_03.1_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 ESP8266, open serial monitor and set baud rate to 74880. And then it will display as follows:



When ESP8266 successfully connects to "ssid_Router", serial monitor will print out the IP address assigned to ESP8266 by the router.

The following is the program code:

```
#include <ESP8266WiFi.h>

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

void setup(){
  Serial.begin(74880);
  delay(2000);
  Serial.println("Setup start");
  WiFi.begin(ssid_Router, password_Router);
  Serial.println(String("Connecting to ")+ssid_Router);
  while (WiFi.status() != WL_CONNECTED){
    delay(500);
    Serial.print(".");
  }
  Serial.println("\nConnected, IP address: ");
  Serial.println(WiFi.localIP());
  Serial.println("Setup End");
}

void loop() {
```

```
}
```

Include the WiFi Library header file of ESP8266.

```
#include <ESP8266WiFi.h>
```

Enter correct router name and password.

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

Set ESP8266 in Station mode and connect it to your router.

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

Check whether ESP8266 has connected to router successfully every 0.5s.

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

Serial monitor prints out the IP address assigned to ESP8266

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

Reference

Class Station

Every time when using WiFi, you need to include header file "ESP8266WiFi.h".

begin(ssid, password,channel, bssid, connect): ESP8266 is used as Station to connect hotspot.

ssid: WiFi hotspot name

password: WiFi hotspot password

channel: WiFi hotspot channel number; communicating through specified channel; optional parameter

bssid: mac address of WiFi hotspot, optional parameter

connect: blloean optional parameter, defaulting to true. If set as false, then ESP8266 won't connect WiFi.

config(local_ip, gateway, subnet, dns1, dns2): set static local IP address.

local_ip: station fixed IP address.

subnet: subnet mask

dns1,dns2: optional parameter. define IP address of domain name server

status: obtain the connection status of WiFI

local IP(): obtian IP address in Station mode

disconnect(): disconnect wifi

setAutoConnect(boolen): set automatic connection Every time ESP8266 is power on, it will connect WiFi aitomatically.

setAutoReconnect(boolen): set automatic reconnection Every time ESP8266 disconnects WiFi, it will reconnect to WiFi automatically.

Project 3.2 AP mode

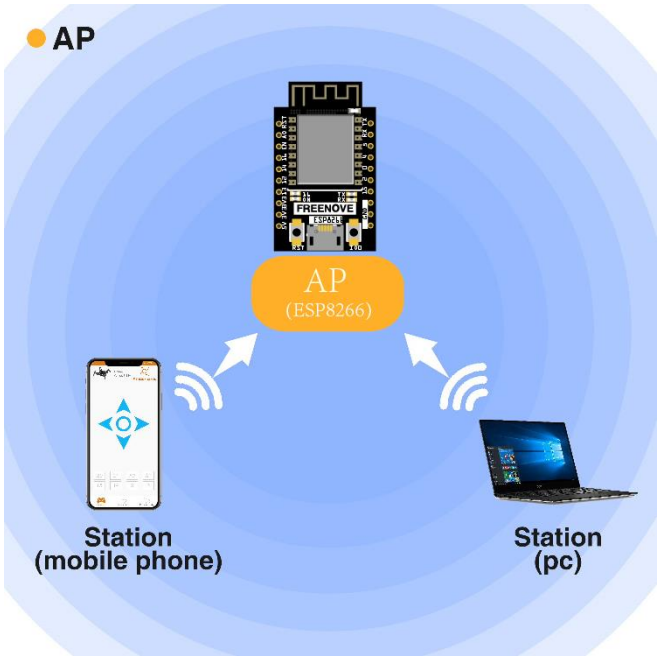
Component List & Circuit

Component List & Circuit are the same as in Section 28.1.

Component knowledge

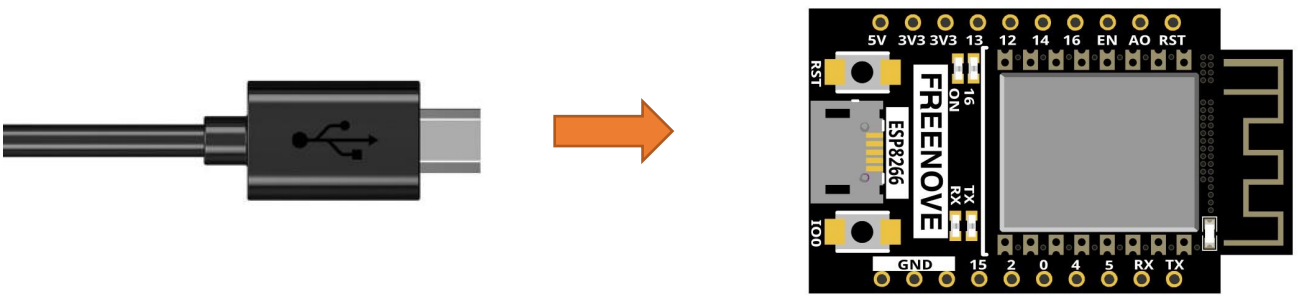
AP mode

When ESP8266 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, ESP8266 is used as a hotspot. If a mobile phone or PC wants to communicate with ESP8266, it must be connected to the hotspot of ESP8266. Only after a connection is established with ESP8266 can they communicate.



Circuit

Connect Freenove ESP8266 to the computer using the USB cable.

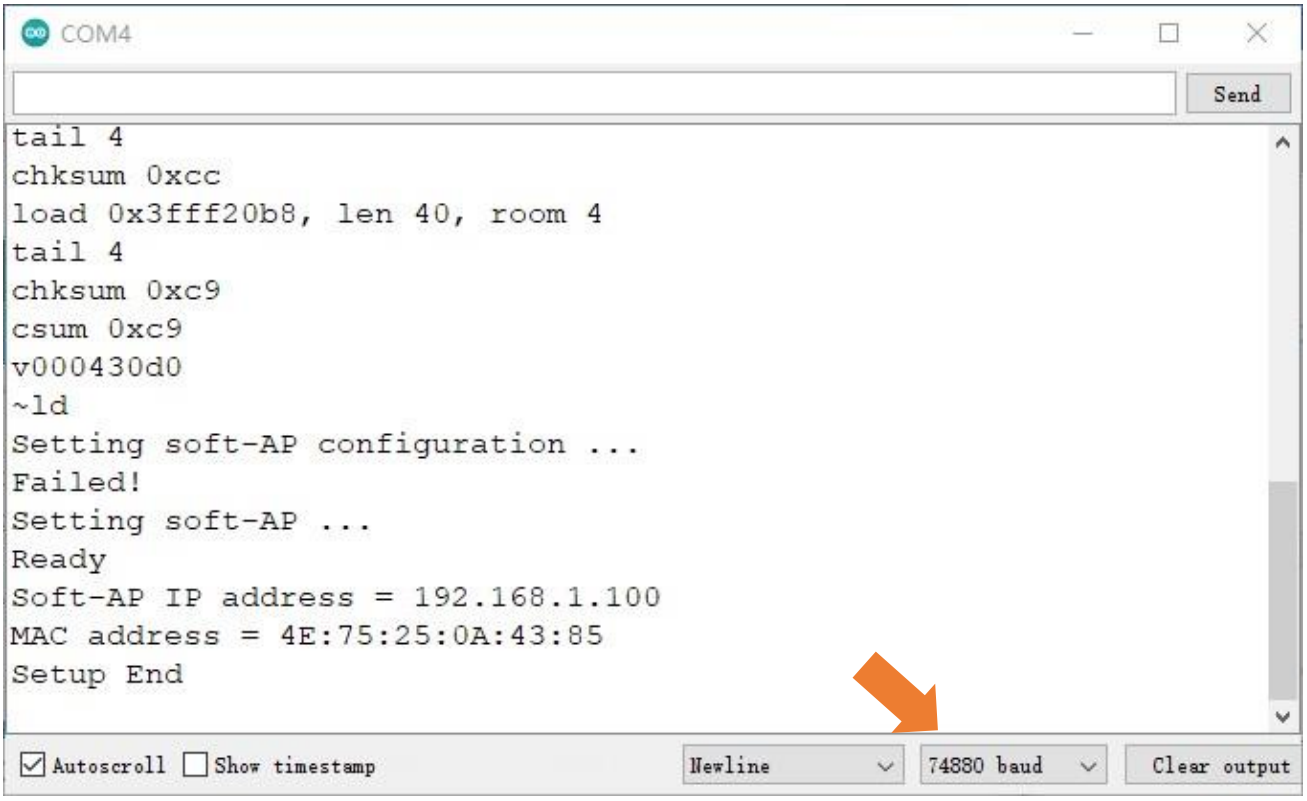


Sketch

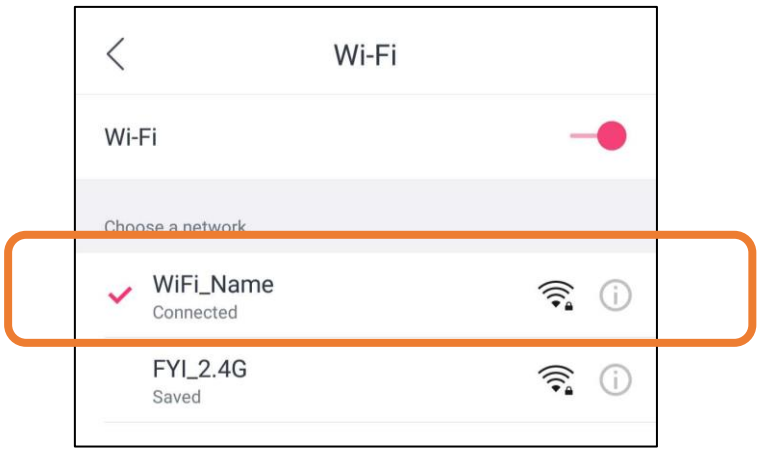
Sketch_03.2_AP_mode

Before the Sketch runs, you can make any changes to the AP name and password for ESP8266 in the box as shown in the illustration above. Of course, you can leave it alone by default.

Compile and upload codes to ESP8266, open the serial monitor and set the baud rate to 74880. 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 ESP8266, 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.



The following is the program code:

```
#include <ESP8266WiFi.h>

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

IPAddress local_IP(192,168,1,100);//Set the IP address of ESP8266 itself
IPAddress gateway(192,168,1,10);   //Set the gateway of ESP8266 itself
IPAddress subnet(255,255,255,0);  //Set the subnet mask for ESP8266 itself

void setup(){
  Serial.begin(74880);
  delay(2000);
  Serial.println("Setting soft-AP configuration ... ");
  WiFi.disconnect();
  WiFi.mode(WIFI_AP);
  Serial.println(WiFi.softAPConfig(local_IP, gateway, subnet) ? "Ready" : "Failed!");
  Serial.println("Setting soft-AP ... ");
  boolean result = WiFi.softAP(ssid_AP, password_AP);
  if(result){
    Serial.println("Ready");
    Serial.println(String("Soft-AP IP address = ") + WiFi.softAPIP().toString());
    Serial.println(String("MAC address = ") + WiFi.softAPmacAddress().c_str());
  }else{
    Serial.println("Failed!");
  }
  Serial.println("Setup End");
}

void loop() {
}
```

Include WiFi Library header file of ESP8266.

```
#include <ESP8266WiFi.h>
```

Enter correct AP name and password.

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

Set ESP8266 in AP mode.

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

Configure IP address, gateway and subnet mask for ESP8266.

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

Turn on an AP in ESP8266, whose name is set by ssid_AP and password is set by password_AP.

```
WiFi.softAP(ssid_AP, password_AP);
```

Check whether the AP is turned on successfully. If yes, print out IP and MAC address of AP established by ESP8266. 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 "ESP8266WiFi.h".

softAP(ssid, password, channel, ssid_hidden, max_connection):

ssid: WiFi hotspot name

password: WiFi hotspot password

channel: Number of WiFi connection channels, range 1-13. The default is 1.

ssid_hidden: Whether to hide WiFi name from scanning by other devices. The default is not hide.

max_connection: Maximum number of WiFi connected devices. The range is 1-4. The default is 4.

softAPConfig(local_ip, gateway, subnet): set static local IP address.

local_ip: station fixed IP address.

Gateway: gateway IP address

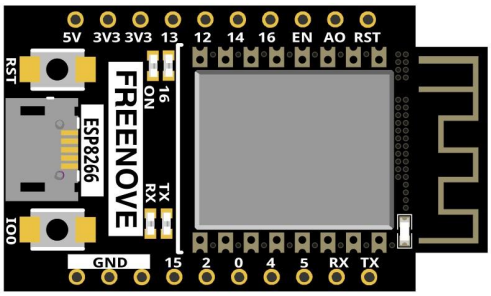
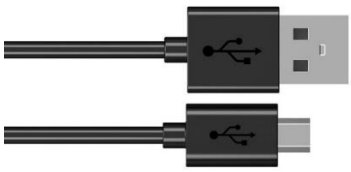
subnet: subnet mask

softAP(): obtian IP address in AP mode

softAPdisconnect (): disconnect AP mode.

Project 3.3 AP+Station mode

Component List

ESP8266 x1 	Micro USB Wire x1 

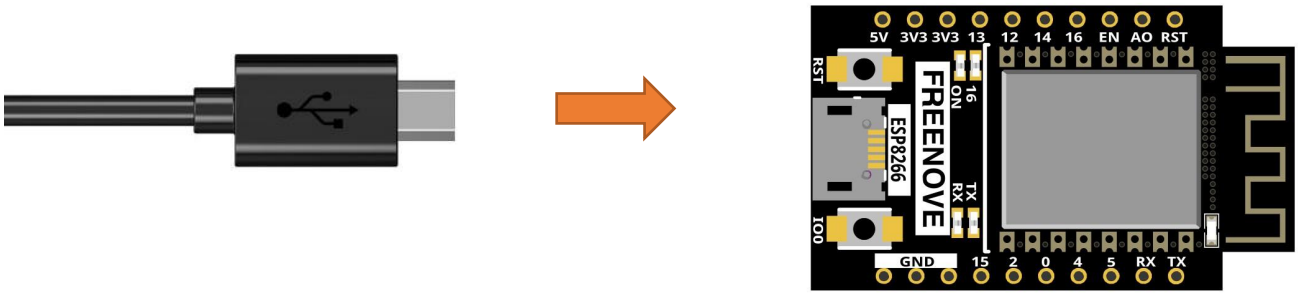
Component knowledge

AP+Station mode

In addition to AP mode and station mode, ESP8266 can also use AP mode and station mode at the same time. This mode contains the functions of the previous two modes. Turn on ESP8266'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 ESP8266.

Circuit

Connect Freenove ESP8266 to the computer using the USB cable.

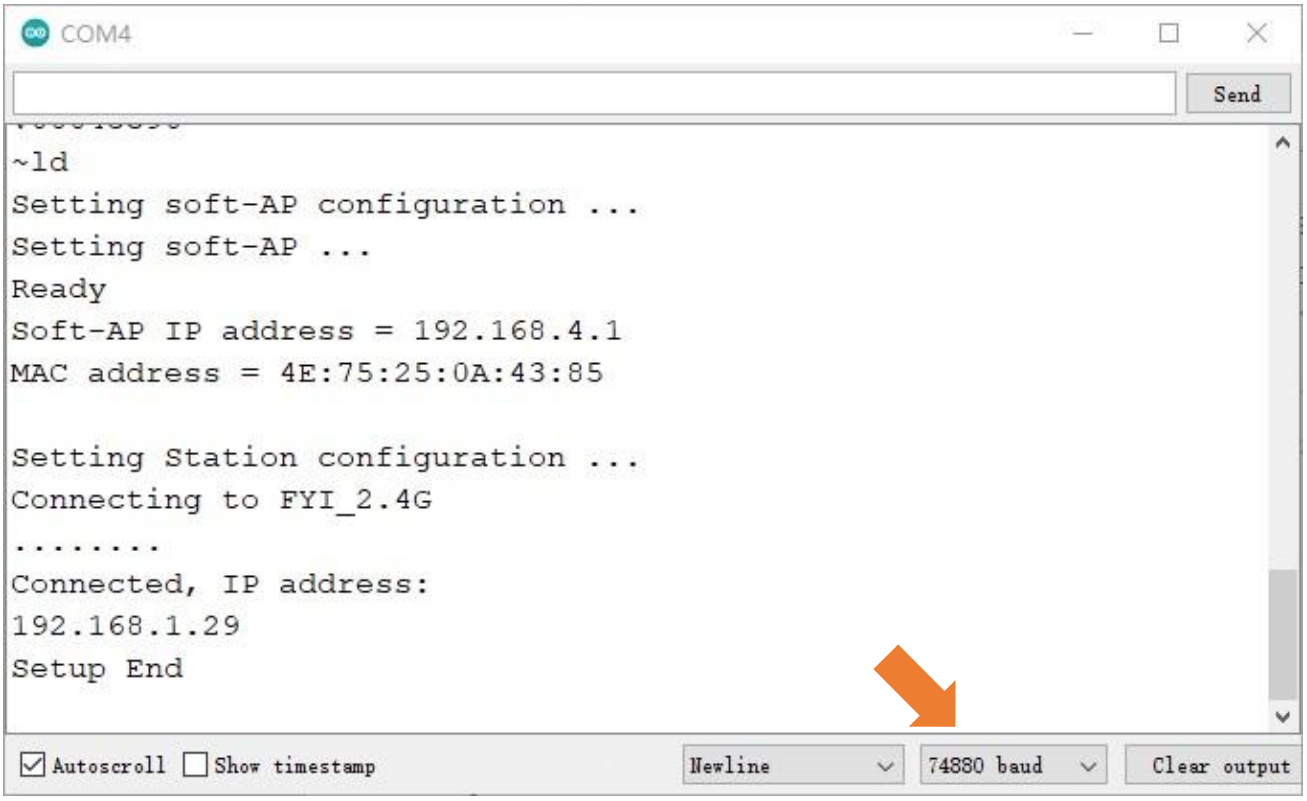


Sketch

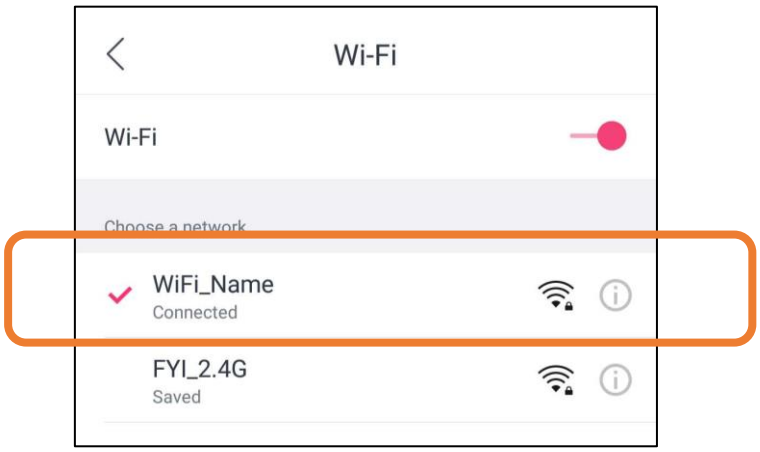
Sketch_03.3_AP_Station_mode

It is analogous to Project 3.1 and Project 3.2. 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 ESP8266, open serial monitor and set baud rate to 74880. 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 ESP8266.



The following is the program code:

```
#include <ESP8266WiFi.h>

const char *ssid_Router     = "********";  //Enter the router name
const char *password_Router = "********";  //Enter the router password
const char *ssid_AP         = "WiFi_Name"; //Enter the AP name
const char *password_AP     = "12345678";  //Enter the AP password

```

```
8  void setup(){
9    Serial.begin(74880);
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 }
```

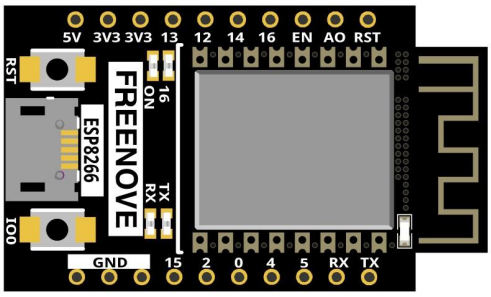
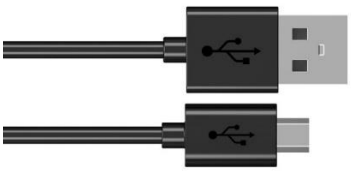
Chapter 4 TCP/IP

In this chapter, we will introduce how ESP8266 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 4.1 As Client

In this section, ESP8266 is used as Client to connect Server on the same LAN and communicate with it.

Component List

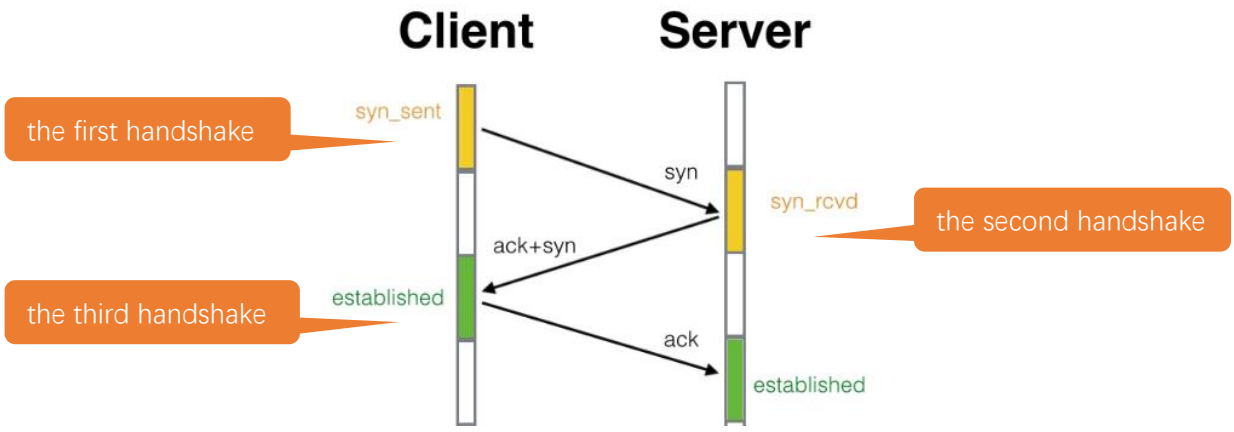
ESP8266 x1 	Micro USB Wire x1 

Component knowledge

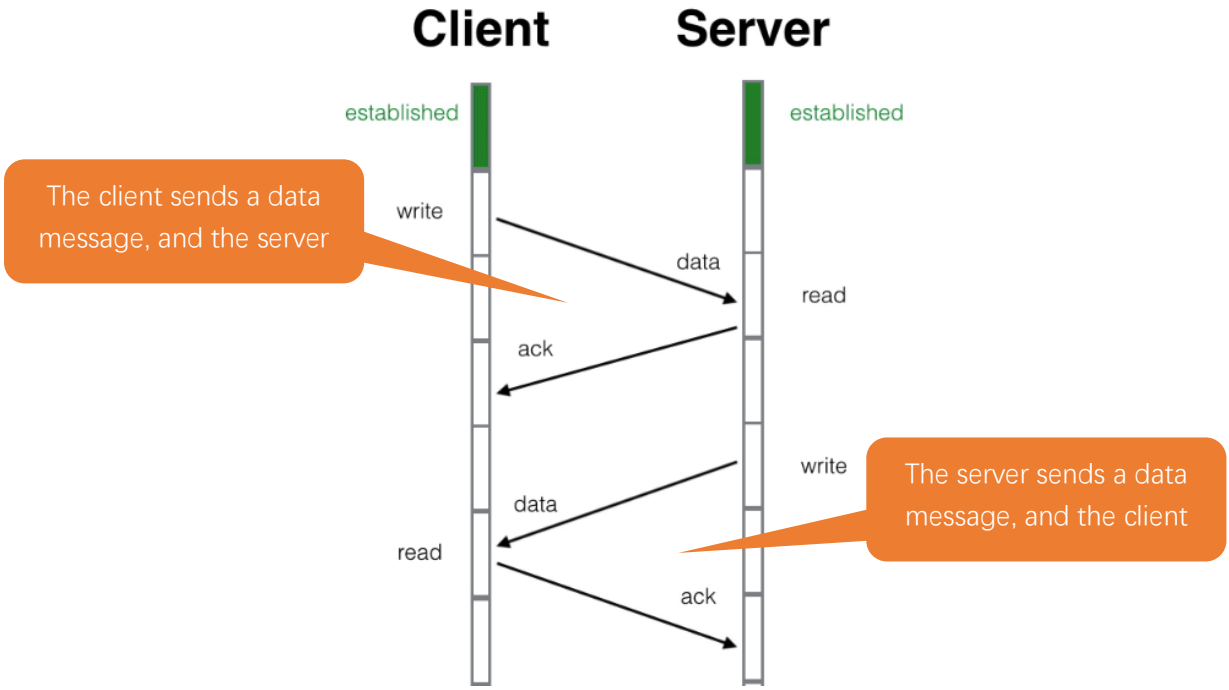
TCP connection

Before transmitting data, TCP needs to establish a logical connection between the sending end and the receiving end. It provides reliable and error-free data transmission between the two computers. In the TCP connection, the client and the server must be clarified. The client sends a connection request to the server, and each time such a request is proposed, a "three-times handshake" is required.

Three-times handshake: In the TCP protocol, during the preparation phase of sending data, the client and the server interact three times to ensure the reliability of the connection, which is called "three-times handshake".
The first handshake, the client sends a connection request to the server and waits for the server to confirm.
The second handshake, the server sends a response back to the client informing that it has received the connection request.
The third handshake, the client sends a confirmation message to the server again to confirm the connection.



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



Install Processing

In this tutorial, we use Processing to build a simple TCP/IP communication platform.

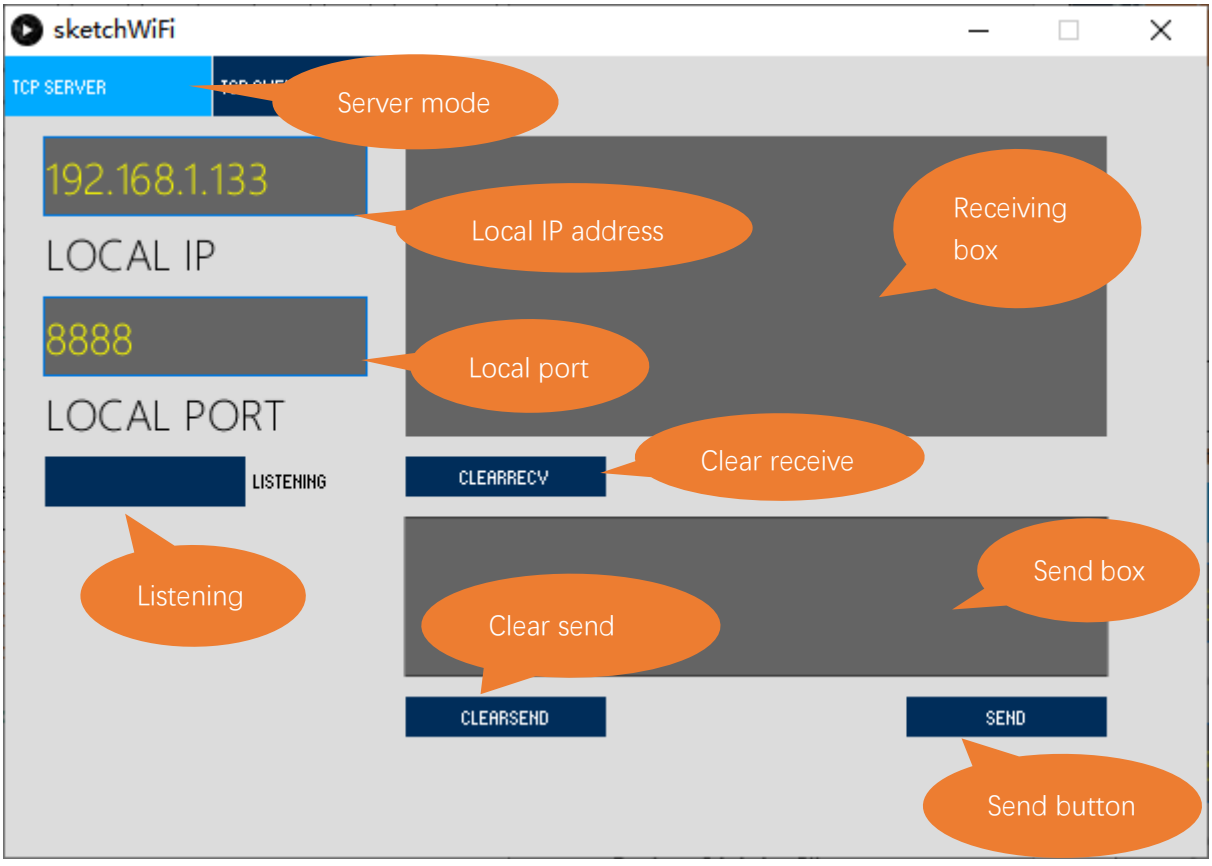
If you've not installed Processing, you can download it by clicking https://processing.org/download/. You can choose an appropriate version to download according to your PC system.

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

Use Server mode for communication

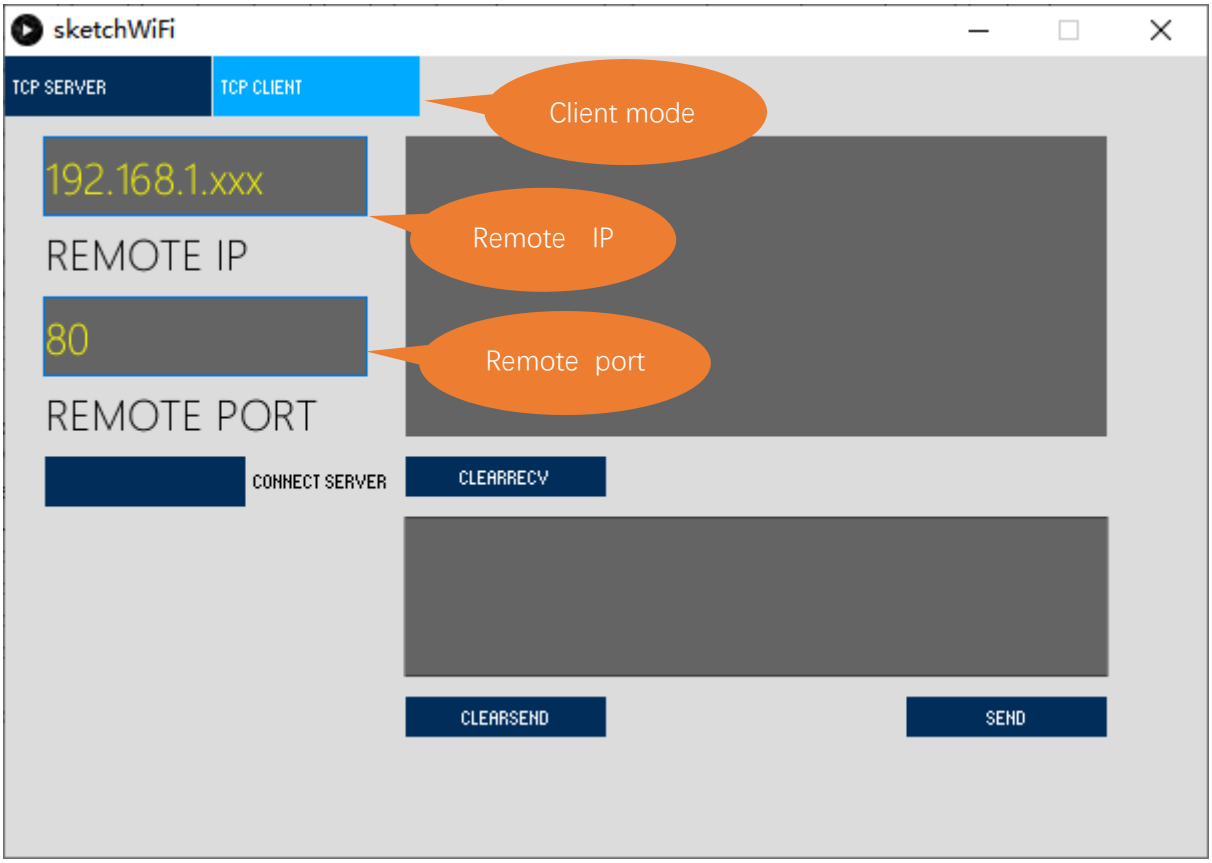
Open the "**Freenove_ESP8266_Board\C\Sketches\Sketch_04.1_WiFiClient\sketchWiFi\sketchWiFi.pde**", and click "Run".

The new pop-up interface is as follows. If ESP8266 is used as client, select TCP SERVER mode for sketchWiFi.



When sketchWiFi selects TCP SERVER mode, ESP8266 Sketch needs to be changed according to sketchWiFi's displaying of LOCAL IP or LOCAL PORT.

If ESP8266 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.

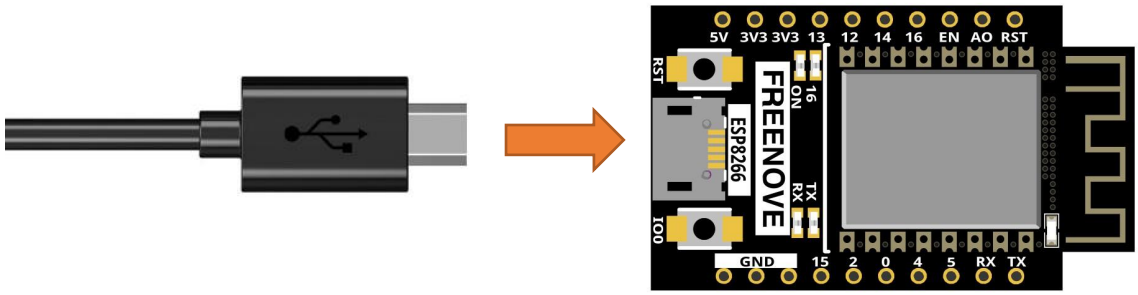
clear receive: clear out the content in the receiving text box

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

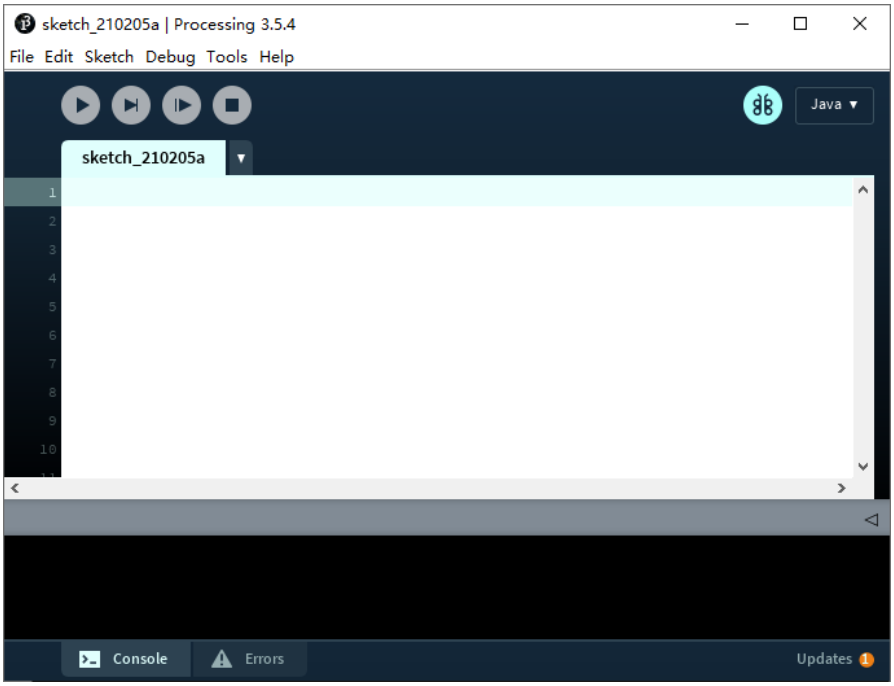
Connect Freenove ESP8266 to the computer using USB cable.



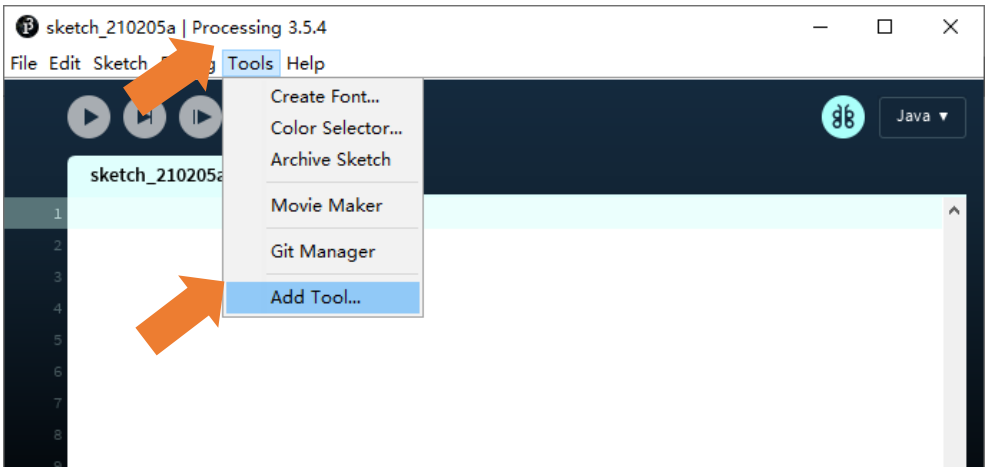
Sketch

If you have not installed "ControlP5", please follow the following steps to continue the installation, if you have installed, please skip this section.

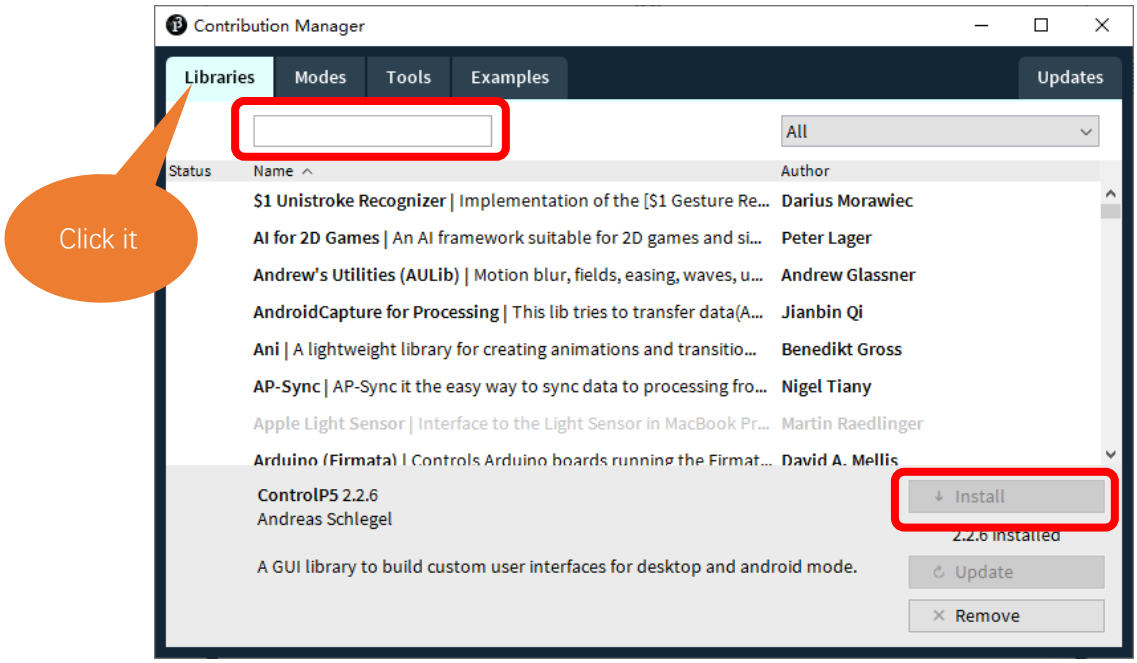
Open Processing.



Click Add Tool under Tools.

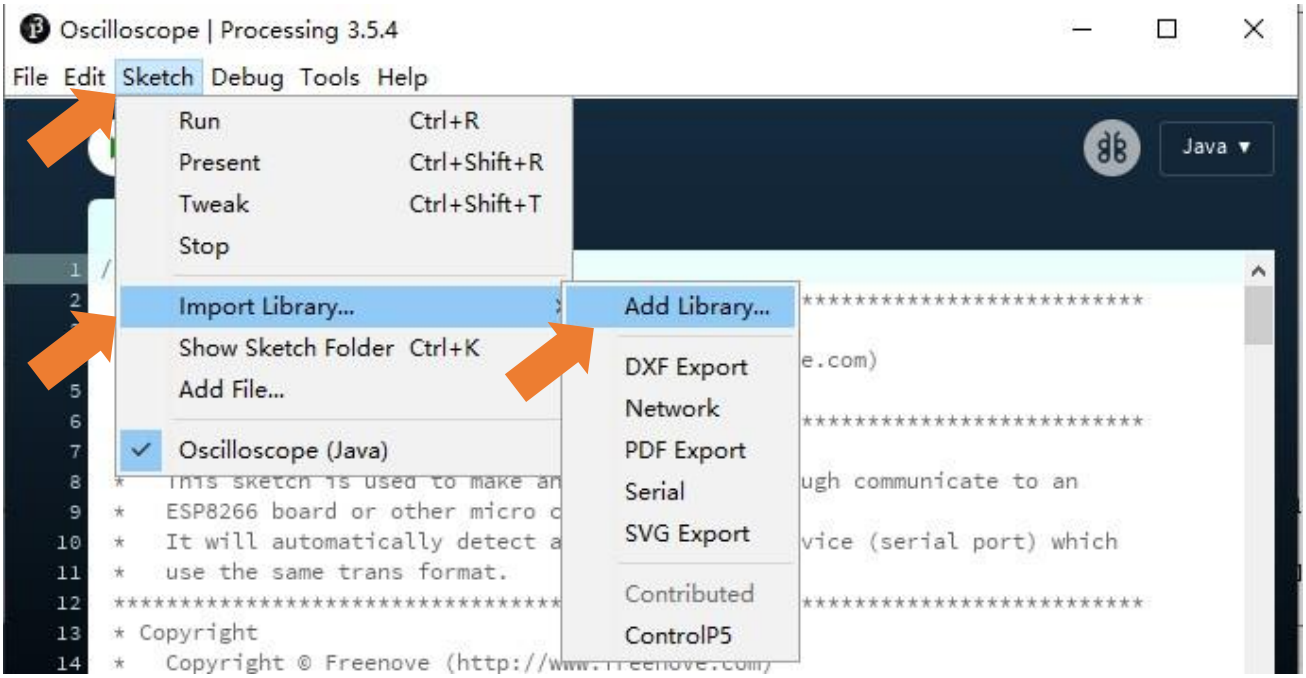


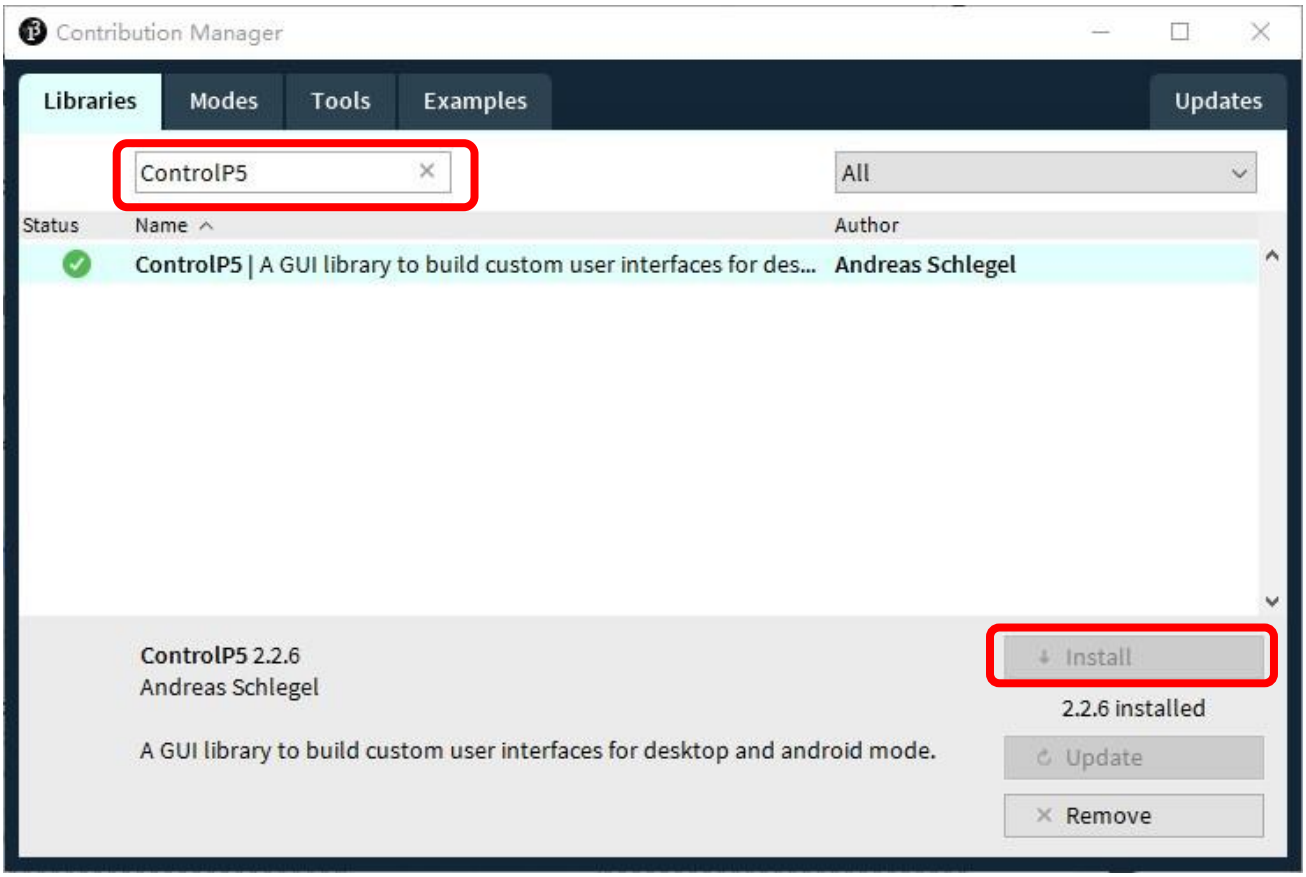
Select Libraries in the pop-up window.



Input "ControlP5" in the searching box, and then select the option as below. Click "Install" and wait for the installation to finish.

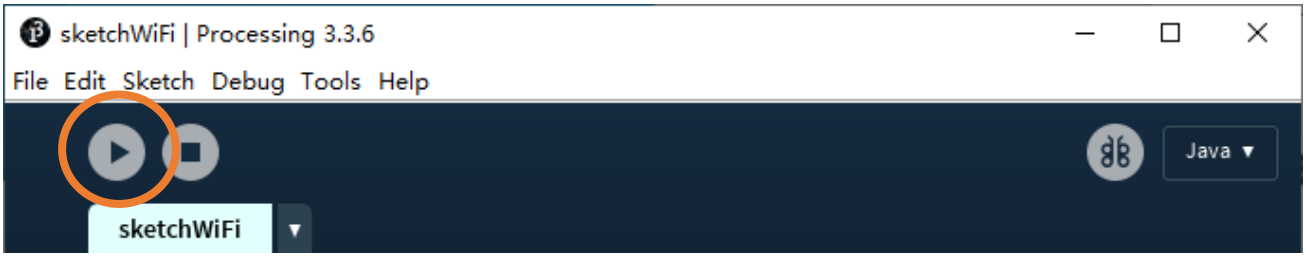
You can also click Add Library under 'Import Library' under 'Sketch'.



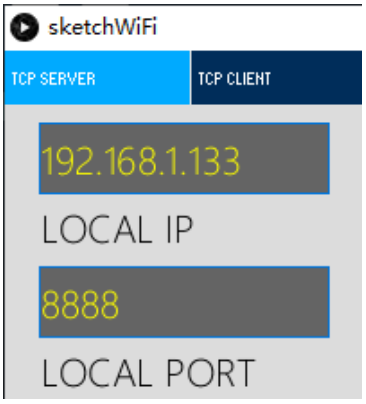


Sketch_04.1_As_Client

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_04.1_WiFiClient.ino. Before running it, please change the following information based on "LOCAL IP" and "LOCAL PORT" in the figure above.

Sketch_29.1_WiFiClient | Arduino 1.8.19

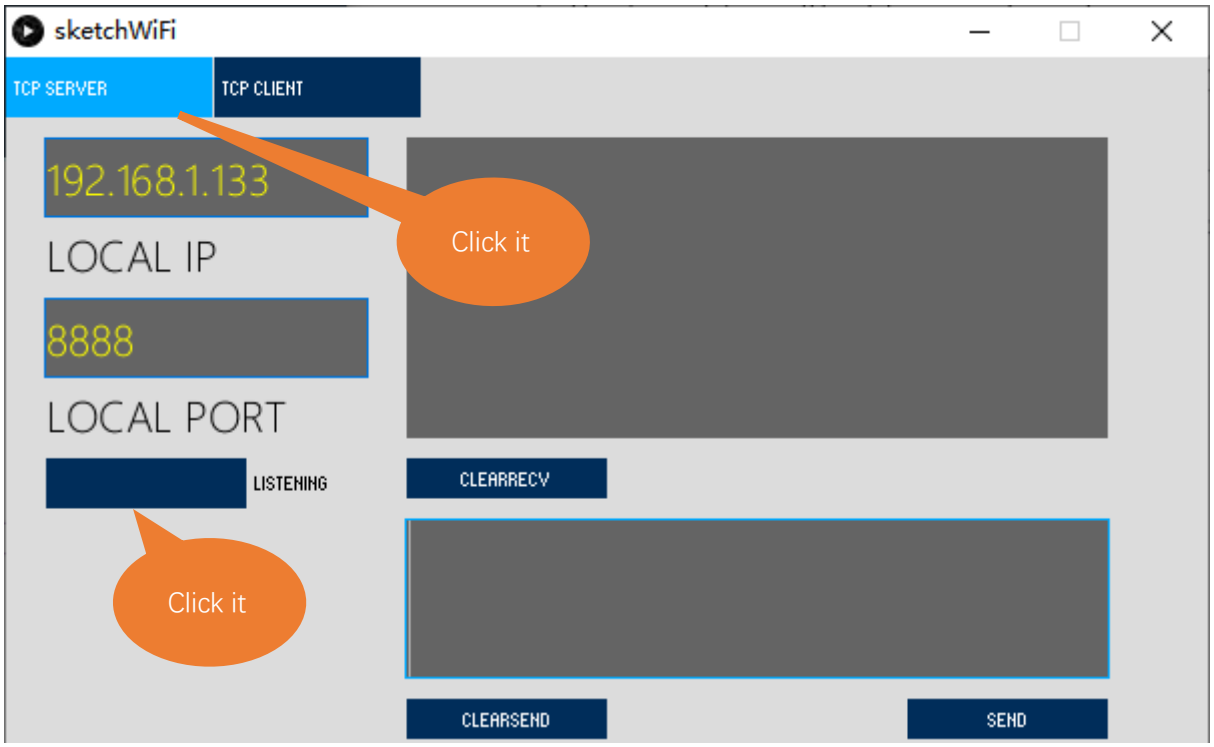
File Edit Sketch Tools Help

Sketch_29.1_WiFiClient

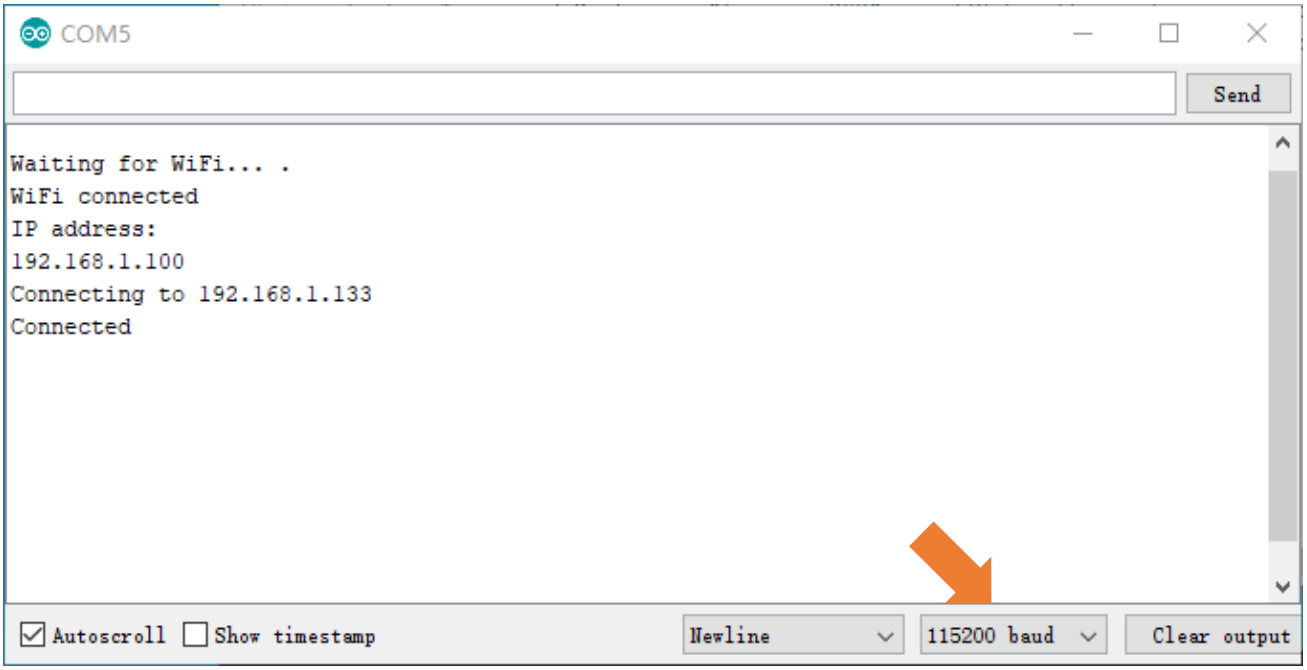
```
#include <ESP8266WiFi.h>

const char *ssid_Router     =  "********"; //Enter the router name
const char *password_Router =  "********"; //Enter the router password
#define     REMOTE_IP          "********"  //input the remote server which is you want to connect
#define     REMOTE_PORT         8888       //input the remote port which is the remote provide
WiFiClient client;

void setup() {
  Serial.begin(115200);
  delay(10);
```

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.

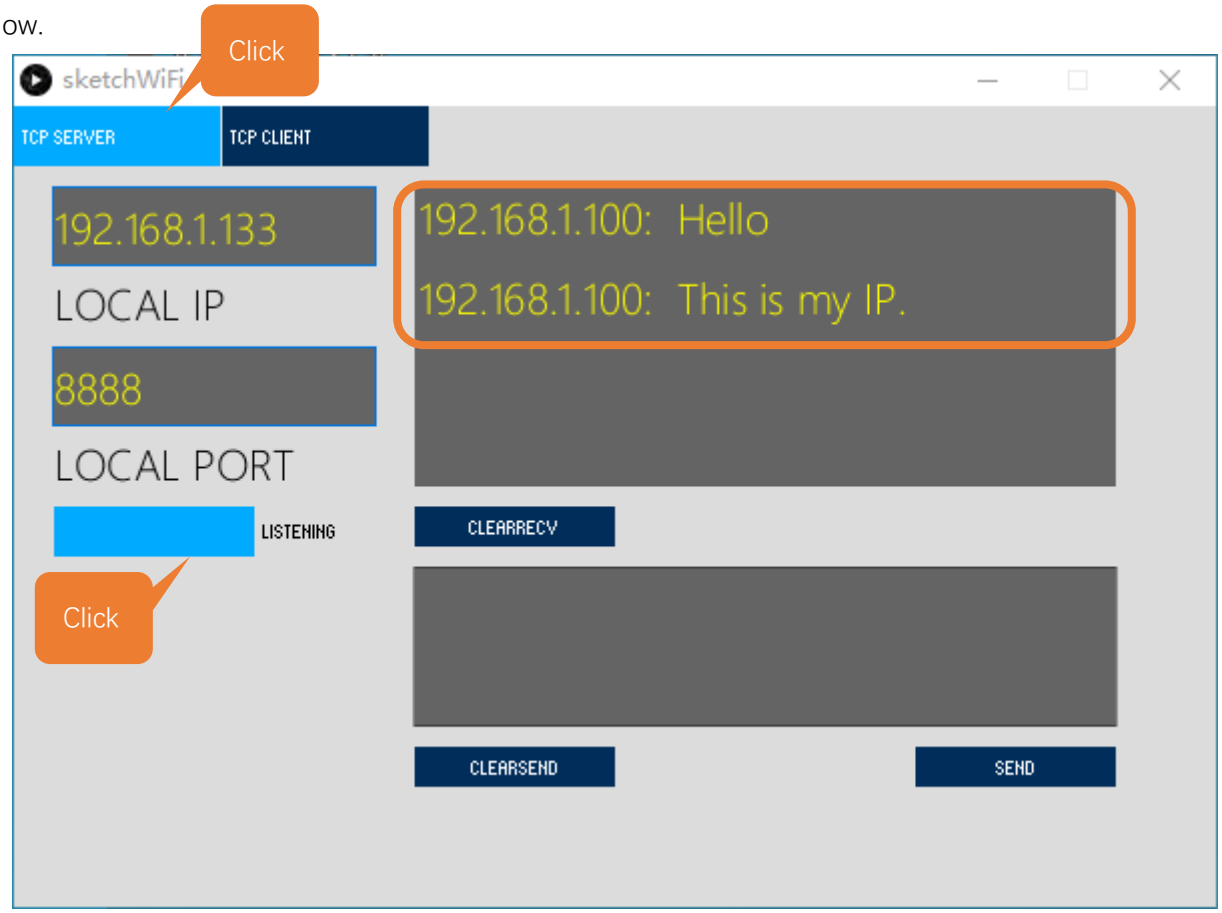
Click LISTENING, turn on TCP SERVER's data listening function and wait for ESP8266 to connect.



Compile and upload code to ESP8266, open the serial monitor and set the baud rate to 115200. ESP8266 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, ESP8266 can send messages to server.


COM5
Send
Waiting for WiFi... .
WiFi connected
IP address:
192.168.1.100
Connecting to 192.168.1.133
Connected
Autoscroll
Show timestamp
Newline
115200 baud
Clear output

ESP8266 connects with TCP SERVER, and TCP SERVER receives messages from ESP8266, as shown in the figure below.



The following is the program code:

```
#include <ESP8266WiFi.h>

const char *ssid_Router     =  "********"; //Enter the router name
const char *password_Router =  "********"; //Enter the router password
#define     REMOTE_IP       "********"  //input the remote server which is you want to connect
#define     REMOTE_PORT      8888       //input the remote port which is the remote provide
WiFiClient client;

void setup() {
  Serial.begin(115200);
  delay(10);

  WiFi.begin(ssid_Router, password_Router);
  Serial.print("\nWaiting for WiFi... ");
  while (WiFi.status() != WL_CONNECTED) {
    Serial.print(".");
    delay(500);
  }
  Serial.println("");
  Serial.println("WiFi connected");
  Serial.println("IP address: ");
  Serial.println(WiFi.localIP());
  delay(500);

  Serial.print("Connecting to ");
  Serial.println(REMOTE_IP);

  while (!client.connect(REMOTE_IP, REMOTE_PORT)) {
    Serial.println("Connection failed.");
    Serial.println("Waiting a moment before retrying...");
  }
  Serial.println("Connected");
  client.print("Hello\n");
  client.print("This is my IP.\n");
}

void loop() {
  if (client.available() > 0) {
    delay(20);
    //read back one line from the server
    String line = client.readString();
    Serial.println(REMOTE_IP + String(":") + line);
  }
```

```
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 <ESP8266WiFi.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 ESP8266 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 ESP8266.

```
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 "ESP8266WiFi.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.

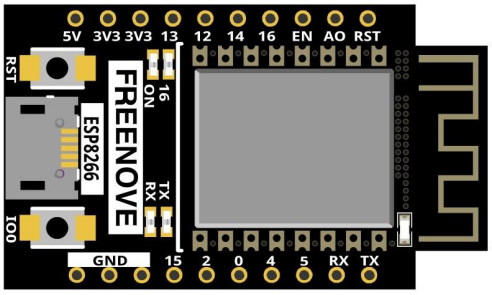
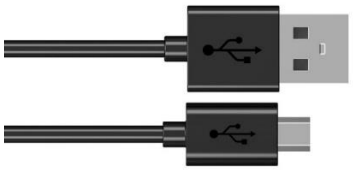
read(): read one byte of data in receive buffer

readString(): read string in receive buffer

Project 4.2 As Server

In this section, ESP8266 is used as a server to wait for the connection and communication of client on the same LAN.

Component List

ESP8266 x1 	Micro USB Wire x1 

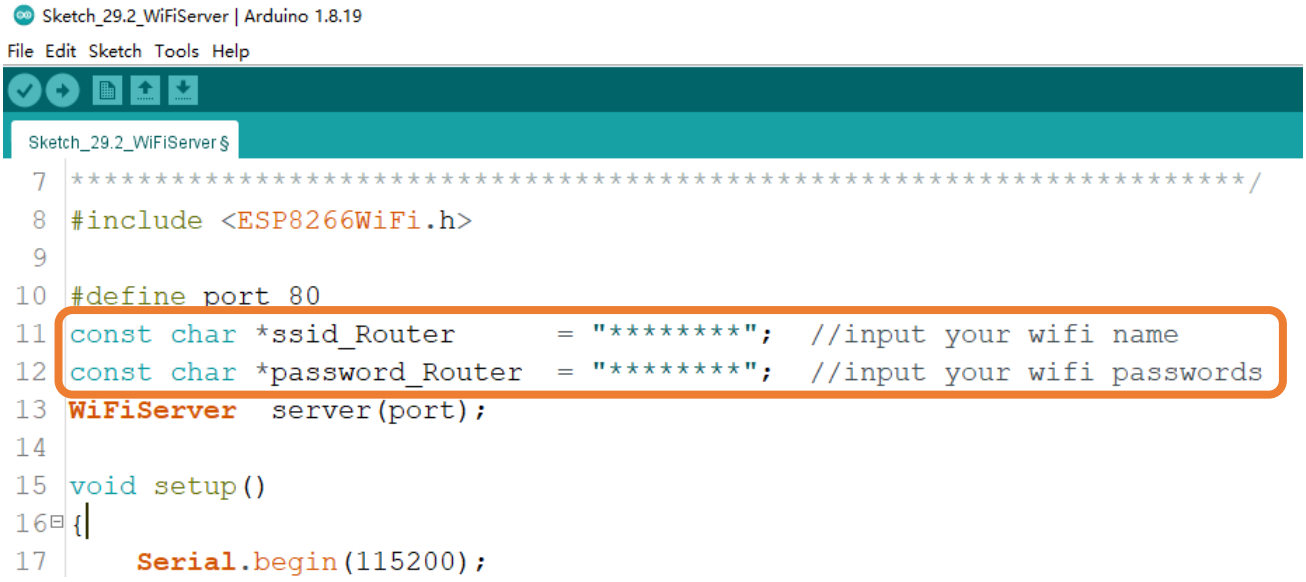
Circuit

Connect Freenove ESP8266 to the computer using a USB cable.

Sketch

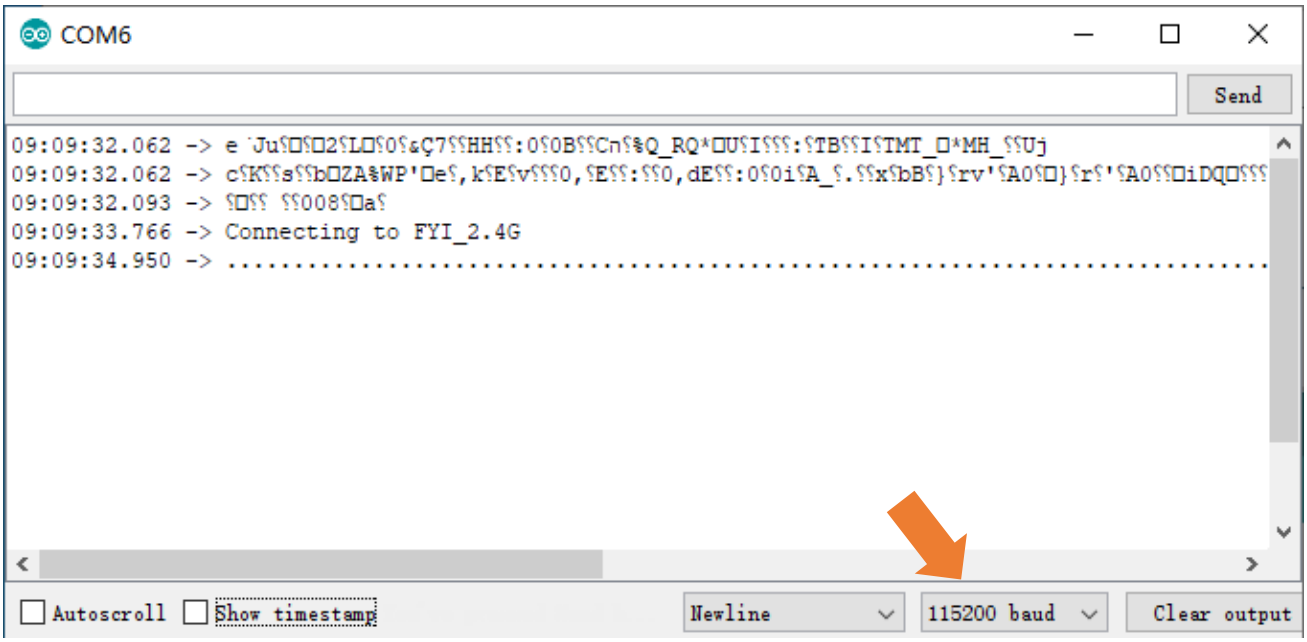
Before running Sketch, please modify the contents of the box below first.

Sketch_04.2_As_Server

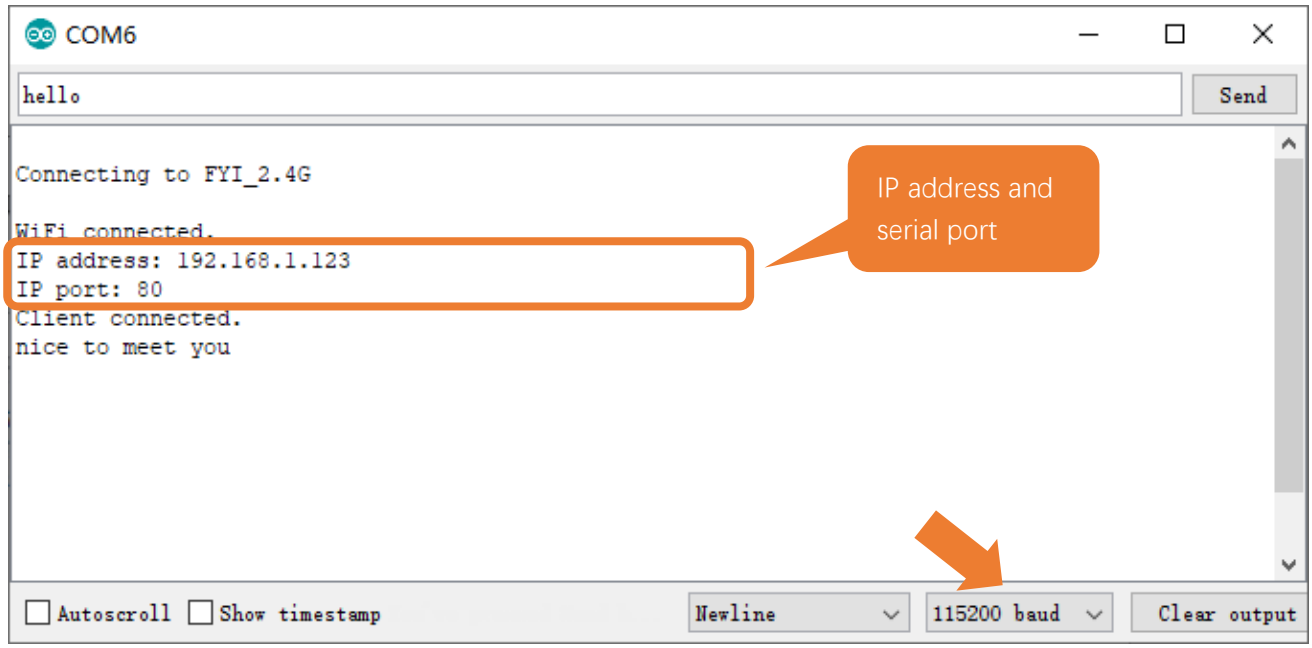


Compile and upload code to ESP8266 board, open the serial monitor and set the baud rate to 115200. Turn on server mode for ESP8266, waiting for the connection of other devices on the same LAN. Once a device connects to server successfully, they can send messages to each other.

If the ESP8266 fails to connect to router, press the reset button as shown below and wait for ESP8266 to run again.



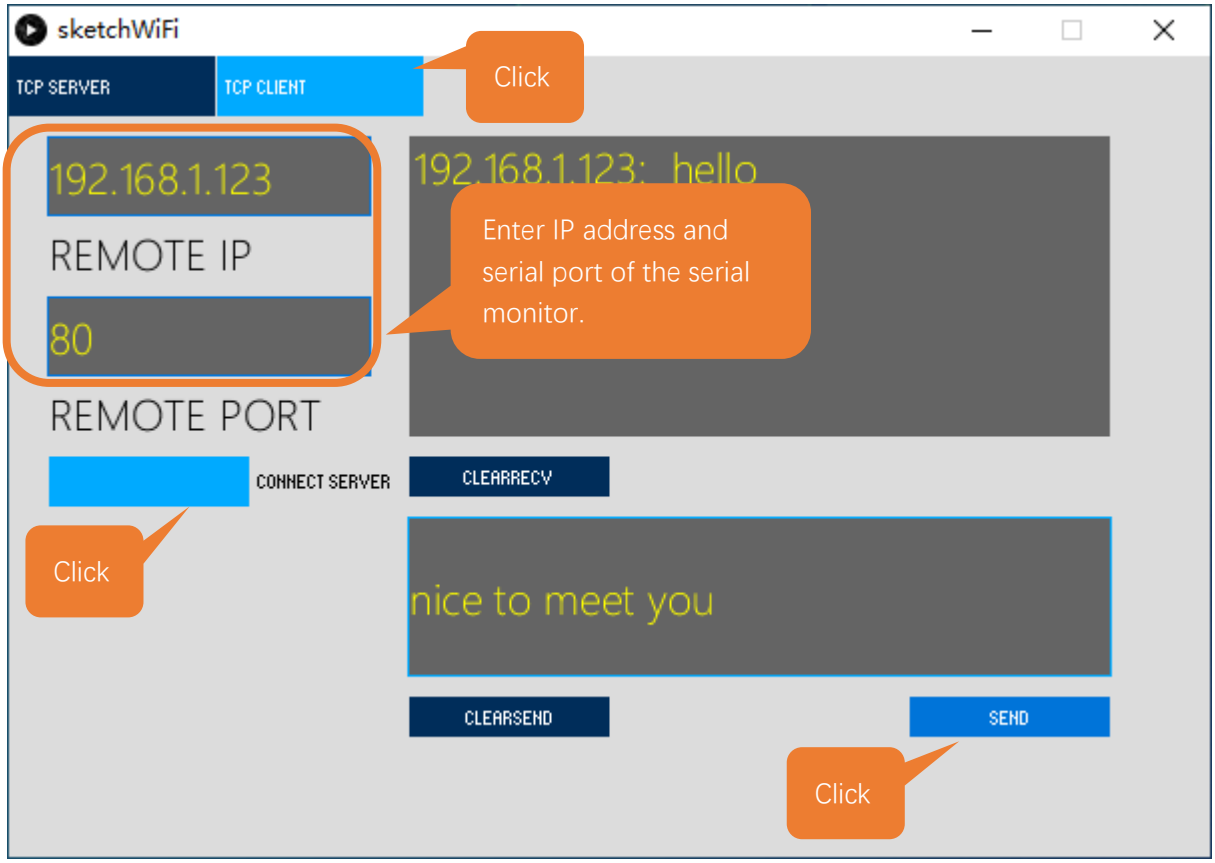
Serial Monitor



Processing:

Open the "**Freenove_ESP8266_Board\C\Sketches\Sketch_04.2_WiFiServer\sketchWiFi\sketchWiFi.pde**".

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



The following is the program code:

```
#include <ESP8266WiFi.h>

#define port 80
const char *ssid_Router      = "********";  //input your wifi name
const char *password_Router  = "********";  //input your wifi passwords
WiFiServer  server(port);

void setup()
{
    Serial.begin(115200);
    Serial.printf("\nConnecting to ");
    Serial.println(ssid_Router);
    WiFi.disconnect();
    WiFi.begin(ssid_Router, password_Router);
    delay(1000);
    while (WiFi.status() != WL_CONNECTED) {
        delay(500);
        Serial.print(".");
    }
    Serial.println("");
    Serial.println("WiFi connected.");
    Serial.print("IP address: ");
    Serial.println(WiFi.localIP());
    Serial.printf("IP port: %d\n",port);
    server.begin(port);
    WiFi.setAutoConnect(true);
    WiFi.setAutoReconnect(true);
}

void loop(){
 WiFiClient client = server.available();            // listen for incoming clients
  if (client) {                                     // if you get a client
    Serial.println("Client connected.");
    while (client.connected()) {                    // loop while the client's connected
      if (client.available()) {                     // if there's bytes to read from the
client
        Serial.println(client.readStringUntil('\n')); // print it out the serial monitor
        while(client.read()>0);                     // clear the wifi receive area cache
      }
      if(Serial.available()){                       // if there's bytes to read from the
serial monitor
        client.print(Serial.readStringUntil('\n')); // print it out the client.
        while(Serial.read()>0);                     // clear the wifi receive area cache
```

```
42        }
43      }
44      client.stop();                                  // stop the client connecting.
45      Serial.println("Client Disconnected.");
46    }
47  }
```

Apply for method class of WiFiServer.

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

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

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

Print out the IP address and port number of ESP8266.

```
22      Serial.print("IP address: ");
23      Serial.println(WiFi.localIP());                  //print out IP address of ESP8266
24      Serial.printf("IP port: %d\n",port);             //Print out ESP8266's port number
```

Turn on server mode of ESP8266, start automatic connection and turn on automatic reconnection.

```
25      server.begin();                                  //Turn ON ESP8266 as Server mode
26      WiFi.setAutoConnect(true);
27      WiFi.setAutoReconnect(true);
```

When ESP8266 receive messages from servers, it will print them out via serial port; Users can also send messages to servers from serial port.

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

Reference

Class Server

Every time use Server functionality, we need to include header file"ESP8266WiFi.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.

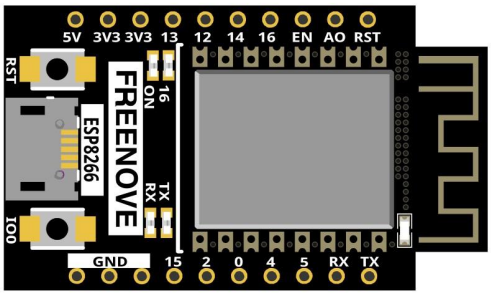
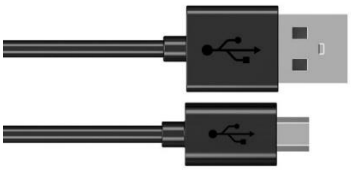
Chapter 5 Smart home

In this chapter, we will use ESP8266 to make a simple smart home.We will learn how to control LED lights through web pages.

Project 5.1 Control_LED_through_Web

In this project, we need to build a Web Service and then use ESP8266 to control the LED through the Web browser of the phone or PC.Through this example, you can remotely control the appliances in your home to achieve smart home.

Component List

ESP8266 x1 	Micro USB Wire x1 

Component knowledge

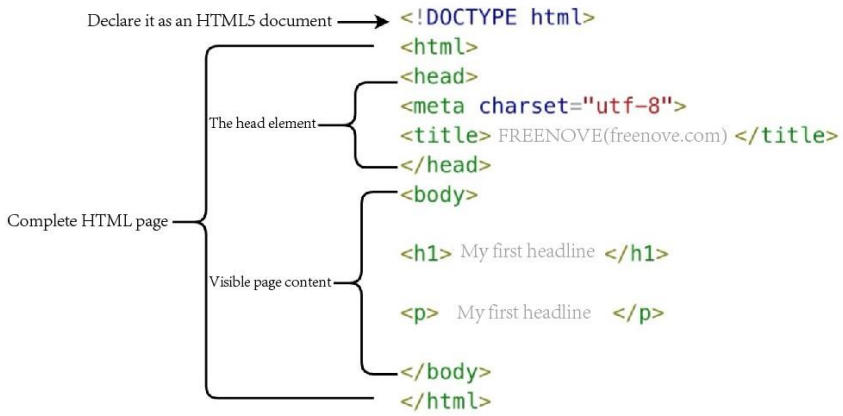
HTML

HyperText Markup Language (HTML) is a standard Markup Language for creating web pages.It includes a set of tags that unify documents on the network and connect disparate Internet resources into a logical whole.HTML text is descriptive text composed of HTML commands that describe text, graphics, animations, sounds, tables, links, etc.The extension of the HTML file is HTM or HTML.Hyper Text is a way to organize information.It uses hyperlinks to associate words and charts in Text with other information media.These related information media may be in the same Text, other files, or files located on a remote computer.This way of organizing information connects the information resources distributed in different places, which is convenient for people to search and retrieve information.

The nature of the Web is hypertext Markup Language (HTML), which can be combined with other Web technologies (e.g., scripting languages, common gateway interfaces, components, etc.) to create powerful Web pages. Thus, HYPERtext Markup Language (HTML) is the foundation of World Wide Web (Web) programming, that is, the World Wide Web is based on hypertext. Hypertext Markup Language is called hypertext Markup language because the text contains so-called "hyperlink" points.

You can build your own WEB site using HTML, which runs on the browser and is parsed by the browser.

Example analysis is shown in the figure below:



<!DOCTYPE html>: Declare it as an HTML5 document

<html>: Is the root element of an HTML page

<head>: Contains meta data for the document, such as < meta charset="utf-8"> Define the web page encoding format to UTF-8.

<title>: Notesthe title of the document

<body>: Contains visible page content

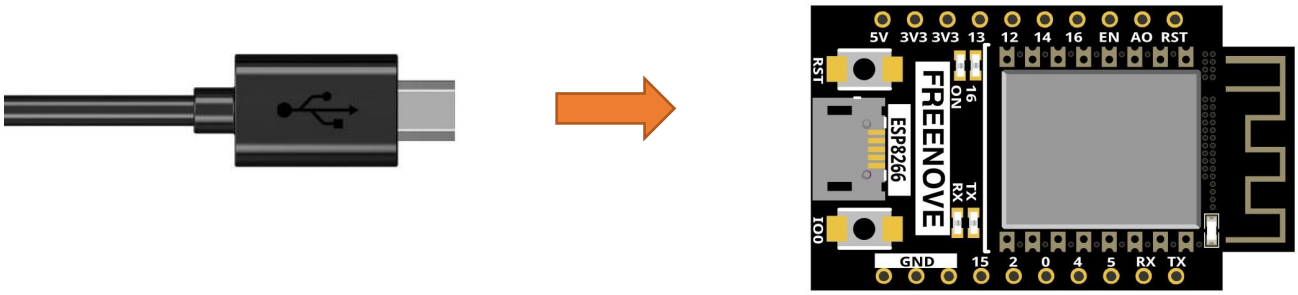
<h1>: Define a big heading

<p>: Define a paragraph

For more information, please visit: https://developer.mozilla.org/en-US/docs/Web/HTML

Circuit

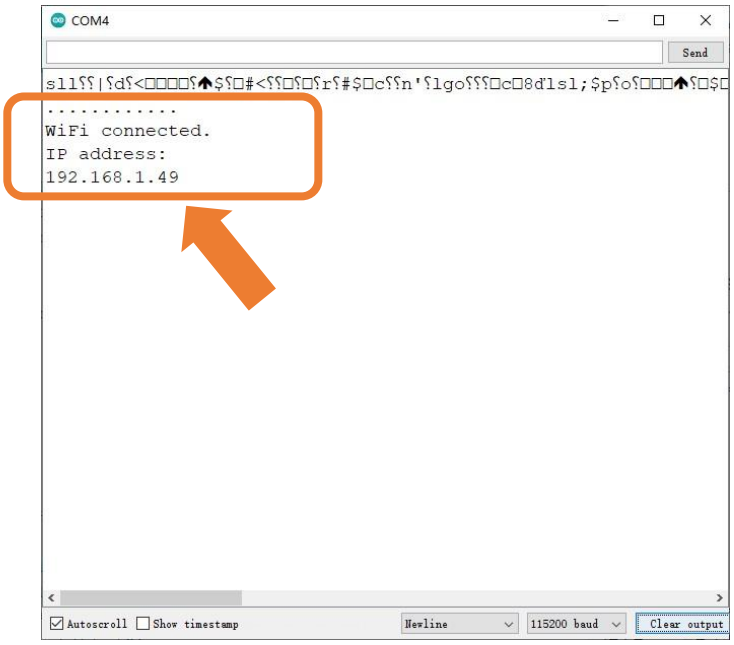
Connect Freenove ESP8266 to the computer using a USB cable.



Sketch

Sketch_05.1_Control_LED_through_Web

Download the code to ESP8266, open the serial port monitor, set the baud rate to 115200 and you can use it to measure the distance between the ultrasonic module and the object. As shown in the following figure:



When ESP8266 successfully connects to "ssid_Router", serial monitor will print out the IP address assigned to ESP8266 by the router. Access http://192.168.1.49 in a computer browser on the LAN. As shown in the following figure:

You can click the corresponding button to control the LED on and off.

The following is the program code:

```
1  #include <ESP8266WiFi.h>
2  // Replace with your network credentials
3  const char* ssid     = "********";  //Enter the router name
4  const char* password = "********";  //Enter the router password
5  // Set web server port number to 80
```

```
6   WiFiServer server(80);
7   // Variable to store the HTTP request
8   String header;
9   // Auxiliar variables to store the current output state
10  String PIN_LEDState = "OFF";
11  // Assign output variables to GPIO pins
12  const int PIN_LED = 2;
13  // Current time
14  unsigned long currentTime = millis();
15  // Previous time
16  unsigned long previousTime = 0;
17  // Define timeout time in milliseconds (example: 2000ms = 2s)
18  const long timeoutTime = 2000;
19
20  void setup() {
21    Serial.begin(115200);
22    // Initialize the output variables as outputs
23    pinMode(PIN_LED, OUTPUT);
24    // Set outputs to LOW
25    digitalWrite(PIN_LED, HIGH);
26    // Connect to Wi-Fi network with SSID and password
27    Serial.print("Connecting to ");
28    Serial.println(ssid);
29    WiFi.begin(ssid, password);
30    while (WiFi.status() != WL_CONNECTED) {
31      delay(500);
32      Serial.print(".");
33    }
34    // Print local IP address and start web server
35    Serial.println("");
36    Serial.println("WiFi connected.");
37    Serial.println("IP address: ");
38    Serial.println(WiFi.localIP());
39    server.begin();
40  }
41  void loop(){
42    WiFiClient client = server.available();   // Listen for incoming clients
43    if (client) {                             // If a new client connects,
44      Serial.println("New Client.");          // print a message out in the serial port
45      String currentLine = "";                // make a String to hold incoming data from the client
46      currentTime = millis();
47      previousTime = currentTime;
48      while (client.connected() && currentTime - previousTime <= timeoutTime) { // loop while
    the client's connected
```

```
49        currentTime = millis();
50        if (client.available()) {             // if there's bytes to read from the client,
51          char c = client.read();             // read a byte, then
52          Serial.write(c);                    // print it out the serial monitor
53          header += c;
54          if (c == '\n') {                    // if the byte is a newline character
55            // if the current line is blank, you got two newline characters in a row.
56            // that's the end of the client HTTP request, so send a response:
57            if (currentLine.length() == 0) {
58              // HTTP headers always start with a response code (e.g. HTTP/1.1 200 OK)
59              // and a content-type so the client knows what's coming, then a blank line:
60              client.println("HTTP/1.1 200 OK");
61              client.println("Content-type:text/html");
62              client.println("Connection: close");
63              client.println();
64              // turns the GPIOs on and off
65              if (header.indexOf("GET /5/ON") >= 0) {
66                Serial.println("GPIO 5 ON");
67                PIN_LEDState = "ON";
68                digitalWrite(PIN_LED, LOW);
69              } else if (header.indexOf("GET /5/OFF") >= 0) {
70                Serial.println("GPIO 5 OFF");
71                PIN_LEDState = "OFF";
72                digitalWrite(PIN_LED, HIGH);
73              }
74              // Display the HTML web page
75              client.println("<!DOCTYPE html><html>");
76              client.println("<head> <title>ESP8266 Web Server</title> <meta name=\"viewport\"
    content=\"width=device-width, initial-scale=1\">");
77              client.println("<link rel=\"icon\" href=\"data:,\">");
78              // CSS to style the on/off buttons
79    // Feel free to change the background-color and font-size attributes to fit your preferences
80              client.println("<style>html {font-family: Helvetica; display:inline-block; margin:
    0px auto; text-align: center;}");
81              client.println(" h1{color: #0F3376; padding: 2vh;} p{font-size: 1.5rem;}");
82              client.println(".button{background-color: #4286f4; display: inline-block; border:
    none; border-radius: 4px; color: white; padding: 16px 40px;text-decoration: none; font-size:
    30px; margin: 2px; cursor: pointer;}");
83              client.println(".button2{background-color: #4286f4;display: inline-block; border:
    none; border-radius: 4px; color: white; padding: 16px 40px;text-decoration: none; font-size:
    30px; margin: 2px; cursor: pointer;}</style></head>");
84              // Web Page Heading
85              client.println("<body><h1>ESP8266 Web Server</h1>");
86              client.println("<p>GPIO state: " + PIN_LEDState + "</p>");
```

```
87            client.println("<p><a href=\"/5/ON\"><button class=\"button
    button2\">ON</button></a></p>");
88            client.println("<p><a href=\"/5/OFF\"><button class=\"button
    button2\">OFF</button></a></p>");
89            client.println("</body></html>");
90            // The HTTP response ends with another blank line
91            client.println();
92            // Break out of the while loop
93            break;
94          } else { // if you got a newline, then clear currentLine
95            currentLine = "";
96          }
97        } else if (c != '\r') {  // if you got anything else but a carriage return character,
98          currentLine += c;      // add it to the end of the currentLine
99        }
100     }
101   }
102   // Clear the header variable
103   header = "";
104   // Close the connection
105   client.stop();
106   Serial.println("Client disconnected.");
107   Serial.println("");
108 }
109 }
```

Include the WiFi Library header file of ESP8266.

```
1   #include <ESP8266WiFi.h>
```

Enter correct router name and password.

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

Set ESP8266 in Station mode and connect it to your router.

```
29    WiFi.begin(ssid, password);
```

Check whether ESP8266 has connected to router successfully every 0.5s.

```
30    while (WiFi.status() != WL_CONNECTED) {
31      delay(500);
32      Serial.print(".");
33    }
```

Serial monitor prints out the IP address assigned to ESP8266.

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

Click the button on the web page to control the LED light on and off.

```
65            if (header.indexOf("GET /5/ON") >= 0) {
66              Serial.println("GPIO 5 ON");
67              PIN_LEDState = "ON";
68              digitalWrite(PIN_LED, LOW);
```

```
69              } else if (header.indexOf("GET /5/OFF") >= 0) {
70                Serial.println("GPIO 5 OFF");
71                PIN_LEDState = "OFF";
72                digitalWrite(PIN_LED, HIGH);
73              }
```

What's next?

Thanks for your reading. This tutorial is all over here. If you find any mistakes, omissions or you have other ideas and questions about contents of this tutorial or the kit and etc., please feel free to contact us:

support@freenove.com

We will check and correct it as soon as possible.

If you want learn more about ESP8266, you view our ultimate tutorial:
https://github.com/Freenove/Freenove_ESP8266_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.