

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.

Any concerns? ✉ support@freenove.com



About Freenove

Freenove provides open source electronic products and services worldwide.

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

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

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

<http://www.freenove.com>

sale@freenove.com

Copyright

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



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

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



Contents

Important Information.....	1
Contents.....	1
Preface.....	1
ESP8266.....	2
CH340	5
Programming Software	15
Environment Configuration	18
Chapter 0 LED	26
Project 0.1 Blink	26
Chapter 1 LED	32
Project 1.1 Blink	32
Chapter 2 Button & LED	39
Project 2.1 Button & LED	39
Project 2.2 MINI table lamp.....	44
Chapter 3 LED Bar	47
Project 3.1 Flowing Light	47
Chapter 4 Analog & PWM	52
Project 4.1 Breathing LED.....	52
Project 4.2 Meteor Flowing Light.....	60
Chapter 5 RGB LED.....	64
Project 5.1 Random Color Light.....	64
Project 5.2 Gradient Color Light.....	69
Chapter 6 LEDPixel	71
Project 6.1 LEDPixel.....	71
Project 6.2 Rainbow Light.....	78
Chapter 7 Buzzer.....	81
Project 7.1 Doorbell.....	81
Project 7.2 Alertor.....	87
Chapter 8 Serial Communication.....	90

Project 8.1 Serial Print.....	90
Project 8.2 Serial Read and Write	96
Chapter 9 ADC Converter	99
Project 9.1 Read the Voltage of Potentiometer.....	99
Chapter 10 Potentiometer & LED.....	105
Project 10.1 Soft Light	105
Project 10.2 Color Light	108
Project 10.3 Soft Rainbow Light.....	112
Chapter 11 Photoresistor & LED.....	116
Project 11.1 NightLamp	116
Chapter 12 Thermistor	121
Project 12.1 Thermometer	121
Chapter 13 74HC595 & LED Bar Graph.....	127
Project 13.1 Flowing Water Light.....	127
Chapter 14 74HC595 & 7-Segment Display.....	133
Project 14.1 1-Digit 7-Segment Display.....	133
Chapter 15.1 Motor & Driver.....	140
Project 15.1 Control Motor with Potentiometer.....	140
Chapter 16 Servo	148
Project 16.1 Servo Sweep	148
Project 16.2 Servo Knop	154
Chapter 17 LCD1602	158
Project 17.1 LCD1602	158
Chapter 18 Ultrasonic Ranging	166
Project 18.1 Ultrasonic Ranging	166
Project 18.2 Ultrasonic Ranging	173
Chapter 19 Infrared Remote	176
Project 19.1 Infrared Remote Control	176
Project 19.2 Control LED through Infrared Remote	184
Chapter 20 WiFi Working Modes	190
Project 20.1 Station mode.....	190
Project 20.2 AP mode	195

Project 20.3 AP+Station mode	200
Chapter 21 TCP/IP	204
Project 21.1 As Client.....	204
Project 21.2 As Server.....	218
Chapter 22 Smart home	224
Project 22.1 Control_LED_through_Web.....	224
Chapter 23 Play music.....	232
Project 23.1 Play_music.....	232
What's next?	242
End of the Tutorial	242

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 Microcontroller(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.

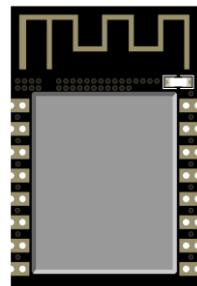
We divide each project into four parts, namely Component List, Component Knowledge, Circuit and Code. Component List helps you to prepare material for the experiment more quickly. Component Knowledge allows you to quickly understand new electronic modules or components, while Circuit helps you understand 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

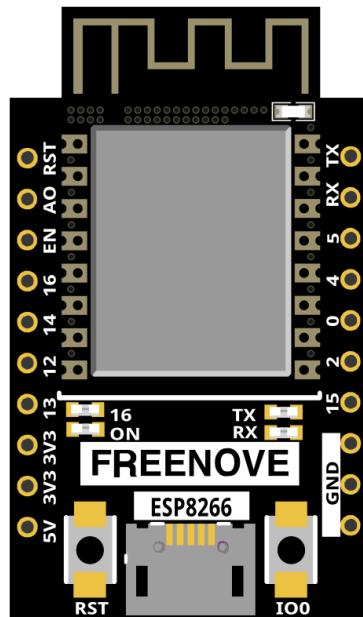
ESP8266 has PCB on-board antenna. The PCB on-board antenna is an integrated antenna in the chip module itself, so it is convenient to carry and design.

PCB on-board antenna

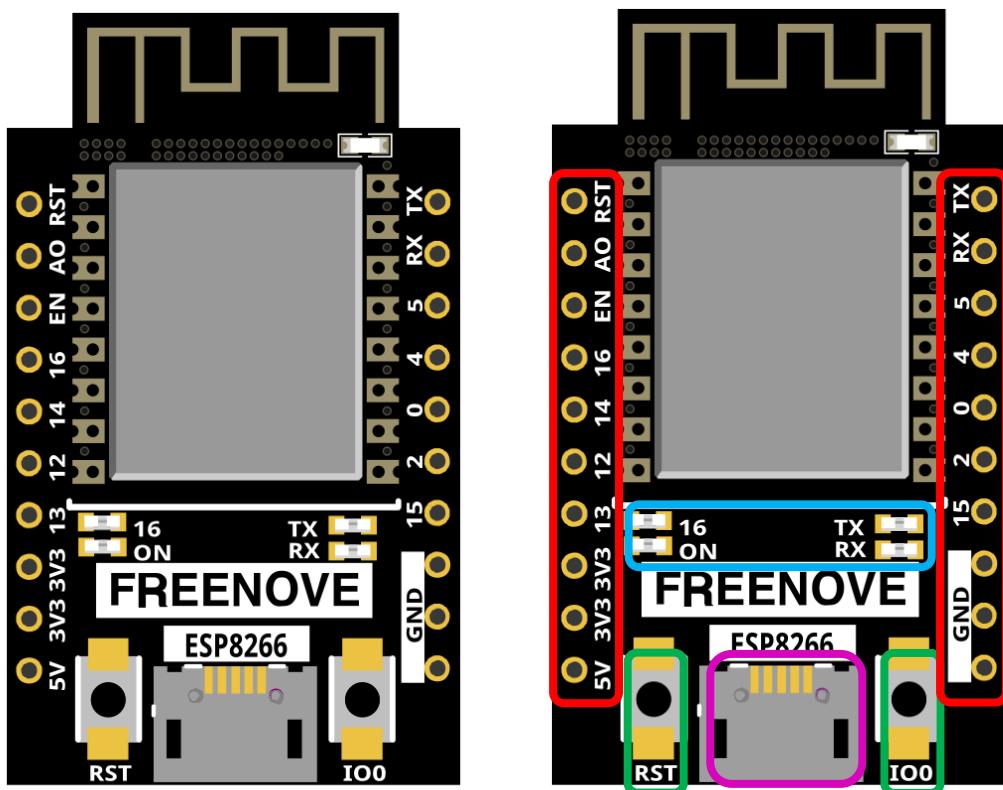


In this tutorial, the ESP8266 development board is designed based on the PCB on-board antenna-packaged ESP8266 module. The following tutorials will be based on the ESP8266 development board.

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 0~1V, the value range is 0~1024.
3	EN	Chip Enabled Pin, Active High
4	IO16	Connect with RST pin to wake up Deep Sleep
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	GPIO2; 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: https://docs.ai-thinker.com/_media/esp8266/docs/esp-12s_product_specification_en.pdf

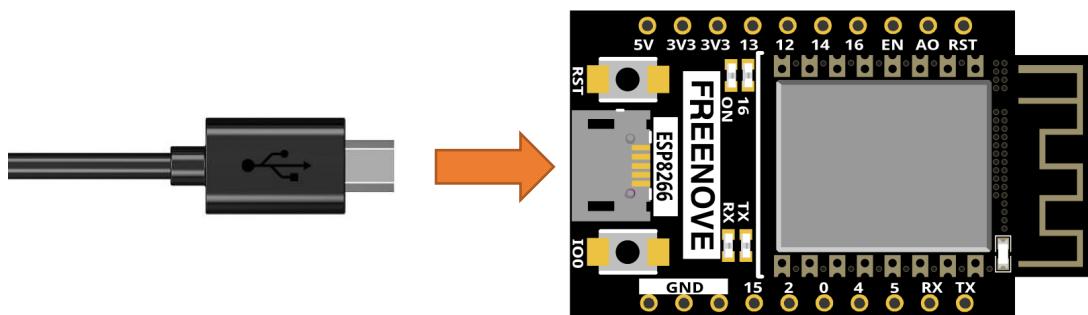
CH340

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

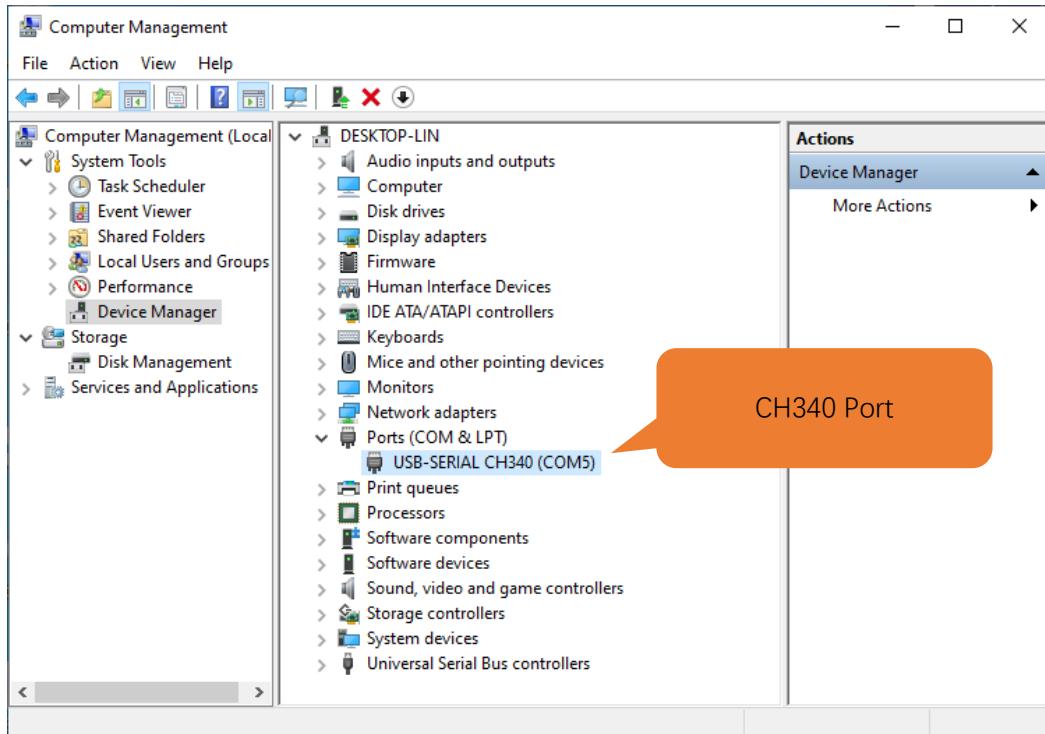
1. Connect your computer and ESP8266 with a USB cable.



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.

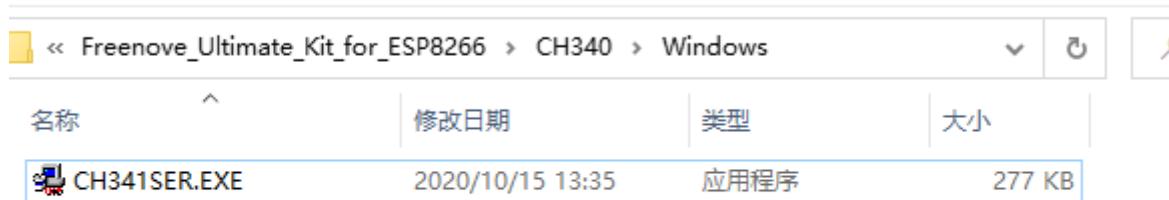
The screenshot shows a search results page for 'CH340' on the Freenove website. The left sidebar has categories: All (14), Downloads (7) (highlighted in blue), Products (4), Application (2), Video (1), and News (0). The main content area is titled 'keyword CH340' and shows 'Downloads(7)'. It lists files under 'file category' Driver&Tools:

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

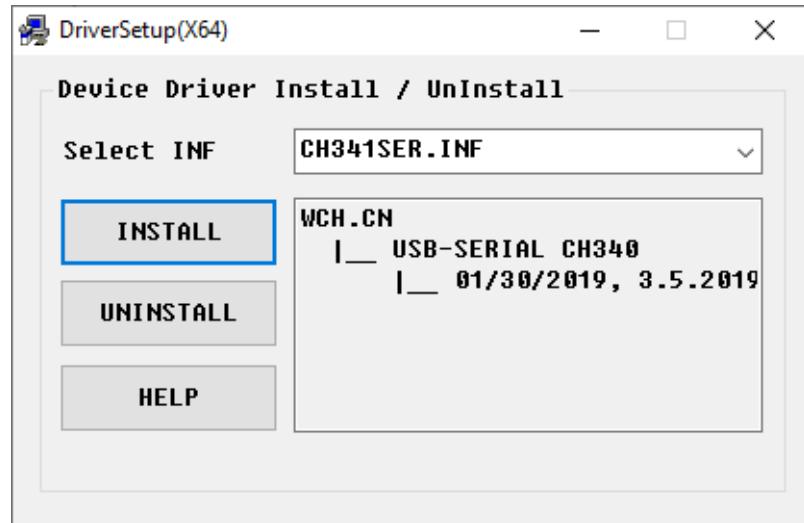
If you would not like to download the installation package, you can open “[Freenove_Super_Starter_Kit_for_ESP8266/CH340](#)”, we have prepared the installation package.

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

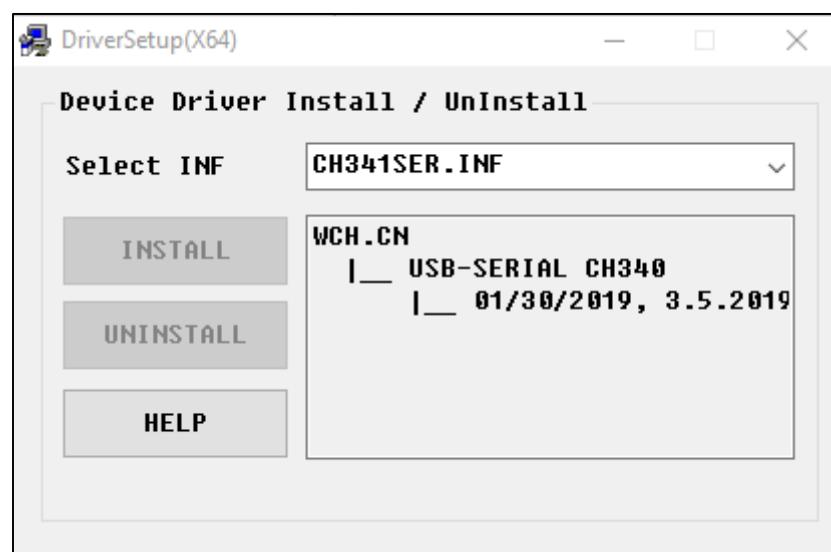
2. Open the folder “Freenove_Super_Starter_Kit_for_ESP8266/CH340/Windows/”



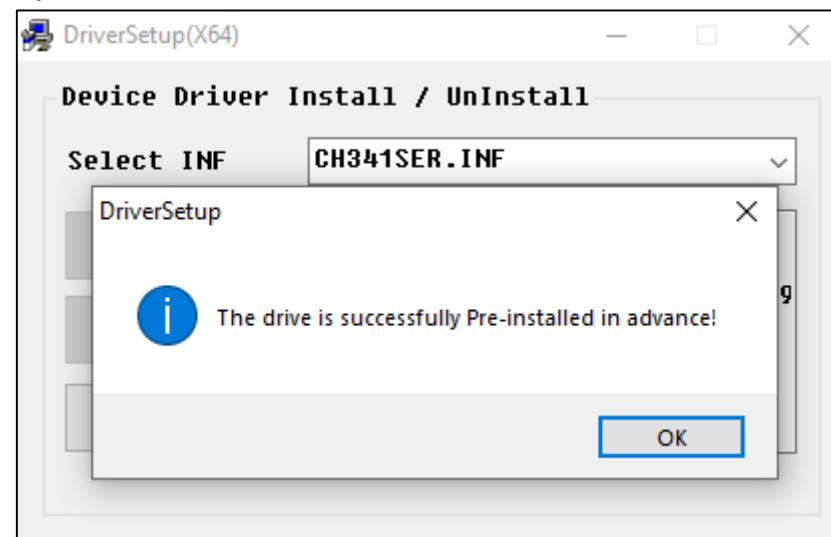
3. Double click “CH341SER.EXE”.



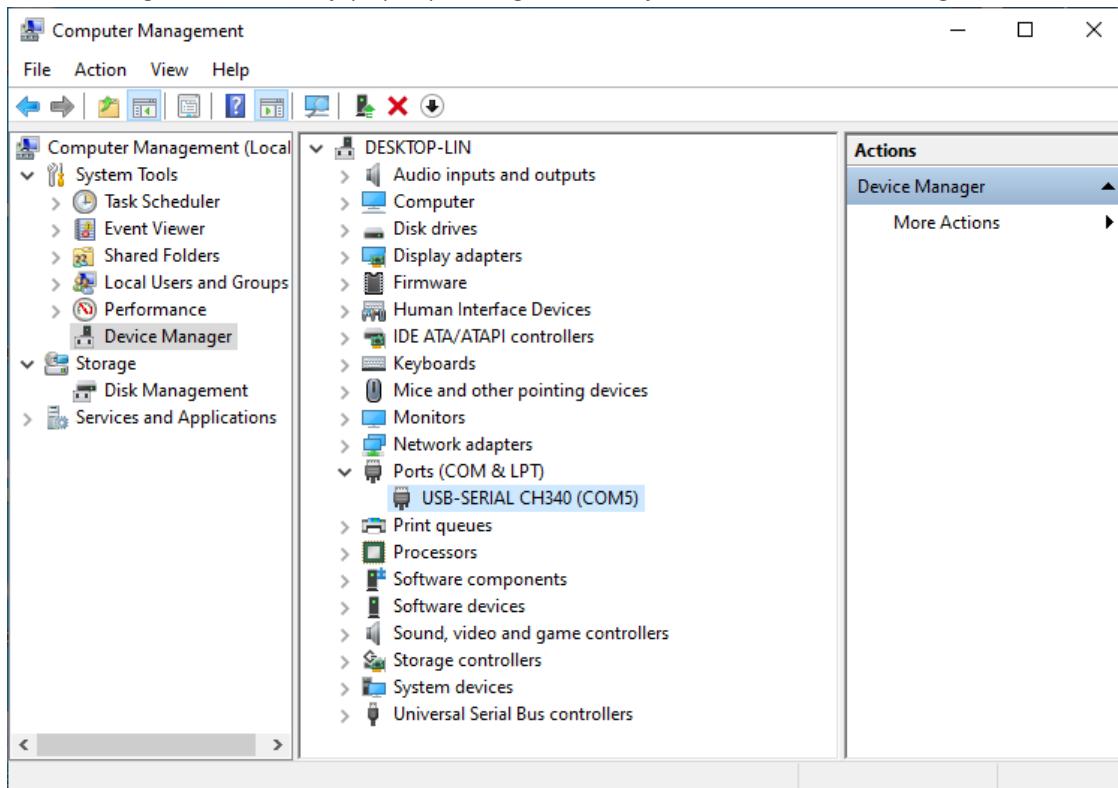
4. Click “INSTALL” and wait for the installation to complete.



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.

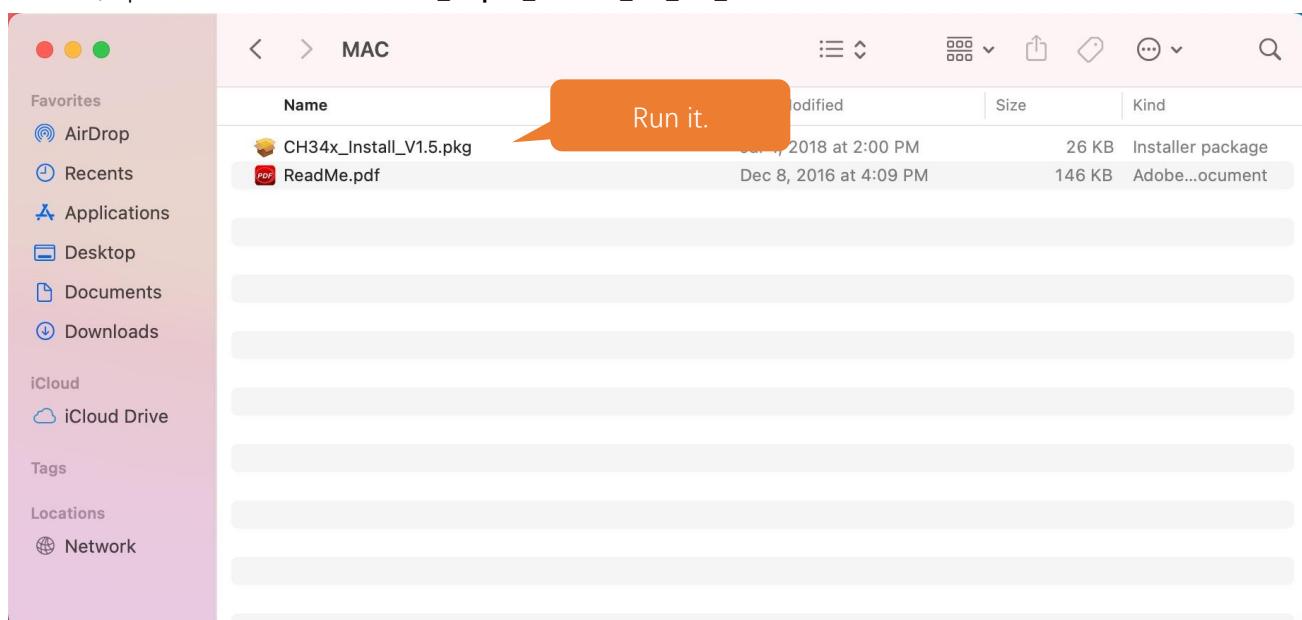
MAC

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.

The screenshot shows a search results page for 'ch340' on the WCH website. The left sidebar has categories: All (14), Downloads (7), Products (4), Application (2), Video (1), and News (0). The main area shows a table of downloads:

file category	file content	version	upload time
Driver&Tools	Windows CH341SER.EXE CH340/CH341 USB to serial port Windows driver, supports 32/64-bit Windows 10/8.1/8/7/VISTA/XP, Server 2016/2012/2008/2003, 2000/ME/98	3.5	2019-03-18
	CH341SER.ZIP CH340/CH341 USB to serial port Windows driver, includes DLL dynamic library and non-standard baud rate settings and other instructions. Supports 32/64-bit Windows 10/8.1/8/7/VISTA/XP, Server 2016/2012/2008/2003, 2000/ME/98	3.5	2019-03-05
	CH341SER_ANDROID... CH340/CH341 USB to serial port Android free drive application library, for Android OS 3.1 and above version which supports USB Host mode already, no need to load Android kernel driver, no root privileges. Contains apk, lib library file (Java Driver), App Demo Examples, and STM32 Demo SDK.	1.6	2019-04-19
	CH341SER_LINUX... CH340/CH341 USB to serial port LINUX driver	1.5	2018-03-18
	CH341SER_MAC.ZI... CH340/CH341 USB to serial port MAC OS driver	1.5	2018-07-05
Others			

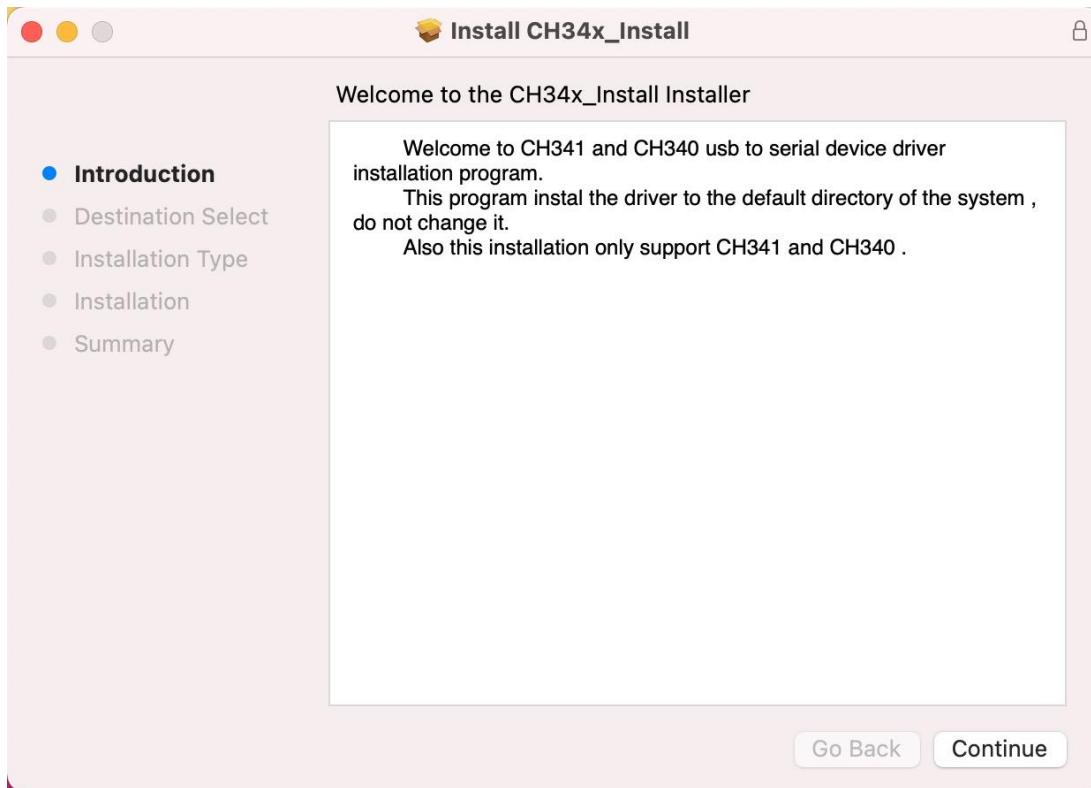
If you would not like to download the installation package, you can open "Freenove_Super_Starter_Kit_for_ESP8266/CH340", we have prepared the installation package. Second, open the folder "Freenove_Super_Starter_Kit_for_ESP8266/CH340/MAC/"



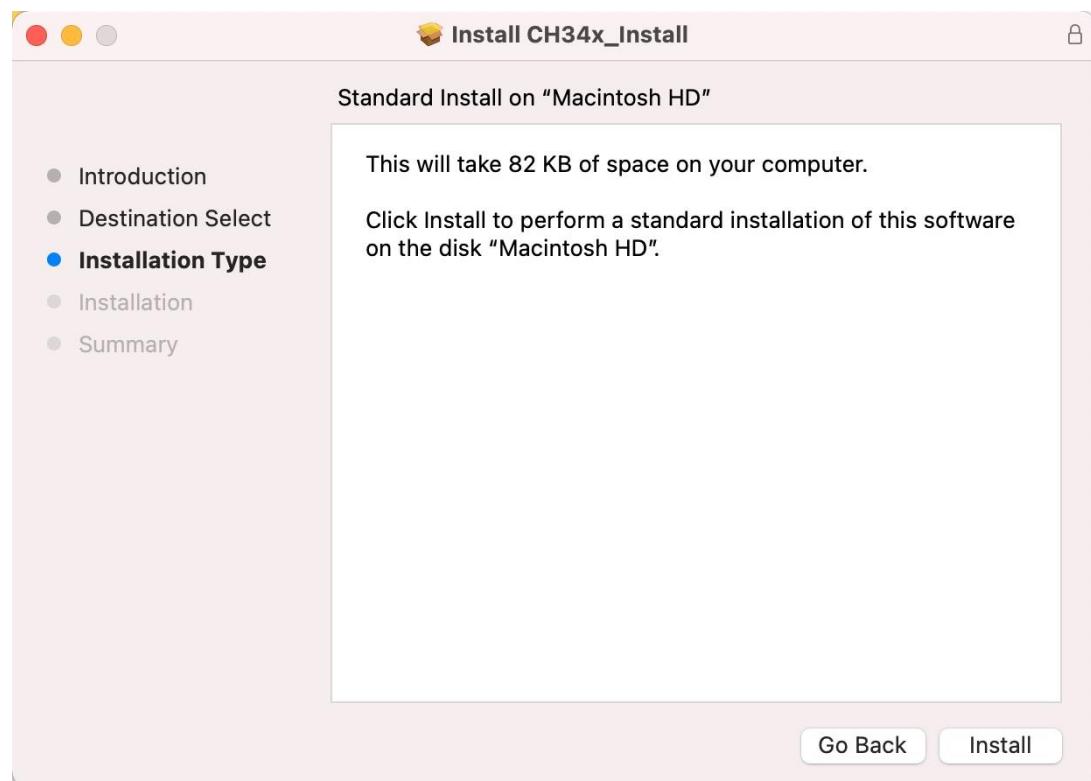
Any concerns? [✉ support@freenove.com](mailto:support@freenove.com)



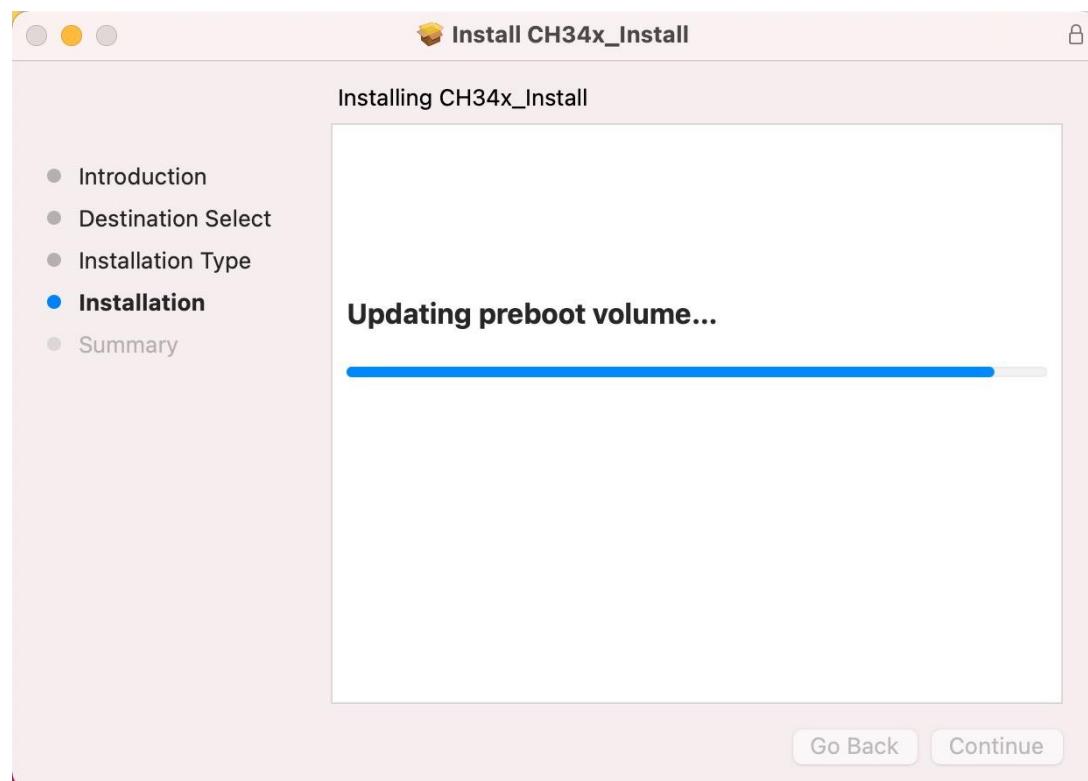
Third, click Continue.



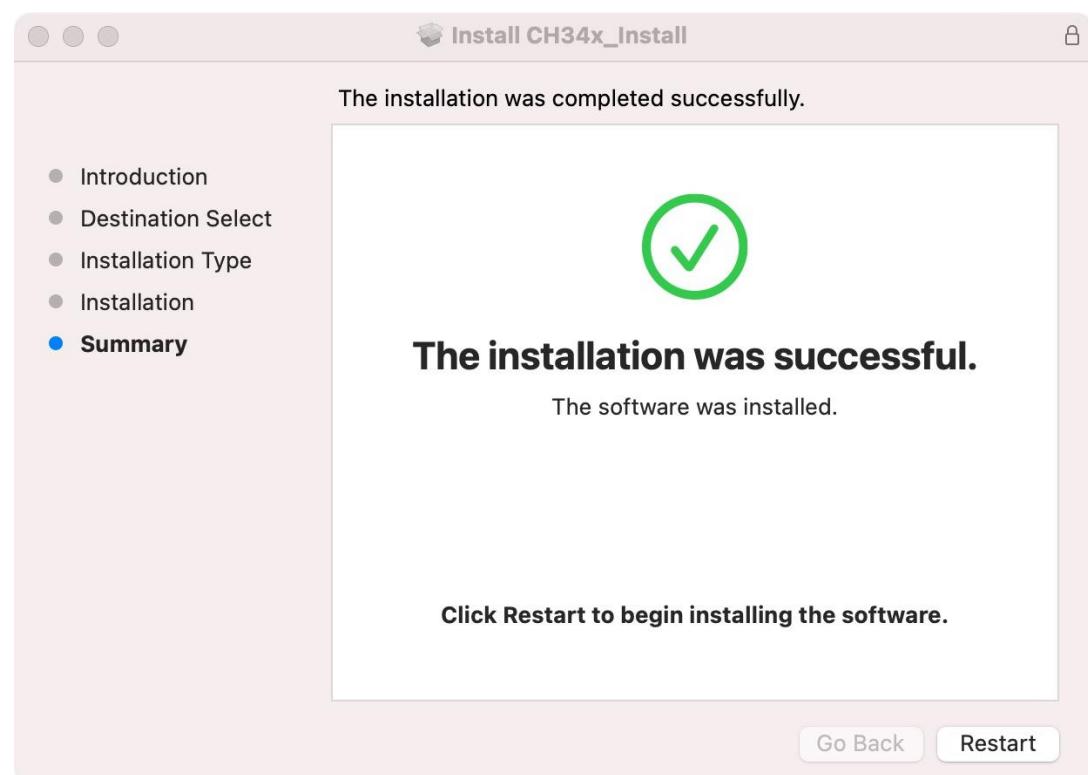
Fourth, click Install.



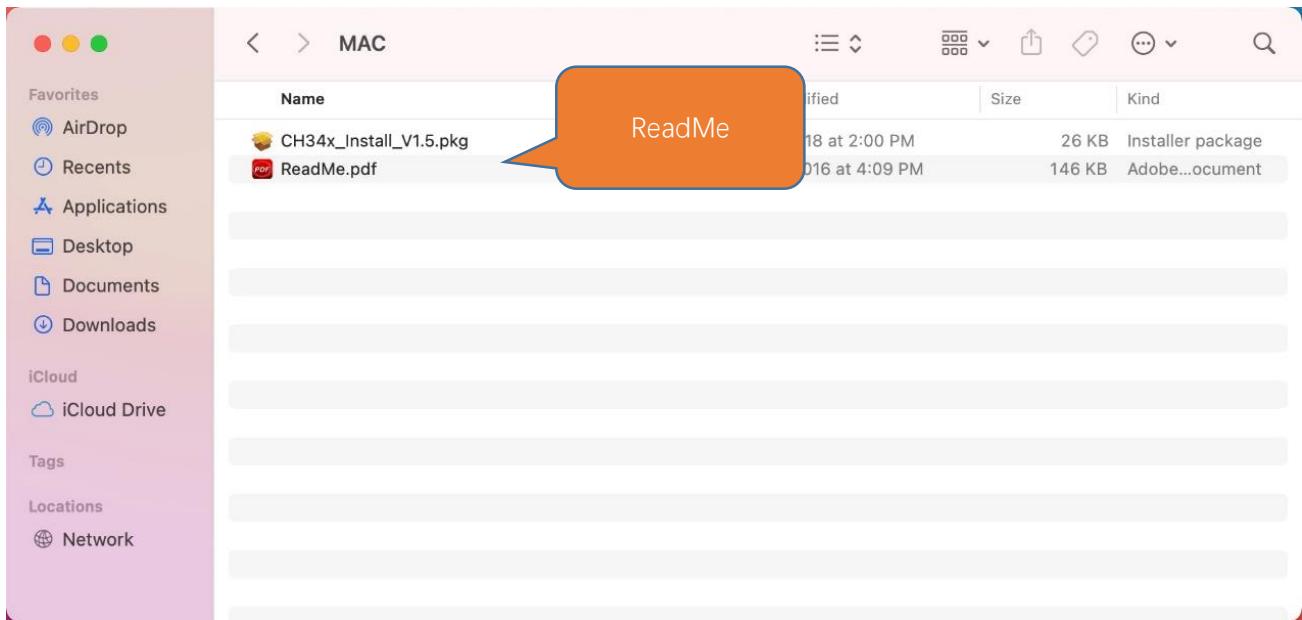
Then, waiting Finsh.



Finally, restart your PC.



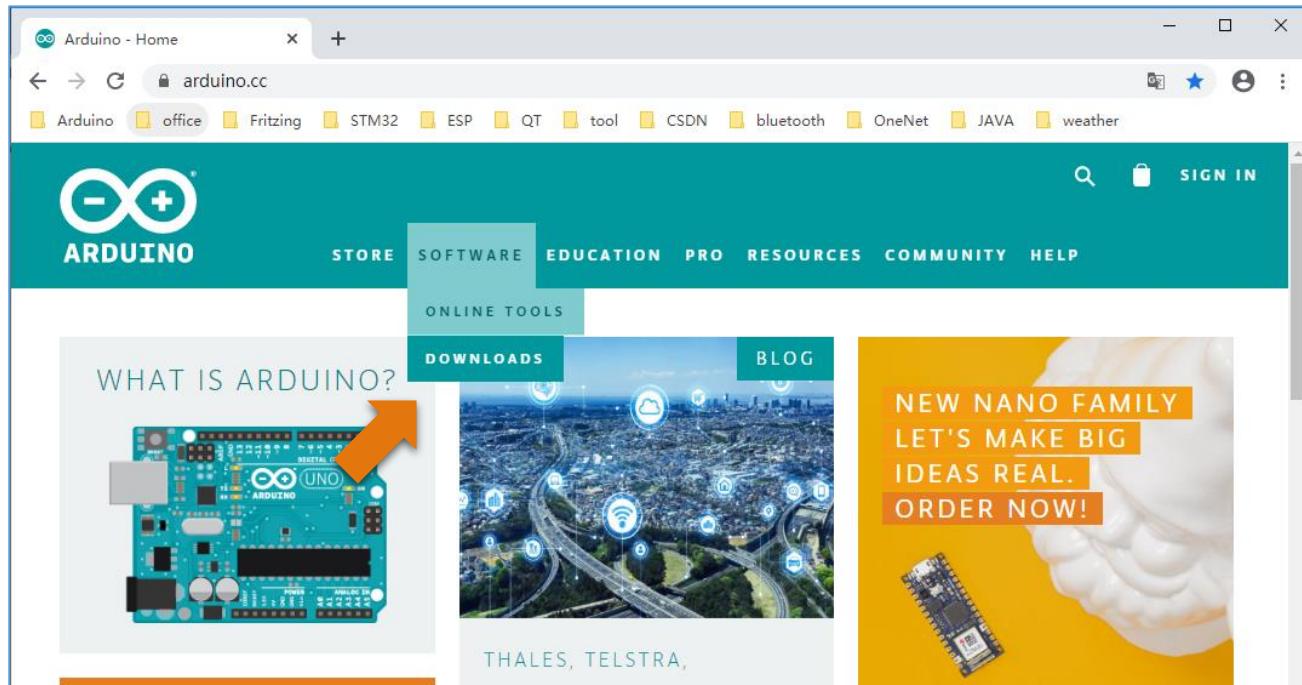
If you still haven't installed the CH340 by following the steps above, you can view `readme.pdf` to install it.



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.

Downloads

A screenshot of the Arduino IDE 1.8.19 download page. The page has a light gray background with a teal sidebar on the right. The main content area features the Arduino logo and the text 'Arduino IDE 1.8.19'. It describes the software as open-source and suitable for any Arduino board. It includes a link to the 'Getting Started' page for installation instructions. The teal sidebar on the right is titled 'DOWNLOAD OPTIONS' and lists download links for Windows (Win 7 and newer, ZIP file), Windows app (Win 8.1 or 10, Get button), Linux (32 bits, 64 bits, ARM 32 bits, ARM 64 bits), Mac OS X (10.10 or newer), Release Notes, and Checksums (sha512). Two orange arrows point from the text above to the 'WHAT IS ARDUINO?' section on the Arduino website and to the 'DOWNLOAD OPTIONS' sidebar.

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.

Any concerns? ✉ support@freenove.com



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.



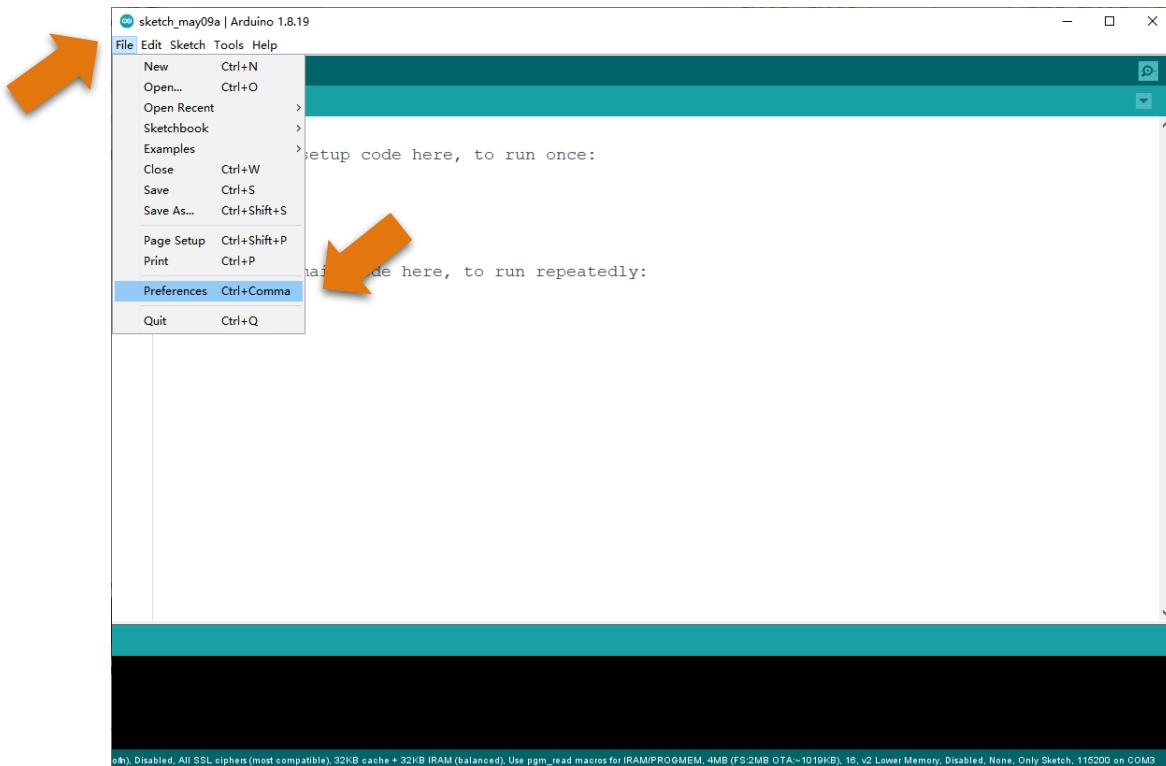
Serial Monitor

Open the serial monitor.

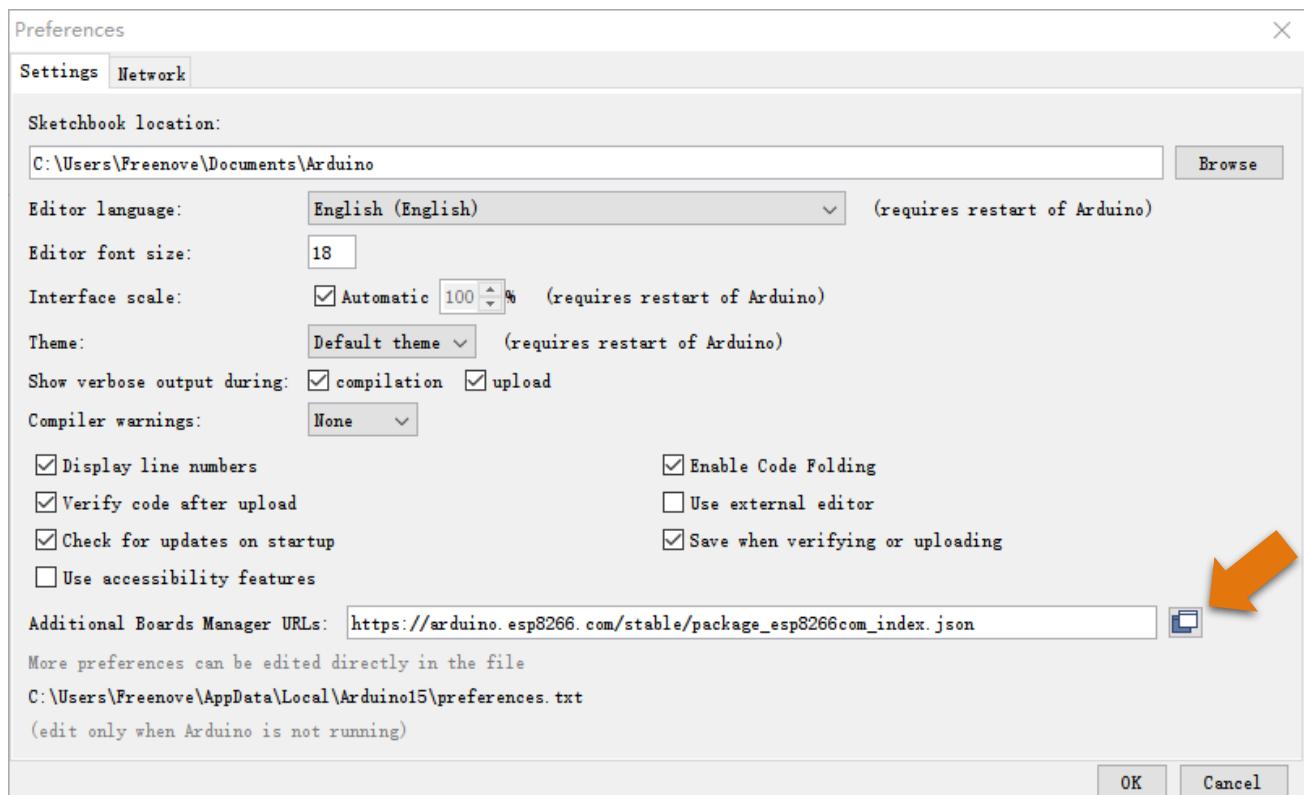
Additional commands are found within the five menus: File, Edit, Sketch, Tools, Help. The menus are context sensitive, which means only those items relevant to the work currently being carried out are available.

Environment Configuration

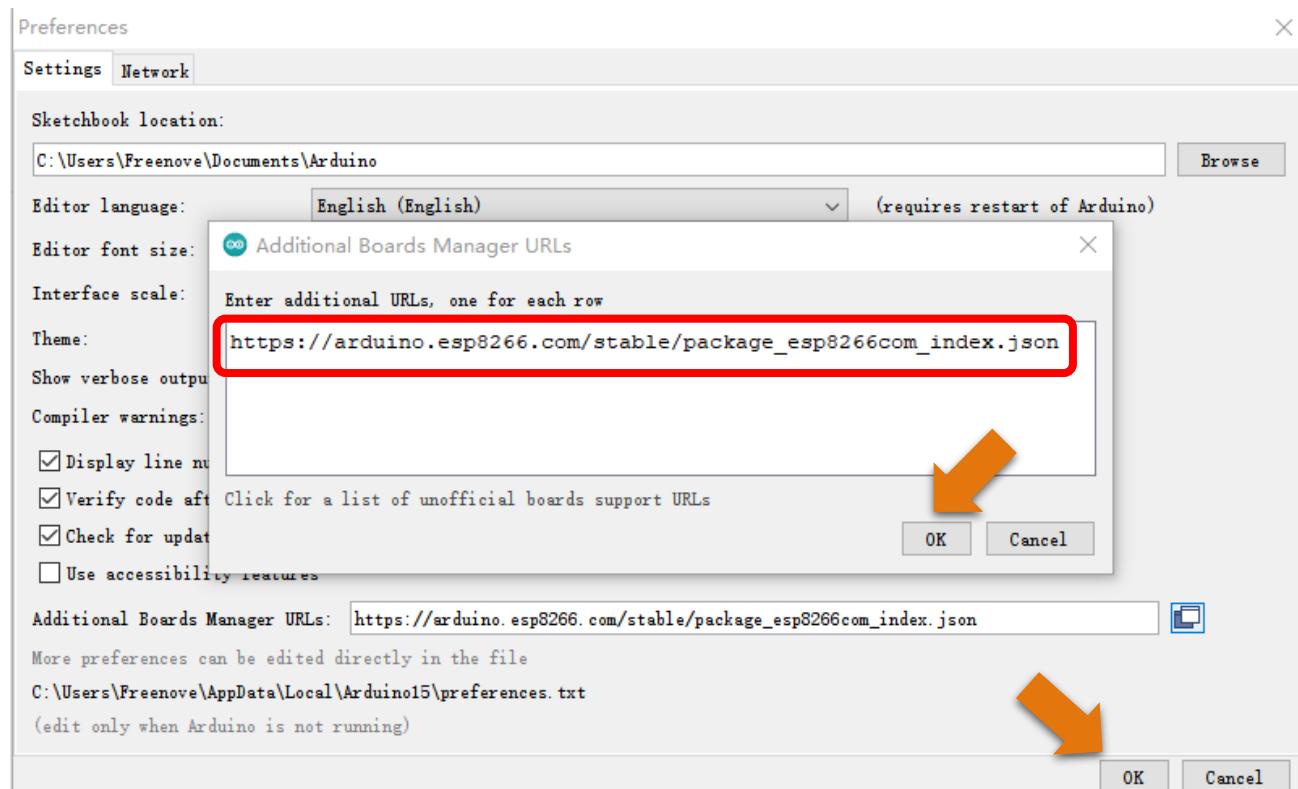
First, open the software platform arduino, and then click File in Menus and select Preferences.



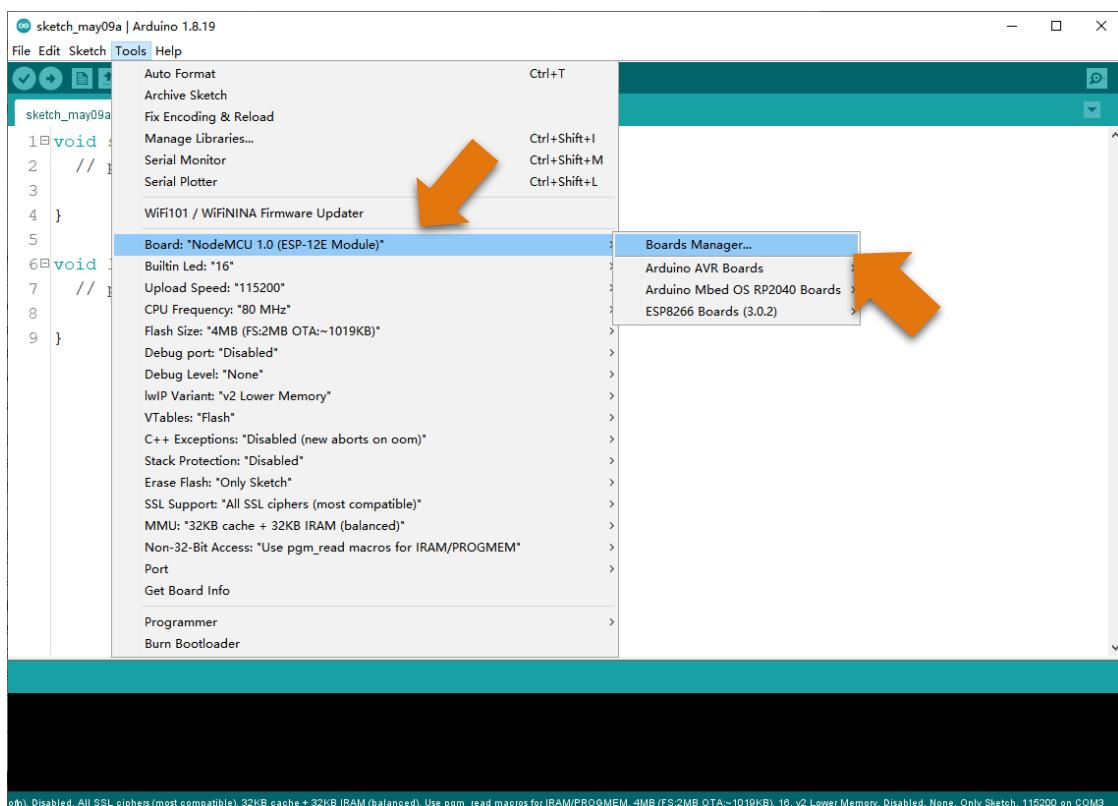
Second, click on the symbol behind "Additional Boards Manager URLs"



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.



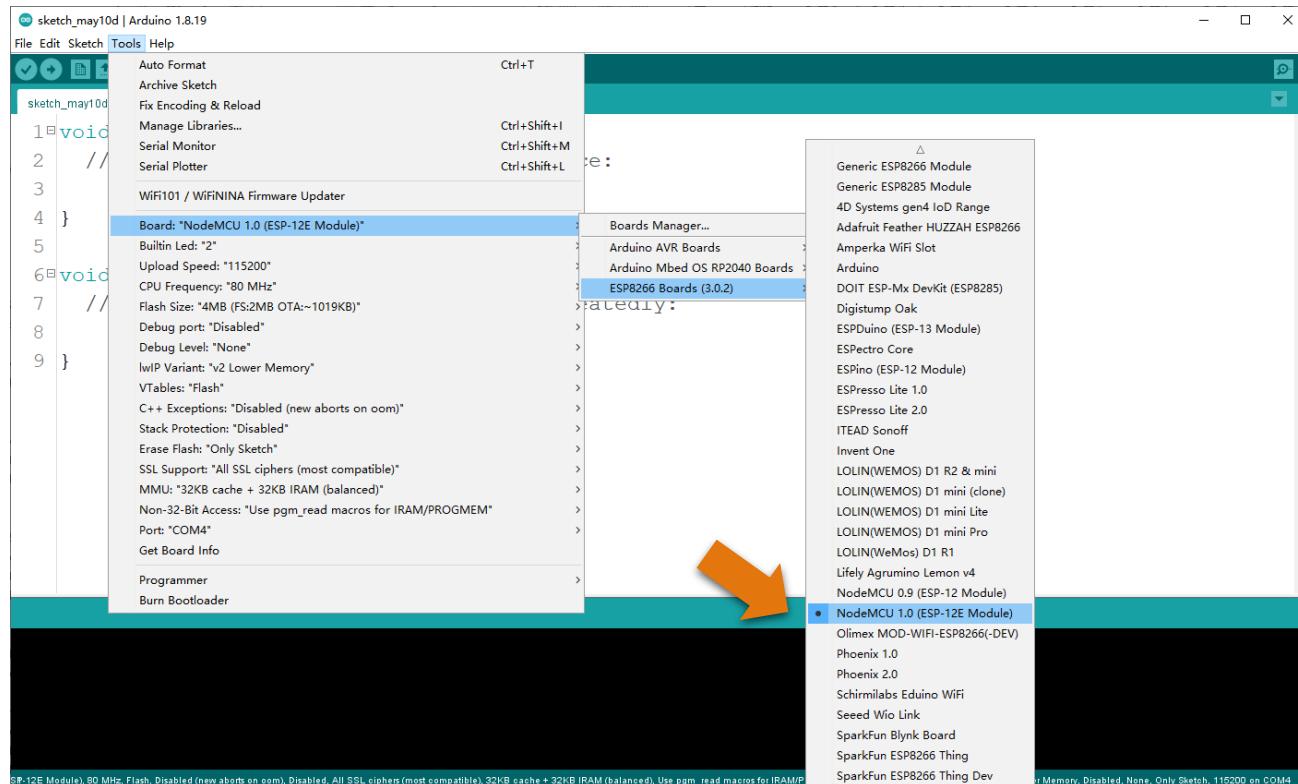
Fourth, click Tools in Menus, select Board:"ArduinoUno", and then select "Boards Manager".



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

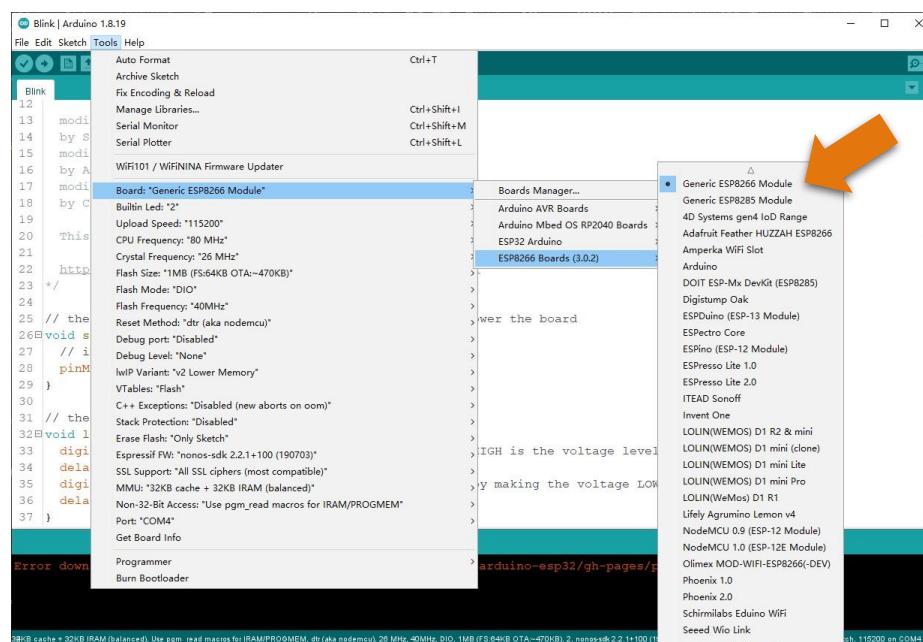


When finishing installation, click Tools in the Menus again and select Board: "NodeMCU 1.0(ESP-12E Module)", and then you can see information of ESP8266 click "NodeMCU 1.0(ESP-12E Module)" so that the ESP8266 programming development environment is configured.



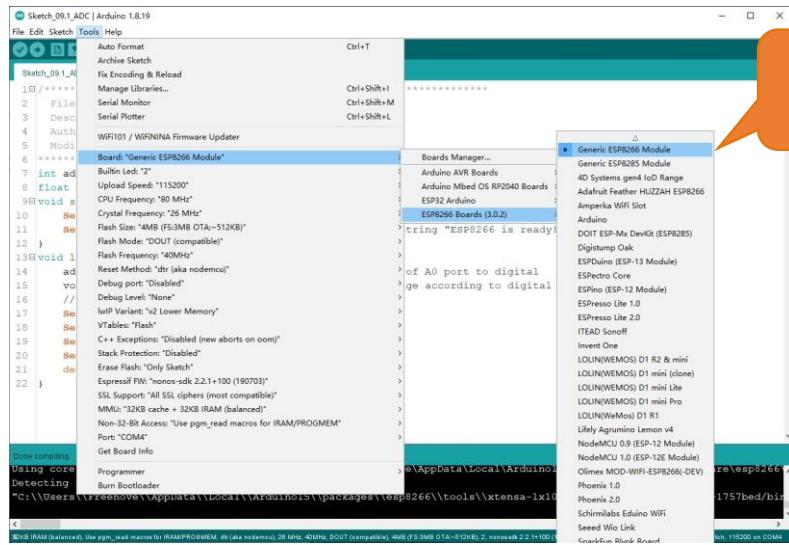
In our tutorial, we chose "NodeMCU 1.0(ESP-12E Module)" as the development board Module. This choice will facilitate learning and understanding of ESP8266. Of course, you can choose "Generic ESP8266 Module". Select "Generic ESP8266 Module" to apply to all Generic ESP8266 modules. Of course, this setup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

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



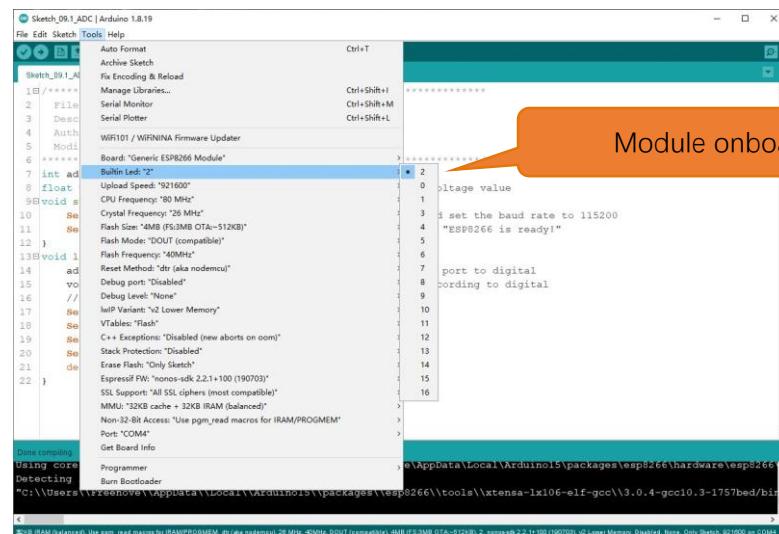
Any concerns? ✉ support@freenove.com

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



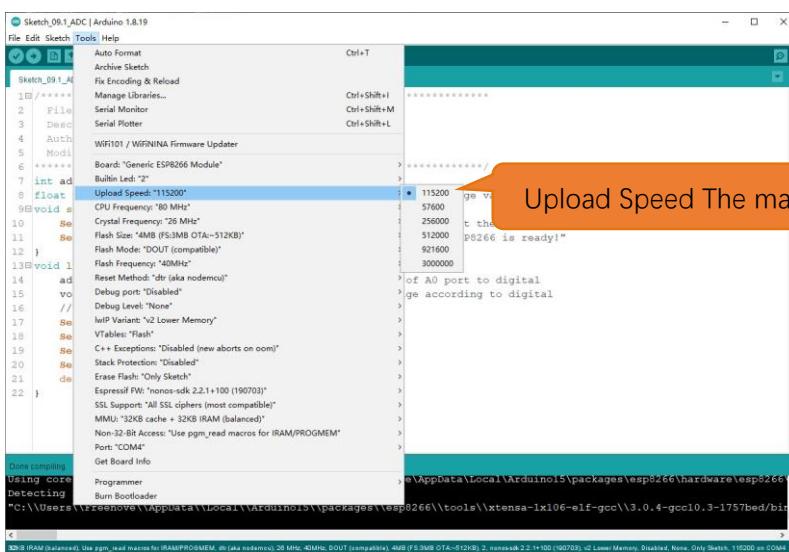
Select the module typehe module type

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



Module onboard LED

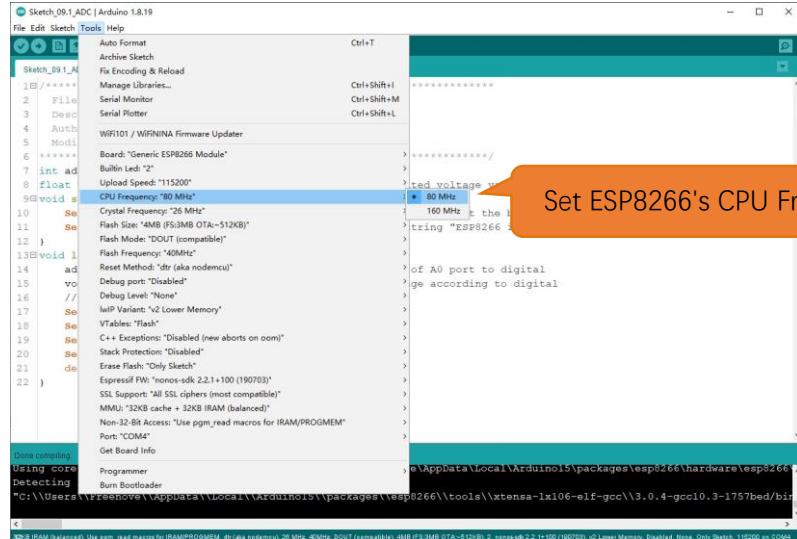
Upload Speed The maximum value is 921600. By default, Upload Speed is 115200. You can choose according to your needs.



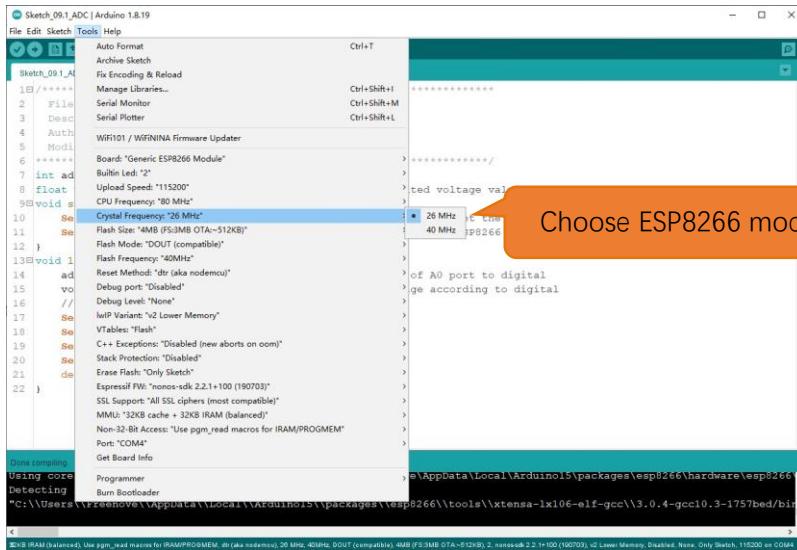
Upload Speed The maximum value is 921600

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

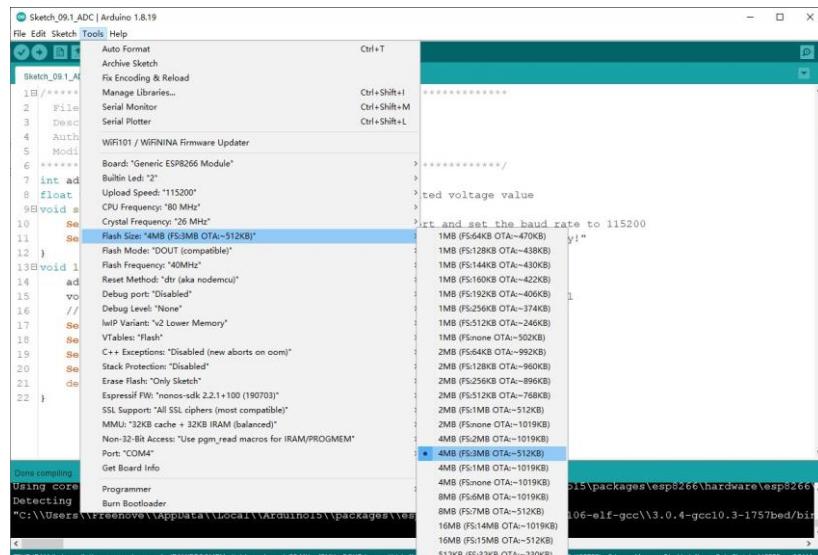
Any concerns? [✉ support@freenove.com](mailto:support@freenove.com)



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).



Any concerns? ✉ support@freenove.com



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.

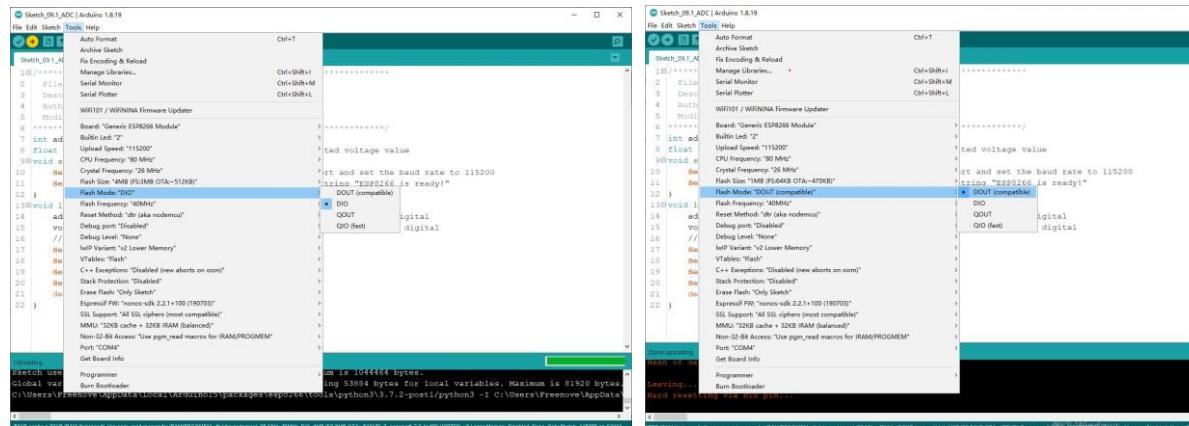
1.DOUT: Address is input in 1-line mode and data is output in 2-line mode.

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

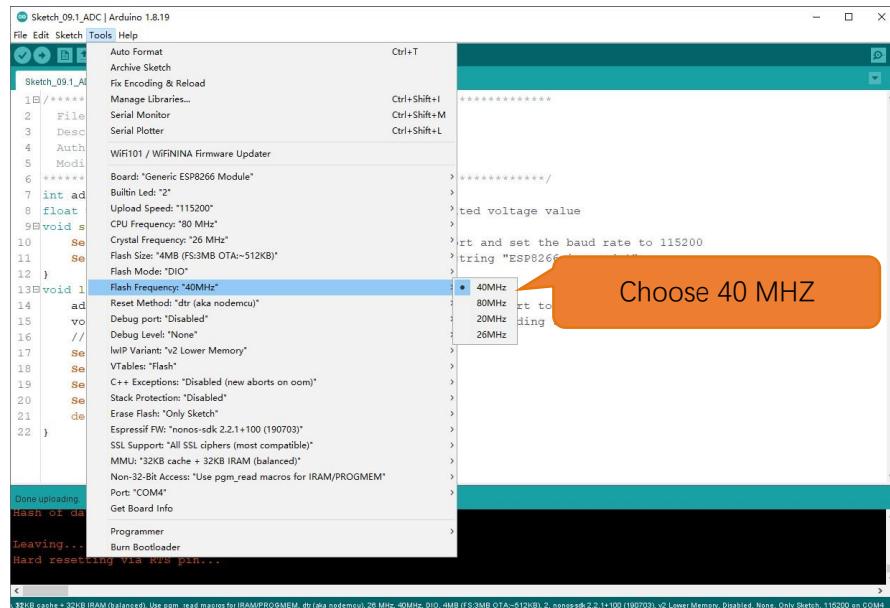
3.QOUT: Address is input in 1-line mode and data is output in 4-line mode.

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

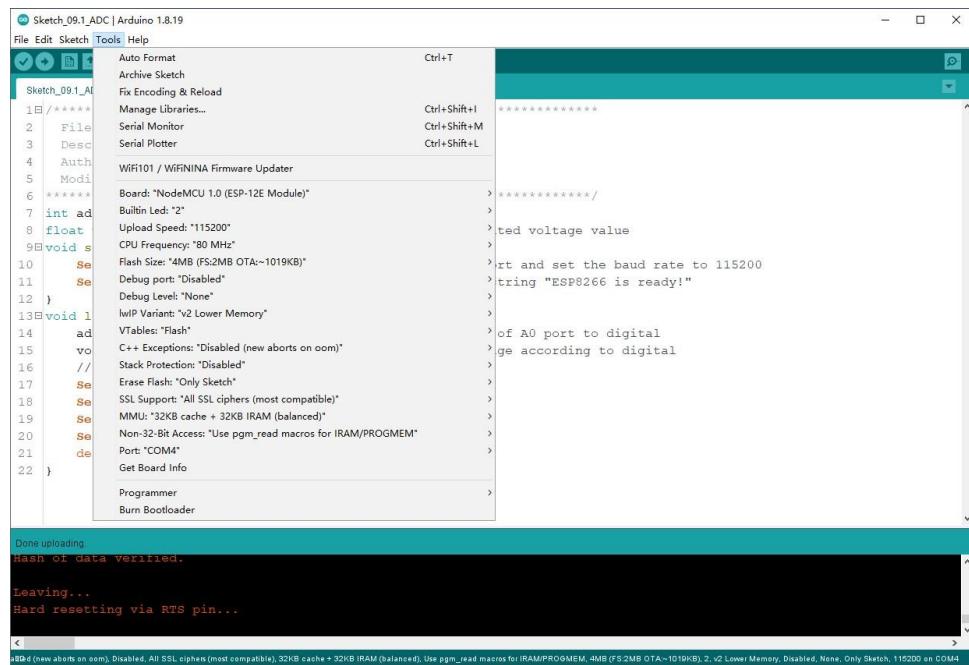
If you need to use the QIO mode, ensure that the Flash supports the QIO mode.



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

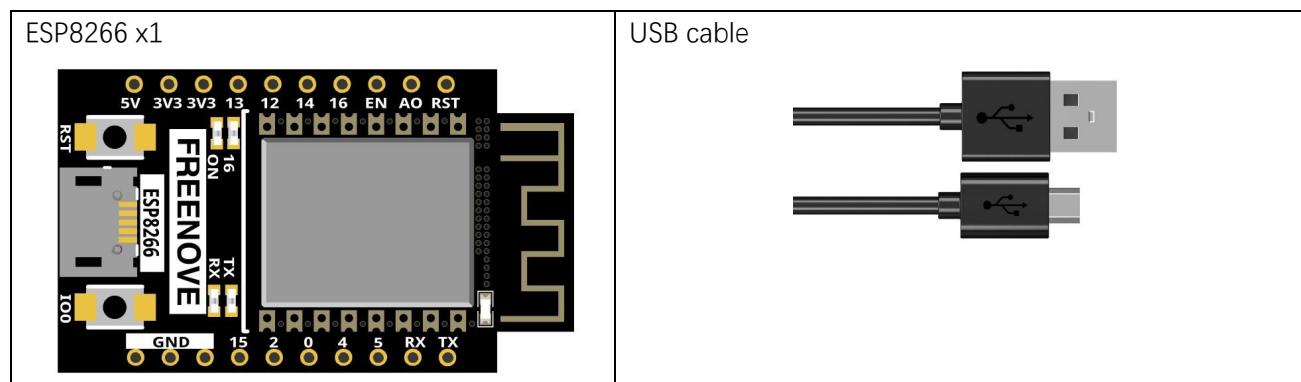
Chapter 0 LED

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

Project 0.1 Blink

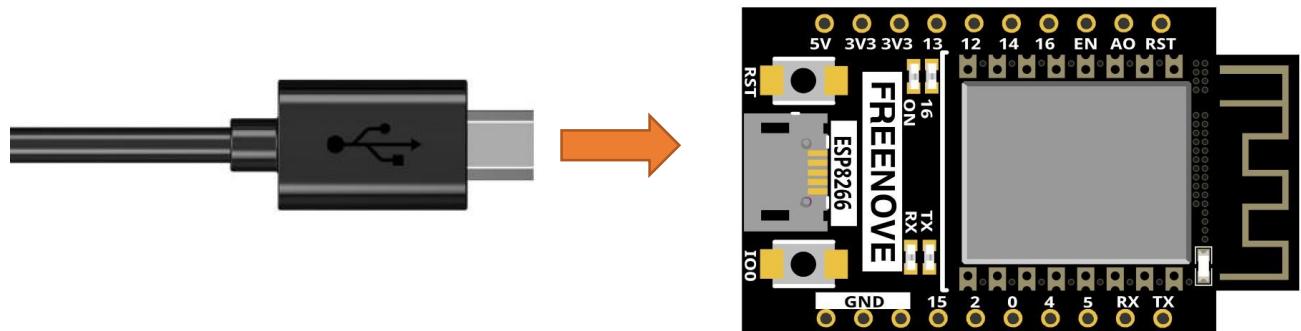
In this project, we will use ESP8266 to control blinking a common LED.

Component List



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.

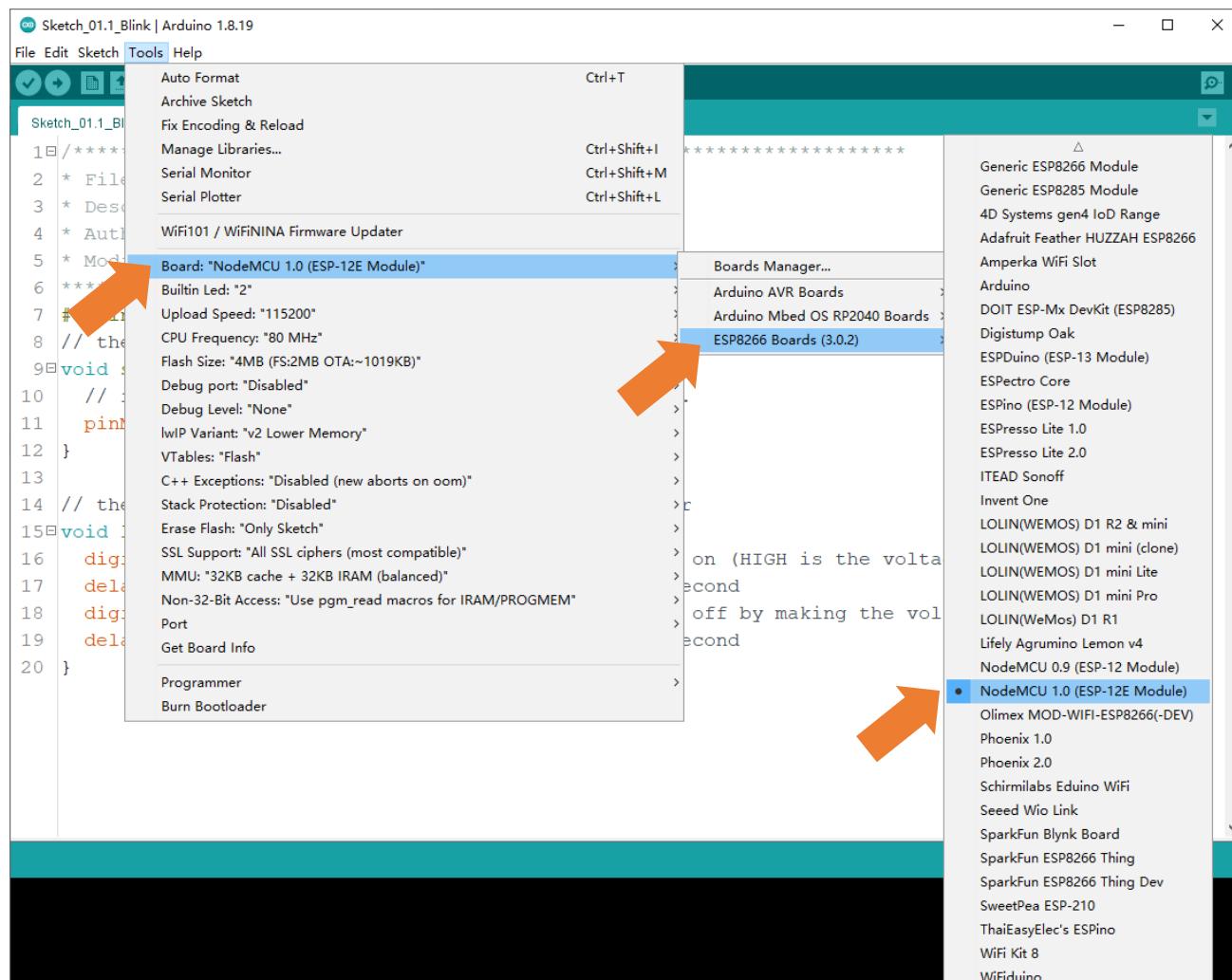
Sketch

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_Super_Starter_Kit_for_ESP8266\Sketches\Sketch_01.1_Blink

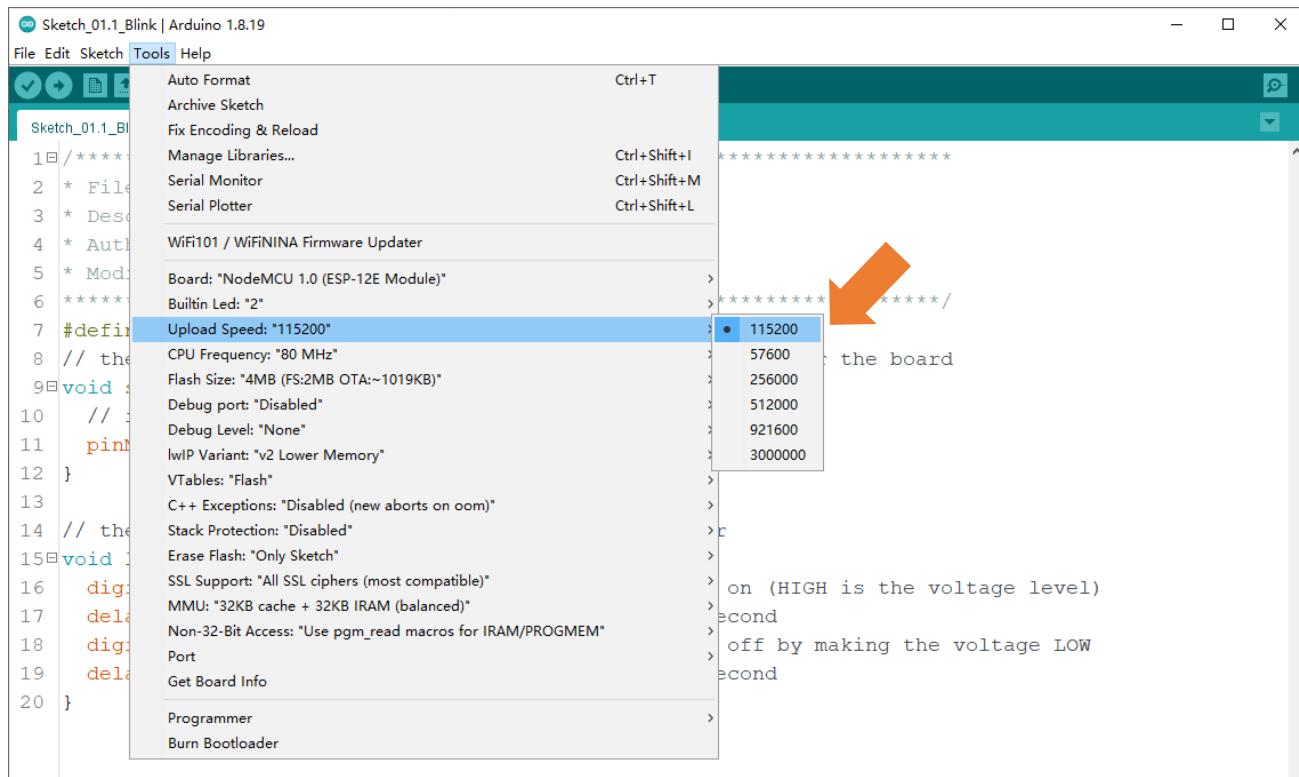
Before uploading the code, click "Tools", "Board" and select "NodeMCU 1.0 (ESP-12E Module)".



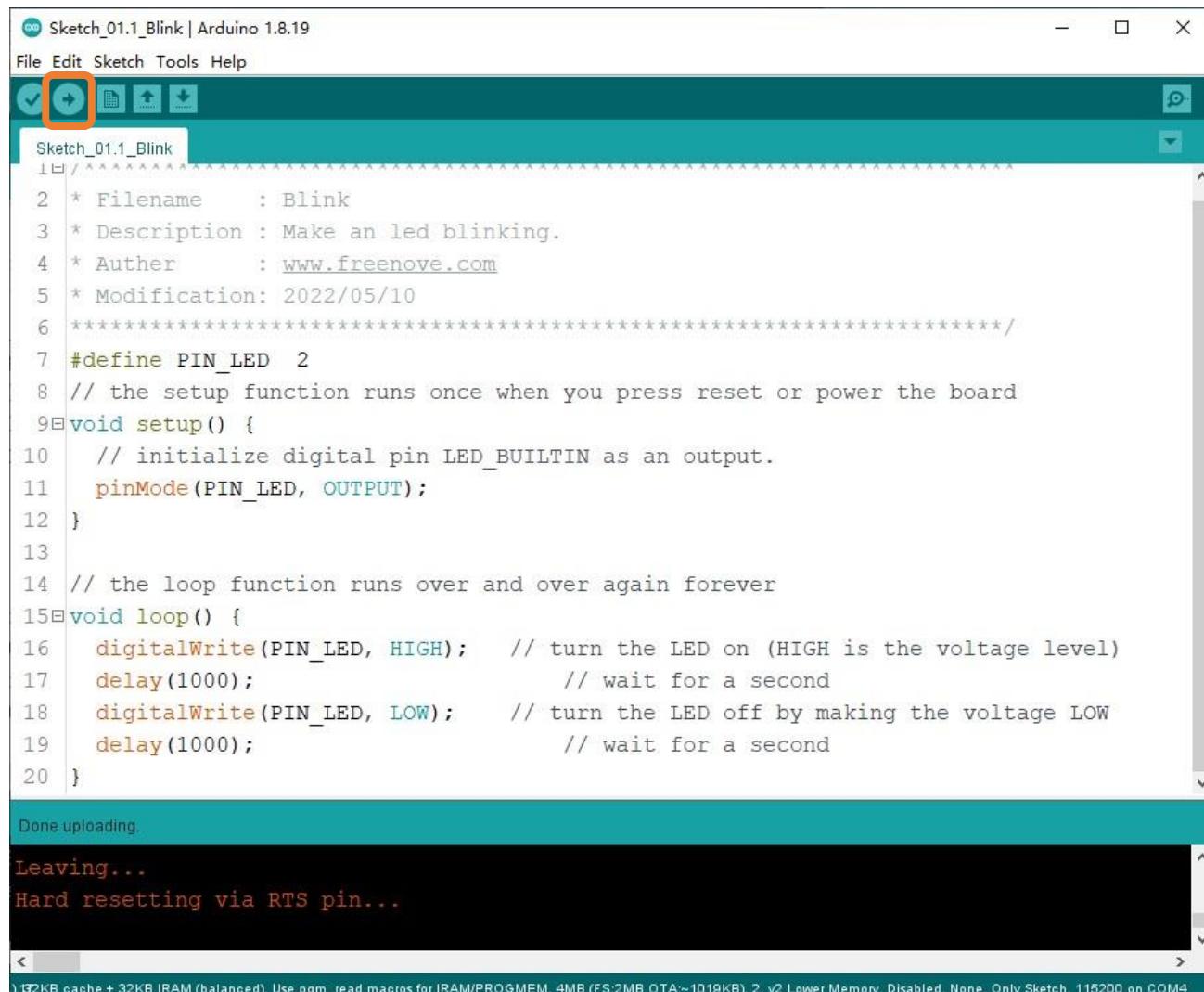
Select the serial port.



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



Sketch_01.1_Blink



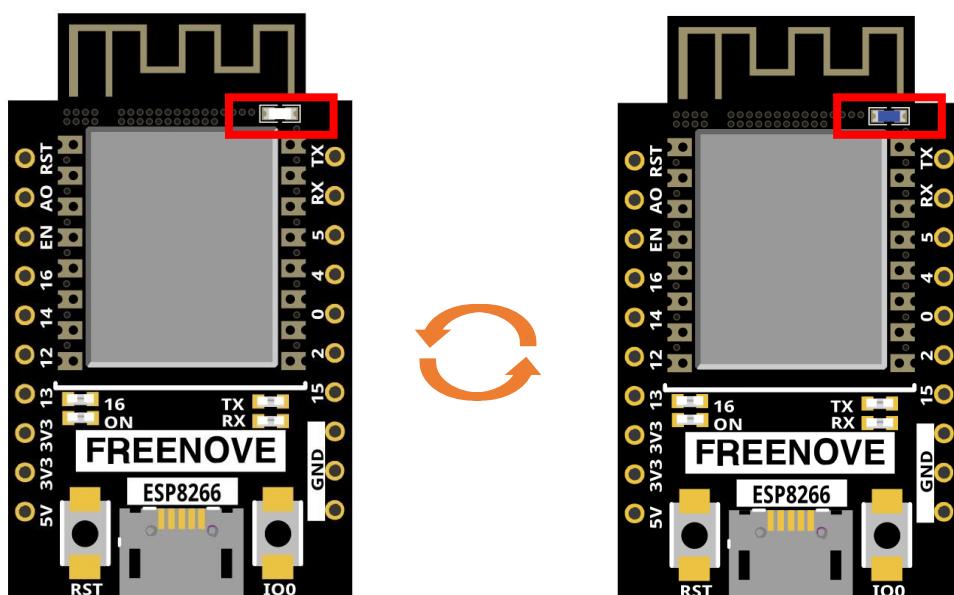
```

Sketch_01.1_Blink | Arduino 1.8.19
File Edit Sketch Tools Help
Sketch_01.1_Blink
1 // ****
2 * Filename      : Blink
3 * Description   : Make an led blinking.
4 * Author        : www.freenove.com
5 * Modification: 2022/05/10
6 ****
7 #define PIN_LED  2
8 // the setup function runs once when you press reset or power the board
9 void setup() {
10    // initialize digital pin LED_BUILTIN as an output.
11    pinMode(PIN_LED, OUTPUT);
12 }
13
14 // the loop function runs over and over again forever
15 void loop() {
16    digitalWrite(PIN_LED, HIGH);    // turn the LED on (HIGH is the voltage level)
17    delay(1000);                  // wait for a second
18    digitalWrite(PIN_LED, LOW);    // turn the LED off by making the voltage LOW
19    delay(1000);                  // wait for a second
20 }

```

Done uploading.
Leaving...
Hard resetting via RTS pin...
,132KB cache + 32KB IRAM (balanced). Use pgm_read macros for IRAM/PROGMEM, 4MB (FS:2MB OTA:~1019KB), 2, v2 Lower Memory, Disabled, None, Only Sketch, 115200 on COM4

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:

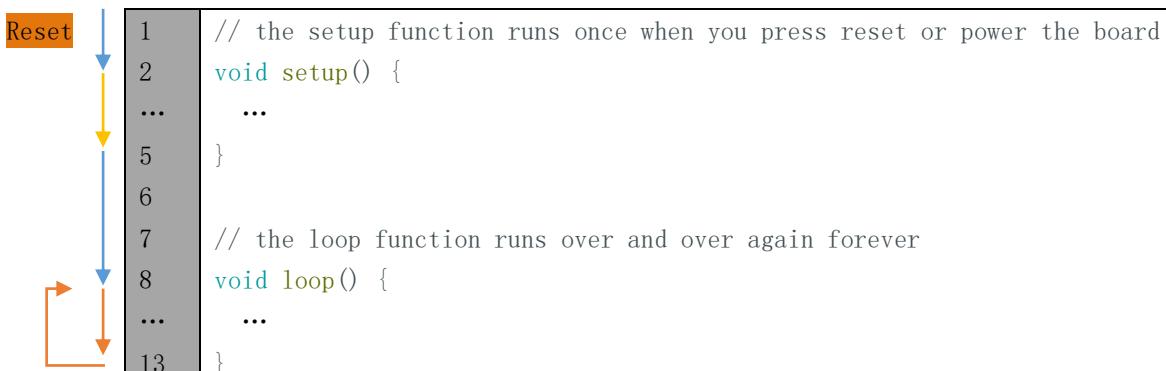
```

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

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

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

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



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.

```
1 #define PIN_LED 2
```

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

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

```

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

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

```
10 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_PULLDOWN, or INPUT_PULLUP.

```
void digitalWrite (int pin, int value);
```

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

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

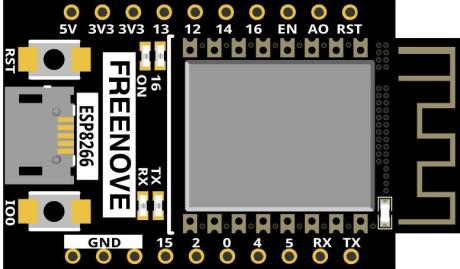
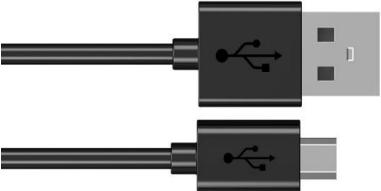
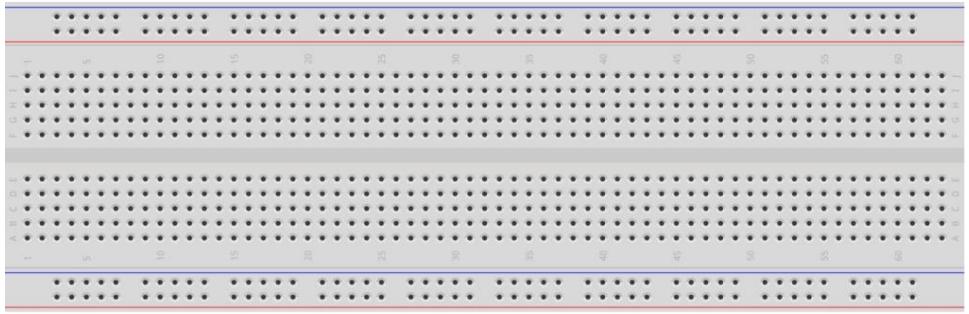
Chapter 1 LED

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

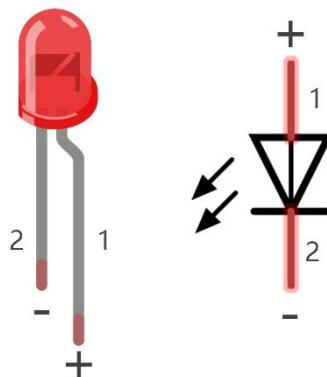
ESP8266 x1	USB cable	
		
Breadboard x1		
		
LED x1	Resistor 220Ω x1	Jumper wire M/M x3

Component knowledge

LED

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

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



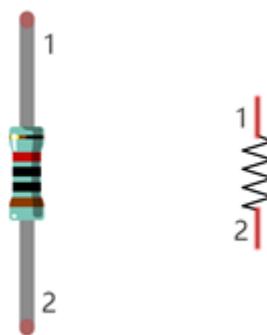
LED	Voltage	Maximum current	Recommended current
Red	1.9 - 2.2V	20mA	10mA
Green	2.9 - 3.4V	10mA	5mA
Blue	2.9 - 3.4V	10mA	5mA
Volt ampere characteristics conform to diode			

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

Resistor

Resistors use Ohms (Ω) as the unit of measurement of their resistance (R). $1M\Omega=1000k\Omega$, $1k\Omega=1000\Omega$.

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

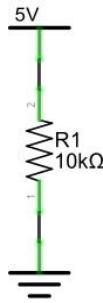


The bands of color on a resistor is a shorthand code used to identify its resistance value. For more details of resistor color codes, please refer to the appendix of this tutorial.

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

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

Any concerns? ✉ support@freenove.com

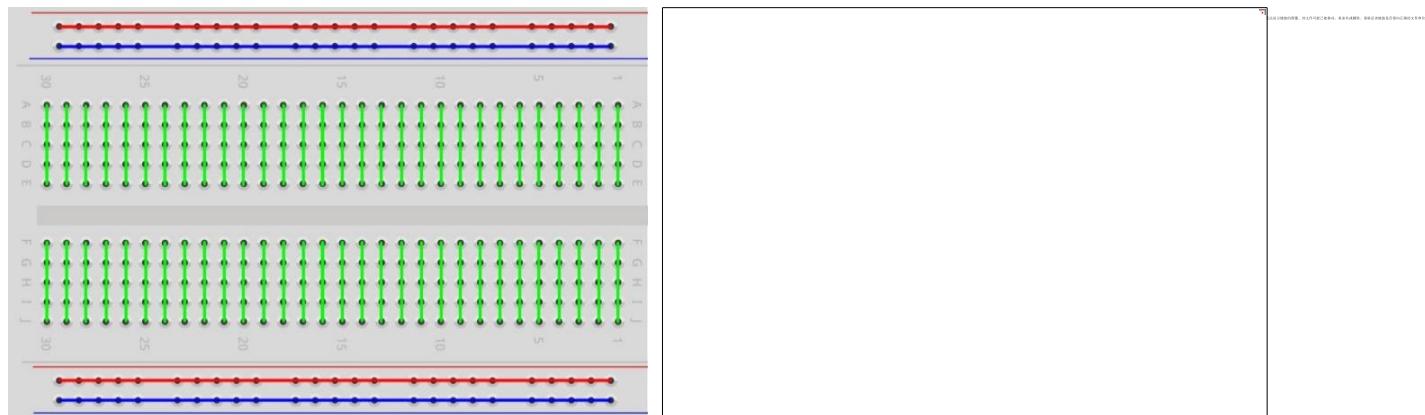


WARNING: Never connect the two poles of a power supply with anything of low resistance value (i.e. a metal object or bare wire) this is a Short and results in high current that may damage the power supply and electronic components.

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

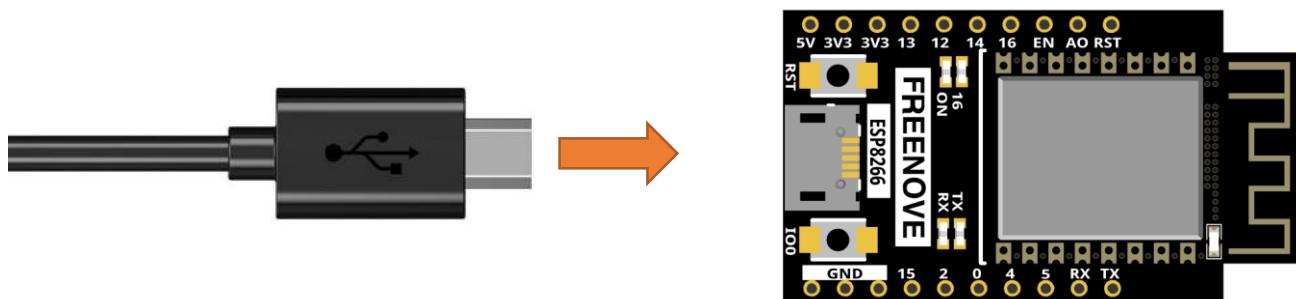
Breadboard

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



Power

ESP8266 needs 5v power supply. In this tutorial, we need connect ESP8266 to computer via USB cable to power it and program it. We can also use other 5v power source to power it.



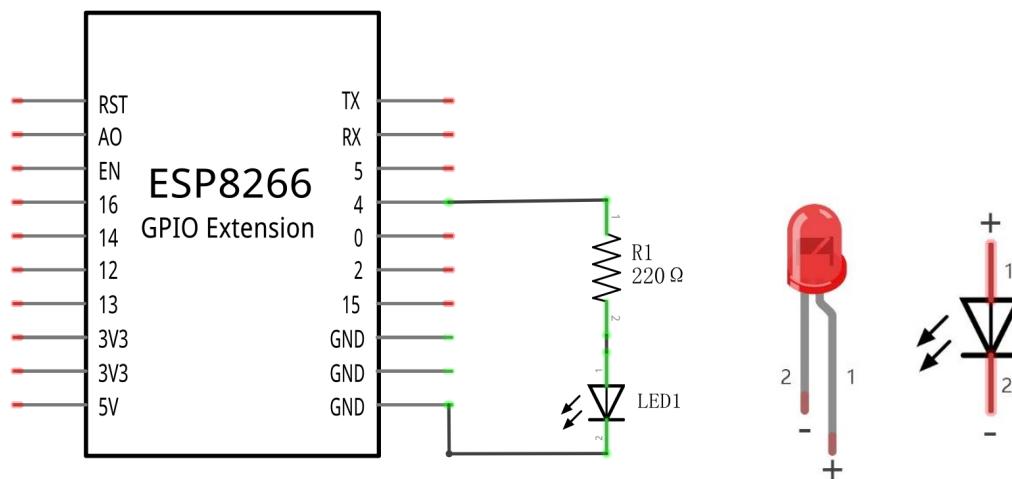
In the following projects, we only use USB cable to power ESP8266 by default.

Circuit

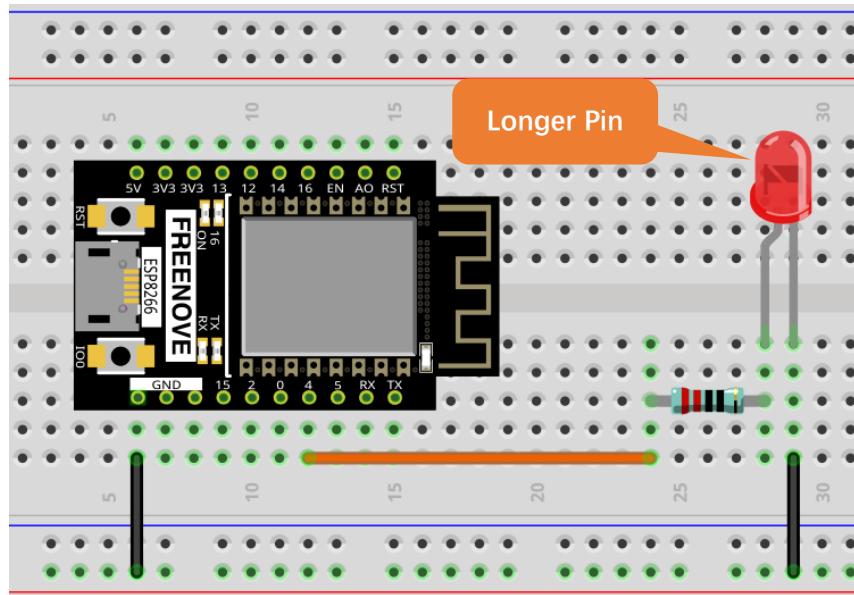
First, disconnect all power from the ESP8266. Then build the circuit according to the circuit and hardware diagrams. After the circuit is built and verified correct, connect the PC to ESP8266.

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

Schematic diagram



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



Sketch

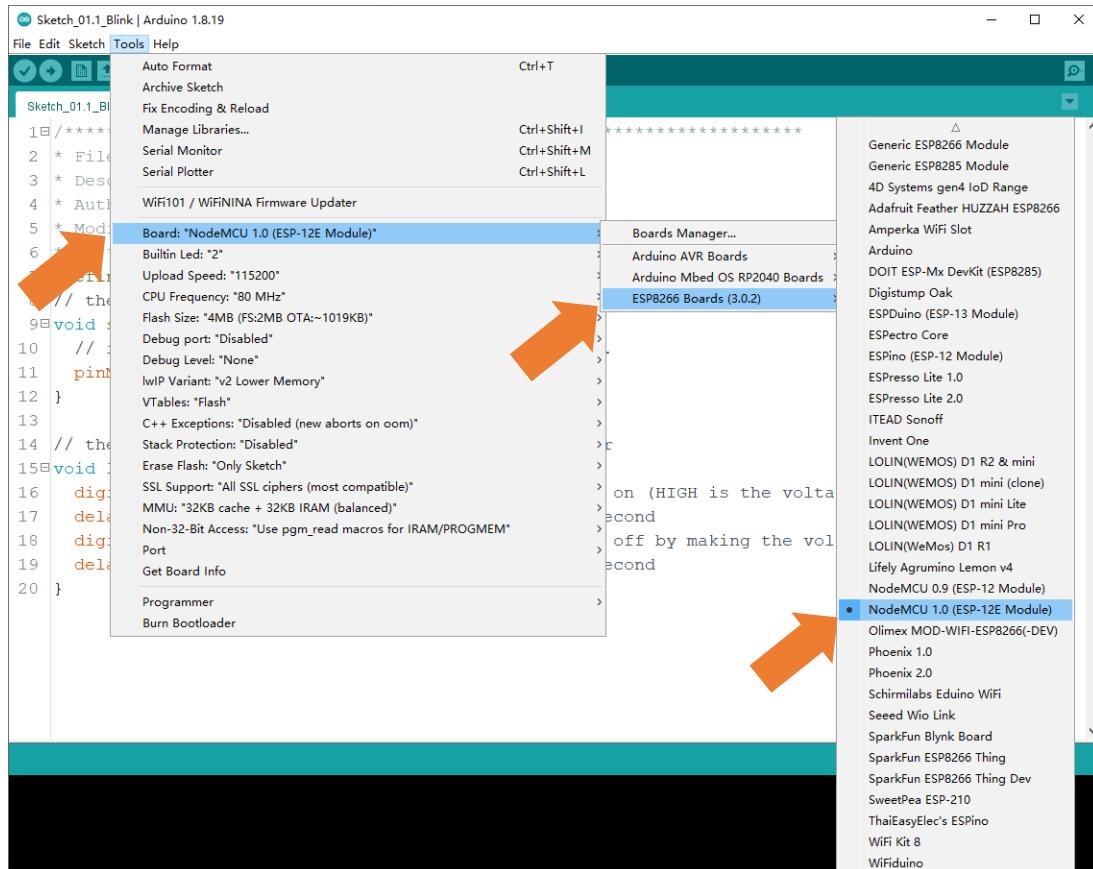
According to the circuit, when the GPIO4 of ESP8266 output level is high, the LED turns ON. Conversely, when the GPIO4 ESP8266 output level is low, the LED turns OFF. Therefore, we can let GPIO4 circularly output high and low level to make the LED blink.

Any concerns? ✉ support@freenove.com

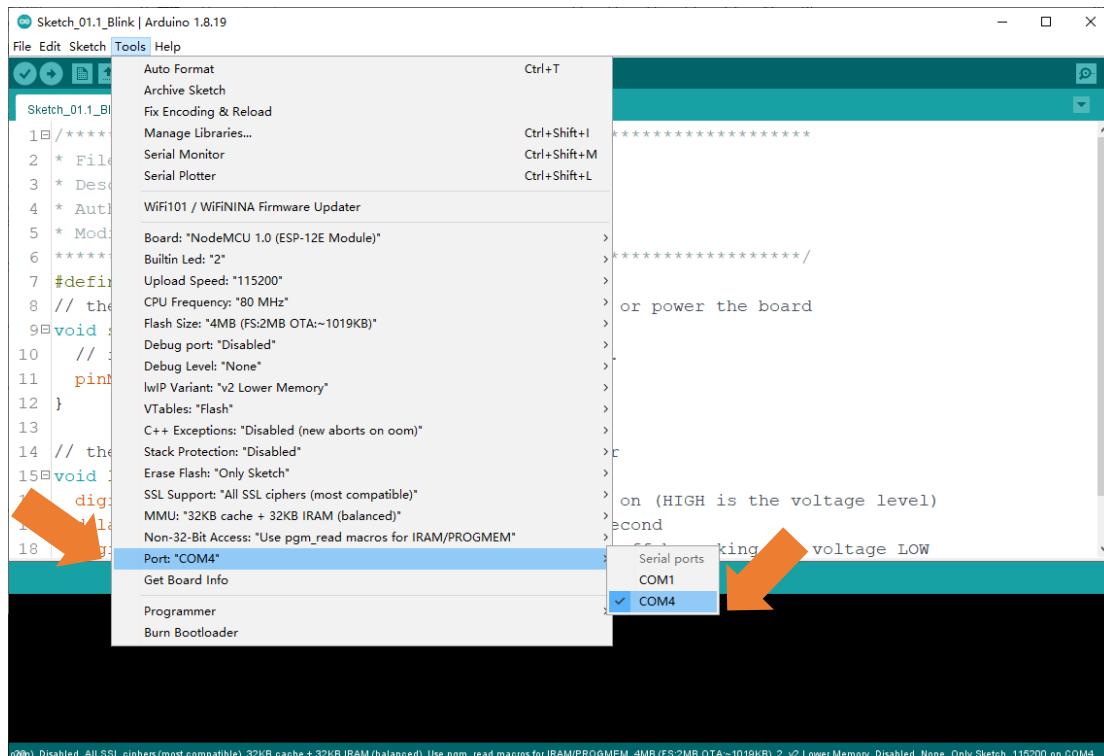
Upload the following Sketch:

Freenove_Super_Starter_Kit_for_ESP8266\Sketches\Sketch_01.1_Blink2

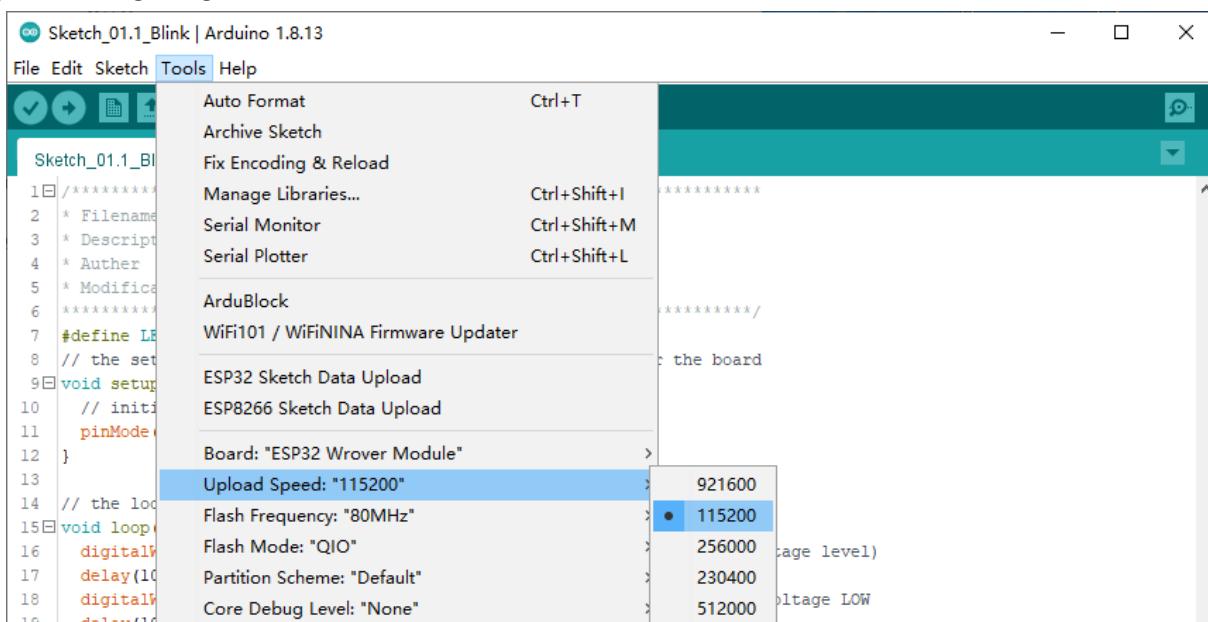
Before uploading the code, click "Tools", "Board" and select "NodeMCU 1.0 (ESP-12E Module)".



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_Blink2

The screenshot shows the Arduino IDE interface with the title bar "Sketch_01.1_Blink | Arduino 1.8.19". The "File" menu is open, and the "Upload" button (represented by a blue arrow icon) is circled in orange. The main code editor window displays a sketch named "Sketch_01.1_Blink2" with the same code as Sketch_01.1_Blink. The bottom status bar shows "Done uploading." and the serial monitor output shows:

```

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

```

The status bar also indicates memory usage: "282KB cache + 32KB IRAM (balanced). Use pgm_read macros for IRAM/PROGMEM. 4MB (FS:2MB OTA:~1019KB), 2, v2 Lower Memory, Disabled, None, Only Sketch, 115200 on COM4".

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



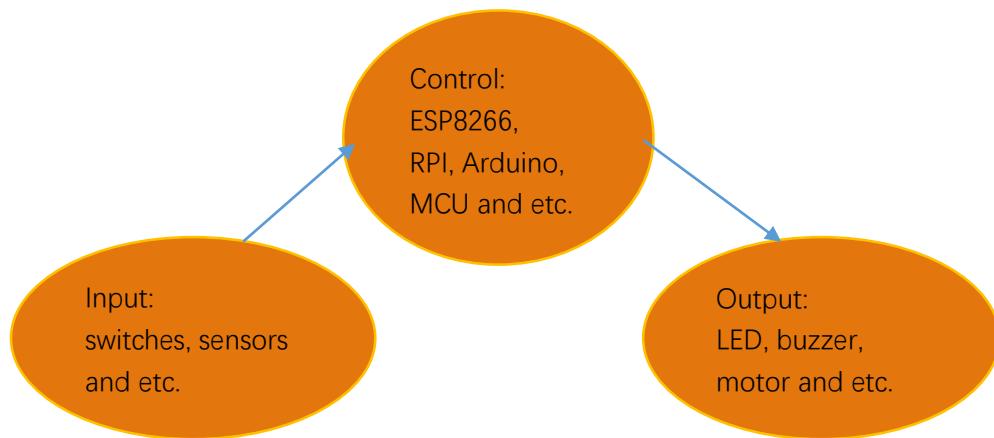
If you have any concerns, please contact us via: support@freenove.com

The following is the program code:

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

Chapter 2 Button & LED

Usually, there are three essential parts in a complete automatic control device: INPUT, OUTPUT, and CONTROL. In last section, the LED module was the output part and ESP8266 was the control part. In practical applications, we not only make LEDs flash, but also make a device sense the surrounding environment, receive instructions and then take the appropriate action such as LEDs light up, make a buzzer turn ON and so on.



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

Project 2.1 Button & LED

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



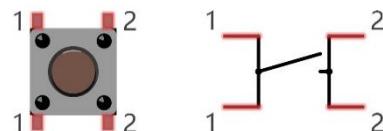
Component List

ESP8266 x1	USB cable			
Breadboard x1				
Jumper wire M/M x6	LED x1	Resistor 220Ω x1	Resistor 10kΩ x2	Push button x1

Component knowledge

Push button

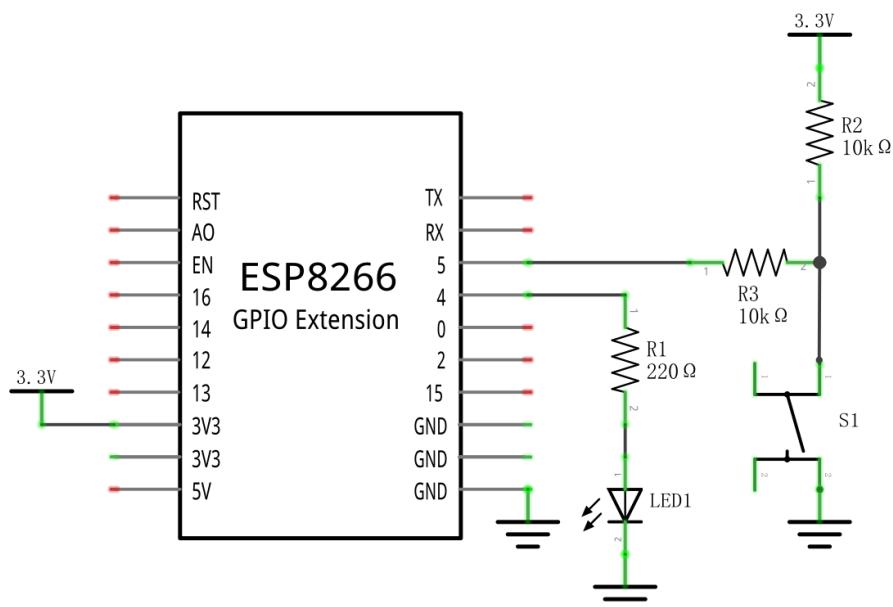
This type of push button switch has 4 pins (2 Pole Switch). Two pins on the left are connected, and both left and right sides are the same per the illustration:



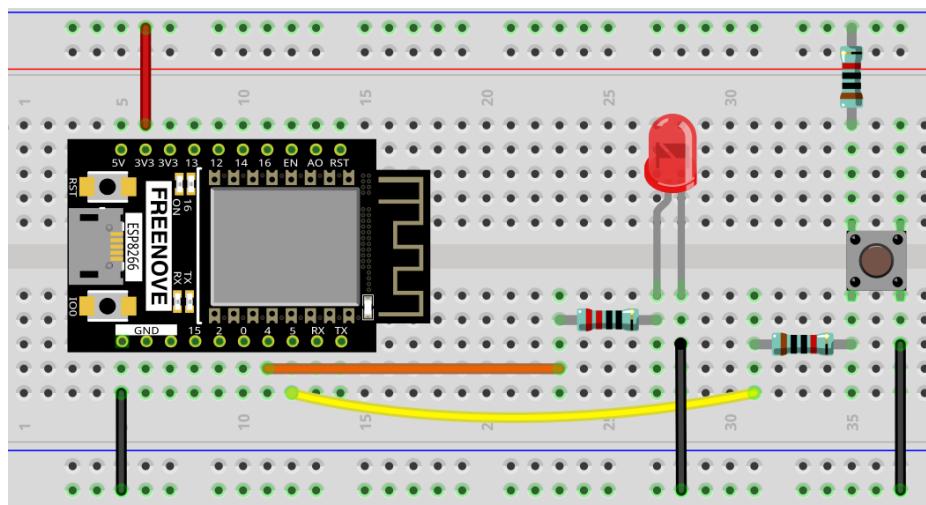
When the button on the switch is pressed, the circuit is completed (your project is powered ON).

Circuit

Schematic diagram



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



Any concerns? ✉ support@freenove.com



Sketch

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

Freenove_Super_Starter_Kit_for_ESP8266\Sketches\Sketch_02.1_ButtonAndLed

Sketch_02.1_ButtonAndLed

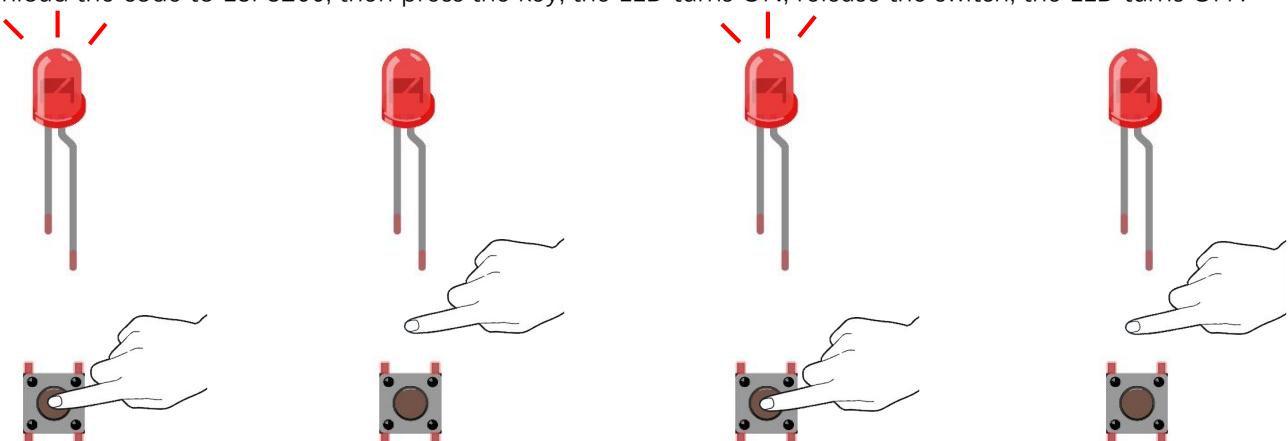
```

Sketch_02.1_ButtonAndLed | Arduino 1.8.19
File Edit Sketch Tools Help
Sketch_02.1_ButtonAndLed
1 // ****
2   Filename      : ButtonAndLed
3   Description   : Control led by button.
4   Author        : www.freenove.com
5   Modification: 2022/05/10
6 ****
7 #define PIN_LED    4
8 #define PIN_BUTTON 5
9 // the setup function runs once when you press reset or power the board
10 void setup() {
11   // initialize digital pin PIN_LED as an output.
12   pinMode(PIN_LED, OUTPUT);
13   pinMode(PIN_BUTTON, INPUT);
14 }
15
16 // the loop function runs over and over again forever
17 void loop() {
18   if (digitalRead(PIN_BUTTON) == LOW) {
19     digitalWrite(PIN_LED, HIGH);
20   }else{
21     digitalWrite(PIN_LED, LOW);
22   }
}
Done uploading.
Leaving...
Hard resetting via RTS pin...

```

s2h oom), Disabled, All SSL ciphers (most compatible), 32KB cache + 32kB IRAM (balanced), Use pgm_read macros for IRAM/PROGMEM, 4MB (FS:2MB OTA:~1019KB), 2, v2 Lower Memory, Disabled, None, Only Sketch, 115200 on COM4

Download the code to ESP8266, then press the key, the LED turns ON, release the switch, the LED turns OFF.



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

Any concerns? ✉ support@freenove.com

The following is the program code:

```

1 #define PIN_LED    4
2 #define PIN_BUTTON 5
3 // the setup function runs once when you press reset or power the board
4 void setup() {
5     // initialize digital pin PIN_LED as an output.
6     pinMode(PIN_LED, OUTPUT);
7     pinMode(PIN_BUTTON, INPUT);
8 }
9
10 // the loop function runs over and over again forever
11 void loop() {
12     if (digitalRead(PIN_BUTTON) == LOW) {
13         digitalWrite(PIN_LED, HIGH);
14     }else{
15         digitalWrite(PIN_LED, LOW);
16     }
17 }
```

In the circuit connection, LED and button are connected with GPIO4 and GPIO5 respectively, so define PIN_LED and PIN_BUTTON as 4 and 5 respectively.

```

1 #define PIN_LED    4
2 #define PIN_BUTTON 5
```

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

```

11 void loop() {
12     if (digitalRead(PIN_BUTTON) == LOW) {
13         digitalWrite(PIN_LED, HIGH);
14     }else{
15         digitalWrite(PIN_LED, LOW);
16     }
17 }
```

Reference

```
int digitalRead (int pin);
```

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



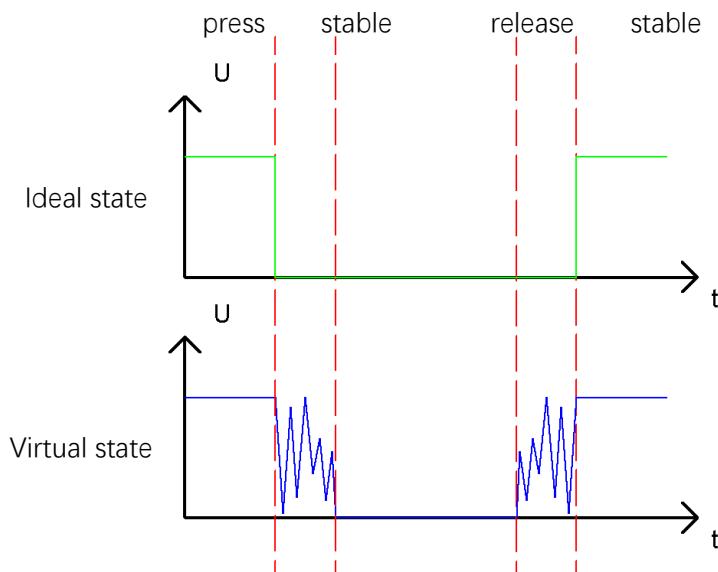
Project 2.2 MINI table lamp

We will also use a push button switch, LED and ESP8266 to make a MINI table lamp but this will function differently: Press the button, the LED will turn ON, and pressing the button again, the LED turns OFF. The ON switch action is no longer momentary (like a door bell) but remains ON without needing to continually press on the Button Switch.

First, let us learn something about the push button switch.

Debounce for Push Button

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



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

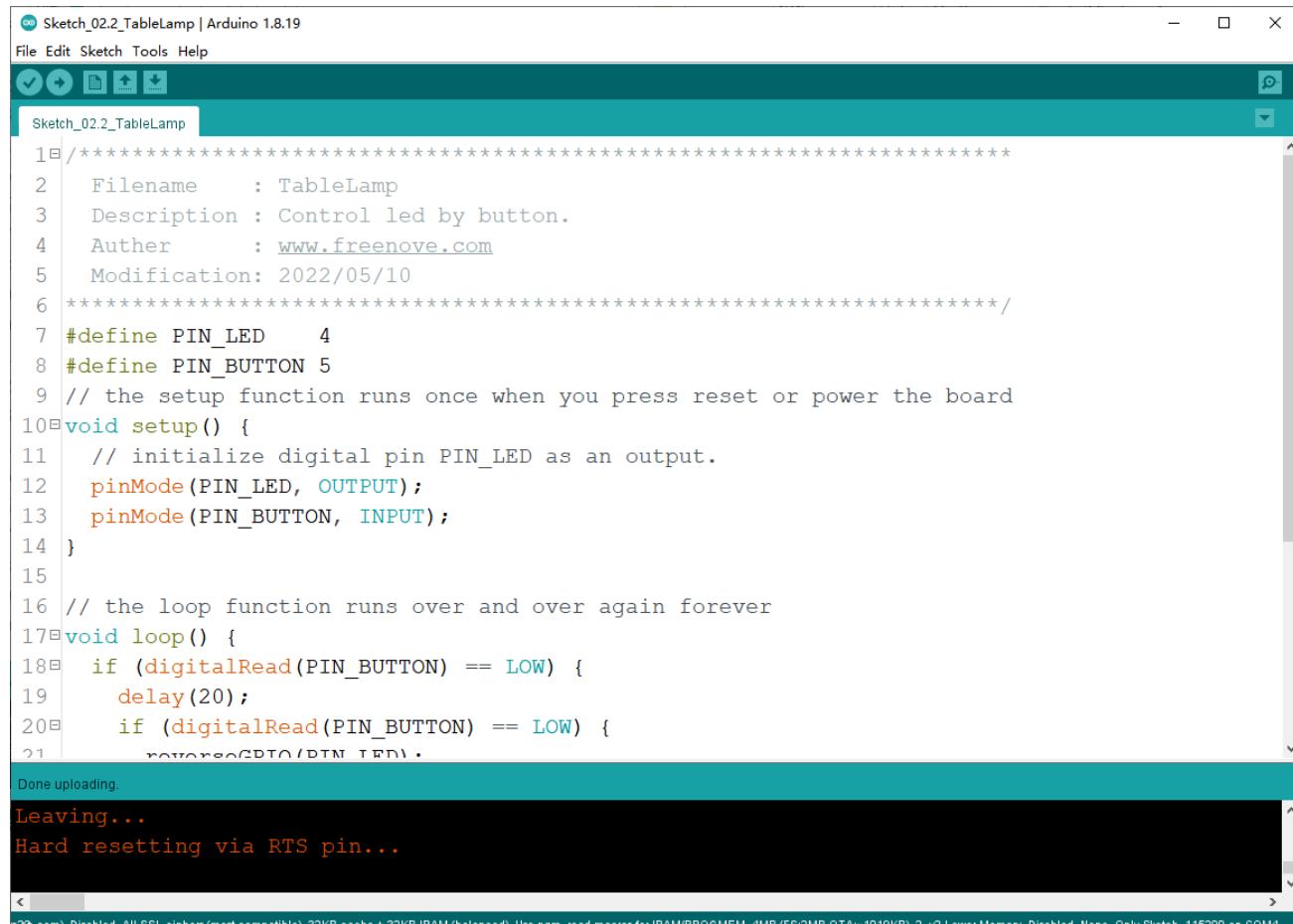
This project needs the same components and circuits as we used in the previous section.

Sketch

Upload following sketch:

Freenove_Super_Starter_Kit_for_ESP8266\Sketches\Sketch_02.2_Tablelamp

Sketch_02.2_Tablelamp



```
Sketch_02.2_TableLamp | Arduino 1.8.19
File Edit Sketch Tools Help
Sketch_02.2_TableLamp
1/* ****
2  Filename      : TableLamp
3  Description   : Control led by button.
4  Author        : www.freenove.com
5  Modification: 2022/05/10
6 ****
7 #define PIN_LED      4
8 #define PIN_BUTTON    5
9 // the setup function runs once when you press reset or power the board
10 void setup() {
11   // initialize digital pin PIN_LED as an output.
12   pinMode(PIN_LED, OUTPUT);
13   pinMode(PIN_BUTTON, INPUT);
14 }
15
16 // the loop function runs over and over again forever
17 void loop() {
18   if (digitalRead(PIN_BUTTON) == LOW) {
19     delay(20);
20     if (digitalRead(PIN_BUTTON) == LOW) {
21       digitalWrite(PIN_LED, !digitalRead(PIN_BUTTON));
}
Done uploading.
Leaving...
Hard resetting via RTS pin...
s2d.com), Disabled, All SSL ciphers (most compatible), 32KB cache + 32kB IRAM (balanced), Use pgm_read macros for IRAM/PROGMEM, 4MB (FS:2MB OTA:~1019KB), 2, v2 Lower Memory, Disabled, None, Only Sketch, 115200 on COM4
```

Download the code to the ESP8266, press the button, the LED turns ON, and press the button again, the LED turns OFF.

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

Any concerns? ✉ support@freenove.com



The following is the program code:

```

1 #define PIN_LED    4
2 #define PIN_BUTTON 5
3 // the setup function runs once when you press reset or power the board
4 void setup() {
5     // initialize digital pin PIN_LED as an output.
6     pinMode(PIN_LED, OUTPUT);
7     pinMode(PIN_BUTTON, INPUT);
8 }
9
10 // the loop function runs over and over again forever
11 void loop() {
12     if (digitalRead(PIN_BUTTON) == LOW) {
13         delay(20);
14         if (digitalRead(PIN_BUTTON) == LOW) {
15             reverseGPIO(PIN_LED);
16         }
17         while (digitalRead(PIN_BUTTON) == LOW);
18         delay(20);
19         while (digitalRead(PIN_BUTTON) == LOW);
20     }
21 }
22
23 void reverseGPIO(int pin) {
24     digitalWrite(pin, ! digitalRead(pin));
25 }
```

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

```

12 if (digitalRead(PIN_BUTTON) == LOW) {
13     delay(20);
14     if (digitalRead(PIN_BUTTON) == LOW) {
15         reverseGPIO(PIN_LED);
16     }
17     while (digitalRead(PIN_BUTTON) == LOW);
18     delay(20);
19     while (digitalRead(PIN_BUTTON) == LOW);
20 }
```

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

```

23 void reverseGPIO(int pin) {
24     digitalWrite(pin, ! digitalRead(pin));
25 }
```

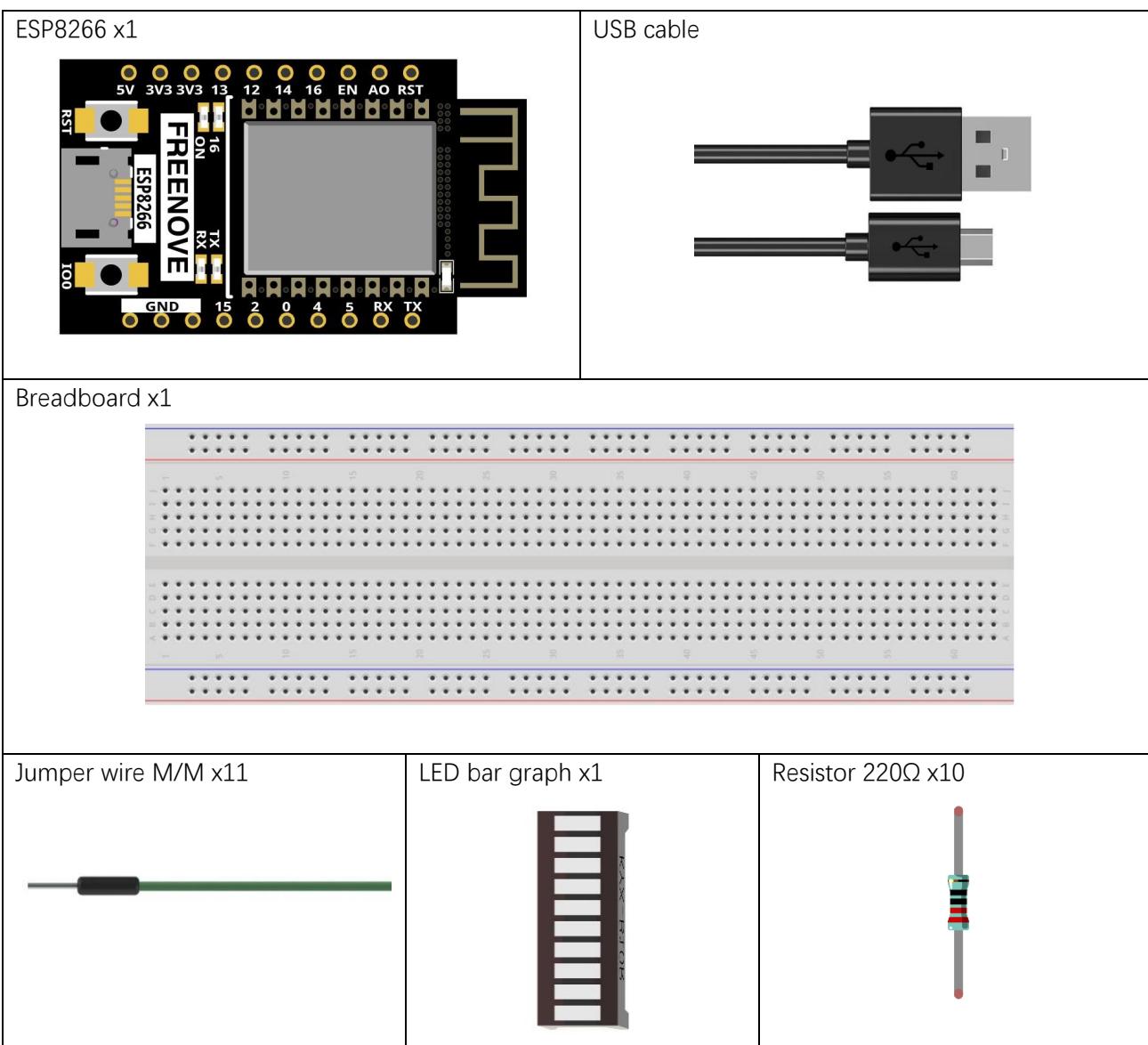
Chapter 3 LED Bar

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

Project 3.1 Flowing Light

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

Component List



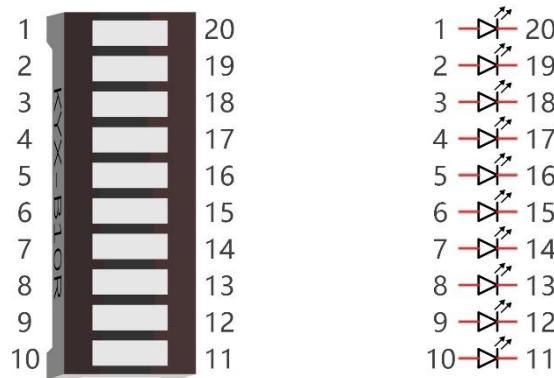


Component knowledge

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

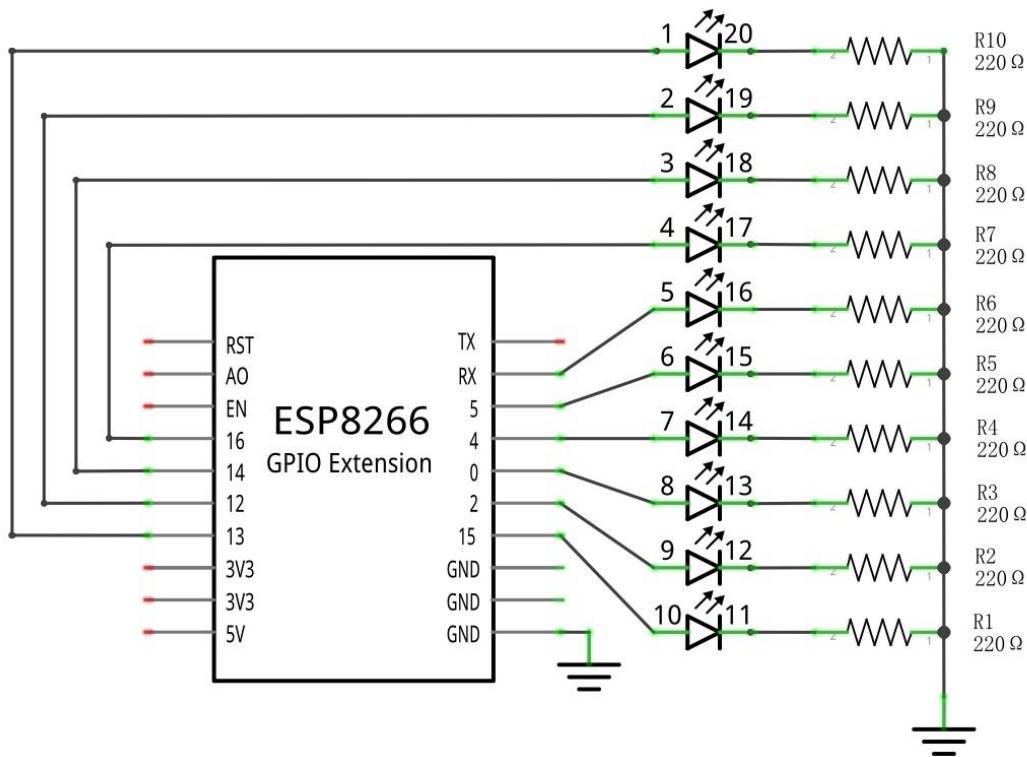
LED bar

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

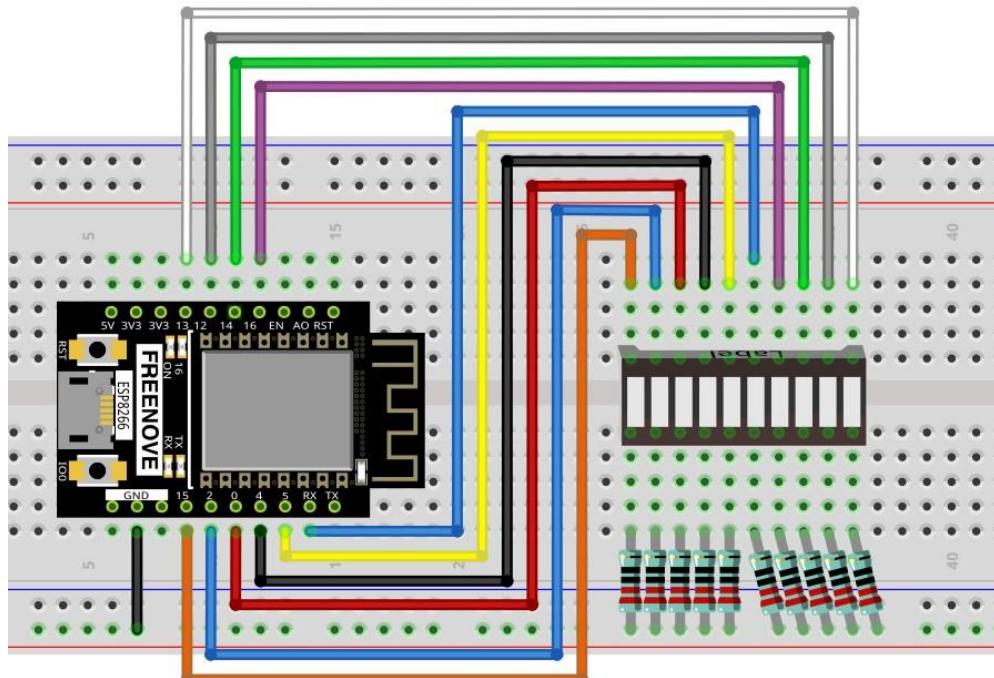


Circuit

Schematic diagram



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



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

Any concerns? ✉ support@freenove.com



Sketch

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

Upload following sketch:

Freenove_Super_Starter_Kit_for_ESP8266\Sketches\Sketch_03.1_FlowingLight

Sketch_03.1_FlowingLight

```

Sketch_03.1_FlowingLight | Arduino 1.8.19
File Edit Sketch Tools Help
Sketch_03.1_FlowingLight
1/* ****
2  Filename      : FlowingLight
3  Description   : Using ledbar to demonstrate flowing lamp.
4  Author        : www.freenove.com
5  Modification: 2022/05/10
6 ****
7 byte ledPins[] = {13, 12, 14, 16, 3, 5, 4, 0, 2, 15};
8 int ledCounts;
9
10 void setup() {
11   ledCounts = sizeof(ledPins);
12   for (int i = 0; i < ledCounts; i++) {
13     pinMode(ledPins[i], OUTPUT);
14   }
15 }
16
17 void loop() {
18   for (int i = 0; i < ledCounts; i++) {
19     digitalWrite(ledPins[i], HIGH);
20     delay(100);
21     digitalWrite(ledPins[i], LOW);
}
Done uploading.
Leaving...
Hard resetting via RTS pin...

```

Download the code to ESP8266 and LED bar graph will light up from left to right and from right to left.



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

Any concerns? ✉ support@freenove.com

The following is the program code:

```

1 byte ledPins[] = {13, 12, 14, 16, 3, 5, 4, 0, 2, 15};
2 int ledCounts;
3
4 void setup() {
5     ledCounts = sizeof(ledPins);
6     for (int i = 0; i < ledCounts; i++) {
7         pinMode(ledPins[i], OUTPUT);
8     }
9 }
10
11 void loop() {
12     for (int i = 0; i < ledCounts; i++) {
13         digitalWrite(ledPins[i], HIGH);
14         delay(100);
15         digitalWrite(ledPins[i], LOW);
16     }
17     for (int i = ledCounts - 1; i > -1; i--) {
18         digitalWrite(ledPins[i], HIGH);
19         delay(100);
20         digitalWrite(ledPins[i], LOW);
21     }
22 }
```

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

```
1 byte ledPins[] = {13, 12, 14, 16, 3, 5, 4, 0, 2, 15};
```

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

```

5 ledCounts = sizeof(ledPins);
6 for (int i = 0; i < ledCounts; i++) {
7     pinMode(ledPins[i], OUTPUT);
8 }
```

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

```

12 for (int i = 0; i < ledCounts; i++) {
13     digitalWrite(ledPins[i], HIGH);
14     delay(100);
15     digitalWrite(ledPins[i], LOW);
16 }
17 for (int i = ledCounts - 1; i > -1; i--) {
18     digitalWrite(ledPins[i], HIGH);
19     delay(100);
20     digitalWrite(ledPins[i], LOW);
21 }
```



Chapter 4 Analog & PWM

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

First, let's learn how to control the brightness of a LED.

Project 4.1 Breathing LED

Breathing light, that is, LED is turned from off to on gradually, and gradually from on to off, just like "breathing". So, how to control the brightness of a LED? We will use PWM to achieve this target.

Component List

ESP8266 x1	USB cable	
Breadboard x1		
LED x1	Resistor 220Ω x1	Jumper wire M/M x3

Related knowledge

Analog & Digital

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



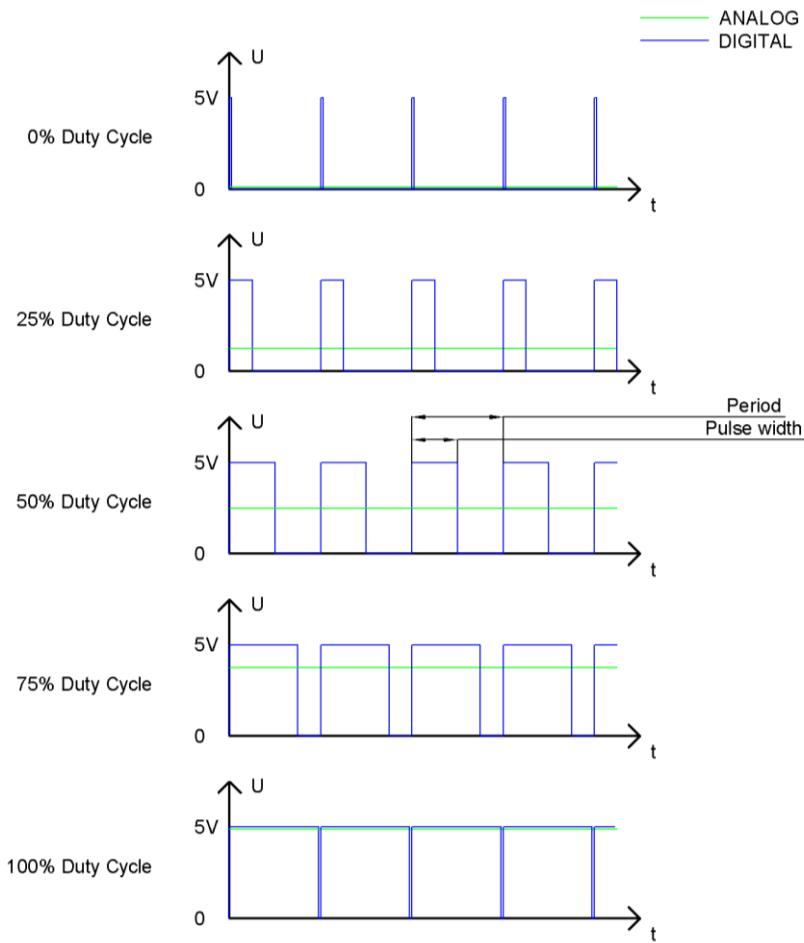
In practical application, we often use binary as the digital signal, that is a series of 0's and 1's. Since a binary signal only has two values (0 or 1), it has great stability and reliability. Lastly, both analog and digital signals can be converted into the other.

PWM

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

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

The longer the outputs of high levels last, the longer the duty cycle and the higher the corresponding voltage in the analog signal will be. The following figures show how the analog signal voltages vary between 0V-5V (high level is 5V) corresponding to the pulse width 0%-100%:

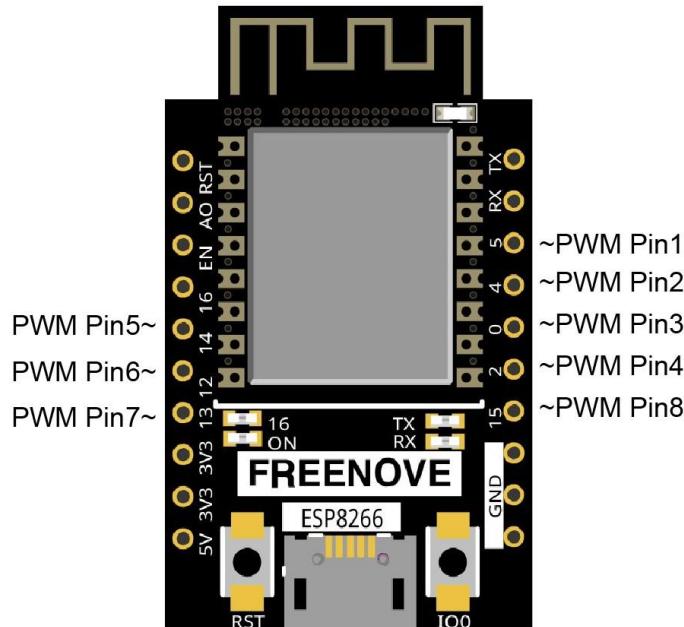


The longer the PWM duty cycle is, the higher the output power will be. Now that we understand this relationship, we can use PWM to control the brightness of a LED or the speed of DC motor and so on. It is evident from the above that PWM is not real analog, and the effective value of the voltage is equivalent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

ESP8266 and PWM

The ESP8266 PWM controller has 8 independent channels, each of which can independently control frequency, duty cycle, and even accuracy. Unlike traditional PWM pins, the PWM output pins of ESP8266 are configurable and they can be configured to PWM.

The ESP8266 supports PWM pins as follows:

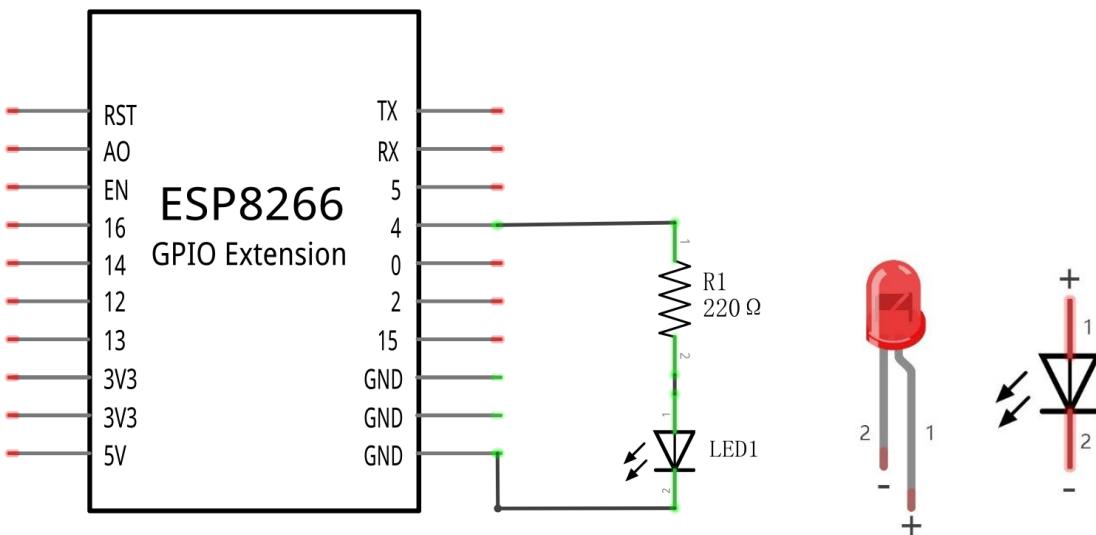


The longer the PWM duty cycle is, the higher the output power will be. Now that we understand this relationship, we can use PWM to control the brightness of an LED or the speed of DC motor and so on.

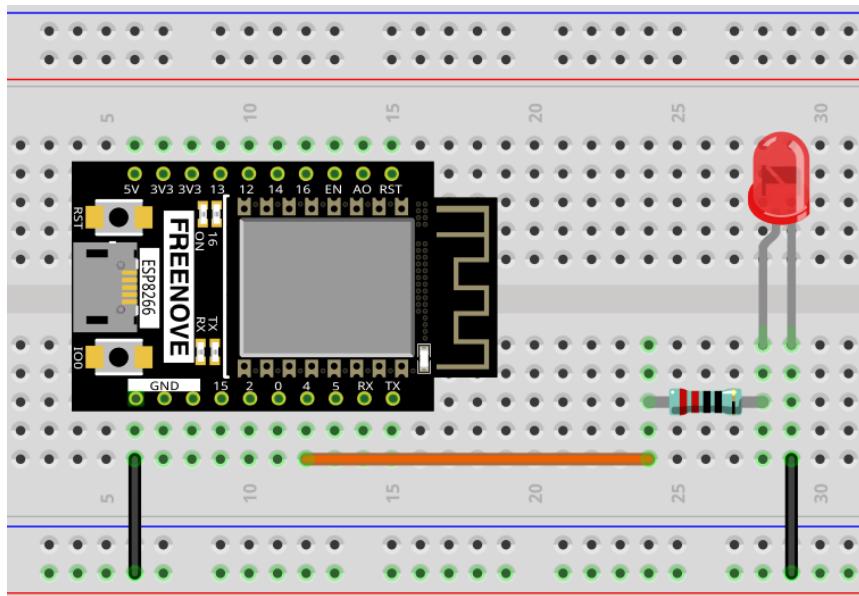
Circuit

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

Schematic diagram



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



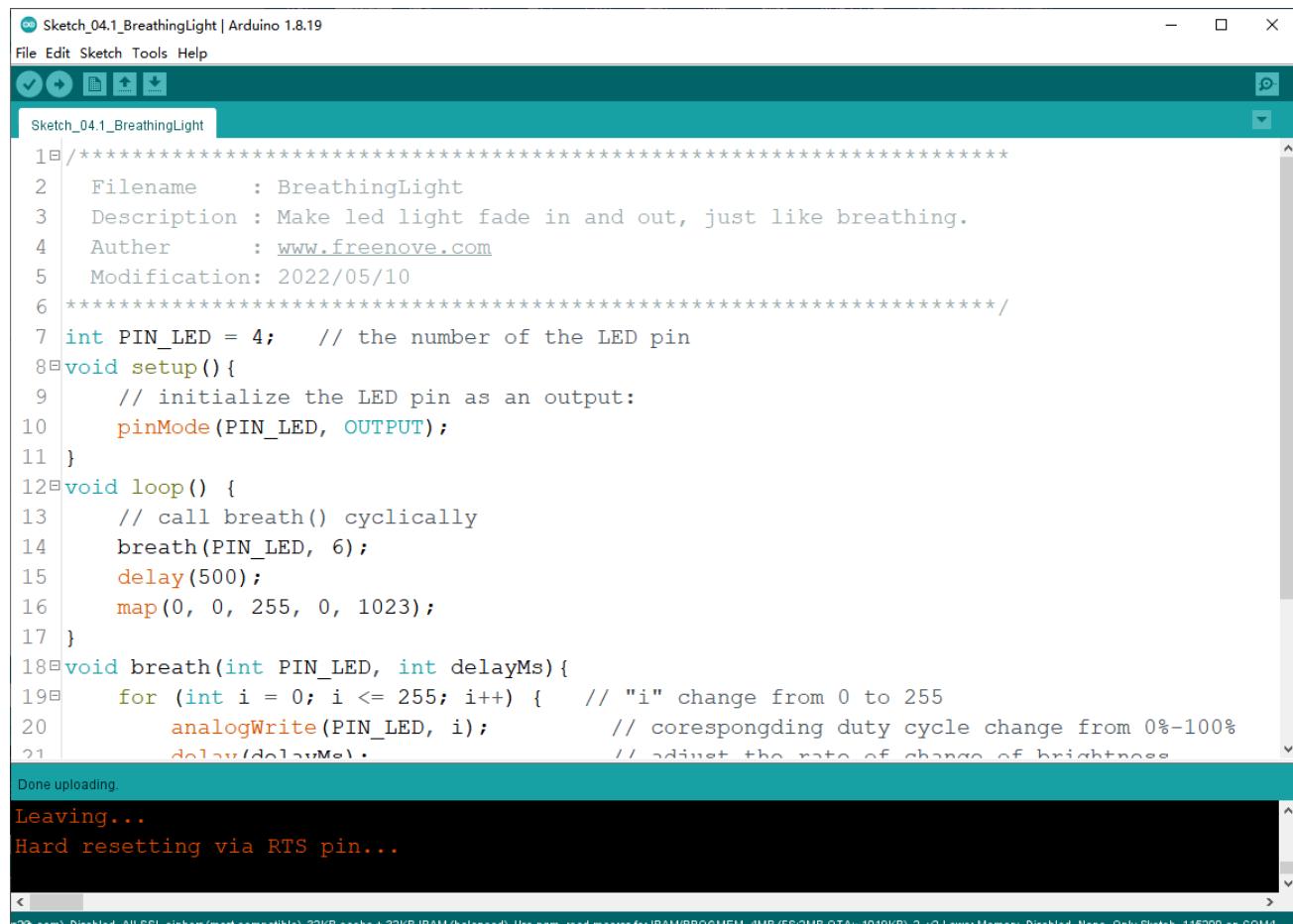
Sketch

This project is designed to make PWM output GPIO4 with pulse width increasing from 0% to 100%, and then reducing from 100% to 0% gradually.

Upload following sketch:

Freenove_Super_Starter_Kit_for_ESP8266\Sketches\Sketch_04.1_BreathingLight

Sketch_04.1_BreathingLight



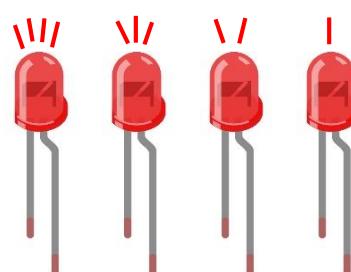
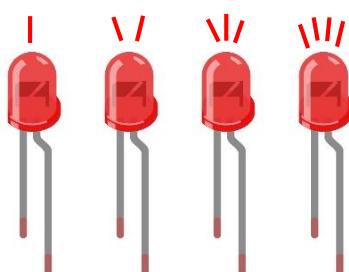
```

Sketch_04.1_BreathingLight | Arduino 1.8.19
File Edit Sketch Tools Help
Sketch_04.1_BreathingLight
1 // ****
2   Filename    : BreathingLight
3   Description : Make led light fade in and out, just like breathing.
4   Author     : www.freenove.com
5   Modification: 2022/05/10
6 ****
7 int PIN_LED = 4;    // the number of the LED pin
8 void setup(){
9     // initialize the LED pin as an output:
10    pinMode(PIN_LED, OUTPUT);
11 }
12 void loop() {
13     // call breath() cyclically
14     breath(PIN_LED, 6);
15     delay(500);
16     map(0, 0, 255, 0, 1023);
17 }
18 void breath(int PIN_LED, int delayMs) {
19     for (int i = 0; i <= 255; i++) {    // "i" change from 0 to 255
20         analogWrite(PIN_LED, i);        // corespongding duty cycle change from 0%-100%
21         delay(delayMs);             // adjust the rate of change of brightness
}
Done uploading.
Leaving...
Hard resetting via RTS pin...

```

s26 (oom), Disabled, All SSL ciphers (most compatible), 32KB cache + 32kB IRAM (balanced), Use pgm_read macros for IRAM/PROGMEM, 4MB (FS:2MB OTA:~1019KB), 2, v2 Lower Memory, Disabled, None, Only Sketch, 115200 on COM4

Download the code to ESP8266, and you'll see that LED is turned from on to off and then from off to on gradually like breathing.



The following is the program code:

```

1 int PIN_LED = 4; // the number of the LED pin
2 void setup() {
3     // initialize the LED pin as an output:
4     pinMode(PIN_LED, OUTPUT);
5 }
6 void loop() {
7     // call breath() cyclically
8     breath(PIN_LED, 6);
9     delay(500);
10    map(0, 0, 255, 0, 1023);
11 }
12 void breath(int PIN_LED, int delayMs) {
13     for (int i = 0; i <= 255; i++) { // "i" change from 0 to 255
14         analogWrite(PIN_LED, i); // corespongding duty cycle change from 0%-100%
15         delay(delayMs); // adjust the rate of change of brightness
16     }
17     for (int i = 255; i >= 0; i--) { // "i" change from 255 to 0
18         analogWrite(PIN_LED, i); // corespongding duty cycle change from 0%-100%
19         delay(delayMs); // adjust the rate of change in brightness
20     }
21 }
```

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

```

13 for (int i = 0; i <= 255; i++) { // "i" change from 0 to 255
14     analogWrite(PIN_LED, i); // corespongding duty cycle change from 0%-100%
15     delay(delayMs); // adjust the rate of change of brightness
16 }
17 for (int i = 255; i >= 0; i--) { // "i" change from 255 to 0
18     analogWrite(PIN_LED, i); // corespongding duty cycle change from 0%-100%
19     delay(delayMs); // adjust the rate of change in brightness
20 }
```

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

Reference

analogWrite(pin, value);	
---------------------------------	--

Arduino IDE provides the function, analogWrite(pin, value), which can make ports directly output PWM waves. In the function called analogWrite(pin, value), the parameter "pin" specifies the port used to output PWM wave. In ESP8266, The range of value is 0-1023, which represents the duty cycle of 0%-100%. In order to use this function, we need to set the port to output mode.

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

This function is used to remap a value, which will return a new value whose percentage in the range of toLow-toHigh is equal to the percentage of "value" in the range of fromLow-fromHigh. For example, 1 is the maximum in the range of 0-1 and the maximum value in the scope of 0-2 is 2, that is, the result value of map (1, 0, 1, 0, 2) is 2.

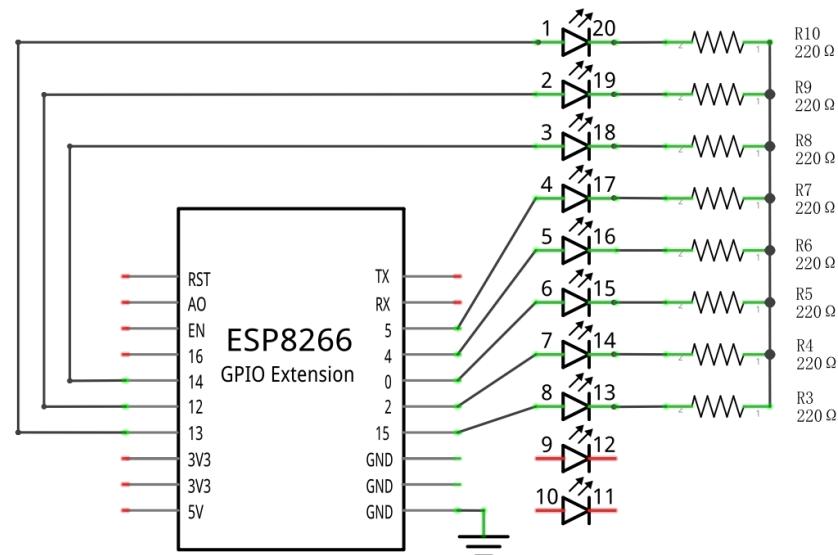
For more related functions, please refer to <https://www.arduino.cc/reference/en/>

Project 4.2 Meteor Flowing Light

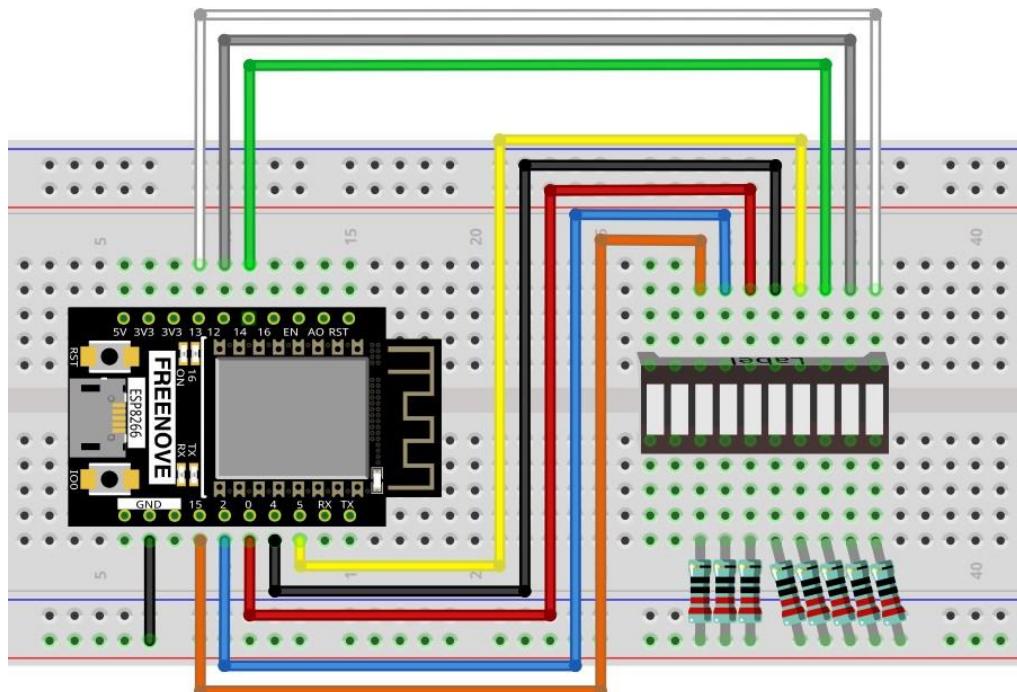
After learning about PWM, we can use it to control LED bar graph and realize a cooler flowing light. The component list, circuit, and hardware are exactly consistent with the project Flowing Light.

Circuit

Schematic diagram



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



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

Any concerns? ✉ support@freenove.com

Sketch

Meteor flowing light will be implemented with PWM.

Upload following sketch:

Freenove_Super_Starter_Kit_for_ESP8266\Sketches\Sketch_04.2_FlowingLight2

Sketch_04.2_FlowingLight2

Download the code to ESP8266, and LED bar graph will gradually light up and out from left to right, then light up and out from right to left.

The screenshot shows the Arduino IDE interface. The title bar says "Sketch_04.2_FlowingLight2 | Arduino 1.8.19". The menu bar includes File, Edit, Sketch, Tools, Help. Below the menu is a toolbar with icons for save, undo, redo, cut, copy, paste, upload, and others. The main code editor window contains the following C++ code:

```

1 // ****
2   Filename      : FlowingLight2
3   Description   : More cool flowing light.
4   Author        : www.freenove.com
5   Modification  : 2022/05/10
6 ****
7 const byte ledPins[] = {13, 12, 14, 5, 4, 0, 2, 15};    //define led pins
8 const int dutys[] = {0, 0, 0, 0, 0, 0, 0, 0, 0,
9   |           1023, 512, 256, 128, 64, 32, 16, 8, 4, 2,
10          0, 0, 0, 0, 0, 0, 0, 0, 0,
11        };      //define the pwm dutys
12 int ledCounts;           //led counts
13 int delayTimes = 50;     //flowing speed ,the smaller, the faster
14 void setup() {
15   ledCounts = sizeof(ledPins); //get the led counts
16 }
17
18 void loop() {
19   for (int i = 0; i < 20; i++) {           //flowing one side to other side
20     for (int j = 0; j < ledCounts; j++) {
21       analogWrite(ledPins[j], dutys[i + j]);
22     }
}

```

Below the code editor, a status bar says "Done uploading." followed by "Leaving... Hard resetting via RTS pin...". At the bottom, a message box displays memory information: "Most compatible), 32KB cache + 32KB IRAM (balanced), Use pgm_read macros for IRAM/PROGMEM, 4MB (FS:3MB OTA:~512KB), 2, v2 Lower Memory, Disabled, None, Only Sketch, 115200 on COM4".

The following is the program code:

```

1 const byte ledPins[] = {13, 12, 14, 5, 4, 0, 2, 15};    //define led pins
2 const int dutys[] = {0, 0, 0, 0, 0, 0, 0, 0, 0,
3   |           1023, 512, 256, 128, 64, 32, 16, 8, 4, 2,
4          0, 0, 0, 0, 0, 0, 0, 0, 0,
5        };      //define the pwm dutys
6 int ledCounts;           //led counts
7 int delayTimes = 50;     //flowing speed ,the smaller, the faster
8 void setup() {
9   ledCounts = sizeof(ledPins); //get the led counts
10 }

```

```
11
12 void loop() {
13     for (int i = 0; i < 20; i++) {          //flowing one side to other side
14         for (int j = 0; j < ledCounts; j++) {
15             analogWrite(ledPins[j], dutys[i + j]);
16         }
17         delay(delayTimes);
18     }
19     for (int i = 0; i < 20; i++) {          //flowing one side to other side
20         for (int j = ledCounts - 1; j > -1; j--) {
21             analogWrite(ledPins[j], dutys[i + (ledCounts - 1 - j)]);
22         }
23         delay(delayTimes);
24     }
25 }
```

Defined 10 GPIO and 30 pulse width values.

```
1 const byte ledPins[] = {13, 12, 14, 5, 4, 0, 2, 15};      //define led pins
2 const int dutys[] = {0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
3                         1023, 512, 256, 128, 64, 32, 16, 8, 4, 2,
4                         0, 0, 0, 0, 0, 0, 0, 0, 0, 0
5 } ;                                         //define the pwm dutys
```

In loop(), a nested for loop is used to control the pulse width of the PWM, and LED bar graph moves one grid after each 1 is added in the first for loop, gradually changing according to the values in the array duties. As shown in the table below, the value of the second row is the value in the array duties, and the 10 green squares in each row below represent the 10 LEDs on the LED bar graph. Every 1 is added to I , the value of the LED bar graph will move to the right by one grid, and when it reaches the end, it will move from the end to the starting point, achieving the desired effect.

0	1	2	3	4	5	7	8	9	1	11	1	1	1	1	1	1	1	2	2	2	2	2	2	2	3	0
d	0	0	0	0	0	0	0	0	0	10	5	2	1	6	3	1	8	4	2	0	0	0	0	0	0	0
i										23	1	5	2	4	2	6										
0																										
1																										
2																										
3																										
...																										
1																										
8																										
1																										
9																										
2																										
0																										

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

```

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

```



Chapter 5 RGB LED

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

Project 5.1 Random Color Light

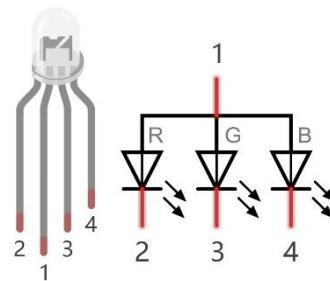
In this project, we will make a multicolored LED. And we can control RGB LED to switch different colors automatically.

Component List

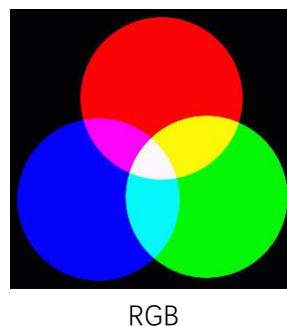
ESP8266 x1		USB cable
Breadboard x1		
RGBLED x1	Resistor 220Ω x3	Jumper wire M/M x5

Related knowledge

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



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

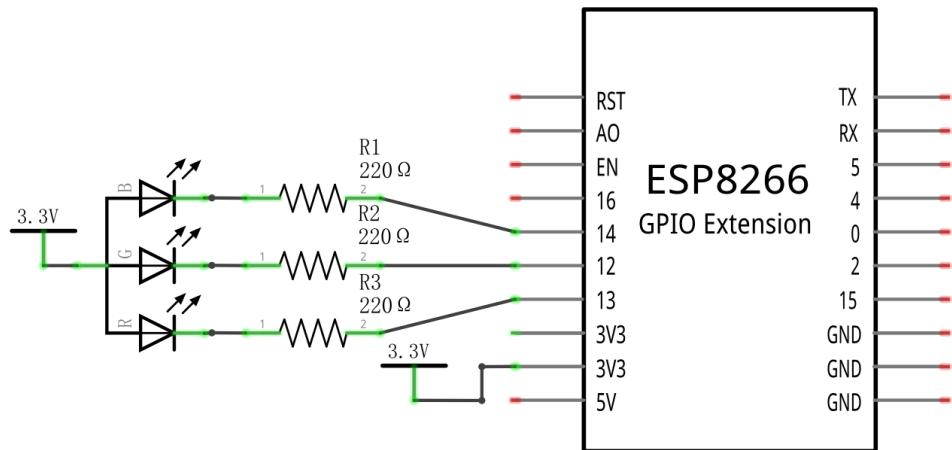


If we use three 8-bit PWMs to control the RGB LED, in theory, we can create $2^8 \times 2^8 \times 2^8 = 16777216$ (16 million) colors through different combinations.

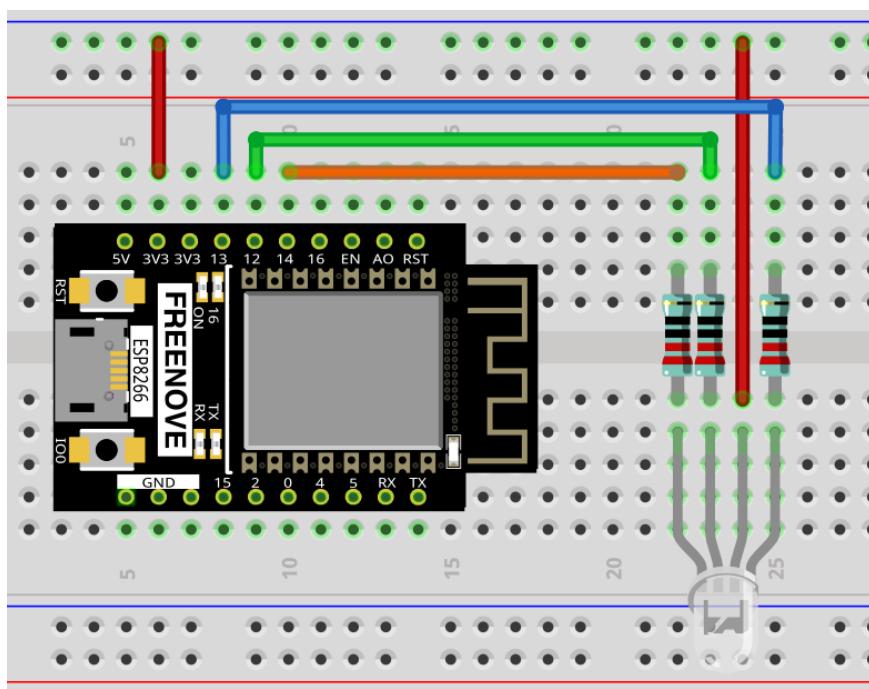


Circuit

Schematic diagram



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



Sketch

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

Upload following sketch:

Freenove_Super_Starter_Kit_for_ESP8266\Sketch_05.1_ColorfulLight

Sketch_05.1_ColorfulLight

The screenshot shows the Arduino IDE interface. The title bar says "Sketch_05.1_RandomColorLight | Arduino 1.8.19". The menu bar includes File, Edit, Sketch, Tools, Help. The toolbar has icons for file operations. The main window displays the code for "Sketch_05.1_RandomColorLight". The code defines a random color LED driver. It includes comments for the filename, description, author, and modification date. It defines three pins (14, 12, 13) for red, green, and blue LEDs. The setup() function initializes the pins. The loop() function generates random values for red, green, and blue, sets the color, and delays for 200ms. A "void setColor(uint8_t r, uint8_t g, uint8_t b)" function is also present. The status bar at the bottom shows "Done uploading.", "Leaving...", "Hard resetting via RTS pin...", and system information like "s2d.com, Disabled, All SSL ciphers (most compatible), 32KB cache + 32kB RAM (balanced), Use pgm_read macros for IRAM/PROGMEM, 4MB (FS:2MB OTA:~1019KB), 2, v2 Lower Memory, Disabled, None, Only Sketch, 115200 on COM4".

```
Sketch_05.1_RandomColorLight | Arduino 1.8.19
File Edit Sketch Tools Help
Sketch_05.1_RandomColorLight
1 //*****
2   Filename      : ColorfulLight
3   Description   : Use RGBLED to show random color.
4   Author        : www.freenove.com
5   Modification: 2022/05/10
6 ****
7 const byte ledPins[] = {14, 12, 13};      //define red, green, blue led pins
8 int red, green, blue;
9 void setup() {
10
11 }
12
13 void loop() {
14   red = random(0, 256);
15   green = random(0, 256);
16   blue = random(0, 256);
17   setColor(red, green, blue);
18   delay(200);
19 }
20
21 void setColor(uint8_t r, uint8_t g, uint8_t b)
Done uploading.
Leaving...
Hard resetting via RTS pin...
s2d.com, Disabled, All SSL ciphers (most compatible), 32KB cache + 32kB RAM (balanced), Use pgm_read macros for IRAM/PROGMEM, 4MB (FS:2MB OTA:~1019KB), 2, v2 Lower Memory, Disabled, None, Only Sketch, 115200 on COM4
```

With the code downloaded to ESP8266, RGB LED begins to display random colors.

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

Any concerns? ✉ support@freenove.com



The following is the program code:

```

1  const byte ledPins[] = {14, 12, 13};      //define red, green, blue led pins
2  int red, green, blue;
3  void setup() {
4
5
6  void loop() {
7    red = random(0, 256);
8    green = random(0, 256);
9    blue = random(0, 256);
10   setColor(red, green, blue);
11   delay(200);
12 }
13
14 void setColor(byte r, byte g, byte b) {
15   analogWrite(ledPins[0], 255 - r); //Common anode LED, low level to turn on the led.
16   analogWrite(ledPins[1], 255 - g);
17   analogWrite(ledPins[2], 255 - b);
18 }
```

Define RGB LED pins

```
1  const byte ledPins[] = {14, 12, 13};      //define red, green, blue led pins
```

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

```

14 void setColor(byte r, byte g, byte b) {
15   analogWrite(chns[0], 255 - r); //Common anode LED, low level to turn on the led.
16   analogWrite(chns[1], 255 - g);
17   analogWrite(chns[2], 255 - b);
18 }
```

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

```

7  red = random(0, 256);
8  green = random(0, 256);
9  blue = random(0, 256);
10  setColor(red, green, blue);
11  delay(200);
```

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

long random(min, max);

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

Project 5.2 Gradient Color Light

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

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

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



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

Sketch

Upload following sketch:

Freenove_Super_Starter_Kit_for_ESP8266\Sketch_05.2_SoftColorfullLight
Sketch_05.2_SoftColorfullLight

```

Sketch_05.2_GradientColorLight | Arduino 1.8.19
File Edit Sketch Tools Help
Sketch_05.2_GradientColorLight
1/* ****
2  Filename   : SoftColorfulLight
3  Description : Colorful light with gradually changing color.
4  Author     : www.freenove.com
5  Modification: 2022/05/10
6 ****
7 const byte ledPins[] = {14, 12, 13};      //define led pins
8 void setup() {
9 }
10 void loop() {
11   for (int i = 0; i < 256; i++) {
12     setColor(wheel(i));
13     delay(20);
14   }
15 }
16 void setColor(long rgb) {
17   analogWrite(ledPins[0], 255 - (rgb >> 16) & 0xFF);
18   analogWrite(ledPins[1], 255 - (rgb >> 8) & 0xFF);
19   analogWrite(ledPins[2], 255 - (rgb >> 0) & 0xFF);
20 }
21
Done uploading.
Leaving...
Hard resetting via RTS pin...

```

s.20.0em), Disabled. All SSL ciphers (most compatible). 32KB cache + 32KB IRAM (balanced). Use pgm_read macros for I^PRAM/PROGMEM. 4MB (FS:2MB OTA:~1019KB), 2, v2 Lower Memory, Disabled, None, Only Sketch, 115200 on COM4

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



The following is the program code:

```

1  const byte ledPins[] = {14, 12, 13};      //define led pins
2  void setup() {
3  }
4  void loop() {
5      for (int i = 0; i < 256; i++) {
6          setColor(wheel(i));
7          delay(20);
8      }
9  }
10 void setColor(long rgb) {
11     analogWrite(ledPins[0], 255 - (rgb >> 16) & 0xFF);
12     analogWrite(ledPins[1], 255 - (rgb >> 8) & 0xFF);
13     analogWrite(ledPins[2], 255 - (rgb >> 0) & 0xFF);
14 }
15
16 long wheel(int pos) {
17     long WheelPos = pos % 0xff;
18     if (WheelPos < 85) {
19         return ((255 - WheelPos * 3) << 16) | ((WheelPos * 3) << 8);
20     } else if (WheelPos < 170) {
21         WheelPos -= 85;
22         return (((255 - WheelPos * 3) << 8) | (WheelPos * 3));
23     } else {
24         WheelPos -= 170;
25         return ((WheelPos * 3) << 16 | (255 - WheelPos * 3));
26     }
27 }
```

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

```

10 void setColor(long rgb) {
11     analogWrite(ledPins[0], 255 - (rgb >> 16) & 0xFF);
12     analogWrite(ledPins[1], 255 - (rgb >> 8) & 0xFF);
13     analogWrite(ledPins[2], 255 - (rgb >> 0) & 0xFF);
14 }
```

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

Chapter 6 LEDPixel

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

Project 6.1 LEDPixel

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

Component List

ESP8266 x1	USB cable
	
Breadboard x1	
Freenove 8 RGB LED Module x1	Jumper wire F/M x4
	

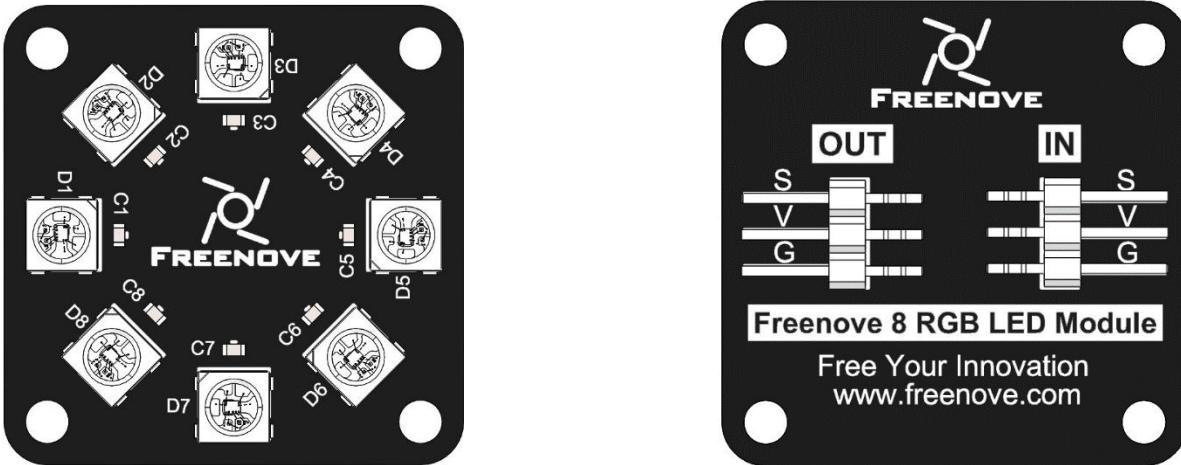
Related knowledge

Freenove 8 RGB LED Module

The Freenove 8 RGB LED Module is as below.

It consists of 8 WS2812, each of which requires only one pin to control and supports cascade. Each WS212 has integrated 3 LEDs, red, green and blue respectively, and each of them supports 256-level brightness adjustment, which means that each WS2812 can emit $2^{24}=16,777,216$ different colors.

You can use only one data pin to control eight LEDs on the module. As shown below:



And you can also control many modules at the same time. Just connect OUT pin of one module to IN pin of another module. In this way, you can use one data pin to control 8, 16, 32 ... LEDs.

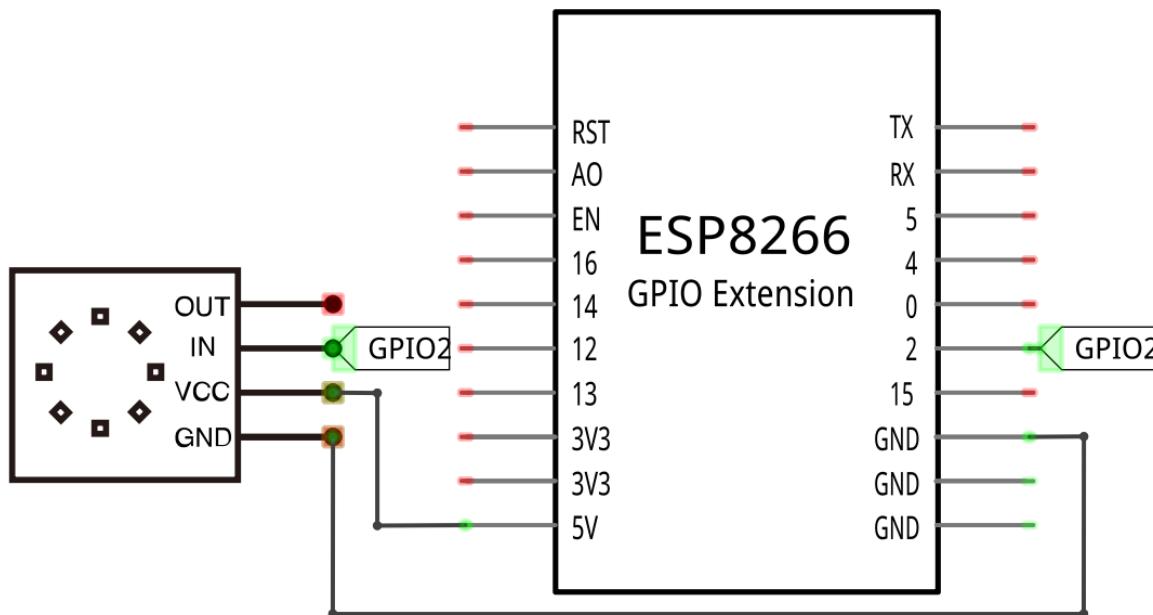


Pin description:

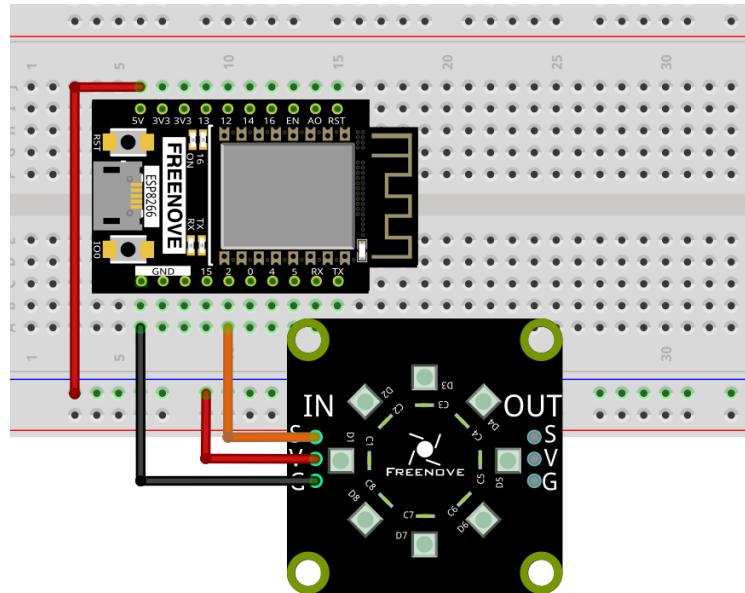
(IN)		(OUT)	
symbol	Function	symbol	Function
S	Input control signal	S	Output control signal
V	Power supply pin, +3.5V~5.5V	V	Power supply pin, +3.5V~5.5V
G	GND	G	GND

Circuit

Schematic diagram



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



Sketch

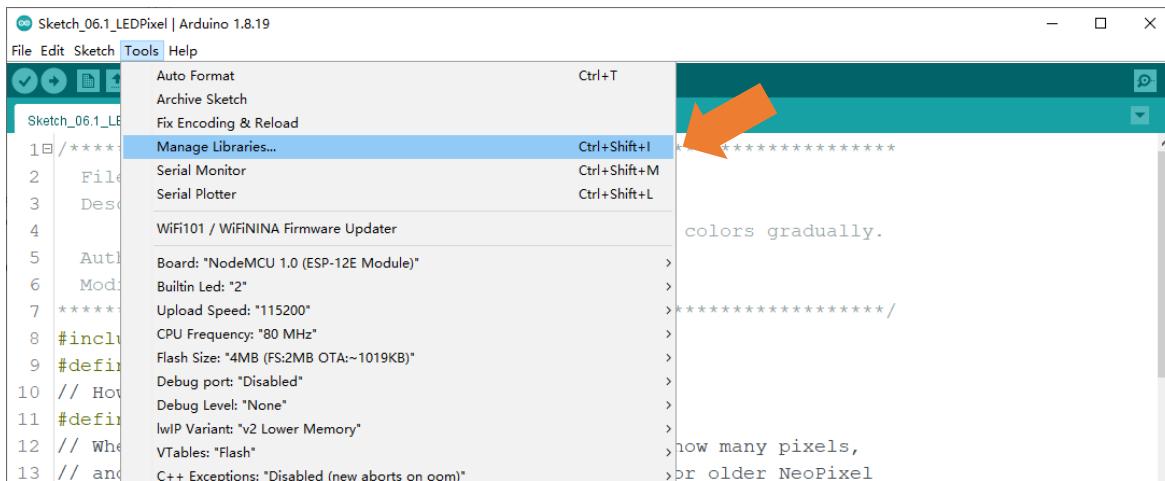
This code uses a library named "Adafruit_NeoPixel", if you have not installed it, please do so first.

Library is an important feature of the open source world, and we know that Arduino is an open source platform that everyone can contribute to. Libraries are generally licensed under the LGPL, which means you can use them for free to apply to your creations.

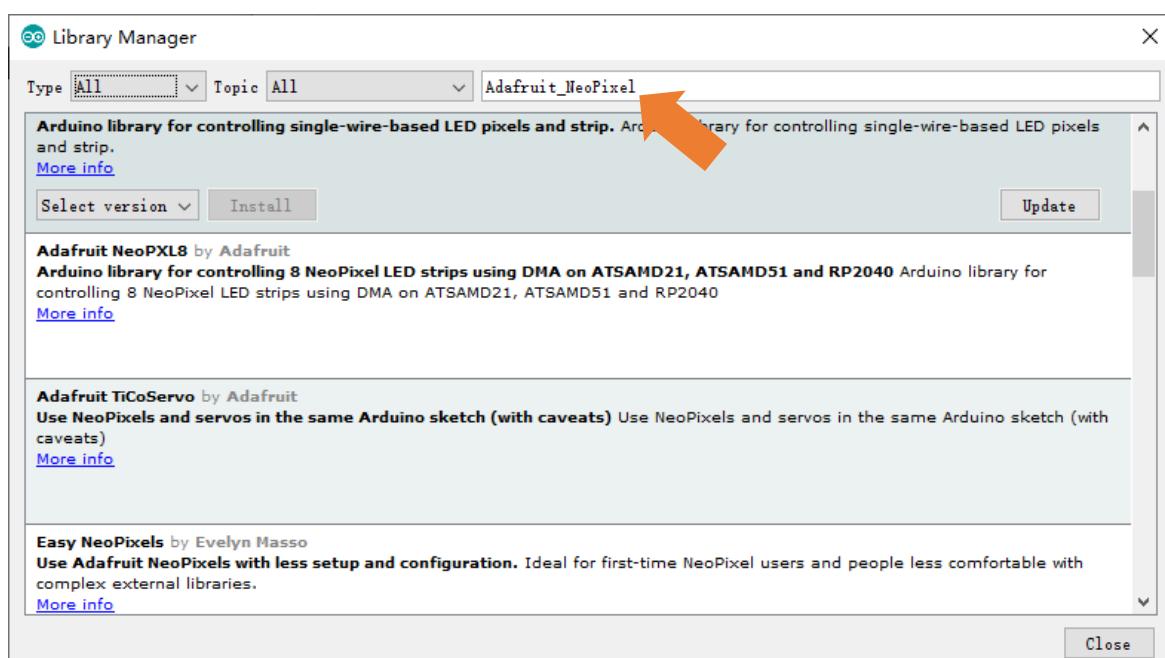
How to install the library

There are two ways to add libraries.

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



In the pop-up window, Library Manager, search for the name of the Library, "Adafruit_NeoPixel". Then click Install.



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



Upload following sketch:

Freenove_Super_Starter_Kit_for_ESP8266\C\Sketch_06.1_LEDPixel

Sketch_06.1_LEDPixel

The screenshot shows the Arduino IDE during the upload process. The top menu bar has 'File', 'Edit', 'Sketch', 'Tools', and 'Help'. The code editor shows the same sketch as before. The status bar at the bottom of the IDE window displays the message 'Done uploading.'.

Download the code to ESP8266 and RGB LED begins to light up in red, green, blue, white and black.



The following is the program code:

```

1 #include <Adafruit_NeoPixel.h>
2 #define PIN      2
3 // How many NeoPixels are attached to the Arduino?
4 #define NUMPIXELS 8 // Popular NeoPixel ring size
5 // When setting up the NeoPixel library, we tell it how many pixels,
6 // and which pin to use to send signals. Note that for older NeoPixel
7 // strips you might need to change the third parameter -- see the
8 // strandtest example for more information on possible values.
9 Adafruit_NeoPixel pixels(NUMPIXELS, PIN, NEO_GRB + NEO_KHZ800);
10 #define DELAYVAL 500 // Time (in milliseconds) to pause between pixels
11 int m_color[5][3] = { {255, 0, 0}, {0, 255, 0}, {0, 0, 255}, {255, 255, 255}, {0, 0, 0} };
12 int delayval = 100;
13 void setup() {
14     pixels.begin(); // INITIALIZE NeoPixel strip object (REQUIRED)
15     pixels.setBrightness(10); // set the brightness.
16 }
17 void loop() {
18     pixels.clear(); // Set all pixel colors to 'off'
19     // The first NeoPixel in a strand is #0, second is 1, all the way up
20     // to the count of pixels minus one.
21     for (int j = 0; j < 5; j++) {
22         for(int i=0; i<NUMPIXELS; i++){ // For each pixel...
23             // pixels.Color() takes RGB values, from 0,0,0 up to 255,255,255
24             pixels.setPixelColor(i, m_color[j][0], m_color[j][1], m_color[j][2]); // Set color
25             data.
26             pixels.show(); // Send the updated pixel colors to the hardware.
27             delay(DELAYVAL); // Pause before next pass through loop
28         }
29         delay(500); // Interval time of each group of colors.
30     }
}

```

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

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

Define the pins connected to the ring and the number of LEDs on the ring.

```
2 #define PIN      2
4 #define NUMPIXELS 8 // Popular NeoPixel ring size
```

Use the above parameters to create a LEDPixel object strip.

```
9 Adafruit_NeoPixel pixels(NEOPIN, PIN, NEO_GRB + NEO_KHZ800);
```

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

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

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

```
10 #define DELAYVAL 500 // Time (in milliseconds) to pause between pixels
```

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

```
14 pixels.begin(); // INITIALIZE NeoPixel strip object (REQUIRED)
15 pixels.setBrightness(10); // set the brightness.
```

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

```
21 for (int j = 0; j < 5; j++) {
22     for(int i=0; i<NUMPIXELS; i++){ // For each pixel...
23         // pixels.Color() takes RGB values, from 0,0,0 up to 255,255,255
24         pixels.setPixelColor(i, m_color[j][0], m_color[j][1], m_color[j][2]); // Set color
25         data.
26         pixels.show(); // Send the updated pixel colors to the hardware.
27         delay(DELAYVAL); // Pause before next pass through loop
28     }
29     delay(500); // Interval time of each group of colors.
}
```

Reference

pixels(NUMPIXELS, PIN, NEO_GRB + NEO_KHZ800);

Constructor to create a LEDPixel object.

Before each use of the constructor, please add "#include <Adafruit_NeoPixel.h>"

Parameters

NUMPIXELS: The number of led.

PIN: A pin connected to an led.

NEO_GRB:

NEO_GRB: The sequence of LEDPixel module loading color is green, red and blue.

void begin(void);

Initialize the LEDPixel object

```
void setLedColorData (u8 index, u8 r, u8 g, u8 b);
void setLedColorData (u8 index, u32 rgb);
void setLedColor (u8 index, u8 r, u8 g, u8 b);
void setLedColor (u8 index, u32 rgb);
```

Set the color of led with order number n.

void show(void);

Send the color data to the led and display the set color immediately.

void setBrightness(uint8_t);

Set the brightness of the LED.

Project 6.2 Rainbow Light

In the previous project, we have mastered the use of LEDPixel. This project will realize a slightly complicated rainbow light. The component list and the circuit are exactly the same as the project fashionable light.

Sketch

Continue to use the following color model to equalize the color distribution of the 8 LEDs and gradually change.



Upload following sketch:

Freenove_Super_Starter_Kit_for_ESP8266\Sketch_06.2_RainbowLight

Sketch_06.2_RainbowLight

```
Sketch_06.2_RainbowLight | Arduino 1.8.19
File Edit Sketch Tools Help
Sketch_06.2_RainbowLight
Sketch_06.2_RainbowLight
1 // ****
2   Filename      : RainbowLight
3   Description   : Make the strip light up in rainbow colors.
4   Author        : www.freenove.com
5   Modification: 2022/05/10
6 ****
7 #include <Adafruit_NeoPixel.h>
8 #define LEDS_PIN      2
9 #define LEDS_COUNT    8 // Popular NeoPixel ring size
10 Adafruit_NeoPixel pixels(LEDS_COUNT, LEDS_PIN, NEO_GRB + NEO_KHZ800);
11 void setup() {
12   pixels.begin(); // INITIALIZE NeoPixel strip object (REQUIRED)
13   pixels.setBrightness(100); // set the brightness.
14 }
15 void loop() {
16   for (int j = 0; j < 255; j += 2) {
17     for (int i = 0; i < LEDS_COUNT; i++) {
18       pixels.setPixelColor(i, Wheel((i * 256 / LEDS_COUNT + j) & 255));
19     }
20   pixels.show();
21 }
Done uploading.
Leaving...
Hard resetting via RTS pin...

```

Download the code to ESP8266, and the Freenove 8 RGB LED Strip displays different colors and the color changes gradually.



The following is the program code:

```

1 #include <Adafruit_NeoPixel.h>
2 #define LEDS_PIN      2
3 #define LEDS_COUNT     8 // Popular NeoPixel ring size
4 Adafruit_NeoPixel pixels(LEDS_COUNT, LEDS_PIN, NEO_GRB + NEO_KHZ800);
5 void setup() {
6   pixels.begin(); // INITIALIZE NeoPixel strip object (REQUIRED)
7   pixels.setBrightness(100); // set the brightness.
8 }
9 void loop() {
10   for (int j = 0; j < 255; j += 2) {
11     for (int i = 0; i < LEDS_COUNT; i++) {
12       pixels.setPixelColor(i, Wheel((i * 256 / LEDS_COUNT + j) & 255));
13     }
14     pixels.show();
15     delay(5);
16   }
17 }
18 uint32_t Wheel(byte pos) {
19   u32 WheelPos = pos % 0xff;
20   if (WheelPos < 85) {
21     return ((255 - WheelPos * 3) << 16) | ((WheelPos * 3) << 8);
22   }
23   if (WheelPos < 170) {
24     WheelPos -= 85;
25     return (((255 - WheelPos * 3) << 8) | (WheelPos * 3));
26   }
27   WheelPos -= 170;
28   return ((WheelPos * 3) << 16 | (255 - WheelPos * 3));
29 }
```

In the loop(), two “for” loops are used, the internal “for” loop(for-j) is used to set the color of each LED, and the external “for” loop(for-i) is used to change the color, in which the self-increment value in i+=1 can be changed to change the color step distance. Changing the delay parameter changes the speed of the color change. Wheel($i * 256 / \text{LEDS_COUNT} + j$) & 255) will take color from the color model at equal intervals starting from i.

```

10   for (int j = 0; j < 255; j += 2) {
11     for (int i = 0; i < LEDS_COUNT; i++) {
```

```
12         pixels.setPixelColor(i, Wheel((i * 256 / LEDS_COUNT + j) & 255));  
13     }  
14     pixels.show();  
15     delay(5);  
16 }
```

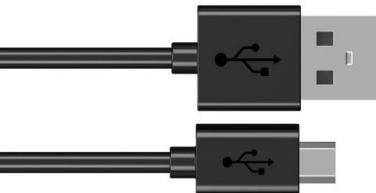
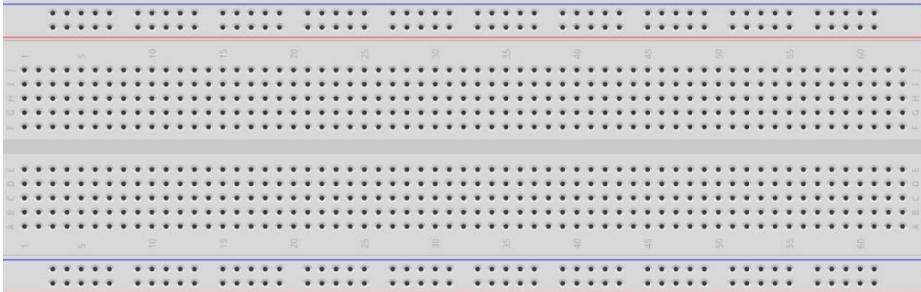
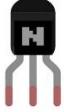
Chapter 7 Buzzer

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

Project 7.1 Doorbell

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

Component List

ESP8266 x1	USB cable			
				
Breadboard x1				
Jumper wire M/M x9				
NPN transistor x1 (S8050)	Active buzzer x1	Push button x1	Resistor 1kΩ x1	Resistor 10kΩ x2
				

Component knowledge

Buzzer

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



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

Next, we will use an active buzzer to make a doorbell and a passive buzzer to make an alarm.

How to identify active and passive buzzer?

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

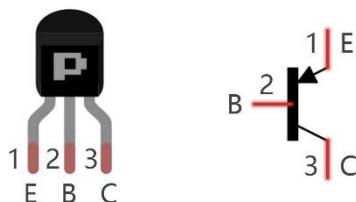


Transistor

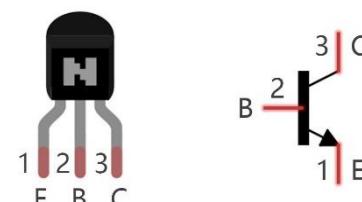
Because the buzzer requires such large current that GPIO of ESP8266 output capability cannot meet the requirement, a transistor of NPN type is needed here to amplify the current.

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

PNP transistor



NPN transistor

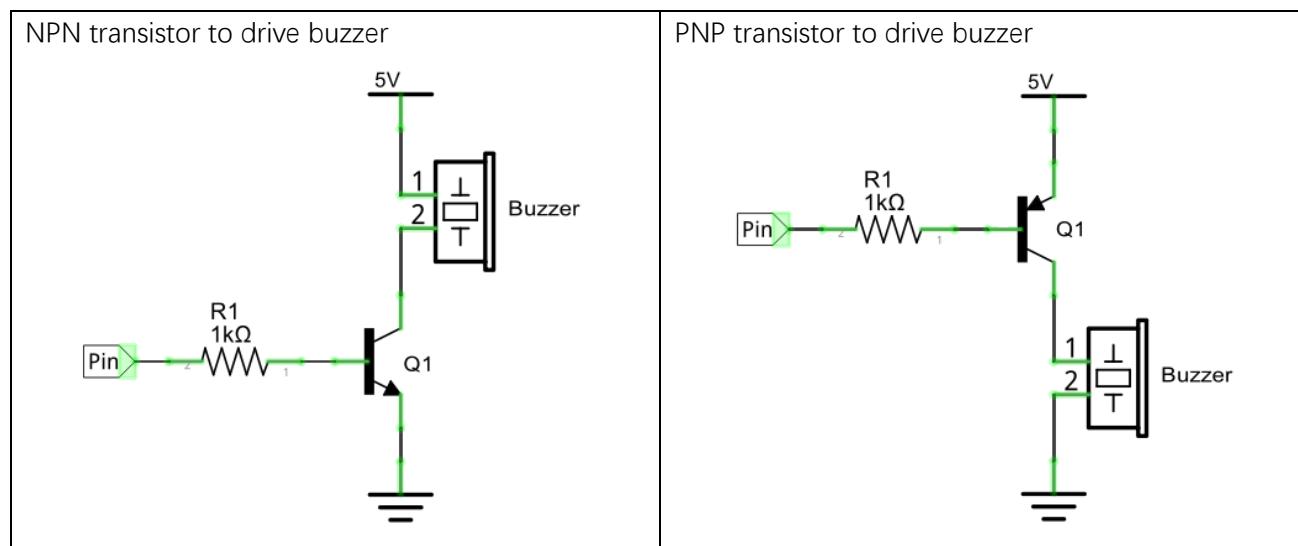


In our kit, the PNP transistor is marked with 8550, and the NPN transistor is marked with 8050.

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

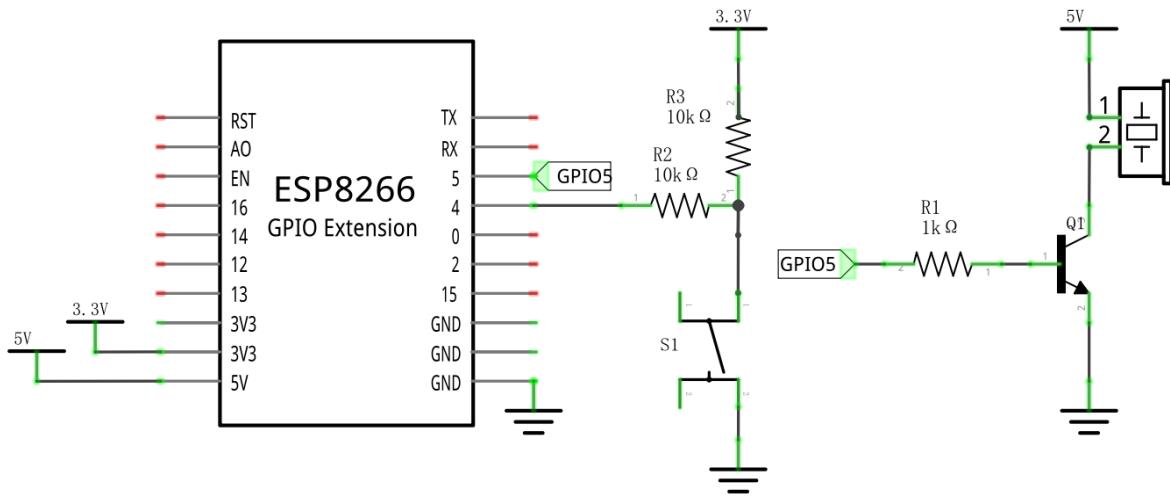
When use NPN transistor to drive buzzer, we often adopt the following method. If GPIO outputs high level, current will flow through R1, the transistor will get conducted, and the buzzer will sound. If GPIO outputs low level, no current flows through R1, the transistor will not be conducted, and buzzer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

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

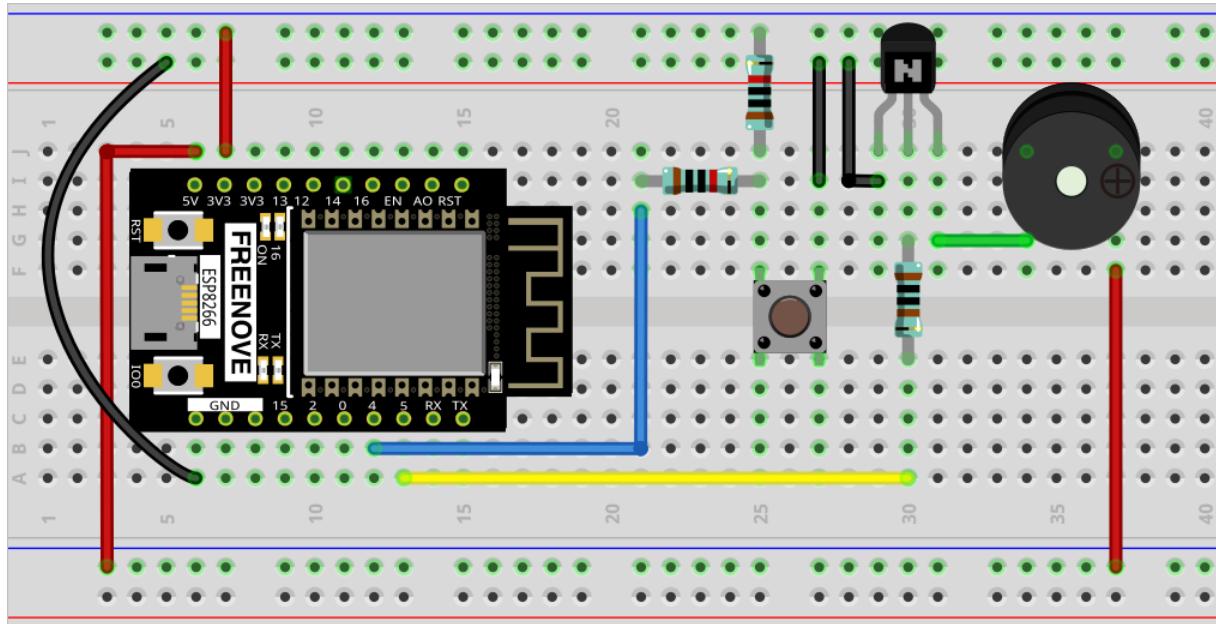


Circuit

Schematic diagram



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



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

Sketch

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

Upload following sketch:

Freenove_Super_Starter_Kit_for_ESP8266\C\Sketch_07.1_Doorbell

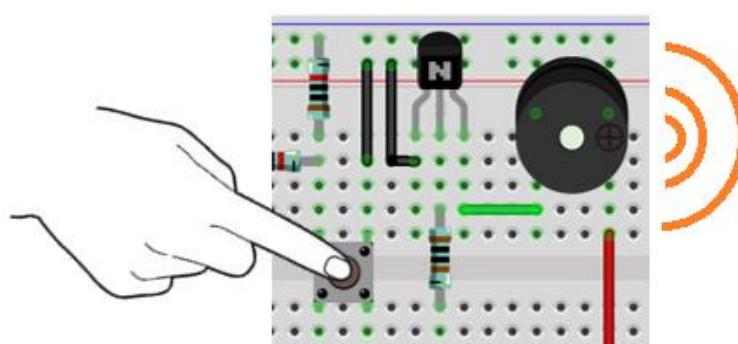
Sketch_07.1_Doorbell

The screenshot shows the Arduino IDE interface. The top menu bar includes File, Edit, Sketch, Tools, and Help. The title bar says "Sketch_07.1_Doorbell | Arduino 1.8.19". The main code area contains the following C code:

```
1 //*****  
2 // Filename : Doorbell  
3 // Description : Control active buzzer by button.  
4 // Author : www.freenove.com  
5 // Modification: 2022/05/10  
6 *****  
7 #define PIN_BUZZER 5  
8 #define PIN_BUTTON 4  
9  
10 void setup() {  
11     pinMode(PIN_BUZZER, OUTPUT);  
12     pinMode(PIN_BUTTON, INPUT);  
13 }  
14  
15 void loop() {  
16     if (digitalRead(PIN_BUTTON) == LOW) {  
17         digitalWrite(PIN_BUZZER, HIGH);  
18     } else{  
19         digitalWrite(PIN_BUZZER, LOW);  
20     }  
21 }
```

The bottom status bar indicates "Done uploading." and "Leaving... Hard resetting via RTS pin...". The serial monitor window shows the message "Hard resetting via RTS pin...".

Download the code to ESP8266, press the push button switch and the buzzer will sound. Release the push button switch and the buzzer will stop.





The following is the program code:

```
1 #define PIN_BUZZER 5
2 #define PIN_BUTTON 4
3
4 void setup() {
5     pinMode(PIN_BUZZER, OUTPUT);
6     pinMode(PIN_BUTTON, INPUT);
7 }
8
9 void loop() {
10    if (digitalRead(PIN_BUTTON) == LOW) {
11        digitalWrite(PIN_BUZZER, HIGH);
12    }else{
13        digitalWrite(PIN_BUZZER, LOW);
14    }
15 }
```

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

Project 7.2 Alertor

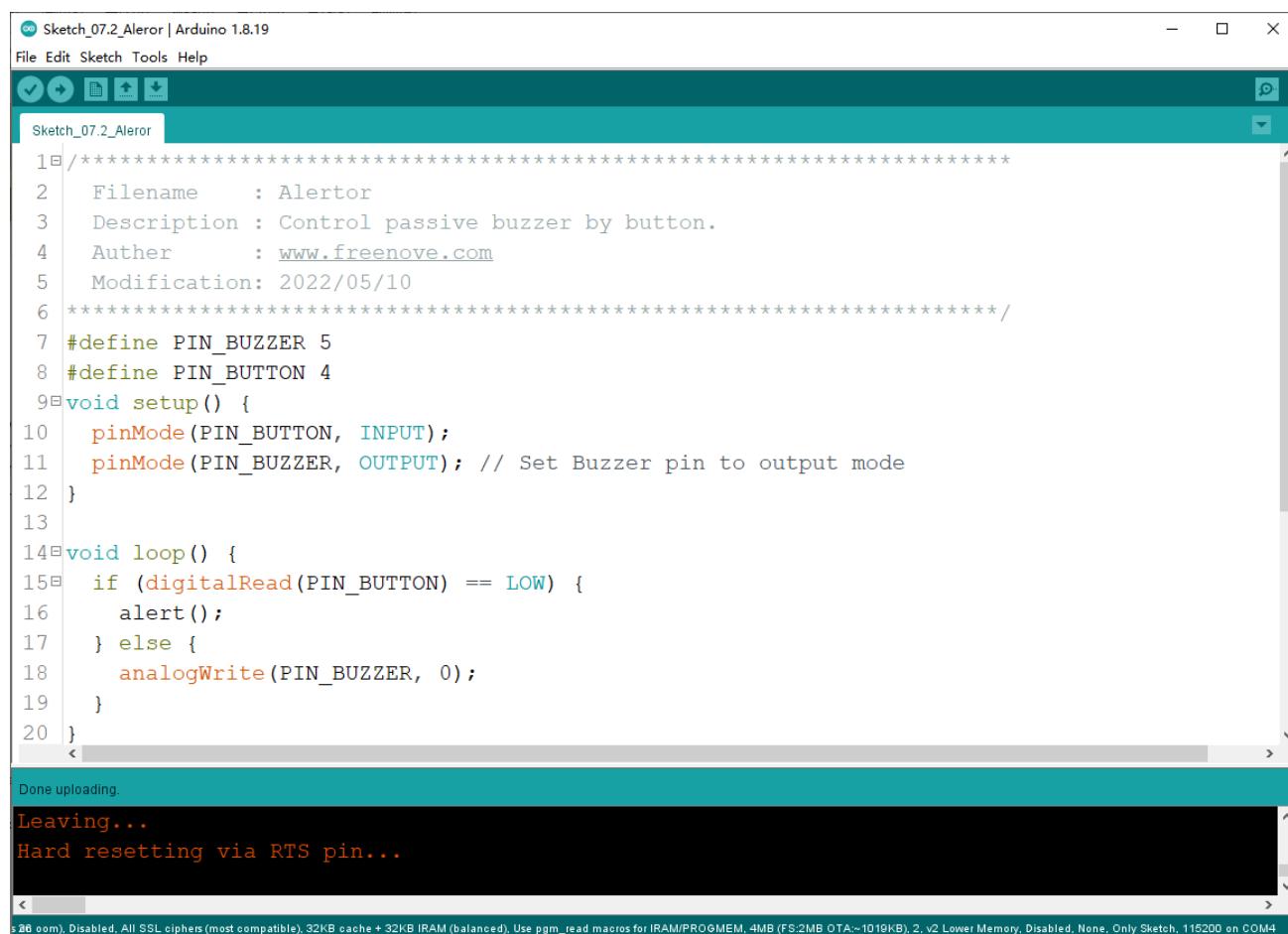
Next, we will use a passive buzzer to make an alarm.

Component list and the circuit is similar to the last section. In the Doorbell circuit only the **active buzzer** needs to be **replaced** with a **passive buzzer**.

Sketch

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

Sketch_07.2_Alertor

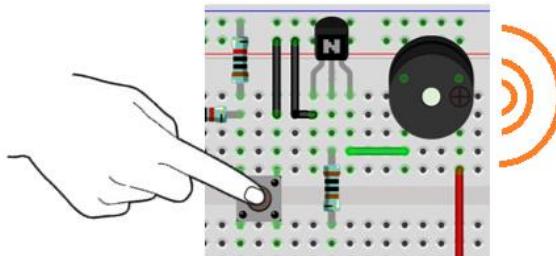


The screenshot shows the Arduino IDE interface with the sketch named "Sketch_07.2_Alertor". The code is as follows:

```
1 // ****
2 Filename      : Alertor
3 Description   : Control passive buzzer by button.
4 Author        : www.freenove.com
5 Modification: 2022/05/10
6 ****
7 #define PIN_BUZZER 5
8 #define PIN_BUTTON 4
9 void setup() {
10    pinMode(PIN_BUTTON, INPUT);
11    pinMode(PIN_BUZZER, OUTPUT); // Set Buzzer pin to output mode
12 }
13
14 void loop() {
15    if (digitalRead(PIN_BUTTON) == LOW) {
16        alert();
17    } else {
18        analogWrite(PIN_BUZZER, 0);
19    }
20 }
```

The serial monitor at the bottom shows the message: "Done uploading. Leaving... Hard resetting via RTS pin..."

Download the code to ESP8266, press the button, then alarm sounds. And when the button is released, the alarm will stop sounding.



The following is the program code:

```

1 #define PIN_BUZZER 5
2 #define PIN_BUTTON 4
3 void setup() {
4     pinMode(PIN_BUTTON, INPUT);
5     pinMode(PIN_BUZZER, OUTPUT); // Set Buzzer pin to output mode
6     Serial.begin(115200);
7 }
8 void loop() {
9     if (digitalRead(PIN_BUTTON) == LOW) {
10         alert();
11     }
12     else {
13         analogWrite(PIN_BUZZER, 0);
14     }
15 }
16 void alert() {
17     float sinVal;           // Define a variable to save sine value
18     int toneVal;           // Define a variable to save sound frequency
19     for (int x = 0; x < 360; x++) { // X from 0 degree->360 degree
20         sinVal = sin(x * (PI / 180)); // Calculate the sine of x
21         toneVal = 500 + sinVal * 500; //Calculate sound frequency according to the sine of x
22         tone(PIN_BUZZER, toneVal);      // Output sound frequency to buzzerPin
23         delay(1);
24         noTone(PIN_BUZZER);
25         delay(1);
26         Serial.printf("Running %d \n", toneVal);
27     }
28 }
```

In the code, use one loop to control the sound frequency, varying according to sine curve in the range of 500 ± 500 .

```

19     for (int x = 0; x < 360; x++) { // X from 0 degree->360 degree
20         sinVal = sin(x * (PI / 180)); // Calculate the sine of x
21         toneVal = 500 + sinVal * 500; //Calculate sound frequency according to the sine of x
```

```
22     tone(PIN_BUZZER, toneVal);           // Output sound frequency to buzzerPin
23     delay(1);
24     noTone(PIN_BUZZER);
25     delay(1);
26     Serial.printf("Running %d \n", toneVal);
27 }
```

The parameter of sin() function is radian, so we need convert unit of π from angle to radian first .

Reference

tone(pin, frequency)

Generates a square wave of the specified frequency (and 50% duty cycle) on a pin.

Verify and upload the code, passive buzzer starts making a warning sound.

You can try using PNP transistor to complete the project of this chapter again.



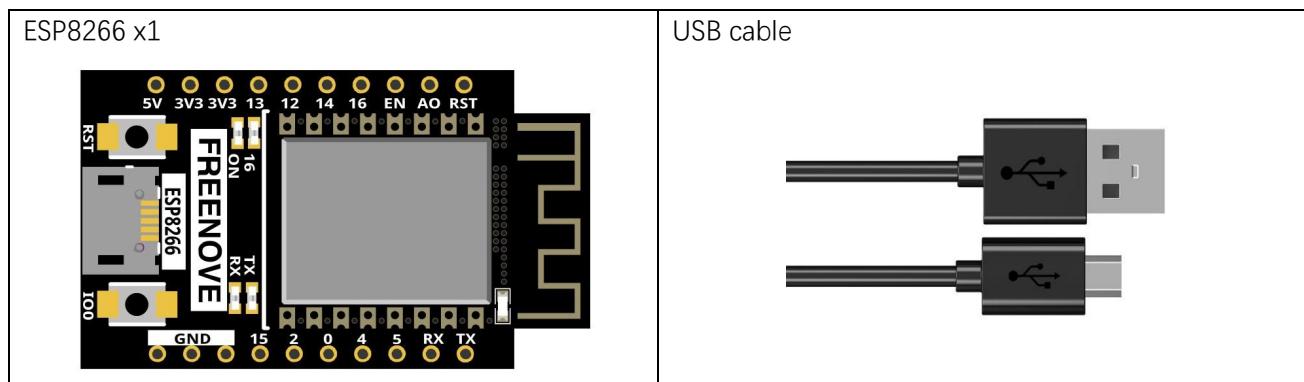
Chapter 8 Serial Communication

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

Project 8.1 Serial Print

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

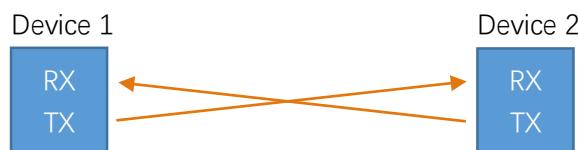
Component List



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.

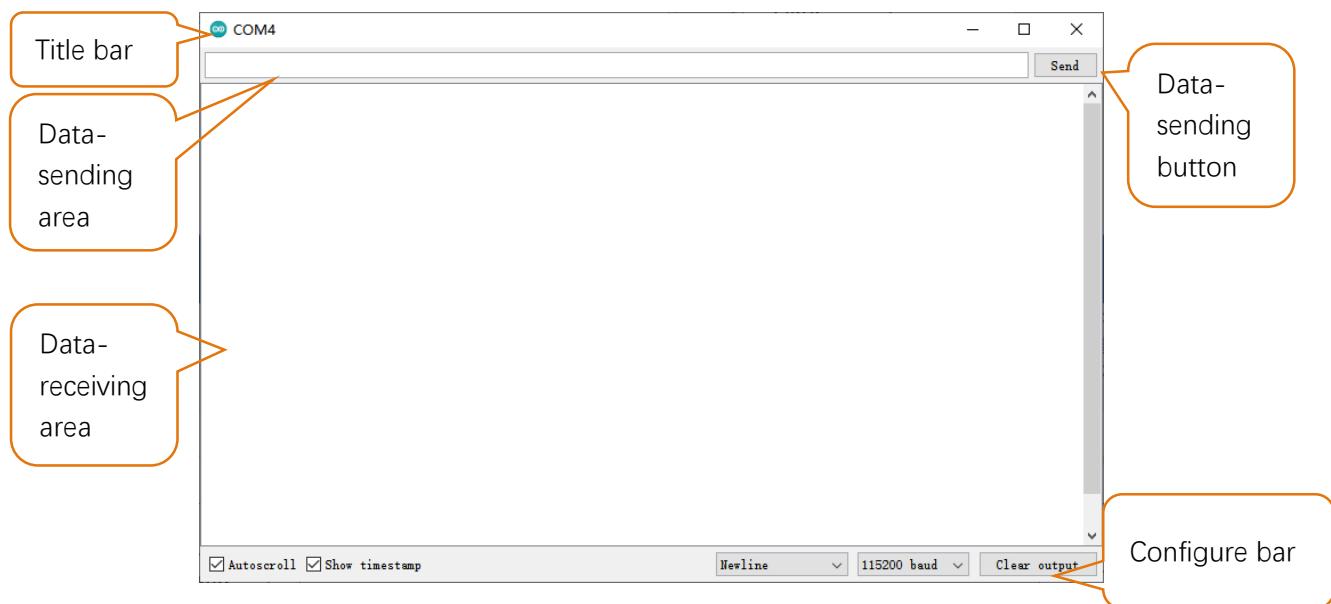


Arduino Software also uploads code to Freenove ESP8266 through the serial connection.

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".



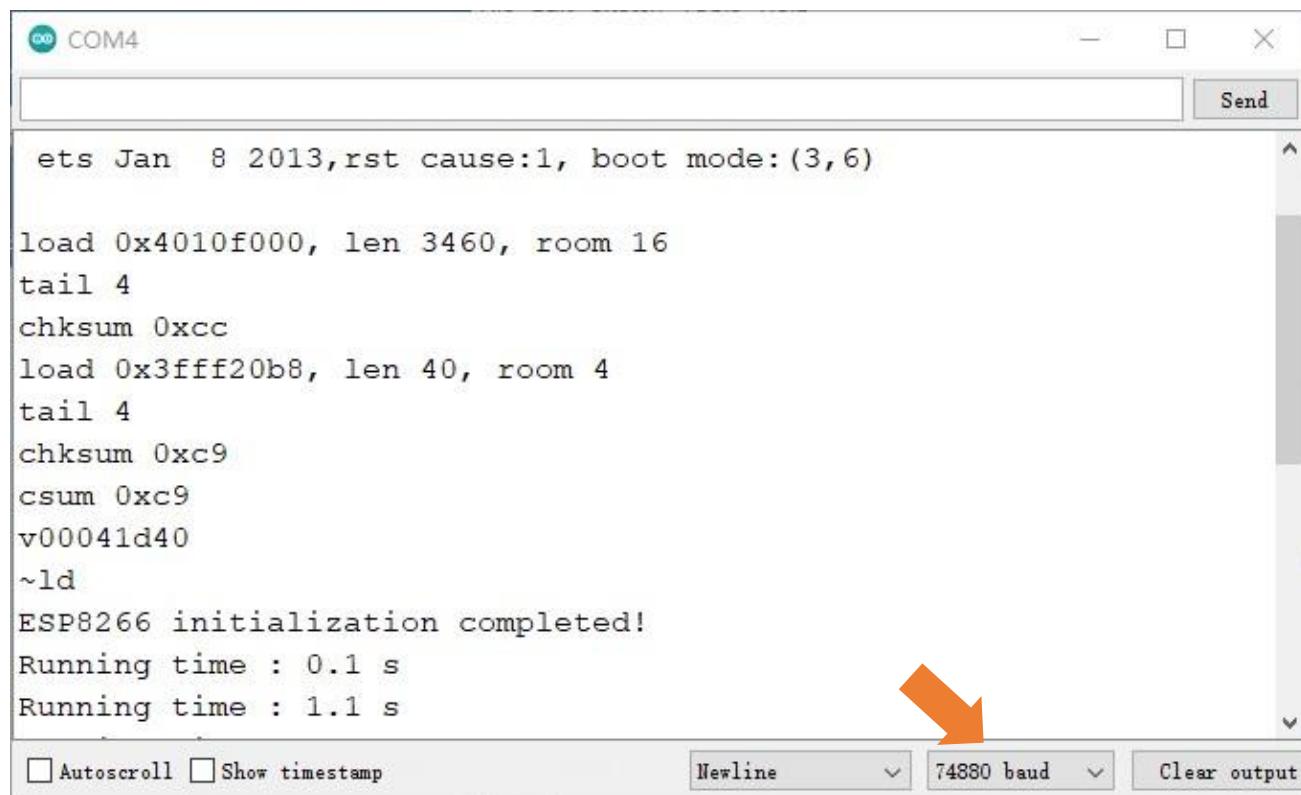


When the ESP8266 is powered on, the default baud rate is 74880. The default communication and serial port in the ESP8266 firmware is 115200. So if you set the serial port to 74880, this time can be displayed normally. If the baud rate is not 74880, some garbled characters will appear. Please do not worry.

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("ESP8266 initialization completed!");
}

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

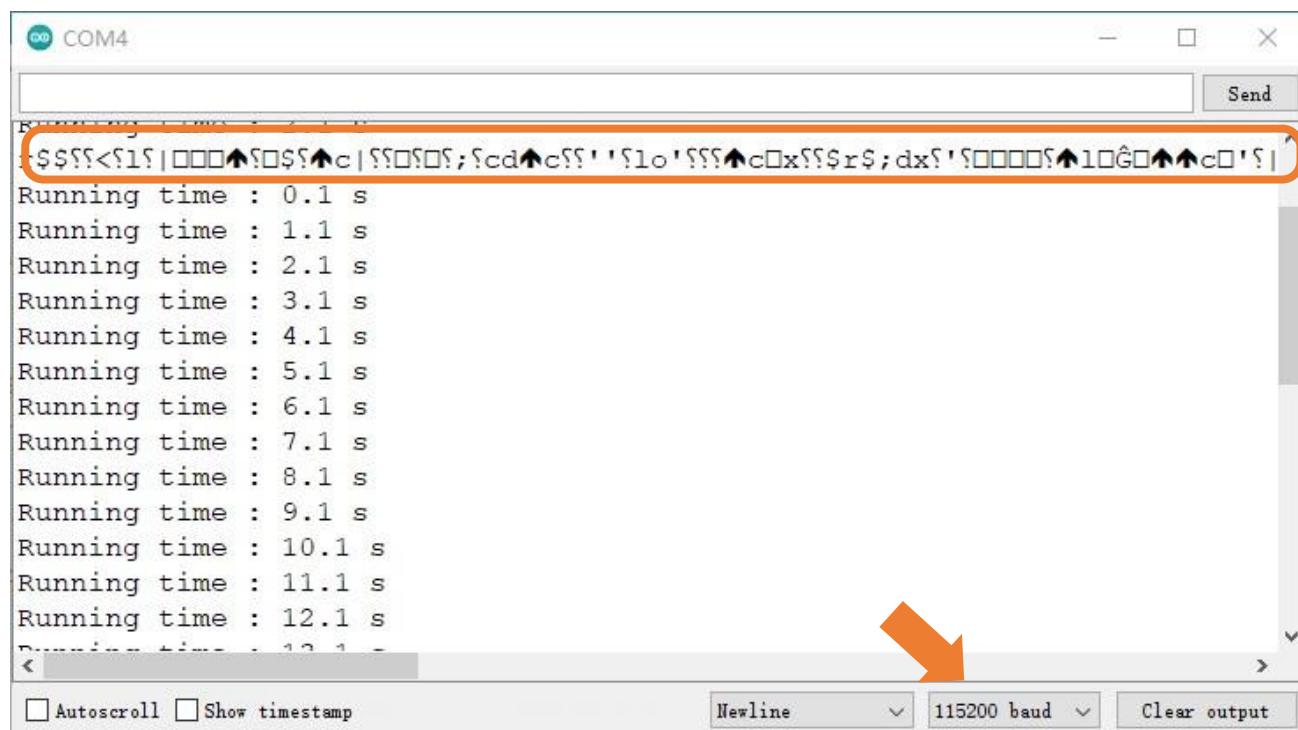


Any concerns? [✉ support@freenove.com](mailto:support@freenove.com)

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

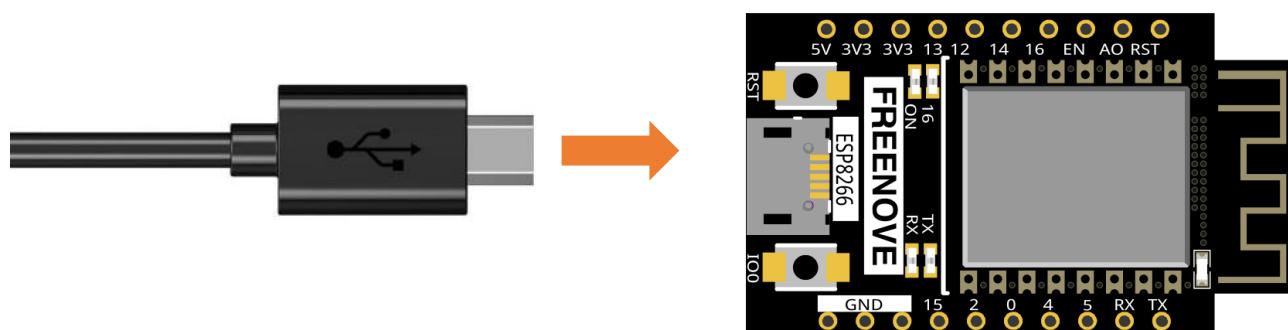
```
void setup() {
    Serial.begin(115200);
    Serial.println("ESP8266 initialization completed!");
}

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_08.1_SerialPrinter

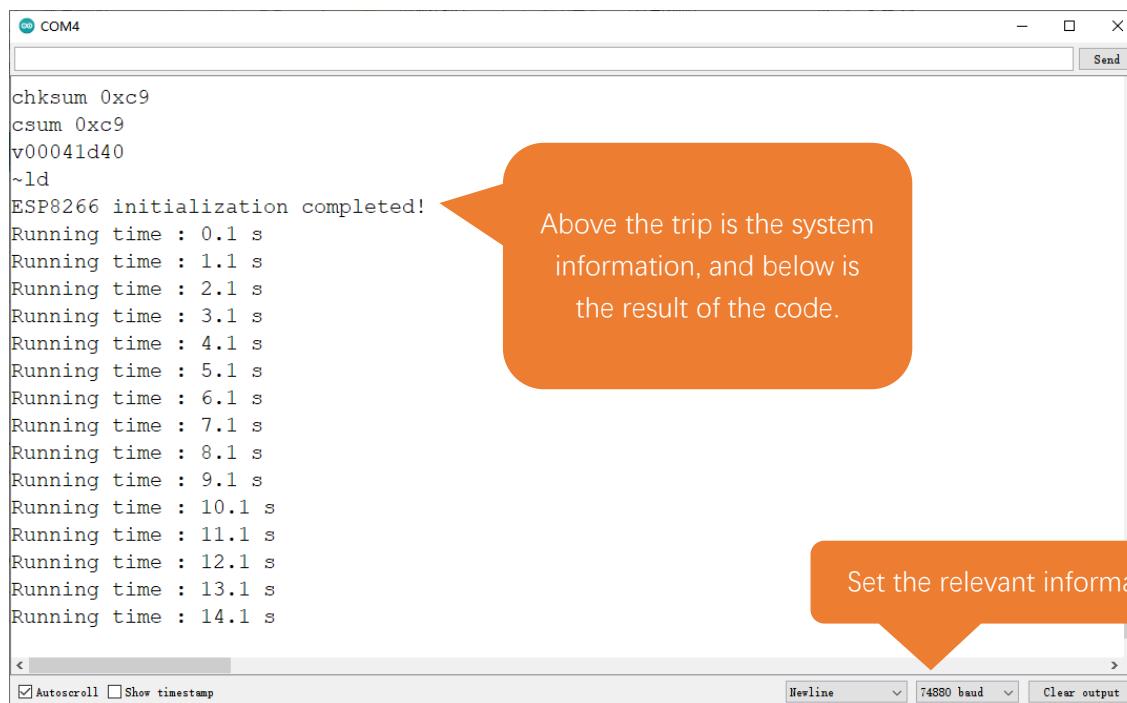
The screenshot shows the Arduino IDE interface with the following details:

- Title Bar:** Sketch_08.1_SerialPrinter | Arduino 1.8.19
- Menu Bar:** File Edit Sketch Tools Help
- Toolbar:** Standard icons for file operations.
- Code Editor:** The code for `Sketch_08.1_SerialPrinter` is displayed. It includes a header block with authorship information and two function blocks: `setup()` and `loop()`.

```
1 // ****
2   Filename      : SerialPrinter
3   Description   : Use UART send some data to PC, and show them on serial mon
4   Author        : www.freenove.com
5   Modification: 2022/05/10
6 ****
7
8 void setup() {
9     Serial.begin(74880);
10    Serial.println("ESP8266 initialization completed!");
11 }
12
13 void loop() {
14     Serial.printf("Running time : %.1f s\n", millis() / 1000.0f);
15     delay(1000);
16 }
```

- Status Bar:** Shows "Done uploading.", "Hash of data verified.", "Leaving...", and "Hard resetting via RTS pin...".
- Bottom Status:** "compatible), 32KB cache + 32KB IRAM (balanced), Use pgm_read macros for IRAM/PROGMEM, 4MB (FS:2MB OTA:~1019KB), 2, v2 Lower Memory, Disabled, None, Only Sketch, 115200 on COM4"

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

<code>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);</code>

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

<code>size_t println(arg);</code>

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

<code>size_t printf(const char * format, ...) __attribute__((format (printf, 2, 3)));</code>
--

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

<code>unsigned long millis();</code>

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



Project 8.2 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_08.2_SerialRW

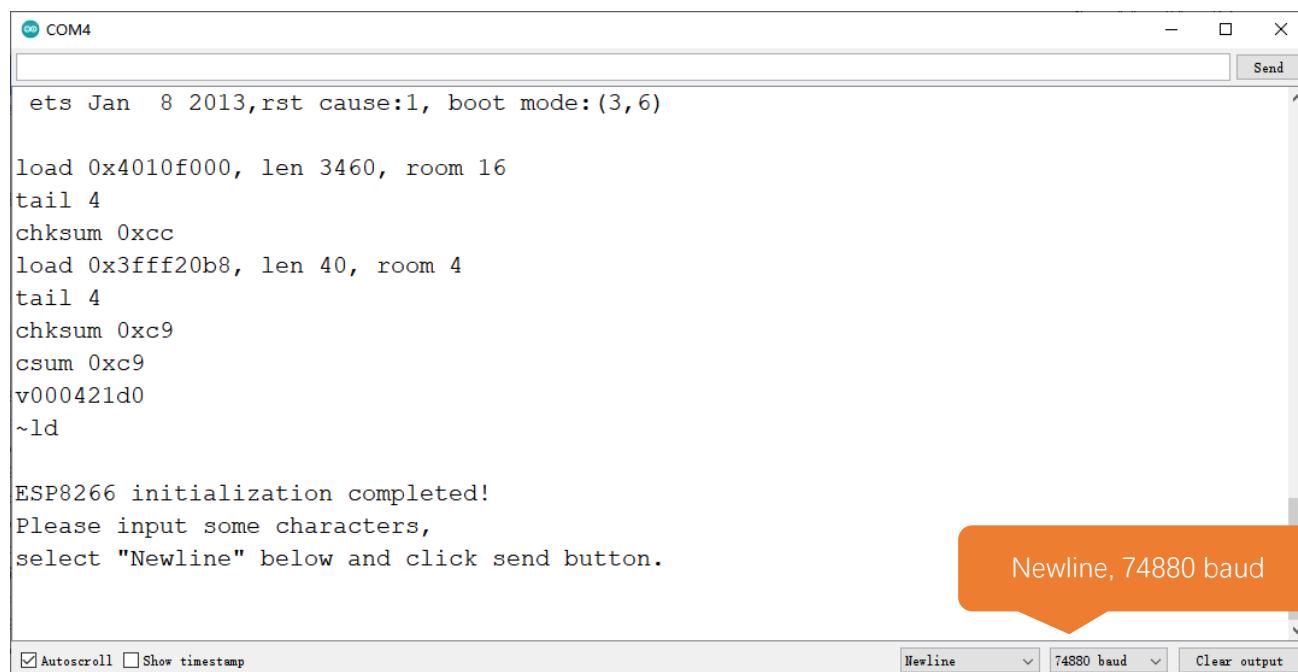
The screenshot shows the Arduino IDE interface. The top menu bar includes File, Edit, Sketch, Tools, and Help. Below the menu is a toolbar with icons for upload, download, and other functions. The main area displays the sketch code, and the bottom part shows the serial monitor output.

```
Sketch_08.2_SerialRW | Arduino 1.8.19
File Edit Sketch Tools Help
Sketch_08.2_SerialRW
8 bool stringComplete = false; // whether the string is complete
9
10 void setup() {
11     Serial.begin(74880);
12     Serial.println(String("\nESP8266 initialization completed!\n")
13                  + String("Please input some characters,\n")
14                  + String("select \"Newline\" below and click send button."));
15 }
16
17 void loop() {
18     if (Serial.available()) { // judge whether data has been received
19         char inChar = Serial.read(); // read one character
20         inputString += inChar;
21         if (inChar == '\n') {
22             stringComplete = true;
23         }
24     }
25     if (stringComplete) {
26         Serial.printf("inputString: %s \n", inputString);
27         inputString = "";
    }
}
Leaving...
Hard resetting via RTS pin...
```

The serial monitor output at the bottom of the IDE shows:

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

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



```
ets Jan  8 2013,rst cause:1, boot mode:(3,6)

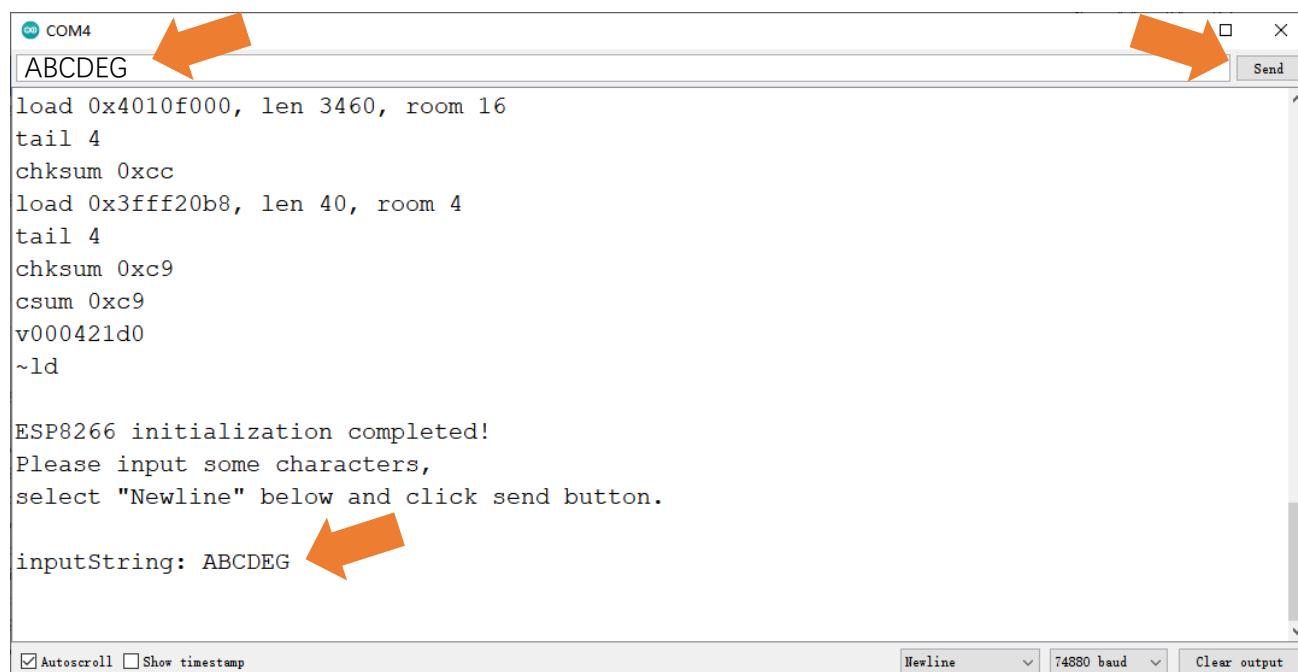
load 0x4010f000, len 3460, room 16
tail 4
chksum 0xcc
load 0x3fff20b8, len 40, room 4
tail 4
chksum 0xc9
csum 0xc9
v000421d0
~ld

ESP8266 initialization completed!
Please input some characters,
select "Newline" below and click send button.
```

Newline, 74880 baud

Autoscroll Show timestamp Newline 74880 baud Clear output

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



```
ABCDEF
```

```
load 0x4010f000, len 3460, room 16
tail 4
chksum 0xcc
load 0x3fff20b8, len 40, room 4
tail 4
chksum 0xc9
csum 0xc9
v000421d0
~ld

ESP8266 initialization completed!
Please input some characters,
select "Newline" below and click send button.

inputString: ABCDEG
```

Autoscroll Show timestamp Newline 74880 baud Clear output

The following is the program code:

```

1  String inputString = "";      //a String to hold incoming data
2  bool stringComplete = false; // whether the string is complete
3
4  void setup() {
5      Serial.begin(74880);
6      Serial.println(String("\nESP8266 initialization completed! \n")
7                      + String("Please input some characters, \n")
8                      + String("select \"Newline\" below and click send button. \n"));
9  }
10
11 void loop() {
12     if (Serial.available()) {      // judge whether data has been received
13         char inChar = Serial.read(); // read one character
14         inputString += inChar;
15         if (inChar == '\n') {
16             stringComplete = true;
17         }
18     }
19     if (stringComplete) {
20         Serial.printf("inputString: %s \n", inputString);
21         inputString = "";
22         stringComplete = false;
23     }
24 }
```

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 9 ADC Converter

We have learned how to control the brightness of LED through PWM and understood that PWM is not the real analog before. In this chapter, we will learn how to read analog, convert it into digital. That is, ADC.

Project 9.1 Read the Voltage of Potentiometer

In this project, ADC is used to convert analog signals into digital signals. Control chip on the control board has integrated this function. Now let us try to use this function to convert analog signals into digital signals.

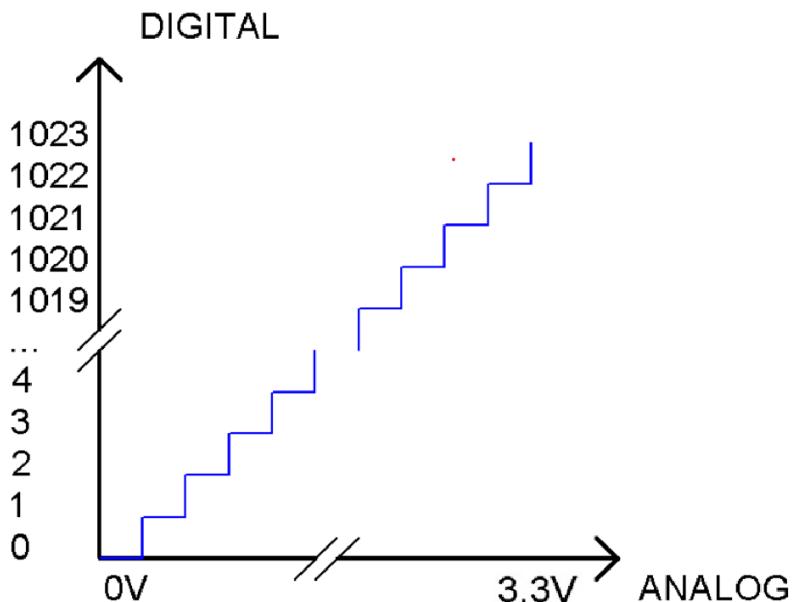
Component List

ESP8266 x1	USB cable
Breadboard x1	
Rotary potentiometer x1	Jumper wire M/M x3

Related knowledge

ADC

An ADC is an electronic integrated circuit used to convert analog signals such as voltages to digital or binary form consisting of 1s and 0s. The range of our ADC on ESP8266 is 10 bits, that means the resolution is $2^{10}=1024$, and it represents a range (at 3.3V) will be divided equally to 1024 parts. The range of analog values corresponds to ADC values. So the more bits the ADC has, the denser the partition of analog will be and the greater the precision of the resulting conversion.



Subsection 1: the analog in rang of 0V---3.3/1023 V corresponds to digital 0;

Subsection 2: the analog in rang of 3.3/1023 V---2*3.3 /1023V corresponds to digital 1;

...

The following analog will be divided accordingly.

The conversion formula is as follows:

$$ADC\ Value = \frac{\text{Analog\ Voltage}}{3.3} * 1023$$

ADC on ESP8266

ESP8266 has one digital analog converters with successive approximations of 10-bit accuracy, and a total of 1 pins can be used to measure analog signals. GPIO analog pin definition are shown in the following table. Note that the input voltage on the ADC pins of the ESP8266 module must be between 0V and 1.0V. For the ESP8266 ESP8266 development board designed by us, its input voltage range has been sampled by resistors. The ADC input voltage of the development board is 0V to 3.3V.

Do not exceed this voltage range when you use the ADC function. Exceeding this voltage range can cause permanent damage to your hardware!

Pin number in ESP8266

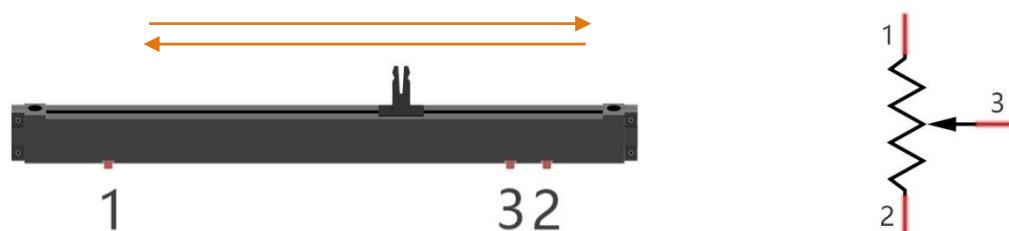
A0

The analog pin number is also defined in ESP8266's code base. For example, you can use A0 in your code.

Component knowledge

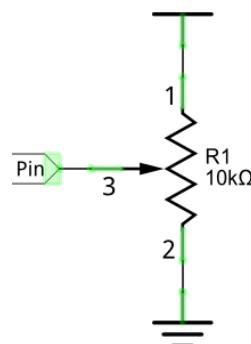
Potentiometer

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



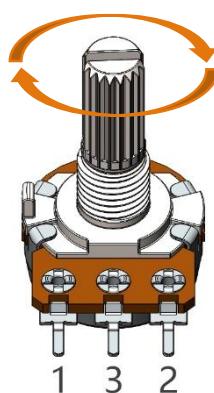
What between potentiometer pin 1 and pin 2 is the resistor body, and pins 3 is connected to brush. When brush moves from pin 1 to pin 2, the resistance between pin 1 and pin 3 will increase up to body resistance linearly, and the resistance between pin 2 and pin 3 will decrease down to 0 linearly.

In the circuit. The both sides of resistance body are often connected to the positive and negative electrode of the power. When you slide the brush pin 3, you can get a certain voltage in the range of the power supply.



Rotary potentiometer

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





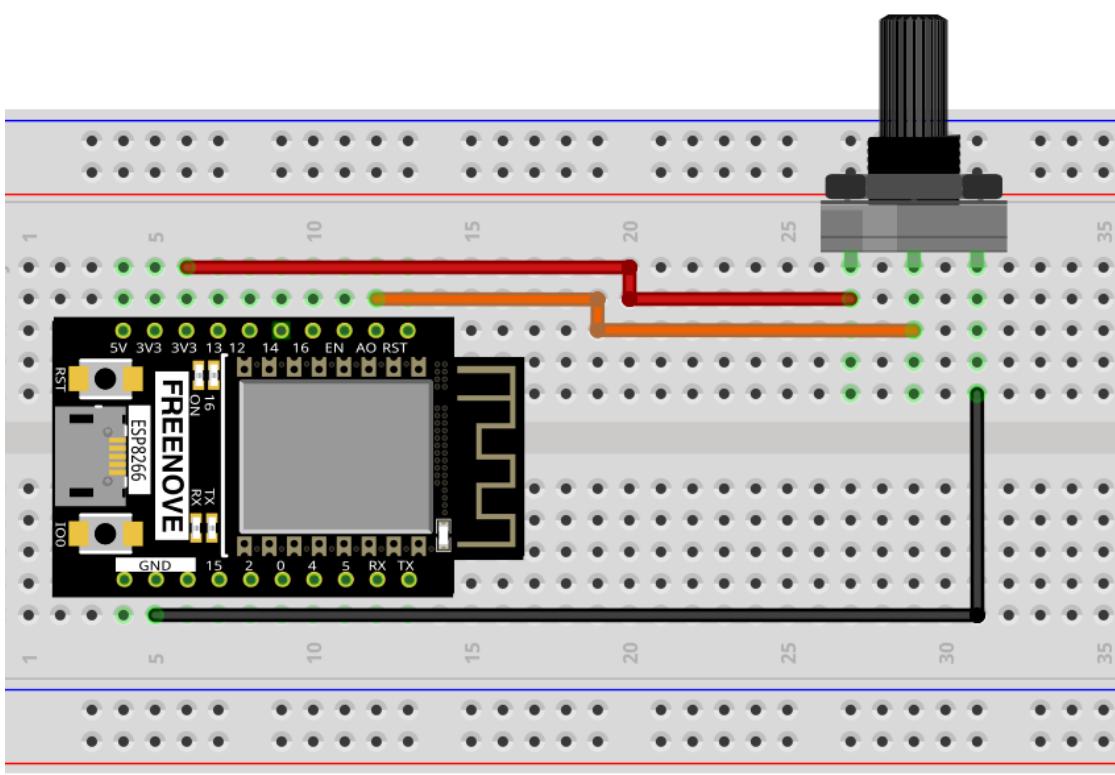
Circuit

Use pin A0 on the control board to detect the voltage of rotary potentiometer.

Schematic diagram

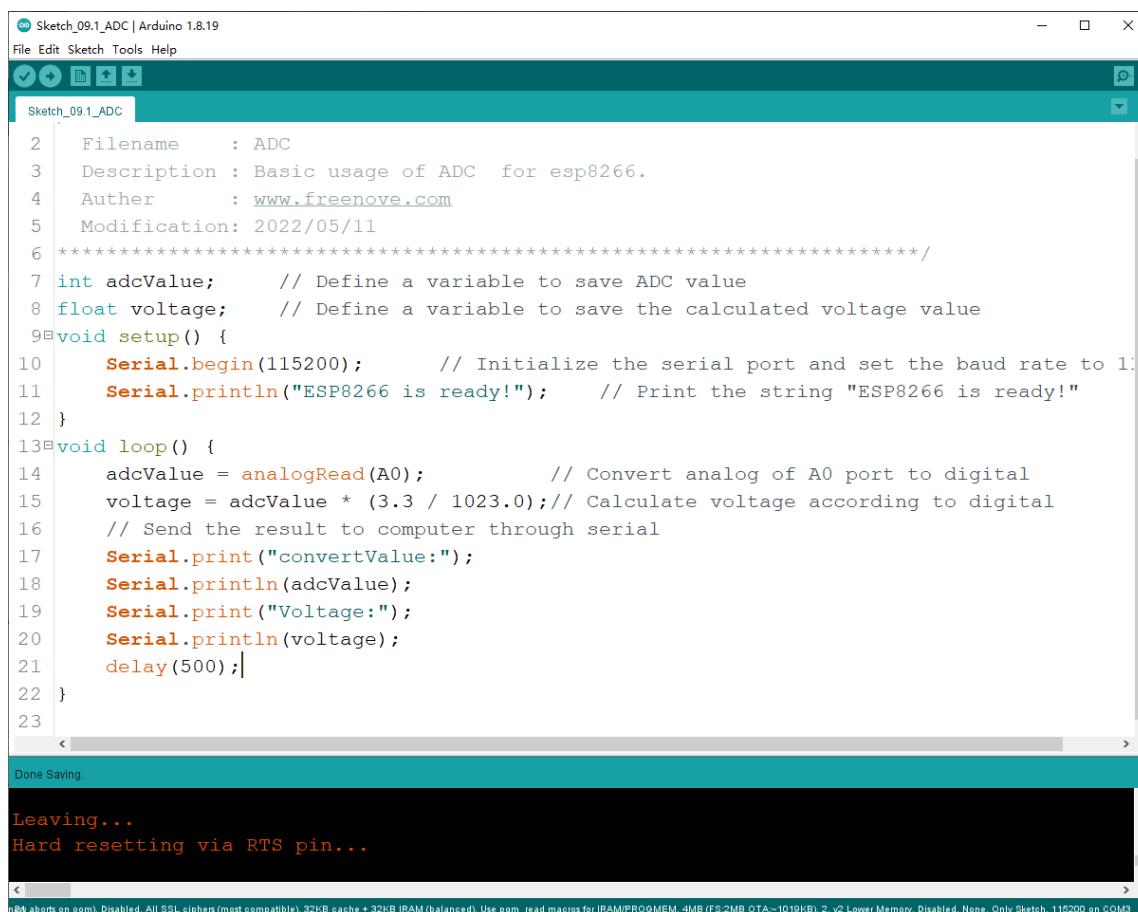


Hardware connection. If you need any support, please feel free to contact us via: support@freenove.com



Sketch

Sketch_09.1_ADC



```

2  Filename      : ADC
3  Description   : Basic usage of ADC  for esp8266.
4  Author        : www.freenove.com
5  Modification: 2022/05/11
6 ****
7 int adcValue;      // Define a variable to save ADC value
8 float voltage;    // Define a variable to save the calculated voltage value
9 void setup() {
10   Serial.begin(115200);      // Initialize the serial port and set the baud rate to 115200
11   Serial.println("ESP8266 is ready!"); // Print the string "ESP8266 is ready!"
12 }
13 void loop() {
14   adcValue = analogRead(A0);      // Convert analog of A0 port to digital
15   voltage = adcValue * (3.3 / 1023.0); // Calculate voltage according to digital
16   // Send the result to computer through serial
17   Serial.print("convertValue:");
18   Serial.println(adcValue);
19   Serial.print("Voltage:");
20   Serial.println(voltage);
21   delay(500);
22 }
23

```

Done Saving.

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

hwN aborts on com), Disabled, All SSL ciphers (most compatible), 32KB cache + 32KB IRAM (balanced). Use pgm_read macros for IRAM/PROGMEM, 4MB (FS:2MB OTA:~1019KB), 2, v2 Lower Memory, Disabled, None, Only Sketch, 115200 on COM3

Download the code to ESP8266, open the serial monitor, and set the baud rate to 115200. As shown in the following figure:





From the code, we get the ADC value of pin A0, then convert it into voltage and sent to the serial port. Verify and upload the code, open the Serial Monitor, and then you will see the original ADC value and converted voltage sent from control board.

Turn the rotary potentiometer shaft, and you can see the voltage change.

The following is the code:

```
1 int adcValue;      // Define a variable to save ADC value
2 float voltage;    // Define a variable to save the calculated voltage value
3 void setup() {
4     Serial.begin(115200);      // Initialize the serial port and set the baud rate to 115200
5     Serial.println("ESP8266 is ready!");    // Print the string "ESP8266 is ready!"
6 }
7 void loop() {
8     adcValue = analogRead(A0);          // Convert analog of A0 port to digital
9     voltage = adcValue * (3.3 / 1023.0); // Calculate voltage according to digital
10    // Send the result to computer through serial
11    Serial.print("convertValue:");
12    Serial.println(adcValue);
13    Serial.print("Voltage:");
14    Serial.println(voltage);
15    delay(500);
16 }
```

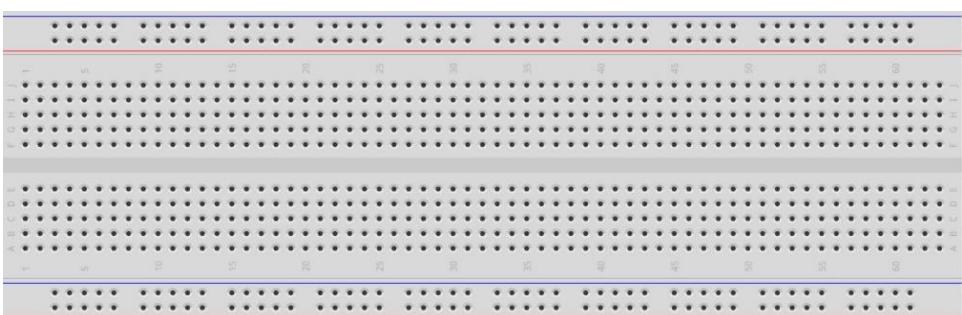
Chapter 10 Potentiometer & LED

In the previous section, we have finished reading ADC value and converting it into voltage. Now, we will try to use potentiometer to control the brightness of LED.

Project 10.1 Soft Light

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

Component List

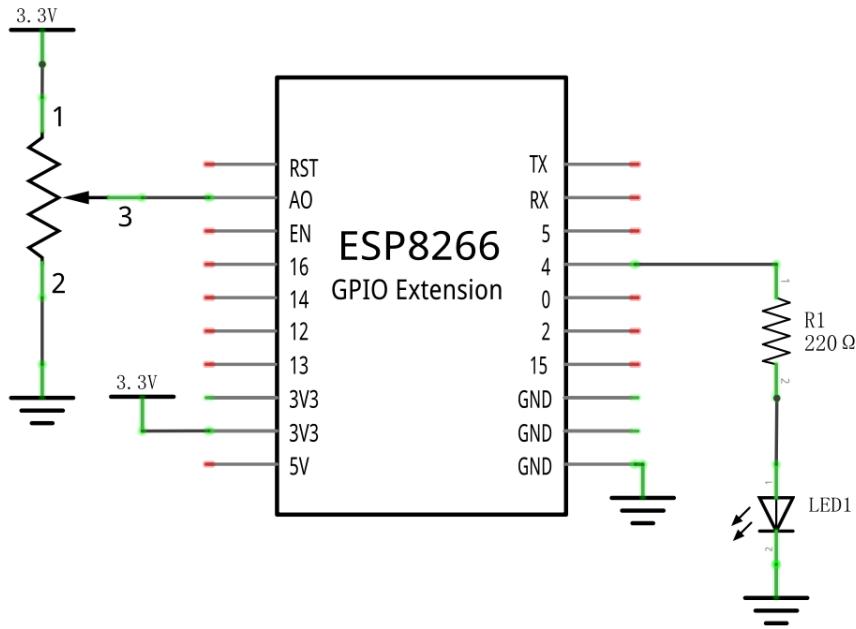
ESP8266 x1	USB cable
	
Breadboard x1	
	
Rotary potentiometer x1	Resistor 220Ω x1
	
LED x1	Jumper wire M/M x8
	



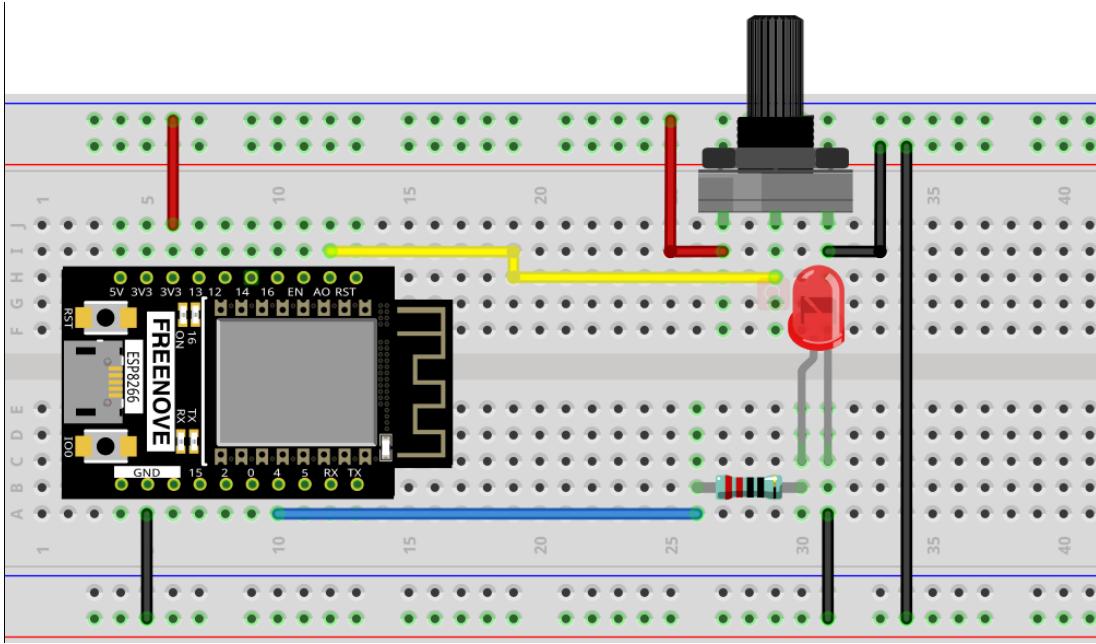
Circuit

Use pin A0 on control board to detect the voltage of rotary potentiometer, and use pin 4 to control one LED.

Schematic diagram



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



Sketch

Sketch_10.1_Softlight



```

Sketch_10.1_SoftLight | Arduino 1.8.19
File Edit Sketch Tools Help
Sketch_10.1_SoftLight
1 // ****
2   Filename      : SoftLight
3   Description   : Controlling the brightness of LED by potentiometer.
4   Author        : www.freenove.com
5   Modification: 2022/05/11
6 ****
7   int adcValue;      // Define a variable to save the ADC value
8   int ledPin = 4;    // Use D9 on Freenove UNO to control the LED
9
10 void setup() {
11     pinMode(ledPin, OUTPUT);           // Initialize the LED pin as an output
12 }
13 void loop() {
14     adcValue = analogRead(A0);         // Convert the analog of A0 port to digital
15     // Map analog to the 0-255 range, and works as PWM duty cycle of ledPin port
16     analogWrite(ledPin, map(adcValue, 0, 1024, 0, 255));
17 }
18

Done Saving.

```

Done Saving.

nde aborts on com), Disabled, All SSL ciphers (most compatible), 32KB cache + 32KB IRAM (balanced), Use pgm_read macros for IRAM/PROGMEM, 4MB (FS:2MB OTA:~1019KB), 2, v2 Lower Memory, Disabled, None, Only Sketch, 115200 on COM3

Download the code to ESP8266, by turning the adjustable resistor to change the input voltage of GPIO4, ESP8266 changes the output voltage of GPIO4 according to this voltage value, thus changing the brightness of the LED.

The following is the code:

```

1   int adcValue;      // Define a variable to save the ADC value
2   int ledPin = 4;
3   void setup() {
4       pinMode(ledPin, OUTPUT);           // Initialize the LED pin as an output
5   }
6   void loop() {
7       adcValue = analogRead(A0);         // Convert the analog of A0 port to digital
8       // Map analog to the 0-255 range, and works as PWM duty cycle of ledPin port
9       analogWrite(ledPin, map(adcValue, 0, 1024, 0, 255));
10  }

```

In the code, we get the ADC value of pin A0 and map it to PWM duty cycle of LED pin port. According to

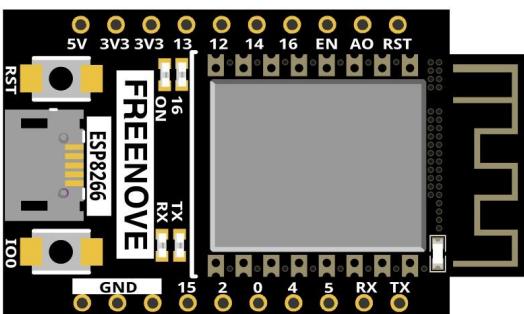
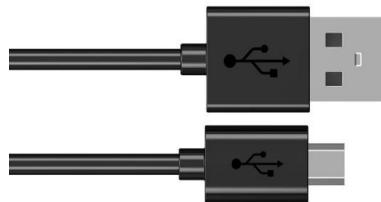
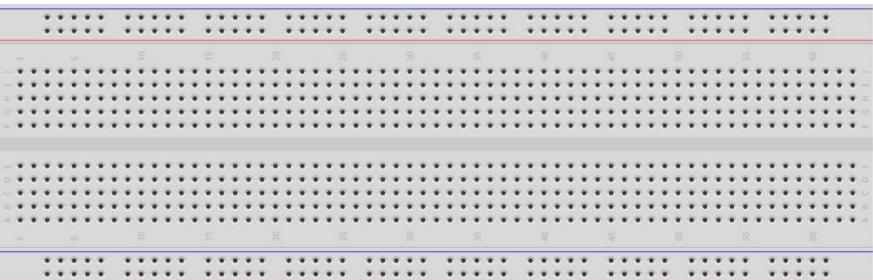
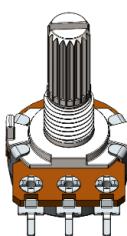
Any concerns? ✉ support@freenove.com

different LED brightness, we can see the changes of voltage easily.

Project 10.2 Color Light

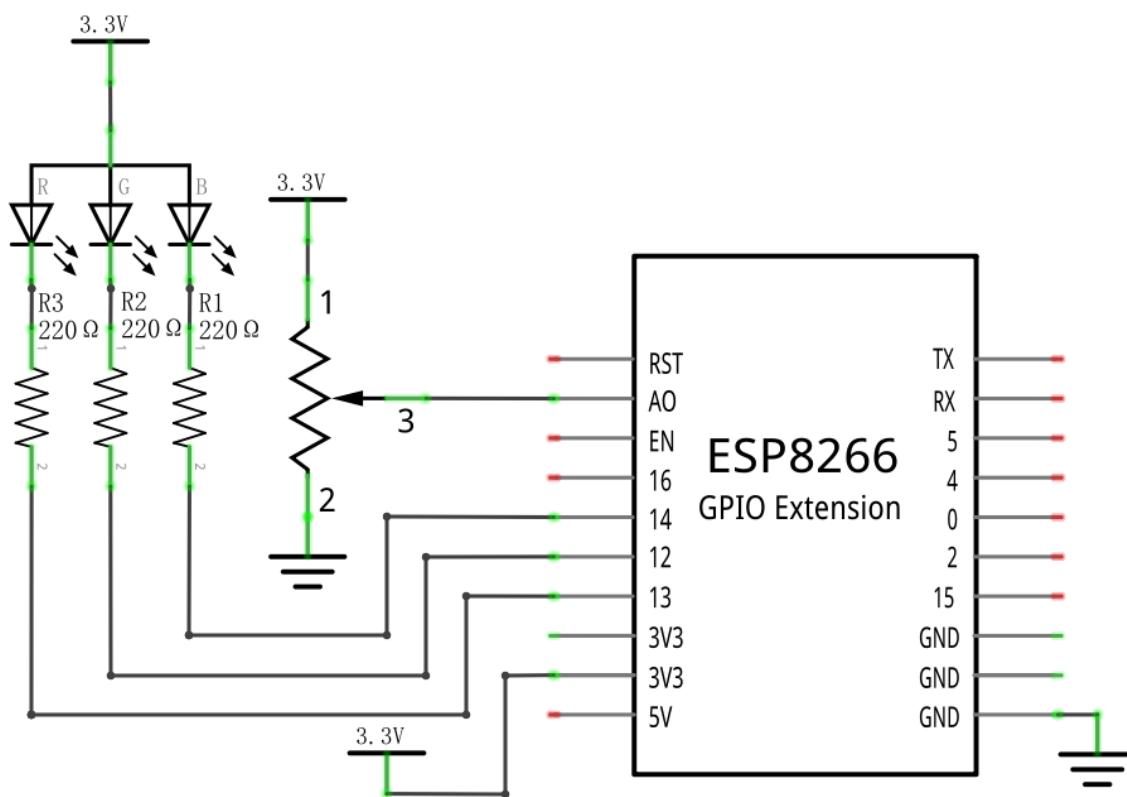
In this project, a potentiometer is used to control the RGB LED. The RGB LED is bright red when the potentiometer is near the midpoint, green when the potentiometer rotates to the "left" and blue when the potentiometer rotates to the "right".

Component List

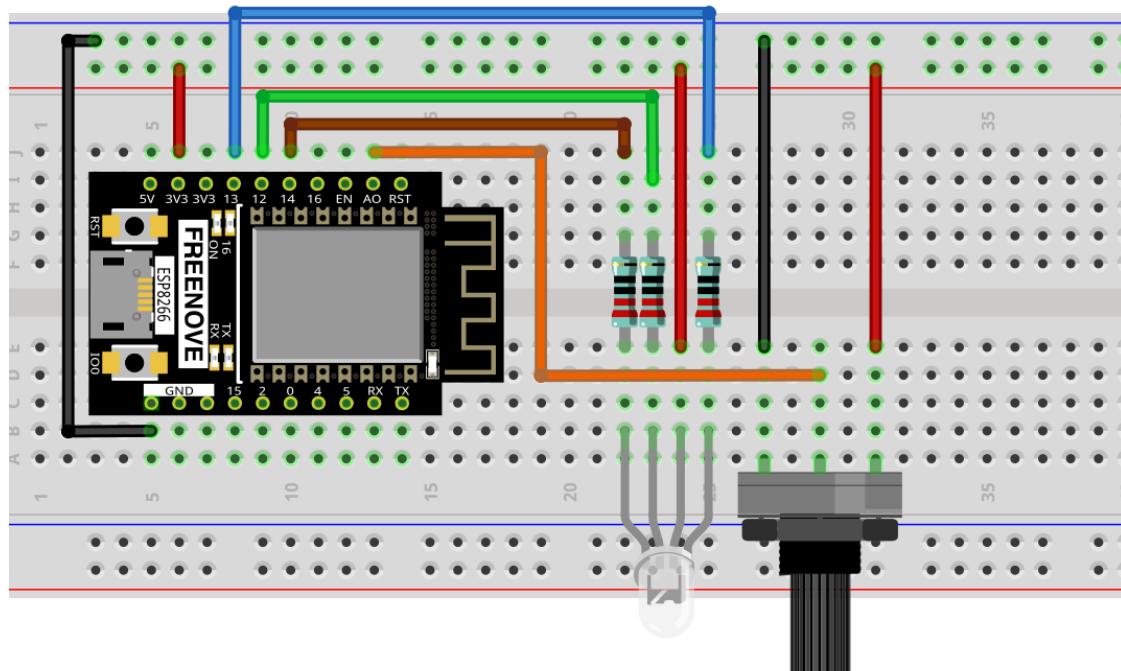
ESP8266 x1	USB cable		
			
Breadboard x1			
Rotary potentiometer x1	Resistor 220Ω x3	RGBLED x1	Jumper wire M/M x9
			

Circuit

Schematic diagram



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



Sketch

Sketch_10.2_ColorLight

The screenshot shows the Arduino IDE interface. The top menu bar includes File, Edit, Sketch, Tools, and Help. Below the menu is a toolbar with icons for upload, refresh, and other functions. The main window displays the code for 'Sketch_10.2_ColorLight'. The code uses analogRead(A0) to get a value from the A0 pin, which is then mapped to a PWM duty cycle for an RGB LED. It defines three color ranges: (255, 0, 255) for 0-345, (0, 255, 255) for 345-680, and (255, 255, 0) for 680-1024. Lines 13 and 14 show analogWrite calls for pins 1 and 2. Lines 16-18 define the setup() function to set the ledPin as an output. Lines 19-31 define the loop() function to read the analog value, map it, and set the color. The bottom status bar indicates the board is an 'esp8266', cache is 32KB, IRAM is balanced, and memory usage is 4MB (FS:2MB OTA:~1019KB), 2. v2 Lower Memory, Disabled, None, Only Sketch, 115200 on COM4.

```
Sketch_10.2_ColorLight | Arduino 1.8.19
File Edit Sketch Tools Help
Sketch_10.2_ColorLight
13     analogWrite(ledPins[1], g);
14     analogWrite(ledPins[2], b);
15 }
16 void setup() {
17     pinMode(ledPin, OUTPUT);          // Initialize the LED pin as an output
18 }
19 void loop() {
20     adcValue = analogRead(A0);        // Convert the analog of A0 port to digital
21     // Map analog to the 0-255 range, and works as PWM duty cycle of ledPin port
22     if(adcValue>=0&&adcValue<=345){
23         setColor(255, 0, 255);
24     }
25     else if(adcValue>345&&adcValue<680){
26         setColor(0, 255, 255);
27     }
28     else if(adcValue>=680&&adcValue<=1024){
29         setColor(255, 255, 0);
30     }
31 }

Done uploading.

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

s@compatible), 32KB cache + 32KB IRAM (balanced). Use pgm_read_macros for IRAM/PROGMEM, 4MB (FS:2MB OTA:~1019KB), 2, v2 Lower Memory, Disabled, None, Only Sketch, 115200 on COM4
```

Download the code to ESP8266, rotate the potentiometers, then the color of RGB LED will change.

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

The following is the program code:

```
1 int adcValue;      // Define a variable to save the ADC value
2 int ledPin = 4;
3 const byte ledPins[] = {13, 14, 12};    //define red, green, blue led pins
4 int red, green, blue;
5 void setColor(byte r, byte g, byte b) {
6     analogWrite(ledPins[0], r); //Common anode LED, low level to turn on the led.
7     analogWrite(ledPins[1], g);
8     analogWrite(ledPins[2], b);
9 }
10 void setup() {
11     pinMode(ledPin, OUTPUT);           // Initialize the LED pin as an output
12 }
13 void loop() {
14     adcValue = analogRead(A0);        // Convert the analog of A0 port to digital
15     // Map analog to the 0–255 range, and works as PWM duty cycle of ledPin port
16     if(adcValue>=0&&adcValue<=345) {
17         setColor(255, 0, 255);
18     }
19     else if(adcValue>345&&adcValue<680) {
20         setColor(0, 255, 255);
21     }
22     else if(adcValue>=680&&adcValue<=1024) {
23         setColor(255, 255, 0);
24     }
25 }
```

In the code, you can read the potentiometer ADC value, judge the range of ADC value, to control the RGB LED color.



Project 10.3 Soft Rainbow Light

In this project, we use potentiometer to control Freenove 8 RGB LED Module.

Component List

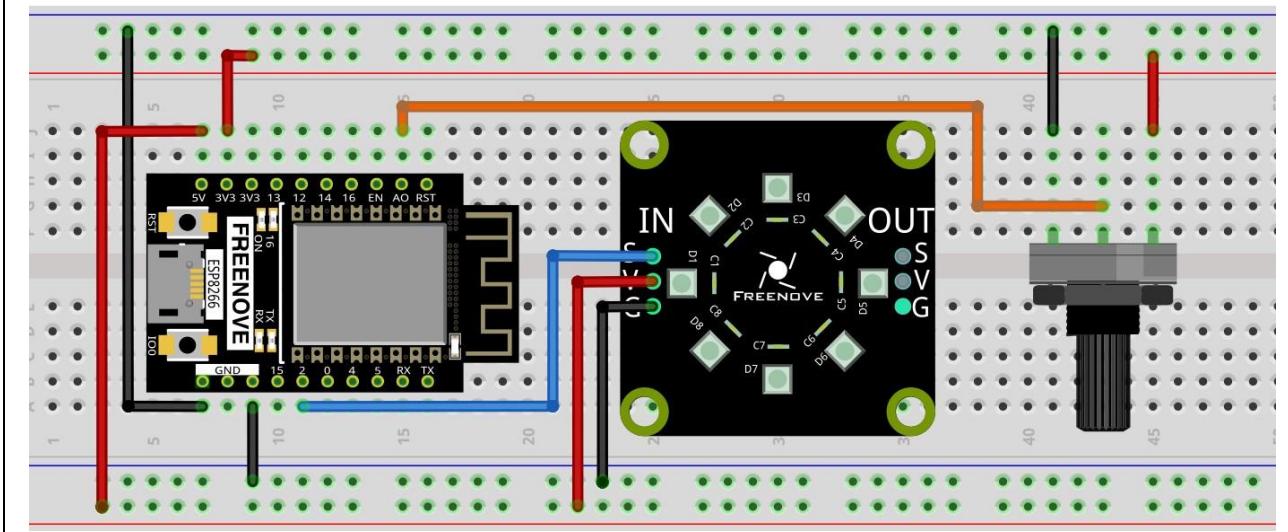
ESP8266 x1	USB cable
Breadboard x1	
Freenove 8 RGB LED Module x1	Rotary potentiometer x1 Jumper wire F/M x3 Jumper wire M/M x7

Circuit

Schematic diagram



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





Sketch

Sketch_10.3_Soft_Rainbow_Light

```

Sketch_10.3_SoftRainbowLight | Arduino 1.8.19
File Edit Sketch Tools Help
Sketch_10.3_SoftRainbowLight
5 #define LEDS_PIN      2 // define the pin connected to the Freenove 8 led strip
6 #define BRIGHTNESS    50 // brightness, the value range is 0-255.
7 Adafruit_NeoPixel pixels(LEDS_COUNT, LEDS_PIN, NEO_GRB + NEO_KHZ800);
8 void setup() {
9     pixels.begin(); // INITIALIZE NeoPixel strip object (REQUIRED)
10    pixels.setBrightness(BRIGHTNESS); // set the brightness.
11 }
12
13 void loop() {
14     int colorPos = map(analogRead(PIN_POT), 0, 1023, 0, 255);
15     for (int i = 0; i < LEDS_COUNT; i++) {
16         pixels.setPixelColor(i,Wheel(colorPos + i * 255 / 8)); // Set color data.
17     }
18     pixels.show(); // Send color data to LED, and display.
19     delay(10);
20 }
21
22 uint32_t Wheel(byte pos){
23     u32 WheelPos = pos % 0xff;
24     if (WheelPos < 85){
25         return ((255 - WheelPos * 3) << 16) | ((WheelPos * 3) << 8);
}

Done uploading.
Wrote 266400 bytes (196039 compressed) at 0x00000000 in 17.3 seconds (effective 123.3 kbit/s)
Hash of data verified.

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

```

Download the code to ESP8266, rotate the handle of the potentiometer, and the color of the lamp ring will change.



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

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

The following is the program code:

```
1 #include <Adafruit_NeoPixel.h>
2
3 #define PIN_POT      A0 // Define analog input pins
4 #define LEDS_COUNT   8 // The number of led
5 #define LEDS_PIN      2 // define the pin connected to the Freenove 8 led strip
6 #define BRIGHTNESS   50 // brightness, the value range is 0-255.
7 Adafruit_NeoPixel pixels(LEDS_COUNT, LEDS_PIN, NEO_GRB + NEO_KHZ800);
8 void setup() {
9     pixels.begin(); // INITIALIZE NeoPixel strip object (REQUIRED)
10    pixels.setBrightness(BRIGHTNESS); // set the brightness.
11 }
12
13 void loop() {
14     int colorPos = map(analogRead(PIN_POT), 0, 1023, 0, 255);
15     for (int i = 0; i < LEDS_COUNT; i++) {
16         pixels.setPixelColor(i, Wheel(colorPos + i * 255 / 8)); // Set color data.
17     }
18     pixels.show(); // Send color data to LED, and display.
19     delay(10);
20 }
21
22 uint32_t Wheel(byte pos) {
23     u32 WheelPos = pos % 0xff;
24     if (WheelPos < 85) {
25         return ((255 - WheelPos * 3) << 16) | ((WheelPos * 3) << 8);
26     }
27     if (WheelPos < 170) {
28         WheelPos -= 85;
29         return (((255 - WheelPos * 3) << 8) | (WheelPos * 3));
30     }
31     WheelPos -= 170;
32     return ((WheelPos * 3) << 16 | (255 - WheelPos * 3));
33 }
```

The overall logical structure of the code is the same as the previous project rainbow light, except that the starting point of the color in this code is controlled by potentiometer.

Chapter 11 Photoresistor & LED

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

Project 11.1 NightLamp

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

Component List

ESP8266 x1		USB cable
Breadboard x1		
Photoresistor x1	Resistor 220Ω x1 10KΩ x1	LED x1 Jumper wire M/M x7

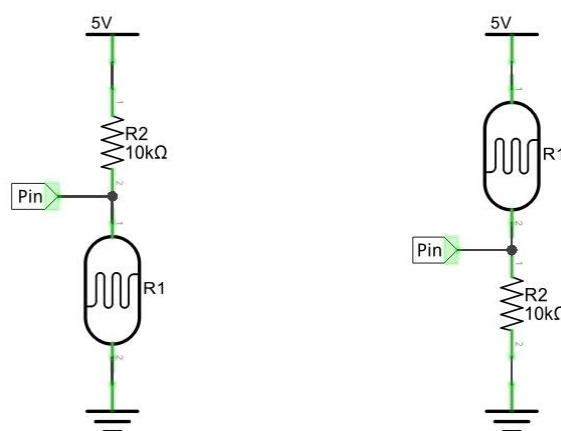
Component knowledge

Photoresistor

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



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



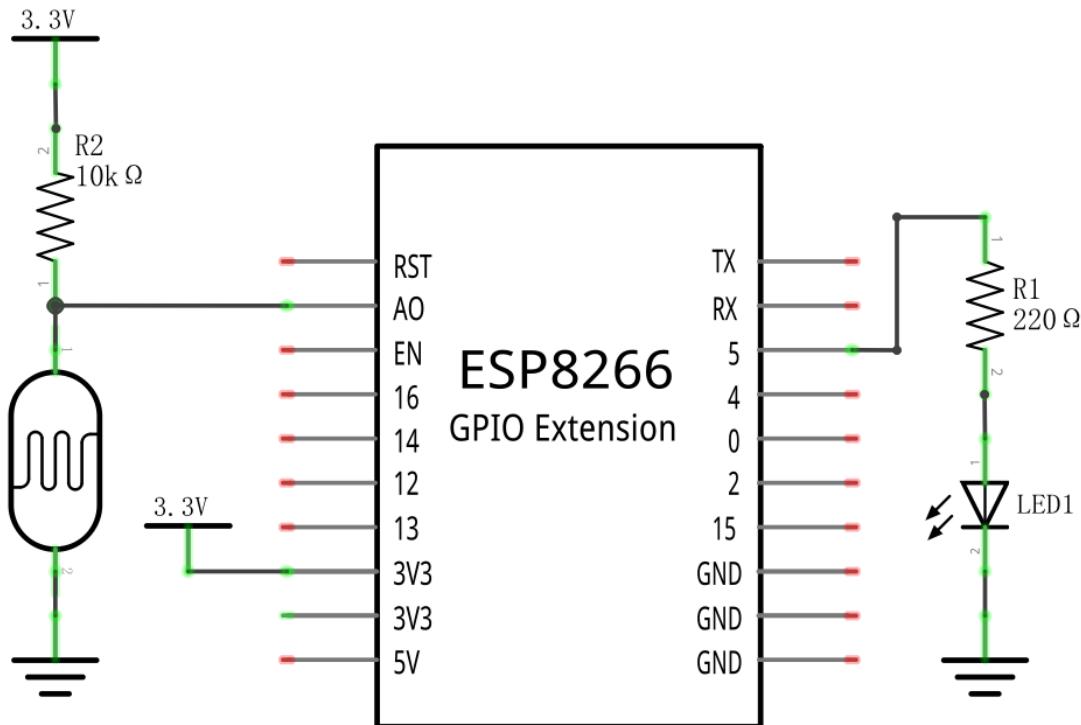
In the above circuit, when a photoresistor's resistance value changes due to a change in light intensity, the voltage between the photoresistor and resistor R1 will also change. Therefore, the intensity of the light can be obtained by measuring this voltage.



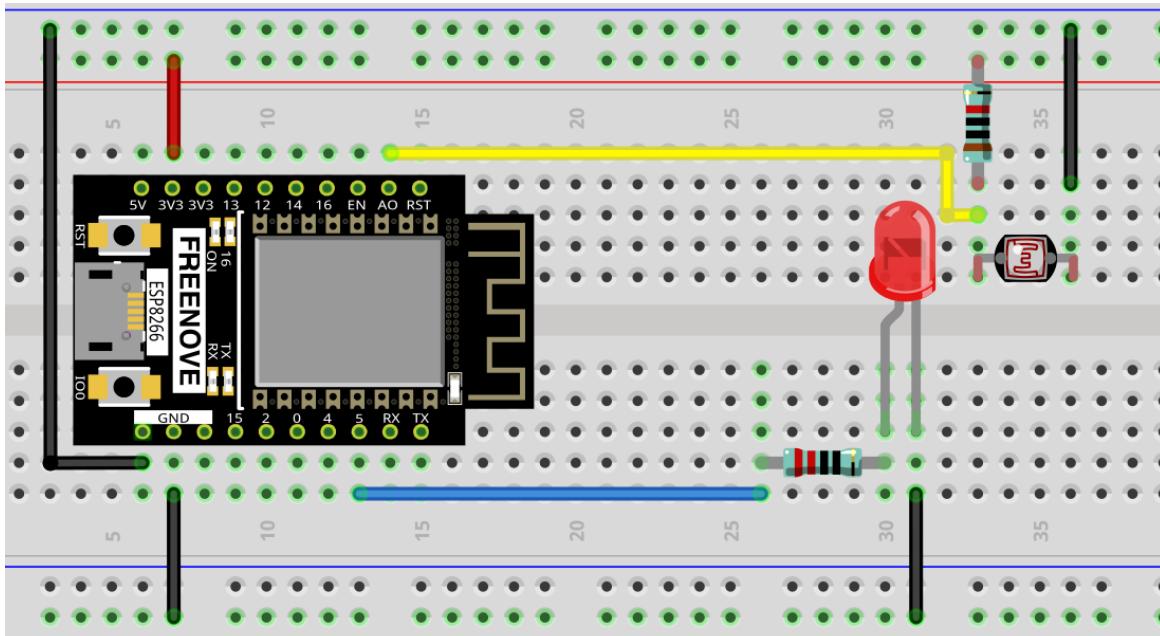
Circuit

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

Schematic diagram



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



Sketch

The circuit used is similar to the project Soft Light. The only difference is that the input signal of the A0 pin of ADC changes from a potentiometer to a combination of a photoresistor and a resistor.

Sketch_11.1_Nightlamp

The screenshot shows the Arduino IDE interface. The top menu bar includes File, Edit, Sketch, Tools, and Help. Below the menu is a toolbar with icons for upload, refresh, and other functions. The main window displays the code for 'Sketch_11.1_NightLamp' with the following content:

```
Sketch_11.1_NightLamp | Arduino 1.8.19
File Edit Sketch Tools Help
Sketch_11.1_NightLamp
1 // *****
2   Filename      : NightLamp
3   Description   : Controlling the brightness of LED by photoresistor.
4   Author        : www.freenove.com
5   Modification  : 2022/05/11
6 *****/
7 #define PIN_ANALOG_IN    A0
8 #define PIN_LED          5
9 #define LIGHT_MIN         301
10 #define LIGHT_MAX         924
11 void setup() {
12     pinMode(PIN_LED, OUTPUT);
13 }
14
15 void loop() {
16     int adcVal = analogRead(PIN_ANALOG_IN); //read adc
17     int pwmVal = map(constrain(adcVal, LIGHT_MIN, LIGHT_MAX), LIGHT_MIN, LIGHT_MAX, 0,
18     analogWrite(PIN_LED, pwmVal); // set the pulse width.
19     delay(10);
20 }
```

Below the code editor, a message says "Done uploading." followed by "Leaving... Hard resetting via RTS pin...". At the bottom, the status bar indicates "blit, 32KB cache + 32KB IRAM (balanced), Use pgm_read macros for IRAM/PROGMEM, 4MB (FS:2MB OTA:~1019KB), 2, v2 Lower Memory, Disabled, None, Only Sketch, 115200 on COM4".

Download the code to ESP8266, if you cover the photoresistor or increase the light shining on it, the brightness of the LED changes accordingly.

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

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



The following is the program code:

```
1 #define PIN_ANALOG_IN A0
2 #define PIN_LED 5
3 #define LIGHT_MIN 301
4 #define LIGHT_MAX 924
5 void setup() {
6     pinMode(PIN_LED, OUTPUT);
7 }
8
9 void loop() {
10    int adcVal = analogRead(PIN_ANALOG_IN); //read adc
11    int pwmVal = map(constrain(adcVal, LIGHT_MIN, LIGHT_MAX), LIGHT_MIN, LIGHT_MAX, 0, 1023);
12 // adcVal re-map to pwmVal
13    analogWrite(PIN_LED, pwmVal); // set the pulse width.
14    delay(10);
15 }
```

Reference

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

Constrain the value amt between low and high.

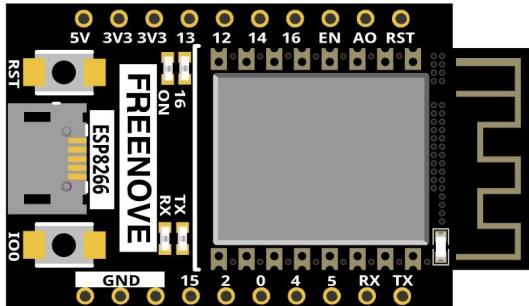
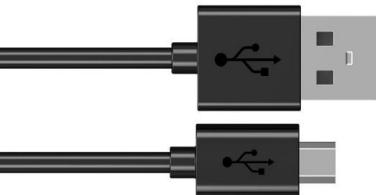
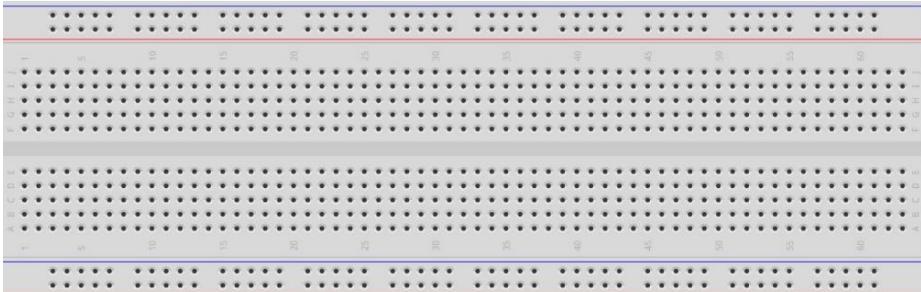
Chapter 12 Thermistor

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

Project 12.1 Thermometer

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

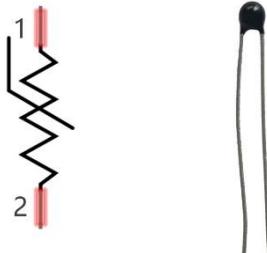
Component List

ESP8266 x1	USB cable		
			
Breadboard x1			
	Thermistor x1	Resistor 10kΩ x1	Jumper wire M/M x4

Component knowledge

Thermistor

A thermistor is a temperature sensitive resistor. When it senses a change in temperature, the resistance of the thermistor will change. We can take advantage of this characteristic by using a thermistor to detect temperature intensity. A thermistor and its electronic symbol are shown below.



The relationship between resistance value and temperature of a thermistor is:

$$R_t = R * \text{EXP} \left[B * \left(\frac{1}{T_2} - \frac{1}{T_1} \right) \right]$$

Where:

Rt is the thermistor resistance under T2 temperature;

R is the nominal resistance of thermistor under T1 temperature;

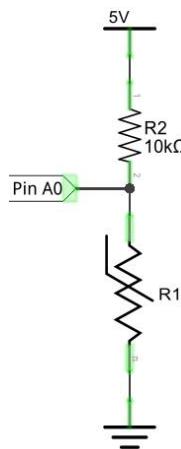
EXP[n] is nth power of E;

B is for thermal index;

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

For the parameters of the thermistor, we use: B=3950, R=10k, T1=25.

The circuit connection method of the thermistor is similar to photoresistor, as the following:



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

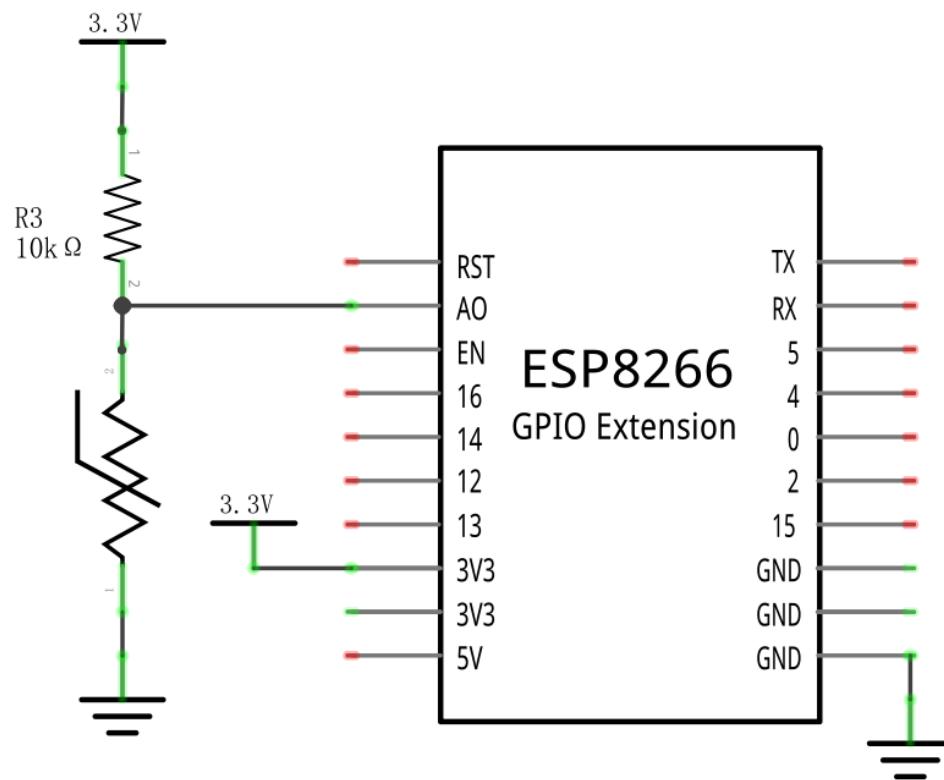
Therefore, the temperature formula can be derived as:

$$T_2 = 1 / \left(\frac{1}{T_1} + \ln \left(\frac{R_t}{R} \right) / B \right)$$

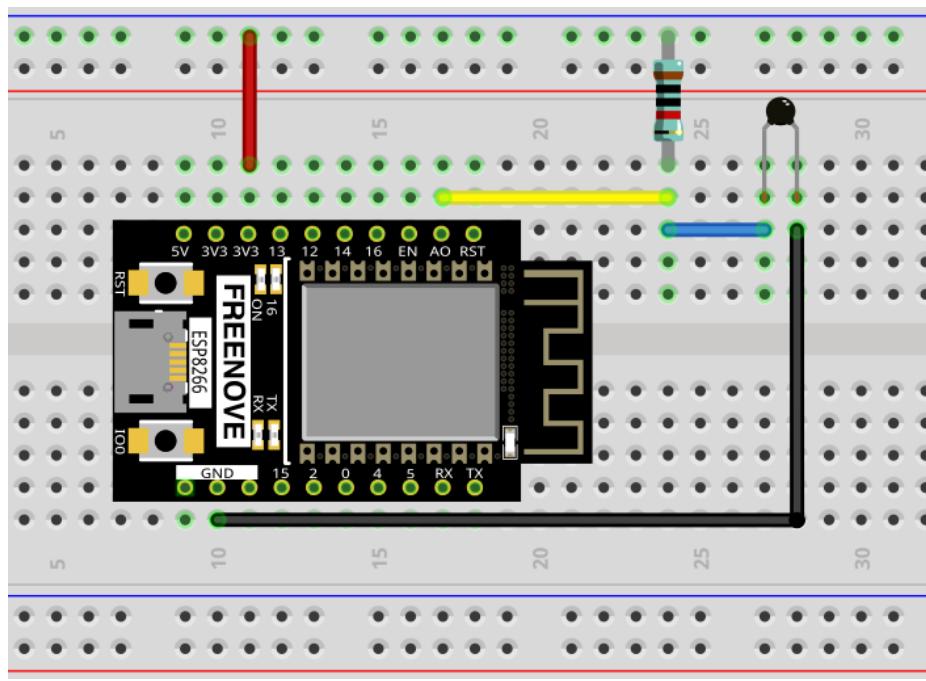
Circuit

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

Schematic diagram



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



Any concerns? ✉ support@freenove.com

Sketch

Sketch_12.1_Thermometer

The screenshot shows the Arduino IDE interface. The top menu bar includes File, Edit, Sketch, Tools, Help, and a Freenove logo. The central code editor window displays the following C++ code for a thermometer project:

```

Sketch_12.1_Thermometer | Arduino 1.8.19
File Edit Sketch Tools Help
Sketch_12.1_Thermometer
1 // ****
2   Filename      : Thermometer
3   Description   : Making a thermometer by thermistor.
4   Author        : www.freenove.com
5   Modification: 2022/05/11
6 ****
7 #define PIN_ANALOG_IN    A0
8 void setup() {
9   Serial.begin(115200);
10 }
11
12 void loop() {
13   int adcValue = analogRead(PIN_ANALOG_IN);           //read ADC pin
14   double voltage = (float)adcValue / 1023.0 * 3.3;      // calculate voltage
15   double Rt = 10 * voltage / (3.3 - voltage);         //calculate resistance
16   double tempK = 1 / (1 / (273.15 + 25) + log(Rt / 10) / 3950.0); //calculate temperature
17   double tempC = tempK - 273.15;                      //calculate temperature
18   Serial.printf("ADC value : %d,\tVoltage : %.2fV, \tTemperature : %.2fC\n", adcValue,
19   delay(1000);
20 }

```

The terminal window at the bottom shows the upload process:

```

Done uploading.
wrote 271104 bytes (199059 compressed) at 0x00000000 in 17.5 seconds (effective 125.7 kbit/s)
Hash of data verified.

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

```

A small note at the bottom of the terminal window states: nd@aborts on com), Disabled, All SSL ciphers (most compatible), 32KB cache + 32KB IRAM (balanced). Use pgm_read macros for IRAM/PROGMEM, 4MB (FS:2MB OTA:~1019KB), 2, v2 Lower Memory, Disabled, None, Only Sketch, 115200 on COM3

Download the code to ESP8266, the terminal window will display the current ADC value, voltage value and temperature value. Try to “pinch” the thermistor (without touching the leads) with your index finger and thumb for a brief time, you should see that the temperature value increases.

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

```
COM3
ADC value : 497, Voltage : 1.60V, Temperature : 26.28C
ADC value : 499, Voltage : 1.61V, Temperature : 26.10C
ADC value : 502, Voltage : 1.62V, Temperature : 25.84C
ADC value : 506, Voltage : 1.63V, Temperature : 25.48C
ADC value : 507, Voltage : 1.64V, Temperature : 25.40C
ADC value : 508, Voltage : 1.64V, Temperature : 25.31C
ADC value : 508, Voltage : 1.64V, Temperature : 25.31C
ADC value : 509, Voltage : 1.64V, Temperature : 25.22C
ADC value : 509, Voltage : 1.64V, Temperature : 25.22C
ADC value : 510, Voltage : 1.65V, Temperature : 25.13C
ADC value : 511, Voltage : 1.65V, Temperature : 25.04C
ADC value : 511, Voltage : 1.65V, Temperature : 25.04C
ADC value : 512, Voltage : 1.65V, Temperature : 24.96C
ADC value : 513, Voltage : 1.65V, Temperature : 24.87C
ADC value : 514, Voltage : 1.66V, Temperature : 24.78C
ADC value : 514, Voltage : 1.66V, Temperature : 24.78C
ADC value : 514, Voltage : 1.66V, Temperature : 24.78C
ADC value : 516, Voltage : 1.66V, Temperature : 24.60C
ADC value : 516, Voltage : 1.66V, Temperature : 24.60C
ADC value : 517, Voltage : 1.67V, Temperature : 24.52C
ADC value : 517, Voltage : 1.67V, Temperature : 24.52C

 Autoscroll  Show timestamp  Newline  115200 baud  Clear output
```

The following is the code:

```
1 #define PIN_ANALOG_IN A0
2 void setup() {
3     Serial.begin(115200);
4 }
5
6 void loop() {
7     int adcValue = analogRead(PIN_ANALOG_IN);           //read ADC pin
8     double voltage = (float)adcValue / 1023.0 * 3.3;    // calculate voltage
9     double Rt = 10 * voltage / (3.3 - voltage);        //calculate resistance value of thermistor
10    double tempK = 1 / (1/(273.15 + 25) + log(Rt / 10)/3950.0); //calculate temperature (Kelvin)
11    double tempC = tempK - 273.15;                     //calculate temperature (Celsius)
12    Serial.printf("ADC value : %d, \tVoltage : %.2fV, \tTemperature : %.2fC\n", adcValue,
13    voltage, tempC);
14 }
```

In the code, the ADC value of ADC module A0 port is read, and then calculates the voltage and the resistance of thermistor according to Ohms Law. Finally, it calculates the temperature sensed by the thermistor, according to the formula.

Chapter 13 74HC595 & LED Bar Graph

We have used LED bar graph to make a flowing water light, in which 10 GPIO ports of ESP8266 is occupied. More GPIO ports mean that more peripherals can be connected to ESP8266, so GPIO resource is very precious. Can we make flowing water light with less GPIO? In this chapter, we will learn a component, 74HC595, which can achieve the target.

Project 13.1 Flowing Water Light

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

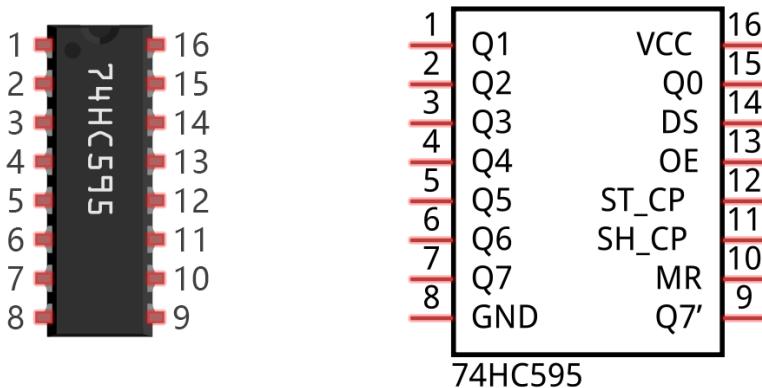
Component List

ESP8266 x1	USB cable		
Breadboard x1			
74HC595 x1	LED Bar Graph x1	Resistor 220Ω x8	Jumper wire M/M x17

Related knowledge

74HC595

A 74HC595 chip is used to convert serial data into parallel data. A 74HC595 chip can convert the serial data of one byte into 8 bits, and send its corresponding level to each of the 8 ports correspondingly. With this characteristic, the 74HC595 chip can be used to expand the IO ports of a ESP8266. At least 3 ports are required to control the 8 ports of the 74HC595 chip.



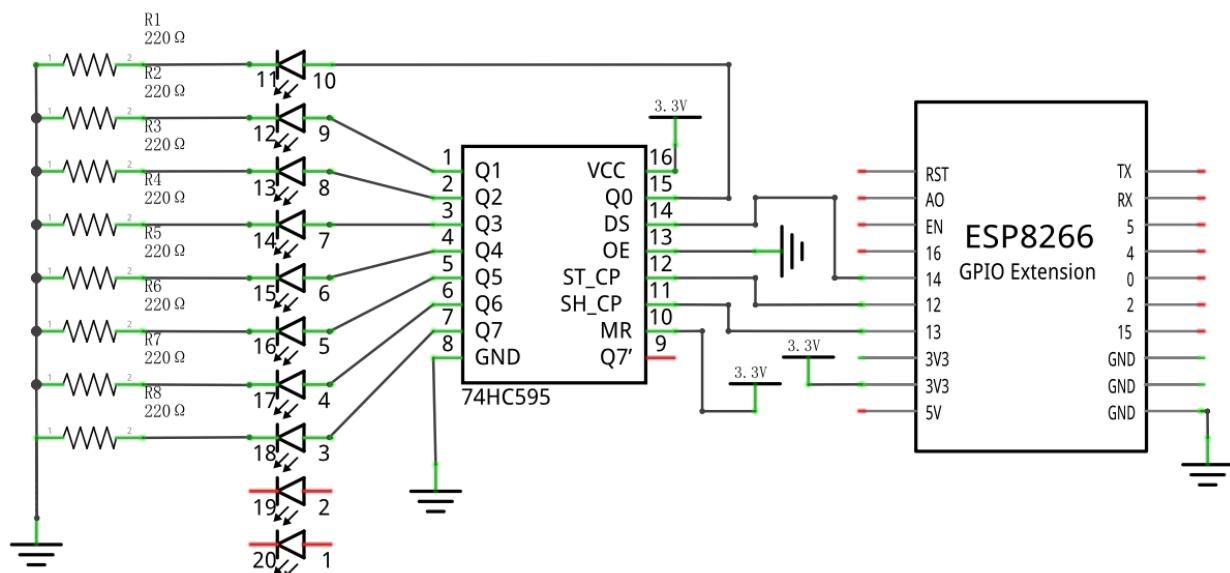
The ports of the 74HC595 chip are described as follows:

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

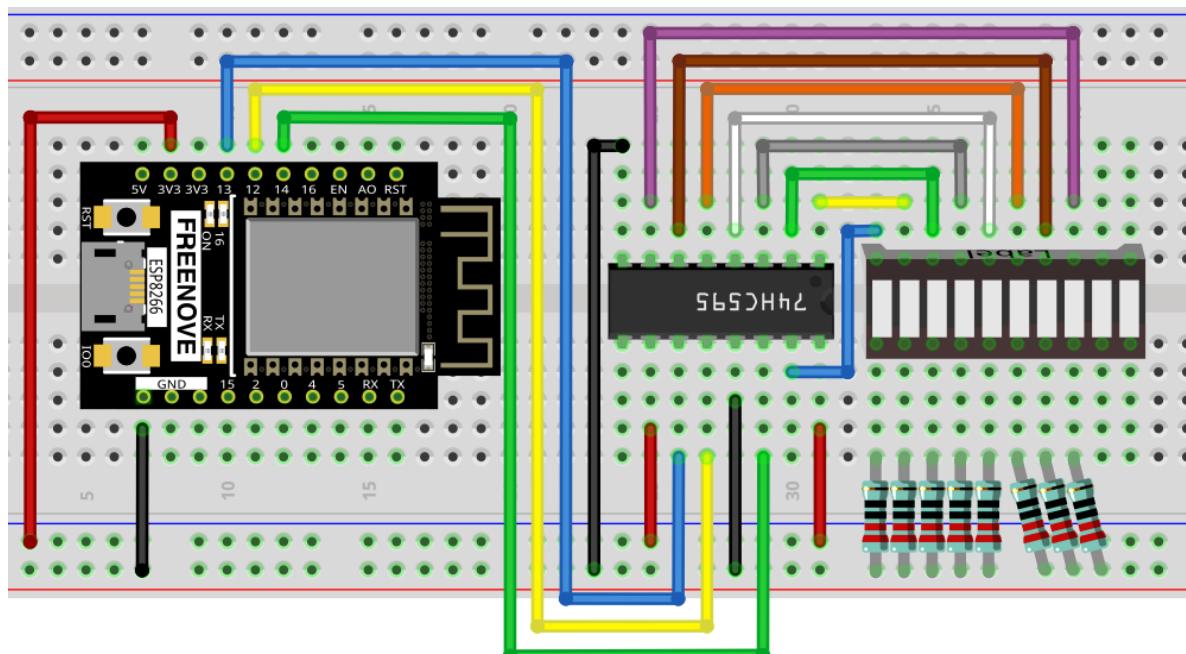
For more detail, please refer to the datasheet on the 74HC595 chip.

Circuit

Schematic diagram



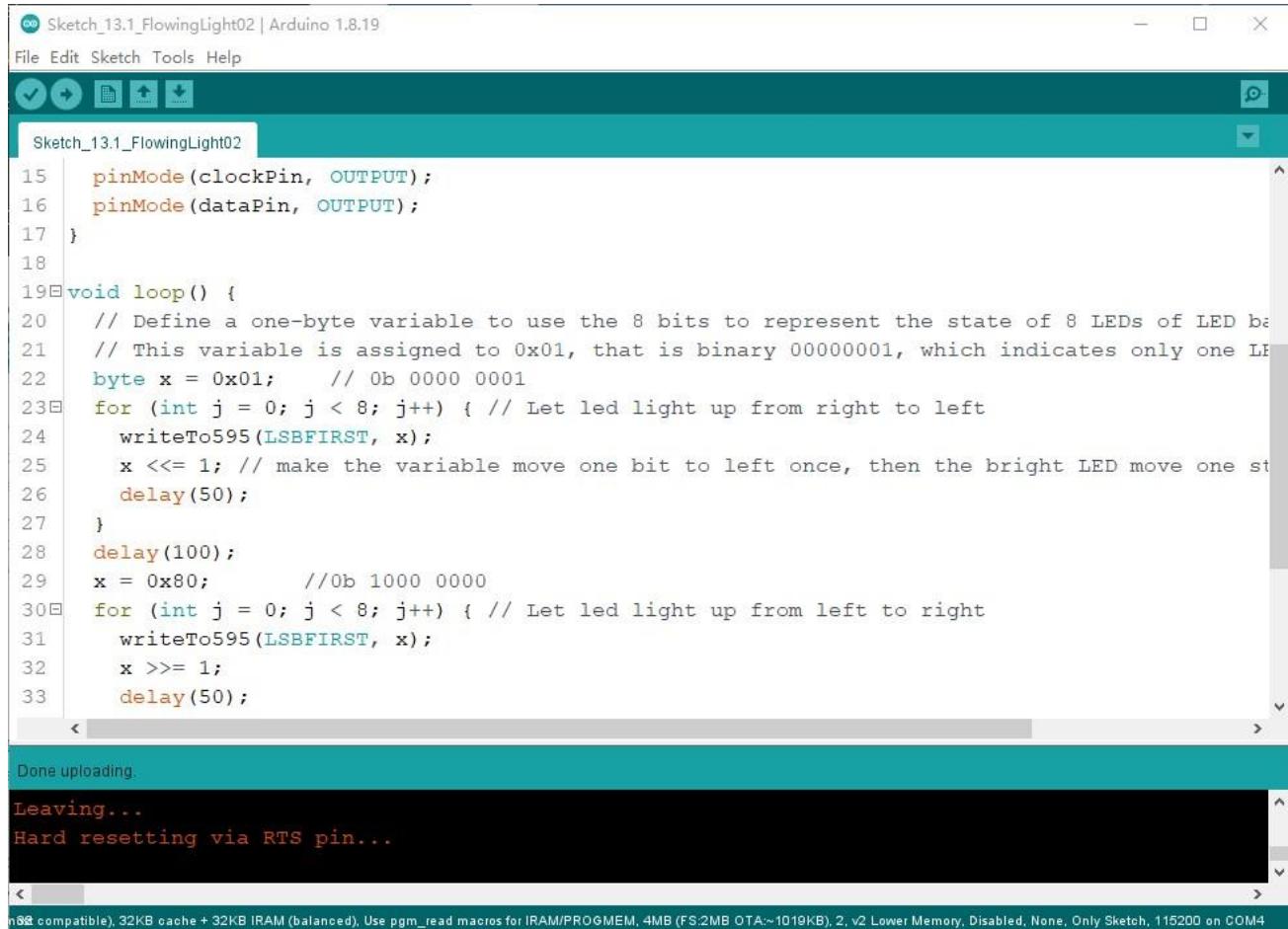
Hardware connection. If you need any support, please feel free to contact us via: support@freenove.com



Sketch

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

Sketch_13.1_FlowingLight02



```

Sketch_13.1_FlowingLight02 | Arduino 1.8.19
File Edit Sketch Tools Help
Sketch_13.1_FlowingLight02
15 pinMode(clockPin, OUTPUT);
16 pinMode(dataPin, OUTPUT);
17 }
18
19 void loop() {
20 // Define a one-byte variable to use the 8 bits to represent the state of 8 LEDs of LED bar
21 // This variable is assigned to 0x01, that is binary 00000001, which indicates only one LED
22 byte x = 0x01; // 0b 0000 0001
23 for (int j = 0; j < 8; j++) { // Let led light up from right to left
24   writeTo595(LSBFIRST, x);
25   x <= 1; // make the variable move one bit to left once, then the bright LED move one step
26   delay(50);
27 }
28 delay(100);
29 x = 0x80; // 0b 1000 0000
30 for (int j = 0; j < 8; j++) { // Let led light up from left to right
31   writeTo595(LSBFIRST, x);
32   x >>= 1;
33   delay(50);
}

```

Done uploading.
Leaving...
Hard resetting via RTS pin...

n82 compatible), 32KB cache + 32KB IRAM (balanced). Use pgm_read macros for IRAM/PROGMEM, 4MB (FS:2MB OTA:~1019KB), 2, v2 Lower Memory, Disabled, None, Only Sketch, 115200 on COM4

Download the code to ESP8266. You will see that LED bar graph starts with the flowing water pattern flashing from left to right and then back from right to left.

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

The following is the program code:

```

1 int latchPin = 12;           // Pin connected to ST_CP of 74HC595(Pin12)
2 int clockPin = 13;          // Pin connected to SH_CP of 74HC595(Pin11)
3 int dataPin = 14;           // Pin connected to DS of 74HC595(Pin14)
4
5 void setup() {
6   // set pins to output
7   pinMode(latchPin, OUTPUT);
8   pinMode(clockPin, OUTPUT);
9   pinMode(dataPin, OUTPUT);
10 }
11

```

Any concerns? ✉ support@freenove.com

```

12 void loop() {
13     // Define a one-byte variable to use the 8 bits to represent the state of 8 LEDs of LED bar
14     // graph.
15     // This variable is assigned to 0x01, that is binary 00000001, which indicates only one LED
16     // light on.
17     byte x = 0x01;      // 0b 0000 0001
18     for (int j = 0; j < 8; j++) { // Let led light up from right to left
19         writeTo595(LSBFIRST, x);
20         x <= 1; // make the variable move one bit to left once, then the bright LED move one step
21         to the left once.
22         delay(50);
23     }
24     delay(100);
25     x = 0x80;          //0b 1000 0000
26     for (int j = 0; j < 8; j++) { // Let led light up from left to right
27         writeTo595(LSBFIRST, x);
28         x >= 1;
29         delay(50);
30     }
31     delay(100);
32 }
33 void writeTo595(int order, byte _data) {
34     // Output low level to latchPin
35     digitalWrite(latchPin, LOW);
36     // Send serial data to 74HC595
37     shiftOut(dataPin, clockPin, order, _data);
38     // Output high level to latchPin, and 74HC595 will update the data to the parallel output
39     // port.
40     digitalWrite(latchPin, HIGH);
41 }
```

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

17	byte x = 0x01; // 0b 0000 0001
----	--------------------------------

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

18	for (int j = 0; j < 8; j++) { // Let led light up from right to left
19	writeTo595(LSBFIRST, x);
20	x <= 1;
21	delay(50);

22 }

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

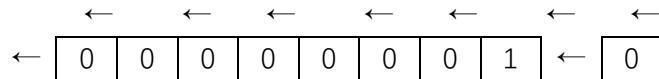
The subfunction writeTo595() is used to write data to 74HC595 and immediately output on the port of 74HC595.

Reference

<< operator

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

byte x = 1 << 1;

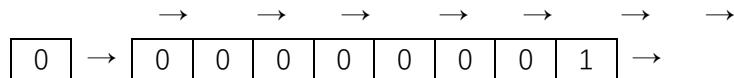


The result of x is 2 (binary 00000010).



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

byte x = 1 >> 1;



The result of x is 0 (00000000).



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

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

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

Parameters

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

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

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

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

For more details about shift function, please refer to:

<https://www.arduino.cc/reference/en/language/functions/advanced-io/shiftdown/>

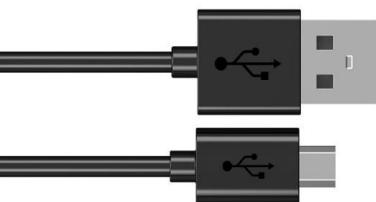
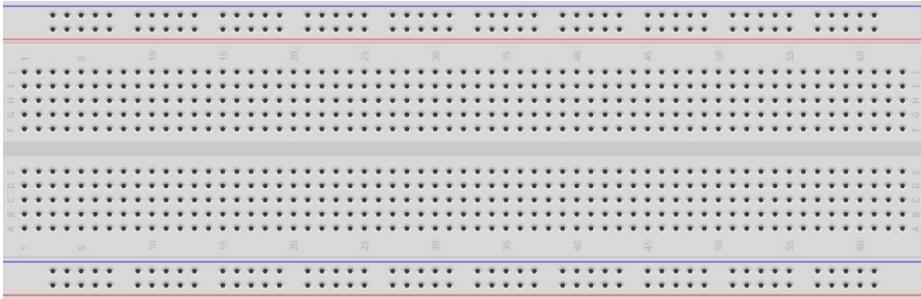
Chapter 14 74HC595 & 7-Segment Display.

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

Project 14.1 1-Digit 7-Segment Display.

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

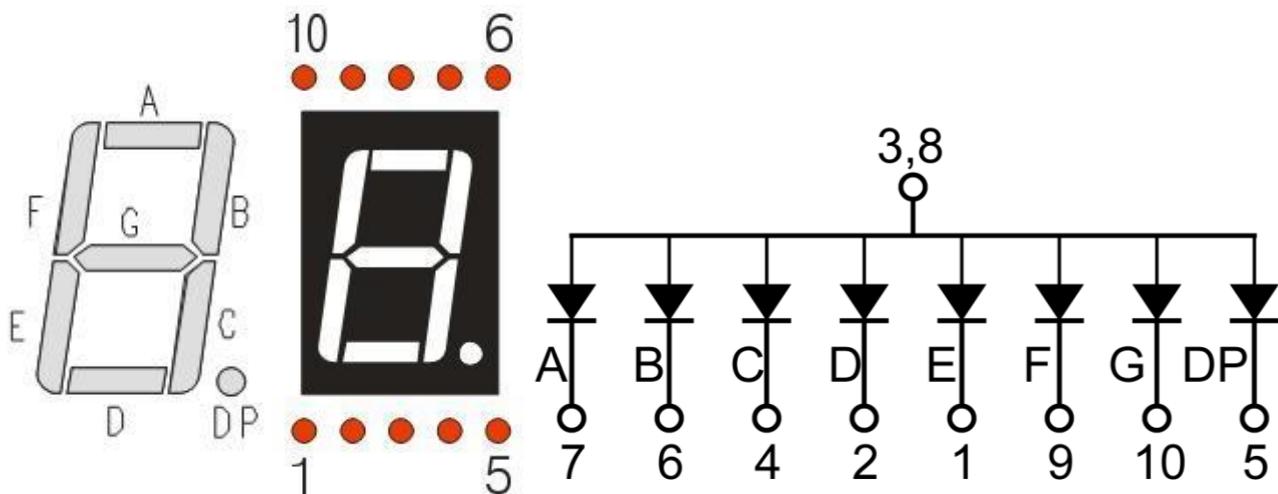
Component List

ESP8266 x1	USB cable		
			
Breadboard x1			
74HC595 x1	7-segment display x1	Resistor 220Ω x8	Jumper wire M/M
			

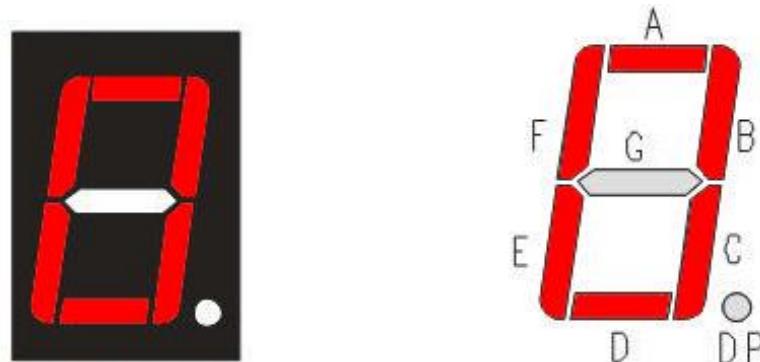
Component knowledge

7-segment display

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



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



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

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

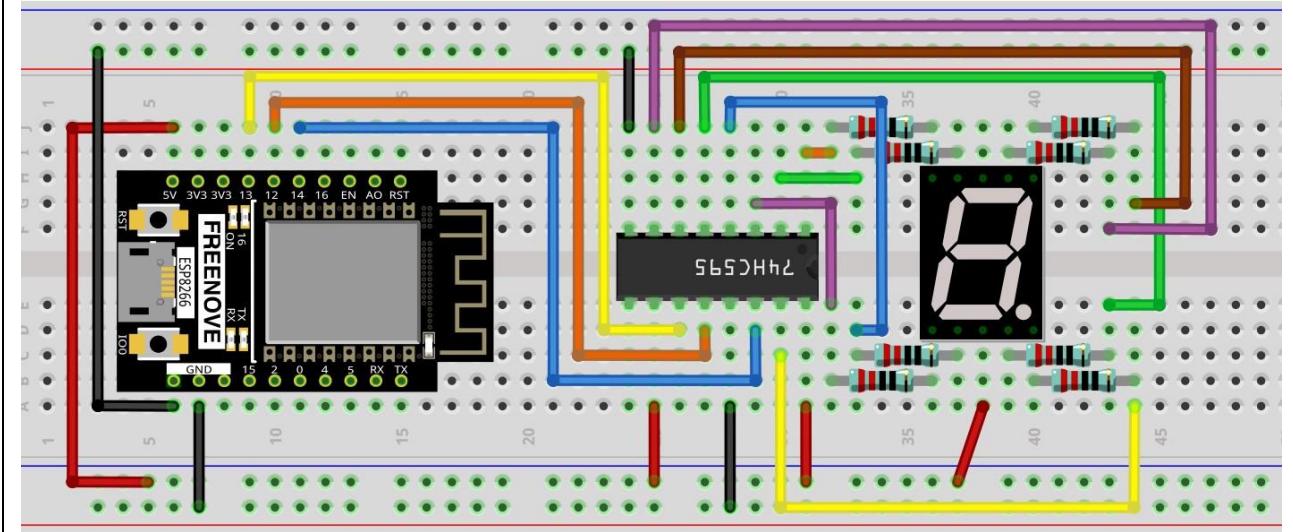
CHAR	DP	G	F	E	D	C	B	A	Hex	ASCII
0	1	1	0	0	0	0	0	0	0xc0	1100 0000
1	1	1	1	1	1	0	0	1	0xf9	1111 1001
2	1	0	1	0	0	1	0	0	0xa4	1010 0100
3	1	0	1	1	0	0	0	0	0xb0	1011 0000
4	1	0	0	1	1	0	0	1	0x99	1001 1001
5	1	0	0	1	0	0	1	0	0x92	1001 0010
6	1	0	0	0	0	0	1	0	0x82	1000 0010
7	1	1	1	1	1	0	0	0	0xf8	1111 1000
8	1	0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

Circuit

Schematic diagram



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



Sketch

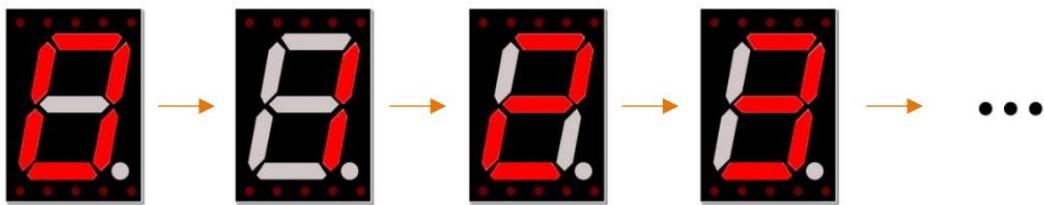
In this section, the 74HC595 is used in the same way as in the previous section, but with different values transferred. We can learn how to master the digital display by sending the coded value of "0" - "F".

Sketch_14.1_1_Digit_7-Segment_Display

The screenshot shows the Arduino IDE interface with the following details:

- Title Bar:** Sketch_14.1_1_Digit_7-Segment_Display | Arduino 1.8.19
- Menu Bar:** File Edit Sketch Tools Help
- Toolbar:** Includes icons for upload, download, and other functions.
- Code Editor:** Displays the C++ code for the sketch. The code initializes pins, sets them as outputs, and then enters a loop where it repeatedly sends data to a 74HC595 chip to display values from 0 to F on a digital tube. It includes a delay of 1 second between each character and clears the display at the end of the loop.
- Serial Monitor:** Shows the status of the upload process: "Done uploading.", "Leaving...", and "Hard resetting via RTS pin...".
- Status Bar:** Shows system information: "idf compatible, 32KB cache + 32KB IRAM (balanced), Use pgm_read macros for IRAM/PROGMEM, 4MB (FS:2MB OTA:~1019KB), 2, v2 Lower Memory, Disabled, None, Only Sketch, 115200 on COM4".

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



The following is the program code:

```

1 int dataPin = 14;           // Pin connected to DS of 74HC595 (Pin14)
2 int latchPin = 12;          // Pin connected to ST_CP of 74HC595 (Pin12)
3 int clockPin = 13;          // Pin connected to SH_CP of 74HC595 (Pin11)
4 // Define the encoding of characters 0-F for the common-anode 7-Segment Display
5 byte num[] = {
6     0xc0, 0xf9, 0xa4, 0xb0, 0x99, 0x92, 0x82, 0xf8,
7     0x80, 0x90, 0x88, 0x83, 0xc6, 0xa1, 0x86, 0x8e
8 };
9
10 void setup() {
11     // set pins to output
12     pinMode(latchPin, OUTPUT);
13     pinMode(clockPin, OUTPUT);
14     pinMode(dataPin, OUTPUT);
15 }
16
17 void loop() {
18     // display 0-F on digital tube
19     for (int i = 0; i < 16; i++) {
20         writeData(num[i]); // Send data to 74HC595
21         delay(1000);      // delay 1 second
22         writeData(0xff); // Clear the display content
23     }
24 }
25
26 void writeData(int value) {
27     // Make latchPin output low level
28     digitalWrite(latchPin, LOW);
29     // Send serial data to 74HC595
30     shiftOut(dataPin, clockPin, LSBFIRST, value);
31     // Make latchPin output high level
32     digitalWrite(latchPin, HIGH);
33 }
```

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

```

4 // Define the encoding of characters 0-F for the common-anode 7-Segment Display
5 byte num[] = {
6     0xc0, 0xf9, 0xa4, 0xb0, 0x99, 0x92, 0x82, 0xf8,
7     0x80, 0x90, 0x88, 0x83, 0xc6, 0xa1, 0x86, 0x8e
8 };

```

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

```

17 void loop() {
18     // display 0-F on digital tube
19     for (int i = 0; i < 16; i++) {
20         writeData(num[i]); // Send data to 74HC595
21         delay(1000);      // delay 1 second
22         writeData(0xff); // Clear the display content
23     }
24 }

```

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

```

26 void writeData(int value) {
27     // Make latchPin output low level
28     digitalWrite(latchPin, LOW);
29     // Send serial data to 74HC595
30     shiftOut(dataPin, clockPin, LSBFIRST, value);
31     // Make latchPin output high level, then 74HC595 will update data to parallel output
32     digitalWrite(latchPin, HIGH);
33 }

```

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

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



Chapter 15.1 Motor & Driver

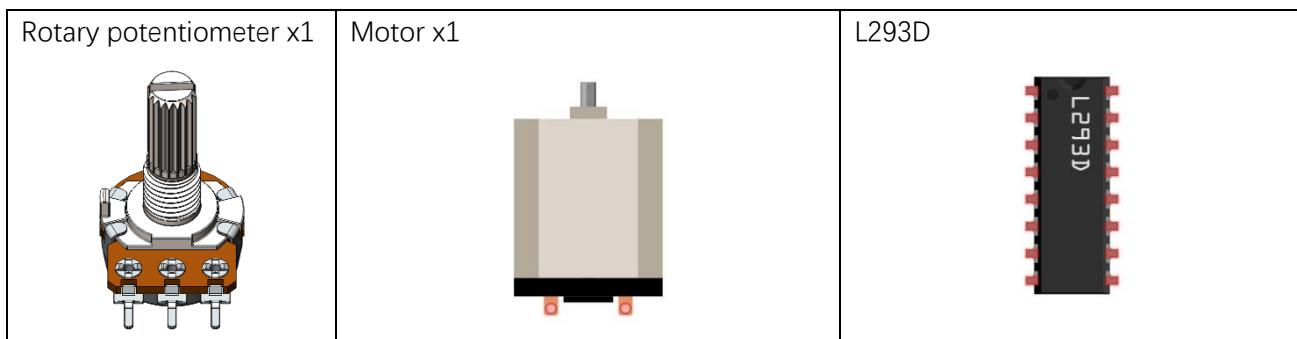
In this chapter, we will learn about DC motors and DC motor drivers and how to control the speed and direction of a DC motor.

Project 15.1 Control Motor with Potentiometer

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

Component List

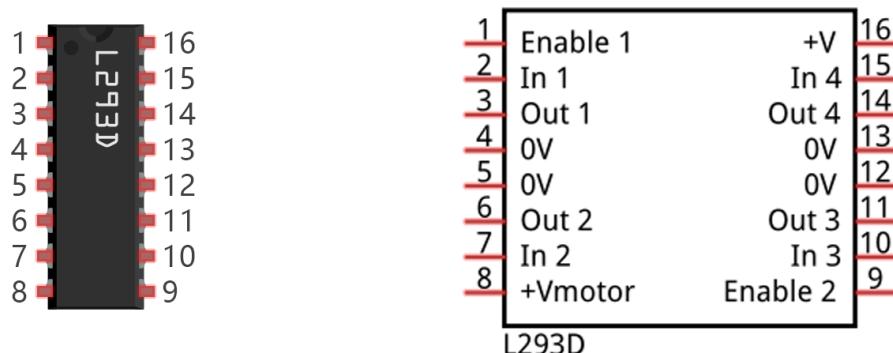
ESP8266 x1	USB cable
Breadboard x1	Breadboard Power module x1
Jumper wire M/M x12	9V battery (prepared by yourself) & battery line



Component knowledge

L293D

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



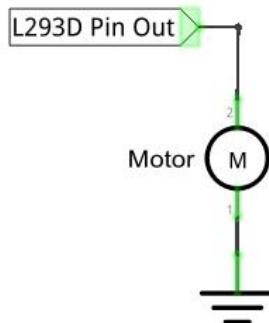
Port description of L293D module is as follows:

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

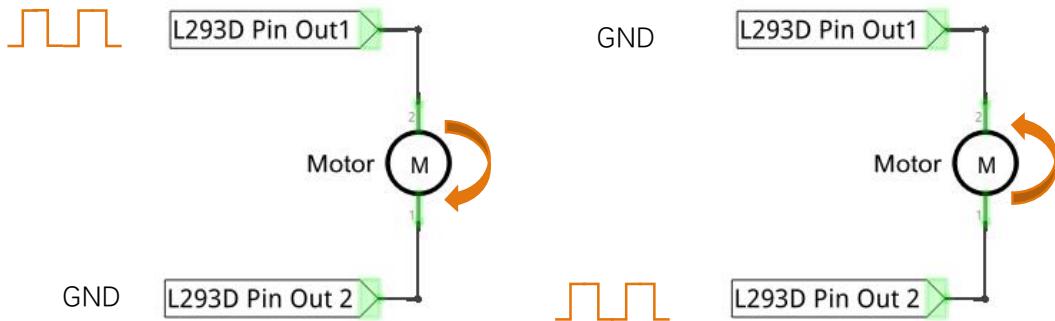
For more detail, please refer to the datasheet for this IC Chip.

When using L293D to drive DC motor, there are usually two connection options.

The following connection option uses one channel of the L293D, which can control motor speed through the PWM, However the motor then can only rotate in one direction.



The following connection uses two channels of the L293D: one channel outputs the PWM wave, and the other channel connects to GND, therefore you can control the speed of the motor. When these two channel signals are exchanged, not only controls the speed of motor, but also can control the steering of the motor.

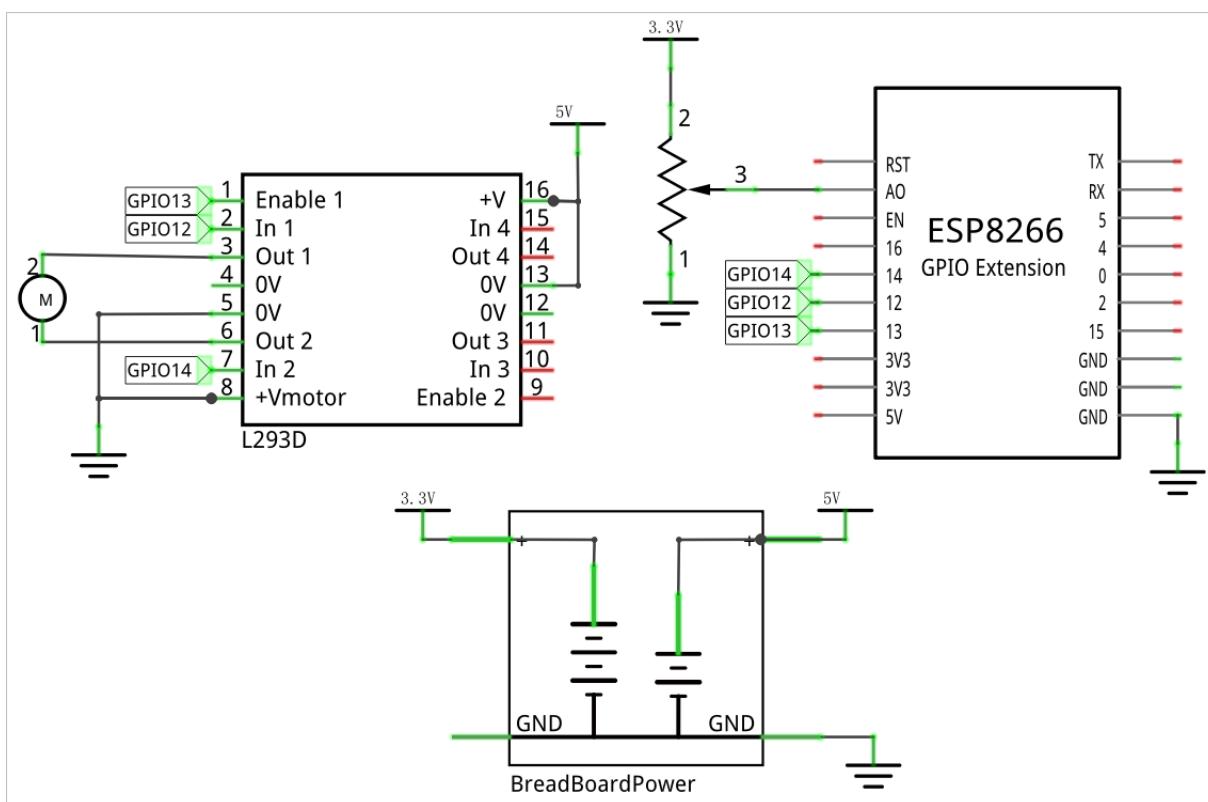


In practical use the motor is usually connected to channel 1 and 2 by outputting different levels to in1 and in2 to control the rotational direction of the motor, and output to the PWM wave to Enable1 port to control the motor's rotational speed. If the motor is connected to channel 3 and 4 by outputting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

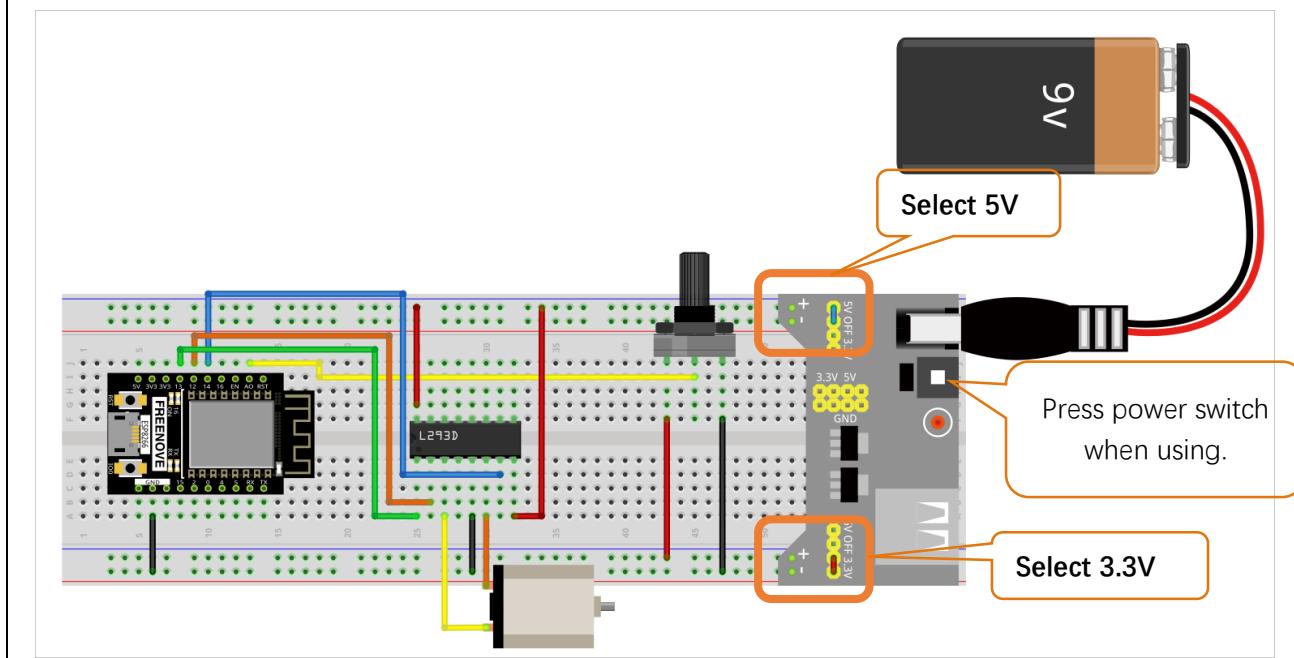
Circuit

Use caution when connecting this circuit because the DC Motor is a high-power component. **Do not use the power provided by the ESP8266 to power the motor directly, as this may cause permanent damage to your RPi!** The logic circuit can be powered by the ESP8266's power or an external power supply, which should share a common ground with ESP8266.

Schematic diagram

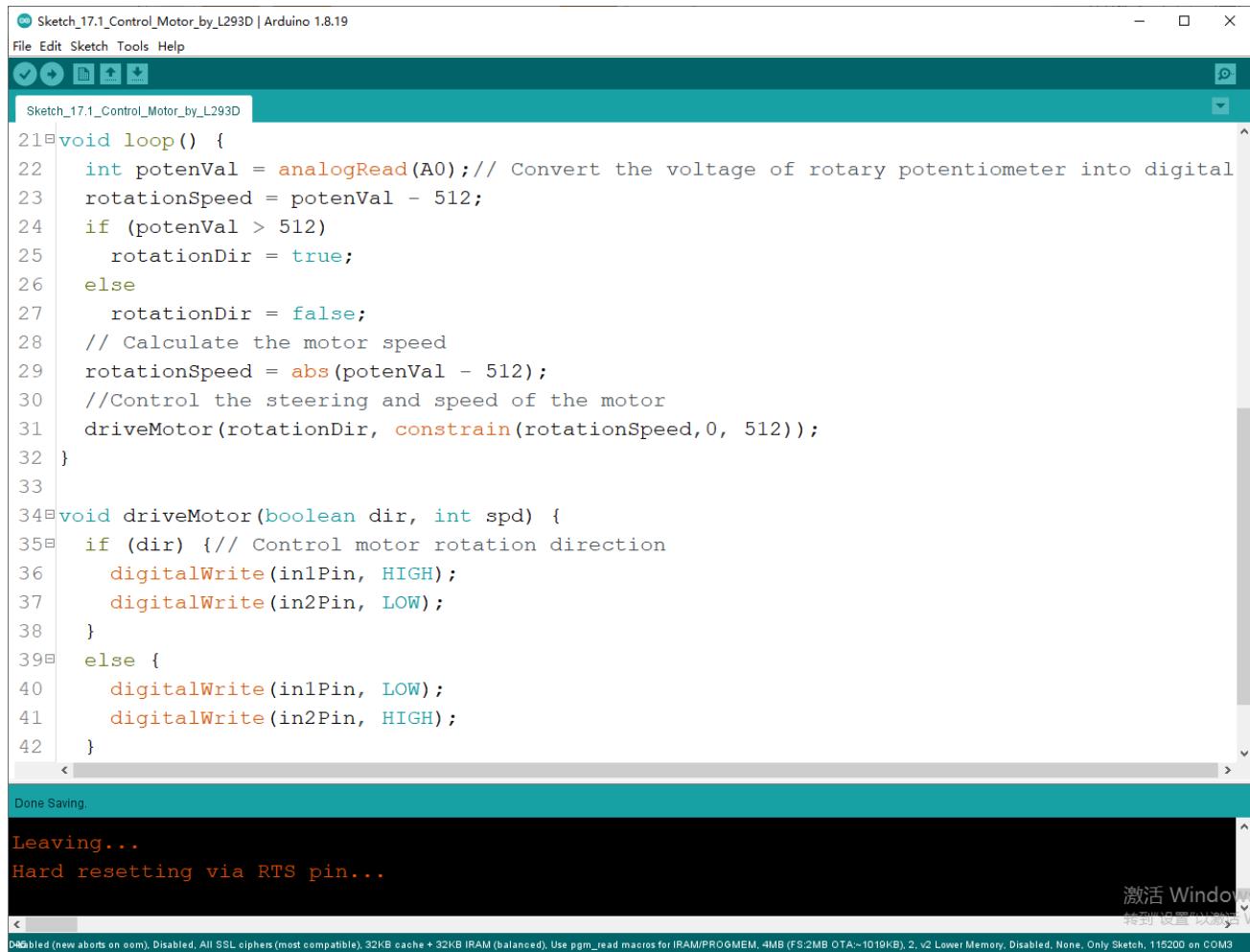


Hardware connection. If you need any support, please feel free to contact us via: support@freenove.com



Sketch

Sketch_15.1_Control_Motor_by_L293D



```

Sketch_15.1_Control_Motor_by_L293D | Arduino 1.8.19
File Edit Sketch Tools Help
Sketch_15.1_Control_Motor_by_L293D
21 void loop() {
22     int potenVal = analogRead(A0); // Convert the voltage of rotary potentiometer into digital
23     rotationSpeed = potenVal - 512;
24     if (potenVal > 512)
25         rotationDir = true;
26     else
27         rotationDir = false;
28     // Calculate the motor speed
29     rotationSpeed = abs(potenVal - 512);
30     //Control the steering and speed of the motor
31     driveMotor(rotationDir, constrain(rotationSpeed, 0, 512));
32 }
33
34 void driveMotor(boolean dir, int spd) {
35     if (dir) { // Control motor rotation direction
36         digitalWrite(in1Pin, HIGH);
37         digitalWrite(in2Pin, LOW);
38     }
39     else {
40         digitalWrite(in1Pin, LOW);
41         digitalWrite(in2Pin, HIGH);
42     }
}
Done Saving.

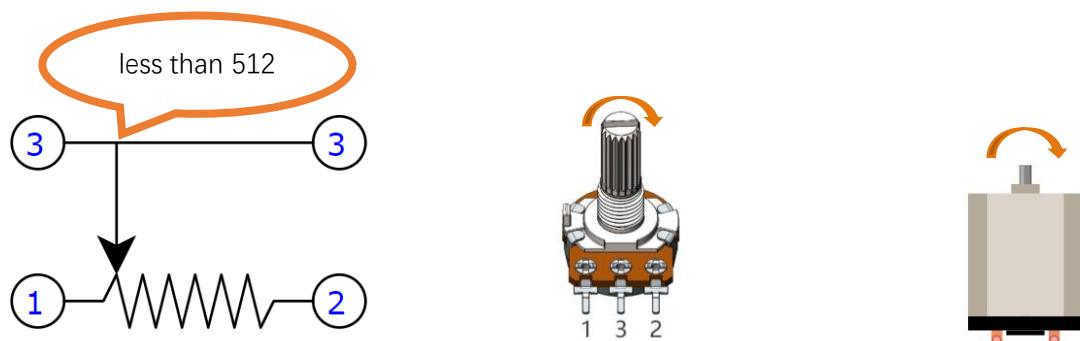
Leaving...
Hard resetting via RTS pin...

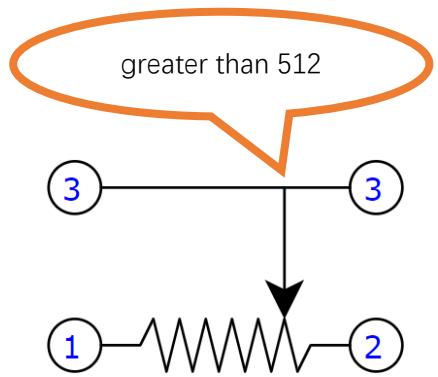
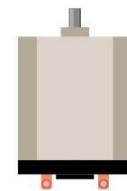
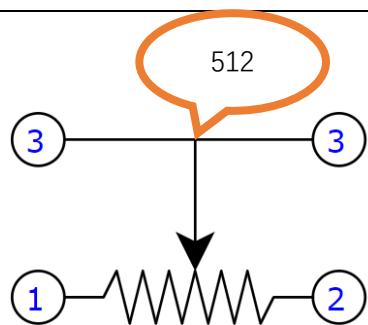
```

激活 Windows
转到设置以激活 V

Disabled (new aborts on com). Disabled, All SSL ciphers (most compatible), 32KB cache + 32KB IRAM (balanced), Use pgm_read macros for IRAM/PROGMEM, 4MB (FS:2MB OTA:~1019KB), 2, v2 Lower Memory, Disabled, None, Only Sketch, 115200 on COM3

Download code to ESP8266, rotate the potentiometer in one direction and the motor speeds up slowly in one direction. And then rotate the potentiometer in the other direction and the motor will slow down to stop. And then rotate it in an inverse direction to accelerate the motor.





The following is the sketch:

```

1 int in1Pin = 12;      // Define L293D channel 1 pin
2 int in2Pin = 14;      // Define L293D channel 2 pin
3 int enable1Pin = 13;  // Define L293D enable 1 pin
4
5 boolean rotationDir; // Define a variable to save the motor's rotation direction
6 int rotationSpeed;   // Define a variable to save the motor rotation speed
7
8 void setup() {
9     // Initialize the pin into an output mode:
10    pinMode(in1Pin, OUTPUT);
11    pinMode(in2Pin, OUTPUT);
12    pinMode(enable1Pin, OUTPUT);
13 }
14
15 void loop() {
16     int potenVal = analogRead(A0); // Convert the voltage of rotary potentiometer into digital
17     rotationSpeed = potenVal - 512;
18     if (potenVal > 512)
19         rotationDir = true;
20     else
21         rotationDir = false;
22     // Calculate the motor speed
23     rotationSpeed = abs(potenVal - 512);
24     //Control the steering and speed of the motor
25     driveMotor(rotationDir, constrain(rotationSpeed, 0, 512));
26 }
27
28 void driveMotor(boolean dir, int spd) {
29     if (dir) { // Control motor rotation direction
30         digitalWrite(in1Pin, HIGH);
31         digitalWrite(in2Pin, LOW);
32     }
33     else {
34         digitalWrite(in1Pin, LOW);
35         digitalWrite(in2Pin, HIGH);
36     }
37     analogWrite(enable1Pin, spd); // Control motor rotation speed
38 }
```

The ADC of ESP8266 has a 10-bit accuracy, corresponding to a range from 0 to 1023. In this program, set the number 512 as the midpoint. If the value of ADC is less than 512, make the motor rotate in one direction. If the value of ADC is greater than 512, make the motor rotate in the other direction. Subtract 512 from the ADC value and take the absolute value and use this result as the speed of the motor.

16	int potenVal = analogRead(A0); // Convert the voltage of rotary potentiometer into digital
----	--

```
17 rotationSpeed = potenVal - 512;
18 if (potenVal > 512)
19     rotationDir = true;
20 else
21     rotationDir = false;
22 // Calculate the motor speed
23 rotationSpeed = abs(potenVal - 512);
24 //Control the steering and speed of the motor
25 driveMotor(rotationDir, constrain(rotationSpeed, 0, 512));
26 }
```

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

```
28 void driveMotor(boolean dir, int spd) {
29     if (dir) {// Control motor rotation direction
30         digitalWrite(in1Pin, HIGH);
31         digitalWrite(in2Pin, LOW);
32     }
33     else {
34         digitalWrite(in1Pin, LOW);
35         digitalWrite(in2Pin, HIGH);
36     }
37     analogWrite(enable1Pin, spd); // Control motor rotation speed
38 }
```

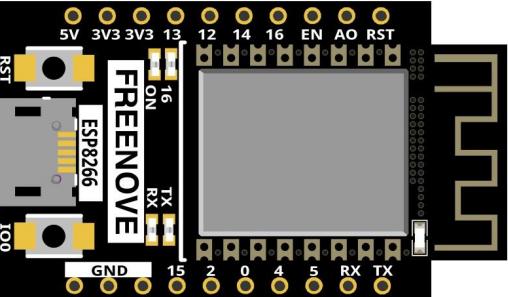
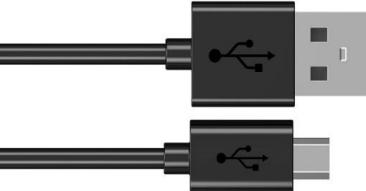
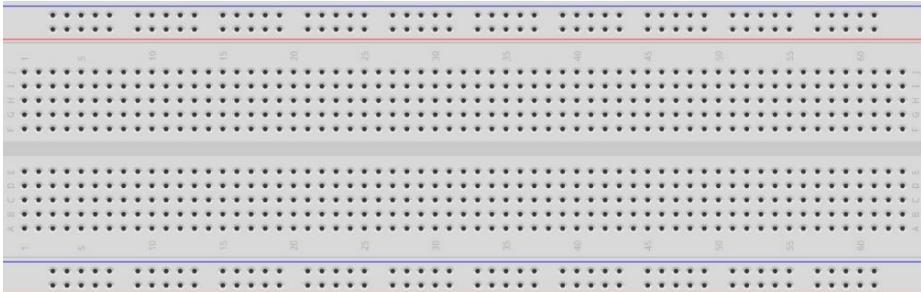
Chapter 16 Servo

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

Project 16.1 Servo Sweep

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

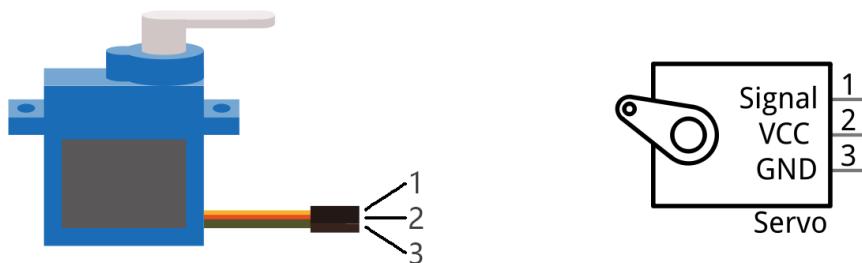
Component List

ESP8266 x1	USB cable
	
Breadboard x1	
Servo x1	Jumper wire M/M x5

Component knowledge

Servo

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



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

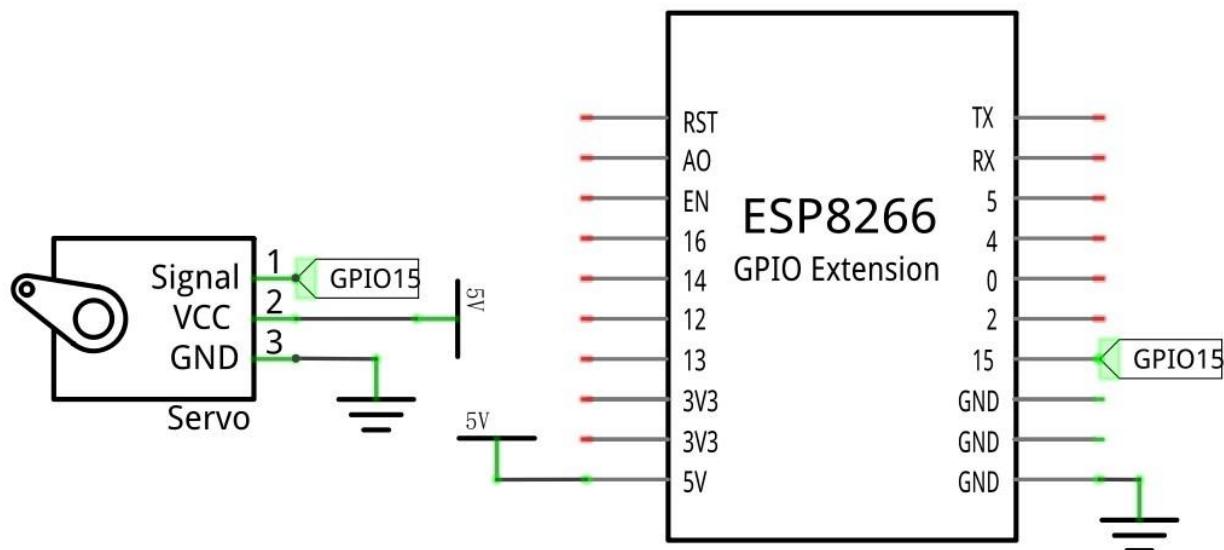
High level time	Servo angle
0.5ms	0 degree
1ms	45 degree
1.5ms	0 degree
2ms	45 degree
2.5ms	180 degree

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

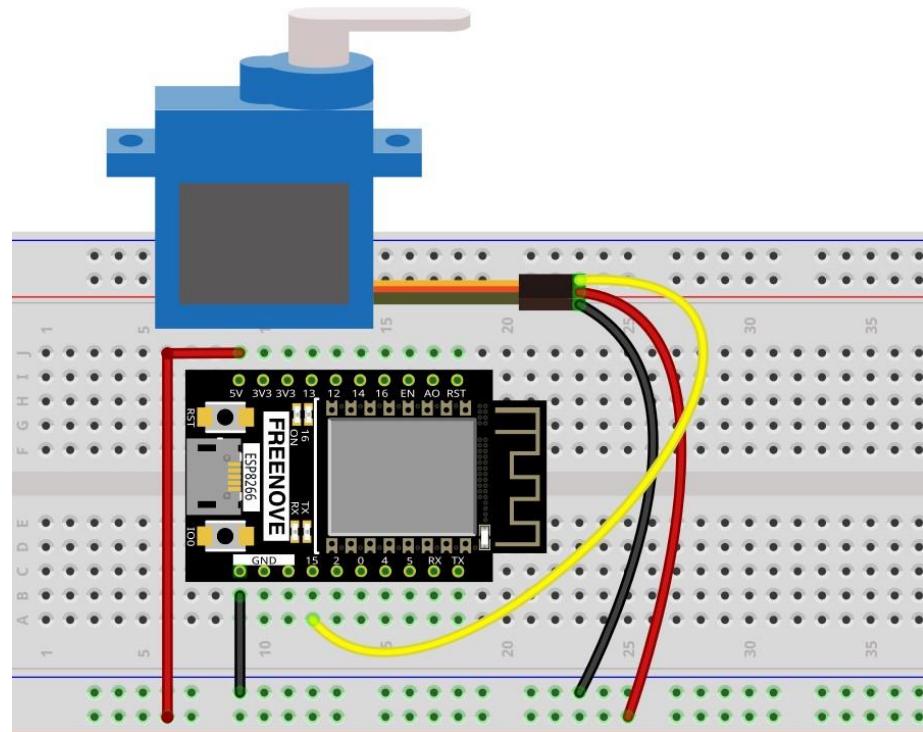
Circuit

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

Schematic diagram



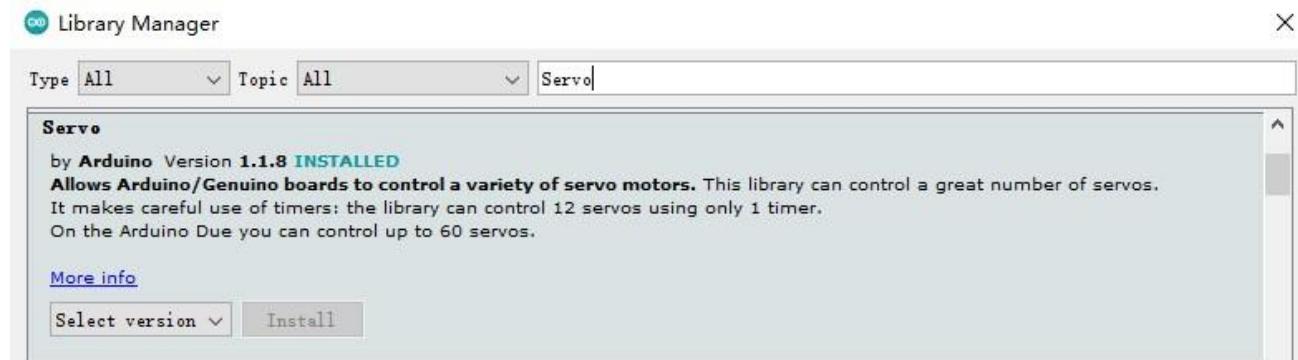
Hardware connection. If you need any support, please feel free to contact us via: support@freenove.com



Sketch

How to install the library

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



Use the Servo library to control the servo motor and let the servo motor rotate back and forth.

Sketch_16.1_Servo_Sweep

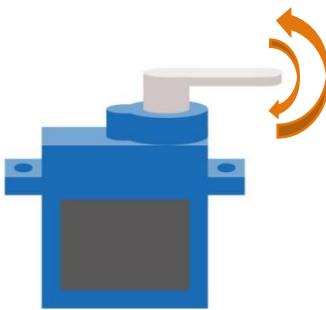
The screenshot shows the Arduino IDE interface with the following details:

- Title Bar:** Sketch_16.1_Servo_Sweep | Arduino 1.8.19
- File Menu:** File Edit Sketch Tools Help
- Toolbar:** Includes icons for Open, Save, Print, and others.
- Code Editor:** Displays the following C++ code for a servo sweep:

```

12 int servoPin = 15; // Servo motor pin
13
14 void setup() {
15   myservo.attach(servoPin, 500, 2500); // attaches the servo on servoPin to the servo object
16 }
17 void loop() {
18
19   for (posVal = 0; posVal <= 180; posVal += 1) { // goes from 0 degrees to 180 degrees
20     // in steps of 1 degree
21     myservo.write(posVal); // tell servo to go to position in variable 'pos'
22     delay(15); // waits 15ms for the servo to reach the position
23   }
24   for (posVal = 180; posVal >= 0; posVal -= 1) { // goes from 180 degrees to 0 degrees
25     myservo.write(posVal); // tell servo to go to position in variable 'pos'
26     delay(15); // waits 15ms for the servo to reach the position
27   }
28 }
```
- Status Bar:** Done Saving.
- Serial Monitor:** Shows the text "Leaving... Hard resetting via RTS pin..."
- Bottom Status:** nRF24L01+ Disabled, All SSL ciphers (most compatible), 32KB cache + 32KB IRAM (balanced), Use pgm_read macros for IRAM/PROGMEM, 4MB (FS:2MB OTA:~1019KB), 2, v2 Lower Memory, Disabled, None, Only Sketch, 115200 on COM3

Compile and upload the code to ESP8266, the servo will rotate from 0 degrees to 180 degrees and then reverse the direction to make it rotate from 180 degrees to 0 degrees and repeat these actions in an endless loop.



The following is the program code:

```

1 #include <Servo.h>
2
3 Servo myservo; // create servo object to control a servo
4 int posVal = 0; // variable to store the servo position
5 int servoPin = 15; // Servo motor pin
6 void setup() {
7     myservo.attach(servoPin, 500, 2500); // attaches the servo on servoPin to the servo object
8 }
9 void loop() {
10     for (posVal = 0; posVal <= 180; posVal += 1) { // goes from 0 degrees to 180 degrees
11         // in steps of 1 degree
12         myservo.write(posVal); // tell servo to go to position in variable 'pos'
13         delay(15); // waits 15ms for the servo to reach the position
14     }
15     for (posVal = 180; posVal >= 0; posVal -= 1) { // goes from 180 degrees to 0 degrees
16         myservo.write(posVal); // tell servo to go to position in variable 'pos'
17         delay(15); // waits 15ms for the servo to reach the position
18     }
}

```

Servo uses the Servo library, like the following reference to Servo library:

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

Servo library provides the Servo class that controls it. Servo class must be instantiated before using:

```
3 Servo myservo; // create servo object to control a servo
```

Set the control servo motor pin, the time range of high level.

```
7 myservo.attach(servoPin, 500, 2500); // attaches the servo on servoPin to the servo object
```

After initializing the servo, you can control the servo to rotate to a specific angle:

```
15 myservo.write(posVal);
```

Reference

```
class Servo
```

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

```
Servo myservo;
```

The function commonly used in the servo class is as follows:

setPeriodHertz(data): Set the frequency of the servo motor.

attach(pin,low,high): Initialize the servo,

pin: the port connected to servo signal line.

low: set the time of high level corresponding to 0 degree.

high: set the time of high level corresponding to 180 degrees.

write(angle): Control servo to rotate to the specified angle.



Project 16.2 Servo Knop

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

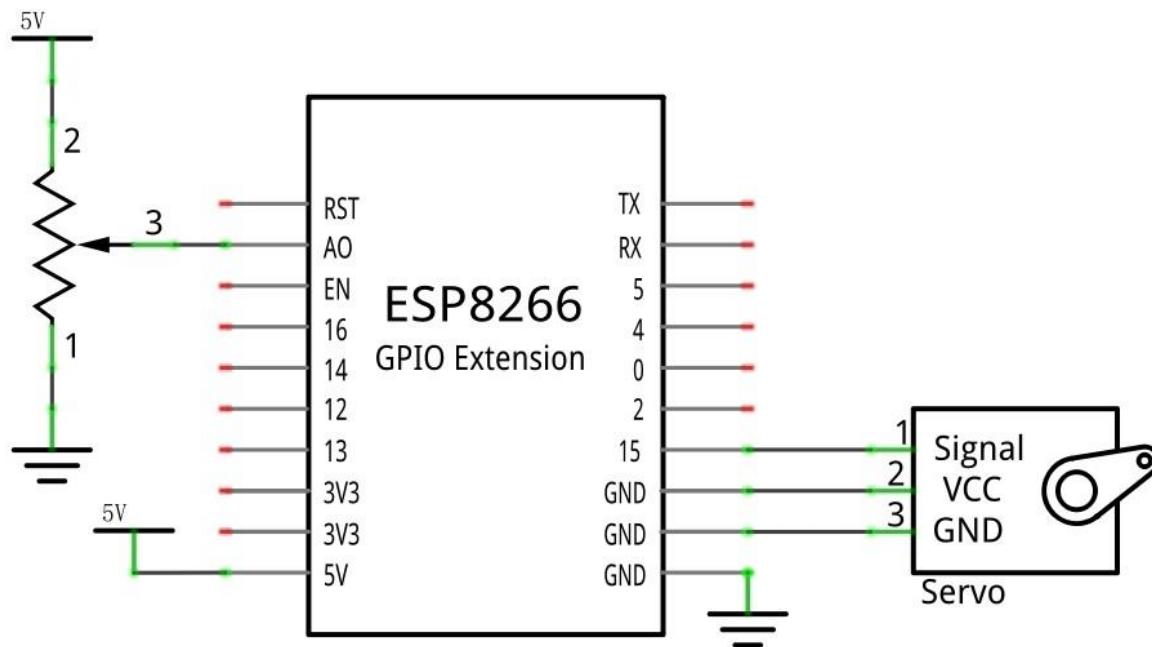
Component List

ESP8266 x1	USB cable	
A black ESP8266 module with a grey PCB. It has several pins labeled: 5V, 3V3, 3V3, 13, 12, 14, 16, EN, AO, RST, GND, 15, 2, 0, 4, 5, RX, TX. The word "FREENOVE" is printed on the PCB.	Two black USB cables, each with a standard A-type connector and a smaller micro-B or similar connector.	
Breadboard x1		
A top-down view of a breadboard with its grid of holes and row numbers (1-30) and column letters (A-H) visible.		
Servo x1	Jumper wire M/M x10	Rotary potentiometer x1
A blue servo motor with a white plastic housing and a metal gear assembly.	A green jumper wire with two black plastic caps on the ends.	A brown rotary potentiometer with a silver shaft and three terminals.

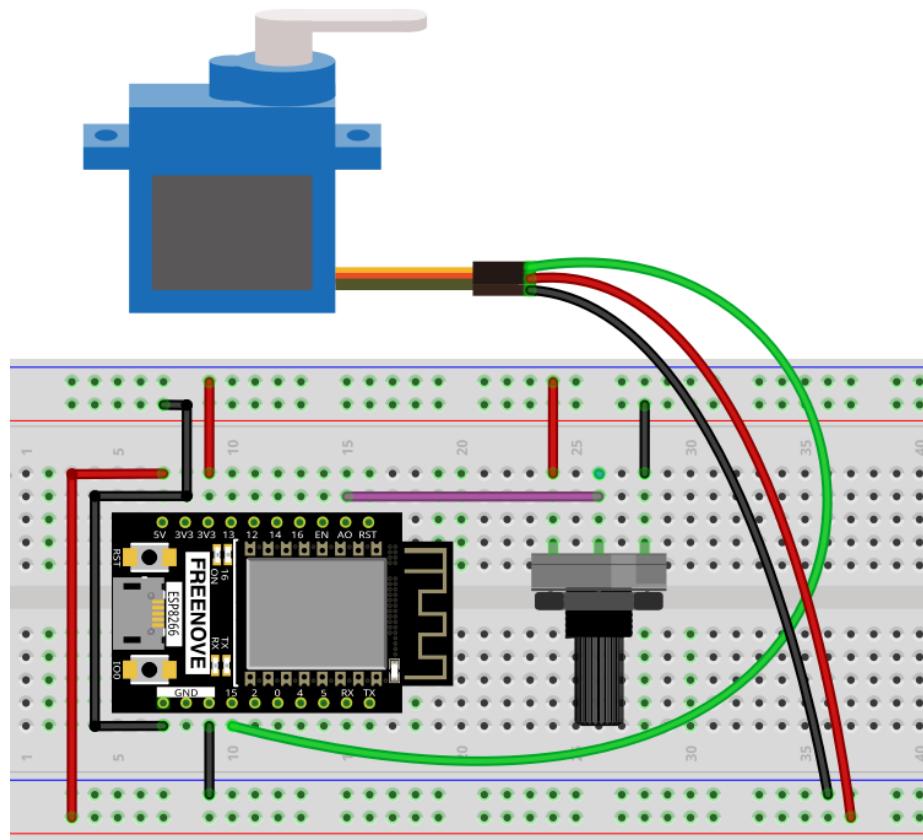
Circuit

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

Schematic diagram



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



Any concerns? ✉ support@freenove.com



Sketch

Sketch_16.2_Control_Servo_by_Potentiometer

```

Sketch_18.2_Control_Servo_by_Potentiometer | Arduino 1.8.19
File Edit Sketch Tools Help
Sketch_18.2_Control_Servo_by_Potentiometer
17 {
18     myservo.attach(servoPin, 500, 2500);
19     Serial.begin(115200);
20 }
21
22 void loop() {
23     // read the value of the potentiometer (value between 0 and 1023)
24     potVal = analogRead(potPin);
25     Serial.printf("potVal_1: %d\t",potVal);
26     // scale it to use it with the servo (value between 0 and 180)
27     potVal = map(potVal, 0, ADC_Max, 0, 180);
28     // set the servo position according to the scaled value
29     myservo.write(potVal);
30     Serial.printf("potVal_2: %d\n",potVal);
31     delay(15); // wait for the servo to get there
32 }

Done uploading.

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

```

ESP8266(new aborts on dom), Disabled, All SSL ciphers (most compatible), 32KB cache + 32KB IRAM (balanced). Use pgm_read macros for IRAM/PROGMEM, 4MB (FS:2MB OTA:~1019KB), 2, v2 Lower Memory, Disabled, None, Only Sketch, 115200 on COM3

Compile and upload the code to ESP8266, twist the potentiometer back and forth, and the servo motor rotates accordingly.



The following is the program code:

```

1 #include <Servo.h>
2 #define ADC_Max 1023    // This is the default ADC max value on the ESP8266 (12 bits ADC
3 width);
4
5 Servo myservo; // create servo object to control a servo
6
7 int servoPin = 15;      // GPIO pin used to connect the servo control (digital out)
8 int potPin = A0;        // GPIO pin used to connect the potentiometer (analog in)

```

```
9 int potVal; //variable to read the value from the analog pin
10
11 void setup()
12 {
13     myservo.attach(servoPin, 500, 2500); // attaches the servo on servoPin to the servo object
14     Serial.begin(115200);
15 }
16
17 void loop() {
18     potVal = analogRead(potPin); // read the value of the potentiometer (value between 0 and
19     Serial.printf("potVal_1: %d\t", potVal);
20     potVal = map(potVal, 0, ADC_Max, 0, 180); // scale it to use it with the servo (value
21     between 0 and 180)
22     myservo.write(potVal); // set the servo position according to the scaled
23     value
24     Serial.printf("potVal_2: %d\n", potVal);
25     delay(15); // wait for the servo to get there
26 }
```

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



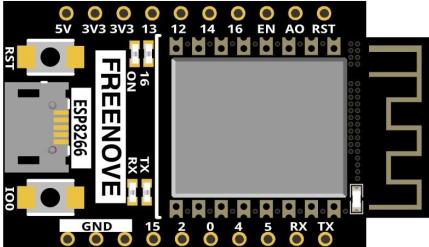
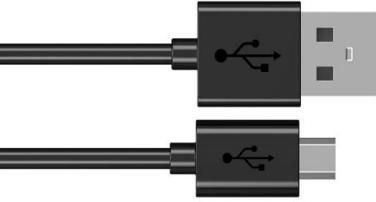
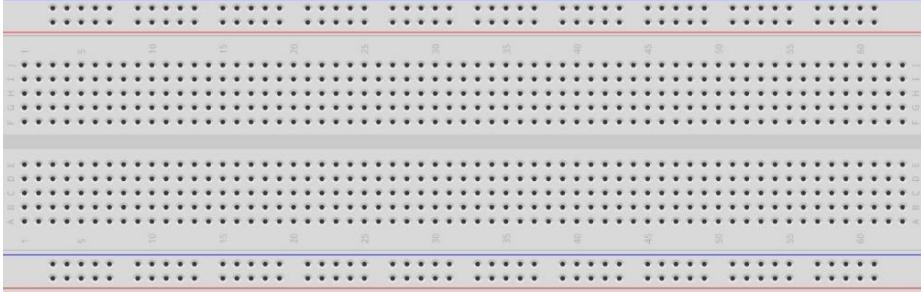
Chapter 17 LCD1602

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

Project 17.1 LCD1602

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

Component List

ESP8266 x1	USB cable
	
Breadboard x1	
LCD1602 Module x1	Jumper wire F/M x6

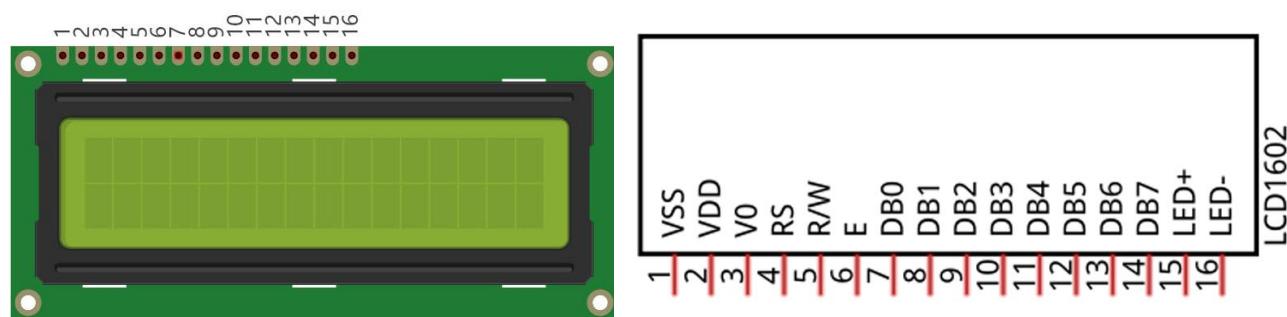
Component knowledge

I2C communication

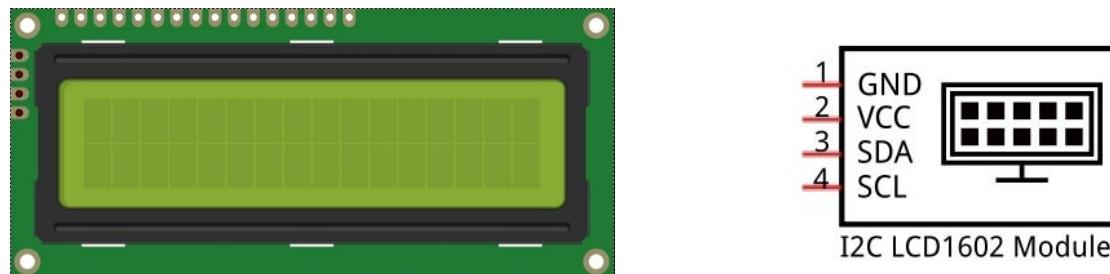
I2C (Inter-Integrated Circuit) is a two-wire serial communication mode, which can be used for the connection of micro controllers and their peripheral equipment. Devices using I2C communication must be connected to the serial data (SDA) line, and serial clock (SCL) line (called I2C bus). Each device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

LCD1602 communication

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

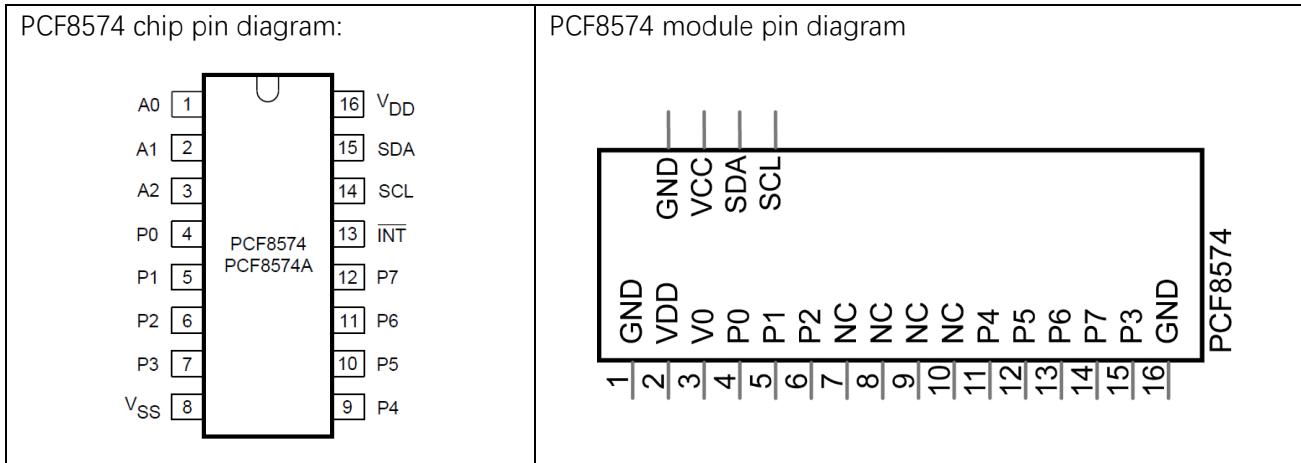


I2C LCD1602 display screen integrates a I2C interface, which connects the serial-input & parallel-output module to the LCD1602 display screen. This allows us to only use 4 lines to operate the LCD1602.

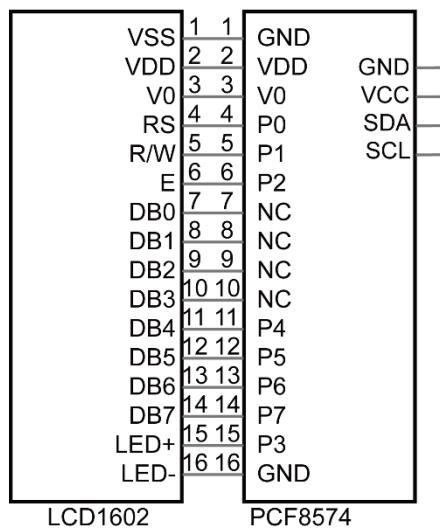


The serial-to-parallel IC chip used in this module is PCF8574T (PCF8574AT), and its default I2C address is 0x27(0x3F).

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



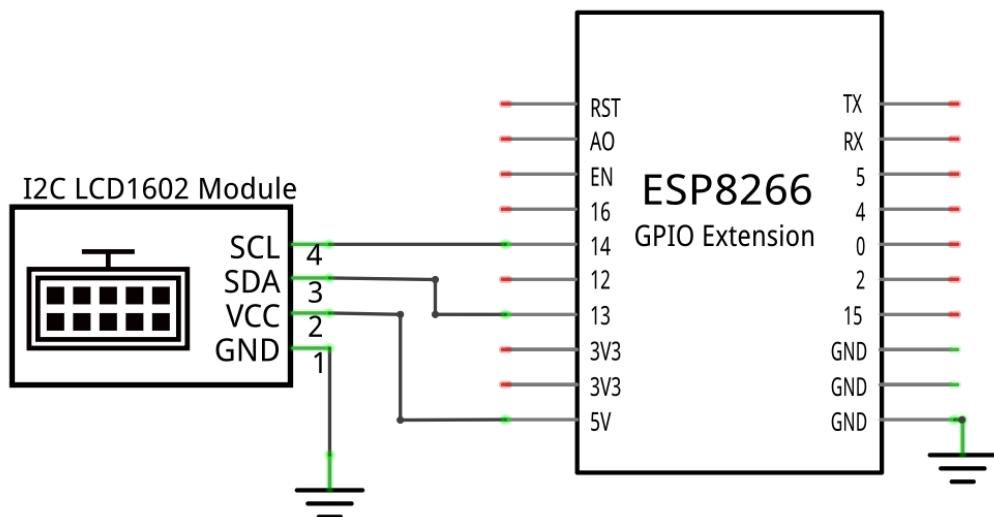
PCF8574 module pin and LCD1602 pin are corresponding to each other and connected with each other:



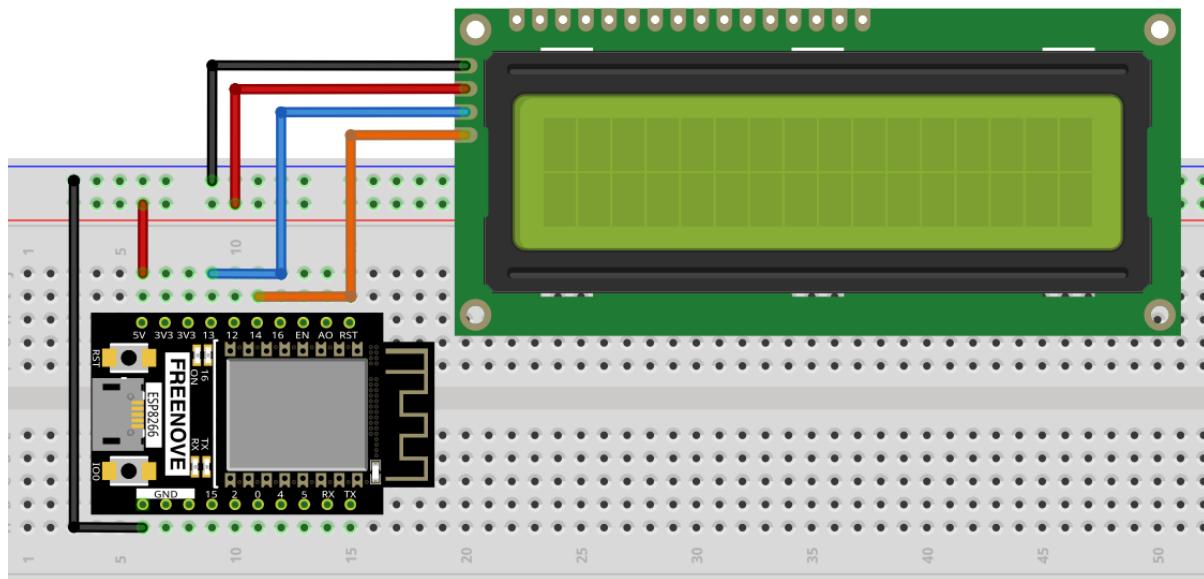
So we only need 4 pins to control the 16 pins of the LCD1602 display screen through the I₂C interface.
In this project, we will use the I₂C LCD1602 to display some static characters and dynamic variables.

Circuit

Schematic diagram



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

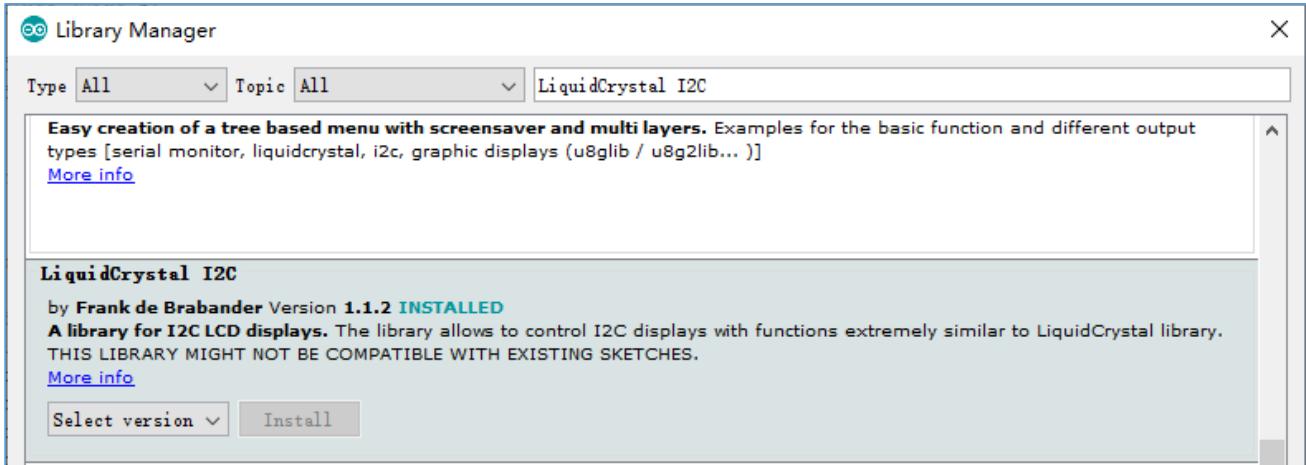




Sketch

How to install the library

We use the third party library LiquidCrystal I2C. If you haven't installed it yet, please do so before learning. The steps to add third-party Libraries are as follows: open arduino->Sketch->Include library-> Manage libraries. Enter " LiquidCrystal I2C " in the search bar and select " LiquidCrystal I2C " for installation.



Use I2C LCD 1602 to display characters and variables.

Sketch_17.1_Display_the_string_on_LCD1602

```
Sketch_20.1_Display_the_string_on_LCD1602 | Arduino 1.8.13 Hourly Build 2020/02/19 03:33
File Edit Sketch Tools Help
Sketch_20.1_Display_the_string_on_LCD1602
1 //*****
2   Filename : Drive LiquidCrystal I2C to display characters
3   Description : I2C is used to control the display characters of LCD1602.
4   Author : www.freenove.com
5   Modification: 2020-3-4
6 ****
7 #include <LiquidCrystal_I2C.h>
8 #include <Wire.h>
9
10 #define SDA 13           //Define SDA pins
11 #define SCL 14           //Define SCL pins
12
13 /*
14   * note:If lcd1602 uses PCF8574T, IIC's address is 0x27,
15   *       or lcd1602 uses PCF8574AT, IIC's address is 0x3F.
16 */
17 LiquidCrystal_I2C lcd(0x27,16,2);
18 void setup() {
19   Wire.begin(SDA, SCL);          // attach the IIC pin
20   lcd.init();                   // LCD driver initialization
21   lcd.backlight();              // Open the backlight
22   lcd.setCursor(0,0);           // Move the cursor to row 0, column 0
23   lcd.print("hello, world!");    // The print content is displayed on the LCD
24 }
25
26 void loop() {
27   lcd.setCursor(0,1);           // Move the cursor to row 1, column 0
28   lcd.print("Counter:");        // The count is displayed every second
29   lcd.print(millis() / 1000);
30   delay(1000);
31 }
```

Done uploading.

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

Compile and upload the code to ESP8266 and the LCD1602 displays characters.





If you cannot see anything on the display or the display is not clear, try rotating the white knob on back of LCD1602 slowly, which adjusts the contrast, until the screen can display clearly.



The following is the program code:

```

1 #include <LiquidCrystal_I2C.h>
2
3
4 #define SDA 13           //Define SDA pins
5 #define SCL 14           //Define SCL pins
6
7 /*
8  * note: If lcd1602 uses PCF8574T, IIC's address is 0x27,
9  *       or lcd1602 uses PCF8574AT, IIC's address is 0x3F.
10 */
11 LiquidCrystal_I2C lcd(0x27, 16, 2);
12
13 void setup() {
14     Wire.begin(SDA, SCL);          // attach the IIC pin
15     lcd.init();                   // LCD driver initialization
16     lcd.backlight();              // Turn on the backlight
17     lcd.setCursor(0, 0);          // Move the cursor to row 0, column 0
18     lcd.print("hello, world! ");
19 }
20
21 void loop() {
22     lcd.setCursor(0, 1);          // Move the cursor to row 1, column 0
23     lcd.print("Counter:");
24     lcd.print(millis() / 1000);
25     delay(1000);
26 }
```

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

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

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

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

Initialize I2C and set its pins as 13,14. And then initialize LCD1602 and turn on the backlight of LCD.

```
14 Wire.begin(SDA, SCL);           // attach the IIC pin
15 lcd.init();                   // LCD driver initialization
16 lcd.backlight();             // Turn on the backlight
```

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

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

Print the number on the second line of LCD1602.

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

Reference

class LiquidCrystal

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

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

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

```
init();
```

Initializes the Lcd1602's device

```
backlight();
```

Turn on Lcd1602's backlight.

```
setCursor(column, row);
```

Sets the screen's column and row.

column: The range is 0 to 15.

row: The range is 0 to 1.

```
print(String);
```

Print the character string on Lcd1602

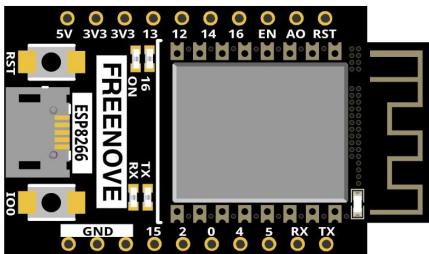
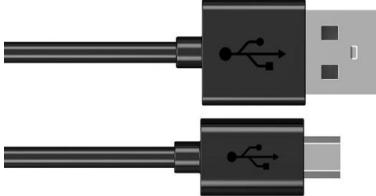
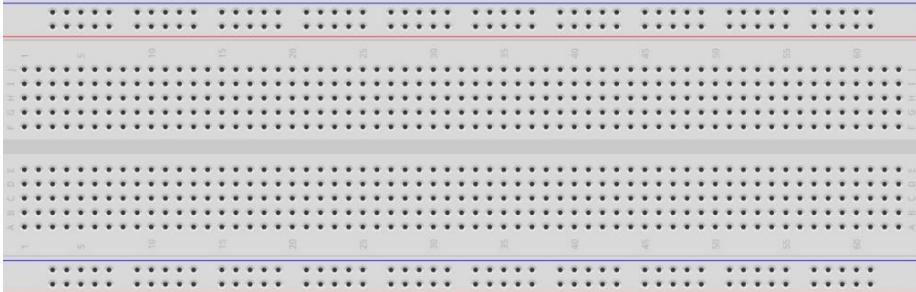
Chapter 18 Ultrasonic Ranging

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

Project 18.1 Ultrasonic Ranging

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

Component List

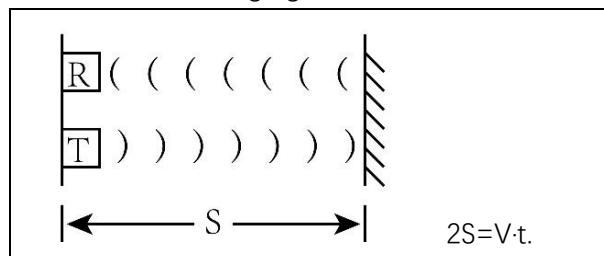
ESP8266 x1	USB cable
	
Breadboard x1	
Jumper wire F/M x4	HC SR04 x1
	

Component Knowledge

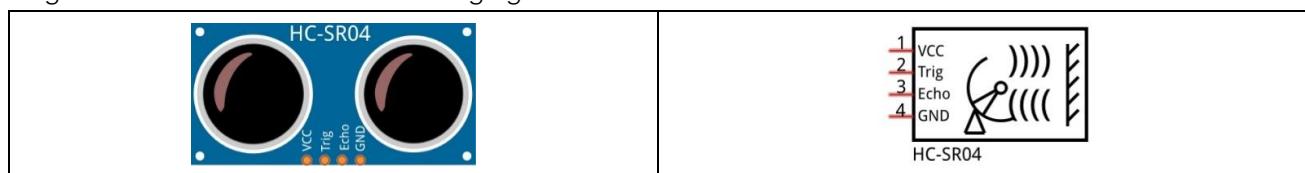
The ultrasonic ranging module uses the principle that ultrasonic waves will be sent back when encounter obstacles. We can measure the distance by counting the time interval between sending and receiving of the

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

ultrasonic waves, and the time difference is the total time of the ultrasonic wave's journey from being transmitted to being received. Because the speed of sound in air is a constant, about $v=340\text{m/s}$, we can calculate the distance between the ultrasonic ranging module and the obstacle: $s=vt/2$.



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



Pin description:

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

Technical specs:

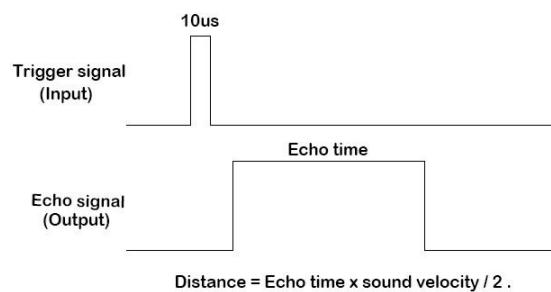
Working voltage: 5V

Working current: 12mA

Minimum measured distance: 2cm

Maximum measured distance: 200cm

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

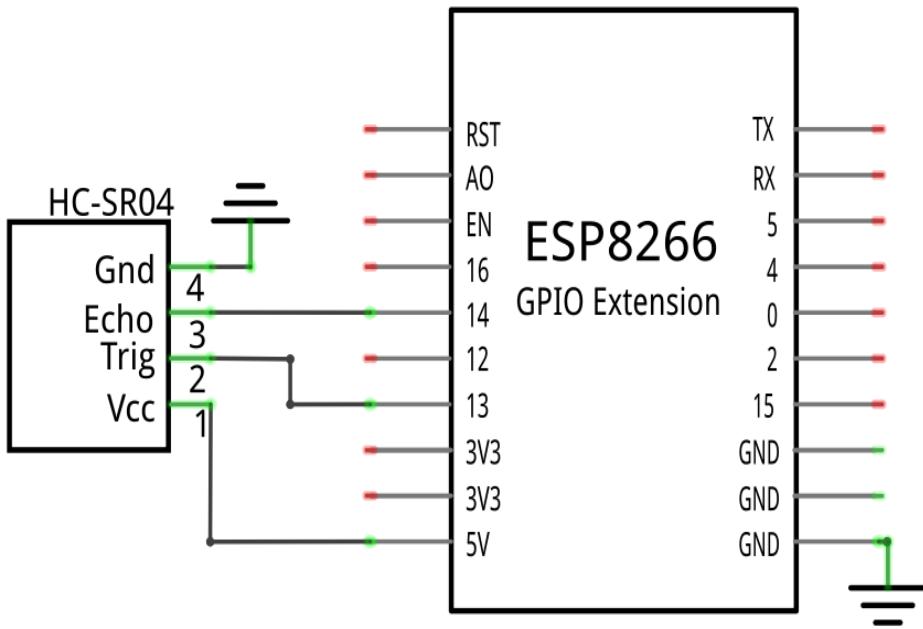


Circuit

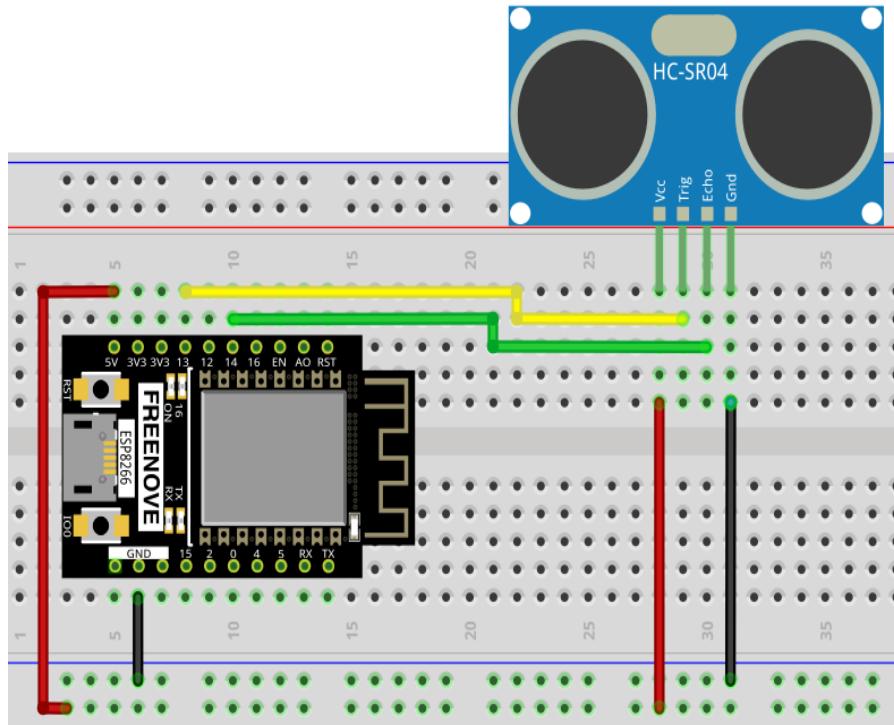
Note that the voltage of ultrasonic module is 5V in the circuit.

Schematic diagram

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



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



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

Sketch

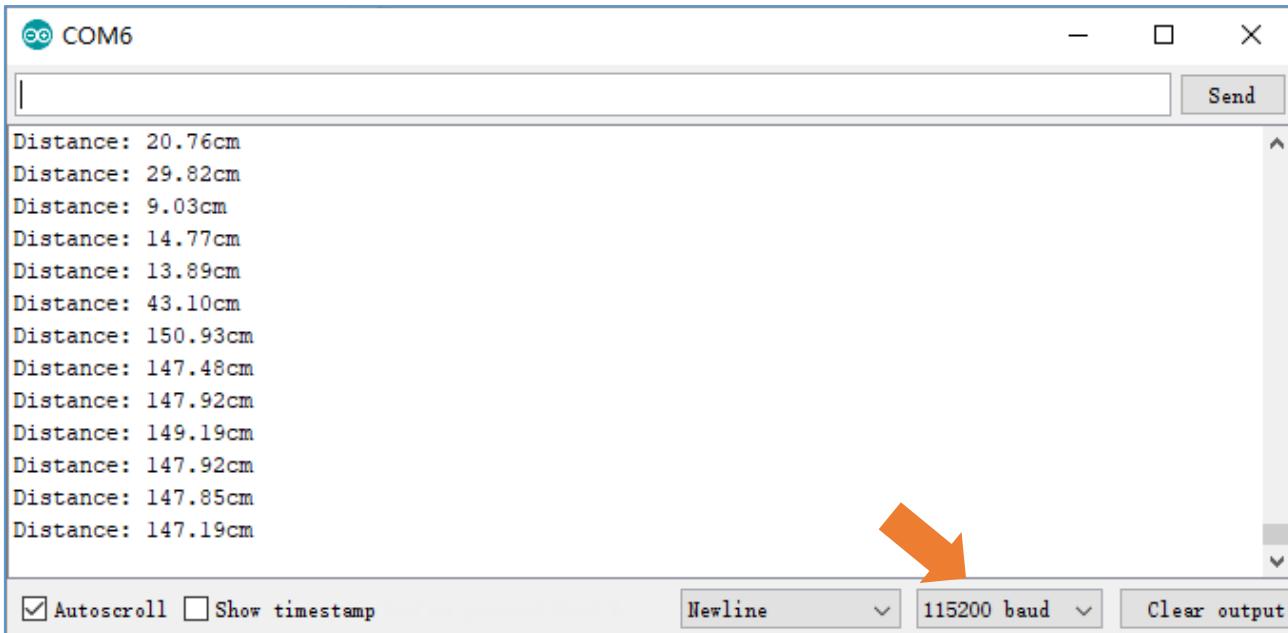
Sketch_18.1_Ultrasonic_Ranging

The screenshot shows the Arduino IDE interface with the following details:

- Title Bar:** Sketch_21.1_Ultrasonic_Ranging | Arduino 1.8.13 Hourly Build 2020/02/19 03:33
- Menu Bar:** File Edit Sketch Tools Help
- Toolbar:** Standard toolbar icons for file operations.
- Sketch Area:** Displays the C++ code for the sketch. The code defines pins for trigPin (13) and echoPin (14), sets MAX_DISTANCE to 700 cm, and defines soundVelocity as 340 m/s. It includes setup() and loop() functions for pinging and printing distance results via Serial. A getSonar() function handles the pulse measurement and calculation.
- Status Bar:** Shows "Done Saving.", "Leaving...", "Hard resetting via RTS pin...", and "ESP32 Wrover Module on COM6".
- Page Number:** 30

```
1 // ****
2   Filename      : Ultrasonic Ranging
3   Description   : Use the ultrasonic module to measure the distance.
4   Author        : www.freenove.com
5   Modification  : 2020-3-4
6 ****
7 #define trigPin 13 // define TrigPin
8 #define echoPin 14 // define EchoPin.
9 #define MAX_DISTANCE 700 // Maximum sensor distance is rated at 400-500cm.
10 //timeOut= 2*MAX_DISTANCE /100 /340 *1000000 = MAX_DISTANCE*58.8
11 float timeOut = MAX_DISTANCE * 60;
12 int soundVelocity = 340; // define sound speed=340m/s
13
14 void setup() {
15   pinMode(trigPin,OUTPUT);// set trigPin to output mode
16   pinMode(echoPin,INPUT); // set echoPin to input mode
17   Serial.begin(115200); // Open serial monitor at 115200 baud to see ping results.
18 }
19
20 void loop() {
21   delay(100); // Wait 100ms between pings (about 20 pings/sec).
22   Serial.print("Distance: ");
23   Serial.print(getSonar()); // Send ping, get distance in cm and print result
24   Serial.println("cm");
25 }
26
27 float getSonar() {
28   unsigned long pingTime;
29   float distance;
30   // make trigPin output high level lasting for 10us to trigger HC_SR04
31   digitalWrite(trigPin, HIGH);
32   delayMicroseconds(10);
33   digitalWrite(trigPin, LOW);
34   // Wait HC-SR04 returning to the high level and measure out this waiting time
35   pingTime = pulseIn(echoPin, HIGH, timeOut);
36   // calculate the distance according to the time
37   distance = (float)pingTime * soundVelocity / 2 / 10000;
38   return distance; // return the distance value
39 }
```

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:



The following is the program code:

```

1 #define trigPin 13 // define trigPin
2 #define echoPin 14 // define echoPin.
3 #define MAX_DISTANCE 700 // Maximum sensor distance is rated at 400~500cm.
4 //timeOut= 2*MAX_DISTANCE /100 /340 *1000000 = MAX_DISTANCE*58.8
5 float timeOut = MAX_DISTANCE * 60;
6 int soundVelocity = 340; // define sound speed=340m/s
7
8 void setup() {
9     pinMode(trigPin, OUTPUT); // set trigPin to output mode
10    pinMode(echoPin, INPUT); // set echoPin to input mode
11    Serial.begin(115200); // Open serial monitor at 115200 baud to see ping results.
12 }
13
14 void loop() {
15     delay(100); // Wait 100ms between pings (about 20 pings/sec).
16     Serial.printf("Distance: ");
17     Serial.print(getSonar()); // Send ping, get distance in cm and print result
18     Serial.println("cm");
19 }
20
21 float getSonar() {
22     unsigned long pingTime;
23     float distance;
24     // make trigPin output high level lasting for 10us to trigger HC_SR04
25     digitalWrite(trigPin, HIGH);

```

```

26 delayMicroseconds(10);
27 digitalWrite(trigPin, LOW);
28 // Wait HC-SR04 returning to the high level and measure out this waiting time
29 pingTime = pulseIn(echoPin, HIGH, timeOut);
30 // calculate the distance according to the time
31 distance = (float)pingTime * soundVelocity / 2 / 10000;
32 return distance; // return the distance value
33 }
```

First, define the pins and the maximum measurement distance.

```

1 #define trigPin 13 // define trigPin
2 #define echoPin 14 // define echoPin.
3 #define MAX_DISTANCE 700 // Maximum sensor distance is rated at 400~500cm.
```

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

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

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

```

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

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

```

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



About function **pulseIn()**:

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

pin: the number of the Arduino pin on which you want to read the pulse. Allowed data types: int.

value: type of pulse to read: either HIGH or LOW. Allowed data types: int.

timeout (optional): the number of microseconds to wait for the pulse to start; default is one second.

Project 18.2 Ultrasonic Ranging

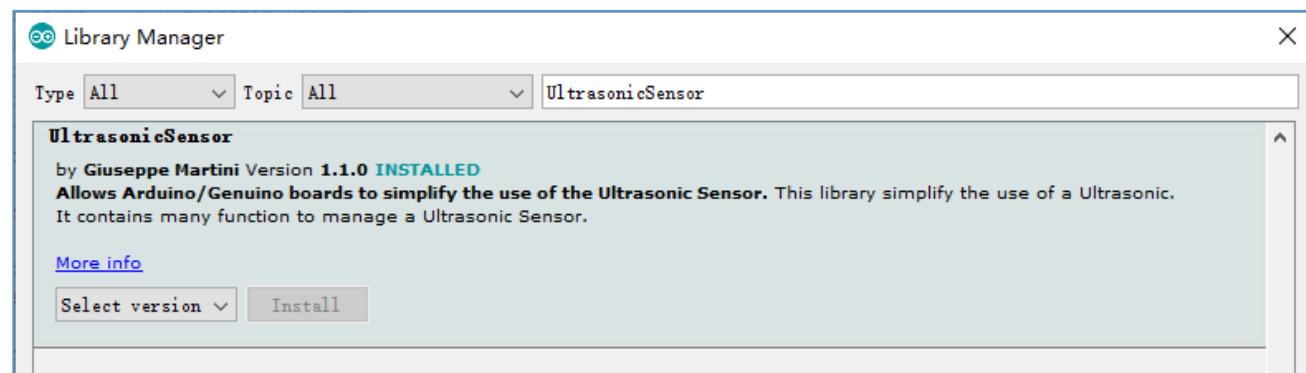
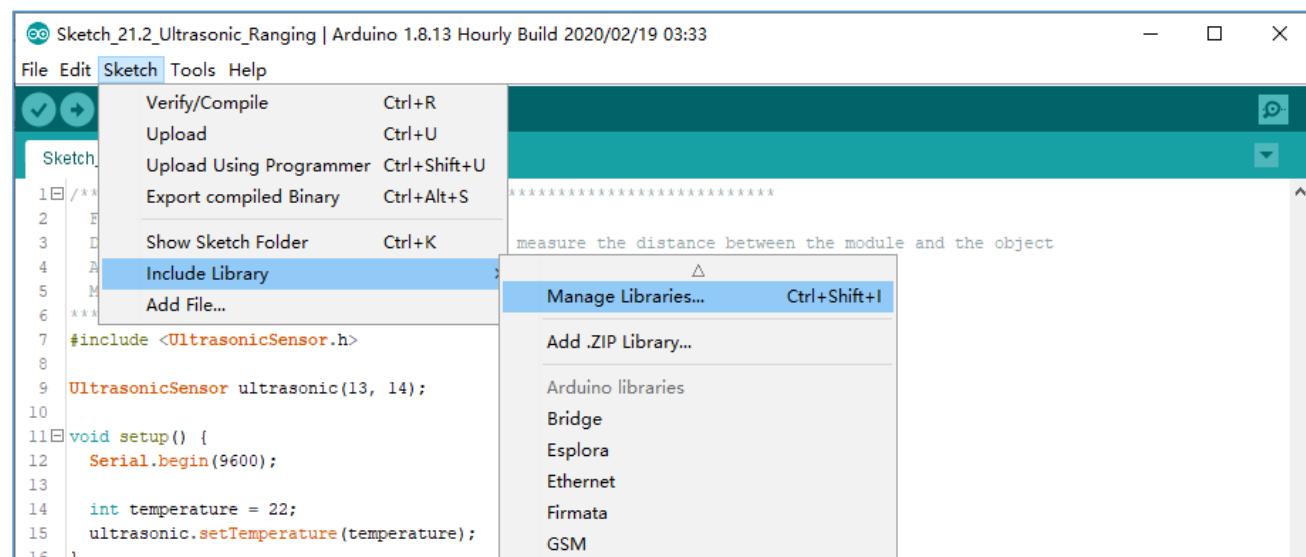
Component List and Circuit

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

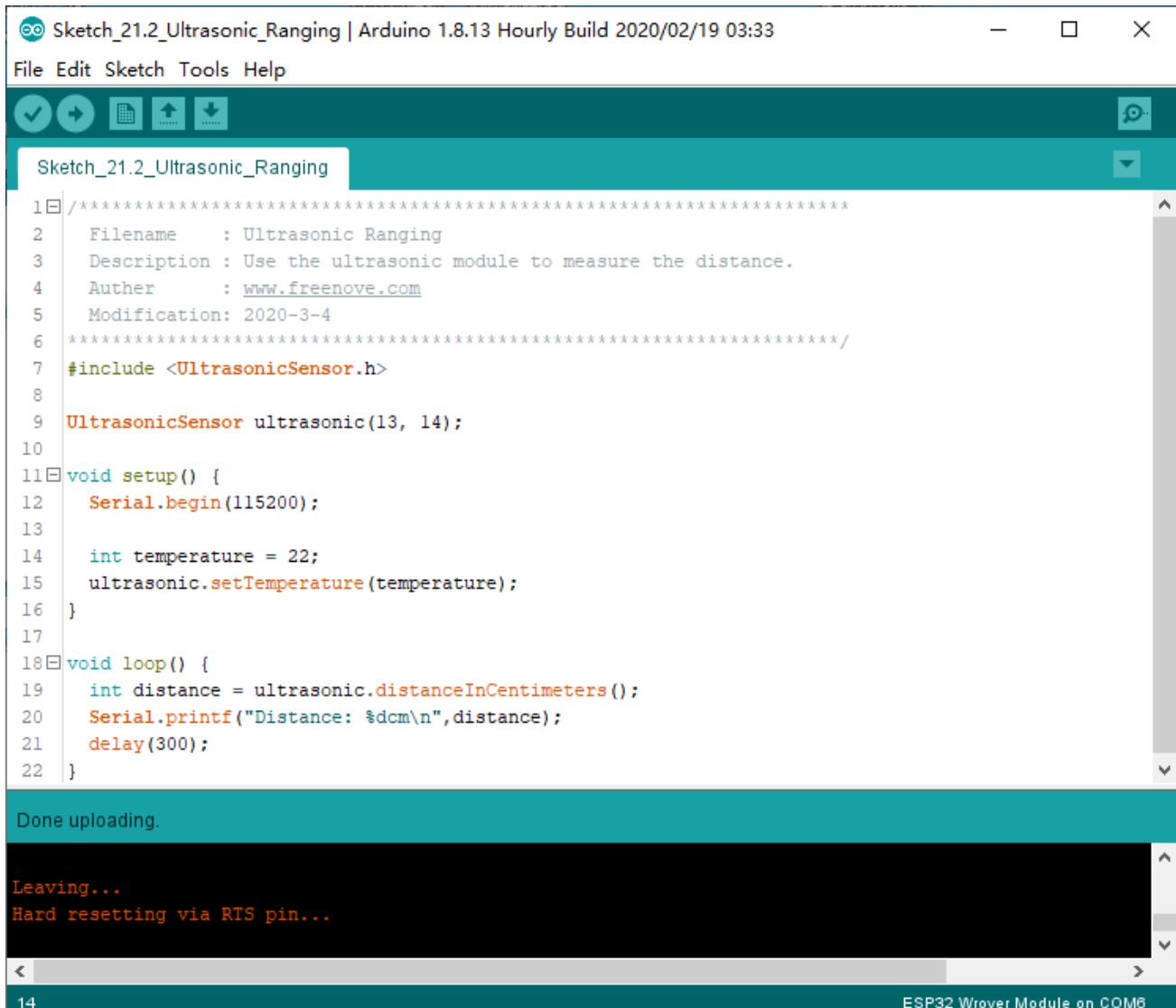
Sketch

How to install the library

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



Sketch_18.2_Ultrasonic_Ranging



```

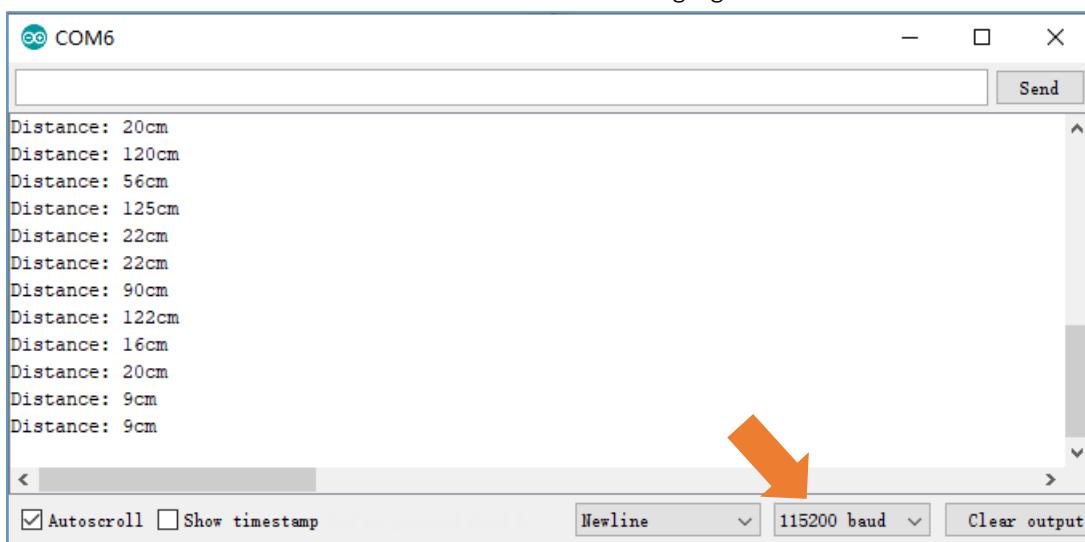
1 // ****
2   Filename    : Ultrasonic Ranging
3   Description : Use the ultrasonic module to measure the distance.
4   Author     : www.freenove.com
5   Modification: 2020-3-4
6 ****
7 #include <UltrasonicSensor.h>
8
9 UltrasonicSensor ultrasonic(13, 14);
10
11 void setup() {
12   Serial.begin(115200);
13
14   int temperature = 22;
15   ultrasonic.setTemperature(temperature);
16 }
17
18 void loop() {
19   int distance = ultrasonic.distanceInCentimeters();
20   Serial.printf("Distance: %dcm\n",distance);
21   delay(300);
22 }

```

Done uploading.

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

Download the code to ESP8266, open the serial port monitor, set the baud rate to 115200. Use the ultrasonic module to measure distance. As shown in the following figure:



Distance: 20cm
 Distance: 120cm
 Distance: 56cm
 Distance: 125cm
 Distance: 22cm
 Distance: 22cm
 Distance: 90cm
 Distance: 122cm
 Distance: 16cm
 Distance: 20cm
 Distance: 9cm
 Distance: 9cm

Autoscroll Show timestamp Newline 115200 baud Clear output

The following is the program code:

```

1 #include <UltrasonicSensor.h>
2 //Attach the trigger and echo pins to pins 13 and 14 of esp8266
3 UltrasonicSensor ultrasonic(13, 14);
4
5 void setup() {
6     Serial.begin(115200);
7     //set the speed of sound propagation according to the temperature to reduce errors
8     int temperature = 22; //Setting ambient temperature
9     ultrasonic.setTemperature(temperature);
10 }
11
12 void loop() {
13     int distance = ultrasonic.distanceInCentimeters();
14     Serial.printf("Distance: %dcm\n", distance);
15     delay(300);
16 }
```

First, add UltrasonicSensor library.

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

Define an ultrasonic object and associate the pins.

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

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

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

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

```

12 void loop() {
13     int distance = ultrasonic.distanceInCentimeters();
14     Serial.printf("Distance: %dcm\n", distance);
15     delay(300);
16 }
```

Reference

class UltrasonicSensor

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

UltrasonicSensor ultrasonic(13, 14);

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

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

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

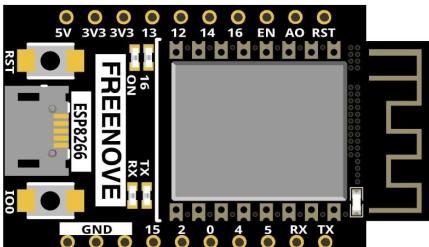
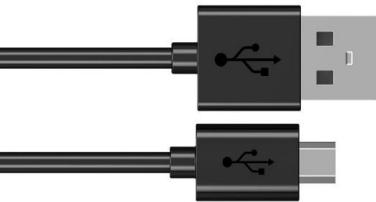
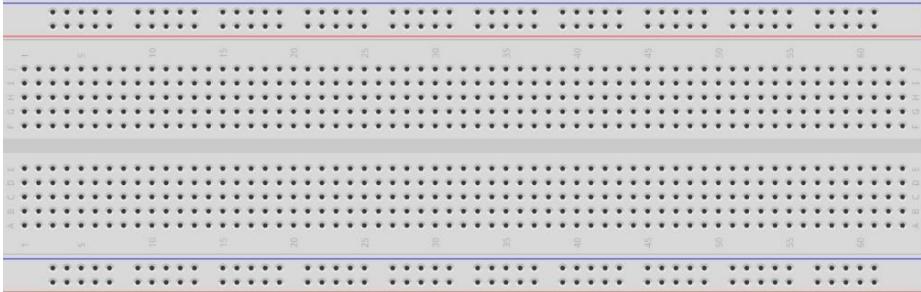
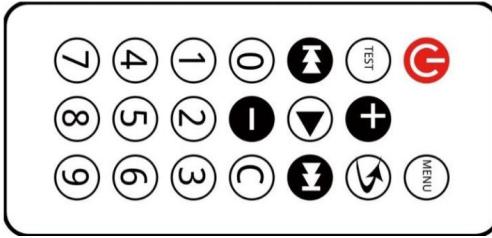
Chapter 19 Infrared Remote

In this chapter, we'll learn how to use an infrared remote control, and control a LED.

Project 19.1 Infrared Remote Control

First, we need to understand how infrared remote control works, then get the command sent from infrared remote control.

Component List

ESP8266 x1	USB cable
	
Breadboard x1	
Jumper wire M/M x6	Infrared Remote x1 (May need CR2025 battery x1, please check the holder)
	
Infrared Remote x1	Resistor 10kΩ x1

Component knowledge

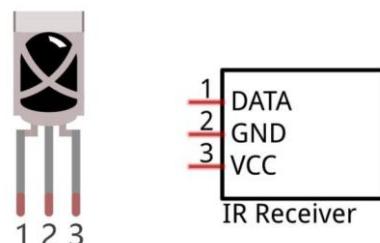
Infrared Remote

An infrared(IR) remote control is a device with a certain number of buttons. Pressing down different buttons will make the infrared emission tube, which is located in the front of the remote control, send infrared ray with different command. Infrared remote control technology is widely used in electronic products such as TV, air conditioning, etc. Thus making it possible for you to switch TV programs and adjust the temperature of the air conditioning when away from them. The remote control we use is shown below:



Infrared receiver

An infrared(IR) receiver is a component which can receive the infrared light, so we can use it to detect the signal emitted by the infrared remote control. DATA pin here outputs the received infrared signal.





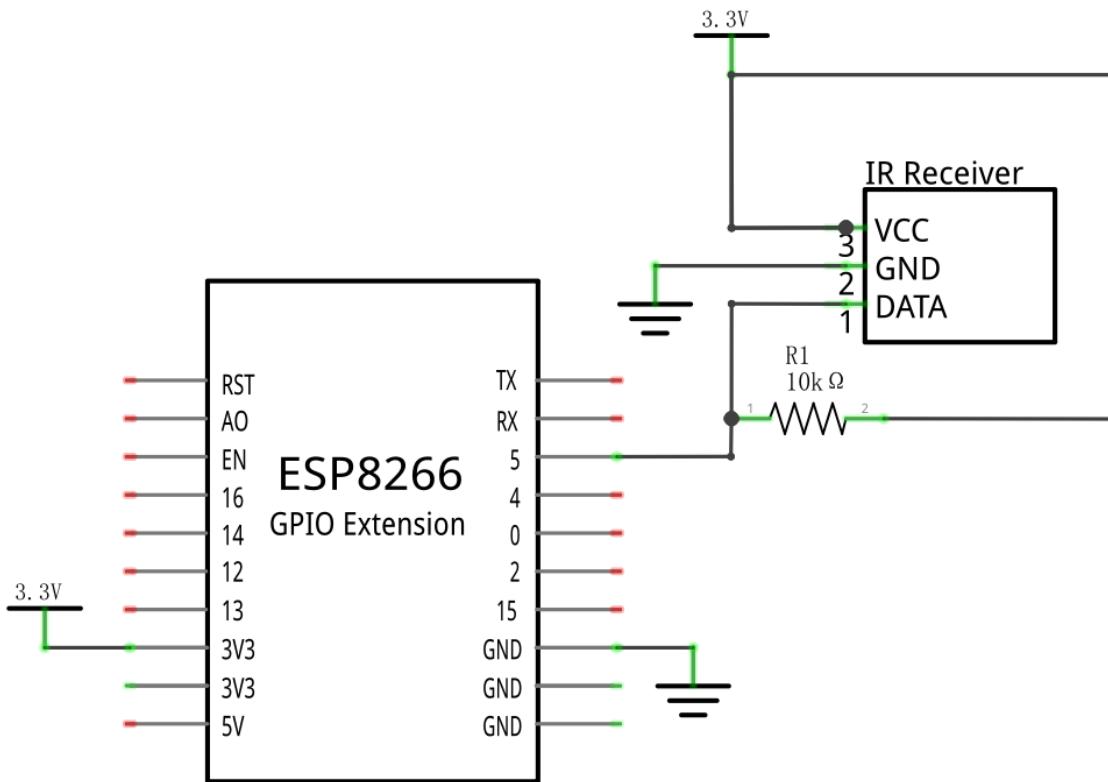
When you use the infrared remote control, the infrared remote control sends a key value to the receiving circuit according to the pressed keys. We can program the ESP8266 to do things like lighting, when a key value is received.

The following is the key value that the receiving circuit will receive when each key of the infrared remote control is pressed.

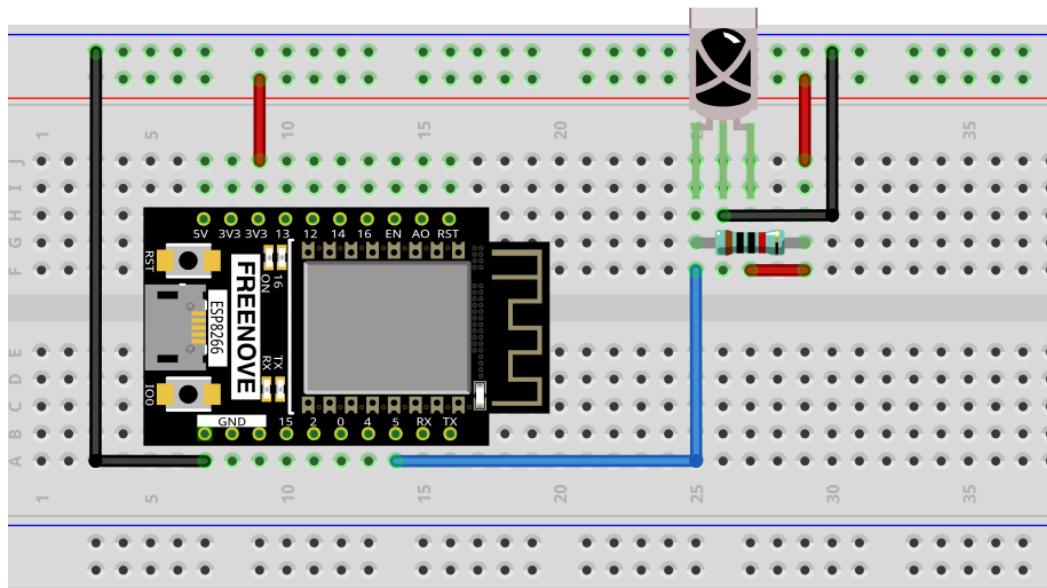
ICON	KEY Value	ICON	KEY Value
	FFA25D		FFB04F
	FFE21D		FF30CF
	FF22DD		FF18E7
	FF02FD		FF7A85
	FFC23D		FF10EF
	FFE01F		FF38C7
	FFA857		FF5AA5
	FF906F		FF42BD
	FF6897		FF4AB5
	FF9867		FF52AD

Circuit

Schematic diagram



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





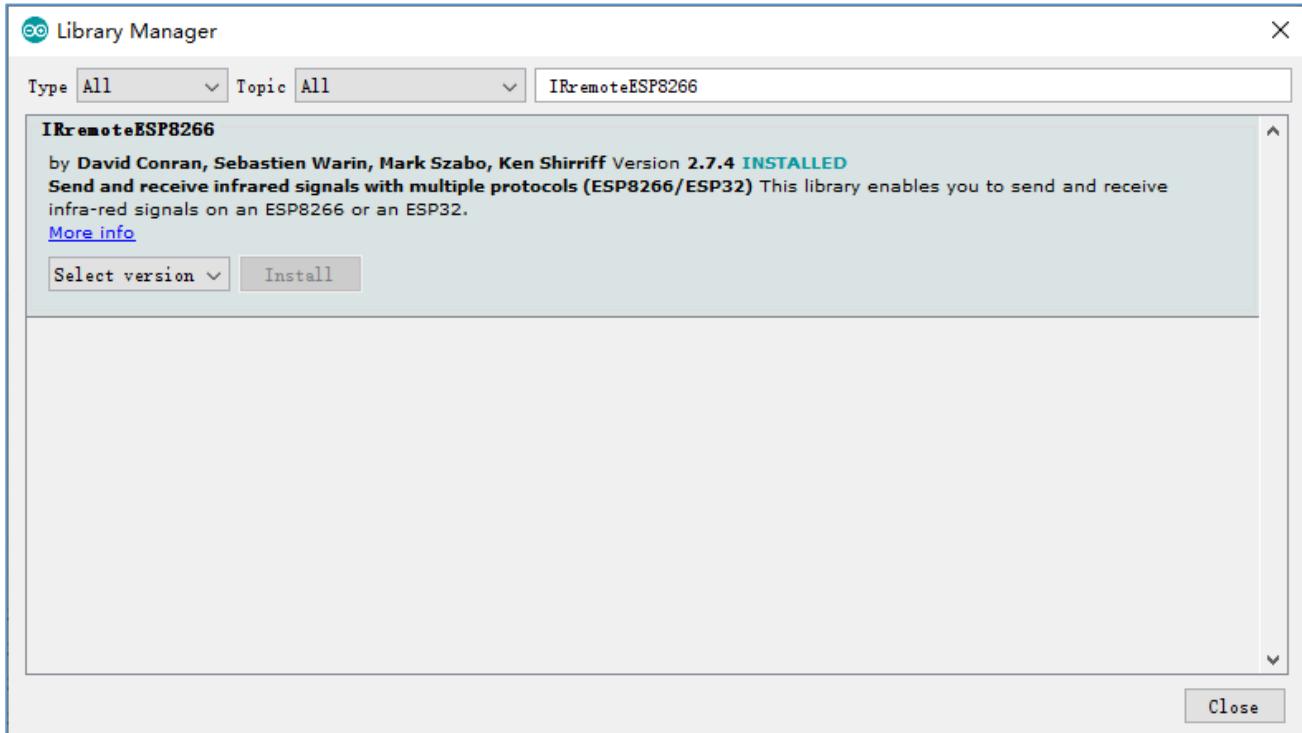
Sketch

This sketch uses the infrared receiving tube to receive the value sent from the infrared remote control, and print it out via the serial port.

How to install the library

We use the third party library IRremoteESP8266. If you haven't installed it yet, please do so first. The steps to add third-party Libraries are as follows: open arduino->Sketch->Include library-> Manage libraries. Enter "IRremoteESP8266" in the search bar and select " IRremoteESP8266" for installation.

Refer to the following operations:



Sketch_19.1_Infrared_Remote_Control

```
#include <Arduino.h>
#include <IRremoteESP8266.h>
#include <IRrecv.h>
#include <IRutils.h>

const uint16_t recvPin = 5; // Infrared receiving pin
IRrecv irrecv(recvPin); // Create a class object used to receive class
decode_results results; // Create a decoding results class object

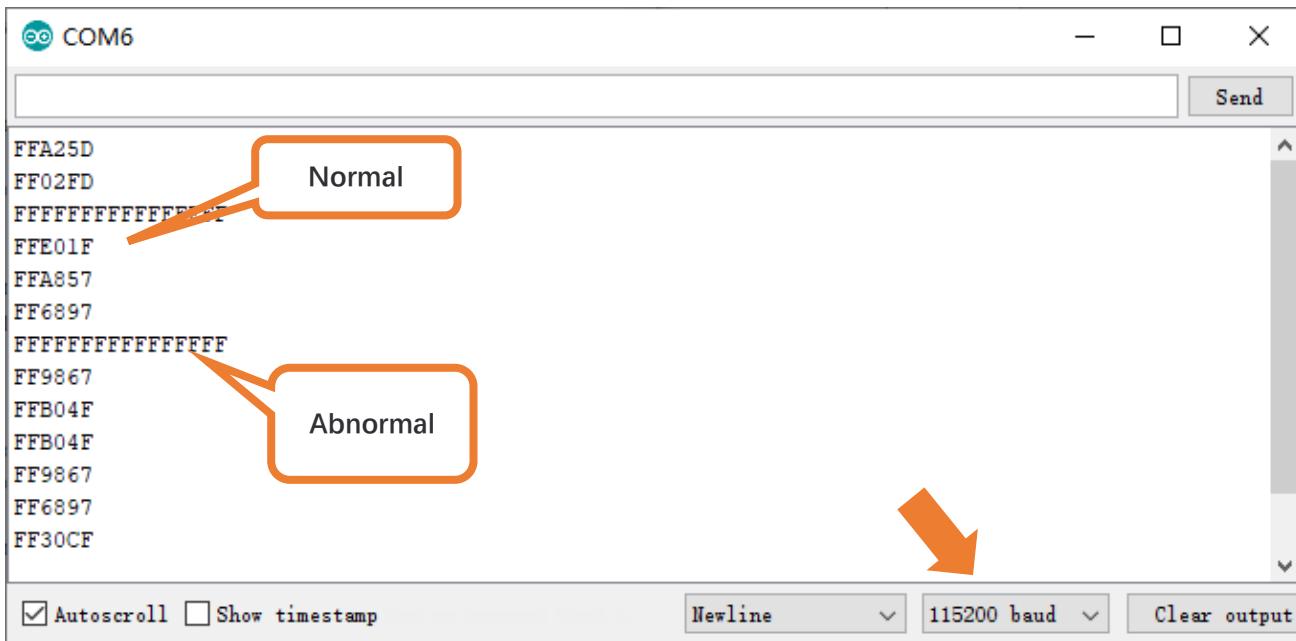
void setup() {
  Serial.begin(115200); // Initialize the serial port and set the baud rate to 115200
  irrecv.enableIRIn(); // Start the receiver
  Serial.print("IRrecvDemo is now running and waiting for IR message on Pin ");
  Serial.println(recvPin); //print the infrared receiving pin
}

void loop() {
  if (irrecv.decode(&results)) { // Waiting for decoding
    serialPrintUint64(results.value, HEX); // Print out the decoded results
    Serial.println("");
    irrecv.resume(); // Release the IRremote. Receive the next value
  }
  delay(1000);
}
```

Done uploading.

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

Download the code to ESP8266, open the serial port monitor, set the baud rate to 115200, press the IR remote control, the pressed keys value will be printed out through the serial port. As shown in the following figure: (Note that when the remote control button is pressed for a long time, the infrared receiving circuit receives a continuous high level, that is, it receives a hexadecimal "F")



The following is the program code:

```

1 #include <Arduino.h>
2 #include <IRremoteESP8266.h>
3 #include <IRrecv.h>
4 #include <IRUtils.h>
5
6 const uint16_t recvPin = 5; // Infrared receiving pin
7 IRrecv irrecv(recvPin); // Create a class object used to receive class
8 decode_results results; // Create a decoding results class object
9
10 void setup() {
11   Serial.begin(115200); // Initialize the serial port and set the baud rate to 115200
12   irrecv.enableIRIn(); // Start the receiver
13   while (! Serial) // Wait for the serial connection to be established.
14     delay(50);
15   Serial.println();
16   Serial.print("IRrecvDemo is now running and waiting for IR message on Pin ");
17   Serial.println(recvPin); //print the infrared receiving pin
18 }
19
20 void loop() {
21   if (irrecv.decode(&results)) { // Waiting for decoding
22     serialPrintUint64(results.value, HEX); // Print out the decoded results
23     Serial.println("");
24     irrecv.resume(); // Release the IRremote. Receive the next value
25   }
26   delay(1000);
27 }
```

First, include header file. Each time you use the infrared library, you need to include the header file at the beginning of the program.

```

1 #include <Arduino.h>
2 #include <IRremoteESP8266.h>
3 #include <IRrecv.h>
4 #include <IRUtils.h>
```

Second, define an infrared receive pin and associates it with the receive class. Apply a decode_results to decode the received infrared value.

```

6 const uint16_t RecvPin = 5; // Infrared receiving pin
7 IRrecv irrecv(RecvPin);      // Create a class object used to receive class
8 decode_results results;     // Create a decoding results class object
```

Third, enable infrared reception function, if you do not use this function, you won't receive the value from the infrared remote control.

```
12 irrecv.enableIRIn();           // Start the receiver
```

Finally, put the received data into the results class and print out the data through the serial port. Note that you must use **resume()** to release the infrared receive function every time when you receive data, otherwise you can only use the infrared receive function once and cannot receive the data next time.

```

20 void loop() {
21     if (irrecv.decode(&results)) {          // Waiting for decoding
22         serialPrintUint64(results.value, HEX); // Print out the decoded results
23         Serial.println("");
24         irrecv.resume();                  // Release the IRremote. Receive the next value
25     }
26     delay(1000);
27 }
```

Reference

class IRrecv You need to add the library each time you use the Infrared Reception.

IRrecv irrecv(Pin): Create a class object used to receive class, and associated with **Pin**.

enableIRIn(): Before using the infrared decoding function, enable the infrared receiving function. Otherwise the correct data will not be received.

decode(&results): Determine whether the infrared has received data, and if so, return true and store the data in the decode_results class. If no data is received, false is returned.

resume(): Release the IRremote. Or, the infrared reception and decoding function cannot be used again.

For more information about Infrared Remote Control, please visit:

<https://github.com/crankyoldgit/IRremoteESP8266/tree/master/src>



Project 19.2 Control LED through Infrared Remote

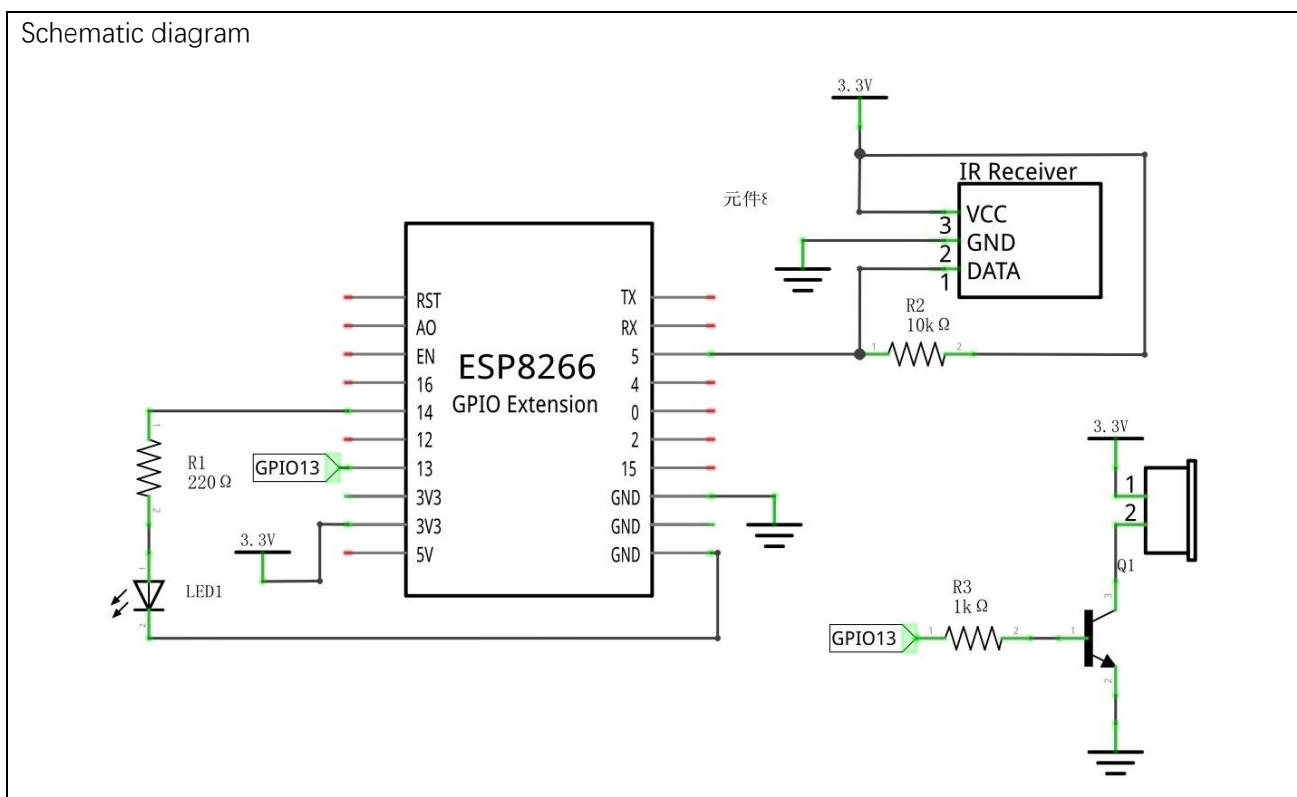
In this project, we will control the brightness of LED lights through an infrared remote control.

Component List

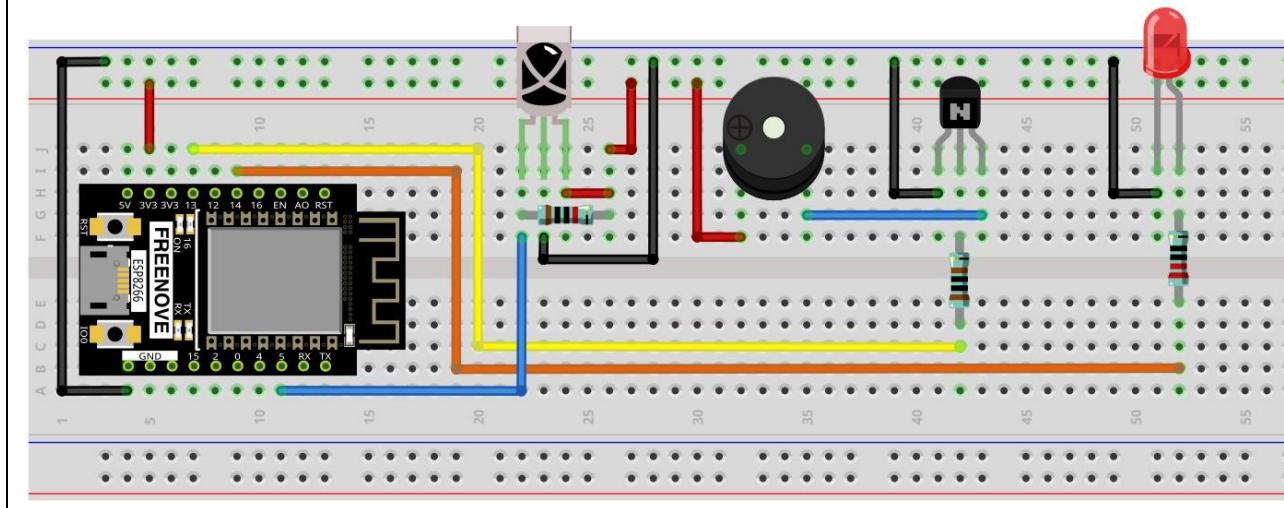
ESP8266 x1	USB cable
Breadboard x1	
Jumper wire M/M x12	Infrared Remote x1 (May need CR2025 battery x1, please check the holder)
	Resistor 220Ω x1
	Resistor 10kΩ x1



Circuit



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

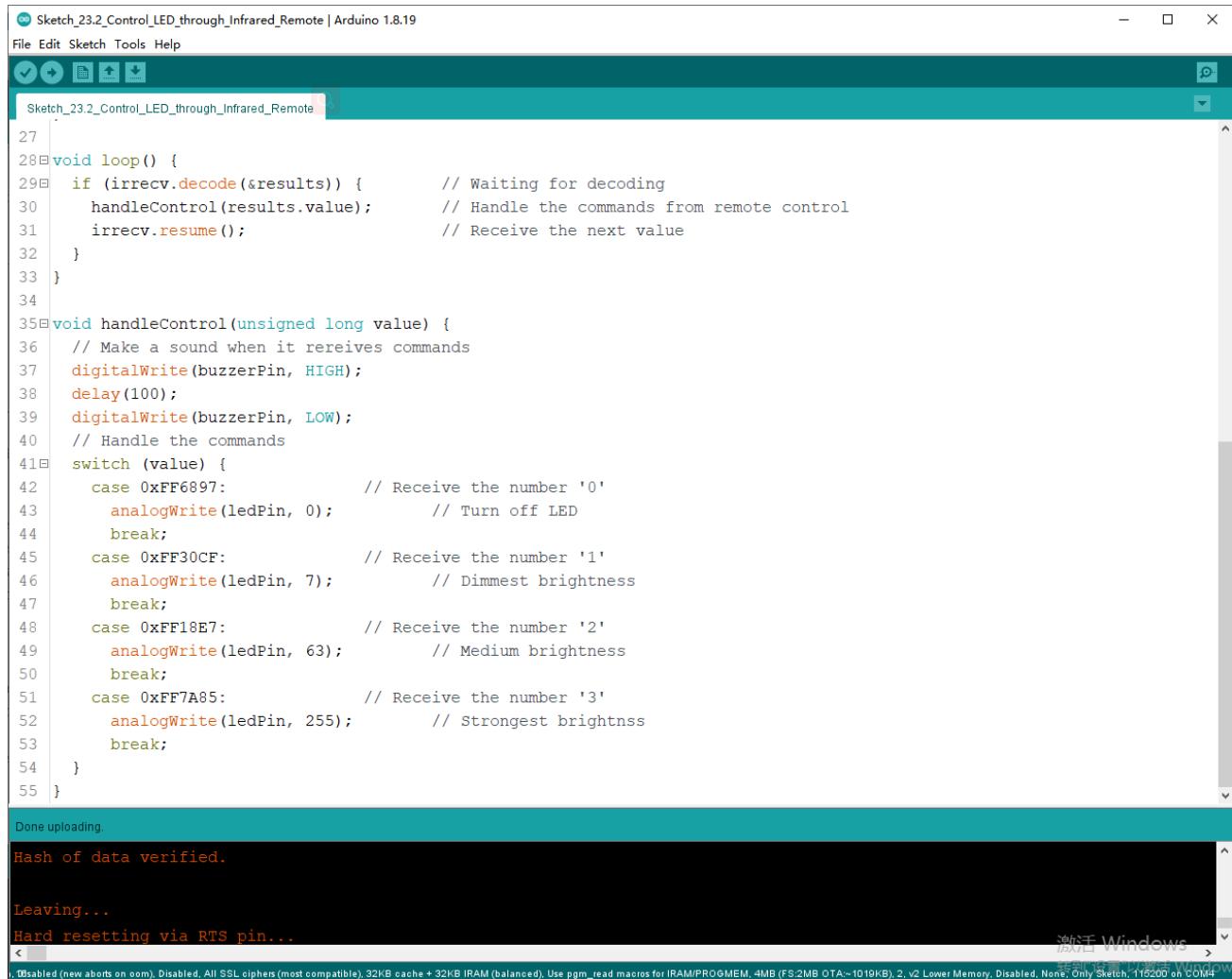


Any concerns? ✉ support@freenove.com

Sketch

The sketch controls the brightness of the LED by determining the key value of the infrared received.

Sketch_19.2_Control_LED_through_Infrared_Remote

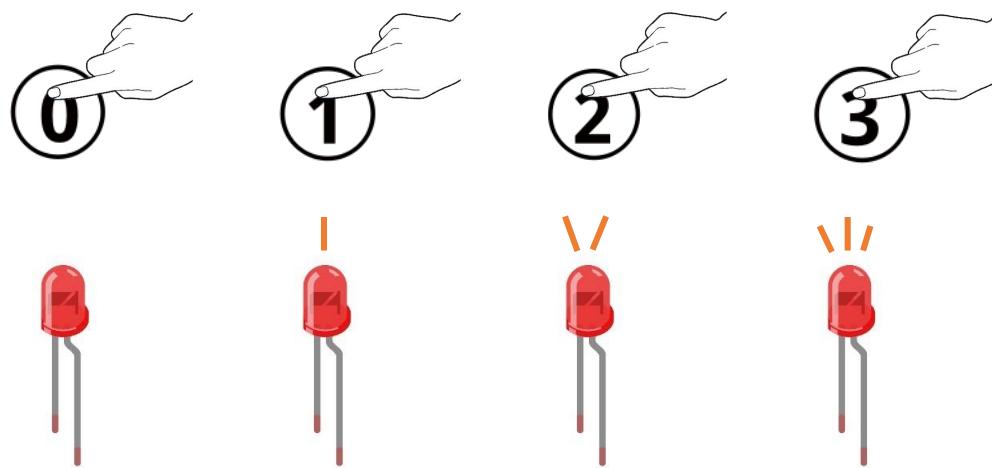


```
Sketch_23.2_Control_LED_through_Infrared_Remote | Arduino 1.8.19
File Edit Sketch Tools Help
Sketch_23.2_Control_LED_through_Infrared_Remote
27
28 void loop() {
29     if (irrecv.decode(&results)) {           // Waiting for decoding
30         handleControl(results.value);      // Handle the commands from remote control
31         irrecv.resume();                  // Receive the next value
32     }
33 }
34
35 void handleControl(unsigned long value) {
36     // Make a sound when it receives commands
37     digitalWrite(buzzerPin, HIGH);
38     delay(100);
39     digitalWrite(buzzerPin, LOW);
40     // Handle the commands
41     switch (value) {
42         case 0xFF6897:                   // Receive the number '0'
43             analogWrite(ledPin, 0);       // Turn off LED
44             break;
45         case 0xFF30CF:                   // Receive the number '1'
46             analogWrite(ledPin, 7);       // Dimmest brightness
47             break;
48         case 0xFF18E7:                   // Receive the number '2'
49             analogWrite(ledPin, 63);      // Medium brightness
50             break;
51         case 0xFF7A85:                   // Receive the number '3'
52             analogWrite(ledPin, 255);     // Strongest brightness
53             break;
54     }
55 }
```

Done uploading.
Hash of data verified.
Leaving...
Hard resetting via RTS pin...<

激活 Windows
安装设置以激活 Windows
禁用 (new aborts on oom). Disabled. All SSL ciphers (most compatible), 32KB cache + 32KB IRAM (balanced). Use pgm_read macros for IRAM/PROGMEM. 4MB (FS:2MB OTA:~1019KB), 2, v2 Lower Memory. Disabled, None, Only Sketch, 115200 on COM4

Compile and upload the code to the ESP8266. When pressing "0", "1", "2", "3" of the infrared remote control, the buzzer will sound once, and the brightness of the LED light will change correspondingly. rendering



The following is the program code:

```

1 #include <Arduino.h>
2 #include <IRremoteESP8266.h>
3 #include <IRrecv.h>
4 #include <IRUtils.h>
5
6 const uint16_t recvPin = 5; // Infrared receiving pin
7 IRrecv irrecv(recvPin); // Create a class object used to receive class
8 decode_results results; // Create a decoding results class object
9
10 int ledPin = 14; // the number of the LED pin
11 int buzzerPin = 13; // the number of the buzzer pin
12
13 void setup()
14 {
15     //Initialize the ledc configuration
16     pinMode(ledPin, OUTPUT); // set led pin into output mode
17     pinMode(buzzerPin, OUTPUT); // set buzzer pin into output mode
18     irrecv.enableIRIn(); // Start the receiver
19     map(0, 0, 255, 0, 1023);
20 }
21
22 void loop() {
23     if (irrecv.decode(&results)) { // Waiting for decoding
24         handleControl(results.value); // Handle the commands from remote control
25         irrecv.resume(); // Receive the next value
26     }
27 }
```

```

28 void handleControl(unsigned long value) {
29     // Make a sound when it receives commands
30     digitalWrite(buzzerPin, HIGH);
31     delay(100);
32     digitalWrite(buzzerPin, LOW);
33     // respond to the commands
34     switch (value) {
35         case 0xFF6897:           // Receive the number '0'
36             analogWrite(ledPin, 0);      // Turn off LED
37             break;
38         case 0xFF30CF:           // Receive the number '1'
39             analogWrite(ledPin, 7);      // Dimmest brightness
40             break;
41         case 0xFF18E7:           // Receive the number '2'
42             analogWrite(ledPin, 63);     // Medium brightness
43             break;
44         case 0xFF7A85:           // Receive the number '3'
45             analogWrite(ledPin, 255);    // Strongest brightness
46             break;
47     }
48 }
```

The handleControl() function is used to execute events corresponding to infrared code values. Every time when the function is called, the buzzer sounds once and determine the brightness of the LED based on the infrared key value. If the key value is not "0", "1", "2", "3", the buzzer sounds once, but the brightness of LED will not change.

```

28 void handleControl(unsigned long value) {
29     // Make a sound when it receives commands
30     digitalWrite(buzzerPin, HIGH);
31     delay(100);
32     digitalWrite(buzzerPin, LOW);
33     // respond to the commands
34     switch (value) {
35         case 0xFF6897:           // Receive the number '0'
36             analogWrite(ledPin, 0);      // Turn off LED
37             break;
38         case 0xFF30CF:           // Receive the number '1'
39             analogWrite(ledPin, 7);      // Dimmest brightness
40             break;
41         case 0xFF18E7:           // Receive the number '2'
42             analogWrite(ledPin, 63);     // Medium brightness
43             break;
44         case 0xFF7A85:           // Receive the number '3'
45             analogWrite(ledPin, 255);    // Strongest brightness
46             break;
47     }
48 }
```

```
47 }  
48 }
```

Each time when the command is received, the function above will be called in the loop() function.

```
22 void loop() {  
23     if (irrecv.decode(&results)) {           // Waiting for decoding  
24         handleControl(results.value);        // Handle the commands from remote control  
25         irrecv.resume();                      // Receive the next value  
26     }  
27 }
```



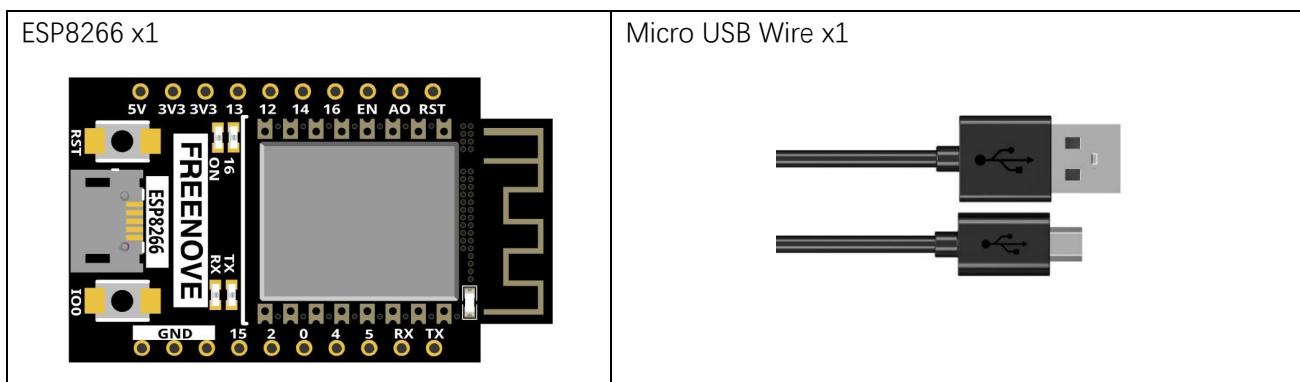
Chapter 20 WiFi Working Modes

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 20.1 Station mode

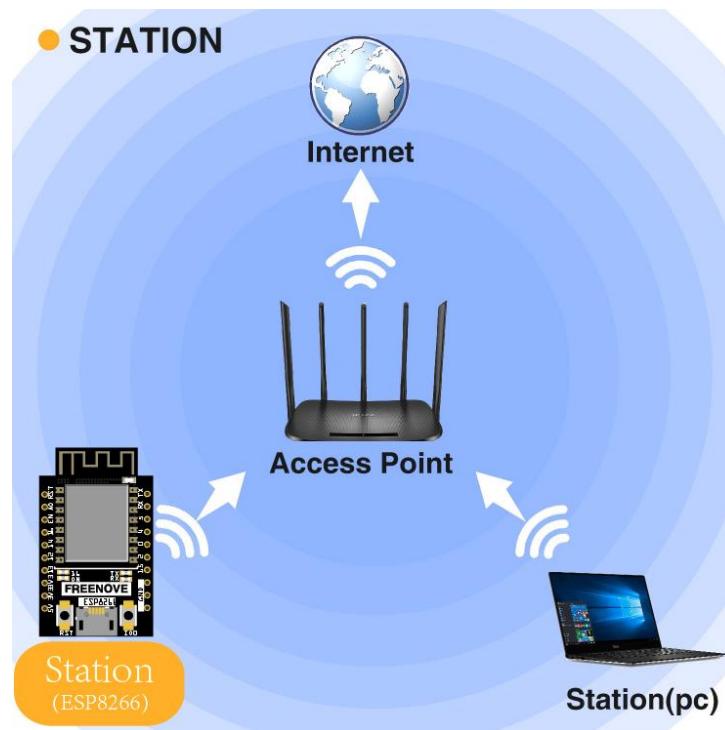
Component List



Component knowledge

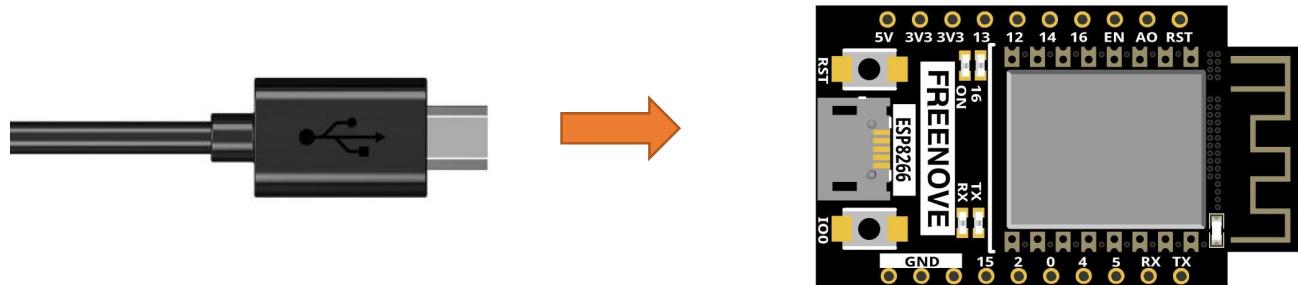
Station mode

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_20.1_Station_mode

```

Sketch_28.1_WiFi_Station | Arduino 1.8.19
File Edit Sketch Tools Help
Sketch_28.1_WiFi_Station §
4 Author : www.freenove.com
5 Modification: 2022/05/11
6 ****
7 #include <ESP8266WiFi.h>
8
9 const char *ssid_Router      = "*****"; //Enter the router name
10 const char *password_Router = "*****"; //Enter the router password
11
12 void setup(){
13     Serial.begin(74880);
14     delay(2000);
15     Serial.println("Setup start");
16     WiFi.begin(ssid_Router, password_Router);
17     Serial.println(String("Connecting to ")+ssid_Router);
18     while (WiFi.status() != WL_CONNECTED){
19         delay(500);
20         Serial.print(".");
21     }
22     Serial.println("\nConnected, IP address: ");
Done uploading.

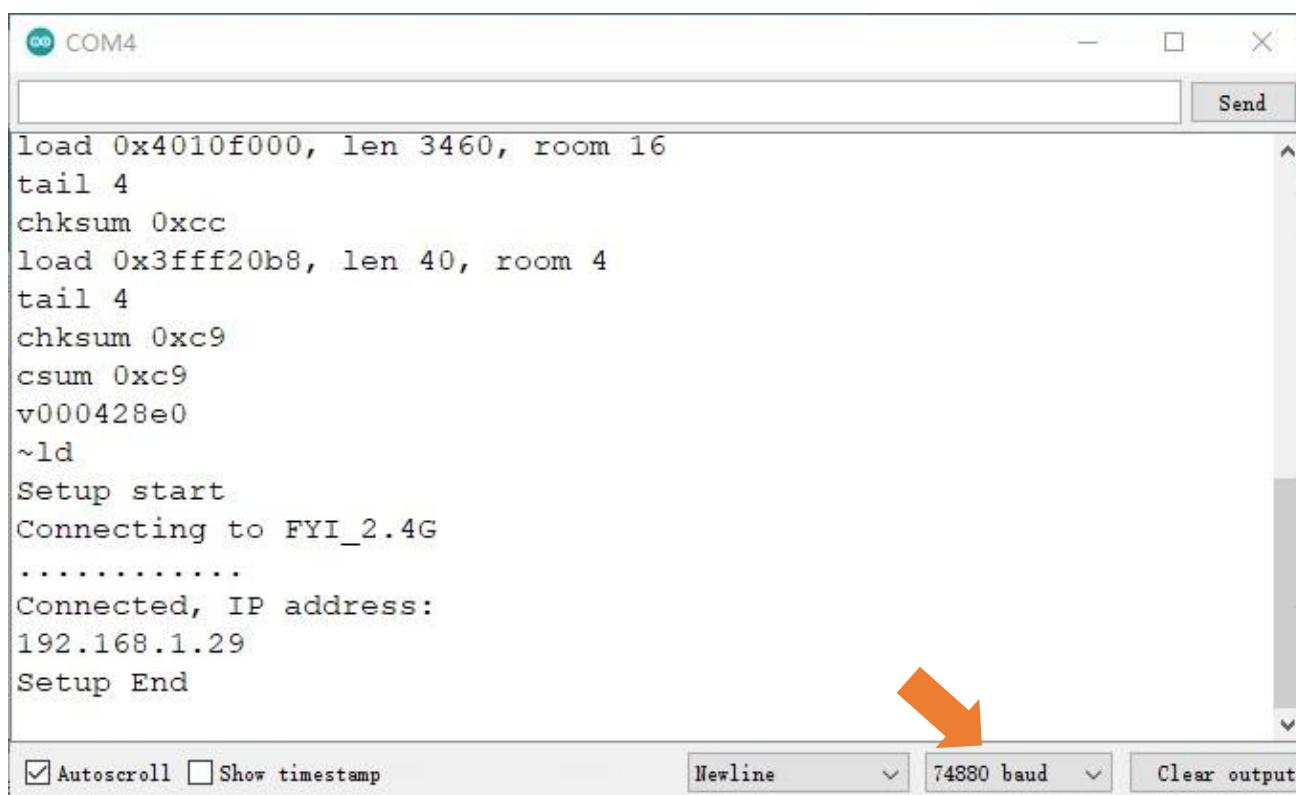
Leaving...
Hard resetting via RTS pin...

```

Bl5ache + 32KB IRAM (balanced), Use pgm_read macros for IRAM/PROGMEM, 4MB (FS:3MB OTA:~512KB), 2, v2 Lower Memory, Disabled, None, Only Sketch, 115200 on COM4

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:

```
1 #include <ESP8266WiFi.h>
2
3 const char *ssid_Router      = "*****"; //Enter the router name
4 const char *password_Router = "*****"; //Enter the router password
5
6 void setup() {
7     Serial.begin(74880);
8     delay(2000);
9     Serial.println("Setup start");
10    WiFi.begin(ssid_Router, password_Router);
11    Serial.println(String("Connecting to ") + ssid_Router);
12    while (WiFi.status() != WL_CONNECTED) {
13        delay(500);
14        Serial.print(".");
15    }
16    Serial.println("\nConnected, IP address: ");
17    Serial.println(WiFi.localIP());
18    Serial.println("Setup End");
19 }
20
21 void loop() {
```

```
22 }
```

Include the WiFi Library header file of ESP8266.

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

Enter correct router name and password.

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

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

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

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

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

Serial monitor prints out the IP address assigned to ESP8266

```
17 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: boolean 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(): obtain IP address in Station mode

disconnect(): disconnect wifi

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

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

Project 20.2 AP mode

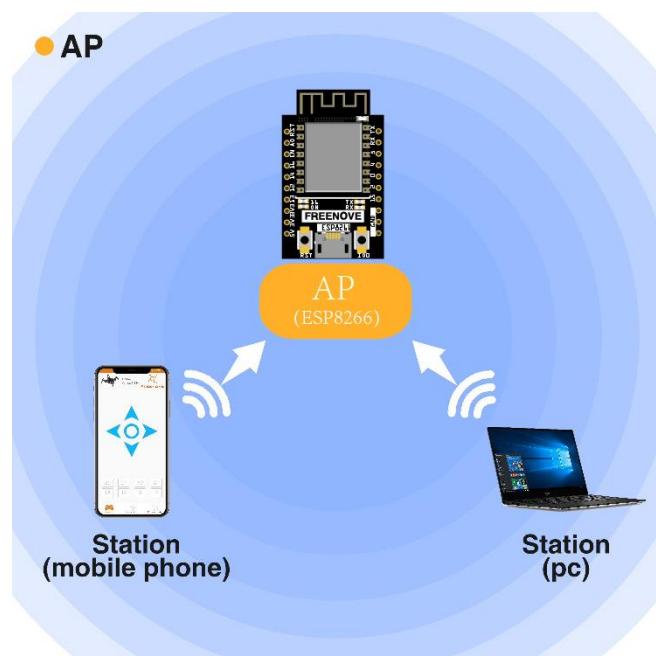
Component List & Circuit

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

Component knowledge

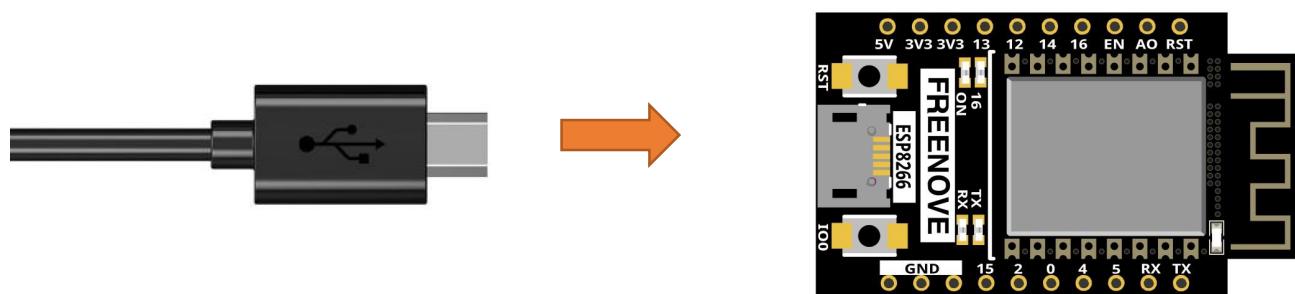
AP mode

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.



Any concerns? ✉ support@freenove.com

Sketch

Sketch_20.2_AP_mode

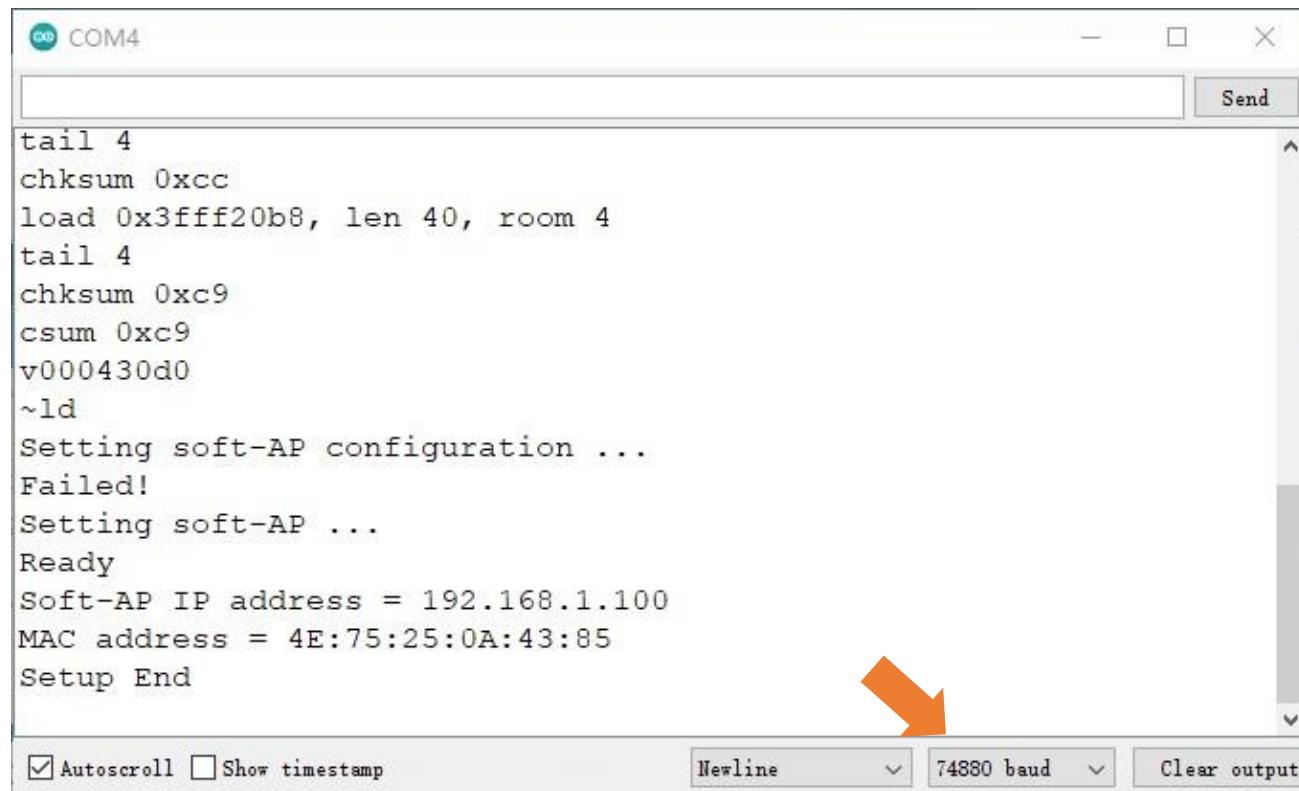
```
Sketch_28.2_WiFi_AP | Arduino 1.8.19
File Edit Sketch Tools Help
Sketch_28.2_WiFi_AP
1 // ****
2   Filename      : WiFi AP
3   Description   : Set ESP8266 AP
4   Author        : www.freenove.com
5   Modification: 2022/05/12
6 ****
7 #include <ESP8266WiFi.h>
8
9 const char *ssid_AP      = "WiFi_Name"; //Enter the router name
10 const char *password_AP = "12345678"; //Enter the router password
11
12 IPAddress local_IP(192,168,1,100); //Set the IP address of ESP8266 itself
13 IPAddress gateway(192,168,1,10); //Set the gateway of ESP8266 itself
14 IPAddress subnet(255,255,255,0); //Set the subnet mask for ESP32 itself
15
16 void setup(){
17   Serial.begin(74880);
18   delay(2000);
< | >
Done uploading.

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

Blaze + 32KB IRAM (balanced), Use pgm_read macros for IRAM/PROGMEM, 4MB (FS:3MB OTA:~512KB), 2, v2 Lower Memory, Disabled, None, Only Sketch, 115200 on COM4

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.

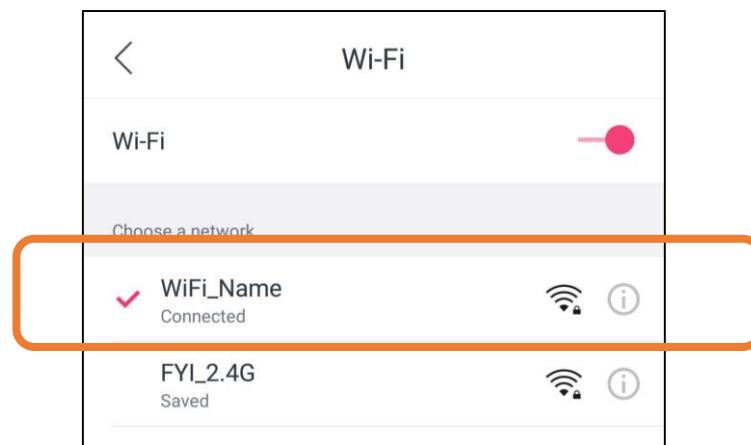


The screenshot shows a Windows-style serial monitor window titled "COM4". The text area contains the following log output:

```
tail 4
checksum 0xcc
load 0x3fff20b8, len 40, room 4
tail 4
checksum 0xc9
csum 0xc9
v000430d0
~ld
Setting soft-AP configuration ...
Failed!
Setting soft-AP ...
Ready
Soft-AP IP address = 192.168.1.100
MAC address = 4E:75:25:0A:43:85
Setup End
```

At the bottom, there are checkboxes for "Autoscroll" and "Show timestamp", and dropdown menus for "Newline" (set to "Newline") and "74880 baud" (set to "74880 baud"). A red arrow points from the text "Setup End" towards the "74880 baud" dropdown.

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:

```

1 #include <ESP8266WiFi.h>
2
3 const char *ssid_AP      = "WiFi_Name"; //Enter the router name
4 const char *password_AP = "12345678"; //Enter the router password
5
6 IPAddress local_IP(192, 168, 1, 100); //Set the IP address of ESP8266 itself
7 IPAddress gateway(192, 168, 1, 10); //Set the gateway of ESP8266 itself
8 IPAddress subnet(255, 255, 255, 0); //Set the subnet mask for ESP8266 itself
9
10 void setup() {
11     Serial.begin(74880);
12     delay(2000);
13     Serial.println("Setting soft-AP configuration ... ");
14     WiFi.disconnect();
15     WiFi.mode(WIFI_AP);
16     Serial.println(WiFi.softAPConfig(local_IP, gateway, subnet) ? "Ready" : "Failed!");
17     Serial.println("Setting soft-AP ... ");
18     boolean result = WiFi.softAP(ssid_AP, password_AP);
19     if(result) {
20         Serial.println("Ready");
21         Serial.println(String("Soft-AP IP address = ") + WiFi.softAPIP().toString());
22         Serial.println(String("MAC address = ") + WiFi.softAPmacAddress().c_str());
23     } else {
24         Serial.println("Failed!");
25     }
26     Serial.println("Setup End");
27 }
28
29 void loop() {
30 }
```

Include WiFi Library header file of ESP8266.

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

Enter correct AP name and password.

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

Set ESP8266 in AP mode.

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

Configure IP address, gateway and subnet mask for ESP8266.

```
16 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.

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

Check whether the AP is turned on successfully. If yes, print out IP and MAC address of AP established by 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(): obtain IP address in AP mode

softAPdisconnect (): disconnect AP mode.



Project 20.3 AP+Station mode

Component List

ESP8266 x1	Micro USB Wire x1
A photograph of an ESP8266 module. It is a small black PCB with various pins and components. The FREENOVE logo is printed on the board. The pins are labeled with numbers and names such as 5V, 3V3, 3V3, 13, 12, 14, 16, EN, AO, RST, GND, 15, 2, 0, 4, 5, RX, and TX.	A diagram showing two black Micro USB cables with their respective connectors.

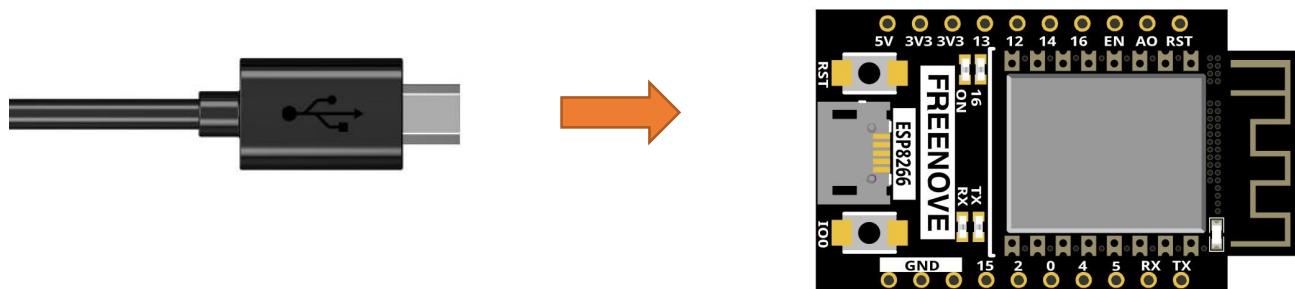
Component knowledge

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_20.3_AP_Station_mode

Please enter the correct names and passwords of Router and AP.

```

7 #include <ESP8266WiFi.h>
8
9 const char *ssid_Router      = "*****"; //Enter the router name
10 const char *password_Router = "*****"; //Enter the router password
11 const char *ssid_AP         = "WiFi_Name"; //Enter the router name
12 const char *password_AP    = "12345678"; //Enter the router password
13
14 void setup(){
15   Serial.begin(74880);
16   Serial.println("Setting soft-AP configuration ... ");
17   WiFi.disconnect();
18   WiFi.mode(WIFI_AP);
19   Serial.println("Setting soft-AP ... ");
20   boolean result = WiFi.softAP(ssid_AP, password_AP);
21   if(result){
22     Serial.println("Ready");
23     Serial.println(String("Soft-AP IP address = ") + WiFi.softAPIP().toString());
24     Serial.println(String("MAC address = ") + WiFi.softAPmacAddress().c_str());
  
```

Done uploading.

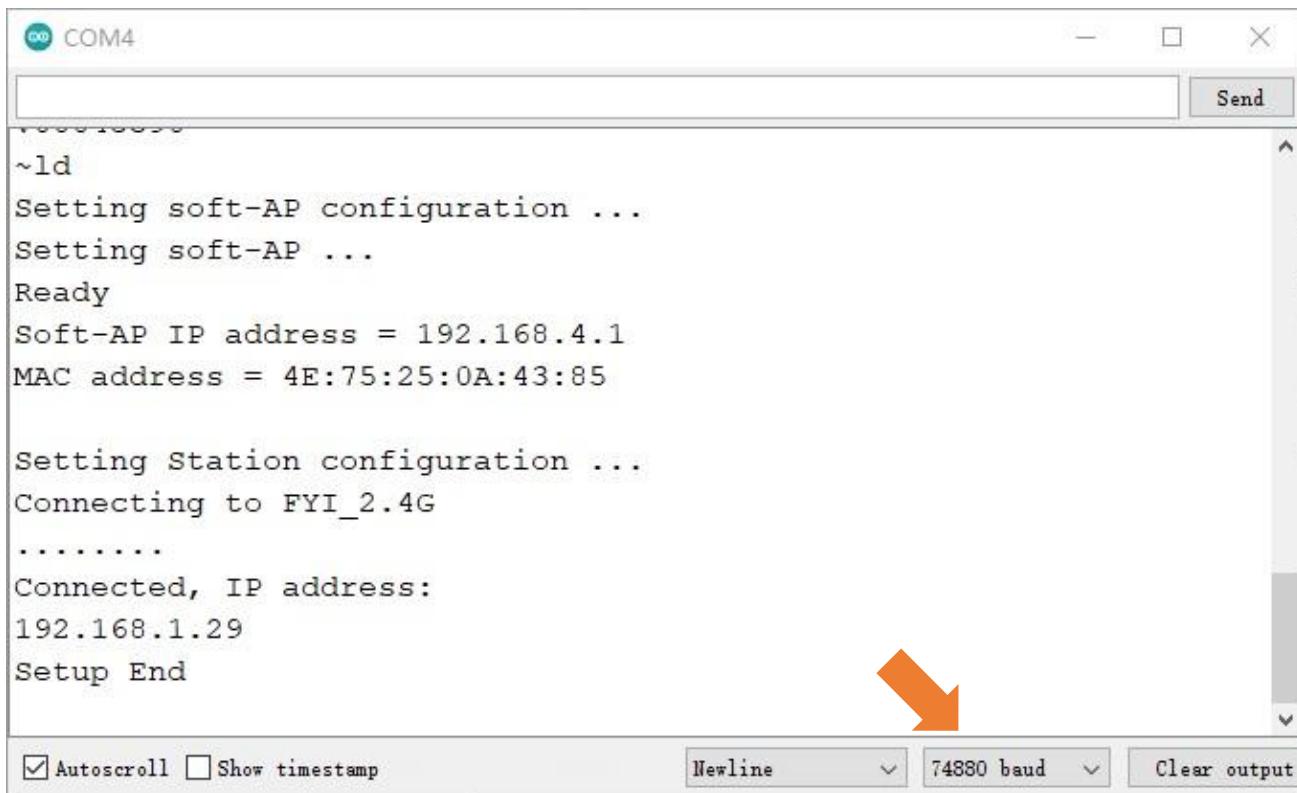
Leaving...
Hard resetting via RTS pin...

8M Cache + 32KB IRAM (balanced), Use pgm_read macros for IRAM/PROGMEM, 4MB (FS:3MB OTA:~512KB), 2, v2 Lower Memory, Disabled, None, Only Sketch, 115200 on COM4

It is analogous to Project 20.1 and Project 20.2. 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.

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

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:



```

COM4

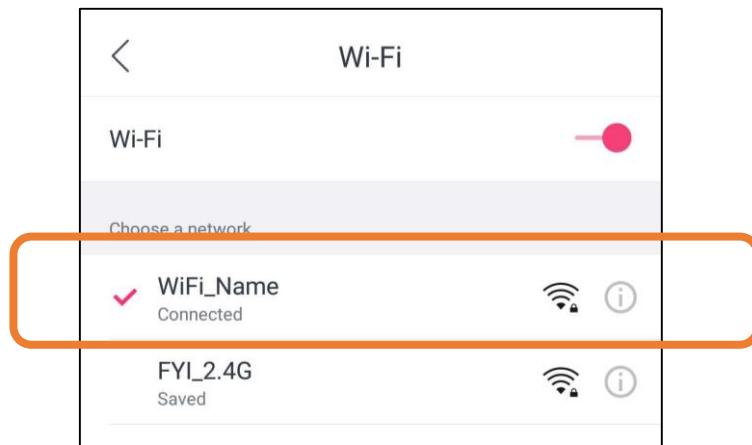
~ld
Setting soft-AP configuration ...
Setting soft-AP ...
Ready
Soft-AP IP address = 192.168.4.1
MAC address = 4E:75:25:0A:43:85

Setting Station configuration ...
Connecting to FYI_2.4G
.....
Connected, IP address:
192.168.1.29
Setup End

```

Autoscroll Show timestamp Newline 74880 baud Clear output

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:

```

1 #include <ESP8266WiFi.h>
2
3 const char *ssid_Router    = "*****"; //Enter the router name
4 const char *password_Router = "*****"; //Enter the router password
5 const char *ssid_AP        = "WiFi_Name"; //Enter the AP name
6 const char *password_AP    = "12345678"; //Enter the AP password
7

```

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

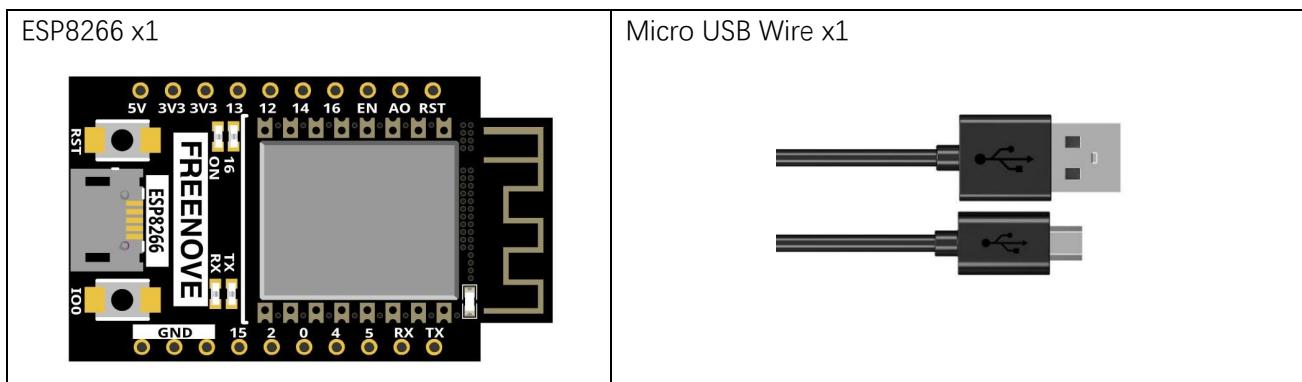
Chapter 21 TCP/IP

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 21.1 As Client

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

Component List



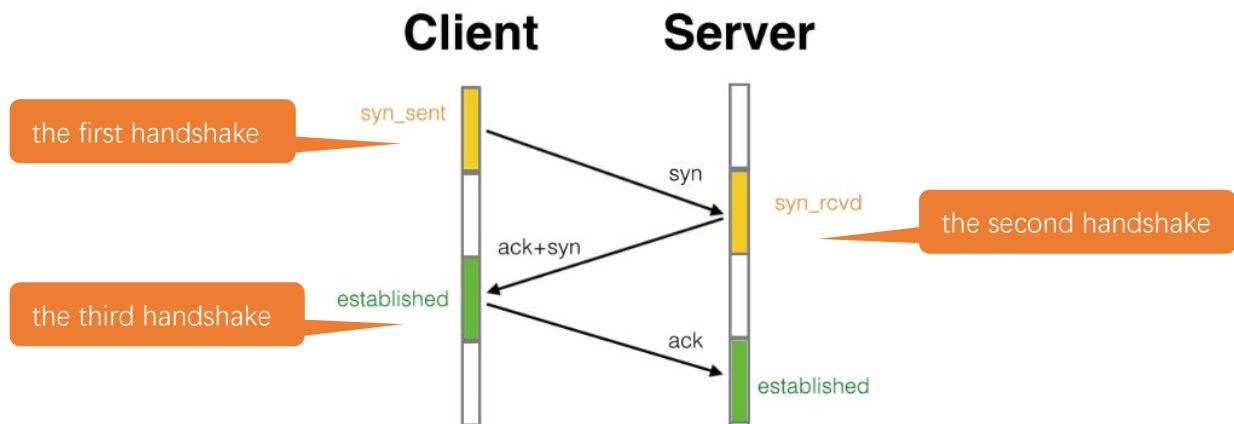
Component knowledge

TCP connection

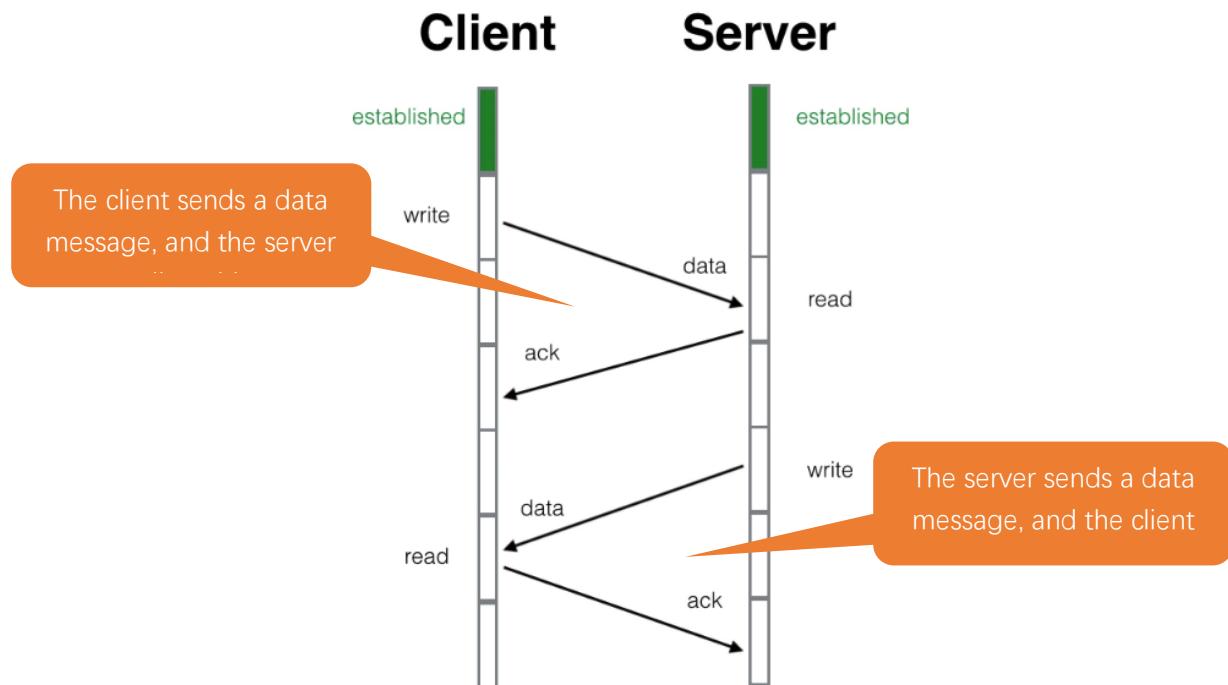
Before transmitting data, TCP needs to establish a logical connection between the sending end and the receiving end. It provides reliable and error-free data transmission between the two computers. In the TCP connection, the client and the server must be clarified. The client sends a connection request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

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

The third handshake, the client sends a confirmation message to the server again to confirm the connection.



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





Install Processing

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

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

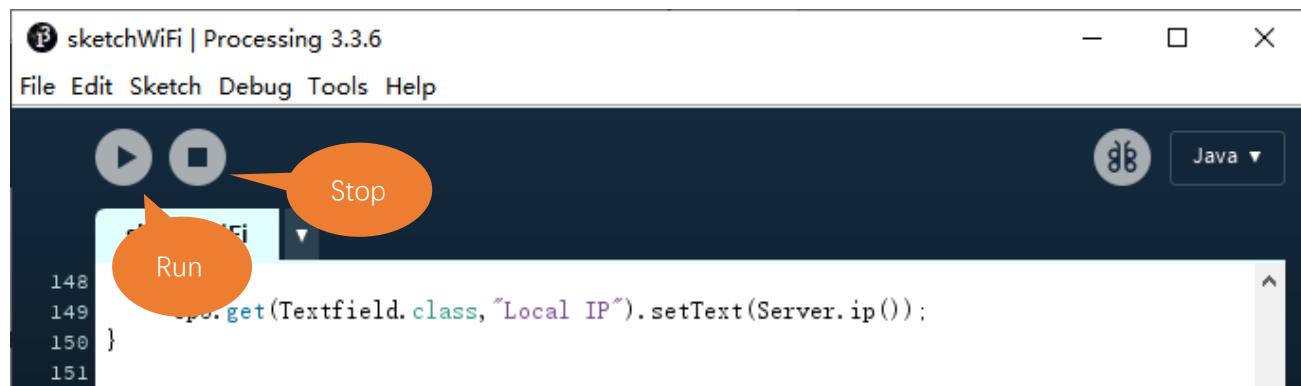
The screenshot shows the official Processing website. At the top, there's a navigation bar with links for "Processing", "p5.js", "Processing.py", "Processing for Android", "Processing for Pi", and "Processing Foundation". Below the navigation bar is a large banner featuring the word "Processing" in a stylized font over a background of abstract geometric shapes. To the right of the banner is a search bar with a magnifying glass icon. On the left side of the main content area, there's a sidebar with links: "Cover", "Download", "Donate", "Exhibition", "Reference", "Libraries", "Tools", "Environment", "Tutorials", "Examples", "Books", "Overview", and "People". In the center, under the heading "Download Processing", it says "Processing is available for Linux, Mac OS X, and Windows. Select your choice to download the software below." Below this, it shows "3.5.4 (17 January 2020)". There are four download links: "Windows 64-bit", "Windows 32-bit", "Linux 64-bit", and "Mac OS X". To the left of the download links is a large circular logo containing a stylized letter "P".

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

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

Use Server mode for communication

Open the “Freenove_Super_Starter_Kit_for_ESP8266\Sketches\Sketch_21.1_WiFiClient\sketchWiFi\sketchWiFi.pde”, and click “Run”.

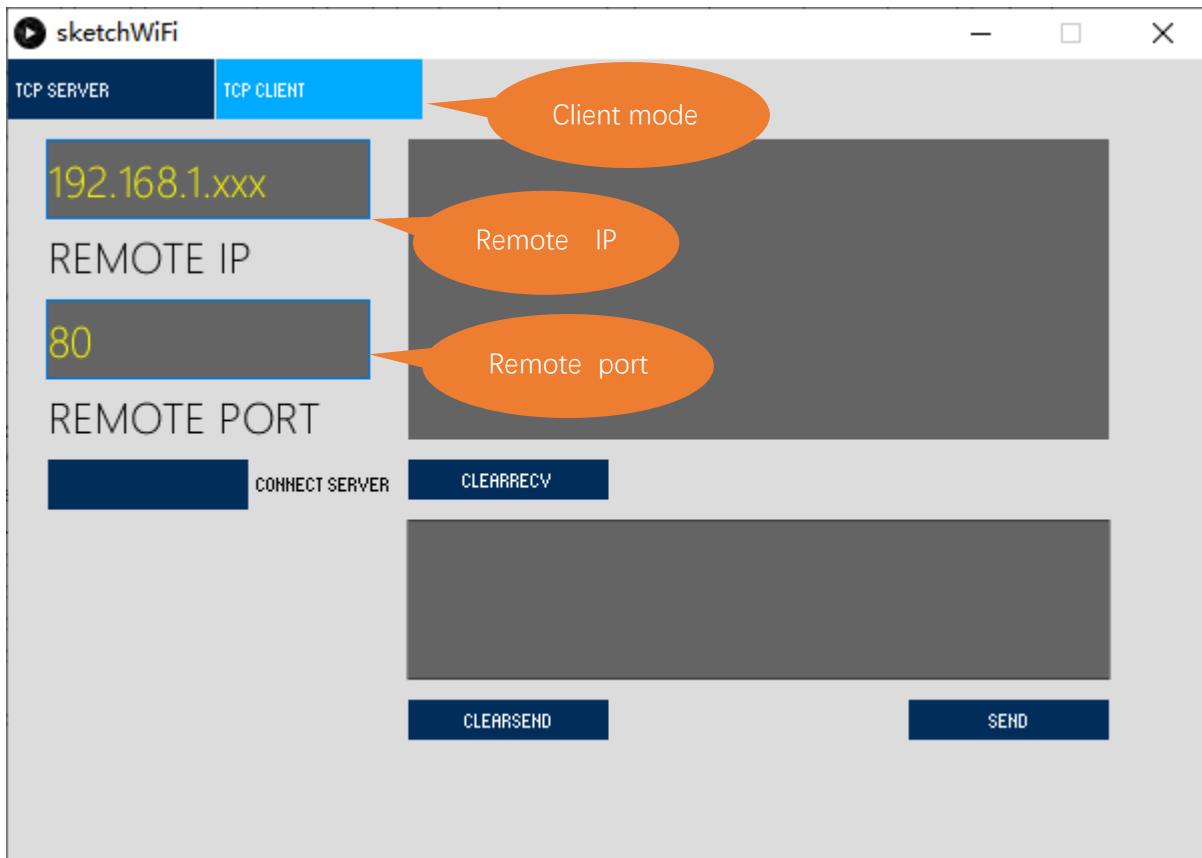


The new pop-up interface is as follows. If ESP8266 is used as client, select TCP SERVER mode for sketchWiFi.



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.

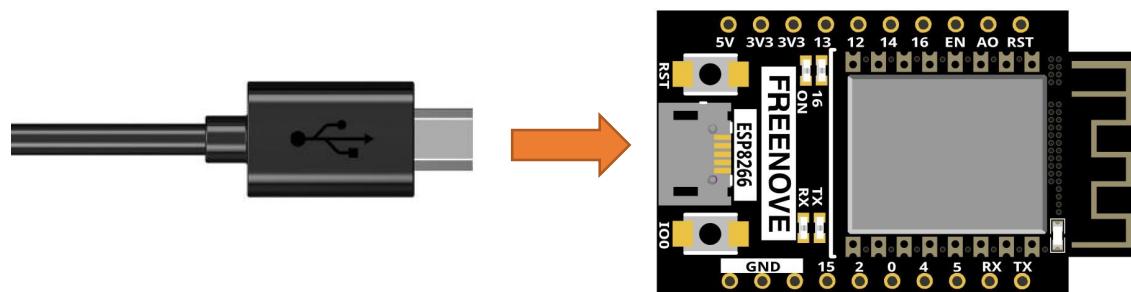
clear receive: clear out the content in the receiving text box

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

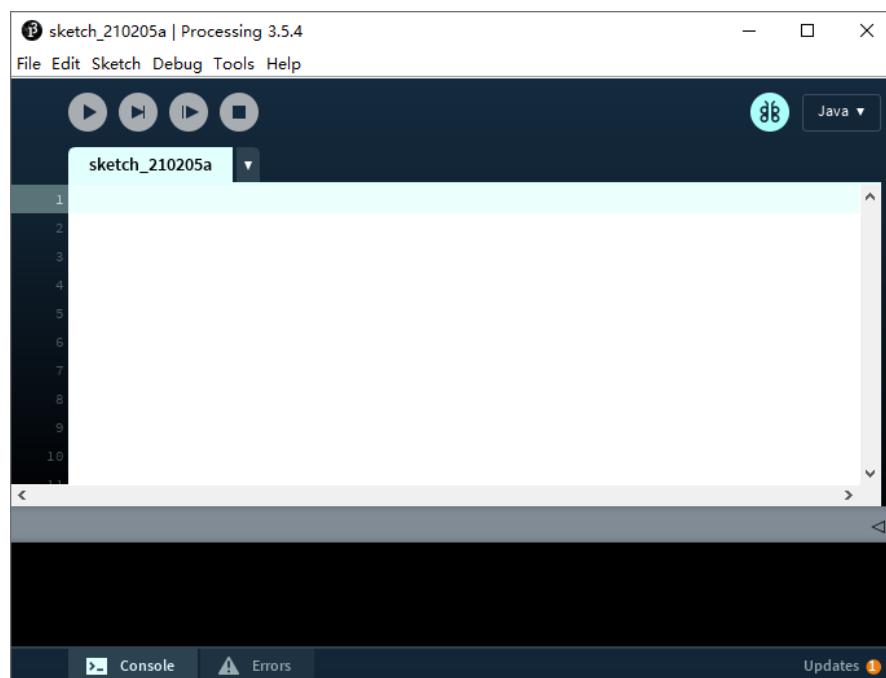
Connect Freenove ESP8266 to the computer using USB cable.



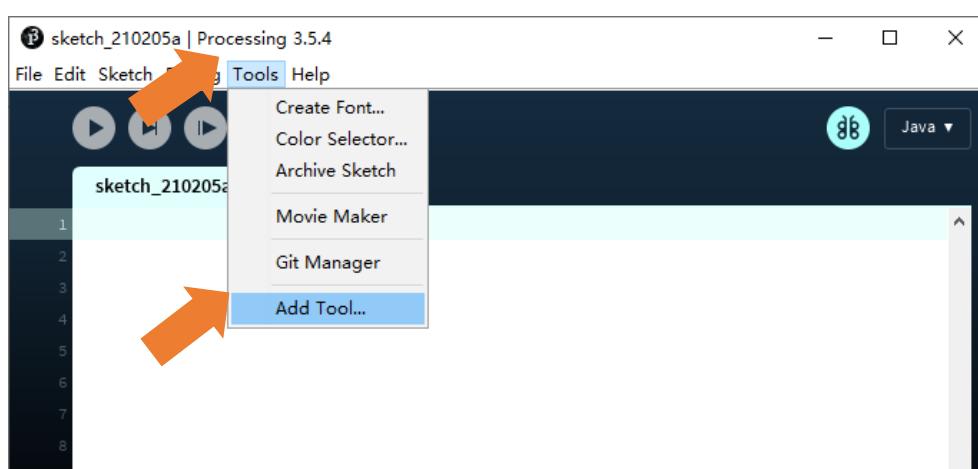
Sketch

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.

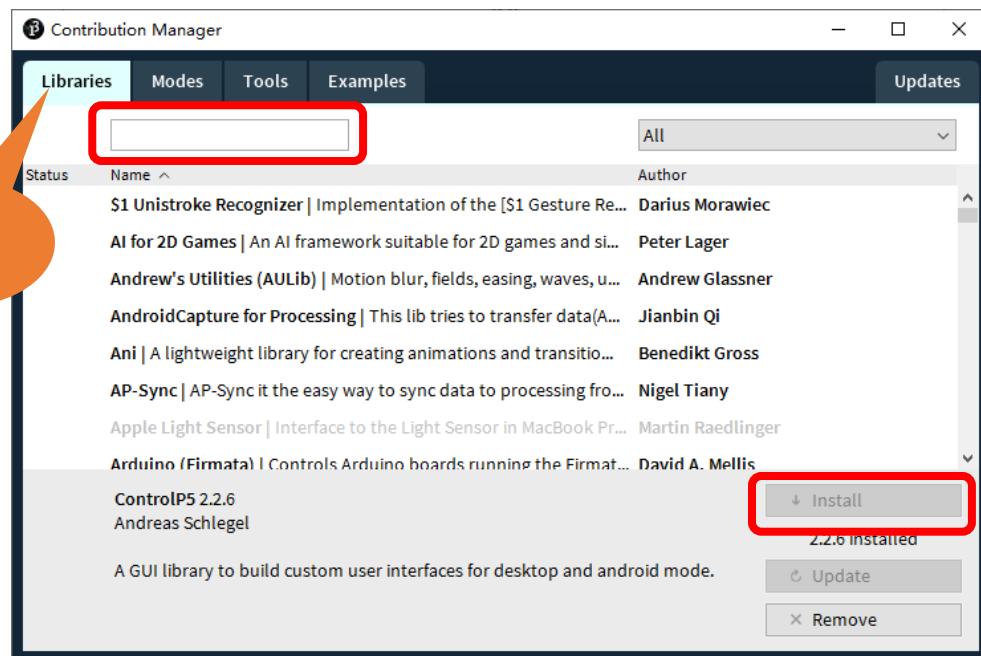


Click Add Tool under Tools.



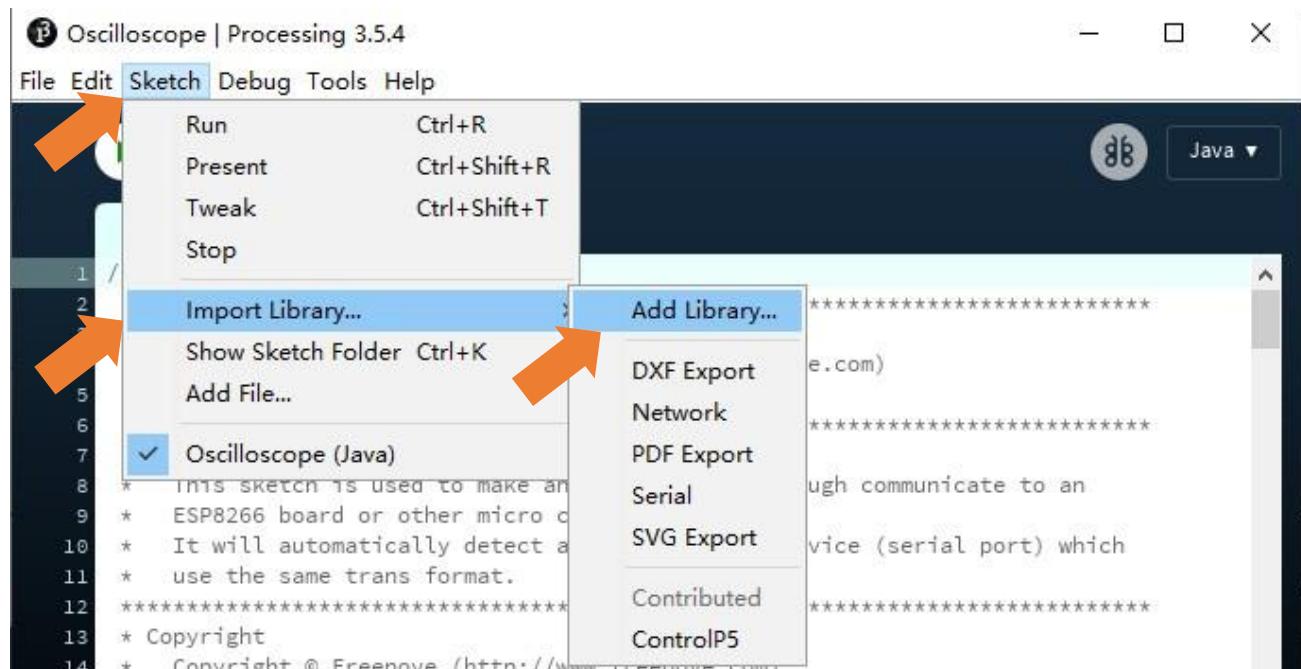
Any concerns? ✉ support@freenove.com

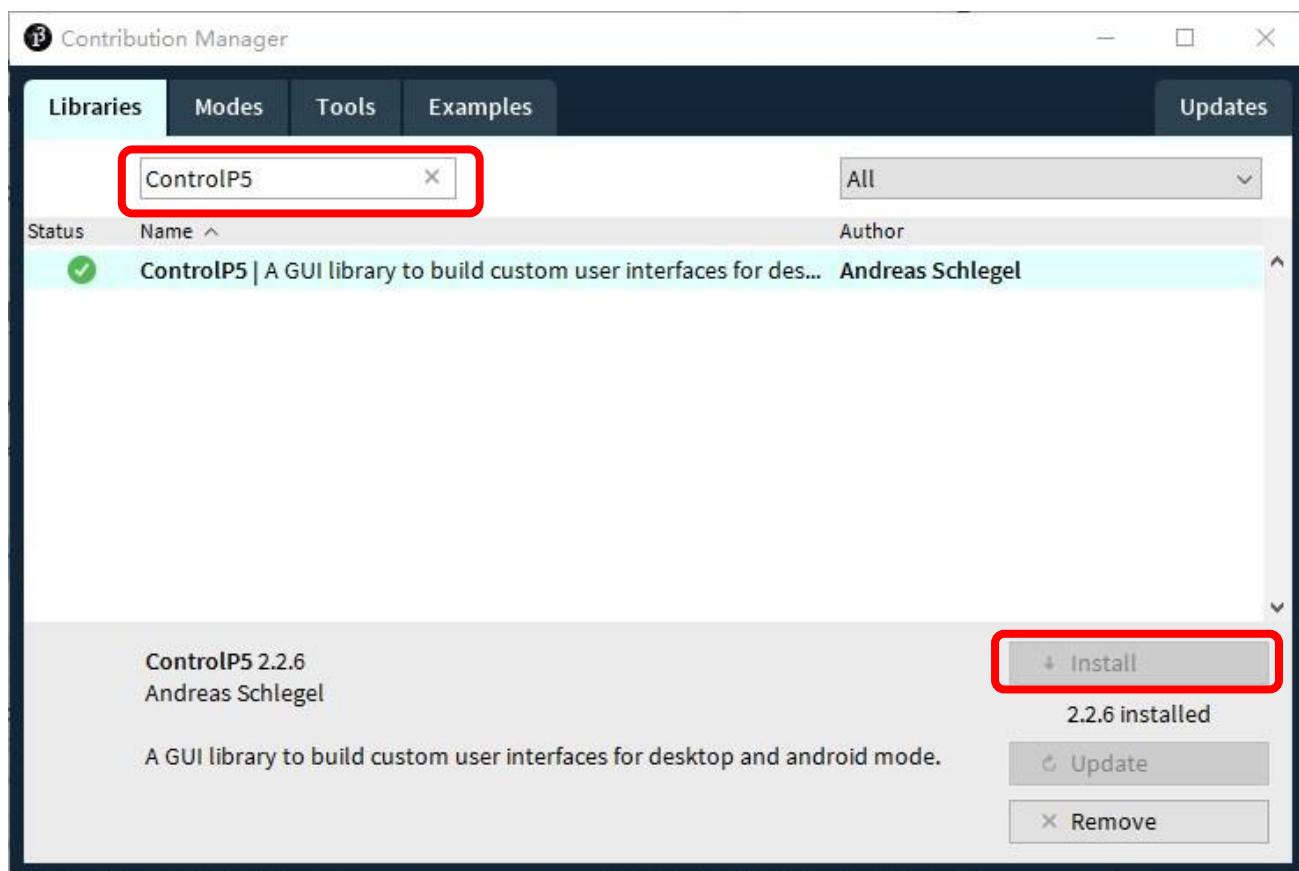
Select Libraries in the pop-up window.



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

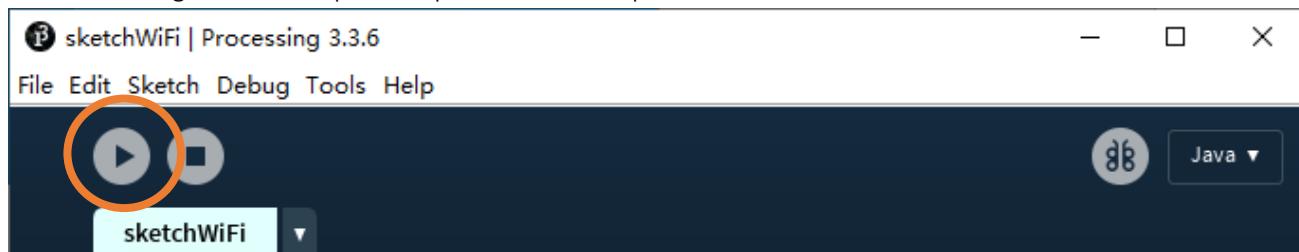
You can also click Add Library under 'Import Library' under 'Sketch'.



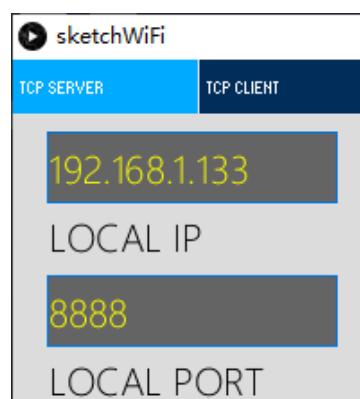


Sketch_21.1_As_Client

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

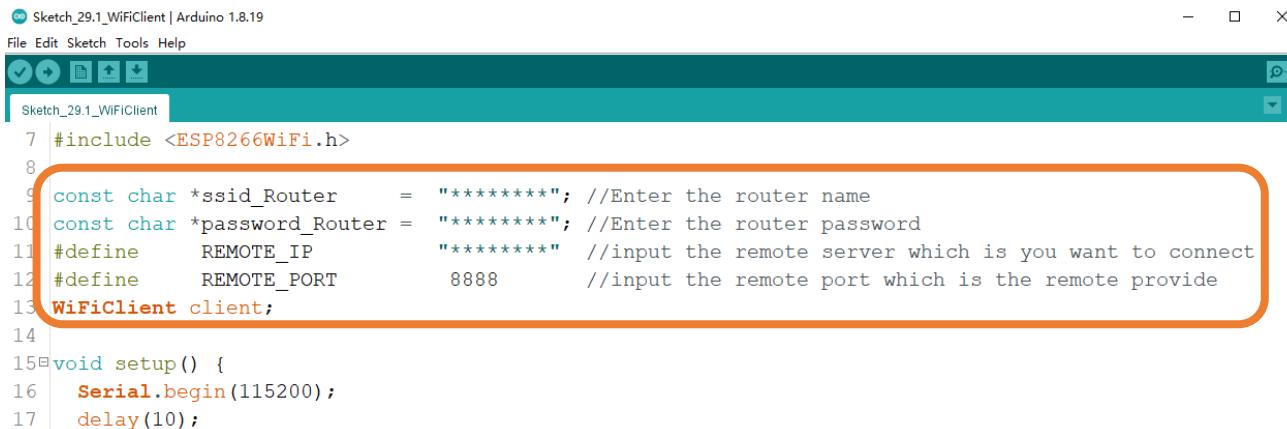


The newly pop up window will use the computer's IP address by default and open a data monitor port.



Next, open Sketch_21.1_WiFiClient.ino. Before running it, please change the following information based on "LOCAL IP" and "LOCAL PORT" in the figure above.

Any concerns? ✉ support@freenove.com



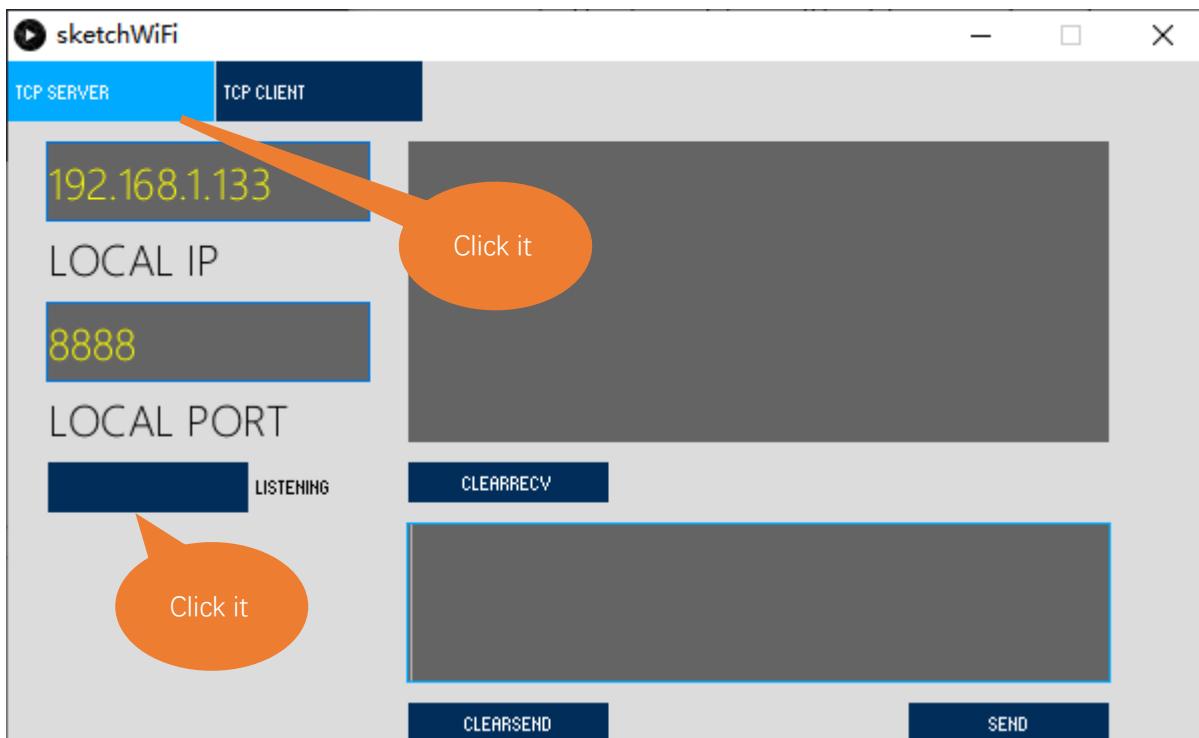
```

Sketch_29.1_WiFiClient | Arduino 1.8.19
File Edit Sketch Tools Help
Sketch_29.1_WiFiClient
7 #include <ESP8266WiFi.h>
8
9 const char *ssid_Router      = "*****"; //Enter the router name
10 const char *password_Router = "*****"; //Enter the router password
11 #define    REMOTE_IP        "*****"   //input the remote server which is you want to connect
12 #define    REMOTE_PORT       8888      //input the remote port which is the remote provide
13 WiFiClient client;
14
15 void setup() {
16   Serial.begin(115200);
17   delay(10);

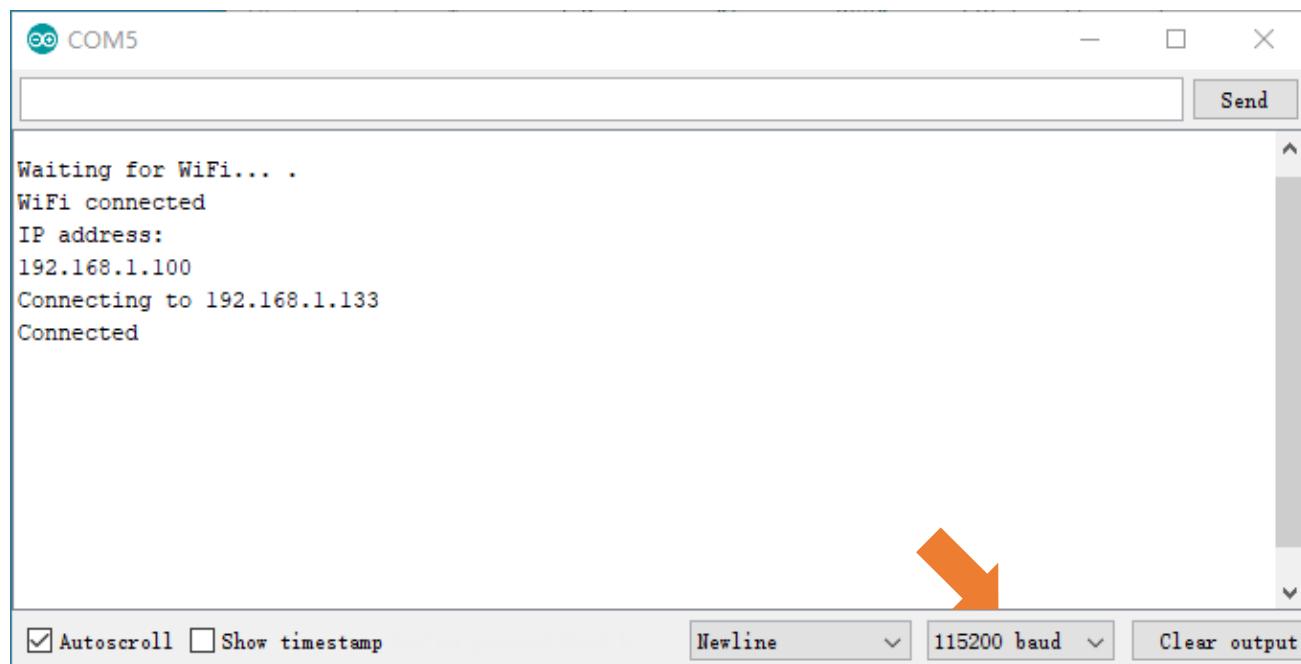
```

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.

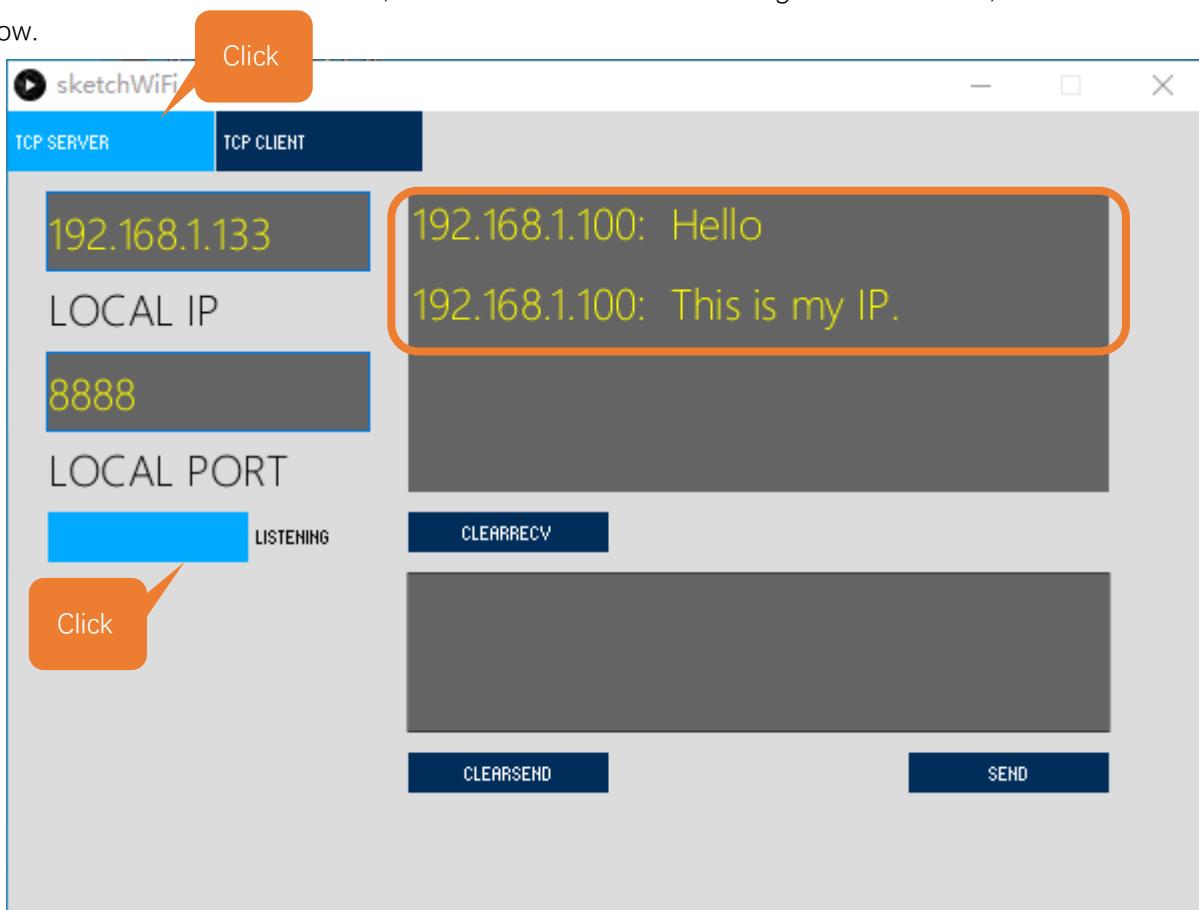
Click LISTENING, turn on TCP SERVER's data listening function and wait for ESP8266 to connect.



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.



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



The following is the program code:

```
1 #include <ESP8266WiFi.h>
2
3 const char *ssid_Router      = "*****"; //Enter the router name
4 const char *password_Router = "*****"; //Enter the router password
5 #define    REMOTE_IP        "*****"   //input the remote server which is you want to connect
6 #define    REMOTE_PORT       8888      //input the remote port which is the remote provide
7 WiFiClient client;
8
9 void setup() {
10     Serial.begin(115200);
11     delay(10);
12
13     WiFi.begin(ssid_Router, password_Router);
14     Serial.print("\nWaiting for WiFi... ");
15     while (WiFi.status() != WL_CONNECTED) {
16         Serial.print(".");
17         delay(500);
18     }
19     Serial.println("");
20     Serial.println("WiFi connected");
21     Serial.println("IP address: ");
22     Serial.println(WiFi.localIP());
23     delay(500);
24
25     Serial.print("Connecting to ");
26     Serial.println(REMOTE_IP);
27
28     while (!client.connect(REMOTE_IP, REMOTE_PORT)) {
29         Serial.println("Connection failed.");
30         Serial.println("Waiting a moment before retrying... ");
31     }
32     Serial.println("Connected");
33     client.print("Hello\n");
34     client.print("This is my IP.\n");
35
36 void loop() {
37     if (client.available() > 0) {
38         delay(20);
39         //read back one line from the server
40         String line = client.readString();
41         Serial.println(REMOTE_IP + String(":") + line);
42     }
}
```

Any concerns? ✉ support@freenove.com

```

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

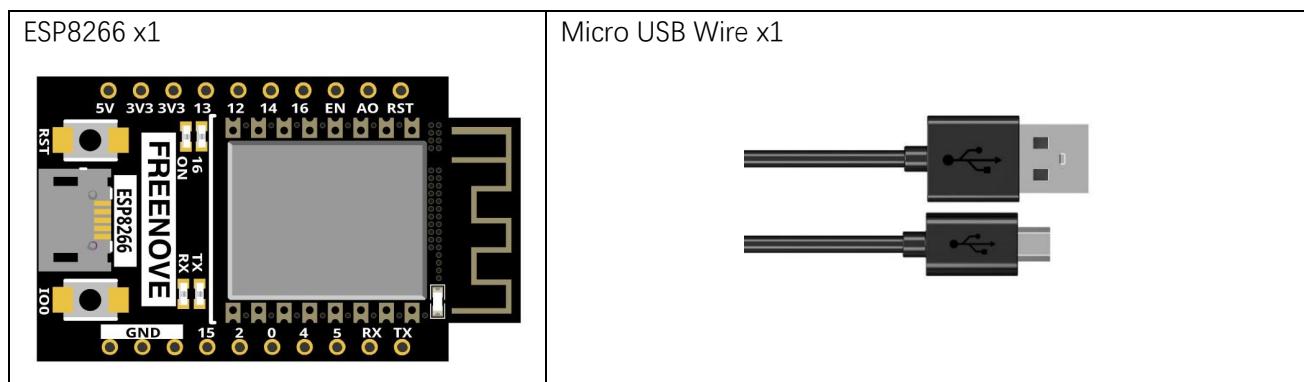
readString(): read string in receive buffer



Project 21.2 As Server

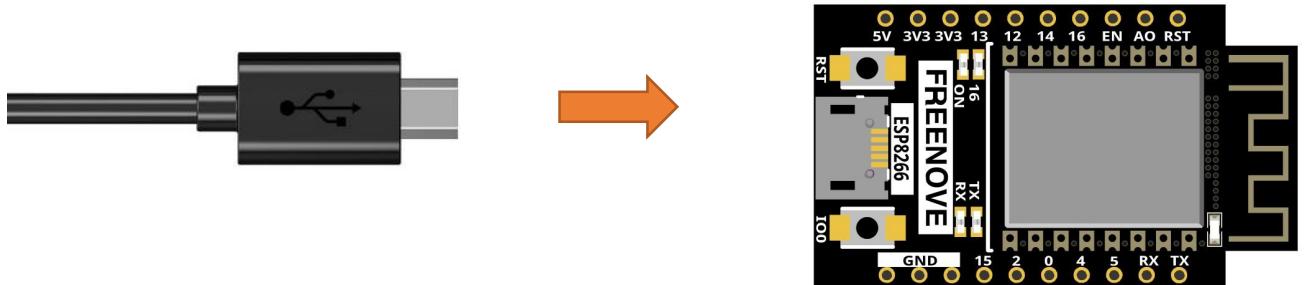
In this section, ESP8266 is used as a server to wait for the connection and communication of client on the same LAN.

Component List



Circuit

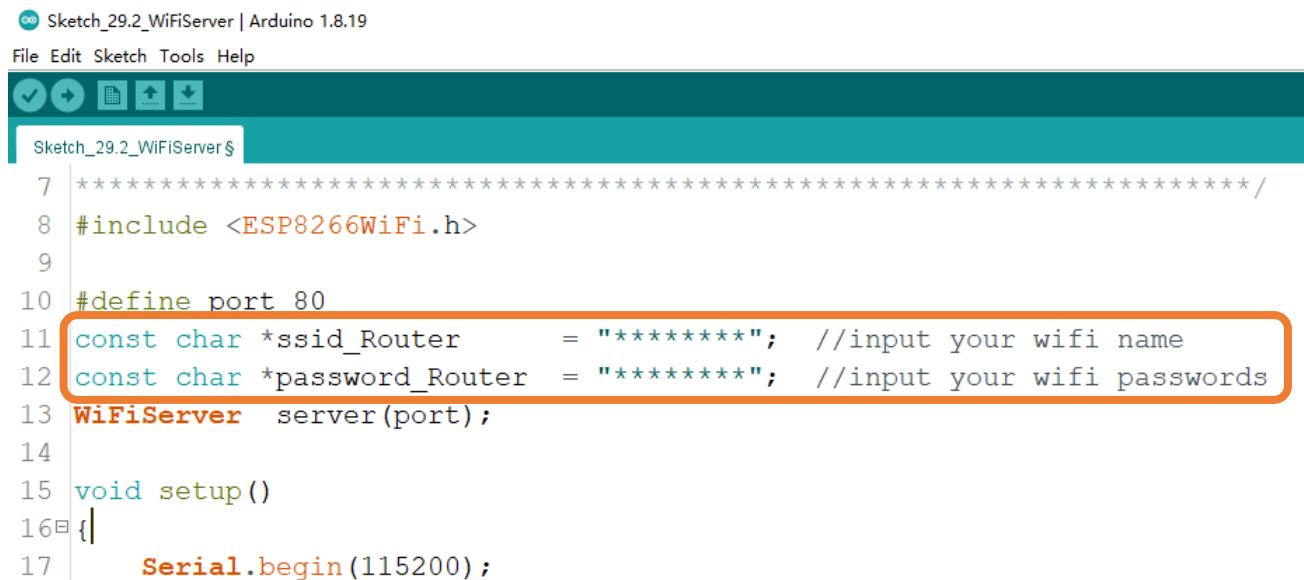
Connect Freenove ESP8266 to the computer using a USB cable.



Sketch

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

Sketch_21.2_As_Server



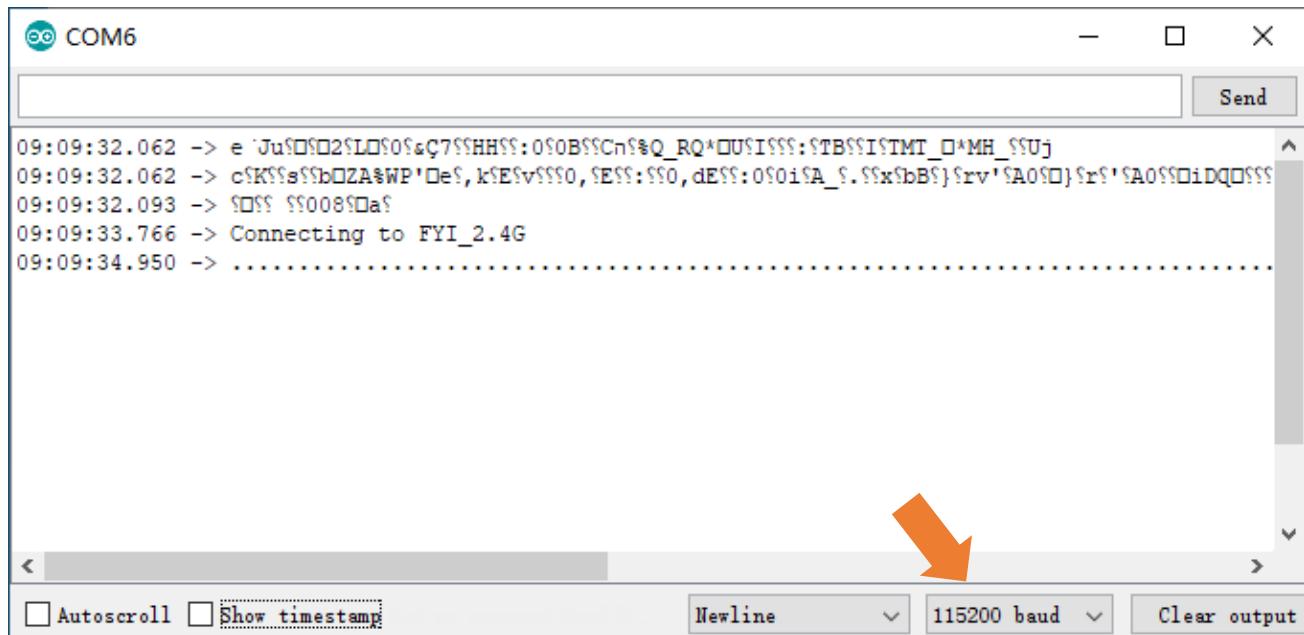
```

Sketch_29.2_WiFiServer | Arduino 1.8.19
File Edit Sketch Tools Help
Sketch_29.2_WiFiServer §
7 ****
8 #include <ESP8266WiFi.h>
9
10 #define port 80
11 const char *ssid_Router      = "*****"; //input your wifi name
12 const char *password_Router = "*****"; //input your wifi passwords
13 WiFiServer server(port);
14
15 void setup()
16 {
17     Serial.begin(115200);

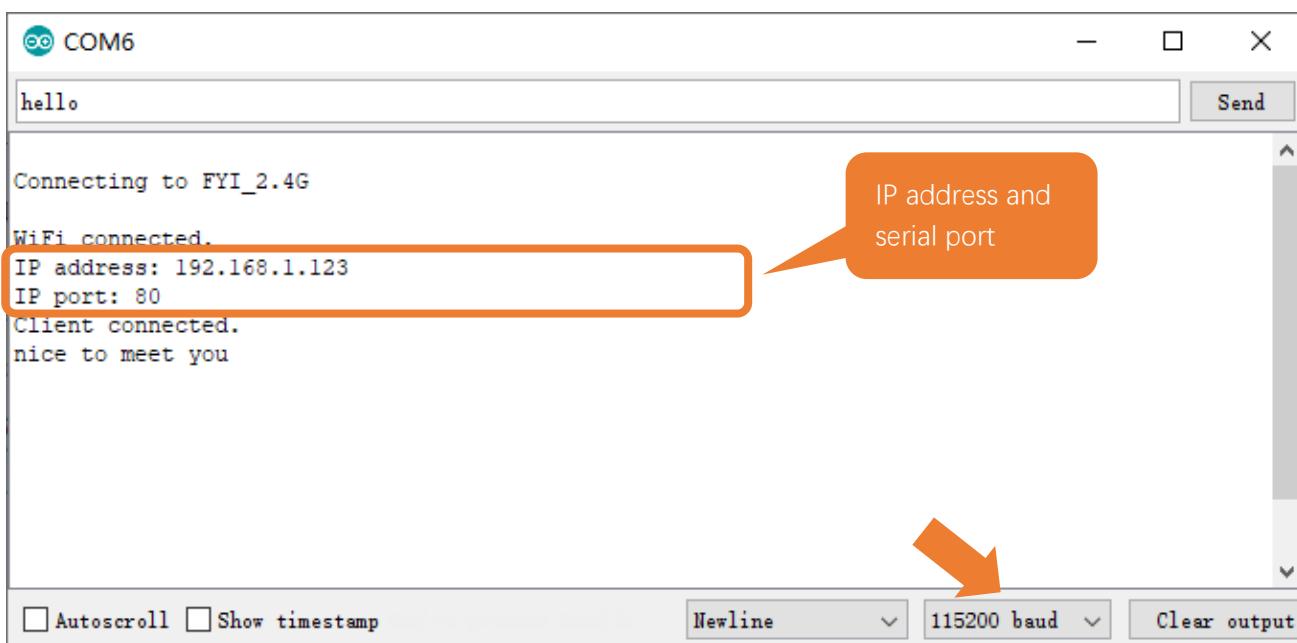
```

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.



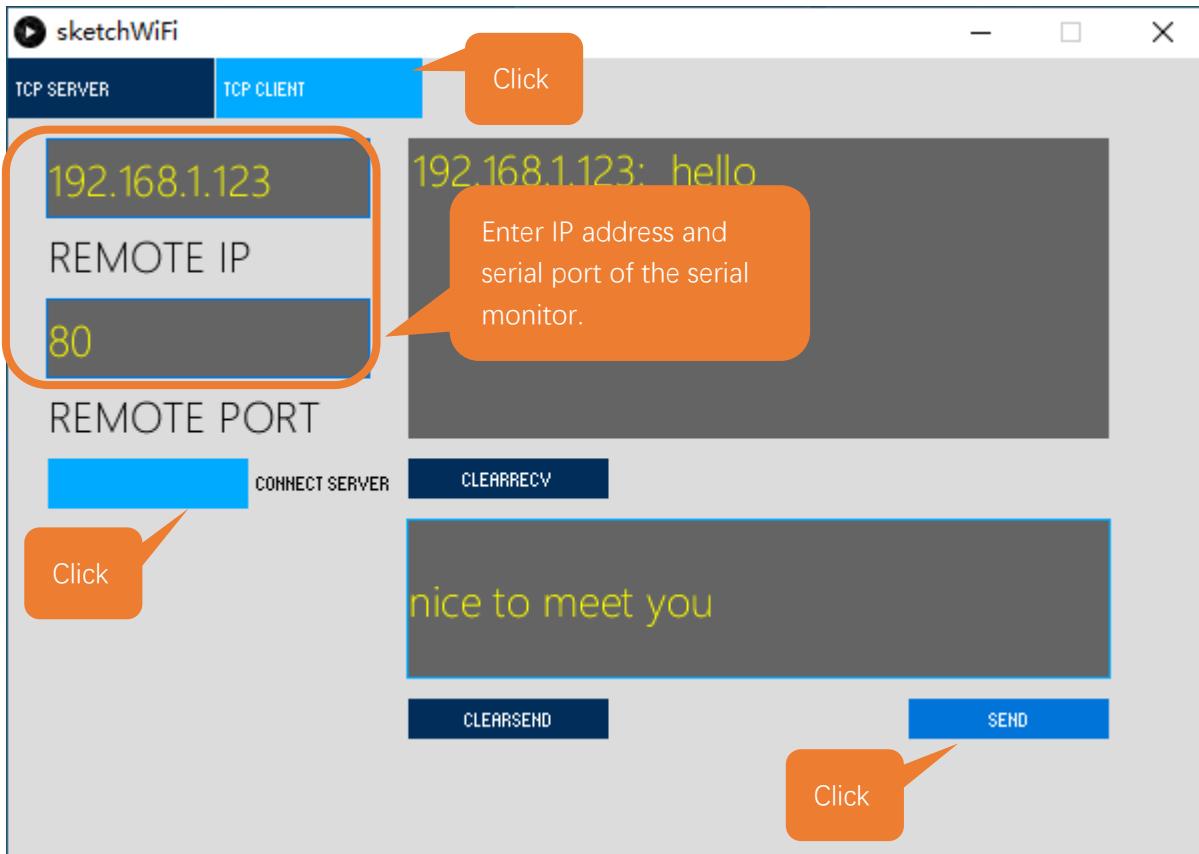
Serial Monitor



Processing:

Open the "Freenove_Super_Starter_Kit_for_ESP8266\C\Sketches\Sketch_21.2_WiFiServer\sketchWiFi\sketchWiFi.pde".

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



The following is the program code:

```
1 #include <ESP8266WiFi.h>
2
3 #define port 80
4 const char *ssid_Router      = "*****"; //input your wifi name
5 const char *password_Router = "*****"; //input your wifi passwords
6 WiFiServer server(port);
7
8 void setup()
9 {
10    Serial.begin(115200);
11    Serial.printf("\nConnecting to ");
12    Serial.println(ssid_Router);
13    WiFi.disconnect();
14    WiFi.begin(ssid_Router, password_Router);
15    delay(1000);
16    while (WiFi.status() != WL_CONNECTED) {
17        delay(500);
18        Serial.print(".");
19    }
20    Serial.println("");
21    Serial.println("WiFi connected.");
22    Serial.print("IP address: ");
23    Serial.println(WiFi.localIP());
24    Serial.printf("IP port: %d\n", port);
25    server.begin(port);
26    WiFi.setAutoConnect(true);
27    WiFi.setAutoReconnect(true);
28 }
29
30 void loop() {
31    WiFiClient client = server.available();           // listen for incoming clients
32    if (client) {                                     // if you get a client
33        Serial.println("Client connected.");
34        while (client.connected()) {                  // loop while the client's connected
35            if (client.available()) {                // if there's bytes to read from the
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
36 serial monitor
37                client.print(Serial.readStringUntil('\n'));// print it out the client.
38                while(Serial.read()>0);                  // clear the wifi receive area cache
39            }
40        }
41    }
42 }
```

Any concerns? ✉ support@freenove.com

```

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

Apply for method class of WiFiServer.

6	<code>WiFiServer server(port);</code> //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	<code>Serial.print("IP address: ");</code>	
23	<code>Serial.println(WiFi.localIP());</code>	//print out IP address of ESP8266
24	<code>Serial.printf("IP port: %d\n", port);</code>	//Print out ESP8266's port number

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

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

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

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.

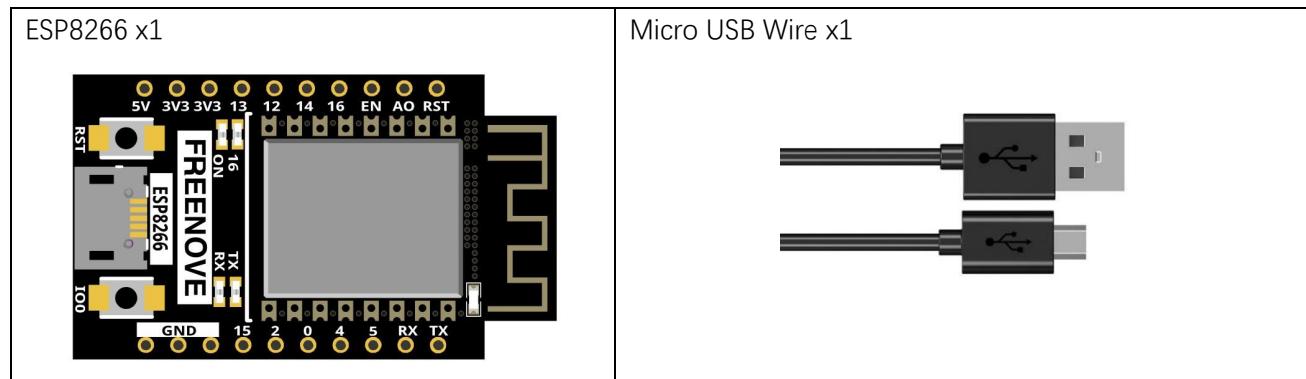
Chapter 22 Smart home

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 22.1 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



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>: Notes the title of the document

<body>: Contains visible page content

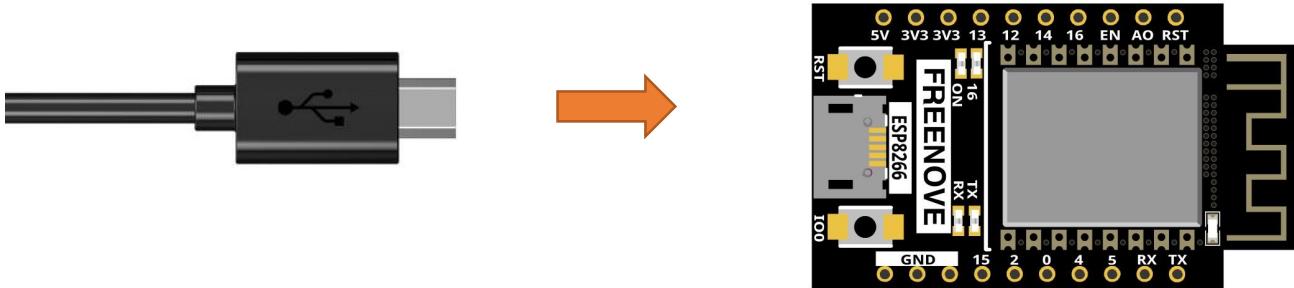
<h1>: Define a big heading

<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_22.1_Control_LED_through_Web

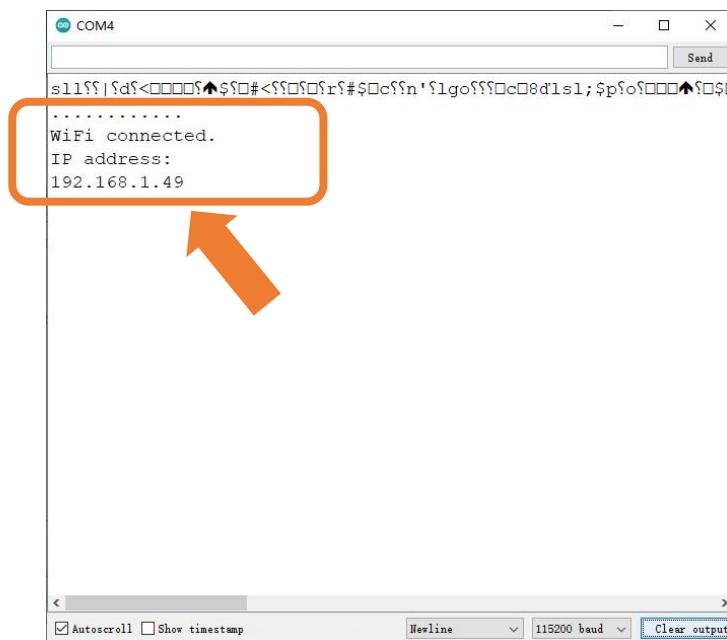
```
Sketch_30.1_Control_LED_through_Web | Arduino 1.8.19
File Edit Sketch Tools Help
Sketch_30.1_Control_LED_through_Web
10 // *****
11 #include <ESP8266WiFi.h>
12 // Replace with your network credentials
13 const char* ssid      = "*****";
14 const char* password = "*****";
15 // Set web server port number to 80
16 WiFiServer server(80);
17 // Variable to store the HTTP request
18 String header;
19 // Auxiliar variables to store the current output state
20 String PIN_LEDState = "off";
21 // Assign output variables to GPIO pins
22 const int PIN_LED = 2;
23 // Current time
24 unsigned long currentTime = millis();
25 // Previous time
26 unsigned long previousTime = 0;
27 // Define timeout time in milliseconds (example: 2000ms = 2s)
28 const long timeoutTime = 2000;
29
30 void setup() {
31   Serial.begin(115200);
32   // Initialize the output variables as outputs
33   pinMode(PIN_LED, OUTPUT);
34   // Set outputs to LOW
35   digitalWrite(PIN_LED, HIGH);
}
Done Saving.

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

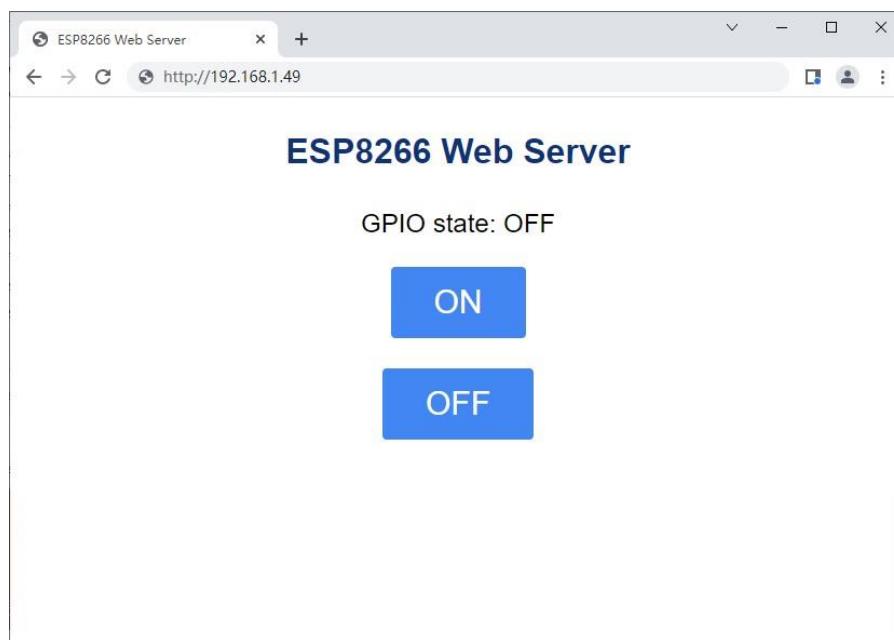
Info: All SSL ciphers (most compatible), 32KB cache + 32KB IRAM (balanced). Use pgm_read macros for IRAM/PROGMEM, 4MB (FS:2MB OTA:~1019KB), 2, v2 Lower Memory, Disabled, None, Only Sketch, 115200 on COM4
```

Enter the correct Router name and password.

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

```

Any concerns? ✉ support@freenove.com

```
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:, \">"); // CSS to style the on/off buttons
78             // Feel free to change the background-color and font-size attributes to fit your preferences
79             client.println("<style>html {font-family: Helvetica; display:inline-block; margin: 0px auto; text-align: center;}</style>"); // h1{color: #0F3376; padding: 2vh;} p{font-size: 1.5rem;}); .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;}); .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>"); // Web Page Heading
80             client.println("<body><h1>ESP8266 Web Server</h1>"); // p>GPIO state: " + PIN_LEDState + "</p>");
```

```

87         client.println("<p><a href=\"/5/ON\"><button class=\"button
88             button2\">ON</button></a></p>");
89         client.println("<p><a href=\"/5/OFF\"><button class=\"button
90             button2\">OFF</button></a></p>");
91         client.println("</body></html>");
92         // The HTTP response ends with another blank line
93         client.println();
94         // Break out of the while loop
95         break;
96     } else { // if you got a newline, then clear currentLine
97         currentLine = "";
98     }
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 }
```

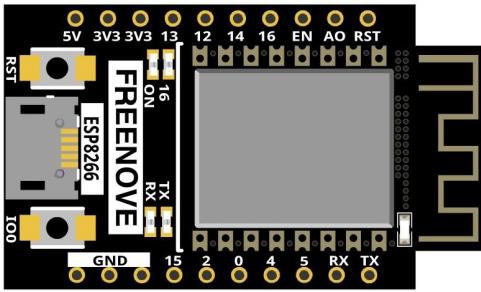
Chapter 23 Play music

In this chapter, we use ESP8266 to output audio signals and play music through ESP8266. Direct audio output through the speaker, this will have significant distortion.

Project 23.1 Play_music

In this project, we use ESP8266 to play local music. When we press the button, the music plays again after a certain amount of time.

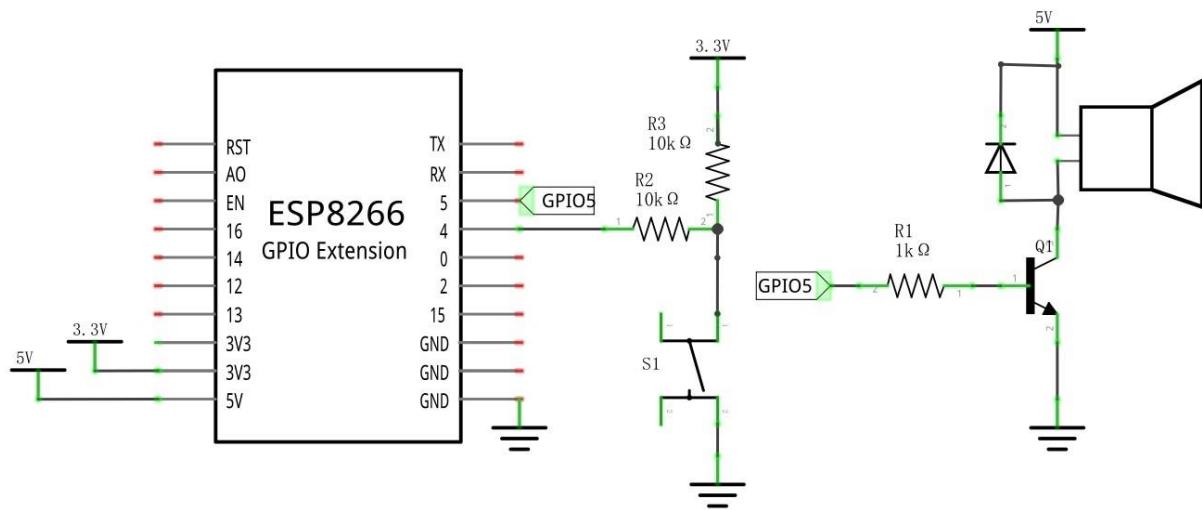
Component List

ESP8266 x1	Micro USB Wire x1			
				
Jumper wire M/M x12	Speaker*1			
NPN transistorx1 (S8050)	Push button x1	Diode x1	Resistor 1kΩ x1	Resistor 10kΩ x2

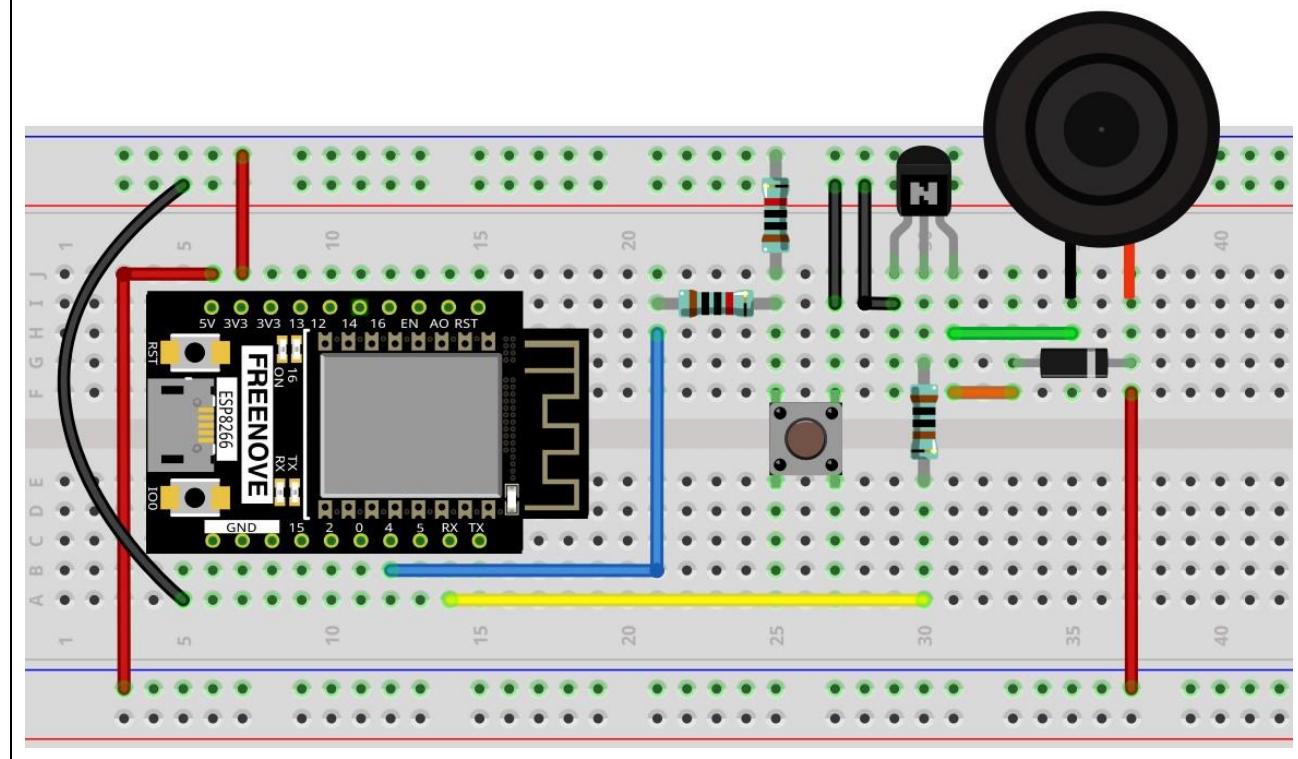
Circuit

In this section, you need to perform operations in the following sequence: Upload audio data in the first step, upload code in the second step, disconnect the power supply in the third step, connect hardware in the fourth step as follows, and connect power supply in the fifth step. Make sure you do this in order to avoid permanent damage to your hardware.

Schematic diagram



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



Any concerns? ✉ support@freenove.com



Sketch

How to install the library

This code is used to play music. We use the third-party library **ESP8266Audio**. If you haven't already installed it, install it now. The steps to add third-party libraries are as follows: The first way, Open Arduino -> Sketch->Include library -> Manage library. Type ESP8266Audio in the search bar and select ESP8266Audio to install.

Refer to the following operations:



The second way, open Arduino IDE, click Sketch→Include Library→Add .ZIP Library, In the pop-up window, find the file named “./Libraries/ESP8266Audio.Zip” which locates in this directory, and click OPEN.

Install the Arduino IDE plug-in Arduino-ESP8266Fs-Plugin

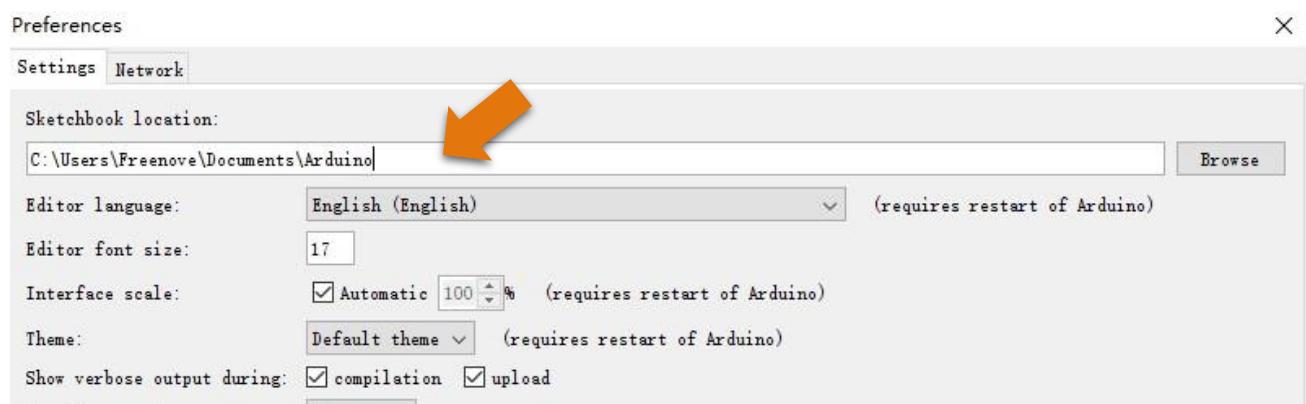
In this tutorial, find a folder named ". /Sketches/ Sketch_23.1_play_music/tools" under that directory and copy that folder into the Arduino IDE environment directory.

The details are as follows:

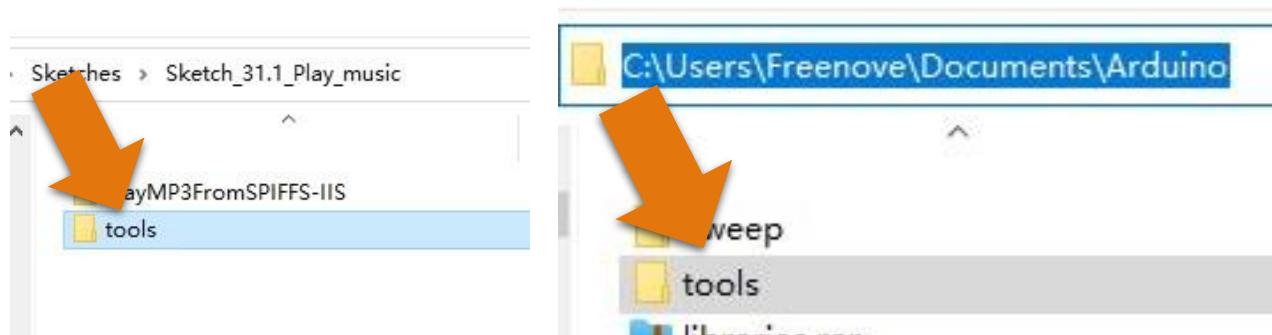
First, open the software platform arduino, and then click File in Menus and select Preferences.



Find the Arduino IDE environment directory location.



Copy "tools" folder to this directory.

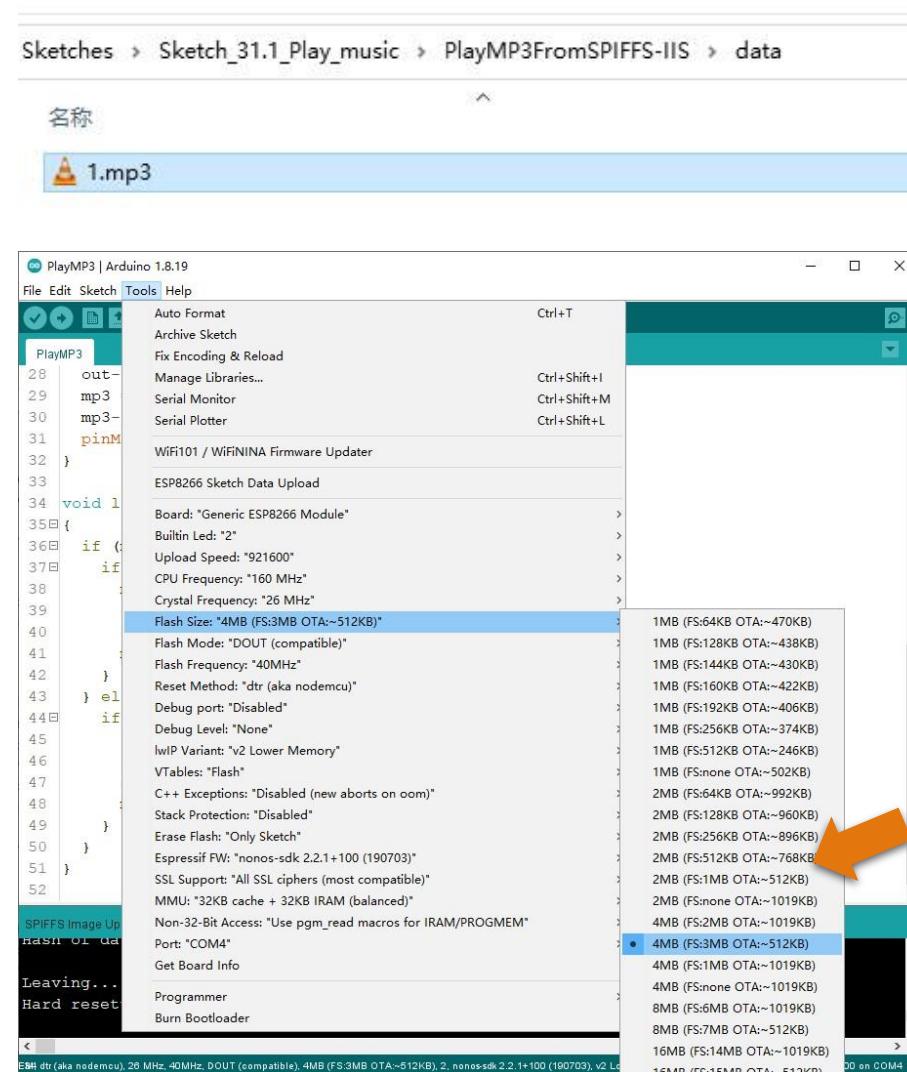


Finally, restart the Arduino IED

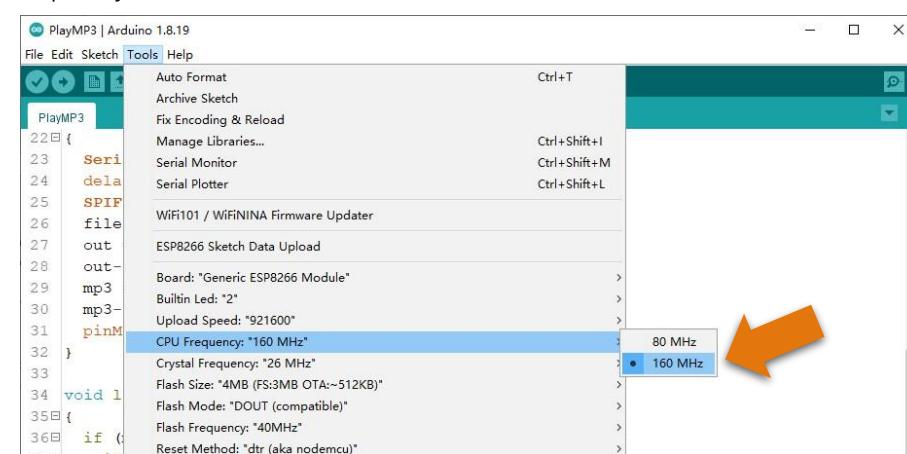
Upload music

Before uploading the music file, in the path ". /Sketches/Sketch_23.1_Play_music/PlayMP3/data" folder, save the music file that needs to be uploaded and make sure the file content is not empty.

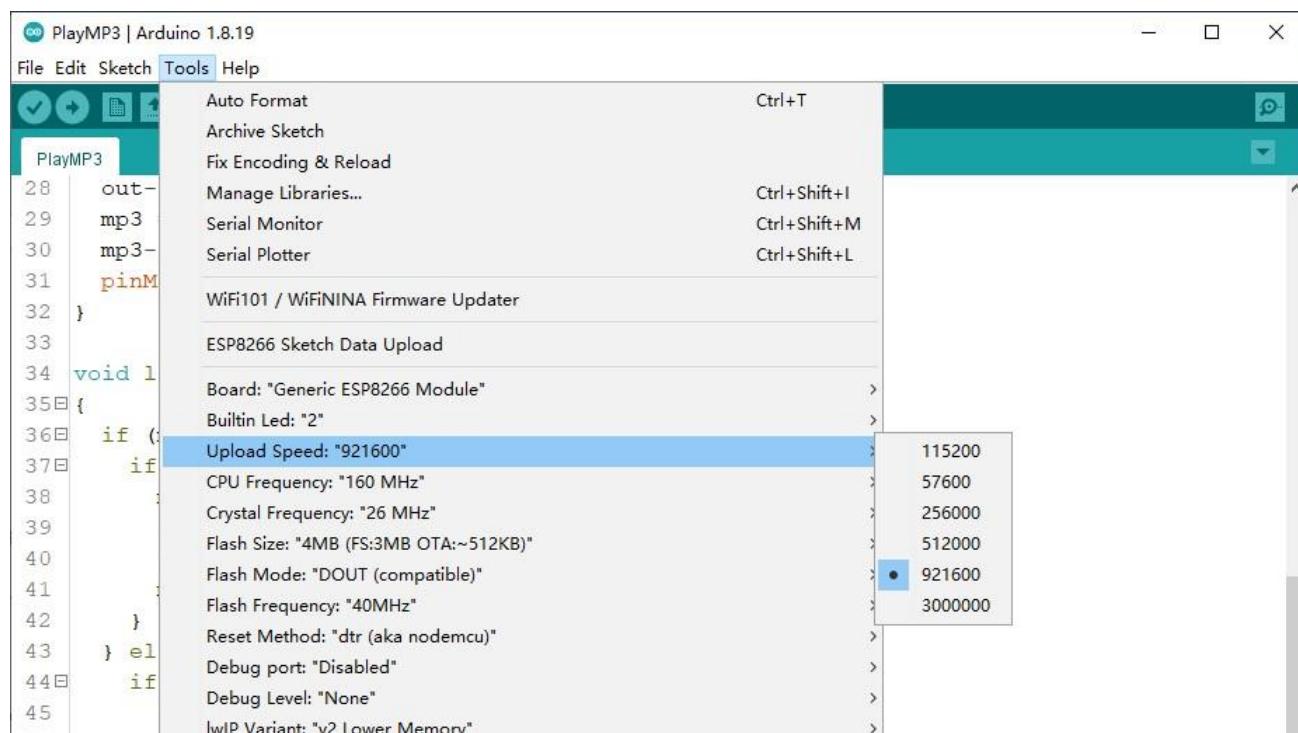
If you do not have a "data" folder, you need to create a "data" folder and put MP3 or WAV files in the "data" folder. In addition, the audio file size cannot exceed the selected FlashSize.



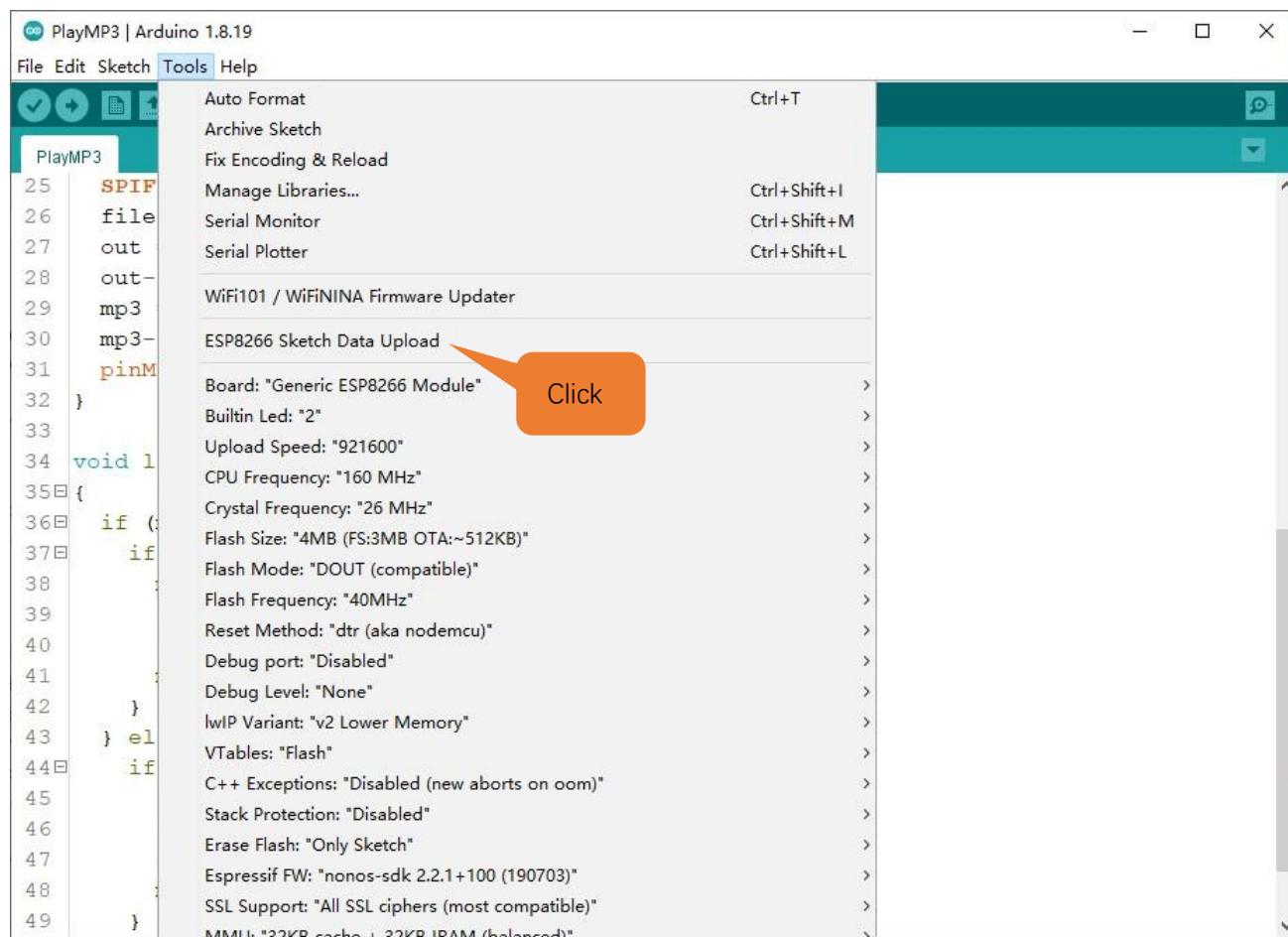
ESP8266's CPU frequency is set to 160MHz.



Because the file size is large, set the upload speed to 921600.



Open the Arduino software platform, click Tools in Menu, and choose **ESP8266 Sketch Data Upload**. If this tool does not appear, you can restart the Arduino IDE and try again.



```

PlayMP3 | Arduino 1.8.19
File Edit Sketch Tools Help
PlayMP3
34 void loop()
35{
36    if (mp3->isRunning()) {
37        if (!mp3->loop()){
38            mp3->stop();
39            delete file;
40            delete mp3;
41            mp3 = new AudioGeneratorMP3();
42        }
43    } else {
44        if (digitalRead(PIN_BUTTON) == LOW){
45            file = new AudioFileSourceSPIFFS("/1.mp3");
46            Serial.printf("MP3 done\n");
47            delay(1000);
48            mp3->begin(file, out);
49        }
50    }
51
52
53
54
55
56
57
58
59
59 SPIFFS Image Uploaded
60 Writing at 0x00228000... (97 %)
61 Writing at 0x0022c000... (98 %)
62 Writing at 0x00230000... (100 %)
63 Wrote 3121152 bytes (1254693 compressed) at 0x00100000 in 28.5 seconds (effective 874.7 kbit/s)
64 Hash of data verified.

65 Leaving...
66 Hard resetting via RTS pin...

```

ESP8266 (aka nodemcu), 26 MHz, 40MHz, DOUT (compatible), 4MB (FS:3MB OTA~512KB), 2, nemos-sdk 2.2.1+100 (190703), v2 Lower Memory, Disabled, None, Only Sketch, 921600 on COM4

After uploading, upload the code to the ESP8266 development.

```

PlayMP3 | Arduino 1.8.19
File Edit Sketch Tools Help
PlayMP3
28     out->SetGain(0.3); //Volume Setup
29     mp3 = new AudioGeneratorMP3();
30     mp3->begin(file, out);
31     pinMode(PIN_BUTTON, INPUT);
32 }
33
34 void loop()
35{
36    if (mp3->isRunning()) {
37        if (!mp3->loop()){
38            mp3->stop();
39            delete file;
40            delete mp3;
41            mp3 = new AudioGeneratorMP3();
42        }
43    } else {
44        if (digitalRead(PIN_BUTTON) == LOW){
45            file = new AudioFileSourceSPIFFS("/1.mp3");
46            Serial.printf("MP3 done\n");
47            delay(1000);
48            mp3->begin(file, out);
49        }
50    }
51
52
53
54
55
56
57
58
59
59 Done uploading.
60 Hash of data verified.

61 Leaving...
62 Hard resetting via RTS pin...

```

ESP8266 (aka nodemcu), 26 MHz, 40MHz, DOUT (compatible), 4MB (FS:3MB OTA~512KB), 2, nemos-sdk 2.2.1+100 (190703), v2 Lower Memory, Disabled, None, Only Sketch, 921600 on COM4

Upload following sketch:

**Freenove_Super_Starter_Kit_for_ESP8266\С\Sketch_23.1_Play_music\PlayMP3
Sketch_23.1_Play_music**

The screenshot shows the Arduino IDE interface with the following details:

- Title Bar:** PlayMP3 | Arduino 1.8.19
- Menu Bar:** File Edit Sketch Tools Help
- Toolbar:** Standard icons for Open, Save, Upload, and Refresh.
- Sketch Area:** The code for "PlayMP3" is displayed. The code initializes an MP3 player using SPIFFS and I2SNoDAC, and sets up a button to start playback.

```
16
17 AudioGeneratorMP3 *mp3;
18 AudioFileSourceSPIFFS *file;
19 AudioOutputI2SNoDAC *out;
20
21 void setup()
22 {
23     Serial.begin(115200);
24     delay(1000);
25     SPIFFS.begin();
26     file = new AudioFileSourceSPIFFS ("/1.mp3");
27     out = new AudioOutputI2SNoDAC ();
28     out->SetGain(0.3); //Volume Setup
29     mp3 = new AudioGeneratorMP3 ();
30     mp3->begin(file, out);
31     pinMode(PIN_BUTTON, INPUT);
32 }
33
34 void loop()
```

- Status Bar:** Done uploading.
- Serial Monitor:** Shows the message "Leaving... Hard resetting via RTS pin..."
- Bottom Status:** - 38KB IRAM (balanced), Use pgm_read macros for IRAM/PROGMEM, 4MB (FS:3MB OTA:~512KB), 2, v2 Lower Memory, Disabled, None, Only Sketch, 115200 on COM9

After uploading the code, wait a moment and you will hear the music playing. Press the button and play the music again.

If no music is played, check whether the hardware circuit is properly connected. If the circuit is properly connected, check that the audio file has been uploaded and that the audio name matches the name in the code.

The following is the program code:

```
1 #include <Arduino.h>
2 #include "AudioFileSourceSPIFFS.h"
3 #include "AudioGeneratorMP3.h"
4 #include "AudioOutputI2SNoDAC.h"
5 #include <ESP8266WiFi.h>
6 #include "AudioFileSourceID3.h"
7 #include "AudioOutputI2S.h"
```

```

8
9 #define PIN_BUTTON 5
10
11 AudioGeneratorMP3 *mp3;
12 AudioFileSourceSPIFFS *file;
13 AudioOutputI2SNoDAC *out;
14
15 void setup()
16 {
17   Serial.begin(115200);
18   delay(1000);
19   SPIFFS.begin();
20   file = new AudioFileSourceSPIFFS("/1.mp3");
21   out = new AudioOutputI2SNoDAC();
22   out->SetGain(2); //Volume Setup:0~4.0
23   mp3 = new AudioGeneratorMP3();
24   mp3->begin(file, out);
25   pinMode(PIN_BUTTON, INPUT);
26 }
27
28 void loop()
29 {
30   if (mp3->isRunning()) {
31     if (!mp3->loop()) {
32       mp3->stop();
33       delete file;
34       delete mp3;
35       mp3 = new AudioGeneratorMP3();
36     }
37   } else {
38     if (digitalRead(PIN_BUTTON) == LOW) {
39       file = new AudioFileSourceSPIFFS("/1.mp3");
40       Serial.printf("MP3 done\n");
41       delay(1000);
42       mp3->begin(file, out);
43     }
44   }
45 }
```

Check whether the file has finished playing in the main loop function.

```

30   if (mp3->isRunning()) {
31     if (!mp3->loop()) {
32       mp3->stop();
33       delete file;
34       delete mp3;
```

```
35     mp3 = new AudioGeneratorMP3();  
36 }  
37 }
```

Play the music again after the button is pressed.

```
38 if (digitalRead(PIN_BUTTON) == LOW) {  
39     file = new AudioFileSourceSPIFFS("/1.mp3");  
40     Serial.printf("MP3 done\n");  
41     delay(1000);  
42     mp3->begin(file, out);  
43 }
```

It is important to note that the name of the uploaded audio file should correspond to the name on the code, changing the name to "1.mp3" in this tutorial. If you upload an audio file with a different name, please change the name in the code. Otherwise, ESP8266 will not play the music.

```
19 audioLogger = &Serial;  
20 file = new AudioFileSourceSPIFFS("/1.mp3");  
21 id3 = new AudioFileSourceID3(file);
```



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_Super_Starter_Kit_for_ESP8266/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.