

Welcome

Thank you for choosing Freenove products!

How to Start

When reading this, you should have downloaded the ZIP file for this product.

Unzip it and you will get a folder containing tutorials and related files. Please start with this PDF tutorial.

- ! Unzip the ZIP file instead of opening the file in the ZIP file directly.
- ! Do not move, delete or rename files in the folder just unzipped.

Get Support

Encounter problems? Don't worry! Refer to "TroubleShooting.pdf" or contact us.

When there are packaging damage, quality problems, questions encountering in use, etc., just send us an email. We will reply to you within one working day and provide a solution.

support@freenove.com

Attention

Pay attention to safety when using and storing this product:

- This product is not suitable for children under 12 years of age because of small parts and sharp parts.
- Minors should use this product under the supervision and guidance of adults.
- This product contains small and sharp parts. Do not swallow, prick and scratch to avoid injury.
- This product contains conductive parts. Do not hold them to touch power supply and other circuits.
- To avoid personal injury, do not touch parts rotating or moving while working.
- The wrong operation may cause overheat. Do not touch and disconnect the power supply immediately.
- Operate in accordance with the requirements of the tutorial. Fail to do so may damage the parts.
- Store this product in a dry and dark environment. Keep away from children.
- Turn off the power of the circuit before leaving.

Any concerns?  support@freenove.com

About

Freenove provides open source electronic products and services.

Freenove is committed to helping customers learn programming and electronic knowledge, quickly implement product prototypes, realize their creativity and launch innovative products. Our services include:

- Kits for learning programming and electronics
- Kits compatible with Arduino®, Raspberry Pi®, micro:bit®, ESP8266®, etc.
- Kits for robots, smart cars, drones, etc.
- Components, modules and tools
- Design and customization

To learn more about us or get our latest information, please visit our website:

<http://www.freenove.com>

Copyright

All the files provided in the ZIP file are released under [Creative Commons Attribution-NonCommercial-ShareAlike 3.0 Unported License](#). You can find a copy of the license in the ZIP file.



It means you can use these files on your own derived works, in part or completely. But not for commercial use.

Freenove® brand and logo are trademarks of Freenove Creative Technology Co., Ltd. Must not be used without permission.



Other registered trademarks and their owners appearing in this document:

Arduino® is a trademark of Arduino LLC (<https://www.arduino.cc/>).

Raspberry Pi® is a trademark of Raspberry Pi Foundation (<https://www.raspberrypi.org/>).

micro:bit® is a trademark of Micro:bit Educational Foundation (<https://www.microbit.org/>).

ESPRESSIF® and ESP8266® are trademarks of ESPRESSIF Systems (Shanghai) Co., Ltd (<https://www.espressif.com/>).

Any concerns? ✉ support@freenove.com

Contents

Welcome.....	i
Contents	1
Prepare.....	2
ESP8266.....	3
Chapter 0 Ready (Important)	6
0.1 Installing Thonny (Important)	6
0.2 Basic Configuration of Thonny	11
0.3 Installing CH340 (Important).....	13
0.4 Burning Micropython Firmware (Important).....	24
0.5 Testing codes (Important).....	31
0.6 Thonny Common Operation.....	39
Chapter 1 LED (Important).....	45
Project 1.1 Blink	45
Chapter 2 Serial Communication.....	54
Project 2.1 Serial Print.....	54
Project 2.2 Serial Read and Write	58
Chapter 3 WiFi Working Modes.....	60
Project 3.1 Station mode	60
Project 3.2 AP mode	65
Project 3.3 AP+Station mode	69
Chapter 4 TCP/IP	73
Project 4.1 As Client	73
Project 4.2 As Server	88
Chapter 5 Smart Home	94
Project 5.1 Control_LED_through_Web	94
What's next?	101
End of the Tutorial.....	101

Prepare

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 both either with C/C++ language or micropython language. In this tutorial, we use micropython. With Micropython is as easy to learn as Python with little code, making it ideal for beginners. 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.

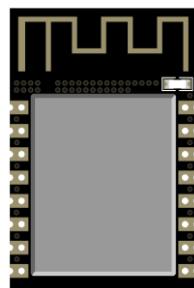
We divide each project into four parts, namely Component List, Component Knowledge, Circuit and Code. Component List helps you to prepare material for the experiment more quickly. Component Knowledge allows you to quickly understand new electronic modules or components, while Circuit helps you understand the operating principle of the circuit. And Code allows you to easily master the use of ESP8266 and its accessory kit. After finishing all the projects in this tutorial, you can also use these components and modules to make products such as smart household, smart cars and robots to transform your creative ideas into prototypes and new and innovative products.

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

ESP8266

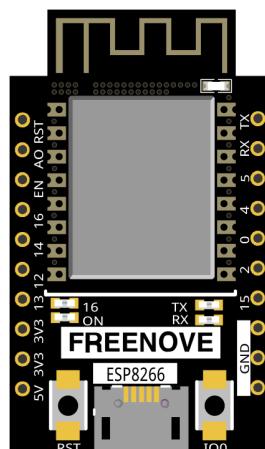
ESP8266 has PCB on-board antenna. The PCB on-board antenna is an integrated antenna in the chip module itself, so it is convenient to carry and design.

PCB on-board antenna

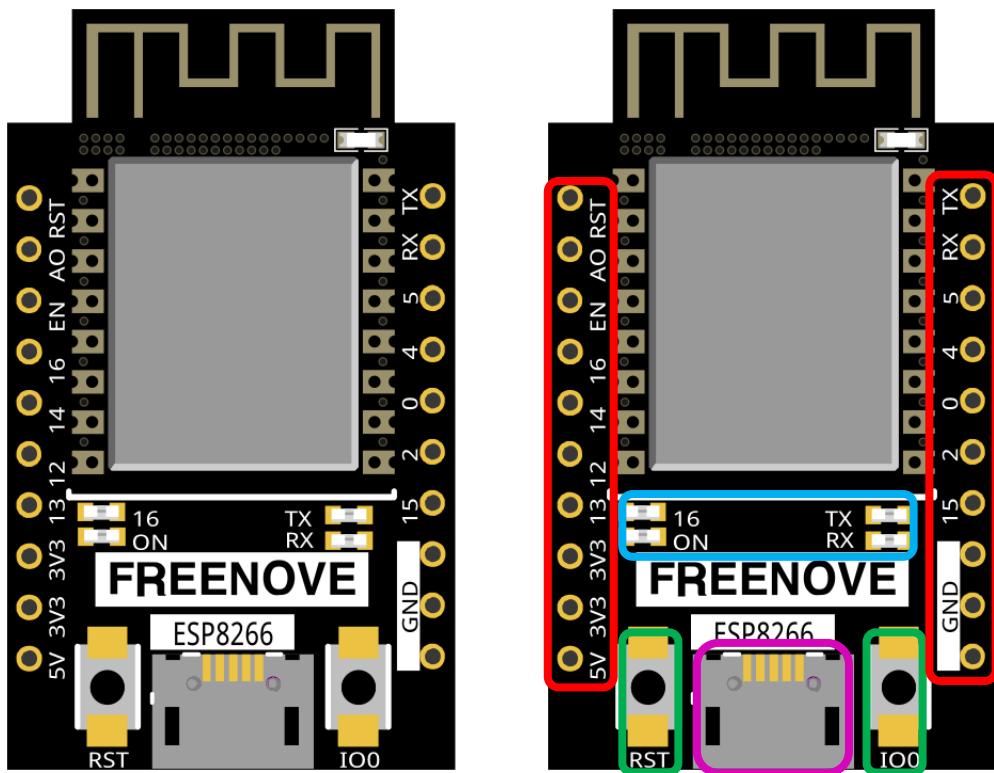


In this tutorial, the ESP8266 development board is designed based on the PCB on-board antenna-packaged ESP8266 module. The following tutorials will be based on the ESP8266 development board.

ESP8266 development board



The hardware interfaces of ESP8266 are distributed as follows:



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

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

NO.	Pin Name	Functional Description
1	RST	Reset Pin, Active Low
2	ADC	AD conversion, Input voltage range 0~3.3V, the value range is 0~1024.
3	EN	Chip Enabled Pin, Active High
4	IO16	Connect with RST pin to wake up Deep Slee
5	IO14	GPIO14; HSPI_CLK
6	IO12	GPIO12; HSPI_MISO
7	IO13	GPIO13; HSPI_MOSI; UART0_CTS
8	VCC	Module power supply pin, Voltage 3.0V ~ 3.6V
9	GND	GND
10	IO15	GPIO15; MTDO; HSPICS; UART0_RTS
11	IO2	GPIO2; UART1_RXD
12	IO0	GPIO0;HSPI_MISO;I2SI_DATA
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.

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

["Freenove_ESP8266_Board/Datasheet/esp-12s_datasheet_en.pdf"](#)

Chapter 0 Ready (Important)

Before starting building the projects, you need to make some preparation first, which is so crucial that you must not skip.

0.1 Installing Thonny (Important)

Thonny is a free, open-source software platform with compact size, simple interface, simple operation and rich functions, making it a Python IDE for beginners. In this tutorial, we use this IDE to develop ESP6266 during the whole process.

Thonny supports various operating system, including Windows、Mac OS、Linux.

Downloading Thonny

Official website of Thonny: <https://thonny.org>

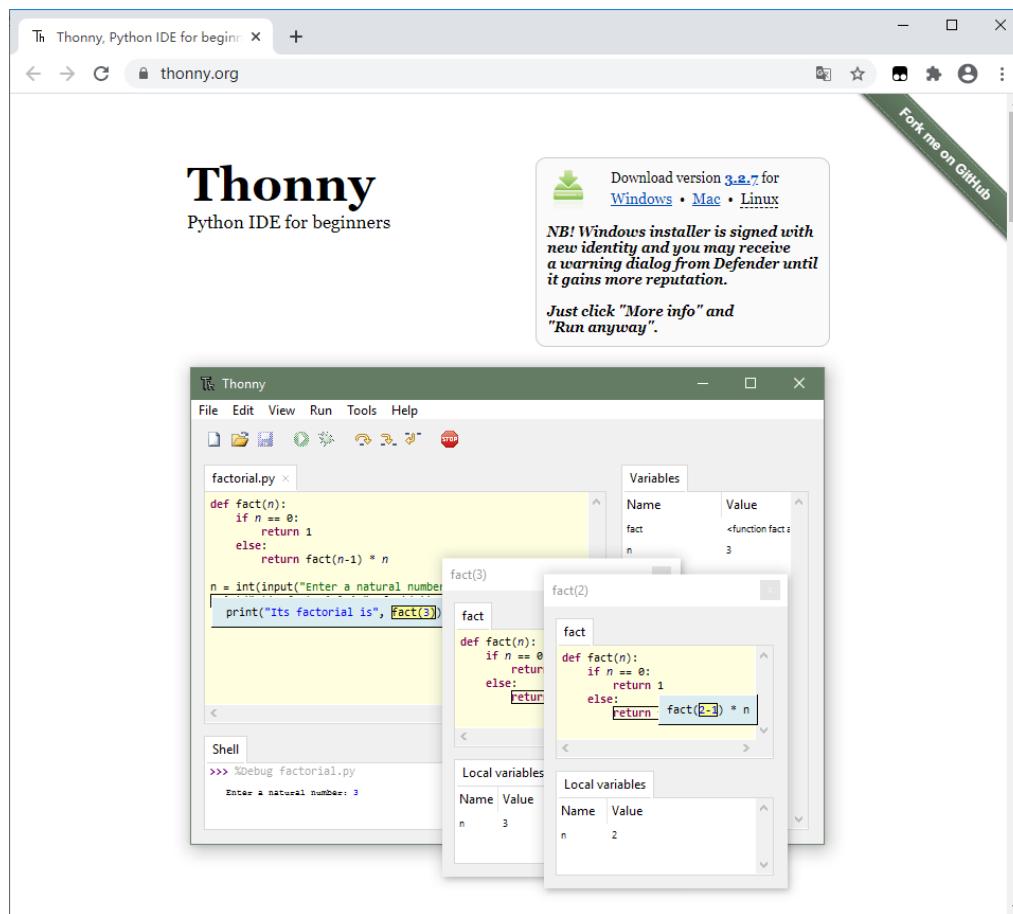
Open-source code repositories of Thonny: <https://github.com/thonny/thonny>

Follow the instruction of official website to install Thonny or click the links below to download and install.
(Select the appropriate one based on your operating system.)

Operating System	Download links/methods
Windows	https://github.com/thonny/thonny/releases/download/v3.2.7/thonny-3.2.7.exe
Mac OS	https://github.com/thonny/thonny/releases/download/v3.2.7/thonny-3.2.7.pkg
Linux	The latest version: Binary bundle for PC (Thonny+Python): bash <(wget -O - https://thonny.org/installer-for-linux) With pip: pip3 install thonny Distro packages (may not be the latest version): Debian, Raspbian, Ubuntu, Mint and others: sudo apt install thonny Fedoras: sudo dnf install thonny

You can also open “**Freenove_ESP8266_Board/Python/Python_Software**”, we have prepared it in advance.

Any concerns? ✉ support@freenove.com



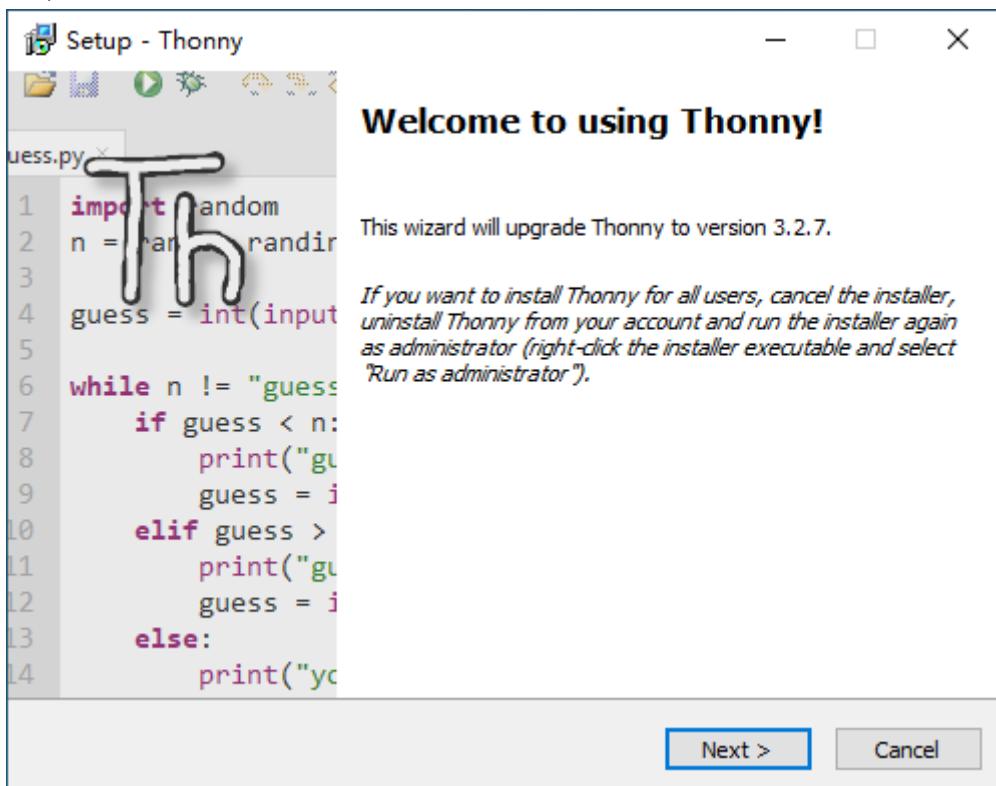
Installing on Windows

The icon of Thonny after downloading is as below. Double click “thonny-3.2.7.exe”.



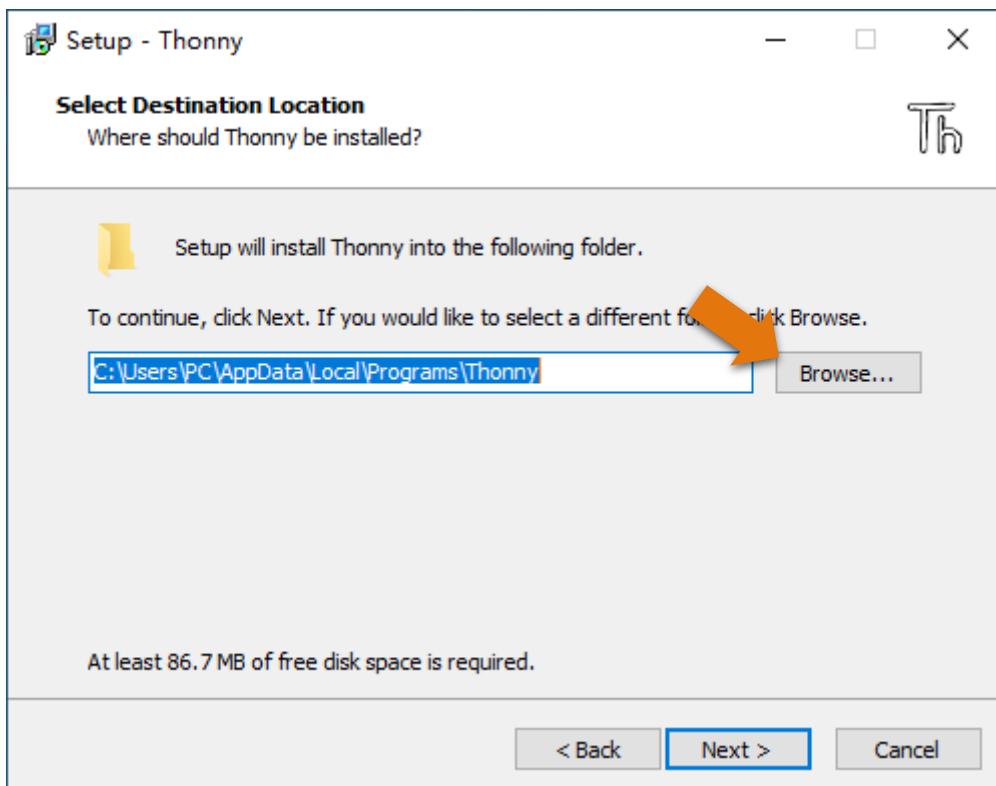


If you're not familiar with computer software installation, you can simply keep clicking "Next" until the installation completes.

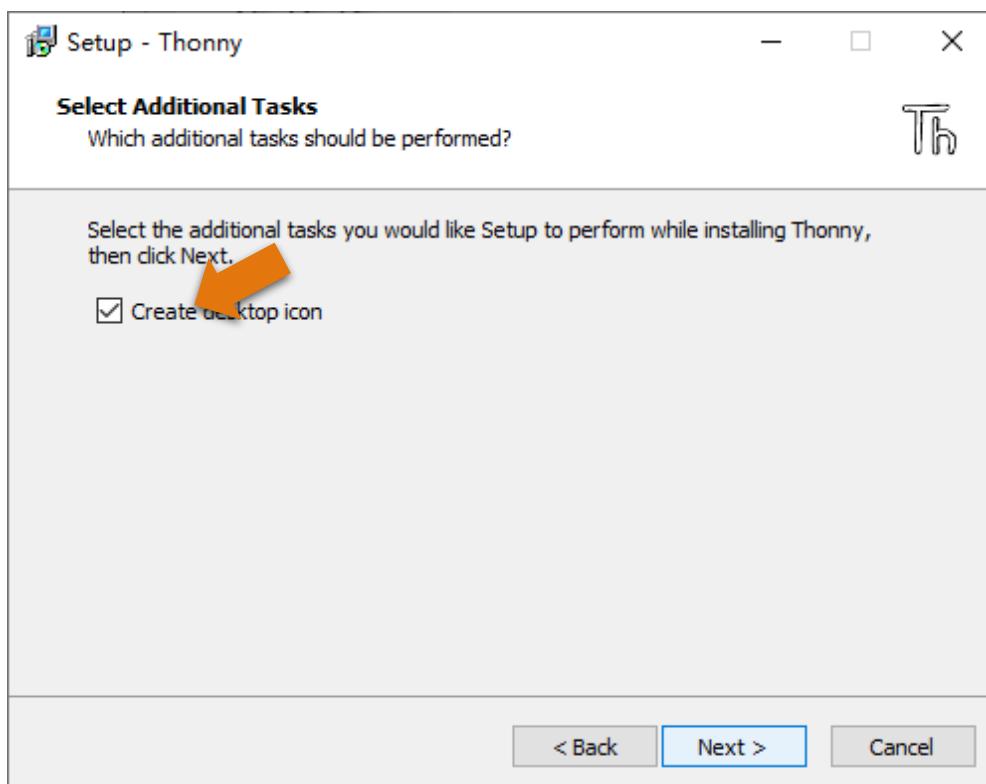


If you want to change Thonny's installation path, you can click "Browse" to modify it. After selecting installation path, click "OK".

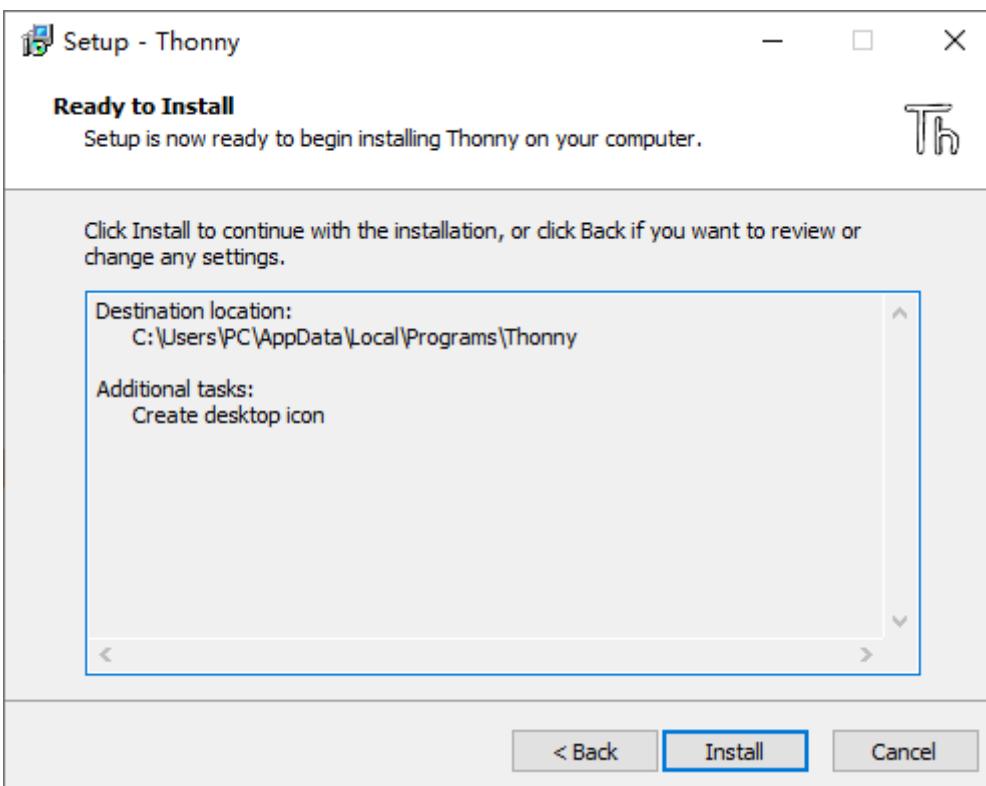
If you do not want to change it, just click "Next".



Check “Create desktop icon” and then it will generate a shortcut on your desktop to facilitate you to open Thonny later.

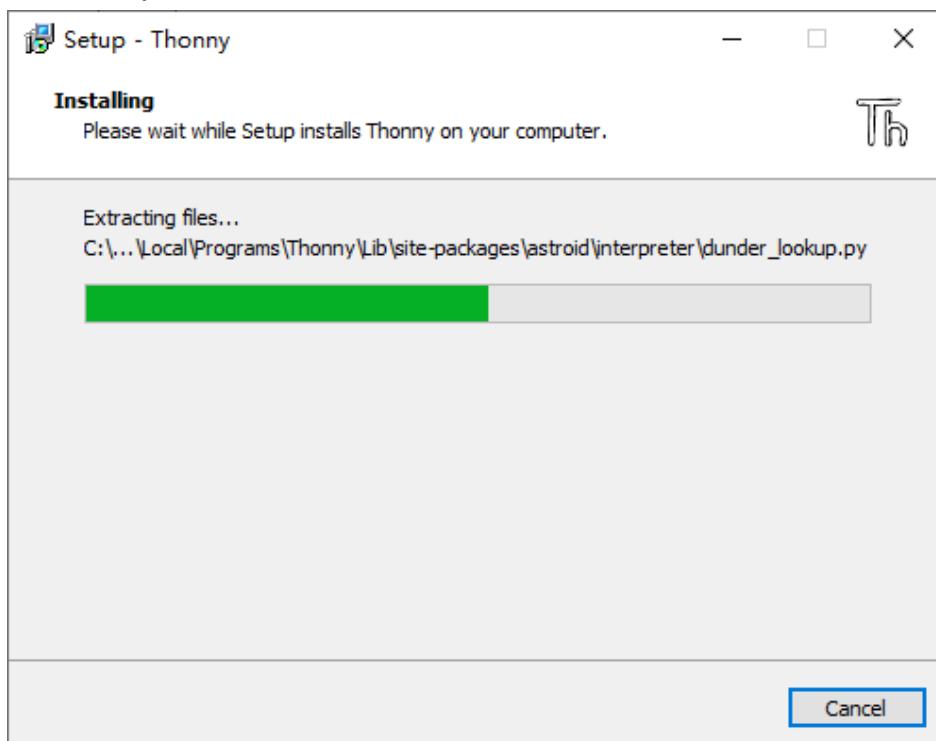


Click “install” to install the software.





During the installation process, you only need to wait for the installation to complete, and you must not click "Cancel", otherwise Thonny will fail to be installed.



Once you see the interface as below, Thonny has been installed successfully.



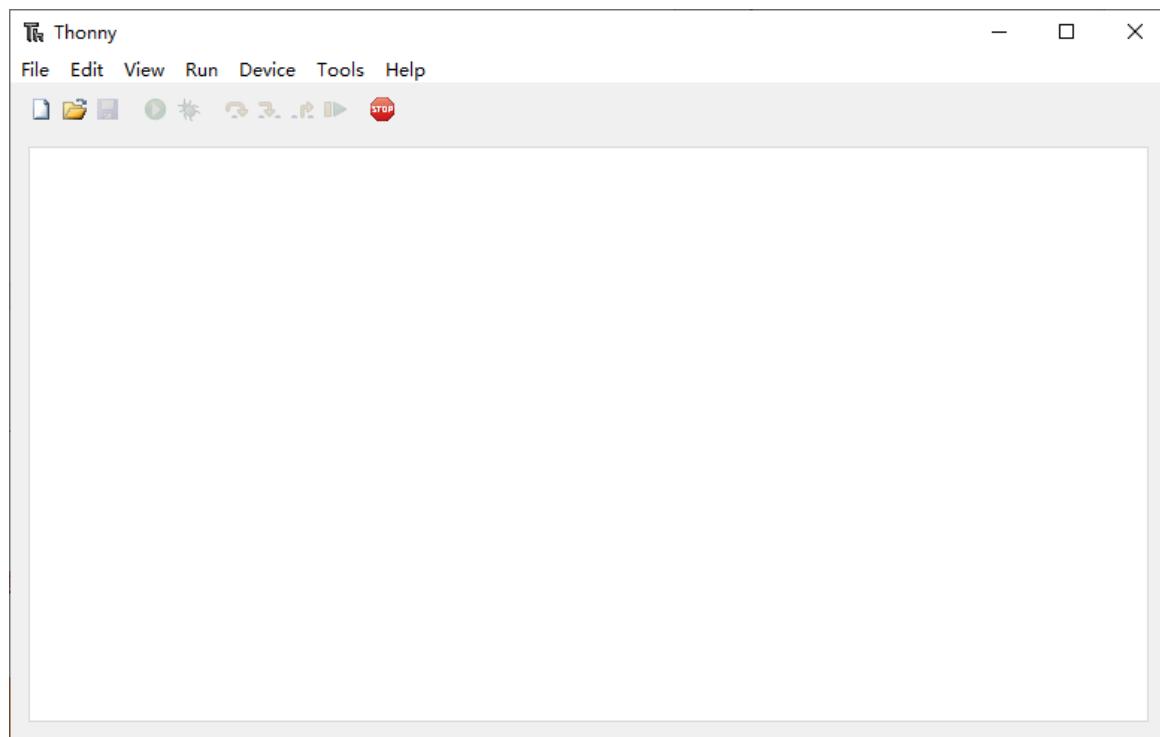
If you've checked "Create desktop icon" during the installation process, you can see the below icon on your desktop.



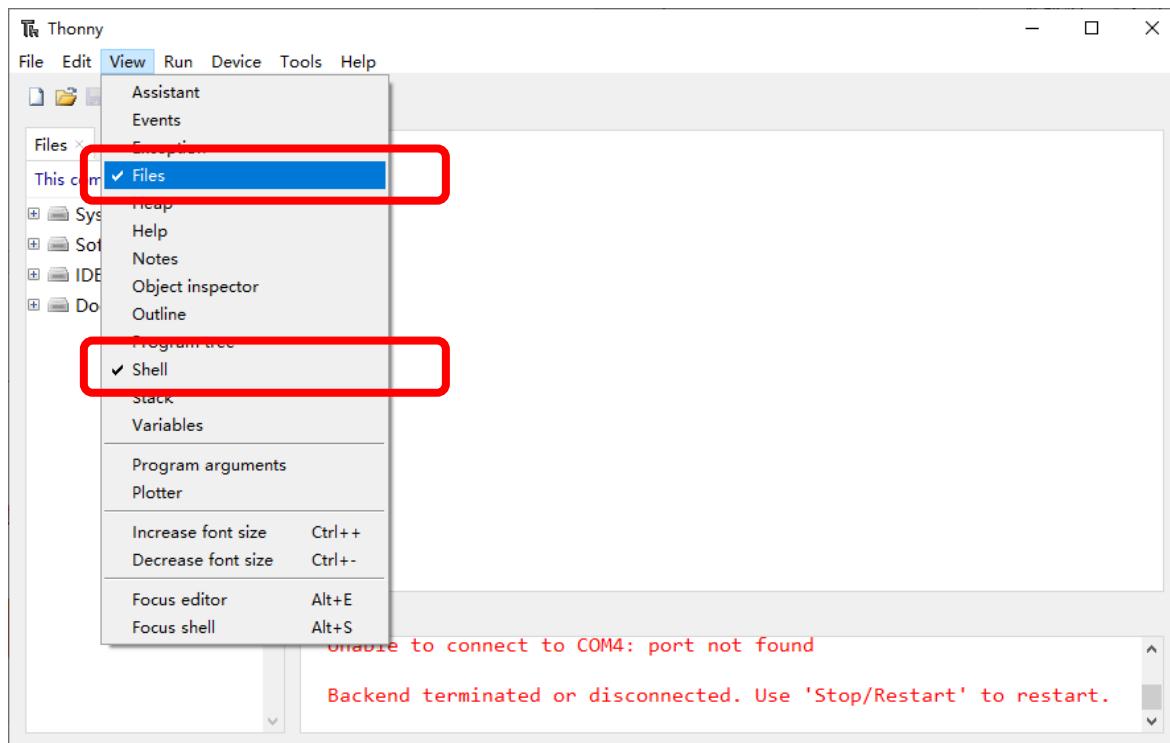
Any concerns? [✉ support@freenove.com](mailto:support@freenove.com)

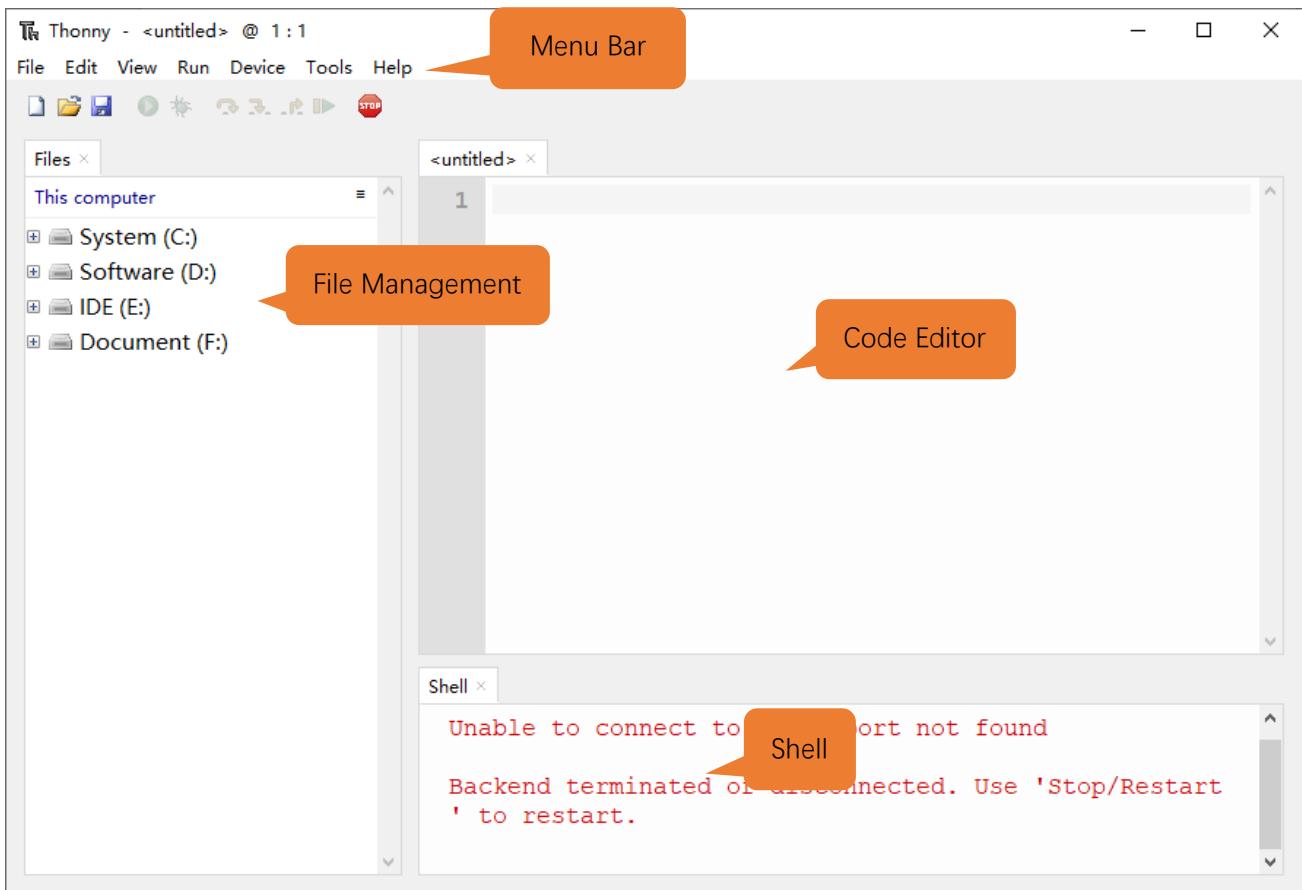
0.2 Basic Configuration of Thonny

Click the desktop icon of Thonny and you can see the interface of it as follows:



Select "View" → "Files" and "Shell".





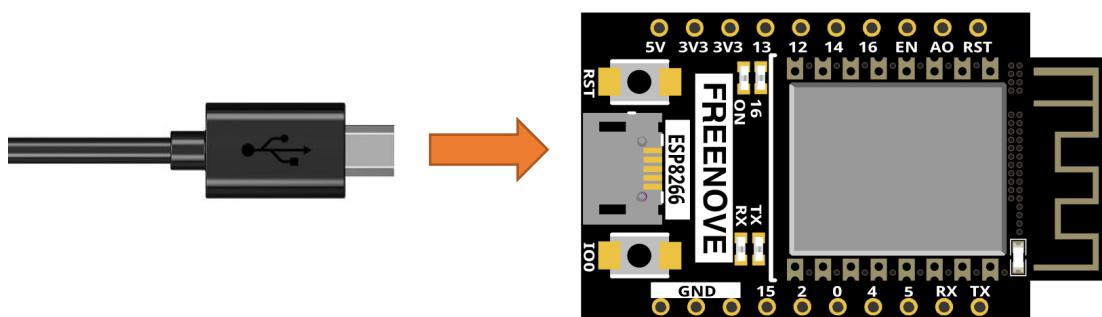
0.3 Installing CH340 (Important)

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

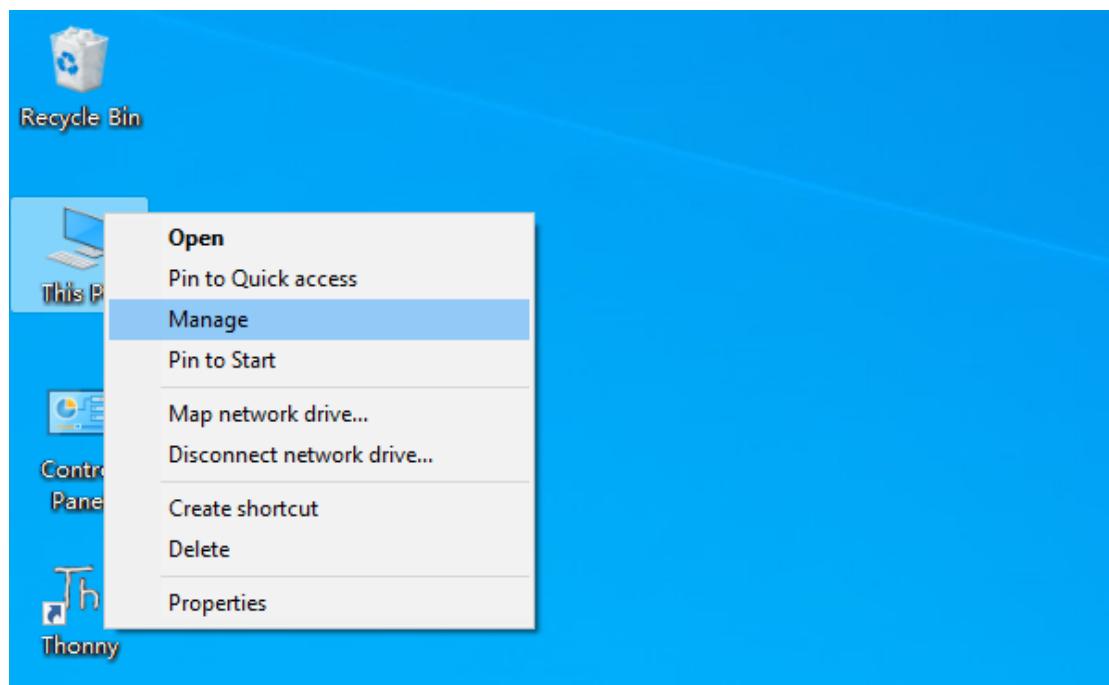
Windows

Check whether CH340 has been installed

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

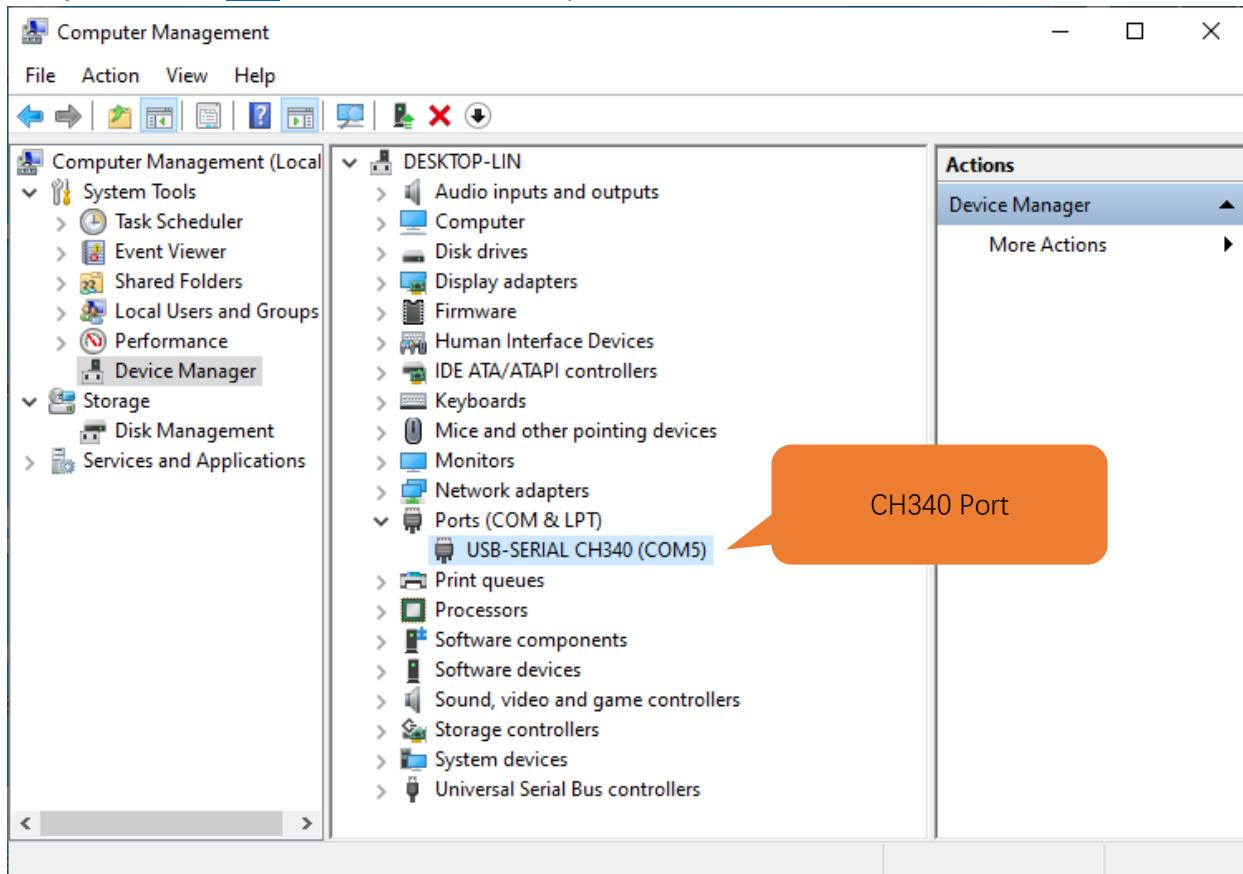


2. Turn to the main interface of your computer, select "This PC" and right-click to select "Manage".





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

- 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 a website. The left sidebar has categories: All (14), Downloads (7) [highlighted in blue], Products (4), Application (2), Video (1), and News (0). The main area is titled 'keyword CH340' and shows 'Downloads(7)'. A table lists the files:

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 (linux Driver), App Demo Example (USB to UART Demo)	1.6	2019-04-19
CH341SER_LINUX...	CH340/CH341 USB to serial port LINUX driver	1.5	2018-03-18
CH341SER_MAC.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

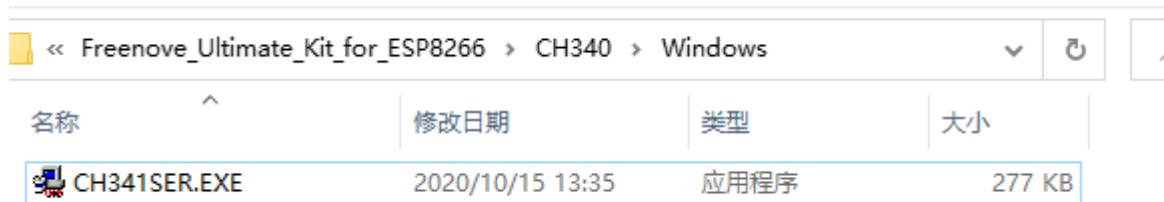
Three orange callout boxes point to specific rows: 'Windows' points to the first two rows, 'Linux' points to the fourth row, and 'MAC' points to the fifth row.

You can also open “Freenove_ESP8266_Board/CH340”, we have prepared the installation package.

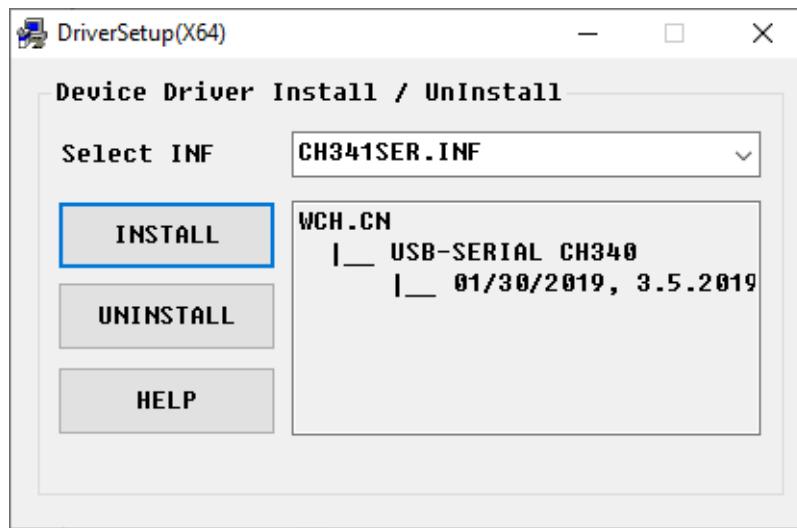
Name	Date modified	Type	Size
Linux	8/14/2020 5:24 PM	File folder	
MAC	8/14/2020 5:23 PM	File folder	
Windows	8/14/2020 5:23 PM	File folder	



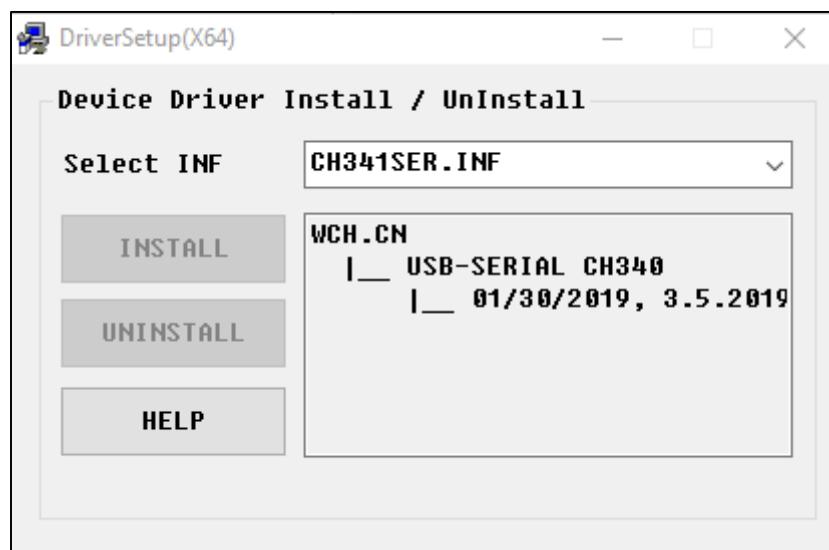
2. Open the folder “**Freenove_ESP8266_Board/CH340/Windows/ch341ser**”



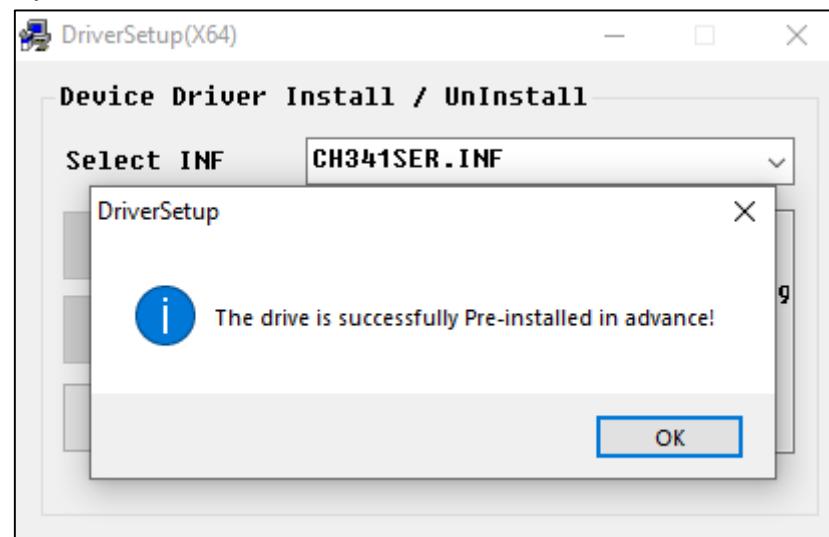
3. Double click “CH341SER.EXE”.



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

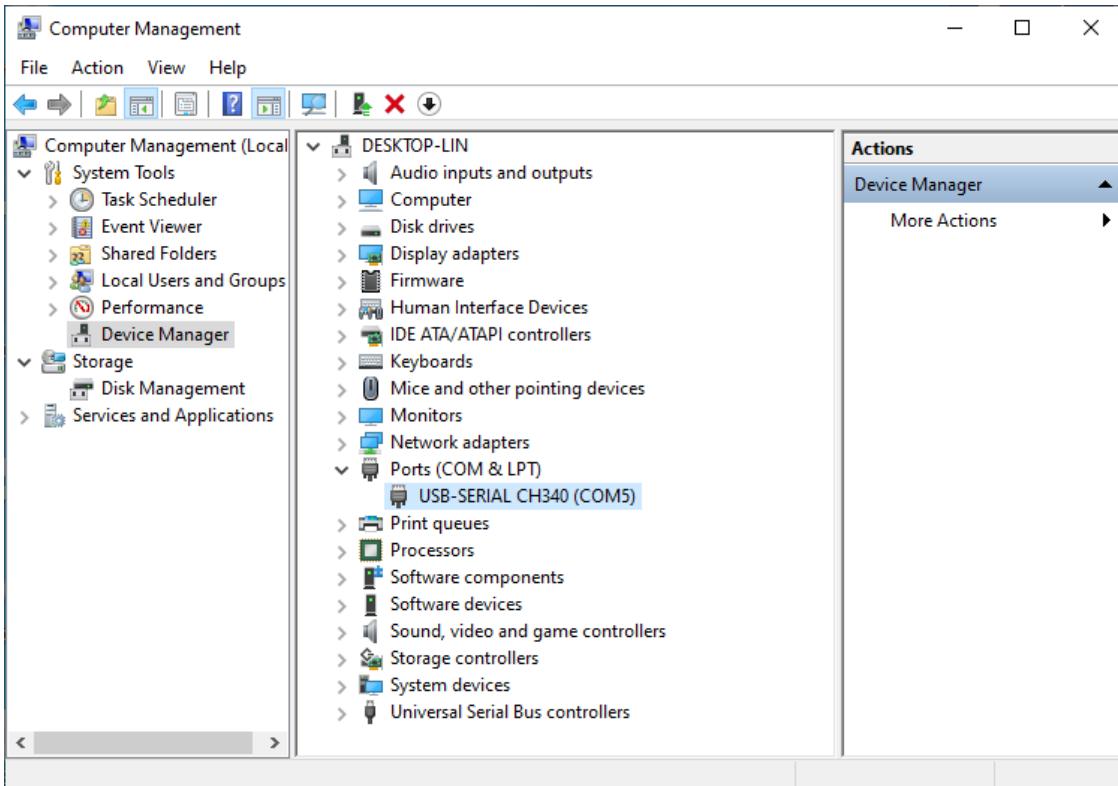


5. Install successfully. Close all interfaces.





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.

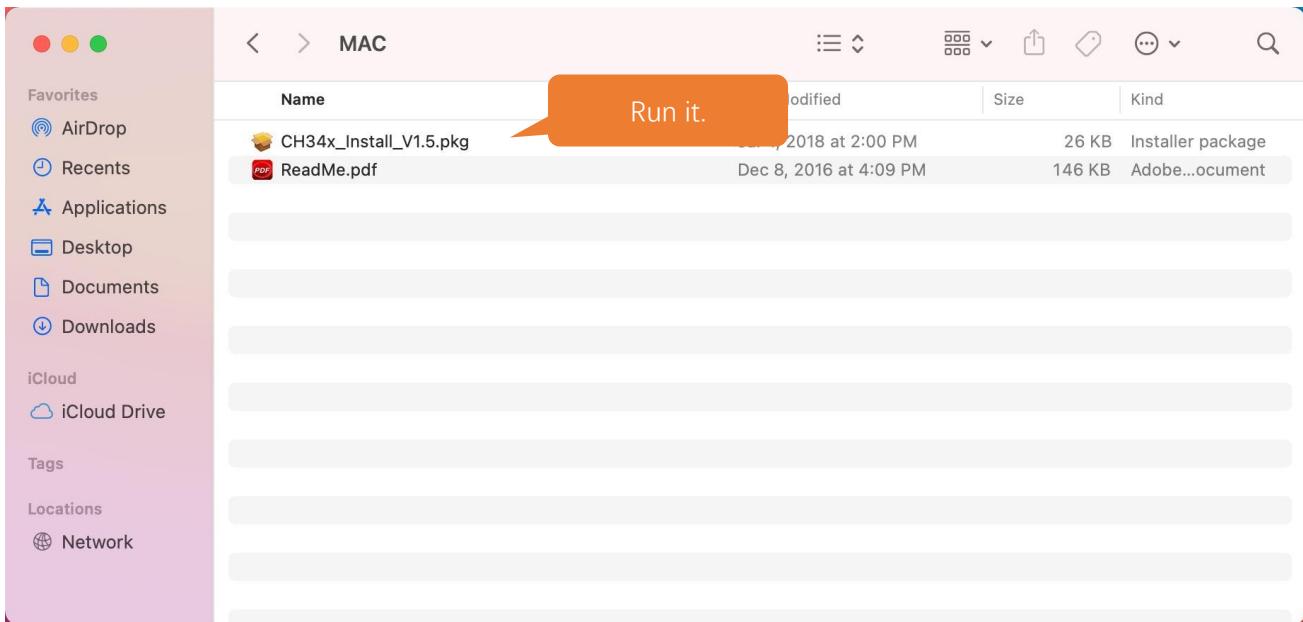
MAC

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.

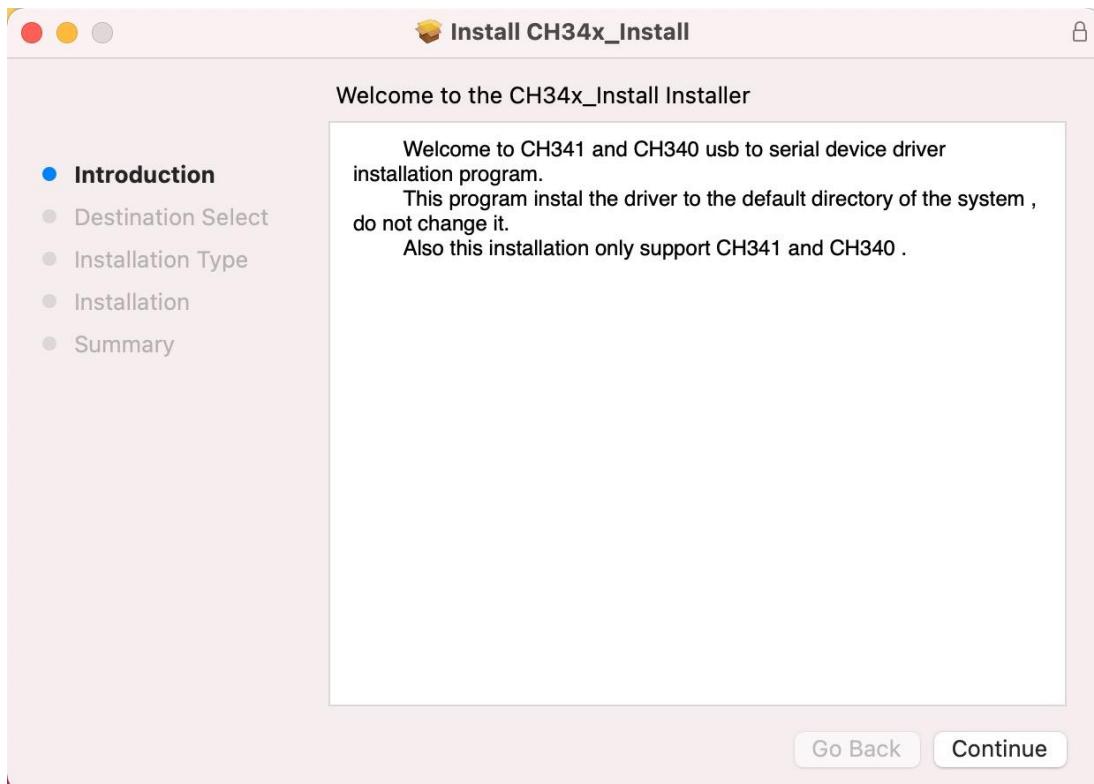
file category	file content	version	upload time
Driver&Tools	CH341SER.EXE CH340/CH341 USB to serial port Windows driver, supports 32/64-bit Windows 10/8.1/8/7/VISTA/XP, Server 2016/2012/2008/2003, 2000/ME/98	3.5	2019-03-18
	CH341SER.ZIP CH340/CH341 USB to serial port Windows driver, includes DLL dynamic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

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

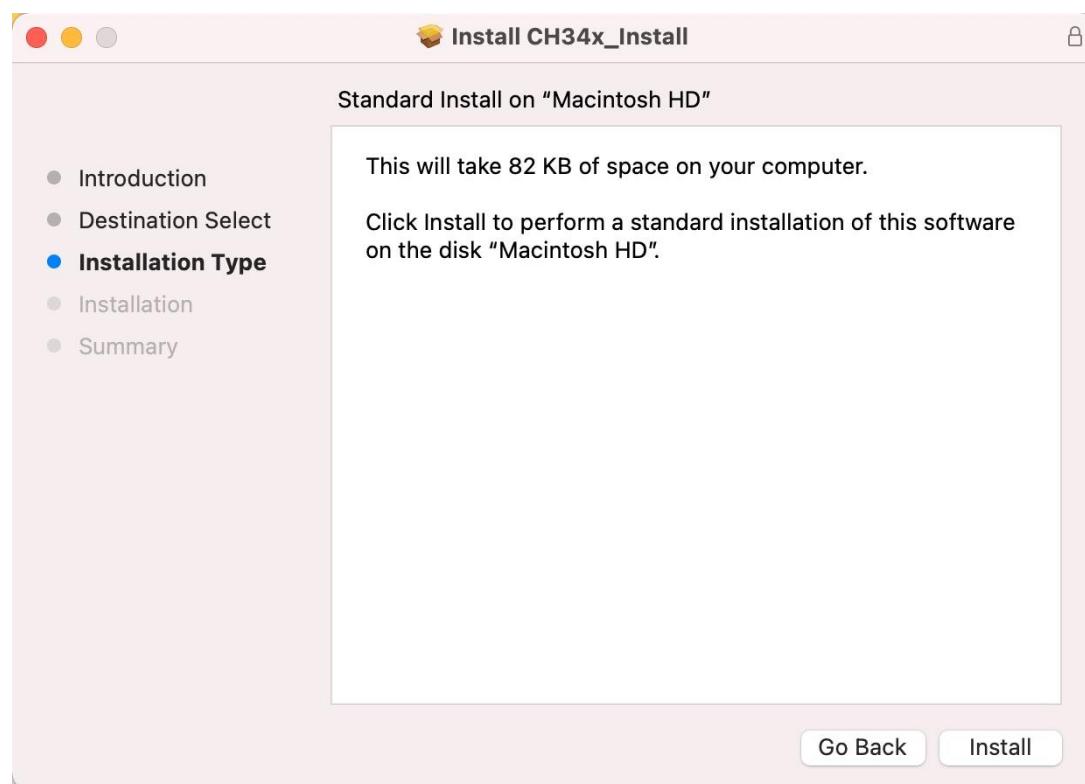
Second, open the folder “**Freenove_ESP8266_Board/CH340/MAC/**”



Third, click Continue.

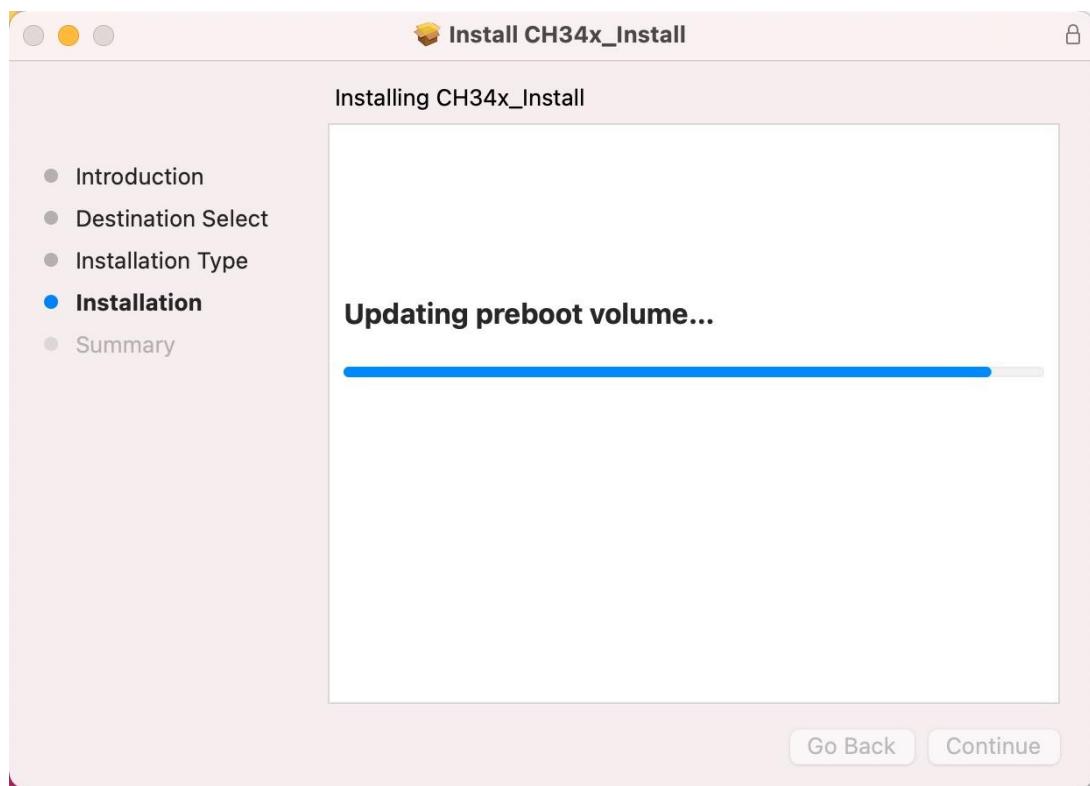


Fourth, click Install.

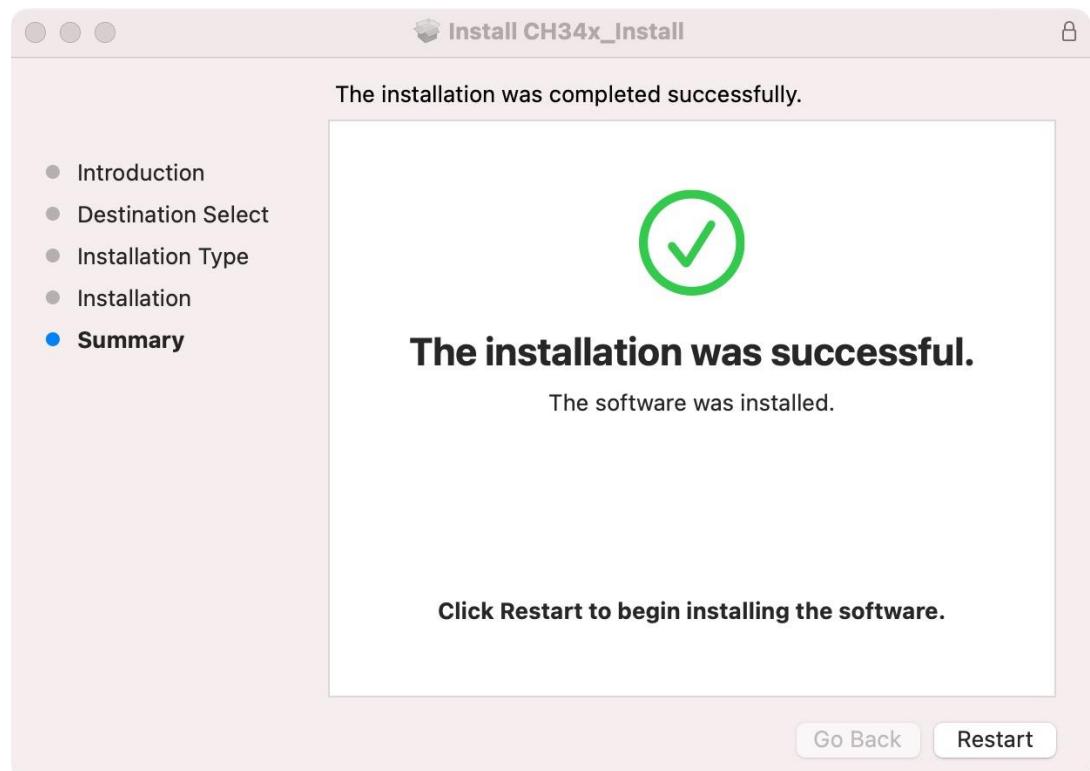




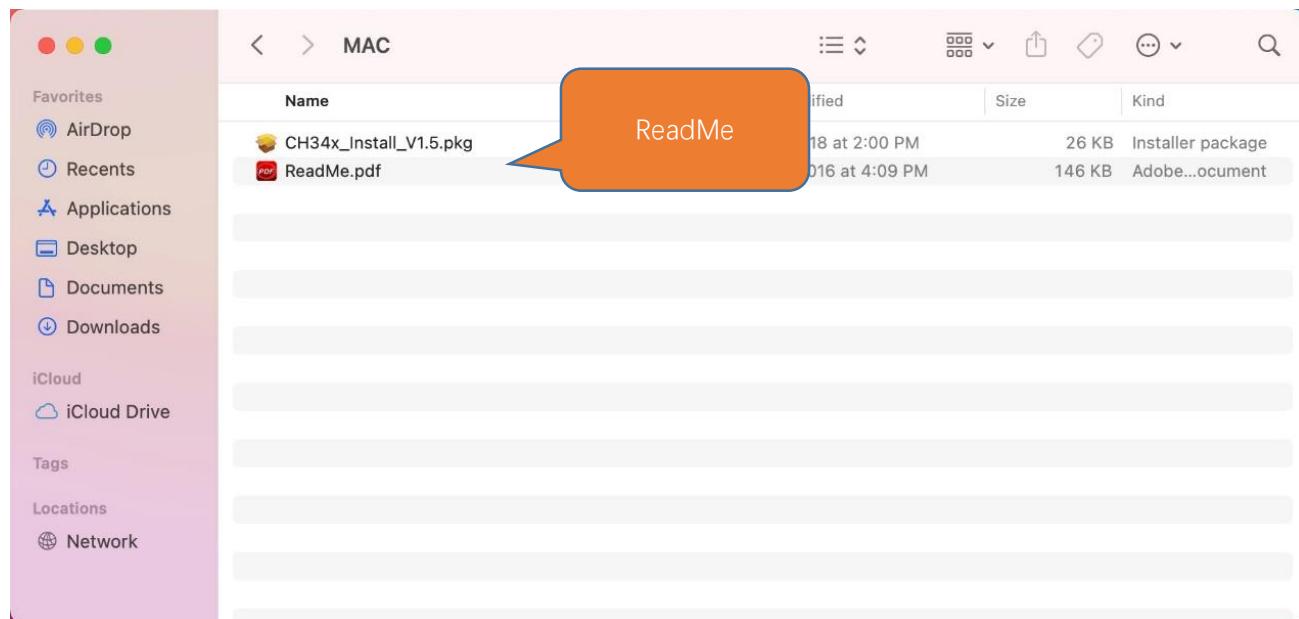
Then, waiting Finsh.



Finally, restart your PC.



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





0.4 Burning Micropython Firmware (Important)

To run Python programs on ESP8266, we need to burn a firmware to ESP8266 first.

Downloading Micropython Firmware

Official website of microPython: <http://micropython.org/>

Webpage listing firmware of microPython for ESP8266: <https://micropython.org/download/esp8266/>

Firmware

Releases

v1.18 (2022-01-17) .bin [.elf] [.map] [Release notes] (latest)

v1.17 (2021-09-02) .bin [.elf] [.map] [Release notes]
v1.16 (2021-06-18) .bin [.elf] [.map] [Release notes]
v1.15 (2021-04-18) .bin [.elf] [.map] [Release notes]
v1.14 (2021-02-02) .bin [.elf] [.map] [Release notes]
v1.13 (2020-09-11) .bin [.elf] [.map] [Release notes]
v1.12 (2019-12-20) .bin [.elf] [.map] [Release notes]
v1.11 (2019-05-29) .bin [.elf] [.map] [Release notes]
v1.10 (2019-01-25) .bin [.elf] [.map] [Release notes]
v1.9.4 (2018-05-11) .bin [.elf] [.map] [Release notes]
v1.9.3 (2017-11-01) .bin [.elf] [.map] [Release notes]
v1.9.2 (2017-08-23) .bin [.elf] [.map] [Release notes]
v1.9.1 (2017-06-12) .bin [.elf] [.map] [Release notes]
v1.9 (2017-05-26) .bin [.elf] [.map] [Release notes]
v1.8.7 (2017-01-08) .bin [.elf] [.map] [Release notes]

Firmware used in this tutorial is **esp8266-20220117-v1.18.bin**

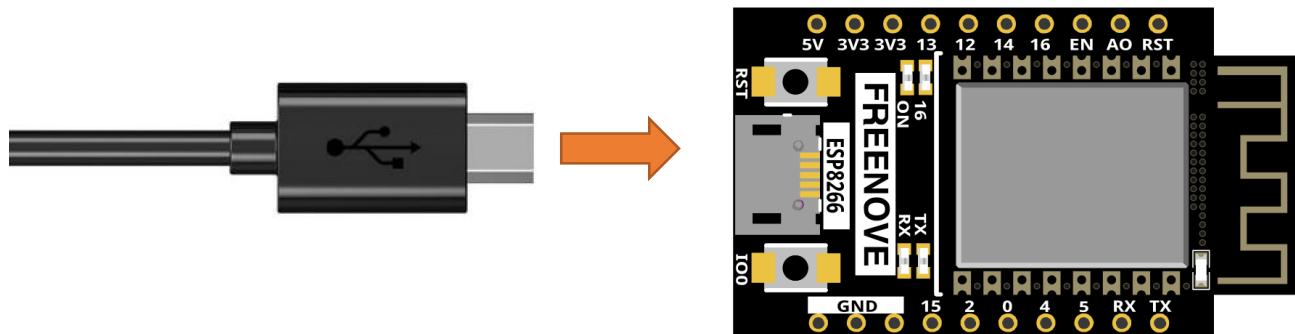
Click the following link to download directly:

<https://micropython.org/resources/firmware/esp8266-20220117-v1.18.bin>

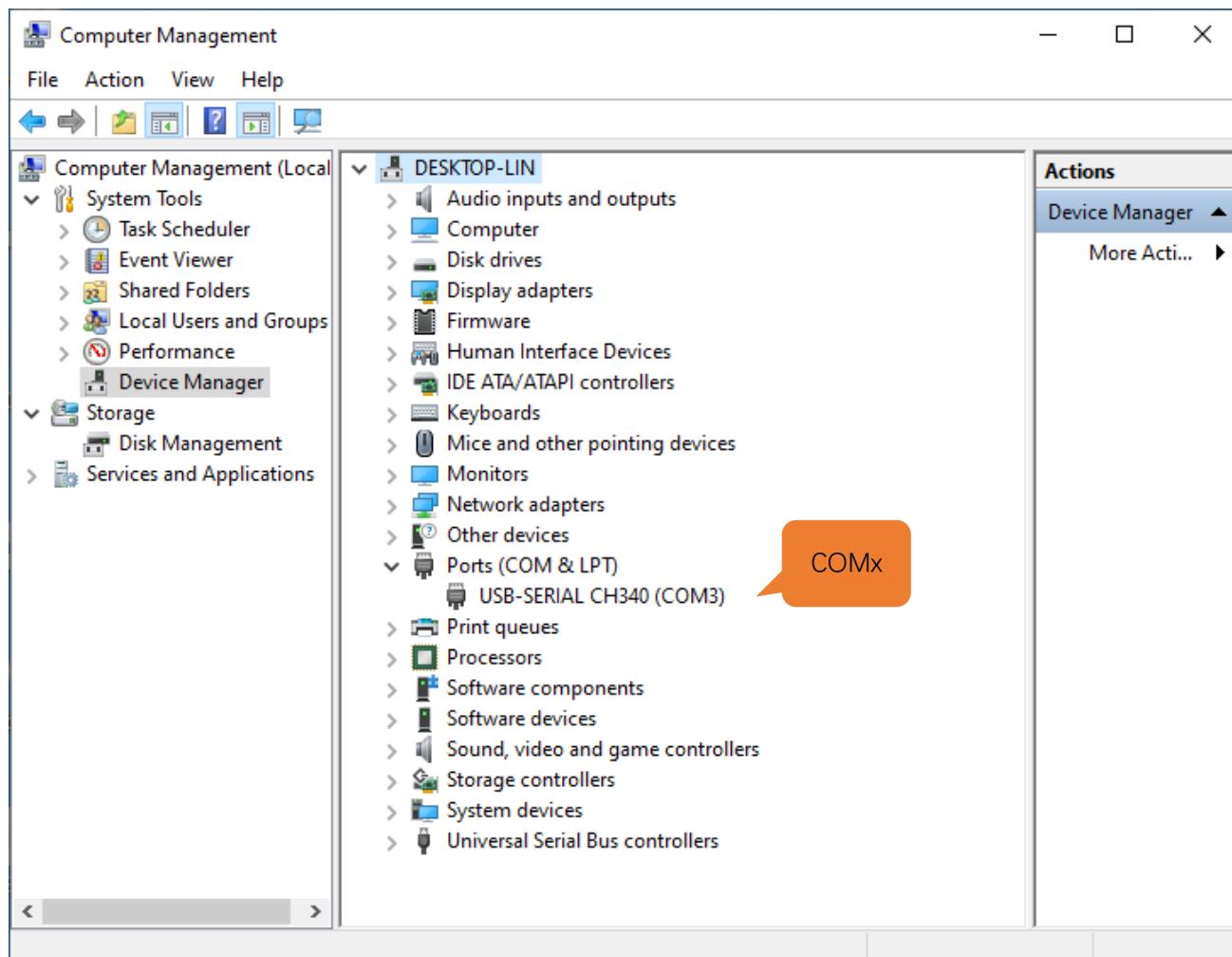
This file is also provided in our data folder "**Freenove_ESP8266_Board /Python/Python_Firmware**".

Burning a Micropython Firmware

Connect your computer and ESP8266 with a USB cable.

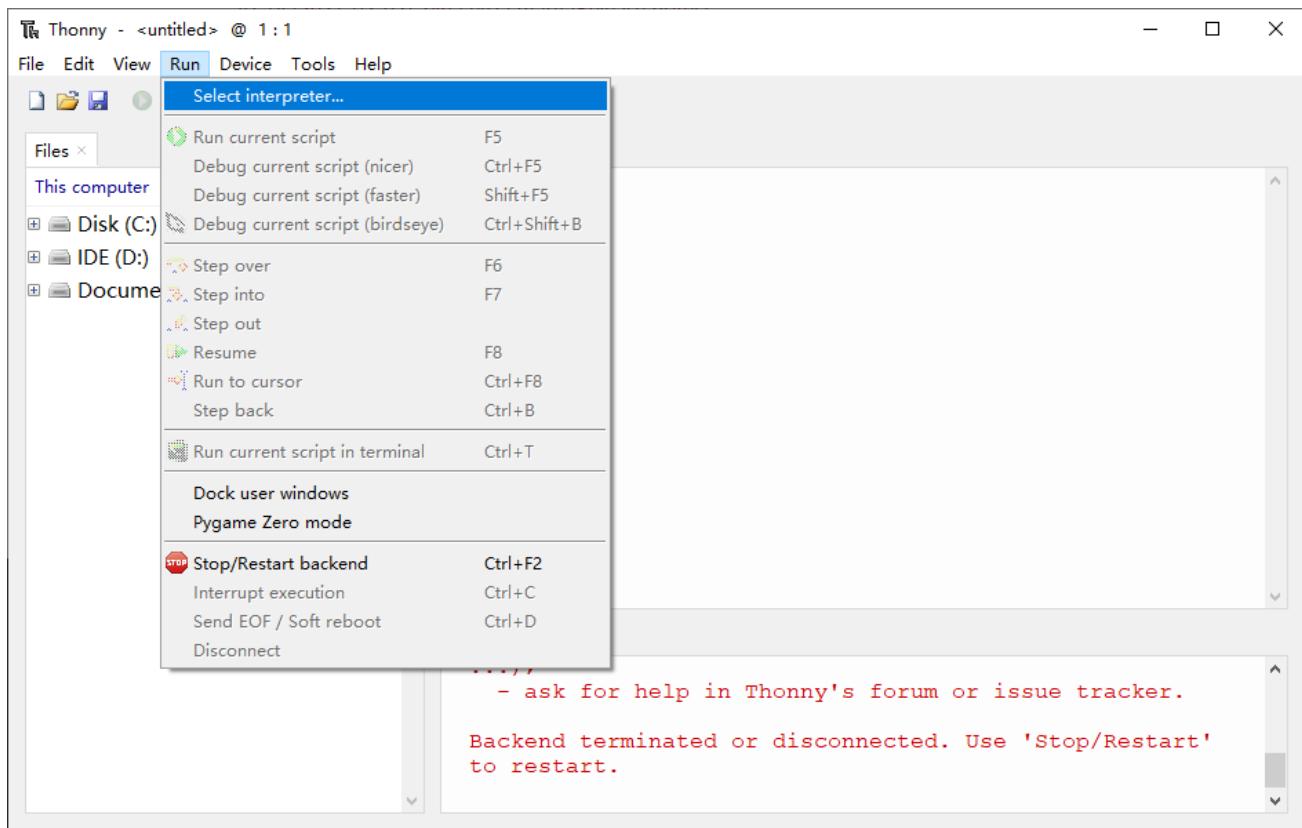


Make sure that the driver has been installed successfully and that it can recognize COM port correctly. Open device manager and expand "Ports".

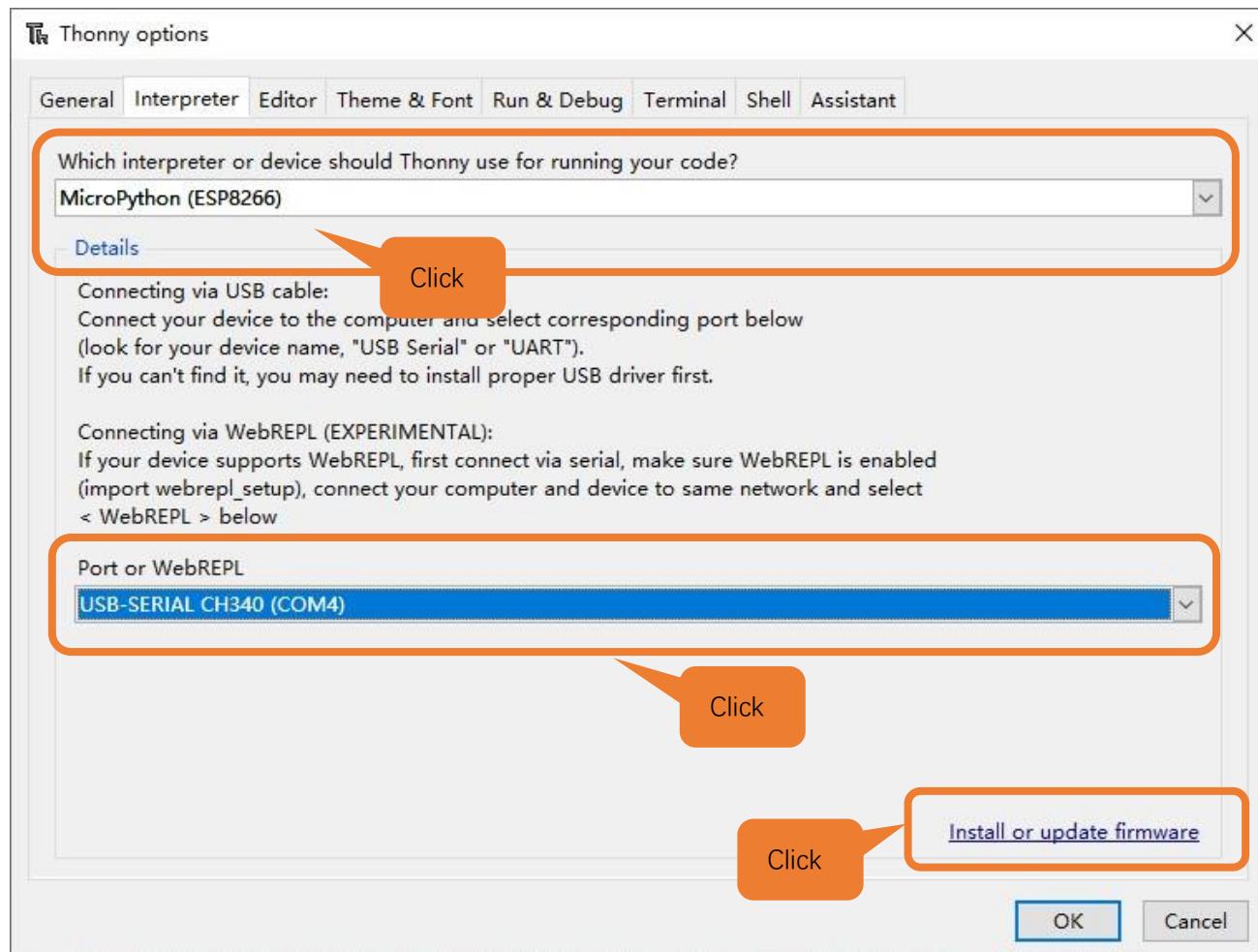


Note: the port of different people may be different, which is a normal situation.

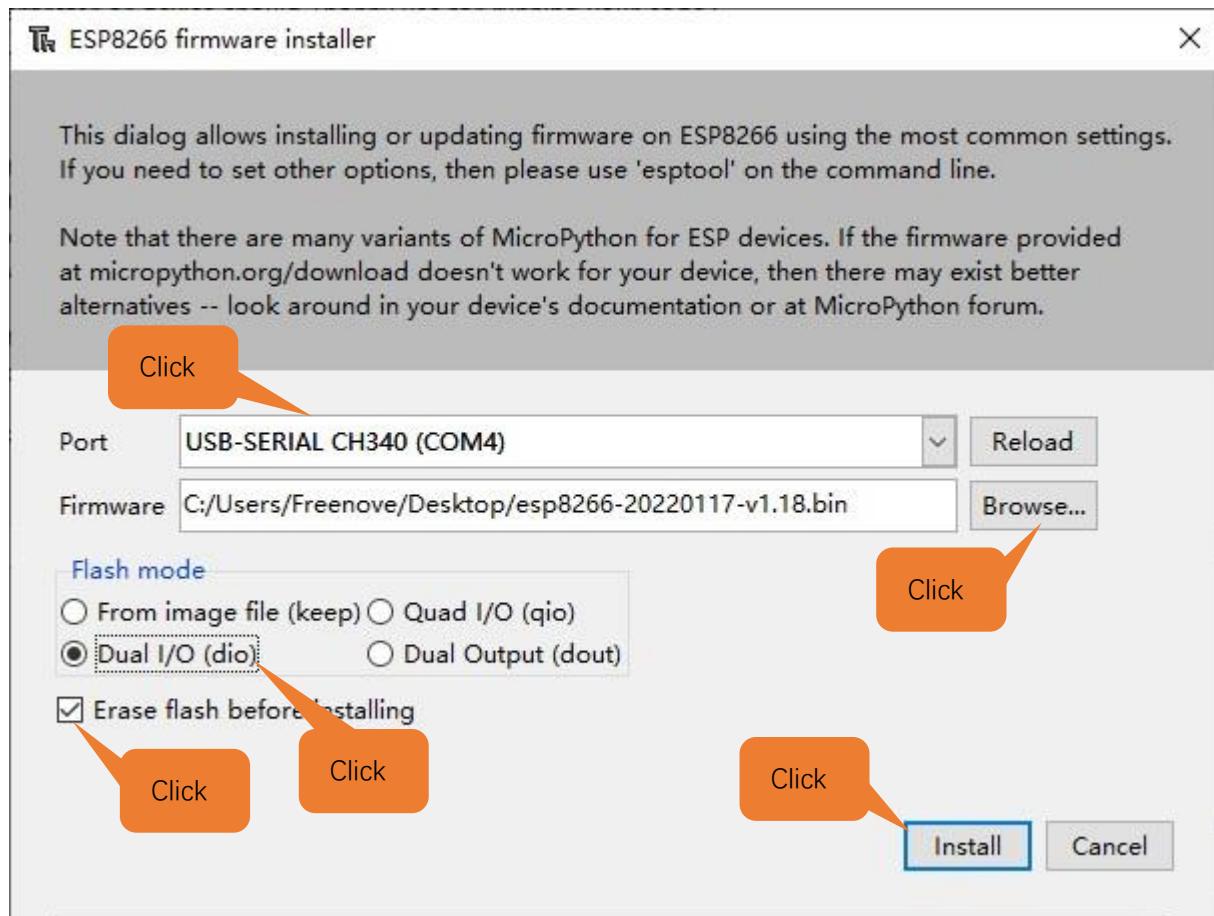
1. Open Thonny, click "run" and select "Select interpreter..."



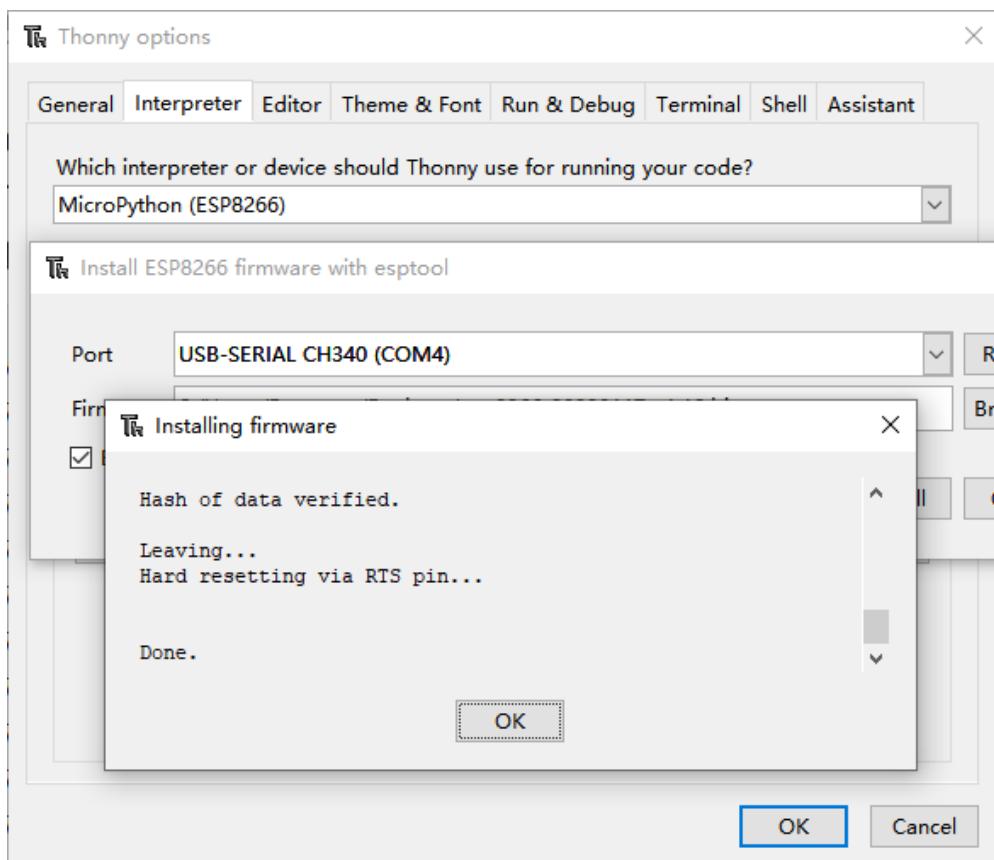
2. Select “Micropython (ESP8266)”, select “USB-SERIAL CH340 (COM4)”, and then click the long button under “Firmware”.



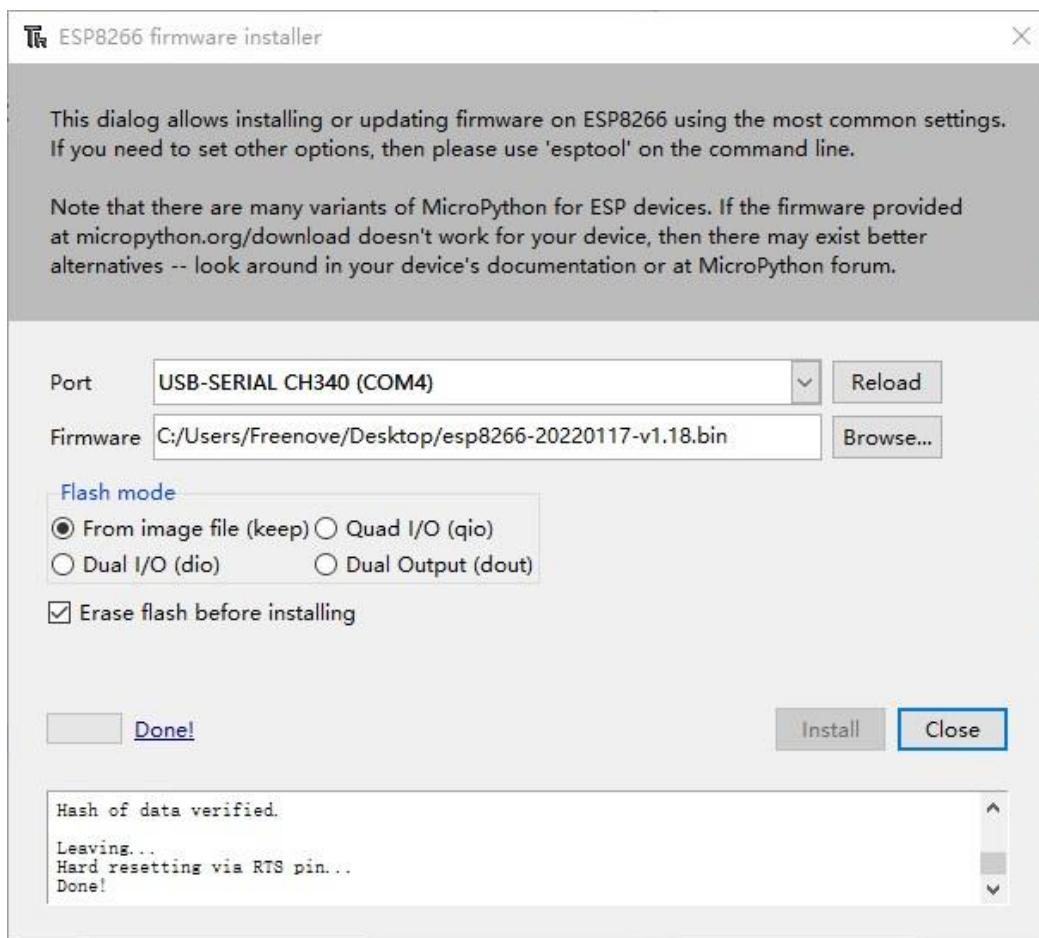
3. The following dialog box pops up. Select “USB-SERIAL CH340 (COM4)” for “Port” and then click “Browse...”. Select the previous prepared microPython firmware “**esp8266-20220117-v1.18.bin**”. Check “Erase flash before installing” and click “install” to wait for the prompt of finishing installation. 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.



- Wait for the installation to be done.



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



After burning the MicroPython firmware, "shell" will display some garbled characters, please do not worry, the garbled characters are displayed as follows:



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

```

COM4
| Send

ets Jan 8 2013, rst cause:1, boot mode:(3,7)

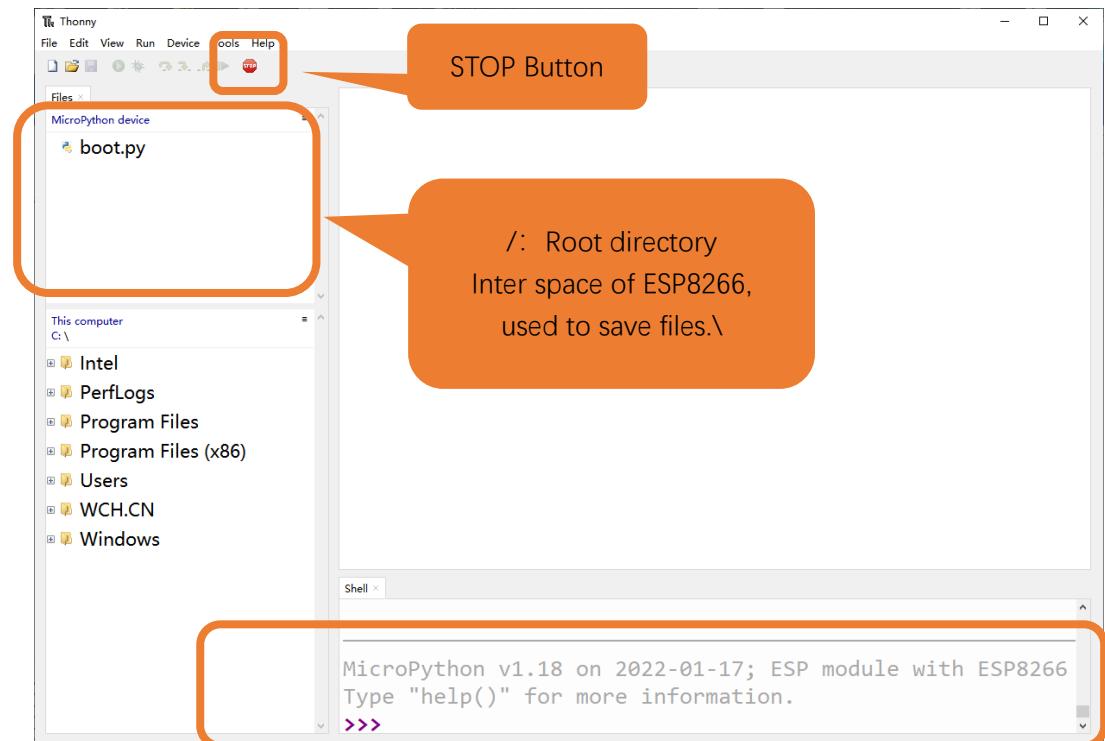
load 0x40100000, len 30720, room 16
tail 0
chksum 0x0f
load 0x3ffe8000, len 1012, room 8
tail 12
chksum 0x00
ho 0 tail 12 room 4
load 0x3ffe8400, len 1080, room 12
tail 12
checksum 0x87
cs 0x87
rf cal sector: 1019
freq trace enable 0
rf[112] : 00
rf[113] : 00
rf[114] : 01

SDK ver: 2.2.0-dev(9422289) compiled @ Nov 3 2017 19:40:05
phy ver: 1136_0, pp ver: 10.2

 Autoscroll  Show timestamp
Newline 74880 baud Clear output

```

- Close all dialog boxes, turn to main interface and click “STOP”. As shown in the illustration below. Ignore the garbled part here.



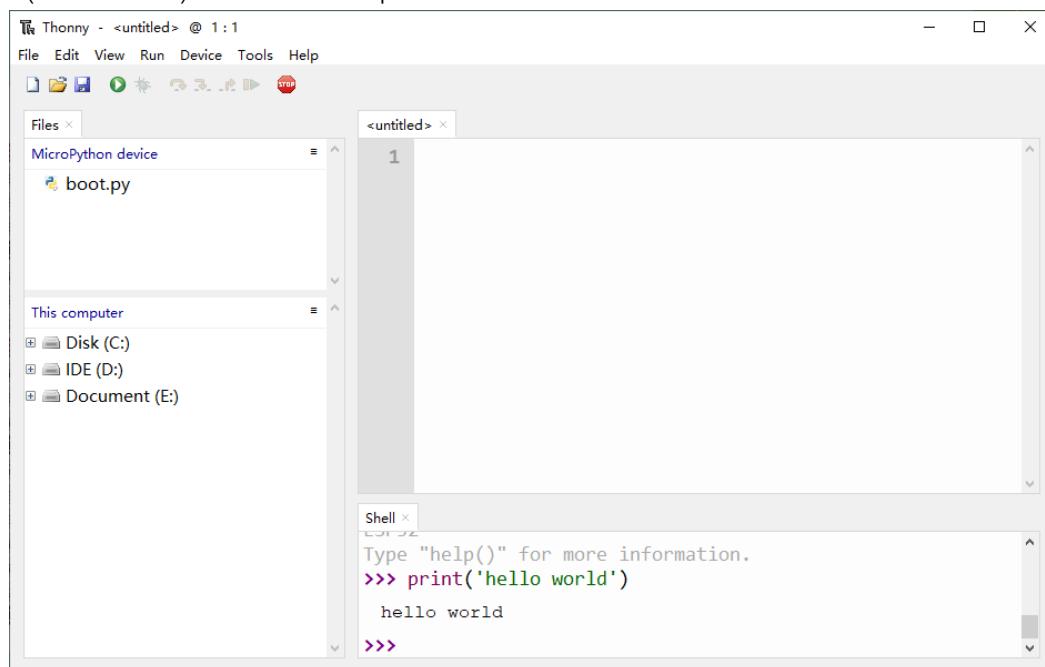
- So far, all the preparations have been made.

Any concerns? ✉ support@freenove.com

0.5 Testing codes (Important)

Testing Shell Command

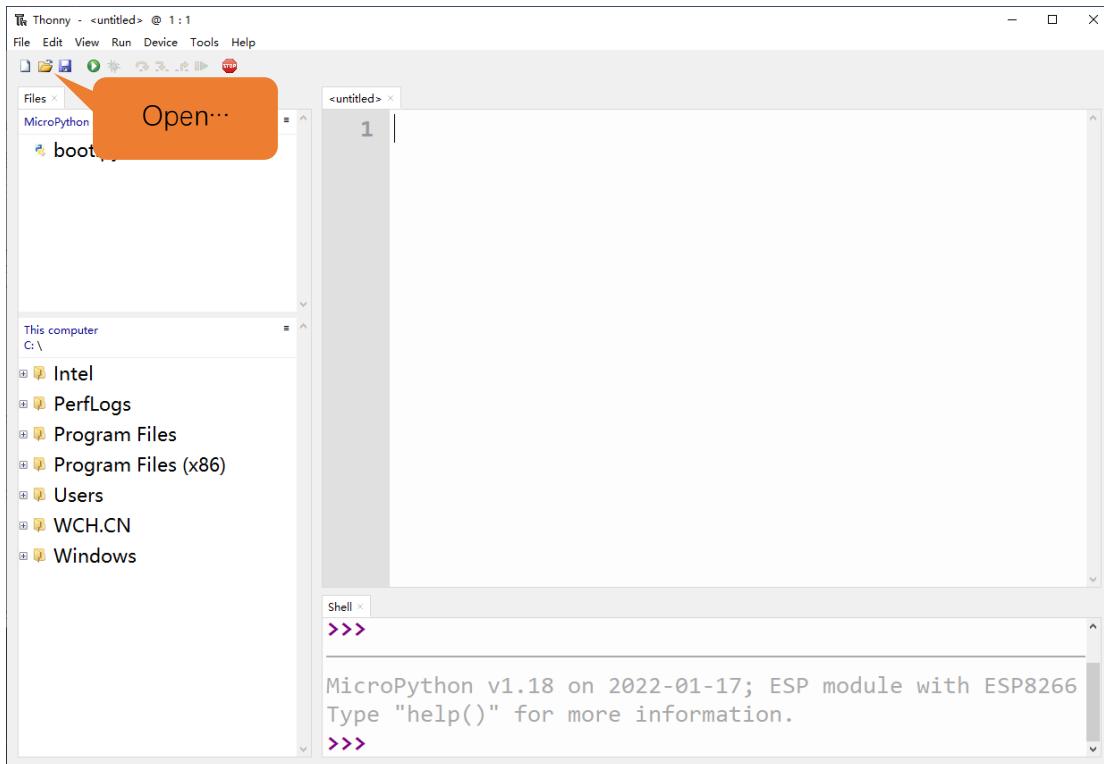
Enter “print('hello world')” in “Shell” and press Enter.



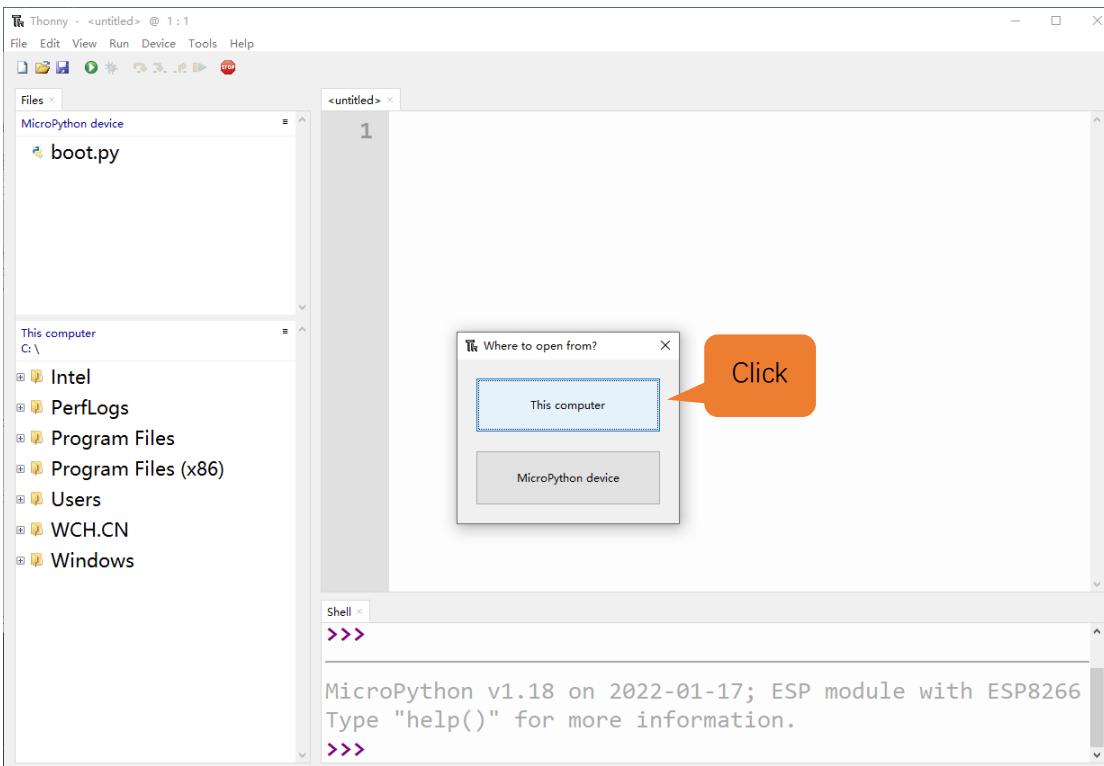
Running Online

ESP8266 needs to be connected to a computer when it is run online. Users can use Thonny to write and debug programs.

1. Open Thonny and click “Open…”.

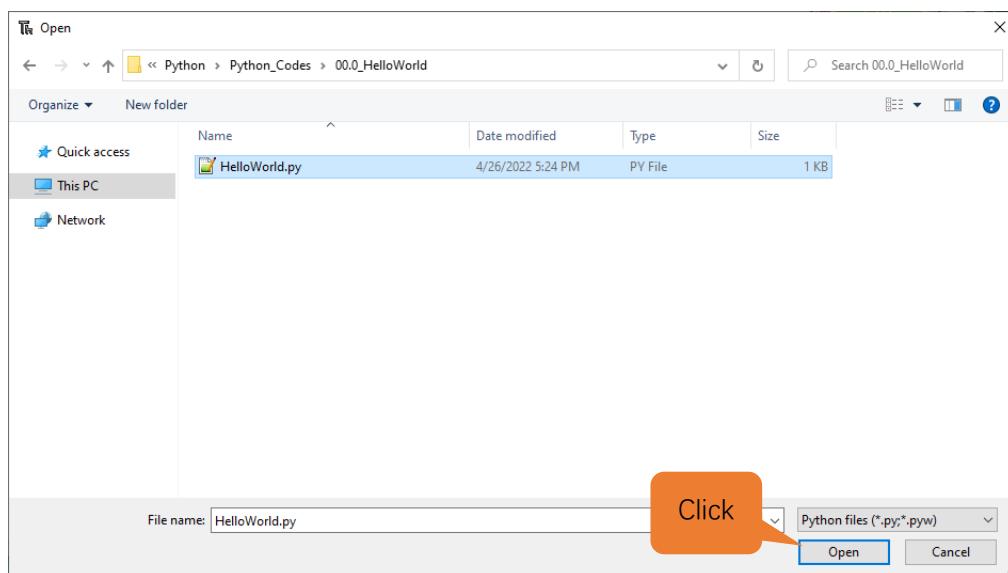


2. On the newly pop-up window, click “This computer”.

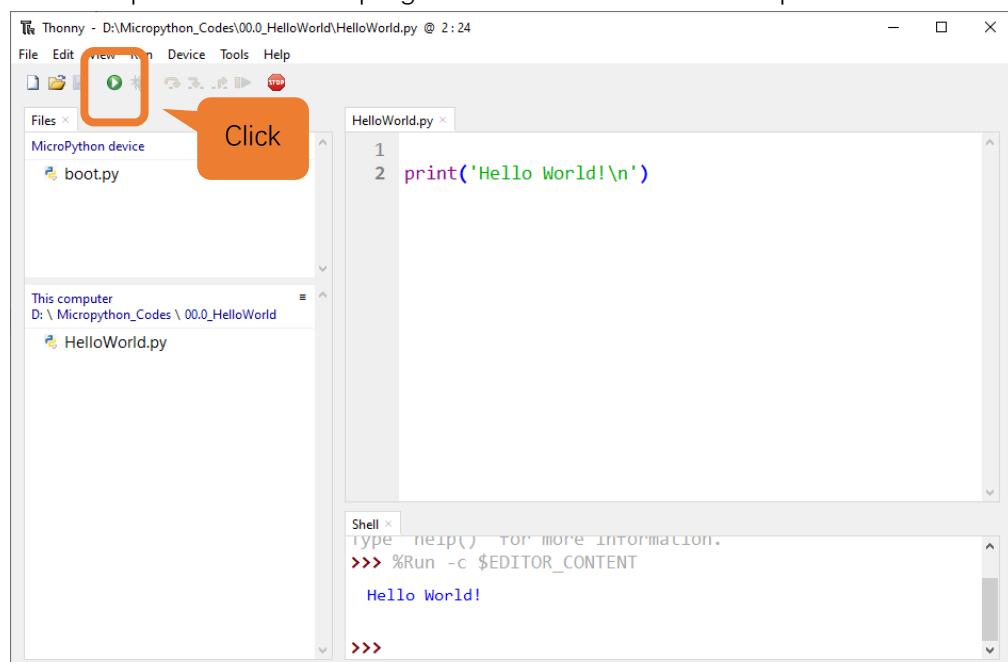


Any concerns? ✉ support@freenove.com

In the new dialog box, select “**HelloWorld.py**” in “**Freenove_ESP8266_Board/Python/Python_Codes/00.0_HelloWorld**” folder.



Click “Run current script” to execute the program and “Hello World” will be printed in “Shell”.



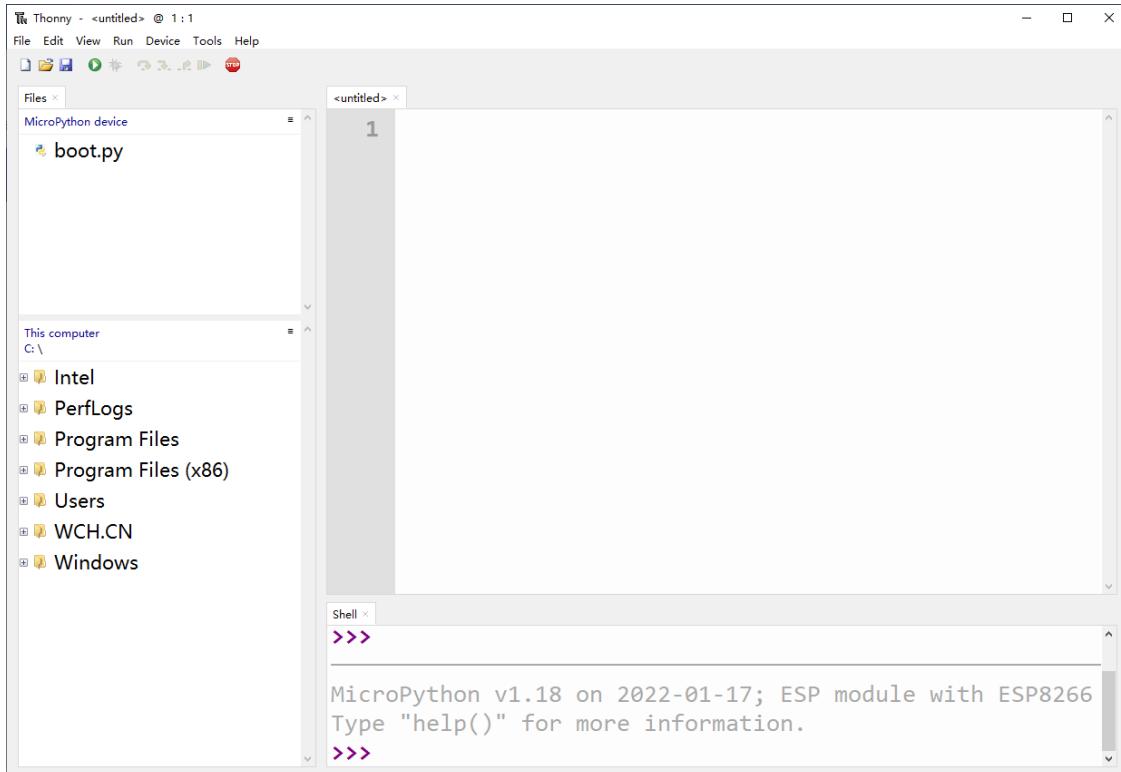
Note: When running online, if you press the reset key of ESP8266, user's code will not be executed again. If you wish to run the code automatically after resetting the code, please refer to the following [Running Offline](#).



Running Offline (Important)

After ESP8266 is reset, it runs the file boot.py in root directory first and then runs file main.py, and finally, it enters "Shell". Therefore, to make ESP8266 execute user's programs after resetting, we need to add a guiding program in boot.py to execute user's code.

1. Move the program folder "**Freenove_ESP8266_Board/Python/Python_Codes**" to disk(D) in advance with the path of "**D:/Micropython_Codes**". Open "Thonny".



2. Expand "00.1_Boot" in the "Micropython_Codes" in the directory of disk(D), and double-click boot.py, which is provided by us to enable programs in "MicroPython device" to run offline.

The screenshot shows the Thonny IDE interface. The top menu bar includes File, Edit, View, Run, Tools, and Help. Below the menu is a toolbar with icons for file operations like Open, Save, and Run. The main window has three tabs: Files, boot.py, and Shell.

- Files Tab:** Shows a file tree with "This computer" expanded, showing "D:\ Micropython_Codes\00.1_Boot" and "boot.py".
- boot.py Tab:** Displays the following Python code:

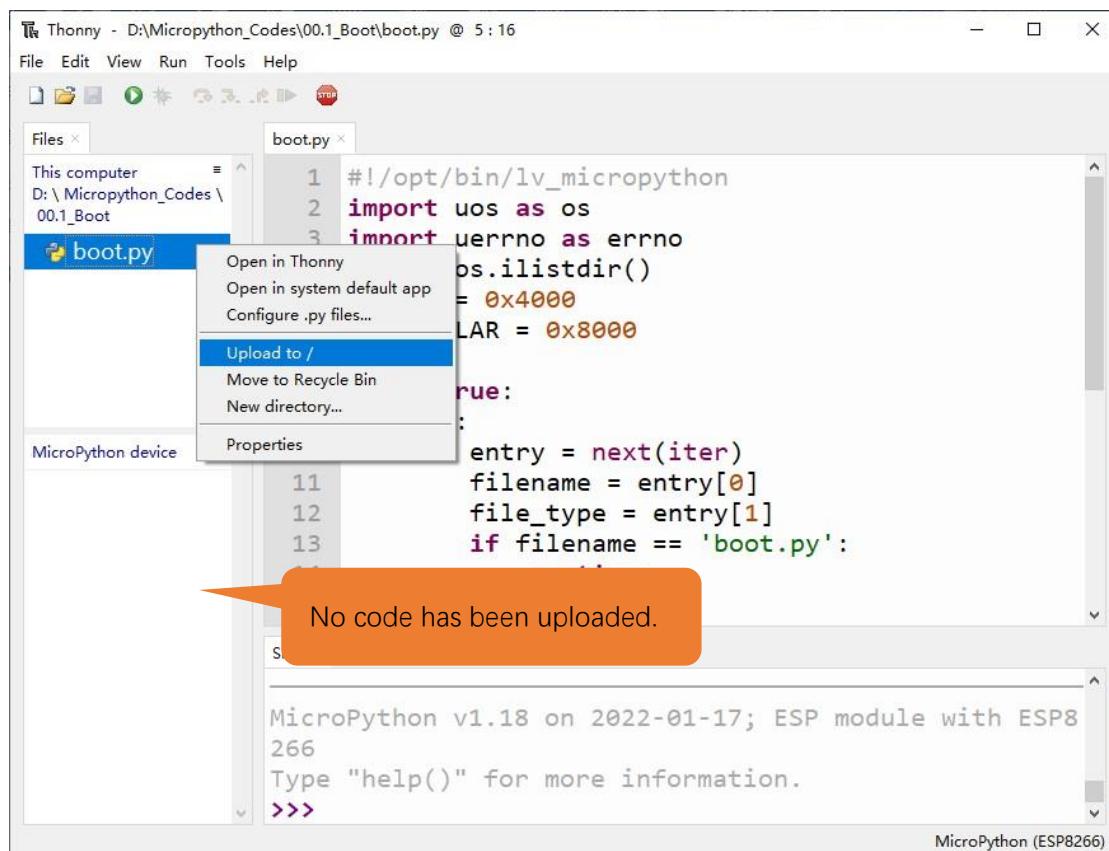

```

1 #!/opt/bin/lv_micropython
2 import uos as os
3 import uerrno as errno
4 iter = os.ilistdir()
5 IS_DIR = 0x4000
6 IS_REGULAR = 0x8000
7
8 while True:
9     try:
10         entry = next(iter)
11         filename = entry[0]
12         file_type = entry[1]
13         if filename == 'boot.py':
14             continue
15         else:
      
```
- Shell Tab:** Displays the MicroPython shell output:

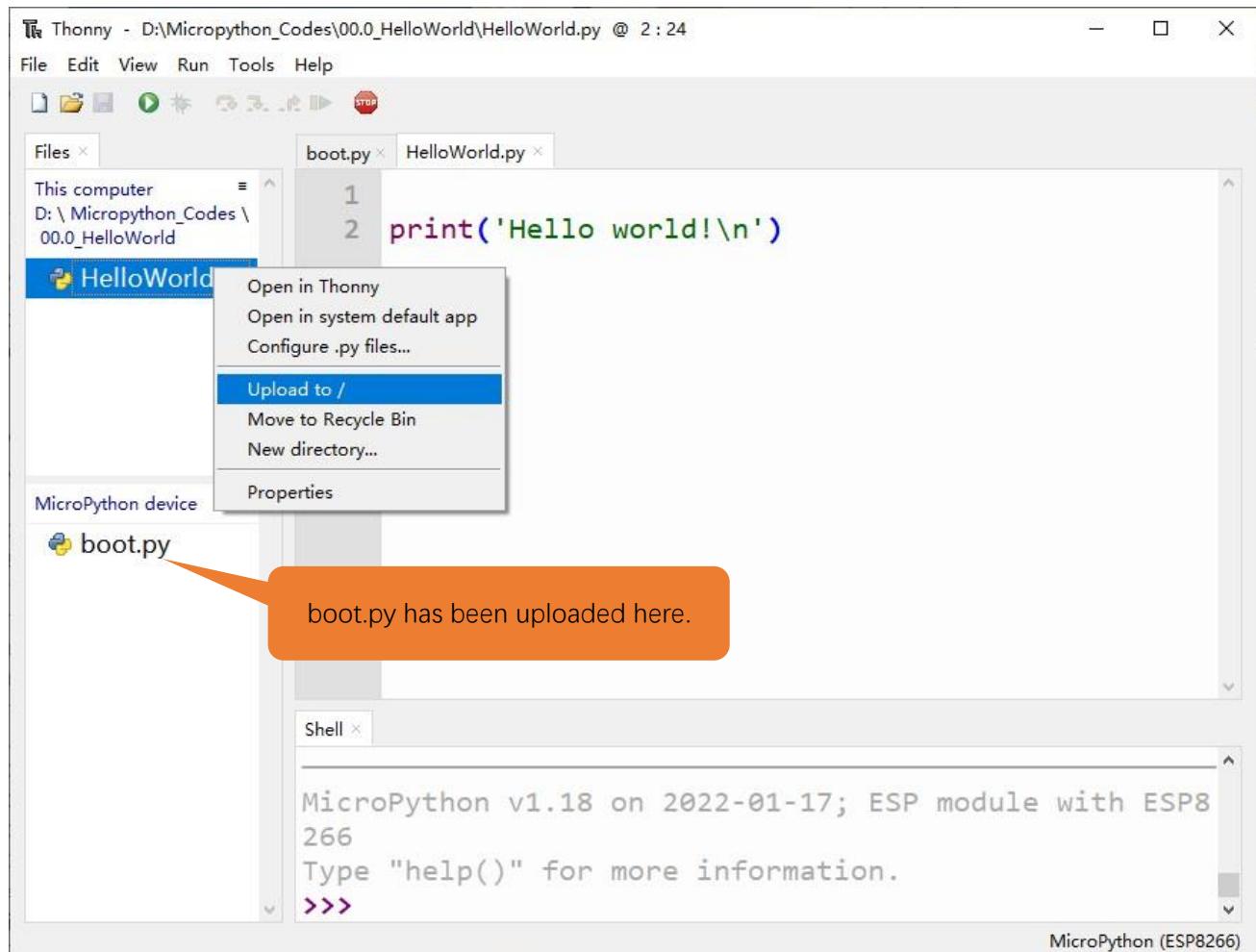

```

MicroPython v1.18 on 2022-01-17; ESP module with ESP8266
66
Type "help()" for more information.
>>>
      
```

If you want your written programs to run offline, you need to upload boot.py we provided and all your codes to “MicroPython device” and press ESP8266’s reset key. Here we use programs 00.0 and 00.1 as examples. Select “boot.py”, right-click to select “Upload to /”.



Similarly, upload “HelloWorld.py” to “MicroPython device”.



3. Press the reset key and in the box of the illustration below, you can see the code is executed.

The screenshot shows the Thonny IDE interface. In the top menu bar, it says "Thonny - D:\Micropython_Codes\00.0_HelloWorld\HelloWorld.py @ 2 : 24". The left sidebar shows a file tree with "This computer" and "D:\Micropython_Codes\00.0_HelloWorld" containing files "boot.py" and "HelloWorld.py". The main area has tabs for "Files" and "HelloWorld.py". The "HelloWorld.py" tab shows the code:

```

1 print('Hello world!\n')
2

```

The "Shell" tab shows the output of the program execution:

```

=====
HelloWorld.py
=====
Hello world!

```

Below the shell, the status message reads: "MicroPython v1.18 on 2022-01-17; ESP module with ESP8266 66 Type "help()" for more information." The bottom right corner of the shell window is labeled "MicroPython (ESP8266)".

When you press the Reset key to run the offline code, the program will continue to execute while the ESP8266 is powered on.

The screenshot shows the Thonny IDE interface. In the top menu bar, it says "Thonny - D:\Micropython_Codes\01.1_Blink\Blink.py @ 14 : 1". The left sidebar shows a file tree with "This computer" and "D:\Micropython_Codes\01.1_Blink" containing files "Blink.py" and "boot.py". The main area has tabs for "Files" and "Blink.py". The "Blink.py" tab shows the code:

```

3 led=Pin(2,Pin.OUT) #create LED object from pin4,Set Pin4 to out
4
5 try:
6     while True:
7         led.value(1)          #Set led turn on
8         sleep_ms(1000)
9         led.value(0)          #Set led turn off
10        sleep_ms(1000)
11    except:
12        pass

```

The "Shell" tab shows the output of the program execution:

```

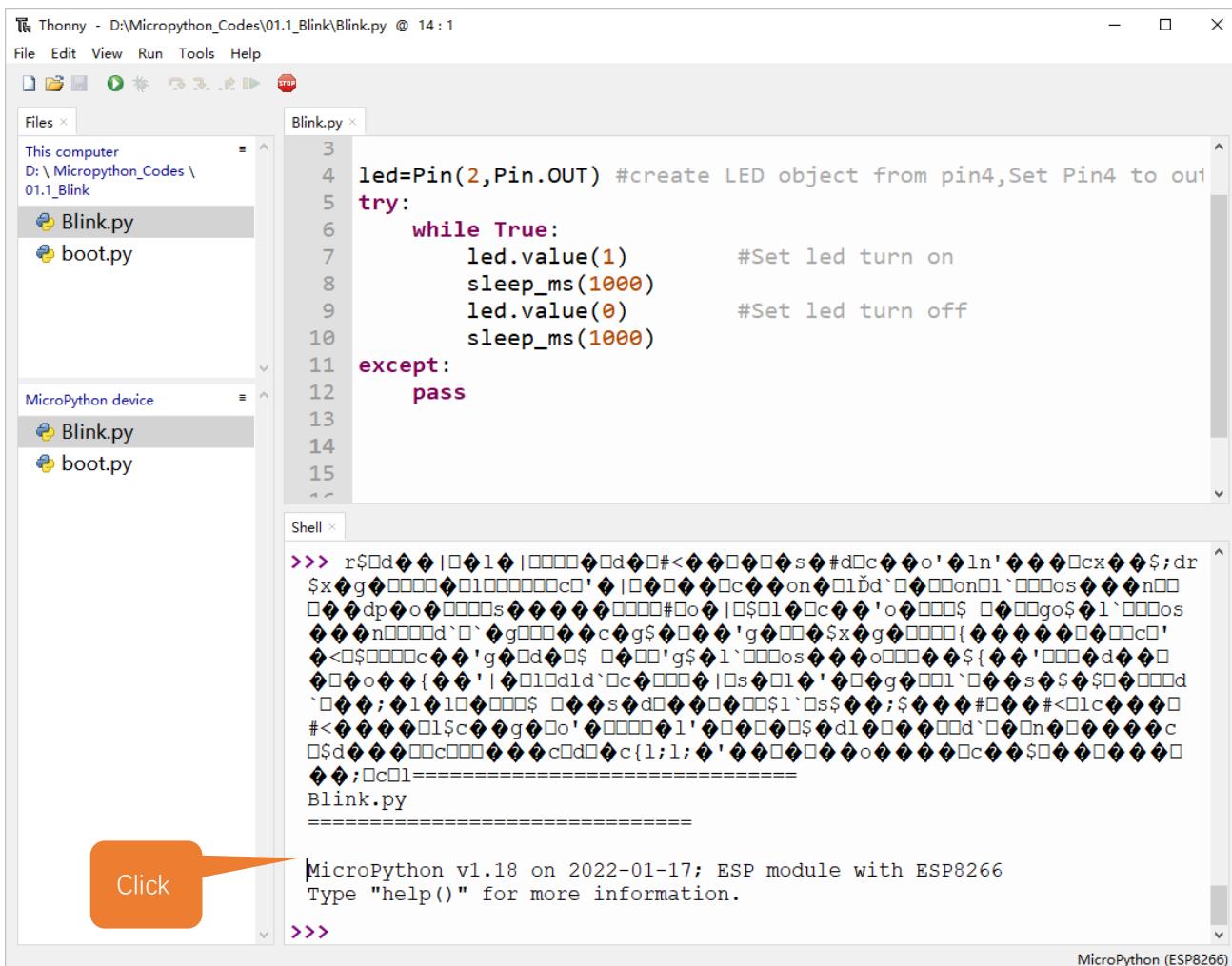
=====
MicroPython v1.18 on 2022-01-17; ESP module with ESP8266
Type "help()" for more information.

```

The output continues with a series of binary characters and control codes, indicating the LED is alternately turning on and off. The bottom right corner of the shell window is labeled "MicroPython (ESP8266)".

When you run offline code, you can exit the running program by pressing "CTRL" and "C" at the same time.

Before pressing the keyboard, click "Shell" with the mouse, and then press the keyboard key.



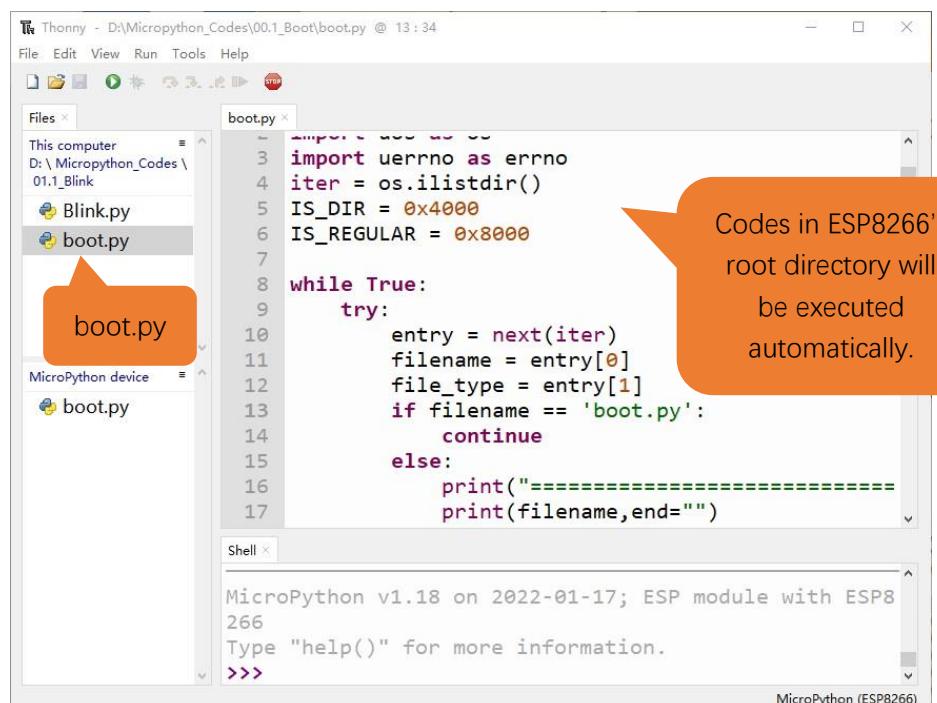
```

Thonny - D:\Micropython_Codes\01.1_Blink\Blink.py @ 14:1
File Edit View Run Tools Help
Files x
This computer
D:\Micropython_Codes\01.1_Blink
Blink.py
boot.py
MicroPython device
Blink.py
boot.py
Blink.py x
3
4 led=Pin(2,Pin.OUT) #create LED object from pin4,Set Pin4 to out
5 try:
6     while True:
7         led.value(1)      #Set led turn on
8         sleep_ms(1000)
9         led.value(0)      #Set led turn off
10        sleep_ms(1000)
11 except:
12     pass
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
310
320
330
340
350
360
370
380
390
400
410
420
430
440
450
460
470
480
490
500
510
520
530
540
550
560
570
580
590
600
610
620
630
640
650
660
670
680
690
700
710
720
730
740
750
760
770
780
790
800
810
820
830
840
850
860
870
880
890
900
910
920
930
940
950
960
970
980
990
1000
1010
1020
1030
1040
1050
1060
1070
1080
1090
1100
1110
1120
1130
1140
1150
1160
1170
1180
1190
1200
1210
1220
1230
1240
1250
1260
1270
1280
1290
1300
1310
1320
1330
1340
1350
1360
1370
1380
1390
1400
1410
1420
1430
1440
1450
1460
1470
1480
1490
1500
1510
1520
1530
1540
1550
1560
1570
1580
1590
1600
1610
1620
1630
1640
1650
1660
1670
1680
1690
1700
1710
1720
1730
1740
1750
1760
1770
1780
1790
1800
1810
1820
1830
1840
1850
1860
1870
1880
1890
1900
1910
1920
1930
1940
1950
1960
1970
1980
1990
2000
2010
2020
2030
2040
2050
2060
2070
2080
2090
2100
2110
2120
2130
2140
2150
2160
2170
2180
2190
2200
2210
2220
2230
2240
2250
2260
2270
2280
2290
2300
2310
2320
2330
2340
2350
2360
2370
2380
2390
2400
2410
2420
2430
2440
2450
2460
2470
2480
2490
2500
2510
2520
2530
2540
2550
2560
2570
2580
2590
2600
2610
2620
2630
2640
2650
2660
2670
2680
2690
2700
2710
2720
2730
2740
2750
2760
2770
2780
2790
2800
2810
2820
2830
2840
2850
2860
2870
2880
2890
2890
2900
2910
2920
2930
2940
2950
2960
2970
2980
2990
3000
3100
3200
3300
3400
3500
3600
3700
3800
3900
4000
4100
4200
4300
4400
4500
4600
4700
4800
4900
5000
5100
5200
5300
5400
5500
5600
5700
5800
5900
6000
6100
6200
6300
6400
6500
6600
6700
6800
6900
7000
7100
7200
7300
7400
7500
7600
7700
7800
7900
8000
8100
8200
8300
8400
8500
8600
8700
8800
8900
8900
9000
9100
9200
9300
9400
9500
9600
9700
9800
9900
10000
10100
10200
10300
10400
10500
10600
10700
10800
10900
11000
11100
11200
11300
11400
11500
11600
11700
11800
11900
12000
12100
12200
12300
12400
12500
12600
12700
12800
12900
13000
13100
13200
13300
13400
13500
13600
13700
13800
13900
14000
14100
14200
14300
14400
14500
14600
14700
14800
14900
15000
15100
15200
15300
15400
15500
15600
15700
15800
15900
16000
16100
16200
16300
16400
16500
16600
16700
16800
16900
17000
17100
17200
17300
17400
17500
17600
17700
17800
17900
18000
18100
18200
18300
18400
18500
18600
18700
18800
18900
19000
19100
19200
19300
19400
19500
19600
19700
19800
19900
20000
20100
20200
20300
20400
20500
20600
20700
20800
20900
21000
21100
21200
21300
21400
21500
21600
21700
21800
21900
22000
22100
22200
22300
22400
22500
22600
22700
22800
22900
23000
23100
23200
23300
23400
23500
23600
23700
23800
23900
24000
24100
24200
24300
24400
24500
24600
24700
24800
24900
25000
25100
25200
25300
25400
25500
25600
25700
25800
25900
26000
26100
26200
26300
26400
26500
26600
26700
26800
26900
27000
27100
27200
27300
27400
27500
27600
27700
27800
27900
28000
28100
28200
28300
28400
28500
28600
28700
28800
28900
28900
29000
29100
29200
29300
29400
29500
29600
29700
29800
29900
30000
31000
32000
33000
34000
35000
36000
37000
38000
39000
40000
41000
42000
43000
44000
45000
46000
47000
48000
49000
50000
51000
52000
53000
54000
55000
56000
57000
58000
59000
60000
61000
62000
63000
64000
65000
66000
67000
68000
69000
70000
71000
72000
73000
74000
75000
76000
77000
78000
79000
80000
81000
82000
83000
84000
85000
86000
87000
88000
89000
89000
90000
91000
92000
93000
94000
95000
96000
97000
98000
99000
100000
101000
102000
103000
104000
105000
106000
107000
108000
109000
110000
111000
112000
113000
114000
115000
116000
117000
118000
119000
120000
121000
122000
123000
124000
125000
126000
127000
128000
129000
130000
131000
132000
133000
134000
135000
136000
137000
138000
139000
140000
141000
142000
143000
144000
145000
146000
147000
148000
149000
150000
151000
152000
153000
154000
155000
156000
157000
158000
159000
160000
161000
162000
163000
164000
165000
166000
167000
168000
169000
170000
171000
172000
173000
174000
175000
176000
177000
178000
179000
180000
181000
182000
183000
184000
185000
186000
187000
188000
189000
189000
190000
191000
192000
193000
194000
195000
196000
197000
198000
199000
200000
201000
202000
203000
204000
205000
206000
207000
208000
209000
210000
211000
212000
213000
214000
215000
216000
217000
218000
219000
220000
221000
222000
223000
224000
225000
226000
227000
228000
229000
230000
231000
232000
233000
234000
235000
236000
237000
238000
239000
240000
241000
242000
243000
244000
245000
246000
247000
248000
249000
250000
251000
252000
253000
254000
255000
256000
257000
258000
259000
260000
261000
262000
263000
264000
265000
266000
267000
268000
269000
270000
271000
272000
273000
274000
275000
276000
277000
278000
279000
280000
281000
282000
283000
284000
285000
286000
287000
288000
289000
289000
290000
291000
292000
293000
294000
295000
296000
297000
298000
299000
300000
310000
320000
330000
340000
350000
360000
370000
380000
390000
400000
410000
420000
430000
440000
450000
460000
470000
480000
490000
500000
510000
520000
530000
540000
550000
560000
570000
580000
590000
600000
610000
620000
630000
640000
650000
660000
670000
680000
690000
700000
710000
720000
730000
740000
750000
760000
770000
780000
790000
800000
810000
820000
830000
840000
850000
860000
870000
880000
890000
890000
900000
910000
920000
930000
940000
950000
960000
970000
980000
990000
1000000
1010000
1020000
1030000
1040000
1050000
1060000
1070000
1080000
1090000
1100000
1110000
1120000
1130000
1140000
1150000
1160000
1170000
1180000
1190000
1200000
1210000
1220000
1230000
1240000
1250000
1260000
1270000
1280000
1290000
1300000
1310000
1320000
1330000
1340000
1350000
1360000
1370000
1380000
1390000
1400000
1410000
1420000
1430000
1440000
1450000
1460000
1470000
1480000
1490000
1500000
1510000
1520000
1530000
1540000
1550000
1560000
1570000
1580000
1590000
1600000
1610000
1620000
1630000
1640000
1650000
1660000
1670000
1680000
1690000
1700000
1710000
1720000
1730000
1740000
1750000
1760000
1770000
1780000
1790000
1800000
1810000
1820000
1830000
1840000
1850000
1860000
1870000
1880000
1890000
1890000
1900000
1910000
1920000
1930000
1940000
1950000
1960000
1970000
1980000
1990000
2000000
2010000
2020000
2030000
2040000
2050000
2060000
2070000
2080000
2090000
2100000
2110000
2120000
2130000
2140000
2150000
2160000
2170000
2180000
2190000
2200000
2210000
2220000
2230000
2240000
2250000
2260000
2270000
2280000
2290000
2300000
2310000
2320000
2330000
2340000
2350000
2360000
2370000
2380000
2390000
2400000
2410000
2420000
2430000
2440000
2450000
2460000
2470000
2480000
2490000
2500000
2510000
2520000
2530000
2540000
2550000
2560000
2570000
2580000
2590000
2600000
2610000
2620000
2630000
2640000
2650000
2660000
2670000
2680000
2690000
2700000
2710000
2720000
2730000
2740000
2750000
2760000
2770000
2780000
2790000
2800000
2810000
2820000
2830000
2840000
2850000
2860000
2870000
2880000
2890000
2890000
2900000
2910000
2920000
2930000
2940000
2950000
2960000
2970000
2980000
2990000
3000000
3100000
3200000
3300000
3400000
3500000
3600000
3700000
3800000
3900000
4000000
4100000
4200000
4300000
4400000
4500000
4600000
4700000
4800000
4900000
5000000
5100000
5200000
5300000
5400000
5500000
5600000
5700000
5800000
5900000
6000000
6100000
6200000
6300000
6400000
6500000
6600000
6700000
6800000
6900000
7000000
7100000
7200000
7300000
7400000
7500000
7600000
7700000
7800000
7900000
8000000
8100000
8200000
8300000
8400000
8500000
8600000
8700000
8800000
8900000
8900000
9000000
9100000
9200000
9300000
9400000
9500000
9600000
9700000
9800000
9900000
10000000
10100000
10200000
10300000
10400000
10500000
10600000
10700000
10800000
10900000
11000000
11100000
11200000
11300000
11400000
11500000
11600000
11700000
11800000
11900000
12000000
12100000
12200000
12300000
12400000
12500000
12600000
12700000
12800000
12900000
13000000
13100000
13200000
13300000
13400000
13500000
13600000
13700000
13800000
13900000
14000000
14100000
14200000
14300000
14400000
14500000
14600000
14700000
14800000
14900000
15000000
15100000
15200000
15300000
15400000
15500000
15600000
15700000
15800000
15900000
16000000
16100000
16200000
16300000
16400000
16500000
16600000
16700000
16800000
16900000
17000000
17100000
17200000
17300000
17400000
17500000
17600000
17700000
17800000
17900000
18000000
18100000
18200000
18300000
18400000
18500000
18600000
18700000
18800000
18900000
18900000
19000000
19100000
19200000
19300000
19400000
19500000
19600000
19700000
19800000
19900000
20000000
20100000
20200000
20300000
20400000
20500000
20600000
20700000
20800000
20900000
21000000
21100000
21200000
21300000
21400000
21500000
21600000
21700000
21800000
21900000
22000000
22100000
22200000
22300000
22400000
22500000
22600000
22700000
22800000
22900000
23000000
23100000
23200000
23300000
23400000
23500000
23600000
23700000
23800000
23900000
24000000
24100000
24200000
24300000
24400000
24500000
24600000
24700000
24800000
24900000
25000000
25100000
25200000
25300000
25400000
25500000
25600000
25700000
25800000
25900000
26000000
26100000
26200000
26300000
26400000
26500000
26600000
26700000
26800000
26900000
27000000
27100000
27200000
27300000
27400000
27500000
27600000
27700000
27800000
27900000
28000000
28100000
28200000
28300000
28400000
28500000
28600000
28700000
28800000
28900000
28900000
29000000
29100000
29200000
29300000
29400000
29500000
29600000
29700000
29800000
29900000
30000000
31000000
32000000
33000000
34000000
35000000
36000000
37000000
38000000
39000000
40000000
41000000
42000000
43000000
44000000
45000000
46000000
47000000
48000000
49000000
50000000
51000000
52000000
53000000
54000000
55000000
56000000
57000000
58000000
59000000
60000000
61000000
62000000
63000000
64000000
65000000
66000000
67000000
68000000
69000000
70000000
71000000
72000000
73000000
74000000
75000000
76000000
77000000
78000000
79000000
80000000
81000000
82000000
83000000
8
```

0.6 Thonny Common Operation

Uploading Code to ESP8266

For convenience, we take the operation on “boot.py” as an example here. We have added “boot.py” to every code directory. Each time when ESP8266 restarts, if there is a “boot.py” in the root directory, it will execute this code first.

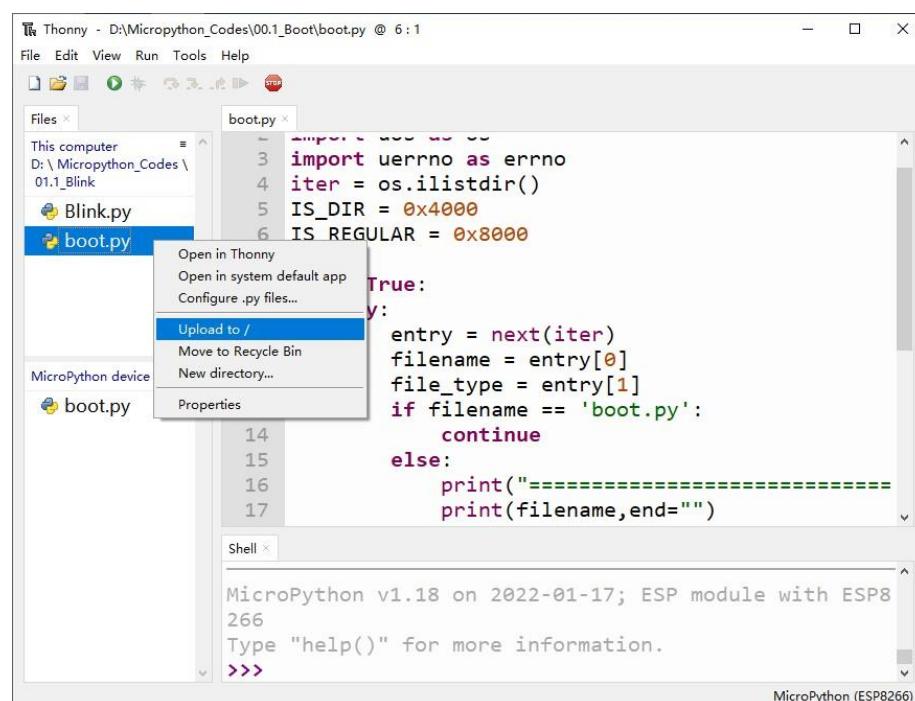


```

    Thonny - D:\Micropython_Codes\00.1_Boot\boot.py @ 13 : 34
    File Edit View Run Tools Help
    Files x boot.py x
    This computer D:\ Micropython_Codes \ 01.1_Blink
    Blink.py boot.py
    MicroPython device x boot.py
    boot.py
    1 import uerrno as errno
    2 iter = os.ilistdir()
    3 IS_DIR = 0x4000
    4 IS_REGULAR = 0x8000
    5
    6 while True:
    7     try:
    8         entry = next(iter)
    9         filename = entry[0]
    10        file_type = entry[1]
    11        if filename == 'boot.py':
    12            continue
    13        else:
    14            print("====")
    15            print(filename, end="")
    16
    17
    Shell x
    MicroPython v1.18 on 2022-01-17; ESP module with ESP8
    266
    Type "help()" for more information.
    >>>
  
```

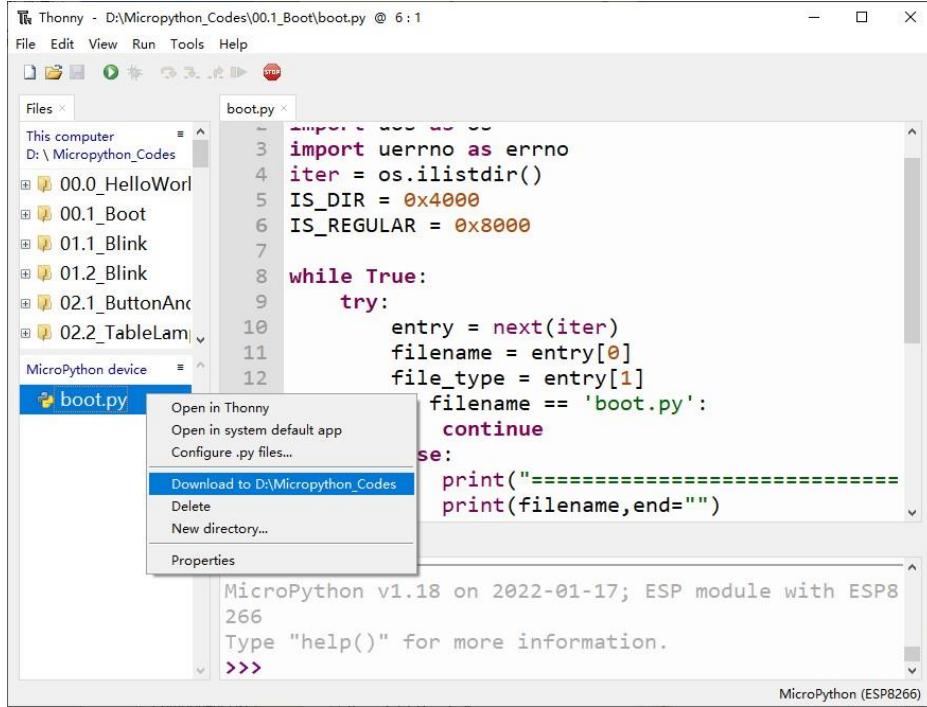
Codes in ESP8266's root directory will be executed automatically.

Select “Blink.py” in “01.1_Blink”, right-click your mouse and select “Upload to /” to upload code to ESP8266's root directory.



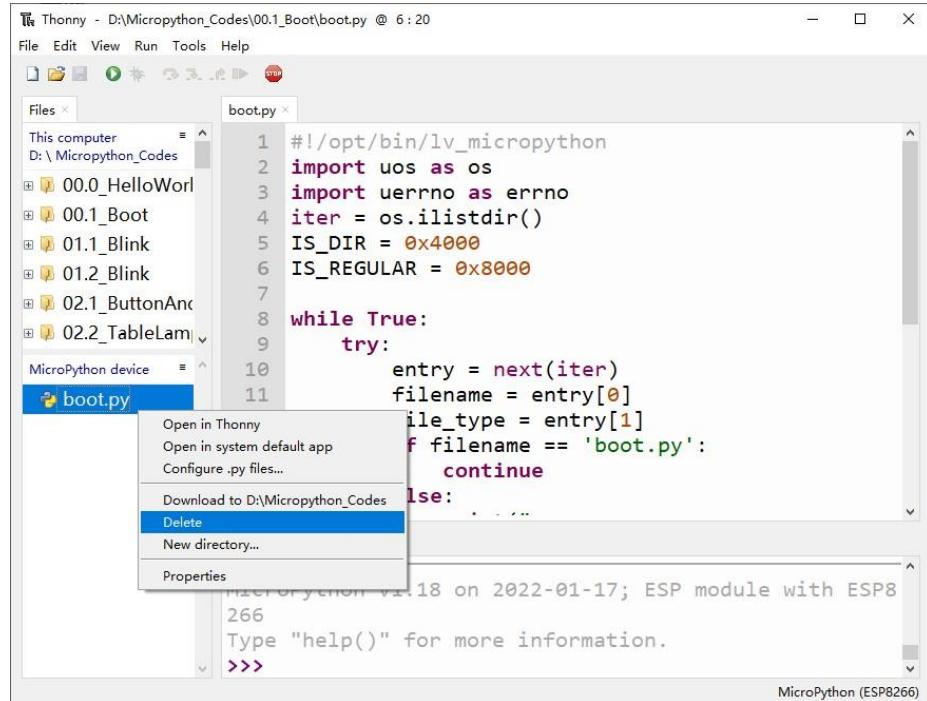
Downloading Code to Computer

Select “boot.py” in “MicroPython device”, right-click to select “Download to ...” to download the code to your computer.



Deleting Files from ESP8266's Root Directory

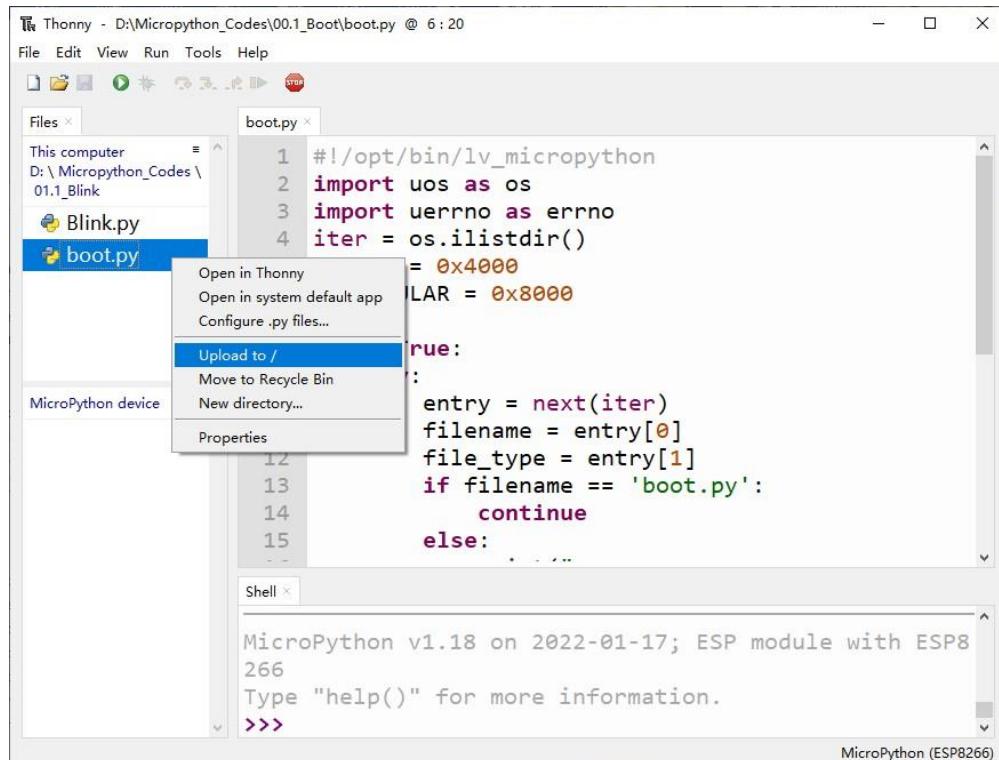
Select “boot.py” in “MicroPython device”, right-click it and select “Delete” to delete “boot.py” from ESP8266’s root directory.



Any concerns? ✉ support@freenove.com

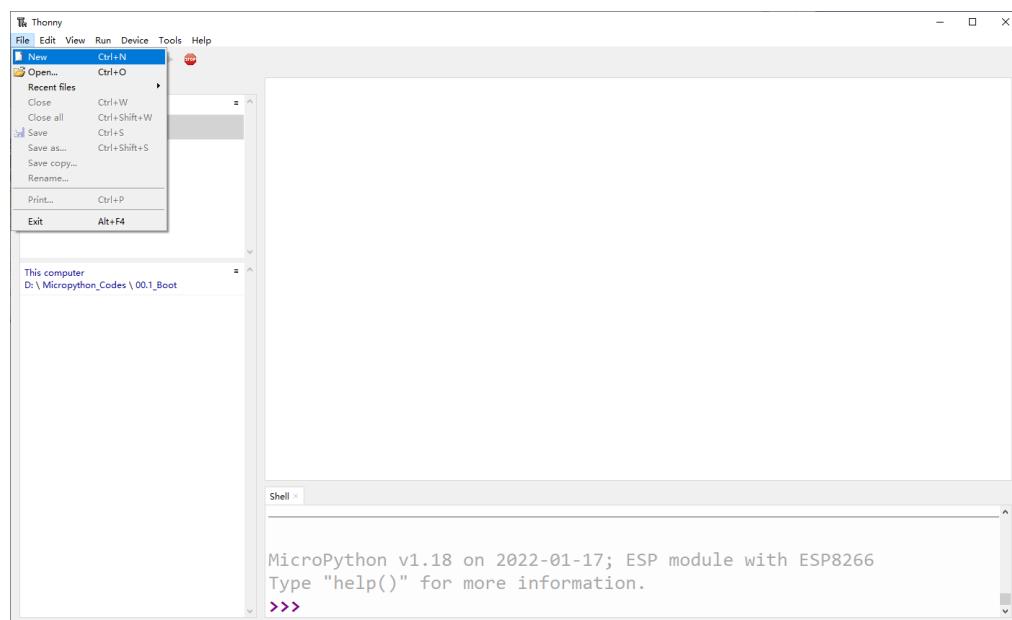
Deleting Files from your Computer Directory

Select “boot.py” in “00.1_Boot”, right-click it and select “Move to Recycle Bin” to delete it from “00.1_Boot”.

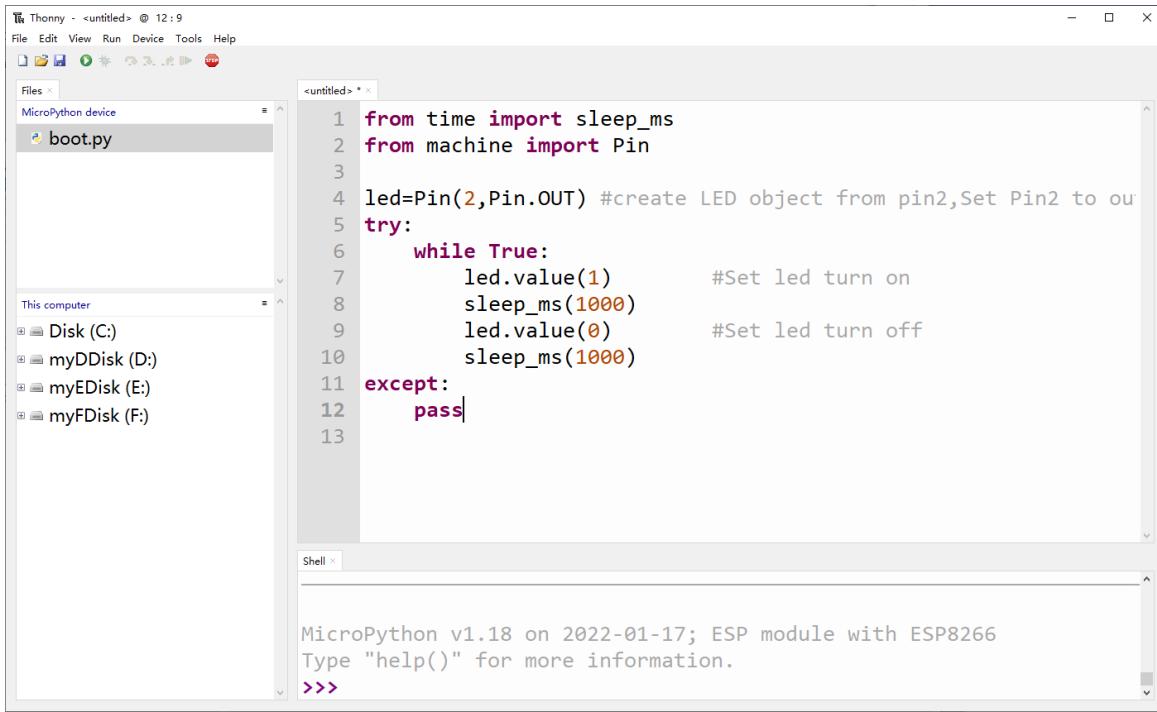


Creating and Saving the code

Click “File”→“New” to create and write codes.



Enter codes in the newly opened file. Here we use codes of “01.1_Blink.py” as an example.



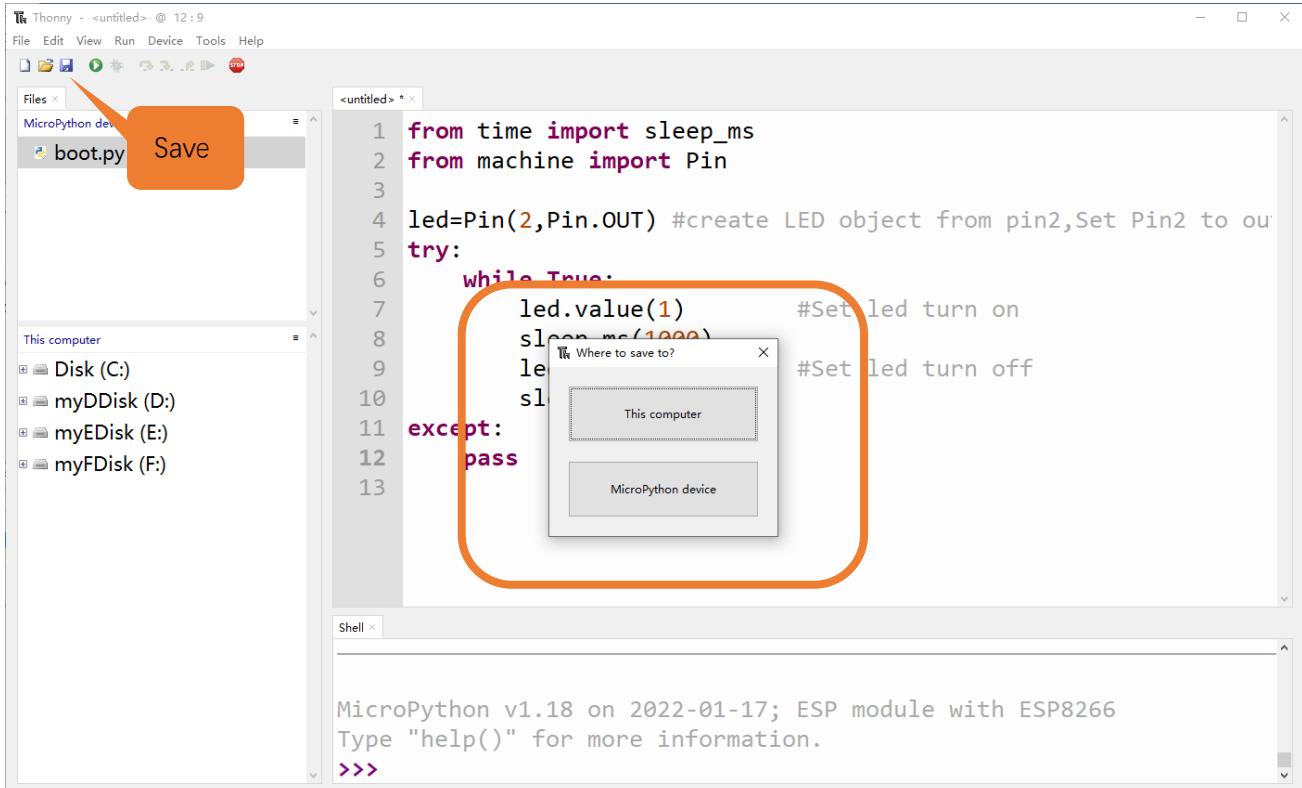
```

from time import sleep_ms
from machine import Pin

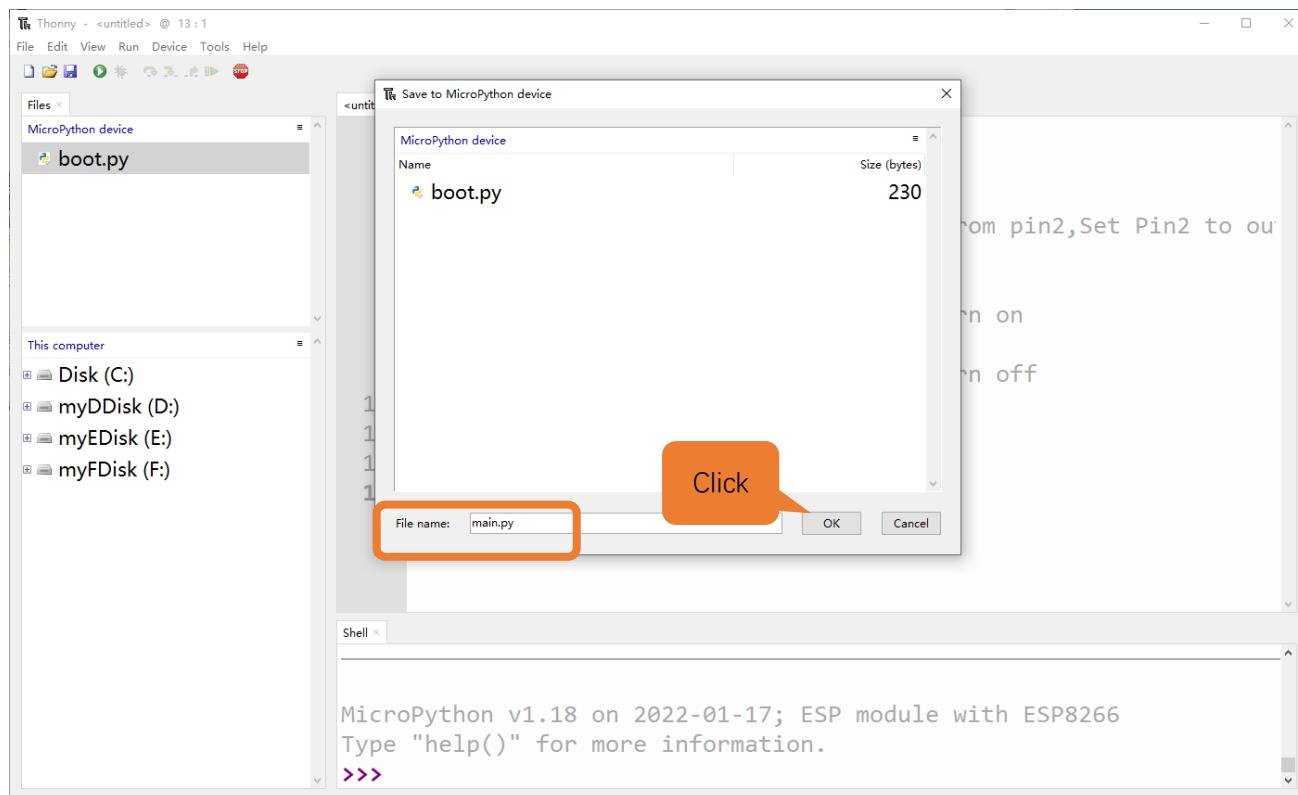
led=Pin(2,Pin.OUT) #create LED object from pin2,Set Pin2 to output
try:
    while True:
        led.value(1)          #Set led turn on
        sleep_ms(1000)
        led.value(0)          #Set led turn off
        sleep_ms(1000)
except:
    pass

```

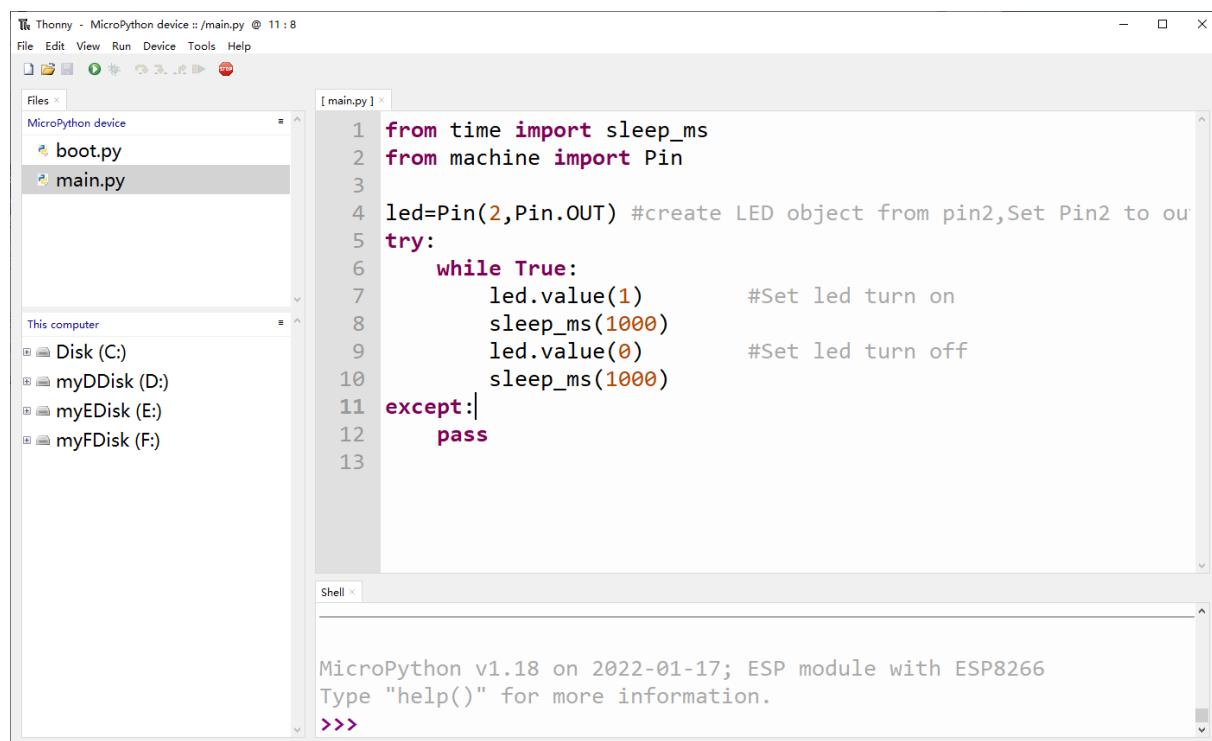
Click “Save” on the menu bar. You can save the codes either to your computer or to ESP8266.



Select “MicroPython device”, enter “main.py” in the newly pop-up window and click “OK”.



You can see that codes have been uploaded to ESP8266.



1, Stop/Restart backend

2, Run current script

```

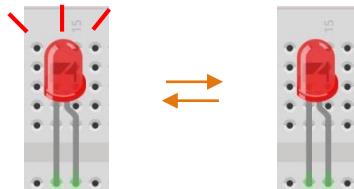
1 from time import sleep_ms
2 from machine import Pin
3
4 led = Pin(2, Pin.OUT) #create LED object from pin2,Set Pin2 to output
5
6 while True:
7     led.value(1)          #Set led turn on
8     sleep_ms(1000)
9     led.value(0)          #Set led turn off
10    sleep_ms(1000)
11 except:
12     pass
13

```

This indicates that the connection is successful.

MicroPython v1.18 on 2022-01-17; ESP module with ESP8266
Type "help()" for more information.
=>

Disconnect and reconnect USB cable, and you can see that LED is ON for one second and then OFF for one second, which repeats in an endless loop.



Chapter 1 LED (Important)

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

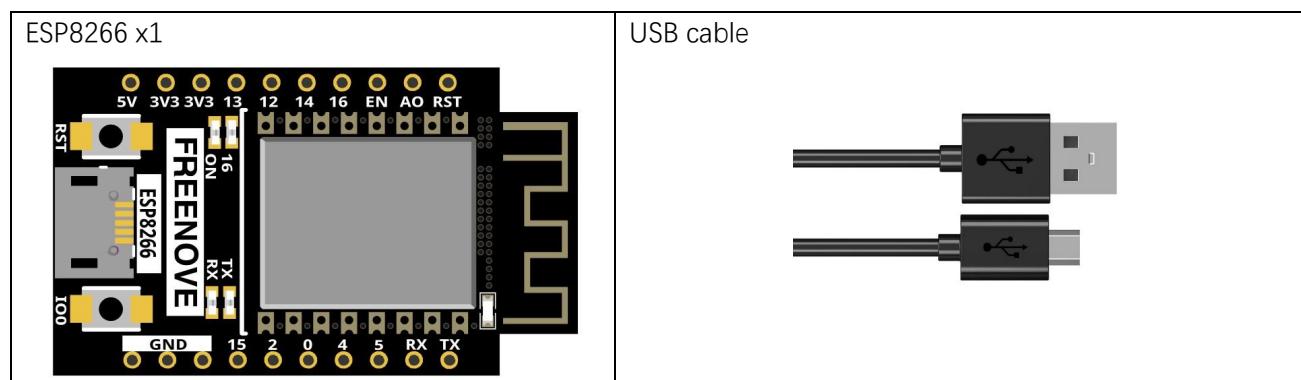
In this project, we will use ESP8266 to control blinking a common LED.

If you have not yet installed Thonny, click [here](#).

If you have not yet downloaded MicroPython Firmware, click [here](#).

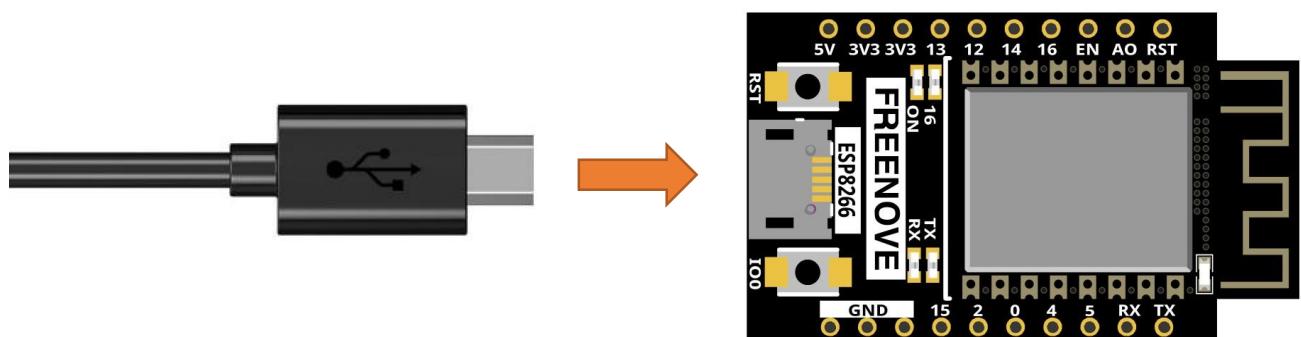
If you have not yet loaded MicroPython Firmware, click [here](#).

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.



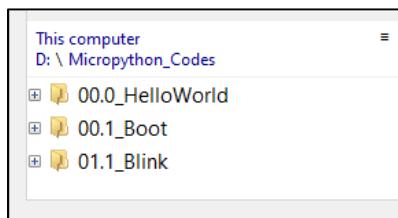
In the following projects, we only use USB cable to power ESP8266 development board by default.

Code

Codes used in this tutorial are saved in “**Freenove_ESP8266_Board/Python/Python_Codes**”. You can move the codes to any location. For example, we save the codes in Disk(D) with the path of “**D:/Micropython_Codes**”.

01.1_Blink

Open “Thonny”, click “This computer”→“D:”→“Micropython_Codes”.



Expand folder "01.1_Blink" and double click "Blink.py" to open it. As shown in the illustration below.



Make sure ESP8266 is properly connected to your computer. Click “Stop/Restart backend” or press the reset button, and then wait to see what interface will show up.

```

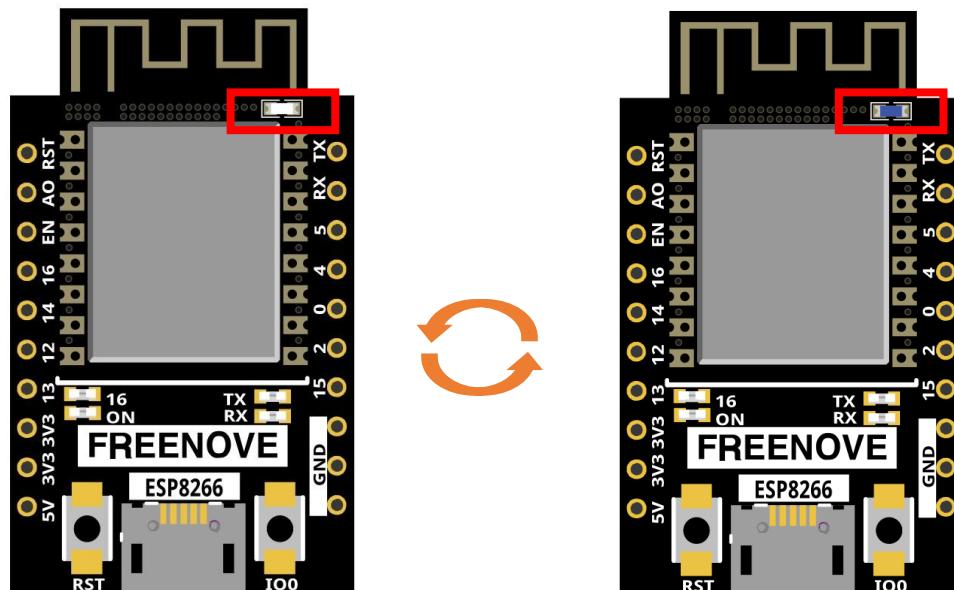
1. Stop/Restart backend
2. Run current script

This indicates
that the
connection is
successful.

File Edit View Run Device Tools Help
D:\Micropython_Codes\01_1_Blink\Blink.py @ 2 : 8
MicroPython device
boot.py
D:\Micropython_Codes\01_1_Blink
Blink.py
1 from time import sleep_ms
2 from machine import Pin
3
4 led = Pin(2, Pin.OUT)
5
6 def main():
7     led.value(1) #Set led turn on
8     sleep_ms(1000)
9     led.value(0) #Set led turn off
10    sleep_ms(1000)
11
12 except:
13     pass
14
15
16
Shell <
MicroPython v1.18 on 2022-01-17; ESP module with ESP8266
Type "help()" for more information.
>>> %Run -c $EDITOR_CONTENT
>>> %Run -c $EDITOR_CONTENT

```

Click “Run current script” shown in the box above, the code starts to be executed and the LED in the circuit starts to blink.



Note:

This is the code [running online](#). If you disconnect USB cable and repower ESP8266 or press its reset key, LED stops blinking and the following messages will be displayed in Thonny.

```

Type "help()" for more information.

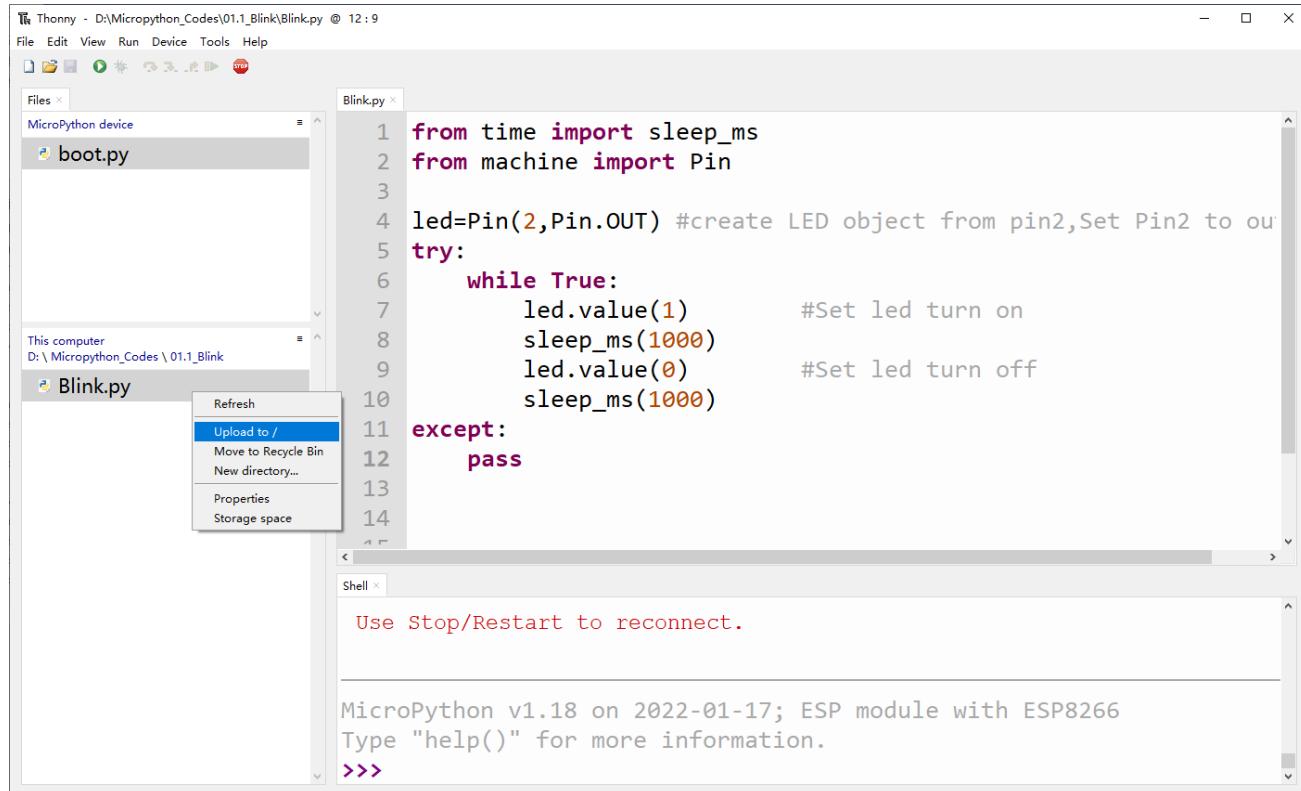
>>>
Connection lost (GetOverlappedResult failed (PermissionError(13, 'Access is denied.', None, 5)))

Use Stop/Restart to reconnect.

```

Uploading code to ESP8266

As shown in the following illustration, right-click the file Blink.py and select “Upload to /” to upload code to ESP8266.



Upload boot.py in the same way.

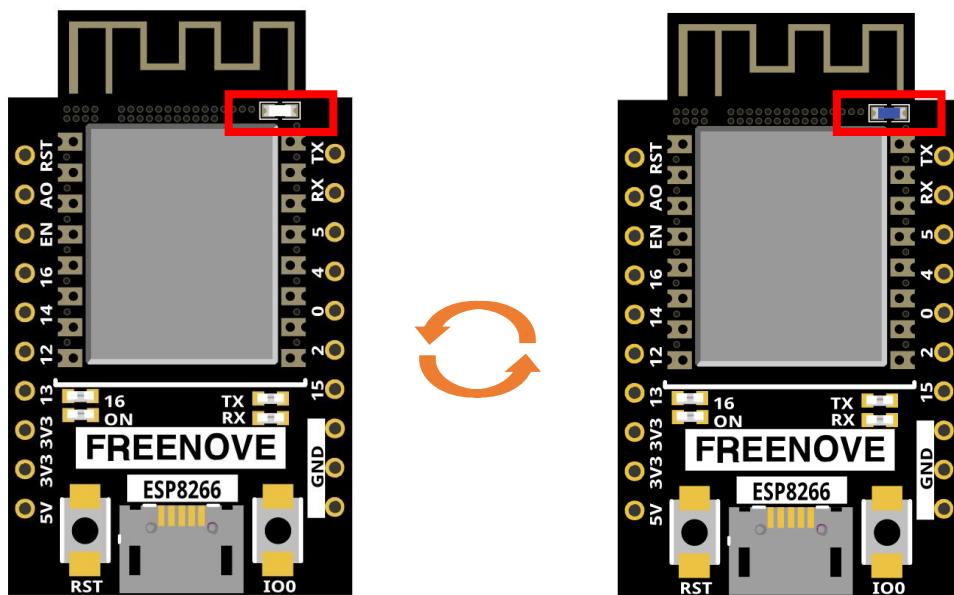
```

Thonny - D:\Micropython_Codes\01.1_Blink\Blink.py @ 4 : 12
File Edit View Run Device Tools Help
File Open Save Run Stop
MicroPython device
Blink.py
boot.py
This computer
D:\ Micropython_Codes \ 01.1_Blink
Blink.py
Refresh
Upload to /
Move to Recycle Bin
New directory...
Properties
Storage space
Blink.py
7
8
9
10
11
12
13
14
15
except:
    pass
led.value(1)
sleep_ms(1000)
led.value(0)
sleep_ms(1000)
#Set led turn on
#Set led turn off
Use Stop/Restart to reconnect.

MicroPython v1.18 on 2022-01-17; ESP module with ESP8266
Type "help()" for more information.
>>>

```

Press the reset key of ESP8266 and you can see LED is ON for one second and then OFF for one second, which repeats in an endless loop.



Note:

Codes here is run offline. If you want to stop running offline and enter Shell, just click "Stop" in Thonny.



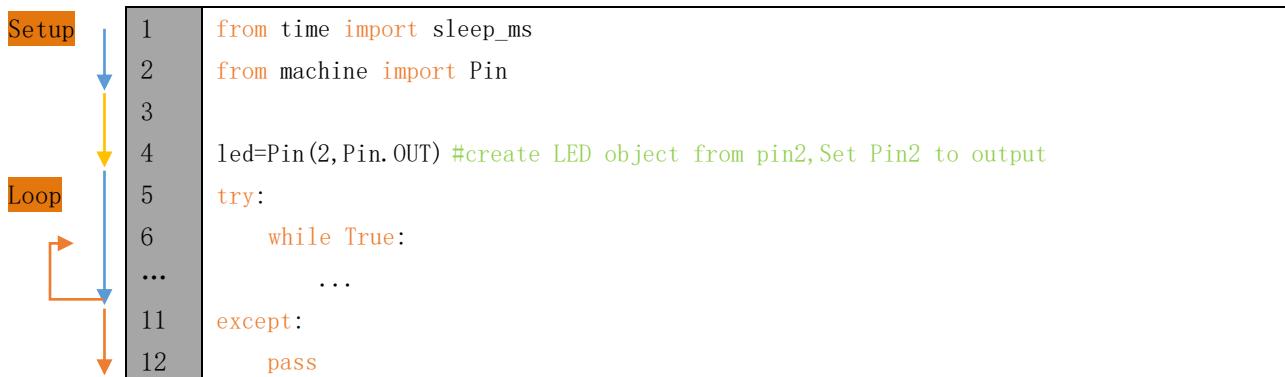
Any concerns? [✉ support@freenove.com](mailto:support@freenove.com)

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

The following is the program code:

```
1 from time import sleep_ms
2 from machine import Pin
3
4 led=Pin(2,Pin.OUT) #create LED object from pin2, Set Pin2 to output
5 try:
6     while True:
7         led.value(1) #Set led turn on
8         sleep_ms(1000)
9         led.value(0) #Set led turn off
10        sleep_ms(1000)
11 except:
12     pass
```

Each time a new file is opened, the program will be executed from top to bottom. When encountering a loop construction, it will execute the loop statement according to the loop condition.



`Print()` function is used to print data to Terminal. It can be executed in Terminal directly or be written in a Python file and executed by running the file.

```
print("Hello world!")
```

Each time when using the functions of ESP8266, you need to import modules corresponding to those functions: Import sleep_ms module of time module and Pin module of machine module.

```
1 from time import sleep_ms  
2 from machine import Pin
```

Configure GPIO2 of ESP8266 to output mode and assign it to an object named "led".

```
4 led=Pin(2,Pin.OUT) #create LED object from pin2, Set Pin2 to output
```

It means that from now on, LED represents GPIO2 that is in output mode.

Set the value of LED to 1 and GPIO2 will output high level.

```
7     led.value(1) #Set led turn on
```

Set the value of LED to 0 and GPIO2 will output low level.

9 led.value(0) #Set led turn on

Execute codes in a while loop

6	...	while True:
---	-----	-------------

Any concerns? support@freenove.com

Put statements that may cause an error in “try” block and the executing statements when an error occurs in “except” block. In general, when the program executes statements, it will execute those in “try” block. However, when an error occurs to ESP8266 due to some interference or other reasons, it will execute statements in “except” block.

“Pass” is an empty statement. When it is executed, nothing happens. It is useful as a placeholder to make the structure of a program look better.

```
5   try:  
...  
11  ...  
12  except:  
    pass
```

The single-line comment of Micropython starts with a “#” and continues to the end of the line. Comments help us to understand code. When programs are running, Thonny will ignore comments.

```
9  #Set led turn on
```

MicroPython uses indentations to distinguish different blocks of code instead of braces. The number of indentations is changeable, but it must be consistent throughout one block. If the indentation of the same code block is inconsistent, it will cause errors when the program runs.

```
6  while True:  
7      led.value(1) #Set led turn on  
8      sleep_ms(1000)  
9      led.value(0) #Set led turn off  
10     sleep_ms(1000)
```

How to import python files

Whether to import the built-in python module or to import that written by users, the command “import” is needed.

If you import the module directly you should indicate the module to which the function or attribute belongs when using the function or attribute (constant, variable) in the module. The format should be: <module name>.<function or attribute>, otherwise an error will occur.

```
import random  
  
num = random.randint(1, 100)  
print(num)
```

If you only want to import a certain function or attribute in the module, use the from...import statement. The format is as follows.

```
from random import randint  
num = randint(1, 100)  
print(num)
```

When using “from...import” statement to import function, to avoid conflicts and for easy understanding, you can use “as” statement to rename the imported function, as follows.

```
from random import randint as rand  
num = rand(1, 100)  
print(num)
```



Reference

Class machine

Before each use of the **machine** module, please add the statement “**import machine**” to the top of python file.

machine.freq(freq_val): When freq_val is not specified, it is to return to the current CPU frequency; Otherwise, it is to set the current CPU frequency.

freq_val: 80000000(80MHz)、160000000(160MHz)、240000000(240MHz)

machine.reset(): A reset function. When it is called, the program will be reset.

machine.unique_id(): Obtains MAC address of the device.

machine.idle(): Turns off any temporarily unused functions on the chip and its clock, which is useful to reduce power consumption at any time during short or long periods.

machine.disable_irq(): Disables interrupt requests and return the previous IRQ state. The disable_irq () function and enable_irq () function need to be used together; Otherwise the machine will crash and restart.

machine.enable_irq(state): To re-enable interrupt requests. The parameter **state** should be the value that was returned from the most recent call to the disable_irq() function

machine.time_pulse_us(pin, pulse_level, timeout_us=1000000):

Tests the duration of the external pulse level on the given pin and returns the duration of the external pulse level in microseconds. When pulse level = 1, it tests the high level duration; When pulse level = 0, it tests the low level duration.

If the setting level is not consistent with the current pulse level, it will wait until they are consistent, and then start timing. If the set level is consistent with the current pulse level, it will start timing immediately.

When the pin level is opposite to the set level, it will wait for timeout and return “-2”. When the pin level and the set level is the same, it will also wait timeout but return “-1”. **timeout_us** is the duration of timeout.

Class Pin(id[, mode, pull, value])

Before each use of the **Pin** module, please add the statement “**from machine import Pin**” to the top of python file.

id: Arbitrary pin number

mode: Mode of pins

Pin.IN: Input Mode

Pin.OUT: Output Mode

Pin.OPEN_DRAIN: Open-drain Mode

Pull: Whether to enable the internal pull up and down mode

None: No pull up or pull down resistors

Pin.PULL_UP: Pull-up Mode, outputting high level by default

Pin.PULL_DOWN: Pull-down Mode, outputting low level by default

Value: State of the pin level, 0/1

Pin.init(mode, pull): Initialize pins

Pin.value([value]): Obtain or set state of the pin level, return 0 or 1 according to the logic level of pins.

Without parameter, it reads input level. With parameter given, it is to set output level.

value: It can be either True/False or 1/0.

Pin.irq(trigger, handler): Configures an interrupt handler to be called when the pin level meets a condition.
trigger:

Pin.IRQ_FALLING: interrupt on falling edge

Pin.IRQ_RISING: interrupt on rising edge

3: interrupt on both edges

Handler: callback function

Class time

Before each use of the **time** module, please add the statement “**import time**” to the top of python file

time.sleep(sec): Sleeps for the given number of seconds

sec: This argument should be either an int or a float.

time.sleep_ms(ms): Sleeps for the given number of milliseconds, ms should be an int.

time.sleep_us(us): Sleeps for the given number of microseconds, us should be an int.

time.time(): Obtains the timestamp of CPU, with second as its unit.

time.ticks_ms(): Returns the incrementing millisecond counter value, which recounts after some values.

time.ticks_us(): Returns microsecond

time.ticks_cpu(): Similar to ticks_ms() and ticks_us(), but it is more accurate(return clock of CPU).

time.ticks_add(ticks, delta): Gets the timestamp after the offset.

ticks: ticks_ms()、ticks_us()、ticks_cpu()

delta: Delta can be an arbitrary integer number or numeric expression

time.ticks_diff(old_t, new_t): Calculates the interval between two timestamps, such as ticks_ms(), ticks_us() or ticks_cpu().

old_t: Starting time

new_t: Ending time



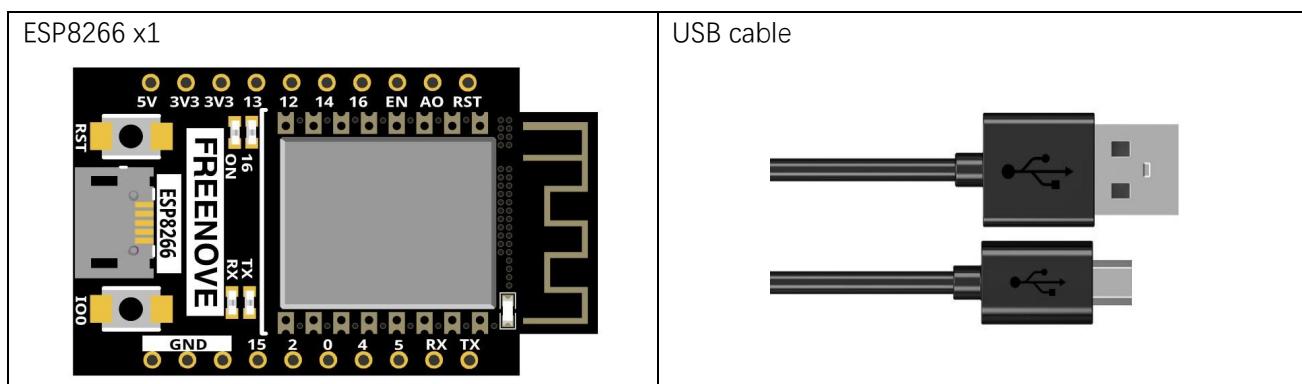
Chapter 2 Serial Communication

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

Project 2.1 Serial Print

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

Component List



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 two devices use 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.

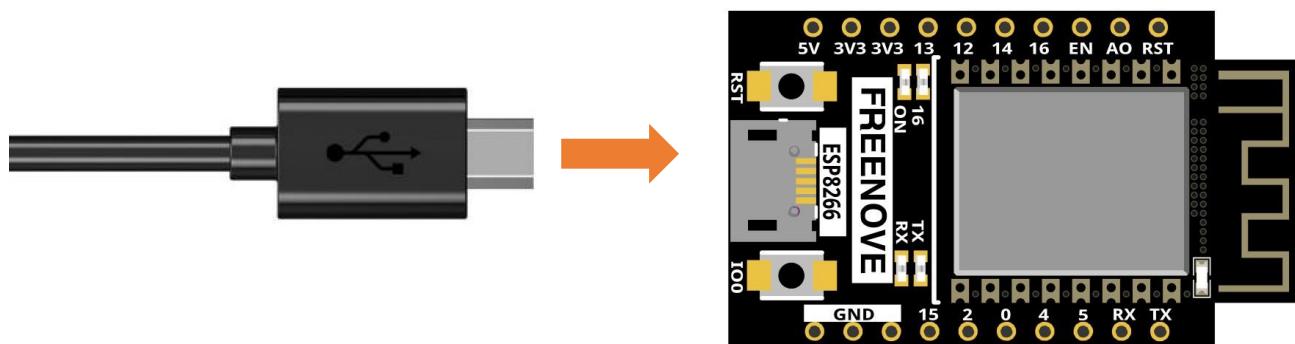
Serial port on ESP8266

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



Circuit

Connect Freenove ESP8266 to the computer with USB cable.

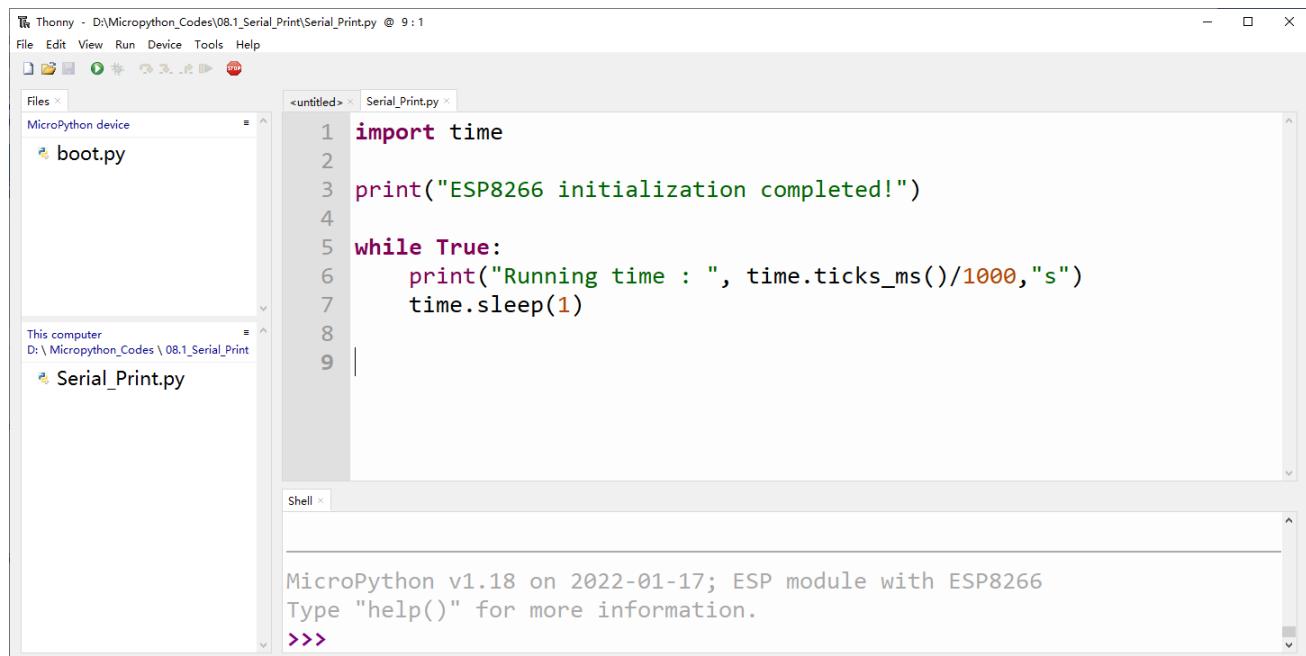


Code

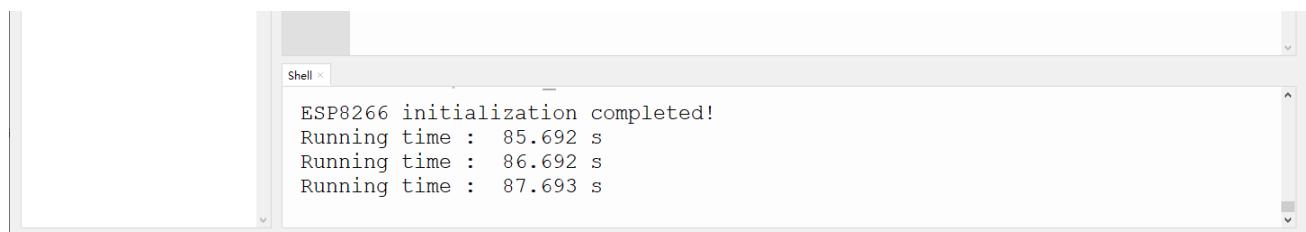
Move the program folder “**Freenove_ESP8266_Board/Python/Python_Codes**” to disk(D) in advance with the path of “**D:/Micropython_Codes**”.

Open “Thonny”, click “This computer” → “D:.” → “Micropython_Codes” → “08.1_Serial_Print” and double “Serial_Print.py”.

08.1_Serial_Print



Click “Run current script” and observe the changes of “Shell”, which will display the time when ESP8266 is powered on once per second.



The following is the program code:

```

1 import time
2
3 print("ESP8266 initialization completed!")
4
5 while True:
6     print("Running time : ", time.ticks_ms()/1000, "s")
7     time.sleep(1)

```

Reference

Class UART

Before each use of **UART** module, please add the statement “**from machine import UART**” to the top of python file.

UART(id, baudrate, bits, parity, rx, tx, stop, timeout): Define serial ports and configure parameters for them.

id: Serial Number. The available serial port number is 1 or 2

baudrate: Baud rate

bits: The number of each character.

parity: Check even or odd, with 0 for even checking and 1 for odd checking.

rx, tx: UAPT's reading and writing pins

Pin(0)、Pin(2)、Pin(4)、Pin(5)、Pin(9)、Pin(10)、Pin(12~19)、Pin(21~23)、Pin(25)、Pin(26)、
Pin(34~36)、Pin(39)

Note: Pin(1) and Pin(3) are occupied and not recommend to be used as tx,rx.

stop: The number of stop bits, and the stop bit is 1 or 2.

timeout: timeout period (Unit: millisecond)

$0 < \text{timeout} \leq 0x7FFF FFFF$ (decimal: $0 < \text{timeout} \leq 2147483647$)

UART.init(baudrate, bits, parity, stop, tx, rx, rts, cts)): Initialize serial ports

tx: writing pins of uart

rx: reading pins of uart

rts: rts pins of uart

cts: cts pins of uart

UART.read(nbytes): Read nbytes bytes

UART.read(): Read data

UART.write(buf): Write byte buffer to UART bus

UART.readline(): Read a line of data, ending with a newline character.

UART.readinto(buf): Read and write data into buffer.

UART.readinto(buf, nbytes): Read and write data into buffer.

UART.any(): Determine whether there is data in serial ports. If yes, return the number of bytes; Otherwise, return 0.

Project 2.2 Serial Read and Write

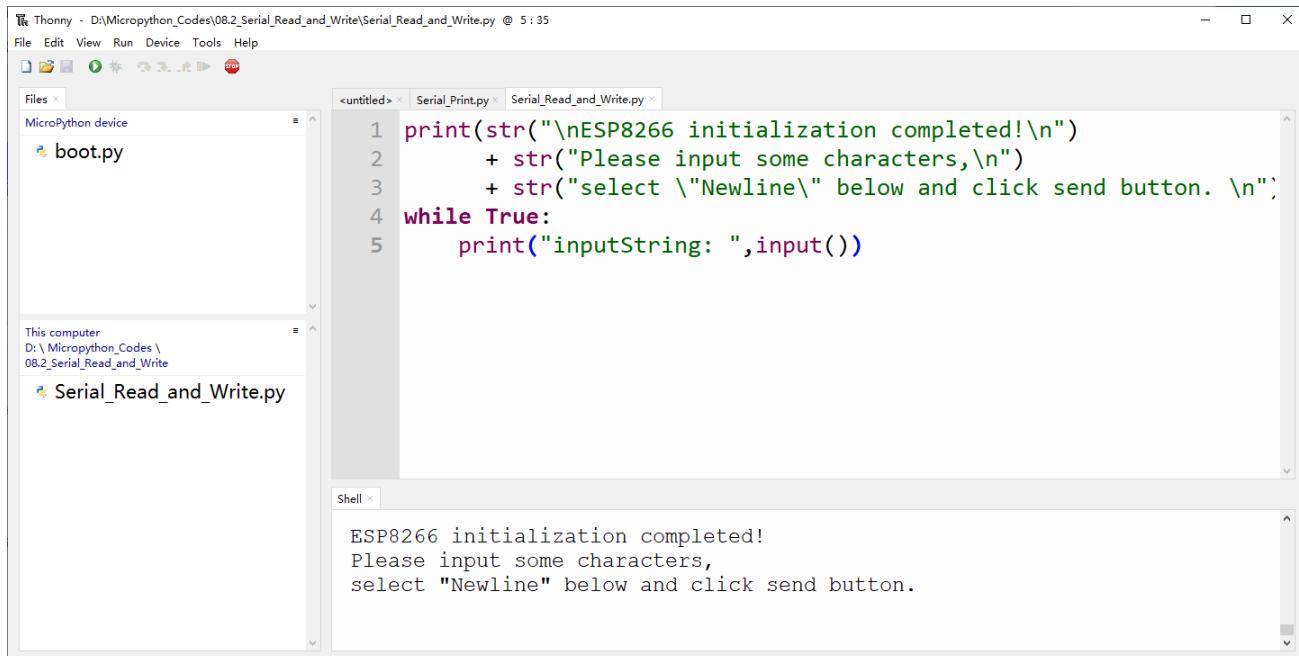
From last section, we use Serial port on Freenove ESP8266 to send data to a computer, now we will use that to receive data from computer.

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

Code

Open “Thonny”, click “This computer” → “D:” → “Micropython_Codes” → “08.2_Serial_Read_and_Write” and double click “Serial_Read_and_Write.py”.

08.2_Serial_Read_and_Write

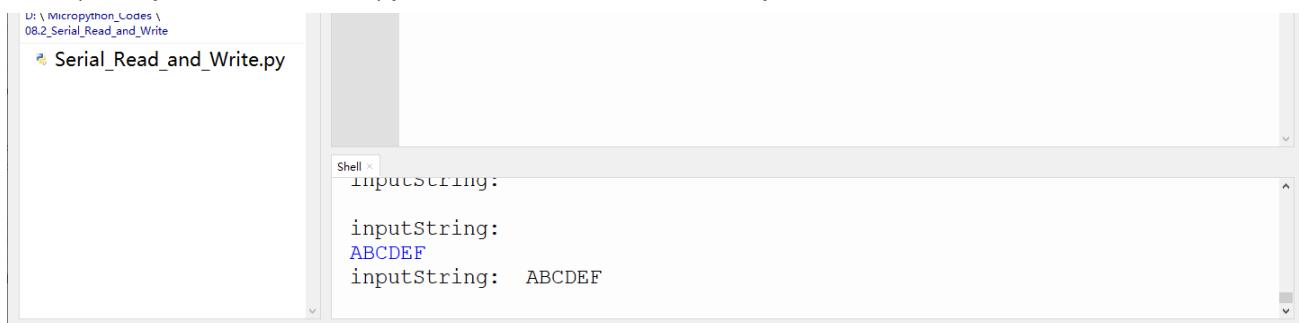


```

print(str("\nESP8266 initialization completed!\n"))
+ str("Please input some characters,\n")
+ str("select \"Newline\" below and click send button. \n")
while True:
    print("inputString: ",input())

```

Click “Run current script” and ESP8266 will print out data at “Shell” and wait for users to enter any messages. Press Enter to end the input, and “Shell” will print out data that the user entered. If you want to use other serial ports, you can use other python files in the same directory.



```

U:\Micropython_Codes\08.2_Serial_Read_and_Write
Serial_Read_and_Write.py

Shell >
inputString:

inputString:
ABCDEF
inputString: ABCDEF

```

The following is the program code:

```
1 print(str("\nESP8266 initialization completed!\n"))
2     + str("Please input some characters, \n")
3     + str("select \"Newline\" below and click send button. \n"))
4 while True:
5     print("inputString: ", input())
```



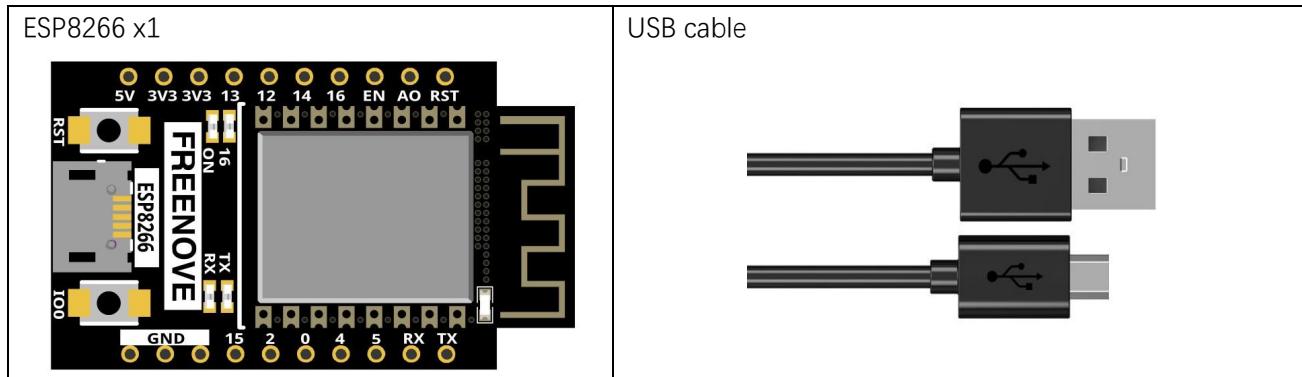
Chapter 3 WiFi Working Modes

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

ESP8266 has 3 different WiFi operating modes: Station mode, AP mode and AP+Station mode. All WiFi programming projects must be configured with WiFi operating mode before using WiFi, otherwise WiFi cannot be used.

Project 3.1 Station mode

Component List



Component knowledge

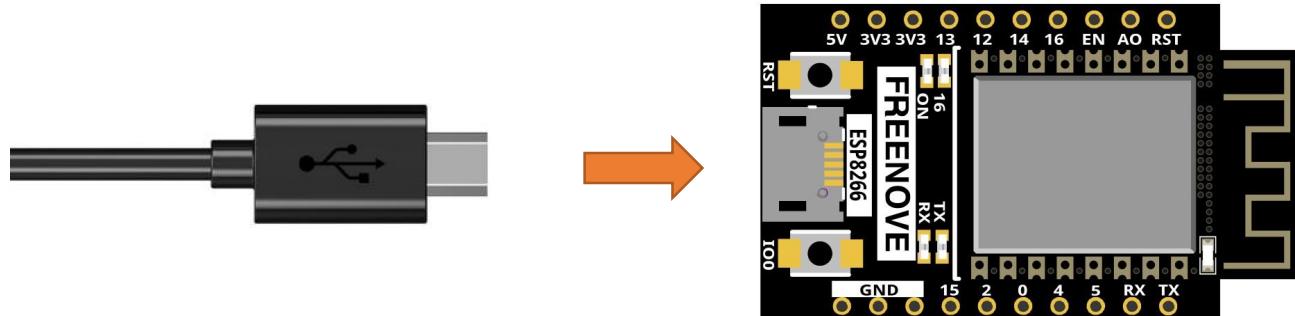
Station mode

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.



Code

Move the program folder “**Freenove_ESP8266_Board/Python/Python_Codes**” to disk(D) in advance with the path of “**D:/Micropython_Codes**”.

Open “Thonny”, click “This computer” → “D:” → “Micropython_Codes” → “28.1_Station_mode” and double click “Station_mode.py”.

28.1_Station_mode

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

- File Menu:** File, Edit, View, Run, Device, Tools, Help.
- File Explorer:** Shows files: boot.py and Station_mode.py (selected).
- Code Editor:** Displays the Python code for WiFi station mode. Lines 4 and 5 are highlighted with an orange rectangle and have a callout bubble pointing to them with the text "Enter the correct Router name and password." The code uses the `network` module to connect to a WiFi router.
- Shell:** Shows the MicroPython environment information: MicroPython v1.18 on 2022-01-17; ESP module with ESP8266. It also shows the prompt >>>

```
import time
import network

ssidRouter = '*****' #Enter the router name
passwordRouter = '*****' #Enter the router password

def STA_Setup(ssidRouter,passwordRouter):
    print("Setup start")
    sta_if = network.WLAN(network.STA_IF)
    if not sta_if.isconnected():
        print('connecting to',ssidRouter)
        sta_if.active(True)
        sta_if.connect(ssidRouter,passwordRouter)
        while not sta_if.isconnected():
            pass
    print('Connected, IP address:', sta_if.ifconfig())
    print("Setup End")

try:
    STA_Setup(ssidRouter,passwordRouter)
except:
    sta_if.disconnect()
```

Because the names and passwords of routers in various places are different, before the Code 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, wait for ESP8266 to connect to your router and print the IP address assigned by the router to ESP8266 in "Shell".

```
Shell < 
MicroPython v1.18 on 2022-01-17; ESP module with ESP8266
Type "help()" for more information.
>>> %Run -c $EDITOR_CONTENT
Setup start
Connected, IP address: ('192.168.1.113', '255.255.255.0', '192.168.1.1', '8.8.8.8')
Setup End
>>>
```

The following is the program code:

```
1 import time
2 import network
3
4 ssidRouter      = '*****' #Enter the router name
5 passwordRouter = '*****' #Enter the router password
6
7 def STA_Setup(ssidRouter,passwordRouter):
8     print("Setup start")
9     sta_if = network.WLAN(network.STA_IF)
10    if not sta_if.isconnected():
11        print(' connecting to',ssidRouter)
12        sta_if.active(True)
13        sta_if.connect(ssidRouter,passwordRouter)
14        while not sta_if.isconnected():
15            pass
16        print('Connected, IP address:', sta_if.ifconfig())
17        print("Setup End")
18
19 try:
20     STA_Setup(ssidRouter,passwordRouter)
21 except:
22     sta_if.disconnect()
```

Import network module.

```
2 import network
```

Enter correct router name and password.

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

Set ESP8266 in Station mode.

```
9 sta_if = network.WLAN(network.STA_IF)
```

Activate ESP8288 Station mode, initiate a connection request to the router and enter the password to connect.

```
12     sta_if.active(True)
13     sta_if.connect(ssidRouter, passwordRouter)
```

Wait for ESP8266 to connect to router until they connect to each other successfully.

```
14     while not sta_if.isconnected():
15         pass
```

Print the IP address assigned to ESP8266 in “Shell”.

```
16     Print('Connected, IP address:', sta_if.ifconfig())
```

Reference

Class network

Before each use of **network**, please add the statement “**import network**” to the top of the python file.

WLAN(interface_id): Set to WiFi mode.

network.STA_IF: Client, connecting to other WiFi access points.

network.AP_IF: Access points, allowing other WiFi clients to connect.

active(is_active): With parameters, it is to check whether to activate the network interface; Without parameters, it is to query the current state of the network interface.

scan(ssid, bssid, channel, RSSI, authmode, hidden): Scan for wireless networks available nearby (only scan on STA interface), return a tuple list of information about the WiFi access point.

bssid: The hardware address of the access point, returned in binary form as a byte object. You can use `ubinascii.hexlify()` to convert it to ASCII format.

authmode: Access type

```
AUTH_OPEN = 0
AUTH_WEP = 1
AUTH_WPA_PSK = 2
AUTH_WPA2_PSK = 3
AUTH_WPA_WPA2_PSK = 4
AUTH_MAX = 6
```

Hidden: Whether to scan for hidden access points

False: Only scanning for visible access points

True: Scanning for all access points including the hidden ones.

isconnected(): Check whether ESP8266 is connected to AP in Station mode. In STA mode, it returns True if it is connected to a WiFi access point and has a valid IP address; Otherwise it returns False.

connect(ssid, password): Connecting to wireless network.

ssid: WiFi name

password: WiFi password

disconnect(): Disconnect from the currently connected wireless network.

Project 3.2 AP mode

Component List & Circuit

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

Component knowledge

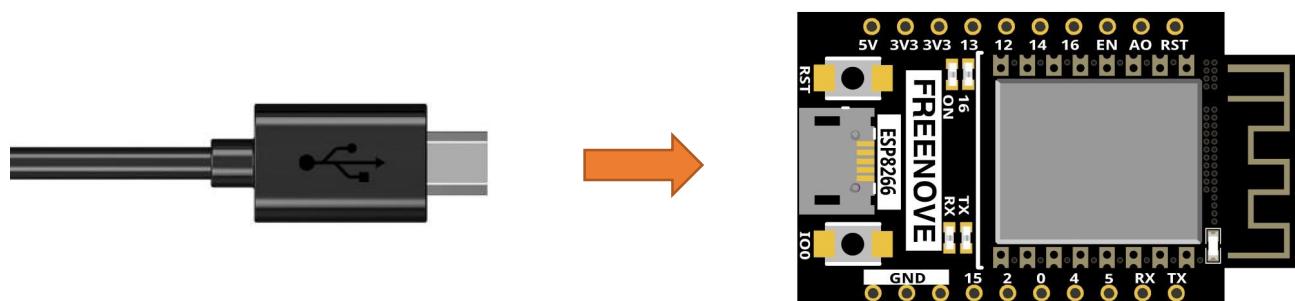
AP mode

When ESP8266 selects AP mode, it creates a hotspot network that is separated 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.

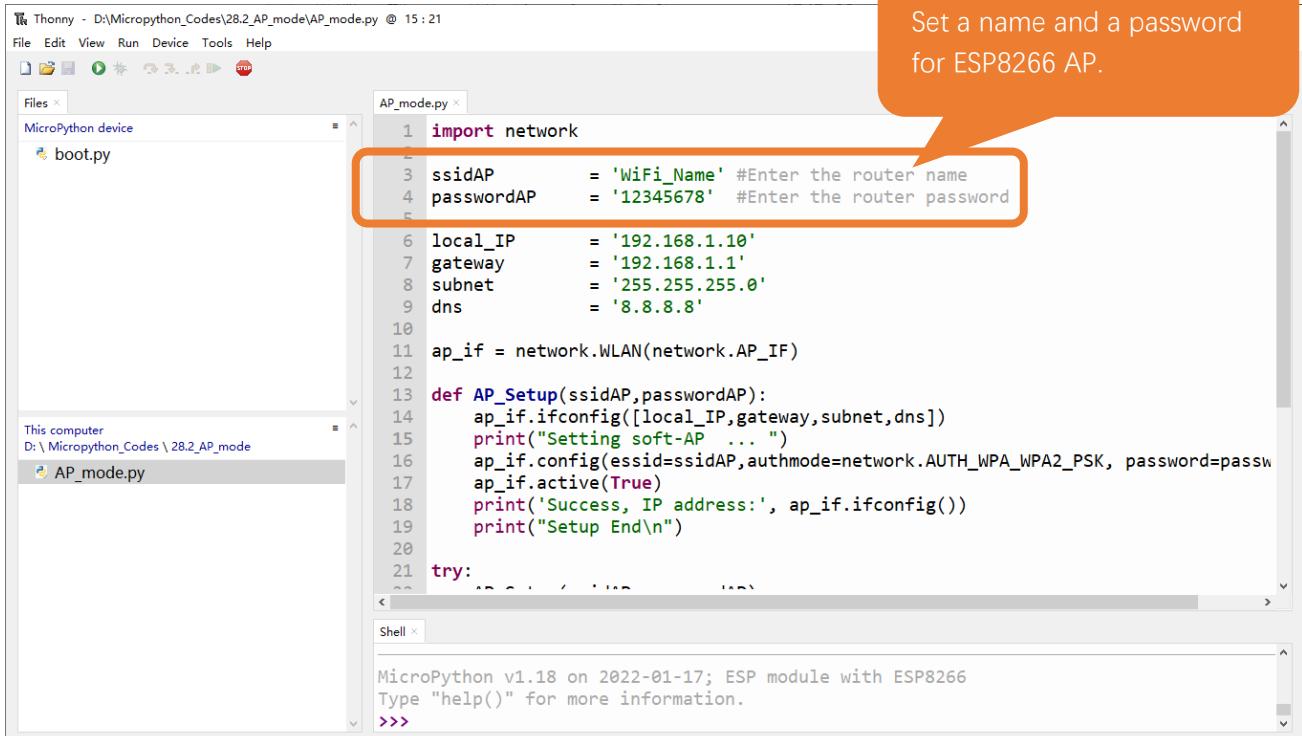


Code

Move the program folder “Freenove_ESP8266_Board/Python/Python_Codes” to disk(D) in advance with the path of “D:/Micropython_Codes”.

Open “Thonny”, click “This computer” → “D:” → “Micropython_Codes” → “28.2_AP_mode”. and double click “AP_mode.py”.

28.2_AP_mode



Before the Code 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.

Click “Run current script”, open the AP function of ESP8266 and print the access point information.

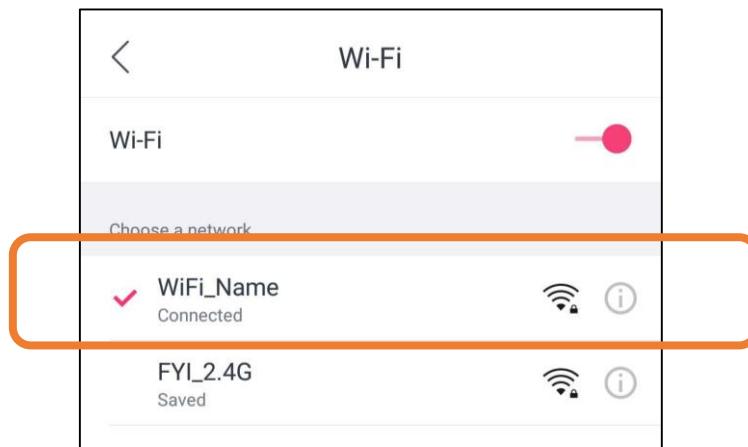
```
Shell x

MicroPython v1.18 on 2022-01-17; ESP module with ESP8266
Type "help()" for more information.
>>> %Run -c $EDITOR_CONTENT

Setting soft-AP ...
Success, IP address: ('192.168.1.10', '192.168.1.1', '255.255.255.0', '8.8.8.8')
Setup End

>>>
```

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 Code. You can enter the password "12345678" to connect it or change its AP name and password by modifying Code.



The following is the program code:

```

1 import network
2
3 ssidAP      = 'WiFi_Name' #Enter the router name
4 passwordAP   = '12345678' #Enter the router password
5
6 local_IP     = '192.168.1.10'
7 gateway      = '192.168.1.1'
8 subnet       = '255.255.255.0'
9 dns          = '8.8.8.8'
10
11 ap_if = network.WLAN(network.AP_IF)
12
13 def AP_Setup(ssidAP, passwordAP):
14     ap_if.ifconfig([local_IP, gateway, subnet, dns])
15     print("Setting soft-AP ... ")
16     ap_if.config(essid=ssidAP, authmode=network.AUTH_WPA_WPA2_PSK, password=passwordAP)
17     ap_if.active(True)
18     print(' Success, IP address:', ap_if.ifconfig())
19     print("Setup End\n")
20
21 try:
22     AP_Setup(ssidAP, passwordAP)
23 except:
24     ap_if.disconnect()

```

Import network module.

1	import network
---	----------------

Enter correct AP name and password.

```
3     ssidAP      = 'WiFi_Name' #Enter the router name
4     passwordAP   = '12345678' #Enter the router password
```

Set ESP8266 in AP mode.

```
11    ap_if = network.WLAN(network.AP_IF)
```

Configure IP address, gateway and subnet mask for ESP8266.

```
14    ap_if.ifconfig([local_IP, gateway, subnet, dns])
```

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

```
16    ap_if.config(essid=ssidAP, authmode=network.AUTH_WPA_WPA2_PSK, password=passwordAP)
17    ap_if.active(True)
```

If the program is running abnormally, the AP disconnection function will be called.

```
14    ap_if.disconnect()
```

Reference

Class network

Before each use of **network**, please add the statement “**import network**” to the top of the python file.

WLAN(interface_id): Set to WiFi mode.

network.STA_IF: Client, connecting to other WiFi access points

network.AP_IF: Access points, allowing other WiFi clients to connect

active(is_active): With parameters, it is to check whether to activate the network interface; Without parameters, it is to query the current state of the network interface

isconnected(): In AP mode, it returns True if it is connected to the station; otherwise it returns False.

connect(ssid, password): Connecting to wireless network

ssid: WiFi name

password: WiFi password

config(essid, channel): To obtain the MAC address of the access point or to set the WiFi channel and the name of the WiFi access point.

ssid: WiFi account name

channel: WiFi channel

ifconfig([(ip, subnet, gateway, dns)]): Without parameters, it returns a 4-tuple (ip, subnet_mask, gateway, DNS_server); With parameters, it configures static IP.

ip: IP address

subnet_mask: subnet mask

gateway: gateway

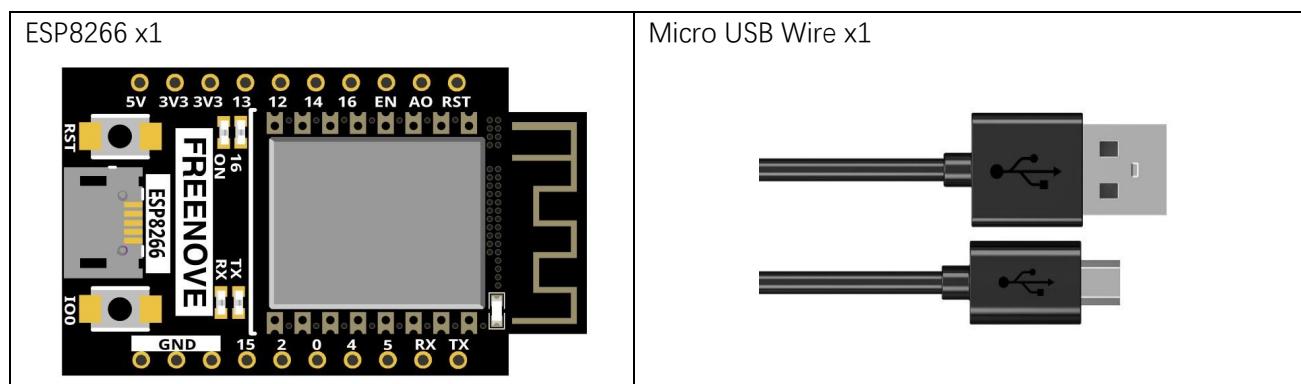
DNS_server: DNS server

disconnect(): Disconnect from the currently connected wireless network

status(): Return the current status of the wireless connection

Project 3.3 AP+Station mode

Component List



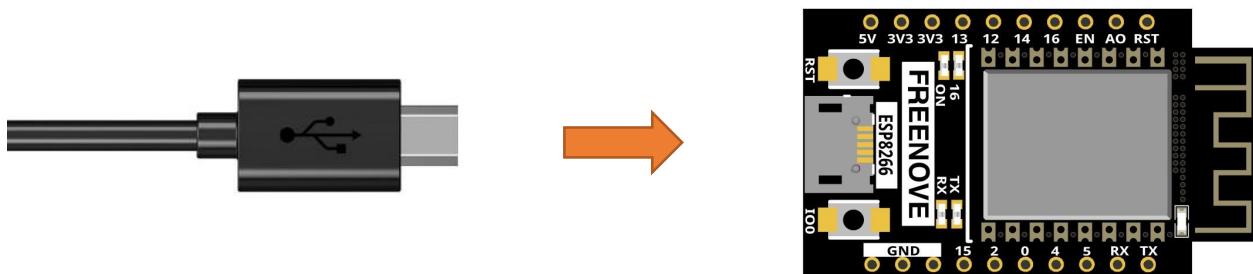
Component knowledge

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.



Code

Move the program folder “**Freenove_ESP8266_Board/Python/Python_Codes**” to disk(D) in advance with the path of “**D:/Micropython_Codes**”.

Open “Thonny”, click “This computer” → “D:” → “Micropython_Codes” → “28.3_AP+STA_mode” and double click “AP+STA_mode.py”.

28.3_AP+STA_mode

```

import network
ssidRouter = '*****' #Enter the router name
passwordRouter = '*****' #Enter the router password
ssidAP = 'WiFi_Name'#Enter the AP name
passwordAP = '12345678' #Enter the AP password
local_IP = '192.168.4.150'
gateway = '192.168.4.1'
subnet = '255.255.255.0'
dns = '8.8.8.8'

sta_if = network.WLAN(network.STA_IF)
ap_if = network.WLAN(network.AP_IF)

def STA_Setup(ssidRouter,passwordRouter):
    print("Setting soft-STA ... ")
    if not sta_if.isconnected():
        print('connecting to',ssidRouter)
        sta_if.active(True)
        sta_if.connect(ssidRouter,passwordRouter)

```

MicroPython v1.18 on 2022-01-17; ESP module with ESP8266
Type "help()" for more information.
>>>

It is analogous to Project 3.1 and Project 3.2. Before running the Code, you need to modify ssidRouter, passwordRouter, ssidAP and passwordAP shown in the box of the illustration above.

After making sure that the code is modified correctly, click “Run current script” and the “Shell” will display as follows:

```

MicroPython v1.18 on 2022-01-17; ESP module with ESP8266
Type "help()" for more information.
>>> %Run -c $EDITOR_CONTENT

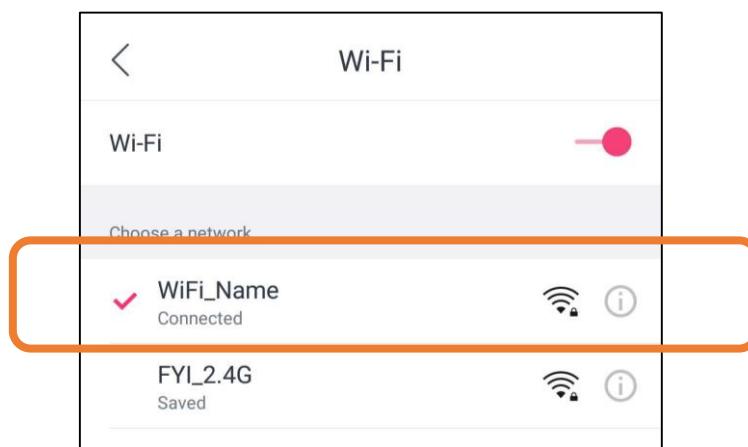
Setting soft-AP ...
Success, IP address: ('192.168.4.150', '192.168.4.1', '255.255.255.0', '8.8.8.8')
Setup End

Setting soft-STA ...
Connected, IP address: ('192.168.1.113', '255.255.255.0', '192.168.1.1', '8.8.8.8')
Setup End

>>>

```

Turn on the WiFi scanning function of your phone, and you can see the ssidAP on ESP8266.



The following is the program code:

```

1 import network
2
3 ssidRouter      = '*****' #Enter the router name
4 passwordRouter = '*****' #Enter the router password
5
6 ssidAP         = 'WiFi_Name' #Enter the AP name
7 passwordAP     = '12345678' #Enter the AP password
8
9 local_IP       = '192.168.4.150'
10 gateway        = '192.168.4.1'
11 subnet         = '255.255.255.0'
12 dns            = '8.8.8.8'
13
14 sta_if = network.WLAN(network.STA_IF)
15 ap_if = network.WLAN(network.AP_IF)
16
17 def STA_Setup(ssidRouter, passwordRouter):
18     print("Setting soft-STA ... ")
19     if not sta_if.isconnected():
20         print('connecting to',ssidRouter)
21         sta_if.active(True)
22         sta_if.connect(ssidRouter, passwordRouter)
23         while not sta_if.isconnected():
24             pass
25     print('Connected, IP address:', sta_if.ifconfig())
26     print("Setup End")
27
28 def AP_Setup(ssidAP, passwordAP):
29     ap_if.ifconfig([local_IP, gateway, subnet, dns])
30     print("Setting soft-AP ... ")

```

```
31     ap_if.config(essid=ssidAP, authmode=network.AUTH_WPA_WPA2_PSK, password=passwordAP)
32     ap_if.active(True)
33     print(' Success, IP address:', ap_if.ifconfig())
34     print("Setup End\n")
35
36 try:
37     AP_Setup(ssidAP, passwordAP)
38     STA_Setup(ssidRouter, passwordRouter)
39 except:
40     sta_if.disconnect()
41     ap_if.disconnect()
```

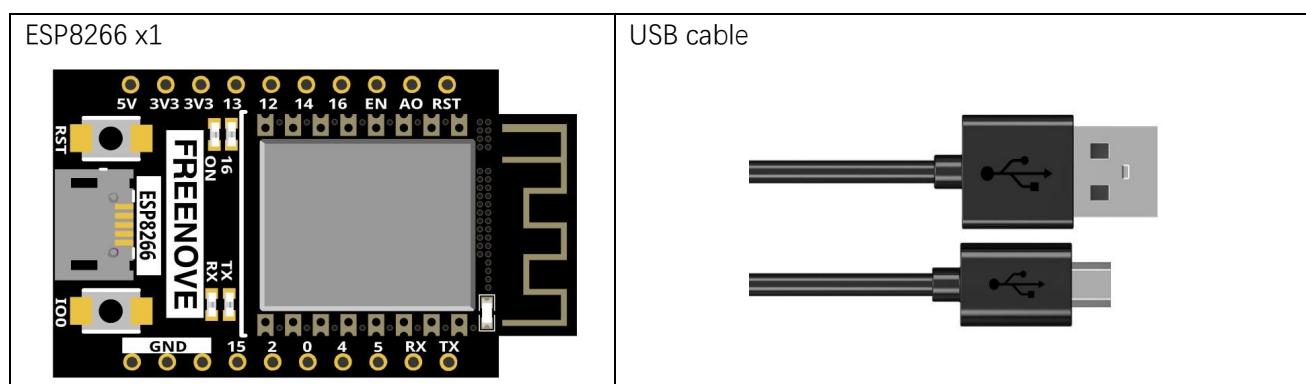
Chapter 4 TCP/IP

In this chapter, we will introduce how ESP8266 implements network communications based on TCP/IP protocol. There are two roles in TCP/IP communication, namely Server and Client, which will be implemented respectively with two projects in this chapter.

Project 4.1 As Client

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

Component List



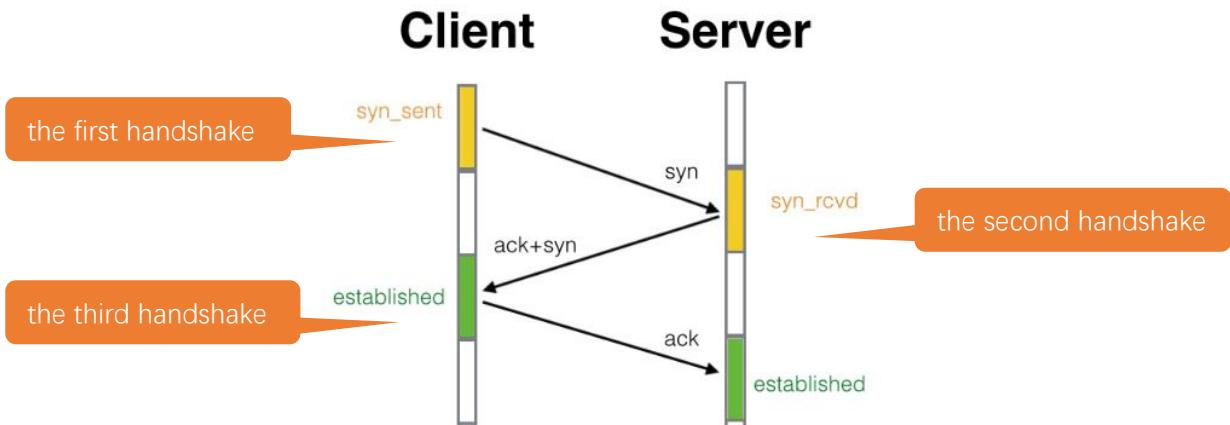
Component knowledge

TCP connection

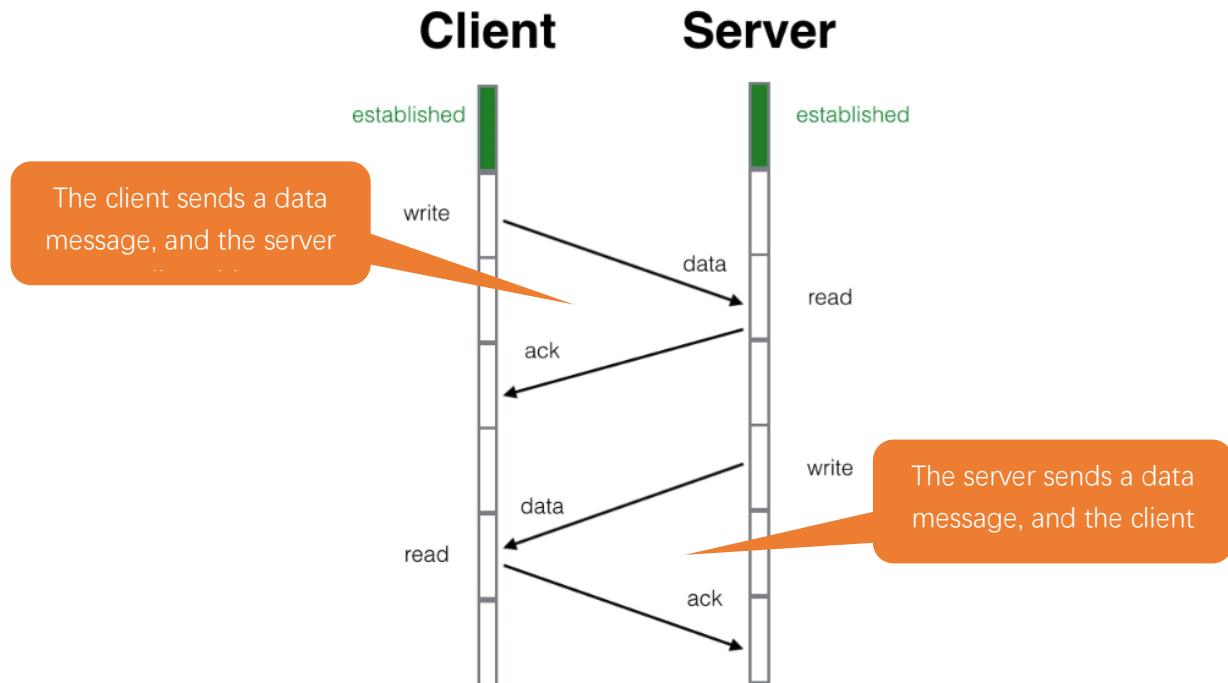
Before transmitting data, TCP needs to establish a logical connection between the sending end and the receiving end. It provides reliable and error-free data transmission between the two computers. In the TCP connection, the client and the server must be clarified. The client sends a connection request 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's download section. At the top, there are links for "Processing", "p5.js", "Processing.py", "Processing for Android", "Processing for Pi", and "Processing Foundation". Below this is a search bar. The main content area features a large "Processing" logo with a geometric background. To the left is a sidebar with links: "Cover", "Download", "Donate", "Exhibition", "Reference", "Libraries", "Tools", "Environment", "Tutorials", "Examples", "Books", "Overview", and "People". In the center, it says "Download Processing. Processing is available for Linux, Mac OS X, and Windows. Select your choice to download the software below." It shows the "3.5.4 (17 January 2020)" release with download links for "Windows 64-bit", "Windows 32-bit", "Linux 64-bit", and "Mac OS X". Below this, there are links to "» Github", "» Report Bugs", "» Wiki", "» Supported Platforms", and a note about "changes in 3.0".

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_ESP8266_Board/Codes/Micropython_Codes/29.1_TCP_as_Client/sketchWiFi/sketchWiFi.pde**”. Click “Run”.

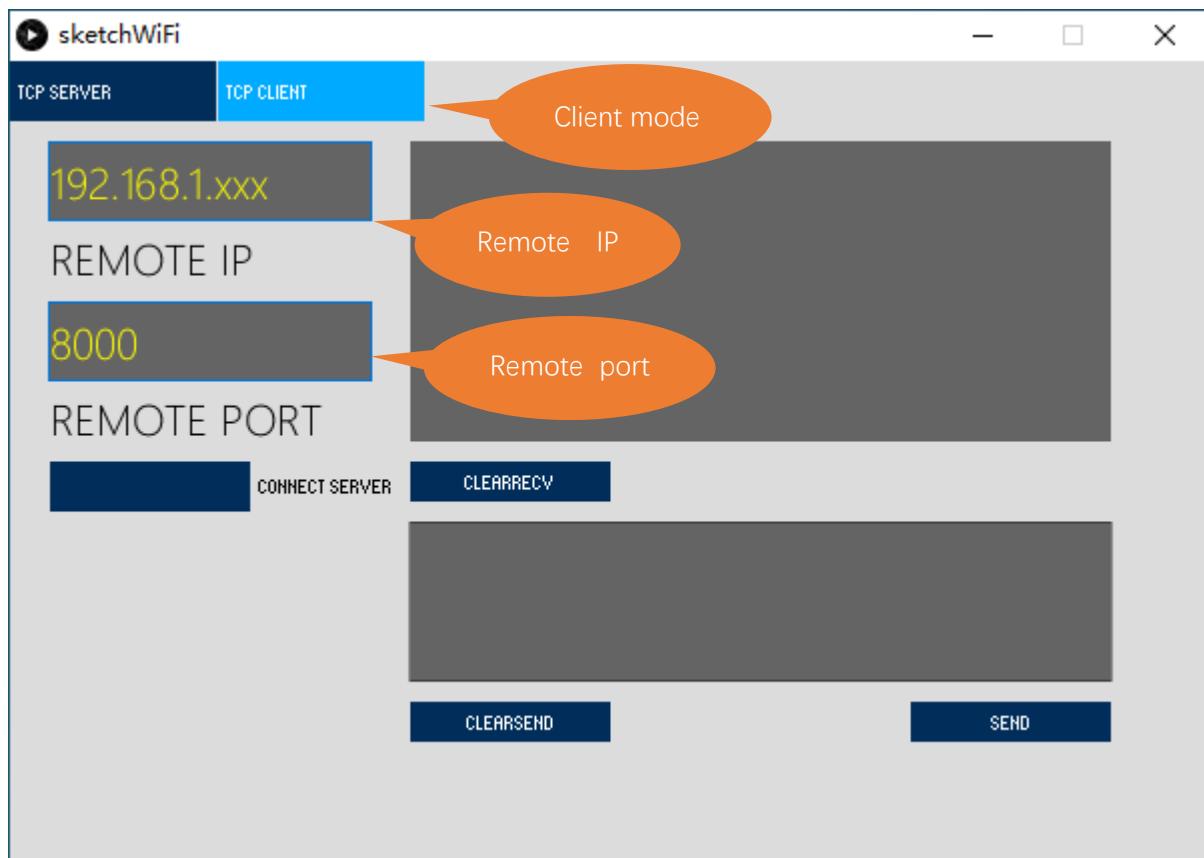


The new pop-up interface is as follows. If ESP8266 is used as Client, select TCP SERVER mode for sketchWiFi.



When sketchWiFi selects TCP SERVER mode, ESP8266 Code 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, and 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.

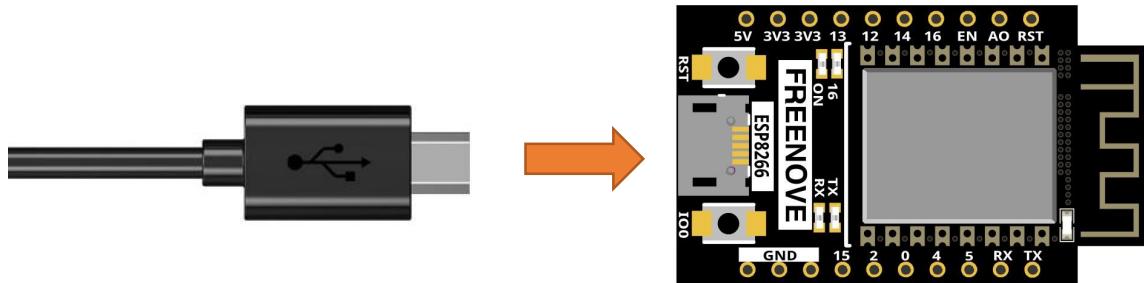
clear receive: clear out the content in the receiving text box

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

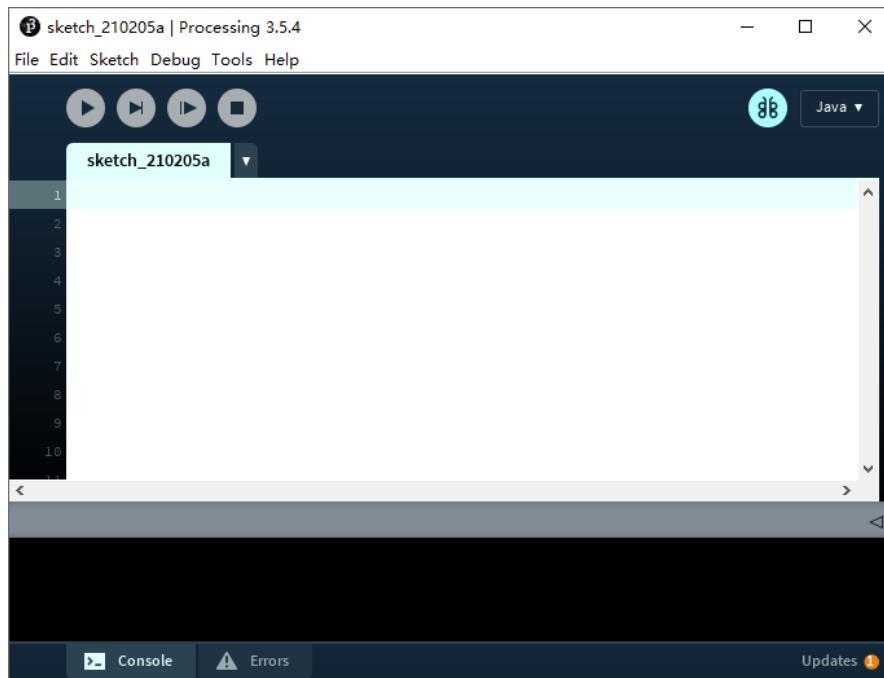
Connect Freenove ESP8266 to the computer using USB cable.



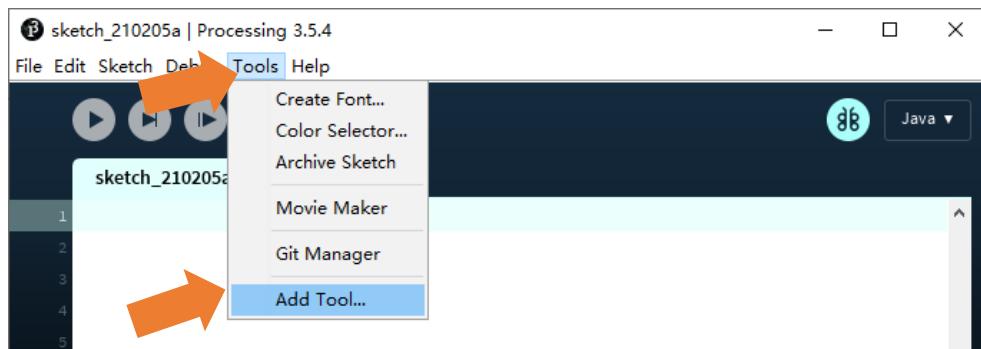
Code

If you have not installed “ControlIP5”, please follow the following steps to continue the installation, if you have installed, please skip this section.

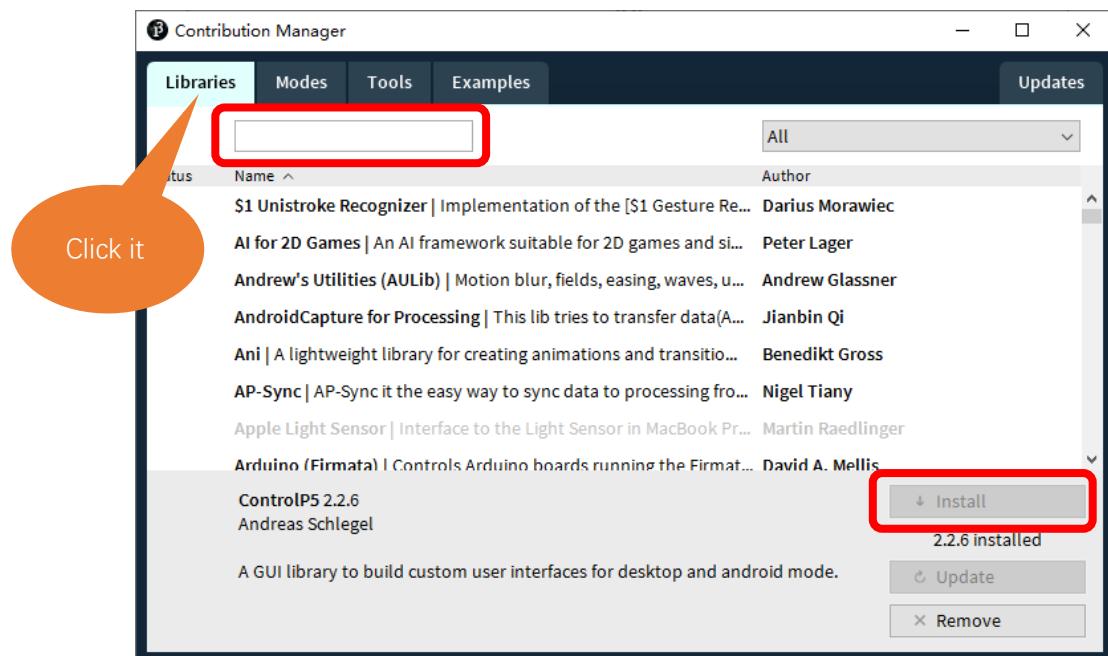
Open Processing.



Click Add Tool under Tools.

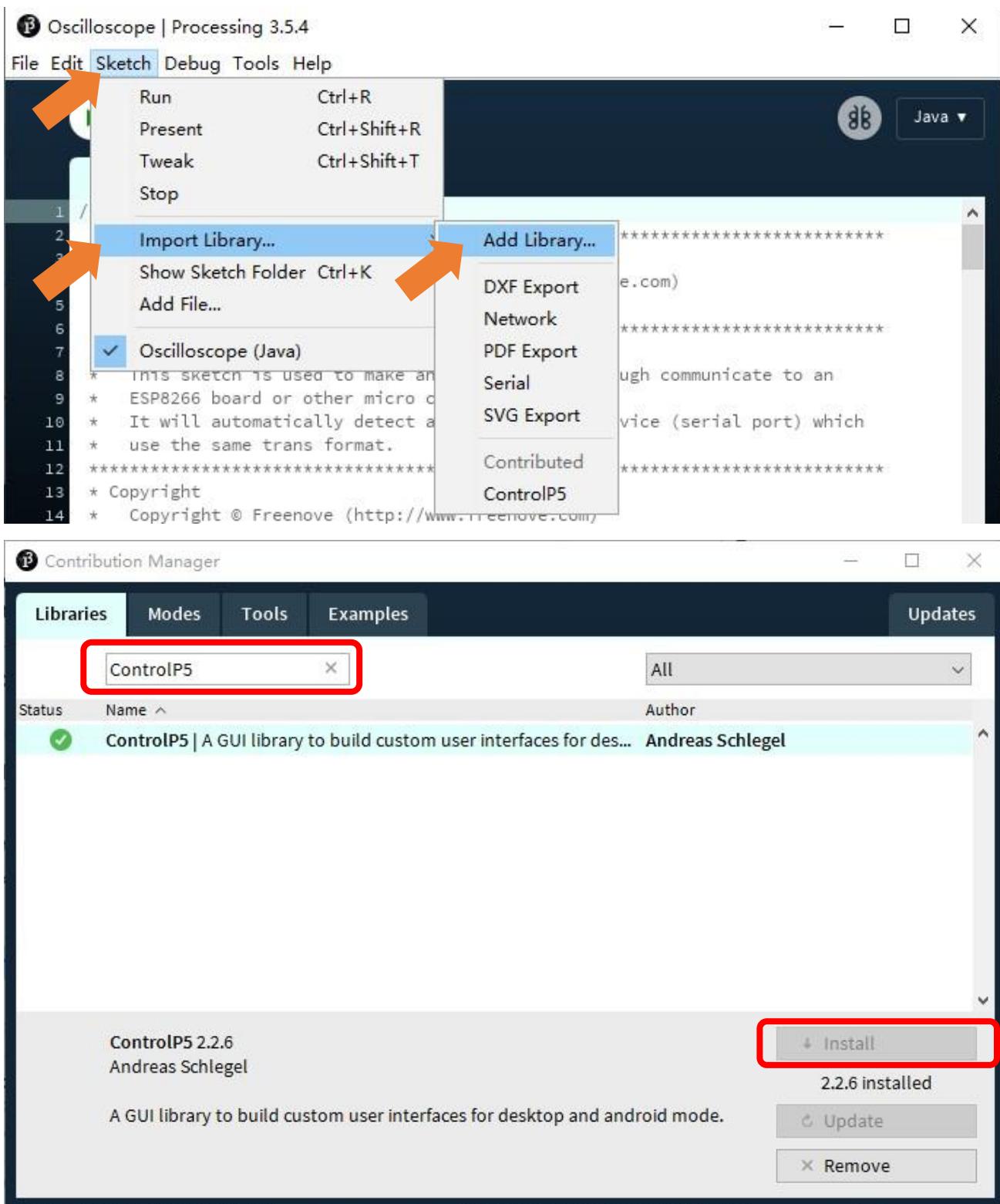


Select Libraries in the pop-up window.

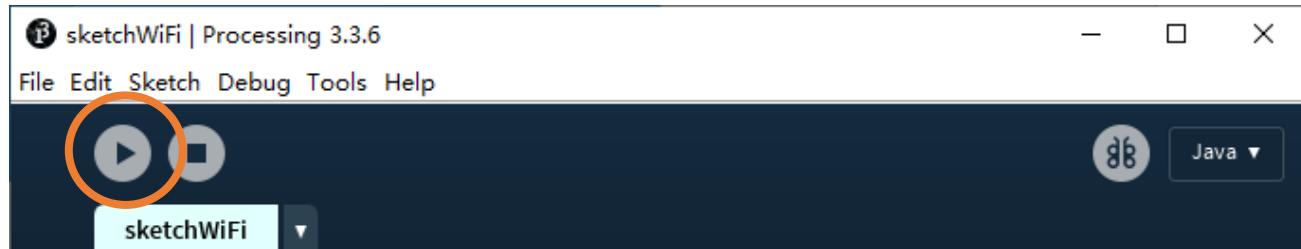


Input "ControlP5" in the searching box, and then select the option as below. Click "Install" and wait for the installation to finish.

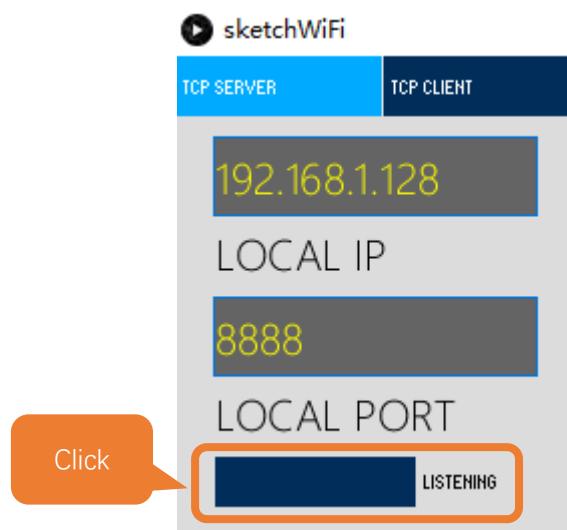
You can also click Add Library under 'Import Library' under 'Sketch'.



Before running the Code, 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. Click “Listening”.



Move the program folder “Freenove_ESP8266_Board/Python/Python_Codes” to disk(D) in advance with the path of “D:/Micropython_Codes”.

Open “Thonny”, click “This computer” → “D:” → “Micropython_Codes” → “29.1_TCP_as_Client” and double click “TCP_as_Client.py”.

Before clicking “Run current script”, please modify the name and password of your router and fill in the “host” and “port” according to the **IP information in processing app** shown in the box below:

29.1_TCP_as_Client

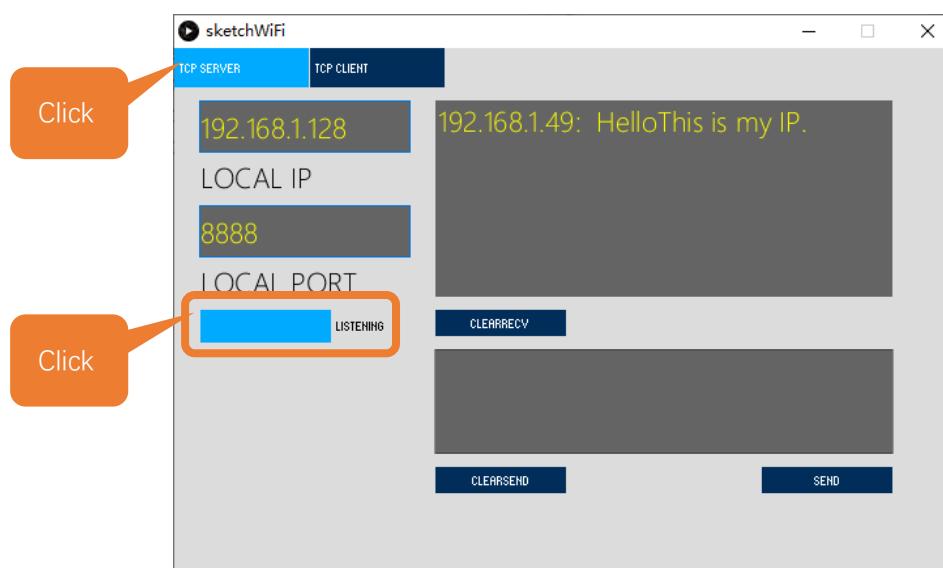
The screenshot shows the Thonny IDE interface with the following details:

- Title Bar:** Thonny - D:\Micropython_Codes\29.1_TCP_as_Client\TCP_as_Client.py @ 13 : 1
- Menu Bar:** File Edit View Run Device Tools Help
- Toolbar:** Includes icons for file operations like Open, Save, Run, and Stop.
- Left Sidebar (Files):** Shows "boot.py" under "MicroPython device".
- Central Editor Area (TCP_as_Client.py):** Contains Python code for a TCP client. A red box highlights the configuration section:

```
5 ssidRouter      = "*****"          #Enter the router name
6 passwordRouter = "*****"          #Enter the router password
7 host            = "192.168.1.128"    #input the remote server
8 port            = 8888             #input the remote port
```
- Bottom Shell Area:** Displays the MicroPython shell output:

```
MicroPython v1.18 on 2022-01-17; ESP module with ESP8266
Type "help()" for more information.
>>> %Run -c $EDITOR_CONTENT
#13 ets_task(4020f560, 28, 3ffff9448, 10)
```

Click "Run current script" and in "Shell", you can see ESP8266 automatically connects to sketchWiFi.



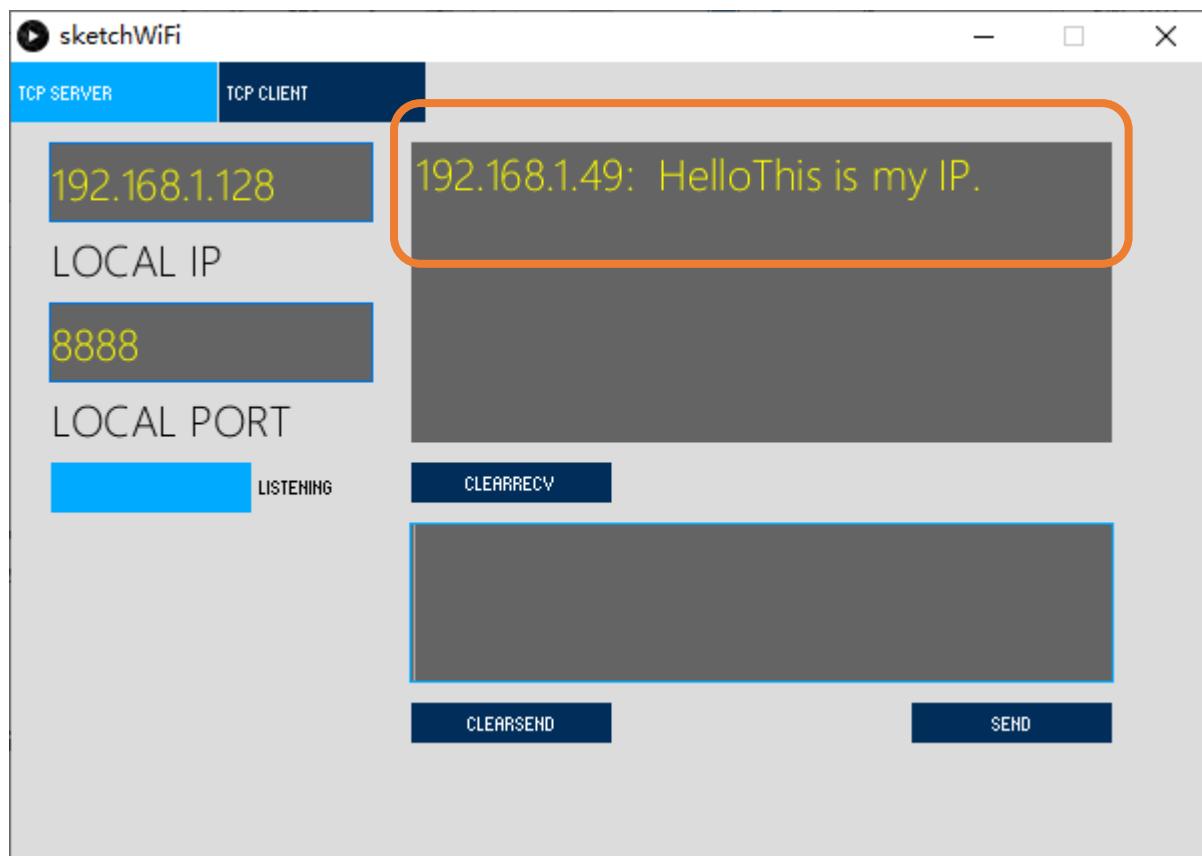
Any concerns? ✉ support@freenove.com

```
Shell <br/>MicroPython v1.18 on 2022-01-17; ESP module with ESP8266<br/>Type "help()" for more information.<br/>>> %Run -c $EDITOR_CONTENT<br/>#5 ets_task(4020f560, 28, 3ffff9ef0, 10)<br/>TCP Connected to: 192.168.1.128 : 8888
```

If you don't click "Listening" for sketchWiFi, ESP8266 will fail to connect and will print information as follows:

```
Shell <br/>TCP Connected to: 192.168.1.142 : 8888<br/>Close socket<br/>>> %Run -c $EDITOR_CONTENT<br/>TCP close, please reset!<br/>>>
```

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 import network<br/>2 import socket<br/>3 import time
```

```

4
5     ssidRouter      = "*****"      #Enter the router name
6     passwordRouter = "*****"      #Enter the router password
7     host           = "*****"      #input the remote server
8     port           = 8888         #input the remote port
9
10    wlan=None
11    s=None
12
13    def connectWifi(ssid,passwd):
14        global wlan
15        wlan= network.WLAN(network.STA_IF)
16        wlan.active(True)
17        wlan.disconnect()
18        wlan.connect(ssid,passwd)
19        while(wlan.ifconfig()[0]=='0.0.0.0'):
20            time.sleep(1)
21        return True
22
23    try:
24        connectWifi(ssidRouter,passwordRouter)
25        s = socket.socket()
26        s.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1)
27        s.connect((host,port))
28        print("TCP Connected to:", host, ":", port)
29        s.send('Hello')
30        s.send('This is my IP.')
31        while True:
32            data = s.recv(1024)
33            if(len(data) == 0):
34                print("Close socket")
35                s.close()
36                break
37            print(data)
38            ret=s.send(data)
39    except:
40        print("TCP close, please reset!")
41        if (s):
42            s.close()
43            wlan.disconnect()
44            wlan.active(False)

```

Import network、socket、time modules.

```

1 import network
2 import socket
3 import time

```

Any concerns? ✉ support@freenove.com

Enter the actual router name, password, remote server IP address, and port number.

```
5 ssidRouter      = "*****"      #Enter the router name  
6 passwordRouter = "*****"      #Enter the router password  
7 host           = "*****"      #input the remote server  
8 port           = 8888         #input the remote port
```

Connect specified Router until it is successful.

```

13 def connectWifi(ssid,passwd):
14     global wlan
15     wlan= network.WLAN(network.STA_IF)
16     wlan.active(True)
17     wlan.disconnect()
18     wlan.connect(ssid,passwd)
19     while(wlan.ifconfig()[0]=='0.0.0.0'):
20         time.sleep(1)
21     return True

```

Connect router and then connect it to remote server.

```

23 connectWifi(ssidRouter,passwordRouter)
24 s = socket.socket()
25 s.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1)
26 s.connect((host,port))
27 print("TCP Connected to:", host, ":", port)

```

Send messages to the remote server, receive the messages from it and print them out, and then send the messages back to the server.

```

28 s.send('Hello')
29 s.send('This is my IP.')
30 while True:
31     data = s.recv(1024)
32     if(len(data) == 0):
33         print("Close socket")
34         s.close()
35         break
36     print(data)
37     ret=s.send(data)

```

If an exception occurs in the program, for example, the remote server is shut down, execute the following program, turn off the socket function, and disconnect the WiFi.

```

39 print("TCP close, please reset!")
40 if (s):
41     s.close()
42     wlan.disconnect()
43     wlan.active(False)

```

Reference

Class socket

Before each use of **socket**, please add the statement “**import socket**” to the top of the python file.

socket([af, type, proto]): Create a socket.

af: address

socket.AF_INET: IPv4

socket.AF_INET6: IPv6

type: type

socket.SOCK_STREAM : TCP stream

socket.SOCK_DGRAM : UDP datagram

socket.SOCK_RAW : Original socket

socket.SO_REUSEADDR : socket reusable

proto: protocol number

socket.IPPROTO_TCP: TCPmode

socket.IPPROTO_UDP: UDPmode

socket.setsockopt(level, optname, value): Set the socket according to the options.

Level: Level of socket option

socket.SOL_SOCKET: Level of socket option. By default, it is 4095.

optname: Options of socket

socket.SO_REUSEADDR: Allowing a socket interface to be tied to an address that is already in use.

value: The value can be an integer or a bytes-like object representing a buffer.

socket.connect(address): To connect to server.

Address: Tuple or list of the server's address and port number

send(bytes): Send data and return the bytes sent.

recv(bufsize): Receive data and return a bytes object representing the data received.

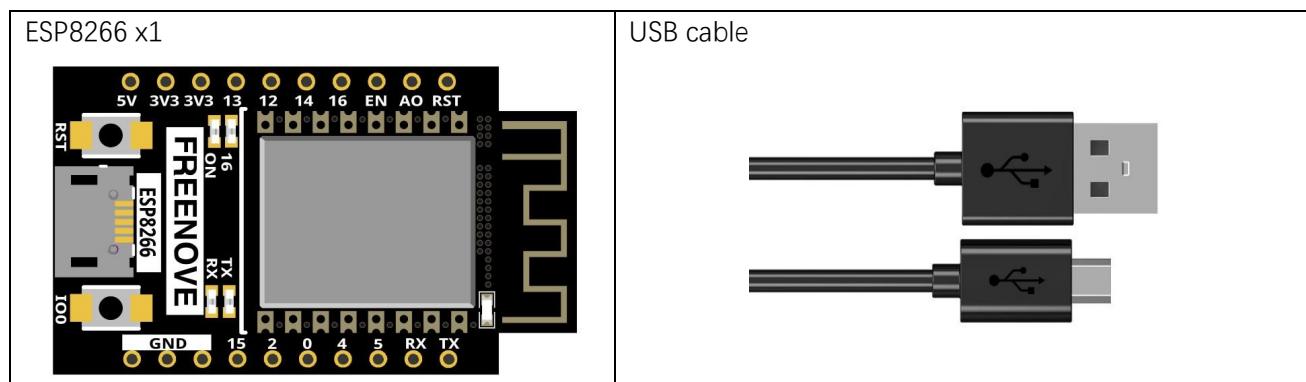
close(): Close socket.

To learn more please visit: <http://docs.micropython.org/en/latest/>

Project 4.2 As Server

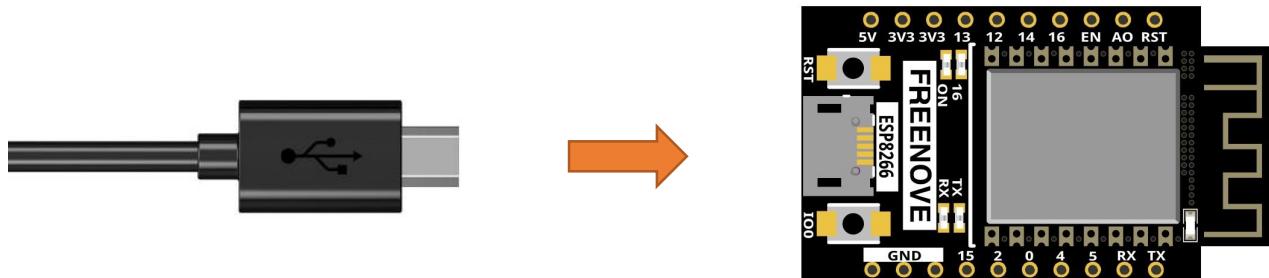
In this section, ESP8266 is used as a Server to wait for the connection and communication with Client on the same LAN.

Component List



Circuit

Connect Freenove ESP8266 to the computer using the USB cable.



Code

Move the program folder “Freenove_ESP8266_Board/Python/Python_Codes” to disk(D) in advance with the path of “D:/Micropython_Codes”.

Open “Thonny”, click “This computer” → “D:” → “Micropython_Codes” → “29.2_TCP_as_Server” and double click “TCP as Server.py”.

Before clicking “Run current script”, please modify the name and password of your router shown in the box below.

29.2 TCP as Server

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

- Title Bar:** Thonny - D:\Micropython_Codes\29.2_TCP_as_Server\TCP_as_Server.py @ 11 : 1
- Menu Bar:** File Edit View Run Device Tools Help
- Toolbar:** Includes icons for file operations like Open, Save, and Run.
- File Explorer:** Shows files in the project structure:
 - MicroPython device
 - boot.py
 - This computer
 - D:\ Micropython_Codes\29.2_TCP_as_Server
 - sketchWiFi
 - TCP_as_Server.py (selected)
- Code Editor:** Displays the contents of `TCP_as_Server.py`. A red box highlights the configuration section:

```
5 ssidRouter      = "*****"          #Enter the router name
6 passwordRouter = "*****"          #Enter the router password
7 port            = 8000             #input the remote port
8 wlan=None
9 listenSocket=None
```
- Shell:** Shows the Python prompt `>>>` and the MicroPython version information: `MicroPython v1.18 on 2022-01-17; ESP module with ESP8266`.

After making sure that the router's name and password are correct, click "Run current script" and in "Shell", you can see a server opened by the ESP8266 waiting to connecting to other network devices.

```
Shell x

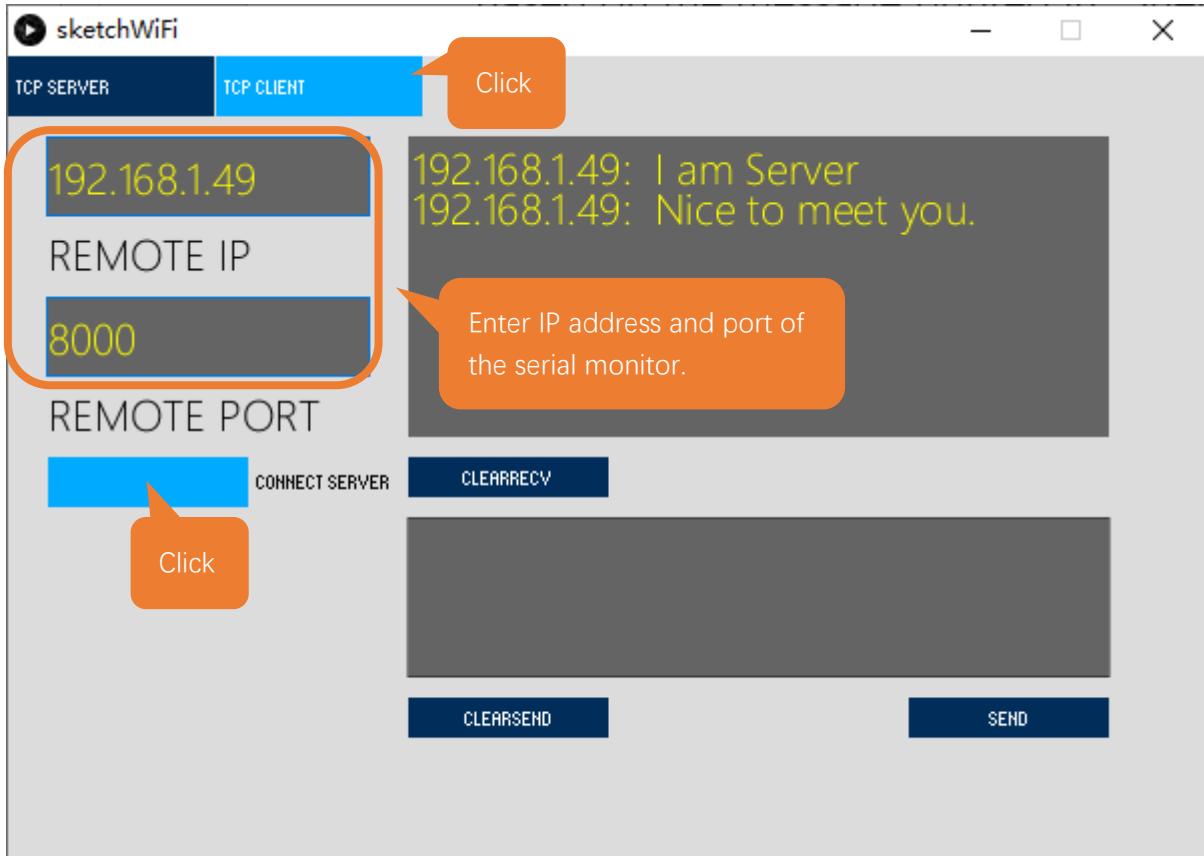
MicroPython v1.18 on 2022-01-17; ESP module with ESP8266
Type "help()" for more information.
>>> %Run -c $EDITOR_CONTENT
#12 ets_task(4020f560, 28, 3fff9448, 10)
tcp waiting...
Server IP: 192.168.1.49          Port: 8000
accepting.....
```



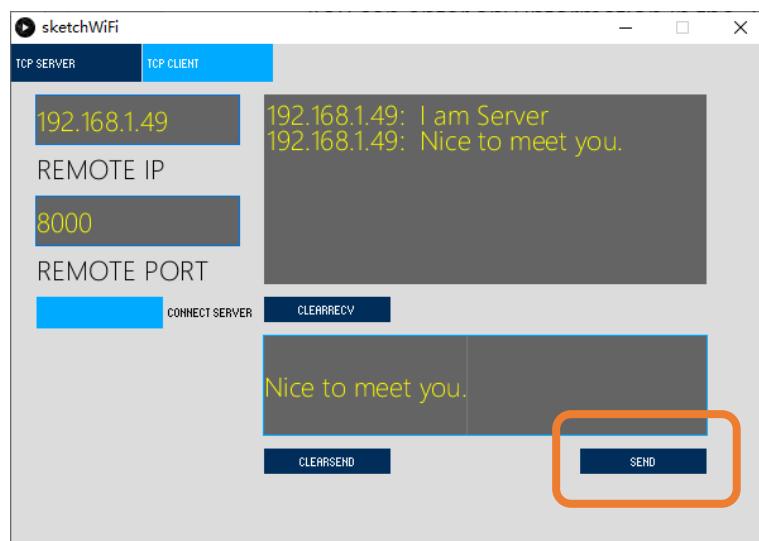
Processing:

Open the “**Freenove_ESP8266_Board/Codes/MicroPython_Codes/29.2_TCP_as_Server/sketchWiFi/sketchWiFi.pde**”.

Based on the message printed in "Shell", enter the correct IP address and port when processing, and click to establish a connection with ESP8266 to communicate.



You can enter any information in the “Send Box” of sketchWiFi. Click “Send” and ESP8266 will print the received messages to “Shell” and send them back to sketchWiFi.



```
Shell x

MicroPython v1.18 on 2022-01-17; ESP module with ESP8266
Type "help()" for more information.
>>> %Run -c $EDITOR_CONTENT
#12 ets_task(4020f560, 28, 3fff9448, 10)
tcp waiting...
Server IP: 192.168.1.49           Port: 8000
accepting.....
('192.168.1.128', 50312) connected
b'Nice to meet you.'
```

The following is the program code:

```
1 import network
2 import socket
3 import time
4
5 ssidRouter      = "*****"          #Enter the router name
6 passwordRouter = "*****"          #Enter the router password
7 port           = 8000             #input the remote port
8 wlan            = None
9 listenSocket    = None
10
11 def connectWifi(ssid,passwd):
12     global wlan
13     wlan=network.WLAN(network.STA_IF)
14     wlan.active(True)
15     wlan.disconnect()
16     wlan.connect(ssid,passwd)
17     while(wlan.ifconfig()[0]=='0.0.0.0'):
18         time.sleep(1)
19     return True
20
21 try:
22     connectWifi(ssidRouter,passwordRouter)
23     ip=wlan.ifconfig()[0]
24     listenSocket = socket.socket()
25     listenSocket.bind((ip,port))
26     listenSocket.listen(1)
27     listenSocket.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1)
28     print('tcp waiting...')
29     while True:
30         print("Server IP:",ip,"\tPort:",port)
31         print("accepting.....")
32         conn,addr = listenSocket.accept()
33         print(addr, "connected")
34         break
35     conn.send('I am Server')
36     while True:
37         data = conn.recv(1024)
38         if(len(data) == 0):
39             print("close socket")
40             listenSocket.close()
41             wlan.disconnect()
42             wlan.active(False)
43             break
```

```

44     else:
45         print(data)
46         ret = conn.send(data)
47     except:
48         print("Close TCP-Server, please reset.")
49         if(listenSocket):
50             listenSocket.close()
51             wlan.disconnect()
52             wlan.active(False)

```

Call function `connectWifi()` to connect to router and obtain the dynamic IP that it assigns to ESP8266.

```

22     connectWifi(ssidRouter, passwordRouter)
23     ip=wlan.ifconfig()[0]

```

Open the socket server, bind the server to the dynamic IP, and open a data monitoring port.

```

24     listenSocket = socket.socket()
25     listenSocket.bind((ip, port))
26     listenSocket.listen(1)
27     listenSocket.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1)

```

Print the server's IP address and port, monitor the port and wait for the connection of other network devices.

```

29     while True:
30         print("Server IP:", ip, "\tPort:", port)
31         print("accepting.....")
32         conn, addr = listenSocket.accept()
33         print(addr, "connected")
34         break

```

Each time receiving data, print them in "Shell" and send them back to the client.

```

36     while True:
37         data = conn.recv(1024)
38         if(len(data) == 0):
39             print("close socket")
40             listenSocket.close()
41             wlan.disconnect()
42             wlan.active(False)
43             break
44         else:
45             print(data)
46             ret = conn.send(data)

```

If the client is disconnected, close the server and disconnect WiFi.

```

47     except:
48         print("Close TCP-Server, please reset.")
49         if(listenSocket):
50             listenSocket.close()
51             wlan.disconnect()
52             wlan.active(False)

```

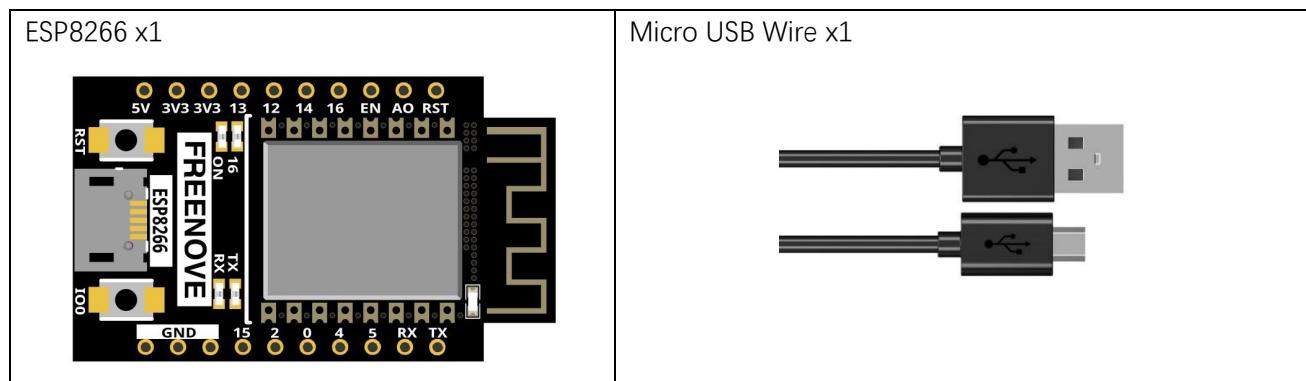
Chapter 5 Smart Home

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

Project 5.1 Control_LED_through_Web

In this project, we need to build a Web Service and then use ESP8266 to control the LED through the Web browser of the 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>: Note the title of the document

<body>: Contains visible page content

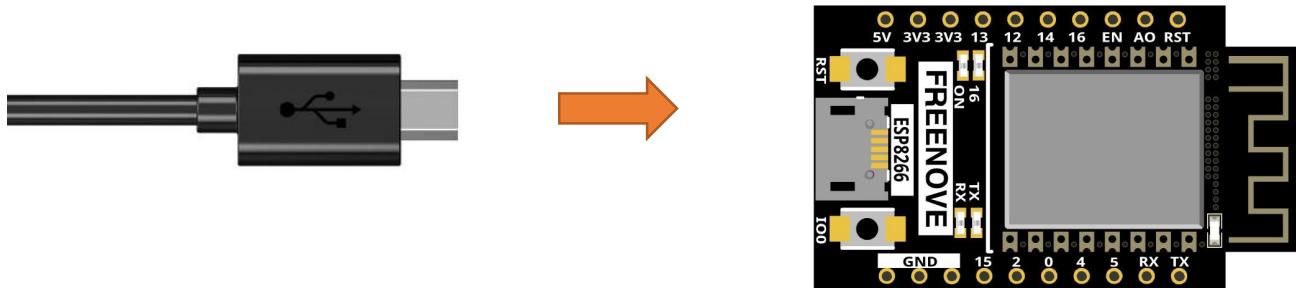
<h1>: Define a big heading

<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.



Code

Move the program folder “Freenove_ESP8266_Board/Python/Python_Codes” to disk(D) in advance with the path of “D:/Micropython_Codes”.

Open “Thonny”, click “This computer” → “D:” → “Micropython_Codes” → “30.1_Control_LED_through_Web”. and double click “Control_LED_through_Web”.

30.1_Control_LED_through_Web

```

from machine import Pin
import time
import socket
import network

# set led pin
led = Pin(2, Pin.OUT)

ssid = '*****'          #Enter the router name
password = '*****'      #Enter the router password

wifi_status = network.WLAN(network.STA_IF)
wifi_status.active(True)
wifi_status.connect(ssid, password)

# check wifi connected
while wifi_status.isconnected() == False:
    print('Wifi lost connect...')

# if connected
print('Wifi connect successful')
print(wifi_status.ifconfig())

def WebPage():
    if led.value() == 1:
        gpio_state = 'OFF'
    else:
        gpio_state = 'ON'

    # html code ...
    html = """<html><head> <title>ESP8266 Web Server</title> <meta name="viewport" content="wid<link rel="icon" href="data:;"/> <style>html{font-family: Helvetica; display:inline-block; margin:0; padding: 0;}</style><body><h1>ESP8266 Web Server</h1><button value="ON">ON</button><button value="OFF">OFF</button></body></html>"""
    print(html)

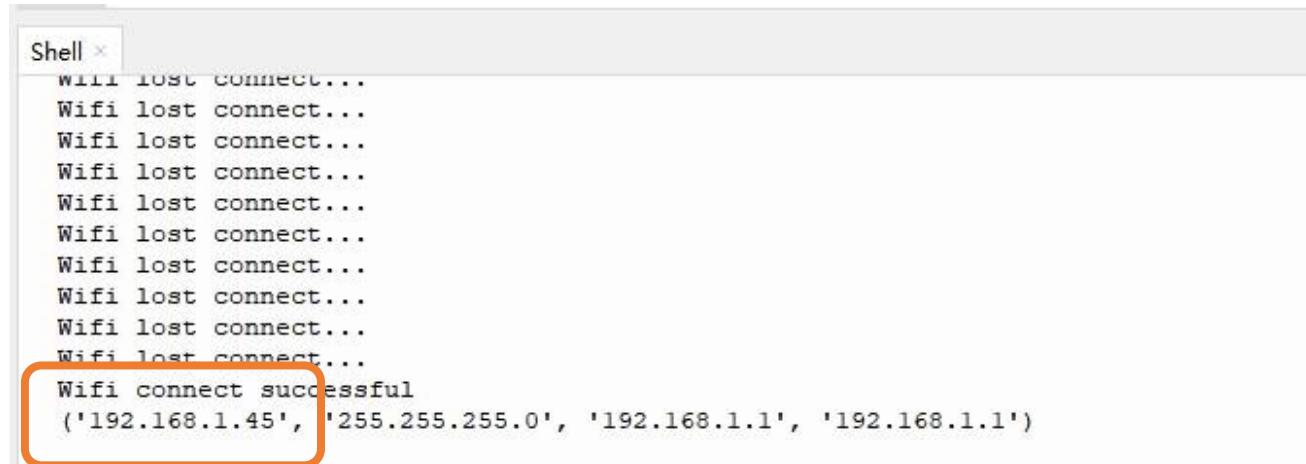
MicroPython v1.18 on 2022-01-17; ESP module with ESP8266
Type "help()" for more information.
>>>

```

Enter the correct Router name and password.

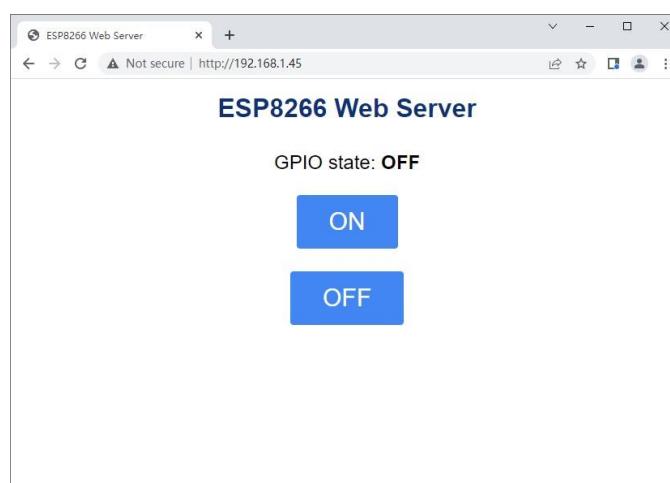
Because the names and passwords of routers in various places are different, before the Code runs, users need to enter the correct router's name and password in the box as shown in the illustration above.

After making sure the router name and password are entered correctly, compile and upload codes to ESP8266, wait for ESP8266 to connect to your router and print the IP address assigned by the router to ESP8266 in "Shell".



```
Shell
Wifi lost connect...
Wifi connect successful
('192.168.1.45', '255.255.255.0', '192.168.1.1', '192.168.1.1')
```

When ESP8266 successfully connects to "ssid", "Shell" displays the IP address assigned to ESP8266 by the router. Access <http://192.168.1.45> 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 from machine import Pin
2 import time
3 import socket
4 import network
5
6 # set led pin
7 led = Pin(2, Pin.OUT)
8
9 ssid = '*****'          #Enter the router name
10 password = '*****'      #Enter the router password
11
```

```
12 wifi_status = network.WLAN(network.STA_IF)
13 wifi_status.active(True)
14 wifi_status.connect(ssid, password)
15 # check wifi connected
16 while wifi_status.isconnected() == False:
17     print('Wifi lost connect...')
18 # if connected
19 print('Wifi connect successful')
20 print(wifi_status.ifconfig())
21
22 def WebPage():
23     if led.value() == 1:
24         gpio_state = 'OFF'
25     else:
26         gpio_state = 'ON'
27
28     # html code ...
29     html = """
30     <html>
31         <head>
32             <title>ESP8266 Web Server</title>
33             <meta name="viewport" content="width=device-width, initial-scale=1">
34             <link rel="icon" href="data:,">
35             <style>
36                 html{font-family: Helvetica; display:inline-block; margin: 0px auto; text-align: center;}
37                     h1{color: #0F3376; padding: 2vh;}
38                     p{font-size: 1.5rem;}
39                     button{display: inline-block; background-color: #4286f4; border: none; border-radius: 4px; color: white; padding: 16px 40px; text-decoration: none; font-size: 30px; margin: 2px; cursor: pointer;}
40                     button2{background-color: #4286f4;}
41             </style>
42         </head>
43         <body> <h1>ESP8266 Web Server</h1>
44             <p>GPIO state: <strong>"""+gpio_state+"""</strong></p>
45             <p><a href="/?led=on"><button class="button">ON</button></a></p>
46             <p><a href="/?led=off"><button class="button button2">OFF</button></a></p>
47         </body>
48     </html>
49     """
50     return html
51
52 s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
```

```

53     s.bind(('', 80))
54     s.listen(5)
55     try:
56         while True:
57             conn, addr = s.accept()
58             print('Connection: %s' % str(addr))
59             req = conn.recv(1024)
60             req = str(req)
61             print('Connect = %s' % req)
62             led_on = req.find('/?led=on')
63             led_off = req.find('/?led=off')
64             if led_on == 6:
65                 print(' LED ON')
66                 led.value(0)
67             else:
68                 print(' LED OFF')
69                 led.value(1)
70             if led.value() == 1:
71                 gpio_state = 'OFF'
72             else:
73                 gpio_state = 'ON'
74             response = WebPage()
75             conn.send('HTTP/1.1 200 OK\n')
76             conn.send('Content-Type: text/html\n')
77             conn.send('Connection: close\n\n')
78             conn.sendall(response)
79             conn.close()
80     except:
81         pass

```

Import socket module and Import network module.

```

3 import socket
4 import network

```

Enter correct AP name and password.

```

3 ssid = '*****'          #Enter the router name
4 password = '*****'      #Enter the router password

```

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

```

12 wifi_status = network.WLAN(network.STA_IF)
13 wifi_status.active(True)
14 wifi_status.connect(ssid, password)

```

"Shell" displays the IP address assigned to ESP8266.

```

20 print(wifi_status.ifconfig())

```

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

```

55     if led_on == 6:
56         print(' LED ON')

```

```
57     led.value(0)
58 else:
59     print(' LED OFF')
60     led.value(1)
61 if led.value() == 1:
62     gpio_state = ' OFF'
63 else:
64     gpio_state = ' ON'
```

What's next?

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

support@freenove.com

We will check and correct it as soon as possible.

If you want learn more about ESP8266, you view our ultimate tutorial:

https://github.com/Freenove/Freenove_ESP8266_Board/archive/master.zip

If you want to learn more about Arduino, Raspberry Pi, smart cars, robots and other interesting products in science and technology, please continue to focus on our website. We will continue to launch cost-effective, innovative and exciting products.

<http://www.freenove.com/>

End of the Tutorial

Thank you again for choosing Freenove products.