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.

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®, ESP32®, 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

Any concerns? ✉ support@freenove.com

Contents

Prepare

ESP32 is a micro control unit with integrated Wi-Fi launched by Espressif, which features strong properties and integrates rich peripherals. It can be designed and studied as an ordinary Single Chip Micyoco(SCM) chip, or connected to the Internet and used as an Internet of Things device.

ESP32 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 ESP32 is completely open-source, so beginners can quickly learn how to develop and design IOT smart household products including smart curtains, fans, lamps and clocks.

We divide each project into four parts, namely Component List, Component Knowledge, Circuit and Code. Component List helps you to prepare material for the experiment more quickly. Component Knowledge allows you to quickly understand new electronic modules or components, while Circuit helps you understand the operating principle of the circuit. And Code allows you to easily master the use of ESP32 and 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

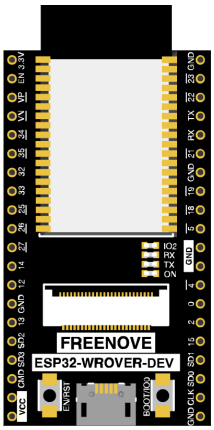
ESP32-WROVER

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

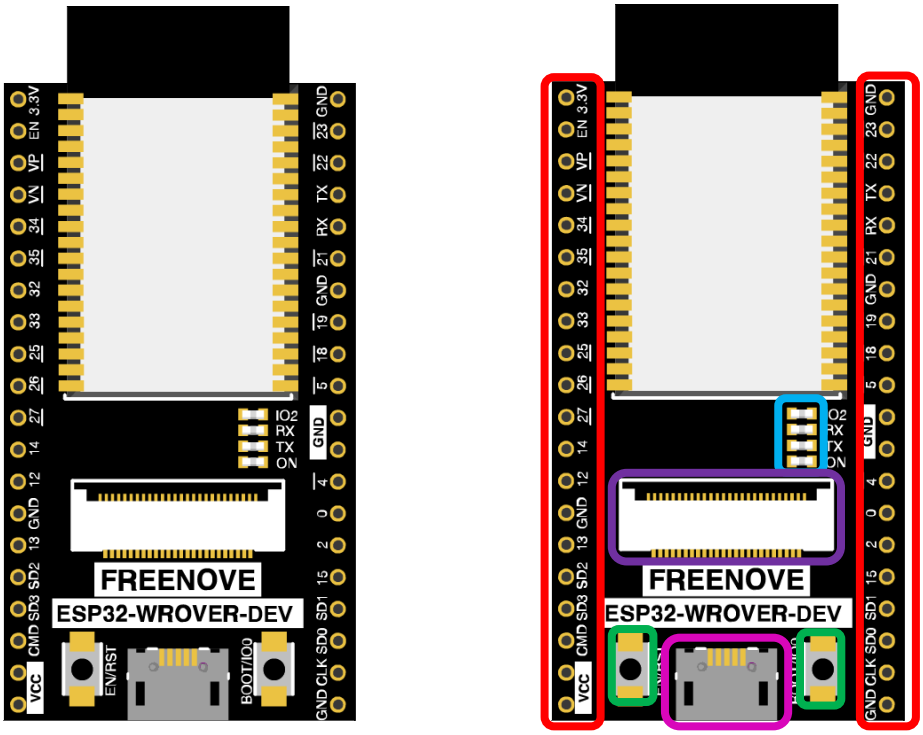
PCB on-board antenna	IPEX antenna
	

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

ESP32-WROVER



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



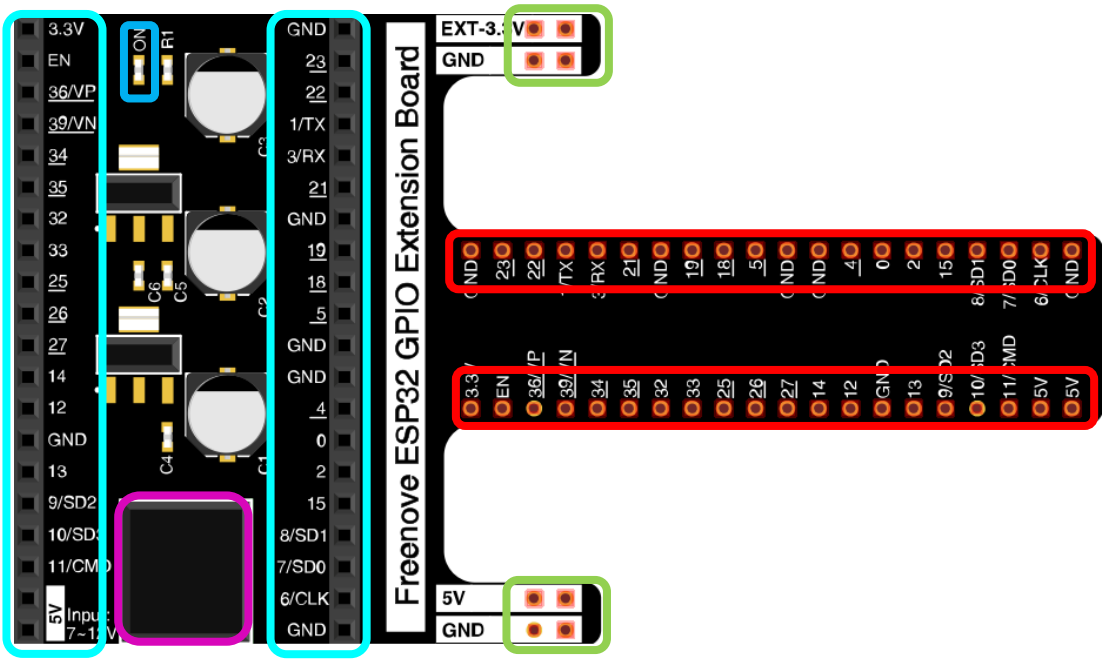
Compare the left and right images. We've boxed off the resources on the ESP32-WROVER in different colors to facilitate your understanding of the ESP32-WROVER.

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

Extension board of the ESP32-WROVER

And we also design an extension board, so that you can use the ESP32 more easily in accordance with the circuit diagram provided. The followings are their photos. All the projects in this tutorial are studied with this ESP32-WROVER.

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



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

Box color	Corresponding resources introduction
	GPIO pin
	LED indicator
	GPIO interface of development board
	Power supplied by the extension board
	External power supply

In ESP32, GPIO is an interface to control peripheral circuit. For beginners, it is necessary to learn the functions of each GPIO. The following is an introduction to the GPIO resources of the ESP32-WROVER development board.

Later, we only use USB cable to power ESP32-WROVER in default.

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

We can also use DC jack of extension board to power ESP32-WROVER. Then 5v and EXT 3.3v on extension board are from external power resource.

For more information, please visit:

https://www.espressif.com/sites/default/files/documentation/esp32-wrover_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 ESP32 during the whole process.
Thonny supports various operating system, including Windows、Mac OS、Linux.

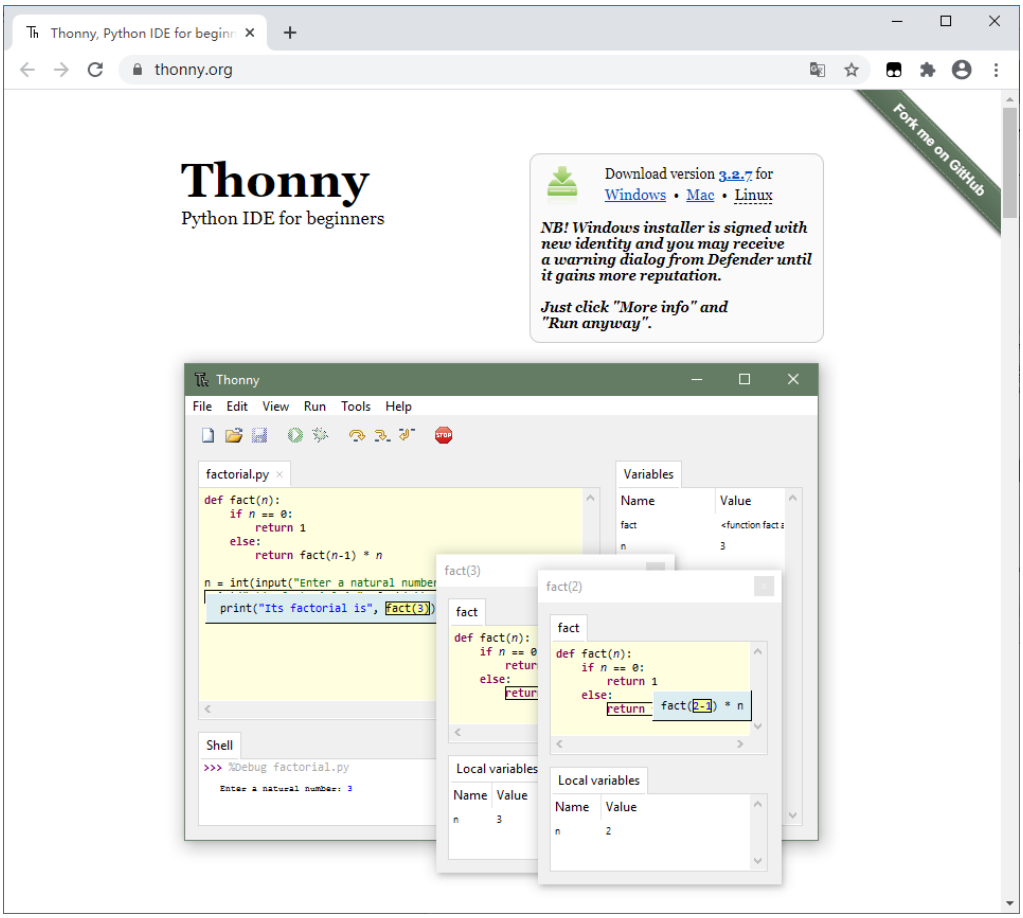
Downloading Thonny

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, Rasbian, Ubuntu, Mint and others:** sudo apt install thonny **Fedora:** sudo dnf install thonny

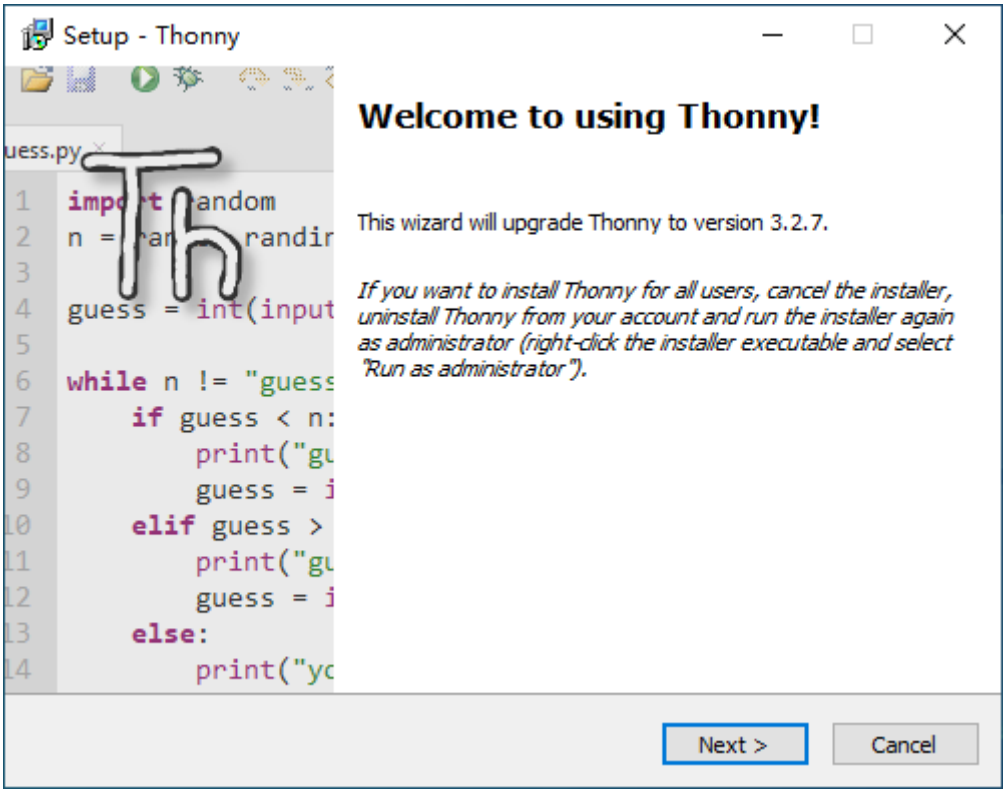
You can also open
"**Freenove_LCD_Module/Freenove_LCD_Module_for_ESP32/Python/Python_Software**",
we have prepared it in advance.



Installing on Windows

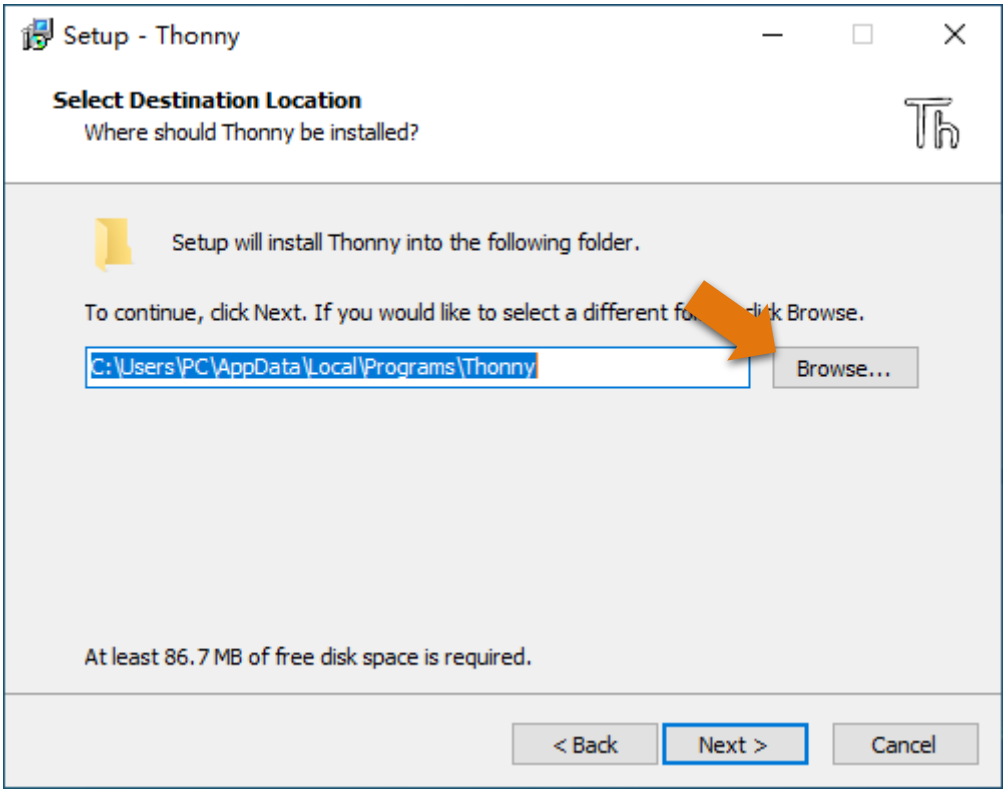
The icon of Thonny after downloading is as below. Double click “thonny-3.2.7.exe”.

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

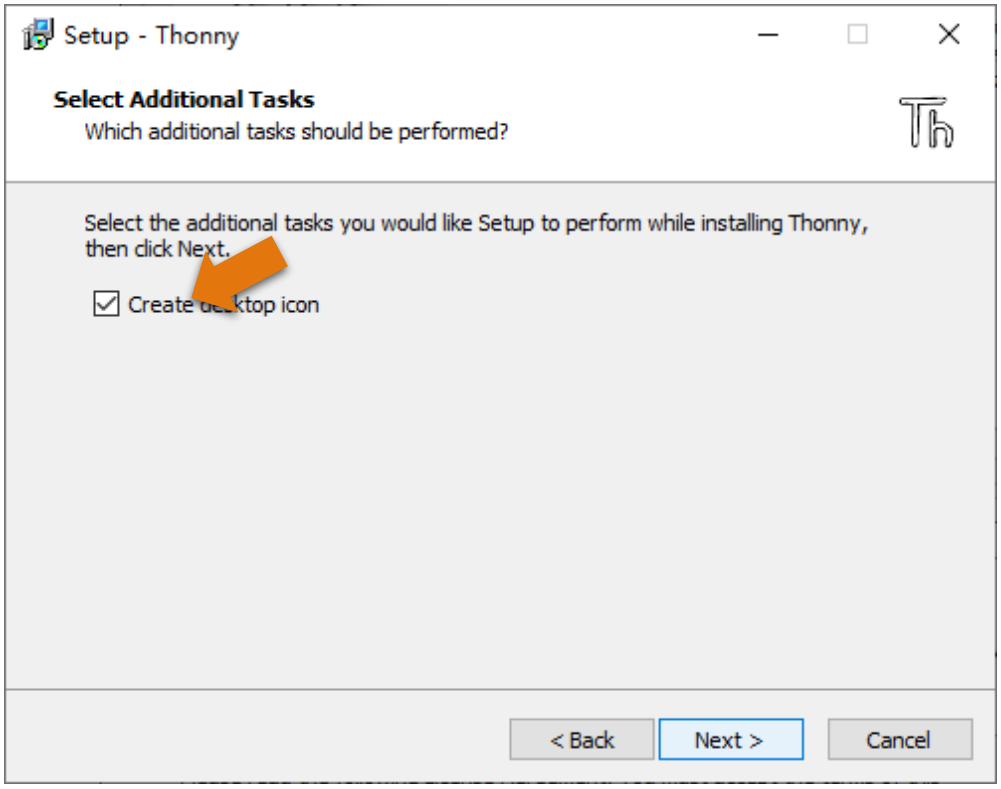


If you want to change Thonny's installation path, you can click "Browse" to modify it. After selecting installation path, click "OK".

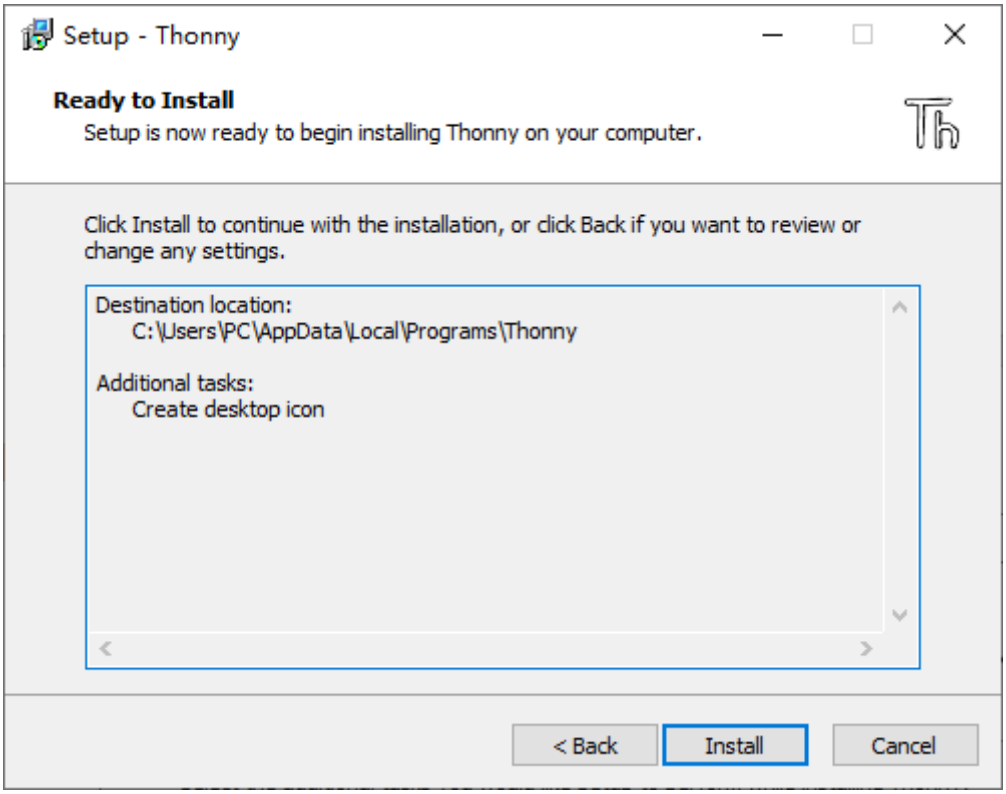
If you do not want to change it, just click "Next".



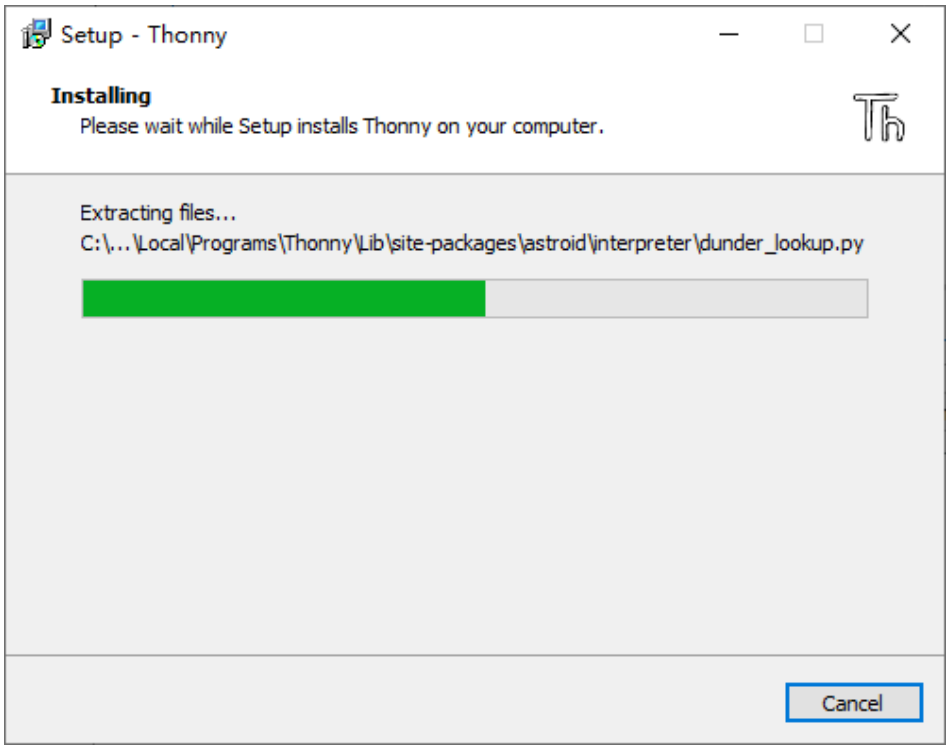
Check "Create desktop icon" and then it will generate a shortcut on your desktop to facilitate you to open Thonny later.



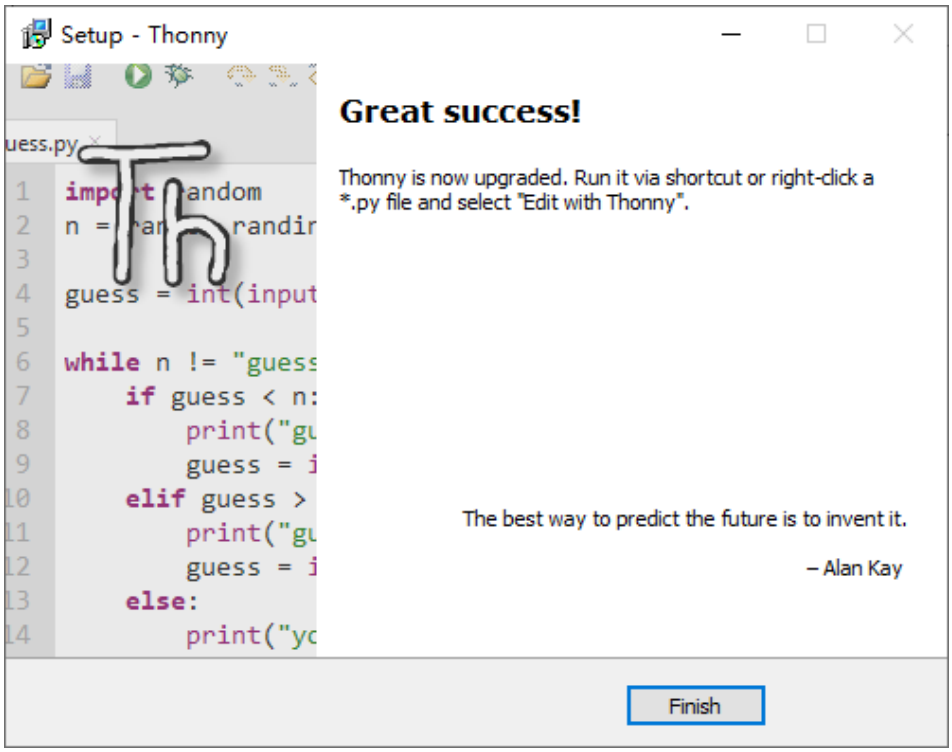
Click "install" to install the software.



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



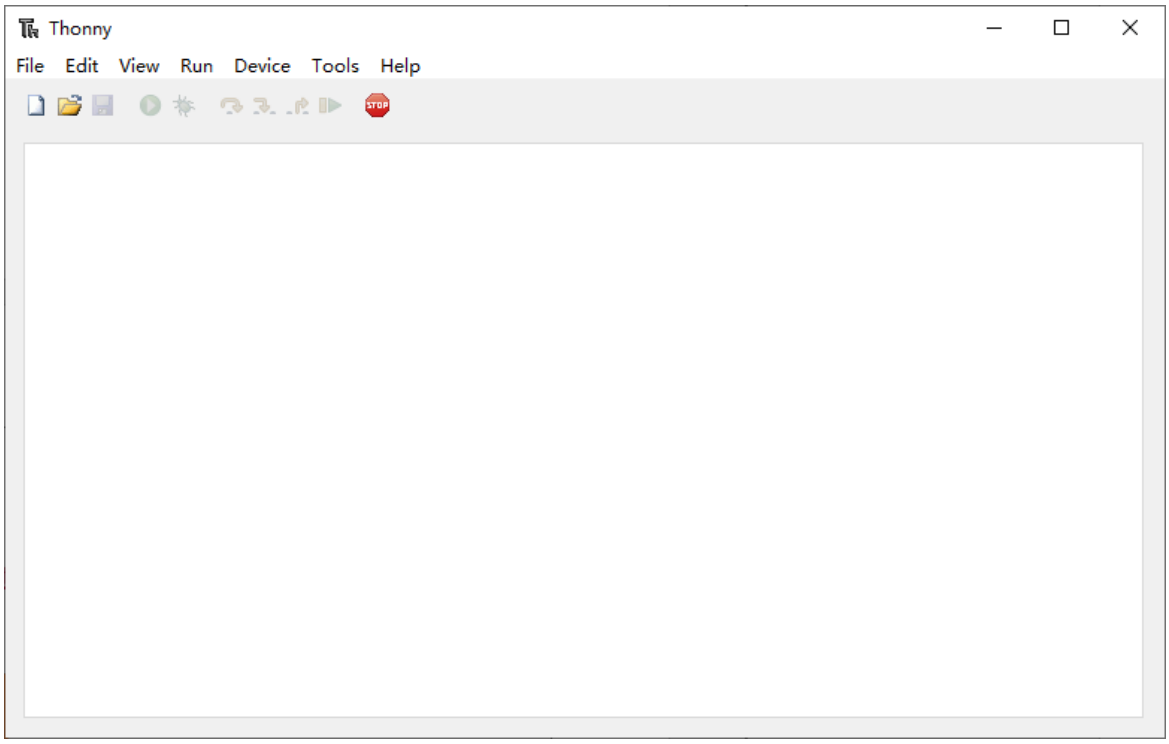
Once you see the interface as below, Thonny has been installed successfully.



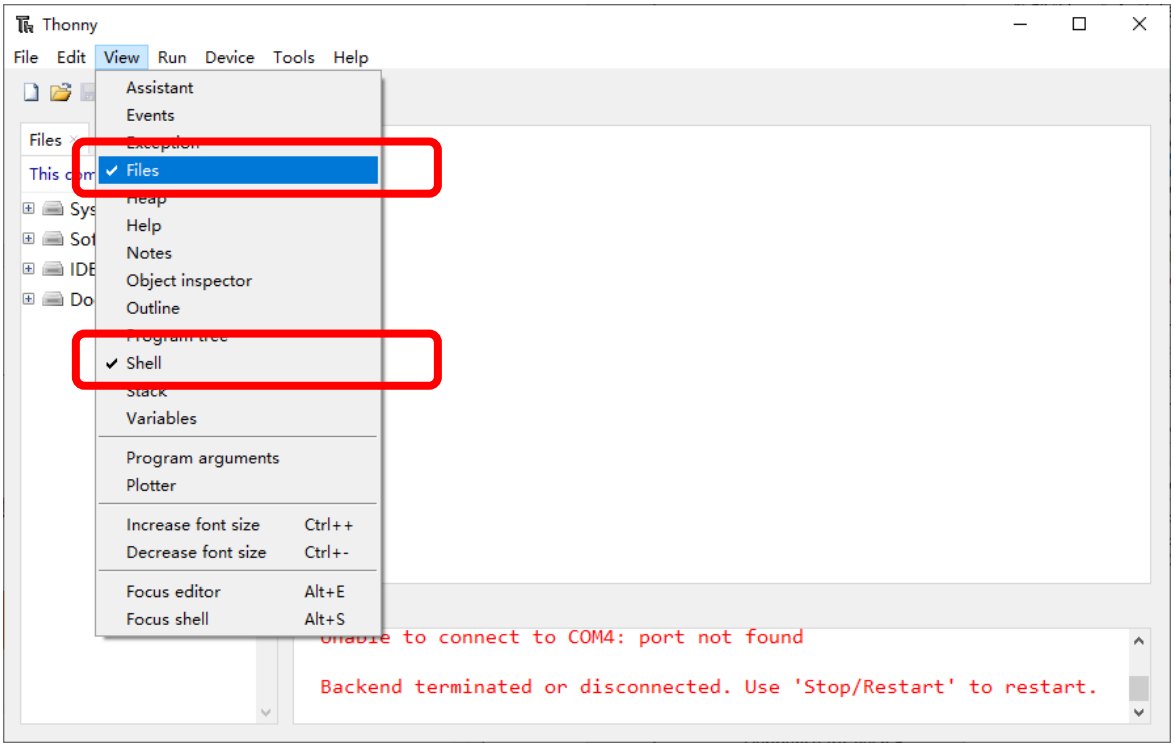
If you've check "Create desktop icon" during the installation process, you can see the below icon on your desktop.。

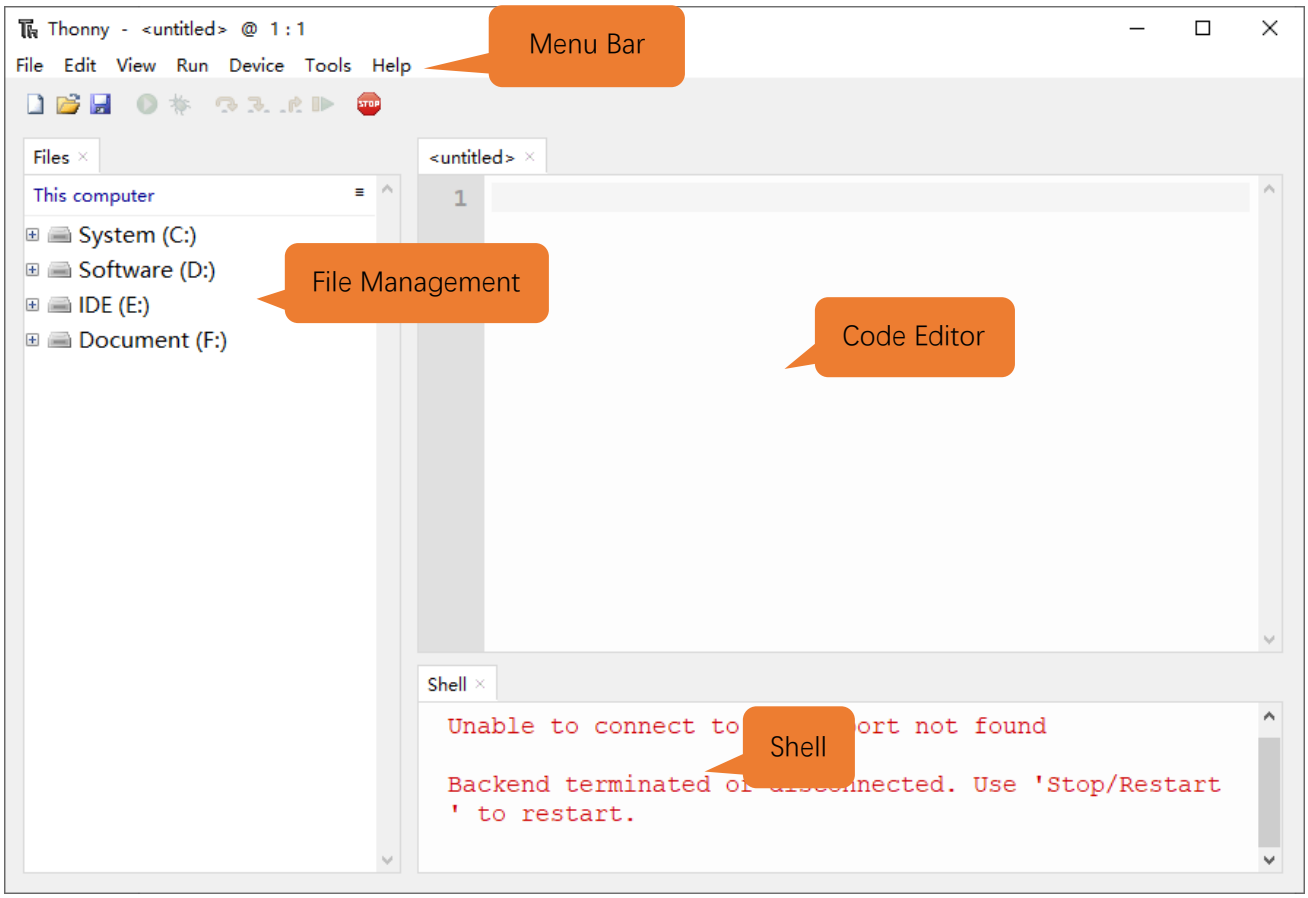
0.2 Basic Configuration of Thonny

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



Select "View"→ "Files" and "Shell".




Thonny - <untitled> @ 1 : 1
File Edit View Run Device Tools Help
Menu Bar
Files
This computer
System (C:)
Software (D:)
IDE (E:)
Document (F:)
File Management
<untitled>
1
Code Editor
Shell
Unable to connect to
ort not found
Backend terminated or disconnected. Use 'Stop/Restart
' to restart.
Shell

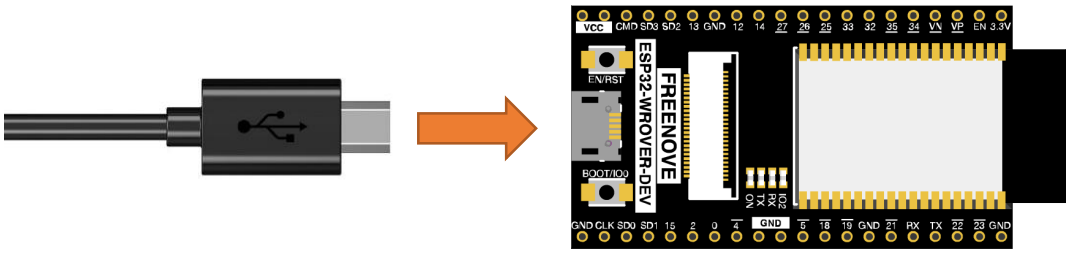
0.3 Installing CH340 (Important)

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

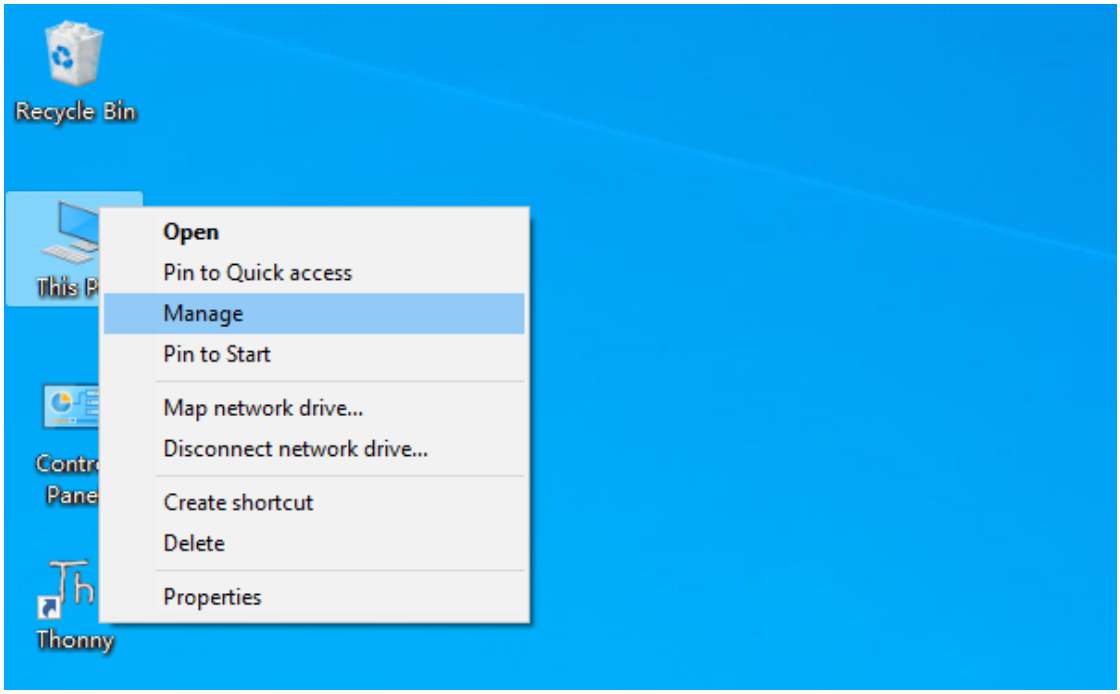
Windows

Check whether CH340 has been installed

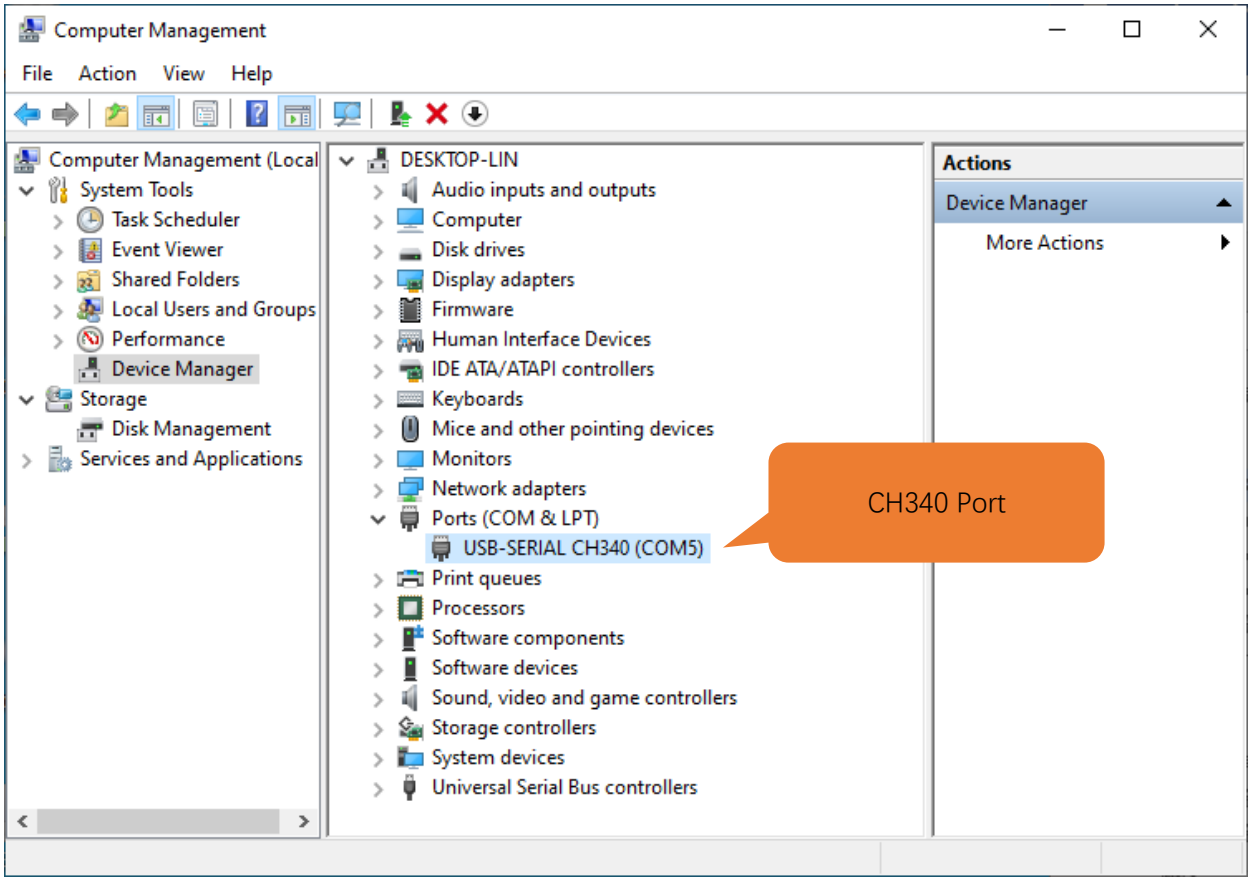
1. Connect your computer and ESP32 with a USB cable.



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

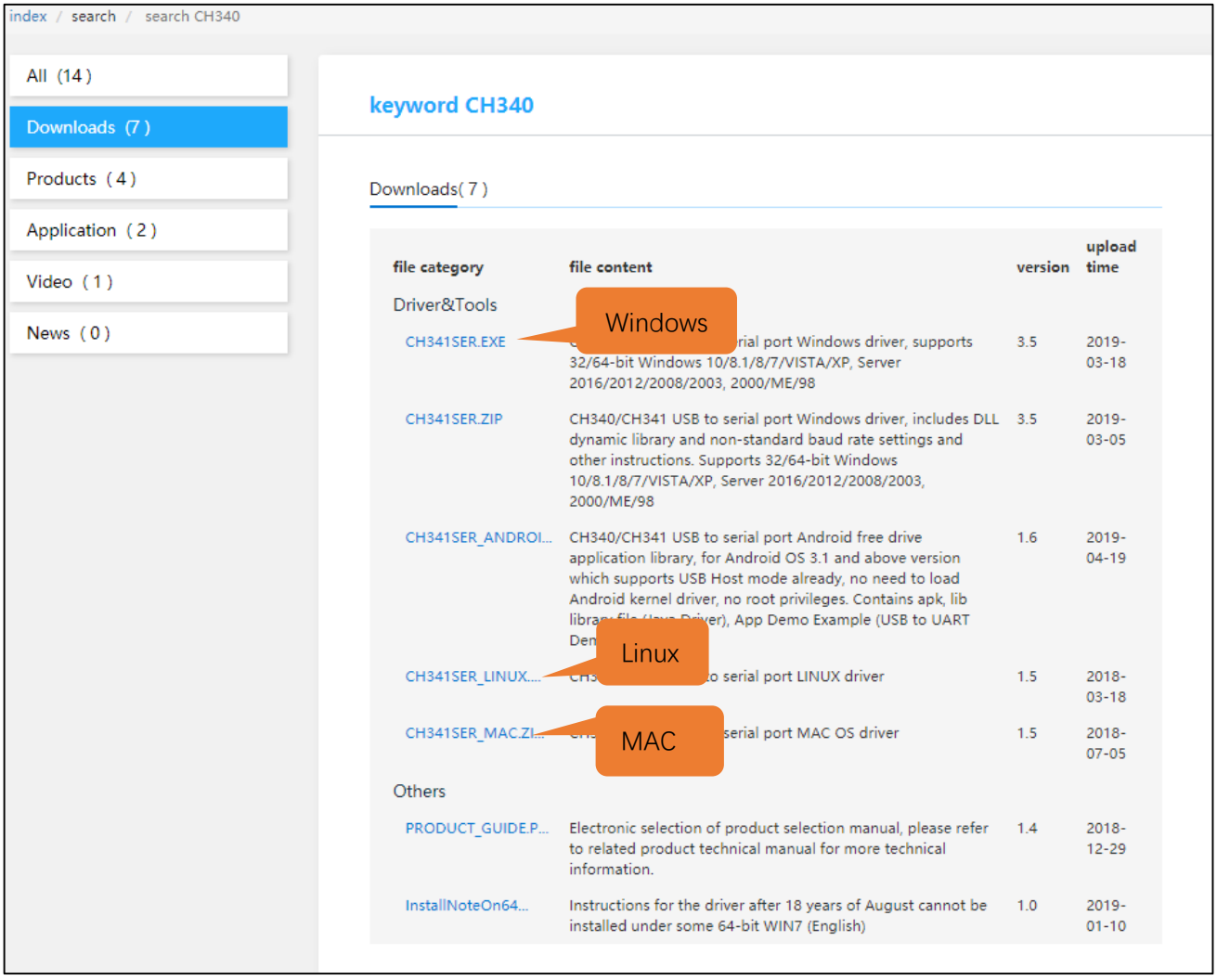


3. Click "Device Manager". If your computer has installed CH340, you can see"USB-SERIAL CH340 (COMx)". And you can click here to move to the next step.

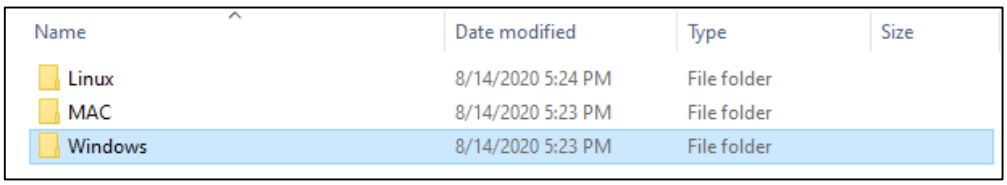


Installing CH340

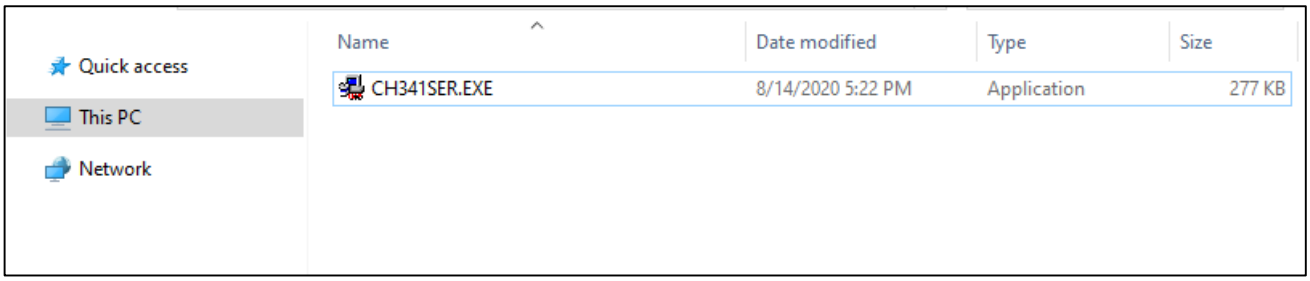
1. First, download CH340 driver, click http://www.wch-ic.com/search?q=CH340&t=downloads to download the appropriate one based on your operating system.



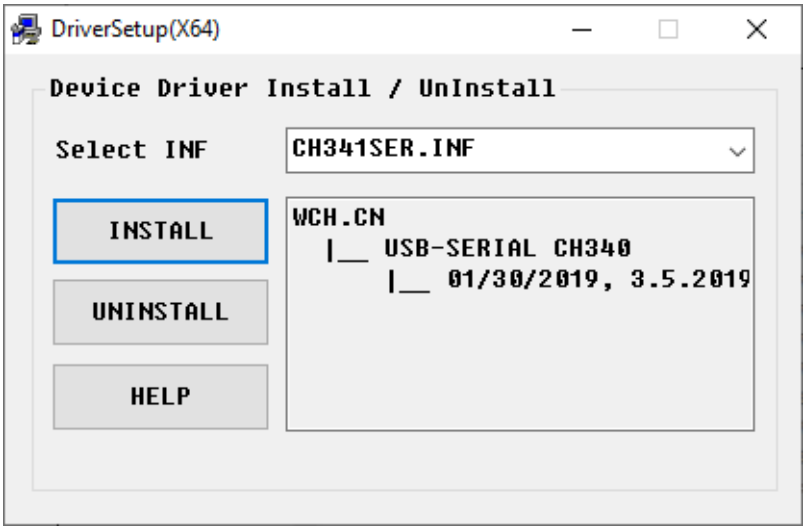
You can also open **"Freenove_LCD_Module/CH340"**, we have prepared the installation package.



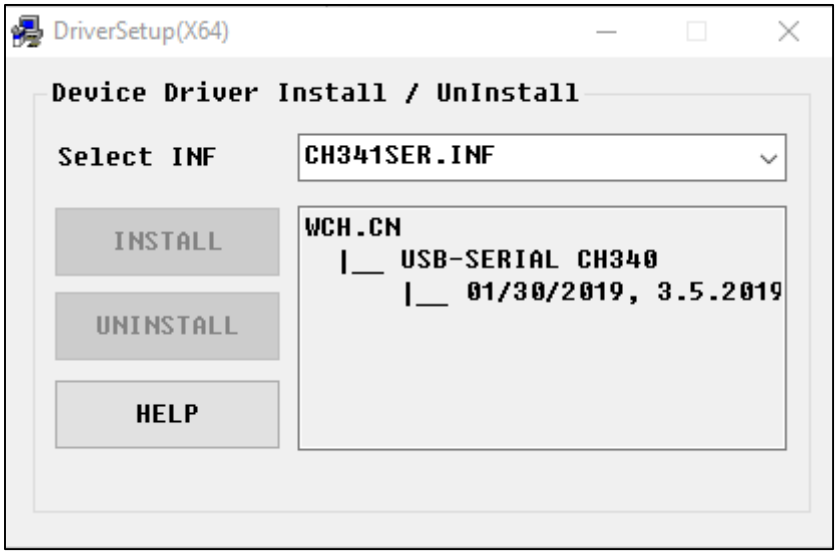
2. Open the folder "**Freenove_LCD_Module/CH340/Windows/ch341ser**"



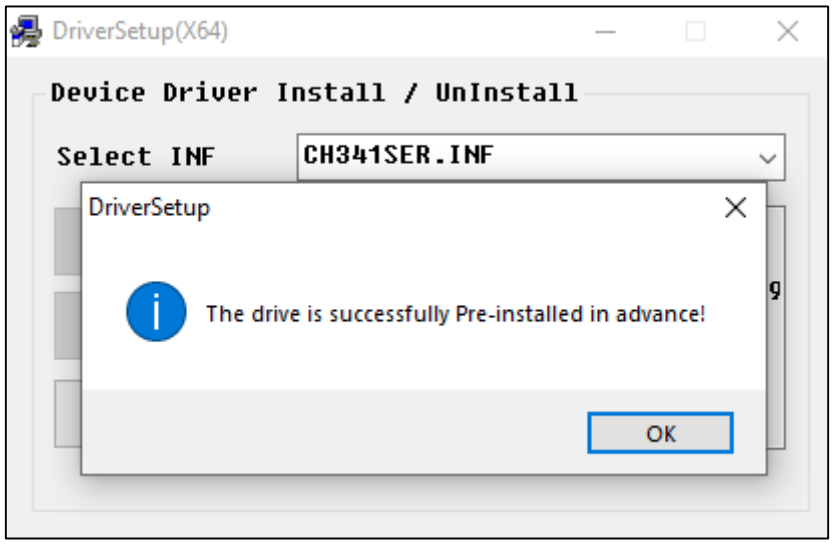
3. Double click "CH341SER.EXE".



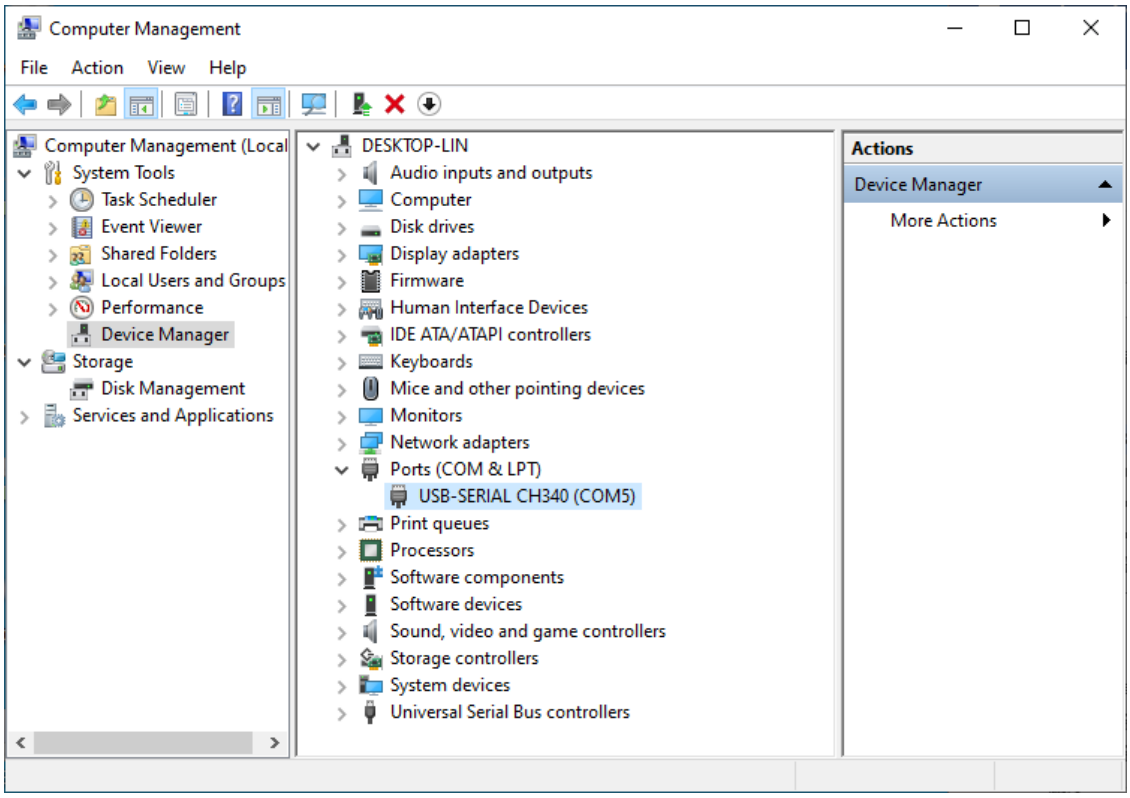
4. Click “INSTALL” and wait for the installation to complete.



5. Install successfully. Close all interfaces.



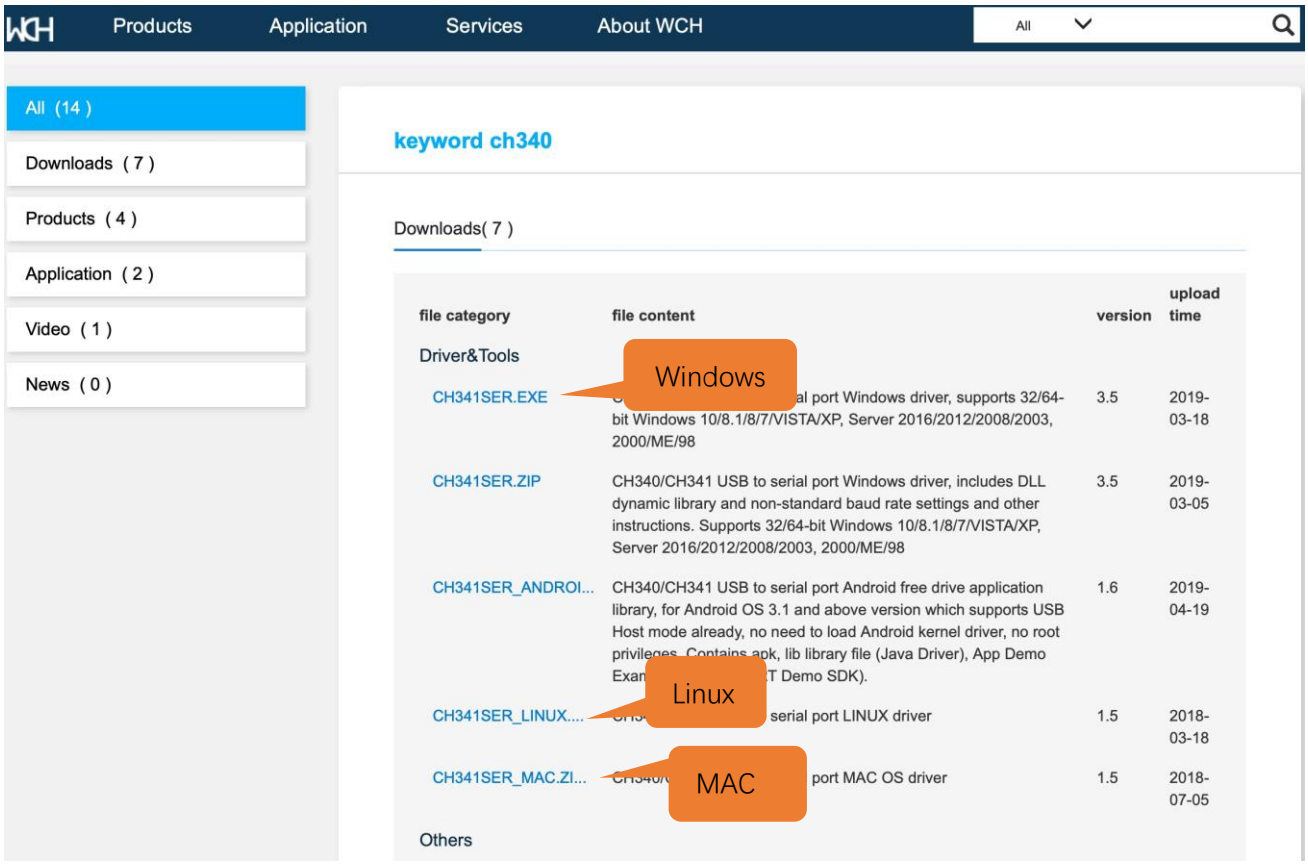
6. When ESP32 is connected to computer, select “This PC”, right-click to select “Manage” and click “Device Manager” in the newly pop-up dialog box, and you can see the following interface.



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

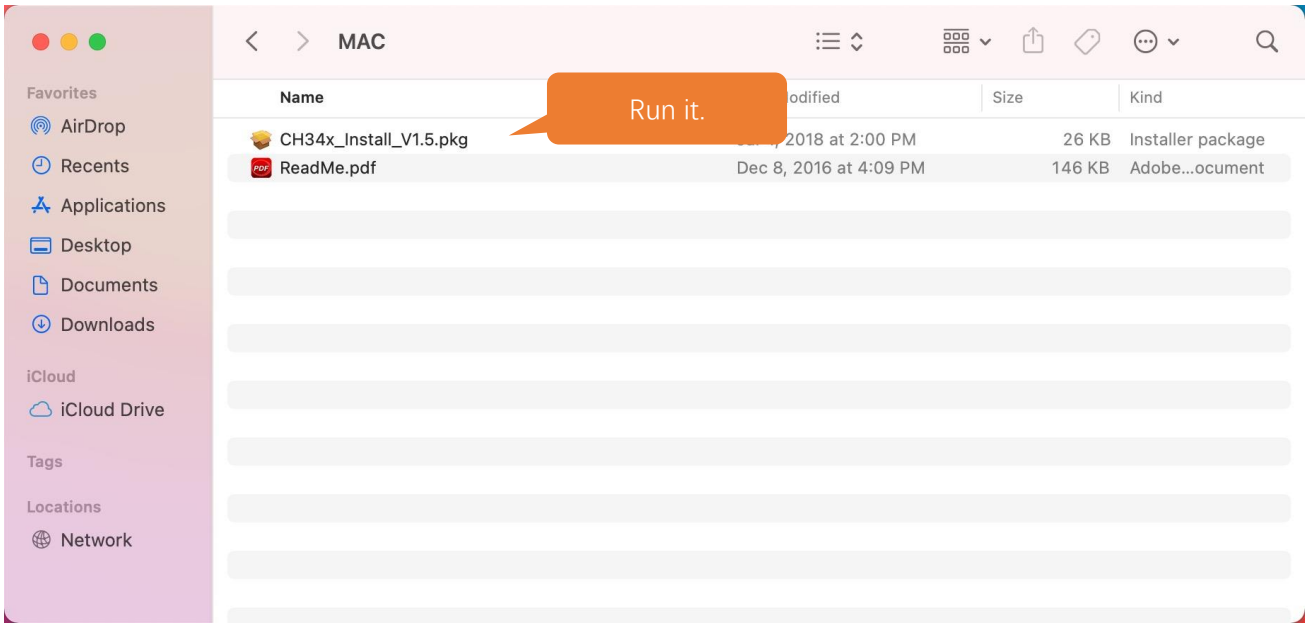
MAC

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

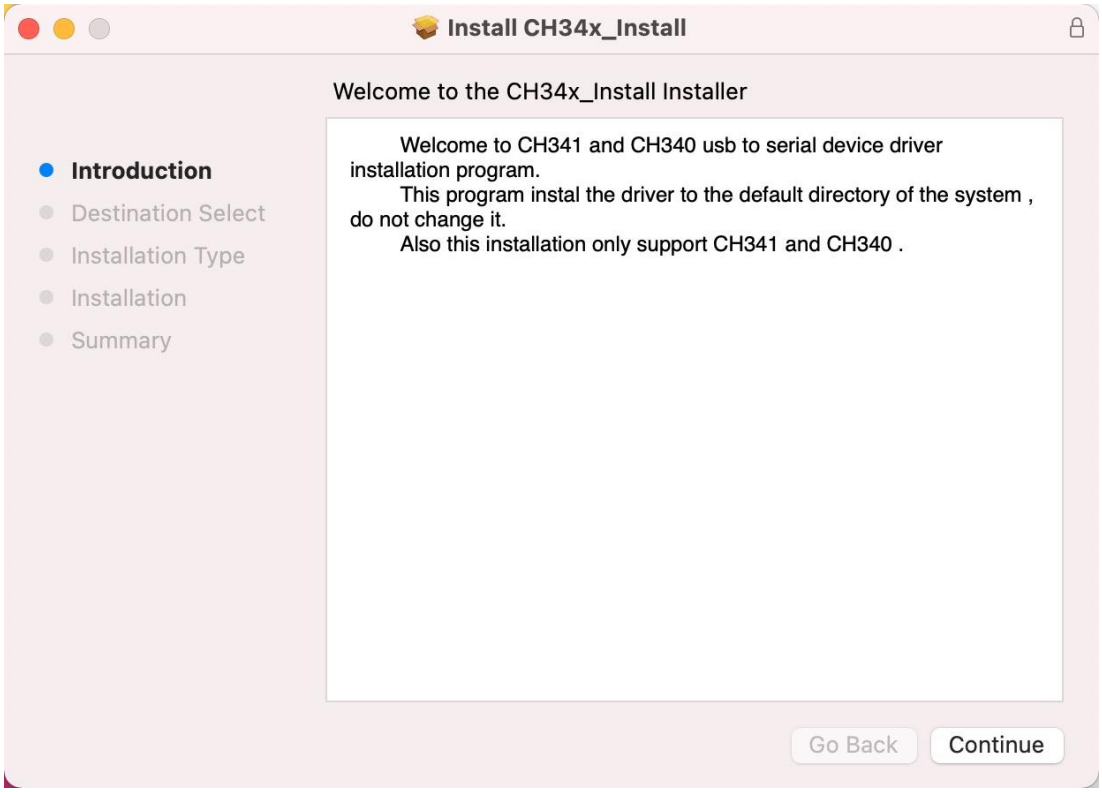


If you would not like to download the installation package, you can open **"Freenove_LCD_Module/CH340"**, we have prepared the installation package.

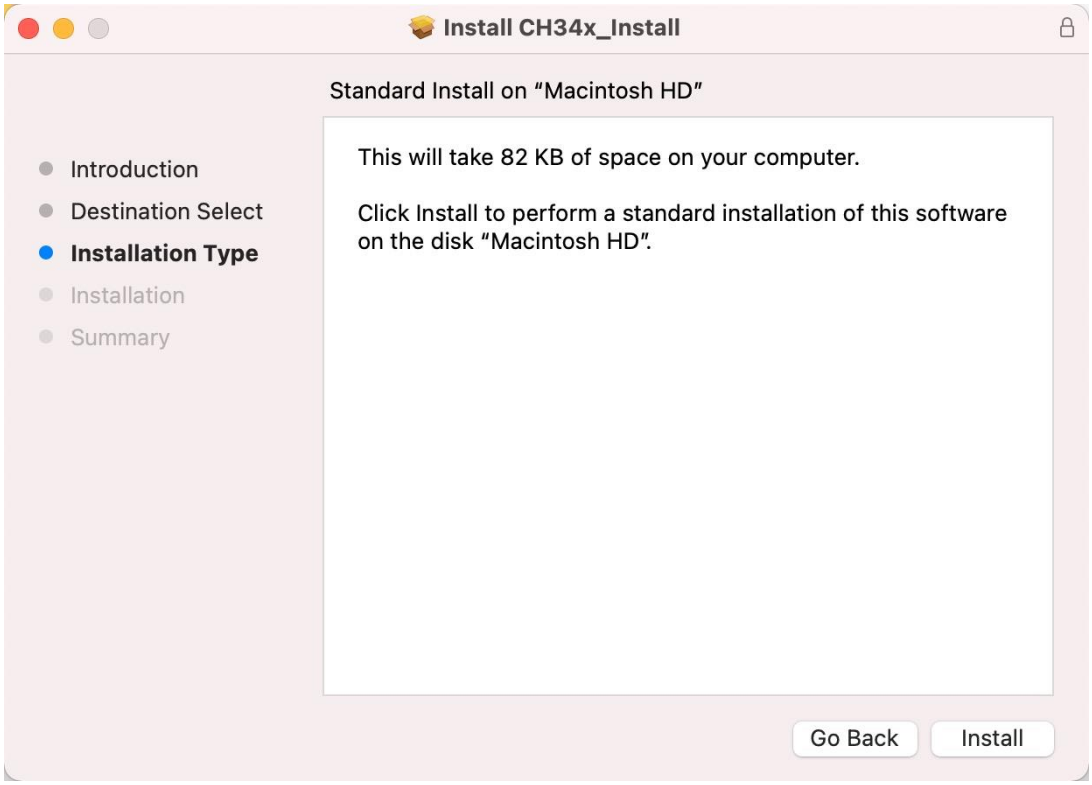
Second, open the folder "**Freenove_LCD_Module/CH340/MAC/**"



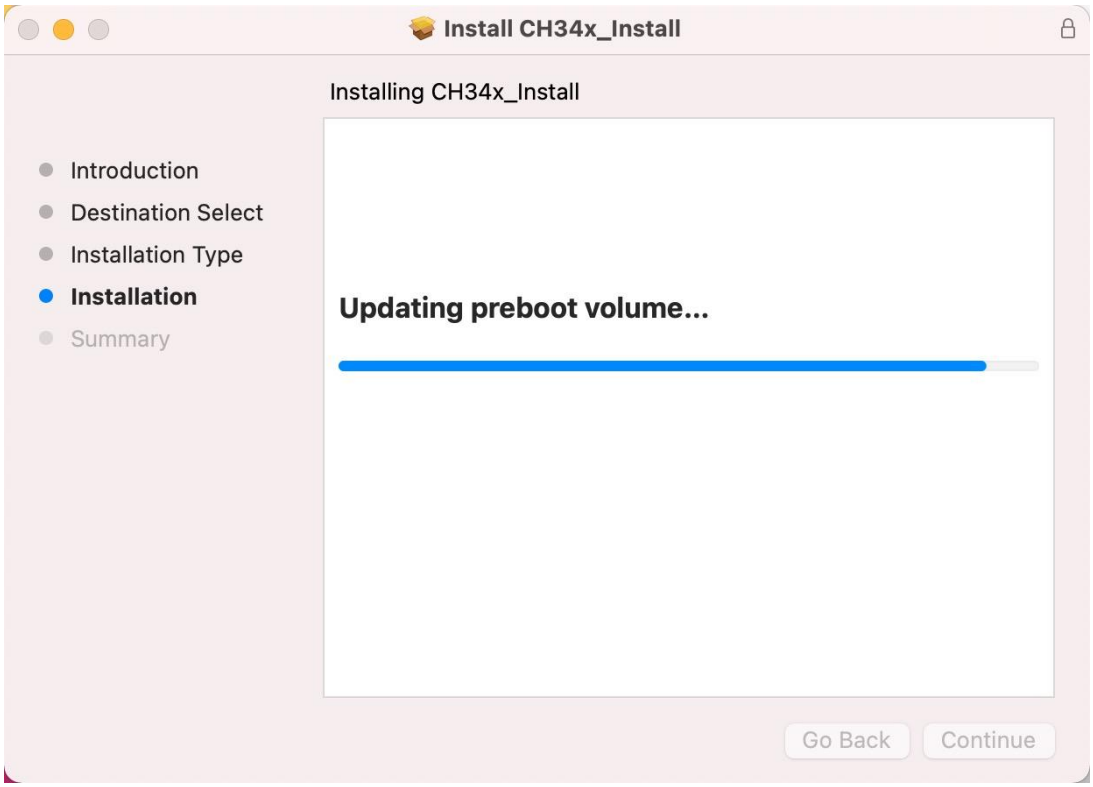
Third, click Continue.



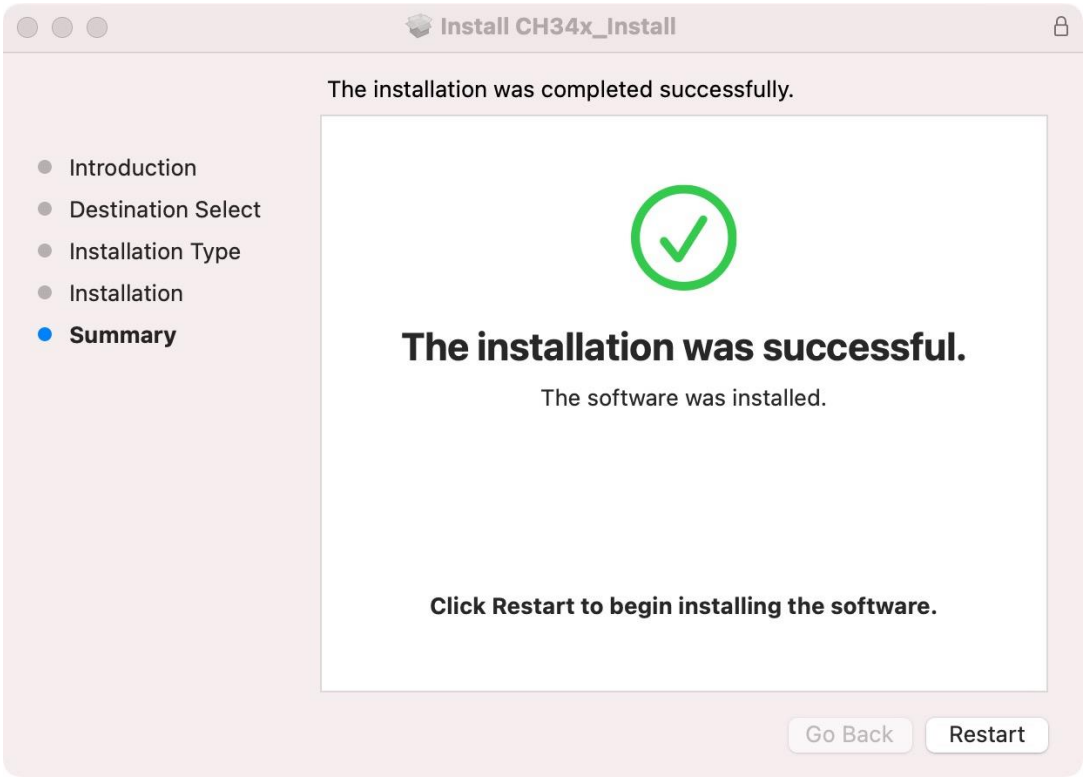
Fourth, click Install.


Install CH34x_Install
Standard Install on "Macintosh HD"
Introduction
Destination Select
Installation Type
Installation
Summary
This will take 82 KB of space on your computer.
Click Install to perform a standard installation of this software on the disk "Macintosh HD".
Go Back
Install

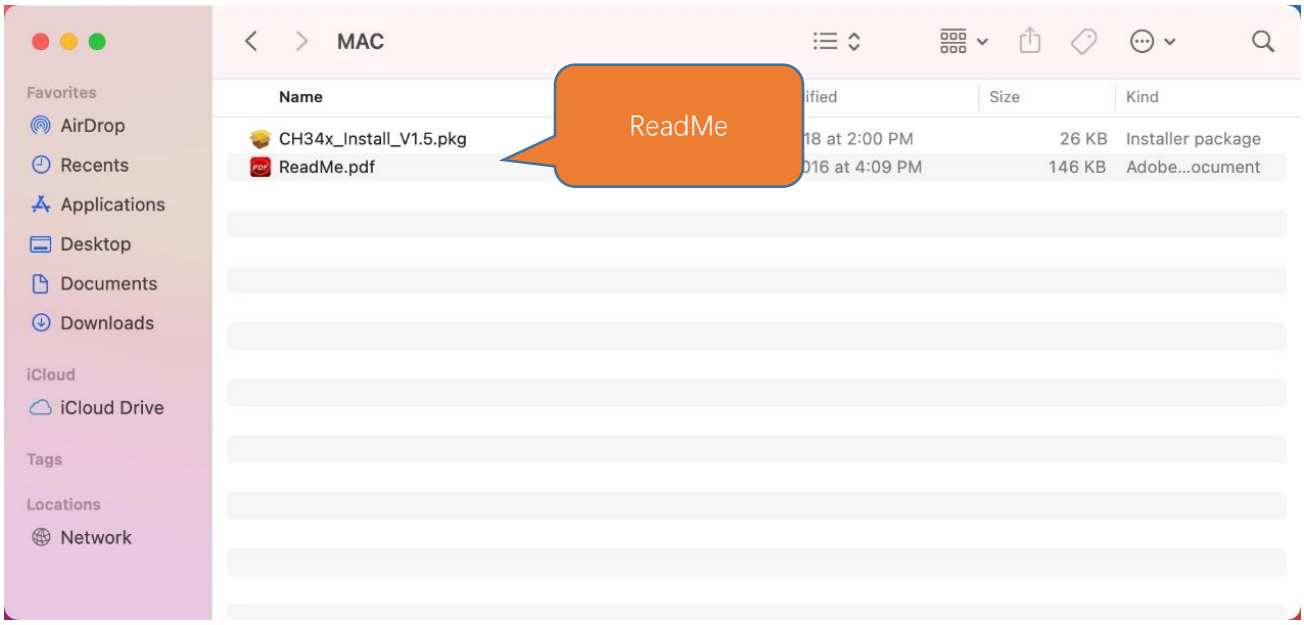
Then, waiting Finsh.



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 ESP32, we need to burn a firmware to ESP32 first.

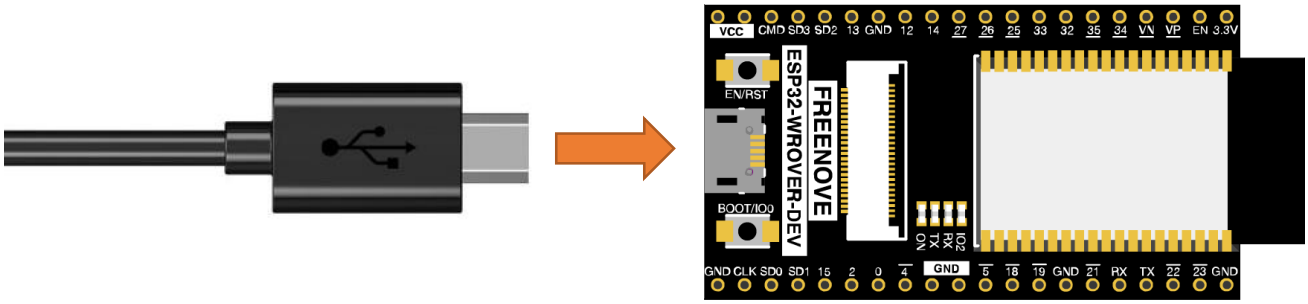
Downloading Micropython Firmware

Official website of microPython: http://micropython.org/
Webpage listing firmware of microPython for ESP32: https://micropython.org/download/esp32spiram/

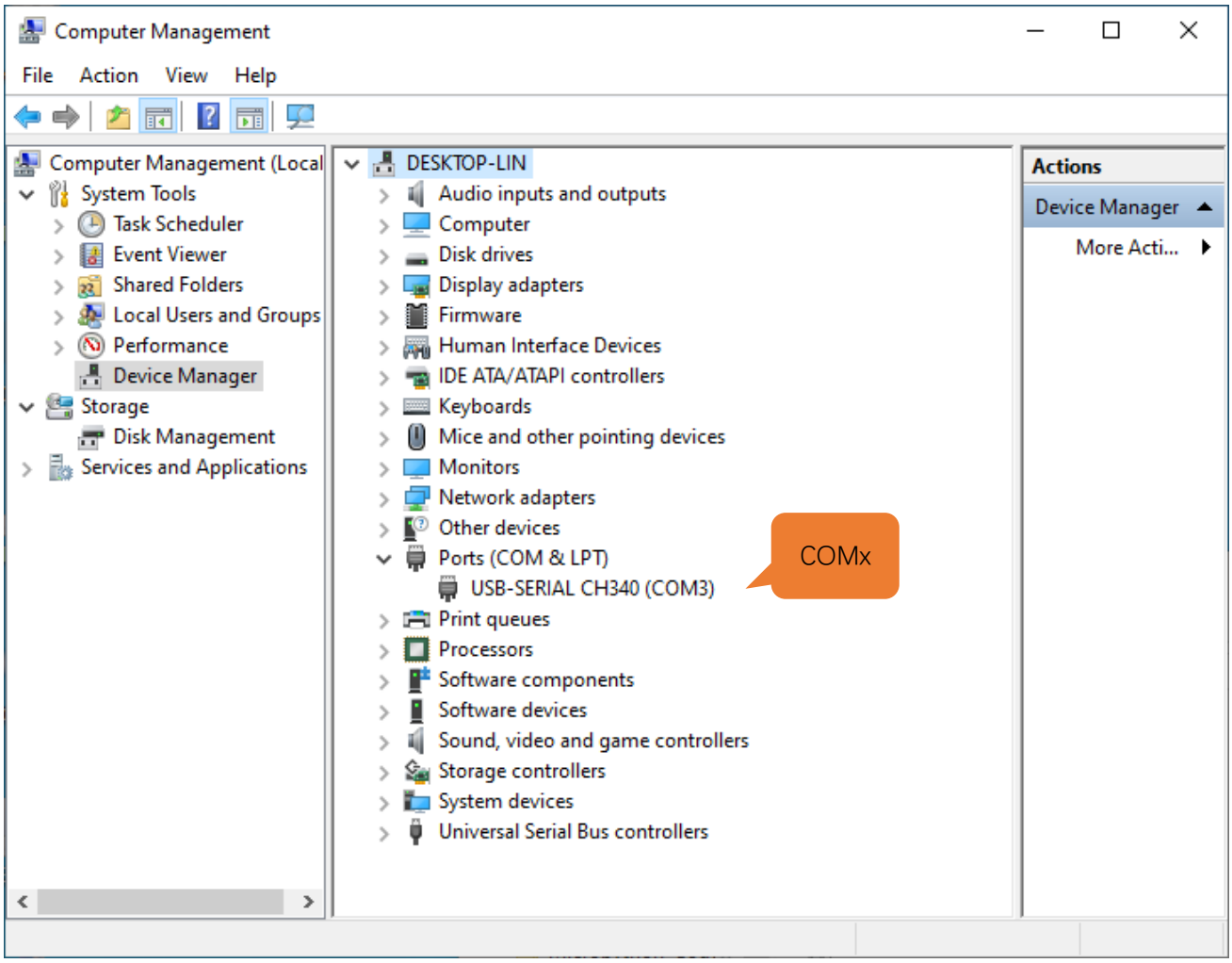
Firmware used in this tutorial is **esp32spiram-20220618-v1.19.1.bin**
This file is also provided in our data folder
"Freenove_LCD_Module/Freenove_LCD_Module_for_ESP32/Python/Python_Firmware".

Burning a Micropython Firmware

Connect your computer and ESP32 with a USB cable.

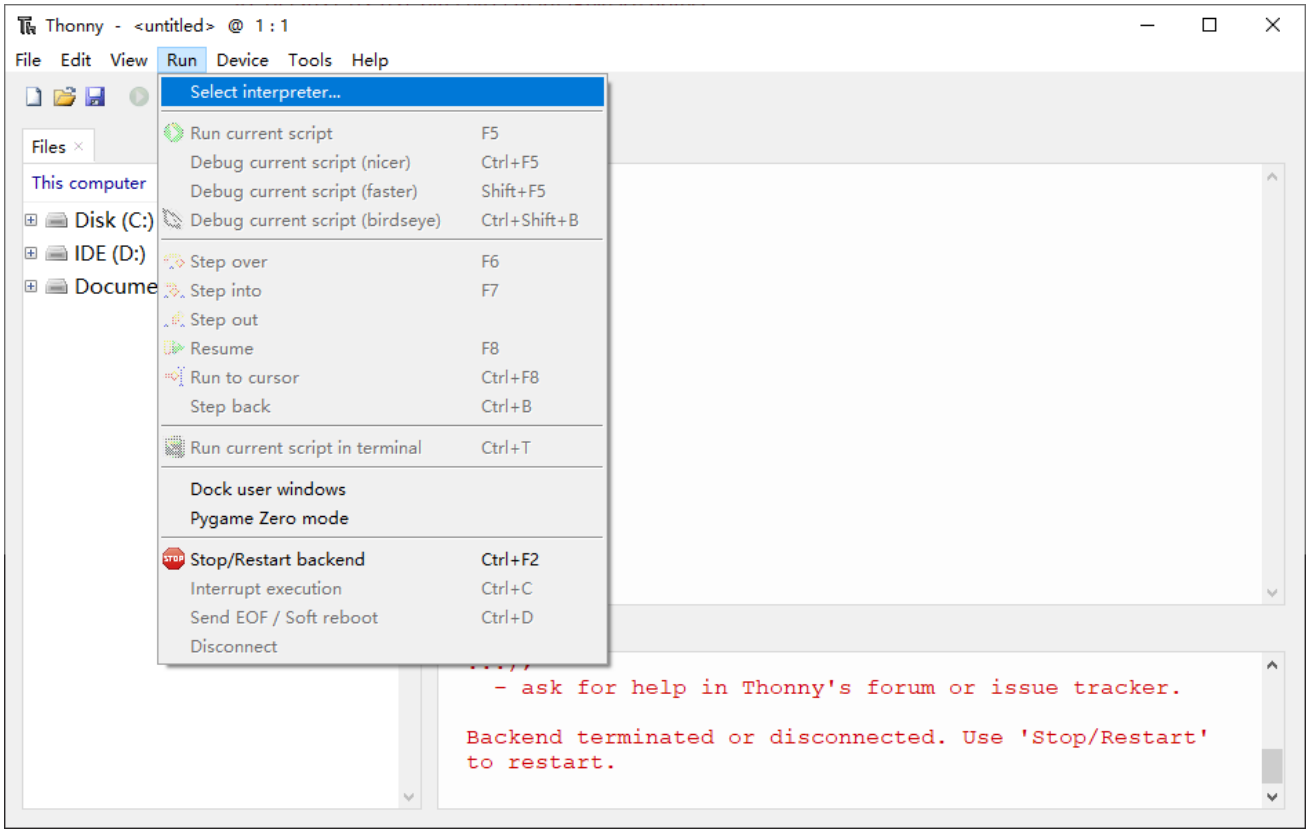


Make sure that the driver has been installed successfully and that it can recognize COM port correctly. Open device manager and expand "Ports".

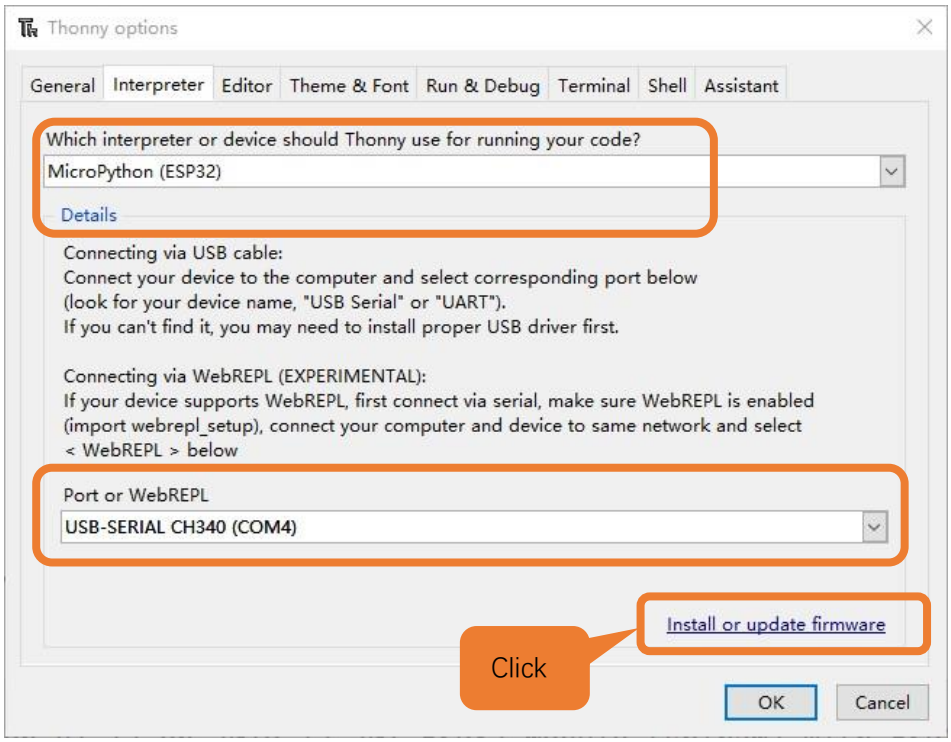


Note: the port of different people may be different, which is a normal situation.

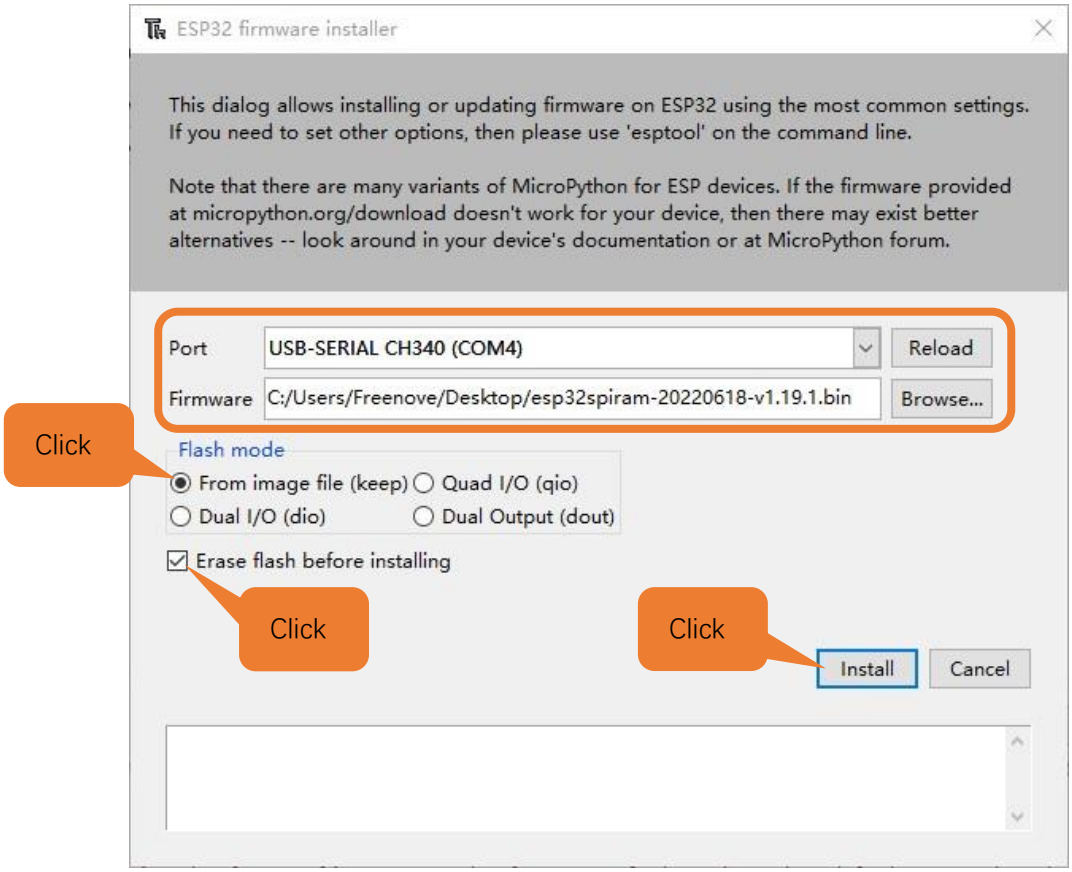
1. Open Thonny, click “run” and select “Select interpreter...””



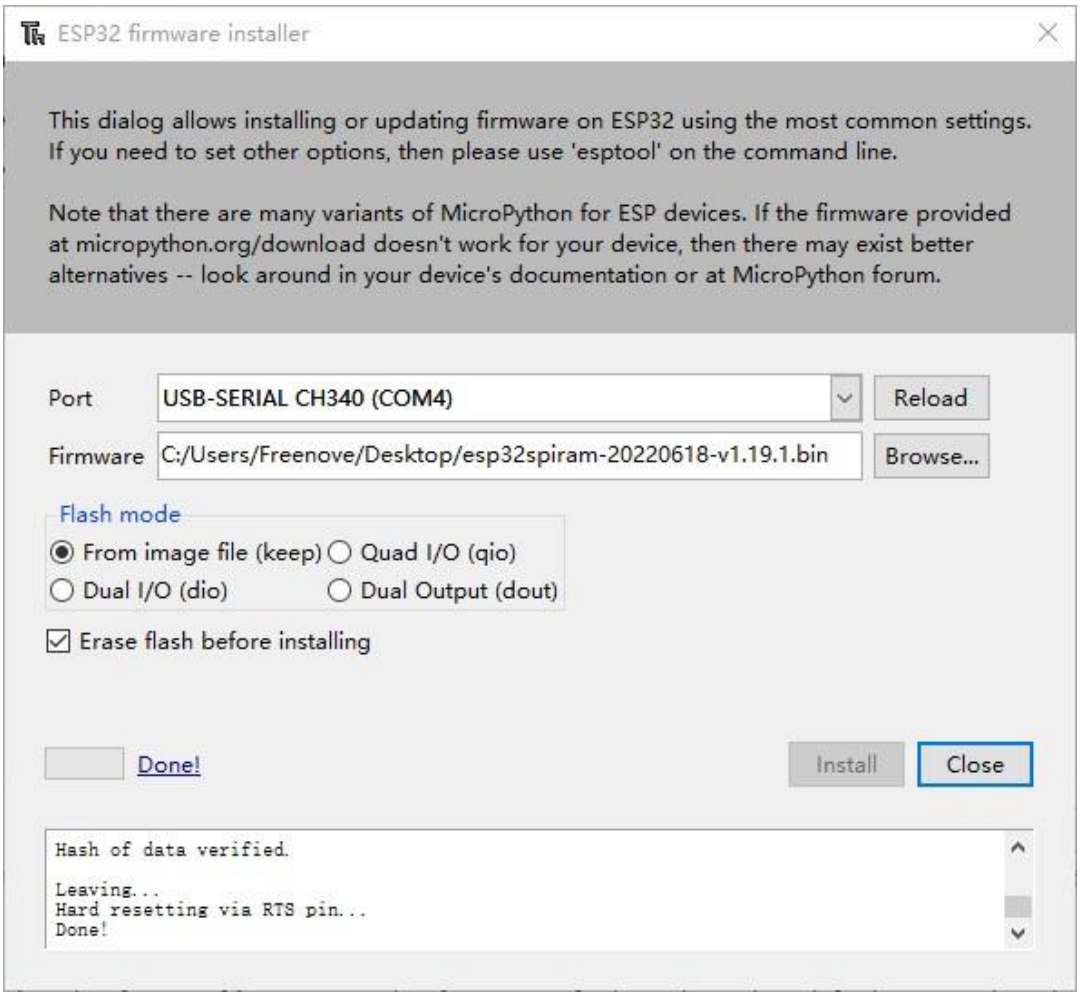
2. Select "Micropython (ESP32)", select "USB-SERIAL CH340 (COM4)", and then click the long button under "Firmware".



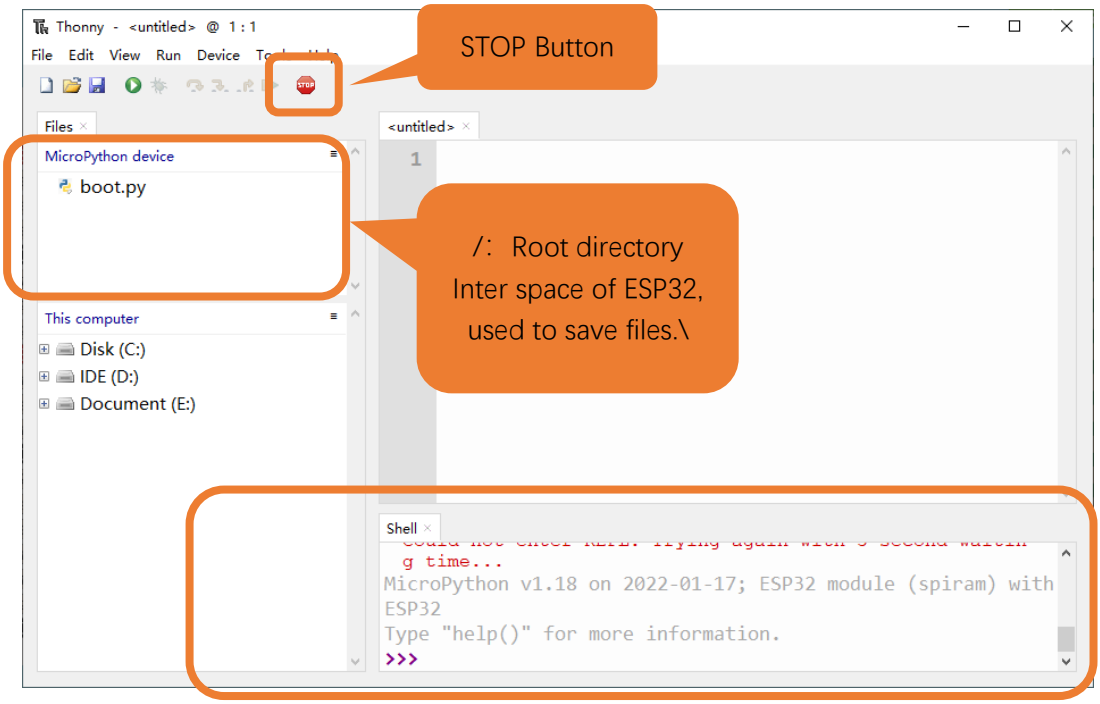
3. The following dialog box pops up. Select "USB-SERIAL CH340 (COM4)" for "Port" and then click "Browse...". Select the previous prepared microPython firmware "**esp32spiram-20220618-v1.19.1.bin**". Check "Erase flash before installing" and click "install" to wait for the prompt of finishing installation.
Here we need to select Flash mode. All four modes are supported on our ESP32 development board. So we're going to leave it as the default.
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.



4. Wait for the installation to be done.



5. Close all dialog boxes, turn to main interface and click “STOP”. As shown in the illustration below

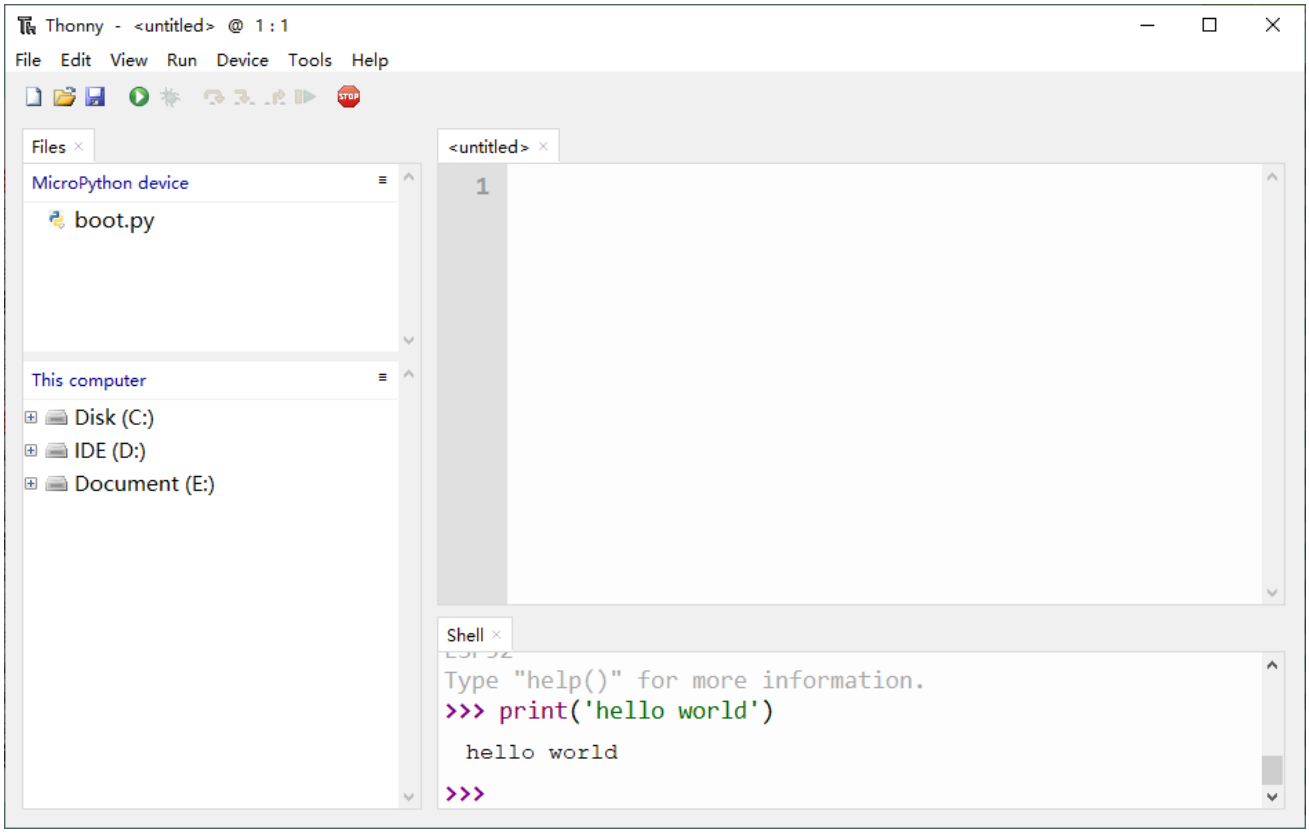


6. So far, all the preparations have been made.

0.5 Testing codes (Important)

Testing Shell Command

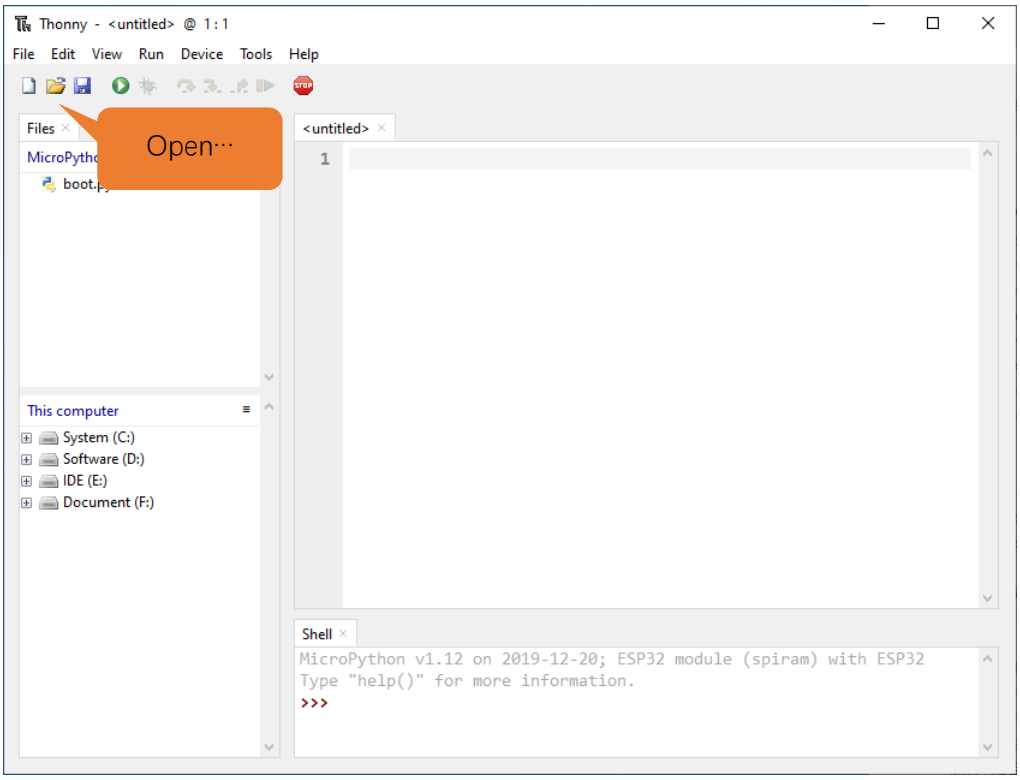
Enter "print('hello world')" in "Shell" and press Enter.



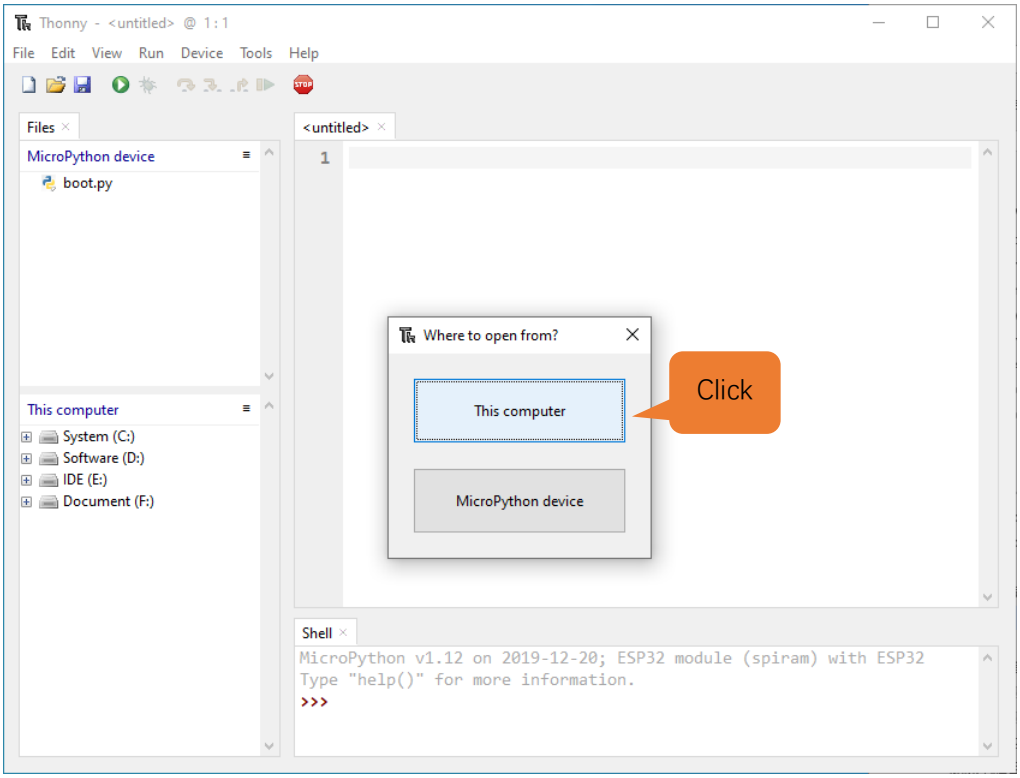
Running Online

ESP32 needs to be connected to a computer when it is run online. Users can use Thonny to writer and debug programs.

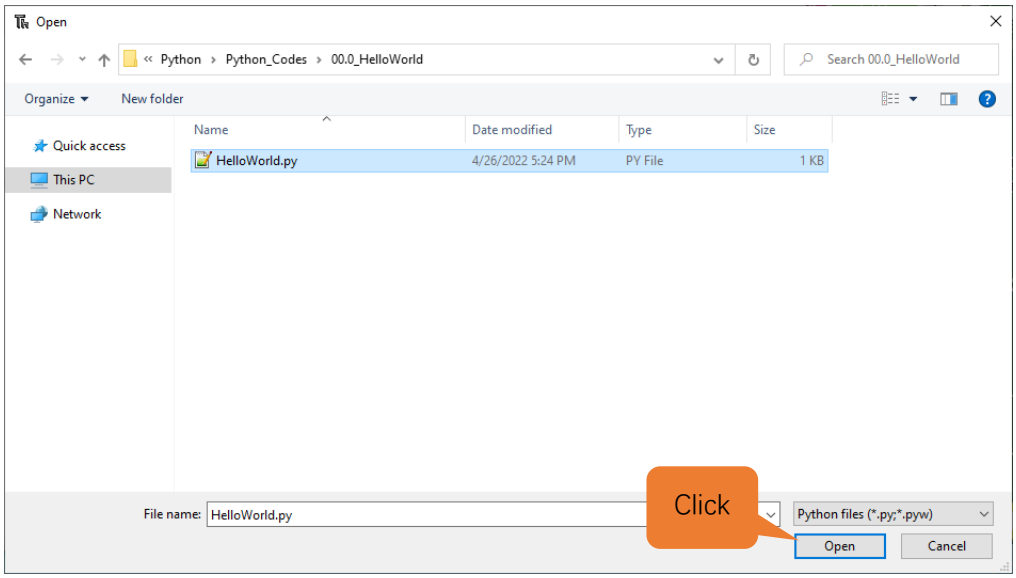
1. Open Thonny and click “Open···”.



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



In the new dialog box, select "**HelloWorld.py**" in "**Freenove_LCD_Module/Freenove_LCD_Module_for_ESP32/Python/Python_Codes/00.0_HelloWorld**" folder.



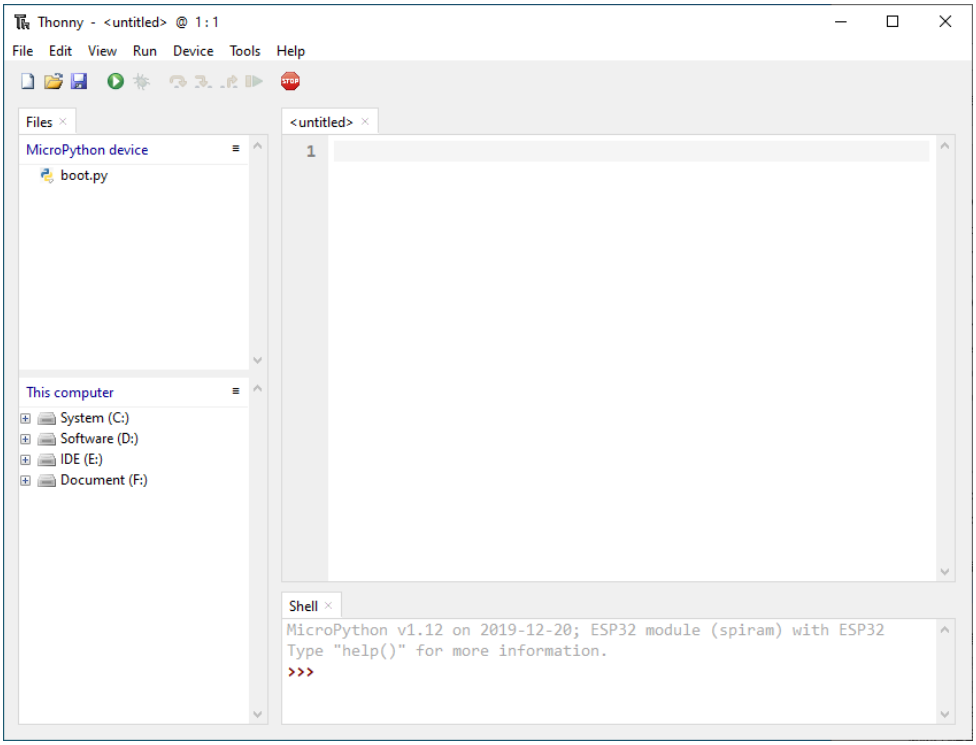
Click "Run current script" to execute the program and "Hello World" will be printed in "Shell".

Note: When running online, if you press the reset key of ESP32, 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.

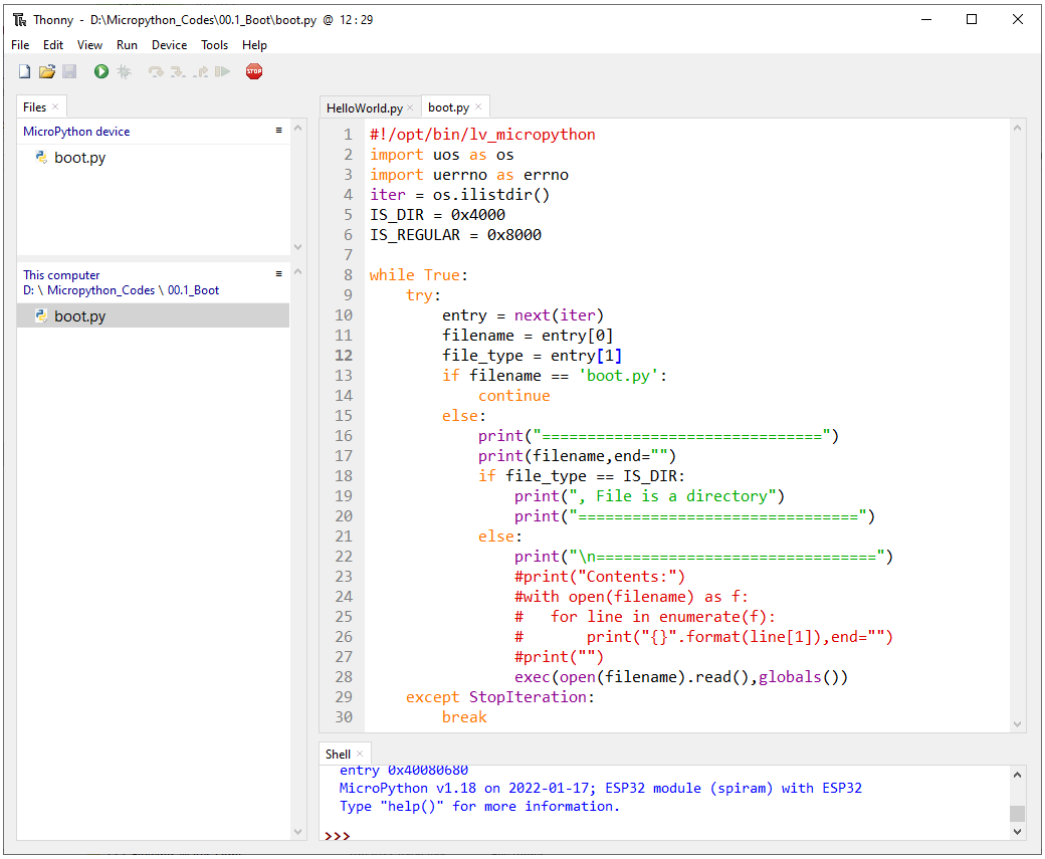
Running Offline （Importance）

After ESP32 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 ESP32 execute user's programs after resetting, we need to add a guiding program in boot.py to execute user's code.

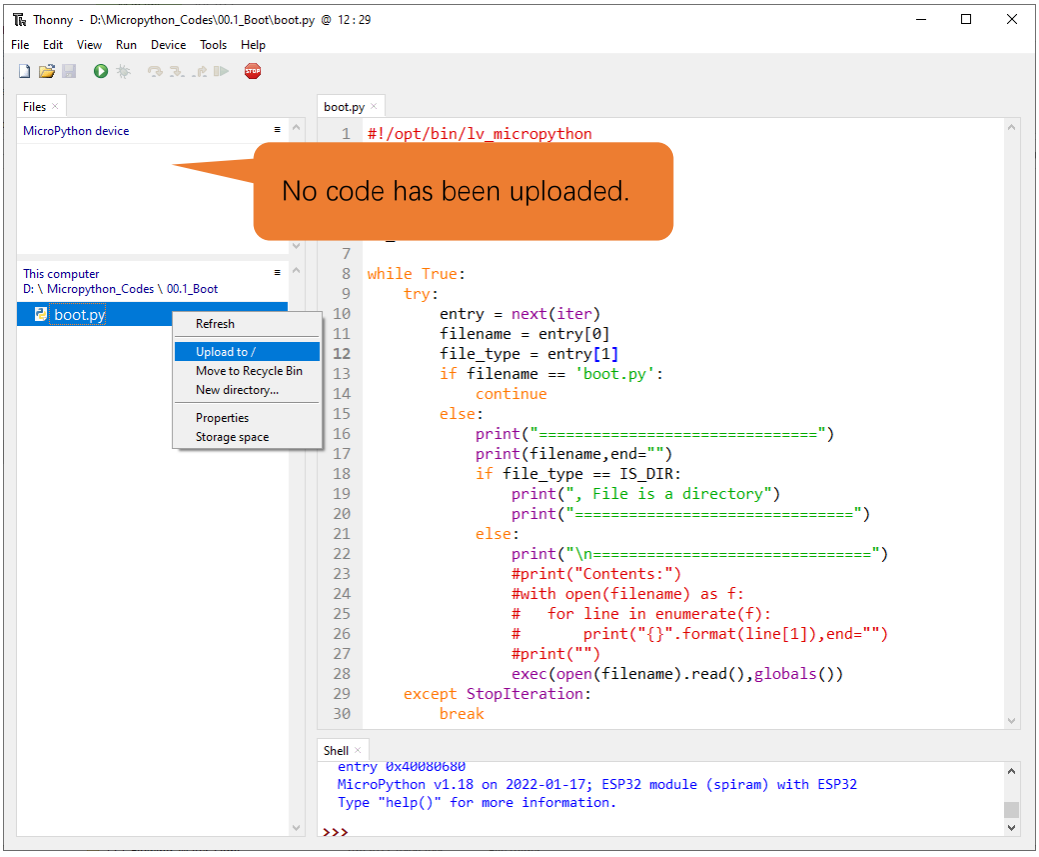
1. Move the program folder “**Freenove_LCD_Module/Freenove_LCD_Module_for_ESP32/Python/Python_Codes**” to disk(D) in advance with the path of “**D:/Micropython_Codes**”. Open “Thonny”.



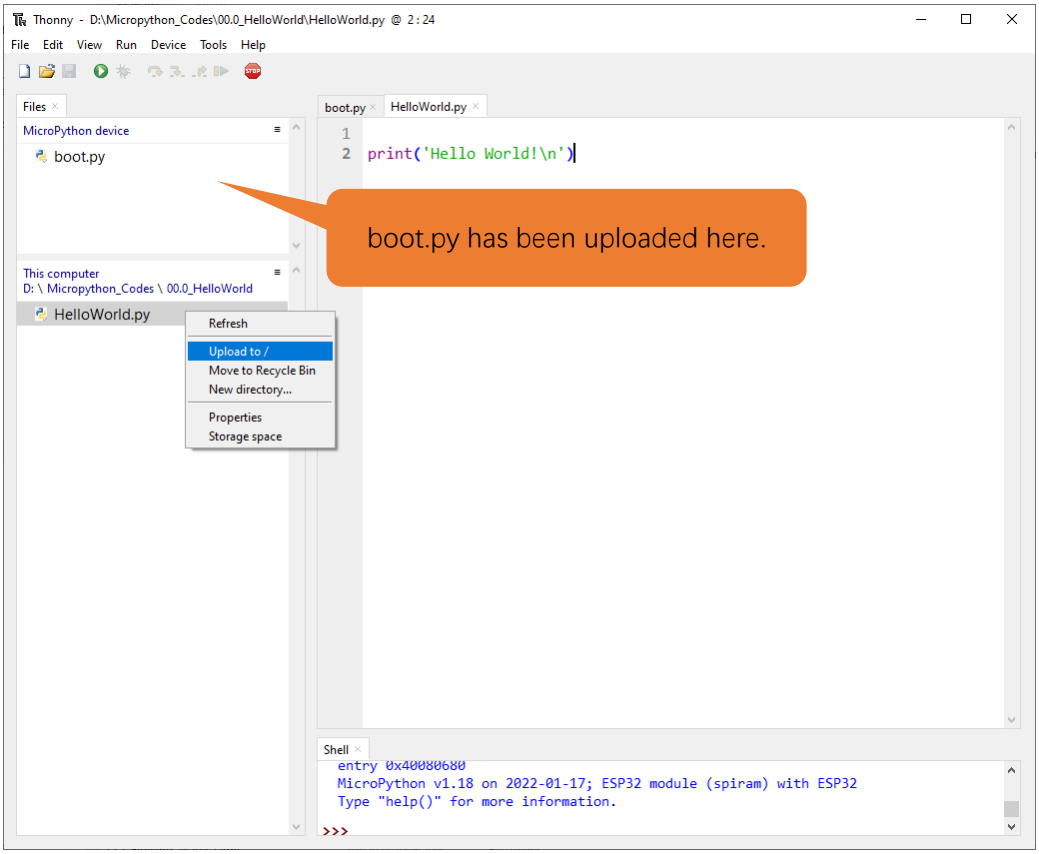
2. Expand “00.1_Boot” in the “Micropython_Codes” in the directory of disk(D), and double-click boot.py, which is provided by us to enable programs in “MicroPython device” to run offline.



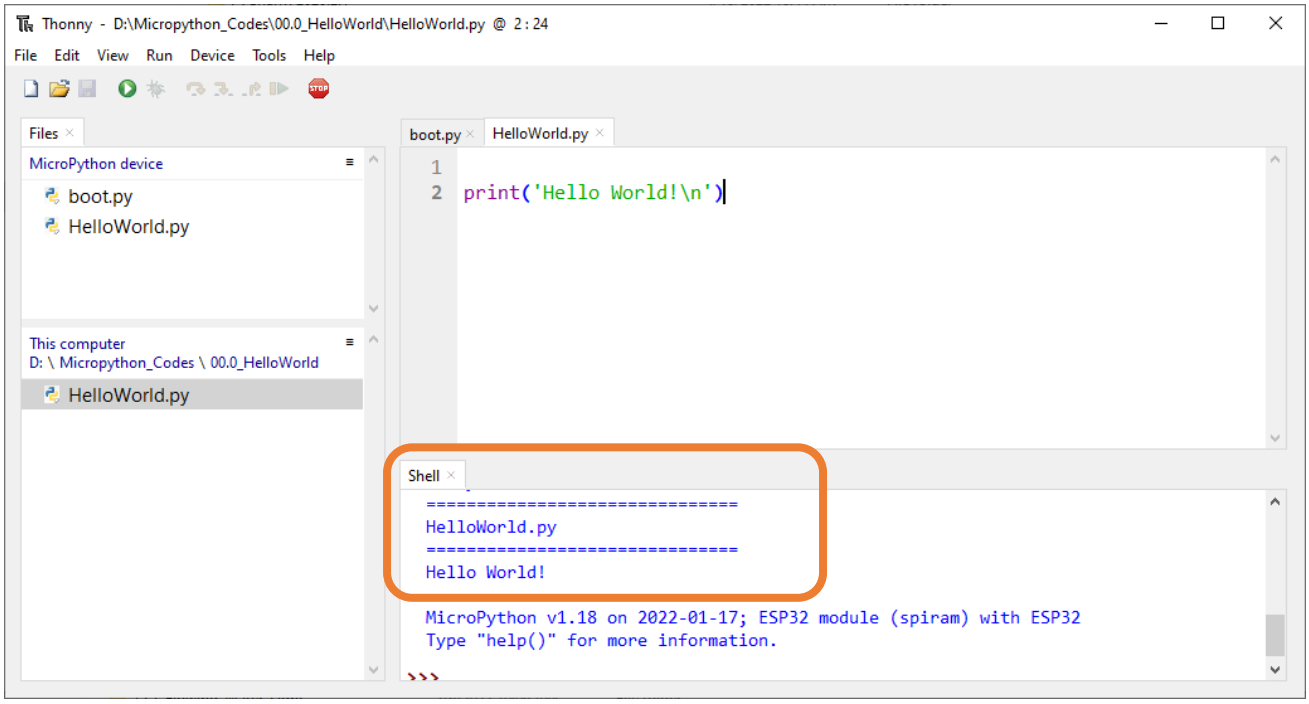
If you want your written programs to run offline, you need to upload boot.py we provided and all your codes to "MicroPython device" and press ESP32's reset key. Here we use programs 00.0 and 00.1 as examples. Select "boot.py", right-click to select "Upload to /".



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

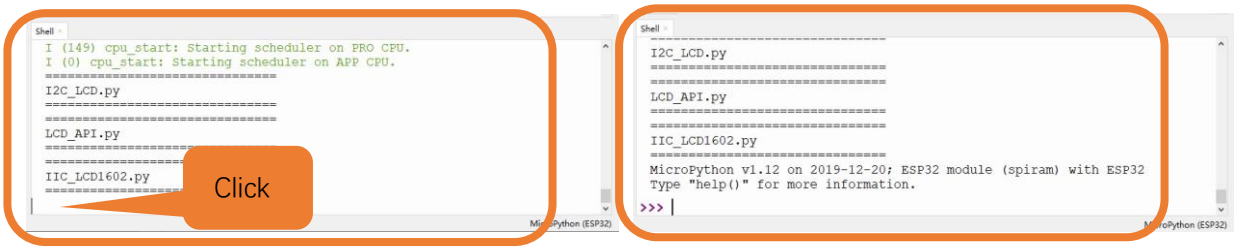


3. Press the reset key and in the box of the illustration below, you can see the code is executed.



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.

When your "Shell" is unresponsive or abnormal, you can exit the running program by pressing "CTRL" and "C" simultaneously.



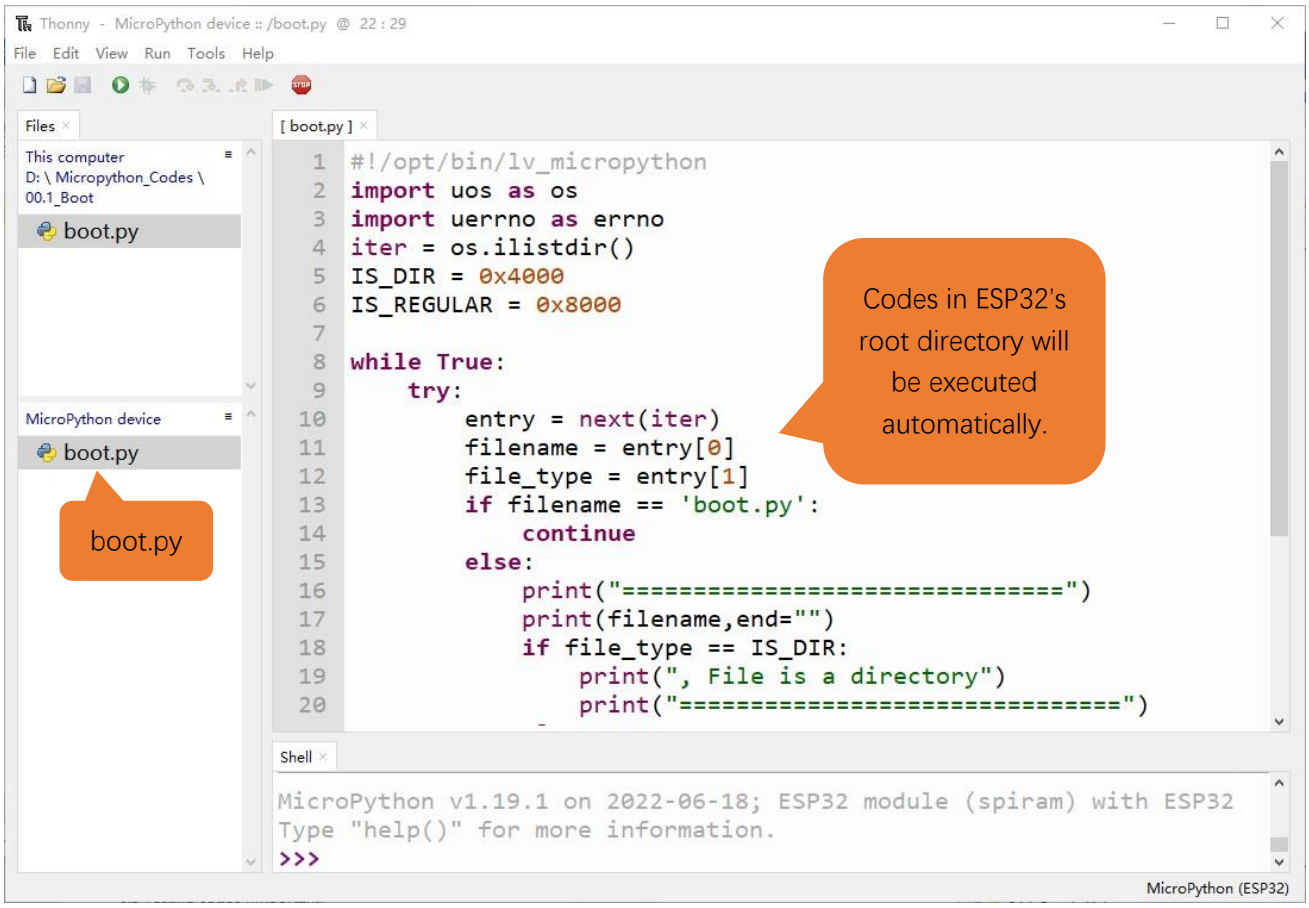
If the ESP8266 does not work properly, you can press CTRL and C at the same time to observe whether the Shell responds. If the ESP8266 still does not work properly, you can also rewrite the Micropython firmware and perform related operations again.

For your convenience, we have placed boot.py in each example.

0.6 Thonny Common Operation

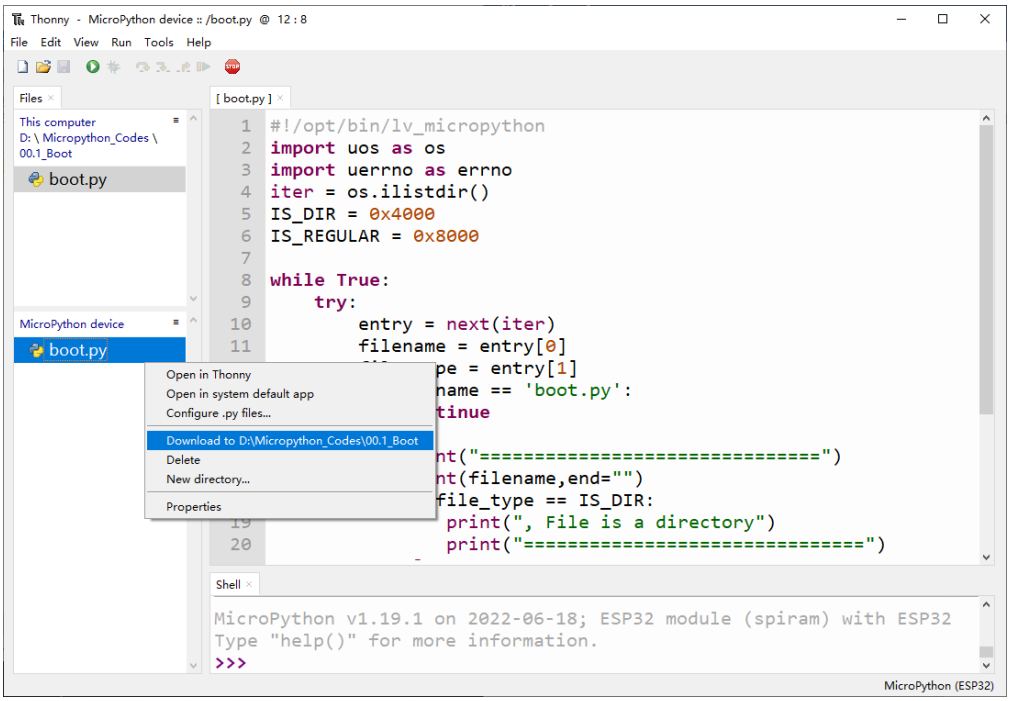
Uploading Code to ESP32

Each time when ESP32 restarts, if there is a “boot.py” in the root directory, it will execute this code first.



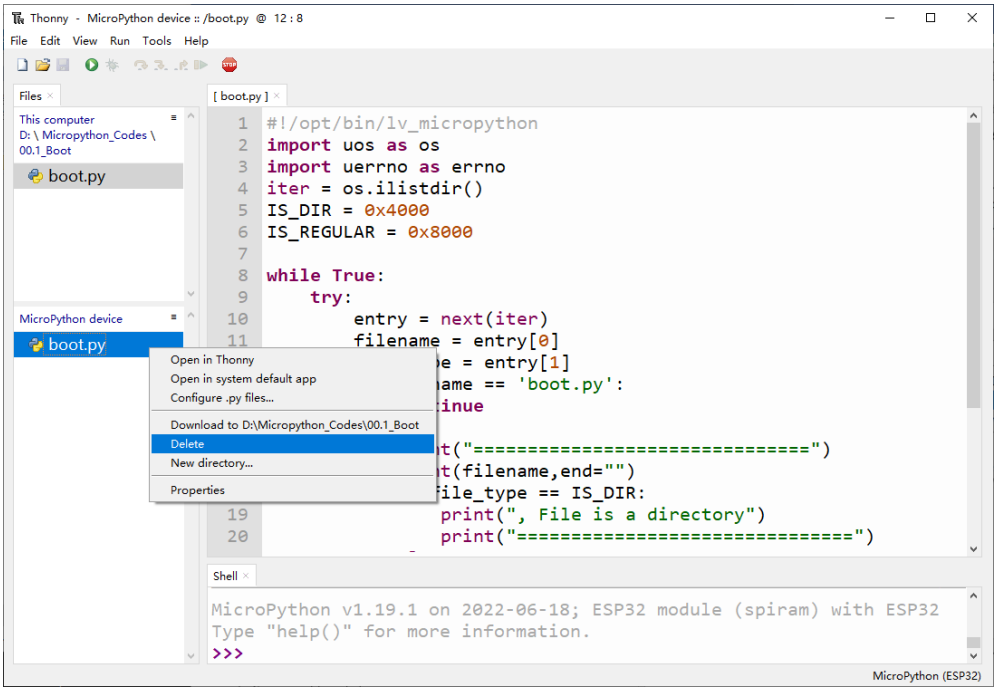
Downloading Code to Computer

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



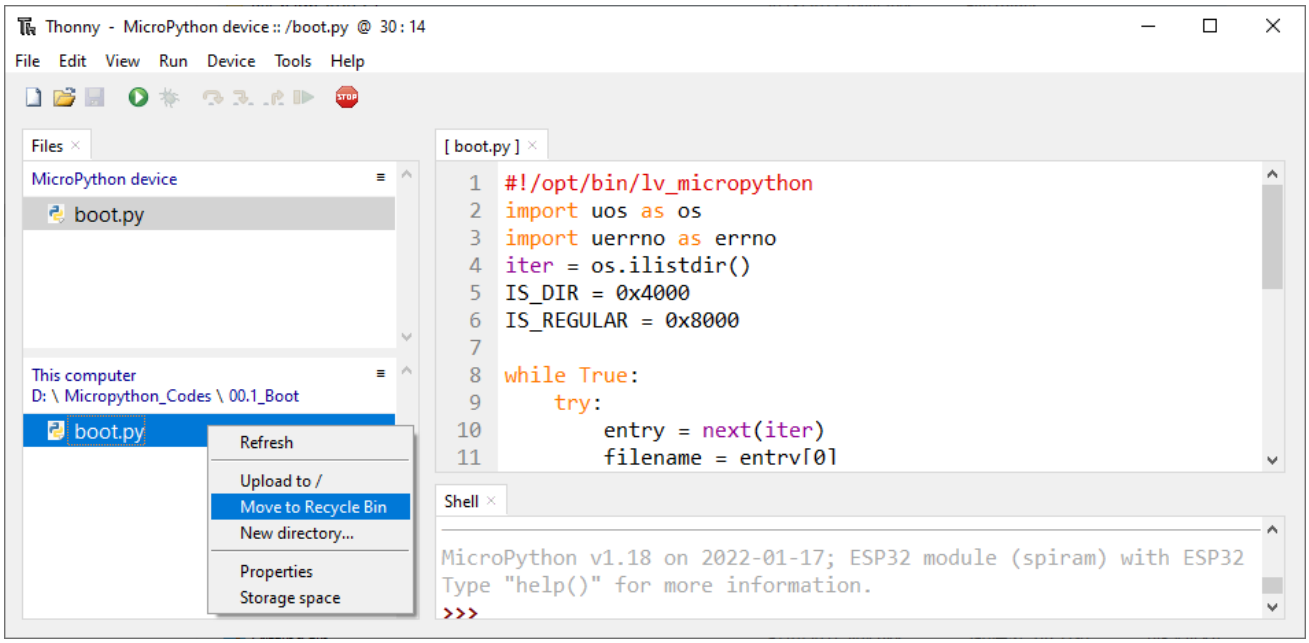
Deleting Files from ESP32’s Root Directory

Select “boot.py” in “MicroPython device”, right-click it and select “Delete” to delete “boot.py” from ESP32’s root directory.



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". Back up your files before deleting them to avoid data loss.



0.7 Note

Though there are many pins available on ESP32, some of them have been connected to peripheral equipment, so we should avoid using such pins to prevent pin conflicts. For example, when downloading programs, make sure that the pin state of Strapping Pin, when resetting, is consistent with the default level; do NOT use Flash Pin; Do NOT use Cam Pin when using Camera function.

Strapping Pin

The state of Strapping Pin can affect the functions of ESP32 after it is reset, as shown in the table below.

Voltage of Internal LDO (VDD_SDIO)			
Pin	Default	3.3 V	1.8 V
MTDI	Pull-down	0	1
Booting Mode			
Pin	Default	SPI Boot	Download Boot
GPIO0	Pull-up	1	0
GPIO2	Pull-down	Don't-care	0
Enabling/Disabling Debugging Log Print over U0TXD During Booting			
Pin	Default	U0TXD Active	U0TXD Silent
MTDO	Pull-up	1	0

Timing of SDIO Slave					
Pin	Default	Falling-edge Sampling Falling-edge Output	Falling-edge Sampling Rising-edge Output	Rising-edge Sampling Falling-edge Output	Rising-edge Sampling Rising-edge Output
MTDO	Pull-up	0	0	1	1
GPIO5	Pull-up	0	1	0	1

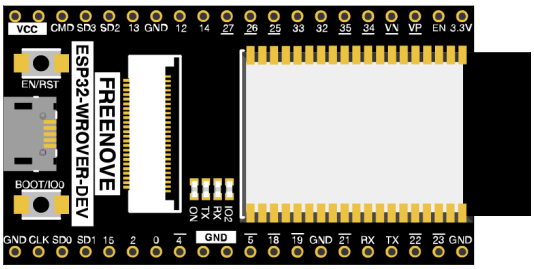
If you have any difficulties or questions with this tutorial or toolkit, feel free to ask for our quick and free technical support through support@freenove.com at any time.
or check: https://www.espressif.com/sites/default/files/documentation/esp32-wrover_datasheet_en.pdf

Flash Pin

GPIO6-11 has been used to connect the integrated SPI flash on the module, and is used when GPIO 0 is power on and at high level. Flash is related to the operation of the whole chip, so the external pin GPIO6-11 cannot be used as an experimental pin for external circuits, otherwise it may cause errors in the operation of the program.
GPIO16-17 has been used to connect the integrated PSRAM on the module.
Because of external pull-up, MTDI pin is not suggested to be used as a touch sensor. For details, please refer to Peripheral Interface and Sensor chapter in "ESP32_Data_Sheet".
For more relevant information, please click:
https://www.espressif.com/sites/default/files/documentation/esp32-wrover_datasheet_en.pdf

Cam Pin

When using the cam camera of our ESP32-WROVER, please check the pins of it. Pins with underlined numbers are used by the cam camera function, if you want to use other functions besides it, please avoid using them.



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

If you have any questions about the information of GPIO, you can click here to go back to ESP32-WROVER to view specific information about GPIO.
Or check: https://www.espressif.com/sites/default/files/documentation/esp32-wrover_datasheet_en.pdf

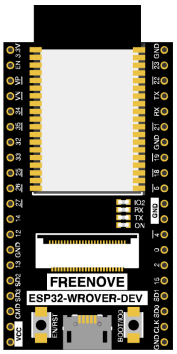
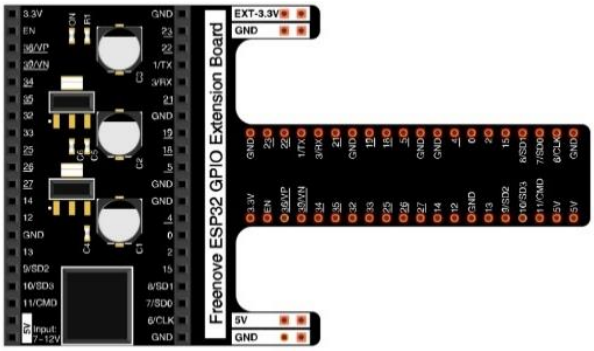
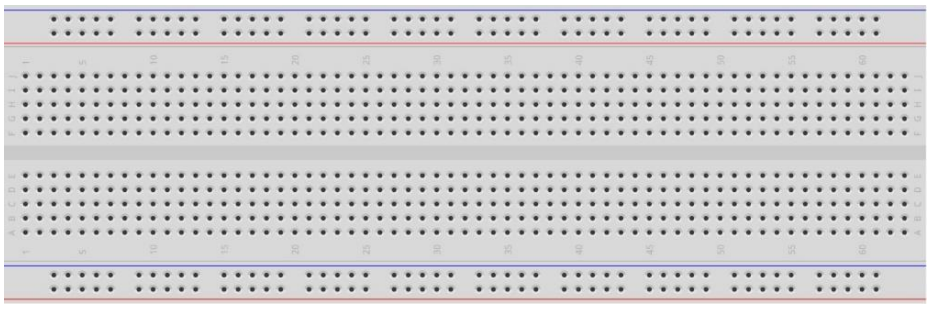
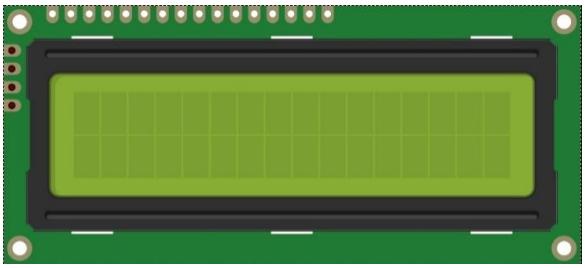
Chapter 1 LCD1602

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

Project 1.1 LCD1602

In this section we learn how to use lcd1602 to display something.
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

ESP32-WROVER x1 	GPIO Extension Board x1 
Breadboard x1 	
LCD1602 Module x1 	Jumper F/M x4 

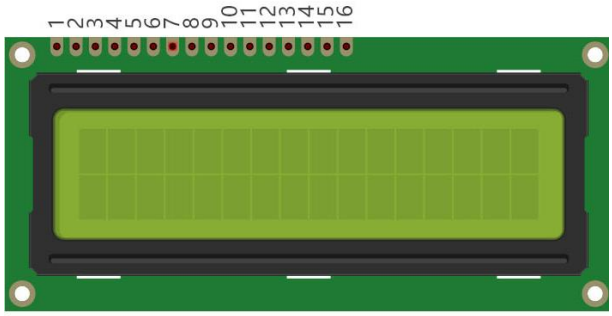
Component knowledge

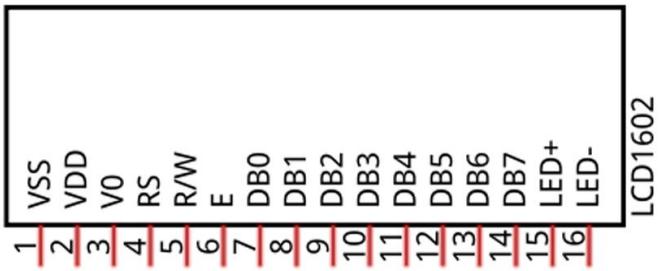
I2C communication

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

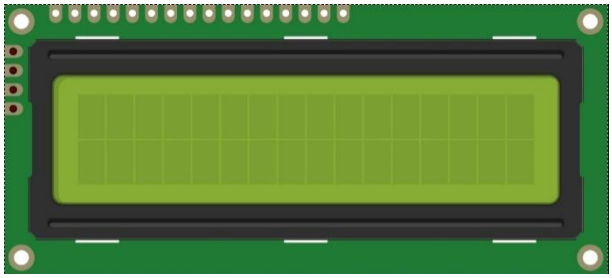
LCD1602 communication

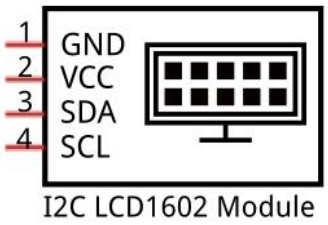
The LCD1602 Display Screen can display 2 lines of characters in 16 columns. It is capable of displaying numbers, letters, symbols, ASCII code and so on. As shown below is a monochrome LCD1602 Display Screen along with its circuit pin diagram





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

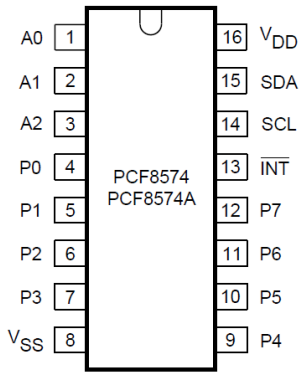




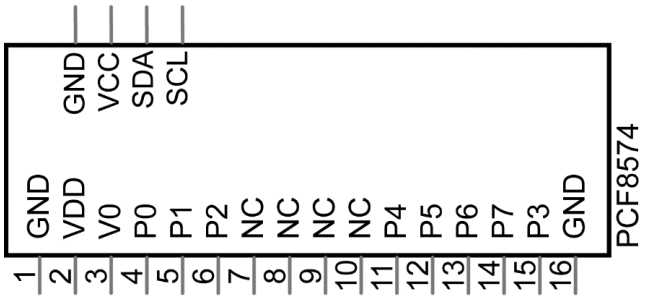
The serial-to-parallel IC chip used in this module is PCF8574T (PCF8574AT), and its default I2C address is 0x27(0x3F).

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

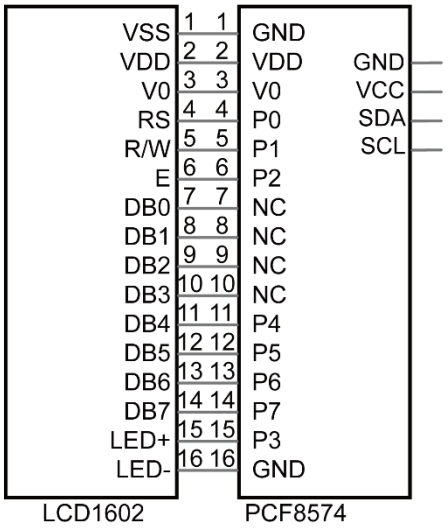
PCF8574 chip pin diagram



PCF8574 module pin diagram



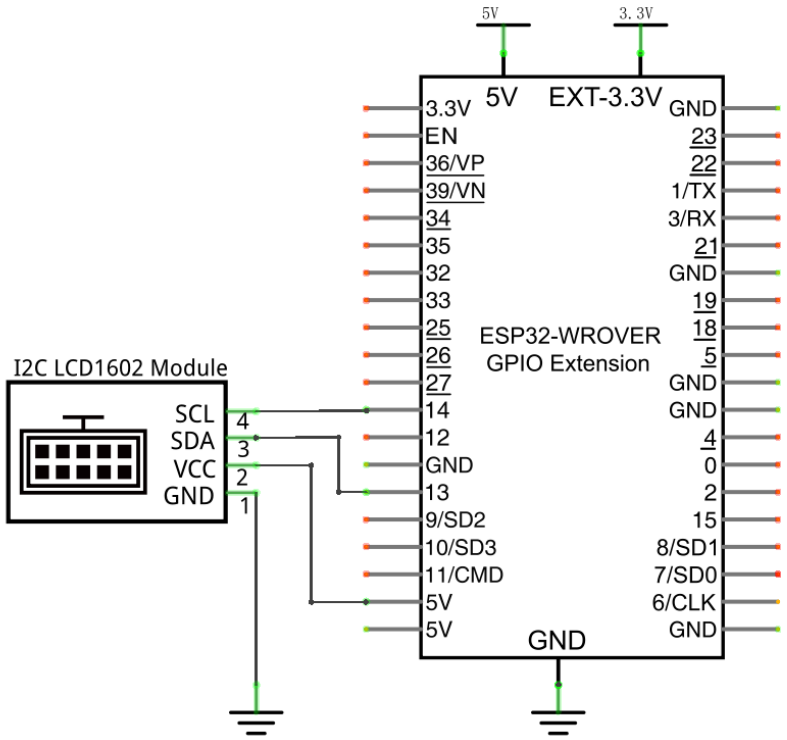
PCF8574 module pin and LCD1602 pin are corresponding to each other and connected with each other:



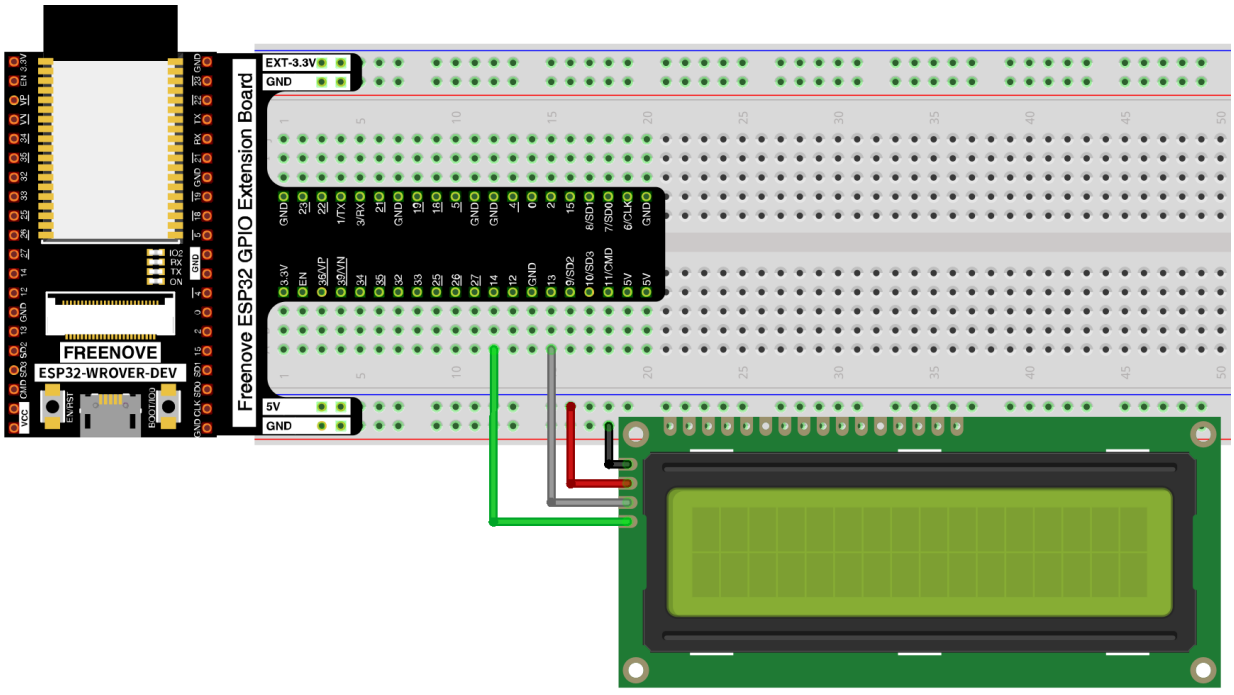
So we only need 4 pins to control the 16 pins of the LCD1602 Display Screen through the I2C interface. In this project, we will use the I2C LCD1602 to display some static characters and dynamic variables.

Circuit

Schematic diagram



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

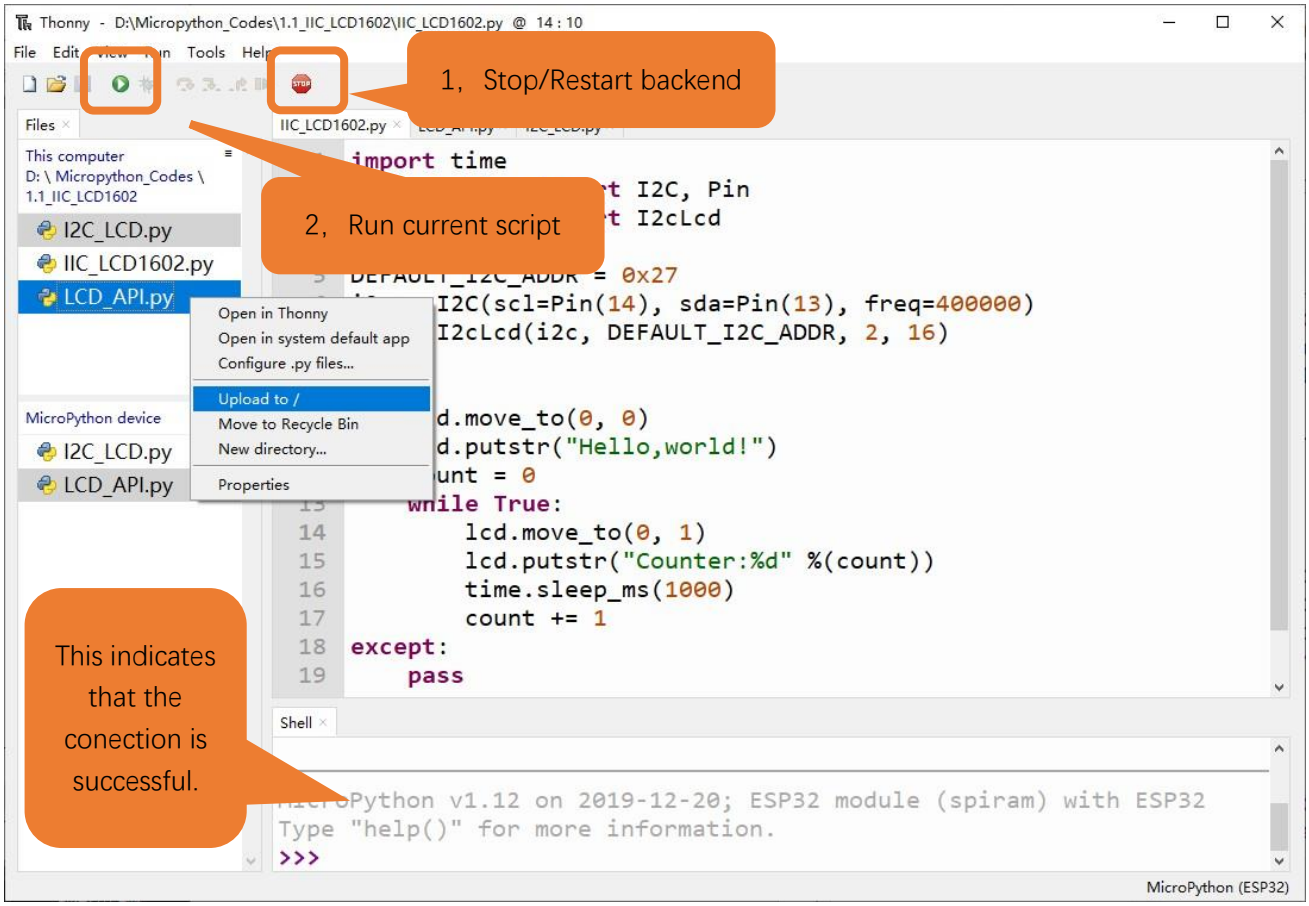


Code

Move the program folder "**Freenove_LCD_Module/Freenove_LCD_Module_for_ESP32/Python/Python_Codes**" to disk(D) in advance with the path of "**D:/Micropython_Codes**".

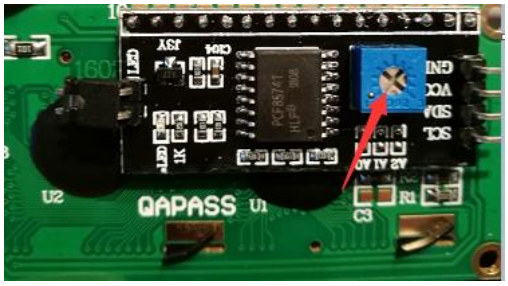
Open "Thonny", click "This computer" → "D:" → "Micropython_Codes" → "1.1_I2C_LCD1602". Select "I2C_LCD.py"and "LCD_API.py", right click your mouse to select "Upload to /", wait for "I2C_LCD.py" and "LCD_API.py" to be uploaded to ESP32-WROVER and then double click "I2C_LCD1602.py".

1.1_I2C_LCD1602



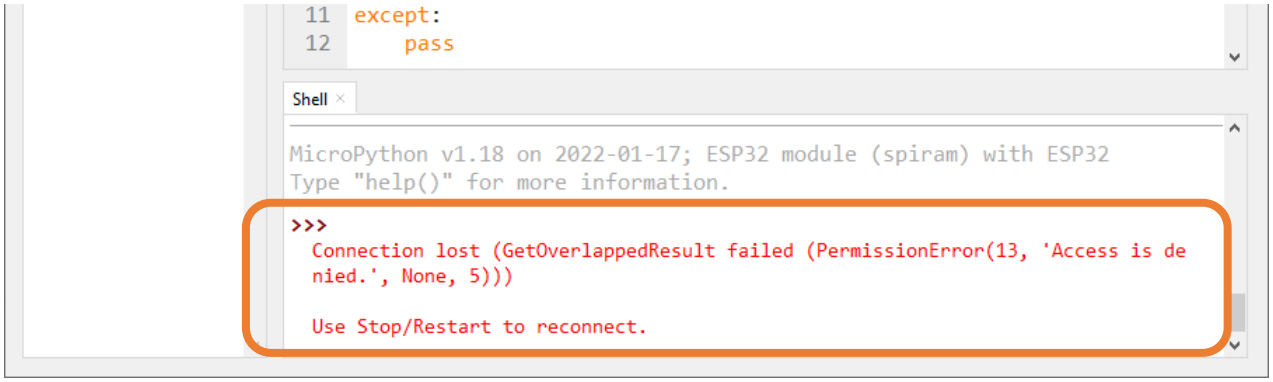
Click "Run current script" and LCD1602 displays some characters.Press Ctrl+C or click "Stop/Restart backend" to exit the program.

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



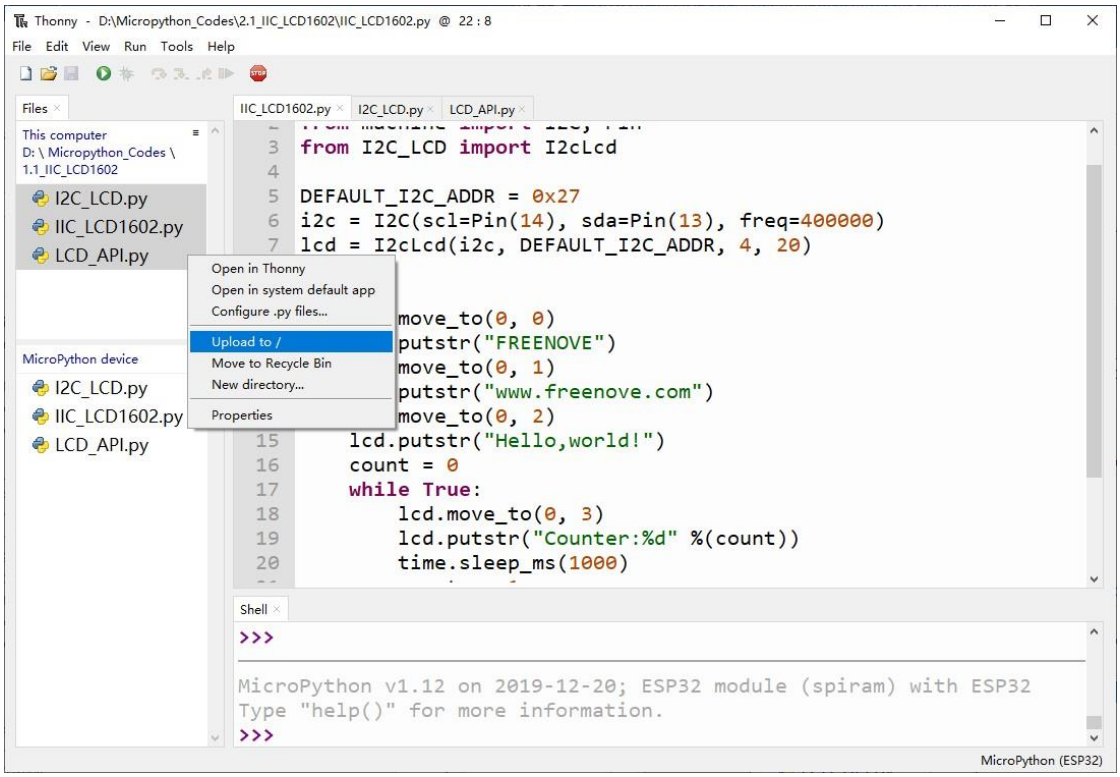
Note:

This is the code running online. If you disconnect USB cable and repower ESP32 or press its reset key, LED stops blinking and the following messages will be displayed in Thonny.

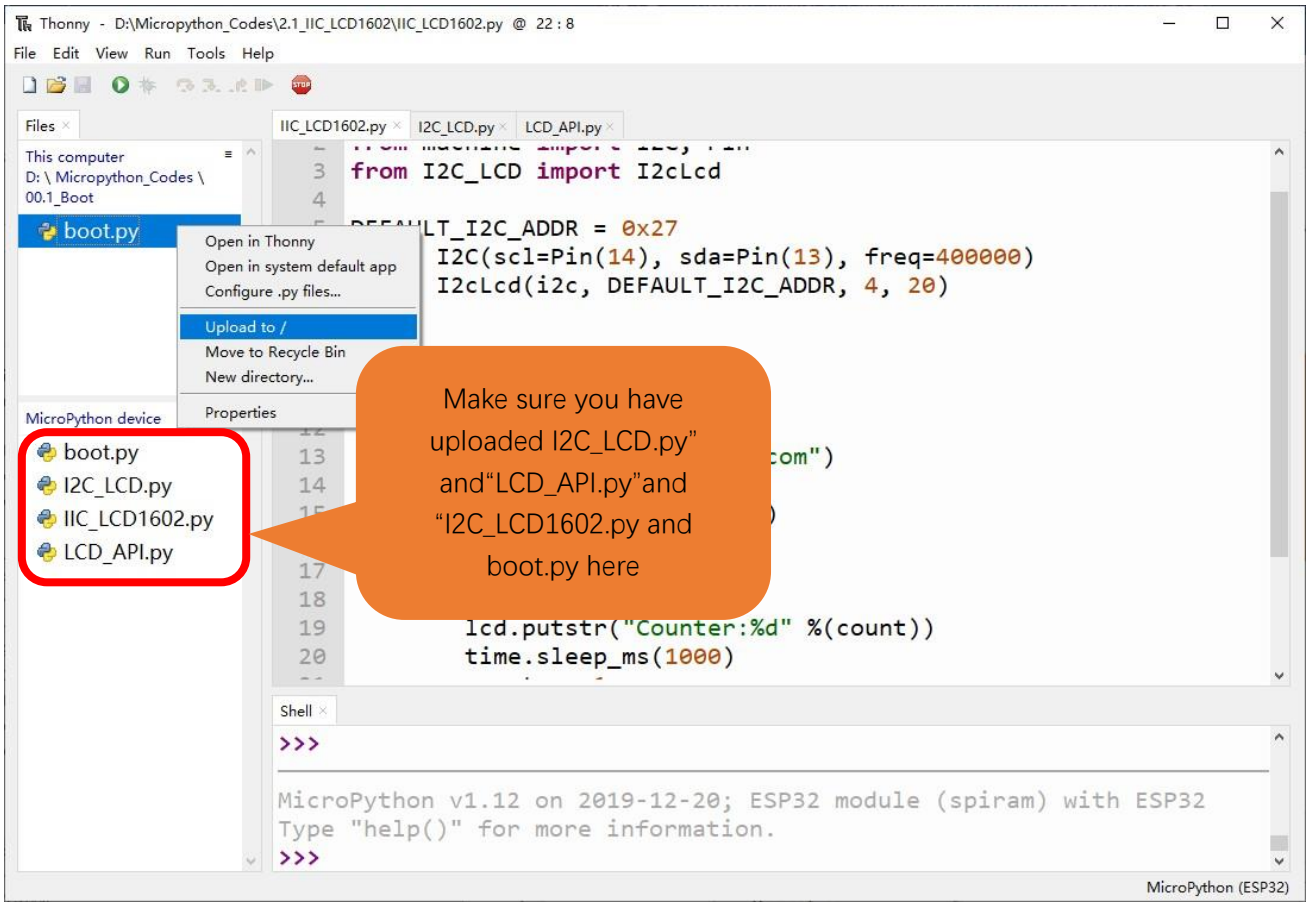


Uploading code to ESP32

As shown in the following illustration, Select "I2C_LCD.py"and"LCD_API.py"and"I2C_LCD1602.py", right click your mouse to select "Upload to /" to upload code to ESP32.



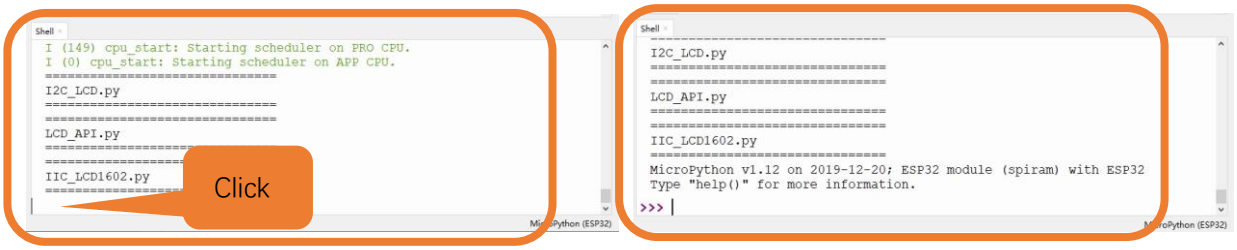
Upload boot.py in the same way.



Press the reset key of ESP32 and you can see LCD1602 displays some characters.

Note:

Codes here is run offline. If you want to stop running offline and enter Shell, You need to use the mouse to click on the Shell,Pressing the keyboard keys "CTRL" and "C" at the same time.

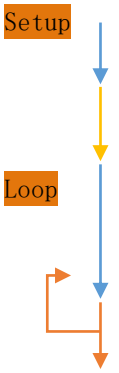


If you have any concerns, please contact us via: support@freenove.com

The following is the program code:

```
1  import time
2  from machine import I2C, Pin
3  from I2C_LCD import I2cLcd
4
5  DEFAULT_I2C_ADDR = 0x27
6  i2c = I2C(scl=Pin(14), sda=Pin(13), freq=400000)
7  lcd = I2cLcd(i2c, DEFAULT_I2C_ADDR, 2, 16)
8
9  try:
10     lcd.move_to(0, 0)
11     lcd.putstr("Hello,world!")
12     count = 0
13     while True:
14         lcd.move_to(0, 1)
15         lcd.putstr("Counter:%d" %(count))
16         time.sleep_ms(1000)
17         count += 1
18 except:
19     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.




```
1  import time
2  from machine import I2C, Pin
3  from I2C_LCD import I2cLcd
…
7  lcd = I2cLcd(i2c, DEFAULT_I2C_ADDR, 2, 16)
…  try:
13     while True:
…          ...
18 except:
19     pass
```

Import time, I2C and I2C_LCD modules.

```
1  import time
2  from machine import I2C, Pin
3  from I2C_LCD import I2cLcd
```

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

```
5  DEFAULT_I2C_ADDR = 0x27
```

Initialize I2C pins and associate them with I2CLCD module, and then set the number of rows and columns for LCD1602.

```
6  i2c = I2C(scl=Pin(14), sda=Pin(13), freq=400000)
```

```
7   lcd = I2cLcd(i2c, DEFAULT_I2C_ADDR, 2, 16)
```

Move the cursor of LCD1602 to the first row, first column, and print out "Hello, world!"

```
10      lcd.move_to(0, 0)
11      lcd.putstr("Hello,world!")
```

The second line of LCD1602 continuously prints the number of seconds after the ESP32 program runs.

```
13      while True:
14          lcd.move_to(0, 1)
15          lcd.putstr("Counter:%d" %(count))
16          time.sleep_ms(1000)
17          count += 1
```

Execute codes in a while loop.

```
13      while True:
...         ...
```

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 ESP32 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.

```
9   try:
...     ...
18  except:
19      pass
```

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.

```
13      while True:
14          lcd.move_to(0, 1)
15          lcd.putstr("Counter:%d" %(count))
16          time.sleep_ms(1000)
17          count += 1
```

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 I2cLcd

Before each use of the object **I2cLcd**, please make sure that **I2C_LCD.py** and **LCD_API.py** have been uploaded to "/" of ESP32, and then add the statement "**from I2C_LCD import I2cLcd**" to the top of the python file.

clear(): Clear the LCD1602 screen display.

show_cursor(): Show the cursor of LCD1602.

hide_cursor(): Hide the cursor of LCD1602.

blink_cursor_on(): Turn on cursor blinking.

blink_cursor_off(): Turn off cursor blinking.

display_on(): Turn on the display function of LCD1602.

display_off(): Turn on the display function of LCD1602.

backlight_on(): Turn on the backlight of LCD1602.

backlight_off(): Turn on the backlight of LCD1602.

move_to(cursor_x, cursor_y): Move the cursor to a specified position.

- **cursor_x:** Column cursor_x
- **cursor_y:** Row cursor_y

putchar(char): Print the character in the bracket on LCD1602

putstr(string): Print the string in the bracket on LCD1602.

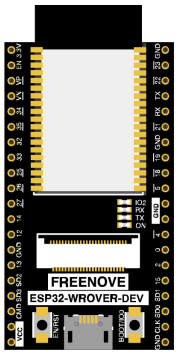
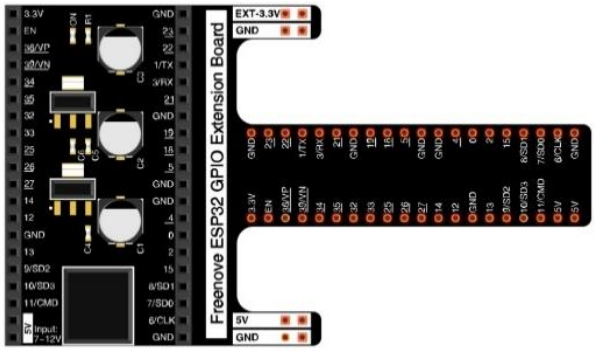
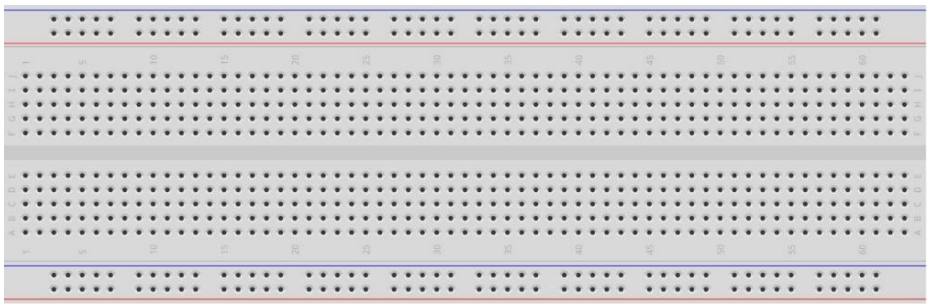
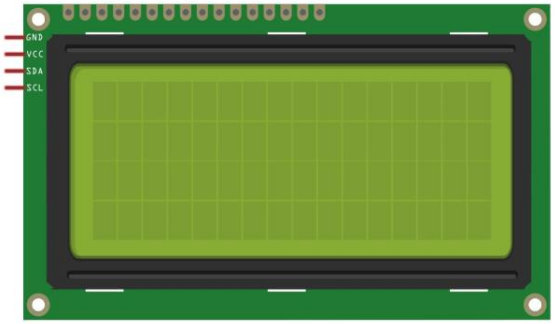
Chapter 2 LCD2004

In the previous chapter, we studied the LCD1602 display. In order to display more content,In this chapter, we will learn about the LCD2004 Display Screen.

Project 1.1 LCD2004

In this section we learn how to use lcd2004 to display something.

Component List

ESP32-WROVER x1 	GPIO Extension Board x1 
Breadboard x1 	
LCD2004 Module x1 	Jumper F/M x4 

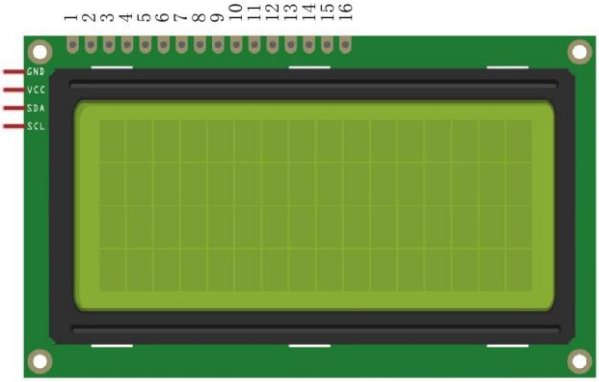
Component knowledge

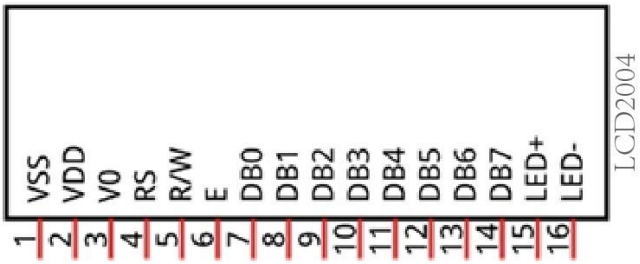
I2C communication

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

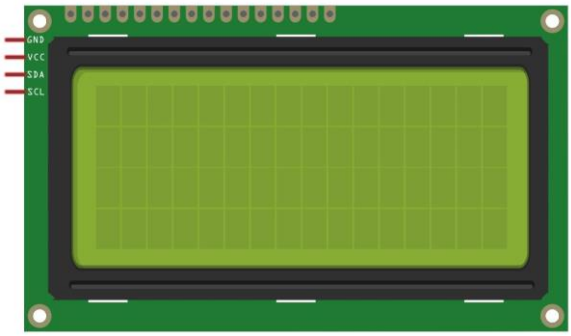
LCD2004 communication

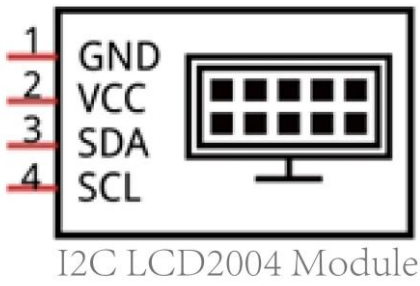
The LCD2004 Display Screen can display 4 lines of characters in 20 columns. It is capable of displaying numbers, letters, symbols, ASCII code and so on. As shown below is a monochrome LCD2004 Display Screen along with its circuit pin diagram.





I2C LCD2004 Display Screen integrates an I2C interface, which connects the serial-input & parallel-output module to the LCD2004 Display Screen. This allows us to use only 4 lines to the operate the LCD2004.

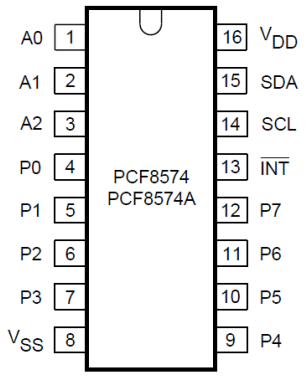




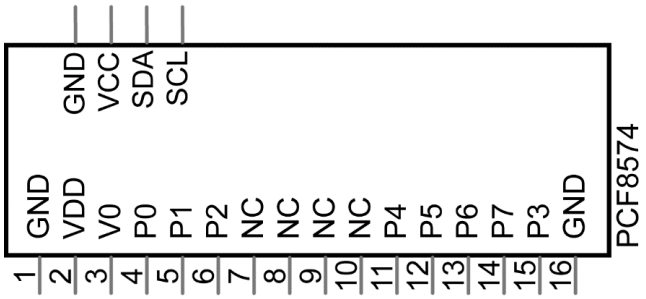
The serial-to-parallel IC chip used in this module is PCF8574T (PCF8574AT), and its default I2C address is 0x27(0x3F).

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

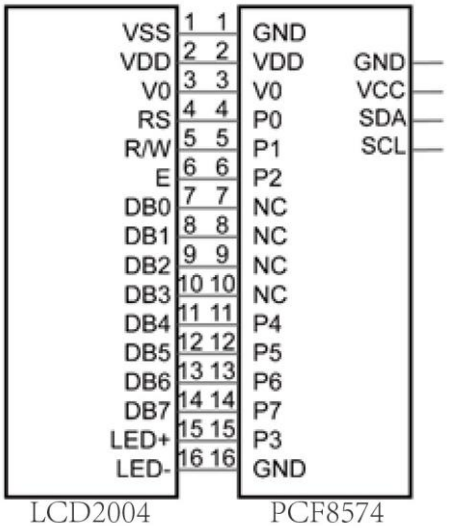
PCF8574 chip pin diagram



PCF8574 module pin diagram



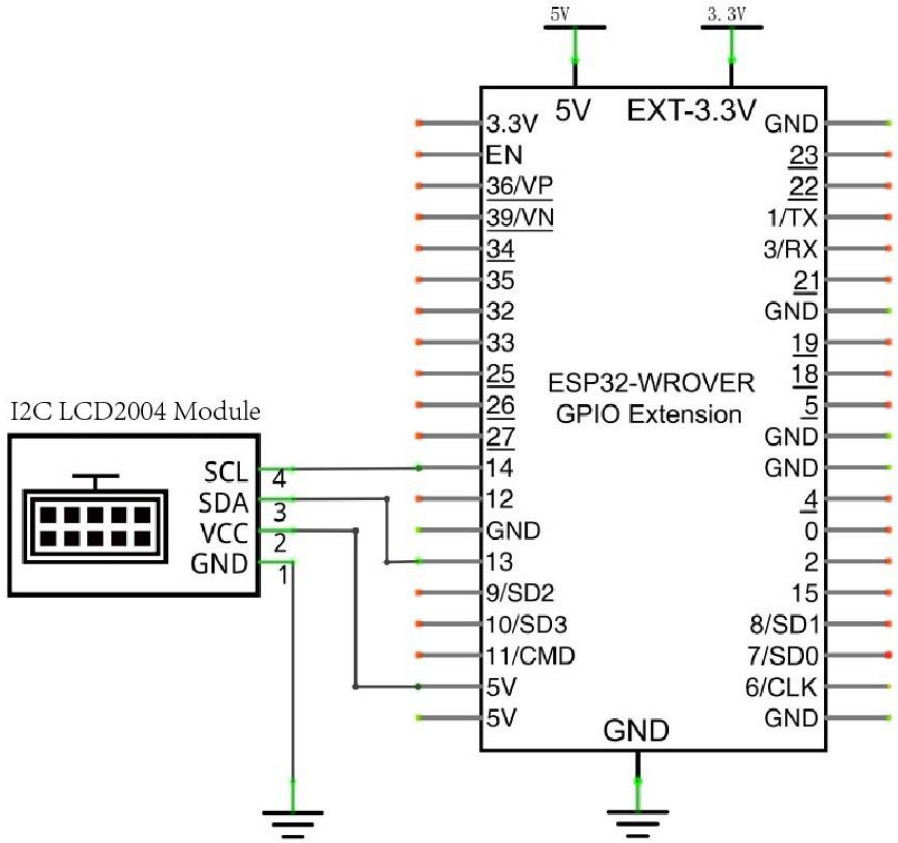
PCF8574 module pin and LCD2004 pin are corresponding to each other and connected with each other:



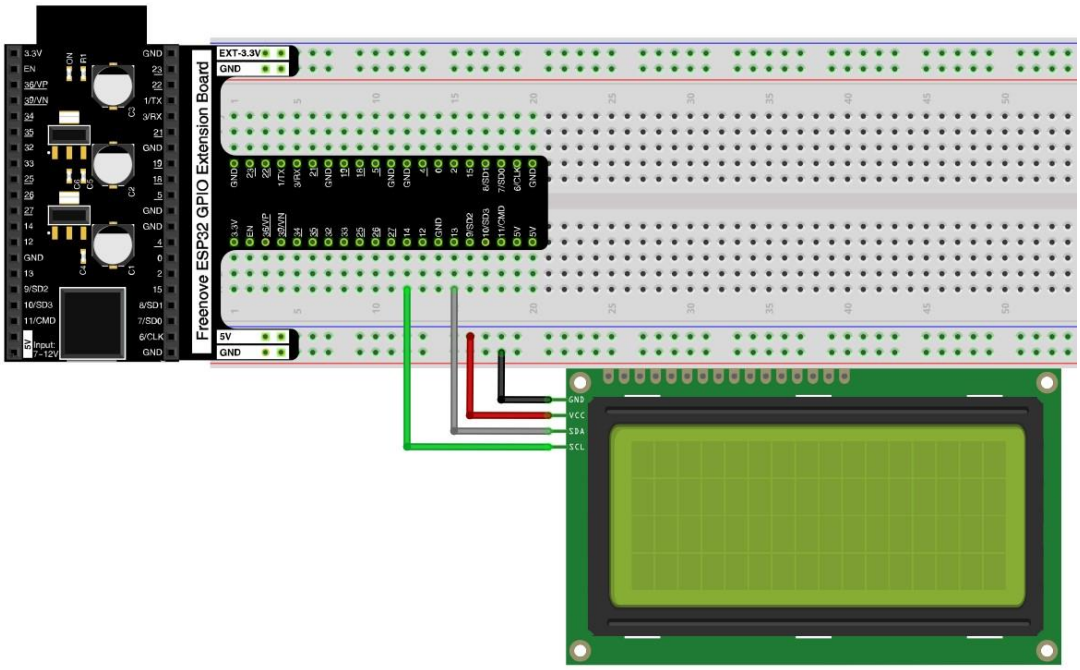
So we only need 4 pins to control the 16 pins of the LCD2004 display screen through the I2C interface.
In this project, we will use the I2C LCD2004 to display some static characters and dynamic variables.

Circuit

Schematic diagram



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

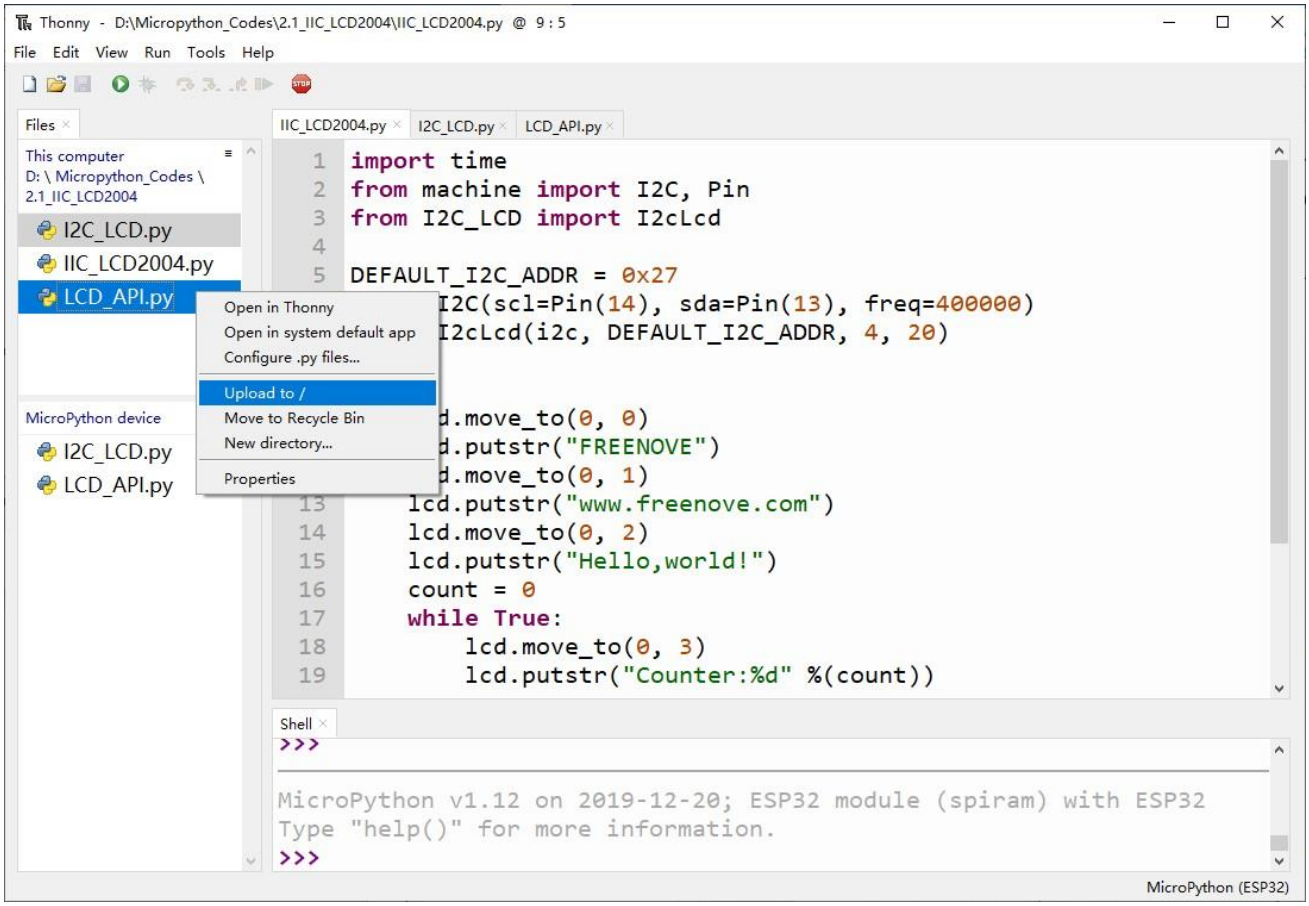


Code

Move the program folder "**Freenove_LCD_Module/Freenove_LCD_Module_for_ESP32/Python/Python_Codes**" to disk(D) in advance with the path of "**D:/Micropython_Codes**".

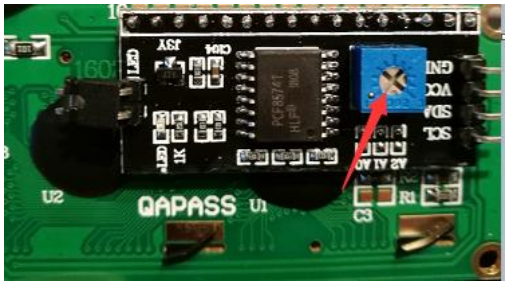
Open "Thonny", click "This computer" → "D:" → "Micropython_Codes" → "2.1_I2C_LCD2004". Select "I2C_LCD.py"and "LCD_API.py", right click your mouse to select "Upload to /", wait for "I2C_LCD.py" and "LCD_API.py" to be uploaded to ESP32-WROVER and then double click "I2C_LCD2004.py".

2.1_I2C_LCD2004



Click "Run current script" and LCD2004 displays some characters.Press Ctrl+C or click "Stop/Restart backend" to exit the program.

Note: If you cannot see anything on the display or the display is not clear, try rotating the white knob on back of LCD2004 slowly, which adjusts the contrast, until the screen can display clearly.



The following is the program code:

```
import time
from machine import I2C, Pin
from I2C_LCD import I2cLcd

DEFAULT_I2C_ADDR = 0x27
i2c = I2C(scl=Pin(14), sda=Pin(13), freq=400000)
lcd = I2cLcd(i2c, DEFAULT_I2C_ADDR, 4, 20)

try:
    lcd.move_to(0, 0)
    lcd.putstr("FREENOVE")
    lcd.move_to(0, 1)
    lcd.putstr("www.freenove.com")
    lcd.move_to(0, 2)
    lcd.putstr("Hello,world!")
    count = 0
    while True:
        lcd.move_to(0, 3)
        lcd.putstr("Counter:%d" %(count))
        time.sleep_ms(1000)
        count += 1
except:
    pass
```

Import time, I2C and I2C_LCD modules.

```
import time
from machine import I2C, Pin
from I2C_LCD import I2cLcd
```

Instantiate the I2C LCD2004 screen. It should be noted here that if your LCD driver chip uses PCF8574T, set the I2C address to 0x27, and if uses PCF8574AT, set the I2C address to 0x3F.

```
DEFAULT_I2C_ADDR = 0x27
```

Initialize I2C pins and associate them with I2CLCD module, and then set the number of rows and columns for LCD2004.

```
i2c = I2C(scl=Pin(14), sda=Pin(13), freq=400000)
lcd = I2cLcd(i2c, DEFAULT_I2C_ADDR, 4, 20)
```

Move the cursor of LCD2004 to the third row, first column, and print out "Hello, world!"

```
14      lcd.move_to(0, 2)
15      lcd.putstr("Hello,world!")
```

The fourth line of LCD2004 continuously prints the number of seconds after the ESP32 program runs.

```
17      while True:
18          lcd.move_to(0, 3)
19          lcd.putstr("Counter:%d" %(count))
20          time.sleep_ms(1000)
21          count += 1
```

Reference

Class I2cLcd

Before each use of the object **I2cLcd**, please make sure that **I2C_LCD.py** and **LCD_API.py** have been uploaded to "/" of ESP32, and then add the statement "**from I2C_LCD import I2cLcd**" to the top of the python file.

clear(): Clear the LCD2004 screen display.

show_cursor(): Show the cursor of LCD2004.

hide_cursor(): Hide the cursor of LCD2004.

blink_cursor_on(): Turn on cursor blinking.

blink_cursor_off(): Turn off cursor blinking.

display_on(): Turn on the display function of LCD2004.

display_off(): Turn on the display function of LCD2004.

backlight_on(): Turn on the backlight of LCD2004.

backlight_off(): Turn on the backlight of LCD2004.

move_to(cursor_x, cursor_y): Move the cursor to a specified position.

cursor_x: Column cursor_x

cursor_y: Row cursor_y

putchar(char): Print the character in the bracket on LCD2004.

putstr(string): Print the string in the bracket on LCD2004.

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 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/

What's next?(others)

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 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.