

ESP32-S3-GEEK

From Waveshare Wiki

Jump to: navigation, search

Overview

Introduction

ESP32-S3-GEEK is a geek development board designed by Waveshare, onboard USB-A port, 1.14-inch LCD screen, a TF card slot, and other peripherals. Supports 2.4GHz WiFi and BLE 5, with integrated 16MB Flash & 2MB PSRAM, provides I2C port, UART port, and GPIO header, bringing more possibilities for your project.

Features

- Adopts **ESP32-S3R2** chip with Xtensa® 32-bit LX7 dual-core processor, capable of running at **240 MHz**.
- Built-in 512KB SRAM, 384KB ROM, 2MB of on-chip PSRAM, and onboard **16MB Flash memory**.
- Onboard 1.14inch 240×135 pixels 65K color **IPS LCD** display.
- Integrated **2.4GHz WiFi and Bluetooth LE** wireless communication.
 - WiFi supports Infrastructure BSS in Station, SoftAP, and Station + SoftAP modes.
 - WiFi supports 1T1R mode with a data rate of up to 150 Mbps.
 - Bluetooth supports high power mode (20dBm).
 - Internal co-existence mechanism between Wi-Fi and Bluetooth to share the same antenna.
- The onboard 3-pin **UART** interface can function as a USB-to-serial adapter.
- The onboard 3-pin **GPIO** interface can be used for debugging other modules or for ADC data collection.
- The onboard 4-pin **I2C** interface is available for testing the target board.
- Equipped with plastic case and cables.



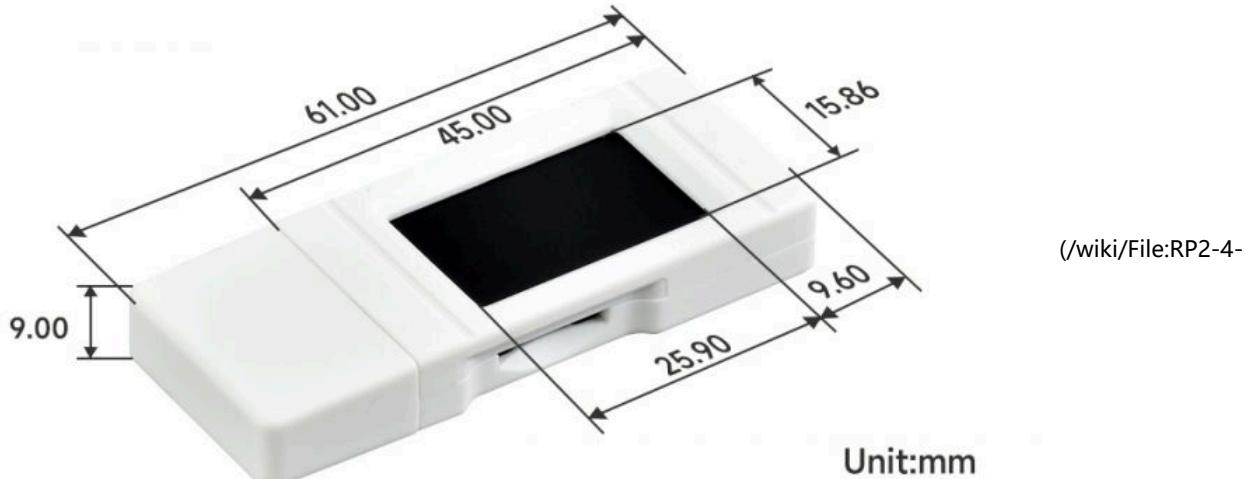
ESP32-S3-GEEK

(<https://www.waveshare.com/esp32-s3-geek.htm>)

ESP32-S3
I2C, UART, USB-A

- Provides online open-source demos and resources, more convenient for learning and development.

Dimensions



Development Environment Setup

- The following development system is Windows by default.

ESP-IDF

[\[Collapse\]](#)

- It is recommended to develop with the VSC plug-in.

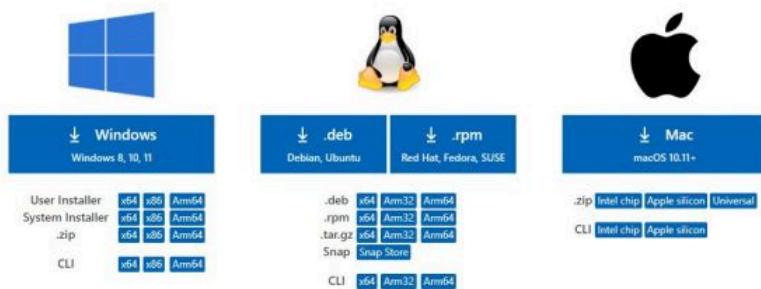
Develop with VSCode Plug-in

Install VSCode

- Open the download page (<https://code.visualstudio.com/download>) of the official VSCode website, and select the corresponding system and system bit to download.

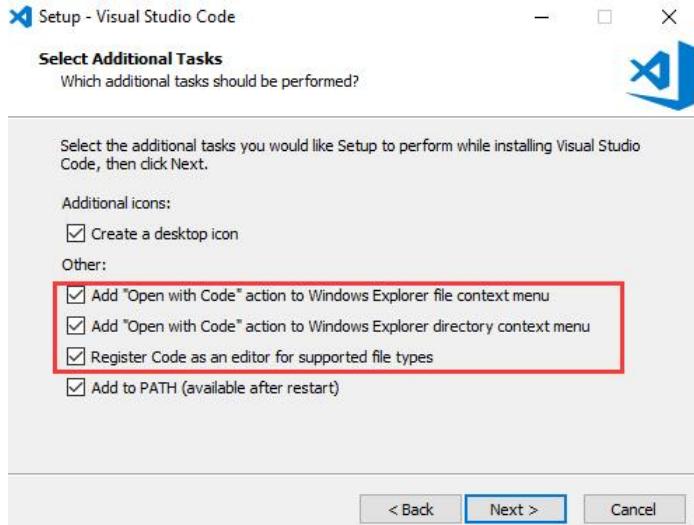
Download Visual Studio Code

Free and built on open source. Integrated Git, debugging and extensions.



(/wiki/File:ESP32-S3-Pico_05.jpg)

- After running the installation package, the rest can be installed by default, but here for the subsequent experience, it is recommended to check boxes 1, 2, and 3.



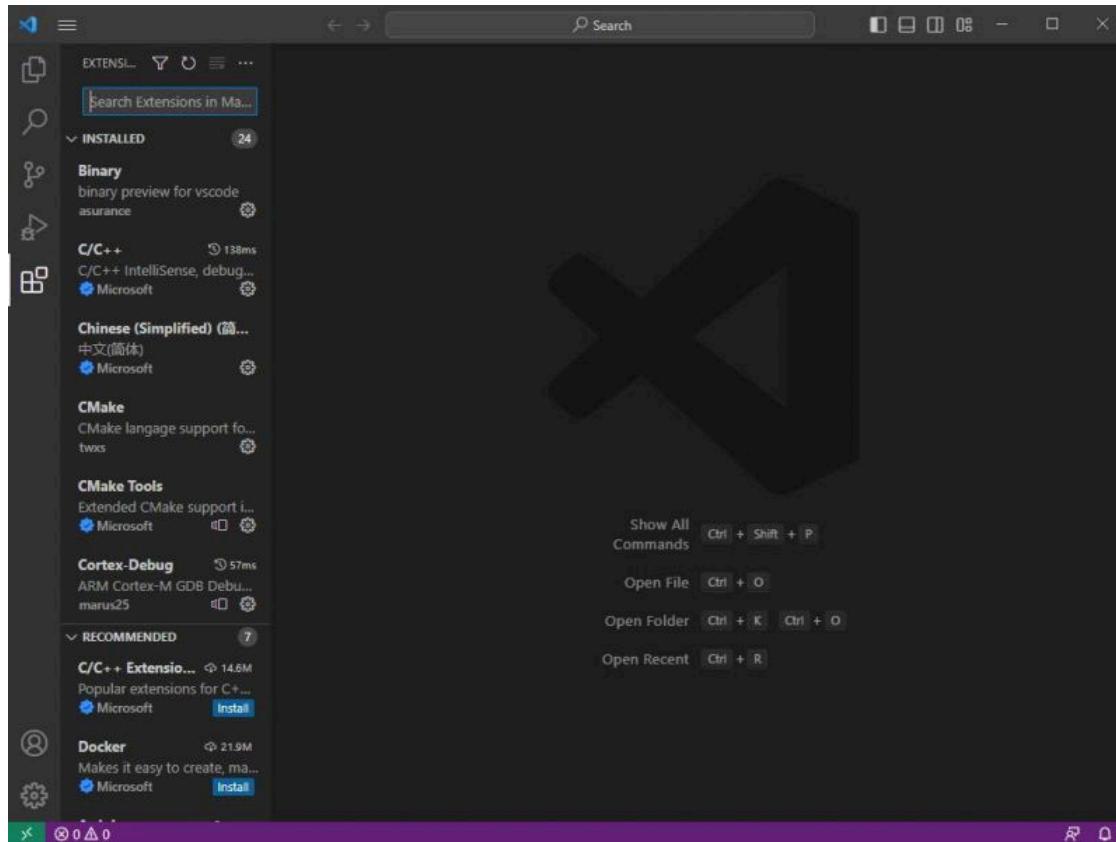
(/wiki/File:ESP32-S3-Pico_06.jpg)

- After the first and second items are enabled, you can open VSCode directly by right-clicking files or directories, which can improve the subsequent user experience.
- After the third item is enabled, you can select VSCode directly when you choose how to open it.

Install Espressif IDF Plug-in

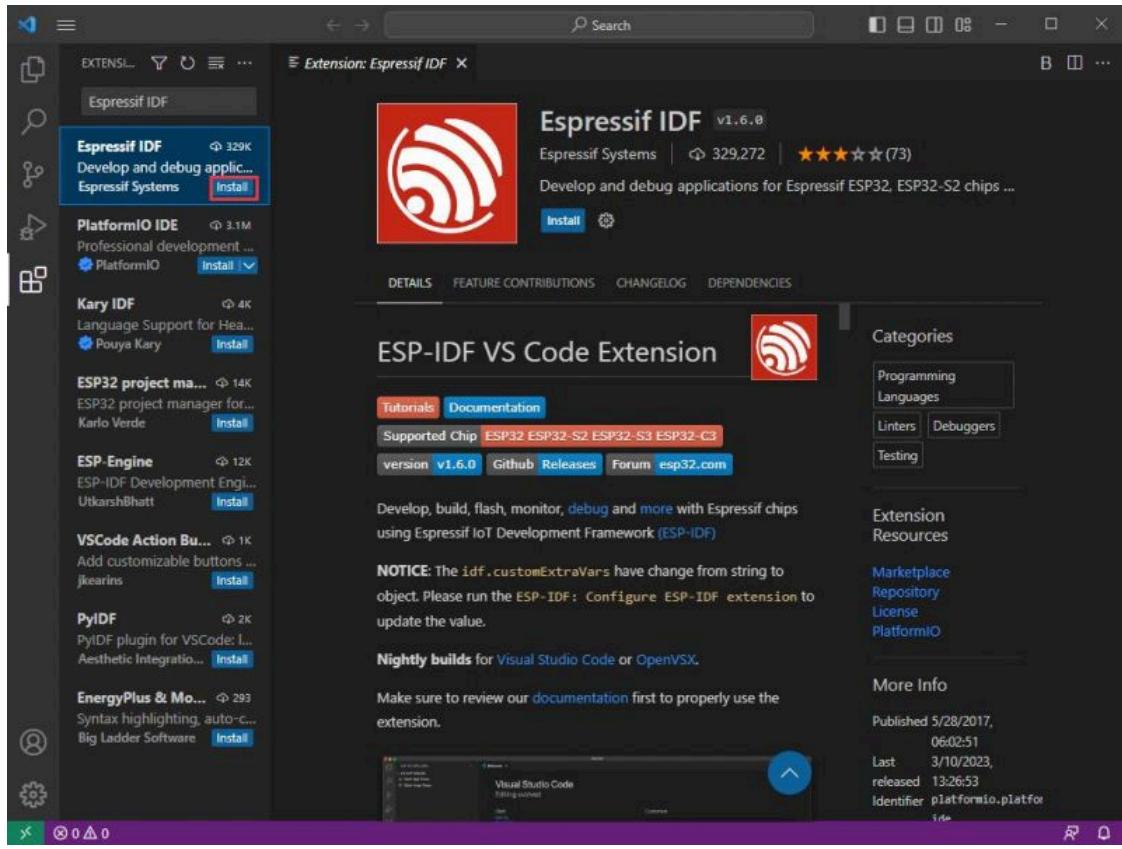
- Note: The latest version of the current plug-in is V1.6.0, for a consistent experience, users can choose the same version as us.

1. Open VSCode and use the shortcut key Shift + Ctrl + X to enter the plugin manager.



(/wiki/File:ESP32-S3-

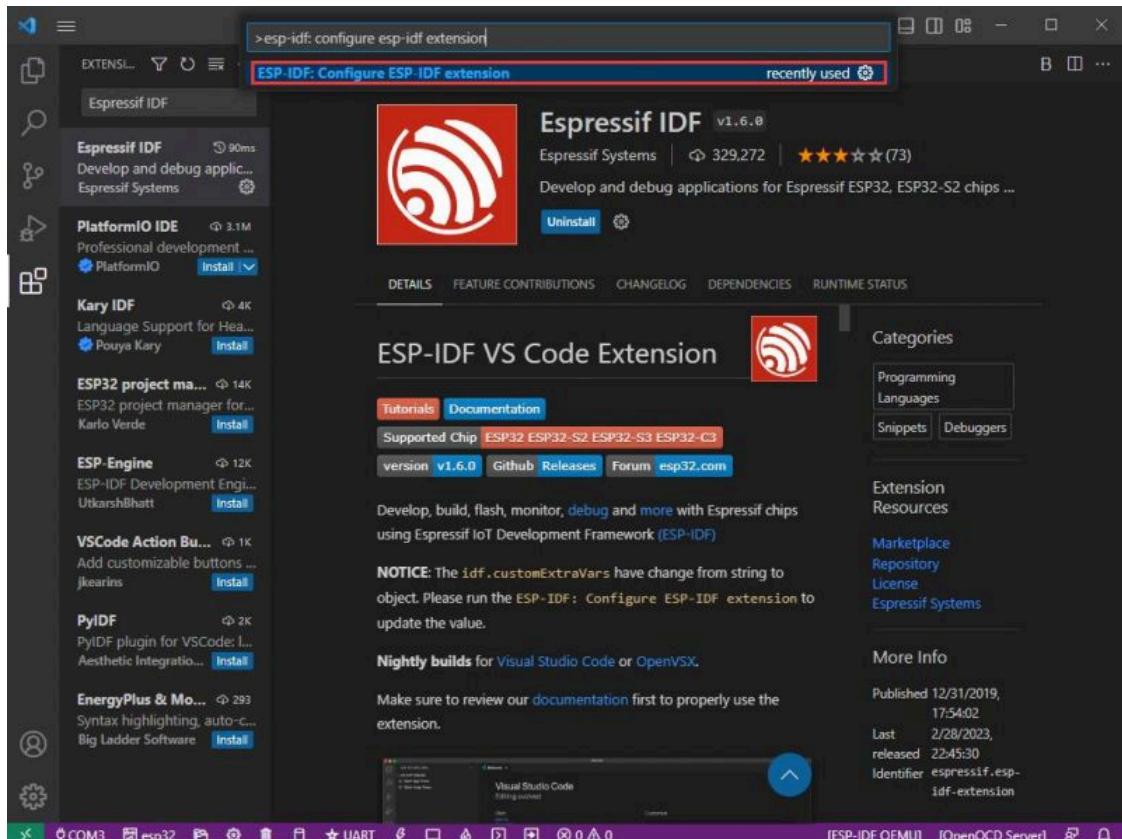
2. In the search bar, type Espressif IDF, select the corresponding plug-in, and click install.



Pico_08.jpg)

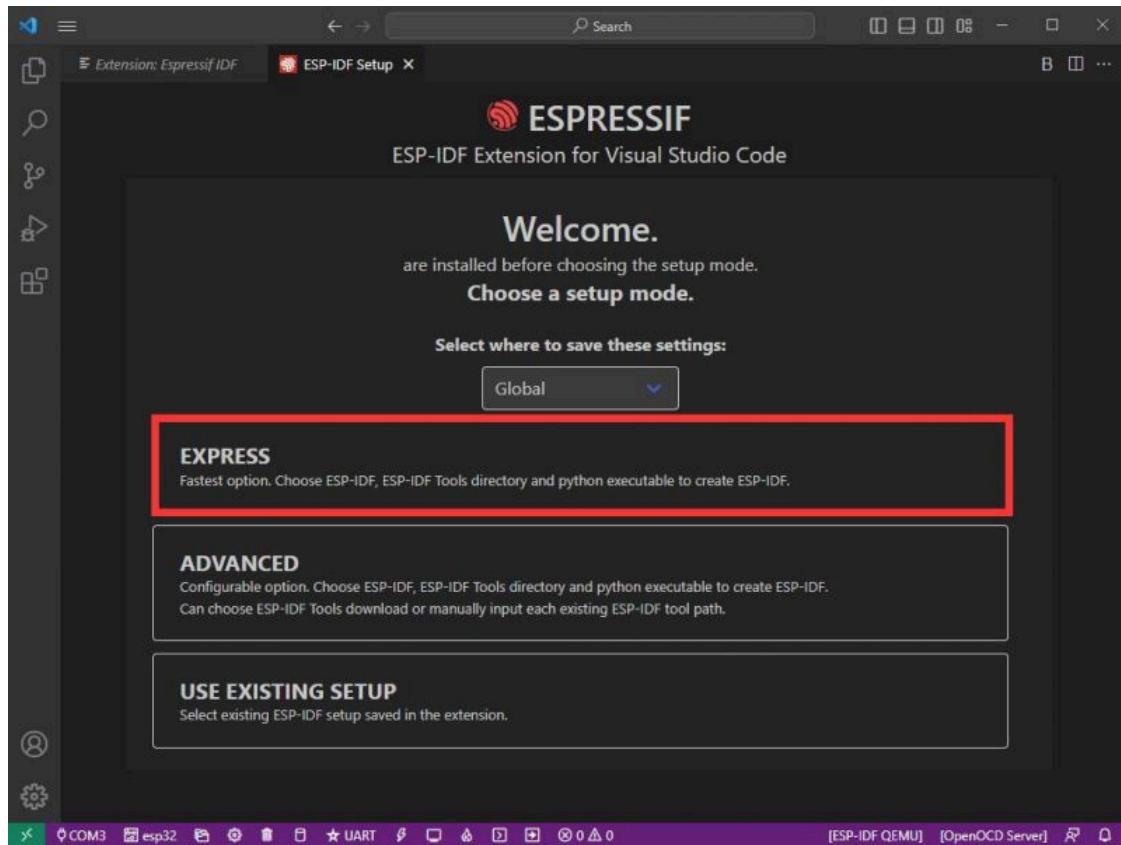
3. Press F1 to enter:

```
esp-idf: configure esp-idf extension
```



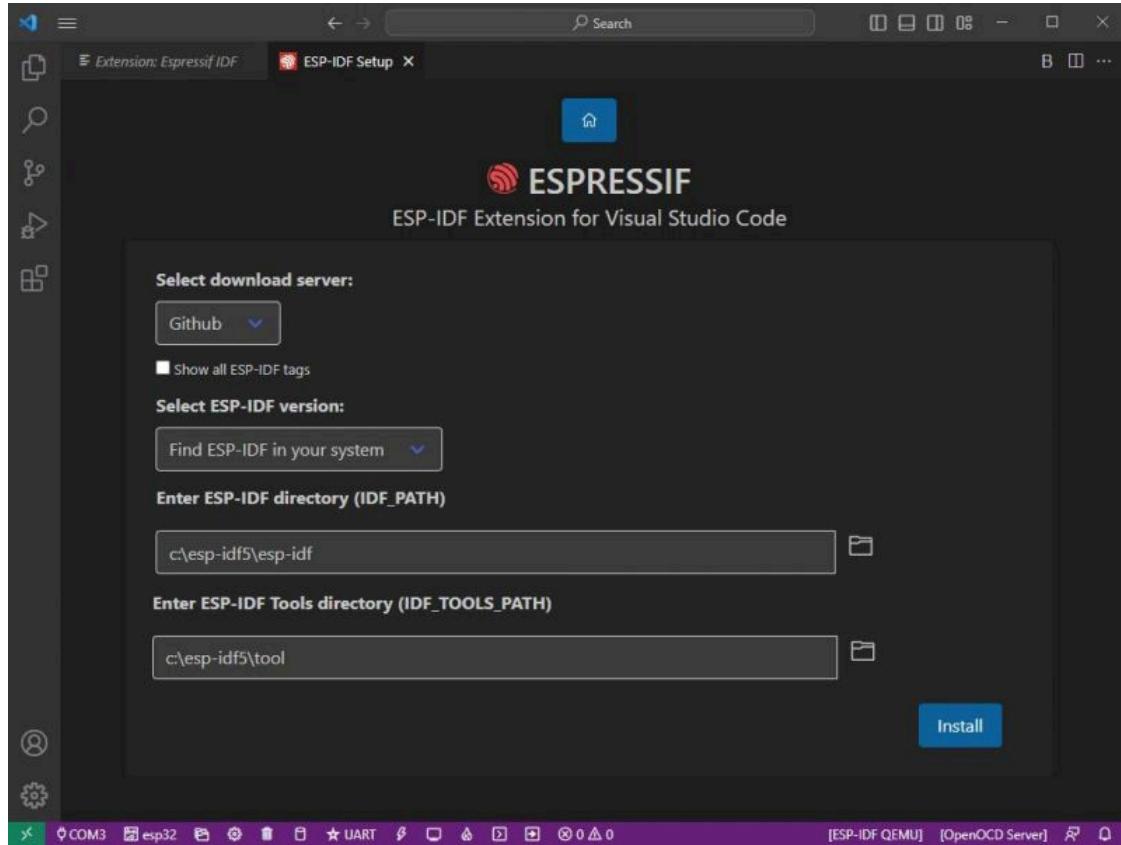
Pico_09.jpg)

4. Choose express (This tutorial is for first-time users, so only the first general installation tutorial is covered.)



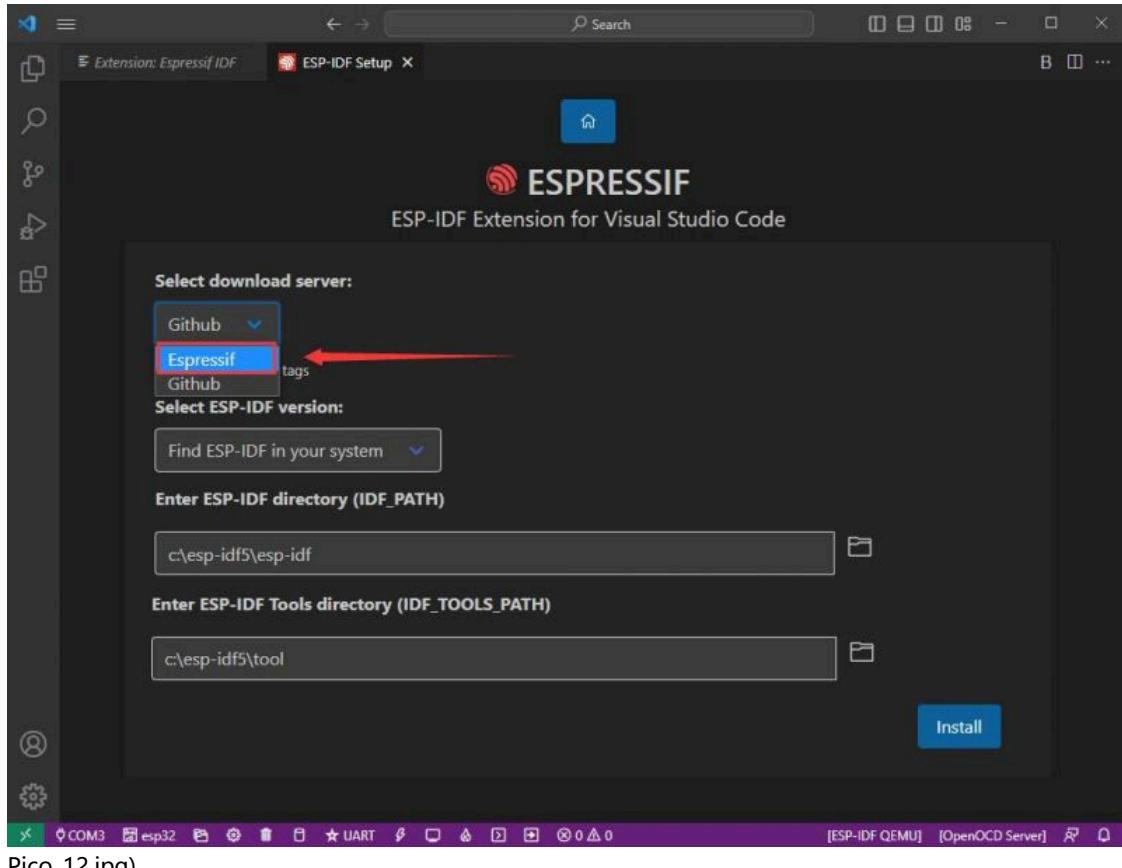
Pico_10.jpg)

5. Open and display this screen.



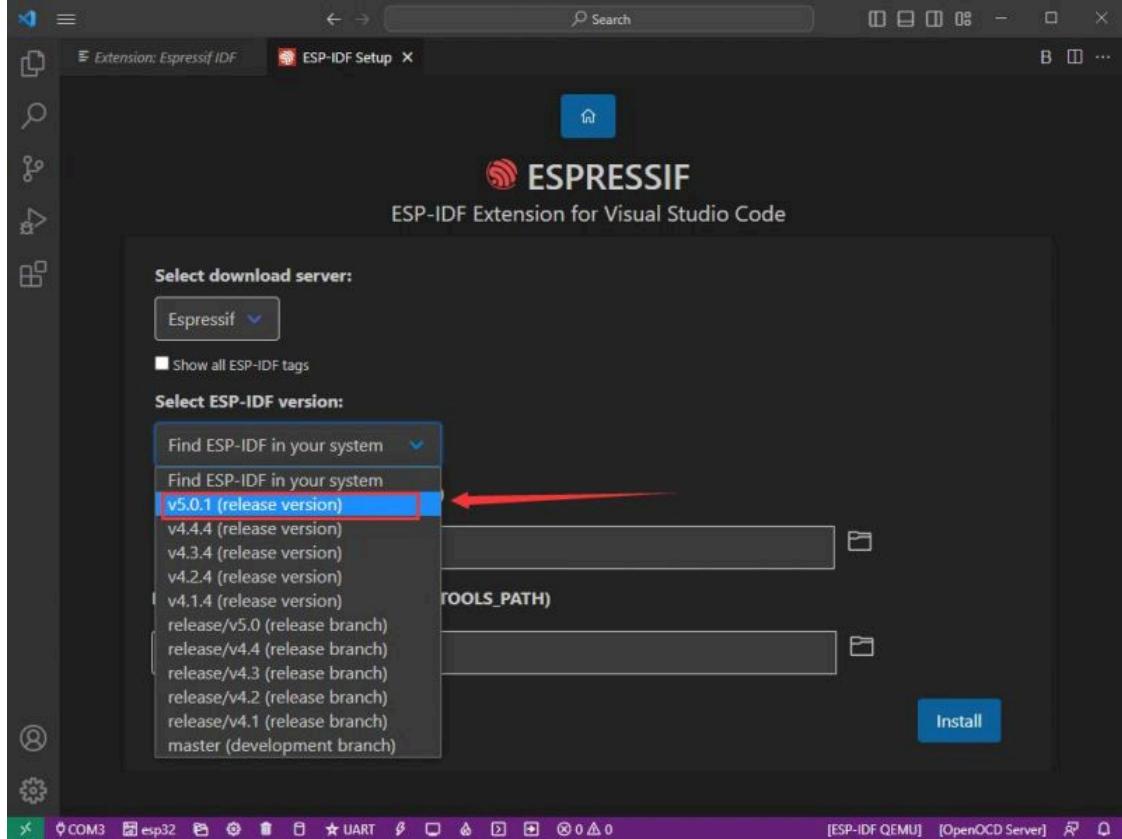
Pico_11.jpg)

6. Choose a server to download.



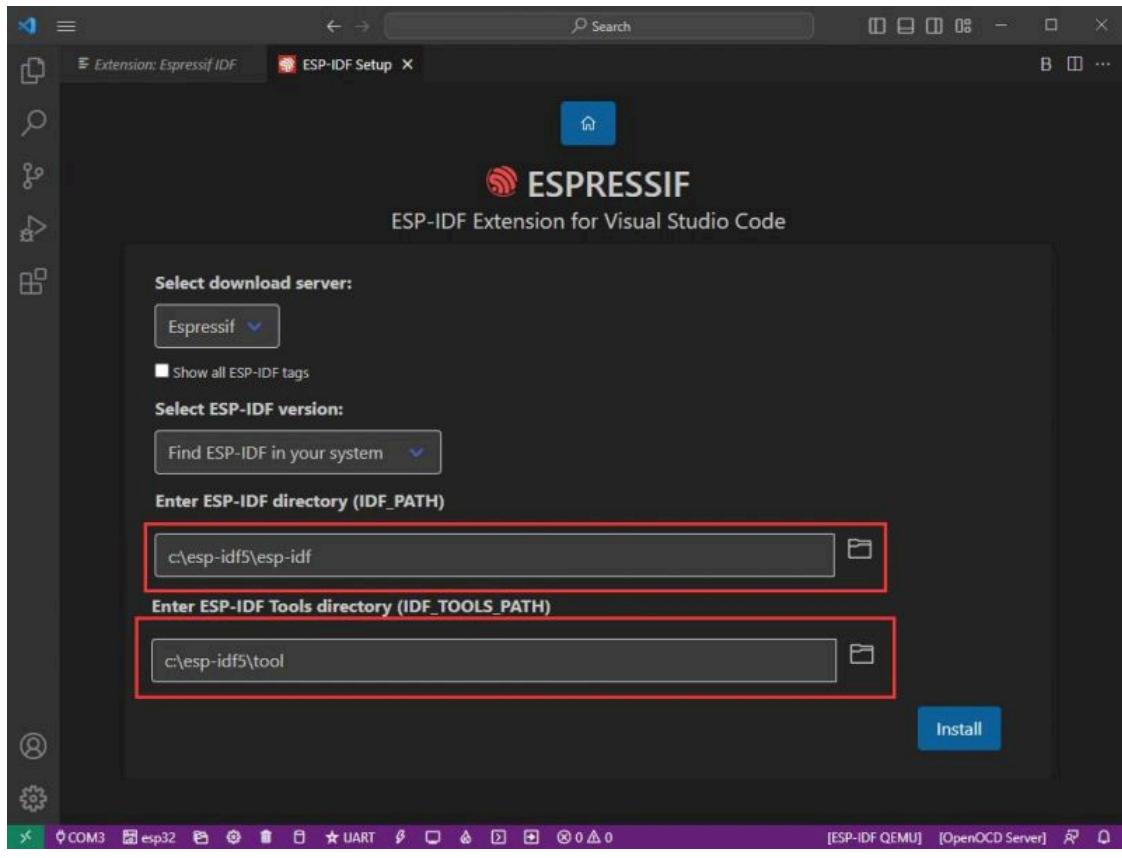
Pico_12.jpg

7. Select the ESP-IDF version you want now, we choose the latest V5.0.1 (note that ESP-IDF started to support ESP32-S3 only after V4.4).



Pico_13.jpg

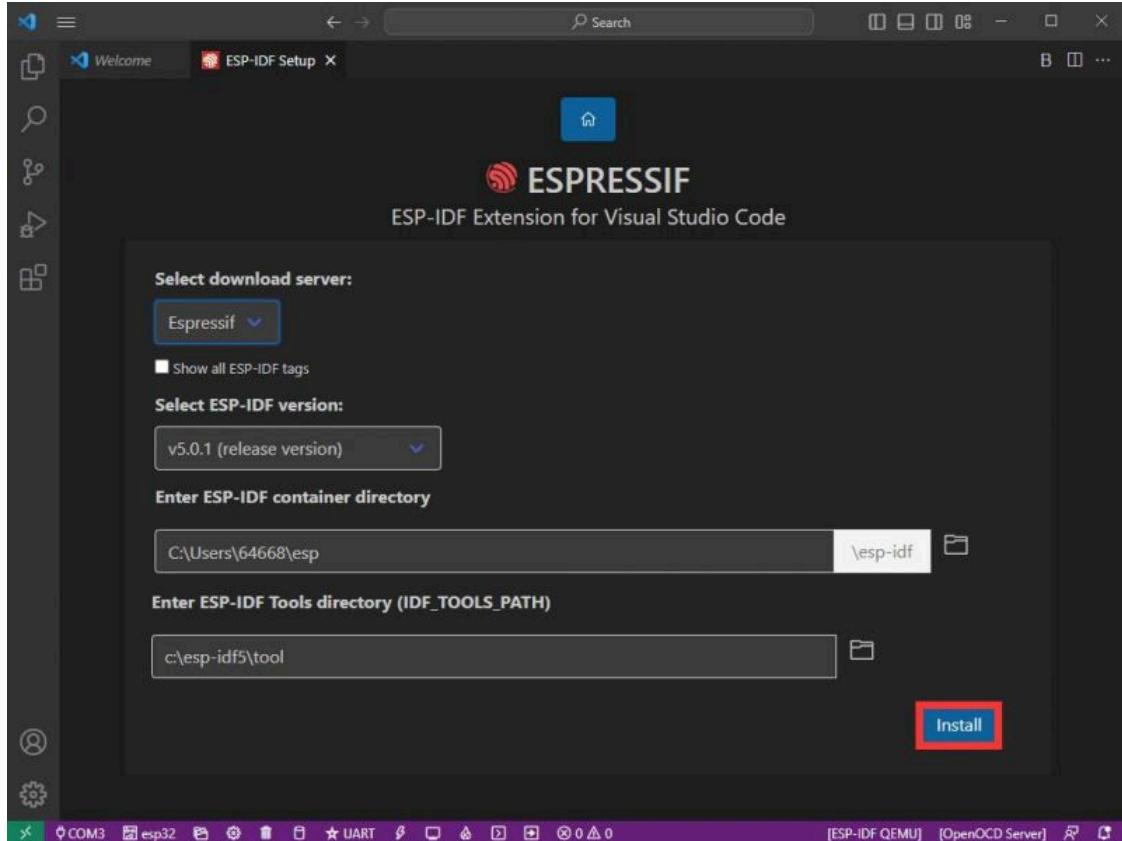
8. The following two are the ESP-IDF directory installation address and the ESP-IDF required tools installation address respectively.



Pico_14.jpg)

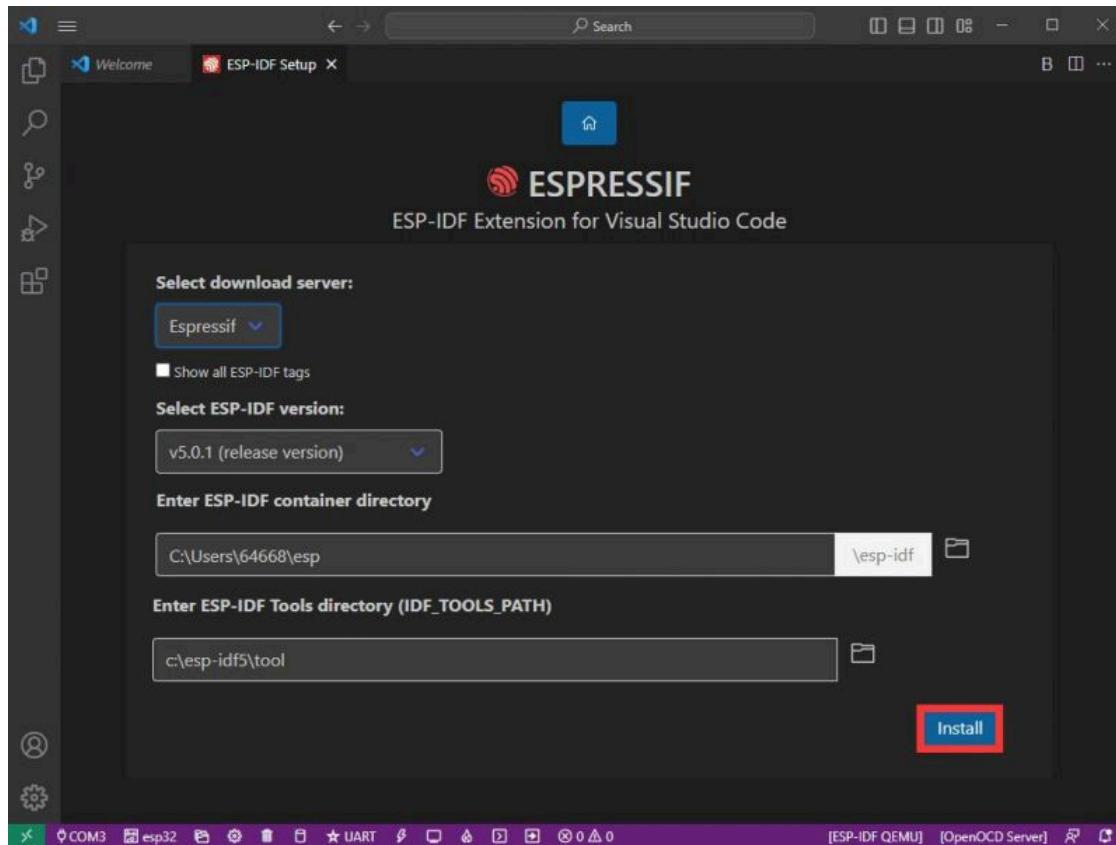
- Note: If you have installed ESP-IDF before, or if it has failed, please make sure to delete the file completely or create a new path.

9. Once the configuration is finished, click "install" to download.

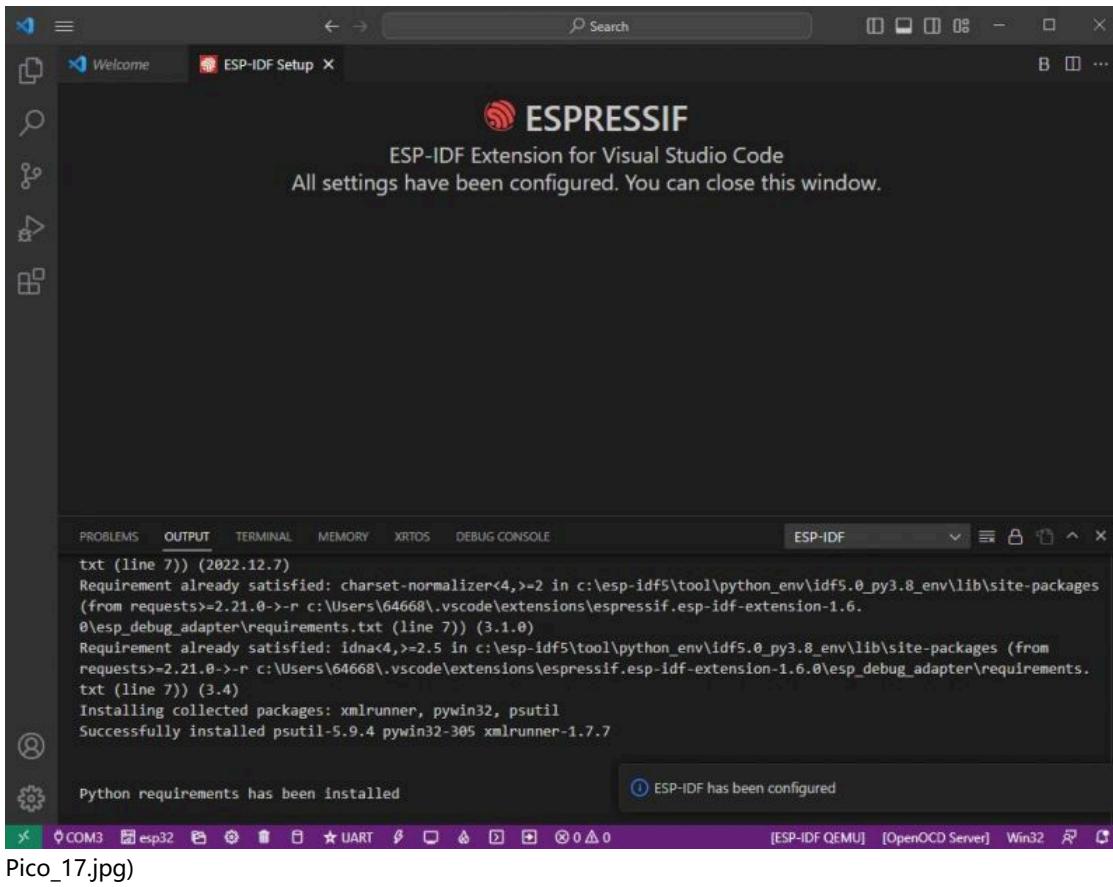


Pico_15.jpg)

10. Enter the download page, and it will automatically install the corresponding tools and environment, just wait a moment.



11. After the installation is completed, it will enter the following screen, indicating that the installation is finished.



Pico_17.jpg

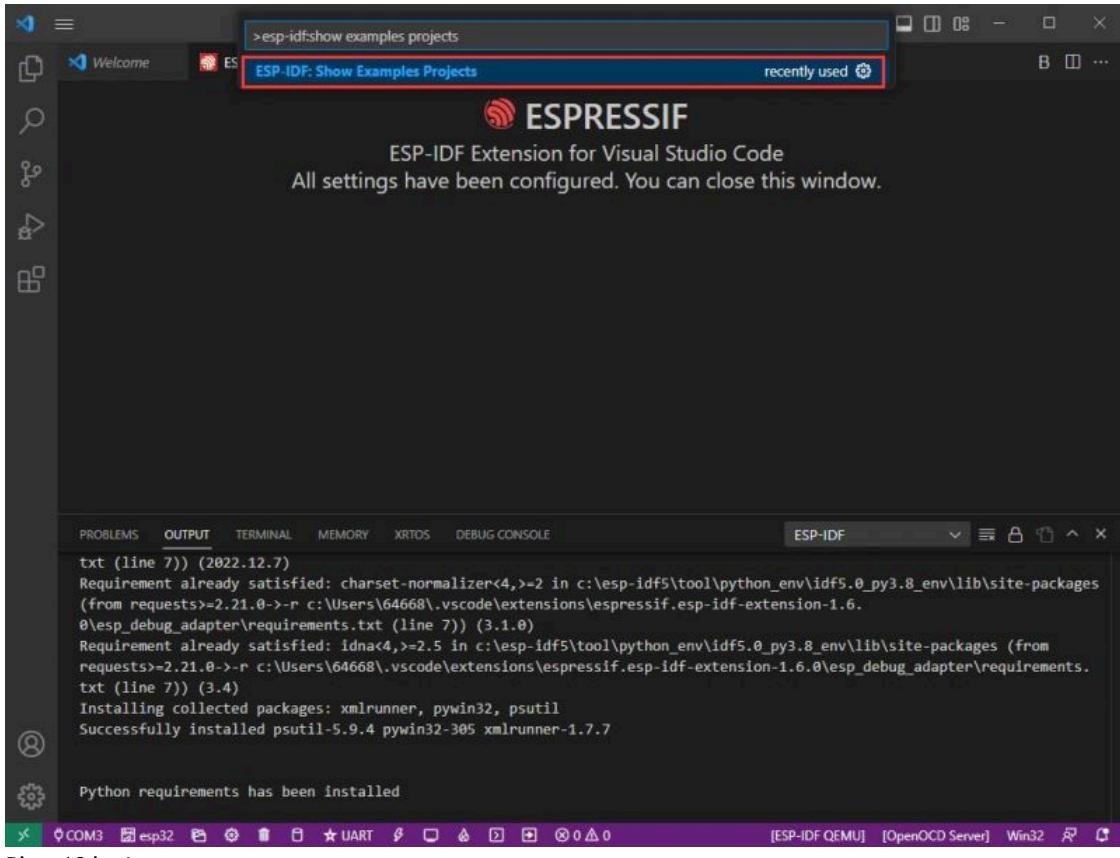
Official Demo Usage

- Click here (<https://github.com/espressif/esp-idf/tree/master/examples>) to view more details provided by the official ESP.

Creating a Demo

- Using the shortcut F1, type:

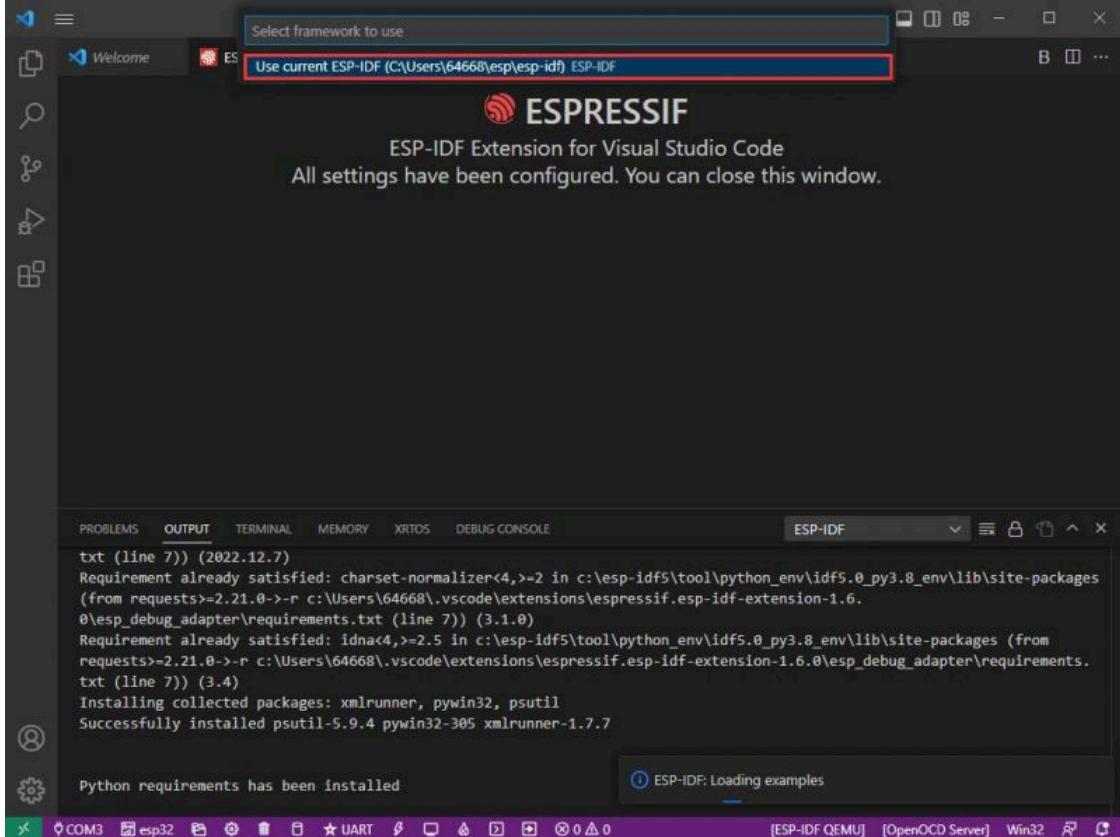
```
esp-idf:show examples projects
```



(/wiki/File:ESP32-S3-

Pico_18.jpg

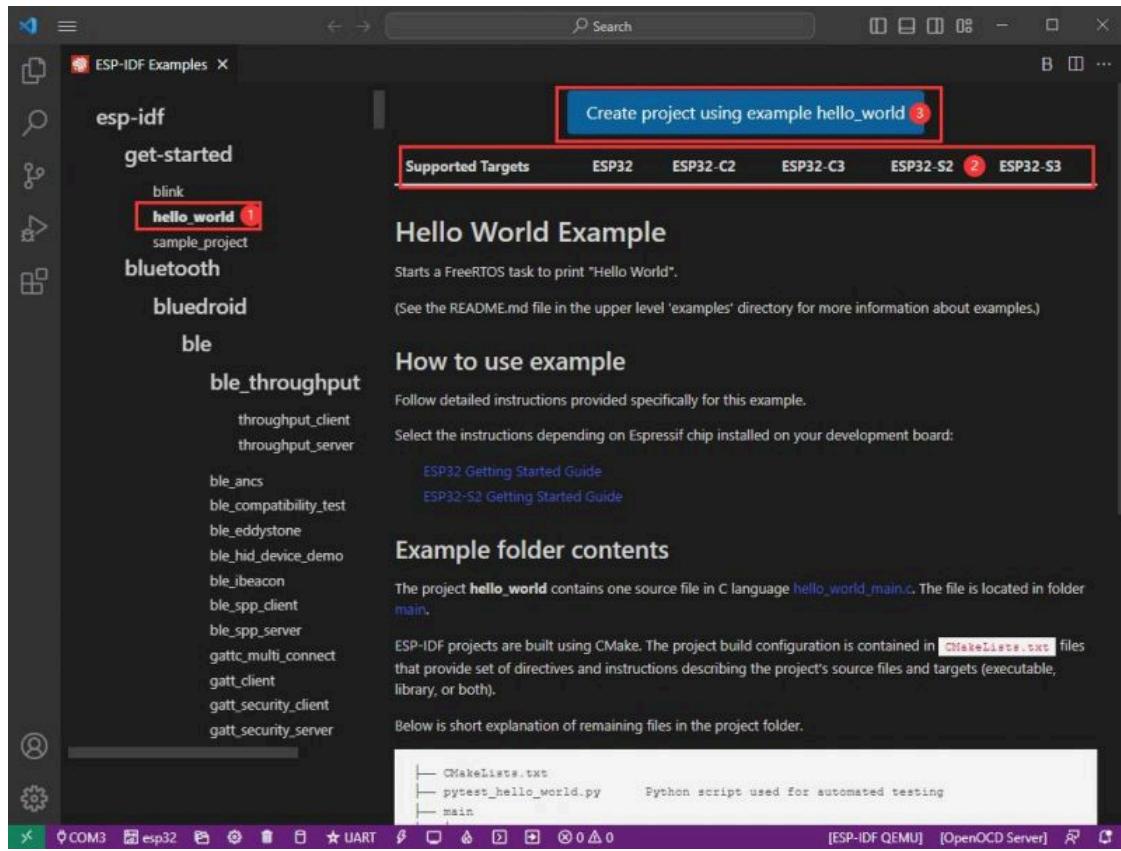
2. Choose your current IDF version:



(/wiki/File:ESP32-S3-

Pico_19.jpg

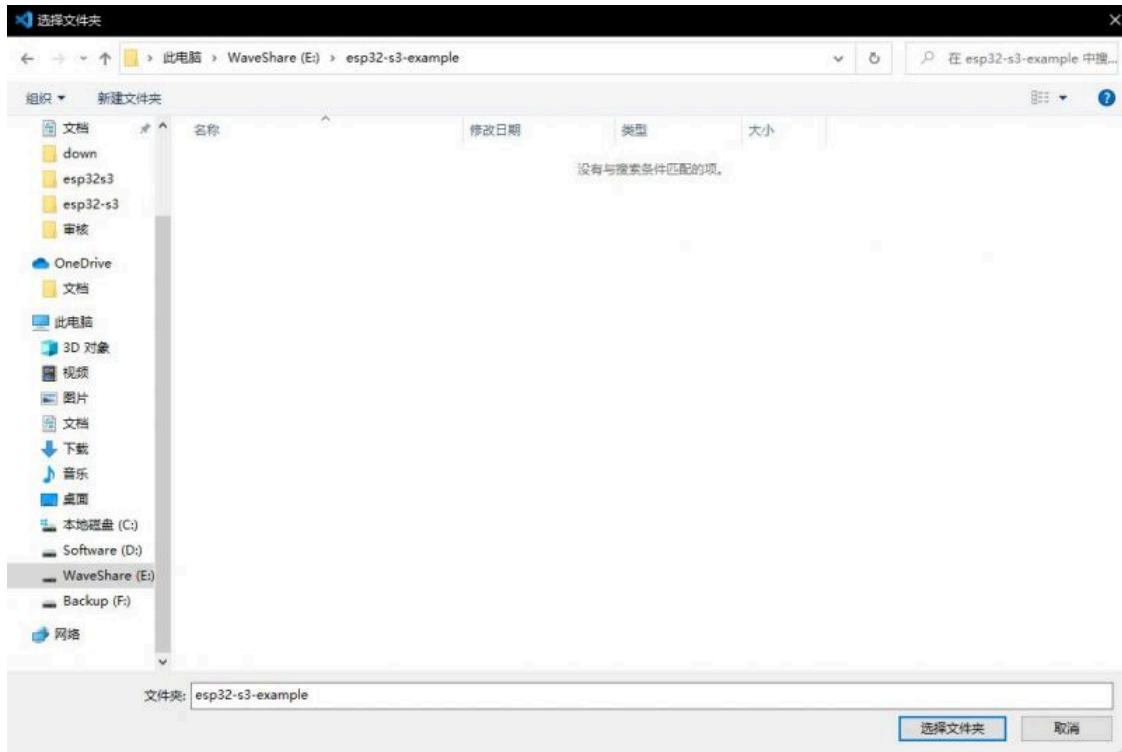
3. Take "Hello World" as an example:



(/wiki/File:ESP32-S3-

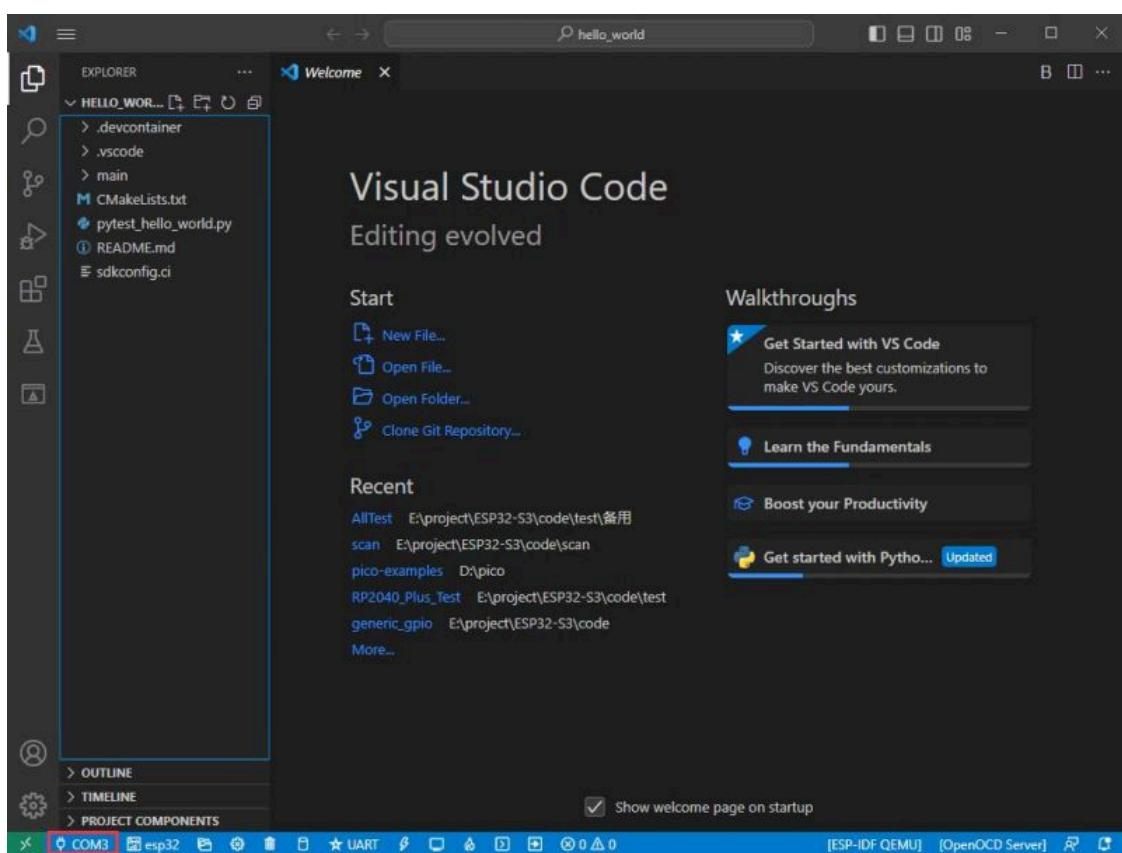
Pico_20.jpg)

4. Choose the corresponding demo.
5. The readme file will explain which chip the demo is suitable for (the following section will introduce how to use the demo and its file structure, which is omitted here).
6. Click to create the demo.
7. Choose the path to place the demo and ensure that there is no folder with the same name as the demo.

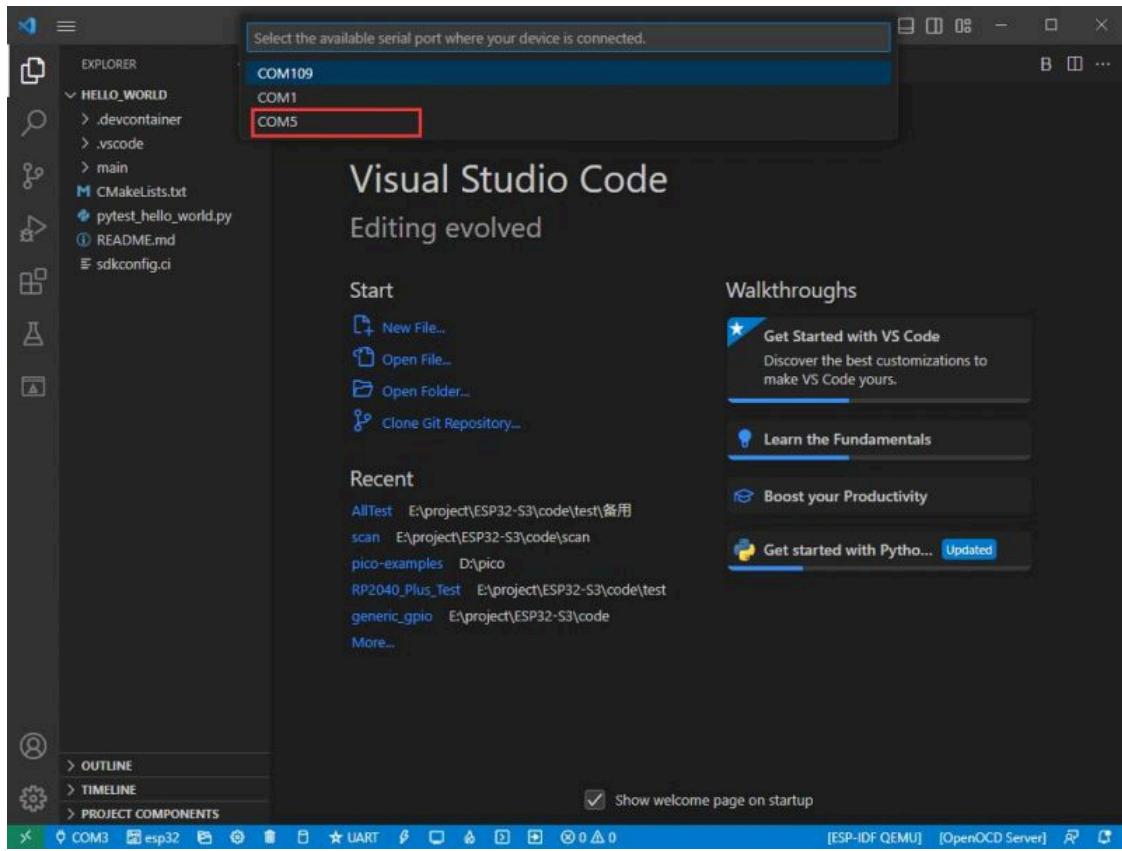


Modify COM Port

1. The corresponding COM port is displayed here, click on it to modify.

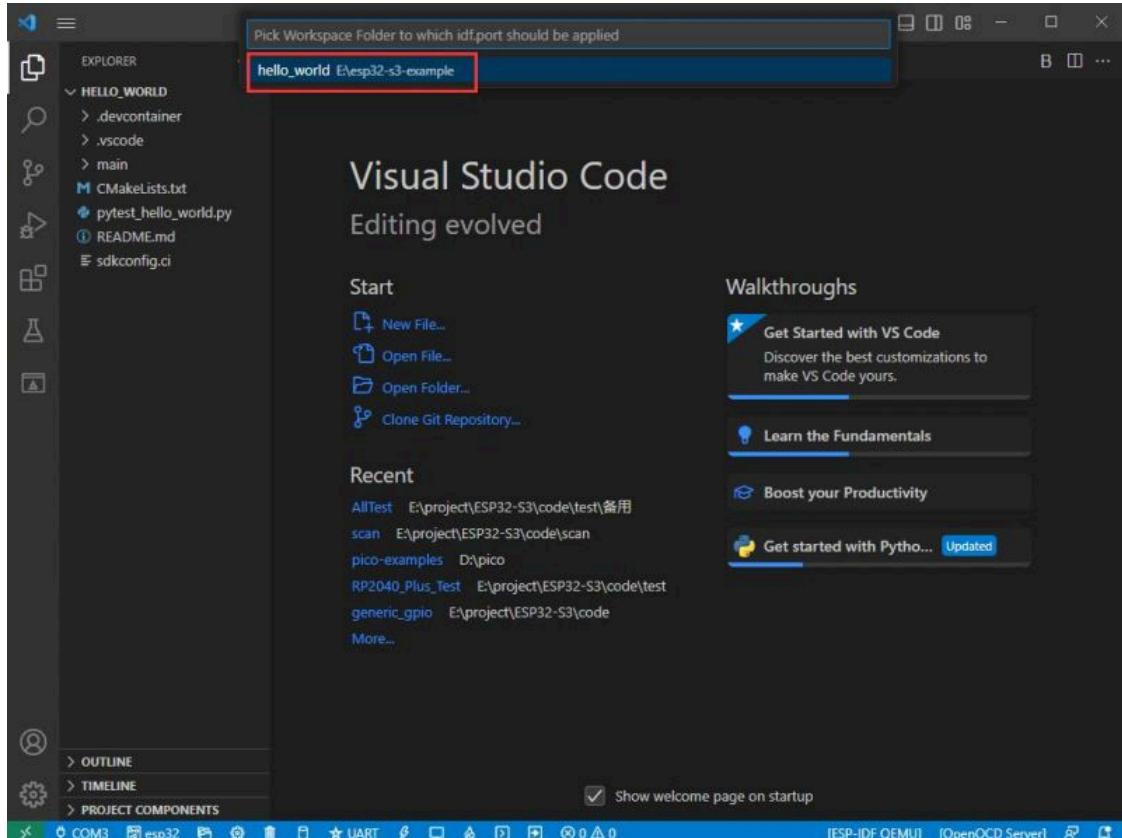


2. We check the device manager COM port, and select COM5, please select your corresponding COM port:



Pico_23.jpg

3. Choose the project and demo.

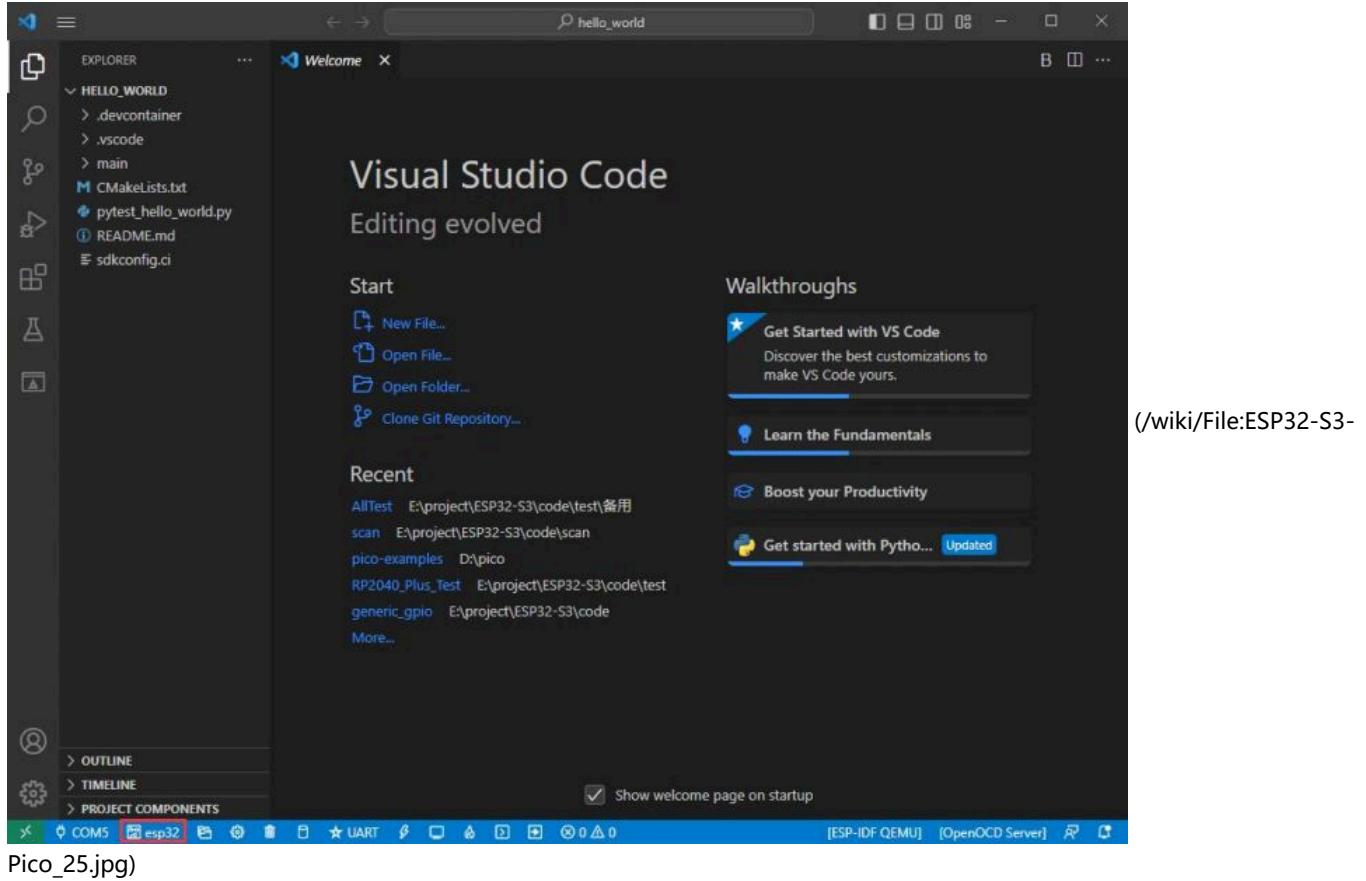


Pico_24.jpg

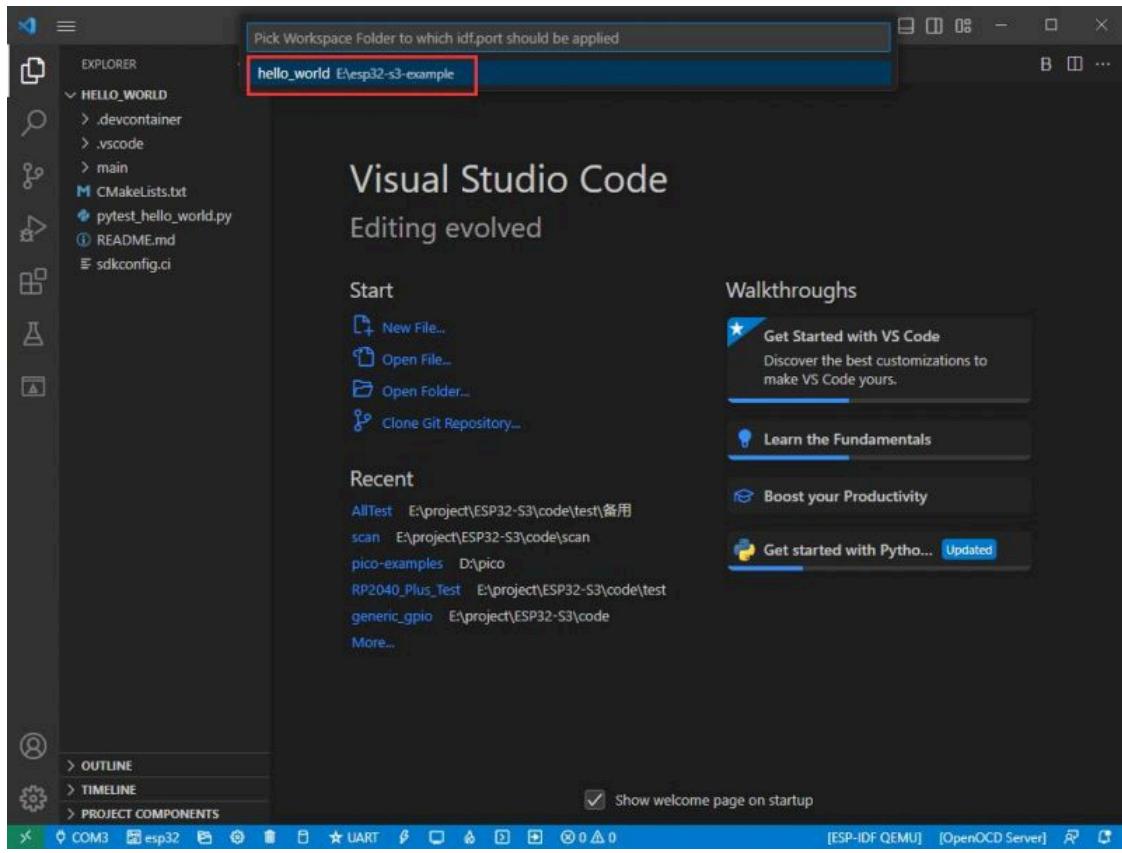
4. Then the COM port is modified.

Modify the Driver

1. Here shows the driver used, click here to modify the corresponding driver:

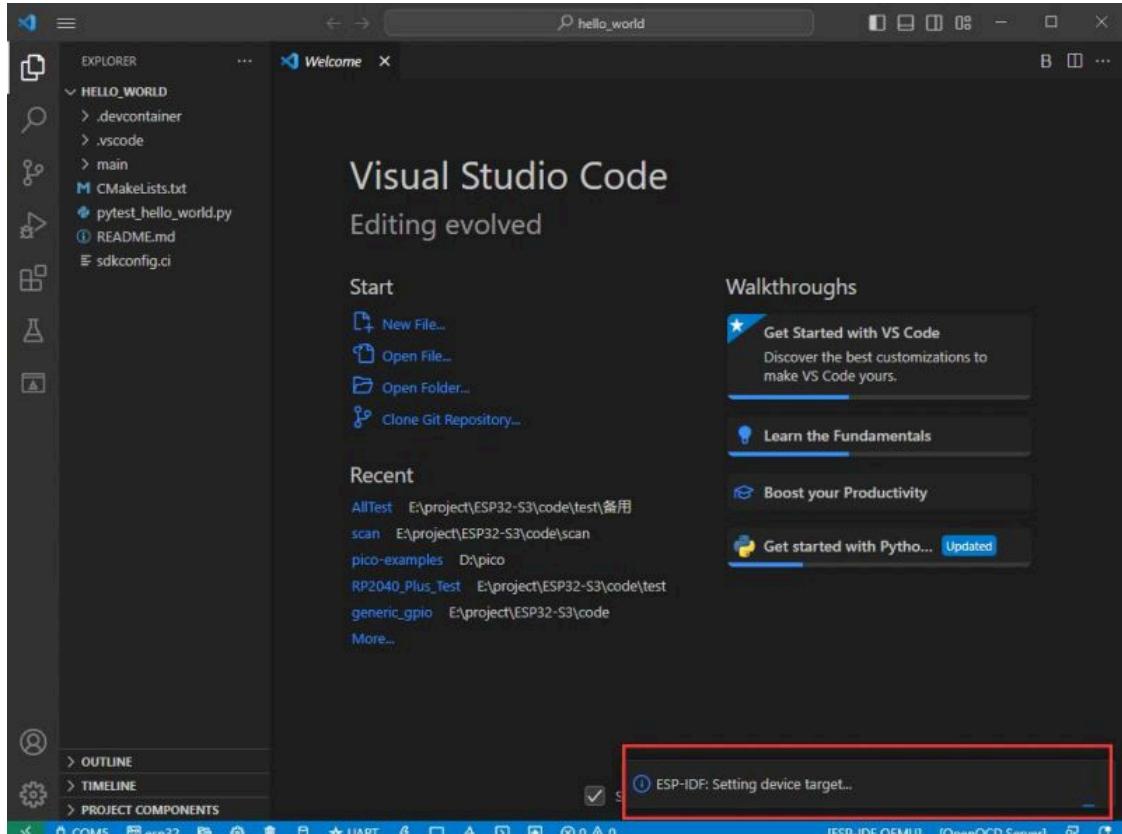


2. Choose the project or demo:



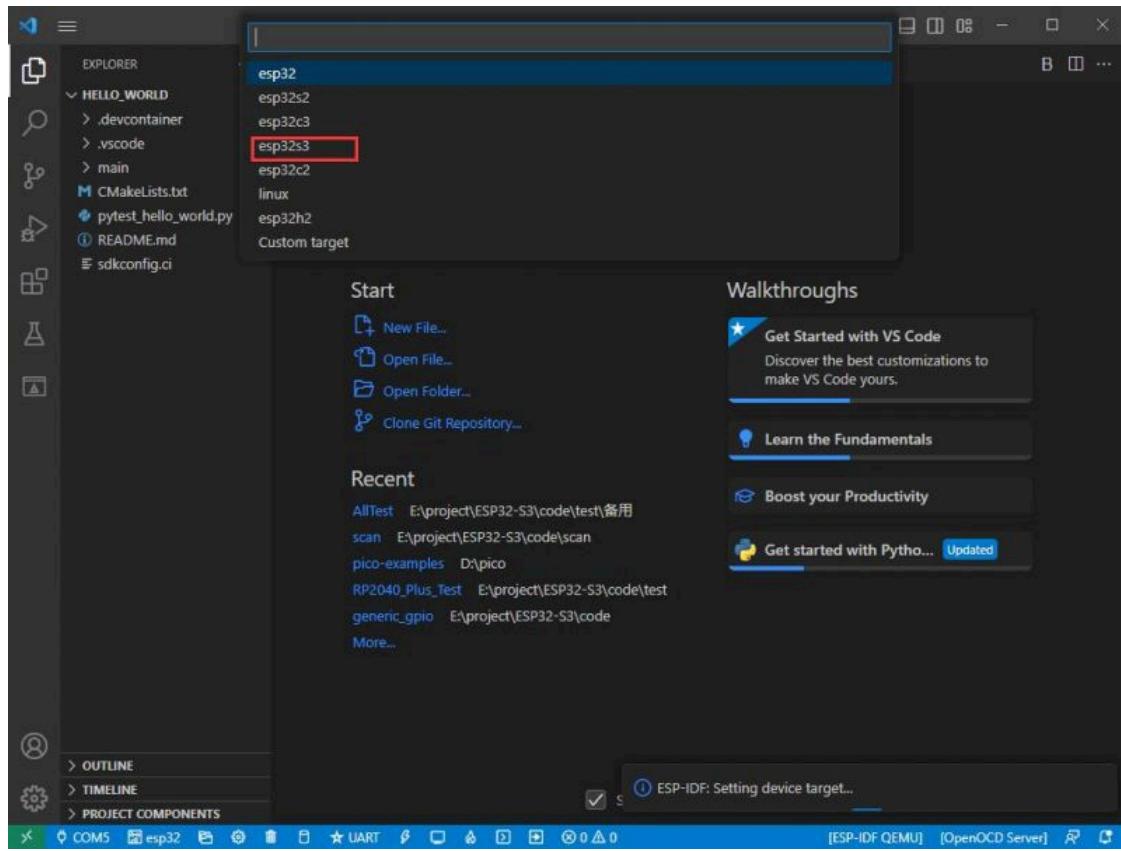
Pico_24.jpg

3. Wait for a few seconds after clicking.



Pico_27.jpg

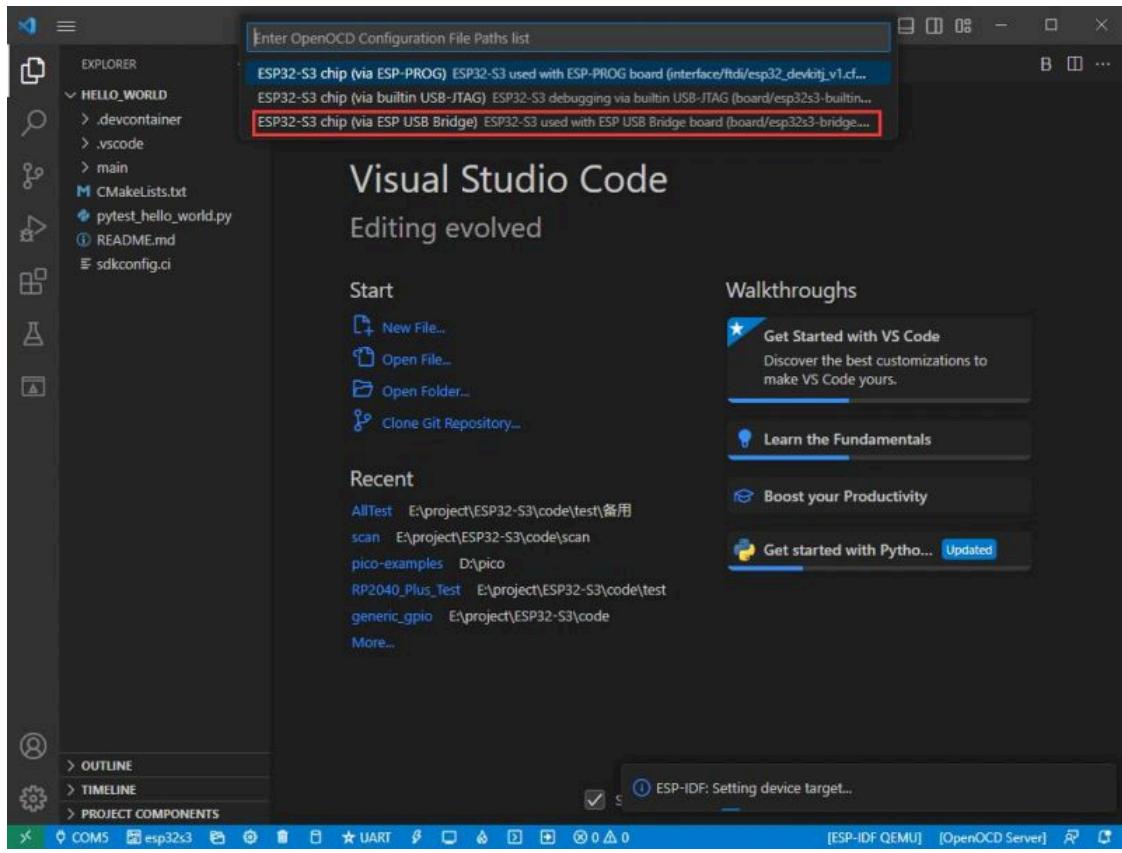
4. Choose the driver we need, that is, the main chip ESP32S3.



(/wiki/File:ESP32-S3-

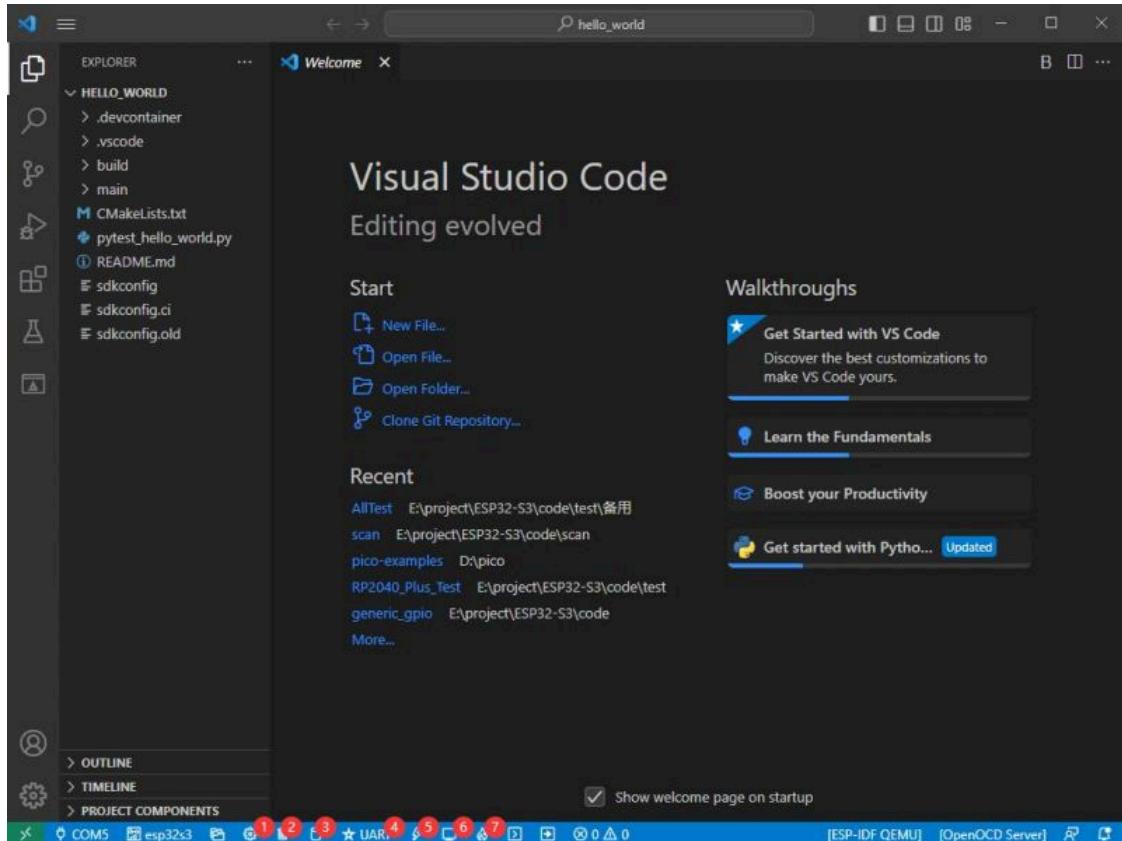
Pico_28.jpg)

5. Choose the openocd path, we can just choose one at random as it doesn't matter.



Pico_29.jpg)

The Rest of the Status Bar Introduction



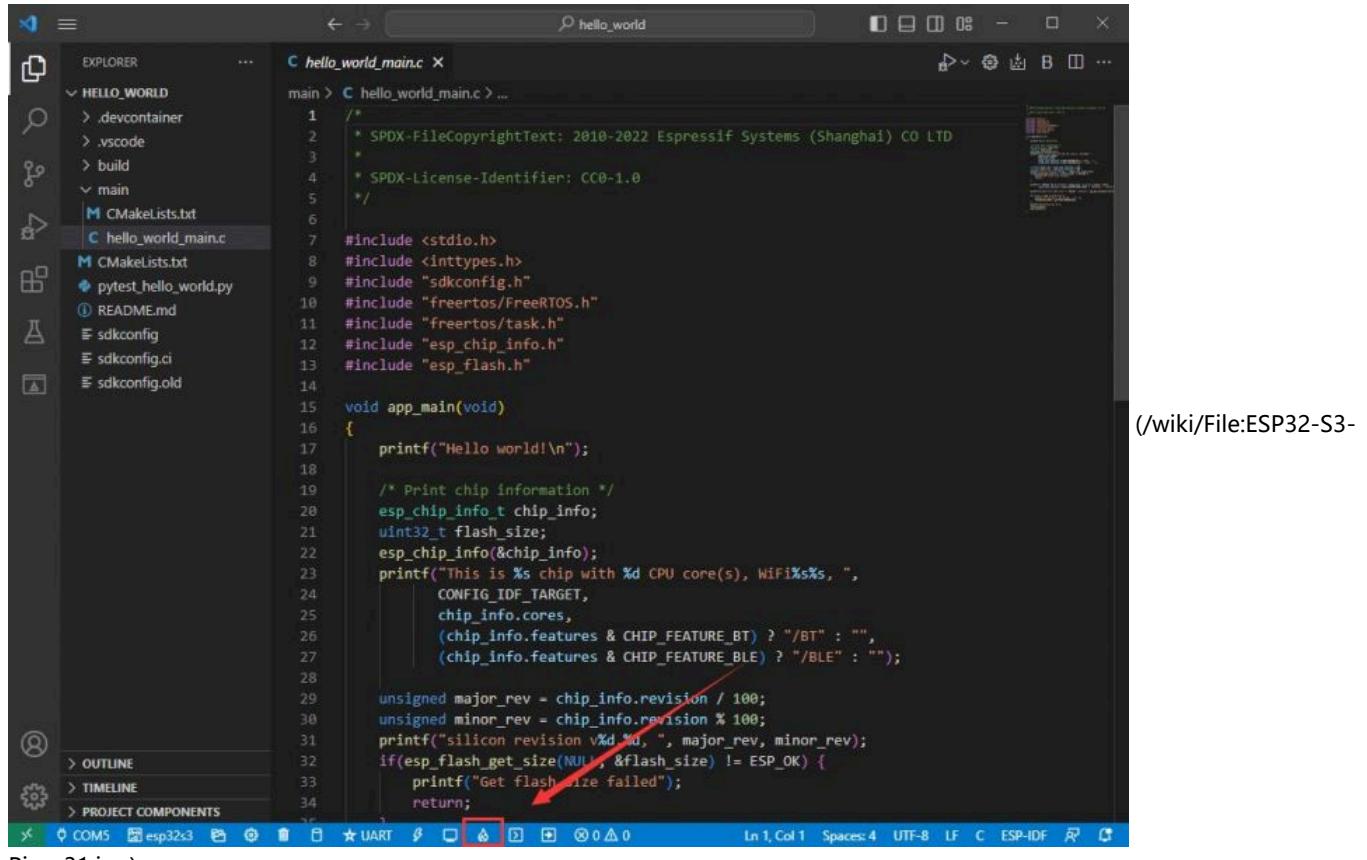
Pico_30.jpg)

- ① SDK configuration editor: many functions and configurations of ESP-IDF can be modified within it.
- ② Clean up everything and delete all compiled files.
- ③ Compile.

- ④ Current download method, default is UART.
- ⑤ Program the current firmware, please do it after compiling.
- ⑥ Open the serial monitor to view serial information.
- ⑦ Combined button for compiling, programming, and opening the serial monitor (most commonly used during debugging).

Compile, Program, and Serial Port Monitoring

1. Click on the Compile, Program, and Open Serial Monitor buttons we described earlier.



Pico_31.jpg)

2. It may take a long time to compile, especially for the first time.

```
[103/106] Linking C static library esp-idf\main\libmain.a
[104/106] Linking C executable bootloader.elf
[105/106] Generating binary image from built executable
esptool.py v4.5.1
Creating esp32s3 image...
Merged 1 ELF section
Successfully created esp32s3 image.
Generated E:/esp32-s3-example/hello_world/build/bootloader/bootloader.bin
[106/106] cmd.exe /C "cd /D E:/esp32-s3-example & idf5\tool\python_env\idf5_0_py3.8_env\Scripts\check_size.py --offset 0x8000 bootloader.elf"
Bootloader binary size 0x5180 bytes. 0x2e88
[655/883] Building C object esp-idf/fuse/Chakravarthy_erase.o
[656/883] Building C object esp-idf/fuse/Chakravarthy_erase_table.C.o

```

Pico_32.jpg

- During this process, ESP-IDF may take up a lot of CPU resources and therefore may cause system lag.

3. Because we use CH343 as a USB to serial port chip, and the on-board automatic download circuit, it can be downloaded automatically without manual operation.

```
Features: WiFi, BLE
Crystal is 40MHz
MAC: 34:85:18:b9:0f:1c
Uploading stub...
Running stub...
Stub running...
Changing baud rate to 460800
Changed.
Configuring flash size...
Flash will be erased from 0x00000000 to 0x80000000
Flash will be erased from 0x00010000 to 0x00020000
Flash will be erased from 0x00008000 to 0x00010000
Compressed 20864 bytes to 1339...
Writing at 0x00000000... (100 %)
```

Pico_33.jpg

4. After successful download, it will automatically enter the serial monitor, and you can see the corresponding information output from the chip and prompt to reboot after 10s.

The screenshot shows the Visual Studio Code interface with the following details:

- Explorer View:** Shows the project structure under "HELLO_WORLD". The file "hello_world_main.c" is selected.
- Code Editor:** Displays the C code for "hello_world_main.c". The code includes standard library includes, FreeRTOS headers, and the main application entry point "app_main".
- Terminal Output:** Shows the boot logs and the "Hello world!" message. It also indicates a detected chip (Winbond) and a minimum free heap size of 391944 bytes, with a restart command issued.
- Bottom Status Bar:** Shows connection status to "COM5" and "esp32c3", along with other system information like memory usage.

Pico_34.jpg)

[Collapse]

Arduino

- If you do not use arduino-esp32 before, you can refer to this link (<https://docs.espressif.com/projects/arduino-esp32/en/latest/index.html>).

Install Arduino IDE

- Open the official software download webpage (<https://www.arduino.cc/en/software>), and choose the corresponding system and system bits to download.

Downloads

The new major release of the Arduino IDE is faster and even more powerful! In addition to a more modern editor and a more responsive interface it features autocompletion, code navigation, and even a live debugger.

For more details, please refer to the [Arduino IDE 2.0 documentation](#).

Nightly builds with the latest bugfixes are available through the section below.

SOURCE CODE

The Arduino IDE 2.0 is open source and its source code is hosted on [GitHub](#).

DOWNLOAD OPTIONS

- Windows** Win 10 and newer, 64 bits
- Windows** MSI installer
- Windows** ZIP file
- Linux** AppImage 64 bits (X86-64)
- Linux** ZIP file 64 bits (X86-64)
- macOS** Intel, 10.14: "Mojave" or newer, 64 bits
- macOS** Apple Silicon, 11: "Big Sur" or newer, 64 bits

[Release Notes](#)

(/wiki/File:ESP32-S3-

Pico_35.jpg)

2. You can choose "Just Download", or "Contribute & Download".

Support the Arduino IDE

Since the release 1.x release in March 2015, the Arduino IDE has been downloaded **70,749,707** times — impressive! Help its development with a donation.

\$3 \$5 \$10 \$25 \$50 Other

JUST DOWNLOAD **CONTRIBUTE & DOWNLOAD**

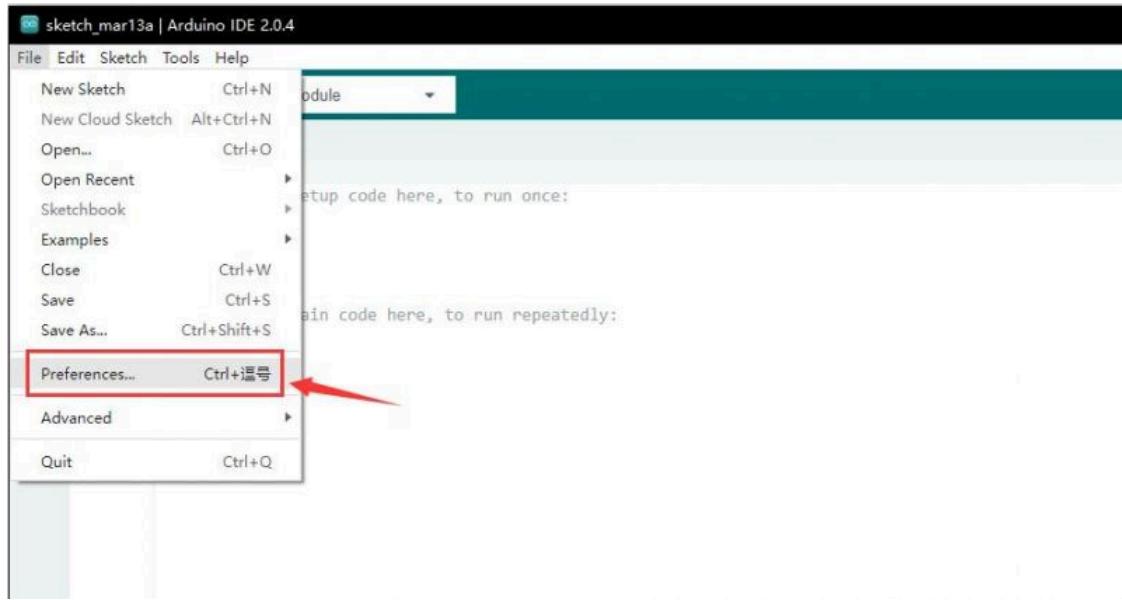
Learn more about [donating to Arduino](#).

(/wiki/File:ESP32-S3-Pico_36.jpg)

3. Run to install the program and install it all by default.

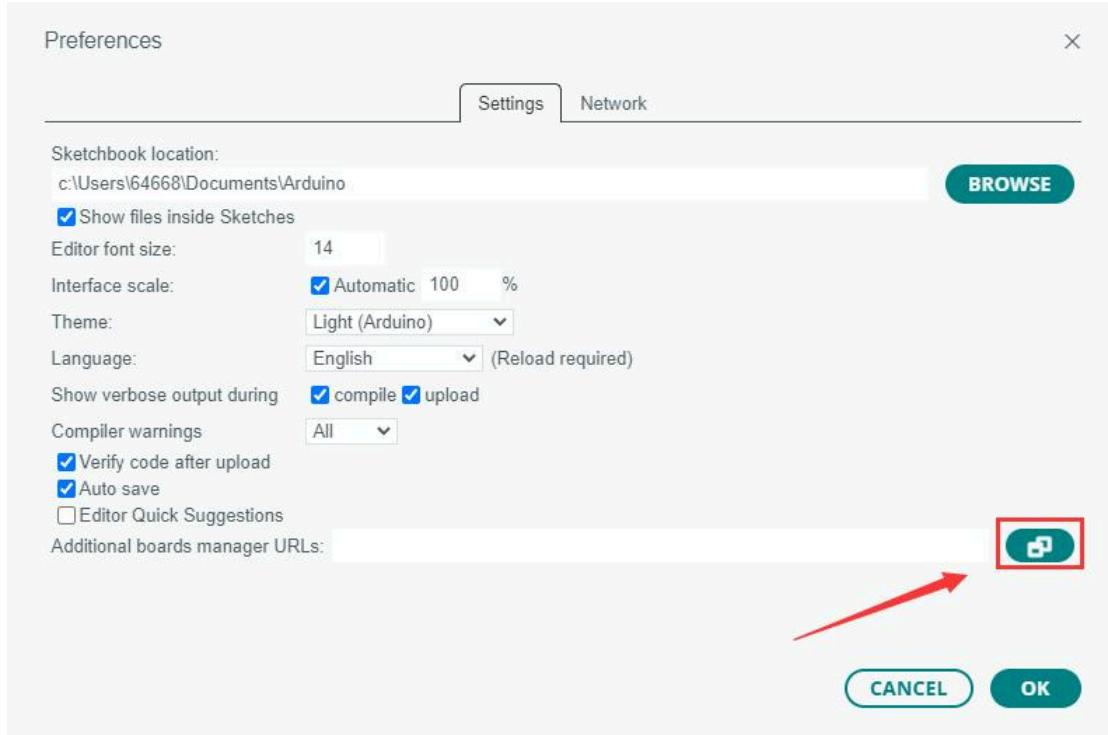
Install arduino-esp32 Online

1. Open Preferences.



Pico_37.jpg

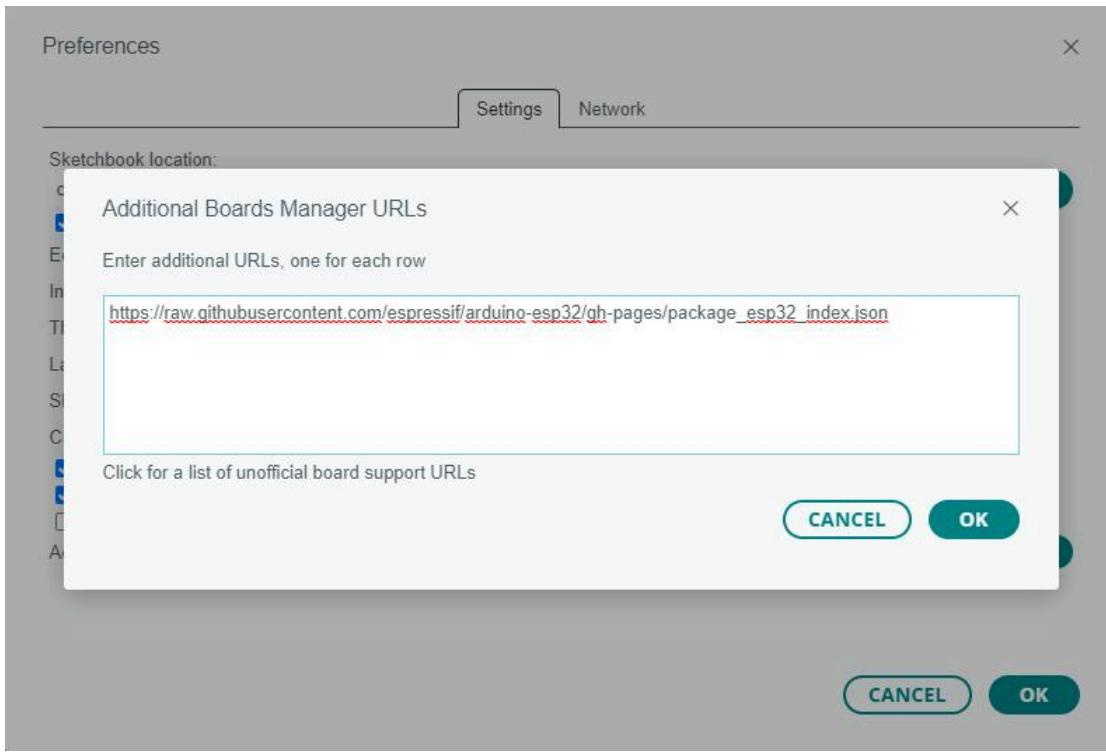
2. Add the corresponding board manager URLs and click the button.



Pico_38.jpg

3. Add the following content in the first blank.

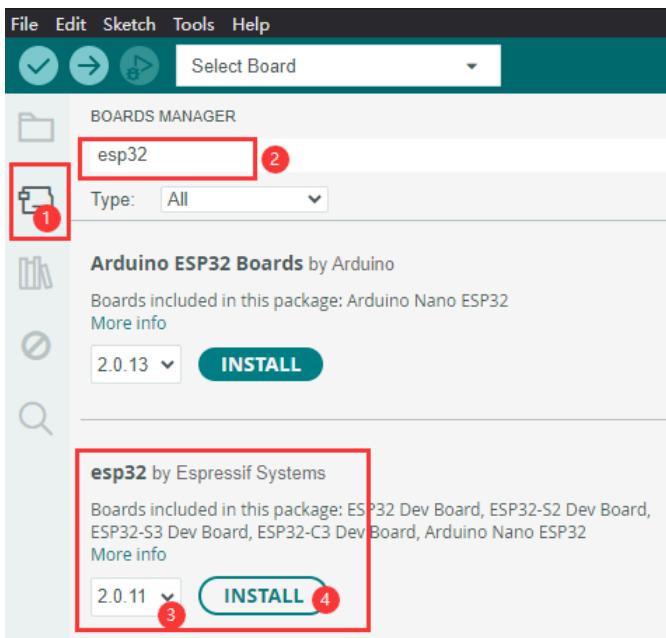
```
https://raw.githubusercontent.com/espressif/arduino-esp32/gh-pages/package_esp32_index.json (https://raw.githubusercontent.com/espressif/arduino-esp32/gh-pages/package_esp32_index.json)
```



(/wiki/File:ESP32-S3-

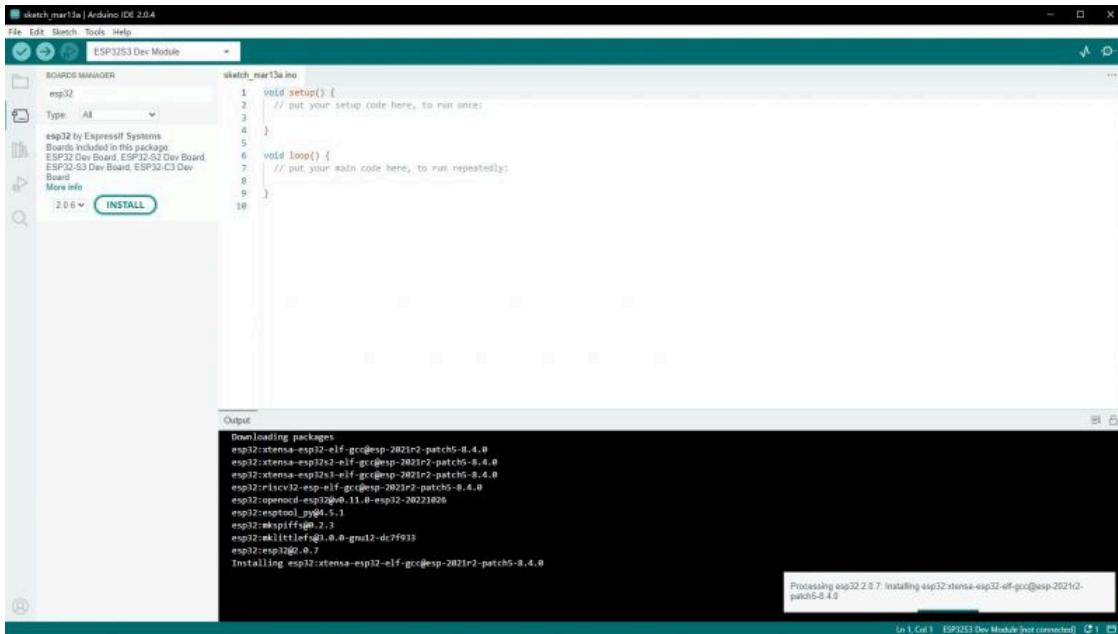
Pico_39.jpg)

4. Save the setting.
5. Open the board manager, enter ESP32 in the search bar, and select version **2.0.11**.



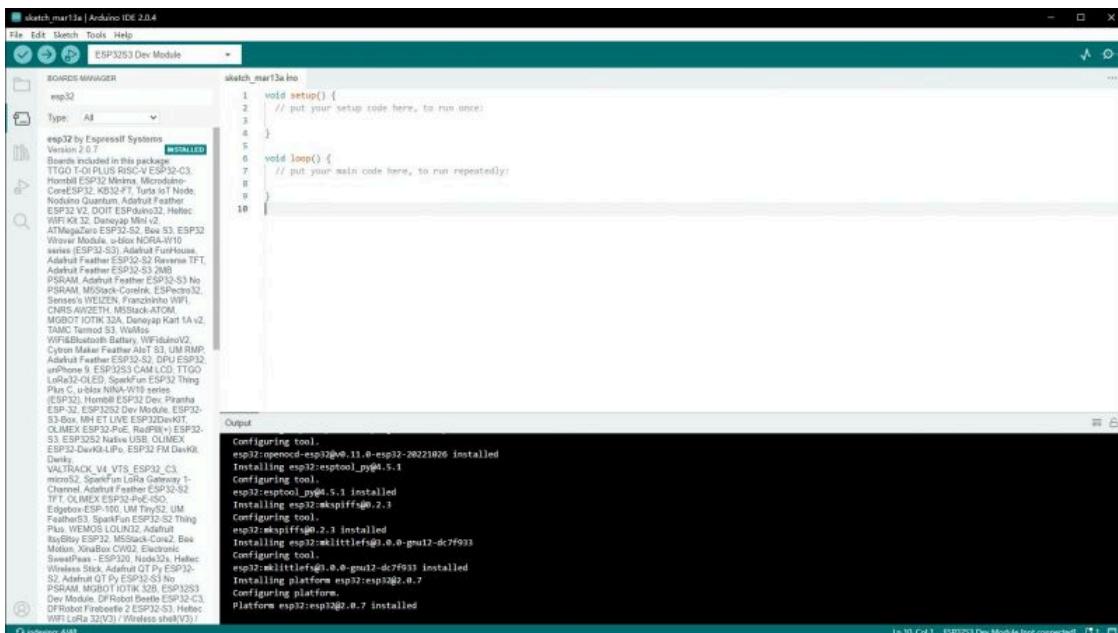
(/wiki/File:ESP32-S3-Pico-Ar-06-240708.png)

6. Wait for downloading.



Pico_41.jpg

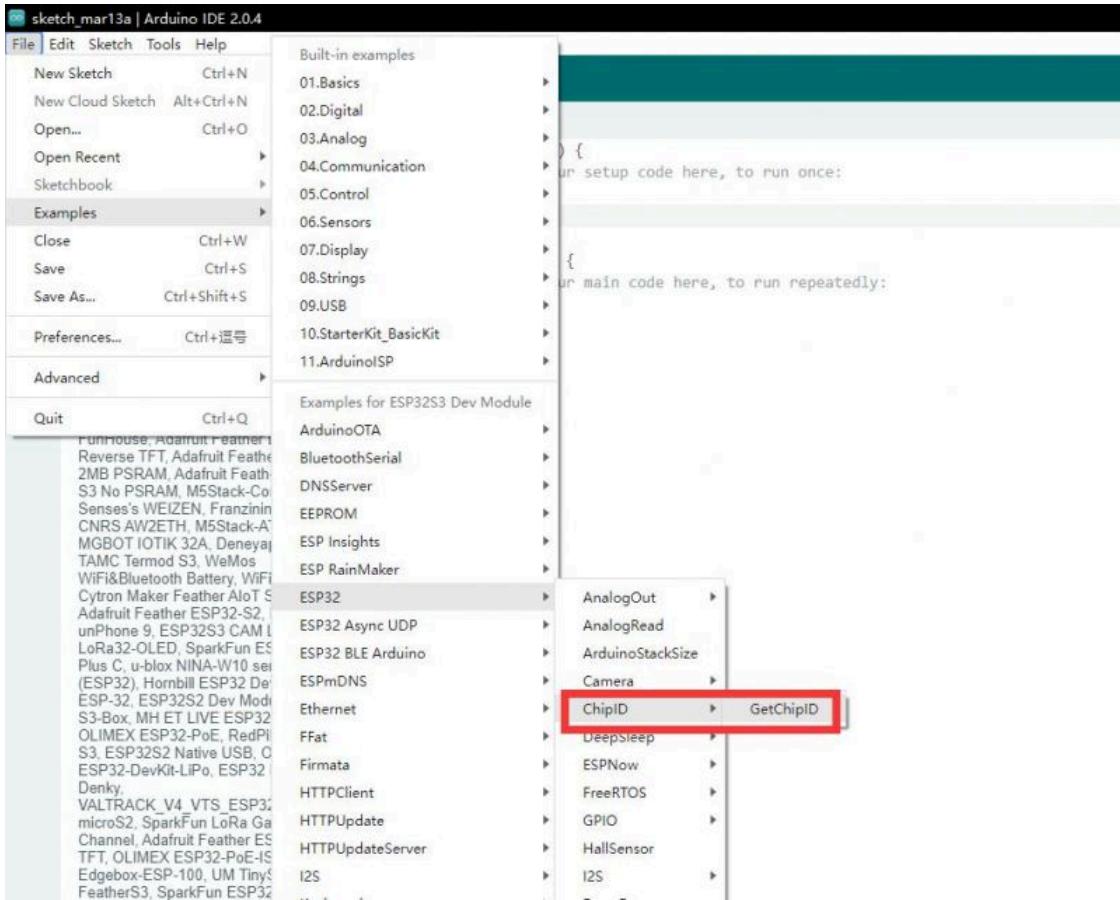
7. arduino-esp32 is downloaded.



Pico_42.jpg

Use Arduino Demo

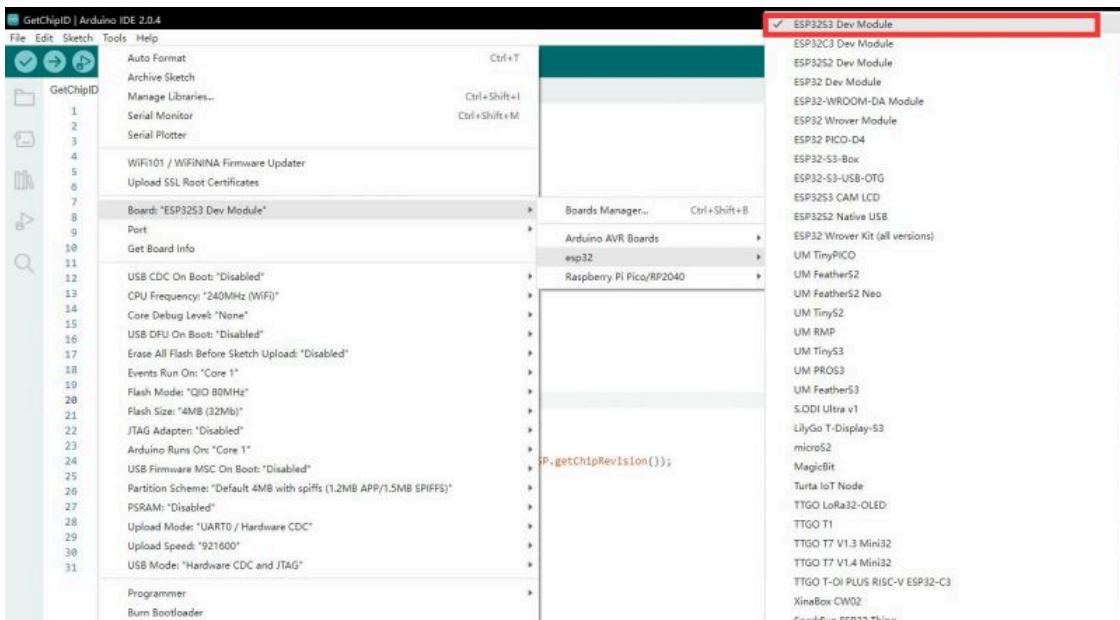
1. Select the demo, here we choose the demo to get the chip ID.



(/wiki/File:ESP32-S3-

Pico_43.jpg

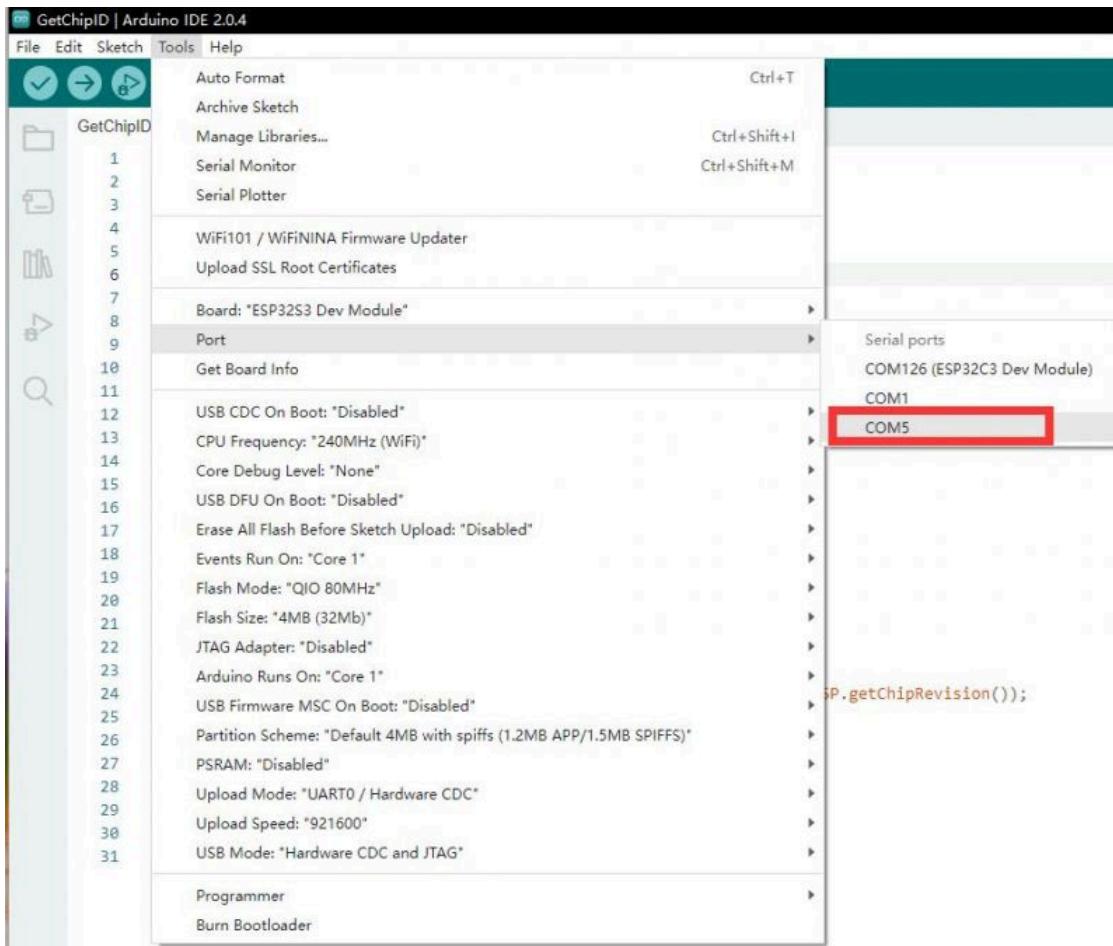
2. Select the board as ESP32S3 Dev Module.



(/wiki/File:ESP32-S3-

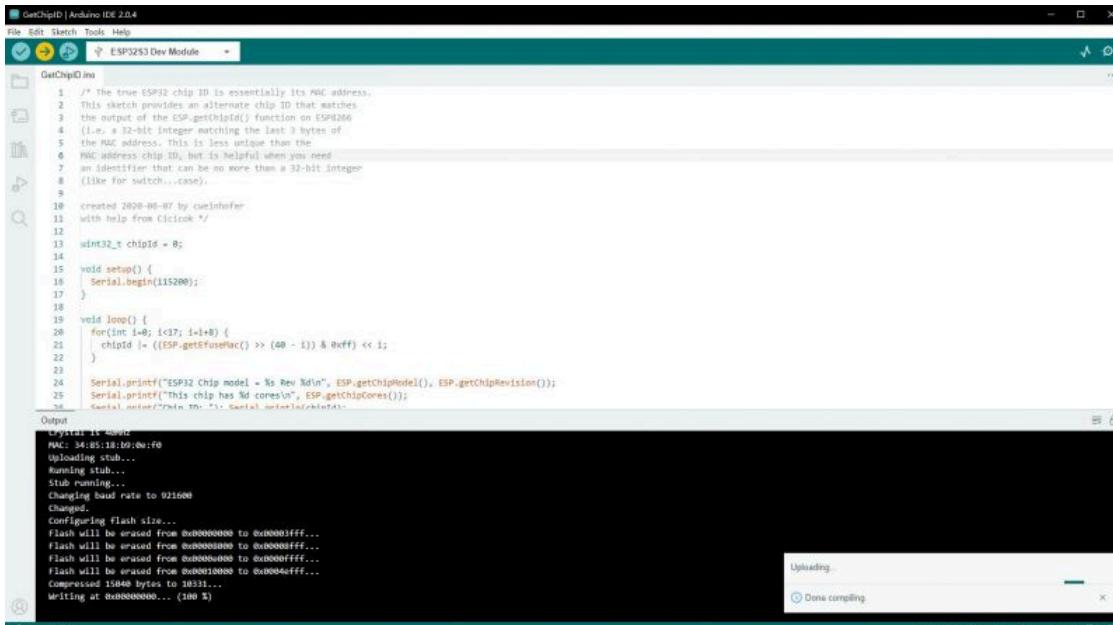
Pico_44.jpg

3. Choose the COM5 port of ESP32-S3 USB.



Pico_45.jpg)

4. Click the download button, then it compiles and downloads automatically.



Pico_47.jpg)

5. Finish.

```
GetChipID.ino
1 /* The true ESP32 chip ID is essentially its MAC address.
2 This sketch provides an alternate chip ID that matches
3 the output of the ESP.getChipId() function on ESP8266
4 (i.e. a 32-bit integer matching the last 3 bytes of
5 the MAC address. This is less unique than the
6 MAC address chip ID, but is helpful when you need
7 an identifier that can be no more than a 32-bit integer
8 (like for switch...case).
9
10 created 2020-06-07 by cweinrhofer
11 with help from Cicicok */
12
13 uint32_t chipId = 0;
14
15 void setup() {
16   Serial.begin(115200);
17 }
18
19 void loop() {
20   for(int i=0; i<17; i+=8) {
21     chipId |= ((ESP.getEfuseMac() >> (40 - i)) & 0xff) << i;
22   }
23
24   Serial.print("ESP32 Chip model = %s Rev %d", ESP.getChipModel(), ESP.getChipRevision());
25   Serial.print("This chip has %d cores\n", ESP.getChipCores());
26   Serial.println("Resetting in 3 seconds");
27 }
Output
Writing at 0x00000000... (11 %)
Writing at 0x0000073d... (22 %)
Writing at 0x0000040c... (33 %)
Writing at 0x00000990c... (44 %)
Writing at 0x00000ee0c... (55 %)
Writing at 0x00000590c... (66 %)
Writing at 0x00000490c... (77 %)
Writing at 0x000004cc4c... (88 %)
Writing at 0x000004ac9c... (99 %)
Write 256752 bytes (143313 compressed) at 0x000010000 in 2.9 seconds (effective 708.3 kbit/s)...
Hash of data verified.

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

(/wiki/File:ESP32-S3-

Pico_48.jpg

6. Open the Serial Port Monitor.

```
GetChipID.ino
1 /* The true ESP32 chip ID is essentially its MAC address.
2 This sketch provides an alternate chip ID that matches
3 the output of the ESP.getChipId() function on ESP8266
4 (i.e. a 32-bit integer matching the last 3 bytes of
5 the MAC address. This is less unique than the
6 MAC address chip ID, but is helpful when you need
7 an identifier that can be no more than a 32-bit integer
8 (like for switch...case).
9
10 created 2020-06-07 by cweinrhofer
11 with help from Cicicok */
12
13 uint32_t chipId = 0;
14
15 void setup() {
16   Serial.begin(115200);
17 }
18
19 void loop() {
20   for(int i=0; i<17; i+=8) {
21     chipId |= ((ESP.getEfuseMac() >> (40 - i)) & 0xff) << i;
22   }
23 }
Output
Uninstalling esp32:mklittlefs@3.0.0-gnu12-dc7f933, tool is no more required
Tool esp32:mklittlefs@3.0.0-gnu12-dc7f933 uninstalled
Uninstalling esp32:mkspiffs@0.2.3, tool is no more required
Tool esp32:mkspiffs@0.2.3 uninstalled
Uninstalling esp32:openocd-esp32@v0.11.0-esp32-20221026, tool is no more required
Tool esp32:openocd-esp32@v0.11.0-esp32-20221026 uninstalled
Uninstalling esp32:iscv32-esp-elf-gcc@esp-2021r2-patch5-8.4.0, tool is no more required
Tool esp32:iscv32-esp-elf-gcc@esp-2021r2-patch5-8.4.0 uninstalled
Uninstalling esp32:xtensa-esp32-elf-gcc@esp-2021r2-patch5-8.4.0, tool is no more required
Tool esp32:xtensa-esp32-elf-gcc@esp-2021r2-patch5-8.4.0 uninstalled
Uninstalling esp32:xtensa-esp32s2-elf-gcc@esp-2021r2-patch5-8.4.0, tool is no more required
Tool esp32:xtensa-esp32s2-elf-gcc@esp-2021r2-patch5-8.4.0 uninstalled
```

(/wiki/File:ESP32-S3-

Pico_49.jpg

7. See the chip ID of the loop output.

```

GetChipID.ino
10  created 2020-06-07 by cweinrhofer
11  with help from Cicicok */
12
13  uint32_t chipId = 0;
14
15  void setup() {
16    Serial.begin(115200);
17 }
18
19  void loop() {
20    for(int i=0; i<17; i+=8) {
21      chipId |= ((ESP.getEfuseMac() >> (40 - i)) & 0xff) << i;
22    }
23
24    Serial.printf("ESP32 Chip model = %s Rev %d\n", ESP.getChipModel(), ESP.getChipRevision());
25    Serial.printf("This chip has %d cores\n", ESP.getChipCores());
26    Serial.print("Chip ID: "); Serial.println(chipId);
27
28    delay(3000);
29
30  }

```

(/wiki/File:ESP32-S3-

Output Serial Monitor X

Message (Enter to send message to 'ESP32S3 Dev Module' on 'COM5')

New Line 115200 baud

```

18:05:42.998 -> SPIWP:0xee
18:05:42.998 -> mode:DIO, clock div:1
18:05:42.998 -> load:0x3fce3800, len:0x44c
18:05:42.998 -> load:0x403c9700, len:0xbe4
18:05:42.998 -> load:0x403cc700, len:0x2a38
18:05:42.998 -> entry:0x403c98d4
18:05:43.041 -> ESP32 Chip model = ESP32-S3 Rev 0
18:05:43.041 -> This chip has 2 cores
18:05:43.041 -> Chip ID: 12127984

```

Ln 8, Col 17 ESP32S3 Dev Module on COM5 3

Pico_50.jpg)

Note: This product uses USB to connect to Type-C. If it cannot output the chip information, please modify "Serial.print" function to "printf" function, and directly output to USB.

MicroPython

[Collapse]

Burn Firmware

1. Download the MicroPython firmware: ESP32-S3-GEEK-MPY-Flash Tool (<https://files.waveshare.com/wiki/ESP32-S3-GEEK/ESP32-S3-GEEK-MPY-Flash.zip>) or official micropython firmware (https://micropython.org/download/ESP32_GENERIC_S3/).

2. Unzip the downloaded package and enter:

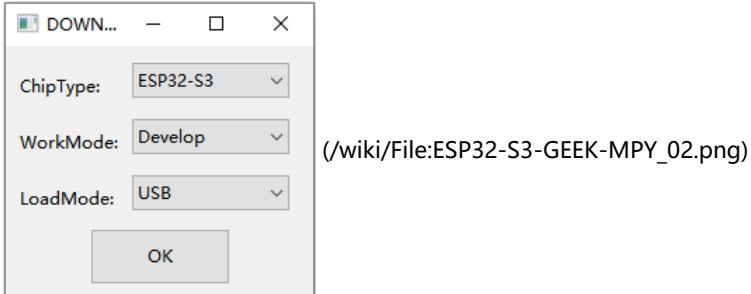
flash_download_tool_3.9.4	2024/1/30 17:38	文件夹			(/wiki/File:ESP32-S3-GEEK-MPY_04.png)
ESP32-S3-GEEK_MPY.bin	2024/1/30 10:22	BIN 文件	1,591 KB		

3. Long-press the ESP32-S3-GEEK key to connect to the USB port of the PC and then release the boot key.

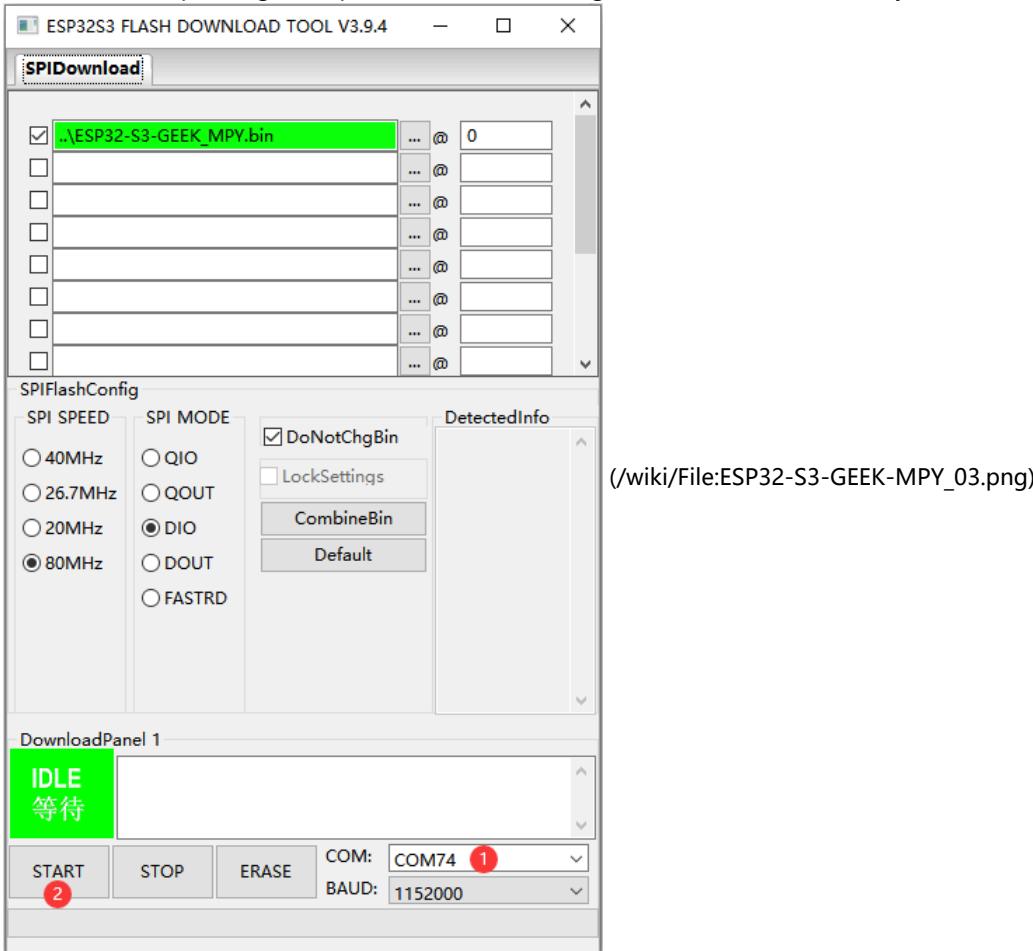
4. Enter the flash_download_tool_3.9.4 file folder, open the flash_download_tool_3.9.4.exe.

bin	2024/1/30 17:53	文件夹			
combine	2023/3/15 18:51	文件夹			
configure	2024/1/30 17:38	文件夹			
dl_temp	2024/1/30 17:38	文件夹			
doc	2024/1/30 17:38	文件夹			
logs	2024/1/30 18:03	文件夹			
flash_download_tool_3.9.4.exe	2023/2/21 11:24	应用程序	15,501 KB		(/wiki/File:ESP32-S3-GEEK-MPY_05.png)

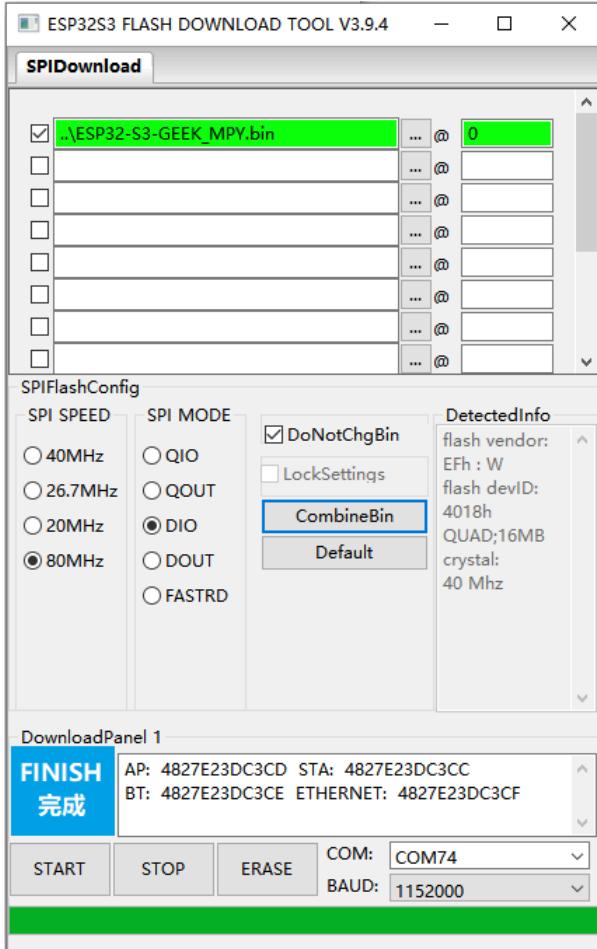
5. Select the corresponding chip, here we select the ESP32-S3 and USB.



6. Select the corresponding COM port, and we have configured other information for you. Just click on "Start" to download.



7. Wait for programming.



(/wiki/File:ESP32-S3-GEEK-MPY_01.png)

Install Thonny

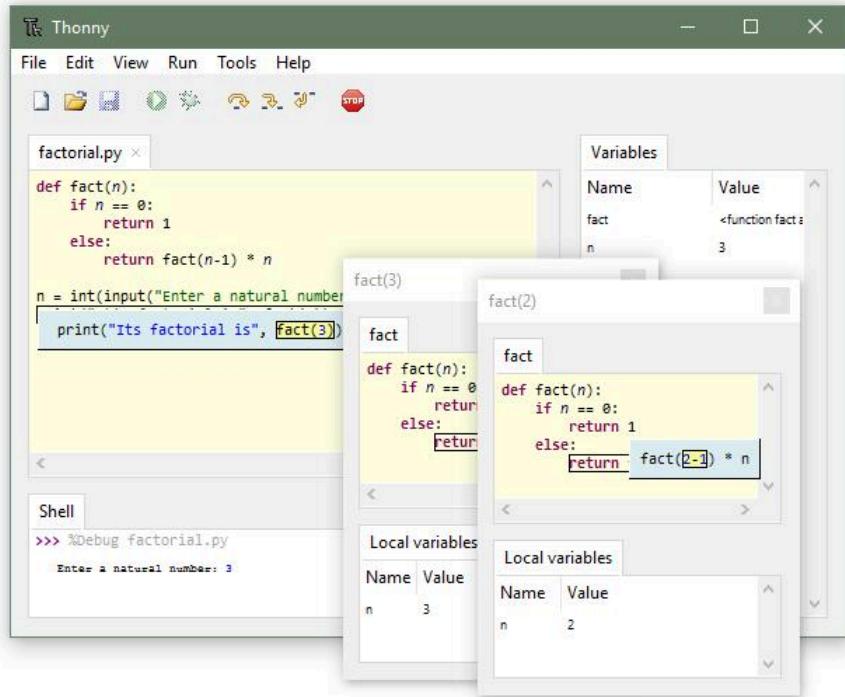
1. Open official Thonny website (<https://thonny.org/>).

Thonny

Python IDE for beginners



Download version [4.0.2](#) for
Windows • Mac • Linux



(/wiki/File:ESP32-S3-Pico_58.jpg)

2. Select the corresponding system and version, here I choose "Windows", and the mouse has to move to Windows, only then will the corresponding information be displayed.

Thonny



Download version [4.0.2](#) for
Windows • Mac • Linux

Official downloads for Windows

Installer with 64-bit Python 3.10, requires 64-bit Windows 8.1 / 10 / 11
[thonny-4.0.2.exe \(20.4 MB\)](#) = recommended for you

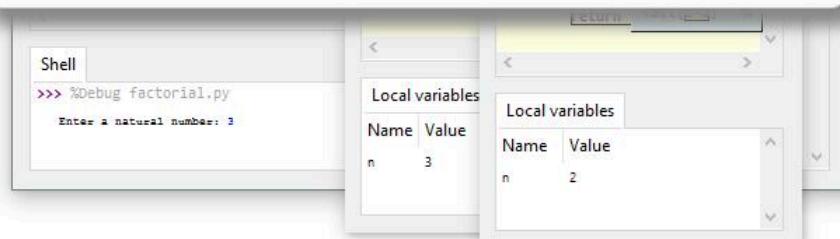
Installer with 32-bit Python 3.8, suitable for all Windows versions since 7
[thonny-py38-4.0.2.exe \(18.9 MB\)](#)

Portable variant with 64-bit Python 3.10
[thonny-4.0.2-windows-portable.zip \(30.5 MB\)](#)

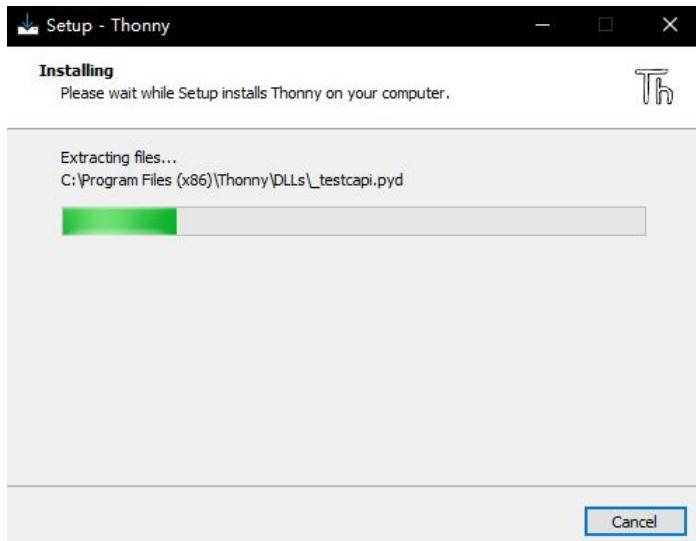
Portable variant with 32-bit Python 3.8
[thonny-py38-4.0.2-windows-portable.zip \(28.6 MB\)](#)

Re-using an existing Python installation (for advanced users)
pip install thonny

(/wiki/File:ESP32-S3-Pico_55.jpg)



3. Just install it by default.



(/wiki/File:ESP32-S3-Pico_56.jpg)

4. Finish.



(/wiki/File:ESP32-S3-Pico_57.jpg)

Demo Download

1. Click here to download (https://files.waveshare.com/wiki/ESP32-S3-GEEK/ESP32-S3-GEEK_Code.zip).

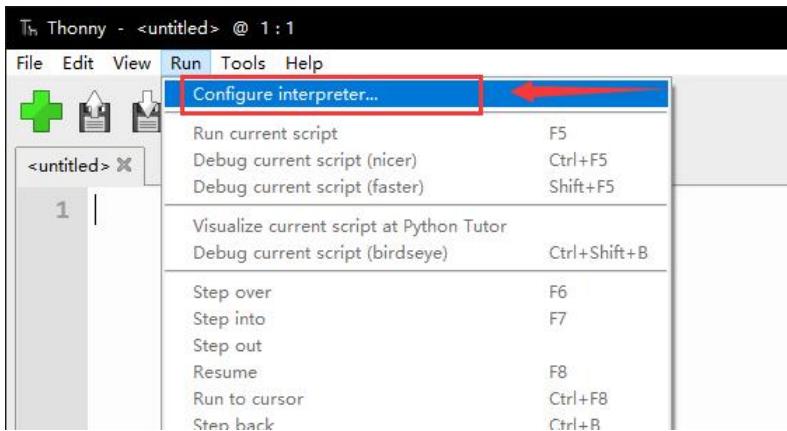
2. Unzip the package.

ESP32-S3-GEEK > ESP32-S3-GEEK_Code > MPY		
	修改日期	更多
BLE	2024/1/31 14:16	...
LCD	2024/1/30 17:10	...
pic	2024/1/30 17:10	...
SD	2024/1/30 17:10	...
UART	2024/1/30 17:10	...
WIFI	2024/1/30 17:10	...
ESP32-S3-GEEK-MPY-Flash.zip	2024/1/30 18:06	...

(/wiki/File:ESP32-S3-GEEK-MPY_09.png)

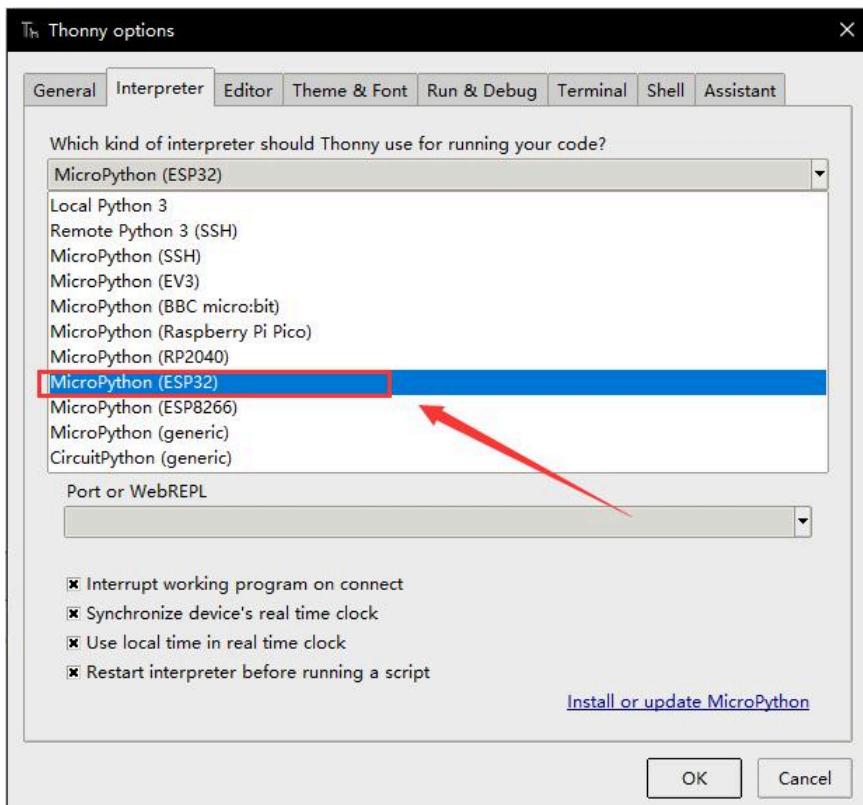
Run the Demo

1. Open Thonny and choose Configure Interpreter.



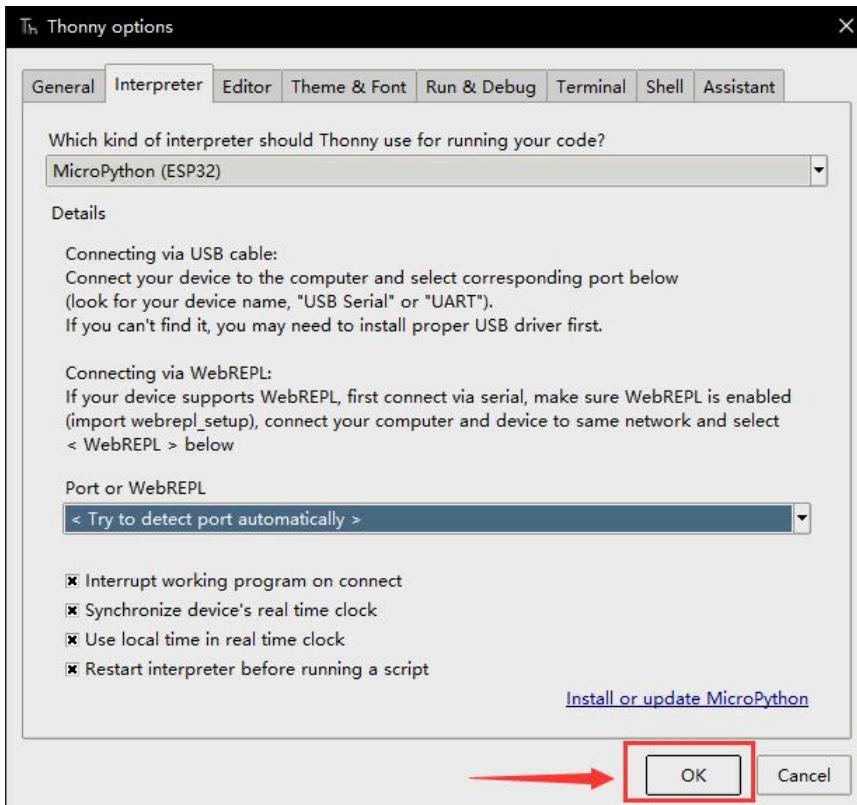
(/wiki/File:ESP32-S3-Pico_90.jpg)

2. Choose MicroPython (ESP32) as the Interpreter.



(/wiki/File:ESP32-S3-Pico_91.jpg)

3. Click OK to save.



(/wiki/File:ESP32-S3-Pico_92.jpg)

4. Click Stop or Ctrl + F2.



(/wiki/File:ESP32-S3-Pico_93.jpg)

5. You can see the MicroPython version information and the board name.

```
Shell X

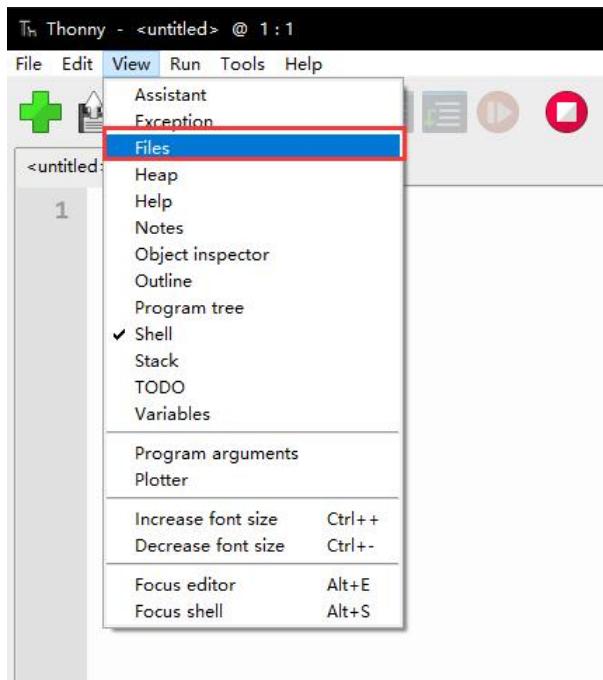
MPY: soft reboot
MicroPython v1.22.1 on 2024-01-05; Generic ESP32S3 module with ESP32S3
Type "help()" for more information.
>>>

MicroPython (ESP32) • Espressif CDC Device @ COM15
```

(/wiki/File:ESP32-S3-

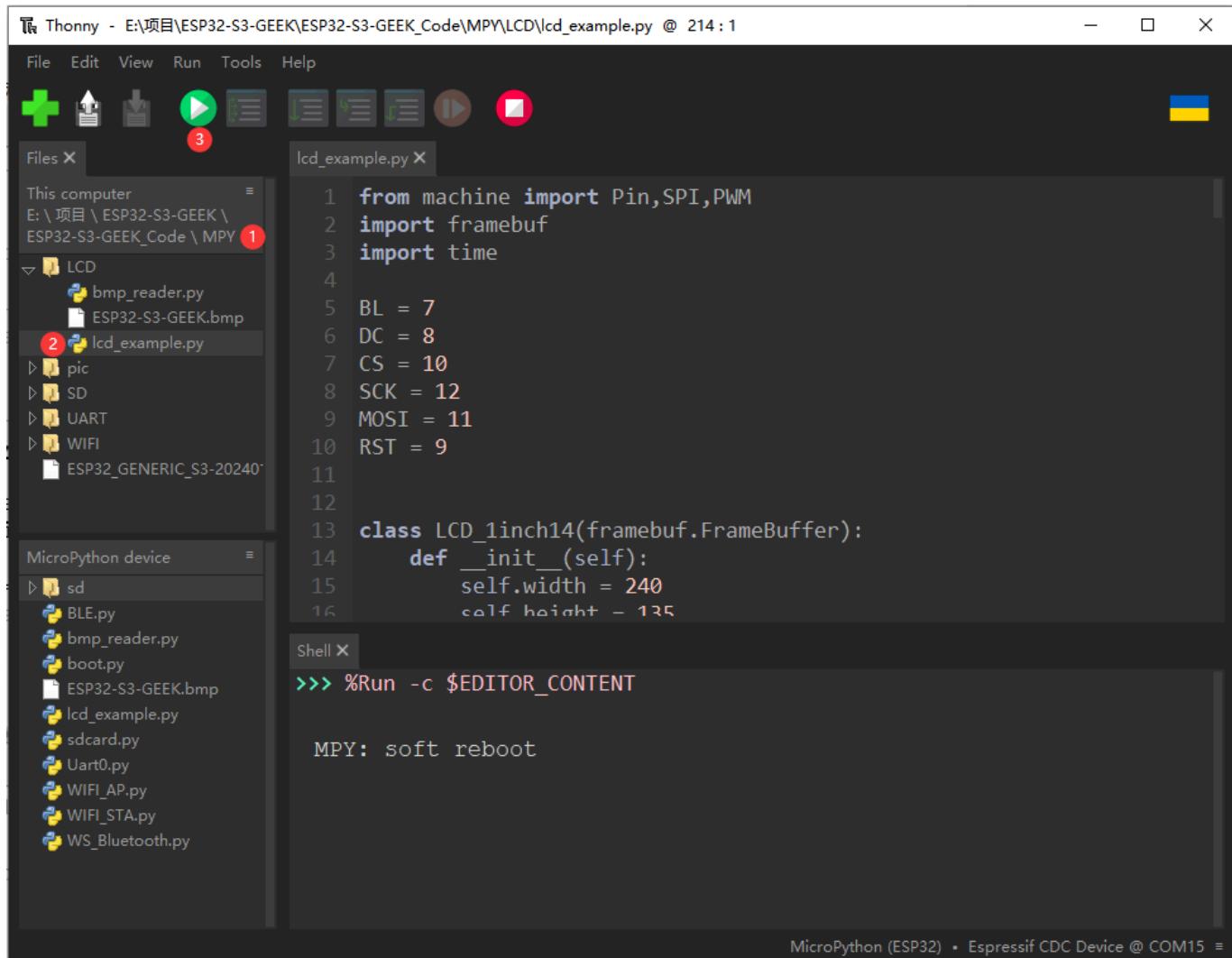
GEEK_94.png)

6. Open the files.



(/wiki/File:ESP-S3-Pico_95.jpg)

7. Choose our demo file and open a demo, here we choose a lcd_example.py demo.



(/wiki/File:ESP-32-GEEK_96.png)

- ①Current local file directory.
- ②The demo we want to open is lcd_example.py.

- ③Click to run or press "F5".

8. Wait a moment and you can see the LCD screen of the ESP32-S3-GEEK displaying the ESP32-S3-GEEK.bmp image.

Arduino Sample Demo

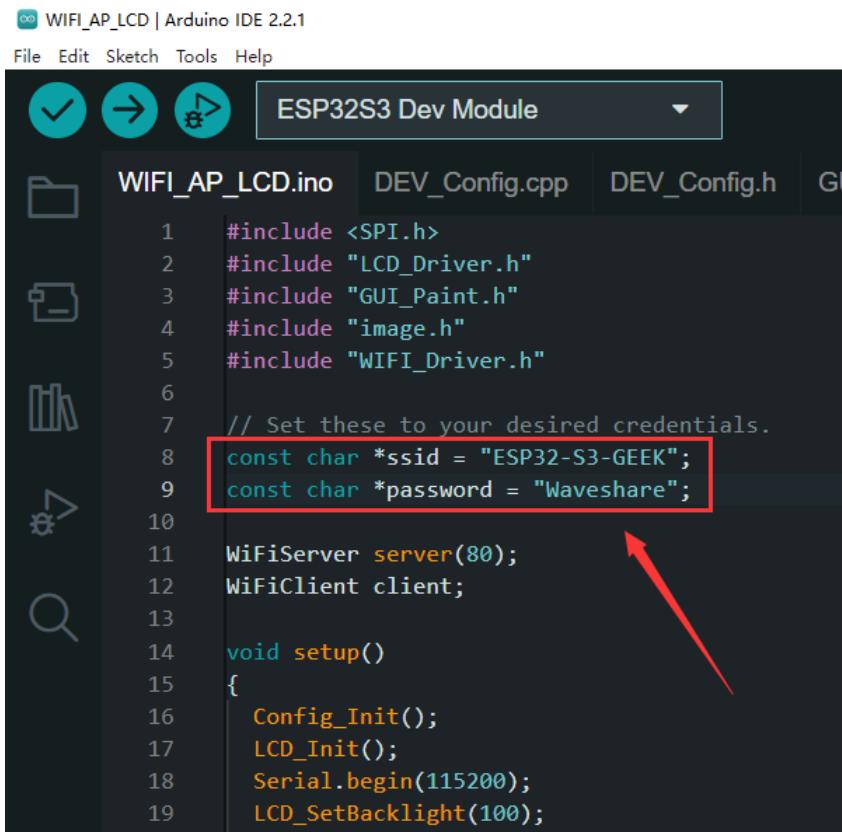
Note: Before using the Arduino demo, please check whether the Arduino environment and the download setting are correctly configured. For more details, you can refer to Arduino (/wiki/ESP32-S3-GEEK#Arduino).

WIFI

01-WIFI_AP_LCD

This code enables ESP32-S3-GEEK to open an Access Point (AP) mode for Wi-Fi. Once a PC is connected to its Wi-Fi network, you can log in to its IP address via a web interface to control the display of images on ESP32-S3-GEEK's LCD.

- "ssid" is the AP name (ESP32-S3-GEEK) created by ESP32-S3-GEEK, and the "password" is the password (Waveshare) for connecting to AP.



The screenshot shows the Arduino IDE interface with the file 'WIFI_AP_LCD.ino' open. The code is as follows:

```

1 #include <SPI.h>
2 #include "LCD_Driver.h"
3 #include "GUI_Paint.h"
4 #include "image.h"
5 #include "WIFI_Driver.h"
6
7 // Set these to your desired credentials.
8 const char *ssid = "ESP32-S3-GEEK";
9 const char *password = "Waveshare";
10
11 WiFiServer server(80);
12 WiFiClient client;
13
14 void setup()
15 {
16     Config_Init();
17     LCD_Init();
18     Serial.begin(115200);
19     LCD_SetBacklight(100);
20 }
21
22 void loop()
23 {
24     if (client.available())
25     {
26         String request = client.readStringUntil('\r');
27         client.flush();
28         ...
29     }
30 }

```

A red box highlights the line 'const char *password = "Waveshare";'. A red arrow points from the text '(/wiki/File:ESP32-S3-GEEK_Demo01.png)' to the start of this line.

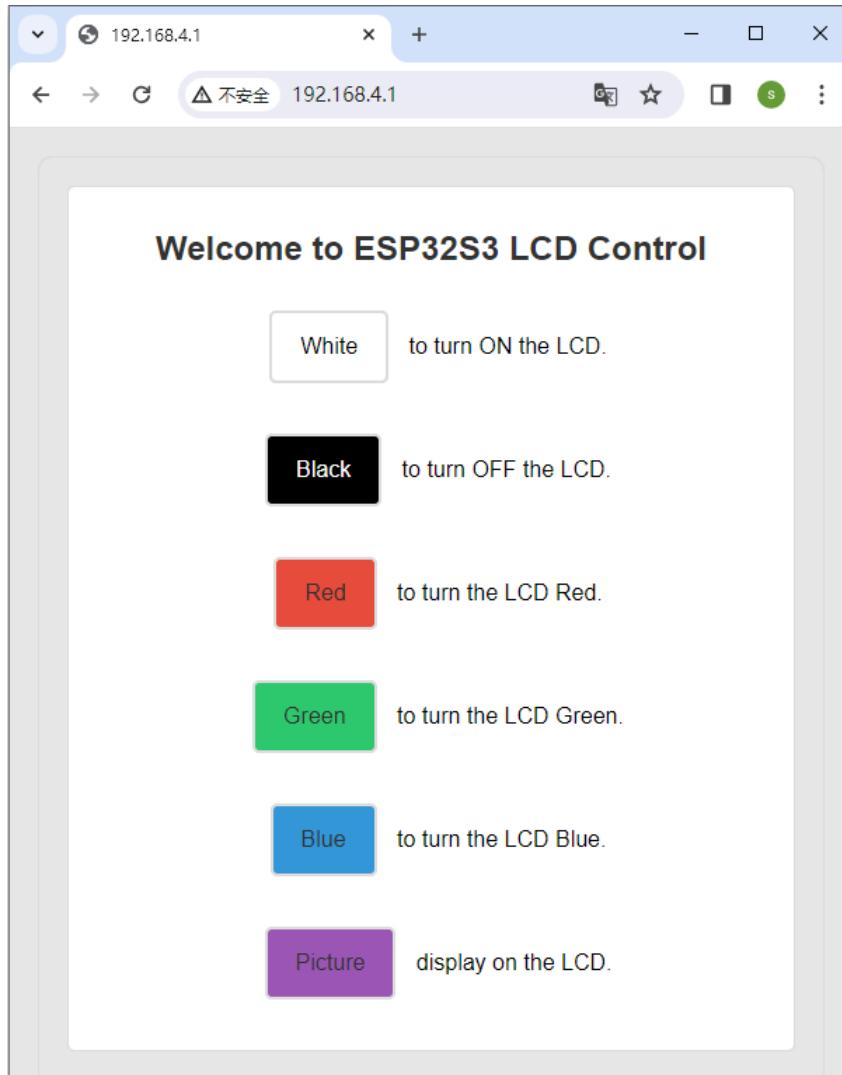
(/wiki/File:ESP32-S3-GEEK_Demo01.png)

- Connect to the AP of the ESP32-S3-GEEK Demo on the PC, and enter the password "Waveshare".
- The LCD will display the IP address of the HTTP server, and you can use the browser to log in the IP: 192.168.4.1.



(/wiki/File:ESP32-S3-GEEK_Demo03.png)

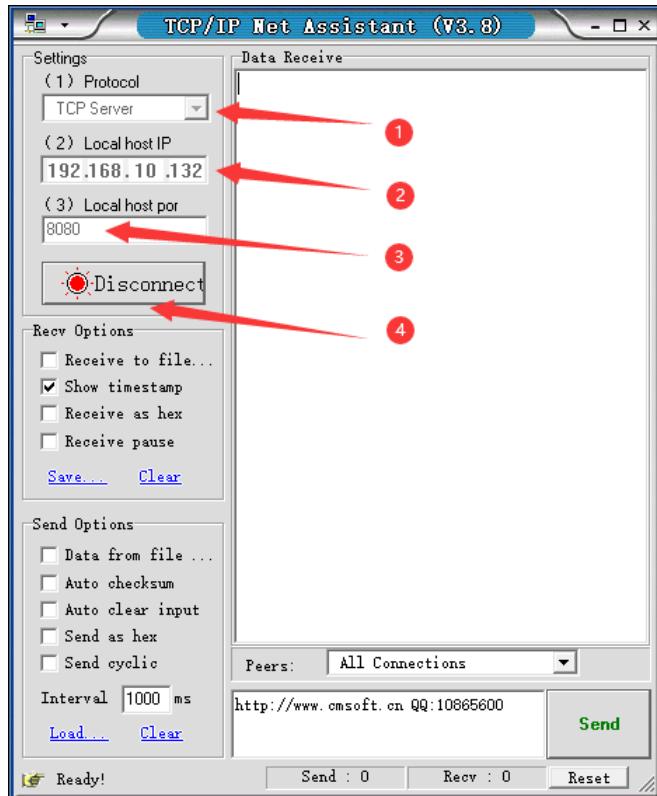
- Control the LCD of the ESP32-S3-GEEK through the button on the server, and you can observe the LCD by clicking on different buttons. For LCD function, you can refer to LCD demo description (https://www.waveshare.com/wiki/1.5inch_LCD_Module#Demo_Remarks_3).



02-WIFI_TCP_Client

With this demo, ESP32-S3-GEEK enters WIFI's STA mode and connects to the same WIFI as the PC or mobile phone. As TCP Client accesses the TCP Server created by the PC or mobile phone, TCP communication is established, and the LCD displays the received content.

Using NetAssist.exe (<https://files.waveshare.com/upload/f/f6/NetAssist.zip>) to enter the TCP Server, the Local host IP is the default one, and the Local host Port is 8080. Click on "Connect" for TCP communication with ESP32-S3-GEEK(TCP Client).



(/wiki/File:ESP32-S3-GEEK_Demo06.png)

- The ESP32-S3-GEEK connects to the same WIFI network as the PC in STA mode. (**The WIFI network that ESP32-S3-GEEK connects to should operate on the 2.4GHz frequency band.** If there's no 2.4GHz frequency band available, the PC can create a hotspot with the network band set to "Any available frequency.") Remember to change the SSID and password to match the WIFI name and password you wish to connect to.

```

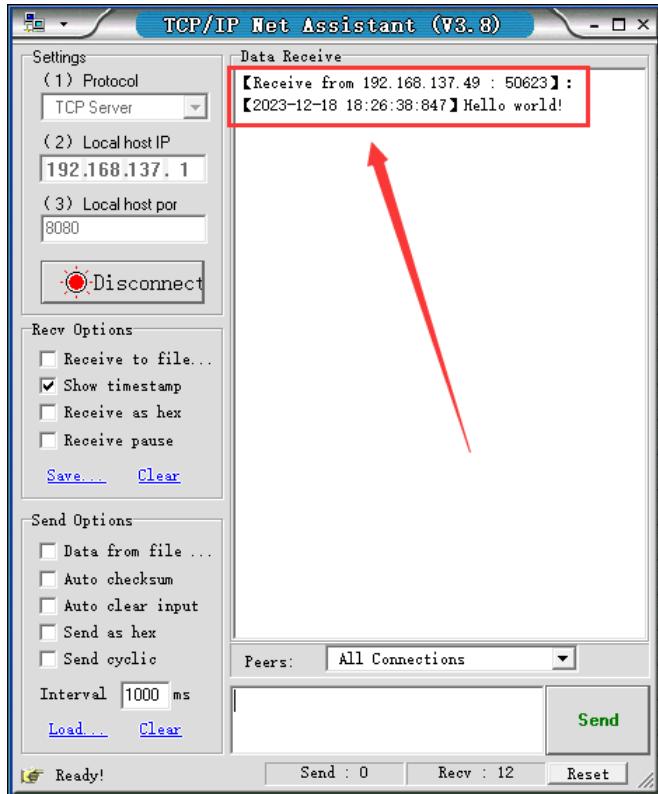
WIFI_TCP_Client | Arduino IDE 2.2.1
File Edit Sketch Tools Help
ESP32S3 Dev Module
WIFI_TCP_Client.ino DEV_Config.cpp DEV_Config.h Debug.h GUI_Paint.cpp GUI ...
1 #include <SPI.h>
2 #include "LCD_Driver.h"
3 #include "GUI_Paint.h"
4 #include "image.h"
5 #include <WiFi.h>
6
7 const char *ssid = "ESP32-S3-GEEK";
8 const char *password = "Waveshare";
9
10 const IPAddress serverIP(192,168,10,132); //Address to be accessed
11 uint16_t serverPort = 8080; //Server port number
12 WiFiClient client; // Declares a client object that is used to ...
13
14 void intToIpAddress(uint32_t ip, char *result) {
15     sprintf(result, "%d.%d.%d.%d", ip & 255,(ip >> 8) & 255,(ip >> 16) & 255,(ip >> 24) & 255);
16 }
17

```

(/wiki/File:ESP32-S3-GEEK_Demo05.png)

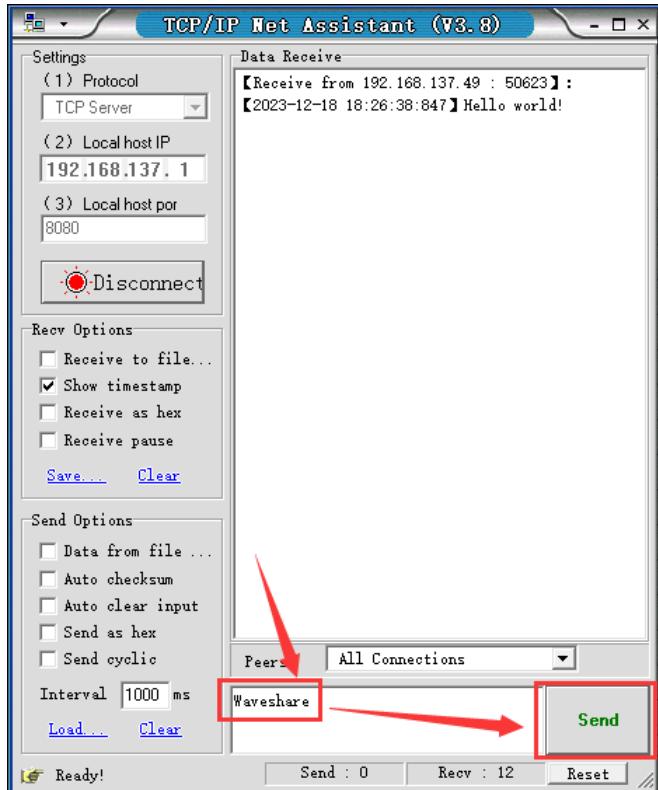
- Modify the serverIP to the TCP Server's IP address obtained from using NetAssist.exe, and keep the serverPort as 8080.**
Upon successful connection, the ESP32-S3-GEEK will send a TCP message "Hello world" to the TCP Server, and the LCD will

display "Access successful".



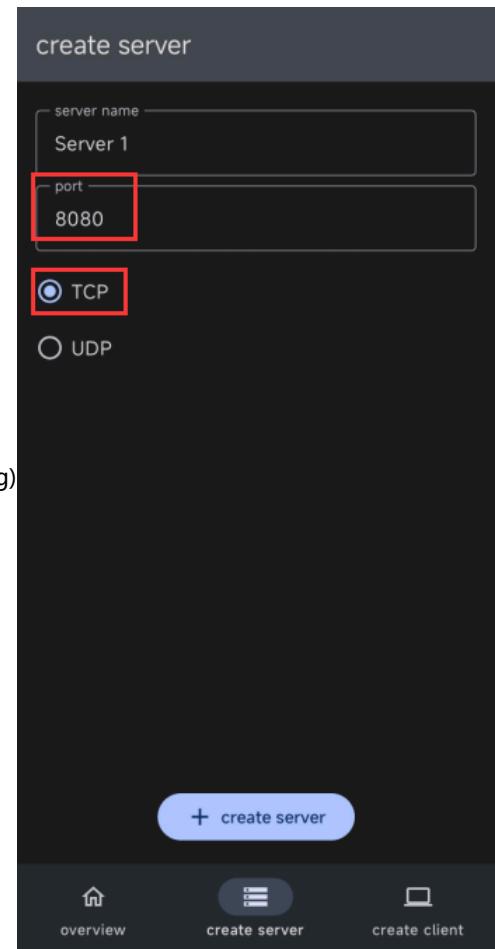
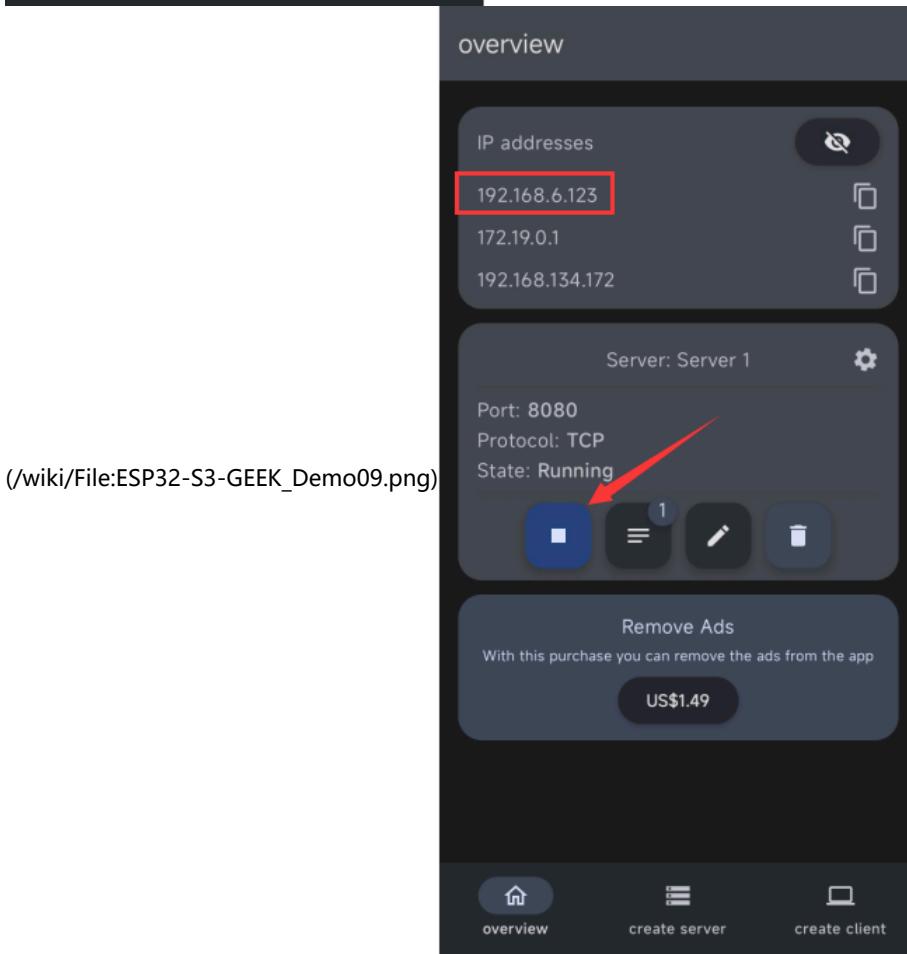
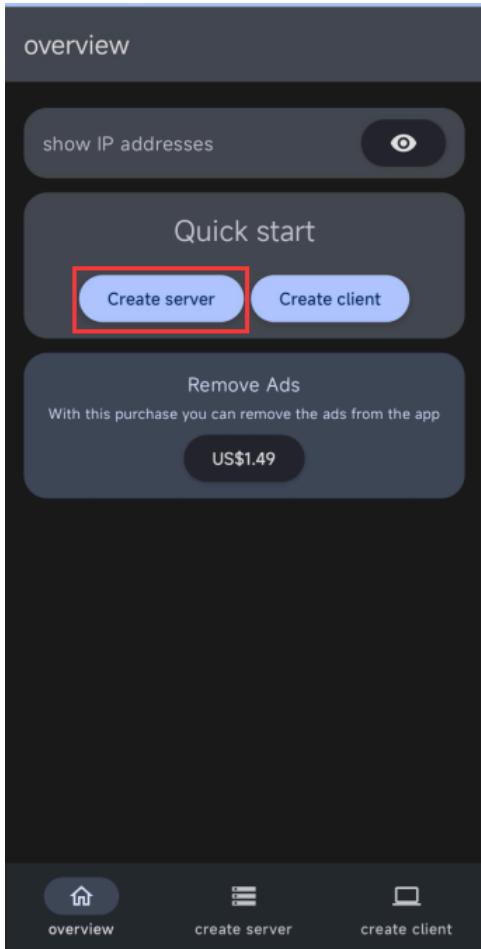
(/wiki/File:ESP32-S3-GEEK_Demo07.png)

- TCP Server sends the TCP message to the ESP32-S3-GEEK on the PC. If sent successfully, ESP32-S3-GEEK as the TCP Client can receive the message and display it on the LCD, and you can observe whether the LCD displays the message.



(/wiki/File:ESP32-S3-GEEK_Demo-tcP_07.png)

- You can also use your phone to create a hotspot with the name and password as "ssid" and "password", respectively, operating on the 2.4GHz frequency band. After setting up the hotspot, you can use a TCP debugging tool (<https://files.waveshare.com/wiki/ESP32-S3-GEEK/Tcp%E8%B0%83%E8%AF%95%E5%8A%A9%E6%89%8B.zip>) to establish TCP communication with ESP32-S3-GEEK.



- Modify the WIFI connected by the code as the hotspot of the smartphone, serverIP is the IP (192.168.6.123) of the TCPserver created by the smartphone in the above picture.

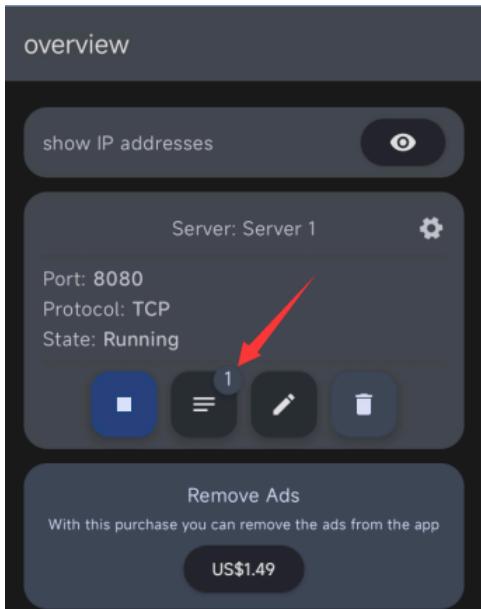
```

WIFI_TCP_Client | Arduino IDE 2.2.1
File Edit Sketch Tools Help
ESP3S3 Dev Module
WIFI_TCP_Client.ino DEV_Config.cpp DEV_Config.h Debug.h GUI_Paint.cpp GUI_Paint.h ...
1 #include <SPI.h>
2 #include "LCD_Driver.h"
3 #include "GUI_Paint.h"
4 #include "image.h"
5 #include <WiFi.h>
6
7 const char *ssid = "ESP32-S3-GEEK"; ①
8 const char *password = "Waveshare";
9
10 const IPAddress serverIP(192,168,6,123); //Address to be accessed ②
11 uint16_t serverPort = 8080;           //Server port number
12 WiFiClient client;                 // Declares a client object that is used to connect
13
14 void intToIpAddress(uint32_t ip, char *result) {
15     sprintf(result, "%d.%d.%d.%d", ip & 255,(ip >> 8) & 255,(ip >> 16) & 255,(ip >> 24) & 255);

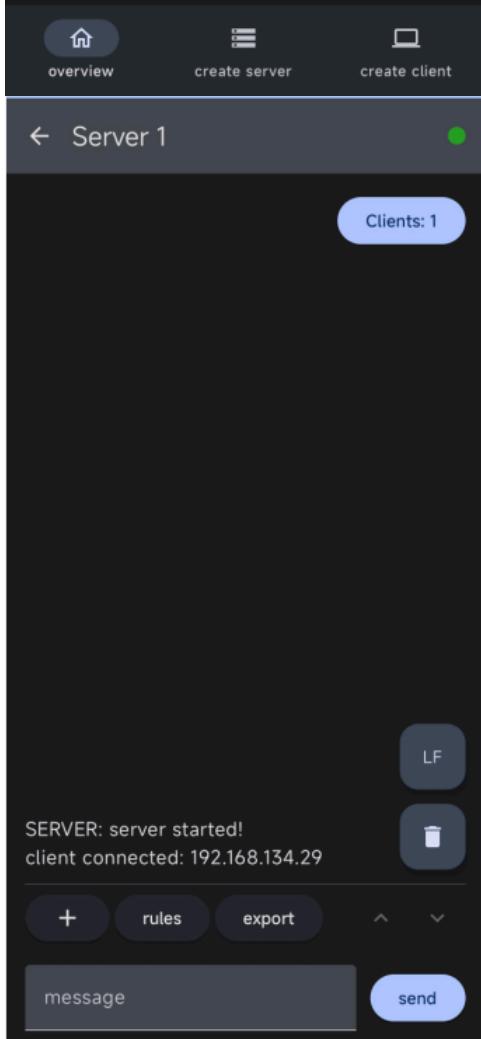
```

(/wiki/File:ESP32-S3-GEEK_Demo-IP07.png)

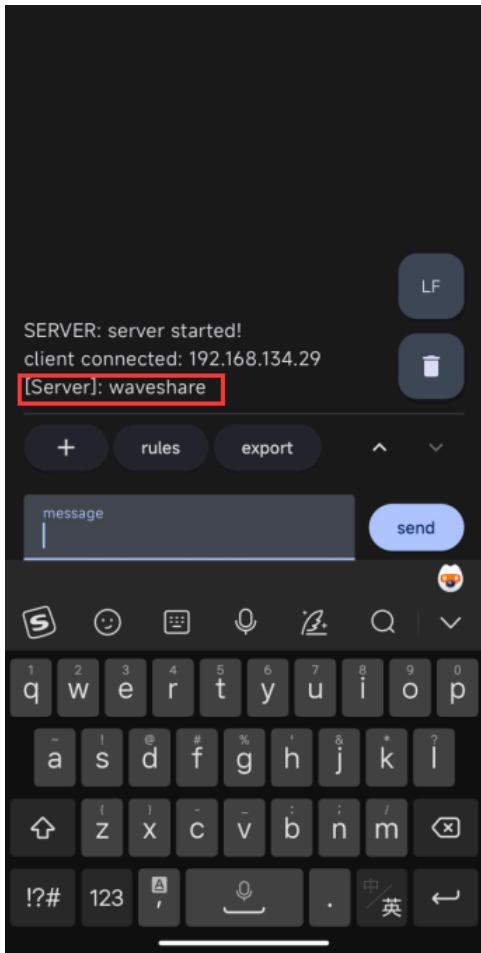
- After programming the demo, once connected, the TCP Server will receive a TCP message "Hello world" from ESP32-S3-GEEK. The LCD will display "Access successful". Using a mobile device's TCP Server, you can send TCP messages to ESP32-S3-GEEK. If the transmission is successful, ESP32-S3-GEEK, acting as a TCP Client, will receive the message and display its content on the LCD. You could observe whether the LCD shows the message.



(/wiki/File:ESP32-S3-GEEK_Demo-ip08.png)



(/wiki/File:ESP32-S3-GEEK_Demo-ip09.png)



(/wiki/File:ESP32-S3-GEEK_Demo-ip10.png)

03-WIFI_TCP_Server

With this demo, ESP32-S3-GEEK enters WIFI's STA mode. After connecting the hotspot on the PC, creates a TCP Server, and the PC creates a TCP Client to access ESP-S3-GEEK. Then, the TCP communication is established between them, and GEEK displays the received content on the LCD.

- ESP32-S3-GEEK connects to the same WIFI on the STA mode. (The WIFI network that ESP32-S3-GEEK connects to should operate on the 2.4GHz frequency band. If there's no 2.4GHz frequency band available, the PC can create a hotspot with the network band set to "Any available frequenc".) Remember to change the ssid and password to match the WIFI name and password you wish to connect to.



The screenshot shows the Arduino IDE interface with the file `WIFI_TCP_Server.ino` open. The code is for a WiFi TCP Server. A red box highlights the following configuration code:

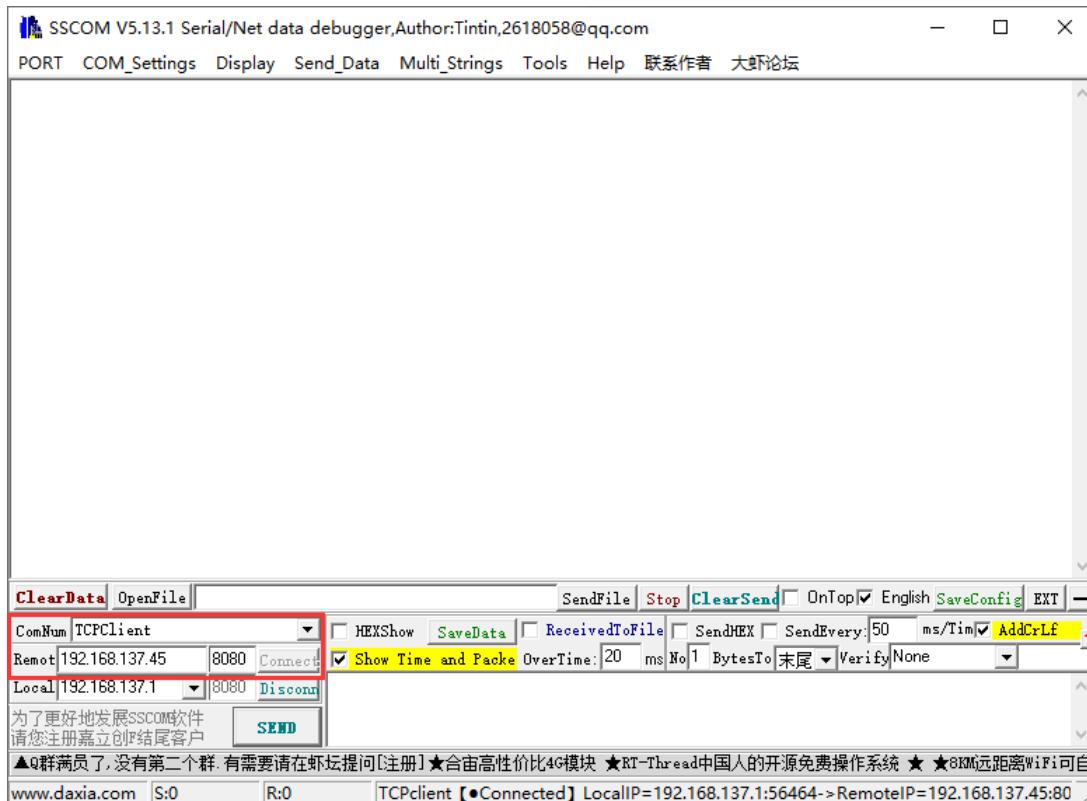
```

1 #include <SPI.h>
2 #include "LCD_Driver.h"
3 #include "GUI_Paint.h"
4 #include "image.h"
5 #include <WiFi.h>
6
7 const char *ssid = "ESP32-S3-GEEK";
8 const char *password = "Waveshare";
9 WiFiServer server; // Declares the server object
10
11 void intToIpAddress(uint32_t ip, char *result) {
12     sprintf(result, "%d.%d.%d.%d", ip & 255,(ip >> 8) & 255,(ip >
13 }

```

GEEK_Demo12.png)

- After connecting to the WiFi, **the LCD will display TCP ServerIP**, and use SSCOM to open TCP Client for ESP32-S3-GEEK(TCP Server) connection with TCP.



GEEK_Demo13.png)

- Send TCP message through TCP Client to ESP32-S3-GEEK(TCP Server), **check the "AddCrLf" options while sending the message.**



(/wiki/File:ESP32-S3-

GEEK_Demo14.png)

- After sending successfully, ESP32-S3-GEEK as the TCP server receives the message and displays it on the LCD, and you can check whether the LCD displays the received message.



(/wiki/File:ESP32-S3-

GEEK_Demo15.png)

04-WIFI_Web_Server

This code allows the ESP32-S3-GEEK to open its WIFI in AP mode. After a PC connects to this WIFI, open a serial debugging assistant. Through the HTTP webpage created by the ESP32-S3-GEEK, messages can be sent to the GEEK. Observe the received content on both the serial debugging assistant and the LCD.

- The "ssid" is the name of the AP created by the ESP32-S3-GEEK, and the "password" is the password required to connect to the AP.

```

WIFI_Web_Server | Arduino IDE 2.2.1
File Edit Sketch Tools Help
Select Board
WIFI_Web_Server.ino DEV_Config.cpp Debug.h GUI_Paint.cpp D ...
1 #include <SPI.h>
2 #include "LCD_Driver.h"
3 #include "GUI_Paint.h"
4 #include "image.h"
5 #include "WIFI_Driver.h"
6
7 // Set these to your desired credentials.
8 const char *ssid = "ESP32-S3-GEEK";
9 const char *password = "Waveshare";
10
11 WiFiServer server(80);
12 WiFiClient client;
13
14 void setup()

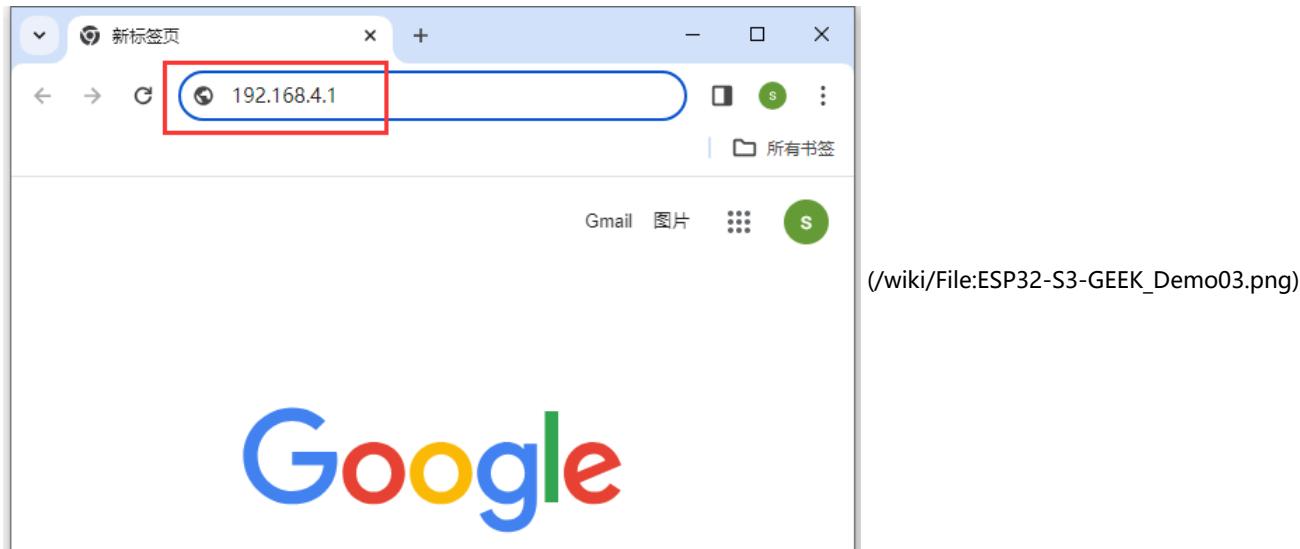
```

(/wiki/File:ESP32-S3-

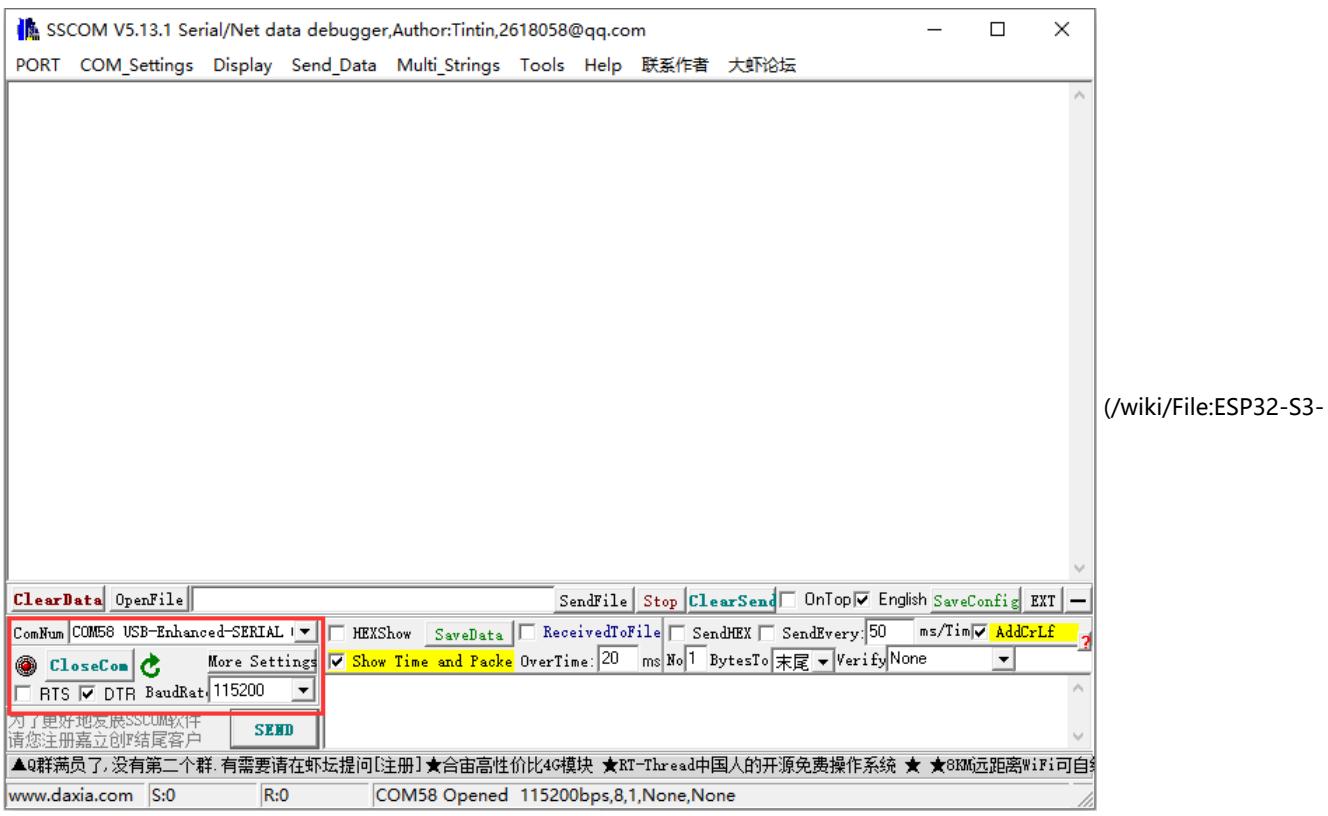
GEEK_Demo17.png)

- Using the PC to connect to the AP of the ESP32-S3-GEEK.

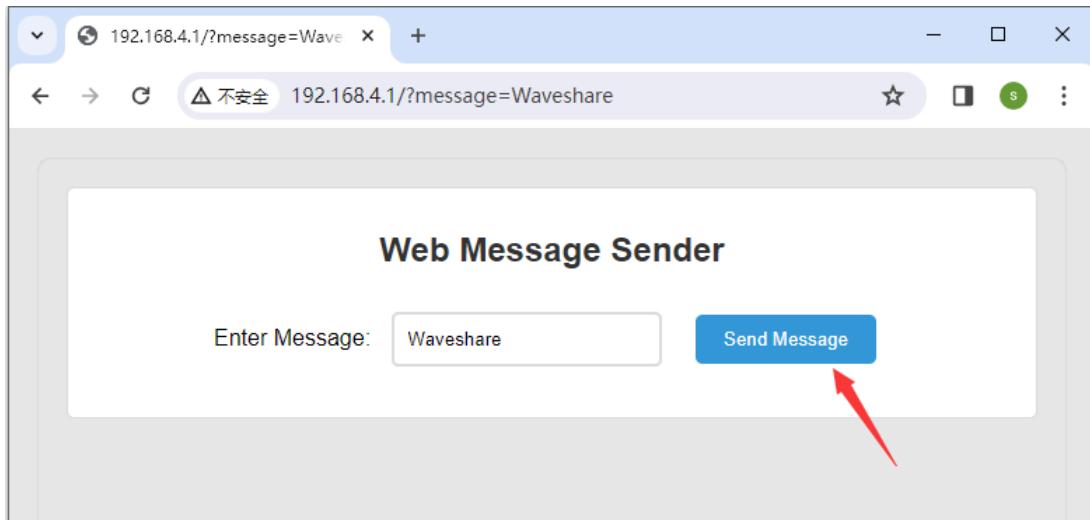
- The LCD will display the HTTP server's IP address. Use a web browser to log in to the IP address: 192.168.4.1.



- Using the ESP32-S3-GEEK's UART interface, connect it to the PC via a USB to UART tool (<https://www.waveshare.com/ch343-usb-uart-board.htm>), then open a serial debugging assistant.

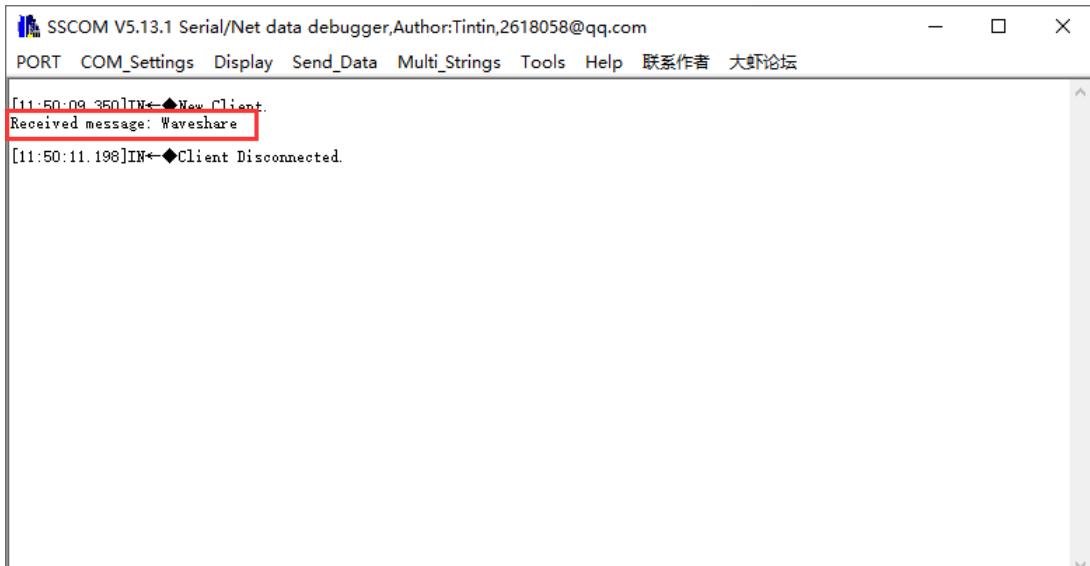


- You can input text content on the HTTP web interface and send an HTTP request to the ESP32-S3-GEEK. The received content can be displayed on both the serial debugging assistant and the LCD.



(/wiki/File:ESP32-S3-

GEEK_Demo21.png)



(/wiki/File:ESP32-S3-



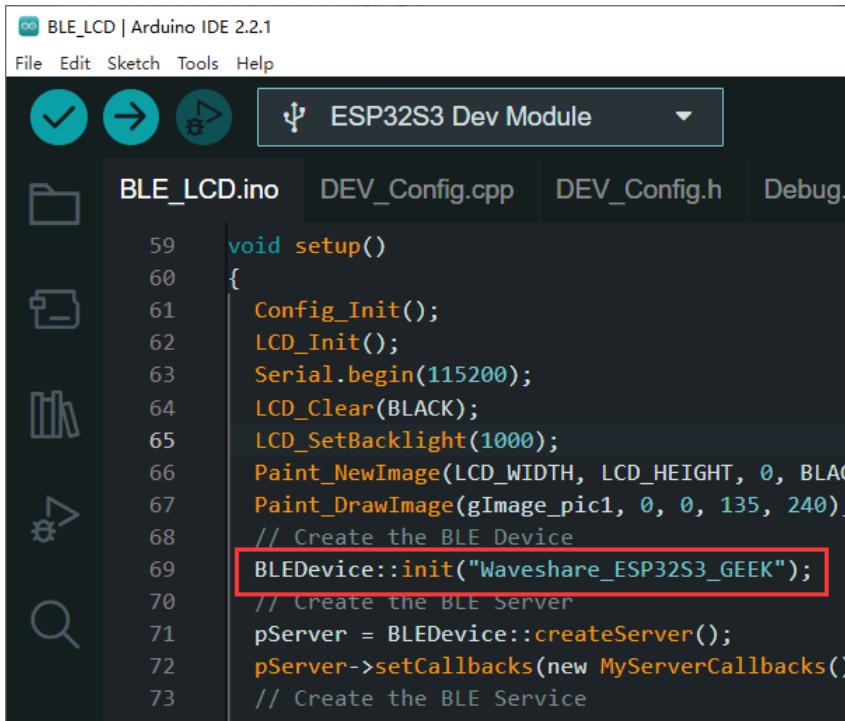
GEEK_Demo22.png)

BLE

01-BLE_LCD

With this demo, ESP32-S3-GEEK enters BLE. Using a mobile phone to open the Bluetooth debugging assistant, connect to ESP32-S3-GEEK, and establish BLE communication with the mobile phone. The message of sending and receiving is displayed on the LCD.

- In the code `BLEDevice::init("Waveshare_ESP32S3_GEEK")`, "Waveshare_ESP32S3_GEEK" is the Bluetooth device name.



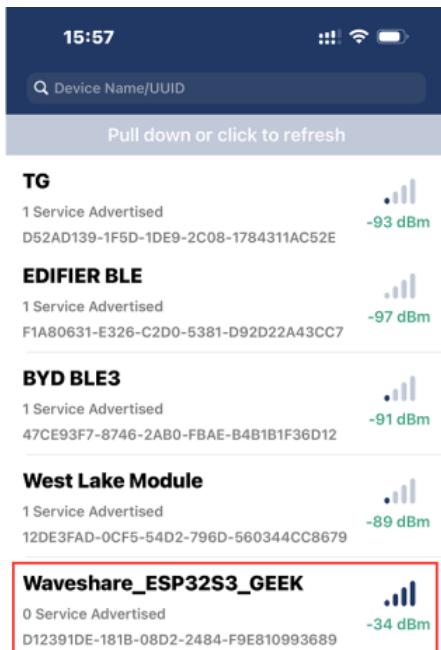
```

void setup()
{
    Config_Init();
    LCD_Init();
    Serial.begin(115200);
    LCD_Clear(BLACK);
    LCD_SetBacklight(1000);
    Paint_NewImage(LCD_WIDTH, LCD_HEIGHT, 0, BLACK);
    Paint_DrawImage(gImage_pic1, 0, 0, 135, 240);
    // Create the BLE Device
    BLEDevice::init("Waveshare_ESP32S3_GEEK");
    // Create the BLE Server
    pServer = BLEDevice::createServer();
    pServer->setCallbacks(new MyServerCallbacks());
    // Create the BLE Service
}

```

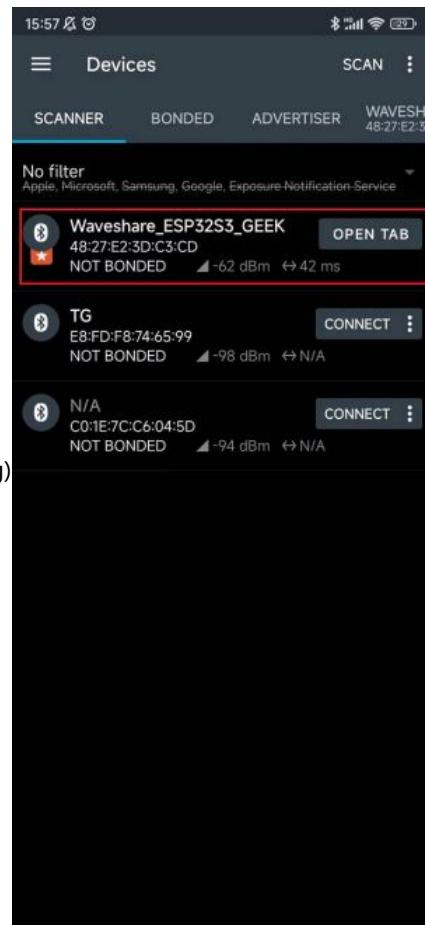
- Use the mobile phone to open the Bluetooth debugging assistant to scan and connect the device.

IOS:



Android:

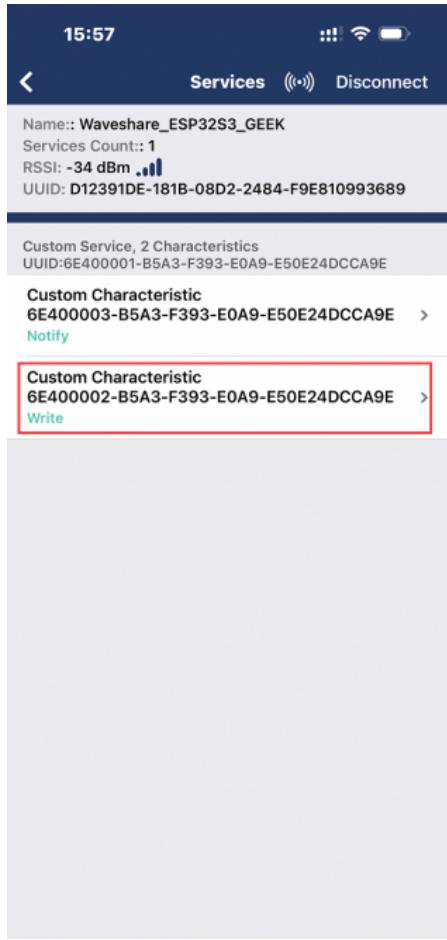
(/wiki/File:ESP32-S3-GEEK_Demo-21.png)



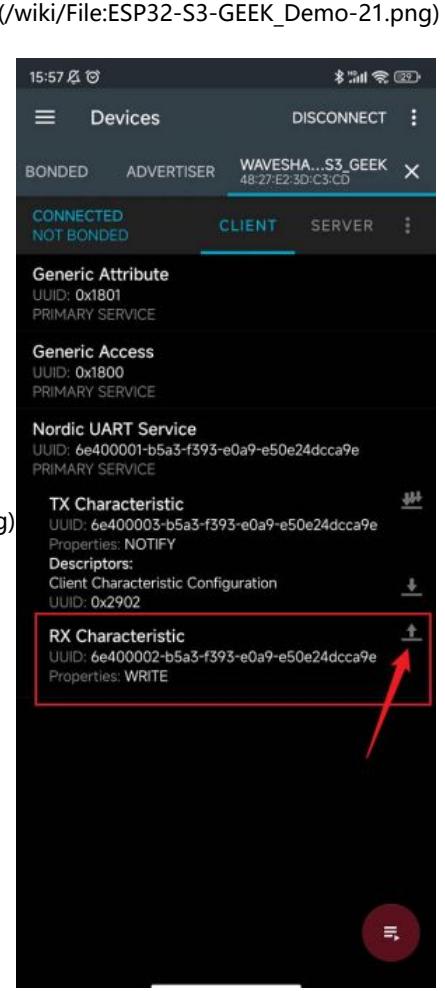
(/wiki/File:ESP32-S3-GEEK_Demo-122.jpg)

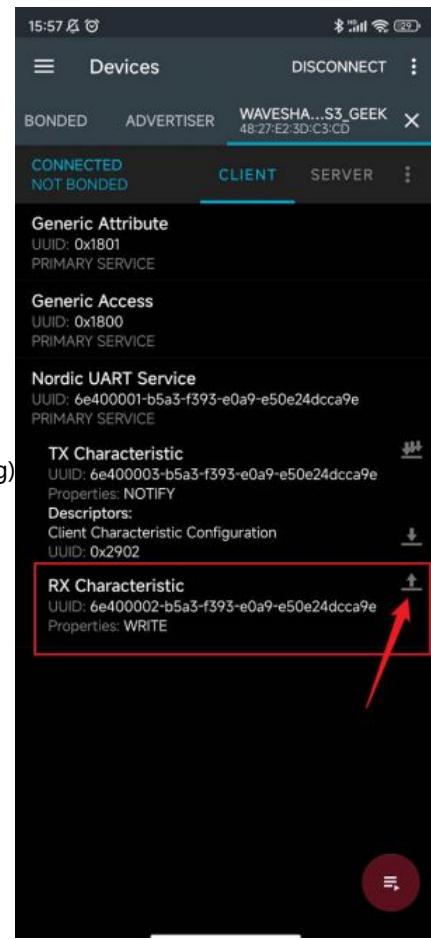
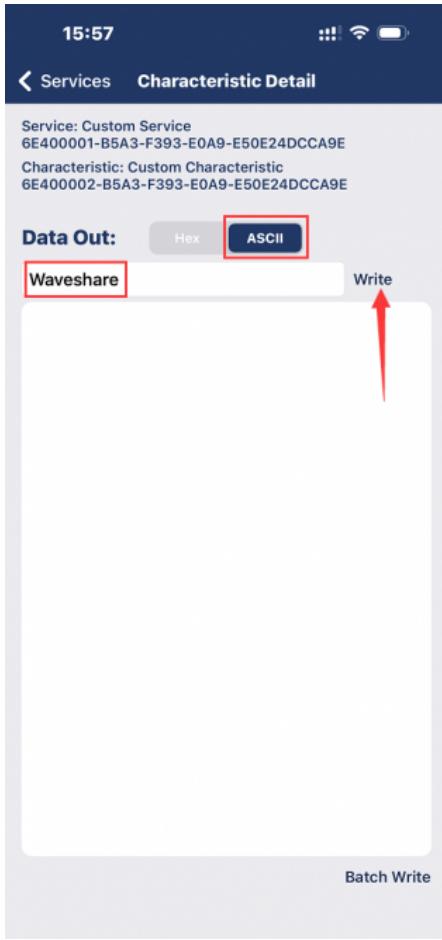
- Use the Bluetooth debugging assistant to send the message to ESP32-S3-GEEK, and the ESP32-S3-GEEK will display the message on the LCD after receiving it.

IOS:



Android:

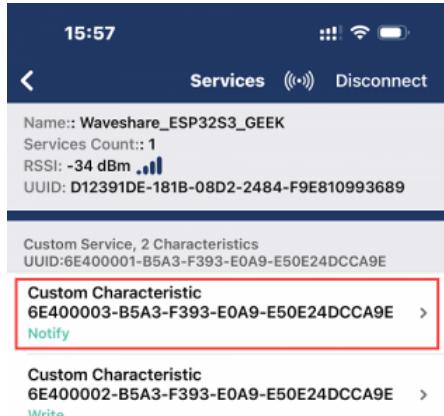




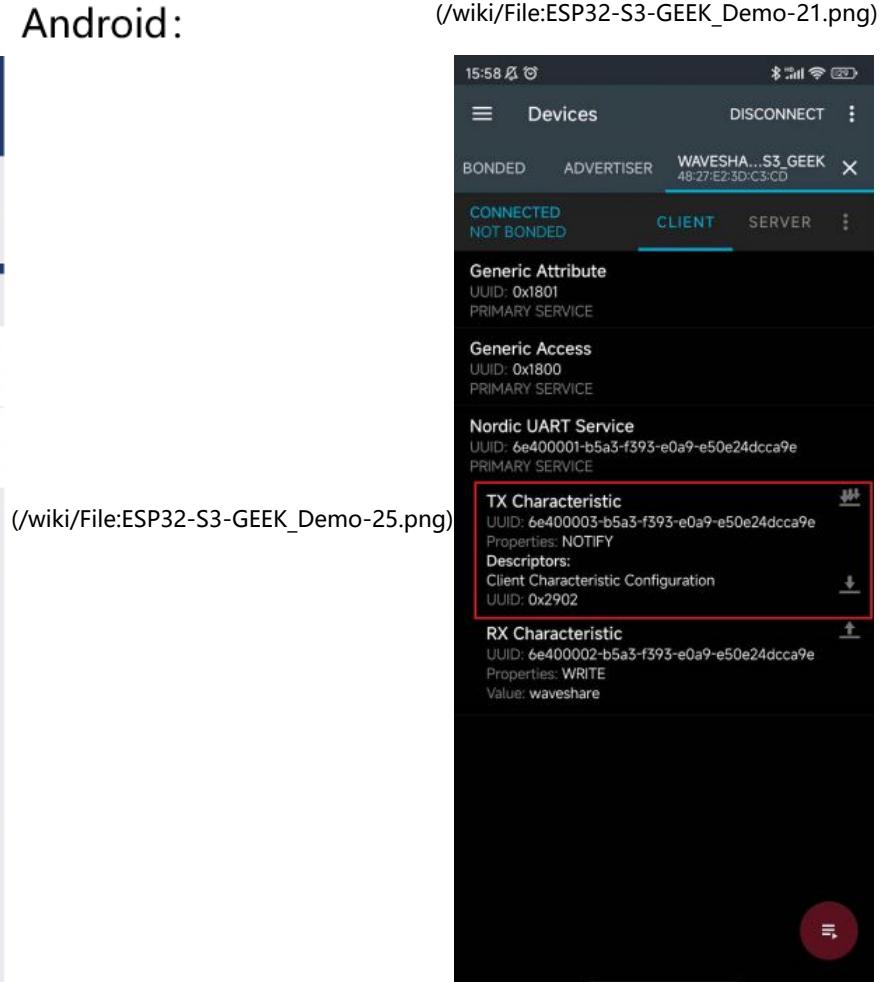
(/wiki/File:ESP32-S3-GEEK_Demo-24.jpg)

- Enter the receiving setting of the Bluetooth debugging assistant on the mobile phone:

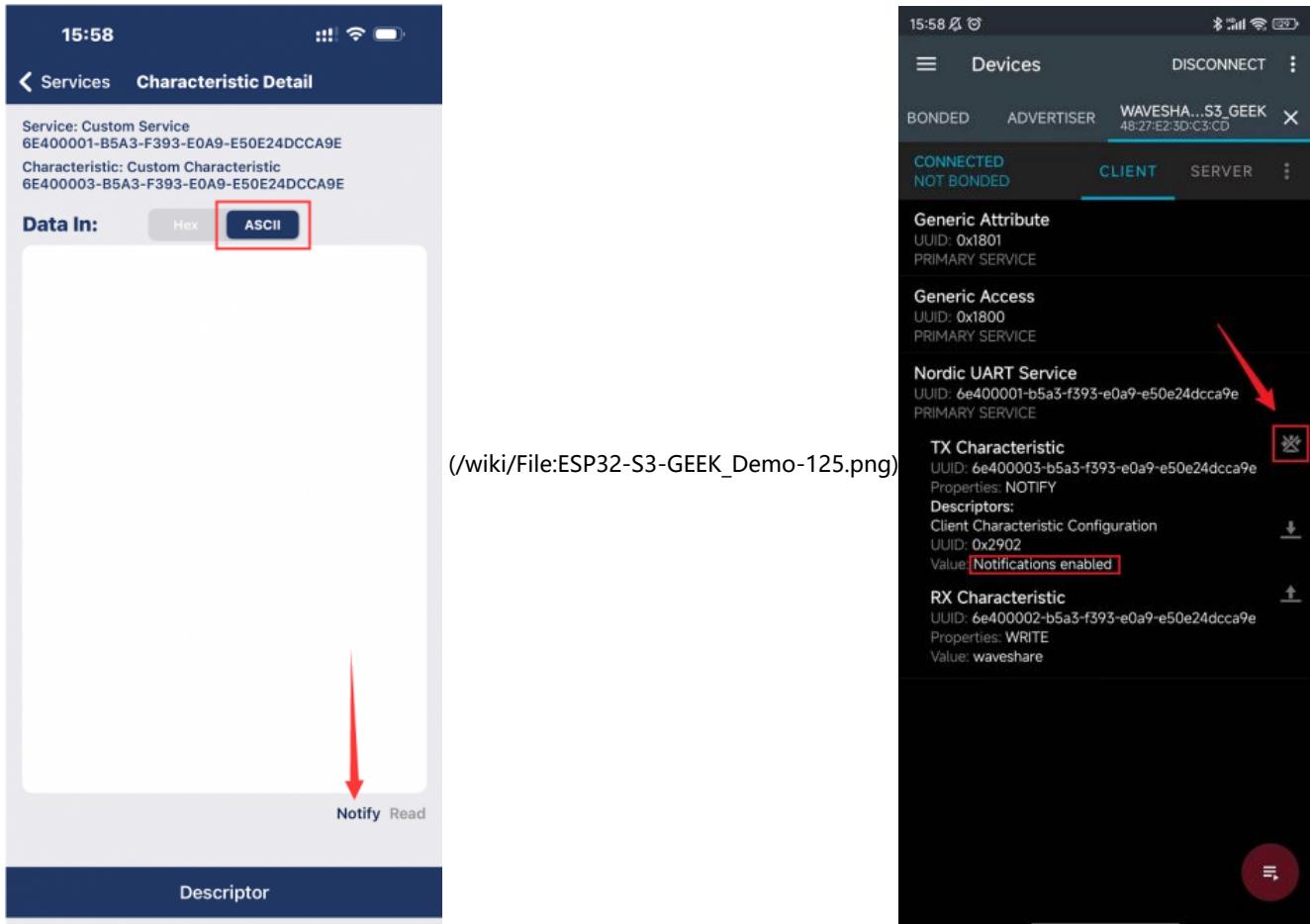
IOS:



Android:

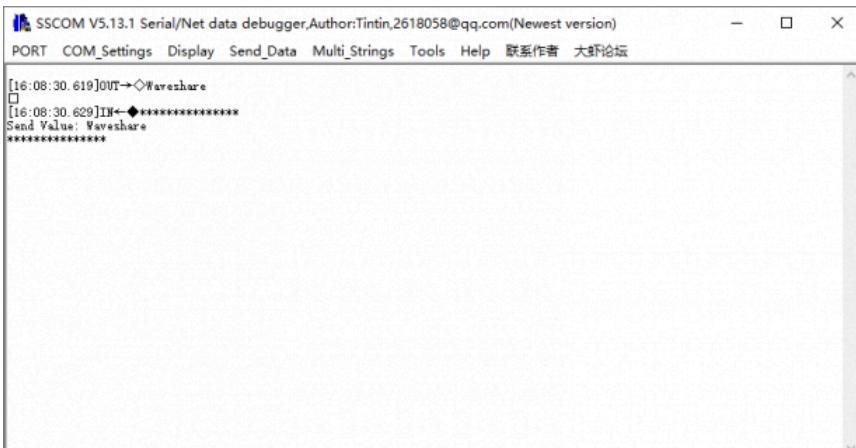


(/wiki/File:ESP32-S3-GEEK_Demo-26.jpg)



(/wiki/File:ESP32-S3-GEEK_Demo-125.png)

- ESP32-S3-GEEK uses USB to UART tool (<https://www.waveshare.com/ch343-usb-uart-board.htm>) to connect to the PC, open the serial debugging assistant, and send the UART message as a Bluetooth message to the mobile phone. Make sure to check "AddCrLf" options when sending the message. The sent message will be displayed on the LCD. On the mobile device, observe if it receives the Bluetooth message.



(/wiki/File:ESP32-S3-GEEK_Demo-29.png)



为了更好地发展SSCOM软件
请您注册嘉立创精英客户

▲QQ群成员了,没有第二个群,有需要请在虾社提问【注册】★合宙高性价比4G模块 ★RT-Thread中国人的开源免费操作系统 ★ ★远距离WiFi可自

www.daxia.com S:11 R:57 COM58 Opened 115200bps,8,1,None,None



(/wiki/File:ESP32-S3-GEEK_Demo-28.png)

(/wiki/File:ESP32-S3-GEEK_Demo-27.jpg)

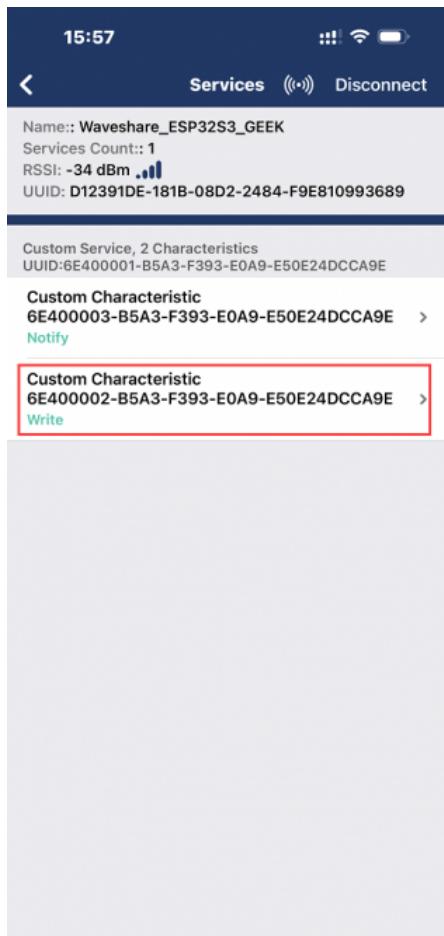
02-BLE_uart

This demo allows the ESP32-S3-GEEK to activate Bluetooth Low Energy (BLE). You can use a smartphone's Bluetooth debugging tool to connect to the ESP32-S3-GEEK, enabling BLE communication between the two devices. The sent and received messages will be displayed on the serial monitor, similar to BLE_LCD. However, this version doesn't activate the LCD, reducing power consumption significantly by using UART to display message content.

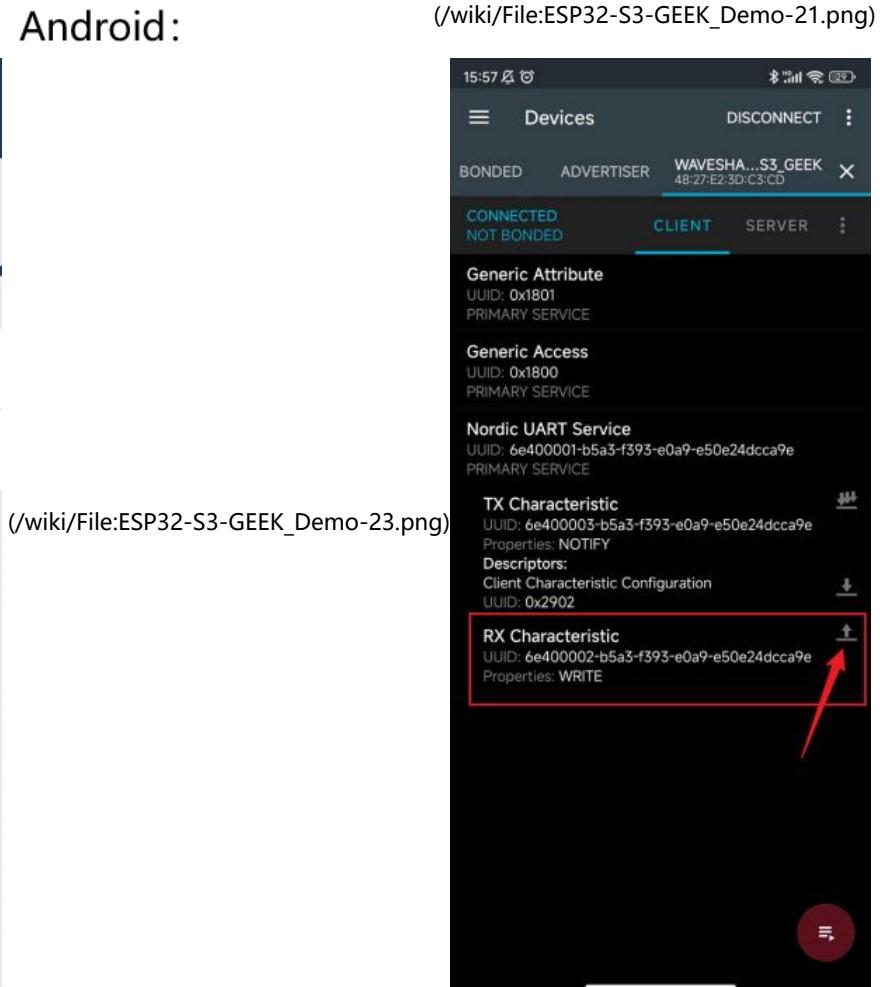
- In the code `BLEDevice::init("Waveshare_ESP32S3_GEEK")`, "Waveshare_ESP32S3_GEEK" is the Bluetooth device name.
- With a mobile Bluetooth debugging tool, scan and connect to devices.

- Using the tool, send Bluetooth messages from the mobile device to the ESP32-S3-GEEK. The received messages will be displayed on the serial debugging assistant of the ESP32-S3-GEEK.

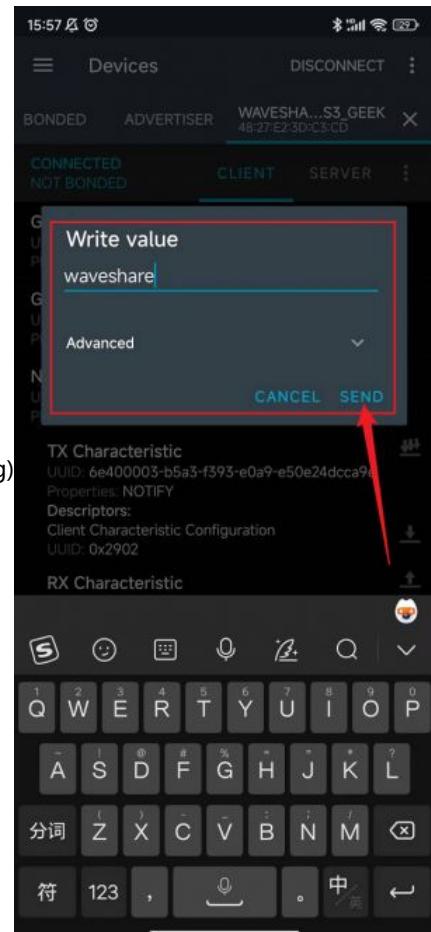
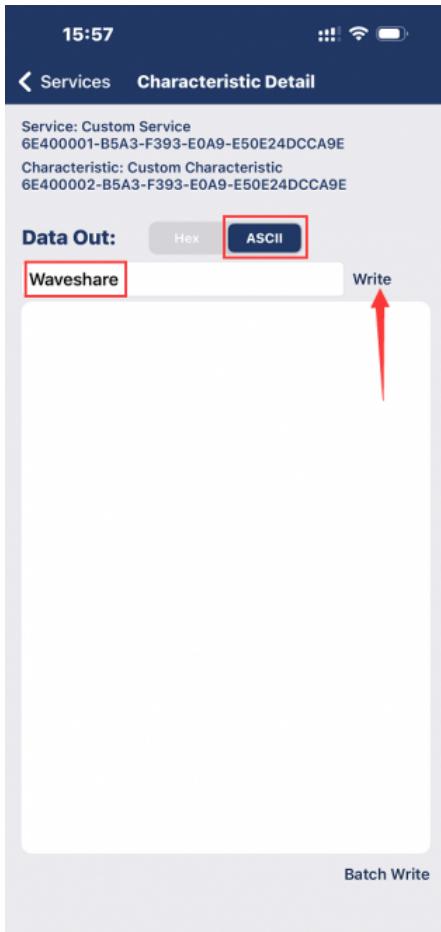
IOS:



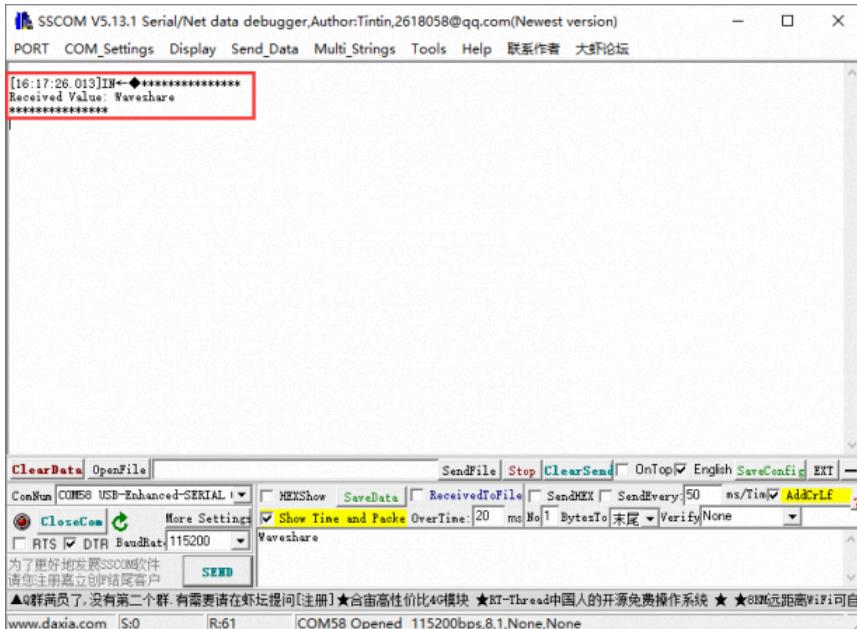
Android:



(/wiki/File:ESP32-S3-GEEK_Demo-24.jpg)

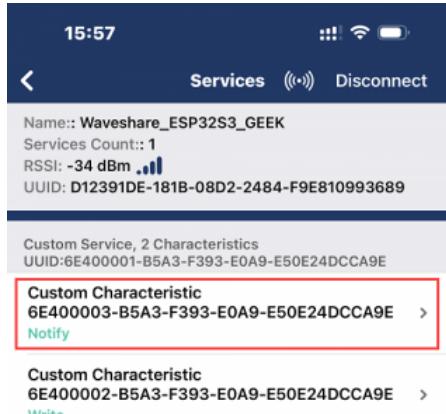


(/wiki/File:ESP32-S3-GEEK_Demo-133.jpg)

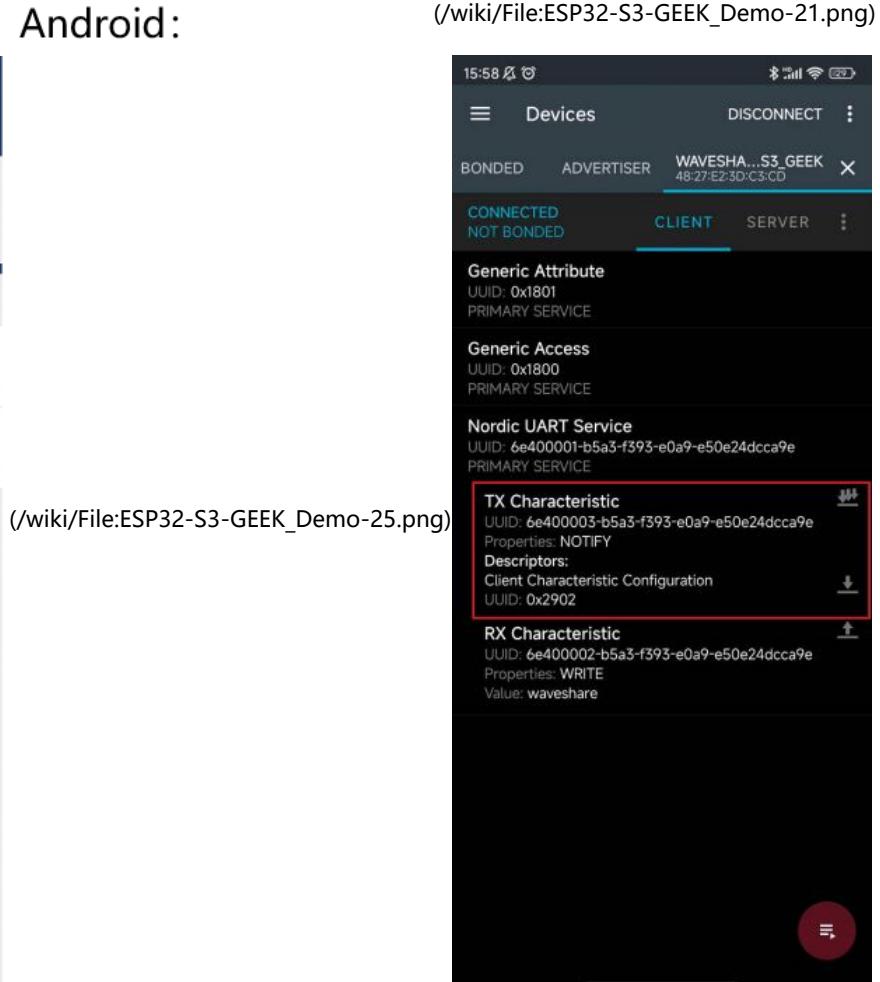


- Enter the receiving setting of the Bluetooth debugging tool on the phone.

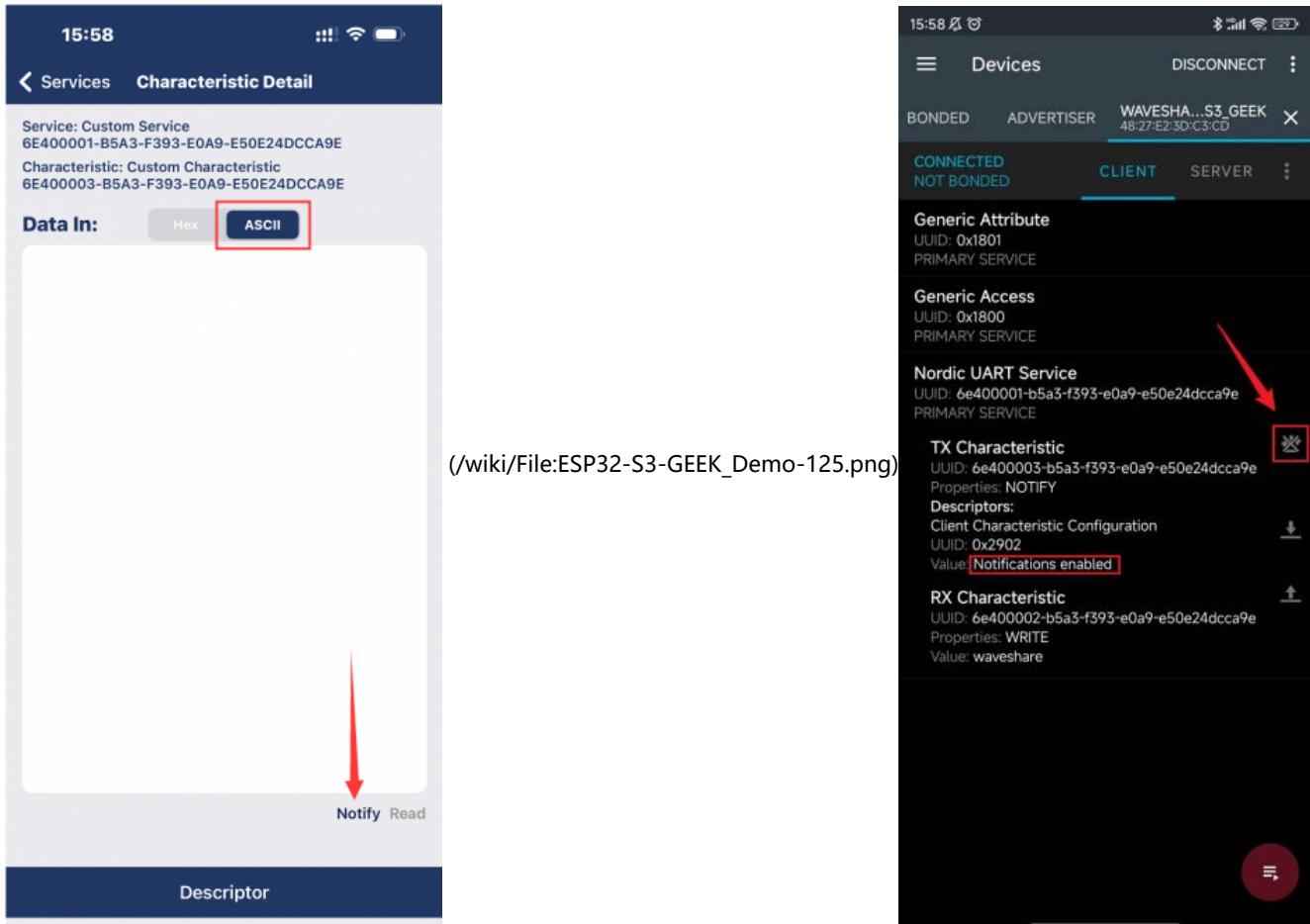
IOS:



Android:

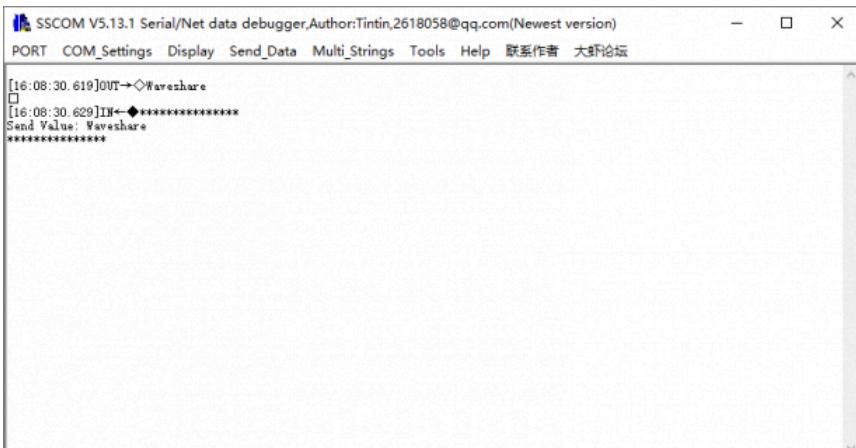


(/wiki/File:ESP32-S3-GEEK_Demo-26.jpg)

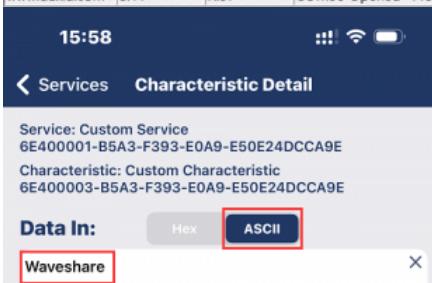
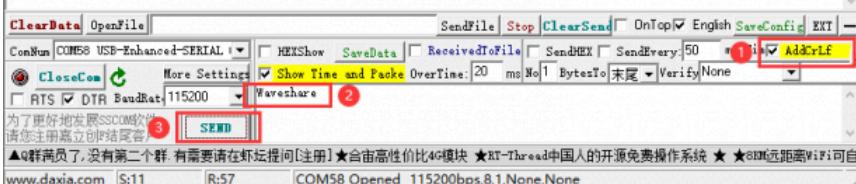


(/wiki/File:ESP32-S3-GEEK_Demo-125.png)

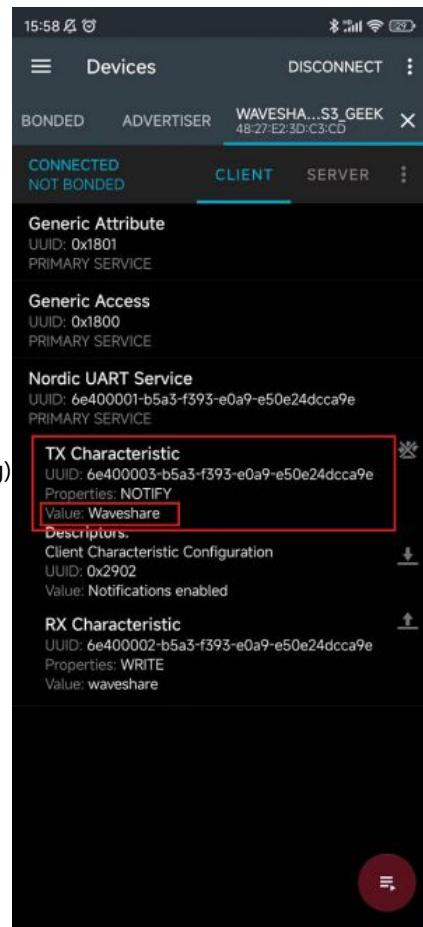
- ESP32-S3-GEEK, connected to the PC via CH343 USB UART Board (mini) (<https://www.waveshare.com/ch343-usb-uart-board.htm>), opens a serial debugging tool. Messages sent via serial are converted into Bluetooth messages to the phone. Ensure to check "AddCrLf". The message content will be displayed on the LCD. Check on the mobile device to observe if it receives the Bluetooth message.



(/wiki/File:ESP32-S3-GEEK_Demo-29.png)



(/wiki/File:ESP32-S3-GEEK_Demo-28.png)



(/wiki/File:ESP32-S3-GEEK_Demo-27.jpg)

03-BLE_Keyboard

This code enables the ESP32-S3-GEEK to function as a Bluetooth keyboard. Once connected via the PC's Bluetooth, you can perform a series of single-key or combination-key operations. The ESP32-S3-GEEK will simulate these key presses and transmit corresponding keyboard inputs to the connected device.

- Before using BLE_Keyboard, you need to add the ESP32-BLE-Keyboard Arduino library folder from the libraries directory into the libraries folder within the installation directory of Arduino IDE.



The screenshot shows the Arduino IDE interface. The title bar reads "BLE_Keyboard | Arduino IDE 2.2.1". The "File" menu is open, with the "Preferences..." option highlighted by a red box and a red number "2" indicating it's the second step in a two-step process. The main code editor area displays a sketch titled "ESP32S3 Dev Module" which turns the ESP32 into a Bluetooth LE keyboard. The code includes imports for <BLEKeyboard.h>, initializes a BLEKeyboard object, and sets up a loop to check for connections and print messages to Serial.

```

ESP32S3 Dev Module

example turns the ESP32 into a Bluetooth LE keyboard that writes to a host computer.

<BLEKeyboard.h>

ard bleKeyboard("ESP32-S3-GEEK", "Waveshare", 100);

up() {
.begin(115200);
.println("Starting BLE work!");

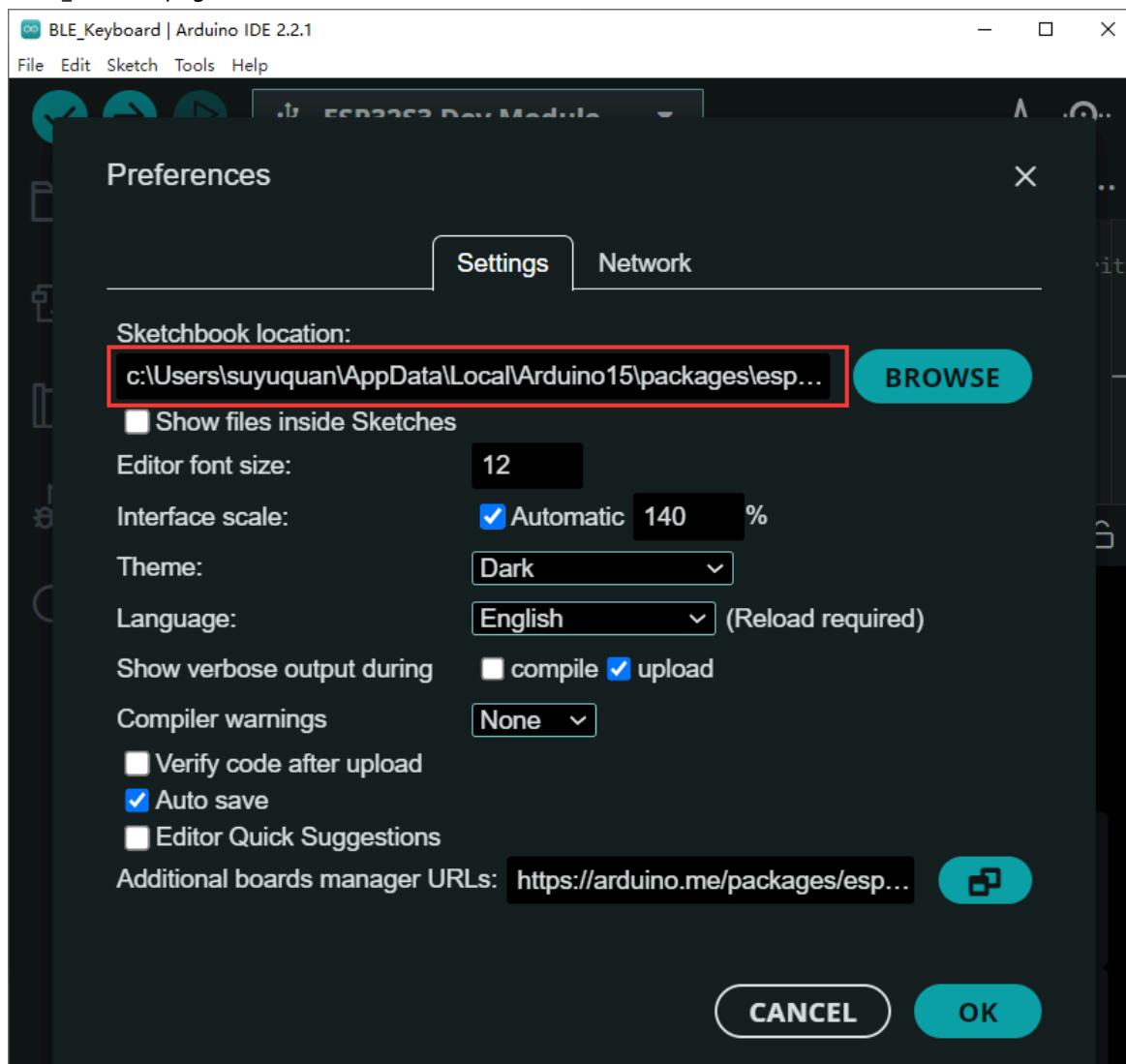
bleKeyboard.begin();

}

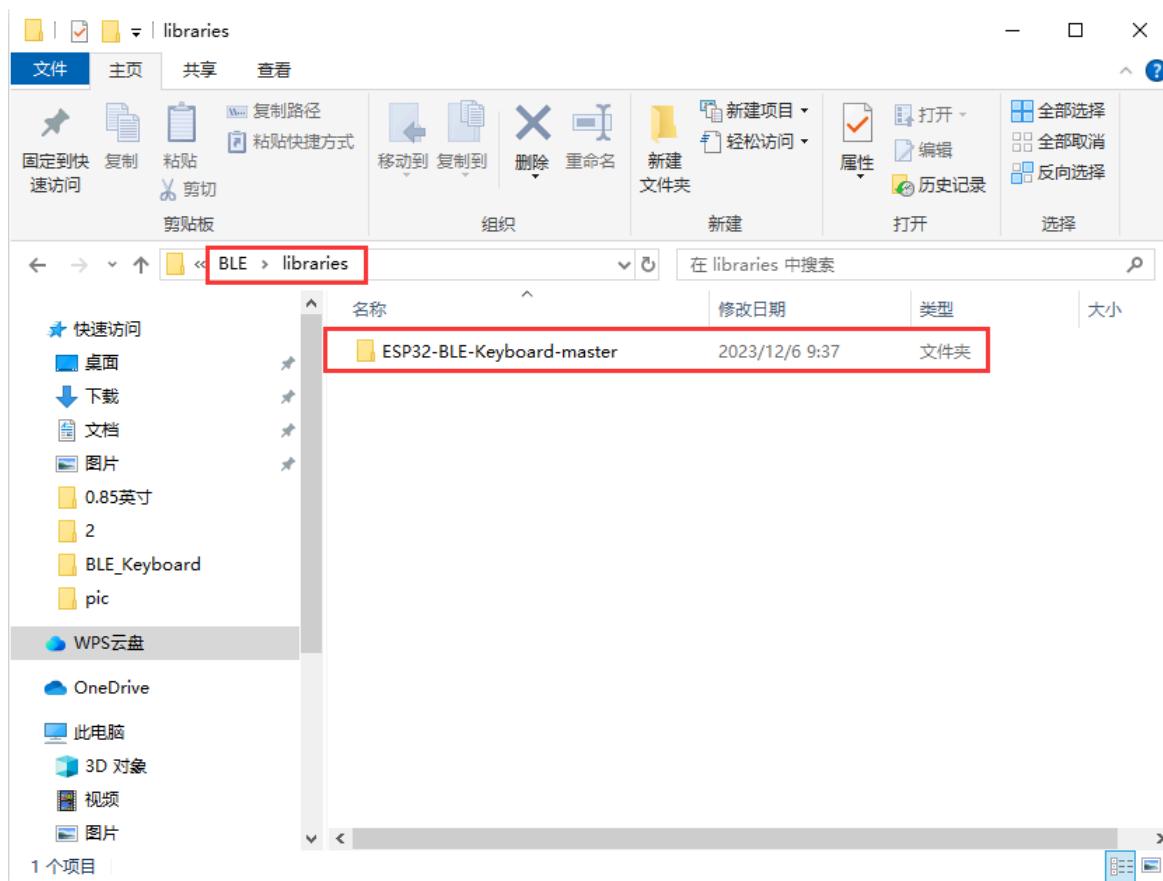
void loop() {
if(bleKeyboard.isConnected()) {
Serial.println("Sending 'Waveshare' ...");
}
}

```

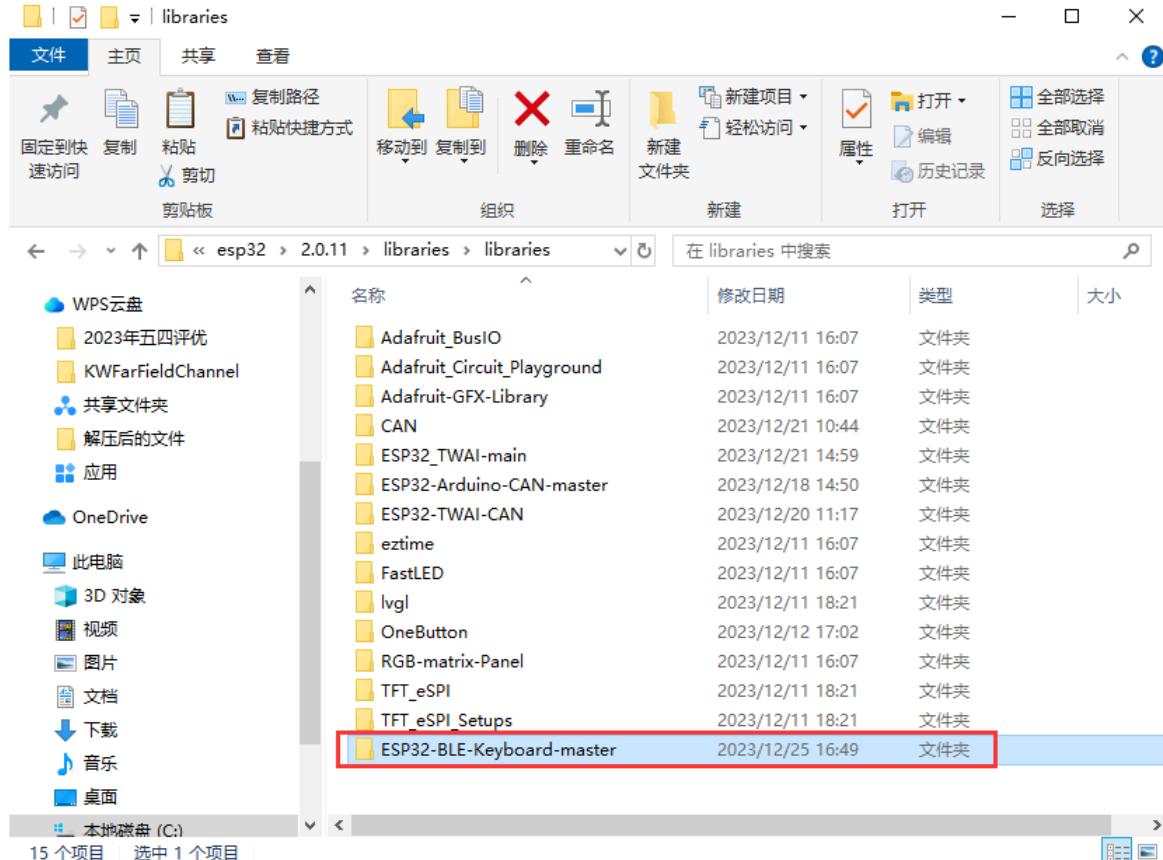
GEEK_Demo40.png



GEEK_Demo41.png



GEEK_Demo42.png)



GEEK_Demo43.png)

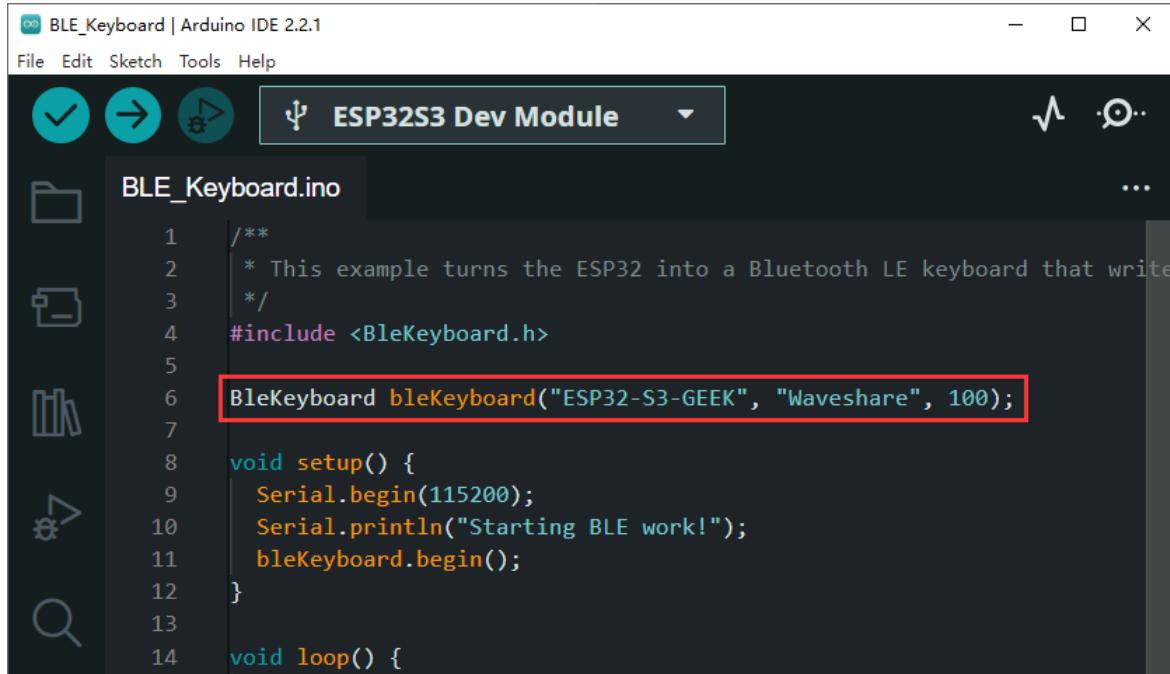
- If the Arduino library of the ESP32-BLE-Keyboard has been installed, pay attention to modifying the BleKeyboard.c file as shown below:

```
BLESecurity *pSecurity = new BLESecurity();
pSecurity->setAuthenticationMode(ESP_LE_AUTH_REQ_SC_MITM_BOND);
```

Modify it as:

```
BLESecurity *pSecurity = new BLESecurity();
pSecurity->setAuthenticationMode(ESP_LE_AUTH_BOND);
```

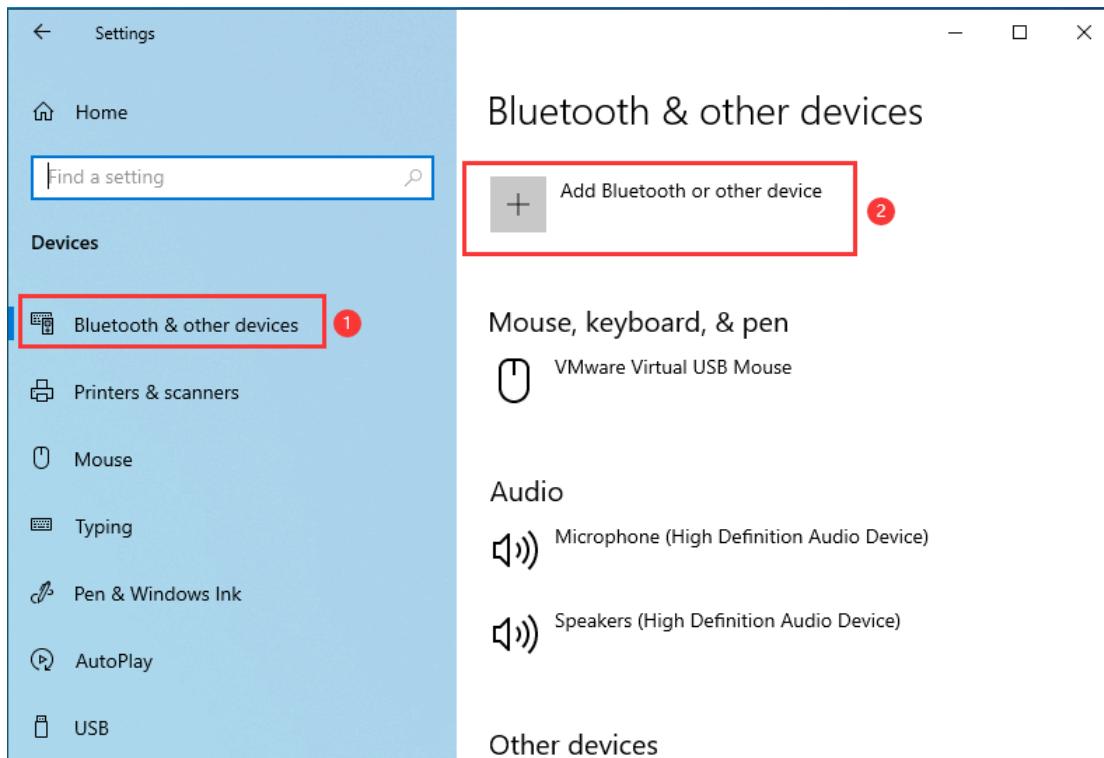
- ESP32-S3-GEEK is the name of the Bluetooth keyboard:



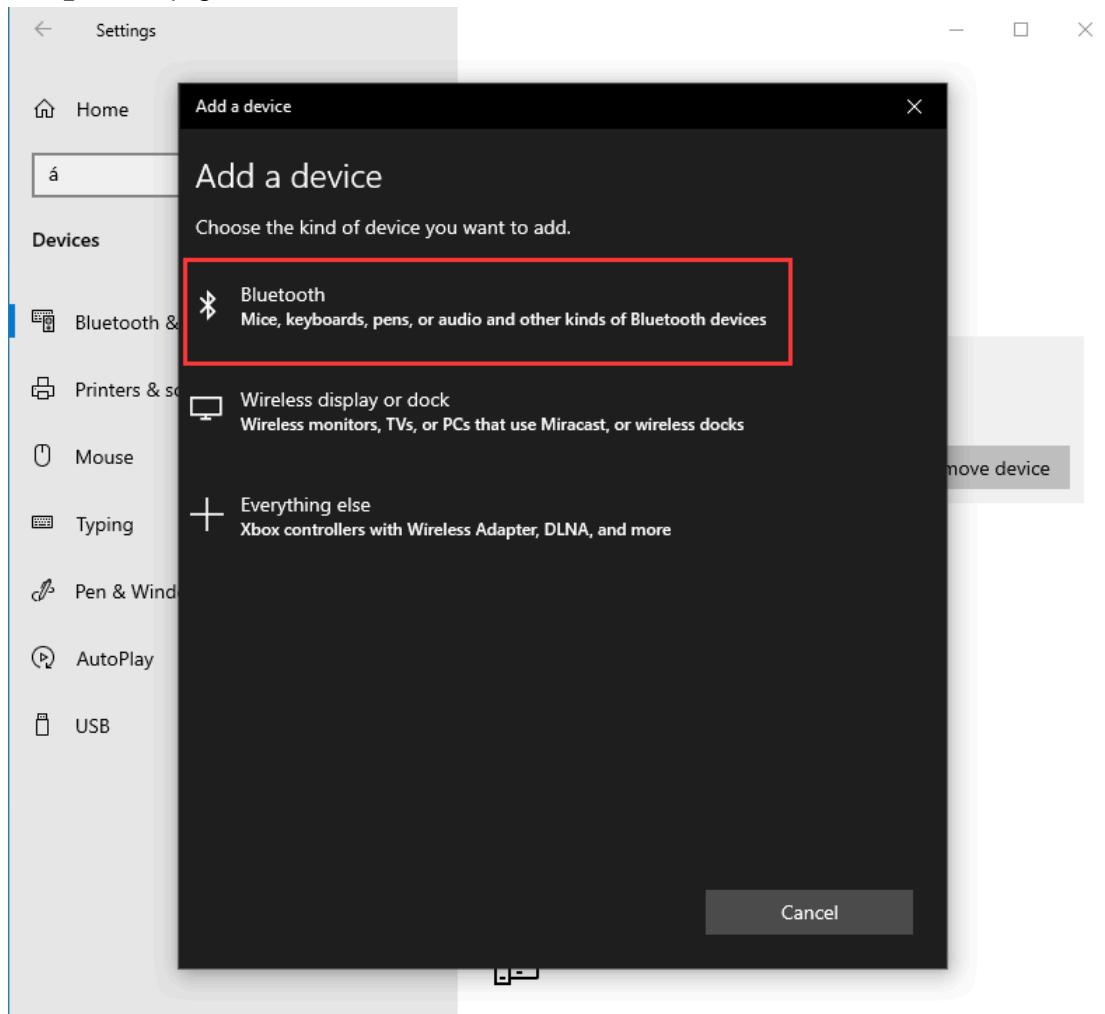
```
BLEKeyboard bleKeyboard("ESP32-S3-GEEK", "Waveshare", 100);
```

GEEK_Demo45.png)

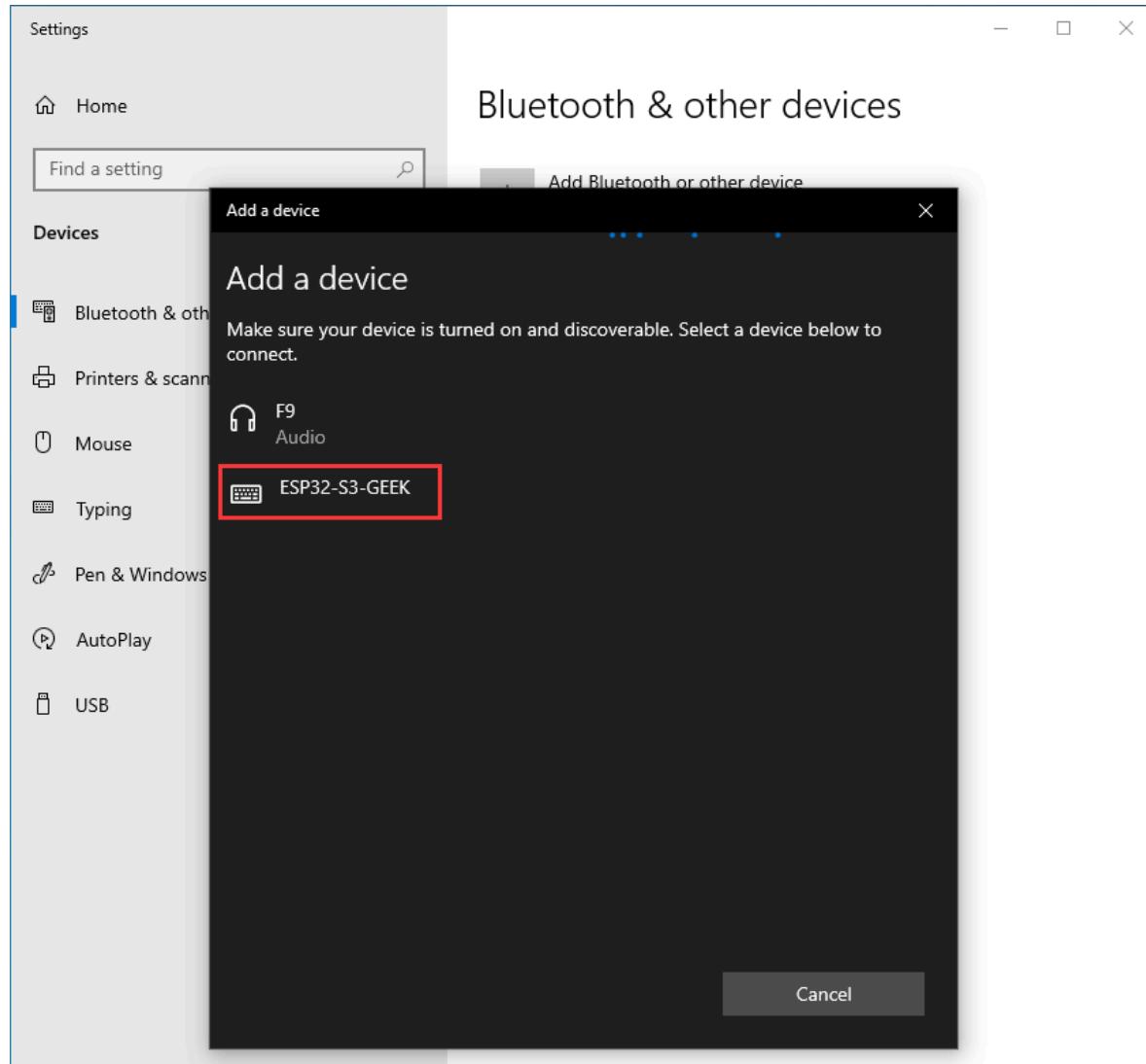
- Use PC to enter Bluetooth to scan and connect the device:



GEEK_Demo46.png)



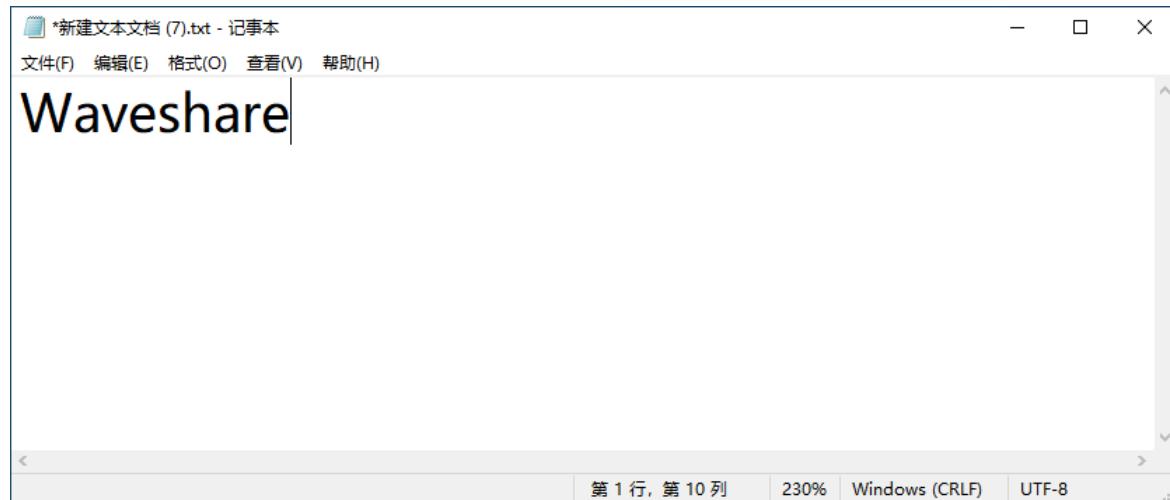
GEEK_Demo47.png)



(/wiki/File:ESP32-S3-

GEEK_Demo48.png)

- After a successful connection, a series of keyboard actions are performed every 5 seconds (output "Waveshare", Ctrl+Alt+Delete).



(/wiki/File:ESP32-S3-

GEEK_Demo49.png)

- Modify the code with your unlock code to enable automatic locking and unlocking operations.

BLE_Keyboard | Arduino IDE 2.2.1

File Edit Sketch Tools Help

ESP32S3 Dev Module

BLE_Keyboard.ino

```
8 void setup() {  
9     Serial.begin(115200);  
10    Serial.println("Starting BLE work!");  
11    bleKeyboard.begin();  
12 }  
13  
14 void loop() {  
15     if(bleKeyboard.isConnected()) {  
16         Serial.println("Sending 'Waveshare'...");  
17         bleKeyboard.print("Waveshare");  
18         delay(1000);  
19         Serial.println("Sending Enter key...");  
20         bleKeyboard.write(KEY_RETURN);  
21     }  
22 }  
23
```

(/wiki/File:ESP32-S3-

GEEK_Demo50.png)

Can be changed to
your screen password

- You can view the value of each key in the BleKeyboard.h file of the libraries file folder.

```
const uint8_t KEY_LEFT_CTRL = 0x80;
const uint8_t KEY_LEFT_SHIFT = 0x81;
const uint8_t KEY_LEFT_ALT = 0x82;
const uint8_t KEY_LEFT_GUI = 0x83;
const uint8_t KEY_RIGHT_CTRL = 0x84;
const uint8_t KEY_RIGHT_SHIFT = 0x85;
const uint8_t KEY_RIGHT_ALT = 0x86;
const uint8_t KEY_RIGHT_GUI = 0x87;

const uint8_t KEY_UP_ARROW = 0xDA;
const uint8_t KEY_DOWN_ARROW = 0xD9;
const uint8_t KEY_LEFT_ARROW = 0xD8;
const uint8_t KEY_RIGHT_ARROW = 0xD7;
const uint8_t KEY_BACKSPACE = 0xB2;
const uint8_t KEY_TAB = 0xB3;
const uint8_t KEY_RETURN = 0xB0;
const uint8_t KEY_ESC = 0xB1;
const uint8_t KEY_INSERT = 0xD1;
const uint8_t KEY_PRTSC = 0xCE;
const uint8_t KEY_DELETE = 0xD4;
const uint8_t KEY_PAGE_UP = 0xD3;
const uint8_t KEY_PAGE_DOWN = 0xD6;
const uint8_t KEY_HOME = 0xD2;
const uint8_t KEY_END = 0xD5;
const uint8_t KEY_CAPS_LOCK = 0xC1;
const uint8_t KEY_F1 = 0xC2;
const uint8_t KEY_F2 = 0xC3;
const uint8_t KEY_F3 = 0xC4;
const uint8_t KEY_F4 = 0xC5;
const uint8_t KEY_F5 = 0xC6;
const uint8_t KEY_F6 = 0xC7;
const uint8_t KEY_F7 = 0xC8;
const uint8_t KEY_F8 = 0xC9;
const uint8_t KEY_F9 = 0xCA;
const uint8_t KEY_F10 = 0xCB;
const uint8_t KEY_F11 = 0xCC;
const uint8_t KEY_F12 = 0xCD;
const uint8_t KEY_F13 = 0xF0;
const uint8_t KEY_F14 = 0xF1;
const uint8_t KEY_F15 = 0xF2;
const uint8_t KEY_F16 = 0xF3;
const uint8_t KEY_F17 = 0xF4;
const uint8_t KEY_F18 = 0xF5;
const uint8_t KEY_F19 = 0xF6;
const uint8_t KEY_F20 = 0xF7;
const uint8_t KEY_F21 = 0xF8;
const uint8_t KEY_F22 = 0xF9;
const uint8_t KEY_F23 = 0xFA;
const uint8_t KEY_F24 = 0xFB;

const uint8_t KEY_NUM_0 = 0xEA;
const uint8_t KEY_NUM_1 = 0xE1;
const uint8_t KEY_NUM_2 = 0xE2;
const uint8_t KEY_NUM_3 = 0xE3;
const uint8_t KEY_NUM_4 = 0xE4;
const uint8_t KEY_NUM_5 = 0xE5;
const uint8_t KEY_NUM_6 = 0xE6;
const uint8_t KEY_NUM_7 = 0xE7;
const uint8_t KEY_NUM_8 = 0xE8;
const uint8_t KEY_NUM_9 = 0xE9;
const uint8_t KEY_NUM_SLASH = 0xDC;
const uint8_t KEY_NUM_ASTERISK = 0xDD;
const uint8_t KEY_NUM_MINUS = 0xDE;
const uint8_t KEY_NUM_PLUS = 0xDF;
const uint8_t KEY_NUM_ENTER = 0xE0;
const uint8_t KEY_NUM_PERIOD = 0xEB;
```

```
typedef uint8_t MediaKeyReport[2];

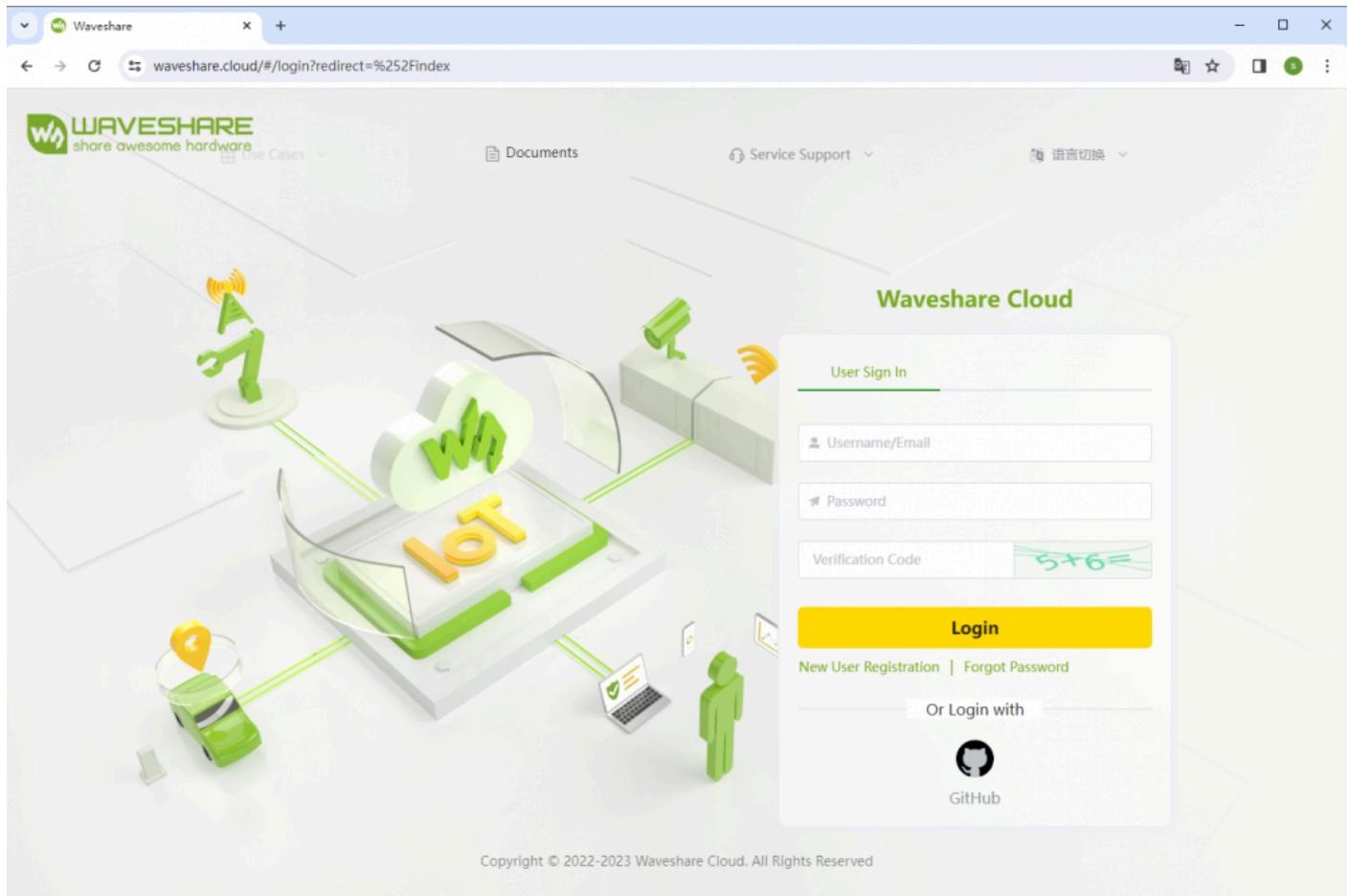
const MediaKeyReport KEY_MEDIA_NEXT_TRACK = {1, 0};
const MediaKeyReport KEY_MEDIA_PREVIOUS_TRACK = {2, 0};
const MediaKeyReport KEY_MEDIA_STOP = {4, 0};
const MediaKeyReport KEY_MEDIA_PLAY_PAUSE = {8, 0};
const MediaKeyReport KEY_MEDIA_MUTE = {16, 0};
const MediaKeyReport KEY_MEDIA_VOLUME_UP = {32, 0};
const MediaKeyReport KEY_MEDIA_VOLUME_DOWN = {64, 0};
const MediaKeyReport KEY_MEDIA_WWW_HOME = {128, 0};
const MediaKeyReport KEY_MEDIA_LOCAL_MACHINE_BROWSER = {0, 1}; // Opens "My Computer" on Windows
const MediaKeyReport KEY_MEDIA_CALCULATOR = {0, 2};
const MediaKeyReport KEY_MEDIA_WWW_BOOKMARKS = {0, 4};
const MediaKeyReport KEY_MEDIA_WWW_SEARCH = {0, 8};
const MediaKeyReport KEY_MEDIA_WWW_STOP = {0, 16};
const MediaKeyReport KEY_MEDIA_WWW_BACK = {0, 32};
const MediaKeyReport KEY_MEDIA_CONSUMER_CONTROL_CONFIGURATION = {0, 64}; // Media Selection
const MediaKeyReport KEY_MEDIA_EMAIL_READER = {0, 128};
```

MQTT

01-MQTT_sub_pub

The demo can use ESP32-S3-GEEK to open the STA mode of WIFI, connect to the WIFI, and then use the Waveshare cloud platform to carry out MQTT communication, subscribe and publish topics, and realize the long-distance transmission of information.

- When you are the first time to register on the Waveshare cloud (<https://waveshare.cloud/#/login?redirect=%252Findex>), you need to create a new device according to the tutorial (<https://mqtt.waveshare.cloud/en/wavesharecloud-wiki.html>).



(/wiki/File:ESP32-S3-GEEK_SUB.png)

This screenshot shows the 'Devices | Attributes' section of the Waveshare Cloud interface. The left sidebar has a red box around the 'Devices | Attributes' item, with a red number 1 next to it. The main area has tabs for 'Dashboard', 'Devices | Attributes' (which is active and highlighted with a red box and red number 2), and 'DevicesTypes'. A modal window titled 'Rapid additions' is open, showing a dropdown for 'Common Device Template' set to 'Please Select' (circled with red number 2) and a dropdown for 'Enter Device Name' with options like 'Modbus POE ETH Relay', 'SIM7028 NB-IoT HAT', 'Raspberry Pi 5', 'ESP32-S3-Relay-6CH', and 'ESP32-S3-GEEK' (circled with red number 3). The table below the modal shows one entry: id 1 and devicesType ESP32.

(/wiki/File:ESP32-S3-GEEK_SUB01.png)

- After creating, you can set the corresponding configuration in the Arduino IDE according to ESP32SDK tutorial (<https://mqtt.waveshare.cloud/en/esp32-connect.html>).

- After configuration, you can find the Client ID, Pub Topic, and Sub Topic of the newly created device on Waveshare's cloud platform under "View Address". You can assign these values in the code to facilitate ESP32-S3-GEEK's connection to your cloud platform device.
- Ensure that ESP32-S3-GEEK is set to STA mode and connected to the same Wi-Fi network as your device. (The Wi-Fi network ESP32-S3-GEEK connects to needs to support the 2.4GHz frequency band. In case it doesn't, you can use a PC to create a hotspot with the network frequency set to "Any available frequency".) Remember to modify the ssid and password with the name and password of the Wi-Fi network you want to connect to.

The screenshot shows the Waveshare cloud interface with a device list. The device table has columns: id, devicesType, Device name, description, Online status, Last up time, and Operation. One row is visible for an ESP32 device named "ESP32-S3-GEEK Demo" with the status "notactivated". The "Operation" column contains four icons: a blue edit icon, a green info icon, a yellow edit icon (which is highlighted with a red box), and a red delete icon. A red arrow points from the bottom right towards the yellow edit icon.

id	devicesType	Device name	description	Online status	Last up time	Operation
1	ESP32	ESP32-S3-GEEK Demo	ESP32-S3-GEEK Demo	notactivated		Edit Info Edit Delete

(/wiki/File:ESP32-S3-GEEK_SUB02.png)

The screenshot shows the Arduino IDE with the sketch "MQTT_sub_pub.ino". The code includes #include statements for ArduinoJson.h, Arduino.h, PubSubClient.h, WiFi.h, and WiFiClientSecure.h. It defines constants DEV_BL_PIN (7) and MSG_BUFFER_SIZE (50). Two sections of code are highlighted with red boxes:

- Line 10: // The name and password of the WiFi access point

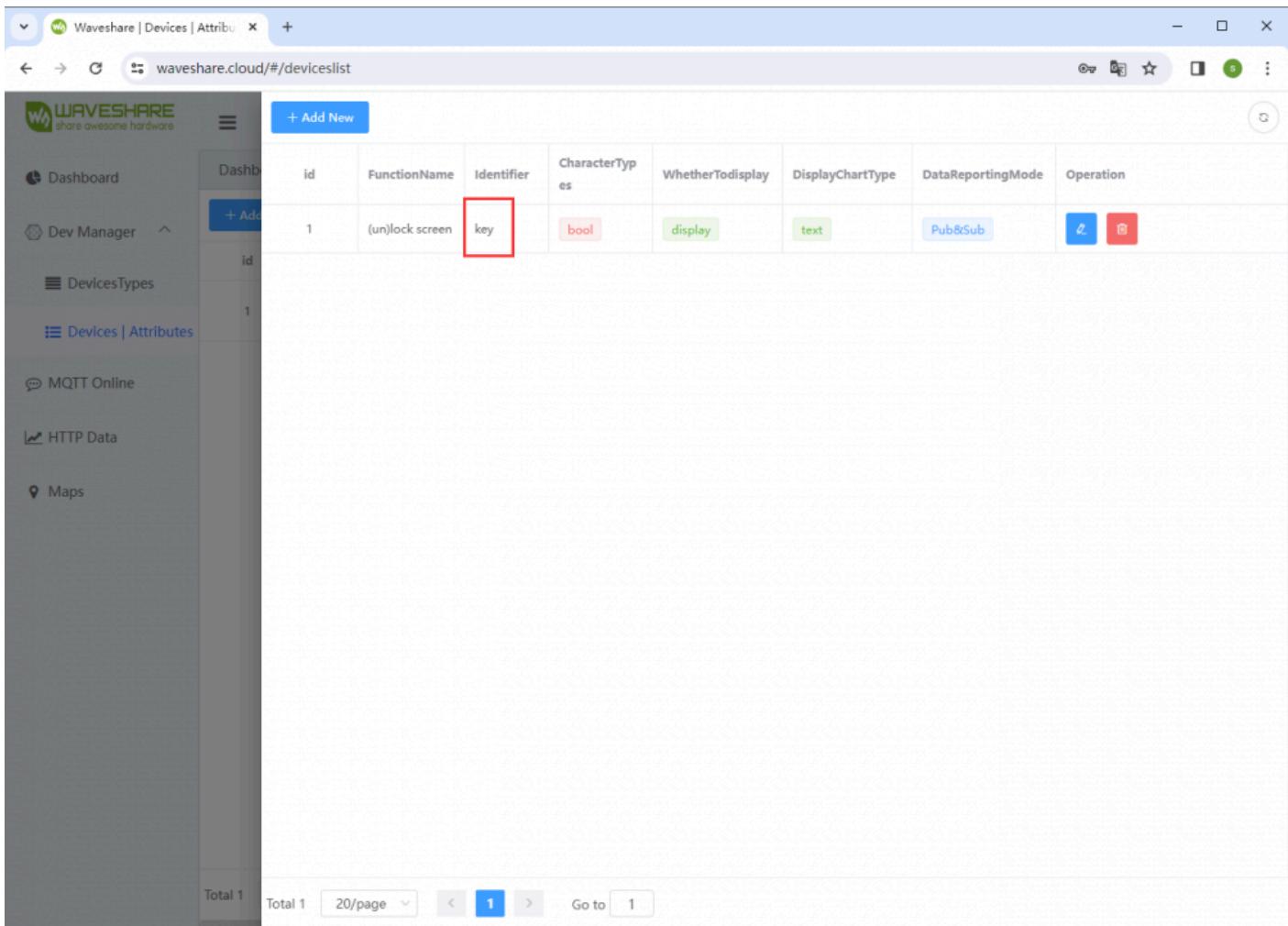

```
#define STASSID "ESP32-S3-GEEK" // Fill in your WIFI name
#define STAPSK "Waveshare" // Fill in your WIFI password
```
- Line 17: const char* ID = ""; // Defining device ID


```
const char* ID = ""; // Defining device ID
char pub[] = ""; // MQTT release topic
char sub[] = ""; // MQTT subscribe to topics
```

(/wiki/File:ESP32-S3-

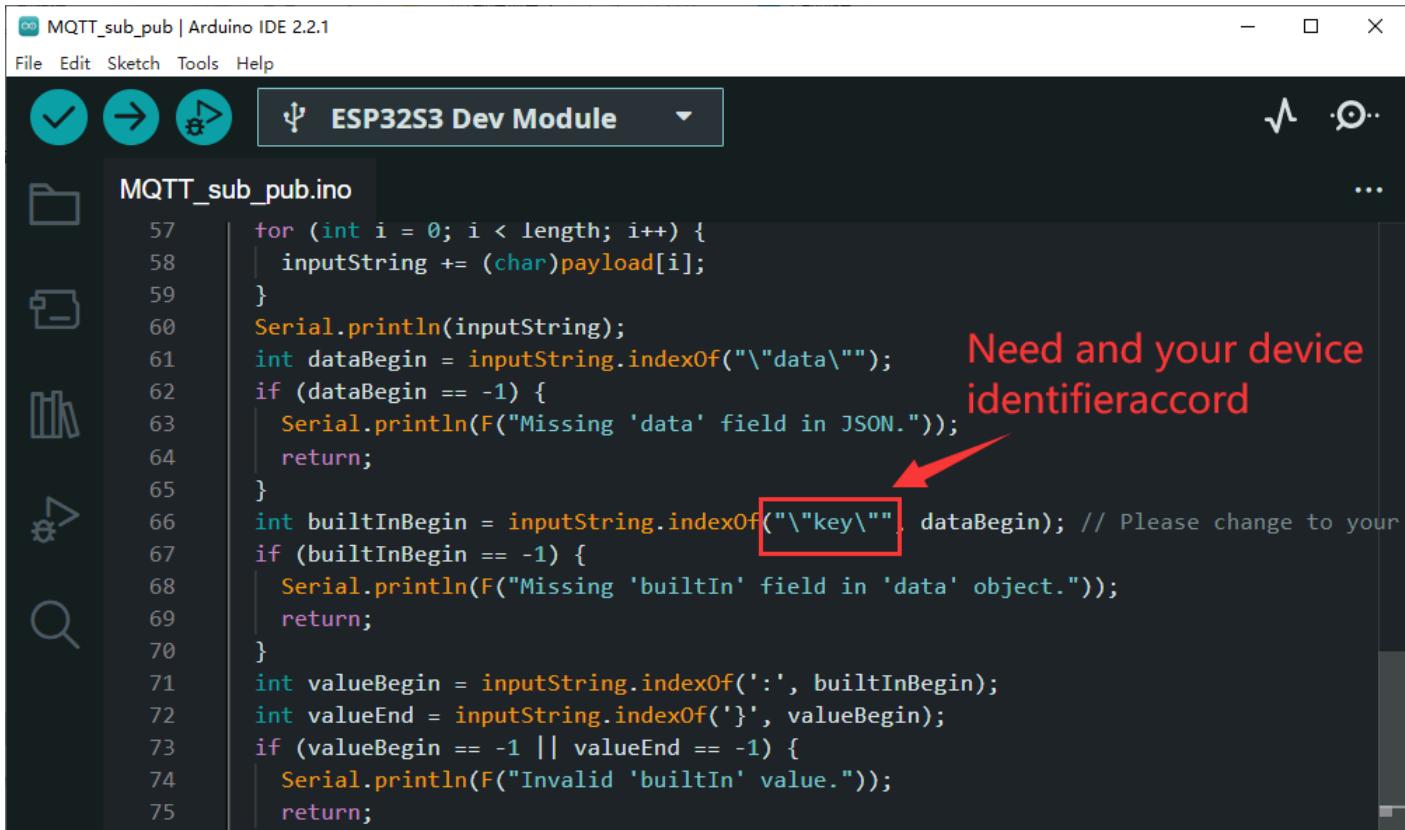
GEEK_SUB03.png)

- In the callback function, you can modify the identification tag to our device attribute identifier (https://mqtt.waveshare.cloud/en/wavesharecloud-wiki.html#3d7f7136_97) created on the cloud platform.



	ID	FunctionName	Identifier	CharacterTypes	WhetherTodisplay	DisplayChartType	DataReportingMode	Operation
	1	(un)lock screen	key	bool	display	text	Pub&Sub	 

(/wiki/File:ESP32-S3-GEEK_SUB04.png)



```

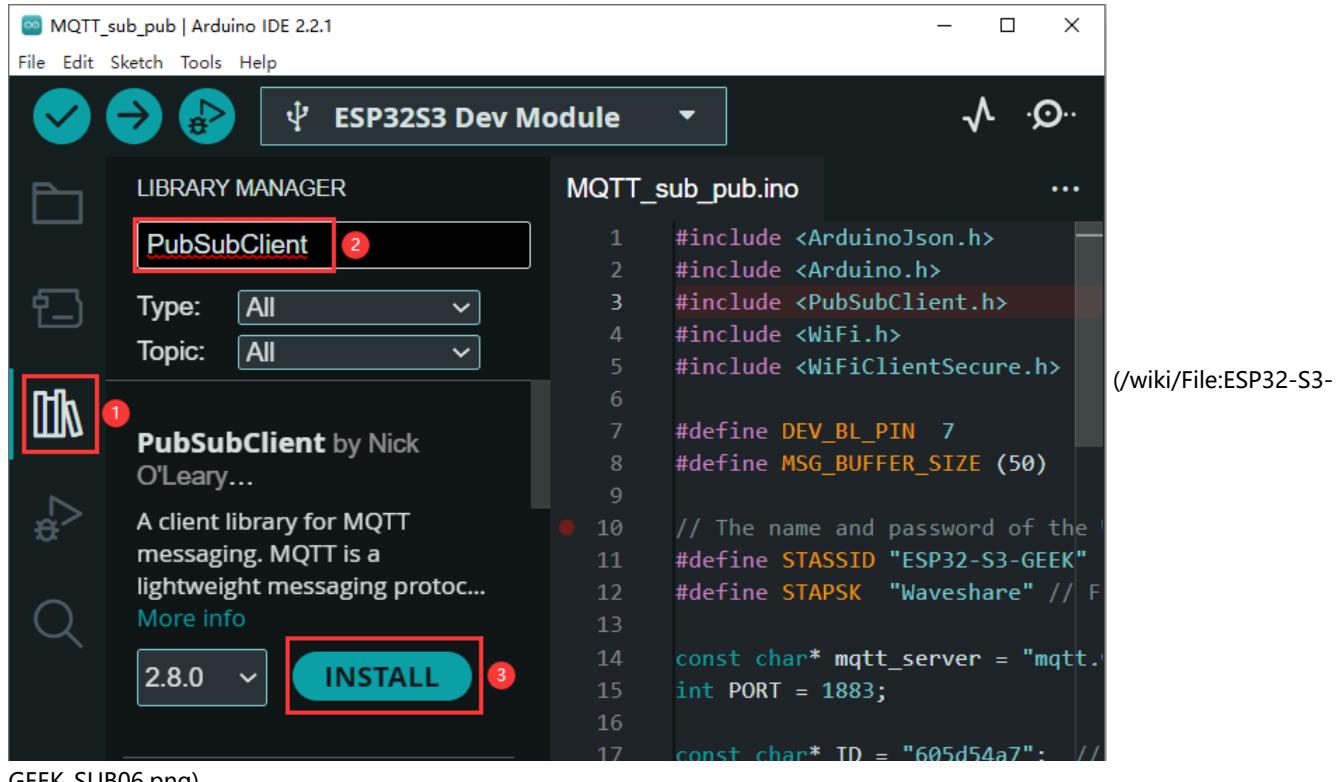
MQTT_sub_pub.ino
57   for (int i = 0; i < length; i++) {
58     inputString += (char)payload[i];
59   }
60   Serial.println(inputString);
61   int dataBegin = inputString.indexOf("data\"");
62   if (dataBegin == -1) {
63     Serial.println(F("Missing 'data' field in JSON."));
64     return;
65   }
66   int builtInBegin = inputString.indexOf("\"key\"", dataBegin); // Please change to your
67   if (builtInBegin == -1) {
68     Serial.println(F("Missing 'builtIn' field in 'data' object."));
69     return;
70   }
71   int valueBegin = inputString.indexOf(':', builtInBegin);
72   int valueEnd = inputString.indexOf('}', valueBegin);
73   if (valueBegin == -1 || valueEnd == -1) {
74     Serial.println(F("Invalid 'builtIn' value."));
75     return;

```

Need and your device identifier accord

(/wiki/File:ESP32-S3-GEEK_SUB05.png)

- Install PubSubClient library, LIBRARY MANAGER -> Search PubSubClient -> INSTALL.



GEEK_SUB06.png

- Program the code, and after connecting to the WIFI, you can observe whether the device on Waveshare cloud platform enters "online" status. If not, you can try to refresh the webpage, use CH343 USB UART Board (mini) (<https://www.waveshare.com/ch343-usb-uart-board.htm>) to connect to the PC, through the serial debugging assistant check the connection between the WIFI and MQTT. The LCD screen will display the connection situation between the WIFI and MQTT.

Waveshare | Devices | Attributes

Devices | Attributes

id	deviceType	Device name	description	Online status	Last up time	Operation
1	ESP32	ESP32-S3-GEEK Demo	ESP32-S3-GEEK Demo	online	2023-12-25 17:35:17	

Total 1 20/page < 1 > Go to 1

System Message
MQTT connection is successful

(/wiki/File:ESP32-S3-GEEK_SUB07.png)

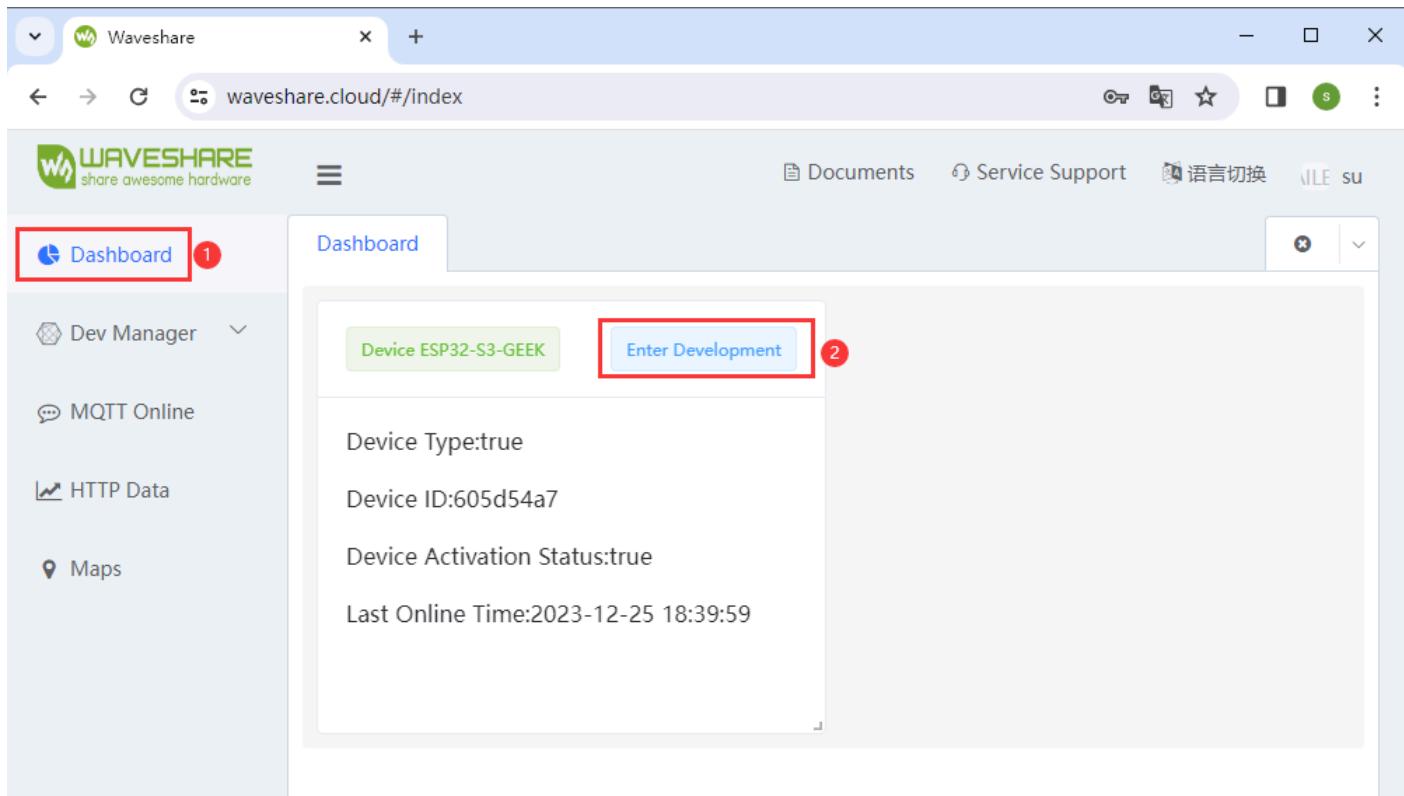


(/wiki/File:ESP32-S3-

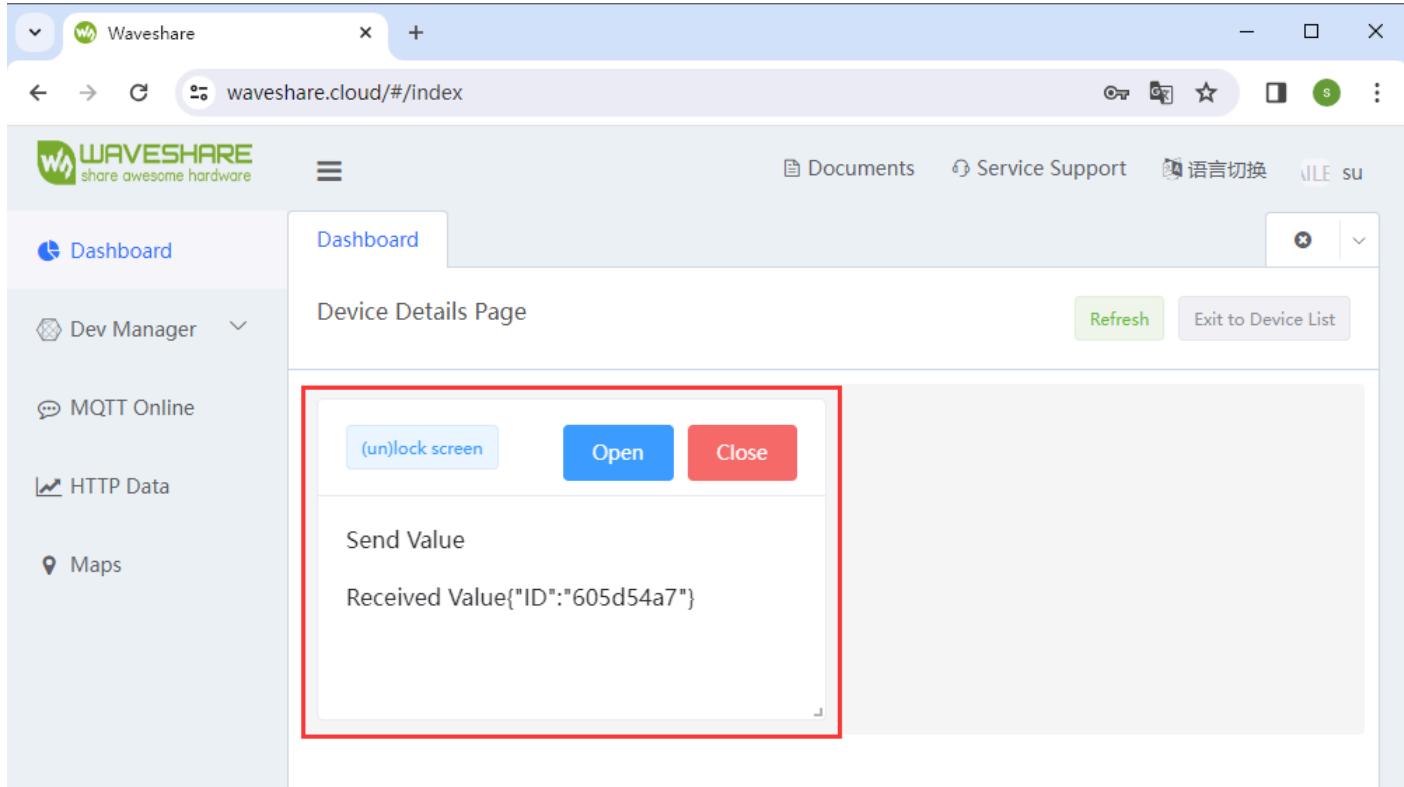


GEEK_SUB08.png)

- After connecting ESP32-S3-GEEK to Waveshare Cloud, you can directly send the MQTT message through the Dashboard.

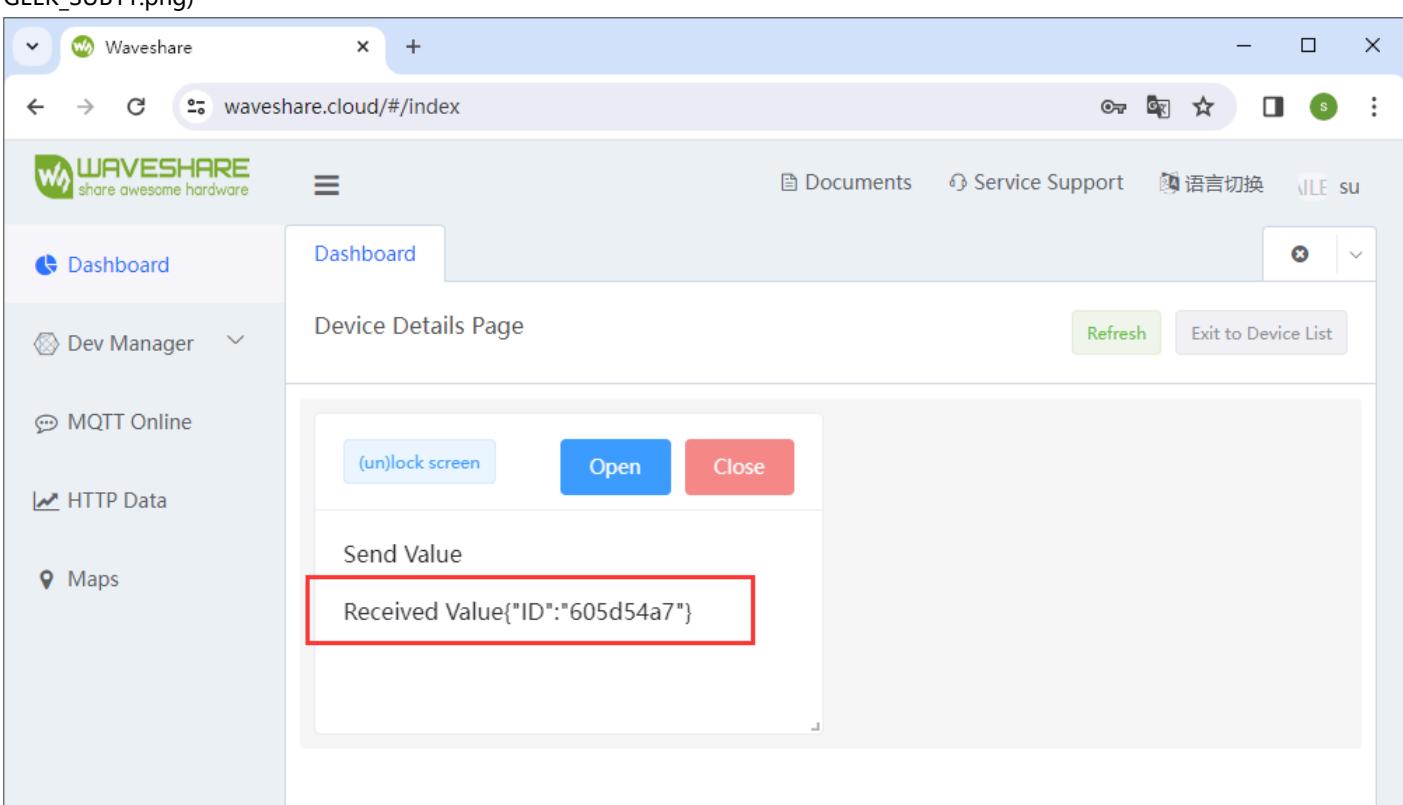
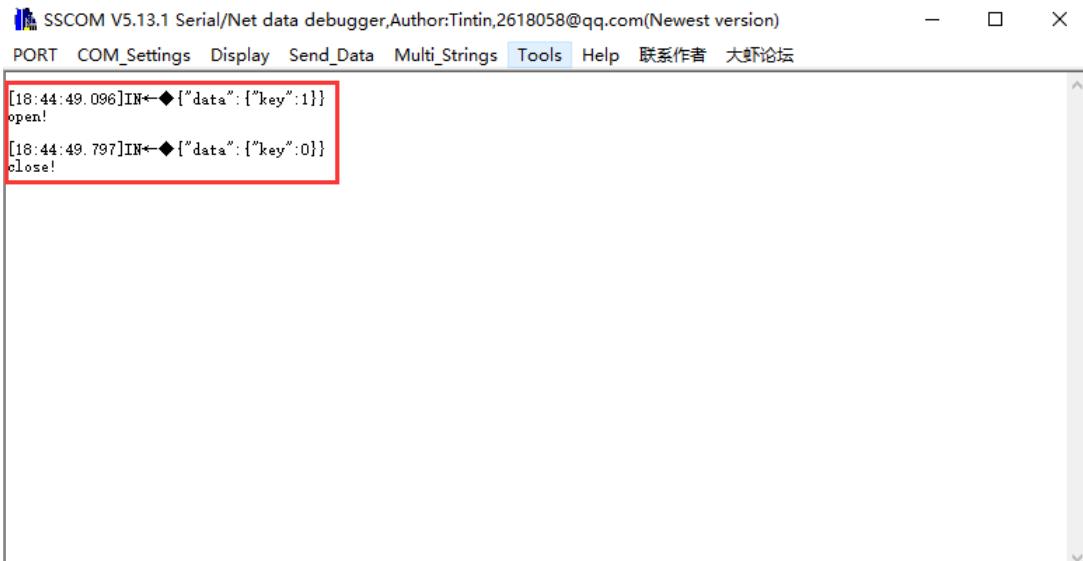


(/wiki/File:ESP32-S3-GEEK_SUB09.png)



(/wiki/File:ESP32-S3-GEEK_SUB10.png)

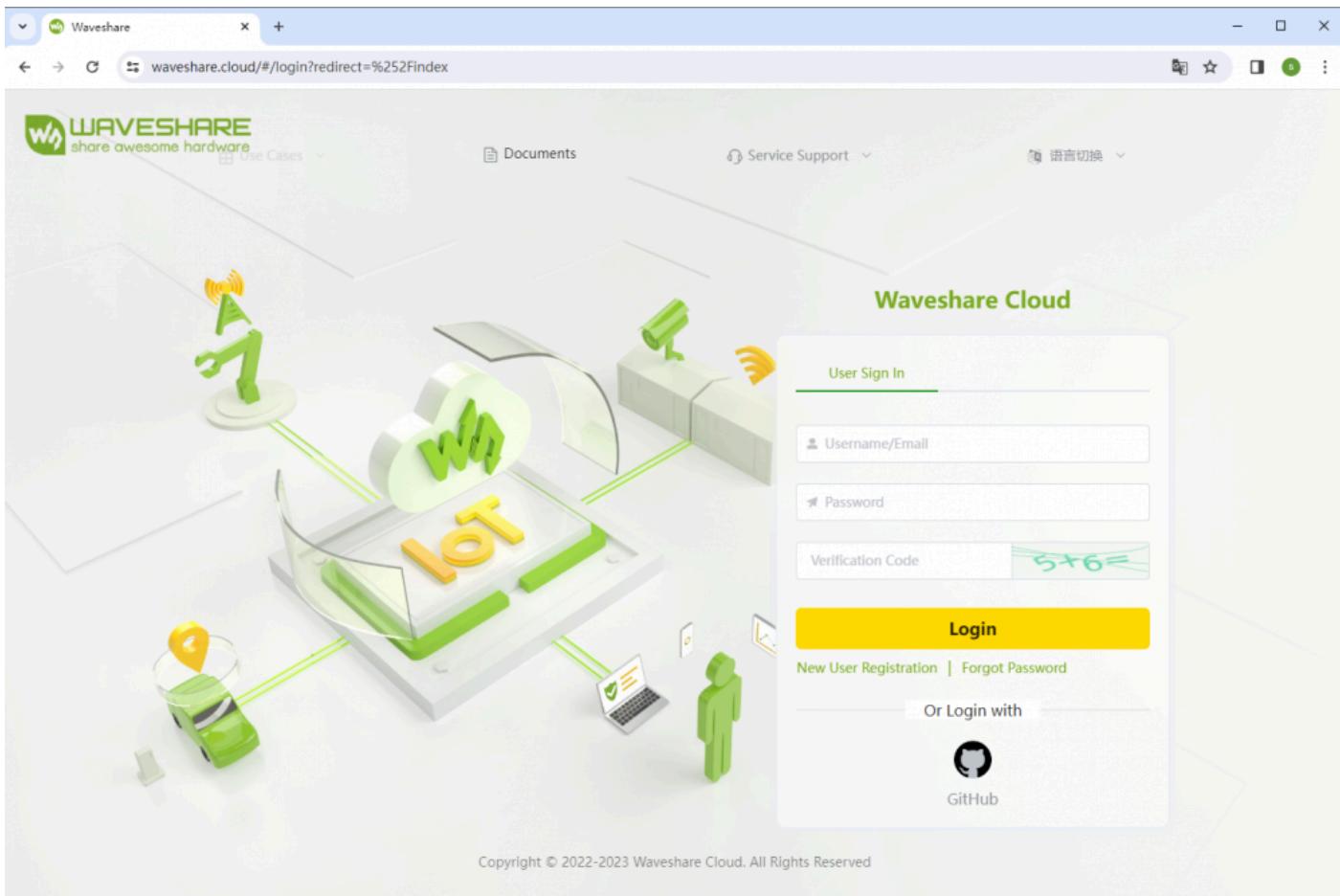
- We can observe different feedback regarding changes in device attribute values (e.g., "key") on both the LCD and the serial debugging assistant. Moreover, within Waveshare's cloud platform, under the device's received values, we can view the data transmitted from ESP32-S3-GEEK to the Waveshare cloud device (the received value corresponds to its Client ID). Subsequently, one can send the return value or status of the 'key' back to the Waveshare cloud. This completes the implementation of MQTT for both data uplink and downlink, as well as topic subscription and publication.



02-MQTT_BLE_Keyboard

This demo enables ESP32-S3-GEEK to enter the STA mode of the WIFI and Bluetooth. After connecting the WIFI and Bluetooth, on the Waveshare Cloud platform, it's possible to remotely lock the screen via Bluetooth and unlock it by entering a password. Additionally, there are more combinations of keys awaiting your development.

- When you are the first time to register on the Waveshare Cloud (<https://waveshare.cloud/>), you need to create a new device according to the tutorial (<https://mqtt.waveshare.cloud/en/wavesharecloud-wiki.html>).



(/wiki/File:ESP32-S3-GEEK_SUB.png)

(/wiki/File:ESP32-S3-GEEK_SUB01.png)

- After creating, you can set the corresponding configuration in the Arduino IDE according to ESP32SDK tutorial (<https://mqtt.waveshare.cloud/en/esp32-connect.html>).
- After the configuration is complete, according to the "View Address" of the newly created device on Waveshare Cloud, you can see the "Client ID, Pub Topic, and Sub Topic" of the device, and you can write them into the demo for assigning values, which can be used for ESP32-S3-GEEK to connect to your cloud platform device.

Waveshare | Devices | Attributes

Dashboard Devices | Attributes DevicesTypes

+ Add New +Rapid additions

id	devicesType	Device name	description	Online status	Last up time	Operation
1	ESP32	ESP32-S3-GEEK	ESP32-S3-GEEK Demo	notactivated		

Total 1 20/page Go to 1

(/wiki/File:ESP32-S3-GEEK_SUB02.png)

MQTT_BLE_Keyboard | Arduino IDE 2.2.1

File Edit Sketch Tools Help

Select Board

```

MQTT_BLE_Keyboard.ino

1 #include <BleKeyboard.h>
2 #include <WiFi.h>
3 #include <PubSubClient.h>
4
5 // WiFi credentials
6 #define SSID      "ESP32-S3-GEEK"      // Fill in your WIFI name
7 #define PASSWORD "Waveshare"          // Fill in your WIFI password
8
9 // MQTT configuration
10 const char* MQTT_SERVER = "mqtt.waveshare.cloud";
11 const int    MQTT_PORT   = 1883;
12
13 const char* Sub_Topic  = "";      // MQTT subscribe to topics
14 const char* Client_ID  = "";      // Defining device ID
15

```

(/wiki/File:ESP32-S3-GEEK_Key03.png)

- ESP32-S3-GEEK connects in STA mode to the same Wi-Fi network (The Wi-Fi network ESP32-S3-GEEK connects to needs to support the 2.4GHz frequency band. If the network doesn't support 2.4GHz, you can create a hotspot on your PC with the network frequency set to "Any available frequency.") Remember to modify the ssid and password with the name and password of the Wi-Fi network you want to connect to.
- In the callback function, you can customize the identification tag to match the device attribute identifier (https://mqtt.waveshare.cloud/en/wavesharecloud-wiki.html#3d7f7136_97) you've created on our cloud platform.

The screenshot shows the Waveshare Cloud Platform interface. On the left, there's a sidebar with options like Dashboard, Dev Manager, Devices | Attributes, MQTT Online, HTTP Data, and Maps. The main area is titled 'Devices | Attributes' and shows a table with one row. The table columns are: id, FunctionName, Identifier, CharacterTypes, WhetherTodisplay, DisplayChartType, DataReportingMode, and Operation. The single row has values: 1, (un)lock screen, key, bool, display, text, Pub&Sub, and two icons. A red box highlights the 'Identifier' column value 'key'.

(/wiki/File:ESP32-S3-GEEK_SUB04.png)

The screenshot shows the Arduino IDE with a sketch named 'MQTT_BLE_Keyboard.ino'. The code includes the following snippet:

```

for (int i = 0; i < length; i++) {
    Serial.print((char)payload[i]);
    payloadString += (char)payload[i];
}

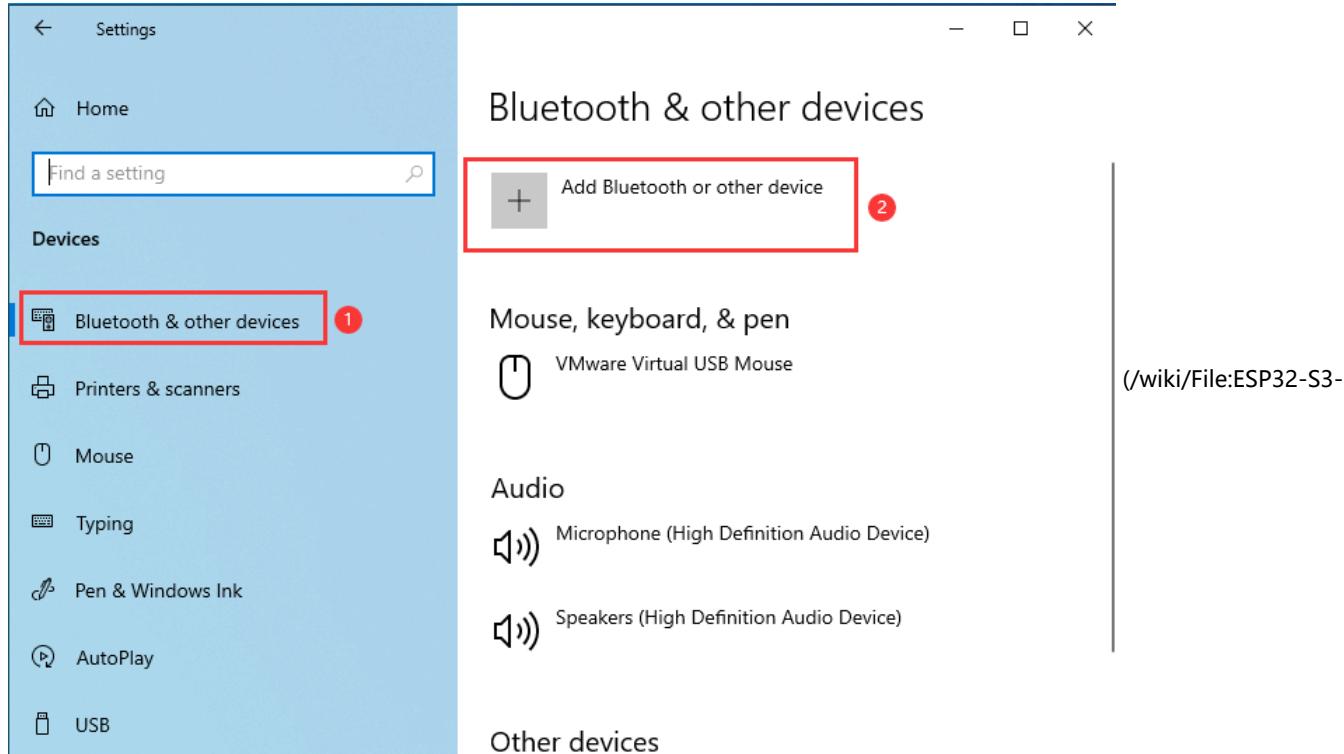
int keyPosition = payloadString.indexOf("\"key\"");
char keyChar1 = payloadString.charAt(keyPosition + (strlen("\"key\"") + 1));
// char keyChar2 = payloadString.charAt(keyPosition + (strlen("\"key\"") + 2)); // If the
if (keyChar1 == '1') Screen_ON();
else Screen_OFF();
}

```

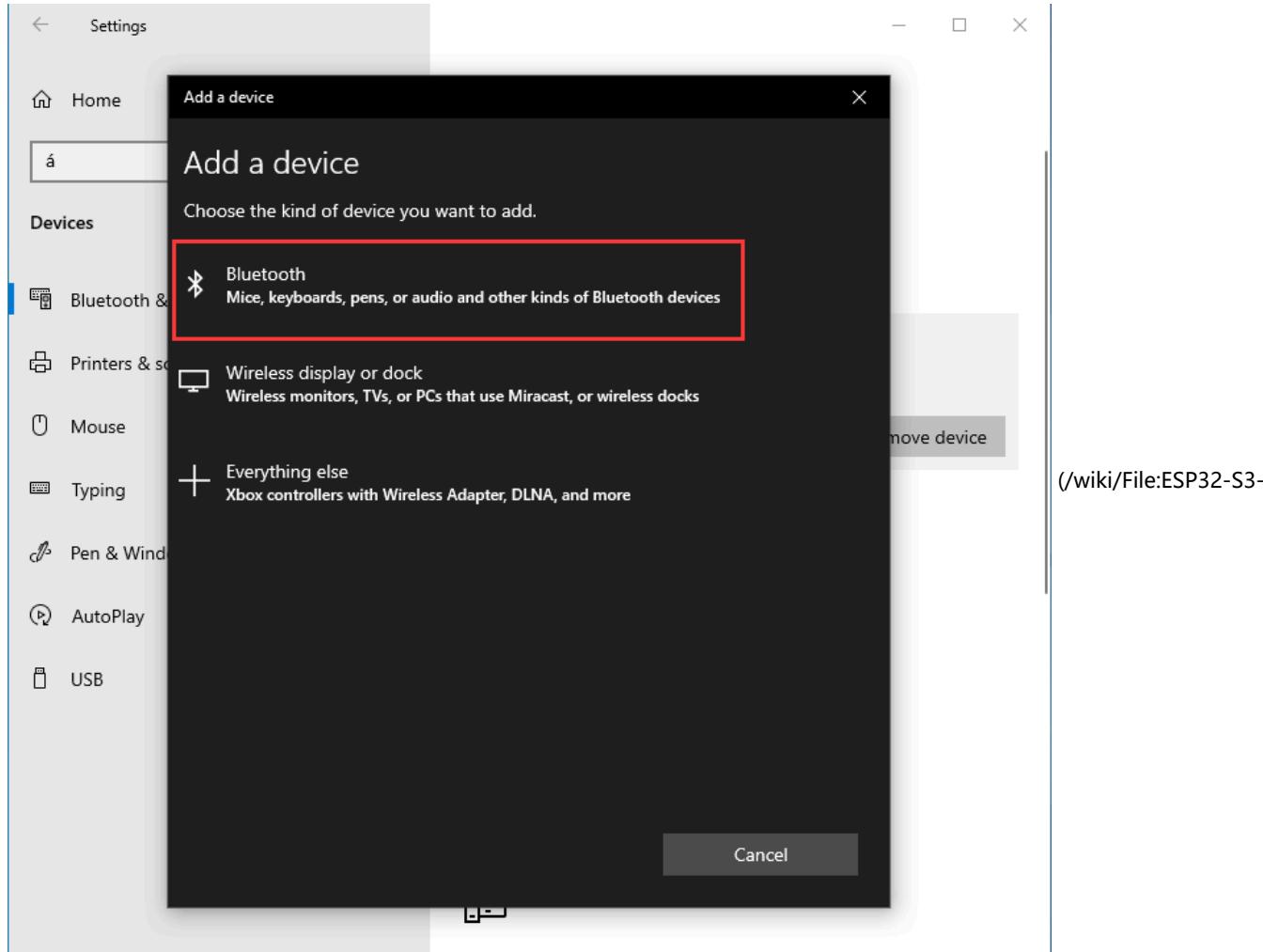
A red box highlights the string `"\key"` in the code. A red callout points from this box to the 'Identifier' field in the Waveshare Cloud Platform screenshot above, with the text 'The value must be the same as the device identifier'.

(/wiki/File:ESP32-S3-GEEK_Key05.png)

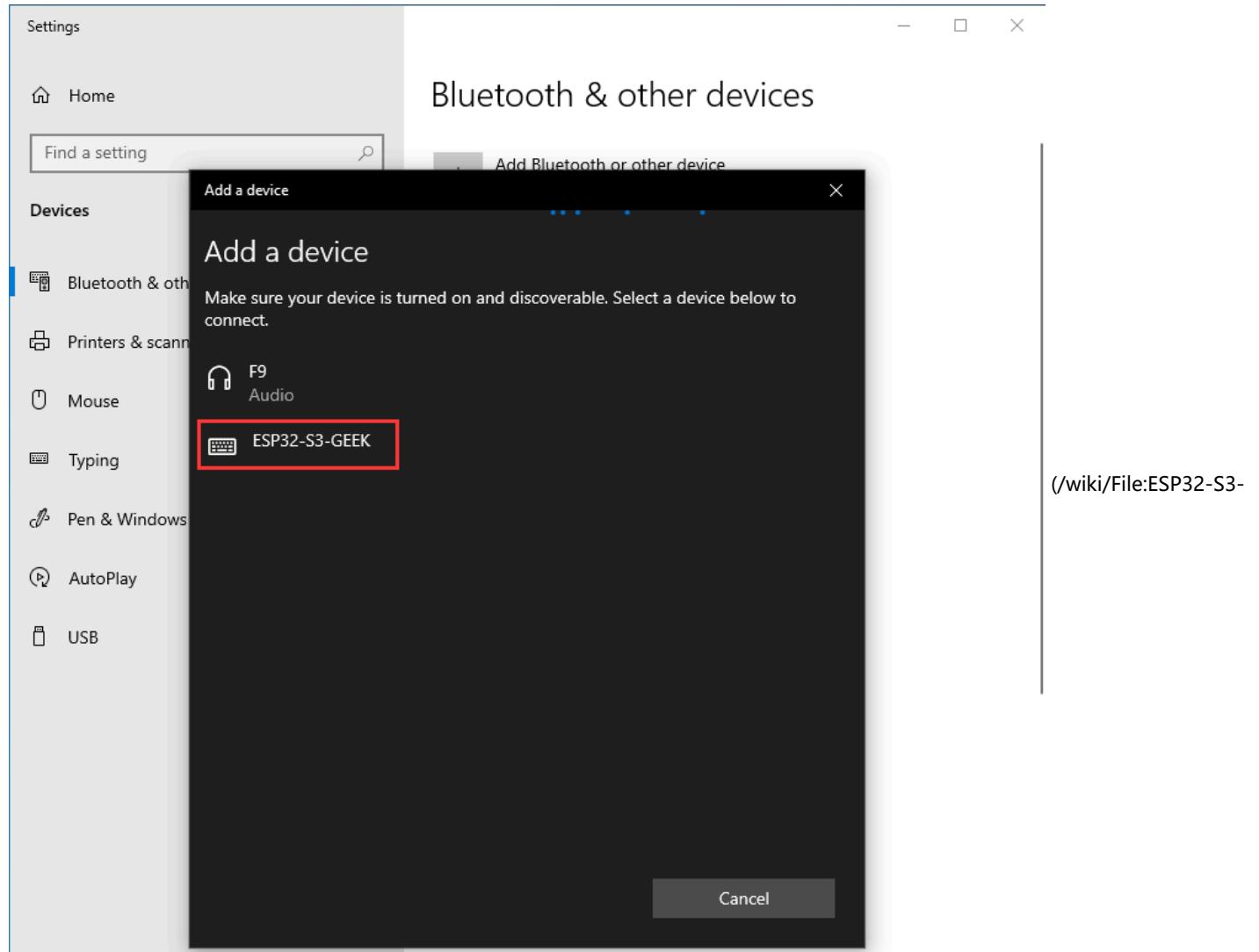
- Programming the code, after connecting to the WIFI, you can open PC Bluetooth to connect the ESP32-S3-GEEK.



GEEK_Demo46.png)



GEEK_Demo47.png)



(/wiki/File:ESP32-S3-

GEEK_Demo48.png)

- Observe whether the device is on the Waveshare cloud platform in the online state, if not, you can try to refresh the web page with the use of CH343_USB_UART_Board (<https://www.waveshare.com/ch343-usb-uart-board.htm>)to connect to the PC through the serial debugging assistant to view the WIFI and MQTT connection, and the LCD will also display the WIFI and MQTT connections.

The screenshot shows the Waveshare Cloud Dashboard interface. On the left, there's a sidebar with links like 'Dashboard', 'Dev Manager', 'DevicesTypes', 'Devices | Attributes', 'MQTT Online', 'HTTP Data', and 'Maps'. The main area is titled 'Devices | Attributes' and contains a table with columns: id, devicesType, Device name, description, Online status, Last up time, and Operation. There is one entry: id 1, devicesType ESP32, Device name ESP32-S3-GEEK, description ESP32-S3-GEEK Demo, Online status online (highlighted with a red box), Last up time 2023-12-25 17:35:17, and Operation with four icons. At the bottom right, a 'System Message' box says 'MQTT connection is successful'.

(/wiki/File:ESP32-S3-GEEK_SUB07.png)

The screenshot shows the SSCom V5.13.1 Serial/Net data debugger window. The menu bar includes PORT, COM_Settings, Display, Send_Data, Multi_Strings, Tools, Help, 联系作者, and 大虾论坛. The main window displays serial port logs:

```
[11:00:59.474]IN<-◆ESP-ROM: esp32s3-20210327
Build:Mar 27 2021
rst:Oxl (POWERON), boot:0x8 (SPI_FAST_FLASH_BOOT)
SPIWF:0xee
mode:DIO, clock div:1
load:0x3fce3808, len:0x44c
load:0x403c9700, len:0xbe4
load:0x403cc700, len:0x2a68
entry 0x403c98d4

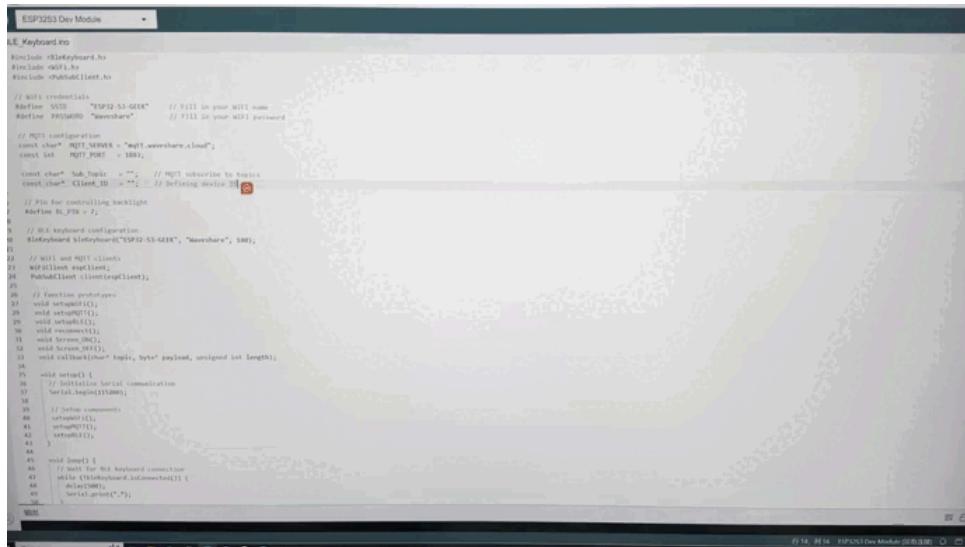
[11:01:05.028]IN<-◆Connecting to WiFi: ...
WiFi connected
IP address: 192.168.137.37
BLE keyboard initialized

[11:01:05.351]IN<-◆.
[11:01:06.763]IN<-◆Attempting MQTT connection...
[11:01:06.894]IN<-◆Connected to MQTT
```

The bottom of the window shows configuration options for the serial port, including 'COMNum: COM58 USB-Enhanced-SERIAL !', 'BaudRate: 115200', and various checkboxes for flow control (RTS, DTR) and settings (HEXShow, SaveData, ReceivedToFile, SendHEX, SendEvery, Show Time and Packe, OverTime, BytesTo, Verify). A status bar at the bottom shows 'www.daxia.com S:0 R:374 COM58 Opened 115200bps,8,1,None,None'.

GEEK_G02.png)

- After the ESP32-S3-GEEK connects to the Waveshare Cloud, you can log in to the Waveshare Cloud Dashboard on your phone remotely lock the PC screen, and input a password for screen unlocking.



```

// E_Keyboard.h
#include <Dynamixel.h>
#include <WiFi.h>
#include <PubSubClient.h>

// WiFi credentials
#define SSID "ESP32-S3-GEEK" // Fill in your WiFi name
#define PASSWORD "waveshare" // Fill in your WiFi password

// MQTT configuration
const char* MQTT_SERVER = " mqtt.waveshare.cloud";
const int MQTT_PORT = 1883;

const char* Sub_Topic = "#"; // MQTT subscribe to topics
const char* Client_ID = "ESP32-S3-GEEK"; // Defining device ID

// Pin for controlling backlight
#define BL_PIN 72

// WiFi keyboard configuration
Keyboard keyboard("ESP32-S3-GEEK", "waveshare", 100);

// WiFi and MQTT clients
WiFiClient WiFiClient;
PubSubClient mqttClient(WiFiClient);
Keyboard keyboard;

// Function prototypes
void setup();
void setupKeyboard();
void setupMQTT();
void setupSerial();
void setupKeyboard();
void setupMQTT();
void setupSerial();
void setupKeyboard();
void setupMQTT();
void setupSerial();

void loop();
// Initialization serial communication
Serial.begin(115200);
// Software components
void setup();
void setupKeyboard();
void setupMQTT();
void setupSerial();
void setupKeyboard();
void setupMQTT();
void setupSerial();

void loop();
// WiFi for Old Keyboard connection
void setupKeyboard();
void setupMQTT();
void setupSerial();
void setupKeyboard();
void setupMQTT();
void setupSerial();
void setupKeyboard();
void setupMQTT();
void setupSerial();

void loop();
// WiFi connected
void setupKeyboard();
void setupMQTT();
void setupSerial();
void setupKeyboard();
void setupMQTT();
void setupSerial();

void loop();
// Attempting MQTT connection...
void setupKeyboard();
void setupMQTT();
void setupSerial();
void setupKeyboard();
void setupMQTT();
void setupSerial();

void loop();
// Connected to MQTT
void setupKeyboard();
void setupMQTT();
void setupSerial();
void setupKeyboard();
void setupMQTT();
void setupSerial();

void loop();
// Message arrived [Sub/68/605d54a7]
void setupKeyboard();
void setupMQTT();
void setupSerial();
void setupKeyboard();
void setupMQTT();
void setupSerial();

void loop();
// Screen OFF
void setupKeyboard();
void setupMQTT();
void setupSerial();
void setupKeyboard();
void setupMQTT();
void setupSerial();

void loop();
// Enter
void setupKeyboard();
void setupMQTT();
void setupSerial();
void setupKeyboard();
void setupMQTT();
void setupSerial();

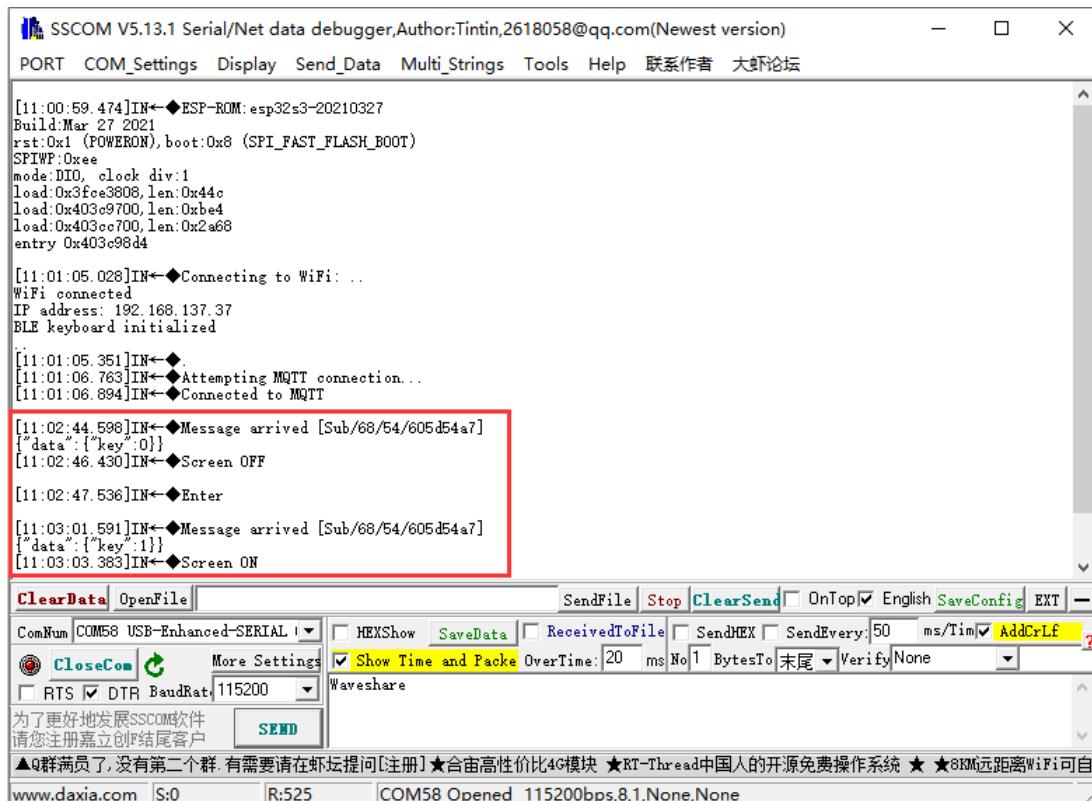
void loop();
// Message arrived [Sub/68/605d54a7]
void setupKeyboard();
void setupMQTT();
void setupSerial();
void setupKeyboard();
void setupMQTT();
void setupSerial();

void loop();
// Screen ON
void setupKeyboard();
void setupMQTT();
void setupSerial();
void setupKeyboard();
void setupMQTT();
void setupSerial();

```

(/wiki/File:ESP32-S3-GEEK_G03.gif)

- We can observe different feedback regarding changes in device attribute values (e.g., "key") on both the LCD and the serial debugging assistant. Additionally, we can further customize the callback function to modify keys into combinations like Ctrl+C, and Ctrl+V, allowing for a personalized DIY remote-controlled Bluetooth keyboard.



[11:00:59.474]IN<-◆ESP-ROM: esp32s3-20210327
Build:Mar 27 2021
rst:0x1 (POWERON), boot:0x8 (SPI_FAST_FLASH_BOOT)
SPIWP:0xee
mode:DIO, clock div:1
load:0x3fce3808, len:0x44c
load:0x403c9700, len:0xbe4
load:0x403cc700, len:0x2a68
entry 0x403c98d4

[11:01:05.028]IN<-◆Connecting to WiFi: ...
WiFi connected
IP address: 192.168.137.37
BLE keyboard initialized

[11:01:05.351]IN<-◆
[11:01:06.763]IN<-◆Attempting MQTT connection...
[11:01:06.894]IN<-◆Connected to MQTT

[11:02:44.598]IN<-◆Message arrived [Sub/68/605d54a7]
{"data": ["key": 0]}
[11:02:46.430]IN<-◆Screen OFF

[11:02:47.536]IN<-◆Enter

[11:03:01.591]IN<-◆Message arrived [Sub/68/605d54a7]
{"data": ["key": 1]}
[11:03:03.383]IN<-◆Screen ON

ClearData OpenFile SendFile Stop **ClearSend** OnTop English SaveConfig EXT

ComNum: COM58 USB-Enhanced-SERIAL ! HEXShow SaveData ReceivedToFile SendHEX SendEvery: 50 ms/Tim AddCrLf ?
CloseCon More Settings Show Time and Packe OverTime: 20 ms No 1 BytesTo 末尾 Verify None
RTS DTR BaudRate: 115200 Waveshare
为了更好地发展SSCom软件
请您注册嘉立创客户

▲Q群满员了,没有第二个群,有需要请在虾坛提问【注册】★合宙高性价比4G模块 ★RT-Thread中国人的开源免费操作系统 ★★★3KM远距离WiFi可自

www.daxia.com S:0 R:525 COM58 Opened 115200bps,8,1,None,None

(/wiki/File:ESP32-S3-

GEEK_G04.png)

- Or you can enter keyboard tester (<https://www.zfrontier.com/lab/keyboardTester>) website to test the keys remotely pressed by the ESP32-S3-GEEK Bluetooth.

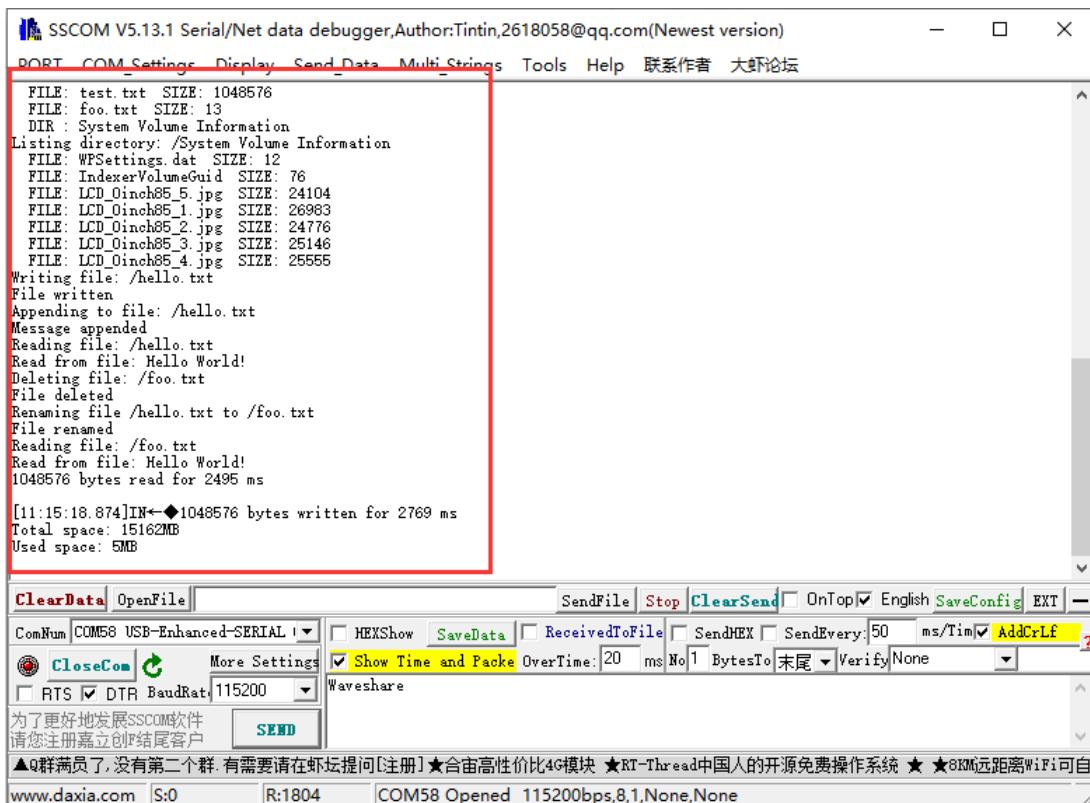


(/wiki/File:ESP32-S3-GEEK_G05.jpg)

SD

01-SD_Test

This demo allows the use of ESP32-S3-GEEK's SD card slot. Insert an SD card into the slot and open the serial debugging assistant. You'll be able to observe ESP32-S3-GEEK performing operations such as creating, deleting, modifying, and checking files on the SD card.



(/wiki/File:ESP32-S3-

GEEK_SD.png)

02-SD_LCD

With this demo, the pictures of the SD card in the SD card slot of the ESP32-S3-GEEK can be read. Store the pictures on the SD card, and insert it into the card slot. The ESP32-S3-GEEK can read and display the pictures of the SD card on the LCD, or it can flash the pictures as a gif.

- Copy the TFT_eSPI file folder of the libraries to the installation directory of Arduino library.



The screenshot shows the Arduino IDE interface. The title bar reads "BLE_Keyboard | Arduino IDE 2.2.1". The "File" menu is open, with the "Preferences..." option highlighted by a red box and a red number "2" indicating it's the second step in a two-step process. The main code editor area displays a sketch titled "ESP32S3 Dev Module" which turns the ESP32 into a Bluetooth LE keyboard. The code includes functions like `bleKeyboard.begin()` and `Serial.println("Starting BLE work!");`.

```

ESP32S3 Dev Module

example turns the ESP32 into a Bluetooth LE keyboard that writes to a host computer.

<BleKeyboard.h>

ard bleKeyboard("ESP32-S3-GEEK", "Waveshare", 100);

up() {
.begin(115200);
.println("Starting BLE work!");

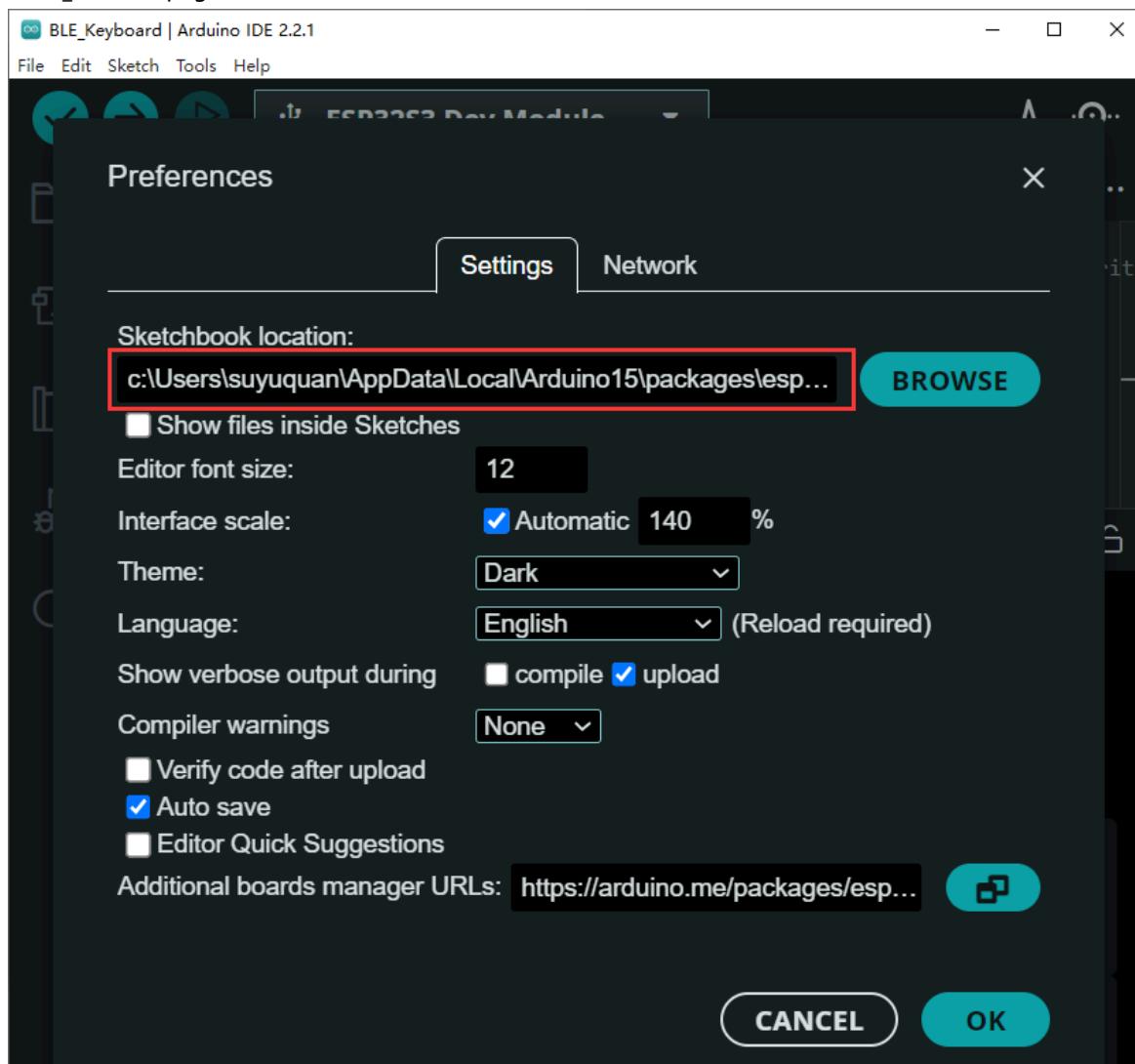
bleKeyboard.begin();

}

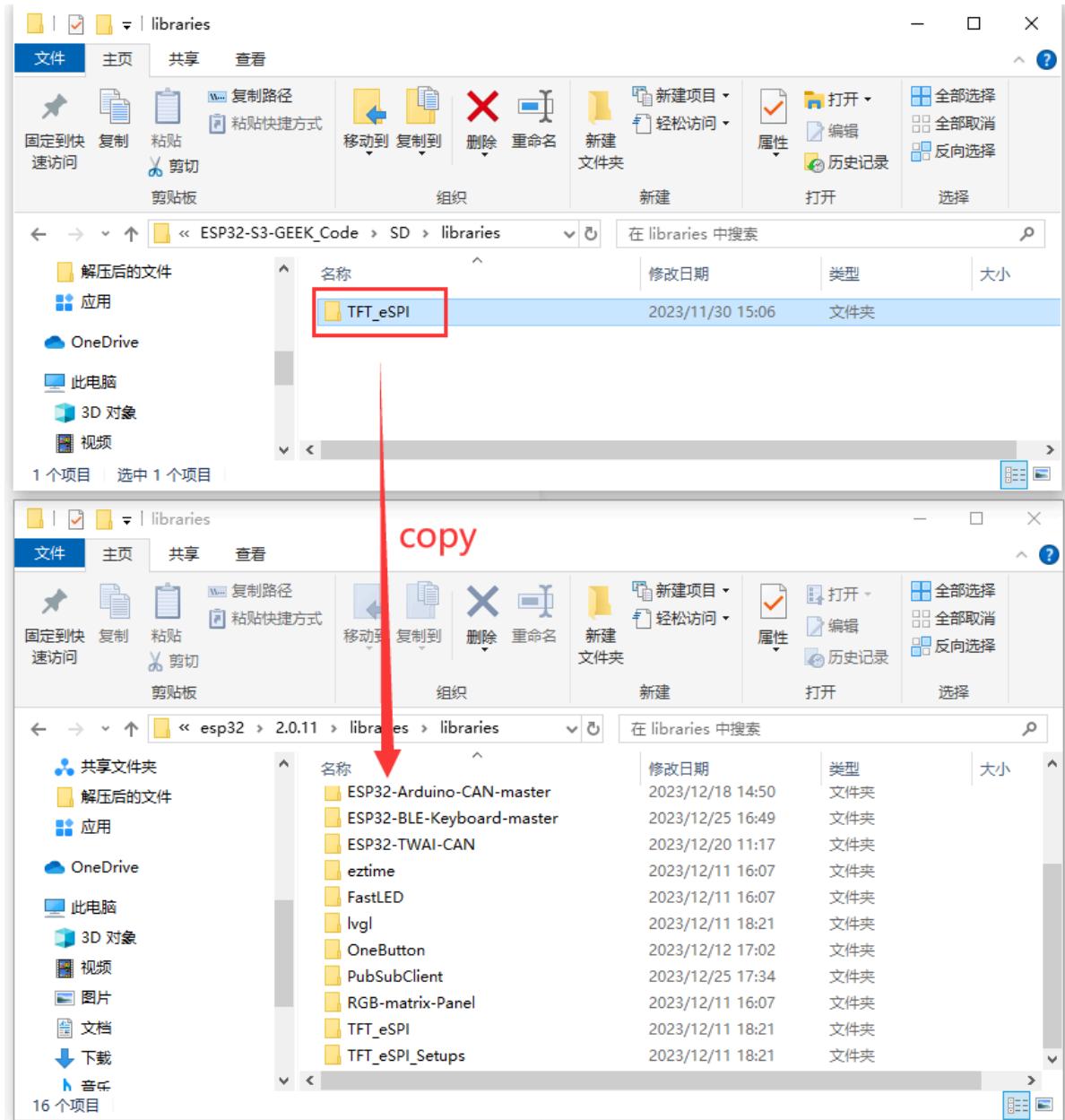
void loop() {
if(bleKeyboard.isConnected()) {
Serial.println("Sending 'Waveshare' ...");
}
}

```

GEEK_Demo40.png

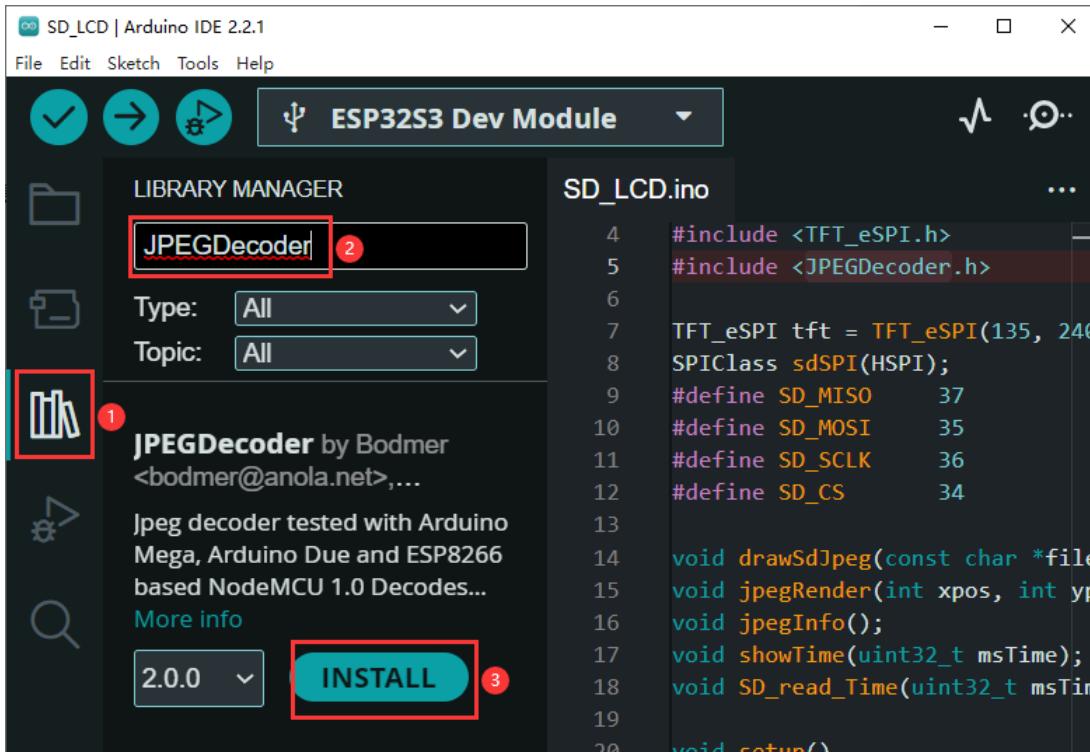


GEEK_Demo41.png)



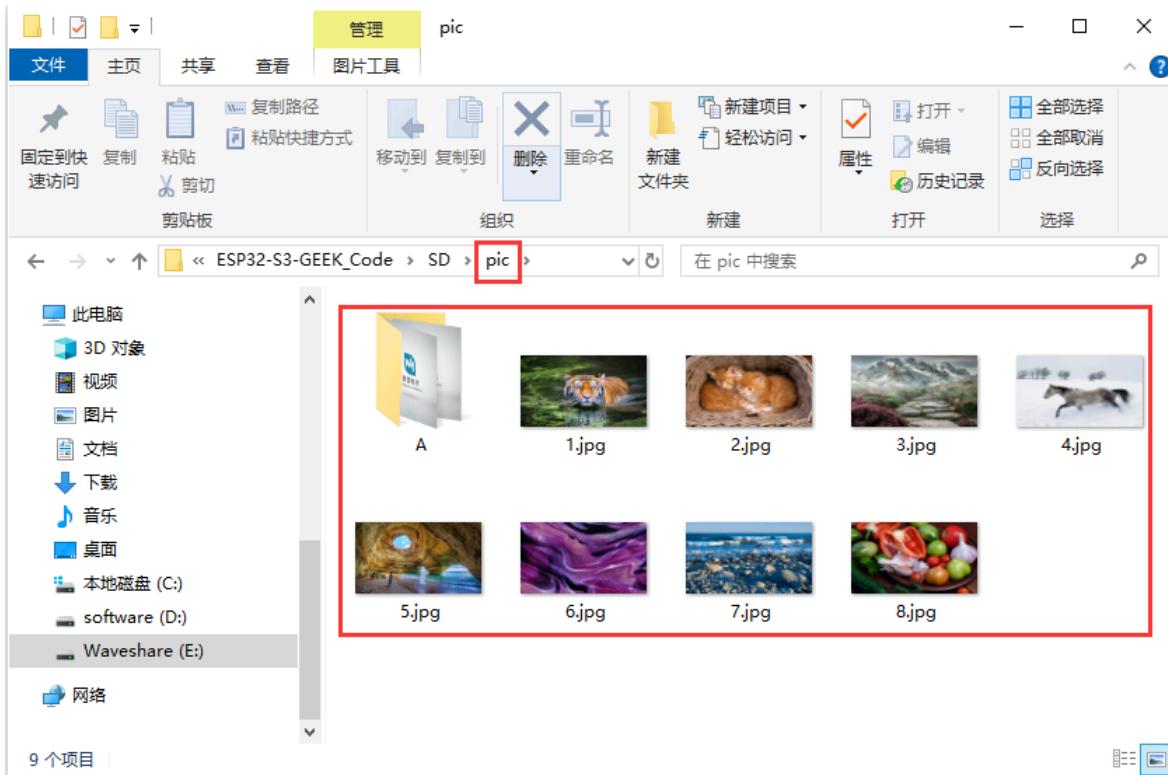
S3-GEEK_LCD03.png)

- Install JPEGDecoder library, LIBRARY MANAGER -> search for JPEGDecoder -> INSTALL.



GEEK_LCD04.png

- You can store the pictures of "pic" on the SD card, or you can store your pictures, and modify the picture size to 240×135, and the display effect is the best.



GEEK_LCD05.png

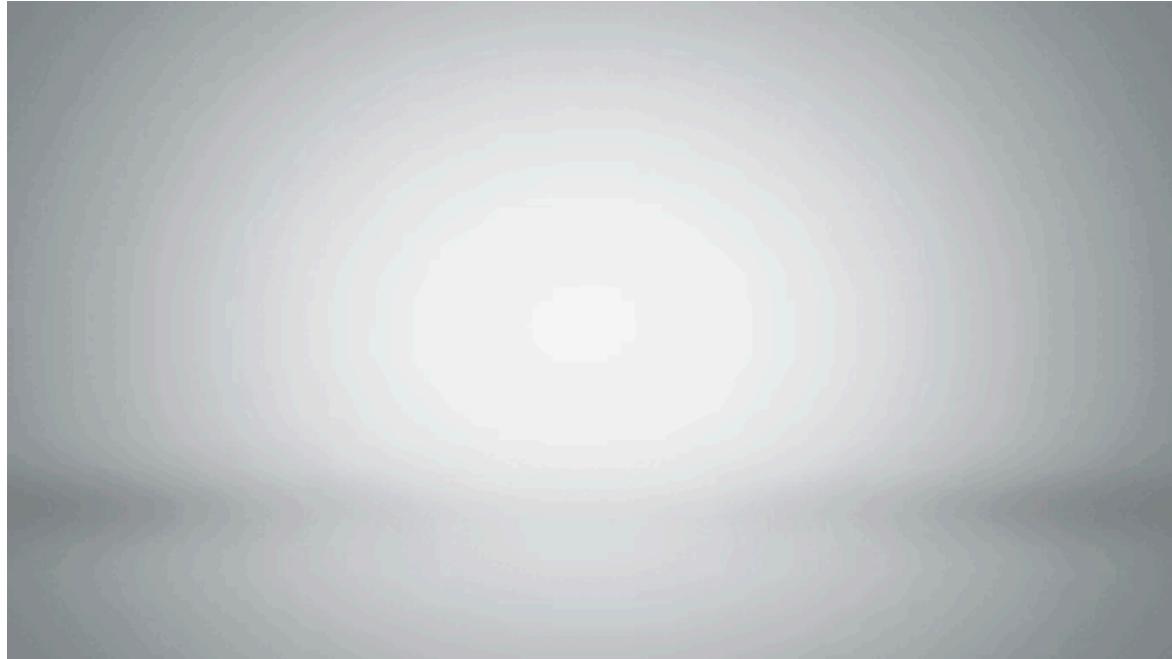
- This part of the comment is removed to play the Waveshare startup animation.

```

SD_LCD | Arduino IDE 2.2.1
File Edit Sketch Tools Help
ESP32S3 Dev Module
SD_LCD.ino
79 void loop() {
80
81     // Test the wallpaper
82     for(int image_num = 1;image_num<=8;image_num++){
83         char FileName[10];
84         sprintf(FileName,"%d.jpg",image_num);
85         drawSdJpeg(FileName, 0, 0);      // This draws a jpeg pulled off the
86         delay(1000);
87     }
88
89     // //Play folder A <--Gif
90     // for(int image_num = 1;image_num<=(105-3);image_num+=1){
91     //     char FileName[10];
92     //     sprintf(FileName,"/A/%d.jpg",image_num);
93     //     drawSdJpeg(FileName, 0, 0);      // This draws a jpeg pulled off the
94     // }
95     // delay(2000);
96
97 }
98

```

GEEK_LCD06.png



(/wiki/File:ESP32-S3-

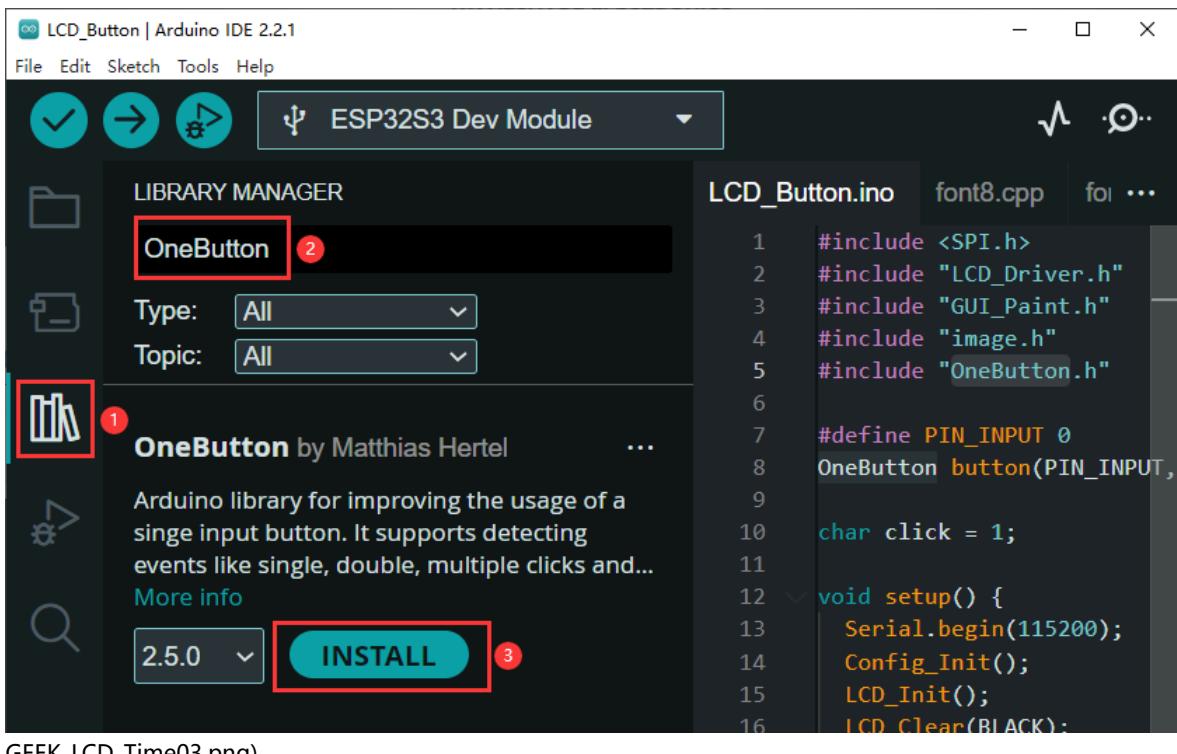
GEEK_LCD07.gif)

LCD

01-LCD_Button

This demo can use the boot button of ESP32-S3-GEEK to realize a short press to light up the LCD and switch to the next picture, and a long press to turn off the LCD.

- Install OneButton library, LIBRARY MANAGER -> search for OneButton -> INSTALL.



The screenshot shows the Arduino IDE Library Manager. In the center, the 'OneButton' library by Matthias Hertel is selected, indicated by a red box and a red number '2'. Below it, the version '2.5.0' is shown with a dropdown arrow, and a large red box surrounds the 'INSTALL' button, with a red number '3' indicating it's the current step. On the right, the code for 'LCD_Button.ino' is visible, starting with #include <SPI.h>. The status bar at the bottom right shows the URL: (/wiki/File:ESP32-S3-).

```

1 #include <SPI.h>
2 #include "LCD_Driver.h"
3 #include "GUI_Paint.h"
4 #include "image.h"
5 #include "OneButton.h"
6
7 #define PIN_INPUT 0
8 OneButton button(PIN_INPUT,
9
10 char click = 1;
11
12 void setup() {
13   Serial.begin(115200);
14   Config_Init();
15   LCD_Init();
16   LCD_Clear(BLACK);

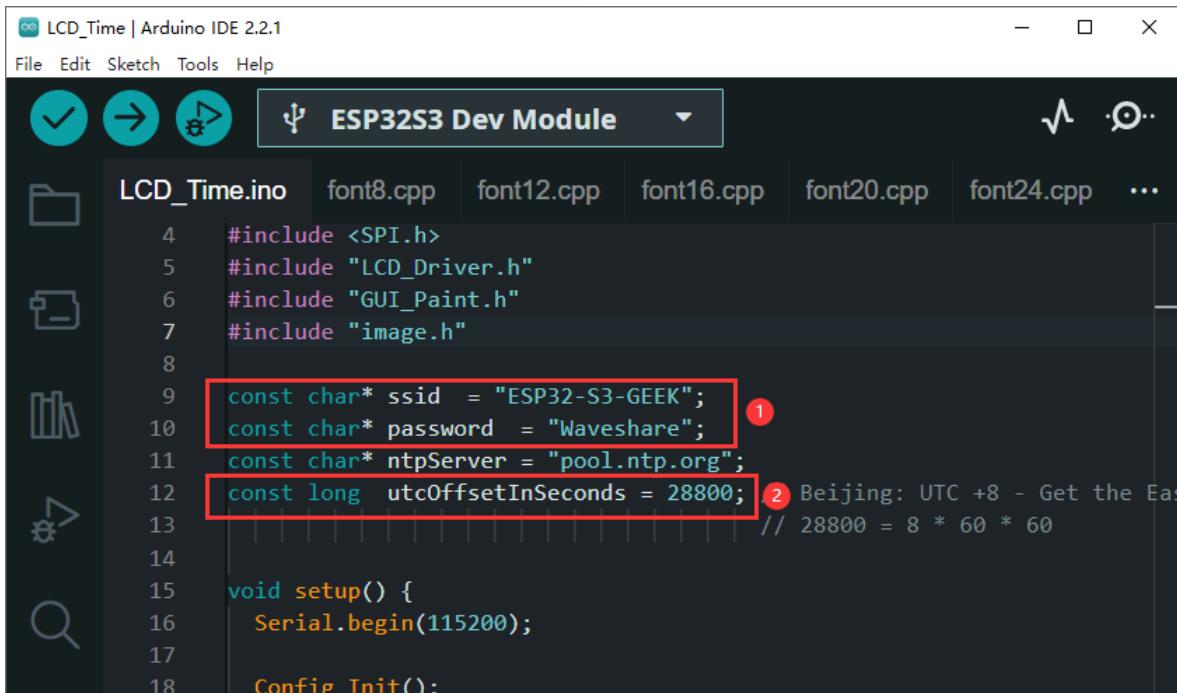
```

GEEK_LCD_Time03.png

02-LCD_Time

This demo allows ESP32-S3-GEEK to connect to WiFi and obtain the current time. It then displays the date and time on both the LCD and the serial debugging assistant.

- Modify the ssid and password to the name and password of the Wi-Fi network you want. (The WIFI of the ESP32-S3-GEEK connected requires the Wi-Fi network of the 2.4GHz frequency band. If there's no 2.4GHz band available, create a hotspot on your PC with the network frequency set to "Any available frequency".) For instance, if you need the time zone for Beijing (GMT+8), set utcOffsetInSeconds to 28800.(8*60*60)



The screenshot shows the Arduino IDE with the 'LCD_Time.ino' sketch open. The code includes configuration constants for WiFi and time synchronization. A red box highlights the WiFi credentials: 'ssid = "ESP32-S3-GEEK"' and 'password = "Waveshare"'. Another red box highlights the UTC offset: 'const long utcOffsetInSeconds = 28800;'. A red number '1' is placed near the WiFi constants, and a red number '2' is placed near the offset constant. The status bar at the bottom right shows the URL: (/wiki/File:ESP32-S3-).

```

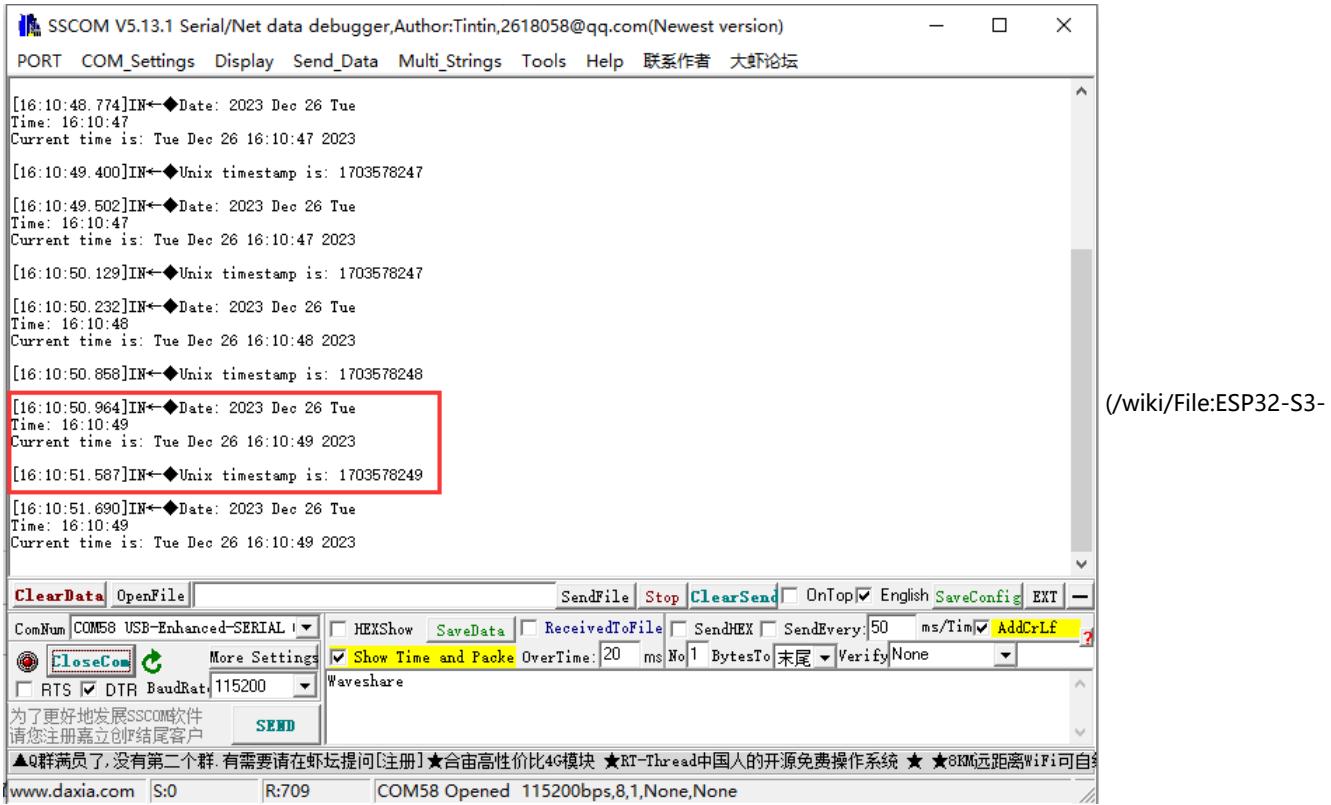
4 #include <SPI.h>
5 #include "LCD_Driver.h"
6 #include "GUI_Paint.h"
7 #include "image.h"
8
9 const char* ssid = "ESP32-S3-GEEK"; 1
10 const char* password = "Waveshare";
11 const char* ntpServer = "pool.ntp.org";
12 const long utcOffsetInSeconds = 28800; 2 Beijing: UTC +8 - Get the East
13 // 28800 = 8 * 60 * 60
14
15 void setup() {
16   Serial.begin(115200);
17
18   Config_Init();

```

GEEK_LCD_Time01.png

- After programming the code, you can see the LCD display the real-time time and date. ESP32-S3-GEEK uses USB to UART to connect to the PC, open the serial port debugging assistant, and you can see the obtained time and date printed on the serial

port debugging assistant.



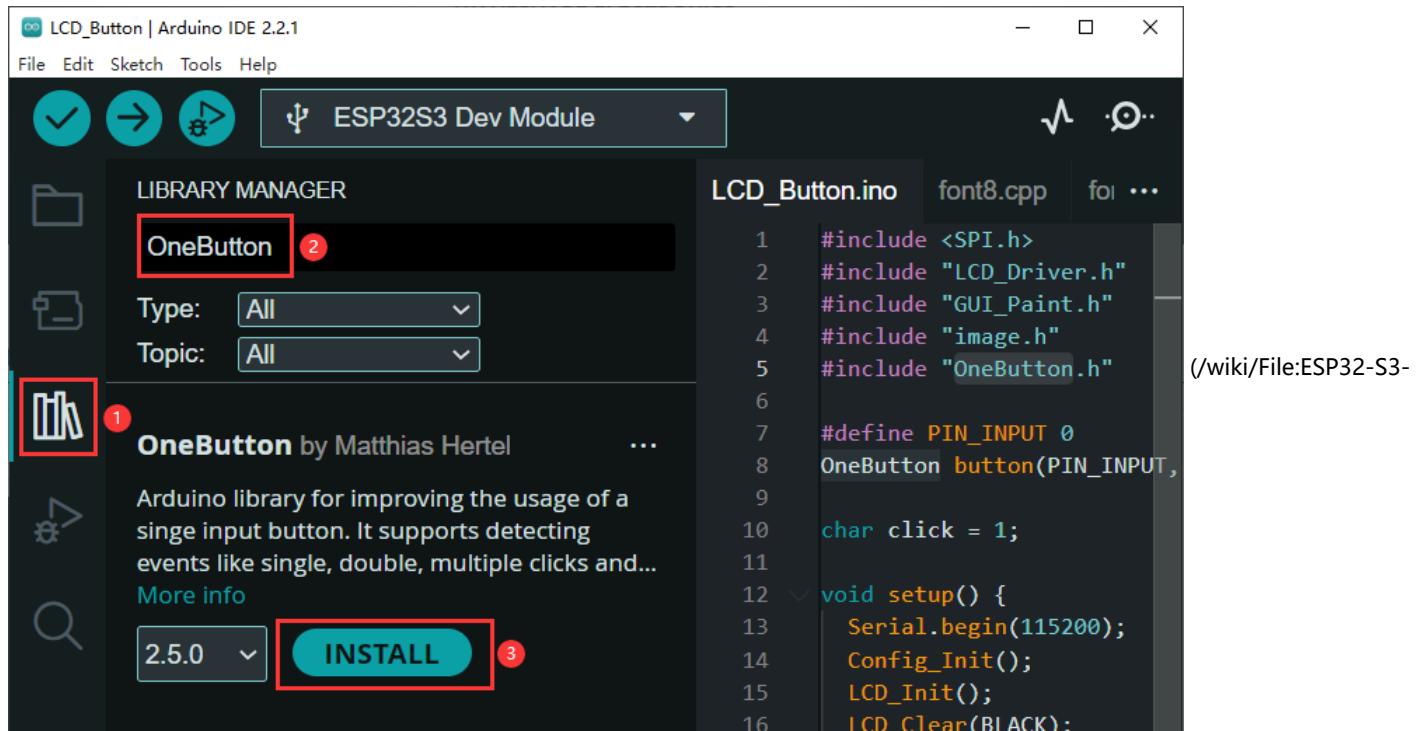
(/wiki/File:ESP32-S3-

Button

01-OneButton

With this demo, the boot key on the ESP32-S3-GEEK is modified as a multi-functional key, capable of executing different operations based on single-click, double-click, or long-press actions.

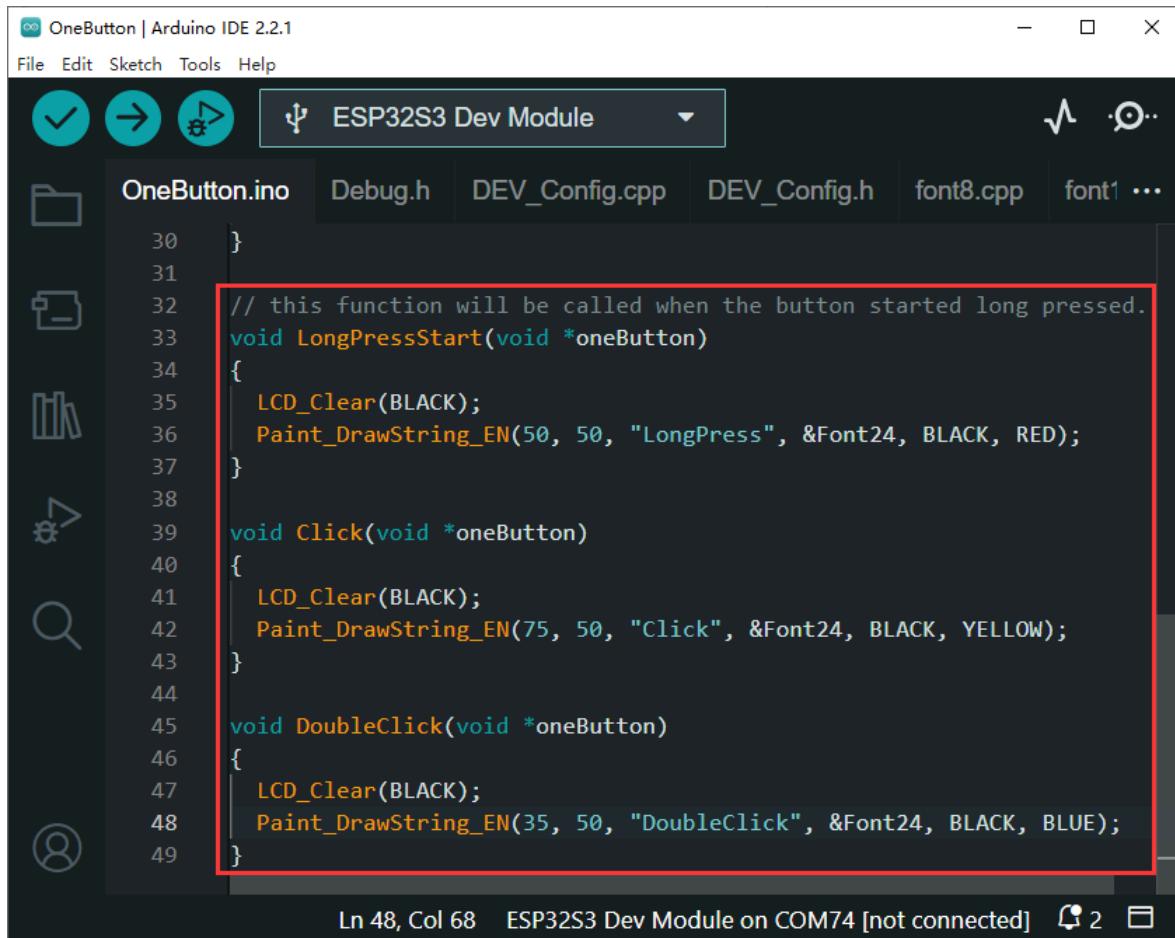
- Install OneButton library, LIBRARY MANAGER -> search for OneButton-> INSTALL.



GEEK_LCD_Time03.png)

- You can modify different processing functions by the following code. For more details, you can click here (<https://github.com/mathertel/OneButton>) to view.

Once a click is detected, it triggers: void Click(void *oneButton)
Once a double click is detected, it triggers: void DoubleClick(void *oneButton)
Once long press is detected, it triggers: void LongPressStart(void *oneButton)



```
OneButton | Arduino IDE 2.2.1
File Edit Sketch Tools Help
ESP32S3 Dev Module
OneButton.ino Debug.h DEV_Config.cpp DEV_Config.h font8.cpp font1 ...
30 }
31
32 // this function will be called when the button started long pressed.
33 void LongPressStart(void *oneButton)
34 {
35     LCD_Clear(BLACK);
36     Paint_DrawString_EN(50, 50, "LongPress", &Font24, BLACK, RED);
37 }
38
39 void Click(void *oneButton)
40 {
41     LCD_Clear(BLACK);
42     Paint_DrawString_EN(75, 50, "Click", &Font24, BLACK, YELLOW);
43 }
44
45 void DoubleClick(void *oneButton)
46 {
47     LCD_Clear(BLACK);
48     Paint_DrawString_EN(35, 50, "DoubleClick", &Font24, BLACK, BLUE);
49 }
```

Ln 48, Col 68 ESP32S3 Dev Module on COM74 [not connected] 2

GEEK_LCD_Time04.png)

UART

01-UART0

This demo uses ESP32-S3-GEEK to open UART0 and the serial port assistant for UART communication.

IIC

01-IIC_BME68X_Sensor

With this demo, the IIC hardware interface of the ESP32-S3-GEEK drives the IIC module.

- The demo uses the BME680 environmental sensor (<https://www.waveshare.com/bme68x-environmental-sensor.htm>). You can find detailed instructions and implementation effects in the tutorial (https://www.waveshare.com/wiki/BME680_Environmental_Sensor#Working_with_ESP32).

ADC

01-ADC_Read

This example allows the use of ESP32-S3-GEEK's GPIO interface for ADC sampling, enabling the reading of voltages within the 3.3V range. When using it, ensure a common ground and avoid exceeding the measurement range.

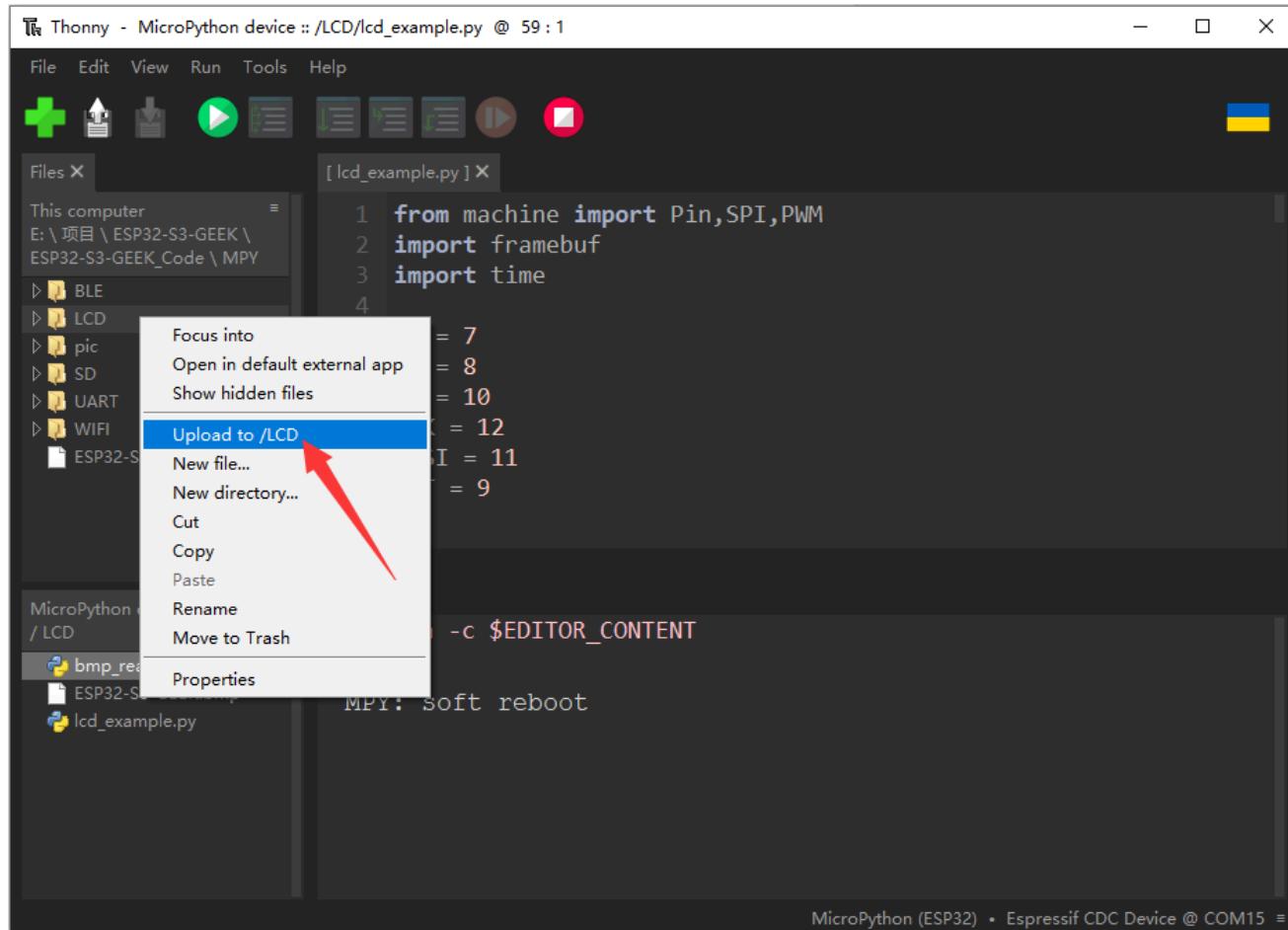
Micropython Sample Demo

Note: Before using the Micropython demo, you need to check whether the Micropython environment and the download setting are correctly configured. For more details, you can refer to [Micropython](#) (/wiki/ESP32-S3-GEEK#MicroPython).

LCD

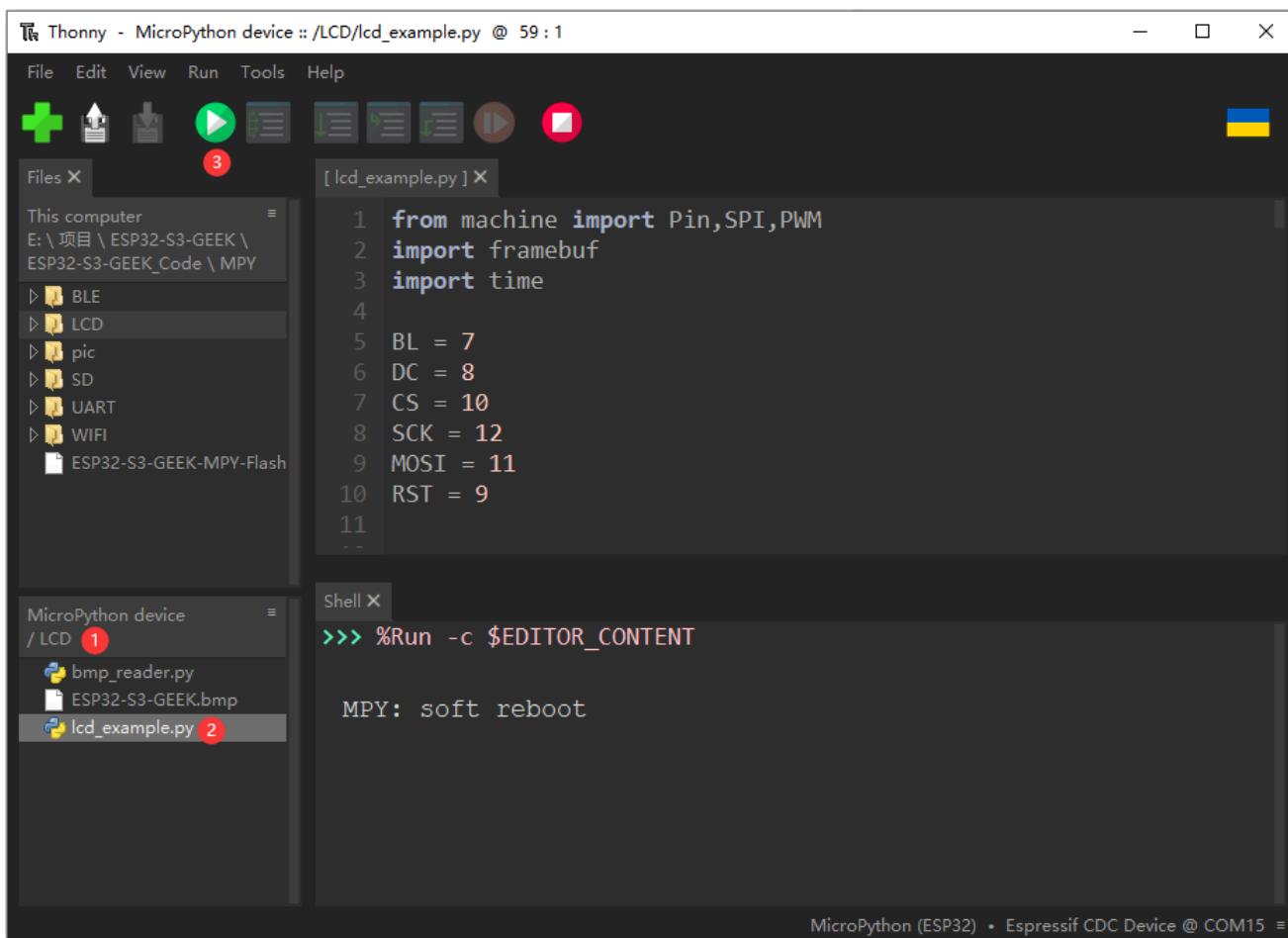
This demo uses the ESP32-S3-GEEK to open the LCD screen to display the text and BMP pictures.

- Upload the LCD file folder of the MPY file to ESP32-S3-GEEK.



(/wiki/File:ESP32-S3-GEEK-MPY_10.png)

- Enter the `lcd_example.py` of the ESP32-S3-GEEK, click to run, and observe the LCD screen display.



(/wiki/File:ESP32-S3-GEEK-MPY_11.png)

SD

This demo can read the SD card slot of the ESP32-S3-GEEK, and directly open the SD file folder of the ESP32-S3-GEEK to browse the files. Before using it, you need to insert the SD card into the card slot.

- Upload two files to the ESP32-S3-GEEK:

The screenshot shows the Thonny IDE interface. The title bar says "Thonny - MicroPython device :: /boot.py @ 25 : 1". The menu bar includes File, Edit, View, Run, Tools, and Help. Below the menu is a toolbar with icons for file operations. The left sidebar is titled "Files" and shows a file tree with "This computer" expanded, displaying "E:\项目\ESP32-S3-GEEK\ESP32-S3-GEEK_Code\MPY\SD" and two files: "boot.py" and "sdcard.py". A red arrow points from the text "Upload to /" in the sidebar towards the "Run" button in the toolbar. The main area has tabs for "[boot.py]" and "Shell". The code in [boot.py] is:

```
1 import machine
2 import sdcard
3 import uos
4
5 # Assign chip select (CS) pin (and start it high)
6 cs = machine.Pin(34, machine.Pin.OUT)
7
8 # Initialize SPI peripheral (start with 1 MHz)
9 spi = machine.SoftSPI(
10             baudrate=10000000,
11             polarity=0,
12             phase=0.
```

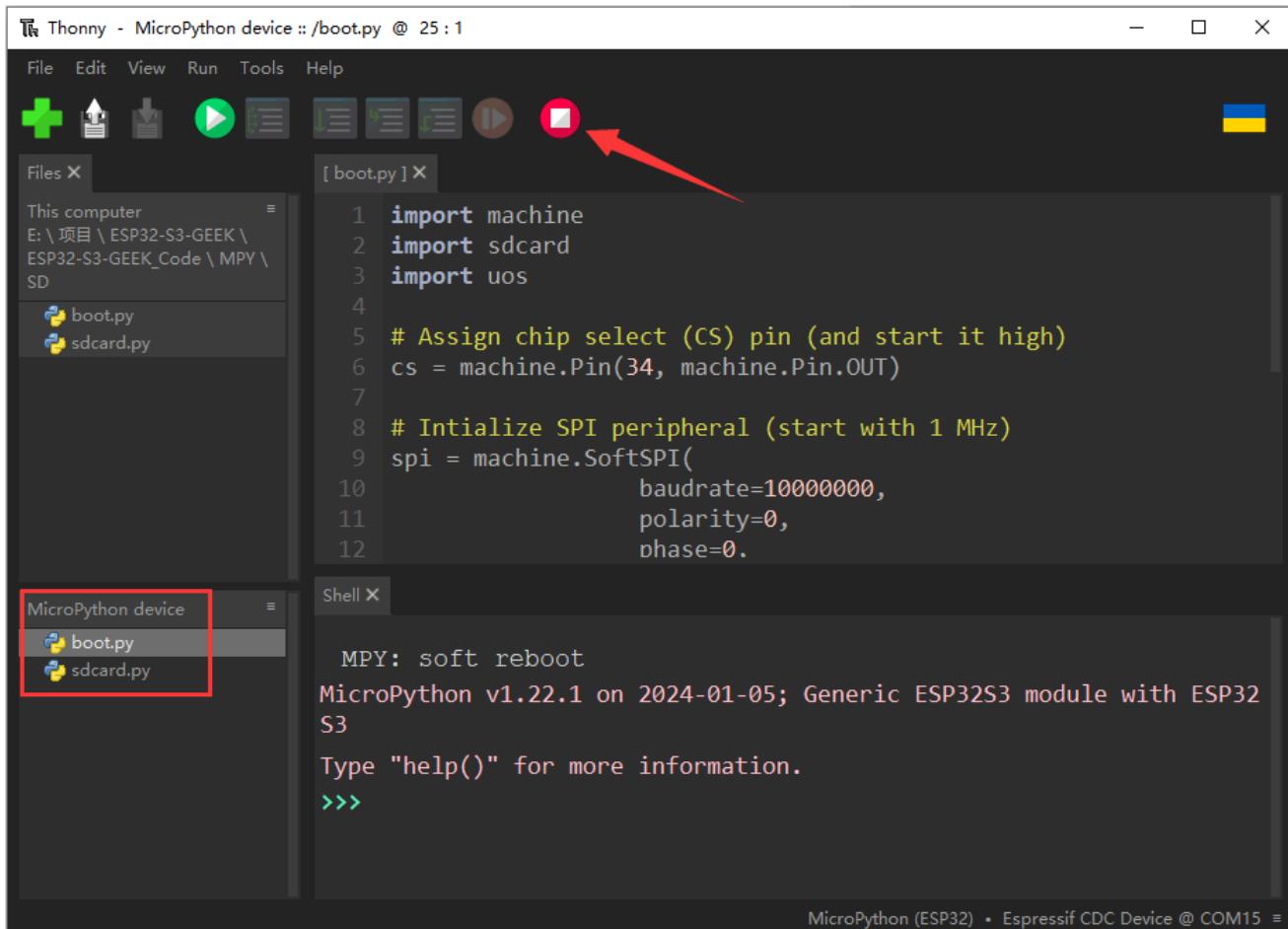
The "Shell" tab shows the MicroPython REPL output:

```
MPY: soft reboot
MicroPython v1.22.1 on 2024-01-05; Generic ESP32S3 module with ESP32 S3
Type "help()" for more information.
>>>
```

At the bottom right, it says "MicroPython (ESP32) • Espressif CDC Device @ COM15".

(/wiki/File:ESP32-S3-GEEK-MPY_12.png)

- After uploading, click on "STOP".



(/wiki/File:ESP32-S3-GEEK-MPY_13.png)

- You can see the SD file folder in the ESP32-S3-GEEK:

The screenshot shows the Thonny IDE interface. The top bar displays "Thonny - MicroPython device :: /boot.py @ 25 : 1". The menu bar includes File, Edit, View, Run, Tools, and Help. Below the menu is a toolbar with icons for file operations. The left sidebar is titled "Files" and shows a directory structure: "This computer" with "E:\项目\ESP32-S3-GEEK\ESP32-S3-GEEK_Code\MPY\SD" expanded, containing "boot.py" and "sdcard.py". The main area has tabs for "[boot.py] X" and "Shell X". The code in "boot.py" is:

```

1 import machine
2 import sdcard
3 import uos
4
5 # Assign chip select (CS) pin (and start it high)
6 cs = machine.Pin(34, machine.Pin.OUT)
7
8 # Initialize SPI peripheral (start with 1 MHz)
9 spi = machine.SoftSPI(
10             baudrate=10000000,
11             polarity=0,
12             phase=0.

```

The "Shell X" tab shows the MicroPython shell output:

```

MPY: soft reboot
MicroPython v1.22.1 on 2024-01-05; Generic ESP32S3 module with ESP32S3
Type "help()" for more information.
>>>

```

A red arrow points to the "sd" folder icon in the "Files" sidebar.

(/wiki/File:ESP32-S3-GEEK-MPY_14.png)

WIFI

01-WIFI_AP

This demo turns on the AP mode of WIFI using the ESP32-S3-GEEK, and the PC can connect to its WIFI.

- essid is the AP name (ESP32-S3-GEEK) created by ESP32-S3-GEEK, the password (Waveshare) is the password used to connect to the AP.

The screenshot shows the Thonny IDE interface. The top bar displays "device :: /WIFI_AP.py @ 16 : 1". The menu bar includes Tools and Help. Below the menu is a toolbar with icons for file operations. The left sidebar is titled "[WIFI_AP.py] X". The main area shows the code for "WIFI_AP.py":

```

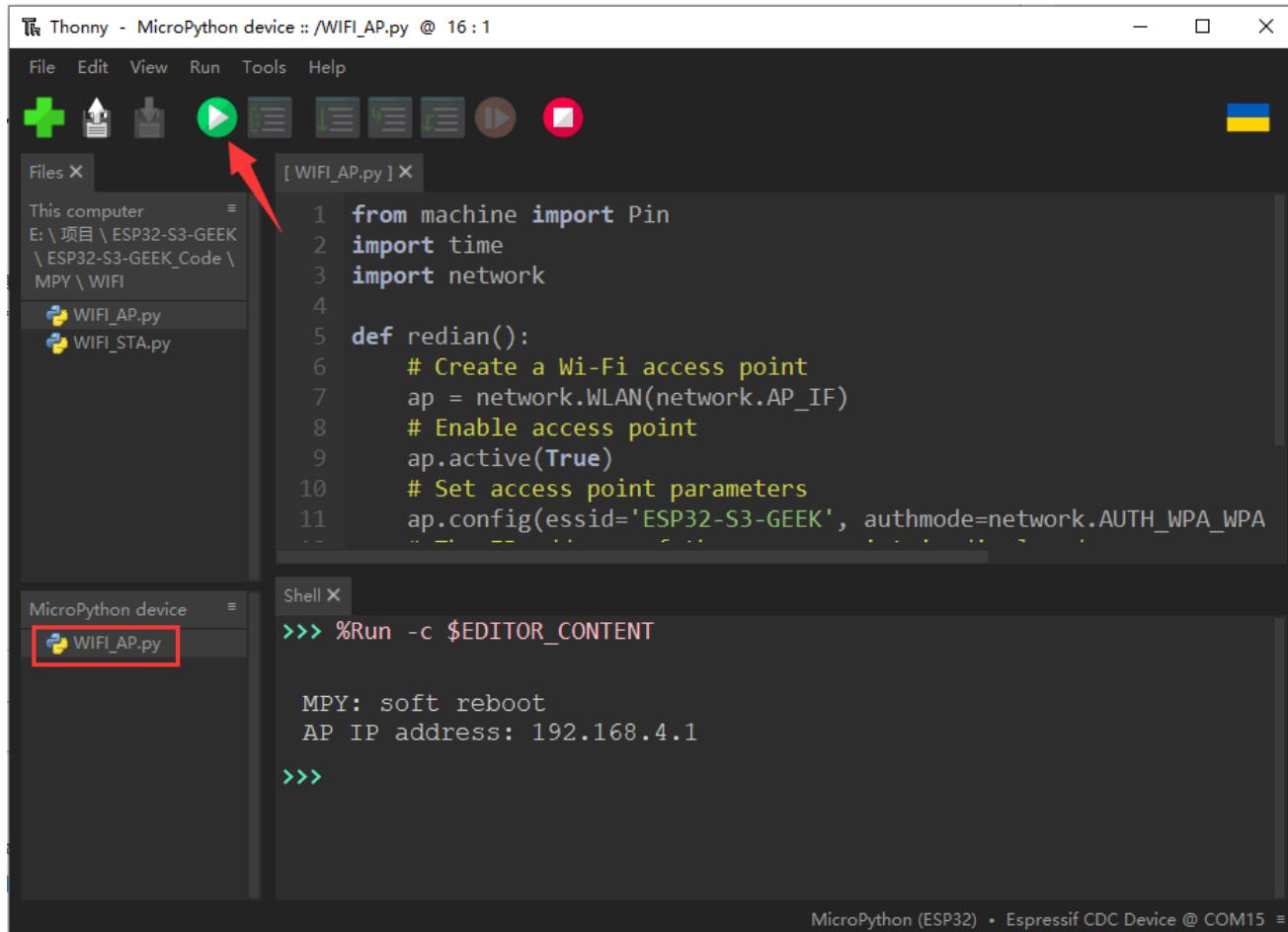
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16

```

Line 11 contains highlighted code: `ifconfig(essid='ESP32-S3-GEEK', authmode=network.AUTH_WPA_WPA2_PSK, password='Waveshare')`. A red box highlights this line.

(/wiki/File:ESP32-S3-GEEK-MPY_27.png)

- Upload the demo to the ESP32-S3-GEEK and click to run.



(/wiki/File:ESP32-S3-GEEK-MPY_15.png)

- Open the WIFI on the PC to connect, input the password: Waveshare.

02-WIFI_STA

This demo can use ESP32-S3-GEEK to open the STA mode of WIFI, and it can connect to other WIFI or hotspot opened on the PC.

- In the STA mode, the ESP32-S3-GEEK connects to the WIFI with ssid of ESP32-S3-GEEK and password of Waveshare.

The screenshot shows the Thonny IDE interface. The top bar displays "Thonny - MicroPython device :: /WIFI_STA.py @ 15 : 13". The menu bar includes File, Edit, View, Run, Tools, and Help. Below the menu is a toolbar with icons for file operations. The left sidebar shows a file tree with "This computer" expanded, displaying "E:\ 项目 \ ESP32-S3-GEEK \ ESP32-S3-GEEK_Code \ MPY \ WiFi" and two files: "WIFI_AP.py" and "WIFI_STA.py". The main code editor window is titled "[WIFI_STA.py] X" and contains the following Python code:

```

4
5 def do_connect():
6     wlan = network.WLAN(network.STA_IF)
7     wlan.active(True)
8     if not wlan.isconnected():
9         print('connecting to network...')
10        wlan.connect('ESP32-S3-GEEK', 'Waveshare')
11        while not wlan.isconnected():
12            pass
13        print('network config:', wlan.ifconfig())
14
15 do_connect()

```

The line `wlan.connect('ESP32-S3-GEEK', 'Waveshare')` is highlighted with a red box. The bottom right corner of the code editor has a small blue and yellow flag icon.

The bottom section is a "Shell" window titled "MicroPython device" with the following content:

```

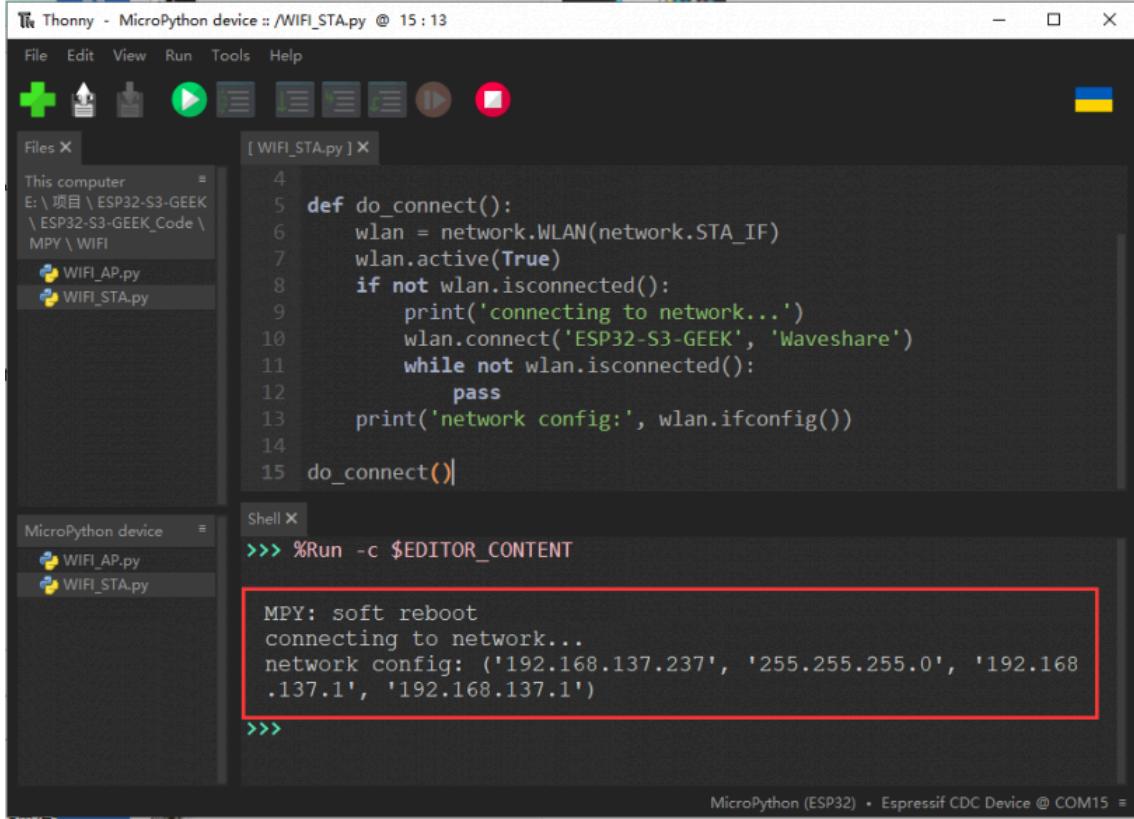
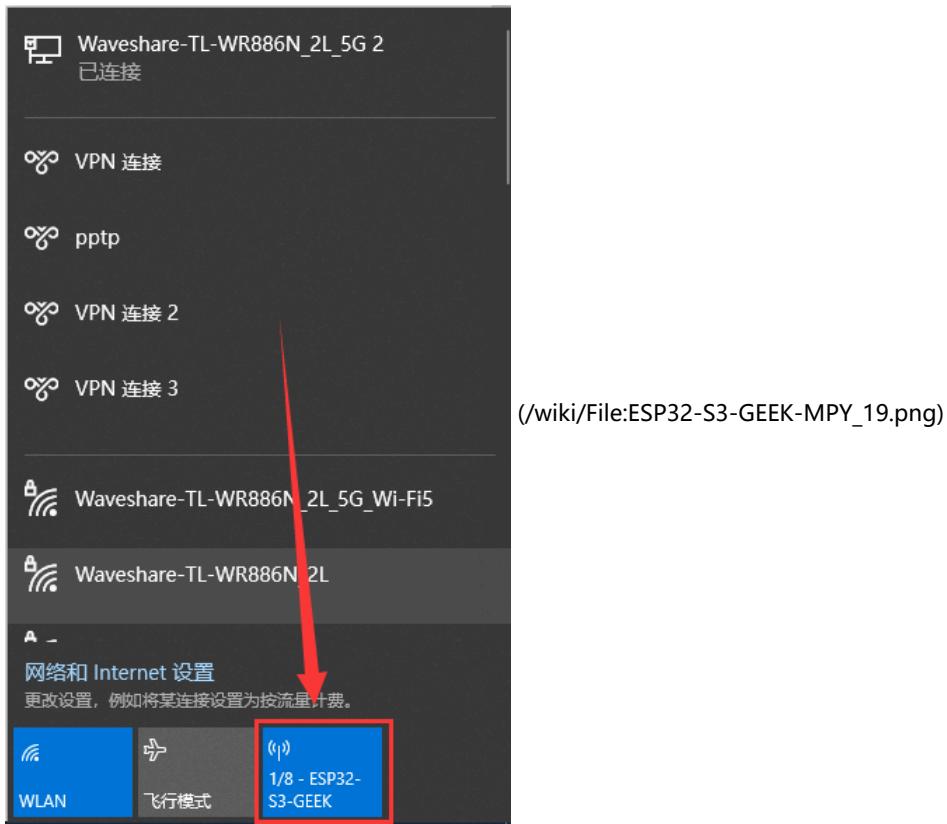
MPY: soft reboot
MicroPython v1.22.1 on 2024-01-05; Generic ESP32S3 module with ESP32S3
Type "help()" for more information.
>>>

```

At the very bottom right of the shell window, it says "MicroPython (ESP32) • Espressif CDC Device @ COM15".

(/wiki/File:ESP32-S3-GEEK-MPY_18.png)

- PC opens a hotspot with ssid "ESP32-S3-GEEK" and password "Waveshare" (The WiFi network connected by ESP32-S3-GEEK needs to be on the 2.4GHz band. If there is no 2.4GHz band available, you can use the PC to open a hotspot, selecting the network band as "any available frequency"). Remember to modify the ssid and password to the desired WiFi network name and password. Upload the demo to ESP32-S3-GEEK and then click to run.



GEEK-MPY_20.png)

BLE

This demo opens the Bluetooth of the ESP32-S3-GEEK and uses the Bluetooth and the Bluetooth debugger on the phone to communicate.

- "ESP32-S3-GEEK" is the Bluetooth name.

The screenshot shows the Thonny IDE interface. The top bar displays "Thonny - MicroPython device :: /WS_Bluetooth.py @ 68 : 1". The menu bar includes File, Edit, View, Run, Tools, and Help. Below the menu is a toolbar with icons for file operations. The left sidebar shows a file tree with "This computer" expanded, showing "E:\ 项目\ ESP32-S3-GEEK\ ESP32-S3-GEEK_Code\ MPY\ BLE" and a file "WS_Bluetooth.py". The main area has two tabs: "[WS_Bluetooth.py]" which contains the Python code, and "Shell". The code imports the `bluetooth` module and defines several variables: `_SERVICE_UUID`, `_RX_CHARACTERISTIC_UUID`, `_TX_CHARACTERISTIC_UUID`, and `BLEDeviceName`. The line `BLEDeviceName= "ESP32-S3-GEEK"` is highlighted with a red rectangle. The "Shell" tab shows the output of running the code: "MPY: soft reboot", "MicroPython v1.22.1 on 2024-01-05; Generic ESP32S3 module with ESP32S3", "Type "help()" for more information.", and a prompt ">>>". The status bar at the bottom right indicates "MicroPython (ESP32) • Espressif CDC Device @ COM15".

(/wiki/File:ESP32-S3-GEEK-MPY_21.png)

- Click the Run button to successfully open Bluetooth and the Bluetooth name will be printed.

The screenshot shows the Thonny IDE interface. At the top, there's a menu bar with File, Edit, View, Run, Tools, Help. Below the menu is a toolbar with icons for file operations like Open, Save, and Run. A red arrow points to the Run icon (a play button). The left sidebar has a 'Files' tab and a tree view showing 'This computer' and a project folder 'E:\项目\ESP32-S3-GEEK\ESP32-S3-GEEK_Code\MPY\BLE'. Inside the tree, 'WS_Bluetooth.py' is selected. The main area contains the code for 'WS_Bluetooth.py':

```

import bluetooth
SERVICE_UUID = '4fafc201-1fb5-459e-8fcc-c5c9c331914a'
RX_CHARACTERISTIC_UUID = 'beb5483e-36e1-4688-b7f5-ea07361b26a6'
TX_CHARACTERISTIC_UUID = 'beb5484a-36e1-4688-b7f5-ea07361b26a6'
BLEDeviceName= "ESP32-S3-GEEK"
Bluetooth_Mode = 2 # Used to distinguish data sources
TxCharacteristic = 0
RxCharacteristic = 0
ble=0

```

Below the code is a 'Shell' window with the command:

```
>>> %Run -c $EDITOR_CONTENT
```

The terminal output shows:

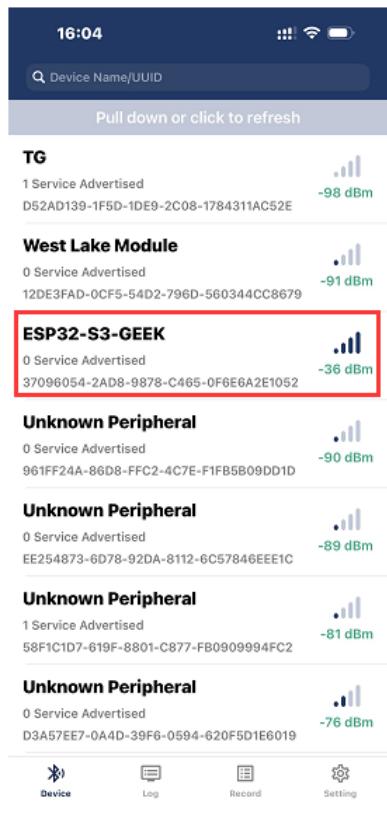
```

MPY: soft reboot
ESP32-S3-GEEK
>>>

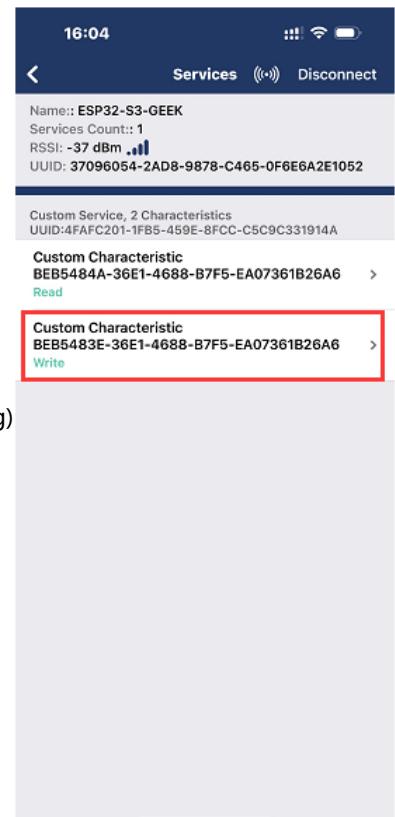
```

(/wiki/File:ESP32-S3-GEEK-MPY_22.png)

- Use your phone's Bluetooth debugging assistant to connect.

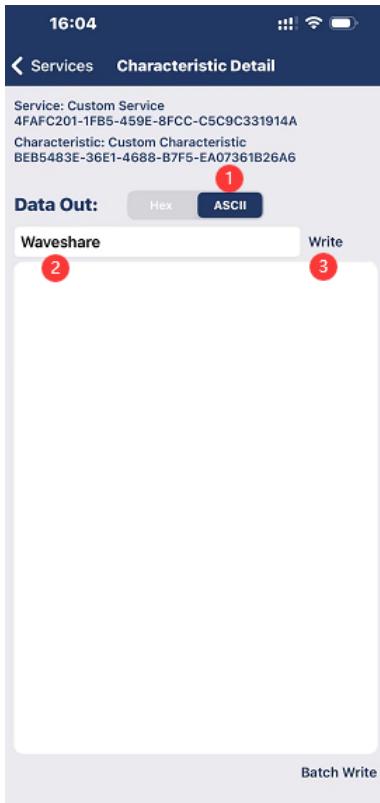


S3-GEEK-MPY_25.png)

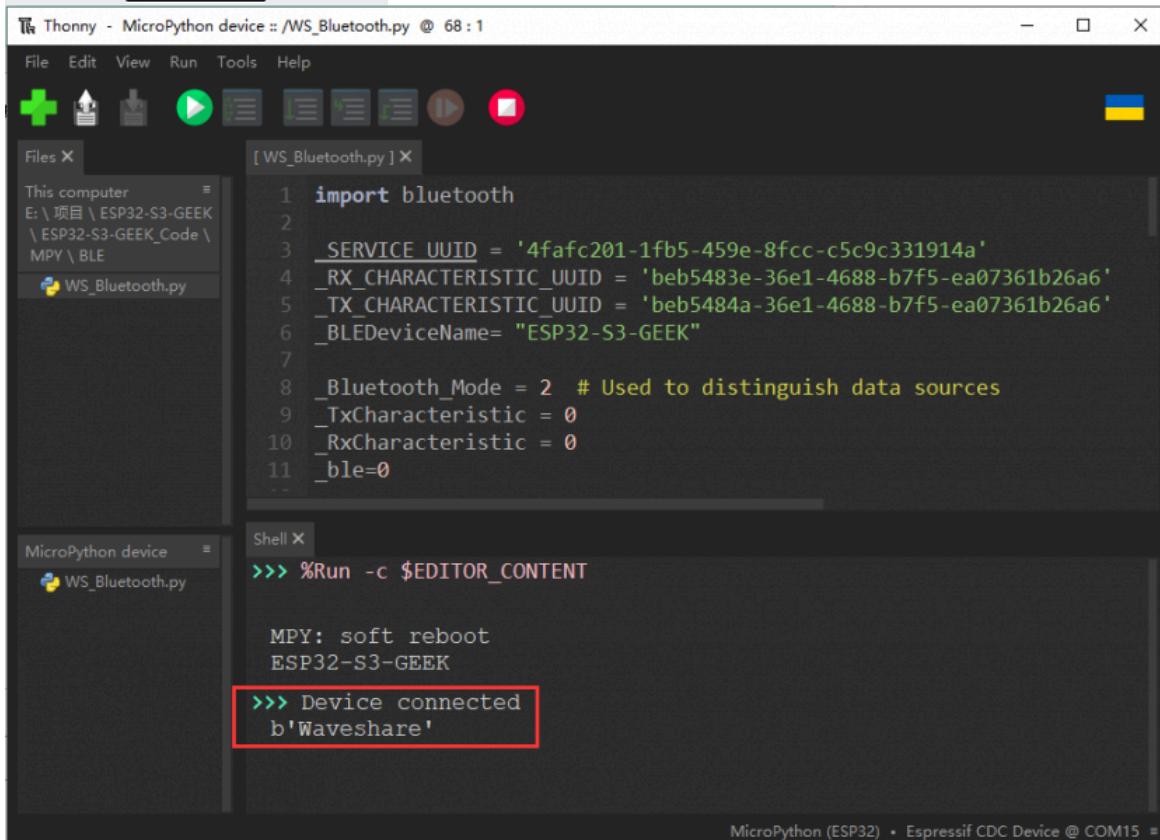


(/wiki/File:ESP32-

- Use your phone's Bluetooth debugging assistant to send a message, ESP32 receives the message and prints it out.



(/wiki/File:ESP32-S3-GEEK-MPY_26.png)



(/wiki/File:ESP32-S3-

GEEK-MPY_23.png)

UART

This demo can use the ESP32-S3-GEEK to open UART0, and open the serial port assistant for UART communication.

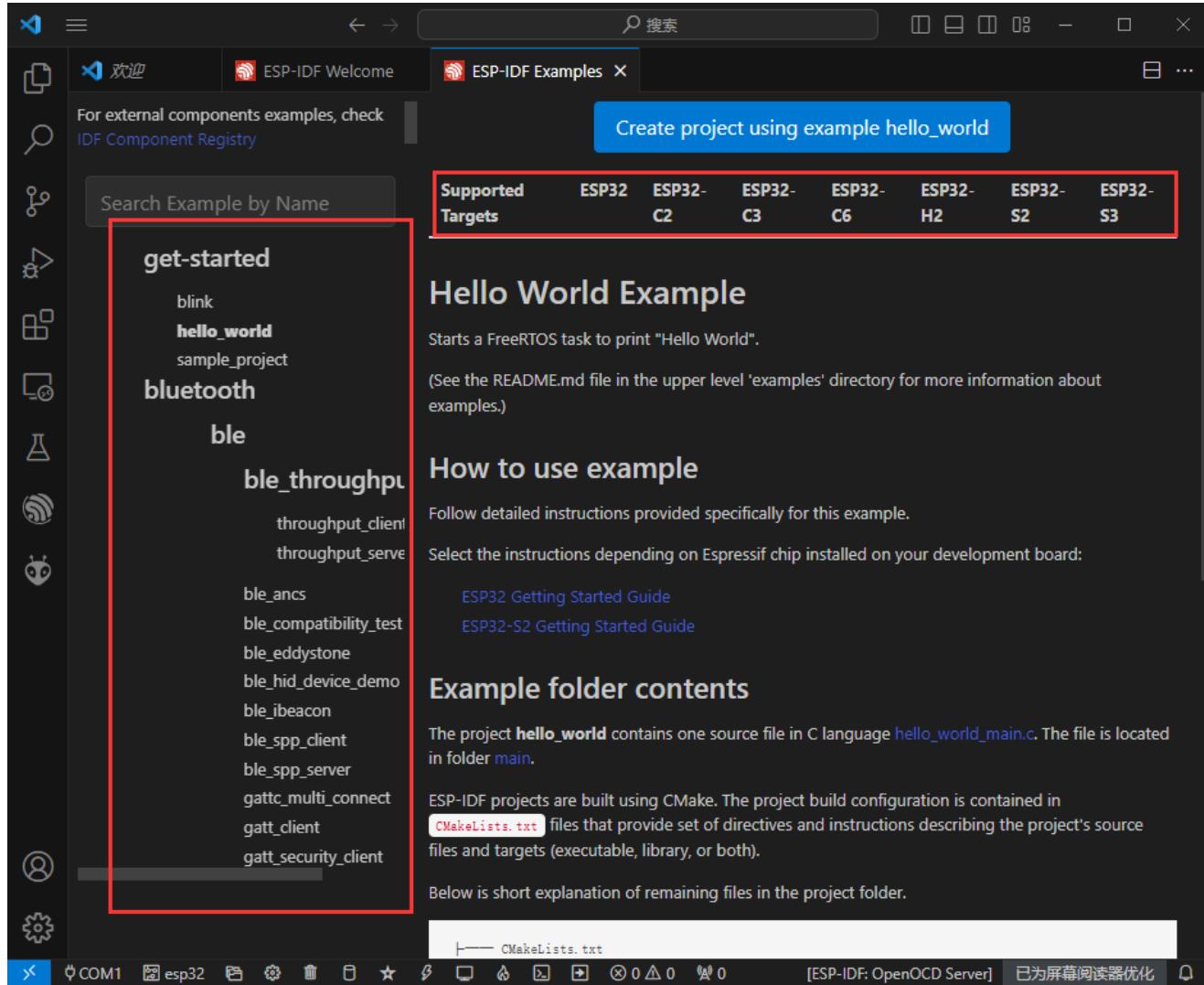
ESP-IDF Sample Demo

Note: Before using ESP-IDF examples, ensure that the ESP-IDF environment and download settings are correctly configured. For specific instructions, refer to #ESP-IDF.

ESP-IDF Official Demo

Can be programmed in ESP-IDF using the official demos, only to provide the official routines to find the use of the method, the specific development needs to be the user's learning and programming.

The use of the official demo can be found in #Official Demo Usage, we need to pay attention to the use of the device COM port, chip selection, and if burning method is correct.



(/wiki/File:ESP32-S3-Geek_Demo_Official.png)

Wireless_USB_flash_drive

This demo allows ESP32-S3-GEEK to function as a USB flash driver with wireless access. When combined with an SD card, it transforms into a high-capacity wireless storage unit. Additionally, it enables connecting to ESP32-S3-GEEK's hotspot to engage in HTTP file server operations for both uploads and downloads, significantly enhancing user convenience.

- Before programming the demo, please insert the SD card into the SD card slot, and check whether the programming chip is esp32s3, and the COM port is correct or not. The programming method selects UART.

The screenshot shows the Visual Studio Code interface for an ESP-IDF project titled "usb_msc_wireless_disk". The left sidebar contains icons for file operations, search, and other development tools. The main workspace shows the file "app_main.c" with code related to USB Mass Storage Driver (MSD) and WiFi. The terminal tab displays the following log output:

```

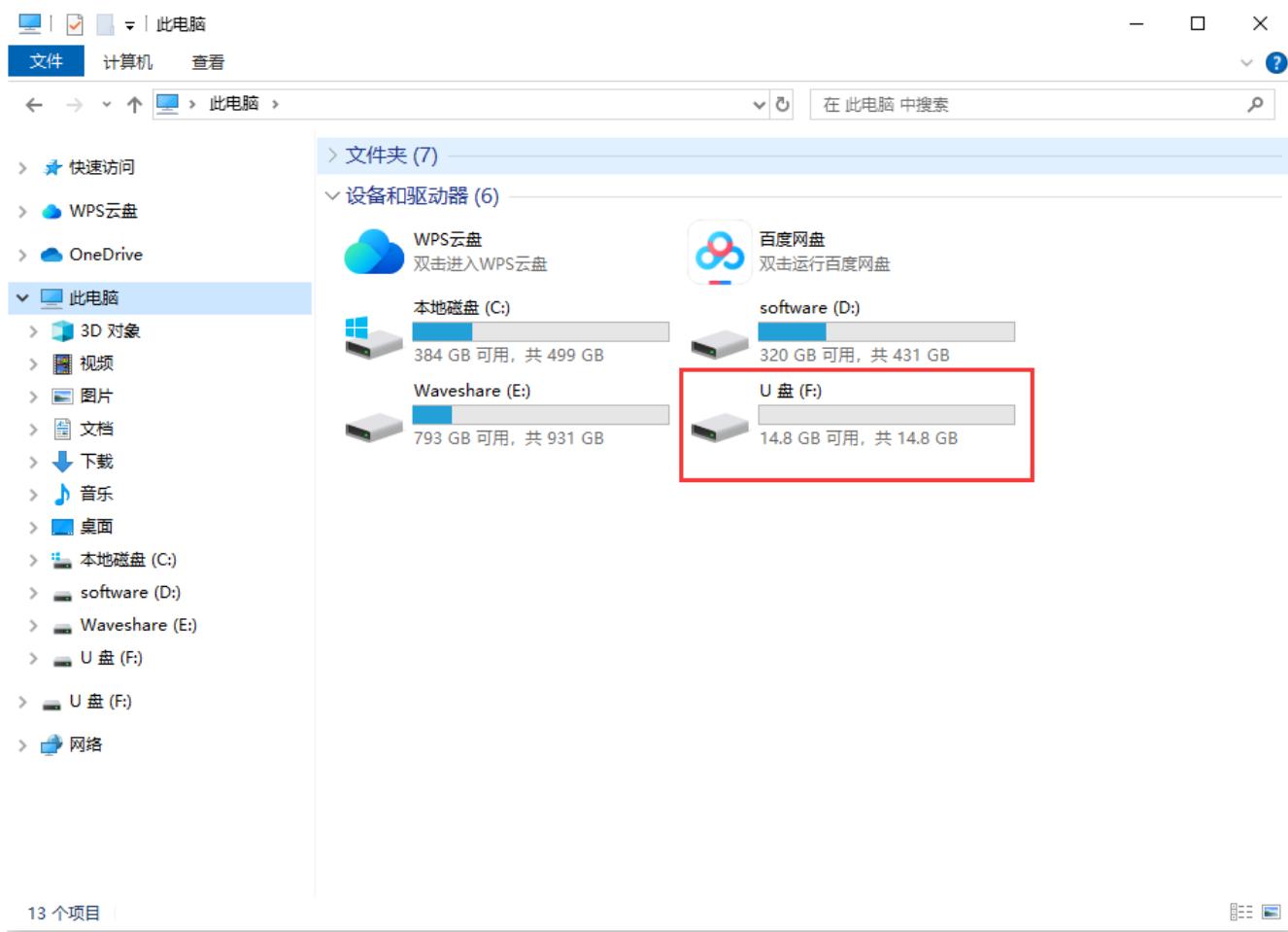
I (781) wifi:Init dynamic tx buffer num: 32
I (791) wifi:Init static tx FG buffer num: 2
I (791) wifi:Init static rx buffer size: 1600
I (791) wifi:Init static rx buffer num: 10
I (801) wifi:Init dynamic rx buffer num: 32
I (801) wifi_init: rx ba win: 6
I (811) wifi_init: tcpip mbox: 32
I (811) wifi_init: udp mbox: 6
I (811) wifi_init: tcp mbox: 6
I (821) wifi_init: tcp tx win: 5744
I (821) wifi_init: tcp rx win: 5744
I (831) wifi_init: WiFi RX IRAM OP enabled
I (1911) wifi: wifi_init_softap finished.SSID:ESP32-S3-GEEK password:Waveshare
I (1911) phy_init: phy_version 601,98f2a71,Jun 29 2023,09:58:12
1961) wi 2 mode : softAP (48:27:e2 3:c3:cd)

```

The terminal also shows a red circle around the number "4" at the bottom right, likely indicating a new message or error. The bottom status bar includes icons for COM7, esp32s3, UART, LF, C, Win32, and a yellow triangle icon.

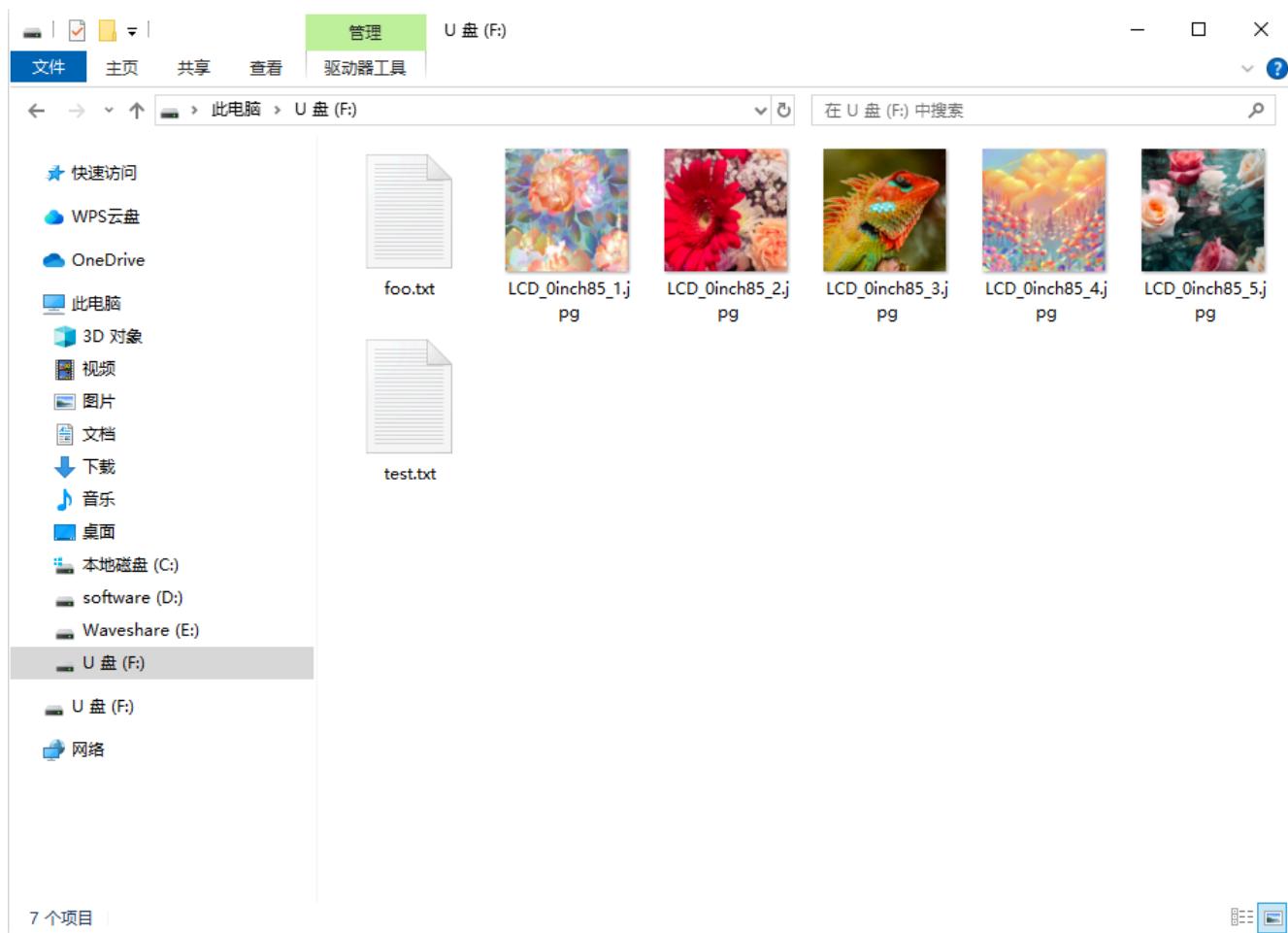
(/wiki/File:ESP32-S3_wireless_disk.png)

- After programming, unplug and plug the ESP32-S3-GEEK again, and you can see a new USB flash driver.



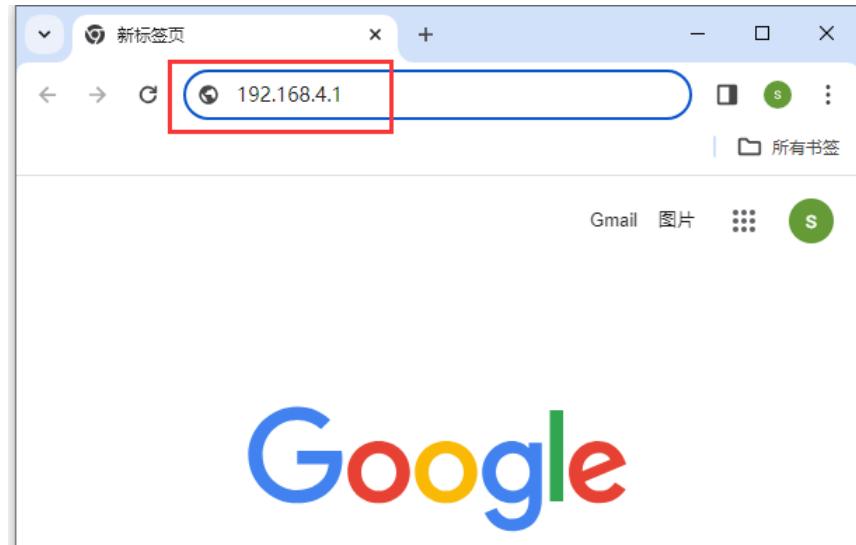
(/wiki/File:ESP32-S3_wireless_disk02.png)

- You can open the USB drive to browse through the files on the SD card and perform various operations such as adding, deleting, modifying, and checking files.



(/wiki/File:ESP32-S3_wireless_disk03.png)

- Enter the PC's WIFI setting, connect to the ESP32-S3-GEEK's AP, and input the password of "Waveshare".
- After connecting successfully, open the browser, and log in to IP: 192.168.4.1.



- After login successfully, you can upload and download files wirelessly.

The screenshot shows a web-based file server interface. At the top right, there are buttons for 'Upload a file' (with a red box around it), 'Set path on server', and 'Upload'. Below this is a table listing files and directories:

Name	Type	Size (Bytes)	Delete
test.txt	file	1048576	Delete
foo.txt	file	13	Delete
System Volume Information	directory	0	Delete
LCD_0inch85_5.jpg	file	24104	Delete
LCD_0inch85_1.jpg	file	26983	Delete
LCD_0inch85_2.jpg	file	24776	Delete
LCD_0inch85_3.jpg	file	25146	Delete
LCD_0inch85_4.jpg	file	25555	Delete

(/wiki/File:ESP32-S3_wireless_disk05.png)

Resource

Schematic

- Schematic (<https://files.waveshare.com/wiki/ESP32-S3-GEEK/ESP32-S3-GEEK-Schematic1.pdf>)

Demo

- Demo (https://files.waveshare.com/wiki/ESP32-S3-GEEK/ESP32-S3-GEEK_Code.zip)

Development Software

MicroPython

- ESP32-S3-GEEK-MPY-Flash (<https://files.waveshare.com/wiki/ESP32-S3-GEEK/ESP32-S3-GEEK-MPY-Flash.zip>)
- MicroPython Official Firmware (https://micropython.org/download/ESP32_GENERIC_S3/)

Image Modeling

- Zimo221.7z (<https://files.waveshare.com/upload/c/c6/Zimo221.7z>)
- Image2Lcd.7z (<https://files.waveshare.com/upload/3/36/Image2Lcd.7z>)
- Image_Extraction (https://www.waveshare.com/wiki/Image_Extraction)
- Ink Screen Font Library Tutorial (https://www.waveshare.com/wiki/Ink_Screen_Font_Library_Tutorial)

Debugging Tool

- sscom serial port assistant (<https://files.waveshare.com/upload/b/b3/Sscom5.13.1.zip>)
- NetAssist tool (<https://files.waveshare.com/upload/9/95/Netassist.zip>)
- IOS Bluetooth assistant (<https://apps.apple.com/us/app/bluetoothassistant/id1536579599>)
- Android Bluetooth debugging tool (<https://files.waveshare.com/wiki/ESP32-S3-GEEK/Android%E8%93%9D%E7%89%99%E8%B0%83%E8%AF%95%E5%A9%E6%89%8B.zip>)

- Android TCP debugging tool (<https://files.waveshare.com/wiki/ESP32-S3-GEEK/Tcp%E8%B0%83%E8%AF%95%E5%8A%A9%E6%89%8B.zip>)
- Espressif APP (https://www.espressif.com.cn/en/support/download/apps?keys=&field_type_tid%5B%5D=842)

Datasheet

- ESP32-S3 Datasheet (https://files.waveshare.com/upload/f/f7/Esp32-s3_datasheet_en_%281%29.pdf)
- ESP32-S3 Technical Reference Manual (https://files.waveshare.com/upload/9/9e/Esp32-s3_technical_reference_manual_en_%281%29.pdf)
- ESP32-S3-WROOM-1 Datasheet (https://files.waveshare.com/upload/8/87/Esp32-s3-wroom-1_wroom-1u_datasheet_en.pdf)

ESP32 Official Document

- ESP32-Arduino Official Document (<https://docs.espressif.com/projects/arduino-esp32/en/latest/index.html>)
- ESP-IDF Official Document (<https://docs.espressif.com/projects/esp-idf/en/release-v5.0/esp32s3/index.html>)

MicroPython Official Document

- MicroPython Official Document (<https://docs.micropython.org/en/latest/>)

FAQ

Question:ESP32-S3-GEEK programming failed?

Answer:

- ① During programming, long-press the Boot button, insert the USB, and then release the button to enter the ESP32-S3-GEEK Download mode for the programming process. After programming, reconnecting the USB will enter the SPI_FAST_FLASH_BOOT mode, run the demo.
- ② Try turning off the PC's Bluetooth switch and then directly burn and run (indicating the conflict between the Bluetooth driver and the ESP32-S3-GEEK's COM driver results in the programming failure).

Question:ESP32-S3-GEEK successfully connected to Waveshare Cloud, why can't it send or receive MQTT messages?

Answer:

Please check if the Pub Topic and Sub Topic are correctly filled in, and verify if the device is online in the Waveshare Cloud. Both factors could impact the ability to send or receive MQTT messages.

Question:When the SD card is inserted into the ESP32-S3-GEEK, can the computer directly function as a USB flash driver? After inserting the SD card, can you directly view its files using Windows File Explorer when connected to the computer?

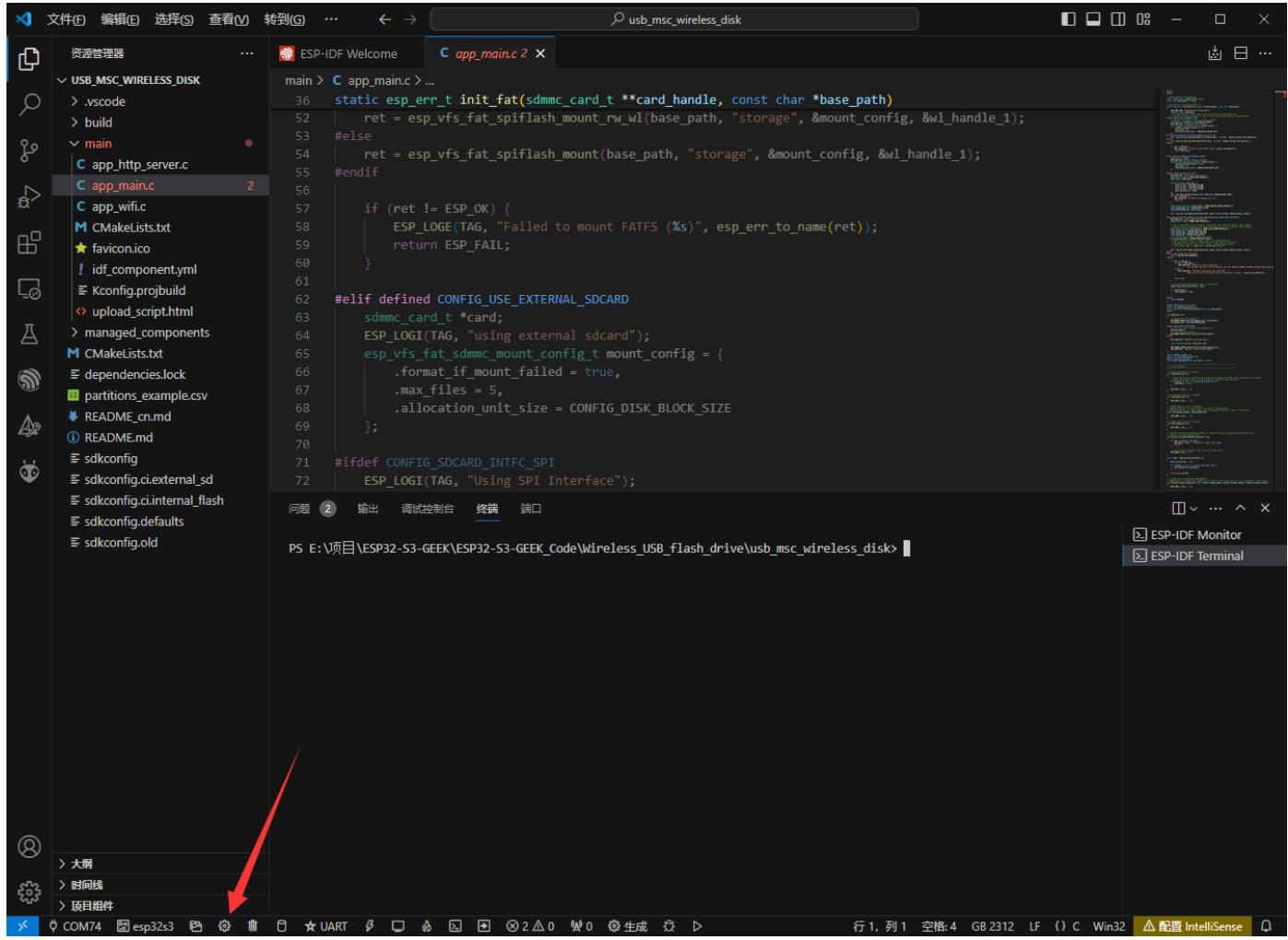
Answer:

Yes, you need to flash #Wireless_USB_flash_drive, in the SD card image, SPIFS only supports up to 32GB.

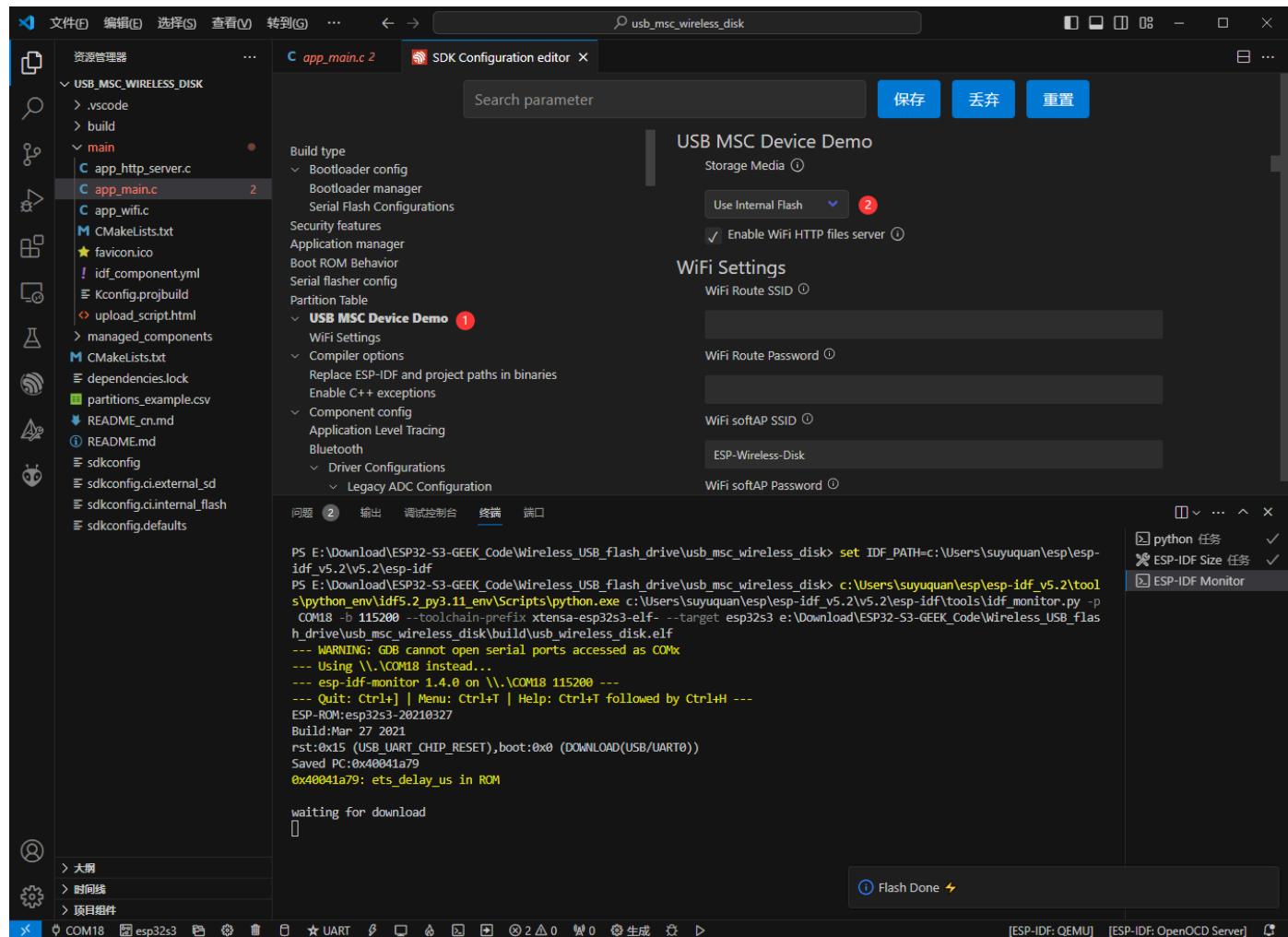
Question:ESP32-S3-GEEK burned the Wireless_USB_flash_drive demo, and the 16G MicroSD card is inserted into the SD card slot, but the display memory is only 1.4M?

Answer:

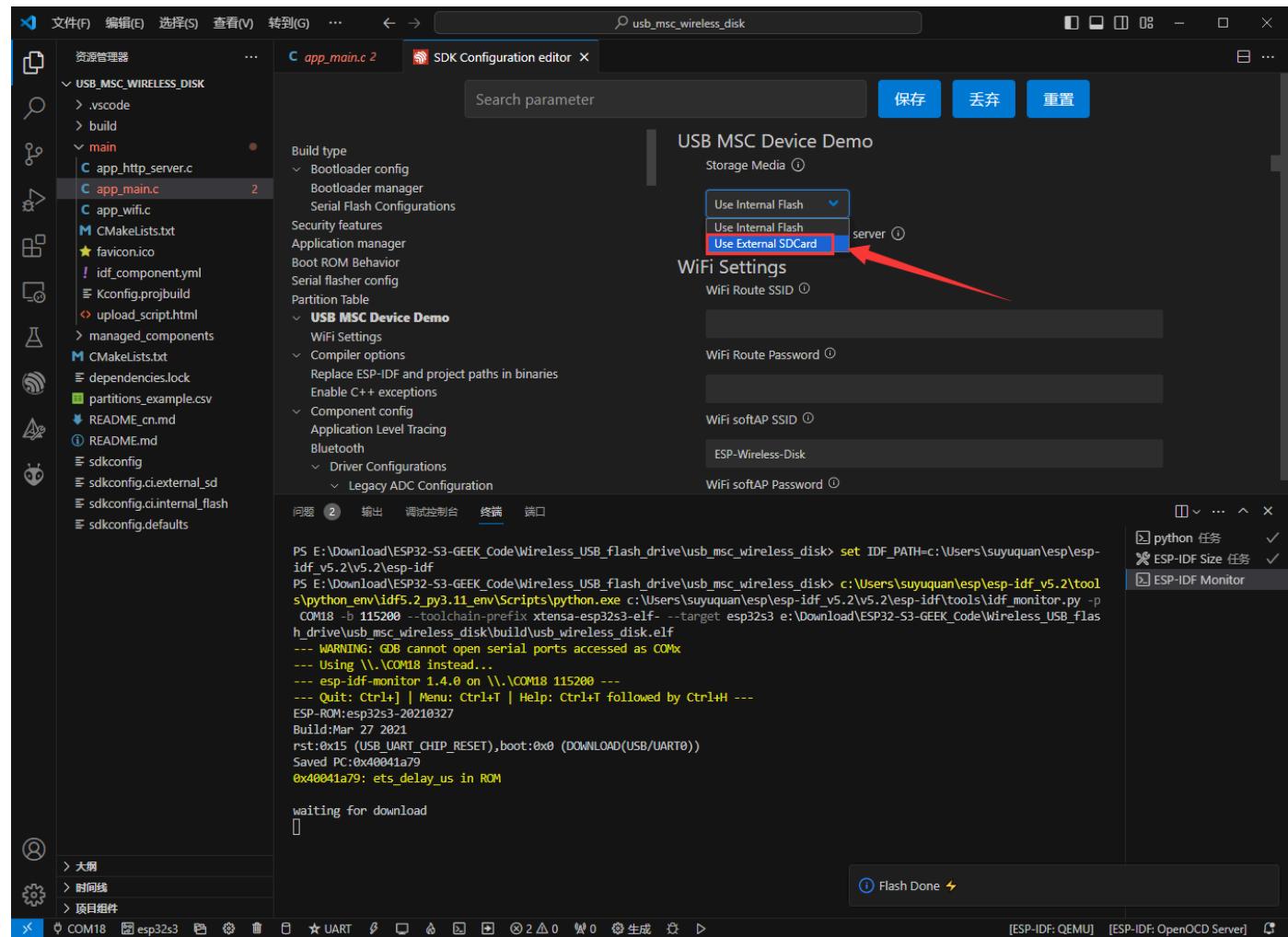
- 1. Click Settings in the lower left corner.



- 2. Modified "Use Internal Flash" to "Use External SDCard".

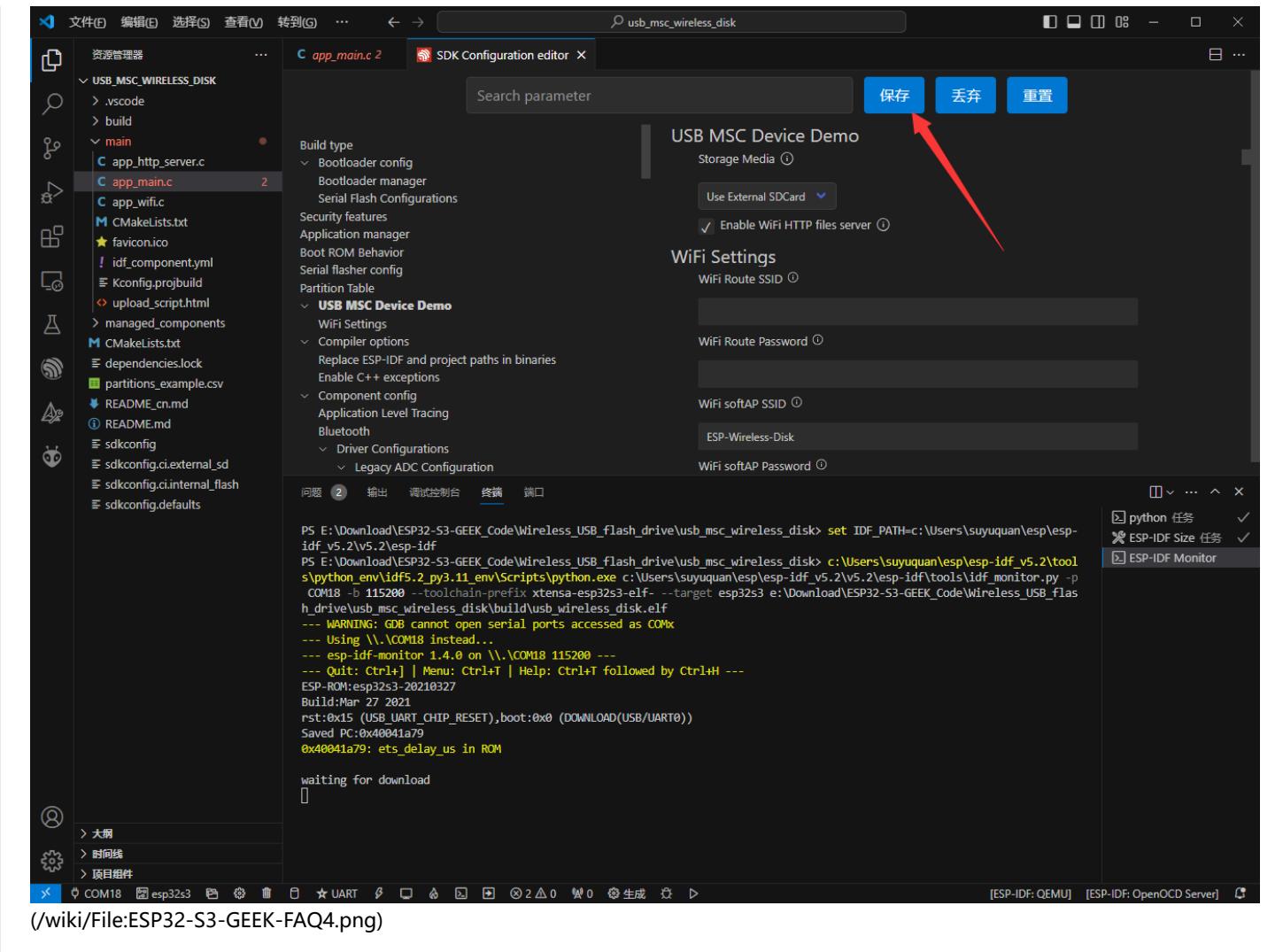


(/wiki/File:ESP32-S3-GEEK-FAQ2.png)



(/wiki/File:ESP32-S3-GEEK-FAQ3.png)

- 3. Save the settings and burn again.



Support

Technical Support

If you need technical support or have any feedback/review, please click the **Submit Now** button to submit a ticket. Our support team will check and reply to you within 1 to 2 working days. Please be patient as we make every effort to help you to resolve the issue.

Working Time: 9 AM - 6 PM GMT+8 (Monday to Friday)

[Submit Now \(https://service.waveshare.com/\)](https://service.waveshare.com/)

Retrieved from "<https://www.waveshare.com/w/index.php?title=ESP32-S3-GEEK&oldid=87326>" (<https://www.waveshare.com/w/index.php?title=ESP32-S3-GEEK&oldid=87326>)