

ESP32-C6-Zero

From Waveshare Wiki

Jump to: navigation, search

Overview

The ESP32-C6-Zero is a low-cost, high-performance microcontroller development board with a compact size and rich peripheral interfaces.

Adopts ESP32-C6FH4 as the main chip, with RISC-V 32-bit single-core processor, support up to 160 MHz, and built-in 320KB ROM, 512KB HP SRAM and 16KB LP SRAM. It can be compatible with multiple peripheral devices and is easier to use in different application scenarios.

You can choose ESP-IDF development environment or Arduino IED for development so that you can easily and quickly get started and apply it to the product.

**ESP32-C6-Zero
Without header**



(<https://www.waveshare.com/esp32-c6-zero.htm>)

ESP32-C6, USB Type-C

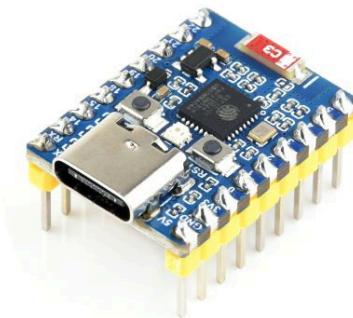
Features

- Adopts ESP32-C6FH4 chip, with RISC-V 32-bit single-core processor, support up to 160 MHz.
- Integrated 320KB ROM, 512KB HP SRAM, 16KB LP SRAM and 4MB Flash memory.

- Integrated WiFi 6, Bluetooth 5 and IEEE 802.15.4 (Zigbee 3.0 and Thread) wireless communication, with superior RF performance.
- Type-C connector, easier to use.
- Rich peripheral interfaces, better compatibility and expandability.
- Castellated module allows soldering directly to carrier boards.
- Support multiple low-power modes, adjustable balance between communication distance, data rate and power consumption to meet the power requirements of various application scenarios.

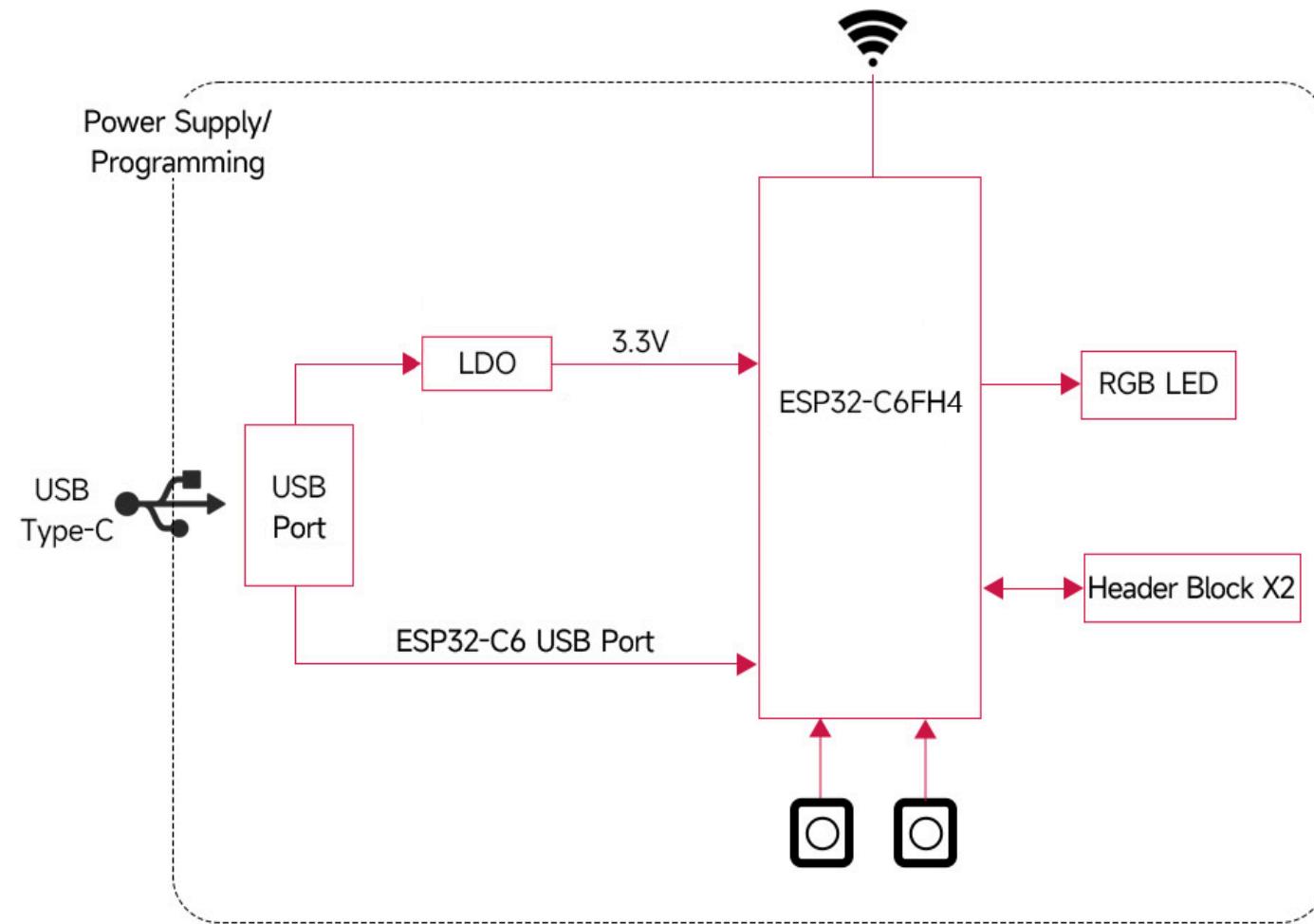
Function Diagram

ESP32-C6-Zero With pre-soldered header



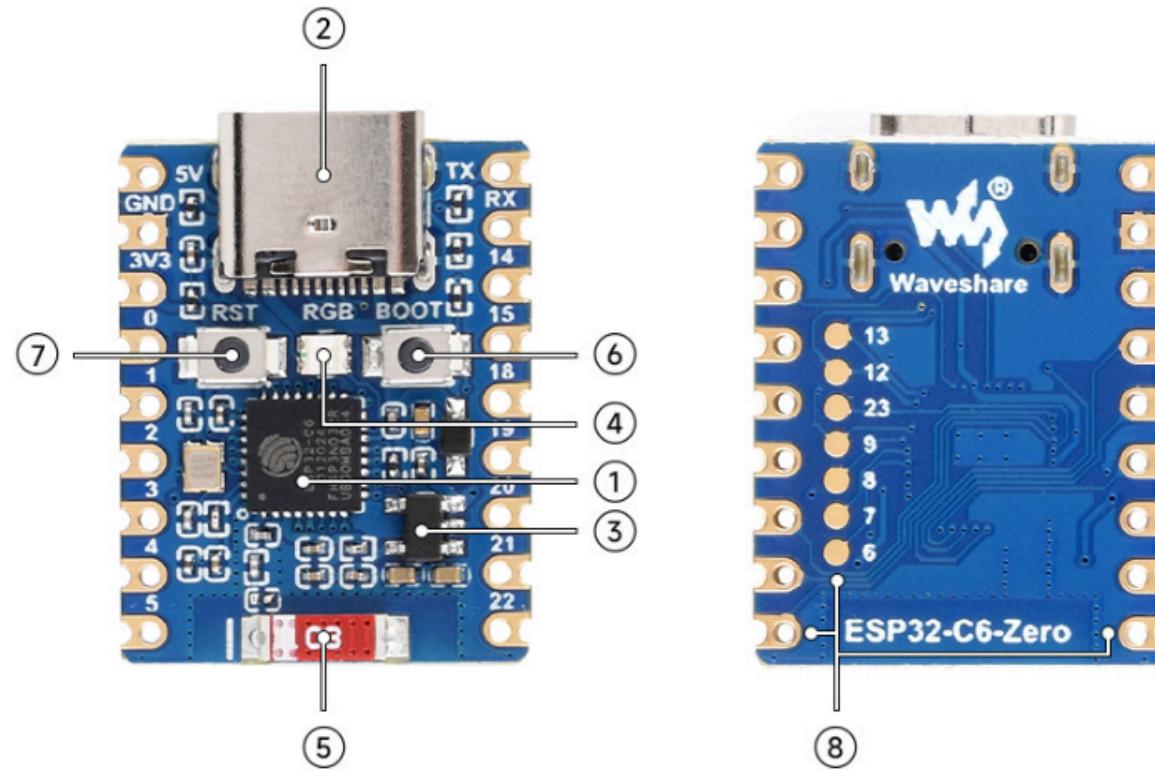
(<https://www.waveshare.com/esp32-c6-zero.htm?sku=26976>)

ESP32-C6, USB Type-C



(/wiki/File:ESP32-C6-Zero-Diagram.png)

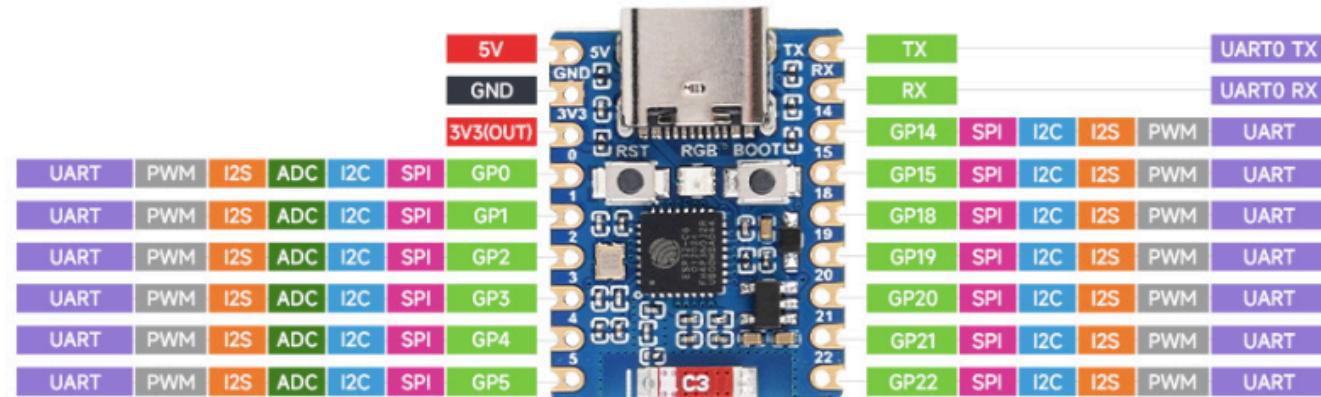
Onboard Interfaces



1. **ESP32-C6FH4**
Built-in dual processors, main frequency is up to 160 MHz
2. **USB Type-C Port**
for downloading programme and debugging
3. **ME6217C33M5G**
low dropout LDO, 800mA (Max)
4. **WS2812 RGB LED**
5. **2.4G ceramic antenna**
6. **BOOT button**
Press it and then press the RESET button to enter download mode
7. **RESET button**
8. **ESP32-C6FH4 pins**

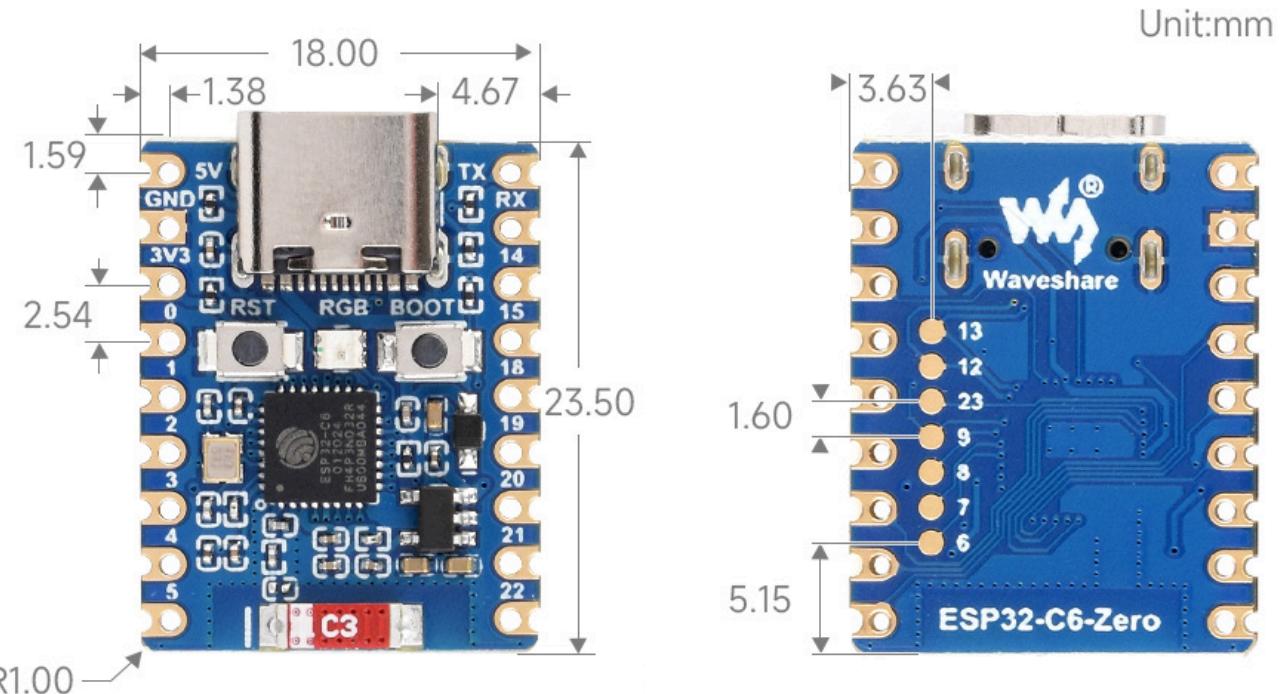
(/wiki/File:ESP32-C6-Zero_Inter.png)

Pinout



(/wiki/File:ESP32-C6-Zero_Pin.png)

Dimensions



(/wiki/File:ESP32-C6-Zero-Diagram-2.png)

Working with ESP-IDF

The following development system defaults to Windows, and it is recommended to use

the VSCode plug-in for development.

Develop with VSCode

Install VSCode

- Open the VSCode website (<https://code.visualstudio.com/download>) to download according to the corresponding system and system bits.

Download Visual Studio Code

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



↓ Windows

Windows 10, 11

User Installer	x64	x86	Arm64
System Installer	x64	x86	Arm64
.zip	x64	x86	Arm64
CLI	x64	x86	Arm64

↓ .deb

Debian, Ubuntu

↓ .rpm

Red Hat, Fedora, SUSE

.deb	x64	Arm32	Arm64
.rpm	x64	Arm32	Arm64
.tar.gz	x64	Arm32	Arm64
Snap	Snap Store		
CLI	x64 Arm32 Arm64		

↓ Mac

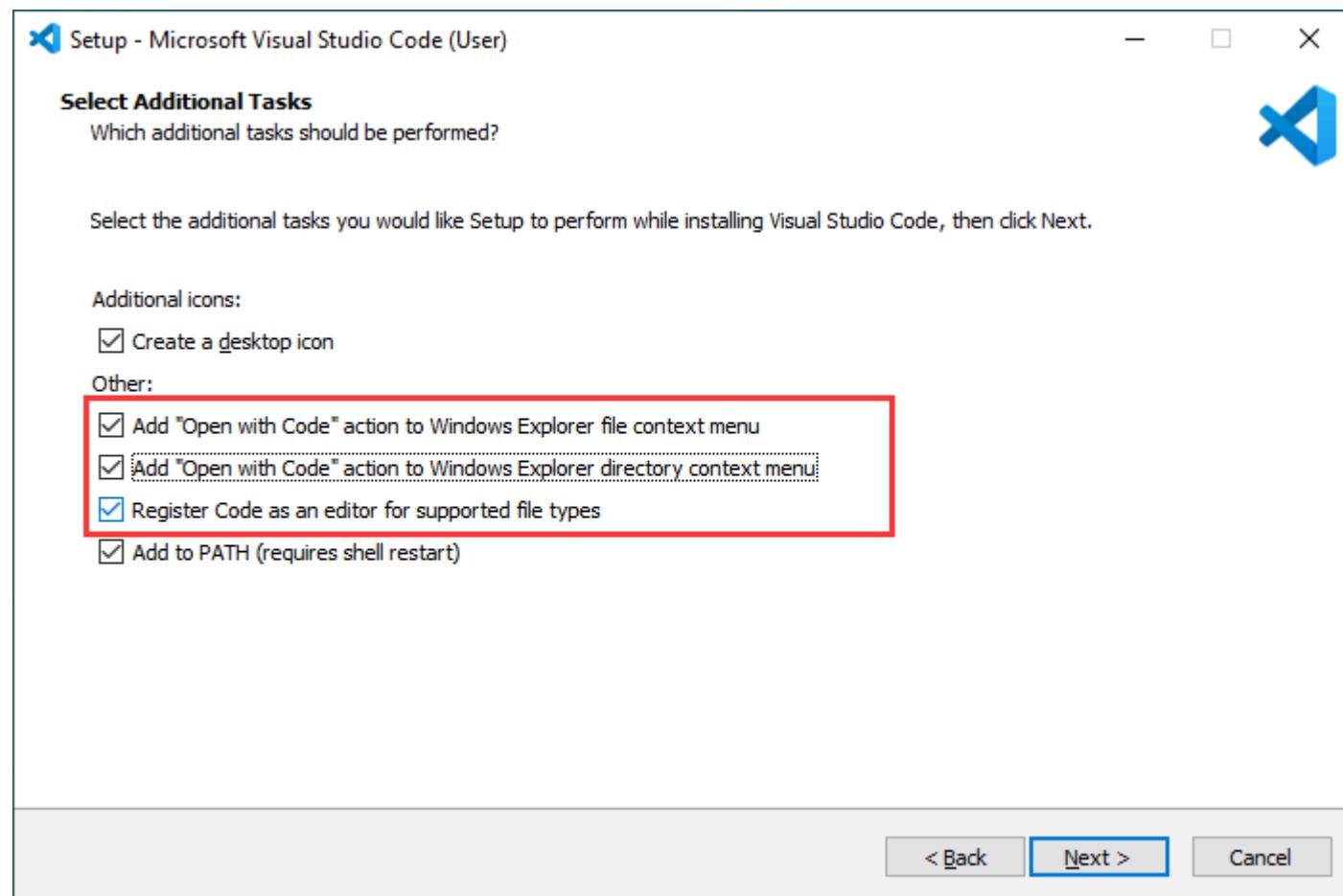
macOS 10.11+

.zip	Intel chip	Apple silicon	Universal
CLI	Intel chip Apple silicon		

By downloading and using Visual Studio Code, you agree to the [license terms](#) and [privacy statement](#).

(/wiki/File:ESP32-C6-DEV-KIT-N8-01.png)

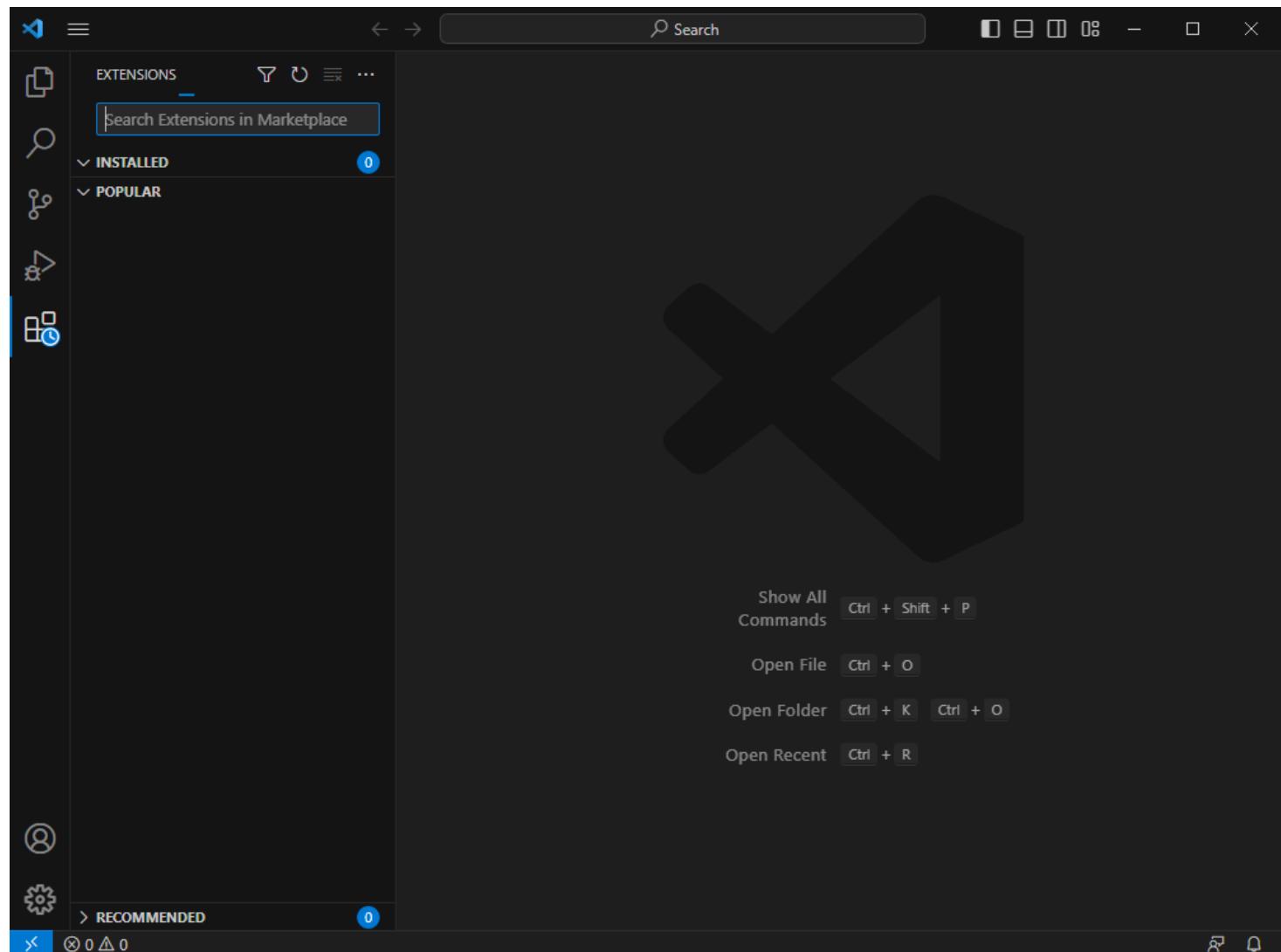
- After running the installation package, the rest can be installed by default, but here for a better experience, it is recommended to check the box in the 1, 2 and 3 items.
 - After enabling the 1st and 2nd items, you can directly open the VScode by right-clicking the file or the directory to improve your experience.
 - After enabling the 3nd items, you can directly select VSCode when choosing how to open.



(/wiki/File:ESP32-C6-DEV-KIT-N8-02.png)

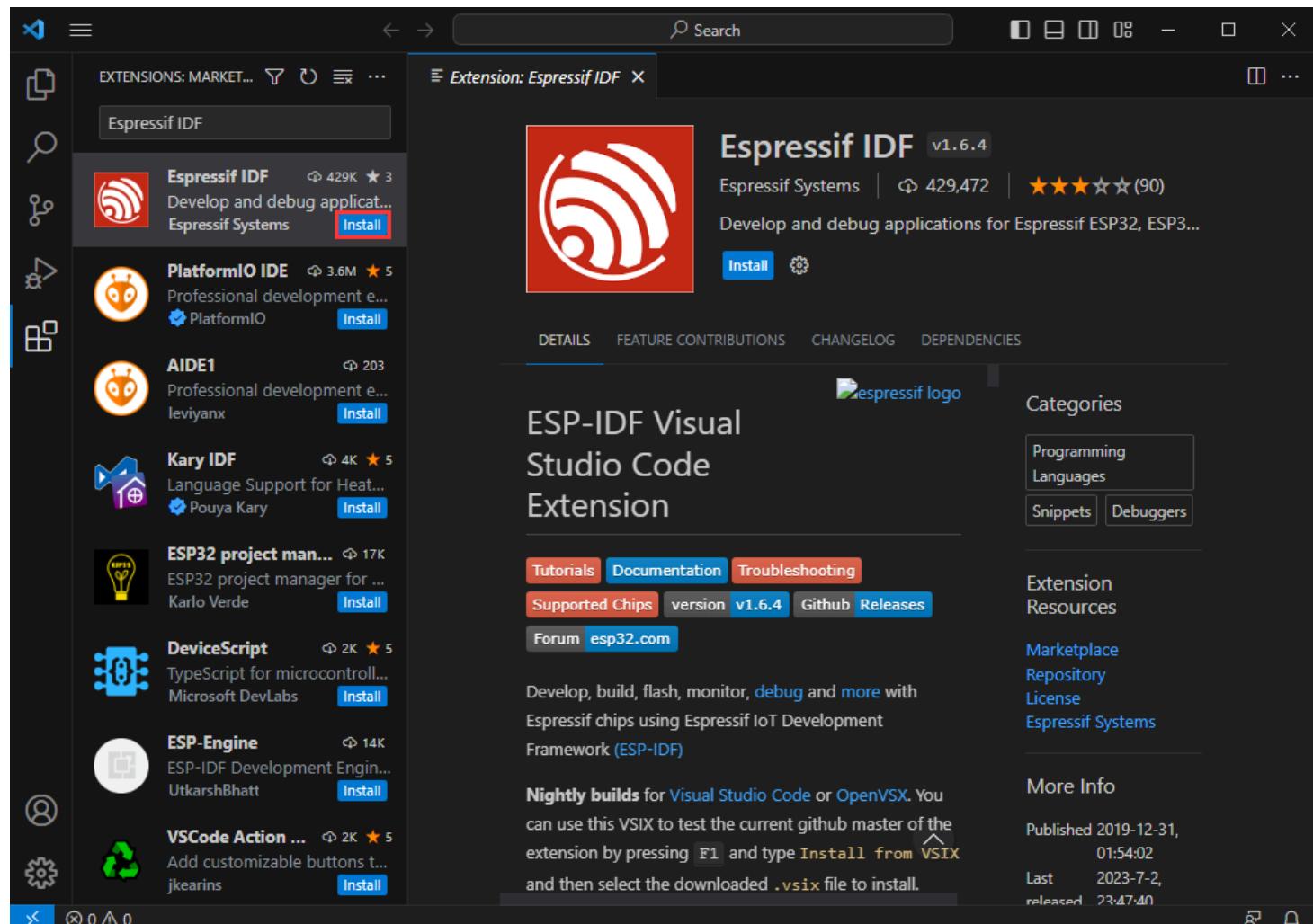
Install Espressif IDF Plug-in

- Note: Currently the latest version of the plugin is V1.6.4, users can choose the same version as us for a consistent experience!
- Open VSCode, use Shift+Ctrl+X to enter the plug-in manager.



(/wiki/File:ESP32-C6-DEV-KIT-N8-03.png)

- In the search bar, enter **Espressif IDF** to select the corresponding plug-in and click "Install".

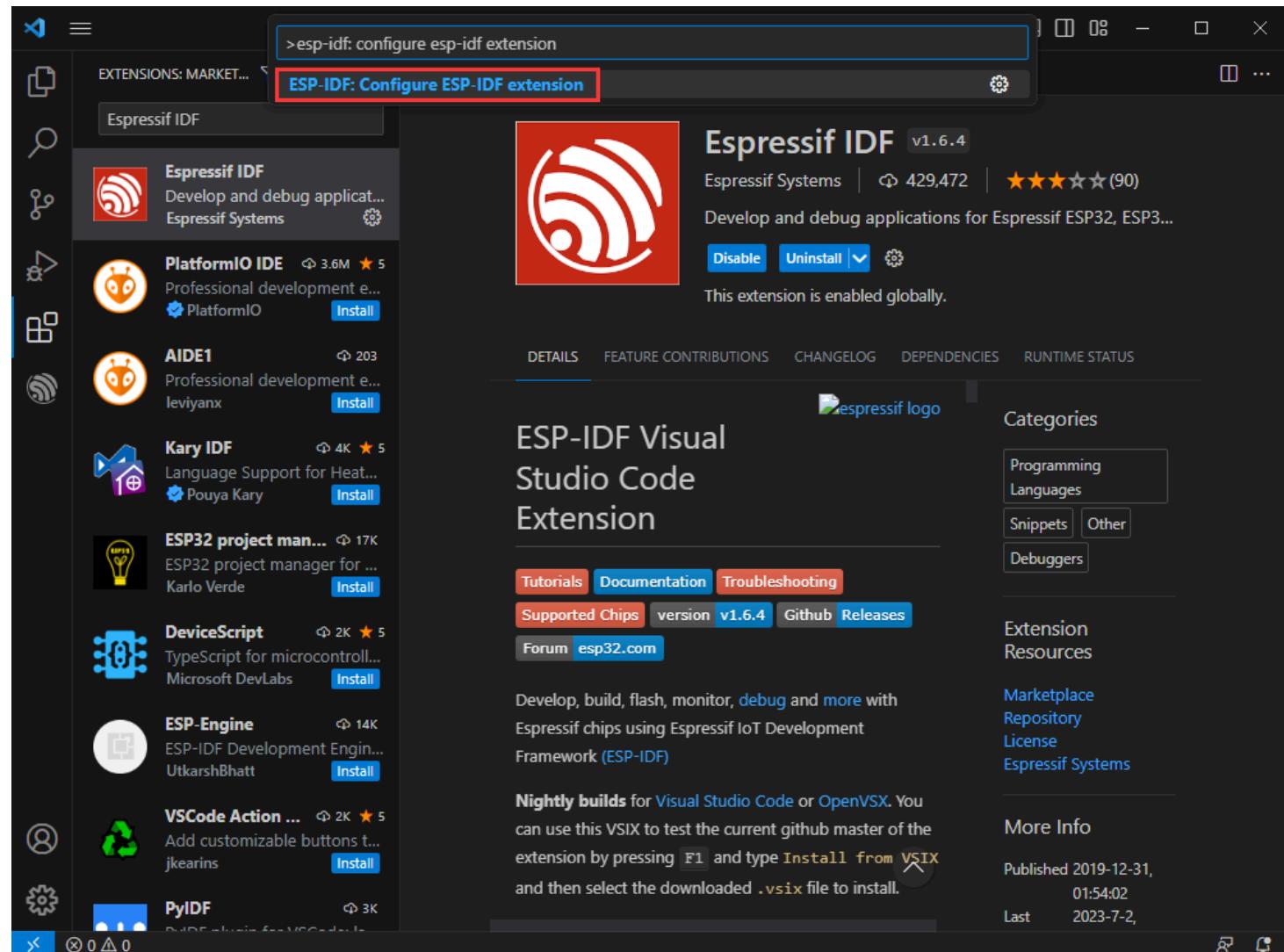


(/wiki/File:ESP32-C6-DEV-KIT-N8-04.png)

(/wiki/File:ESP32-C6-DEV-KIT-N8-05.png)

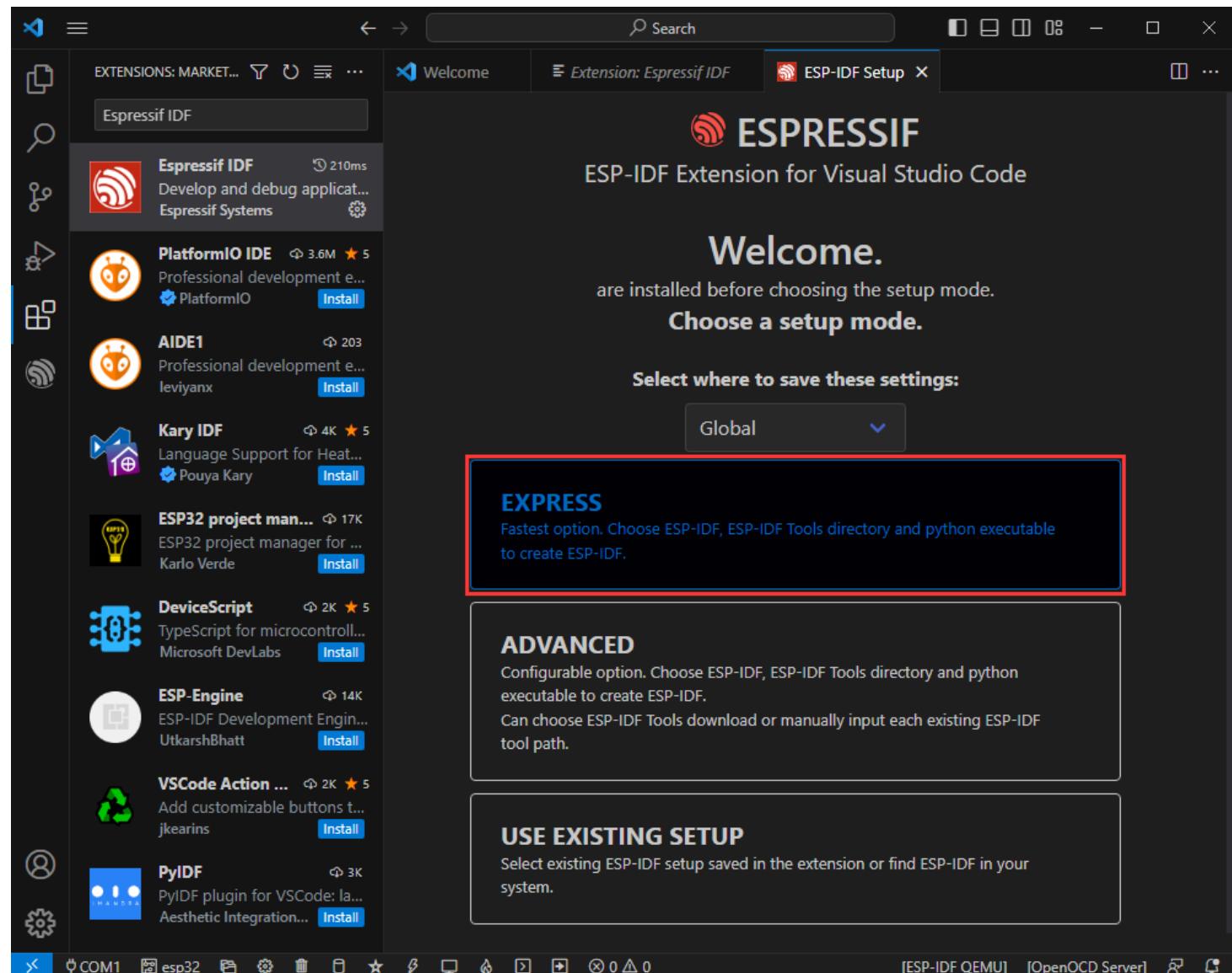
- Press **F1** to input:

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



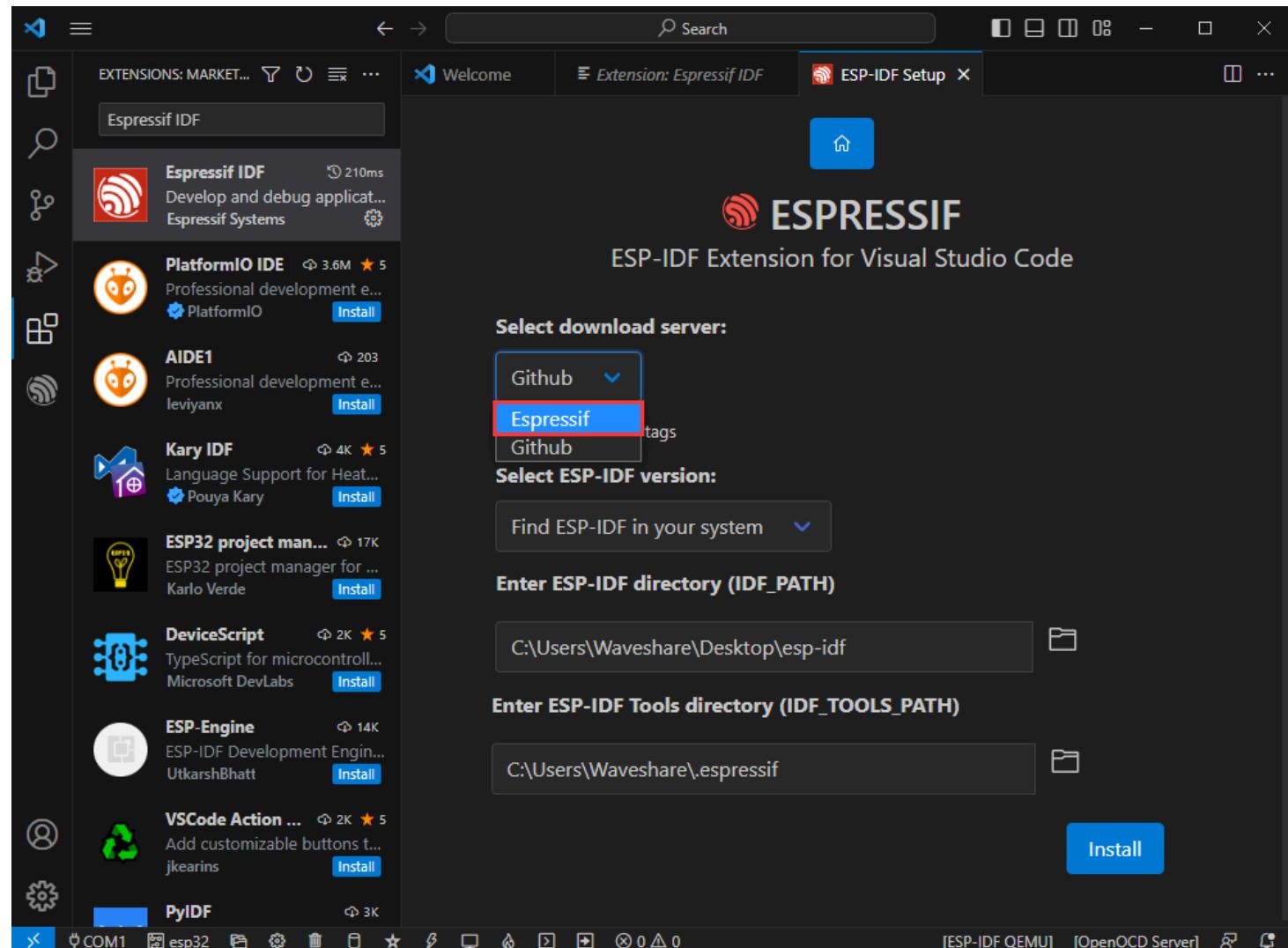
(/wiki/File:ESP32-C6-DEV-KIT-N8-06.png)

- Select express (this guide is for users who install it for the first time).



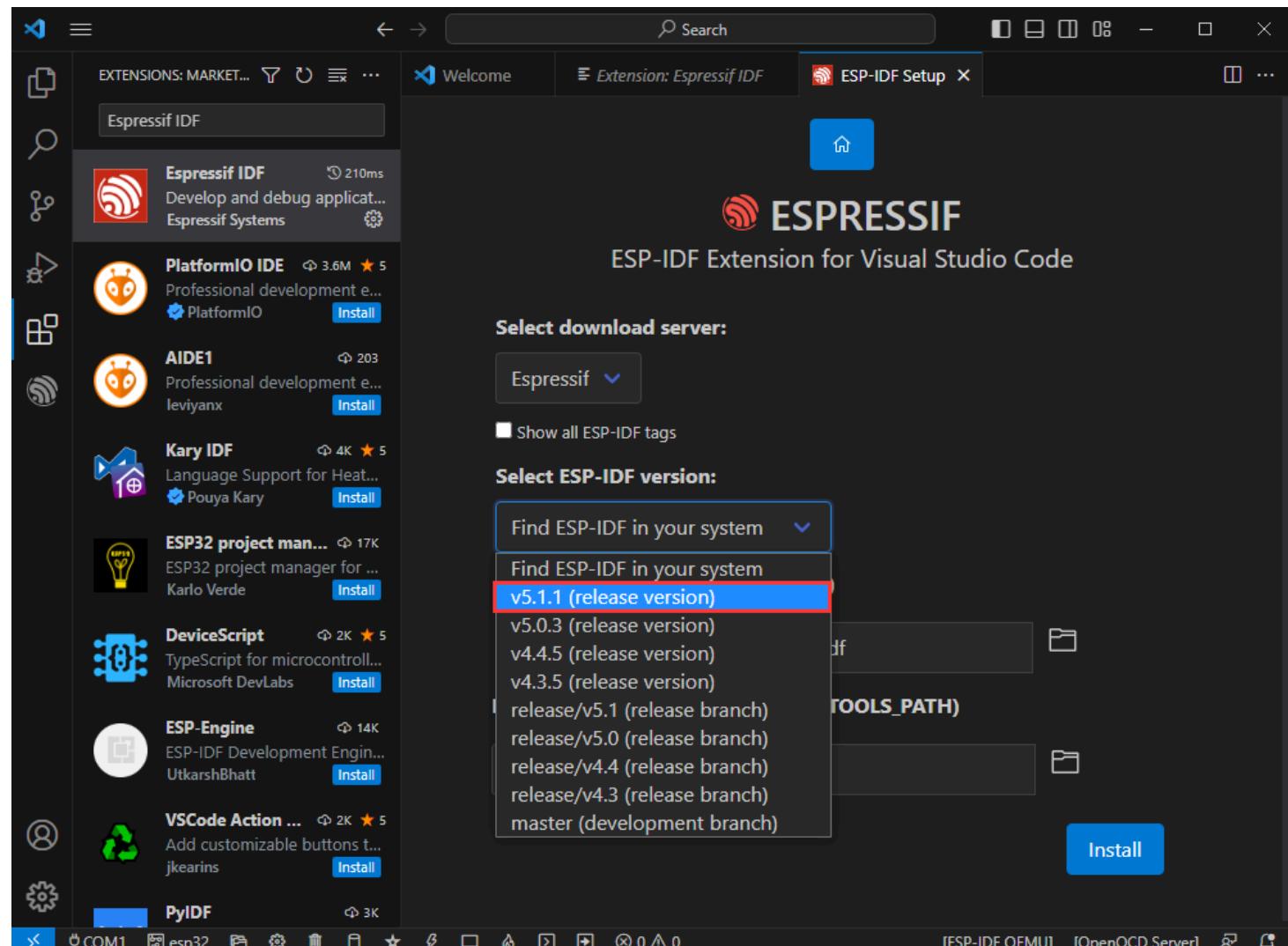
(/wiki/File:ESP32-C6-DEV-KIT-N8-07.png)

- Select download sever.



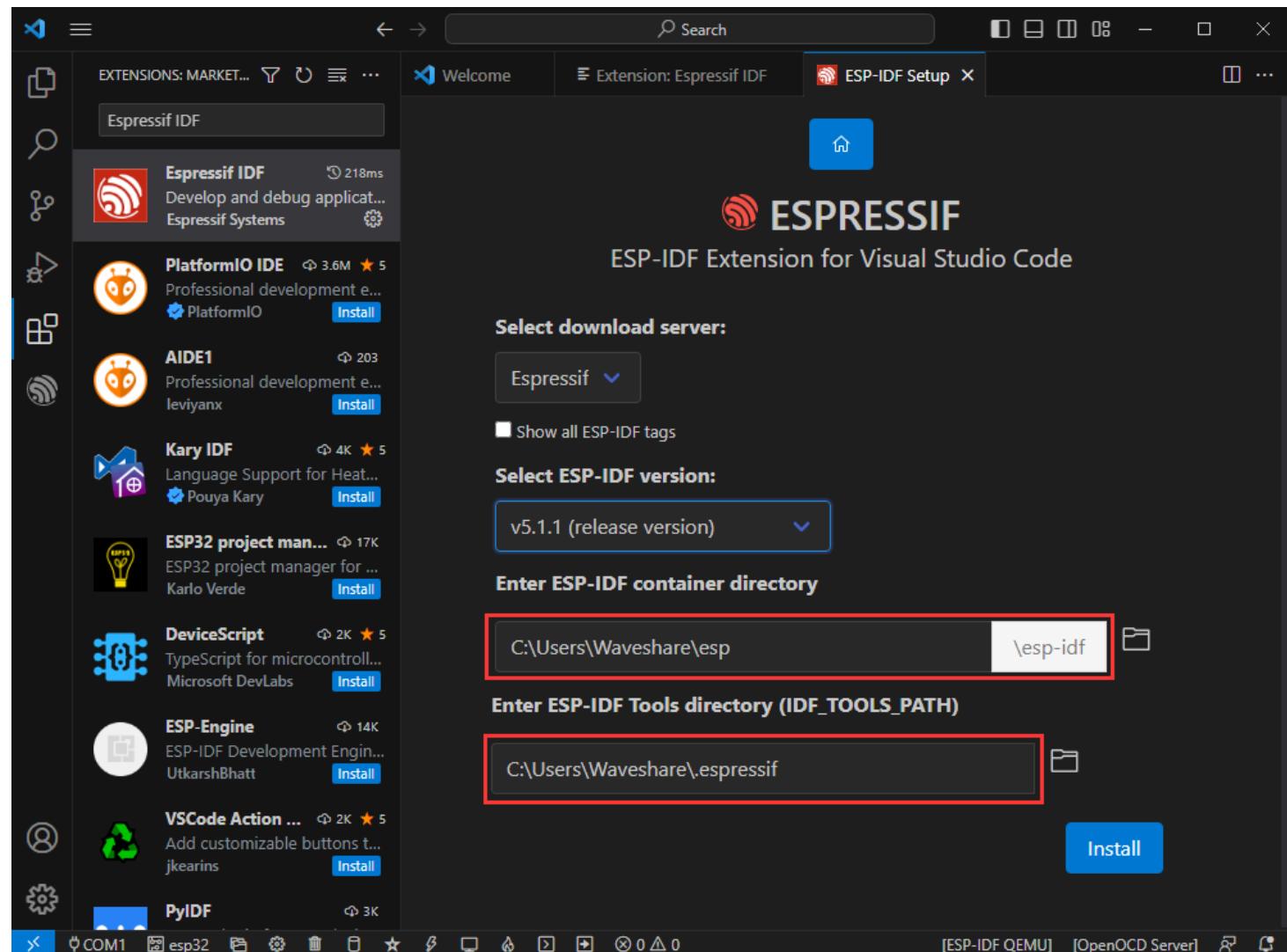
(/wiki/File:ESP32-C6-DEV-KIT-N8-08.png)

- Select the version of ESP-IDF you want to use now, we choose the latest V5.1.1 (note that only after V5.1 did ESP-IDF start to support ESP32-C6).



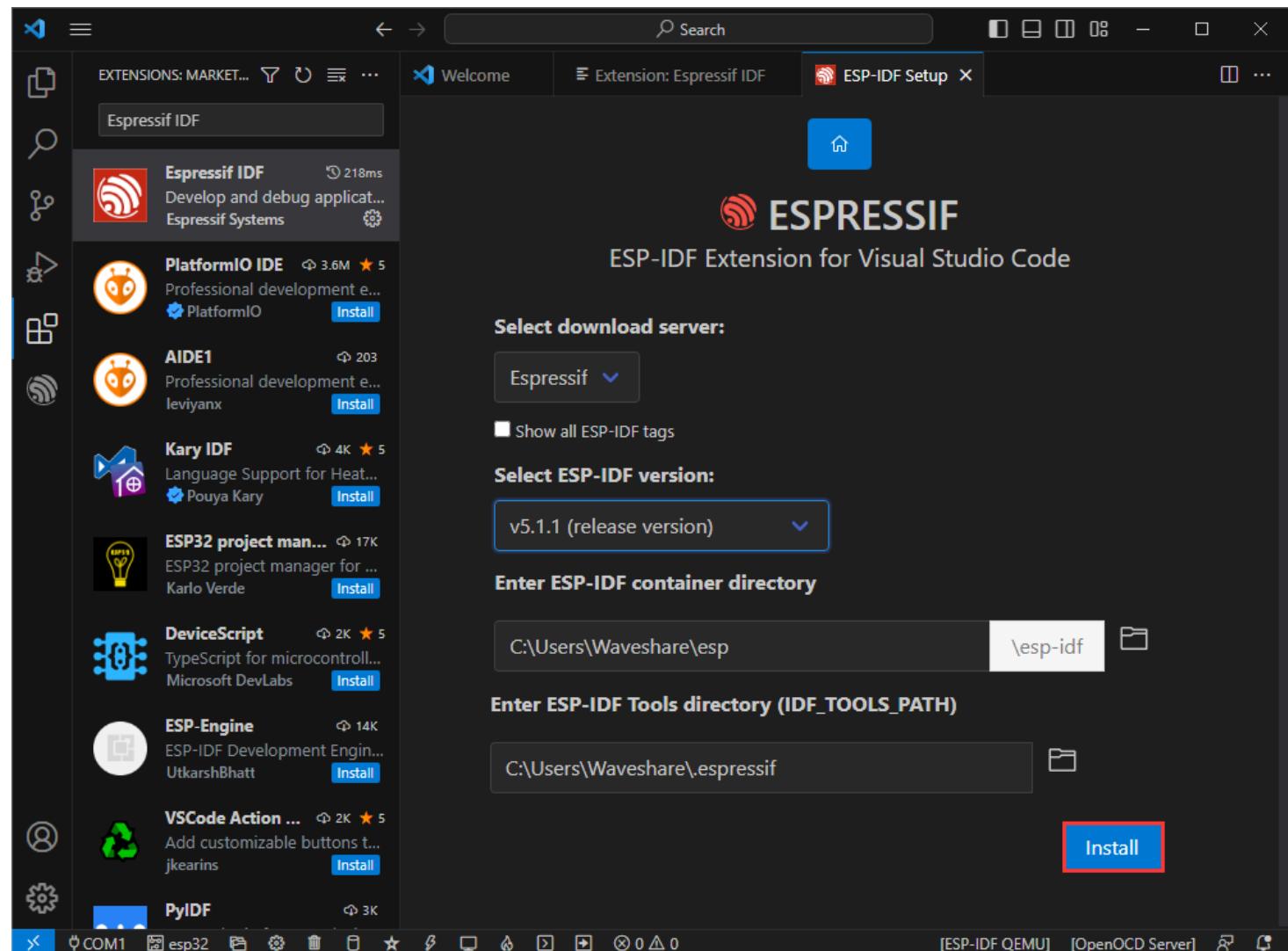
(/wiki/File:ESP32-C6-DEV-KIT-N8-09.png)

- The following two are the installation paths respectively for the ESP-IDF container directory and the ESP-IDF Tools directory.



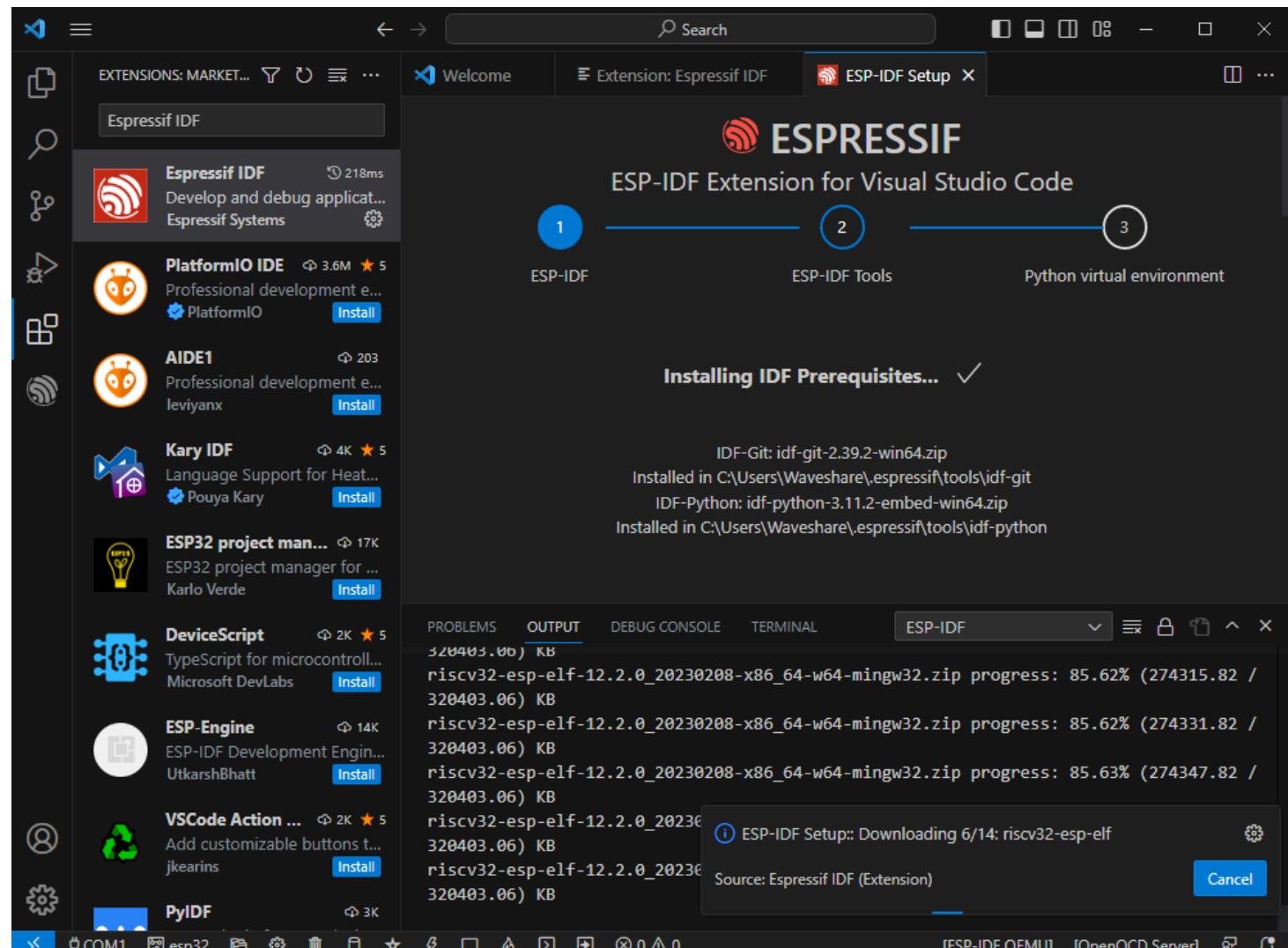
(/wiki/File:ESP32-C6-DEV-KIT-N8-10.png)

- **Note: If you have installed ESP-IDF before, or failed to do so, please be sure to delete the file completely .**
- After configuring, click "Install" to download.



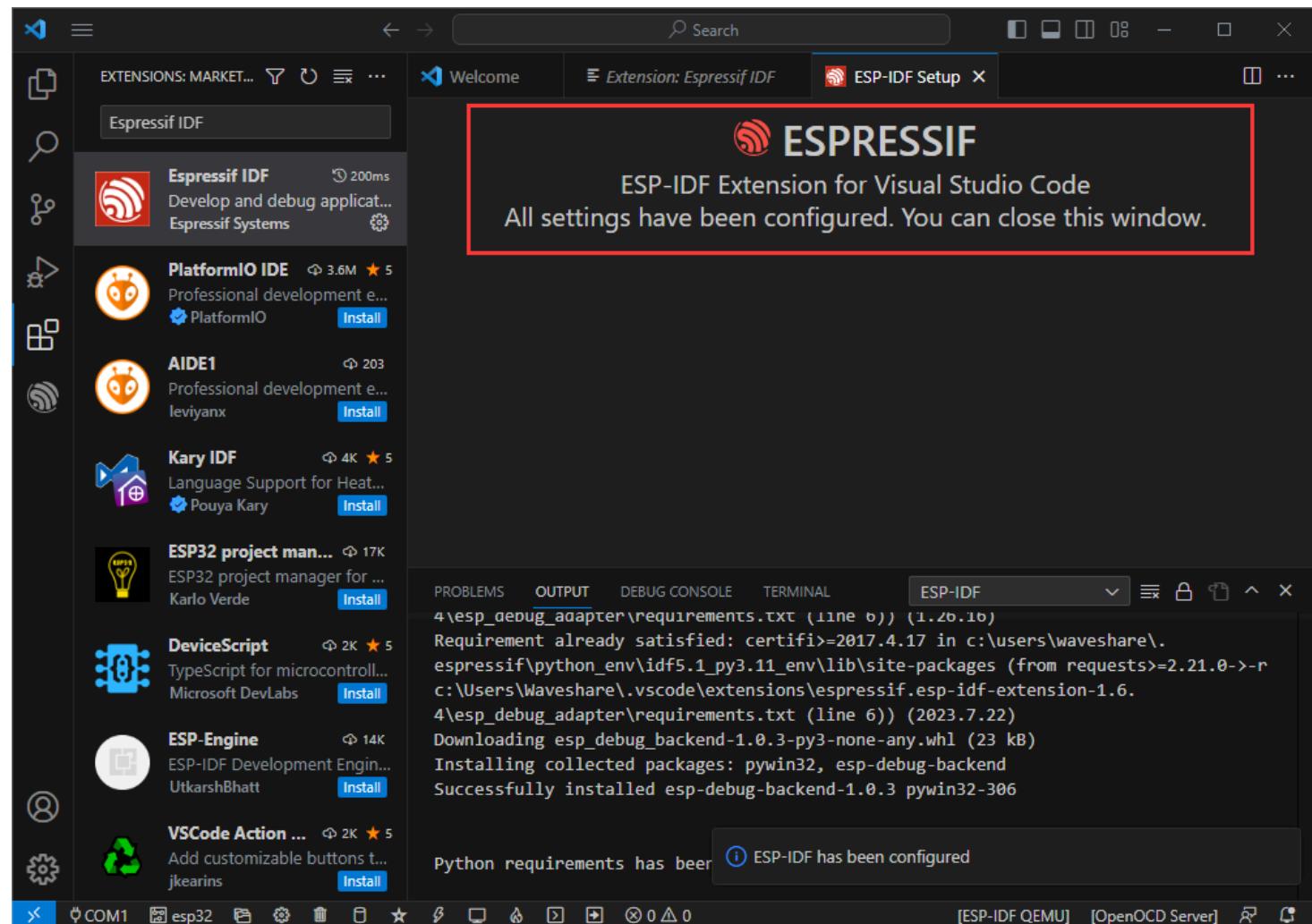
(/wiki/File:ESP32-C6-DEV-KIT-N8-19.png)

- Enter the download interface, and then it will automatically install the corresponding tools and environment, just wait for a second.



(/wiki/File:ESP32-C6-DEV-KIT-N8-11.png)

- After the installation is complete, you will enter the following interface, indicating that the installation is finished.



(/wiki/File:ESP32-C6-DEV-KIT-N8-12.png)

Official Demo Usage GUIDE

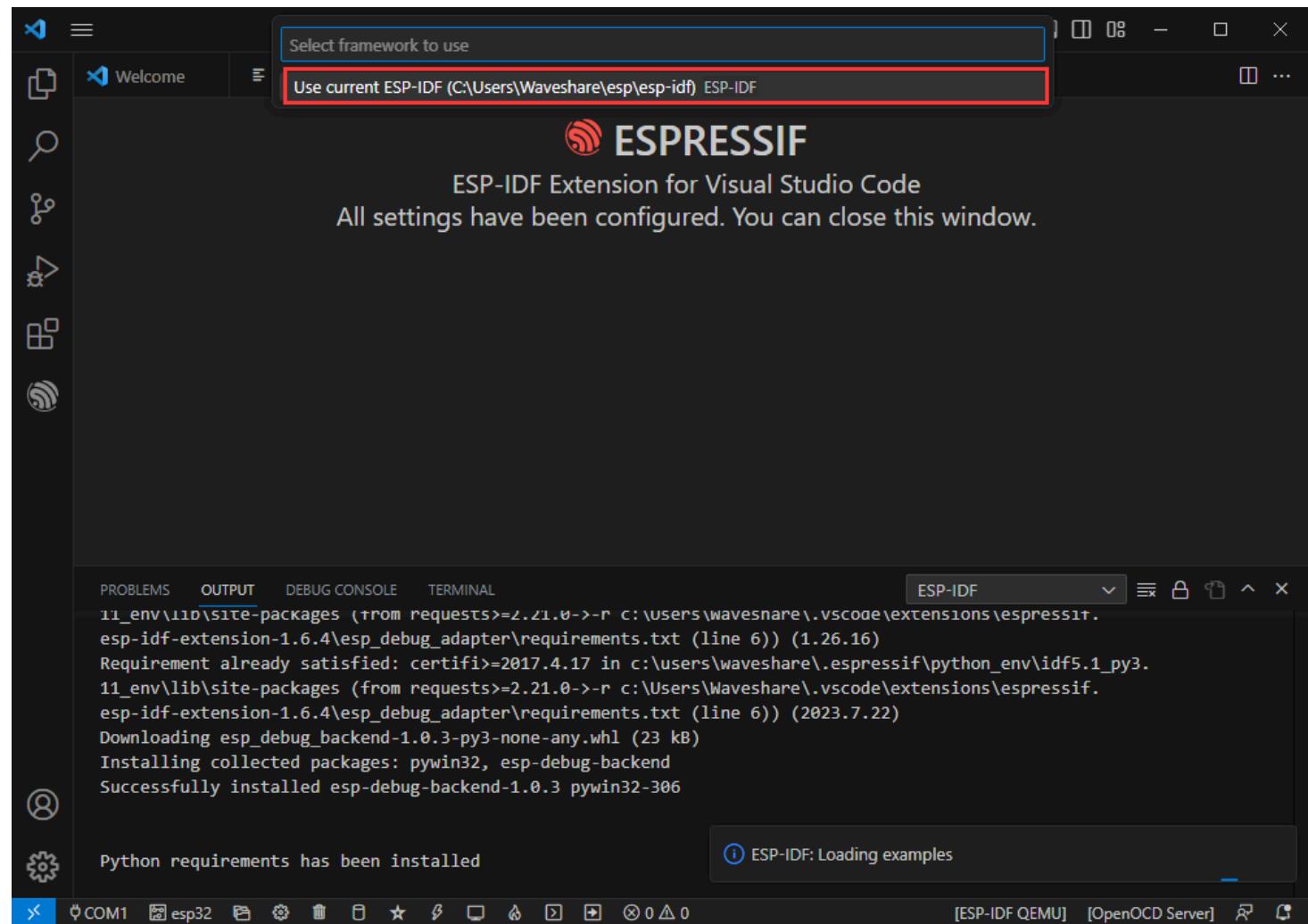
Create Demo (#Demo Example)

- Press **F1** to enter:

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

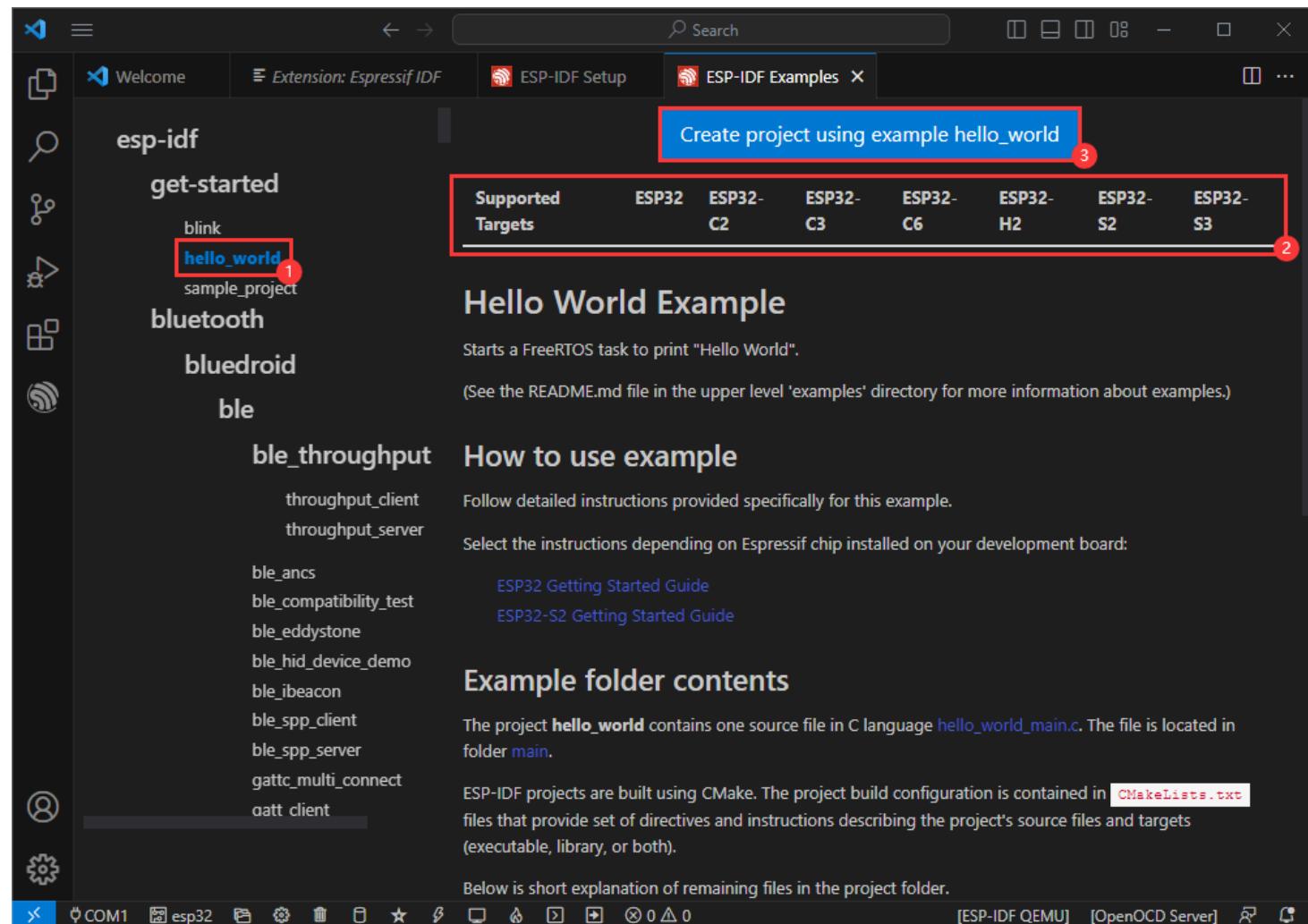
(/wiki/File:ESP32-C6-DEV-KIT-N8-13.png)

- Select the corresponding IDF version.



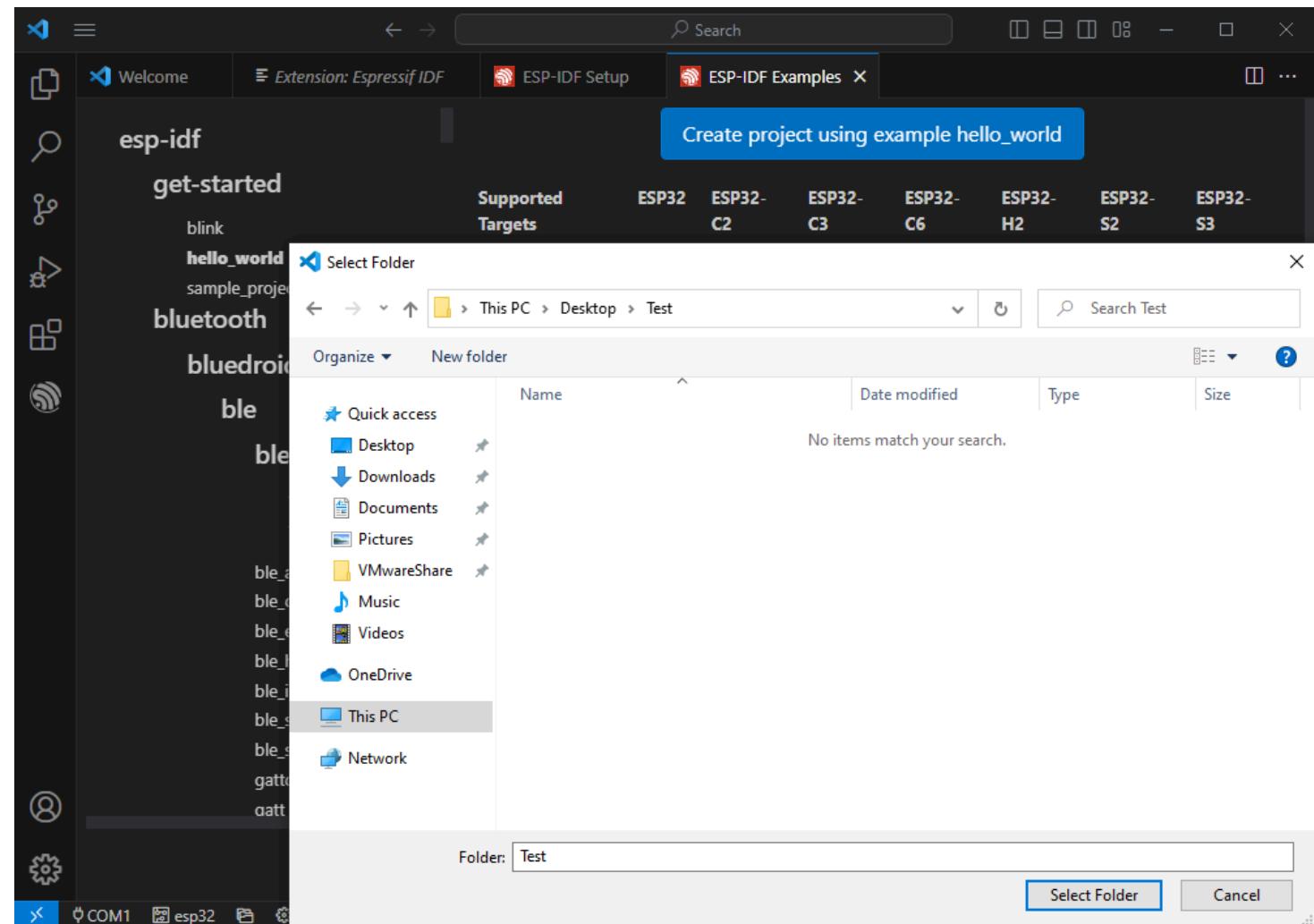
(/wiki/File:ESP32-C6-DEV-KIT-N8-14.png)

- Take the Hello World demo as an example:
 - ①Select the corresponding demo.
 - ②Its readme will state what chip the demo applies to (how the demo is used and the file structure are described below, omitted here).
 - ③Click to create the demo.



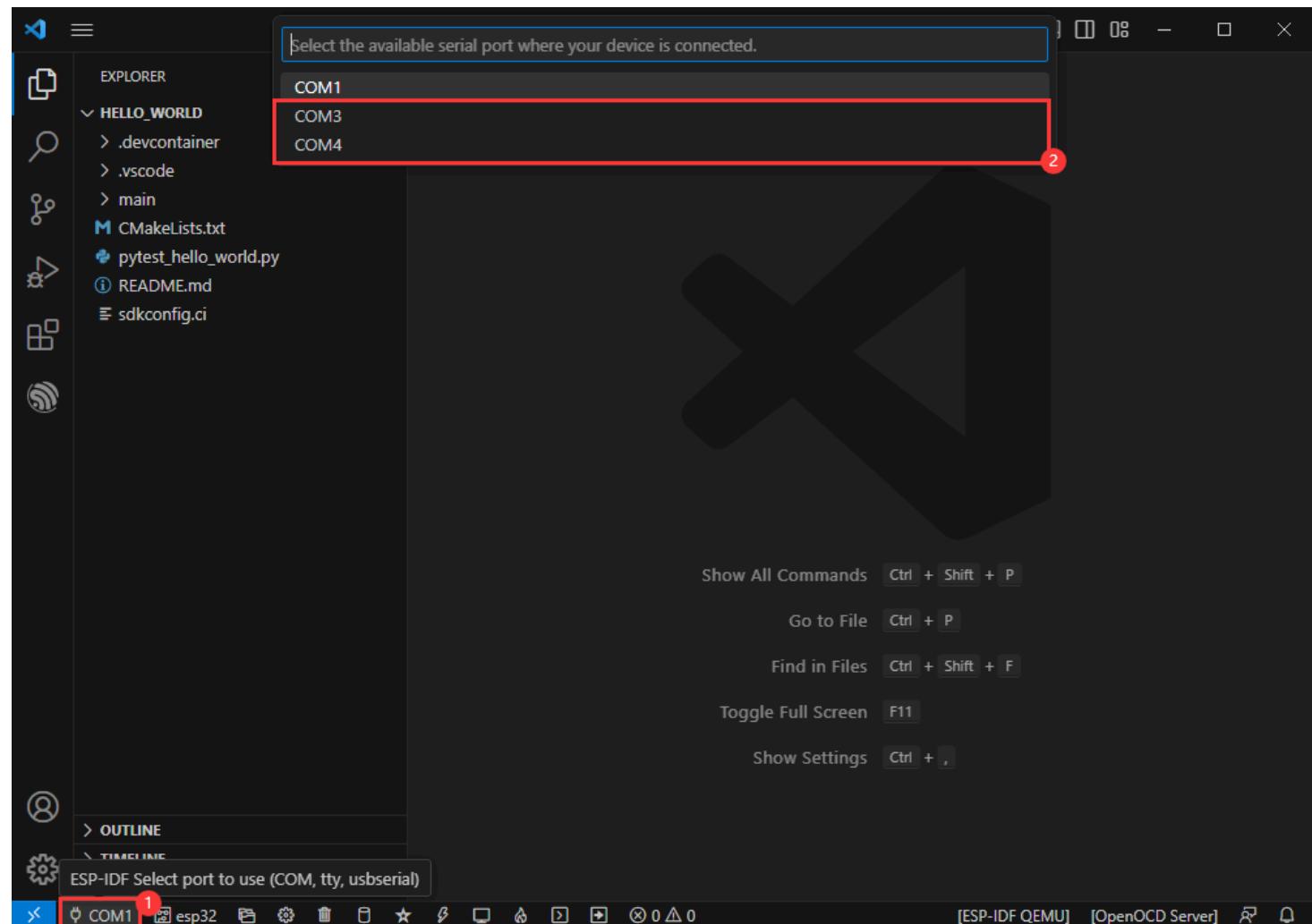
(/wiki/File:ESP32-C6-DEV-KIT-N8-15.png)

Select the path to place the demo, and no folder with the same name as the demo is required.



Modify COM Port

- The corresponding COM ports are shown here, click to modify them.
- Please select the COM ports according to your device. **It is recommended to prioritize the use of the COM port corresponding to USB (can be viewed through the device manager).**
- In case of a download failure, please press the reset button for more than 1 second and wait for the PC to recognize the device again before downloading once more.



(/wiki/File:ESP32-C6-DEV-KIT-N8-17.png)

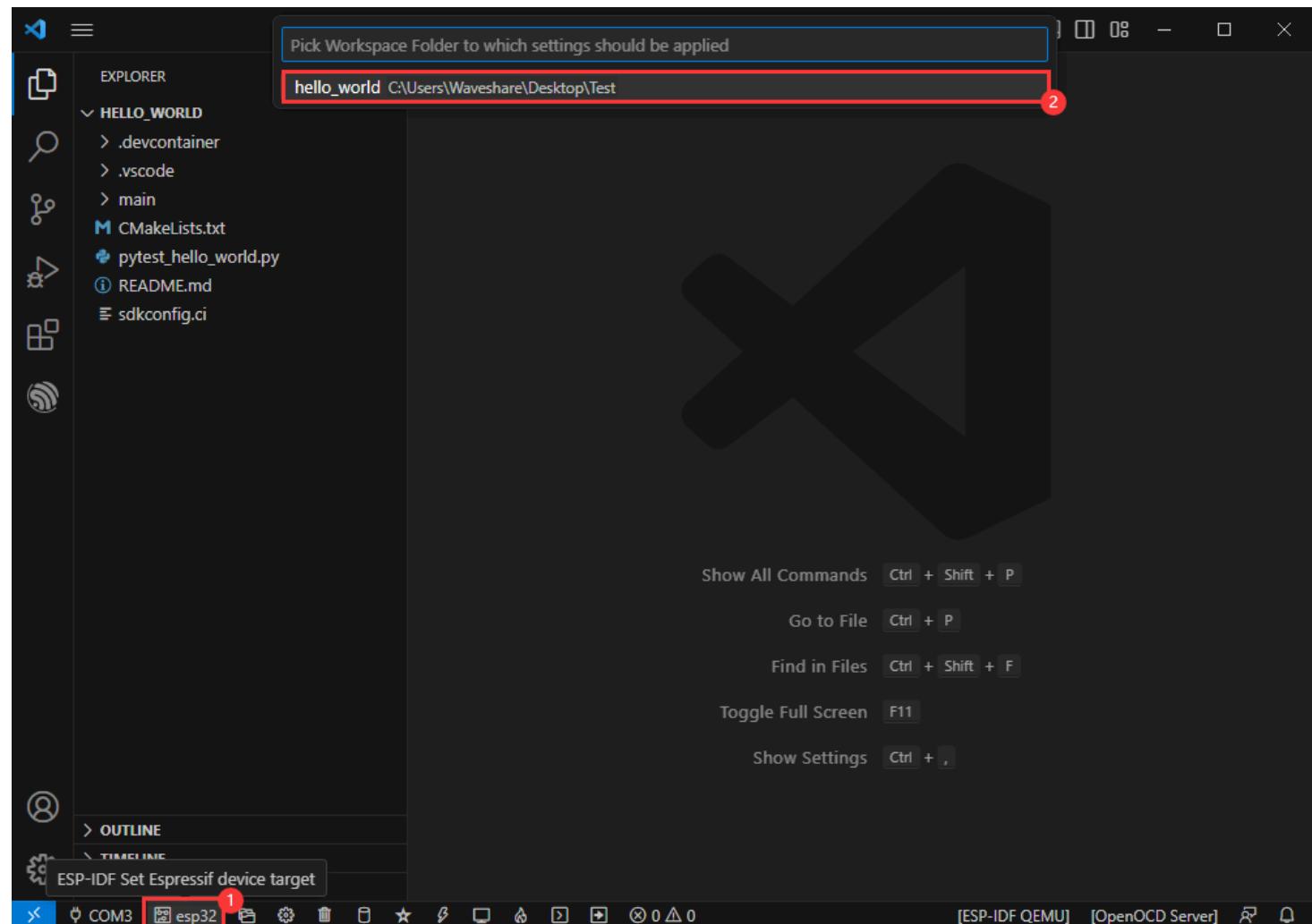
- Select the project or demo to use.

(/wiki/File:ESP32-C6-DEV-KIT-N8-18.png)

- Then we finish the modification of the COM ports.

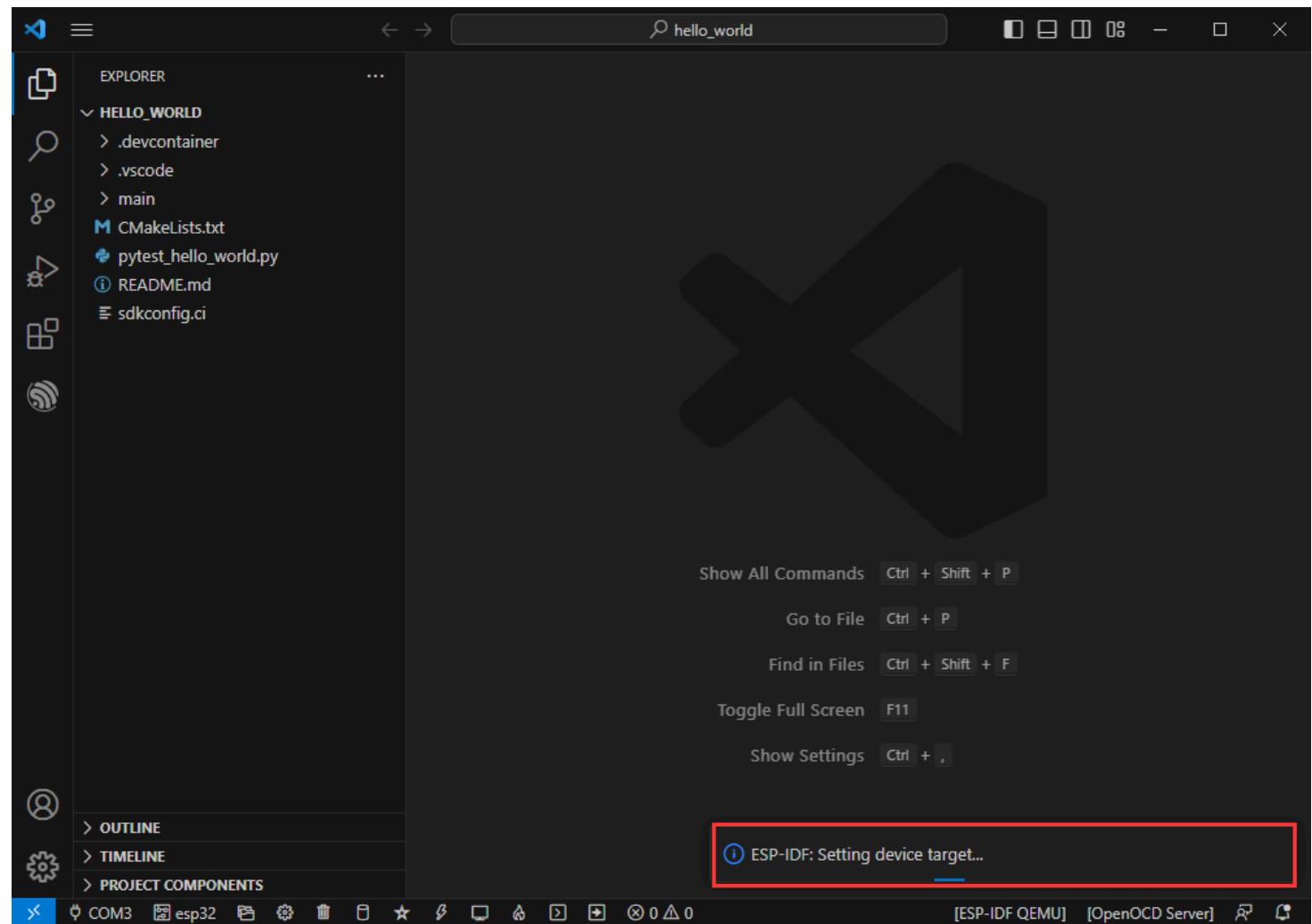
Modify the Driver Object

- The driver object is displayed here, and you can modify it by clicking on it.
- Select the project or demo to use.



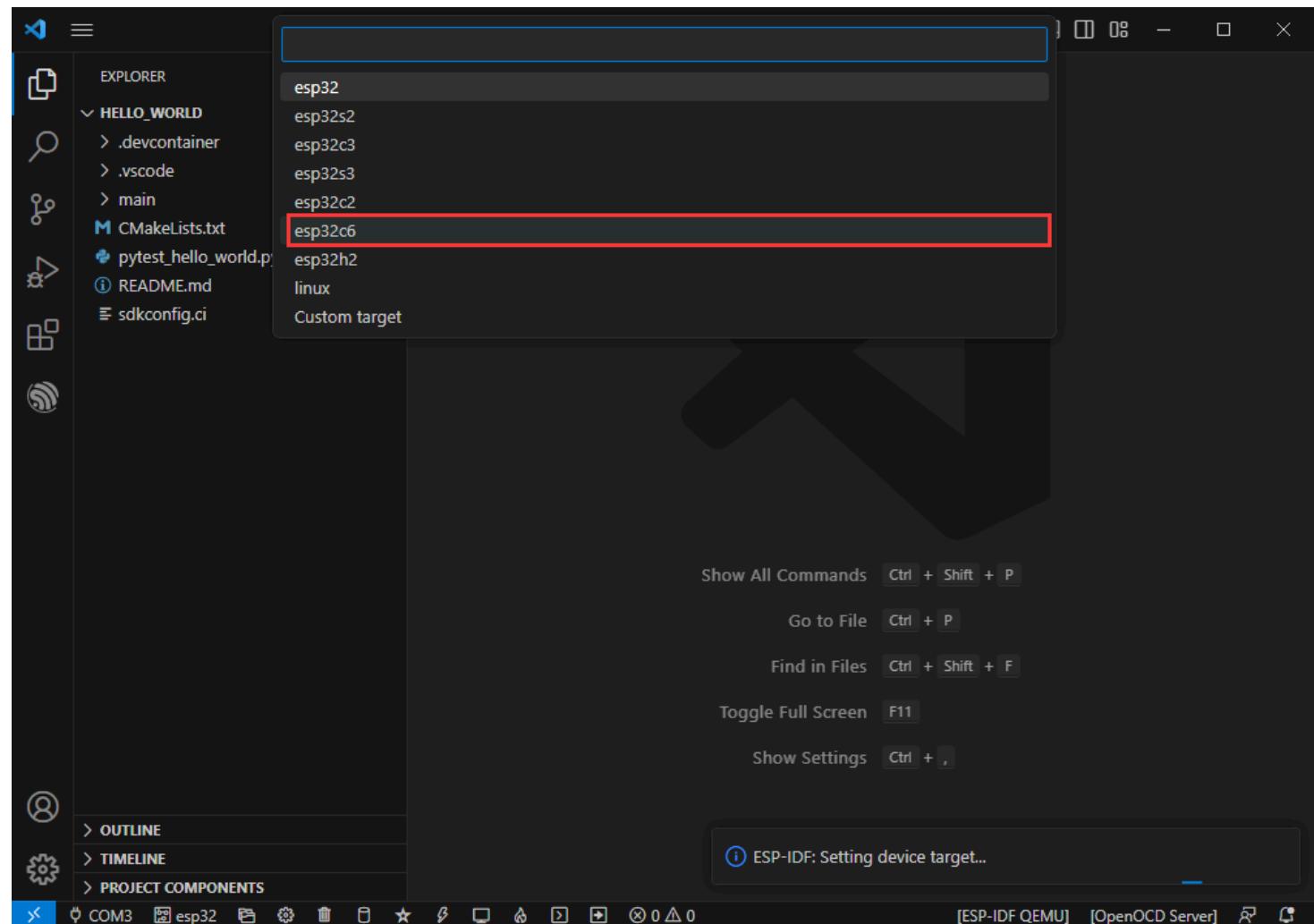
(/wiki/File:ESP32-C6-DEV-KIT-N8-20.png)

- Wait for a minute after clicking.



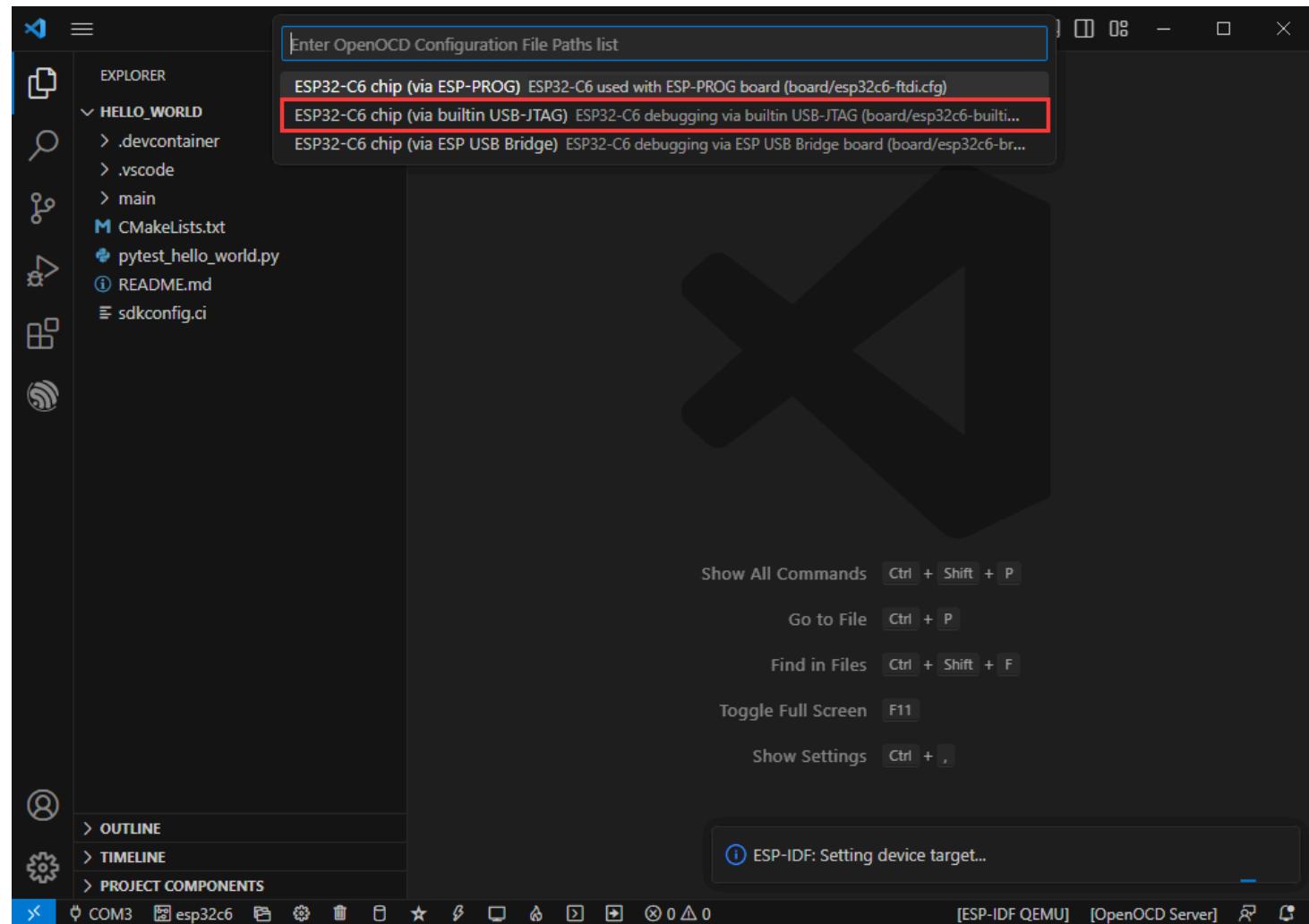
(/wiki/File:ESP32-C6-DEV-KIT-N8-21.png)

- Select the object we need to drive, which is our main chip ESP32C6.



(/wiki/File:ESP32-C6-DEV-KIT-N8-22.png)

- Choose the path to openocd, it doesn't affect us here, so let's just choose one at random.

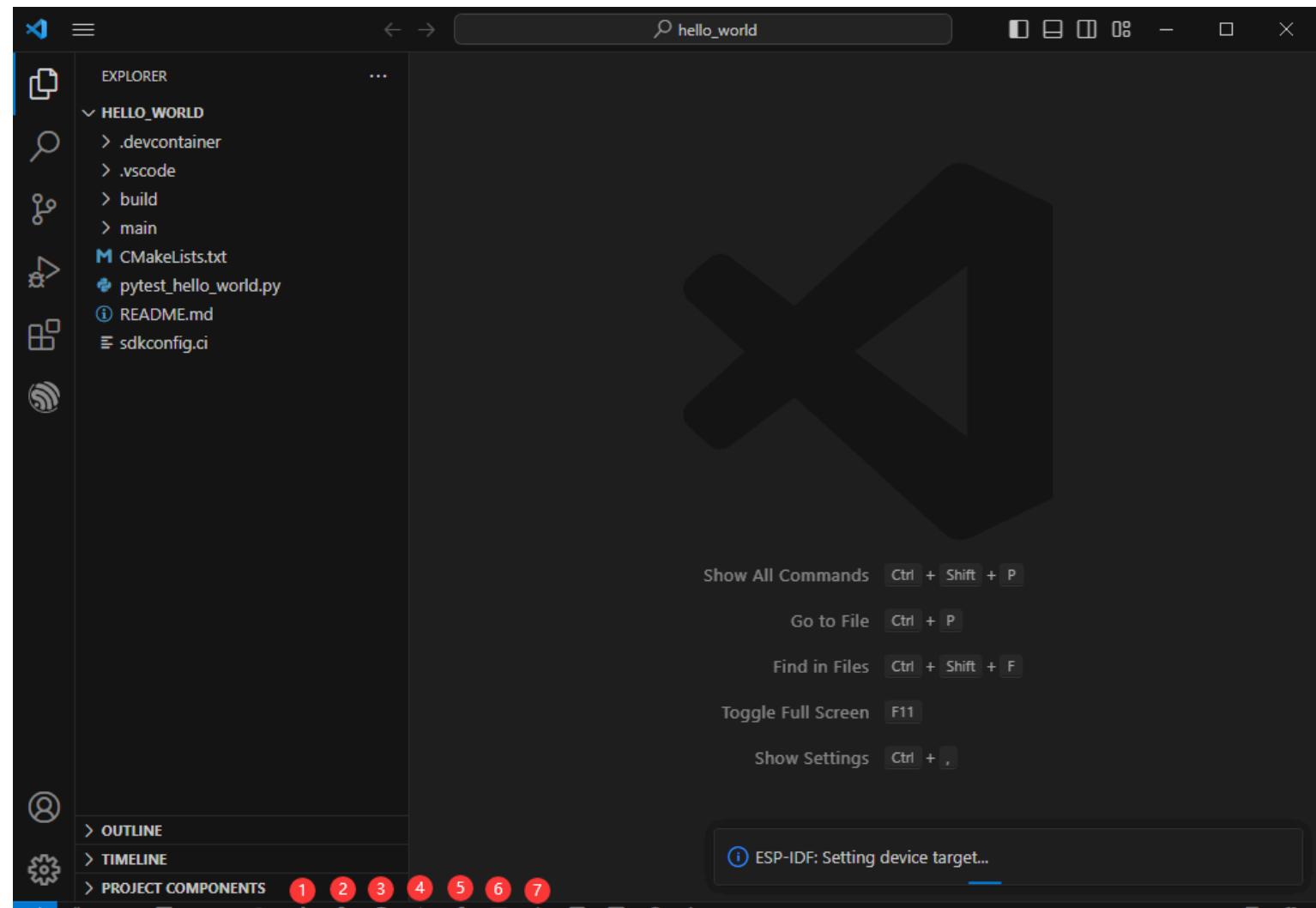


(/wiki/File:ESP32-C6-DEV-KIT-N8-23.png)

The Rest of the Status Bar

- ①SDK configuration editor, supports modifying most functions of ESP-IDF.
- ②All cleanup, and clear all compiled files.
- ③Compile.
- ④Current download mode, default is UART.
- ⑤Flash the current firmware, please do it after compiling.

- ⑥Open the serial port monitor, which is used to view the serial port information.
- ⑦All-in-one button, compile, flash and open the serial monitor (most commonly used for debugging).



(/wiki/File:ESP32-C6-DEV-KIT-N8-24.png)

Compile, Program, Serial Port Monitoring

- Click on the all-in-one button we described before to compile, program and open the serial port monitor.

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 and highlighted in blue.
- Code Editor:** Displays the content of "hello_world_main.c". The code includes standard library includes, FreeRTOS headers, and a main function that prints "Hello world!\n" and then prints chip information.
- Bottom Status Bar:** Shows connection to "COM3", board "esp32c6", build "Release", and other status indicators.

```
/* SPDX-FileCopyrightText: 2010-2022 Espressif Systems (Shanghai) CO LTD
 * SPDX-License-Identifier: CC0-1.0
 */
#include <stdio.h>
#include <inttypes.h>
#include "sdkconfig.h"
#include "freertos/FreeRTOS.h"
#include "freertos/task.h"
#include "esp_chip_info.h"
#include "esp_flash.h"

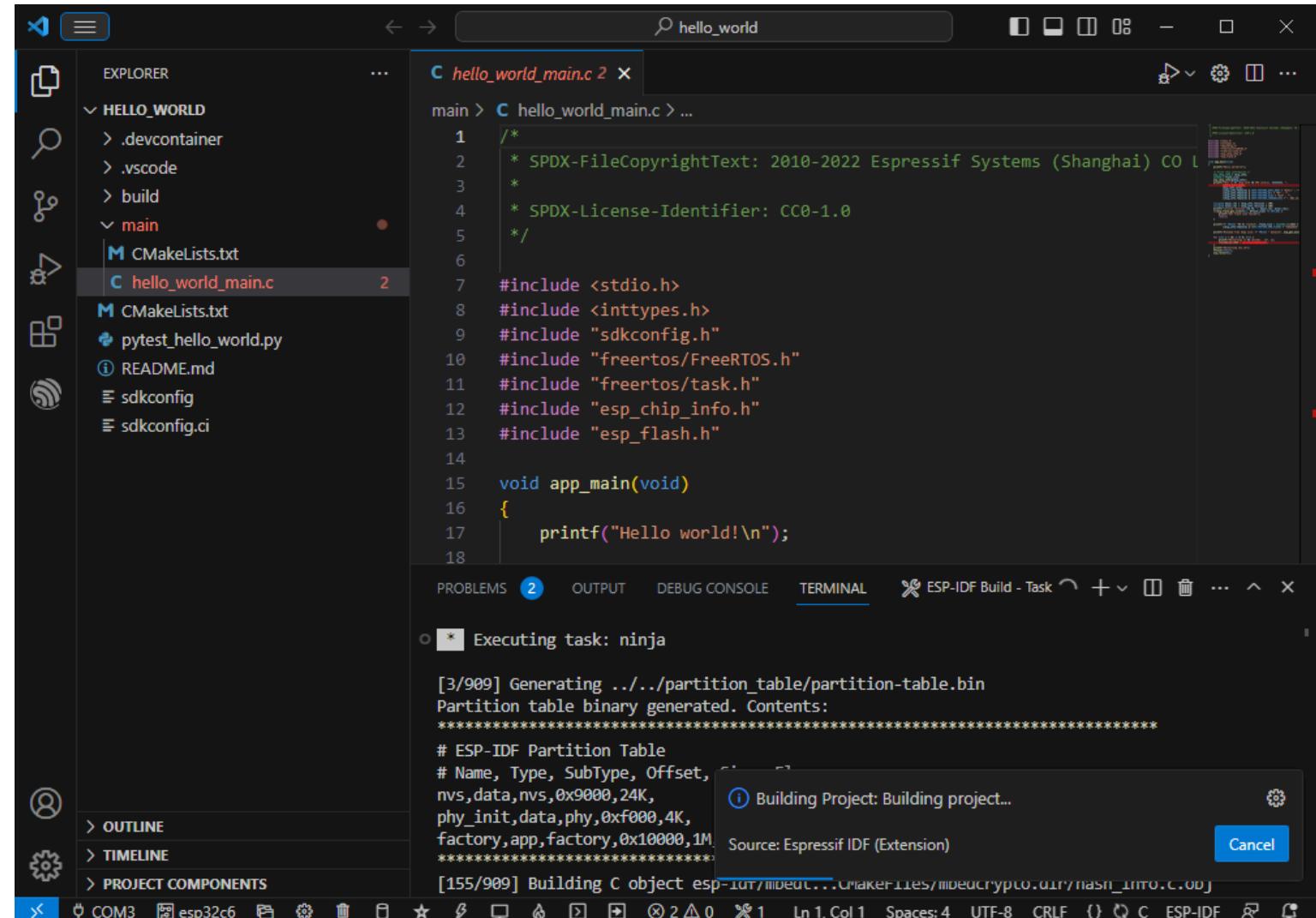
void app_main(void)
{
    printf("Hello world!\n");

    /* Print chip information */
    esp_chip_info_t chip_info;
    uint32_t flash_size;
    esp_chip_info(&chip_info);
    printf("This is %s chip with %d CPU core(s), %s%s%s%s, %s\n",
        CONFIG_IDF_TARGET,
        chip_info.cores,
        (chip_info.features & CHIP_FEATURE_WIFI_BGN) ? "WiFi/" : "",
        (chip_info.features & CHIP_FEATURE_BT) ? "BT" : "",
        (chip_info.features & CHIP_FEATURE_BLE) ? "BLE" : "",
        (chip_info.features & CHIP_FEATURE_IEEE802154) ? ", 802.15.",

    unsigned major_rev = chip_info.revision / 100;
```

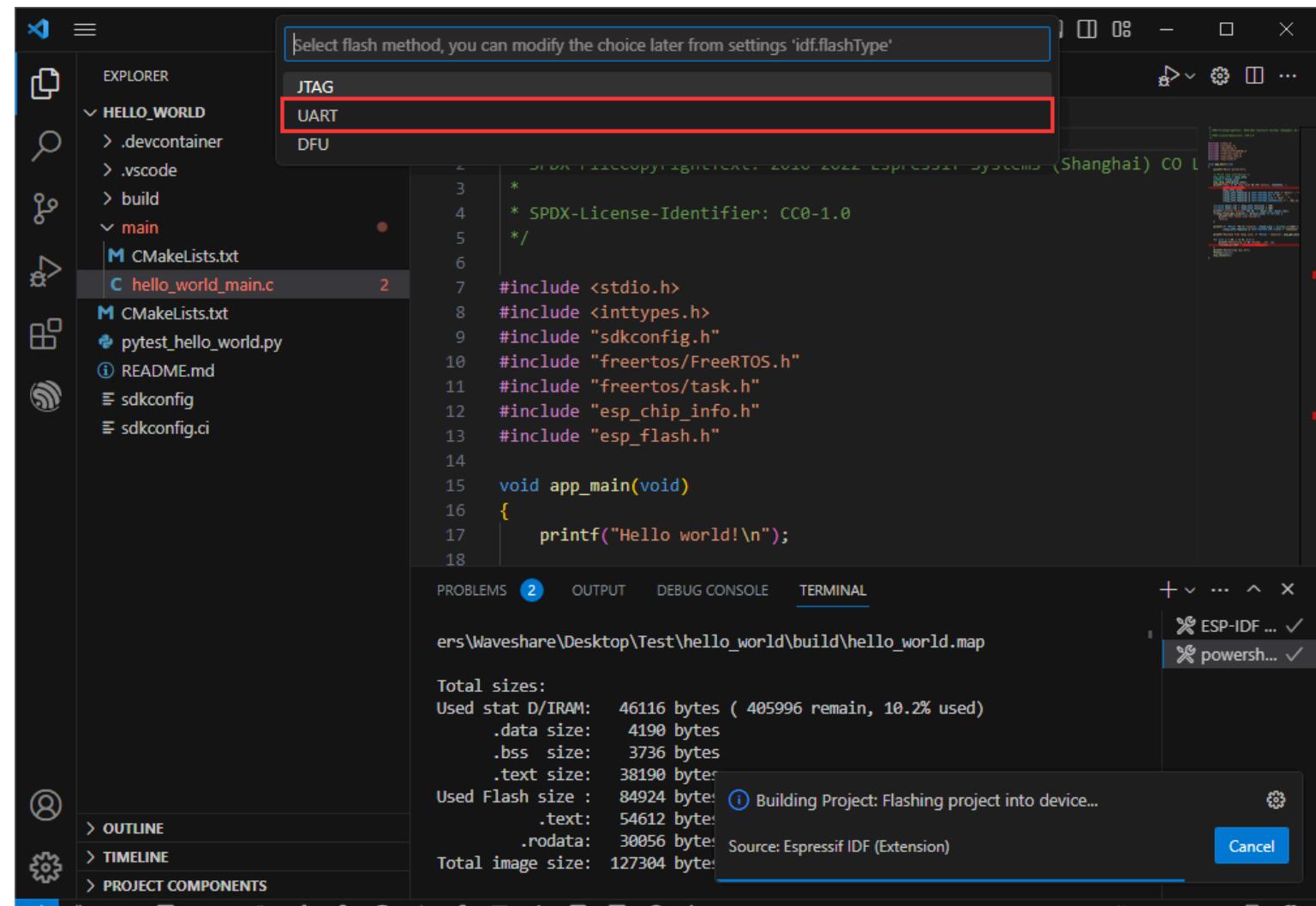
(/wiki/File:ESP32-C6-DEV-KIT-N8-25.png)

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



(/wiki/File:ESP32-C6-DEV-KIT-N8-26.png)

- During this process, the ESP-IDF may take up a lot of CPU resources, so it may cause the system to lag.
 - Flashing the demo for a new project for the first time, you will need to select the download method, and select **UART**.



(/wiki/File:ESP32-C6-DEV-KIT-N8-27.png)

- This can also be changed later in the **Download Methods** section (click on it to bring up the options).

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 and highlighted in the list.
- Code Editor:** Displays the content of "hello_world_main.c". The code includes standard library includes, a main function that prints "Hello world!\n", and chip information printing logic using the esp_chip_info.h header.
- Bottom Status Bar:** Shows connection status to "COM3" and "esp32c6", and a red box highlights the "UART" tab in the status bar.

```
/* SPDX-FileCopyrightText: 2010-2022 Espressif Systems (Shanghai) CO LTD
 * SPDX-License-Identifier: CC0-1.0
 */
#include <stdio.h>
#include <inttypes.h>
#include "sdkconfig.h"
#include "freertos/FreeRTOS.h"
#include "freertos/task.h"
#include "esp_chip_info.h"
#include "esp_flash.h"

void app_main(void)
{
    printf("Hello world!\n");

    /* Print chip information */
    esp_chip_info_t chip_info;
    uint32_t flash_size;
    esp_chip_info(&chip_info);
    printf("This is %s chip with %d CPU core(s), %s%s%s%s, %s\n",
        CONFIG_IDF_TARGET,
        chip_info.cores,
        (chip_info.features & CHIP_FEATURE_WIFI_BGN) ? "WiFi/" : "",
        (chip_info.features & CHIP_FEATURE_BT) ? "BT" : "",
        (chip_info.features & CHIP_FEATURE_BLE) ? "BLE" : "",
        (chip_info.features & CHIP_FEATURE_IEEE802154) ? ", 802.15",
        unsigned major_rev = chip_info.revision / 100;
```

(/wiki/File:ESP32-C6-DEV-KIT-N8-28.png)

- As it comes with the onboard automatic download circuit, there is no need for manual operation to download automatically.

The screenshot shows the Visual Studio Code interface for an ESP-IDF project named "hello_world". The Explorer sidebar on the left lists files and folders: ".devcontainer", ".vscode", "build", "main", "CMakeLists.txt", "hello_world_main.c", "CMakeLists.txt", "pytest_hello_world.py", "README.md", "sdkconfig", and "sdkconfig.ci". The "hello_world_main.c" file is selected and shown in the main editor area. The code contains the following C code:

```
1  /*
2  * SPDX-FileCopyrightText: 2010-2022 Espressif Systems (Shanghai) CO LTD
3  *
4  * SPDX-License-Identifier: CC0-1.0
5  */
6
7 #include <stdio.h>
8 #include <inttypes.h>
9 #include "sdkconfig.h"
10 #include "freertos/FreeRTOS.h"
11 #include "freertos/task.h"
12 #include "esp_chip_info.h"
13 #include "esp_flash.h"
14
15 void app_main(void)
16 {
17     printf("Hello world!\n");
18 }
```

The terminal tab at the bottom shows the output of the build and flashing process:

```
MAC_EXT: ff:fe
Uploading stub...
Running stub...
Stub running...
Changing baud rate to 460800
Changed.
Configuring flash size...
Flash will be erased from 0x00000000 to 0x00000000
Flash will be erased from 0x00000000 to 0x00000000
Flash will be erased from 0x00000000 to 0x00000000
Compressed 21248 bytes to 1289
Writing at 0x00000000... (100 %)
```

A progress bar in the terminal indicates the writing process is at 100%. A tooltip for the progress bar says "Building Project: Flashing project into device..." and "Source: Espressif IDF (Extension)".

(/wiki/File:ESP32-C6-DEV-KIT-N8-29.png)

- After successful download, automatically enter the serial monitor, you can see the chip output the corresponding information and be prompted to restart after 10S.

The screenshot shows the Visual Studio Code interface for an ESP-IDF project named "HELLO_WORLD". The Explorer sidebar lists files like ".devcontainer", ".vscode", "build", "main", "CMakeLists.txt", "hello_world_main.c", "CMakeLists.txt", "pytest_hello_world.py", "README.md", "sdkconfig", and "sdkconfig.ci". The "hello_world_main.c" file is open in the editor, showing the standard "Hello World" code. The terminal window at the bottom shows the program output: "Hello world!", followed by details about the esp32c6 chip and a restart loop. A status bar at the bottom indicates the connection is COM3, the board is esp32c6, and the port is UART.

(/wiki/File:ESP32-C6-DEV-KIT-N8-30.png)

Demo Demonstration

Hello World

The official example path: get-started -> hello_world.

The example effect: Output **Hello World!** on the **TERMINAL** window every 10s.

Software Operation

- Create the official example "hello_world" according to the above tutorial. (Create Example)
- The demo is compatible with ESP32-C6, and you can directly use it with no need to modify the demo.
- Modify the COM port and the driver object, click on the compile and flash to run the demo.

```

/* SPDX-FileCopyrightText: 2010-2022 Espressif Systems (Shanghai) CO LTD
 * * SPDX-License-Identifier: CC0-1.0 */

#include <stdio.h>
#include <inttypes.h>
#include "sdkconfig.h"
#include "freertos/FreeRTOS.h"
#include "freertos/task.h"
#include "esp_chip_info.h"
#include "esp_flash.h"

void app_main(void)
{
    printf("Hello world!\n");
}

```

PROBLEMS 2 OUTPUT DEBUG CONSOLE TERMINAL

I (270) coexist: coexist rom version 5b8dcfa
I (275) app_start: Starting scheduler on CPU0
I (280) main_task: Started on CPU0
I (280) main_task: Calling app_main()
Hello world!
This is esp32c6 chip with 1 CPU core(s), WiFi/BLE, 802.15.4 (Zigbee/Thread),
silicon revision v0.0, 2MB external flash
Minimum free heap size: 473432 bytes
Restarting in 10 seconds...
Restarting in 9 seconds...
Restarting in 8 seconds...

(/wiki/File:ESP32-C6-DEV-KIT-N8-30.png)

RGB

Official example path: get-started -> blink.

Example effect: onboard RGB beads blink at 1-second intervals.

Software Operation

- Follow the tutorial above to create the official example blink. (Create Example)

- The demo is compatible with ESP32-C6 and can be used without modifying the demo content.
- Modify the COM port and the driver object (**It is recommended to prioritize the use of the COM port corresponding to USB (can be viewed through the device manager)**), click compile and flash to run the demo.

The screenshot shows the VS Code interface with the following details:

- EXPLORER:** Shows the project structure under "UNTITLED (WORKSPACE)" with files like .devcontainer, .vscode, build, main, CMakeLists.txt, idf_component.yml, Kconfig.projbuild, managed_components, dependencies.lock, pytest_blink.py, README.md, and sdkconfig.
- EDITOR:** Displays the content of `blink_example_main.c`. The code includes includes for stdio.h, FreeRTOS headers, driver/gpio.h, esp_log.h, led_strip.h, and sdkconfig.h. It defines a static const char *TAG and contains comments about the example being in the public domain and the software being distributed "AS IS".
- TERMINAL:** Shows serial output from the device. Log messages include:
 - I (274) app_start: Starting scheduler on CPU0
 - I (279) main_task: Started on CPU0
 - I (279) main_task: Calling app_main()
 - I (279) example: Example configured to blink addressable LED!
 - I (289) gpio: GPIO[8]| InputEn: 0| OutputEn: 1| OpenDrain: 0| Pullup: 1| Pardown: 0| Intr:0
 - I (299) example: Turning the LED OFF!
 - I (1299) example: Turning the LED ON!
 - I (2299) example: Turning the LED OFF!
 - I (3299) example: Turning the LED ON!
- STATUS BAR:** Shows connection to COM3, board esp32c6, and other system information like spaces: 4, UTF-8, CRLF, and ESP-IDF.

(/wiki/File:ESP32-C6-DEV-KIT-N8-41.png)

UART

Official example path: peripherals -> uart-> uart_async_rxxtasks.

Example effect: shorting GPIO4 and GPIO5 to send/receive UART data.

Hardware Connection

ESP32-C6	ESP32-C6 (the same one)
GPIO4	GPIO5

Software Operation

- Create the official example `uart_async_rxxtasks` according to the tutorial above. (Create Example).
- The demo is compatible with ESP32-C6 and can be used without modifying the demo content.
- Modify the COM port and driver object, (**It is recommended to prioritize the use of the COM port corresponding to USB (can be viewed through the device manager)**). click compile and flash to run the demo.

```
/* UART asynchronous example, that uses separate RX and TX tasks
This example code is in the Public Domain (or CC0 licensed, at your
discretion). Unless required by applicable law or agreed to in writing, this
software is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR
CONDITIONS OF ANY KIND, either express or implied.

#include "freertos/FreeRTOS.h"
#include "freertos/task.h"
#include "esp_system.h"
#include "esp_log.h"
#include "driver/uart.h"
#include "string.h"
#include "driver/gpio.h"

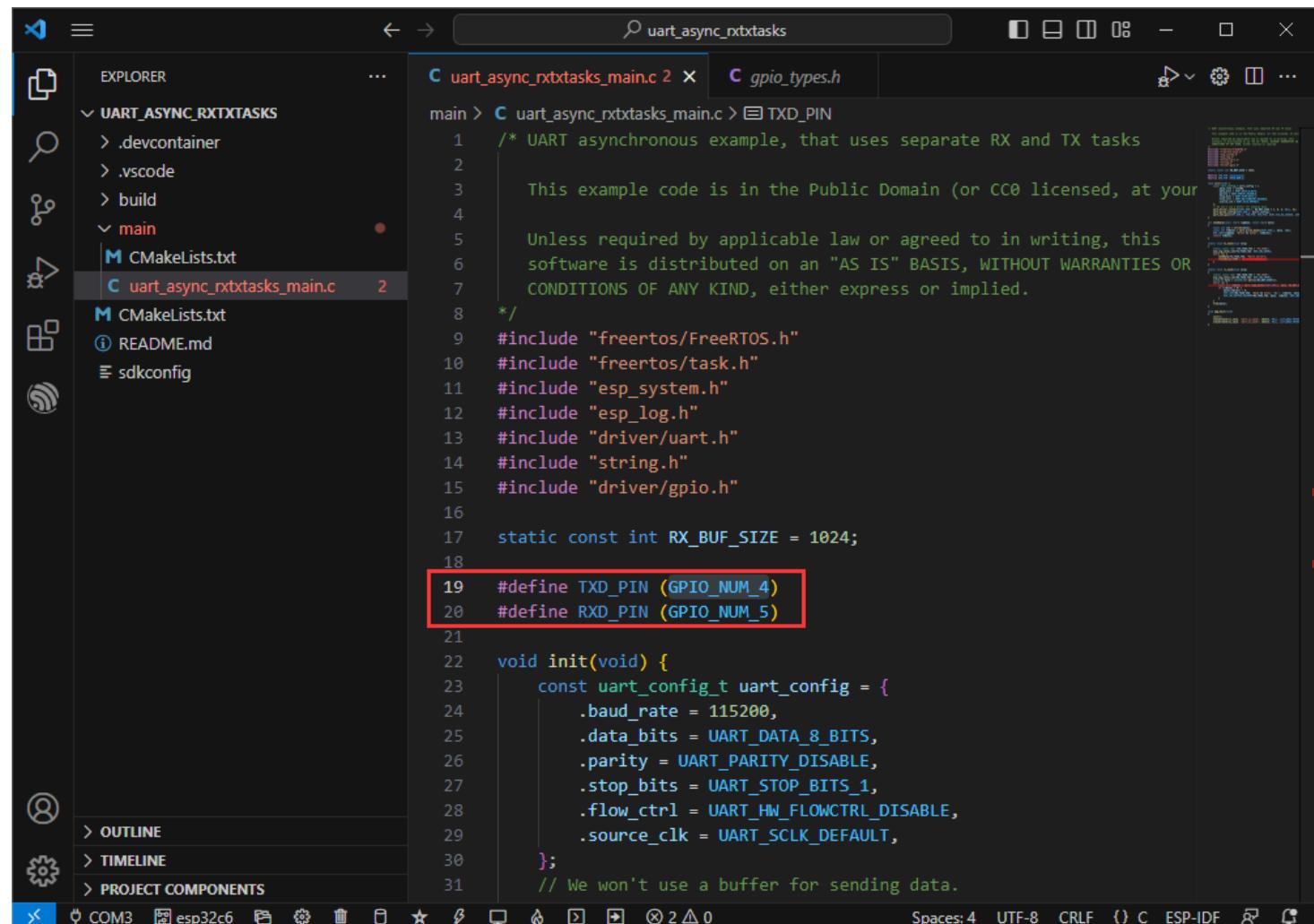
static const int RX_BUF_SIZE = 1024;

#define TXD_PIN (GPIO_NUM_4)
```

I (265) coexist: coex firmware version: 80b0d89
I (270) coexist: coexist rom version 5b8dcfa
I (275) app_start: Starting scheduler on CPU0
I (280) main_task: Started on CPU0
I (280) main_task: Calling app_main()
I (280) TX_TASK: Wrote 11 bytes
I (290) main_task: Returned from app_main()
I (1290) RX_TASK: Read 11 bytes: 'Hello world'
I (1290) RX_TASK: 0x4087e22c 48 65 6c 6c f6 20 77 6f 72 6c 64
|Hello world|
I (2290) TX_TASK: Wrote 11 bytes

(/wiki/File:ESP32-C6-DEV-KIT-N8-77.png)

- Hardware connection according to the GPIO used.



```
/* UART asynchronous example, that uses separate RX and TX tasks
This example code is in the Public Domain (or CC0 licensed, at your
discretion). Unless required by applicable law or agreed to in writing, this
software is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR
CONDITIONS OF ANY KIND, either express or implied.

#define TXD_PIN (GPIO_NUM_4)
#define RXD_PIN (GPIO_NUM_5)

void init(void) {
    const uart_config_t uart_config = {
        .baud_rate = 115200,
        .data_bits = UART_DATA_8_BITS,
        .parity = UART_PARITY_DISABLE,
        .stop_bits = UART_STOP_BITS_1,
        .flow_ctrl = UART_HW_FLOWCTRL_DISABLE,
        .source_clk = UART_SCLK_DEFAULT,
    };
    // We won't use a buffer for sending data.
```

(/wiki/File:ESP32-C6-DEV-KIT-N8-78.png)

- You can go to the definition file to see the actual GPIOs used (check **GPIO_NUM_4** -> Right click -> **Go to Definition**).

The screenshot shows the Visual Studio Code interface with the ESP-IDF extension installed. The current file is `uart_async_rxtxtasks_main.c`. A context menu is open over the symbol `GPIO_NUM_4`, which is highlighted with a red circle labeled '1'. The menu item `Go to Definition` is also highlighted with a red rectangle and has a red circle labeled '2' next to it. Other menu items include `Go to Declaration`, `Go to Type Definition`, `Go to References`, `Peek`, `Find All References`, `Show Call Hierarchy`, `Rename Symbol`, `Change All Occurrences`, `Format Document`, `Format Document With...`, `Format Selection`, `Refactor...`, `Cut`, `Copy`, `Paste`, `Switch Header/Source`, `Go to Symbol in Editor...`, `Go to Symbol in Workspace...`, `Run Code Analysis on Active File`, `Restart IntelliSense for Active File`, `Add Debug Configuration`, and `Generate Dependency Graph`.

```

main > C uart_async_rxtxtasks_main.c 2 X gp
6     software is distributed
7     CONDITIONS OF ANY KIND,
8 */
9 #include "freertos/FreeRTOS
10 #include "freertos/task.h"
11 #include "esp_system.h"
12 #include "esp_log.h"
13 #include "driver/uart.h"
14 #include "string.h"
15 #include "driver/gpio.h"
16
17 static const int RX_BUF_SIZE = 1024;
18
19 #define TXD_PIN (GPIO_NUM_4) 1
20 #define RXD_PIN (GPIO_NUM_5)
21
22 void init(void) {
23     const uart_config_t uart_config = {
24         .baud_rate = 115200,
25         .data_bits = UART_DATABITS_8,
26         .parity = UART_PARITY_NONE,
27         .stop_bits = UART_STOPBITS_1,
28         .flow_ctrl = UART_HW_FLOWCTRL_DISABLE,
29         .source_clk = UART_SCLK_SRC_XTAL,
30     };
31     // We won't use a buffer
32     uart_driver_install(UART_NUM_1, RX_BUF_SIZE, TXD_PIN, RXD_PIN, &uart_config, 10);
33     uart_param_config(UART_NUM_1, &uart_config);
34     uart_set_pin(UART_NUM_1, TXD_PIN, RXD_PIN, GPIO_NUM_5, GPIO_NUM_4);
35 }
36

```

(/wiki/File:ESP32-C6-DEV-KIT-N8-79.png)

Bluetooth

Official sample path: bluetooth -> bluedroid -> ble -> gatt_server.

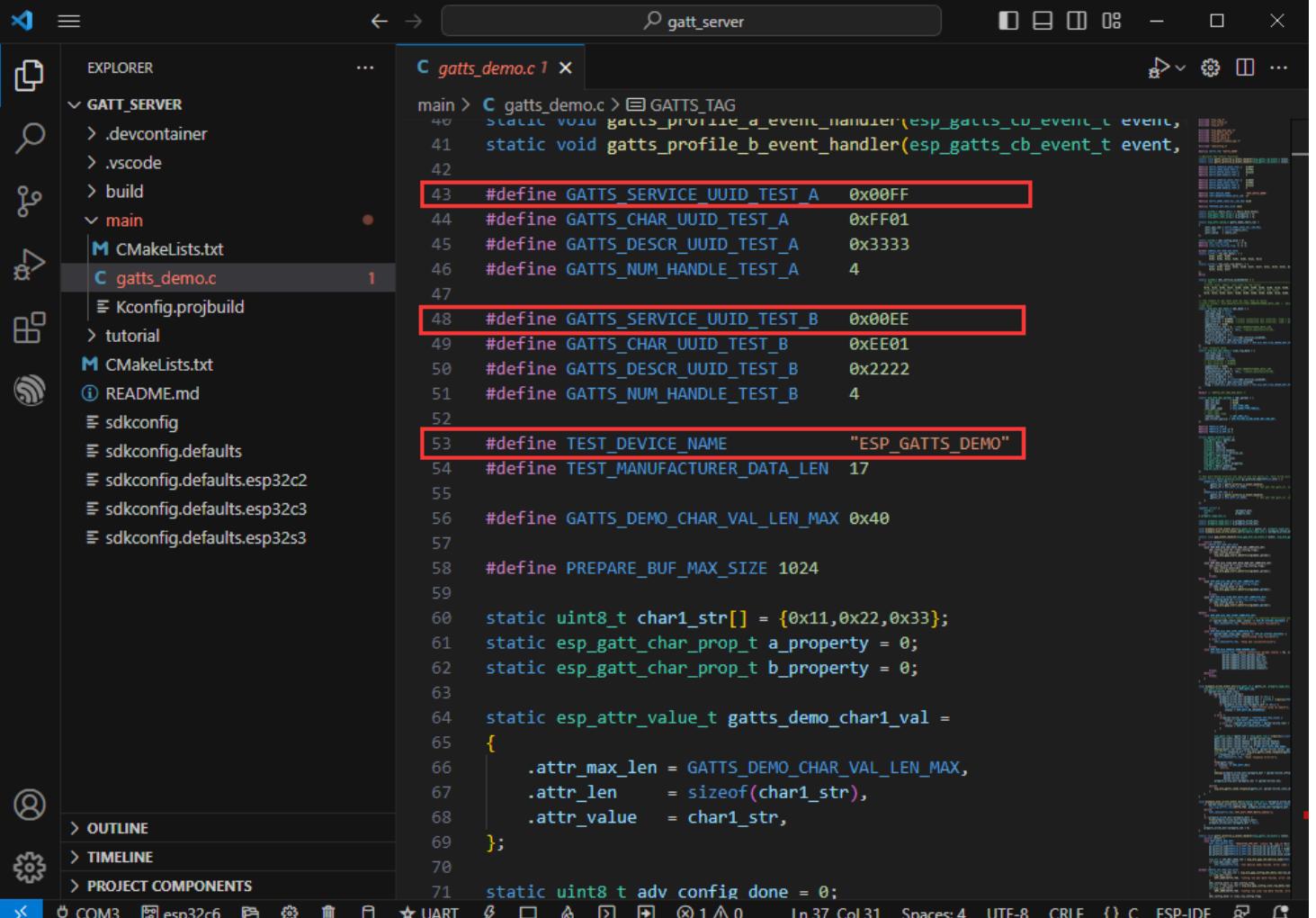
Example effect: ESP32-C6 and cell phone Bluetooth debugging assistant for data transmission.

Software Operation

- Install the Bluetooth debugging assistant (<https://files.waveshare.com/wiki/ESP32-C6-DEV-KIT>

-N8/ESP32-C6_TO_BLEAssist.ZIP) on your phone.

- Follow the tutorial above to create the official example gatt_server. (Create Example)
- The demo is compatible with ESP32-C6 and can be used without modifying the demo content.
- Bluetooth name and UUID, Bluetooth name is **ESP_GATTS_DEMO**.



The screenshot shows the Visual Studio Code interface with the file `gatts_demo.c` open. The code defines service UUIDs and device information:

```
main > C gatts_demo.c > GATTS_TAG
40 static void gatts_profile_a_event_handler(esp_gatts_cb_event_t event,
41     static void gatts_profile_b_event_handler(esp_gatts_cb_event_t event,
42
43 #define GATTS_SERVICE_UUID_TEST_A 0x00FF
44 #define GATTS_CHAR_UUID_TEST_A 0xFF01
45 #define GATTS_DESCR_UUID_TEST_A 0x3333
46 #define GATTS_NUM_HANDLE_TEST_A 4
47
48 #define GATTS_SERVICE_UUID_TEST_B 0x00EE
49 #define GATTS_CHAR_UUID_TEST_B 0xEE01
50 #define GATTS_DESCR_UUID_TEST_B 0x2222
51 #define GATTS_NUM_HANDLE_TEST_B 4
52
53 #define TEST_DEVICE_NAME "ESP_GATTS_DEMO"
54 #define TEST_MANUFACTURER_DATA_LEN 17
55
56 #define GATTS_DEMO_CHAR_VAL_LEN_MAX 0x40
57
58 #define PREPARE_BUF_MAX_SIZE 1024
59
60 static uint8_t char1_str[] = {0x11,0x22,0x33};
61 static esp_gatt_char_prop_t a_property = 0;
62 static esp_gatt_char_prop_t b_property = 0;
63
64 static esp_attr_value_t gatts_demo_char1_val =
65 {
66     .attr_max_len = GATTS_DEMO_CHAR_VAL_LEN_MAX,
67     .attr_len     = sizeof(char1_str),
68     .attr_value   = char1_str,
69 };
70
71 static uint8_t adv_config_done = 0;
```

(/wiki/File:ESP32-C6-DEV-KIT-N8-50.png)

- Modify the COM port and the driver object, (**It is recommended to prioritize the use of the COM port corresponding to USB (can be viewed through the device manager)**). click on the compile and flash to run the demo.

```
#define GATTS_SERVICE_UUID_TEST_A 0x00FF
#define GATTS_CHAR_UUID_TEST_A 0xFF01
#define GATTS_DESCR_UUID_TEST_A 0x3333
#define GATTS_NUM_HANDLE_TEST_A 4
#define GATTS_SERVICE_UUID_TEST_B 0x00EE
#define GATTS_CHAR_UUID_TEST_B 0xEE01
#define GATTS_DESCR_UUID_TEST_B 0x2222
#define GATTS_NUM_HANDLE_TEST_B 4
#define TEST_DEVICE_NAME "ESP_GATTS_DEMO"
#define TEST_MANUFACTURER_DATA_LEN 17
#define GATTS_DEMO_CHAR_VAL_LEN_MAX 0x40
```

I (568) GATTS_DEMO: CREATE_SERVICE_EVT, status 0, service_handle 44
I (578) GATTS_DEMO: SERVICE_START_EVT, status 0, service_handle 44
I (578) GATTS_DEMO: ADD_CHAR_EVT, status 0, attr_handle 46, service_handle 44
I (588) GATTS_DEMO: ADD_DESCR_EVT, status 0, attr_handle 47, service_handle 44
I (598) main_task: Returned from app_main()

(/wiki/File:ESP32-C6-DEV-KIT-N8-51.png)

- Connecting the ESP_GATTS_DEMO Bluetooth device on the phone.

BLE调试助手

Scanner Bonded Services Log

N/A
39:54:CE:BA:AA:F9
NOT BONDED -97 dBm

ESP_GATTs_DEMO
40:4C:CA:45:FE:46
NOT BONDED -27 dBm

N/A
58:D5:3E:6B:B4:D2
NOT BONDED -93 dBm

N/A
31:8F:08:40:2C:8D
NOT BONDED -71 dBm

N/A
32:8C:86:B9:40:3F
NOT BONDED -91 dBm

N/A
23:77:42:FC:F4:DC
NOT BONDED -94 dBm

N/A
35:45:3D:36:F7:3B
NOT BONDED -84 dBm

CONNECT :

Generic Attribute
UUID: 00001801-0000-1000-8000-00805f9b34fb
PRIMARY SERVICE

Generic Access
UUID: 00001800-0000-1000-8000-00805f9b34fb
PRIMARY SERVICE

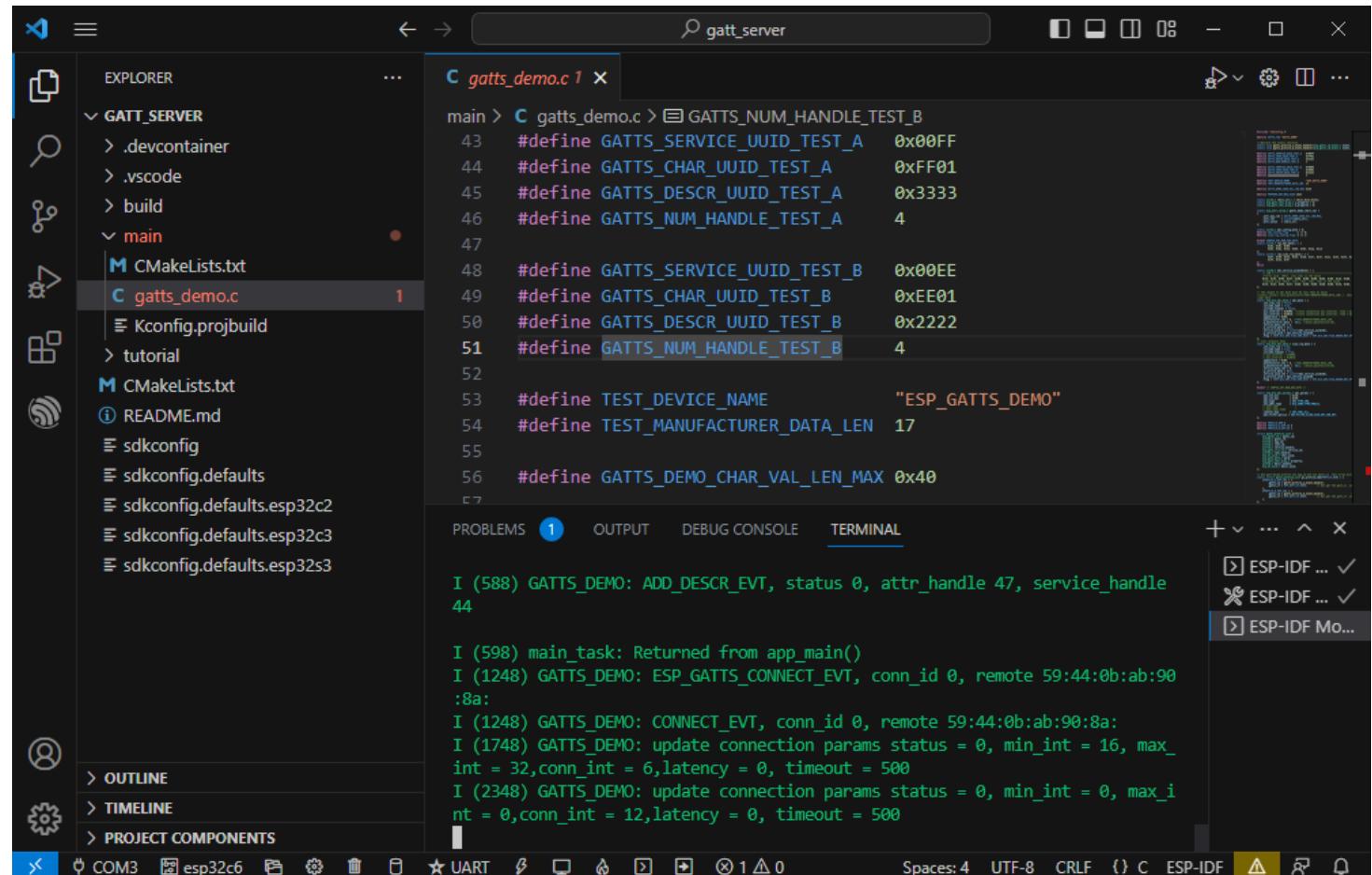
Unknown Service
UUID: 000000ff-0000-1000-8000-00805f9b34fb
PRIMARY SERVICE

Unknown Service
UUID: 000000ee-0000-1000-8000-00805f9b34fb
PRIMARY SERVICE

✓ 蓝牙连接成功

(/wiki/File:1000px-ESP32-C6_TO_Sample_27.png)

- The effect of a successful connection is shown below:



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

- Explorer View:** Shows the project structure under "GATT_SERVER". The "main" folder contains "CMakeLists.txt", "gatt_demo.c", "Kconfig.projbuild", "tutorial", and several "sdkconfig.*" files.
- Editor View:** The main editor window displays the content of "gatt_demo.c". The code defines two GATT services (TEST_A and TEST_B) with specific UUIDs and characteristics.
- Terminal View:** The bottom right shows the terminal output with log messages indicating the connection process between the ESP32 and a host device.

```

main > C gatt_demo.c 1
main > C gatt_demo.c > GATT_NUM_HANDLE_TEST_B
43 #define GATT_SERVICE_UUID_TEST_A 0x00FF
44 #define GATT_CHAR_UUID_TEST_A 0xFF01
45 #define GATT_DESCR_UUID_TEST_A 0x3333
46 #define GATT_NUM_HANDLE_TEST_A 4
47
48 #define GATT_SERVICE_UUID_TEST_B 0x00EE
49 #define GATT_CHAR_UUID_TEST_B 0xEE01
50 #define GATT_DESCR_UUID_TEST_B 0x2222
51 #define GATT_NUM_HANDLE_TEST_B 4
52
53 #define TEST_DEVICE_NAME "ESP_GATT_DEMO"
54 #define TEST_MANUFACTURER_DATA_LEN 17
55
56 #define GATT_DEMO_CHAR_VAL_LEN_MAX 0x40
57

PROBLEMS 1 OUTPUT DEBUG CONSOLE TERMINAL

I (588) GATT_DEMO: ADD_DESCR_EVT, status 0, attr_handle 47, service_handle 44
I (598) main_task: Returned from app_main()
I (1248) GATT_DEMO: ESP_GATT_CONNECT_EVT, conn_id 0, remote 59:44:0b:ab:90:8a:
I (1248) GATT_DEMO: CONNECT_EVT, conn_id 0, remote 59:44:0b:ab:90:8a:
I (1748) GATT_DEMO: update connection params status = 0, min_int = 16, max_int = 32, conn_int = 6, latency = 0, timeout = 500
I (2348) GATT_DEMO: update connection params status = 0, min_int = 0, max_int = 0, conn_int = 12, latency = 0, timeout = 500

```

(/wiki/File:ESP32-C6-DEV-KIT-N8-52.png)

- Based on the UUID value in the demo, select one of the two servers for upstream transmission.

The screenshot shows two main screens of the ESP-GATTs mobile application.

Left Screen (Service Discovery):

- Services:** A tab labeled "Services" is selected, showing a list of discovered services.
- Generic Attribute:** A service with UUID: 00001801-0000-1000-8000-00805f9b34fb, labeled as PRIMARY SERVICE.
- Generic Access:** A service with UUID: 00001800-0000-1000-8000-00805f9b34fb, labeled as PRIMARY SERVICE.
- Unknown Service:** A service with UUID: 000000ff-0000-1000-8000-00805f9b34fb, labeled as PRIMARY SERVICE. This service is expanded to show its characteristics.
- Unknown Characteristic:** A characteristic with UUID: 0000ff01-0000-1000-8000-00805f9b34fb. Properties: READ WRITE NOTIFY. Descriptors: Client Characteristic Configuration. UUID: 00002902-0000-1000-8000-00805f9b34fb.
- Unknown Service:** A service with UUID: 000000ee-0000-1000-8000-00805f9b34fb, labeled as PRIMARY SERVICE. This service is expanded to show its characteristics.
- Unknown Characteristic:** A characteristic with UUID: 0000ee01-0000-1000-8000-00805f9b34fb. Properties: READ WRITE NOTIFY. Descriptors: Client Characteristic Configuration. UUID: 00002902-0000-1000-8000-00805f9b34fb.

Right Screen (Data Transmission):

- 发送数据 (Send Data):** A screen for sending data to the selected service and characteristic.
- Transmission Mode:** Set to "定时发送" (Periodic Send), indicated by a red box.
- Data Length:** 320 Byte, HEX mode is selected.
- Transmission Interval:** 1000 ms, indicated by a red box.
- Data Value:** Hexadecimal value: 00 11 22 33 44 55 66 77 88 99, indicated by a red box.
- Buttons:** 清空 (Clear) and 停止 (Stop).
- Keyboard:** A numeric keyboard for entering data values.



(/wiki/File:1000px-ESP32-C6_TO_Sample_29.png)

- The ESP32-C6 receives data.

```

main > C gatts_demo.c > GATTS_NUM_HANDLE_TEST_B
43 #define GATTS_SERVICE_UUID_TEST_A 0x00FF
44 #define GATTS_CHAR_UUID_TEST_A 0xFF01
45 #define GATTS_DESCR_UUID_TEST_A 0x3333
46 #define GATTS_NUM_HANDLE_TEST_A 4
47
48 #define GATTS_SERVICE_UUID_TEST_B 0x00EE
49 #define GATTS_CHAR_UUID_TEST_B 0xEE01
50 #define GATTS_DESCR_UUID_TEST_B 0x2222
51 #define GATTS_NUM_HANDLE_TEST_B 4
52
53 #define TEST_DEVICE_NAME "ESP_GATTS_DEMO"
54 #define TEST_MANUFACTURER_DATA_LEN 17
55
56 #define GATTS_DEMO_CHAR_VAL_LEN_MAX 0x40

```

PROBLEMS 1 OUTPUT DEBUG CONSOLE TERMINAL

```

I (457848) GATTS_DEMO: GATT_WRITE_EVT, value len 10, value :
I (457858) GATTS_DEMO: 00 11 22 33 44 55 66 77 88 99
I (458888) GATTS_DEMO: GATT_WRITE_EVT, conn_id 0, trans_id 27, handle 42
I (458888) GATTS_DEMO: GATT_WRITE_EVT, value len 10, value :
I (458888) GATTS_DEMO: 00 11 22 33 44 55 66 77 88 99
I (459948) GATTS_DEMO: GATT_WRITE_EVT, conn_id 0, trans_id 28, handle 42
I (459948) GATTS_DEMO: GATT_WRITE_EVT, value len 10, value :
I (459958) GATTS_DEMO: 00 11 22 33 44 55 66 77 88 99
I (460988) GATTS_DEMO: GATT_WRITE_EVT, conn_id 0, trans_id 29, handle 42
I (460988) GATTS_DEMO: GATT_WRITE_EVT, value len 10, value :
I (460988) GATTS_DEMO: 00 11 22 33 44 55 66 77 88 99

```

(/wiki/File:ESP32-C6-DEV-KIT-N8-54.png)

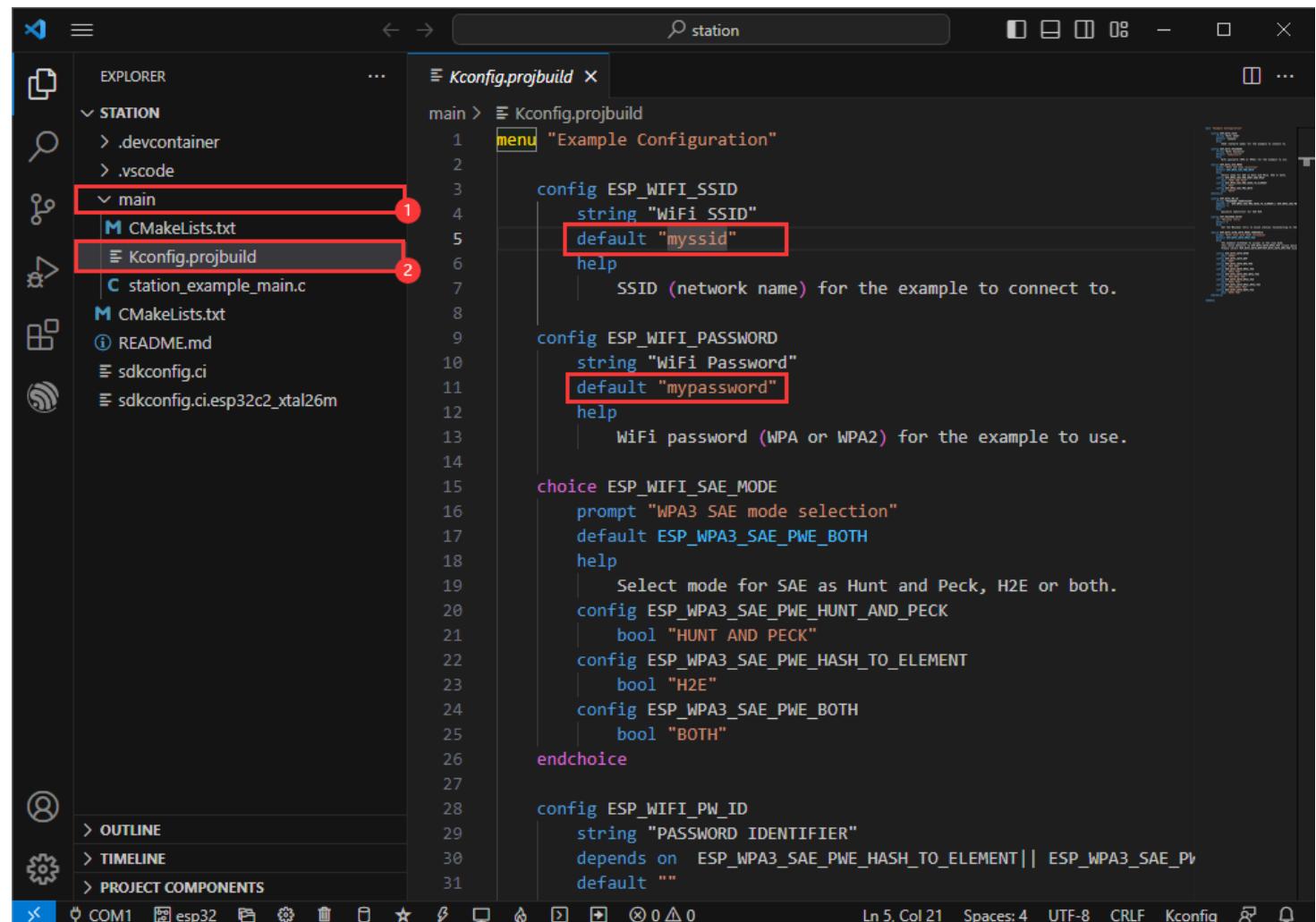
WIFI

Official example path: wifi -> getting_started -> station.

Example effect: ESP32-C6 connects to WIFI.

Software Operation

- Create the official example station according to the tutorial above. (Create Example)
- Modify the contents of the demo to connect to the required WiFi.
- Go to the **Kconfig.projbuild** file.



```
main > Kconfig.projbuild
1 menu "Example Configuration"
2
3 config ESP_WIFI_SSID
4     string "WiFi SSID"
5     default "myssid"
6     help
7         SSID (network name) for the example to connect to.
8
9 config ESP_WIFI_PASSWORD
10    string "WiFi Password"
11    default "mypassword"
12    help
13        WiFi password (WPA or WPA2) for the example to use.
14
15 choice ESP_WIFI_SAE_MODE
16     prompt "WPA3 SAE mode selection"
17     default ESP_WPA3_SAE_PWE_BOTH
18     help
19         Select mode for SAE as Hunt and Peck, H2E or both.
20     config ESP_WPA3_SAE_PWE_HUNT_AND_PECK
21         bool "HUNT AND PECK"
22     config ESP_WPA3_SAE_PWE_HASH_TO_ELEMENT
23         bool "H2E"
24     config ESP_WPA3_SAE_PWE_BOTH
25         bool "BOTH"
26     endchoice
27
28 config ESP_WIFI_PW_ID
29     string "PASSWORD IDENTIFIER"
30     depends on ESP_WPA3_SAE_PWE_HASH_TO_ELEMENT || ESP_WPA3_SAE_PWE_HUNT_AND_PECK
31     default ""
```

(/wiki/File:ESP32-C6-DEV-KIT-N8-60.png)

- Change the original **WiFi SSID** and **WiFi Password** to the WiFi information you want to connect to.

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

- Explorer View:** Shows a project structure under "STATION". The "main" folder contains "CMakeLists.txt", "Kconfig.projbuild" (which is selected), "station_example_main.c", and other files like "README.md" and "sdkconfig.ci".
- Code Editor:** Displays the "Kconfig.projbuild" file content. The file defines a menu for configuration, including options for WiFi SSID and Password, and a choice for SAE mode selection. The "default" values for WiFi SSID ("Waveshare-TL-WR886N_2L") and WiFi Password ("waveshare0755") are highlighted with red boxes.
- Search Bar:** Contains the text "station".
- Bottom Status Bar:** Shows "Ln 11, Col 31 (13 selected)" and other standard status bar information.

```
menu "Example Configuration"
config ESP_WIFI_SSID
    string "WiFi SSID"
    default "Waveshare-TL-WR886N_2L"
    help
        SSID (network name) for the example to connect to.

config ESP_WIFI_PASSWORD
    string "WiFi Password"
    default "waveshare0755"
    help
        WiFi password (WPA or WPA2) for the example to use.

choice ESP_WIFI_SAE_MODE
    prompt "WPA3 SAE mode selection"
    default ESP_WPA3_SAE_PWE_BOTH
    help
        Select mode for SAE as Hunt and Peck, H2E or both.
    config ESP_WPA3_SAE_PWE_HUNT_AND_PECK
        bool "HUNT AND PECK"
    config ESP_WPA3_SAE_PWE_HASH_TO_ELEMENT
        bool "H2E"
    config ESP_WPA3_SAE_PWE_BOTH
        bool "BOTH"
endchoice

config ESP_WIFI_PW_ID
    string "PASSWORD IDENTIFIER"
    depends on ESP_WPA3_SAE_PWE_HASH_TO_ELEMENT || ESP_WPA3_SAE_PWE_BOTH
    default ""
```

(/wiki/File:ESP32-C6-DEV-KIT-N8-61.png)

- Modify the COM port and the driver object. **It is recommended to prioritize the use of the COM port corresponding to USB (can be viewed through the device manager)**. click on the compile and flash to run the demo.

The screenshot shows the Visual Studio Code interface for an ESP32 project named "STATION". The left sidebar displays project files including CMakeLists.txt, Kconfig.projbuild, station_example_main.c (which is currently selected), and various configuration files. The main editor window shows the source code for station_example_main.c, which includes WiFi configuration settings. The bottom right corner of the editor shows a preview of the ESP-IDF documentation. The terminal tab at the bottom shows the serial port output, indicating successful WiFi connection and beacon transmission.

```

main > C station_example_main.c > EXAMPLE_ESP_WIFI_SSID
12 #include "freertos/event_groups.h"
13 #include "esp_system.h"
14 #include "esp_wifi.h"
15 #include "esp_event.h"
16 #include "esp_log.h"
17 #include "nvs_flash.h"
18
19 #include "lwip/err.h"
20 #include "lwip/sys.h"
21
22 /* The examples use WiFi configuration that you can set via project config
23    If you'd rather not, just change the below entries to strings with
24    the config you want - ie #define EXAMPLE_WIFI_SSID "mywifissid"
25 */
26
27 #define EXAMPLE_ESP_WIFI_SSID      CONFIG_ESP_WIFI_SSID
28 #define EXAMPLE_ESP_WIFI_PASS     CONFIG_ESP_WIFI_PASSWORD

```

PROBLEMS 7 OUTPUT DEBUG CONSOLE TERMINAL

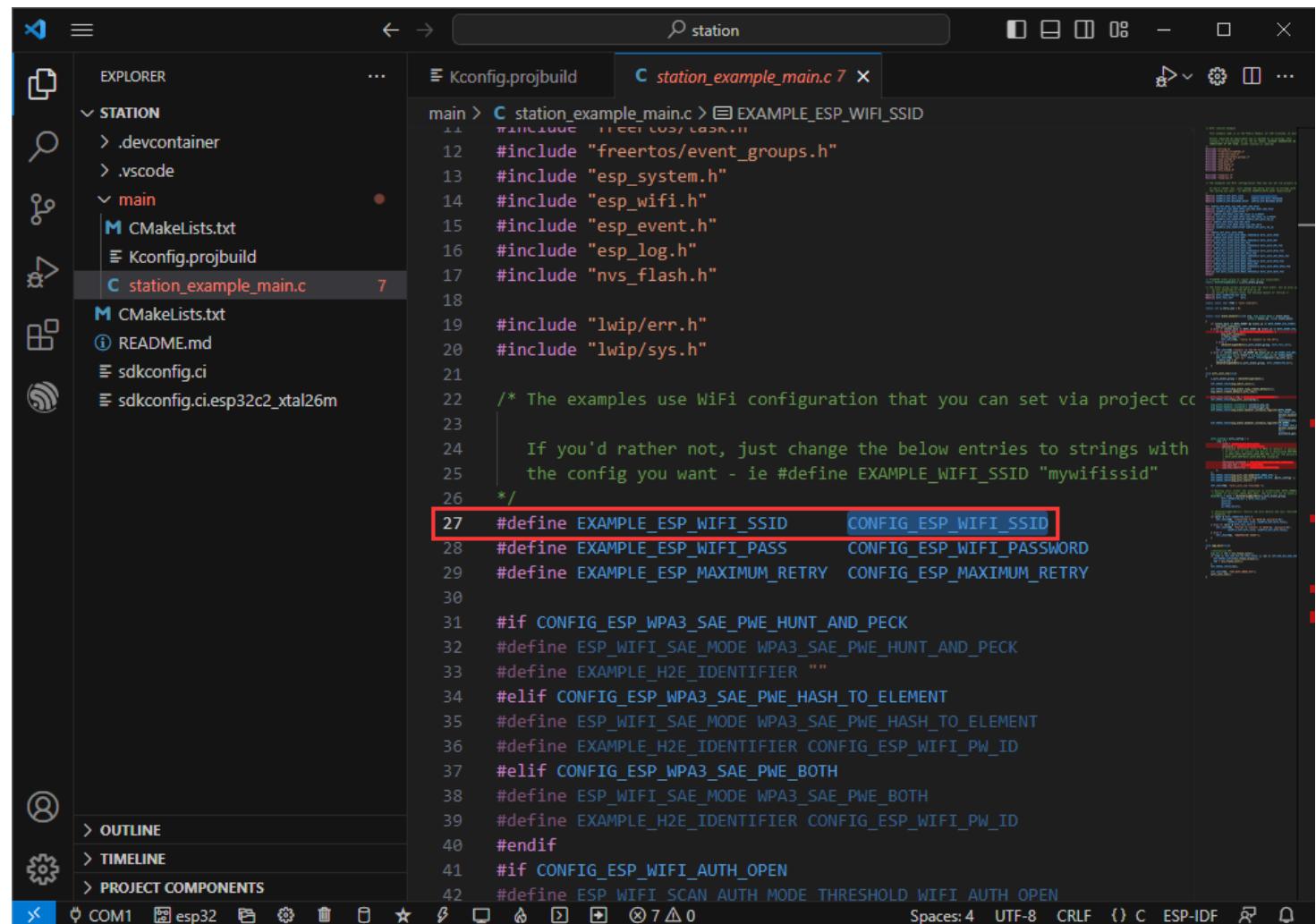
```

I (1598) wifi:[ADDBA]RX addba response, status:0, tid:7/tb:1(0xa1), bufsize:64, batimeout:0, txa_wnd:64
I (1608) wifi:[ADDBA]RX addba response, status:0, tid:5/tb:1(0xa1), bufsize:64, batimeout:0, txa_wnd:64
I (1658) wifi:AP's beacon interval = 102400 us, DTTIM period = 1
I (2588) esp_netif_handlers: sta ip: 192.168.6.69, mask: 255.255.255.0, gw: 192.168.6.1
I (2588) wifi station: got ip:192.168.6.69
I (2588) wifi station: connected to ap SSID:Waveshare-TL-WR886N_2L password: waveshare0755
I (2598) main_task: Returned from app_main()

```

(/wiki/File:ESP32-C6-DEV-KIT-N8-62.png)

- You can check the value of **CONFIG_ESP_WIFI_SSID**.
- Go to the **station_example_main.c** file.

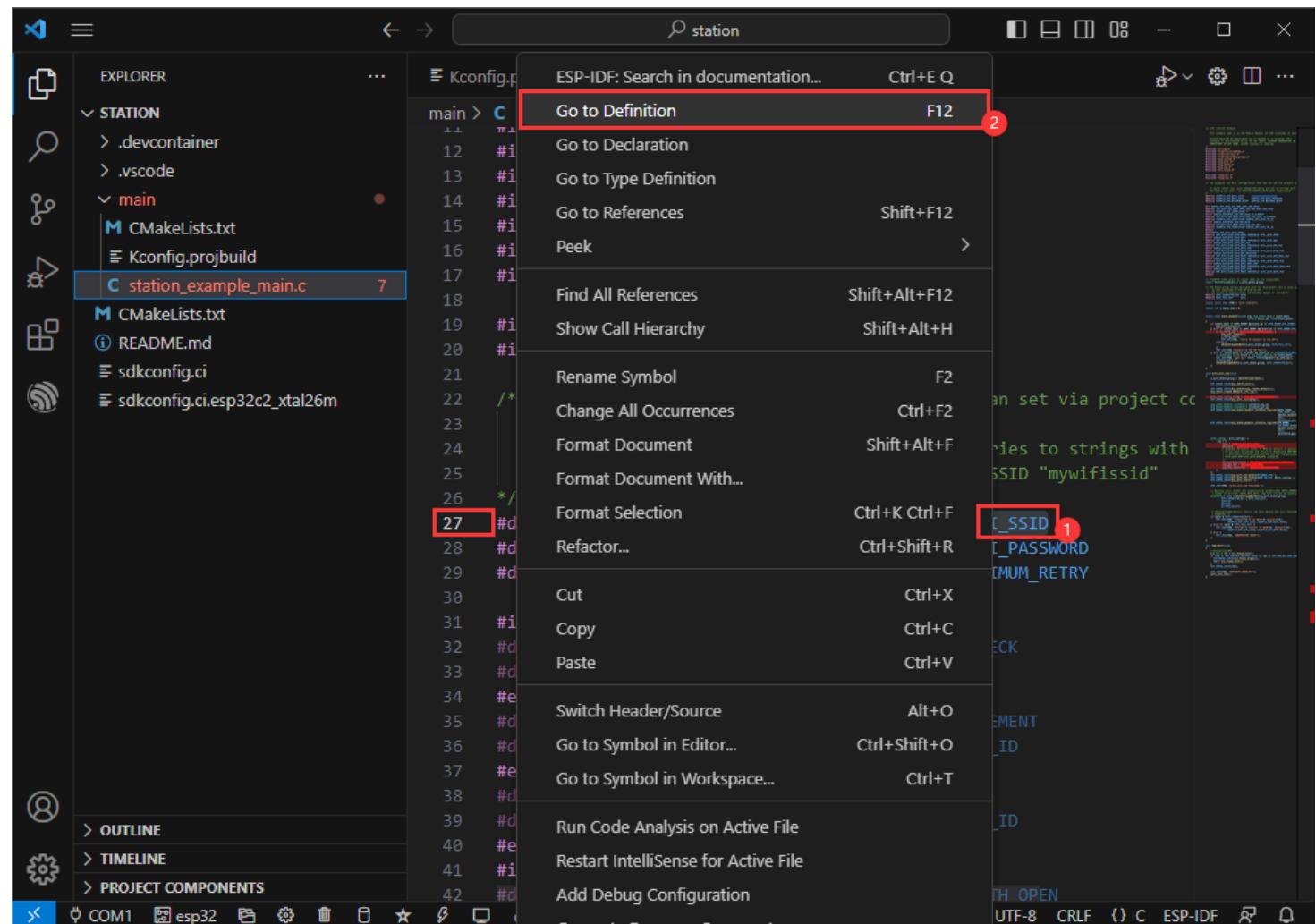


```
main > C station_example_main.c > EXAMPLE_ESP_WIFI_SSID
12 #include "freertos/event_groups.h"
13 #include "esp_system.h"
14 #include "esp_wifi.h"
15 #include "esp_event.h"
16 #include "esp_log.h"
17 #include "nvs_flash.h"
18
19 #include "lwip/err.h"
20 #include "lwip/sys.h"
21
22 /* The examples use WiFi configuration that you can set via project config
23
24     If you'd rather not, just change the below entries to strings with
25     the config you want - ie #define EXAMPLE_WIFI_SSID "mywifissid"
26 */
27 #define EXAMPLE_ESP_WIFI_SSID      CONFIG_ESP_WIFI_SSID
28 #define EXAMPLE_ESP_WIFI_PASS     CONFIG_ESP_WIFI_PASSWORD
29 #define EXAMPLE_ESP_MAXIMUM_RETRY CONFIG_ESP_MAXIMUM_RETRY
30
31 #if CONFIG_ESP_WPA3_SAE_PWE_HUNT_AND_PECK
32 #define ESP_WIFI_SAE_MODE WPA3_SAE_PWE_HUNT_AND_PECK
33 #define EXAMPLE_H2E_IDENTIFIER ""
34 #elif CONFIG_ESP_WPA3_SAE_PWE_HASH_TO_ELEMENT
35 #define ESP_WIFI_SAE_MODE WPA3_SAE_PWE_HASH_TO_ELEMENT
36 #define EXAMPLE_H2E_IDENTIFIER CONFIG_ESP_WIFI_PW_ID
37 #elif CONFIG_ESP_WPA3_SAE_PWE_BOTH
38 #define ESP_WIFI_SAE_MODE WPA3_SAE_PWE_BOTH
39 #define EXAMPLE_H2E_IDENTIFIER CONFIG_ESP_WIFI_PW_ID
40 #endif
41 #if CONFIG_ESP_WIFI_AUTH_OPEN
42 #define ESP_WIFI_SCAN_AUTH_MODE_THRESHOLD WIFI_AUTH_OPEN

```

(/wiki/File:ESP32-C6-DEV-KIT-N8-63.png)

- Right-click to Go to Definition.



(/wiki/File:ESP32-C6-DEV-KIT-N8-64.png)

- The previously set value can be seen as:

```

build > config > C sdkconfig.h > CONFIG_ESP_WIFI_SSID
376 #define CONFIG_ESPTOOLPY_MONITOR_BAUD 115200
377 #define CONFIG_PARTITION_TABLE_SINGLE_APP 1
378 #define CONFIG_PARTITION_TABLE_CUSTOM_FILENAME "partitions.csv"
379 #define CONFIG_PARTITION_TABLE_FILENAME "partitions_singleapp.csv"
380 #define CONFIG_PARTITION_TABLE_OFFSET 0x8000
381 #define CONFIG_PARTITION_TABLE_MDS 1
382 #define CONFIG_ESP_WIFI_SSID "Waveshare-TL-WR886N_2L" 383 #define CONFIG_ESP_WIFI_PASSWORD "waveshare0755"
384 #define CONFIG_ESP_WPA3_SAE_PWE_BOTH 1
385 #define CONFIG_ESP_WIFI_PW_ID ""
386 #define CONFIG_ESP_MAXIMUM_RETRY 5
387 #define CONFIG_ESP_WIFI_AUTH_WPA2_PSK 1
388 #define CONFIG_COMPILER_OPTIMIZATION_DEFAULT 1
389 #define CONFIG_COMPILER_OPTIMIZATION_ASSERTIONS_ENABLE 1
390 #define CONFIG_COMPILER_FLOAT_LIB_FROM_RVFPLIB 1
391 #define CONFIG_COMPILER_OPTIMIZATION_ASSERTION_LEVEL 2
392 #define CONFIG_COMPILER_HIDE_PATHS_MACROS 1
393 #define CONFIG_COMPILER_STACK_CHECK_MODE_NONE 1
394 #define CONFIG_APPTRACE_DEST_NONE 1
395 #define CONFIG_APPTRACE_DEST_UART_NONE 1
396 #define CONFIG_APPTRACE_UART_TASK_PRIO 1
397 #define CONFIG_APPTRACE_LOCK_ENABLE 1
398 #define CONFIG_SPI_MASTER_ISR_IN_IRAM 1
399 #define CONFIG_SPI_SLAVE_ISR_IN_IRAM 1
400 #define CONFIG_EFUSE_MAX_BLK_LEN 256
401 #define CONFIG_ESP_TLS_USINGMBEDTLS 1
402 #define CONFIG_ESP_TLS_USE_DS_PERIPHERAL 1
403 #define CONFIG_COEX_SW_COEXIST_ENABLE 1
404 #define CONFIG_ESP_ERR_TO_NAME_LOOKUP 1
405 #define CONFIG_ETH_ENABLED 1
406 #define CONFIG_ETH_USE_SPI_ETHERNET 1

```

(/wiki/File:ESP32-C6-DEV-KIT-N8-65.png)

Zigbee

- Official example 1 path: Zigbee -> light_sample -> HA_on_off_switch.
- Official example 2 path: Zigbee -> light_sample -> HA_on_off_light.
- Example effect: use 2 ESP32-C6 boards, use the BOOT key of one ESP32-C6 board (flash HA_on_off_switch demo) to control the RGB LED ON/OFF on the other one.

- **Note: Please flash the HA_on_off_switch demo to one board first, and then flash the HA_on_off_light demo to the other board.**

Software Operation 1

- Create the official example HA_on_off_switch according to the tutorial above. (Create Example)
- The demo is compatible with ESP32-C6 and can be used without modifying the demo content.
- Modify the COM port and the driver object. **It is recommended to prioritize the use of the COM port corresponding to USB (can be viewed through the device manager.)** click compile and flash to run the demo.

The screenshot shows the Visual Studio Code interface for the HA_on_off_switch project. The Explorer sidebar on the left lists the project structure, including .devcontainer, .vscode, build, main, CMakeLists.txt, esp_zb_switch.c, esp_zb_switch.h, idf_component.yml, switch_driver.c, switch_driver.h, managed_components, and partitions.csv. The main editor window displays the content of esp_zb_switch.c, which includes a copyright notice and an include directive for "esp_log.h". The bottom status bar shows terminal output with log messages from phy_init, main_task, and ESP_ZB_ON_OFF_SWITCH. A sidebar on the right lists ESP-IDF components.

(/wiki/File:ESP32-C6-DEV-KIT-N8-66.png)

Software Operation 2

- Follow the tutorial above to create the official example HA_on_off_light. (Create Example)
- The demo is compatible with ESP32-C6 and can be used without modifying the demo.
- Modify the COM port and driver object, click compile and flash to run the demo (**you need to wait for a moment for the two chips to establish a connection**).

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

- EXPLORER** pane on the left displays the project structure for "HA_ON_OFF_LIGHT". The file "esp_zb_light.c" is currently selected.
- CODE** pane shows the content of "esp_zb_light.c". The code includes headers for esp_log.h, FreeRTOS, task.h, ha/esp_zigbee_ha_standard.h, esp_zb_light.h, and nvs_flash.h. It defines a static const char *TAG and a static void function bdb_start_top_level_commissioning_cb.
- PROBLEMS** pane at the bottom shows one error: "#error Define ZB_ED_ROLE in idf.py menuconfig to compile light (End Def)".
- OUTPUT** pane shows log messages from the serial port, including network initialization and joining the network.
- TERMINAL** pane at the bottom right shows the command "ESP-IDF Mo...".

(/wiki/File:ESP32-C6-DEV-KIT-N8-70.png)

- If the device remains unconnected, it may be due to residual network information on the device, so you can erase the device information (Erase Tutorial) and reorganize the network.

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

- Explorer View:** Shows the project structure under "HA_ON_OFF_LIGHT". Files listed include ".devcontainer", ".vscode", "build", "main", "CMakeLists.txt", "esp_zb_light.c", "esp_zb_light.h", "idf_component.yml", "light_driver.c", "light_driver.h", "managed_components", "CMakeLists.txt", "dependencies.lock", "partitions.csv", "README.md", "sdkconfig", and "sdkconfig.defaults".
- Code Editor:** The active file is "esp_zb_light.c". The code includes #include directives for esp_log.h, freertos/FreeRTOS.h, freertos/task.h, ha/esp_zigbee_ha_standard.h, esp_zb_light.h, and nvs_flash.h. It also contains comments about the possibility of damage and notes on the zigbee component's role.
- Terminal:** The terminal tab shows the following log output during the boot process:


```
I (339) sleep: Enable automatic switching of GPIO sleep configuration
I (346) coexist: coex firmware version: 80b0d89
I (351) coexist: coexist rom version 5b8dcfa
I (356) app_start: Starting scheduler on CPU0
I (361) main_task: Started on CPU0
I (361) main_task: Calling app_main()
I (371) gpio: GPIO[8]| InputEn: 0| OutputEn: 1| OpenDrain: 0| Pullup: 1| Pulldown: 0| Intr:0
I (371) phy_init: phy_version 202,b4b3263,May 17 2023,20:14:14
I (431) phy: libbtbb version: b684fc5, May 17 2023, 20:14:35
I (431) main_task: Returned from app_main()
I (441) ESP_ZB_ON_OFF_LIGHT: ZDO signal: 23, status: -1
```
- Right Sidebar:** Includes sections for "PROBLEMS", "OUTPUT", "DEBUG CONSOLE", and "TERMINAL". A "PROBLEMS" section shows one error related to the ZB_ED_ROLE. The "OUTPUT" and "DEBUG CONSOLE" tabs are empty. The "TERMINAL" tab shows the log output.
- Bottom Status Bar:** Displays connection status to COM3, the project name esp32c6, and various system metrics like Ln 1, Col 1, Spaces: 4, UTF-8, CRLF, and file encoding.

(/wiki/File:ESP32-C6-DEV-KIT-N8-71.png)

JTAG Debug

Software Operation

- Create a debugging example, this example uses the official example hello_world. (Create

Example)

- Modify the **launch.json** file.

The screenshot shows the VS Code interface with the following details:

- EXPLORER View:** Shows the project structure under `.vscode`: `c_cpp_properties.json` (1) and `launch.json` (2).
- Code Editor:** Displays the `launch.json` file content (3). The code is:

```
{  
  "version": "0.2.0",  
  "configurations": [  
    {  
      "type": "espidoftool",  
      "name": "Launch",  
      "request": "launch"  
    }  
  ]  
}
```

(/wiki/File:ESP32-C6-DEV-KIT-N8-72.png)

- Input the following content:

```
{  
    "version": "0.2.0",  
    "configurations": [  
        {  
            "name": "GDB",  
            "type": "cppdbg",  
            "request": "launch",  
            "MIMode": "gdb",  
            "miDebuggerPath": "${command:espIdf.getXtensaGdb}",  
            "program": "${workspaceFolder}/build/${command:espIdf.getProjectName}.elf",  
            "windows": {  
                "program": "${workspaceFolder}\\.\\build\\\$${command:espIdf.getProjectName}.elf"  
            },  
            "cwd": "${workspaceFolder}",  
            "environment": [{ "name": "PATH", "value": "${config:idf.customExtraPaths}" }],  
            "setupCommands": [  
                { "text": "target remote :3333" },  
                { "text": "set remote hardware-watchpoint-limit 2"},  
                { "text": "mon reset halt" },  
                { "text": "thb app_main" },  
                { "text": "flushregs" }  
            ],  
            "externalConsole": false,  
            "logging": {  
                "engineLogging": true  
            }  
        }  
    ]  
}
```

(/wiki/File:ESP32-C6-DEV-KIT-N8-73.png)

- The demo is compatible with ESP32-C6 and can be used without modifying the demo content.
- Modify the COM port and the driver object (**Please use the USB interface; the UART interface does not support JTAG debugging. The corresponding COM port can be checked through the Device Manager.**), click compile and flash to run the demo.

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 content of "hello_world_main.c". The code includes standard library includes, task definitions, and the main function which prints "Hello world!".
- Terminal:** Shows the output of the build and run process. It includes initialization logs, task start messages, and the application output "Hello world!".
- Bottom Status Bar:** Shows connection status to COM3, esp32c6, and UART, along with other system information.

(/wiki/File:ESP32-C6-DEV-KIT-N8-30.png)

- Press F1 and input:

ESP-IDF:Device configuration

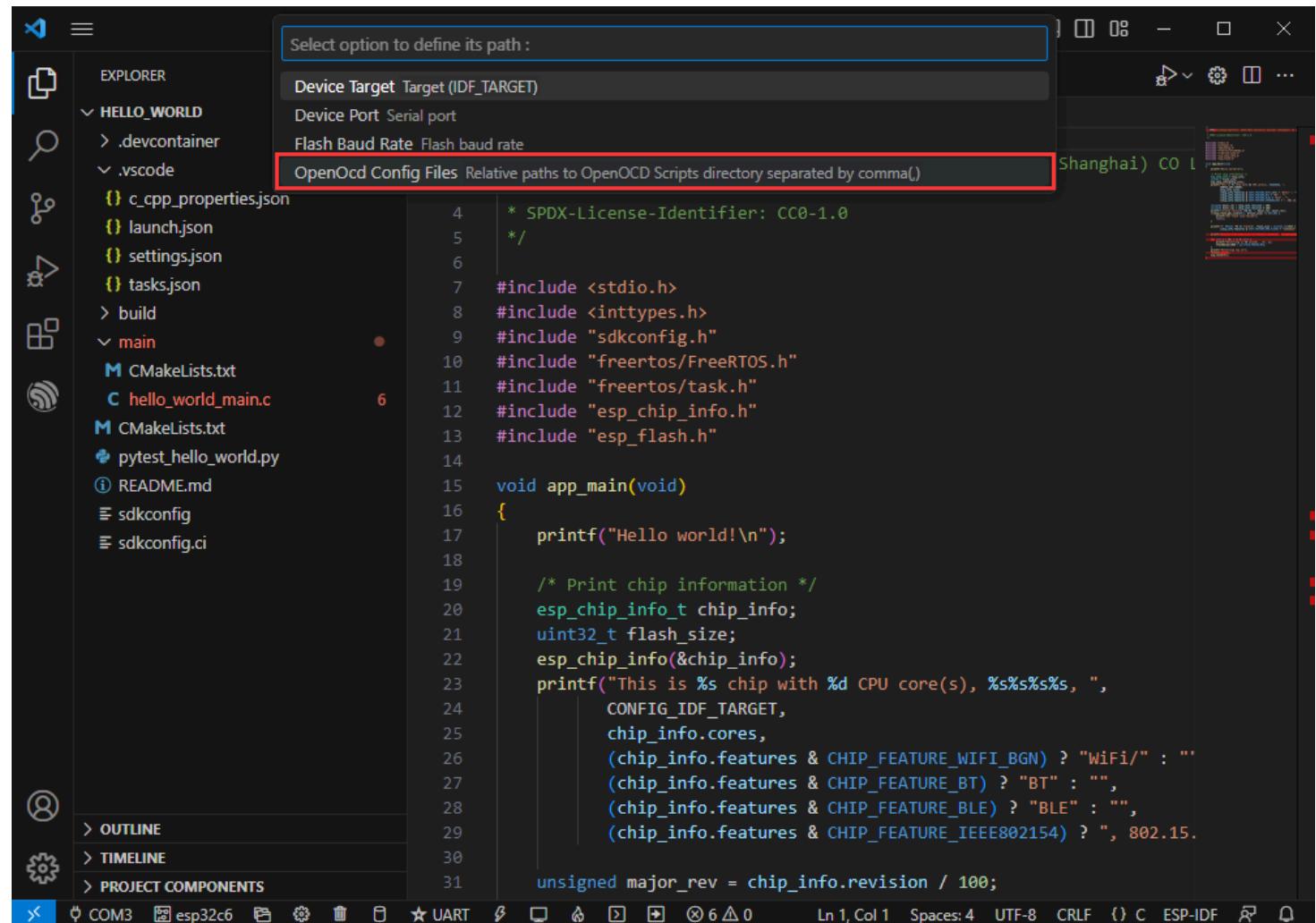
The screenshot shows the Visual Studio Code interface with the following details:

- Explorer View:** Shows the project structure under "HELLO_WORLD".
 - Subfolders: ".devcontainer", ".vscode".
 - Files: "c_cpp_properties.json", "launch.json", "settings.json", "tasks.json", "build", "main".
 - Content of "main": "CMakeLists.txt", "hello_world_main.c", "CMakeLists.txt", "pytest_hello_world.py", "README.md", "sdkconfig", "sdkconfig.ci".
- Editor View:** The file "hello_world_main.c" is open, showing the following code:

```
1  /*
2  * SPDX-FileCopyrightText: 2010-2022 Espressif Systems (Shanghai) CO LTD
3  *
4  * SPDX-License-Identifier: CC0-1.0
5  */
6
7 #include <stdio.h>
8 #include <inttypes.h>
9 #include "sdkconfig.h"
10 #include "freertos/FreeRTOS.h"
11 #include "freertos/task.h"
12 #include "esp_chip_info.h"
13 #include "esp_flash.h"
14
15 void app_main(void)
16 {
17     printf("Hello world!\n");
18
19     /* Print chip information */
20     esp_chip_info_t chip_info;
21     uint32_t flash_size;
22     esp_chip_info(&chip_info);
23     printf("This is %s chip with %d CPU core(s), %s%s%s%s, ",
24           CONFIG_IDF_TARGET,
25           chip_info.cores,
26           (chip_info.features & CHIP_FEATURE_WIFI_BGN) ? "WiFi/" : "",
27           (chip_info.features & CHIP_FEATURE_BT) ? "BT" : "",
28           (chip_info.features & CHIP_FEATURE_BLE) ? "BLE" : "",
29           (chip_info.features & CHIP_FEATURE_IEEE802154) ? ", 802.15",
30
31     unsigned major_rev = chip_info.revision / 100;
```
- Bottom Status Bar:** Shows connection to COM3, board esp32c6, and other system information.

(/wiki/File:ESP32-C6-DEV-KIT-N8-75.png)

- Select **OpenOcd Config Files**.



The screenshot shows the VS Code interface with the Explorer sidebar open, displaying a project structure for a 'HELLO_WORLD' application. The 'vscode' folder contains several configuration files: c_cpp_properties.json, launch.json, settings.json, tasks.json, build, main, CMakeLists.txt, hello_world_main.c, CMakeLists.txt, pytest_hello_world.py, README.md, sdkconfig, and sdkconfig.ci. A context menu is open over the 'OpenOcd Config Files' setting in the top right corner of the code editor. The menu items are: Device Target (IDF_TARGET), Device Port Serial port, Flash Baud Rate Flash baud rate, and OpenOcd Config Files (Relative paths to OpenOCD Scripts directory separated by comma,). The 'OpenOcd Config Files' item is highlighted with a red box.

```

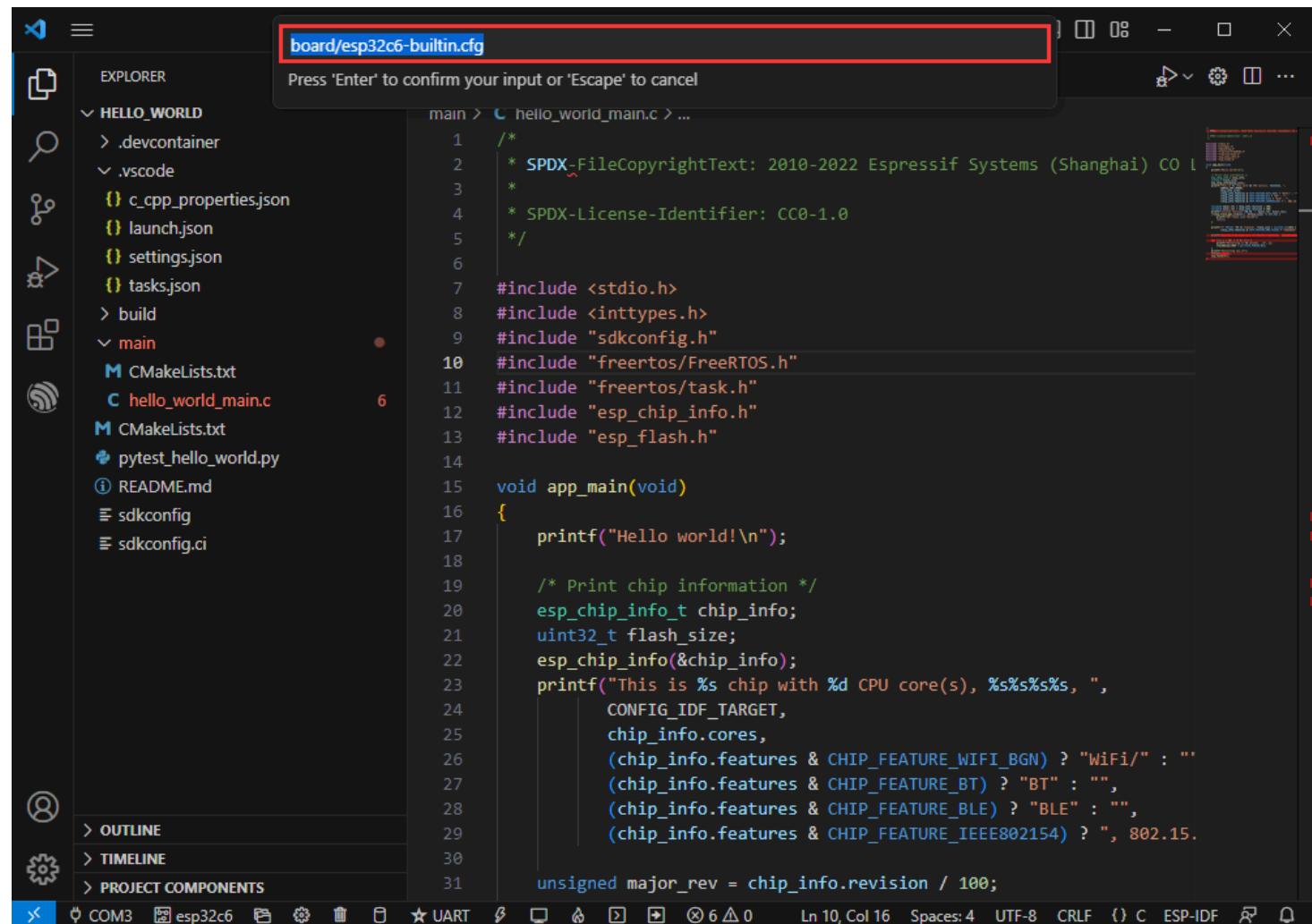
4 * SPDX-License-Identifier: CC0-1.0
5 */
6
7 #include <stdio.h>
8 #include <inttypes.h>
9 #include "sdkconfig.h"
10 #include "freertos/FreeRTOS.h"
11 #include "freertos/task.h"
12 #include "esp_chip_info.h"
13 #include "esp_flash.h"
14
15 void app_main(void)
16 {
17     printf("Hello world!\n");
18
19     /* Print chip information */
20     esp_chip_info_t chip_info;
21     uint32_t flash_size;
22     esp_chip_info(&chip_info);
23     printf("This is %s chip with %d CPU core(s), %s%s%s%s, ",
24           CONFIG_IDF_TARGET,
25           chip_info.cores,
26           (chip_info.features & CHIP_FEATURE_WIFI_BGN) ? "WiFi/" : "",
27           (chip_info.features & CHIP_FEATURE_BT) ? "BT" : "",
28           (chip_info.features & CHIP_FEATURE_BLE) ? "BLE" : "",
29           (chip_info.features & CHIP_FEATURE_IEEE802154) ? "", 802.15.
30
31     unsigned major_rev = chip_info.revision / 100;

```

(/wiki/File:ESP32-C6-DEV-KIT-N8-76.png)

- Type **board/esp32c6-builtin.cfg** (if this is the default, just enter)

board/esp32c6-builtin.cfg



```

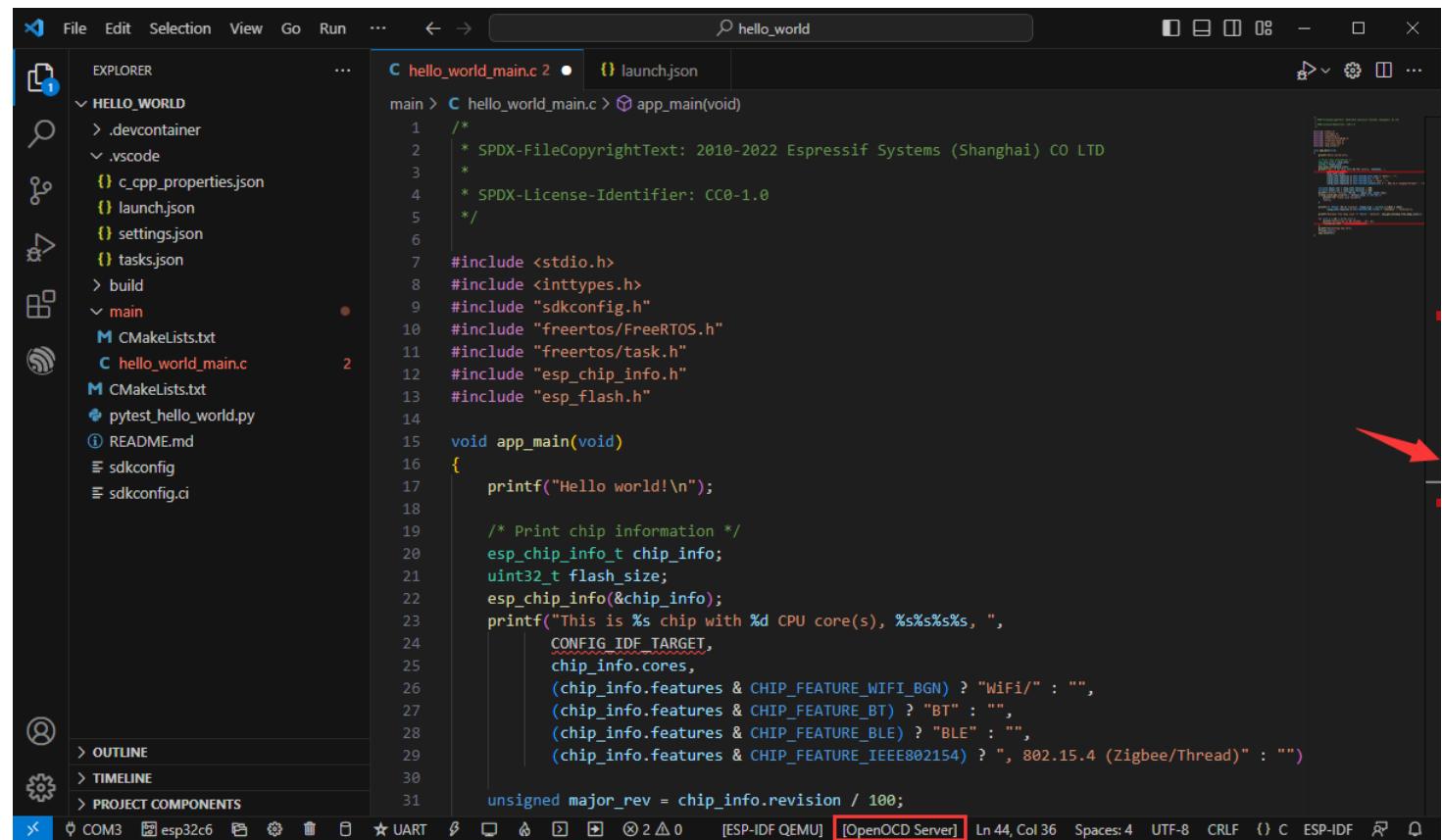
board/esp32c6-builtin.cfg
Press 'Enter' to confirm your input or 'Escape' to cancel

main > hello_world_main.c ...
1  /*
2   * SPDX-FileCopyrightText: 2010-2022 Espressif Systems (Shanghai) CO LTD
3   *
4   * SPDX-License-Identifier: CC0-1.0
5   */
6
7 #include <stdio.h>
8 #include <inttypes.h>
9 #include "sdkconfig.h"
10 #include "freertos/FreeRTOS.h"
11 #include "freertos/task.h"
12 #include "esp_chip_info.h"
13 #include "esp_flash.h"
14
15 void app_main(void)
16 {
17     printf("Hello world!\n");
18
19     /* Print chip information */
20     esp_chip_info_t chip_info;
21     uint32_t flash_size;
22     esp_chip_info(&chip_info);
23     printf("This is %s chip with %d CPU core(s), %s%s%s%s, ",
24           CONFIG_IDF_TARGET,
25           chip_info.cores,
26           (chip_info.features & CHIP_FEATURE_WIFI_BGN) ? "WiFi/" : "",
27           (chip_info.features & CHIP_FEATURE_BT) ? "BT" : "",
28           (chip_info.features & CHIP_FEATURE_BLE) ? "BLE" : "",
29           (chip_info.features & CHIP_FEATURE_IEEE802154) ? ", 802.15",
30
31     unsigned major_rev = chip_info.revision / 100;

```

(/wiki/File:ESP32-C6-DEV-KIT-N8077.png)

- Stretch the width of the window until [OpenOCD Server] is displayed at the bottom.



The screenshot shows the Visual Studio Code interface for an ESP32-C6-Zero project named "HELLO_WORLD". The Explorer sidebar on the left lists files like .devcontainer, .vscode, CMakeLists.txt, and hello_world_main.c. The main editor area displays the "hello_world_main.c" file, which contains the "app_main(void)" function. The bottom toolbar includes buttons for COM3, esp32c6, UART, [ESP-IDF QEMU], [OpenOCD Server] (which is highlighted with a red border), and other developer tools.

```
/*
 * SPDX-FileCopyrightText: 2010-2022 Espressif Systems (Shanghai) CO LTD
 *
 * SPDX-License-Identifier: CC0-1.0
 */
#include <stdio.h>
#include <inttypes.h>
#include "sdkconfig.h"
#include "freertos/FreeRTOS.h"
#include "freertos/task.h"
#include "esp_chip_info.h"
#include "esp_flash.h"

void app_main(void)
{
    printf("Hello world!\n");

    /* Print chip information */
    esp_chip_info_t chip_info;
    uint32_t flash_size;
    esp_chip_info(&chip_info);
    printf("This is %s chip with %d CPU core(s), %s%s%s%s, %s\n",
        CONFIG_IDF_TARGET,
        chip_info.cores,
        (chip_info.features & CHIP_FEATURE_WIFI_BGN) ? "WiFi/" : "",
        (chip_info.features & CHIP_FEATURE_BT) ? "BT" : "",
        (chip_info.features & CHIP_FEATURE_BLE) ? "BLE" : "",
        (chip_info.features & CHIP_FEATURE_IEEE802154) ? ", 802.15.4 (Zigbee/Thread)" : "");
    unsigned major_rev = chip_info.revision / 100;
```

(/wiki/File:ESP32-C6-DEV-KIT-N8078.png)

- Click **[OpenOCD Server]** and select **Start OpenOCD**.

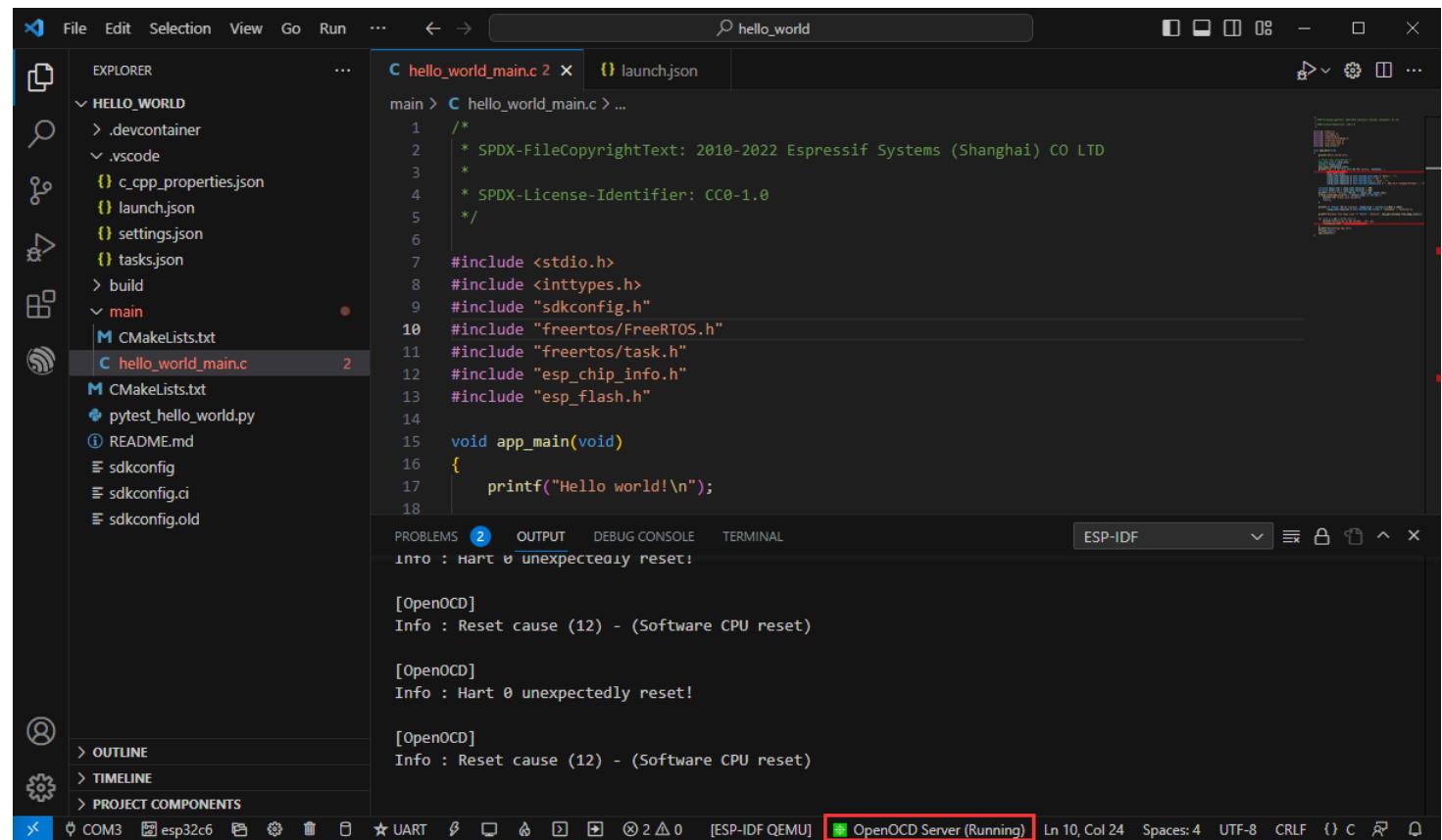
The screenshot shows the Visual Studio Code interface with the following details:

- File Explorer:** Shows the project structure under "HELLO_WORLD".
- Command Palette:** The "Start OpenOCD" command is highlighted with a red box and the number "2".
- Code Editor:** Displays the content of "hello_world_main.c".
- Bottom Status Bar:** Shows connection status (COM3, esp32c6), terminal tabs (QEMU, OpenOCD Server), and file statistics (Ln 44, Col 36, Spaces: 4, UTF-8, CRLF).

```
main > C hello_world_main.c > app_main(void)
1  /*
2   * SPDX-FileCopyrightText: 2010-2022 Espressif Systems (Shanghai) CO LTD
3   *
4   * SPDX-License-Identifier: CC0-1.0
5   */
6
7 #include <stdio.h>
8 #include <inttypes.h>
9 #include "sdkconfig.h"
10 #include "freertos/FreeRTOS.h"
11 #include "freertos/task.h"
12 #include "esp_chip_info.h"
13 #include "esp_flash.h"
14
15 void app_main(void)
16 {
17     printf("Hello world!\n");
18
19     /* Print chip information */
20     esp_chip_info_t chip_info;
21     uint32_t flash_size;
22     esp_chip_info(&chip_info);
23     printf("This is %s chip with %d CPU core(s), %s%s%s%s, ",
24           CONFIG_IDF_TARGET,
25           chip_info.cores,
26           (chip_info.features & CHIP_FEATURE_WIFI_BGN) ? "WiFi/" : "",
27           (chip_info.features & CHIP_FEATURE_BT) ? "BT" : "",
28           (chip_info.features & CHIP_FEATURE_BLE) ? "BLE" : "",
29           (chip_info.features & CHIP_FEATURE_IEEE802154) ? ", 802.15.4 (Zigbee/Thread)" : "")
30
31     unsigned major_rev = chip_info.revision / 100;
```

(/wiki/File:ESP32-C6-DEV-KIT-N8079.png)

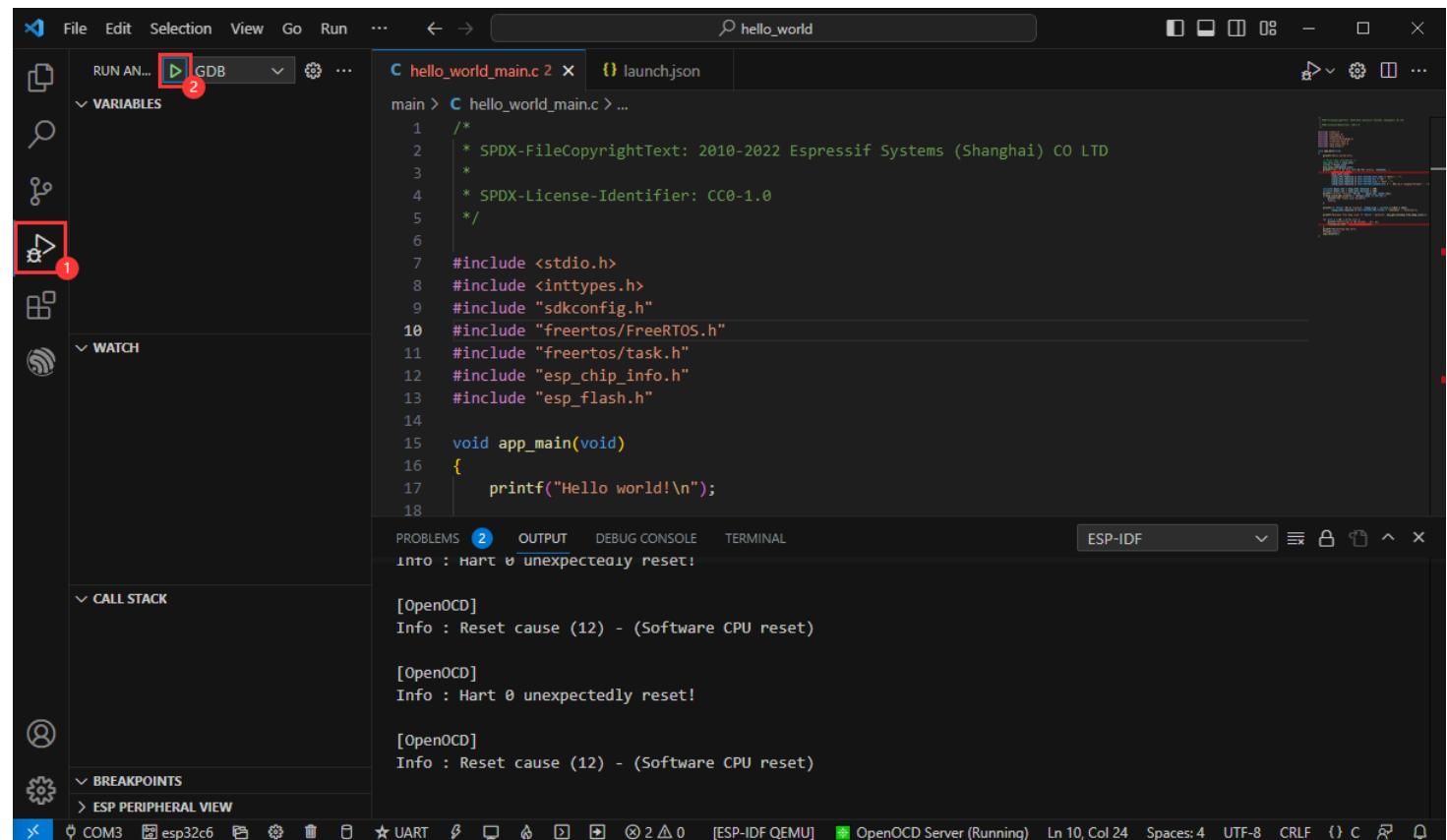
- Successfully opened as follows:



```
/*
 * SPDX-FileCopyrightText: 2010-2022 Espressif Systems (Shanghai) CO LTD
 *
 * SPDX-License-Identifier: CC0-1.0
 */
#include <stdio.h>
#include <inttypes.h>
#include "sdkconfig.h"
#include "freertos/FreeRTOS.h"
#include "freertos/task.h"
#include "esp_chip_info.h"
#include "esp_flash.h"
void app_main(void)
{
    printf("Hello world!\n");
}
```

(/wiki/File:ESP32-C6-DEV-KIT-N8-80.png)

- Go to the debug function and click Debug.



(/wiki/File:ESP32-C6-DEV-KIT-N8-81.png)

- Successfully enter the debugging interface.

The screenshot shows the ESP-IDF IDE interface. The main area displays the C code for 'hello_world_main.c'. The terminal window at the bottom shows the output of the GDB session, including the printf message and some internal GDB commands. The call stack and breakpoints tabs are visible on the left.

```

1 /*
2  * SPDX-FileCopyrightText: 2010-2022 Espressif Systems (Shanghai) CO LTD
3  *
4  * SPDX-License-Identifier: CC0-1.0
5  */
6
7 #include <stdio.h>
8 #include <inttypes.h>
9 #include "sdkconfig.h"
10 #include "freertos/FreeRTOS.h"
11 #include "freertos/task.h"
12 #include "esp_chip_info.h"
13 #include "esp_flash.h"
14
15 void app_main(void)
16 {
17     printf("Hello world!\n");
18 }
```

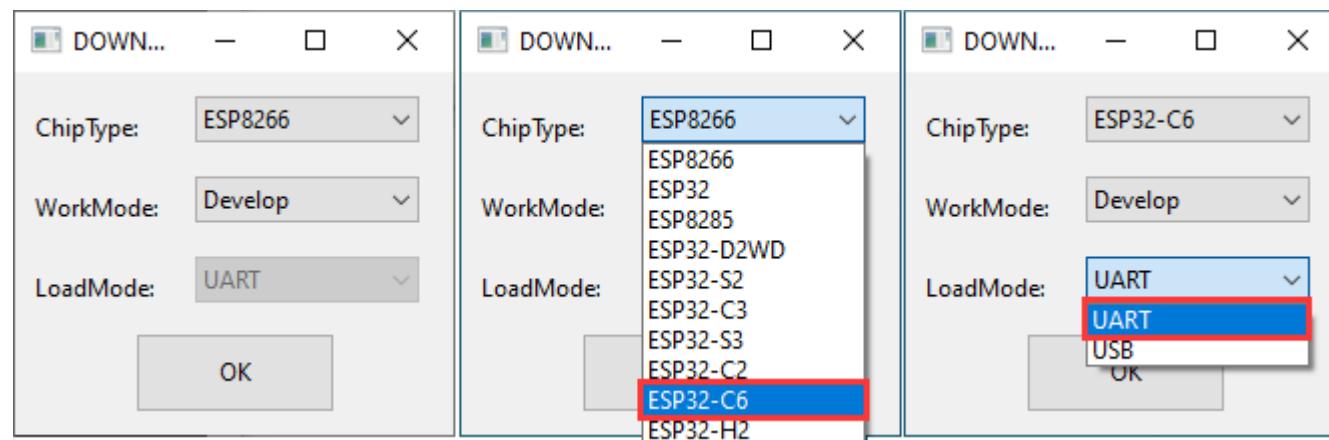
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
Filter (e.g. text, !exclude)

rs/Waveshare/esp/esp-idf/components/freertos/FreeRTOS-Kernel/portable/riscv/port.c", fullname="C:\Users\Waveshare\esp\esp-idf\components\freetos\FreeRTOS-Kernel\portable\riscv\port.c", line="384", arch="riscv:rv32"}, state="stopped"]
1: (27279) ->(gdb)
1: (27279) ->&"\n"
1: (27279) ->^done
1: (27280) ->(gdb)
1: (27287) 1019: elapsed time 77
1: (27287) 1020: elapsed time 77
1: (27288) <-1021-stack-list-frames 0 1000

(/wiki/File:ESP32-C6-DEV-KIT-N8-82.png)

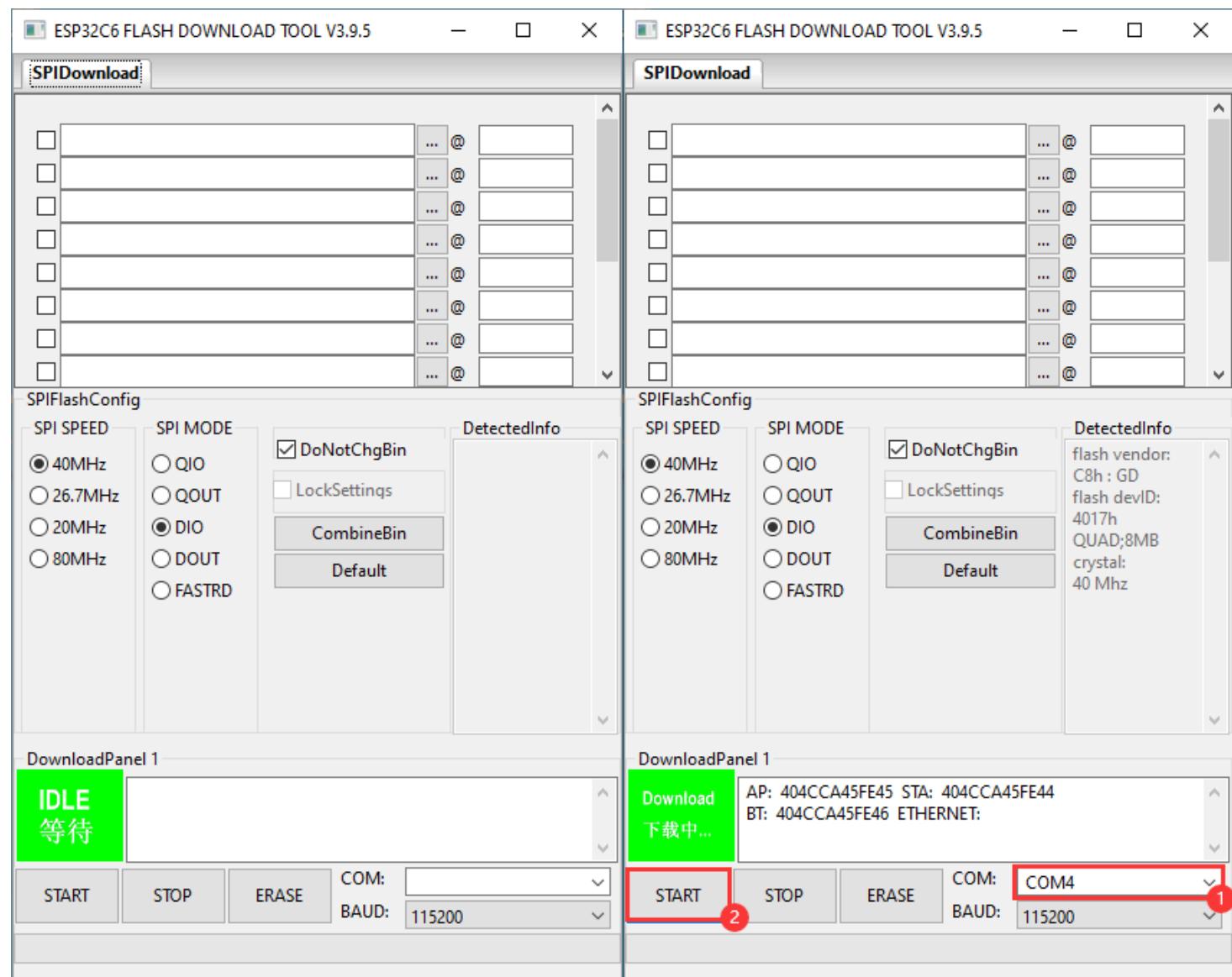
Erase Device Flash

- Unpack the software resource package (Flash debugging software (https://files.waveshare.com/wiki/ESP32-C6-DEV-KIT-N8/Flash_download_tool_3.9.5_0.zip)).
- Open **flash_download_tool_3.9.5.exe** software, select ESP32-C6 and UART.



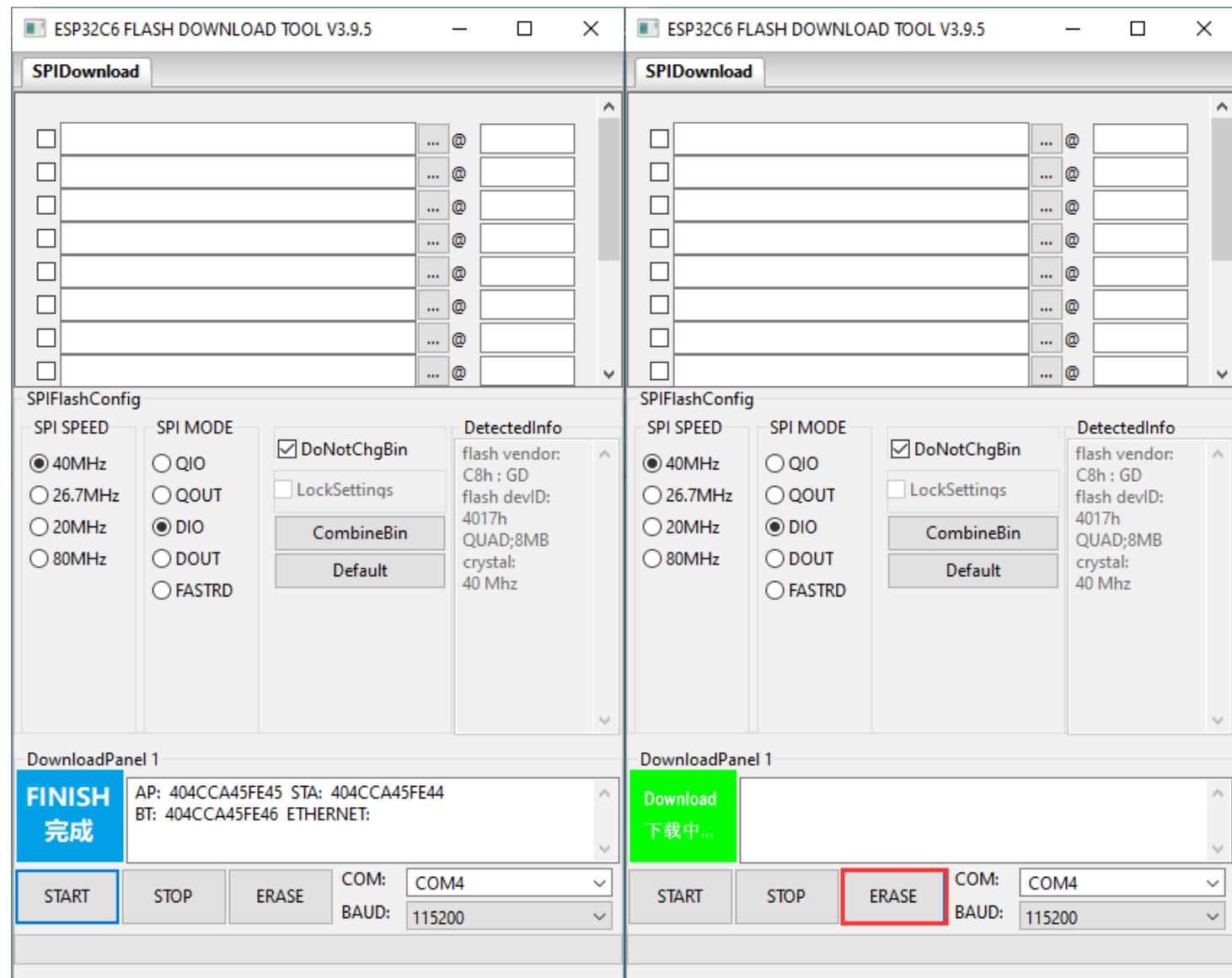
(/wiki/File:ESP32-C6-DEV-KIT-N8-83.png)

- Select the UART port number, and click **START** (not select any bin file).



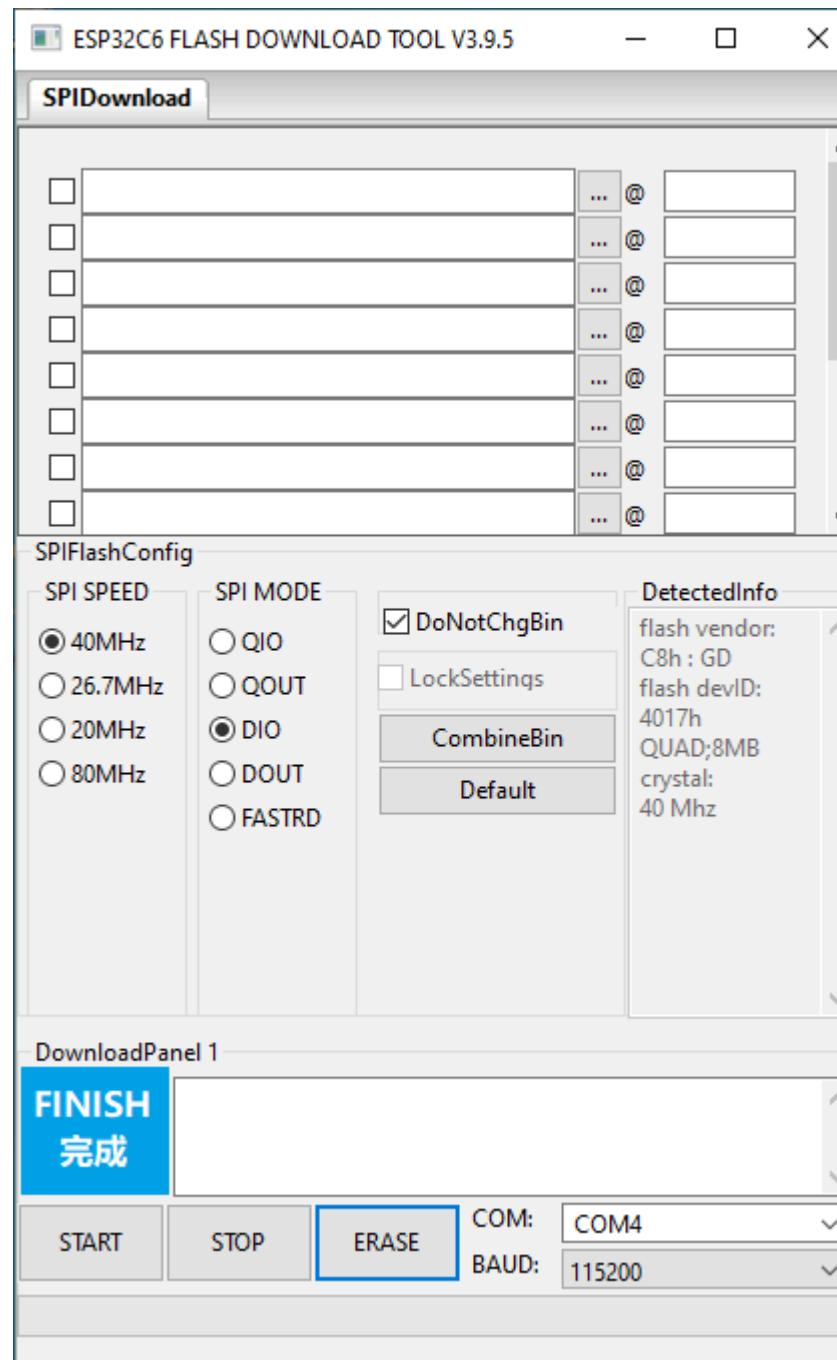
(/wiki/File:ESP32-C6-DEV-KIT-N8-84.png)

- After programming, click on "ERASE".



(/wiki/File:ESP32-C6-DEV-KIT-N8-85.png)

- Waiting for Erase to Finish.



86.png)

(/wiki/File:ESP32-C6-DEV-KIT-N8-

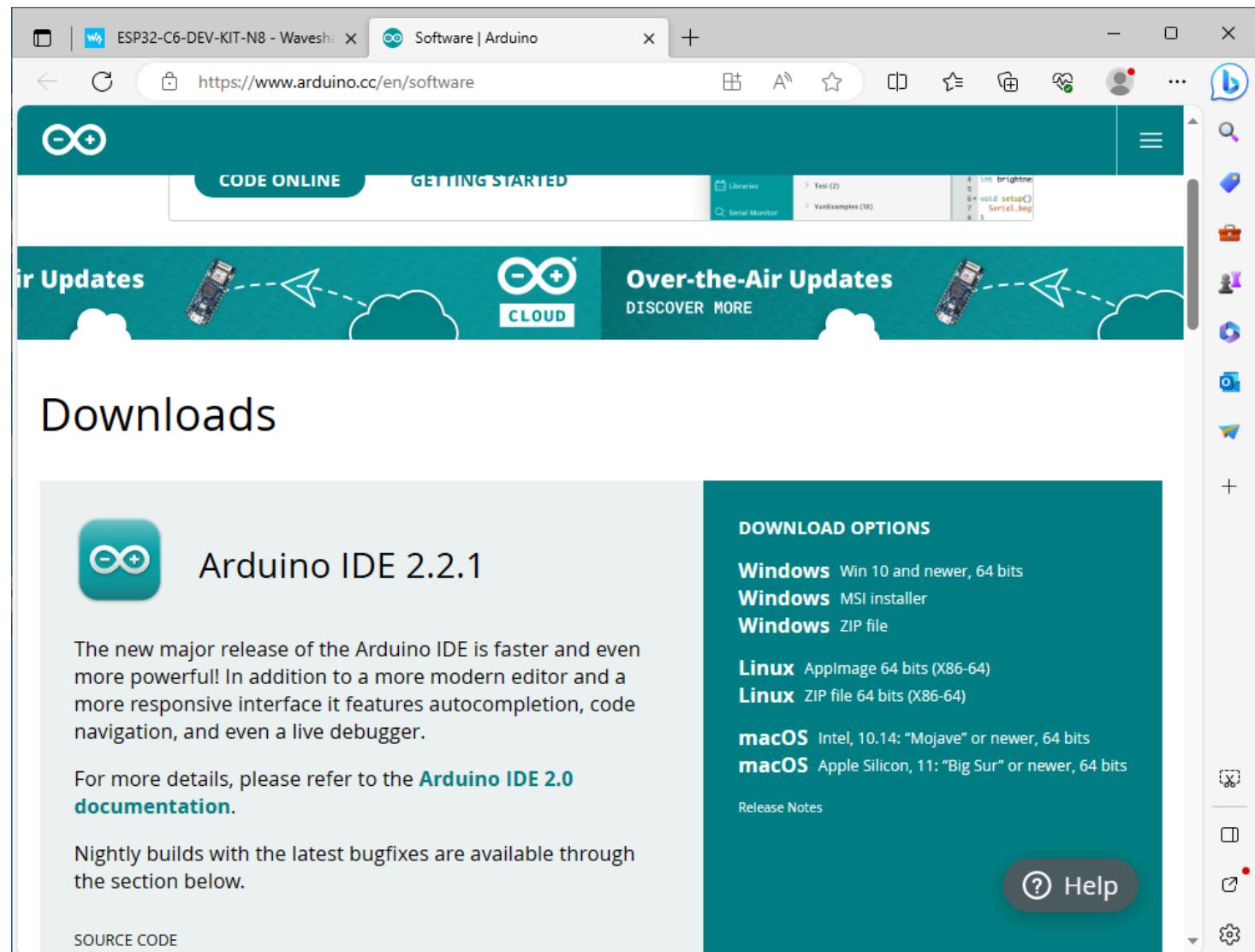
Working with Arduino

Please note that Arduino 3.0.0-alpha is based on ESP-IDF v5.1, which is quite different from the previous ESP-IDF V4.X. The original program may need to be adjusted after the following operations.

- Please do not use the computer username in Chinese as it will cause compilation errors.

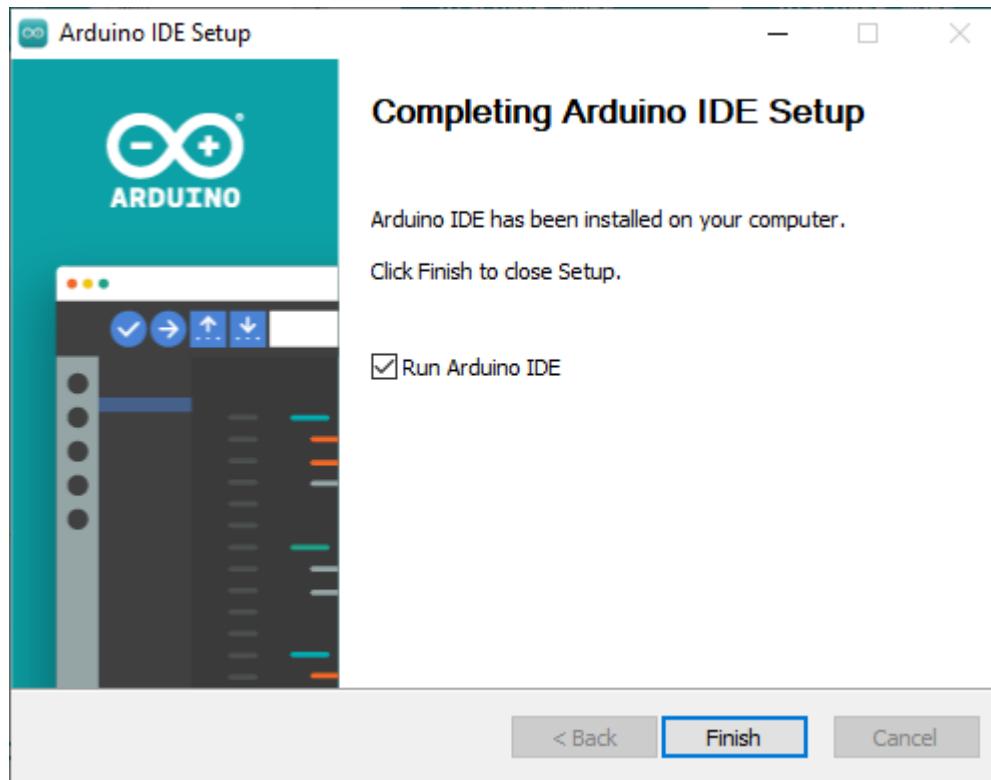
Environment Set-up

- Install Arduino IDE (<https://www.arduino.cc/en/software/>).



(/wiki/File:ESP32-C6-DEV-KIT-N8-Arduino01.png)

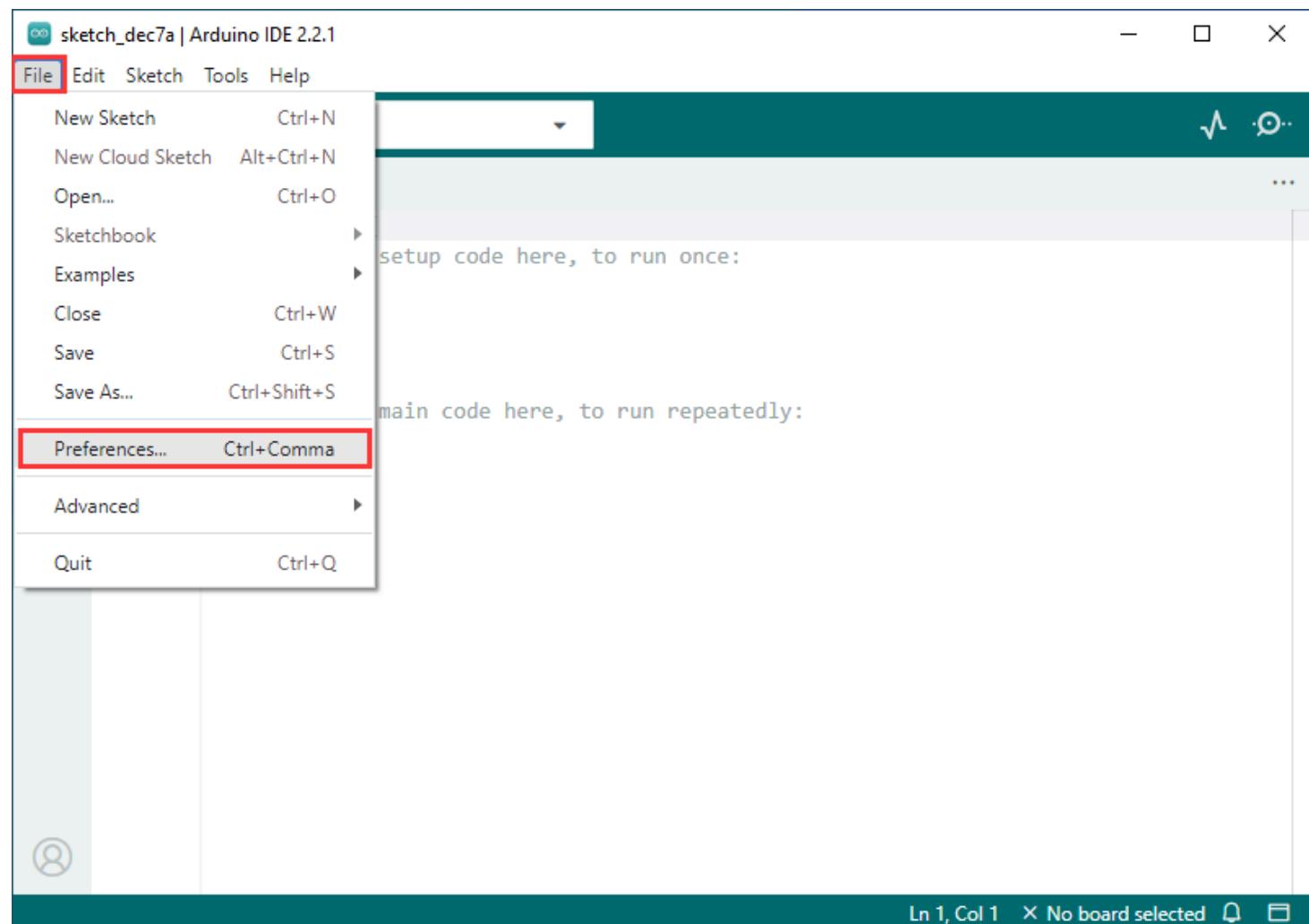
- Enter Arduino IDE after installation.



KIT-N8-Arduino02.png

- Enter Preferences.

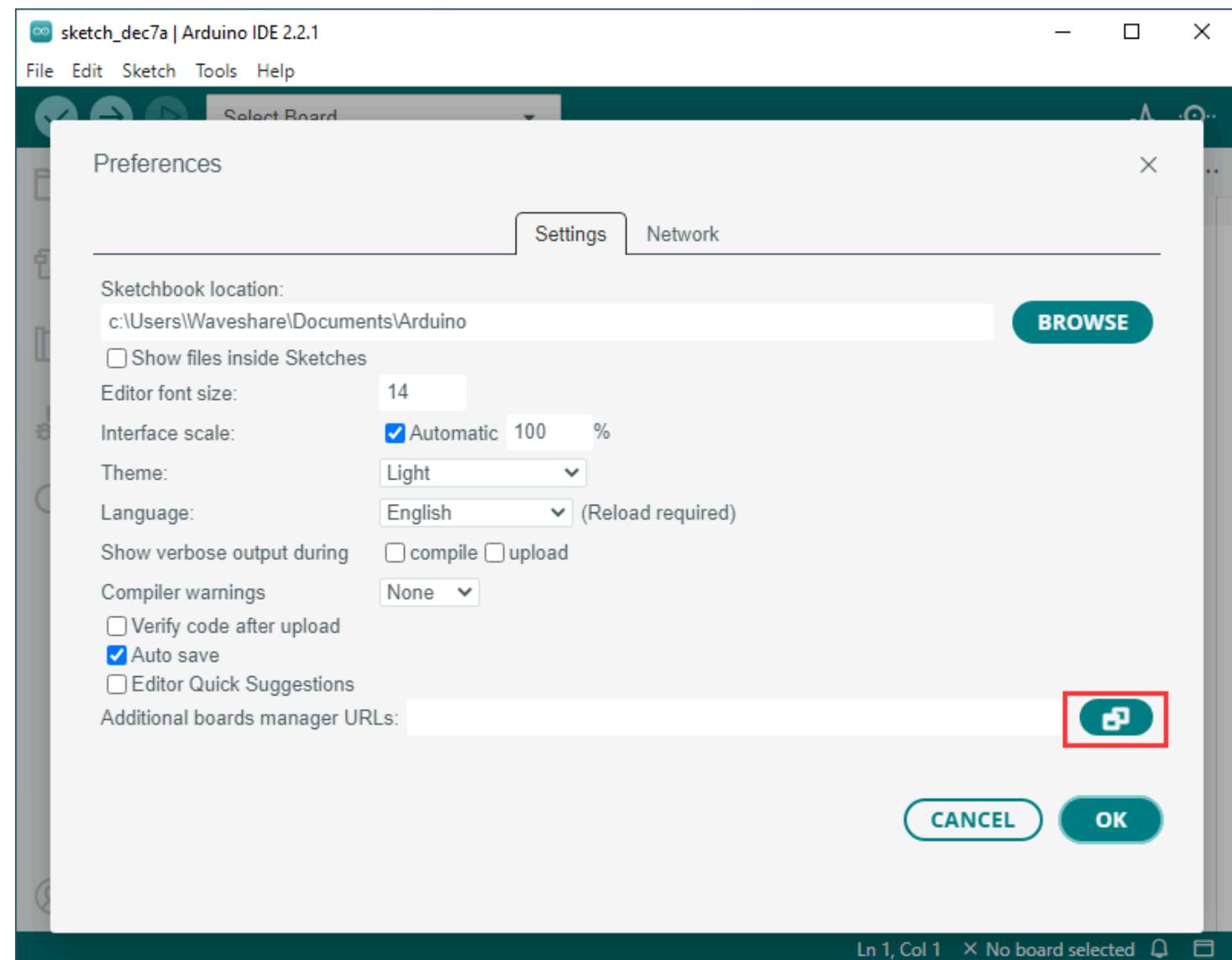
(/wiki/File:ESP32-C6-DEV-



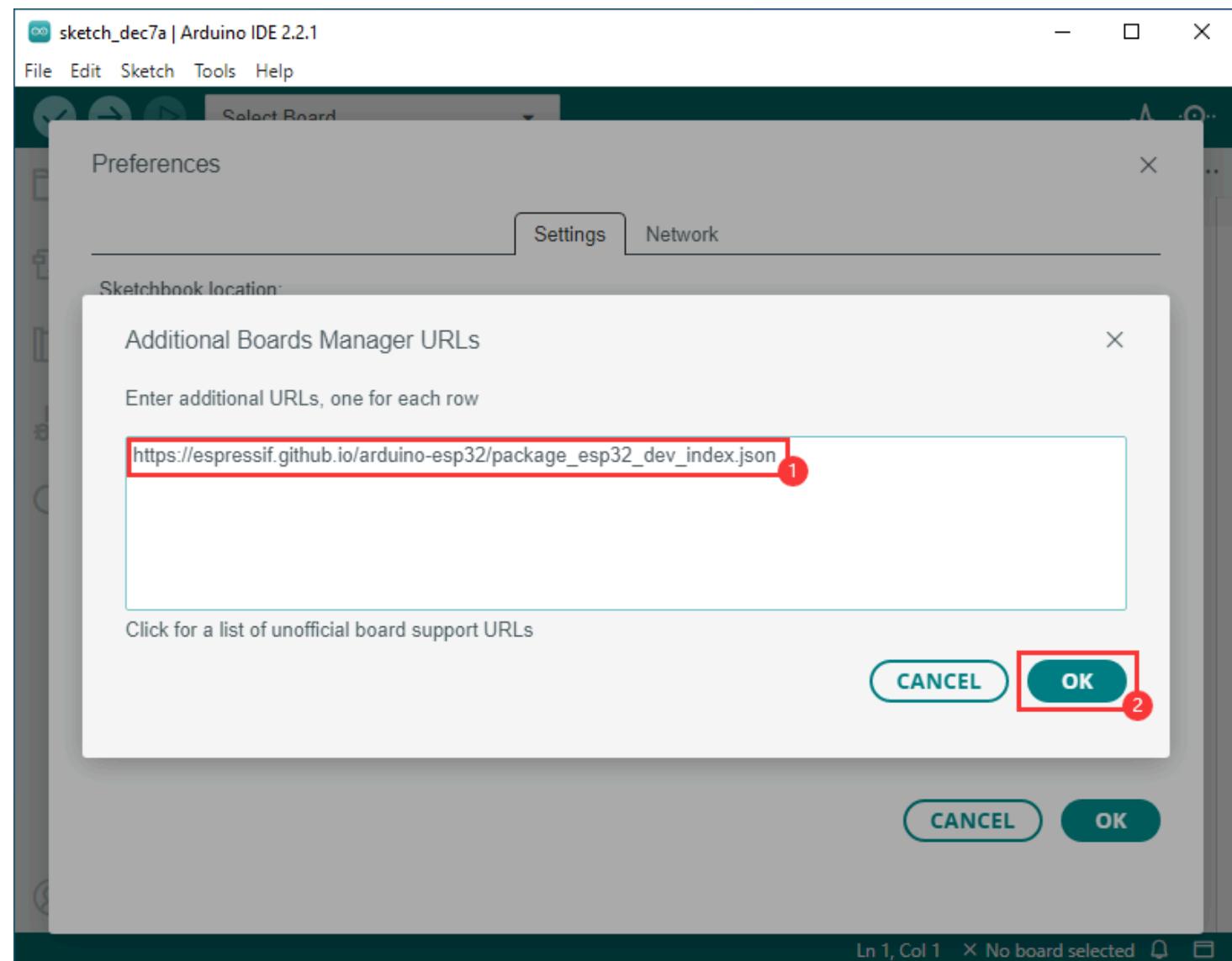
(/wiki/File:ESP32-C6-DEV-KIT-N8-Arduino03.png)

- Add JSON link:

https://espressif.github.io/arduino-esp32/package_esp32_dev_index.json (https://espressif.github.io/arduino-esp32/package_esp32_dev_index.json)

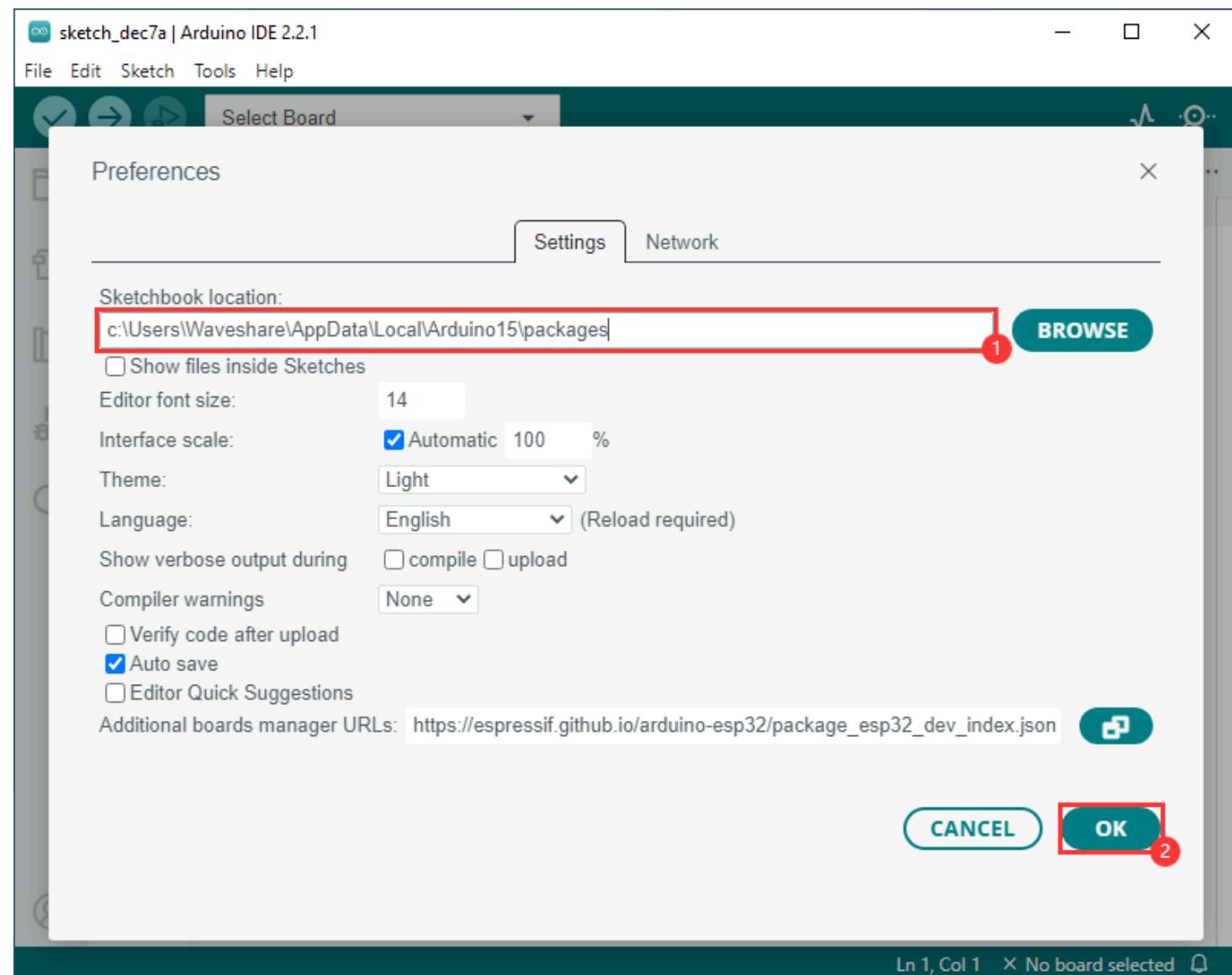


(/wiki/File:ESP32-C6-DEV-KIT-N8-Arduino04.png)



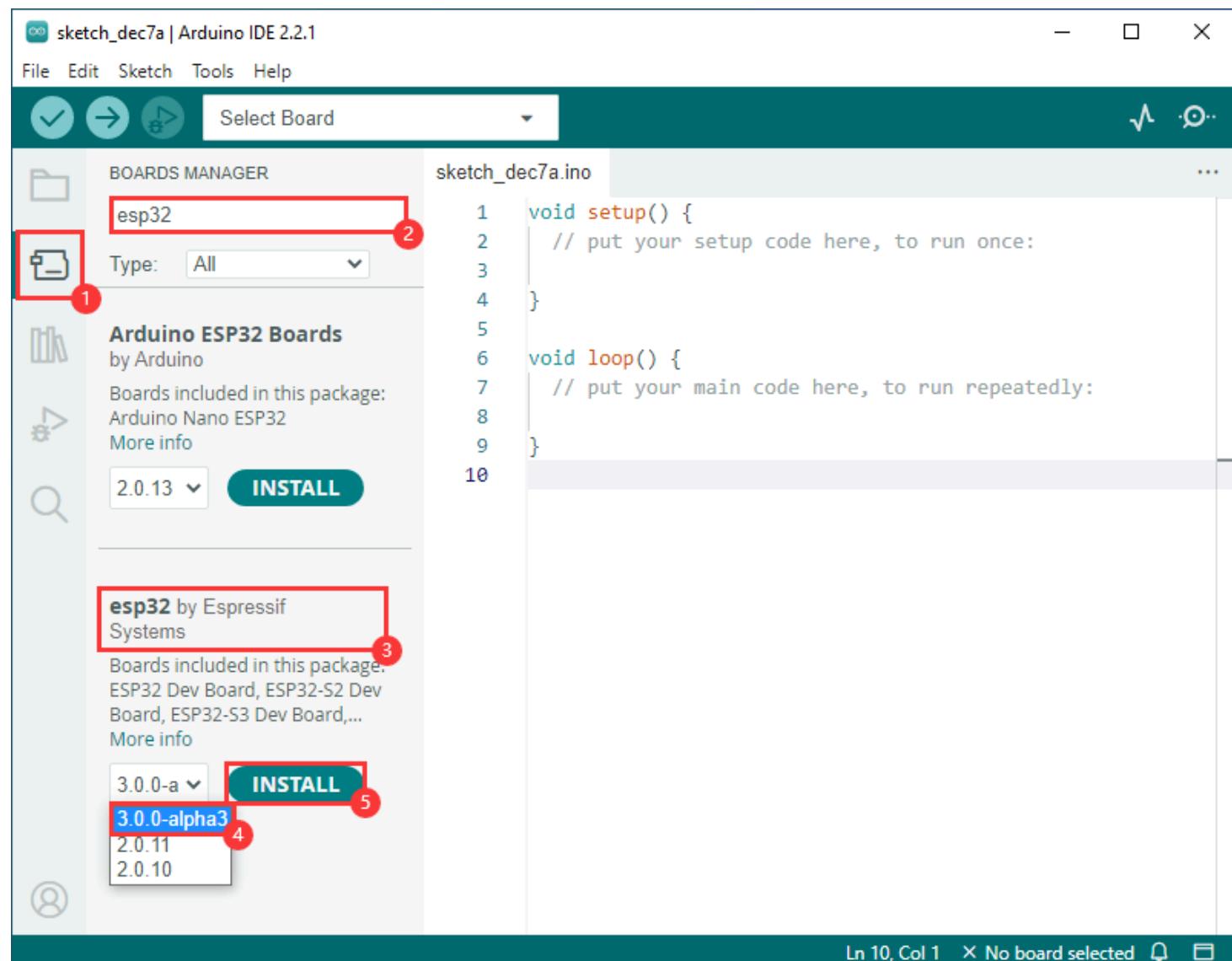
(/wiki/File:ESP32-C6-DEV-KIT-N8-Arduino05.png)

- Modify the project file folder as **C:\Users\Waveshare\AppData\Local\Arduino15\packages** (Waveshare is the username).

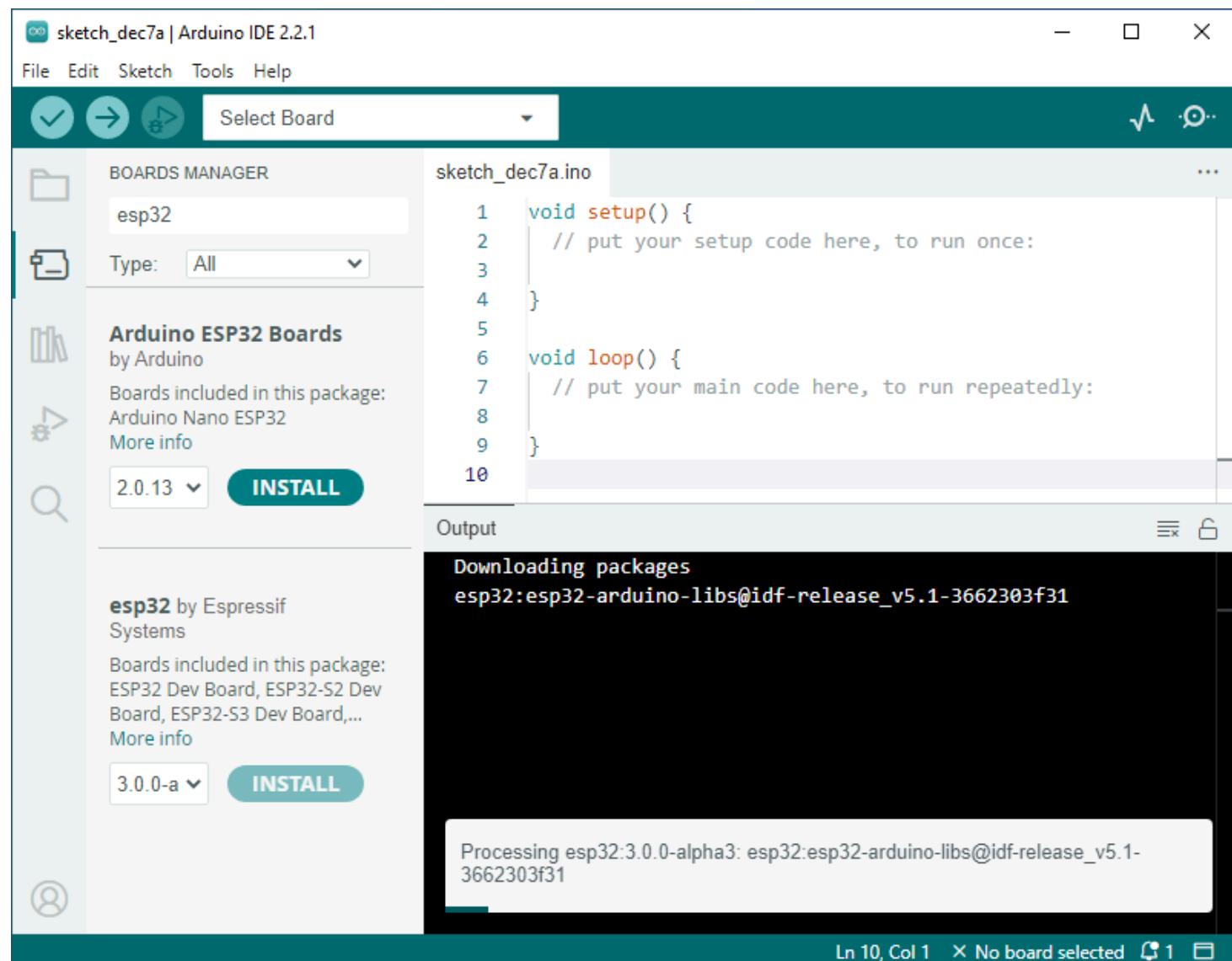


(/wiki/File:ESP32-C6-DEV-KIT-N8-Arduino06.png)

- Enter the development board manager, search for "esp32", select version 3.0.0-alpha3 under "esp32 by Espressif Systems" below, and click to install. (If installation fails, try using a mobile hotspot.)

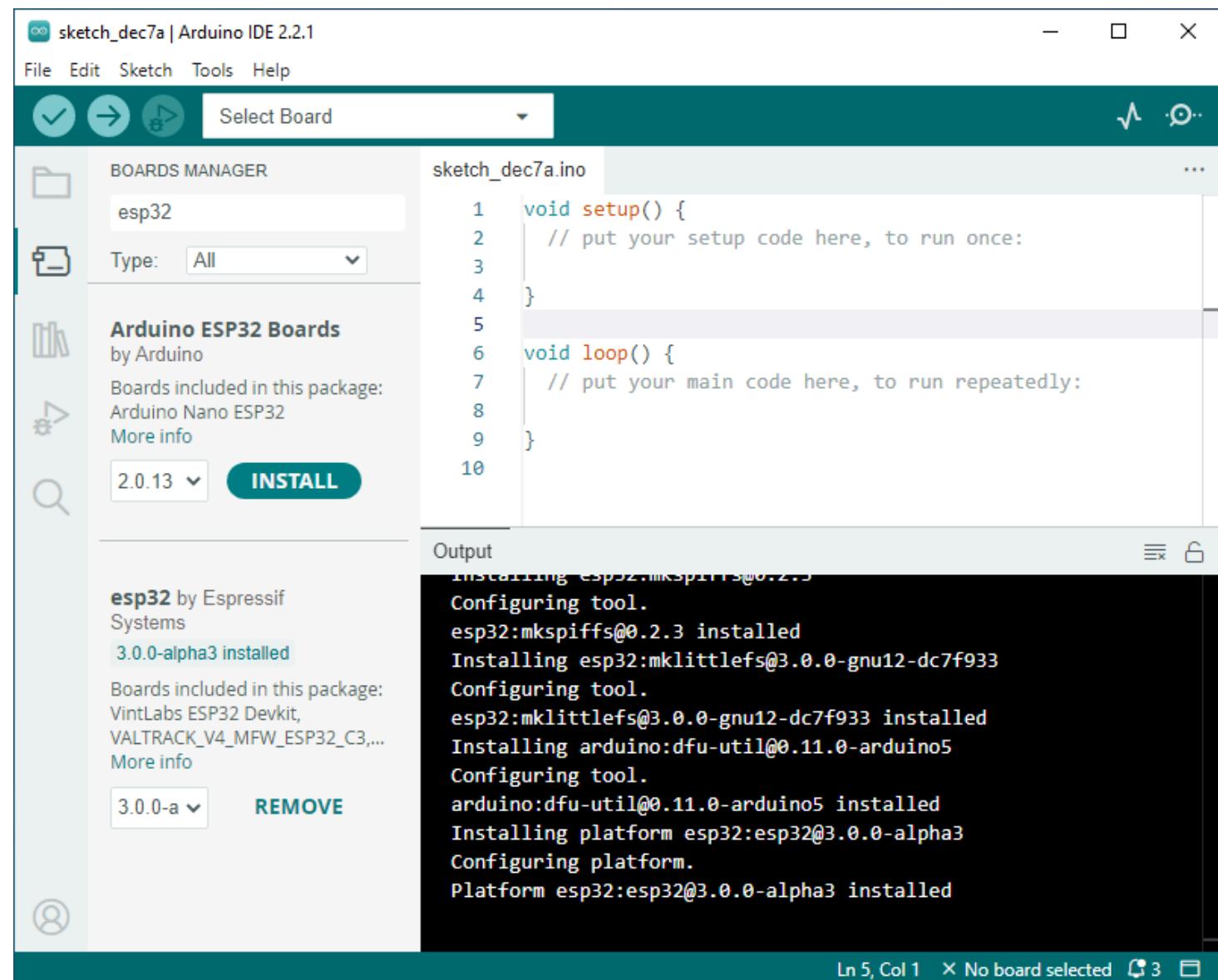


(/wiki/File:ESP32-C6-DEV-KIT-N8-Arduino07.png)



(/wiki/File:ESP32-C6-DEV-KIT-N8-Arduino08.png)

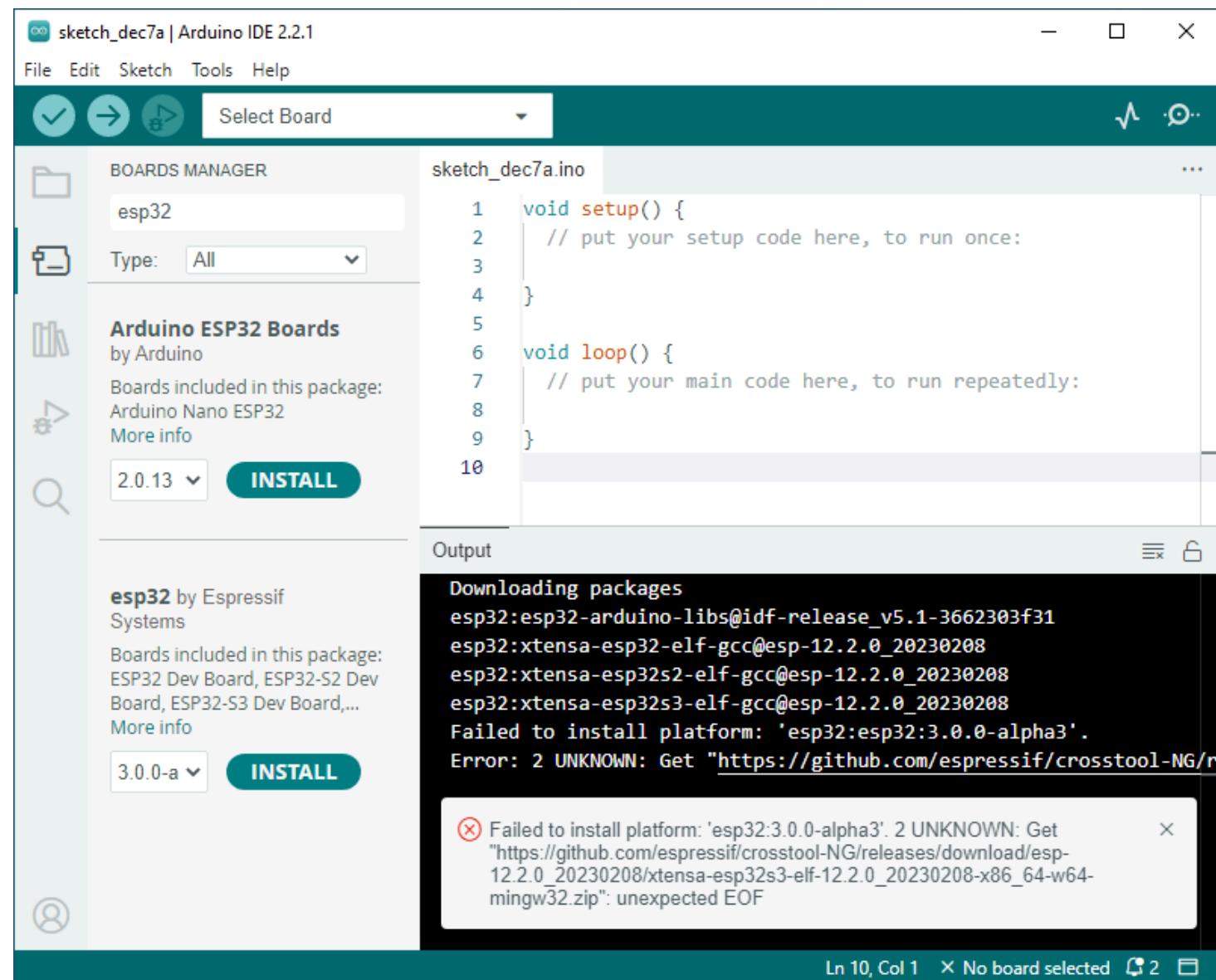
- Restart the Arduino IDE after installation, and then you can use it now.



(/wiki/File:ESP32-C6-DEV-KIT-N8-Arduino09.png)

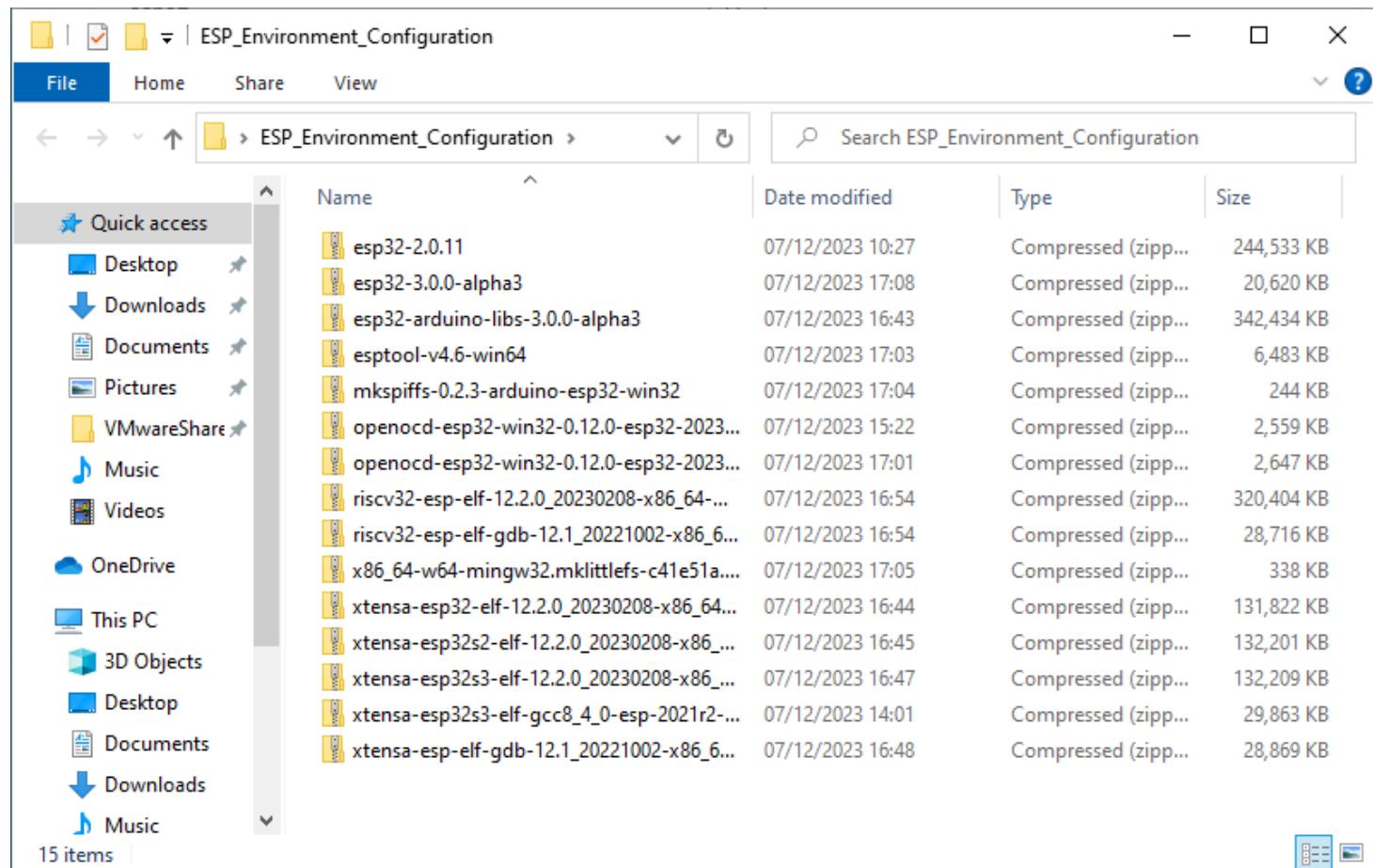
If the Installation Fails

- Failed to install version 3.0.0-alpha3:



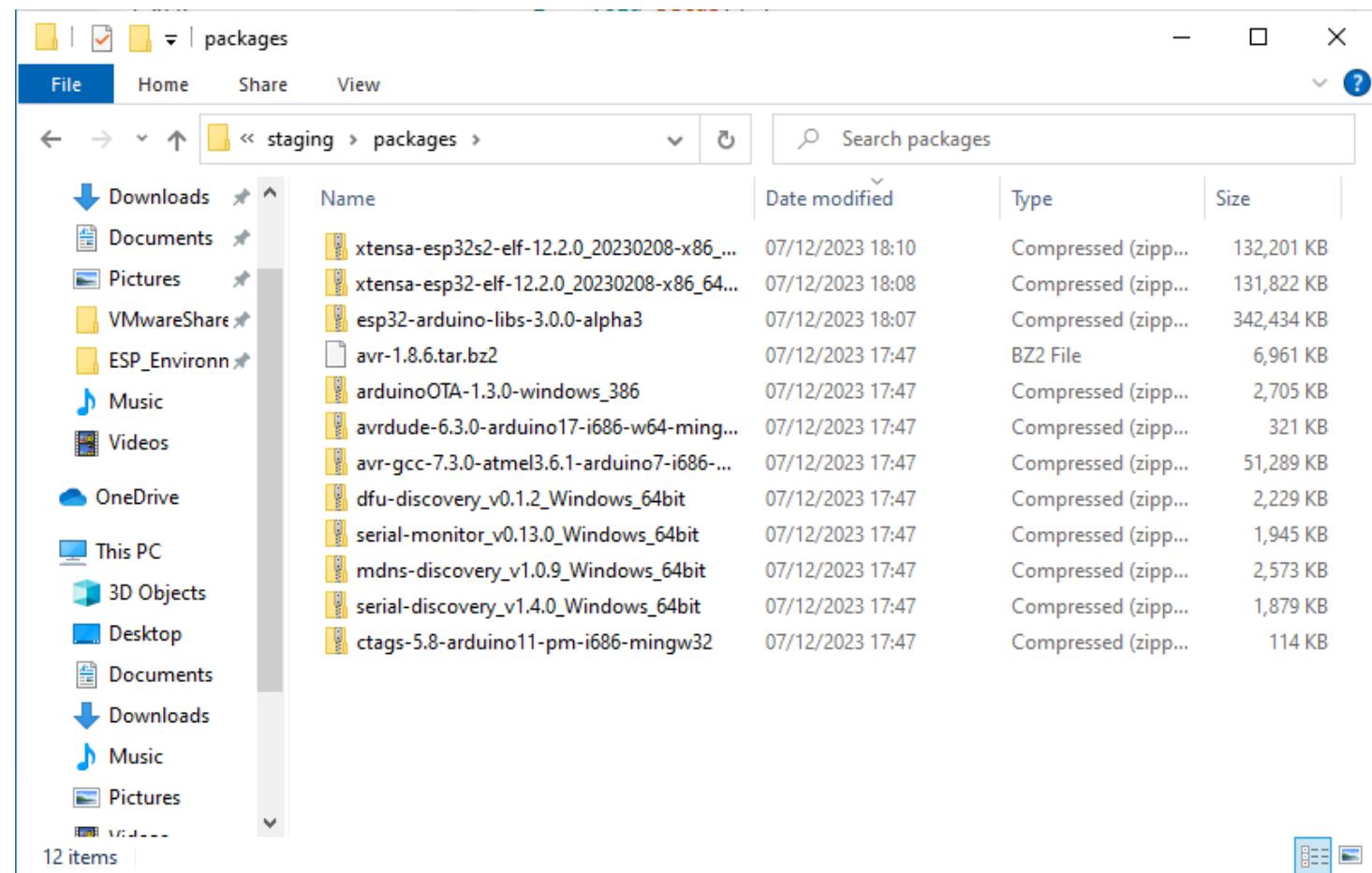
(/wiki/File:ESP32-C6-DEV-KIT-N8-install.png)

- Download the resource file (<https://drive.google.com/file/d/19gMF3WHR4OreN26u2jYtGXKotRn8K16I/view?usp=sharing>).



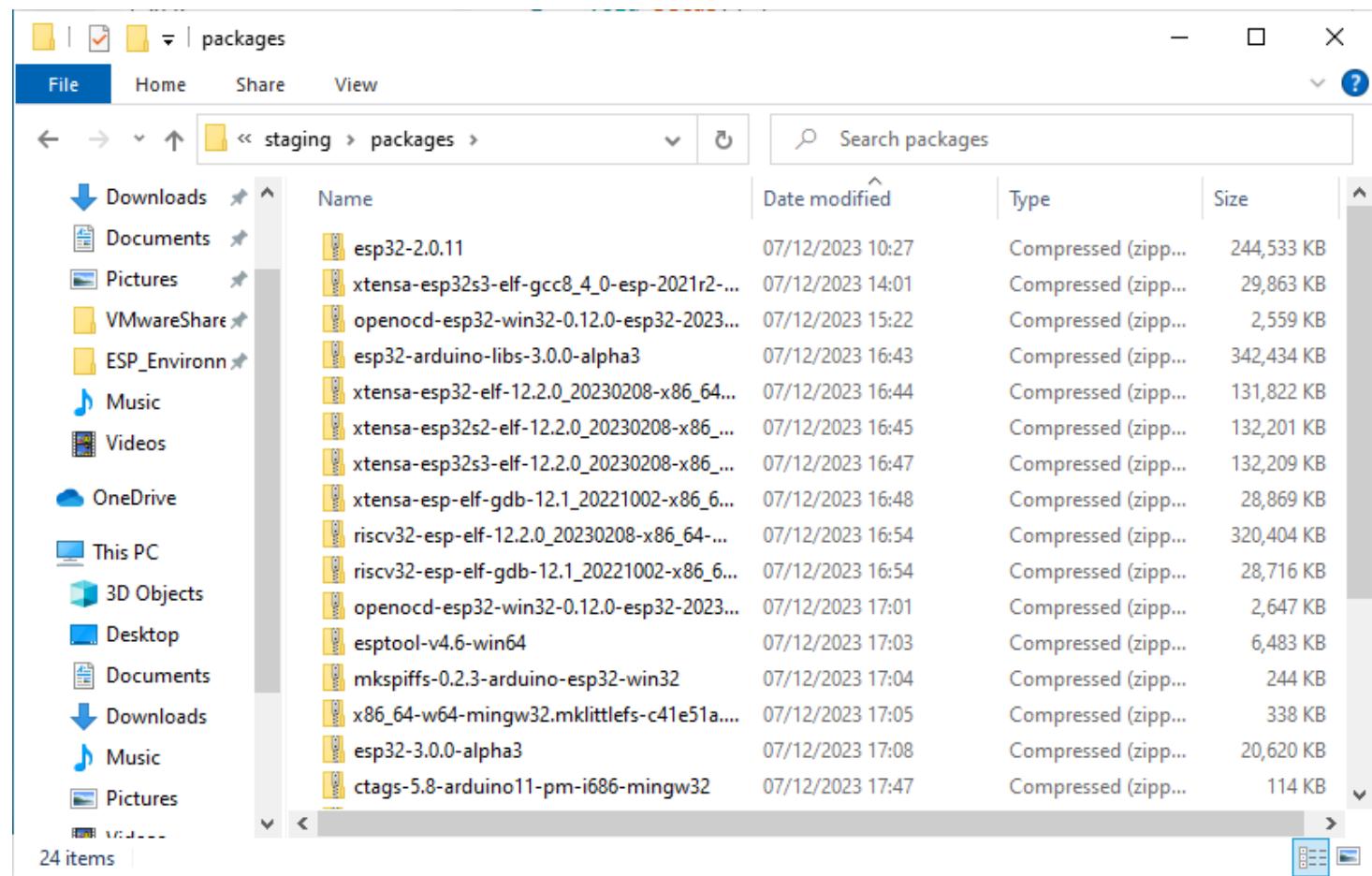
(/wiki/File:ESP32-C6-DEV-KIT-N8-install02.png)

- Click on the path "c:\Users\Waveshare\AppData\Local\Arduino15\packages" (where Waveshare is the user name of the computer, and you need to turn on Show Hidden Files).



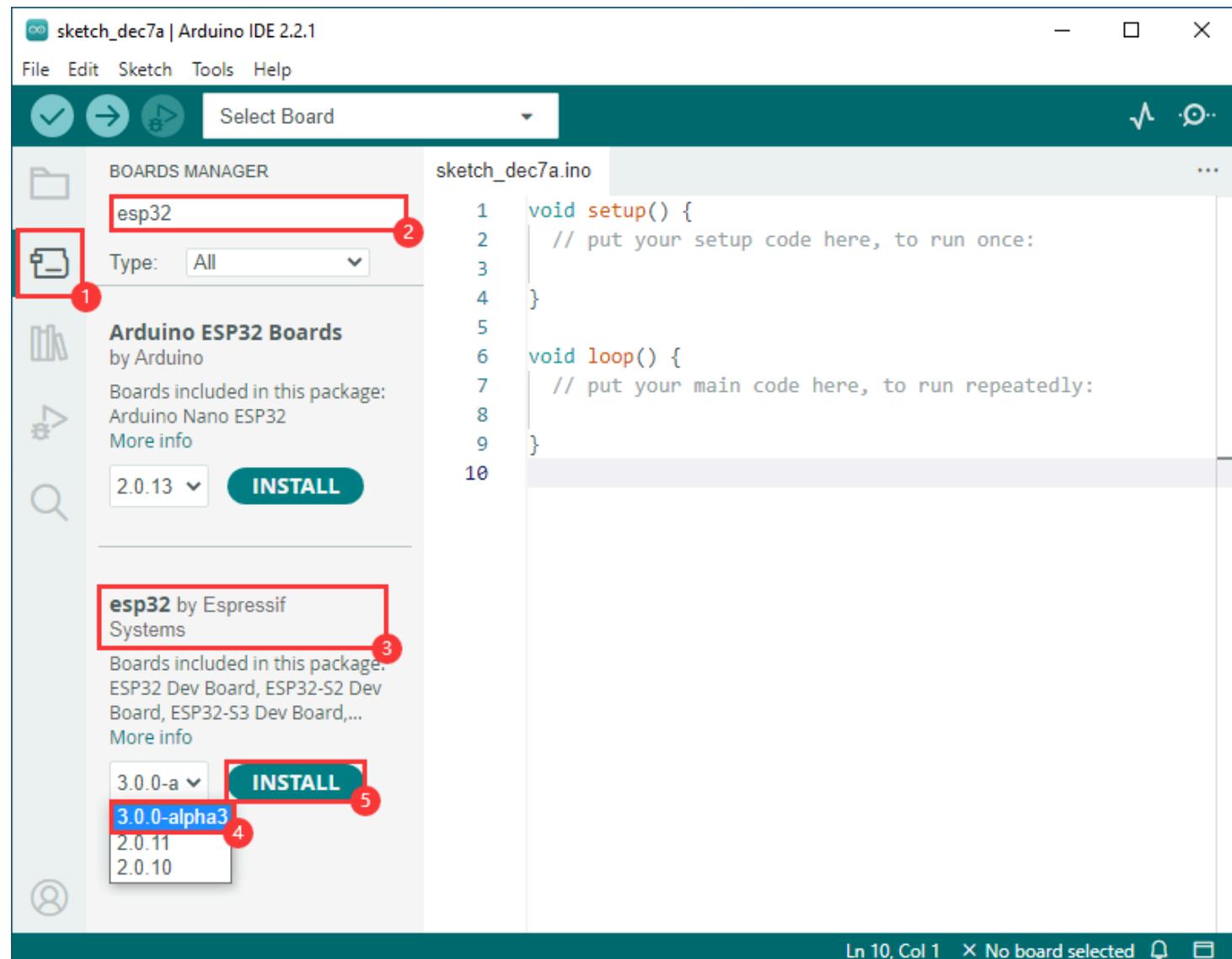
(/wiki/File:ESP32-C6-DEV-KIT-N8-install03.png)

- Unzip the downloaded files to the packages file folder.



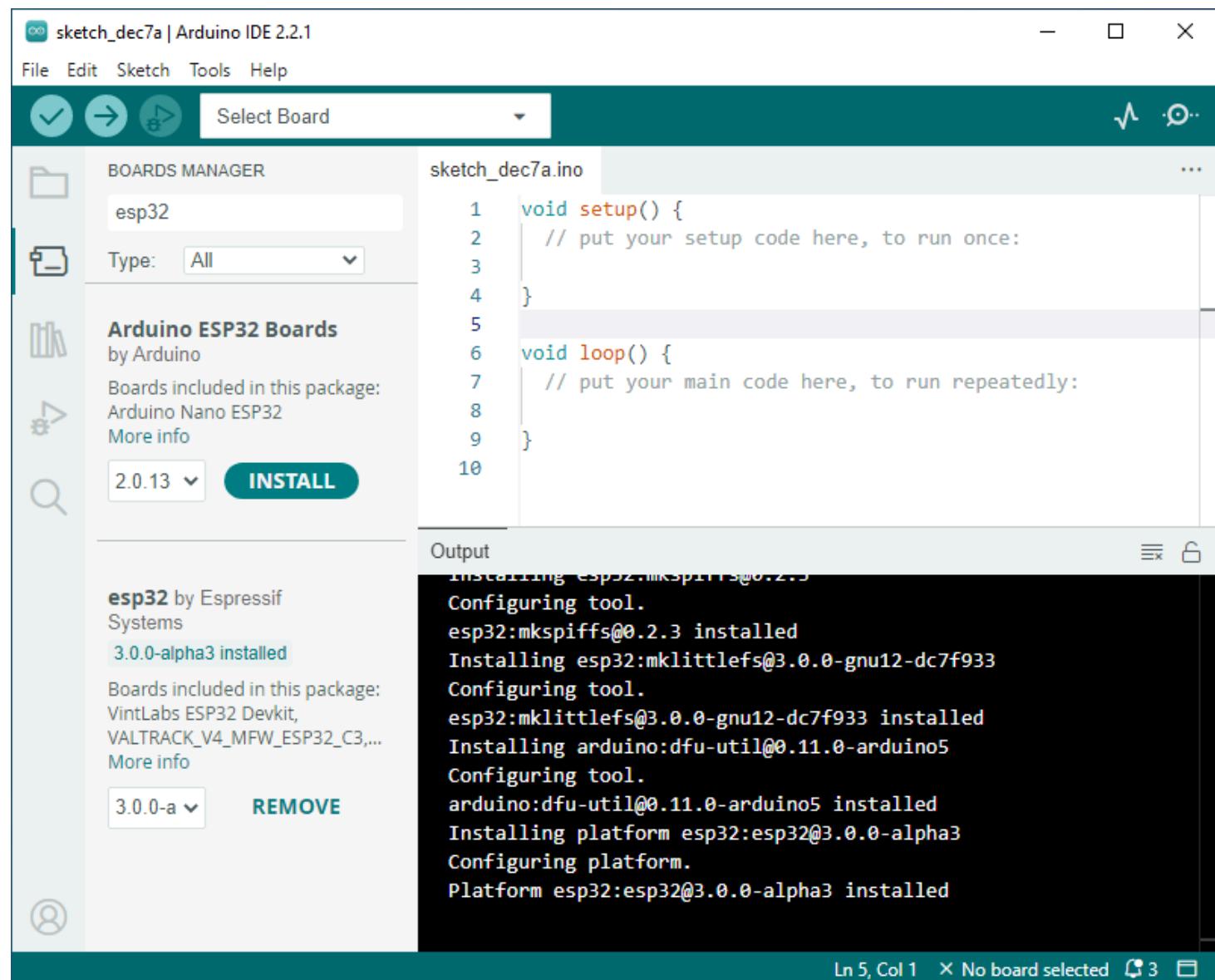
(/wiki/File:ESP32-C6-DEV-KIT-N8-install04.png)

- Install it again.



(/wiki/File:ESP32-C6-DEV-KIT-N8-Arduino07.png)

- Restart the Arduino IDE after installation and you're ready to go!

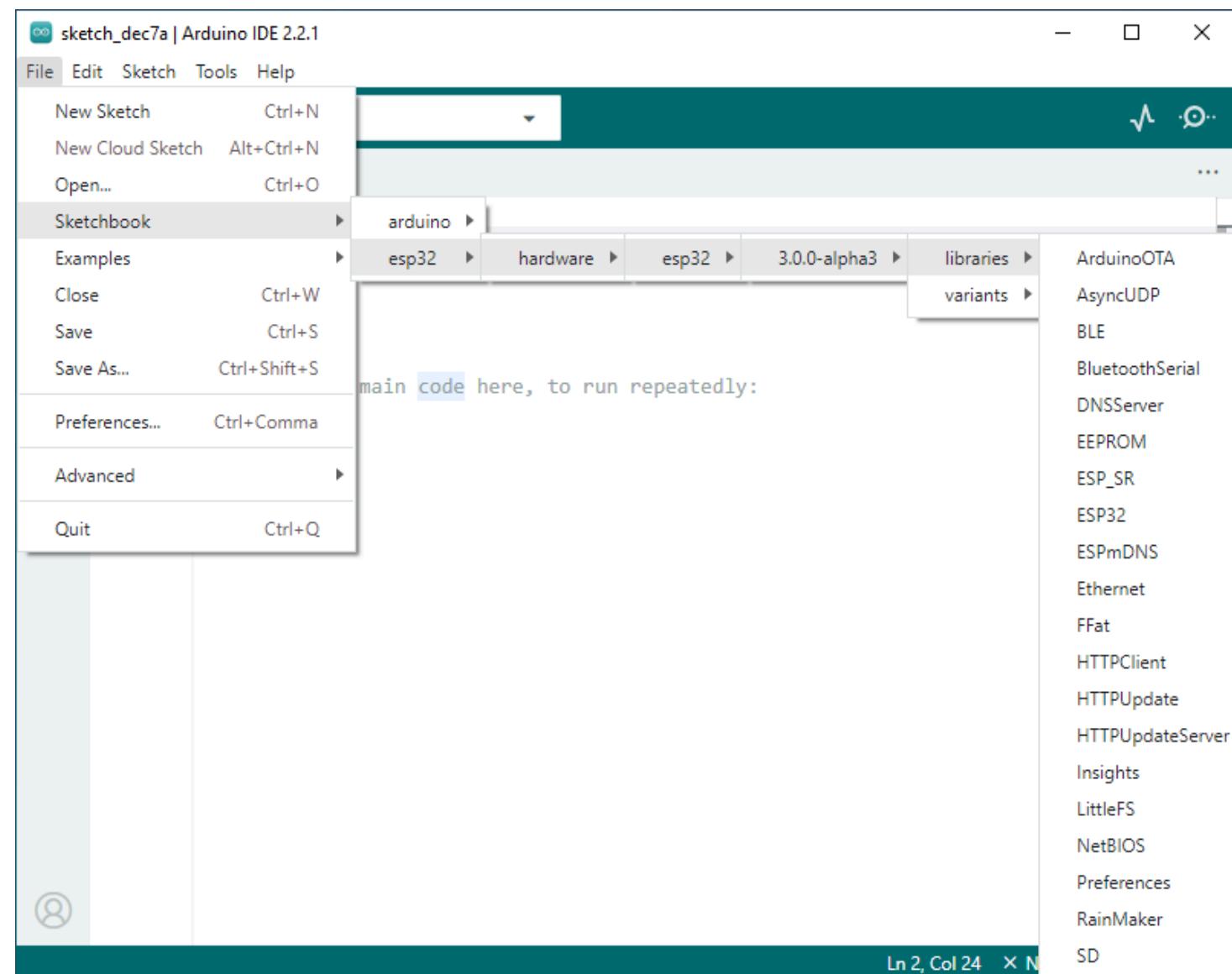


(/wiki/File:ESP32-C6-DEV-KIT-N8-Arduino09.png)

Create Example

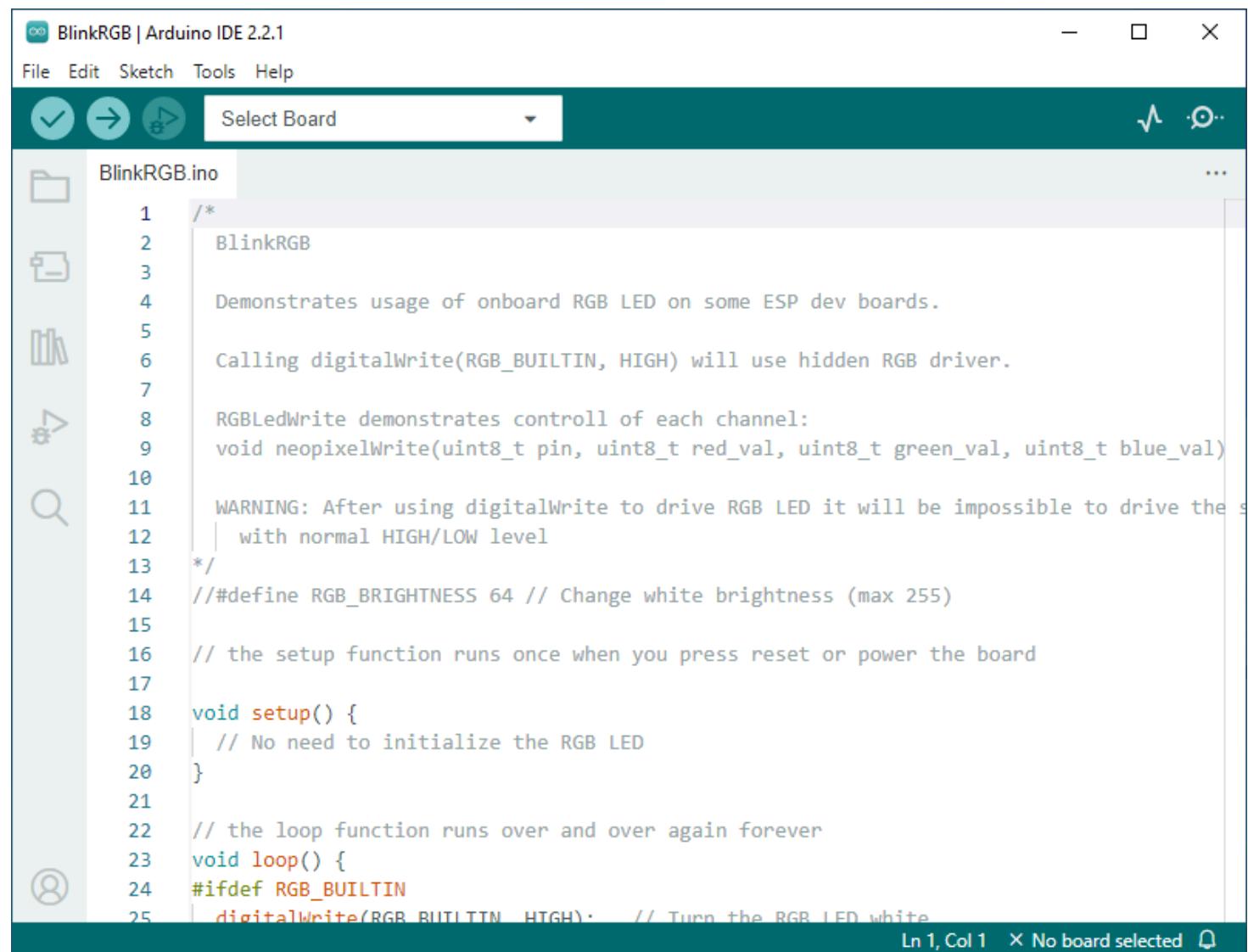
- After changing the project folder above to

c:\Users\Waveshare\AppData\Local\Arduino15\packages ("Waveshare" is the computer username), you can create demos using the examples in the project folder under the files.



(/wiki/File:ESP32-C6-DEV-KIT-N8-Arduino10.png)

- The following is the RGB flashing example (File -> Sketchbook -> esp32 -> hardware -> esp32 -> 3.0.0-alpha3 -> libraries -> ESP32 -> examples -> BlinkRGB under GPIO).



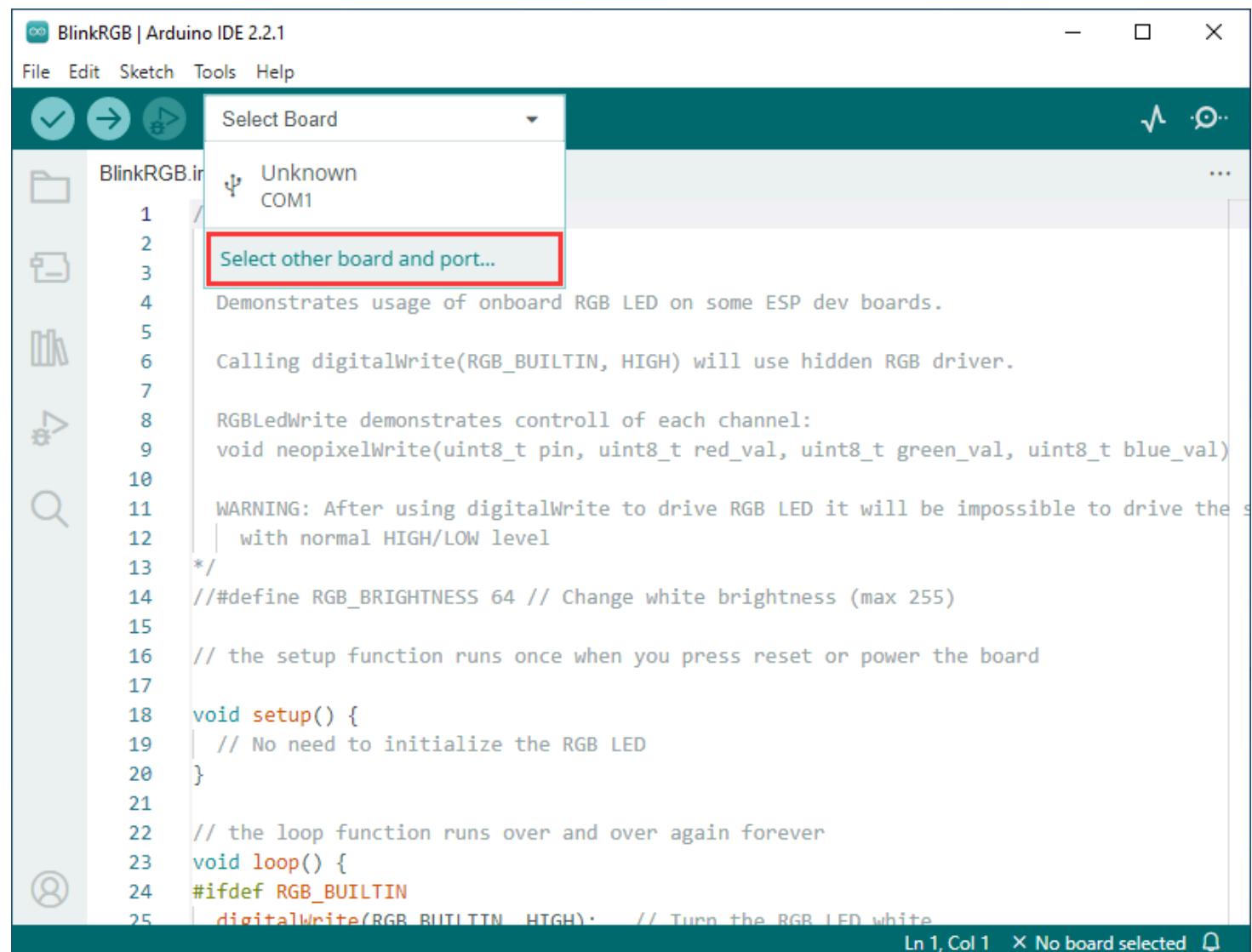
The screenshot shows the Arduino IDE interface with the title bar "BlinkRGB | Arduino IDE 2.2.1" and the subtitle "ESP32-C6-Zero - Waveshare Wiki". The menu bar includes File, Edit, Sketch, Tools, and Help. A toolbar with icons for save, upload, and refresh is visible. The main area shows the code for "BlinkRGB.ino". The code is as follows:

```
1  /*
2   * BlinkRGB
3   *
4   * Demonstrates usage of onboard RGB LED on some ESP dev boards.
5   *
6   * Calling digitalWrite(RGB_BUILTIN, HIGH) will use hidden RGB driver.
7   *
8   * RGBLedWrite demonstrates control of each channel:
9   * void neopixelWrite(uint8_t pin, uint8_t red_val, uint8_t green_val, uint8_t blue_val)
10  *
11  * WARNING: After using digitalWrite to drive RGB LED it will be impossible to drive the
12  *          | with normal HIGH/LOW level
13  */
14 // #define RGB_BRIGHTNESS 64 // Change white brightness (max 255)
15
16 // the setup function runs once when you press reset or power the board
17
18 void setup() {
19     // No need to initialize the RGB LED
20 }
21
22 // the loop function runs over and over again forever
23 void loop() {
24     #ifdef RGB_BUILTIN
25         digitalWrite(RGB_BUILTIN, HIGH); // Turn the RGB LED white
26     #endif
27 }
```

At the bottom right, the status bar displays "Ln 1, Col 1" and "No board selected".

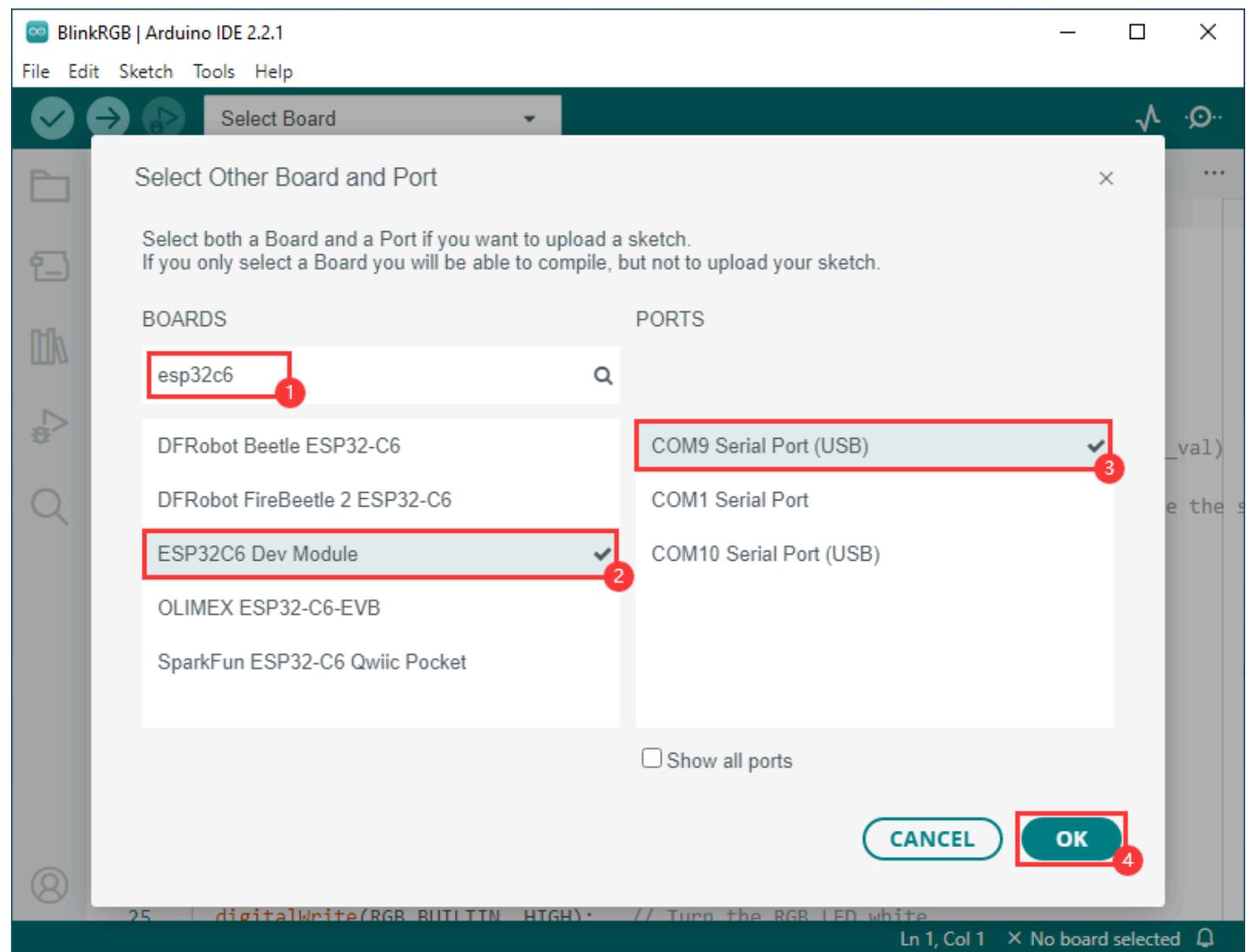
(/wiki/File:ESP32-C6-DEV-KIT-N8-Arduino11.png)

- Select the development board and port.



(/wiki/File:ESP32-C6-DEV-KIT-N8-Arduino12.png)

- Search esp32c6, select ESP32C6 Dev Module and the port to download.



(/wiki/File:ESP32-C6-DEV-KIT-N8-Arduino13.png)

- After selecting, click to upload and Arduino IDE will start to compile and flash the demo.

The screenshot shows the Arduino IDE interface with the title bar "BlinkRGB | Arduino IDE 2.2.1" and the subtitle "ESP32C6 Dev Module". The central area displays the code for "BlinkRGB.ino". The code is a basic example for an RGB LED, demonstrating its usage on some ESP dev boards. It includes comments explaining the setup and loop functions, and a note about using digitalWrite for RGB_BUILTIN pins. The code ends with a conditional compilation directive for RGB_BUILTIN. The status bar at the bottom right indicates "Ln 1, Col 1 ESP32C6 Dev Module on COM9". A red box highlights the upload icon (a right-pointing arrow) in the toolbar.

```
1  /*
2   * BlinkRGB
3   *
4   * Demonstrates usage of onboard RGB LED on some ESP dev boards.
5   *
6   * Calling digitalWrite(RGB_BUILTIN, HIGH) will use hidden RGB driver.
7   *
8   * RGBLedWrite demonstrates control of each channel:
9   * void neopixelWrite(uint8_t pin, uint8_t red_val, uint8_t green_val, uint8_t blue_val)
10  *
11  * WARNING: After using digitalWrite to drive RGB LED it will be impossible to drive the
12  *          | with normal HIGH/LOW level
13  */
14 // #define RGB_BRIGHTNESS 64 // Change white brightness (max 255)
15
16 // the setup function runs once when you press reset or power the board
17
18 void setup() {
19     // No need to initialize the RGB LED
20 }
21
22 // the loop function runs over and over again forever
23 void loop() {
24     #ifdef RGB_BUILTIN
25         digitalWrite(RGB_BUILTIN, HIGH); // Turn the RGB LED white

```

(/wiki/File:ESP32-C6-DEV-KIT-N8-Arduino14.png)

- After uploading, you can see the effect on the development board.

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

- Title Bar:** BlinkRGB | Arduino IDE 2.2.1
- File Menu:** File Edit Sketch Tools Help
- Toolbar:** Checkmark, Upload, Refresh, ESP32C6 Dev Module dropdown, Save, Undo, Redo.
- Code Editor:** Displays the `BlinkRGB.ino` file content. The code demonstrates the usage of an onboard RGB LED on some ESP dev boards, including the ability to control each channel separately using `RGBLedWrite`. It also includes a warning about using `digitalWrite` on the RGB pins after using `RGBLedWrite`, and a note about changing white brightness.
- Output Window:** Shows the upload progress:
 - Writing at 0x0003b059... (87 %)
 - Writing at 0x00041539... (100 %)
 - Wrote 210464 bytes (119505 compressed) at 0x00010000 in 1.1 seconds (effective 1553.0 kbit/s)..
 - Hash of data verified.
- Status Bar:** Ln 25, Col 27 ESP32C6 Dev Module on COM9 4 2

(/wiki/File:ESP32-C6-DEV-KIT-N8-Arduino15.png)

Resources

Software

Compile

- VScode (<https://code.visualstudio.com/download>)
- Arduino IDE (<https://www.arduino.cc/en/software>)

UART

- SSCOM5.13.1 ([https://files.waveshare.com/wiki/LC29H\(XX\)-GPS-RTK-HAT/Sscom5.13.1.zip](https://files.waveshare.com/wiki/LC29H(XX)-GPS-RTK-HAT/Sscom5.13.1.zip))

Flash

- Flash (https://files.waveshare.com/wiki/ESP32-C6-DEV-KIT-N8/Flash_download_tool_3.9.5_0.zip)

Bluetooth

- Bluetooth debugging assistant (https://files.waveshare.com/wiki/ESP32-C6-DEV-KIT-N8/ESP32-C6_TO_BLEAssist.ZIP)

Schematic

- Schematic (<https://files.waveshare.com/wiki/ESP32-C6-Zero/ESP32-C6-Zero-Sch.pdf>)

3D/2D Files

- 3D File (<https://files.waveshare.com/wiki/ESP32-C6-Zero/ESP32-C6-Zero-3D.zip>)

- 2D File (<https://files.waveshare.com/wiki/ESP32-C6-Zero/ESP32-C6-Zero-2D.zip>)

Datasheets

- ESP32-C6 Technical Reference Manual (https://files.waveshare.com/wiki/ESP32-C6-DEV-KIT-N8/ESP32-C6_Technical_Reference_Manual.pdf)
- ESP32-C6 Series Datasheet (https://files.waveshare.com/wiki/ESP32-C6-Pico/ESP32-C6_Series_Datasheet.pdf)

Official Datasheet

- ESP-IDF Datasheet (<https://docs.espressif.com/projects/esp-idf/en/latest/esp32c6/index.html>)

FAQ

Question:The module appears to be reset all the time, and the recognition status will be flashing when viewed from the device manager?

Answer:

This situation may be caused by the Flash blank causing the USB port to be unstable, you can long-press the BOOT button, press RESET at the same time, and then release RESET, and then release the BOOT button, at this time the module can enter the download mode to flash the firmware (program) to solve the situation.

Question:After the module downloads the demo and re-downloads it,

sometimes it fails to connect to the serial port, or the flashing fails?

Answer:

- Method 1: Press the reset button for more than one second, wait for the PC to recognize the device again, and then download it again.
- Method 2: You can long-press the BOOT button, simultaneously press the RESET button, then release the RESET button, and finally release the BOOT button. This will put the module into download mode and can resolve most download issues.

Question:No ESP option below when setting up an environment or building a project?

Answer:

In VSCode, click the shortcut **F1**, and search for **Espressif IDF**, you will find that it is designated as an untrusted extension, set it as trusted.

Question:Switch to the same ESP model and encounter issues with program flashing and program execution?

Answer:

Please select the COM port and driver object again after switching ESP, then compile and flash.

Question:After powering up the module, the recognized serial devices and USB ports keep resetting and restarting?

Answer:

Check whether the power supply voltage for the USB port is less than 5V, in general, if it is 4.9V or more, the module's two USB ports can be used normally. If it is lower than 4.9V, the power supply may be insufficient and the USB port may disconnect. In this case, you should replace it with a USB port with sufficient voltage.

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

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

issue.

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

*Retrieved from "<https://www.waveshare.com/w/index.php?title=ESP32-C6-Zero&oldid=104257>"
(<https://www.waveshare.com/w/index.php?title=ESP32-C6-Zero&oldid=104257>)"*
