

ESP32-C6-DEV-KIT-N8

From Waveshare Wiki

Jump to: navigation, search

Overview

Introduction

The ESP32-C6-DEV-KIT-N8 is a compact microcontroller development board with multiple peripheral interfaces. In terms of hardware, it adopts the ESP32-C6-WROOM-1 module with a RISC-V 32-bit single-core processor. It supports up to 160MHz clock frequency with built-in 320KB ROM, 512KB HP SRAM and 16KB LP SRAM. Onboard CH343 UART and CH334 USB HUB chips, it supports USB and UART development at the same time via a USB-C port. Compatible with the pinout of the ESP32-C6-DevKitC-1-N8 development board, it is more convenient to use and expand a variety of peripheral modules.

In terms of software, you can choose the ESP-IDF

ESP32-C6-DEV-KIT-N8



(<https://www.waveshare.com/esp32-c6-dev-kit-n8.htm?sku=25723>)

development environment or the Arduino IED to develop, so that you can easily and quickly get started and apply it to the product.

Features

- Adopts ESP32-C6-WROOM-1-N8 module with RISC-V 32-bit single-core processor, up to 160MHz main frequency.
- Integrated 320KB ROM, 512KB HP SRAM, 16KB LP SRAM and 8MB Flash memory.
- Integrated 2.4GHz WiFi 6, Bluetooth Low Energy (Bluetooth LE) and IEEE 802.15.4 (Zigbee 3.0 and Thread) wireless communication, with superior RF performance.
- Type-C connector, easier to use.
- Onboard CH343 and CH334 chips can meet the needs of USB and UART development via a Type-C interface.
- Rich peripheral interfaces, compatible with the pinout of the ESP32-C6-DevKitC-1-N8 development board, offer strong compatibility and expandability.
- The castellated module allows soldering directly to carrier boards.

ESP32-C6-DEV-KIT-N8 with Pre-soldered Header

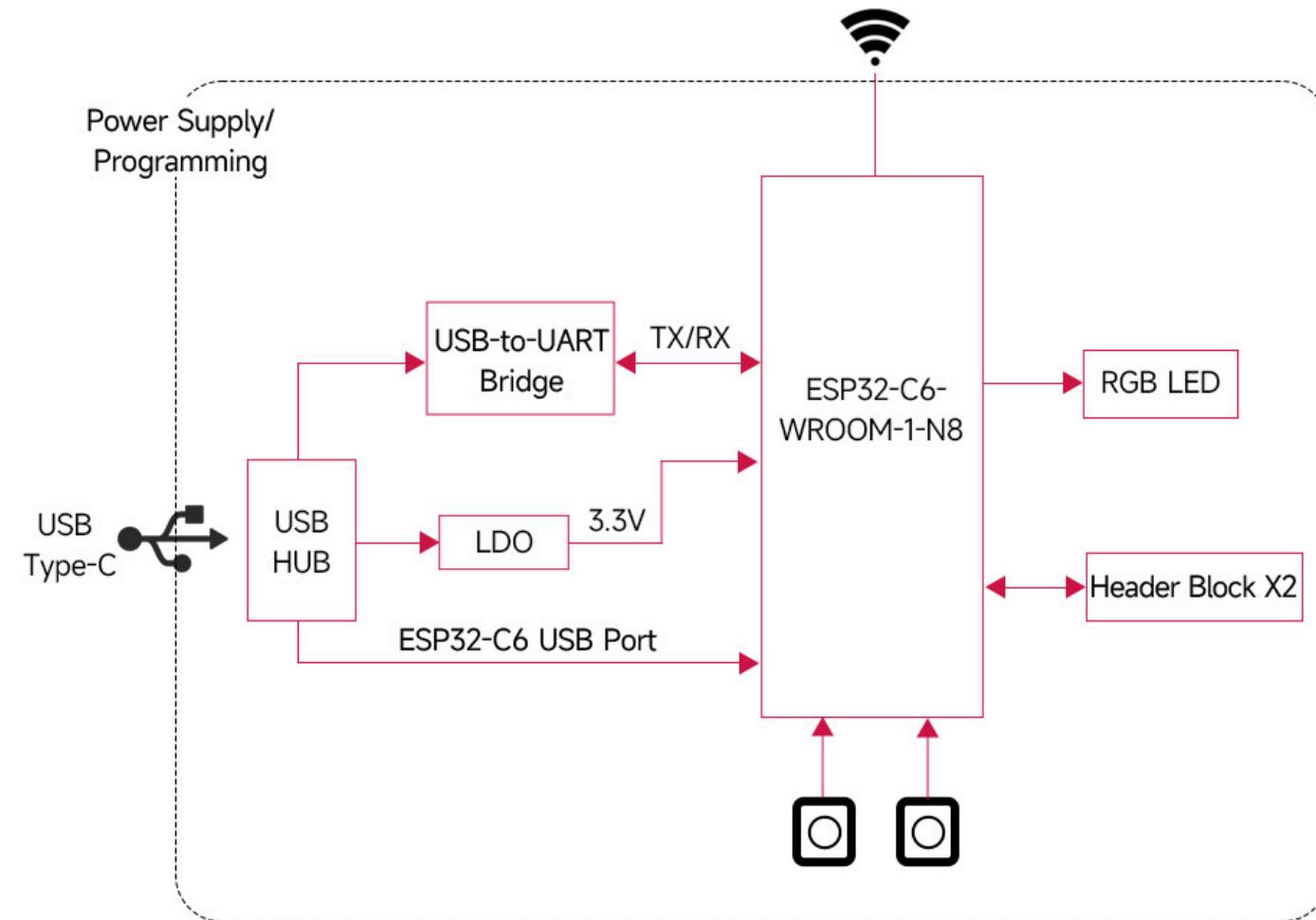


(<https://www.waveshare.com/esp32-c6-dev-kit-n8.htm?sku=25563>)

ESP32-C6
USB Type-C

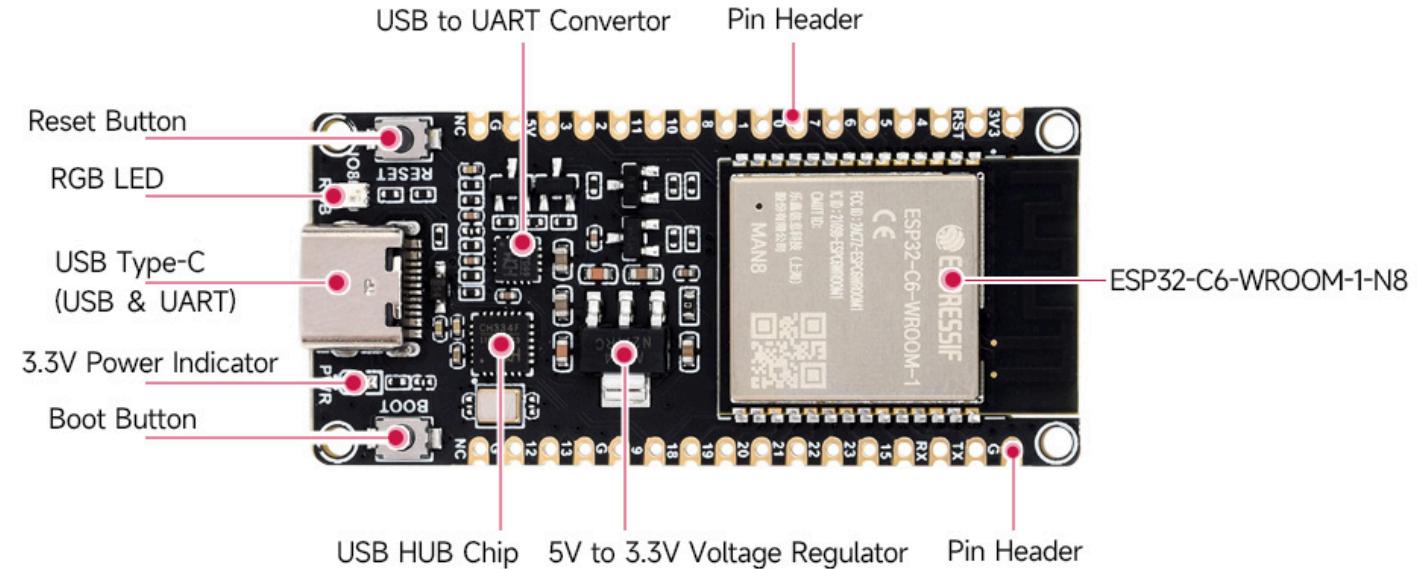
- Supports a variety of low-power operating states, adjustable balance among communication distance, data rate and power consumption, to meet the power requirements of various application scenarios.

Fuction Block Diagram



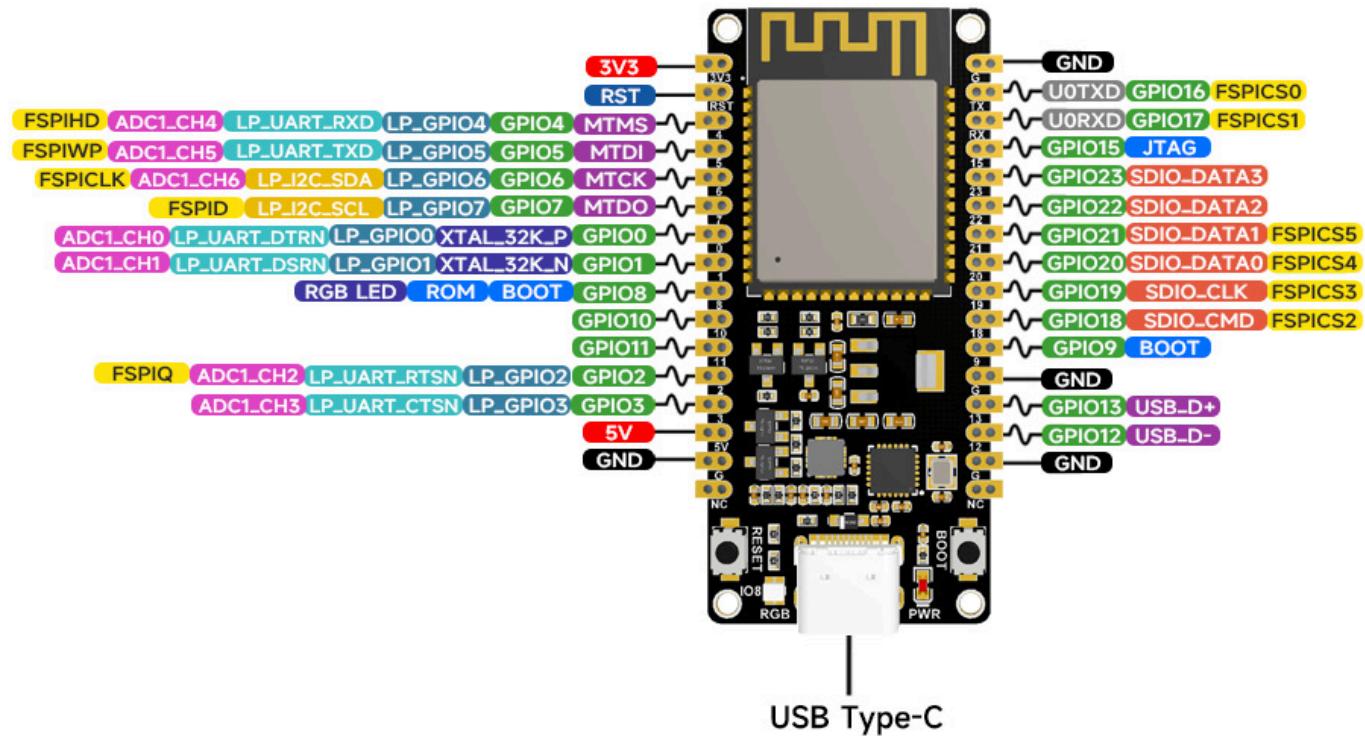
(/wiki/File:ESP32-C6-DEV-KIT-N803.jpg)

Onboard Resources



(/wiki/File:ESP32-C6-DEV-KIT-N804.jpg)

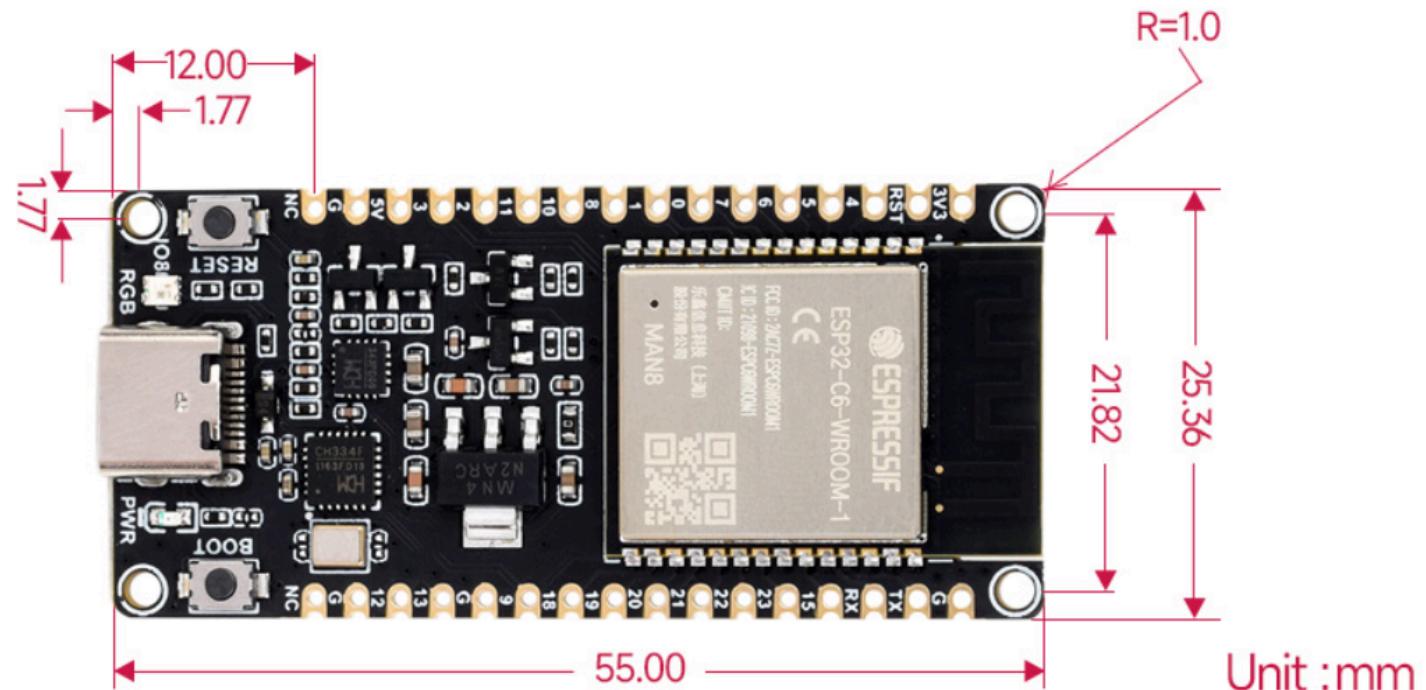
Pinout



PWM Capable Pin	FSPI Fast SPI Functions
GPIOX GPIO Input and Output	LP_UART Low-Power UART Functions
LP_I2C Low-Power I2C Functions	SDIO SDIO Functions
OTHER Other Related Functions	STRAP Strapping Pin Functions
JTAG/USB JTAG for Debugging and USB	ADCX_CH Analog-to-Digital Converter
SERIAL Serial for Debug/Programming	LP_GPIOX Low-Power GPIO Functions
PWR Power Rails (3V3 and 5V)	GND Ground Plane

(/wiki/File:ESP32-C6-DEV-KIT-N805.jpg)

Dimensions



(/wiki/File:ESP32-C6-DEV-KIT-N806.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

↓ .deb
Debian, Ubuntu

↓ .rpm
Red Hat, Fedora, SUSE

↓ Mac

macOS 10.11+

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

CLI x64 x86 Arm64

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

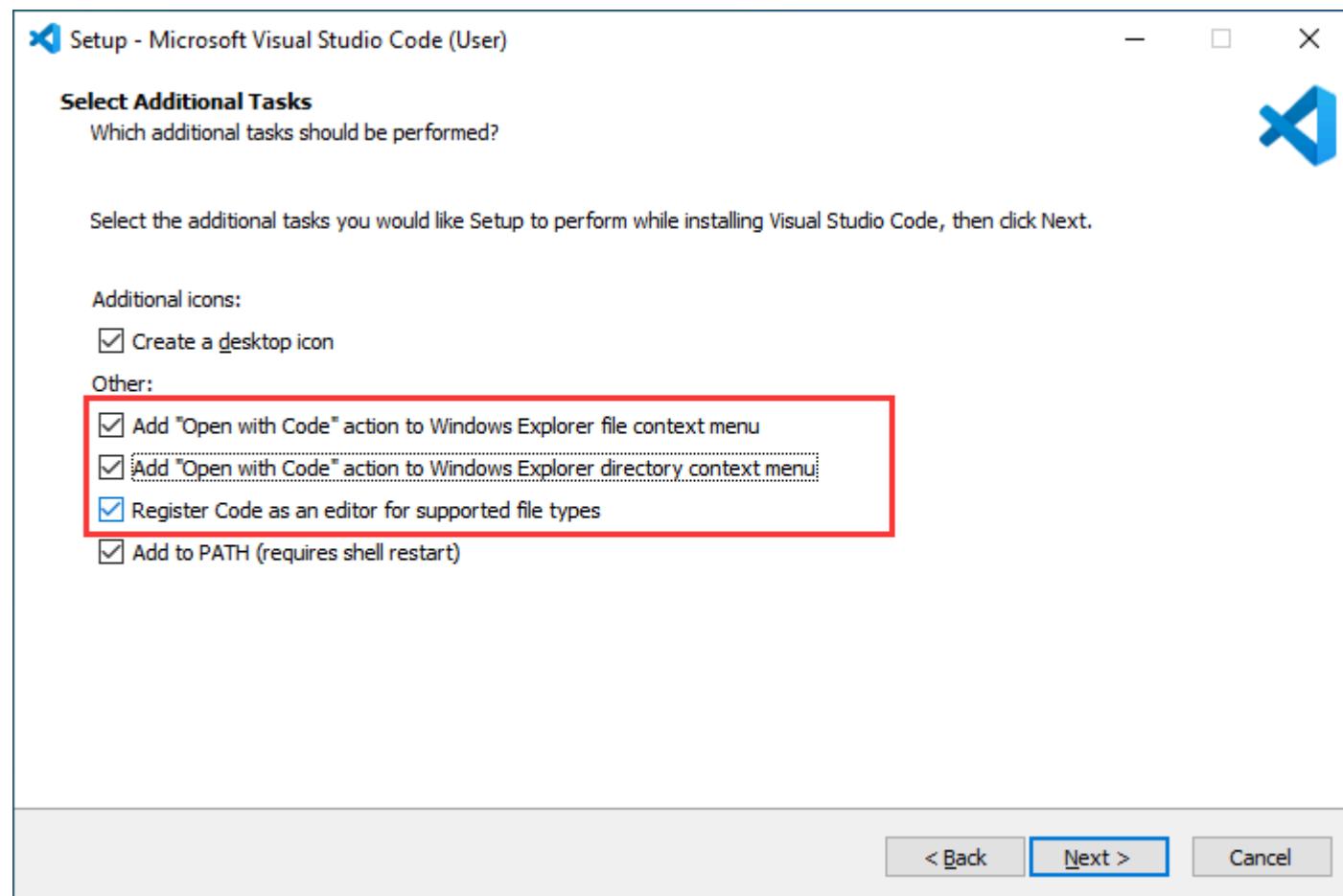
CLI x64 Arm32 Arm64

.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)

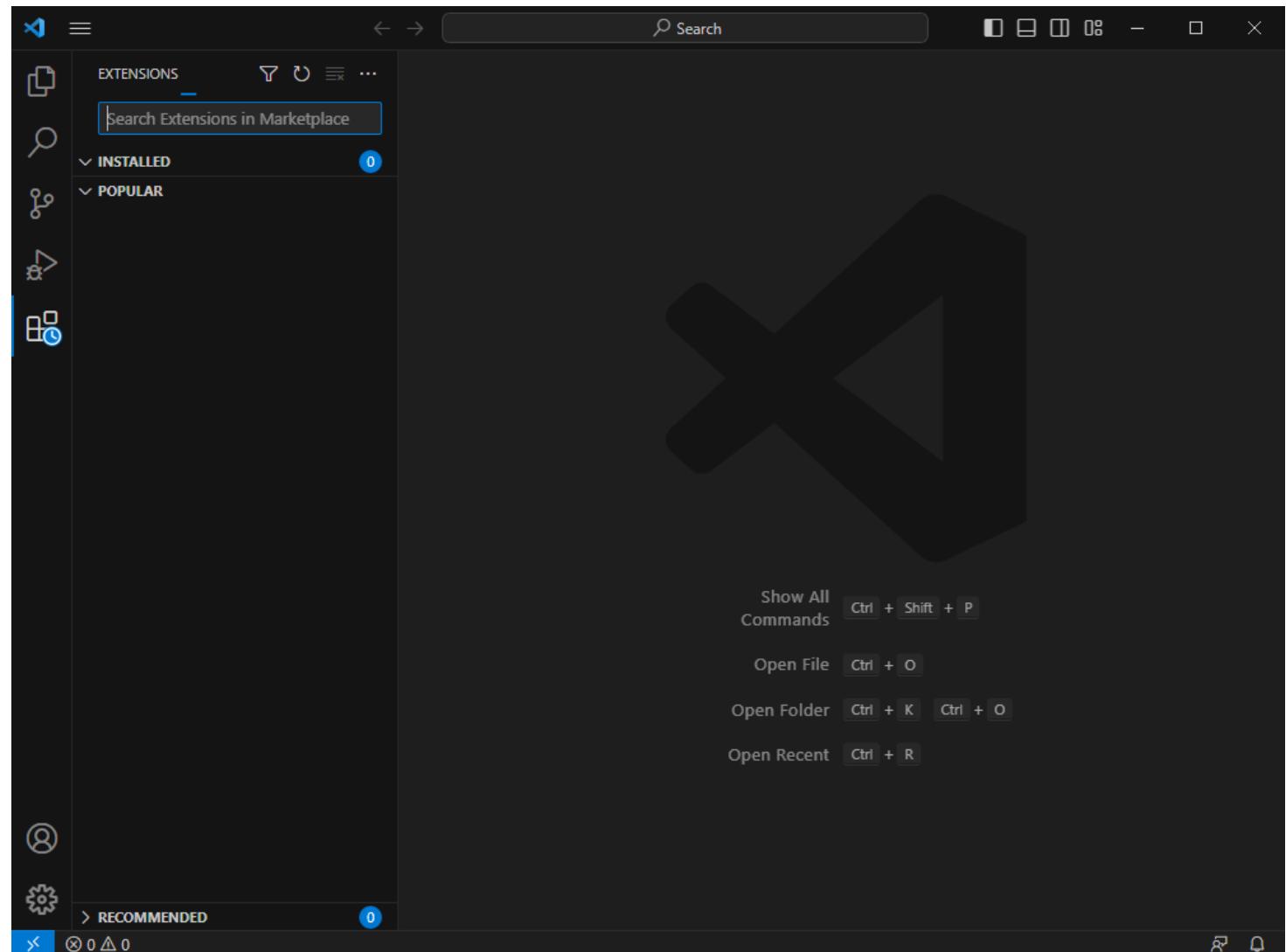
- After running the installation package, the rest can be installed by default, but here for a better experience, it is recommended to check the box here in the 1, 2 and 3 items.
 - After enabling the 1st and 2nd items, you can directly open the VScode by right-clicking the file or the directory to improve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



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

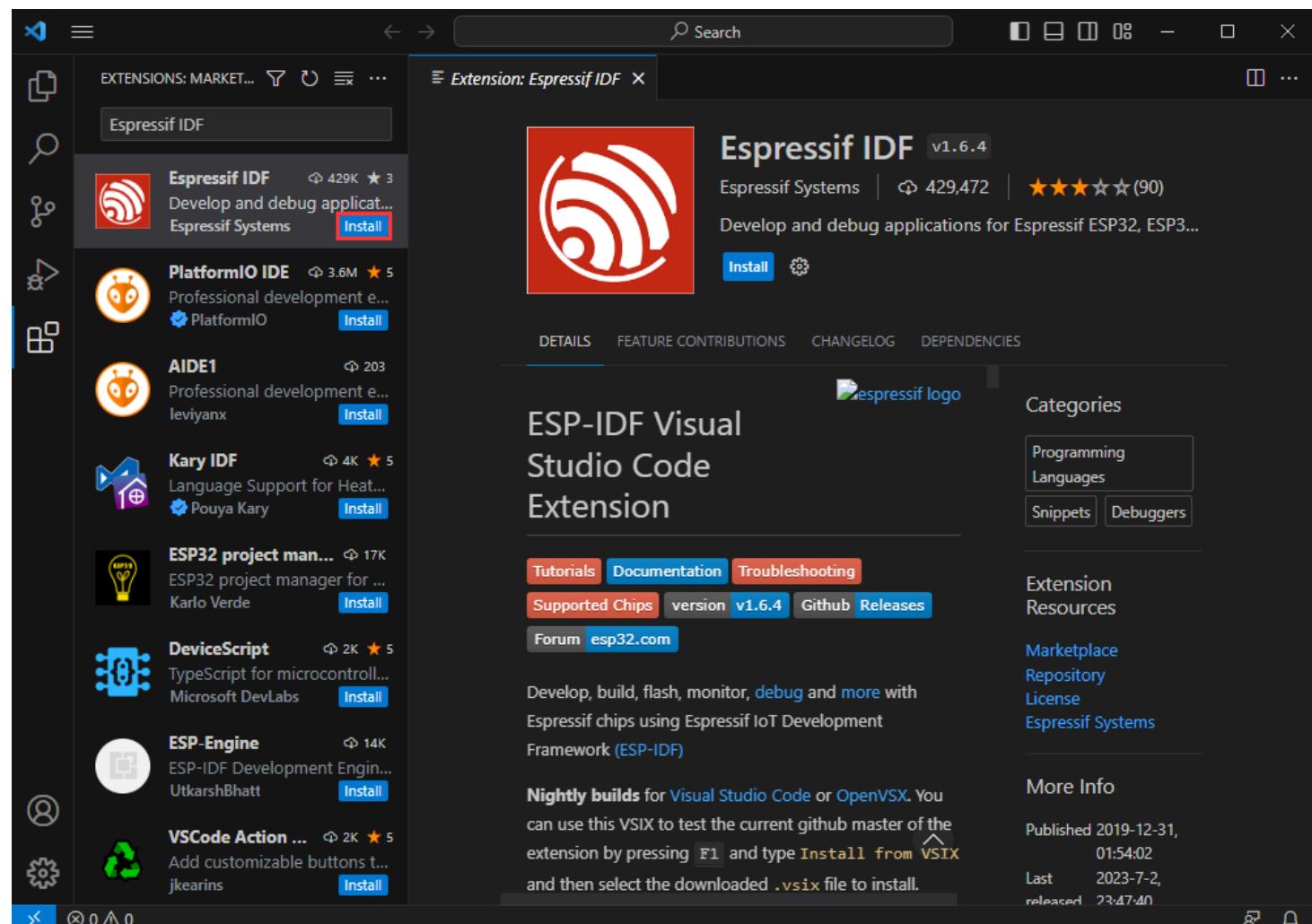
Install Espressif IDF Plug-in

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

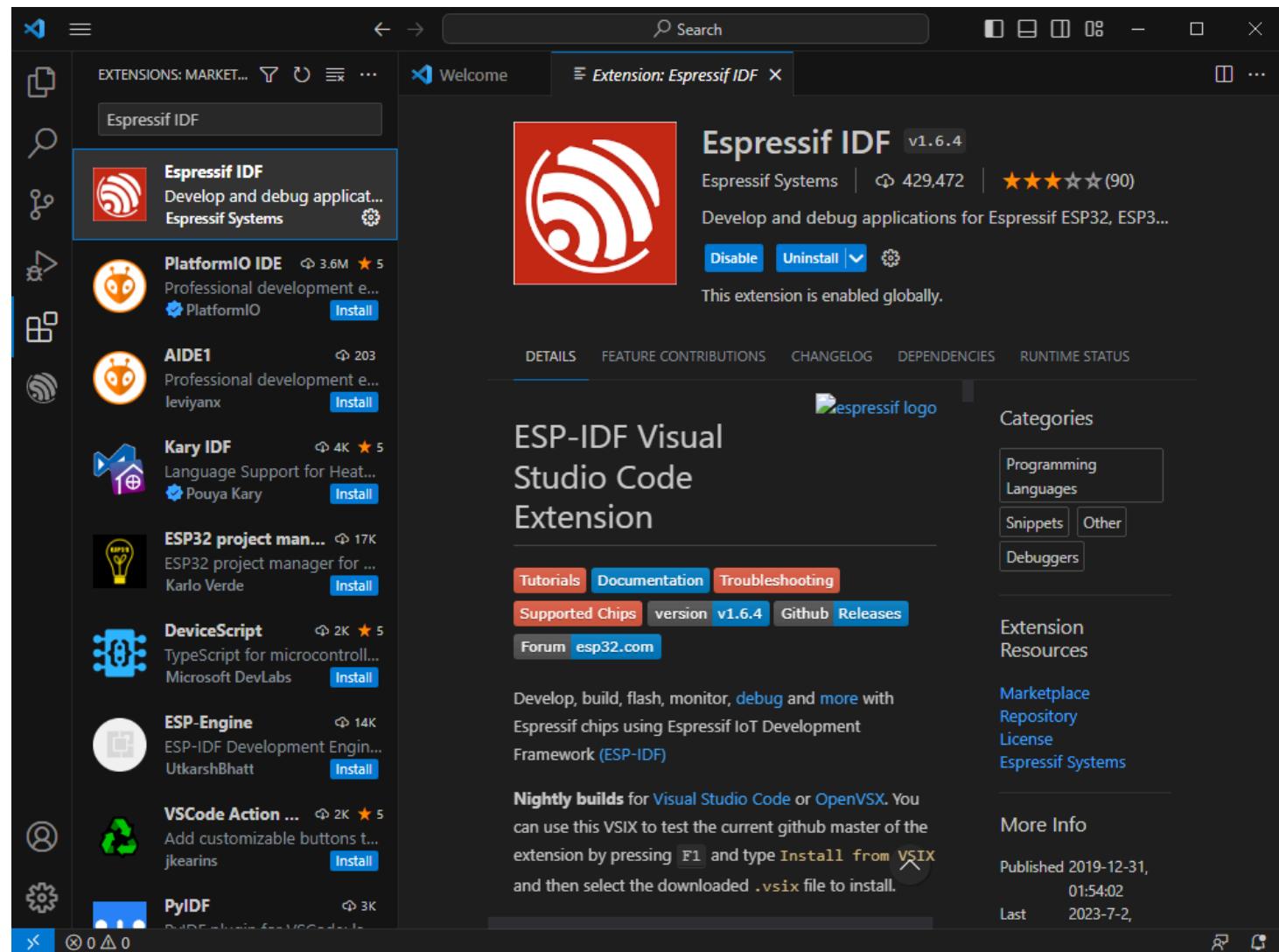


(/wiki/File:ESP32-C6-DEV-KIT-N8-03.png)

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



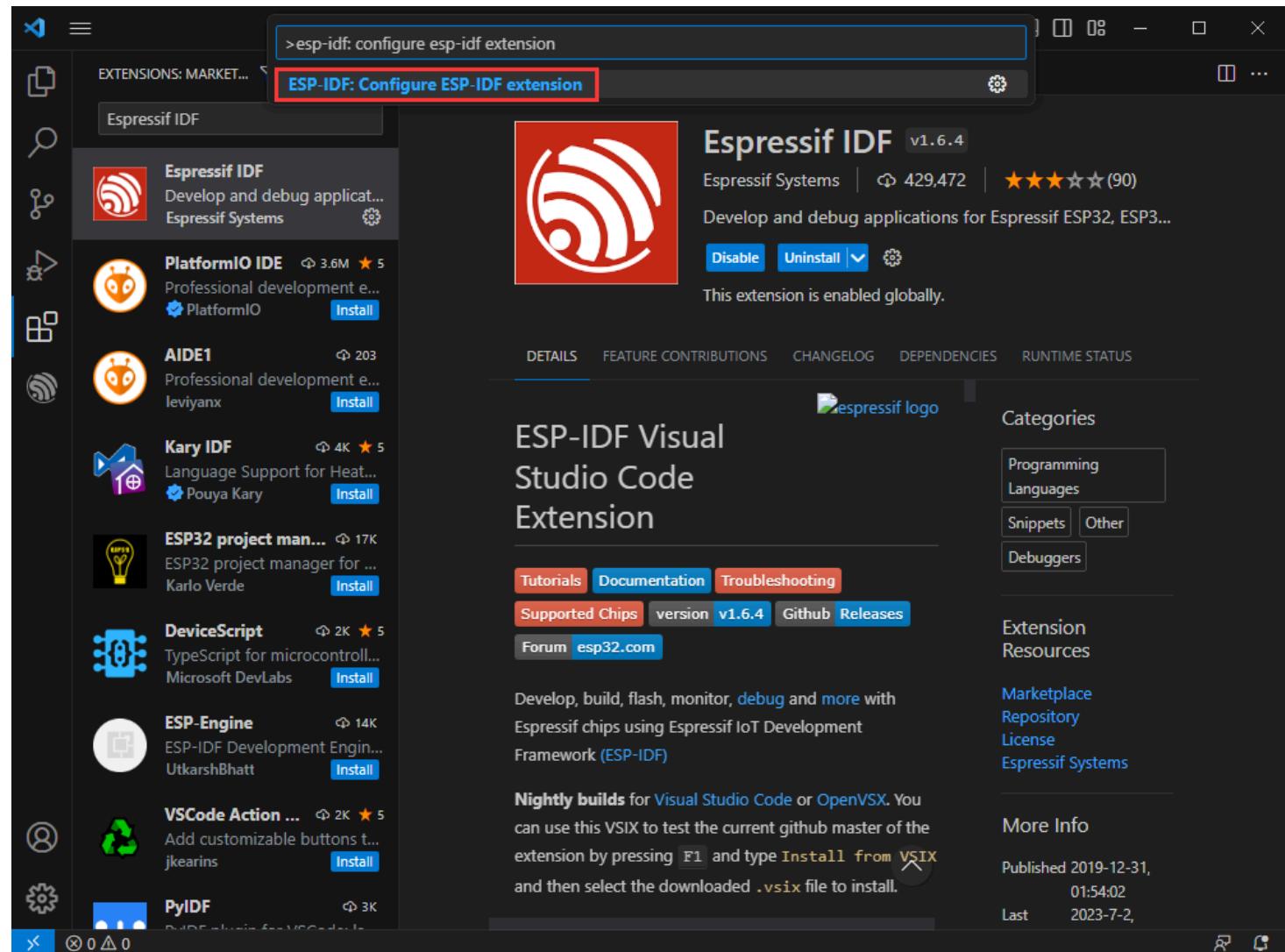
(/wiki/File:ESP32-C6-DEV-KIT-N8-04.png)



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

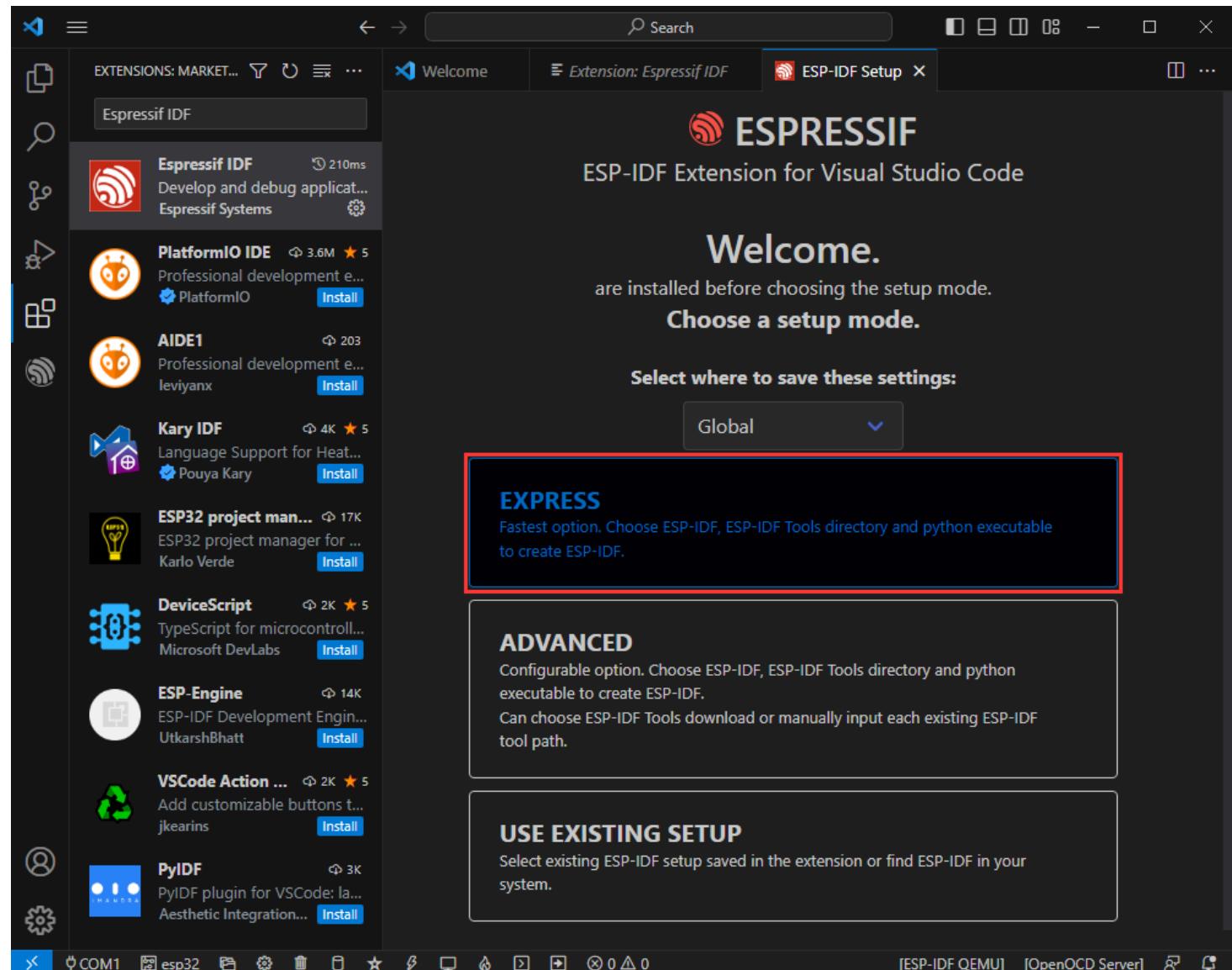
- Press **F1** to input:

esp-idf: configure esp-idf extension



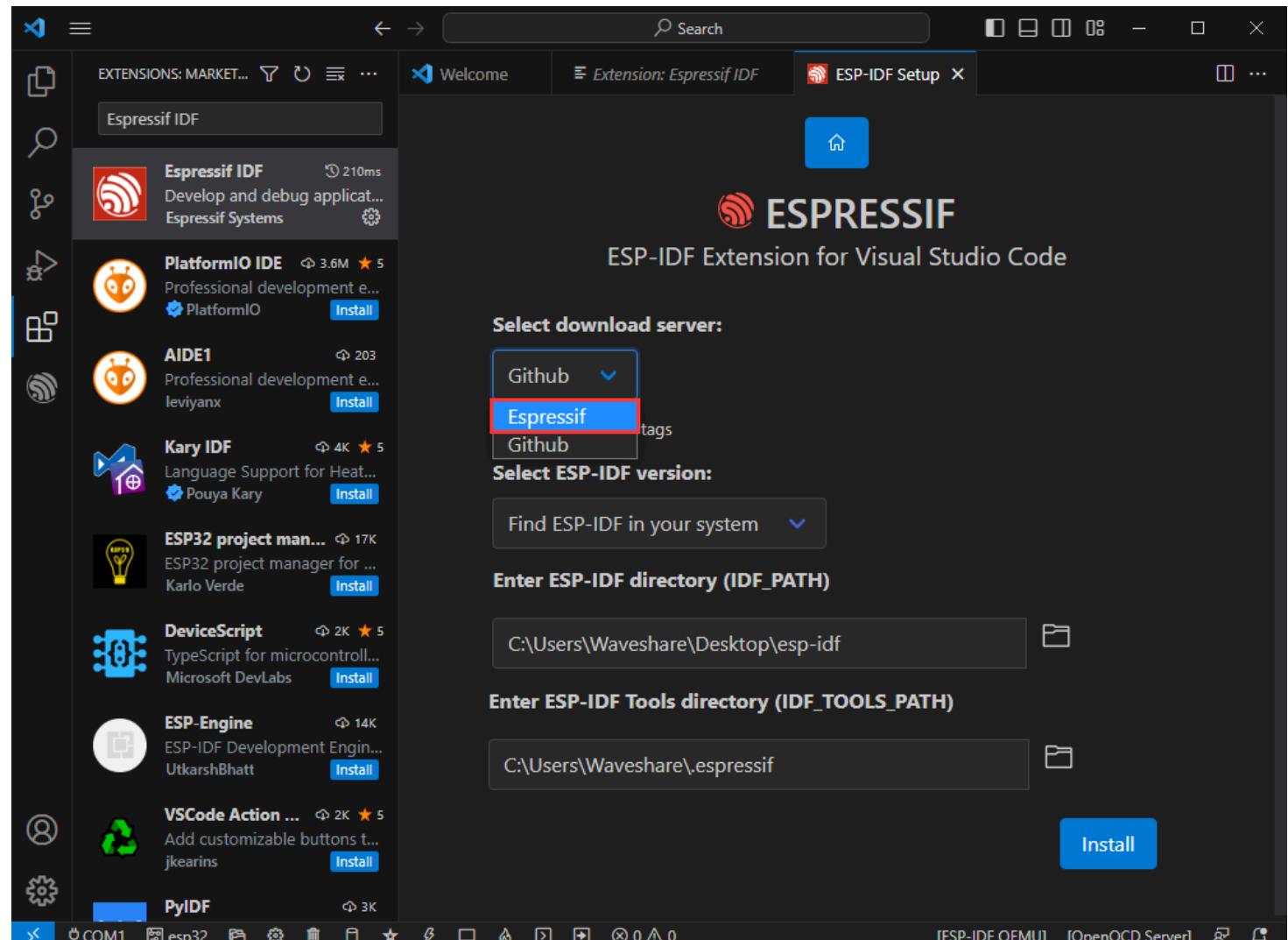
(/wiki/File:ESP32-C6-DEV-KIT-N8-06.png)

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



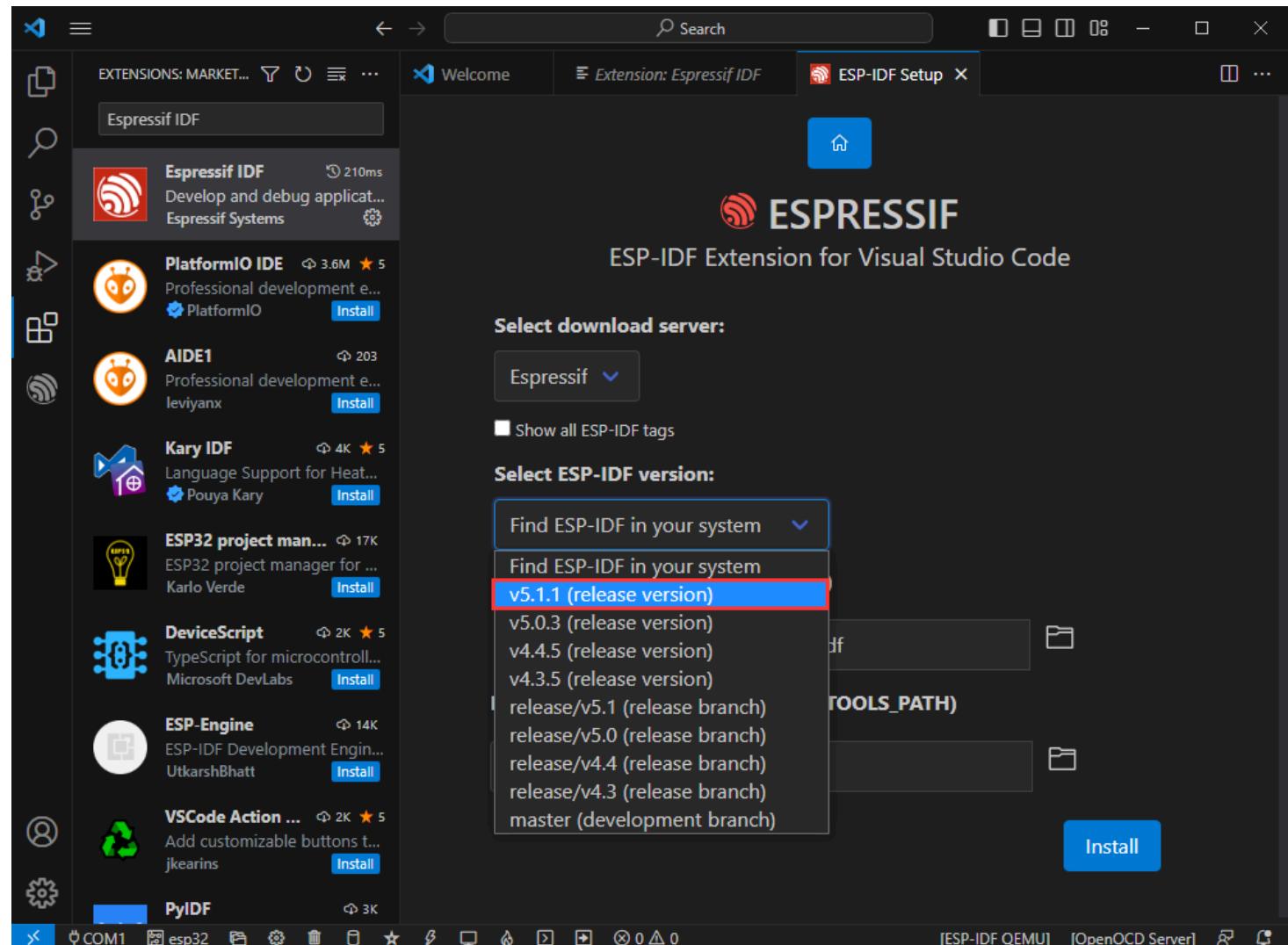
(/wiki/File:ESP32-C6-DEV-KIT-N8-07.png)

- Select download sever.



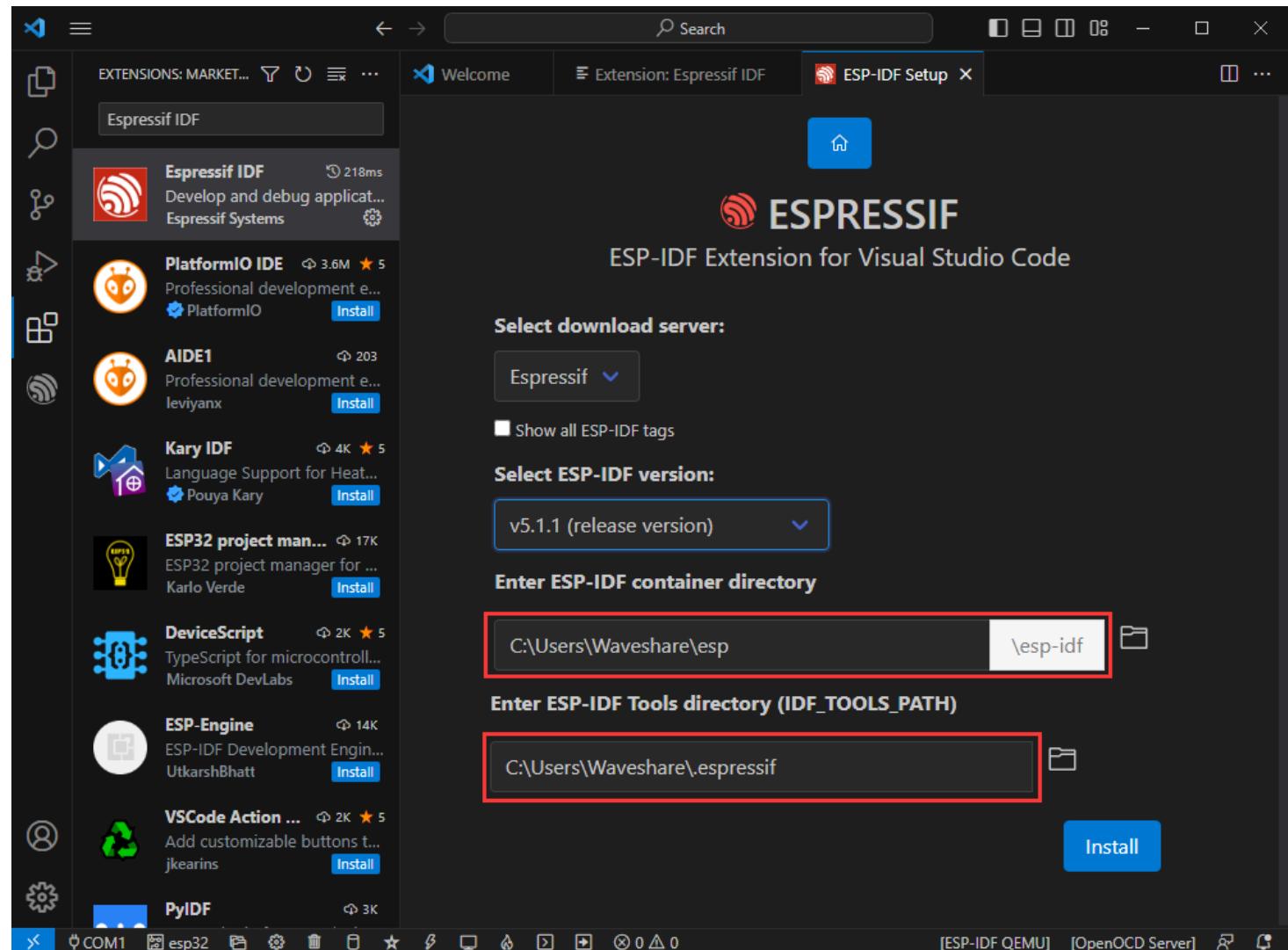
(/wiki/File:ESP32-C6-DEV-KIT-N8-08.png)

- 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).



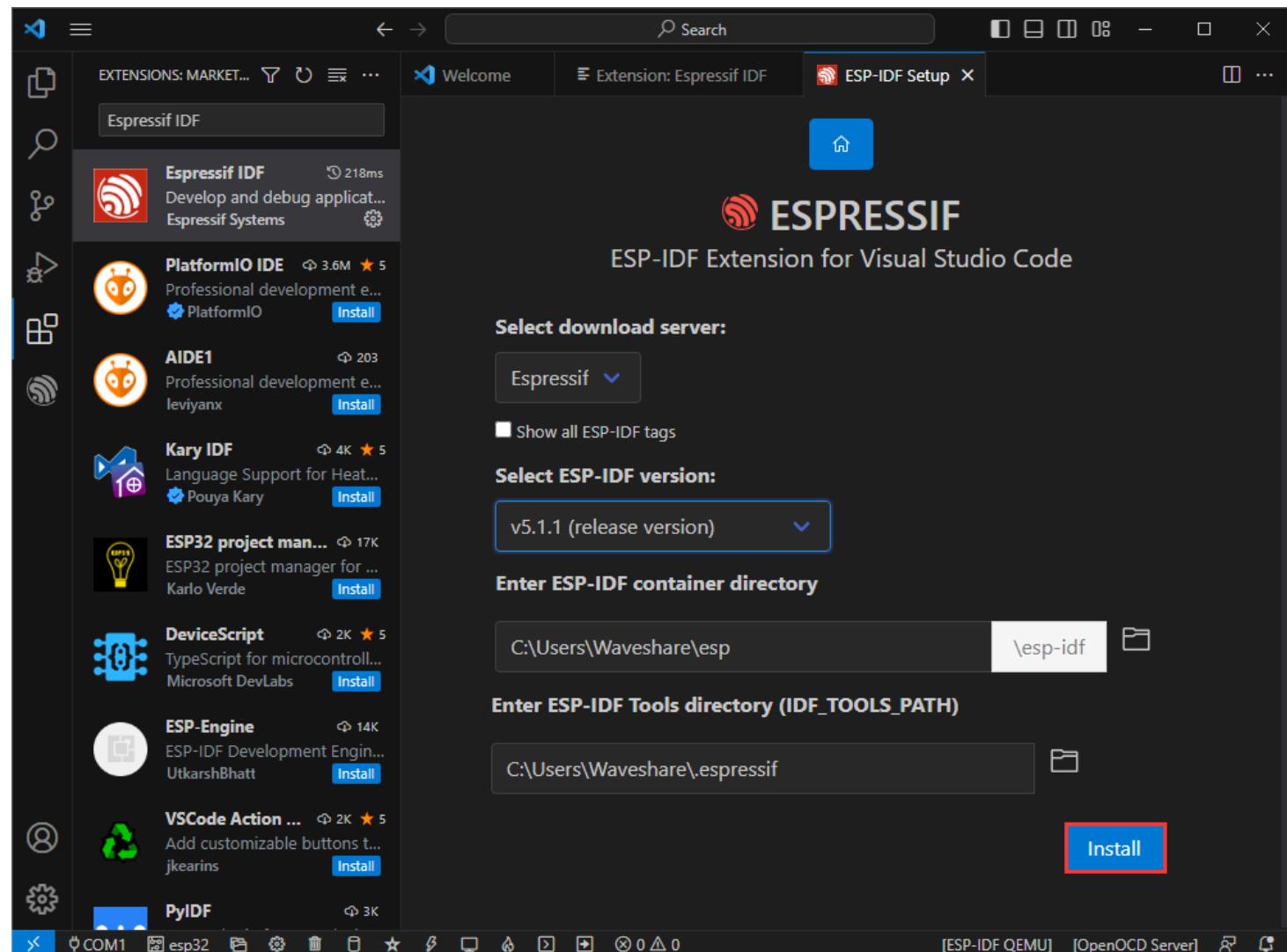
(/wiki/File:ESP32-C6-DEV-KIT-N8-09.png)

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



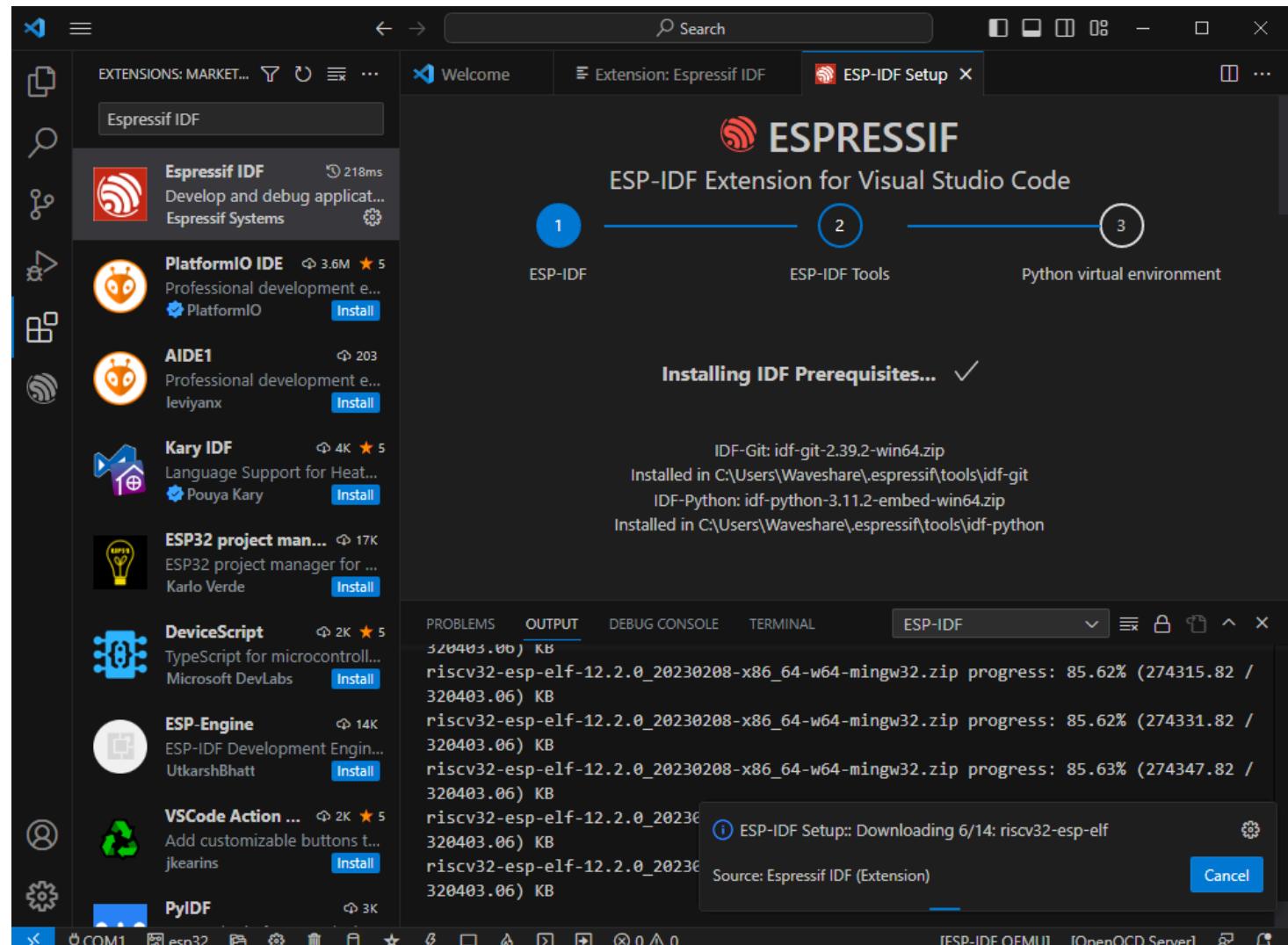
(/wiki/File:ESP32-C6-DEV-KIT-N8-10.png)

- **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:



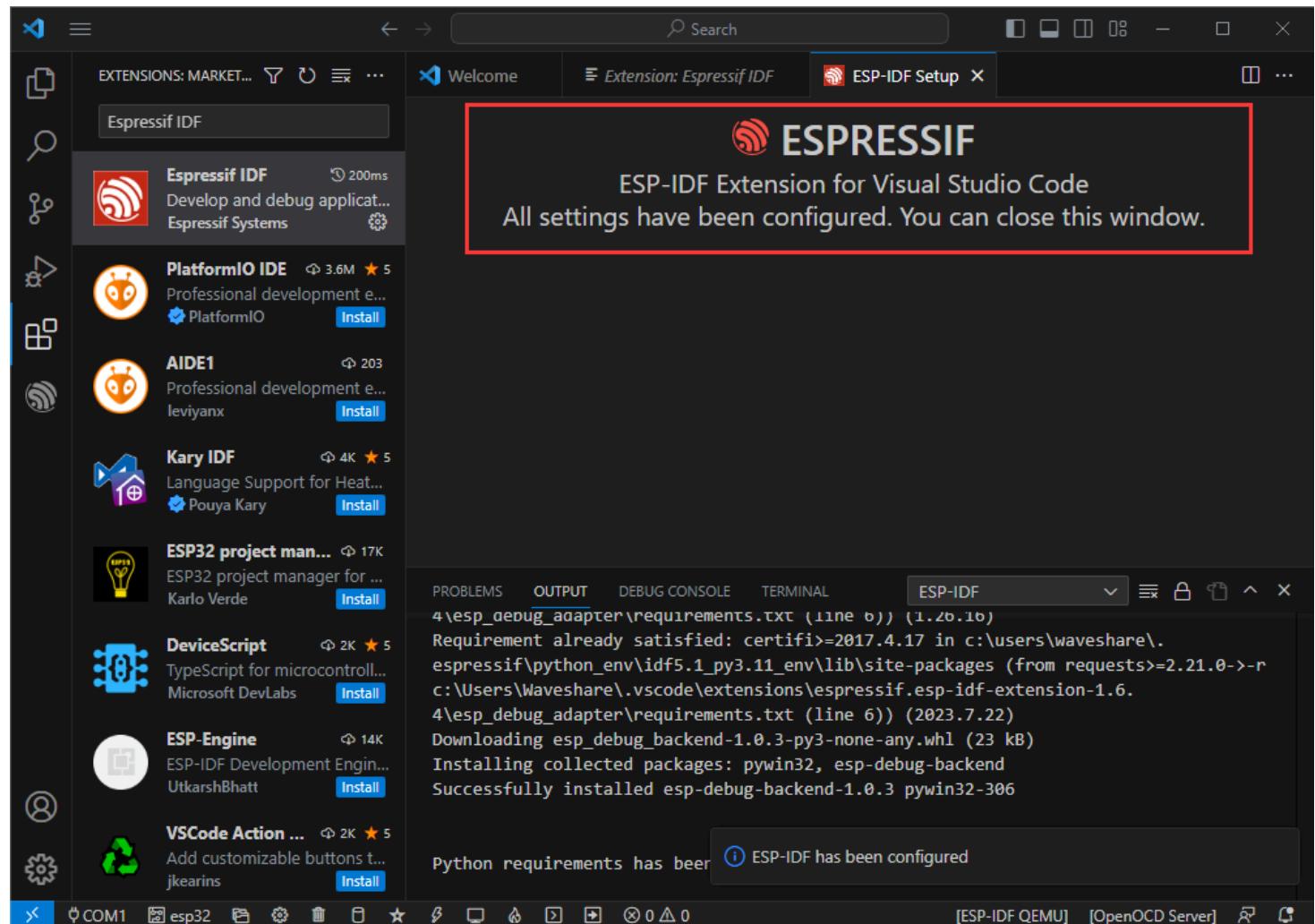
(/wiki/File:ESP32-C6-DEV-KIT-N8-19.png)

- 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.

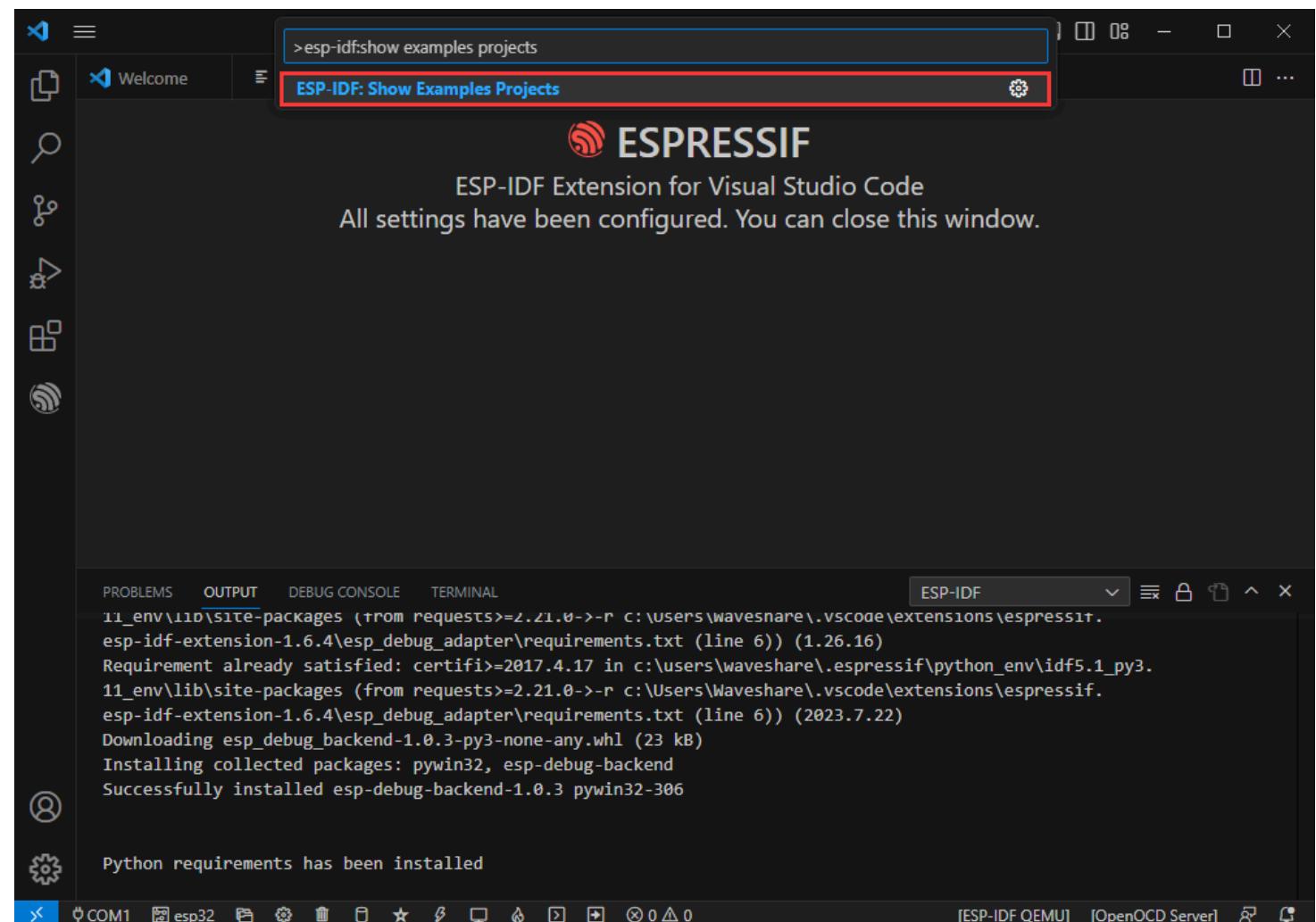


Official Demo Usage GUIDE

Create Demo (Demo Example)

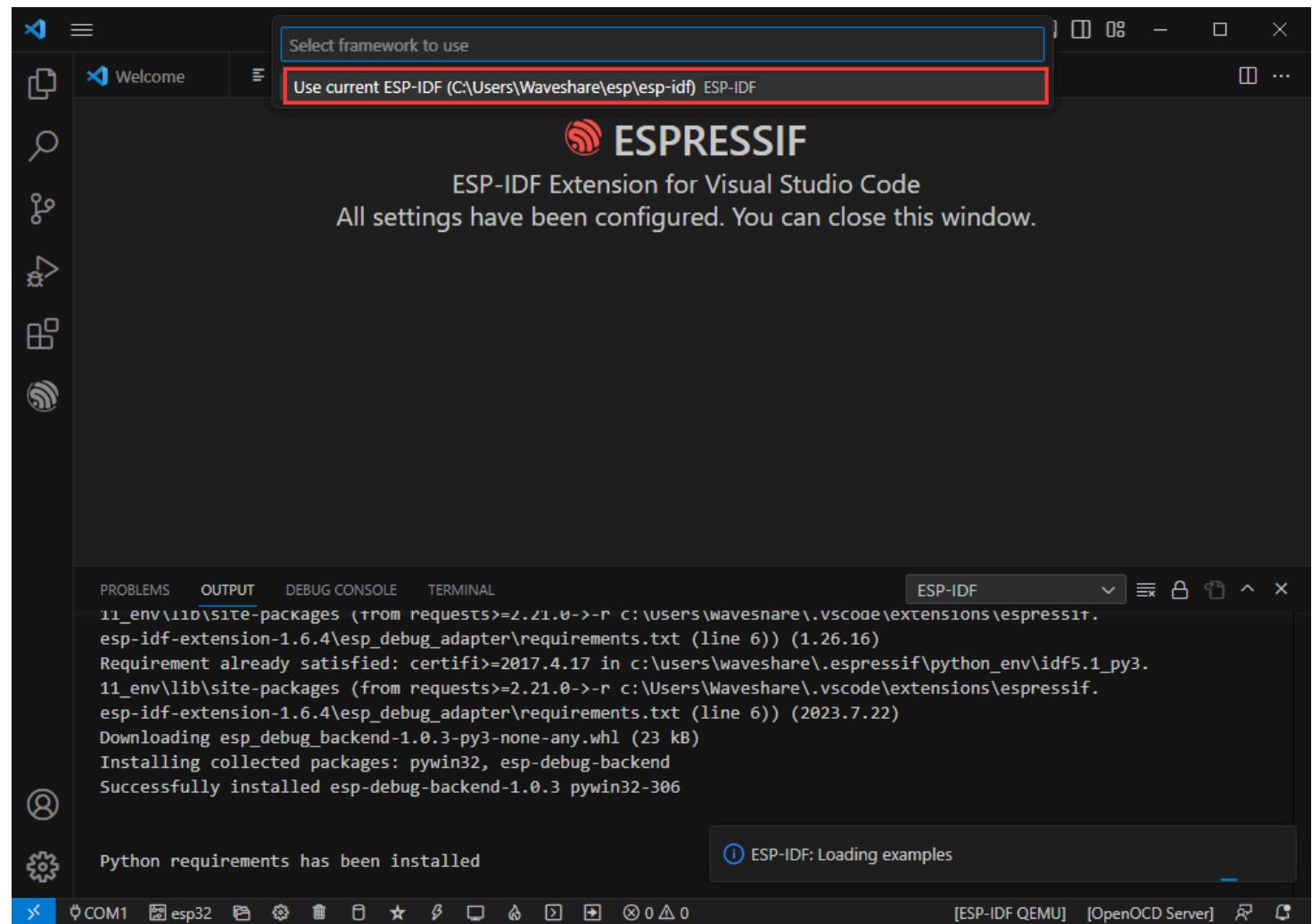
- Press **F1** to enter:

esp-idf:show examples projects



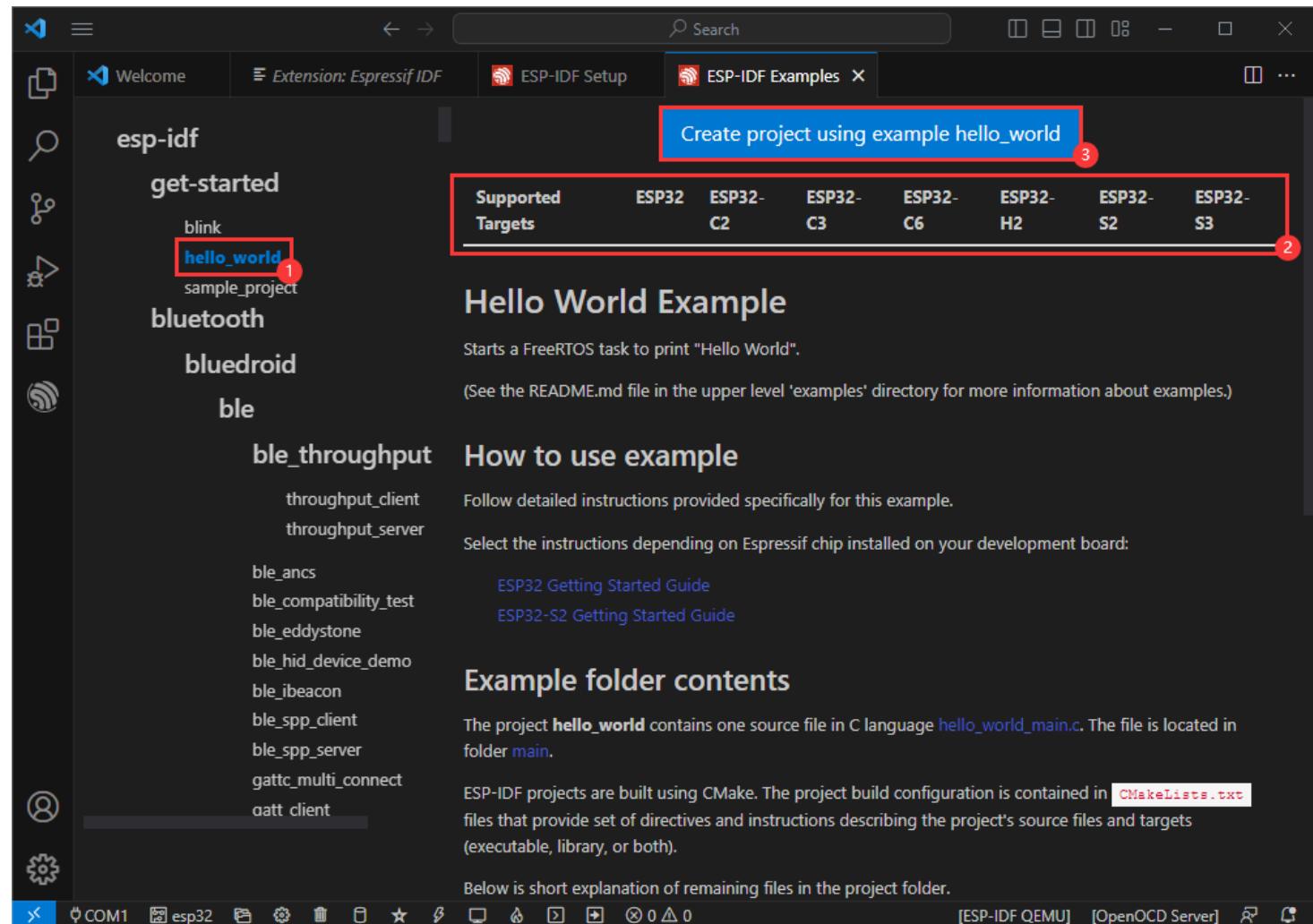
(/wiki/File:ESP32-C6-DEV-KIT-N8-13.png)

- Select the corresponding IDF version:



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

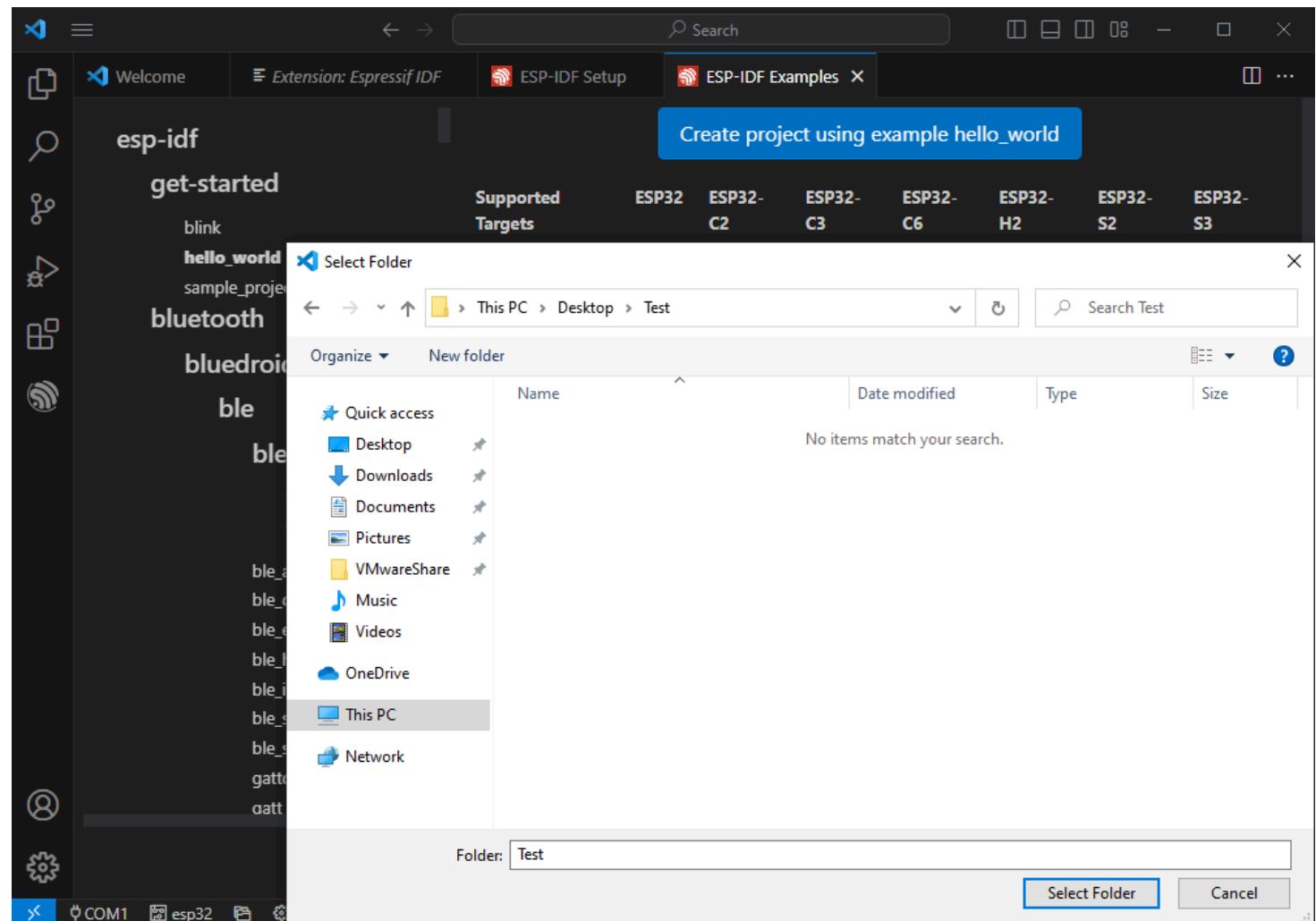
- Take the Hello World demo as an example:
 - ①Select the corresponding demo.
 - ②Its readme will state what chip the demo applies to (how the demo is used and the file structure are described below, omitted here).
 - ③Click to create the demo.



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

Select the path to place the demo, and the path for placing demos should not contain a folder

with the same name as the demo.

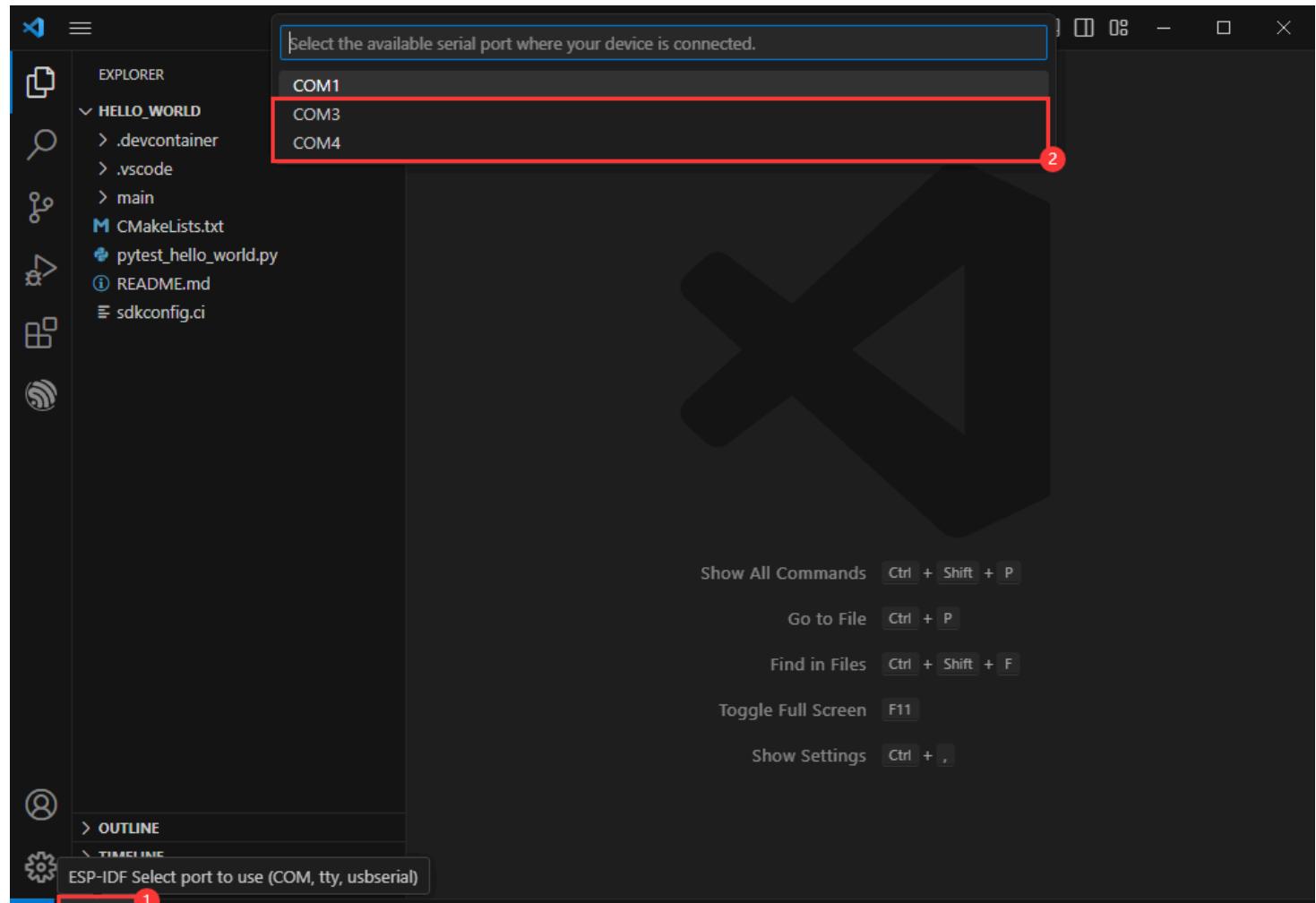


(/wiki/File:ESP32-C6-DEV-KIT-N8-16.png)

Modify COM Port

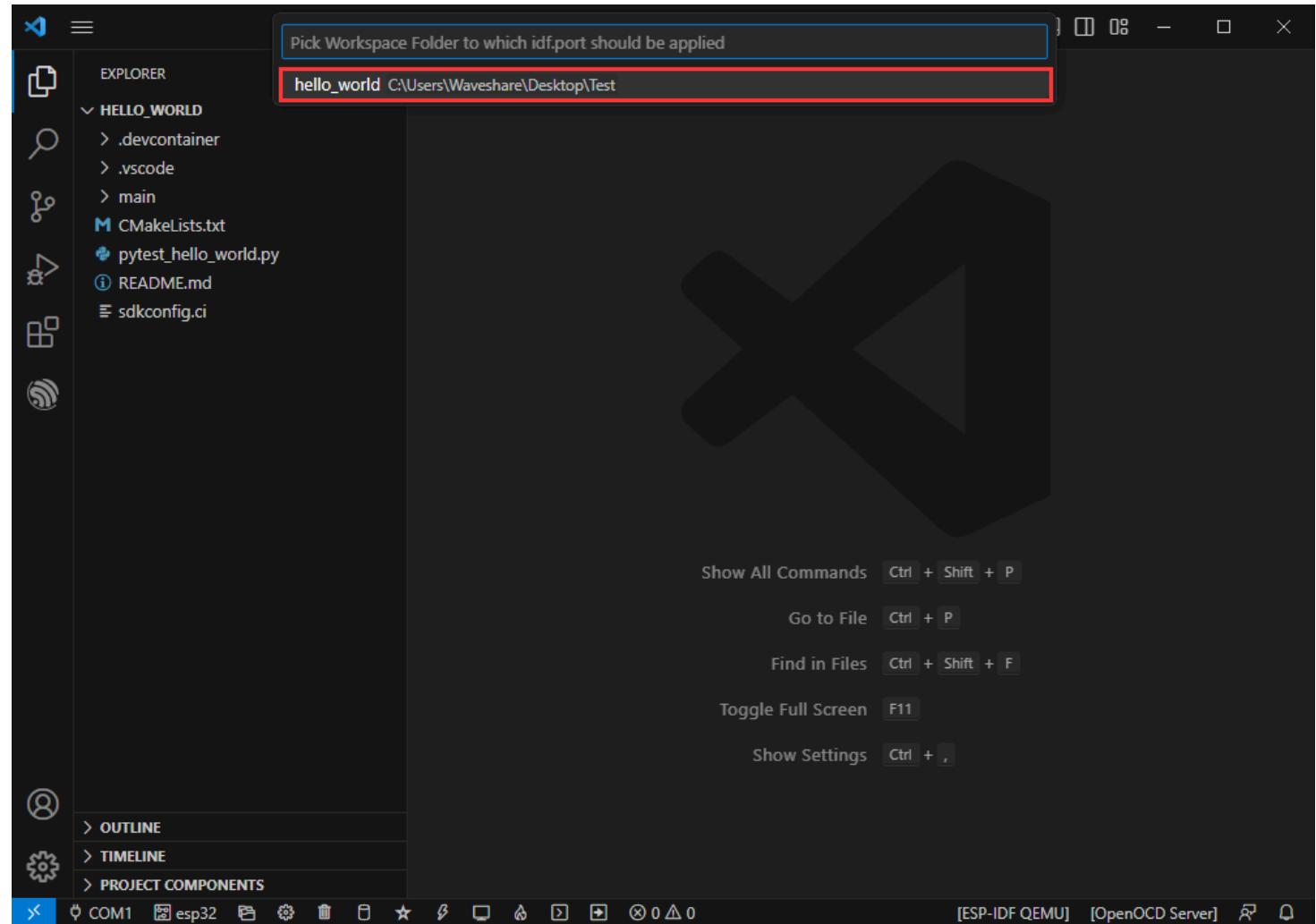
- The corresponding COM ports are shown here, click to modify them.
- Please select the COM ports according to your device. **It is recommended to use the COM port corresponding to the USB connector. (You can view it from the device manager.)**

- 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:

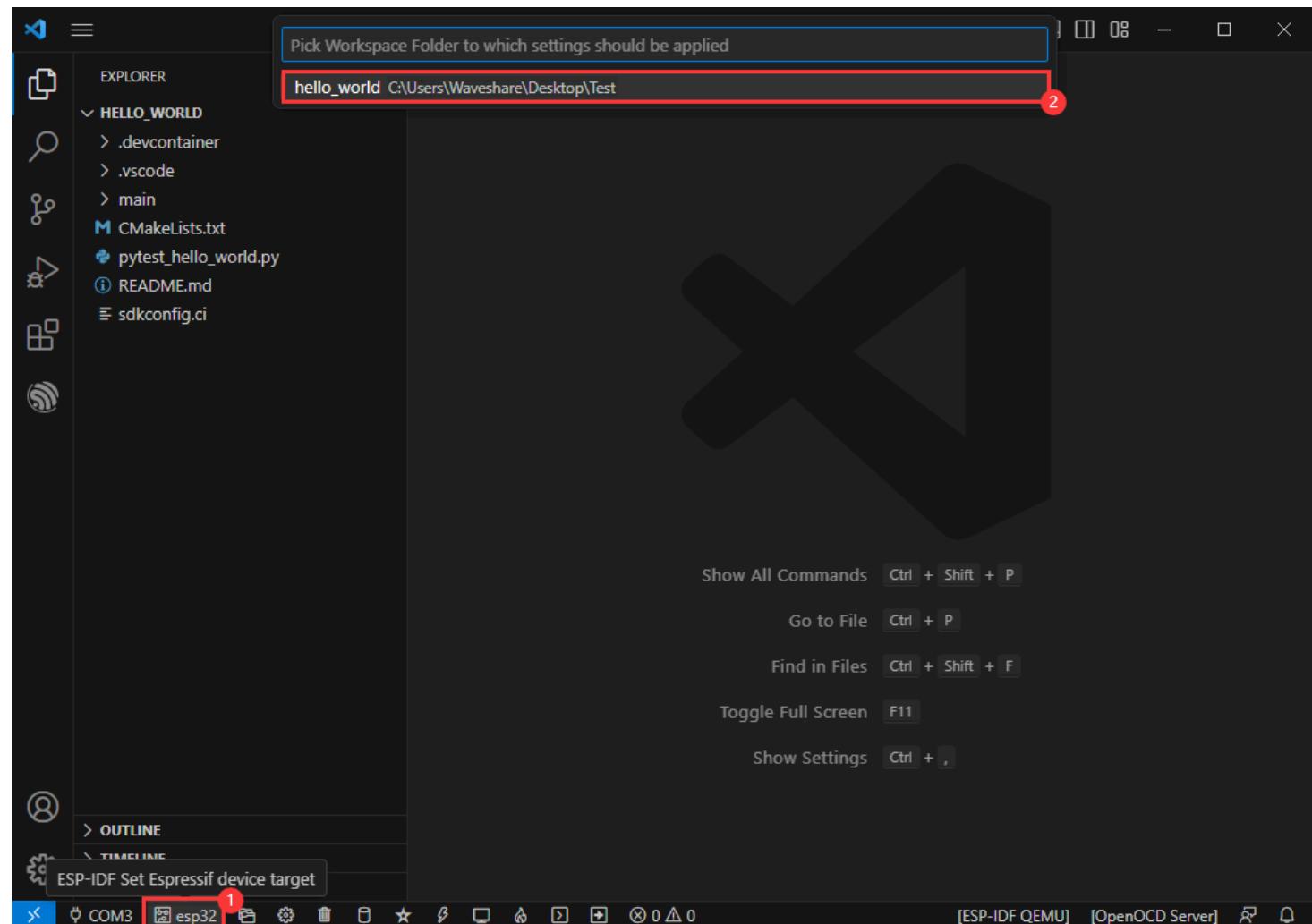


(/wiki/File:ESP32-C6-DEV-KIT-N8-18.png)

- Then we finish the modification of the COM ports.

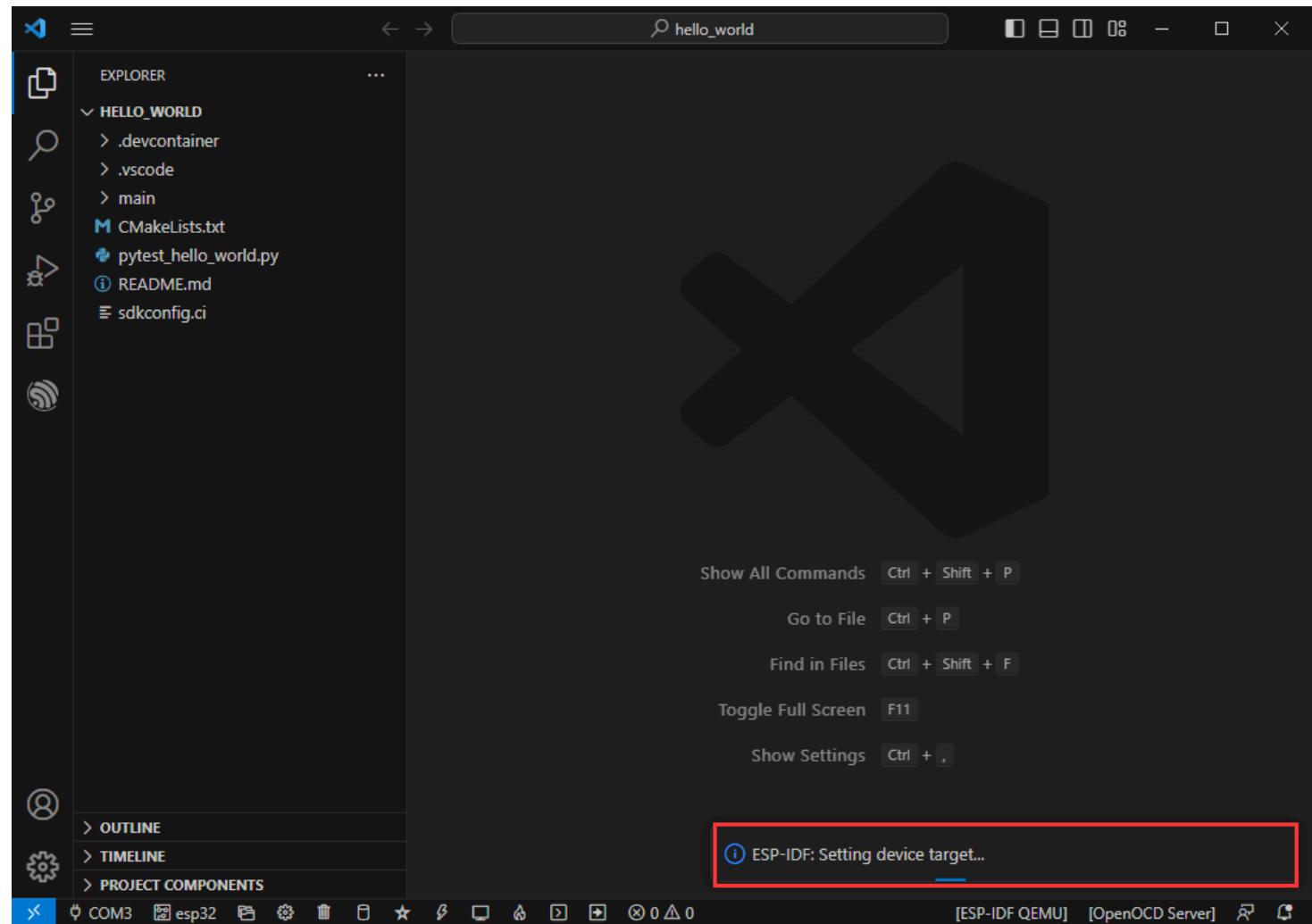
Modify the Driver Object

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



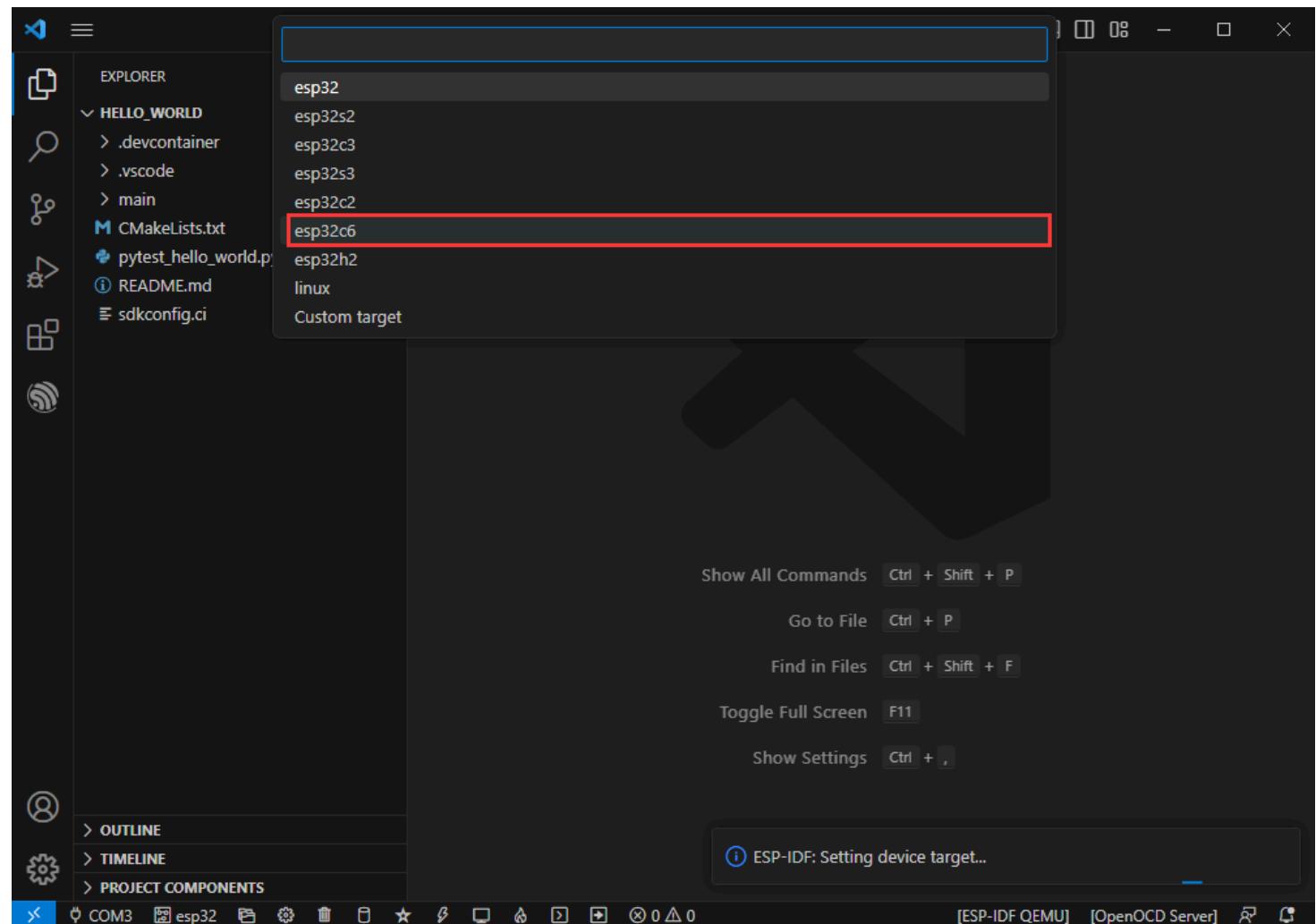
(/wiki/File:ESP32-C6-DEV-KIT-N8-20.png)

- Wait for a minute after clicking.



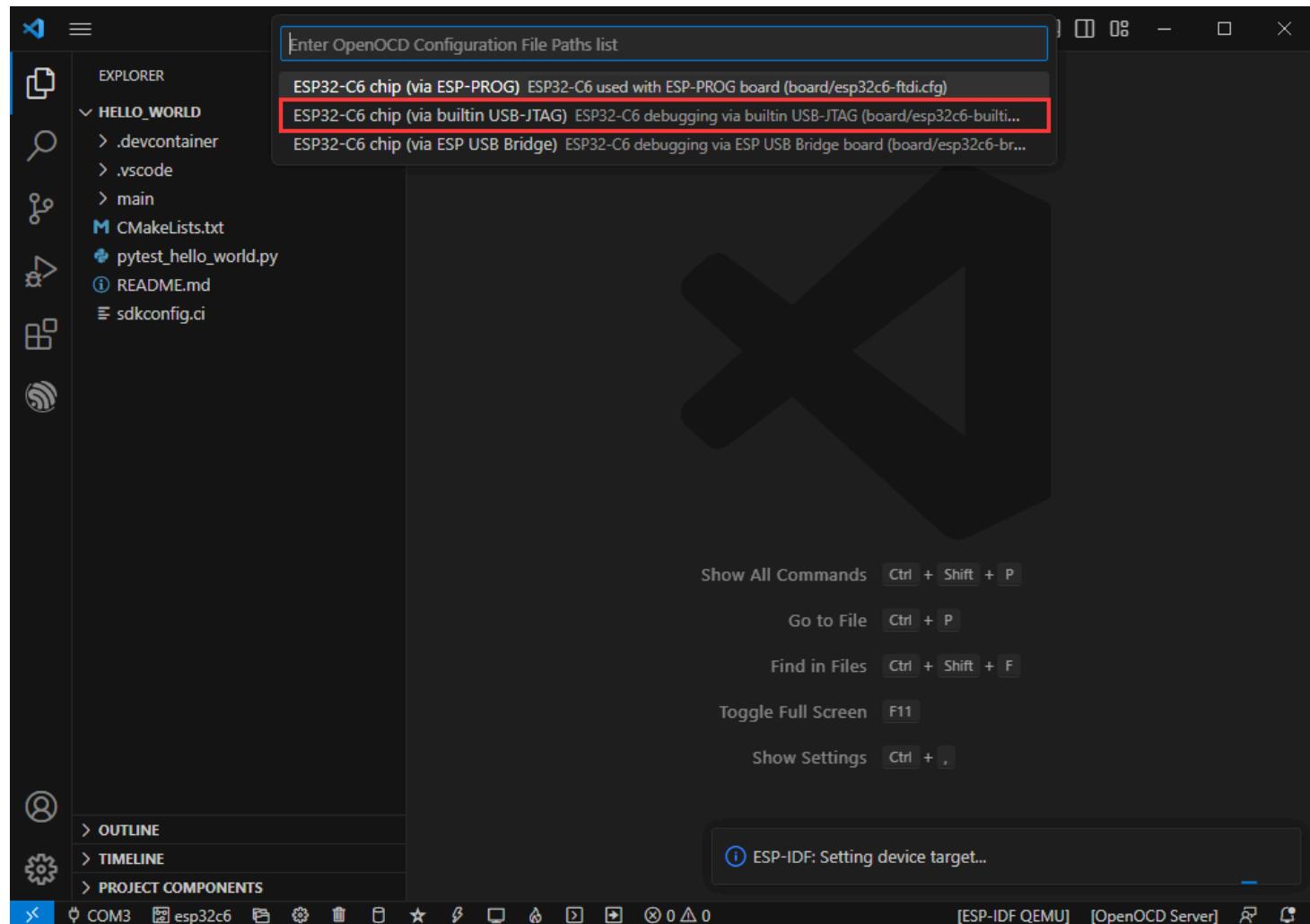
(/wiki/File:ESP32-C6-DEV-KIT-N8-21.png)

- 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.

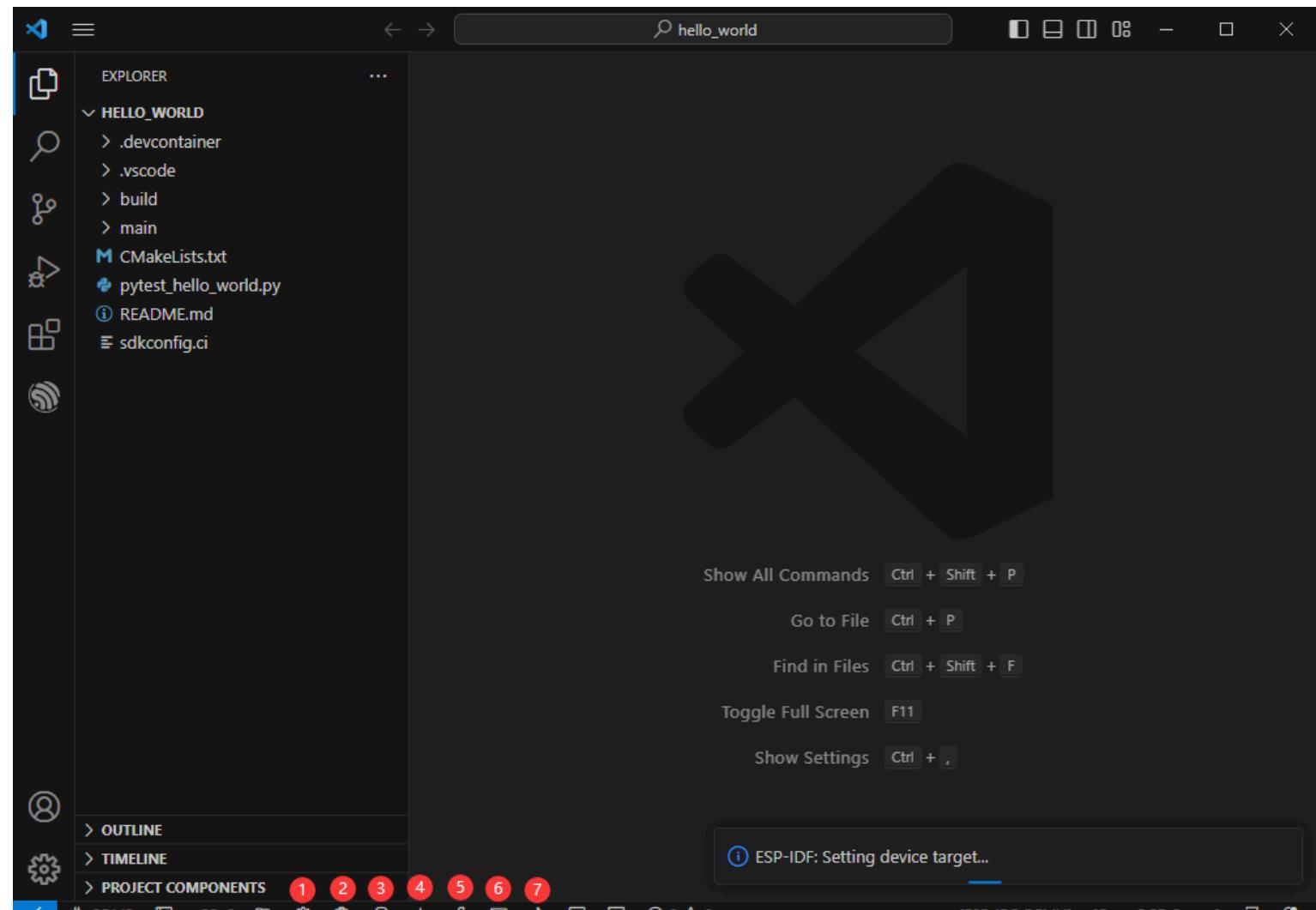


(/wiki/File:ESP32-C6-DEV-KIT-N8-23.png)

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, flash 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.
- Editor View:** 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"

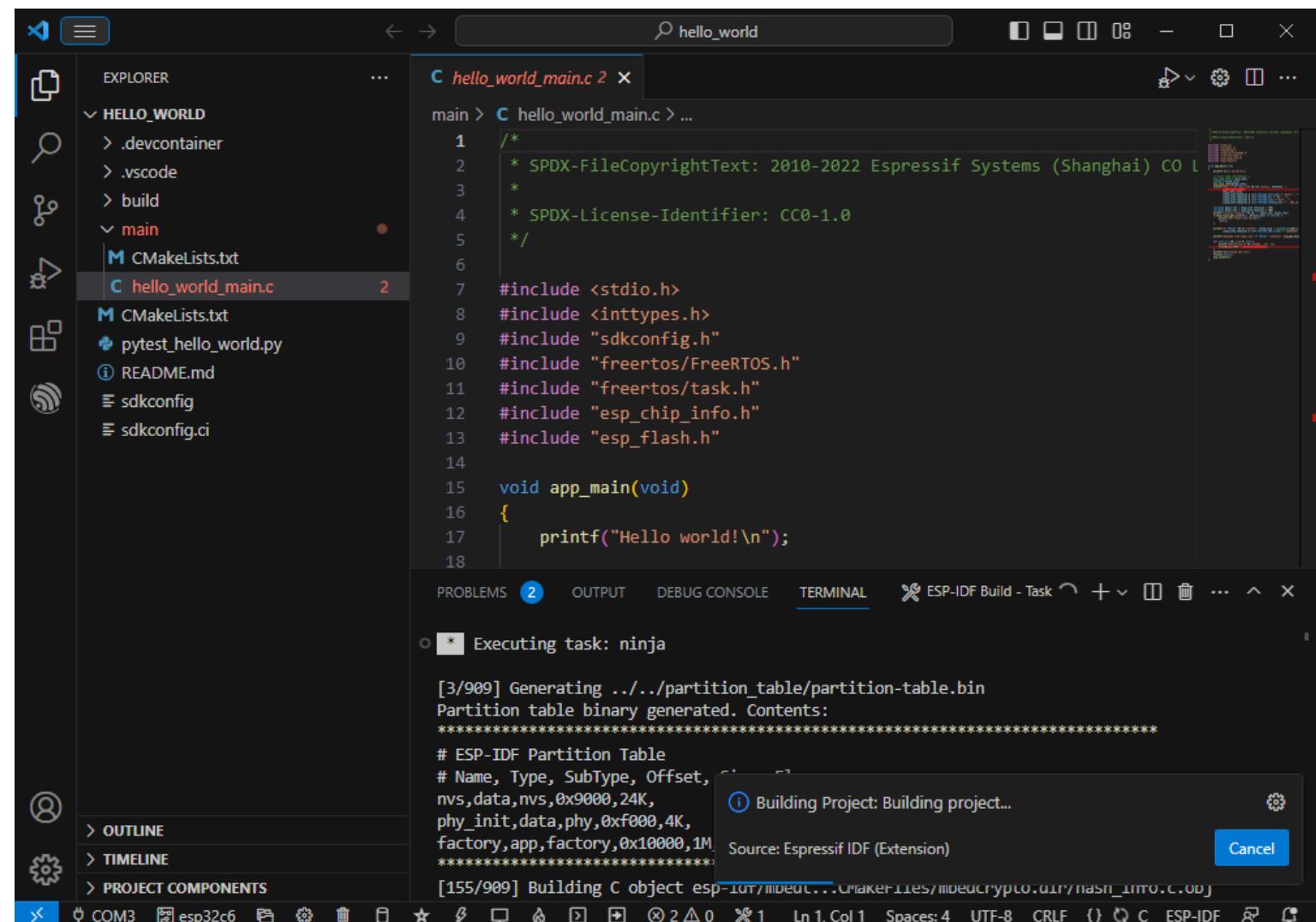
void app_main(void)
{
    printf("Hello world!\n");

    /* 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;
```

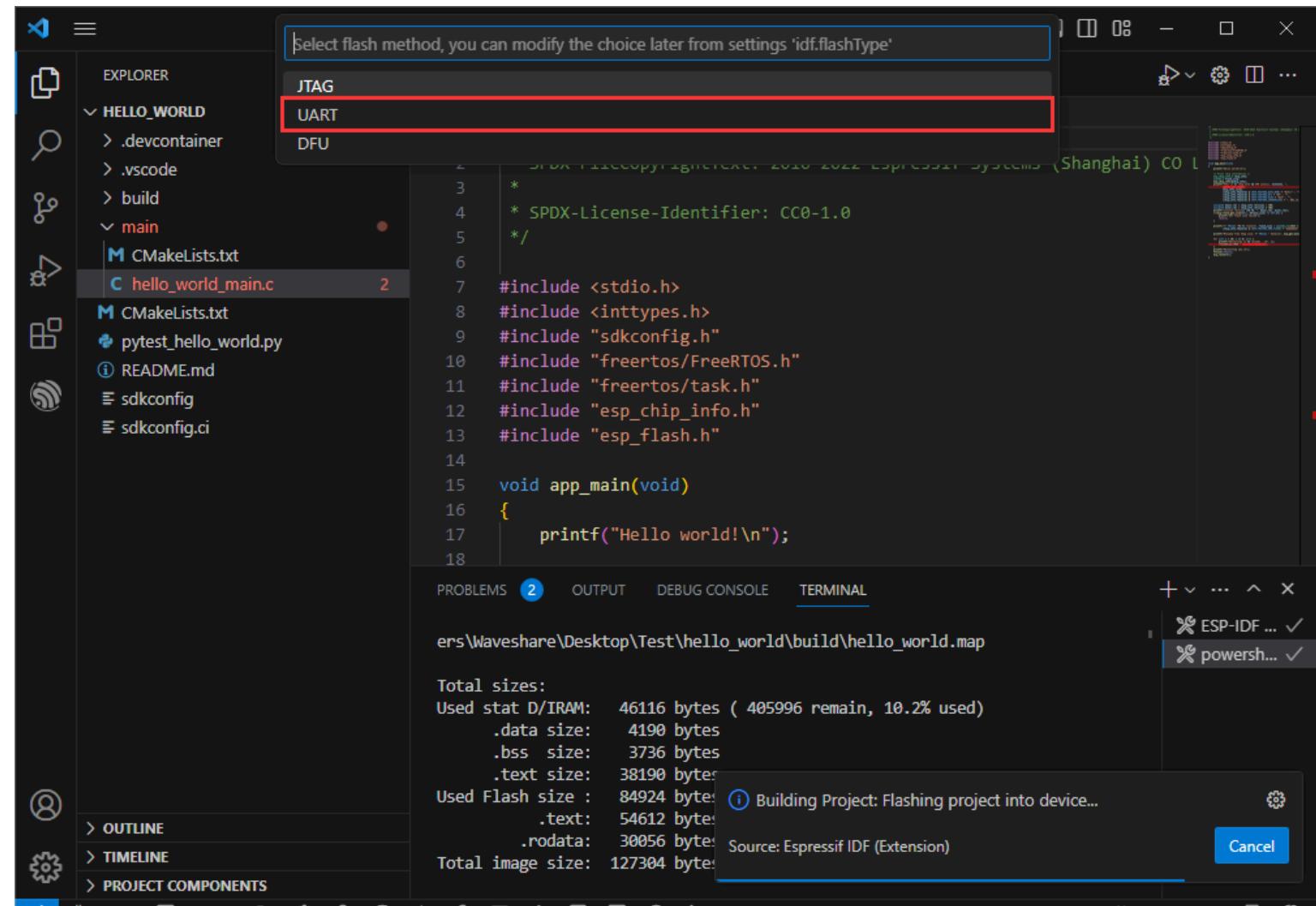
(/wiki/File:ESP32-C6-DEV-KIT-N8-25.png)

- 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.
 - If it is the first time to flash the program for a new project, 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).

```
/* 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 build 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 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 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 the standard header files and defines the entry point "app_main".
- Terminal:** Shows the output of the build process and the boot logs. It includes messages like "Hello world!" and information about the chip and memory usage.
- Bottom Bar:** Includes icons for serial port selection (COM3, esp32c6), connection status (UART, WiFi), and build options (Build, Flash, Run).

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

Demo Example

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 (**it is recommended to prioritize the COM port corresponding to the USB, which can be viewed through the device manager**), click on the compile and flash to run the demo.

```

1  /*
2  * SPDX-FileCopyrightText: 2010-2022 Espressif Systems (Shanghai) CO L
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 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)

GPIO

Official path: peripherals -> gpio -> generic_gpio.

Sample effect: LED blinks at 1-second intervals.

Hardware Connection

ESP32-C6	LED
----------	-----

GPIO18(or GPIO19)	LED+
GND	LED-

Software Operation

- Follow the tutorial above to create the official example generic_gpio. (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 COM port corresponding to the USB, which can be viewed through the device manager**), click compile and flash to run the demo.

The screenshot shows the Visual Studio Code (VS Code) interface for an ESP32-C6-DEV-KIT-N8 project named "generic_gpio".

- EXPLORER:** Shows the project structure:
 - GENERIC_GPIO:** Contains files like flash_args.in, generic_gpio.bin, generic_gpio.elf, generic_gpio.map, kconfigs_projbuild.in, kconfigs.in, ldgen_libraries, ldgen_libraries.in, partition-table-flash_args, project_description.json, project_elf_src_esp32c6.c, and x509_crt_bundle.S.
 - main:** Contains CMakeLists.txt and gpio_example_main.c (highlighted).
- EDITOR:** Displays the content of gpio_example_main.c, which includes defines for GPIO_OUTPUT_IO_0, GPIO_OUTPUT_IO_1, and GPIO_OUTPUT_PIN_SEL, along with comments explaining their binary representations.
- TERMINAL:** Shows the output of the build process and the result of a "make flash" command, indicating a successful flash operation to device 0.
- STATUS BAR:** Shows connection to COM3, board esp32c6, and various terminal settings like Spaces: 4, UTF-8, CRLF, and ESP-IDF.

(/wiki/File:ESP32-C6-DEV-KIT-N8-37.png)

- Go to the demo macro definition location to see what GPIOs are handled.

```
* Test:  
* Connect GPIO18(8) with GPIO4  
* Connect GPIO19(9) with GPIO5  
* Generate pulses on GPIO18(8)/19(9), that triggers interrupt on GPIOC  
*/  
#define GPIO_OUTPUT_IO_0    CONFIG_GPIO_OUTPUT_0  
#define GPIO_OUTPUT_IO_1    CONFIG_GPIO_OUTPUT_1  
#define GPIO_OUTPUT_PIN_SEL ((1ULL<<GPIO_OUTPUT_IO_0) | (1ULL<<GPIO_OUTPUT_IO_1))  
/*  
 * Let's say, GPIO_OUTPUT_IO_0=18, GPIO_OUTPUT_IO_1=19  
 * In binary representation,  
 * 1ULL<<GPIO_OUTPUT_IO_0 is equal to 0000000000000000000000000000001000000000000
```

(/wiki/File:ESP32-C6-DEV-KIT-N8-38.png)

- Right-click and select "Go to Declaration".



(/wiki/File:ESP32-C6-DEV-KIT-N8-39.png)

- The actual GPIOs are GPIO18, GPIO19.

(/wiki/File:ESP32-C6-DEV-KIT-N8-40.png)

RGB

Official example path: get-started -> blink.

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

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 COM port corresponding to USB, which can be viewed 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:** Shows the project structure under "UNTITLED (WORKSPACE)".
 - blink:** Contains ".devcontainer", ".vscode", and "build".
 - main:** Contains "blink_example_main.c" (highlighted), "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 comments about the Public Domain license and includes stdio.h, freertos/FreeRTOS.h, freertos/task.h, driver/gpio.h, esp_log.h, led_strip.h, and sdkconfig.h. It defines a static const char *TAG = "example"; and includes a note about choosing the GPIO in the project configuration menu.
- Terminal:** Shows the serial output from the ESP32:

```
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!
```

(/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 (https://www.waveshare.com/wiki/ESP32-C6-DEV-KIT-N8#Official_Demo_Usage_GUI_DE))
- 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 COM port corresponding to USB, which 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"

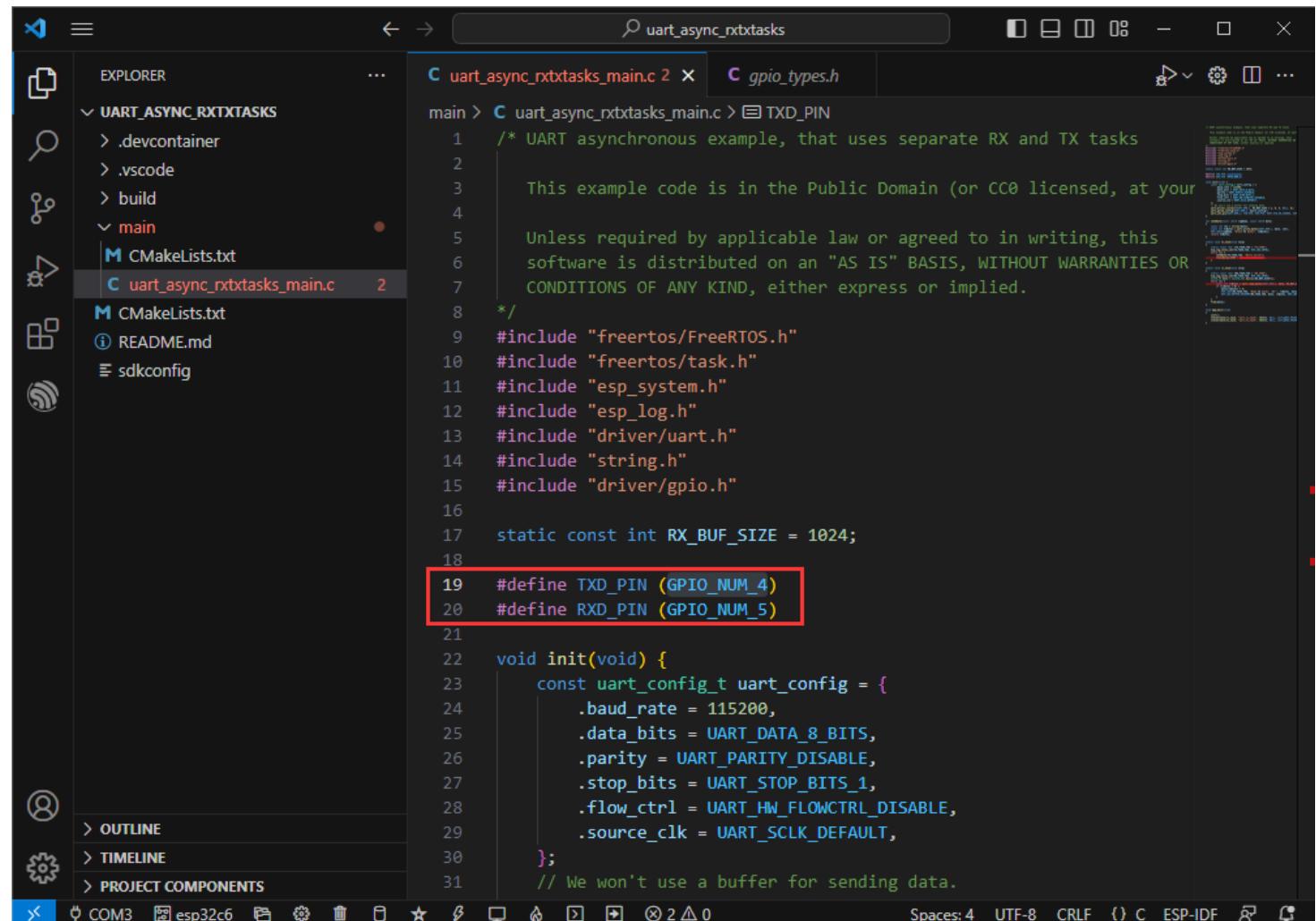
static const int RX_BUF_SIZE = 1024;

#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 line `#define TXD_PIN (GPIO_NUM_4)`, with the 'Go to Definition' option highlighted by a red box and the number 2.

```

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_D8,
26         .parity = UART_PARITY_NONE,
27         .stop_bits = UART_STOP_BITS_1,
28         .flow_ctrl = UART_HW_FLOWCTRL,
29         .source_clk = UART_1_SOURCE_XTAL
30     };
31     // We won't use a buffer
32     uart_driver_install(UART_NUM_1, RX_BUF_SIZE, TX_BUF_SIZE, 10, &uart_config, 0);
33     uart_param_config(UART_NUM_1, &uart_config);
34     uart_set_pin(UART_NUM_1, TXD_PIN, RXD_PIN, GPIO_NUM_1, GPIO_NUM_0);
35 }
36

```

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

I²C

The official example path: peripherals -> lcd-> i2c_oled.

Example effect: turns on the 0.96-inch OLED (A) (<https://www.waveshare.com/0.96inch-oled-a>.

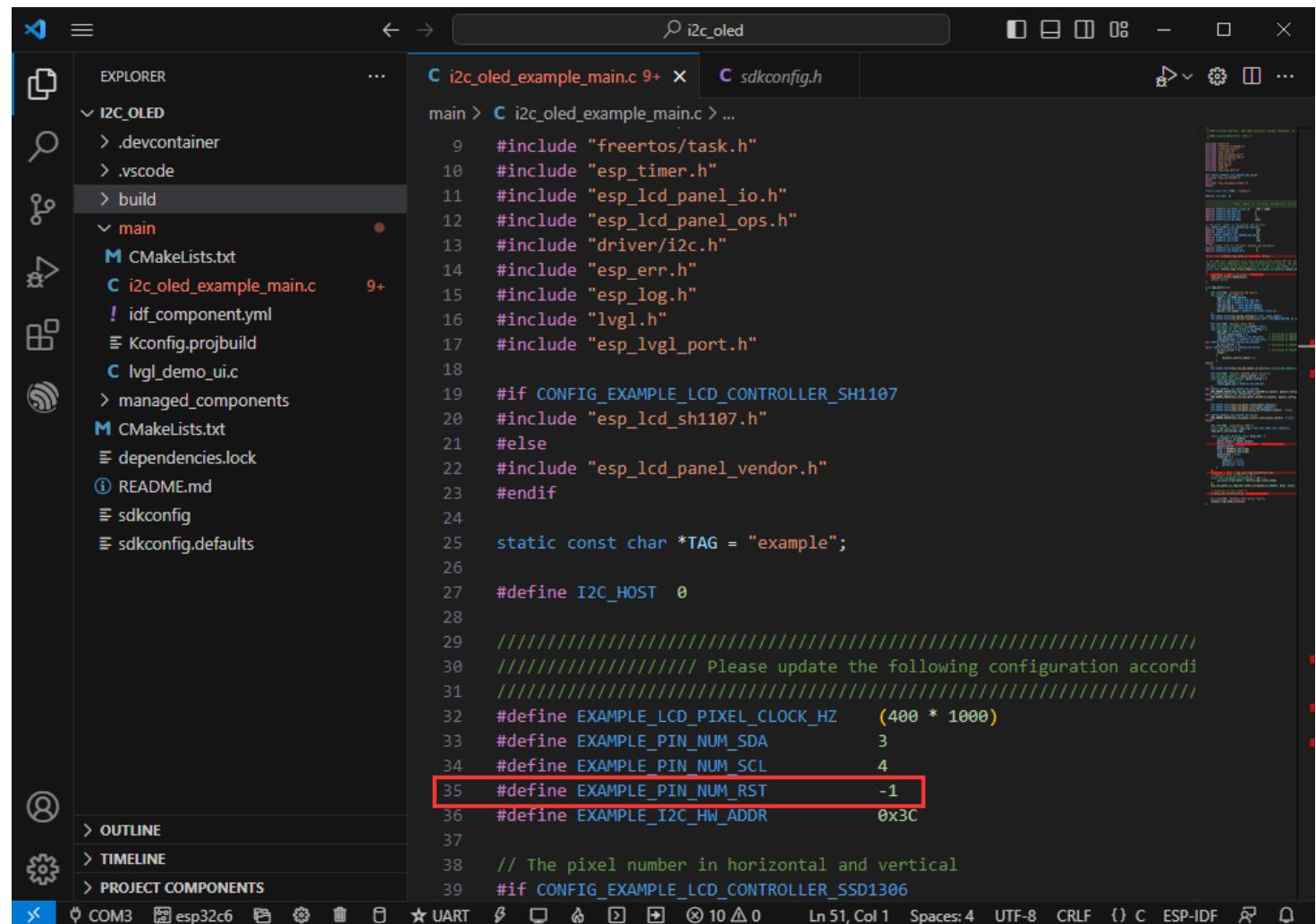
htm) and displays a character.

Hardware Connection

0.96inch OLED (A)	ESP32-C6
VCC	3V3
GND	GND
DIN	GPIO3
CLK	GPIO4
CS	GND
D/C	GND
RES	GPIO9

Software Operation

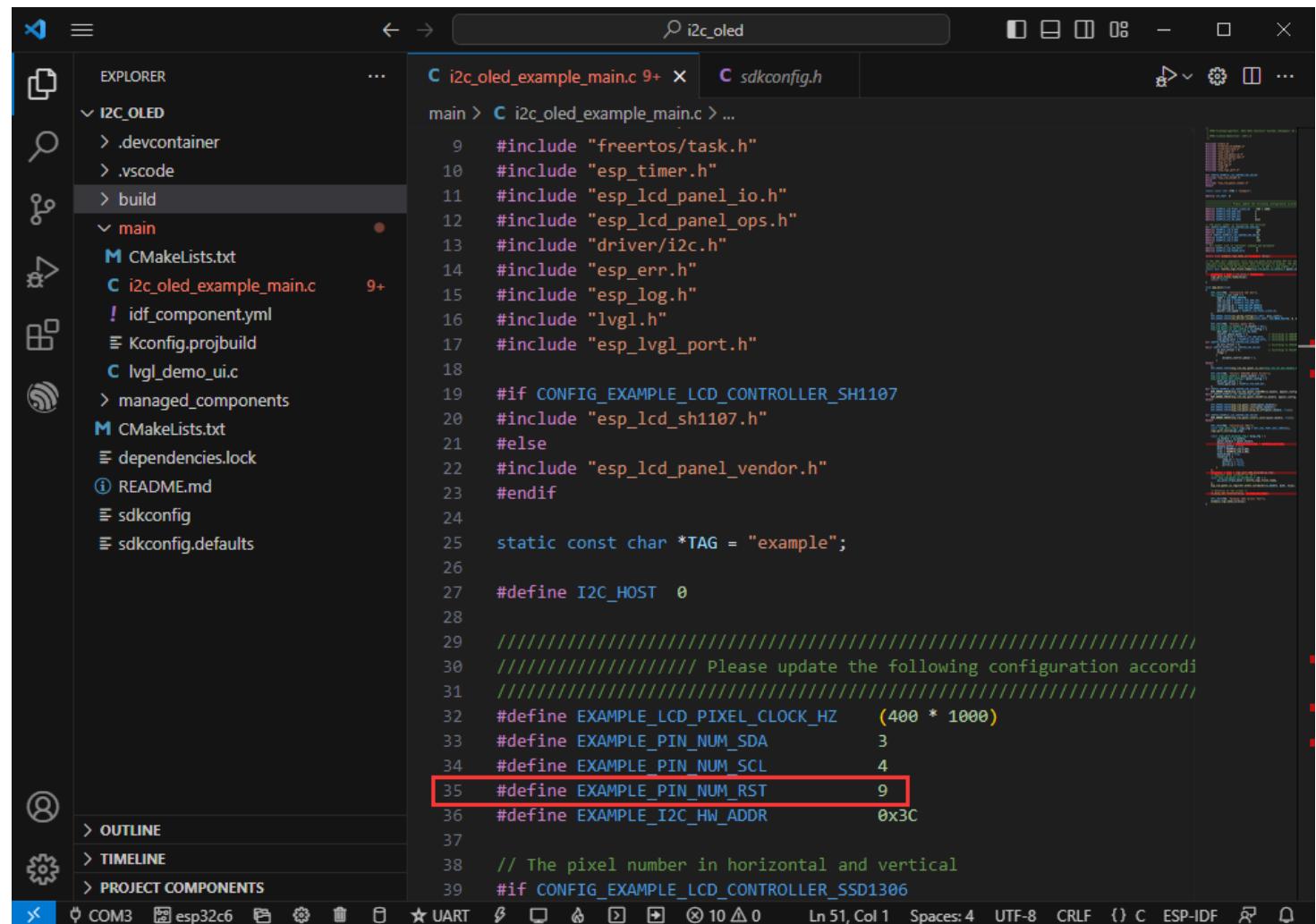
- Create the official example i2c_oled according to the tutorial above. (Create Example (https://www.waveshare.com/wiki/ESP32-C6-DEV-KIT-N8#Official_Demo_Usage_GUIDE))
- Modify the demo to be compatible with 0.96-inch OLED (A) (<https://www.waveshare.com/0.96-inch-oled-a.htm>).



```
main > C i2c_oled_example_main.c > ...
9 #include "freertos/task.h"
10 #include "esp_timer.h"
11 #include "esp_lcd_panel_io.h"
12 #include "esp_lcd_panel_ops.h"
13 #include "driver/i2c.h"
14 #include "esp_err.h"
15 #include "esp_log.h"
16 #include "lvgl.h"
17 #include "esp_lvgl_port.h"
18
19 #if CONFIG_EXAMPLE_LCD_CONTROLLER_SH1107
20 #include "esp_lcd_sh1107.h"
21 #else
22 #include "esp_lcd_panel_vendor.h"
23 #endif
24
25 static const char *TAG = "example";
26
27 #define I2C_HOST 0
28
29 /////////////////////////////////////////////////////////////////// Please update the following configuration according to your board
30 ///////////////////////////////////////////////////////////////////
31
32 #define EXAMPLE_PIXEL_CLOCK_HZ (400 * 1000)
33 #define EXAMPLE_PIN_NUM_SDA 3
34 #define EXAMPLE_PIN_NUM_SCL 4
35 #define EXAMPLE_PIN_NUM_RST -1
36 #define EXAMPLE_I2C_HW_ADDR 0x3C
37
38 // The pixel number in horizontal and vertical
39 #if CONFIG_EXAMPLE_LCD_CONTROLLER_SSD1306
```

(/wiki/File:ESP32-C6-DEV-KIT-N8-31.png)

- Adapts to 0.96-inch OLED (A), defines RES pin as GPIO9.



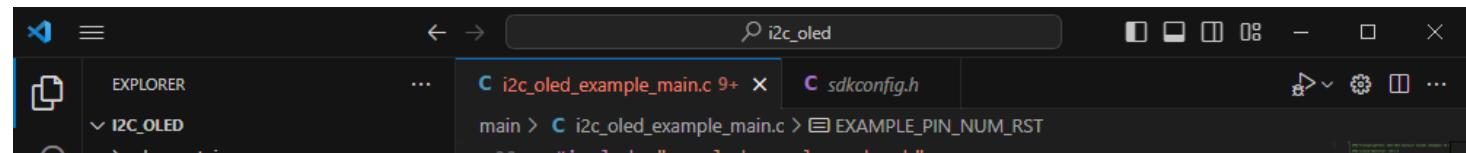
```

EXPLORER          i2c_oled_example_main.c 9+  X  sdkconfig.h
main > C i2c_oled_example_main.c > ...
9  #include "freertos/task.h"
10 #include "esp_timer.h"
11 #include "esp_lcd_panel_io.h"
12 #include "esp_lcd_panel_ops.h"
13 #include "driver/i2c.h"
14 #include "esp_err.h"
15 #include "esp_log.h"
16 #include "lvgl.h"
17 #include "esp_lvgl_port.h"
18
19 #if CONFIG_EXAMPLE_LCD_CONTROLLER_SH1107
20 #include "esp_lcd_sh1107.h"
21 #else
22 #include "esp_lcd_panel_vendor.h"
23 #endif
24
25 static const char *TAG = "example";
26
27 #define I2C_HOST 0
28
29 /////////////////////////////////////////////////////////////////// Please update the following configuration according to your board
30 /////////////////////////////////////////////////////////////////// Please update the following configuration according to your board
31 /////////////////////////////////////////////////////////////////// Please update the following configuration according to your board
32 #define EXAMPLE_PIXEL_CLOCK_HZ      (400 * 1000)
33 #define EXAMPLE_PIN_NUM_SDA         3
34 #define EXAMPLE_PIN_NUM_SCL         4
35 #define EXAMPLE_PIN_NUM_RST         9
36 #define EXAMPLE_I2C_HW_ADDR        0x3C
37
38 // The pixel number in horizontal and vertical
39 #if CONFIG_EXAMPLE_LCD_CONTROLLER_SSD1306

```

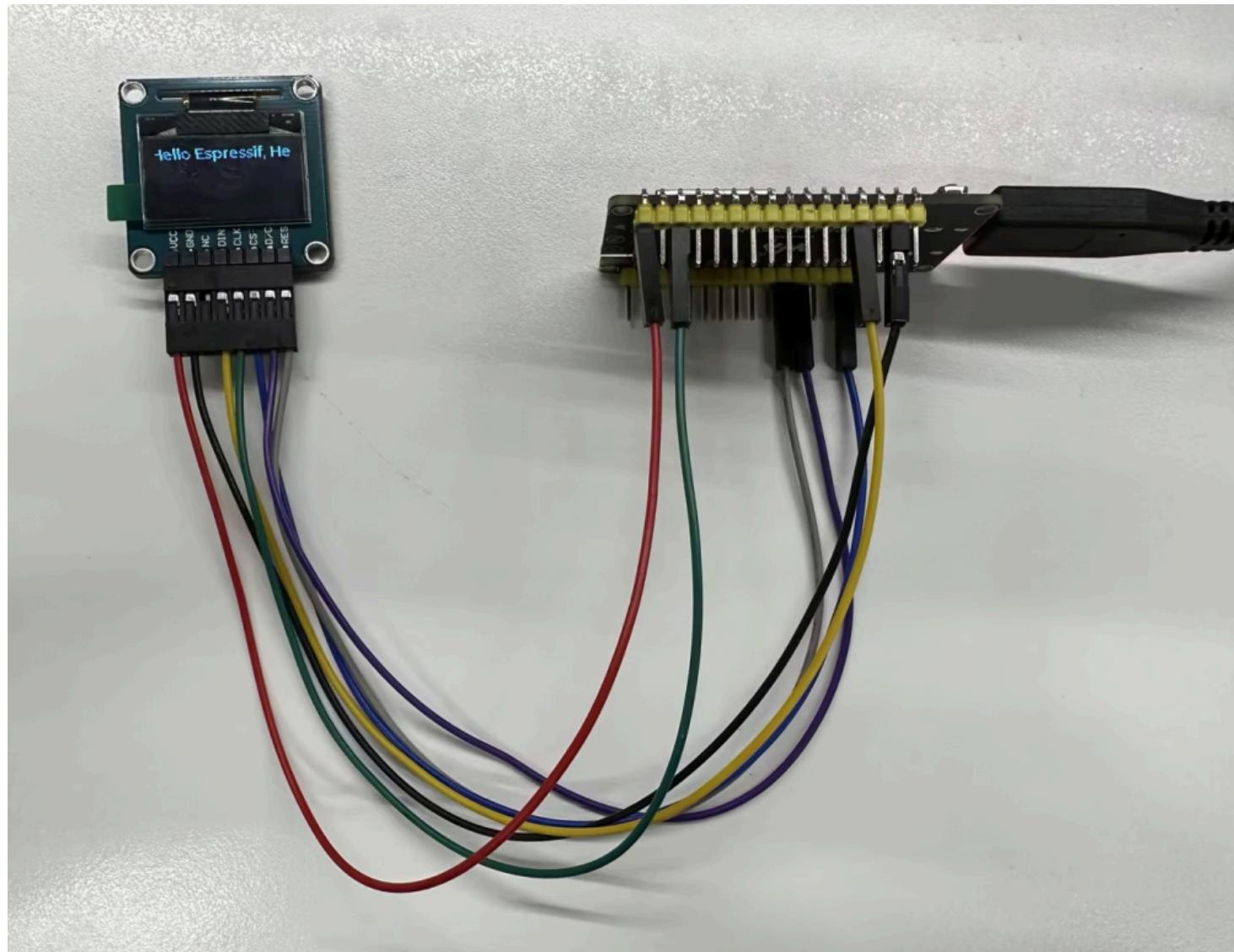
(/wiki/File:ESP32-C6-DEV-KIT-N8-32.png)

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



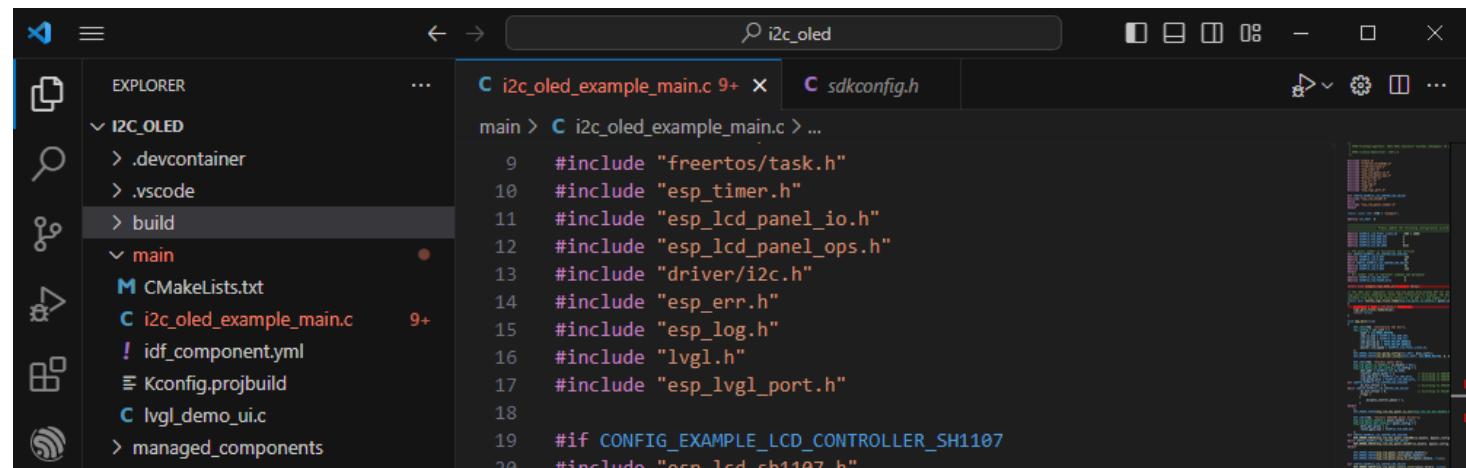
(/wiki/File:ESP32-C6-DEV-KIT-N8-35.png)

- The effect is as shown below:



(/wiki/File:ESP32-C6-DEV-KIT-N8-33.png)

- You can view the actual use of GPIO:



The screenshot shows the Visual Studio Code interface for an ESP32 project named "i2c_oled". The left sidebar displays the project structure:

- EXPLORER: Shows files and folders like ".devcontainer", ".vscode", "build", "main", "CMakeLists.txt", "i2c_oled_example_main.c", "idf_component.yml", "Kconfig.projbuild", "lvgl_demo_ui.c", and "managed_components".
- SEARCH: A search bar at the top right contains the text "i2c_oled".
- PROBLEMS: A small icon indicates no errors.

The main editor area shows the content of "i2c_oled_example_main.c" with line numbers 9 through 20. The code includes various header file includes such as "freertos/task.h", "esp_timer.h", "esp_lcd_panel_io.h", "esp_lcd_panel_ops.h", "driver/i2c.h", "esp_err.h", "esp_log.h", "lvgl.h", and "esp_lvgl_port.h". It also features a conditional compilation directive "#if CONFIG EXAMPLE LCD CONTROLLER SH1107" followed by "#include <esp_lcd_sh1107.h>".

(/wiki/File:ESP32-C6-DEV-KIT-N8-34.png)

SPI

The path of the official example: peripherals -> spi_master-> lcd.

The example effect: dynamic displaying picture effect on the 2.4inch LCD Module (<https://ww>

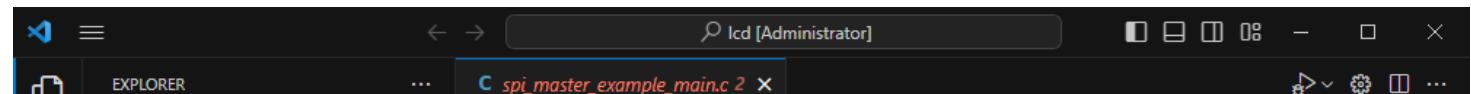
w.waveshare.com/2.4inch-lcd-module.htm).

Hardware Connection

2.4inch LCD Module	ESP32-C6
VCC	3V3
GND	GND
DIN	GPIO7
CLK	GPIO6
CS	GPIO10
D/C	GPIO11
RES	GPIO4
BL	GPIO5

Software Operation

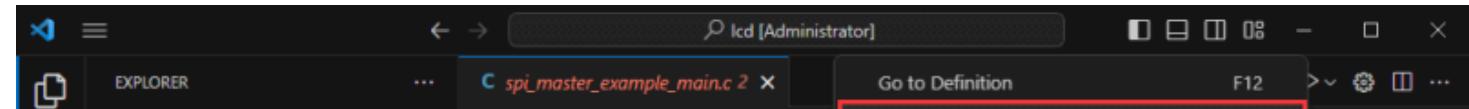
- Right-click on the VScode icon and run VScode as administrator:



(/wiki/File:ESP32-C6_TO_Sample_15.png)

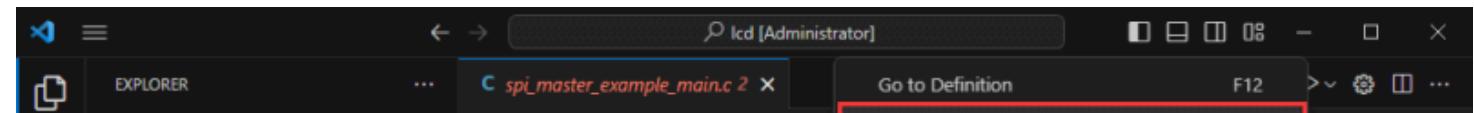
- Create the official example lcd according to the tutorial above. (Create Example (https://www.waveshare.com/wiki/ESP32-C6-DEV-KIT-N8#Official_Demo_Usage_GUIDE))

Modify the program to make it compatible with a 2.4inch LCD Module (<https://www.waveshare.com/2.4inch-lcd-module.htm>).



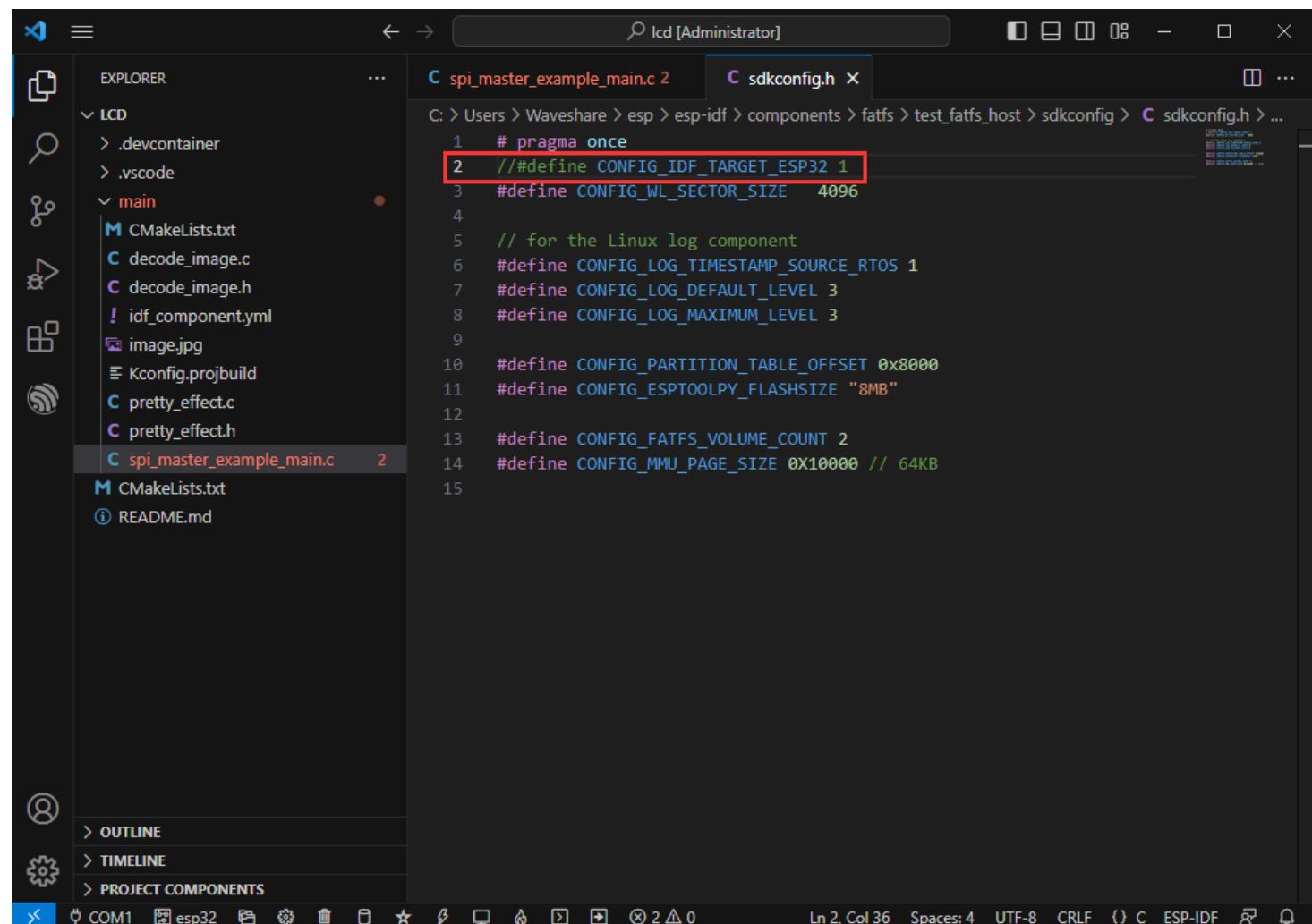
(/wiki/File:ESP32-C6-DEV-KIT-N8-36.png)

- Select "Go to Declaration".



(/wiki/File:ESP32-C6-DEV-KIT-N8-36.png)

- Note that currently using "ESP32-C6", and then blocks other chip definitions.



The screenshot shows a Windows 10 desktop environment with a dark-themed Visual Studio Code (VS Code) window. The title bar of the window reads "Lcd [Administrator]". The left sidebar shows a file tree for a project named "LCD" containing ".devcontainer", ".vscode", and a "main" folder with files like "CMakeLists.txt", "decode_image.c", "decode_image.h", "idf_component.yml", "image.jpg", "Kconfig.projbuild", "pretty_effect.c", "pretty_effect.h", and "spi_master_example_main.c". The "spi_master_example_main.c" file is currently selected. The main editor area displays the "sdkconfig.h" file with the following content:

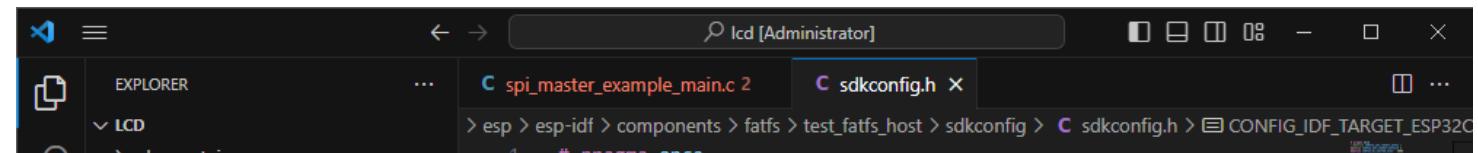
```
C: > Users > Waveshare > esp > esp-idf > components > fatfs > test_fatfs_host > sdkconfig > C sdkconfig.h > ...
1 # pragma once
2 // #define CONFIG_IDF_TARGET_ESP32 1
3 #define CONFIG_WL_SECTOR_SIZE 4096
4
5 // for the Linux log component
6 #define CONFIG_LOG_TIMESTAMP_SOURCE_RTOS 1
7 #define CONFIG_LOG_DEFAULT_LEVEL 3
8 #define CONFIG_LOG_MAXIMUM_LEVEL 3
9
10 #define CONFIG_PARTITION_TABLE_OFFSET 0x8000
11 #define CONFIG_ESPTOOLPY_FLASHSIZE "8MB"
12
13 #define CONFIG_FATFS_VOLUME_COUNT 2
14 #define CONFIG_MMU_PAGE_SIZE 0x10000 // 64KB
15
```

The line "#define CONFIG_IDF_TARGET_ESP32 1" is highlighted with a red rectangular box. The status bar at the bottom of the VS Code window shows "Ln 2, Col 36" and other standard build and file navigation icons.

(/wiki/File:ESP32-C6-DEV-KIT-N8-036.png)

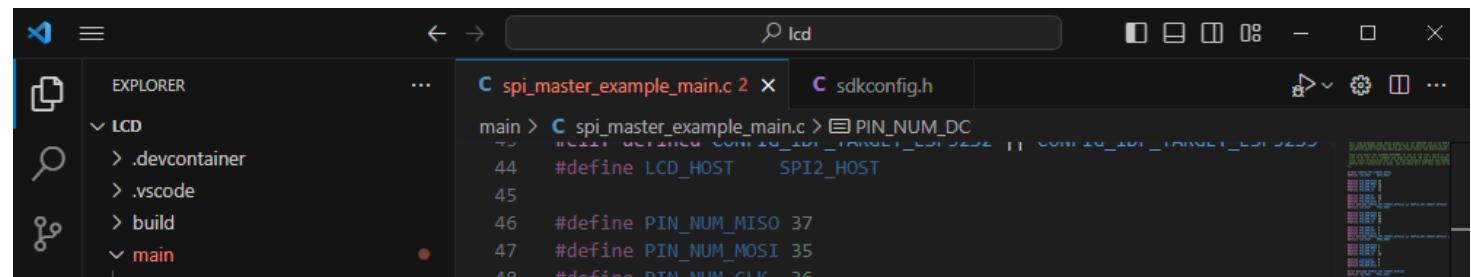
- And add macro-define ESP32-C6, **CONFIG_IDF_TARGET_ESP32C6**.

```
//#define CONFIG_IDF_TARGET_ESP32 1
#define CONFIG_IDF_TARGET_ESP32C6 1
```



(/wiki/File:ESP32-C6-DEV-KIT-N8-42.png)

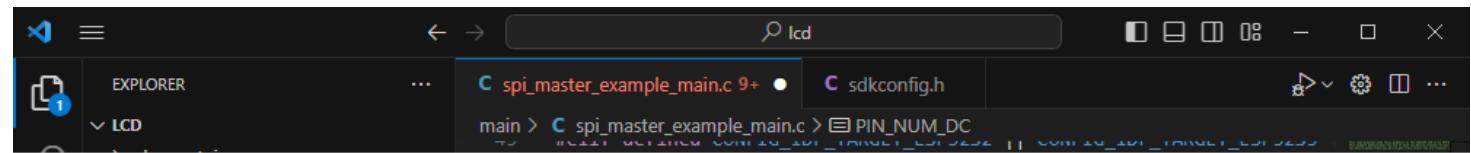
- Modify D/C using IO.
 - Enter the 62 line **spi_master_example_main.c**.



```
main > C spi_master_example_main.c 2 X C sdkconfig.h
44 #define LCD_HOST      SPI2_HOST
45
46 #define PIN_NUM_MISO 37
47 #define PIN_NUM_MOSI 35
48 #define PIN_NUM_CLK  36
```

(/wiki/File:ESP32-C6-DEV-KIT-N8-43.png)

- Modify D/C using IO to avoid the download circuit (using GPIO11 instead of GPIO9).

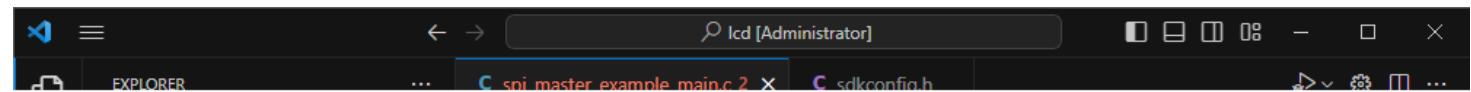


(/wiki/File:ESP32-C6-DEV-KIT-N8-44.png)

- Modify the backlight.

(/wiki/File:ESP32-C6-DEV-KIT-N8-45.png)

- Modify it as **gpio_set_level(PIN_NUM_BCKL, 1);**

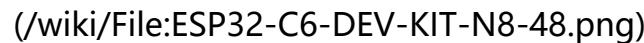


(/wiki/File:ESP32-C6-DEV-KIT-N8-46.png)

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

(/wiki/File:ESP32-C6-DEV-KIT-N8-47.png)

- The effect is as shown below:



(/wiki/File:ESP32-C6-DEV-KIT-N8-48.png)

Bluetooth

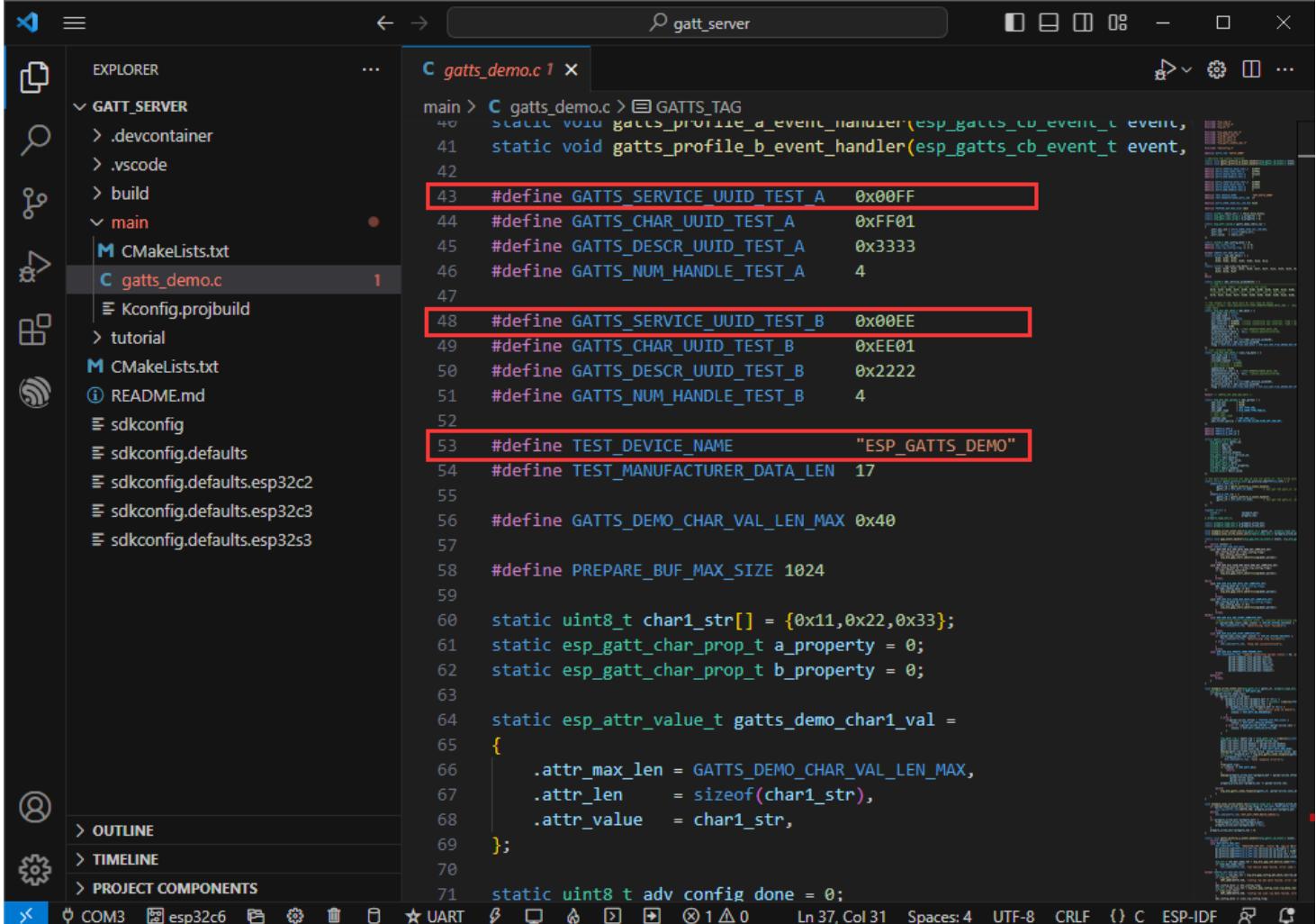
Official sample path: Bluetooth -> bluedroid -> ble -> gatt_server.

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 (https://www.waveshare.com/wiki/ESP32-C6-DEV-KIT-N8#Official_Demo_Usage_GUIDE))

- 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**.



```

EXPLORER          gatt_server
GATT_SERVER
  .devcontainer
  .vscode
  build
    main
      CMakeLists.txt
      gatts_demo.c 1
      Kconfig.projbuild
      tutorial
      CMakeLists.txt
      README.md
      sdkconfig
      sdkconfig.defaults
      sdkconfig.defaults.esp32c2
      sdkconfig.defaults.esp32c3
      sdkconfig.defaults.esp32s3

C gatts_demo.c X
main > C gatts_demo.c > GATT_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 COM port corresponding to USB, which can be viewed through the device manager**), click compile and flash to run the demo.

```
main > C gatts_demo.c > ...
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 (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)

- Connect the ESP_GATTS_DEMO Bluetooth device on the phone.
- The effect of a successful connection is shown below:

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

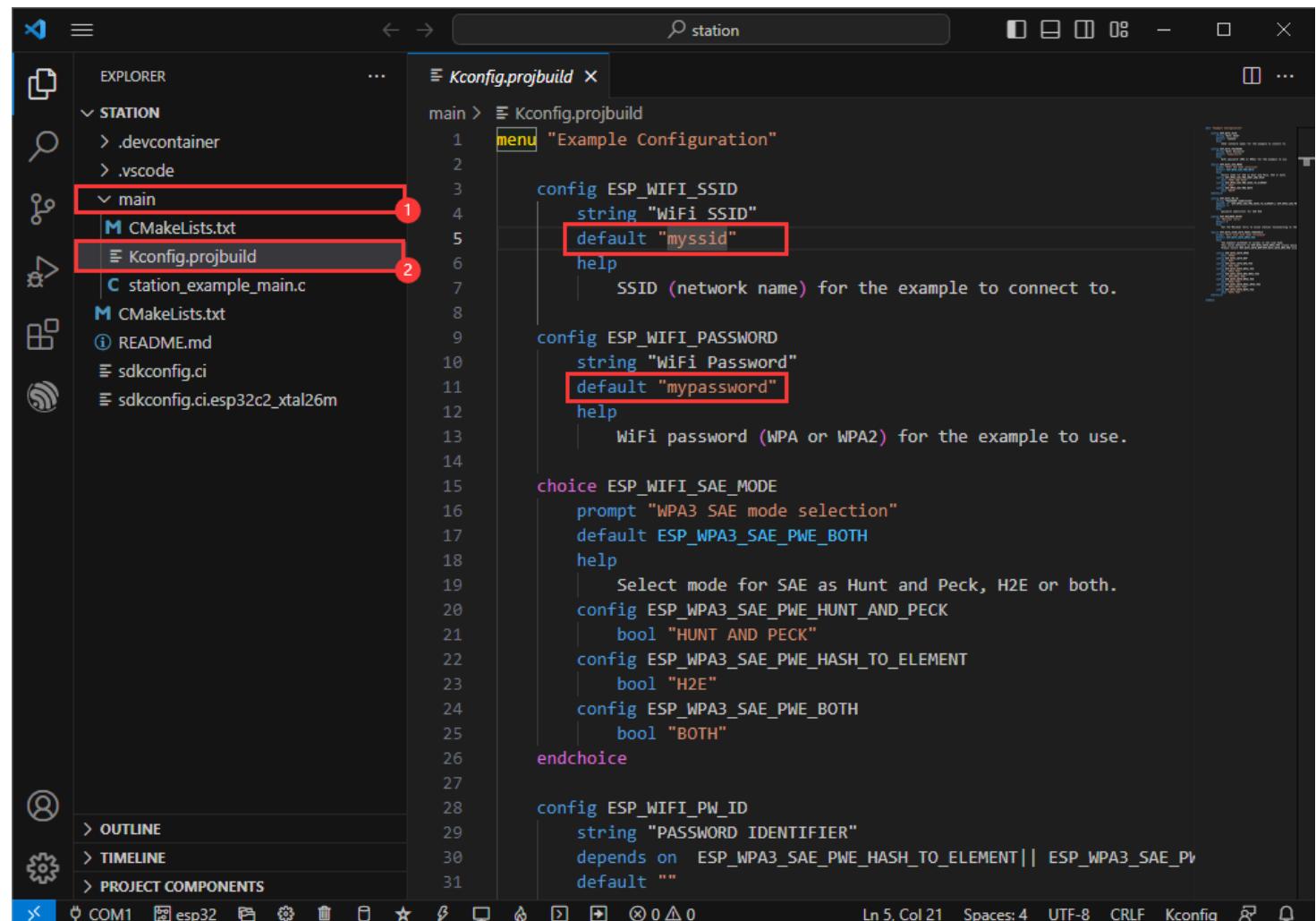
Official example path: wifi-> getting_started-> station.

Example effect: ESP32-C6 connects to WIFI.

Software Operation

- Create the official example station according to the tutorial above. (Create Example (https://www.waveshare.com/wiki/ESP32-C6-DEV-KIT-N8#Official_Demo_Usage_GUIDE))

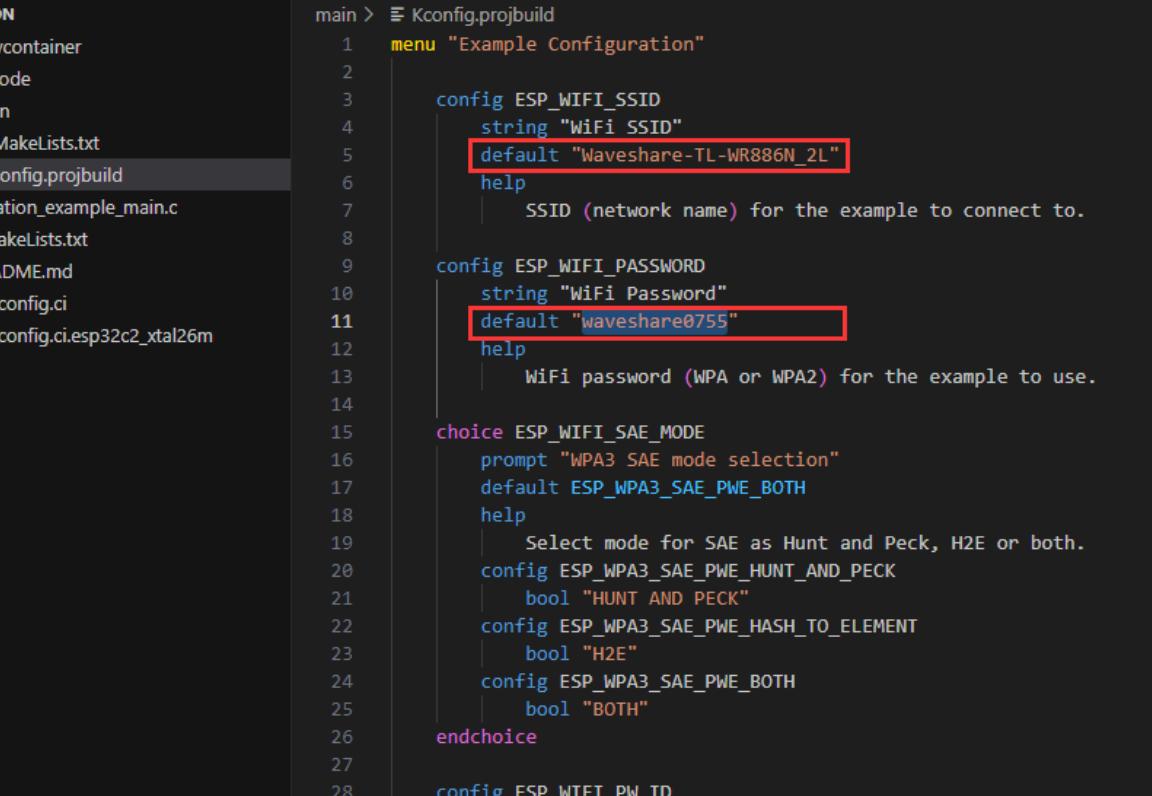
- Modify the contents of the demo to connect to the required WiFi.
- Go to the **Kconfig.projbuild** file.



```
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_PN_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.



```
main > Kconfig.projbuild
      1 menu "Example Configuration"
      2
      3 config ESP_WIFI_SSID
      4     string "WiFi SSID"
      5     default "Waveshare-TL-WR886N_2L"
      6     help
      7         SSID (network name) for the example to connect to.
      8
      9 config ESP_WIFI_PASSWORD
     10    string "WiFi Password"
     11    default "waveshare0755"
     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-N8-61.png)

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

The screenshot shows the Visual Studio Code interface for an ESP32 project. The Explorer sidebar on the left lists files and folders, including .devcontainer, .vscode, build, main (containing CMakeLists.txt, Kconfig.projbuild, and station_example_main.c), README.md, sdkconfig, sdkconfig.ci, sdkconfig.ci.esp32c2_xtal26m, and sdkconfig.old. The station_example_main.c file is currently selected. The code editor displays the following content:

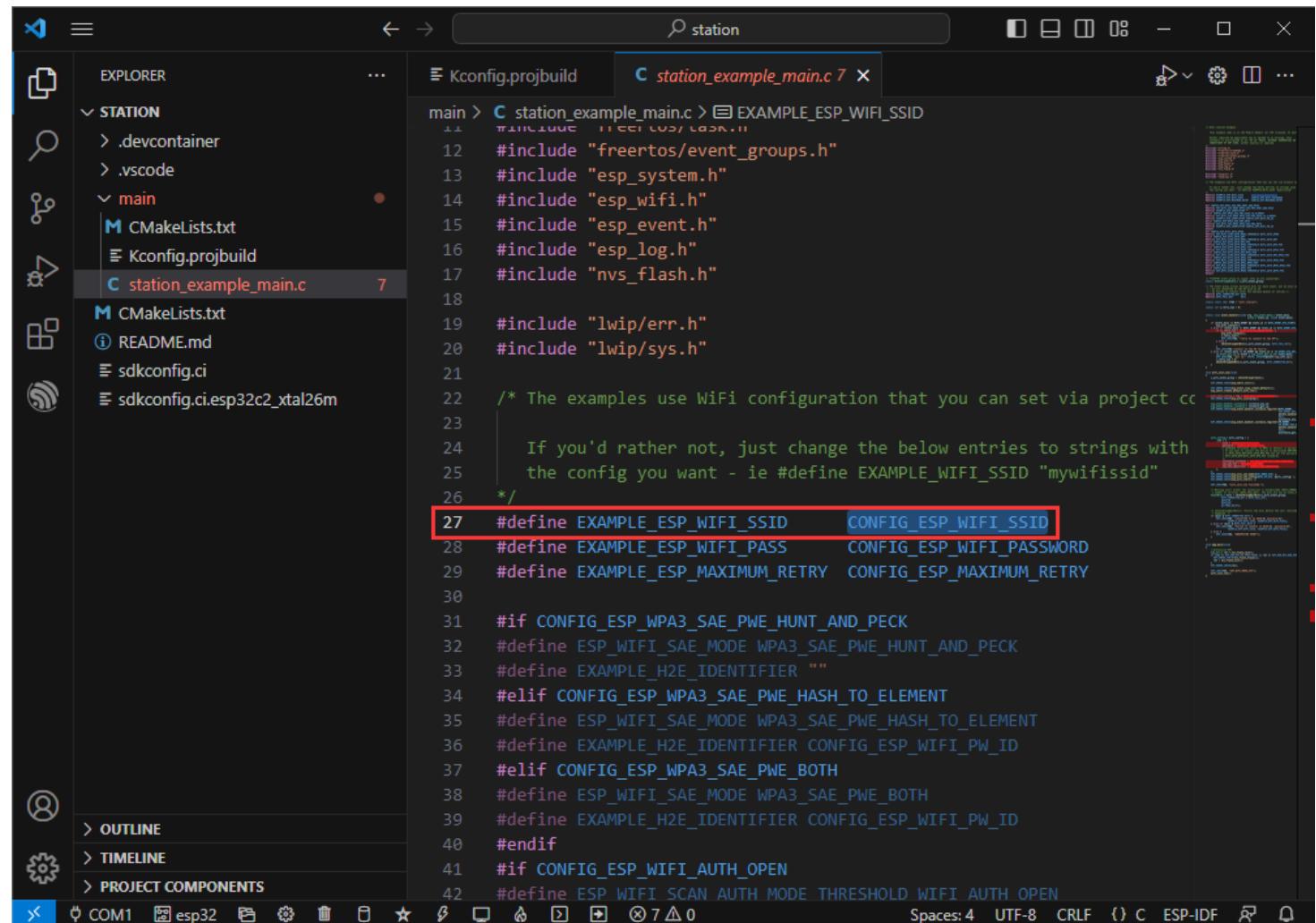
```
main > C station_example_main.c > EXAMPLE_ESP_WIFI_SSID
12 #include "freertos/event_groups.h"
13 #include "esp_system.h"
14 #include "esp_wifi.h"
15 #include "esp_event.h"
16 #include "esp_log.h"
17 #include "nvs_flash.h"
18
19 #include "lwip/err.h"
20 #include "lwip/sys.h"
21
22 /* The examples 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
```

The terminal tab at the bottom shows the following serial port output:

```
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.

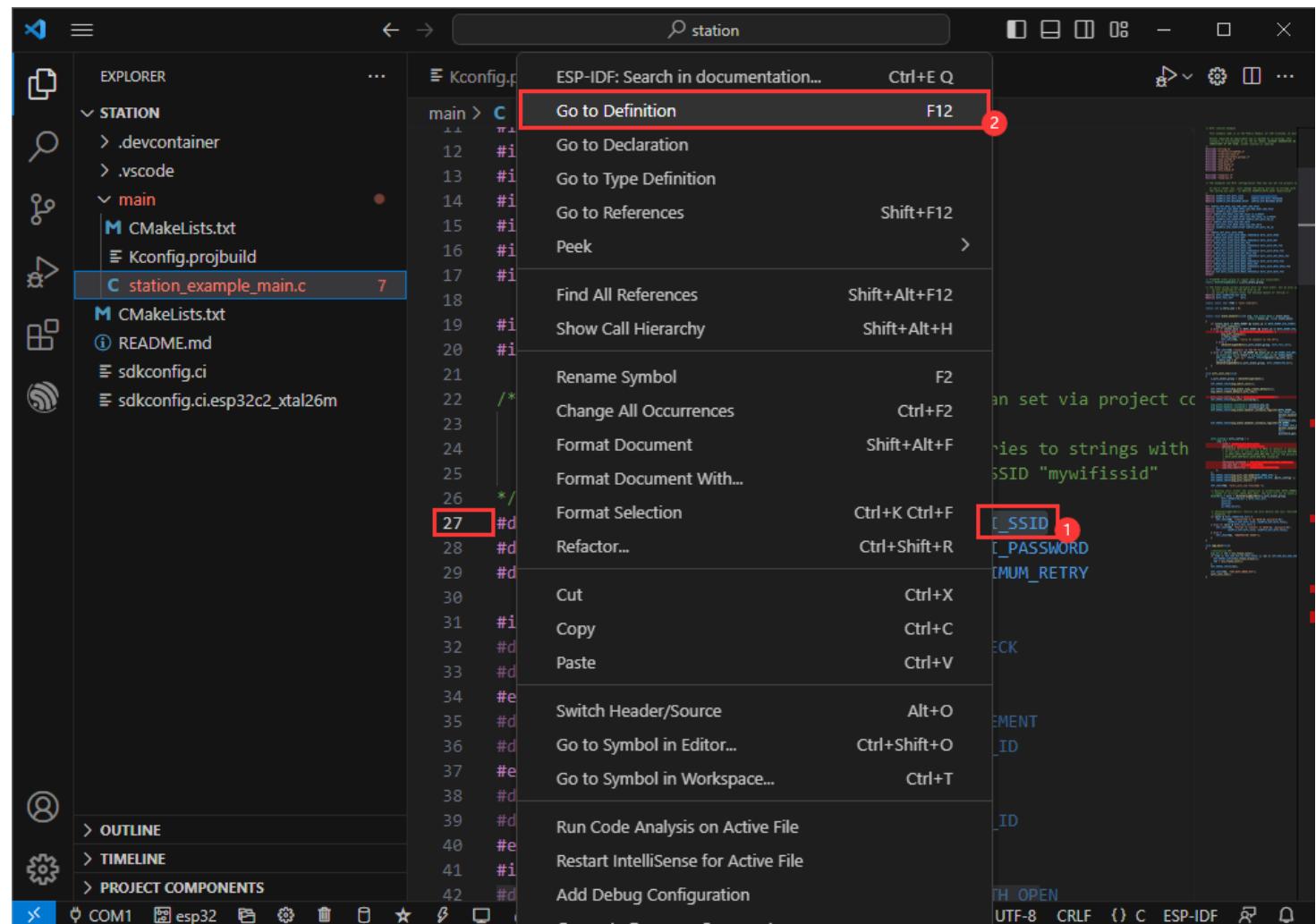


```
main > C station_example_main.c > EXAMPLE_ESP_WIFI_SSID
12 #include "freertos/event_groups.h"
13 #include "esp_system.h"
14 #include "esp_wifi.h"
15 #include "esp_event.h"
16 #include "esp_log.h"
17 #include "nvs_flash.h"
18
19 #include "lwip/err.h"
20 #include "lwip/sys.h"
21
22 /* The examples 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 the definition.



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

- The previously set value can be seen as:

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

- EXPLORER:** Shows the project structure with files like .devcontainer, .vscode, build, bootloader, CMakeFiles, config, kconfig_menus.json, sdkconfig.cmake, sdkconfig.h (selected), sdkconfig.json, esp-idf, log, partition_table, .bin_timestamp, .ninja_deps, .ninja_log, app-flash_args, bootloader-flash_args, build.ninja, cmake_install.cmake, CMakeCache.txt, compile_commands.json, config.env, and OUTLINE, TIMELINE, PROJECT COMPONENTS.
- SEARCH:** Search bar at the top right with the text "station".
- EDITOR:** The code editor displays the `station_example_main.c` file, which includes the `station` header. A red box highlights the following configuration lines in the `station` section of `station_example_main.c`:


```

382 #define CONFIG_ESP_WIFI_SSID "Waveshare-TL-WR886N_2L"
383 #define CONFIG_ESP_WIFI_PASSWORD "waveshare0755"
      
```
- STATUS BAR:** Shows the connection COM3, port esp32c6, baud rate 115200, and other build-related information.

(/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 (https://www.waveshare.com/wiki/ESP32-C6-DEV-KIT-N8#Official_Demo_Usage_GUIDE))
- 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 COM port corresponding to USB, which 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 project files 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 esp_zb_switch.c file is currently selected and displayed in the main editor area. The editor shows code related to Zigbee network formation, including comments about the MIT license and the start of network formation. The Terminal tab at the bottom displays log messages from the ESP32 chip, such as phy_init, phy_version, and main_task, indicating successful network formation and steering. 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 (http://www.waveshare.com/wiki/ESP32-C6-DEV-KIT-N8#Official_Demo_Usage_GUIDE))
- The demo is compatible with ESP32-C6 and can be used without modifying the demo content.

- 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:** Shows the project structure under "HA_ON_OFF_LIGHT". The file "esp_zb_light.c" is selected.
- CODE EDITOR:** Displays 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 for commissioning.
- TERMINAL:** Shows log output from the ESP32 board, indicating the main task starting, GPIO configuration, phy_init, and Zigbee stack initialization.
- STATUS BAR:** Shows the COM port as COM3, driver as esp32c6, and other system information like UART, CRLF, and file encoding.

(/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.

```

C esp_zb_light.c 1
main > C esp_zb_light.c > ...
35     * POSSIBILITY OF SUCH DAMAGE.
36     */
37
38 #include "esp_log.h"
39 #include "freertos/FreeRTOS.h"
40 #include "freertos/task.h"
41 #include "ha/esp_zigbee_ha_standard.h"
42 #include "esp_zb_light.h"
43 #include "nvs_flash.h"
44
45 /**
46     * @note Make sure set idf.py menuconfig in zigbee component as zigbee endpoint
47     */
48 #if !defined ZB_ED_ROLE
49 #error Define ZB_ED_ROLE in idf.py menuconfig to compile light (End Device)
50 #endif
51
52 static const char *TAG = "ESP_ZB_ON_OFF_LIGHT";
53 //***** Define functions *****/
54 static void bdb_start_top_level_commissioning_cb(uint8_t mode_mask)
55 {
56
57 }

```

PROBLEMS 1 OUTPUT DEBUG CONSOLE TERMINAL

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

(/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 (https://www.waveshare.com/wiki/ESP32-C6-DEV-KIT-N8#Official_Demo_Usage_GUI_DE)

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

The screenshot shows the VS Code interface with a dark theme. The Explorer sidebar on the left lists project files: .devcontainer, .vscode (with c_cpp_properties.json and launch.json), build, main (with CMakeLists.txt, hello_world_main.c, and other build files), and CMakeLists.txt, pytest_hello_world.py, README.md, sdkconfig, and sdkconfig.ci. The launch.json file is selected in the Explorer and is also the active tab in the main editor area. The code in launch.json is highlighted with a red rectangle and numbered 1 through 10 to indicate specific parts:

```
1 {  
2     "version": "0.2.0",  
3     "configurations": [  
4         {  
5             "type": "espifdf",  
6             "name": "Launch",  
7             "request": "launch"  
8         }  
9     ]  
10 }
```

The status bar at the bottom shows various tool icons and the text: Spaces: 2, UTF-8, LF, JSON with Comments, ESP-IDF, and a bell icon.

(/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  
            }  
        }  
    ]  
}
```

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

- Explorer View:** Shows the project structure under "HELLO_WORLD".
- Search View:** A magnifying glass icon.
- File View:** A folder icon with a blue dot.
- Task View:** A gear icon.
- Output View:** An "OUTLINE" tab is selected.
- Terminal View:** A terminal window showing "COM3" and "esp32c6".

The main editor area displays the `launch.json` file content, which is highlighted with a red border:

```
1  {
2    "version": "0.2.0",
3    "configurations": [
4      {
5        "name": "GDB",
6        "type": "cppdbg",
7        "request": "launch",
8        "MIMode": "gdb",
9        "miDebuggerPath": "${command:espIdf.getXensaGdb}",
10       "program": "${workspaceFolder}/build/${command:espIdf.getProjectName}",
11       "windows": {
12         "program": "${workspaceFolder}\\build\\${command:espIdf.getProjectName}",
13       },
14       "cwd": "${workspaceFolder}",
15       "environment": [
16         { "name": "PATH", "value": "${config:idf.customEnvPath}" }
17       ],
18       "setupCommands": [
19         { "text": "target remote :3333" },
20         { "text": "set remote hardware-watchpoint-limit 2" },
21         { "text": "mon reset halt" },
22         { "text": "thb app_main" },
23         { "text": "flushregs" }
24       ],
25       "externalConsole": false,
26       "logging": {
27         "engineLogging": true
28       }
29     }
30   }
```

A blue button labeled "Add Configuration..." is located at the bottom right of the code editor.

(/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 "app_main" function which prints "Hello world!\\n".
- Terminal:** Shows the output of the build and run process:

```
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...
```
- Taskbar:** Shows multiple open terminals, with the current one labeled "ESP-IDF".

(/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:

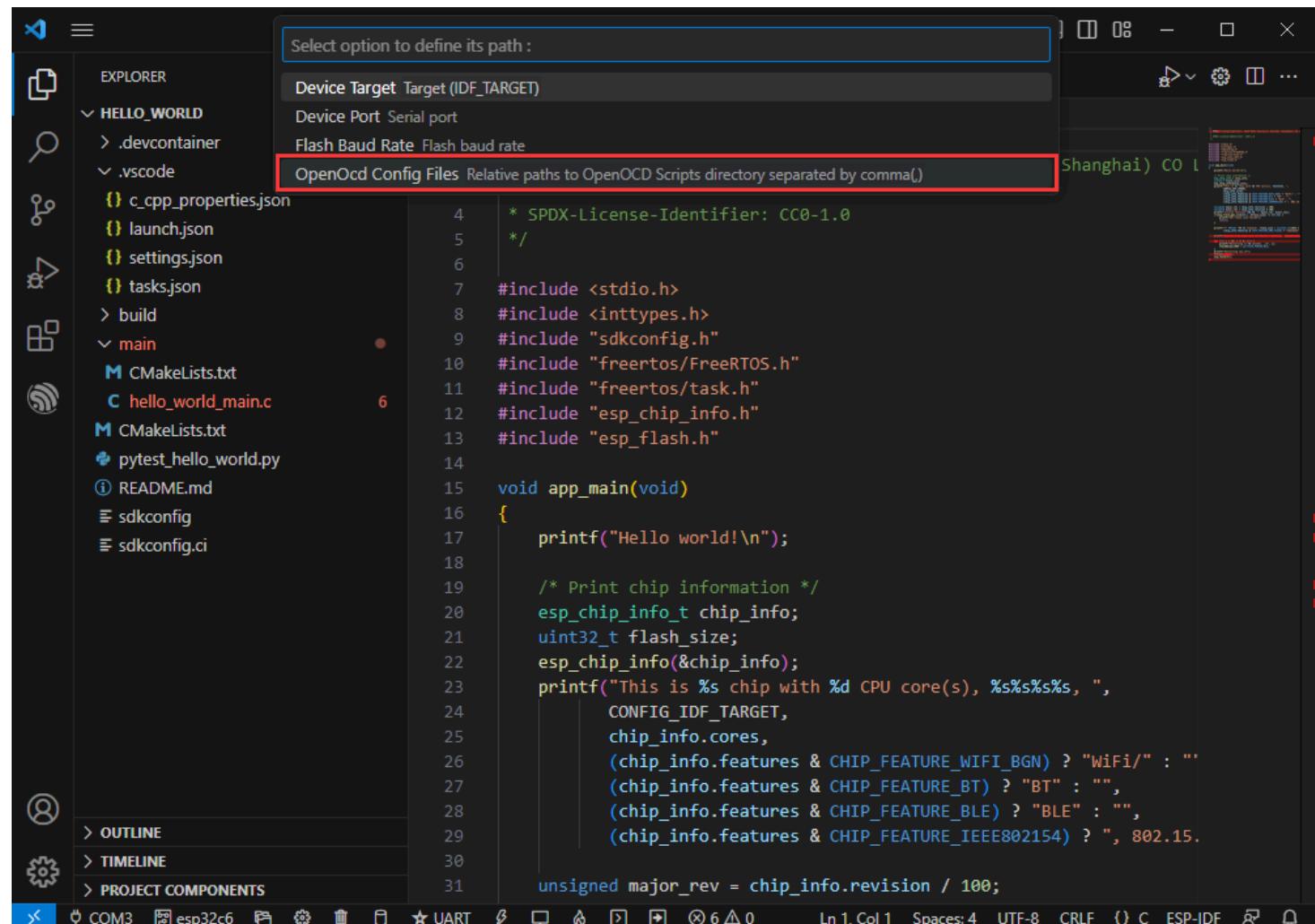
- Explorer View:** Shows the project structure under "HELLO_WORLD".
 - Subfolders: ".devcontainer", ".vscode".
 - Files: "c_cpp_properties.json", "launch.json", "settings.json", "tasks.json", "build", "main".
 - Content of "main": "CMakeLists.txt", "hello_world_main.c", "CMakeLists.txt", "pytest_hello_world.py",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

```
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;
```

The code prints "Hello world!" and then prints information about the chip, including its name, number of cores, and supported features.

(/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 'HELLO_WORLD'. The 'main' folder contains files like 'hello_world_main.c', 'CMakeLists.txt', and 'sdkconfig'. In the center, a code editor window shows a C file with code for a 'Hello world!' application. At the top of the code editor, there is a status bar with various icons and text. A red box highlights the 'OpenOCD Config Files' input field in the status bar, which contains the relative path 'board/esp32c6-builtin.cfg'. The status bar also includes other information such as 'Ln 1, Col 1', 'Spaces: 4', 'UTF-8', 'CRLF', and '{} C ESP-IDF'.

(/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

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

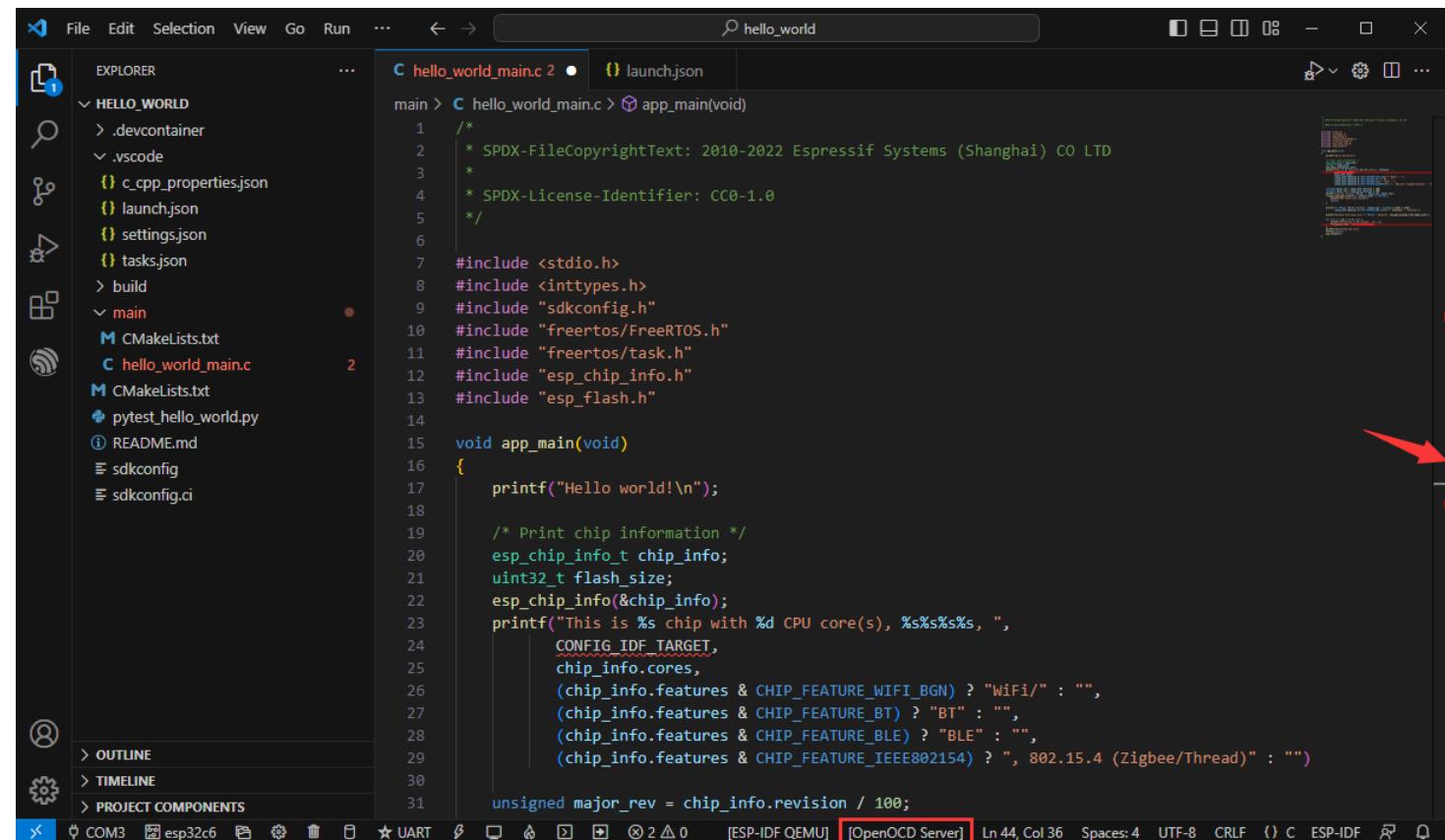
- Explorer View:** Shows the project structure under "HELLO_WORLD".
 - Subfolders: ".devcontainer", ".vscode".
 - Files: "c_cpp_properties.json", "launch.json", "settings.json", "tasks.json", "build", "main".
 - Content of "main": "CMakeLists.txt", "hello_world_main.c", "CMakeLists.txt", "pytest_hello_world.py", "README.md", "sdkconfig", "sdkconfig.ci".
- Search View:** Shows search results for "board/esp32c6-builtin.cfg".
- Code Editor:** Displays the "hello_world_main.c" file content.
- Bottom Status Bar:** Shows connection status (COM3), board selection (esp32c6), terminal tabs, and other system information.

```
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 with the following details:

- File Bar:** File, Edit, Selection, View, Go, Run, ...
- Search Bar:** hello_world
- Explorer:** HELLO_WORLD folder containing .devcontainer, .vscode, CMakeLists.txt, hello_world_main.c, launch.json, README.md, sdkconfig, and sdkconfig.ci.
- Editor:** hello_world_main.c file open, showing code for app_main(void) including printf("Hello world!\n"); and chip information printing.
- Bottom Toolbar:** Buttons for COM3, esp32c6, UART, [ESP-IDF QEMU], [OpenOCD Server] (which is highlighted with a red border), and other developer tools.

(/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 "hello_world_main.c" file with the following code:

```
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;
```
- Bottom Status Bar:** Shows "COM3", "esp32c6", "UART", "[ESP-IDF QEMU]", "[OpenOCD Server]", "Ln 44, Col 36", "Spaces: 4", "UTF-8", "CRLF", "C", "ESP-IDF", and a refresh icon.

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

- Successfully opened as follows:

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

- File Explorer:** Shows the project structure under "HELLO_WORLD". Files include ".devcontainer", ".vscode", "c_cpp_properties.json", "launch.json", "settings.json", "tasks.json", "build", "main", "CMakeLists.txt", and "hello_world_main.c".
- Code Editor:** Displays the content of "hello_world_main.c".

```
main > C 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 }
```
- Terminal:** Shows logs from OpenOCD:

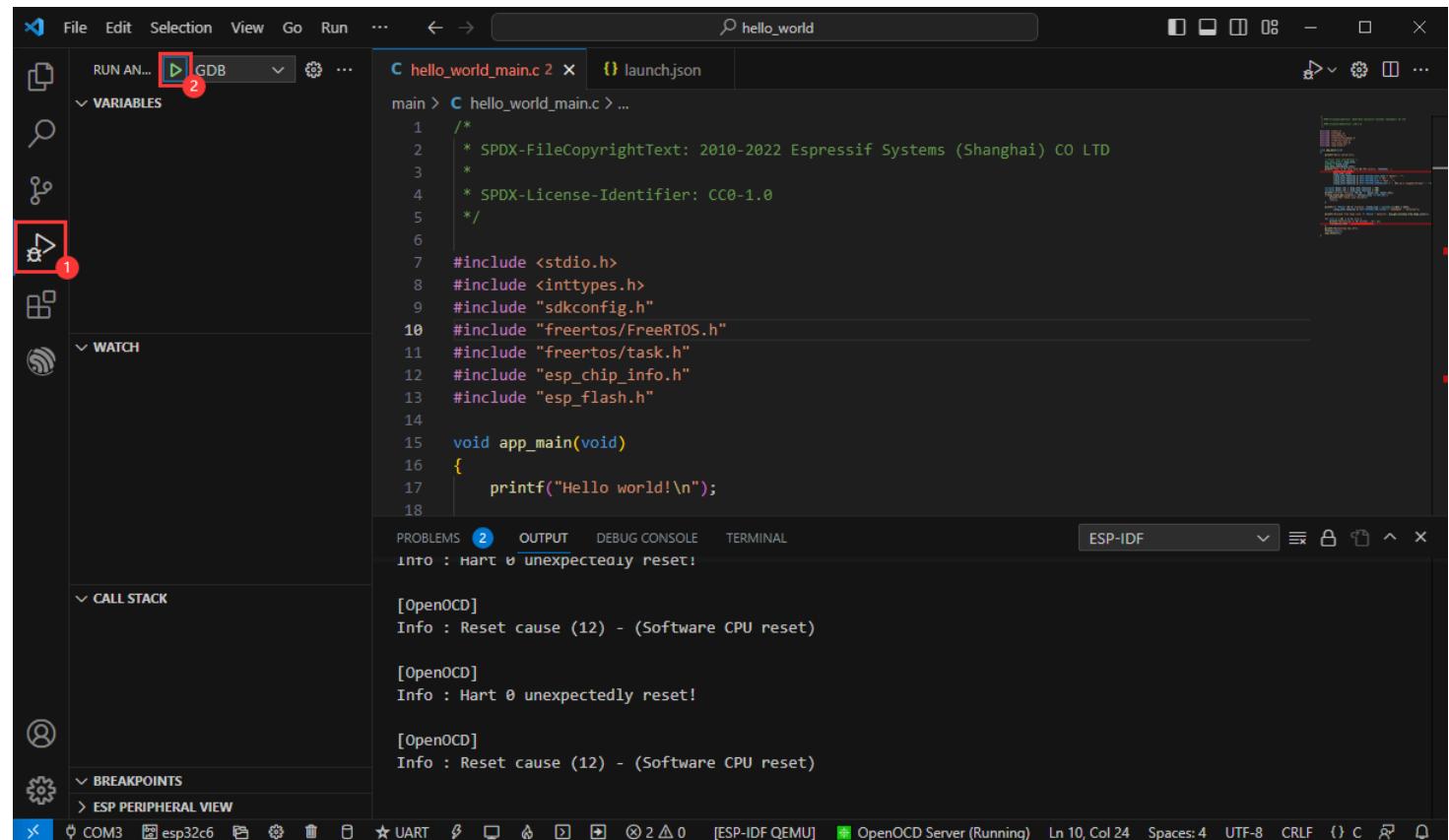
```
[OpenOCD]
Info : Reset cause (12) - (Software CPU reset)

[OpenOCD]
Info : Hart 0 unexpectedly reset!

[OpenOCD]
Info : Reset cause (12) - (Software CPU reset)
```
- Status Bar:** Shows "OpenOCD Server (Running)" with a green icon.

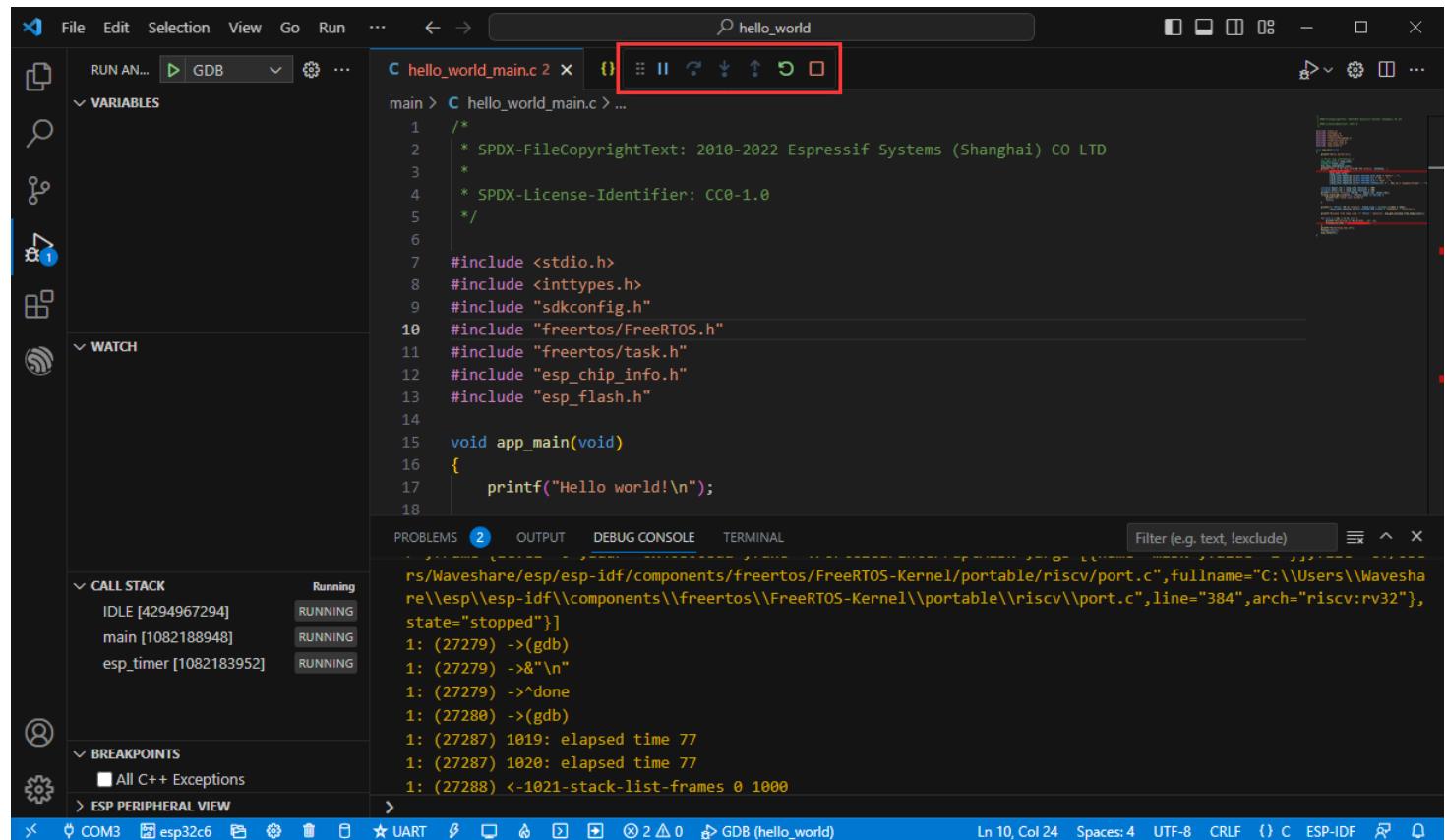
(/wiki/File:ESP32-C6-DEV-KIT-N8-80.png)

- Go to the debug function and click Debug:



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

- Successfully enter the debugging interface:

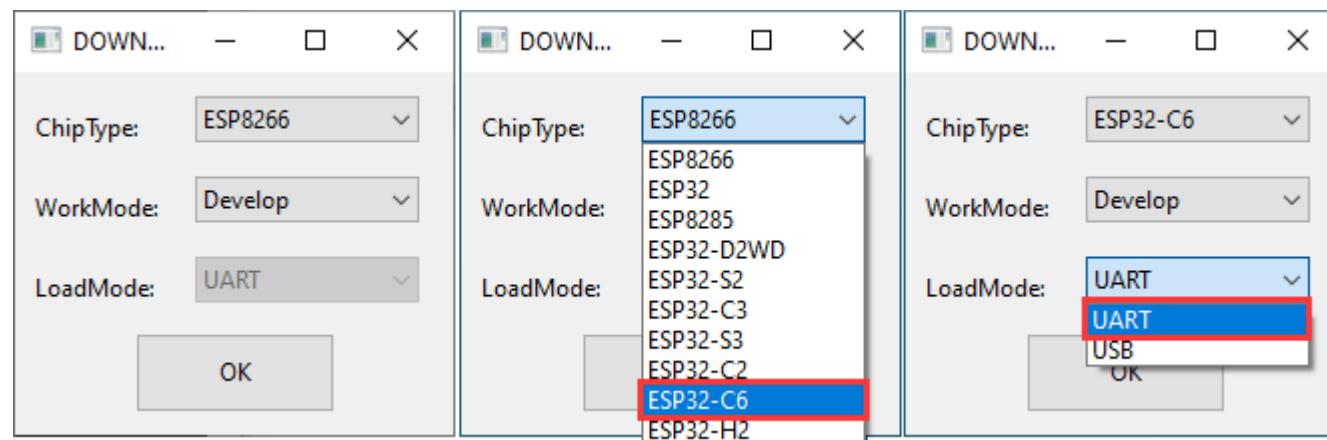


The screenshot shows the ESP-IDF IDE interface. The main window displays the source code for `hello_world_main.c`. The code includes standard headers and the `app_main` function which prints "Hello world!\n". The left sidebar contains toolbars for RUN AN..., VARIABLES, WATCH, CALL STACK, BREAKPOINTS, and ESP PERIPHERAL VIEW. The CALL STACK panel shows tasks: IDLE [4294967294] (RUNNING), main [1082188948] (RUNNING), and esp_timer [1082183952] (RUNNING). The BREAKPOINTS panel shows a single breakpoint at line 1. The bottom status bar shows various connection and terminal options, along with the current line (Ln 10, Col 24) and encoding (Spaces: 4, UTF-8, CRLF).

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

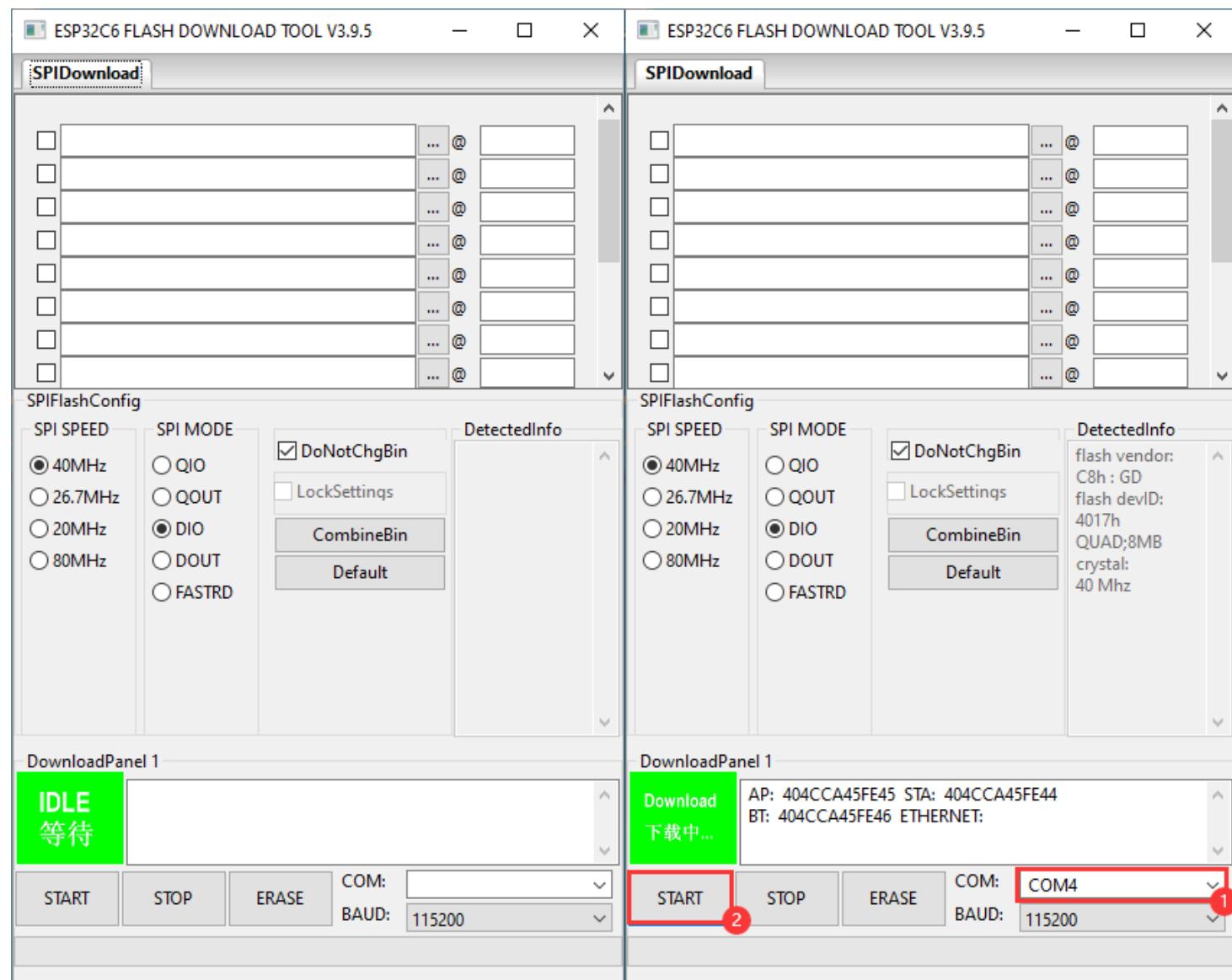
Erase Device Flash

- 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.



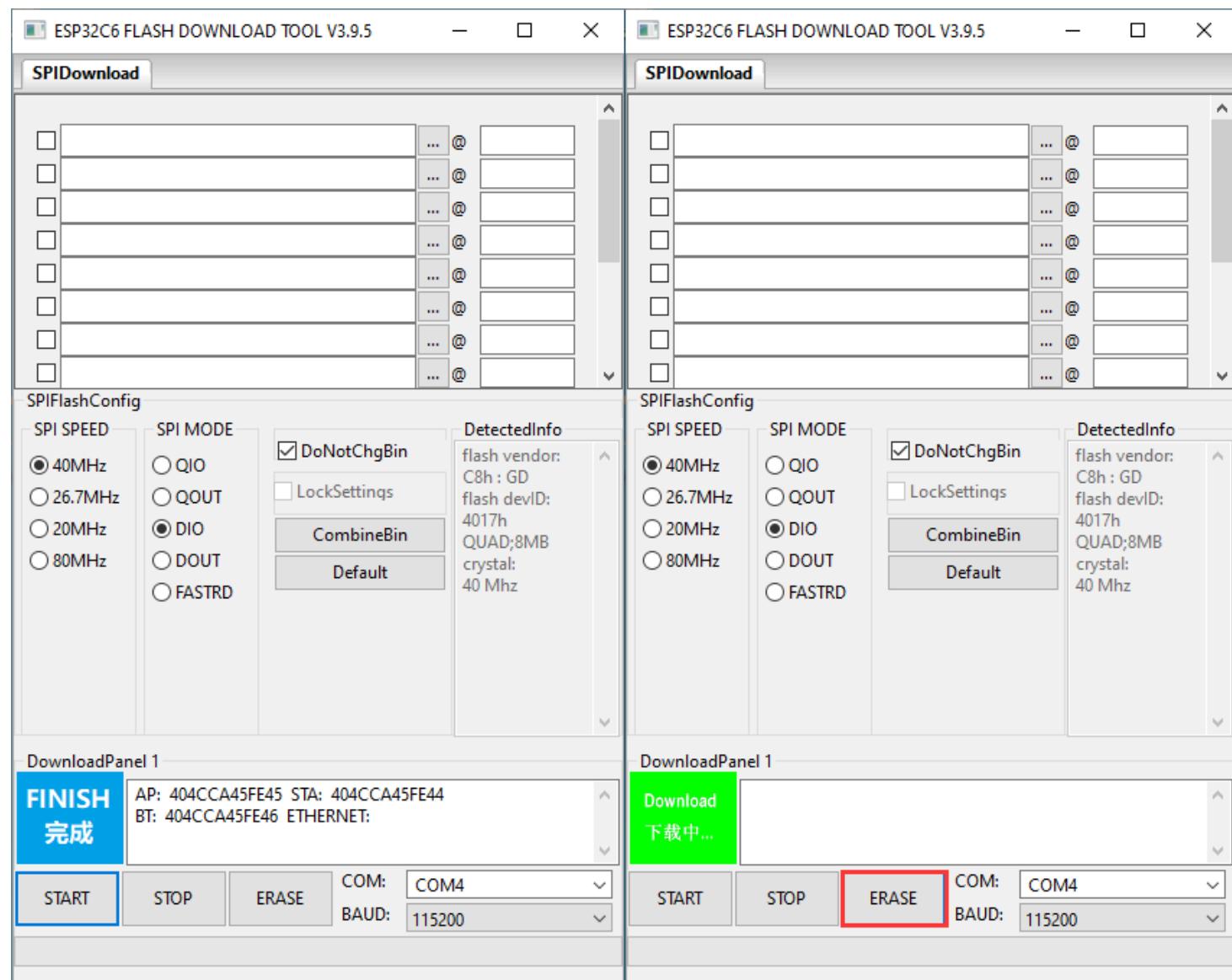
(/wiki/File:ESP32-C6-DEV-KIT-N8-83.png)

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



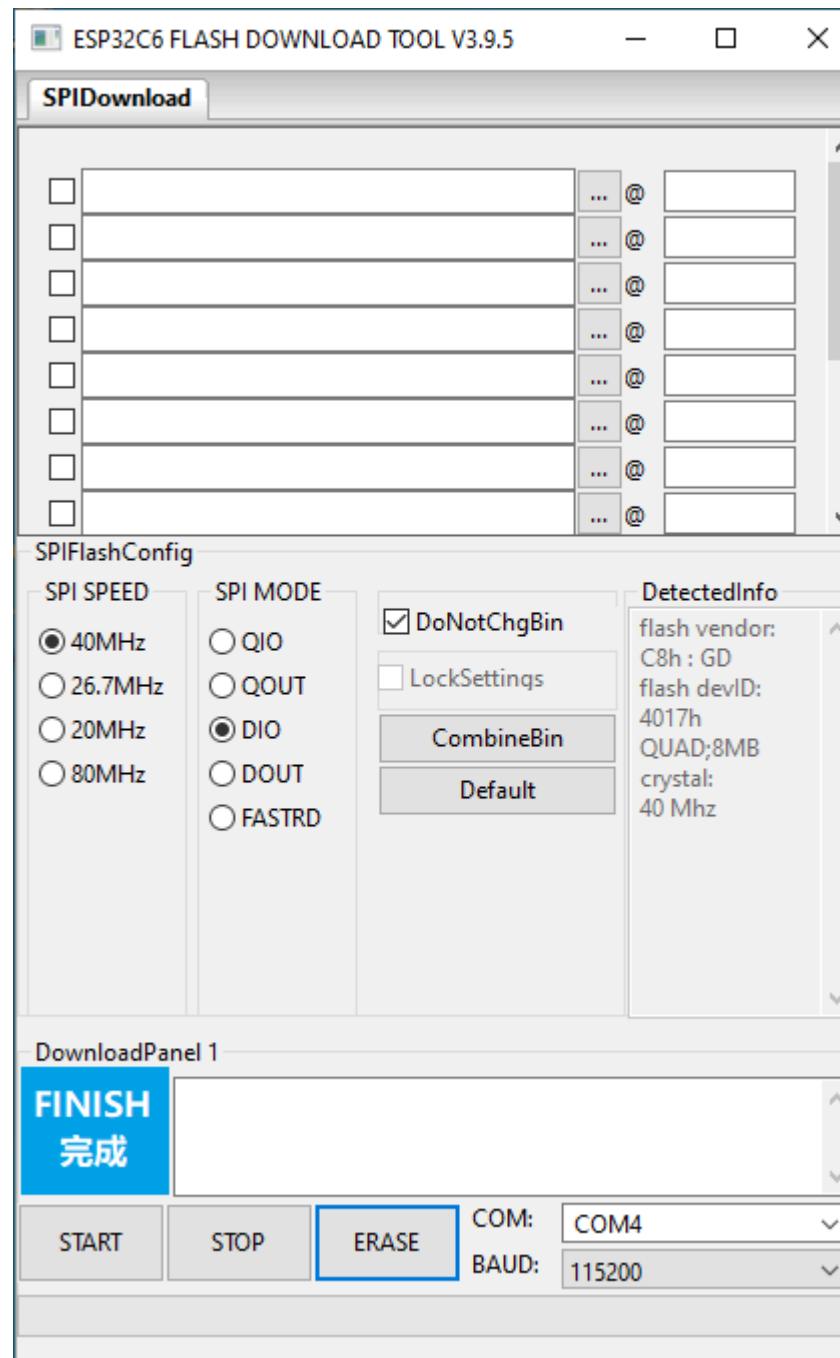
(/wiki/File:ESP32-C6-DEV-KIT-N8-84.png)

- After programming, click on "ERASE".



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

- Waiting for Erase to Finish.



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

86.png)

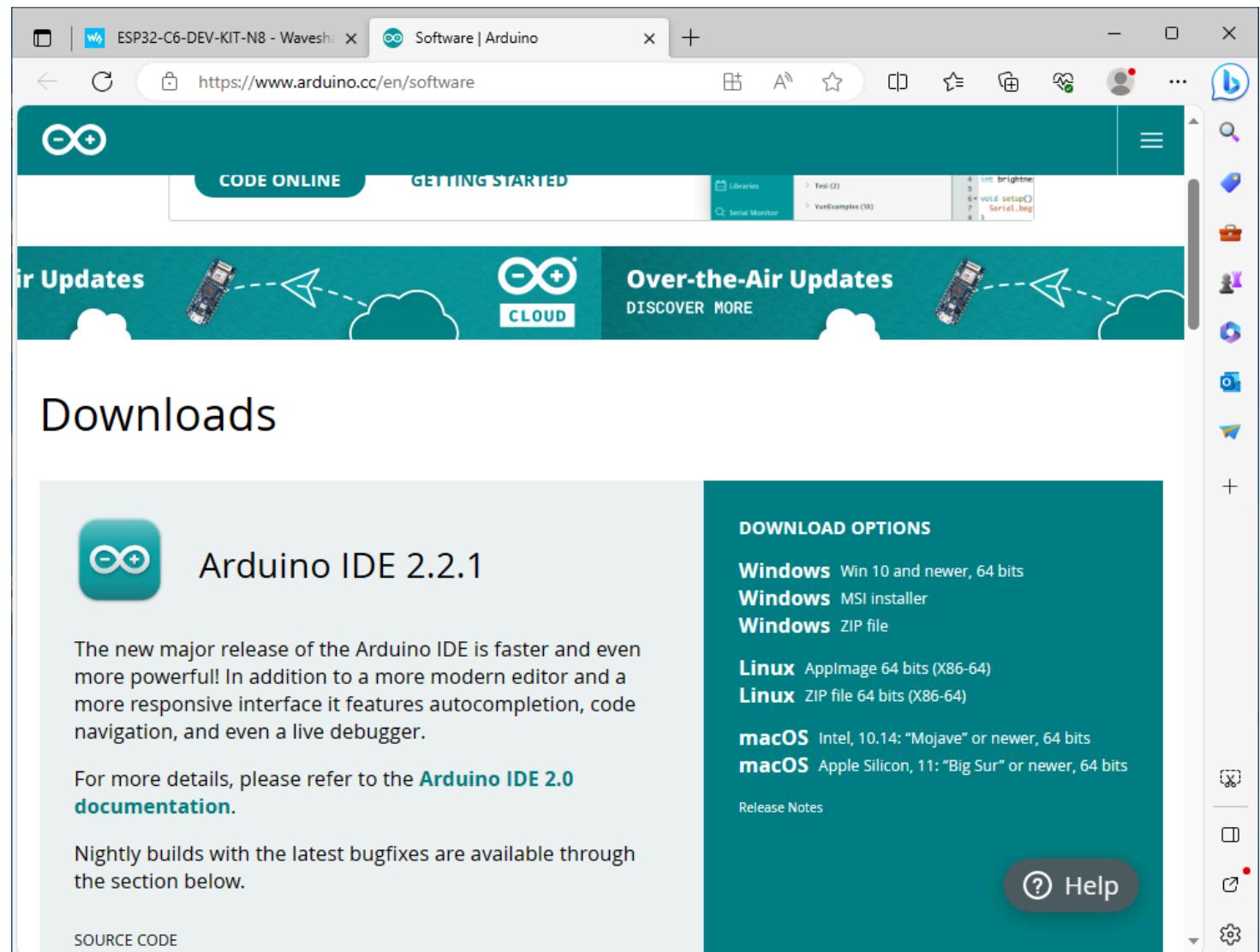
Working with Arduino

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.

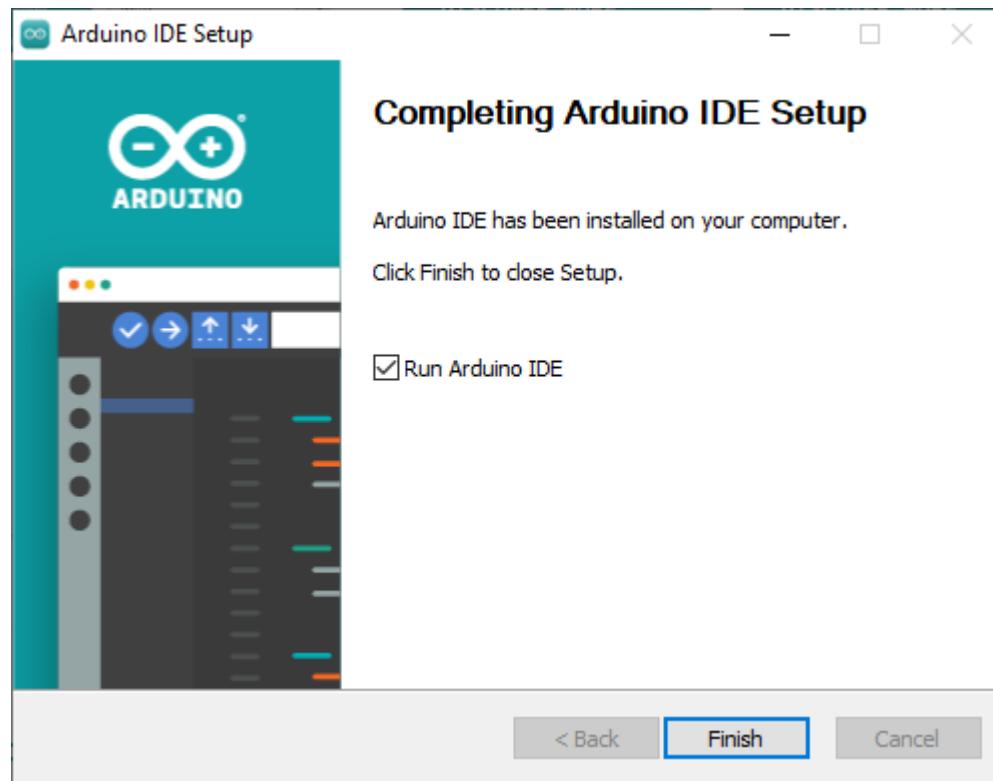
Environment Set-up

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



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

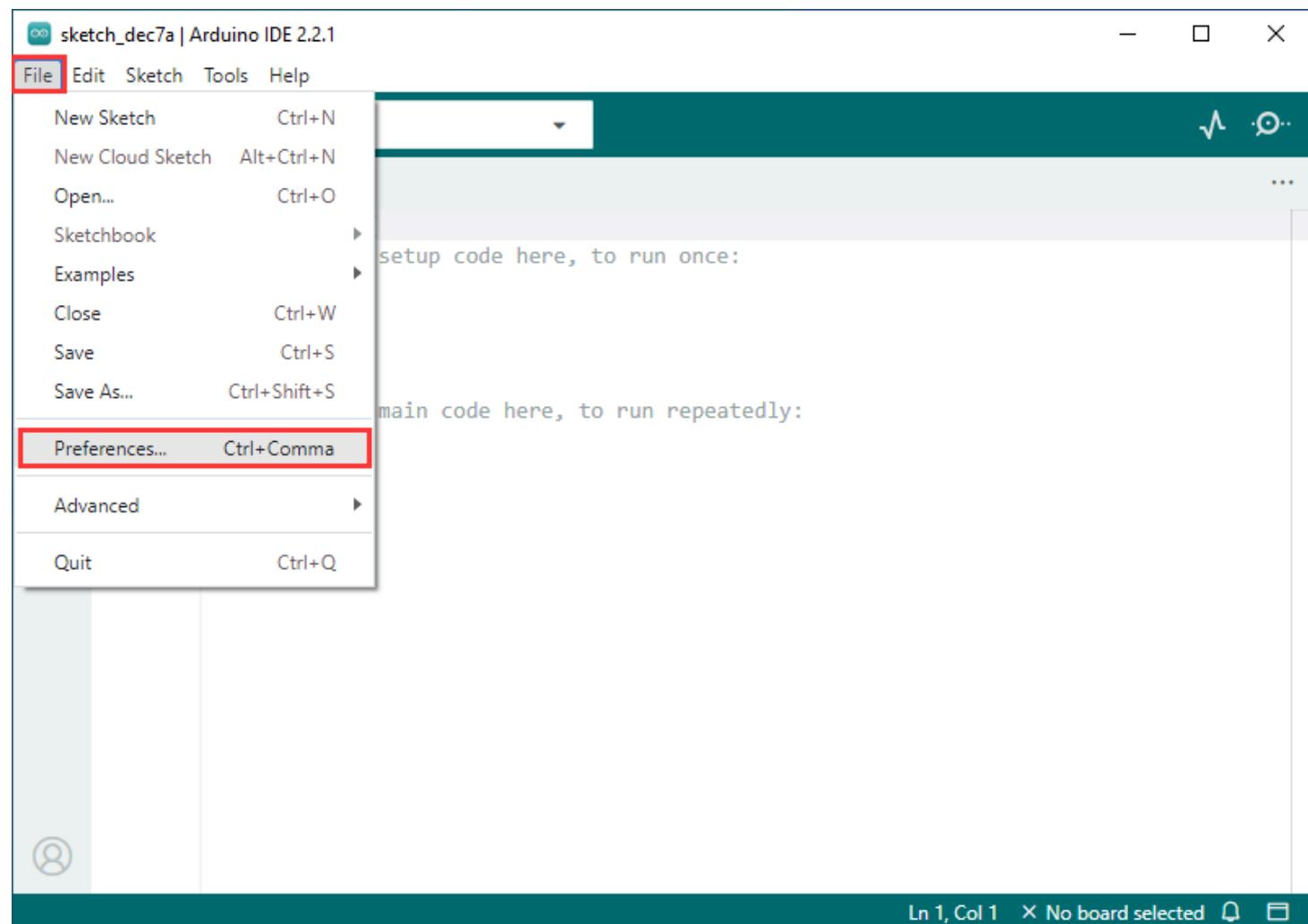
- Enter Arduino IDE after installation.



KIT-N8-Arduino02.png

- Enter Preferences.

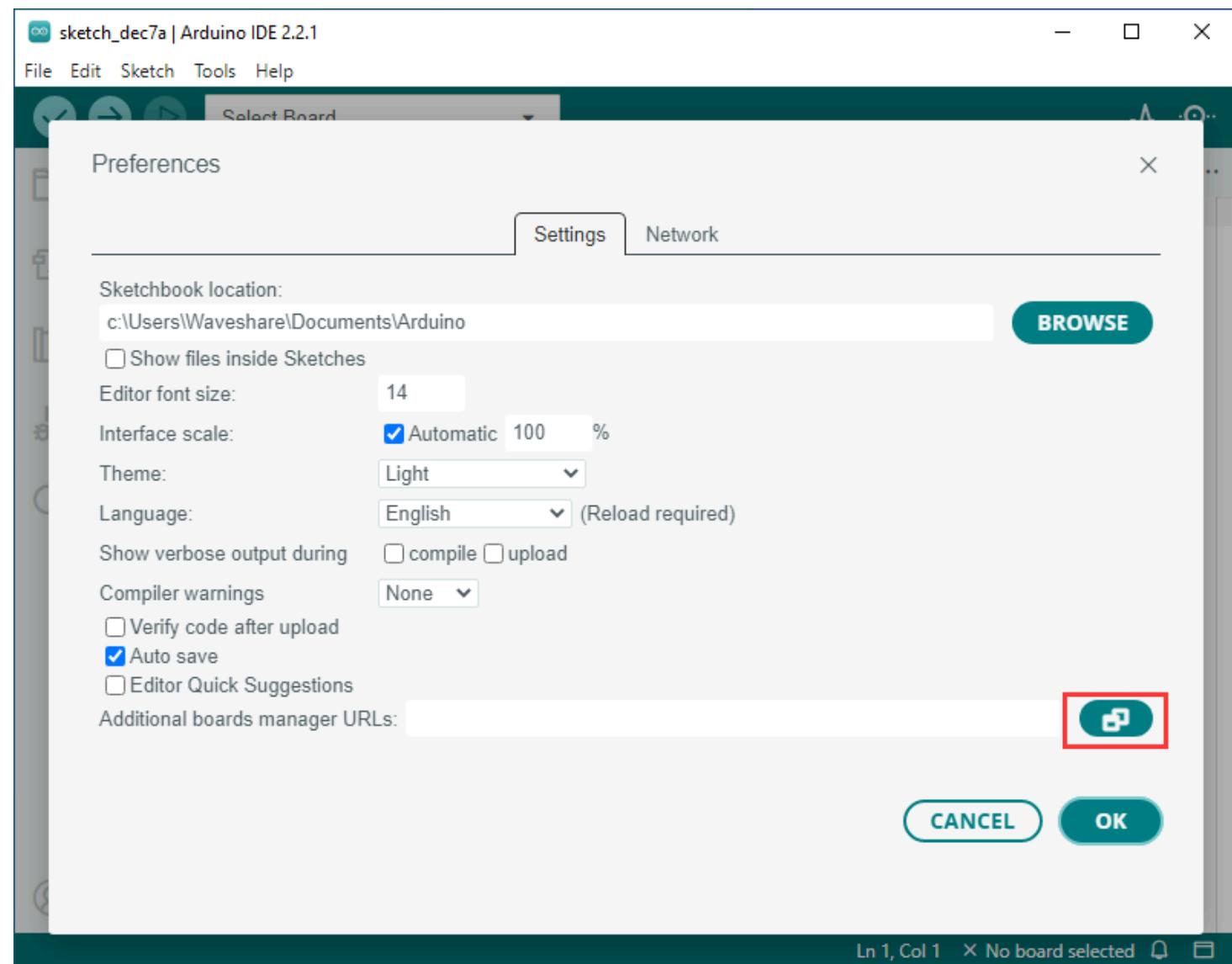
(/wiki/File:ESP32-C6-DEV-



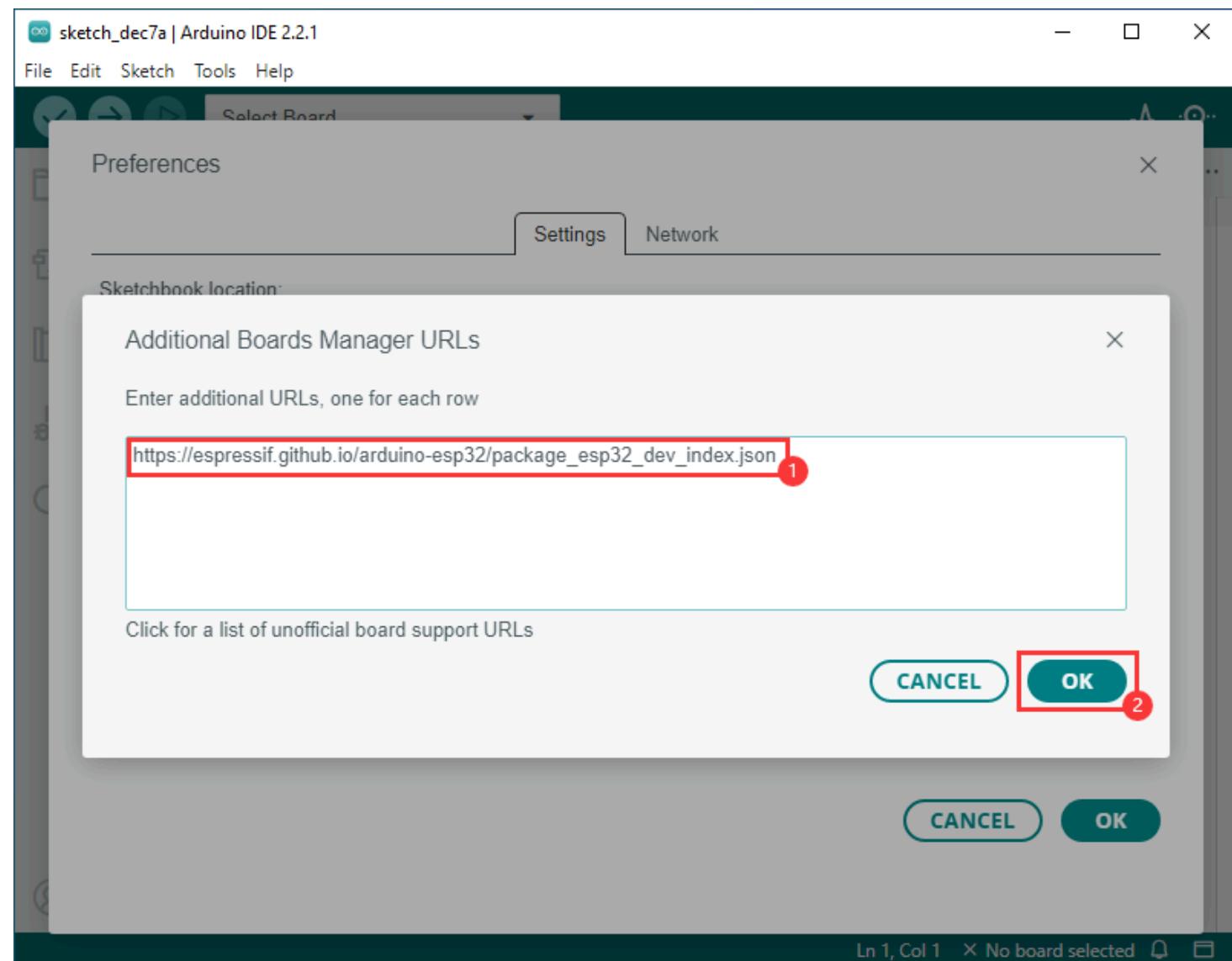
(/wiki/File:ESP32-C6-DEV-KIT-N8-Arduino03.png)

- 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)

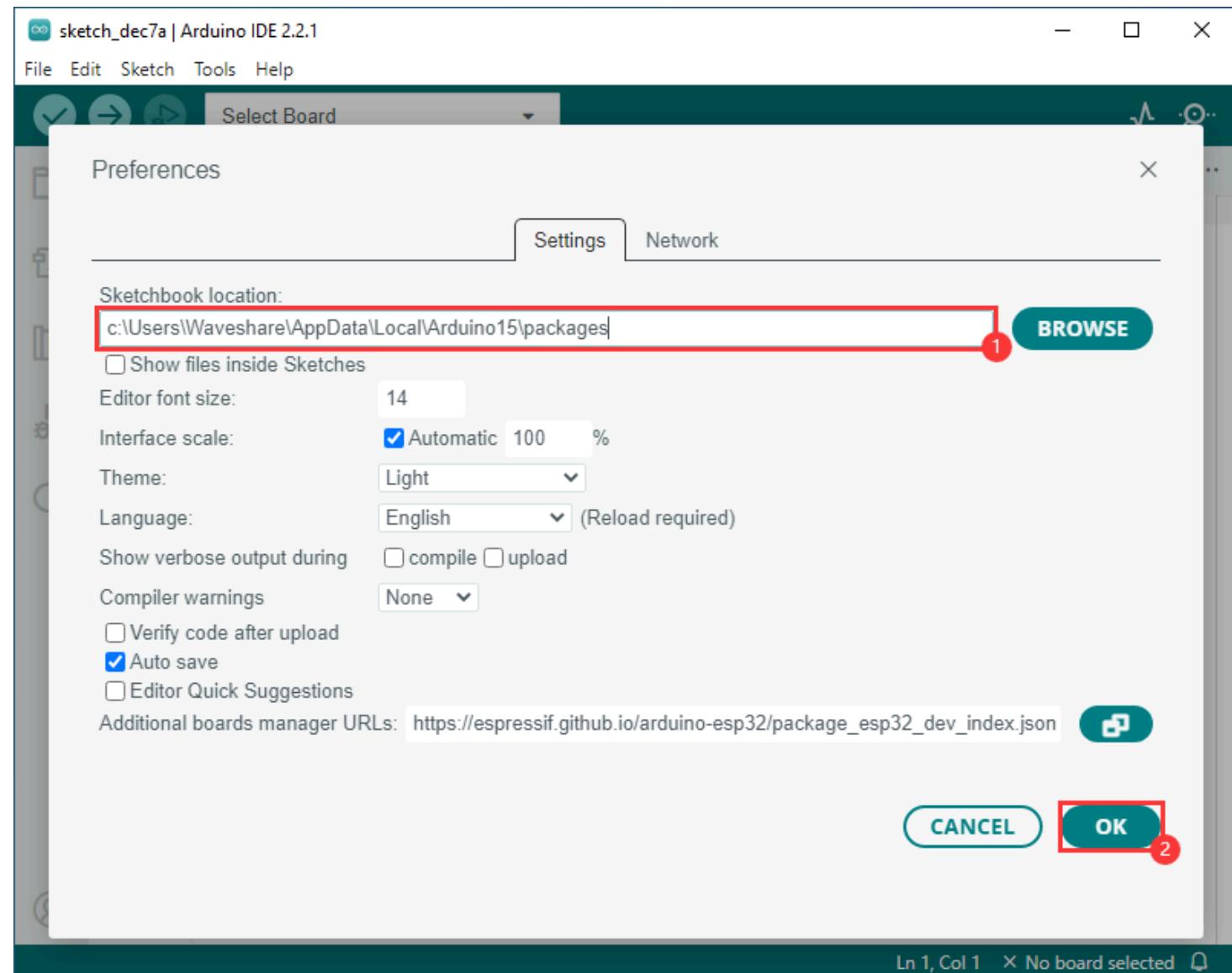


(/wiki/File:ESP32-C6-DEV-KIT-N8-Arduino04.png)



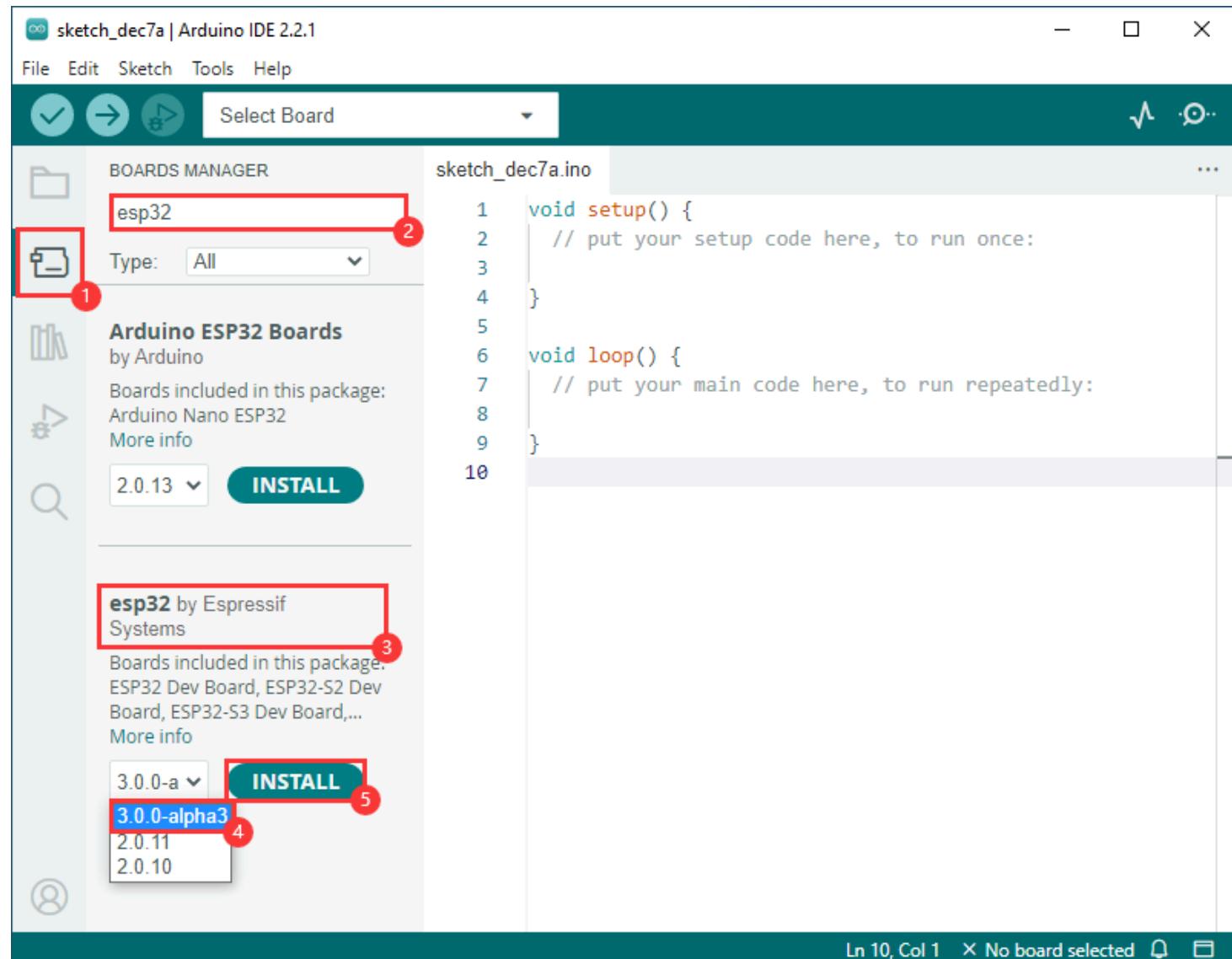
(/wiki/File:ESP32-C6-DEV-KIT-N8-Arduino05.png)

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

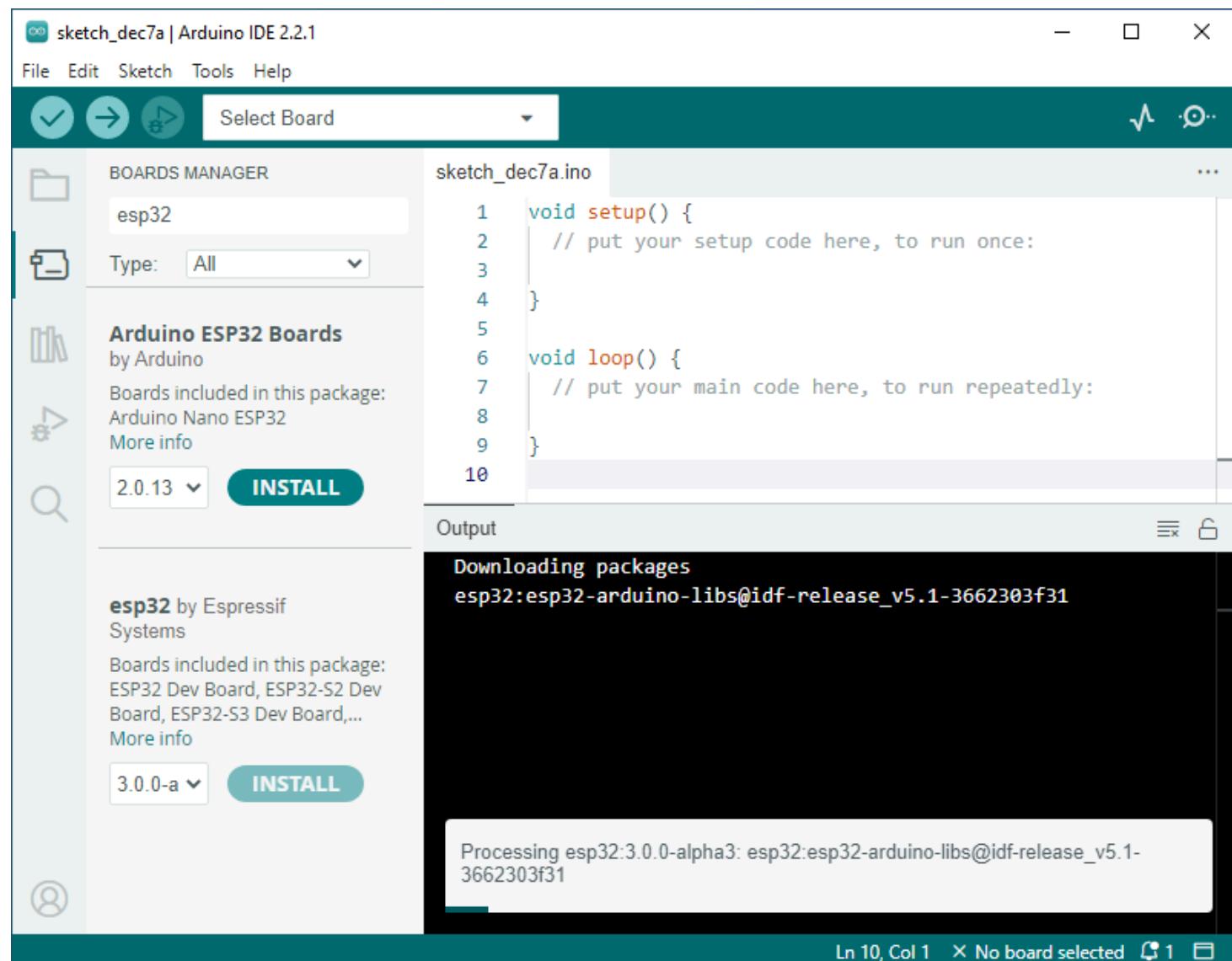


(/wiki/File:ESP32-C6-DEV-KIT-N8-Arduino06.png)

- 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.)

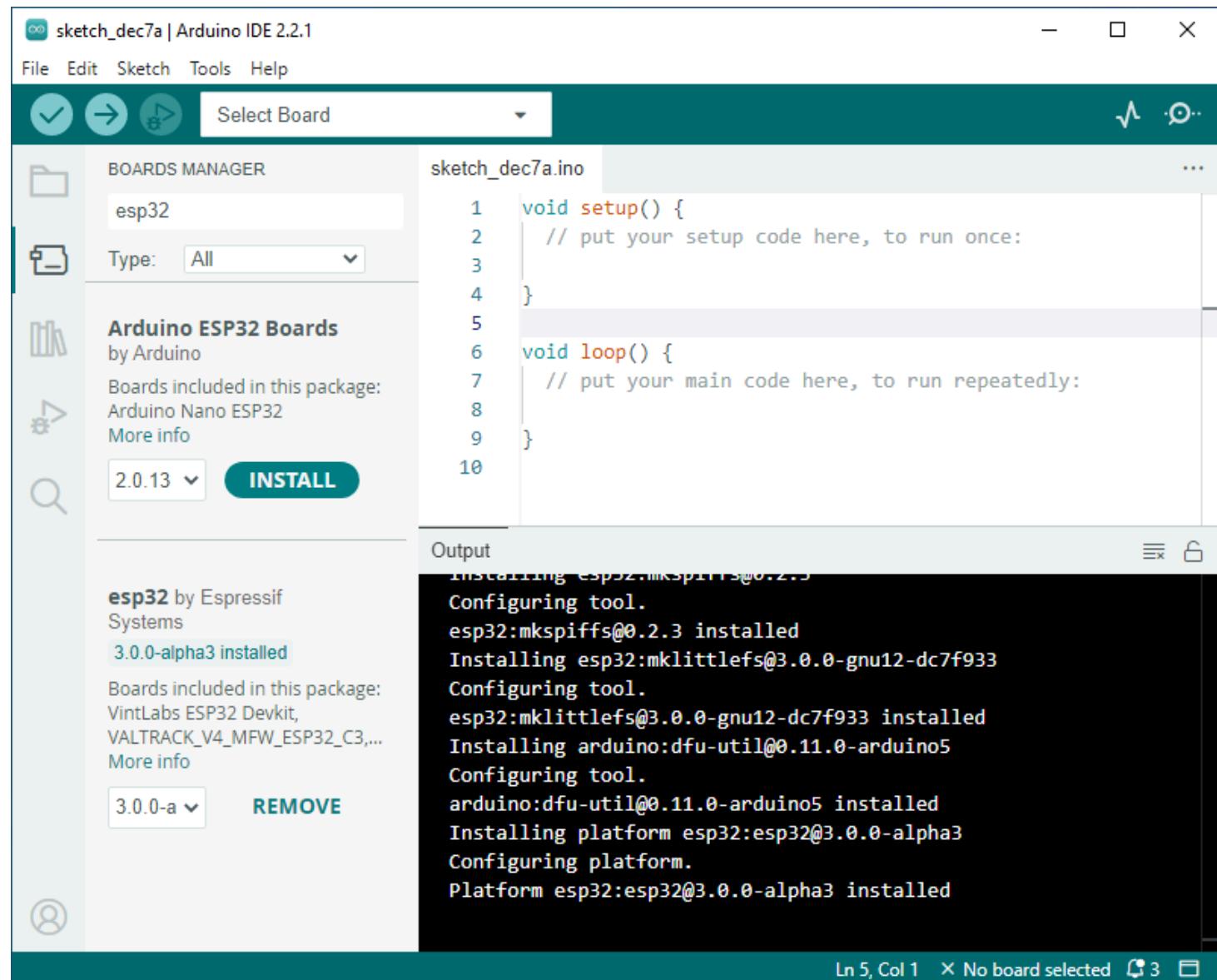


(/wiki/File:ESP32-C6-DEV-KIT-N8-Arduino07.png)



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

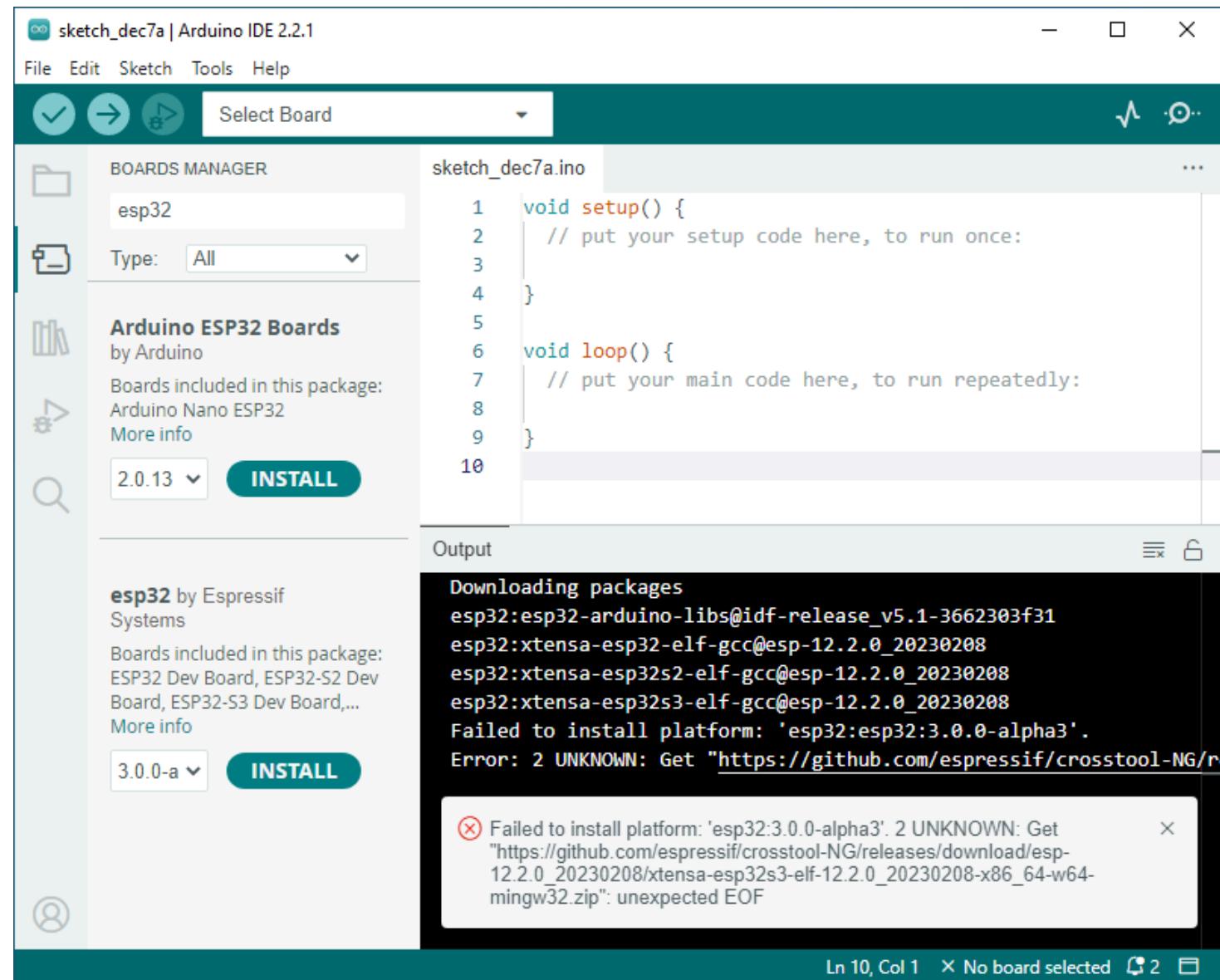
- Restart the Arduino IDE after installation, and then you can use it now.



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

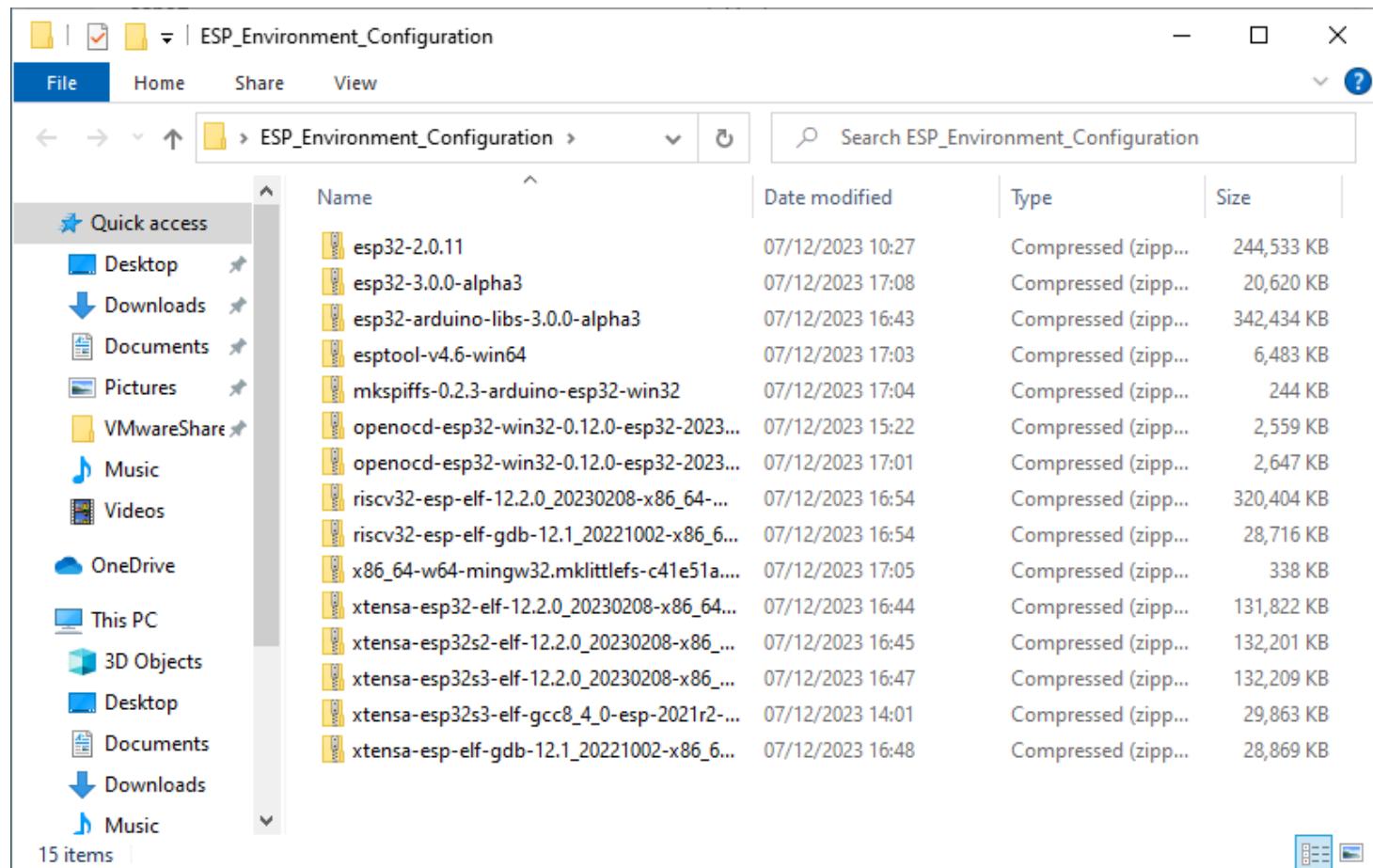
If the Installation Fails

- Failed to install version 3.0.0-alpha3:



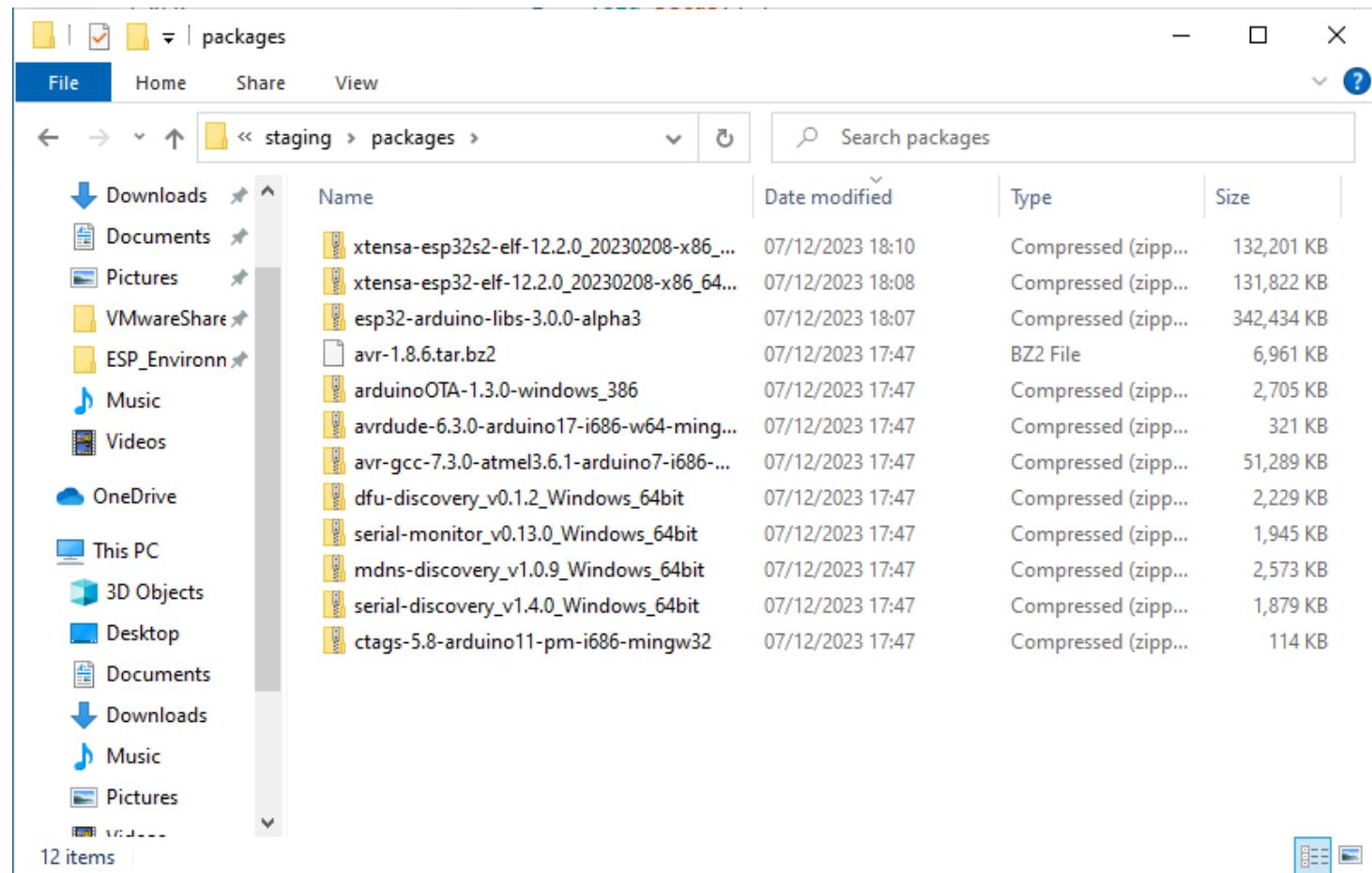
(/wiki/File:ESP32-C6-DEV-KIT-N8-install.png)

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



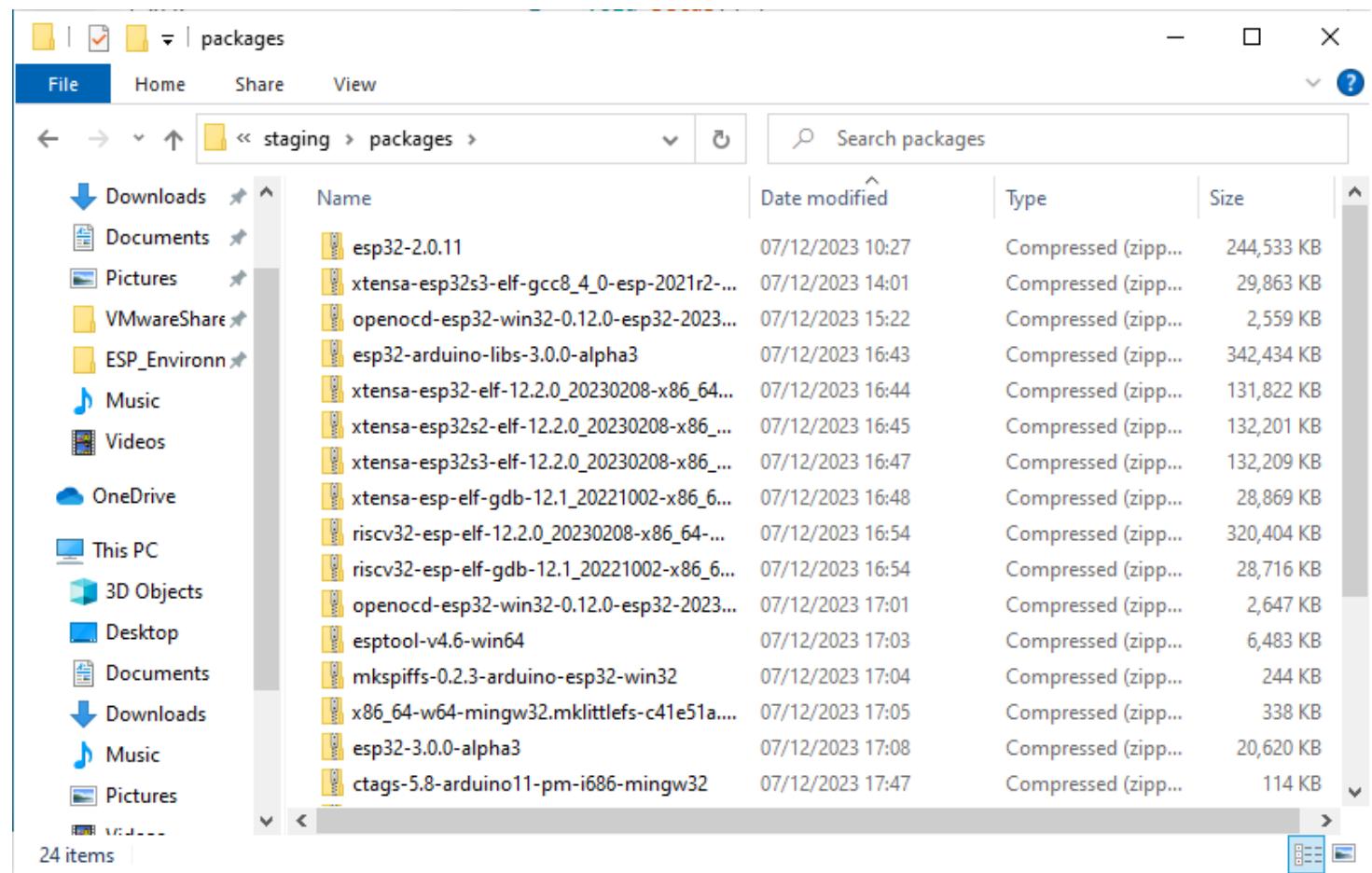
(/wiki/File:ESP32-C6-DEV-KIT-N8-install02.png)

- 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).



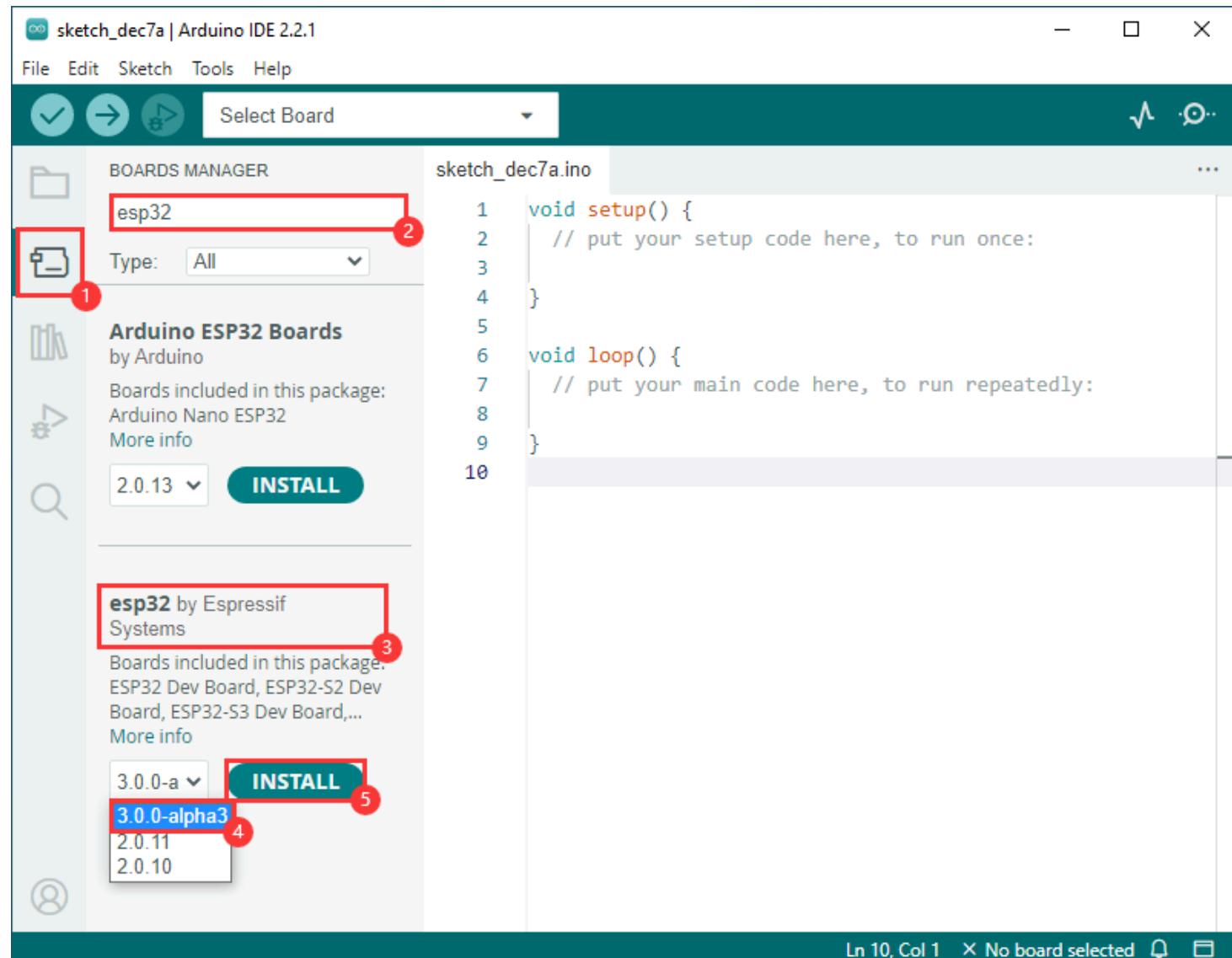
(/wiki/File:ESP32-C6-DEV-KIT-N8-install03.png)

- Unzip the downloaded files to the packages folder.



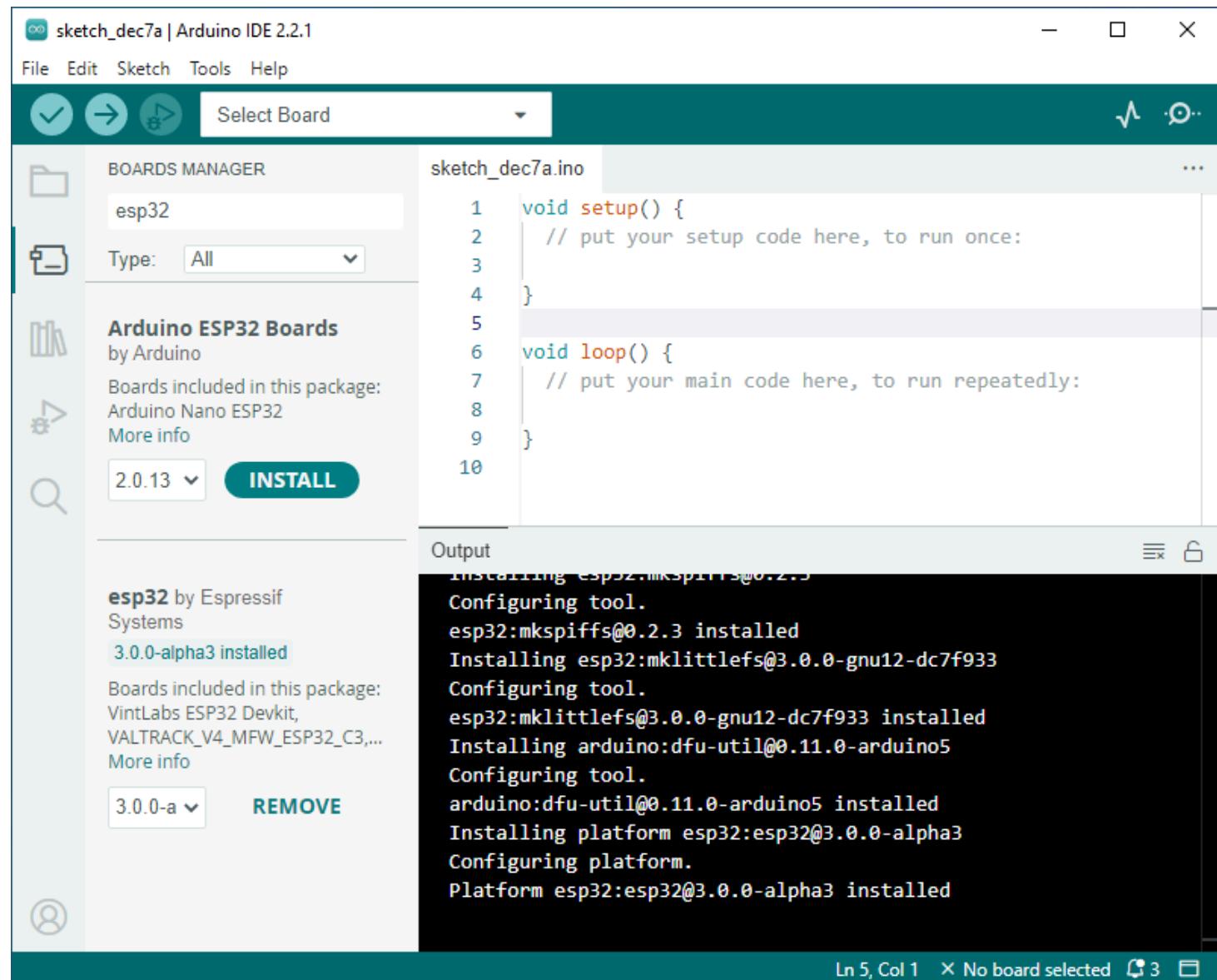
(/wiki/File:ESP32-C6-DEV-KIT-N8-install04.png)

- Install it again.



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

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

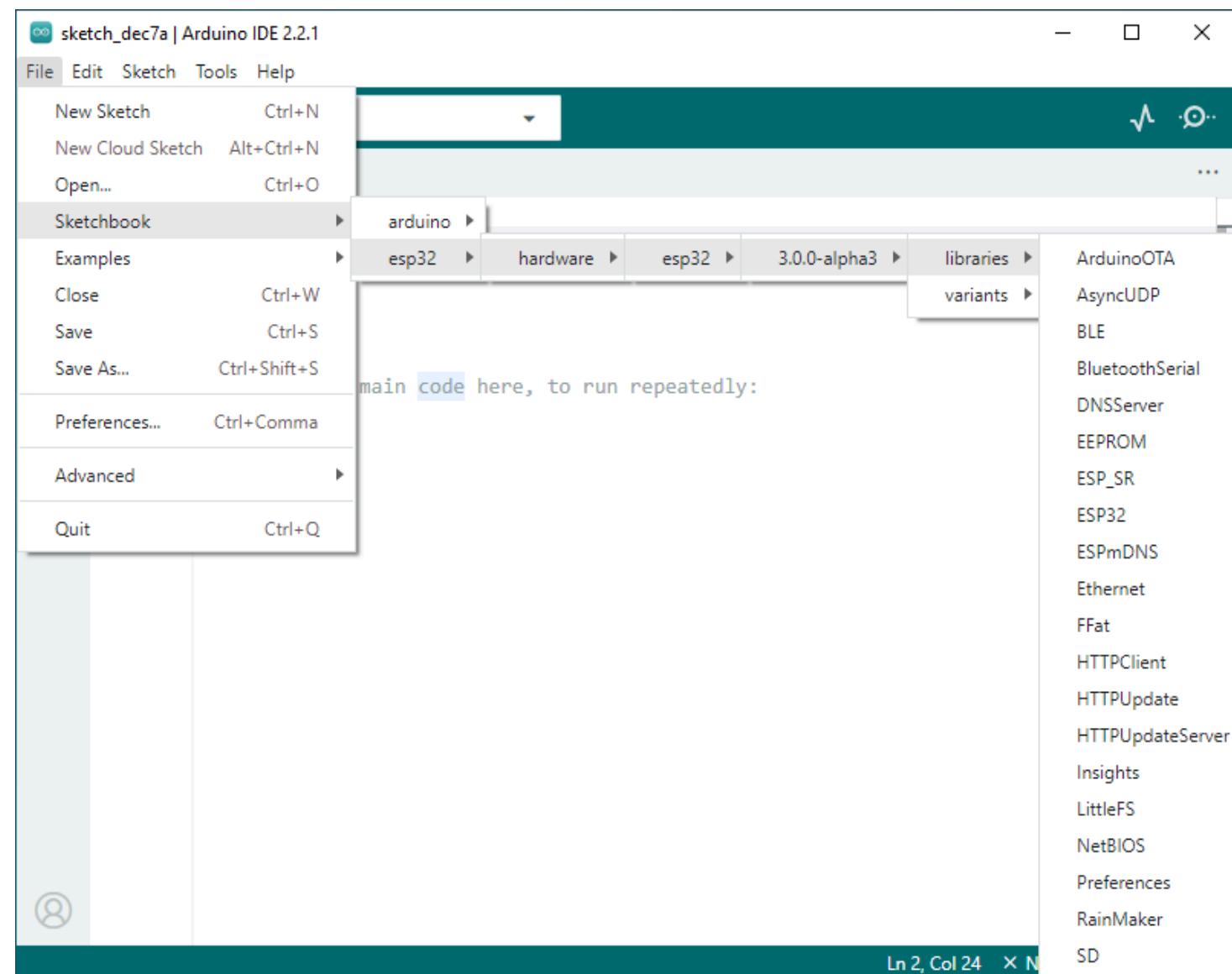


(/wiki/File:ESP32-C6-DEV-KIT-N8-Arduino09.png)

Create Example

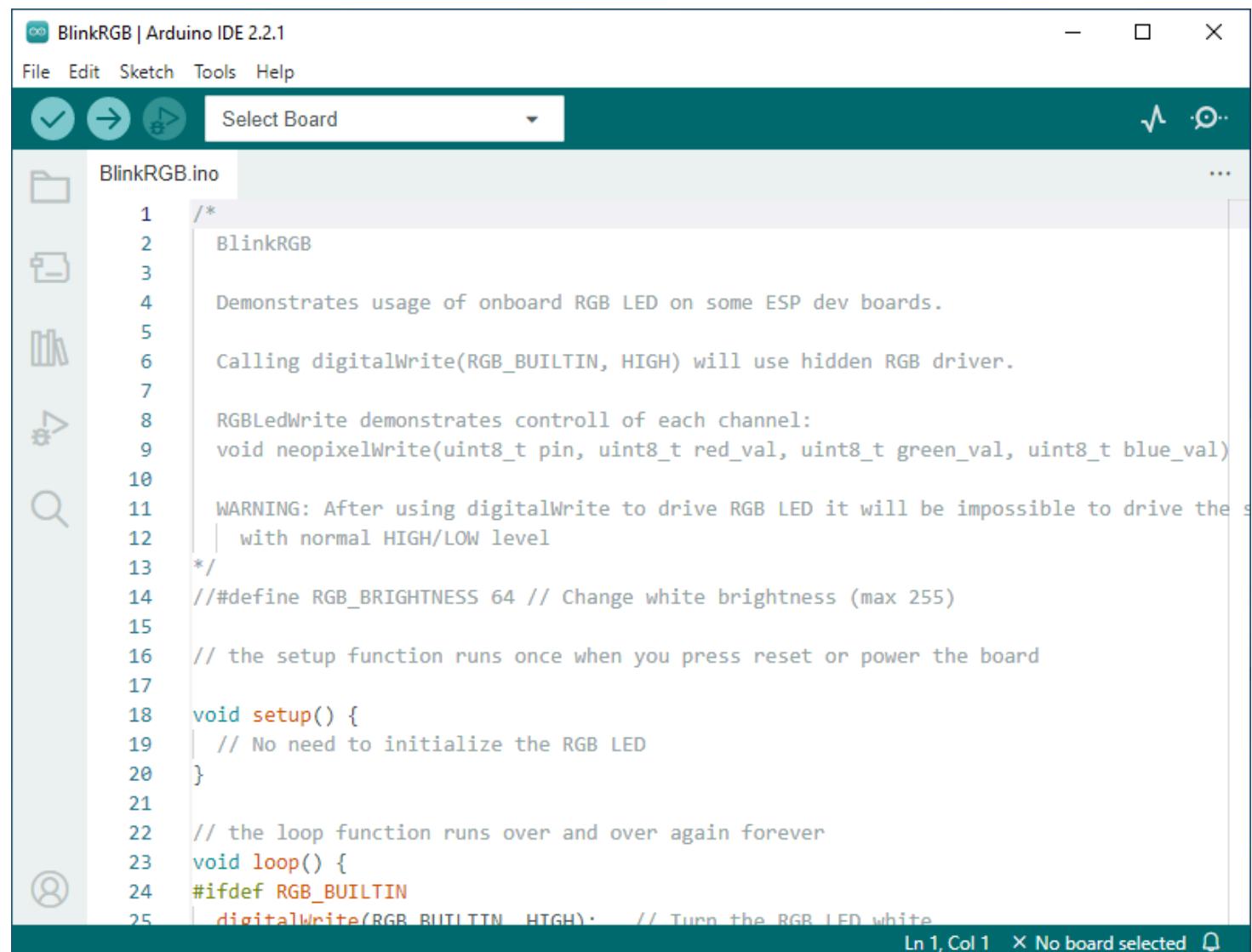
- After changing the project folder above to

c:\Users\Waveshare\AppData\Local\Arduino15\packages, you can create demos using the examples in the project folder under the File.



(/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).



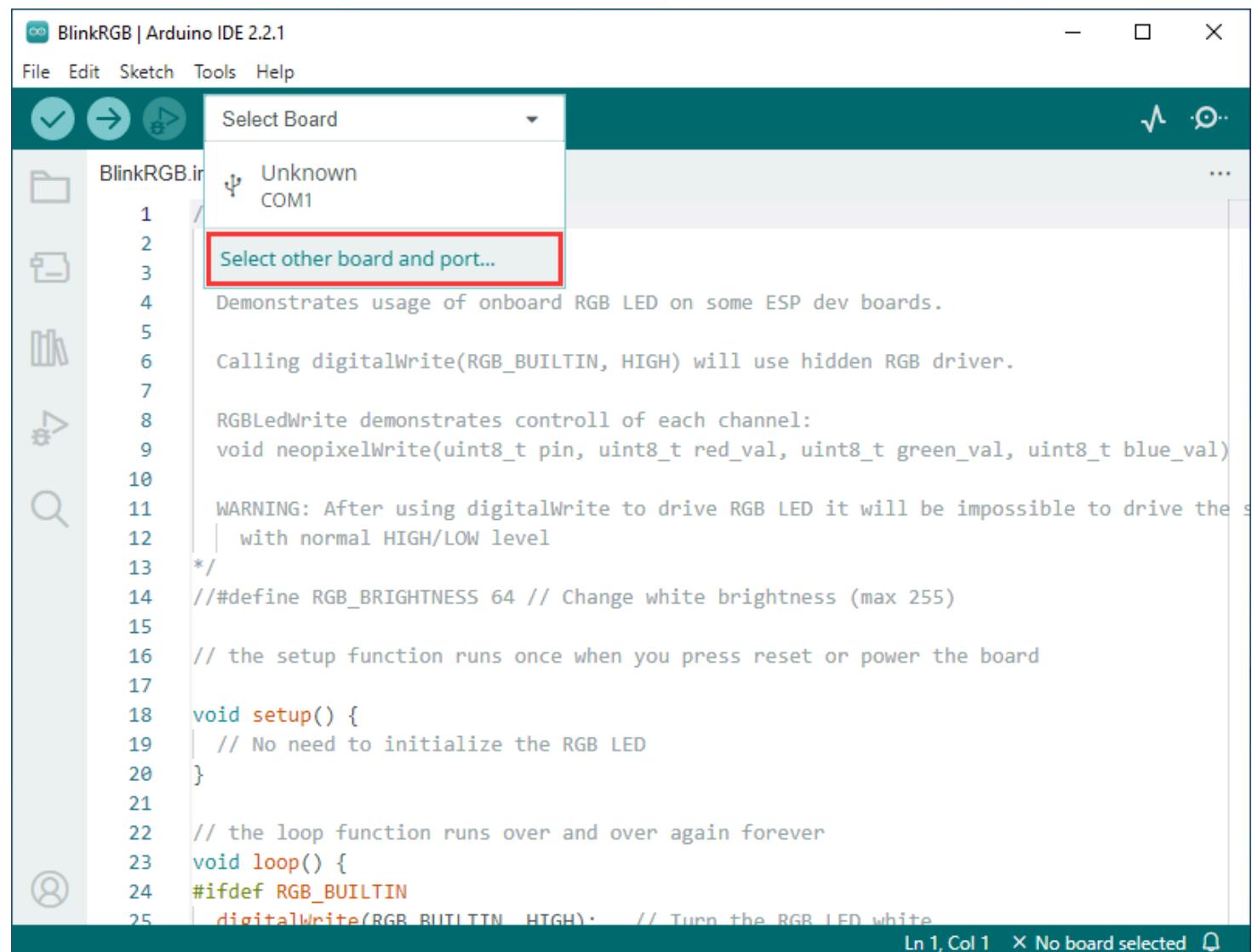
The screenshot shows the Arduino IDE 2.2.1 interface with the title bar "BlinkRGB | Arduino IDE 2.2.1". The menu bar includes File, Edit, Sketch, Tools, and Help. A toolbar with icons for file operations (checkmark, arrow, etc.) is above the main editor area. The central editor window displays the "BlinkRGB.ino" sketch. 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 }
```

The status bar at the bottom right shows "Ln 1, Col 1" and "No board selected".

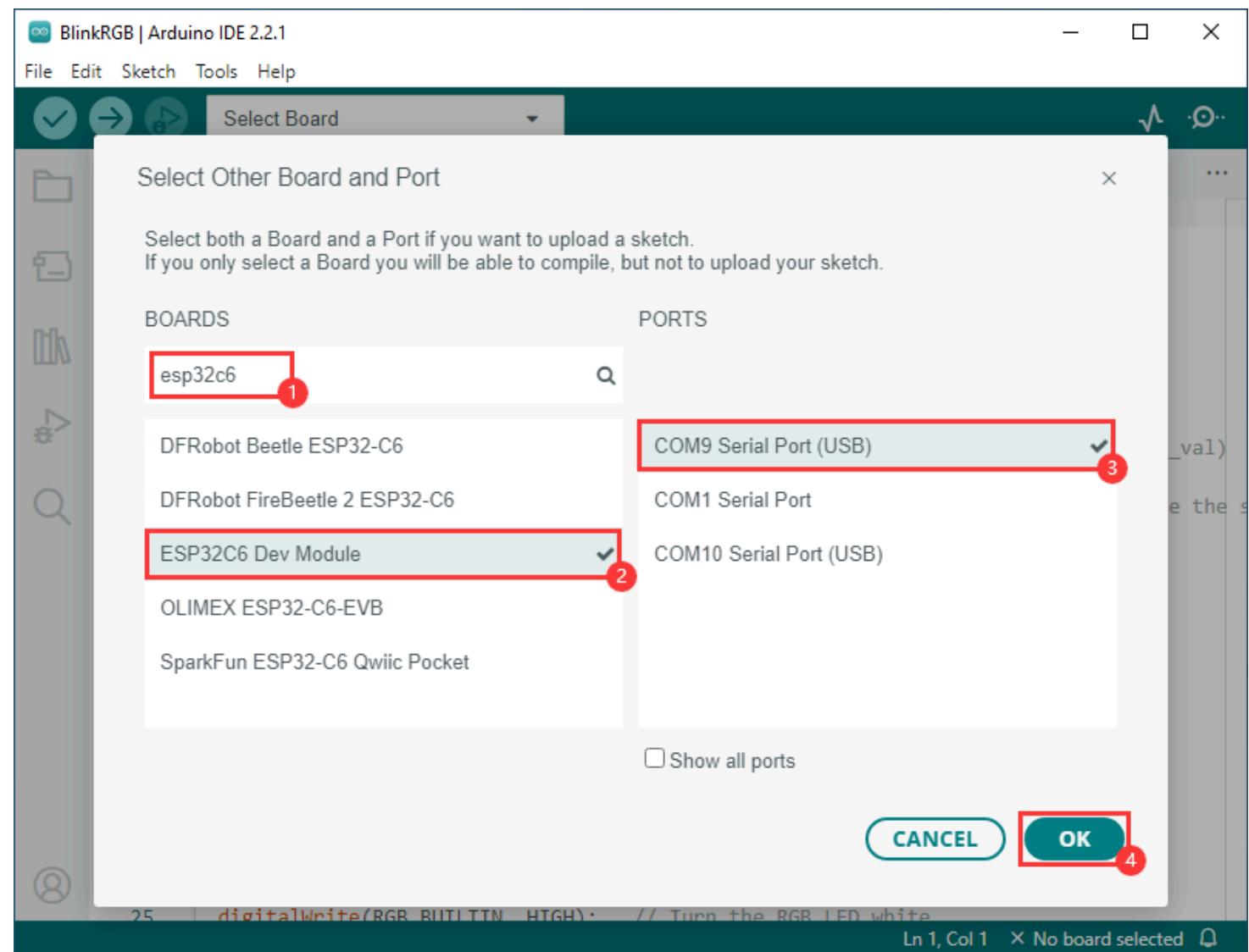
(/wiki/File:ESP32-C6-DEV-KIT-N8-Arduino11.png)

- Select the development board and port.



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

- Search for 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 "BlinkRGB | Arduino IDE 2.2.1". The menu bar includes File, Edit, Sketch, Tools, and Help. The toolbar features icons for checkmark, upload (highlighted with a red box), and refresh. The board selector dropdown shows "ESP32C6 Dev Module". The code editor displays the "BlinkRGB.ino" sketch:

```
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 }
```

The status bar at the bottom right indicates "Ln 1, Col 1 ESP32C6 Dev Module on COM9".

(/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
- Menu Bar:** File Edit Sketch Tools Help
- Toolbar:** Checkmark, Upload, Refresh, ESP32C6 Dev Module dropdown, Save, Undo, Redo.
- Code Editor:** Shows the `BlinkRGB.ino` file content. The code demonstrates the usage of the onboard RGB LED on some ESP dev boards, including the ability to control each channel separately using `neopixelWrite`. A warning is present about using `digitalWrite` on the RGB pins after the RGB driver has been initialized.
- Output Window:** Displays 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

A message box in the bottom right corner indicates "Done uploading."

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

Resource

Software

Compile

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

CH343

- Window VCP driver (<https://files.waveshare.com/upload/f/f1/CH343SER.7z>)
- Android APP (https://files.waveshare.com/upload/2/22/WCHUARTDemo_V1.3.7z)

- MAC driver (https://files.waveshare.com/upload/0/04/CH34XSER_MAC.7z)

UART

- SS COM5.13.1 ([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 Download Tool

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

Bluetooth Debug

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

Schematics

- Schematic (<https://files.waveshare.com/wiki/ESP32-C6-DEV-KIT-N8/ESP32-C6-DEV-KIT-N8-Schematic.pdf>)

Datasheet

ESP32-C6

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

- ESP32-C6-WROOM-1 Datasheet (https://files.waveshare.com/wiki/ESP32-C6-DEV-KIT-N8/ESP32-C6-WROOM-1_Datasheet.pdf)

CH343

- CH343 Datasheet (<https://www.waveshare.com/w/upload/a/a3/CH343DS1-en.PDF>)

Official Document

ESP32

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

FAQ

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 1 second, wait for the PC to recognize the device again, and then proceed with the download.

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:When the module uses UART(CH343) to download a demo, an unrecognized USB device pop-up appears?

Answer:

It has no effect on program flashing. If you want to use the corresponding COM port of USB after the pop-up of an unrecognized USB device, please disconnect the cable from the computer and reconnect the device. You can also erase the demo by software or instruction, after erasing the demo, the USB port will be recognized again.

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 issue.

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

Submit Now (<https://service.waveshare.com/>)

*Retrieved from "<https://www.waveshare.com/w/index.php?title=ESP32-C6-DEV-KIT-N8&oldid=104260>
<https://www.waveshare.com/w/index.php?title=ESP32-C6-DEV-KIT-N8&oldid=104260>"*
