ESP32IDF development environment setup

Installation link: [Standard Setup of Toolchain for Windows - ESP32-S3 - — ESP-IDF Programming Guide v5.3.2 documentation]

ESP-IDF requires the installation of some prerequisite tools to build firmware around ESP32-S3, including Python, Git, cross compiler, CMake and Ninja compilation tools.

1. Installation prerequisites

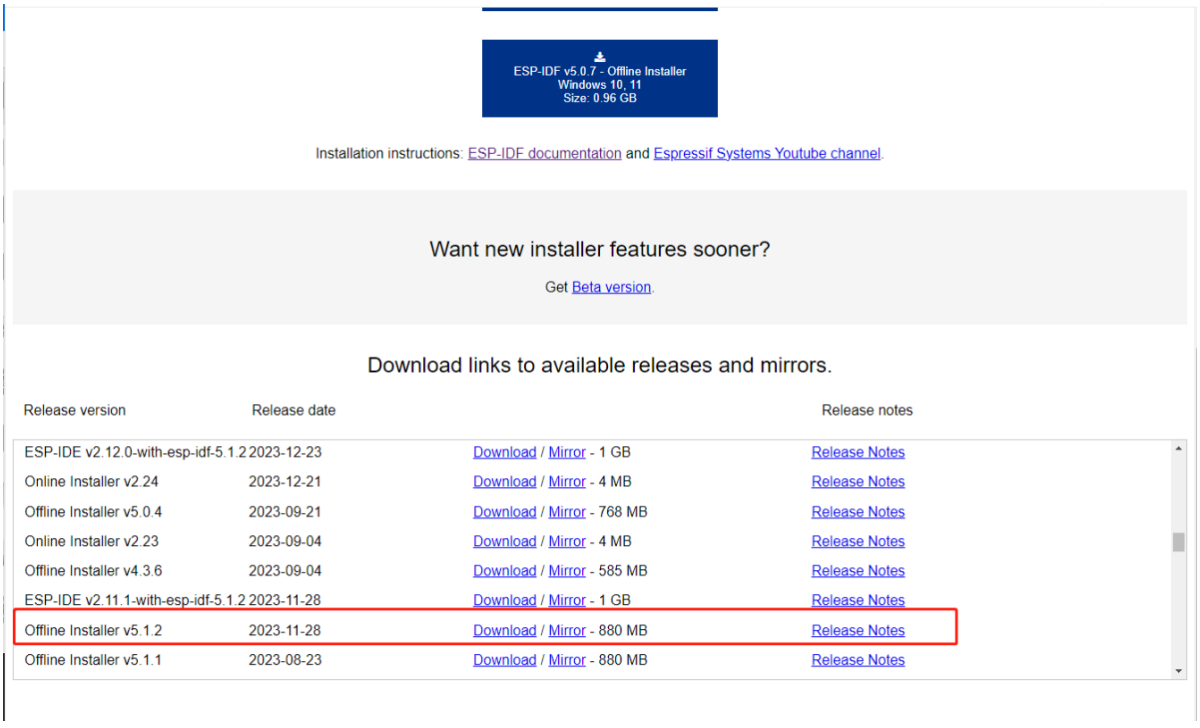
- Please note that the installation path of ESP-IDF and ESP-IDF tools cannot exceed 90 characters. Too long installation paths may cause build failures.
- The installation path of Python or ESP-IDF must not contain spaces or brackets.
- Unless the operating system is configured to support Unicode UTF-8, Python or ESP-IDF installation paths cannot include special characters (non-ASCII characters)

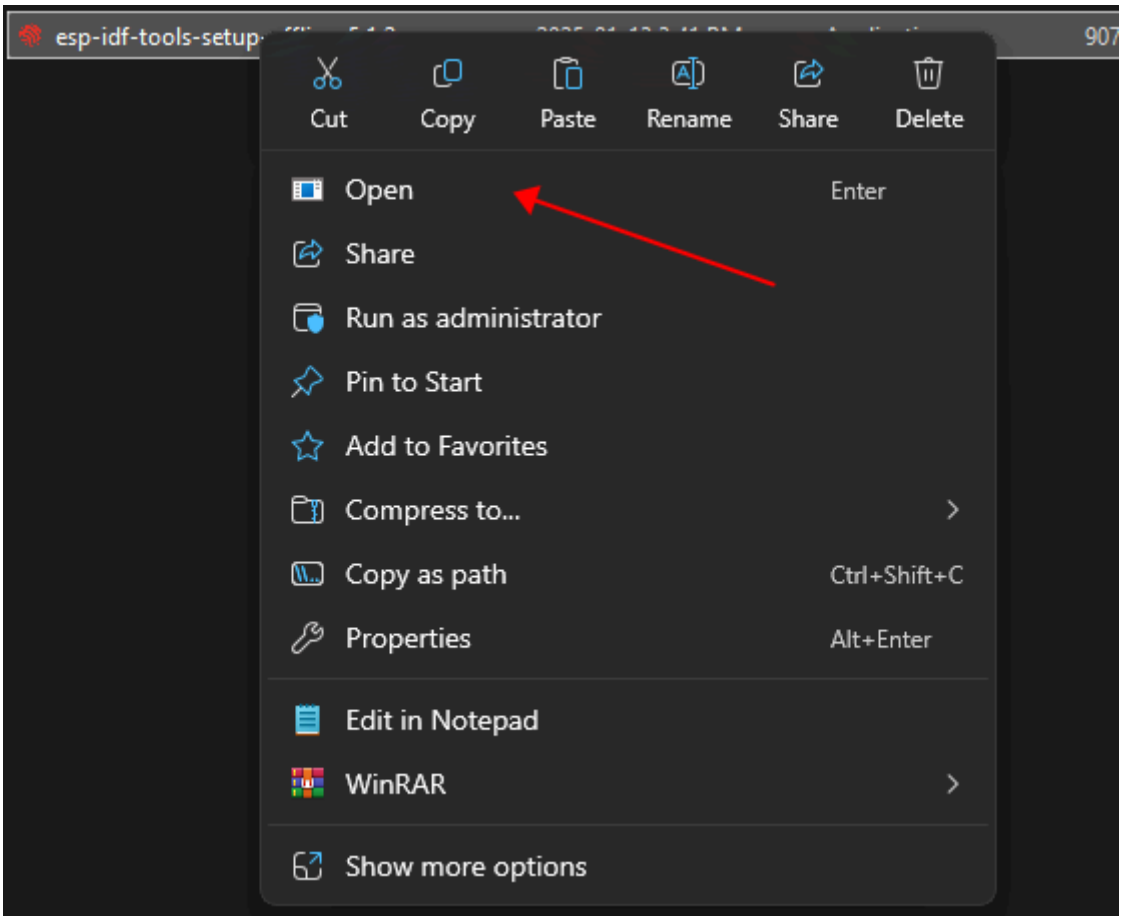
2. ESP-IDF Tool Installer

The easiest way to install the ESP-IDF essential tools is to download an ESP-IDF Tool Installer.

Online download link: dl.espressif.com.cn/dl/esp-idf/?idf=4.4

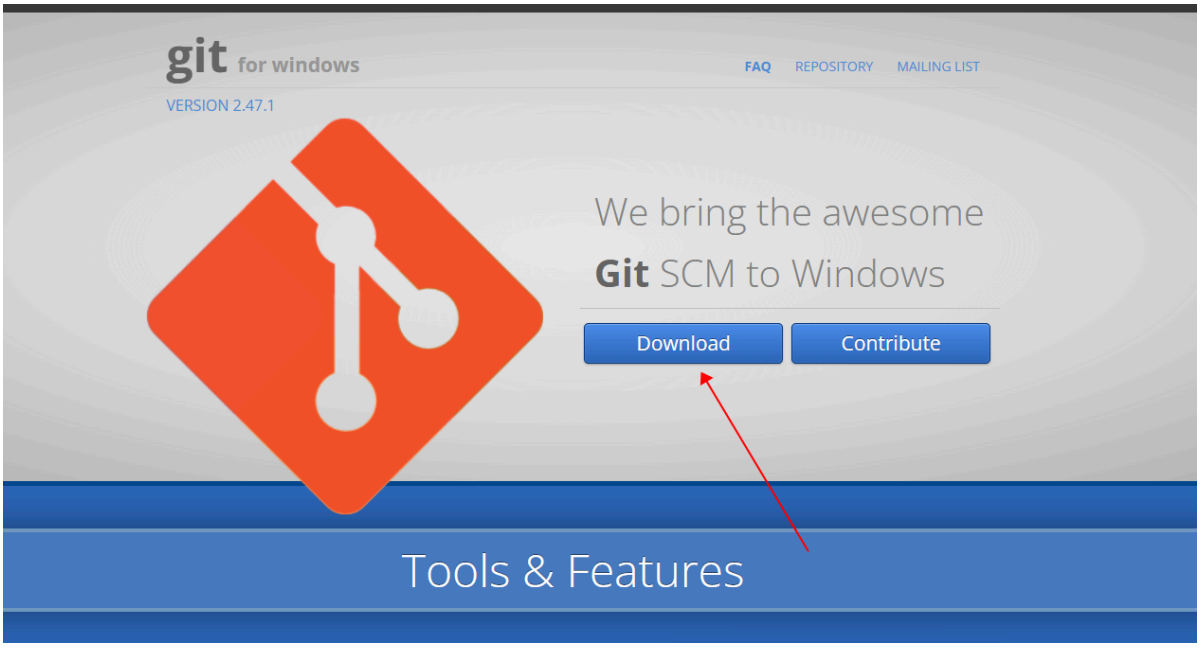
Drag to the 5.1.2 version below, or select other versions, but some functions may need to be replaced with the corresponding version of the function.



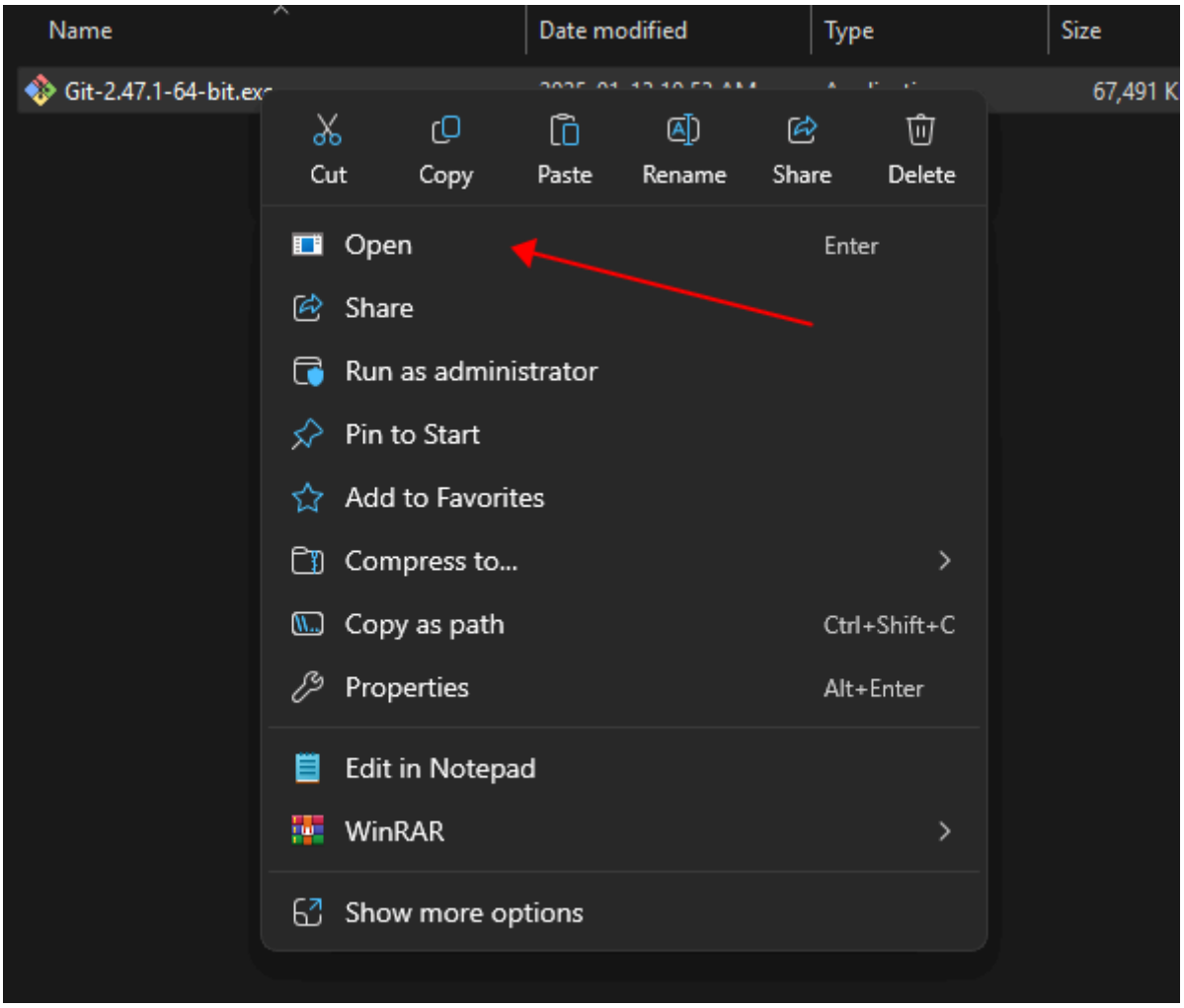


After installing the esp-ide tool, install a git.

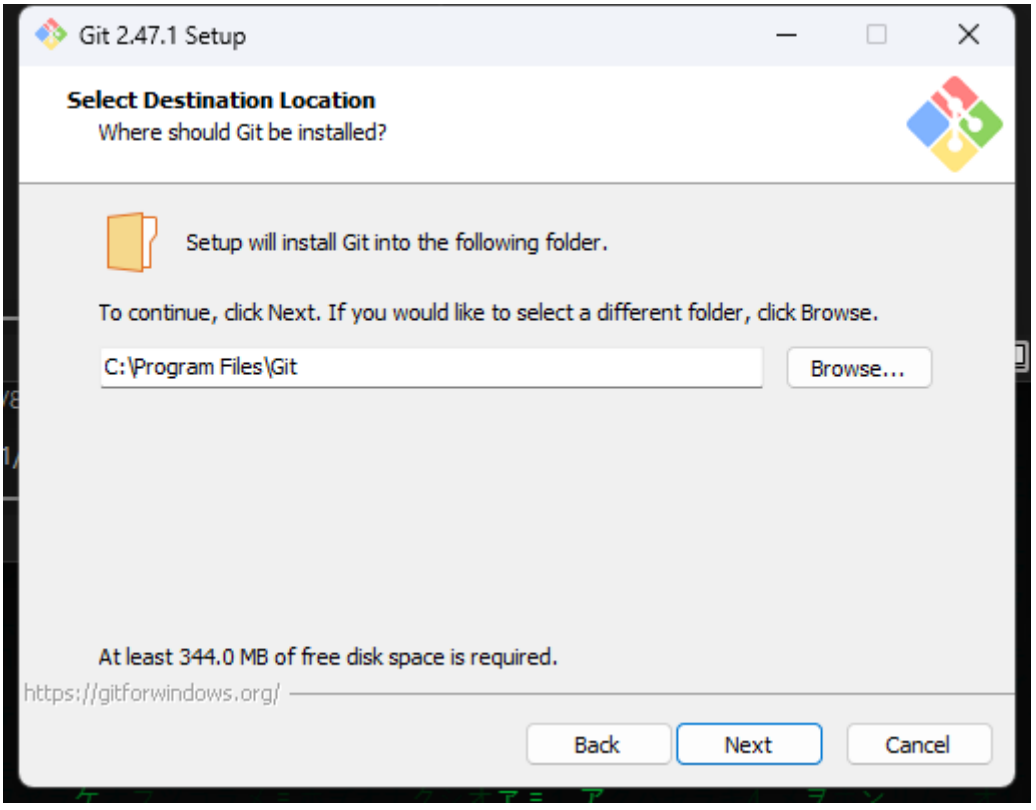
Link: [Git for Windows]



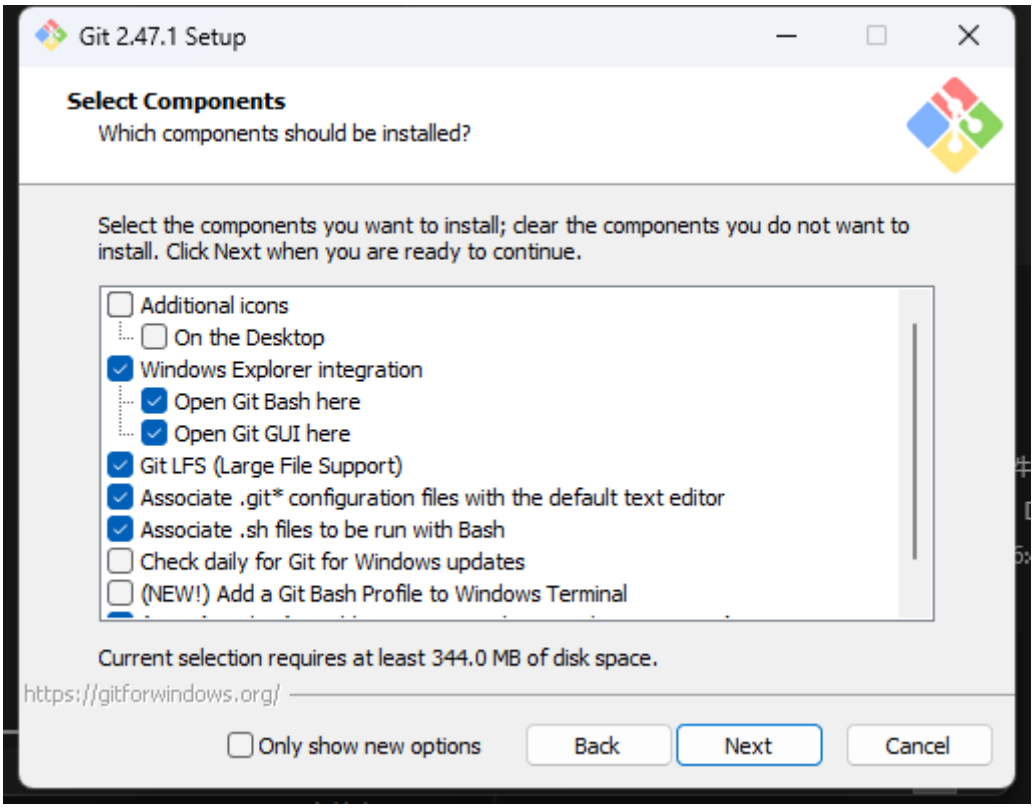
After downloading, open the installation,



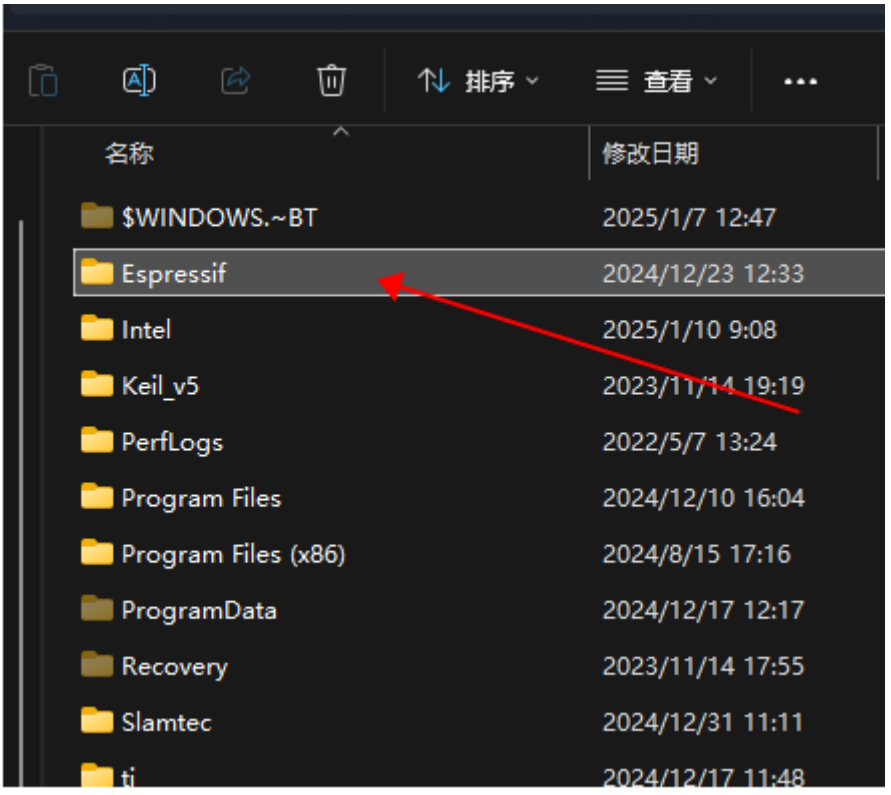
Select the installation path,



Always next,



Go to the back and install. After the installation is successful, a folder like this will appear on the C drive (it may also be on other drive letters)



This means that the installation is successful.

3. Start the ESP-IDF environment

At the end of the installation, if `Run ESP-IDF PowerShell Environment` or `Run ESP-IDF Command Prompt (cmd.exe)` is checked, the installer will start ESP-IDF in the selected prompt window.

Double-click to open the esp-ide software that appears on the desktop.


```
ESP-IDF 5.1 CMD - "C:\Espress
    C:\Espressif\tools\esp32ulp-elf\2.35_20220830\esp32ulp-elf\bin
    C:\Espressif\tools\cmake\3.24.0\bin
    C:\Espressif\tools\openocd-esp32\v0.12.0-esp32-20230921\openocd-esp32\bin
    C:\Espressif\tools\ninja\1.10.2\
    C:\Espressif\tools\idf-exe\1.0.3\
    C:\Espressif\tools\ccache\4.8\ccache-4.8-windows-x86_64
    C:\Espressif\tools\dfu-util\0.11\dfu-util-0.11-win64
    C:\Espressif\frameworks\esp-idf-v5.1.2\tools

Checking if Python packages are up to date...
Constraint file: C:\Espressif\espidf.constraints.v5.1.txt
Requirement files:
 - C:\Espressif\frameworks\esp-idf-v5.1.2\tools\requirements\requirements.core.txt
Python being checked: C:\Espressif\python_env\idf5.1_py3.11_env\Scripts\python.exe
Python requirements are satisfied.

Detected installed tools that are not currently used by active ESP-IDF version.
For removing old versions of idf-driver, idf-python-wheels use command 'python.exe C:\Espressif\frameworks\esp-idf-v5.1.
2\tools\idf_tools.py uninstall'
For free up even more space, remove installation packages of those tools. Use option 'python.exe C:\Espressif\frameworks
\esp-idf-v5.1.2\tools\idf_tools.py uninstall --remove-archives'.


Done! You can now compile ESP-IDF projects.
Go to the project directory and run:

  idf.py build


C:\Espressif\frameworks\esp-idf-v5.1.2>
```

The path to our esp32 environment is at the bottom: C:\Espressif\frameworks\esp-idf-v5.1.2

4. Start creating a project

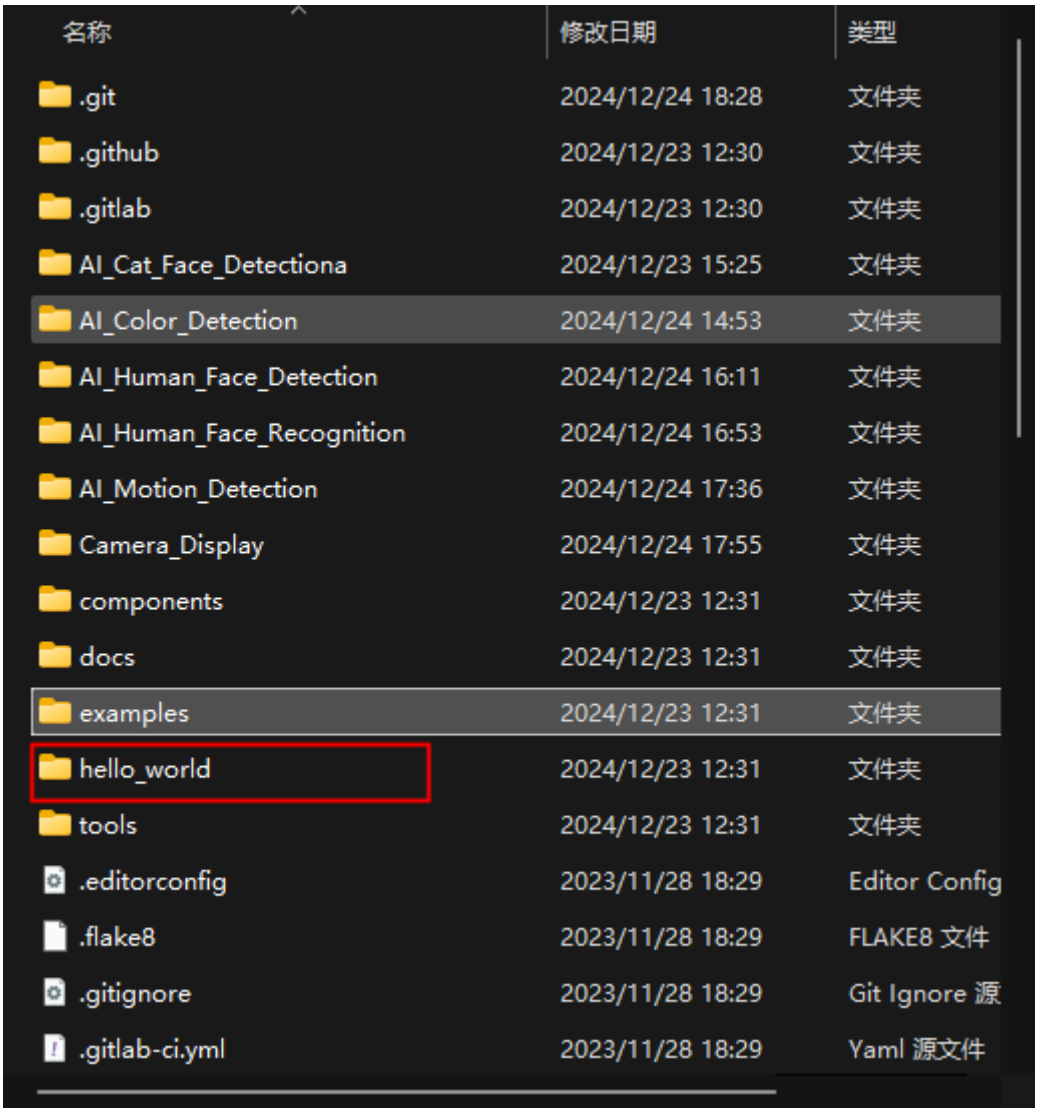
Now, you are ready to develop ESP32-S3 applications. You can start from the get-started/hello_world project in the examples directory in ESP-IDF.

Copy the get-started/hello_world project to the local `~/esp` directory:

```
xcopy /e /i examples\get-started\hello_world hello_world
```

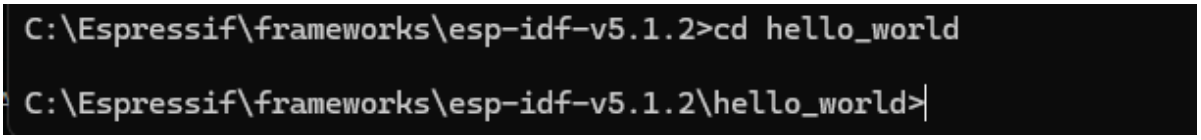
```
C:\Espressif\frameworks\esp-idf-v5.1.2>xcopy /e /i examples\get-started\hello_world hello_world
覆盖 C:\Espressif\frameworks\esp-idf-v5.1.2\hello_world\CMakeLists.txt (Y:是/N:否/A:全部)?y
examples\get-started\hello_world\CMakeLists.txt
覆盖 C:\Espressif\frameworks\esp-idf-v5.1.2\hello_world\pytest_hello_world.py (Y:是/N:否/A:全部)?y
examples\get-started\hello_world\pytest_hello_world.py
覆盖 C:\Espressif\frameworks\esp-idf-v5.1.2\hello_world\README.md (Y:是/N:否/A:全部)?a
examples\get-started\hello_world\README.md
examples\get-started\hello_world\sdkconfig.ci
examples\get-started\hello_world\main\CMakeLists.txt
examples\get-started\hello_world\main\hello_world_main.c
复制了 6 个文件

C:\Espressif\frameworks\esp-idf-v5.1.2>
```

You can see that it has been created,



Configure the new project,

```
cd hello_world
```



After opening a new project, you should first use `idf.py set-target esp32s3` to set the "target" chip. Note that this operation will clear and initialize the previous compilation and configuration of the project (if any). You can also directly configure the "target" as an environment variable (you can skip this step in this case).

```
idf.py set-target esp32s3
```

Setting successful:

```
f-v5.1.2/components/esp_local_ctrl C:/Espressif/frameworks/esp-idf-v5.1.2/components/esp_mm C:/Espressif/frameworks/esp-
idf-v5.1.2/components/esp_netif C:/Espressif/frameworks/esp-idf-v5.1.2/components/esp_netif_stack C:/Espressif/framework
s/esp-idf-v5.1.2/components/esp_partition C:/Espressif/frameworks/esp-idf-v5.1.2/components/esp_phy C:/Espressif/framewo
rks/esp-idf-v5.1.2/components/esp_pm C:/Espressif/frameworks/esp-idf-v5.1.2/components/esp_psram C:/Espressif/frameworks
/esp-idf-v5.1.2/components/esp_ringbuf C:/Espressif/frameworks/esp-idf-v5.1.2/components/esp_rom C:/Espressif/frameworks
/esp-idf-v5.1.2/components/esp_system C:/Espressif/frameworks/esp-idf-v5.1.2/components/esp_timer C:/Espressif/framework
s/esp-idf-v5.1.2/components/esp_wifi C:/Espressif/frameworks/esp-idf-v5.1.2/components/espcoredump C:/Espressif/framewor
ks/esp-idf-v5.1.2/components/esptool_py C:/Espressif/frameworks/esp-idf-v5.1.2/components/fatfs C:/Espressif/frameworks/
esp-idf-v5.1.2/components/freertos C:/Espressif/frameworks/esp-idf-v5.1.2/components/hal C:/Espressif/frameworks/esp-idf
-v5.1.2/components/heap C:/Espressif/frameworks/esp-idf-v5.1.2/components/http_parser C:/Espressif/frameworks/esp-idf-v5
.1.2/components/idf_test C:/Espressif/frameworks/esp-idf-v5.1.2/components/ieee802154 C:/Espressif/frameworks/esp-idf-v5
.1.2/components/json C:/Espressif/frameworks/esp-idf-v5.1.2/components/log C:/Espressif/frameworks/esp-idf-v5.1.2/compon
ents/lwip C:/Espressif/frameworks/esp-idf-v5.1.2/hello_world/main C:/Espressif/frameworks/esp-idf-v5.1.2/components/mbed
tls C:/Espressif/frameworks/esp-idf-v5.1.2/components/mqtt C:/Espressif/frameworks/esp-idf-v5.1.2/components/newlib C:/E
spressif/frameworks/esp-idf-v5.1.2/components/nvs_flash C:/Espressif/frameworks/esp-idf-v5.1.2/components/openthread C:/
Espressif/frameworks/esp-idf-v5.1.2/components/partition_table C:/Espressif/frameworks/esp-idf-v5.1.2/components/perfmon
 C:/Espressif/frameworks/esp-idf-v5.1.2/components/protobuf-c C:/Espressif/frameworks/esp-idf-v5.1.2/components/protocom
m C:/Espressif/frameworks/esp-idf-v5.1.2/components/pthread C:/Espressif/frameworks/esp-idf-v5.1.2/components/sdmmc C:/E
spressif/frameworks/esp-idf-v5.1.2/components/soc C:/Espressif/frameworks/esp-idf-v5.1.2/components/spi_flash C:/Espress
if/frameworks/esp-idf-v5.1.2/components/spiffs C:/Espressif/frameworks/esp-idf-v5.1.2/components/tcp_transport C:/Espres
sif/frameworks/esp-idf-v5.1.2/components/touch_element C:/Espressif/frameworks/esp-idf-v5.1.2/components/ulp C:/Espressi
f/frameworks/esp-idf-v5.1.2/components/unity C:/Espressif/frameworks/esp-idf-v5.1.2/components/usb C:/Espressif/framewor
ks/esp-idf-v5.1.2/components/vfs C:/Espressif/frameworks/esp-idf-v5.1.2/components/wear_levelling C:/Espressif/framework
s/esp-idf-v5.1.2/components/wifi_provisioning C:/Espressif/frameworks/esp-idf-v5.1.2/components/wpa_supplicant C:/Espres
sif/frameworks/esp-idf-v5.1.2/components/xtensa
-- Configuring done
-- Generating done
-- Build files have been written to: C:/Espressif/frameworks/esp-idf-v5.1.2/hello_world/build

C:\Espressif\frameworks\esp-idf-v5.1.2\hello_world>
```

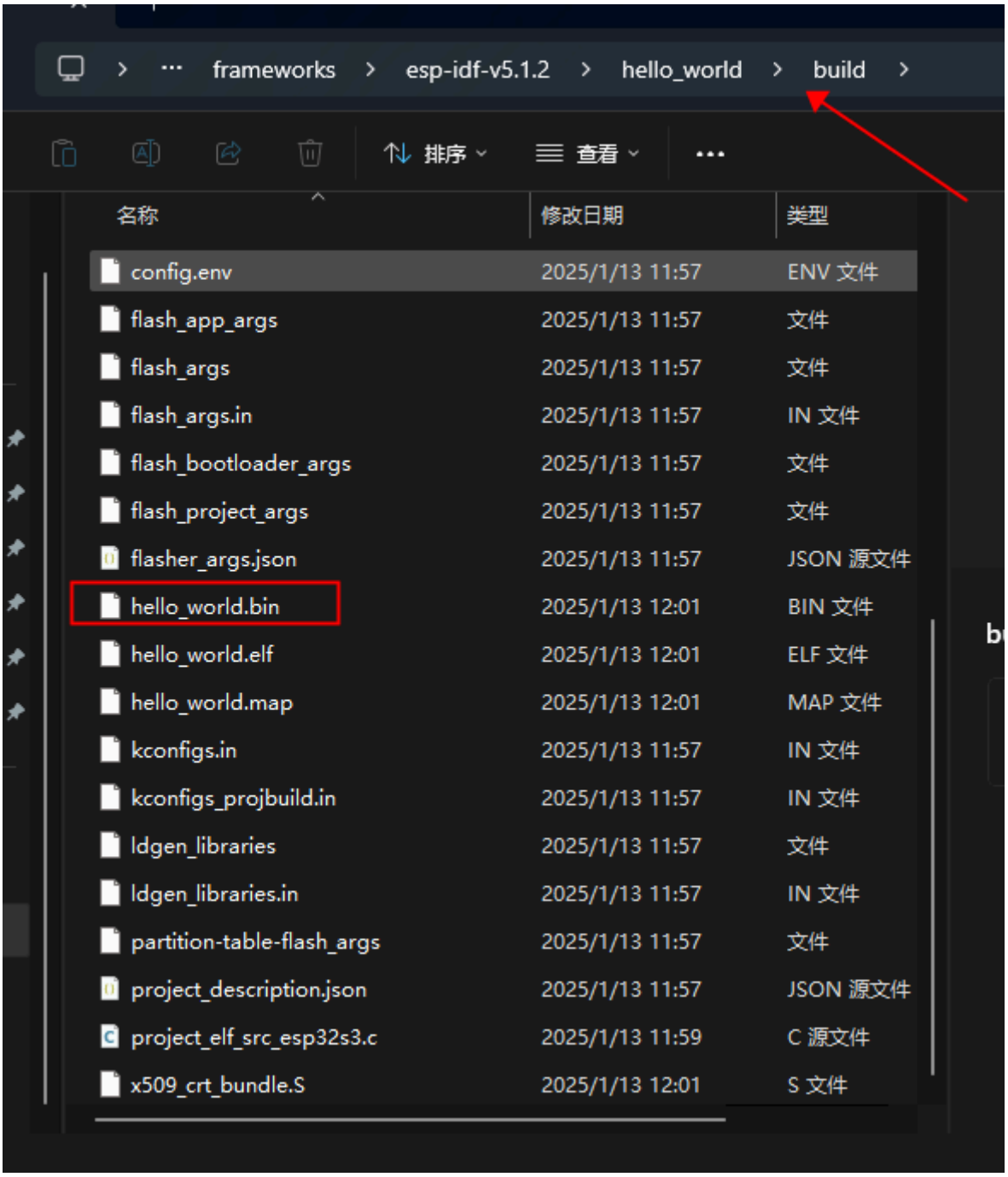
Compile the project,

```
idf.py build
```

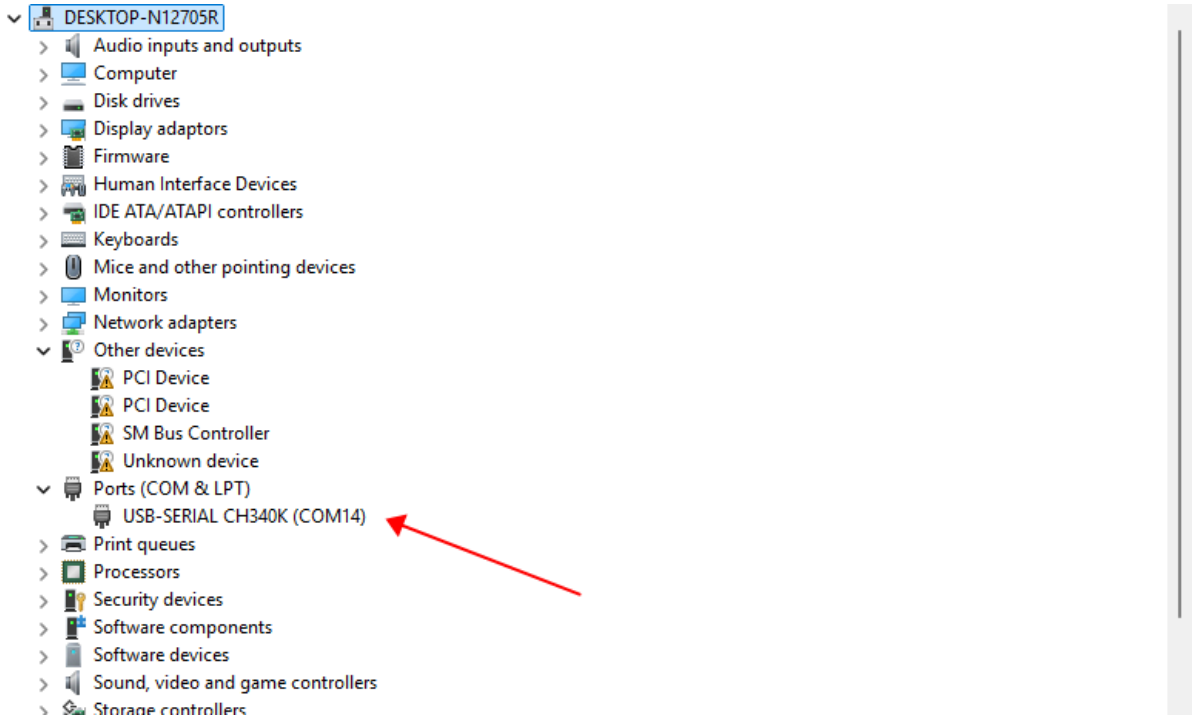
```
f-v5.1.2/components/log C:/Espressif/frameworks/esp-idf-v5.1.2/components/bootloader/subproject/main C:/Espressif/framew
orks/esp-idf-v5.1.2/components/bootloader/subproject/components/micro-ecc C:/Espressif/frameworks/esp-idf-v5.1.2/compone
nts/newlib C:/Espressif/frameworks/esp-idf-v5.1.2/components/partition_table C:/Espressif/frameworks/esp-idf-v5.1.2/comp
onents/soc C:/Espressif/frameworks/esp-idf-v5.1.2/components/spi_flash C:/Espressif/frameworks/esp-idf-v5.1.2/components
/xtensa
-- Configuring done
-- Generating done
-- Build files have been written to: C:/Espressif/frameworks/esp-idf-v5.1.2/hello_world/build/bootloader
[109/110] Generating binary image from built executableesptool.py v4.7.dev3
Creating esp32s3 image...
Merged 1 ELF section
Successfully created esp32s3 image.
Generated C:/Espressif/frameworks/esp-idf-v5.1.2/hello_world/build/bootloader/bootloader.bin
[110/110] cmd.exe /C "cd /D C:\Espressif\frameworks\esp-id...sp-idf-v5.1.2/hello_world/build/bootloader/bootloader.bin"B
ootloader binary size 0x51c0 bytes. 0x2e40 bytes (36%) free.
[942/943] Generating binary image from built executableesptool.py v4.7.dev3
Creating esp32s3 image...
Merged 2 ELF sections
Successfully created esp32s3 image.
Generated C:/Espressif/frameworks/esp-idf-v5.1.2/hello_world/build/hello_world.bin
[943/943] cmd.exe /C "cd /D C:\Espressif\frameworks\esp-id...ameworks/esp-idf-v5.1.2/hello_world/build/hello_world.bin"h
ello_world.bin binary size 0x31640 bytes. Smallest app partition is 0x100000 bytes. 0xce9c0 bytes (81%) free.

Project build complete. To flash, run this command:
C:\Espressif\python_env\idf5.1_py3.11_env\Scripts\python.exe ..\components\esptool_py\esptool\esptool.py -p (PORT) -b 46
0800 --before default_reset --after hard_reset --chip esp32s3  write_flash --flash_mode dio --flash_size 2MB --flash_fre
q 80m 0x0 build\bootloader\bootloader.bin 0x8000 build\partition_table\partition-table.bin 0x10000 build\hello_world.bin
or run 'idf.py -p (PORT) flash'

C:\Espressif\frameworks\esp-idf-v5.1.2\hello_world>
```

If everything goes well, a .bin file will be generated after the compilation is complete.

It will be stored in the build folder.



Connect the device to the computer, and you can directly use the COM port to burn. Open the computer's device manager to check what port is recognized.



Enter the following command in the terminal to write the program in.

```
idf.py -p com14 flash
```

```
Writing at 0x0017ec00... (99 %)
Writing at 0x0017f000... (99 %)
Writing at 0x0017f400... (99 %)
Writing at 0x0017f800... (99 %)
Writing at 0x0017fc00... (99 %)
Writing at 0x00180000... (99 %)
Writing at 0x00180400... (99 %)
Writing at 0x00180800... (99 %)
Writing at 0x00180c00... (99 %)
Writing at 0x00181000... (99 %)
Writing at 0x00181400... (99 %)
Writing at 0x00181800... (99 %)
Writing at 0x00181c00... (99 %)
Writing at 0x00182000... (99 %)
Writing at 0x00182400... (100 %)
Wrote 1517568 bytes at 0x00010000 in 47.0 seconds (258.2 kbit/s)...
Hash of data verified.
Erasing flash...
Took 0.07s to erase flash block
Writing at 0x00008000... (33 %)
Writing at 0x00008400... (66 %)
Writing at 0x00008800... (100 %)
Wrote 3072 bytes at 0x00008000 in 0.1 seconds (271.6 kbit/s)...
Hash of data verified.

Leaving...
Hard resetting via RTS pin...
Done

C:\Espressif\frameworks\esp-idf-v5.1.2\AI_Cat_Face_Detectiona\esp32Board_wifi>
```

If the above log appears, the download is successful.