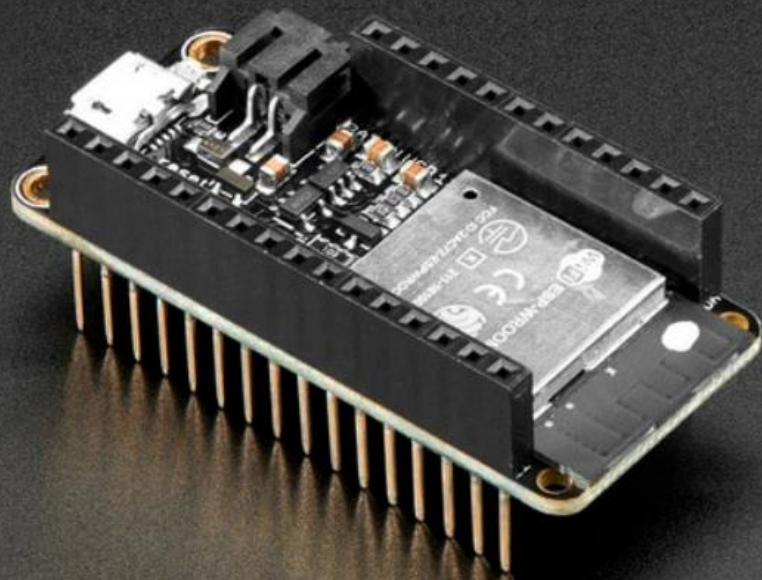


ESP32

온라인 워크숍 초급과정



과정 목차

❖ 1일차

- > ESP32 소개
- > 개발환경 설정 (Window Version)
- > ESP-IDF 설치
- > 빌드환경 설정
- > Edit Tool 설정
- > Extra Session : app_main

❖ 2일차

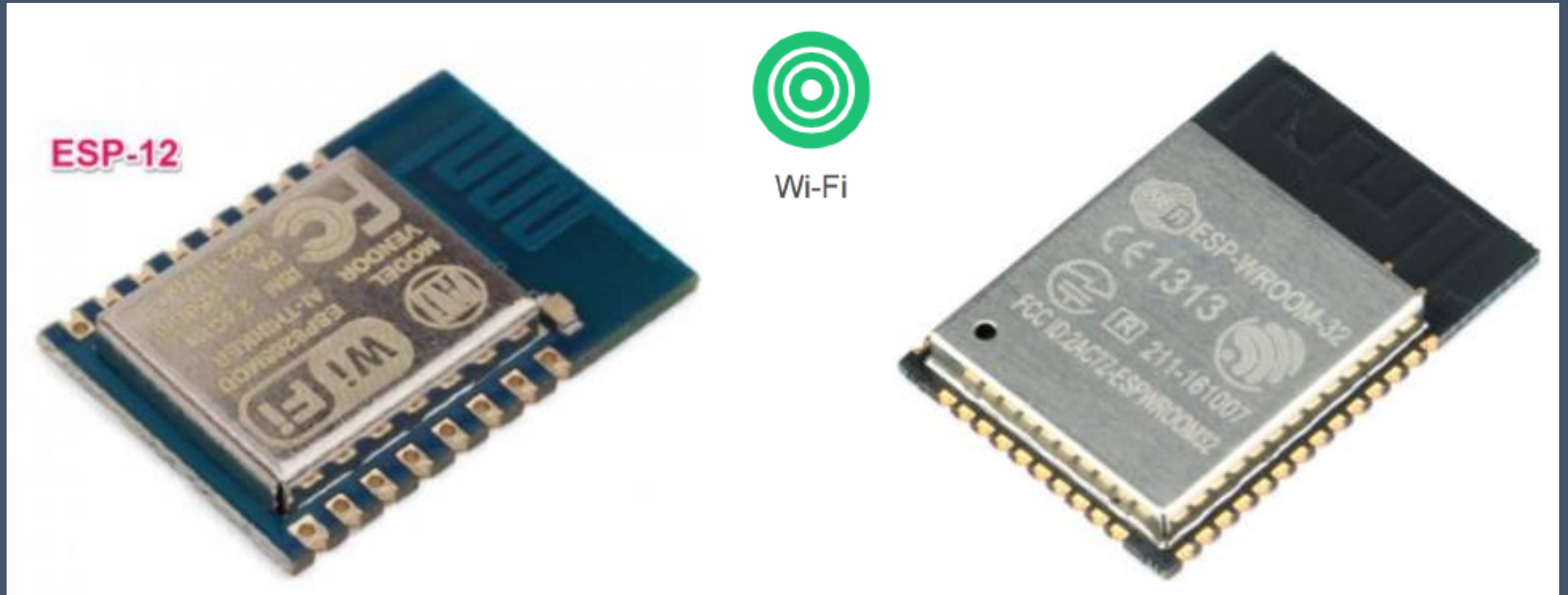
- > GPIO & GPIO Interrupt
 - Output : LED 실습
 - Input : Button 실습
- > UART
 - UART Echo 실습
- > Timer & Timer Interrupt
 - 사용자용 Timer 작성

❖ 3일차

- > Analog Input
 - 가변저항 읽기 실습
- > PWM
 - LED 디밍 실습 (HS mode/ LS mode 체크)
- > I2C
 - 다른 프로젝트 소스코드 포팅, HTS221 온습도 센서 실습

1. ESP32 소개

Why ESP32?



乐鑫信息科技

ESPRESSIF SYSTEMS

ESP8266 specification

- Processor: L106 32-bit RISC microprocessor core based on the Tensilica Xtensa Diamond Standard 106Micro running at 80 MHz[†]
- Memory:
 - 32 KiB instruction RAM
 - 32 KiB instruction cache RAM
 - 80 KiB user data RAM
 - 16 KiB ETS system data RAM
- External QSPI flash: up to 16 MiB is supported (512 KiB to 4 MiB typically included)
- IEEE 802.11 b/g/n Wi-Fi
 - Integrated TR switch, balun, LNA, power amplifier and matching network
 - WEP or WPA/WPA2 authentication, or open networks

Price 1/10 !!!

ESP32 specification

- Processors:

- CPU: Xtensa dual-core (or single-core) 32-bit LX6 microprocessor, operating at 160 or 240 MHz and performing at up to 600 DMIPS
- Ultra low power (ULP) co-processor

- Memory: 520 KiB SRAM

- Wireless connectivity:

- Wi-Fi: 802.11 b/g/n
- Bluetooth: v4.2 BR/EDR and BLE

How did they get quality in a short time?

https://www.espressif.com/en/media_overview/news/bug-bounty

Chip	SDK	Award
ESP8266 / ESP8285	ESP8266_NONOS_SDK	2,000 USD
ESP8266 / ESP8285	ESP8266_RTOS_SDK	2,000 USD
ESP32 Security - Bugs	ESP-IDF	500 USD
ESP32 Security - Proof of Concept	ESP-IDF	1,729 USD



ESP32 Features

Wi-Fi Key Features

- 802.11 b/g/n
- 802.11 n (2.4 GHz), up to 150 Mbps

BT Key Features

- Compliant with Bluetooth v4.2 BR/EDR and BLE specifications
- Class-1, class-2 and class-3 transmitter without external power amplifier
- Enhanced Power Control
- +12 dBm transmitting power
- NZIF receiver with -94 dBm BLE sensitivity
- Adaptive Frequency Hopping (AFH)
- Standard HCI based on SDIO/SPI/UART
- High-speed UART HCI, up to 4 Mbps
- Bluetooth 4.2 BR/EDR BLE dual mode controller
- Synchronous Connection-Oriented/Extended (SCO/eSCO)
- CVSD and SBC for audio codec
- Bluetooth Piconet and Scatternet
- Multi-connections in Classic BT and BLE
- Simultaneous advertising and scanning

MCU and Advanced Features

1.CPU and Memory

- Xtensa® single-/dual-core 32-bit LX6 microprocessor(s), up to 600 MIPS (200 MIPS for ESP32-S0WD/ESP32-U4WDH, 400 MIPS for ESP32-D2WD) **240 MHz**
- 448 KB ROM
- 520 KB SRAM
- 16 KB SRAM in RTC
- QSPI supports multiple flash/SRAM chips

2.Clocks and Timers

- Internal 8 MHz oscillator with calibration
- Internal RC oscillator with calibration
- External 2 MHz ~ 60 MHz crystal oscillator (40 MHz only for Wi-Fi/BT functionality)
- External 32 kHz crystal oscillator for RTC with calibration
- Two timer groups, including **2 × 64-bit timers** and **1 × main watchdog** in each group
 - **One RTC timer**
 - **RTC watchdog**

3.Advanced Peripheral Interfaces

- 34 × programmable GPIOs
- 12-bit SAR ADC up to 18 channels
- 2 × 8-bit DAC
- 10 × touch sensors
- 4 × SPI
- 2 × I²S
- 2 × I²C
- 3 × UART
- 1 host (SD/eMMC/SDIO)
- 1 slave (SDIO/SPI)
- Ethernet MAC interface with dedicated DMA and IEEE 1588 support
- CAN 2.0
- IR (TX/RX)
- Motor PWM
- LED PWM up to 16 channels
- Hall sensor

Security

- Secure boot
- Flash encryption
- 1024-bit OTP, up to 768-bit for customers
- Cryptographic hardware acceleration:
 - AES
 - Hash (SHA-2)
 - RSA
 - ECC
 - Random Number Generator (RNG)

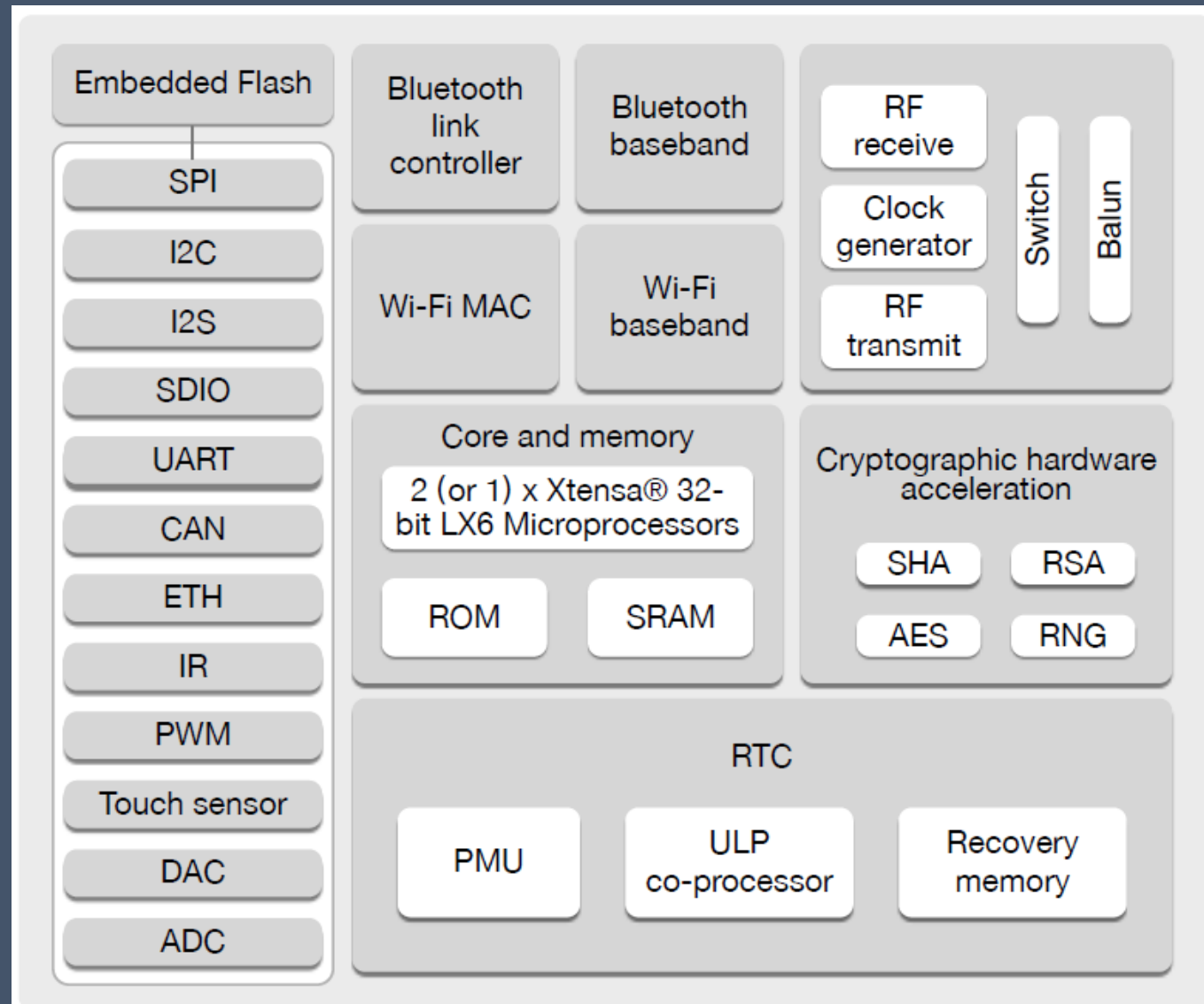
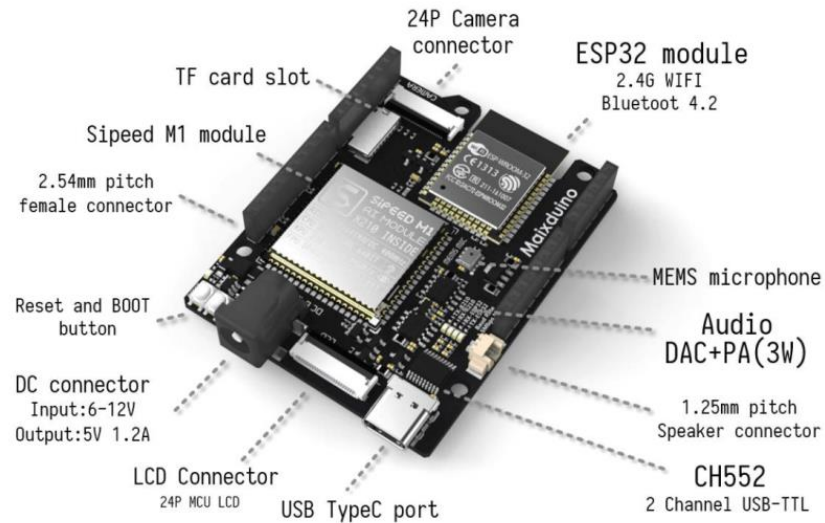


Figure 1: Functional Block Diagram

ESP32 Commercial Products



2. ESP32 개발환경 설정

References

- ✓ esp32_datasheet_en.pdf : CPU DataSheet
- ✓ esp32_technical_reference_manual_en.pdf : CPU Programming Guide
- ✓ esp32-wroom-32_datasheet_en.pdf : CPU Module DataSheet
- ✓ esp32_devkitc_v4-sch.pdf : ESP32 Devkit Schematics
- ✓ SDK Guide : <https://docs.espressif.com/projects/esp-idf/en/latest/esp32/index.html>



개발을 위한 준비 *ESP-IDF (Espressif IoT Development Framework)*

✓ 하드웨어

- ESP32 Board
- USB Cable
- Window (Linux, macOS도 지원) : 실제 개발환경 구성은 윈도우 보다는 리눅스 또는 맥이 유리

✓ 소프트웨어

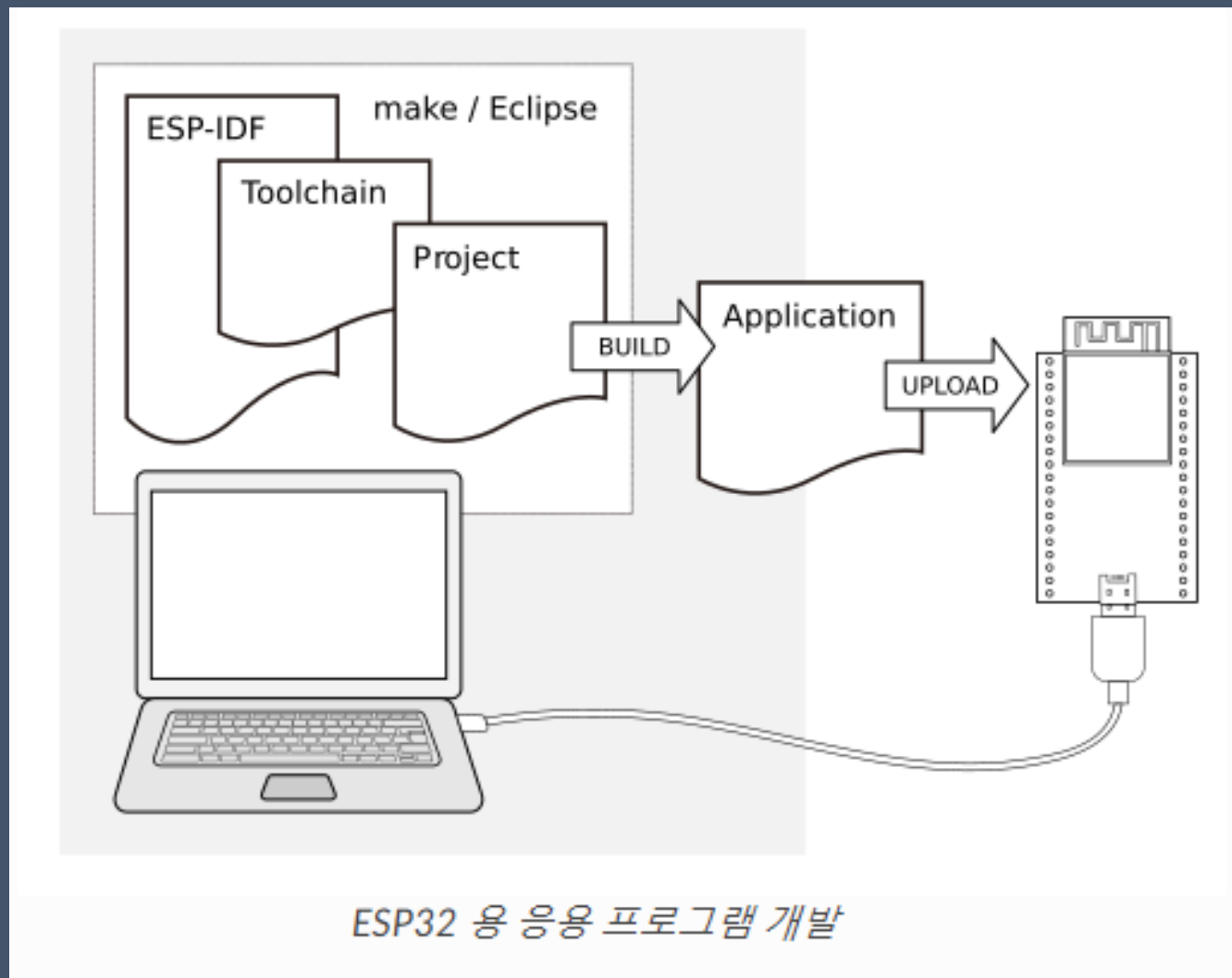
- ESP32 코드 컴파일을 위한 툴체인
- 빌드도구 : Cmake 및 Ninja 전체 애플리케이션 빌드
- ESP32용 API(Application Programming Interface, 여기서는 라이브러리 및 소스코드 포함)와 툴체인을 운영하는 스크립트(py)를 포함하는 ESP-IDF
- C 프로그램 (**프로젝트**)을 작성하는 **텍스트 편집기** (예 : Eclipse, VSCode)

툴체인 : 개발도구를 모아둔 것 (컴파일러, 링커, 디버거, 라이브러리, 어셈블러, 로더, libc 등등, 자세한 내용은 툴체인 매뉴얼 참고)

https://ko.wikipedia.org/wiki/GNU_%ED%88%B4%EC%B2%B4%EC%9D%B8

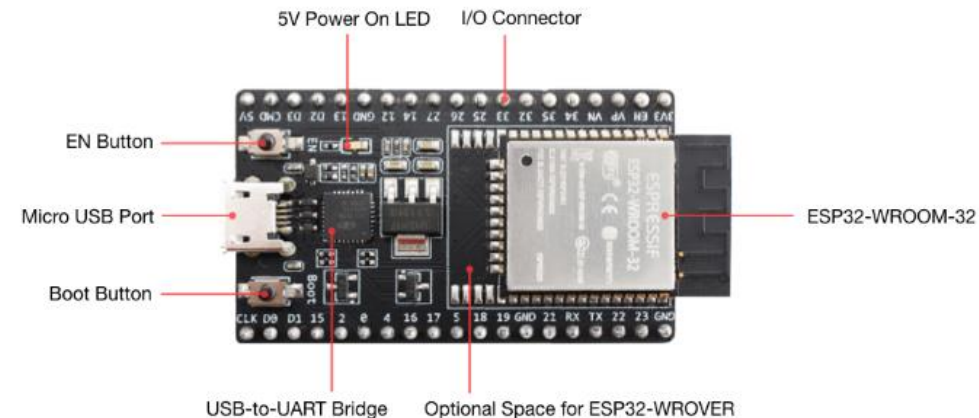
https://en.wikipedia.org/wiki/GNU_toolchain

전체 개발 흐름



전체 개발 흐름

- ✓ 핀 D0, D1, D2, D3, CMD 및 CLK는 내부적으로 ESP32와 SPI 플래시 메모리 간의 통신에 사용됩니다. USB 커넥터 근처의 양쪽에 그룹화되어 있습니다. 이러한 핀은 SPI 플래시 메모리 / SPI RAM에 대한 액세스를 방해 할 수 있으므로 사용하지 마십시오.
- ✓ GPIO16 및 GPIO17 핀은 ESP32-WROOM 및 ESP32-SOLO-1 모듈이 있는 보드에서만 사용할 수 있습니다. ESP32-WROVER 모듈이 있는 보드에는 내부용으로 예약 된 핀이 있습니다.



ESP32-DevKitC V4 with ESP32-WROOM-32 module soldered

Key Component	Description
ESP32-WROOM-32	A module with ESP32 at its core.
EN	Reset button.
Boot	Download button. Holding down Boot and then pressing EN initiates Firmware Download mode for downloading firmware through the serial port.
USB-to-UART Bridge	Single USB-UART bridge chip provides transfer rates of up to 3 Mbps.
Micro USB Port	USB interface. Power supply for the board as well as the communication interface between a computer and the ESP32 module.
5V Power On LED	Turns on when the USB or an external 5V power supply is connected to the board. For details see the schematics in Related Documents .
I/O	Most of the pins on the ESP module are broken out to the pin headers on the board. You can program ESP32 to enable multiple functions such as PWM, ADC, DAC, I2C, I2S, SPI, etc.

전원 공급 장치 옵션

보드에 전원을 공급하는 상호 배타적 인 세 가지 방법이 있습니다.

- 마이크로 USB 포트, 기본 전원 공급 장치
- 5V / GND 헤더 핀
- 3V3 / GND 헤더 핀

ⓘ 경고

위의 옵션 중 하나만 사용하여 전원 공급 장치를 제공해야 합니다. 그렇지 않으면 보드 및 / 또는 전원 공급 장치가 손상 될 수 있습니다.

3. ESP-IDF 개발도구 설치

64비트 윈도우버전만 지원, 32비트 윈도우는 레거시 GNU Make Build System 사용

64비트 윈도우버전에서 작업 진행할 것

Python, Git, Cross Compiler

크로스 컴파일러

크로스 컴파일러(cross compiler)는 **컴파일러**가 실행되는 플랫폼이 아닌 다른 플랫폼에서 실행 가능한 코드를 생성할 수 있는 **컴파일러**이다. **크로스 컴파일러** 툴은 임베디드 시스템 혹은 여러 플랫폼에서 실행파일을 생성하는데 사용된다.

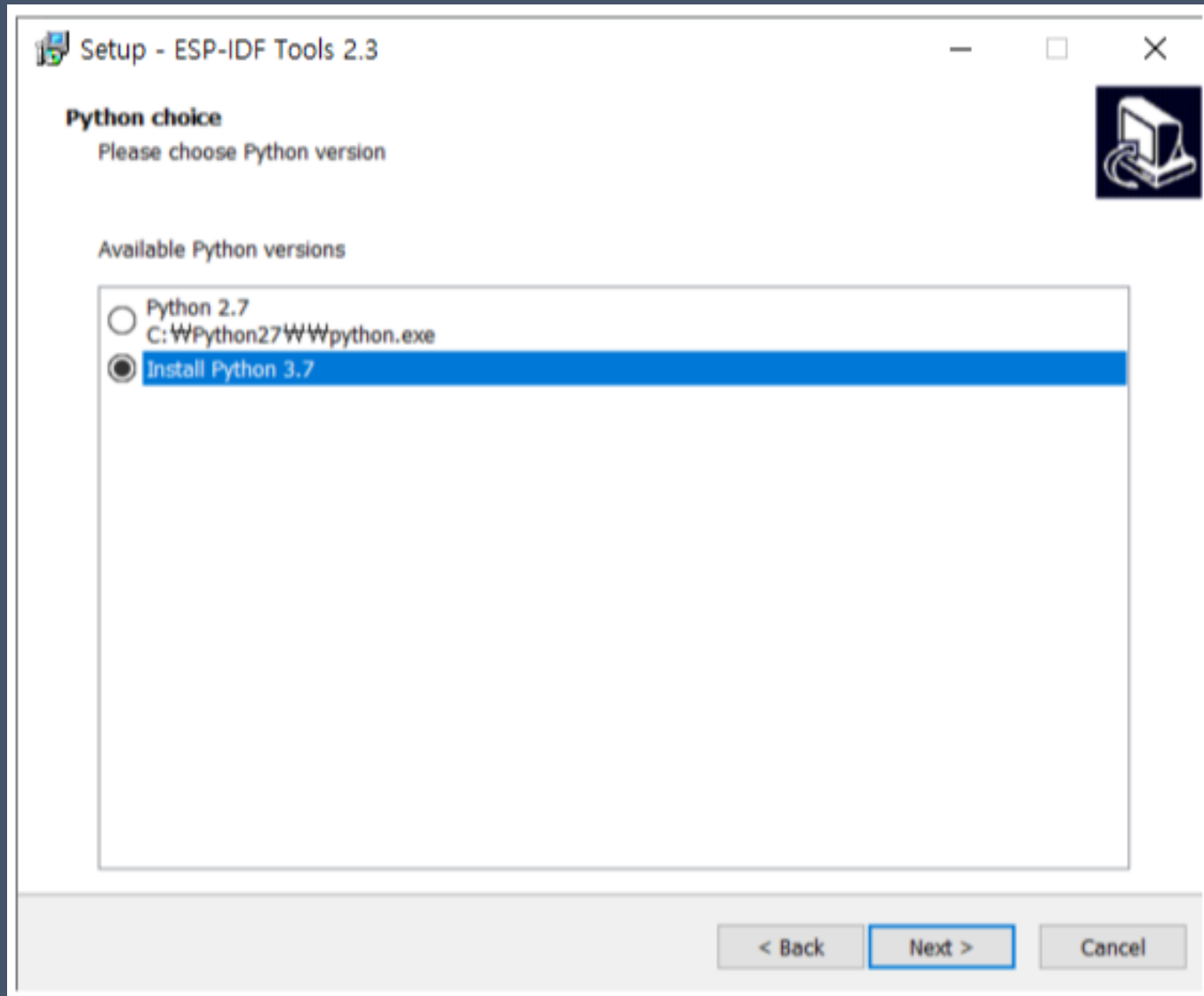
ESP-IDF 도구 설치 프로그램

ESP-IDF 필수 구성 요소를 설치하는 가장 쉬운 방법은 다음 URL에서 ESP-IDF 도구 설치 관리자를 다운로드하는 것입니다.

<https://dl.espressif.com/dl/esp-idf-tools-setup-2.3.exe>

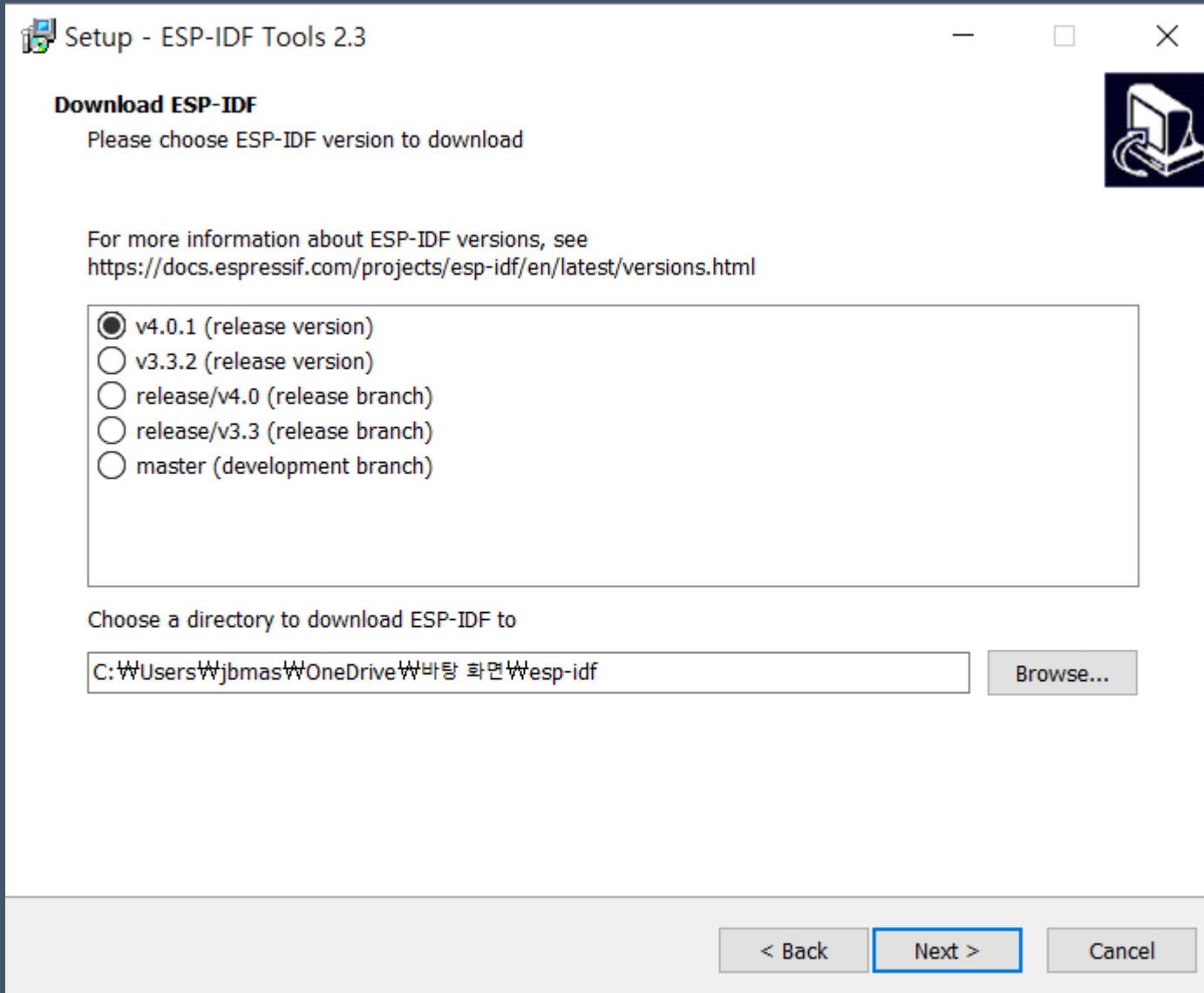
크로스 컴파일러, OpenOCD, cmake 및 Ninja 빌드 도구가 포함됩니다. 설치 프로그램은 Python 3.7 및 Git For Windows 용 설치 프로그램이 컴퓨터에 아직 설치되어 있지 않은 경우 다운로드하여 실행할 수 있습니다 .

개발도구 설치



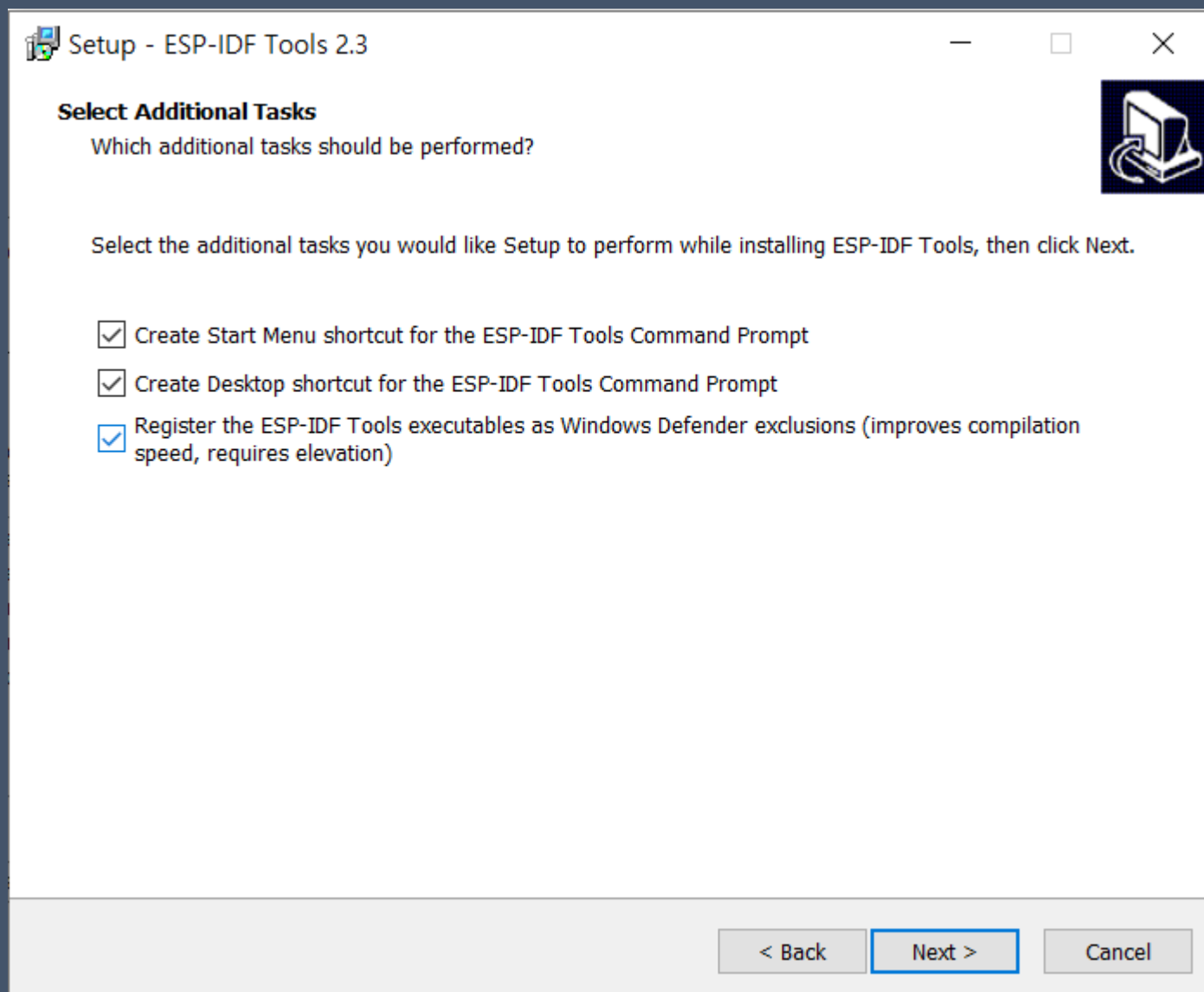
✓ 기존에 구축한 시스템과 별도로 설치하는 것이 유리합니다. 특히 파이썬 버전을 맞춰두고 빌드하는 다른 시스템이 있으면 필히 Intall 지정해서 설치하시기 바랍니다.

개발도구 설치



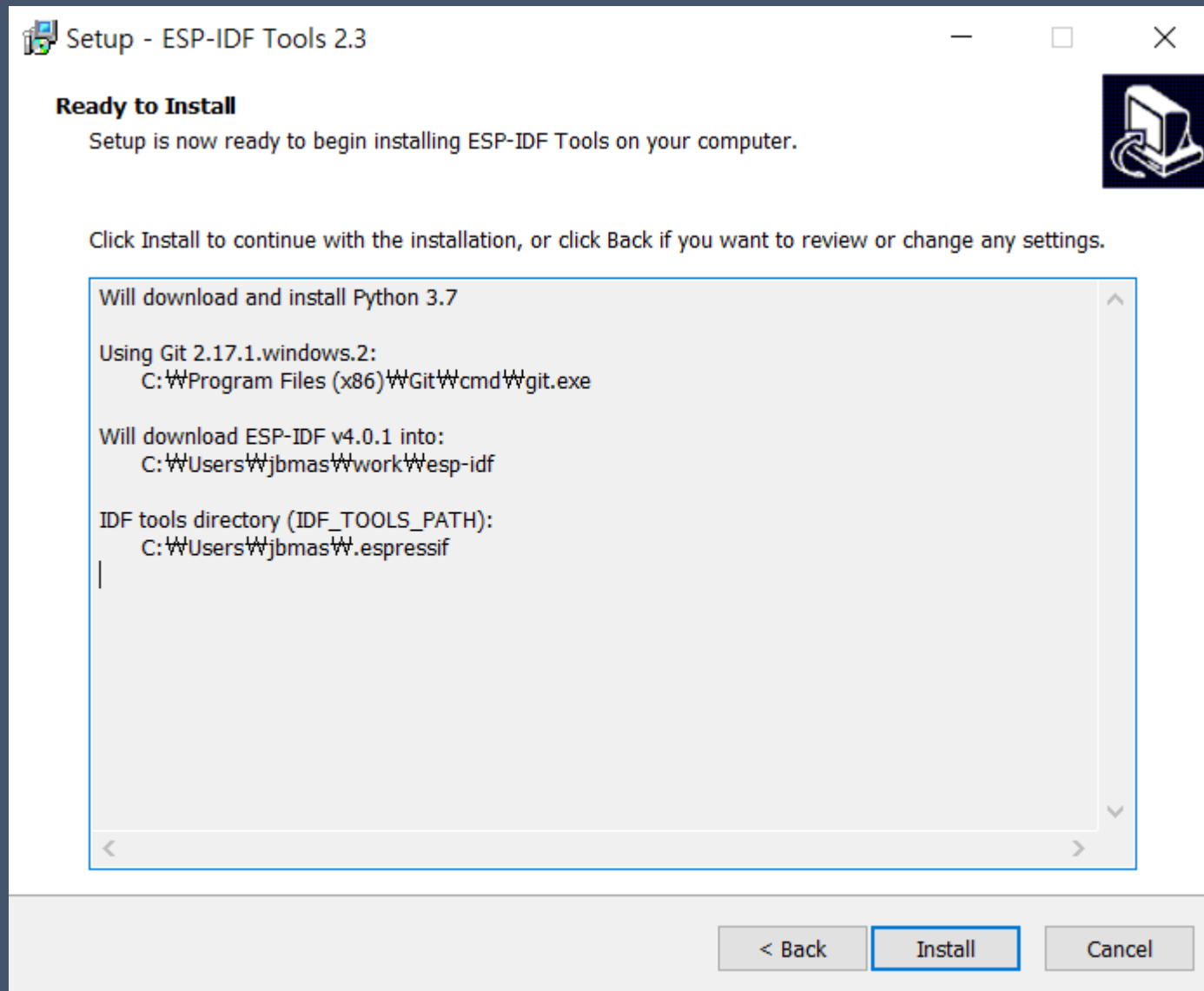
✓ V4.0.1 (릴리즈버전 사용)
다운로드 받을 디렉토리 주소 체크
(한글 디렉토리X)

개발도구 설치

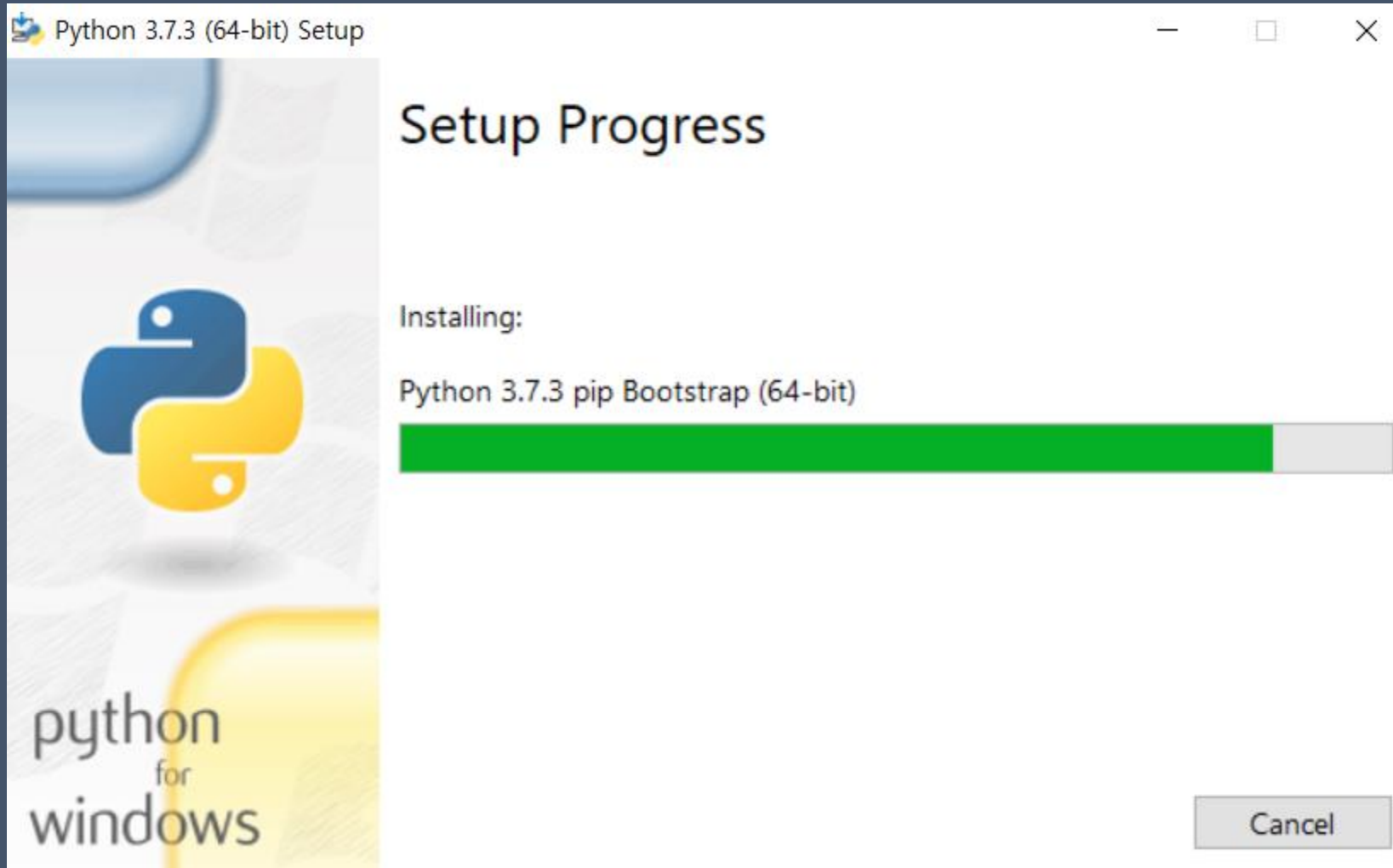


개발도구 설치

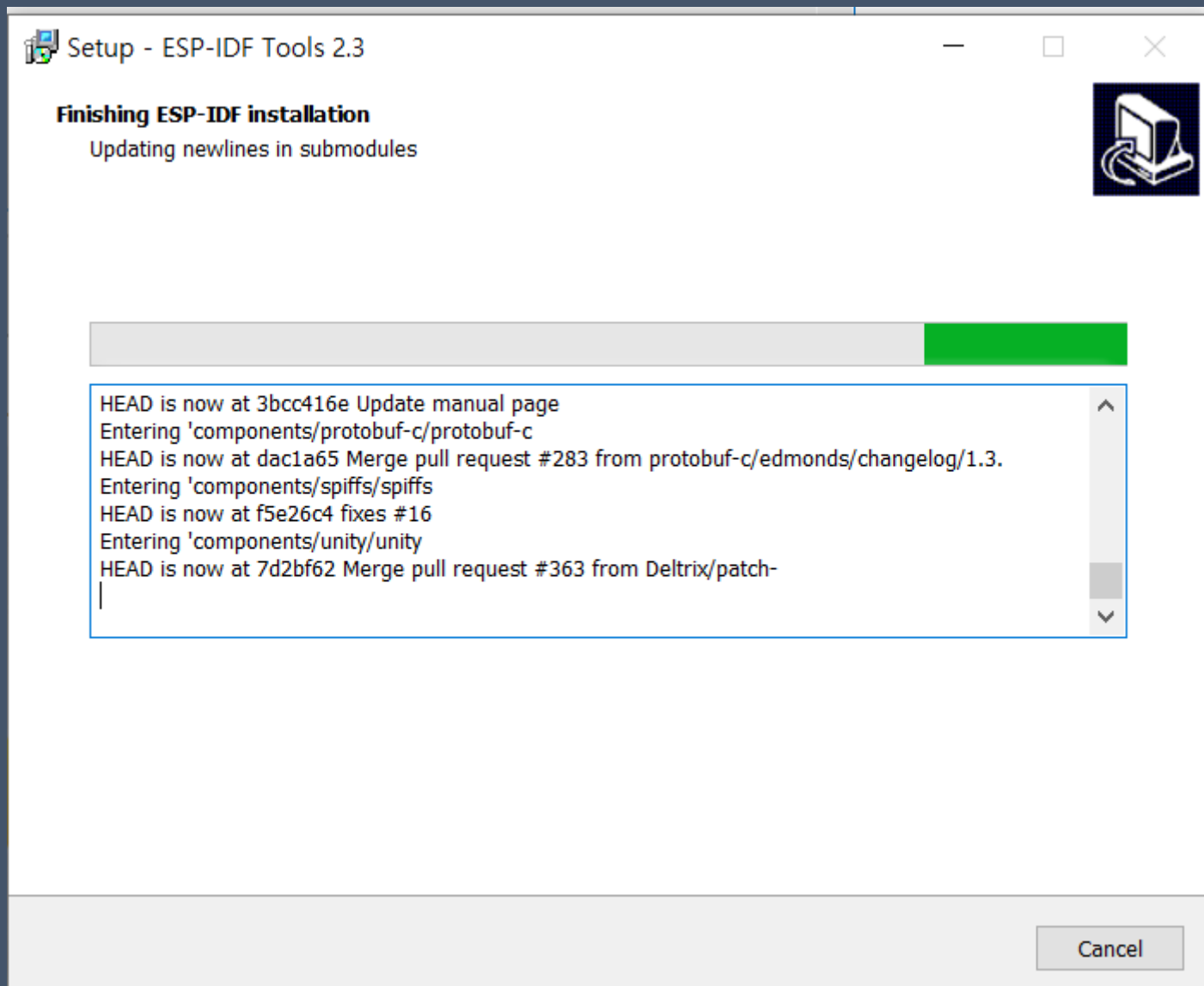
설치 전 최종 확인



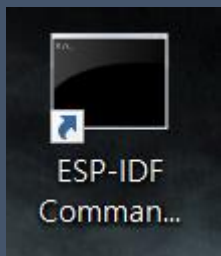
개발도구 설치



개발도구 설치



개발도구 설치



단축 아이콘 클릭



```
C:\Users\jibmas\AppData\Local\Programs\Python\Python37\cmd.exe - "C:\Users\jibmas\AppData\Local\Programs\Python\Python37\cmd.exe"
Using Python in C:\Users\jibmas\AppData\Local\Programs\Python\Python37\Python
Python 3.7.3
Using Git in C:\Program Files (x86)\Git\cmd\git
git version 2.17.1.windows.2
Setting IDF_PATH: C:\Users\jibmas\work\esp-idf

Adding ESP-IDF tools to PATH...
Not using an unsupported version of tool ninja found in PATH: 1.5.3.
C:\Users\jibmas\AppData\Local\Programs\Python\Python37\Scripts\python.exe C:\Users\jibmas\work\esp-idf\tools\xtensa-esp32-elf\bin\xtensa-esp32-elf.exe
C:\Users\jibmas\AppData\Local\Programs\Python\Python37\Scripts\python.exe C:\Users\jibmas\work\esp-idf\tools\esp32ulp-elf\bin\esp32ulp-elf.exe
C:\Users\jibmas\AppData\Local\Programs\Python\Python37\Scripts\python.exe C:\Users\jibmas\work\esp-idf\tools\cmake\bin\cmake.exe
C:\Users\jibmas\AppData\Local\Programs\Python\Python37\Scripts\python.exe C:\Users\jibmas\work\esp-idf\tools\openocd-esp32\bin\openocd-esp32.exe
C:\Users\jibmas\AppData\Local\Programs\Python\Python37\Scripts\python.exe C:\Users\jibmas\work\esp-idf\tools\mconf\bin\idf-20190628.exe
C:\Users\jibmas\AppData\Local\Programs\Python\Python37\Scripts\python.exe C:\Users\jibmas\work\esp-idf\tools\ninja\bin\ninja.exe
C:\Users\jibmas\AppData\Local\Programs\Python\Python37\Scripts\python.exe C:\Users\jibmas\work\esp-idf\tools\idf-exe\bin\idf-exe.exe
C:\Users\jibmas\AppData\Local\Programs\Python\Python37\Scripts\python.exe C:\Users\jibmas\work\esp-idf\tools\ccache\bin\ccache.exe
C:\Users\jibmas\AppData\Local\Programs\Python\Python37\Scripts\python.exe C:\Users\jibmas\work\esp-idf\tools\python_env\idf4.0_py3.7_env\Scripts\python.exe
C:\Users\jibmas\work\esp-idf\tools

Checking if Python packages are up to date...
Python requirements from C:\Users\jibmas\work\esp-idf\requirements.txt are satisfied.

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

idf.py build

C:\Users\jibmas\work\esp-idf>
```

4. 빌드환경 설정

```
C:\Users\jbmas\work>mkdir esp_test
```

```
C:\Users\jbmas\work>cd esp_test
```

```
C:\Users\jbmas\work\esp_test>xcopy /e /i %IDF_PATH%\examples\get-started\hello_world  
hello_world
```

```
C:\Users\jbmas\work\esp-idf\examples\get-started\hello_world\CMakeLists.txt
```

```
C:\Users\jbmas\work\esp-idf\examples\get-started\hello_world\Makefile
```

```
C:\Users\jbmas\work\esp-idf\examples\get-started\hello_world\README.md
```

```
C:\Users\jbmas\work\esp-idf\examples\get-started\hello_world\main\CMakeLists.txt
```

```
C:\Users\jbmas\work\esp-idf\examples\get-started\hello_world\main\component.mk
```

```
C:\Users\jbmas\work\esp-idf\examples\get-started\hello_world\main\hello_world_main.c
```

6개 파일이 복사되었습니다.

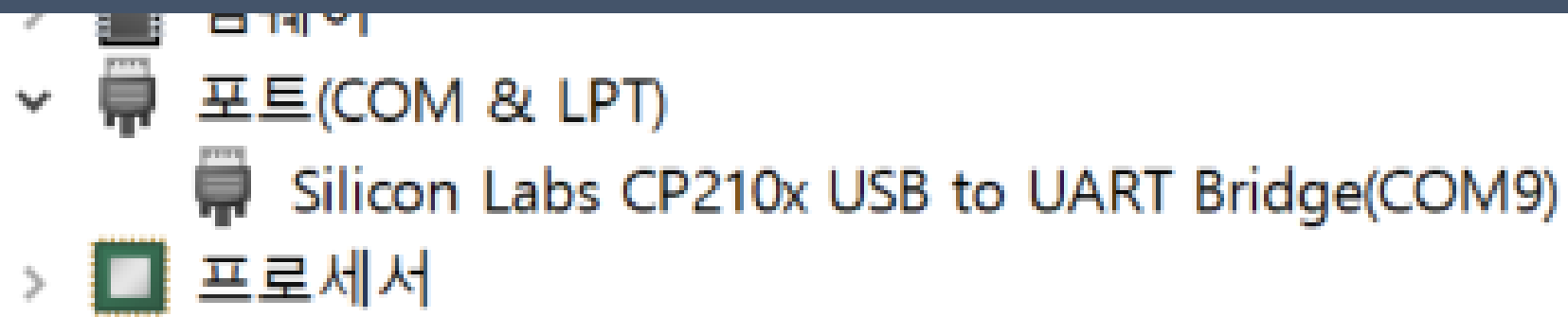
```
C:\Users\jbmas\work\esp_test>
```



```
W>idf.py menuconfig
```

[illegible]

장치관리자 --> 포트 확인



₩>idf.py build

₩>idf.py -p COM9 -b 115200 flash

Leaving...

Hard resetting via RTS pin...

Done

```
detecting chip type... ESP32
Chip is ESP32D0WDQ6 (revision 1)
Features: WiFi, BT, Dual Core, 240MHz, VRef calibration in efuse, Coding Scheme None
Crystal is 40MHz
MAC: 3c:71:bf:4b:e7:7c
Uploading stub...
Running stub...
Stub running...
Configuring flash size...
Compressed 3072 bytes to 103...
Wrote 3072 bytes (103 compressed) at 0x00008000 in 0.0 seconds (effective 947.7 kbit/s)...
Flash of data verified.
Compressed 25376 bytes to 14952...
Wrote 25376 bytes (14952 compressed) at 0x00001000 in 1.3 seconds (effective 152.1 kbit/s)...
Flash of data verified.
Compressed 147808 bytes to 76827...
Writing at 0x00020000... (100 %)

```

<펌웨어 Flash 실패>

esp-idf 개발 보드를 수동으로 재설정하려면 **부팅** 버튼 (GPIO0)을 누른 상태에서 **EN** 버튼 (CHIP_PU)을 누릅니다 .

```
COM9 - Tera Term VT
File Edit Setup Control Window Help
I (153) esp_image: segment 4: paddr=0x00020018 vaddr=0x400d0018 size=0x12ca0 ( 76960) map
I (181) esp_image: segment 5: paddr=0x00032cc0 vaddr=0x40088610 size=0x01474 ( 5236) load
I (190) boot: Loaded app from partition at offset 0x10000
I (190) boot: Disabling RNG early entropy source...
I (190) cpu_start: Pro cpu up.
I (194) cpu_start: Application information:
I (199) cpu_start: Project name:      hello-world
I (204) cpu_start: App version:      1
I (209) cpu_start: Compile time:      Jun 26 2020 01:37:20
I (215) cpu_start: ELF file SHA256:  879a865b2298fbf6...
I (221) cpu_start: ESP-IDF:          v4.0.1-dirty
I (226) cpu_start: Starting app cpu, entry point is 0x40081038
I (212) cpu_start: App cpu up.
I (237) heap_init: Initializing. RAM available for dynamic allocation:
I (243) heap_init: At 3FFAE6E0 len 00001920 (6 KiB): DRAM
I (249) heap_init: At 3FFB3100 len 0002CF00 (179 KiB): DRAM
I (256) heap_init: At 3FFE0440 len 00003AE0 (14 KiB): D/IRAM
I (262) heap_init: At 3FFE4350 len 0001BCB0 (111 KiB): D/IRAM
I (268) heap_init: At 40089A84 len 0001657C (89 KiB): IRAM
I (275) cpu_start: Pro cpu start user code
I (293) spi_flash: detected chip: generic
I (293) spi_flash: flash io: dio
W (294) spi_flash: Detected size(4096k) larger than the size in the binary image header(2048k). Using the size in the binary image header.
I (304) cpu_start: Starting scheduler on PRO CPU.
I (0) cpu_start: Starting scheduler on APP CPU.
Hello world!
This is ESP32 chip with 2 CPU cores, WiFi/BT/BLE, silicon revision 1, 2MB external flash
Restarting in 10 seconds...
Restarting in 9 seconds...
Restarting in 8 seconds...
Restarting in 7 seconds...
Restarting in 6 seconds...
Restarting in 5 seconds...
```

```

Tera Term - [disconnected] VT
File Edit Setup Control Window Help
GPIO[4] intr, val: 0
GPIO[4] intr, val: 0
GPIO[4] intr, val: 0
GPIO[4] intr, val: 0
cnt: 17
GPIO[4] intr, val: 0
GPIO[4] intr, val: 0
GPIO[4] intr, val: 0
GPIO[4] intr, val: 0
GPIO[4] intr, val: 0
cnt: 18
GPIO[4] intr, val: 0
GPIO[4] intr, val: 1
GPIO[4] intr, val: 0
cnt: 19
GPIO[4] intr, val: 1
GPIO[4] intr, val: 0
cnt: 20
GPIO[4] intr, val: 0
cnt: 21
GPIO[4] intr, val: 0
cnt: 22
cnt: 23
cnt: 24
cnt: 25
cnt: 26
cnt: 27
cnt: 28
cnt: 29
cnt: 30
cnt: 31
cnt: 32
cnt: 33
cnt: 34

```

```
File "C:\Users\jbmas\work\esp-idf\components\esptool_py\esptool\esptool.py", line 3201, in <module>
    _main()
File "C:\Users\jbmas\work\esp-idf\components\esptool_py\esptool\esptool.py", line 3194, in _main
    main()
File "C:\Users\jbmas\work\esp-idf\components\esptool_py\esptool\esptool.py", line 2883, in main
    esp = ESPLoader.detect_chip(each_port, initial_baud, args.before, args.trace)
File "C:\Users\jbmas\work\esp-idf\components\esptool_py\esptool\esptool.py", line 273, in detect_chip
    detect_port = ESPLoader(port, baud, trace_enabled=trace_enabled)
File "C:\Users\jbmas\work\esp-idf\components\esptool_py\esptool\esptool.py", line 237, in __init__
    self._port = serial.serial_for_url(port)
File "C:\Users\jbmas\work\esp-idf\components\esptool_py\esptool\esptool.py", line 88, in serial
    for_url
        instance.open()
File "C:\Users\jbmas\work\esp-idf\components\esptool_py\esptool\esptool.py", line 62, in open
    raise SerialException("could not open port {!r}: {!r}".format(self.portstr, ctypes.WinError()))
serial.serialutil.SerialException: could not open port 'COM9': PermissionError(13, '액세스가 거부되었습니다.', None, 5)
esptool.py failed with exit code 1

C:\Users\jbmas\work\esp_test\hello_world>
```

시리얼 모니터 프로그램을 띄어놓으면 액세스 거부 하기 때문에
시리얼 모니터 프로그램은 끄고 펌웨어 Flash 할 것

5. Edit-tool 설정

✓ 개발툴 설정

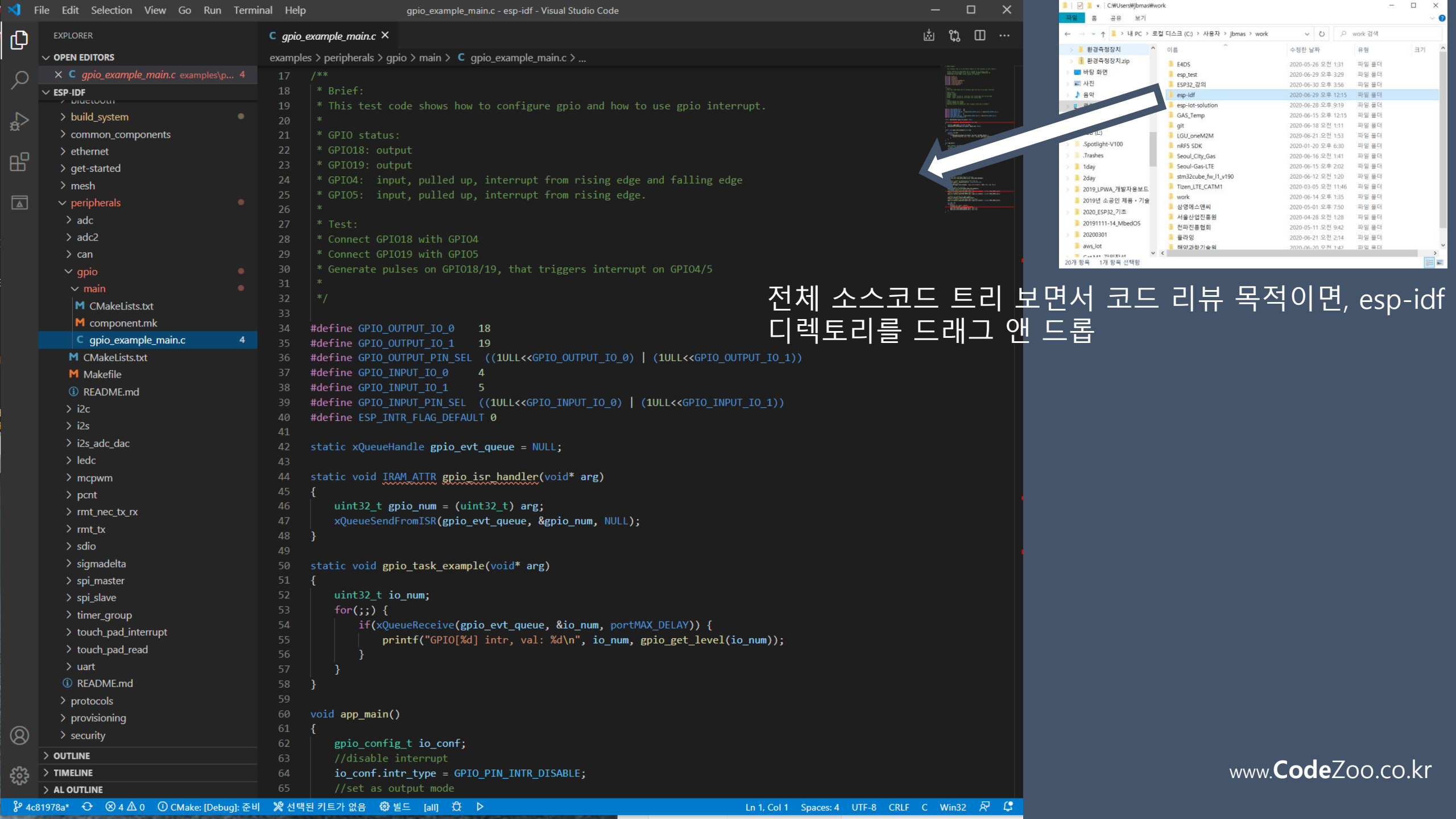
이클립스 : <https://docs.espressif.com/projects/esp-idf/en/latest/esp32/get-started/eclipse-setup.html>

VSCode : <https://github.com/Deous/VSC-Guide-for-esp32>

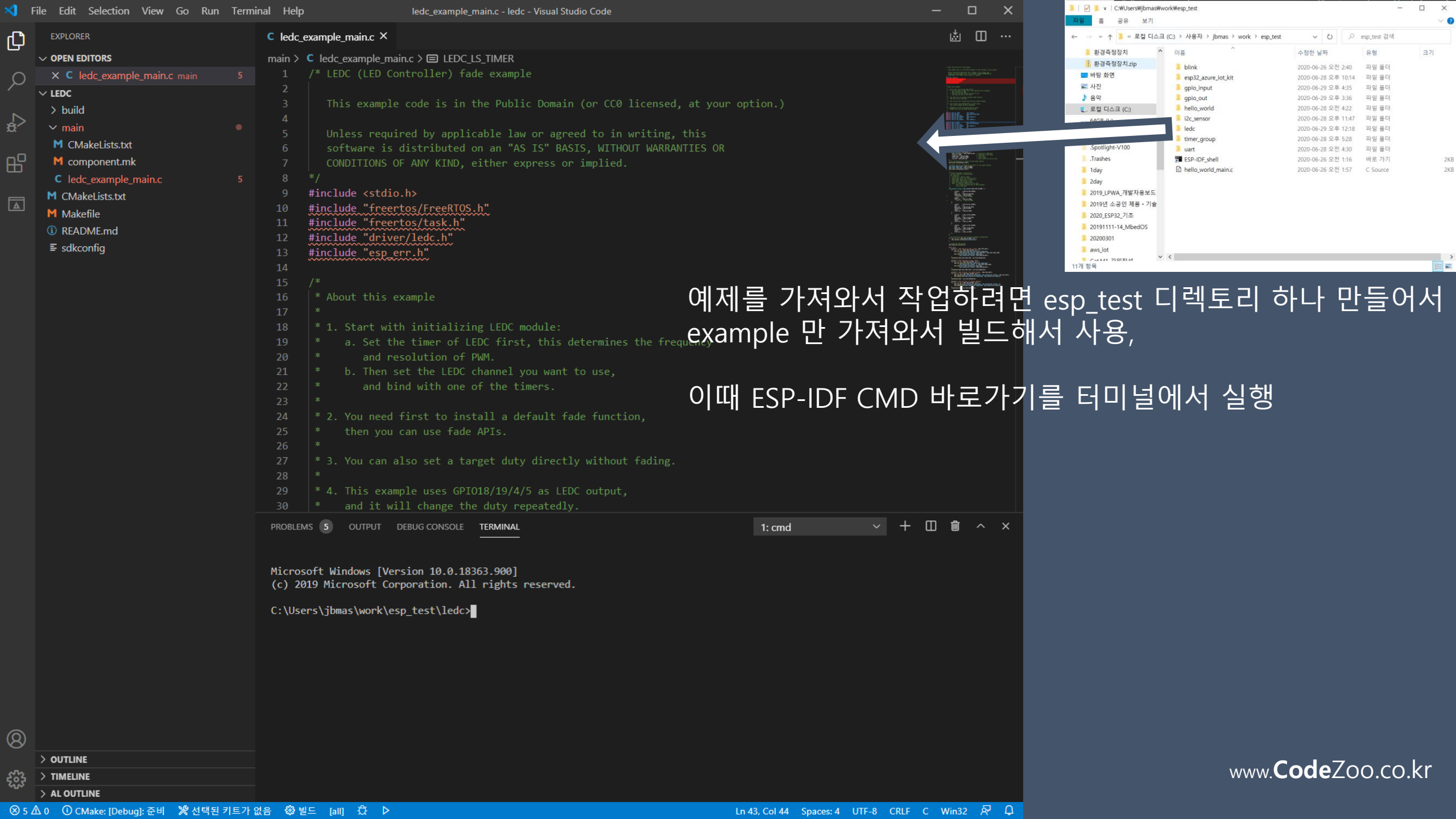
✓ Another

이유 : 다른 CLI 툴을 사용하고 있을경우 파이썬과 다른 툴의 버전이 달라서 둘다 망가트릴수 있다.

MbedOS CLI를 사용하는 입장에서 python2.7 과 python3.7을 혼용하는 입장이라 ESP-IDF Command Prompt와 VSCode를 같이 사용하기로 함.



전체 소스코드 트리 보면서 코드 리뷰 목적이면, esp-idf 디렉토리를 드래그 앤 드롭



예제를 가져와서 작업하려면 esp_test 디렉토리 하나 만들어서 example 만 가져와서 빌드해서 사용,

이때 ESP-IDF CMD 바로가기를 터미널에서 실행

C:\Users\jbmas\work\esp_test 디렉터리

2020-06-29	오전 12:32	<DIR>	.
2020-06-29	오전 12:32	<DIR>	..
2020-06-26	오전 02:40	<DIR>	blink
2020-06-26	오전 01:16		1,072 ESP-IDF_shell.lnk
2020-06-28	오후 10:14	<DIR>	esp32_azure_iot_kit
2020-06-26	오전 02:04	<DIR>	gpio
2020-06-28	오전 04:22	<DIR>	hello_world
2020-06-26	오전 01:57		1,273 hello_world_main.c
2020-06-28	오후 11:47	<DIR>	i2c_sensor
2020-06-29	오전 12:36	<DIR>	ledc
2020-06-28	오후 05:28	<DIR>	timer_group
2020-06-28	오전 04:30	<DIR>	uart
		2개 파일	2,345 바이트
		10개 디렉터리	805,432,242,176 바이트 남음

C:\Users\jbmas\work\esp_test>ESP-IDF_shell.lnk

PROBLEMS

5

OUTPUT

DEBUG CONSOLE

TERMINAL

1: cmd

```
-- Detecting CXX compile features - done
CMake Error at CMakeLists.txt:5 (message):
  Current directory 'C:/Users/jbmas/work/esp-idf' is not buildable.  Change
  directories to one of the example projects in
  'C:/Users/jbmas/work/esp-idf/examples' and try again.
```

```
-- Configuring incomplete, errors occurred!
See also "C:/Users/jbmas/work/esp-idf/build/CMakeFiles/CMakeOutput.log".
cmake failed with exit code 1
```

```
C:\Users\jbmas\work\esp-idf>cd ..
```

```
C:\Users\jbmas\work>cd esp_test
```

```
C:\Users\jbmas\work\esp_test>cd ledc
```

```
C:\Users\jbmas\work\esp_test\ledc>idf.py build
```

없음  빌드 [all]  ▶

Ln 173, Col 2

```
../main/ledc_example_main.c:114:9: note: (near initialization for 'ledc_channel')  
../main/ledc_example_main.c:122:9: warning: excess elements in array initializer
```

```
{  
^
```

```
../main/ledc_example_main.c:122:9: note: (near initialization for 'ledc_channel')
```

```
[827/827] Generating binary image from built executable
```

```
esptool.py v2.8
```

```
Generated C:/Users/jbmas/work/esp_test/ledc/build/ledc.bin
```

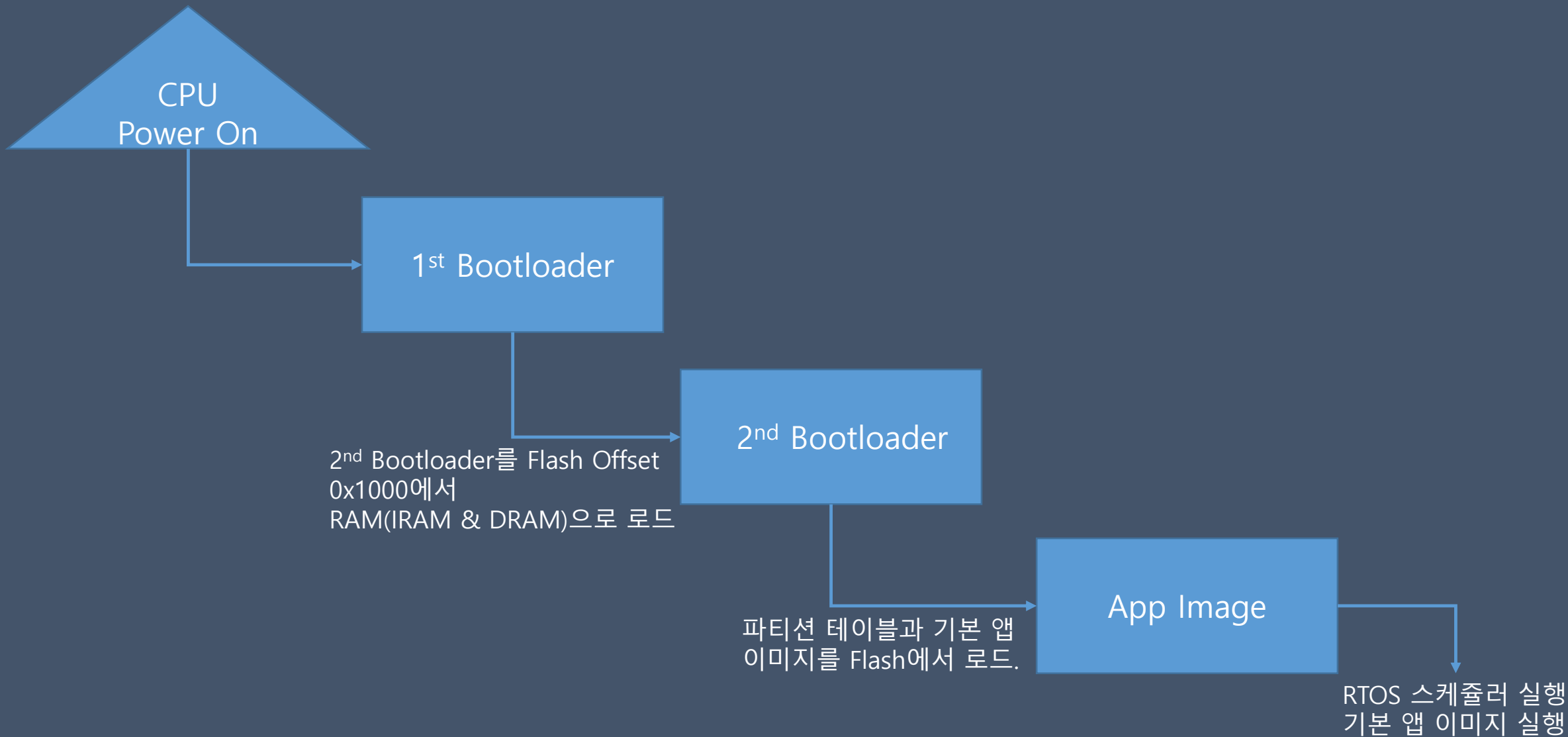
Project build complete. To flash, run this command:

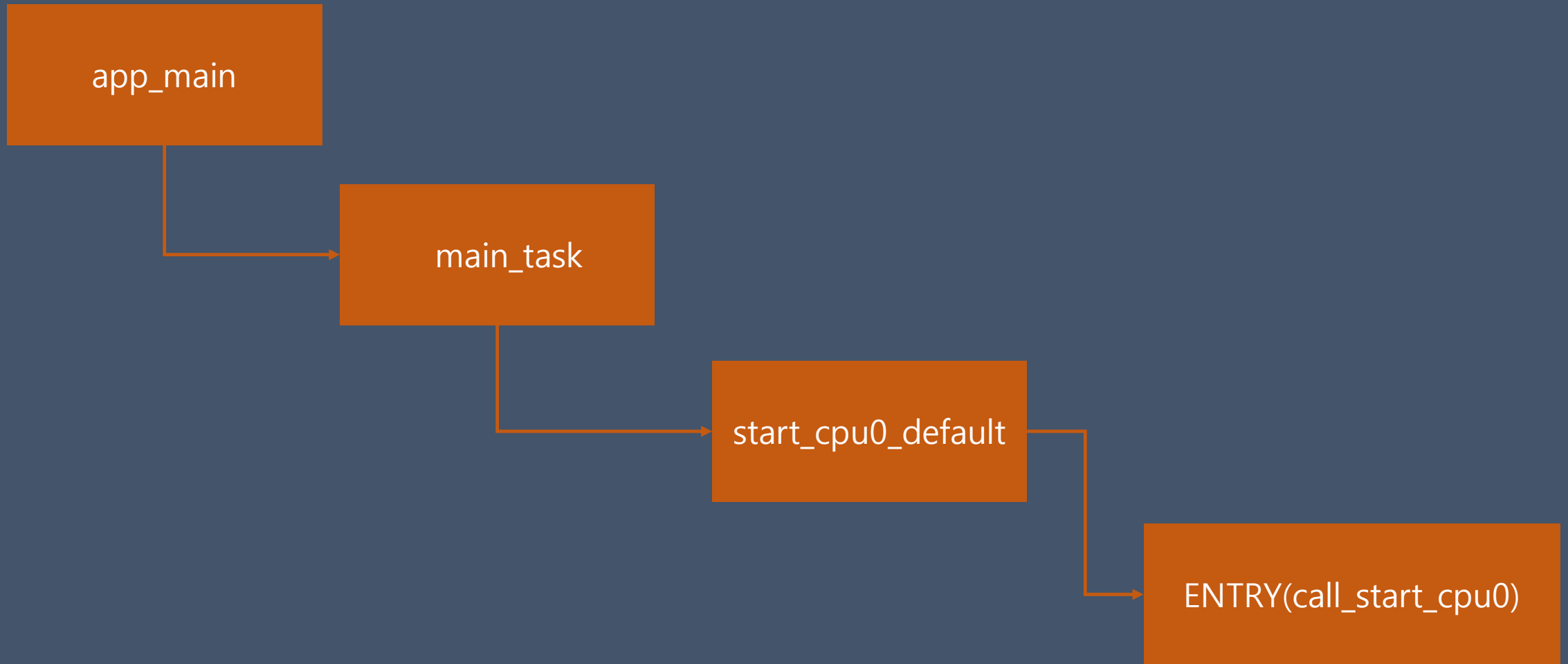
```
C:\Users\jbmas\.espressif\python_env\idf4.0_py3.7_env\Scripts\python.exe ..\..\esp-idf\components\esptool_py\esptool\esptool.  
py -p (PORT) -b 460800 --before default_reset --after hard_reset write_flash --flash_mode dio --flash_size detect --flash_fre  
q 40m 0x1000 build\bootloader\bootloader.bin 0x8000 build\partition_table\partition-table.bin 0x10000 build\ledc.bin  
or run 'idf.py -p (PORT) flash'
```

```
C:\Users\jbmas\work\esp_test\ledc>idf.py -p COM28 flash
```

6. Extra Session – app_main()

```
13 #include "esp_system.h"
14 #include "esp_spi_flash.h"
15
16 void app_main(void)
17 {
18     printf("Hello world!\n");
19
20     /* Print chip information */
21     esp_chip_info_t chip_info;
22     esp_chip_info(&chip_info);
23     printf("This is %s chip with %d CPU cores, WiFi%s%s, ",
24           CONFIG_IDF_TARGET,
25           chip_info.cores,
26           (chip_info.features & CHIP_FEATURE_BT) ? "/BT" : "",
27           (chip_info.features & CHIP_FEATURE_BLE) ? "/BLE" : "");
28
29     printf("silicon revision %d, ", chip_info.revision);
30
31     printf("%dMB %s flash\n", spi_flash_get_chip_size() / (1024 * 1024),
32           (chip_info.features & CHIP_FEATURE_EMB_FLASH) ? "embedded" : "external");
33
34     printf("Free heap: %d\n", esp_get_free_heap_size());
35
36     for (int i = 10; i >= 0; i--) {
37         printf("Restarting in %d seconds.\n", i);
38         vTaskDelay(1000 / portTICK_PERIOD_MS);
39     }
40     printf("Restarting now.\n");
41     fflush(stdout);
42     esp_restart();
43 }
```





```

529 static void main_task(void* args)
530 {
531 #if !CONFIG_FREERTOS_UNICORE
532     // Wait for FreeRTOS initialization to finish on APP CPU, before replacing its startup stack
533     while (port_xSchedulerRunning[1] == 0) {
534         <> Code; Issues 44 Pull requests 7 Actions Projects Wiki Security Insid
535     }
536 #endif
537     //Enable allocation in region where the startup stacks were located.
538     heap_caps_enable_nonos_stack_heaps();
539
540     // Now we have startup stack RAM available for heap, enable any DMA pool memory
541 #if CONFIG_SPIRAM_MALLOC_RESERVE_INTERNAL
542     esp_err_t r = esp_spiram_reserve_dma_pool(CONFIG_SPIRAM_MALLOC_RESERVE_INTERNAL);
543     if (r != ESP_OK) {
544         ESP_EARLY_LOGE(TAG, "Could not reserve internal/DMA pool (error/0x%x)", r);
545         abort();
546     }
547 #endif
548
549     //Initialize task wdt if configured to do so
550 #ifdef CONFIG_ESP_TASK_WDT_PANIC
551     ESP_ERROR_CHECK(esp_task_wdt_init(CONFIG_ESP_TASK_WDT_TIMEOUT_S, true));
552 #elif CONFIG_ESP_TASK_WDT
553     ESP_ERROR_CHECK(esp_task_wdt_init(CONFIG_ESP_TASK_WDT_TIMEOUT_S, false));
554 #endif
555 }
556
557 //components/esp32/cpu_start.c" 587 lines --89%--

```

```

570
571 // Now that the application is about to start, disable boot watchdog
572 #ifndef CONFIG_BOOTLOADER_WDT_DISABLE_IN_USER_CODE
573     wdt_hal_context_t rtc_wdt_ctx = {.inst = WDT_RWDT, .rwdt_dev = &RTCCNTL};
574     wdt_hal_write_protect_disable(&rtc_wdt_ctx);
575     wdt_hal_disable(&rtc_wdt_ctx);
576     wdt_hal_write_protect_enable(&rtc_wdt_ctx);
577 #endif
578 #ifdef CONFIG_BOOTLOADER_EFUSE_SECURE_VERSION_EMULATE
579     const esp_partition_t *efuse_partition = esp_partition_find_first(ESP_PARTITION_TYPE_DATA, ESP_PARTITION_SUBTYPE_DATA_EFUSE, NULL);
580     if (efuse_partition) {
581         esp_efuse_init(efuse_partition->address, efuse_partition->size);
582     }
583 #endif
584     app_main();
585     vTaskDelete(NULL);
586 }
587
588 .gitignore

```

```

321 void start_cpu0_default(void)
322 {
323     esp_err_t err;
324     esp_setup_syscall_table();
325     if (s_spiram_okay) {
326 #if CONFIG_SPIRAM_BOOT_INIT && (CONFIG_SPIRAM_USE_CAPS_ALLOC || CONFIG_SPIRAM_USE_MALLOC)
327         esp_err_t r=esp_spiram_add_to_heapalloc();
328         if (r != ESP_OK) {
329             ESP_EARLY_LOGE(TAG, "External RAM could not be added to heap!");
330             abort();
331         }
332     }
333 #if CONFIG_SPIRAM_USE_MALLOC
334     heap_caps_malloc_extmem_enable(CONFIG_SPIRAM_MALLOC_ALWAYSINTERNAL);
335 #endif
336 #endif
337 }
338
339 //Enable trace memory and immediately start trace.
340 #if CONFIG_ESP32_TRAX
341 #if CONFIG_ESP32_TRAX_TWOBANKS
342     trax_enable(TRAX_ENA_PRO_APP);
343 #else
344     trax_enable(TRAX_ENA_PRO);
345 #endif
346 #endif
347 #endif
348 trax_start_trace(TRAX_DOWNCOUNT_WORDS);
349 esp_clk_init();
350 esp_perip_clk_init();
351 intr_matrix_clear();
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000

```

```

455     .min_freq_mhz = xtal_freq,
456 };
457 esp_pm_configure(&cfg);
458 #endif //CONFIG_PM_DFS_INIT_AUTO
459 #endif //CONFIG_PM_ENABLE
460
461 #if CONFIG_ESP32_ENABLE_COREDUMP
462     esp_core_dump_init();
463 #endif
464
465 #if CONFIG_ESP32_WIFI_SW_COEXIST_ENABLE
466     esp_coex_adapter_register(&g_coex_adapter_funcs);
467     coex_pre_init();
468 #endif
469
470 portBASE_TYPE res = xTaskCreatePinnedToCore(&main_task, "main",
471                                             ESP_TASK_MAIN_STACK, NULL,
472                                             ESP_TASK_MAIN_PRIORITY, NULL, 0);
473 assert(res == pdTRUE);
474 ESP_LOGI(TAG, "Starting scheduler on PRO CPU.");
475 vTaskStartScheduler();
476 abort(); /* Only get to here if not enough free heap to start scheduler */
477 }

```



```

83 void start_cpu0(void) __attribute__((weak, alias("start_cpu0_default"))) __attribute__((noreturn));
84 void start_cpu0_default(void) IRAM_ATTR __attribute__((noreturn));
85 #if !CONFIG_FREERTOS_UNICORE
86 static void IRAM_ATTR call_start_cpu1(void) __attribute__((noreturn));
87 void start_cpu1(void) __attribute__((weak, alias("start_cpu1_default"))) __attribute__((noreturn));
88 void start_cpu1_default(void) IRAM_ATTR __attribute__((noreturn));
89 static bool app_cpu_started = false;
90 #endif // !CONFIG_FREERTOS_UNICORE
91
92 static void do_global_ctors(void);
93 static void main_task(void* args);
94 extern void app_main(void);
95 extern esp_err_t esp_thread_init(void);
96
97 extern int _bss_start;
98 extern int _bss_end;
99
100 // components/esp32/cpu_start.c" 587 lines --14%--

```

```

128 void IRAM_ATTR call_start_cpu0(void)
129 {
130 #if CONFIG_FREERTOS_UNICORE
131     RESET_REASON rst_reas[1];
132 #else
133     RESET_REASON rst_reas[2];
134 #endif
135
136     bootloader_init_mem();
137
138     // Move exception vectors to IRAM
139     cpu_hal_set_vecbase(&_init_start);
140
141     rst_reas[0] = rtc_get_reset_reason(0);
142
143 #if !CONFIG_FREERTOS_UNICORE
144     rst_reas[1] = rtc_get_reset_reason(1);
145 #endif
146
147     // from panic handler we can be reset by RWD or TGOWDT
148     if (rst_reas[0] == RTCWDT_SYS_RESET || rst_reas[0] == TGOWDT_SYS_RESET
149 #if !CONFIG_FREERTOS_UNICORE
150         || rst_reas[1] == RTCWDT_SYS_RESET || rst_reas[1] == TGOWDT_SYS_RESET
151 #endif
152     ) {

```

```

265     memory also used by the ROM. Starting the app cpu will let its ROM initialize that memory,
266     corrupting those linked lists. Initializing the allocator *after* the app cpu has booted
267     works around this problem.
268     With SPI RAM enabled, there's a second reason: half of the SPI RAM will be managed by the
269     app CPU, and when that is not up yet, the memory will be inaccessible and heap_caps_init may
270     fail initializing it properly. */
271     heap_caps_init();
272
273     ESP_EARLY_LOGI(TAG, "Pro cpu start user code");
274     start_cpu0();
275 }

```

```

1 /* Default entry point: */ https://github.com/espressif/esp-iot-solution
2 ENTRY(call_start_cpu0);
3
4 SECTIONS
5 {
6     /* RTC fast memory holds RTC wake stub code,
7        including from any source file named rtc_wake_stub*.c
8     */
9     .rtc.text :
10     {
11         = ALIGN(4);
12
13         mapping[rtc_text]
14
15         *rtc_wake_stub*.*(.literal .text .literal.* .text.*)
16         _rtc_text_end = ABSOLUTE(.);
17     } > rtc_iram_seg
18
19     /* costaud committed 40cec13 on Sep 4, 2019 ...
20        This section is required to skip rtc.text area because rtc_iram_seg and
21        rtc_data_seg are reflect the same address space on different buses.
22     */
23     .rtc.dummy :
24     {
25         _rtc_dummy_start = ABSOLUTE(.);
26         _rtc_fast_start = ABSOLUTE(.);
27         = SIZEOF(.rtc.text);
28         _rtc_dummy_end = ABSOLUTE(.);
29     } > rtc_data_seg
30
31     /* This section located in RTC FAST Memory area.
32     ./components/esp32/ld/esp32.project.ld.in" 378 lines --0%--

```

ENTRY
어플리케이션의
시작포인트

감사합니다.