메이커 스페이스
G·캠프

ESP32

온라인 워크숍 초급과정

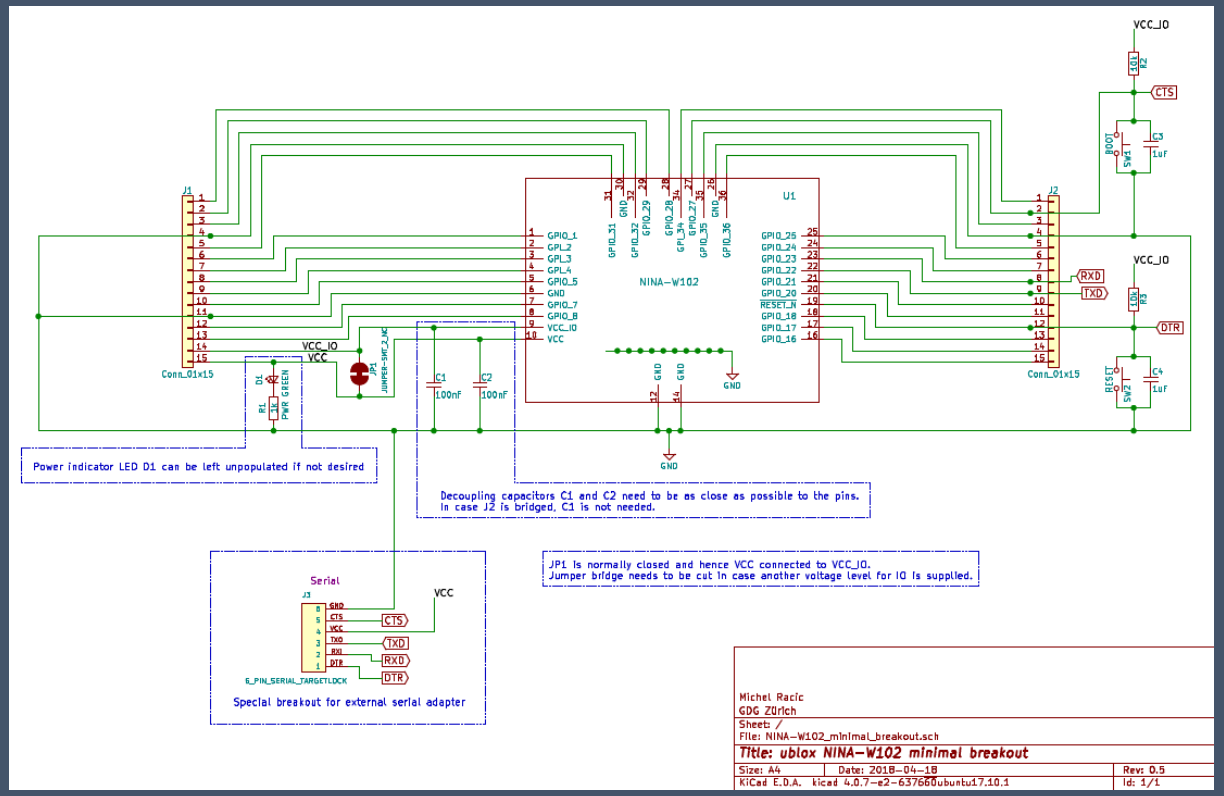
과정 목차

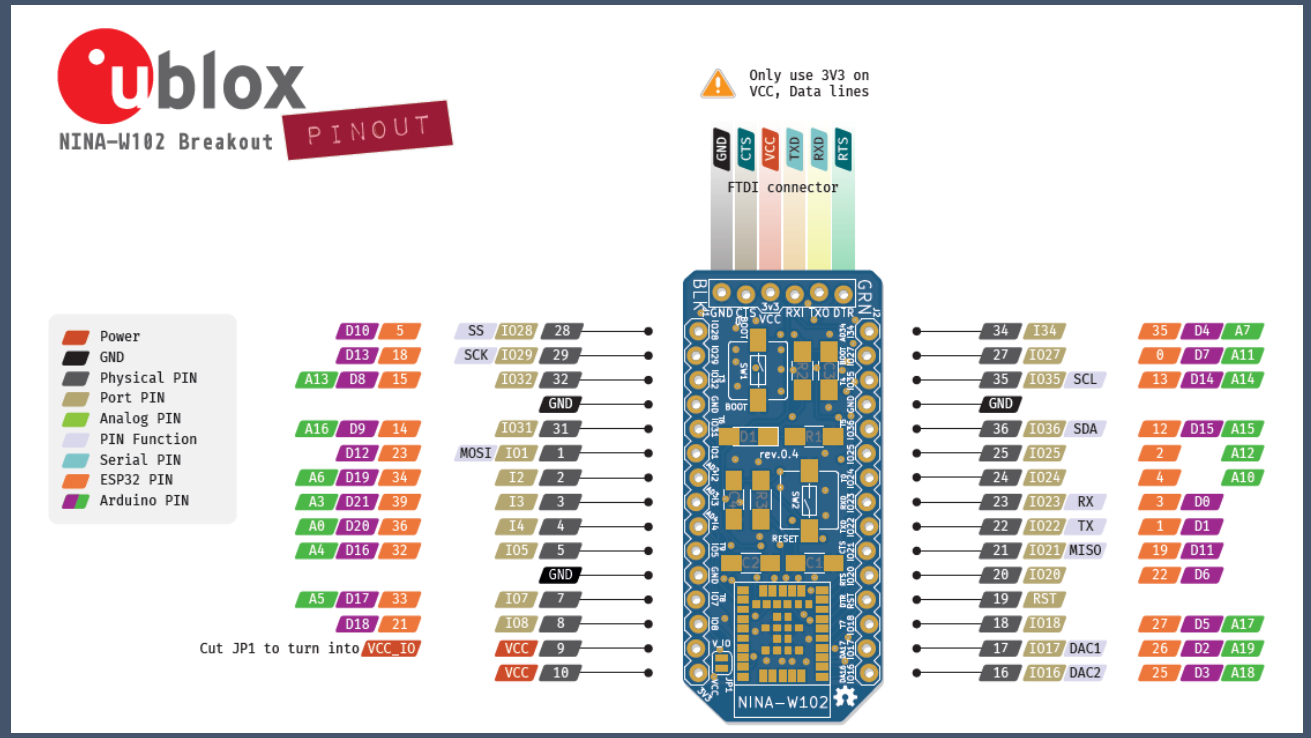
❖ 1일차
> 시작하기 전에
> ESP32 소개
> 개발환경 설정 (Window Version)
> ESP-IDF 설치
> 빌드환경 설정
> Edit Tool 설정
> Extra Session : app_main

❖ 2일차
> GPIO & GPIO Interrupt
  - Output : LED 실습
  - Input : Button 실습
>UART
  - UART Echo 실습
>Timer & Timer Interrupt
  - 사용자용 Timer 작성

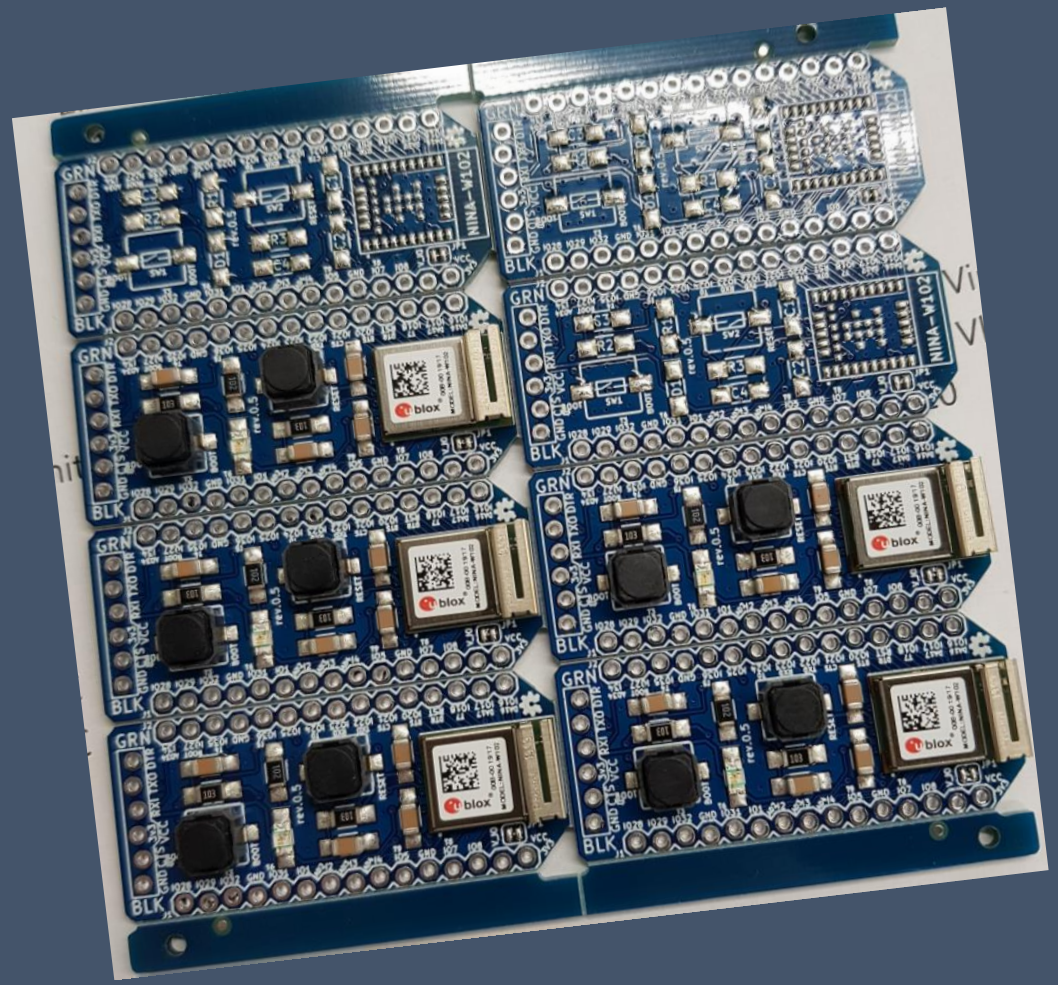
❖ 3일차
> Analog Input
  - 가변저항 읽기 실습
> PWM
  - LED 디밍 실습 (HS mode/ LS mode 체크)
> I2C
  - 다른 프로젝트 소스코드 포팅, HTS221 온습도 센서 실습

0. 시작하기 전에

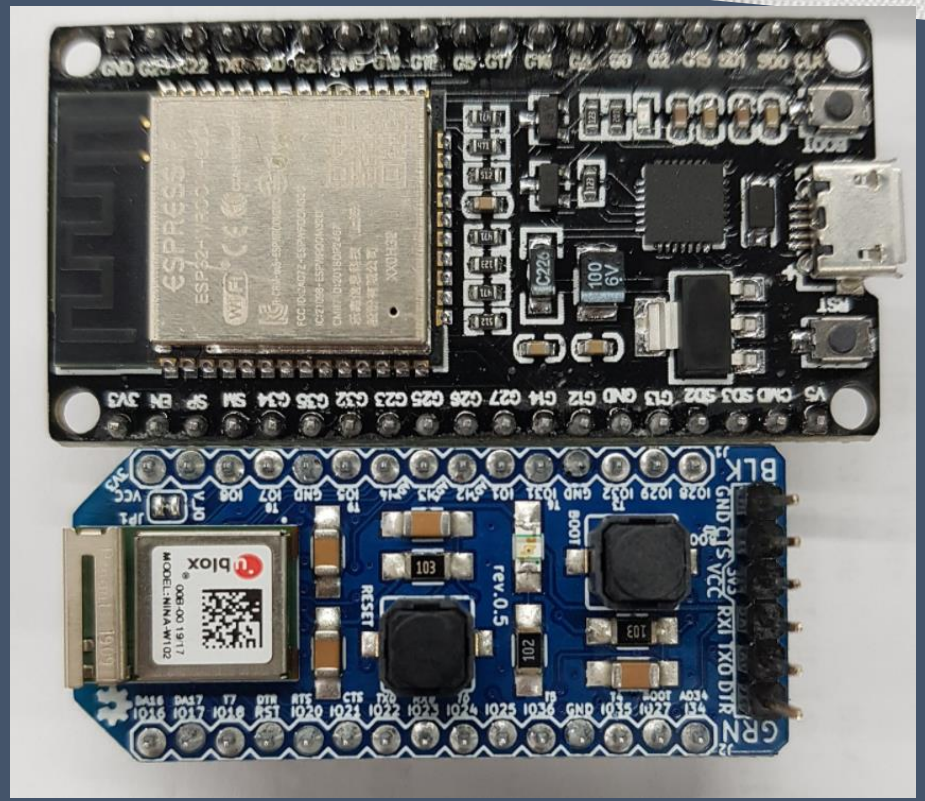




0. 시작하기 전에

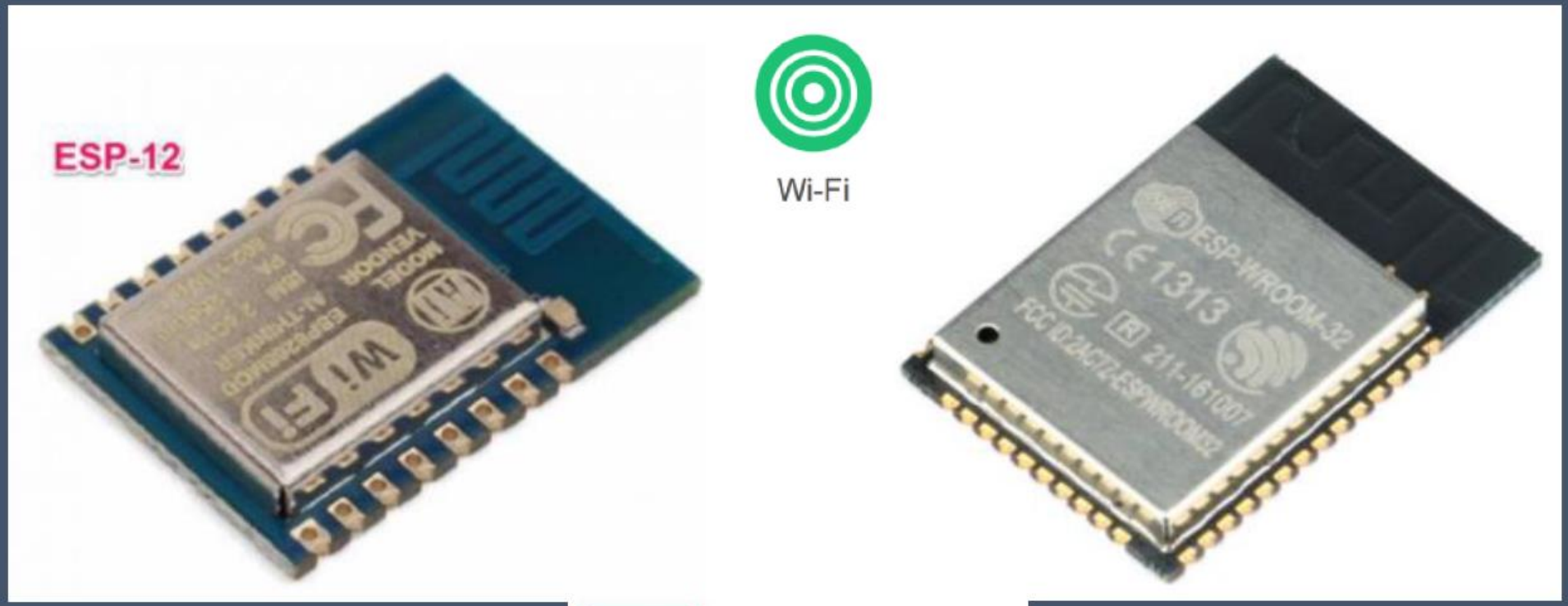



NINA-W102 minimal breakout
Rev.0.5
Thanks for Michel Racic
GCamp www.g.camp
CodeZoo www.codezoo.co.kr



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/news/bug-bounty
https://www.espressif.com/en/news/The_ESP32_Security_Bug_Bounty_Is_Still_On

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) (80, 160, **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)

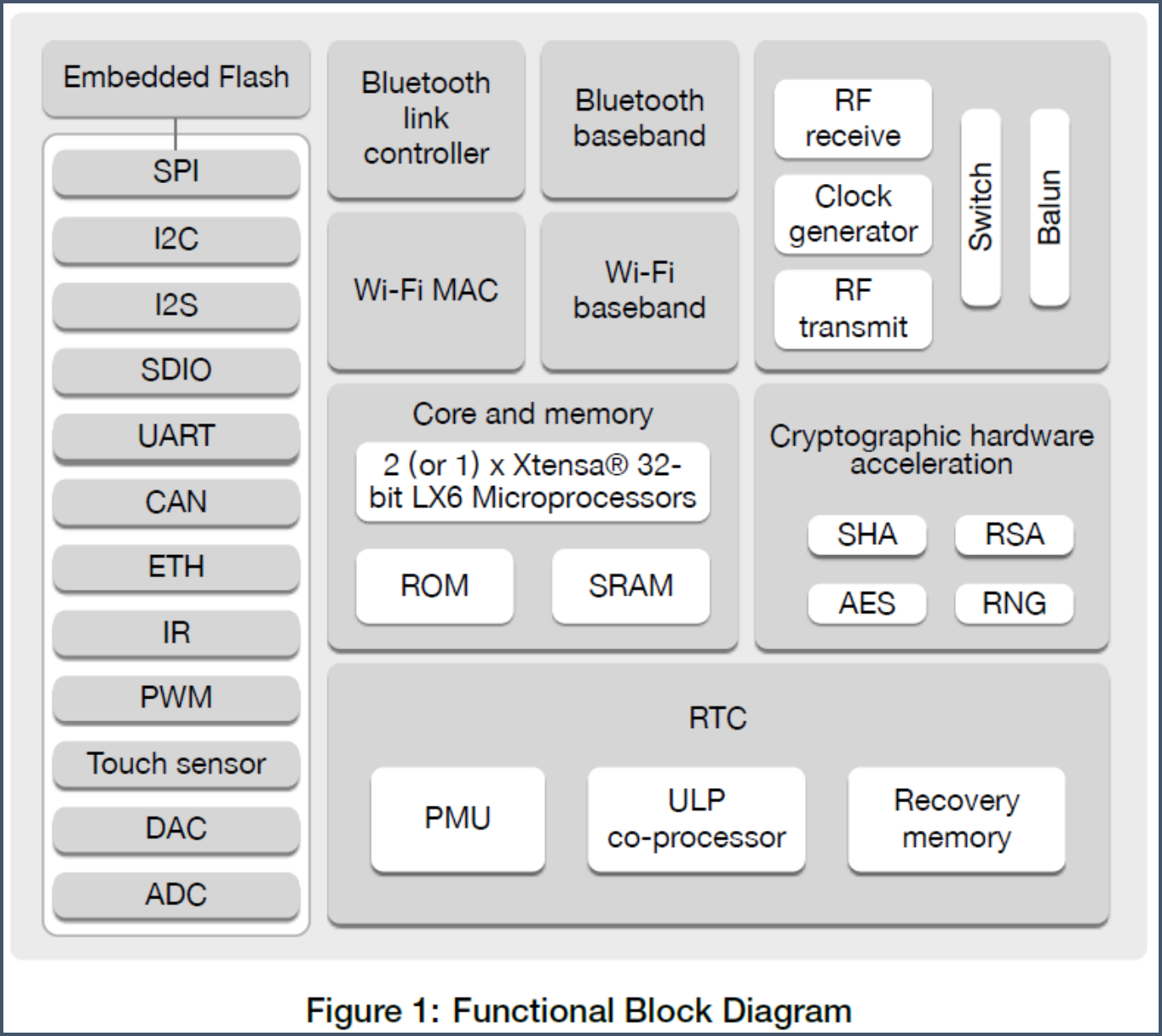
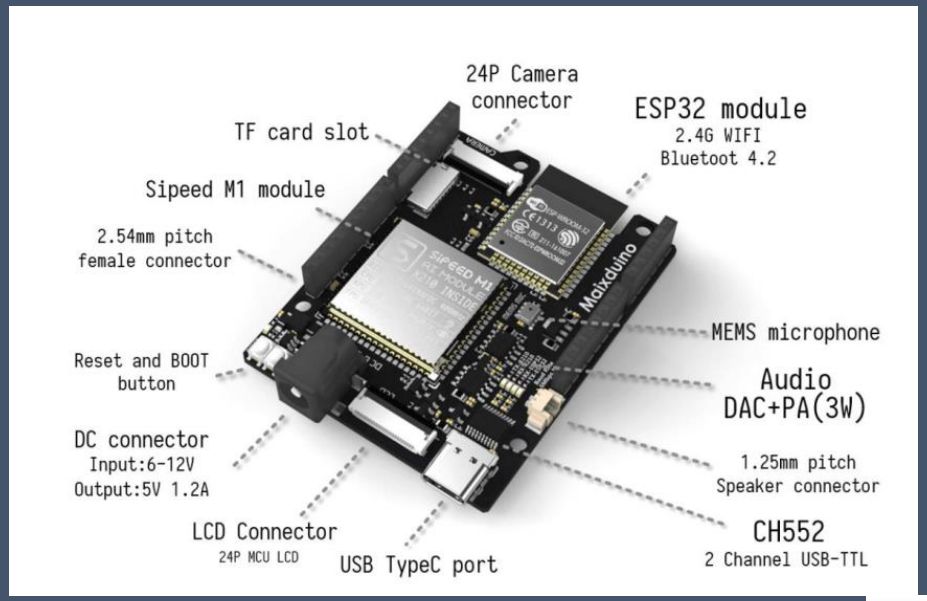
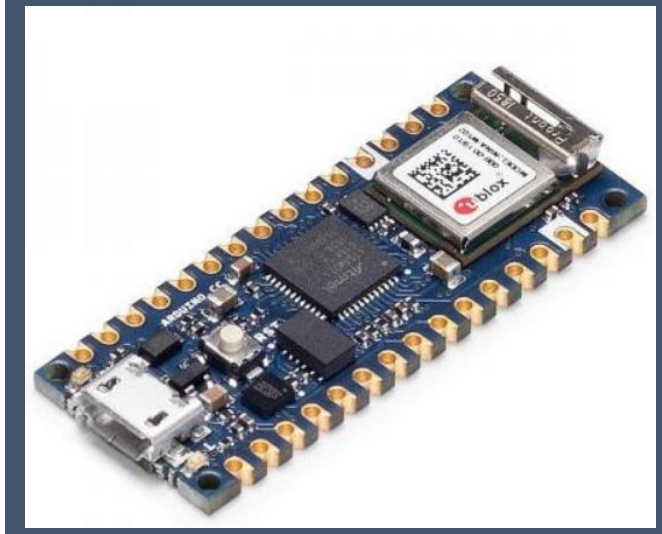


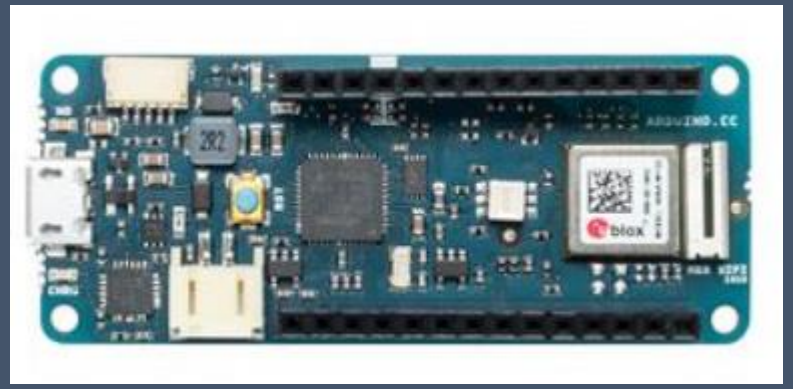
Figure 1: Functional Block Diagram

ESP32 Commercial Products



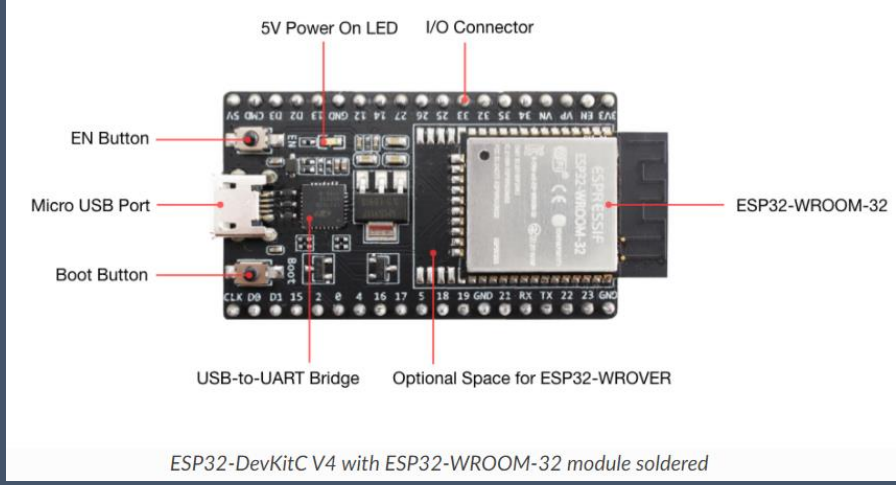
ESP32 Commercial Products





2. ESP32 개발환경 설정

References



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

- ✓ 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

https://docs.espressif.com/projects/esp-idf/en/latest/esp32/hw-reference/esp32/get-started-devkitc.html

ESP-IDF Programming Guide

ESPRESSIF

ESP32

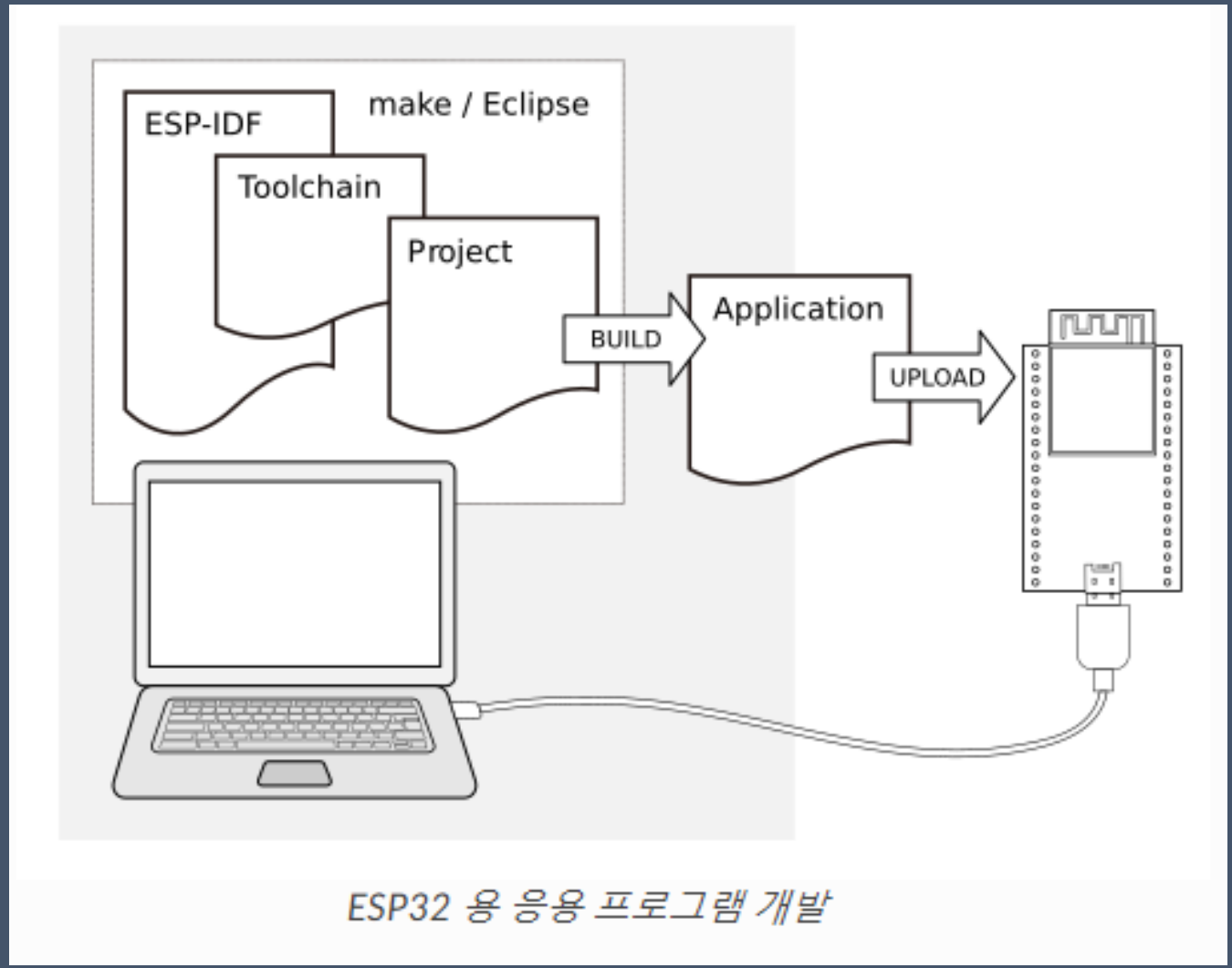
개발을 위한 준비 *ESP-IDF (Espressif IoT Development Framework)*

✓ 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

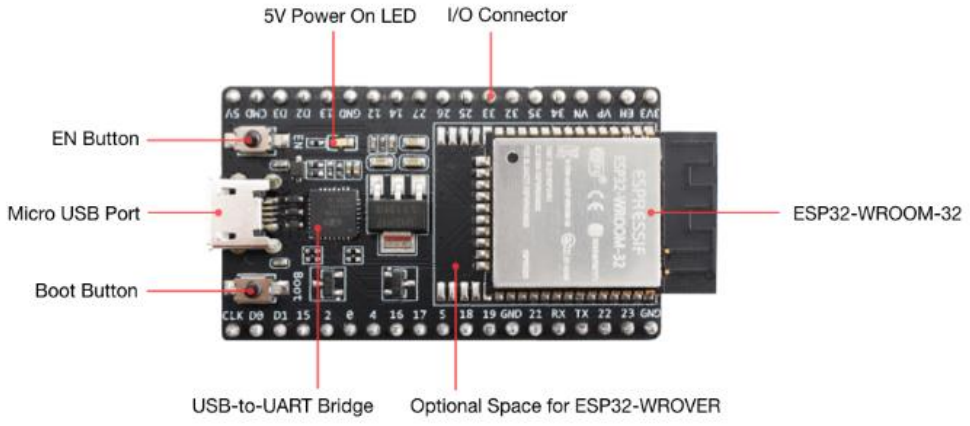


ESP32 용 응용 프로그램 개발

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.

전체 개발 흐름

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 제공해야합니다 . 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.

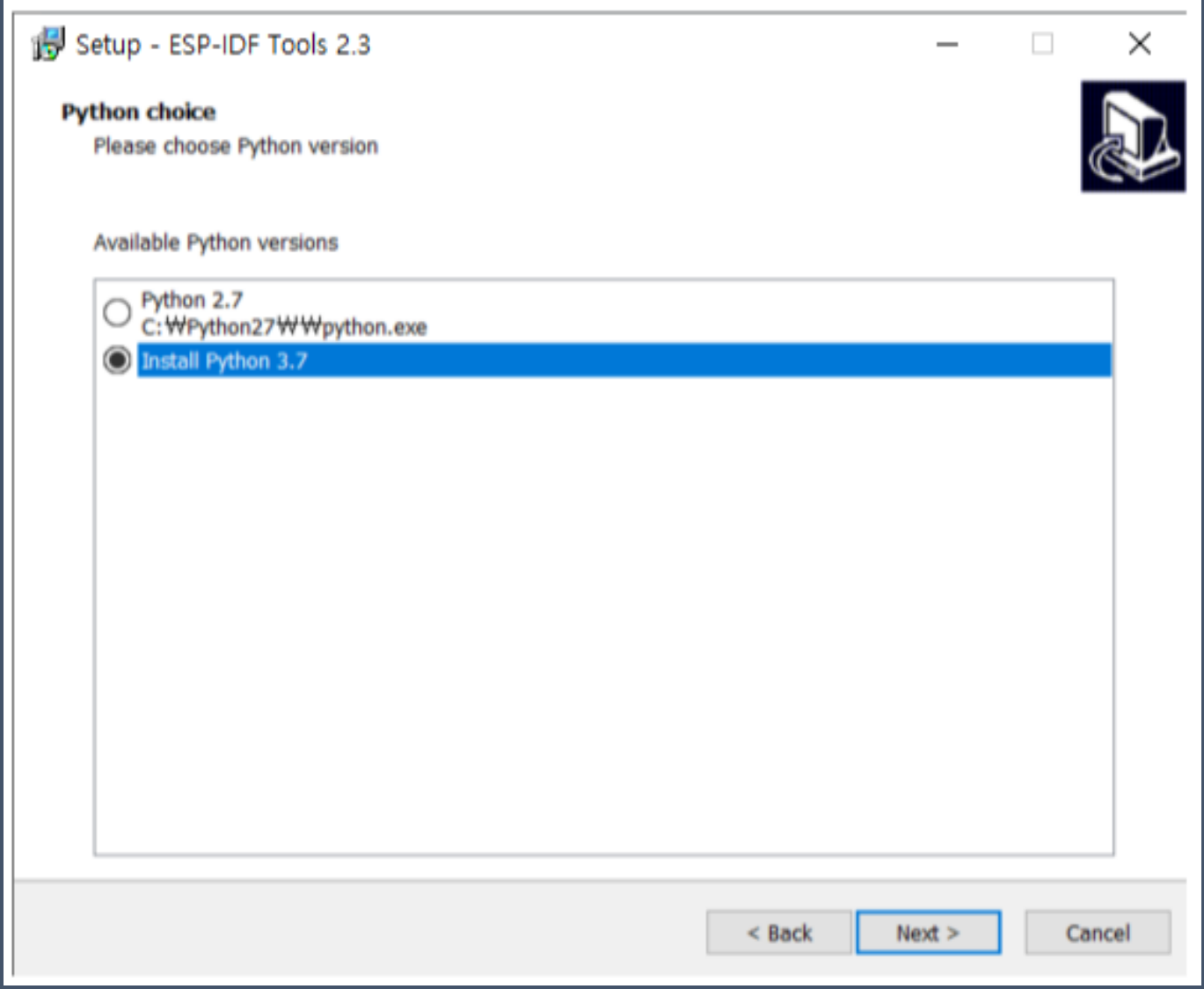
ESP-IDF 도구 설치 프로그램

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

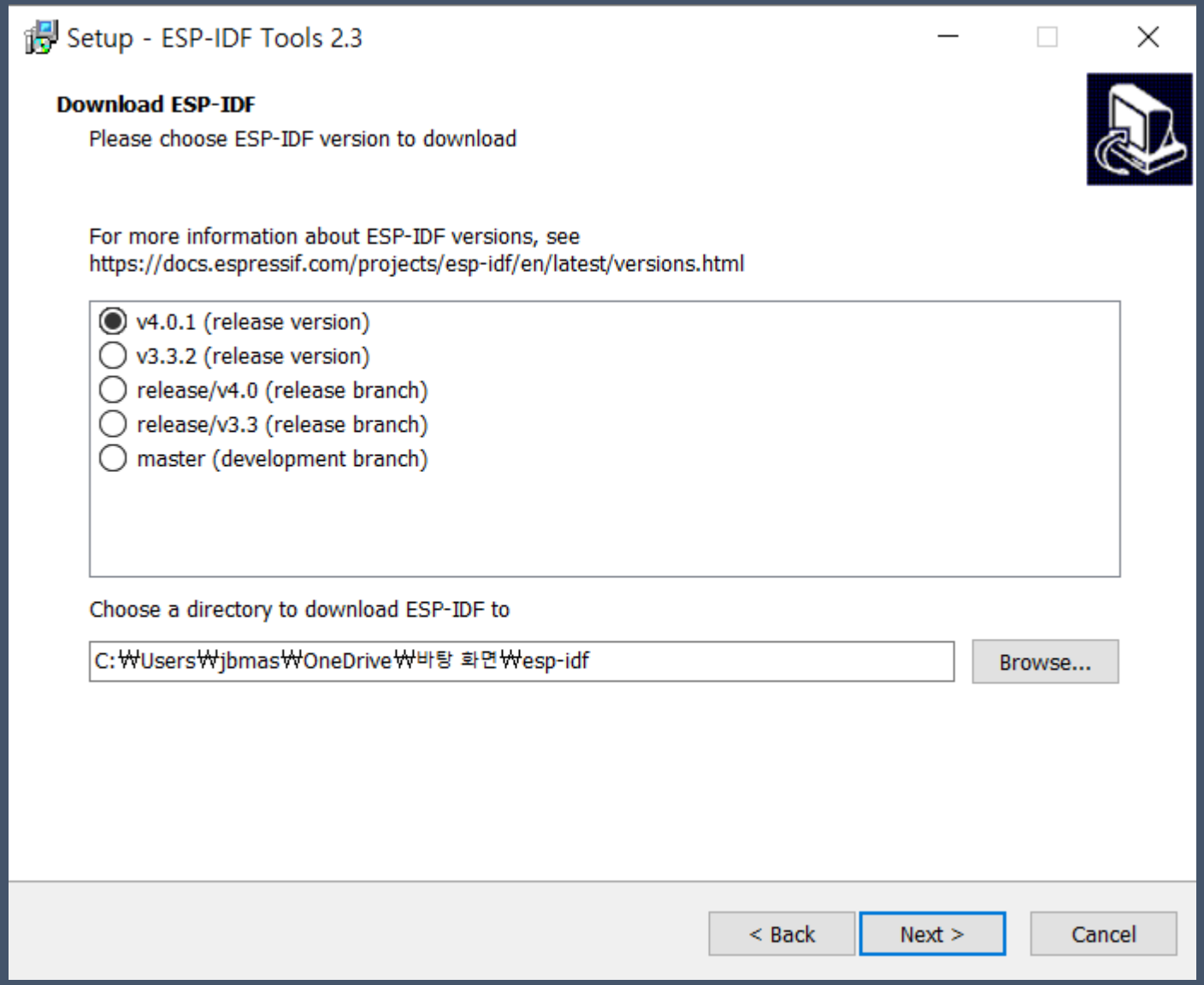
크로스 컴파일러, OpenOCD, cmake 및 Ninja 빌드 도구가 포함됩니다. 설치 프로그램은 Python 3.7 및 Git For Windows 용 설치 프로그램이 컴퓨터에 아직 설치되어 있지 않은 경우 다운로드하여 실행할 수 있습니다 .

개발도구 설치



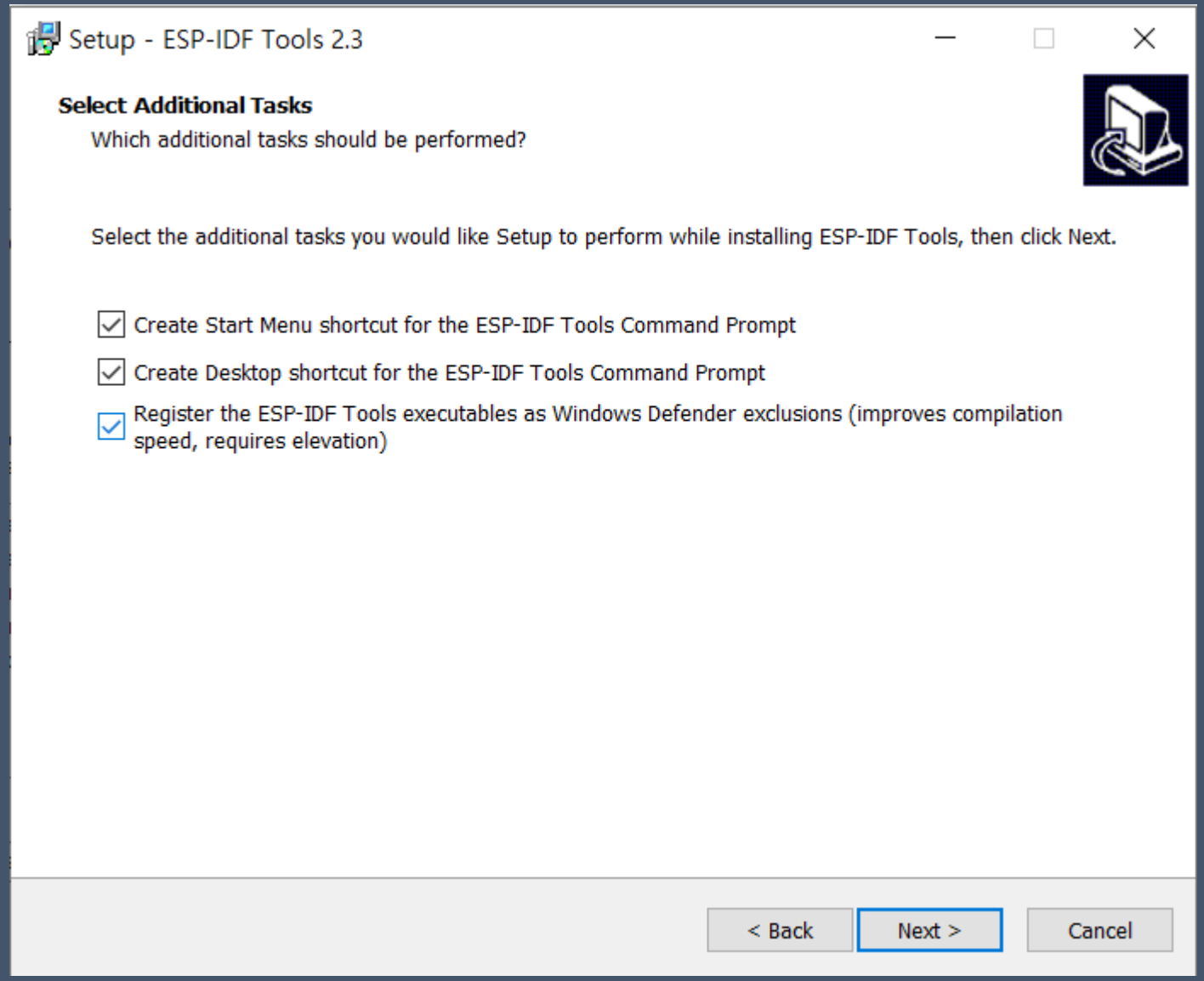
✓ 기존에 구축한 시스템과 별도로 설치하는 것이 유리합니다. 특히 파이썬 버전을 맞춰두고 빌드하는 다른 시스템이 있으면 필히 Intall 지정해서 설치하시기 바랍니다.

개발도구 설치



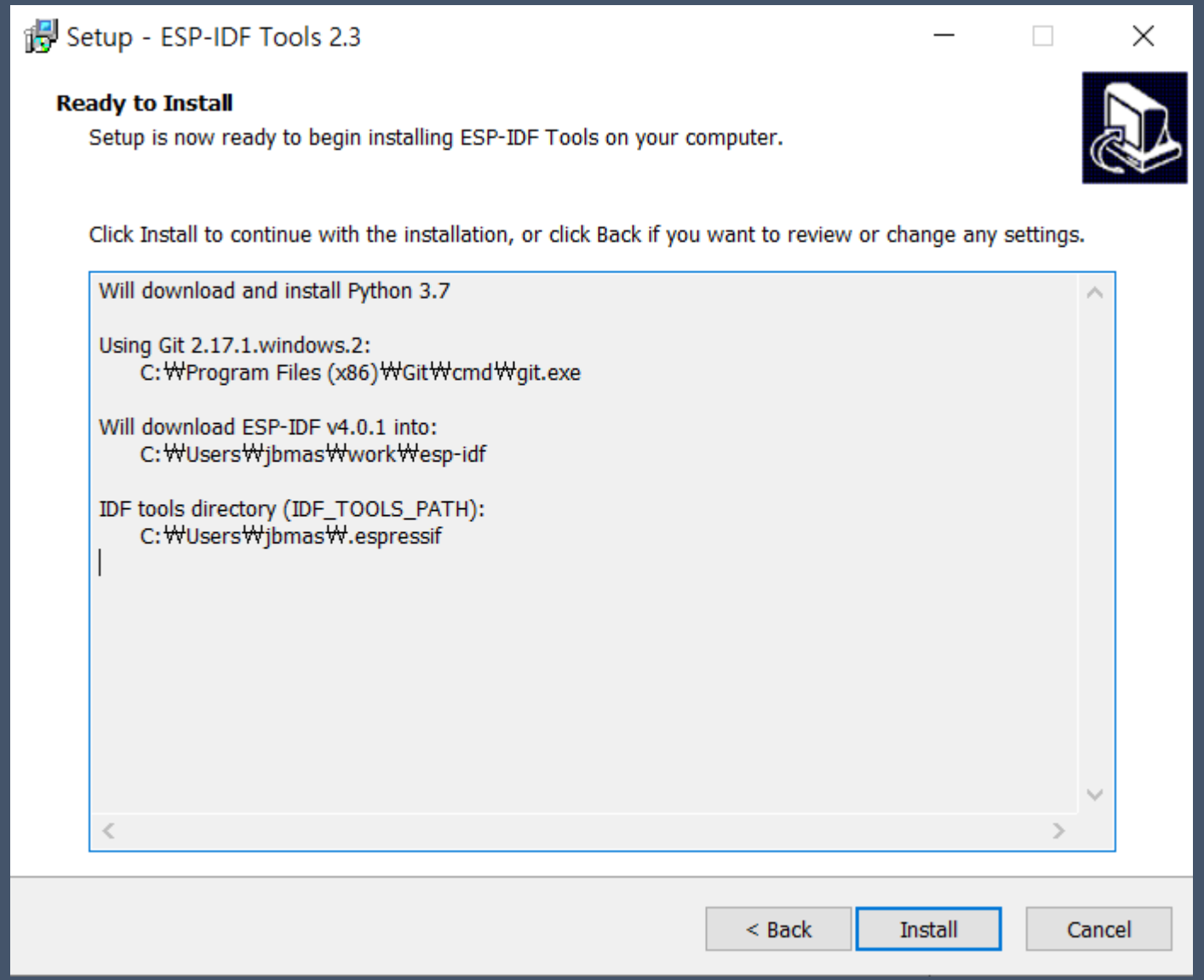
✓ V4.0.1 (릴리즈버전 사용)
다운로드 받을 디렉토리 주소 체크
(한글 디렉토리X)

개발도구 설치

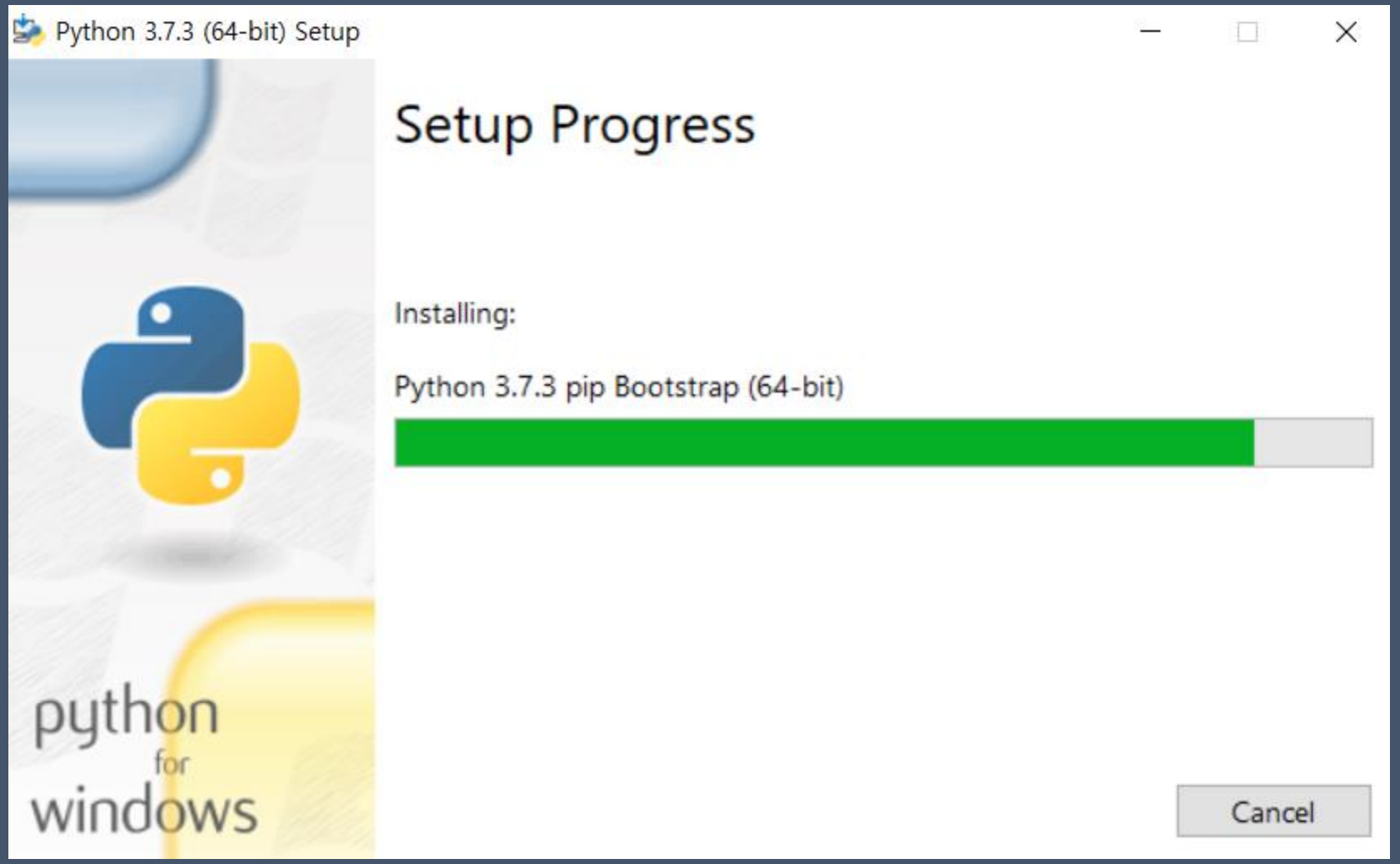


개발도구 설치

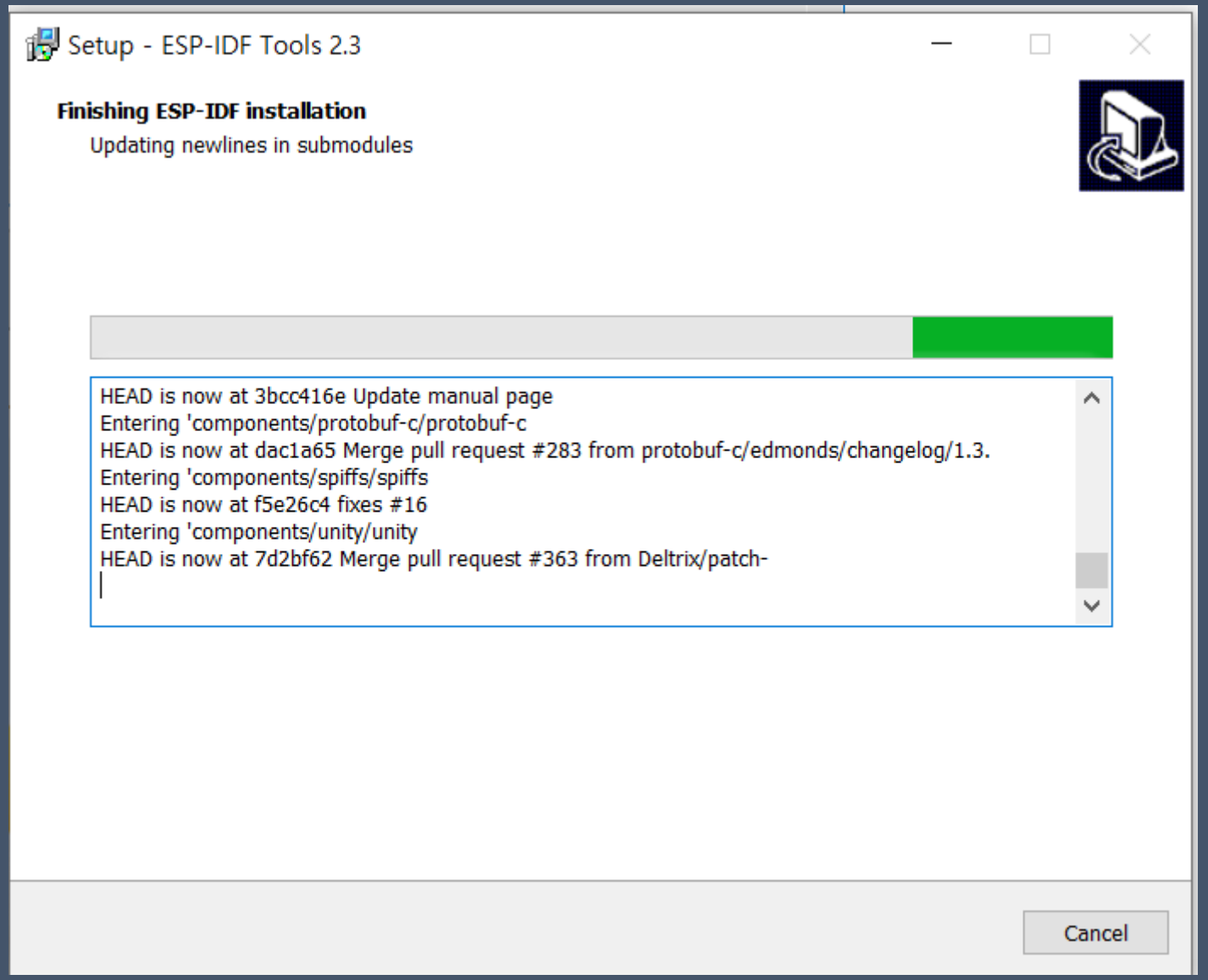
설치 전 최종 확인



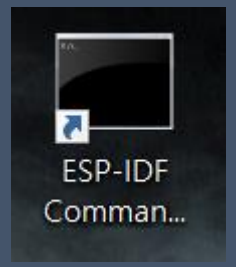
개발도구 설치


Python 3.7.3 (64-bit) Setup
Setup Progress
Installing:
Python 3.7.3 pip Bootstrap (64-bit)
Cancel
python
for
windows

개발도구 설치


Setup - ESP-IDF Tools 2.3
Finishing ESP-IDF installation
Updating newlines in submodules
HEAD is now at 3bcc416e Update manual page
Entering 'components/protobuf-c/protobuf-c
HEAD is now at dac1a65 Merge pull request #283 from protobuf-c/edmonds/changelog/1.3.
Entering 'components/spiffs/spiffs
HEAD is now at f5e26c4 fixes #16
Entering 'components/unity/unity
HEAD is now at 7d2bf62 Merge pull request #363 from Deltrix/patch-
Cancel

개발도구 설치



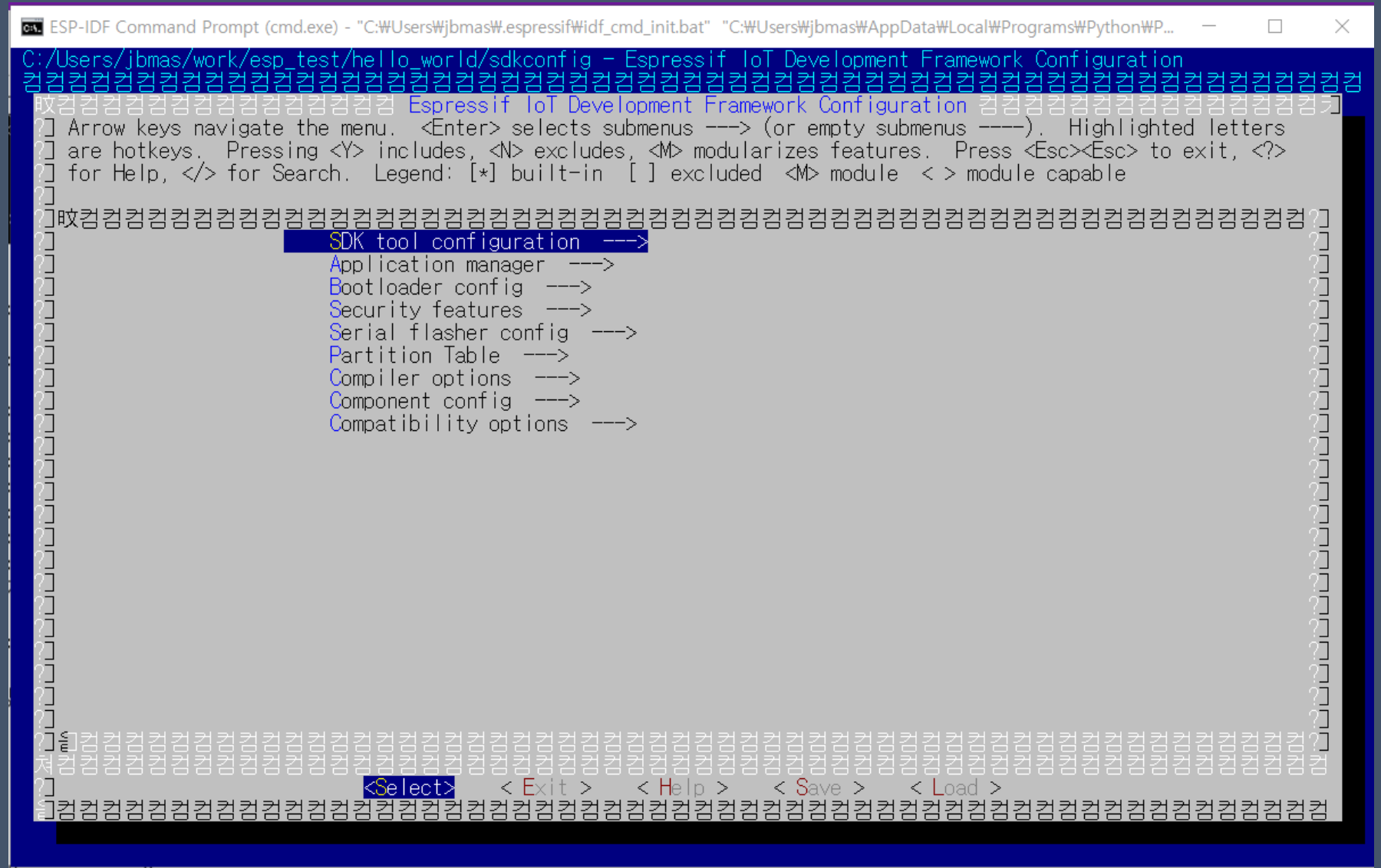
단축 아이콘 클릭

```
ESP-IDF Command Prompt (cmd.exe) - "C:₩Users₩jbmas₩.espressif₩idf_cmd_init.bat"  "C:₩Users₩jbmas₩AppData₩Local₩Programs₩Python₩Pyt...
Using Python in C:₩Users₩jbmas₩AppData₩Local₩Programs₩Python₩Python37₩
Python 3.7.3
Using Git in C:₩Program Files (x86)₩Git₩cmd₩
git version 2.17.1.windows.2
Setting IDF_PATH: C:₩Users₩jbmas₩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₩jbmas₩.espressif₩tools₩xtensa-esp32-elf₩esp-2019r2-8.2.0₩xtensa-esp32-elf₩bin
    C:₩Users₩jbmas₩.espressif₩tools₩esp32ulp-elf₩2.28.51.20170517₩esp32ulp-elf-binutils₩bin
    C:₩Users₩jbmas₩.espressif₩tools₩cmake₩3.13.4₩bin
    C:₩Users₩jbmas₩.espressif₩tools₩openocd-esp32₩v0.10.0-esp32-20190313₩openocd-esp32₩bin
    C:₩Users₩jbmas₩.espressif₩tools₩mconf₩v4.6.0.0-idf-20190628₩
    C:₩Users₩jbmas₩.espressif₩tools₩ninja₩1.9.0₩
    C:₩Users₩jbmas₩.espressif₩tools₩idf-exe₩1.0.1₩
    C:₩Users₩jbmas₩.espressif₩tools₩ccache₩3.7₩
    C:₩Users₩jbmas₩.espressif₩python_env₩idf4.0_py3.7_env₩Scripts
    C:₩Users₩jbmas₩work₩esp-idf₩tools

Checking if Python packages are up to date...
Python requirements from C:₩Users₩jbmas₩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₩jbmas₩work₩esp-idf>
```

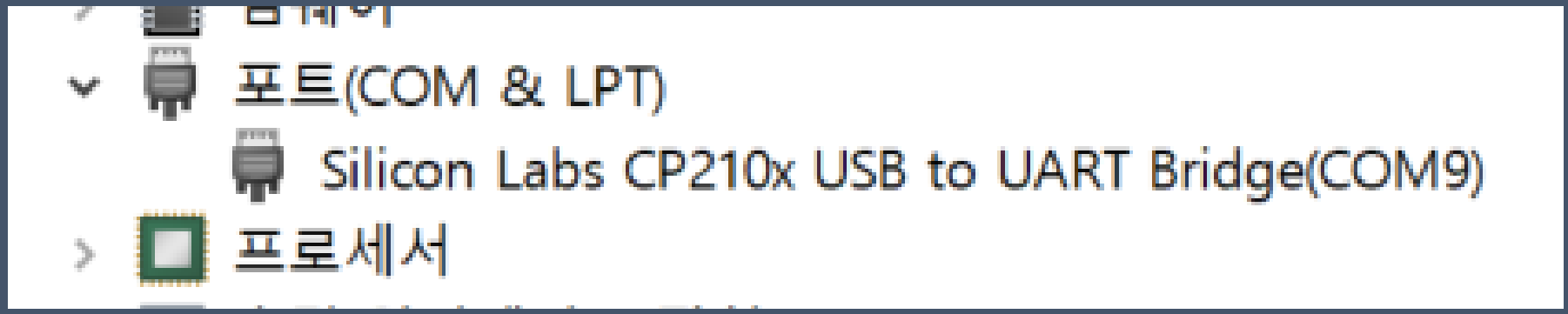
4. 빌드환경 설정

```
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>
```

₩>idf.py menuconfig



장치관리자 --> 포트 확인



₩>idf.py build

₩>idf.py -p COM9 -b 115200 flash

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

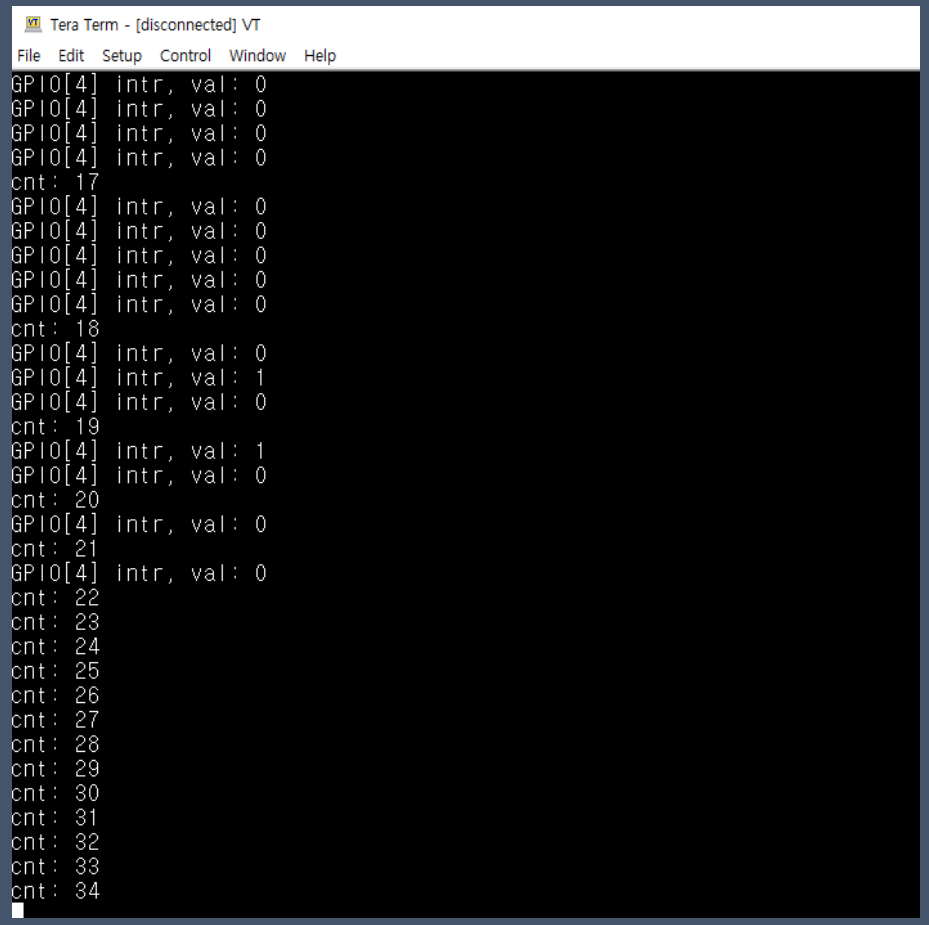
```
Detecting chip type... ESP32
Chip is ESP32D0WDQ6 (revision 1)
Features: WiFi, BT, Dual Core, 240MHz, VRef calibration in efuse, Coding Scheme None
Crystal is 40MHz
MAC: 3c:71:bf:4b:e7:7c
Uploading stub...
Running stub...
Stub running...
Configuring flash size...
Compressed 3072 bytes to 103...
Wrote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

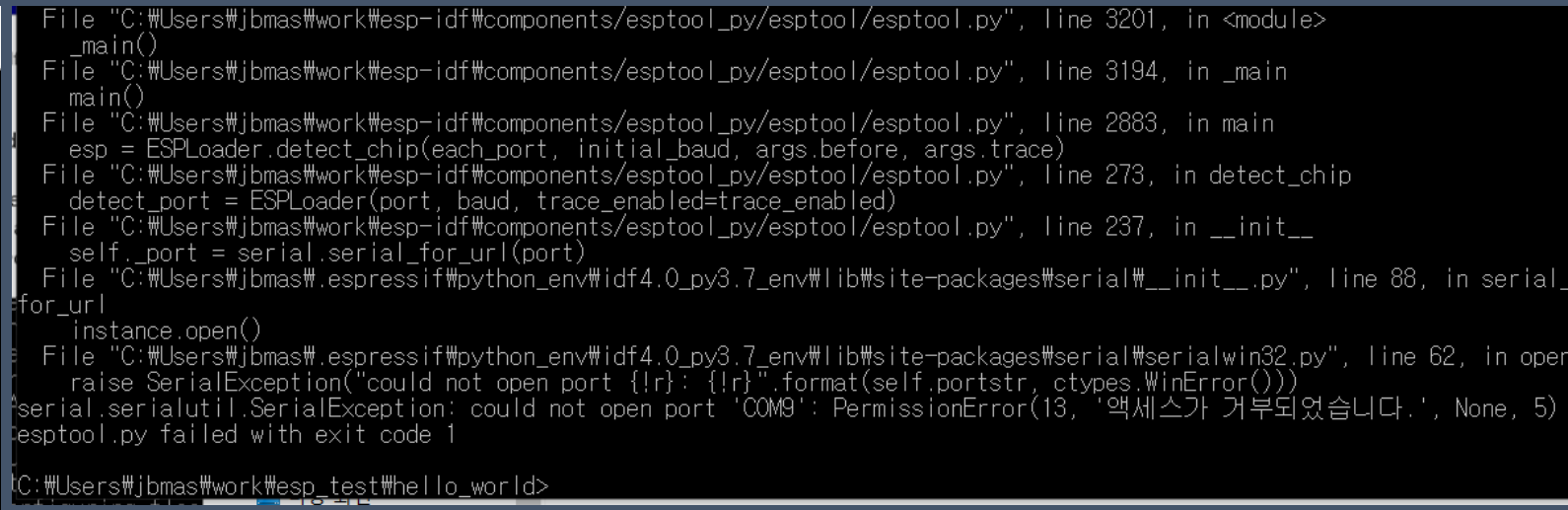
<펌웨어 Flash 실패>
esp-idf 개발 보드를 수동으로 재설정하려면 **부팅** 버튼 (GPIO0)을 누른 상태에서 **EN** 버튼 (CHIP_PU)을 누릅니다 .

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(20
48k). 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...
```





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

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(20
48k). 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...
```

ESP32 DOWNLOAD TOOL V3.8.5

SPIDownload | HSPIDownload | GPIOConfig

☑	ld₩build₩partition_table₩partition-table.bin	... @	0x8000
☑	C:₩Users₩jbmas₩work₩gcamp₩get-started	... @	0x1000
☑	C:₩Users₩jbmas₩work₩gcamp₩get-started	... @	0x10000
☐		... @	
☐		... @	
☐		... @	
☐		... @	
☐		... @	

SpiFlashConfig

SPI SPEED: ◉ 40MHz ○ 26.7MHz ○ 20MHz ○ 80MHz

CombineBin | Default

SPI MODE: ○ QIO ○ QOUT ◉ DIO ○ DOUT ○ FASTRD

FLASH SIZE: ○ 8Mbit ○ 16Mbit ◉ 32Mbit ○ 64Mbit ○ 128Mbit

☐ SpiAutoSet ☐ DoNotChgBin

☐ LOCK SETTINGS

DETECTED INFO

flash vendor:
20h : N/A
flash devID:
4016h
QUAD;32Mbit
crystal:
40 Mhz

Download Panel 1

FINISH 完成

AP: CC50E3925965 STA: CC50E3925964
BT: CC50E3925966 ETHERNET: CC50E3925967

START | STOP | ERASE

COM: COM28

BAUD: 921600

www.CodeZoo.co.kr

https://www.espressif.com/en/support/download/other-tools

```
ninja: no work to do.
Executing action: flash
Running esptool.py in directory c:\Users\jbmas\work\gcamp\get-started\hello_world\build
Executing "C:\Users\jbmas\.espressif\python_env\idf4.0_py3.7_env\Scripts\python.exe C:\Users\jbmas\work\esp-idf\components/esptool_py/esptool/esptool.
py -p COM28 -b 460800 --before default_reset --after hard_reset write_flash @flash_project_args"...
esptool.py -p COM28 -b 460800 --before default_reset --after hard_reset write_flash --flash_mode dio --flash_freq 40m --flash_size 2MB 0x8000 partitio
n_table/partition-table.bin 0x1000 bootloader/bootloader.bin 0x10000 hello-world.bin
```

```
45  CONFIG_ESPTOOLPY_FLASHFREQ="40m"
46  # CONFIG_ESPTOOLPY_FLASHSIZE_1MB is not set
47  CONFIG_ESPTOOLPY_FLASHSIZE_2MB=y
48  # CONFIG_ESPTOOLPY_FLASHSIZE_4MB is not set
49  # CONFIG_ESPTOOLPY_FLASHSIZE_8MB is not set
```

```
46  # CONFIG_ESPTOOLPY_FLASHSIZE_1MB is not set
47  # CONFIG_ESPTOOLPY_FLASHSIZE_2MB is not set
48  CONFIG_ESPTOOLPY_FLASHSIZE_4MB=y
49  # CONFIG_ESPTOOLPY_FLASHSIZE_8MB is not set
50  # CONFIG_ESPTOOLPY_FLASHSIZE_16MB is not set
```

```
I (47) boot: SPI Speed      : 40MHz
I (51) boot: SPI Mode       : DIO
I (55) boot: SPI Flash Size : 4MB
I (59) boot: Partition Table:
I (62) boot: ## Label            Usage          Type ST Offset   Length
I (70) boot:  0 nvs              WiFi data        01 02 00009000 00006000
I (77) boot:  1 phy_init         RF data          01 01 0000f000 00001000
I (85) boot:  2 factory          factory app      00 00 00010000 00100000
I (92) boot: End of partition table
I (96) boot_comm: chip revision: 1, min. application chip revision: 0
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
I (294) 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, 4MB external flash
Restarting in 10 seconds...
Restarting in 9 seconds...
Restarting in 8 seconds...
Restarting in 7 seconds...
```

Clear !!!

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.

File Edit Selection View Go Run Terminal Help — gpio_example_main.c - esp-idf - Visual Studio Code

EXPLORER

OPEN EDITORS
- gpio_example_main.c examples\p... 4

ESP-IDF
- build_system
- common_components
- ethernet
- get-started
- mesh
- peripherals
  - adc
  - adc2
  - can
  - gpio
    - main
      - CMakeLists.txt
      - component.mk
      - gpio_example_main.c 4
    - CMakeLists.txt
    - Makefile
    - README.md
  - i2c
  - i2s
  - i2s_adc_dac
  - ledc
  - mcpwm
  - pcnt
  - rmt_nec_tx_rx
  - rmt_tx
  - sdio
  - sigmadelta
  - spi_master
  - spi_slave
  - timer_group
  - touch_pad_interrupt
  - touch_pad_read
  - uart
  - README.md
- protocols
- provisioning
- security

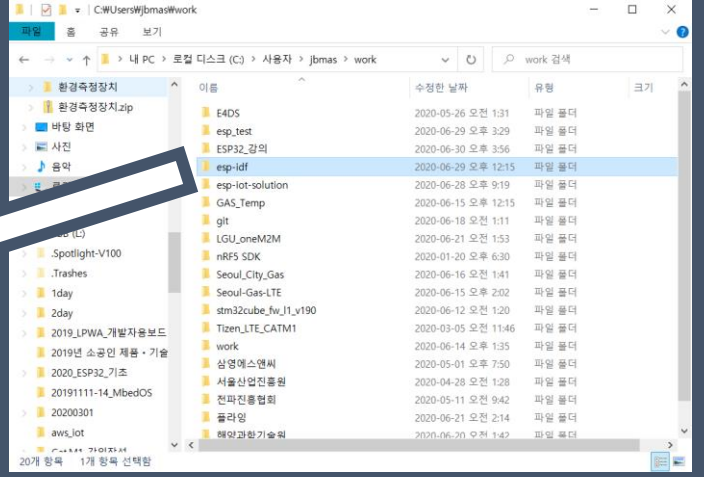
OUTLINE
TIMELINE
AL OUTLINE

gpio_example_main.c

examples > peripherals > gpio > main > C gpio_example_main.c > ...

```c
/**
 * Brief:
 * This test code shows how to configure gpio and how to use gpio interrupt.
 *
 * GPIO status:
 * GPIO18: output
 * GPIO19: output
 * GPIO4:  input, pulled up, interrupt from rising edge and falling edge
 * GPIO5:  input, pulled up, interrupt from rising edge.
 *
 * Test:
 * Connect GPIO18 with GPIO4
 * Connect GPIO19 with GPIO5
 * Generate pulses on GPIO18/19, that triggers interrupt on GPIO4/5
 *
 */

#define GPIO_OUTPUT_IO_0    18
#define GPIO_OUTPUT_IO_1    19
#define GPIO_OUTPUT_PIN_SEL  ((1ULL<<GPIO_OUTPUT_IO_0) | (1ULL<<GPIO_OUTPUT_IO_1))
#define GPIO_INPUT_IO_0     4
#define GPIO_INPUT_IO_1     5
#define GPIO_INPUT_PIN_SEL  ((1ULL<<GPIO_INPUT_IO_0) | (1ULL<<GPIO_INPUT_IO_1))
#define ESP_INTR_FLAG_DEFAULT 0

static xQueueHandle gpio_evt_queue = NULL;

static void IRAM_ATTR gpio_isr_handler(void* arg)
{
    uint32_t gpio_num = (uint32_t) arg;
    xQueueSendFromISR(gpio_evt_queue, &gpio_num, NULL);
}

static void gpio_task_example(void* arg)
{
    uint32_t io_num;
    for(;;) {
        if(xQueueReceive(gpio_evt_queue, &io_num, portMAX_DELAY)) {
            printf("GPIO[%d] intr, val: %d\n", io_num, gpio_get_level(io_num));
        }
    }
}

void app_main()
{
    gpio_config_t io_conf;
    //disable interrupt
    io_conf.intr_type = GPIO_PIN_INTR_DISABLE;
    //set as output mode
```

4c81978a* ⊗ 4 ⚠ 0 CMake: [Debug]: 준비 선택된 키트가 없음 빌드 [all] Ln 1, Col 1 Spaces: 4 UTF-8 CRLF C Win32



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

File Edit Selection View Go Run Terminal Help — ledc_example_main.c - ledc - Visual Studio Code

EXPLORER

OPEN EDITORS
- ledc_example_main.c main 5

LEDC
- build
- main
  - CMakeLists.txt
  - component.mk
  - ledc_example_main.c 5
- CMakeLists.txt
- Makefile
- README.md
- sdkconfig

OUTLINE
TIMELINE
AL OUTLINE

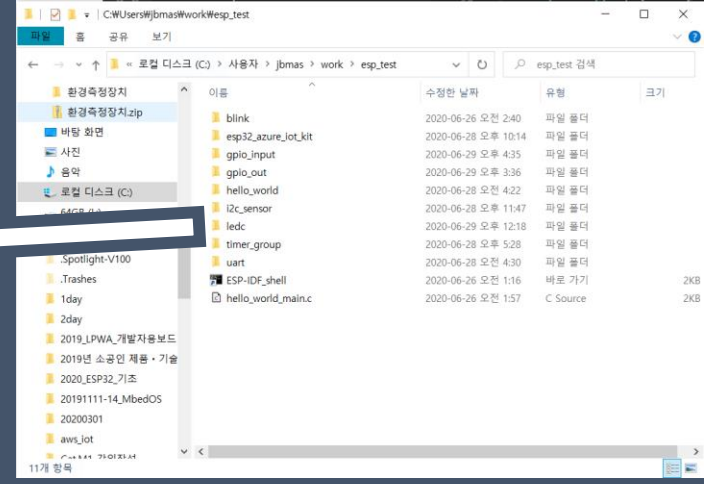
ledc_example_main.c

main > ledc_example_main.c > LEDC_LS_TIMER

```c
/* LEDC (LED Controller) fade example

   This example code is in the Public Domain (or CC0 licensed, at your option.)

   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 <stdio.h>
#include "freertos/FreeRTOS.h"
#include "freertos/task.h"
#include "driver/ledc.h"
#include "esp_err.h"

/*
 * About this example
 *
 * 1. Start with initializing LEDC module:
 *    a. Set the timer of LEDC first, this determines the frequency
 *       and resolution of PWM.
 *    b. Then set the LEDC channel you want to use,
 *       and bind with one of the timers.
 *
 * 2. You need first to install a default fade function,
 *    then you can use fade APIs.
 *
 * 3. You can also set a target duty directly without fading.
 *
 * 4. This example uses GPIO18/19/4/5 as LEDC output,
 *    and it will change the duty repeatedly.
```

PROBLEMS 5 OUTPUT DEBUG CONSOLE TERMINAL — 1: cmd

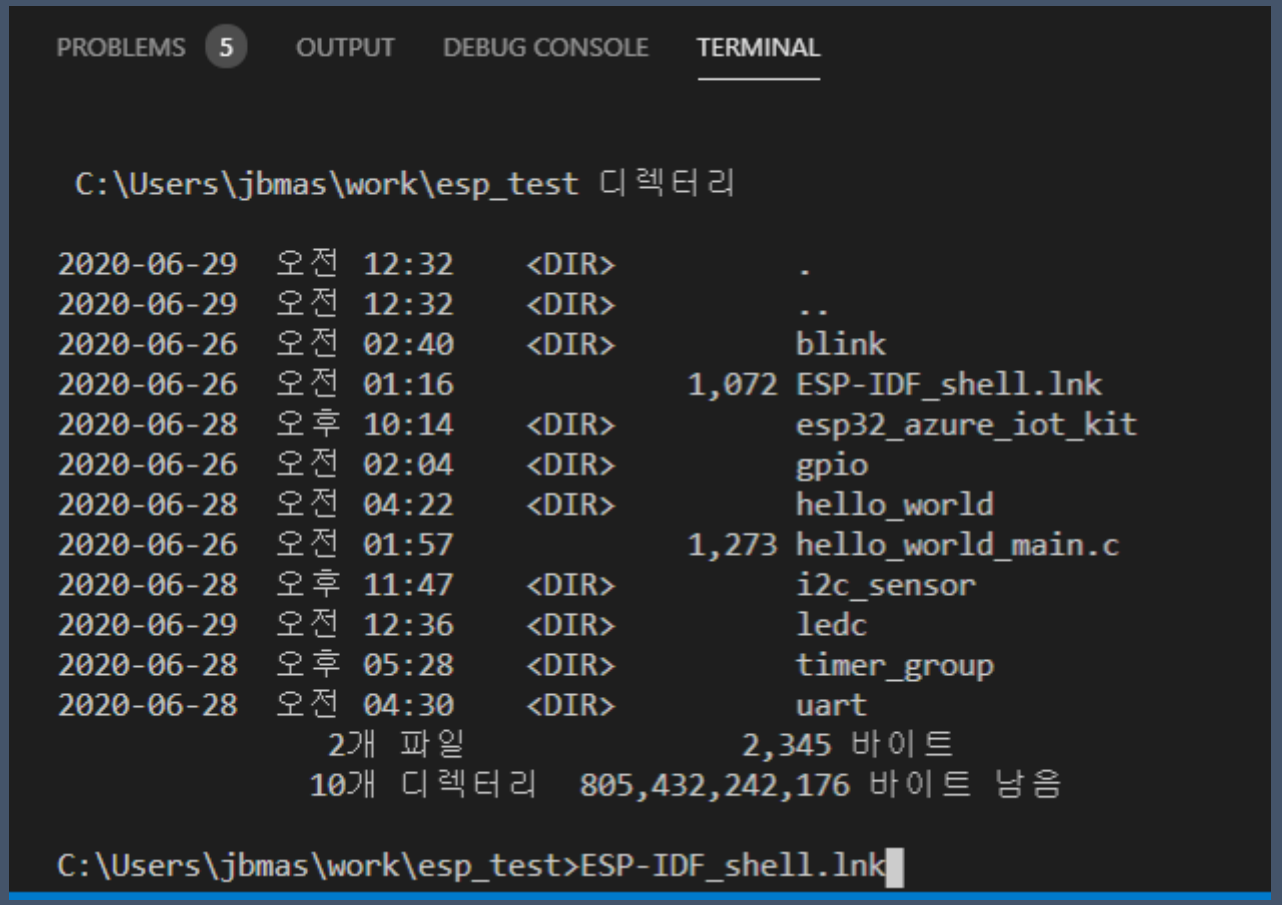
```
Microsoft Windows [Version 10.0.18363.900]
(c) 2019 Microsoft Corporation. All rights reserved.

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

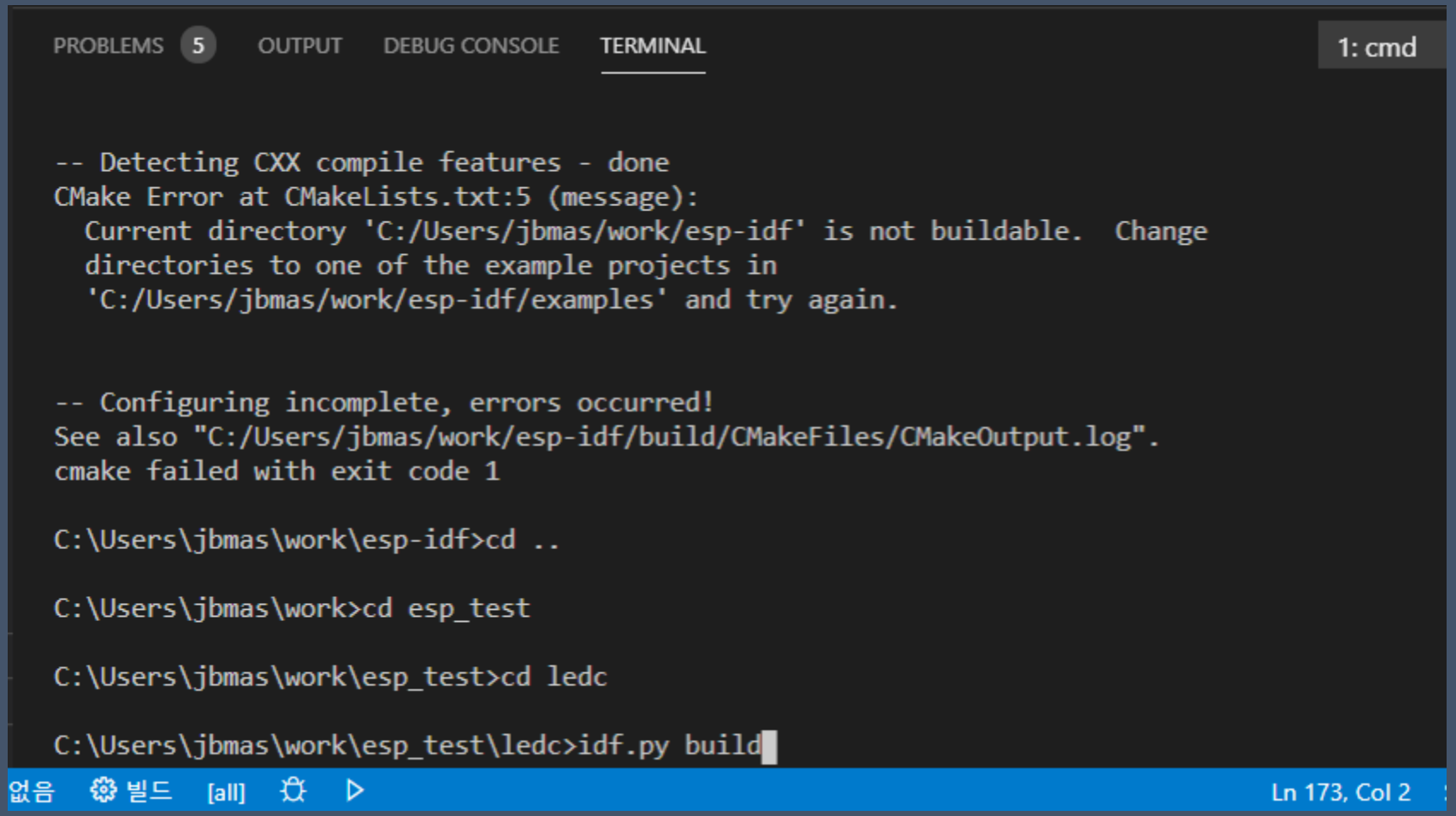


예제를 가져와서 작업하려면 esp_test 디렉토리 하나 만들어서 example 만 가져와서 빌드해서 사용,

이때 ESP-IDF CMD 바로가기를 터미널에서 실행

5 ⚠ 0 | CMake: [Debug]: 준비 | 선택된 키트가 없음 | 빌드 | [all] | Ln 43, Col 44 | Spaces: 4 | UTF-8 | CRLF | C | Win32

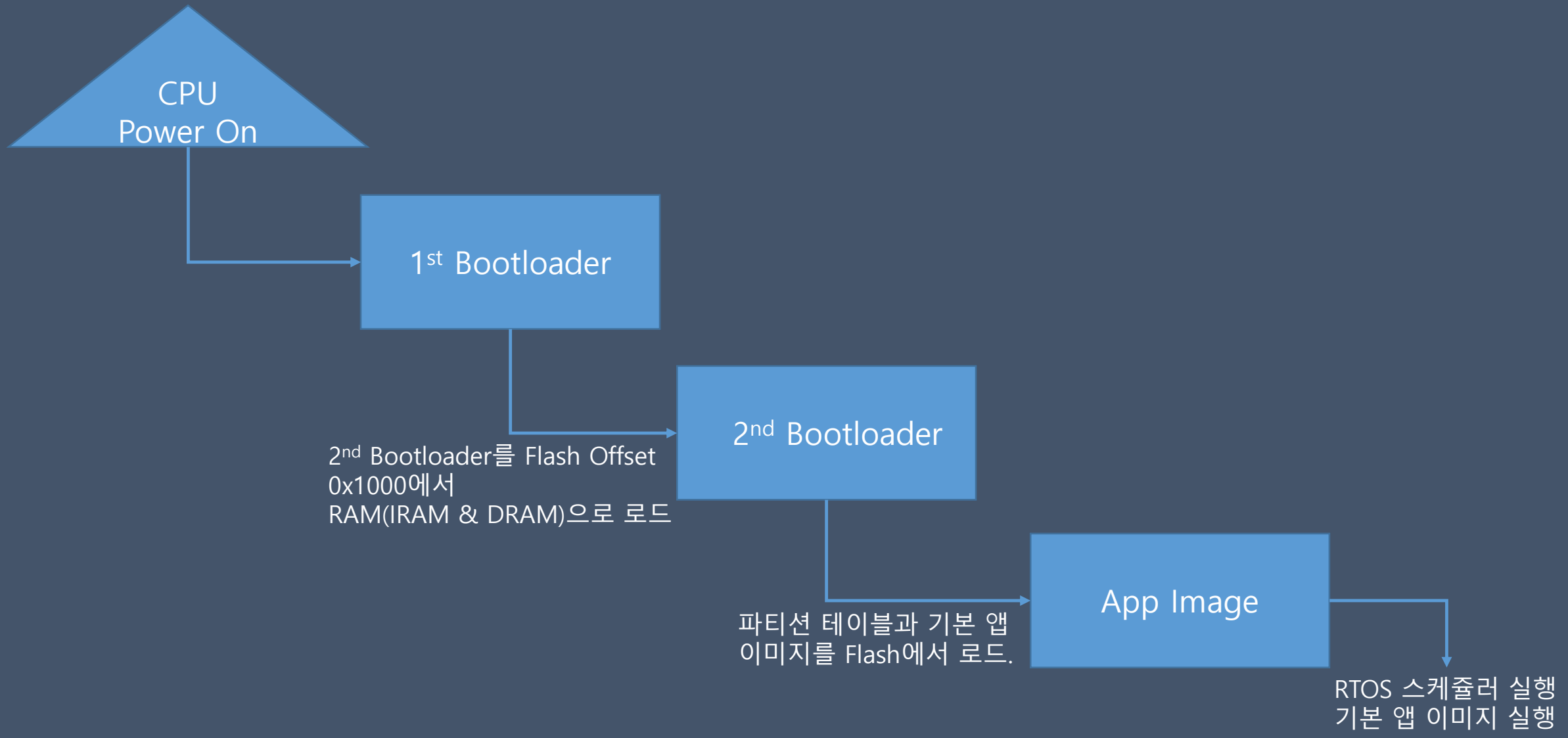


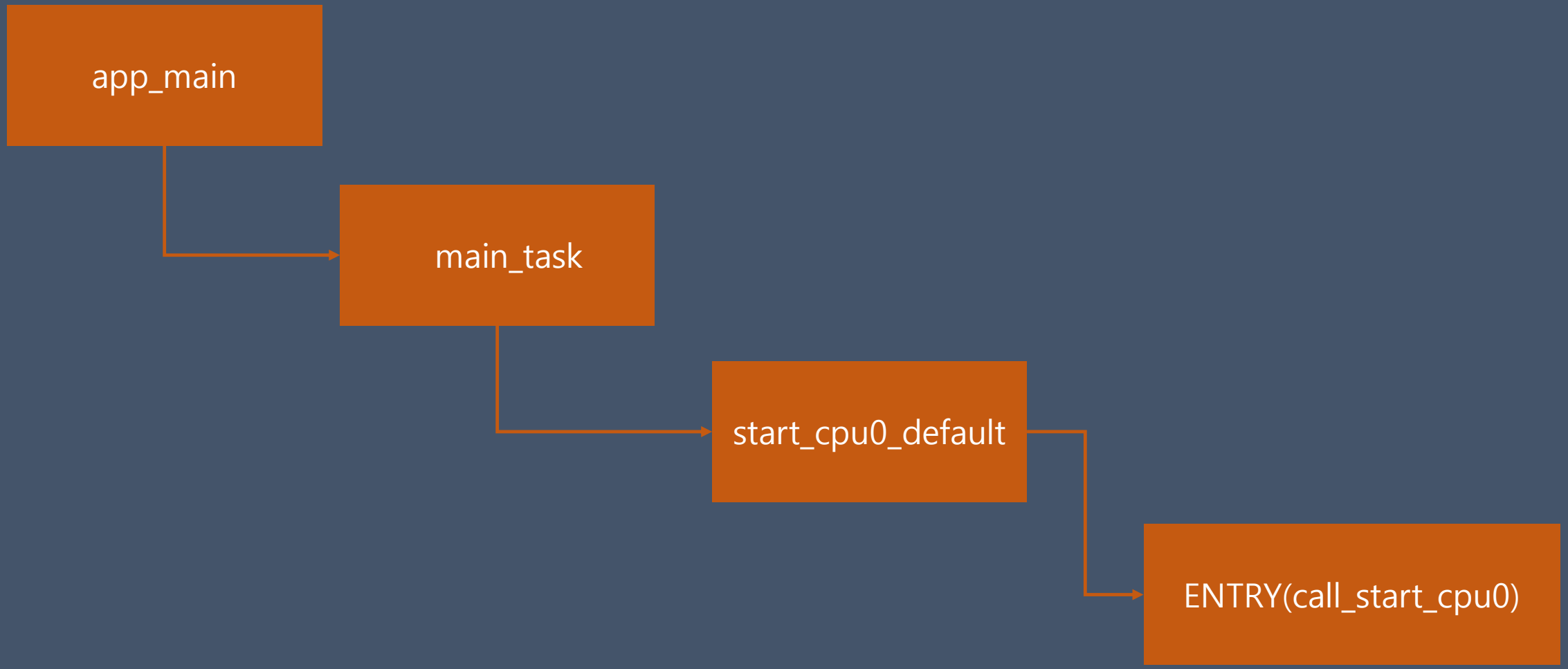


```
../main/ledc_example_main.c:114:9: note: (near initialization for 'ledc_channel')
../main/ledc_example_main.c:122:9: warning: excess elements in array initializer
        {
        ^
../main/ledc_example_main.c:122:9: note: (near initialization for 'ledc_channel')
[827/827] Generating binary image 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()

```c
#include "esp_system.h"
#include "esp_spi_flash.h"

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

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

    printf("silicon revision %d, ", chip_info.revision);

    printf("%dMB %s flash\n", spi_flash_get_chip_size() / (1024 * 1024),
            (chip_info.features & CHIP_FEATURE_EMB_FLASH) ? "embedded" : "external");

    printf("Free heap: %d\n", esp_get_free_heap_size());

    for (int i = 10; i >= 0; i--) {
        printf("Restarting in %d seconds...\n", i);
        vTaskDelay(1000 / portTICK_PERIOD_MS);
    }
    printf("Restarting now.\n");
    fflush(stdout);
    esp_restart();
}
```


CPU
Power On
1st Bootloader
2nd Bootloader를 Flash Offset
0x1000에서
RAM(IRAM & DRAM)으로 로드
2nd Bootloader
파티션 테이블과 기본 앱
이미지를 Flash에서 로드.
App Image
RTOS 스케쥴러 실행
기본 앱 이미지 실행


app_main
main_task
start_cpu0_default
ENTRY(call_start_cpu0)

```
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         ;
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
"./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_
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
```

```
321 void start_cpu0_default(void)
322 {
323     esp_err_t err;
324     esp_setup_syscall_table();
325
326     if (s_spiram_okay) {
327 #if CONFIG_SPIRAM_BOOT_INIT && (CONFIG_SPIRAM_USE_CAPS_ALLOC || CONFIG_SPIRAM_USE_MALLOC)
328         esp_err_t r=esp_spiram_add_to_heapalloc();
329         if (r != ESP_OK) {
330             ESP_EARLY_LOGE(TAG, "External RAM could not be added to heap!");
331             abort();
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     trax_start_trace(TRAX_DOWNCOUNT_WORDS);
347 #endif
348     esp_clk_init();
349     esp_perip_clk_init();
350     intr_matrix_clear();
351
"./components/esp32/cpu_start.c" 587 lines --59%--
```

```
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_PRIO, 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 }
```

```
78 void start_cpu0(void) __attribute__((weak, alias("start_cpu0_default"))) __attribute__((noreturn));
79 void start_cpu0_default(void) IRAM_ATTR __attribute__((noreturn));
80 #if !CONFIG_FREERTOS_UNICORE
81 static void IRAM_ATTR call_start_cpu1() __attribute__((noreturn));
82 void start_cpu1(void) __attribute__((weak, alias("start_cpu1_default"))) __attribute__((noreturn));
83 void start_cpu1_default(void) IRAM_ATTR __attribute__((noreturn));
84 static bool app_cpu_started = false;
85 #endif //!CONFIG_FREERTOS_UNICORE
86
87 static void do_global_ctors(void);
88 static void main_task(void* args);
89 extern void app_main(void);
90 extern esp_err_t esp_pthread_init(void);
91
92 extern int _bss_start;
93 extern int _bss_end;
94 extern int _rtc_bss_start;
95 extern int _rtc_bss_end;
96 #if CONFIG_SPIRAM_ALLOW_BSS_SEG_EXTERNAL_MEMORY
"components/esp32/cpu_start.c" 556 lines --13%--                                77,0
```

```
119 void IRAM_ATTR call_start_cpu0()
120 {
121 #if CONFIG_FREERTOS_UNICORE
122     RESET_REASON rst_reas[1];
123 #else
124     RESET_REASON rst_reas[2];
125 #endif
126     cpu_configure_region_protection();
127     cpu_init_memctl();
128
129     //Move exception vectors to IRAM
130     asm volatile (\
131                   "wsr    %0, vecbase\n" \
132                   ::"r"(&_init_start));
133
134     rst_reas[0] = rtc_get_reset_reason(0);
135
136 #if !CONFIG_FREERTOS_UNICORE
137     rst_reas[1] = rtc_get_reset_reason(1);
138 #endif
139
140     // from panic handler we can be reset by RWDT or TG0WDT
141     if (rst_reas[0] == RTCWDT_SYS_RESET || rst_reas[0] == TG0WDT_SYS_RESET
142 #if !CONFIG_FREERTOS_UNICORE
143         || rst_reas[1] == RTCWDT_SYS_RESET || rst_reas[1] == TG0WDT_SYS_RESET
144 #endif
```

```
248     /* Initialize heap allocator. WARNING: This *needs* to happen *after* the app cpu has booted.
249        If the heap allocator is initialized first, it will put free memory linked list items into
250        memory also used by the ROM. Starting the app cpu will let its ROM initialize that memory,
251        corrupting those linked lists. Initializing the allocator *after* the app cpu has booted
252        works around this problem.
253        With SPI RAM enabled, there's a second reason: half of the SPI RAM will be managed by the
254        app CPU, and when that is not up yet, the memory will be inaccessible and heap_caps_init may
255        fail initializing it properly. */
256     heap_caps_init();
257
258     ESP_EARLY_LOGI(TAG, "Pro cpu start user code");
259     start_cpu0();
260 }
```

```
/* Simplified memory map for the bootloader

   The main purpose is to make sure the bootloader can load into main memory
   without overwriting itself.
*/

MEMORY
{
  /* I/O */
  dport0_seg (RW) :                 org = 0x3FF00000, len = 0x10
  /* IRAM POOL1, used for APP CPU cache. Bootloader runs from here during the final stage of loa
  iram_loader_seg (RWX) :           org = 0x40078000, len = 0x8000  /* 32KB, APP CPU cache */
  /* 63kB, IRAM. We skip the first 1k to prevent the entry point being
     placed into the same range as exception vectors in the app.
     This leads to idf_monitor decoding ROM bootloader "entry 0x40080xxx"
     message as one of the exception vectors, which looks scary to users.
  */
  iram_seg (RWX) :                  org = 0x40080400, len = 0xfc00
  /* 64k at the end of DRAM, after ROM bootloader stack */
  dram_seg (RW) :                   org = 0x3FFF0000, len = 0x10000
}

/*  Default entry point:  */
ENTRY(call_start_cpu0);


SECTIONS
{

  .iram_loader.text :
  {
    . = ALIGN (16);
    _loader_text_start = ABSOLUTE(.);
    *(.stub .gnu.warning .gnu.linkonce.literal.* .gnu.linkonce.t.*.literal .gnu.linkonce.t.*)
     *(.iram1 .iram1.*) /* catch stray IRAM_ATTR */
    *liblog.a:(.literal .text .literal.* .text.*)
    *libgcc.a:(.literal .text .literal.* .text.*)
    *libbootloader_support.a:bootloader_clock.*(.literal .text .literal.* .text.*)
    *libbootloader_support.a:bootloader_common.*(.literal .text .literal.* .text.*)
"./components/bootloader/subproject/main/esp32.bootloader.ld" 169L, 5070C
```

ENTRY
어플리케이션의
시작포인트

감사합니다.