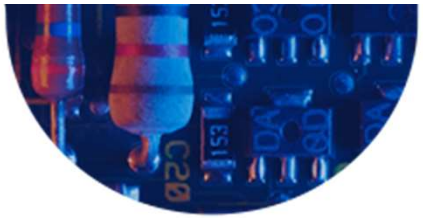


Gamba Labs

13주차 실험:감바랩스 ESP-IDF

실험 목적

- ESP 기반 MCU 펌웨어 포팅의 이해
- ESP-IDF 프레임워크 활용
- TensorFlow Lite 및 TensorFlow Lite for micro에 대한 이해

ESP-IDF

-Gamba Labs edu kit는 Espressif의 ESP32-S3 모듈을 사용
    -ESP32-S3 mcu chip의 펌웨어은 ESP-IDF 프레임워크를 사용

-ESP-IDF (Espressif IoT Development Framework)
    -ESP chip을 개발하기 위한 공식 프레임워크
    -공식문서 및 깃허브에서 제공하는 component 및 examples를 활용하여 개발 가능
    -공식문서: https://docs.espressif.com/projects/esp-idf/en/v5.0.7/esp32s3/get-started/index.html
    -깃허브: https://github.com/espressif/esp-idf

ESP-IDF 주요 명령어

- idf.py: ESP-IDF CLI build management tool을 사용하기 위한 기본 명령어
  - 사용법: idf.py [OPTIONS] COMMAND1 [ARGS]... [COMMAND2 [ARGS]...]...

  - **idf.py set-target [TARGET CHIP NAME]**: build를 위한 타겟 chip 설정
  - **idf.py menuconfig**: 보드에 대한 기본 설정(블루투스, 메모리 설정 등...)
  - **idf.py build**: 현재 프로젝트를 빌드(컴파일과 유사, 펌웨어 포팅을 위한 준비단계)
  - **idf.py flash**: build된 펌웨어를 보드에 flash
  - **idf.py monitor**: 연결된 보드의 실시간 로그를 확인할 수 있는 monitor 확인
  - **idf.py fullclean**: 현재 빌드를 모두 삭제 (프로젝트 환경이 바뀐 경우 삭제 후 다시 build 해야함)

실험 과정

환경설정

Requirements

❖ OS: Windows

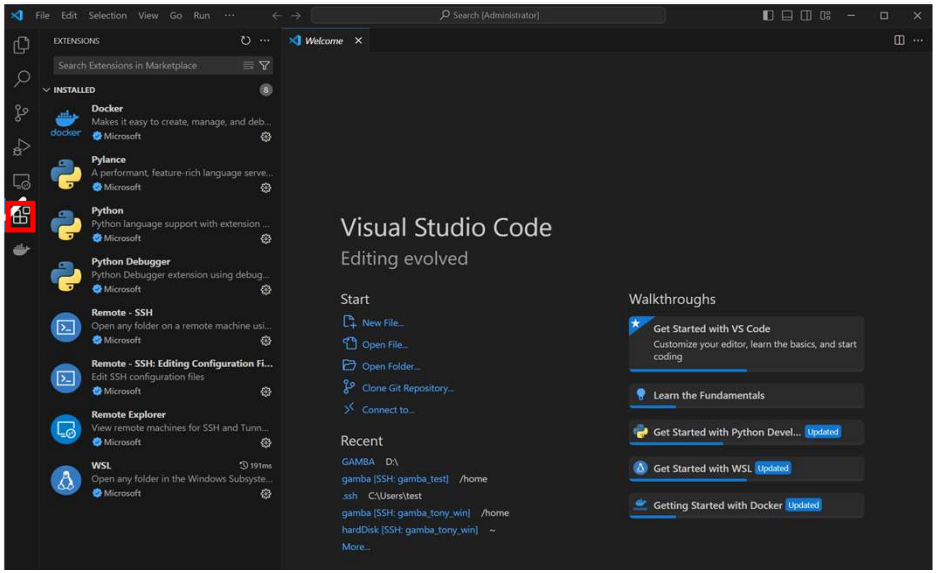
❖ IDE: Visual Studio Code (lastest recommended)
- VScode-esp-idf-extention(Git)
  https://github.com/espressif/vscode-esp-idf-extension/blob/master/docs/tutorial/install.md

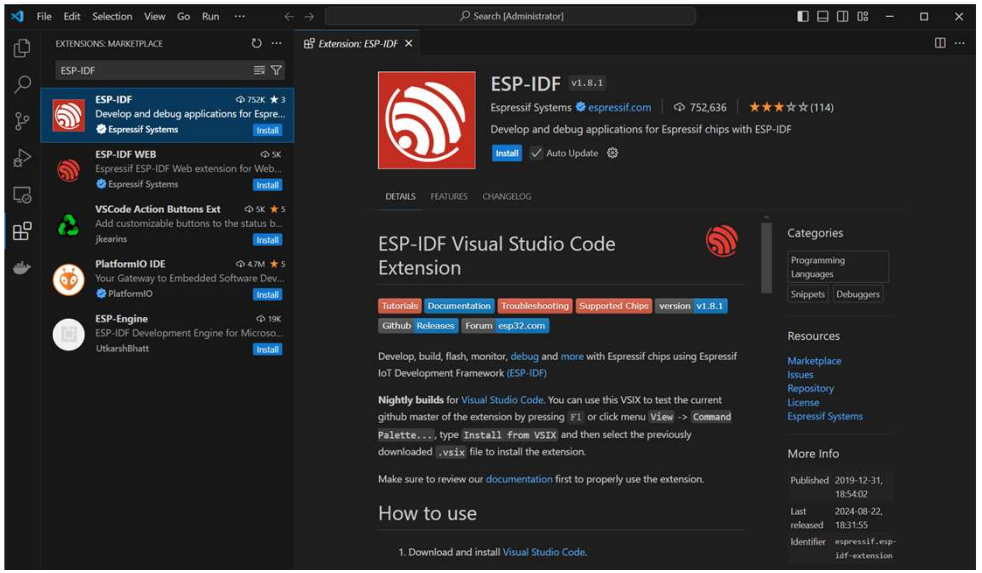
- VScode-esp-idf-extention(Git)(reference)
  https://github.com/espressif/vscode-esp-idf-extension.git

실험 과정

환경설정: VS Code ESP-IDF Extension 설치

VS Code의 Extensions항목에서 ESP-IDF 설치
- Extensions 이동: Ctrl + Shift + X
- Search "ESP-IDF"
- Install the Extension



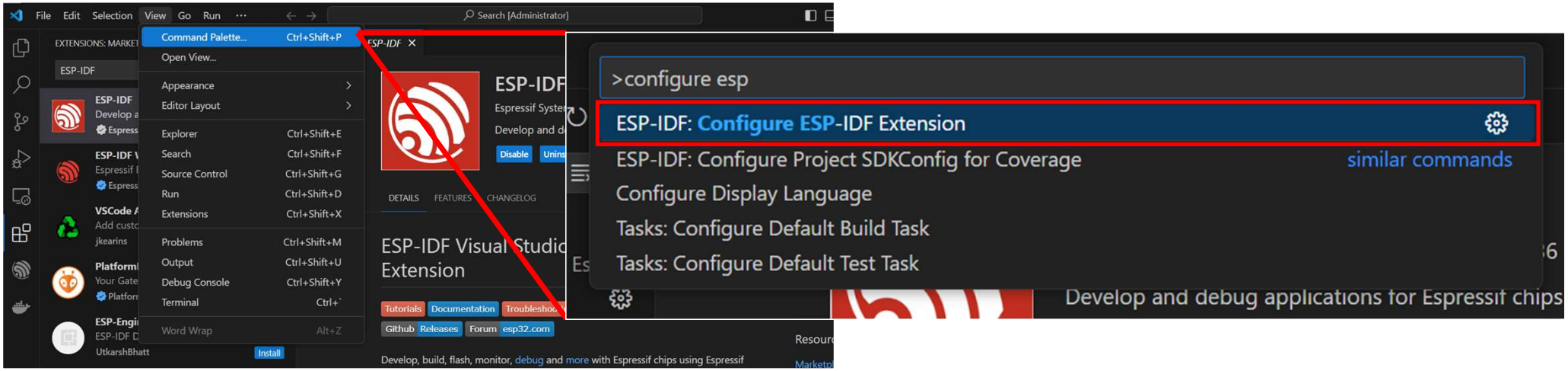


실험 과정

환경설정: VS Code ESP-IDF Extension 설치

Command Palette에서 configure esp 검색
- Ctrl+Shift+P or F1

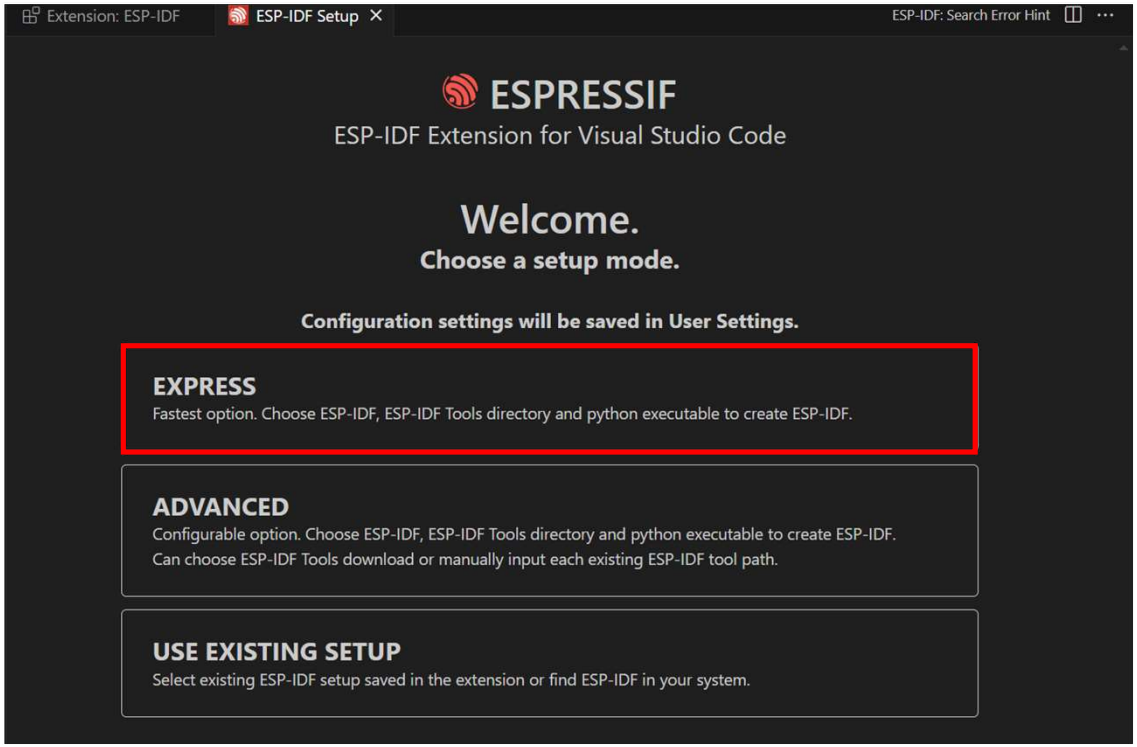
ESP-IDF: Configure ESP-IDF Extension 클릭



실험 과정

환경설정: VS Code ESP-IDF Extension 설치

EXPRESS 클릭



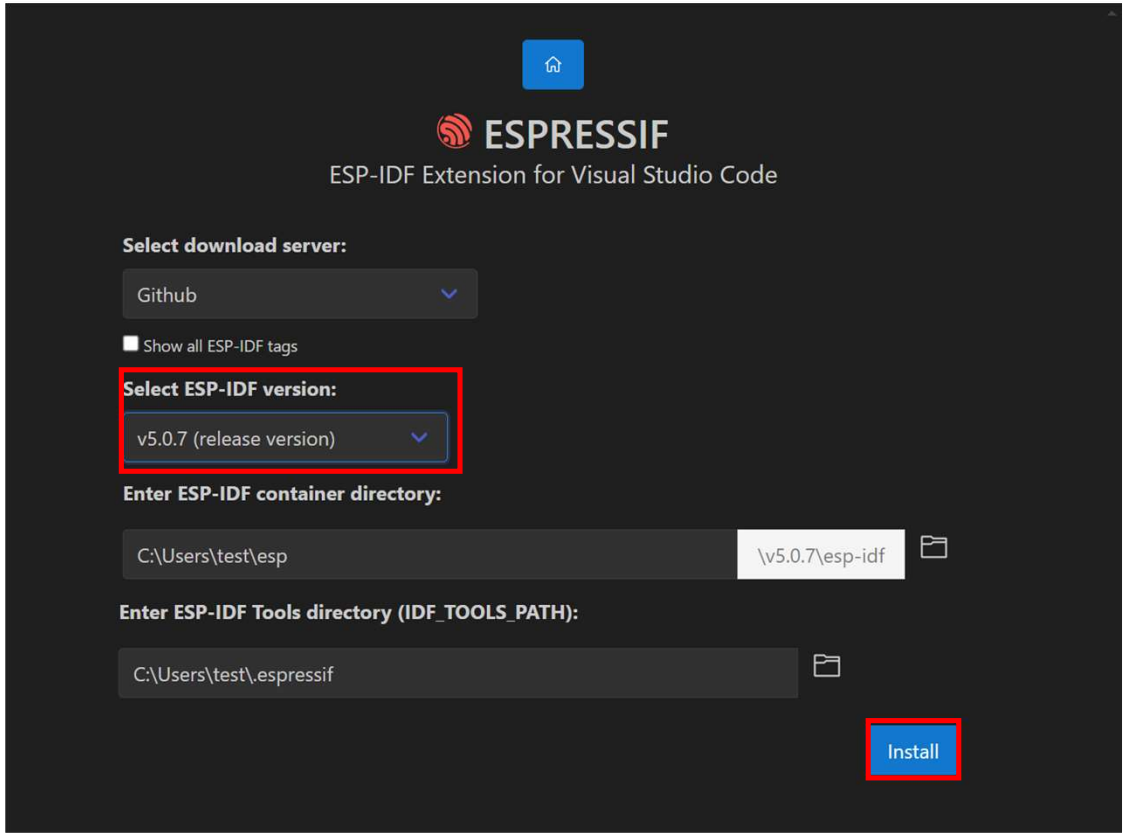
실험 과정

환경설정: VS Code ESP-IDF Extension 설치

ESP-IDF versio을 v5.0.7 (release version) 으로 선택 후 Install

디스크에 15GB의 공간이 필요

ESP-IDF Tools 를 저장할 공간을 선택할 수 있으나, 기본 설정을 따르는 것을 추천

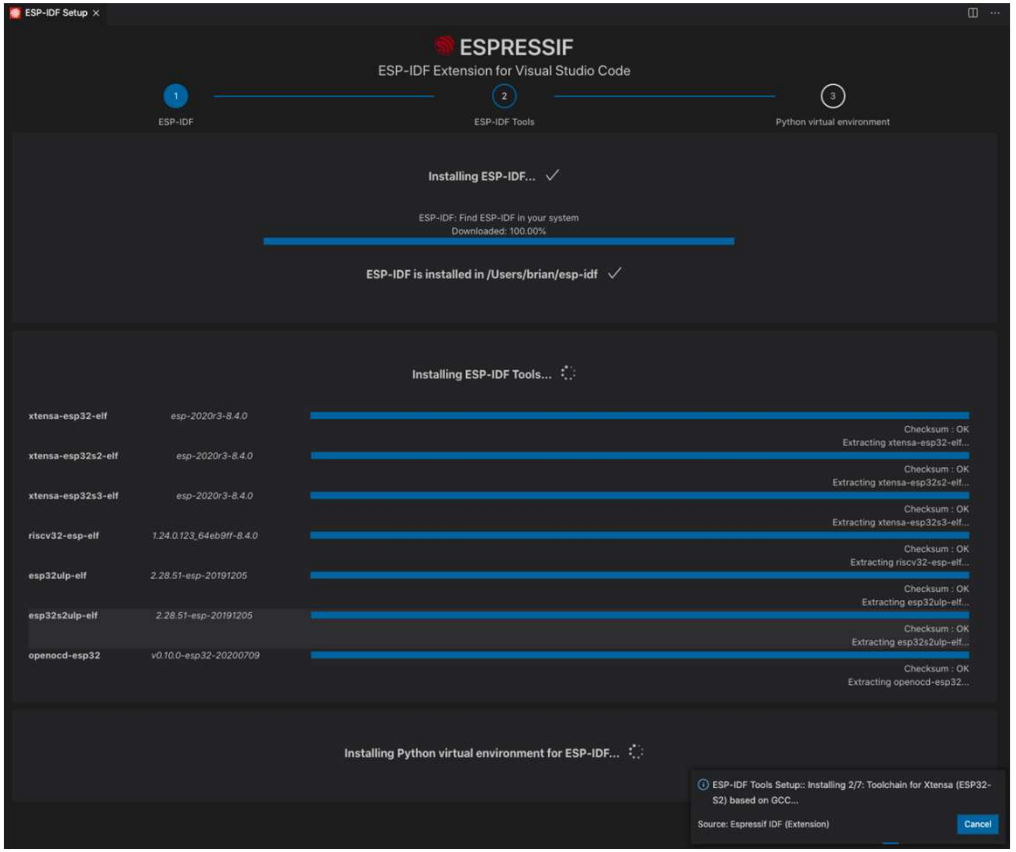


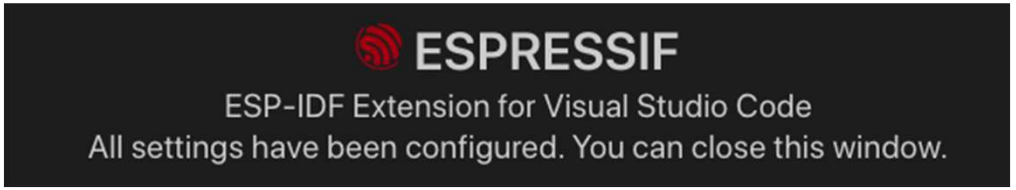
실험 과정

환경설정: VS Code ESP-IDF Extension 설치

설치에 몇 분의 시간 소요

설치 완료 시, 아래와 같은 메시지 확인가능



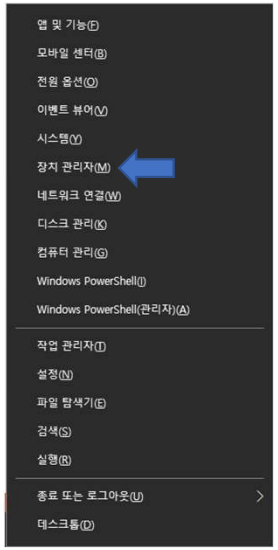


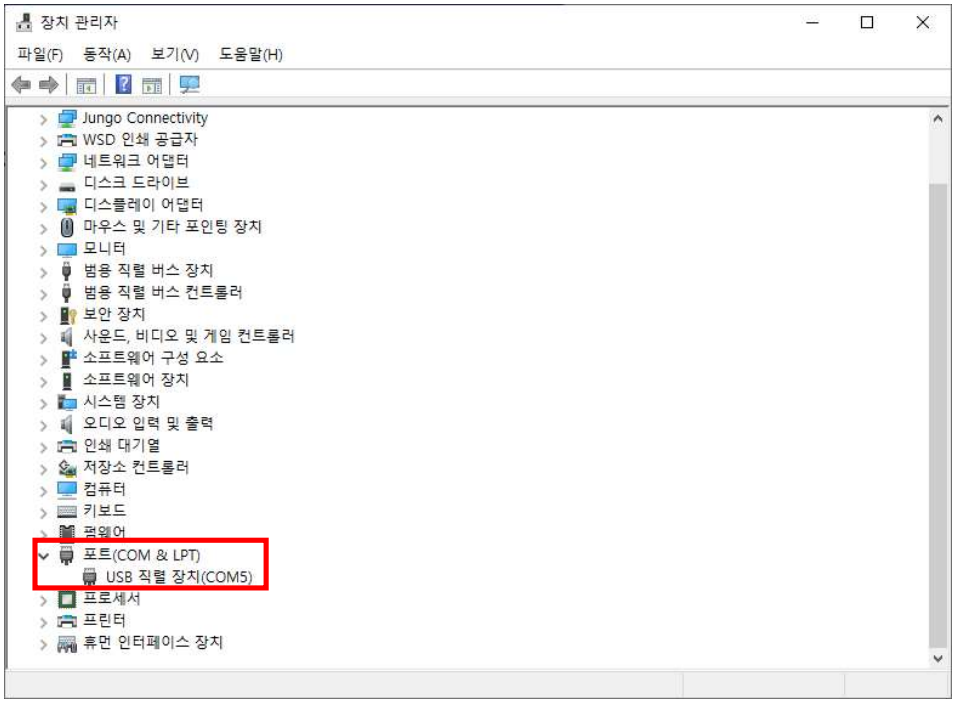
실험 과정

환경설정: 시리얼 포트 찾기

장치 관리자를 통해 연결된 보드의 포트 확인

Windows에서 "장치 관리자" 검색
Win key + X and M 을 통해서도 실행가능



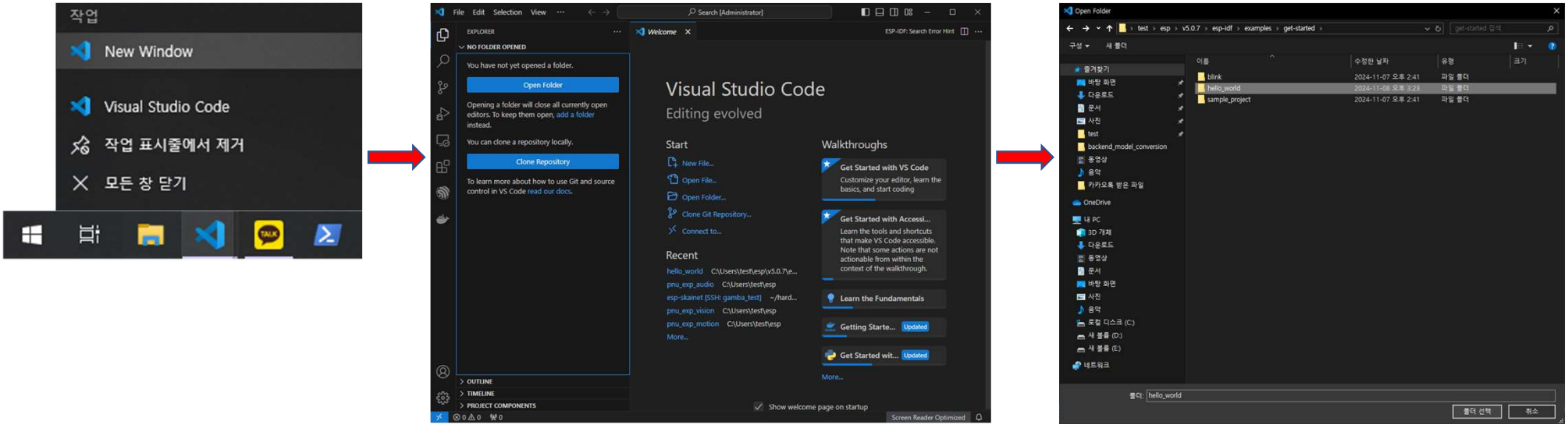


실험 과정

Example #0: Hello World

ESP-IDF를 설치했던 경로의 아래 예제 프로젝트 폴더를 열기

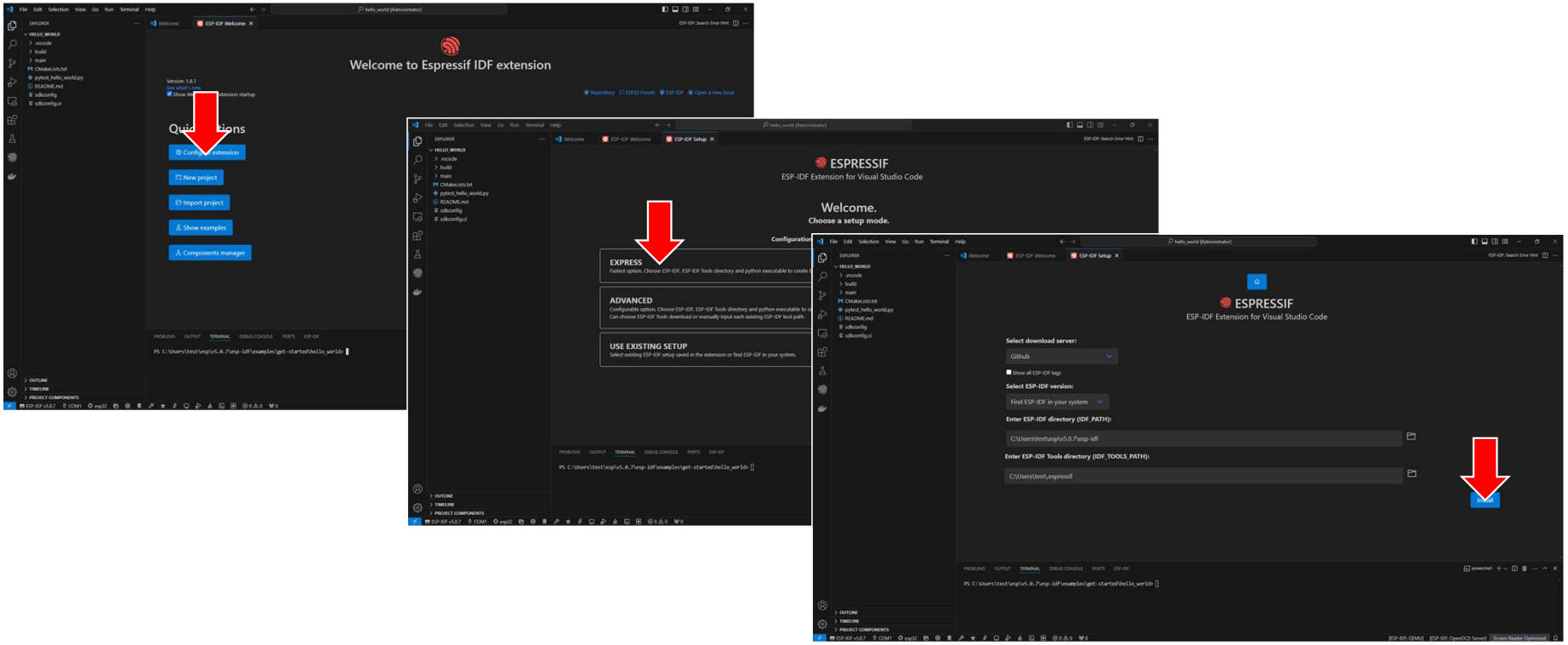
path : $IDF_PATH₩esp-idf₩examples₩get-started₩get-started₩hello_world
PS C:₩Users₩test₩esp₩v5.0.7₩esp-idf₩examples₩get-started₩hello_world>



실험 과정

Example #0: Hello World

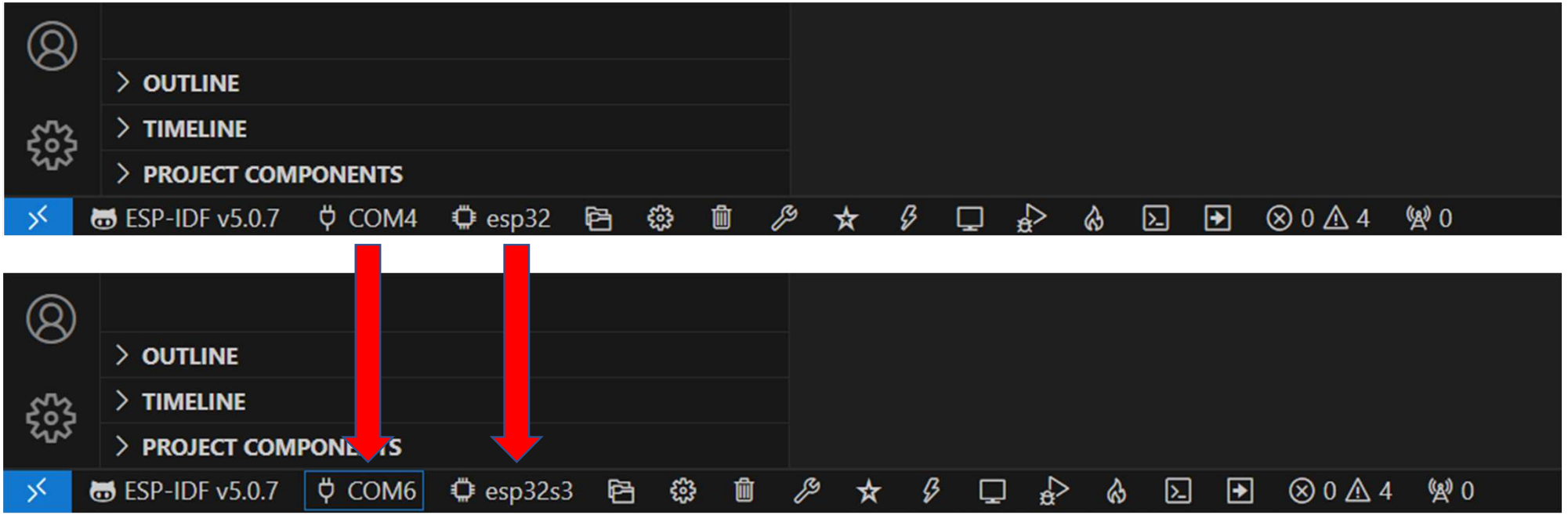
아래의 순서대로 프로젝트 Setup



실험 과정

Example #0: Hello World

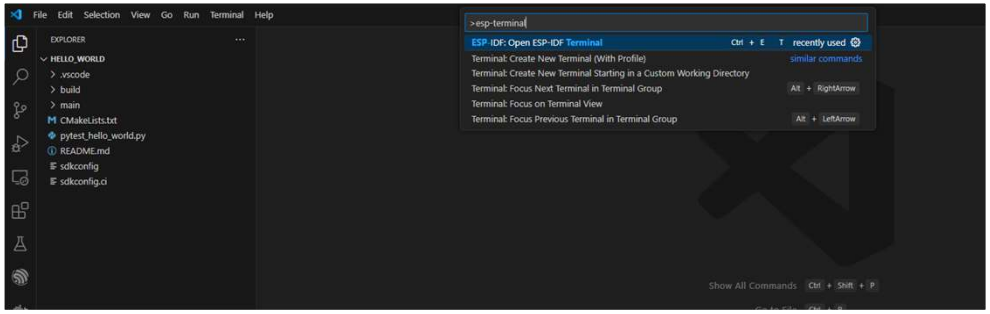
VS Code 하단의 시리얼 포트와 타겟 보드를 알맞게 변경
- 시리얼 포트: 11p에서 확인한 보드의 시리얼 포트 COMx
- 타겟 보드: esp32s3



실험 과정

Example #0: Hello World

ESP-IDF terminal 실행
- F1 클릭, esp-idf terminal 검색 후 ESP-IDF: Open ESP-IDF Terminal 클릭
- VS Code 하단의 터미널 아이콘 클릭



실험 과정

Example #0: Build

펌웨어 Flash(포팅) 이전에 프로젝트 빌드가 필요 (컴파일 과정 등)
ESP-IDF Terminal에서 idf.py build 명령어 실행

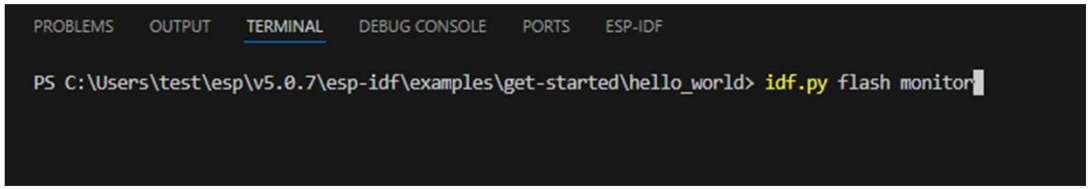
```
PROBLEMS  OUTPUT  TERMINAL  DEBUG CONSOLE  PORTS  ESP-IDF

PS C:\Users\test\esp\v5.0.7\esp-idf\examples\get-started\hello_world> idf.py build
Executing action: all (aliases: build)
Running ninja in directory C:\Users\test\esp\v5.0.7\esp-idf\examples\get-started\hell
Executing "ninja all"...
[0/1] Re-running CMake...
-- Adding linker script C:/Users/test/esp/v5.0.7/esp-idf/components/bootloa
-- Adding linker script C:/Users/test/esp/v5.0.7/esp-idf/components/bootloa
-- Components: bootloader bootloader_support efuse esp_app_format esp_commo
-- Component paths: C:/Users/test/esp/v5.0.7/esp-idf/components/bootloader
idf/components/esp_app_format C:/Users/test/esp/v5.0.7/esp-idf/components/e
0.7/esp-idf/components/esp_system C:/Users/test/esp/v5.0.7/esp-idf/componen
p-idf/components/log C:/Users/test/esp/v5.0.7/esp-idf/components/bootloader
nents/newlib C:/Users/test/esp/v5.0.7/esp-idf/components/partition_table C:
s/xtensa
-- Configuring done
-- Generating done
-- Build files have been written to: C:/Users/test/esp/v5.0.7/esp-idf/examp
[1/1] cmd.exe /C "cd /D C:\Users\test\esp\v5.0.7\esp-idf\examples\get-start
Bootloader binary size 0x6760 bytes. 0x8a0 bytes (8%) free.

Project build complete. To flash, run this command:
C:\Users\test\.espressif\python_env\idf5.0_py3.11_env\Scripts\python.exe ..
h_mode dio --flash_size 2MB --flash_freq 40m 0x1000 build\bootloader\bootlo
or run 'idf.py -p (PORT) flash'
PS C:\Users\test\esp\v5.0.7\esp-idf\examples\get-started\hello_world>
```

idf.py build

실험 과정

Example #0: Flash and Monitor

펌웨어 Flash(포팅) 이전에 프로젝트 빌드가 필요 (컴파일 과정 등)
ESP-IDF Terminal에서 idf.py build 명령어 실행



idf.py flash monitor

idf.py –p COMx flash monitor

-p 옵션을 통해 포트 지정 가능
지정 안한 경우 자동으로 탐색

"Ctrl+]" 를 통해 monitor 출력 종료

실험 과정

Example #0: Troubleshooting

빌드나 타겟 보드 설정 과정에서 문제가 발생시

프로젝트 폴더의 build 폴더 삭제 후 idf.py clean, idf.py fullclean 명령어 입력 후 다시 설정

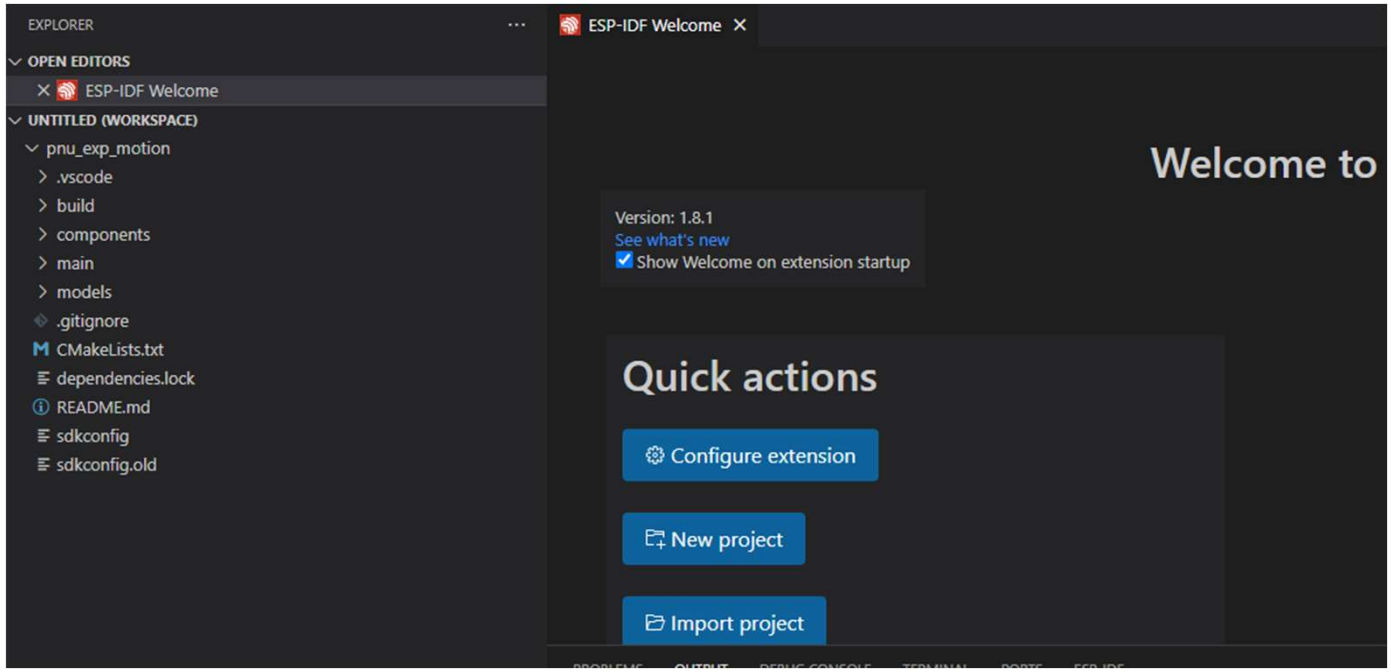
```
idf.py clean
```

```
idf.py fullclean
```

실험 과정

Example #1-3: Assignment

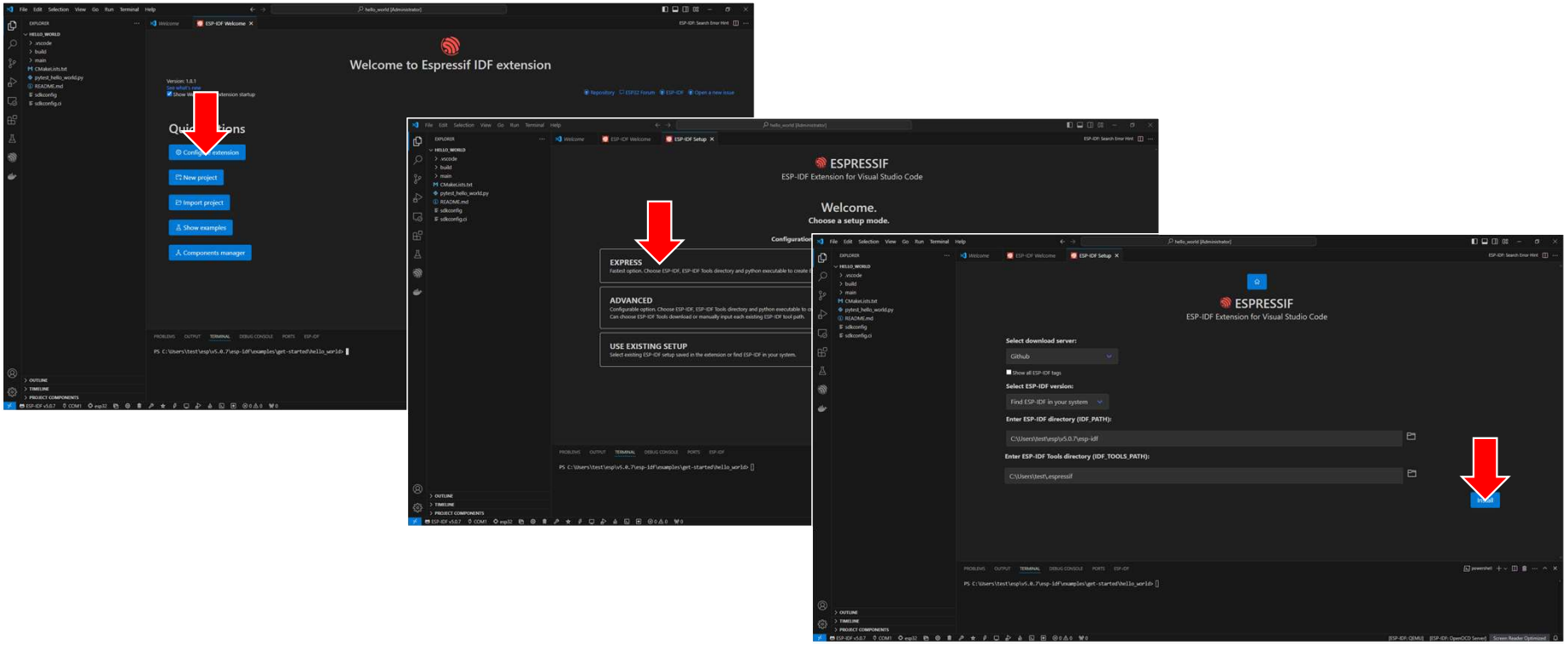
VS Code를 통해 이번주차 제공파일 중 하나의 프로젝트 폴더 오픈
pnu_exp_motion, pnu_exp_audio, pnu_exp_vision중 하나 선택



실험 과정

Example #1-3: Assignment

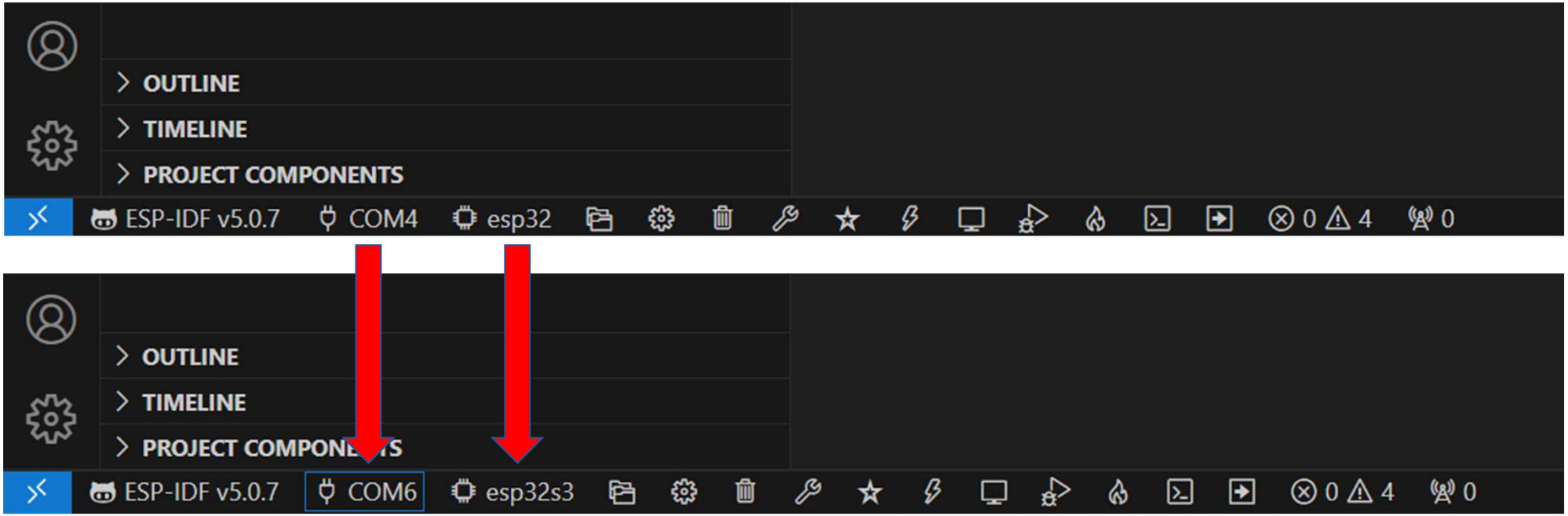
아래의 순서대로 프로젝트 Setup



실험 과정

Example #1-3: Assignment

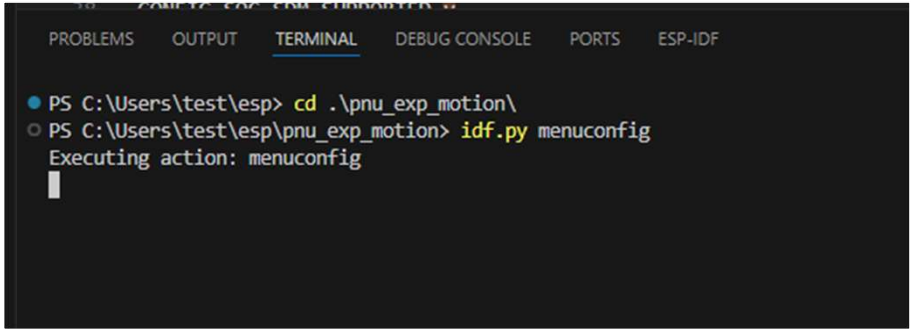
VS Code 하단의 시리얼 포트와 타겟 보드를 알맞게 변경
- 시리얼 포트: 11p에서 확인한 보드의 시리얼 포트 COMx
- 타겟 보드: esp32s3



실험 과정

Example #1-3: Assignment

ESP-IDF Terminal 실행 후 pnu_exp_motion 폴더로 이동
idf.py menuconfig 명령어를 통해 프로젝트 및 보드에 대한 설정

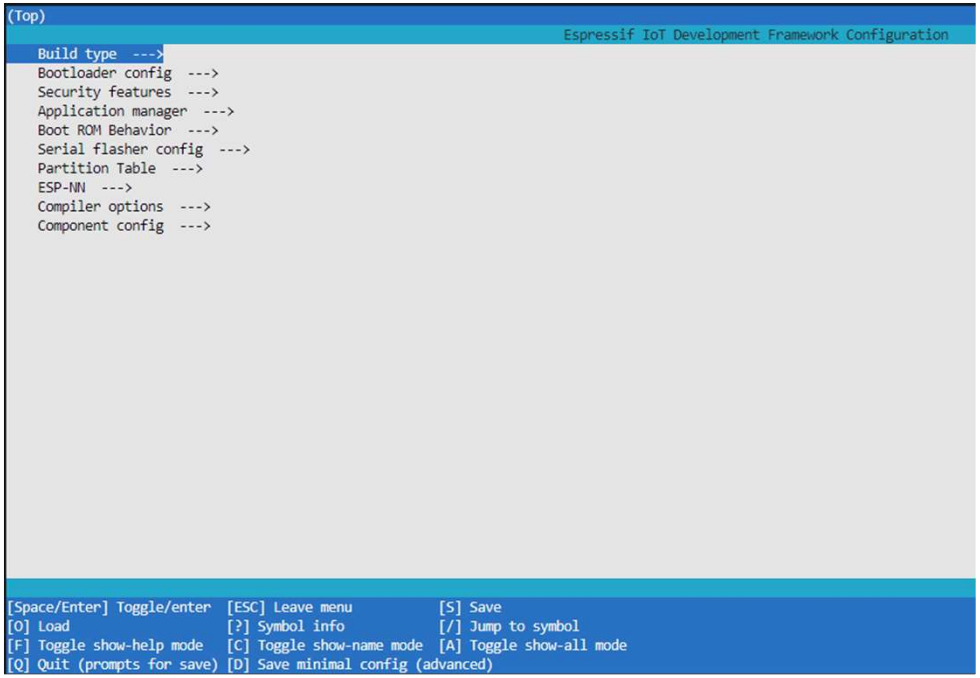


idf.py menuconfig

실험 과정

Example #1-3: SDK Configuration

Gamba labs edu kit의 하드웨어 스펙에 맞게 설정



Press [/] and "QIO" [enter] and [space]
SPI Flash의 데이터 전송 모드를 Quad I/O로 설정

Press [/] and "4MB" [enter] and [space]
칩에 연결된 외부 Flash memory의 크기를 4MB로 설정

Press [/] and "SPI-" [enter] and [space]
외부 PSRAM을 SPI 인터페이스로 사용하도록 설정

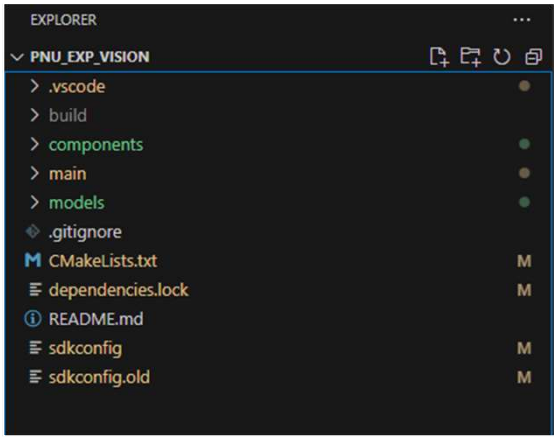
Press [/] and "Octal mode" [enter] and [space]
PSRAM의 데이터 전송 모드를 8비트로 설정

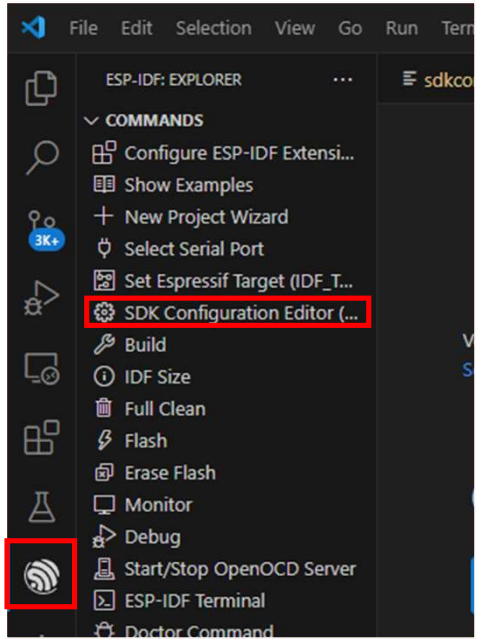
Press [q] and [y](영타)

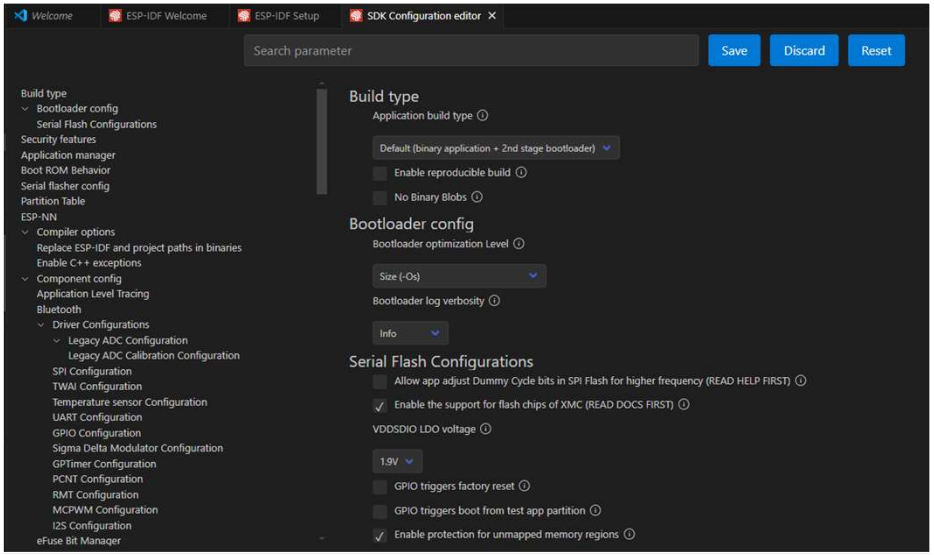
실험 과정

Example #1-3: SDK Configuration

VS Code의 Workspace를 단일 프로젝트로 열었다면 Extension 메뉴에서도 이전 페이지와 같은 설정 가능
"ESP-IDF" Extension 선택, SDL Configuration Editor 선택 -> idf.py menuconfig와 같은 설정 가능







실험 과정

Example #1-3: Troubleshooting

Monitor에서 다음과 같은 에러가 발생한다면,
idf.py menuconfig에서 아래와 같은 설정

Press [/] and "0 Idle" [enter] and [space]
Press [/] and "1 Idle" [enter] and [space]

```
[*]     Also watch CPU1 tick interrupt
[*] Enable Task Watchdog Timer
[*]     Initialize Task Watchdog Timer on startup
[ ]         Invoke panic handler on Task Watchdog timeout
(5)         Task Watchdog timeout period (seconds)
[ ]         Watch CPU0 Idle Task
[ ]         Watch CPU1 Idle Task
[ ] Place panic handler code in IRAM
[ ] OpenOCD debug stubs
[*] Make exception and panic handlers JTAG/OCD aware
```

Press [q] and flash monitor

```
E (57280) task_wdt: Task watchdog got triggered. The following tasks/users did not reset the watchdog in time:
E (57280) task_wdt:  - IDLE0 (CPU 0)
E (57280) task_wdt: Tasks currently running:
E (57280) task_wdt: CPU 0: main
E (57280) task_wdt: CPU 1: IDLE1
E (57280) task_wdt: Print CPU 0 (ccam_hal: EV-EOF-OVF
cam_hal: EV-VSYNC-OVF
urrent core) backtrace


Backtrace: 0x42033B26:0x3FC96CF0 0x42033CD2:0x3FC96D10 0x4037766D:0x3FC96D30 0x4200A37D:0x3FC9E410 0x4204935F:0x3FC9E440 0x4037D235:0x3FC9E470
0x42033b26: task_wdt_timeout_handling at C:/Users/test/esp/v5.0.7/esp-idf/components/esp_system/task_wdt/task_wdt.c:461 (discriminator 3)

0x42033cd2: task_wdt_isr at C:/Users/test/esp/v5.0.7/esp-idf/components/esp_system/task_wdt/task_wdt.c:585

0x4037766d: _xt_lowint1 at C:/Users/test/esp/v5.0.7/esp-idf/components/freertos/FreeRTOS-Kernel/portable/xtensa/xtensa_vectors.S:1122

0x4200a37d: app_main at C:/Users/test/esp/pnu_exp_vision/main/app_main.c:116

0x4204935f: main_task at C:/Users/test/esp/v5.0.7/esp-idf/components/freertos/app_startup.c:208 (discriminator 13)

0x4037d235: vPortTaskWrapper at C:/Users/test/esp/v5.0.7/esp-idf/components/freertos/FreeRTOS-Kernel/portable/xtensa/port.c:149
```

실험 과정

Example #1: pnu_exp_motion

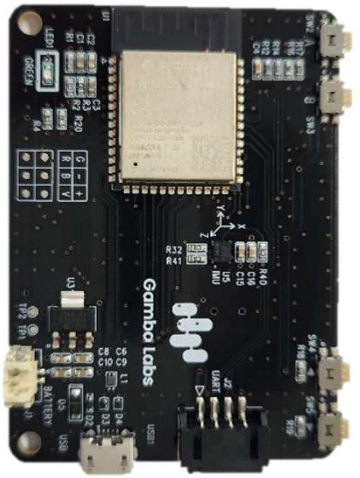
Borad에 부착된 가속도 센서를 통해 움직임을 감지하고,
Chip에 flash된 모델이 움직임을 다음과 같이 분류

- 0: 칩을 바라본 기준으로 오른쪽
- 1: 칩을 바라본 기준으로 왼쪽
- 2: 칩을 바라본 기준으로 뒤쪽 (몸 안쪽)
- 3: 칩을 바라본 기준으로 앞쪽 (몸 바깥쪽)

출력은 각 클래스(라벨)별 확률을 출력하고 추론에 걸린 시간을 표시
움직임이 감지될 때만 추론

```
0:  0.004193,   1:  0.030625,   2:  0.957324,   3:  0.007857,   [0.114ms]
0:  0.005987,   1:  0.050667,   2:  0.931655,   3:  0.011691,   [0.114ms]
0:  0.058658,   1:  0.013128,   2:  0.031722,   3:  0.896491,   [0.114ms]
0:  0.003471,   1:  0.523909,   2:  0.127059,   3:  0.345561,   [0.114ms]
0:  0.015175,   1:  0.024203,   2:  0.943866,   3:  0.016756,   [0.114ms]
0:  0.010448,   1:  0.011960,   2:  0.953114,   3:  0.024477,   [0.114ms]
0:  0.015484,   1:  0.018780,   2:  0.075918,   3:  0.889817,   [0.114ms]
0:  0.001998,   1:  0.057229,   2:  0.933519,   3:  0.007254,   [0.114ms]
0:  0.107945,   1:  0.042762,   2:  0.080914,   3:  0.768378,   [0.114ms]
```

기준 상태



실험 과정

Example #2: pnu_exp_speech

Board에 후면에 부착된 마이크 모듈을 통해 소리를 감지하고,
Chip에 flash된 모델이 음성을 다음과 같이 분류

- 0: 출발
- 1: 정지
- 2: IDLE state

출력은 각 클래스(라벨)별 확률을 출력하고 추론에 걸린 시간을 표시
상시 추론

```
0:  0.082031,   1:  0.082031,   2:  0.832031,   [119.191ms]
0:  0.082031,   1:  0.082031,   2:  0.832031,   [119.241ms]
0:  0.082031,   1:  0.082031,   2:  0.832031,   [119.196ms]
0:  0.082031,   1:  0.082031,   2:  0.832031,   [119.226ms]
0:  0.082031,   1:  0.082031,   2:  0.832031,   [119.196ms]
0:  0.082031,   1:  0.082031,   2:  0.832031,   [119.239ms]
0:  0.082031,   1:  0.082031,   2:  0.832031,   [119.186ms]
0:  0.082031,   1:  0.082031,   2:  0.832031,   [119.241ms]
0:  0.082031,   1:  0.082031,   2:  0.832031,   [119.195ms]
```

실험 과정

Example #3: pnu_exp_vision

Board에 추가적으로 장착된 카메라 모듈로 사진을 촬영하며,
촬영된 이미지는 0(White)과 1(Black)로 분류
사진 촬영은 장치 전면의 "A 버튼"을 눌러 실행

- 0: Bright or white place
- 1: Dark or black place

출력은 각 클래스(라벨)별 확률을 출력하고 추론에 걸린 시간을 표시
"A 버튼" 누를 시 추론

```
I (384950) Shot: Cheeeeeeese
0:  0.922,   1:  0.078,    [1630.424ms]
```

```
I (375800) Shot: Cheeeeeeese
0:  0.000,   1:  0.996,    [1630.320ms]
```

이미지 캡처 및 추론 버튼

실험 과정

모델 변경

TinyWebTrainer를 통해 학습/변환한 model.c 파일의 모델 데이터를
예제 프로젝트의 model.c 파일에 적용

```
const unsigned char g_model[] = {
  0x1c, 0x00, 0x00, 0x00, 0x54, 0x46, 0x4c, 0x33, 0x14, 0x00, 0x20, 0x00,
  0x1c, 0x00, 0x18, 0x00, 0x14, 0x00, 0x10, 0x00, 0x0c, 0x00, 0x00, 0x00,
  0x08, 0x00, 0x04, 0x00, 0x14, 0x00, 0x00, 0x00, 0x1c, 0x00, 0x00, 0x00,
  0xa0, 0x00, 0x00, 0x00, 0xf8, 0x00, 0x00, 0x00, 0x88, 0x6d, 0x00, 0x00,
  0x98, 0x6d, 0x00, 0x00, 0xd0, 0x73, 0x00, 0x00, 0x03, 0x00, 0x00, 0x00,
  0x01, 0x00, 0x00, 0x00, 0x10, 0x00, 0x00, 0x00, 0x00, 0x00, 0x0a, 0x00,
  0x10, 0x00, 0x0c, 0x00, 0x08, 0x00, 0x04, 0x00, 0x0a, 0x00, 0x00, 0x00,
  0x0c, 0x00, 0x00, 0x00, 0x1c, 0x00, 0x00, 0x00, 0x40, 0x00, 0x00, 0x00,
```

g_model 배열에 변환한 model.c 파일의 값들을 복사하여 사용

실험 과정

펌웨어 초기화

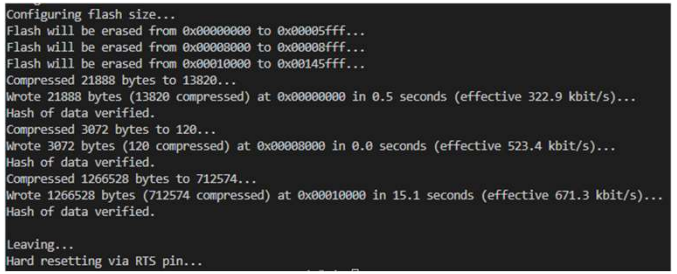
보드에 새로운 펌웨어를 Flash 한 경우 기존의 TinyWebTrainer와 연결이 불가

따라서, TinyWebTrainer와 연결을 위한 펌웨어 초기화 필요

Plato에 제공된 "Web trainer 펌웨어"을 사용하여 아래의 단계를 수행

1. 파이썬 라이브러리 esptool 설치 (esptool: ESP 펌웨어 관리를 위한 도구) `pip install esptool`
2. Web trainer 펌웨어 폴더 내에서 아래의 명령어 입력
   - 이때, COMx 시리얼 포트는 장치 설정에 맞게 수정

```
python -m esptool -p COMx -b 460800 --before default_reset --after hard_reset --chip esp32s3  write_flash --flash_mode dio --flash_size 4MB --flash_freq 80m 0x0 bootloader.bin 0x8000 partition-table.bin 0x10000 gamba_ai_edukit.bin
```



Flash 성공시 출력

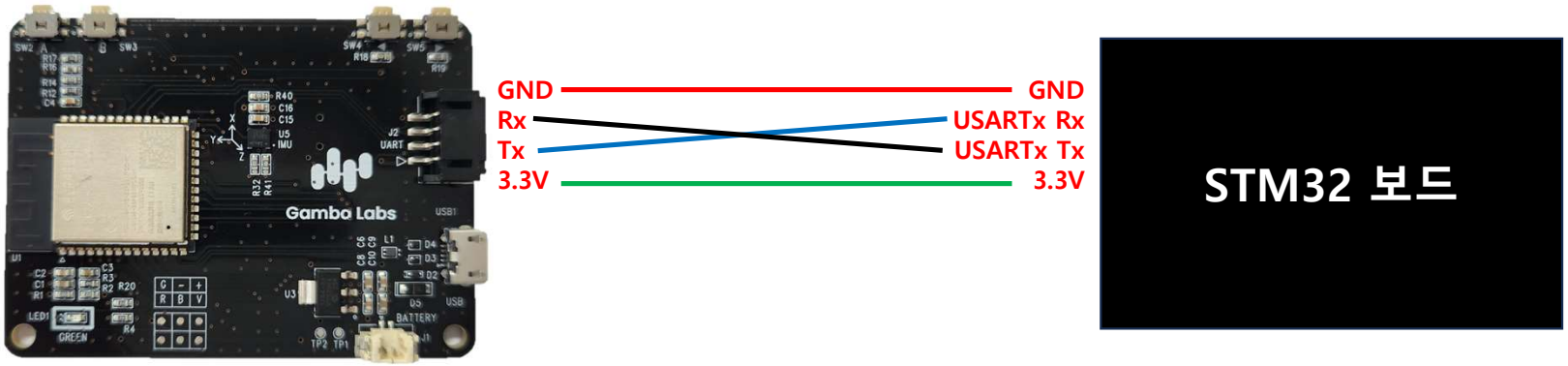
실험 검사

- TinyWebTrainer를 통해 학습/변환한 모델을 보드에 탑재 후 검사
- Motion, Speech, Vision중 하나를 선택
- 예제와 다른 클래스를 가지는 모델을 학습 및 탑재
- TinyWebTrainer 학습 페이지 확인 및 보드에서의 동작 검사
  - 학습 페이지에서 학습시킨 모델을 보드에 탑재했는지 검사
- Plato에 보드 동작 영상 및 실험 보고서 업로드

텀프로젝트 활용방안

UART를 통한 결과 전송

Gamba labs edu kit는 모델의 추론 결과를 UART 시리얼 통신을 통해 송신
STM32 보드에서 모델 추론 결과를 "Winner [class번호]" 형식으로 수신
STM32 보드에서 모델 추론 결과에 따른 연결 동작으로 구현 가능



텀프로젝트 활용방안

UART를 통한 결과 전송

UART 설정값은 ESP 프로젝트 각 main 폴더의 app_uart.c, app_uart.h를 통해 확인 가능

- Motion: IMU의 움직임이 감지되었을 때만 추론 및 결과 전송
- Speech: 상시 추론 및 결과 전송 -> 상시 추론이기에 idle 클래스가 반드시 필요
- Vision: 보드의 "버튼 A"를 눌렀을 경우에만 추론 및 결과 전송

```c
void init_uart()
{
    uart_config_t uart_config = {
        .baud_rate  = UART_BAUD_RATE,
        .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,
    };
    int intr_alloc_flags = 0;
```

app_uart.c

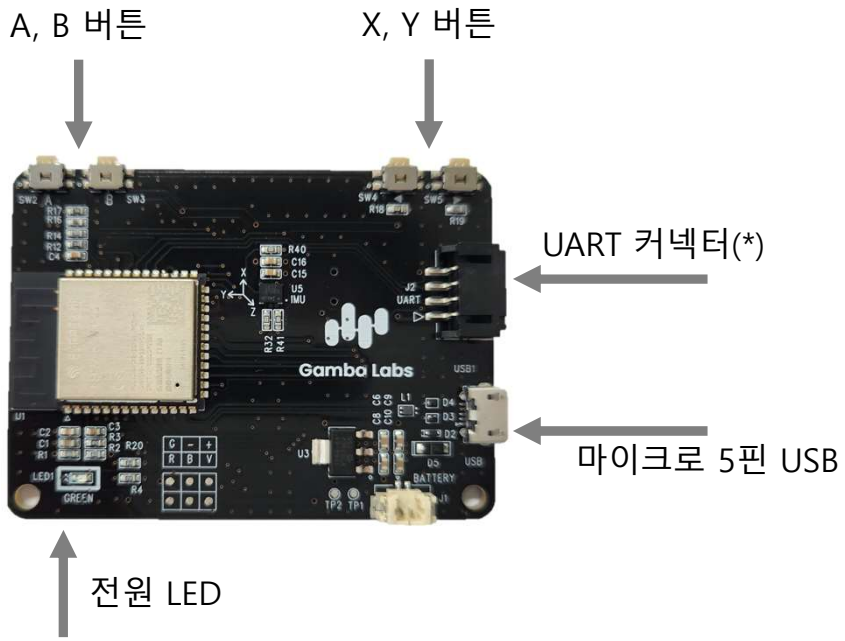
```c
#define UART_BAUD_RATE (115200)
#define UART_CHANNEL (1)
#define UART_BUF_SIZE (512)
#define TIMEOUT (5)
```

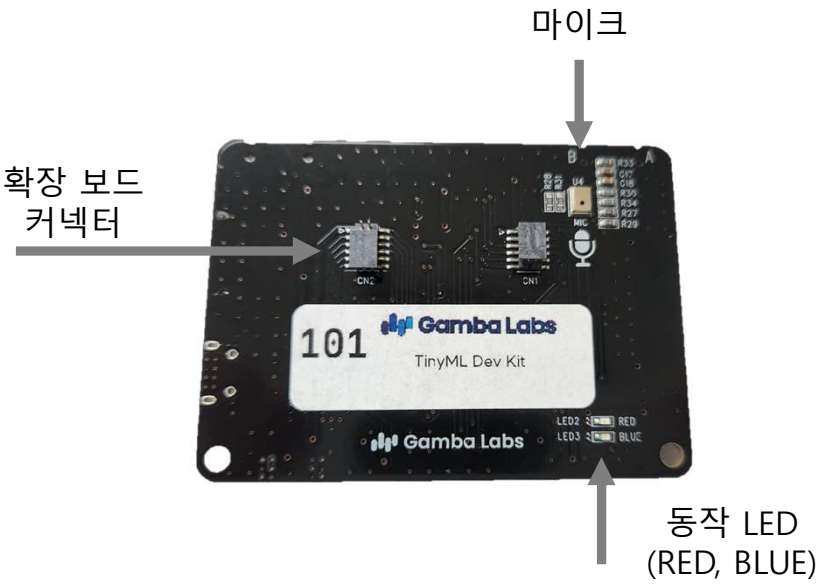
app_uart.h

UART 수신 예시

Appendix: Edu-kit

ESP 기반 보드





(*) UART 커넥터는 ▷표시가 1번 핀이며, 1번부터 3.3V, Tx, Rx, Gnd 핀

Appenddix: Edu-kit

확장 보드

액정 디스플레이

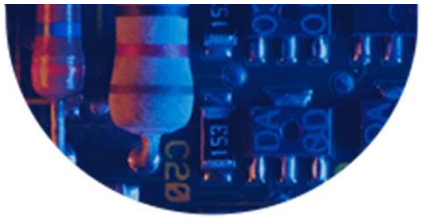
카메라

확장 보드
커넥터

동작 LED
(RED, BLUE)

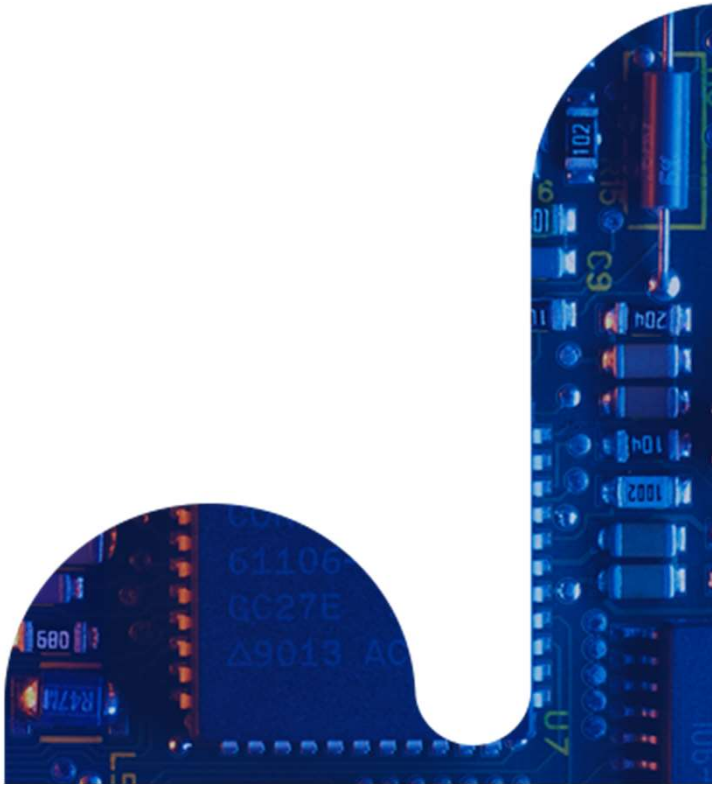
Appendix: Edu-kit

결합된 보드 형태



Gamba Labs

Thank you



Demo : https://gambalabs.ai/demo/
Contact : contact@gambalabs.ai