실험 결과 보고서

13 주차 감바랩스 ESP-IDF

실험날짜: 2024-11-26

실험자:3 조(강태진, 김기윤, 김도완, 임성표)

목차

1. 실험 목적

(1)ESP 기반 MCU 펌웨어 포팅의 이해

(2)ESP-IDF 프레임워크 활용

(3)TensorFlow Lite 및 TensorFlow Lite for micro 에 대한 이해

2. 세부 목표

(1) VS Code ESP-IDF Extension 설치 및 환경설정

(2) TinyWebTrainer 를 통해 학습한 모델을 보드에 탑재

(3) Motion, Seepch, Vision 중 하나를 선택

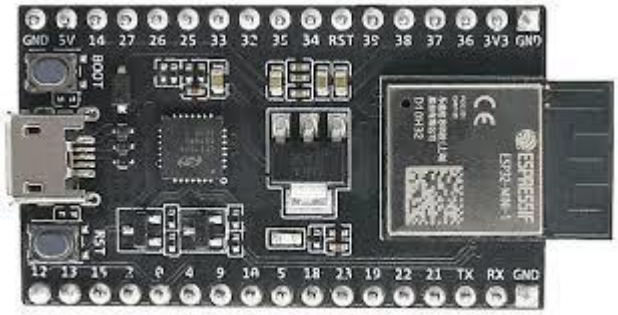
(4) 보드에 탑재된 센서를 통해 예제와 다른 모델을 학습

(5) TinyWebTrainer 학습 페이지 확인 및 보드에서의 동작 검사

(6) TinyWebTrainer 를 통해 학습/변환한 model.c 파일의 모델 데이터를 예제 프로젝트의 model.c 파일에 적용

3. 실험 과정

(1) 배경 지식



ESP32 Board

1) ESP-IDF

- Espressif Systems 에서 개발한 IoT**(**사물인터넷**)** 개발 프레임워크. 주로 ESP32 및 ESP8266 와 같은 Espressif 의 마이크로컨트롤러 칩을 기반으로 한 IoT 애플리케이션을 개발하는 데 사용한다.
- 스마트 홈, 산업용 IoT, 환경 모니터링, 웨어러블 디바이스, 스마트 헬스케어 등 다양한 분야에서 활용하며 저전력 및 실시간 처리가 필요한 애플리케이션에 적합하다.

2) ESP-IDF 의 주요 명령어 및 기능

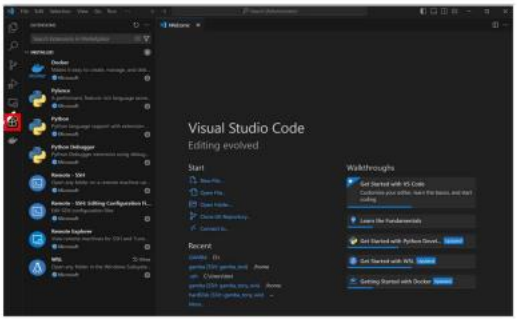
- Idf.py build: 프로젝트 빌드, 전체 프로젝트를 컴파일 및 빌드파일 생성
- Idf.py flash: 설정된 시리얼 포트로 펌웨어를 플래시
- Idf.py monitor: 디바이스의 직렬 모니터링을 실행
- Idf.py clean: 빌드 파일 삭제
- Idf.py fullclean: 모든 빌드 파일과 설정 삭제
- Idf.py menuconfig: 프로젝트의 구성 옵션 조정
- Idf.py set-target esp32: 특정 타겟 칩 지정
- Idf.py create-project project_name: 새 프로잭트 생성
- Idf.py add-dependency packacge_name: 의존성 추가

(2) 실험 진행

1) VS Code ESP-IDF Extension 설치

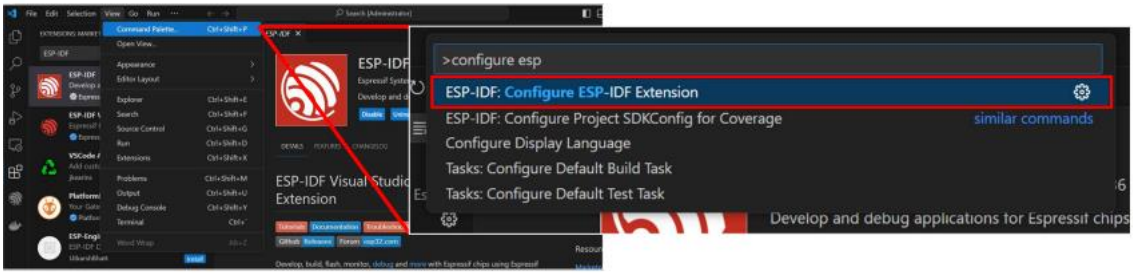
환경설정: VS Code ESP-IDF Extension 설치

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

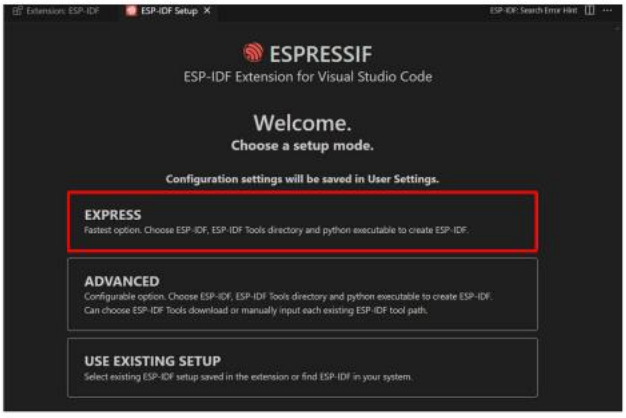


Command Palette에서 configure esp 검색
- Ctrl+Shift+P or F1

ESP-IDF: Configure ESP-IDF Extension 클릭



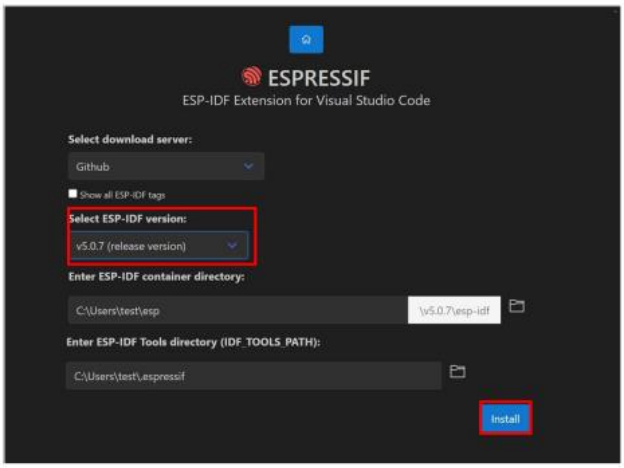
EXPRESS 클릭



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

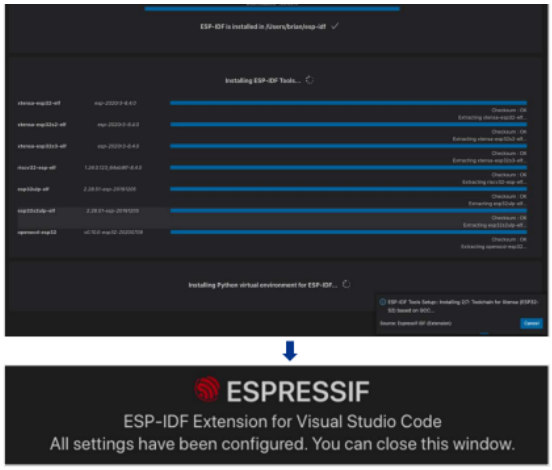
디스크에 15GB의 공간이 필요

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



설치에 몇 분의 시간 소요

설치 완료 시, 아래와 같은 메시지 확인가능

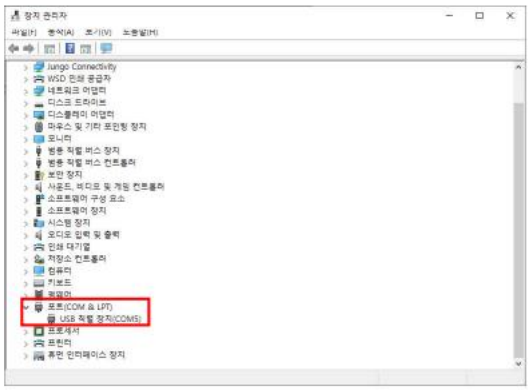


2) 시리얼 포트 환경설정 및 프로젝트 Setup

환경설정: 시리얼 포트 찾기

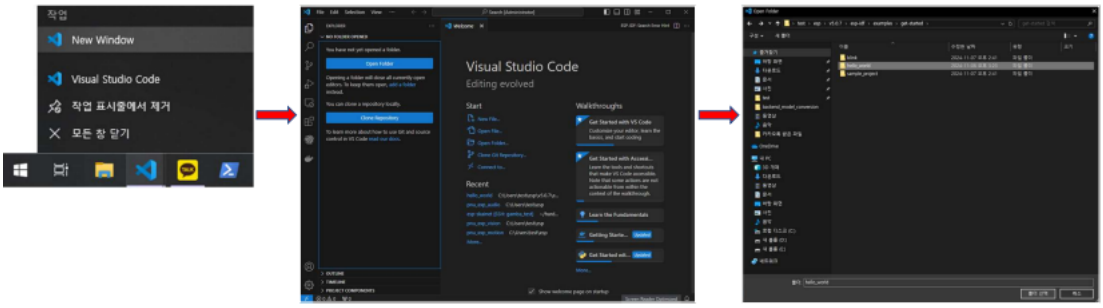
장치 관리자를 통해 연결된 보드의 포트 확인

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

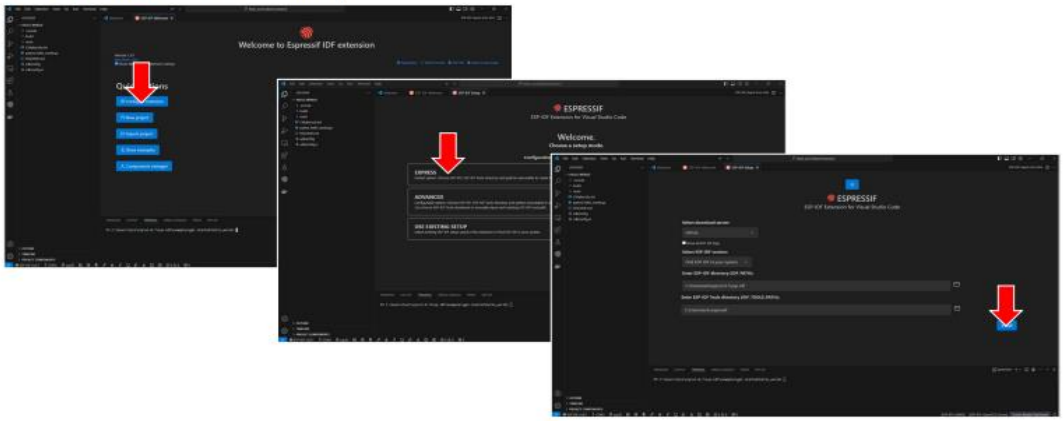


ESP-IDF를 설치했던 경로의 아래 예제 프로젝트 폴더를 열기

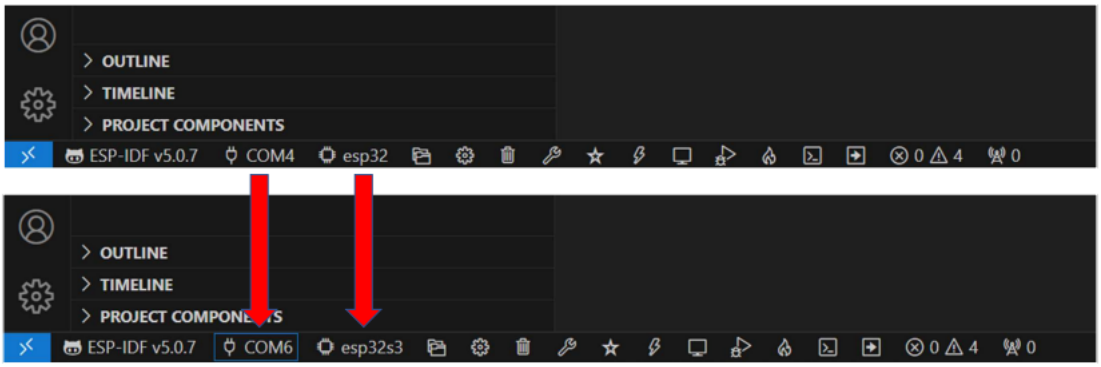
path : $IDF_PATH₩esp-idf₩examples₩get-started₩get-started₩hello_world
PS C:₩Users₩test₩esp₩v5.0.7₩esp-idf₩examples₩get-started₩hello_world>



아래의 순서대로 프로젝트 Setup



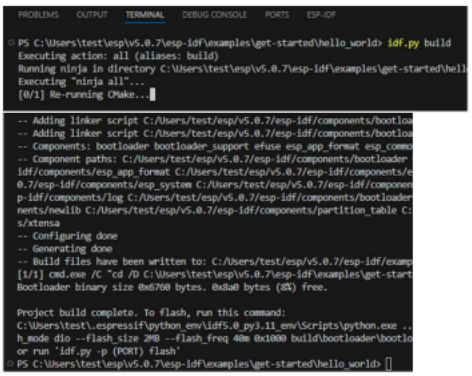
VS Code 하단의 시리얼 포트와 타겟 보드를 알맞게 변경
- 시리얼 포트: 11p에서 확인한 보드의 시리얼 포트 COMx
- 타겟 보드: esp32s3



3) 프로젝트 빌드

Example #0: Build

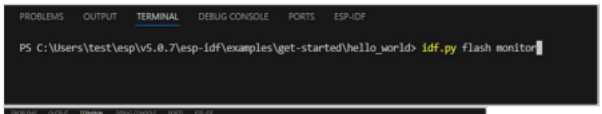
펌웨어 Flash(포팅) 이전에 프로젝트 빌드가 필요 (컴파일 과정 등)
ESP-IDF Terminal에서 idf.py build 명령어 실행



idf.py build

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

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

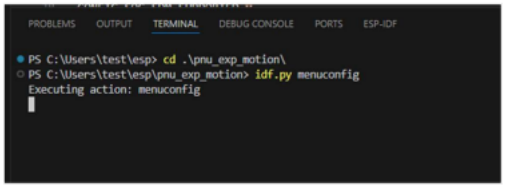
idf.py clean

idf.py fullclean

4) 프로젝트 및 보드 설정 변경

Example #1-3: Assignment

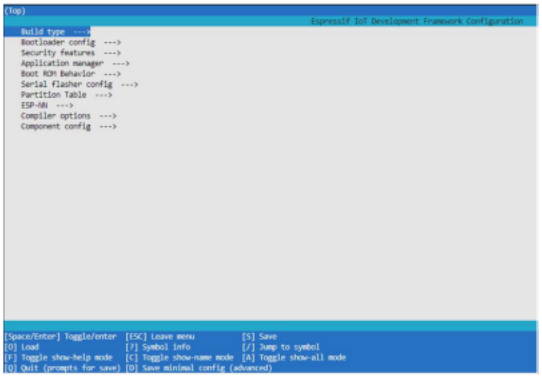
ESP-IDF Terminal 실행 후 pnu_exp_motion 폴더로 이동
idf.py menuconfig 명령어를 통해 프로젝트 및 보드에 대한 설정



idf.py menuconfig

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

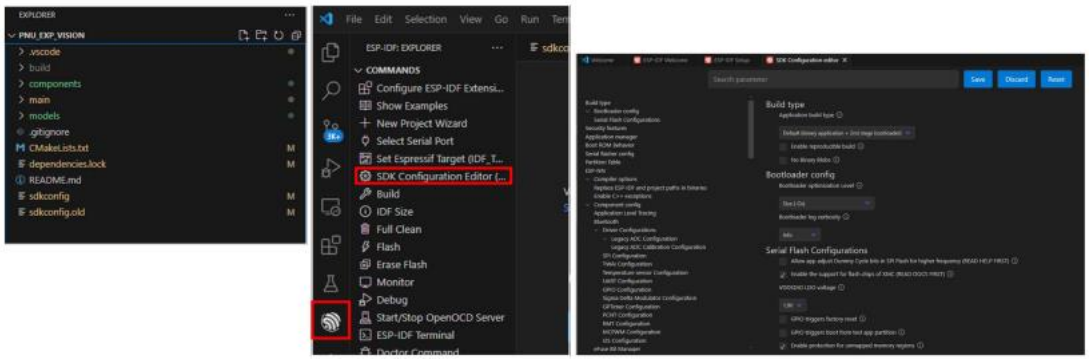
Press [/] and "4MB" [enter] and [space]
칩에 연결된 외부 Flash memory의 크기를 4MB로 설정

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)

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



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]

Press [q] and flash monitor

5) 학습된 예제 모델(motion) model.c 를 보드에 탑재 및 출력결과 확인

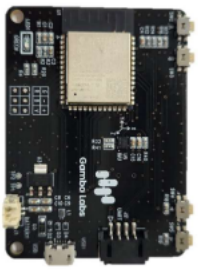
Example #1: pnu_exp_motion

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

기준 상태




```
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]
```

6) 모델 변경

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

7) 펌웨어 초기화

펌웨어 초기화

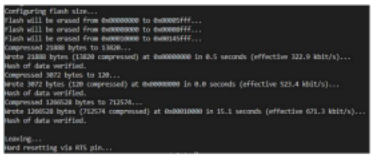
보드에 새로운 펌웨어를 Flash 한 경우 기존의 TinyWebTrainer와 연결이 불가

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

4. 실험 결과

```
0:  0.551699,   1:  0.007109,   2:  0.337459,   3:  0.103733,    [0.114ms]
0:  0.395421,   1:  0.019672,   2:  0.335765,   3:  0.249142,    [0.114ms]
0:  0.058033,   1:  0.058288,   2:  0.826433,   3:  0.057245,    [0.114ms]
0:  0.020751,   1:  0.054468,   2:  0.885257,   3:  0.039524,    [0.114ms]
0:  0.013881,   1:  0.058366,   2:  0.877384,   3:  0.050369,    [0.114ms]
0:  0.140093,   1:  0.015115,   2:  0.819334,   3:  0.025459,    [0.114ms]
0:  0.008314,   1:  0.362613,   2:  0.584105,   3:  0.044969,    [0.114ms]
0:  0.137996,   1:  0.025663,   2:  0.776818,   3:  0.059523,    [0.114ms]
0:  0.014331,   1:  0.036285,   2:  0.925618,   3:  0.023766,    [0.114ms]
0:  0.175094,   1:  0.015774,   2:  0.759977,   3:  0.049155,    [0.114ms]
0:  0.000005,   1:  0.979120,   2:  0.019163,   3:  0.001712,    [0.114ms]
0:  0.721636,   1:  0.001856,   2:  0.262827,   3:  0.013681,    [0.114ms]
0:  0.917494,   1:  0.000779,   2:  0.070400,   3:  0.011327,    [0.114ms]
0:  0.001697,   1:  0.856037,   2:  0.071281,   3:  0.070985,    [0.114ms]
0:  0.060532,   1:  0.100687,   2:  0.701347,   3:  0.137434,    [0.114ms]
0:  0.190583,   1:  0.177858,   2:  0.450603,   3:  0.180956,    [0.114ms]
0:  0.200902,   1:  0.124788,   2:  0.215448,   3:  0.458863,    [0.114ms]
0:  0.006306,   1:  0.728781,   2:  0.144425,   3:  0.120487,    [0.114ms]
0:  0.074643,   1:  0.205830,   2:  0.433405,   3:  0.286122,    [0.114ms]
0:  0.016356,   1:  0.503937,   2:  0.297374,   3:  0.182333,    [0.114ms]
0:  0.125487,   1:  0.099023,   2:  0.605506,   3:  0.169984,    [0.114ms]
0:  0.045376,   1:  0.341657,   2:  0.357603,   3:  0.255364,    [0.114ms]
0:  0.330741,   1:  0.035711,   2:  0.528013,   3:  0.105535,    [0.114ms]
0:  0.054252,   1:  0.113274,   2:  0.721940,   3:  0.110534,    [0.114ms]
0:  0.867864,   1:  0.001250,   2:  0.119987,   3:  0.010899,    [0.114ms]
0:  0.000057,   1:  0.967459,   2:  0.024495,   3:  0.007989,    [0.114ms]
0:  0.677075,   1:  0.004291,   2:  0.278003,   3:  0.040632,    [0.114ms]
0:  0.000004,   1:  0.993747,   2:  0.003990,   3:  0.002259,    [0.114ms]
0:  0.621483,   1:  0.015919,   2:  0.235416,   3:  0.127182,    [0.114ms]
0:  0.051185,   1:  0.066324,   2:  0.797347,   3:  0.085145,    [0.114ms]
0:  0.029974,   1:  0.393890,   2:  0.223139,   3:  0.352997,    [0.114ms]
```

1) UP 모션: 보드를 위로 들었다가 아래로 내리는 모션->1

2) DOWN 모션: 보드를 아래로 내렸다가 올리는 모션 -> 0

3) 결과분석

- 12 주차에서 학습했던 모델은 up, down 모션을 학습했기 때문에 모션에 따른 0 과 1 의 결과값이 1 에 수렴해야 하지만 학습 과정에서 데이터 수집이 부족하여 다소 부정확하거나 낮은 수치로 결과값이 출력됨

- 정확한 값을 출력하거나 높은 정확도를 위해서는 학습 과정에서 보다 많은 데이터를 학습시킬 필요가 있음

5. 참고 문헌

1) 13 주차 강의자료 감바랩스 ESP-IDF