

ESP32 Manual

Tensorflow Lite Micro, Edge Impulse

1.0.0

2021. 05. 04

SunBeen Moon

Contents

1. ESP32 보드 소개	4
2. 초기 세팅	8
2.1. ESPRESSIF	8
2.2. TFLM	9
2.3. Edge Impulse	10
2.4. Issues	14
3. Tensorflow Lite Micro	15
3.1. Overview of loading Example in TFLM	15
3.2. Examples	15
3.2.1. Hello World	15
3.2.2. Micro Speech	16
3.2.3. Person Detection	17
3.2.4. Door Bell	17
3.3. Issues	19
4. Edge Impulse	20
4.1. Overview of loading Example in Edge Impulse	20
4.2. Tests	22
4.2.1. Static Buffer	22

4.2.2. Edge Impulse Official Github	22
4.2.3. Others	23
4.3. Issues	24
 5. ESP32's Custom Project	 28
5.1. ESP32-WHO	28
5.1.1. Introduction	28
5.1.2. 설치 방법	29
5.1.3. Example code	32
5.1.3.1. Face detection with Command Line	32
5.1.3.2. Camera Web-Server	34
5.2. ESP32-FACE	38
5.3. Issues	39

1. ESP32 보드 소개

ESP32 보드는 MCU(Wi-Fi, Bluetooth, 통신 등) 중 하나이다.

ESP32 사에서 나오는 보드 군은 'ESP32-C3 Series DevKits', 'ESP32-S2 Series DevKits', 'ESP32 Series DevKits', 'ESP8266 Series DevKits'와 '다른 IoTDevKits'로 나뉘어 진다.

소유하고 있는 2 가지 보드는 모두 'ESP32 Series DevKits'군에 속한다. ESPRESSIF 툴킷(ESP-IDF)을 설치하거나 할 때, ESP32 로 검색하여서 찾아야 한다. 보드의 정식 명칭은 'ESP32-DEVKITC-32D', 'ESP-EYE'이다. 각 보드에 대한 설명은 아래와 같다.

(ESPRESSIF 본사 사이트) <https://www.espressif.com/en>

(ESP Forum) <https://esp32.com/viewforum.php?f=23>

(Wikipedia) <https://en.wikipedia.org/wiki/ESP32>

(note) ESP Forum 이 활성화되어 있다. 영어와 중국어 버전 두가지가 존재하며 질문을 올리고 나서는, 검열을 거친 후에 질문을 올릴 수 있다. 질문에 대한 답변을 달아도, 검열을 거쳐야 한다.

✓ ESP32-DEVKITC-32D

ESP32-DEVKITC-32D 보드는 ESP32-WROOM-32D 칩을 내장하고 있다. 칩에 대한 상세한 정보(MCU 특성을 설명하는)는 아래 Fig 1 과 같다. 칩에 대한 메모리 특징은 Table 1 과 같다.

Flash	4MB
External Flash	11MB + 248KB
SRAM	4MB
CPU	Dual-core Xtensa 32-bit LX6 MCU
Internal ROM	448KB
Internal SRAM	520KB

Table 1. Memory of ESP32-DevKITC-32D

Categories	Items	Specifications
Certification	RF Certification	FCC/CE-RED/IC/TELEC/KCC/SRRC/NCC
	Wi-Fi Certification	Wi-Fi Alliance
	Bluetooth certification	BQB
	Green Certification	REACH/RoHS
Test	Reliability	HTOL/HTSL/uHAST/TCT/ESD
Wi-Fi	Protocols	802.11 b/g/n (802.11n up to 150 Mbps)
		A-MPDU and A-MSDU aggregation and 0.4 μ s guard interval support
	Frequency range	2.4 GHz ~ 2.5 GHz
Bluetooth	Protocols	Bluetooth v4.2 BR/EDR and BLE specification
	Radio	NZIF receiver with -97 dBm sensitivity
		Class-1, class-2 and class-3 transmitter
		AFH
	Audio	CVSD and SBC
Hardware	Module interfaces	SD card, UART, SPI, SDIO, I ² C, LED PWM, Motor PWM, I ² S, IR, pulse counter, GPIO, capacitive touch sensor, ADC, DAC, Two-Wire Automotive Interface (TWAI [®] , compatible with ISO11898-1)
	On-chip sensor	Hall sensor
	Integrated crystal	40 MHz crystal
	Integrated SPI flash ¹	4 MB
	Operating voltage/Power supply	3.0 V ~ 3.6 V
	Operating current	Average: 80 mA
	Minimum current delivered by power supply	500 mA
	Recommended operating temperature range ²	-40 °C ~ +85 °C
	Moisture sensitivity level (MSL)	Level 3

Fig 1. Specification about ESP32-WROOM-32D

ESP32-WROOM-32D 에 대한 정보를 더 얻고 싶다면 아래 링크를 참조하면 된다.

(ESP32-WROOM-32D 가 설명된 링크) <https://docs.espressif.com/projects/esp-idf/en/release-v4.0/hw-reference/modules-and-boards.html#esp-modules-and-boards-esp32-wroom-32d-and-u>

(ESPRESSIF 에서 ESP32-WROOM-32D 데이터 시트를 얻을 수 있는 창 : 'ESP32-WROOM-32D & ESP32-WROOM-32U Datasheet'를 다운로드 하면 된다)

https://www.espressif.com/en/support/documents/technical-documents?keys=&field_type_tid%5B%5D=266

(ESP32-WROOM-32D 가 자세히 기술된 블로그) <https://m.blog.naver.com/cimygy/221677513320>

✓ ESP-EYE

ESP-EYE 보드는 ESP32-D0WD 칩을 내장하고 있다. 칩에 대한 상세한 정보는 fig 2 와 같다. 칩에 대한 메모리 특징은 Table 2 와 같다.

Flash	4MB
External Flash	8MB
SRAM	4MB
CPU	Dual-core Xtensa 32-bit LX6 MCU
Internal ROM	448KB
Internal SRAM	520KB

Table 2. Memory of ESP-D0WD

Category	Target	Start Address	End Address	Size
Embedded Memory	Internal ROM 0	0x4000_0000	0x4005_FFFF	384 KB
	Internal ROM 1	0x3FF9_0000	0x3FF9_FFFF	64 KB
	Internal SRAM 0	0x4007_0000	0x4009_FFFF	192 KB
	Internal SRAM 1	0x3FFE_0000	0x3FFF_FFFF	128 KB
	Internal SRAM 2	0x400A_0000	0x400B_FFFF	200 KB
	RTC FAST Memory	0x3FFA_E000	0x3FFD_FFFF	8 KB
	RTC SLOW Memory	0x400C_0000	0x400C_1FFF	8 KB
External Memory	External Flash	0x5000_0000	0x5000_1FFF	4 MB
	External RAM	0x3F40_0000	0x3F7F_FFFF	11 MB+248 KB
	External RAM	0x400C_2000	0x40BF_FFFF	4 MB
Peripheral	DPort Register	0x3F80_0000	0x3FBF_FFFF	4 KB
	AES Accelerator	0x3FF0_0000	0x3FF0_0FFF	4 KB
	RSA Accelerator	0x3FF0_1000	0x3FF0_1FFF	4 KB
	SHA Accelerator	0x3FF0_2000	0x3FF0_2FFF	4 KB
	Secure Boot	0x3FF0_3000	0x3FF0_3FFF	4 KB
	Secure Boot	0x3FF0_4000	0x3FF0_4FFF	4 KB
	Cache MMU Table	0x3FF0_5000	0x3FF0_5FFF	16 KB
	PID Controller	0x3FF1_0000	0x3FF1_3FFF	4 KB
	UART0	0x3FF1_F000	0x3FF1_FFFF	4 KB
	UART0	0x3FF4_0000	0x3FF4_0FFF	4 KB
	SPI1	0x3FF4_1000	0x3FF4_1FFF	4 KB
	SPI0	0x3FF4_2000	0x3FF4_2FFF	4 KB
	GPIO	0x3FF4_3000	0x3FF4_3FFF	4 KB
	GPIO	0x3FF4_4000	0x3FF4_4FFF	4 KB
	RTC	0x3FF4_5000	0x3FF4_5FFF	4 KB
	IO MUX	0x3FF4_6000	0x3FF4_6FFF	4 KB
	SDIO Slave	0x3FF4_7000	0x3FF4_7FFF	4 KB
	UDMA1	0x3FF4_8000	0x3FF4_8FFF	4 KB
	UDMA1	0x3FF4_9000	0x3FF4_9FFF	4 KB
	I2S0	0x3FF4_A000	0x3FF4_AFFF	4 KB
	UART1	0x3FF4_B000	0x3FF4_BFFF	4 KB
	I2C0	0x3FF4_C000	0x3FF4_CFFF	4 KB
	UDMA0	0x3FF4_D000	0x3FF4_DFFF	4 KB
	SDIO Slave	0x3FF4_E000	0x3FF4_EFFF	4 KB
	RMT	0x3FF4_F000	0x3FF4_FFFF	4 KB
	PCNT	0x3FF5_0000	0x3FF5_0FFF	4 KB
	SDIO Slave	0x3FF5_1000	0x3FF5_1FFF	4 KB
	LED PWM	0x3FF5_2000	0x3FF5_2FFF	4 KB
	eFuse Controller	0x3FF5_3000	0x3FF5_3FFF	4 KB
	Flash Encryption	0x3FF5_4000	0x3FF5_4FFF	4 KB
	PWM0	0x3FF5_5000	0x3FF5_5FFF	4 KB
	TIMG0	0x3FF5_6000	0x3FF5_6FFF	4 KB
	TIMG1	0x3FF5_7000	0x3FF5_7FFF	4 KB
	SPI2	0x3FF5_8000	0x3FF5_8FFF	4 KB
	SPI3	0x3FF5_9000	0x3FF5_9FFF	4 KB

Fig 2. Specifcaton about ESP-D0WD

ESP-D0WD 에 대한 정보를 더 얻고 싶다면 아래 링크를 참조하면 된다.

(ESP-EYE Overview) https://github.com/espressif/esp-who/blob/master/docs/en/get-started/ESP-EYE_Getting_Started_Guide.md

(ESPRESSIF 에서 ESP-D0WD 에 대한 데이터시트를 얻을 수 있는 창 : 'ESP32 Datasheet'를 다운로드하면 된다) https://www.espressif.com/en/support/documents/technical-documents?keys=&field_type_tid%5B%5D=492

2. 초기세팅

TFLM 을 구동하기 위해서, ESPRESSIF 에서 제공하는 환경을 구축해주어야 하며, Edge Impulse 를 구동하기 위해서는 Arduino 를 설치해야 한다.

2.1 ESPRESSIF

ESP32 를 구동하기 위해서는 해당 보드를 구동할 수 있는 환경을 구축해주어야 한다. 환경을 구축하기 위해서는 ESPRESSIF 에서 나온, 툴킷(ESP-IDF)을 활용해야 한다. 툴을 통해 빌드된 'idf.py' 파일로, 메뉴를 띄우고 빌드하고 보드에 deploy 하는 데에도 위 파일이 필요하다. 아래 사이트(ESPRESSIF tutorial 사이트)에 적힌 순서대로 진행하면 된다.

(ESPRESSIF tutorial 사이트 v4.0) <https://docs.espressif.com/projects/esp-idf/en/release-v4.0/get-started/index.html>

(ESP-IDF Official Github) <https://github.com/espressif/esp-idf>

(note) TFLM 에서 사용한 버전이 ESP-IDF version 4.0 이다. 따라서, 위 링크의 v4.0 을 설치하여야 한다.

(note) 원하는 OS 환경에 맞춰서 빌드를 한 후, 'Hello World' 예제까지 확인해 보기를 권장한다.

```

1 (186) cpu_start: App cpu up.
1 (213) cpu_start: Pro cpu start user code
1 (213) cpu_start: cpu freq: 160000000
1 (213) cpu_start: Application Information:
1 (218) cpu_start: Project name:      hello-world
1 (223) cpu_start: App version:      1
1 (228) cpu_start: Compile time:     Apr  5 2021 12:18:08
1 (234) cpu_start: ELF file SHA256:  30a4d415cc538628...
1 (240) cpu_start: ESP-IDF:         v4.4-dev-744-g1cb31e509
1 (246) heap_init: Initializing. RAM available for dynamic allocation:
1 (253) heap_init: At 3FFAE6E0 len 00001920 (6 KiB): DRAM
1 (259) heap_init: At 3FFB3258 len 0002CDAB (179 KiB): DRAM
1 (266) heap_init: At 3FFE0440 len 00003AE0 (14 KiB): D/IRAM
1 (272) heap_init: At 3FFE4350 len 0001BCB0 (111 KiB): D/IRAM
1 (278) heap_init: At 400BADF4 len 0001520C (84 KiB): IRAM
1 (286) spi_flash: detected chip: generic
1 (289) spi_flash: flash io: dio
W (293) spi_flash: Detected size(4096k) larger than the size in the binary image header(2048k). Using the size in the binary image header.
1 (307) cpu_start: Starting scheduler on PRO CPU.
1 (0) cpu_start: Starting scheduler on APP CPU.
Hello world!
This is esp32 chip with 2 CPU core(s), WiFi/BT/BLE, silicon revision 1, 2MB external flash
Minimum free heap size: 291240 bytes
Restarting in 10 seconds...
Restarting in 9 seconds...
Restarting in 8 seconds...
Restarting in 7 seconds...
Restarting in 6 seconds...
Restarting in 5 seconds...
Restarting in 4 seconds...
Restarting in 3 seconds...
Restarting in 2 seconds...
Restarting in 1 seconds...

```

Fig 3. Hello World 예제 화면

(note) 위 Hello World 예제는 ESPRESSIF 환경 자체에 귀속되어 있는 예제이다. 즉, TFLM 예제와 다른 방법으로 build 된다.

(note) idf.py 는 flash, monitor, build 등 사용할 때마다 설치된 폴더에서 불러와 새롭게 빌드해주어야 한다. 해당 명령어는 아래와 같다.

`./%esp-idf path%/esp/esp-idf/export.sh`

2.2 TFLM

Tensorflow 버전은 2 가지가 존재한다. Tensorflow official github 가 있고, official github 을 ESPRESSIF 사에서 fork 해서 customize 한 github 가 있다. 아래 Table 3 에서 확인할 수 있듯이, 두 github 에서 구동할 수 있는 예제가 다르다.

예제 이름	TFLM official	ESPRESSIF
Hello World	O	O
Micro Speech	O	O
Person Detection	X	O
Door Bell	X	O

Table 3. TFLM official github vs ESPRESSIF fork github

모든 예제는 1 절에서 설명한 보드에서 작동이 가능하다. 예제는 'ESP32-DevKit'와 'ESP32-EYE'에서 작동이 가능하다. 하지만, 'ESP32-DevKit'에서 Person Detection 을 구동하기 위해서는 카메라 모듈이 필요하다. 아래 링크를 따라 설치하면 된다.

(Tensorflow official github) <https://github.com/tensorflow/tensorflow>

(ESPRESSIF forked TF) <https://github.com/espressif/tensorflow>

설치할 때, 체크해야 할 사항은 아래와 같다.

1. The IDF_PATH environment variable is set
2. idf.py and Xtensa-esp32 tools (e.g. xtensa-esp32-elf-gcc) are in \$PATH
3. esp32-camera should be downloaded in components/ dir of example as explained in Building the example(below)
 - ESPRESSIF 환경 설정을 할 때, '.install.sh' 명령어를 입력하면 2 번이 자동적으로 충족된다.
 - ESPRESSIF 환경 설정을 할 때, '.export.sh' 명령어를 입력하면 idf.py 가 설치되며 idf.py 를 사용할 때마다 이 명령어를 다시 불러와 idf.py 를 빌드 해주어야 한다.
 - 3 번의 esp32-camera 는 따로 다운로드 받아야 한다.(다운로드 받는 경로는, TFLM readme 파일 뒤 쪽에 기술되어 있다.)

(note) ESPRESSIF 깃허브에서 다운로드 받으면, fetch 할 필요없이 구동 가능하다.

(note) ESPRESSIF 사에서 TF 을 따로 fork 해서 customize 한 이유는, TF 는 너무 빠르게 변화하고 있다. 그 속도를 따라가기 힘들다. 예를 들어, Person Detection 예제에서는 int8 만을 지원하기로 TF 가 update 되었다. 하지만, ESP 사는 int8 로만 구동하기 어렵기 때문에, 수정이 필요하다.

2.2 Edge Impulse

Edge Impulse 를 ESP32 보드 위에서 구동하기 위해서는 Arduino 를 설치해야 한다. 아래 순서에 따라 설치하면 된다.

1. Arduino 설치

ESP32 보드를 실행하기 위해 'Arduino IDE 1.8.13'를 설치해야 한다. ESP 제조사 github 에서 소개하는 Arduino 설치 순서대로 설치를 진행한다.

(Arduino 다운로드 웹 페이지) <https://www.arduino.cc/en/software>

(ESP 제조사 github 에서 소개하는 Arduino 설치법) <https://github.com/espressif/arduino-esp32/blob/master/docs/arduino-ide/windows.md>

(ESPRESSIF 의 official Arduino-esp32 github) <https://github.com/espressif/arduino-esp32>

(note) Edge Impulse 사 document 'On your Arduino' 맨 아래 Trouble Shooting 에 'Note: This issue is reportedly fixed in Arduino CLI v0.14 and in the Arduino IDE 1.9.'라고 나와있지만, 'Arduino IDE 1.8.13(당시 최신버전)을 설치해야 한다.

'파일→환경설정'에서 Fig 4 에서 말한 부분에 .json 파일을 해 준다. 아래 링크를 참조한다.

(Board Manager 추가 링크) https://github.com/espressif/arduino-esp32/blob/master/docs/arduino-ide/boards_manager.md

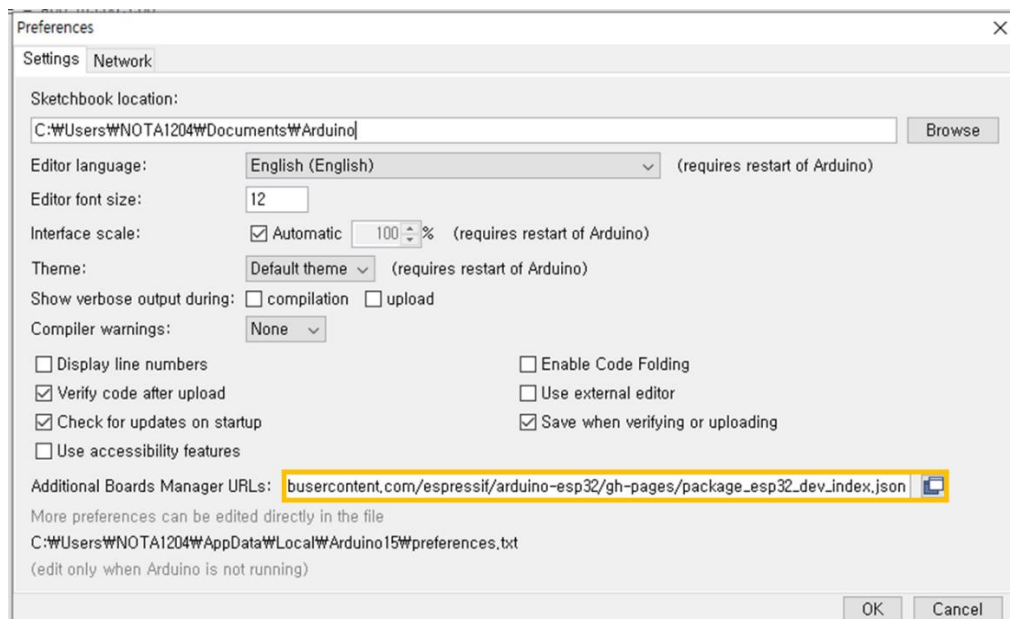


Fig 4. Board Manager 추가 URL

'툴→보드→보드 매니저'에서 ESP32 보드를 검색하여 아래 Fig 처럼 설치한다.



Fig 5. 보드 매니저에서 ESP32 를 설치한 모습

(note) 다양한 버전이 존재하지만 '1.0.4'을 설치한다. Edge impulse 를 통해 esp32 를 구동한 개발자들이 '1.0.4'버전을 설치했기 때문이다.

2. Arduino Window Error Fix

Edge Impulse 모델을 Arduino 를 통해서 구동하면 파일 이름이 너무 길어서 build 할 수 없다는 error 메시지가 나온다. 이를 해결하기 위해 아래와 같은 방법으로 설정을 해야 한다.

(Edge Impulse 'On your Arduino' Document) <https://docs.edgeimpulse.com/docs/running-your-impulse-arduino>

위 document 맨 아래 쪽을 보면 ESP32 에 맞는 .txt 파일을 다운로드 받을 수 있다.

If updating your Arduino IDE does not work you can overcome this issue by downloading the platform.local.txt below for your target:

- [mbed platform.local.txt](#). Copy this file under the Arduino mbed directory, i.e:
C:\Users\MYUSER\AppData\Local\Arduino15\packages\arduino\hardware\mbed\1.1.4\
- [SAMD21 platform.local.txt](#). Copy this file under the Arduino SAMD directory, i.e:
C:\Users\MYUSER\AppData\Local\Arduino15\packages\arduino\hardware\samd\1.8.9\
- [SAMD51 \(Adafruit\) platform.local.txt](#). Copy this file under the Arduino Adafruit SAMD directory, i.e:
C:\Users\MYUSER\AppData\Local\Arduino15\packages\adafruit\hardware\samd\1.6.3\
- [ESP32 platform.local.txt](#). Copy this file under the Arduino ESP32 directory, i.e:
C:\Users\MYUSER\AppData\Local\Arduino15\packages\esp32\hardware\esp32\1.0.4\
- [STM32 platform.local.txt](#). Copy this file under the Arduino STM32 directory, i.e:
C:\Users\MYUSER\AppData\Local\Arduino15\packages\esp32\hardware\stm32\1.9.0\

Fig 6. esp32 에 맞는 .txt 파일 다운로드

파일을 다운로드 해서 'platform.local.txt'라고 저장한다.

'C:\Users\사용자이름\AppData\Local\Arduino15\packages\esp32\hardware\esp32\1.0.4'에 저장한다. 실행화면은 아래와 같다.

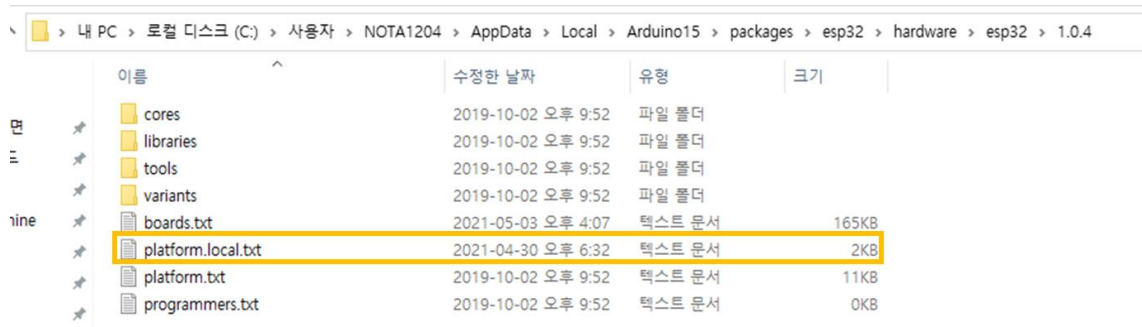


Fig 7. 'platform.local.txt' 다운로드하여 저장한 모습

두 과정을 거쳐서 파일을 다운로드 받고 저장하면 window 에서 파일이름이 길어 build 가 될 수 없다는 오류를 해결할 수 있다.

3. 각 ESP32 보드 위에서 Arduino 가 제대로 구동하는 지 확인

보드 이름	보드 매니저에서 선택해야 하는 이름
ESP32-DevKITC-32D	ESP32 Wrover Module
ESP-EYE	DOIT ESP32 DEVKIT V1

Table 4. ESP board – Arduino

위 Table 4 처럼 Arduino 를 setting 하고 올바르게 Arduino 가 보드 위에서 구동하는 지 간단한 예제를 통해 확인한다. '파일→예제→WiFi→WiFiScan' 파일을 구동시켜서 올바르게 deploy 되는 지 확인한다.

4. Edge Impulse 모델 생성

Edge Impulse 에서 모델을 생성하는 과정은 이전 소개된 edge impulse 내용 확인하고 싶다면, 이전 보고서('Himax WE-1_final report (ver1) 210420.pdf')에 35 페이지를 확인하면 된다.

(Himax WE-1 final report) <https://www.notion.so/notaai/Himax-WE-1-TFLM-Edge-Impulse-b0bf7df6bf664de5ab122363dfe88ff3>

(note) 'Image Classification' 모델을 만들 때, 48*48 크기로 전처리를 해야 한다.

(note) model 사이즈가 너무 크면 보드 위에 올라가지 않음으로, 'MobileNetV1 0.1' 혹은 'MobileNetV1 0.2'를 선택하여 transfer learning 해야 한다.

5. Arduino 에서 구동하기

'Deployment'에서 'Arduino library'로 '.zip'파일을 다운로드 받는다. 아래 링크에서 설명한 대로, '스케치→라이브러리 포함하기→.zip 라이브러리 추가'로 압축을 풀지 않고 .zip 파일을 업로드 한다. 업로드한 파일은 '파일→예제→사용자 지정 라이브러리'에서 확인할 수 있다.

(Edge Impulse 에서 소개하는 Arduino 설치 Tutorial) <https://docs.edgeimpulse.com/docs/running-your-impulse-arduino>

사용자가 직접 추가한 라이브러리는 'C:\#사용자\#사용자이름\#문서\#Arduino\#libraries'에서 확인할 수 있다.

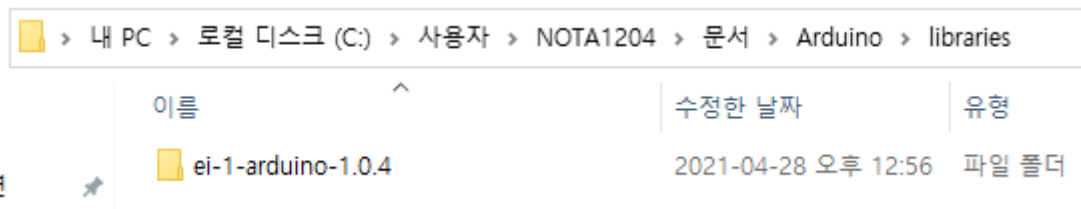


Fig 8. 사용자가 직접 추가한 라이브러리 위치

2.3 Issues

<Issue 1> Tensorflow 에서 Person Detection 은 구동되지 않는다. build 까지 되고 flash 가 되지 않았다.(필자가 시도한 곳까지 구동하고 싶다면)기존 Tensorflow 만 다운로드 받은 후 build 하면 실행이 되지 않음으로, 아래 명령어를 입력하여 실행하여야 한다.

'git cherry-pick f543fld'

3. Tensorflow Lite Micro

3.1 Overview of Loading Example in TFLM

각 예제의 readme 파일을 보면서 진행해도 충분하지만, 예제 구동의 패턴이 정형화되어 있으므로 아래에서 간략하게 설명한다.

1. 미리 ESPRESSIF 환경을 구축해 둔다.
2. `make -f tensorflow/lite/micro/tools/make/Makefile TARGET=Example 이름`
3. `cd tensorflow/lite/micro/tools/make/gen/esp_xtensa-esp32/prj/Example 이름/esp-idf`
4. `git clone https://github.com/espressif/esp32-camera.git %esp-idf_path%components/esp32-camera`
5. `./%esp-idf_path%/esp/esp-idf/export.sh`
6. `ldf.py`
*idf.py 를 빌드해준다.
7. `idf.py --port /dev/ttyUSB0 flash monitor`

3.2 Examples

Table 3 에서 설명한 대로, 예제가 구동되는 깃허브를 체크해야 한다. 아래 예제는 모두 ESPRESSIF forked Tensorflow github 에서 구동한 예제 파일이다. Tensorflow Official github 에서도 구동함을 확인하였으나, 결과 창을 담지 않았다.

실상, Tensorflow Official Github 에서 ESPRESSIF 사가 fork 했기 때문에, Person Detection 과 Door Bell 이외의 예제들은 내용이 모두 동일하다.

3.2.1 Hello World

아래 링크를 따라 구동하면 문제없이 실행할 수 있다. Hello World 예제는 두 github 모두 동일함으로 원하는 github 로 취사선택해서 실행하면 된다.

(Tensorflow Official Github)

https://github.com/tensorflow/tensorflow/tree/master/tensorflow/lite/micro/examples/hello_world#deploy-to-esp32

(ESPRESSIF forked TF)

https://github.com/espressif/tensorflow/tree/master/tensorflow/lite/micro/examples/hello_world#deploy-to-esp32

<Result>

```
x_value: 1.2566366*2^1, y_value: 1.0674761*2^-1
x_value: 1.4137159*2^1, y_value: 1.8977352*2^-3
x_value: 1.5707957*2^1, y_value: 1.0844199*2^-7
x_value: 1.7278753*2^1, y_value: -1.2199725*2^-2
x_value: 1.8849551*2^1, y_value: -1.0674761*2^-1
x_value: 1.0210171*2^2, y_value: -1.5588537*2^-1
x_value: 1.0995567*2^2, y_value: -1.9316229*2^-1
x_value: 1.1780966*2^2, y_value: -1.1098361*2^0
x_value: 1.2566366*2^2, y_value: -1.965511*2^-1
```

Fig 9. Hello World 결과 창

<Analysis>

예상된 결과인 사인파가 정상적으로 출력되었다.

3.2.2 Micro Speech

아래 링크를 따라 구동하면 문제없이 실행할 수 있다. Micro Speech 예제는 두 github 모두 동일함으로 원하는 github 로 취사선택해서 실행하면 된다.

(Tensorflow Official Github)

https://github.com/tensorflow/tensorflow/tree/master/tensorflow/lite/micro/examples/micro_speech#deploy-to-esp32

(ESPRESSIF forked TF)

https://github.com/espressif/tensorflow/tree/master/tensorflow/lite/micro/examples/micro_speech#deploy-to-esp32

<Result>

```
Heard unknown (203) @69700ms
Heard no (220) @83900ms
```

Fig 10. Micro Speech 결과 창

<Analysis>

예상된 결과대로 Yes, No, Unknown 이 감지되었다. 다른 소리에 비해 'yes'가 감지가 부정확하게 되는 경향이 있다.

3.2.3 Person Detection

아래 링크를 따라 구동하면 문제없이 실행할 수 있다. Person Detection 예제는 ESPRESSIF forked TF 에서만 구동이 가능하다.

(ESPRESSIF forked TF)

https://github.com/espressif/tensorflow/tree/master/tensorflow/lite/micro/examples/person_detection#running-on-esp32

<Result>

```
person score:71 no person score 235
Image Captured

person score:252 no person score 43
Image Captured
```

Fig 11. Person Detection Result

<Analysis>

Person Detection 이 잘 잘 되지만, threshold 가 따로 지정되어 있지 않아 사용자가 알아서 score 를 보고 분별해야 한다. 예를 들면, person score 가 71 이고 no person score 가 235 임으로 사람이 없는 경우이다. Person Score 가 252 이고 no person score 가 43 임으로 사람이 있는 경우이다.

3.2.1 Door Bell

아래 링크를 따라 구동하면 문제없이 실행할 수 있다. Door Bell 예제는 ESPRESSIF forked TF 에서만 구동이 가능하다. 이 예제는 ESP-EYE 로만 구동할 수 있다.

(Door Bell 예제 tutorial 및 구동영상) <https://blog.tensorflow.org/2020/08/announcing-tensorflow-lite-micro-esp32.html>

(ESPRESSIF forked TF)

https://github.com/espressif/tensorflow/tree/master/tensorflow/lite/micro/examples/doorbell_camera

(note) SMTP Configuration 을 설정하는 곳에서 아래와 같이 설정하고, gmail 에서 less secure apps 를 설정해 주어야 한다. Sender email 은 메일을 보내는 곳의 주소이고 비밀번호이다. Recipient email 은 메일 받는 곳의 주소이다.

```
(smtp.googlemail.com) Mail Server
(587) Mail port number
(shxk.tjsqls@gmail.com) Sender email
( ) Sender password
(sunbeen.moon@nota.ai) Recipient email
```

Fig 12. 이메일 설정 화면

<Result>

```
Camera Initialized
Image Captured
Image Captured
Image Captured
Image Captured
Image Captured
Image Captured
Image Captured
Image Captured
Image Captured
Image Captured
person detected
```

Fig 13. Door Bell 예제 결과 창

Build 가 되고 나서, 계속해서 image 를 캡처하면서 감지한다. 감지하다가 사람이 나타난 경우, person detected 를 창의 띄우고 모델을 재 시작한다.

<Analysis>

사람을 잘 감지하고, 구동이 잘 되나 메모리 문제로 인해, 메일이 보내지지 않는다. 이에 대한 사항은 issue 칸에 상세히 기재한다.

3.3 Issues

<Issue 1> Person Detection 예제에서 사람이 있는 경우와 없는 경우는 분명하게 구별해 주지만, 아예 사람이 없는 경우(Ex. 카메라가 천장을 바라볼 경우)에는 score 의 차이가 크지 않다. 사람이 등장하고 없을 때는 급격하게 person score 와 no person score 의 점수 차이가 생기지만, 오랜 시간 사람이 없을 경우 점수차가 거의 나지 않는다.

예를 들면, Person Score : 151, No Person Score : 183 식이다.

<Issue 2> Tensorflow 예제에서 코드를 수정하고자 한다면(필자의 경우, 사람이 감지되었을 경우 threshold 을 결정해 주어 화면에 띄우고자 하였다) 수정이 쉽지 않다. Edge 기기 위에 deploy 하기 위해서는 낮은 단까지 내려가야 하기 때문이다. 이와 관련해 해결법은 추후에 업로드 할 예정이다.

<Issue 3> Door Bell 예제에서 메일까지 보내는 일련의 과정이 진행되어야 하는데, 메일이 메모리 부족 문제로 보내지지 않는다. 메일은 보내지지 않지만, 사람이 detect 되고 나서 모델일 새로 시작되는 시점까지 진행된다.

```
I (16205) smtp_client: Connecting to smtp.googlemail.com:587...
I (16455) smtp_client: Connected.
E (17995) smtp_client: mbedtls_ssl_handshake returned -0x7f00

E (17995) smtp_client: Error in sending email
E (17995) smtp_client: Last error was: -0x7f00 - SSL - Memory allocation failed
E (18005) tf_responder: Failed to send the email, returned FFFFFFFF
```

Fig 14. 메모리 오류가 나 메일이 송부되지 않는 오류 창

4. Edge Impulse

Edge Impulse 관한 Introduction 은 'Himax WE-1 : TFLM & Edge Impulse'와 'Edge Impulse : Object Detection'에 이미 소개되어 있으므로 생략한다.

4.1 Overview of Loading Model in Edge Impulse

Camera 를 이용한 Image Classification 을 기준으로 설명을 진행한다.

- ✓ Edge Impulse 에서 다운로드 받은 예제 그대로 사용하는 방법

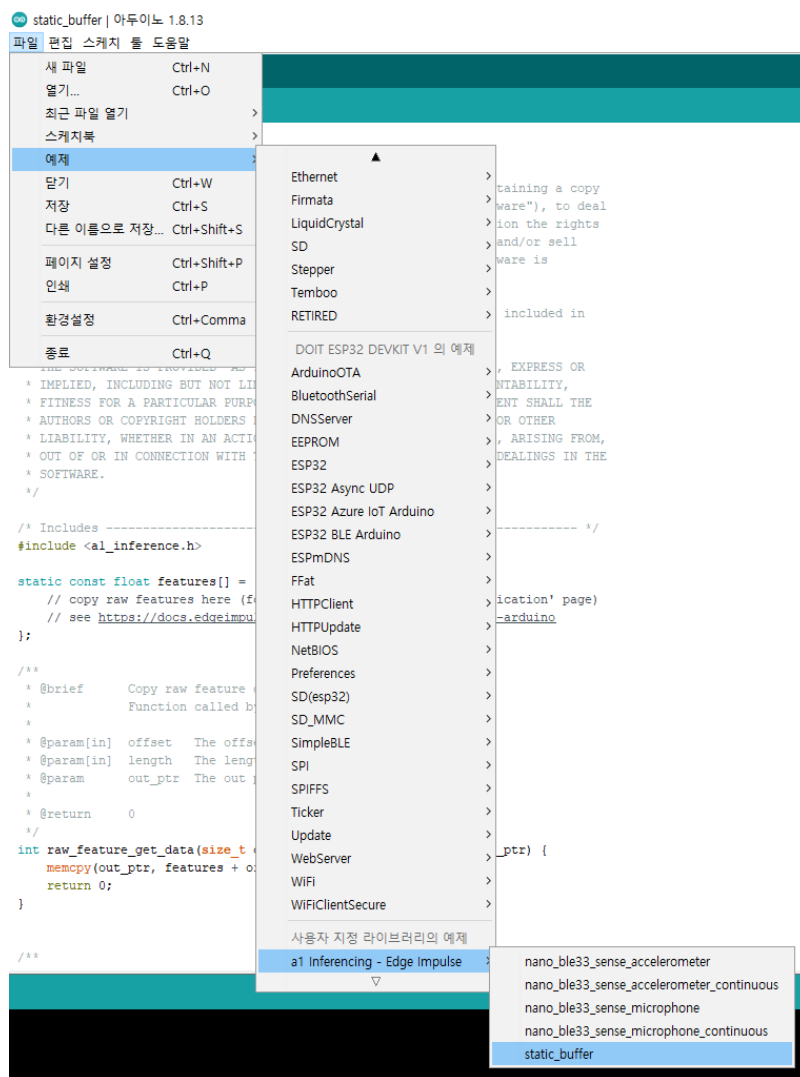


Fig 15. Edge Impulse 'static_buffer' 예제 선택 화면

'파일'→'예제'→'사용자 지정 라이브러리 예제'→'static_buffer'을 선택한다. 실시간 모델 구동이 아니라, 이미 지정되어 있는 사진을 테스트할 수 있도록 구성되어 있는 예제이다. 아래 코드를 수정하여야 한다.

```
static const float features[] = {
    // copy raw features here (for example from the 'Live classification' page)
    // see https://docs.edgeimpulse.com/docs/running-your-impulse-arduino
};
```

Edge Impulse project 에서 'Live Classification'에서 한 사진을 선정하여 classification 을 진행한다. Image Classification 에서 나온 feature 를 이용해야 한다고 edge impulse forum 에서 소개하지만 정상적으로 구동되지 않았다. 이와 관련된 issue 는 'issues' 칸에서 소개한다.

- ✓ Edge Impulse 에서 다운로드 받은 모델을 활용하여 구동하는 방법

Edge Impulse 에서 다운로드 받은 모델은 아래 Fig 처럼 구성된다.

이름	수정한 날짜	유형	크기
examples	2021-05-03 오전 10:12	파일 폴더	
src	2021-05-03 오전 10:12	파일 폴더	
library.properties	2021-05-03 오전 10:12	PROPERTIES 파일	1KB

Fig 16. Edge Impulse 다운로드 모델 구성

위 모델을 사용하고 싶다면, 'src' 폴더와 'library.properties'를 복사하여 원하는 폴더에 붙여 넣는다. 다운로드한 모델 이름은 src 폴더 안에 .h 파일 이름으로 확인할 수 있다. 아래 예시 Fig 처럼 원하는 곳에 붙여 넣으면 된다.

이름	수정한 날짜	유형	크기
src	2021-04-28 오후 3:17	파일 폴더	
.DS_Store	2021-04-16 오후 11:55	DS_STORE 파일	7KB
Basic-Image-Classification.ino	2021-05-03 오후 4:23	Arduino file	11KB
camera_index.h	2021-04-16 오후 11:55	C/C++ Header	52KB
camera_pins.h	2021-04-16 오후 11:55	C/C++ Header	4KB
library.properties	2021-04-28 오후 12:56	PROPERTIES 파일	1KB

Fig 17. 원하는 파일로 복사 붙여 넣은 예시

4.2 Tests

4.2.1 Static Buffer

정상적으로 구동되지 않음으로 issue 칸에서 소개한다.

4.2.3 Edge Impulse

아래 과정은 edge impulse official ESP32 Cam Example github 에서 다운로드 받은 두 가지 모델을 실행한 예제이다. 링크는 아래와 같다.

(edge Impulse official ESP32 Cam Example github) <https://github.com/edgeimpulse/example-esp32-cam>

(Edge Impulse 사와 연관된 esp32-arduino 예제 github) <https://github.com/v12345vtm/esp32-cam-webserver-arduino-simplified-arduino-html>

위 링크에서 파일을 다운로드 받으면 두 가지 예제를 실행할 수 있다.

1. Edge impulse 를 통해 빌드한 모델의 src 를 해당 폴더에 복사한다.
2. 다운로드한 모델의 이름으로 include 파일을 수정한다.

```
// Modify the following line according to your project name
// Do not forget to import the library using "Sketch">"Include
Library">"Add .ZIP Library..."
//default 로 있었던 model
//#include <esp32-cam_image-classification_inference.h>
//edge impulse 로 새롭게 다운로드 받은 모델
#include <a1_inference.h>
```

3. 해당 보드에 맞게 Camera Model 을 변경한다.

```
// Select camera model
//#define CAMERA_MODEL_WROVER_KIT // Has PSRAM
#define CAMERA_MODEL_ESP_EYE // Has PSRAM
//#define CAMERA_MODEL_M5STACK_PSRAM // Has PSRAM
//#define CAMERA_MODEL_M5STACK_V2_PSRAM // M5Camera version B Has PSRAM
//#define CAMERA_MODEL_M5STACK_WIDE // Has PSRAM
//#define CAMERA_MODEL_M5STACK_ESP32CAM // No PSRAM
//#define CAMERA_MODEL_AI_THINKER // Has PSRAM
//#define CAMERA_MODEL_TTGO_T_JOURNAL // No PSRAM
```

4. WiFi와 비밀번호를 수정한다

```
const char* ssid = "nota 3F AP2";  
const char* password = "*****";
```

5. *FRAMESIZE_240*240*을 *FRAMESIZE_QVGA*로 변경한다.

✓ Basic-Image-Classification

Build도 되고, ESP-EYE 보드 위에 deploy되나 error가 발생한다. 뒤 issues 칸에서 소개한다.

✓ Advanced-Image-Classification

Build도 되고, ESP-EYE 보드 위에 deploy되나 error가 발생한다. 뒤 issues 칸에서 소개한다.

4.2.4 Others

참고할 만한 타 사이트의 예시를 소개한다.

(TinyML Edge Impulse Image Classification blog) <https://www.survivingwithandroid.com/tinymml-esp32-cam-edge-image-classification-with-edge-impulse/>

(ESP-CAM Face Recognition for Home Automation) <https://lastminuteengineers.com/esp32-arduino-ide-tutorial/>

4.2 Issues

<Issue 1> static_buffer 예제에 'Live Classification'에서 나온 'Raw features'로 input 을 넣을 경우.

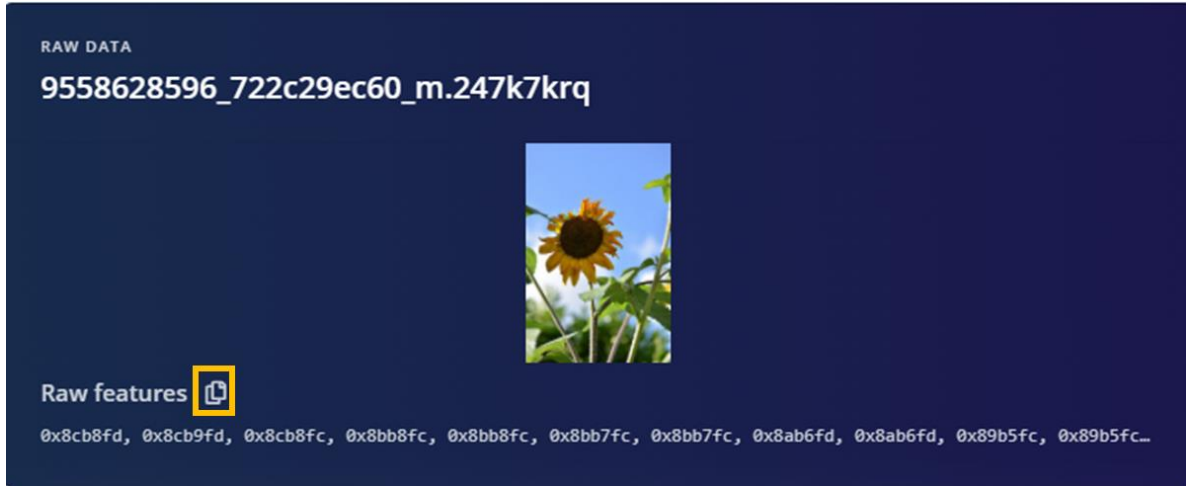


Fig 18. Raw features 를 선택한 모습

노랑색 박스 친 부분을 클릭하여 아래 Fig 처럼 코드에 삽입한 후, 보드 위에 deploy 한다.

```
static const float features[] = {
    0x8cb8fd, 0x8cb9fd, 0x8cb8fc, 0x8bb8fc, 0x8bb8fc, 0x8bb7fc, 0x8bb7fc, 0x8ab6fd,
    // copy raw features here (for example from the 'Live classification' page)
    // see https://docs.edgeimpulse.com/docs/running-your-impulse-arduino
};
```

Fig 19. Raw features 를 삽입한 모습

실제 deploy 시 아래 Fig 처럼 오류가 난다.

```
Rebooting...
ets Jun  9 2016 00:22:57

rst:0xc (SW_CPU_RESET),boot:0x13 (SPI_FAST_FLASH_BOOT)
configsip: 0, SPIWP:0xee
clk_drv:0x00,q_drv:0x00,d_drv:0x00,cs0_drv:0x00,hd_drv:0x00,wp_drv:0x00
mode:DIO, clock div:1
load:0x3fff0018,len:4
load:0x3fff001c,len:1044
load:0x40078000,len:8896
load:0x40080400,len:5816
entry 0x400806ac
Edge Impulse Inferencing Demo
Edge Impulse standalone inferencing (Arduino)
Guru Meditation Error: Core 1 panic'ed (InstrFetchProhibited). Exception was unhandled.
Core 1 register dump:
PC      : 0x00000000  PS      : 0x00060930  A0      : 0x800e953c  A1      : 0x3ffb1db0
A2      : 0x3ffc3e2c  A3      : 0x00000000  A4      : 0x3ffc3d2c  A5      : 0x00000009
A6      : 0x00000000  A7      : 0x3ffc272c  A8      : 0x800ecc1e  A9      : 0x00000002
A10     : 0x3ffc3e2c  A11     : 0x3f41bbf1  A12     : 0x00000001  A13     : 0x00000001
A14     : 0x00000002  A15     : 0x3f4198c0  SAR     : 0x00000007  EXCCAUSE: 0x00000014
EXCVADDR: 0x00000000  LBEG    : 0x4000c2e0  LEND    : 0x4000c2f6  LCOUNT  : 0x00000000

Backtrace: 0x00000000:0x3ffb1db0 0x400e9539:0x3ffb1e40 0x400e8a63:0x3ffb1e60 0x400e8fa1:0x3ffb1e90 0x400e9177:0x3ffb1f10 0x400e920d:0x3ffb1f50 0x400e675d:0x3ffb1fb0 0x40088215:0x3ffb1fd0
```

Fig 20. Raw features error 발생 창

<Issue 3> Edge Impulse ESP32 github Basic-Image-Classification 문제.

Compile 되고 ESP-EYE 보드 위에 Deploy 되나, 아래 Fig 와 같은 error 발생한다. Forum 을 통해 해결하는 중이다.

```
ets Jun  8 2016 00:22:57

rst:0x1 (POWERON_RESET),boot:0x13 (SPI_FAST_FLASH_BOOT)
configsip: 0, SPIWP:0xee
clk_drv:0x00,q_drv:0x00,d_drv:0x00,cs0_drv:0x00,hd_drv:0x00,wp_drv:0x00
mode:DIO, clock div:1
load:0x3fff0018,len:4
load:0x3fff001c,len:1044
load:0x40078000,len:8896
load:0x40080400,len:5816
entry 0x400806ac
```

Fig 24. Basic Image Classification 에러

<Issue 4> Edge Impulse ESP32 github Advanced-Image-Classification 문제.

Compile 되고 ESP-EYE 보드 위에 Deploy 되나, 아래 Fig 와 같은 error 발생한다. Forum 을 통해 해결하는 중이다.

```
COM3

ets Jun  8 2016 00:22:57

rst:0x1 (POWERON_RESET),boot:0x13 (SPI_FAST_FLASH_BOOT)
configsip: 0, SPIWP:0xee
clk_drv:0x00,q_drv:0x00,d_drv:0x00,cs0_drv:0x00,hd_drv:0x00,wp_drv:0x00
mode:DIO, clock div:1
load:0x3fff0018,len:4
load:0x3fff001c,len:1044
load:0x40078000,len:8896
load:0x40080400,len:5816
entry 0x400806ac
```

Fig 25. Advanced-Image-Classification 에러

<Issue 3> 메모리가 부족하여 모델이 올라가지 않는 이슈.

완전히 해결되지 않았으나, 해결책을 제시하는 youtube 와 링크.

'C:\사용자\사용자이름\AppData\Local\Arduino15\packages\esp32\hardware\esp32\1.0.4\boards.txt' 파일을 연다.

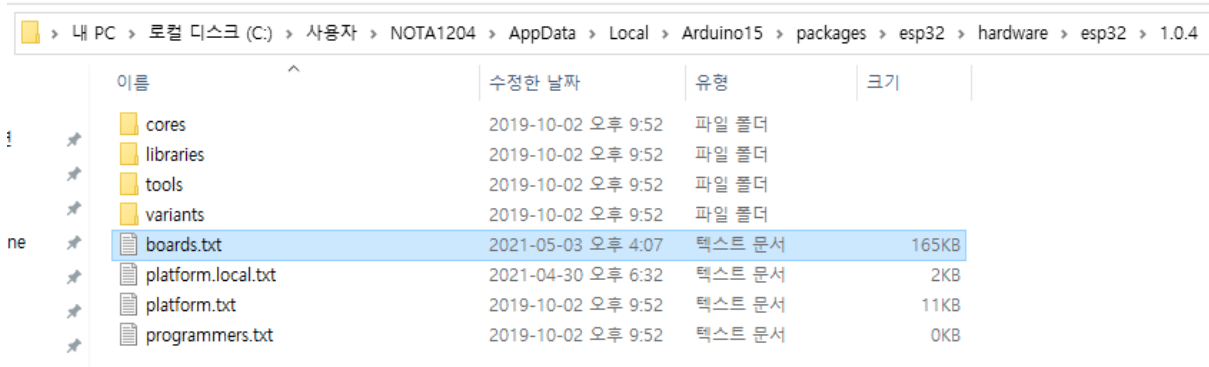


Fig 26. 수정해야 하는 boards.txt 위치

'doit'보드를 검색해서 아래에 붙여 넣는다.

```
esp32doit-devkit-v1.menu.PartitionScheme.default=Default
esp32doit-devkit-v1.menu.PartitionScheme.default.build.partitions=default
esp32doit-devkit-v1.menu.PartitionScheme.minimal=Minimal (2MB FLASH)
esp32doit-devkit-v1.menu.PartitionScheme.minimal.build.partitions=minimal
esp32doit-devkit-v1.menu.PartitionScheme.no_ota=No OTA (Large APP)
esp32doit-devkit-v1.menu.PartitionScheme.no_ota.build.partitions=no_ota
esp32doit-devkit-v1.menu.PartitionScheme.no_ota.upload.maximum_size=2097152
esp32doit-devkit-v1.menu.PartitionScheme.min_spiffs=Minimal SPIFFS (Large APPS with OTA)
esp32doit-devkit-v1.menu.PartitionScheme.min_spiffs.build.partitions=min_spiffs
esp32doit-devkit-v1.menu.PartitionScheme.min_spiffs.upload.maximum_size=1966080
esp32doit-devkit-v1.menu.PartitionScheme.fatflash=16M Fat
esp32doit-devkit-v1.menu.PartitionScheme.fatflash.build.partitions=ffat
```

(Youtube) <https://www.youtube.com/watch?v=5VoXNloOwZE>

(파일 링크) <https://www.xtronical.com/esp32memoryproblems/>

4. ESP32's Custom Project

5.1 ESP-Who

5.1.1 Introduction

ESP-WHO 는 ESP 사에서 만든 face detection 과 face recognition 을 체크 해보기 위해서 만든 platform 이다. ESP32 Chip 에 바로 deploy 할 수 있도록 제작되어 있으며, 아래와 같이 다양한 예제를 지원하고 있다. 위 앱을 구동하기 위해서는 external 4MB Flash 가 필요로 하며, ESP32-Wroom(칩)와 ESP-EYE 모두 충족한다. 카메라가 내장되어 있지 않은 보드에서 구동하고 싶다면, 추가적인 카메라가 필요하다. 또한, GPIO 연결이 필요하다.

(ESP-WHO Official Github) <https://github.com/espressif/esp-who>

ESP-WHO 실행 가능 한 예제 목록은 아래와 같다.

- Camera web server
- Detection with web
- Face detection with command line
- Face recognition solution
- Face recognition with command line
- Hand detection with command line
- Handpose estimation web

이 중에서 두 가지 예제를 실행하여 5.2.3 절에 설명해 두었다.

5.1.2 설치방법

ESP-WHO 를 실행하기 위해서, 아래 블로그 글을 참조하면 된다.(아래 블로그에서 실행한 예제는, 5.2.3.2 절 Camera Web Server 예제이다) 블로그를 예시로 링크해 둔 이유는, 구체적인 tutorial 이 github 에 친절하게 나와 있지 않다. 블로그를 참조하여 예제를 실행시키는 과정을 습득한다고 생각하면 된다.

하지만, 아래 블로그 글은 linux 환경이 아니라 window 환경에서 Mingw 를 이용해서 구동하는 방법을 과정을 설명하고 있다. 아래 블로그 글처럼 꼭 window 환경에서 구동하지 않아도 되며, linux 환경에서 구동해도 된다.

(ESP-WHO 를 ESP 사에서 배포한 깃허브로 실행한 블로그) <https://robotzero.one/esp32-camera-module/>

(ESP-WHO 를 내장되어 있는 아두이노 예제를 통해 실행한 블로그) <https://robotzero.one/esp32-cam-arduino-ide/>

(note) 아두이노에서 ESP32 를 구동하는 내용은, 2.2 절에서 확인하면 된다.

환경 설정(1. ESPRESSIF 툴킷이 설치되어 있다. 2. ESP-WHO 깃허브가 다운로드 되어 있다.)이 완료되어 있으며, 보드에 카메라가 연결되어 있다는 가정하에 예제를 구동하는 과정은 아래와 같다.

```
cd ~/esp-who/examples/single_chip/%예제 이름%/
```

먼저 esp-who 속에 예제 폴더 속으로 이동한다.

(note) 앞서 언급했듯이, linux 버전에서 실행하기 위해서는 idf.py 를 계속해서 불러와 재생성 해주어야 한다.

```
idf.py menuconfig (linux ver)
make menuconfig (window ver)
```

ESP-WHO 깃허브만 보고는 menuconfig 에서 설정해 주어야 하는 부분을 파악하기 어렵다. 블로그 내용을 참조하여 간단하게 사용자가 설정할 수 있는 부분을 정리해 보았다.

(1 번과 2 번은 필수 체크이며, 3 번과 4 번에 예제에 따라 WiFi 에 연결하여 web 으로 카메라를 확인하고 싶을 경우에는 3 번과 4 번도 체크한다.

1. Serial flasher config → Default serial port (Change to the port number shown in Device Manager – i.e. COM3)

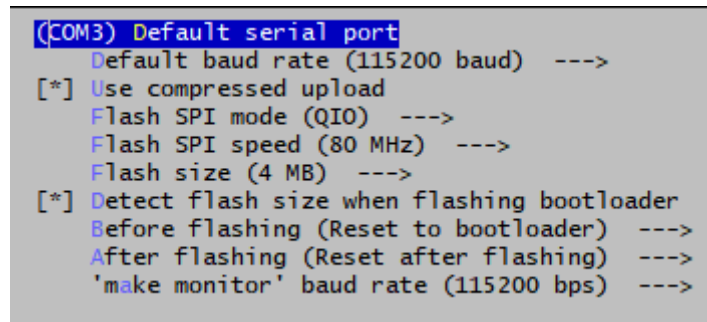


Fig 27. Com Number 를 설정해 주는 화면

2. Component config → ESP32-specific → SPI Ram config → Type of SPI RAM chip (Select Auto-detect)

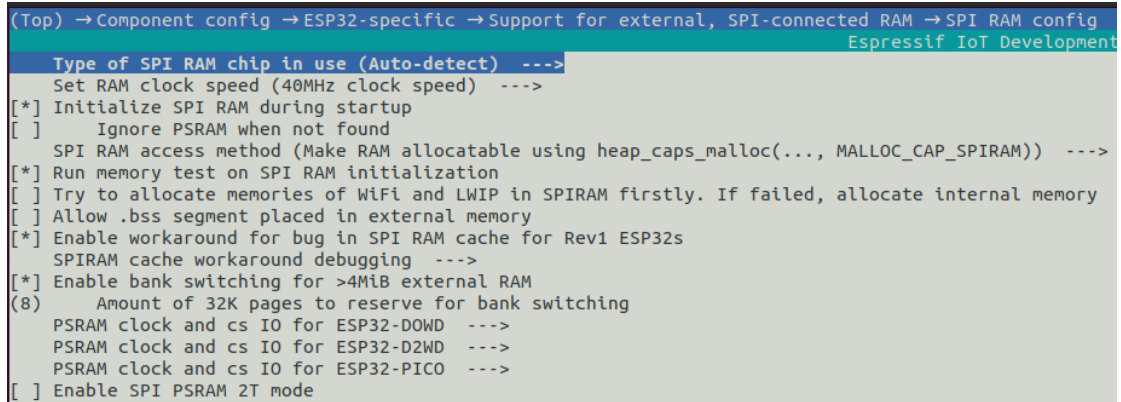


Fig 28. RAM 사용량을 자동적으로 찾아내도록 설정하는 창

3. Camera Web Server -> WiFi Settings -> (Add your WiFi SSID and Password)

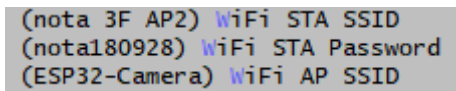


Fig 29. WiFi 를 설정해 주는 화면

4. Camera Web Server -> Camera Pins -> Select Camera Pinout -> (Select 해당 보드의 카메라)

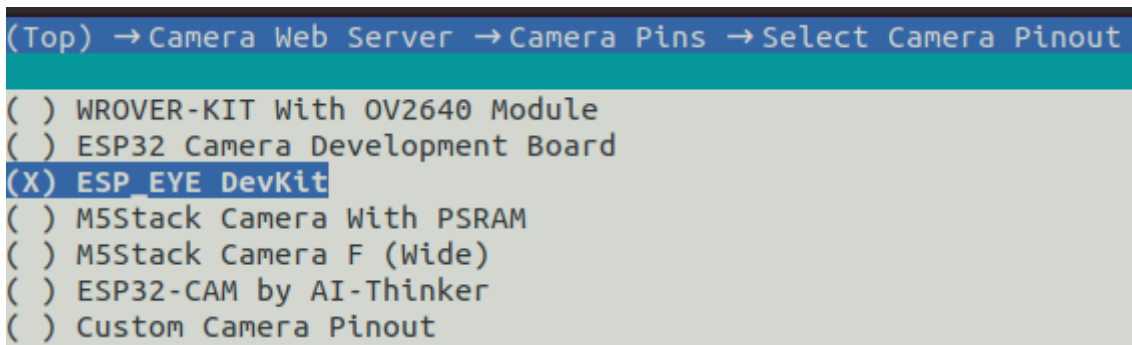


Fig 30. Camera Pin 을 설정해 주는 창

```
idf.py build (linux ver)
make build (window ver)
```

프로젝트를 빌드 해준다.

```
idf.py flash (linux ver)
make flash (window ver)
```

설정이 완료되었으며 보드 위에 flash 해준다.

```
idf.py --port /dev/ttyUSB0 monitor (linux ver)
make monitor (window ver)
```

monitor 창을 띄워서 모델이 구동되는 것을 web 사이트에서 확인할 수 있다. (예제에 따라 상의하다)

```
I (2269) event: sta ip: 192.168.1.84, mask: 255.255.255.0, gw: 192.168.1.1
I (2269) camera wifi: got ip:192.168.1.84
```

Fig 31. Make monitor

Web server 를 지원하는 예제의 경우 ip 를 확인하여 web server 에 연결할 수 있다. http: // %ip% 로 들어가면, Fig 와 같이 화면을 확인할 수 있다.

(note) web server 를 지원해 주지 않는 예제는, make monitor 를 실행하면 5.3.2.1 절에서 확인할 수 있듯이, Command Line 에서 모델의 결과를 보여준다.

5.1.3 Example Code

5.1.3.1 Face Detection with Command Line

<Installation>

위 예제는 아래 링크를 따라서 진행하면 된다. 아래 링크 설명이 충분하지 않음으로, 환경 설정(1. ESPRESSIF 툴킷이 설치되어 있다. 2. ESP-WHO 깃허브가 다운로드 되어 있다.)을 완료한 이후에 5.2.2 절에서 설명한 순서대로 진행하면 된다.

(ESP-WHO official github)

https://github.com/espressif/esp-who/tree/master/examples/single_chip/face_detection_with_command_line

<Result>

```
I (4152) app_process: Detection time consumption: 205ms
I (4192) app_process: Get one frame in 34 ms.
I (4422) app_process: Detection time consumption: 237ms
I (4432) app_process: DETECTED: 1

I (4452) app_process: Get one frame in 21 ms.
I (4702) app_process: Detection time consumption: 249ms
I (4702) app_process: DETECTED: 2

I (4732) app_process: Get one frame in 30 ms.
I (4992) app_process: Detection time consumption: 262ms
I (4992) app_process: DETECTED: 3

I (5032) app_process: Get one frame in 36 ms.
I (5262) app_process: Detection time consumption: 230ms
I (5262) app_process: DETECTED: 4
```

Fig 32. Face Detection 결과 창

<Analysis>

카메라를 통해 사람의 얼굴을 찾게 될 경우에, DETECTED 숫자가 하나씩 올라간다. 한 프레임을 찍는데 걸리는 시간과, inference time(=Detection time consumption)을 알 수 있다. 사람 얼굴을 찾는 모델의 성능은 우수하다. 하지만, 사람을 detect 하는 게 새로운 사람을 찾을 때마다 하는 것이 아니라 한 프레임 안에 사람이 있을 때마다 숫자를 센다. 예를 들어, 카메라 안에 같은 단 한 사람이 계속 찍혀도 1, 2, 3 순차적으로 Detected 되는 숫자가 올라간다.

5.1.3.1 Camera Web-Server

<Installation>

Official github 의 설명이 충분하지 않음으로 5.2.2 절과 블로그를 참조하면 문제없이 구동할 수 있다.

(ESP-WHO Official Github)

https://github.com/espressif/esp-who/tree/master/examples/single_chip/camera_web_server

(ESP-WHO 를 ESP 사에서 배포한 깃허브로 실행한 블로그) <https://robotzero.one/esp32-camera-module/>

<Progress>

모델에 두 가지 옵션이 있는 데, 'Face Recognition'은 따로 설정해 줄 필요 없다. 아래 Table 은 'Face Detection' 과정을 설명한다.





1		3	
2		4	

Table 5. Face Recognition 과정

위 Table 5 은 Face Recognition 과정을 설명해 준다. 1 번 input 을 결정해 주고, 2 번으로 넘어가 얼굴을 인식한다. 얼굴 인식을 하여 이전에 학습된 이미지인지 아닌 지 판별하며, 학습되지 않은 이미지라면 ID number 를 붙여준다. 3 번으로 넘어가, 학습되지 않은 이미지라면 4 장의 사진을 찍어 학습을 진행한다. 학습이 진행된 후, 4 번처럼 다시 사진을 보여줬을 때 'Hello Subject #num'으로 띄우는 것을 알 수 있다.

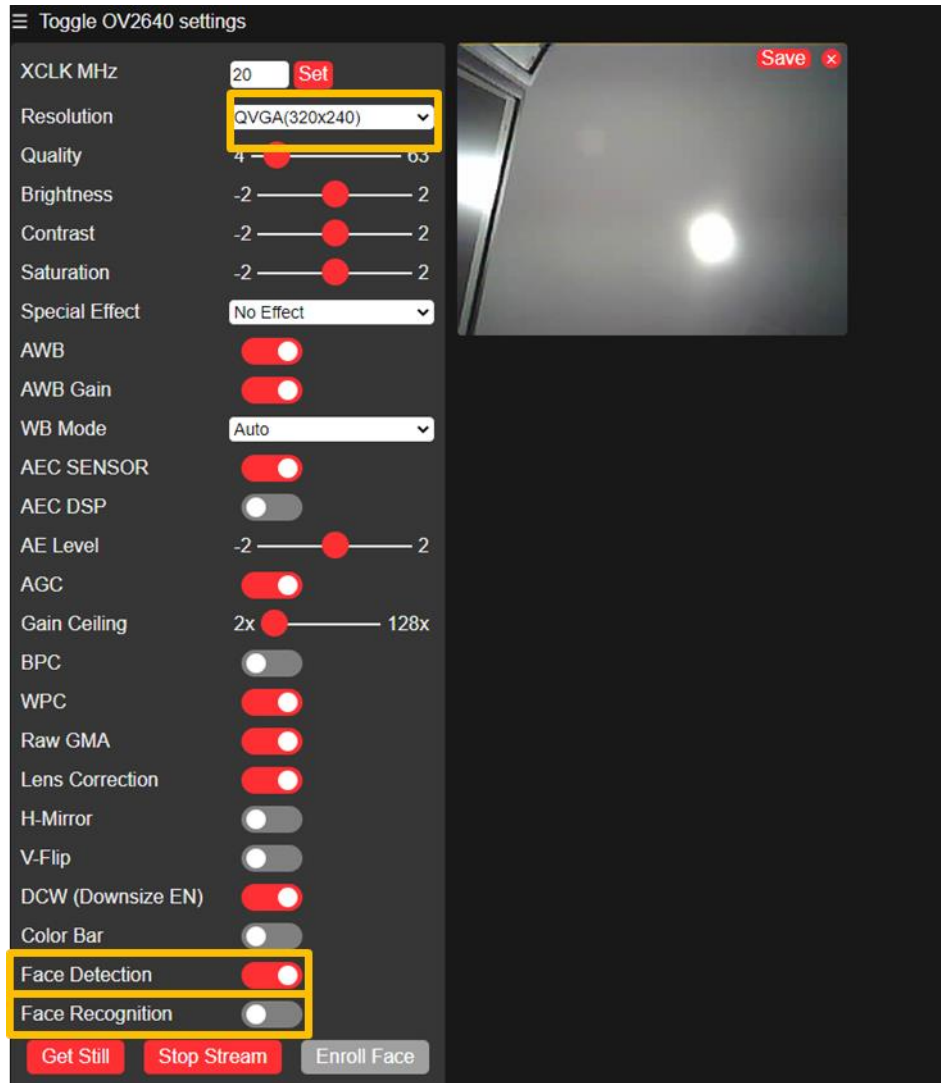


Fig 33. ESP32 보드로 카메라 Web Server 에 연결한 모습

맨 위에 노랑색 box 에서 확인할 수 있듯이 QVGA 이하의 화소로만 Face Detection 과 Face Recognition 을 진행할 수 있다.

밑에 두 box 의 option 을 통해 Face Detection 모델을 실행하거나 Face Recognition 모델을 실행할 수 있다.

맨 아래 빨강색 option 을 통해 'Get Still'을 선택하면 화면을 카메라처럼 캡처할 수 있다. 'Stream' option 을 선택하면 순간 캡처가 아니라 streaming 동영상으로 확인할 수 있다.

<Result – Face Detection>



Fig 34. Face Detection 결과 창

<Analysis>

Face Detection 은 화소 규정(QVGA 이하의 화소)만 충족해 주면 위 결과와 같이 얼굴이 카메라 앞에 확인될 경우, box 를 만들어 준다.

<Result – Face Recognition>


Input		Output	
			
			
			
			

Table 6. Face Recognition 결과 창

<Analysis>

위 모듈로는 한 사람의 다양한 모습을 측정할 수 없다. 한 사람의 한 이미지만 input 으로 넣고 그 이미지만 판별할 수 있다. 동일한 사람이라도 모습이 조금이라도 달라지는 경우 같은 사람이라고 인식을 하지 못한다.

5.2 ESP-Face

ESP-FACE 는 ESP-WHO 에서 사용되는 Face Detection 과 Face Recognition 에 사용되는 함수들과 API 를 모아 놓은 component 이다. 모델에 대한 정보와 모델 구성에 대한 내용이 설명되어 있다. MTMN 과 FRMN 모델 모두 MobileNetV2 에 기반을 두고 있다.

링크는 아래와 같다.

(ESP-FACE Official Github) <https://github.com/espressif/esp-face/tree/master>

(Face Detection 모델-MTMN-에 대한 설명)

https://github.com/espressif/esp-face/tree/master/face_detection

(Face Recognition 모델-FRMN-에 대한 설명)

https://github.com/espressif/esp-face/tree/master/face_detection

(ESP-FACE Tutorial) <https://github.com/espressif/esp-face/tree/master/tutorial>

Tutorial 도 제공하고 있으나, 위 tutorial 은 얼굴 인식을 input 으로 실행하는 모델이 아니라, MNIST 데이터로 만들어진 모델이다. 또한, ESP 사에서 코드를 공개하였지만, ESP 사만의 것으로 custom 되어 있기 때문에 해석하기 어렵다. Weight 와, data 모두 binary 형태로 변환된 상태에서 불러와서 실행한다.

5.3 Issues

<Issue 1> ESP-FACE 를 실제 보드 위에 구동하지 못했다.