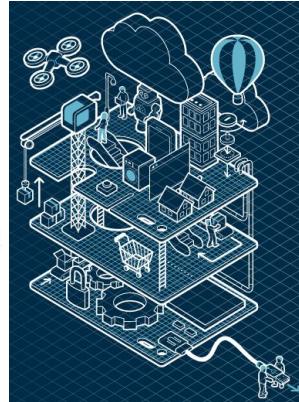




# arm MBED OS



# ARM Mbed OS

12.14.2019 ~

Doc. Revision: 0.4

---

Michael Chunghan Yi(michael@2ipco.com)

2ip, Inc.

Hannam dong, Yongsan-gu

Seoul, South Korea

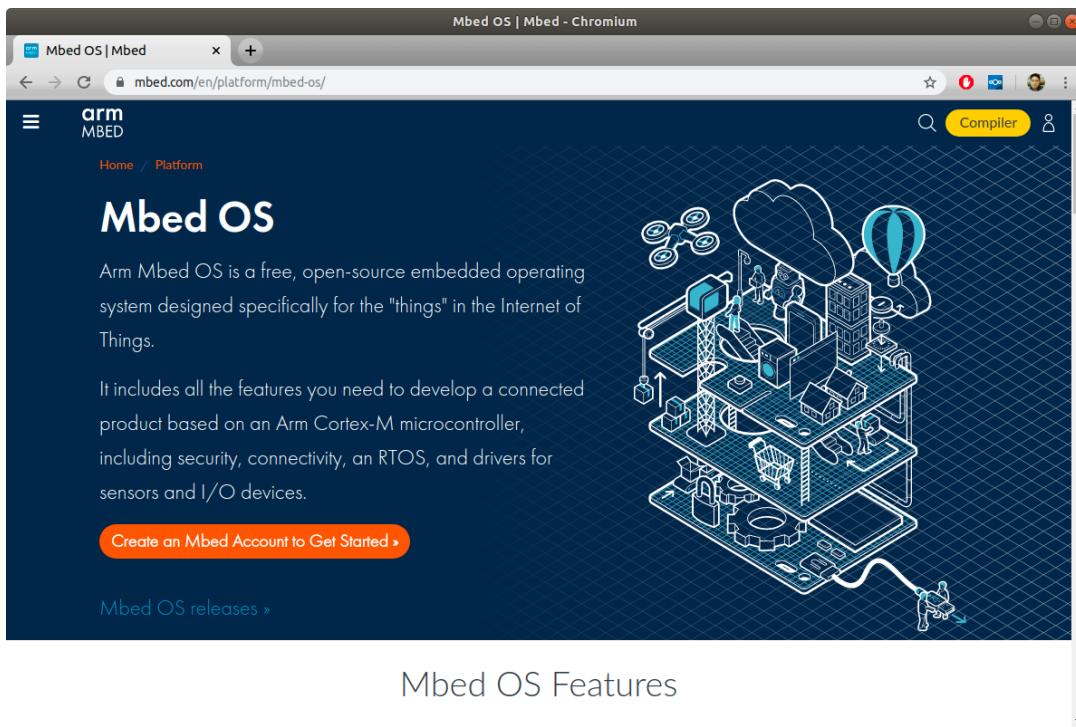
본 문서는 ARM 사에서 개발한 Mbed OS를 이용하여 각종 IoT 장치(Resource constrained device)를 개발하는 과정을 소개하기 위해 작성하였다.

# Contents

1. ARM Mbed OS 소개
2. Mbed CLI w/ STM32 Nucleo F103RB
3. Mbed Studio
4. Mbed Online Compiler
5. Mbed OS 해부
6. ARM Cortex-M Micro-Controller 해부
7. DISCO-L072CZ-LRWAN1 보드

# 1. ARM Mbed OS 소개

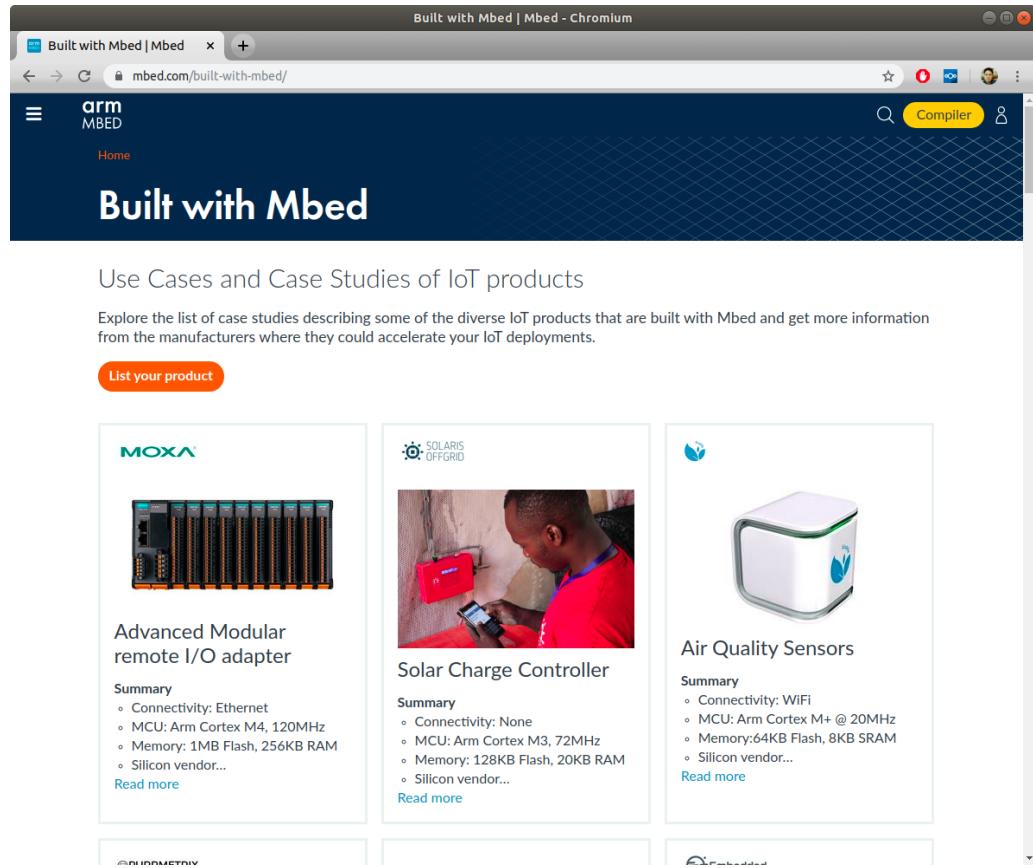
ARM Mbed OS(이하 Mbed OS로 통칭)는 FreeRTOS(<https://www.freertos.org/>), Zephyr(<https://www.zephyrproject.org/>) 등과 견줄만한 open source IoT OS 이다(ARM 사가 밀고 있어 파급력이 매우 큰 느낌이다. 그래서 선택했다^^). Security, Connectivity, RTOS 적인 특징을 갖추고 있으며, 주로 ARM Cortex-M micro controller에 탑재하는 용도로 사용할 수 있다. 따라서 Arduino(AVR 계열) 등에 올리기에는 좀 덩치가 크다고 볼 수 있다~



[그림 1.1] ARM Mbed OS home page

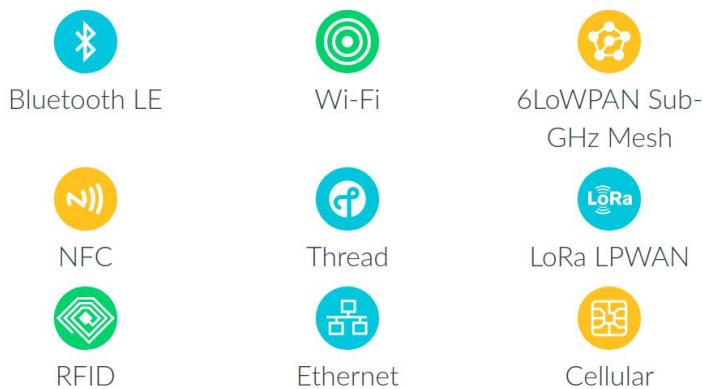
Mbed OS가 활용되는 분야는 매우 다양하다. 이를 확인해 볼 수 있는 site가 아래에 있으니 한번 훑어 보시기 바란다. 재밌는 내용이 아주 많다~ 이 중, 우리에게 도움이 될만한 부분이 없을까?

<https://www.mbed.com/built-with-mbed/>



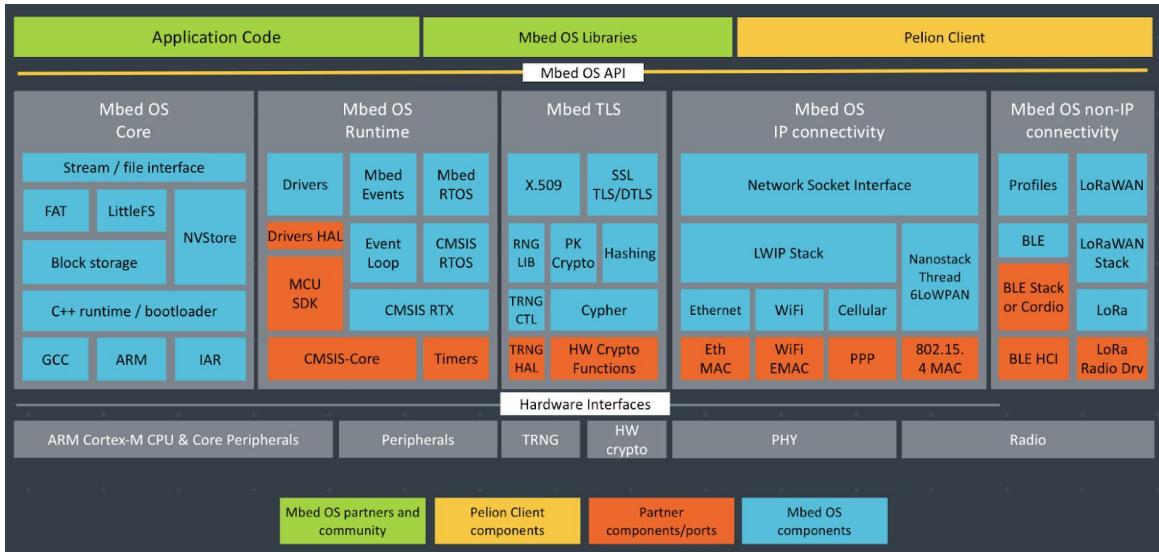
[그림 1.2] ARM Mbed OS가 사용된 다양한 예

**Mbed OS**가 갖고 있는 가장 큰 장점은 뭐니 뭐니 해도 **다양한 connectivity 환경을 지원**한다는 점이 아닐까 싶다. 이 중 우리는 LoRaWAN에 관심이 큰 만큼 앞으로 이 부분을 집중적으로 살펴보아야 할 것이다.



[그림 1.3] ARM Mbed OS Connectivity

Mbed OS의 전체 architecture는 다음과 같다. 이와 관련해서는 추후 하나씩 세세하게 들어보아야 할 것이다.



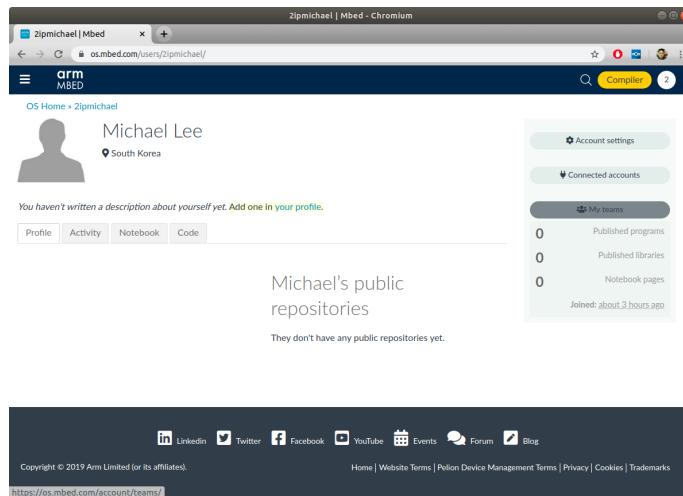
[그림 1.4] ARM Mbed OS Architecture

먼저, **Mbed OS를 사용하기 전에 자신의 계정을 등록**(그림 1.1의 아래 버튼 선택) 하도록 하자. 계정 등록 방법은 아주 간단한 사항이라 여기서는 별도로 정리하지는 않겠다. 각자 시도해 보기 바란다~

Create an Mbed Account to Get Started ➔

[그림 1.5 계정 등록 버튼]

참고: Mbed OS 사용자 계정은 Mbed Studio, Mbed Online Compiler 사용시 반드시 필요하다.



[그림 1.6] Mbed OS 계정 등록 모습

## <Mbed OS 교육 과정>

국내 ARM 교육 기관에서 시행하는 **Mbed OS 교육 과정**을 참고삼아 정리해 보았다. 앞으로 이런 정도의 내용을 파악해야 한다는 야그^^

### 1 일차

#### 1. Open-source Hardware Platform

- Open source hardware 소개
- mbed 소개
- mbed를 이용한 개발 사례 소개

#### 2. ARM Processor

- Arm Processor 소개
- Arm Processor 종류
- Arm Interrupt Controller (GIC, NVIC)
- Arm SysTick Timer

- STM32 Cube & Hal Library

- Discussion

### 3. Mbed

- Mbed ?

- mbed 개발환경 소개

- mbed 프로그래밍 흐름

- mbed 시작하기 : 웹컴파일러를 이용한 예제 빌드 및 실행

- mbed CLI : 환경 설정 및 예제 빌드 및 실행

- mbed Studio 설치 및 기본예제 실행

### 2 일차

### 4. Mbed DataType

- mbed 데이터형

- mbed.h 분석

### 5. Peripheral 이론 및 실습

- GPIO (LED, Button)

- GPIO BusOut (7Segment)

- Analog Input/Output (가변저항, LED)

- Timer & Interrupt

- PWM (LED, Brightness)

- I2C (STM Sensor Hub Control)

- SPI (SD Block Driver)

- QSPI, Little FileSystem

- Watchdog & ResetReason

- MbedCRC

3 일차

## 6. Mbed-OS

- RTOS

- mbed-os

- mbed-os Thread

- mbed-os Mutex

- mbed-os Semaphore

- mbed-os Queue

- mbed-os MemoryPool

- mbed-os Mail

- mbed-os EventFlags

- mbed-os ConditionVariable

- mbed-os RtosTimer

4 일차

- mbed-os EventQueue 인터럽트 처리

- mbed-os EventQueue Shared Event

- mbed-os Event

- mbed-os Callback

## 7. Mbed-OS Configuration System

- configuration parameter 정의

- mbed\_lib.json 분석

- mbed\_app.json 분석

## 5 일차

## 8. Pelion Device Management

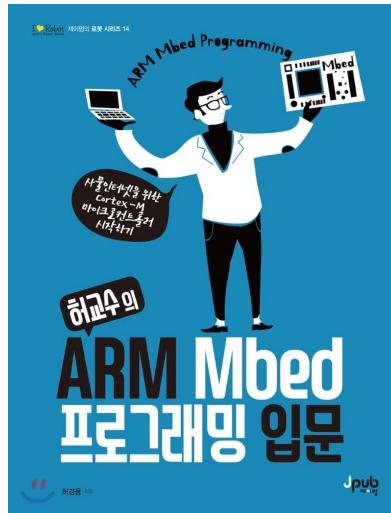
- Pelion Service ?

- LWM2M

- Pelion 계정생성

- DISCO\_L475E\_IOT01A 보드를 사용하여 Pelion 연결하기

교육 기관에 불필요하게 돈을 기부(?)할 필요는 없다. 적당한 개발 보드를 하나 구입한 후, 아래 책 정도를 보면서 study를 진행하는 것으로 충분하다 :) **사실 Mbed OS와 같은 RTOS 그렇게 어렵지 않다?! 그냥 하는 소리가 아니다 :) Linux가 (상대적으로) 훨씬 어렵다...**



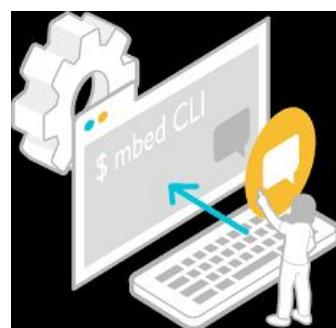
[그림 1.7] 참고 도서

## 2. Mbed CLI w/ STM32 Nucleo F103RB

Mbed OS 환경에서 firmware를 개발하는데 사용되는 도구로는 "Mbed Studio", "Mbed Online Compiler", "Mbed CLI" 세 종류가 있다. 이를 하나씩 하나씩 사용해 보기로 하자.

먼저, 이번 절에서는 (linux 개발자를 위해)Mbed OS CLI를 이용하여 sample code(mbed-os 포함)를 build하는 과정을 소개해 보기로 하겠다. 동작 시험을 위해서는 아래 **STM32 Nucleo-F103RB** 보드를 사용하였다.

"Mbed CLI 만 있어도 개발하는데 아무런 문제가 없다 - [michael@2ipco.com](mailto:michael@2ipco.com)"



[그림 2.1] mbed CLI



[그림 2.2] Nucleo F103RB 보드

<https://os.mbed.com/platforms/ST-Nucleo-F103RB/>

참고: Nucleo F103RB 보드는 가격이 저렴(만원이 조금 넘음)하다.

### a) Ubuntu 18.04에 Mbed CLI 설치하기

지금부터 설명하는 내용은 Ubuntu 18.04 desktop을 기준으로 하며, 아래 site의 내용을 기초로 하였다.

<https://os.mbed.com/docs/mbed-os/v5.14/tools/manual-installation.html>

#### <Install dependencies>

```
$ sudo apt install python2.7 python-pip git mercurial
```

→ 보통 이정도는 이미 설치되어 있다. 혹시 설치 되어 있지 않다면, 각자 설치하기 바란다.

#### <Install Mbed CLI>

```
$ pip install mbed-cli
```

→ Mbed CLI는 python으로 구현되어 있다.

```
$ mbed --help
```

→ mbed가 정상적으로 설치되었는지를 확인하기 위해 위의 명령 실행

### <Install a compiler>

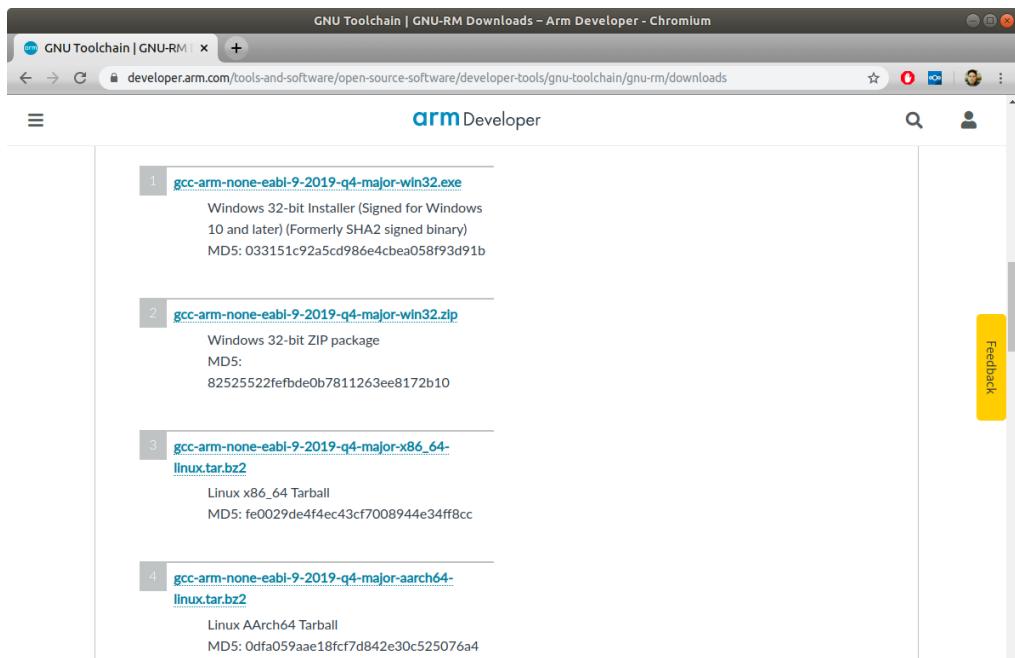
<https://developer.arm.com/open-source/gnu-toolchain/gnu-rm/downloads>

→ 헐, 여기서는 download할 수가 없군...

<https://developer.arm.com/tools-and-software/open-source-software/developer-tools.gnu-toolchain/gnu-rm/downloads>

→ 그럼 여기서 받도록 하자.  
 → gcc-arm-none-eabi-9-2019-q4-major-x86\_64-linux.tar.bz2

참고: ARM Mbed OS를 build하기 위해 사용 가능한 compiler로는 **GCC Arm, Arm Compiler 5, Arm Compiler 6 or IAR** 등을 생각해 볼 수 있다. **우리는 이중에서 open source인 GCC Arm compiler를 사용할 것이다.**



[그림 2.3] GNU-RM Downloads site

```
$ cd ~/workspace/MbedOS
```

```
$ tar xvjf gcc-arm-none-eabi-9-2019-q4-major-x86_64-linux.tar.bz2
```

```
$ sudo mv ./gcc-arm-none-eabi-9-2019-q4-major /opt
```

→ 반드시 /opt 아래로 이동시켜야 하는 것은 아니다.

```
$ vi ~/.bashrc
```

```
...
export MBED_GCC_ARM_PATH=/opt/gcc-arm-none-eabi-9-2019-q4-major/bin
export PATH=/opt/gcc-arm-none-eabi-9-2019-q4-major/bin:$PATH
~
```

```
$ source ~/.bashrc
```

→ 터미널을 다시 실행해도 된다.

참고: 이상의 내용은 (일반적으로) toolchain 설정시 공통적으로 수행하는 부분에 해당한다.

```
chyi@mars:/opt/gcc-arm-none-eabi-9-2019-q4-major/bin$ ./arm-none-eabi-gcc --version
arm-none-eabi-gcc (GNU Tools for Arm Embedded Processors 9-2019-q4-major) 9.2.1 20191025 (release) [ARM/arm-9-branch revision 277599]
Copyright (C) 2019 Free Software Foundation, Inc.
This is free software; see the source for copying conditions. There is NO
warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
```

[그림 2.4] ARM gcc version - gcc 9.2.1 버전

```
$ mbed config -G GCC_ARM_PATH /opt/gcc-arm-none-eabi-9-2019-q4-major/bin
```

→ Mbed CLI가 arm compiler의 위치를 알 수 있도록 해 줌.

### <Bash completion>

→ mbed-cli bash completion(자동 완성 기능)을 설치하기 위해서는 아래 절차를 따른다.

참고: bash completion은 bash 환경에서 명령 전체를 입력할 필요 없이 <TAB> 키를 사용하여 명령을 좀 더 편하게 입력하는 방식을 말한다.

```
$ git clone https://github.com/ARMmbed/mbed-cli
```

→ Mbed-cli를 내려 받는다.

```
'mbed-cli'에 복제합니다...
remote: Enumerating objects: 49, done.
remote: Counting objects: 100% (49/49), done.
remote: Compressing objects: 100% (39/39), done.
remote: Total 3693 (delta 23), reused 26 (delta 9), pack-reused 3644
오브젝트를 받는 중: 100% (3693/3693), 1.31 MiB | 1.40 MiB/s, 완료.
델타를 알아내는 중: 100% (2078/2078), 완료.
```

```
$ cd mbed-cli/tools/bash_completion
```

```
$ mkdir -p ~/.bash_completion.d/
```

```
$ cp mbed ~/.bash_completion.d/
```

→ 이후 터미널을 다시 실행해야 한다.

```
chyi@earth:~/workspace/MbedOS$ mbed config --list
```

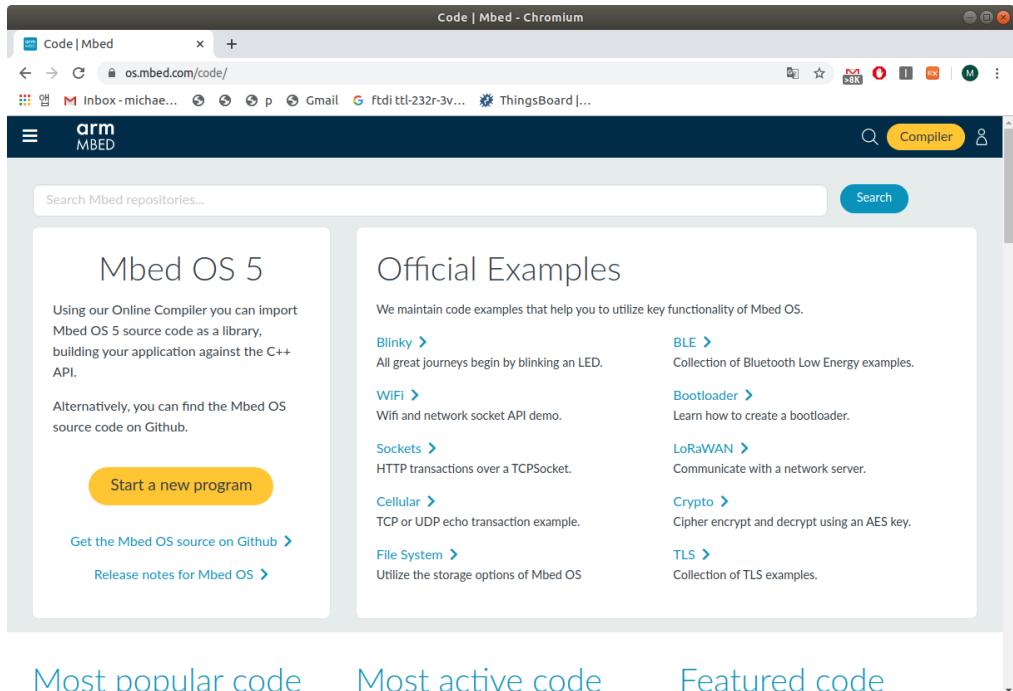
[mbed] Global config:

GCC\_ARM\_PATH=/opt/gcc-arm-none-eabi-9-2019-q4-major/bin

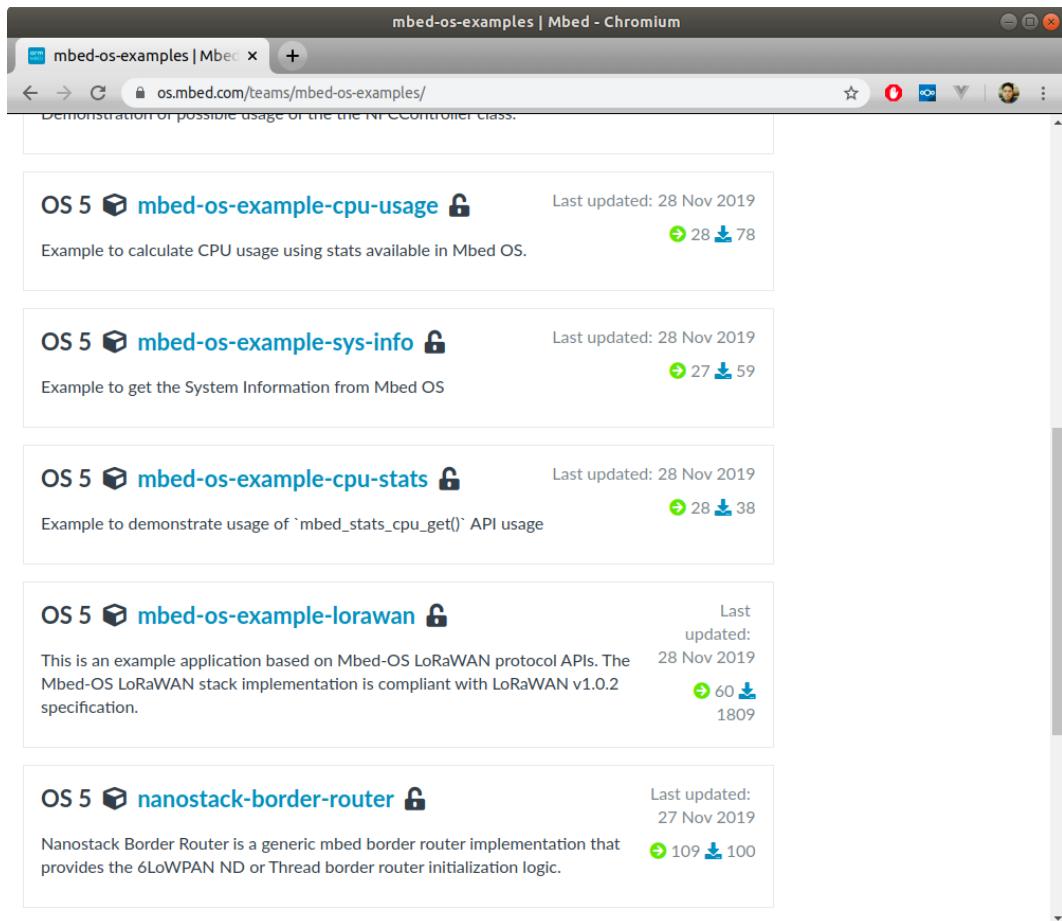
[mbed] Local config (/home/chyi/workspace/MbedOS):

Couldn't find valid mbed program in /home/chyi/workspace/MbedOS

## b) LED Blinky 예제 코드 들려 보기



[그림 2.5] mbed-os 예제 코드(1)



[그림 2.6] mbed-os 예제 코드(2)

```
$ cd ~/workspace/MbedOS
$ mkdir projects; cd projects
$ mbed import
https://github.com/ARMmbed/mbed-os-example-blinky#mbed-os-5.11.0
```

```
[mbed] Working path "/home/chyi/workspace/MbedOS/projects" (directory)
[mbed] Importing program "mbed-os-example-blinky" from
"https://github.com/ARMmbed/mbed-os-example-blinky" at branch/tag "mbed-os-5.11.0"
[mbed] Adding library "mbed-os" from "https://github.com/ARMmbed/mbed-os" at rev #6a0a86538c0b
```

```
$ cd mbed-os-example-blinky
```

```
$ ls -la
```

```
합계 52
drwxr-xr-x 4 chyi chyi 4096 12월 14 17:27 .
```

```
drwxr-xr-x 6 chyi chyi 4096 12월 14 17:27 ..
drwxr-xr-x 8 chyi chyi 4096 12월 14 17:27 .git
-rw-r--r-- 1 chyi chyi 32 12월 14 17:27 .gitignore
-rw-r--r-- 1 chyi chyi 7 12월 14 17:27 .mbed
-rw-r--r-- 1 chyi chyi 5942 12월 14 17:27 README.md
-rw-r--r-- 1 chyi chyi 522 12월 14 17:27 main.cpp
drwxr-xr-x 18 chyi chyi 4096 12월 14 17:27 mbed-os
-rw-r--r-- 1 chyi chyi 77 12월 14 17:27 mbed-os.lib
-rw-r--r-- 1 chyi chyi 354 12월 14 17:27 mbed_app.json
-rw-r--r-- 1 chyi chyi 1339 12월 14 17:27 mbed_settings.py
-rw-r--r-- 1 chyi chyi 3699 12월 14 17:27 stats_report.h
```

```
/* mbed Microcontroller Library
 * Copyright (c) 2018 ARM Limited
 * SPDX-License-Identifier: Apache-2.0
 */

#include "mbed.h"
#include "stats_report.h"

DigitalOut led1(LED1);

// main() runs in its own thread in the OS
int main()
{
    SystemReport sys_state(500 /* Loop delay time in ms */);

    while (true) {
        // Blink LED and wait 0.5 seconds
        led1 = !led1;
        wait(0.5f);

        // Following the main thread wait, report on the current system status
        sys_state.report_state();
    }
}
```

[그림 2.7] LED blinky 예제 코드 - main.cpp

\$ mbed compile -m K64F -t GCC\_ARM

- -m: target board, -t: toolchain
- NXP K64F board(가격: \$35정도 함)는 아래 그림과 같이 생겼는데, 여기서는 이 board에 올릴 수 있는 firmware image를 생성해 보도록 해 보겠다.
- 참고: mbed OS 문서 내용이 이 보드를 기준으로 설명되어 있다.



[그림 2.8] NXP FRDM-K64F 보드

```
[mbed] Working path "/home/chyi/workspace/MbedOS/projects/mbed-os-example-blinky" (program)
Building project mbed-os-example-blinky (K64F, GCC_ARM)
Scan: mbed-os-example-blinky
Compile [ 0.1%]: except.S
Compile [ 0.3%]: mbed_tz_context.c
Compile [ 0.4%]: rf_configuration.c
[Warning] rf_configuration.c@104,25: comparison of integer expressions of different signedness: 'uint32_t'
{aka 'long unsigned int'} and 'int' [-Wsign-compare]
Compile [ 0.5%]: MCR20Drv.c
Compile [ 0.7%]: mbed_fault_handler.c
Compile [ 0.8%]: at24mac.cpp
Compile [ 0.9%]: psa_prot_internal_storage.cpp
Compile [ 1.1%]: fslittle_test.c
Compile [ 1.2%]: fsfat_test.c
Compile [ 1.3%]: pits_impl.cpp
Compile [ 1.5%]: main.cpp
Compile [ 1.6%]: FlashIAPBlockDevice.cpp
Compile [ 1.7%]: AnalogIn.cpp
Compile [ 1.9%]: NanostackRfPhyMcr20a.cpp
Compile [ 2.0%]: SDBlockDevice.cpp
[Warning] SDBlockDevice.cpp@794,24: this statement may fall through [-Wimplicit-fallthrough=]
Compile [ 2.2%]: NanostackRfPhyAtmel.cpp
Compile [ 2.3%]: NanostackRfPhys2Ip.cpp
Compile [ 2.4%]: BusIn.cpp
Compile [ 2.6%]: CAN.cpp
Compile [ 2.7%]: BusInOut.cpp
Compile [ 2.8%]: ESP8266.cpp
[Warning] mbed_error.h@52,55: left shift of negative value [-Wshift-negative-value]
[Warning] mbed_error.h@52,55: left shift of negative value [-Wshift-negative-value]
Compile [ 3.0%]: BusOut.cpp
Compile [ 3.1%]: Ethernet.cpp
...[중간 생략]...
Compile [ 99.6%]: sleep.c
Compile [ 99.7%]: i2c_api.c
Compile [ 99.9%]: fsl_common.c
Compile [100.0%]: rtc_api.c
Link: mbed-os-example-blinky
Elf2Bin: mbed-os-example-blinky
| Module      | .text | .data | .bss |
|-----|-----|-----|-----|
| [fill]       | 160(+160) | 4(+4) | 2122(+2122) |
| [lib]/c.a    | 32042(+32042) | 2472(+2472) | 89(+89) |
```

[lib]/gcc.a	3104(+3104)	0(+0)	0(+0)
[lib]/misc	180(+180)	4(+4)	28(+28)
[lib]/nosys.a	32(+32)	0(+0)	0(+0)
[lib]/stdc++.a	4(+4)	0(+0)	0(+0)
main.o	906(+906)	0(+0)	12(+12)
mbed-os/cmsis	1033(+1033)	0(+0)	0(+0)
mbed-os/components	210(+210)	0(+0)	0(+0)
mbed-os/drivers	633(+633)	0(+0)	0(+0)
mbed-os/features	156(+156)	0(+0)	192(+192)
mbed-os/hal	2311(+2311)	8(+8)	152(+152)
mbed-os/platform	4507(+4507)	260(+260)	226(+226)
mbed-os/rtos	9379(+9379)	168(+168)	6029(+6029)
mbed-os/targets	9235(+9235)	12(+12)	382(+382)
Subtotals	63892(+63892)	2928(+2928)	9232(+9232)
Total Static RAM memory (data + bss): 12160(+12160) bytes			
Total Flash memory (text + data): 66820(+66820) bytes			

Image: ./BUILD/K64F/GCC\_ARM/mbed-os-example-blinky.bin

\$ cd BUILD/

\$ ls -la

```
합계 12
drwxr-xr-x 3 chyi chyi 4096 12월 14 17:28 .
drwxr-xr-x 3 chyi chyi 4096 12월 14 17:28 ..
drwxr-xr-x 3 chyi chyi 4096 12월 14 17:28 GCC_ARM
chiy@earth:~/workspace/MbedOS/projects/mbed-os-example-blinky/BUILD/K64F$ cd GCC_ARM/
chiy@earth:~/workspace/MbedOS/projects/mbed-os-example-blinky/BUILD/K64F/GCC_ARM$ ls -la
합계 5728
drwxr-xr-x 3 chyi chyi 4096 12월 14 17:28 .
drwxr-xr-x 3 chyi chyi 4096 12월 14 17:28 ..
-rw-r--r-- 1 chyi chyi 13879 12월 14 17:28 .includes_90b1235f63435861e0c680c5463f9e2f.txt
-rw-r--r-- 1 chyi chyi 0 12월 14 17:28 .includes_d41d8cd98f00b204e9800998ecf8427e.txt
-rw-r--r-- 1 chyi chyi 62218 12월 14 17:28 .link_options.txt
-rw-r--r-- 1 chyi chyi 4526 12월 14 17:28 .link_script.ld
-rw-r--r-- 1 chyi chyi 356 12월 14 17:28 .profile-asm
-rw-r--r-- 1 chyi chyi 2687 12월 14 17:28 .profile-c
-rw-r--r-- 1 chyi chyi 2729 12월 14 17:28 .profile-cxx
-rw-r--r-- 1 chyi chyi 2634 12월 14 17:28 .profile-ld
-rw-r--r-- 1 chyi chyi 10111 12월 14 17:28 main.d
-rw-r--r-- 1 chyi chyi 62912 12월 14 17:28 main.o
drwxr-xr-x 11 chyi chyi 4096 12월 14 17:28 mbed-os
-rwxr-xr-x 1 chyi chyi 67552 12월 14 17:28 mbed-os-example-blinky.bin ← 애가 실제 board에 탑재될 firmware 이미지 파일임.
-rwxr-xr-x 1 chyi chyi 1548092 12월 14 17:28 mbed-os-example-blinky.elf
-rw-r--r-- 1 chyi chyi 3946951 12월 14 17:28 mbed-os-example-blinky.map
-rw-r--r-- 1 chyi chyi 2227 12월 14 17:28 mbed-os-example-blinky_map.csv
-rw-r--r-- 1 chyi chyi 44670 12월 14 17:28 mbed-os-example-blinky_map.html
-rw-r--r-- 1 chyi chyi 3935 12월 14 17:28 mbed-os-example-blinky_map.json
-rw-r--r-- 1 chyi chyi 46172 12월 14 17:28 mbed_config.h
```

### c) Nucleo F103RB 보드에 mbed-os(firmware) 올리기

(개인적으로) mbed-os를 지원하는 STM32 **Nucleo F103RB** 보드가 하나 있어, 여기에 mbed-os를 올려 보도록 하겠다.

```
$ cd ~/workspace/new_boards/MbedOS/mbed-os-example-blinky
```

```
$ mbed target NUCLEO_F103RB
```

→ Target을 지정한다.

or

```
$ mbed target detect
```

→ 특정 target 명을 알 수 없을 경우, 이렇게 할 수도 있다.

[mbed] Working path

"/home/chyi/workspace/new\_boards/MbedOS/mbed-os-example-blinky" (program)

[mbed] detect now set as default target in program "mbed-os-example-blinky"

```
$ mbed toolchain GCC_ARM
```

→ Toolchain을 지정한다. 이렇게 지정하는 이유는 compile 시 일일히 지정하지 않기 위함이다.

```
$ mbed compile -c
```

→ clean build를 한다.

```
$ mbed compile -f
```

→ flash writing을 한다.

```
[mbed] Working path "/home/chyi/workspace/new_boards/MbedOS/mbed-os-example-blinky" (program)
```

```
[mbed] Detected "NUCLEO_F103RB" connected to "/media/chyi/NODE_F103RB" and using com port  
"/dev/ttyACM0"
```

```
[Warning] @,: Compiler version mismatch: Have 9.2.1; expected version >= 6.0.0 and < 7.0.0
```

```
Building project mbed-os-example-blinky (NUCLEO_F103RB, GCC_ARM)
```

```
Scan: mbed-os-example-blinky
```

```
Link: mbed-os-example-blinky
```

```
Elf2Bin: mbed-os-example-blinky
```

```
| Module | .text | .data | .bss |
```

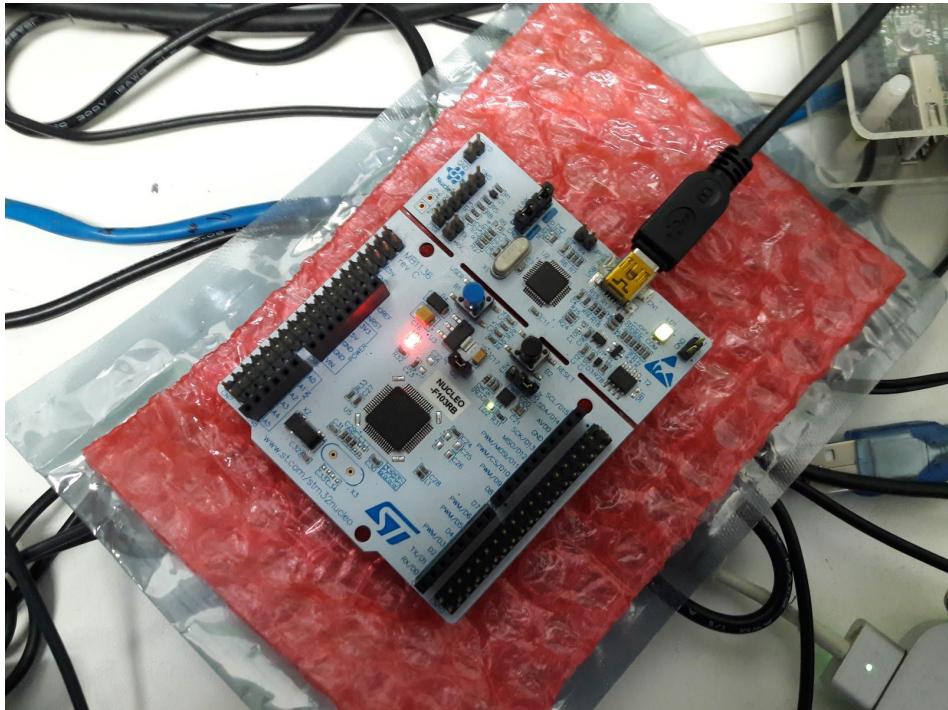
[fill]	92(+0)	12(+0)	32(+0)	
[lib]/c.a	27650(+0)	2472(+0)	89(+0)	
[lib]/gcc.a	4152(+0)	0(+0)	0(+0)	
[lib]/misc	208(+0)	12(+0)	28(+0)	
main.o	918(+0)	4(+0)	36(+0)	
mbed-os/cmsis	1033(+0)	0(+0)	84(+0)	
mbed-os/components	70(+0)	0(+0)	0(+0)	
mbed-os/drivers	855(+0)	4(+0)	100(+0)	
mbed-os/features	172(+0)	4(+0)	192(+0)	
mbed-os/hal	2067(+0)	4(+0)	68(+0)	
mbed-os/platform	4027(+0)	260(+0)	210(+0)	
mbed-os/rtos	8100(+0)	168(+0)	5969(+0)	
mbed-os/targets	6774(+0)	4(+0)	368(+0)	
Subtotals	56118(+0)	2944(+0)	7176(+0)	

Total Static RAM memory (data + bss): 10120(+0) bytes

Total Flash memory (text + data): 59062(+0) bytes

Image: ./BUILD/NUCLEO\_F103RB/GCC\_ARM/mbed-os-example-blinky.bin

자, flash writing에 성공하였다. F103RB 보드에 LED가 1초 간격으로 깜빡거린다.



[그림 2.9] Nucleo F103RB 보드에 firmware(mbed-os)를 올린 모습 - 가운데 녹색 LED blinking 중

**\$ mbed compile -f --sterm**

→ Minicom 등과 같은 serial console emulator를 통해 board 상태를 확인하기 위해 설정해 준다.

```
[mbed] Working path "/home/chyi/workspace/new_boards/MbedOS/mbed-os-example-blinky" (program)
[mbed] Detected "NUCLEO_F103RB" connected to "/media/chyi/NODE_F103RB" and using com port
"/dev/ttyACM0"
[Warning] @,: Compiler version mismatch: Have 9.2.1; expected version >= 6.0.0 and < 7.0.0
Building project mbed-os-example-blinky (NUCLEO_F103RB, GCC_ARM)
Scan: mbed-os-example-blinky
Link: mbed-os-example-blinky
Elf2Bin: mbed-os-example-blinky
| Module      | .text | .data | .bss |
|-----|-----|-----|-----|
| [fill]      | 92(+0) | 12(+0) | 32(+0) |
| [lib]/c.a   | 27650(+0) | 2472(+0) | 89(+0) |
| [lib]/gcc.a | 4152(+0) | 0(+0) | 0(+0) |
```

```
| [lib]/misc      | 208(+0) | 12(+0) | 28(+0) |
| main.o         | 918(+0) | 4(+0)  | 36(+0) |
| mbed-os/cmsis | 1033(+0) | 0(+0)  | 84(+0) |
| mbed-os/components | 70(+0) | 0(+0) | 0(+0) |
| mbed-os/drivers | 855(+0) | 4(+0)  | 100(+0) |
| mbed-os/features | 172(+0) | 4(+0)  | 192(+0) |
| mbed-os/hal    | 2067(+0) | 4(+0)  | 68(+0) |
| mbed-os/platform | 4027(+0) | 260(+0) | 210(+0) |
| mbed-os/rtos   | 8100(+0) | 168(+0) | 5969(+0) |
| mbed-os/targets | 6774(+0) | 4(+0)  | 368(+0) |
| Subtotals      | 56118(+0) | 2944(+0) | 7176(+0) |
Total Static RAM memory (data + bss): 10120(+0) bytes
Total Flash memory (text + data): 59062(+0) bytes
Image: ./BUILD/NUCLEO_F103RB/GCC_ARM/mbed-os-example-blinky.bin
```

\$ **sudo minicom -s**

→ /dev/ttyACM0, 115200, 8N1

```

chyi@mars:~$ sudo minicom -s
[sudo] chyi의 암호:

Welcome to minicom 2.7.1

OPTIONS: I18n
Compiled on Aug 13 2017, 15:25:34.
Port /dev/ttyACM0, 11:09:05

Press CTRL-A Z for help on special keys

ello World from arm!
threadA: Hello World from arm!
threadB: Hello World from arm!
threadA: Hello World from arm!
threadB: Hello World from arm!
threadA: Hello World from arm!
threadB: Hello World from arm!
threadA: Hello World from arm!
threadB: Hello World from arm!
threadA: Hello World from arm!
threadB: Hello World from arm!
threadA: Hello World from arm!
t=====
Idle: 0% Usage: 100%
===== HEAP STATS =====
Current heap: 1216
Max heap size: 1248
===== THREAD STATS =====
ID: 0x200016b4

```

[그림 2.10] Nucleo F103RB 보드 - minicom 설정

어떤가 ? 간단하지 않은가 ?

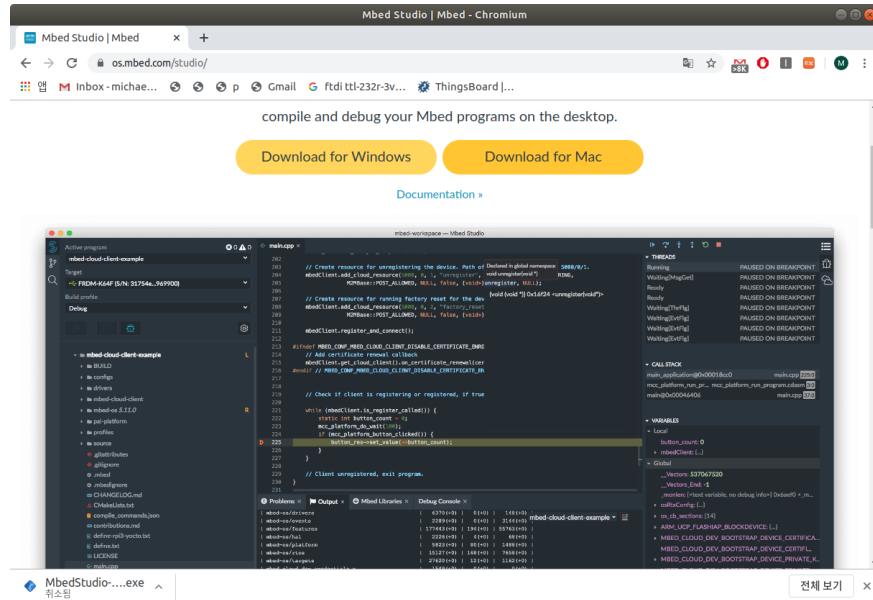
<TBD>

F103RB 보드에 주변 장치(예: LED)를 불인 상태에서 이를 제어하는 코드를 작성하여 돌려 보자.

### 3. Mbed Studio

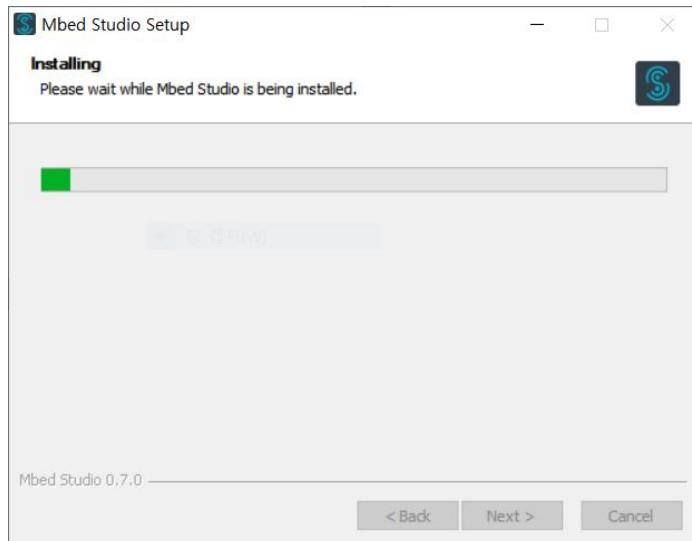
Mbed Studio는 MS Visual Studio 처럼 IDE 환경에서 개발하고자 할 때 설치하여 사용한다. 현재는 Windows & MacOS 만을 지원한다. Linux 개발자라면 Mbed CLI를 이용하면 된다. 따라서 본 문서에서는 Mbed Studio 설치 과정만 간단히 정리해 보았다. 나머지 build 이후의 작업은 직접 해 보시길~

<https://os.mbed.com/studio/>

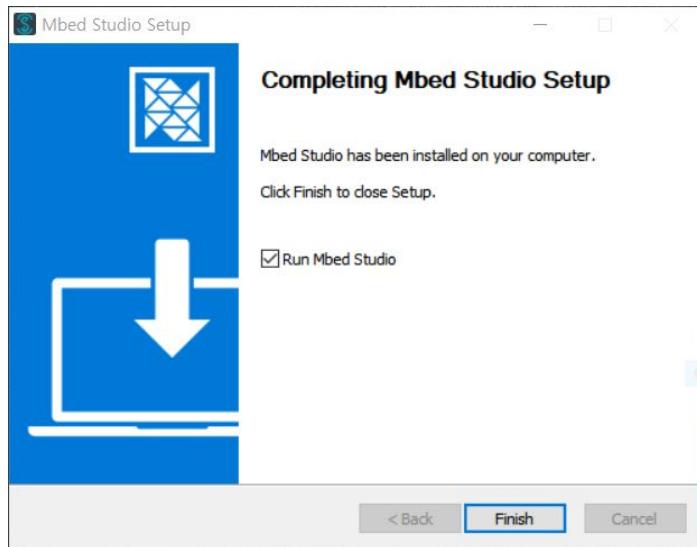


[그림 3.1] Mbed Studio 설치 Page

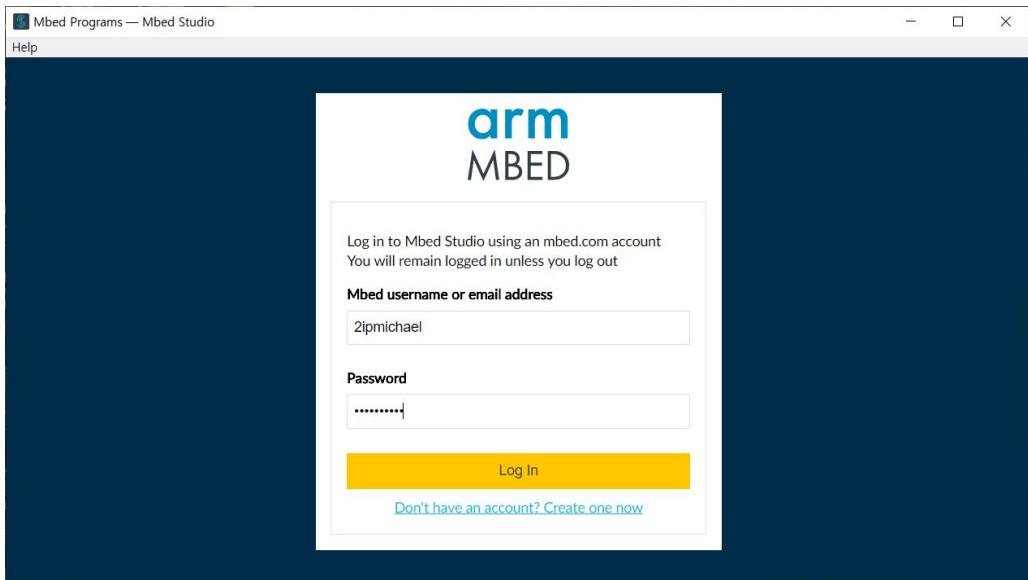
그럼, 이제부터 Windows 10에 설치해 보도록 하겠다.



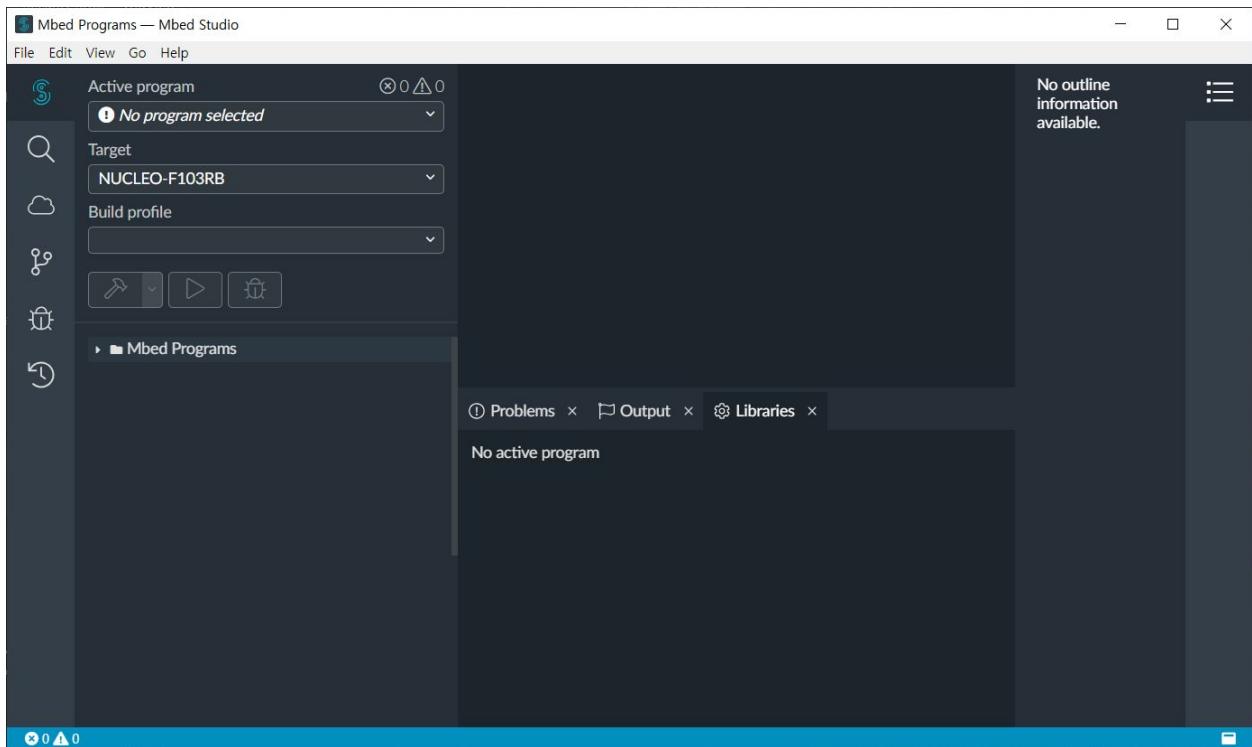
[그림 3.2] Mbed Studio 설치 과정(1)



[그림 3.3] Mbed Studio 설치 과정(2)



[그림 3.4] Mbed Studio 설치 과정(3) - 1장에서 등록한 사용자 계정 사용



[그림 3.5] Mbed Studio 실행 모습

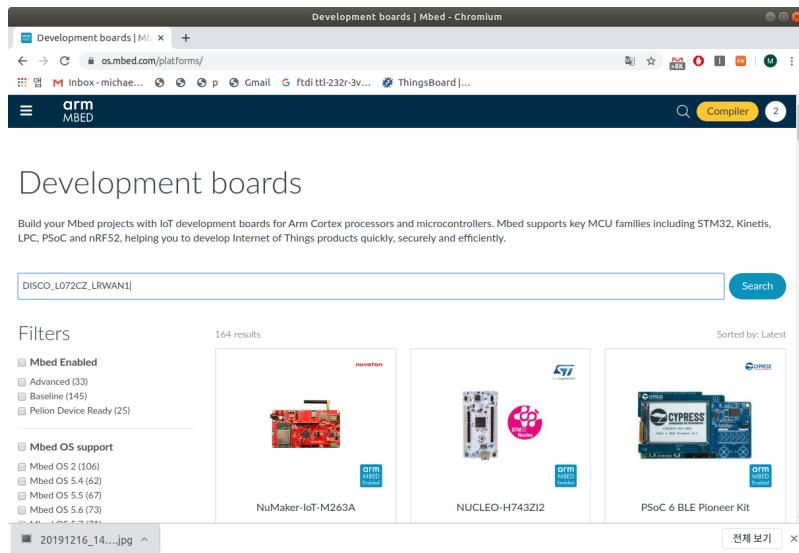
<TBD>

Example code를 내려 받아 build해 보자.

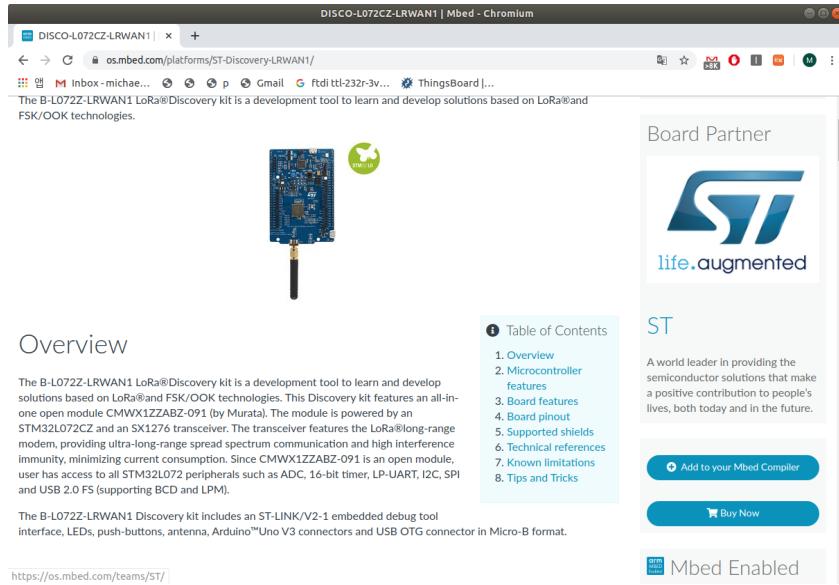
## 4. Mbed Online Compiler

지금 부터는 Mbed Online Compiler를 사용하여 firmware를 개발하는 방법을 소개해 보도록 하겠다. **Web을 통해 개발을 할 수 있는 아주 편리한 방법이다 :)**

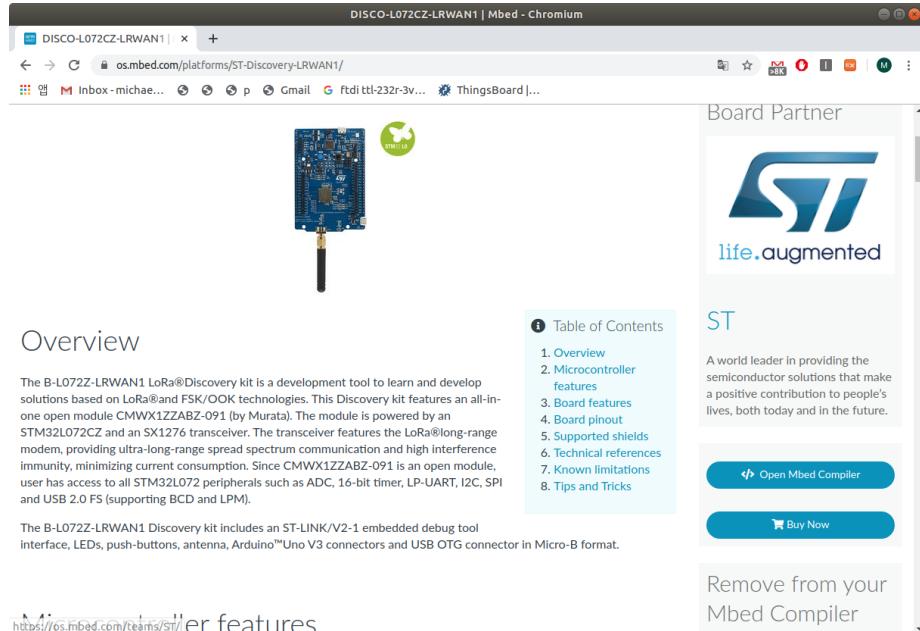
<https://os.mbed.com/docs/mbed-os/v5.14/quick-start/online-with-the-online-compiler.html>



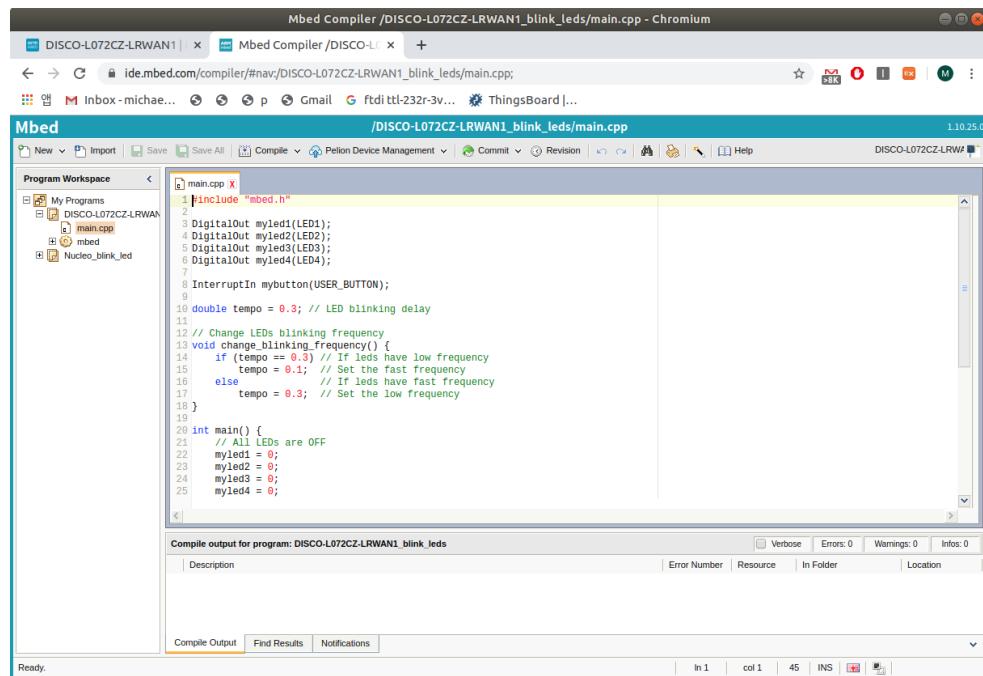
[그림 4.1] Mbed OS Online Compiler - 사용하려는 보드 선택(검색)(1)



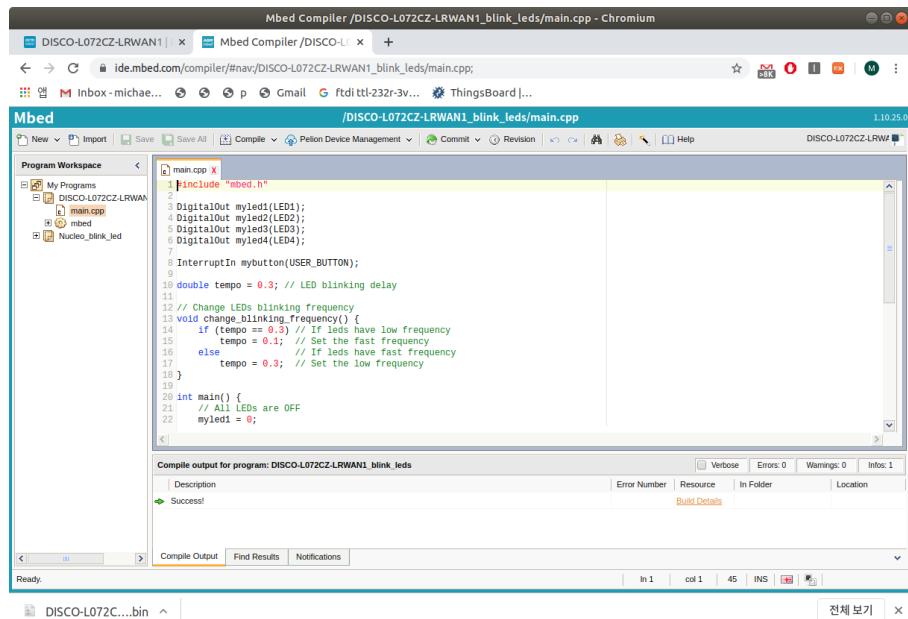
[그림 4.2] Mbed OS Online Compiler - 우측 "Add to your Mbed Compiler" 버튼 선택



[그림 4.3] Mbed OS Online Compiler - 우측 "Open Mbed Compiler" 버튼 선택



[그림 4.4] Mbed OS Online Compiler 화면



[그림 4.5] Mbed OS Online Compiler - Compile 모습

참고: Compile이 정상적으로 완료되면 bin 파일이 자동으로 download 된다.

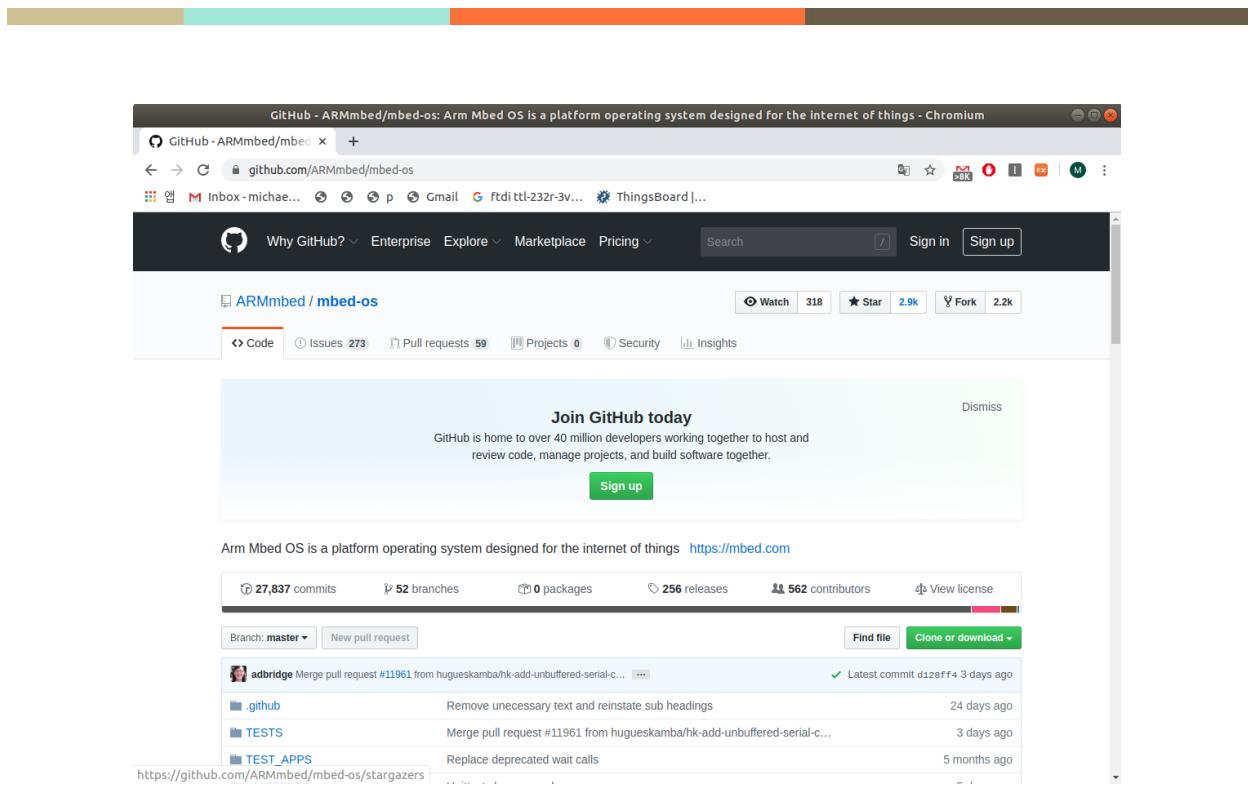
### <firmware writing하기>

Online compiler에서 compile하여 생성한 bin 파일을 Mbed 보드의 USB device folder로 복사한 후, reset 해 주면 된다.

- F103RB 상에서 이렇게 했는데, 잘 안되는 것 같다(물론 LED Blinky code를 테스트해 봄)... 나중에 다시 시도해 보기로 하자.

## 5. Mbed OS 해부

<TBD>



[그림 5.1] Mbed OS github

## 6. ARM Cortex-M Micro-Controller 해부

아래 서적을 읽어 보자.

[1] The Definitive Guide to the ARM Cortex-M3 2nd edition

<https://www.eecs.umich.edu/courses/eecs373/labs/refs/M3%20Guide.pdf>

[2] The Definitive Guide to ARM® Cortex®-M3 and Cortex®-M4 Processors  
3rd Edition

## 7. DISCO-L072CZ-LRWAN1 보드

이번 장에서는 DISCO\_L072CZ-LRWAN1 보드에 lorawan firmware를 올려 보도록 하겠다.

## a) STM32 LoRa and Sigfox Discovery Kit

<https://os.mbed.com/platforms/ST-Discovery-LRWAN1/>

The screenshot shows a web browser window titled "DISCO-L072CZ-LRWAN1 | Mbed - Chromium". The address bar shows the URL "os.mbed.com/platforms/ST-Discovery-LRWAN1/". The main content area displays the "DISCO-L072CZ-LRWAN1" board. A photograph of the blue printed circuit board (PCB) is shown, featuring an STM32L072CZ microcontroller and a LoRa module. To the right of the board, there is a sidebar with the following information:

- Compiler:** A button labeled "Compiler" with a count of "2".
- Board Partner:** The ST logo with the tagline "life.augmented".
- ST:** A brief description: "A world leader in providing the semiconductor solutions that make a positive contribution to people's lives, both today and in the future."
- Tip:** A note: "To compile a program for this board using Mbed CLI, use disco\_l072cz\_lrwan1 as the target name."

The left sidebar includes links for "Boards" and "DISCO-L072CZ-LRWAN1". The main content area also contains a "Table of Contents" section with links to various board features.

[그림 7.1] DISCO\_L072CZ-LRWAN1 보드

[https://kr.mouser.com/ProductDetail/STMicroelectronics/B-L072Z-LRWAN1?qs=PzRbFReKxL2UVPVODSIYig%3D%3D&gclid=EA1alQobChMI9bXag6m55glVy2kqCh0uSAiwEAQYASABEgL7BPD\\_BwE](https://kr.mouser.com/ProductDetail/STMicroelectronics/B-L072Z-LRWAN1?qs=PzRbFReKxL2UVPVODSIYig%3D%3D&gclid=EA1alQobChMI9bXag6m55glVy2kqCh0uSAiwEAQYASABEgL7BPD_BwE)

참고: ₩60,915 (좀 비싸다)

아래에 Microcontroller(줄여서 micom)와 보드의 주요 특징을 열거해 보았다.

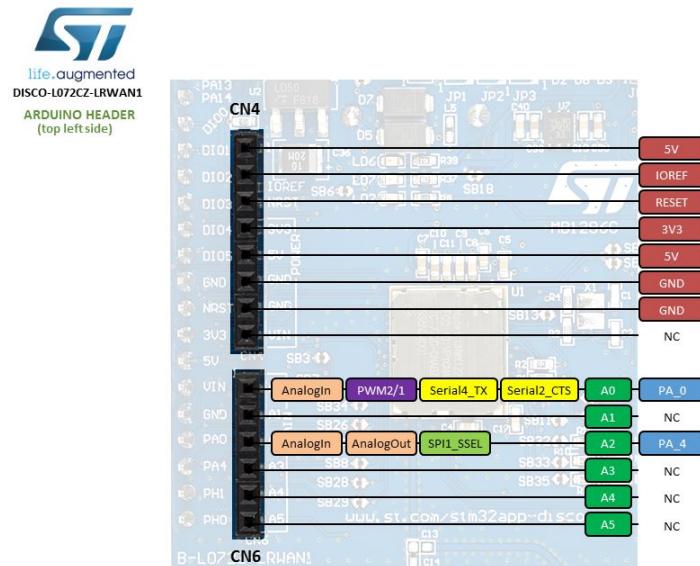
## Microcontroller features

- CMWX1ZZABZ-091 LoRa® module with STM32L072CZ
- ARM® 32-bit Cortex®-M0+ CPU
- 32 MHz max CPU frequency
- VDD from 1.65 V to 3.6 V
- 192 KB Flash
- 20 KB SRAM
- GPIO (40) with external interrupt capability
- General-purpose Timer (4)
- Basic Timer (2)
- Low Power Timer
- SPI (6)
- I2S
- I2C (3)
- USART (4)
- Low-power UART
- USB 2.0 full-speed
- 12-bit ADC with 13 channels
- 12-bit DAC (2) with 1 channel each
- Comparators (2)
- RTC
- Capacitive sensing channels (19)
- Random Generator (TRNG for HW entropy)

## Board features

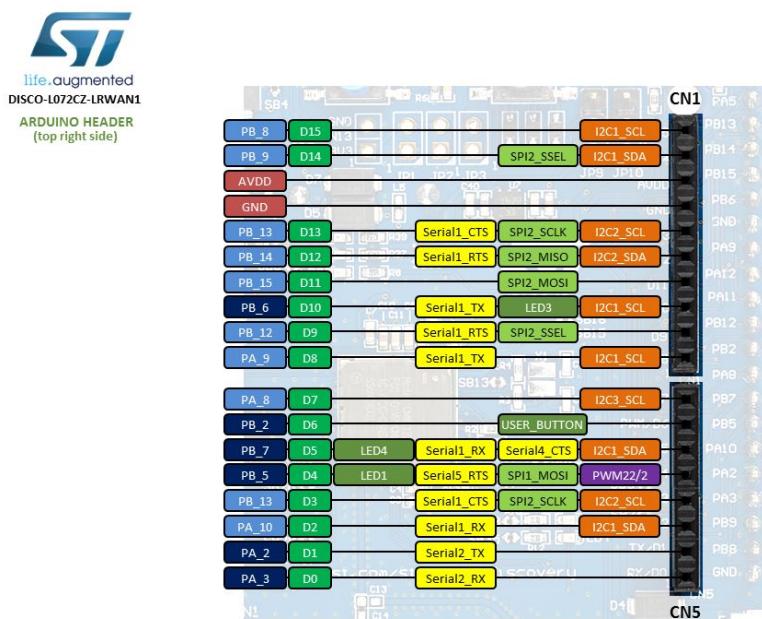
- Two types of extension resources
  - Arduino Uno Revision 3 connectivity
  - STMicroelectronics extension pin headers for full access to all STM32 I/Os
- On-board ST-LINK/V2-1 debugger/programmer with SWD connector
  - Selection-mode switch to use the kit as a standalone ST-LINK/V2-1
- Flexible board power supply
  - USB VBUS or external source (3.3 V, 5 V, 7 - 12 V)
  - Power management access point
- User LED (LD1, LD2, LD3, LD4)
- Two push buttons: USER and RESET
- USB re-enumeration capability: three different interfaces supported on USB
  - Virtual Com port
  - Mass storage (USB Disk drive) for drag'n'drop programming
  - Debug port

다음 4개의 그림은 실제 programming 시 중요한 부분(센서 연결시 필요)이라고 할 수 있는 확장 pin header에 대한 pin map 정보를 표현한 것이다.

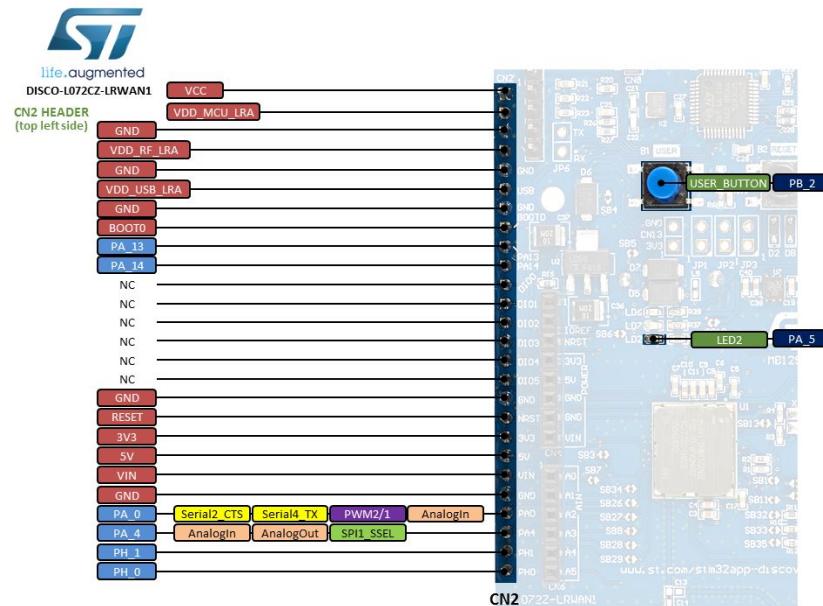


[그림 7.2] Arduino-compatible pin header(보드 top 왼쪽 pin - CN4, CN6)

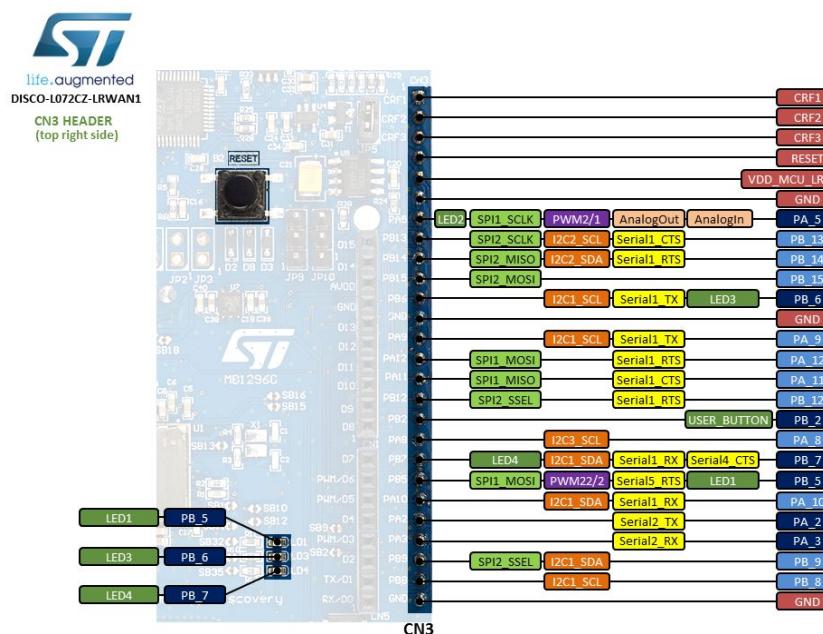
참고: 위의 그림은 Antenna가 아래 방향에 있는 그림이다.



[그림 7.3] Arduino-compatible pin header(보드 top 오른쪽 pin - CN1)



[그림 7.4] Morpho pin header(보드 top 왼쪽 pin - CN2)



[그림 7.5] Morpho pin header(보드 top 왼쪽 pin - CN3)

## b) lorawan example build하기

지금 부터 설명하는 내용은 아래 site를 기초로 한다.

<https://os.mbed.com/teams/mbed-os-examples/code/mbed-os-example-lorawan/>

```
$ cd ~/workspace/new_boards/MbedOS
```

```
$ mbed import mbed-os-example-lorawan
```

```
[mbed] Working path "/home/chyi/workspace/new_boards/MbedOS" (directory)
[mbed] Importing program "mbed-os-example-lorawan" from
"https://github.com/ARMmbed/mbed-os-example-lorawan" at latest revision in the current branch
[mbed] Adding library "mbed-lora-radio-drv" from
"https://github.com/ARMmbed/mbed-semtech-lora-rf-drivers" at rev #6012fa43cf9f
[mbed] Adding library "mbed-os" from "https://github.com/ARMmbed/mbed-os" at rev #b6e5a0a8afa3
[mbed] Auto-installing missing Python modules (pycryptodome, psutil, click)...
[mbed] ERROR: Missing Python modules were not auto-installed.

The Mbed OS tools in this program require the following Python modules: pycryptodome, psutil, click
You can install all missing modules by running "pip install -r requirements.txt" in
"/home/chyi/workspace/new_boards/MbedOS/mbed-os-example-lorawan/mbed-os"
On Posix systems (Linux, etc) you might have to switch to superuser account or use "sudo"
--
```

```
$ cd mbed-os-example-lorawan/mbed-os
```

```
$ sudo pip install -r requirements.txt
```

→ pycryptodome, psutil, click 등 필요한 python package를 자동으로 설치해 준다.

### <Supported MCUs>

```
ARCH_BLE,          ARCH_BLE_BOOT,      ARCH_BLE_OTA,
ARCH_GPRS,         ARCH_LINK,        ARCH_LINK_BOOT,
ARCH_LINK_OTA,     ARCH_MAX,        ARCH_PRO,
ARDUINO_NANO33BLE, ARM_CM3DS_MPS2,   ARM_IOTSS_BEID,
ARM_MPS2_M0,       ARM_MPS2_M0P,     ARM_MPS2_M3,
ARM_MPS2_M4,       ARM_MPS2_M7,     ARM_MUSCA_A1_NS,
ARM_MUSCA_A1_S,   B96B_F446VE,    BLUEPILL_F103C8,
CC3220SF_LAUNCHXL, CY8CKIT_062S2_43012, CY8CKIT_062_BLE,
CY8CKIT_062_WIFI_BT, CY8CPROTO_062_4343W, CY8CPROTO_064_SB,
CYW943012P6EVB_01, DELTA_DFBM_NQ620,  DELTA_DFCM_NNN40,
DELTA_DFCM_NNN40_BOOT, DELTA_DFCM_NNN40_OTA, DELTA_DFCM_NNN50,
DELTA_DFCM_NNN50_BOOT, DELTA_DFCM_NNN50_OTA, DISCO_F051R8,
DISCO_F100RB,      DISCO_F303VC,     DISCO_F334C8,
DISCO_F401VC,      DISCO_F407VG,     DISCO_F413ZH,
DISCO_F429ZI,      DISCO_F469NI,     DISCO_F746NG,
DISCO_F769NI,      DISCO_L053C8,    DISCO_L072CZ_LRUMAN1, ↵ 이거네 ...
DISCO_L475VG_IOT01A, DISCO_L476VG,   DISCO_L496AG,
DISCO_L4R9I,        EFM32GG11_STK3701, EFM32GG_STK3700,
EFM32HG_STK3400,   EFM32LG_STK3600,  EFM32PG12_STK3402,
EFM32PG_STK3401,   EFM32WG_STK3800,  EFM32ZG_STK3200,
ELEKTOR_COCORICO,  ELMO_F411RE,    EP_AGORA,
EV_COG_AD3029LZ,   EV_COG_AD4050LZ,  FF1705_L151CC,
```

FF_LPC546XX,	FUTURE_SEQUANA,	FUTURE_SEQUANA_M0,
FVP MPS2_M0,	FVP MPS2_M0P,	FVP MPS2_M3,
FVP MPS2_M4,	FVP MPS2_M7,	GD32_E103VB,
GD32_F307VG,	GD32_F450ZI,	GR_LYCHEE,
HEXIWEAR,	HRM1017,	HRM1017_BOOT,
HRM1017_OTA,	IM880B,	K20D50M,
K22F,	K64F,	K66F,
K82F,	KL05Z,	KL25Z,
KL26Z,	KL27Z,	KL43Z,
KL46Z,	KL82Z,	KW24D,
KW41Z,	LPC1114,	LPC11C24,
LPC11U24,	LPC11U24_301,	LPC11U34_421,
LPC11U35_401,	LPC11U35_501,	LPC11U35_501_IBDAP,
LPC11U35_Y5_MBUG,	LPC11U37H_401,	LPC11U37_501,
LPC11U68,	LPC1347,	LPC1549,
LPC1768,	LPC1769,	LPC4088,
LPC4088_DM,	LPC4330_M0,	LPC4330_M4,
LPC4337,	LPC54114,	LPC546XX,
LPC55S69_NS,	LPC55S69_S,	LPC810,
LPC812,	LPC824,	LPCCAPPUCCINO,
MAX32600MBED,	MAX32620FTHR,	MAX32620HSP,
MAX32625MBED,	MAX32625NEXPAQ,	MAX32625PICO,
MAX32630FTHR,	MAXWSNENV,	MBED_CONNECT_ODIN,
MCU_LPC4088,	MICRONFCBOARD,	MIMXRT1050_EVK,
MOTE_L152RC,	MTB_ACONNO_ACN52832,	MTB_ADV_WISE_1510,
MTB_ADV_WISE_1530,	MTB_ADV_WISE_1570,	MTB_LAIRD_BL600,
MTB_LAIRD_BL652,	MTB_LAIRD_BL654,	MTB_MTS_DRAGONFLY,
MTB_MTS_XDOT,	MTB_MURATA_ABZ,	MTB_MURATA_WSM_BL241,
MTB_MXCHIP_EMW3166,	MTB_RAK811,	MTB_STM32_F439,
MTB_STM_L475,	MTB_STM_S2LP,	MTB_UBLOX_NINA_B1,
MTB_UBLOX_ODIN_W2,	MTB_USI_WM_BN_BM_22,	MTM_MTCONNECT04S,
MTM_MTCONNECT04S_BOOT,	MTM_MTCONNECT04S_OTA,	MTS_DRAGONFLY_F411RE,
MTS_DRAGONFLY_L471QG,	MTS_GAMBIT,	MTS_MDOT_F405RG,
MTS_MDOT_F411RE,	NCS36510,	NRF51822,
NRF51822_BOOT,	NRF51822_OTA,	NRF51822_Y5_MBUG,
NRF51_DK,	NRF51_DK_BOOT,	NRF51_DK_LEGACY,
NRF51_DK_OTA,	NRF51_DONGLE,	NRF51_DONGLE_BOOT,
NRF51_DONGLE_LEGACY,	NRF51_DONGLE_OTA,	NRF51_MICROBIT,
NRF51_MICROBIT_B,	NRF51_MICROBIT_BOOT,	NRF51_MICROBIT_B_BOOT,
NRF51_MICROBIT_B_OTA,	NRF51_MICROBIT_OTA,	NRF52840_DK,
NRF52_DK,	NUCLEO_F030R8,	NUCLEO_F031K6,
NUCLEO_F042K6,	NUCLEO_F070RB,	NUCLEO_F072RB,
NUCLEO_F091RC,	NUCLEO_F103RB,	NUCLEO_F207ZG,
NUCLEO_F302R8,	NUCLEO_F303K8,	NUCLEO_F303RE,
NUCLEO_F303ZE,	NUCLEO_F334R8,	NUCLEO_F401RE,
NUCLEO_F410RB,	NUCLEO_F411RE,	NUCLEO_F412ZG,
NUCLEO_F413ZH,	NUCLEO_F429ZI,	NUCLEO_F439ZI,
NUCLEO_F446RE,	NUCLEO_F446ZE,	NUCLEO_F746ZG,
NUCLEO_F756ZG,	NUCLEO_F767ZI,	NUCLEO_H743ZI,
NUCLEO_H743ZI2,	NUCLEO_L011K4,	NUCLEO_L031K6,
NUCLEO_L053R8,	NUCLEO_L073RZ,	NUCLEO_L152RE,
NUCLEO_L432KC,	NUCLEO_L433RC_P,	NUCLEO_L476RG,
NUCLEO_L486RG,	NUCLEO_L496ZG,	NUCLEO_L496ZG_P,
NUCLEO_L4R5ZI,	NUCLEO_L4R5ZI_P,	NUCLEO_WB55RG,
NUMAKER_IOT_M263A,	NUMAKER_IOT_M487,	NUMAKER_PFM_M453,
NUMAKER_PFM_M487,	NUMAKER_PFM_NANO130,	NUMAKER_PFM_NUC472,
NU_PFM_M2351_NPSA_NS,	NU_PFM_M2351_NPSA_S,	NZ32_SC151,
OC_MBUIINO,	OSHCHIP,	RAPIDIOT_K64F,
RAPIDIOT_KW41Z,	RBLAB_BLENANO,	RBLAB_BLENANO2,
RBLAB_BLENANO_BOOT,	RBLAB_BLENANO_OTA,	RBLAB_NRF51822,

```
RBLAB_NRF51822_BOOT, RBLAB_NRF51822_OTA, RDA5981X,
REALTEK RTL8195AM, RHOMBIO_L476DMW1K, RM6100,
RM7100, RO359B, RZ_A1H,
RZ_A1XX, SAKURAIO_EVB_01, SAMD21G18A,
SAMD21J18A, SAMG55J19, SAML21J18A,
SAMR21G18A, SARA_NBIOT_EVK, SDP_K1,
SDT32620B, SDT32625B, SDT51822B,
SDT52832B, SDT64B, SEEED_TINY_BLE,
SEEED_TINY_BLE_BOOT, SEEED_TINY_BLE_OTA, SILICA_SENSOR_NODE,
SSCI824, STEVAL_3DP001V1, TB_SENSE_1,
TB_SENSE_12, TEENSY3_1, TPM066,
TPMPM3H6, TPMPM3HQ, TPM46B,
TPMPM4G9, TT_M3HQ, TT_M4G9,
TY51822R3, TY51822R3_BOOT, TY51822R3_OTA,
UBLOX_C027, UBLOX_C030_N211, UBLOX_C030_R410M,
UBLOX_C030_R412M, UBLOX_C030_R41XM, UBLOX_C030_U201,
UBLOX_EVA_NINA, UBLOX_EVK_NINA_B1, UBLOX_EVK_ODIN_W2,
UBRIDGE, UHURU_RAVEN, UNO_91H,
USENSE, VBLUNO51, VBLUNO51_BOOT,
VBLUNO51_LEGACY, VBLUNO51_OTA, VBLUNO52,
VK_RZ_A1H, WALLBOT_BLE, WALLBOT_BLE_BOOT,
WALLBOT_BLE_OTA, WIO_3G, WIO_BG96,
WIZWIKI_W7500, WIZWIKI_W7500ECO, WIZWIKI_W7500P,
XADOW_M0, XBED_LPC1768, XDOT_L151CC
```

\$ cd ..

\$ mbed compile -m DISCO\_L072CZ\_LRWN1 -t GCC\_ARM

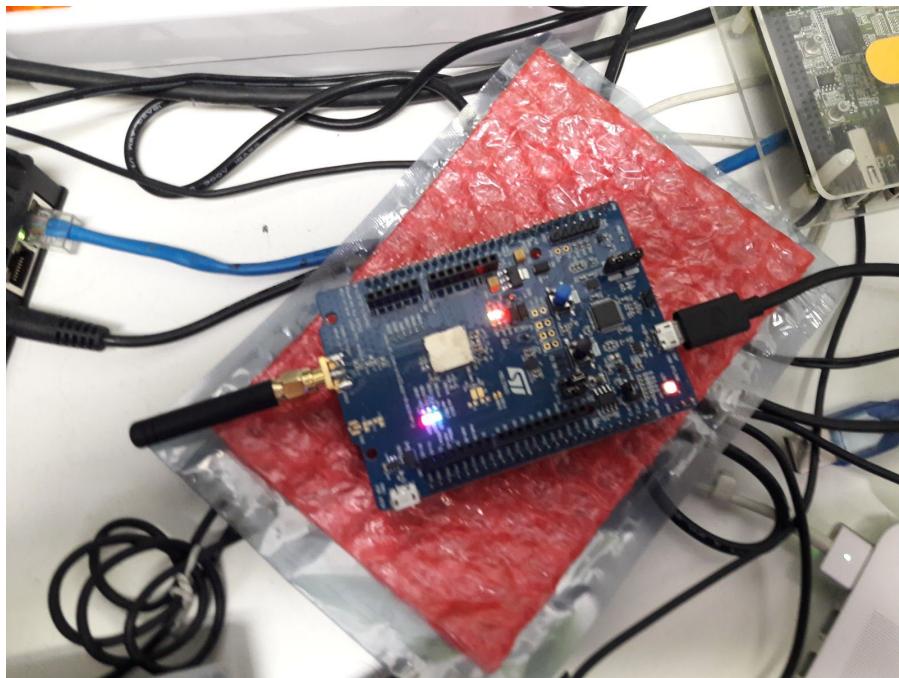
```
[mbed] Working path "/home/chyi/workspace/new_boards/MbedOS/mbed-os-example-lorawan" (program)
[Warning] @: Compiler version mismatch: Have 9.2.1; expected version >= 6.0.0 and < 7.0.0
Building project mbed-os-example-lorawan (DISCO_L072CZ_LRWN1, GCC_ARM)
Scan: mbed-os-example-lorawan
Compile [ 0.1%]: mbed_tz_context.c
Compile [ 0.2%]: MCR20Drv.c
Compile [ 0.3%]: at24mac.cpp
Compile [ 0.5%]: main.cpp
[Warning] main.cpp@160,44: format '%d' expects argument of type 'int', but argument 2 has type 'int32_t'
{aka 'long int'} [-Wformat=]
[Warning] main.cpp@167,70: format '%d' expects argument of type 'int', but argument 3 has type 'int32_t'
{aka 'long int'} [-Wformat=]
[Warning] main.cpp@167,25: '%d' directive writing between 1 and 11 bytes into a region of size 8
[-Wformat-overflow=]
Compile [ 0.6%]: rf_configuration.c
Compile [ 0.7%]: NanostackRfPhyMcr20a.cpp
Compile [ 0.8%]: SX126X_LoRaRadio.cpp
...[중간 생략]...
Compile [ 99.2%]: rtc_api.c
Compile [ 99.3%]: trace_helper.cpp
Compile [ 99.4%]: serial_api.c
Compile [ 99.5%]: sleep.c
Compile [ 99.7%]: trng_api.c
Compile [ 99.8%]: watchdog_api.c
Compile [ 99.9%]: us_ticker.c
Compile [100.0%]: stm_spi_api.c
Link: mbed-os-example-lorawan
```

```
Elf2Bin: mbed-os-example-lorawan
| Module           | .text | .data | .bss |
|-----|-----|-----|-----|
| [fill]           | 126(+126) | 8(+8) | 27(+27) |
| [lib]/c.a        | 29124(+29124) | 2472(+2472) | 89(+89) |
| [lib]/gcc.a      | 12740(+12740) | 0(+0) | 0(+0) |
| [lib]/m.a        | 552(+552) | 0(+0) | 0(+0) |
| [lib]/misc       | 192(+192) | 4(+4) | 28(+28) |
| main.o          | 978(+978) | 0(+0) | 4020(+4020) |
| mbed-lora-radio-drv/SX126X | 54(+54) | 0(+0) | 0(+0) |
| mbed-lora-radio-drv/SX1272 | 292(+292) | 0(+0) | 0(+0) |
| mbed-lora-radio-drv/SX1276 | 8474(+8474) | 0(+0) | 0(+0) |
| mbed-os/components | 80(+80) | 0(+0) | 0(+0) |
| mbed-os/drivers   | 2192(+2192) | 0(+0) | 596(+596) |
| mbed-os/events    | 1560(+1560) | 0(+0) | 0(+0) |
| mbed-os/features  | 26910(+26910) | 4(+4) | 0(+0) |
| mbed-os/hal       | 1732(+1732) | 8(+8) | 130(+130) |
| mbed-os/platform  | 4714(+4714) | 260(+260) | 412(+412) |
| mbed-os/rtos      | 10598(+10598) | 168(+168) | 3164(+3164) |
| mbed-os/targets   | 13180(+13180) | 4(+4) | 1230(+1230) |
| trace_helper.o    | 2(+2) | 0(+0) | 0(+0) |
| Subtotals         | 113500(+113500) | 2928(+2928) | 9696(+9696) |
Total Static RAM memory (data + bss): 12624(+12624) bytes
Total Flash memory (text + data): 116428(+116428) bytes
```

Image: ./BUILD/DISCO\_L072CZ\_LRWAN1/GCC\_ARM/mbed-os-example-lorawan.bin

### c) lorawan firmware 수정해서 보드에 올리기

보드(이하 disco 보드로 통칭)가 도착했으니, 이제부터 firmware를 올려 보도록 하자. :)



[그림 7.2] DISCO\_L072CZ-LRWAN1 보드 전원 연결 모습

먼저 mbed-os-example-lorawan/**mbed\_app.json** 파일을 열고, KR920 주파수 설정 및 OTAA 설정 변경을 해 보자.

```

"platform.stdio-convert-newlines": true,
"platform.stdio-baud-rate": 115200,
"platform.default-serial-baud-rate": 115200,
"lora.over-the-air-activation": true,
"lora.duty-cycle-on": true,
"lora.phy": "EU868",
"lora.device-eui": "{ 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00 }",
"lora.application-eui": "{ 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00 }",
"lora.application-key": "{ 0x00, 0x00 }",
},
"K64F": {
    "lora-spi-mosi": "D11",
    "lora-spi-miso": "D12",
    "lora-spi-sclk": "D13",
    "lora-cs": "D10",
    "lora-reset": "A0",
    "lora-dio0": "D2",
    "lora-dio1": "D3",
    "lora-dio2": "D4",
    "lora-dio3": "D5",
    "lora-dio4": "D8",
    "lora-dio5": "D9",
    "lora-rf-switch-ctl1": "NC",
    "lora-rf-switch-ctl2": "NC",
    "lora-txctl": "NC",
    "lora-rxctl": "NC",
    "lora-ant-switch": "A4",
    "lora-pwr-amp-ctl": "NC",
    "lora-tcxo": "NC"
},
"DISCO_L072CZ_LRWAN1": [
    "main_stack_size": 1024,
    "lora-radio": "SX1276",
    "lora-spi-mosi": "PA_7",
    "lora-spi-miso": "PA_6",
    "lora-spi-sclk": "PB_3",
    "lora-cs": "PA_15",
    "lora-reset": "PC_0",
    "lora-dio0": "PB_4",
]

```

[그림 7.3] mbed\_app.json 파일 내용

```

    "target_overrides": {
      "*": {
        "platform.stdio-convert-newlines": true,
        "platform.stdio-baud-rate": 115200,
        "platform.default-serial-baud-rate": 115200,
        "lora.over-the-air-activation": true,
        "lora.duty-cycle-on": true,
        "lora.phy": "KR920",
        "lora.device-eui": "{ 0x11, 0x11, 0x11, 0x11, 0x22, 0x22, 0x22, 0x22 }",
        "lora.application-eui": "{ 0x01, 0x02, 0x03, 0x04, 0x10, 0x20, 0x30, 0x40 }",
        "lora.application-key": "{ 0x04, 0x40, 0x96, 0x53, 0x92, 0xfd, 0x00, 0x01, 0xfb, 0x27, 0xb7, 0x
85, 0x3a, 0x6a, 0xdb, 0x85 }"
      },
    }
  }
}

```

[그림 7.4] mbed\_app.json 파일 수정

참고: LoRaServer에서는 application-eui를 사용하지 않는다.

수정한 내용을 반영하기 위해 코드를 재 build해 보도록 하자.

\$ mbed compile -m DISCO\_L072CZ\_LRWAN1 -t GCC\_ARM

Build에 문제가 없으니, 바로 flash writing을 해 보기로 하자.

chyi@mars:~/workspace/new\_boards/MbedOS/mbed-os-example-lorawan\$ **mbed compile -m DISCO\_L072CZ\_LRWAN1 -t GCC\_ARM -f**

→ Flash에 write하자.

```

[mbed] Working path "/home/chyi/workspace/new_boards/MbedOS/mbed-os-example-lorawan" (program)
[Warning] @: Compiler version mismatch: Have 9.2.1; expected version >= 6.0.0 and < 7.0.0
Building project mbed-os-example-lorawan (DISCO_L072CZ_LRWAN1, GCC_ARM)
Scan: mbed-os-example-lorawan
Link: mbed-os-example-lorawan
Elf2Bin: mbed-os-example-lorawan

| Module           |   .text |   .data |   .bss |
|-----|-----|-----|-----|
| [fill]          | 126(+0) |  8(+0) | 27(+0) |
| [lib]/c.a       | 29124(+0) | 2472(+0) |  89(+0) |
| [lib]/gcc.a     | 12740(+0) |  0(+0) |  0(+0) |
| [lib]/m.a       |  552(+0) |  0(+0) |  0(+0) |
| [lib]/misc      |  192(+0) |  4(+0) | 28(+0) |
| main.o          |  978(+0) |  0(+0) | 4020(+0) |
| mbed-lora-radio-drv/SX126X |  54(+0) |  0(+0) |  0(+0) |

```

mbed-lora-radio-drv/SX1272	292(+0)	0(+0)   0(+0)
mbed-lora-radio-drv/SX1276	8474(+0)	0(+0)   0(+0)
mbed-os/components	80(+0)	0(+0)   0(+0)
mbed-os/drivers	2192(+0)	0(+0)   596(+0)
mbed-os/events	1560(+0)	0(+0)   0(+0)
mbed-os/features	27678(+0)	4(+0)   0(+0)
mbed-os/hal	1732(+0)	8(+0)   130(+0)
mbed-os/platform	4714(+0)	260(+0)   412(+0)
mbed-os/rtos	10598(+0)	168(+0)   3164(+0)
mbed-os/targets	13180(+0)	4(+0)   1230(+0)
trace_helper.o	2(+0)	0(+0)   0(+0)
Subtotals	114268(+0)	2928(+0)   9696(+0)

Total Static RAM memory (data + bss): 12624(+0) bytes

**Total Flash memory (text + data): 117196(+0) bytes**

Image: ./BUILD/DISCO\_L072CZ\_LRWAN1/GCC\_ARM/mbed-os-example lorawan.bin

**주의:** 장치를 PC에서 인식 못할 경우, PC를 재 부팅해 주면 해결될 수 있다.

Flash에 writing은 정상적으로 진행된 듯 한데, LoRaWAN 연결(<=> LoRaServer)이 안되는 것 같다.

**참고:** LoRaServer에 disco board를 Application으로 등록하는 절차는 (반복되는 내용이라) 별도로 정리하지 않았다.

```

OnRxTimeout
    OnTxDone
        txdone
Mbed LoRaWANstack initialized

CONFIRMED message retries : 3

Adaptive data rate (ADR) - Enabled

Connection - In Progress ...

CTRL-A Z for help | 115200 8N1 | NOR | Minicom 2.7.1 | VT102 | Offline | ttyACM0

```

[그림 7.5] minicom(/dev/ttyACM0, 115200, 8N1) 실행 모습

어라, LoRaServer application key 설정이 잘 못 되었다. 아래와 같이 다시 조정해 보니, 정상 동작한다.

The screenshot shows the LoRa Server web interface. The left sidebar has a tree view with nodes like Network-servers, Gateway-profiles, Organizations, All users, loraserver, Org. settings, Org. users, Service-profiles, Device-profiles, Gateways, Applications, and Multicast-groups. The main content area is titled 'Applications / disco / Devices / disco'. It has tabs for DETAILS, CONFIGURATION, KEYS (OTAA), ACTIVATION, and DEVICE. Under KEYS (OTAA), there is a section for 'Application key \*' containing the value '04 40 96 53 92 fd 00 01 fb 27 b7 85 3a 6a db 85'. Below it, there is a note: 'For LoRaWAN 1.0 devices. In case your device supports LoRaWAN 1.1, update the device-profile first.' There is also a 'Gen Application key' field with the value '00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00'. At the bottom right of this section is a 'SET DEVICE-KEYS' button. The top of the browser window shows the URL '13.124.231.29:7194/#/organizations/1/applications/11/devices/1111111122222222/keys' and the title 'LoRa Server - Chromium'.

[그림 7.6] LoRaServer OTAA 설정 변경

```
Dummy Sensor Value = 9
23 bytes scheduled for transmission
Message Sent to Network Server

Dummy Sensor Value = 11
24 bytes scheduled for transmission
Message Sent to Network Server

Dummy Sensor Value = 13
24 bytes scheduled for transmission
Message Sent to Network Server

Dummy Sensor Value = 15
24 bytes scheduled for transmission
Message Sent to Network Server

Dummy Sensor Value = 17
24 bytes scheduled for transmission
Message Sent to Network Server

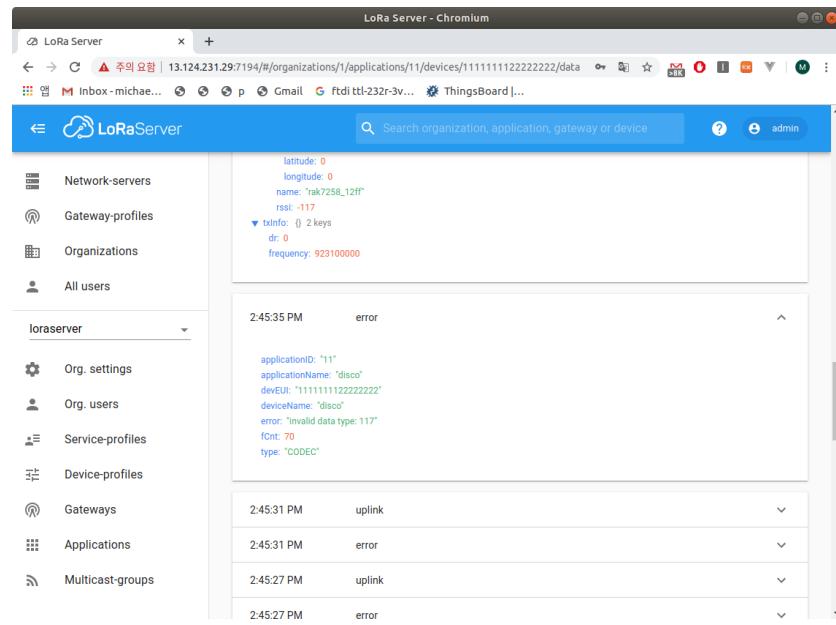
CTRL-A Z for help | 115200 8N1 | NOR | Minicom 2.7.1 | VT102 | Offline | ttyACM0
```

[그림 7.7] minicom(/dev/ttyACM0, 115200, 8N1) 실행 모습 - OK

Time	Event	Action
2:44:31 PM	uplink	▼
2:44:31 PM	error	▼
2:44:27 PM	uplink	▼
2:44:27 PM	error	▼
2:44:22 PM	uplink	▼
2:44:22 PM	error	▼

[그림 7.8] LoRaServer application page 모습

근데, 왜 여러 packet이 보일까 ?



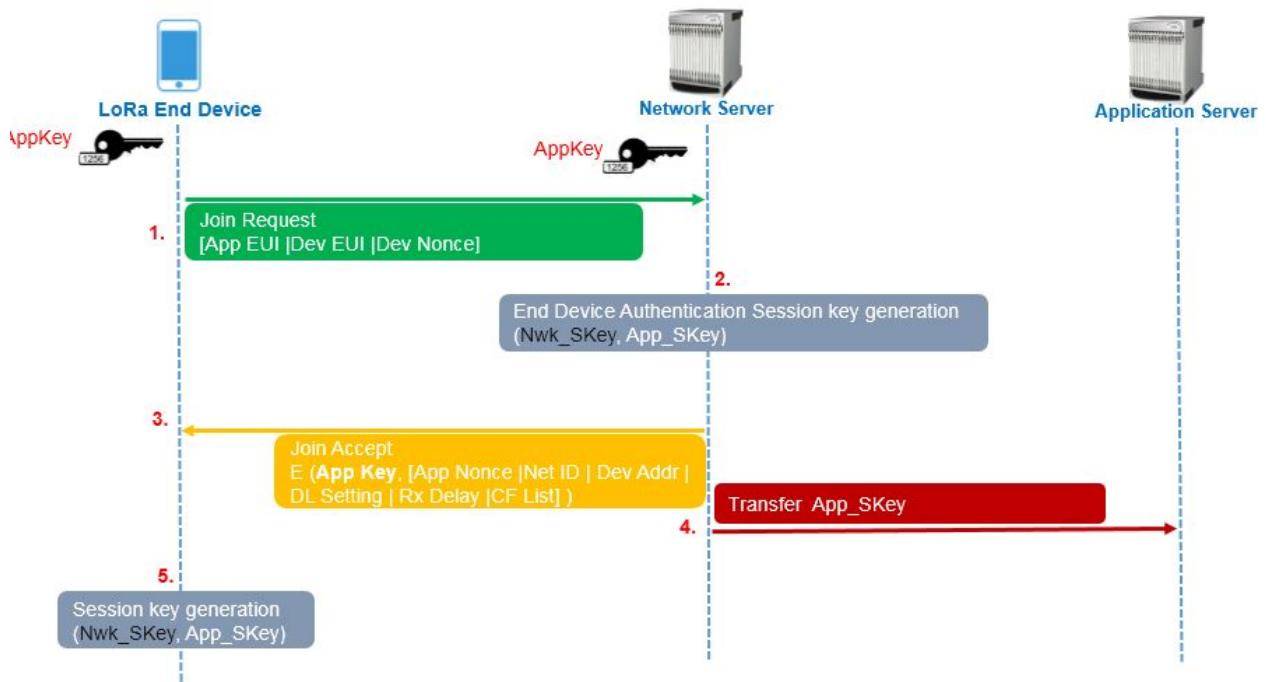
[그림 7.9] LoRaServer application page - error packet

내용을 보니, **error: "invalid data type: 117"** 인데 ... 실제 sensor를 달지 않아서 제대로 된 data가 없어서 그런가? **아직은 정확한 이유를 알 수 있으니, 좀 더 두고 볼 일이다. :(**

**[나중에 정리한 것임] ThingsBoard로 전달할 message format(Codec 설정)이 안 맞아서 발생한 에러이다.**

<여기서 잠깐!>

Over-The-Air Activation



[그림 7.10] OTAA Procedure

#### D) lorawan example 코드 분석하기

<TBD>

#### E) PM2.5 미세먼지 센서 붙여 보기

<TBD>

