



SPN v3.0(a.k.a IoTSec) - LoRa Security Project *(Part I)*

08.21.2019 ~

Doc. Revision: 1.5

We Secure the Internet of Things with 2STON SPN.

Chunghan.Yi(michael@2ipco.com)
R&D Center
2ip Inc.

Table of Contents

Part I.

1. LoRa 개요
2. RAKWireless RAK831 LoRaWAN Kit
3. Dragino IoT Kit v2
4. Dragino LG308 LoRa Gateway
5. RAKWireless **RAK7258** LoRa Gateway
6. RAKWireless **RAK7249** Outdoor LoRa Gateway
7. MatchX **MX1702** LoRa Gateway(**LBT 지원 모델**)

Part II.

8. MultiTech **MultiConnect Conduit IP67** LoRa Gateway
9. LoRa Gateway에 SPN S/W Porting하기
10. **LoRaServer Project 1 - 설치 & 운용**
11. **LoRaServer Project 2 - External Interface**
12. **ThingsBoard Integration - 대박 :)**
13. Our LoRa Viewer: **OLoRa(= ThingsBoard)**

Part III.

14. Outdoor LoRa Node - Libelium
15. URSAlink LoRa Products
16. LoRaWAN Stack

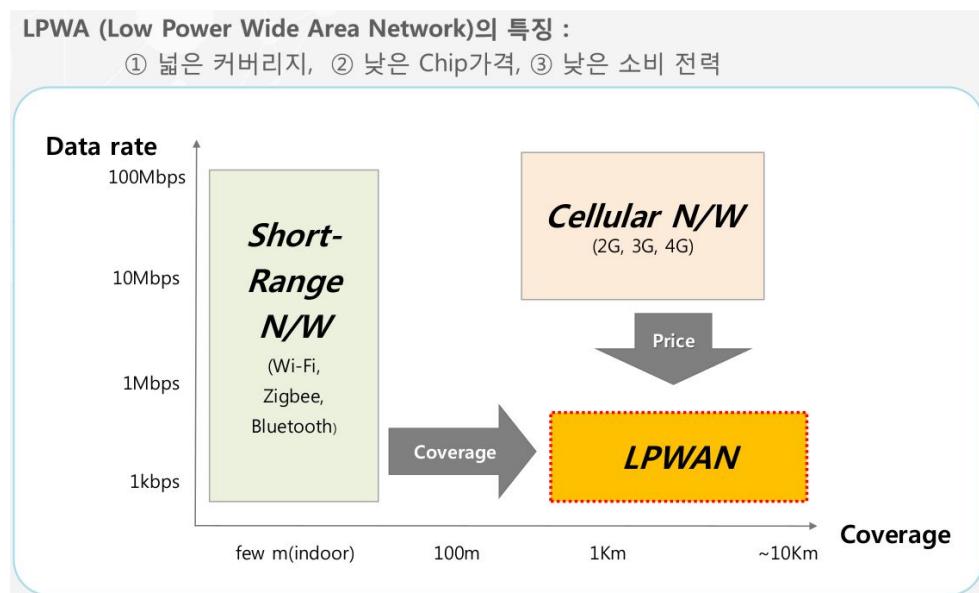
Part I. Chapter 1 ~ Chapter 7

1. LoRa 개요

LoRa는 'Long Range'의 약자로 광범위한 커버리지와 적은 대역폭, 긴 배터리 수명과 저전력 등의 특징을 갖춘 IoT 전용 네트워크 기술로 저전력 장거리 통신 기술(LPWA) 중 유력한 후보에 속한다.

"NB-IoT는 기존 LTE 대역을 활용하여 통신 가능하다. 현재 200여 개국에서 650여 개 LTE 망이 운영되고 있어, NB-IoT 망은 향후 빠르게 확대될 전망이다. NB-IoT 망이 LTE 주파수 사업권을 가진 통신사업자 위주로 운영되는 반면, 비면허 대역을 활용하는 LoRa 망은 통신사업자 외에 중소기업들도 운영할 수 있고 사설망 구축도 용이하다."

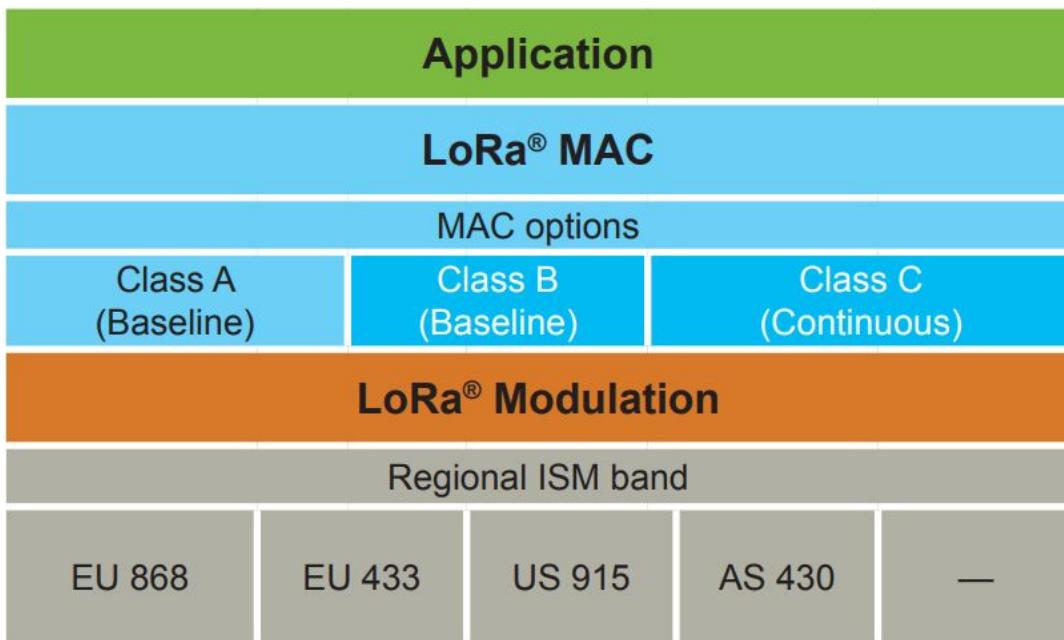
참고: LoRaWAN – LoRa MAC layer implementation



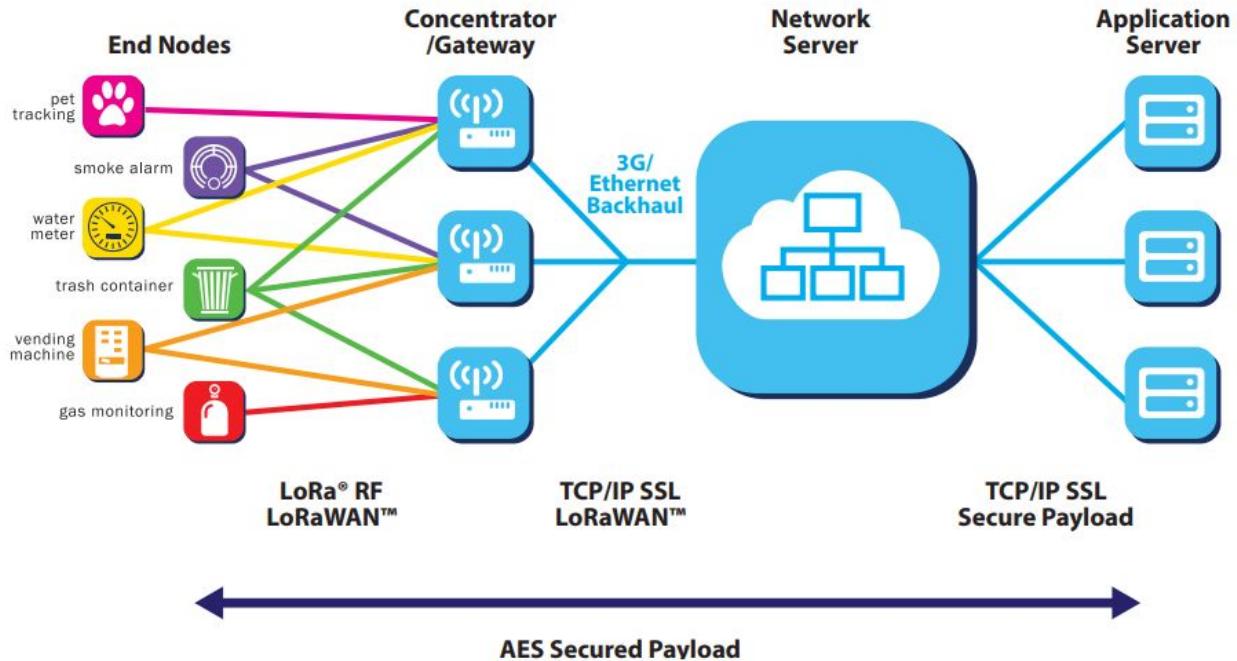
[그림 1.1] LoRa 기술 개요(1) - LPWAN

구분	LoRa	Sigfox	LTE-M	LTE NB-IoT
주파수 대역	▪ 비 면허 대역 (920MHz)	▪ 비 면허 대역 (920MHz)	▪ 면허 대역 (800MHz, 1.8GHz, ...)	▪ 면허 대역 (In-band, Guard-band)
표준화 단체	▪ LoRa Alliance	▪ ETSI	▪ 3GPP	▪ 3GPP
표준화 단계	▪ 표준 완료	▪ 표준 완료	▪ Category 0/1: 표준 완료 ▪ Category M1: 표준화 진행 중 (Rel.13)	▪ 표준 진행 중 (Rel.13)
Max. Data Rate	▪ 5.4kbps	▪ 1kbps	▪ Cat.1: DL/UL 10/5Mbps ▪ Cat.0: DL/UL 1Mbps ▪ Cat.M1: DL/UL 0.2~1Mbps	▪ 수백kbps
상용화	▪ 상용화 (SK Telecom, Orange 등 글로벌 사업자 다수)	▪ 상용화	▪ Cat.1: VZW, SKT등 서비스中	▪ '17년 상반기 예상

[그림 1.2] LoRa 기술 개요(2) - 다른 유사 기술과의 비교

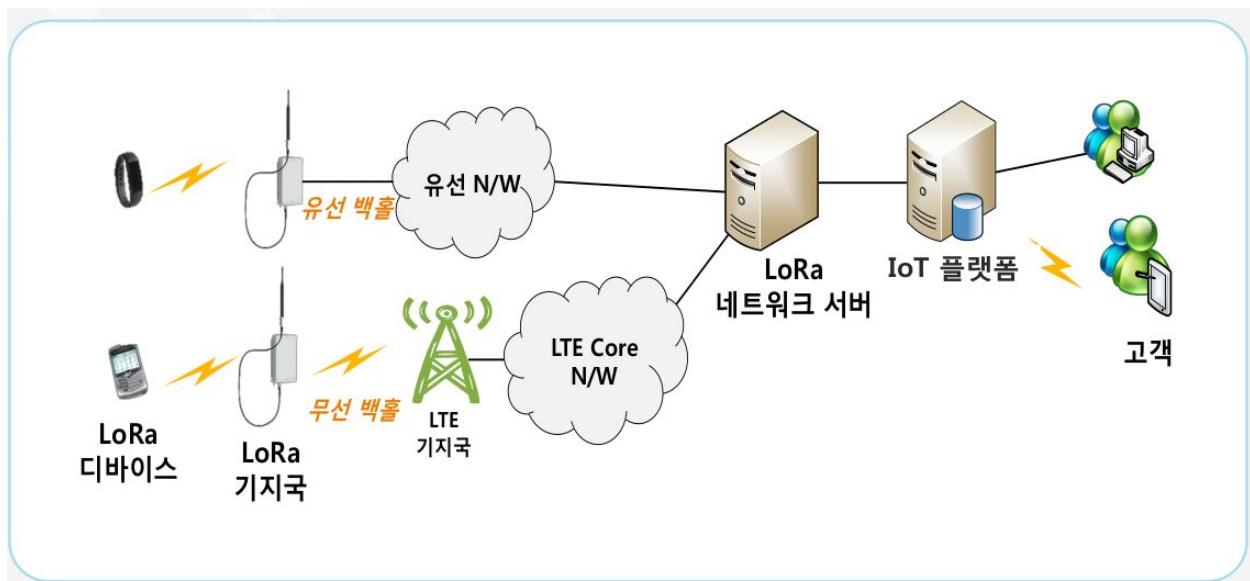


[그림 1.3] LoRaWAN 아키텍쳐(1)



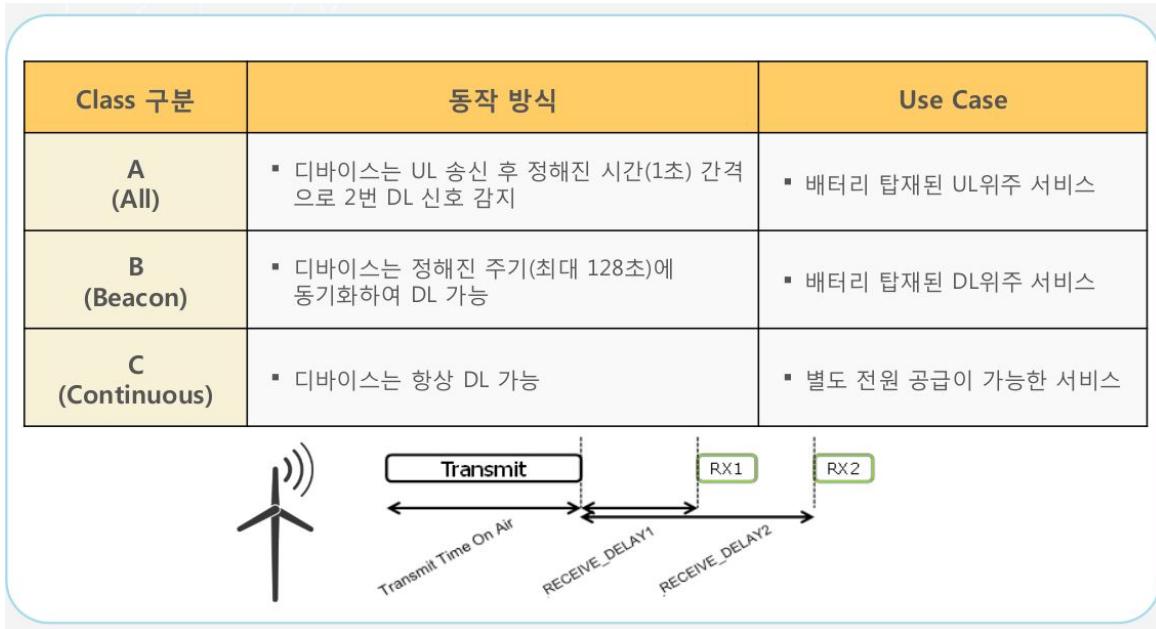
[그림 1.4] LoRaWAN 아키텍쳐(2)

- ❑ LoRa 디바이스: 센서 데이터를 수집하여 LoRa 표준 규격으로 무선 송신(전달)하는 장치
- ❑ LoRa 기지국: LoRa 디바이스와 LoRa 네트워크 서버간의 패킷 forwarding 역할 담당 Gateway(LoRa Gateway)
- ❑ LoRa 네트워크 서버: LoRa MAC 프로토콜 처리, Device 인증 및 관리, IoT 플랫폼 연동 역할을 하는 서버

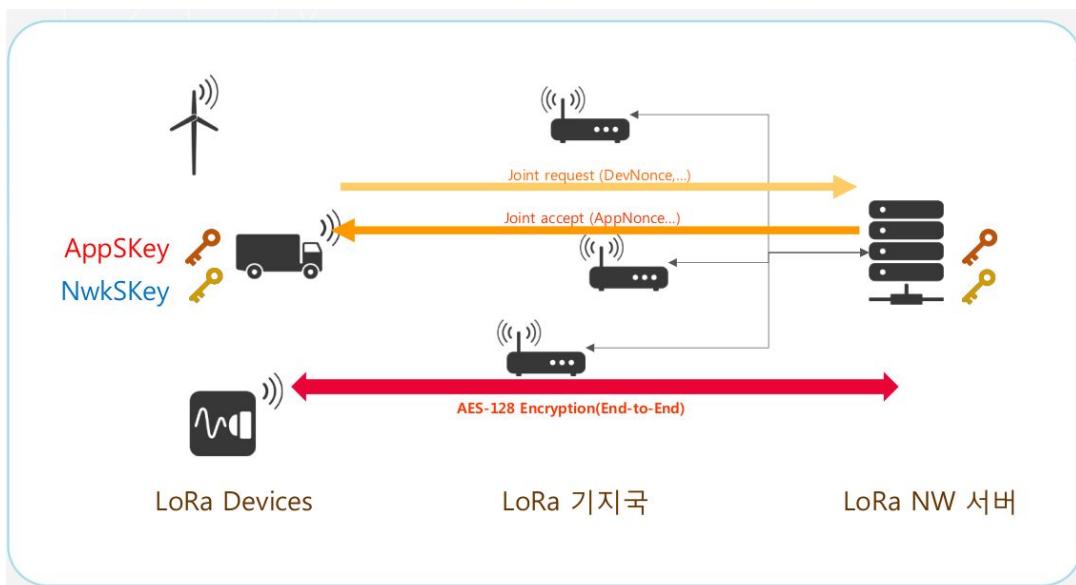


[그림 1.5] LoRa 네트워크 구성(SKT 예)

LoRa 디바이스는 DL 수신 가능 시간에 따라 Class A/B/C 타입으로 구분 가능하다.



[그림 1.6] LoRa Device Classes



[그림 1.7] LoRa Security Protocol(1)

□ Device EUI (Extended Unique ID) (64bit)

Device EUI (64 bit): 개통 절차에서 Provisioning되는 값이고, 단말 별로 할당

IEEE EUI64 포맷(Globally Unique)

□ Application EUI(Extended Unique ID) (64bit)

App EUI (64 bit): 개통 절차에서 Provisioning 되는 값이고, 단말 별로 할당

서비스 별로 구분되는 구분자이고, SKT에서 정의

□ Application Key (128bit)

Application Key(128 bit): 개통 절차에서 할당되며, 단말 별로 할당

AppKey 할당 과정에서 생성하며, NW/App Session Key를 생성하기 위한 root key

[그림 1.8] LoRa Security Protocol(2)

참고: 위의 내용은 SKT LoRa 문서를 참조하여 정리한 것이라, "SKT에서 정의"라는 문구가 포함되어 있다. 실제로 우리(2ip)가 사용할 LoRa Server(loraserver.io)에서는 App EUI 값은 사용하지 않는다.

<몇가지 용어 정리>

DEV EUI - Unique ID code for a particular device.

APP EUI - ID code for an Application defined in TTN.

Device Addr: LoRa packet에 들어가는 device addr임. 4 bytes

APP Key - NW/App Session Key를 생성하기 위한 root key

Network Session Key (NwkSKey) - node(device)와 network server간의 MIC(integrity checking) 용으로 사용

Application Session Key (AppSKey) - node(device)와 network server or application 간의 실제 payload를 암호화하는데 사용되는 key

OTAA (Over-the-Air Activation) - join message를 교환하는 방식, end device와 server는 각각 DevNonce, AppNonce를 교환하고, 이걸 이용해 NwkSKey, AppSkey를 생성, ABP 보다 보안성 높음.

ABP (Activation by Personalization) - join message를 교환하지 않는 방식, end device에 DevAddr, NwkSKey, AppSKey가 할당되고, 서버에 저장, pre-shared secret 느낌

	Europe	North America	China	Korea	Japan	India
Frequency band	867-869MHz	902-928MHz	470-510MHz	920-925MHz	920-925MHz	865-867MHz
Channels	10	64 + 8 +8				
Channel BW Up	125/250kHz	125/500kHz				
Channel BW Dn	125kHz	500kHz				
TX Power Up	+14dBm	+20dBm typ (+30dBm allowed)				
TX Power Dn	+14dBm	+27dBm				
SF Up	7-12	7-10				
Data rate	250bps- 50kbps	980bps-21.9kbps				
Link Budget Up	155dB	154dB	In definition by Technical Committee	In definition by Technical Committee	In definition by Technical Committee	
Link Budget Dn	155dB	157dB				

[그림 1.9] LoRaWAN 주파수 대역

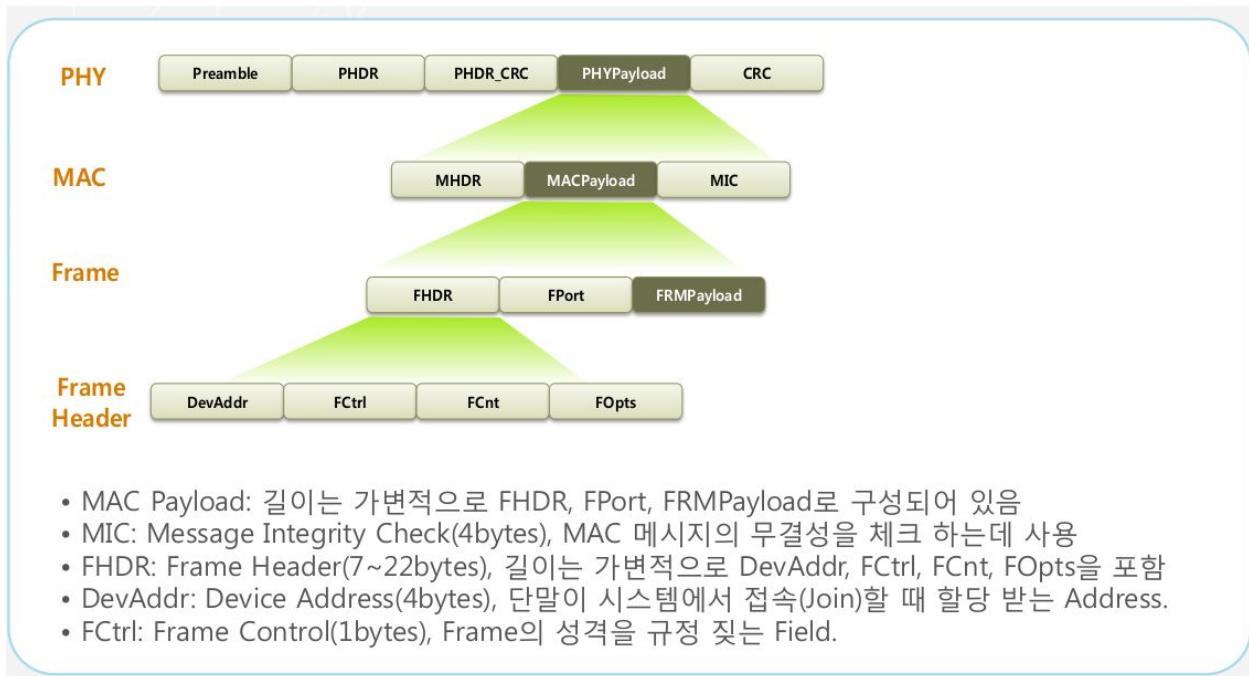
Channel 번호	Center Freq. (BW: 125kHz)
25	921.9 MHz
26	922.1 MHz
27	922.3 MHz
28	922.5 MHz
29	922.7 MHz
30	922.9 MHz
31	923.1 MHz
32	923.3 MHz

표. SKT LoRa 사용 주파수 대역

- LoRa 단말은 SX1276 칩을 사용하고, 한국 기술기준에 따라 최대 출력은 25번 채널에서 10mW이고, 26번~32번 채널에서 25mW임
- LoRa 기지국은 SX1301 칩을 사용하고, 한국 기술기준에 따라 최대 출력은 25번 ~32번 채널에서 200mW임

[그림 1.10] LoRa 한국 주파수 대역

LoRa 메시지 중 MAC protocol과 관련 있는 정보들은 아래 그림과 같이 MAC/Frame header에 포함되어 있다.



[그림 1.11] LoRa Mac Layer

기타 LoRa 관련 내용은 아래와 인터넷에서 쉽게 찾을 수 있다.

<https://blog.naver.com/PostView.nhn?blogId=opusk&logNo=220984482677>

<LoRa Use Cases>

<https://tektelic.com/usecases/>

Smart Cities

- » Waste Management
- » Parking
- » Metering (Water, Gas)
- » Smart Signs
- » Streetlighting
- » Air Quality Monitoring
- » Tracking
- » Flood Monitoring
- » Connected Airports
- » Bridge Stress Monitoring
- » Urban Agriculture Management
- » Snow Removal and Street Cleaning Management
- » Smart Public Transit



[그림 1.12] LoRa Use Cases - Smart Cities

Building Management

- » Temperature and Humidity Monitoring
- » Room Occupancy
- » Motion Detection
- » Chair Occupancy
- » Desk Presence Monitoring
- » Point of Entry Security / Access
- » Building Entrance Counter (Virtual Turnstile)
- » Indoor/Outdoor Illumination Management
- » Smart Maintenance
- » Water Consumption Monitoring
- » Leak Detection
- » Fire/Smoke/Noxious Gas Detection
- » Smart Boardrooms
- » Smart Restrooms
- » Smart Elevators
- » Indoor Air Quality Monitoring
- » Automated Customer Feedback Systems

[IoT for Smart Buildings White Paper](#)



[그림 1.13] LoRa Use Cases - Building Management

Smart Agriculture

- » Asset Tracking
- » Environmental Monitoring
- » Irrigation / Soil Moisture Monitoring
- » Smart Water Management
- » Precision Agriculture
- » Pest Control
- » Livestock Tracking and Monitoring
- » Remote Equipment Management
- » Smart Fish Farming



[그림 1.14] LoRa Use Cases - Smart Agriculture

Oil & Gas

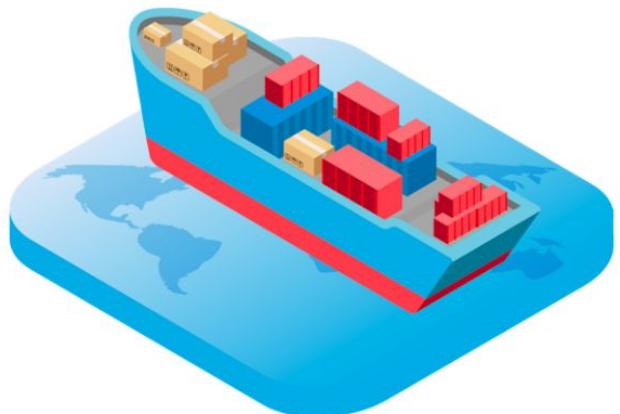
- » Lone Worker Safety
- » Wellhead / Oil Rig Monitoring
- » Pipeline Monitoring
- » Preventative Maintenance
- » Asset Tracking and Remote Monitoring
- » Environmental Monitoring
- » Asset Integrity
- » Tank Monitoring



[그림 1.15] LoRa Use Cases - Oil & Gas

Smart Tracking

- » Pallet Tracking
- » Vehicle Tracking
- » Animal Tracking
- » Asset Tracking
- » Railway Cars
- » Lone worker safety
- » Geo-fencing



[그림 1.16] LoRa Use Cases - Smart Tracking

Food Services

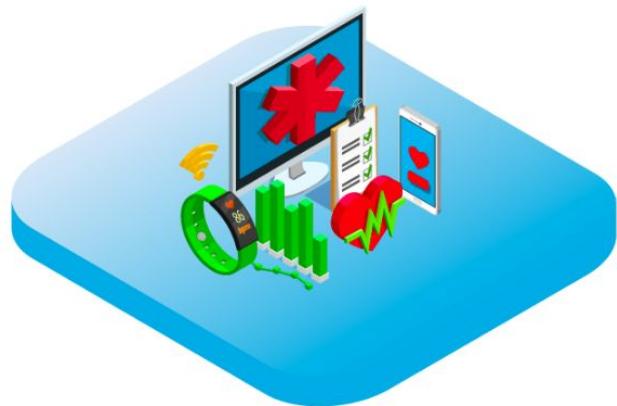
- » Cold Chain Monitoring
- » Spoilage Detection
- » Smart Cleaning and Maintenance
- » Appliance Maintenance and Usage Monitoring
- » Pest Control



[그림 1.17] LoRa Use Cases - Food Services

Health & Safety

- » Senior Citizen Care
- » Student Tracking and Monitoring
- » Smart Wearables
- » Personal Safety/Security
- » Air Quality Monitoring
- » Child Tracking



[그림 1.18] LoRa Use Cases - Health & Safety

Logistics

- » Inventory Management
- » Smart Warehousing
- » Asset Tracking (Indoor/Outdoor)
- » Process Automation
- » Machine Usage Monitoring (Preventative Maintenance)



[그림 1.19] LoRa Use Cases - Logistics

Smart Metering

- » Water Metering
- » Electric Metering
- » Gas Metering



[그림 1.20] LoRa Use Cases - Smart Metering

Recreation

- » Fitness Monitors
- » Elite Sports Performance Monitoring
- » Golf Cart Tracking
- » Public Use Space Maintenance and Scheduling (Courts, Fields, Arenas etc.)
- » Orienteering



[그림 1.21] LoRa Use Cases - Recreation

Children Monitoring

- » Real Time School Bus Tracking
- » Child Location Tracking and Monitoring
- » Off Site (Field Trip) Student Monitoring
- » New Driver Tracking and Monitoring



[그림 1.22] LoRa Use Cases - Children Monitoring

Early Fire Detection

- » N2, CO, CO2, PM10 Monitoring
- » Rapid Temperature and Humidity Detection
- » Lightning/Thunder Detection and Count
- » Near Real-Time Alerts and Notifications
 - Current technologies often have up to 24 hour cycle to visualize collected data



[그림 1.23] LoRa Use Cases - Early Fire Detection

Connected Airports

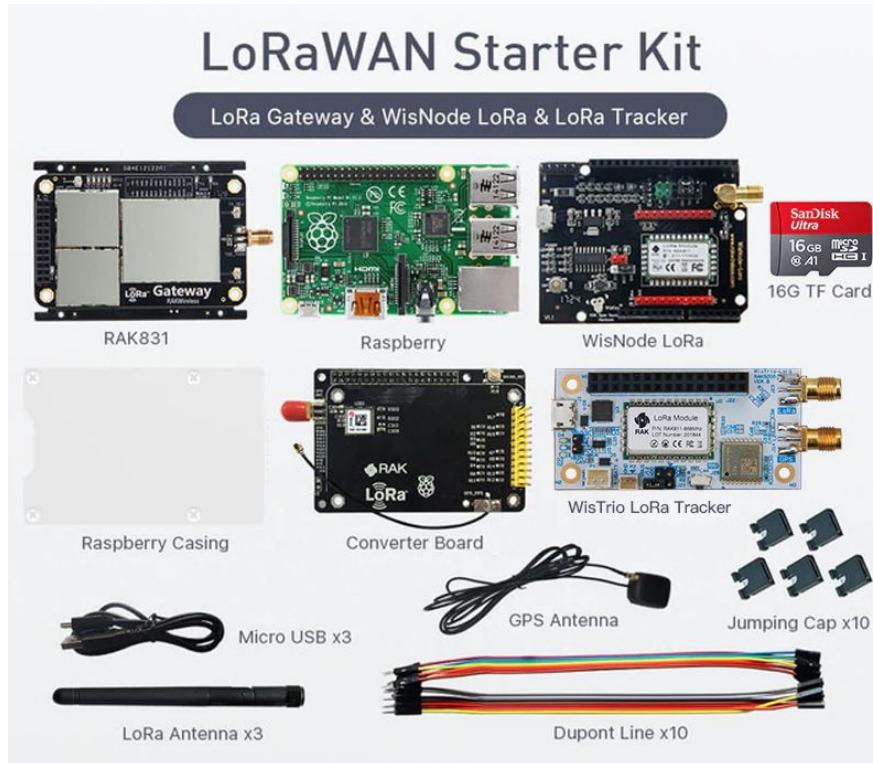
- » Location Tracking
 - Fleet vehicles and equipment
 - Luggage carts
 - Staff
 - In Transit Cargo and Luggage
- » Cold chain monitoring
- » Equipment Performance Monitoring for Maintenance Scheduling
- » HVAC Automation and Control
- » People Counting
- » Point of Entry Security/Access



[그림 1.24] LoRa Use Cases - Connected Airports

2. RAKWireless RAK831 LoRaWAN Kit

이번 절에서는 맛보기 차원에서 RAKWireless 사에서 판매하는 RAK831 LoRa Kit를 사용하여 LoRa망을 구축해 보도록 하겠다. 3절에서 설명할 Dragino IoT Kit를 모든 인원에게 전달할 수 있으니, 일부 인원은 RAK831 Kit를 활용하도록 하는게 좋겠다.



[그림 2.1] RAK831 LoRaWAN Starter Kit

주: 실제 작년에 구매한 Kit의 내용은 위의 내용과 약간 차이가 있다. 특히 WisTrio LoRa Tracker가 보이질 않는다.

자세한 내용은 아래 site(짝퉁 영어로 쓰여 있음)를 참조하였다.

<Reference 1>

<https://www.hackster.io/naresh-krish/getting-started-with-the-rak831-lora-gateway-and-rpi-3-e3351d>

<Reference 2>

<https://create.arduino.cc/projecthub/naresh-krish/using-the-rak811-lora-module-with-arduino-a38de8>

<Reference 3>

<https://www.hackster.io/naresh-krish/getting-started-with-the-rak811-lora-node-67f157>

Raspberry Pi 3에 Raspbian을 올리고, OS를 upgrade하는 등의 간단한 작업은 본 문서에서는 설명하지 않기로 하겠다. 즉, 본 문서에서는 LoRaWAN 망 구축에 반드시 필요한 내용만을 설명하도록 할 것이다. 따라서 보다 구체적인 내용은 위의 site 내용을 꼼꼼히 읽어보기 바란다.

1) RAK831(LoRaWAN) ⇔ Raspberry Pi 3(Host) 보드 연결하기

먼저 Raspberry Pi와 RAK831 보드를 아래와 같이 연결하도록 하자. 둘간을 연결하기 위해서는 아래 그림 2.5를 참조해야만 한다.



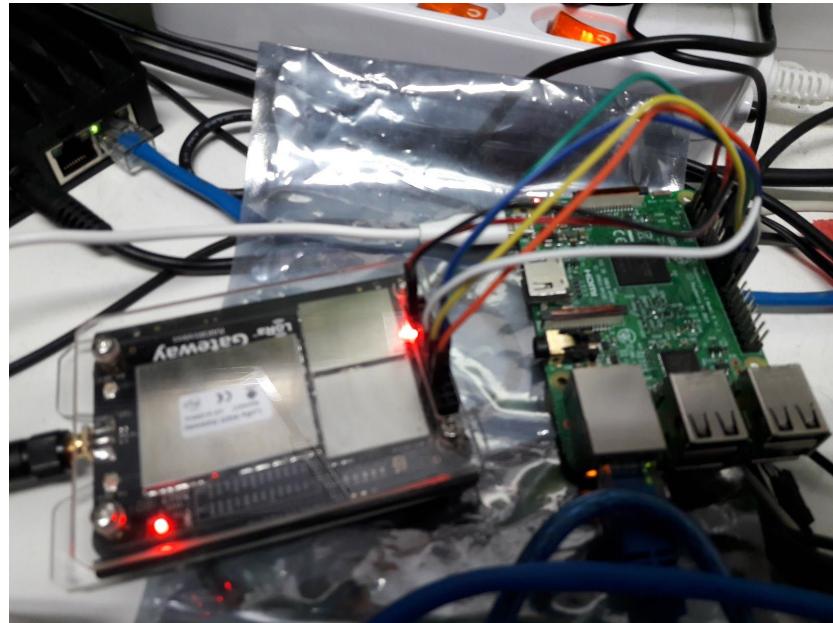
[그림 2.2] RAK831 LoRaWAN Gateway 보드



[그림 2.3] Raspberry Pi 3 보드(Host Computer)

참고: Raspberry Pi 3(이하 RPi3)와 RAK831은 SPI를 통해 통신을 하는 구조이다. 따라서 사전에 RPi3에서는 SPI를 enable 시켜 주어야만 한다.

주의: jumper cable 연결 시에는 전원(Vcc) 및 Ground pin을 정확히 연결해 주어야 한다. 만일 잘못 연결할 경우, board에 무리(damage)가 갈 수 있으니, 주의하기 바란다.



[그림 2.4] RAK831 LoRaWAN Gateway와 RAK811 LoRa Node 연결 모습

RAK 831 Pin	Description on silk screen	RPi physical pin
1	+5V	2
3	GND	6
19	RST (Reset pin)	22
18	SCK (SPI Clock)	23
17	MISO	21
16	MOSI	19
15	CSN (Chip Select)	24

[그림 2.5] RAK831 ⇄ Raspberry Pi 3(Host) 간의 Interfacing pin layout

2) RPi3에 Semtech packet forwarder program 설치하기

이와 관련해서는 아래 site를 참조할 필요가 있다.

<https://github.com/ttn-zh/ic880a-gateway>

```
$ git clone -b spi https://github.com/ttn-zh/ic880a-gateway.git ~/ic880a-gateway
$ cd ~/ic880a-gateway
$ sudo ./install.sh spi      ← spi option을 꼭 줘야 하는지 잘 모르겠다 ?
```

```
The Things Network Gateway installer
Version spi
Updating installer files...
Already up to date.
Gateway configuration:
Detected EUI B827EBFFFFE2881FF from enxb827eb2881ff
Do you want to use remote settings file? [y/N]y
Updating hostname to 'ttn-gateway'...
/opt/ttn-gateway /home/pi/ic880a-gateway
Cloning into 'lora_gateway'...
...
...
...
Cloning into 'gateway-remote-config'...
remote: Enumerating objects: 6466, done.
remote: Total 6466 (delta 0), reused 0 (delta 0), pack-reused 6466
Receiving objects: 100% (6466/6466), 2.23 MiB | 1.91 MiB/s, done.
Resolving deltas: 100% (3519/3519), done.
/opt/ttn-gateway/gateway-remote-config /opt/ttn-gateway /home/pi/ic880a-gateway
/opt/ttn-gateway /home/pi/ic880a-gateway
/home/pi/ic880a-gateway
Gateway EUI is: B827EBFFFFE2881FF
The hostname is: ttn-gateway
Open TTN console and register your gateway using your EUI:
https://console.thethingsnetwork.org/gateways

Installation completed.
Created symlink /etc/systemd/system/multi-user.target.wants/ttn-gateway.service →
/lib/systemd/system/ttn-gateway.service.
The system will reboot in 5 seconds...
Connection to 192.168.5.51 closed by remote host.
Connection to 192.168.5.51 closed.
```

정상적으로 설치가 되면, 아래의 내용이 보일 것이다.

```
pi@ttn-gateway:/opt $ cd ttn-gateway/
pi@ttn-gateway:/opt/ttn-gateway $ ls -la
total 64
drwxr-xr-x 6 root root 4096 Aug 20 07:01 .
drwxr-xr-x 7 root root 4096 Aug 20 07:00 ..
drwxr-xr-x 2 root root 4096 Aug 20 07:01 bin
drwxr-xr-x 3 root root 45056 Aug 20 07:01 gateway-remote-config
drwxr-xr-x 8 root root 4096 Aug 20 07:00 lora_gateway
drwxr-xr-x 10 root root 4096 Aug 20 07:00 packet_forwarder
pi@ttn-gateway:/opt/ttn-gateway $ cd bin/
pi@ttn-gateway:/opt/ttn-gateway/bin $ ls -la
total 24
drwxr-xr-x 2 root root 4096 Aug 20 07:01 .
drwxr-xr-x 6 root root 4096 Aug 20 07:01 ..
-rw-r--r-- 1 root root 7440 Aug 20 07:01 global_conf.json
lrwxrwxrwx 1 root root 60 Aug 20 07:01 local_conf.json ->
/opt/ttn-gateway/gateway-remote-config/B827EBFFE2881FF.json
lrwxrwxrwx 1 root root 59 Aug 20 07:01 poly_pkt_fwd ->
/opt/ttn-gateway/packet_forwarder/poly_pkt_fwd/poly_pkt_fwd
-rwxr-xr-x 1 root root 1882 Aug 20 07:01 start.sh
```

위의 설치 과정에서 아래 파일이 생성되게 된다. 아래 내용을 보면 알 수 있듯이, RPi3 부팅 시, 이 파일을 통해 /opt/ttn-gateway/bin/start.sh script가 구동되게 된다.

```
</lib/systemd/system/ttn-gateway.service>
-----
[Unit]
Description=The Things Network Gateway

[Service]
WorkingDirectory=/opt/ttn-gateway/bin/
ExecStart=/opt/ttn-gateway/bin/start.sh           <== 애가 시작 파일임.
SyslogIdentifier=ttn-gateway
Restart=on-failure
RestartSec=5

[Install]
WantedBy=multi-user.target
```

아래 start.sh 파일의 내용을 보면 알겠지만, 제일 먼저 RAK831 board를 reset 한 후, 마지막에 **poly_pkt_fwd** daemon을 구동시켜 준다. 애(poly_pkt_fwd)가 LoRa packet을 서버로 전달해 주는 녀석으로 보인다.

```
# Reset iC880a PIN
SX1301_RESET_BCM_PIN=25
echo "$SX1301_RESET_BCM_PIN" > /sys/class/gpio/export
echo "out" > /sys/class/gpio/gpio$SX1301_RESET_BCM_PIN/direction
echo "0" > /sys/class/gpio/gpio$SX1301_RESET_BCM_PIN/value
sleep 0.1
echo "1" > /sys/class/gpio/gpio$SX1301_RESET_BCM_PIN/value
sleep 0.1
echo "0" > /sys/class/gpio/gpio$SX1301_RESET_BCM_PIN/value
sleep 0.1
echo "$SX1301_RESET_BCM_PIN" > /sys/class/gpio/unexport

# Test the connection, wait if needed.
while [[ $(ping -c1 google.com 2>&1 | grep " 0% packet loss") == "" ]]; do
    echo "[TTN Gateway]: Waiting for internet connection..."
    sleep 30
done

# If there's a remote config, try to update it
if [ -d ../gateway-remote-config ]; then
    # First pull from the repo
    pushd ../gateway-remote-config/
    git pull
    git reset --hard
    popd

    # And then try to refresh the gateway EUI and re-link local_conf.json

    # Same network interface name detection as on install.sh
    # Get first non-loopback network device that is currently connected
    GATEWAY_EUI_NIC=$(ip -oneline link show up 2>&1 | grep -v LOOPBACK | sed -E 's/^([0-9a-z]+): ([0-9a-z]+):.*$/1/' | head -1)
    if [[ -z $GATEWAY_EUI_NIC ]]; then
        echo "ERROR: No network interface found. Cannot set gateway ID."
        exit 1
    fi

    # Then get EUI based on the MAC address of that device
    GATEWAY_EUI=$(cat /sys/class/net/$GATEWAY_EUI_NIC/address | awk -F\: '{print $1$2$3"FFFE"$4$5$6}')
    GATEWAY_EUI=${GATEWAY_EUI^^} # toupper

    echo "[TTN Gateway]: Use Gateway EUI $GATEWAY_EUI based on $GATEWAY_EUI_NIC"
    INSTALL_DIR="/opt/ttn-gateway"
    LOCAL_CONFIG_FILE=$INSTALL_DIR/bin/local_conf.json

    if [ -e $LOCAL_CONFIG_FILE ]; then rm $LOCAL_CONFIG_FILE; fi;
    ln -s $INSTALL_DIR/gateway-remote-config/$GATEWAY_EUI.json $LOCAL_CONFIG_FILE
fi

# Fire up the forwarder.
./poly_pkt_fwd
```

[그림 2.6] /opt/ttn-gateway/bin/start.sh 스크립트

3) EUI.json 파일 생성하기

아래와 같이 LoRaWAN gateway를 위한 EUI.json 파일을 생성하도록 한다. Gateway ID, server address, LoRaWAN gateway가 위치한 위치 정보(위도/경도/고도), mail 주소 등을 적어 주면 된다.

/opt/ttn-gateway/gateway-remote-config/B827EBFFF2881FF.json

```
{
  "gateway_conf" : {
    "gateway_ID" : "B827EBFFF2881FF",
    "servers" : [
      {
        "server_address" : "router.kr.thethings.network",
        "serv_port_up" : 1700,
        "serv_port_down" : 1700,
        "serv_enabled" : true
      }
    ],
    "ref_latitude" : 37.5396825,
    "ref_longitude" : 127.002632,
    "ref_altitude" : 773,
    "contact_email" : "michael@2ipco.com",
    "description" : "SPN RoLa Gateway Test"
  }
}
```

[그림 2.7] gateway EUI.json 파일

참고 1 - <https://github.com/ttn-zh/gateway-remote-config> site에 위의 내용을 등록한 후, 자동으로 받아서 사용하는 방법도 있으나, 본 시험에서는 위와 같이 해당 파일을 직접 추가하는 형태로 시험해 보았다.

참고 2[나중에 정리한 것임] - 위의 내용과 3장의 Dragino LG01-N gateway 설정(LuCi 화면 설정) 내용을 비교해 보라. 비슷한 것을 알 수 있다.

4) global_conf.json 파일 수정하기

아래 site의 내용을 참조하여,

https://github.com/TheThingsNetwork/gateway-conf/blob/master/KR-global_conf.json

```
{
    "SX1301_conf": {
        "lorawan_public": true,
        "clksrc": 1,
        "clksrc_desc": "radio_1 provides clock to concentrator for most devices except MultiTech
. For MultiTech set to 0.",
        "antenna_gain": 0,
        "antenna_gain_desc": "antenna gain, in dBi",
        "lbt_cfg": [
            "enable": true,
            "rss_i_target": -80,
            "chan_cfg": [
                { "freq_hz": 922100000, "scan_time_us": 128 },
                { "freq_hz": 922300000, "scan_time_us": 128 },
                { "freq_hz": 922500000, "scan_time_us": 128 },
                { "freq_hz": 922700000, "scan_time_us": 128 },
                { "freq_hz": 922900000, "scan_time_us": 128 },
                { "freq_hz": 923100000, "scan_time_us": 128 },
                { "freq_hz": 923300000, "scan_time_us": 128 }
            ],
            "sx127x_rssi_offset": -4
        ],
        "radio_0": {
            "enable": true,
            "type": "SX1257",
            "freq": 922400000,
            "rss_i_offset": -166.0,
            "tx_enable": true,
            "tx_freq_min": 920900000,
            "tx_freq_max": 923300000
        },
        "radio_1": {
            "enable": true,
            "type": "SX1257",
            "freq": 923000000,
            "rss_i_offset": -166.0,
            "tx_enable": false
        },
        "chan_multiSF_0": {
            "desc": "Lora MAC, 125kHz, all SF, 922.1 MHz",
            "enable": true,
            "radio": 0,
            "if": -300000
        },
        "chan_multiSF_1": {
            "desc": "Lora MAC, 125kHz, all SF, 922.3 MHz",
            "enable": true,
            "radio": 0,
            "if": -100000
        },
        "chan_multiSF_2": {
    
```

[그림 2.8] global_conf.json 파일 수정 내용

지금까지 (아무 생각 없이 ... 하라는 대로)LoRaWAN Gateway 설정을 해 보았다. 이제부터는 지금까지 설정한 Gateway를 아래 그림과 같이 우측의 서버 및 Application과 연결시켜 보도록 하겠다.

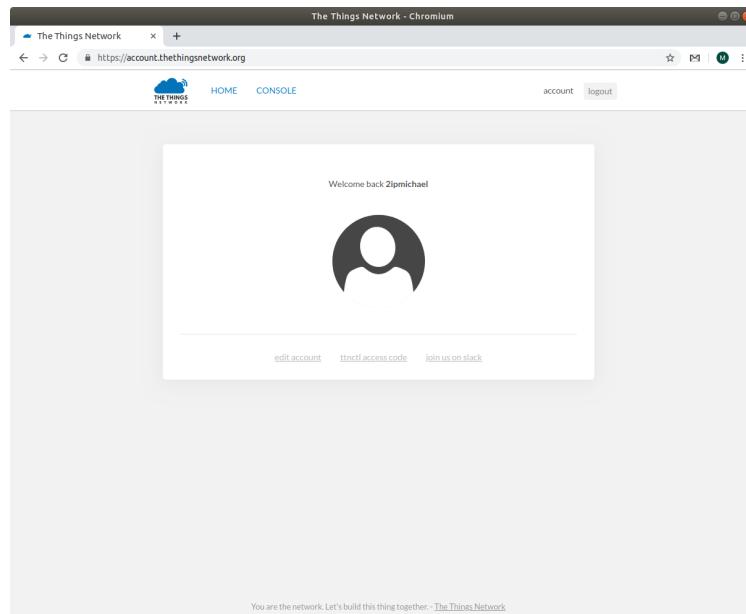


[그림 2.9] The Things Network(TTN) 구조

5) TTN(TheThingsNetwork) network에 gateway 등록하기

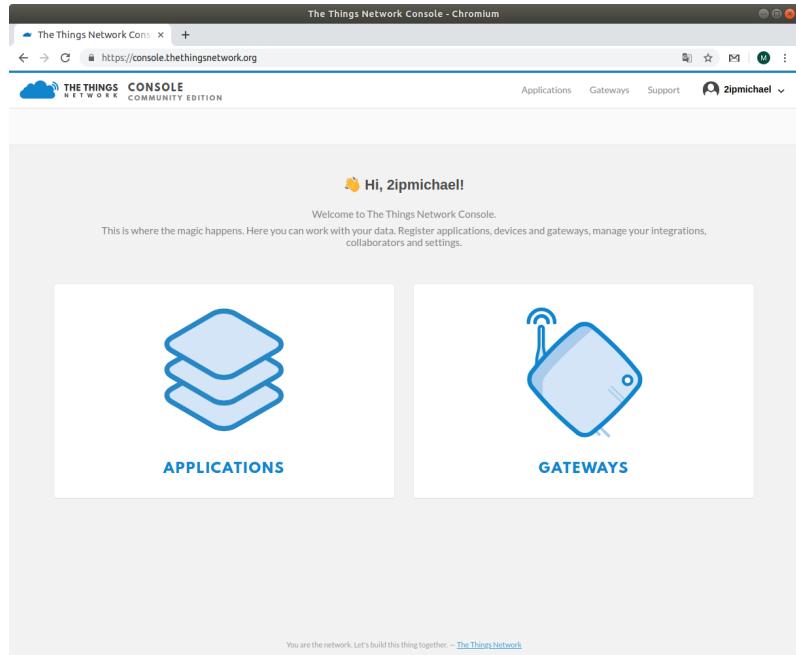
우선 아래 site에서 자신의 계정을 하나 생성하도록 하자. Id, password, email 주소 정도를 입력하면 간단히 끝난다.

<https://account.thethingsnetwork.org/>



[그림 2.10] TTN 계정 생성 모습

다음으로 상단의 CONSOLE 버튼을 선택하여 Gateway or Application을 생성해 보도록 하자.



[그림 2.11] TTN network console

The screenshot shows the "REGISTER GATEWAY" form in the The Things Network Console. It includes fields for "Gateway EUI" (B8 27 EB FF FE 28 81 FF), "Description" (Zipco test lora gateway1), "Frequency Plan" (Korea 920-923MHz), and "Router" (ttn-router-asia-se).

[그림 2.12] LoRaWAN Gateway 등록 모습

The screenshot shows the 'Gateway Overview' page for a gateway with EUI eui-b827ebffff2881ff. The details include:

- Gateway ID:** eui-b827ebffff2881ff
- Description:** 2ipco test lora gateway1
- Owner:** zipmichael (with a 'Transfer ownership' link)
- Status:** connected
- Frequency Plan:** Korea 920-923MHz
- Router:** ttn-router-asia-se
- Gateway Key:** A long string of dots representing the key, with a 'base64' link.
- Last Seen:** 13 seconds ago
- Received Messages:** 20
- Transmitted Messages:** 0

[그림 2.13] LoRaWAN Gateway 등록 후 상태 보기

The screenshot shows the 'Gateway Traffic' page for the same gateway. It displays a table of recent traffic events:

time	frequency	mod.	CR	data rate	airtime (ms)	cnt	dev addr	payload size
17:15:38	922.5	lora	4/5	SF 12 BW 125	2302	34	1B 08 05 65	46 bytes
17:14:19	922.3	lora	4/5	SF 12 BW 125	1810.4	165	1B 05 11 EE	35 bytes
17:13:26	922.3	lora	4/5	SF 12 BW 125	1810.4	1228	1B 0F 12 DD	35 bytes
17:11:10	922.5	lora	4/5	SF 12 BW 125	1810.4	164	1B 05 11 EE	35 bytes
17:11:04	923.3	lora	4/5	SF 11 BW 125	1970.2	3681	1A 00 27 C2	86 bytes
17:11:04	922.7	lora	4/5	SF 10 BW 125	452.6	1322	1B 01 19 60	33 bytes

[그림 2.14] LoRaWAN Gateway로 부터 올라오는 Traffic 살펴 보기

참고[나중에 정리한 것임] - 3장의 Dragino LG01-N Gateway를 TTN Server에 연결할 경우, 위와 같이 Gateway Traffic이 발생하는 것을 볼 수 없다. 즉, 정상적인 LoRa Gateway라면 LoRa

Node(device)로 부터 실제 LoRa packet이 올라올 경우에만 Server로 전달하게 된다. 한가지 더, 위의 LoRa Traffic이 내가 설치한 LoRa Node로 부터 전달된 packet이 아니라, 다른 사람이 설치한 LoRa Node로 부터 올라온 packet일 가능성이 크다. 그 이유는 LoRa Gateway는 주파수만 맞으면 무조건 해당 packet을 server로 옮겨 주기 때문이다.

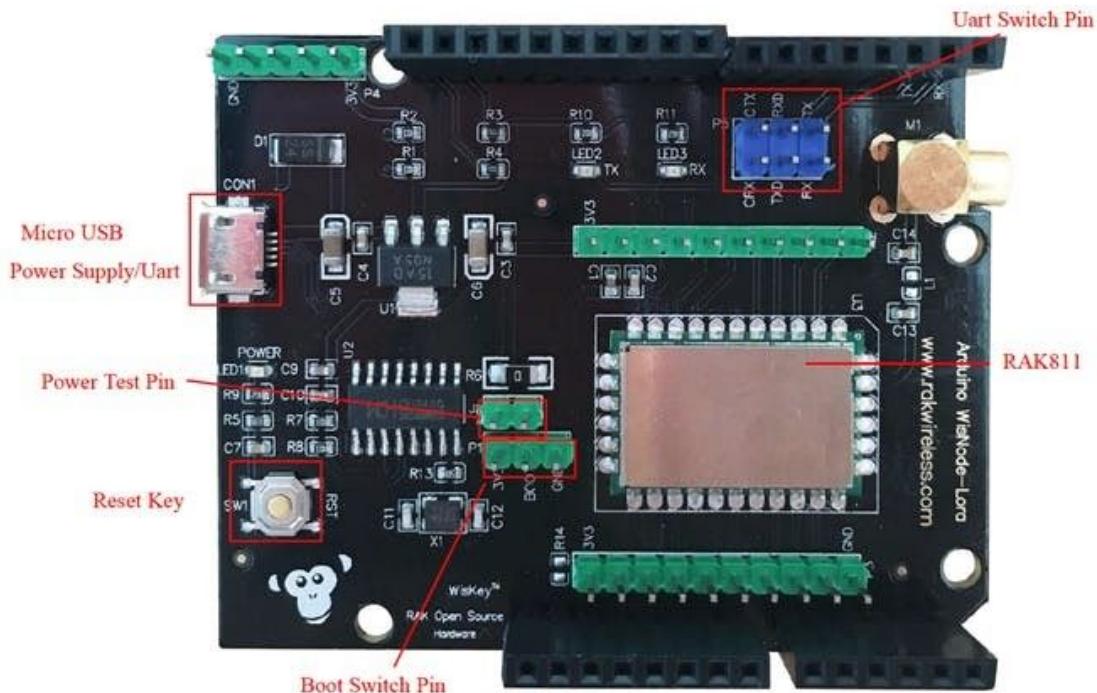
6) LoRaWAN Node 설치하기

지금까지 LoRaWAN gateway와 LoRa network server(TTN server)를 연결해 보았으니, 이제부터는 LoRa Node를 LoRaWAN gateway에 연결한 후, TTN Application(network server)과 정상 통신하는지를 확인해 보도록 하자.

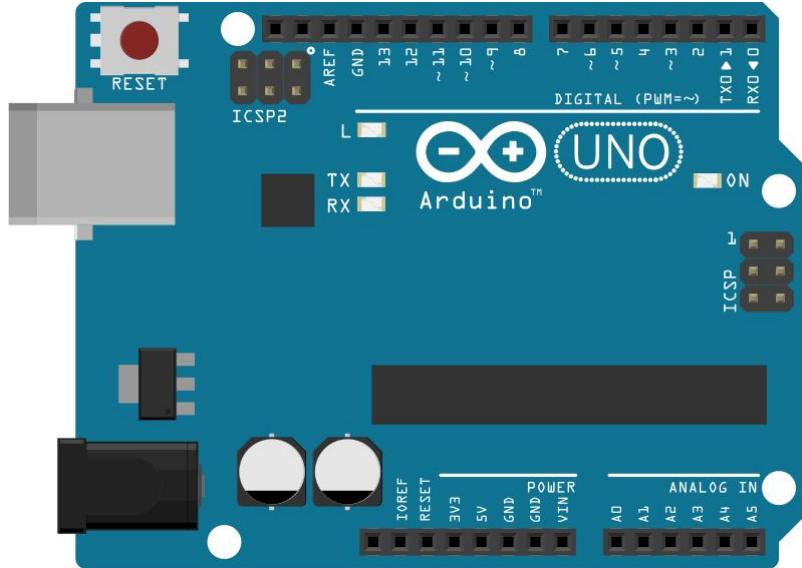
여기서 부터는 아래 site의 내용을 참조할 필요가 있겠다. 먼저 아두이노(UNO)를 Host로 하여 data를 LoRaWAN gateway로 전달하도록 설정해 주어야 한다.

<https://www.hackster.io/naresh-krish/using-the-rak811-lora-module-with-arduino-a38de8>

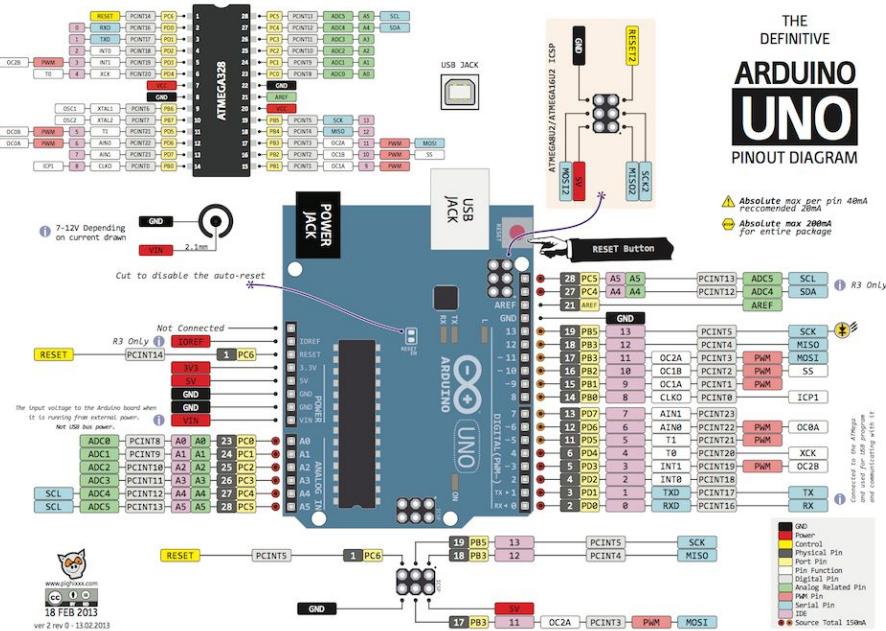
참고: 위의 site에서는 Arduino Mega와 RAK811을 붙이는 것을 설명하고 있으나, 본 문서에서는 Arduino Uno와 RAK811을 연결하는 것을 설명하였다. 사실상 커다란 차이는 없다.



[그림 2.15] RAK811 LoRa Shield

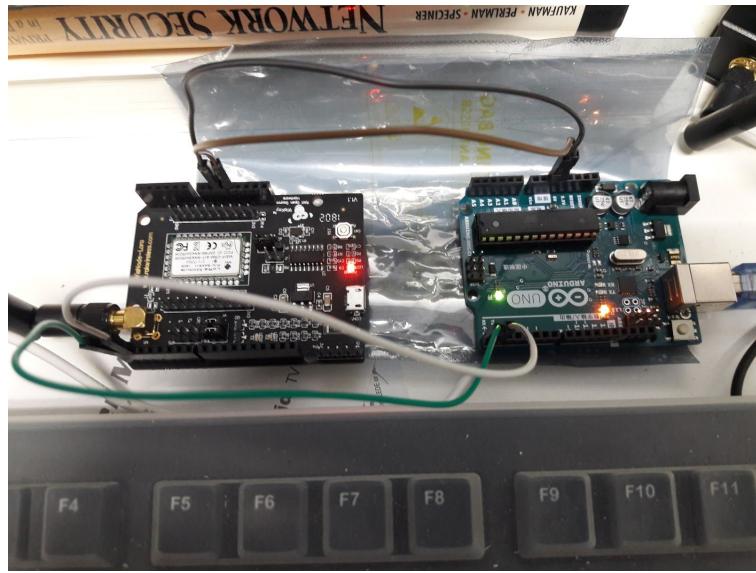


[그림 2.16] Arduino Uno 보드



[그림 2.17] Arduino Uno 보드 Pinmap

여기서 특별히 자세한 설명은 안하겠지만, Arduino를 잘 다루기 위해서는 아두이노에 대한 사전 이해가 필요하다. 자신에 맞는 적절한 서적을 선택하여 읽어 보기 권한다.



[그림 2.18] Arduino Uno와 RAK811 보드 연결하기

이제부터는 Arduino에 program을 하나 올리고, 이를 이용하여 RAK811을 제어하는 일(실제로 이게 일이군 ...)만 남았다. 예를 들어 아래와 같은 AT command를 RAK811로 보내는 arduino program을 하나 만들어 보기로 하자[TBD].

<OTAA mode 예>

```
Welcome to RAK811
at+mode=0 /* SET LoraWAN work mode */
OK
at+get_config=dev_eui /* GET Dev_EUI check */
OK3037343644357402
at+set_config=rx2:3,868500000
at+set_config=app_eui:39d7119f920f7952&app_key:a6b08140dae1d795ebfa5a6dee1f4d
bd
/* SET LoraGateway app_eui and app_key , big endian*/
OK
at+join=otaa /* Join OTAA type*/
OK
at+recv=3,0,0 /* Join status success*/ if you get 6,0,0 you need to recheck your
configurations
```

<ABP mode 예>

```
Welcome to RAK811
at+mode=0 /* SET LoraWAN work mode */
```

```

OK
at+set_config=rx2:3,868500000
at+set_config=dev_addr:00112233&nwks_key:3432567afde4525e7890cfea234a5821&app
s_key:a48adfc393a0de4
58319236537a11d90 /* SET LoraGateway dev_addr nwks_key and apps_key , big
Endian*/
OK
at+join=abp /* Join ABP type*/
OK
at+recv=3,0,0 /* Join status success*/

```

<RAK831에서 LoRa Gateway로 실제 data 내보내기>

```

/*After join gateway success, then can send and receive data*/

at+send=0,2,0000000000000007F0000000000000000 /*APP port:2, battery level 50%,
unconfirmed message*/

at+recv=2,0,0 /*unconfirmed mean tx success*/

at+send=1,2,0000000000000007F0000000000000000 /*APP port :2, battery level 50%,
confirmed message*/

at+recv=1,0,0 /*confirmed mean receive ack from gateway*/

```

<TBD> - Arduino programming

```

rak811 | 아두이노 1.8.9
rak811 $ //RAK811 Control

#include <SoftwareSerial.h>
SoftwareSerial Serial1(0, 1); // RX, TX [TBD]

void setup() {
    //configure Serial1, this could also be a
    //software serial.
    Serial1.begin(115200);
    //configure the mail RX0 and TX0 port on arduino
    Serial.begin(115200);
}

//string to hold the response of a command in rak811
String response = "";

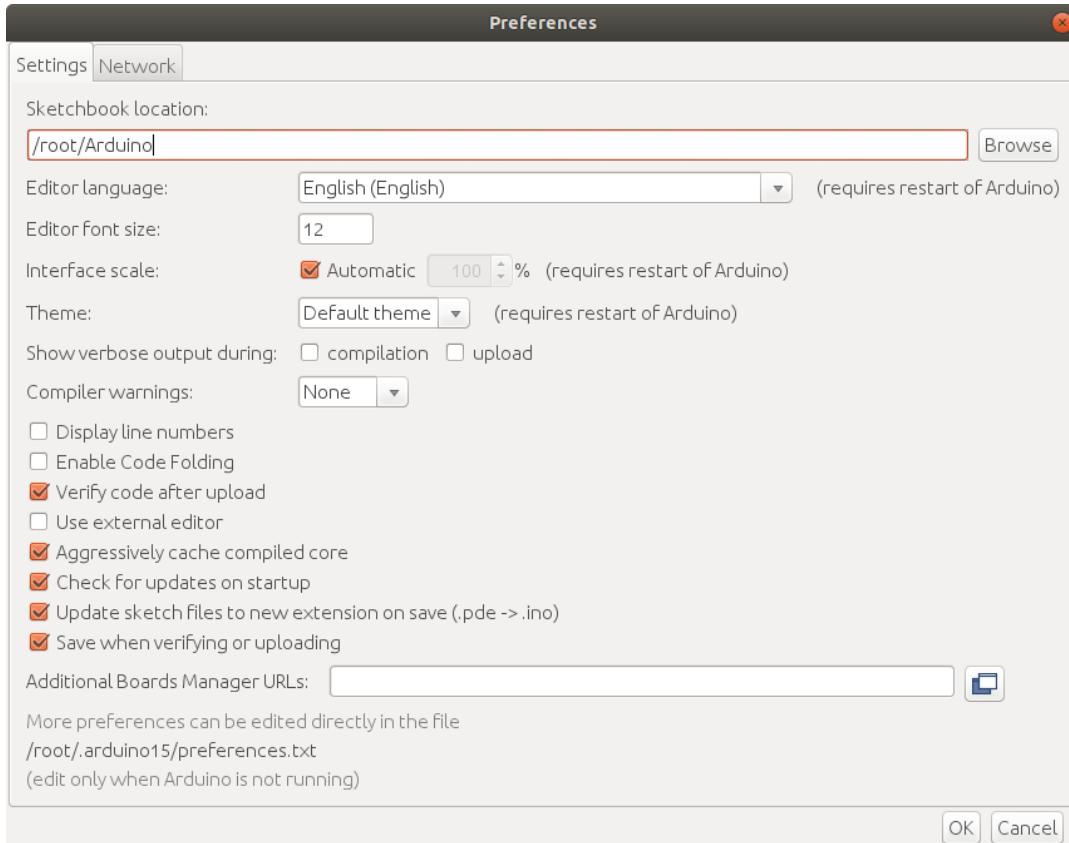
//the famous arduino loop function. runs continuosly
void loop() {
    //try getting the version of the board firmware
    sendCommand("at+version\r\n");
    //set conn config
    setConnConfig("dev_eui", "B827EBFFFFE2881FF");
    setConnConfig("app_eui", "your_app_eui_here");
    setConnConfig("app_key", "your_app_key_here");
    //join the connection
#if 0 /* ORIG_CODE - michael@2019.08.22 */
    sendJoinReq();
#else
    sendJoinReq("test");
#endif
    //send data to gateway
    ---DATA---(1, 0, "00000000000000000000000000000000")
}

23 Arduino/Genuino Uno

```

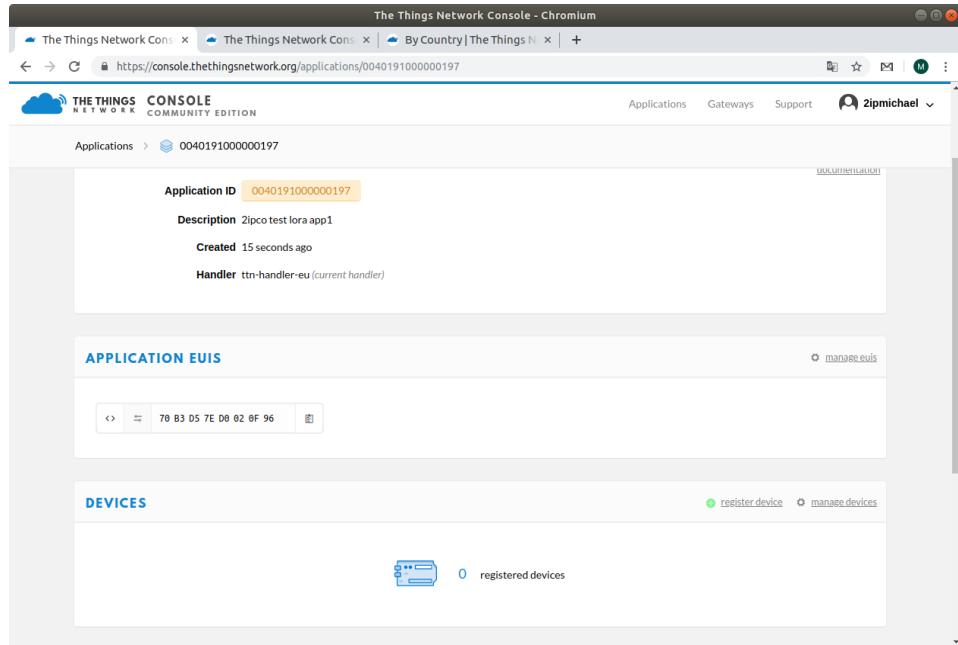
[그림 2.19] Arduino Program - RAK831 제어 코드

참고: 위와 같이 메뉴 글씨가 깨지는 것은 한글 출력과 관련이 있다. 따라서 File -> Preferences에서 아래와 같이 영문으로 변경하여 사용하기 바란다.



[그림 2.20] Arduino 영문 메뉴로 변경

7) TTN(TheThingsNetwork) network에 Application 등록하기[TBD]



[그림 2.21] Application 등록 모습

앞서 Arduino program이 정상적으로 동작한다면, Application 화면에 LoRa data가 정상적으로 출력될 것이다. :(:(:(실제로 해 보아야 한다. Dragino IoT Kit가 도착했으니, 이걸 먼저 사용해 보기로 하자.

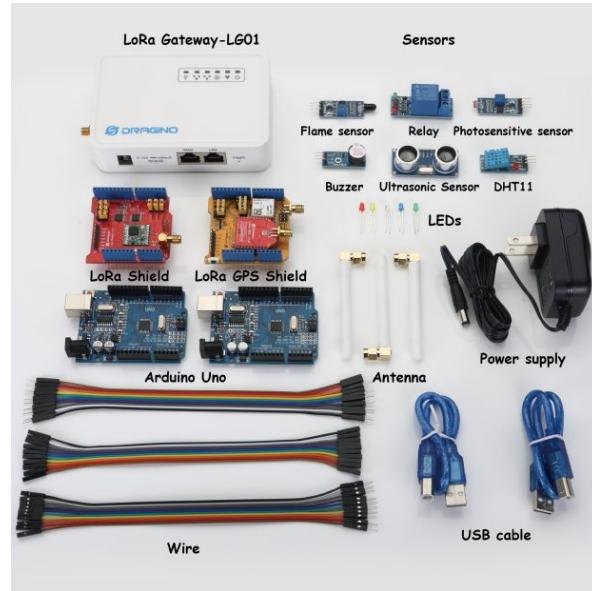
3. Dragino IoT Kit v2

이제 부터는 Dragino 사에서 제작하여 판매하는 IoT Kit v2를 이용하여 LoRa 망을 구축해 보도록 하겠다. Fu**ing Ali-Express를 통해 의도치 않게 3 set나 구매하게 되었는데, Kit 구성 내용물이 기대 이상으로 구성되어 있어 한결 마음이 가볍다^^.



[그림 3.1] Dragino LoRa IoT Kit v2(1)

센서가 어디갔다 보았더니, 박스 하단에 각종 센서용 별도의 상자가 있다.

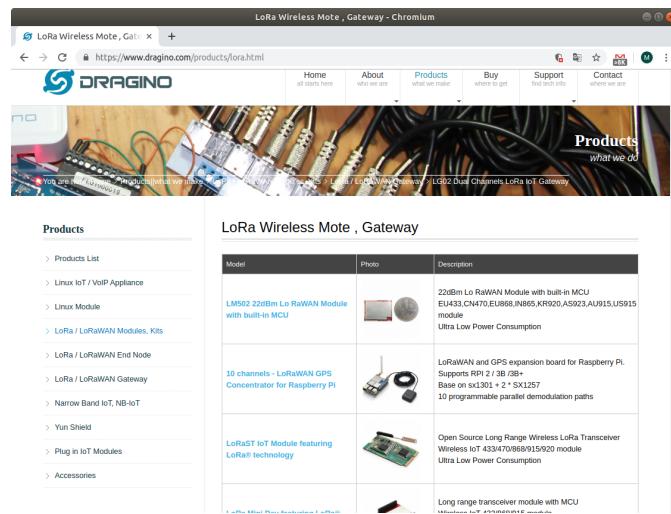


[그림 3.2] Dragino LoRa IoT Kit v2(2)

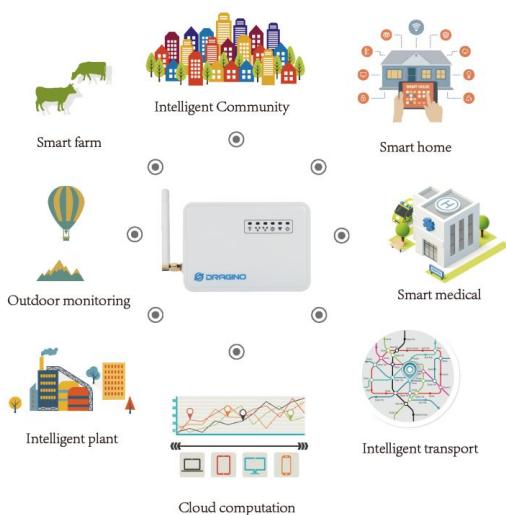
지금부터 소개하는 내용은 dragino site에서 공식적으로 제공하는 **Single Channel LoRa IoT Kit v2 User Manual_v1.0.5.pdf** 문서를 기초로하여 작성하였다. 따라서 본 문서를 읽기 전에 아래 site의 내용을 면밀히 훑어 보기 바란다.

https://www.dragino.com/downloads/index.php?dir=LoRa_IoT_Kit/v2-Kit/

참고: 아래 Home Page의 각 제품을 하나씩 선택하면, 관련 사용자 매뉴얼을 download 받을 수 있다.

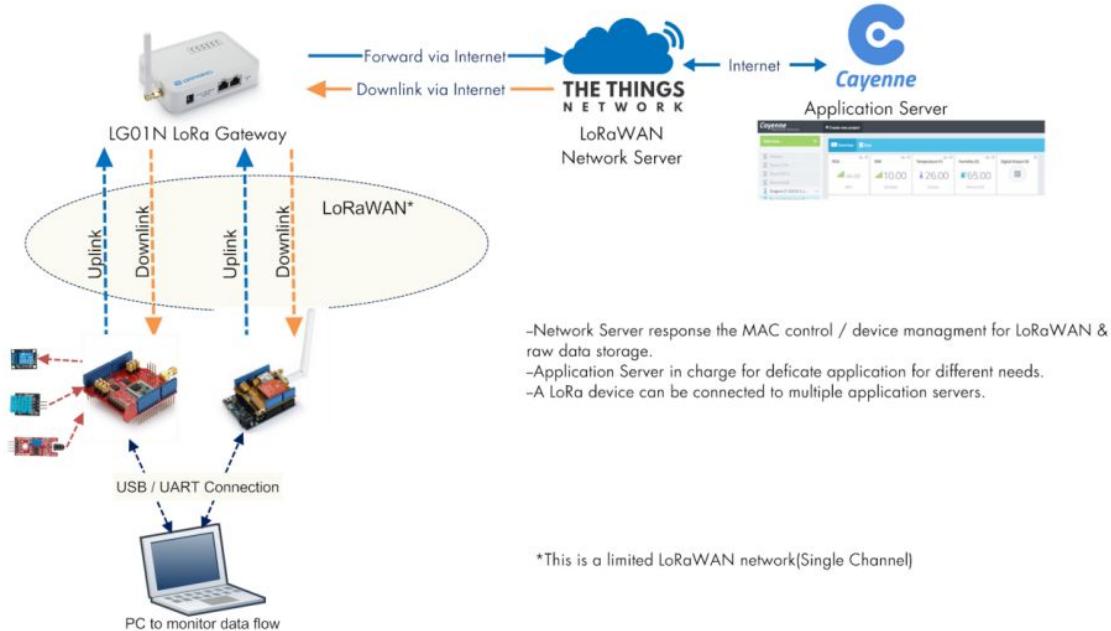


[그림 3.3] Dragino Home Page



[그림 3.4] LoRa target 시장

지금부터는 아래와 같은 구성(단, 우측 Cayenne Application Server 부분은 일단 제외)이 가능하도록 LoRa Gateway(LG01-N), LoRa Node, TTN(The Things Network) server를 차례로 설정하도록 할 것이다. 이 과정에는 LoRa node의 firmware를 수정(Korea 주파수 관련 작업을 위해)하는 작업도 포함되어 있다.



[그림 3.5] 지금부터 꾸밀 테스트 베드

앞으로 진행할 작업 절차를 간략히 요약해 보면 다음과 같다[중요].

1. LG01-N LoRa gateway 기본 설정

- LuCi Service menu로 진입
- 연결할 서버 주소(서버 포트 포함) 지정
- Gateway ID 지정
- LoRa radio 설정(주파수, Spreading Factor, Signal Bandwidth...) ... 요게 좀 어렵다.

2. TTN에 계정 등록 후, 위의 Gateway 등록

- <https://console.thethingsnetwork.org/>
- <https://console.thethingsnetwork.org/gateways/register>

3. LoRa Node 준비

- Jumper cable을 이용해 필요한 연결하기
- LoRa shield pinmap 관련하여 사전에 파악해 두어야 함.

4. TTN에 Application 등록 - LoRa Node(device)를 추가

- <https://console.thethingsnetwork.org/applications/add>

5. LoRa Node(Arduino + LoRa shield)에 firmware 올리기

- KR920(Korea 주파수 band) 관련 작업이 되어 있는 코드를 Arduino flash memory에 loading(programming).
- 주의1: dragino에서 porting한 LMIC를 arduino에 설치해 주어야 함.
- 주의2: 이후, 이 LMIC code를 KR920용으로 porting해 주어야 함.
- 주의3: APPEUI, DEVEUI, APPKEY 등을 위의 4번 단계에서 설정한 내용과 맞추어 주어야 함.

5. TTN Gateway & Application 화면을 통해 LoRa 패킷 수신 여부 확인

1) LG01-N Gateway 설정하기

Dragino LG01-N gateway의 외관은 아래와 같다. 외부에 2개의 ethernet port와 1개의 LoRa 안테나가 눈에 들어온다. 내부적으로는 802.11b/g/n용 wifi chip이 장착되어 있으며, LTE module도 장착(optional) 가능하다. LoRa & LTE 부분만 제외한다면 **GI.iNet의 USB150 수준(CPU, memory 등 일함)**의 제품이다. LoRa 자체가 아주 작은 data를 전송하는 용도인 점을 감안하면, 이 정도의 성능 spec은 어찌보면 당연한 얘기이기도 하겠다^^. **이거 LoRa 없이 SPNBox로 사용해도 되겠는걸~**



[그림 3.6] Dragino LG01-N Gateway

<주요 spec>

CPU: 400Mhz ar9331 processor

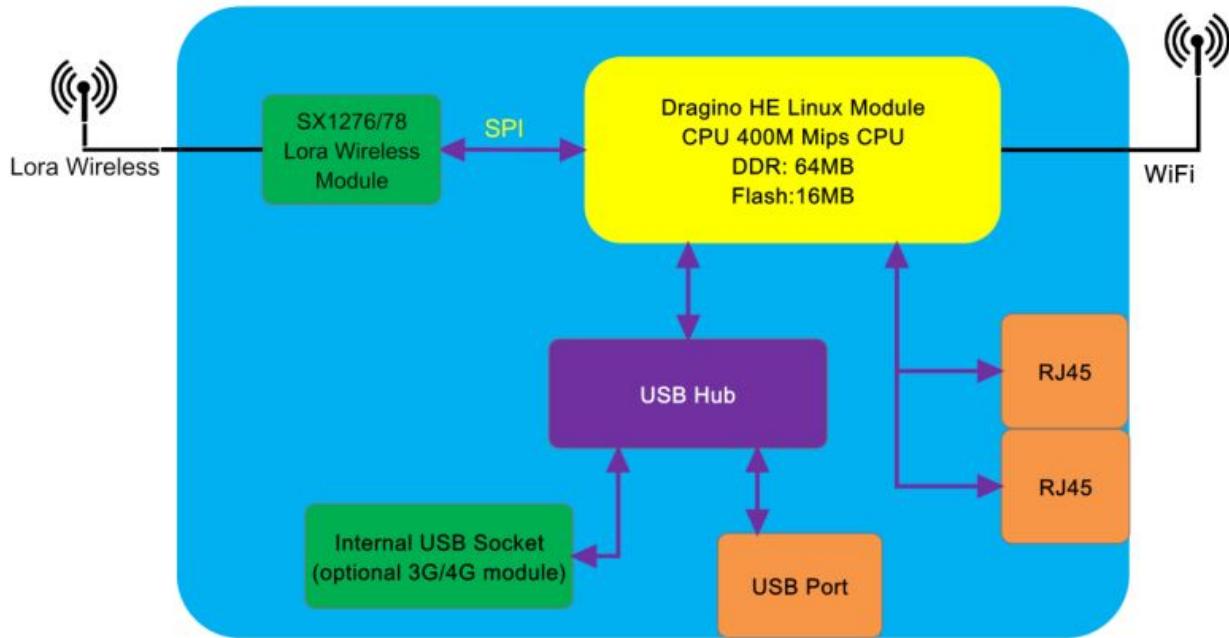
Memory: 64MB RAM, 16MB Flash

Interface:

- 10M/100M RJ45 Ports x 2
- WiFi : 802.11 b/g/n

- LoRa Wireless
- Power Input: 12V DC
- USB 2.0 host connector x 1
- USB 2.0 host internal interface x 1
- 1 x LoRa Interfaces
- Quectel EC25 LTE module(선택 사항)
- Micro SIM Slot(선택 사항)

아래 그림은 LG01-N 보드 내부를 좀 더 자세하게 소개한 block도이다. 좌측 상단의 SX1276/78이 LoRa wireless module에 해당하며, CPU(ar9331)하고는 SPI 인터페이스를 통해 통신하는 것을 알 수 있다. 참고로, 아래 그림에는 LTE module 자체는 빠져 있다.



[그림 3.7] Dragino LG01-N 시스템 아키텍쳐

속제 : Dragino LG01-N source code 내용 중, 위의 LoRa module과 통신(SPI 통신)하는 부분을 찾아 보라.

Dragino LG01-N은 OpenWrt를 기반으로 동작하는데, 관련 source는 아래 위치에서 확인 가능하다.

https://github.com/dragino/openwrt_lede-18.06

Dragino LG01-N용 openwrt를 build 절차는 매우 간단하다. 중간에 한 차례 에러가 나긴 했으나, 두번 정도 시도해 보니, 결과물이 정상적으로 만들어 진다.

<OpenWrt build 절차>

```
$ git clone https://github.com/dragino/openwrt_lede-18.06 dragino-lede-18.06
$ cd dragino-lede-18.06
$ ./set_up_build_environment.sh
$ ./build_image.sh

$ cd image/
$ cd LG02_LG08--build-v5.2.1565754694-20190814-1313/
$ ls -la
합계 16052
drwxr-xr-x 2 chyi chyi 4096 8월 14 13:13 .
drwxr-xr-x 3 chyi chyi 4096 8월 14 13:13 ..
-rw-r--r-- 1 chyi chyi 33166 8월 14 13:13 custom_config.tar.gz
-rw-r--r-- 1 chyi chyi 1441792 8월 14 13:13
dragino-LG02_LG08--v5.2.1565754694-kernel.bin
-rw-r--r-- 1 chyi chyi 6815744 8월 14 13:13
dragino-LG02_LG08--v5.2.1565754694-rootfs-squashfs.bin
-rw-r--r-- 1 chyi chyi 8126468 8월 14 13:13
dragino-LG02_LG08--v5.2.1565754694-squashfs-sysupgrade.bin
-rw-r--r-- 1 chyi chyi 472 8월 14 13:13 sha256sums
```

Openwrt build 결과물을 LG01-N gateway에 적용해 보는 것은 추후에 진행해 보기로 하고,
이제부터는 본격적으로 LoRa Gateway 설정을 진행해 보도록 하자.

참고: 아래와 같이 dragino openwrt에는 wireguard package 자체가 지원되지 않고 있으니,
 나중에 직접 WireGuard kernel을 integration 시켜 보아야 겠다.

```
# opkg update
# opkg install wireguard

Installing wireguard (0.0.20190702-1) to root...

Downloading
http://downloads.openwrt.org/snapshots/packages/mips_24kc/base/wireguard_0.0.2019
0702-1_mips_24kc.ipk

Collected errors:

* satisfy_dependencies_for: Cannot satisfy the following dependencies for wireguard:
  *      kernel (= 4.14.125-1-4798db7ce04dcb4337e44dbaa2a958a0) *      kernel (= 4.14.125-1-4798db7ce04dcb4337e44dbaa2a958a0) *      kernel (= 4.14.125-1-4798db7ce04dcb4337e44dbaa2a958a0) *
```

```
* opkg_install_cmd: Cannot install package wireguard.
```

예상했겠지만, LG01-N에는 LuCi 기반의 WebUI가 존재한다. 하지만, GL.iNet과 같이 화려한 UI는 존재하지 않는다.

LG01-N을 아래와 같이 연결한 후, 접속을 시도해 보도록 하자(이후 내용은 다 아는 내용이라 더 이상의 내용은 생략하겠다).



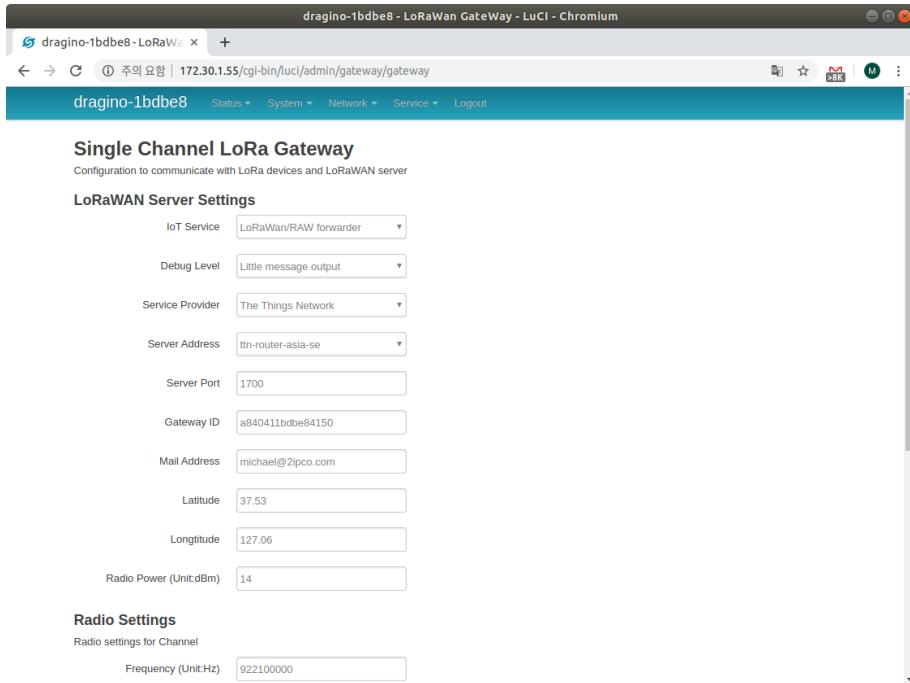
[그림 3.8] Dragino LG01-N 연결 모습(주변 연결이 내 마음처럼 복잡하다^^)

<Dragino LG01-N WebUI 접속 방법>

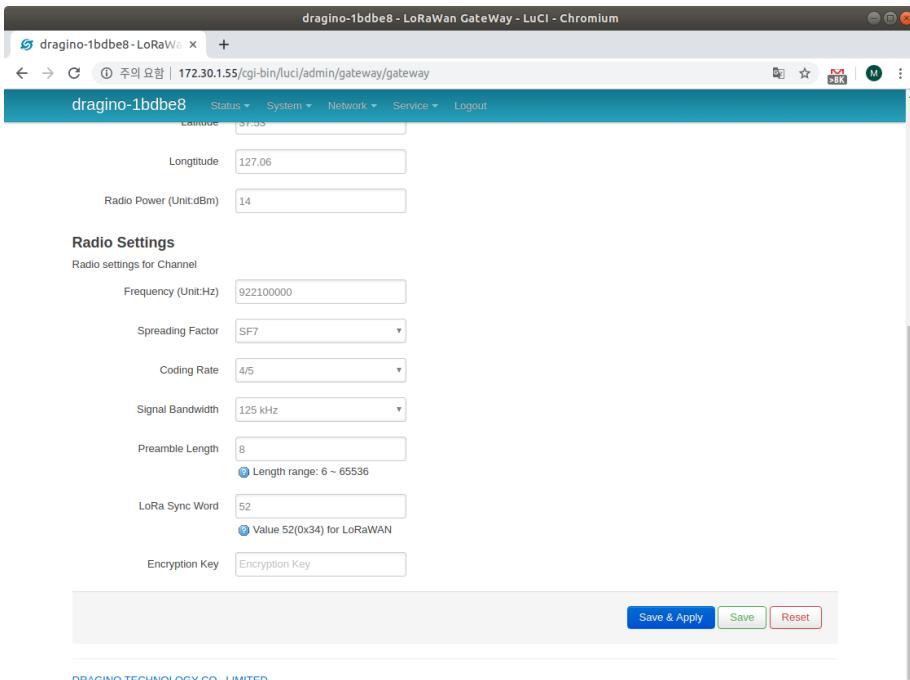
```
http://10.130.1.1
```

```
Id: root, passwd: dragino
```

각설하고, LoRa 설정을 위해서는 WebUI Service menu에서 아래와 같이 입력하도록 하자.



[그림 3.9] LG01-N LoRa Gateway 설정(1)



[그림 3.10] LG01-N LoRa Gateway 설정(2)

<반드시 입력해야 하는 사항>

<LoRaWAN Server Settings>

1. IoT Service : **LoRaWAN/RAW forwarder** 선택
2. Service Provider: **The Things Network**
3. Server Address : **ttn-router-asia-se** (이건 다른 것으로 지정해도 됨)
4. Server Port: **1700**
5. Gateway ID : **TTN에서 LoRa Gateway를 식별하는 인자**(Unique한 값이어야 함. 미리 지정되어 있는 값 그대로 사용하면 됨)
6. Your Mail address
7. LoRa gateway가 위치한 위도(Latitude), 경도(Longitude) 값 입력 - TTN server에서 지도록 위치 보여주는데 사용함.
8. Radio Power(dbm) : 14 입력(이건 확실치 않음, 일단 동작에 큰 영향을 주지 않는 듯)

<Radio Settings>

9. Frequency(Hz): **922100000** ← KR920 채널 중 하나 입력
10. Spreading Factor: **SF7**
11. Coding Rate : **4 / 5**
12. Signal Bandwidth: **125kHz**
13. Preamble Length: **8**
14. LoRa Sync Word: **52**
15. Encryption Key: 입력 안함.

참고: Radio 설정 중, 명확치 않은 부분도 있으나, 대략 위와 같이 설정해 주니 정상 동작한다.

<여기서 잠깐 !>

이미 1장에서도 한번 언급했지만, LoRa용 Korea 주파수와 관련해서는 아래 site를 참고하기 바란다.

<https://www.thethingsnetwork.org/docs/lorawan/frequency-plans.html>

KR920-923

Uplink:

922.1 - SF7BW125 to SF12BW125

922.3 - SF7BW125 to SF12BW125

922.5 - SF7BW125 to SF12BW125

922.7 - SF7BW125 to SF12BW125

922.9 - SF7BW125 to SF12BW125

923.1 - SF7BW125 to SF12BW125

923.3 - SF7BW125 to SF12BW125

none

Downlink:

Uplink channels 1-7

921.9 - SF12BW125 (RX2 downlink only; SF12BW125 might be changed to SF9BW125)

2) TTN에 LoRa Gateway 등록하기

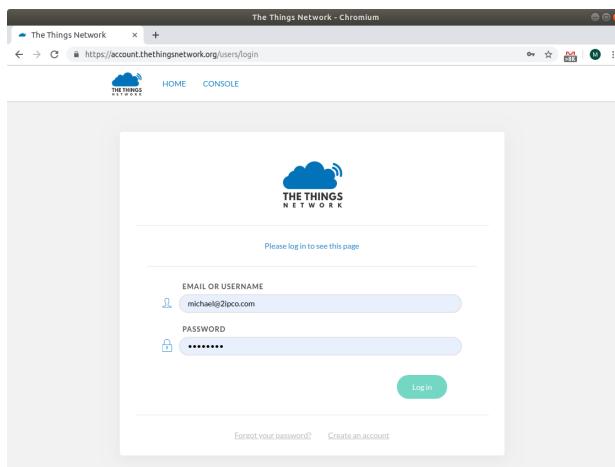
LoRa Gateway 설정이 끝났으니, 이제 부터는 TTN(The Things Network) 서버에 LG01-N gateway를 등록할 차례이다. TTN이 무엇을 하는 site인지는 아래 page를 참조해 보기 바란다.

<https://www.thethingsnetwork.org/>

TTN이 무엇을 하는 site인지 알아 보았으니, 이제는 TTN console을 통해 나만의 gateway를 등록해 보기로 하자.

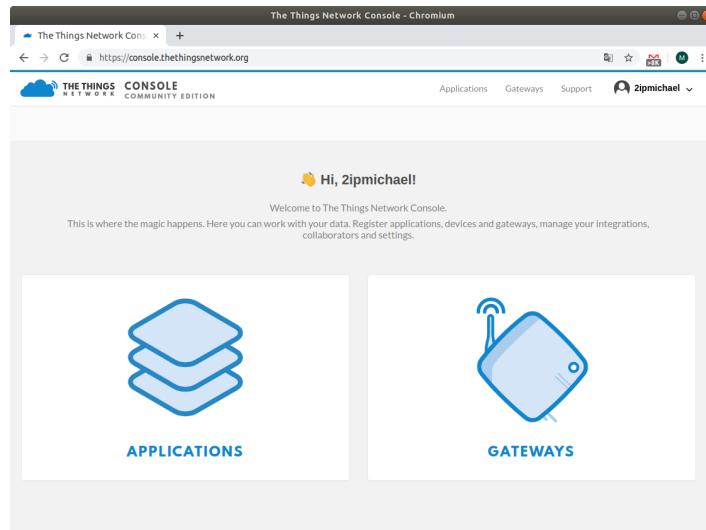
<https://console.thethingsnetwork.org/>

TTN gateway or Application을 등록하려면 먼저 계정이 하나 있어야 한다(이미 2장에서 설명한 내용임). 계정 등록 절차는 매우 간단하므로 별도로 설명하지 않기로 한다.



[그림 3.11] TTN 계정 등록 후 로그인하기

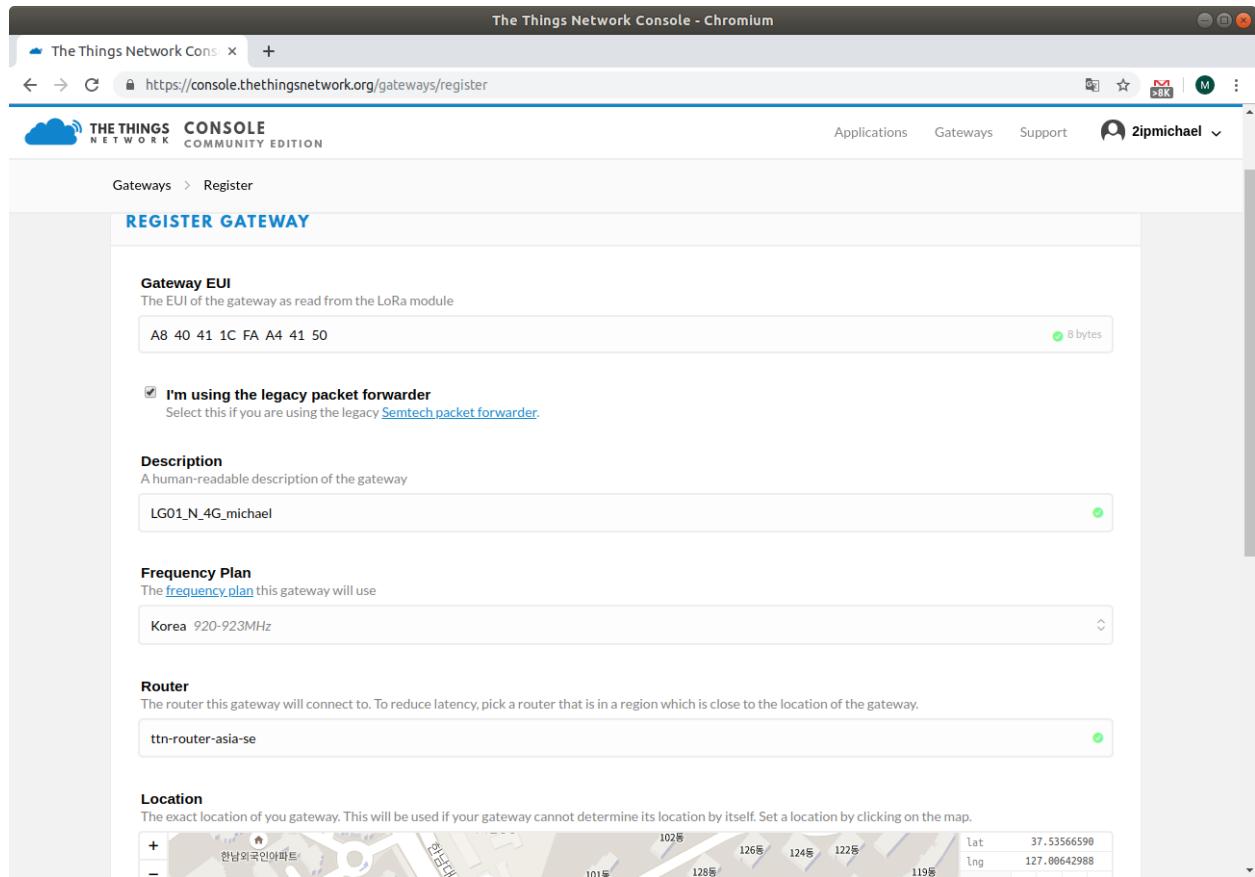
이후, 자신의 계정으로 로긴하면, 아래와 같이 Application과 Gateway를 선택하는 화면을 만나게 된다.



[그림 3.12] TTN Console Applications & Gateways 등록 화면

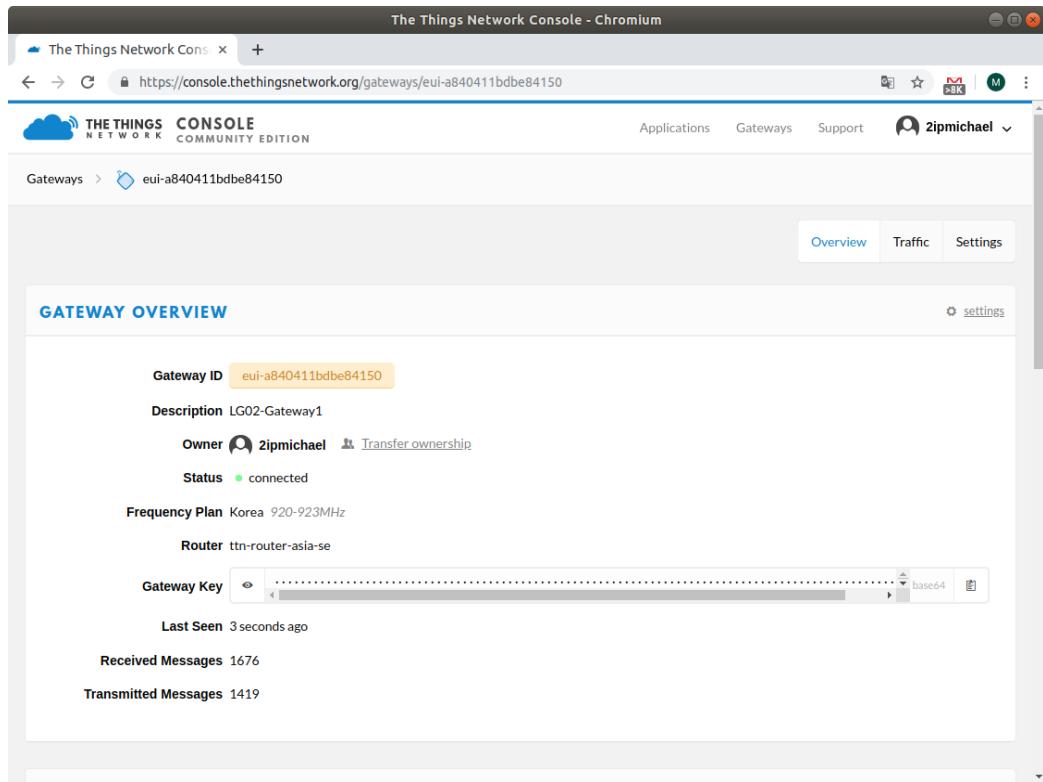
먼저 우측의 GATEWAYS 버튼을 선택하여, 자신의 gateway를 하나 등록하도록 하자. 아래 그림의 입력에 필요한 gateway 정보는 앞선 **LG01-N Gateway 설정하기** 절에서 모두 언급한 내용이니 특별히 설정에 어려움은 없을 것으로 보인다.

주의: Gateway EUI는 gateway ID 값을 말하며(단 hexa 대문자로 입력), **I'm using the legacy packet forwarder**를 체크해 주어야 한다.



[그림 3.13] TTN Console - 신규 Gateway 추가 화면

정상적으로 등록이 되고, 만일 Gateway가 정상적으로 동작하고 있다면 아래와 같이 Status 값이 **Connected**로 출력되는 것을 확인할 수 있을 것이다. Not Connected로 출력된다면 Gateway 설정을 다시한번 확인해 보기 바란다.



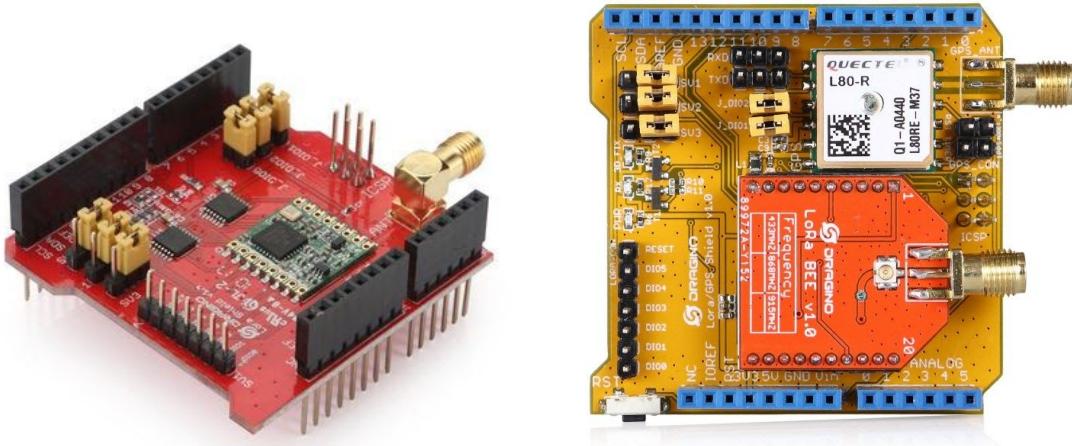
[그림 3.14] TTN Console - Gateway Overview 화면

이후의 내용(Gateway Traffic)은 LoRa Node를 연결한 후, 다시 살펴 보기로 하자. **LoRa Node가 준비되지 않으면, Gateway로는 packet이 전달되지 않는다.**

주의: 자신의 LoRa Node가 준비되어 있지 않았음에도 불구하고, Gateway Traffic 확인 시 packet이 보이는 경우가 종종 있는데, 이는 해당 주파수를 사용하는 다른 LoRa Gateway(예: SK LoRa 망)로 부터 packet이 도달했기 때문이다. 따라서 해당 packet의 내용을 꼭 확인해 보시길~

3) LoRa Node 준비하기

LoRa Gateway가 어느 정도 준비되었으니, 이제 부터는 LoRa Node를 준비할 차례이다. Dragino LoRa Kit v2에는 2개의 LoRa Node가 포함되어 있다. 하나는 일반적인 sensor(예: 온도/습도, 적외선, 초음파/거리, flame 센서 등)를 연결할 수 있는 shield를 장착한 node이고, 다른 하나는 GPS 용 수신용 shield를 장착한 node이다.



[그림 3.15] Dragino LoRa Shield(좌측) and LoRa GPS shield(우측)

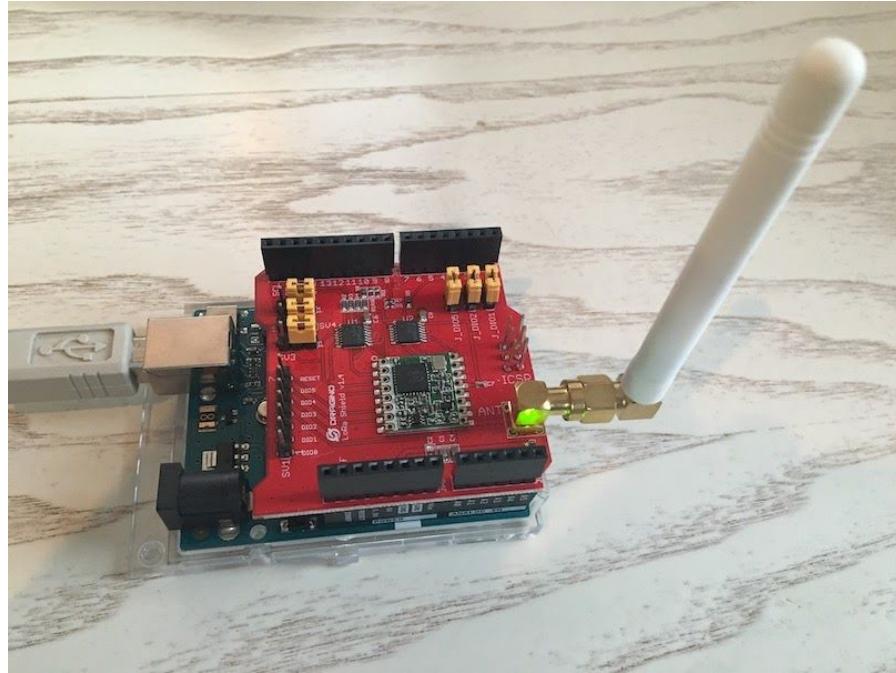
Dragino LoRa shield 관련 자세한 hardware spec은 아래 site를 반드시 참조하기 바란다.

https://wiki.dragino.com/index.php?title=Lora_Shield

아래 blog(my blog)는 적외선, 초음파/거리, 조도 센서를 사용하는 예를 소개한 site이니, 함께 참고하기 바란다.

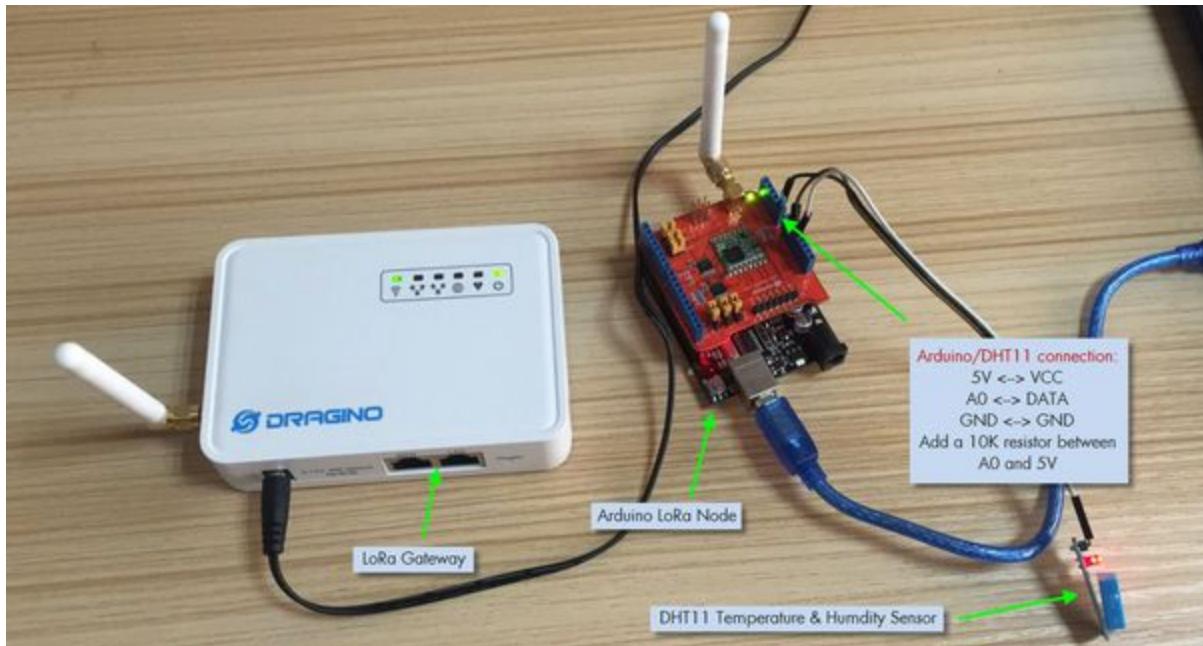
<http://slowbootkernelhacks.blogspot.com/2017/04/linux-device-driver-programming-using.html>

LoRa Shield는 자체만 가지고 동작하기 어렵고, 필요한 programming을 위해 아두이노(8bit computer) 같은 host computer가 동반되어야 한다. 아래 그림은 LoRa shield와 Arduino UNO를 연결한 모습이다(안테나도 달았다).



[그림 3.16] Dragino LoRa Shield + Arduino UNO(host computer)

아래 그림은 Dragino LoRa Kit에 포함되어 있는 DHT11 온도/습도 센서를 LoRa Node에 장착한 후, LG01-N Gateway와 함께 찍은 사진이 되겠다. 실제로 DHT11 센서를 연결하는 절차는 여기서는 별도로 설명하지는 않는다.



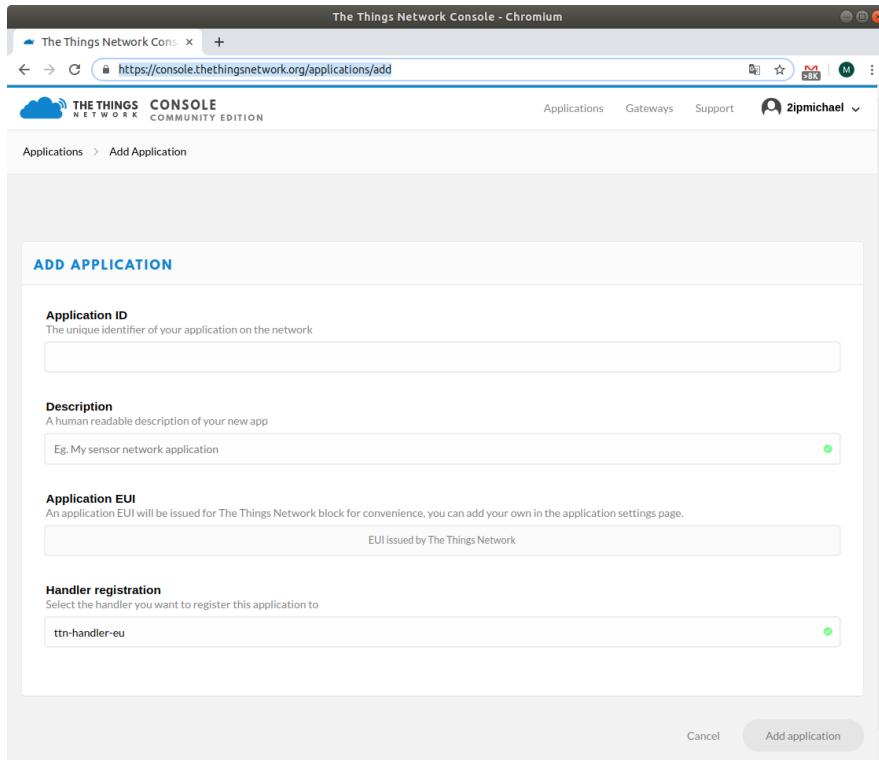
[그림 3.17] LG01-N Gateway와 Dragino LoRa Node1 연결 모습

주의: 이 절에서는 LoRa shield와 각종 센서 연결 부분을 아주 간단하게 설명하였다. 하지만 초보 개발자에게는 이 부분이 다소 어렵게 느껴질 수도 있다. 따라서 반드시 사전에 관련 서적이나 인터넷을 살펴 볼 것을 권한다.

4) TTN 서버에 LoRa Node용 Application 등록하기

LoRa Node에 대한 물리적인 연결(준비) 작업이 끝났으니, 다음으로는 TTN에 Application을 하나 등록하고, 다시 그 아래에 device(LoRa Node)를 등록하는 절차를 소개하도록 하겠다.

먼저 TTN Application은 아래 화면을 통해 등록할 수 있다. 역시 등록 당시의 내용을 사전에 capture해 둔 것이 없으니, 각자 하나씩 설정해보기 바란다. **참고로 아래 내용 중 Application EUI 값은 TTN server에서 자동으로 생성해 주는 값이다.**



[그림 3.18] TTN Application 등록 화면

TTN Application이 하나 추가되면, 그 결과로 아래와 같은 내용(Overview 화면에서 확인)이 출력되게 된다.

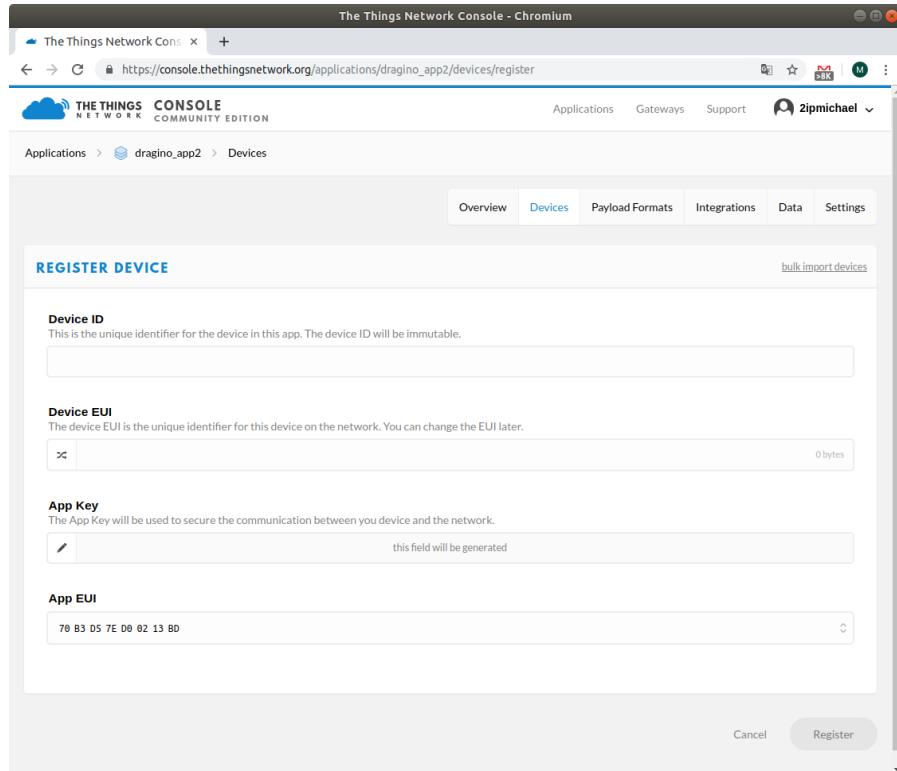
The screenshot shows the The Things Network Console interface. At the top, there's a header with the title 'The Things Network Console - Chromium' and a URL 'https://console.thethingsnetwork.org/applications/dragino_app2'. Below the header, there's a navigation bar with links for 'Applications', 'Gateways', 'Support', and a user profile 'zipmichael'. The main content area is divided into three main sections:

- APPLICATION OVERVIEW:** Shows details about the application, including 'Application ID: dragino_app2', 'Description: dragino app test', 'Created: 2 days ago', and 'Handler: ttn-handler-asia-se'. There's also a link to 'documentation'.
- APPLICATION EUIS:** Displays a hex string '70 B3 D5 7E D0 02 13 BD' with copy and paste options.
- DEVICES:** Shows a single registered device icon and the text '1 registered device'.

At the bottom of the main content area, there are tabs for 'Overview', 'Devices', 'Payload Formats', 'Integrations', 'Data', and 'Settings'.

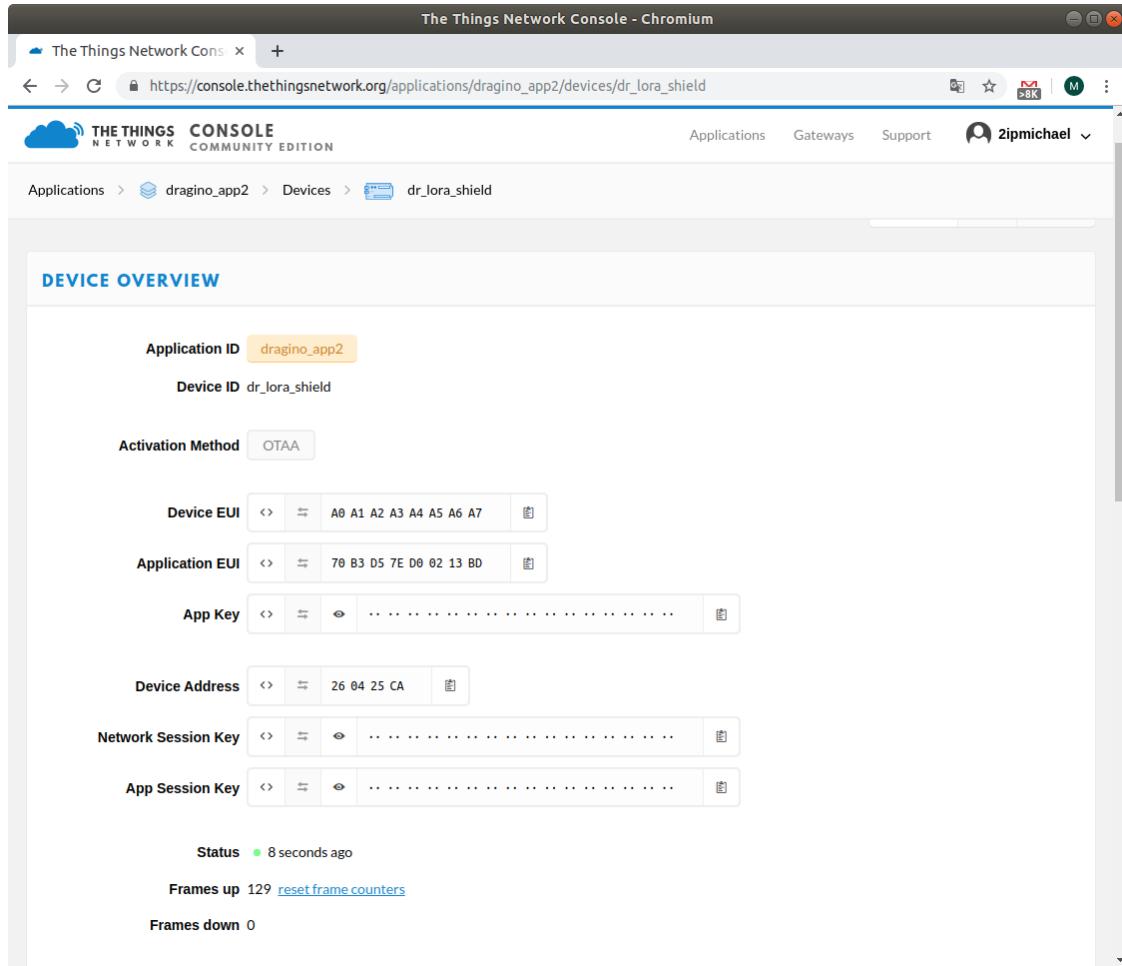
[그림 3.19] TTN Application Overview 화면

다음으로, 앞 화면 우측의 **register device** link를 눌러 device 즉, LoRa Node를 하나 추가해 보도록 하자. 이 page에서 사용자가 입력하거나 TTN이 자동으로 생성해 주는 정보 즉, **Device EUI**, **App Key**, **App EUI**는 LoRa Node programming시 사용되는 정보이므로 꼭 기억해 두기 바란다.



[그림 3.20] TTN Application에 device 추가 화면

마지막으로 아래 그림은 지금 단계에서 확인할 수 있는 부분은 아니지만, 최종적으로 LoRa Node가 TTN Server와 정상 연결되었을 경우를 보여준다(미리 보여주는 것임~).



[그림 3.21] TTN device Overview 화면

5) LoRa Node Programming하기

여기까지 준비되었으면, 이제 부터는 LoRa Node1(Arduino UNO + LoRa Shield)에 firmware를 올려 볼 차례이다. 이를 위해서는 Arduino IDE(개발환경)와 테스트용 program이 필요하다.

Arduino IDE는 아래 site에서 내려 받을 수 있다(설치 및 자세한 사용법에 관한 부분은 생략).

<https://www.arduino.cc/en/main/software>



[그림 3.22] Arduino IDE 실행 모습

Dragino LoRa Node를 구동하기 위해서는 몇가지 arduino library(C/C++ codes)가 필요한데, 이를 정리해 보면 다음과 같다.

1) **Arduino-LMIC**: LoRaWAN library(LoRa node와 LoRa gateway 통신용), dragino 보드에 맞게 일부 내용 수정(porting)한 version임. **Dragino 장비로 LoRaWAN 통신을 하고자 할 경우 반드시 필요한 library(단, KR920 주파수 환경을 위해서는 추가로 수정 작업을 해야 함)**

<https://github.com/dragino/arduino-lmic>

2) **LoRa-raw**: private LoRa protocol을 사용하여 LoRa 망을 구축하고자 하는 경우 사용하는 library(ID control, encryption 기능 없음)

<https://github.com/sandeepmistry/arduino-LoRa>

3) **DHTlib**: DHT11 temperature & humidity sensor를 위한 library

<https://github.com/goodcheney/Lora/blob/patch-1/Lora%20Shield/Examples/DHTlib.zip>

4) **TinyGPS**: Library for LoRa GPS Shield(GPS data 수신 관련)

<http://arduiniana.org/libraries/tinygps/>

이상의 library를 Arduino 환경에 포함시키는 절차는 **Single Channel LoRa IoT Kit v2 User Manual_v1.0.5.pdf** 문서를 참조하도록 하자(실은 별거 없다. Arduino libraries folder에 source만 가져다 놓으면 끝...).

자, 그럼 이제부터는 LoRaWAN 통신용 sample program을 하나 돌려 보기로 하겠다. Dragino 문서에서는 아래 예제를 소개하고 있으나, 그 보다 더 간단한 예제(LoRaWAN을 타고 "hello. world !" string을 보내는 예제)를 먼저 소개하는 것을 맞을 듯 싶다. 그 이유는 여러가지 센서를 특별히 shield에 연결하지 않고 덜 수 있기 때문이다.

<DHT11/flame/relay sensor용 예제>

https://github.com/dragino/Arduino-Profile-Examples/tree/master/libraries/Dragino/examples/IoTServer/Cayenne%20and%20TTN/cayenne%20and%20ttn%20example/lora_shield%20DHT11%20and%20Relay%20examples/lora_shield_cayenne_and_ttn-otaaClient

<hello world 예제>

arduino-1.8.9/libraries/arduino-lmic/examples/ttn-otaa/**ttn-otaa.ino**

Arduino IDE 상태에서 코드를 open한 후, compile & 실행 버튼을 눌러 보자. 아래 그림과 같이 출력되면 정상으로 loading된 것이다. 당연한 거지만, 에러가 발생할 수도 있는데, 어려운 것이 아니니 각자 해결해 보기 바란다.

```

ttn-otaa | Arduino 1.8.9
File Edit Sketch Tools Help
ttn-otaa §
/*
* This example sends a valid LoRaWAN packet with payload "Hello,
* world!", using frequency and encryption settings matching those of
* the The Things Network.
*
* This uses OTAA (Over-the-air activation), where a DevEUI and
* application key is configured, which are used in an over-the-air
* activation procedure where a DevAddr and session keys are
* assigned/generated for use with all further communication.
*
* Note: LoRaWAN per sub-band duty-cycle limitation is enforced (1% in
* g1, 0.1% in g2), but not the TTN fair usage policy (which is probably
* violated by this sketch when left running for longer)!
*
* To use this sketch, first register your application and device with
* the things network, to set or generate an AppEUI, DevEUI and AppKey.
* Multiple devices can use the same AppEUI, but each device has its own
* DevEUI and AppKey.
*
* Do not forget to define the radio type correctly in config.h.
*/
#include <lmic.h>
#include <hal/hal.h>
#include <SPI.h>

// This EUI must be in little-endian format, so least-significant-byte
// first. When copying an EUI from ttncctl output, this means to reverse
// the bytes. For TTN issued EUIs the last bytes should be 0xD5, 0xB3,
// 0x70.
...
Done uploading.

Sketch uses 22216 bytes (68%) of program storage space. Maximum is 32256 bytes.
Global variables use 991 bytes (48%) of dynamic memory, leaving 1057 bytes for local variables. Maximum is 2048 bytes.

169 Arduino/Genuino Uno on /dev/ttyUSB2

```

[그림 3.23] Arduino program 둘려 보기

이 상태로 LoRaWAN이 정상 동작한다면, 너무 싱겁지 않겠는가? 그래서 아래 세가지 문제(?)를 남겨 두었다.

1) Arduino-LMIC 수정 - KR920 주파수 반영 ***** 난이도 상 !!!

- 설명할 부분이 좀 많아서 자세한 사항은 생략함. 코드는 git에 올려 둘 예정임(위치는 메일로 공지 예정).
- 이거 확인하는데 3일 걸렸음 :(
- 결국은 Dragino Lmic 라이브러리 내용 중 EU868 code를 기반으로 KR920 code 작업을 하여 연결에 성공함 :(

<KR920 관련 참조 site>

<https://github.com/mcci-catena/arduino-lmic>

<https://github.com/promwungkwa/LMIC-Arduino-AS923-upper>

2) ttn-otaa.ino 예제 수정 - APPEUI, DEVEUI, APPKEY 정보 일치 시키기

- 먼저 아래 내용을 참조하여 APPEUI, DEVEUI, APPKEY를 TTN Application Device 정보와 일치시켜야 한다.

3) Pin mapping 정보 일치시키기

```
// Pin mapping
const lmic_pinmap lmic_pins = {
    .nss = 10,
    .rxtx = LMIC_UNUSED_PIN,
    .rst = 9,
    .dio = {2, 6, 7},
};
```

Application ID: dragino_test_application1
Device ID: otaa-device-1
Activation Method: OTAA

Device EUI	<code>lsb [0x90, 0x78, 0x56, 0x34, 0x12, 0x41, 0x40, 0xA8]</code>
Application EUI	<code>lsb [0x18, 0x46, 0x00, 0xF0, 0x7E, 0xD5, 0xB3, 0x70]</code>
App Key	<code>lsb [0xC3, 0x95, 0x15, 0x93, 0xAD, 0x55, 0x1A, 0x83, 0x2F, 0x31, 0x25, 0xB6, 0x7A, ...]</code>

Device Address: 26 01 2D 5E
Network Session Key:
App Session Key:

```
#include <SPI.h>

// This EUI must be in little-endian format, so least-significant-byte
// first. When copying an EUI from ttnc1 output, this means to reverse
// the bytes. For TTN issued EUIs the last bytes should be 0xD5, 0xE3,
// 0x70.
static const ul_t PROGMEM APPEUI[8]={ 0x18, 0x46, 0x00, 0xF0, 0x7E, 0xD5, 0xB3, 0x70 };

void os_getArtEui (ul_t* buf) { memcpy_P(buf, APPEUI, 8);}

// This should also be in little_endian format, see above.
static const ul_t PROGMEM DEVEUI[8]={ 0x90, 0x78, 0x56, 0x34, 0x12, 0x41, 0x40, 0xA8 };

void os_getDevEui (ul_t* buf) { memcpy_P(buf, DEVEUI, 8);}

// This key should be in big endian format (or, since it is not really a
// number but a block of memory, endianness does not really apply). In
// practice, a key taken from ttnc1 can be copied as-is.
// The key shown here is the semtech default key
static const ul_t PROGMEM APPKEY[16] = { 0xC3, 0x95, 0x15, 0x93, 0xAD, 0x55, 0x1A, 0x83, 0x2F, 0x31, 0x25, 0xB6, 0x7A, 0xF5, 0x74, 0x1D };

void os_getDevKey (ul_t* buf) { memcpy_P(buf, APPKEY, 16);}
```

[그림 3.24] APPEUI, DEVEUI, APPKEY 정보 일치 시키기

다시 program을 build & 실행시켜 보자. 아래와 같이 아두이노 시리얼 모니터를 통해 LoRaWAN 통신 과정을 살펴 보면 문제 해결에 도움이 된다. 와우 ~ **EV_JOINED message(LoRaWAN protocol이 정상적으로 동작함을 의미)가 드디어(3일만에) 보인다.**

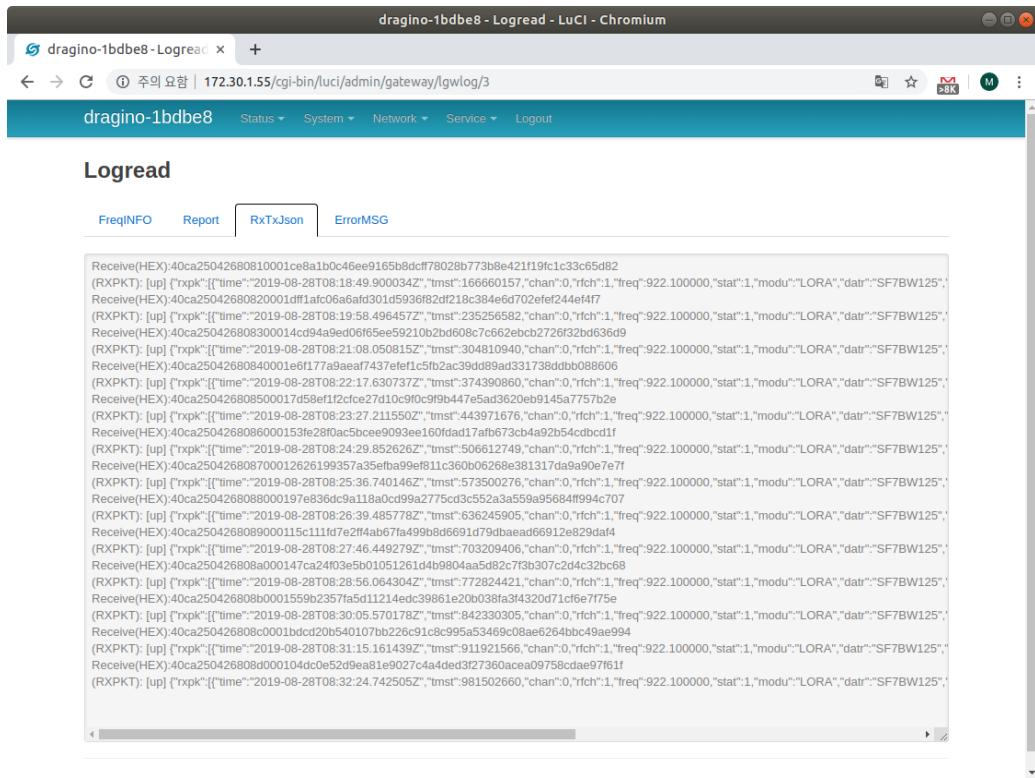
[그림 3.25] Arduino 시리얼 모니터(디버그 모니터)

참고: Dragino LMIC code 뭔가 좀 빠리(?)하다... RAKWireless는 다른 방식으로 구현되어 있을 텐데, 확인해 보아야겠다.

6) LoRa Node ↔ TTN Server 연결 확인하기

이제 우여곡절 끝에 모든 것이 준비되었다. 따라서 LoRa Node \leftrightarrow LoRa Gateway \leftrightarrow TTN Server 간에 LoRa packet이 정상적으로 송/수신되는지를 확인하는 절차만 남았을 뿐이다.

먼저 LoRa Gateway의 디버그 화면(LuCI: Service -> Logread -> RxTxJson tab 선택) 내용을 살펴보도록 하자. 원가 예전에는 없던 많은 내용이 보인다. 자세히 살펴 보면, packet이 up/down되는 것을 알 수 있다.



[그림 3.26] LG01-N Gateway RX/TX 확인

다음으로 TTN Application -> Device Overview 화면을 보니, 역시나 그 동안 없던 내용(OTAA, Network Session Key, App Session Key 등)이 보인다. 이는 곧 정상적으로 LoRaWAN negotiation이 진행되었다는 것을 뜻한다.

The Things Network Console - Chromium
https://console.thethingsnetwork.org/applications/dragino_app2/devices/dr_lora_shield

DEVICE OVERVIEW

Application ID: dragino_app2
Device ID: dr_lora_shield
Activation Method: OTAA

Device EUI: A0 A1 A2 A3 A4 A5 A6 A7
Application EUI: 70 B3 D5 7E D0 02 13 BD
App Key: (redacted)
Device Address: 26 04 25 CA
Network Session Key: (redacted)
App Session Key: (redacted)

Status: 27 seconds ago
Frames up: 97 [reset frame counters](#)
Frames down: 0

[그림 3.27] TTN Application/Device Overview 화면

그 다음으로 확인할 내용은 Gateway Traffic 부분이다. 뭔가 보이지 않던 많은 내용이 올라와 있다.

The Things Network Console - Chromium
https://console.thethingsnetwork.org/gateways/eui-a840411bdbbe84150/traffic

GATEWAY TRAFFIC beta

time	frequency	mod.	CR	data rate	airtime (ms)	cnt	dev addr:	payload size:
11:19:35	922.1	lora	4/5	SF 7 BW 125	77.1	0	26 04 2D 67	35 bytes
11:19:33	922.1		4/5	SF 7 BW 125	71.9			
11:19:29	922.1		4/5	SF 7 BW 125	61.7		app eui: 70 B3 D5 7E D0 02 13 BD dev eui: A0 A1 A2 A3 A4 A5 A6,	
11:18:15	922.1	lora	4/5	SF 7 BW 125	61.7	1	26 04 21 73	26 bytes
11:17:10	922.1	lora	4/5	SF 7 BW 125	61.7	0	26 04 21 73	26 bytes
11:17:09	922.1		4/5	SF 7 BW 125	71.9			
11:17:05	922.1		4/5	SF 7 BW 125	61.7		app eui: 70 B3 D5 7E D0 02 13 BD dev eui: A0 A1 A2 A3 A4 A5 A6,	
11:15:41	922.1		4/5	SF 7 BW 125	61.7		app eui: 70 B3 D5 7E D0 02 13 BD dev eui: A0 A1 A2 A3 A4 A5 A6,	

[그림 3.28] TTN Gateway Traffic 흐름 화면

이 중 한 세션(LoRaWAN에 이런 표현은 없으나)의 내용을 차례로 선택해 보면, Join Request, Join Accept, Uplink 메시지가 보임을 알 수 있다.

The screenshot shows the 'GATEWAY TRAFFIC' section of the The Things Network Console. It lists three traffic entries:

- 18:03:02: uplink, frequency 922.1, mod. lora, CR 4/5, SF 7 BW 125, data rate 77.1, airtime 0 bytes, dev addr: 26 04 2E 4F, payload size: 35 bytes
- 18:03:01: downlink, frequency 922.1, mod. 4/5, SF 7 BW 125, data rate 71.9, airtime 0 bytes, dev addr: 26 04 2E 4F, payload size: 35 bytes
- 18:02:57: uplink, frequency 922.1, mod. 4/5, SF 7 BW 125, data rate 61.7, airtime 0 bytes, app eui: 70 B3 D5 7E D0 02 13 BD, dev eui: A0 A1 A2 A3 A4 A5 A6.

Join Request

Dev EUI
A8 A1 A2 A3 A4 A5 A6 A7

App EUI
70 B3 D5 7E D0 02 13 BD

Physical Payload
00 B0 13 02 D0 7E D5 B3 70 A7 A6 A5 A4 A3 A2 A1 A0 91 58 1C 0F 79 94

[그림 3.29] TTN Gateway Traffic - Join Request Packet

The screenshot shows the 'GATEWAY TRAFFIC' section of the The Things Network Console. It lists two traffic entries:

- 18:03:02: uplink, frequency 922.1, mod. lora, CR 4/5, SF 7 BW 125, data rate 77.1, airtime 0 bytes, dev addr: 26 04 2E 4F, payload size: 35 bytes
- 18:03:01: downlink, frequency 922.1, mod. 4/5, SF 7 BW 125, data rate 71.9, airtime 0 bytes, dev addr: 26 04 2E 4F, payload size: 35 bytes

Join Accept

Physical Payload
20 3A 7F D1 F2 28 48 51 95 76 33 14 98 28 71 A1 B8 96 F2 75 4D 5B 8C 00 63 F0 1F B3 A0 D0 3C 0E D8

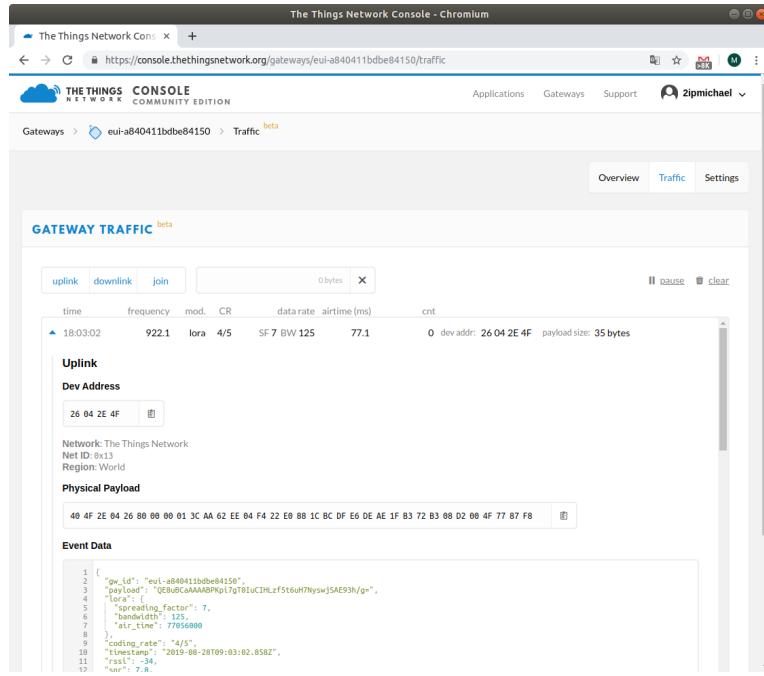
Event Data

```

1 [ {
2   "gw_id": "eui-a840411dbe84150",
3   "payload": "D0@f1rSFQWjMkCxobhJ8nVh4wAY/Afs6QPA7Y",
4   "lorawan": {
5     "frequency": 922.1,
6     "bandwidth": 125,
7     "air_time": 71936000
8   },
9   "coding_rate": "4/5",
10  "timestamp": "2019-08-28T09:03:01.563Z",
11  "frequency": 922100000
12 }
]

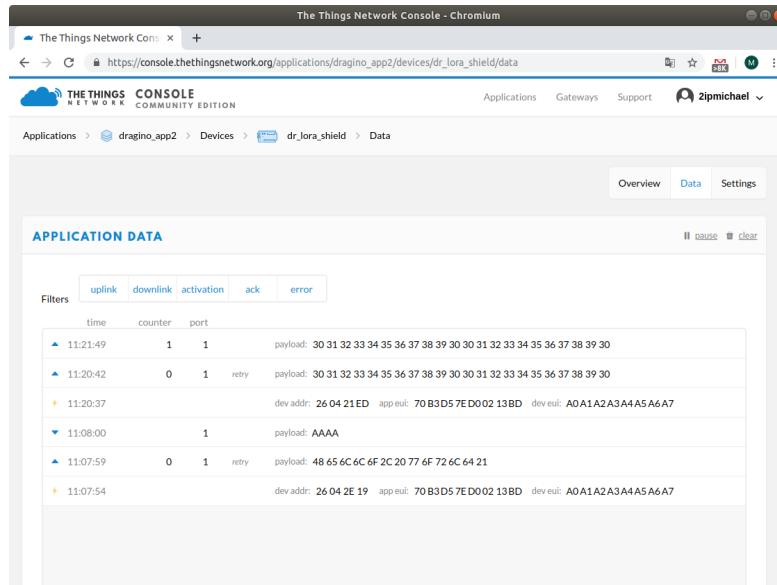
```

[그림 3.30] TTN Gateway Traffic - JoinAccept Packet



[그림 3.31] TTN Gateway Traffic - Uplink Packet

끝으로 Application Data page 내용을 살펴 보도록 하자. 앞서 LoRa Node에서는 LoRa 망을 타고 "hello, world!" string을 보낸 바 있다. 아래 그림에서 이를 확인할 수가 있다. 어디 보이는가 관련 내용이 ?



[그림 3.32] TTN Application Data 출력 화면

여기까지 가장 기본적인 LoRa 네트워크 구성 & 동작 시험을 진행해 보았다.

<TBD>

- 1) 지금까지는 OTAA mode로 통신하는 과정을 해 보았는데, ABP mode 테스트도 해 보아야 한다.
- 2) "hello, world!"가 아닌 실제 sensor로 부터의 data가 LoRa server로 전달되도록 해 보아야 한다.
- 3) Cayenne Application Server, MQTT IoT Server와의 연동 시험도 진행해 보자.

4. Dragino LG308 LoRa Gateway

이번 절에서는 Dragino LG308 LoRa Gateway와 LGT-92 GPS Tracker를 연결시켜 보도록 하겠다. 지난주에 주문했는데, 몇일만에 바로 왔다~

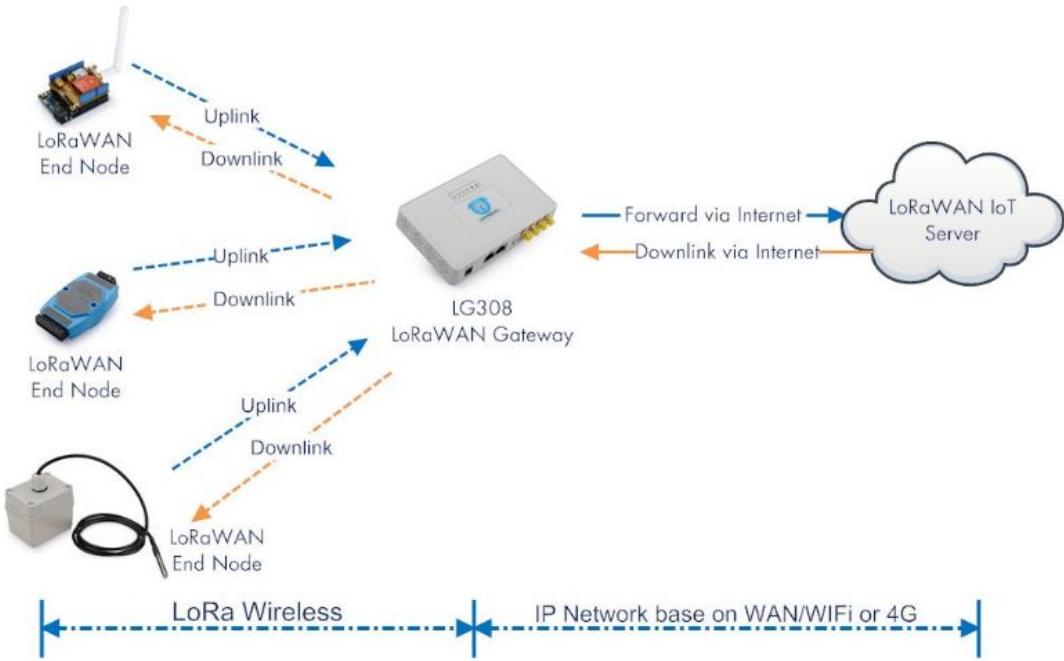
여기서 설명하는 내용은 아래 두개의 문서를 참조하여 정리하였다. 따라서, 본 절의 내용을 따라해 보기 전에, 아래 두 문서를 미리 읽어 보길 권한다.

<LG308 LoRa Gateway>

https://www.dragino.com/downloads/index.php?dir=LoRa_Gateway/LG308-LG301/

<LGT-92 LoRa GPS Tracker>

https://www.dragino.com/downloads/index.php?dir=LGT_92/



[그림 4.1] LG308을 이용한 LoRa Network 구성도

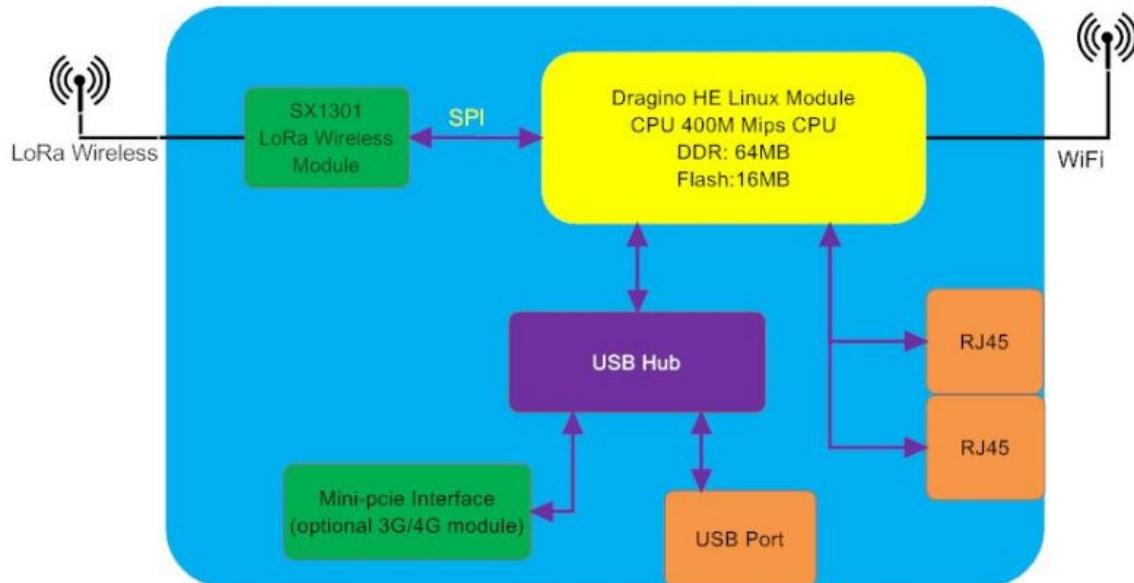
1) LG308 LoRa Gateway 설정하기

LG01-N LoRa Gateway와 달리 **LG308 LoRa Gateway는 다중 채널을 지원하는 녀석(?)으로 상용 버전으로 사용가능하다(과연 그럴까 ?).**



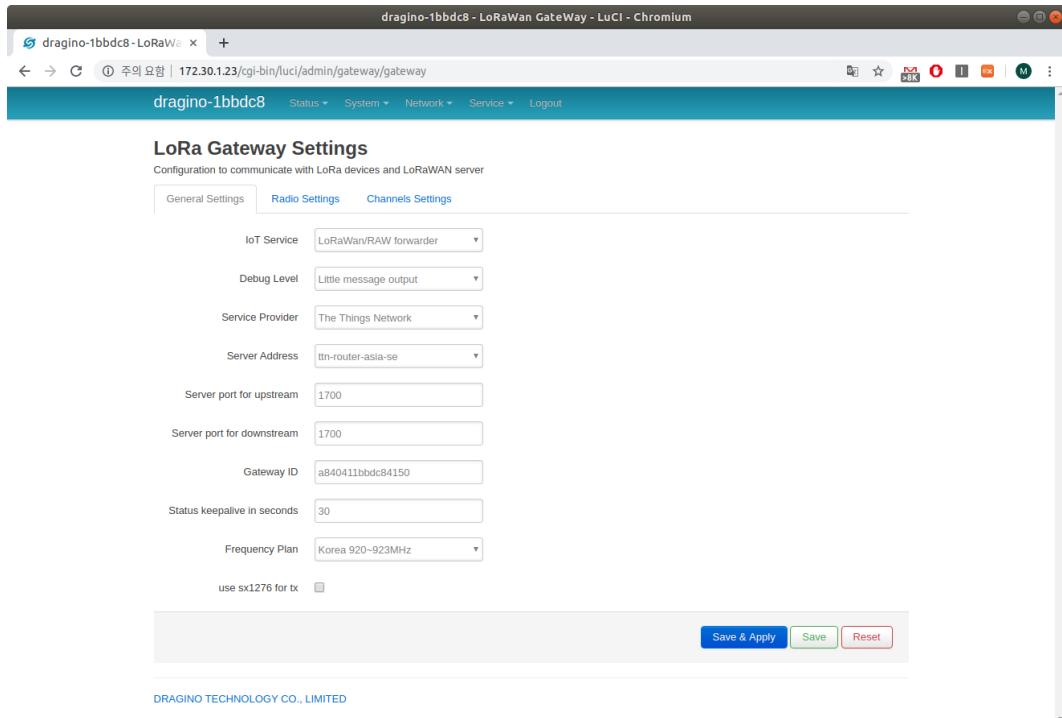
[그림 4.2] LG308 LoRa Gateway 연결 모습

아래 그림은 LG308 보드의 block diagram이다. SX1301은 Gateway에서 사용(다중 채널 지원용)하는 LoRa module인데, CPU(AR9331)하고는 역시 SPI로 연결되어 있다. 나머지 구조는 LG01-N과 크게 다를 바가 없다.



[그림 4.3] LG308 Block Diagram

내부 구조를 살펴 보았으니, WebUI 화면을 통해 LoRa Gateway 설정을 할 차례가 되었다. 아래 그림을 보면 알겠지만, 설정 자체는 크게 어려움이 없다. LG01-N과 약간 달라진 부분이 있는데, 차이점을 파악해 보기 바란다.



[그림 4.4] LG308 LoRa Gateway WebUI 설정 화면

<참고 사항 - LoRa Gateway 설정 시 꼭 알아두어야 하는 내용>

1. IoT Service: **LoRaWan/RAW forwarder**
2. Service Provider: **The Things Network**
3. Server Address: **ttn-router-asia-se** (얘는 다른 것으로 지정해도 됨)
4. Server port for upstream/downstream: **1700/1700** (LoRa packet forwarder는 server랑 UDP 통신을 함)
5. Gateway ID: **default** 값 그대로 사용 (끝 자리 숫자는 변경해도 됨. 단, 전세계에서 유일해야 함)
6. Frequency Plan: **Korea 920~923Mhz**
7. Use sx1276 for tx: **선택 안함.**

자, Gateway 설정이 끝났으니, 이제부터는 TTN 서버에 LG308 Gateway를 등록해 보자. 등록 절차는 3장에서 이미 자세히 설명하였으므로, 여기서는 등록 후의 모습만 살펴 보기로 하자.

The screenshot shows the 'Gateway Overview' section for a gateway with ID 'eui-a840411bbdc84150'. Key details include:

- Gateway ID:** eui-a840411bbdc84150
- Description:** LG308_LoRa_Gateway_Michael
- Owner:** zipmichael (Transfer ownership)
- Status:** connected
- Frequency Plan:** Korea 920-923MHz
- Router:** ttn-router-asia-se
- Gateway Key:** (redacted)
- Last Seen:** 21 seconds ago
- Received Messages:** 688
- Transmitted Messages:** 39

[그림 4.5] TTN - LG308 Gateway 등록 모습(1)

등록 후, Gateway Traffic을 살펴 보면, LoRa Node가 설치되지 않았음에도 불구하고 아래와 같이 많은 내용이 gateway로 부터 올라옴을 알 수 있다. 이 트래픽을 하나씩 선택해서 살펴 본 바, SK Telecom이라는 문구가 보인다. 그렇다면, 이 트래픽의 주변에 있는 SK LoRa Node로 부터 올라온 packet들이라고 봐야 할 것이다.

time	frequency	mod.	CR	data rate	airtime (ms)	cnt
▲ 17:56:50	923.1	lora	4/5	SF 11 BW 125	741.4	1945 dev addr: 1A 00 EB E9 payload size: 19 bytes
▲ 17:56:43	922.3	lora	4/5	SF 11 BW 125	1806.3	1944 dev addr: 1A 00 EB E9 payload size: 77 bytes
▲ 17:56:40	922.5	lora	4/5	SF 11 BW 125	987.1	345 dev addr: 1A 01 1D 3C payload size: 35 bytes
▲ 17:56:28	922.7	lora	4/5	SF 11 BW 125	987.1	344 dev addr: 1A 01 1D 3C payload size: 35 bytes
▲ 17:56:26	922.9	lora	4/5	SF 11 BW 125	1069.1	14225 dev addr: 1A 00 BE SC payload size: 37 bytes
▲ 17:56:00	922.3	lora	4/5	SF 11 BW 125	1069.1	14224 dev addr: 1A 00 BE SC payload size: 37 bytes
▲ 17:55:39	922.9	lora	4/5	SF 12 BW 125	1810.4	70 dev addr: 1B 0F 12 DD payload size: 35 bytes
▲ 15:01:46	922.9	lora	4/5	SF 11 BW 125	1970.2	7442 dev addr: 1A 00 27 C2 payload size: 86 bytes
▲ 15:01:30	922.1	lora	4/5	SF 12 BW 125	1810.4	18 dev addr: 1B 0F 12 DD payload size: 35 bytes
▲ 15:01:29	922.5	lora	4/5	SF 11 BW 125	987.1	221 dev addr: 1A 01 1D 3C payload size: 35 bytes

[그림 4.6] TTN - LG308 Gateway 등록 모습(2)

2) LGT-92 LoRa GPS Tracker

다음으로 LGT-92 GPS Tracker(LoRa Node)를 준비해 보자. GPS Tracker는 GPS 위성으로 부터 위치 정보(위도/경도)를 수신한 후, 이를 LoRa 망을 통해 서버(예: TTN)로 전달하는 장치이다.



[그림 4.7] LGT-92 LoRaWAN GPS Tracker

LGT-92 GPS Tracker의 주요 스펙을 아래에 간단히 정리해 보면 다음과 같다. Micro controller(MCU)는 ST Micro의 STM32L072CZT6로, 블록도를 아래에 그려 놓았다.

Micro Controller:

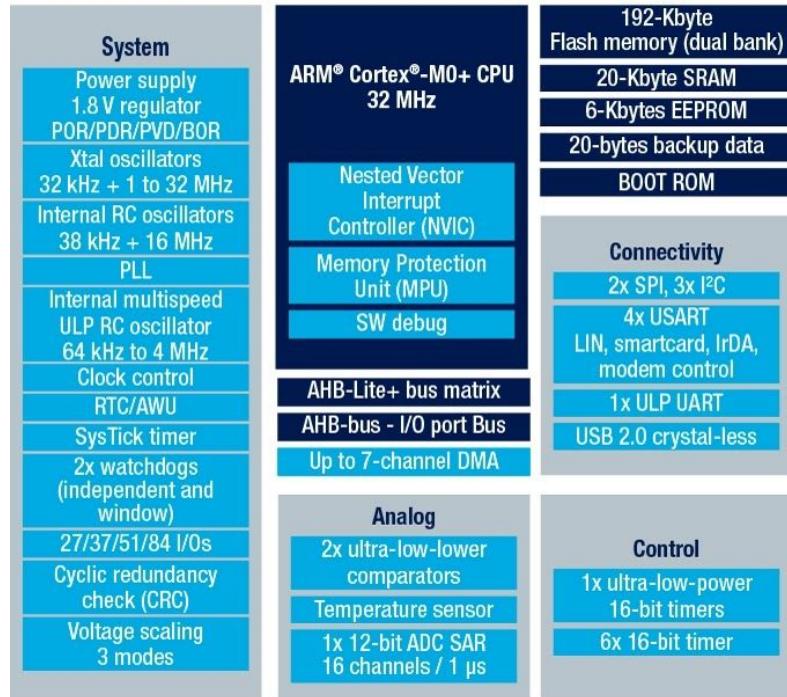
- STM32L072CZT6 MCU
- MCU: STM32L072CZT6
- Flash: 192KB
- RAM: 20KB
- EEPROM: 6KB
- Clock Speed: 32Mhz

Features:

- LoRaWAN 1.0.2 Class A, Class C
- STM32L072CZT6 MCU
- SX1276/78 Wireless Chip
- Pre-load bootloader on USART1/USART2
- MDK-ARM Version 5.24a IDE
- Preamble detection
- Frequency bands CN470/EU433/KR920/US915/IN865
EU868/AS923/AU915
- Open source hardware / software
- Regular/ Real-time GPS tracking
- Built-in 9 axis accelerometer (MPU9250)
- Motion sensing capability
- Power Monitoring

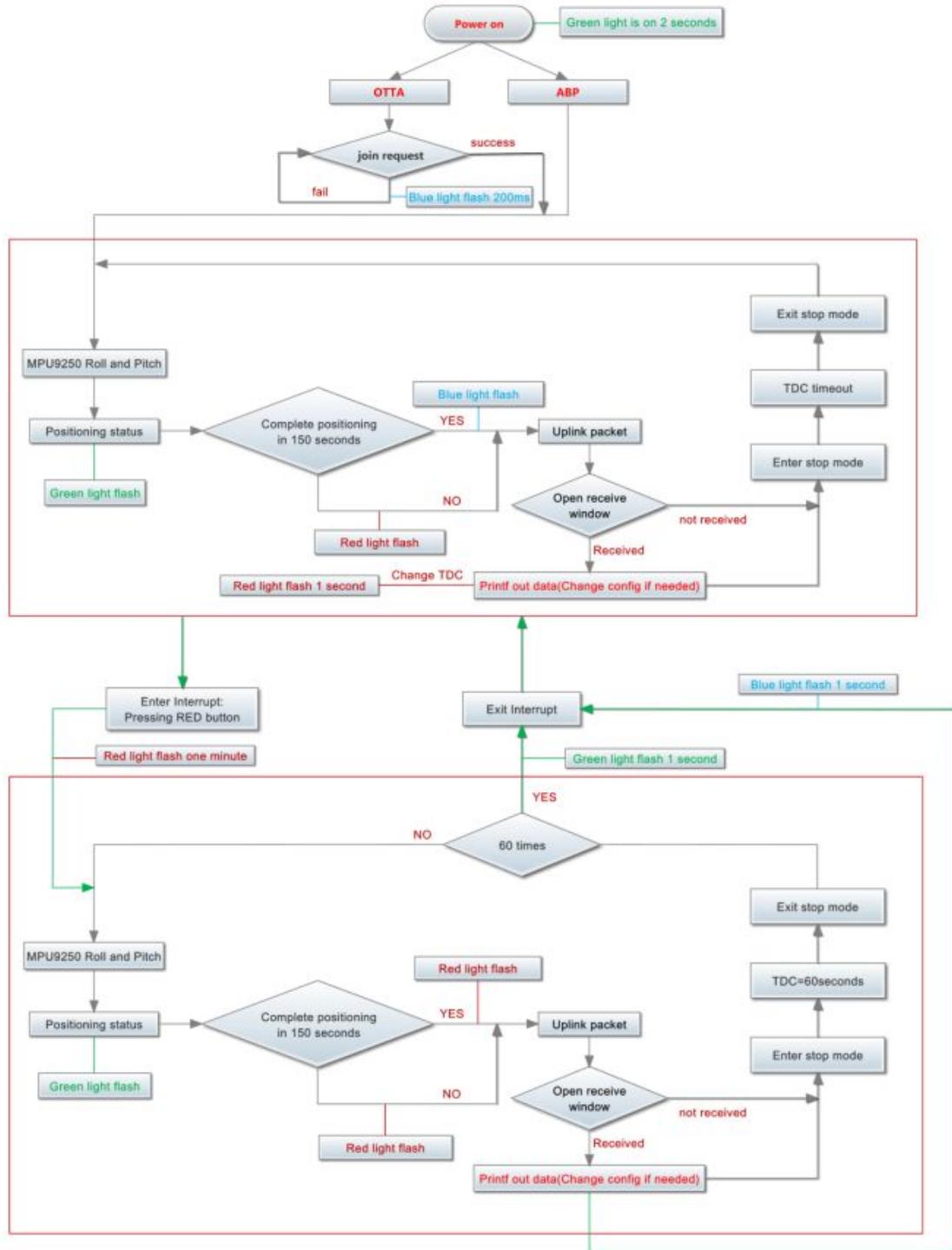
Charging circuit via USB port (for model LGT-92-LI)
 1000mA Li-on Battery power (for model LGT-92-LI)
 2 x AA battery holder for 1.5v AA battery (for model LGT-92-AA)
 Tri-color LED, Alarm button

STM32L072xZ



[그림 4.8] STM32L072CZT6 MCU

LGT-92 GPS Tracker의 동작 원리가 궁금한데, 이는 다음 그림과 같다(자세한 내용은 Dragino 문서 참조 요망).



[그림 4.9] LGT92 LoRaWAN GPS Tracker firmware 동작도

3장에서 설명한 것처럼, LoRa Node를 Network 서버와 OTAA를 통해 통신하기 위해서는 사전에 몇가지 정보를 맞추는 과정이 필요하다. GPS Tracker 포장 box 안쪽을 보면 아래와 같은 sticker가 하나 붙어 있다. 이 중, OTAA 설정을 위해서는 DEV EUI, APP EUI, APP Key가 필요(TTN application device 설정시 적용)하다.

참고: APPSKEY, NETSKEY: OTAA mode의 경우는 서버와 packet을 교환하는 과정에서 자동으로 생성되는 키를 이용하면 된다. 따라서 여기에 표시된 두 값은 ABP mode에서 사용하기 위한 것이다.

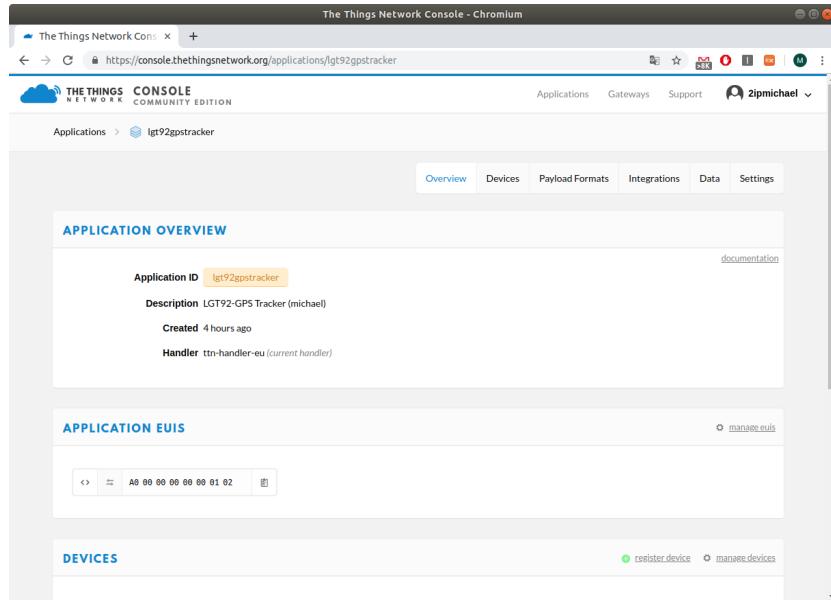


[그림 4.10] LGT92 LoRaWAN GPS Tracker 설정 정보

<OTAA 설정시 필요한 정보>

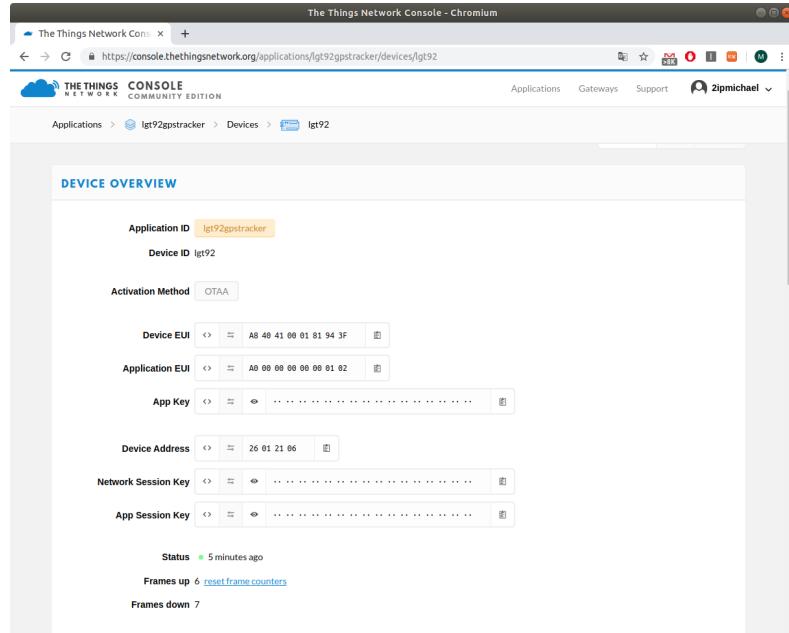
DEV EUI : A84041000181943F
 APP EUI : A0000000000000102
 APP Key : { 0xD8, 0x1C, 0xB3, 0x23, 0x5C, 0xEA, 0x33, 0xD8, 0xCC, 0xD1, 0xBF, 0x6B, 0x1D, 0x54, 0x29, 0xD1 }

자, 이제부터는 TTN 서버에 LGT-92를 위한 Application & device를 각각 등록해 보기로 하겠다. 역시 자세한 사항은 이미 소개한 바 있으므로 여기서는 등록 후의 모습만을 정리해 보았다.



[그림 4.11] TTN - Application/LGT-92 device 등록 모습(1)

참고: Device를 등록하면, 자동으로 APP EUI(APPLICATION EUI)가 생성되는데, 이 값은 그림 4.10의 그것과 다른 값이다. 따라서, 4.10의 내용으로 APP EUI를 갱신(기존 것은 삭제)해 주어야 한다.



[그림 4.12] TTN - Application/LGT-92 device 등록 모습(2)

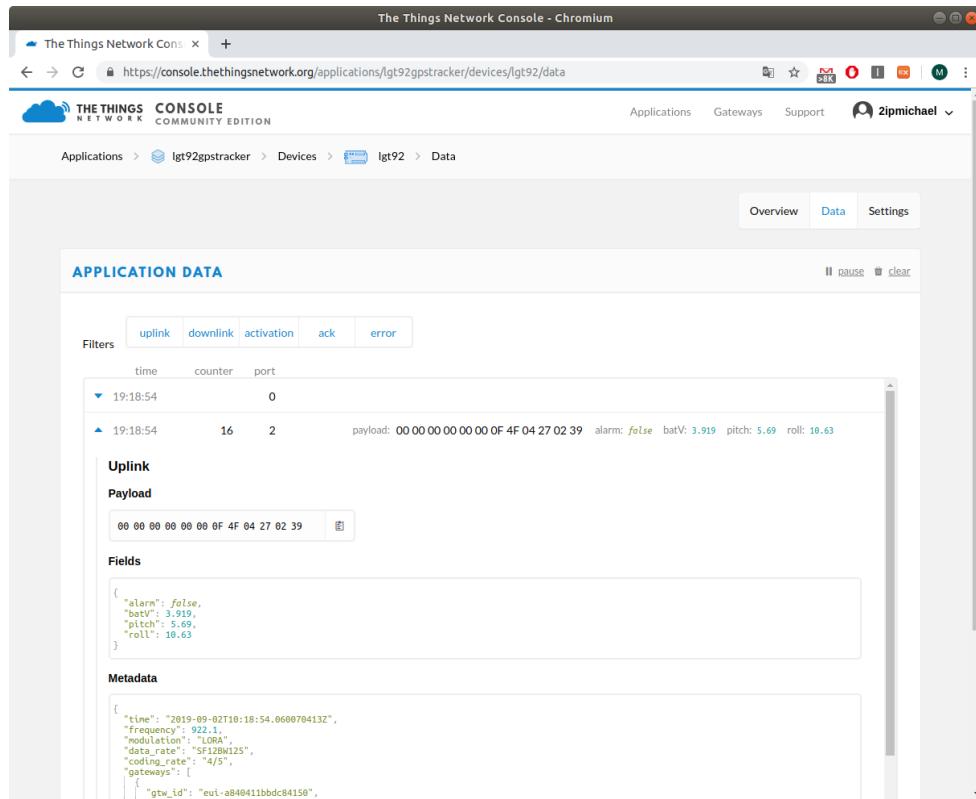
자, 모든 것이 준비되었다. 아래 그림을 보면 알 수 있듯이, GPS Tracker로 부터 data가 올라옴(5분 주기)을 알 수 있다. **근데, 아무리 봐도 payload 내용 중에 위도/경도 관련 내용이 없고, 그 자리(처음 6 byte)에 0이 채워져 있다. GPS data가 정상 수신되지 않고 있는 느낌인데.... 왜일까?**

time	counter	port	payload	alarm	batV.	pitch	roll
18:04:07	0		00 00 00 00 00 0F 61 FF E1 FDBE	false	3.937	-5.78	-8.31
18:04:06	7	2	00 00 00 00 00 0F 7C FF E2 FD B8	false	3.964	-5.84	-8.3
17:52:56	0		00 00 00 00 00 0F 5E FF E4 FD B5	false	3.934	-5.87	-8.28
17:52:55	5	2	00 00 00 00 00 0F 6A FF E3 FD A8	false	3.946	-5.94	-8.29
17:47:20	0		00 00 00 00 00 0F 74 FF FD FD B8	false	3.956	-6.27	-8.83
17:47:20	4	2	00 00 00 00 00 0F 6F F3 2A 1B 00	false	3.951	-69.12	-32.86
17:41:45	0						
17:41:45	3	2					
17:36:10	0						
17:36:09	2	2					
17:30:34	0						
17:30:34	1	2					

[그림 4.13] TTN - Application/LGT-92 device 트래픽 확인

<LoRa Payload format>

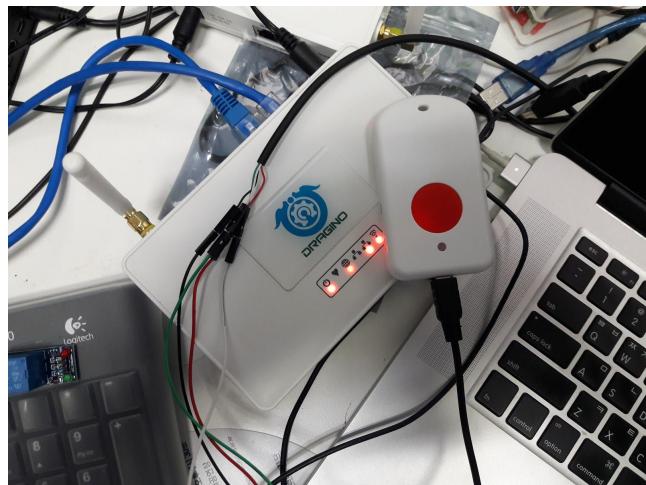
Size(bytes)	3	3	2	2	2
Value	Latitude	Longitude	Alarm & BAT	Roll	Pitch



[그림 4.14] TTN - Application/LGT-92 device GPS data 분석

3) LGT-92 LoRa GPS Tracker 디버깅하기

왜 GPS data가 전달되지 않을까 ? 원인 파악을 위해 몇가지 시도를 해 보기로 하자.



[그림 4.15] LGT-92 GPS Tracker 시리얼 케이블 연결 모습

주의 : Serial cable의 RX/TX 선을 cross로 연결해 주어야만 한다.

정상 연결 시, 아래와 같이 LGT-92의 동작 내용이 콘솔로 출력될 것이다. 이 상태에서 LGT-92용 AT command를 입력하면 해당 명령이 먹히는 것을 확인할 수 있다.

```

chyi@mars: ~
파일(F) 편집(E) 보기(V) 검색(S) 터미널(T) 도움말(H)

Welcome to minicom 2.7.1

OPTIONS: I18n
Compiled on Aug 13 2017, 15:25:34.
Port /dev/ttyUSB0, 19:24:51

Press CTRL-A Z for help on special keys

♦
LGT-92 Device
Image Version: v1.4
Frequency Band: KR920
DevEui= A8 40 41 00 01 81 94 3F

***** UplinkCounter= 0 *****
TX on freq 922500000 Hz at DR 5
txDone
rxDone
JOINED
■

CTRL-A Z for help | 9600 8N1 | NOR | Minicom 2.7.1 | VT102 | Offline | ttyUSB0

```

[그림 4.16] LGT92 GPS Tracker - Serial Console(9600, 8N1) 모습(minicom)

참고: AT command 제어를 위해 minicom 대신 microcom을 사용해 보도록 하자.

<TBD>

4) LGT-92 LoRa GPS Tracker Firmware Writing

<TBD>

ST-Link V2 장치가 있어야 함. 구매하자 ~~

5) LGT-92 LoRa GPS Tracker source code build하기

<TBD - Keil>

애는 꼭 알아두어야 하는 내용이다. 정말 field에서 많이 사용한다. 직원들에게 꼭 소개시켜 주어야 겠다.

5. RAKWireless RAK7258 LoRa Gateway

이번 장에서는 RAKWireless 사 제품인 **RAK7258 RAK Micro Gateway**를 검토해 보기로 하겠다.
오늘(2019. 09.03 화) 장비가 도착했다 :) 결론 부터 얘기하자면, 아무래도 Dragino 보다는 RAKWireless로 가야할 것 같다. 완성도에서 차이가 난다.

<물건 구매 site>

<https://store.rakwireless.com/products/rak7258-micro-gateway>

<RAKWireless 제품 document site - 여기에 있는 문서를 읽어 보라>

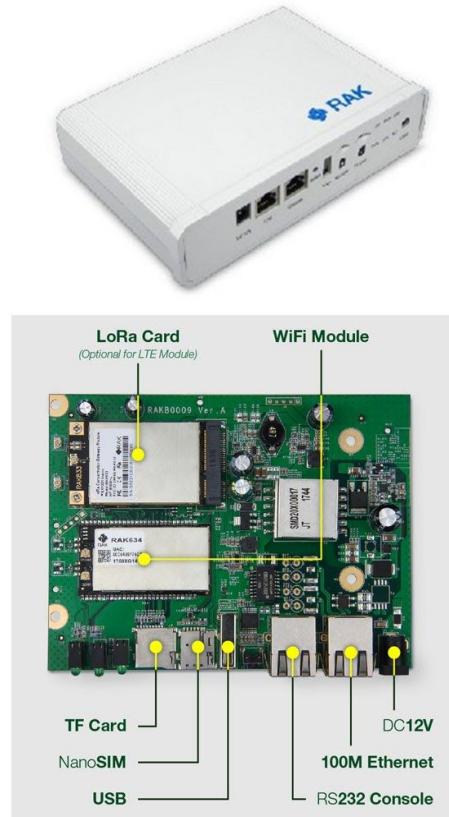
<https://downloads.rakwireless.com/en/>

1) RAK7258 LoRa Gateway Overview

RAK7258의 장비 외관 및 보드 구성 내용은 다음과 같다. 비슷한 내용 여러번 정리하려니 지친다. 따라서 설명은 생략 :)



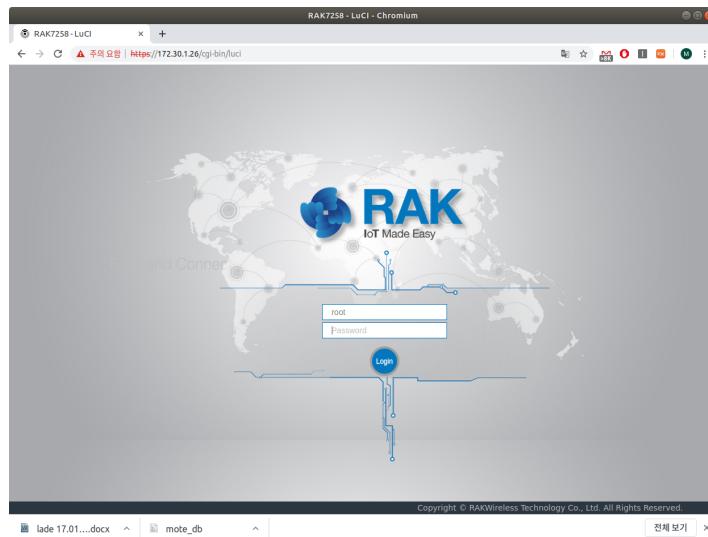
[그림 5.1] RAK7258 LoRa Gateway



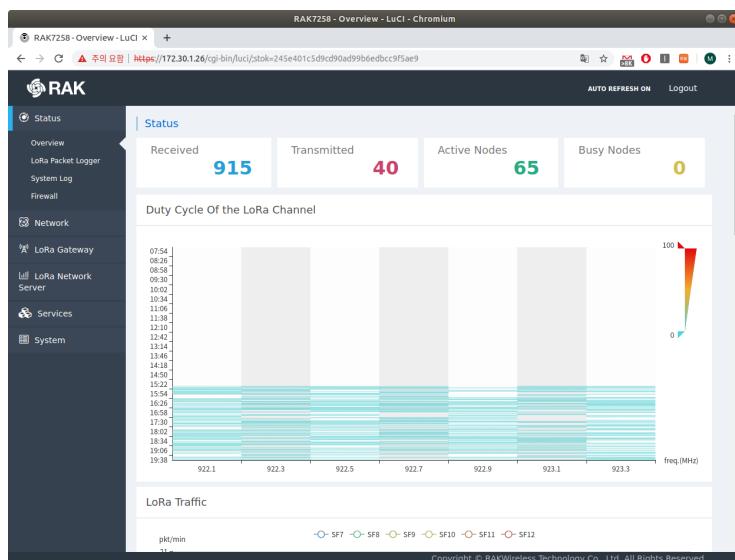
[그림 5.2] RAK7258 LoRa Gateway & 보드 개요

2) RAK7258 LoRa Gateway 설정하기

RAK7258 WebUI는 LuCI를 기반으로 하고 있으나, 모양은 완전 새롭다. 여느 상용 제품과 비교해도 크게 손색이 없을 만큼 잘 만들어진 느낌이다.



[그림 5.3] RAK7258 LoRa Gateway WebUI - 로긴 화면(id/pass: root/root)



[그림 5.4] RAK7258 LoRa Gateway WebUI - Status 화면

자, 그럼 이제부터는 LoRa Gateway 설정을 시작할 차례이다. 아래 내용을 보면 알겠지만, 크게 보아 아래 몇가지 설정만 변경해 주면 된다. 간단하다~

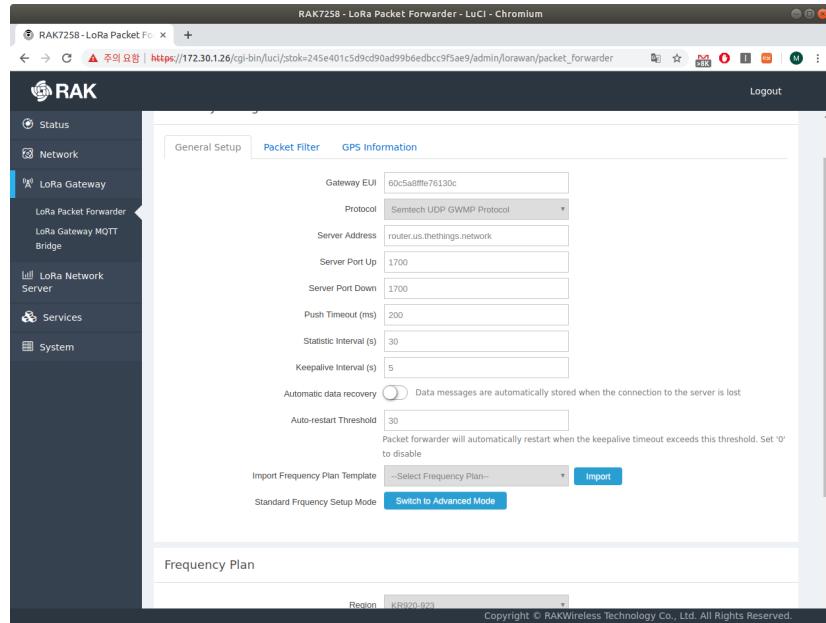
<LoRa Gateway 설정>

Protocol : Semtech UDP GWMP Protocol

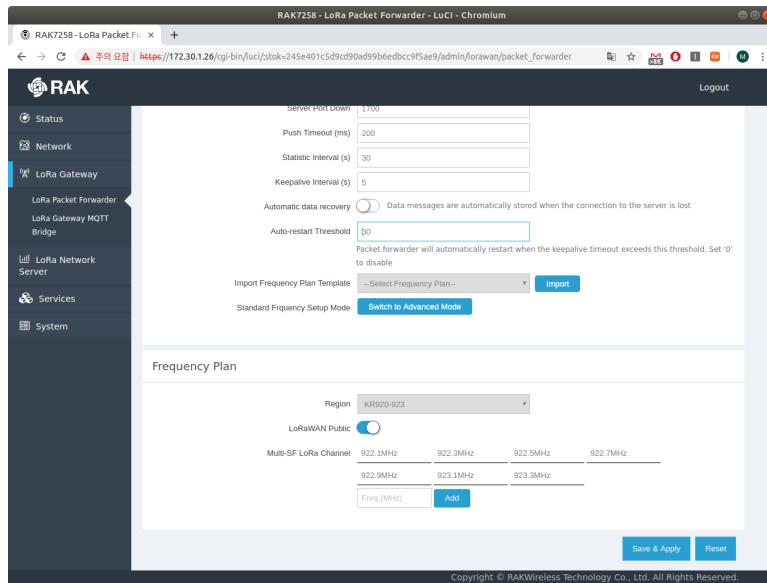
Server Address : **router.us.thethings.network** (이건 다른 걸로 지정해도 됨)

Server Port Up/Down : **1700/1700**

Import Frequency Plan Template : KR920-923

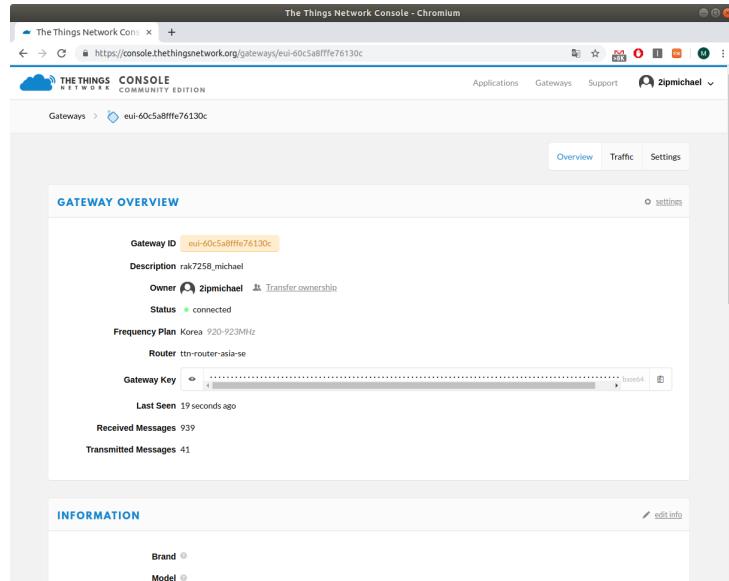


[그림 5.5] RAK7258 LoRa Gateway WebUI - LoRa Gateway 설정 화면(1)



[그림 5.6] RAK7258 LoRa Gateway WebUI - LoRa Gateway 설정 화면(2) - KR920 주파수 설정

Gateway 설정(packet forwarder 설정)이 끝났으니, TTN 서버에 Gateway를 등록해 보자(이미 앞서 여러 차례 설명한 내용이니 자세한 내용은 생략).



[그림 5.7] RAK7258 LoRa Gateway - TTN Gateway 설정(1)

GATEWAY TRAFFIC beta						
uplink	downlink	join	0bytes	X	pause	clear
▲ 19:49:34	922.7	lora 4/5	SF 11 BW 125	1806.3	3610	dev addr: 1A 00 EB E2 payload size: 77 bytes
▲ 19:49:30	922.3	lora 4/5	SF 7 BW 125	97.5	13	dev addr: 26 01 2D B6 payload size: 49 bytes
▲ 19:49:24	923.1	lora 4/5	SF 11 BW 125	823.3	3609	dev addr: 1A 00 EB E2 payload size: 27 bytes
▼ 19:49:12	921.9	lora 4/5	SF 12 BW 125	1318.9	26	dev addr: 26 01 27 56 payload size: 17 bytes
▲ 19:49:11	922.3	lora 4/5	SF 12 BW 125	1482.8	27	dev addr: 26 01 27 56 payload size: 25 bytes
▲ 19:48:41	923.1	lora 4/5	SF 11 BW 125	1970.2	695	dev addr: 1A 00 CFD payload size: 86 bytes
▲ 19:48:35	923.3	lora 4/5	SF 12 BW 125	1810.4	533	dev addr: 1B 0F 12 DD payload size: 35 bytes
▲ 19:47:59	922.9	lora 4/5	SF 12 BW 125	2957.3	31	dev addr: 1B 05 04 87 payload size: 66 bytes
▲ 19:47:58	922.9	lora 4/5	SF 7 BW 125	97.5	12	dev addr: 26 01 2D B6 payload size: 49 bytes
▲ 16:47:38	923.3	lora 4/5	SF 12 BW 125	1318.9	4	dev addr: 1A 01 07 D9 payload size: 18 bytes
▲ 16:47:21	922.7	lora 4/5	SF 12 BW 125	1646.6	2	dev addr: 1A 01 07 D9 payload size: 27 bytes
▼ 16:47:11	922.3	4/5	SF 12 BW 125	1482.8	app euil: 02 50 79 10 00 00 02 01 dev euil: D0 25 44 FF FE FC 33	

[그림 5.8] RAK7258 LoRa Gateway - TTN Gateway 설정(2)

3) RAK7200 LoRa Tracker 설정하기

RAK7258 LoRa Gateway와 함께 **RAK7200 GPS tracker**도 같이 구입했다. 생각보다 Tracker의 모양이 예쁘지 않다. 역시 중국산 :(어쨌거나, 충전부터 하자.

이제 부터는 얘를 LoRa 망에 연결해 보기로 하겠다.

<참고 문서>

https://downloads.rakwireless.com/en/LoRa/RAK7200-Tracker/Application_Notes/

→ Get_Start_with_RAK7200_Tracker_Device.pdf



[그림 5.9] RAK7200 LoRa GPS Tracker

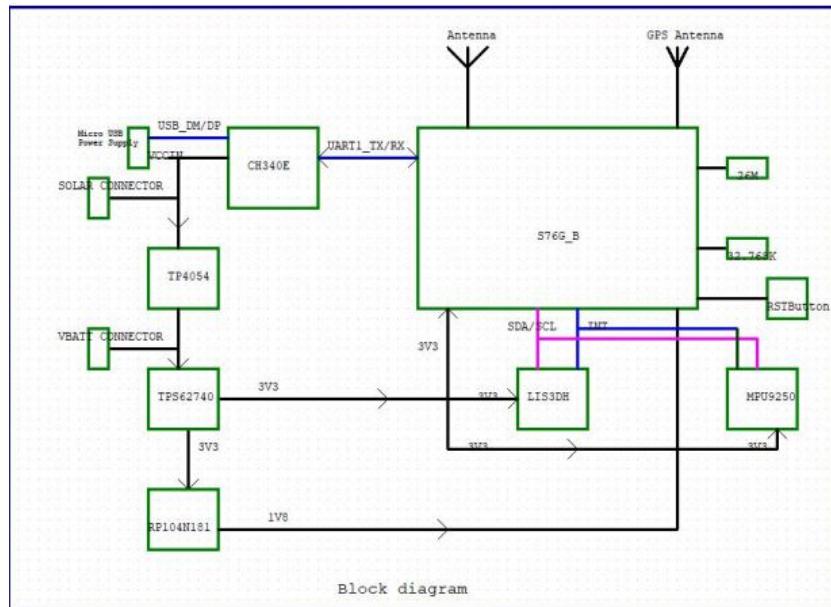
<RAK7200 LoRa Tracker>

"The RAK7200 is a LoRa Tracker device with a 3.7V rechargeable Li-Ion battery and a GPS modem. It is built around a low-power LoRa module, which integrates the ultra low power STM32L073 micro-controller, the SX1276 LoRa long range modem and the CXD5603GF GPS modem.

The device can be used as a quick prototyping tool for LoRaWAN application development. It is suited for IoT Applications such as asset tracking, smart vehicle management and location-based services."

<RAK7200 주요 특징>

- 1) MCU - **STM32L073** ← 역시나 (이 영역의 강자인) STMicro chip을 사용하고 있다.
- 2) **SX1276 LoRa module**
- 3) **LoRaWAN 1.0.2 protocol stack 지원** - OTAA/ABP mode 지원
- 4) 지원 가능한 LoRa Band - **EU868, US915, AS923, AU915, KR920, IN865**
- 5) **CXD5603CF GPS modem**
- 6) **motion sensor** - LIS3DH
- 7) **9-axis(9축) sensor** - 3축 gyroscope, 3축 accelerometer, 3축 magnetometer로 구성



[그림 5.10] RAK7200 LoRa GPS Tracker H/W 블록도

<여기서 잠깐 - RAK7200 firmware upgrade에 관하여>

RAK7200 firmware upgrade에 관해서는 아래 문서에 잘 정리되어 있다.

https://downloads.rakwireless.com/en/LoRa/RAK7200-Tracker/Application_Notes/Get_Started_with_RAK7200_Tracker_Device.pdf

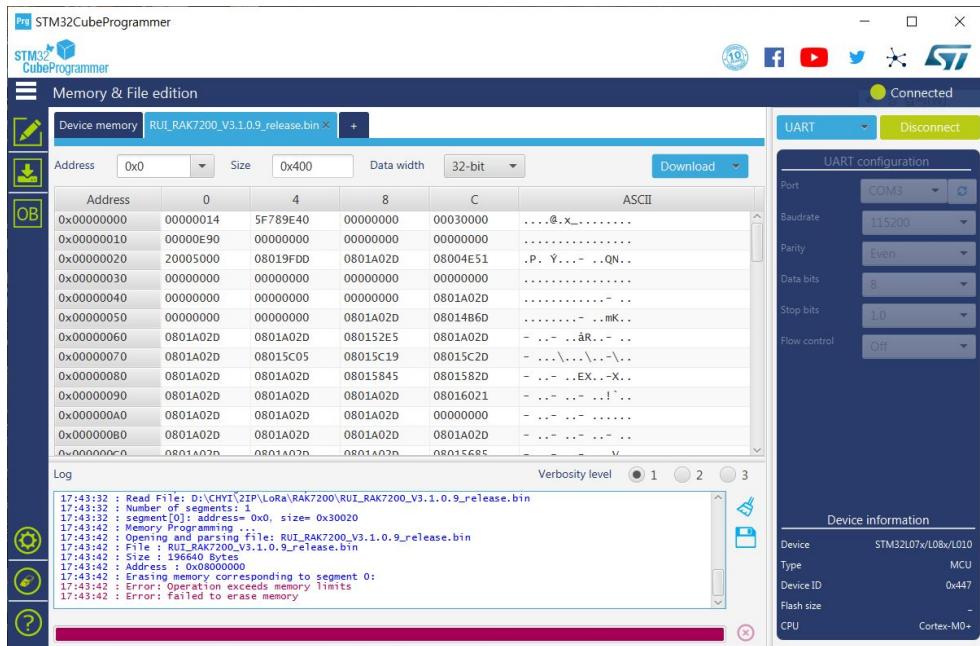
<Firmware Upgrade 절차 요약>

1. RAK7200 firmware download - 위의 문서 site
RUI_RAK7200_V3.1.0.9_release.rar => RUI_RAK7200_V3.1.0.9_release.bin
2. ST사에 제공하는 STM32CubeProgrammer tool 설치
3. RAK7200으로 부트 모드로 진입하도록 함(BOOT0 button, Reset Button 이용)
4. STM32CubeProgrammer tool 실행 후, UART mode로 변경
5. Flash memory erase
6. RUI_RAK7200_V3.1.0.9_release.bin 파일 open 후, File download 버튼 선택하여 flash memory에 firmware writing

**(문제 발생) Memory limit 에러 발생하며, writing 실패함. Shit~ 아무래도 file size가
큰 모양임. RAK 애들이 file을 잘 못 올린게 분명함. Shit~ F**king~**

5단계에서 flash memory를 erase(boot 영역을 제외한 모든 영역)한 관계로 이후 부팅 안됨.

아무래도 항의 메일을 보내야 겠다.



RAKWireless forum에 계정을 등록하고, 아래 내용을 올렸다.

<https://forum.rakwireless.com/t/rak7200-firmware-upgrade-failure/873>

RAK7200 firmware upgrade failure 🖊

■ LoRa/LoRaWAN



Michael Chunghan Yi

1m

Hi,

I tried firmware upgrade for RAK7200 using the RUI_RAK7200_V3.1.0.9_release.bin file. But the STM32CubeProgrammer tool showed "Error: Operation exceeds memory limits" popup after erasing flash memory.

It seems that firmware file size(196640 bytes) on your web site is greater than the flash size of STM32L073 MCU which is 192 Kbytes.

Can anyone help this issue ?

=====

```
-rw-r--r-- 1 chyi chyi 196640 9월 30 18:14 RUI_RAK7200_V3.1.0.9_release.bin
-rw-rw-r-- 1 chyi chyi 87950 10월 2 16:20 RUI_RAK7200_V3.1.0.9_release.rar
```

... Reply

 Robo14850 Charlie Heath 4h

Michael,

You should be using "RAK LoRaButton Upgrade Tool V1.0" to upgrade the firmware. "STM32CubeProgrammer" is just for upgrading the boot loader. You need to issue the "at+set_config=device:boot" command before upgrading the firmware.

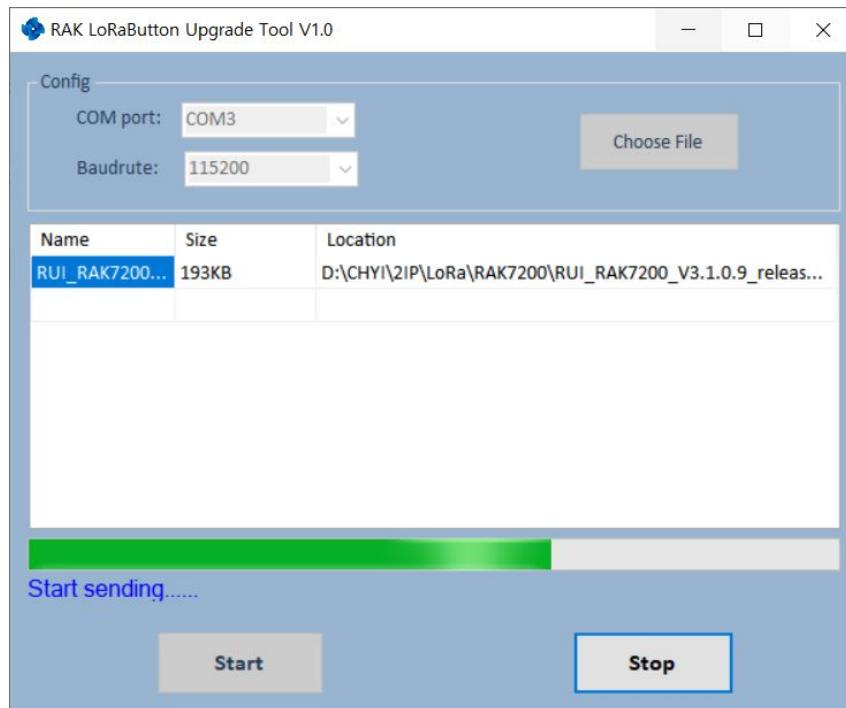
HTH,
Charlie

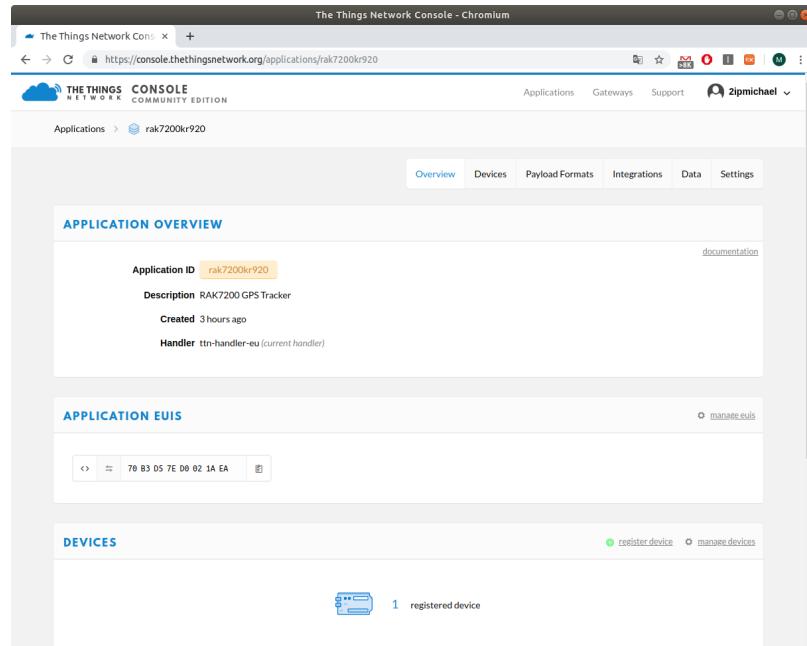
   

결국, RAK LoRaButton Upgrade Tool V1.0을 사용하여 겨우 upgrade 성공했다. :)

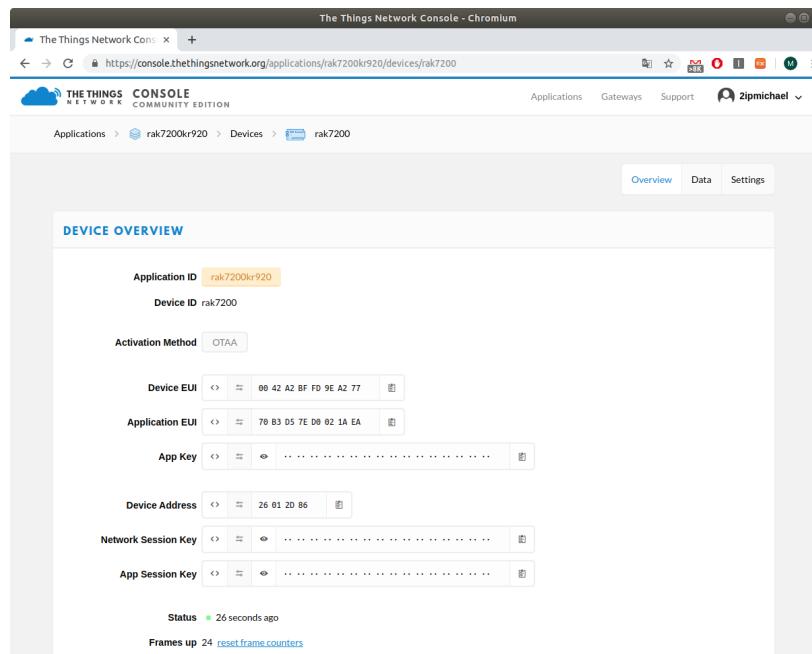
아무리 봐도 program이 좀 이상하다.

- a) STM32CubeProgrammer tool로 erase all
- b) STM32CubeProgrammer tool bootloader writing(RUI_RAK7200_S76G_BOOT.bin)
- c) 아래 Tool 띄운 후, 파일 open
- d) Start 버튼 선택 - 에러 발생
- e) Stop 후, reset button 선택 후, 다시 Start ⇒ OK writing 시작 ~





[그림 5.11] RAK7200 LoRa GPS Tracker - TTN Application 추가(1)



[그림 5.12] RAK7200 LoRa GPS Tracker - TTN Application 추가(2)

RAK7200 LoRa Tracker의 아래 세가지 설정을 TTN Application의 그것과 동일하게 맞춰야 한다.

<LoRa Node와 TTN Server간에 맞추어야 하는 정보>

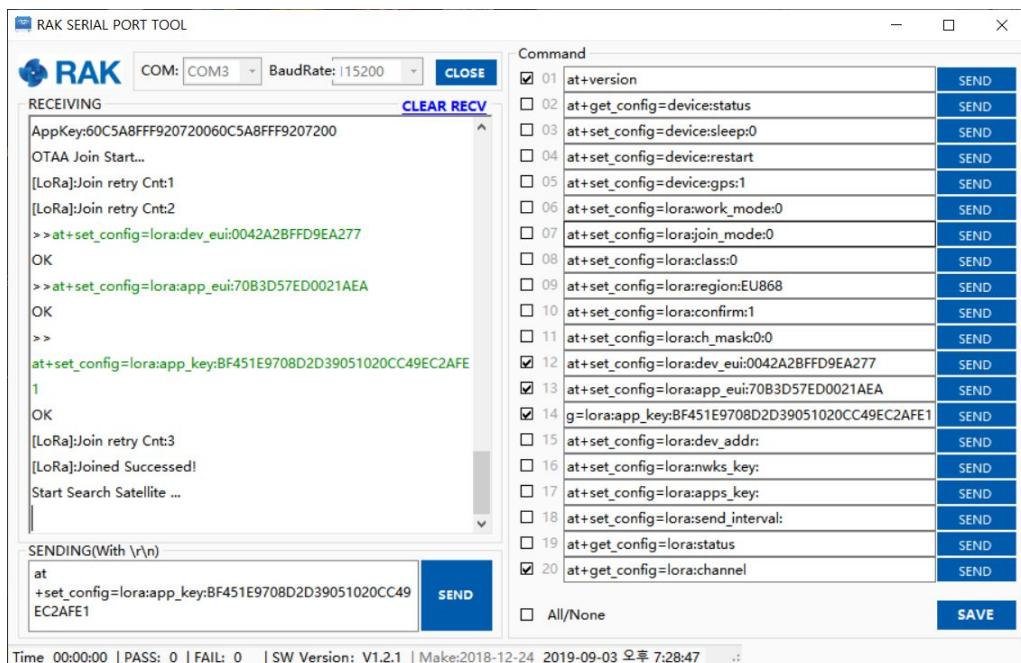
Device EUI - 0042A2BFFD9EA277
Application EUI - 70B3D57ED0021AEA
App Key - BF451E9708D2D39051020CC49EC2AFE1

RAK7200의 설정을 바꾸기 위해서는 **RAK에서 개발한 serial emulator program을 설치**(RAK에서 제공하는 문서 참조)한 후, RAK7200용 AT command를 사용해야만 한다.

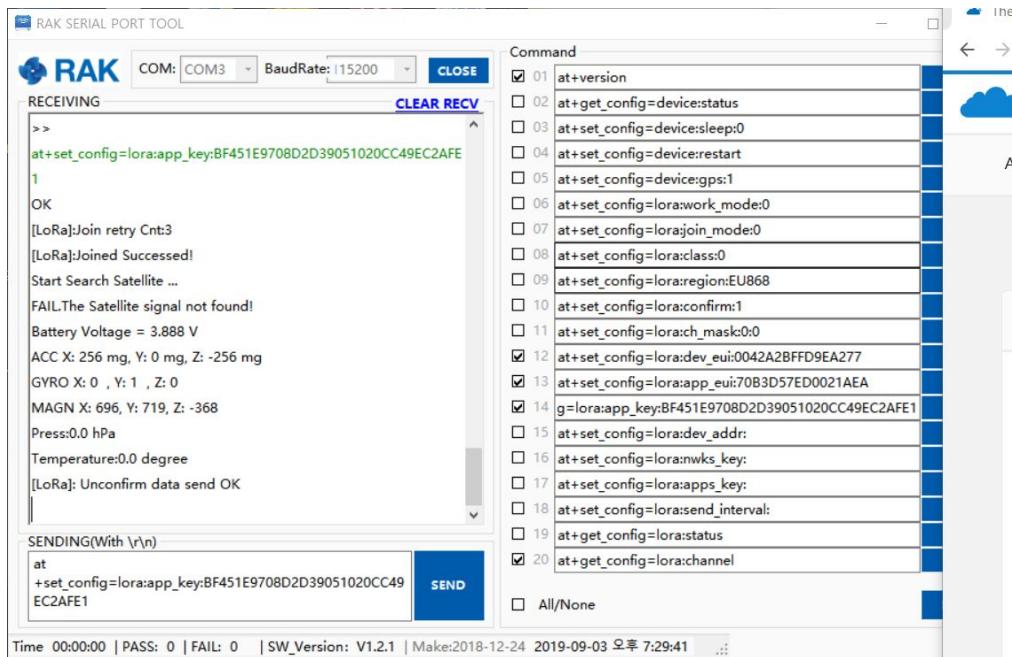
사용 방법은 아주 간단하다. 장치 인식 후(COM port, 115200 baudrate), 우측의 Command 부분에 아래 EUI or key 값을 넣은 후, SEND 버튼을 누르면 된다.

<AT command 예제>

```
at+set_config=lora:dev_eui:0042A2BFFD9EA277
at+set_config=lora:app_eui:70B3D57ED0021AEA
at+set_config=lora:app_key:BF451E9708D2D39051020CC49EC2AFE1
```



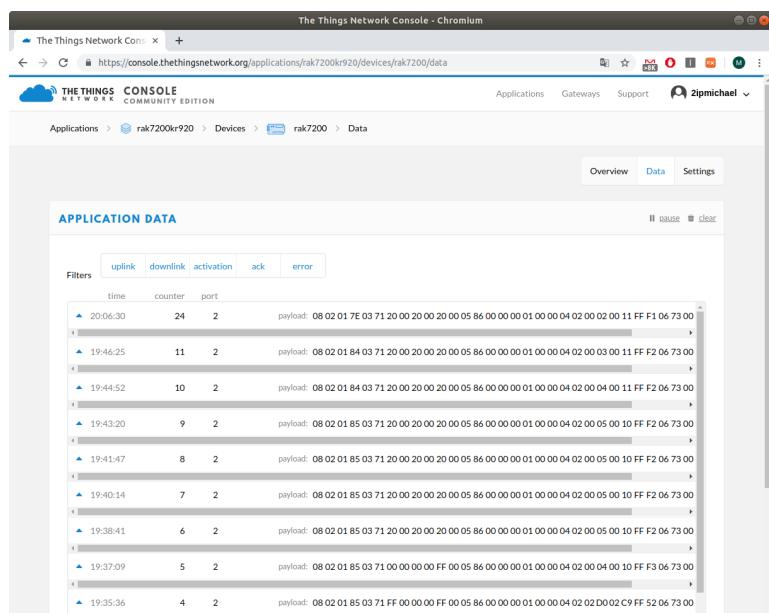
[그림 5.13] RAK7200 LoRa GPS Tracker 설정 변경(1)



[그림 5.14] RAK7200 LoRa GPS Tracker 설정 변경(2)

참고: 위의 AT command가 제대로 먹히게 하려면, RAK7200을 reset(reset 버튼을 sharp 펜으로 살짝 눌러 주자)해 준 상태에서 시도하면 된다.

정상적으로 설정이 되었다면, 아래와 같이 Application/Device 화면에 uplink packet이 하나씩 올라오는 것이 보일 것이다(아래 내용은 시간이 좀 지난 후, capture한 것이라 내용이 많음).



[그림 5.15] RAK7200 LoRa GPS Tracker - TTN Application 추가(3)

4) TTN 서버 대신 Built-in LoRa Server 사용하기

RAK7258은 아주 재미있게도, **내부에 별도의 LoRa Server**(RAK에서 별도로 개발한 듯)를 장착하고 있다(Oh~ 아주 훌륭한데). 아래 그림에서 /usr/sbin/lorasrv가 이에 해당한다. 따라서 애플리케이션을 활용한다면, 굳이 외부 TTN 서버를 이용하지 않아도 된다.

```

179 root      0 SW  [spi32766]
250 root      0 SW< [deferwq]
337 root      0 SW  [mmcqd/0]
340 root      0 SW  [kworker/u2:3]
396 root      0 SWN [jffs2_gcd_mtd6]
478 root      904 S /sbin/ubusd
561 root      768 S /sbin/askfirst /bin/login.sh
1191 root     1208 S logread -f -F /mnt/mmcblk0p1/syslog/20190903-112641.log -p /var/run/rot
1226 root     1056 S /sbin/logd -S 16
1235 root     1820 S /sbin/rpcd
1280 root     1604 S /sbin/netifd
1318 root     1500 S /usr/sbin/crond -f -c /etc/crontabs -l 5
1384 root     1500 S udhcpc -p /var/run/udhcpc-eth0.2.pid -s /lib/netifd/dhcp.script -f -t 0
1388 root     1148 S /usr/sbin/dropbear -F -P /var/run/dropbear.1.pid -p 22 -K 300
1391 root      0 SW  [RtmpCmd0Task]
1392 root      0 SW  [RtmpWscTask]
1393 root      0 SW  [RtmpMlmeTask]
1441 root     2484 S /usr/sbin/uhttpd -f -h /www -r RAK7258 -x /cgi-bin -u /ubus -t 60 -T 30
1456 root      776 S /usr/sbin/rteswplugged -r /etc/eswplug.action
1465 mosquitt 3548 S /usr/sbin/mosquitto -c /etc/mosquitto/mosquitto.conf
1475 root     2260 S lora_pkt_fwd -g /var/etc/global_conf.json
1477 root     22904 S lora_pkt_fwd -g /var/etc/global_conf.json
1501 root     4644 S /usr/sbin/loragwbr -C /var/etc/loragwbridge.json
1503 root     4804 S /usr/sbin/loragwbr -C /var/etc/loragwbridge.json
1544 root     1504 S /usr/sbin/ntpd -n -S /usr/sbin/ntpd-hotplug -p 0.openwrt.pool.ntp.org -
1551 root     812 S /usr/bin/mpstat -f -t -u 5
1641 nobody   984 S /usr/sbin/dnsmasq -C /var/etc/dnsmasq.conf -k -x /var/run/dnsmasq/dnsma
3130 root     1216 S /usr/sbin/dropbear -F -P /var/run/dropbear.1.pid -p 22 -K 300
3158 root     1508 S -ash
4307 root      0 SW  [kworker/0:2]
4402 root     4644 S /usr/sbin/lorasrv -C /var/etc/loraserver.json -D /mnt/mmcblk0p1/lorasrv
4403 root     5000 S /usr/sbin/lorasrv -C /var/etc/loraserver.json -D /mnt/mmcblk0p1/lorasrv
12752 root    1500 R ps
root@RAK7258:/tmp/etc# 
```

[그림 5.16] RAK7258 - ssh login 후 ps 명령 실행 모습(lorasrv 주목)

< TestBed >

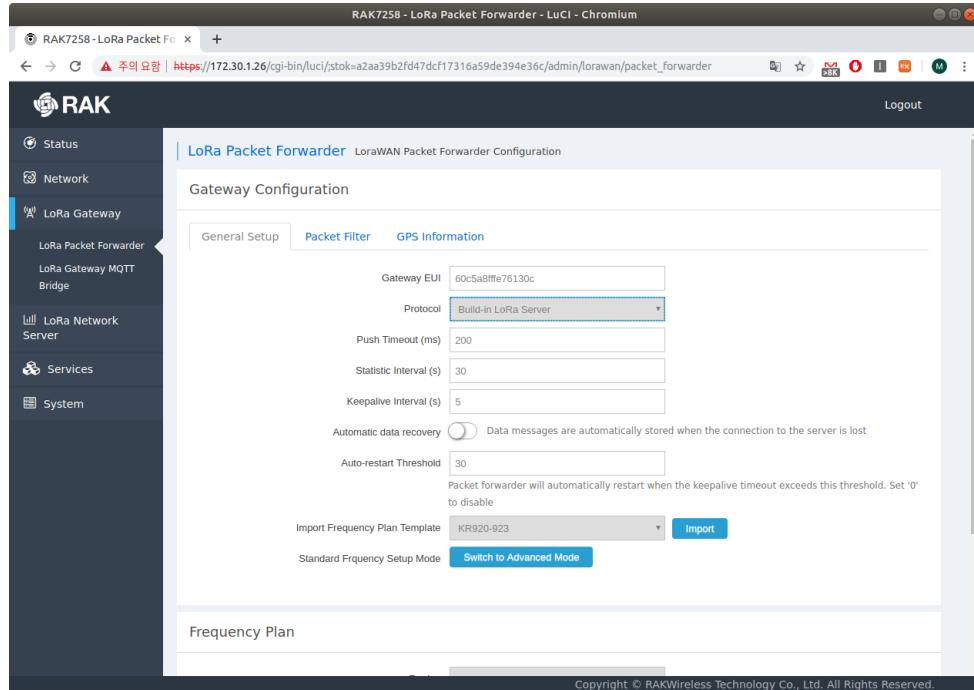
LoRa Node(RAK7200 GPS Tracker) ⇒ RAK7258 LoRa Gateway ⇒ RAK7258 LoRa Server(LoRa Gateway와 같은 장비)

<참고 문서>

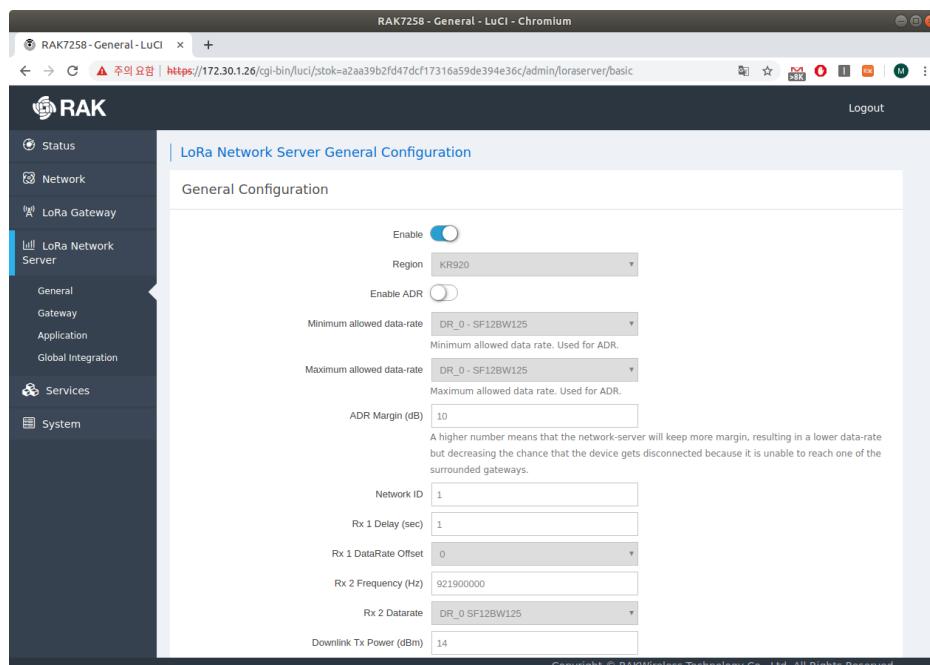
<https://downloads.rakwireless.com/en/LoRa/Indoor-Gateway-RAK7258/Application-Notes/>

→ LoRaWAN_Deployment_Scenario1_V1.0.pdf

실제 설정 작업은 매우 간단하여, 여기서는 화면 capture만으로 설명을 대신하기로 하겠다. 자세한 사항은 위의 pdf 파일을 참고하기 바란다.



[그림 5.17] RAK7258 Built-in LoRa Server 구동(1)



[그림 5.18] RAK7258 Built-in LoRa Server 구동(2)

RAK7258 - Gateway - LuCI - Chromium

RAK7258 - Gateway - LuCI - Chromium

LoRa Network Server Gateway Overview

Gateway EUI	Name	Add time	Description	Last Seen
60c5a8ffe76130c	rak7258_michael1	Wed Sep 4 01:30:04 2019	this gateway	0 seconds ago

Gateway Backend Configuration

General Setup MQTT Topic

MQTT Broker Address: 127.0.0.1
 MQTT Broker Port: 1883
 Client ID:
 Clean Session:
 Will Retain:
 QoS: 1 - At least Once
 keepalive: 10

[그림 5.19] Built-in LoRa Network Server - Gateway 등록

RAK7258 - Application - LuCI - Chromium

RAK7258 - Application - LuCI - Chromium

LoRa Network Server Application Overview

ID	Name	Devices	Creation Date	Description
1	rak7200_tracker1	1	Wed Sep 4 01:33:33 2019	LoRa Tracker

Please input application name Add

Save & Apply Reset

[그림 5.20] Built-in LoRa Network Server - Application/Device 등록

[그림 5.21] Built-in LoRa Network Server - LoRa node 패킷 수신 확인(1)

LoRa Network Server ⇒ Application ⇒ Edit ⇒ Device name(한개 선택) ⇒ Live Deve Data tab을 선택하면, RAK7200 GPS Tracker로 부터 패킷이 정상적으로 올라오는 것을 확인할 수 있다.

[그림 5.22] Built-in LoRa Network Server - LoRa node 패킷 수신 확인(2)

5) RAK7258을 external LoRaServer에 연결하기

이번 절에서는 RAK7258 LoRa Gateway를 external LoRaServer(<https://www.loraserver.io/>)에
붙이는 과정을 소개해 보고자 한다. **이 부분이 이 문서를 통틀어 본 저자가 궁극적으로 확인해
보고자 하는 내용이라고 말할 수 있다.** 이게 된다면, LoRa Node, LoRa Gateway, LoRa Server가
준비되는 셈이고, 우리는 MQTT Broker(mosquitto MQTT broker 사용 예정)와 자체 app(Web
기반 & android/iOS 기반) 만 개발하면 된다는 얘기가 된다.

본론으로 들어가기 전에 반드시 아래 파일을 읽어 보기 바란다.

<https://downloads.rakwireless.com/en/LoRa/Indoor-Gateway-RAK7258/Application-Notes/>

→ LoRaWAN_Gateway_MQTT_Bridge+TLS_Configuration_Guide_V1.1.pdf

위 문서에서는 TLS certificate을 이용하여 RAK7258과 LoRa Server(network server)를 안전하게
연결하는 방법을 소개하고 있다. (결론부터 얘기하자면) 근데, 문서대로 해 보니 잘 안된다.

참고: 안되도 상관 없다. 우리는 SPN으로 할 거니까~

<발생한 문제점>

- 1) TLS Certificate 관련하여 비검증된 CA에서 발급한 것이라 사용할 수 없다는 에러 발생
 - 2) TLS 인증을 하지 않고 할 경우, MQTT broker와 lora-gateway-bridge 간의 문제로 역시 동작
안함.
- 이 부분은 아직 정확한 원인 파악 못함(TBD)

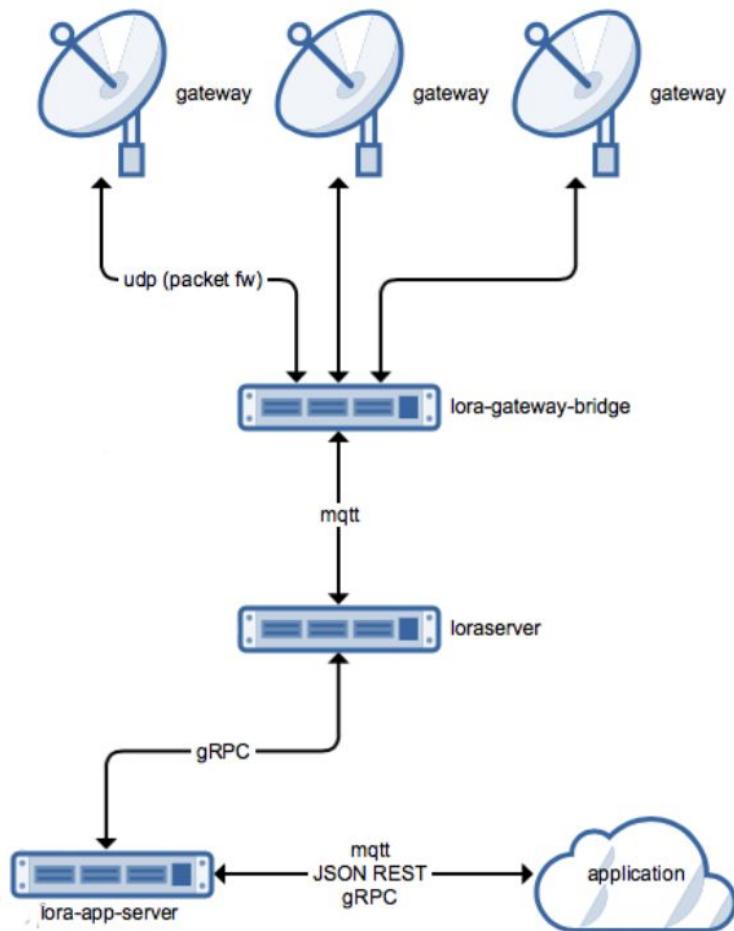
따라서 위 문서 내용과는 약간 다르지만 TTN network server에 연결하는 방식과 동일한
스타일로 문제를 해결해 보고자 한다.

<위 문서 내용 기준 TestBed>

**LoRa Node(RAK7200 GPS Tracker) ⇒ RAK7258 LoRa Gateway ⇒ (MQTT bridge ⇔ MQTT
Broker) ⇒ LoRa Server(external PC)**

<새로운 TestBed>

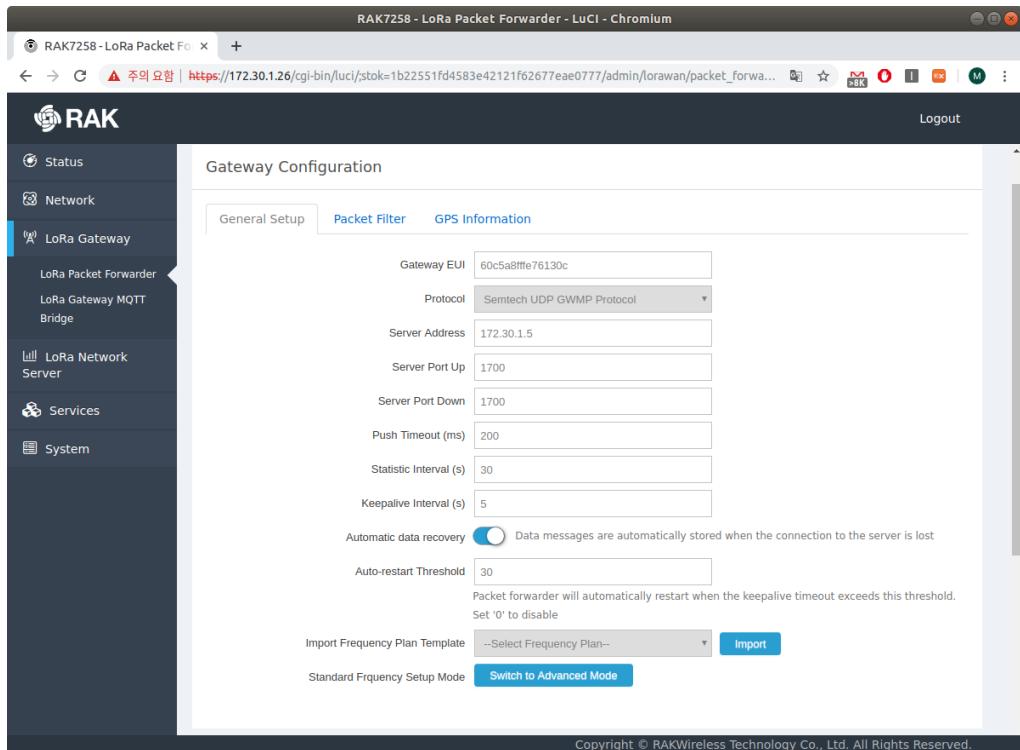
**LoRa Node(RAK7200 GPS Tracker) ⇒ RAK7258 LoRa Gateway ⇒ (packet forwarder ⇔
lora-gateway-bridge) ⇒ LoRa Server(external PC) => LoRa App Server(external PC)**



[그림 5.23] 새로운 테스트 베드 관련 네트워크 구성

위 문서에서는 **LoRa Gateway MQTT Bridge**를 활용(여기서 직접 LoRa Server도 패킷 전달)하도록 하고 있으나, 여기서는 **Semtech Packet forwarder**를 사용하여 **LoRa Gateway Bridge**로 패킷을 전달하는 방식(위 그림 5.23 방식)을 사용하기로 하겠다.

먼저 RAK7258 LoRa Gateway의 설정을 아래와 같이 변경하도록 하자. 즉 Protocol을 **Semtech UDP GWMP Protocol**로 변경하고, Server Address는 LoRa Server(실제로는 LoRa Gateway Bridge)의 ip인 172.30.1.5(my Linux PC)로 맞추어 주도록 하자.



[그림 5.24] RAK7258 WebUI 설정 - Protocol 변경

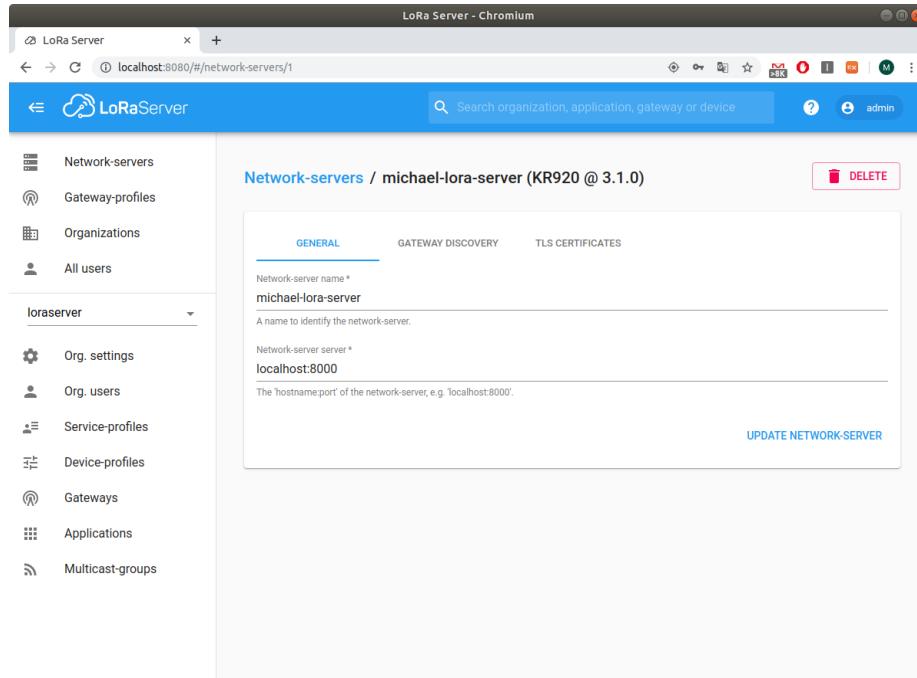
LoRa Gateway 설정이 끝났으니, 이제는 LoRa Server 쪽(정확히는 LoRa App Server 쪽) 설정을 할 차례이다.

LoRa App Server를 설정하는 방법은 TTN 과 유사하다. 자세한 방법은 아래 URL을 참조하기 바란다.

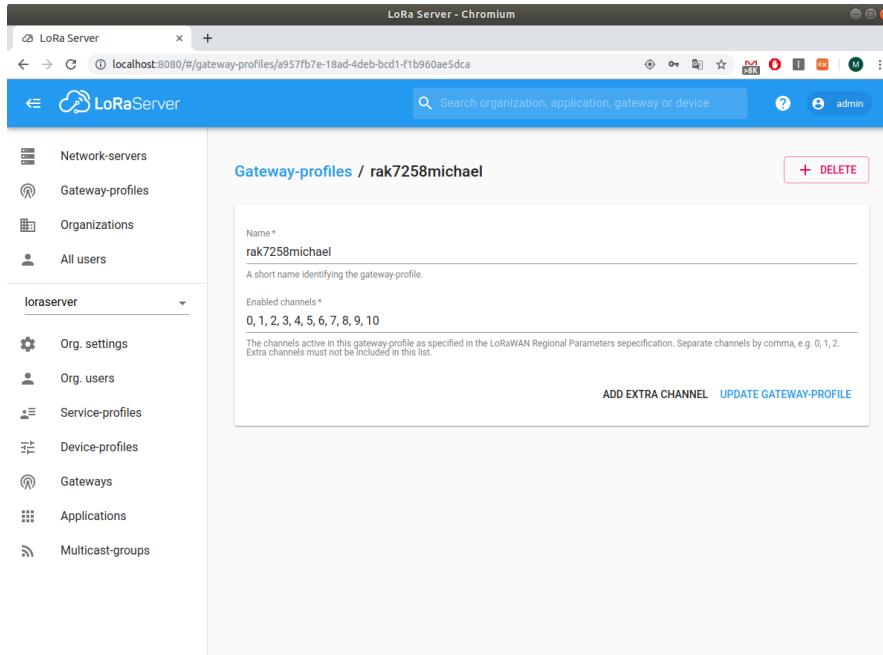
<https://www.loraserver.io/guides/first-gateway-device/>

참고: Open source LoRaServer를 설치하는 방법과 관련해서는 8장(LoRaServer Project)에서 별도로 다룬 것이지만, 그 방법이 궁금한 분들은 아래 site를 면밀히 살펴 보기 바란다. 실제로 아래 테스트를 진행하기 위해서는 LoRaServer를 미리 설치해 두어야 한다.

<https://www.loraserver.io/guides/debian-ubuntu/>



[그림 5.25] LoRa App Server - Network-Servers 등록 화면



[그림 5.26] LoRa App Server - Gateway Profiles 등록 화면

LoRa Server - Chromium

localhost:8080/#/organizations/1/service-profiles/1a356413-cd48-4ba0-a6e5-fdc74d54e7f7

Service-profiles / michael-profile1

Service-profile name *
michael-profile1

A name to identify the service-profile.

Add gateway meta-data
GW metadata (RSSI, SNR, GW geoloc., etc.) are added to the packet sent to the application-server.

Enable network geolocation
When enabled, the network-server will try to resolve the location of the devices under this service-profile. Please note that you need to have gateways supporting the fine-timestamp feature and that the network-server needs to be configured in order to provide geolocation support.

Device-status request frequency
0

Frequency to initiate an End-Device status request (request/day). Set to 0 to disable.

Minimum allowed data-rate *
0

Minimum allowed data rate. Used for ADR.

Maximum allowed data-rate *
0

Maximum allowed data rate. Used for ADR.

[그림 5.27] LoRa App Server - Service Profiles 등록 화면

LoRa Server - Chromium

localhost:8080/#/organizations/1/gateways/60c5a8fffe76130c

Gateways / rak7258michael

GATEWAY DETAILS **GATEWAY CONFIGURATION** **GATEWAY DISCOVERY** **LIVE LORAWAN FRAMES**

Gateway details

Gateway ID
60c5a8fffe76130c

Altitude
3 meters

GPS coordinates
37.5371163, 127.0078127

Last seen
a few seconds ago

Frames received

localhost:8080/#/organizations/1/gateways/60c5a8ff... |

[그림 5.28] LoRa App Server - Gateways 등록 화면

The screenshot shows the LoRa Server interface for managing device profiles. On the left, a sidebar lists various management categories like Network-servers, Gateway-profiles, Organizations, and users. The main content area is titled "Device-profiles / rak7200". It contains several tabs: GENERAL (selected), JOIN (OTAA / ABP), CLASS-B, CLASS-C, and CODEC. Under the GENERAL tab, there are fields for "Device-profile name" (set to "rak7200"), "LoRaWAN MAC version" (set to "1.0.2"), "LoRaWAN Regional Parameters revision" (set to "B"), and "Max EIRP" (set to "0"). At the bottom right of the form is a blue "UPDATE DEVICE-PROFILE" button.

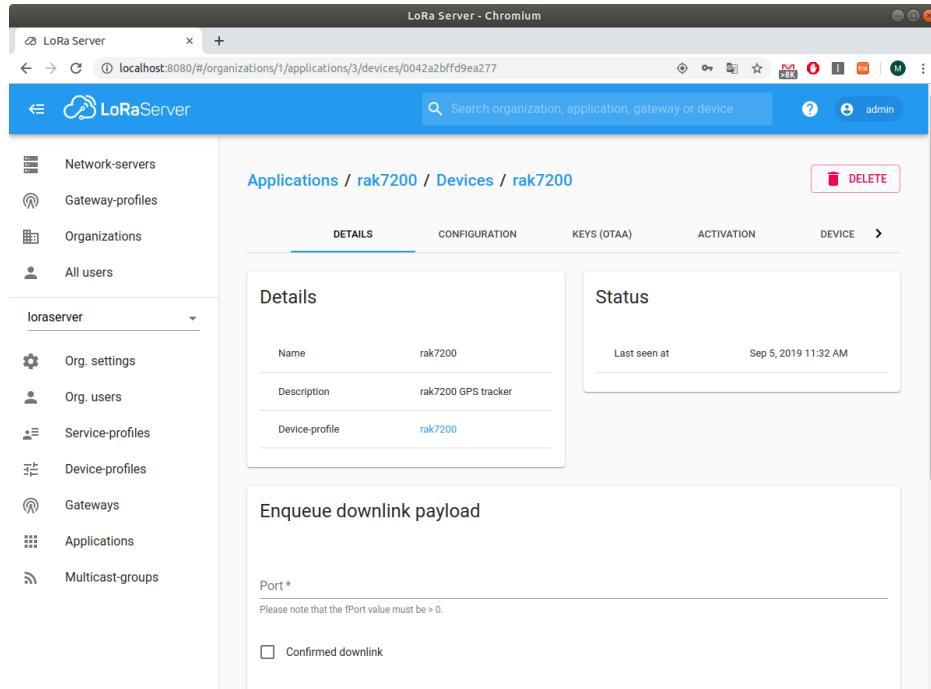
[그림 5.29] LoRa App Server - Device Profiles 등록 화면

정상적으로 설정했다면, Gateway로 패킷이 들어오는 것을 볼 수 있다.

The screenshot shows the LoRa Server interface for monitoring gateway activity. The left sidebar is identical to the previous screenshot. The main content area is titled "Gateways / rak7258michael". It features four tabs: GATEWAY DETAILS, GATEWAY CONFIGURATION, GATEWAY DISCOVERY, and LIVE LORAWAN FRAMES (which is selected). Below the tabs are four buttons: HELP, PAUSE, DOWNLOAD, and CLEAR. The main table lists received frames with columns for Type, Time, and Content. The table contains the following data:

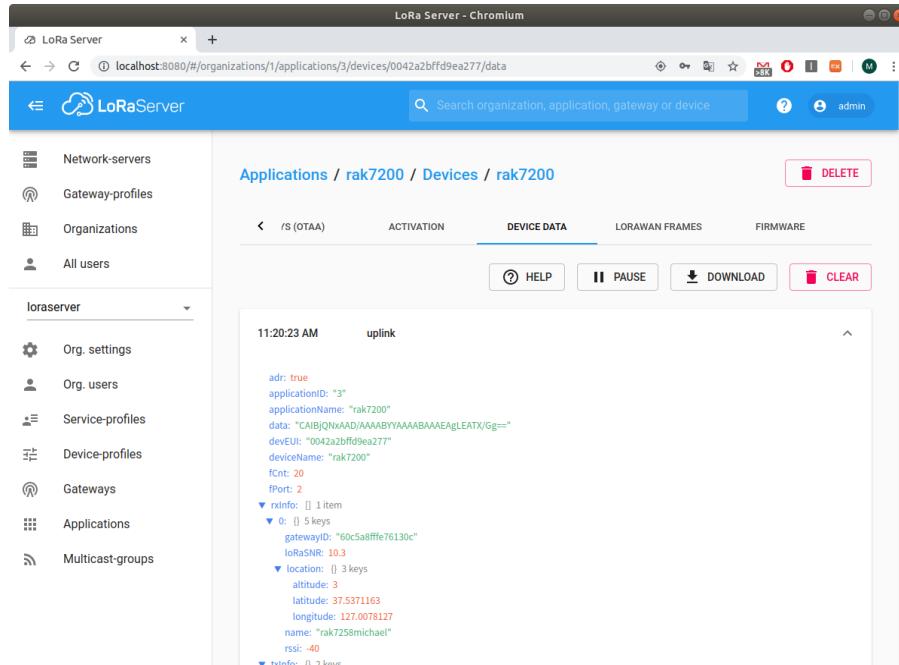
DOWNLINK	11:15:49 AM	UnconfirmedDataDown	01fda2a9
UPLINK	11:15:49 AM	UnconfirmedDataUp	01fda2a9
UPLINK	11:15:43 AM	UnconfirmedDataUp	26012a02
UPLINK	11:15:38 AM	JoinRequest	00ebe388cb1094e4

[그림 5.30] LoRa App Server - Gateway Live LoRaWAN Frames 확인 화면



[그림 5.31] LoRa App Server - rak7200 device 등록 화면

마침내, Device(RAK7200 GPS tracker)로 부터 uplink packet이 올라옴을 확인할 수 있다.



[그림 5.32] LoRa App Server - Device Data 수신 화면

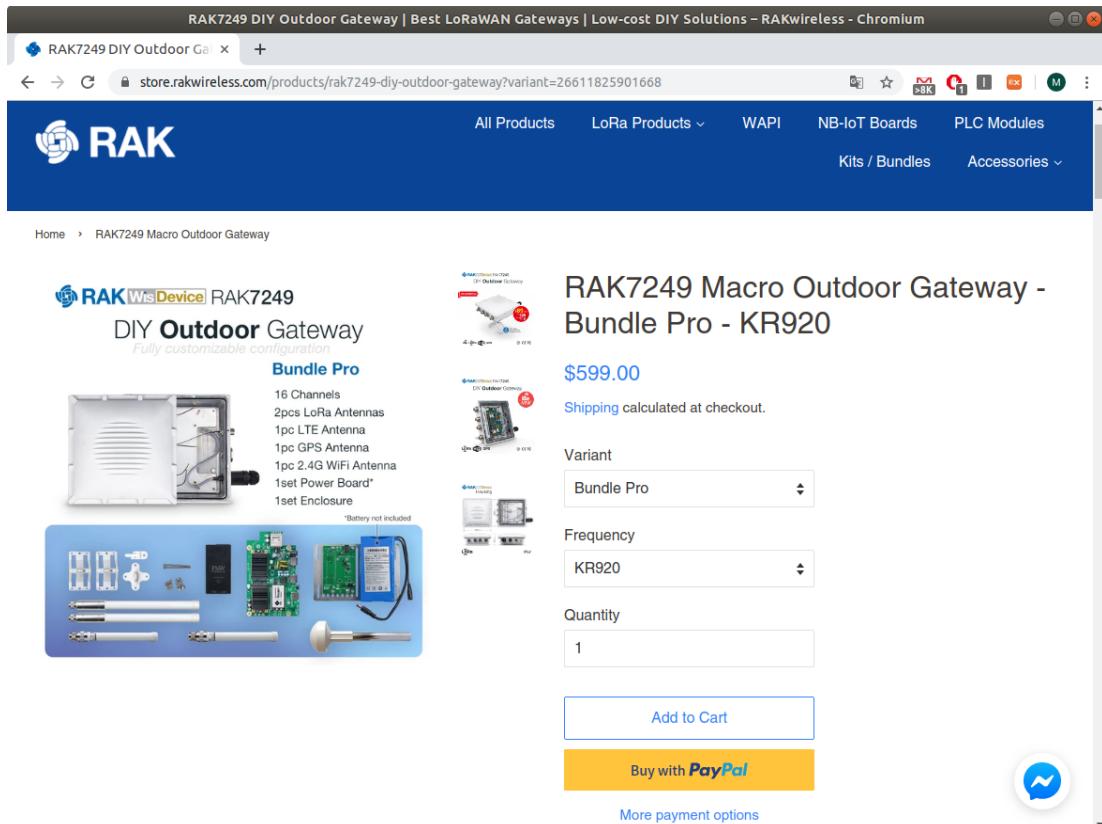
Type	Time	Data	...
UPLINK	11:13:42 AM	UnconfirmedDataUp 26012e6e	▼
UPLINK	11:13:37 AM	ConfirmedDataUp 1b0f12dd	▼
UPLINK	11:13:30 AM	ConfirmedDataUp 1b0511ee	▼
UPLINK	11:13:01 AM	UnconfirmedDataUp 1a000cf0	▼
UPLINK	11:12:52 AM	JoinRequest a84041000181943f	▼
UPLINK	11:12:51 AM	UnconfirmedDataUp 1a000cf0	▼
UPLINK	11:12:46 AM	UnconfirmedDataUp 01fda2a9	▼
UPLINK	11:12:38 AM	UnconfirmedDataUp 26012e6e	▼
UPLINK	11:12:33 AM	JoinRequest 00ebe388cb1094e4	▼
UPLINK	11:12:01 AM	JoinRequest d02544fffff110a2	▼

[그림 5.33] LoRa App Server - Device Live LoRaWAN Frames 확인 화면

참고: LoRa App Server WebUI 설정 관련하여 분명히 모호한 내용이 있을 수 있다. 크게 개의치 말고 이것 저것 설정해 보기 바란다.

6. RAKWireless RAK7249 Outdoor LoRa Gateway

대표님이 구매를 승인하셨다. 미친 척하고 최고로 비싼 놈(\$599)으로 주문했다. 네네~ 압니다.



[그림 6.1] RAK7249 Outdoor LoRa Gateway(1)



[그림 6.2] RAK7249 Outdoor LoRa Gateway(2)

1) RAK7249 구성물 소개



[그림 6.3] RAK7249 Outdoor LoRa Gateway(3) - 외관

- **Main Board**
- **Enclosure**
- **Backup Battery**
- **Accessories**



[그림 6.4] RAK7249 Outdoor LoRa Gateway(4) - 구성 요소

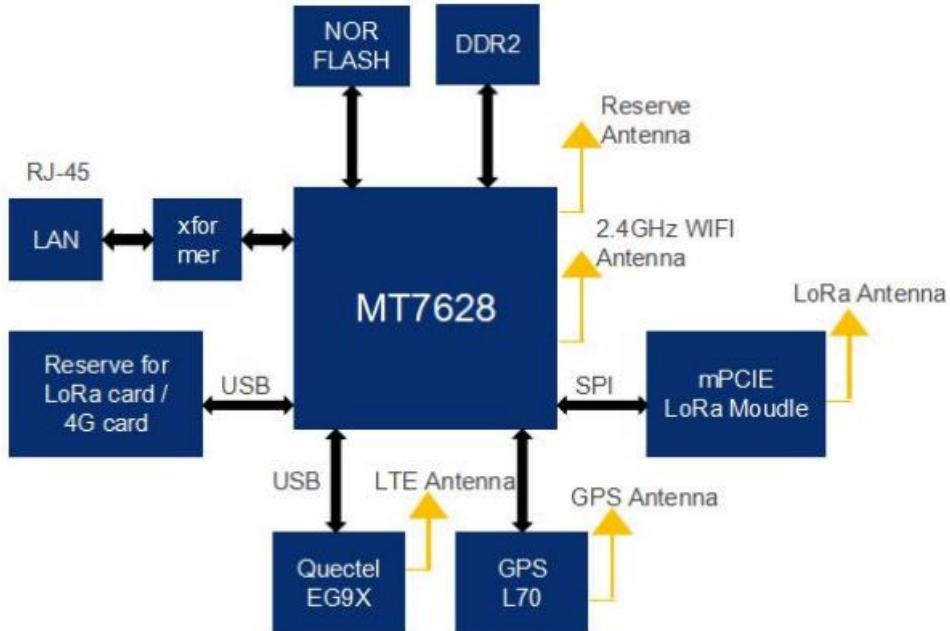
Key Features

Computing	• MT7628, DDR2RAM 128MB
WIFI Feature	<ul style="list-style-type: none"> • Frequency: 2.400 - 2.4835GHz • (802.11b/g/n) • RX Sensitivity: -95dBm (Min) • TX Power: 20dBm (Max)
LoRa Feature	<ul style="list-style-type: none"> • SX1301 Mini PCIe card • 8 Channels (Optional: 16 channels) • TX Power: 27dBm (Max) • RX Sensitivity: -142dBm (Min)
Cellular	<ul style="list-style-type: none"> • EG95: LTE CAT 4 • Variant for Europe, LTE FDD: B1/B3/B7/B8/B20/B28A; WCDMA: B1/B8; GSM: 900/1800MHz • Variant for North America LTE FDD: B2/B4/B5/B12/B13; WCDMA: B2/B4/B5
Power Supply	<ul style="list-style-type: none"> • POE (IEEE 802.3af), 42~57VDC
Power Consumption	<ul style="list-style-type: none"> • 12W (typical)
ETH	<ul style="list-style-type: none"> • RJ45 (10/100M)
Antenna	<ul style="list-style-type: none"> • 5 N-Type connectors
Ingress Protection	<ul style="list-style-type: none"> • IP67
Enclosure Material	<ul style="list-style-type: none"> • Aluminum
Weight	<ul style="list-style-type: none"> • Approximately 111.11oz (3.15kg with mounting kit)
Dimension	<ul style="list-style-type: none"> • 220mm x 220mm x 104mm
Operating Temp.	<ul style="list-style-type: none"> • -30 to 65 °C
Installation Method	<ul style="list-style-type: none"> • Pole or Wall mounting

RF Specifications

Wireless Standard	<ul style="list-style-type: none"> • IEEE 802.11b/g/n
Wi-Fi Operating Frequency	<ul style="list-style-type: none"> • ISM band: 2.412~2.472(GHz)
Wi-Fi Operation Channels	<ul style="list-style-type: none"> • 2.4GHz: 1-13
Wi-Fi Transmit Power	<ul style="list-style-type: none"> • 802.11b (The maximum power may be different depending on local regulations) 19dBm@ 1Mbps • 802.11g 18dBm@ 6Mbps 16dBm@ 54Mbps • 802.11n(2.4G) 18dBm@MCS0 (HT20) 16dBm@MCS7 (HT20) 17dBm@MCS0 (HT40) 15dBm@MCS7 (HT40)
Wi-Fi Receiver Sensitivity (Typical)	<ul style="list-style-type: none"> • 802.11b -95dBm@ 1Mbps • 802.11g -90dBm @6 Mbps -75dBm@54Mbps • 802.11n(2.4G) -89dBm@MCS0 (HT20) -72dBm @MCS7(HT20) -86dBm @MCS0(HT40) -68dBm @MCS7(HT40)
LoRa Operating Frequency	<ul style="list-style-type: none"> • EU433 / CN470 / EU868 / US915 / AU915 / AS923 / AS920 / KR920 / IN865
LoRa TX Power	<ul style="list-style-type: none"> • 27 dBm (Max)
LoRa RX Sensitivity	<ul style="list-style-type: none"> • -142 dBm (Min)

[그림 6.5] RAK7249 Outdoor LoRa Gateway(5) - key feature & spec



[그림 6.6] RAK7249 Board block diagram



[그림 6.7] RAK7249 PCB

<관련 문서>

<https://downloads.rakwireless.com/en/LoRa/DIY-Gateway-RAK7249/>

LoRa Gateway(기지국)는 대개의 경우 옥외(Outdoor)에 설치해야 한다. 따라서 지금부터는 RAK7249를 제대로 검토해 보기로 하겠다.

2) RAK7249 켜기 |

아래 문서 내용을 보면 알 수 있듯이, 전원을 넣기 전에 Antenna를 먼저 연결하자.

Industrial Outdoor LoRa Gateway RAK7249 Quick Start Guide

Step 1: Attach the antennas

First and foremost screw on the antennas. All 5 of them should be installed (WiFi, LoRa, LTE on the top, and GPS on the bottom).

Note: Do not power the device if the antenna port has been left open (not connected to the antenna).

Step 2: Power on the Gateway

It is recommended to use a CAT5 cable to provide power to the Gateway. Attach one end to the PoE injector and the other to the Ethernet port on the bottom of the casing.

RAK7249 package에 포함되어 있는 안테나는 LoRa 2개, LTE 1개, Wi-Fi 1개, GPS 1개로 총 5개나 된다. 그 중, LoRa 안테나(Fiber Glass type)의 스펙을 정리해 보면 다음과 같다.

- Frequency Range: 860~930MHz
- One antenna to suit both 868MHz and 915 MHz bands
- Max Gain: 3dBi
- High efficiency
- Vertically polarized monopole



The figure consists of two parts. On the left is a table titled 'RAK7249 LoRa Antenna Spec' with 12 rows of technical parameters. On the right is a photograph of the RAK7249 LoRa antenna, which is a vertical cylindrical device with a clear radome cover and a metallic connector at the bottom.

Frequency Range (MHz)	860 - 930
Gain (dBi)	2.6 - 3.1
VSWR	≤ 2.5
Efficiency	60%
Radiation	360°
Impedance (Ohms)	50Ω
Polarization	Vertical
Radome Body	Fiber glass
Connector	N-Type Male
Dimensions (mm)	Φ 25 x L360±10mm
Operation Temp (°C)	-20 ~ +65
Storage Temp (°C)	-30 ~ +75

[그림 6.8] RAK7249 LoRa Antenna Spec

RAK7249 accessory(안테나, Battery, PoE 등) 관련 자세한 정보(spec)는 아래 site에서 확인 가능하다.

<https://downloads.rakwireless.com/en/LoRa/DIY-Gateway-RAK7249/Accessory/>

다음으로 PoE Injector를 사용하여 UTP cable을 통해 전원을 인가해 보자. PoE 사용법은 아래 그림에 잘 표시되어 있다.

[RAK7249 LoRa Gateway] --- UTP cable --- [POE port] [LAN port] ---- L2 Switch or 공유기

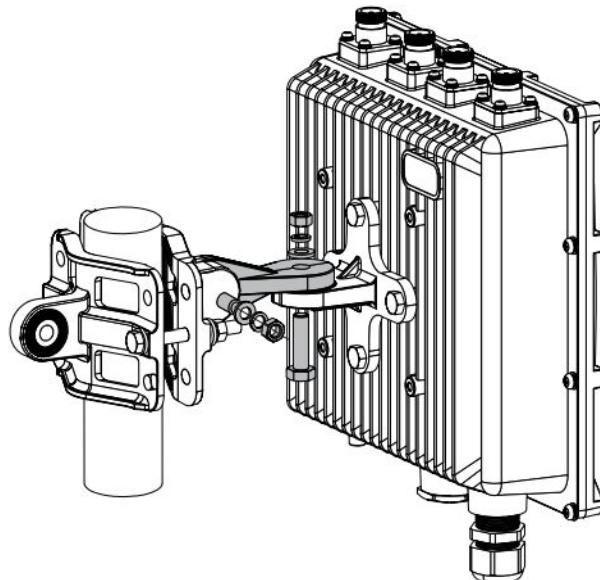




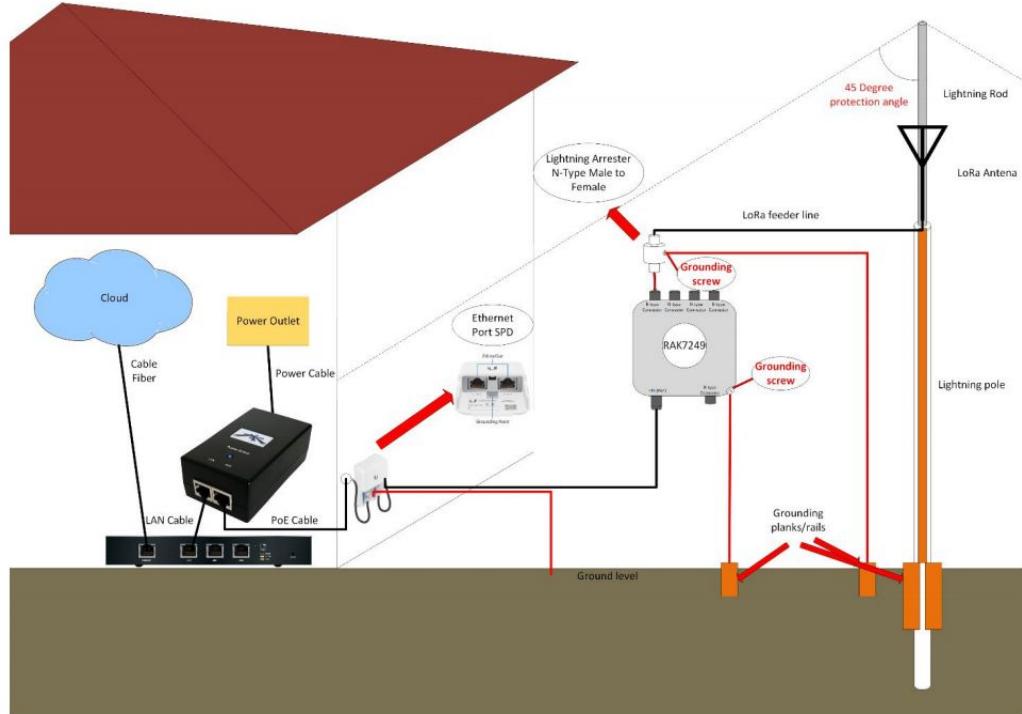
[그림 6.9] PoE Injector로 power 연결하기

3) RAK7249 외부에 설치하기

<TBD> 나중에 해야 할 일이지만, 중요한 부분이라서 미리 정리해 둔다.



[그림 6.10] RAK7249 옥외 설치 준비



[그림 6.11] RAK7249 옥외 설치 모습 - 주의 사항(피뢰침 설치해야 함)

1. Lightning arrestor for the LoRa, LTE, and Wi-Fi antennas



RAKwireless recommends using a high quality Coax Gas discharge suppress such as the [AIR802](#). It should have an N plug (Male) to N jack (Female) bulkhead connector and be operational in the 0 to 6GHz RF range.

[그림 6.12] AIR802 Lightning arrestor(피뢰침)

자세한 사항은 아래 문서를 참조하도록 하자.

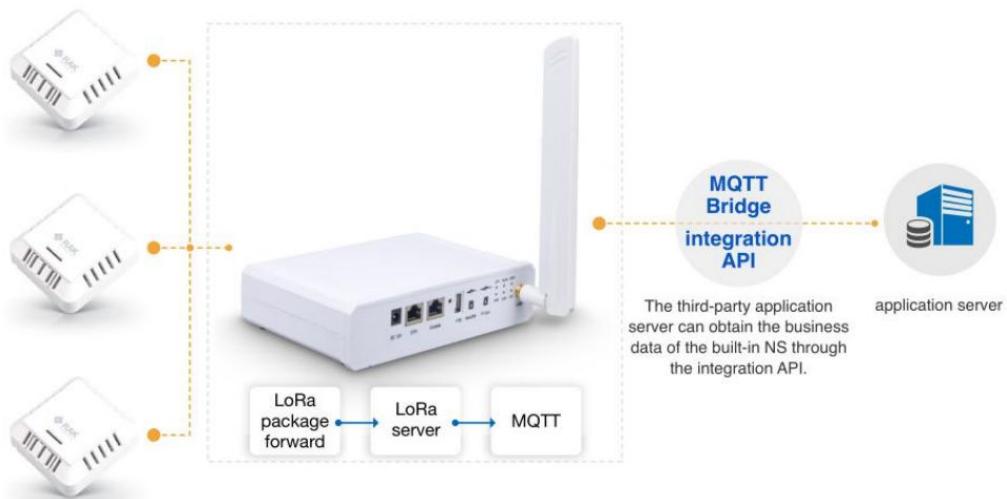
https://downloads.rakwireless.com/en/LoRa/DIY-Gateway-RAK7249/Application-Notes/Lighting_Protection_Manual_V1.3.pdf

4) RAK7204 Node 소개

RAK7204는 온도(temperature), 습도(humidity), 가스 압력(gas pressure), 실내 공기 질(IAQ: Indoor Air Quality)을 측정(환경 센서임)하여 LoRa Gateway로 전달하는 device(Node)로 아래 그림과 같이 생겼다.



[그림 6.13] RAK7204 Node



[그림 6.14] RAK7204 Node + RAK7259 + External Application Server 구성도

<주의 사항>

"The included battery is non rechargeable. In case you need to plug in the Micro USB port for the purpose of configuring the node you should remove the battery. There is no protection circuitry and leaving the battery inside the device when connected via the Micro USB interface will damage it and might result in the device overheating and catching fire. Please make sure you remove the battery immediately after you open the casing, before performing any other configuration or maintenance."

RAK7204 LoRa Node 설정을 위해 micro USB port를 연결한 경우에는 battery를 연결해서는 안된다. 과부하가 되어 불이 날 수도 있단다. 헐~

아래 내용은 RAK7204에 포함되어 있는 주요 센서의 스펙 정보이다. 한번씩 둘러 보기 바란다.

Temperature Sensor Specifications

Parameter	Min.	Typical.	Max.
Temperature Range	-40 °C	+25 °C	+85 °C
Accuracy			0.5 °C
Output Resolution			0.01 °C

Humidity Sensor Specifications

Parameter	Min.	Typical.	Max.
Humidity Range	0% r.H.		100 % r.H.
Accuracy			+3% r.H.
Output Resolution			0.008% r.H.

Gas Pressure Sensor Specifications

Parameter	Min.	Typical.	Max.
Range	300 hPa		300 hPa
Accuracy		+/-0.6 hPa	
Output Resolution		0.18 Pa.	

IAQ Sensor Specifications

Parameter	Min.	Typical.	Max.
IAQ Range	0.		500.
Accuracy		15.	
Output Resolution		1	

<STM32CubeProgrammer tool 설치>

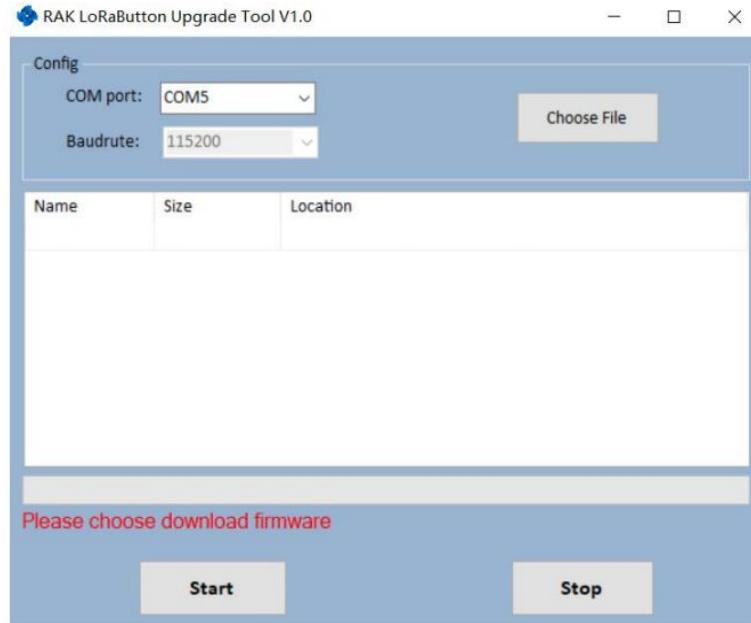
https://www.st.com/content/st_com/en/products/development-tools/software-development-tools/stm32-software-development-tools/stm32-programmers/stm32cubeprog.html#overview

[TBD] 이걸 설치하여 bootloader를 RAK7204 board에 올릴 수 있도록 해 보자. bootloader writing을 할 때는 boot mode로 바꿔주기 위해 jumper를 교체해 주어야 한다. 단, firmware version이 V3.0.0.0 이상인 경우는 이 과정을 수행할 필요 없음. 필요없겠군~

<RAK Upgrade Tool 설치>

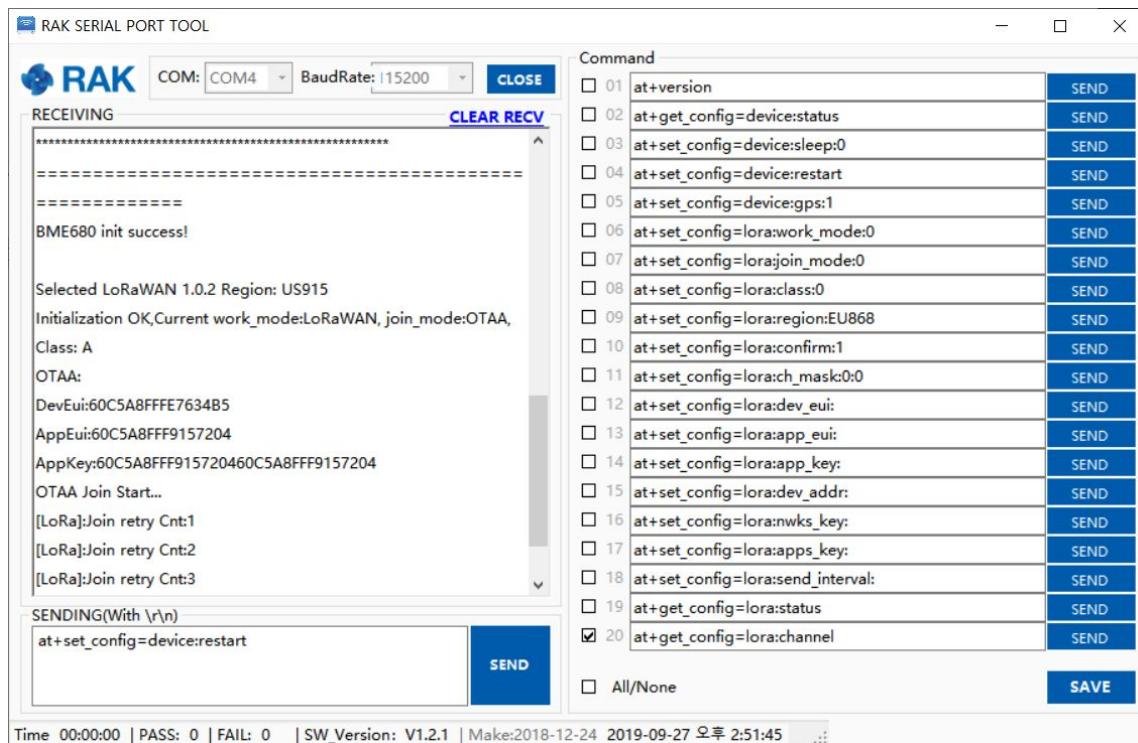
RAK7204 firmware를 writing하고자 한다면 아래 tool을 설치해서 사용해야 한다.

https://downloads.rakwireless.com/en/LoRa/RAK612-LoRaButton/Tools/RAK_LoraButton_Upgrade_Tool_V1.0.zip



[그림 6.15] RAK Firmware Upgrade Tool 화면

<UART debugging tool로 연결하기>



[그림 6.16] RAK Serial Port Tool 화면

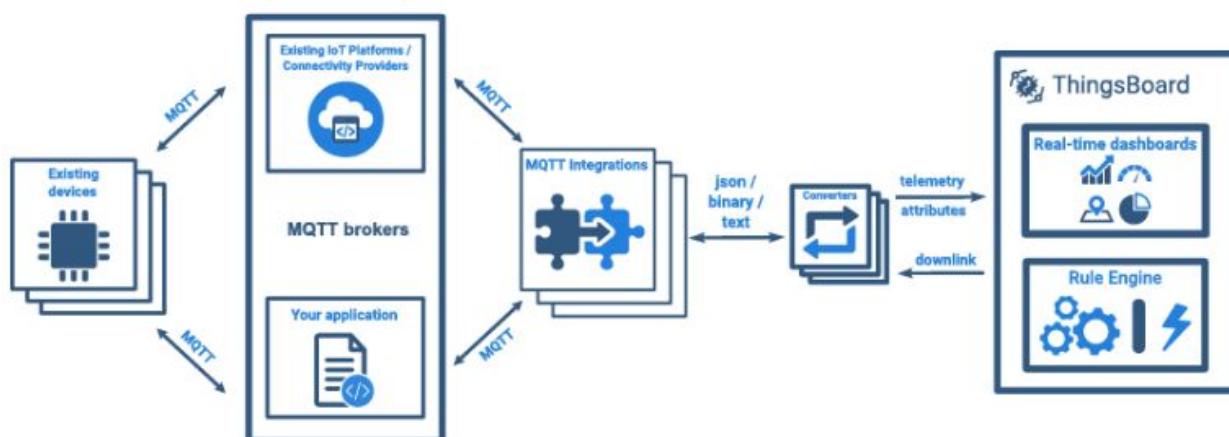
5) RAK7204 + RAK7249 Built-in LoRa Server + ThingsBoard(CE) 연동하기

이 절에서는 RAK7249의 Built-in LoRa Server를 켜 둔 상태에서 MQTT broker를 통해 LoRaServer(<https://loraserver.io>) 없이 direct로 ThingsBoard와 연동 가능한지를 탐진해 보기로 한다.

https://downloads.rakwireless.com/en/LoRa/DIY-Gateway-RAK7249/Application-Notes/LoRaWAN_Deployment_Scenario1_V1.0.pdf

결론 부터 얘기하자면, **MQTT를 통해 RAK7249 Built-in Server와 ThingsBoard CE를 연동하는 것은 문제가 있다.** MQTT integration 방법은 아래 site에 잘 설명이 되어 있는데, 불행히도 PE(Professional Edition)에서만 지원된다. **흠 그럼 그렇지~ PE가 왜 있나 했더니 ... 껌**

<https://thingsboard.io/docs/user-guide/integrations/mqtt/>



[그림 6.17] ThingsBoard MQTT Integration

그렇다고, 이대로 포기할 수는 없다. 이 없으면 잇몸으로라도 씹어 먹어야지 않겠는가 ~ ㅎㅎ

RAK7249 Built-in LoRa Server ⇒ MQTT(client) ⇒ **MQTT_HTTP Converter**(구현하자) ⇒ HTTP(json format) ⇒ ThingsBoard

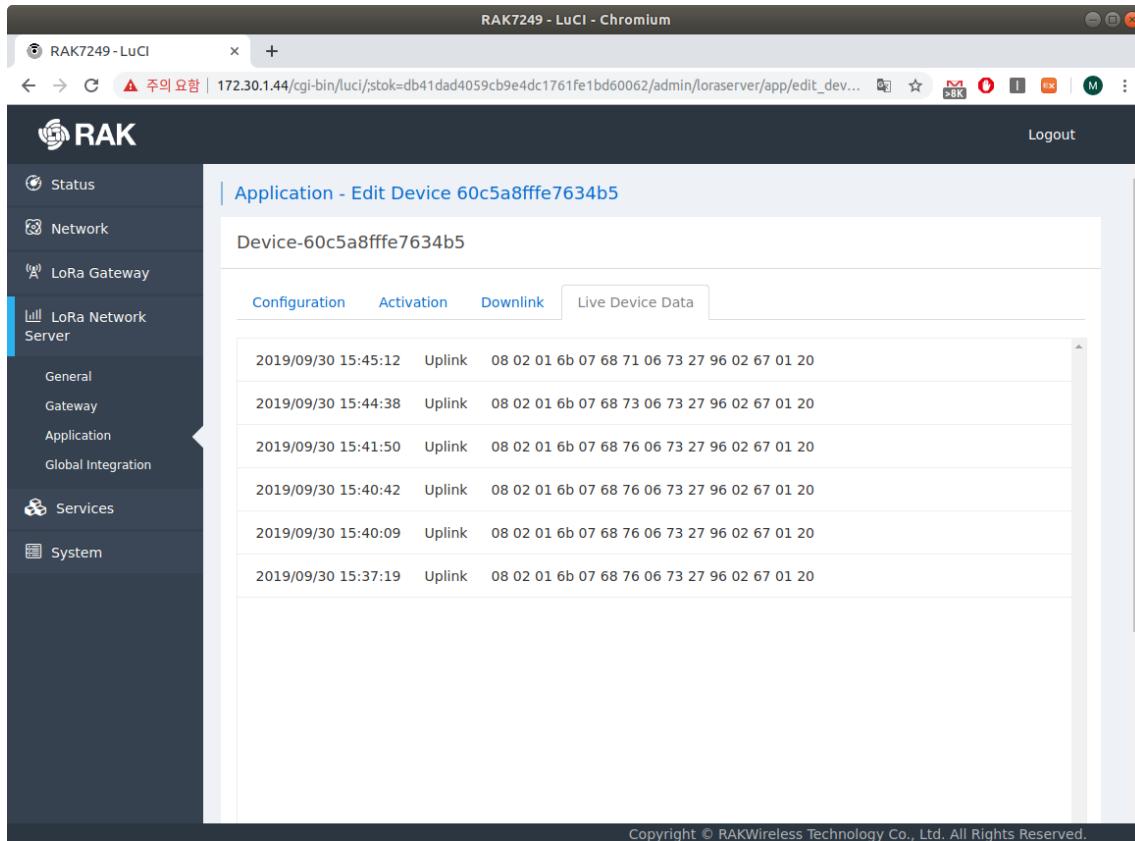
참고: **MQTT_HTTP Converter**는 base64 encoding된 data를 받아서 base64 decoding한 후, 다시 json format으로 바꾸어 ThingsBoard로 전달하는 놈이다.

<LoRa Server가 설치된 desktop PC>

```
$ sudo mosquitto_sub -v -t "application/#" -h localhost -p 1883
```

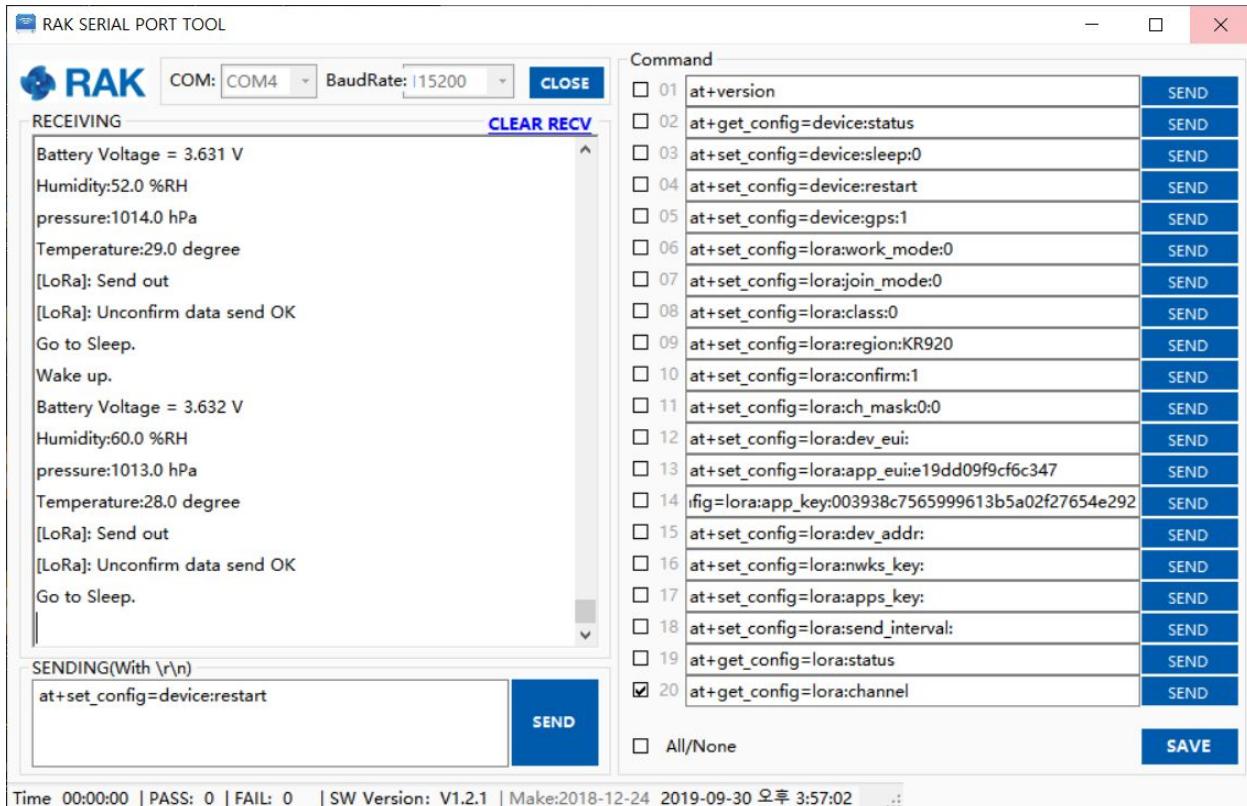
```
application/1/device/60c5a8ffe7634b5/rx
{"applicationID": "1", "applicationName": "rak7204_sensor", "devEUI": "60c5a8ffe7634b5", "deviceName": "dev-60c5a8ffe7634b5", "timestamp": 1569825709, "fCnt": 467, "fPort": 3, "adr": true, "data": "CAIBawdodwZzJ5YCZwEf", "data_encode": "base64", "rxInfo": [{"gatewayID": "60c5a8ffe7614c3"}, {"IoRaSNR": 10.8, "rssI": -36, "frequency": 922500000, "location": {"latitude": 0.000000, "longitude": 0.000000, "altitude": 0}}], "txInfo": {"frequency": 922500000, "dr": 5}}
```

아래 그림의 Live Device Data 최 우측 필드의 내용은 위에서 보이는
data(**CAIBawdodwZzJ5YCZwEf**) 필드의 내용을 base64 decoding한 후, hexa 값(**08 02 01 6b
07 68 77 06 73 27 96 02 67 01 1f**)으로 표시한 것이다.



[그림 6.18] RAK7249 Live Device Data

그렇다면 남은 문제는 위 hexa string으로 부터 어떻게 json format으로 변경하느냐 인데 ... 이를 위해서는 RAK7204 LoRa Node에서 어떤 data를 보내는지를 먼저 파악해 볼 필요가 있다. 이건 특별한 방법이 있다기보다는 **LoRa Node에서 보내는 내용(문서를 참조해야함)을 보고 상황에 맞게 json 형태로 변경해 주는 수 밖에 없을 듯하다.**



[그림 6.19] RAK7204 LoRa 패킷 전달 모습

참고: RAK7204 문서를 찾아보고 있는데, 아래 내용에 해당하는 상세 정보가 표시된 것이 없다. 즉, 아래 hexa 값 중, 어디서부터 얼마 만큼이 Battery Voltage에 해당하고, 어디서부터 얼마 만큼이 Humidity에 해당하는지 등에 관한 정보가 필요한데 ... 없다. :(

08 02 01 6b 07 68 77 06 73 27 96 02 67 01 1f

<RAK7204가 전달하는 data>

- 1) Battery Voltage: 예) 3.632V
- 2) Humidity: 60.0 %RH
- 3) Pressure: 1013.0 hPa
- 4) Temperature: 28.0 degree

<생성해야 할 json data>

```
{ "battery_volate":"3.632", "humidity":"60.0", "pressure":"1013.0", "temperature":"28.0" }
```

<TBD> mosquitto source code를 수정하여, 위의 action(즉, data 부분을 base64로 decoding한 후, 다시 json으로 변경)을 취하는 코드를 구현하자.

아래 site도 참조해 보자. **뭔가 보내는 data에 형식이 정해져 있는 것 같다. 그럼 그렇지 ...**

<https://developers.mydevices.com/cayenne/docs/lora/>

Uplink Payload Structure

1 Byte	1 Byte	N Bytes	1 Byte	1 Byte	M Bytes	...
Data1 Ch.	Data1 Type	Data1	Data2 Ch.	Data2 Type	Data2	...

- **Data Channel:** Uniquely identifies each sensor in the device across frames, eg. "indoor sensor"
- **Data Type:** Identifies the data type in the frame, eg. "temperature".

LPP_DATA_TYPE = IPSO_OBJECT_ID - 3200

[표 6.1] IPSO Alliance Smart Objects Guidelines에 따른 data type

Type	IPSO	LPP	Hex	Data Size	Data Resolution per bit
Digital Input	3200	0	0	1	1
Digital Output	3201	1	1	1	1
Analog Input	3202	2	2	2	0.01 Signed
Analog Output	3203	3	3	2	0.01 Signed
Illuminance Sensor	3301	101	65	2	1 Lux Unsigned MSB
Presence Sensor	3302	102	66	1	1
Temperature Sensor	3303	103	67	2	0.1 °C Signed MSB
Humidity Sensor	3304	104	68	1	0.5 % Unsigned
Accelerometer	3313	113	71	6	0.001 G Signed MSB per axis
Barometer	3315	115	73	2	0.1 hPa Unsigned MSB
Gyrometer	3334	134	86	6	0.01 °/s Signed MSB per axis
GPS Location	3336	136	88	9	Latitude : 0.0001 ° Signed MSB
					Longitude : 0.0001 ° Signed MSB
					Altitude : 0.01 meter Signed MSB

Cayenne LPP(Low Power Payload) source code도 존재한다.

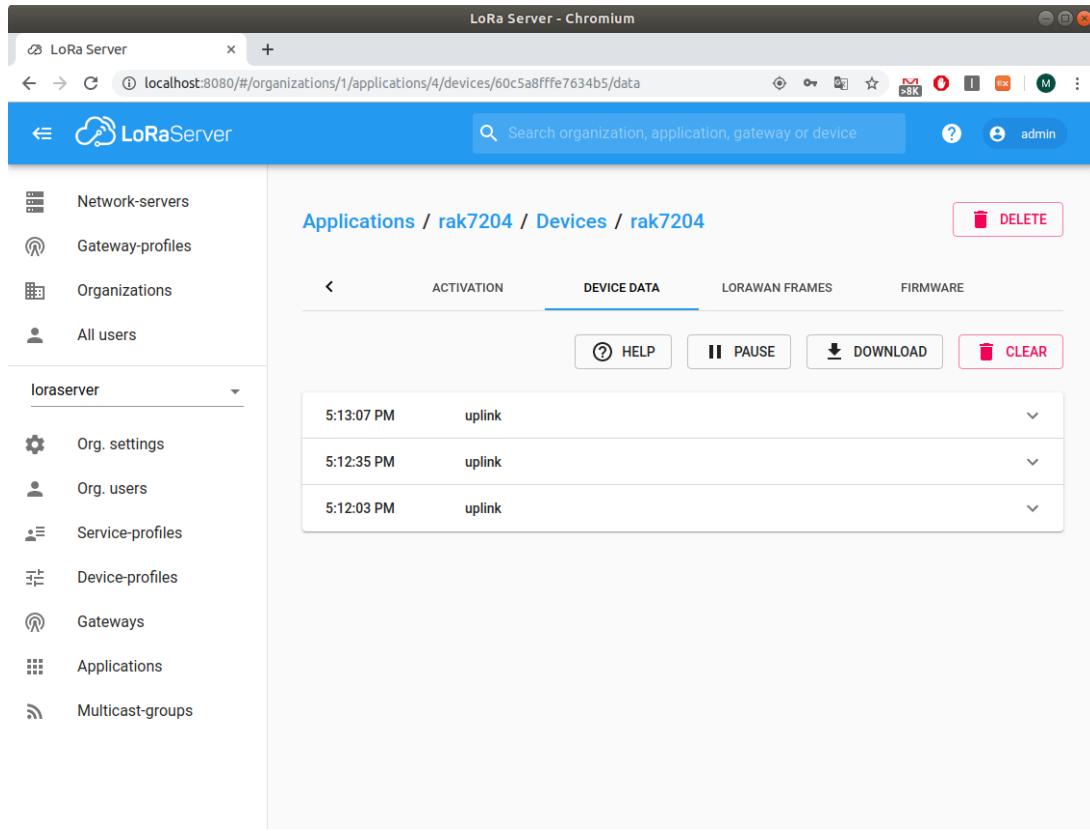
<https://github.com/myDevicesIoT/CayenneLPP>

6) RAK7204 + RAK7249 + External LoRa Server + ThingsBoard(CE) 연동하기

지금 부터는 아래 문서를 참조하여 RAK7204를 LoRa Server에 연동시켜 보도록 하겠다. (아래 문서에는 없지만) 이후 LoRa Server를 다시 ThingsBoard에 연결시켜 보도록 하자.

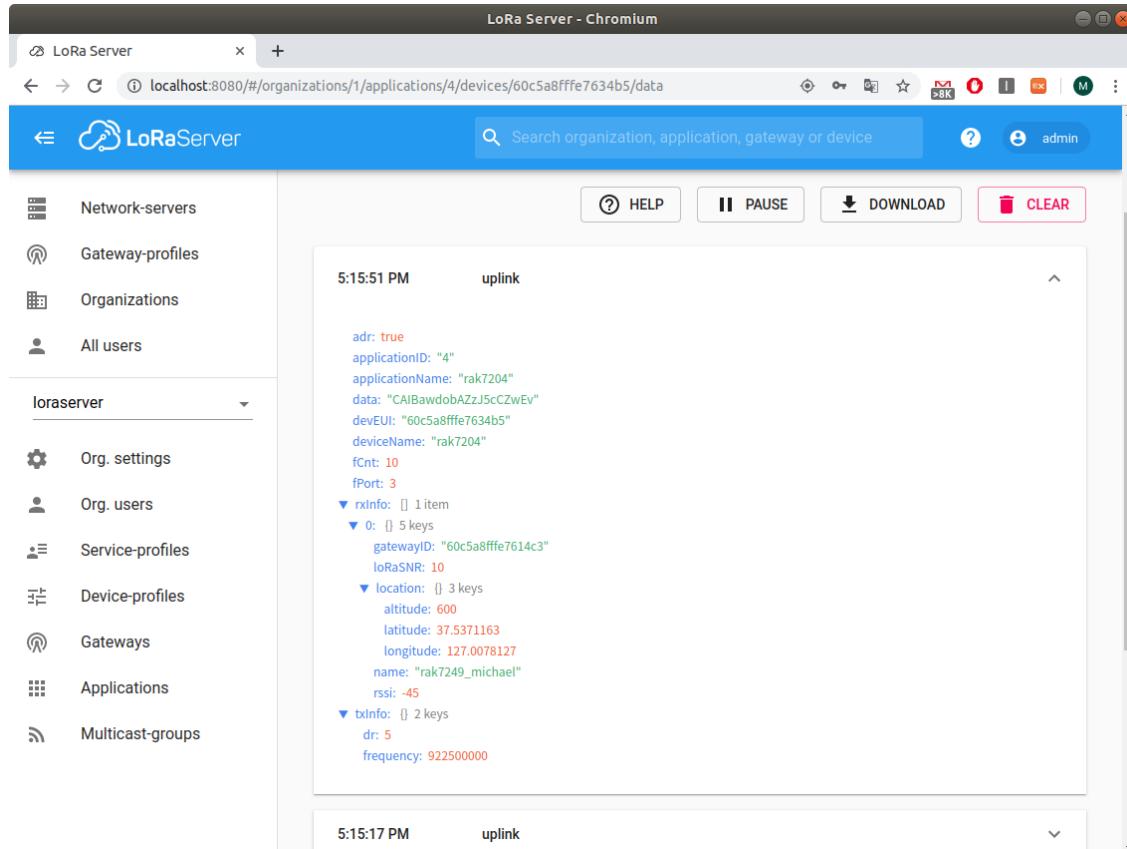
참고: 이 장의 내용은 11장의 내용을 정리한 후, 작성한 것이다. 따라서 ThingsBoard 관련 상세한 내용은 11장을 참조하면 된다.

https://downloads.rakwireless.com/en/LoRa/RAK7204/Application_Notes/Get_Start_with_RAK7204.pdf



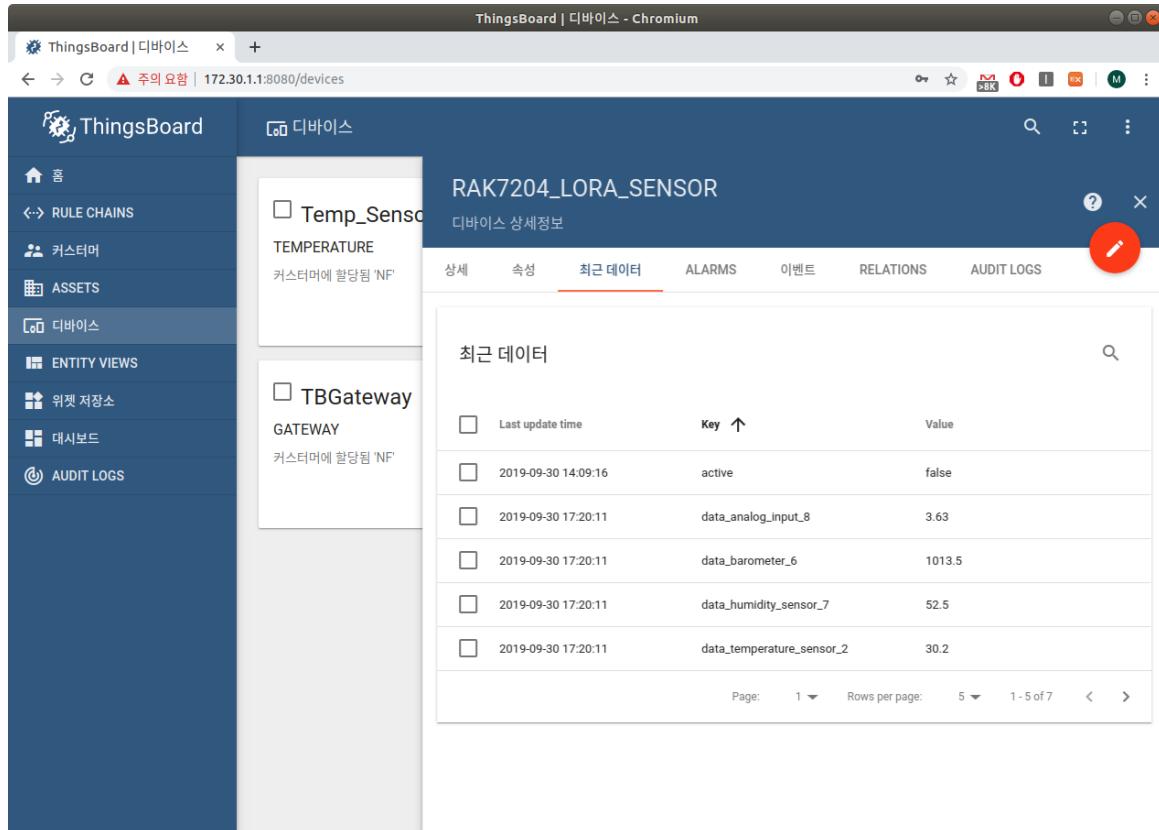
[그림 6.20] LoRa App Server - RAK7204 LoRa 패킷 전달 모습(1)

참고: LoRa Server 설정과 관련해서는 5장을 참조하기 바란다.



[그림 6.21] LoRa App Server - RAK7204 LoRa 패킷 전달 모습(2)

RAK7204 ⇒ RAK7249로 부터 LoRa Server ⇒ LoRa App Server로 패킷이 정상적으로 들어온다. 따라서 지금부터는 LoRa App Server와 ThingsBoard를 연결해 보도록 하겠다.



[그림 6.22] ThingsBoard - RAK7204 LoRa 패킷 전달 모습

Payload Codec을 Cayenne LPP로 지정할 경우, 위의 그림과 같이 정상적으로 data가 구분(json format이라는 얘기)되어 ThingsBoard로 전달되는 것을 알 수 있다. 신기한 일이다 :)

<ThingsBoard 디바이스 - 최근 데이터에 표시되는 내용>

```

data_analog_input_8 : 3.63
  → Battery voltage
data_barometer_6 : 1013.5
  → 압력(pressure)
data_humidity_sensor_7 : 53.5
  → 습도
data_temperature_sensor_2 : 30.3
  → 온도
  
```

The screenshot shows the ThingsBoard web interface. On the left, there's a sidebar with navigation links: Home, RULE CHAINS, Customer, ASSETS, Devices, ENTITY VIEWS, Widgets, Dashboards, and AUDIT LOGS. The main area is titled 'RAK7204_LORA_SENSOR' and 'Device Status'. It shows two entities: 'Temp_Sensor' (TEMPERATURE) and 'TBGateway' (GATEWAY). The 'Recent Data' tab is active, showing a table with the following data:

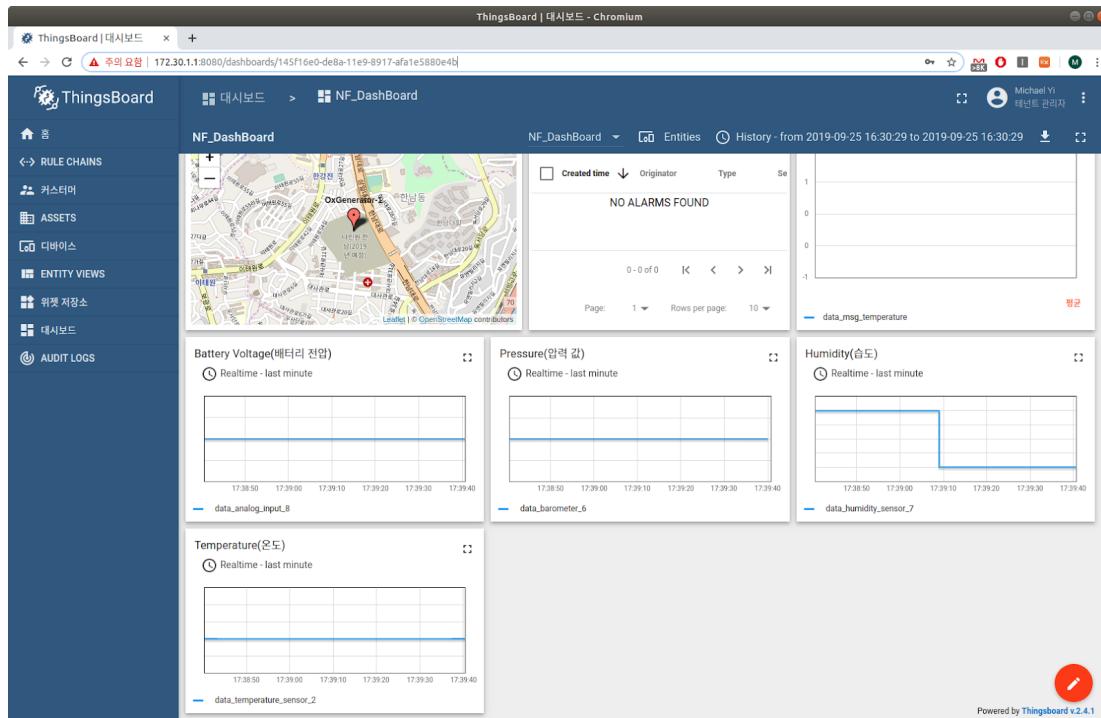
	Key ↑	Value
<input checked="" type="checkbox"/> Last update time		
<input checked="" type="checkbox"/> 2019-09-30 14:09:16	active	false
<input checked="" type="checkbox"/> 2019-09-30 17:29:23	data_analog_input_8	3.64
<input checked="" type="checkbox"/> 2019-09-30 17:29:23	data_barometer_6	1013.5
<input checked="" type="checkbox"/> 2019-09-30 17:29:23	data_humidity_sensor_7	53
<input checked="" type="checkbox"/> 2019-09-30 17:29:23	data_temperature_sensor_2	30.2

At the bottom right of the data table, there's a red circle containing a white pencil icon, which likely represents an edit or configuration option.

[그림 6.23] ThingsBoard - RAK7204 LoRa 패킷(최신 데이터)로 위젯 만들기(1)

주의: 위의 그림에서는 전체 항목(row)을 선택했으나, 실제로는 각각 선택 후, "위젯 보기" 버튼을 눌러서 각각(배터리 전압, 습도, 압력, 온도)의 위젯을 생성해 주어야 한다.

아래 그림은 4개의 Chart를 이용해 RAK7204의 센서 정보(배터리 전압, 습도, 압력, 온도)를 실시간으로 표현해 주는 것을 capture한 화면이다.



[그림 6.24] ThingsBoard - RAK7204 LoRa 패킷(최신 데이터)로 위젯 만들기(2)

7. MatchX MX1702 LoRa Gateway(LBT 지원 모델)

드디어 기다리던 녀석이 도착(10/04/2019)했다. 이번 장에서는 LBT를 지원하는 MatchX LoRa Gateway(Indoor 및 Outdoor 용으로 활용 가능)를 검토해 보기로 하겠다.



[그림 7.1] MatchX MX1702 LoRa Gateway

참고: MatchX MX1702는 탁 트인 야외에서 최대 20Km 이상을 cover할 수 있으며, 최대 65535개의 node를 수용할 수 있다고 한다. 과연 진짜 그럴까?



[그림 7.2] MatchX MX1702 LoRa Gateway Package 내용물

Description

The MatchX LPWAN Gateway is a feature rich LoRaWAN gateway with models certified around the globe. With Listen-Before-Talk technology (available in Europe, Japan and South Korea), MatchX Gateways are built to avoid data-collision, ensuring the reliable transmission of your data.

Focus on Energy Efficiency

Each MatchX Gateway is as energy efficient as the average lightbulb. Ensuring a low cost of operation, and reducing your carbon footprint.

Securely Encrypted

End-to-end encryption increases the security of your data. MatchX Gateways relay information, however are unable to decrypt it themselves.

Choose the device you need for your country

Device	Channel Plan	Supporting Countries
MX1701	<ul style="list-style-type: none"> • EU863-870 • IN765-867 	Please contact us if you aren't certain.
MX1702	<ul style="list-style-type: none"> • US902-928 • AU915-928 • AS923 • KR 920-923 	Please contact us if you aren't certain.

[그림 7.3] MatchX MX1702 설명

참고1: MX1702는 KR920-923/LBT를 지원한다. 또한 OpenVPN이 암호 통신을 위해 기본적으로 사용된다.

참고2: MatchX는 독일 회사이다.

MX1702의 주요 스펙을 나열해 보면 다음과 같다. 일일이 관련 내용을 설명하지는 않겠다.

Item	Description
CPU	MT7688AN 580MHz MIPS
Memory	128MB DDR2 RAM/ 32MB FLASH
GPS	UBlox Max 7Q
LAN	10/100 Mbit LAN with 24V POE
Interface	USB-C connector (no USB interface) with GPIO, console UART and I2C; USB-A 2.0
Enclosure	ASA plastic, anti-UV
Size	78 x 340 x 30mm
Operating Temperature	-20°C to 55°C
Extreme Temperature ¹	-40°C to 85°C
Power	3.5W in average, peak 6W

[그림 7.4] MatchX MX1702 스펙

Item	Description
TX	Maximum +27dBm in 868 version
TX	Maximum +30dBm in 915 version
RX	-128dBm at SF7BW125
RX	-143dBm at SF12BW125

[그림 7.5] MatchX MX1702 LoRa RF Performance

Item	Description
Frequency range	863-873MHz
Or	902-928MHz
Impedance	50ohms
VSWR	<1.2:1
Max gain	2.5dbi
Polarization	Vertical
Radiation Pattern	Omni-directional
Connector	SMA(M)
Length	108mm
IP Rating	IP65

[그림 7.6] MatchX MX1702 LoRa Antenna Performance

MatchX Box(MX1702)를 기다렸던 이유는 다른아닌 LBT(경청 모드 ?) 때문이었다(LBT를 지원하는 LoRa Gateway를 찾는게 하늘에 별따기(?) 수준이라 할 수 있다). 우리나라에는 땅덩이가 좁아서 그런가 규제가 너무나도 많다 :(

- 2) LBT(Listen Before Talk): 송신 전 5ms 이상 수신하여 그 수신 신호의 세기가 -65dBm 이하인 경우에 한하여 전파를 발사하고, 4초 이내에 송신을 중단하며 50ms 이상 휴지하여야 함. 국내 917~923.5MHz 대역은 LBT를 적용하여 4초를 송신하던지 0.4초를 송신 하던지를 선택해야 함.

주파수 간섭을 해결하기 위한 방법

3.1.2. LBT 방식

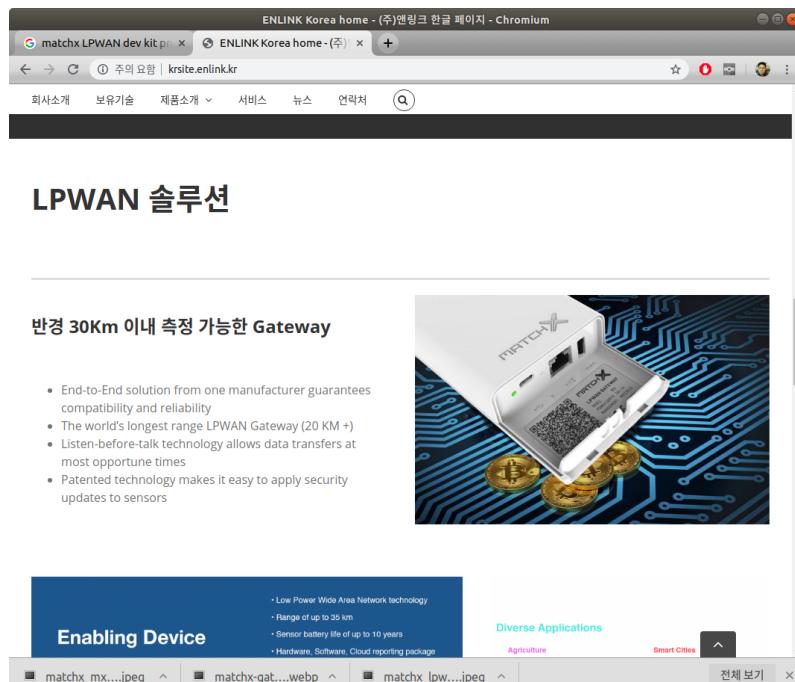
LBT 방식은 유무선 채널에서 널리 사용하는 방식으로 각 디바이스는 전송하기 전에 채널이 비어있는지 먼저 확인을 한 다음에 채널이 비어 있으면 데이터를 전송하고, 만약 동일한 채널을 다른 사용자가 먼저 선점하고 있으면 주어진 시간만큼 백오프를 한 다음 재전송하는 방식이다. LBT를 위해서 최소 송신기 off 시간, 최소 수신기 listen 시간, 단일 송신을 위한 최대 송신기 on 시간, ACK 전송을 위한 최대 송신기 on 시간을 가지도록 한다. 규격은 송신전 5msec 이상 -65dBm의 신호가 있는지 확인한다. 최대 송신기 on 시간은 4초 이내로 하고, 최소 송신기 off시간은 50msec 이상으로 한다.

This module contains functions to configure and use the "Listen-Before-Talk" feature (referred as LBT below). It depends on the loragw_fpga and loragw_radio modules.

LBT feature is only available on SX1301AP2 reference design, which provides the FPGA and the SX127x radio required to accomplish the feature.

[그림 7.7] LBT 문제 정리

헐, 국내 업체 중에 MatchX MX1702를 취급하는 회사가 있다. 그것도 수원에 ... 짹

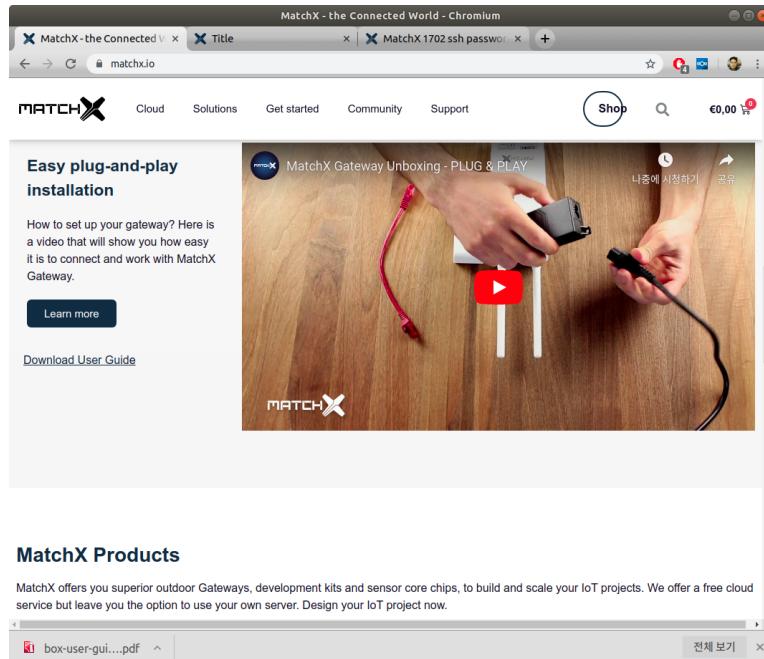


[그림 7.8] MatchX 제품을 취급하는 국내 회사

1) MatchX MX1702 설정

지금부터는 MX1702를 살펴보도록 하겠다. 먼저 아래 동영상을 참조하여 Cable & 전원(PoE)을 연결하도록 하자.

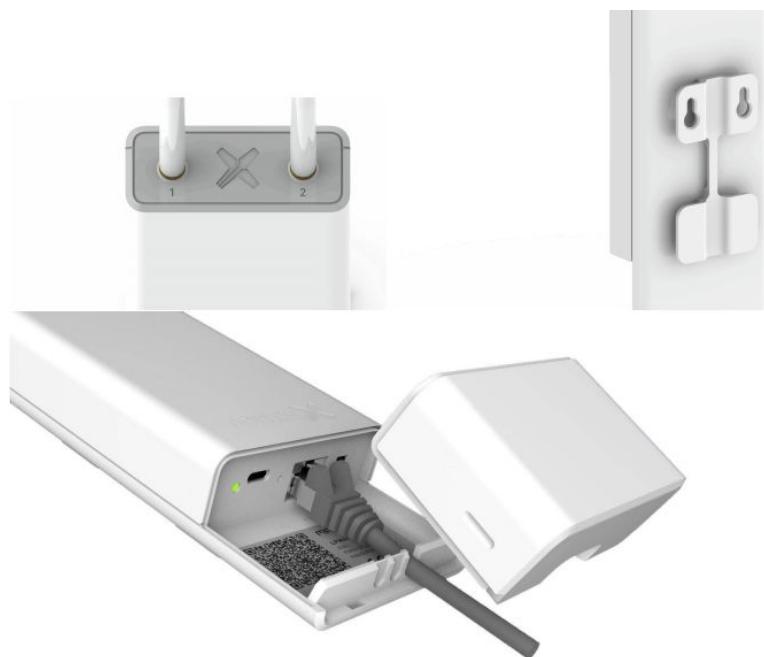
<https://www.matchx.io/>



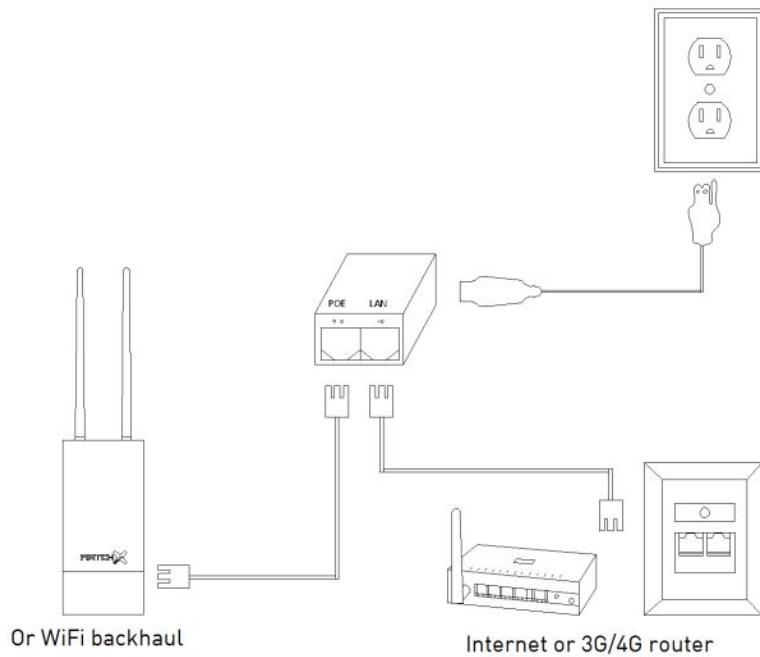
[그림 7.9] MatchX MX1702 연결 및 설정 방법 - 동영상 및 매뉴얼



[그림 7.10] MatchX MX1702 Front Panel



[그림 7.11] MatchX MX1702 외관



[그림 7.12] MatchX MX1702 PoE 연결



[그림 7.13] MatchX MX1702 Wall Mounting 모습

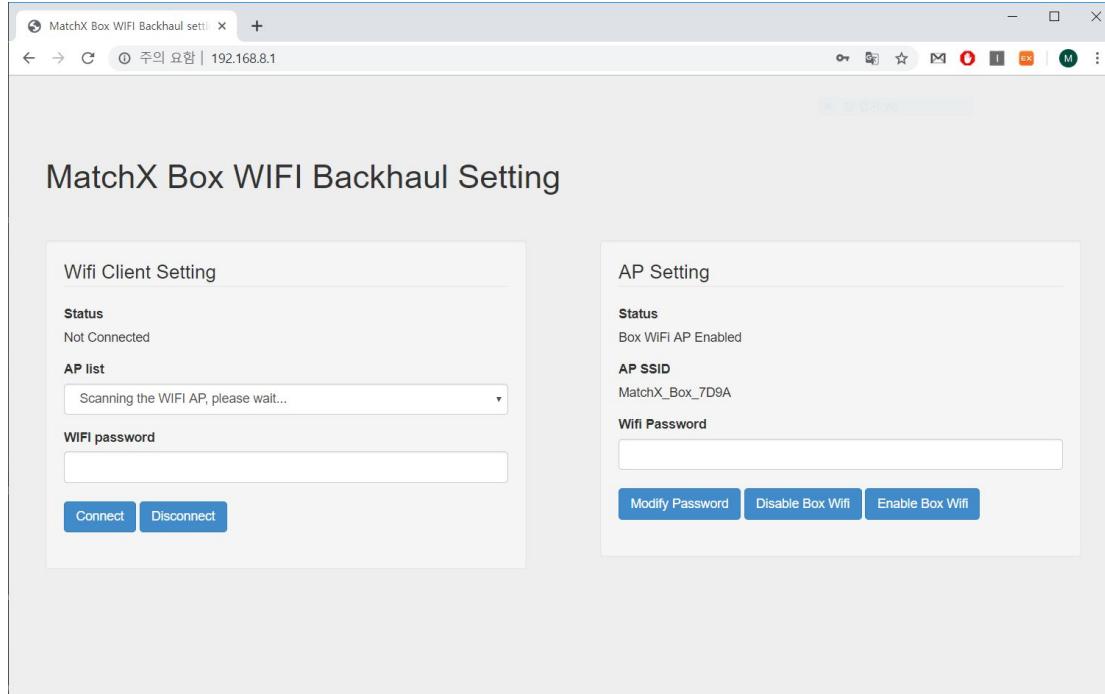


[그림 7.14] MatchX MX1702 Pole Mounting 모습

정상적으로 전원이 on되었으니, 다음으로 Wi-Fi로 MatchX Box(SSID: MatchX_Box_7D9A)에 연결을 시도해 보도록 하자.

주의: wifi password는 장비에 표시(하단 뚜껑을 열어야 함)되어 있는 S/N(예: MXM335XMF)를 사용하면 된다.

<http://192.168.8.1>로 접속(중국애들 style인데 ..)을 시도해 보니, 정상적으로 연결된다. 근데, 이게 뭐냐 ... 꿀랑 1 page만 보인다[**1차 실망**]. 이게 뭐람 ... 그것도 login 정보(id/pass)를 물어보지도 않는다.



[그림 7.15] MatchX MX1702 WebUI 로긴

(나중에 정리한 것임) S/W upgrade(그림 7.18 화면 참조) 후, 아래와 같이 LuCi 화면이 보인다.

The screenshot shows the MXM335XMF LuCI web interface. The top navigation bar includes the device name 'MXM335XMF', the language '개요 - LuCI', and the browser title 'MXM335XMF - 개요 - LuCI - Chromium'. The address bar shows the URL '172.30.1.55/cgi-bin/luci/stok=330be54aecdde9dd82093ca21ad8a6bb'. The main content area has tabs for '상태', '시스템', '메모리', and '네트워크'. The '상태' tab is active, displaying system information such as host name (MXM335XMF), model (BOX-MX1702), firmware version (OpenWrt Chaos Calmer 15.05.1 r49363 / LuCI branch (git-19.270.58466-d87dbc5dc)), kernel version (3.18.29), uptime (Mon Oct 14 05:20:02 2019), boot time (5d 22h 56m 47s), and coordinates (1.23, 0.73, 0.55). The '메모리' tab shows memory usage: total (91880 kB / 126344 kB (72%)), available (88824 kB / 126344 kB (70%)), and free (3056 kB / 126344 kB (2%)). The '네트워크' tab displays the IPv4 WAN status for 'etho.2', which is using DHCP and has an IP of 172.30.1.55, subnet mask 255.255.255.0, gateway 172.30.1.254, and DNS servers 168.126.63.1 and 168.126.63.2.

(실망감을 잠시 뒤로 하고) 이번에는 ssh 연결을 시도해 보기로 하자. Ssh 접속을 위해 인터넷에서 아래 문서를 하나 찾았다. 이 문서에는 default root password가 root로 표시되어 있다.

<https://fccid.io/2AMPF-MX1702/User-Manual/Users-Manual-3507859>

<ssh 접속 시도>

```
$ ssh root@192.168.8.1
```

Passwd: root

뭐냐, 접속이 안된다. 패스워드가 root가 아니다[2차 실망]. 일단 MatchX forum에 문의 글을 남겨 두고, 동영상 중간쯤 부터 보이는 Cloud 서버 연결을 시도해 보기로 하자.

MatchX 1702 ssh password



Michael

3d

Hi,

I got your MatchX 1702 Box today.

I'm trying ssh connection now after confirming web connection, but I don't know the ssh root password.

I just found that root password is 'root' referencing your Users-Manual but I can't still login using it.

Can you help me for this issue ?

Thanks.

Michael

...

Reply


piotr

3d

You can find your SSH password on the Cloud server, on the GWs management page, after checking the checkbox on your organization settings page and accepting the terms and conditions.

Cheers,

Piotr

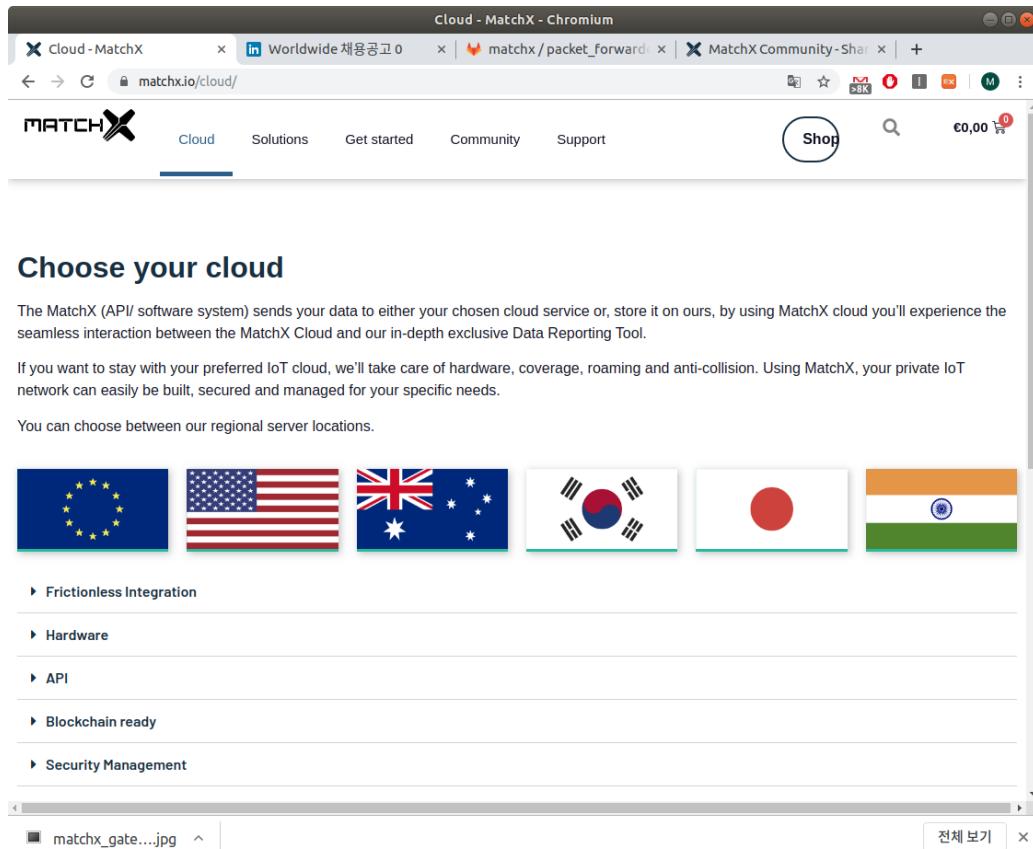
...

Reply

2) MatchX Cloud 설정

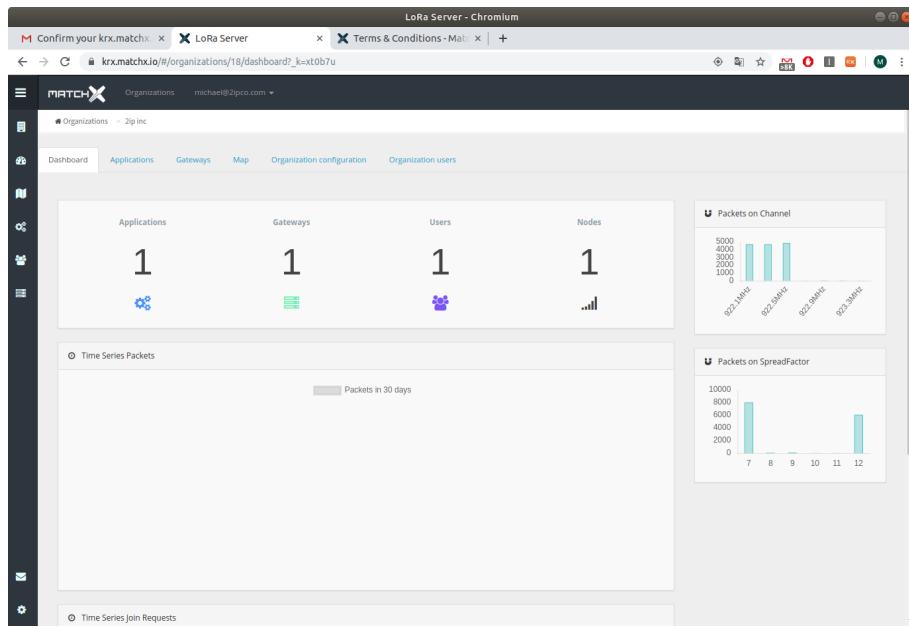
아래 site에서 태극기 아이콘을 선택하여 계정을 하나 등록하자. Email 주소만 입력하면 되는 아주 간단한 작업이라, 추가 설명은 생략한다.

<https://www.matchx.io/cloud/>



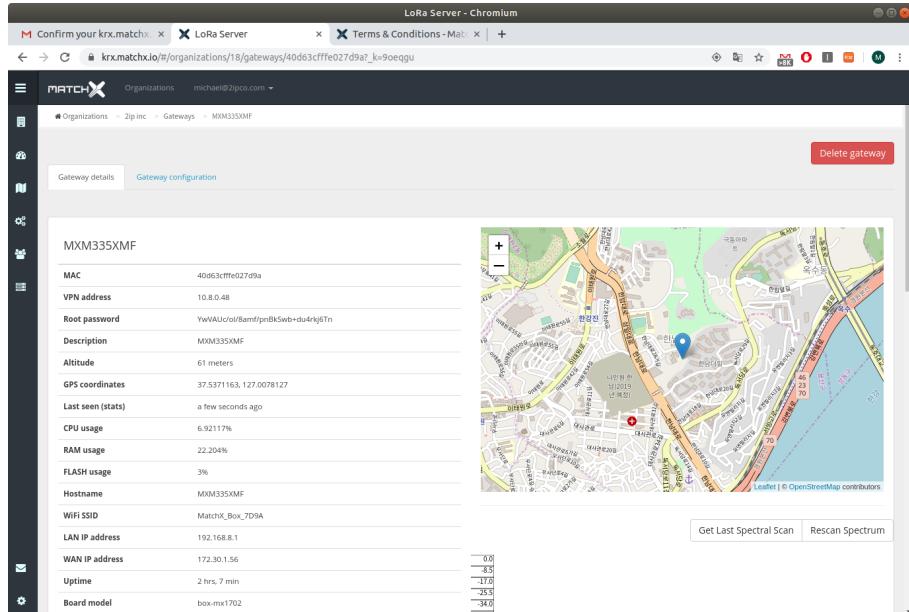
[그림 7.16] MatchX Cloud 선택

계정을 등록한 후에는 로긴하여, LoRa Server 설정을 진행한다. 즉, Gateway를 하나 등록하고, 이어서 Application / Device를 연이어 추가하도록 하자. 이 부분은 TTN, LoRaServer(loraserver.io)와 크게 다르지 않은 부분이라 일일이 설명을 하지는 않겠다. ([한대를 더 주문할 계획이니, 추가 장비가 도착하면](#)) 직접 해 보기 바란다~.

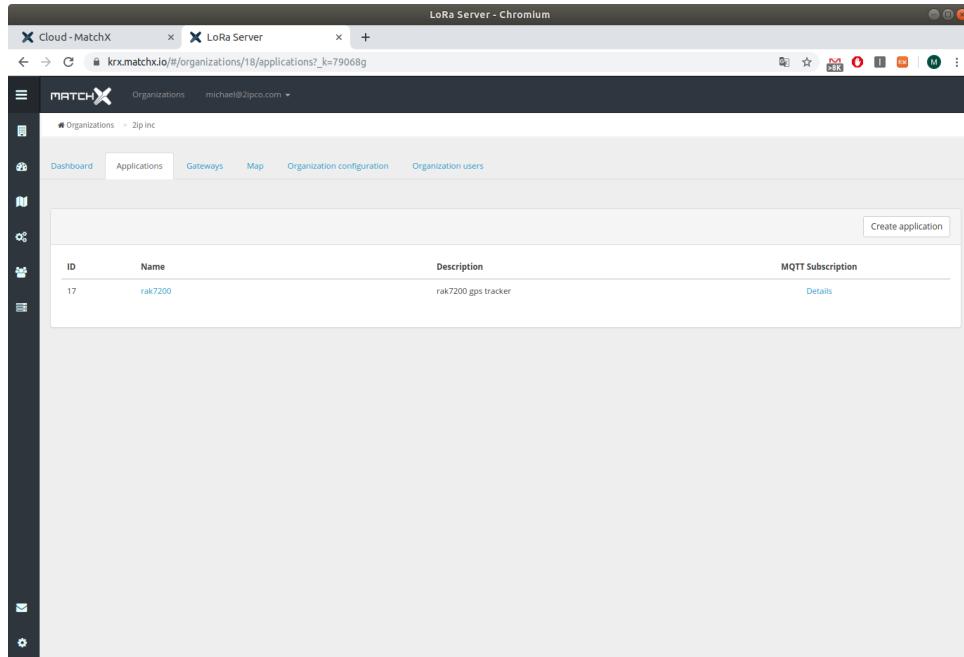


[그림 7.17] MatchX Cloud 화면(1) - dashboard 화면

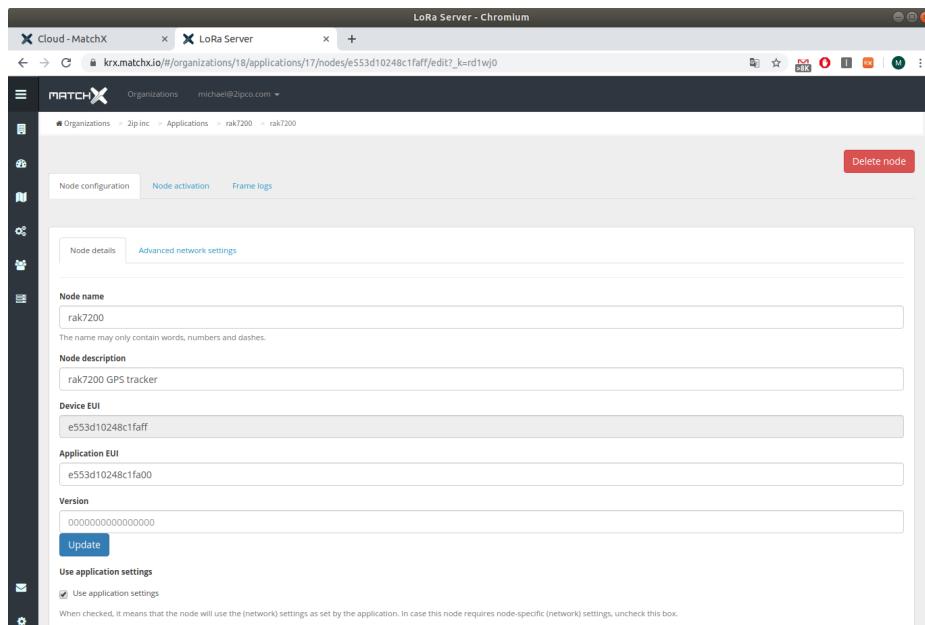
MX1702를 관리하는 WebUI가 없어 자세한 설정 정보를 알 수 없었는데, gateway 화면에서 관련 내용을 확인할 수 있다. 독특하다~.



[그림 7.18] MatchX Cloud 화면(2) - gateway 추가 화면

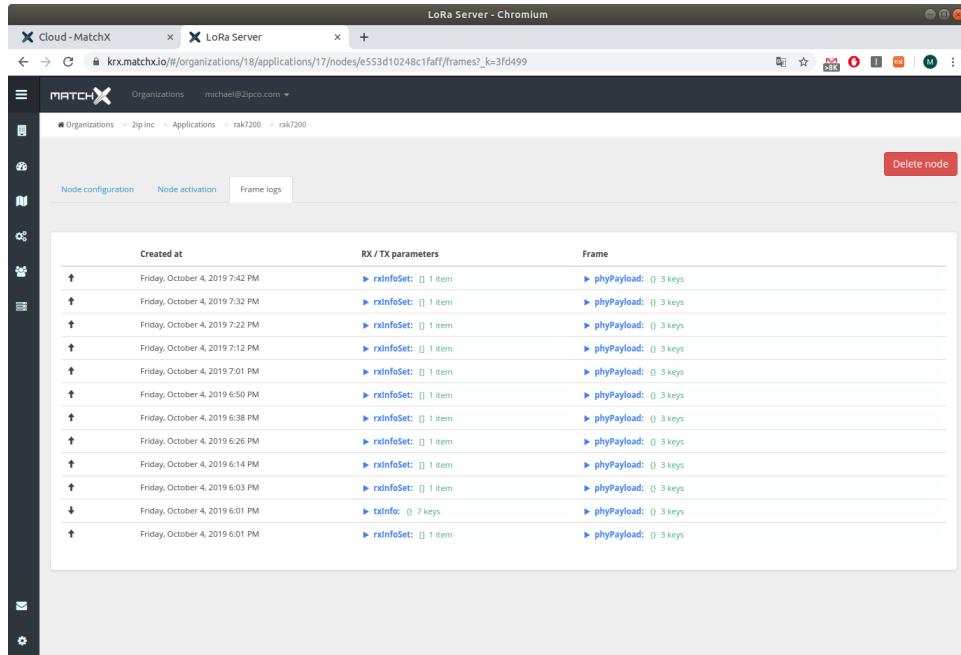


[그림 7.19] MatchX Cloud 화면(3) - Application 화면



[그림 7.20] MatchX Cloud 화면(4) - device 설정 화면

크게 문제 없이, LoRa Node(RAK7200을 사용했음)로 부터 packet이 정상적으로 올라옴을 알 수 있다.



[그림 7.21] MatchX Cloud 화면(5) - LoRa data 수신 화면

<여기서 잠깐>

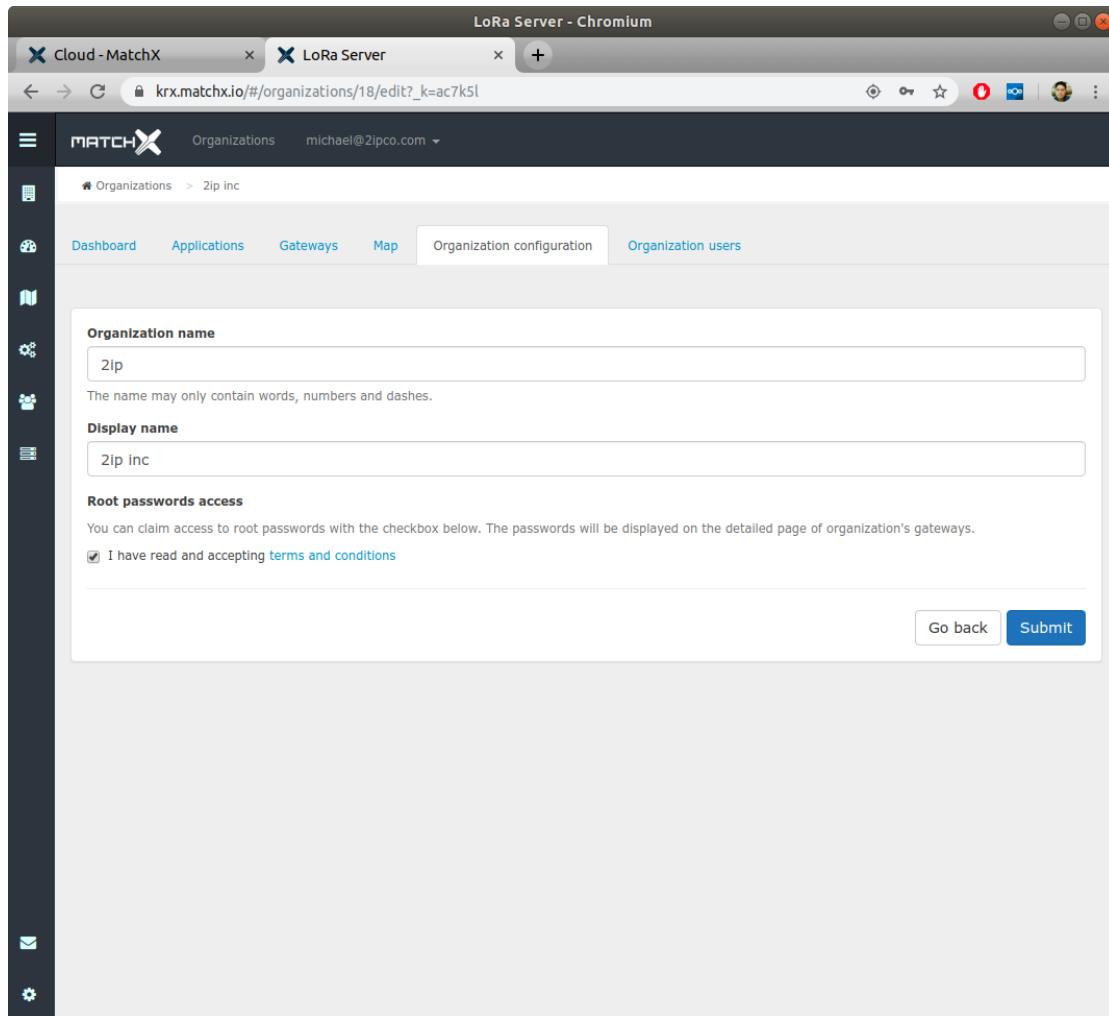
앞서 ssh login시 password를 알 수가 없어, 로긴을 못한 바 있다. 희한하게도 MatchX에서는 아래 그림에 보이는 바와 같이 체크박스를 선택해 좀으로써 root password를 획득(gateway page에서 확인 가능)할 수 있도록 되어 있다. 이후 ssh를 시도해 보니 정상 login이 된다.

Root passwords access

You can claim access to root passwords with the checkbox below. The passwords will be displayed on the detailed page of organization's gateways.

I have read and accepting [terms and conditions](#)

주의: 위의 terms and conditions를 면밀히 살펴볼 필요가 있다.



[그림 7.22] MatchX MX1702 ssh password 설정

참고: 자동으로 생성되어서 그런지 password(예: YwVAUc/oI/8amf/pnBkSwb+du4rkj6Tn)가 무지 같다.

```
chyi@mars:~$ ssh root@172.30.1.56
root@172.30.1.56's password:

BusyBox v1.23.2 (2018-04-04 12:43:18 CEST) built-in shell (ash)

[ _ _ ] [ . . . . . ] [ _ _ _ ] [ . . . . . ]
| - | | - | - | | | | | | - | | - | | - |
|_ _| |_ _| |_ _| |_ _| |_ _| |_ _| |_ _| |_ _|
          | W I R E L E S S   F R E E D O M

-----
CHAOS CALMER (Chaos Calmer, r49363)
-----
* 1 1/2 oz Gin           Shake with a glassful
* 1/4 oz Triple Sec      of broken ice and pour
* 3/4 oz Lime Juice      unstrained into a goblet.
* 1 1/2 oz Orange Juice
* 1 tsp. Grenadine Syrup

-----
root@MXM335XMF:~# ps
```

```

root@MXM335XMF:~# ps
  PID USER      VSZ STAT COMMAND
    1 root      1436 S /sbin/procd
    2 root          0 SW  [kthreadd]
    3 root          0 SW  [ksoftirqd/0]
    5 root          0 SW< [kworker/0:0H]
    6 root          0 SW  [kworker/u2:0]
    7 root          0 SW< [khelper]
   67 root          0 SW< [writenode]
   69 root          0 SW< [bioset]
   72 root          0 SW< [kblockd]
   74 root          0 SW  [kswapd0]
   75 root          0 SW  [kworker/0:1]
  108 root          0 SW  [fsnotify_mark]
  163 root          0 SW  [spi32766]
  215 root          0 SW< [deferwq]
  277 root          0 SW  [kworker/u2:2]
  330 root          0 SWN [jffs2_gcd_mtd6]
  397 root         900 S /sbin/ubusd
  398 root         768 S /sbin/askfirst /bin/login
  547 root          0 SW< [ipv6_addrconf]
  706 root        1056 S /sbin/logd -S 16
  740 root        1600 S /sbin/netifd
  761 root        1196 S /usr/sbin/odhcpd
  892 root        1148 S /usr/sbin/dropbear -F -P /var/run/dropbear.1.pid -p 22 -K 300
 1005 root          0 SW  [RtmpCmdQTask]
 1007 root          0 SW  [RtmpMlmeTask]
 1035 root        1484 S udhcpc -p /var/run/udhcpc-eth0.2.pid -s /lib/netifd/dhcp.script -f -t 0 -
 1064 root        1440 S /usr/sbin/uhttpd -f -h /www -r MXM335XMF -x /cgi-bin -t 60 -T 30 -k 20 -A
 1101 nobody     3756 S /usr/sbin/openvpn --syslog openvpn(matchx_client) --status /var/run/openv
 1110 root        896 S /usr/sbin/mxconfd
 1116 root          0 SW  [spio]
 1274 nobody     992 S /usr/sbin/dnsmasq -C /var/etc/dnsmasq.conf -k -x /var/run/dnsmasq/dnsmasq
 1363 root        1484 S {mxledd} /bin/sh /usr/sbin/mxledd
 1409 root        1488 S /usr/sbin/ntpd -n -S /usr/sbin/ntpd-hotplug -p 0.openwrt.pool.ntp.org -p
 2147 root          0 SW  [kworker/0:2]
 7906 root        1484 S {lora_pkt_fwd.sh} /bin/sh /usr/bin/lora_pkt_fwd.sh
 7910 root        13712 S ./lora_pkt_fwd -d
 14238 root       1216 R /usr/sbin/dropbear -F -P /var/run/dropbear.1.pid -p 22 -K 300
 14244 root       1476 S sleep 5
 14248 root       1488 S -ash
 14260 root        1484 S udhcpc -p /var/run/udhcpc-apcli0.pid -s /lib/netifd/dhcp.script -f -t 0 -
 14265 root       1484 R ps
root@MXM335XMF:~#

```

[그림 7.23] MatchX MX1702 ssh login 모습

<openvpn daemon 구동 모습>

```

/usr/sbin/openvpn --syslog openvpn(matchx_client) --status
/var/run/openvpn.matchx_client.status --cd /var/etc --config openvpn-matchx_client.conf

```

3) MatchX MX1702 & Cloud LoRa Server 장단점

MatchX MX1702를 설치하는 과정에서 느낀 점 몇가지를 간단히 정리해 보면 다음과 같다.

<문제점>

- 1) 전용 WebUI가 없는 관계로 MatchX MX1702 장비의 설정을 변경할 수가 없다.
- 2) MatchX Cloud server로 전달된 data를 제 3의 application server로 전달할 방법이 없다(Web page에서 설정하는 부분은 없는 듯 보임)

→ 나중에 안 사실이지만, MQTT를 이용하면 된다.
- 3) 또한, MX1702(Gateway)에서 곧 바로 다른 LoRa Server로 packet을 전달할 방법도 없다. Cloud server를 MatchX에서 관리할 터인데, 서버에 문제가 발생할 경우, 대략

난감해 질 수 있겠다.

→ 나중에 안 사실이지만, gateway configuration을 변경하면 가능하다. 즉, 우리 LoRaServer와 연동 가능하다.

<장점>

- 1) Cloud 기반 LoRa Server(ex: krx.matchx.io)가 나름 잘 설계된 느낌이다.
→ 아무리 봐도 LoRaServer(loraserver.io) project를 기반으로 만들어진 것 같다.
- 2) OpenVPN 기반 암호 통신(MX1702 ⇄ Cloud 기반 LoRa Server)이 default로 동작한다.
→ SPN을 적용해 볼 수 있는 여지가 없다(우리에게는 단점).
→ 아니다, SPN 적용이 가능할 것 같다.
- 3) 복잡한 설정을 모두 빼 버리고, 간결하게 만들었다.

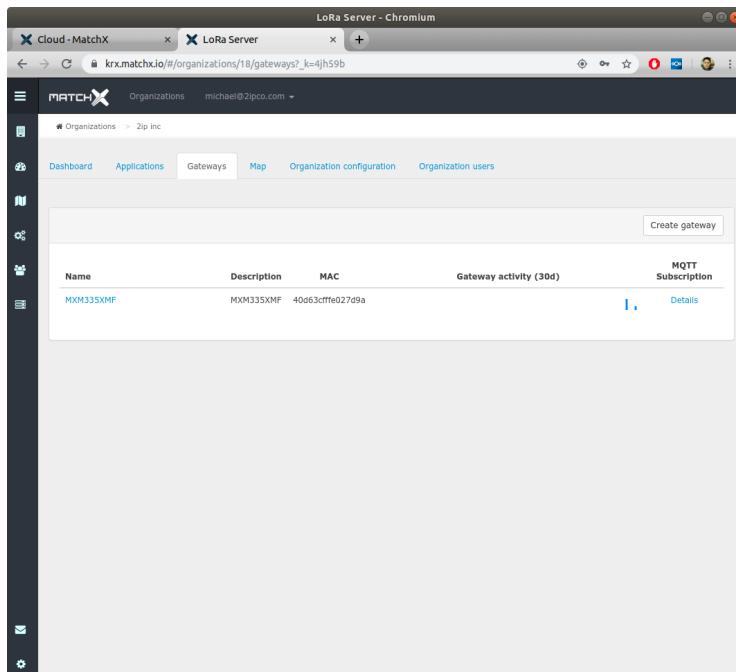
다시 정리해 놓고 보니, 크게 단점을 찾을 수 없을 것 같기도 하다.

4) Cloud 기반 LoRa Server ⇄ External Application Server 연동하기(1)

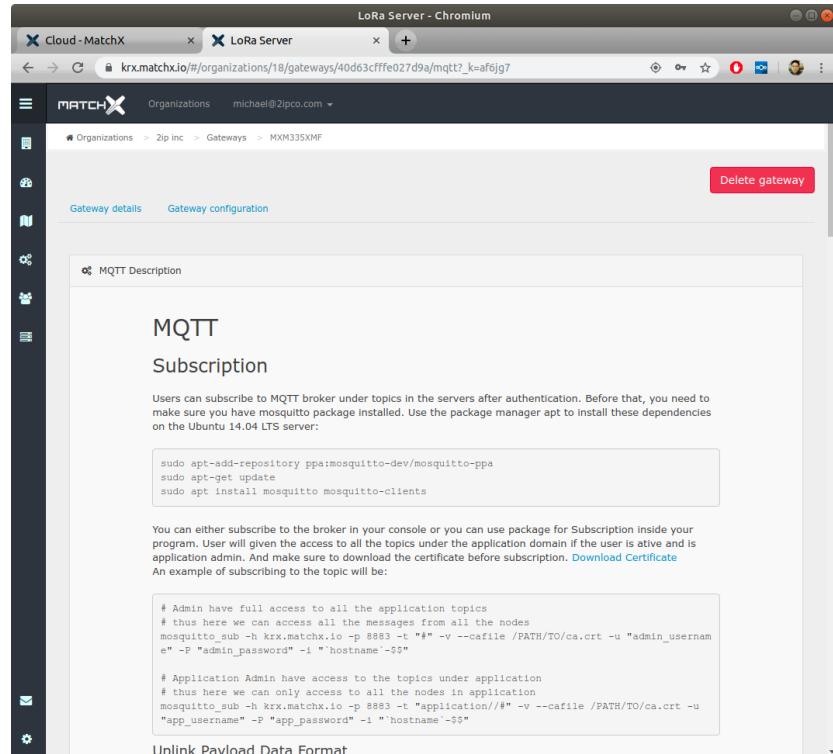
MatchX Cloud web page 상에는 이와 관련한 내용이 보이질 않는다. 다만 아래 화면에 보이는 것과 같이 MQTT subscription 방법을 이용하면 가능할 것 같아 보인다.

<Ubuntu>

```
$ mosquitto_sub -h krx.matchx.io -p 8883 -t "#" -v --cafile /PATH/TO/ca.crt -u "admin_username" -P "admin_password" -i "`hostname`-$$"
```



[그림 7.24] MatchX Gateway MQTT Subscription(1)



[그림 7.25] MatchX Gateway MQTT Subscription(2)

Uplink Payload Data Format

After you have successfully subscribe to the MQTT broker, then you would possibly see something in your console as follow:

```
# data field in json is your payload
{"applicationName":"APP_NAME","nodeName":"stick-0005","devEUI":"001122fffe334455","rxInfo": [{"mac":"667788fffe99aabb","rssi":-108,"loRaSNR":-9.8,"name":"Gateway 667788fffe99aabb","latitude":52.5201761,"longitude":13.4036008,"altitude":0}],"txInfo":{"frequency":867500000,"dataRate":{"modulation":"LORA","bandwidth":125,"spreadFactor":11},"adr":true,"codeRate":"4/5"}, "fcnt":11,"fPort":1,"data":"IdccAQH81OABb7V6AIEC"}
```

Then you can try to parse the data use the following specifications:

```
echo IdccAQH81OABb7V6AIEC | base64 --decode | hexdump -C
00000000  21 d7 1c 01 01 fc d4 e0  01 6f b5 7a 00 81 02      |!.....o.z...|
0000000f
```

[그림 7.26] MatchX Uplink Payload Data Format

이 내용을 토대로 ThingsBoard와 연결하는 작업을 시도해 볼 필요가 있어 보인다. 처음에는 몇가지 실망스러운 부분도 있기는 했지만, 지금은 느낌이 꽤 괜찮다. **이정도만 된다면, LBT가 지원되는 LoRa Gateway를 확보하는 셈이 될 것도 같다.**

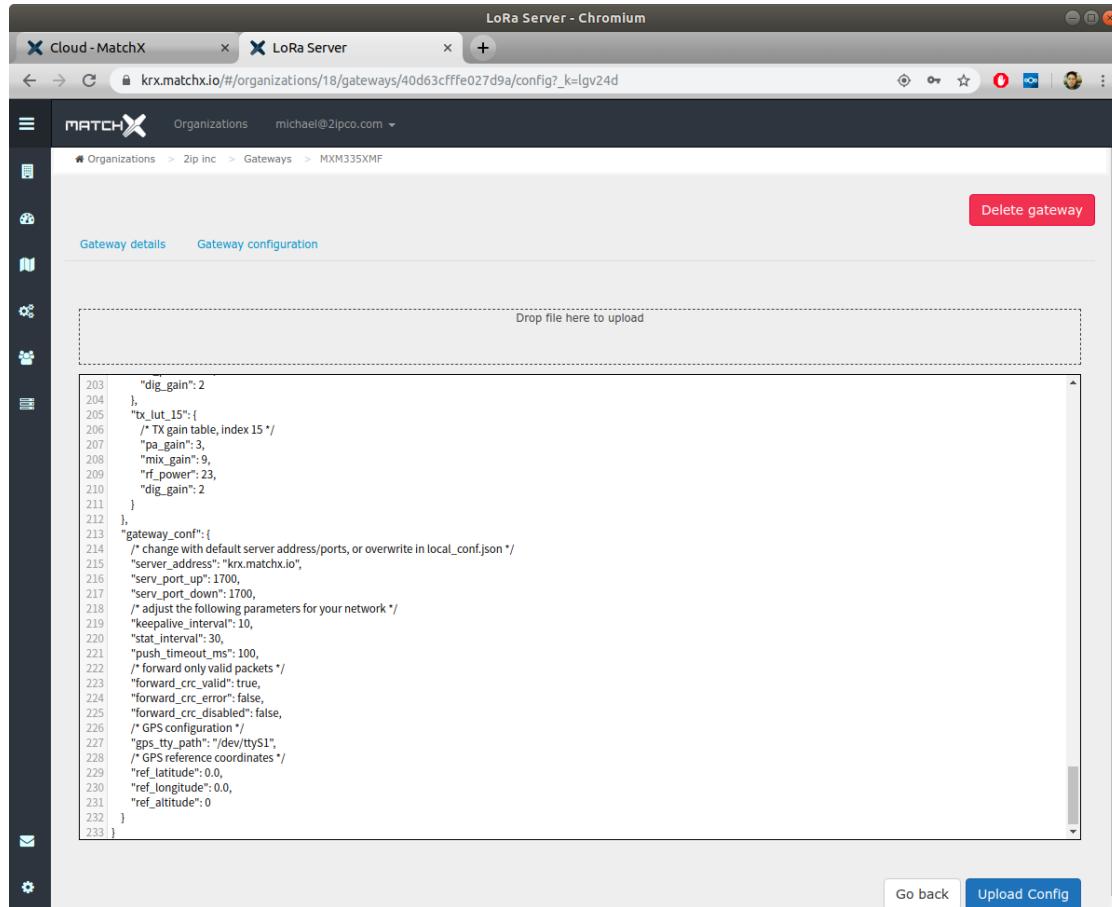
5) Cloud 기반 LoRa Server ⇄ External Application Server 연동하기(2)

어라, 가만 보니, gateway configuration을 바꿀 수 있는 방법을 제공하고 있는 것 같다. 얘를 이용하면 LoRaServer(loraserver.io)와 연동시킬 수도 있겠다. 월요일에 사무실에 출근하면 테스트해 보아야 겠다. 아 ~ 근데, OpenVPN 설정 때문에 안될 가능성도 있어 보인다. 아니다, 어차피 OpenVPN도 ip를 base로 움직이는 것이므로 OpenVPN IP 대역(예: 10.8.0.0/24) 이외의 ip를 server_address로 지정하면 VPN의 영향에서 벗어나서 통신이 가능할 것이다.

```

"gateway_conf": {
    /* change with default server address/ports, or overwrite in local_conf.json */
    "server_address": "krx.matchx.io", /* LoRa Server ip로 바꾸면 될 듯 */
    "serv_port_up": 1700,
    "serv_port_down": 1700,
    /* adjust the following parameters for your network */
    "keepalive_interval": 10,
    "stat_interval": 30,
    "push_timeout_ms": 100,
    /* forward only valid packets */
    "forward_crc_valid": true,
    "forward_crc_error": false,
    "forward_crc_disabled": false,
    /* GPS configuration */
    "gps_tty_path": "/dev/ttyS1",
    /* GPS reference coordinates */
    "ref_latitude": 0.0,
    "ref_longitude": 0.0,
    "ref_altitude": 0
}

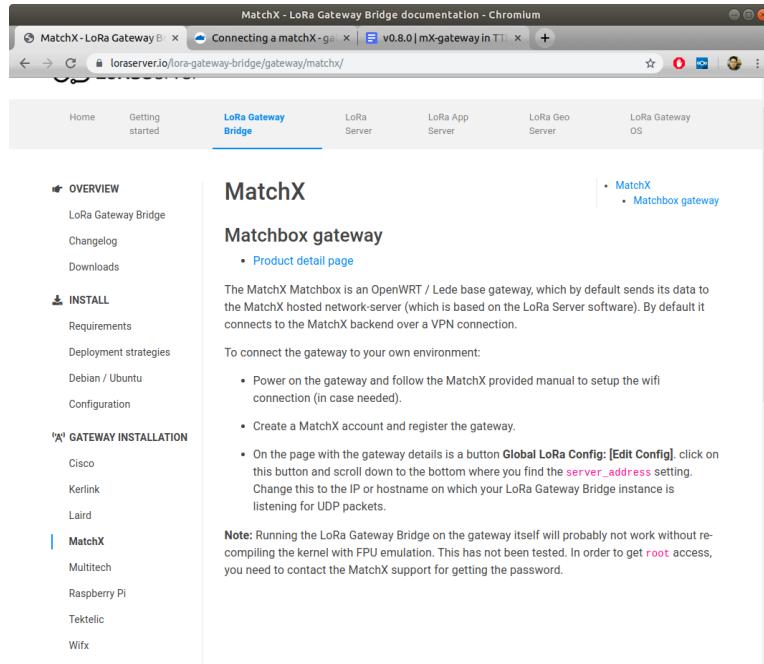
```



[그림 7.27] MatchX MX1702 Gateway Config 설정

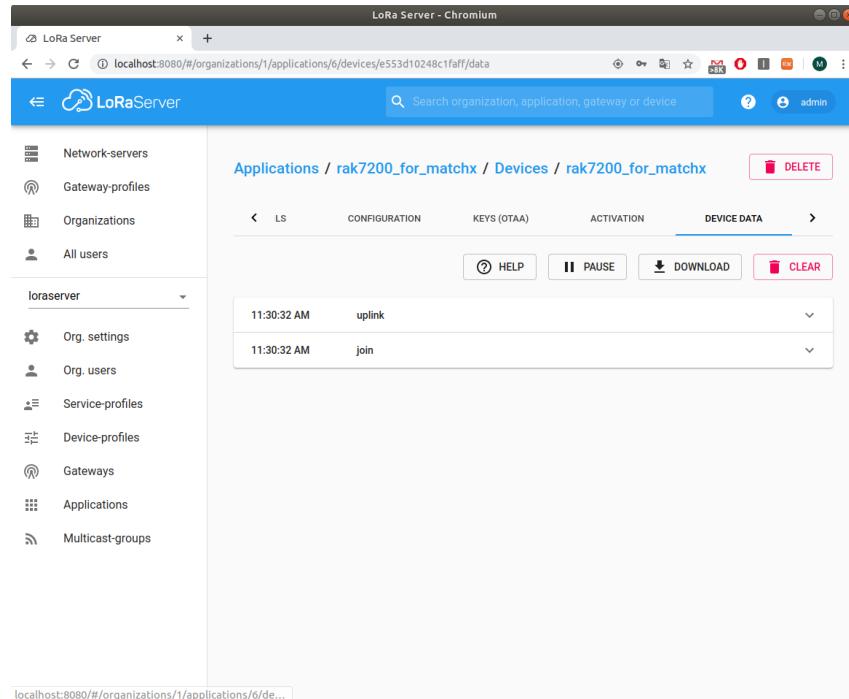
ㅎㅎ 이를 뒷 받침하는 또 다른 내용이 LoRa Server web site에도 나와 있다.

<https://www.loraserver.io/lora-gateway-bridge/gateway/matchx/>



[그림 7.28] MatchX and LoRaServer Integration 관련

연결해 보자. OK 된다 :) LoRa Server 설정과 관련해서는 이미 5장에서 상세하게 소개하고 있으니, 여기서는 중복해서 설명하지는 않도록 하겠다.



[그림 7.29] MatchX and LoRaServer Integration 화면

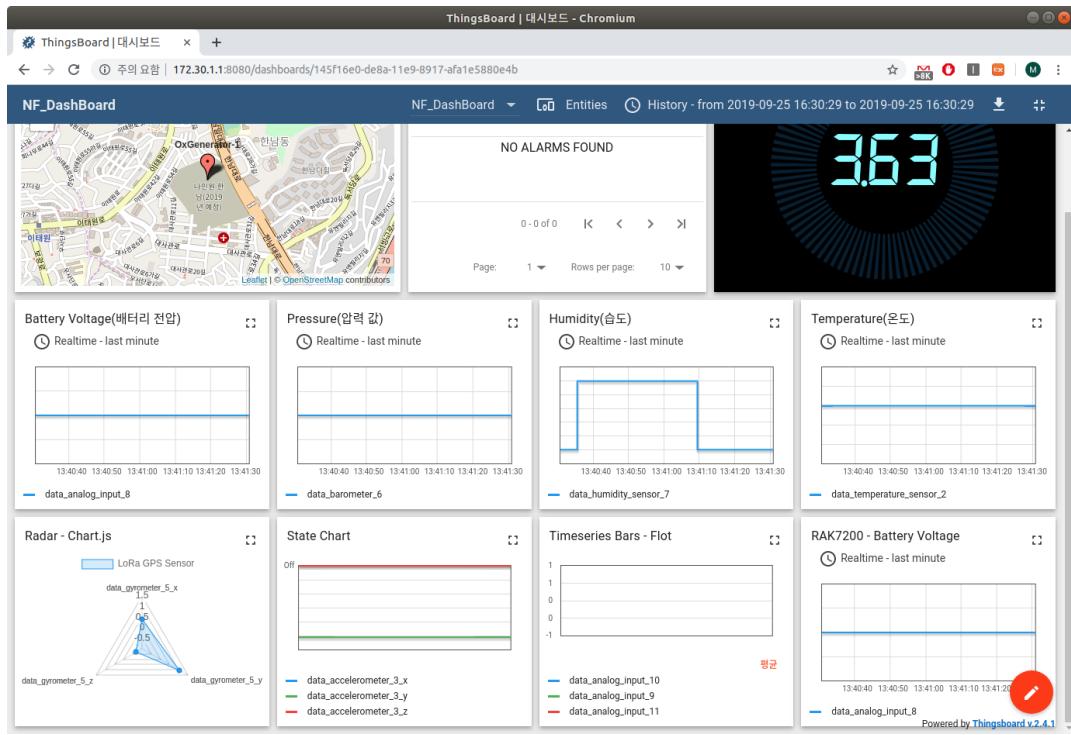
6) ThingsBoard와 연동하기

Cayenne LPP(6장 끝 부분 및 12장 중후반부에 정리되어 있음)로 LoRa Server와 ThingsBoard를 연결시킨 후, LORA GPS SENSOR 디바이스 "최근 데이터" 정보를 확인해 보니, 센서 정보(Battery Voltage, Accelerometer, Gyro, Magnetometer 정보가 제대로(?) 들어옴을 알 수 있다. **단, 이들 값 중 일부 값이 + ⇒ -로 표기되는 문제가 있다. 왜 그럴까? <TBD>**

The screenshot shows the ThingsBoard web interface. On the left sidebar, under the 'Devices' section, there are two entries: 'Temp_Sensor' and 'RAK7204_LoRa_Sensor'. The 'RAK7204_LoRa_Sensor' entry is expanded, showing its entity type as 'LORA_GPS_SENSOR' and its description as 'LORA_TEMP_HUMI_PRESSURE'. Below this, it says 'Customer에 할당됨 'NF''. To the right of the sidebar, a modal window titled 'LORA GPS SENSOR' displays the 'Recent Data' tab. This tab lists various data points with their last update times and values. A red circle highlights the edit icon (pencil) in the top right corner of the modal.

상세	속성	최근 데이터	ALARMS	이벤트	RELATIONS	AUDIT LOGS
<input type="checkbox"/> Last update time	Key ↑	Value				
<input type="checkbox"/> 2019-10-07 13:26:27	data_accelerometer_3_x	0.015				
<input type="checkbox"/> 2019-10-07 13:26:27	data_accelerometer_3_y	-0.987				
<input type="checkbox"/> 2019-10-07 13:26:27	data_accelerometer_3_z	-0.091				
<input type="checkbox"/> 2019-10-07 13:26:27	data_analog_input_10	43.65				
<input type="checkbox"/> 2019-10-07 13:26:27	data_analog_input_11	-40.2				
<input type="checkbox"/> 2019-09-25 16:30:17	data_analog_input_3	-3.41				
<input type="checkbox"/> 2019-09-25 18:01:19	data_analog_input_4	7.28				
<input type="checkbox"/> 2019-10-07 13:26:27	data_analog_input_8	3.59				
<input type="checkbox"/> 2019-10-07 13:26:27	data_analog_input_9	108.9				
<input type="checkbox"/> 2019-09-25 18:01:19	data_analog_output_3	-3.2				
<input type="checkbox"/> 2019-10-07 13:26:27	data_barometer_6	0				

[그림 7.30] LoRaServer와 ThingsBoard Integration 화면(1)



[그림 7.31] LoRaServer와 ThingsBoard Integration 화면(2)

7) Mesh SPN Porting하기

이미 언급한 바와 같이 MatchX MX1702는 OpenVPN이 기본적으로 동작하고 있다. LoRa Server를 2ip 것으로 교체할 경우에는 OpenVPN을 사용할 수 없는(설정이 불가하기 때문) 바, 우리는 Mesh SPN을 사용하도록 만들어야 한다.

```
1635 root 1484 S udhcpc -p /var/run/udhcpc-eth0.2.pid -s /lib/netifd/dhcp.script -f -t 0 -i eth0.2 -C
1064 root 1440 S /usr/sbin/uhttpd -f -h /www -r MXM335XMF -x /cgi-bin -t 60 -T 30 -k 20 -A 1 -n 3 -N 100 -R -p 0.0.0.0:80 -p [::]:80
1101 nobody 3756 S /usr/sbin/openvpn --syslog openvpn(matchx_client) --status /var/run/openvpn.matchx_client.status --cd /var/etc --config openvpn-matchx_client.conf
1110 root 896 S /usr/sbin/mxconfd
```

[그림 7.32] MatchX MX1702 board에서 동작 중인 openvpn daemon

Mesh SPN Porting과 관련한 구체적인 내용은 9장에서 자세히 소개하고 있다. 다만, 9장에서는 RAK7258을 target으로 porting 과정을 설명하고 있는데, 문제는 이러한 내용이 MX1702에도 그대로 적용 가능한가이다. 결과만 간단히 요약해 보면 다음과 같다.

1. MX1702는 OpenWrt style의 WebUI firmware upgrade 방식을 지원하지 않는다. 또한 MatchX에서 release한 firmware file이 없다. 따라서, 이미지를 분해하여 SPN S/W를 탑재하는 형태의 s/w upgrade 방식은 현재로써는 불가하다.

2. 하지만, 개별 binary(실행 파일)은 테스트해 보니, 동작한다. 따라서 Mesh SPN은 porting이 가능하다.
3. 설치는 SPNBox X86 series 처럼 수동 설치를 해야만 한다.
4. (아래 테스트 내용을 봄서는) Vtysh도 x86 series 처럼 build해야 할 듯 보인다.

아래 그림은 RAK7258 환경에서 build한 vtysh과 2ston_tinc를 실행한 모습(결과)이다. **Vtysh**은 library 문제로 실행이 안되지만, **2ston_tinc**는 실행인 된다.

```
root@MXM335XMF:~/workspace/spnbox_install/spnbox_pkg/usr/lib# export LD_LIBRARY_PATH=/root/workspace/spnbo
x_install/spnbox_pkg/usr/lib:$LD_LIBRARY_PATH
root@MXM335XMF:~/workspace/spnbox_install/spnbox_pkg/usr/lib#
root@MXM335XMF:~/workspace/spnbox_install/spnbox_pkg/usr/lib# ls -la
drwxr-xr-x 2 1000 1000 0 Oct 8 00:48 .
drwxr-xr-x 4 1000 1000 0 Oct 8 00:48 ..
lrwxrwxrwx 1 root root 14 Oct 8 00:48 libcrypto.so -> libcrypto.so.3
-rwxr-xr-x 1 1000 1000 2952012 Sep 18 10:42 libcrypto.so.3
lrwxrwxrwx 1 root root 16 Oct 8 00:48 liblzo2.so -> liblzo2.so.2.0.0
lrwxrwxrwx 1 root root 16 Oct 8 00:48 liblzo2.so.2 -> liblzo2.so.2.0.0
-rwxr-xr-x 1 1000 1000 134444 Sep 18 10:37 liblzo2.so.2.0.0
-rw-r--r-- 1 1000 1000 622074 Sep 18 10:43 libncurses.a
lrwxrwxrwx 1 root root 18 Oct 8 00:48 libreadline.so -> libreadline.so.8.0
lrwxrwxrwx 1 root root 18 Oct 8 00:48 libreadline.so.8 -> libreadline.so.8.0
-rwxr-xr-x 1 1000 1000 347988 Sep 18 10:43 libreadline.so.8.0
lrwxrwxrwx 1 root root 11 Oct 8 00:48 libssl.so -> libssl.so.3
-rwxr-xr-x 1 1000 1000 587580 Sep 18 10:42 libssl.so.3
```

```
root@MXM335XMF:~/workspace/spnbox_install/spnbox_pkg/sbin# ./vtysh
./vtysh: can't load library 'libreadline.so.6'
root@MXM335XMF:~/workspace/spnbox_install/spnbox_pkg/sbin# ls -la
drwxr-xr-x 2 1000 1000 0 Oct 8 00:48 .
drwxr-xr-x 7 root root 0 Oct 8 00:48 ..
-rwxr-xr-x 1 1000 1000 260676 Oct 8 00:40 2ston_tinc
-rwxr-xr-x 1 1000 1000 828 Apr 17 08:17 fw3.sh
-rwxr-xr-x 1 1000 1000 358716 Oct 8 00:40 tincd
-rwxr-xr-x 1 1000 1000 155959 Sep 18 09:02 vtysh
root@MXM335XMF:~/workspace/spnbox_install/spnbox_pkg/sbin# ./2ston_tinc
tinc>
tinc> ?
Unknown command `?'.
tinc> help
Usage: ./2ston_tinc [options] command

Valid options are:
  -b, --batch           Don't ask for anything (non-interactive mode).
  -c, --config=DIR      Read configuration options from DIR.
  -n, --net=NETNAME     Connect to net NETNAME.
  --pidfile=FILENAME   Read control cookie from FILENAME.
  --force               Force some commands to work despite warnings.
  --help                Display this help and exit.
  --version             Output version information and exit.

Valid commands are:
  init [name]            Create initial configuration files.
  get VARIABLE           Print current value of VARIABLE
  set VARIABLE VALUE    Set VARIABLE to VALUE
  add VARIABLE VALUE    Add VARIABLE with the given VALUE
  del VARIABLE [VALUE]  Remove VARIABLE [only ones with watching VALUE]
  start [tincd options] Start tincd.
  stop                  Stop tincd.
  restart [tincd options] Restart tincd.
  reload                Partially reload configuration of running tincd.
```

[그림 7.33] MatchX MX1702 board에서 2ston_tinc binary 실행 모습

(나중에 정리한 것임) MatchX_MX1702 model에 대해 build한 결과물을 MX1702 box에 설치한 모습은 다음과 같다.

```
<Desktop PC>
$ scp ./spnbox_install.tar.gz root@172.30.1.55:~/workspace
$ ssh root@172.30.1.55

<target board>
# cd ~/workspace
# tar xvzf spnbox_install.tar.gz
# cd spnbox_install
# ./Install.sh
...
...
# reboot

<Desktop PC>
$ ssh root@172.30.1.55
```

```
* 1 1/2 oz Gin           Shake with a glassful
* 1/4 oz Triple Sec      of broken ice and pour
* 3/4 oz Lime Juice      unstrained into a goblet.
* 1 1/2 oz Orange Juice
* 1 tsp. Grenadine Syrup

Build On Oct 11 2019 11:04:54
MXM335XMF>
enable  Turn on privileged mode command
exit    Exit current mode and down to previous mode
ping    Send echo messages
show    Show running system information
ssh     Open a ssh connection
MXM335XMF> en
MXM335XMF# configure terminal
MXM335XMF(config)# show running-config
#Writed on Mon Oct 14 06:22:57 2019
password 8 spYzDw10qDeMQ
!
MXM335XMF(config)#  
bridge      add/modify the bridge information
date        Set the date
enable      Modify enable password parameters
exit        Exit current mode and down to previous mode
factory    Go back to the factory default state
hostname   Set system's network name
ip          IP information set
meshvpn    Configure meshvpn tunnel
nameserver Config the dns server
no         Negate a command or set its defaults
p2p        Configure p2p tunnel
password   Modify password parameters
ping       Send echo messages
se         Configure SoftEther VPN
sfirewall  Configure spn firewall rules
show       Show running system information
spn        Configure SPN rules
ssh        Open a ssh connection
swupgrade  spnbox software upgrade
write      Write running configuration to memory, network, or terminal
MXM335XMF(config)#
```

8) LBT 동작 확인

<TBD> 근데, 이걸 어찌 확인하지 ? 아래 gateway configuration 내용으로 봐서는 분명히 LBT가 enable되어 있긴 한데 ...

```

1  {
2    "SX1301_conf": {
3      "lorawan_public": true,
4      "clksrc": 1,
5      "antenna_gain": 2.5,
6      "lbt_cfg": {
7        "enable": true,
8        "rssi_target": -81,
9        "chan_cfg": [
10          { "freq_hz": 922100000, "scan_time_us": 128 },
11          { "freq_hz": 922300000, "scan_time_us": 128 },
12          { "freq_hz": 922500000, "scan_time_us": 128 },
13          { "freq_hz": 922700000, "scan_time_us": 128 },
14          { "freq_hz": 922900000, "scan_time_us": 128 },
15          { "freq_hz": 923100000, "scan_time_us": 128 },
16          { "freq_hz": 923300000, "scan_time_us": 128 }
17        ],
18        "sx127x_rssi_offset": -7
19      },

```

[그림 7.34] MatchX MX1702 LBT enable

9) LPWAN Dev Kit 사용해 보기

MX1702와 함께 LPWAN Dev Kit도 하나 구매를 했다. 애도 테스트해 보아야 한다. LBT는 LoRa Gateway 만의 문제가 아니다. LoRa Node도 이를 지원해야 한다(맞나 ???).

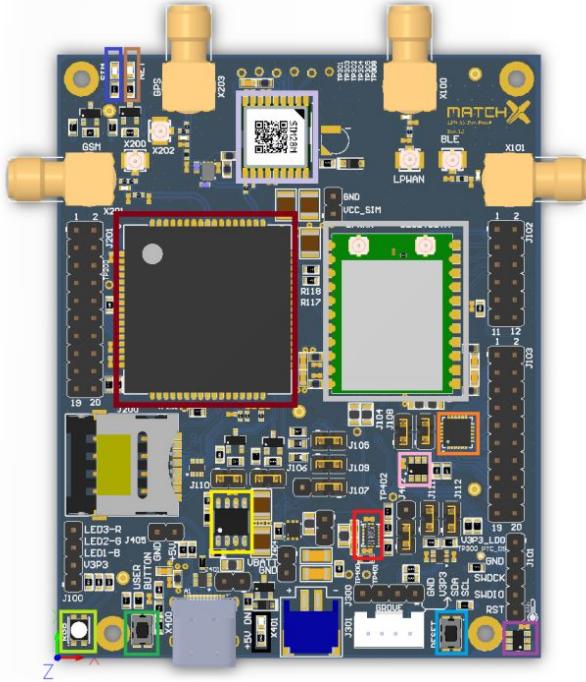


[그림 7.35] MatchX LPWAN Dev Kit(1)

아래 web page에 “Dev Kit User Guide”가 있으니, download 받아 확인해 보도록 하자.

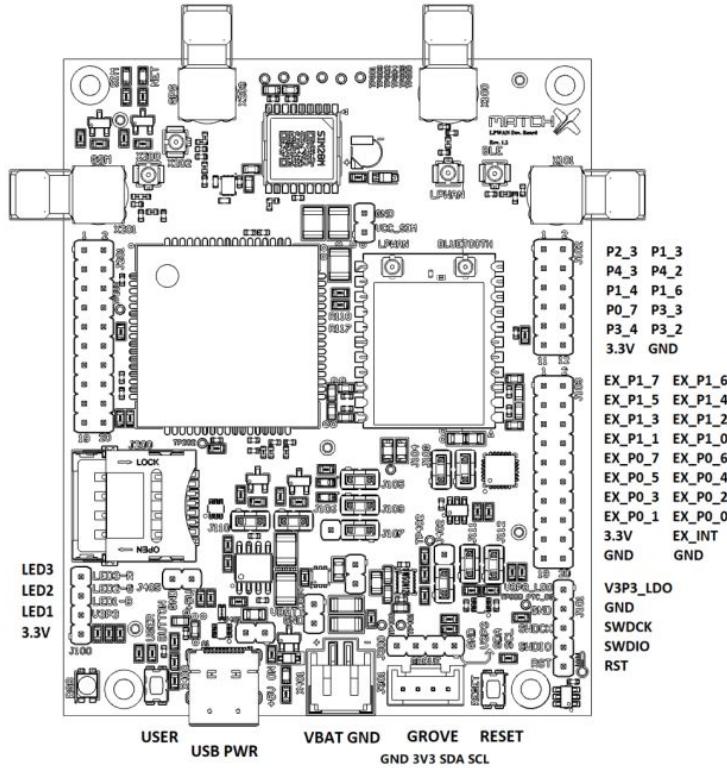
<https://www.matchx.io/getstarted/>

9-1) MatchX Dev Kit H/W Review



Item	Description
MCU	DA14680, 0 Hz up to 96 MHz 32-bit ARM Cortex-M0
Memory	8Mb Flash, 64kB OTP, 128kB ROM, 144kB SRAM
Interfaces	I ² C, I ² S, PCM, SPI, UART, USB, GPIOs
Wireless	Bluetooth 4.1 and LoRa
Battery	2.0mm pitch connector
Size	32 x 148 x 32mm (including Hat)

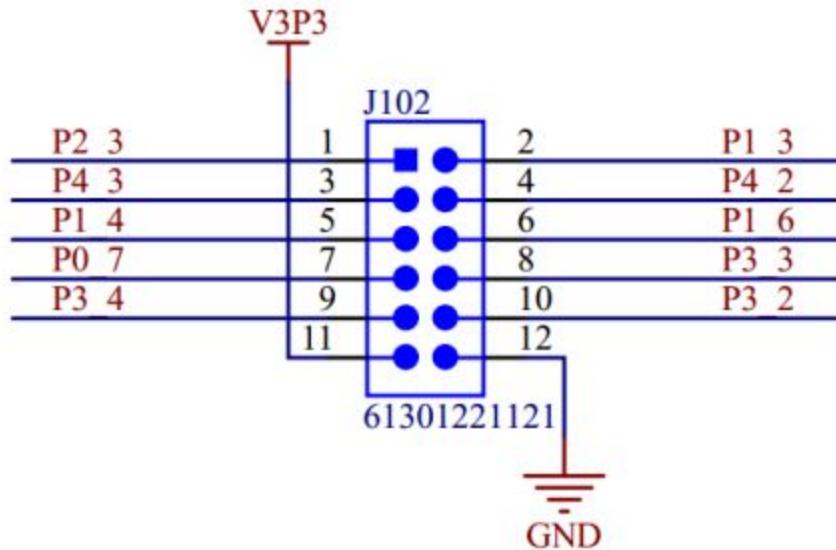
[그림 7.36] MatchX LPWAN Dev Kit(2)



[그림 7.37] MatchX LPWAN Dev Kit(3) - Connector Pin Outs

Pin	Name	Function	Description
1	P2_3	UART_SIM_RX	UART interface to SIMCom module, connected with J109 jumper
2	P1_3	UART_SIM_TX	UART interface to SIMCom module, connected with J105 jumper
3	P4_3	I2C_SCL	Main I2C bus, connected with J104 jumper
4	P4_2	I2C_SDA	Main I2C bus, connected with J108 jumper
5	P1_4	-	Not used
6	P1_6	-	Not used
7	P0_7	-	Not used
8	P3_3	-	Not used
9	P3_4	PS_EN	controls enable pin of TPS62740
10	P3_2	USR_BUTTON	User button connection
11	V3P3	3.3V	3.3V power, the source selectable by J402 jumper
12	GND	GND	Ground

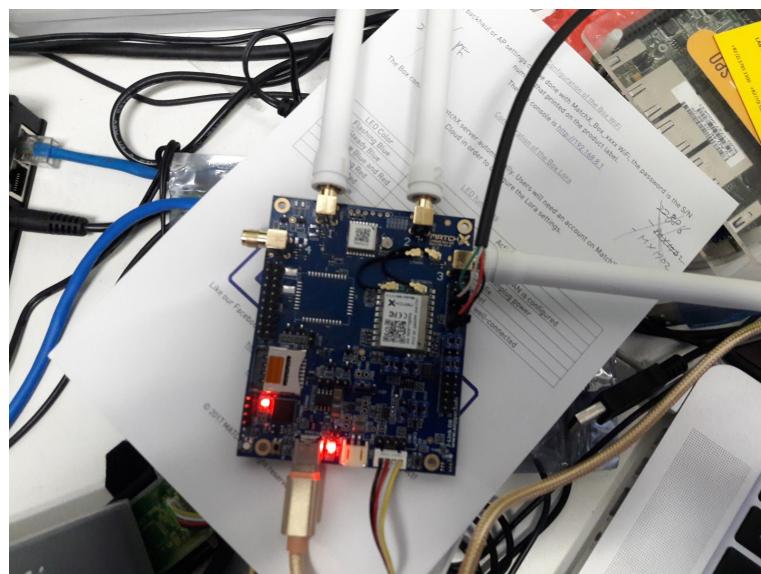
[그림 7.38] MatchX LPWAN Dev Kit(4) - MatchX SoM GPIO



[그림 7.39] MatchX LPWAN Dev Kit(5) - J102 Connector

9-2) MatchX Dev Kit Console 연결하기

그림 15.5의 J102 Connector 중, Pin 5, Pin 6이 각각 UART TX, RX에 해당한다. 따라서 아래 그림처럼 **Pin 5에는 RX**, **Pin 6에는 TX**, **Pin 11에는 V3.3**, **Pin 12에는 Ground** 연결하는 UART-to-USB cable을 PC(Linux)와 연결하도록 한다. 이후 MatchX Dev Kit에 포함되어 있는 USB-C cable을 역시 PC에 연결(power on)하도록 하자.



[그림 7.40] MatchX LPWAN Dev Kit - USB-C power & Serial 케이블 연결하기

아래 그림은 Linux PC minicom(115200, 8N1)으로 확인한 내용이다.

```

chyl@mars: ~
파일(F) 편집(E) 보기(V) 검색(S) 터미널(T) 도움말(H)
00
70b3d5ffe1cb44e
0:01.050+11 lora reset #0
region 00
*** DevKit 1.2 ***
70b3d51cb44e
70b3d500001cb44e
07848a01e93948af9d0f7f5ed1aa430c
00
00
ff
00
70b3d5ffe1cb44e
0:01.050+03 lora reset #0
region 00
*** DevKit 1.2 ***
70b3d51cb44e
70b3d500001cb44e
07848a01e93948af9d0f7f5ed1aa430c
00
00
ff
00
70b3d5ffe1cb44e
0:01.050+09 lora reset #0
region 00
*** DevKit 1.2 ***
70b3d51cb44e
70b3d500001cb44e
07848a01e93948af9d0f7f5ed1aa430c
00
00
ff
00
70b3d5ffe1cb44e
0:01.050+02 lora reset #0
region 00
[ctrl-A Z for help | 115200 8N1 | NOR | Minicom 2.7.1 | VT102 | Online 0:8 | ttyUSB0 ]

```

[그림 7.41] MatchX LPWAN Dev Kit - Serial Console Emulator(minicom)

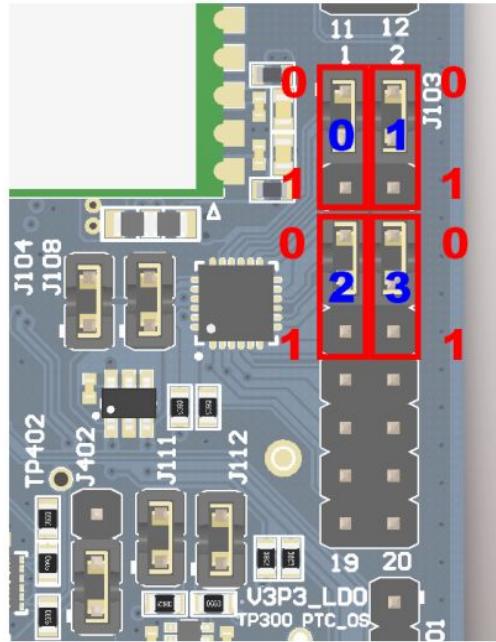
9-3) MatchX Dev Kit 설정 바꾸기

Console 상에서 설정을 바꾸기 위해서는 "param x value" 명령을 실행해 주면 된다.

<KR region 변경하기>

param 5 02

Region을 변경하기 위해서는 위와 같이 console 상에서 명령을 실행해도 되고, 아래와 같이 jumper를 변경해 주어도 된다. 우선 순위는 s/w 방법에 있다. 즉, 아래 jumper를 변경하더라도 위와 같이 설정 변경을 했다면, 이를 기준으로 동작하게 된다.



Region number	Jumper Position				Region	Description
	3	2	1	0		
00	0	0	0	0	EU	Europe region
01	0	0	0	1	AS1	Asia AS923MHz ISM Band
02	0	0	1	0	KR	Korea region
08	0	1	0	0	US	USA region, 8 channels
09	0	1	0	1	AU	Australia region, 8 channels
18	1	1	0	0	US(full)	USA region, full 64 channels
19	1	1	0	1	AU(full)	Australia region, full 64 channels

[그림 7.42] MatchX LPWAN Dev Kit - Region 선택용 jumper

<DevEUI 변경하기>

param 0 XXXXXXXXX

<AppEUI 변경하기>

param 1 XXXXXXXXXXXXXXXXXXXX

<AppKey 변경하기>

param 2 XXXXXXXXXXXXXXXXXXXXXXXX

9-4) MatchX Dev Kit - S/W 개발 환경 꾸미기

<Dialog DA14680 microcontroller 관련 Reference 문서>

- DA14680-01 DS, Datasheet, Dialog Semiconductor
- UM-B-057-SmartSnippets Studio user guide, User manual, Dialog Semiconductor
- UM-B-047 DA1468x Getting Started, User manual, Dialog Semiconductor
- UM-B-044 DA1468x Software Platform Reference, User manual, Dialog Semiconductor
- UM-B-056 DA1468x Software Developer's Guide, Dialog Semiconductor

<준비물>

- MatchX Development Kit
- USB-C cable and 5V charger
- UART-USB converter
- JLink programmer ↳ 헐, 이게 필요하네 ...
- **Dialog's Semiconductor SmartSnippets DA1468x SDK**
- SmartSnippets Studio package
- **MatchX Dev Kit Firmware**
- Operating System (Windows or Linux)

뭐야, MatchX Dev Kit Firmware source code를 찾을 수가 없다. JLink programmer를 보드에 연결하려고 보니, pin이 나와 있지 않다. 헐~ 도대체 어쩌라는 걸까? 아무래도 MatchX Dev Kit은 별로 도움이 안될 모양이다.

