Computer Network Term Project

Chong-kwon Kim
2017

Project Outline

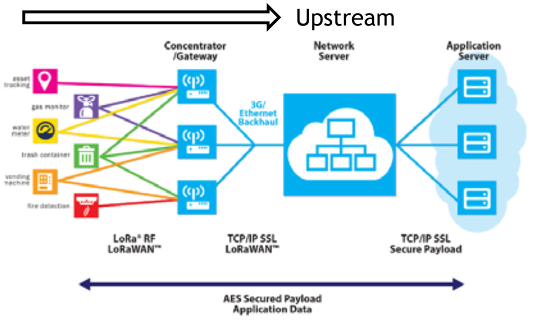
- Purpose
  - Earn real network protocol design and implementation experiences
  - Understand Low Power Wide Area (LPWA) protocol called LoRaWAN
- Team
  - 2-people teams
- Submission
  - To network_ta@popeye.snu.ac.kr
  - Mail subject : CN Term Project – Team number
- Question
  - Via E-mail

Project Milestones

- 11/1 (Wed) : Make teams and notify team members to TA (e-mail)
- 11/13 (Mon) : Progress Report 1 (Report)
  - Study LoRaWAN Specification and Source Code
    - Class A, Transmission parameters, OTAA join procedure
  - Project plan & LoRaWAN specification
- 11/22 (Wed): Progress Report 2 (Presentation)
  - Install and run LoRa end node, gateway, network server
  - Design beacon based bi-directional communications
- 12/06 (Wed): Progress Report 3 (Report)
  - Project status & source code
- 12/18 (Mon) : Final Report & Demo
  - Demo

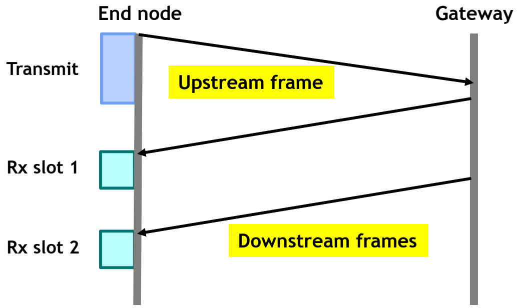
Background

- LoRa
  - Wireless technology for LPWA by Semtech
  - Defines physical layer
- LoRaWAN
  - Network protocol based on LoRa by LoRa Alliance
  - Defines MAC layer



LoRaWAN Class A Device

- LoRaWAN defines Class A/B/C devices
- Class A end node only supports limited half duplex communications
  - End nodes turn off communication unit to save energy
  - End nodes can initiate upstream frames as needed but gateway cannot trigger downstream communications
    - Downstream as a response to upstream



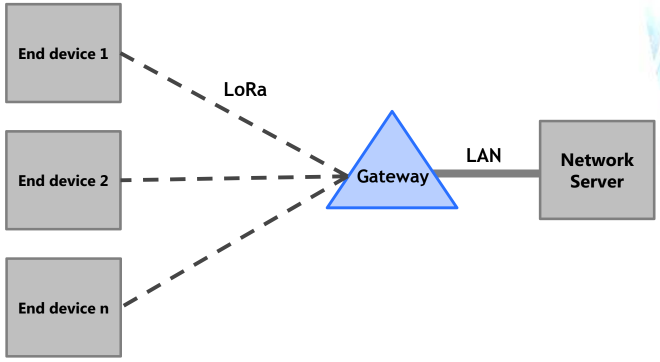
Goal

- Design and implement bi-directional communications
  - End-nodes can initiate upstream communication as needed
  - Enables downstream communications based on duty-cycles

  ➔ Similar to **WiFi PSM mechanism**

- Beacon
  - Gateway and end-nodes agree on beacon intervals
  - An end-node seeks beacon and wakes up every beacon interval
  - A gateway transmits beacons periodically to alert end-nodes with pending downstream frames
  - An end node with pending frames listens the medium until it receives frames or to the next beacon
    - Other nodes enter into sleep mode until the next beacon

Overview



* Data & Beacon reception

* Uplink transmission

* Forwarding data

* Beacon scheduling

* Downlink transmission

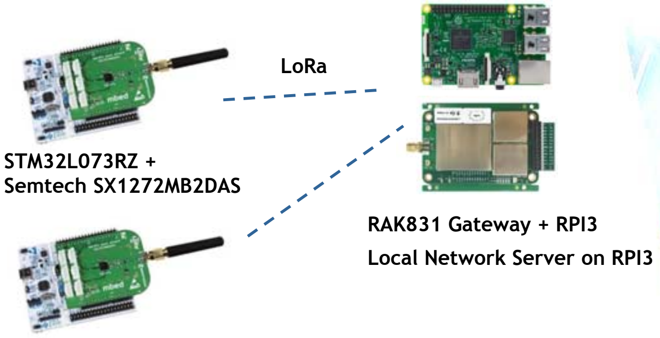
Specification

- End Node should
  - Use OTAA join procedure at first to join to a network server
  - Transmit data as needed
  - Wakeup & sleep periodically to listen beacons (duty-cycling)
  - Be ready to receive downstream data if it knows the network server has pending downstream frames
- Network Server should
  - Schedule periodic beacons
  - Transmit downstream data
  - Manage joined devices
- Beacon
  - Basic: Design beacon packet structure and a handshaking mechanism
    - Number of channels, which channel to use, …
  - Extra: Any performance enhancement schemes

Specification

- 1. End Node implementation
  - Modification of Class A source code
- 2. Network Server implementation
  - Devices for Network Server and gateway will be deployed in 302 bldg. 310-1
  - Use remote access (SSH, Web)
    - IP and port number will be announced
- 3. Gateway implementation
  - Packet Forwarder & Driver/HAL

Environment



LoRa

STM32L073RZ +
Semtech SX1272MB2DAS

RAK831 Gateway + RPI3

Local Network Server on RPI3

End node	Gateway	Network server
I-CUBE-LRWAN by semtech, ST	Packet forwarder by semtech	Open source LoRaWAN Network server
	HAL for gateway by semtech	

Deliverables

- Each Progress report according to the milestones
- Source codes of both end-node and network server implementation
- Final Report
  - Detailed Instruction of implementation
  - Performance evaluation
- DEMO
  - Will be announced later

Information

- Software will be uploaded on server
  - cn.snucse.org (147.46.242.74)
  - /home/FILES

● I-CUBE-LRWAN
en.i-cube_lrwan.zip

● DFP for STM32L0
Keil.STM32L0xx_DFP.1.6.1.pack

● Gateway configuration file (KR channel support)
global_conf.json

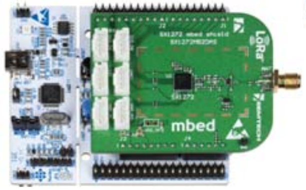
● LoRaWAN spec 1.0.2 & LoRaWAN Regional Parameter 1.0.2
LoRaWAN102-20161012_1398_1.pdf
LoRaWANRegionalParametersv1.0.2_final_1944_1.pdf

● Gateway reset source (using wiringPi for GPIO control)
reset.c

● ST Utility
STM32 ST-LINK Utility.zip

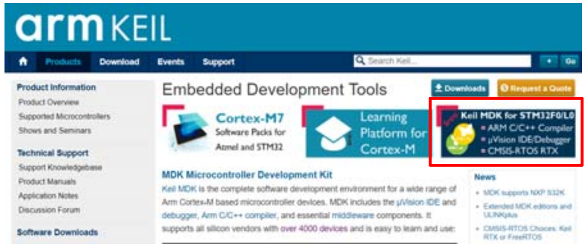
End Node Implementation

- Platform
  - STM32L073RZ + SX1272mb2das
- Open software
  - I-CUBE-LRWAN by ST, Semtech
  - LoRaWAN endpoint stack implementation and example projects supporting STM32L073RZ
- Development toolchains
  - ARM Keil
- Virtual COM port
  - Tera Term



Development tool chain

- ARM KEIL
  - C compiler for micro controller
  - Only support Windows OS
  - Free license for our device STM32L073RZ



Product Information
- Product Overview
- Supported Microcontrollers
- Shows and Seminars

Technical Support
- Support Knowledgebase
- Product Manuals
- Application Notes
- Discussion Forum

Software Downloads

Embedded Development Tools

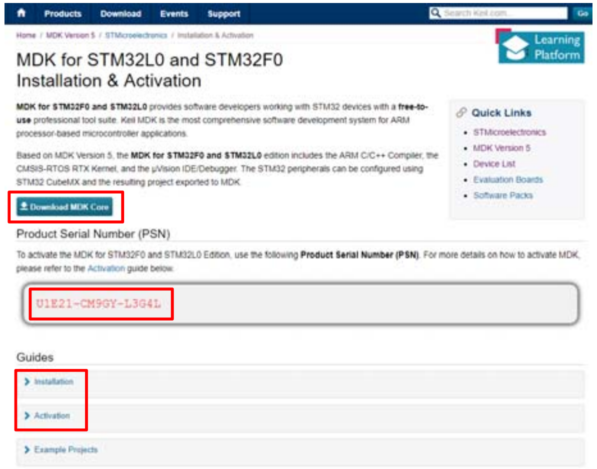
MDK Microcontroller Development Kit

Keil MDK is the complete software development environment for a wide range of Arm Cortex-M based microcontroller devices. MDK includes the µVision IDE and debugger, Arm C/C++ compiler, and essential middleware components. It supports all silicon vendors with over 4000 devices and is easy to learn and use.

News
- MDK supports NXP S32K
- Extended MDK editions and ULINKplus
- CMSIS-RTOS Choices: Keil RTX or FreeRTOS

Development tool chain

- You can download software and get license



MDK for STM32L0 and STM32F0
Installation & Activation

MDK for STM32F0 and STM32L0 provides software developers working with STM32 devices with a **free-to-use** professional tool suite. Keil MDK is the most comprehensive software development system for ARM processor-based microcontroller applications.

Based on MDK Version 5, the **MDK for STM32F0 and STM32L0** edition includes the ARM C/C++ Compiler, the CMSIS-RTOS RTX Kernel, and the µVision IDE/Debugger. The STM32 peripherals can be configured using STM32 CubeMX and the resulting project exported to MDK.

Quick Links
- STMicroelectronics
- MDK Version 5
- Device List
- Evaluation Boards
- Software Packs

Product Serial Number (PSN)

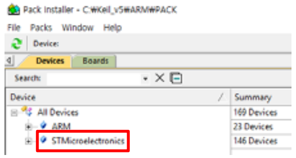
To activate the MDK for STM32F0 and STM32L0 Edition, use the following **Product Serial Number (PSN)**. For more details on how to activate MDK, please refer to the Activation guide below.

```
U1E21-CM9GY-L3G4L
```

Guides
- Installation
- Activation
- Example Projects

Environment Setup

- After getting license, you can use KEIL IDE for developing end node's firmware
- KEIL will try to download devices DFP automatically when it starts
  - If pack installer has no STMicroelectronics option, you have to install DFP directly
  - Install file is on the server
    - Keil.STM32L0xx_DFP.1.6.1.pack

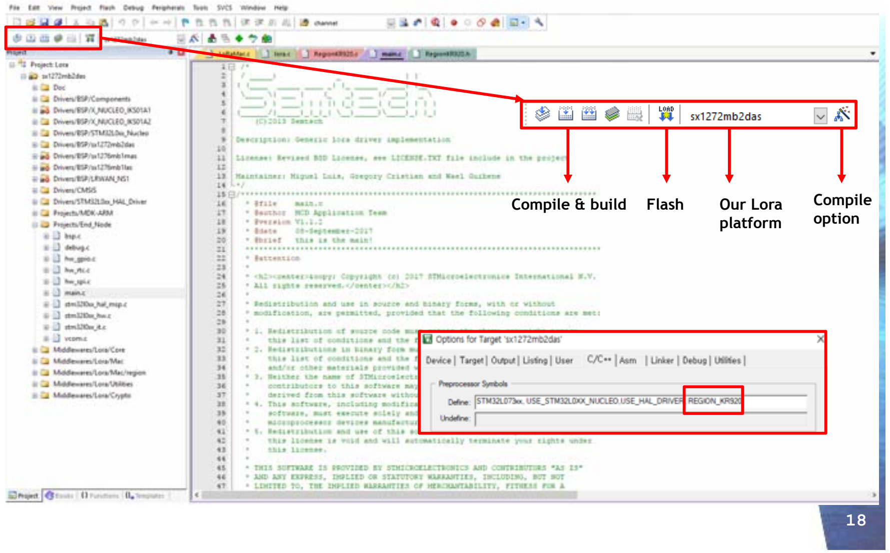


Device	Summary
All Devices	169 Devices
ARM	23 Devices
STMicroelectronics	146 Devices

Source compile & Flashing

- KEIL project file for a LoRaWAN class A application is available on directory below
  - en.icube_lrwan\STM32CubeExpansion_LRWAN_V1.1.2\Projects \Multi\Applications\LoRa\End_Node\MDK-ARM\STM32L073RZ-Nucleo
  - en.icube_lrwan is on the server
    - en.i-cube_lrwan.zip
- Manual about source codes is available by ST
  - http://www.st.com/content/ccc/resource/technical/document/user_manual/group0/31/96/2f/3b/df/c1/40/2e/DM00300436/files/DM00300436.pdf/jcr:content/translations/en.DM00300436.pdf

Source compile & Flashing



```
/*
 (C)2013 Semtech

Description: Generic lora driver implementation

License: Revised BSD License, see LICENSE.TXT file include in the project

Maintainer: Miguel Luis, Gregory Cristian and Wael Guibene
*/
/******************************************************************************
  * @file    main.c
  * @author  MCD Application Team
  * @version V1.1.2
  * @date    08-September-2017
  * @brief   this is the main!
  ******************************************************************************
  * @attention
  *
  * <h2><center>&copy; Copyright (c) 2017 STMicroelectronics International N.V.
  * All rights reserved.</center></h2>
  *
  * Redistribution and use in source and binary forms, with or without
  * modification, are permitted, provided that the following conditions are met:
  *
  * 1. Redistribution of source code mu
  *    this list of conditions and the f
  * 2. Redistributions in binary form mu
  *    this list of conditions and the f
  *    and/or other materials provided w
  * 3. Neither the name of STMicroelectr
  *    contributors to this software may
  *    derived from this software withou
  * 4. This software, including modifica
  *    software, must execute solely and
  *    microprocessor devices manufactur
  * 5. Redistribution and use of this so
  *    this license is void and will automatically terminate your rights under
  *    this license.
  *
  * THIS SOFTWARE IS PROVIDED BY STMICROELECTRONICS AND CONTRIBUTORS "AS IS"
  * AND ANY EXPRESS, IMPLIED OR STATUTORY WARRANTIES, INCLUDING, BUT NOT
  * LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A
```

Options for Target 'sx1272mb2das'

Device | Target | Output | Listing | User | C/C++ | Asm | Linker | Debug | Utilities |

Preprocessor Symbols

Define: STM32L073xx, USE_STM32L0XX_NUCLEO,USE_HAL_DRIVER REGION_KR920

Undefine:

Compile & build

Flash

Our Lora platform

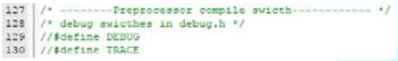
Compile option

Source compile & Flashing

- Install STM32 ST-LINK Utility.zip for device recognition
  - STM32 ST-LINK Utility.zip is uploaded on server
- Connect your device through cable and click FLASH button to flash your hex file
  - Board's LED will be blinking indicating it is downloading the firmware
  - You can restart your device using reset button

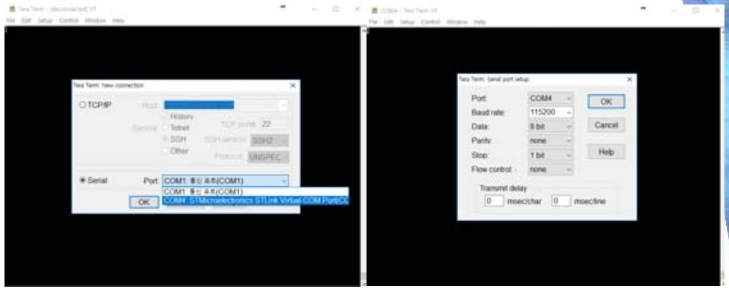
Virtual Comport

- Tera Term
  - Tool to see what's going on in your device
  - Select COM# for ST device
  - Setup Baudrate
    - Setup -> Serial port



```
/* ---------Preprocessor compile swicth------------- */
/* debug swicthes in debug.h */
//#define DEBUG
//#define TRACE
```

- You can activate debug mode in hw_conf.h file



Tera Term: New connection

○ TCP/IP Host: History Service: Telnet SSH Other TCP port#: 22 SSH version: SSH2 Protocol: UNSPEC

● Serial Port: COM1: 통신 포트(COM1)

COM1: 통신 포트(COM1)

COM4: STMicroelectronics STLink Virtual COM Port(CO

Tera Term: Serial port setup

Port: COM4

Baud rate: 115200

Data: 8 bit

Parity: none

Stop: 1 bit

Flow control: none

Transmit delay: 0 msec/char 0 msec/line

OK
Cancel
Help
Tera Term - [disconnected] VT
COM4 - Tera Term VT
File Edit Setup Control Window Help

Gateway & Network Server Implementation

- Raspberry Pi 3 model B + RAK831(SX1301)
  - 1.2GHz 64-bit quad-core ARM Cortex-A53 CPU (BCM2837)
  - Raspbian Jessie OS which is based on Linux will be used
- Install wiringPi

```
apt-get install wiringpi
```

  - Compile reset.c with –lwiringPi option
  - For resetting RAK831

Gateway implementation

- Install git

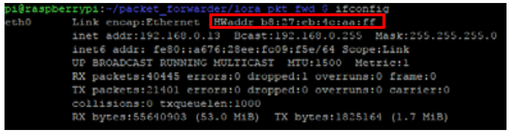
```
sudo apt-get update
```

```
apt-get install git
```

- To make RPI to act as a LoRaWAN gateway, two stacks are needed
  - Packet forwarder
  - HAL (Hardware Abstraction Layer) for SX1301
- You can download each source form github
  - Use git clone command
    - ● **LoRaWAN gateway HAL**
      https://github.com/Lora-net/lora_gateway
    - ● **LoRaWAN packet forwarder**
      https://github.com/Lora-net/packet_forwarder

Configuration of Channel Frequency

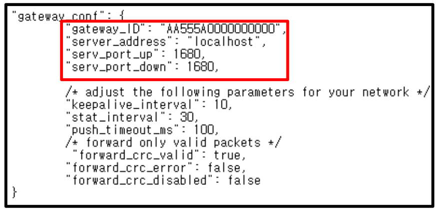
- Gateway configuration file
  - You should change configuration file for KR channel and your own network server ip address
    - packet_forwarder/lora_pkt_fwd/global_conf.json
  - global_conf.json for KR channel is already uploaded on the server
- Gateway address for network server
  - Use ifconfig command to get eth0 mac address
  - Transform your own mac address to EUI-64 form
    - You can find such calculator on internet



```
pi@raspberrypi:~/packet_forwarder/lora_pkt_fwd $ ifconfig
eth0      Link encap:Ethernet  HWaddr b8:27:eb:4c:aa:ff
          inet addr:192.168.0.13  Bcast:192.168.0.255  Mask:255.255.255.0
          inet6 addr: fe80::a676:28ee:fc09:f5e/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:40445 errors:0 dropped:1 overruns:0 frame:0
          TX packets:21401 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:55640903 (53.0 MiB)  TX bytes:1825164 (1.7 MiB)
```

Configuration of Channel Frequency

- Change your gateway_ID in global_conf.json file to your own EUI-64 form mac address



```
"gateway_conf": {
        "gateway_ID": "AA555A0000000000",
        "server_address": "localhost",
        "serv_port_up": 1680,
        "serv_port_down": 1680,

        /* 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
}
```

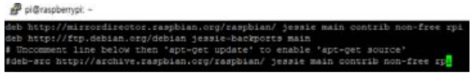
Network Server implementation

- LoRaWAN Network Server
  - Opensource LoRaWAN Network Server can be downloaded on below github repo
  - Server is based on Erlang language

● Opensource LoRaWAN network server
https://github.com/gotthardp/lorawan-serv

- Erlang OTP installation

```
add deb http://ftp.debian.org/debian jessie-backports main to /etc/apt/sources.list
```



pi@raspberrypi: ~

```
deb http://mirrordirector.raspbian.org/raspbian/ jessie main contrib non-free rpi
deb http://ftp.debian.org/debian jessie-backports main
# Uncomment line below then 'apt-get update' to enable 'apt-get source'
#deb-src http://archive.raspbian.org/raspbian/ jessie main contrib non-free rpi
```

```
sudo apt-get update
sudo apt-get -t jessie-backports install erlang
```

- For compiling & developing, npm is required

```
sudo wget http://node-arm.herokuapp.com/node_latest_armhf.deb
sudo dpkg -i node_latest_armhf.deb
```

Network Server implementation

- Already compiled Network Server Debian package

```
wget https://github.com/gotthardp/lorawan-server/releases/download/v0.4.12/lorawan-server_0.4.12_all.deb
```

```
sudo dpkg -i lorawan-server_0.4.12_all.deb
```

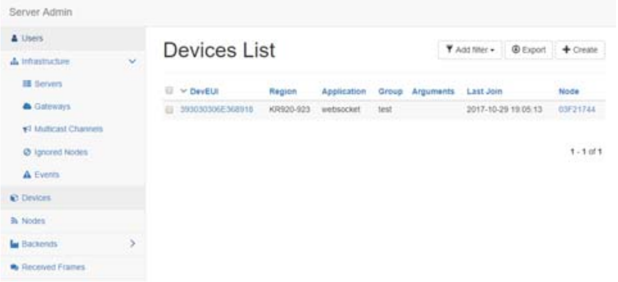
- For compiling & making new Debian package, see **Build Instructions** guide
  - https://github.com/gotthardp/lorawan-server/blob/master/doc/Installation.md

Network Server Admin Web UI

- You can start the server

```
systemctl start lorawan-server
```

- Network server provides admin page for registering & managing devices and monitoring packets
- Gateway information and node information should be registered in server before deploying network



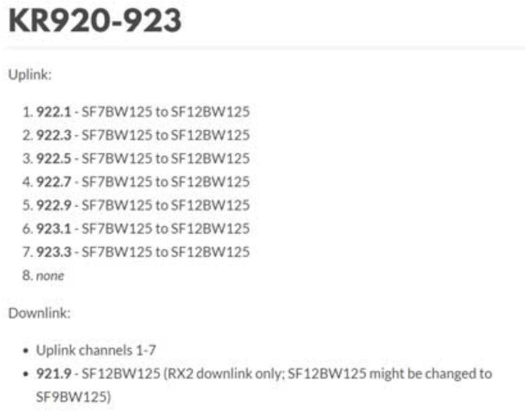
Devices List

DevEUI	Region	Application	Group	Arguments	Last Join	Node
393030306E368918	KR920-923	websocket	test		2017-10-29 19:05:13	03F21744

1 - 1 of 1

Appendix

- Korea Frequency Channel Plan



KR920-923

Uplink:

1. **922.1** - SF7BW125 to SF12BW125
2. **922.3** - SF7BW125 to SF12BW125
3. **922.5** - SF7BW125 to SF12BW125
4. **922.7** - SF7BW125 to SF12BW125
5. **922.9** - SF7BW125 to SF12BW125
6. **923.1** - SF7BW125 to SF12BW125
7. **923.3** - SF7BW125 to SF12BW125
8. *none*

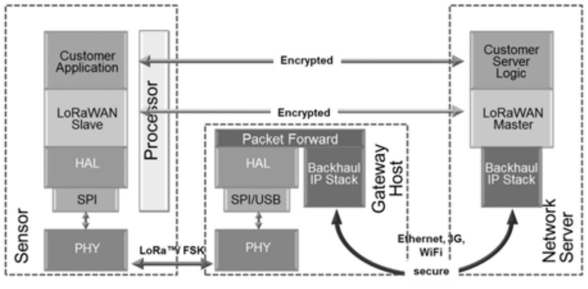
Downlink:

- Uplink channels 1-7
- **921.9** - SF12BW125 (RX2 downlink only; SF12BW125 might be changed to SF9BW125)

Cited from TheThingsNetwork
https://www.thethingsnetwork.org/wiki/LoRaWAN/Frequencies/Frequency-Plans

Appendix

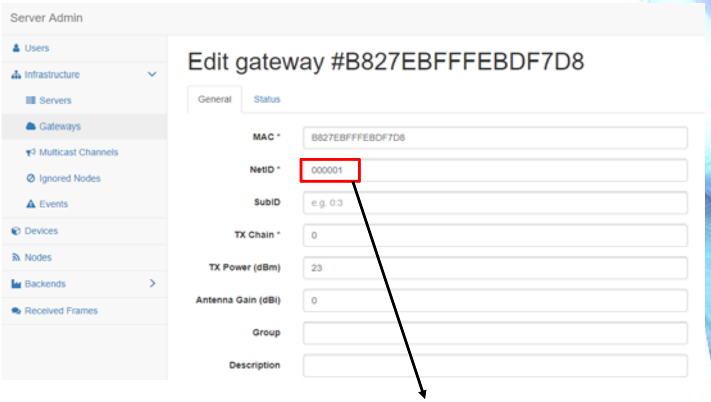
- LoRaWAN architecture



Cited from LoRa Alliance
https://www.lora-alliance.org/technology

Appendix

- Gateway registration on server admin web UI



Edit gateway #B827EBFFFEBDF7D8

MAC * B827EBFFFEBDF7D8

NetID * 000001

SubID e.g. 0:3

TX Chain * 0

TX Power (dBm) 23

Antenna Gain (dBi) 0

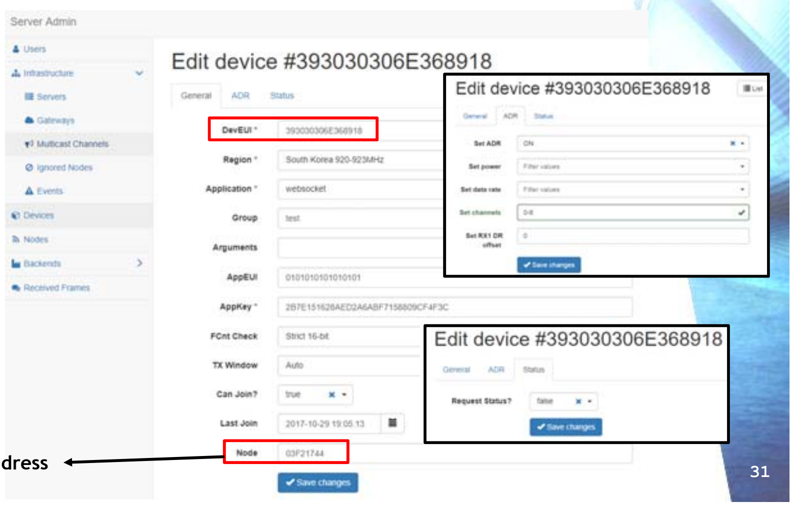
Group

Description

Private network server ID

Appendix

- Node registration on server admin web UI



Edit device #393030306E368918

DevEUI * 393030306E368918

Region * South Korea 920-923MHz

Application * websocket

Group test

Arguments

AppEUI 0101010101010101

AppKey * 2B7E151628AED2A6ABF7158809CF4F3C

FCnt Check Strict 16-bit

TX Window Auto

Can Join? true

Last Join 2017-10-29 19:05:13

Node 03F21744

✔ Save changes

Edit device #393030306E368918

Set ADR ON

Set power Filter values

Set data rate Filter values

Set channels 0-8

Set RX1 DR offset 0

✔ Save changes

Edit device #393030306E368918

Request Status? false

✔ Save changes

Device Address

Reference

- LoRaWAN gateway HAL
  - https://github.com/Lora-net/lora_gateway
- LoRaWAN packet forwarder
  - https://github.com/Lora-net/packet_forwarder
- Opensource LoRaWAN network server
  - https://github.com/gotthardp/lorawan-server
- I-CUBE-LRWAN
  - http://www.st.com/en/embedded-software/i-cube-lrwan.html