LG01N/OLG01N LoRa Gateway User Manual

Document Version: 1.2.1

Firmware Version: LG02_LG08—v5.1.15

Version	Description	Date
1.0	Release	2018-Dec-28
1.1	Add Customized Script Feature (firmware ver >LG02_LG08--build-v5.1.1547896817-20190119-1921)	2019-Jan-19
1.2	--Add Downlink support and example. (firmware ver >LG02_LG08--build-v5.1.1548820215-20190130-1151) --Correct typo for the UNO code of example for lg02_single_rx_tx	2019-Jan-30
1.2.1	-- Add OLG01 connector photo -- Add how to control LEDs -- Modify MQTT instruction	2019-Jun-19
1.2.2	--Add photo for OLG01 4G installation	2019-Nov-1
1.2.3	-- Change the HTTP Port and SSH port for firmware version > v5.3	2019-Nov-26

1. Introduction

1.1 What is LG01N & OLG01N

LG01N & OLG01N are an open source **single channel LoRa Gateway**. It lets you bridge LoRa wireless network to an IP network via WiFi, Ethernet, 3G or 4G cellular. The LoRa wireless allows users to send data and reach extremely long ranges at low data-rates. It provides ultra-long range spread spectrum communication and high interference immunity.

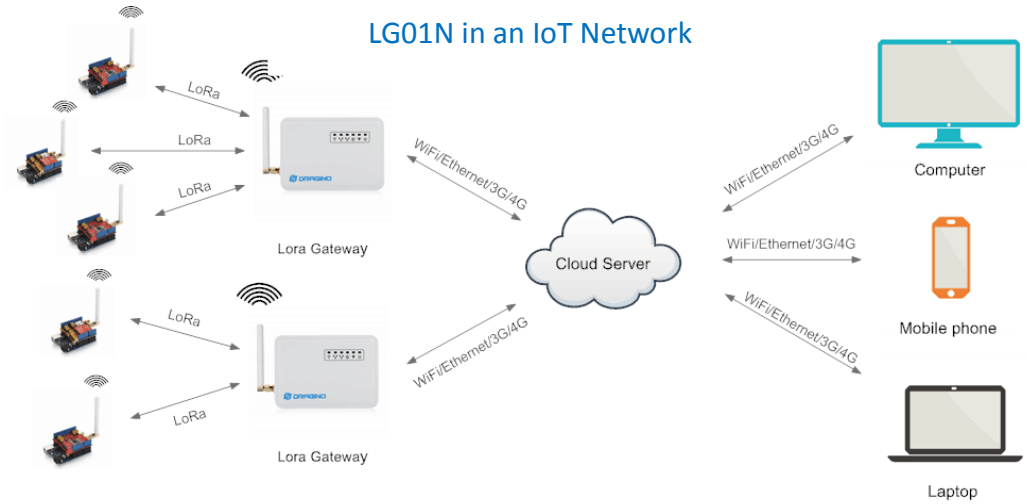
LG01N & OLG01N have rich internet connection method such as **WiFi interface, Ethernet port and 3G/4G Cellular**. These Interfaces provide flexible methods for users to connect their sensor networks to Internet.

LG01N & OLG01N can support the LoRaWAN protocol in single frequency and customized LoRa transition protocol.

LG01N can be used to provide a low cost IoT wireless solution to support 50~100 sensor nodes.

Except limited LoRaWAN mode, LG01N can support multiply working mode such as: MQTT mode, TCP/IP Client mode to fit different requirement for IoT connection.

LG01N & OLG01N provide a low cost for your IoT network connection. Compare to the cost with normal SX1301 LoRaWAN solution. LG01N & OLG01N is only of its 1/4 or less cost. This makes the LG01N very suitable to set up small scale LoRa network or use it to extend the coverage of current LoRaWAN network.



1.2 Specifications

Hardware System:

Linux Part:

- 400Mhz ar9331 processor
- 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

WiFi Spec:

- IEEE 802.11 b/g/n
- Frenquency Band: 2.4 ~ 2.462GHz
- Tx power:
  - ✓ 11n tx power : mcs7/15: 11db mcs0 : 17db
  - ✓ 11b tx power: 18db
  - ✓ 11g 54M tx power: 12db
  - ✓ 11g 6M tx power: 18db
- Wifi Sensitivity
  - ✓ 11g 54M : -71dbm
  - ✓ 11n 20M : -67dbm

LoRa Spec:

- Frequency Range:
  - ✓ Band 1 (HF): 862 ~ 1020 Mhz
  - ✓ Band 2 (LF): 410 ~ 528 Mhz
- 168 dB maximum link budget.
- +20 dBm - 100 mW constant RF output vs.
- +14 dBm high efficiency PA.
- Programmable bit rate up to 300 kbps.
- High sensitivity: down to -148 dBm.
- Bullet-proof front end: IIP3 = -12.5 dBm.
- Excellent blocking immunity.
- Low RX current of 10.3 mA, 200 nA register retention.
- Fully integrated synthesizer with a resolution of 61 Hz.
- FSK, GFSK, MSK, GMSK, LoRaTM and OOK modulation.

- Built-in bit synchronizer for clock recovery.
- Preamble detection.
- 127 dB Dynamic Range RSSI.
- Automatic RF Sense and CAD with ultra-fast AFC.
- Packet engine up to 256 bytes with CRC.
- Built-in temperature sensor and low battery indicator.

Cellular 4G LTE (optional):

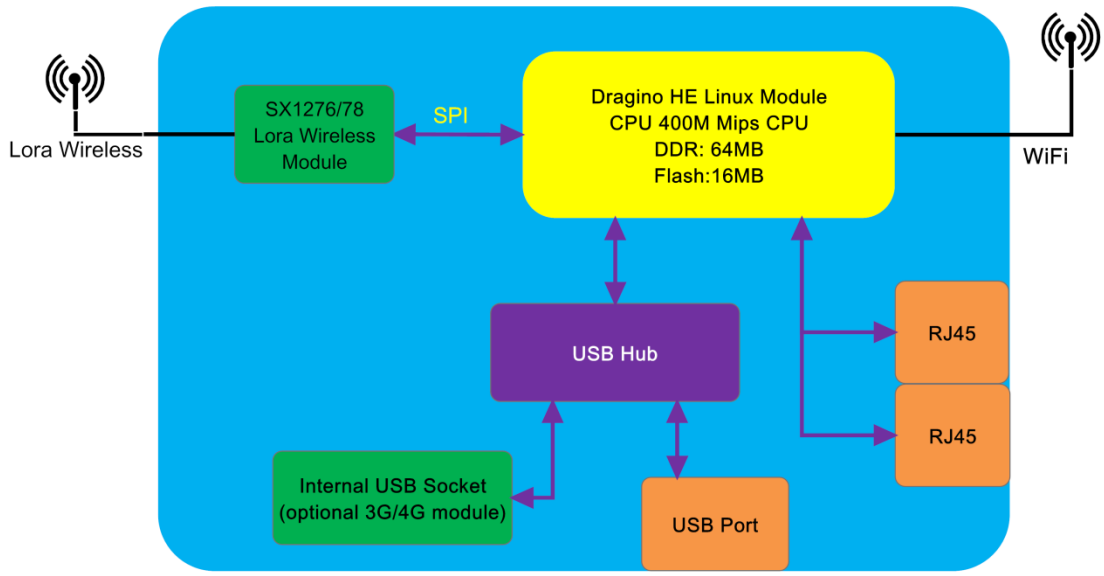
- Quectel **EC25 LTE module**
- Micro SIM Slot
- Internal 4G Antenna + External 4G Sticker Antenna.
- Up to 150Mbps downlink and 50Mbps uplink data rates
- Worldwide LTE,UMTS/HSPA+ and GSM/GPRS/EDGE coverage
- MIMO technology meets demands for data rate and link reliability in modem wireless communication systems

1.3 Features

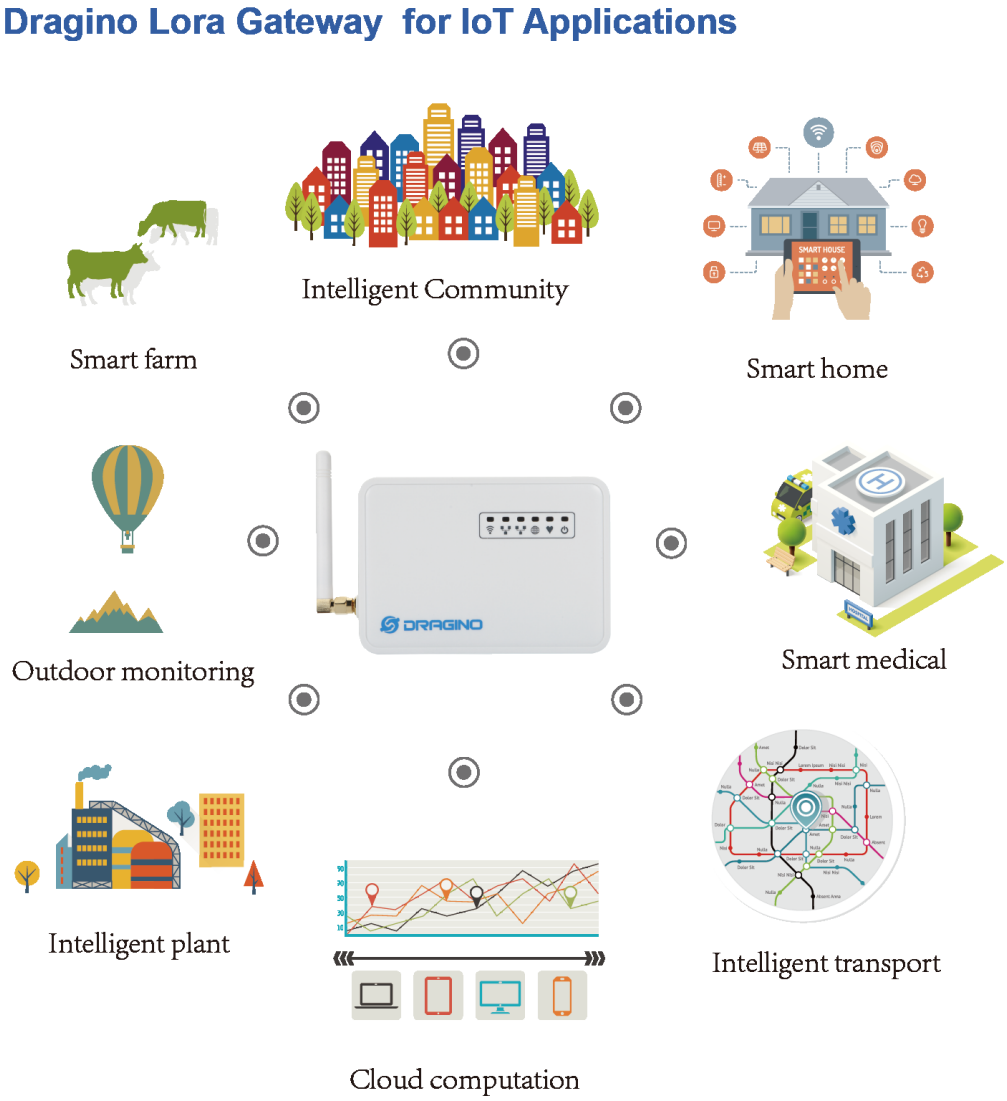
- ✓ Open Source OpenWrt LEDE system
- ✓ Low power consumption
- ✓ Firmware upgrade via Web
- ✓ Software upgradable via network
- ✓ Flexible protocol to connect to IoT servers
- ✓ Auto-Provisioning
- ✓ Built-in web server
- ✓ Managed by Web GUI, SSH via LAN or WiFi
- ✓ Internet connection via LAN, WiFi, 3G or 4G
- ✓ Failsafe design provides robustly system
- ✓ 1 x SX1276/SX1278 LoRa modules
- ✓ Full – duplex LoRa transceiver
- ✓ Two receive channels, and one transmit channel
- ✓ Limited support in LoRaWAN/ Support Private LoRa protocol
- ✓ Support upto 100 nodes
- ✓ LoRa band available at 433/868/915/920 Mhz
- ✓ Max range in LoRa: 5~10 km. Density Area:>500m

1.4 System Structure

LG01N System Overview:



1.5 Applications



1.6 Hardware Variants

The LG01N and OLG01N use the same firmware and have the same feature in the software side. In this document, we will use LG01N as the model number to explain the feature.

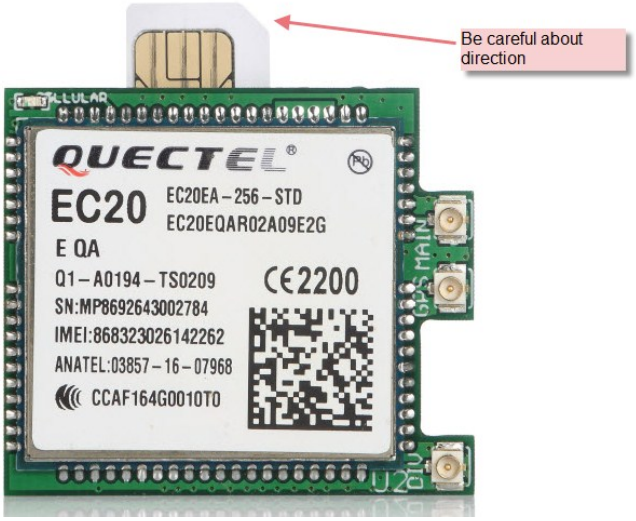
Model	Photo	Description
LG01N		Indoor version for single channel LoRa Gateway,
OLG01N		Outdoor version for dual channel LoRa Gateway

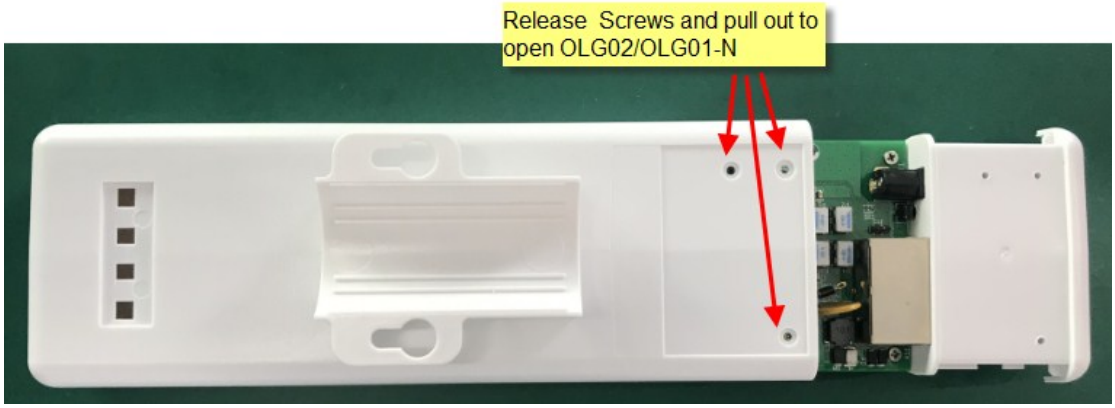
1.7 Interfaces

OLG01N Version Interface:

1.8 Install SIM card in 4G module

LG01N & OLG01N has optional built-in 4G module version. For the 4G version, devices will be shipped with screws un assembly, please open the box and use below direction to install the SIM card (Micro SIM)





1.9 Firmware Change log

Please see this link for firmware change log:
http://www.dragino.com/downloads/index.php?dir=LoRa_Gateway/LG02-OLG02/Firmware/&file=ChangeLog

2. Access LG01N

Access and configure LG01

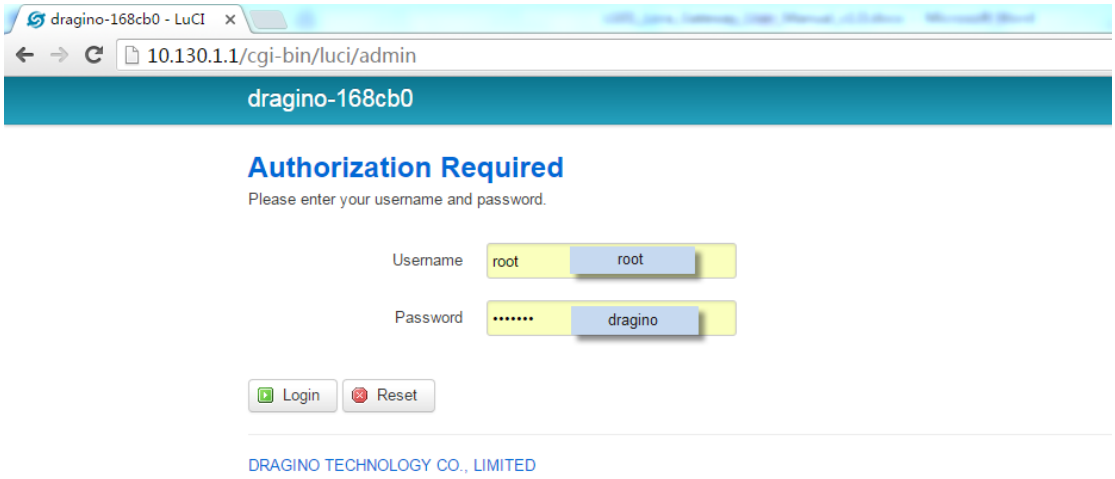
The LG01N is configured as a WiFi AP by factory default. User can access and configure the LG01N after connect to its WiFi network.

At the first boot of LG01N, it will auto generate an unsecure WiFi network call dragino-xxxxxx

User can use the laptop to connect to this WiFi network. The laptop will get an IP address 10.130.1.xxx and the LG01 has the default IP 10.130.1.1

Open a browser in the laptop and type
http://10.130.1.1/cgi-bin/luci/admin
User will see the login interface of LG01N.
The account for Web Login is:
User Name: root
Password: dragino

Note: the LG01 can also be accessed via WAN interface (WAN port or WiFi when device acts as WiFi Client). But for security reason, for firmware version >5.3, the http access on WAN interface has been set to 8000, SSH access has been set to 2222.



Notice: In case the WiFi network is disabled, user can connect the PC to LG01N's LAN port, the PC will get DHCP from LG01N, and be able to access it.

3. Typical Network Setup

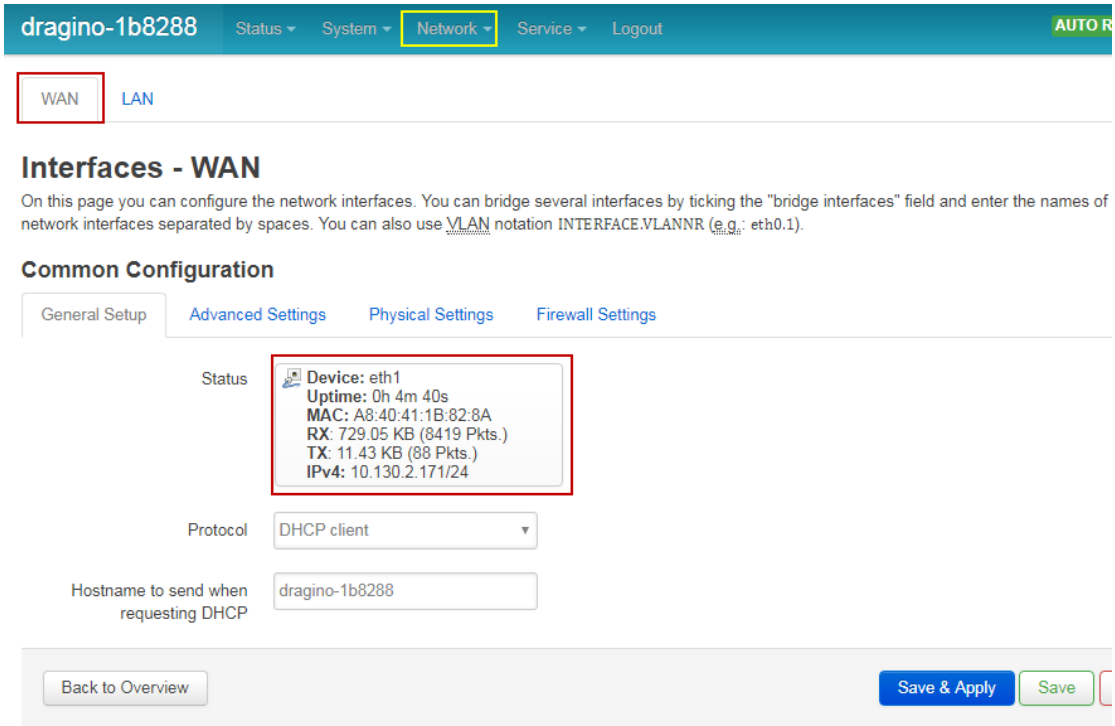
3.1 Overview

LG01N supports flexible network set up for different environment. This section describes the typical network topology can be set in LG01N. The typical network set up includes:

- ✓ **WAN Port Internet Mode**
- ✓ **WiFi Client Mode**
- ✓ **WiFi AP Mode**
- ✓ **USB Dial Up Mode**

3.2 Use WAN port to access Internet

By default, the LG01N set to use WAN port as network connection. When connect LG01N's WAN port to router, LG01N will get IP from router and have internet access. The network status is as below:



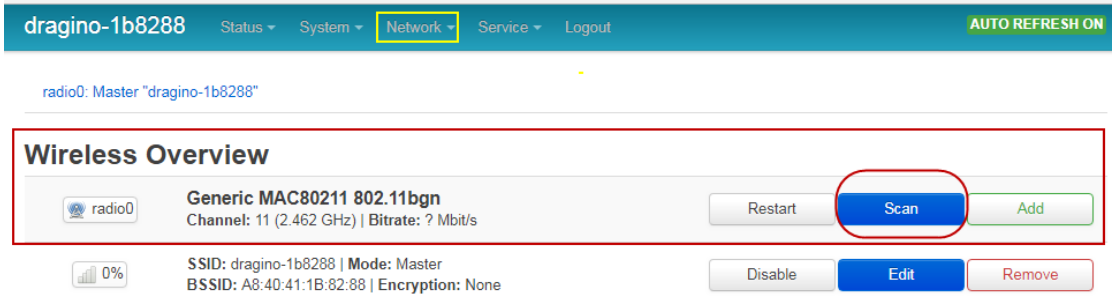
3.3 Access Internet as a WiFi Client.

In the WiFi Client Mode, Dragino acts as a WiFi client and gets IP from uplink router via WiFi.

The step to set is as below:

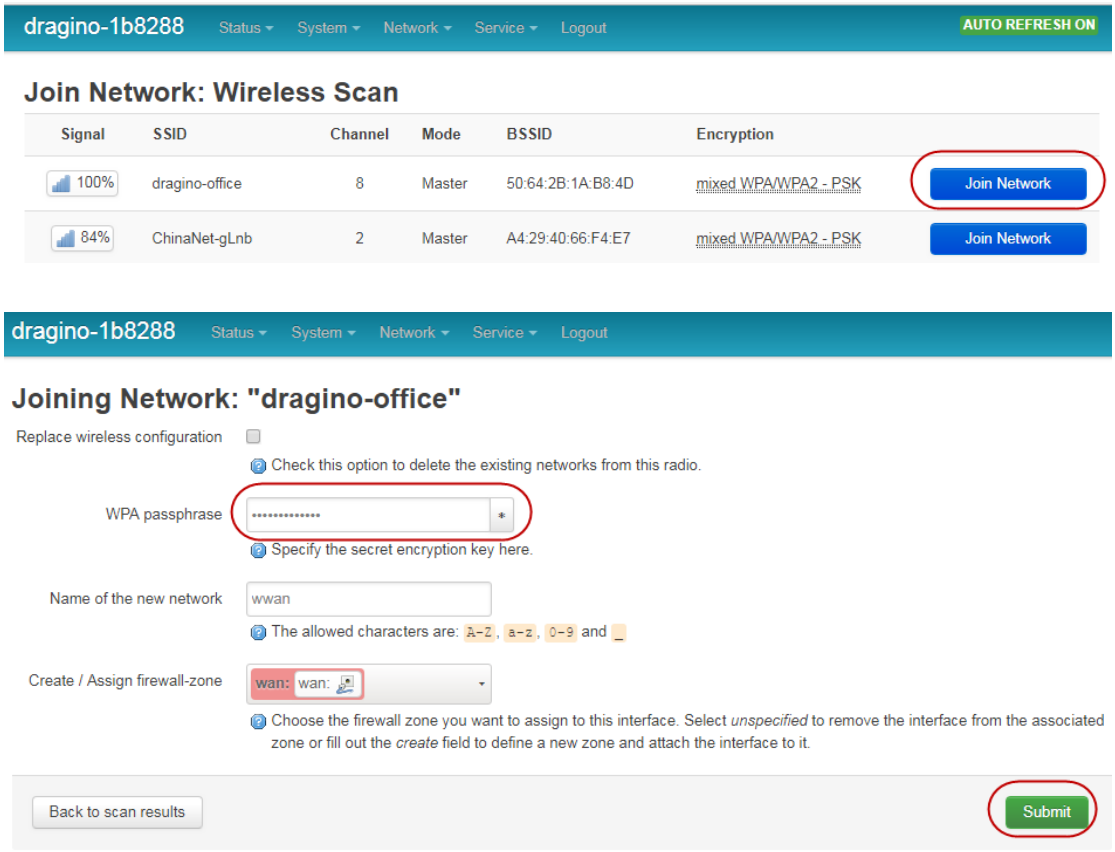
Step1:

In network -> Wireless, select Radio0 interface and scan.



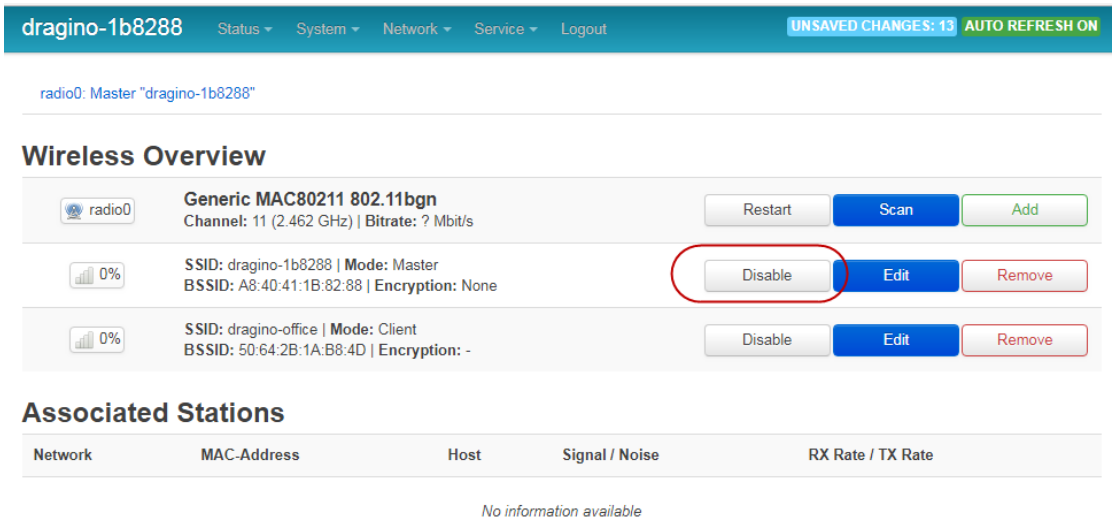
Step2:

Select the wireless AP and join:



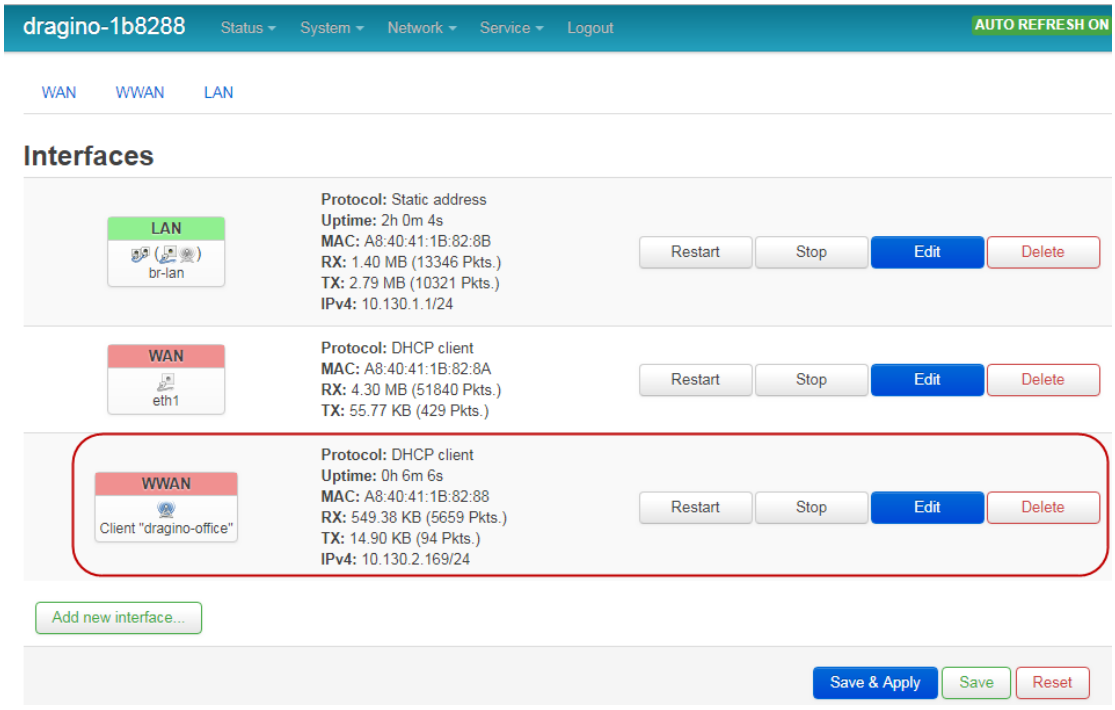
Step3:

In network->wireless page, disable WiFi AP network. Notice: After doing that, you will lose connection if your computer connects to the LG01N via LG01N's wifi network.



(Note:make sure click the Save & Apply after configure)

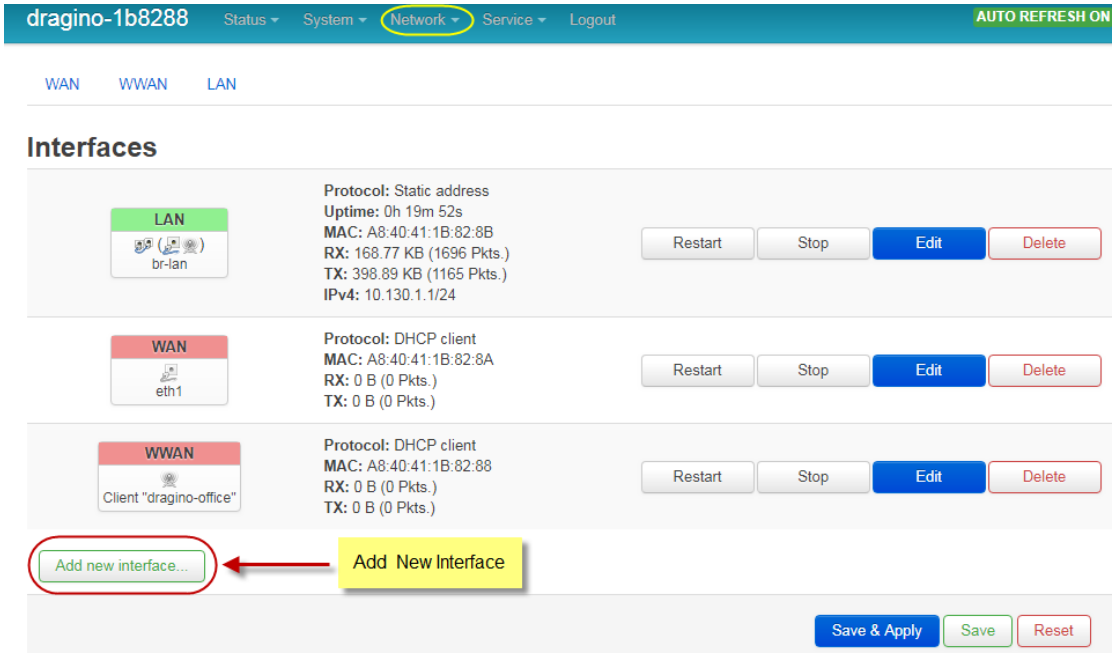
After successful associate, the WiFi network interface can be seen in the same page:

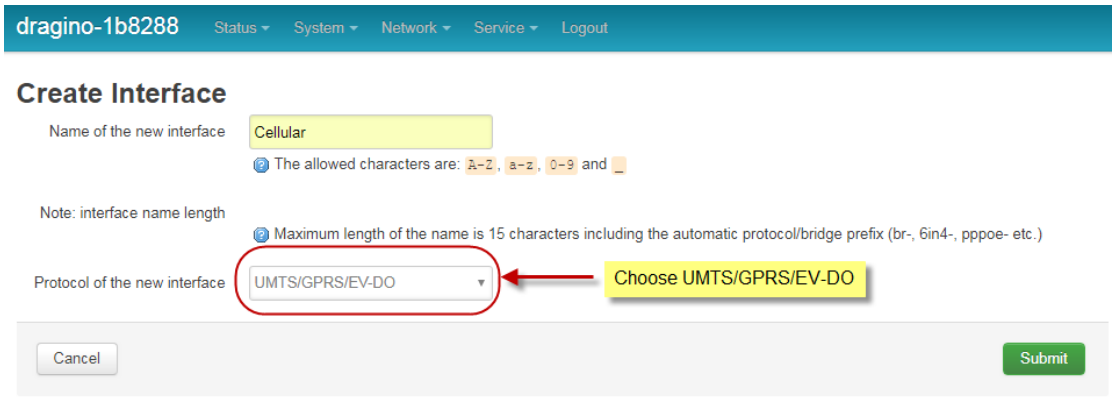


3.4 Use built-in 4G modem for internet access

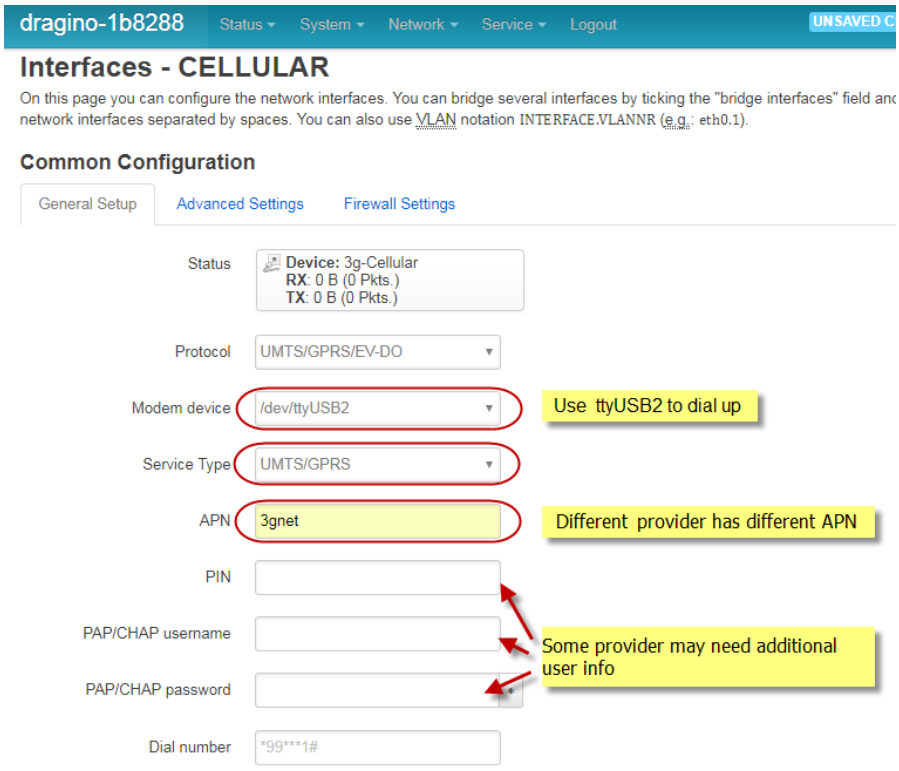
For the LG01N with built-in 4G version, user can configure the modem for internet access.

Step 1: Add New Interface

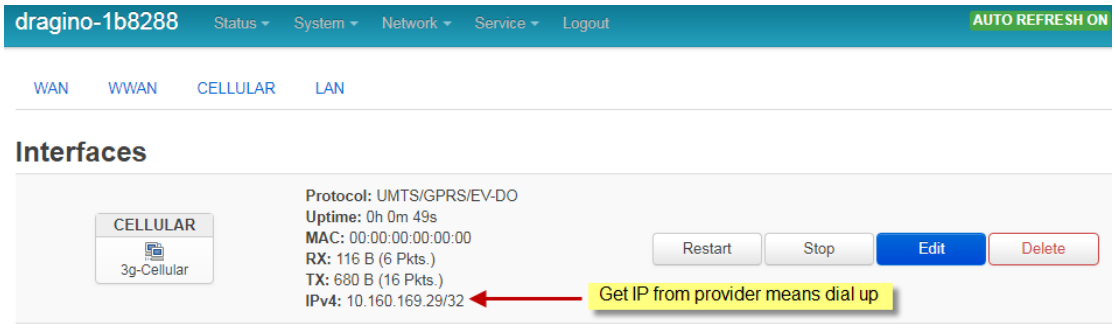




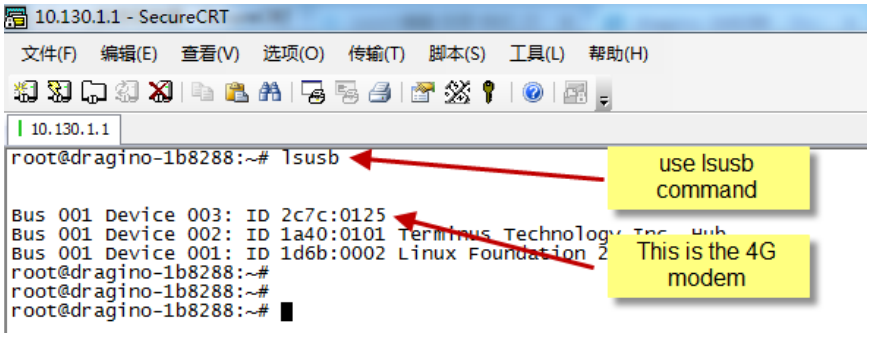
Step 2: Configure cellular interface



Step 3: Check Result

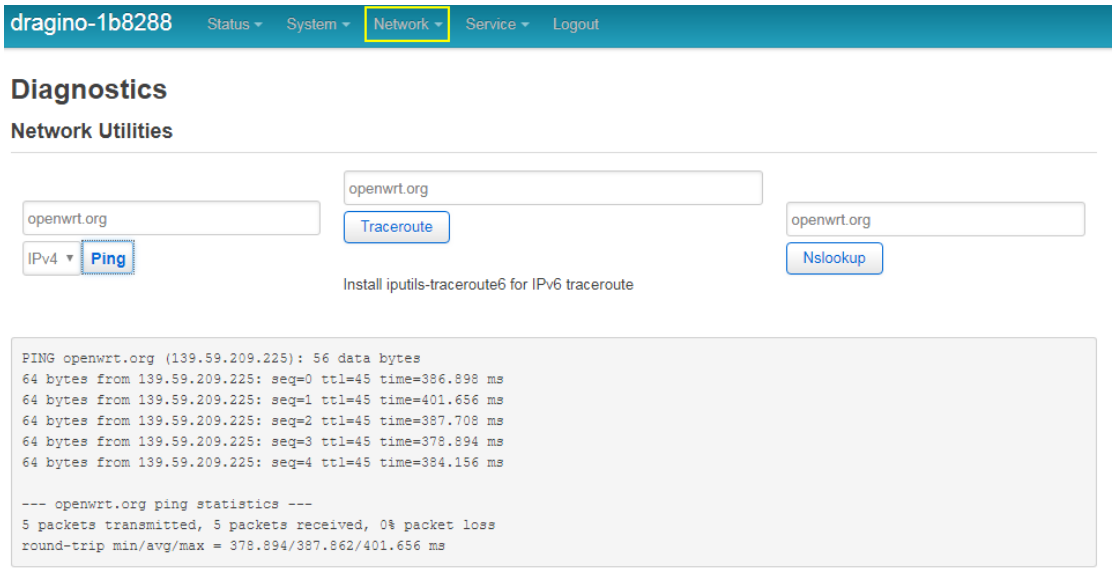


Note: In case you don't know if your device has 4G modem, you can run lsusb command in SSH access to check, as below:



3.5 Check Internet connection

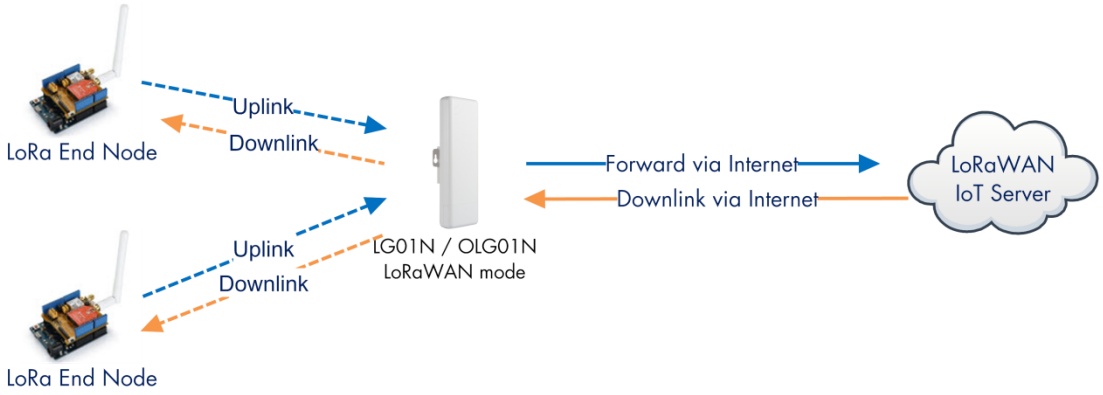
User can use the diagnostics page to check and analyze Internet connection.



4. Example 1: Configure as a LoRaWAN gateway – Limited LoRaWAN mode

LoRaWAN mode:

Use LG01N / OLG01N as a LoRaWAN gateway* to forward packet to LoRaWAN IoT Server



Operate Principle:

- LG01N/OLG01N running packet forward and will forward the uplink LoRa packet from end node to LoRaWAN server.
- It will also forward downlink LoRa packet from LoRaWAN server to end node.
- The end node can use OTAA or ABP mode in the LoRaWAN protocol.

Limitation:

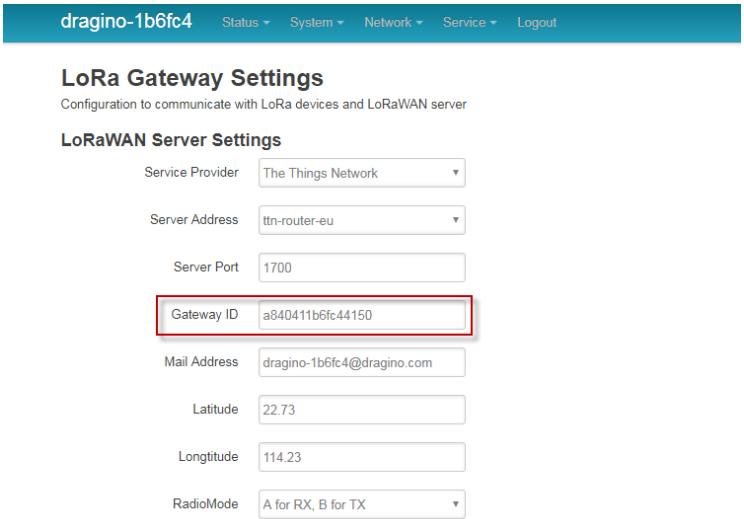
- The LG01 only support one LoRaWAN frequency for uplink. So the end node should be set to fix frequency.
- If end node use muliply frequencies to transfer, The LG01 will only be able to receive the same frequency set in LG01N.

This chapter describes how to use LG01N to work with TTN LoRaWAN Server. The method to work with other LoRaWAN Server is similar.

4.1 Create a gateway in TTN Server

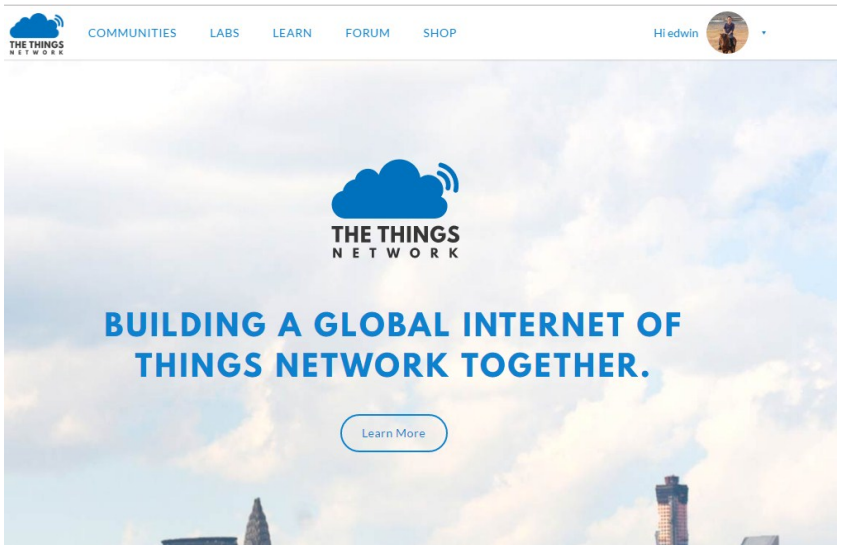
Step 1: Get a Unique gateway ID.

Every LG01N has a unique gateway id. The id can be found at LoRaWAN page:

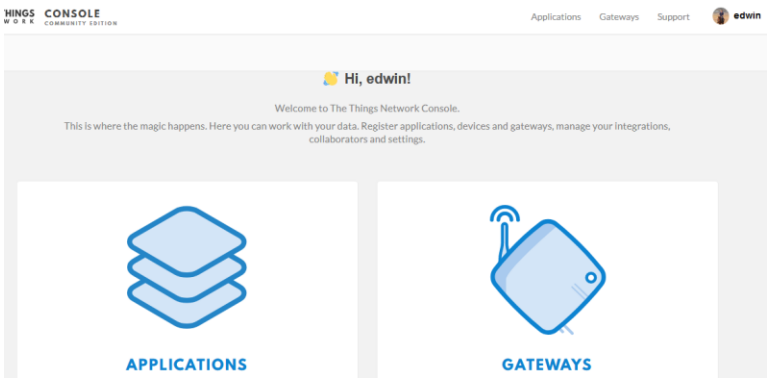


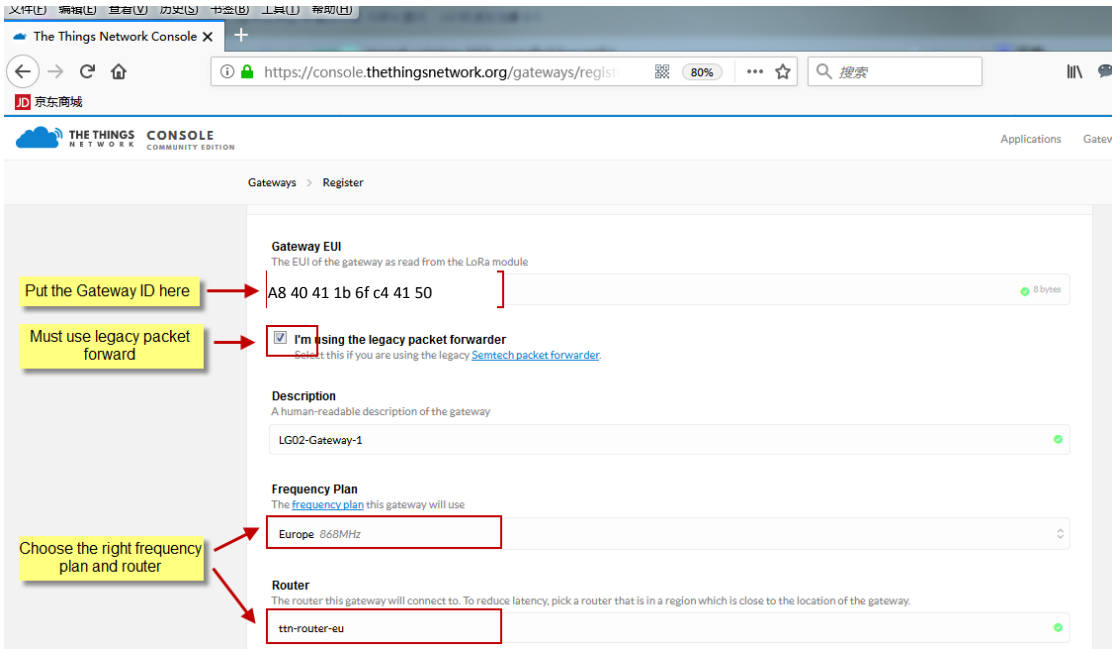
The gateway id is: **a840411b6fc44150**

Step 2: Sign up an user account in TTN server

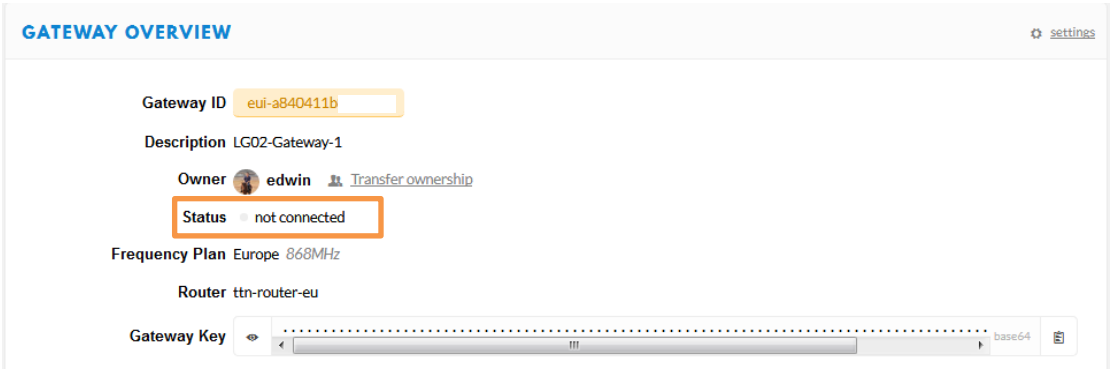


Step 3: Create a Gateway in TTN





After create the gateway, we can see the gateway info, as below, the Status shows “not connected” because the LG01N doesn’t configure to send update status yet.

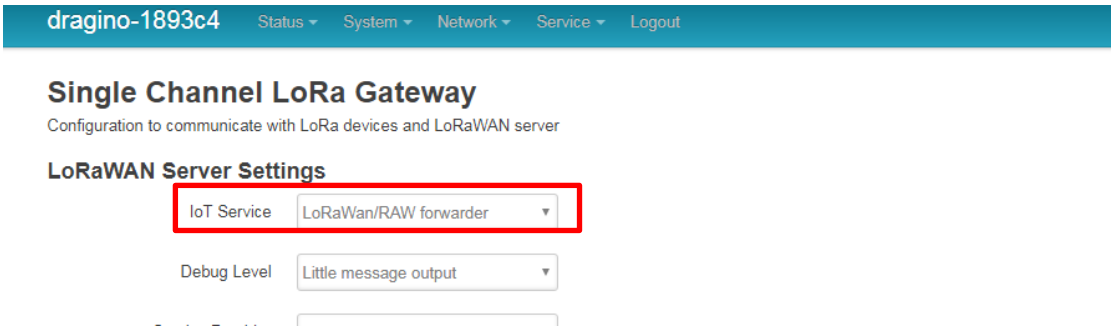


4.2 Configure LG01N Gateway

4.2.1 Configure to connect to LoRaWAN server

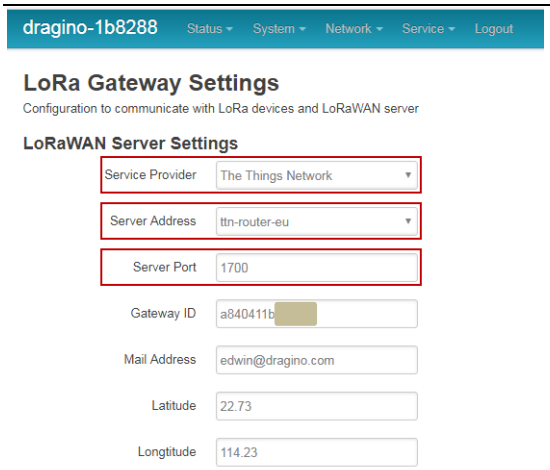
We should configure the LG01N now to let it connect to TTN network. Make sure your LG01N has Internet Connection first.

Step1: Configure LG01N to act as raw forwarder mode



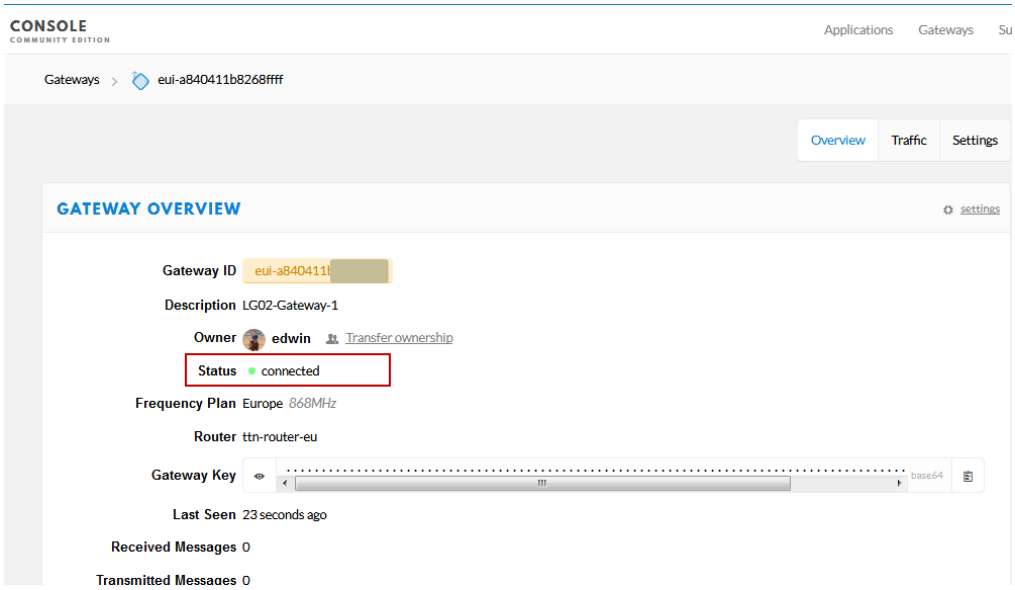
Step2: Input server info and gateway id

Choose the correct the server address and gateway ID.



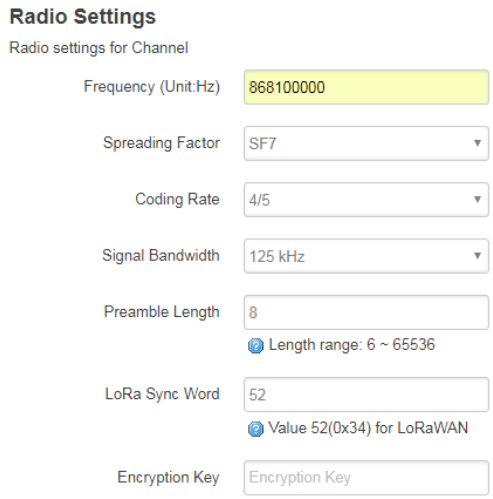
Check Result

After above settings, the LG01N should be able to connect to TTN, below is the result seen from TTN:



4.2.2 Configure LG01's Radio frequency

Now we should configure LG01N's radio parameter to receive the LoRaWAN packets. we configure is to use 868.1Mhz (868100000 Hz) as below.



4.3 Create LoRa End Node

4.3.1 About Limited support for LoRaWAN

LG01N supports LoRaWAN End Node, in LoRaWAN protocol, it requires LoRaWAN node to send data in a hopping frequency. Since LG01N only support one single frequency, it will only be able to receive the packet which is of the same radio parameters in LG01N.

For example, in EU868, a standard LoRaWAN device may send the data in eight frequencies with different Frequency & SF, such as:

```
LMIC_setupChannel(0, 868100000, DR_RANGE_MAP(DR_SF12, DR_SF7),    BAND_CENTI);        // g-band
LMIC_setupChannel(1, 868300000, DR_RANGE_MAP(DR_SF12, DR_SF7B), BAND_CENTI);        // g-band
LMIC_setupChannel(2, 868500000, DR_RANGE_MAP(DR_SF12, DR_SF7),    BAND_CENTI);        // g-band
LMIC_setupChannel(3, 867100000, DR_RANGE_MAP(DR_SF12, DR_SF7),    BAND_CENTI);        // g-band
LMIC_setupChannel(4, 867300000, DR_RANGE_MAP(DR_SF12, DR_SF7),    BAND_CENTI);        // g-band
LMIC_setupChannel(5, 867500000, DR_RANGE_MAP(DR_SF12, DR_SF7),    BAND_CENTI);        // g-band
LMIC_setupChannel(6, 867700000, DR_RANGE_MAP(DR_SF12, DR_SF7),    BAND_CENTI);        // g-band
LMIC_setupChannel(7, 867900000, DR_RANGE_MAP(DR_SF12, DR_SF7),    BAND_CENTI);        // g-band
LMIC_setupChannel(8, 868800000, DR_RANGE_MAP(DR_FSK,   DR_FSK),    BAND_MILLI);        // g2-band
```

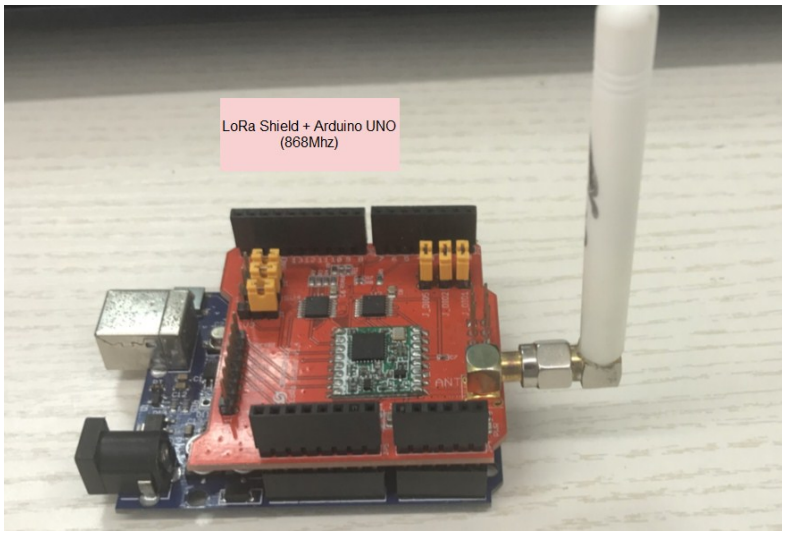
So the LG01N will only able to receive the 868100000, SF7 packet and will not receive others. Means only one packet will arrive the TTN server in every 8 packet sent from the LoRaWAN end node.

If user want all the packets from LoRaWAN end device can arrive LoRaWAN server, user need to set up the LoRaWAN node to send packet in a single frequency.

In this section, we will use LoRa Shield and a modify LMIC Library to show how to configure LoRaWAN end node and work in single frequency.

4.3.2 Preparation

LoRaWAN End device Hardware:



Software Library for LoRaWAN End device:

Install this library https://github.com/dragino/arduino-lmic to the Arduino Library path. Before compiling the End Device software, User needs to change the Frequency Band to use with LG02. What user need to change is in the file **arduino\libraries\arduino-lmic \src\lmic\config.h.**

Changes are as below:

```
#define CFG_eu868 1
//#define CFG_us915 1
//#define CFG_au921 1
//#define CFG_as923 1
//#define CFG_in866 1

#define LG02_LG01 1

//US915: DR_SF10=0, DR_SF9=1, DR_SF8=2, DR_SF7=3, DR_SF8C=4
//       DR_SF12CR=8, DR_SF11CR=9, DR_SF10CR=10, DR_SF9CR=11, DR_SF8CR=12, DR_SF7CR
#if defined(CFG_us915) && defined(LG02_LG01)
// CFG_us915 || CFG_as923
#define LG02_UPFREQ   902320000
#define LG02_DNWFREQ  923300000
#define LG02_RXSF     3       // DR_SF7
#define LG02_TXSF     8       // DR_SF12CR
#elif defined(CFG_eu868) && defined(LG02_LG01)
// CFG_eu868
//EU868: DR_SF12=0, DR_SF11=1, DR_SF10=2, DR_SF9=3, DR_SF8=4, DR_SF7=5, DR_SF7B=1, DR_FSK, DR_NONE
#define LG02_UPFREQ   868100000
#define LG02_DNWFREQ  869525000
#define LG02_RXSF     5       // DR_SF7
#define LG02_TXSF     0       // DR_SF12
#endif
```

Choose the Frequency Band, same as in LoRaWAN server

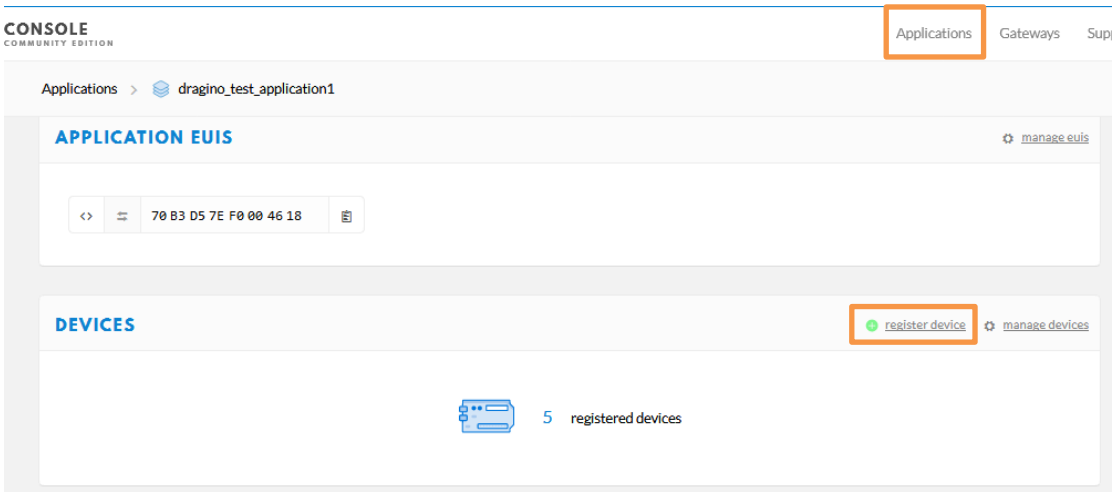
uncomment this for LG01 / LG02

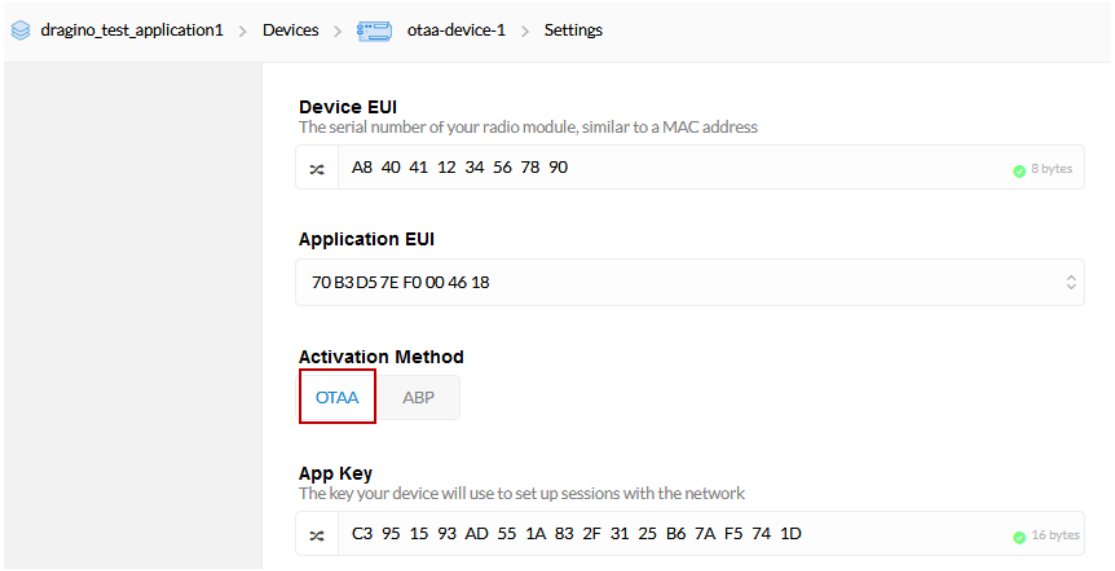
LG02_UPFREQ:End Device Uplink Frequency
LG02_DNWFREQ:End Device Uplink Frequency
LG02_RXSF: End Device Uplink (transmit) SF
LG02_TXSF: End Device Downlink (receive) SF

The TXSF is now set to default value:
US915/AS923 : 923300000 , SF12BW500
EU868: 869525000, SF12BW125

4.3.3 Test with OTAA LoRa end node (LoRa Shield + UNO)

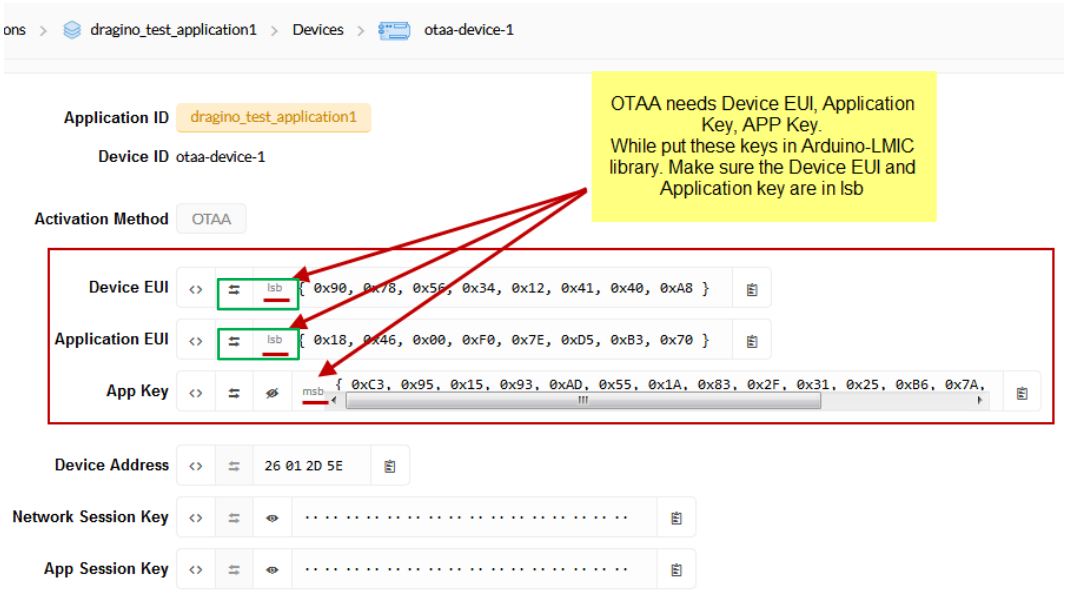
Step 1: Create an OTAA device in TTN server -- > Application page.

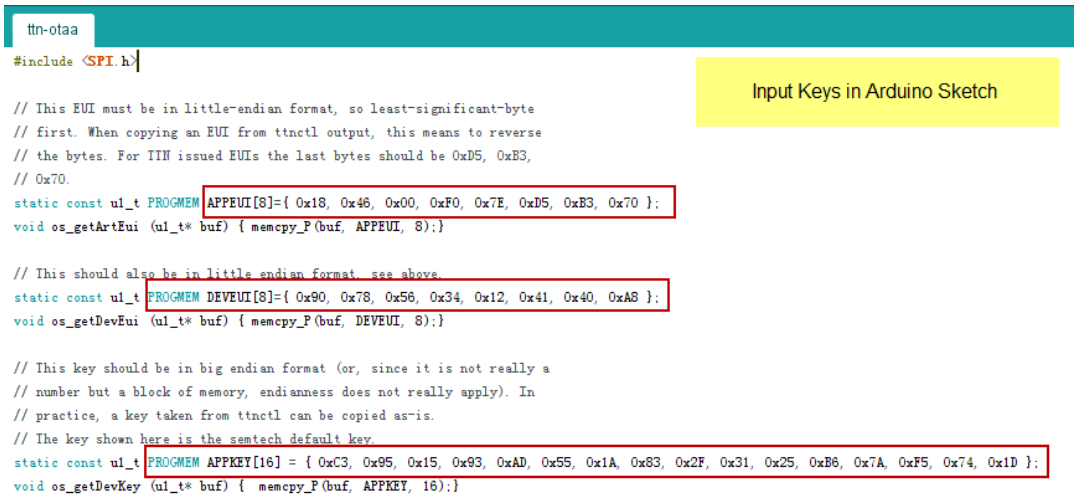




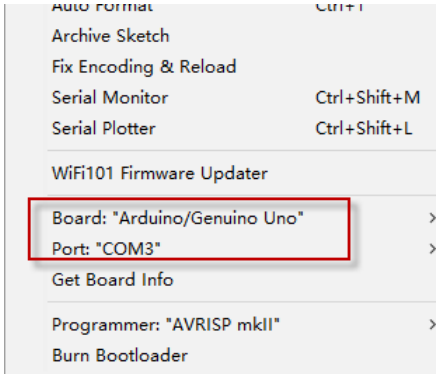
Step 2: Input keys into Arduino Sketch.

The sketch for the LoRa Shield is in Arduino –IDE --> Examples -->LMIC_Arduino→ ttn-otaa

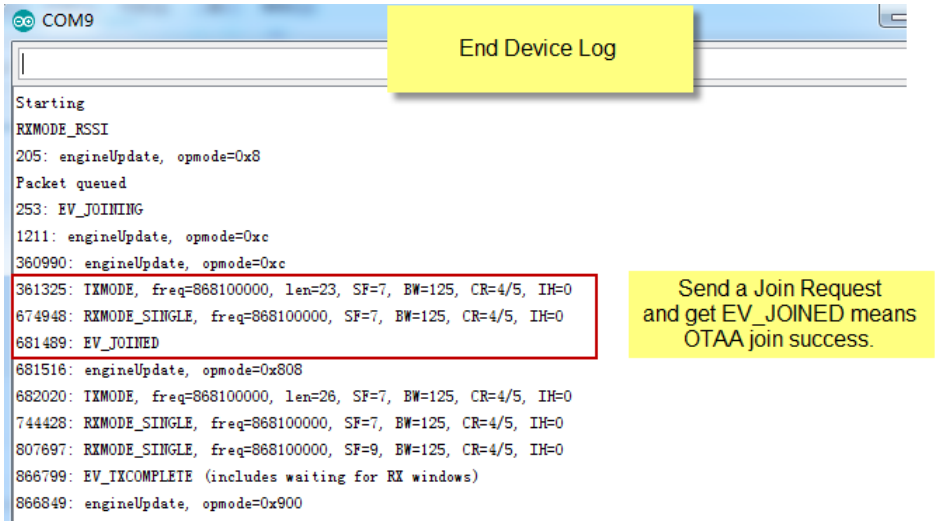


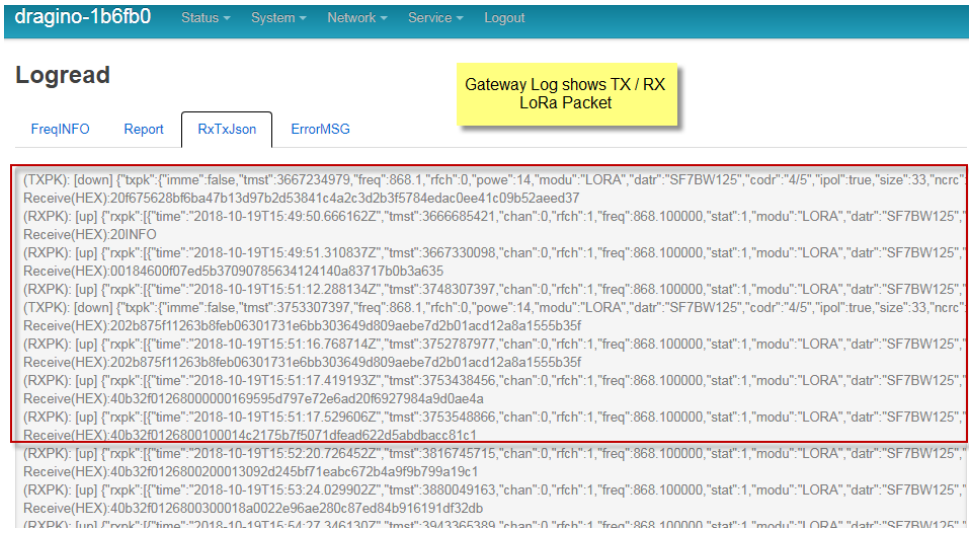


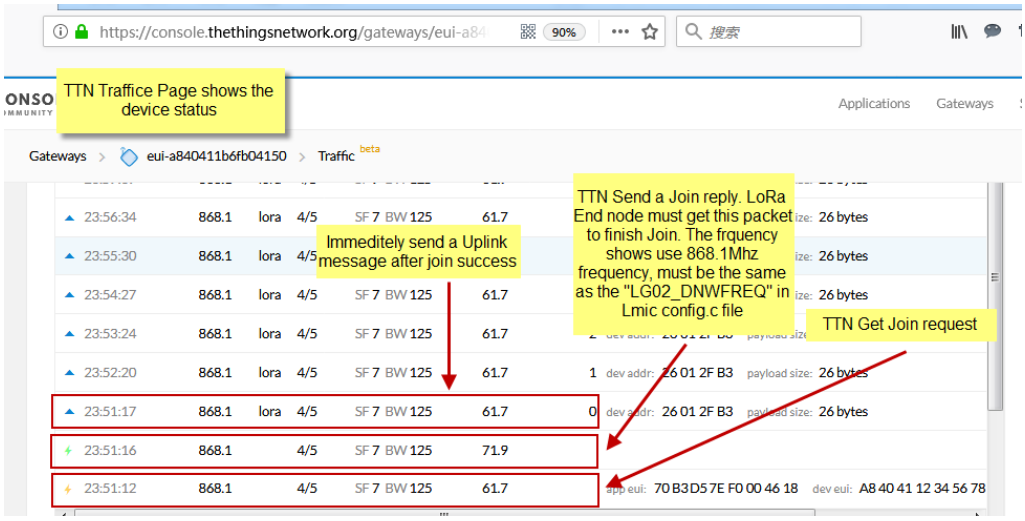
Choose Arduino UNO to upload the sketch to LoRa Shield and UNO



Step 3: Check Result for OTAA



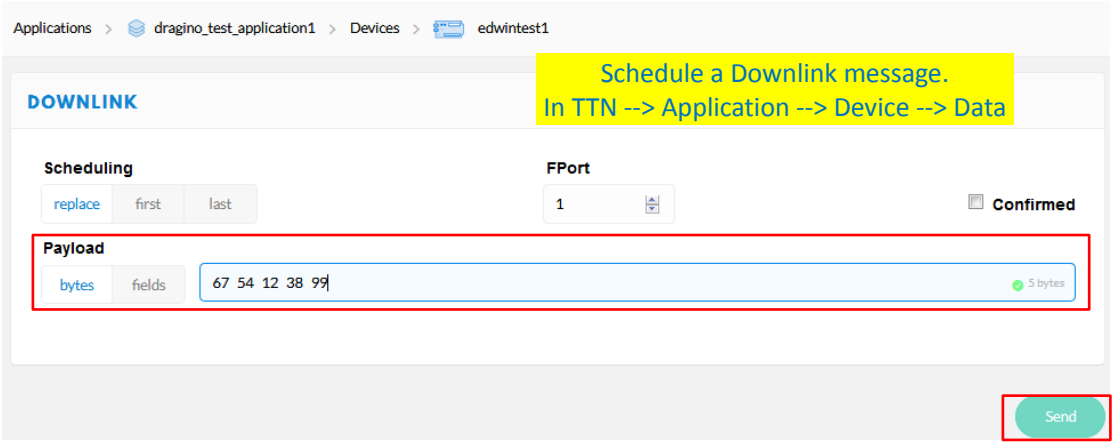


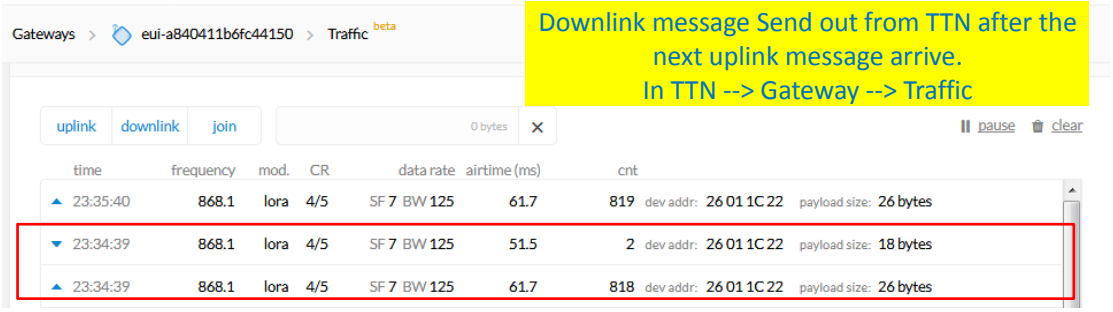


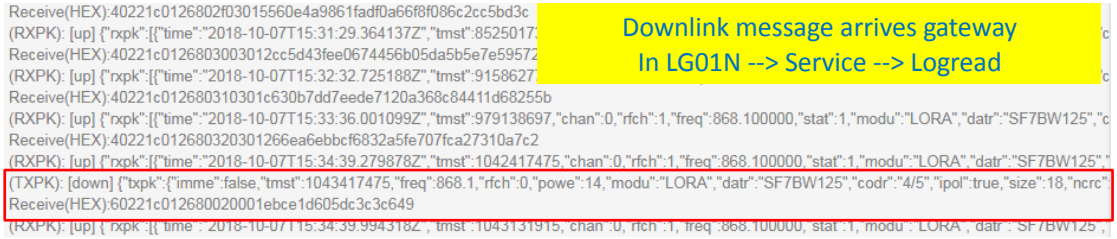
Note: The LG02_DNWFREQ value in Arduno_LMIC/src/lmic/config.h should match downlink frequency from TTN.

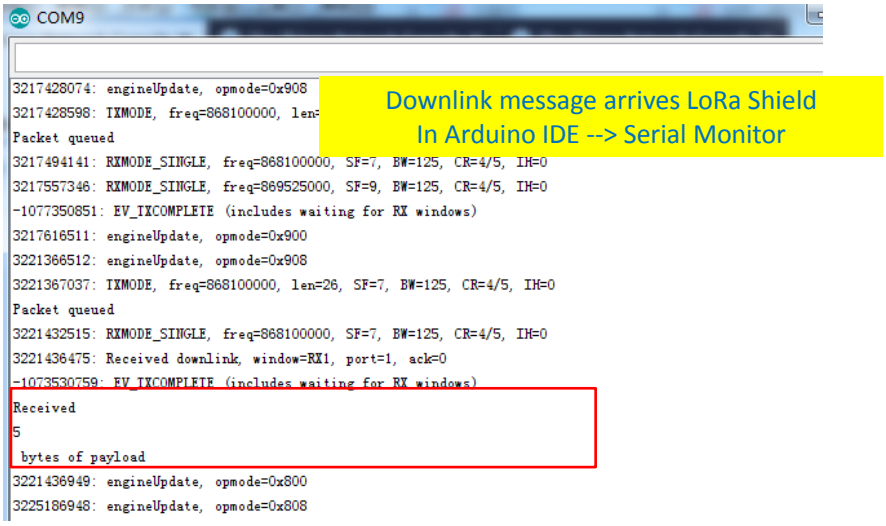
TTN shows 868.1 here, So LG02_DNWFREQ should be 868100000

Step 4: Test Downlink



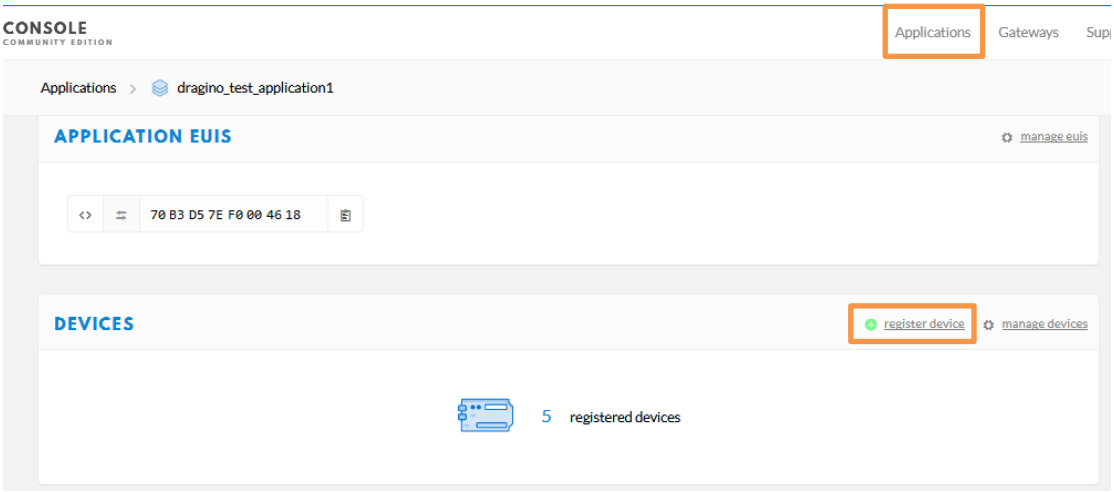


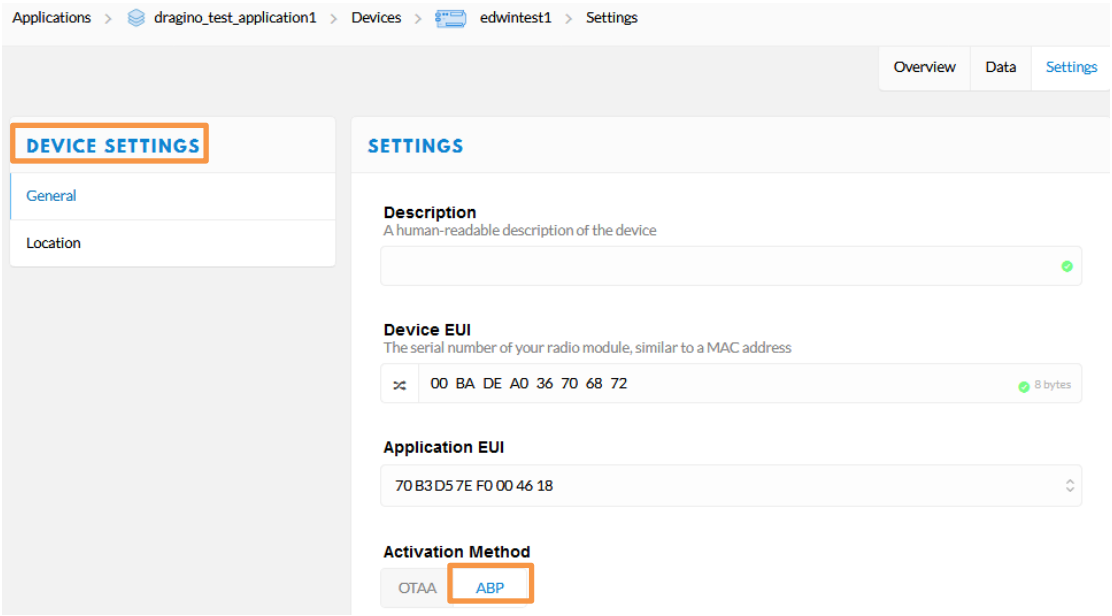




4.3.4 Test with ABP LoRa end node (LoRa Shield + UNO)

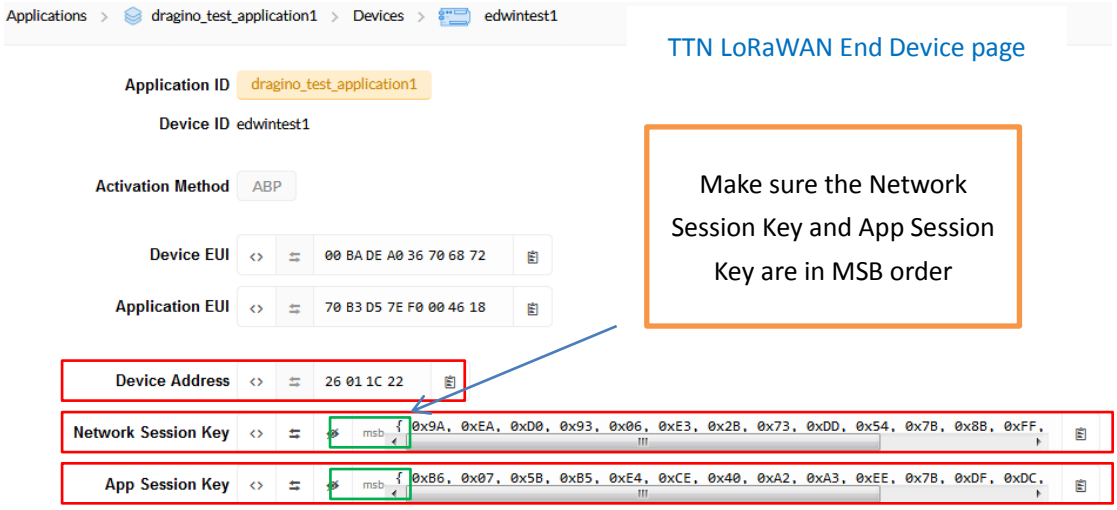
Step 1: Create an ABP device in TTN server -- > Application page. And change it to ABP mode.

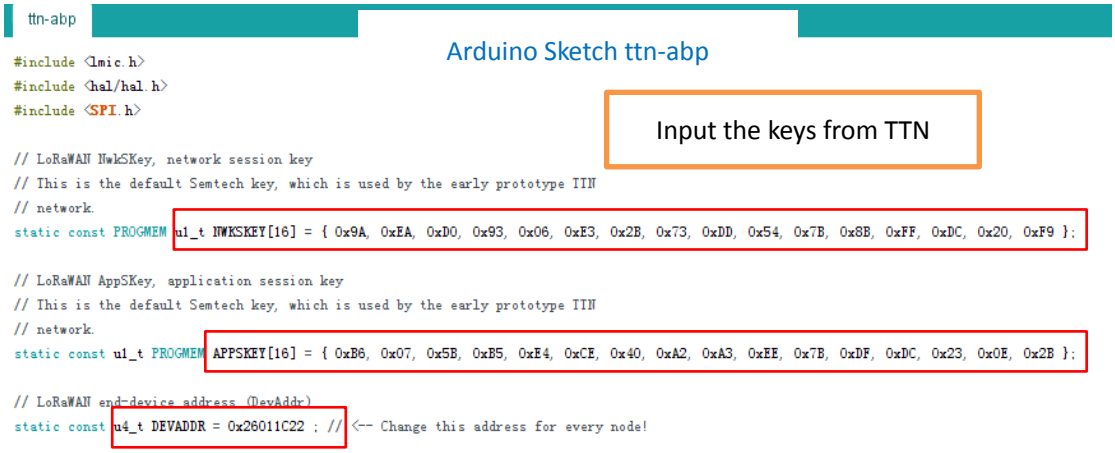




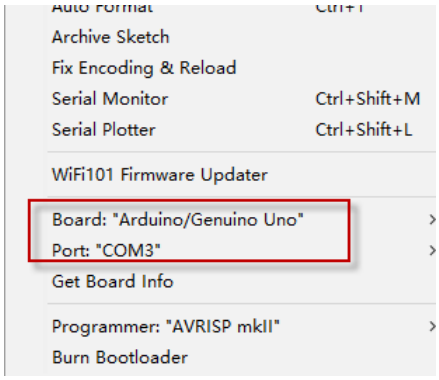
Step 2: Input keys into Arduino Sketch.

The sketch for the LoRa Shield is in Arduino –IDE --> Examples -->LMIC_Arduino→ ttn-abp

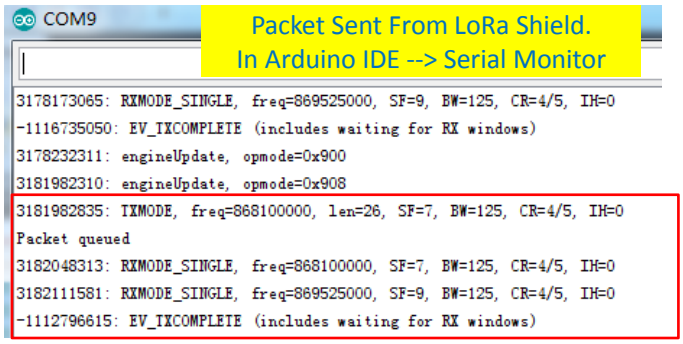


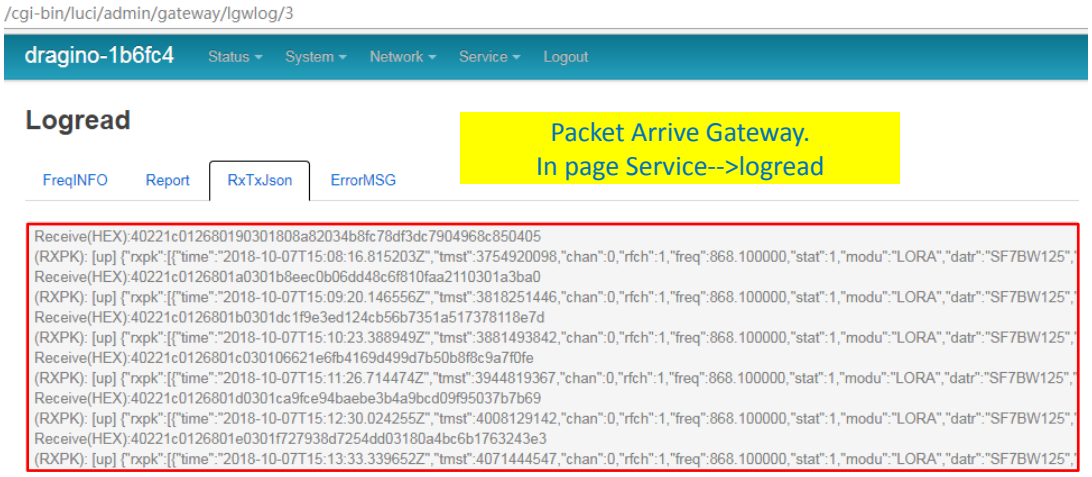


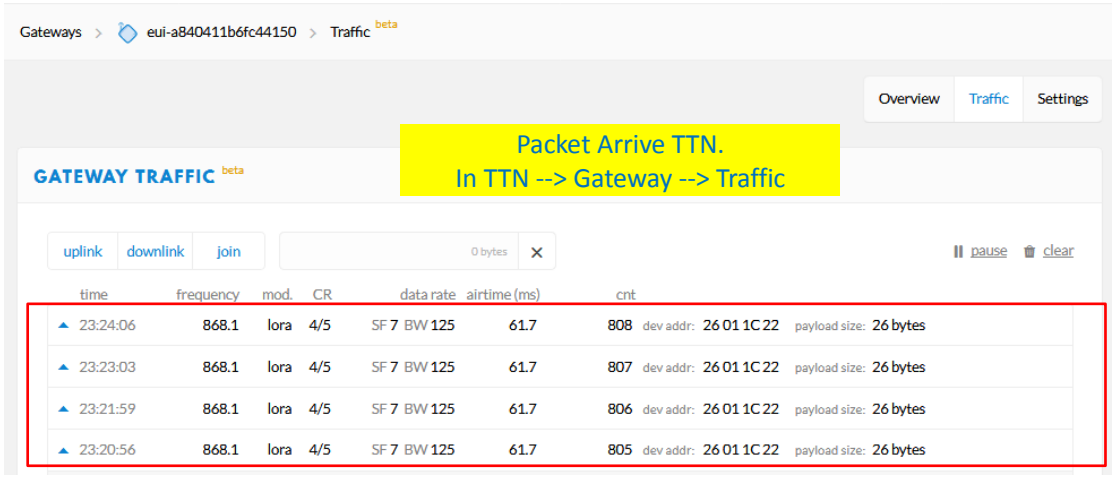
Choose Arduino UNO to upload the sketch to LoRa Shield and UNO

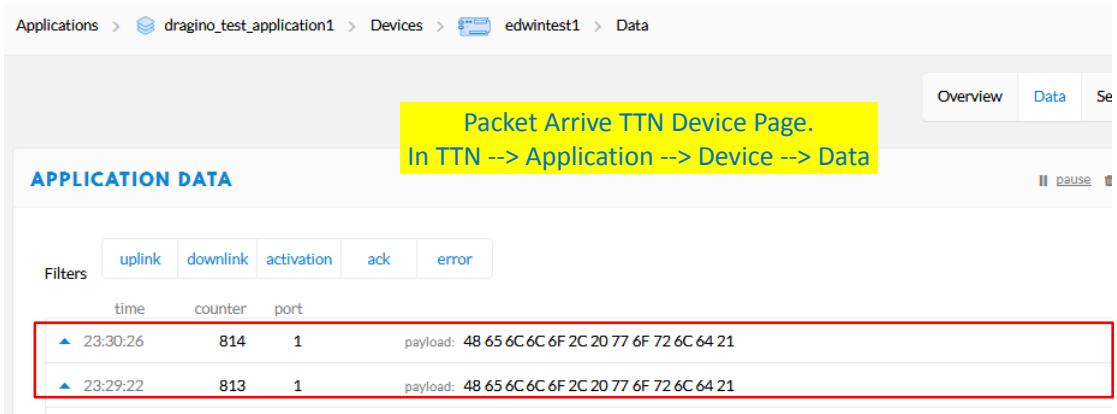


Step 3: Check Result for Uplink

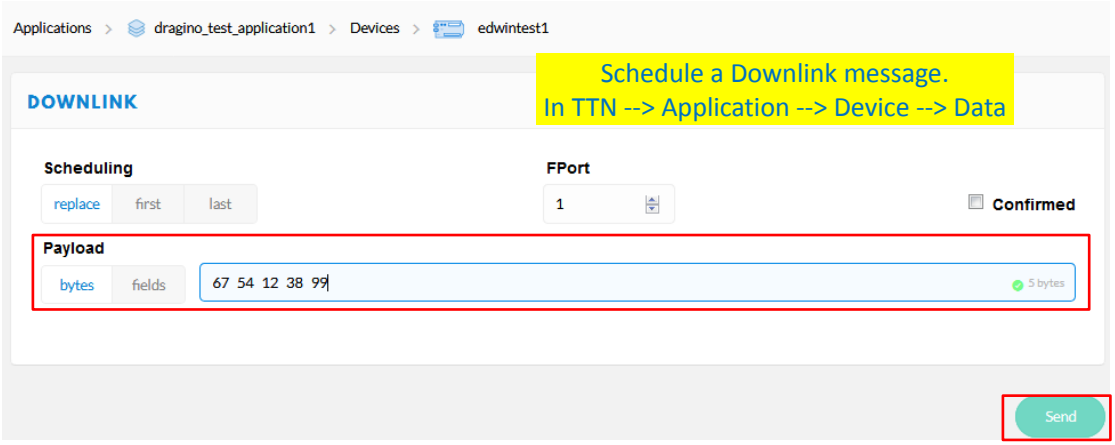


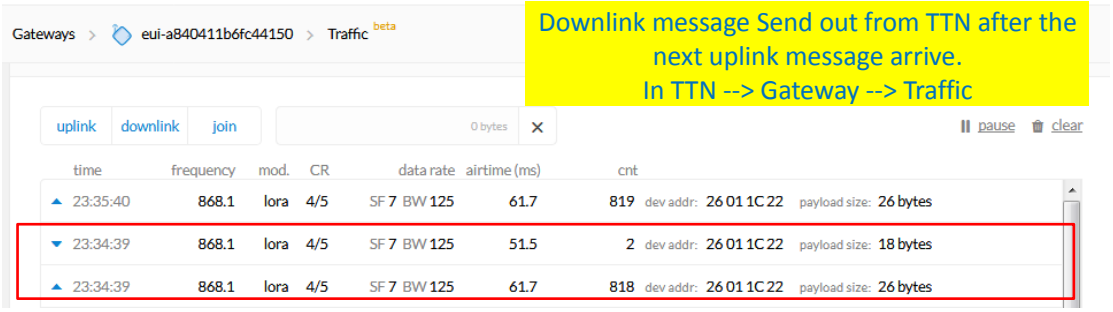


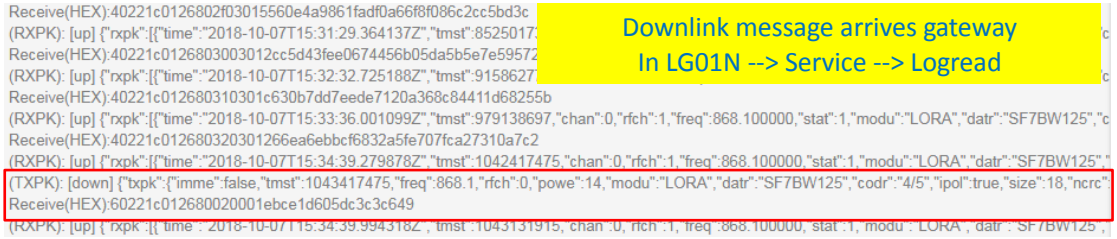


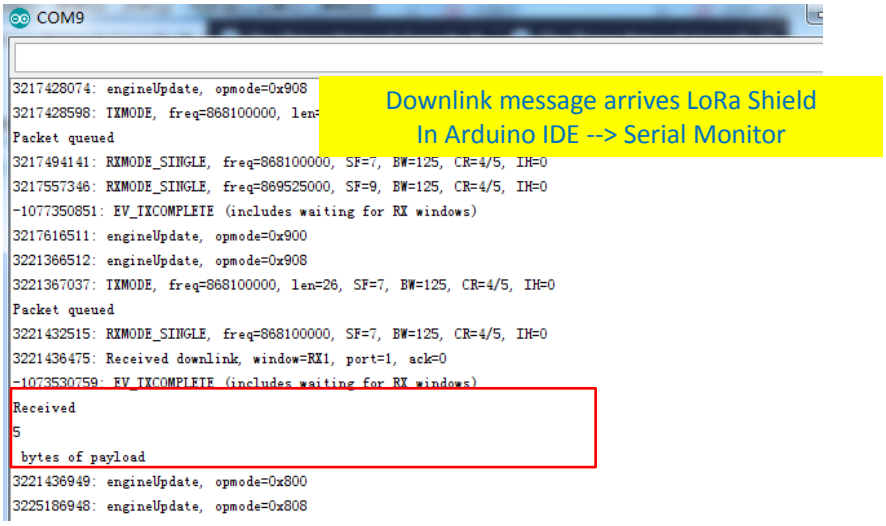


Step 4: Test Downlink









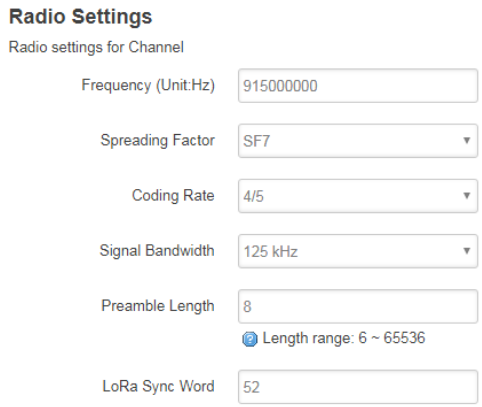
5. Example 2: Manually send / receive LoRa packets

There are two ways to use the LoRa Radio of Gateway: a) Through pkt_fwd process , b) Use the Radio separately.

5.1 User LoRa Radio via pkt_fwd

5.1.1 Use pkt_fwd to receive

When user chooses the MQTT/TCP-IP/Customized mode, the lg01_pkt_fwd will auto start. It will listen the LoRa Radio Channel base on the setting in the web setting.



If the LoRa end node send data in the match format, the pkt_fwd will store the data for further use, the logic of this receive part please see Customized_Script.

5.1.2 Use pkt_fwd to transmit

(This is a new feature since 2019-Jan-30)

The pkt_fwd also open a thread to listen to local files under directory **/var/iot/push/**. Once there is a file in this directory, the thread will check if it is an outgoing file and send out the LoRa message if format match. Below is the file example (json format):

```
{"txpk":{"imme":false,"tmst":861608339,"freq":925.1,"rfch":0,"powe":20,"modu":"LORA","datr":"SF7BW500","codr":"4/5","ipol":true,"size":22,"ncrc":true,"data":"YEklBCaqCgADQAIAcQM6AP8B9TYzUA=="}}
```

Explain:

Name	Type	Function
imme	bool	Send packet immediately (will ignore tmst & time)
tmst	number	Send packet on a certain timestamp value (will ignore time)
tmms	number	Send packet at a certain GPS time (GPS synchronization required)
freq	number	TX central frequency in MHz (unsigned float, Hz precision)
rfch	number	Concentrator "RF chain" used for TX (unsigned integer)
powe	number	TX output power in dBm (unsigned integer, dBm precision)
modu	string	Modulation identifier "LORA" or "FSK"
datr	string	LoRa datarate identifier (eg. SF12BW500)

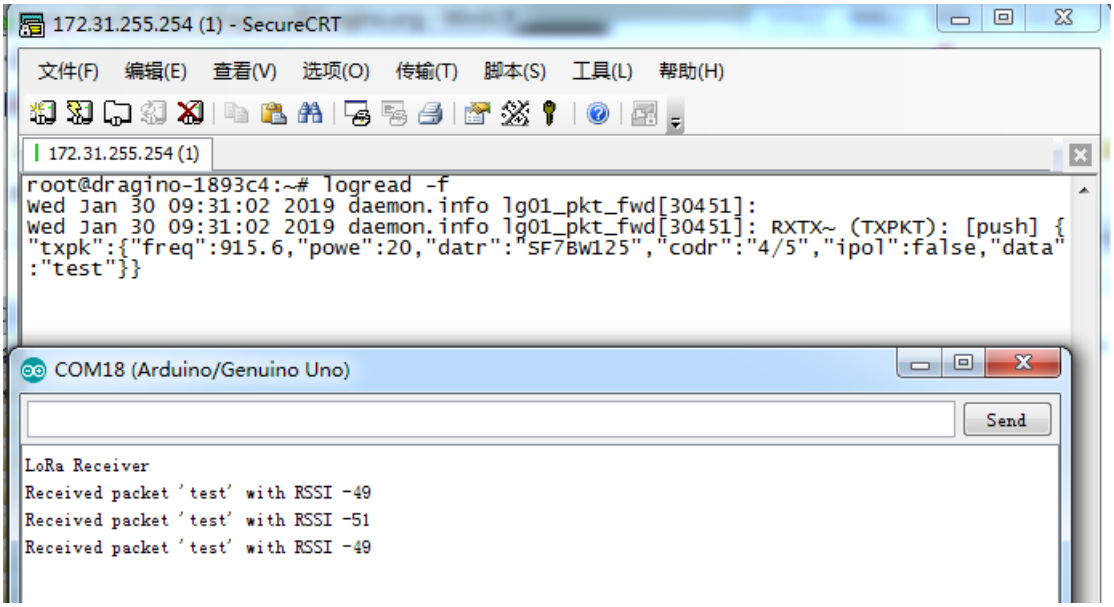
datr | number | FSK datarate (unsigned, in bits per second)
codr | string | LoRa ECC coding rate identifier
fdev | number | FSK frequency deviation (unsigned integer, in Hz)
ipol | bool | Lora modulation polarization inversion
prea | number | RF preamble size (unsigned integer)
size | number | RF packet payload size in bytes (unsigned integer)
data | string | Base64 encoded RF packet payload, padding optional
ncrc | bool | If true, disable the CRC of the physical layer (optional)

Not all fields are necessary, below is an example:

1) First set up a LoRa Shield with this code: LoRaReceiver. So the LoRa Shield will receive the data at frequency 915.6Mhz, SF7BW125, CR: 4/5
2) Edit a file (any name) under **/var/iot/push/** with below content**.**

{"txpk":{"freq":915.6,"powe":20,"datr":"SF7BW125","codr":"4/5","ipol":false,"data":"test"}}

And then we can see below output

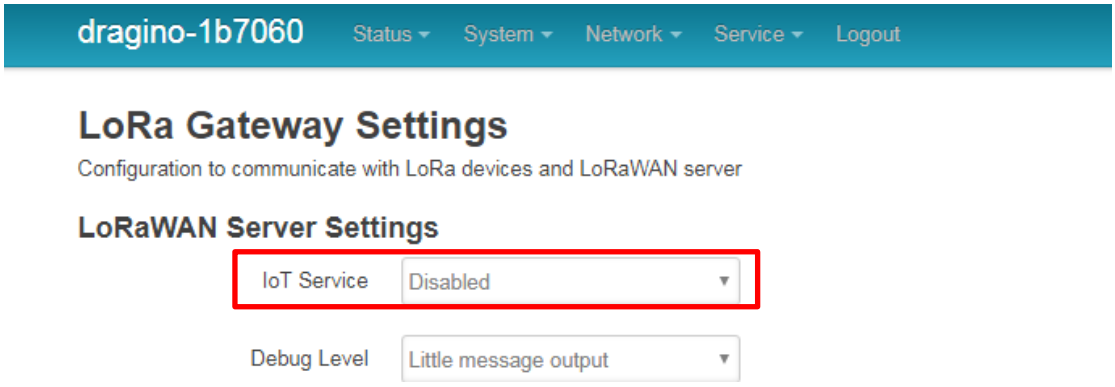


5.2 Use LoRa radio device directly

The LoRa Radio of LG01N is a SPI device, user can use lg02_single_rx_tx to control this SPI device for transmit and receive. When use the lg02_single_rx_tx command to transmit, it will initiate the SPI device on each call and it will add delay to start transmit, it is will be slower than above method (via pkg_fwd)

Step 1: Disable packet forward

With firmware higher than version LG02_LG08--build-v5.1.1545908833-20181227-1908, select "Disabled" in IoT Service page.



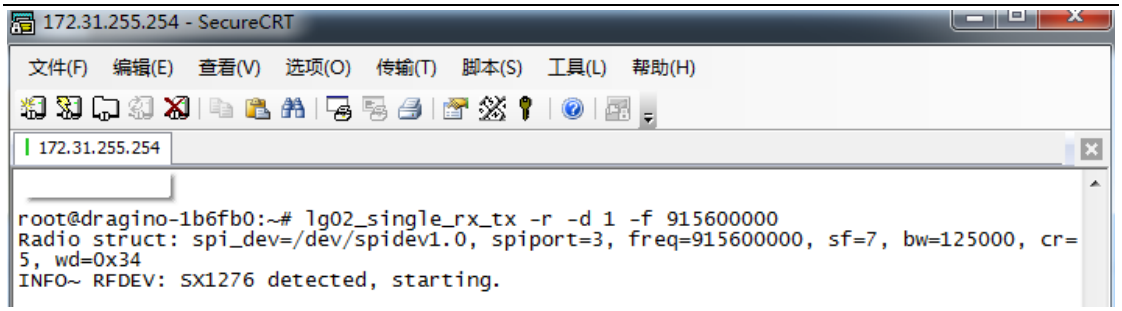
Step 2: Use lg02_single_rx_tx to receive, for LG01N, the option [-d] is 2

Usage: lg02_single_rx_tx [-d radio_dev] select radio 1 or 2 (default:1)
[-t] set as tx
[-r] set as rx
[-f frequence] (default:868500000)
[-s spreadingFactor] (default: 7)
[-b bandwidth] default: 125k
[-w syncword] default: 52(0x34)reserver for lorawan
[-m message] message to send
[-o filepath] payload output to file
[-v] show version
[-h] show this help and exit Use Radio 1 to transmit:

Command:

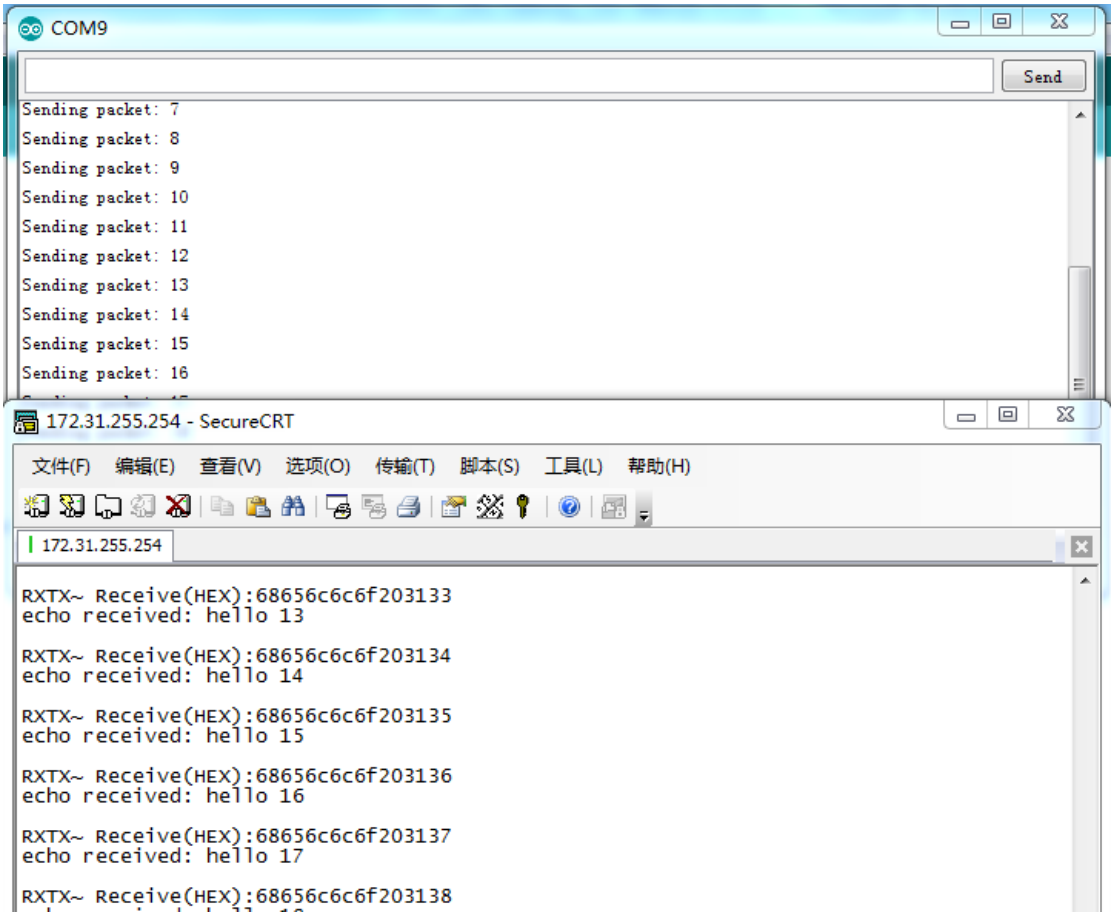
```
root@dragino-1b6fb0:~# lg02_single_rx_tx -r -d 2 -f 915600000
```

Set up the radio as receiver at frequency 9156000000,SF7BW125,SyncWord:0x34



Then set up a LoRa node to send out LoRa packet, we use LoRa Shield + UNO in this example. The library use in Arduino UNO is LoRa-Master. And the source code is LoRaSender.

Result screen shot:



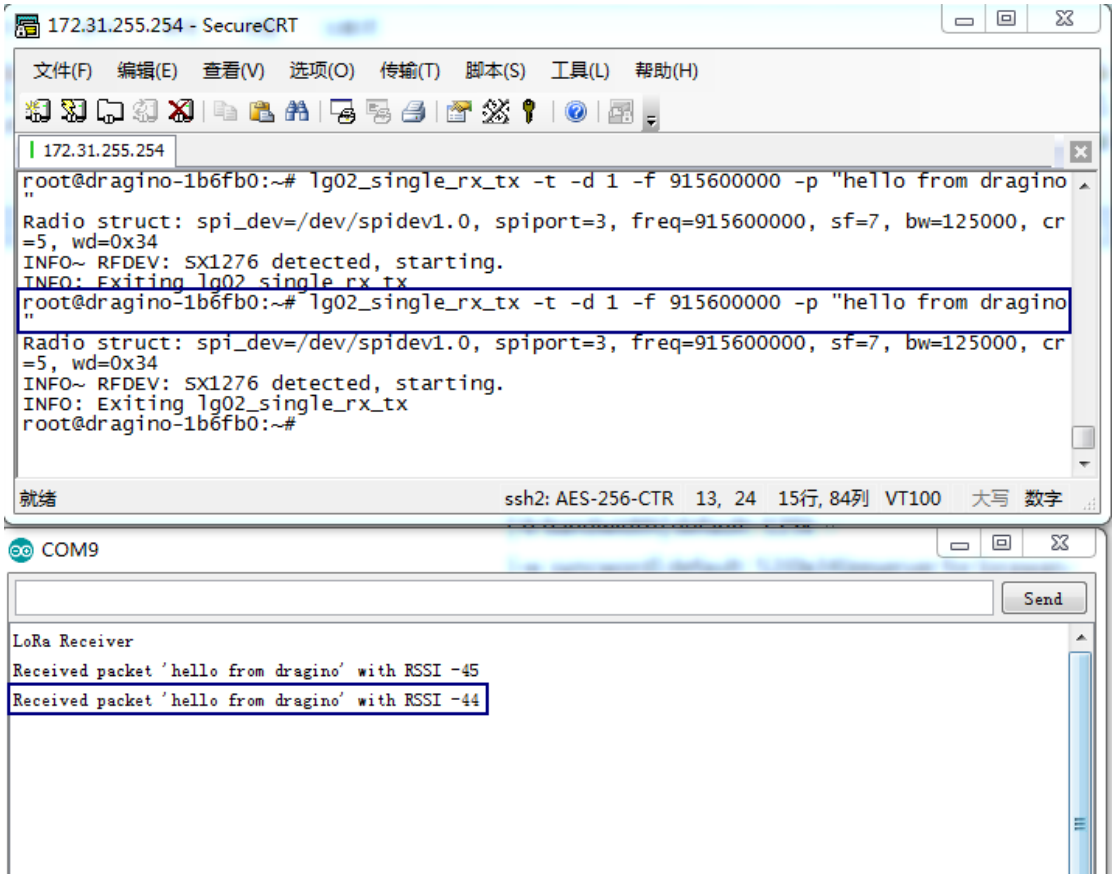
Step 3: Use lg02_single_rx_tx to transmit

Command:

root@dragino-1b6fb0:~# lg02_single_rx_tx -t -d 2 -f 915600000 –m “hello from dragino”

Set up radio to transmit a message at frequency 9156000000

Set up a LoRa node to send out LoRa packet, we use LoRa Shield + UNO in this example. The library use in Arduino UNO is LoRa-Master. And the source code is LoRaReceiver.

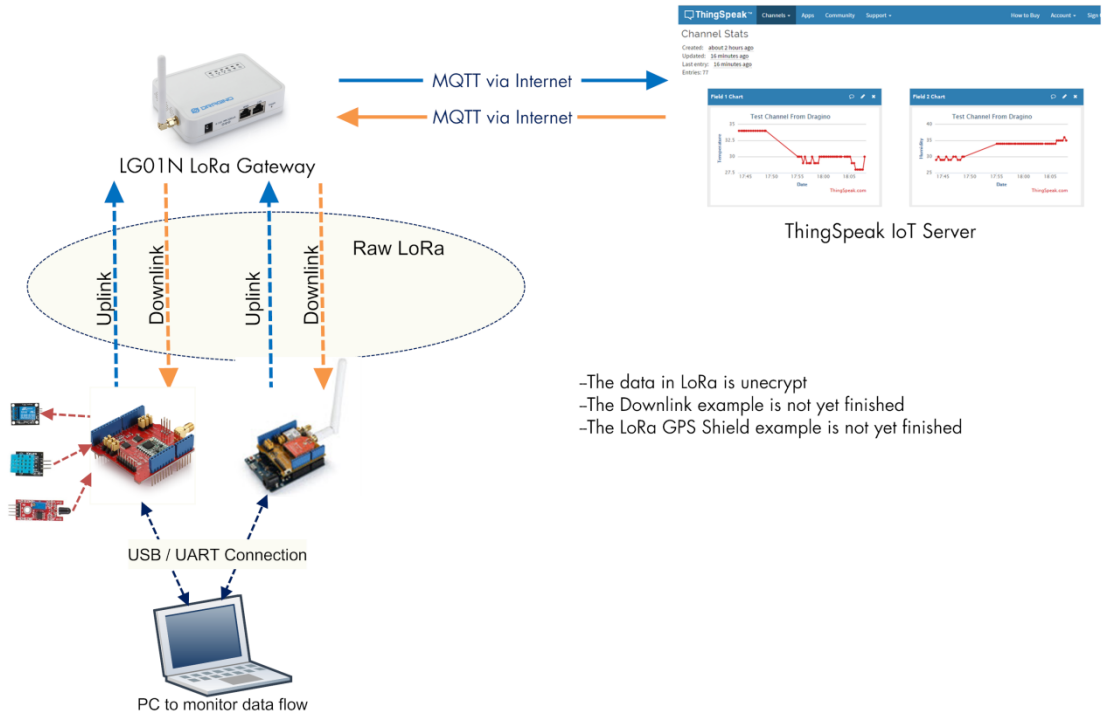


6. Example 3: MQTT Transfer Mode

MQTT is a machine-to-machine (M2M)/"Internet of Things" connectivity protocol. It was designed as an extremely lightweight publish/subscribe messaging transport. It is useful for connections with remote locations where a small code footprint is required and/or network bandwidth is at a premium. For example, it has been used in sensors communicating to a broker via satellite link, over occasional dial-up connections with healthcare providers, and in a range of home automation and small device scenarios.

Most IoT server support MQTT connection, for those servers, we can use MQTT to connect it to publish data or subscribe to a channel.

Topology for ThingSpeak Connection:

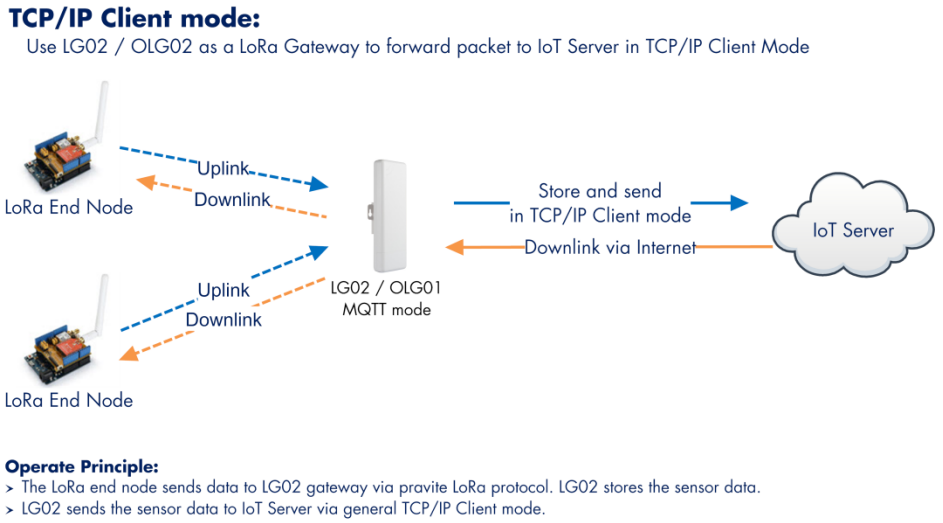


Most IoT server support MQTT connection, for those servers, we can use MQTT to connect it to publish data or subscribe to a channel.

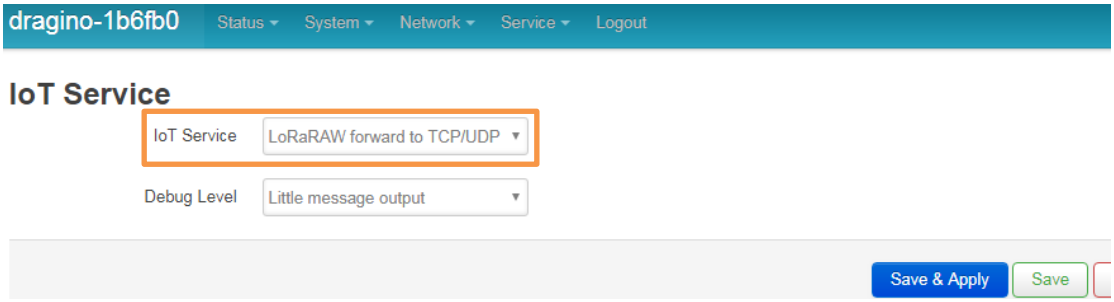
A detail of how to use MQTT plus Video instruction can be found at **Example 2: Test with a MQTT IoT Server** from user manual http://www.dragino.com/downloads/index.php?dir=LoRa_IoT_Kit/v2-Kit/ with version higher than v1.0.3

7. Example 4: TCP IP Client Mode

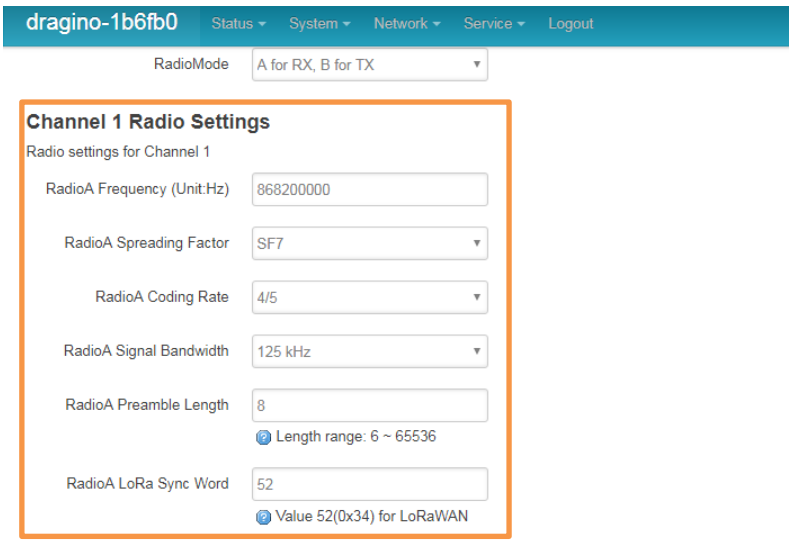
In the TCP IP Client mode, LG01N can accept LoRa packets and send it to the TCP-IP server. The working topology is as below. In this mode, The Uplink LoRa packets should use a customized format.



Step1: Select TCP-IP Client mode

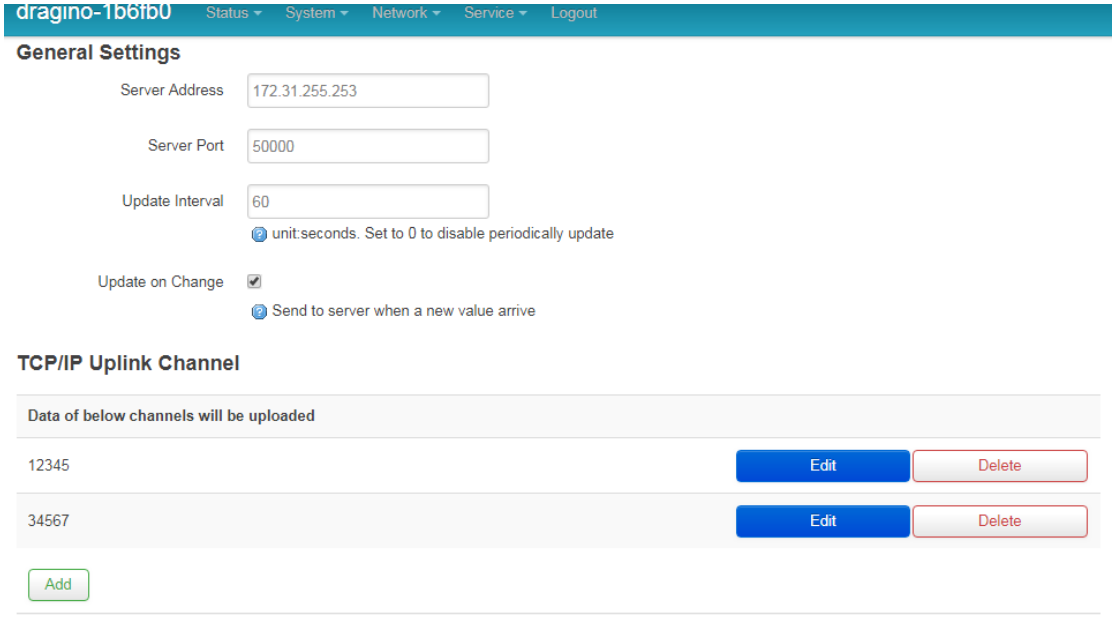


Step2: Configure the Radio channel with the match radio settings frequency as the LoRa End Node



Step3: Configure TCP Server Info

Note: Gateway may receive many LoRa packets, it will only transfer the packet with the same ID as specify in the channel.



Step4: About uplink data format

The LoRa end node should upload the data with below format:

Uplink Format: **<Channel_ID>data**

For example, if we have configured 2 channels 12345 and 34567.

And there is are three LoRa End nodes sending: 12345,34567,78

The LG02 will accept the data from 12345 and 34567, it will ignore the data from Node 78

Case 1:

Node 12345 send <12345>field1=0.0&field2=1102.0

Node 34567 doesn't send anything

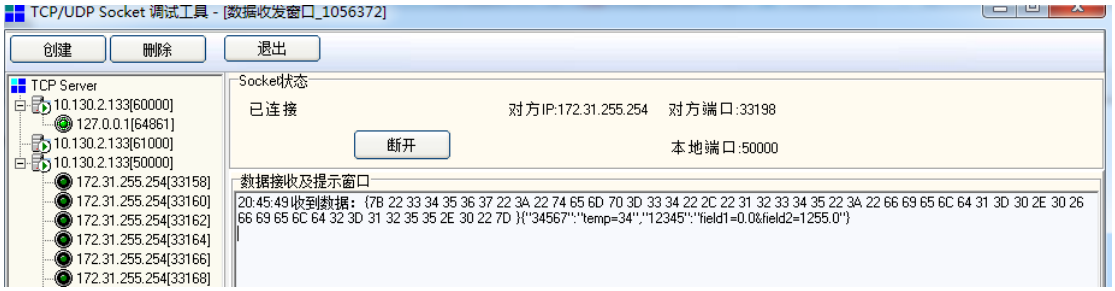
The TCP/IP server will get {"12345":"field1=0.0&field2=1102.0"}

Case 2:

Node 12345 send <12345>field1=0.0&field2=1102.0

Node 34567 send <34567>temp=34

The TCP/IP server will get {"34567":"temp=34","12345":"field1=0.0&field2=1102.0"}

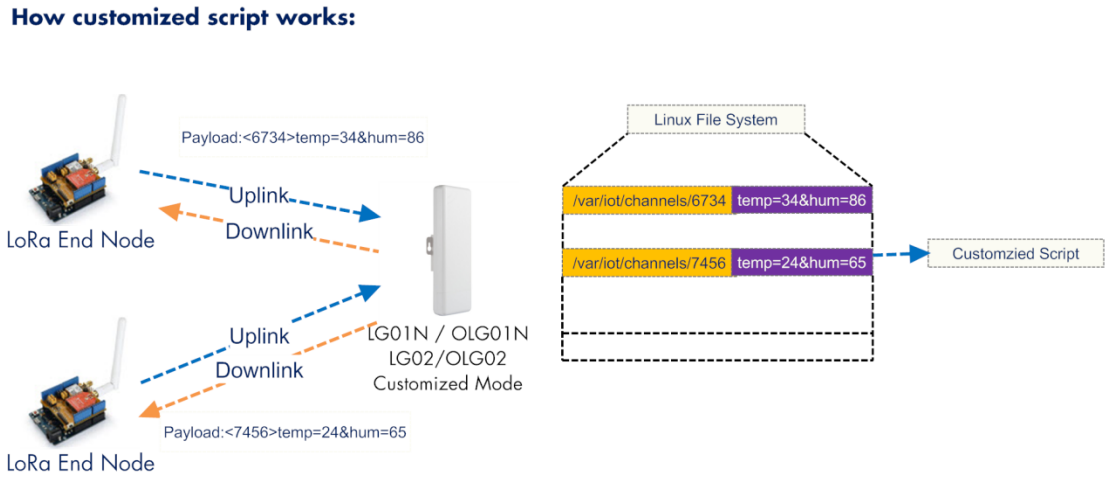


LoRa End Device reference source code: [check this link](#).

8. Example 5: Write a customized script

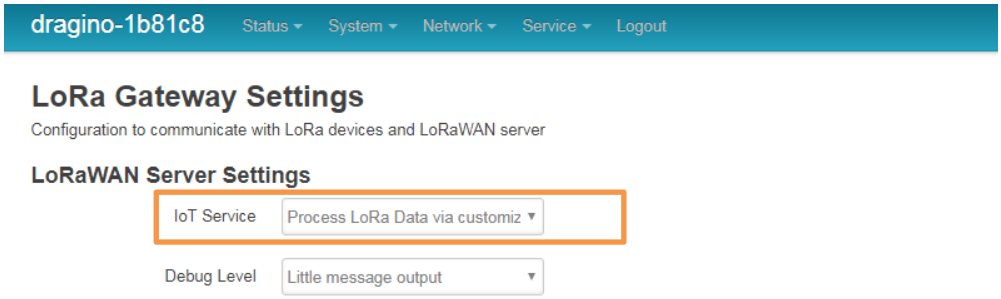
LG01N supports customized script to process LoRa data. This chapter describes about the data format from LoRa End node and How to write the script.

The data flow from LoRa End Node to LG01N is as below:

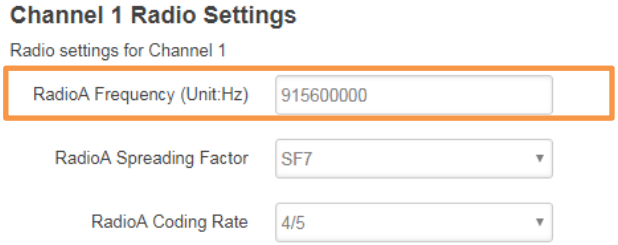


Example: Store Data in a file.

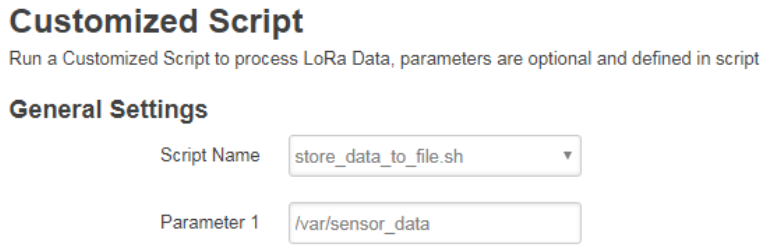
Step 1: Choose LoRa customized script mode



Step 2: Configure LoRa Frequency



Step 3: Choose the customized script



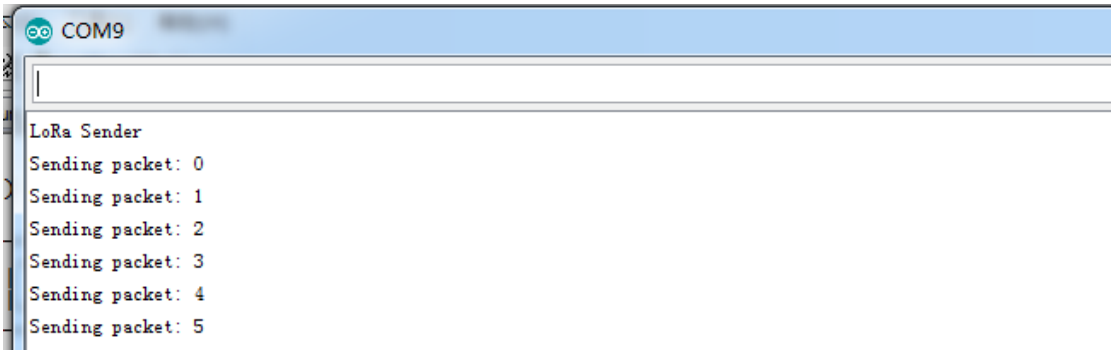
The directory to store customized script is in **/etc/lora/customized_scripts/**. User can write a new script and put it under this directory for their application. The web will auto detect it.

Step 4: Configure the LoRa End Device to send sensor data.

Here is an example code for LoRa Shield: **End Device Code**

Outputs:

End node send out packages:



Gateway receive packet & Script find packet

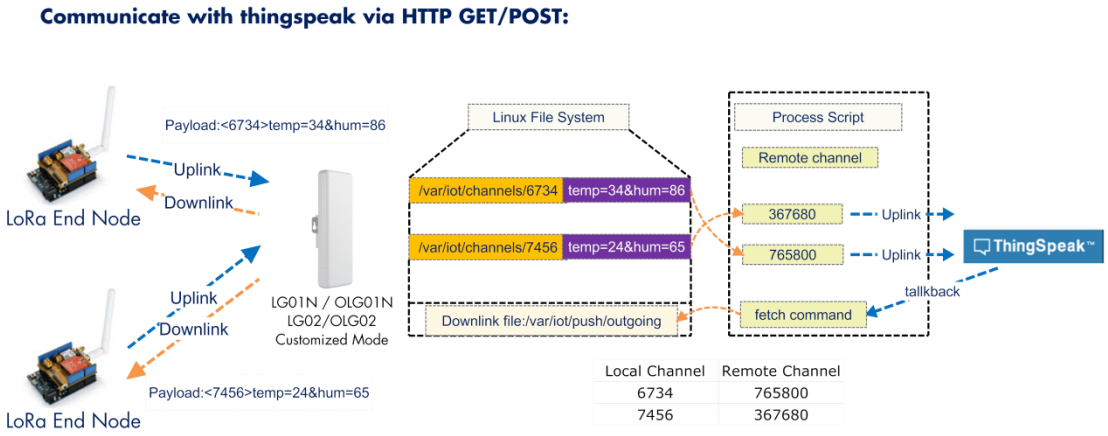
```
root@dragino-1b81c8:~# logread -f
Sun Jan  1 00:47:08 2012 user.notice root: [IoT]: Found field1=25&field2=87 at Local Channel: 10009
Sun Jan  1 00:47:08 2012 user.notice root: [IoT]: Append at /var/sensor_data
Sun Jan  1 00:47:13 2012 daemon.info lg02_pkt_fwd[31105]:
Sun Jan  1 00:47:13 2012 daemon.info lg02_pkt_fwd[31105]: RXTX~ Receive(HEX):3c31303030393e6669656c64313d3239266669656c64323d3933
Sun Jan  1 00:47:14 2012 user.notice root: [IoT]: Found field1=29&field2=93 at Local Channel: 10009
Sun Jan  1 00:47:14 2012 user.notice root: [IoT]: Append at /var/sensor_data
Sun Jan  1 00:47:23 2012 daemon.info lg02_pkt_fwd[31105]:
Sun Jan  1 00:47:23 2012 daemon.info lg02_pkt_fwd[31105]: RXTX~ Receive(HEX):3c31303030393e6669656c64313d3238266669656c64323d3934
Sun Jan  1 00:47:26 2012 user.notice root: [IoT]: Found field1=28&field2=94 at Local Channel: 10009
Sun Jan  1 00:47:26 2012 user.notice root: [IoT]: Append at /var/sensor_data
```

Script store data into file

```
root@dragino-1b81c8:~# cat /var/sensor_data
Sun Jan 1 00:15:26 UTC 2012 :<1234> 123443
Sun Jan 1 00:46:26 UTC 2012 :<10009> field1=32&field2=94
Sun Jan 1 00:46:44 UTC 2012 :<10009> field1=32&field2=94
Sun Jan 1 00:46:56 UTC 2012 :<10009> field1=28&field2=93
Sun Jan 1 00:47:08 UTC 2012 :<10009> field1=25&field2=87
Sun Jan 1 00:47:14 UTC 2012 :<10009> field1=29&field2=93
Sun Jan 1 00:47:26 UTC 2012 :<10009> field1=28&field2=94
Sun Jan 1 00:47:38 UTC 2012 :<10009> field1=25&field2=90
Sun Jan 1 00:47:44 UTC 2012 :<10009> field1=27&field2=87
Sun Jan 1 00:47:56 UTC 2012 :<10009> field1=32&field2=88
Sun Jan 1 00:48:08 UTC 2012 :<10009> field1=32&field2=94
Sun Jan 1 00:48:20 UTC 2012 :<10009> field1=25&field2=87
Sun Jan 1 00:48:26 UTC 2012 :<10009> field1=28&field2=94
Sun Jan 1 00:48:38 UTC 2012 :<10009> field1=34&field2=92
Sun Jan 1 00:48:50 UTC 2012 :<10009> field1=25&field2=88
Sun Jan 1 00:48:56 UTC 2012 :<10009> field1=34&field2=93
Sun Jan 1 00:49:08 UTC 2012 :<10009> field1=31&field2=90
Sun Jan 1 00:49:20 UTC 2012 :<10009> field1=32&field2=91
Sun Jan 1 00:49:26 UTC 2012 :<10009> field1=27&field2=92
Sun Jan 1 00:49:38 UTC 2012 :<10009> field1=25&field2=88
```

9. Example 6: Communicate to a HTTP server

Here shows an example for how to communicate to ThingSpeak server via HTTP protocol.

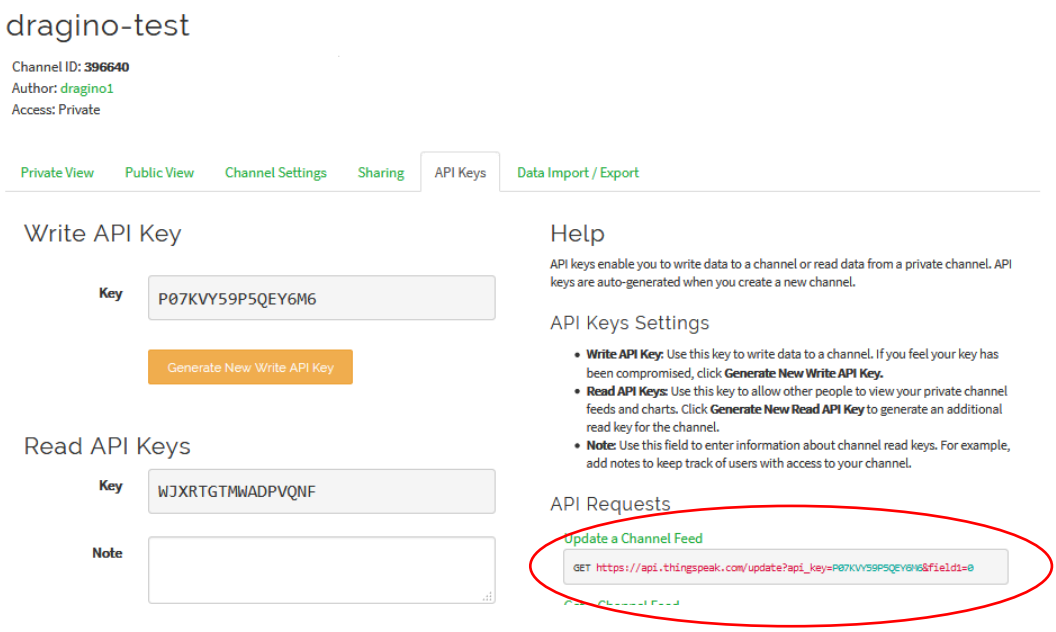


Operate Principle:

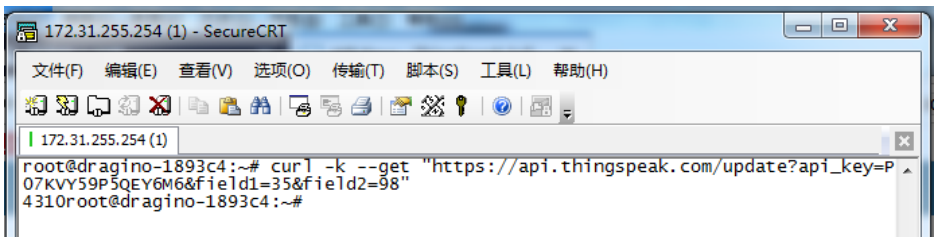
- LoRa End Node sends the data to gateway in specify format: <node_ID>value
- Gateway get the data and will put the data in corresponding files under /var/iot/channels.
- HTTP Process Script will put the data to remote channel according to the pre-configure mapping
- HTTP Process Script will run curl command to check the talkback command from server. If there is talkback command, it willlll construct a outgoing file under /var/iot/push for downlink purpose.

9.1 Test uplink and downlink via Linux command

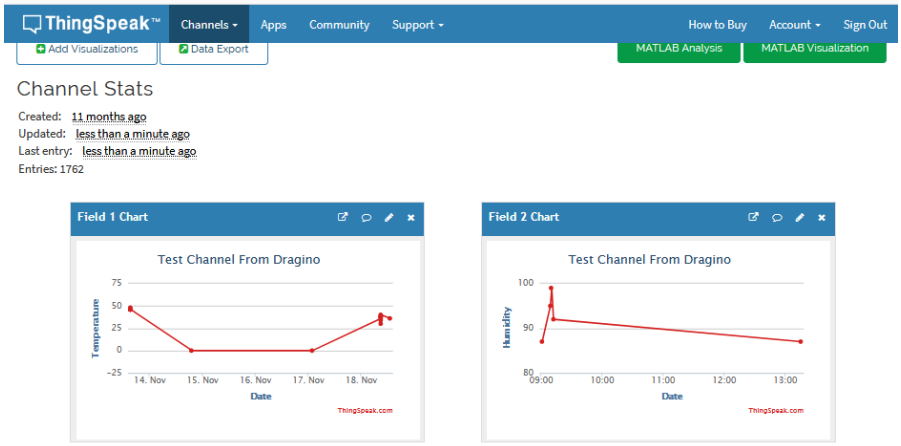
We can see the API requests method in ThingSpeak API keys tab.



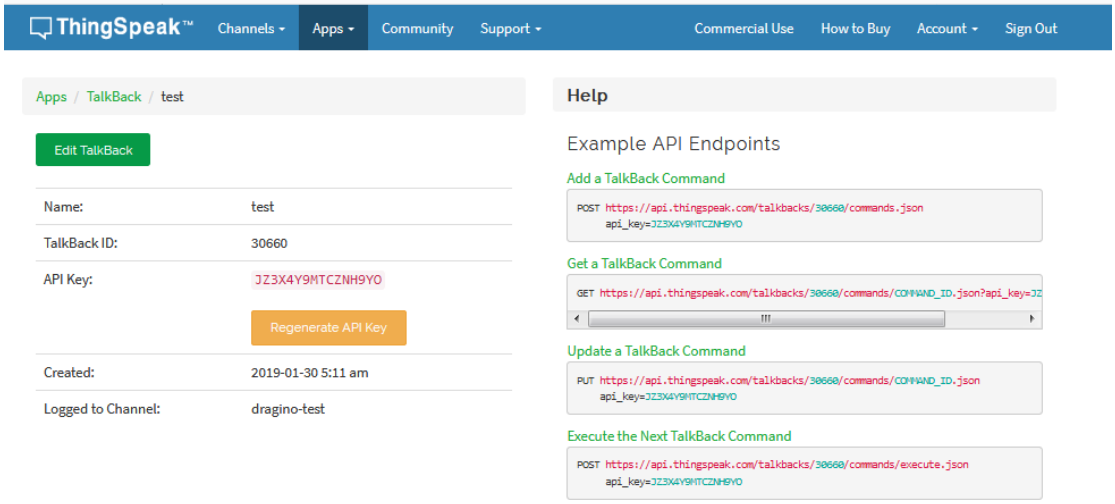
Run curl command to use this API (update a channel feed) :



And we will be able to see the update in the feed:



ThingSpeak has a talkback API which can dispatch command, it is under Apps → Talkback



We can use curl command to get command_string, as below:

9.2 Test uplink and downlink in LoRa

9.2.1 Set up on gateway

Step1:

Run below commands to download the customized script for ThingSpeak:

```
root@dragino-1893c4:~# wget http://www.dragino.com/downloads/downloads/LoRa_Gateway/LG02-OLG02/Firmware/customized_script/uplink_downlink_ThingSpeak.sh
root@dragino-1893c4:~# chmod +x uplink_downlink_ThingSpeak.sh
root@dragino-1893c4:~# mv uplink_downlink_ThingSpeak.sh /etc/lora/customized_scripts/
```

Step2:

Modify the script for your channels:
There are three place need to modify:

1. Replace the channel with the corresponding channel ID and API KEY

```
if [ "$channel" == "396640" ];then
        WRITE_API_KEY="P07KVY59P5QEY6M6"
fi
```

2.

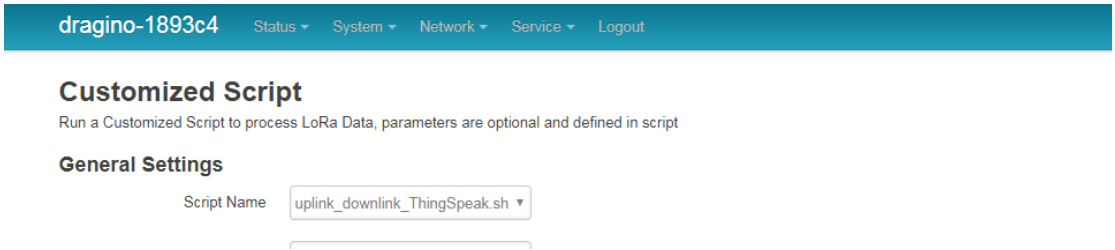
talkback=`curl …. Replace with the actually talk back URL

3. Modify this line with the suitable frequency.

echo "{\"txpk\":{\"freq\":915.0,\"powe\":2

Step3:

Select ThingSpeak script as the customized script.

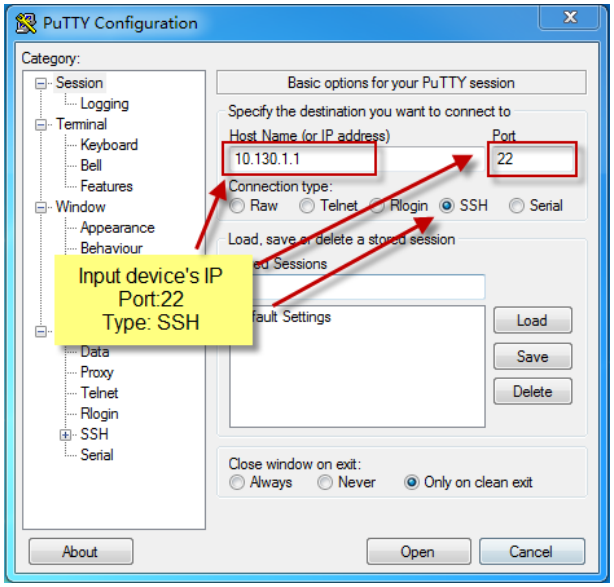


10. Linux System

The LG01N bases on OpenWrt Linux System. It is open source, and user are free to configure and modify the inside Linux settings.

10.1 SSH Access for Linux console

User can access to the Linux console via SSH protocol. Make sure your PC and the LG01 is in the same network, then use a SSH tool (such as putty) to access it. Below are screenshots:



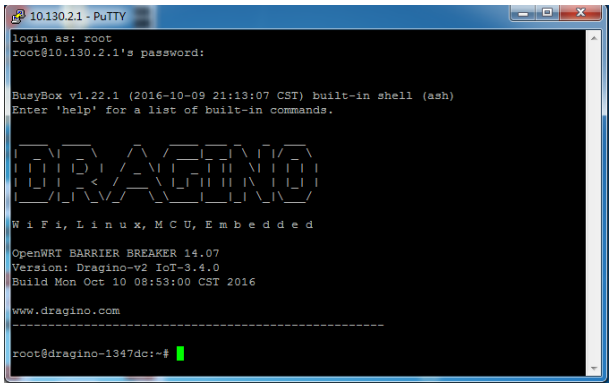
IP address: IP address of LG01N

Port: 22 or 2222

User Name: **root**

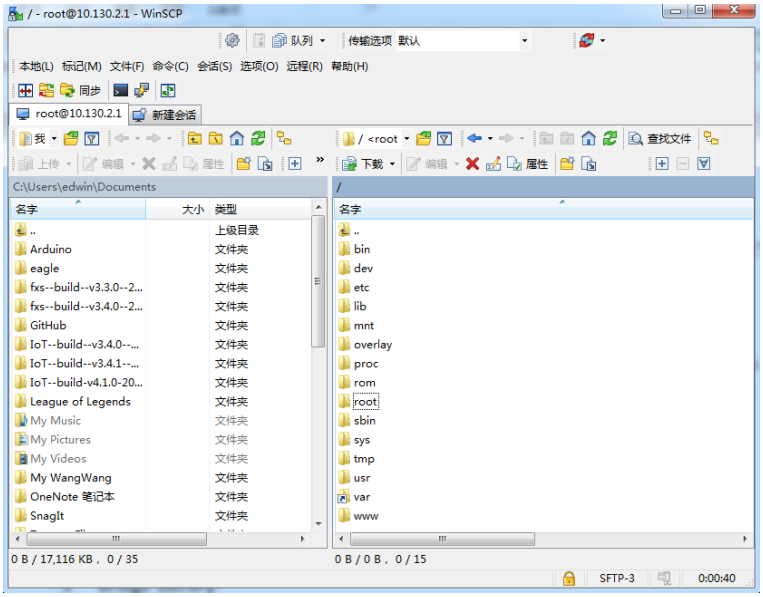
Password: **dragino** (default)

After log in, you will be in the Linux console and type command here.



10.2 Edit and Transfer files

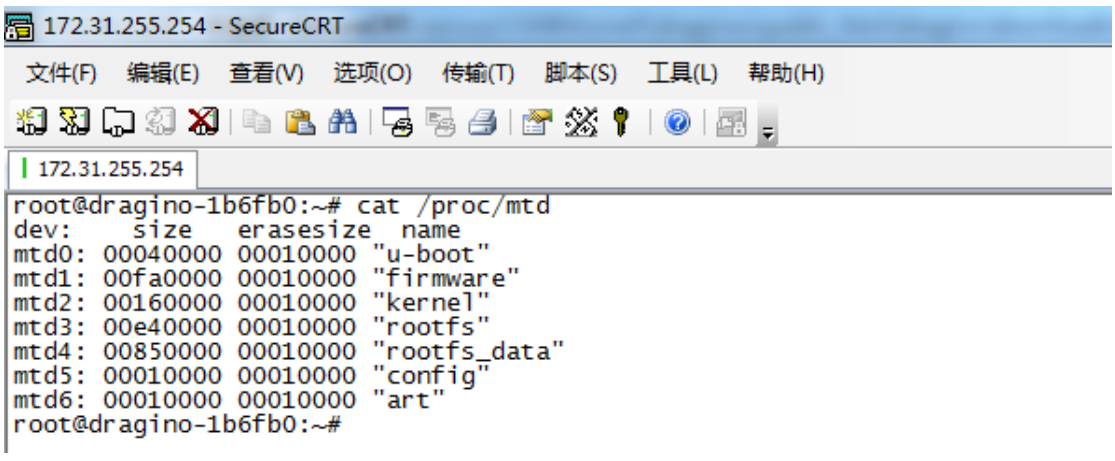
The LG01N support **SCP protocol** and has a built **SFTP server**. There are many ways to edit and transfer files using these two protocols. One of the easiest is through WinSCP utility. After access via WinSCP to the device, use can use a FTP alike window to drag / drop files to the LG01N or Edit the files directly in the windows. Screenshot is as below:



10.3 File System

The LG01N has a 16MB flash and a 64MB RAM. The /var and /tmp directory are in the RAM, contents stored in /tmp and /var will be erased after reboot the device. Other directories are in the flash and will keep after reboot.

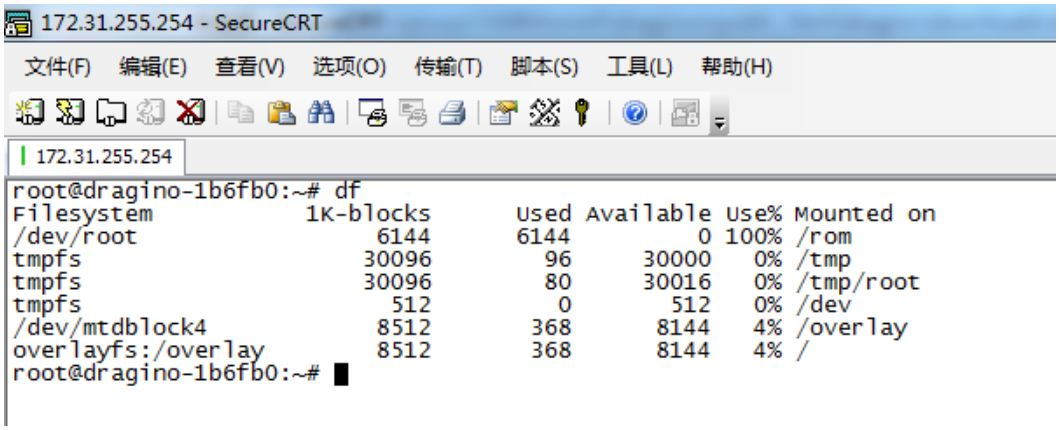
Use cat /proc/mtd to see all blocks/partitions.



- ✓ "u-boot" // for boot-loader
- ✓ "firmware" // combination of kernel & rootfs
- ✓ "kernel" // Linux kernel
- ✓ "rootfs" // Linux rootfs

- ✓ "rootfs_data" //inside rootfs, all data store here.
- ✓ "config" // a separate partition doesn't include file system
- ✓ "art" // radio data and board ID.

Use df command to see available flash & RAM:



tmpfs 30096 96 30000 0% /tmp // RAM: reset after reboot,

/dev/mtdblock4 8512 368 8144 4% /overlay //Flash: Remain after reboot

Reset to factory default:

mtd erase rootfs_data -r

Except /tmp and /var. all data will be store in flash. /tmp and /var are store in RAM

10.4 Package maintain system

LG01N uses **OPKG package maintain system**. There are more than 3000+ packages available in our package server for user to install for their applications. For example, if user wants to add iperf tool, they can install the related packages and configure LG01N to use iperf

Below is some examples opkgs command, more please refer **OPKG package maintain system**

In Linux Console run:

root@dragino-169d30:~# opkg update // to get the latest packages list

root@dragino-169d30:~# opkg list //shows the available packages

root@dragino-169d30:~# opkg install iperf // install iperf, it will auto install the required packages.

root@dragino-169d30:/etc/opkg# opkg install iperf

Installing iperf (2.0.12-1) to root...

Downloading http://downloads.openwrt.org/snapshots/packages/mips_24kc/base/iperf_2.0.12-1_mips_24kc.ipk

Installing uclibcxx (0.2.4-3) to root...

Downloading

http://downloads.openwrt.org/snapshots/packages/mips_24kc/base/uclibcxx_0.2.4-3_mips_24kc.ipk

Configuring uclibcxx.

Configuring iperf.

11. Upgrade Linux Firmware

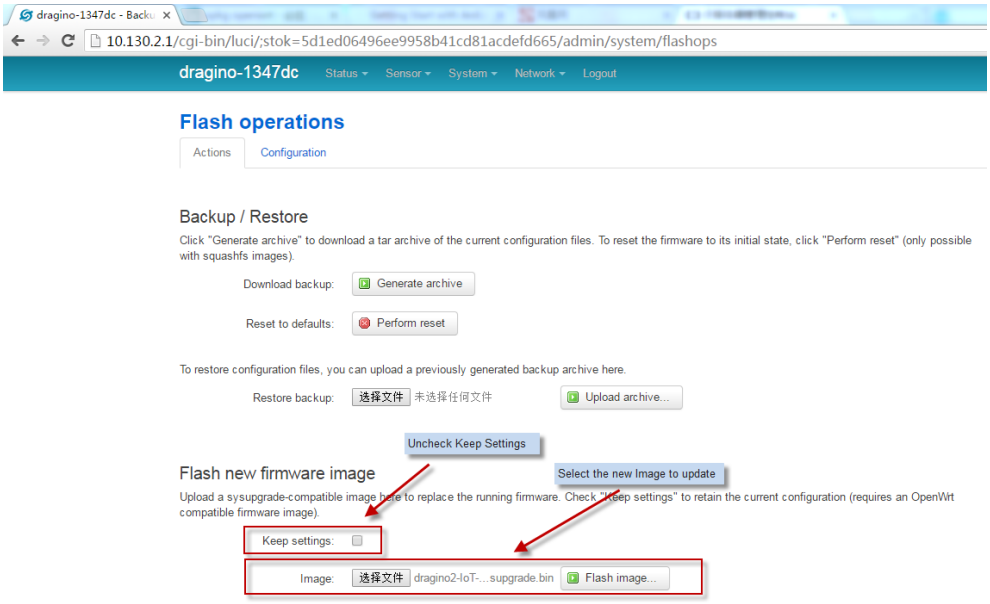
We keep improving the LG01N Linux side firmware for new features, bug fixes. The latest firmware can be found on LG01N Firmware & release note

The file named as **dragino-LG02_LG08----xxxxx-squashfs-sysupgrade.bin** is the upgrade Image. There are different methods to upgrade, as below:

11.1 Upgrade via Web UI

Go to the page: **Web --> System --> Back Up and flash firmware,** Select the image and click Flash Image, the image will be uploaded to the device and then click Process Update to upgrade.

System will auto boot to the new firmware after upgrade.



11.2 Upgrade via Linux console

SCP the firmware to the system **/var** directory and then run

root@OpenWrt:~# /sbin/sysupgrade -n /var/Your_Image

note: it is important to transfer the image in the /var directory, otherwise it may exceed the flash size.

12. FAQ

12.1 Why there is 433/868/915 version LoRa part?

Different country has different rules for the ISM band for using the LoRa. Although the LoRa chip can support a wide range of Frequency, we provide different version for best tune in the LoRa part. That is why we provide different version of LoRa.

12.2 What is the frequency range of LG01N LoRa part?

The chip used in the LoRa part is:

Version	LoRa IC	Support Frequency	Best Tune Frequency
433	Semtech SX1278	Band2(LF): 410 ~525Mhz Band3(LF): 137 ~175Mhz	433Mhz
868	Semtech SX1276	Band1(HF): 862 ~1020Mhz	868Mhz
915	Semtech SX1276	Band1(HF): 862 ~1020Mhz	915Mhz

User can set the LoRa within above frequency range in the software.

12.3 What does “Limited support on LoRaWAN”?

The base requirement to fully compatible with LoRaWAN protocol requires the gateway support 8 channels. The LG01N only support two channels and can only support limited LoRaWAN protocol.

Because of this limitation, if user wants to use a standard LoRaWAN device with LG01N, user has to modify this LoRaWAN node to run in single frequency to work with LG01N.

For example, in EU868 frequency plan, a standard LoRaWAN node will send the LoRa packet in hoping frequency (normally in 8 different frequencies). So a full compatible LoRaWAN gateway will be able to receive all packets while LG01N will miss 7 packets (according to the current software design, only one rx channel support).

So LG01N is not recommended for high density LoRa deployment or the LoRa Node can’t be configured to run in single frequency.

12.4 Can I develop my own LoRa protocol and other software for LG01N?

Yes, the fastest way to develop own software is through the SDK. The instruction is here:

https://github.com/dragino/openwrt_lede-18.06/blob/master/README.md#how-to-develop-a-c-software-before-build-the-image

12.5 Can I make my own firmware for LG01N? Where can I find the source code of LG01N?

Yes, User can make own firmware for LG01N for branding purpose or add customized application.

The LG01N source code and compile instruction can be found at:

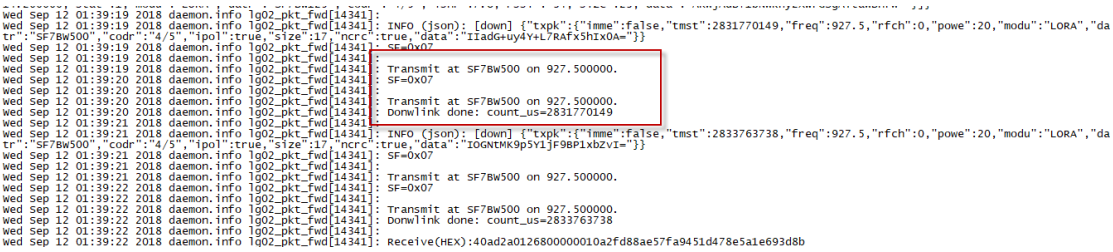
https://github.com/dragino/openwrt_lede-18.06

12.6 On OTAA mode, if I use the other frequency, how should I modify in the library?

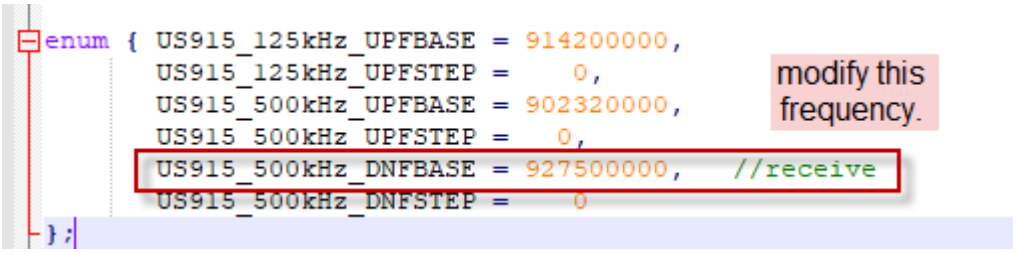
In page OTAA, We use frequency 904.6Mhz for sending. According the LoRaWAN protocol, if the device Join the network successfully, the server will downlink the reply. The different intervals of frequency, the receiving frequency of the end node is also different.

Ex1: We use 914.2Mhz frequency.

We can input the command: logread -f



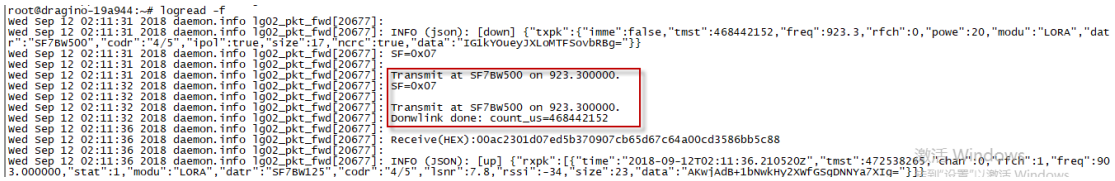
We should modify this on <lorabase.h>, save and re-upload the sketch.



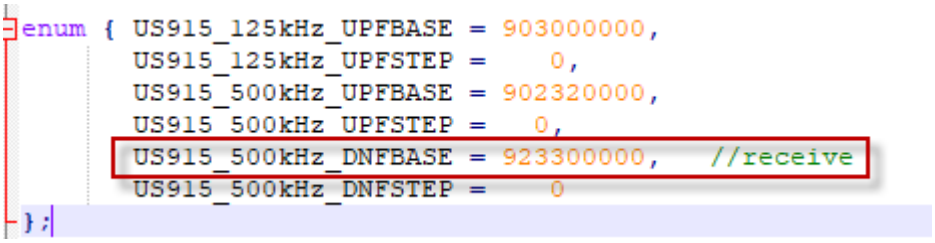
For the result:

Ex2: We use 903.0Mhz frequency

We can input the command: logread -f

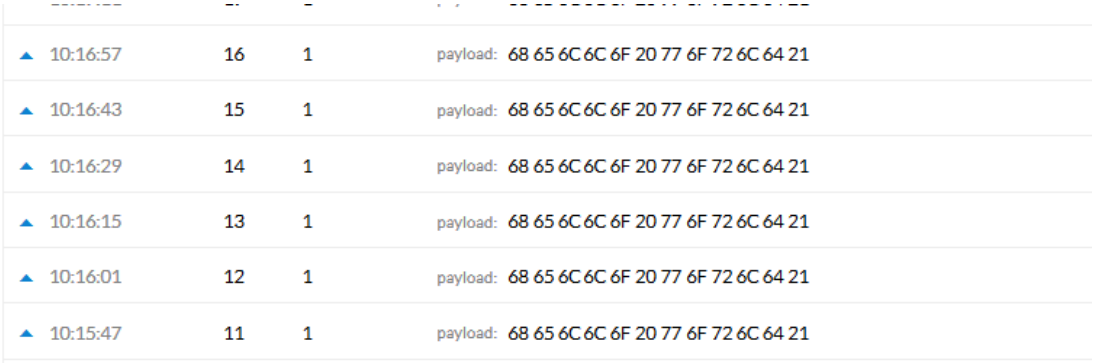


We should modify this on <lorabase.h>, save and re-upload the sketch.



```
enum { US915_125kHz_UPFBASE = 903000000,
       US915_125kHz_UPFSTEP =    0,
       US915_500kHz_UPFBASE = 902320000,
       US915_500kHz_UPFSTEP =   0,
       US915_500kHz_DNFBASE = 923300000,   //receive
       US915_500kHz_DNFSTEP =    0
};
```

For the result:



12.7 How can I reset the device to factory default?

User can reset the device to factory default in different ways:

Method 1:

Reset via Web UI. Click the button in Web UI --> System --> Back up / Flash firmware --> Perform Reset

Method 2:

Reset in Linux console, command is below:

root@dragino-1b8288:~# firstboot

This will erase all settings and remove any installed packages. Are you sure? [N/y]

y

/dev/mtdblock4 is mounted as /overlay, only erasing files

root@dragino-1b8288:~# reboot

12.8 Can I control the LEDs?

Except the PWR LED and sensor LED. All other LEDs can be controlled by developer.

Control Globe LED:

ON: echo 1 > /sys/class/leds/dragino2\:red\:wlan/brightness

OFF: echo 0 > /sys/class/leds/dragino2\:red\:wlan/brightness

12.9 Can I upgrade the LG01-P / LG01-S to LG01-N?

If user has LG01-P / LG01-S, they can upgrade their model to LG01-N by:

1) Change the Inside LoRa module to the module used in LG01-N.
2) Upgrade the firmware to the LG01-N firmware.

12.10 More FAQs about general LoRa questions

We keep updating more FAQs in our WiKi about some general questions. The link is here:

http://wiki.dragino.com/index.php?title=LoRa_Questions

13. Trouble Shooting

13.1 I get kernel error when install new package, how to fix?

In some case, when install package, it will generate kernel error such as below:

```
root@dragino-16c538:~# opkg install kmod-dragino2-si3217x_3.10.49+0.2-1_ar71xx.ipk
Installing kmod-dragino2-si3217x (3.10.49+0.2-1) to root...
Collected errors:
* satisfy_dependencies_for: Cannot satisfy the following dependencies for
kmod-dragino2-si3217x:
*       kernel (= 3.10.49-1-4917516478a753314254643facdf360a) *
 * opkg_install_cmd: Cannot install package kmod-dragino2-si3217x.
```

In this case, user can use the –force-depends option to install such package.

```
opkg install kmod-dragino2-si3217x_3.10.49+0.2-1_ar71xx.ipk --force-depends
```

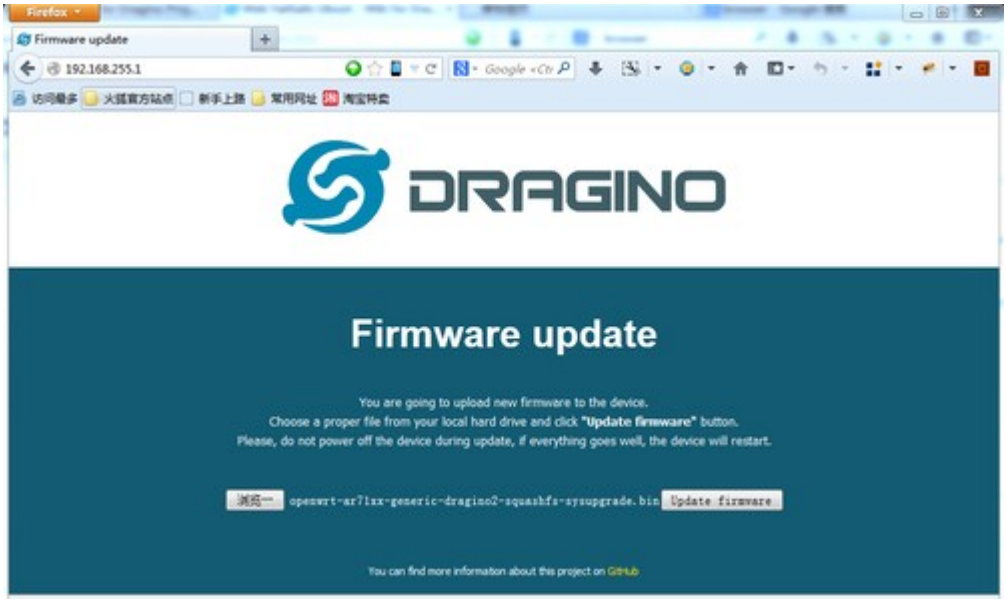
13.2 How to recover the LG01N if firmware crash

LG01N provides user a full control on its Linux system, it is possible that the device will brick and can't boot after improper modification in some booting files.

In this case, user can recover the whole Linux system by uploading a new firmware via Web Failsafe mode.

Procedure is as below:

1. Use a RJ45 cable to connect the PC to LG01N's LAN port directly.
2. Set the PC to ip 192.168.255.x, netmask 255.255.255.0
3. Pressing the toggle button and power on the device
4. All LEDs of the device will blink, release the toggle button after four blinks
5. All LEDs will then blink very fast once, this means device detect a network connection and enter into the web-failsafe mode. Your PC should be able to ping 192.168.255.1 after device enter this mode.
6. Open 192.168.255.1 in web broswer
7. Select a squashfs-sysupgrade type firmware and update firmware.



Note: If user sees all LEDs blink very fast in Step 5. This means the network connection is established. If in this case, PC still not able to see the web page, user can check:

- ✓ Try different browser.
- ✓ Check if your PC is in 192.168.255.x
- ✓ Check if you have connected two RJ45 cable to device, If so, remove the unused one

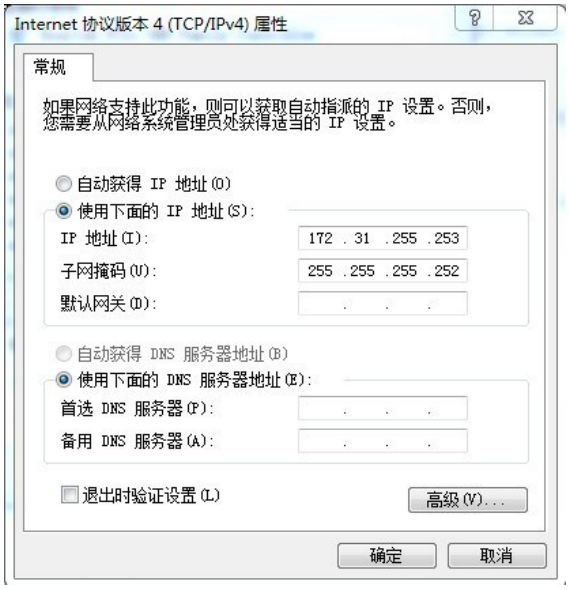
13.3 I configured LG01N for WiFi access and lost its IP. What to do now?

The LG01 has a fall-back ip in its LAN port. This IP is always enabled so user can use fall back ip to access LG01N no matter what the WiFi IP is. The fall back ip is useful for connect and debug the unit.

(Note: fallback ip can be disabled in the LAN and DHCP page)

Steps to connect via fall back IP:

1. Connect PC's Ethernet port to LG01's LAN port
2. Configure PC's Ethernet port has IP: 172.31.255.253 and netmask: 255.255.255.252
   As below photo:



3. In PC, use 172.31.255.254 to access LG01 via Web or Console.

14. Order Info

***PART: LG01N-XXX-YYY*:**

XXX: Frequency Band

- **433**: LoRa Gateway best tune to 433 MHz.
- **868**: LoRa Gateway best tuned to 868 MHz.
- **915**: LoRa Gateway best tuned to 915 MHz

YYY: 4G Cellular Option

- **EC25-E**: EMEA, Korea, Thailand, India.
- **EC25-A**: North America/ Rogers/AT&T/T-Mobile.
- **EC25-AU**: Latin America, New Zeland, Taiwan
- **EC25-J**: Japan, DOCOMO/SoftBank/ KDDI

More info about valid bands, please see EC25-E product page.

15. Packing Info

Package Includes:

- ✓ LG01N or OLG01N LoRa Gateway x 1
- ✓ Stick Antenna for LoRa RF part. Frequency is one of 433 or 868 or 915Mhz depends the model ordered
- ✓ Power Adapter: EU/AU/US type power adapter depends on country to be used
- ✓ Packaging with environmental protection paper box

Dimension and weight:

- ✓ Device Size: 12 x 8.5 x 3 cm
- ✓ Device Weight: 150g
- ✓ Package Size / pcs : 21.5 x 10 x 5 cm
- ✓ Weight / pcs : 360g
- ✓ Carton dimension: 45 x 31 x 34 cm. 36pcs per carton
- ✓ Weight / carton : 12.5 kg

16. Support

- Try to see if your questions already answered in the wiki.
- Support is provided Monday to Friday, from 09:00 to 18:00 GMT+8. Due to different timezones we cannot offer live support. However, your questions will be answered as soon as possible in the before-mentioned schedule.
- Provide as much information as possible regarding your enquiry (product models, accurately describe your problem and steps to replicate it etc) and send a mail to

support@dragino.com

17. Reference

- Source code for LG01N LoRa Gateway
  https://github.com/dragino/openwrt_lede-18.06

- OpenWrt official Wiki
  http://www.openwrt.org/

- Download of this manual or Update version
  http://www.dragino.com/downloads/index.php?dir=UserManual/LG02_OLG02/

- LMIC library for Arduino LoRaWAN end device use with LG01N.
  https://github.com/dragino/arduino-lmic