



LG01N/OLG01N LoRa Gateway User Manual

Document Version: 1.1

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. Introduction.....	5
1.1 What is LG01N & OLG01.....	5
1.2 Specifications.....	6
1.3 Features.....	8
1.4 System Structure	8
1.5 Applications.....	9
1.6 Hardware Variants	10
1.7 Install SIM card in 4G module.....	10
1.8 Firmware Change log	10
2. Access LG01N.....	11
3. Typical Network Setup	12
3.1 Overview	12
3.2 Use WAN port to access Internet.....	12
3.3 Access Internet as a WiFi Client.....	13
3.4 Use built-in 4G modem for internet access	15
3.5 Check Internet connection.....	17
4. Example 1: Configure as a LoRaWAN gateway – Limited LoRaWAN mode.....	18
4.1 Create a gateway in TTN Server.....	18
4.2 Configure LG01N Gateway	20
4.2.1 Configure to connect to LoRaWAN server	20
4.2.2 Configure LG01's Radio frequency.....	21
4.3 Create LoRa End Node.....	22
4.3.1 About Limited support for LoRaWAN	22
4.3.2 Preparation	23
4.3.3 Test with OTAA LoRa end node (LoRa Shield + UNO).....	24
4.3.4 Test with ABP LoRa end node (LoRa Shield + UNO).....	28
5. Example 2: Control LoRa radio directly as general LoRa transceiver	32
6. Example 3: MQTT Transfer Mode.....	35
6.1 What is MQTT API?	35
6.2 Step by Step Uplink Test.....	35

6.2.1	Simulate MQTT Publish via PC's MQTT tool	36
6.2.2	Try MQTT Publish with LG01N Linux command.....	37
6.2.3	Test LG01N MQTT routine service.	39
6.2.4	Configure LoRa End node	42
7.	Example 4: TCP IP Client Mode.....	43
8.	Example 5: Write a customized script.....	45
9.	Linux System	47
9.1	SSH Access for Linux console	47
9.2	Edit and Transfer files.....	48
9.3	File System.....	48
9.4	Package maintain system	50
10.	Upgrade Linux Firmware	51
10.1	Upgrade via Web UI.....	51
10.2	Upgrade via Linux console.....	51
11.	FAQ.....	52
11.1	Why there is 433/868/915 version LoRa part?.....	52
11.2	What is the frequency range of LG01N LoRa part?	52
11.3	What does "Limited support on LoRaWAN"?	52
11.4	Can I develop my own software for LG01N?.....	53
11.5	Can I make my own firmware for LG01N? Where can I find the source code of LG01N?	53
11.6	On OTAA mode, if I use the other frequency, how should I modify in the library?.....	53
11.7	How can I reset the device to factory default?	54
11.8	More FAQs about general LoRa questions.....	55
11.9	Can I upgrade the LG01-P / LG01-S to LG01-N?.....	55
12.	Trouble Shooting.....	56
12.1	I get kernel error when install new package, how to fix?	56
12.2	How to recover the LG01N if firmware crash.....	57
12.3	I configured LG01N for WiFi access and lost its IP. What to do now?	58
13.	Order Info	59
14.	Packing Info	59

15. Support.....	59
16. Reference.....	60

1. Introduction

1.1 What is LG01N & OLG01

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.

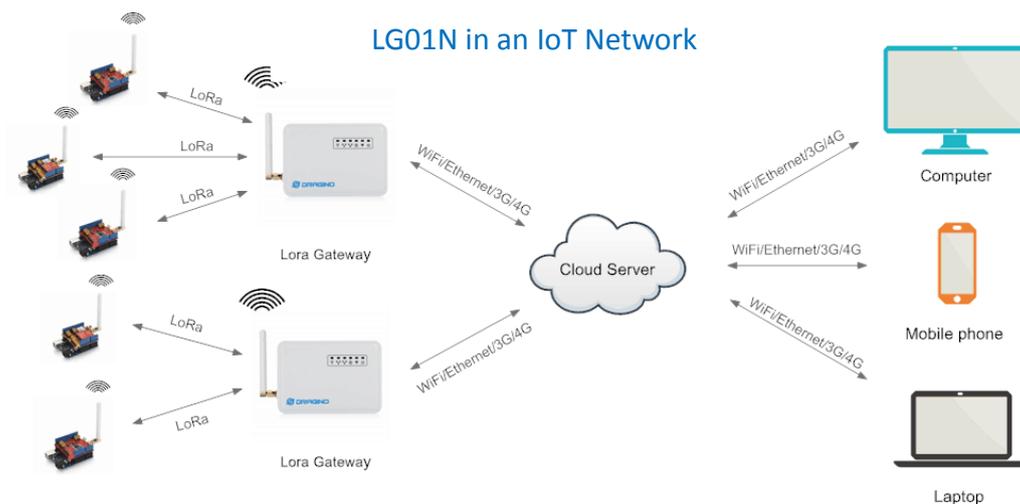
LG01N & OLG01N have rich internet connection method such as **WiFi interface, Ethernet port and 3G/4G Cellular**. These Interfaces provide flexible methods for users to connect their sensor networks to Internet.

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
- Frequency 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):

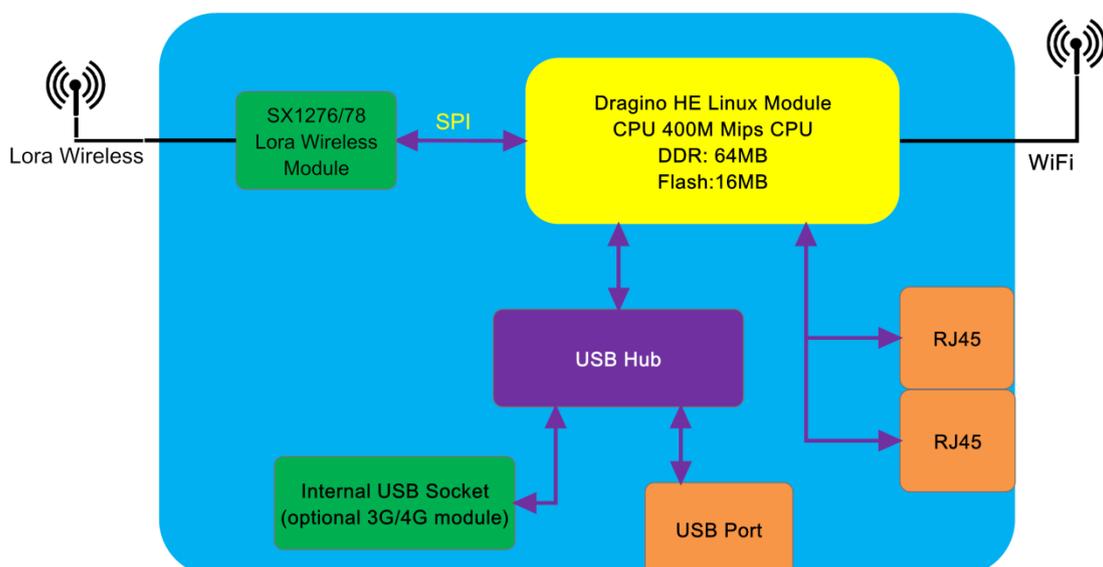
- Quectel [EC25 LTE module](#)
- Micro SIM Slot
- Internal 4G Antenna + External 4G Sticker Antenna.
- Up to 150Mbps downlink and 50Mbps uplink data rates
- Worldwide LTE,UMTS/HSPA+ and GSM/GPRS/EDGE coverage
- MIMO technology meets demands for data rate and link reliability in modem wireless communication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

Dragino Lora Gateway for IoT 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 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)



1.8 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

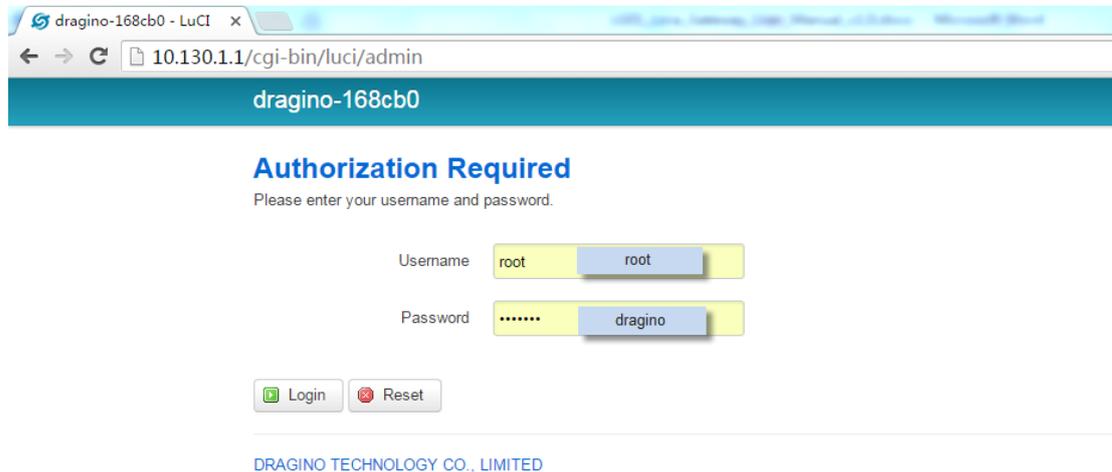
<http://10.130.1.1/cgi-bin/luci/admin>

User will see the login interface of LG01N.

The account for Web Login is:

User Name: root

Password: dragino



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

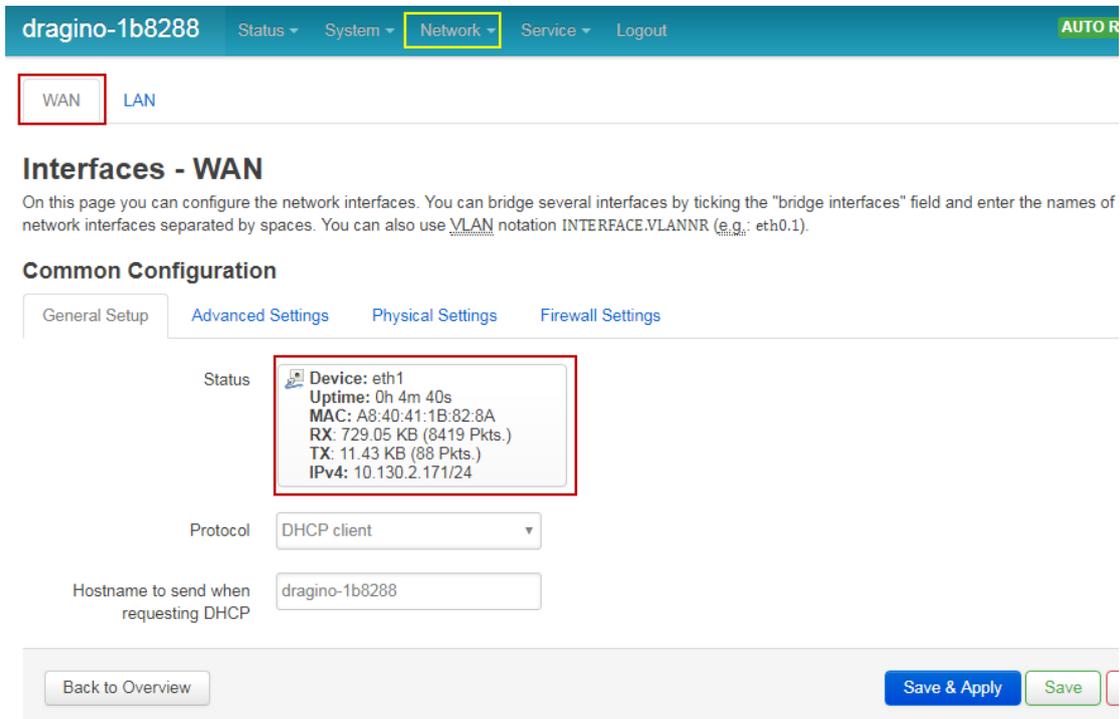
3.1 Overview

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:



dragino-1b8288 Status System **Network** Service Logout AUTO R

WAN LAN

Interfaces - WAN

On this page you can configure the network interfaces. You can bridge several interfaces by ticking the "bridge interfaces" field and enter the names of network interfaces separated by spaces. You can also use VLAN notation INTERFACE.VLANNR (e.g.: eth0.1).

Common Configuration

General Setup **Advanced Settings** Physical Settings Firewall Settings

Status Device: eth1
Uptime: 0h 4m 40s
MAC: A8:40:41:1B:82:8A
RX: 729.05 KB (8419 Pkts.)
TX: 11.43 KB (88 Pkts.)
IPv4: 10.130.2.171/24

Protocol DHCP client

Hostname to send when requesting DHCP dragino-1b8288

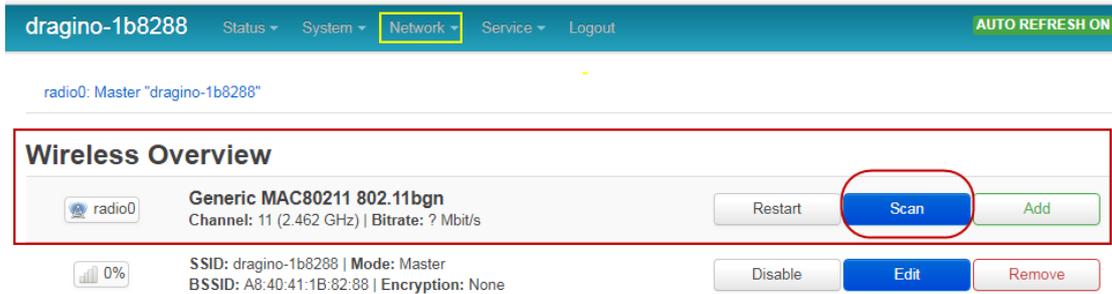
Back to Overview Save & Apply Save

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:

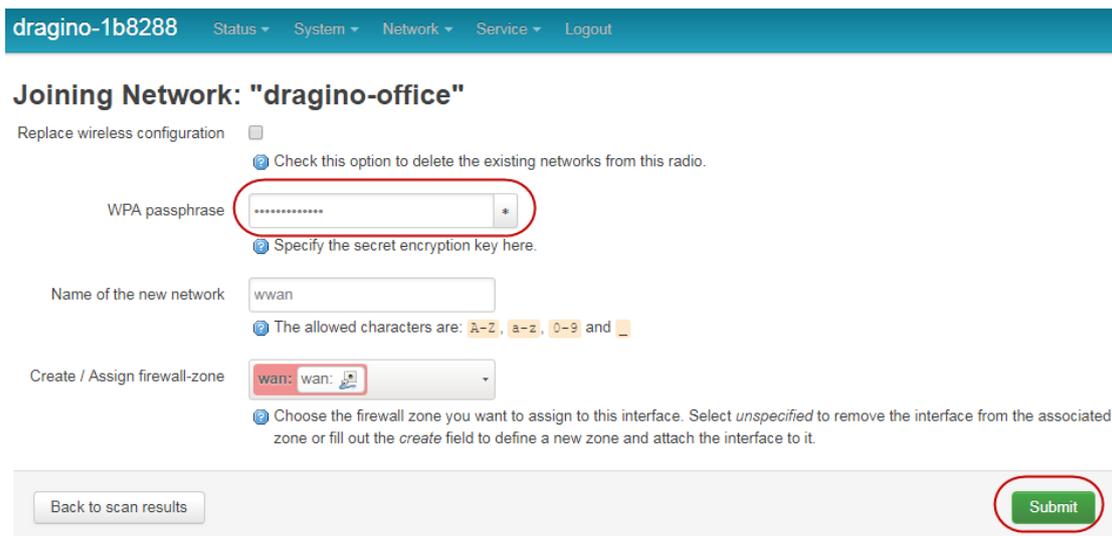
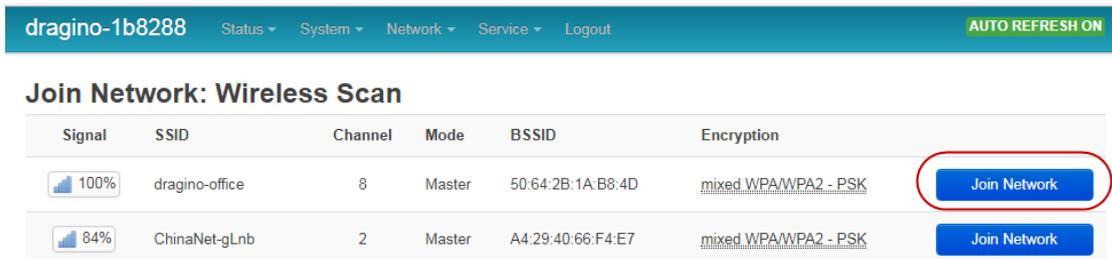
Step1:

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.

radio0: Master "dragino-1b8288"

Wireless Overview

radio0	Generic MAC80211 802.11bgn Channel: 11 (2.462 GHz) Bitrate: ? Mbit/s	Restart	Scan	Add
0%	SSID: dragino-1b8288 Mode: Master BSSID: A8:40:41:1B:82:88 Encryption: None	Disable	Edit	Remove
0%	SSID: dragino-office Mode: Client BSSID: 50:64:2B:1A:B8:4D Encryption: -	Disable	Edit	Remove

Associated Stations

Network	MAC-Address	Host	Signal / Noise	RX Rate / TX Rate
---------	-------------	------	----------------	-------------------

No information available

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

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

WAN WWAN LAN

Interfaces

LAN br-lan	Protocol: Static address Uptime: 2h 0m 4s MAC: A8:40:41:1B:82:8B RX: 1.40 MB (13346 Pkts.) TX: 2.79 MB (10321 Pkts.) IPv4: 10.130.1.1/24	Restart	Stop	Edit	Delete
WAN eth1	Protocol: DHCP client MAC: A8:40:41:1B:82:8A RX: 4.30 MB (51840 Pkts.) TX: 55.77 KB (429 Pkts.)	Restart	Stop	Edit	Delete
WWAN Client "dragino-office"	Protocol: DHCP client Uptime: 0h 6m 6s MAC: A8:40:41:1B:82:88 RX: 549.38 KB (5659 Pkts.) TX: 14.90 KB (94 Pkts.) IPv4: 10.130.2.169/24	Restart	Stop	Edit	Delete

Add new interface...

Save & Apply Save Reset

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

dragino-1b8288 Status System **Network** Service Logout AUTO REFRESH ON

WAN WWAN LAN

Interfaces

Interface	Protocol	Uptime	MAC	RX	TX	IPv4	Buttons
LAN (br-lan)	Static address	0h 19m 52s	A8:40:41:1B:82:8B	168.77 KB (1696 Pkts.)	398.89 KB (1165 Pkts.)	10.130.1.1/24	Restart Stop Edit Delete
WAN (eth1)	DHCP client		A8:40:41:1B:82:8A	0 B (0 Pkts.)	0 B (0 Pkts.)		Restart Stop Edit Delete
WWAN (Client "dragino-office")	DHCP client		A8:40:41:1B:82:88	0 B (0 Pkts.)	0 B (0 Pkts.)		Restart Stop Edit Delete

Add New Interface Add new interface...

Save & Apply Save Reset

dragino-1b8288 Status System Network Service Logout

Create Interface

Name of the new interface: Cellular
The allowed characters are: A-Z, a-z, 0-9 and _

Note: interface name length
Maximum length of the name is 15 characters including the automatic protocol/bridge prefix (br-, gin4-, pppoe- etc.)

Protocol of the new interface: UMTS/GPRS/EV-DO
Choose UMTS/GPRS/EV-DO

Cancel Submit

Step 2: Configure cellular interface

dragino-1b8288 Status System Network Service Logout UNSAVED C

Interfaces - CELLULAR

On this page you can configure the network interfaces. You can bridge several interfaces by ticking the "bridge interfaces" field and network interfaces separated by spaces. You can also use VLAN notation INTERFACE.VLANNR (e.g.: eth0.1).

Common Configuration

General Setup **Advanced Settings** Firewall Settings

Status Device: 3g-Cellular
RX: 0 B (0 Pkts.)
TX: 0 B (0 Pkts.)

Protocol UMTS/GPRS/EV-DO

Modem device /dev/ttyUSB2 Use ttyUSB2 to dial up

Service Type UMTS/GPRS

APN 3gnet Different provider has different APN

PIN

PAP/CHAP username Some provider may need additional user info

PAP/CHAP password

Dial number *99***1#

Step 3: Check Result

dragino-1b8288 Status System Network Service Logout AUTO REFRESH ON

WAN WWAN **CELLULAR** LAN

Interfaces

CELLULAR
3g-Cellular

Protocol: UMTS/GPRS/EV-DO
Uptime: 0h 0m 49s
MAC: 00:00:00:00:00:00
RX: 116 B (6 Pkts.)
TX: 680 B (16 Pkts.)
IPv4: 10.160.169.29/32 Get IP from provider means dial up

Restart Stop Edit Delete

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:

10.130.1.1 - SecureCRT

文件(F) 编辑(E) 查看(V) 选项(O) 传输(T) 脚本(S) 工具(L) 帮助(H)

10.130.1.1

```
root@dragino-1b8288:~# lsusb
Bus 001 Device 003: ID 2c7c:0125
Bus 001 Device 002: ID 1a40:0101 Terminus Technology, Inc. Hub
Bus 001 Device 001: ID 1d6b:0002 Linux Foundation 2.0 USB Hub
root@dragino-1b8288:~#
root@dragino-1b8288:~#
root@dragino-1b8288:~#
```

use lsusb command

This is the 4G modem

3.5 Check Internet connection

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

dragino-1b8288 Status ▾ System ▾ **Network ▾** Service ▾ Logout

Diagnostics

Network Utilities

openwrt.org openwrt.org openwrt.org

IPv4 ▾ **Ping** **Traceroute** **Nslookup**

Install iputils-traceroute6 for IPv6 traceroute

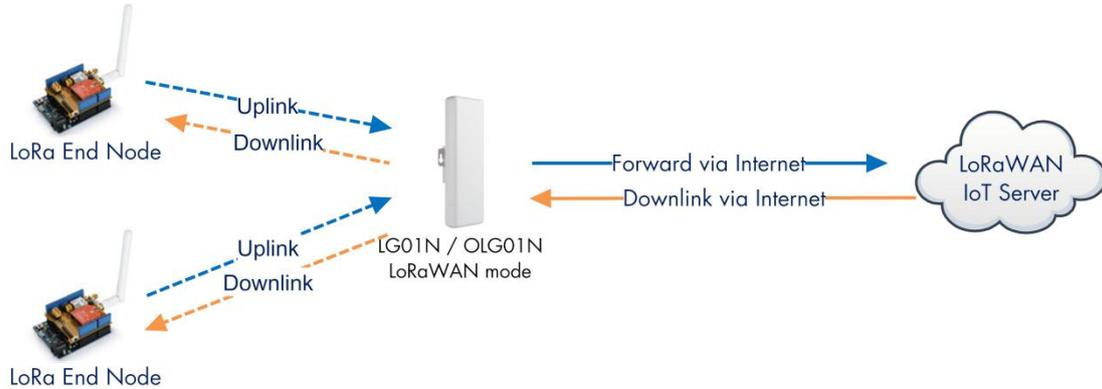
```
PING openwrt.org (139.59.209.225): 56 data bytes
64 bytes from 139.59.209.225: seq=0 ttl=45 time=386.898 ms
64 bytes from 139.59.209.225: seq=1 ttl=45 time=401.656 ms
64 bytes from 139.59.209.225: seq=2 ttl=45 time=387.708 ms
64 bytes from 139.59.209.225: seq=3 ttl=45 time=378.894 ms
64 bytes from 139.59.209.225: seq=4 ttl=45 time=384.156 ms

--- openwrt.org ping statistics ---
5 packets transmitted, 5 packets received, 0% packet loss
round-trip min/avg/max = 378.894/387.862/401.656 ms
```

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 multiply 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:

dragino-1b6fc4
Status ▾ System ▾ Network ▾ Service ▾ Logout

LoRa Gateway Settings

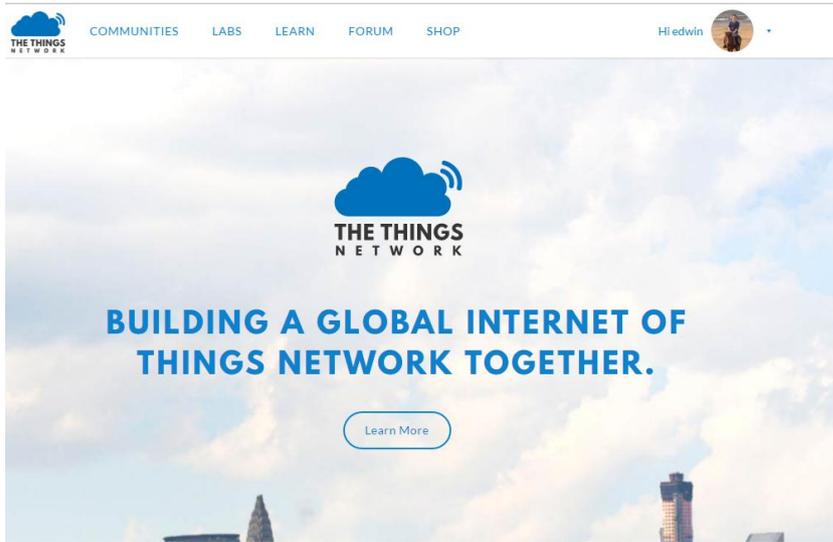
Configuration to communicate with LoRa devices and LoRaWAN server

LoRaWAN Server Settings

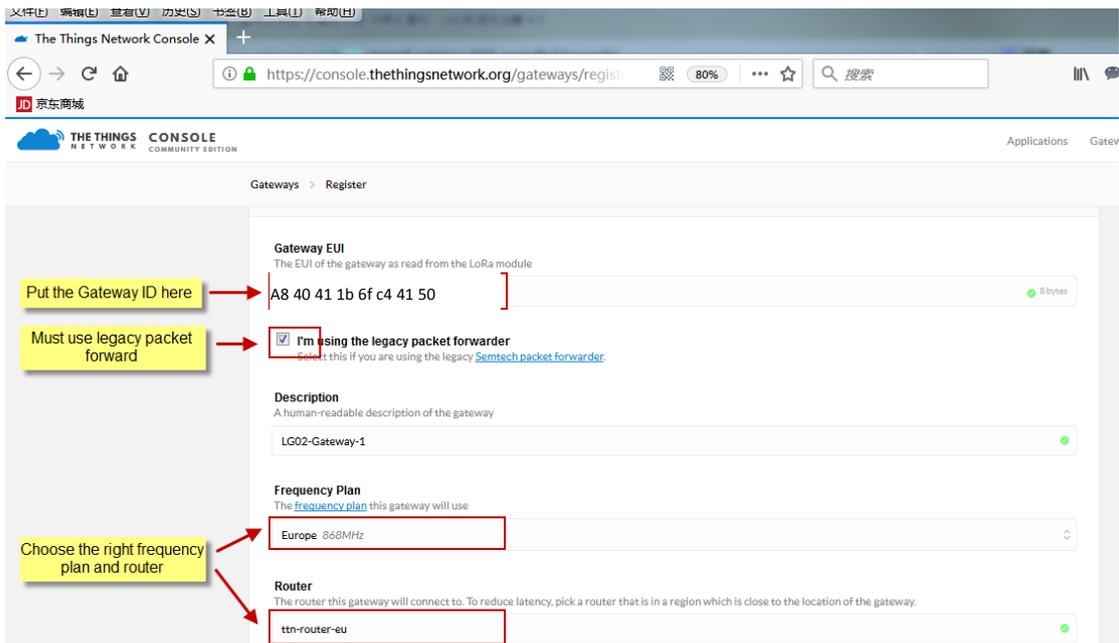
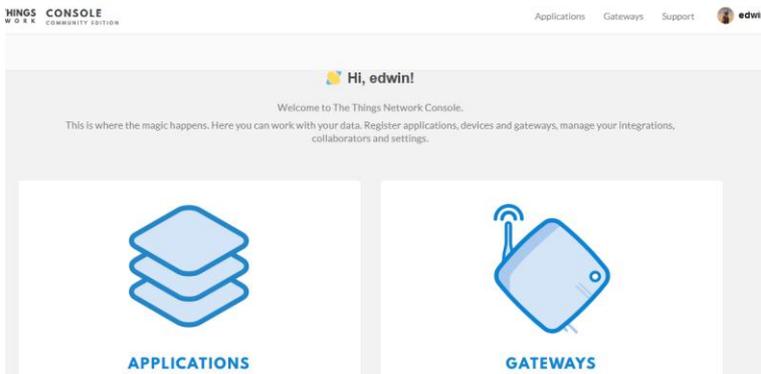
Service Provider	<input type="text" value="The Things Network"/>
Server Address	<input type="text" value="ttn-router-eu"/>
Server Port	<input type="text" value="1700"/>
Gateway ID	<input style="border: 2px solid red;" type="text" value="a840411b6fc44150"/>
Mail Address	<input type="text" value="dragino-1b6fc4@dragino.com"/>
Latitude	<input type="text" value="22.73"/>
Longitude	<input type="text" value="114.23"/>
RadioMode	<input type="text" value="A for RX, B for TX"/>

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.

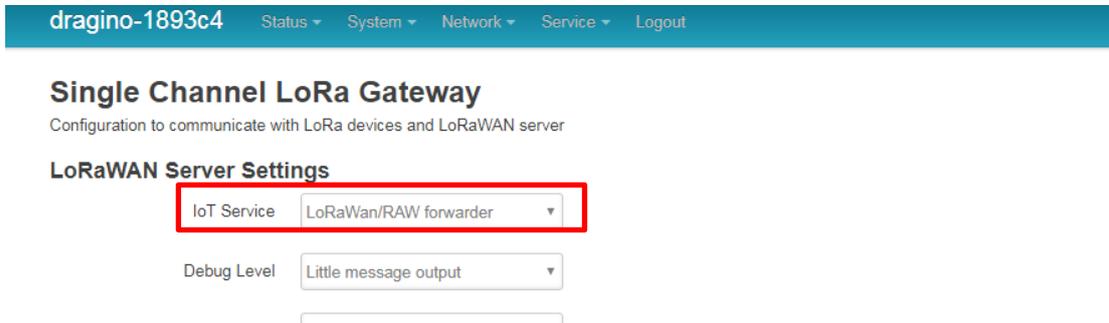


4.2 Configure LG01N Gateway

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.

LoRa Gateway Settings

Configuration to communicate with LoRa devices and LoRaWAN server

LoRaWAN Server Settings

Service Provider	The Things Network
Server Address	ttn-router-eu
Server Port	1700
Gateway ID	a840411b
Mail Address	edwin@dragino.com
Latitude	22.73
Longitude	114.23

Check Result

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

The screenshot shows the 'GATEWAY OVERVIEW' page in the TTN Console. The gateway ID is eui-a840411b8268ffff. The description is 'LG02-Gateway-1'. The owner is 'edwin'. The status is 'connected'. The frequency plan is 'Europe 868MHz'. The router is 'ttn-router-eu'. The gateway key is displayed in a masked format. The last seen time is '23 seconds ago'. There are 0 received and 0 transmitted messages.

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.

Radio Settings

Radio settings for Channel

Frequency (Unit:Hz)	<input type="text" value="868100000"/>
Spreading Factor	<input type="text" value="SF7"/>
Coding Rate	<input type="text" value="4/5"/>
Signal Bandwidth	<input type="text" value="125 kHz"/>
Preamble Length	<input type="text" value="8"/> <small>Length range: 6 ~ 65536</small>
LoRa Sync Word	<input type="text" value="52"/> <small>Value 52(0x34) for LoRaWAN</small>
Encryption Key	<input type="text" value="Encryption Key"/>

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
    
```

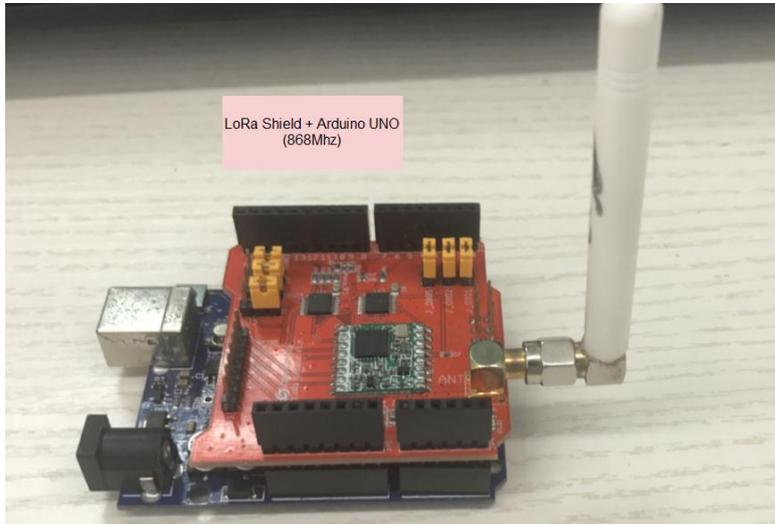
So the LG01N will only able to receive the 868100000, SF7 packet and will not receive others. Means only one packet will arrive the TTN server in every 8 packet sent from the LoRaWAN end node.

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_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
#elseif 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

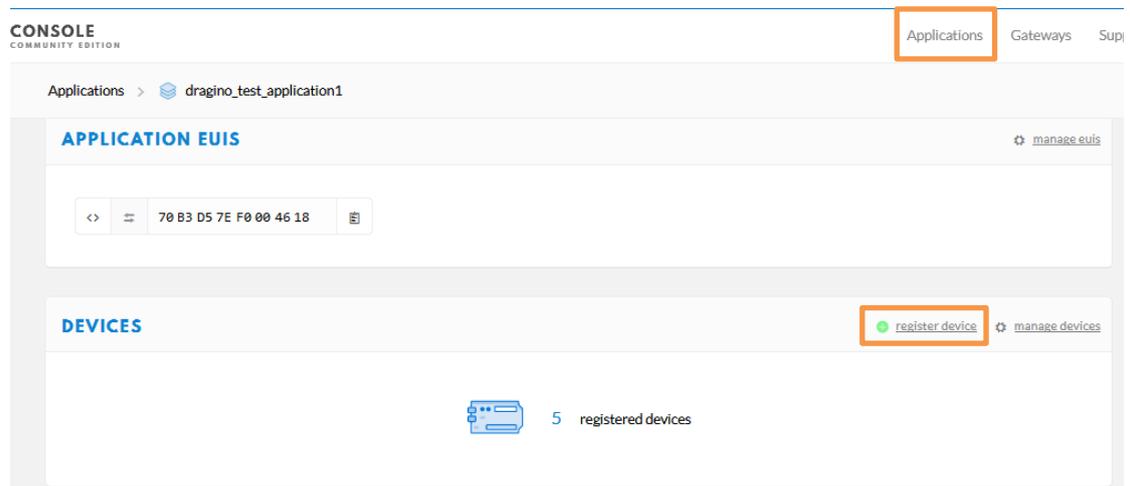
uncomment this for LG01 / LG02

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

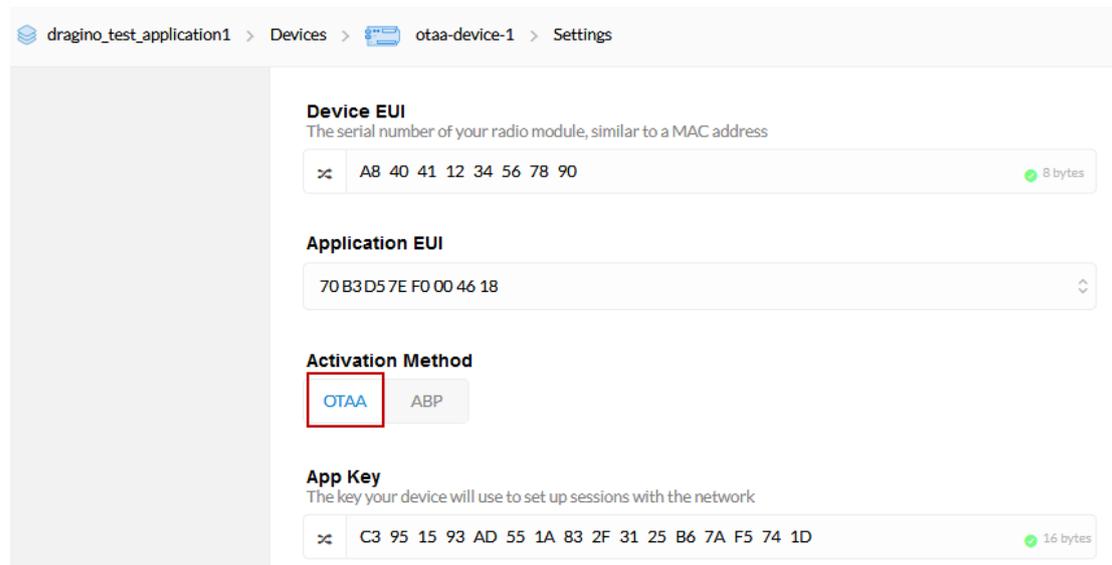
The TXSF is now set to default value:
 US915/AS923 : 923300000 , SF12BW500
 EU868: 869525000, SF12BW125

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

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



The screenshot shows the 'CONSOLE COMMUNITY EDITION' interface. At the top right, there are navigation tabs: 'Applications' (highlighted with an orange box), 'Gateways', and 'Supp'. Below the navigation, the breadcrumb path is 'Applications > dragino_test_application1'. The main content area is divided into two sections: 'APPLICATION EUIs' and 'DEVICES'. The 'APPLICATION EUIs' section shows a single EUI: '70 B3 D5 7E F0 00 46 18'. The 'DEVICES' section shows '5 registered devices' and a 'register device' button (highlighted with an orange box) and a 'manage devices' link.



The screenshot shows the 'Settings' page for the device 'otaa-device-1'. The breadcrumb path is 'dragino_test_application1 > Devices > otaa-device-1 > Settings'. The settings are as follows:

- Device EUI:** The serial number of your radio module, similar to a MAC address. Value: A8 40 41 12 34 56 78 90 (8 bytes).
- Application EUI:** Value: 70 B3D57E F0 00 46 18.
- Activation Method:** Two buttons are shown: 'OTAA' (highlighted with a red box) and 'ABP'.
- App Key:** The key your device will use to set up sessions with the network. Value: C3 95 15 93 AD 55 1A 83 2F 31 25 B6 7A F5 74 1D (16 bytes).

Step 2: Input keys into Arduino Sketch.

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

The screenshot shows the configuration interface for a LoRaWAN device. The 'Activation Method' is set to 'OTAA'. A yellow callout box points to the 'Device EUI', 'Application EUI', and 'App Key' fields, stating: "OTAA needs Device EUI, Application Key, APP Key. While put these keys in Arduino-LMIC library. Make sure the Device EUI and Application key are in lsb". The 'Device EUI' field contains the hex value { 0x90, 0x78, 0x56, 0x34, 0x12, 0x41, 0x40, 0xA8 }. The 'Application EUI' field contains { 0x18, 0x46, 0x00, 0xF0, 0x7E, 0xD5, 0xB3, 0x70 }. The 'App Key' field contains { 0xC3, 0x95, 0x15, 0x93, 0xAD, 0x55, 0x1A, 0x83, 0x2F, 0x31, 0x25, 0xB6, 0x7A, 0x1D }.

The screenshot shows the Arduino IDE with the 'ttn-otaa' sketch loaded. A yellow callout box points to the key definitions in the code, stating: "Input Keys in Arduino Sketch". The code defines the following keys:

```

static const uint8_t PROGMEM APPEUI[8] = { 0x18, 0x46, 0x00, 0xF0, 0x7E, 0xD5, 0xB3, 0x70 };
void os_getArtEui (uint8_t* buf) { memcpy_P(buf, APPEUI, 8);}

// This should also be in little endian format see above.
static const uint8_t PROGMEM DEVEUI[8] = { 0x90, 0x78, 0x56, 0x34, 0x12, 0x41, 0x40, 0xA8 };
void os_getDevEui (uint8_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 ttnctl can be copied as-is.
// The key shown here is the semtech default key.
static const uint16_t PROGMEM APPKEY[16] = { 0xC3, 0x95, 0x15, 0x93, 0xAD, 0x55, 0x1A, 0x83, 0x2F, 0x31, 0x25, 0xB6, 0x7A, 0xF5, 0x74, 0x1D };
void os_getDevKey (uint8_t* buf) { memcpy_P(buf, APPKEY, 16);}
    
```

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

The screenshot shows the 'Tools' menu in the Arduino IDE. The 'Board' dropdown is set to 'Arduino/Genuino Uno'. The 'Port' dropdown is set to 'COM3'. A red box highlights the 'Board: "Arduino/Genuino Uno"' and 'Port: "COM3"' options.

Step 3: Check Result for OTAA

```

COM9
End Device Log

Starting
RXMODE_RSSI
205: engineUpdate, opmode=0x8
Packet queued
253: EV_JOINING
1211: engineUpdate, opmode=0xc
360990: engineUpdate, opmode=0xc
361325: IXMODE, freq=868100000, len=23, SF=7, BW=125, CR=4/5, IH=0
674948: RXMODE_SINGLE, freq=868100000, SF=7, BW=125, CR=4/5, IH=0
681489: EV_JOINED
681516: engineUpdate, opmode=0x808
682020: IXMODE, freq=868100000, len=26, SF=7, BW=125, CR=4/5, IH=0
744428: RXMODE_SINGLE, freq=868100000, SF=7, BW=125, CR=4/5, IH=0
807697: RXMODE_SINGLE, freq=868100000, SF=9, BW=125, CR=4/5, IH=0
866799: EV_IXCOMPLETE (includes waiting for RX windows)
866849: engineUpdate, opmode=0x900
    
```

Send a Join Request and get EV_JOINED means OTAA join success.

dragino-1b6fb0 Status System Network Service Logout

Logread

Gateway Log shows TX / RX LoRa Packet

FreqINFO Report RxTxJson ErrorMessage

```

(TXPK) [down] {"bpk":{"imme":false,"lms":3667234979,"freq":868.1,"rfch":0,"pove":14,"modu":"LORA","datr":"SF7BW125","codr":4/5,"ipof":true,"size":33,"ncrc":
Receive(HEX)20f675628bf6ba47b13d97b2d53841c4a2c3d2b3f5784edac0ee41c09b52aed37
(RXPK) [up] {"rpk":{"time":"2018-10-19T15:49:50.666162Z","lms":3666685421,"chan":0,"rfch":1,"freq":868.100000,"stat":1,"modu":"LORA","datr":"SF7BW125",
Receive(HEX)20f1f0
(RXPK) [up] {"rpk":{"time":"2018-10-19T15:49:51.310837Z","lms":3667330098,"chan":0,"rfch":1,"freq":868.100000,"stat":1,"modu":"LORA","datr":"SF7BW125",
Receive(HEX)00184600f07ed5b37090785634124140a83717b0b3a635
(RXPK) [up] {"rpk":{"time":"2018-10-19T15:51:12.288134Z","lms":3748307397,"chan":0,"rfch":1,"freq":868.100000,"stat":1,"modu":"LORA","datr":"SF7BW125",
(TXPK) [down] {"bpk":{"imme":false,"lms":3753307397,"freq":868.1,"rfch":0,"pove":14,"modu":"LORA","datr":"SF7BW125","codr":4/5,"ipof":true,"size":33,"ncrc":
Receive(HEX)202b875f11263b8feb06301731e6bb303649d809aeb7d2b01acd12a8a1555b35f
(RXPK) [up] {"rpk":{"time":"2018-10-19T15:51:16.768714Z","lms":3752787977,"chan":0,"rfch":1,"freq":868.100000,"stat":1,"modu":"LORA","datr":"SF7BW125",
Receive(HEX)202b875f11263b8feb06301731e6bb303649d809aeb7d2b01acd12a8a1555b35f
(RXPK) [up] {"rpk":{"time":"2018-10-19T15:51:17.419193Z","lms":3753438456,"chan":0,"rfch":1,"freq":868.100000,"stat":1,"modu":"LORA","datr":"SF7BW125",
Receive(HEX)40b32f0126800000169595d797e72e6ad20f6927984a9d0ae4a
(RXPK) [up] {"rpk":{"time":"2018-10-19T15:51:17.529606Z","lms":3753548866,"chan":0,"rfch":1,"freq":868.100000,"stat":1,"modu":"LORA","datr":"SF7BW125",
Receive(HEX)40b32f0126800100014c2175b7f5071dfead522d5abdbacc81c1
(RXPK) [up] {"rpk":{"time":"2018-10-19T15:52:20.726452Z","lms":3816745715,"chan":0,"rfch":1,"freq":868.100000,"stat":1,"modu":"LORA","datr":"SF7BW125",
Receive(HEX)40b32f0126800200013092d245bf71eabc672b4a9f96799a19c1
(RXPK) [up] {"rpk":{"time":"2018-10-19T15:53:24.029902Z","lms":3880049163,"chan":0,"rfch":1,"freq":868.100000,"stat":1,"modu":"LORA","datr":"SF7BW125",
Receive(HEX)40b32f0126800300018a0022e96ae280c87ed84b916191df32db
(RXPK) [up] {"rpk":{"time":"2018-10-19T15:54:27.346130Z","lms":3943365389,"chan":0,"rfch":1,"freq":868.100000,"stat":1,"modu":"LORA","datr":"SF7BW125",
    
```

https://console.thethingsnetwork.org/gateways/eui-a84

TTN Traffic Page shows the device status

Applications Gateways

Gateways > eui-a840411b6fb04150 > Traffic beta

Time	Freq	Mod	CR	SF	BW	SNR	Dev Addr	Payload Size
23:56:34	868.1	lora	4/5	SF 7	BW 125	61.7		26 bytes
23:55:30	868.1	lora	4/5	SF 7	BW 125	61.7		26 bytes
23:54:27	868.1	lora	4/5	SF 7	BW 125	61.7		26 bytes
23:53:24	868.1	lora	4/5	SF 7	BW 125	61.7		26 bytes
23:52:20	868.1	lora	4/5	SF 7	BW 125	61.7	1 dev addr: 26 01 2F B3	payload size: 26 bytes
23:51:17	868.1	lora	4/5	SF 7	BW 125	61.7	0 dev addr: 26 01 2F B3	payload size: 26 bytes
23:51:16	868.1	lora	4/5	SF 7	BW 125	71.9		
23:51:12	868.1	lora	4/5	SF 7	BW 125	61.7	70 B3D57E F0 00 46 18	dev eui: A8 40 41 12 34 56 78

TTN Send a Join reply. LoRa End node must get this packet to finish Join. The frequency shows use 868.1Mhz frequency, must be the same as the "LG02_DNWFREQ" in Lmic config.c file

TTN Get Join request

Immediately send a Uplink message after join success

Note: The LG02_DNWFREQ value in Arduino_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

Applications > dragino_test_application1 > Devices > edwintest1

DOWNLINK

Schedule a Downlink message.
In TTN --> Application --> Device --> Data

Scheduling: replace first last

FPort: 1

Confirmed

Payload: bytes fields 67 54 12 38 99 5 bytes

Send

Gateways > eui-a840411b6fc44150 > Traffic ^{beta}

uplink downlink join 0 bytes X pause clear

time	frequency	mod.	CR	data rate	airtime (ms)	cnt	
23:35:40	868.1	lora	4/5	SF 7 BW 125	61.7	819	dev addr: 26 01 1C 22 payload size: 26 bytes
23:34:39	868.1	lora	4/5	SF 7 BW 125	51.5	2	dev addr: 26 01 1C 22 payload size: 18 bytes
23:34:39	868.1	lora	4/5	SF 7 BW 125	61.7	818	dev addr: 26 01 1C 22 payload size: 26 bytes

Downlink message Send out from TTN after the next uplink message arrive.
In TTN --> Gateway --> Traffic

```

Receive(HEX):40221c0126802f03015560e4a9861fadf0a66f8f086c2cc5bd3c
(RXPK):[up]{"rxpk":{"time":"2018-10-07T15:31:29.364137Z","tmst":"8525017
Receive(HEX):40221c0126803003012cc5d43fee0674456b05da5b5e7e59572
(RXPK):[up]{"rxpk":{"time":"2018-10-07T15:32:32.725188Z","tmst":"9158627
Receive(HEX):40221c012680310301c630b7dd7eede7120a368c84411d68255b
(RXPK):[up]{"rxpk":{"time":"2018-10-07T15:33:36.001099Z","tmst":"979138697,"chan":0,"rfch":1,"freq":868.100000,"stat":1,"modu":"LORA","datr":"SF7BW125","c
Receive(HEX):40221c012680320301266ea6ebbcf6832a5fe707fca27310a7c2
(RXPK):[up]{"rxpk":{"time":"2018-10-07T15:34:39.279878Z","tmst":"1042417475,"chan":0,"rfch":1,"freq":868.100000,"stat":1,"modu":"LORA","datr":"SF7BW125","c
(TXPK):[down]{"txpk":{"immo":false,"tmst":"1043417475","freq":868.1,"rfch":0,"pwr":14,"modu":"LORA","datr":"SF7BW125","codr":"4/5","ipol":true,"size":18,"ncrc":
Receive(HEX):60221c012680020001ebce1d605dc3c3c649
(RXPK):[up]{"rxpk":{"time":"2018-10-07T15:34:39.994318Z","tmst":"1043131915,"chan":0,"rfch":1,"freq":868.100000,"stat":1,"modu":"LORA","datr":"SF7BW125"

```

Downlink message arrives gateway
In LG01N --> Service --> Logread

COM9

```

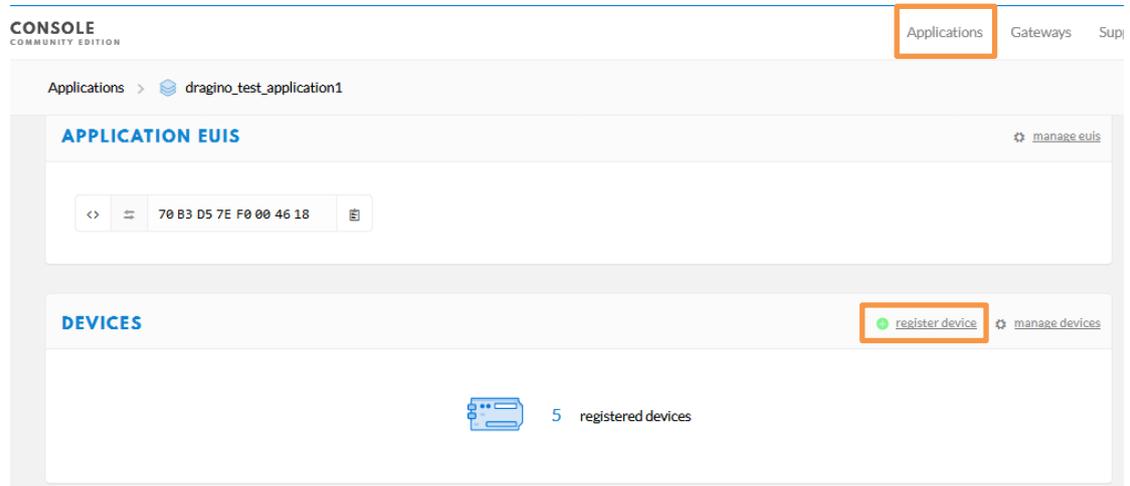
3217428074: engineUpdate, opmode=0x908
3217428598: IXMODE, freq=868100000, len=
Packet queued
3217494141: RXMODE_SINGLE, freq=868100000, SF=7, BW=125, CR=4/5, IH=0
3217557346: RXMODE_SINGLE, freq=868525000, SF=9, BW=125, CR=4/5, IH=0
-1077350851: EV_IXCOMPLETE (includes waiting for RX windows)
3217616511: engineUpdate, opmode=0x900
3221366512: engineUpdate, opmode=0x908
3221367037: IXMODE, freq=868100000, len=26, SF=7, BW=125, CR=4/5, IH=0
Packet queued
3221432515: RXMODE_SINGLE, freq=868100000, SF=7, BW=125, CR=4/5, IH=0
3221436475: Received downlink, window=RX1, port=1, ack=0
-1073530759: EV_IXCOMPLETE (includes waiting for RX windows)
Received
5
bytes of payload
3221436949: engineUpdate, opmode=0x800
3225186948: engineUpdate, opmode=0x808

```

Downlink message arrives LoRa Shield
In Arduino IDE --> Serial Monitor

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.



CONSOLE
COMMUNITY EDITION

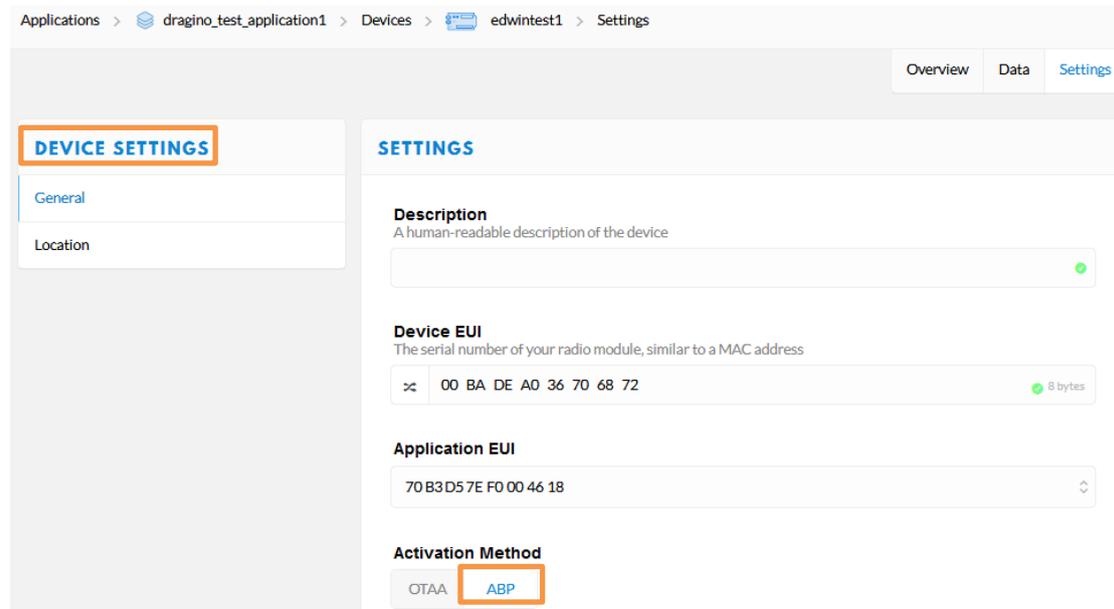
Applications > dragino_test_application1

APPLICATION EUIs [manage euis](#)

<> 70 B3 D5 7E F0 00 46 18

DEVICES [register device](#) [manage devices](#)

5 registered devices



Applications > dragino_test_application1 > Devices > edwintest1 > Settings

Overview Data Settings

DEVICE SETTINGS

General
Location

SETTINGS

Description
A human-readable description of the device

Device EUI
The serial number of your radio module, similar to a MAC address

00 BA DE A0 36 70 68 72 8 bytes

Application EUI
70 B3 D5 7E F0 00 46 18

Activation Method
OTAA **ABP**

Step 2: Input keys into Arduino Sketch.

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

Applications > dragino_test_application1 > Devices > edwintest1

TTN LoRaWAN End Device page

Application ID dragino_test_application1

Device ID edwintest1

Activation Method ABP

Device EUI <> 00 BA DE A0 36 70 68 72

Application EUI <> 70 B3 D5 7E F0 00 46 18

Make sure the Network Session Key and App Session Key are in MSB order

Device Address <> 26 01 1C 22

Network Session Key <> msb { 0x9A, 0xEA, 0xD0, 0x93, 0x06, 0xE3, 0x2B, 0x73, 0xDD, 0x54, 0x7B, 0x8B, 0xFF, 0xDC, 0x20, 0xF9 };

App Session Key <> msb { 0xB6, 0x07, 0x5B, 0xB5, 0xE4, 0xCE, 0x40, 0xA2, 0xA3, 0xEE, 0x7B, 0xDF, 0xDC, 0x23, 0x0E, 0x2B };

Arduino Sketch ttn-abp

```

ttn-abp
#include <lmic.h>
#include <hal/hal.h>
#include <SPI.h>

// LoRaWAN NwkSKey, network session key
// This is the default Semtech key, which is used by the early prototype IIII
// network
static const PROGMEM u1_t NWKSKKEY[16] = { 0x9A, 0xEA, 0xD0, 0x93, 0x06, 0xE3, 0x2B, 0x73, 0xDD, 0x54, 0x7B, 0x8B, 0xFF, 0xDC, 0x20, 0xF9 };

// LoRaWAN AppSKey, application session key
// This is the default Semtech key, which is used by the early prototype IIII
// network
static const u1_t PROGMEM APPSKKEY[16] = { 0xB6, 0x07, 0x5B, 0xB5, 0xE4, 0xCE, 0x40, 0xA2, 0xA3, 0xEE, 0x7B, 0xDF, 0xDC, 0x23, 0x0E, 0x2B };

// LoRaWAN end-device address (DevAddr)
static const u4_t DEVADDR = 0x26011C22 ; // <-- Change this address for every node!
    
```

Input the keys from TTN

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

- Auto Format
- Archive Sketch
- Fix Encoding & Reload
- Serial Monitor Ctrl+Shift+M
- Serial Plotter Ctrl+Shift+L
- WiFi101 Firmware Updater
- Board: "Arduino/Genuino Uno" >
- Port: "COM3" >
- Get Board Info
- Programmer: "AVRISP mkII" >
- Burn Bootloader

Step 3: Check Result for Uplink

```

COM9
Packet Sent From LoRa Shield.
In Arduino IDE --> Serial Monitor

3178173065: RXMODE_SINGLE, freq=869525000, SF=9, BW=125, CR=4/5, IH=0
-1116735050: EV_IXCOMPLETE (includes waiting for RX windows)
3178232311: engineUpdate, opmode=0x900
3181982310: engineUpdate, opmode=0x908
3181982835: TXMODE, freq=868100000, len=26, SF=7, BW=125, CR=4/5, IH=0
Packet queued
3182048313: RXMODE_SINGLE, freq=868100000, SF=7, BW=125, CR=4/5, IH=0
3182111581: RXMODE_SINGLE, freq=869525000, SF=9, BW=125, CR=4/5, IH=0
-1112796615: EV_IXCOMPLETE (includes waiting for RX windows)
    
```

/cgi-bin/luci/admin/gateway/lgwlog/3

dragino-1b6fc4 Status System Network Service Logout

Logread

Packet Arrive Gateway.
In page Service-->logread

```

FreqINFO Report RxTxJson ErrorMSG

Receive(HEX):40221c012680190301808a82034b8fc78df3dc7904968c850405
(RXPk): [up] [{"rxpk":{"time":"2018-10-07T15:08:16.815203Z","tmst":3754920098,"chan":0,"rfch":1,"freq":868.100000,"stat":1,"modu":"LORA","datr":"SF7BW125"},
Receive(HEX):40221c0126801a0301b8e0c0b06dd48c6f810faa2110301a3ba0
(RXPk): [up] [{"rxpk":{"time":"2018-10-07T15:09:20.146556Z","tmst":3818251446,"chan":0,"rfch":1,"freq":868.100000,"stat":1,"modu":"LORA","datr":"SF7BW125"},
Receive(HEX):40221c0126801b0301dc1f9e3ed124cb56b7351a517378118e7d
(RXPk): [up] [{"rxpk":{"time":"2018-10-07T15:10:23.388949Z","tmst":3881493842,"chan":0,"rfch":1,"freq":868.100000,"stat":1,"modu":"LORA","datr":"SF7BW125"},
Receive(HEX):40221c0126801c030106621e6fb4169d499d7b50b8f8c9a7f0fe
(RXPk): [up] [{"rxpk":{"time":"2018-10-07T15:11:26.714474Z","tmst":3944819367,"chan":0,"rfch":1,"freq":868.100000,"stat":1,"modu":"LORA","datr":"SF7BW125"},
Receive(HEX):40221c0126801d0301ca9fce94baebe3b4a9bcd09f95037b7b69
(RXPk): [up] [{"rxpk":{"time":"2018-10-07T15:12:30.024255Z","tmst":4008129142,"chan":0,"rfch":1,"freq":868.100000,"stat":1,"modu":"LORA","datr":"SF7BW125"},
Receive(HEX):40221c0126801e0301f727938d7254dd03180a4bc6b1763243e3
(RXPk): [up] [{"rxpk":{"time":"2018-10-07T15:13:33.339652Z","tmst":4071444547,"chan":0,"rfch":1,"freq":868.100000,"stat":1,"modu":"LORA","datr":"SF7BW125"},
    
```

Gateways > eui-a840411b6fc44150 > Traffic ^{beta}

Overview Traffic Settings

GATEWAY TRAFFIC ^{beta}

Packet Arrive TTN.
In TTN --> Gateway --> Traffic

uplink downlink join 0 bytes x pause clear

time	frequency	mod.	CR	data rate	airtime(ms)	cnt	
23:24:06	868.1	lora	4/5	SF 7 BW 125	61.7	808	dev addr: 26 01 1C 22 payload size: 26 bytes
23:23:03	868.1	lora	4/5	SF 7 BW 125	61.7	807	dev addr: 26 01 1C 22 payload size: 26 bytes
23:21:59	868.1	lora	4/5	SF 7 BW 125	61.7	806	dev addr: 26 01 1C 22 payload size: 26 bytes
23:20:56	868.1	lora	4/5	SF 7 BW 125	61.7	805	dev addr: 26 01 1C 22 payload size: 26 bytes

Applications > dragino_test_application1 > Devices > edwintest1 > Data

Overview Data Settings

APPLICATION DATA

Packet Arrive TTN Device Page.
In TTN --> Application --> Device --> Data

uplink downlink activation ack error

Filters

time	counter	port	
23:30:26	814	1	payload: 48 65 6C 6C 6F 2C 20 77 6F 72 6C 64 21
23:29:22	813	1	payload: 48 65 6C 6C 6F 2C 20 77 6F 72 6C 64 21

Step 4: Test Downlink

Applications > dragino_test_application1 > Devices > edwintest1

DOWNLINK

Schedule a Downlink message.
In TTN --> Application --> Device --> Data

Scheduling replace first last **FPort** 1 Confirmed

Payload bytes fields 67 54 12 38 99 5 bytes

Send

Gateways > eui-a840411b6fc44150 > Traffic ^{beta}

uplink downlink join 0 bytes × || pause 🗑 clear

time	frequency	mod.	CR	data rate	airtime (ms)	cnt	
▲ 23:35:40	868.1	lora	4/5	SF 7 BW 125	61.7	819	dev addr: 26 01 1C 22 payload size: 26 bytes
▼ 23:34:39	868.1	lora	4/5	SF 7 BW 125	51.5	2	dev addr: 26 01 1C 22 payload size: 18 bytes
▲ 23:34:39	868.1	lora	4/5	SF 7 BW 125	61.7	818	dev addr: 26 01 1C 22 payload size: 26 bytes

Downlink message Send out from TTN after the next uplink message arrive.
In TTN --> Gateway --> Traffic

```

Receive(HEX):40221c0126802f03015560e4a9861fadf0a66f8f086c2cc5bd3c
(RXPK):[up]{ "rxpk":{"time":"2018-10-07T15:31:29.364137Z","tmst":"8525017
Receive(HEX):40221c0126803003012cc5d43fee0674456b05da5b5e7e59572
(RXPK):[up]{ "rxpk":{"time":"2018-10-07T15:32:32.725188Z","tmst":"9158627
Receive(HEX):40221c012680310301c630b7dd7eede7120a368c84411d68255b
(RXPK):[up]{ "rxpk":{"time":"2018-10-07T15:33:36.001099Z","tmst":"979138697,"chan":0,"rfch":1,"freq":868.100000,"stat":1,"modu":"LORA","datr":"SF7BW125",
Receive(HEX):40221c012680320301266ea6ebbcf6832a5fe707fca27310a7c2
(RXPK):[up]{ "rxpk":{"time":"2018-10-07T15:34:39.279878Z","tmst":"1042417475,"chan":0,"rfch":1,"freq":868.100000,"stat":1,"modu":"LORA","datr":"SF7BW125",
(TXPK):[down]{ "txpk":{"imme":false,"tmst":"1043417475,"freq":868.1,"rfch":0,"pwr":14,"modu":"LORA","datr":"SF7BW125","codr":"4/5","ipol":true,"size":18,"ncrc":
Receive(HEX):60221c012680020001ebce1d605dc3c3c649
(RXPK):[up]{ "rxpk":{"time":"2018-10-07T15:34:39.994318Z","tmst":"1043131915,"chan":0,"rfch":1,"freq":868.100000,"stat":1,"modu":"LORA","datr":"SF7BW125",

```

Downlink message arrives gateway
In LG01N --> Service --> Logread

COM9

```

3217428074: engineUpdate, opmode=0x908
3217428598: IXMODE, freq=868100000, len=
Packet queued
3217494141: RXMODE_SINGLE, freq=868100000, SF=7, BW=125, CR=4/5, IH=0
3217557346: RXMODE_SINGLE, freq=868525000, SF=9, BW=125, CR=4/5, IH=0
-1077350851: EV_IXCOMPLETE (includes waiting for RX windows)
3217616511: engineUpdate, opmode=0x900
3221366512: engineUpdate, opmode=0x908
3221367037: IXMODE, freq=868100000, len=26, SF=7, BW=125, CR=4/5, IH=0
Packet queued
3221432515: RXMODE_SINGLE, freq=868100000, SF=7, BW=125, CR=4/5, IH=0
3221436475: Received downlink, window=RX1, port=1, ack=0
-1073530759: EV_IXCOMPLETE (includes waiting for RX windows)
Received
5
bytes of payload
3221436949: engineUpdate, opmode=0x800
3225186948: engineUpdate, opmode=0x808

```

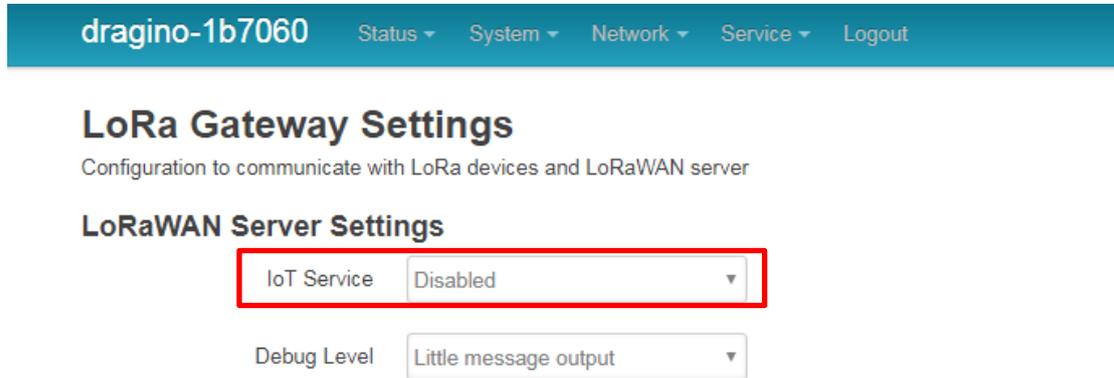
Downlink message arrives LoRa Shield
In Arduino IDE --> Serial Monitor

5. Example 2: Control LoRa radio directly as general LoRa transceiver

The LG02 has two separate LoRa radio, user can use them as raw LoRa transceiver.

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 frequency] (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

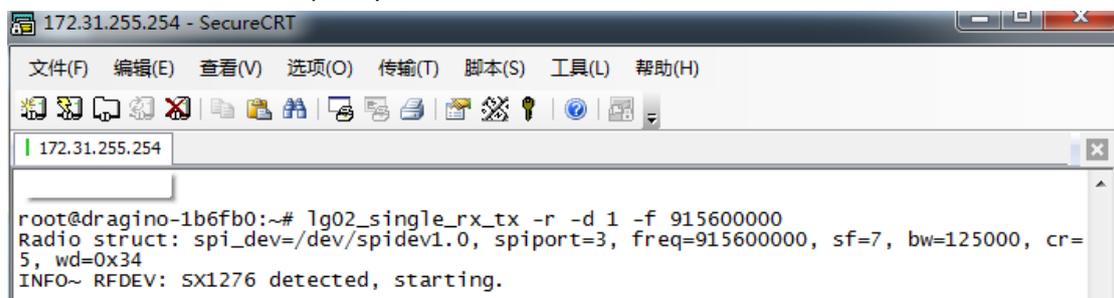
[-v] show version

[-h] show this help and exit Use Radio 1 to transmit:

Command:

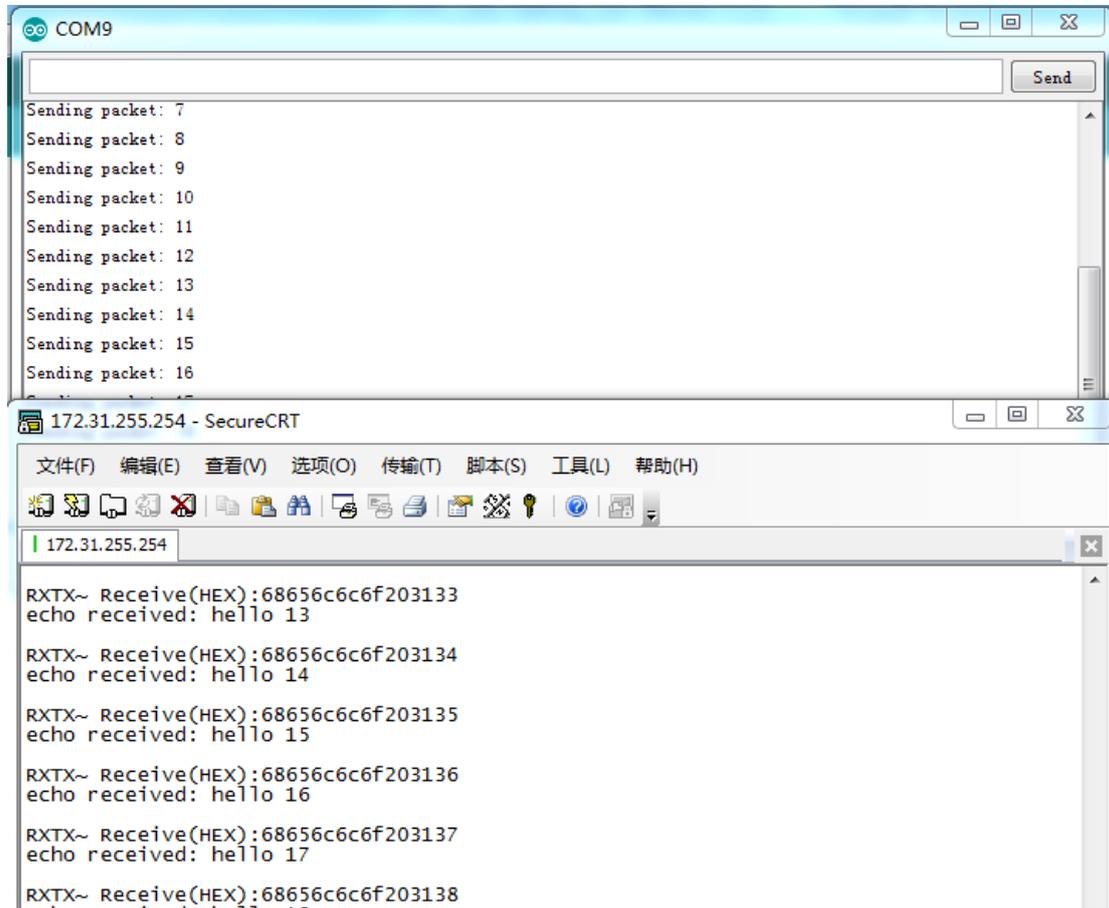
```
root@dragino-1b6fb0:~# lg02_single_rx_tx -r -d 2 -f 915600000
```

Use radio to receive at frequency 915600000



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 [LoRaReceiver](#).

Result screen shot:



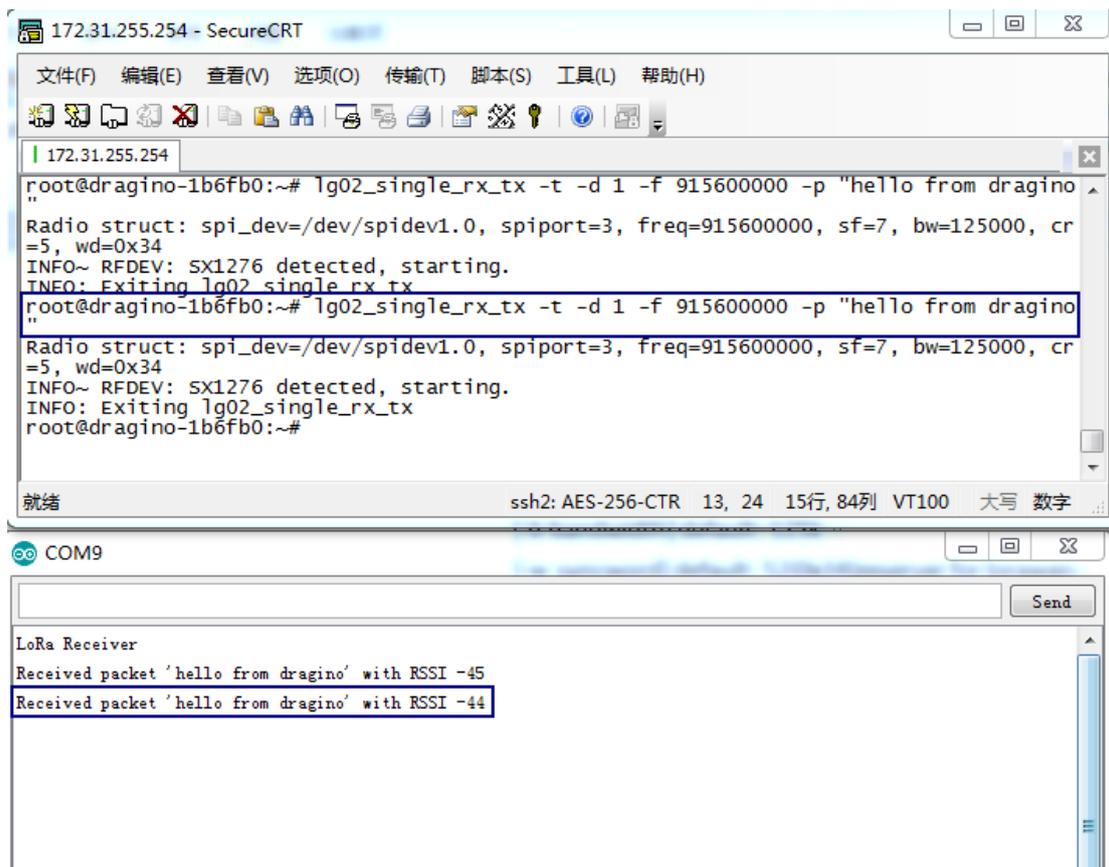
Step 3: Use lg02_single_rx_tx to transmit

Command:

```
root@dragino-1b6fb0:~# lg02_single_rx_tx -t -d 2 -f 915600000 -m "hello from dragino"
```

Use radio 2 to transmit a message at frequency 915600000

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](#).



6. Example 3: MQTT Transfer Mode

6.1 What is MQTT API?

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.

This example will show how to use LG01N to connect to the IoT Server via MQTT.

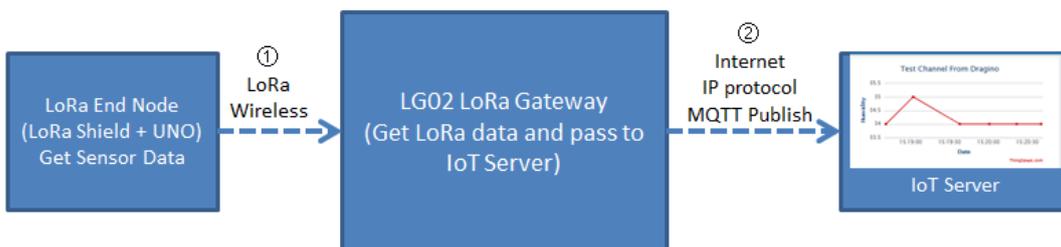
The server we use here is [ThingSpeak](#). They have the [MQTT API documented here](#). User can register an account in ThingSpeak and configure IoT server. The parameters we need here are:

- ✓ **Account User ID:** Can be found in "Account → My Profile → User ID"
- ✓ **MQTT API:** Can be found in "Account → My Profile → User ID"
- ✓ **Channel ID:** Which channel we want to publish data or subscribe.
- ✓ **Channel API:** the write API key for this channel.

6.2 Step by Step Uplink Test

In this section, we will try to program LG01N to uplink data to ThingSpeak. The data flow in this example is as below:

LoRa to MQTT Integration: Uplink Data Flow to ThingSpeak



Data Flow:

- ①: LoRa end node get data from sensor and send out via LoRa wireless protocol. The data is in a pre-define format.
- ②: LG02 get the sensor data and parse it into channel:data pair. Then LG02 will send out sensor data to IoT server via MQTT Publish.

We will try the step ② first, after it work as expect, we will integrate it with step ① for a complete uplink example.

6.2.1 Simulate MQTT Publish via PC's MQTT tool

This step is not necessary, it just to help user to understand the MQTT protocol and simulate the MQTT connection to ThingSpeak. Make sure the account info is valid and correct.

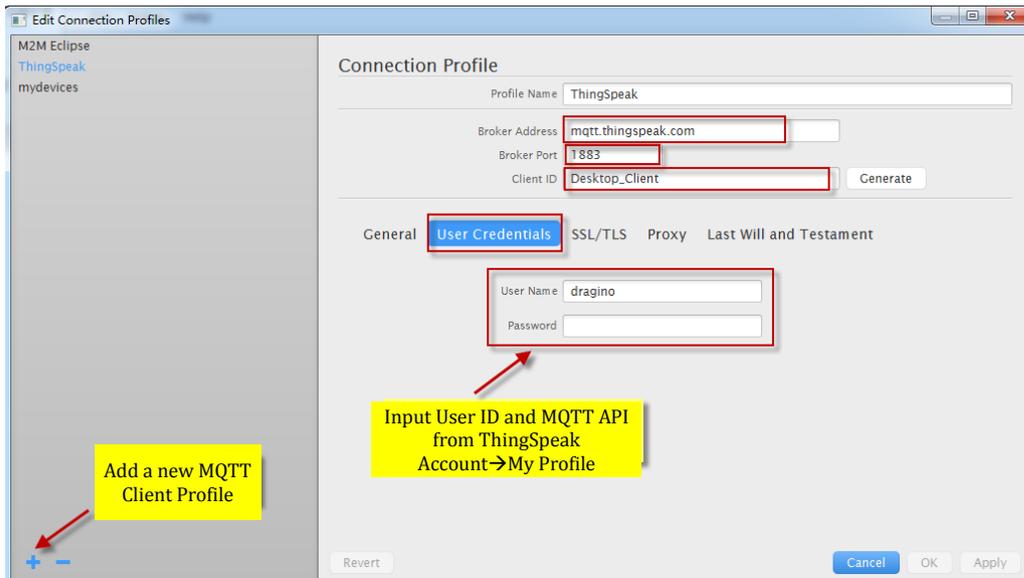
In the PC, download and install [MQTT.fx](#).

Open MQTT.fx and configure add a new MQTT client, as below:

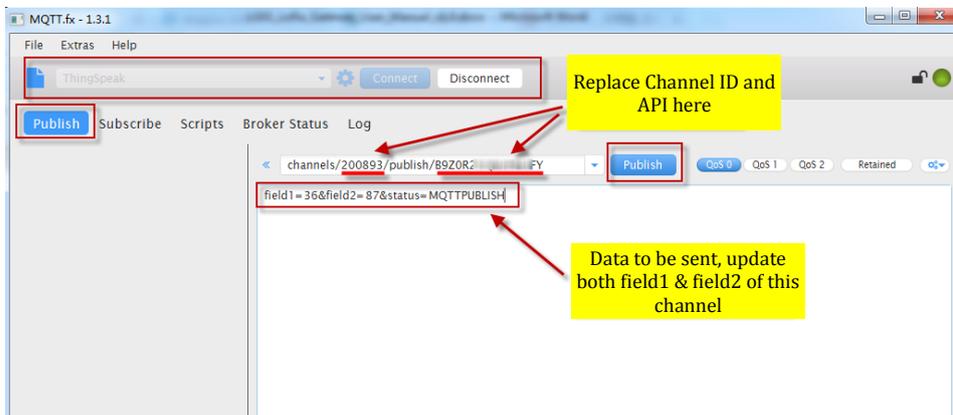
Broker Address: mqtt.thingspeak.com

Broker Port: 1883

Client ID: User Defined.



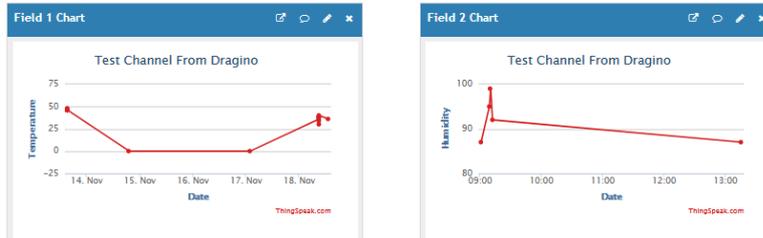
After add the profile, connect it and publish to the corresponding Channel with correct API key.



And we can see the update in the channel:

Channel Stats

Created: 11 months ago
 Updated: less than a minute ago
 Last entry: less than a minute ago
 Entries: 1762



6.2.2 Try MQTT Publish with LG01N Linux command

This step is not necessary; it is to help user to the basic command LG01N use for MQTT connection and will help for further debug when connection fails.

First, we need to make sure the LG01N has internet access. We can log in the SSH and ping an Internet address and see if it get through. As below:

```
172.31.255.254 - SecureCRT
文件(F) 编辑(E) 查看(V) 选项(O) 传输(T) 脚本(S) 工具(L) 帮助(H)
| Aliyun_美国服务器 | 172.31.255.254
root@dragino-146d78:~# ping www.163.com
PING www.163.com (58.63.233.35): 56 data bytes
64 bytes from 58.63.233.35: seq=0 ttl=54 time=8.231 ms
64 bytes from 58.63.233.35: seq=1 ttl=54 time=8.709 ms
64 bytes from 58.63.233.35: seq=2 ttl=54 time=8.313 ms
64 bytes from 58.63.233.35: seq=3 ttl=54 time=7.953 ms
64 bytes from 58.63.233.35: seq=4 ttl=54 time=8.539 ms
^C
--- www.163.com ping statistics ---
5 packets transmitted, 5 packets received, 0% packet loss
round-trip min/avg/max = 7.953/8.349/8.709 ms
root@dragino-146d78:~#
```

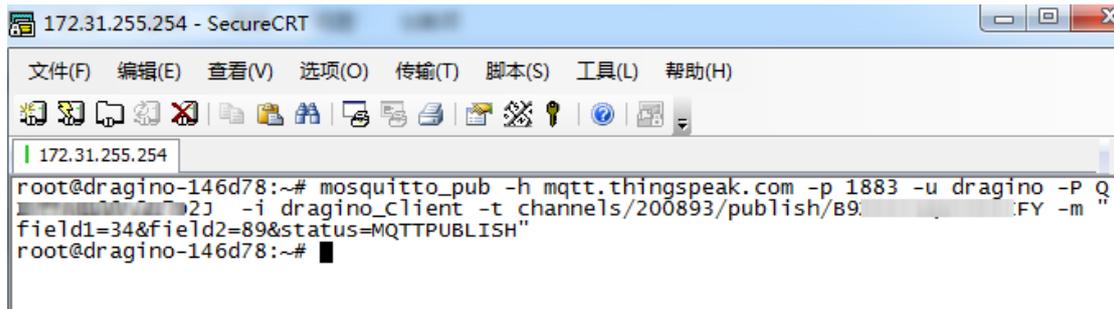
LG01N has built-in Linux tool [mosquitto_pub](#). We can use this command to publish the data to ThingSpeak.

The command to update a feed is as below:

```
mosquitto_pub -h mqtt.thingspeak.com -p 1883 -u dragino -P QZXTxxxxxO2J -i
dragino_Client -t channels/200893/publish/B9Z0R25QNVEBKIFY -m
"field1=34&field2=89&status=MQTTPUBLISH"
```

(Make sure the "" is included, otherwise you only one data is upload)

Below is the output window:



```

172.31.255.254 - SecureCRT
文件(F) 编辑(E) 查看(V) 选项(O) 传输(T) 脚本(S) 工具(L) 帮助(H)
172.31.255.254
root@dragino-146d78:~# mosquitto_pub -h mqtt.thingspeak.com -p 1883 -u dragino -P Q
-2j -i dragino_client -t channels/200893/publish/B9. 'FY -m "
field1=34&field2=89&status=MQTTPUBLISH"
root@dragino-146d78:~#

```

After running this command, we can see the data are updated to ThingSpeak, which has same result as what we did at mqtt.fx

So we success to use LG01N to uplink data to ThingSpeak, the **mosquitto_pub** command is executed in the Linux side, finally, we will have to call **mosquitto_pub** command while the LoRa sensor data arrive. We will explain how to do that in next step.

6.2.3 Test LG01N MQTT routine service.

Above process help us to understand how MQTT works in LG01N, now we can move further, try to use the built-in MQTT routine.

Step1: Select MQTT Transfer mode

dragino-1b7060 Status ▾ System ▾ Network ▾ Service ▾ Logout

LoRa Gateway Settings

Configuration to communicate with LoRa devices and LoRaWAN server

LoRaWAN Server Settings

IoT Service LoRaRAW forward to MQTT ser ▾

Debug Level Little message output ▾

Service Provider The Things Network ▾

Step2: Configure MQTT parameter.

MQTT Server Settings

Configuration to communicate with MQTT server

Configure MQTT Server

Select Server ThingSpeak MQTT ▾

User Name [-u] dragino1

Password [-P] 32W6GMEXYTEQ7049

Client ID [-i] dragino_Client

MQTT Channel

Match between Local Channel and remote channel

Local Channel in /var/iot/channels/	Remote Channel in IoT Server	Write API Key	
10009	396640	P07KVY59P5QEY6M6	Edit Delete

Add

Save & Apply
Save
Reset

Step3: Simulate channel data

The MQTT process will keep checking if there is data in the directory: `/var/iot/channels`. If there is new data, the process will check if this data match the local channel ID. If match, the process will update it to MQTT server.

For example:

In step 2, we have below settings:

- ✓ UserName[-u option]: dragino1
- ✓ Password[-P option]: 32W6GMEXYTEQ7049
- ✓ Client_ID[-i]: dragino_Client
- ✓ Because we choose Thingspeak so we have below pre-set options but not show in web
 - Broker Address[-h]: mqtt.thingspeak.com
 - Broker Port[-p]: 1883
 - Topic Format[-t]: channels/CHANNEL/publish/WRITE_API
 - Data String Format[-m]: DATA&status=MQTTPUBLISH

And we configure this channel:

- ✓ Local Channel ID: 10009
- ✓ Remote Channel ID: 396640
- ✓ Write_api_key: P07KVY59P5QEY6M6

All above info are used to compose the `mosquitto_pub` which use to upload data to MQTT server.

To test; run this command:

```
echo "field1=34&field2=89" > /var/iot/channels/10009
```

This command will create a file name 10009 with the content field1=34&field2=89.

The MQTT process will detect this file and found it match the local channel setting. Then composed the `mosquitto_pub`, most option is easy understood, except the `-t` and `-m` option.

The `-t` and `-m` option use a format to fit different server.

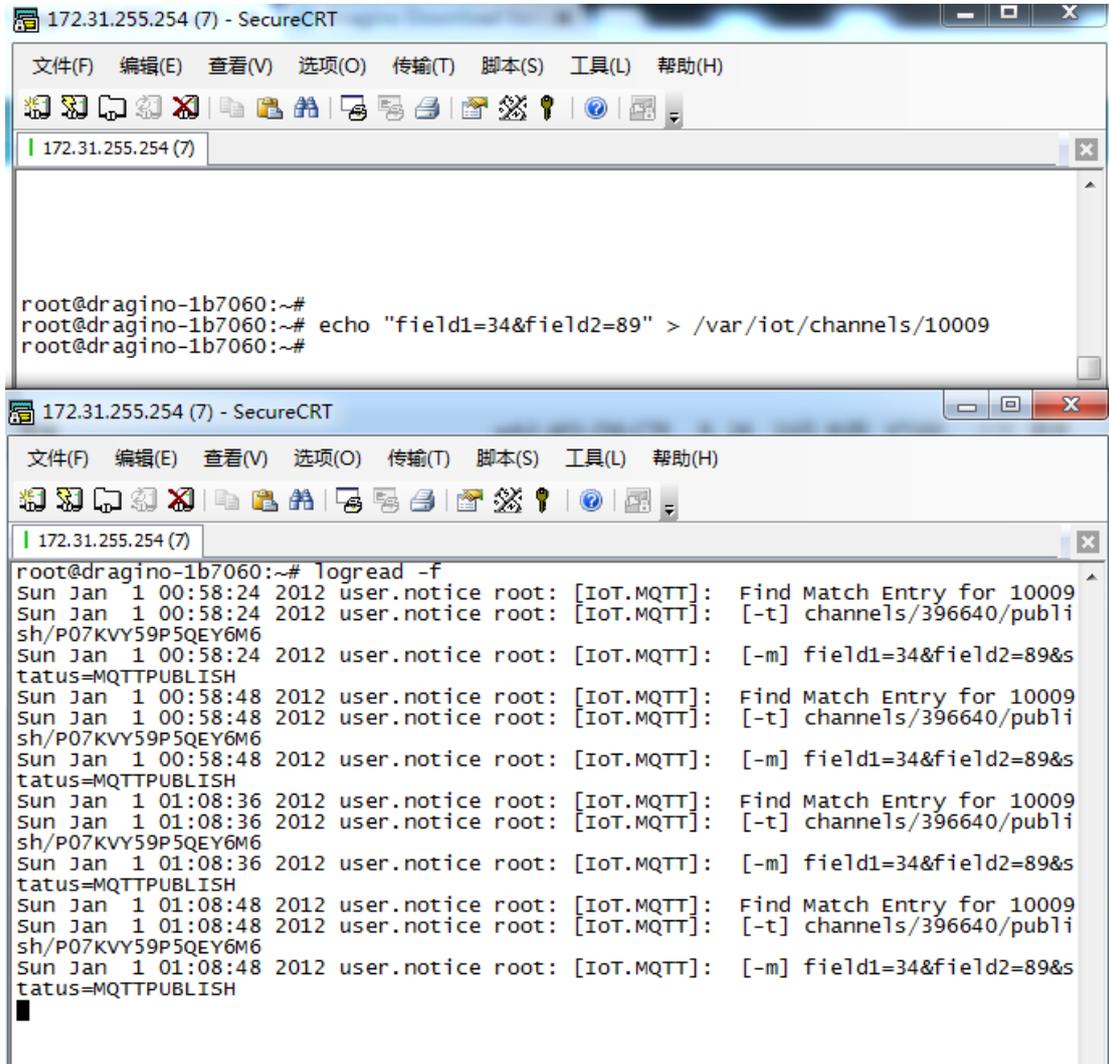
The convert method is:

- ✓ [-t] Format: channels/CHANNEL/publish/WRITE_API
 - CHANNEL replace with Remote Channel ID: 396640
 - WRITE_API replace with Write_api_key: P07KVY59P5QEY6M6
 - So [-t] is: channels/396640/ publish/ P07KVY59P5QEY6M6
- ✓ [-m] format: DATA&status=MQTTPUBLISH
 - DATA replace with the content "field1=34&field2=89"
 - So [-m] is : field1=34&field2=89&status=MQTTPUBLISH

So the LG01N will send out this command:

```
mosquitto_pub -h mqtt.thingspeak.com -p 1883 -u dragino1 -P 32W6GMEXYTEQ7049 -i
dragino_Client -t channels/396640/ publish/ P07KVY59P5QEY6M6 -m
"field1=34&field2=89&status=MQTTPUBLISH"
```

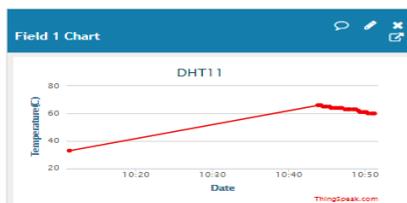
The output in logread -f is:



And see update:

Channel Stats

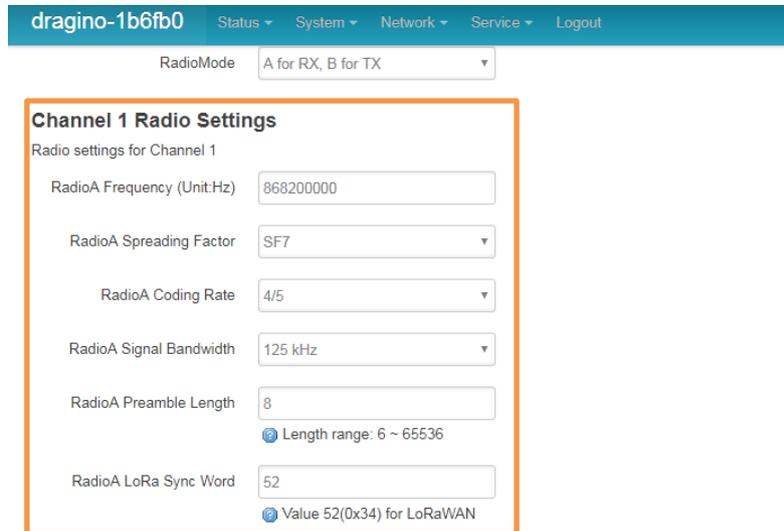
Created: [about an hour ago](#)
 Updated: [less than a minute ago](#)
 Last entry: [less than a minute ago](#)
 Entries: 22



6.2.4 Configure LoRa End node

The target for us now is how to get the remote sensor data into `/var/iot/channels/`.

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



dragino-1b6fb0 Status System Network Service Logout

RadioMode A for RX, B for TX

Channel 1 Radio Settings

Radio settings for Channel 1

RadioA Frequency (Unit:Hz) 868200000

RadioA Spreading Factor SF7

RadioA Coding Rate 4/5

RadioA Signal Bandwidth 125 kHz

RadioA Preamble Length 8
Length range: 6 ~ 65536

RadioA LoRa Sync Word 52
Value 52(0x34) for LoRaWAN

Now the LG01N will listen on this LoRa channel. If the received data match the pre-define data format, LG01N will store it in `/var/iot/channels/` and the MQTT process can handle it for upload.

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 local channels 12345 and 34567.

And there 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

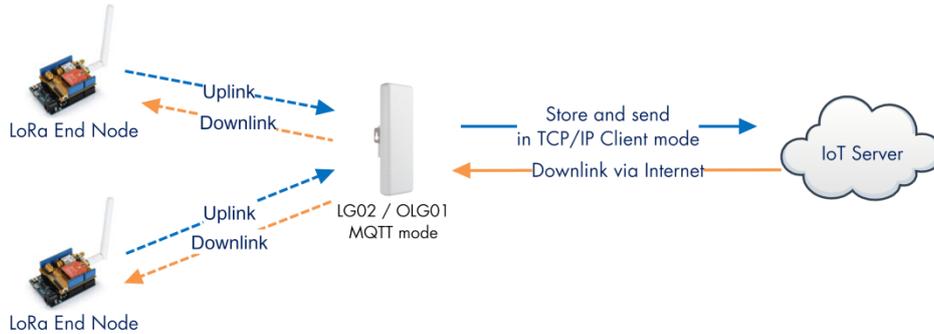
LoRa End Device reference source code: [check this link](#).

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.

TCP/IP Client mode:

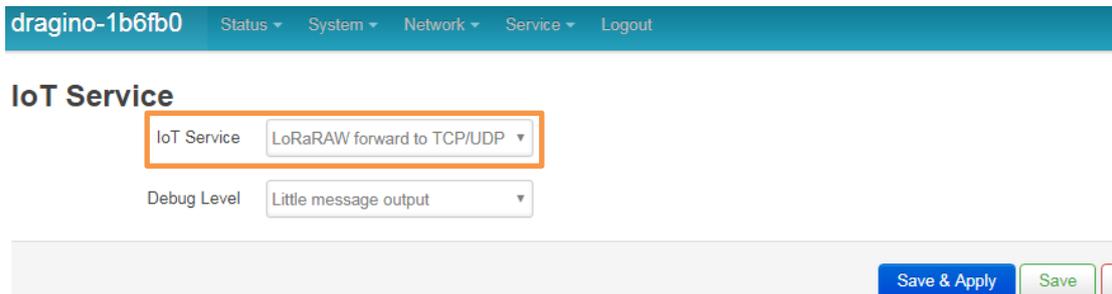
Use LG02 / OLG02 as a LoRa Gateway to forward packet to IoT Server in TCP/IP Client Mode



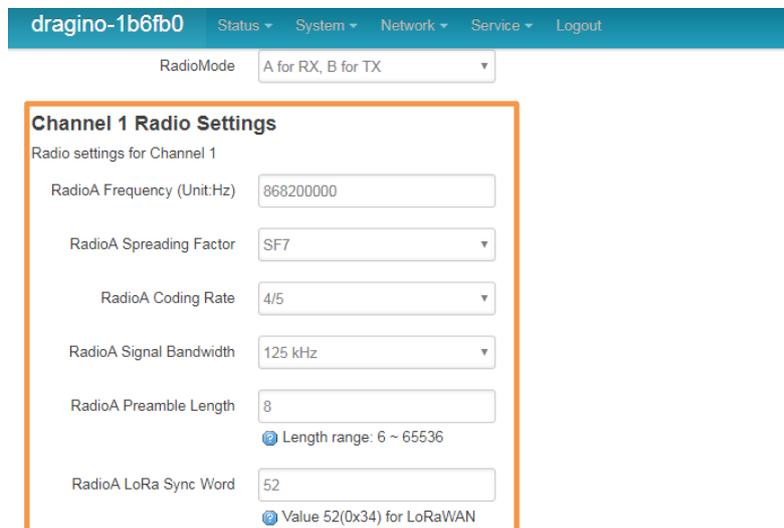
Operate Principle:

- > The LoRa end node sends data to LG02 gateway via private LoRa protocol. LG02 stores the sensor data.
- > LG02 sends the sensor data to IoT Server via general TCP/IP Client mode.

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

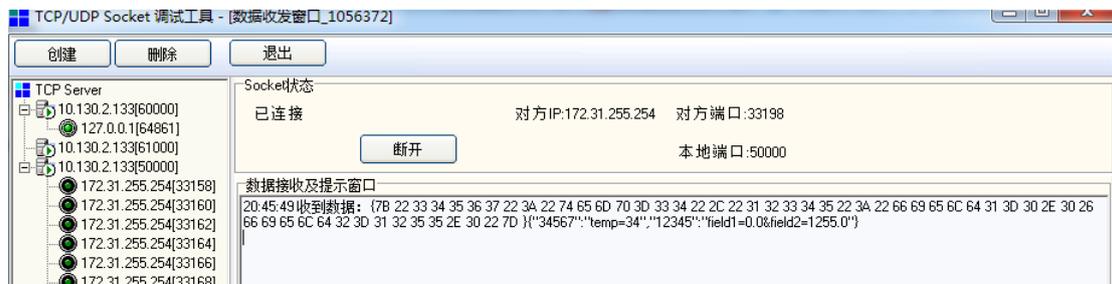
The TCP/IP server will get {"12345":"field1=0.0&field2=1102.0"}

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"}



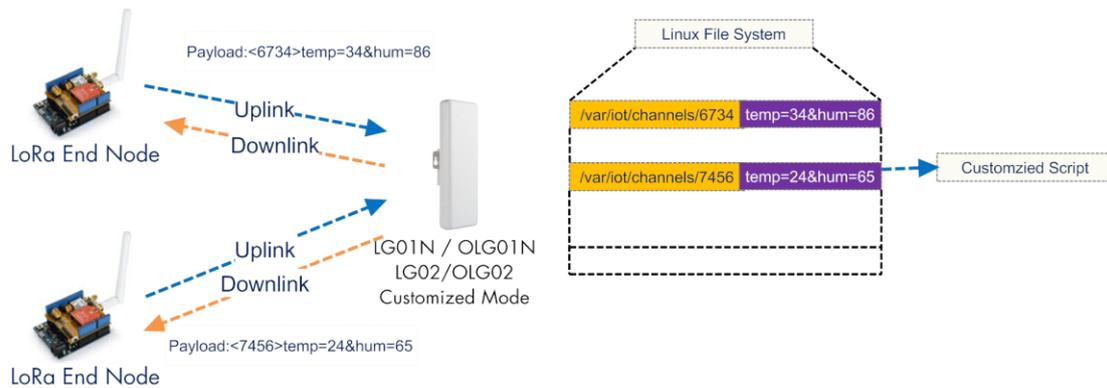
LoRa End Device reference source code: [check this link](#).

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:

How customized script works:

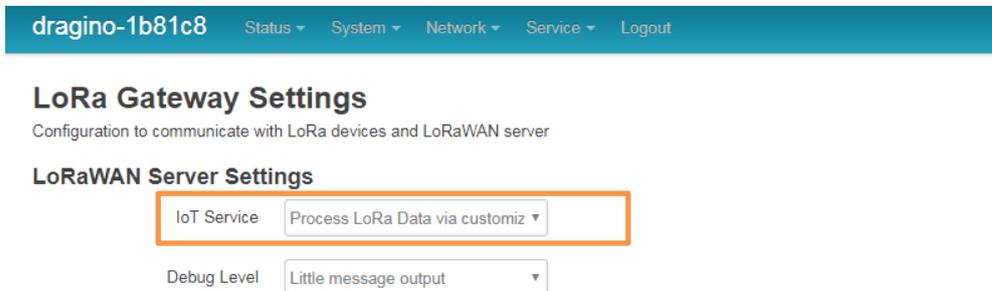


Operate Principle:

- > 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.
- > The customized script interact with these channels files. So developer can focus on writing this script.

Example: Store Data in a file.

Step 1: Choose LoRa customized script mode



Step 2: Configure LoRa Frequency

Channel 1 Radio Settings

Radio settings for Channel 1

RadioA Frequency (Unit:Hz)

RadioA Spreading Factor

RadioA Coding Rate

Step 3: Choose the customized script

Customized Script

Run a Customized Script to process LoRa Data, parameters are optional and defined in script

General Settings

Script Name

Parameter 1

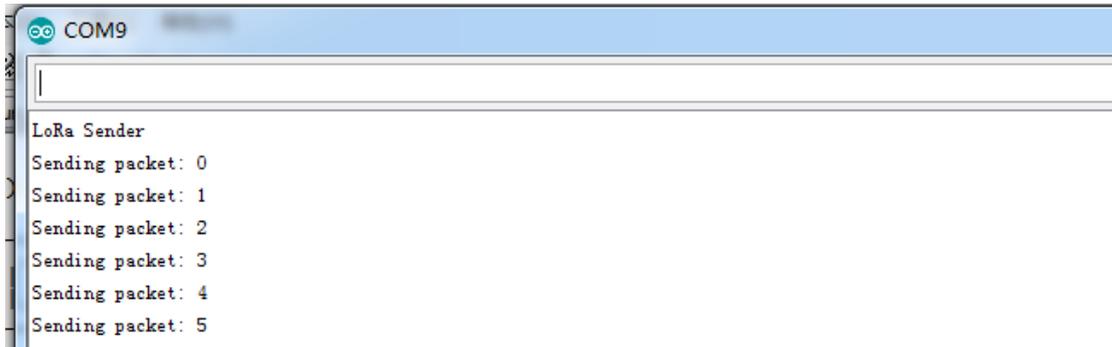
The directory to store customized script is in `/etc/lora/customized_scripts/`. User can write a new script and put it under this directory for their application. The web will auto detect it.

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

```

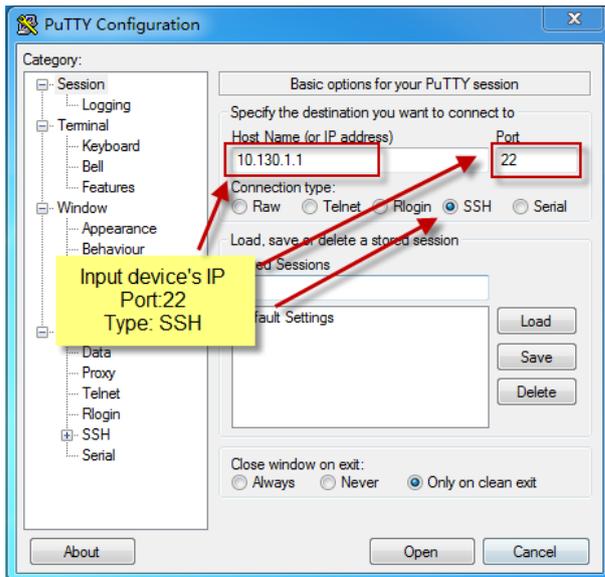
root@dragino-1b81c8:~# cat /var/sensor_data
Sun Jan 1 00:15:26 UTC 2012 :<1234> 123443
Sun Jan 1 00:46:26 UTC 2012 :<10009> field1=32&field2=94
Sun Jan 1 00:46:44 UTC 2012 :<10009> field1=32&field2=94
Sun Jan 1 00:46:56 UTC 2012 :<10009> field1=28&field2=93
Sun Jan 1 00:47:08 UTC 2012 :<10009> field1=25&field2=87
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. 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.

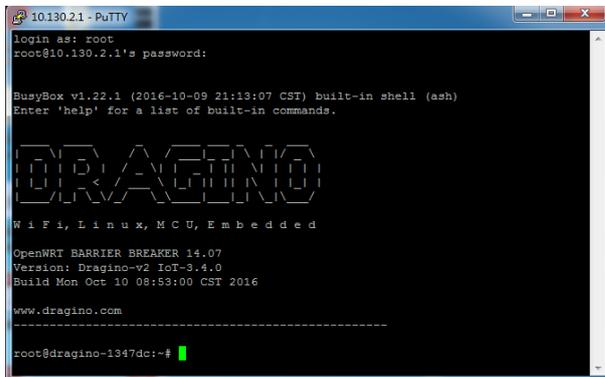
9.1 SSH Access for Linux console

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:



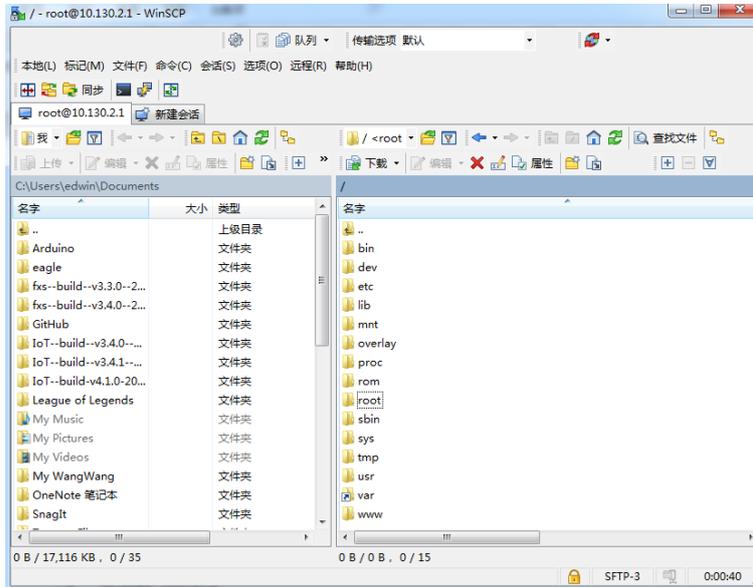
IP address: IP address of LG01N
Port: 22
User Name: **root**
Password: **dragino** (default)

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



9.2 Edit and Transfer files

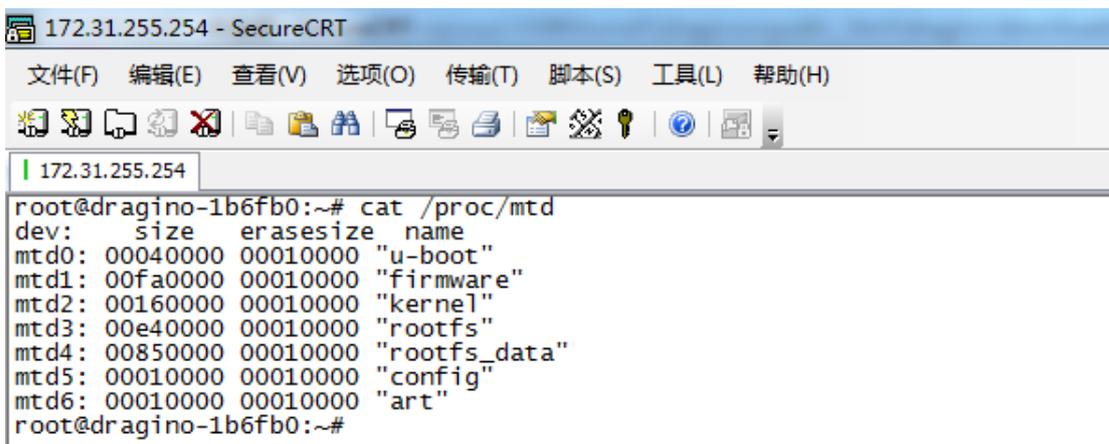
The LG01N support **SCP protocol** and has a built **SFTP server**. There are many ways to edit and transfer files using these two protocols. One of the easiest is through [WinSCP](#) utility. After access via WinSCP to the device, use can use a FTP alike window to drag / drop files to the LG01N or Edit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



9.3 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.

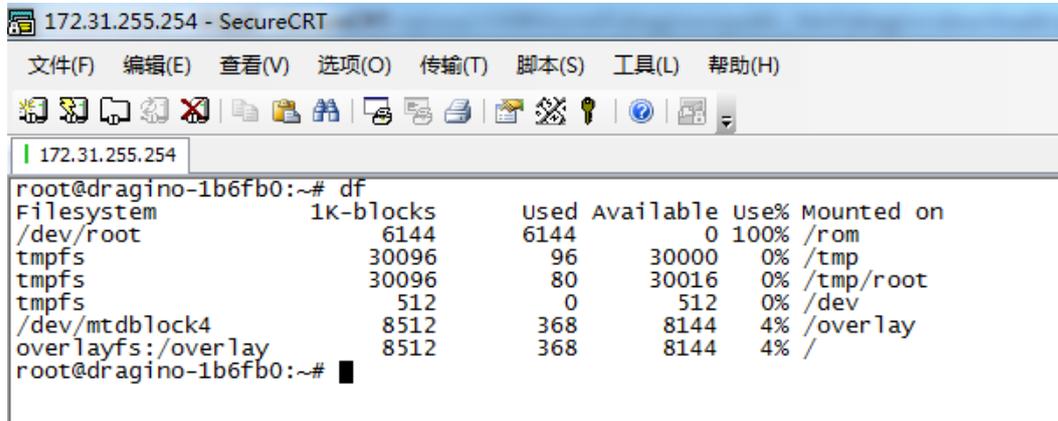
Use cat /proc/mtd to see all blocks/partitions.



- ✓ "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:



```

172.31.255.254 - SecureCRT
文件(F) 编辑(E) 查看(V) 选项(O) 传输(T) 脚本(S) 工具(L) 帮助(H)
172.31.255.254
root@dragino-1b6fb0:~# df
Filesystem          1K-blocks      Used Available  Use% Mounted on
/dev/root            6144          6144         0 100% /rom
tmpfs                30096           96    30000     0% /tmp
tmpfs                30096           80    30016     0% /tmp/root
tmpfs                 512             0         512     0% /dev
/dev/mtdblock4       8512          368     8144     4% /overlay
overlayfs:/overlay  8512          368     8144     4% /
root@dragino-1b6fb0:~#

```

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:

mtid erase rootfs_data -r

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

9.4 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.
```

10. Upgrade Linux Firmware

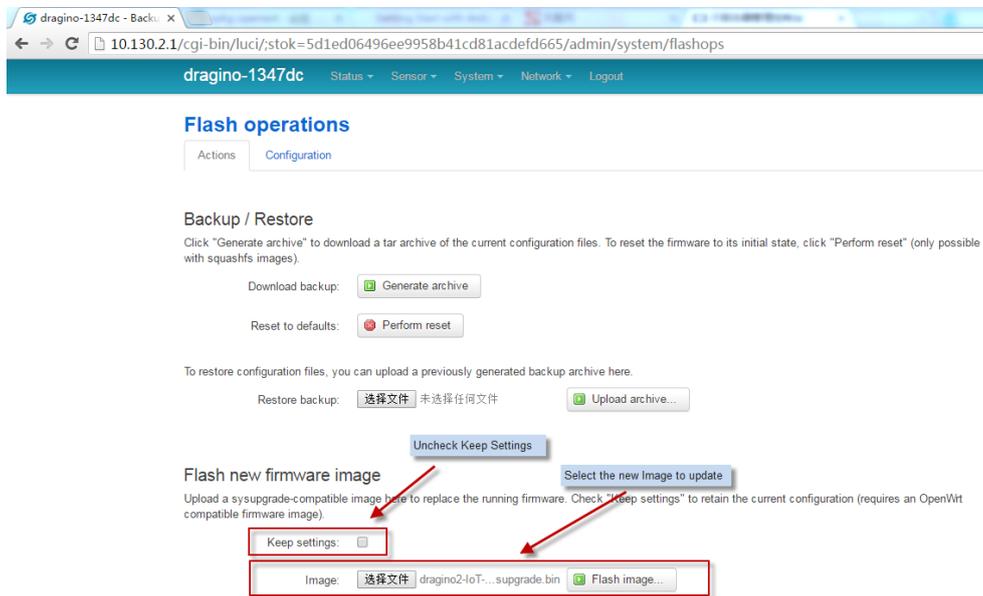
We keep improving the LG01N Linux side firmware for new features, bug fixes. The latest firmware can be found on [LG01N Firmware & release note](#)

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

10.1 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.



10.2 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.

11. FAQ

11.1 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.

11.2 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.

11.3 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 hopping 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.

11.4 Can I develop my own 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

11.5 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

11.6 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`

```

-----
Wed Sep 12 01:39:19 2018 daemon.info lg02_pkt_fwd[14341]: INFO (json): [down] [{"txpk":{"fchme":false,"tmst":2831770149,"freq":927.5,"rfch":0,"pwr":20,"modu":"LORA","da
tr":"SF7Bw500","codr":"4/5","ipol":true,"size":17,"ncrc":true,"data":"IIdG+uy4yL7RAFxSHIX0A="}}]
Wed Sep 12 01:39:19 2018 daemon.info lg02_pkt_fwd[14341]: SF=0x07
Wed Sep 12 01:39:19 2018 daemon.info lg02_pkt_fwd[14341]: Transmit at SF7Bw500 on 927.500000.
Wed Sep 12 01:39:20 2018 daemon.info lg02_pkt_fwd[14341]: SF=0x07
Wed Sep 12 01:39:20 2018 daemon.info lg02_pkt_fwd[14341]: Transmit at SF7Bw500 on 927.500000.
Wed Sep 12 01:39:20 2018 daemon.info lg02_pkt_fwd[14341]: Downlink done: count_us=2831770149
Wed Sep 12 01:39:21 2018 daemon.info lg02_pkt_fwd[14341]: INFO (json): [down] [{"txpk":{"fchme":false,"tmst":2833763738,"freq":927.5,"rfch":0,"pwr":20,"modu":"LORA","da
tr":"SF7Bw500","codr":"4/5","ipol":true,"size":17,"ncrc":true,"data":"IGNTMK9p5y1jF9BP1xbZvI="}}]
Wed Sep 12 01:39:21 2018 daemon.info lg02_pkt_fwd[14341]: SF=0x07
Wed Sep 12 01:39:21 2018 daemon.info lg02_pkt_fwd[14341]: Transmit at SF7Bw500 on 927.500000.
Wed Sep 12 01:39:22 2018 daemon.info lg02_pkt_fwd[14341]: SF=0x07
Wed Sep 12 01:39:22 2018 daemon.info lg02_pkt_fwd[14341]: Transmit at SF7Bw500 on 927.500000.
Wed Sep 12 01:39:22 2018 daemon.info lg02_pkt_fwd[14341]: Downlink done: count_us=2833763738
Wed Sep 12 01:39:22 2018 daemon.info lg02_pkt_fwd[14341]: Receive(HEX):40ad2a012680000010a2fd88ae57fa9451d478e5a1e693d8b
-----

```

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

```

enum {
  US915_125kHz_UPFBASE = 914200000,
  US915_125kHz_UPFSTEP = 0,
  US915_500kHz_UPFBASE = 902320000,
  US915_500kHz_UPFSTEP = 0,
  US915_500kHz_DNFBASE = 927500000, //receive
  US915_500kHz_DNFSTEP = 0
};

```

For the result:

Time	Packet ID	Length	Payload
10:06:25	116	1	payload: 68 65 6C 6C 6F 20 77 6F 72 6C 64 21
10:06:11	115	1	payload: 68 65 6C 6C 6F 20 77 6F 72 6C 64 21
10:05:57	114	1	payload: 68 65 6C 6C 6F 20 77 6F 72 6C 64 21
10:05:43	113	1	payload: 68 65 6C 6C 6F 20 77 6F 72 6C 64 21
10:05:29	112	1	payload: 68 65 6C 6C 6F 20 77 6F 72 6C 64 21

Ex2: We use 903.0Mhz frequency

We can input the command: `logread -f`

```

root@dragino-19a944:~# logread -f
wed Sep 12 02:11:31 2018 daemon.info lg02_pkt_fwd[20677]:
wed Sep 12 02:11:31 2018 daemon.info lg02_pkt_fwd[20677]: INFO (json): [down] [{"txpk":{"imme":false,"tmst":468442152,"freq":923.3,"rfch":0,"pove":20,"modu":"LORA","dat
r":{"SF7Bw500","codr":"4/5","tpol":true,"size":17,"hcrcl":true,"data":"lgkY0uey3XLqHTFSovbRBg="}}]
wed Sep 12 02:11:31 2018 daemon.info lg02_pkt_fwd[20677]: SF=0x07
wed Sep 12 02:11:31 2018 daemon.info lg02_pkt_fwd[20677]:
wed Sep 12 02:11:32 2018 daemon.info lg02_pkt_fwd[20677]: Transmit at SF7Bw500 on 923.300000.
wed Sep 12 02:11:32 2018 daemon.info lg02_pkt_fwd[20677]: SF=0x07
wed Sep 12 02:11:32 2018 daemon.info lg02_pkt_fwd[20677]:
wed Sep 12 02:11:32 2018 daemon.info lg02_pkt_fwd[20677]: Transmit at SF7Bw500 on 923.300000.
wed Sep 12 02:11:32 2018 daemon.info lg02_pkt_fwd[20677]: Downlink done: count_us=468442152
wed Sep 12 02:11:36 2018 daemon.info lg02_pkt_fwd[20677]:
wed Sep 12 02:11:36 2018 daemon.info lg02_pkt_fwd[20677]: Receive(HEX):00ac2301d07ed5b370907cb65d67c64a00cd3586bb5c88
wed Sep 12 02:11:36 2018 daemon.info lg02_pkt_fwd[20677]:
wed Sep 12 02:11:36 2018 daemon.info lg02_pkt_fwd[20677]: INFO (JSON): [up] [{"txpk":{"time":"2018-09-12T02:11:36.210520Z","tmst":472538265,"chan":0,"rfch":1,"freq":90
3.000000,"stat":1,"modu":"LORA","datr":{"SF7Bw125","codr":"4/5","tsnr":7.8,"rssi":-34,"size":23,"data":"AKWjADb+1BNwKH2XwFGsqDNNya7X1q="}}]

```

▲ 10:13:33	1	1		payload: 68 65 6C 6C 6F 20 77 6F 72 6C 64 21
▼ 10:13:21		0		
▲ 10:13:20	0	1	retry	payload: 68 65 6C 6C 6F 20 77 6F 72 6C 64 21
+	10:13:15			dev addr: 26 01 20 71 app eui: 70 B3D5 7E D001 23AC dev eui: 00 4AC6 67 5D B67C 90

If join the network successfully, it will send a reply.

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:

▲ 10:16:57	16	1		payload: 68 65 6C 6C 6F 20 77 6F 72 6C 64 21
▲ 10:16:43	15	1		payload: 68 65 6C 6C 6F 20 77 6F 72 6C 64 21
▲ 10:16:29	14	1		payload: 68 65 6C 6C 6F 20 77 6F 72 6C 64 21
▲ 10:16:15	13	1		payload: 68 65 6C 6C 6F 20 77 6F 72 6C 64 21
▲ 10:16:01	12	1		payload: 68 65 6C 6C 6F 20 77 6F 72 6C 64 21
▲ 10:15:47	11	1		payload: 68 65 6C 6C 6F 20 77 6F 72 6C 64 21

11.7 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

11.8 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

11.9 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. Trouble Shooting

12.1 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
```

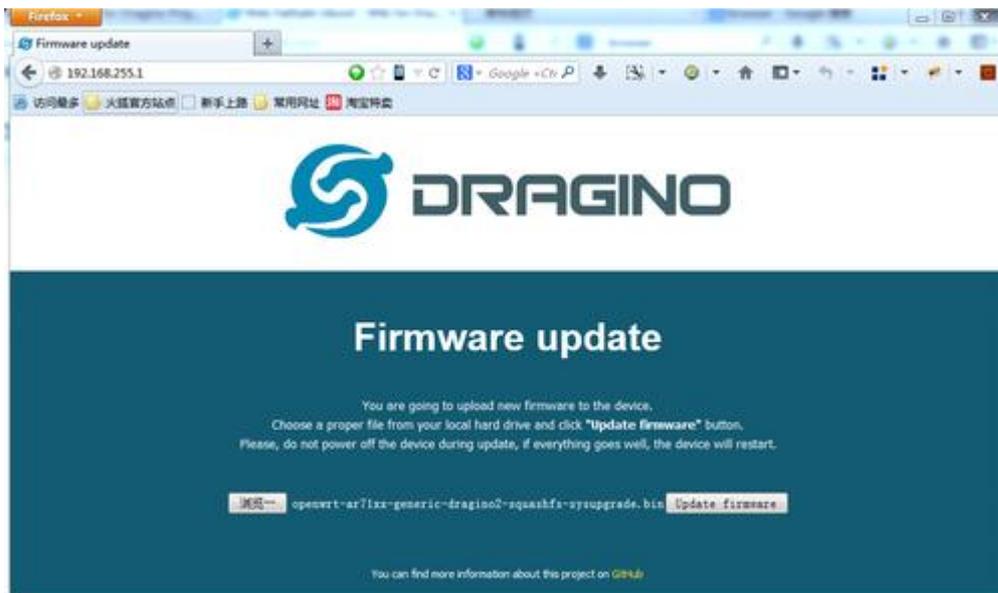
12.2 How to recover the LG01N if firmware crash

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 browser
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

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

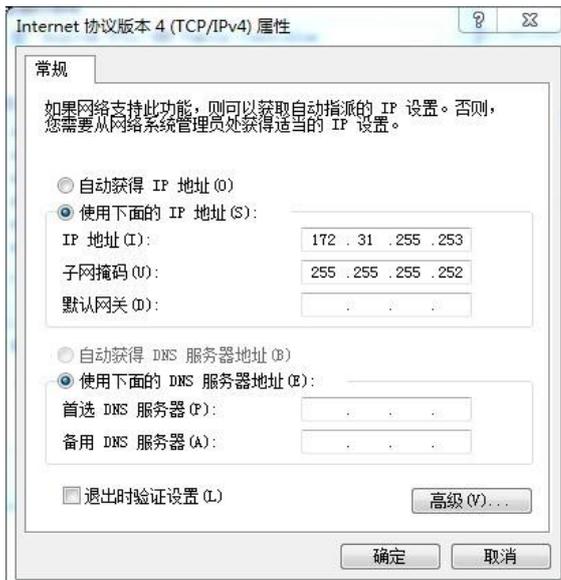
The LG01 has a fall-back ip in its LAN port. This IP is always enabled so user can use fall back ip to access LG01N no matter what the WiFi IP is. The fall back ip is useful for connect and debug the unit.

(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.

13. 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](#).

14. 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

15. 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

16. 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>