



LG01N/OLG01N LoRa Gateway User Manual

Document Version: 1.4.0

Firmware Version: lgw--build-v5.4.1607519907-20201209-2120

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
1.3.0	-- Add more features, remote access	2020-Mar-02
1.3.1	-- Add contents for access to the device	2020-Mar-16
1.3.2	-- Add TTN Server notice	
1.4.0	-- Change to New UI manual	2020-Dec-23

1. Introduction	5
1.1 What is LG01N & OLG01N	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 Interfaces	10
1.8 Install SIM card in 4G module	11
1.9 Firmware Change log	12
2. Access and configure LG01N	13
2.1 Find IP address of LG01N	13
2.1.1 Connect via WiFi	13
2.1.2 Connect via WAN port with DHCP IP from router	13
2.1.3 Connect via LAN port with direct connection from PC	14
2.1.4 Connect via WiFi with DHCP IP from router	14
2.1.5 Connect via LAN port by fall back ip	14
2.2 Access Configure Web UI	15
3. Typical Network Setup	16
3.1 Overview	16
3.2 Use WAN port to access Internet	16
3.3 Access Internet as a WiFi Client	17
3.4 Use built-in 4G modem for internet access	18
3.5 Check Internet connection	19
4. Example 1: Manually send / receive LoRa packets	20
4.1 Use <code>pkt_fwd</code> to receive	20
4.2 Use <code>pkt_fwd</code> to transmit	20
5. Example 2: MQTT Transfer Mode	22
6. Example 3: TCP IP Client Mode	23
7. Example 4: Write a customized script	23

8.	Example 5: Communicate to a HTTP server	26
8.1	Test uplink and downlink via Linux command	26
8.2	Test uplink and downlink in LoRa	28
8.2.1	Set up on gateway.....	28
9.	Example 6: Configure as a LoRaWAN gateway – Limited LoRaWAN mode.....	29
10.	More features	30
10.1	Remote Access	30
10.2	More instructions	30
11.	Linux System	30
11.1	SSH Access for Linux console.....	30
11.2	Edit and Transfer files	32
11.3	File System	32
11.4	Package maintain system	34
12.	Upgrade Linux Firmware	35
12.1	Upgrade via Web UI.....	35
12.2	Upgrade via Linux console.....	35
13.	FAQ	36
13.1	Why there is 433/868/915 version LoRa part?.....	36
13.2	What is the frequency range of LG01N LoRa part?	36
13.3	What does “Limited support on LoRaWAN”?	36
13.4	Can I develop my own LoRa protocol and other software for LG01N?	37
13.5	Can I make my own firmware for LG01N? Where can I find the source code of LG01N?	37
13.6	On OTAA mode, if I use the other frequency, how should I modify in the library?.....	37
13.7	How can I reset the device to factory default?	38
13.8	Can I control the LEDs?.....	39
14.	Trouble Shooting	39
14.1	I get kernel error when install new package, how to fix?	39
14.2	How to recover the LG01N if firmware crash.....	40
14.3	I configured LG01N for WiFi access and lost its IP. What to do now?.....	41

15.	Order Info.....	42
16.	Packing Info.....	42
17.	Support	42
18.	Reference	43

1. Introduction

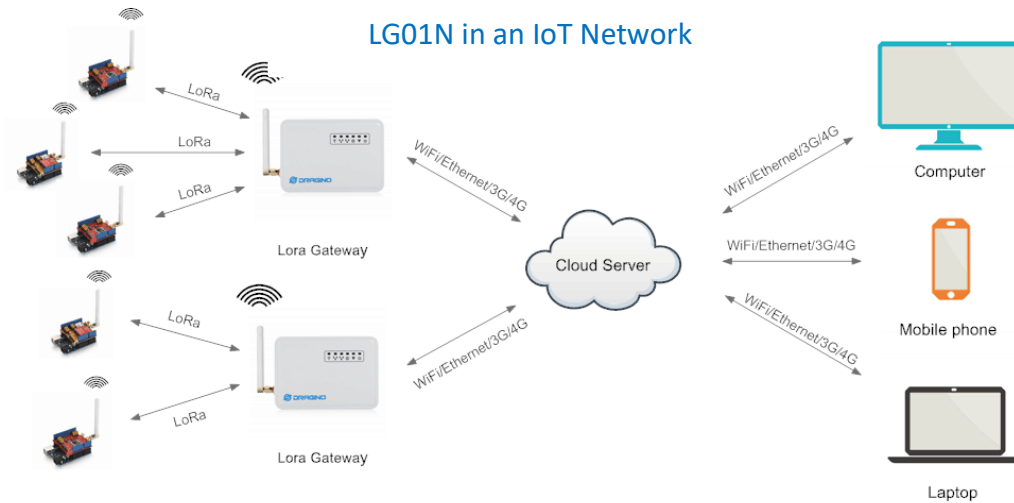
1.1 What is LG01N & OLG01N

LG01N & OLG01N are an open-source **single channel LoRa Gateway**. It lets you bridge LoRa wireless network to an IP network via WiFi, Ethernet or optional 3G/4G cellular network. 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 optional 3G/4G Cellular**. These interfaces provide flexible methods for users to connect their sensor networks to Internet.

LG01N can be used to provide a low cost IoT wireless solution to support 10~100 sensor nodes.

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



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):

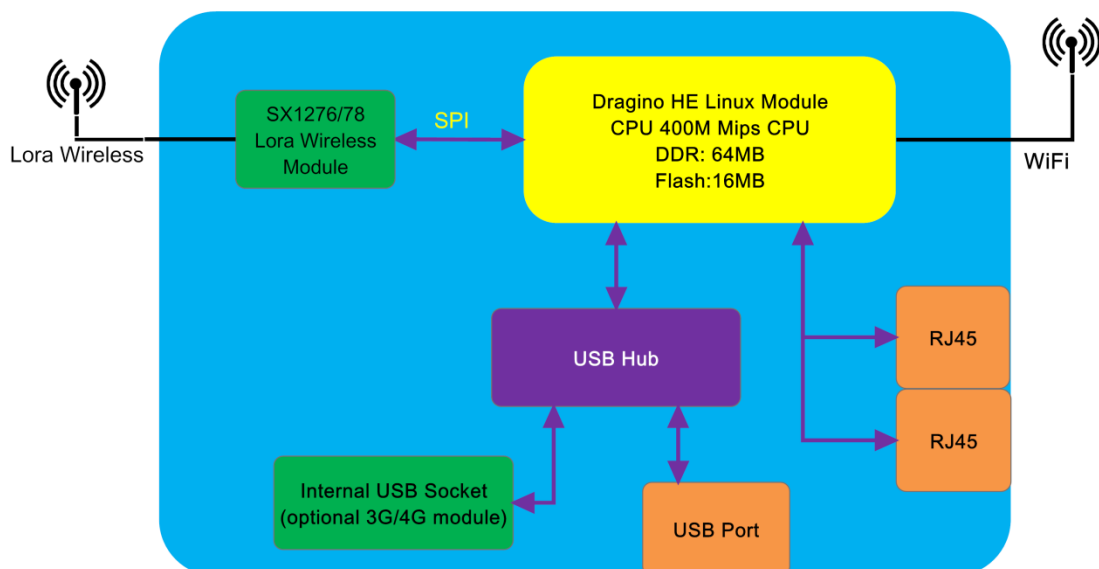
- Quectel [EC25 LTE module](#)
- Micro SIM Slot
- Internal 4G Antenna + External 4G Sticker Antenna.
- Up to 150Mbps downlink and 50Mbps uplink data rates
- Worldwide LTE, UMTS/HSPA+ and GSM/GPRS/EDGE coverage
- MIMO technology meets demands for data rate and link reliability in modem wireless communication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



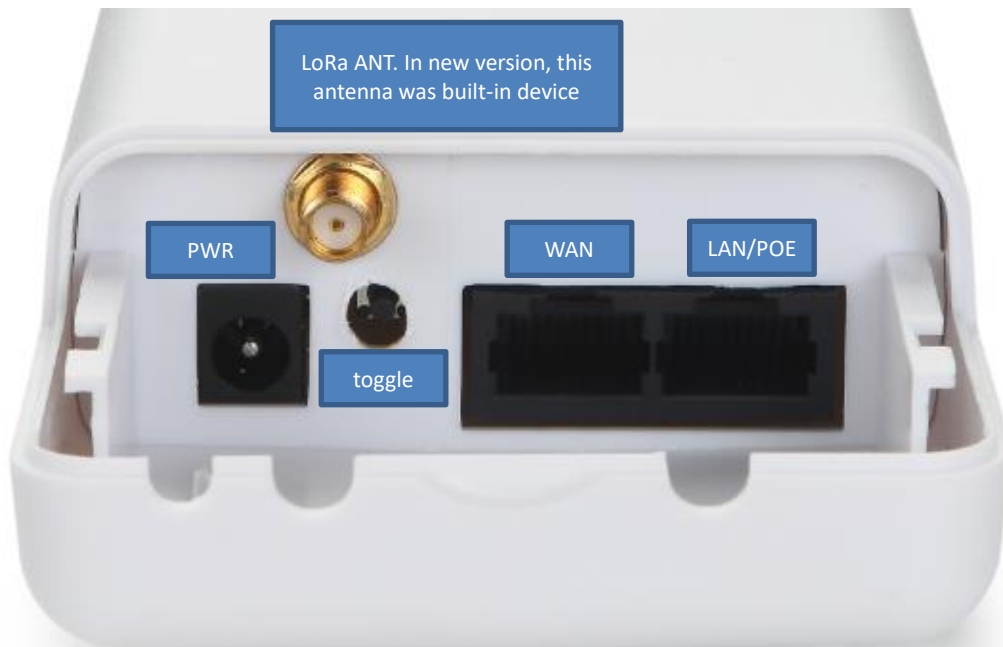
1.6 Hardware Variants

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

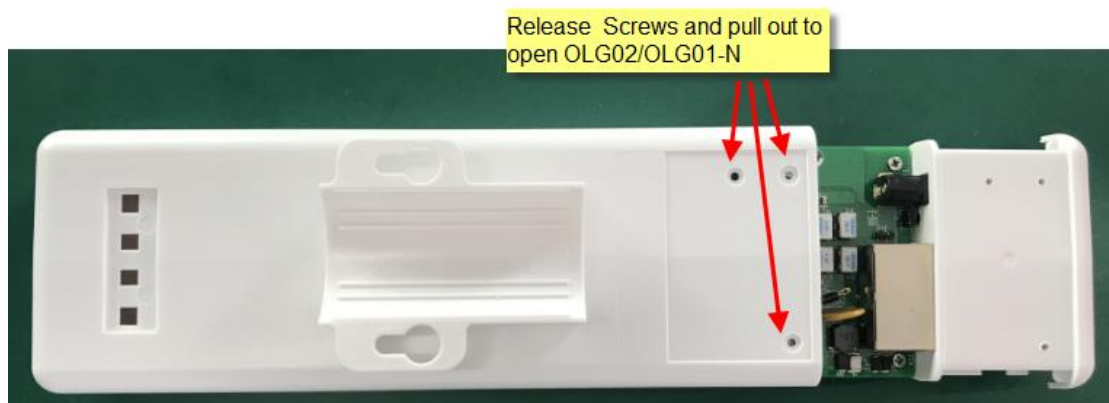
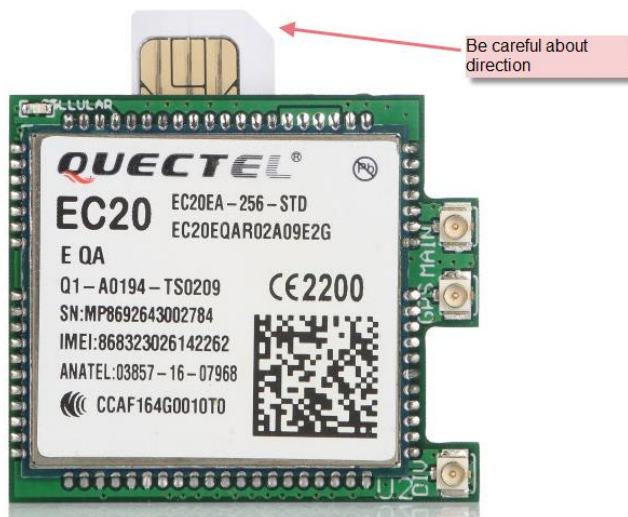
OLG01N Version Interface:



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 unassembled, please open the box and use below direction to install the SIM card (Micro SIM)

Note: Please power off when install the SIM card, and power on again after installation.



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 and configure LG01N

The LG01N is configured as a WiFi Access Point by default. User can access and configure the LG01N after connecting to its WiFi network, or via its Ethernet ports.

2.1 Find IP address of LG01N

2.1.1 Connect via WiFi



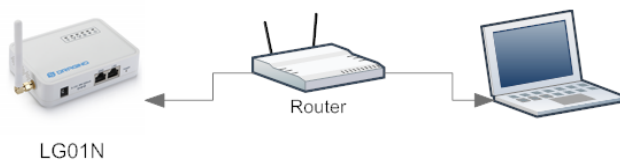
At the first boot of LG01N, it will auto generate an unsecure WiFi network call **dragino-xxxxxx**

Note: In latest version firmware, it has been password protected and the password is:
dragino+dragino

User can use the laptop to connect to this WiFi network. The laptop will get an IP address 10.130.1.xxx and the LG01N has the default IP **10.130.1.1**



2.1.2 Connect via WAN port with DHCP IP from router



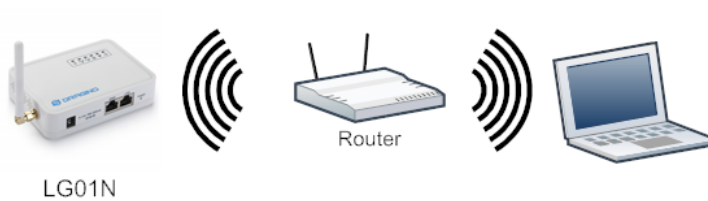
Alternatively, connect the LG01N **WAN port** to your router and LG01N will obtain an IP address from your router. In the router's management portal, you should be able to find what IP address the router has assigned to the LG01N. You can also use this IP to connect.

2.1.3 Connect via LAN port with direct connection from PC



The LG01N **LAN port** is configured as DHCP router by default, user can connect the PC to LAN port and set PC to DHCP mode, the PC will get IP from LG01N's LAN port and be able to access to the device. The default IP in LG01N LAN port is 10.130.1.1

2.1.4 Connect via WiFi with DHCP IP from router



If the LG01N already connect to the router via WiFi, use can use the WiFi IP to connect to LG01N.

2.1.5 Connect via LAN port by fall back ip

The **LAN port** also has a [fall back ip address](#) for access if user doesn't connect to uplink router.

2.2 Access Configure Web UI

Web Interface

Open a browser on the PC and type the LG01N ip address (depends on your connect method)

<http://10.130.1.1/> (Access via WiFi AP network)

or

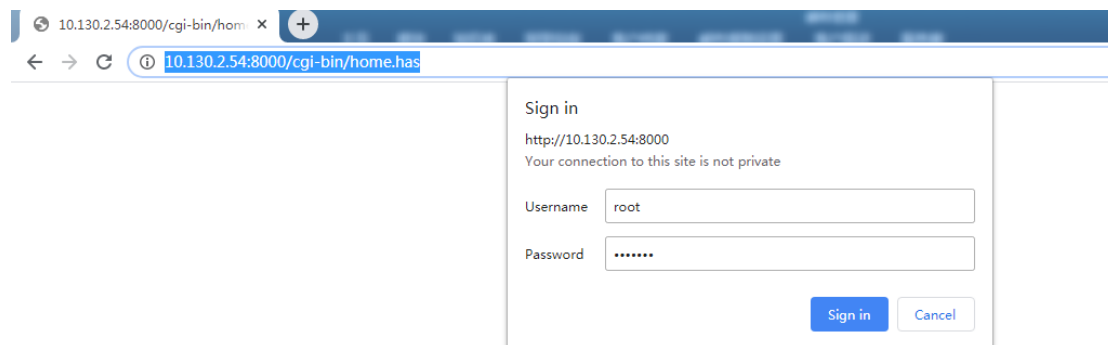
http://IP_ADDRESS or http://IP_ADDRESS:8000 (If the IP is assigned by uplink router)

You will see the login interface of LG01N as shown below.

The account details for Web Login are:

User Name: root

Password: dragino



3. Typical Network Setup

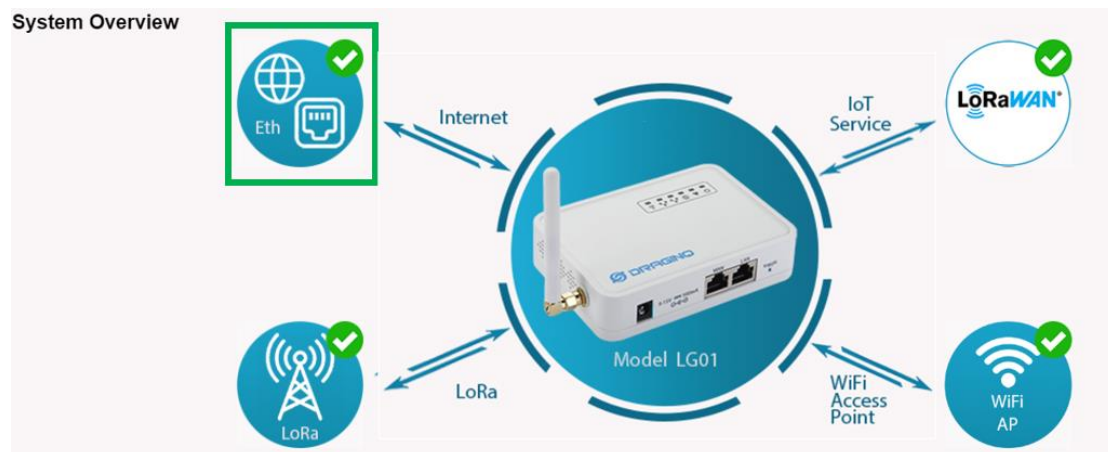
3.1 Overview

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

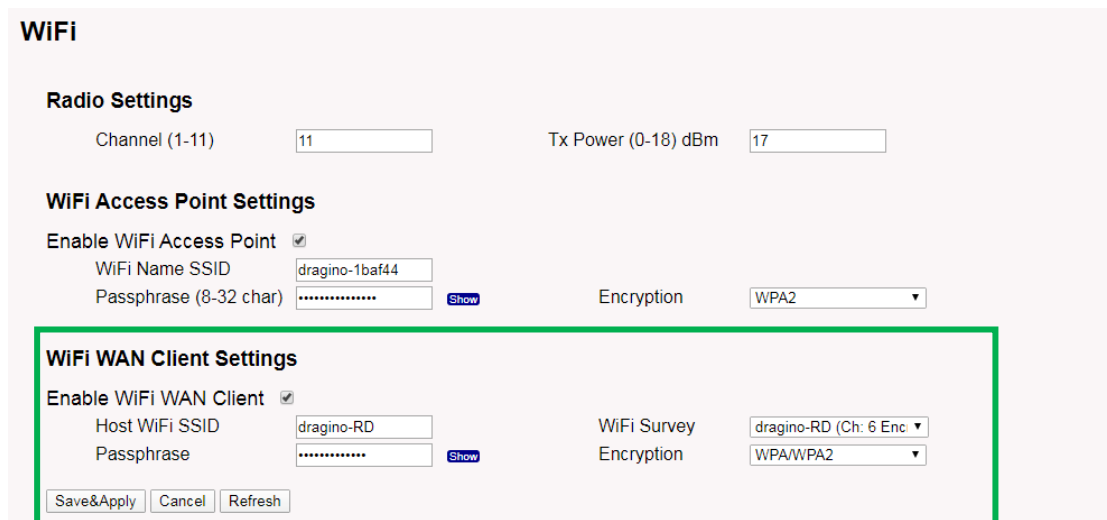
By default, LG01N is set to use the WAN port to connect to an upstream network. When you connect the LG01N's WAN port to an upstream router, LG01N will get an IP address from the router and have Internet access via the upstream router. The network status can be checked in the [home page](#):



3.3 Access Internet as a WiFi Client.

In the WiFi Client Mode, LG01N acts as a WiFi client and gets DHCP from an upstream router via WiFi.

The settings for WiFi Client are under page [System](#) → [WiFi](#) → [WiFi WAN Client Settings](#)



WiFi

Radio Settings

Channel (1-11) Tx Power (0-18) dBm

WiFi Access Point Settings

Enable WiFi Access Point

WiFi Name SSID

Passphrase (8-32 char) [Show](#) Encryption

WiFi WAN Client Settings

Enable WiFi WAN Client

Host WiFi SSID

Passphrase [Show](#) WiFi Survey

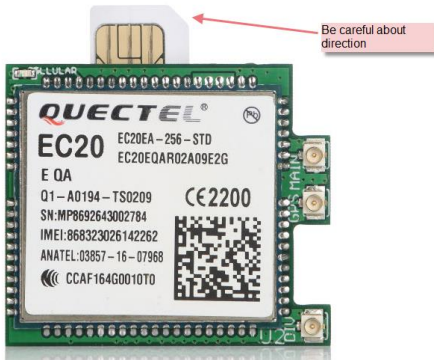
Encryption

In the WiFi Survey Choose the WiFi AP, and input the Passphrase then click Save & Apply to connect.

3.4 Use built-in 4G modem for internet access

If the LG01N has 3G/4G Cellular modem, user can use it as main internet connection or back up.

First, install the Micro SIM card as below direction



Second, Power off/ ON LG01N to let it detect the SIM card.

The set up page is [System](#) → [Cellular](#)

While use the cellular as Backup WAN, device will use Cellular for internet connection while WAN port or WiFi is not valid and switch back to WAN port or WiFi after they recover.

Cellular Settings

- Enable Cellular WAN
- Use Cellular as Backup WAN

APN:

Service:

Dial Number:




Pincode:

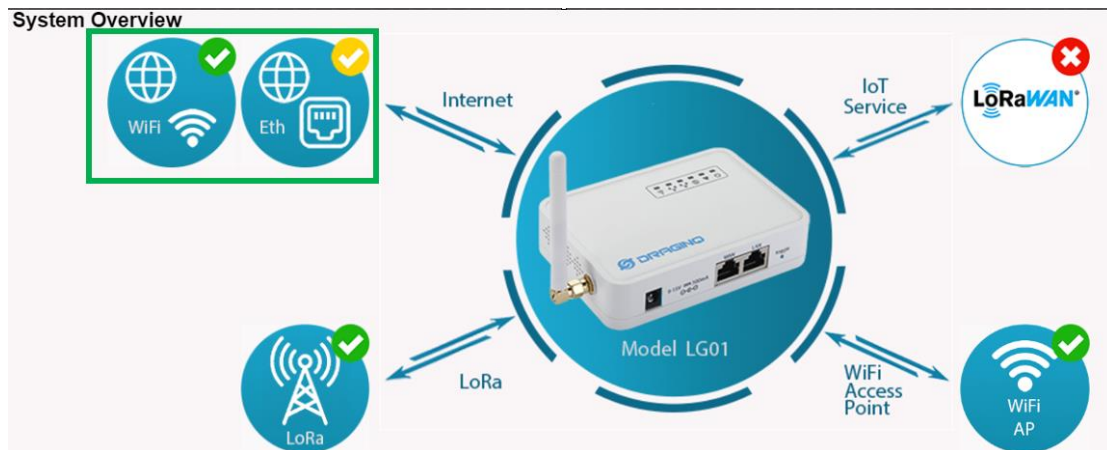
Username:

Password: [Show](#)

3.5 Check Internet connection

In the [Home](#) page, we can check the Internet connection.

- GREEN Tick  : This interface has Internet connection.
- Yellow Tick  : This interface has IP address but don't use it for internet connection.
- RED Cross  : This interface doesn't connected.



4. Example 1: Manually send / receive LoRa packets

4.1 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.

LoRa Configuration

Debug Level

Radio Settings

Frequency (Hz)	<input type="text" value="915000000"/>	RF Bandwidth (Hz)	<input type="text" value="125kHz"/>
Spreading Factor	<input type="text" value="SF10"/>	Coding Rate	<input type="text" value="4/5"/>
Preamble Length	<input type="text" value="8"/>	LoRa Sync Word	<input type="text" value="52"/>
RF Power (0-20) dBm	<input type="text" value="20"/>		

Static GPS coordinates ?

Enable Static GPS	<input checked="" type="checkbox"/>	Altitude (m)	<input type="text" value="450"/>
Latitude	<input type="text" value="22.700000"/>	Longitude	<input type="text" value="114.240000"/>

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](#).

4.2 Use pkt_fwd to transmit

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": "YEklB CaqCgADQAIAcQM6AP8B9TYzUA==" } }
```

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)

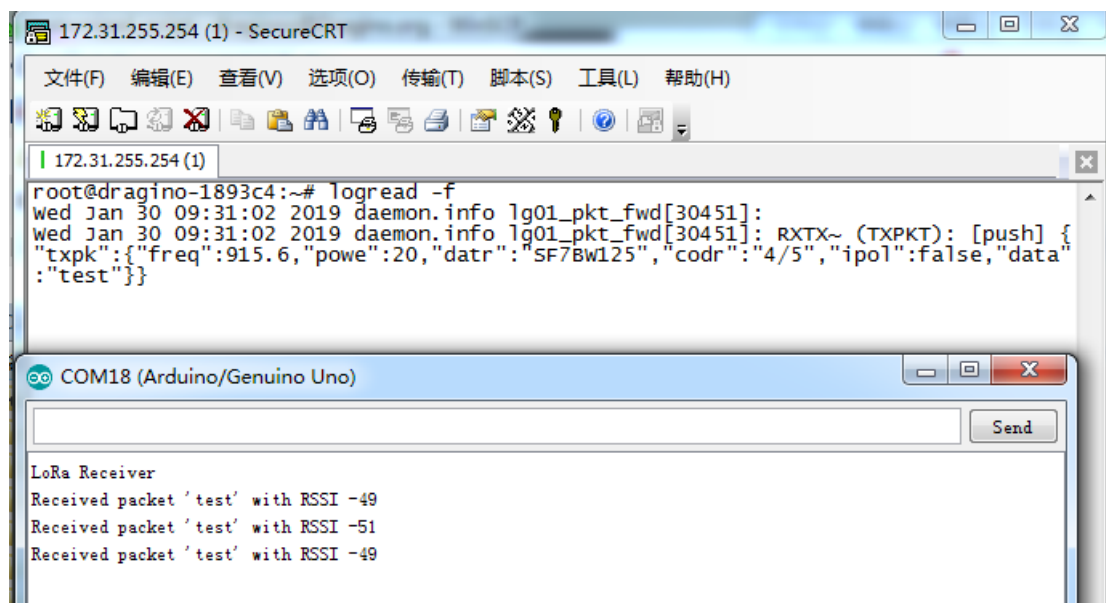
datr | number | FSK datarate (unsigned, in bits per second)
 codr | string | LoRa ECC coding rate identifier
 fdev | number | FSK frequency deviation (unsigned integer, in Hz)
 ipol | bool | Lora modulation polarization inversion
 prea | number | RF preamble size (unsigned integer)
 size | number | RF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

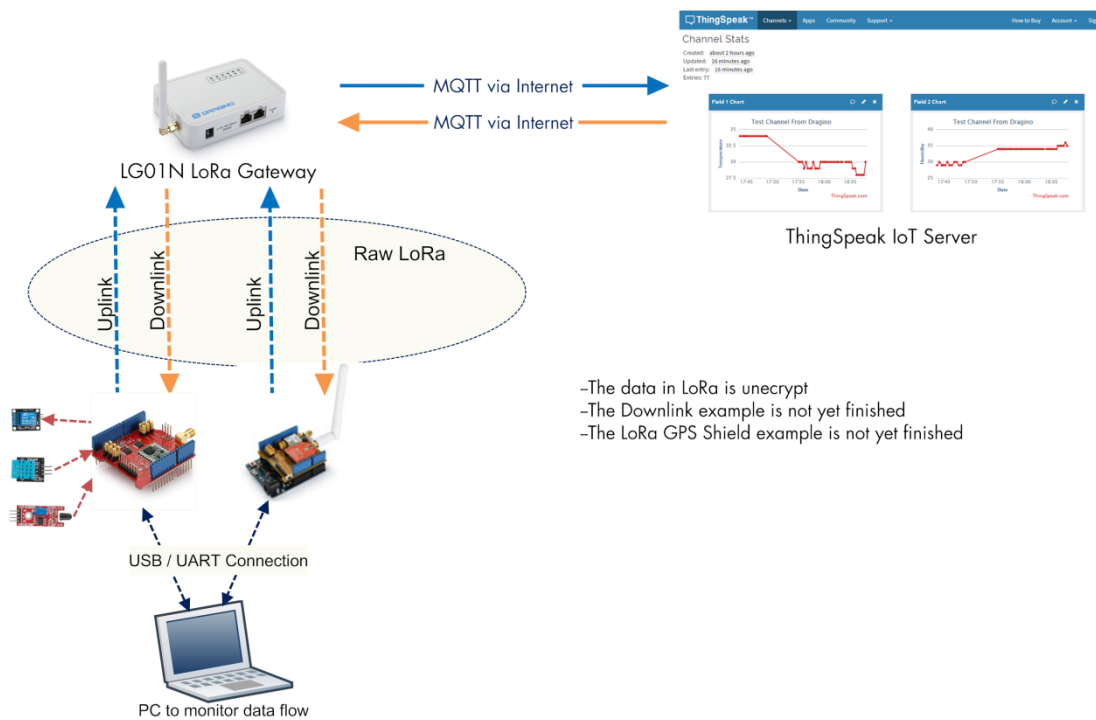


5. Example 2: MQTT Transfer Mode

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:

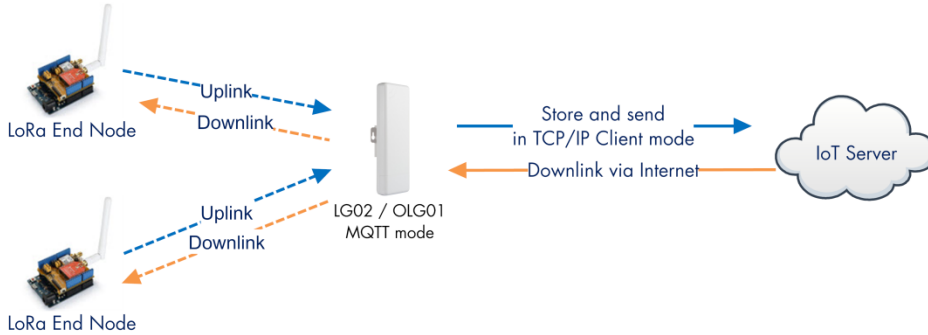
http://wiki.dragino.com/index.php?title=MQTT_Forward_Instruction

6. Example 3: 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.

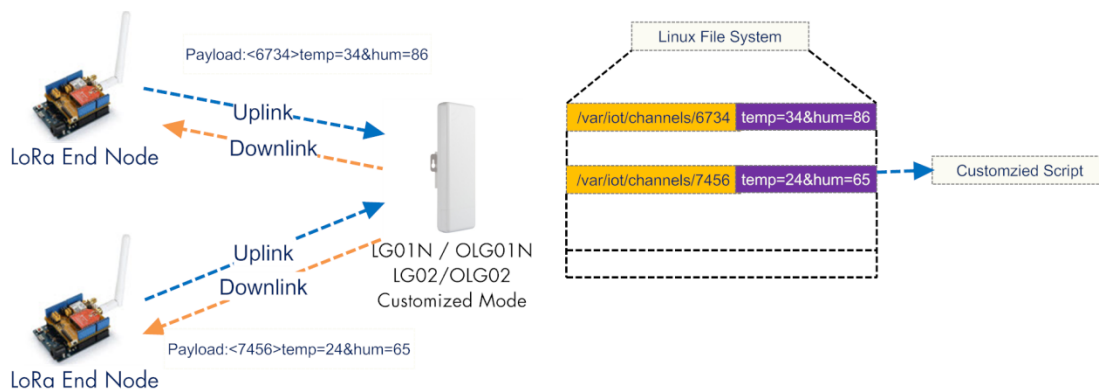
Tutorial: http://wiki.dragino.com/index.php?title=TCP_Connection_Instruction

7. Example 4: 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

 DRAGINO
LoRa LoRaWAN HTTP MQTT TCP Custom Network System

Custom Script

Run a Custom Script to process LoRa Data. Parameters are optional and defined in the script.

Script Name	<input type="text" value="store_data_to_file"/>		
Parameter 1	<input type="text" value="/var/sensor_data"/>	Parameter 2	<input type="text"/>
Parameter 3	<input type="text"/>	Parameter 4	<input type="text"/>
Parameter 5	<input type="text"/>	Parameter 6	<input type="text"/>
Parameter 7	<input type="text"/>	Parameter 8	<input type="text"/>
Parameter 9	<input type="text"/>	Parameter 10	<input type="text"/>

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 2: Configure LoRa Radio parameters to match the LoRa End Node.

LoRa Configuration

Debug Level

Radio Settings

Frequency (Hz)	<input type="text" value="915000000"/>	RF Bandwidth (Hz)	<input type="text" value="125kHz"/>
Spreading Factor	<input type="text" value="SF10"/>	Coding Rate	<input type="text" value="4/5"/>
Preamble Length	<input type="text" value="8"/>	LoRa Sync Word	<input type="text" value="52"/>
RF Power (0-20) dBm	<input type="text" value="20"/>		

Static GPS coordinates ?

Enable Static GPS	<input checked="" type="checkbox"/>	Altitude (m)	<input type="text" value="450"/>
Latitude	<input type="text" value="22.700000"/>	Longitude	<input type="text" value="114.240000"/>

Step 3: 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:

```
COM9
LoRa Sender
Sending packet: 0
Sending packet: 1
Sending packet: 2
Sending packet: 3
Sending packet: 4
Sending packet: 5
```

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
```

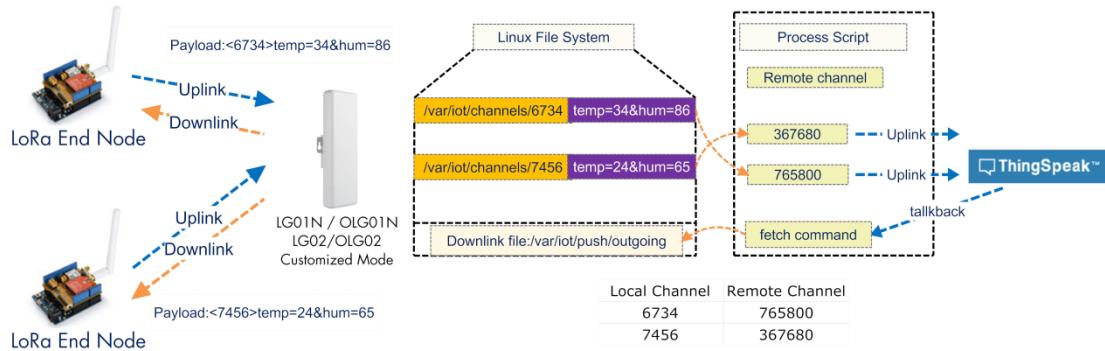
Script store data into file

```
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
```

8. Example 5: Communicate to a HTTP server

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

Communicate with thingspeak via HTTP GET/POST:



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.

8.1 Test uplink and downlink via Linux command

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

dragino-test

Channel ID: 396640
Author: dragino1
Access: Private

Private View Public View Channel Settings Sharing API Keys Data Import / Export

Write API Key

Key: P07KVY59P5QEY6M6

Generate New Write API Key

Read API Keys

Key: WJXRGTGMWADPVQNF

Note:

Help

API keys enable you to write data to a channel or read data from a private channel. API keys are auto-generated when you create a new channel.

API Keys Settings

- **Write API Key:** Use this key to write data to a channel. If you feel your key has been compromised, click **Generate New Write API Key**.
- **Read API Keys:** Use this key to allow other people to view your private channel feeds and charts. Click **Generate New Read API Key** to generate an additional read key for the channel.
- **Note:** Use this field to enter information about channel read keys. For example, add notes to keep track of users with access to your channel.

API Requests

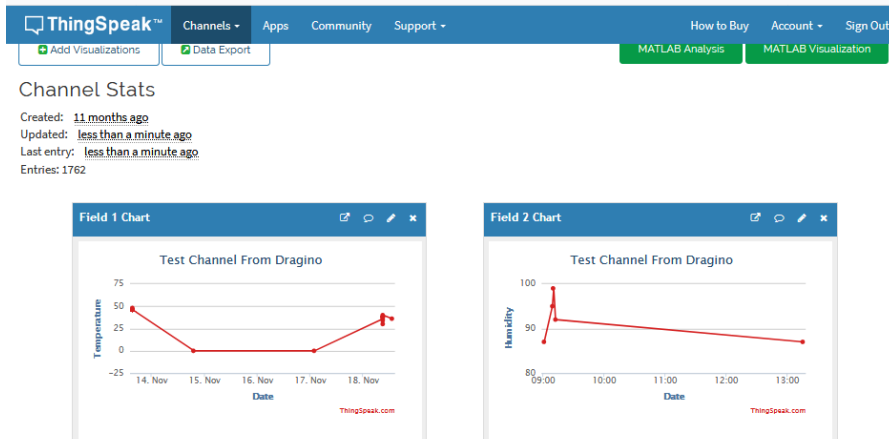
Update a Channel Feed

GET https://api.thingspeak.com/update?api_key=P07KVY59P5QEY6M6&field1=0

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

```
172.31.255.254 (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)
172.31.255.254 (1)
root@dragino-1893c4:~# curl -k --get "https://api.thingspeak.com/update?api_key=P07KVY59P5QEY6M6&field1=35&field2=98"
4310root@dragino-1893c4:~#
```

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

The screenshot shows the 'TalkBack' configuration page in the ThingSpeak interface. It includes a 'Help' section with 'Example API Endpoints':

- Add a TalkBack Command:** POST `https://api.thingspeak.com/talkbacks/30660/commands.json` with `api_key=J23X4Y9MTCZNH9Y0`
- Get a TalkBack Command:** GET `https://api.thingspeak.com/talkbacks/30660/commands/COMMAND_ID.json?api_key=J23X4Y9MTCZNH9Y0`
- Update a TalkBack Command:** PUT `https://api.thingspeak.com/talkbacks/30660/commands/COMMAND_ID.json` with `api_key=J23X4Y9MTCZNH9Y0`
- Execute the Next TalkBack Command:** POST `https://api.thingspeak.com/talkbacks/30660/commands/execute.json` with `api_key=J23X4Y9MTCZNH9Y0`

The main configuration area shows: Name: test, TalkBack ID: 30660, API Key: J23X4Y9MTCZNH9Y0, Created: 2019-01-30 5:11 am, and Logged to Channel: dragino-test.

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

```
172.31.255.254 (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)
172.31.255.254 (1)
root@dragino-1893c4:~# curl -k "https://api.thingspeak.com/talkbacks/30660/commands/execute.json" --data "api_key=J23X4Y9MTCZNH9Y0"
root@dragino-1893c4:~# curl -k "https://api.thingspeak.com/talkbacks/30660/commands/execute.json" --data "api_key=J23X4Y9MTCZNH9Y0"
{"id":15071098,"command_string":"downlinkcommand","position":null,"executed_at":"2019-01-30T10:22:38Z","created_at":"2019-01-30T10:22:29Z"}root@dragino-1893c4:~#
```

8.2 Test uplink and downlink in LoRa

8.2.1 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 `uplink_downlink_ThingSpeak.sh` for your channels:

There are three places 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.

Custom Script

Run a Custom Script to process LoRa Data. Parameters are optional and defined in the script.

Script Name

Parameter 1 Parameter 2

Parameter 3 Parameter 4

Parameter 5 Parameter 6

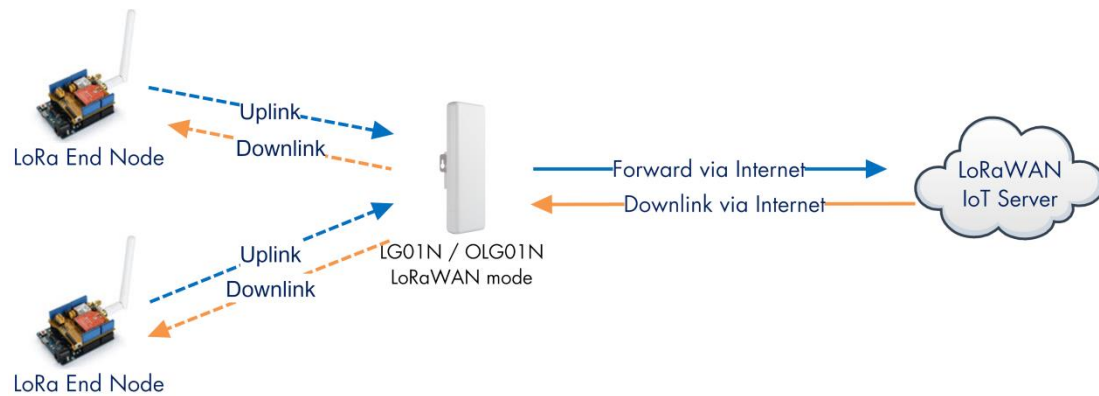
Parameter 7 Parameter 8

Parameter 9 Parameter 10

9. Example 6: 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.

The LG01-N is not recommend to be used a LoRaWAN gateway, Due to the limitation mentioned here: http://wiki.dragino.com/index.php?title=Limitation_of_Single_Channel_Gateway

If user can meet the limitation for the end node, user can refer this instruction:

http://wiki.dragino.com/index.php?title=Limitation_of_Single_Channel_Gateway#Set_Up_Dragino_End_Node_to_work_for_Single_Channel_Gateway_such_as_LG01.2FLG02 for how to set up the LG01-N to use for LoRaWAN network.

This chapter describes how to use LG01N to work with [TTN LoRaWAN Server](#). The method to work with other LoRaWAN Server is similar.

10. More features

10.1 Remote Access

Remote Access Devices for management:

See

http://wiki.dragino.com/index.php?title=Main_Page#Remote_Access_Gateway_via_Reverse_SSH

10.2 More instructions

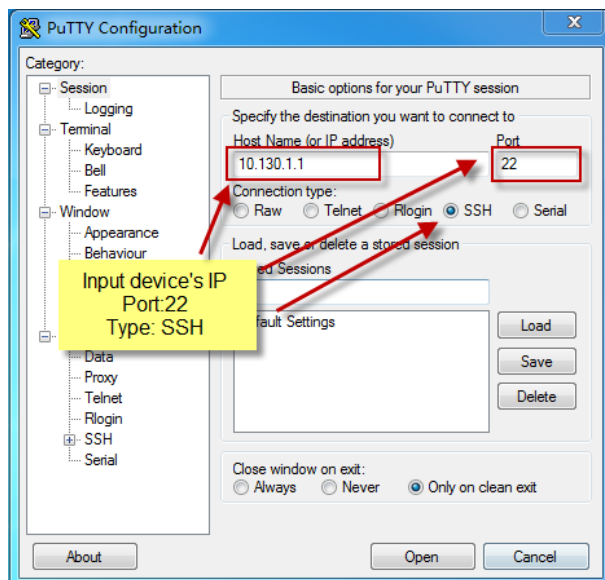
http://wiki.dragino.com/index.php?title=Main_Page#LoRa.2FLoRaWAN_Gateway_Instruction

11. 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.

11.1 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:



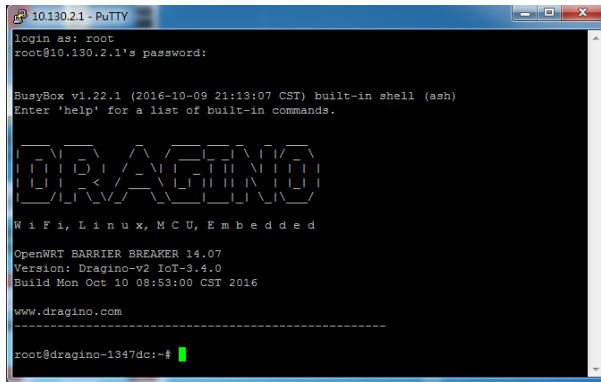
IP address: IP address of LG01N

Port: 22 or 2222

User Name: **root**

Password: **dragino** (default)

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



```
1013021 - PuTTY
login as: root
root@10.130.2.1's password:

BusyBox v1.22.1 (2016-10-09 21:13:07 CST) built-in shell (ash)
Enter 'help' for a list of built-in commands.

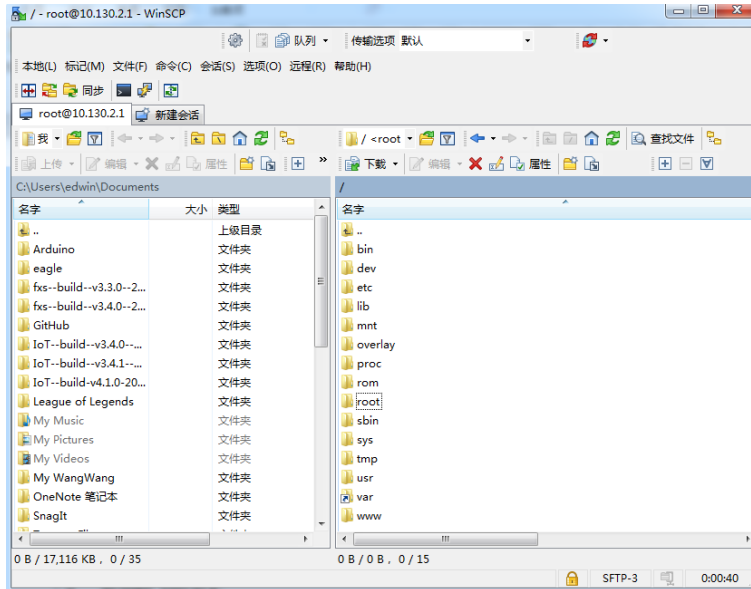
DRAGINO
W i F i , L i n u x , M C U , E m b e d d e d

OpenWRT BARRIER BREAKER 14.07
Version: Dragino-V2 IoT-3.4.0
Build Mon Oct 10 08:53:00 CST 2016

www.dragino.com
-----
root@dragino-1347dc:~#
```

11.2 Edit and Transfer files

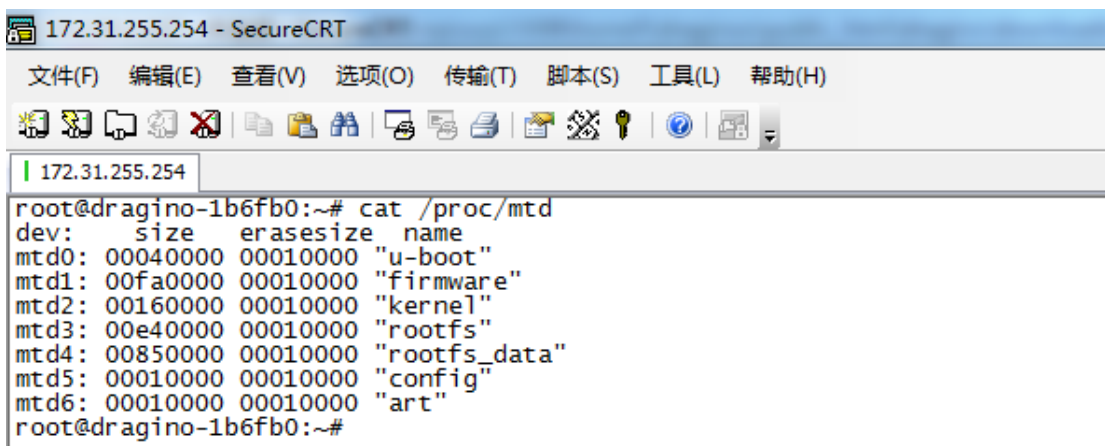
The LG01N support **SCP protocol** and has a built **SFTP server**. There are many ways to edit and transfer files using these two protocols. One of the easiest is through [WinSCP](#) utility. After access via WinSCP to the device, use can use a FTP alike window to drag / drop files to the LG01N or Edit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



11.3 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.

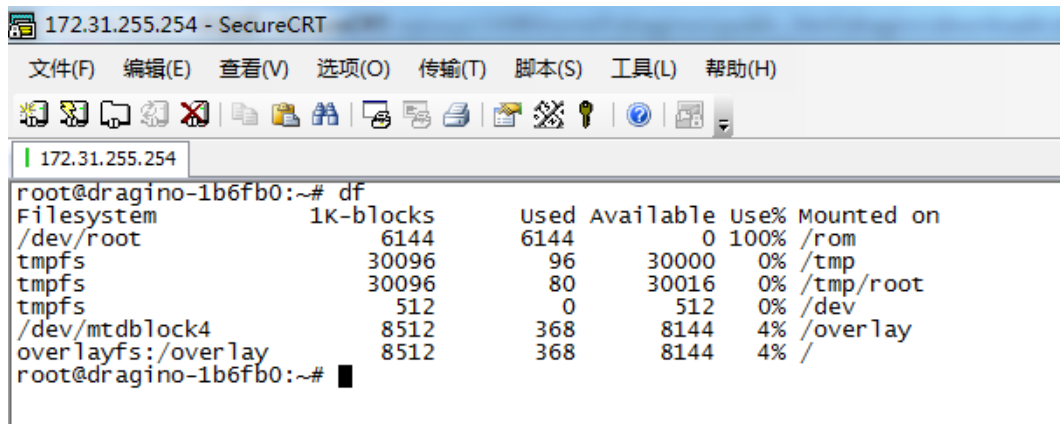
Use cat /proc/mtd to see all blocks/partitions.



- ✓ "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

11.4 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.
```

12. Upgrade Linux Firmware

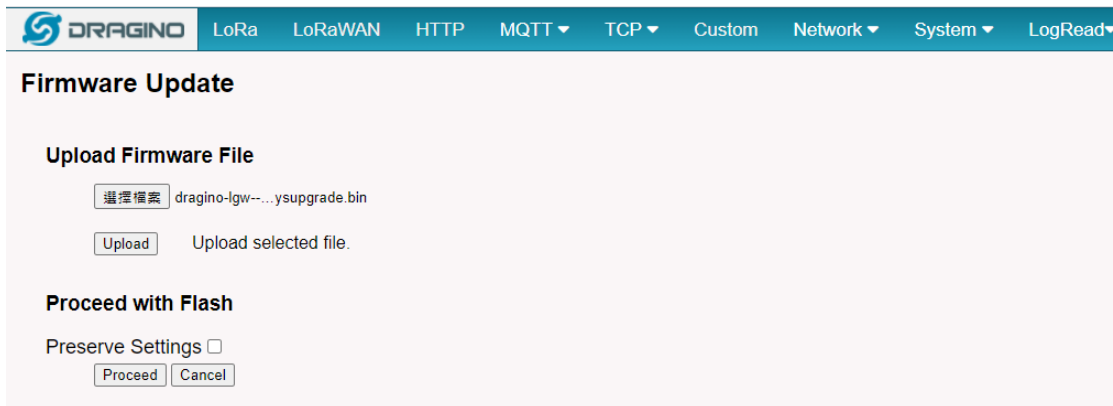
We keep improving the LG01N Linux side firmware for new features, bug fixes. The latest firmware can be found on [LG01N Firmware & release note](#)

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

12.1 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.



12.2 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.

13. FAQ

13.1 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.

13.2 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.

13.3 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.

13.4 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

13.5 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

13.6 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]:
wed Sep 12 01:39:19 2018 daemon.info lg02_pkt_fwd[14341]: INFO (json): [down] [{"txpk":{"time":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":"IIadGvuy4vL7RAFx5HIX0a="}]
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":{"time":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":"I0GNTMK9p5v1jF9BP1xbzvi="}]
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]:
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	Direction	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","ipol":true,"size":17,"ncrc":true,"data":"IglkyoueyXLoMTFSovbRB8="}}]
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]: 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]: 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]: Receive(HEX):00ac2301d07ed5b370907cb65d67c64a00cd3586bb5c88
wed Sep 12 02:11:36 2018 daemon.info lg02_pkt_fwd[20677]: INFO (json): [up] [{"rxpk":{"time":"2018-09-12T02:11:36.210520Z","tmst":472538269,"chan":0,"pfch":1,"freq":90
3.000000,"stat":1,"modu":"LORA","datr":{"sf7bw125","codr":"4/5","lsnr":7.8,"rssi":-34,"size":23,"data":"AkWjAd8+1bNwKHy2XwFGsqDNNya7XIq="}}]}]
注:此设备为开源设备,请在 Windows
```

▲ 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 D0 01 23 AC dev eui: 00 4A C6 67 5D B6 7C 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

13.7 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
```

13.8 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`

14. Trouble Shooting

14.1 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
```

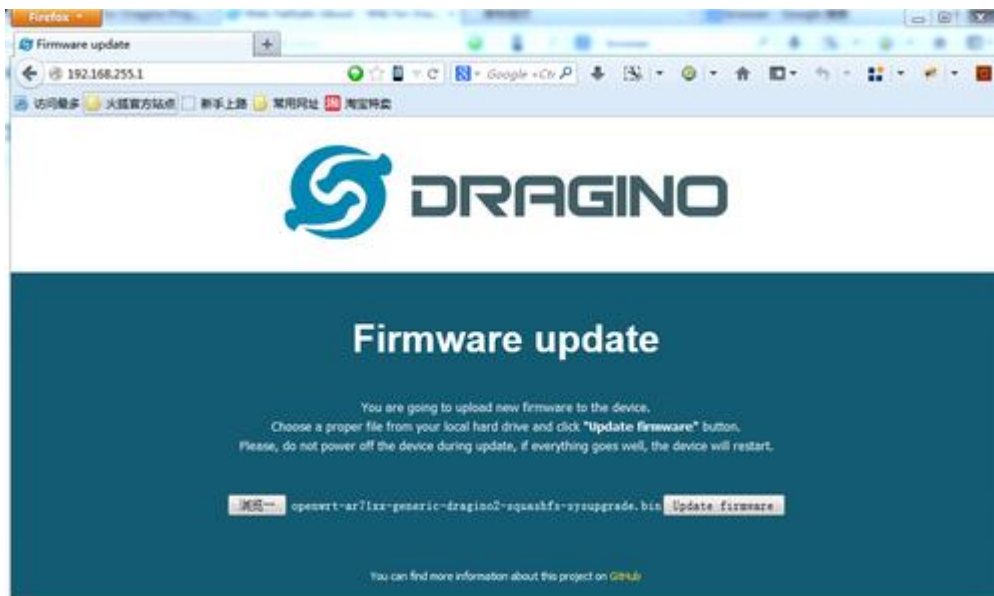
14.2 How to recover the LG01N if firmware crash

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

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



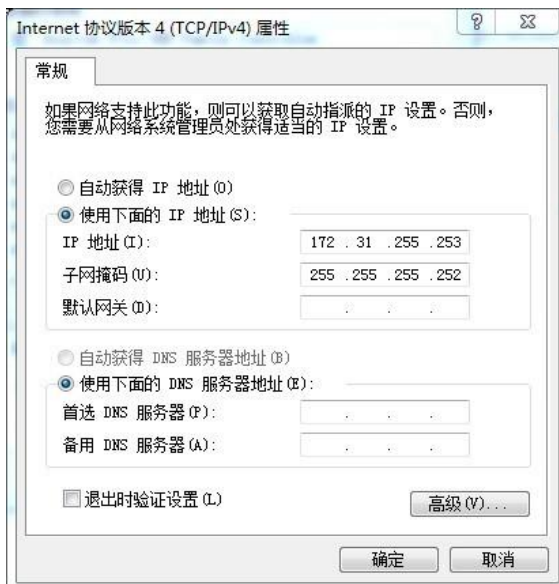
The LG01N has a fall-back ip in its LAN port. This IP is always enabled so user can use fall back ip to access LG01N no matter what the WiFi IP is. The fall back ip is useful for connect and debug the unit.

(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 LG01N'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 LG01N via Web or Console.

15. 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](#).

16. 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

17. 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

18. 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>