



LG01 LoRa Gateway User Manual

Document Version: 1.6

Firmware Version: IoT Mesh v4.3.5

Version	Description	Date
0.1	Initiate	2016-Oct-29
1.0	Release, Add ThingSpeak Examples	2016-Dec-9
1.1	Add Example List for connecting to TTN	2017-May-17
1.2	Add Link for RN2483 Support, OLG01 Antenna description , modify fallback ip description, Add More FAQ links, Console README, Web update sketch, Add Hardware Source code	2017-Sep-7
1.3	Fix fall back ip typo, Point to latest source code in both reference and how to. Modify Radiohead install text, modify simple LoRa client simple text. Improve Example with more explanation	2017-11-5
1.4	Add MQTT Examples and add watch dog feature	2018-04-03

1.5	Add the TCP example and fix the Gateway settings can't take effect bug and the problem with the lg01_pkt_fwd process.	2018-05-08
1.6	Fixed security holes, Update dropbear to 2017.7.5.Fix internet /DNS check issue.	2018-06-20

Index:

1. Introduction.....	6
1.1 What is LG01.....	6
1.2 Specifications.....	6
1.3 Features.....	9
1.4 System Structure.....	9
1.5 Applications.....	10
1.6 Hardware Variants.....	11
1.7 Install SIM card in EC20/UC20 3G/4G module.....	11
2. Quick Start Guide.....	12
2.1 Access and config LG01.....	12
2.2 Program microcontroller.....	13
2.2.1 Download and configure Arduino IDE.....	13
2.2.2 Upload a sketch in the MCU.....	15
2.3 Simple LoRa wireless example.....	17
2.3.1 Installing LoRa Library.....	18
2.3.2 Upload Sketch to LoRa Client.....	19
2.3.3 Upload Sketch to LoRa Gateway LG01.....	20
2.3.4 Analyze Test Result.....	21
3. Typical Network Setup.....	22
3.1 Overview.....	22
3.2 General WiFi AP Network.....	23
3.3 WAN Port Internet Mode.....	24
3.4 WiFi Client Mode.....	24
3.5 Mesh WiFi Network.....	25
3.5.1 Mesh Gateway Set Up.....	25
3.6 USB Dial Up Modem Set Up.....	29
3.7 USB 3G/4G Ethernet Dongle.....	30
4. Linux System.....	32
4.1 SSH Access for Linux console.....	32
4.2 Edit and Transfer files.....	33
4.3 File System.....	33
4.4 Package maintain system.....	34
5. Bridge Library.....	35
5.1 The Use of Console.....	36
6. Advance Management.....	37
6.1 Reset Network or Reset to Factory Default.....	37
7. Upgrade Linux Firmware.....	38
7.1 Upgrade via Web UI.....	38

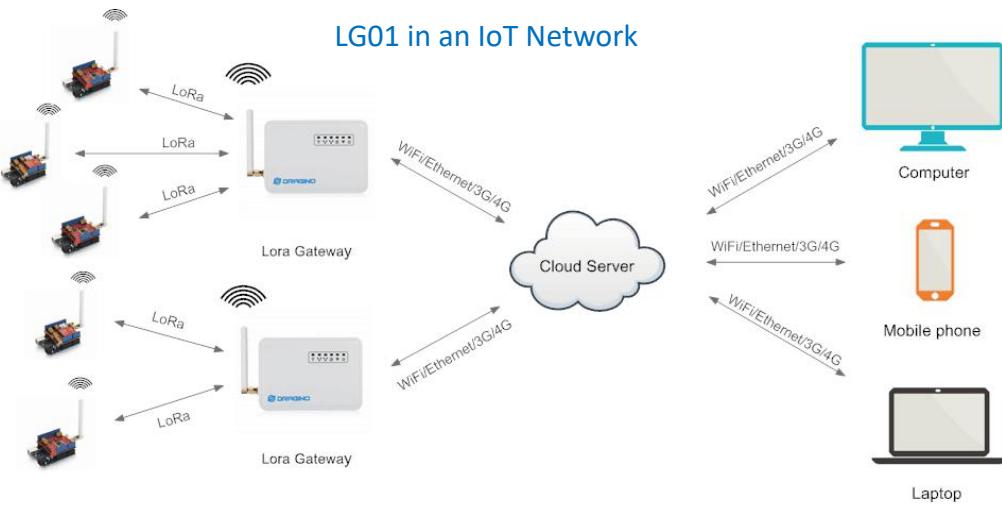
7.2 Upgrade via Linux console.....	38
8. Upgrade Micro Controller Sketch.....	39
8.1 Upgrade Sketch via Arduino IDE.....	39
8.2 Upgrade Sketch via Web UI.....	39
8.3 Auto update MCU.....	40
9. Example: Integrate LoRa with RESTFul API.....	41
9.1 What is RESTFul API?.....	41
9.2 Configure IoT Server.....	41
9.3 Step by Step Uplink Test.....	43
9.3.1 Try RESTful API call in web.....	44
9.3.2 Try RESTful API call with LG01 Linux command.....	46
9.3.3 Integrate LoRa, Bridge and Curl.....	47
9.4 Step by Step Downlink Test.....	48
9.4.1 Create Talkback command and try RESTful API call in web.....	49
9.4.2 Try RESTful API call with LG01 Linux command.....	51
9.4.3 Integrate LoRa, Bridge and Curl.....	52
10. Example: Integrate LoRa with MQTT API.....	53
10.1 What is MQTT API?.....	53
10.2 Step by Step Uplink Test.....	53
10.2.1 Simulate MQTT Publish via Desktop MQTT tool.....	54
10.2.2 Try MQTT Publish with LG01 Linux command.....	55
10.2.3 Integrate LoRa, Bridge and mosquitto_pub.....	57
10.3 Step by Step Downlink Test.....	61
11. Example: Integrate LoRa with TCP.....	62
11.1 What is TCP?.....	62
11.2 Gateway configuration.....	62
11.3 Simulation.....	64
11.4 Send data.....	67
12. Advance Examples.....	68
12.1 Example for Connecting to TTN LoRaWAN server.....	68
12.2 Multiple Nodes examples.....	68
12.3 How to use the sensor pin of LG01-S?.....	70
12.4 More Examples.....	72
13. FAQ.....	73
13.1 Why there is 433/868/915 version LoRa part?.....	73
13.2 What is the frequency range of LG01 LoRa part?.....	73
13.3 What kind of LoRa devices can the gateway support?.....	73
13.4 How many nodes can the LG01 support?.....	73
13.5 What kind of Server the LG01 can support?.....	73
13.6 Can I make my own firmware for LG01? Where can I find the source code of LG01?.....	74
13.7 How to get more examples for this device?.....	74

13.8 What is the Antenna require for OLG01?.....	74
13.9 More FAQs about general LoRa questions.....	74
14. Trouble Shooting.....	75
14.1 I can't download the Dragino profile in Arduino IDE.....	75
14.2 Bridge between MCU and Linux module doesn't work.....	76
14.3 Arduino IDE doesn't detect LG01.....	76
14.4 I get kernel error when install new package, how to fix?.....	76
14.5 How to recover the LG01 if firmware crash.....	77
14.6 I configured LG01 for WiFi access and lost its IP. What to do now?.....	78
14.7 See GPIO20 issue when uploading.....	78
15. Order Info.....	79
16. Packing Info.....	79
17. Support.....	79
18. Reference.....	81

1. Introduction

1.1 What is LG01

The LG01 is an open source single channel LoRa Gateway. It lets you bridge LoRa wireless network to an IP network base on WiFi, Ethernet, 3G or 4G cellular. LG01 runs on open source embedded Linux system; it has USB host port and has full Ethernet and 802.11 b/g/n WiFi capabilities. The USB host port can be used to connect cellular modules so LG01 is very flexible to bridge LoRa Network to different kinds of network to fit user's requirement.



1.2 Specifications

Hardware System:

Linux Part:

- 400Mhz ar9331 processor
- 64MB RAM
- 16MB Flash

MCU Part:

- MCU: ATmega328P
- Flash: 32KB
- SRAM: 2KB
- EEPROM: 1KB

Interface:

- Power Input: 9 ~ 24v DC
- 2 x RJ45 ports
- USB 2.0 Host port x 1
- Internal USB 2.0 Host Interface x 1

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 [EC20 LTE module](#)
- Micro SIM Slot
- Internal 4G Antenna + External 4G Sticker Antenna.
- Up to 100Mbps 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

Cellular 3G UMTS/HSPA+ (optional):

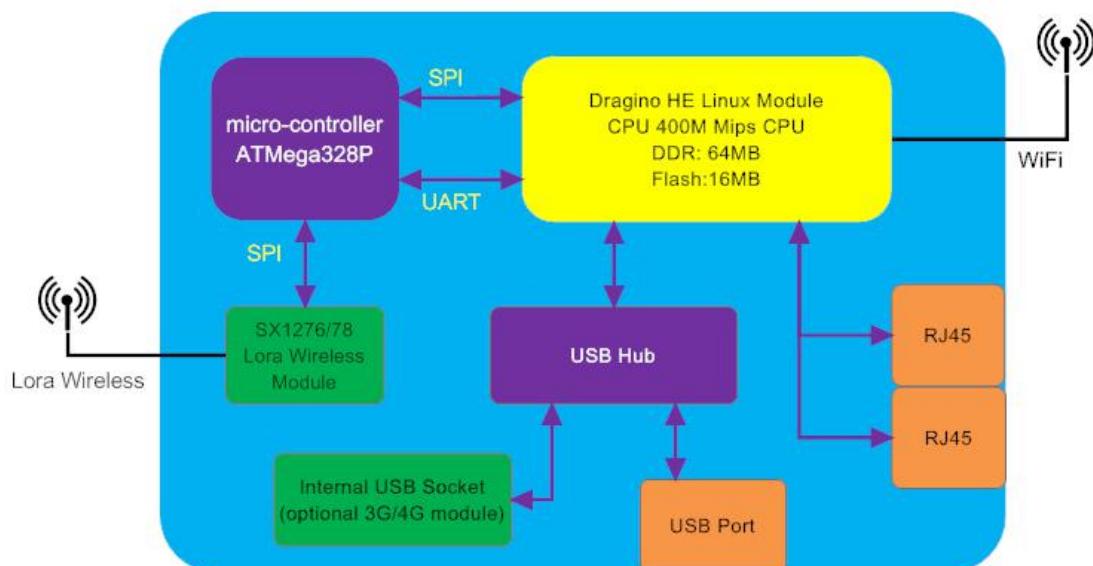
- Quectel [UC20 LTE module](#)
- Micro SIM Slot
- Internal 3G/4G Antenna + External 3G/4G Sticker Antenna.
- Up to 14.4Mbps downlink and 5.76Mbps uplink data rates
- Worldwide UMTS/HSPA+ and GSM/GPRS/EDGE coverage
- High-quality data and image transmission even in harsh environment
- Primary and diversity receive paths are designed for equivalent noise-figure performance

1.3 Features

- ✓ Open source Linux (OpenWrt) inside. User can modify or compile the firmware with custom features and own brand.
- ✓ Low power consumption.
- ✓ Compatible with Arduino IDE 1.5.4 or later, user can program, debug or upload sketch to the MCU via Arduino IDE.
- ✓ Managed by Web GUI, SSH via LAN or WiFi.
- ✓ Software upgradable via network.
- ✓ Auto-Provisioning.
- ✓ Built-in web server.
- ✓ Support internet connection via LAN port, WiFi or 3G /4G dongle.
- ✓ Failsafe design provides robustly system.

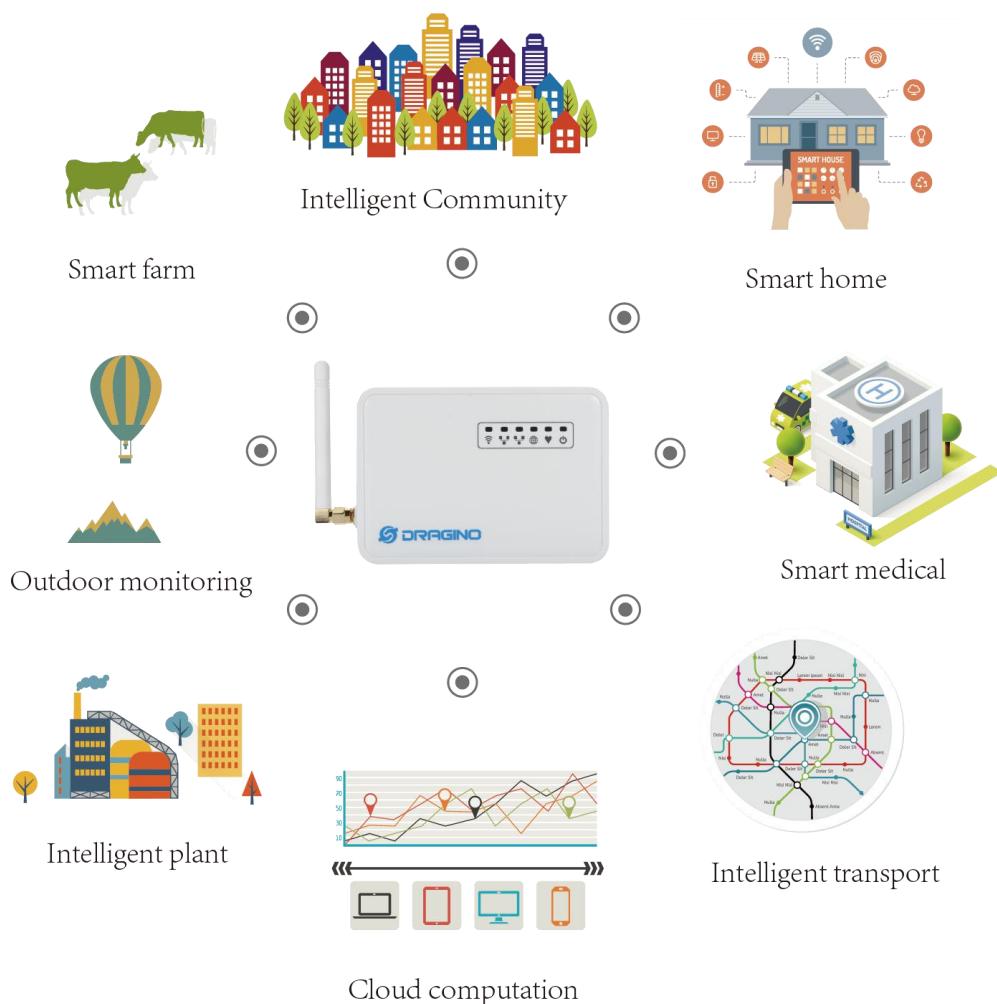
1.4 System Structure

LG01 System Overview:



1.5 Applications

Dragino Lora Gateway for IoT Applications



1.6 Hardware Variants

There are different LG01 variants for difference applications. Below table shows the difference between these models.

Model	Photo	Description
LG01-P		The most general version can be used as a LoRa Gateway.
LG01-S		Include screw terminal which can connect to external sensors
OLG01		Outdoor version, this version doesn't include LoRa antenna, instead, it has a SMA connector, user can connect it to a high gain LoRa antenna. It can be powered by a passive PoE adapter.

1.7 Install SIM card in EC20/UC20 3G/4G module

Please use below direction to install the SIM card.



2. Quick Start Guide

2.1 Access and config LG01

The LG01 is configured as a WiFi AP by factory default. User can access and configure the LG01 after connect to its WiFi network.

At the first boot of LG01, it will auto generate an unsecure WiFi network call **dragino2-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**



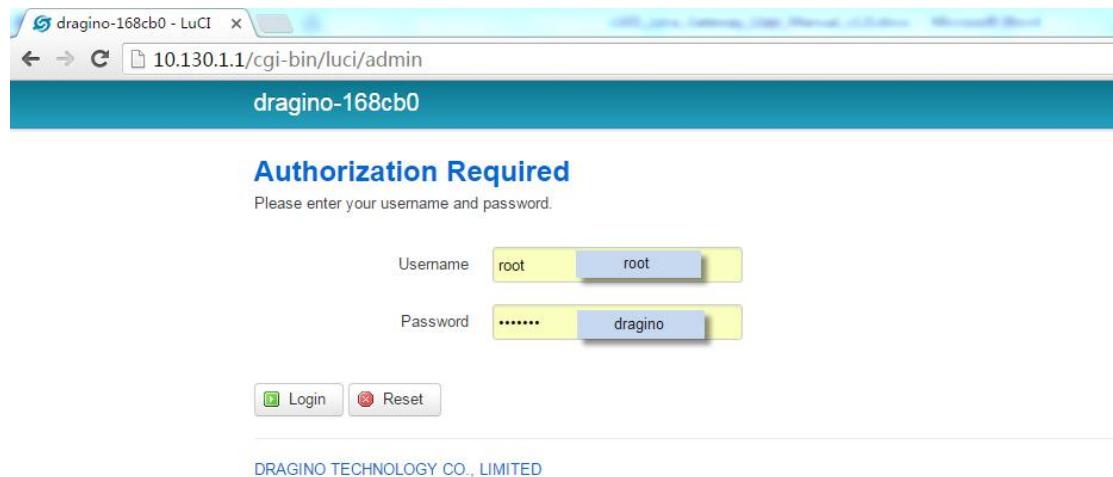
Open a browser in the laptop and type 10.130.1.1

User will see the login interface of LG01.

The account for Web Login is:

User Name: root

Password: dragino

A screenshot of a web browser window titled 'dragino-168cb0 - LuCI'. The URL bar shows '10.130.1.1/cgi-bin/luci/admin'. The main content area displays a 'Authorization Required' page. It asks 'Please enter your username and password.' There are two input fields: 'Username' containing 'root' and 'Password' containing 'dragino'. Below the fields are 'Login' and 'Reset' buttons. At the bottom of the page, it says 'DRAGINO TECHNOLOGY CO., LIMITED'.

2.2 Program microcontroller.

The MCU (microcontroller) M328P is used to communicate with LoRa Radio part and Dragino Linux module. The program language for the MCU is based on C and program tool is Arduino IDE. Below the way shows how to do program it.

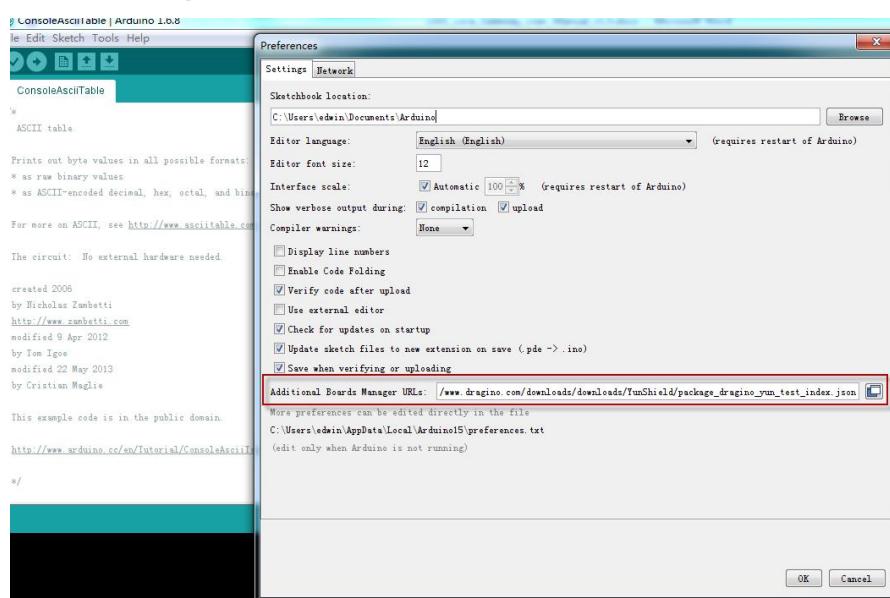
2.2.1 Download and configure Arduino IDE

- Download the latest Arduino Software(IDE) from Arduino official site:

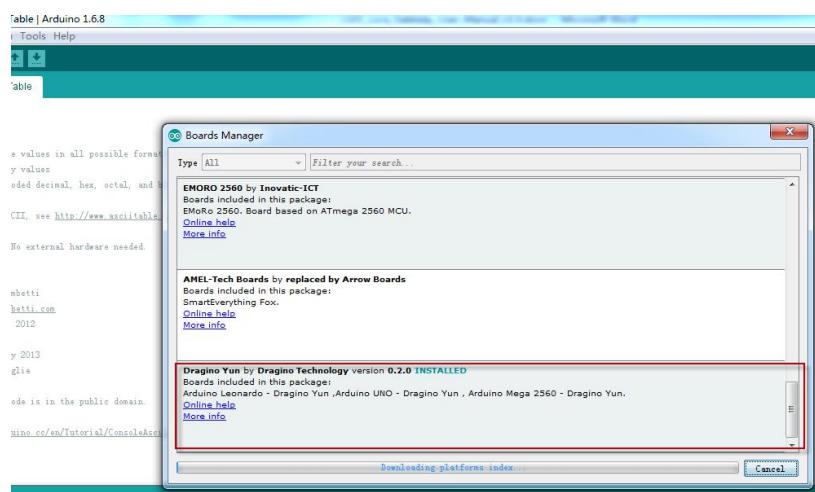
<https://www.arduino.cc/en/Main/Software>

Install the IDE in the PC, open it and click **File --> Preference**, add below URL

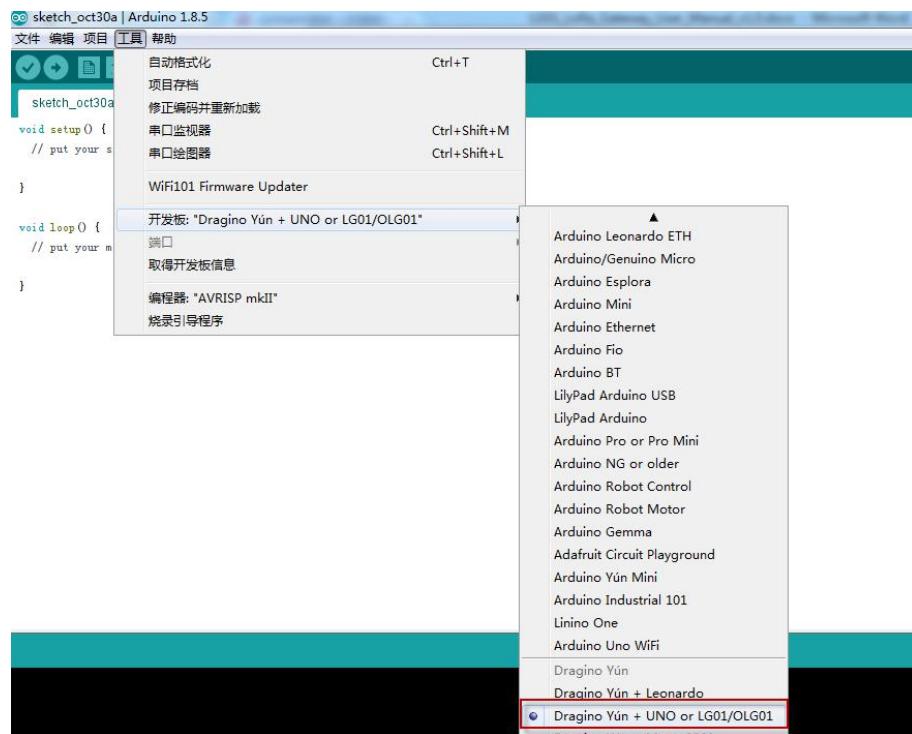
http://www.dragino.com/downloads/downloads/YunShield/package_dragino_yun_test_index.json in the **Additional Boards Manager URLs**.



- Go to **tools --> Boards --> Boards Manager**, find Dragino boards info and install it.



- After install Dragino's board info in the IDE, we can see the boards info from the IDE, as below screenshot. For LG01, we should choose: **Dragino Yun-UNO or LG01/OLG01**.

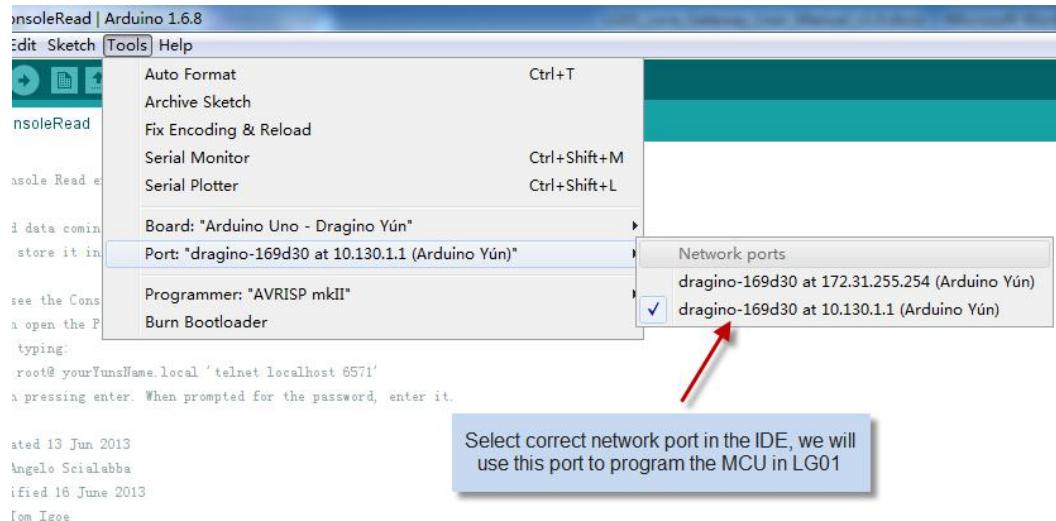


Note: If user has trouble to install via Board Manager. User can [manually add the board profile](#).

2.2.2 Upload a sketch in the MCU

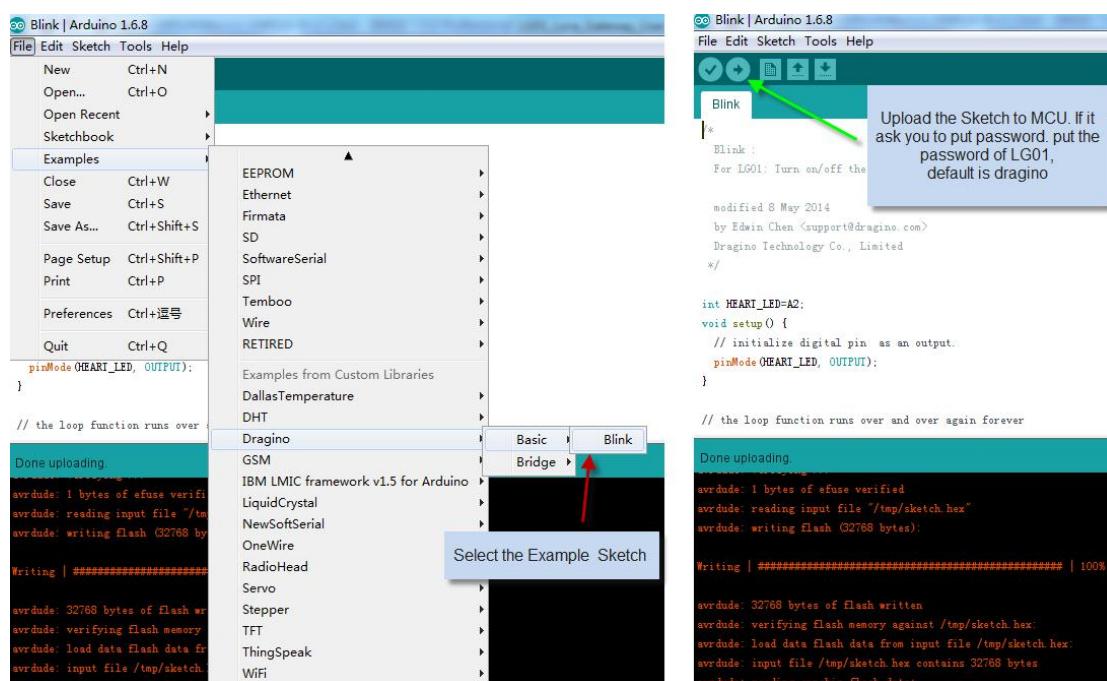
For the very start, we can try to upload a simple sketch to the MCU and see how it works.

- Make sure your computer and the LG01 is in the same network, if you already connect to the LG01 WiFi SSID, then both devices are in the same WiFi network. In the IDE, select the correct port as below screenshot.



- Select the example from **IDE --> File --> Examples --> Dragino --> Basic --> Blink**

Click **Upload** to upload the sketch to LG01, the LG01 may ask you password to upload, if so, type the password of LG01.



➤ Check result

The blink sketch will set the A2 pin of the MCU to periodically high and low. This pin connects to the **HEART LED** of the LG01. So if successfully upload this sketch, user can see the **HEART LED** turn on and turn off periodically.

2.3 Simple LoRa wireless example

To test LoRa wireless, we at least need 2 devices both support LoRa. In this example we will use below devices:

- ✓ LoRa Gateway: LG01 ;
- ✓ LoRa Client: LoRa Shield + Arduino Uno

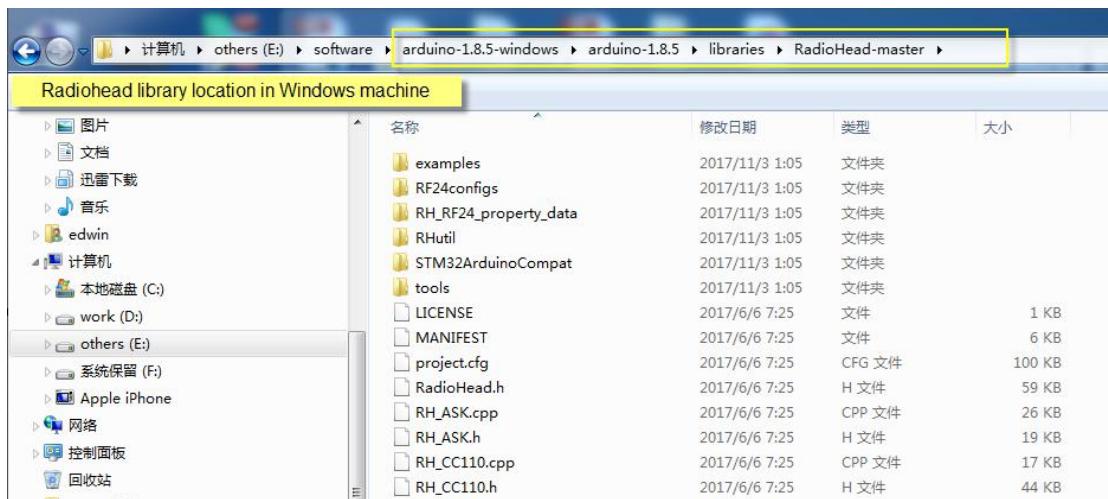


In this example, we will show the basic LoRa Communication: The LoRa Client will broadcast a data via LoRa wireless, the LG01 gateway will get this data and show the data in the PC.

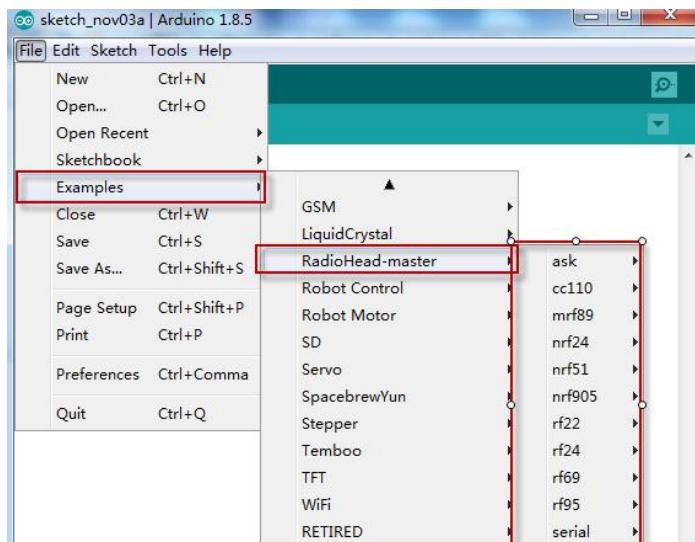
2.3.1 Installing LoRa Library

The Library used here for LoRa communication is called Radiohead; our sketch will use this library so we need to download it from:

<https://github.com/dragino/RadioHead/archive/master.zip>. Unzip and put it in the Arduino Library Folder, the final location looks as below:



To make sure the library is installed correctly, we can restart the Arduino IDE and see if we can find it in the example code, as shown below:

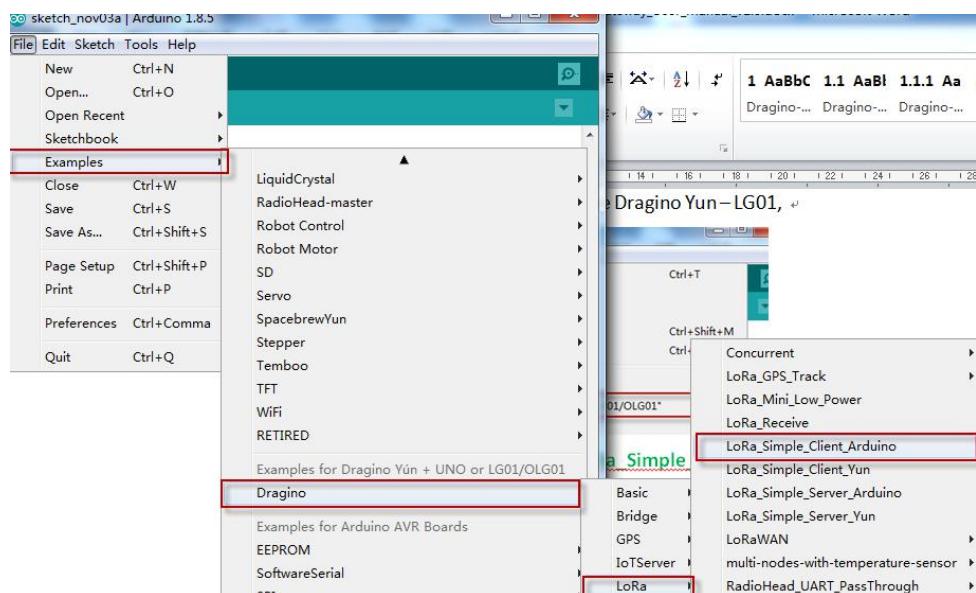


2.3.2 Upload Sketch to LoRa Client

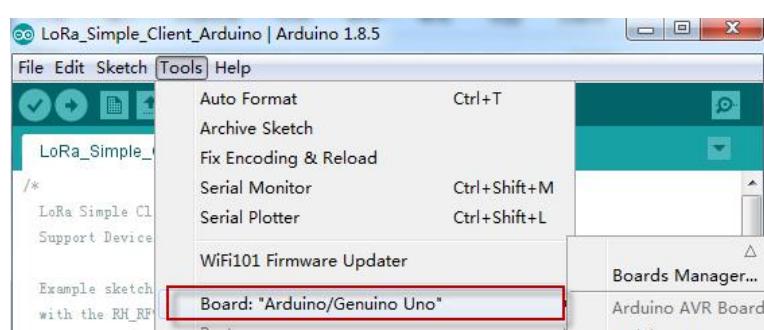
- a) In the Arduino IDE, first choose Dragino Yun – LG01,



- b) Then choose the example: [LoRa_Simple_Client_Arduino](#)



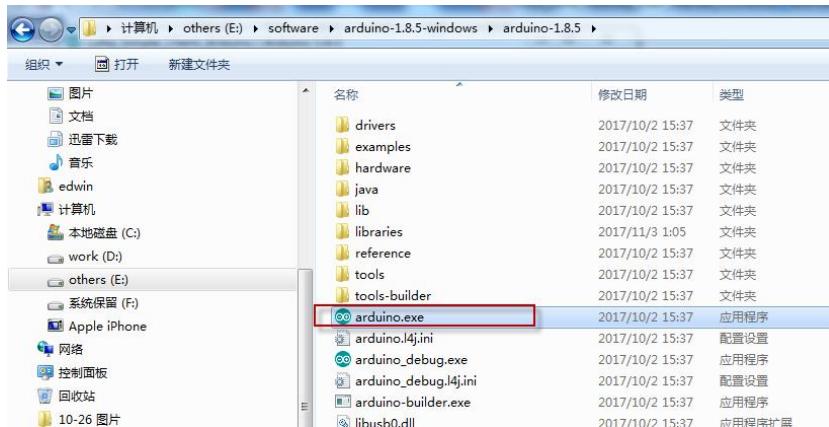
- c) In the window of [LoRa_Simple_Client_Arduino](#), choose the board Arduino Uno which is the correct board for LoRa Shield + UNO:



- d) Unload the [LoRa_Simple_Client_Arduino](#) example sketch to LoRa Shield + UNO via the USB com port. And then open serial monitor to see the output.

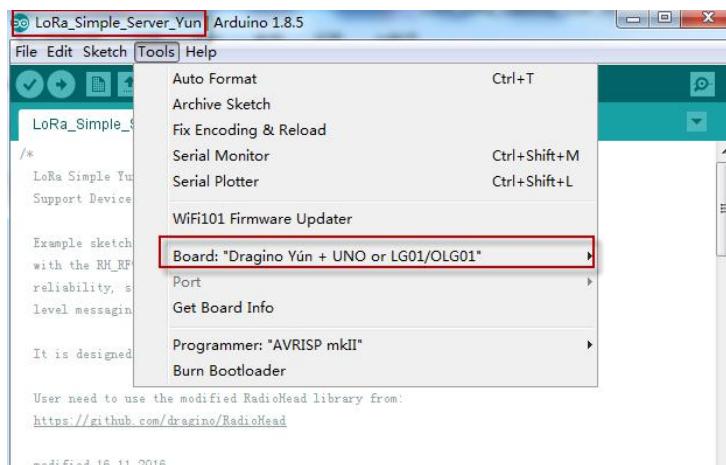
2.3.3 Upload Sketch to LoRa Gateway LG01

- a) Click the Arduino.exe to open another new instance. It is very important to open a new instance so we can two serial monitor, one for LoRa Client and one for LG01.

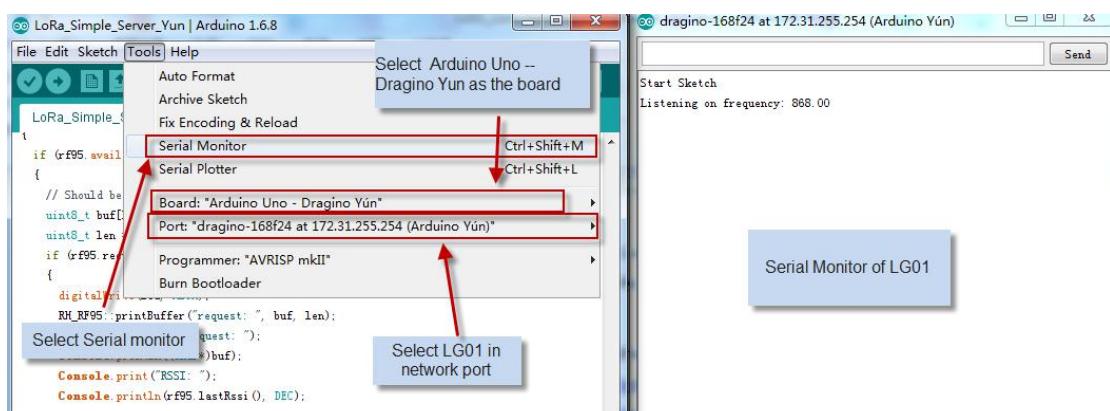


- b) In this new instance, Select LG01 board in Arduino IDE and choose the example

LoRa_Simple_Server_Yun



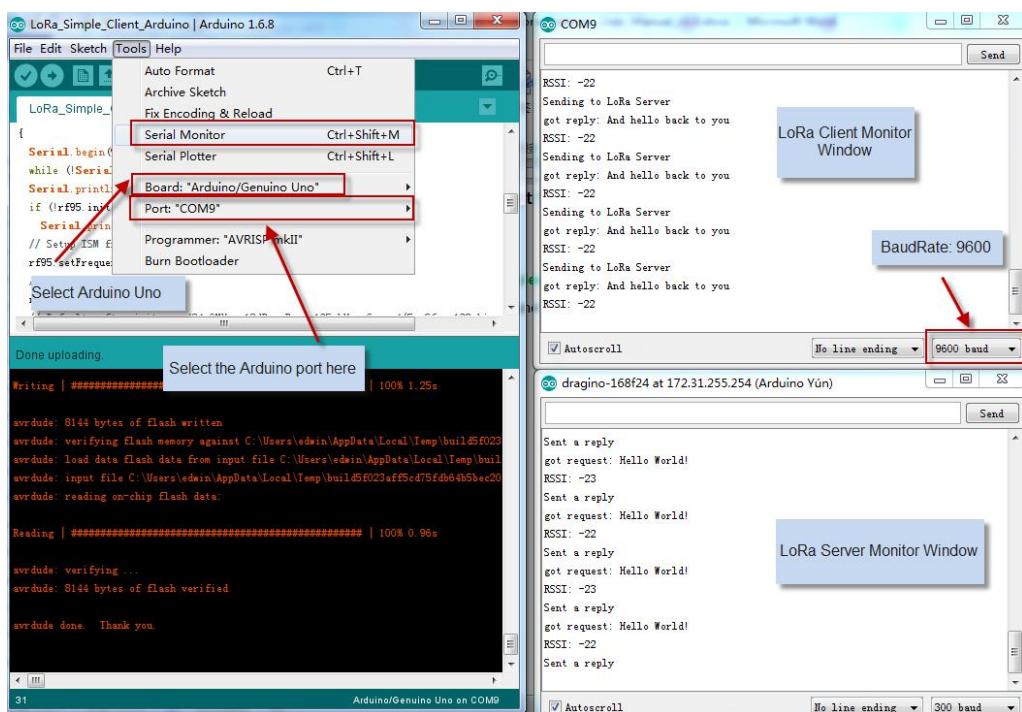
- c) Upload this Sketch to the LG01, then open the serial monitor to monitor the LG01 status.



2.3.4 Analyze Test Result

Below screenshot shows the result of this example.

- ✓ The Upper window shows the LoRa Client keep sending the LoRa packet out, and wait for the reply of this packet.
- ✓ The lower window is from LG01, which shows LG01 get a “Hello World” packet via LoRa, after LG01 get the “Hello World” packet, it will sent a broadcast LoRa packet “And hello back to you”, the LoRa Client will then receive it and print it on the serial monitor.



Notice: in the example code, the LoRa client will broadcast LoRa Packet once power on. But the LG01 will only receive the LoRa packet and response after the serial monitor of LG01 is opened, the reason is we have this code

```
while (!Console) ; // Wait for console port to be available
```

which mean the program will loop here until we open the serial monitor.

It is possible to use another LG01 as LoRa Client:

Method is same as above, but the example sketch is:

IDE --> File --> Examples --> Dragino --> LoRa --> LoRa_Simple_Client_Yun

3. Typical Network Setup

3.1 Overview

LG01 support flexible network set up for different environment. This section describes the typical network topology can be set in LG01. The typical network set up includes:

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

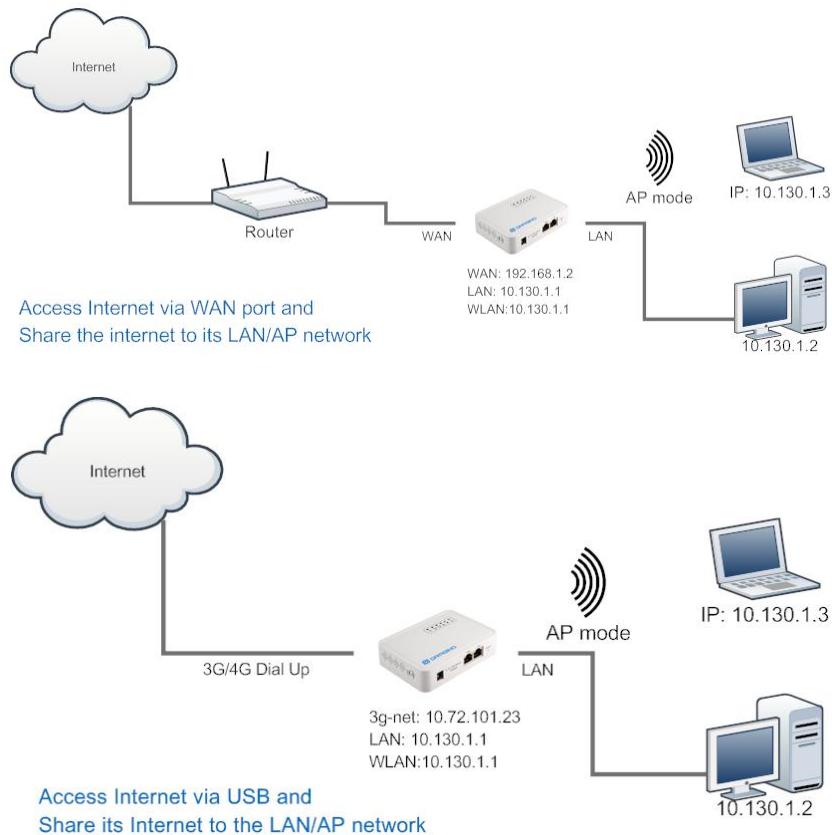
Before setting network parameters, user can set up the Display Net Connection under **Web --> Network --> Internet Access** for easily checking network status. LG01 will check the network connection to the host and show the status in **GLOBAL LED**. The **GLOBAL LED** will blink if LG01 has network connection to that host.



The network settings are under the network section, in the follow section of this chapter, we will show how to configure the LG01 for typically network usage.

3.2 General WiFi AP Network

In the General AP Mode, LG01 gets internet access from its WAN port or USB 3G/4G/GPRS. LG01 itself acts as a WiFi Access Point and provide a WiFi AP network. LG01 shares the internet to its AP network or LAN interface. Diagram is as below:



Set Up in Web UI for General WiFi AP mode

Network --> Internet Access:

- ✓ Access Internet via WAN PORT or USB Modem

Network --> LAN and DHCP

- ✓ Enable DHCP server in its LAN port

Network --> Access Point

- ✓ Enable WiFi AP
- ✓ Input SSID/ Encryption/ Passphrase

Network --> Mesh Network:

- ✓ Disable WiFi Mesh Network

3.3 WAN Port Internet Mode

The LG01 will use WAN port for internet connection. When connect LG01's WAN port to router, LG01 will get IP from router and have internet access. This is the default setting. It also shares the internet to its LAN port and WiFi AP network for other devices.

3.4 WiFi Client Mode

In the WiFi Client Mode, Dragino acts as a WiFi client and gets DHCP from uplink router via WiFi. It also shares the internet to its LAN port for other devices.



Set Up in Web UI

Network --> Internet Access:

- ✓ Access Internet via WiFi Client
- ✓ Way to Get IP: DHCP
- ✓ Input correct SSID, Password and Encryption.

Network --> LAN and DHCP

- ✓ Enable DHCP server in its LAN port

Network --> Access Point

- ✓ Disable WiFi AP

Network -->Mesh Network

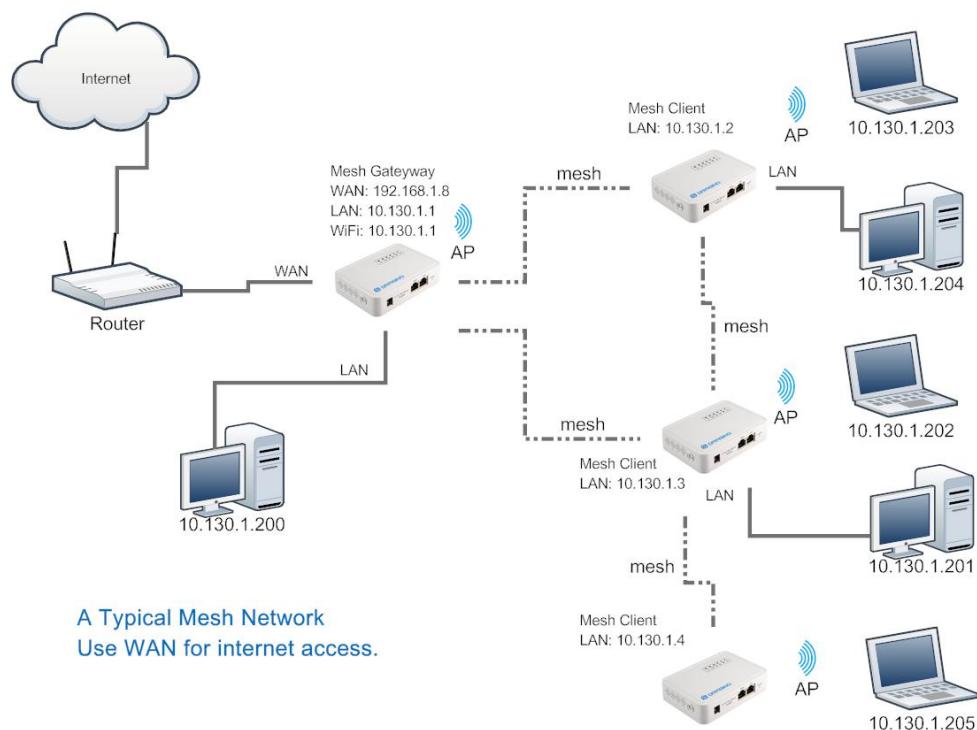
- ✓ Disable WiFi Mesh Network

3.5 Mesh WiFi Network

In the mesh network topology, user should choose the device to be a mesh gateway node or mesh client node

Mesh Gateway: use WAN port or USB 3G/4G modem to get internet access from uplink router. It also shares the internet to its Mesh Network for other Mesh Clients. The Mesh Gateway also acts as a DHCP router for its mesh network.

Mesh Client: Connects to the Mesh Gateway via mesh network, it also bridge internet via the LAN and WiFi AP interface.



3.5.1 Mesh Gateway Set Up

Network --> Internet Access

Access Internet via WAN PORT or USB Modem

dragino2-3ccaeef	Status	System	Sensor	Network	Logout
------------------	--------	--------	--------	---------	--------

Small Enterprise-Campus Network

Internet Access

Access Internet Via	<input type="button" value="WAN Port"/> <input type="button" value="USB Modem"/>
Way to Get IP	<input type="button" value="DHCP"/> <input type="button" value="Static"/>
Display Net Connection	<input type="text" value="www.163.com"/> <input type="checkbox"/> Continously Check Net Connection

- ✓ Select Internet Connection Method for mesh gateway

Network --> LAN and DHCP

dragino2-4dffbf Status ▾ System ▾ Sensor ▾ Network ▾ Logout

Small Enterprise-Campus Network

LAN and DHCP

Gateway Node Settings

IP Address	10.130.1.1	IP Address for its LAN and AP interface.
Enable DHCP	<input checked="" type="checkbox"/>	Enable DHCP Server
Authoritative	<input type="checkbox"/>	Enable DHCP Authoritative
LAN Gateway	255.255.255.255	Packets from LAN port and WiFi Interface (AP and Mesh) will be forwarded to its WAN interface
Subnet Mask	255.255.255.0	
DHCP Start IP	10.130.1.200	

- ✓ Enable DHCP server in its LAN port
- ✓ Set Gateway to 255.255.255.255

Network --> Access Point

- ✓ Enable WiFi AP (not necessary), Can set same SSID in the mesh network

Small Enterprise-Campus Network

Access Point

Enable WiFi AP	<input checked="" type="checkbox"/>
Station ID	Dragino2-3ccae
Encryption	WPA2
Passphrase	*****
Channel	Channel 6
AP Connections	30

Network --> Mesh Network

- ✓ Enable WiFi Mesh Network
- ✓ Input Mesh Group //Note: Mesh Device within same group can communicate with each other. Mesh Group is a translation for BSSID for easy configure and remember.

dragino-169d30 Status ▾ Sensor ▾ System ▾ Network ▾ Logout

Small Enterprise-Campus Network

Mesh Setting

Mesh devices with the same group ID and AP wifi channel can communicate with each other

Enable Mesh	<input checked="" type="checkbox"/>
Group ID	10000
<small>Input a number between 1 ~ 1099511627775</small>	

Mesh Gateway

Gateway Mode	OFF
--------------	-----

Mesh Client Set Up

Network --> Internet Access

- ✓ Access Internet set to Disable

dragino2-f531b1 Status ▾ System ▾ Sensor ▾ Network ▾ Logout

Small Enterprise-Campus Network

Internet Access

Access Internet Via

Disable its WAN access, so packets will pass to Mesh Interface.

Display Net Connection Continously Check Net Connection

Network --> LAN and DHCP

- ✓ Disable DHCP server in its LAN port and
- ✓ Set gateway point to mesh gateway.

dragino2-f531b1 Status ▾ System ▾ Sensor ▾ Network ▾ Logout

Small Enterprise-Campus Network

LAN and DHCP

IP Address Set a unique IP address for its LAN and WiFi interface.

Enable DHCP Enable DHCP Server Disable DHCP server in this device.

LAN Gateway Use the Gateway Node as Default Gateway

Enable Fallback IP Fallback IP is permanent IP in LAN port, active after reboot

Network --> Access Point

- ✓ Enable WiFi AP (not necessary, can use same SSID or difference SSID with other mesh node)

dragino2-b170b1 Network

No password set!

There is no password set on this router. Please configure a root password to protect the web interface and
[Go to password configuration...](#)

Small Enterprise-Campus Network

Access Point

Enable WiFi AP Enable WiFi AP

Station ID

Encryption

Passphrase 

CAN/US Reg

Channel

Network --> Mesh Network

- ✓ Enable WiFi Mesh Network
- ✓ Input Mesh Group //Note: Mesh Device within same group can communicate with each other. Mesh Group is a translation for BSSID for easy configure and remember.

dragino-169d30 Status ▾ Sensor ▾ System ▾ Network ▾ Logout

Small Enterprise-Campus Network

Mesh Setting

Mesh devices with the same group ID and AP wifi channel can communicate with each other

Enable Mesh Enable Mesh Network

Group ID
Input a number between 1 ~ 1099511627775

Mesh Gateway

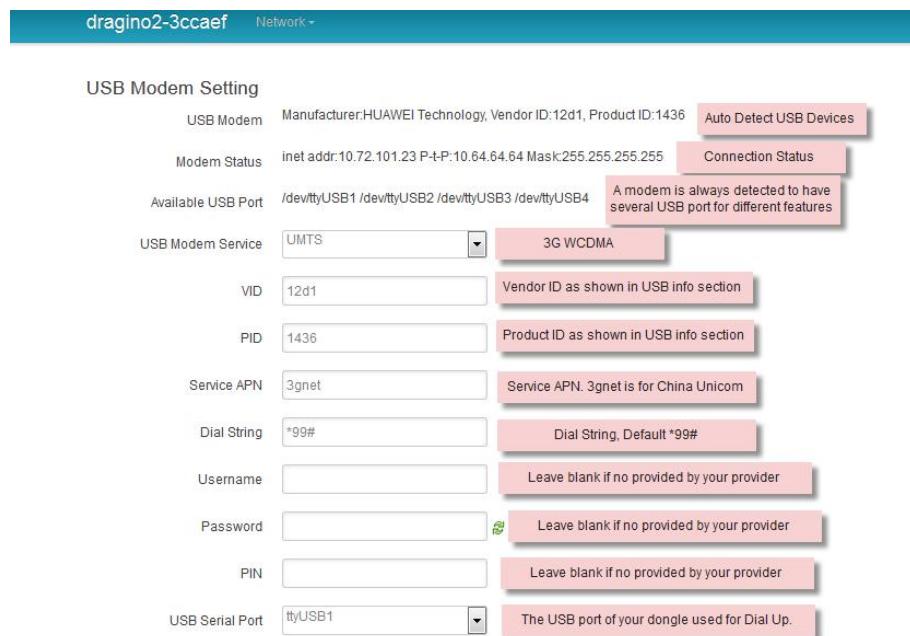
Gateway Mode

3.6 USB Dial Up Modem Set Up

Dragino USB interface can be used for GPRS/3G/4G connections. Below are some set up examples.

Note: if user use EC20/UC20 module, he just need to choose USB modem in the internet setting and reboot the device. the device will auto configure to support these modules.

WCDMA set up example:



USB Modem Setting	
USB Modem	Manufacturer:HUAWEI Technology, Vendor ID:12d1, Product ID:1436
Modem Status	inet addr:10.72.101.23 P-t-P:10.64.64.64 Mask:255.255.255.255
Available USB Port	/dev/ttyUSB1 /dev/ttyUSB2 /dev/ttyUSB3 /dev/ttyUSB4 A modem is always detected to have several USB port for different features
USB Modem Service	UMTS 3G WCDMA
VID	12d1 Vendor ID as shown in USB info section
PID	1436 Product ID as shown in USB info section
Service APN	3gnet Service APN. 3gnet is for China Unicom
Dial String	*99# Dial String, Default *99#
Username	<input type="text"/> Leave blank if no provided by your provider
Password	<input type="text"/> Leave blank if no provided by your provider
PIN	<input type="text"/> Leave blank if no provided by your provider
USB Serial Port	ttyUSB1 The USB port of your dongle used for Dial Up.

3G EV-DO/CDMA2000 Set Up Example for China Telecom:

dragino2-3ccaeef Status ▾ System ▾ Sensor ▾ Network ▾ Logout

USB Modem Setting

3G EV-DO dial up example:
 Provider: China Telecom 3G
 USB Dongle: ZTE AC582

USB Modem Manufacturer:ZTE, Vendor ID:19d2, Product ID:0152

Modem Status

Available USB Port /dev/ttyUSB0 /dev/ttyUSB1 /dev/ttyUSB2 /dev/ttyUSB3 /dev/ttyUSB4

USB Modem Service	EV-DO	<input type="button" value="Choose EV-DO"/>
VID	19d2	<input type="button" value="Input USB dongle VID"/>
PID	0152	<input type="button" value="Input USB dongle PID"/>
Service APN		
Dial String	#777	<input type="button" value="Dial String for Chinatelecom"/>
Username	ctnet@mycdma.cn	<input type="button" value="User Name"/>
Password	*****	<input type="button" value="Password"/>
PIN		
USB Serial Port	ttyUSB0	<input type="button" value="Choose USB Serial Port for 3G"/>

3.7 USB 3G/4G Ethernet Dongle

Some USB dongles are not using the dial up to connect internet. Instead, they appear as a network interface and has built-in router feature. Huawei Hilink dongles are a typical of these. When user plug such dongle into computer, it will auto connect to Internet and redirect to a web interface, when plug such dongle into the MS14, a new interface will appear (typically eth2eth2 or usb0) by running command "ifconfig -a"

User can use the Web UI to configure use these dongle for internet connection directly.

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)

173.236.176.38-dreamhost 172.31.255.254

```

collisions:0 txqueuelen:1000
RX bytes:138038 (134.8 Kib) TX bytes:490130 (478.6 Kib)
Interrupt:5

eth1      Link encap:Ethernet HWaddr A8:40:41:14:31:E6
          BROADCAST MULTICAST MTU:1500 Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 B) TX bytes:0 (0.0 B)
          Interrupt:4

eth2      Link encap:Ethernet HWaddr 58:2C:80:13:92:63
          inet addr:192.168.1.100 Bcast:192.168.1.255 Mask:255.255.255.0
          inet6 addr: fe80::5a2c:80ff:fe13:9263/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
          RX packets:331 errors:0 dropped:0 overruns:0 frame:0
          TX packets:325 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:32990 (32.2 Kib) TX bytes:26875 (26.2 Kib)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1 Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING MTU:65536 Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:0 (0.0 B) TX bytes:0 (0.0 B)

```

new interface
from USB
modem

a new interface generate by USB Ethernet modem



Small Enterprise-Campus Network

Internet Access

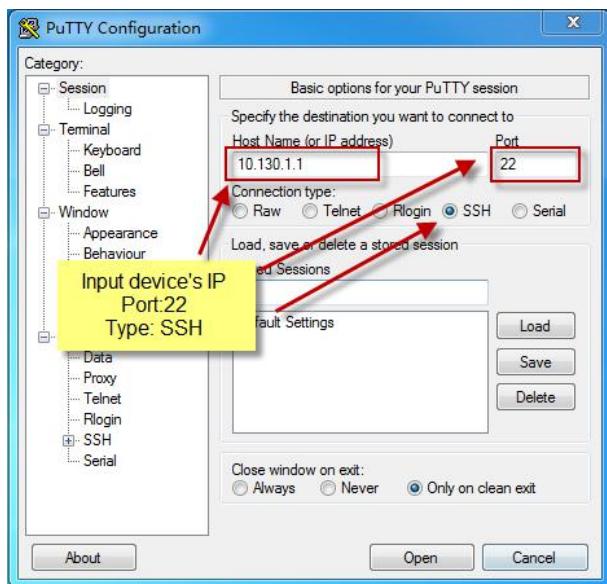
Access Internet Via	<input type="radio"/> USB Ethernet Modem	Use USB Ethernet Modem for network connection
USB Ethernet Interface	<input type="text" value="eth2"/>	input the USB Ethernet Interface

4. Linux System

The LG01 bases on OpenWrt Linux System. It is open source, and user are free to configure and modify the inside Linux settings.

4.1 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:



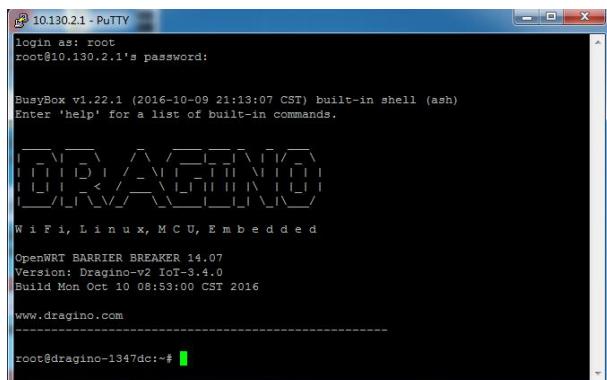
IP address: IP address of LG01

Port: 22

User Name: root

Password: dragino (default)

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



The screenshot shows the PuTTY terminal window titled '10.130.2.1 - PuTTY'. The session has been saved as 'Input device's IP' with 'Port:22' and 'Type: SSH'. The terminal displays the following text:

```

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.

[REDACTED] (busybox logo)

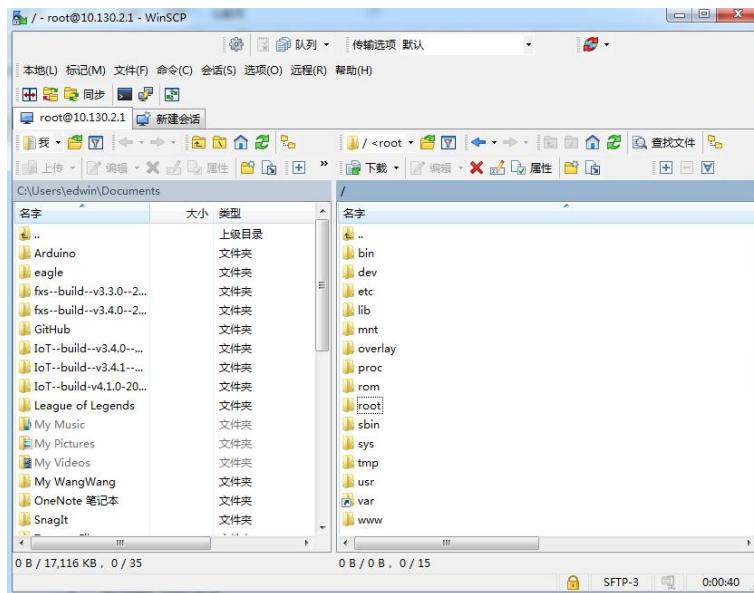
Wi-Fi, Linux, MCU, Embedded

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:~#

```

4.2 Edit and Transfer files

The LG01 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, user can use a FTP alike window to drag / drop files to the LG01 or Edit the files directly in the windows. Screenshot is as below:



4.3 File System

The LG01 has a 16MB flash and a 64MB RAM. The /var and /tmp directory are in the RAM, that means content in /tmp and /var will be erased after reboot the device. Other directories are in the flash and will keep after reboot.

The Linux system uses around 8MB ~10MB flash size which means there is not much room for user to store data in the LG01 flash. User can use an external USB flash to extend the size for storage.

4.4 Package maintain system

LG01 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 MQTT support, they can install the related packages and configure LG01 to support MQTT

Below is some examples opkg 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 mosquitto-client // install MQTT client, it will auto install
the required packages.

Installing mosquitto-client (1.3.5-1) to root...
Downloading
http://downloads.openwrt.org/barrier_breaker/14.07/ar71xx/generic/packages/packages/mosquitto-client_1.3.5
-1_ar71xx.ipk.

Installing libcares (1.10.0-1) to root...
Downloading
http://downloads.openwrt.org/barrier_breaker/14.07/ar71xx/generic/packages/packages/libcares_1.10.0-1_ar7
1xx.ipk.

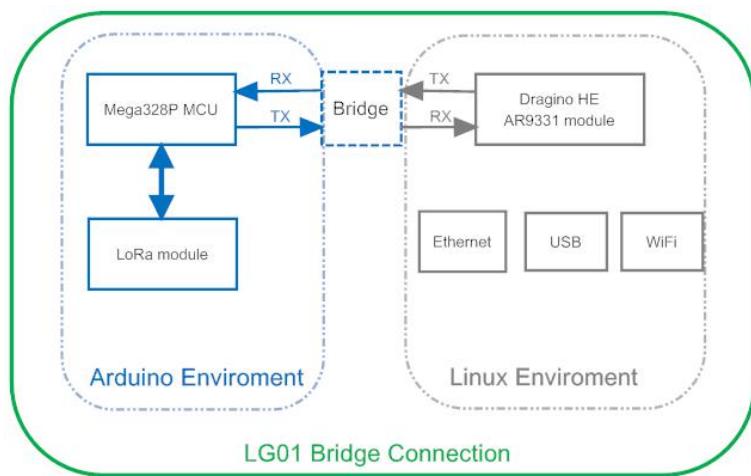
Installing libmosquitto (1.3.5-1) to root...
Downloading
http://downloads.openwrt.org/barrier_breaker/14.07/ar71xx/generic/packages/packages/libmosquitto_1.3.5-1_
ar71xx.ipk.

Configuring libcares.
Configuring libmosquitto.
Configuring mosquitto-client.
```

5. Bridge Library

The bridge library is the most important part of LG01. Bridge library defines a mechanism how the MCU talk to the CPU (ar9331). With the bridge library, the MCU can send data to CPU, get commands result from CPU or call commands in CPU.

The bridge Library use UART port to communicate between MCU and ar9331. Below is the block diagram shows the bridge connection between the Mega328P MCU and Linux.

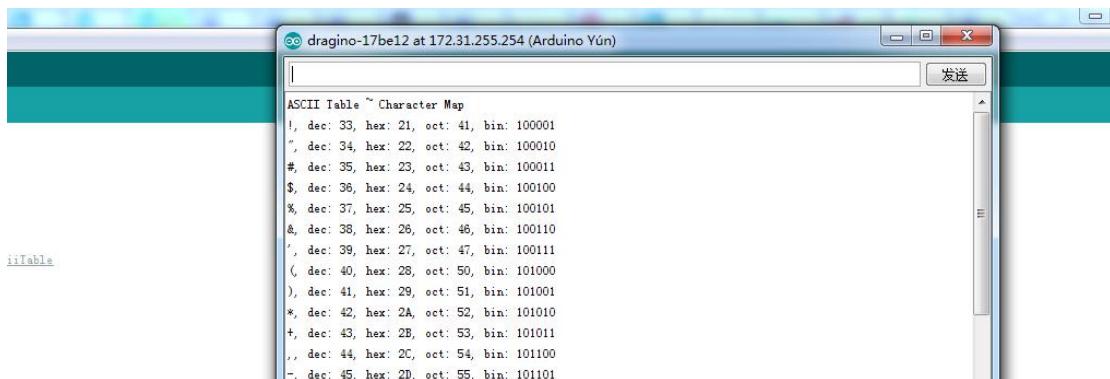


Detail instruction of how to use the bridge library can be found in [Arduino Bridge Library](#). Due to hardware difference, there are some points when we read and use the Bridge Examples from Arduino site:

- We recommend user to try the examples under [Arduino IDE --> Files --> Examples --> Dragino](#) first.
- When use the Bridge class, user need to call `Bridge.begin(115200)` in the sketch for LG01.
- In the default bridge examples from Arudino IDE, it uses Serial class to print debug info. This doesn't work in LG01. Because the [Serial Class](#) will call Mega328P's hardware serial port , it will be conflict with the Bridge Library. If user needs to print debug info, please use the console class.

5.1 The Use of Console

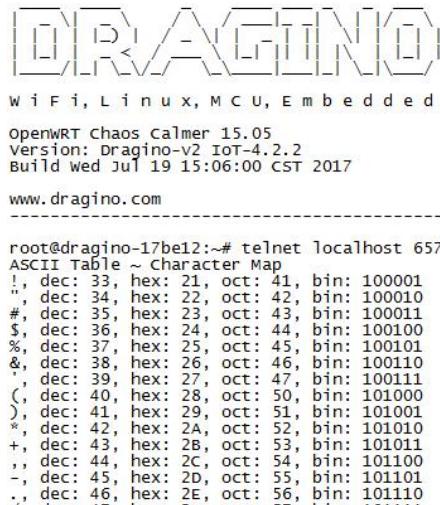
Console Class in Arduino is a good way to output debug info to computer. The **Arduino IDE --> Files --> Examples --> Dragino-->Bridge-->ConsoleRead** shows how to use this class to print data to Arduino IDE's Serial Monitor. as below



User can also don't use the Serial Monitor , instead, user can [use SSH to log in LG01](#) and run **telnet localhost 6571** to get the Console result, as below:



BusyBox v1.23.2 (2017-06-24 23:34:27 CST) built-in shell (ast)



6. Advance Management

6.1 Reset Network or Reset to Factory Default

LG01 provide ways for user to reset the device. When the Linux system is running, user can press the toggle button to reset the device. the pressing time will determine which part is to be reset.

- Pressing the toggle button, the **GLOBAL LED** will blink, release the button after 5 seconds, device will reset the network setting and reboot (GLOBAL/LAN/WAN/WiFi blink once), other settings will be kept.
- Pressing the toggle button, the **GLOBAL LED** will blink, release the button after 30 seconds, device will reset ALL the setting to factory default and reboot (GLOBAL/LAN/WAN/WiFi blink once) .

7. Upgrade Linux Firmware

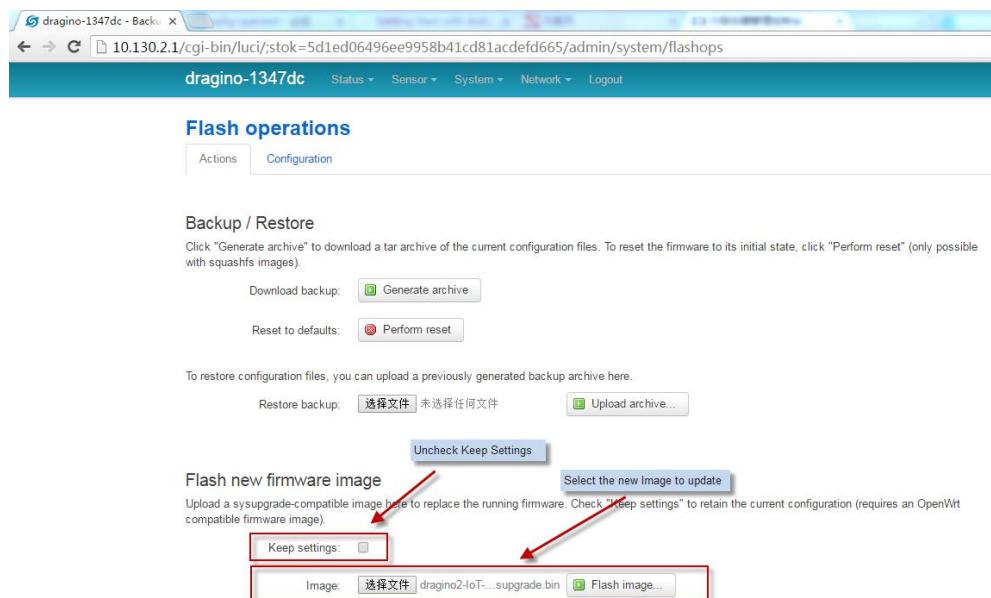
We keep improving the LG01 Linux side firmware for new features, bug fixes. The latest firmware can be found on [IoT Mesh Firmware](#), and the Change Log can be found here: [Firmware Change Log](#).

The file named as **dragino2-IoT--xxxxx-squashfs-sysupgrade.bin** is the upgrade Image. There are different method to upgrade, as below:

7.1 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.



7.2 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.

8. Upgrade Micro Controller Sketch

There are three ways for use to upgrade the sketch to the MCU use LG01.

8.1 Upgrade Sketch via Arduino IDE

We have already introduce this method in above, please see from [here](#)

8.2 Upgrade Sketch via Web UI

Upgrade Sketch via Web UI is a good way to distribute production sketch in hex format.

User can get the hex production file and below is the upgrade step:

- In LG01, Go to page **Sensor --> Flash MCU** , Select the hex file and upload it to LG01.
 - Reboot LG01, After reboot, check page **Sensor --> MicroController** , If the Sketch has defined the MCU version, you can see from this page and check if upgrade is correctly.

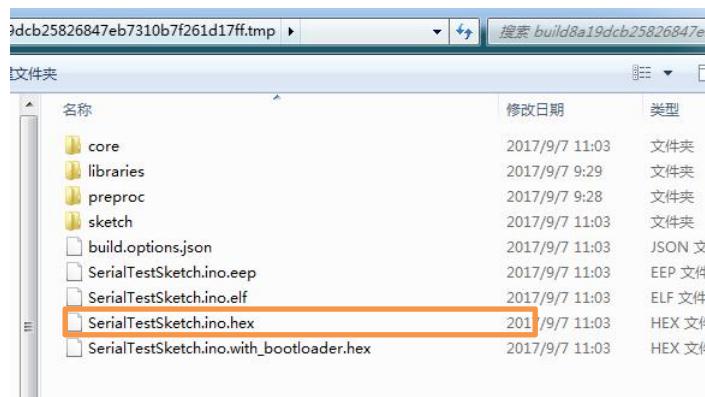
How to get the HEX file?

In the Arduino IDE, when compile the Sketch, User can see where the code is located as below:

The screenshot shows the Arduino IDE interface with the following details:

- Title Bar:** SerialTestSketch | Arduino 1.6.8
- Menu Bar:** 文件 编辑 项目 工具 帮助
- Toolbar:** Includes icons for back, forward, search, and other common functions.
- Code Editor:** Displays the `SerialTestSketch` code. The code includes a note: "This example code is in the public domain." It defines a `SoftwareSerial` object `mySerial` and sets up serial communication at 9600 bps. It then prints "Hello, world?" to the serial port.
- Terminal Window:** Shows the output of the serial communication. The text "Hello, world?" is displayed, preceded by several lines of system initialization messages.

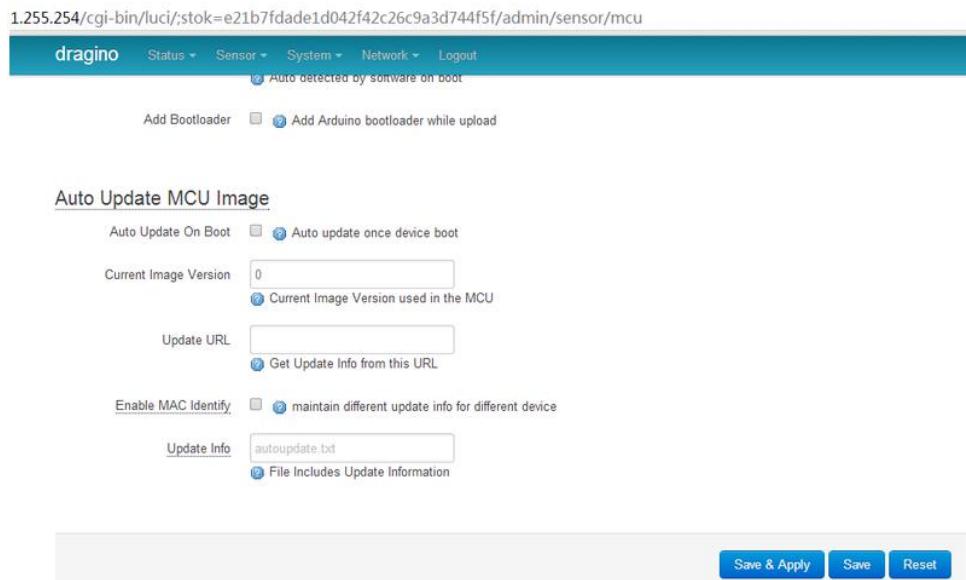
Then open the directory and get the hex file (DO NOT use the one with bootloader)



8.3 Auto update MCU

Since Firmware IoT-4.1.1, auto update sketch is supported. With this feature, the Dragino's will connect to a http/https server and get the latest sketch version and upload the sketch with this version. The purpose of this feature is to reduce the tech support cost / time for remote installation.

The feature can be configured in the page [sensors -> microcontroller](#).



1.255.254/cgi-bin/luci;/stok=e21b7fdade1d042f42c26c9a3d744f5f/admin/sensor/mcu

dragino Status Sensor System Network Logout

Auto detected by software on boot

Add Bootloader Add Arduino bootloader while upload

Auto Update MCU Image

Auto Update On Boot Auto update once device boot

Current Image Version Current Image Version used in the MCU

Update URL Get Update Info from this URL

Enable MAC Identify maintain different update info for different device

Update Info File Includes Update Information

Save & Apply Save Reset

- **Auto Update On Boot:** While this option is enabled. Device will connect to the auto server on every boot and check if there is new version of Sketch to be update. If Device find newer version on the auto update server, device will download it from the server and update the mcu with this new version.
- **Current Image Version:** Shows the current sketch version. By default it is 0. Device will update this version to the latest version number only after auto update successful.
- **Update URL:** This URL contact the update information and the sketch.hex file. Device will connect to this URL to check if there is newer version in the server.
- **Update Info:** The text file includes the update information. An example for this file can be found here: [example for update information file](#). It should include:
 - **image:** the sketch used for auto update
 - **md5sum:** md5sum for this sketch.
 - **version:** the latest version number.
- **Enable MAC Identify:** Instead of getting update information as specified in **Update Info**, The device will look for the update information from the file: [wifi_mac.txt](#). Which means, if the device has wifi mac address A840417867AF, device will download the file: \$Update_URL/A840417867AF.txt for auto update information.

Procedure for Auto Update Sketch:

Assume we have below configured:

Auto Update On Boot: checked

Update URL: <http://www.dragino.com/downloads/downloads/tmp/autoupdate/>

Update Info: update_info

Enable MAC Identify: unchecked

After reboot, the device will do auto update as below:

1. Download the update information
from http://www.dragino.com/downloads/downloads/tmp/autoupdate/update_info
2. Compare the Latest version and the version on the device
3. If server has a higher version, Device will
download <http://www.dragino.com/downloads/downloads/tmp/autoupdate/sketch.hex>
4. Do a md5sum check to verify the downloaded sketch is fine
5. Update the MCU with the newer version sketch
6. Update the version number to the latest version number

9. Example: Integrate LoRa with RESTful API

9.1 What is RESTful API?

A RESTful API is an application program interface (API) that uses HTTP requests to GET, PUT, POST and DELETE data.

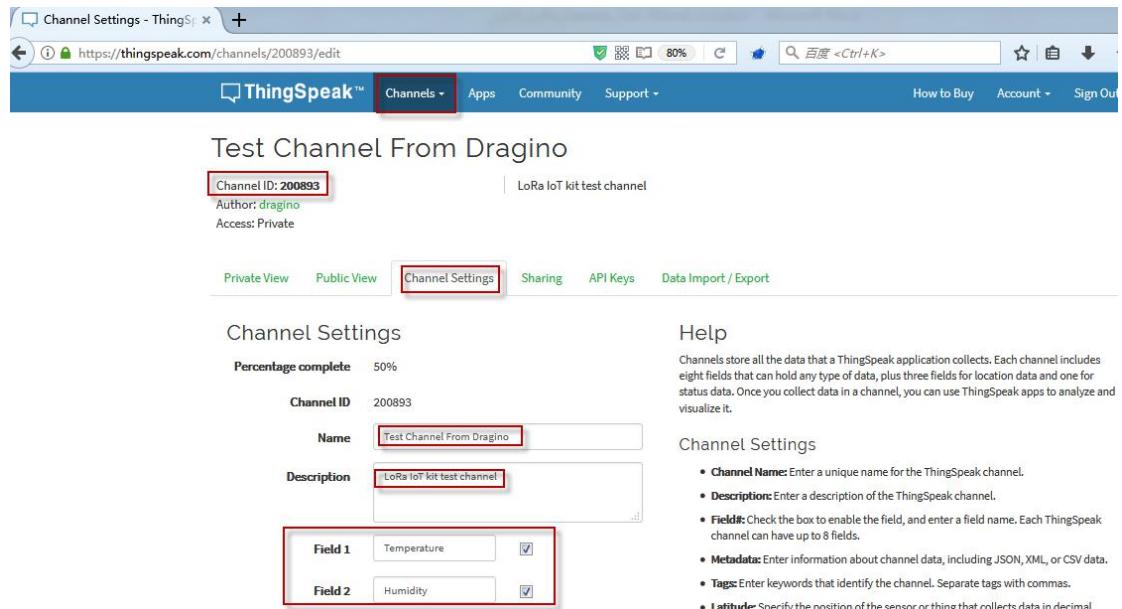
Many IoT services provide RESTful API as one of the uplink/downlink method for sensor data communication.

This example will show how to use LG01 to communicate with IoT server via RESTful API, so to achieve the goal to upload sensor data to IoT server or download commands from IoT server.

9.2 Configure IoT Server

Many servers support RESTful API, the server we use here is [ThingSpeak](#) which has an intuitive chart to show the test result for our test. The method here is general and can be used with other IoT servers for RESTful connection as well.

To use the server, we need to [register an account](#) on Thingspeak. Then [create a channel](#) and type the channel info. As shown below, the Channel ID is the unique ID to store our data in ThingSpeak.



The screenshot shows the 'Channel Settings - ThingSpeak' page for a channel named 'Test Channel From Dragino'. The channel ID is 200893, and it is a LoRa IoT kit test channel with private access. The 'Channel Settings' tab is selected. The channel is 50% complete. It has two fields: 'Field 1' (Temperature) and 'Field 2' (Humidity), both checked. A red box highlights the 'Channel Settings' tab in the navigation bar and the field settings below.

Channel Settings

Percentage complete 50%

Channel ID 200893

Name Test Channel From Dragino

Description LoRa IoT kit test channel

Field 1 Temperature

Field 2 Humidity

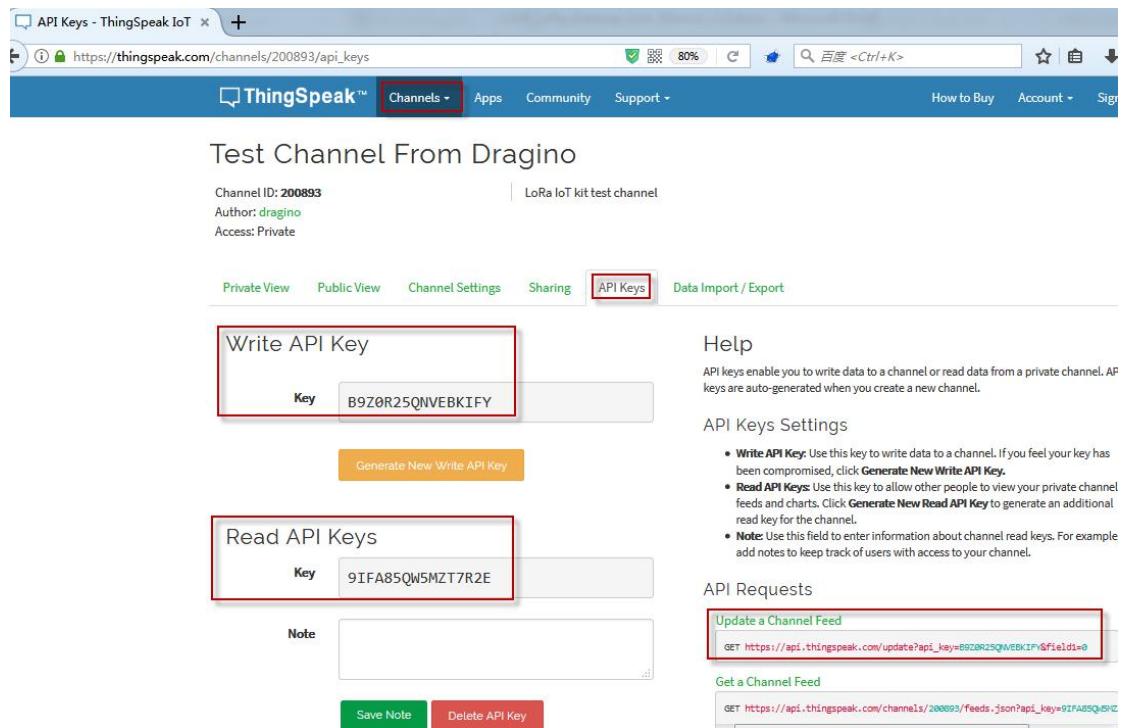
Help

Channels store all the data that a ThingSpeak application collects. Each channel includes eight fields that can hold any type of data, plus three fields for location data and one for status data. Once you collect data in a channel, you can use ThingSpeak apps to analyze and visualize it.

Channel Settings

- **Channel Name:** Enter a unique name for the ThingSpeak channel.
- **Description:** Enter a description of the ThingSpeak channel.
- **Fields:** Check the box to enable the field, and enter a field name. Each ThingSpeak channel can have up to 8 fields.
- **Metadata:** Enter information about channel data, including JSON, XML, or CSV data.
- **Tags:** Enter keywords that identify the channel. Separate tags with commas.
- **Latitude:** Specify the position of the sensor or thing that collects data in decimal degrees.

To let the LG01 communicate with the channel, we also need the API keys and API method. In ThingSpeak, we can find API keys and API call method in API Keys page:



9.3 Step by Step Uplink Test

In this section, we will try to program LG01 to uplink data to ThingSpeak. The data flow in this example is as below:

LoRa to RESTful Integration:
Uplink Data Flow to ThingSpeak



Data Flow:

- ①: LoRa end node get data from sensor and send out via LoRa wireless protocol
- ②: LoRa/MCU part in LG01 get the sensor data from LoRa wireless, and pass the data to Linux side
- ③: Linux part in LG01 send the sensor data to IoT server in RESTful API format.

We have already tried ① and ② in the above simple LoRa example. Now we will try the step ③ first, after it work as expect, we will integrate these three steps together for a complete uplink example.

9.3.1 Try RESTful API call in web

We can see the API from ThingSpeak page as below:

The screenshot shows the ThingSpeak API Keys page for Channel ID 200893. The page has tabs for Private View, Public View, Channel Settings, Sharing, API Keys (which is selected), and Data Import / Export. Under the API Keys tab, there are two main sections: "Write API Key" and "Read API Keys". In the "Write API Key" section, the key is B9Z0R25QNVEBKIFY. In the "Read API Keys" section, the key is 9IFAB5QW5MZT7R2E. To the right, there is a "Help" section explaining API keys, an "API Keys Settings" section with notes about write and read keys, and an "API Requests" section showing examples for "Update a Channel Feed" and "Get a Channel Feed".

As above, the API call to update a channel feed is as below:

```
GET https://api.thingspeak.com/update?api_key=B9Z0R25QNVEBKIFY&field1=0
```

In above API call, there are three variables:

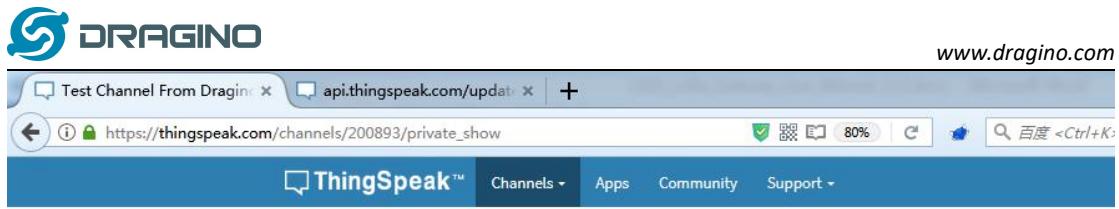
- ✓ **api_key**: Define on which channel you will upload the data.
- ✓ **Field**: Each channel have max 8 fields, Field1 , Field2 define which exactly field to be updated
- ✓ **Value**: Field1=0, means update the Field1 with value 0.

To test this API call, we can simply put the URL in web browser and test. For example, below shows the action to set the field1 data to 34, and we can see the result immediately in the private view.

The screenshot shows a web browser window with the URL https://api.thingspeak.com/update?api_key=B9Z0R25QNVEBKIFY&field1=34. The browser status bar also displays the number 1346.

1346

Result is as below:



Test Channel From Dragino

Channel ID: **200893**

Author: **dragino**

Access: Private

LoRa IoT kit test channel

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

Add Visualizations

Data Export

MAT

Channel Stats

Created: [11 months ago](#)
Updated: [about an hour ago](#)
Last entry: [about an hour ago](#)
Entries: 1346

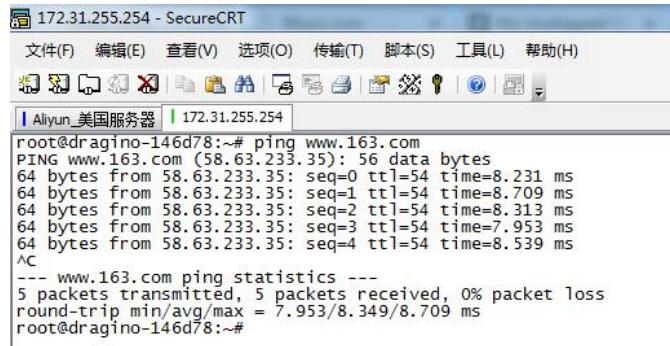


So we have tested that the RESTful API calls works as the document they provide, next step we will try in LG01 to use this API call.

Note: RESTful API is a design style, not a standard. Different servers may have different RESTful call format. Some format can't be executed in web browser. Developer should follow the documents provided by the IoT server. POSTMAN is very powerful tool for debug the RESTful API.

9.3.2 Try RESTful API call with LG01 Linux command

First, we need to make sure the LG01 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)
阿里云_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
...
--- 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:~#
```

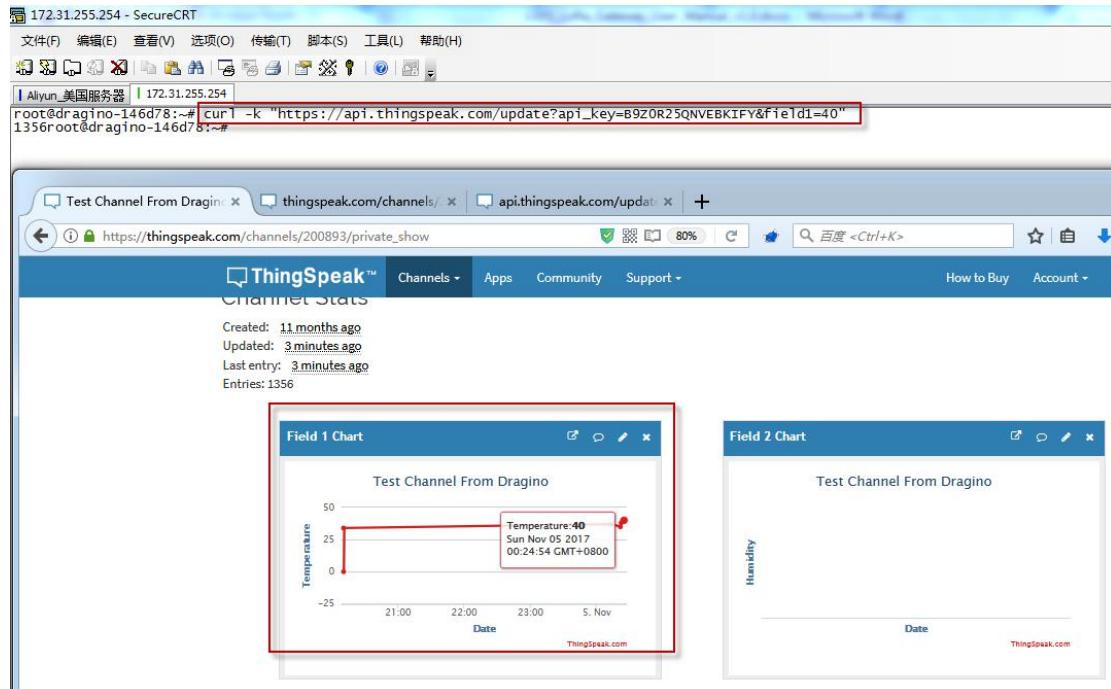
LG01 has built-in Linux tool curl. It is a very powerful tool for http communication. We can use this tool to handle RESTful API call in LG01.

The command to update a feed is as below:

```
curl -k "https://api.thingspeak.com/update?api_key=B9Z0R25QNVEBKIFY&field1=40"
```

(Make sure the "" is included , otherwise you will get null value in ThingSpeak)

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)
阿里云_美国服务器 | 172.31.255.254
root@dragino-146d78:~# curl -k "https://api.thingspeak.com/update?api_key=B9Z0R25QNVEBKIFY&field1=40"
1356root@dragino-146d78:~#

Test Channel From Dragino | thingspeak.com/channels/ | api.thingspeak.com/update | +
https://thingspeak.com/channels/200893/private_show 80% C 百度 <Ctrl+K> ☆ | 自 下
ThingSpeak™ Channels Apps Community Support How to Buy Account S

Channel Stats
Created: 11 months ago
Updated: 3 minutes ago
Last entry: 3 minutes ago
Entries: 1356

Field 1 Chart
Test Channel From Dragino
Temperature: 40
Sun Nov 05 2017
00:24:54 GMT+0800

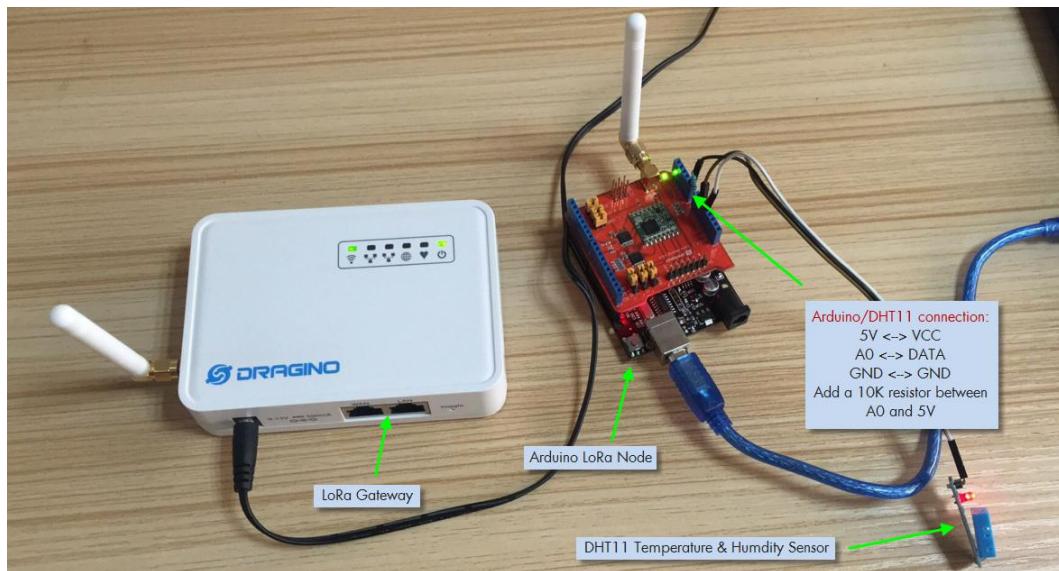
Field 2 Chart
Test Channel From Dragino

So we success to use LG01 to uplink data to ThingSpeak, the curl command is executed in the Linux side, finally, we will have to call **curl** command with sensor data variable in Arduino side. This is through the **process class** in Arduino and we will show it in the final sketch.

9.3.3 Integrate LoRa, Bridge and Curl

Here we will provide a complete example for our system. The hardware set up is as below:

- ✓ **LoRa End Node:** LoRa Shied + UNO + DHT11 Temperature/Humidity sensor. The LoRa End node keeps getting temperature and humidity from the sensor and sends out via LoRa periodically.
- ✓ **LoRa Gateway LG01:** Listening on the LoRa wireless channel, while there is new LoRa packet arrives, parse it and send out to IoT Server.



The source code used here are:

- LoRa Shield + UNO : [Client Source Code](#)
- LG01 LoRa Gateway: [Gateway Source Code](#)

In the source code, there is well explaining about the functions and the code. Below is the result for the testing. We can see the

The image shows two terminal windows side-by-side. The left window is titled "COM9" and the right window is titled "dragino-146d78 at 172.31.255.254 (Arduino Yun)". Both windows show the same communication logs, indicating the exchange of LoRa packets between the two nodes.

Debug window for LoRa End Node. Shows the data to be sent.

```

LoRa End Node Example --
DHT11 Temperature and Humidity Sensor

LoRa End Node ID: 111
#####
count=1 #####
Current humidity = 62.0% temperature = 25.0C
Data to be sent (without CRC): 1 1 1 3E 0 19 0 C8 60
Data to be sent (with CRC): 1 1 1 3E 0 19 0 C8 60
Got Reply from Gateway: Server ACK
#####
count=2 #####
Current humidity = 62.0% temperature = 25.0C
Data to be sent (without CRC): 1 1 1 3E 0 19 0 C8 60
Data to be sent (with CRC): 1 1 1 3E 0 19 0 C8 60
Got Reply from Gateway: Server ACK

```

Debug window for LoRa Gateway Shows the data get from LoRa End Node.

```

Get LoRa Packet: 1 1 1 3E 0 19 0 C8 60
Get Temperature:25.0
Get Humidity:62.0
Call Linux Command to Send Data
Feedback from Linux: 1490
Call Finished
#####

Get LoRa Packet: 1 1 1 3E 0 19 0 C8 60
Get Temperature:25.0
Get Humidity:62.0
Call Linux Command to Send Data
Feedback from Linux: 1491
Call Finished
#####

Get LoRa Packet: 1 1 1 3E 0 19 0 C8 60
Get Temperature:25.0
Get Humidity:62.0
Call Linux Command to Send Data
Feedback from Linux: 1492
Call Finished
#####

Get LoRa Packet: 1 1 1 3E 0 19 0 C8 60
Get Temperature:25.0
Get Humidity:62.0
Call Linux Command to Send Data
Feedback from Linux: 1493
Call Finished
#####

```

9.4 Step by Step Downlink Test

In this section, we will try to program LG01 to fetch download data from ThingSpeak, then broadcast this data to local LoRa network. The end node will get this message and check if they need to do something.

LoRa to RESTful Integration:
Downlink Data Flow , from ThingSpeak to Sensor



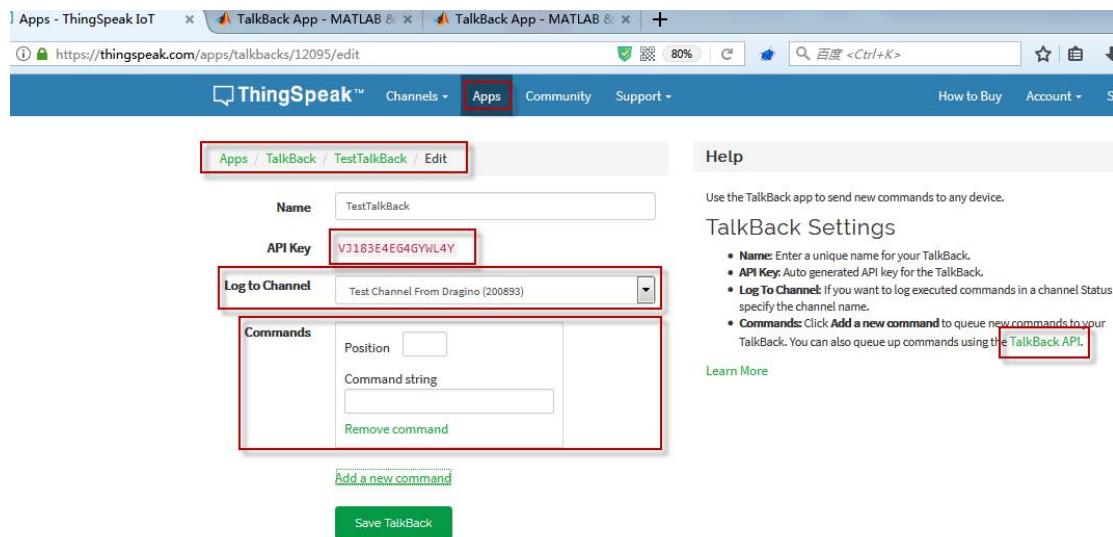
Data Flow:

- ①: LoRa MCU part send a request to Linux side, ask the Linux side to check if there is command from IoT Server
- ②: Linux send this request to server via RESTful call
- ③: If there is new command, server send a new command to Linux
- ④: Linux pass this command to MCU/LoRa.
- ⑤: LG01 MCU part broadcast this command to its LoRa network. The LoRa end node will get this message and check if they should execute it.

Similar with Uplink Example, we will first try to do it in PC, and then do it in Linux side, and finally integrate it with LoRa.

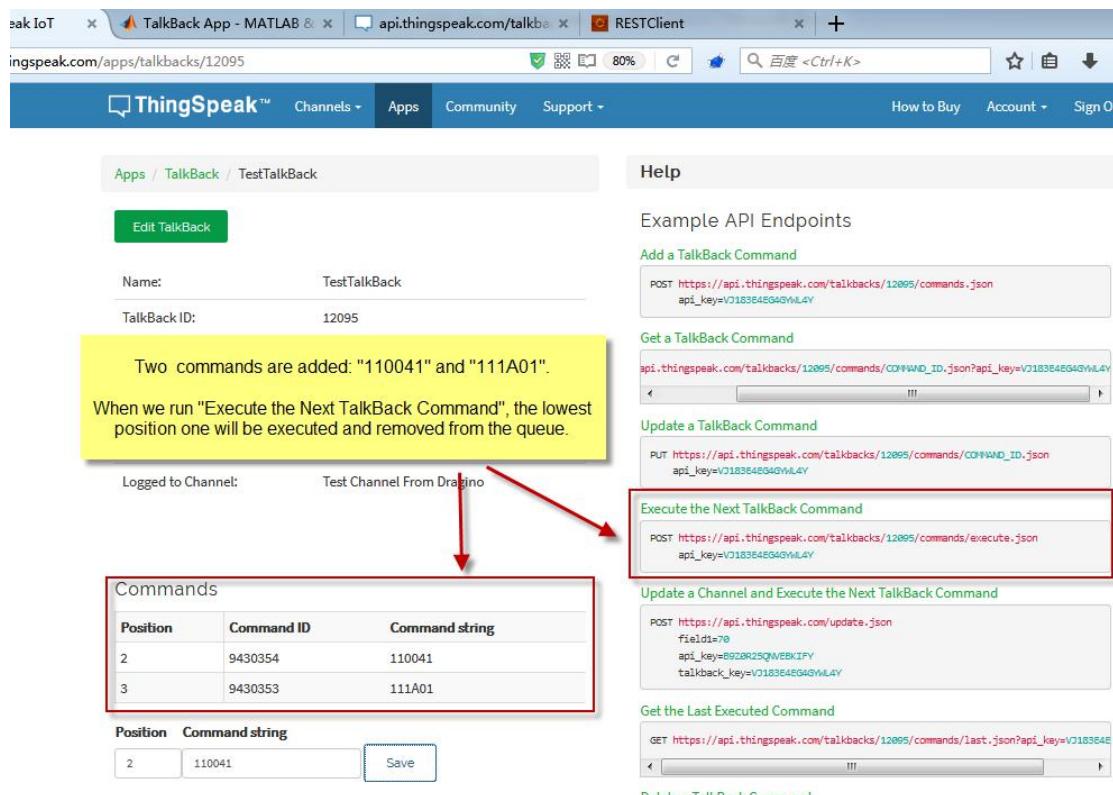
9.4.1 Create Talkback command and try RESTful API call in web

To do the downlink test we need to first create a talkback command in ThingSpeak. As below, From this page, we can get the talkback API key and set the command to be sent to the LoRa End Device.



The screenshot shows the 'Edit TalkBack' page for a 'TestTalkBack' application. The 'Name' field is 'TestTalkBack'. The 'API Key' field is 'VJ183E4EG4GYWL4Y'. The 'Log to Channel' dropdown is set to 'Test Channel From Dragino (200893)'. The 'Commands' section contains two entries: 'Position' and 'Command string'. A green 'Add a new command' button is present. To the right, a 'TalkBack Settings' sidebar explains how to use the TalkBack app to send commands to devices.

Then we add talkback command and click **Save TalkBack**. The API for how to call this will appear in the right, as below:



The screenshot shows the 'Edit TalkBack' page for a 'TestTalkBack' application after saving. It displays a message: 'Two commands are added: "110041" and "111A01".' A red arrow points from this message to a table of commands. Another red arrow points from the table to the 'Execute the Next TalkBack Command' API endpoint on the right. The table shows two commands: Position 2 (Command ID 9430354, Command string 110041) and Position 3 (Command ID 9430353, Command string 111A01). The 'Logged to Channel' dropdown is 'Test Channel From Dragino'.

Example API Endpoints

- Add a TalkBack Command**: POST <https://api.thingspeak.com/talkbacks/12095/commands.json> api_key=VJ183E4EG4GYWL4Y
- Get a TalkBack Command**: GET https://api.thingspeak.com/talkbacks/12095/commands/COMMAND_ID.json api_key=VJ183E4EG4GYWL4Y
- Update a TalkBack Command**: PUT https://api.thingspeak.com/talkbacks/12095/commands/COMMAND_ID.json api_key=VJ183E4EG4GYWL4Y
- Execute the Next TalkBack Command**: POST <https://api.thingspeak.com/talkbacks/12095/commands/execute.json> api_key=VJ183E4EG4GYWL4Y
- Update a Channel and Execute the Next TalkBack Command**: POST <https://api.thingspeak.com/update.json> field1=70 api_key=BZ2R2SQNEWBKIFY talkback_key=VJ183E4EG4GYWL4Y
- Get the Last Executed Command**: GET <https://api.thingspeak.com/talkbacks/12095/commands/last.json> api_key=VJ183E4EG4GYWL4Y
- Delete a TalkBack Command**: DELETE https://api.thingspeak.com/talkbacks/12095/commands/COMMAND_ID.json api_key=VJ183E4EG4GYWL4Y

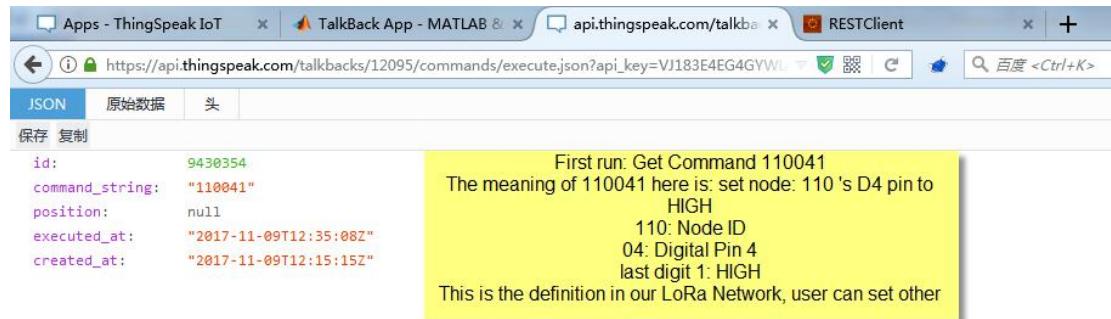
We can run the talkback API in browser to see if we can get the talkback commands:

The API we are going to use is:

https://api.thingspeak.com/talkbacks/12095/commands/execute.json?api_key=VJ183E4EG4GYWL4Y

(User should replace the 12095 and api_key with what they have in their command.)

Below are the results:

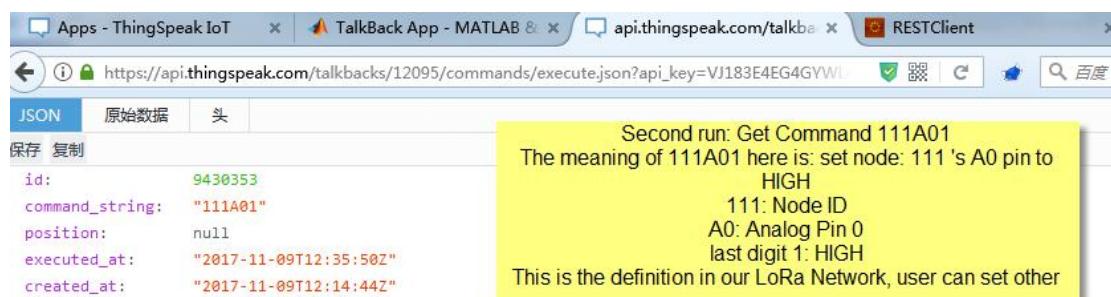


The screenshot shows a browser window with three tabs: "Apps - ThingSpeak IoT", "TalkBack App - MATLAB", and "api.thingspeak.com/talkbacks/12095/commands/execute.json?api_key=VJ183E4EG4GYWL4Y". The third tab is active, displaying a JSON response. The JSON object has the following fields:

```
id: 9430354
command_string: "110041"
position: null
executed_at: "2017-11-09T12:35:08Z"
created_at: "2017-11-09T12:15:15Z"
```

A yellow callout box highlights the command_string field and its meaning:

First run: Get Command 110041
The meaning of 110041 here is: set node: 110's D4 pin to HIGH
110: Node ID
04: Digital Pin 4
last digit 1: HIGH
This is the definition in our LoRa Network, user can set other



The screenshot shows a browser window with three tabs: "Apps - ThingSpeak IoT", "TalkBack App - MATLAB", and "api.thingspeak.com/talkbacks/12095/commands/execute.json?api_key=VJ183E4EG4GYWL4Y". The third tab is active, displaying a JSON response. The JSON object has the following fields:

```
id: 9430353
command_string: "111A01"
position: null
executed_at: "2017-11-09T12:35:50Z"
created_at: "2017-11-09T12:14:44Z"
```

A yellow callout box highlights the command_string field and its meaning:

Second run: Get Command 111A01
The meaning of 111A01 here is: set node: 111's A0 pin to HIGH
111: Node ID
A0: Analog Pin 0
last digit 1: HIGH
This is the definition in our LoRa Network, user can set other



The screenshot shows a browser window with three tabs: "Apps - ThingSpeak IoT", "TalkBack App - MATLAB", and "api.thingspeak.com/talkbacks/12095/commands/execute.json?api_key=VJ183E4EG4GYWL4Y". The third tab is active, displaying a JSON response. The JSON object has the following fields:

```
id: null
command_string: null
position: null
executed_at: null
created_at: null
```

A yellow callout box highlights the message:

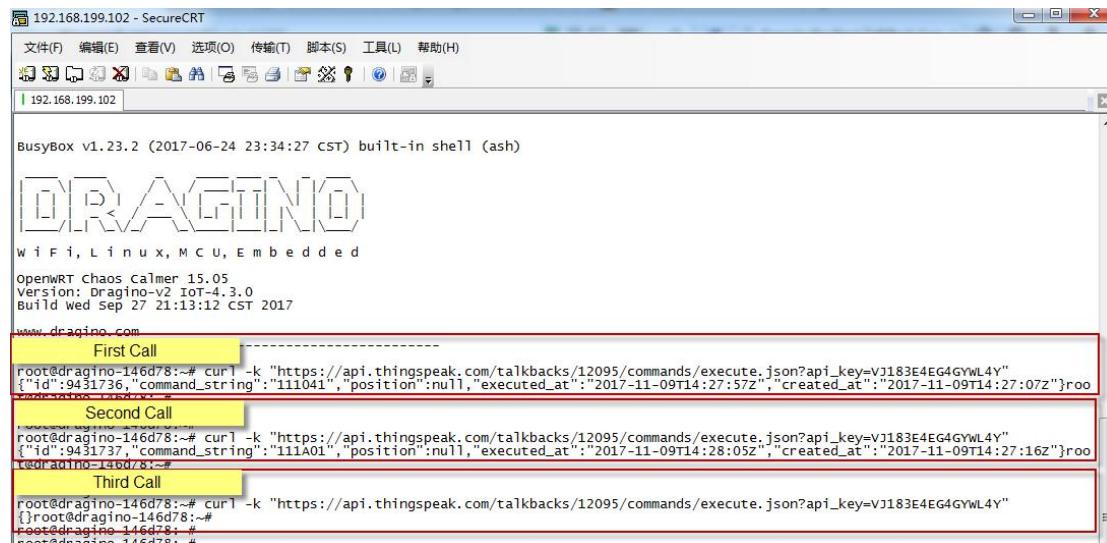
Third run: no command, get nothing.

9.4.2 Try RESTful API call with LG01 Linux command

The command to be used is **Execute the Next TalkBack Command**, with curl, it is

```
curl -k "https://api.thingspeak.com/update?api_key=B9Z0R25QNVEBKIFY&field1=40"
```

Below is the output for the Linux command test:



The screenshot shows a SecureCRT session connected to 192.168.199.102. The terminal window displays the following text:

```
BusyBox v1.23.2 (2017-06-24 23:34:27 CST) built-in shell (ash)

[DRAGINO]
Wi-Fi, Linux, MCU, Embedded
OpenWRT Chaos Calmer 15.05
Version: Dragino-v2 IoT-4.3.0
Build wed Sep 27 21:13:12 CST 2017
www.dragino.com

First Call
root@dragino-146d78:~# curl -k "https://api.thingspeak.com/talkbacks/12095/commands/execute.json?api_key=vJ183E4EG4GYwL4Y"
{"id":9431736,"command_string":"111041","position":null,"executed_at":"2017-11-09T14:27:57Z","created_at":"2017-11-09T14:27:07Z"}root

Second Call
root@dragino-146d78:~# curl -k "https://api.thingspeak.com/talkbacks/12095/commands/execute.json?api_key=vJ183E4EG4GYwL4Y"
{"id":9431737,"command_string":"111A01","position":null,"executed_at":"2017-11-09T14:28:05Z","created_at":"2017-11-09T14:27:16Z"}root

Third Call
root@dragino-146d78:~# curl -k "https://api.thingspeak.com/talkbacks/12095/commands/execute.json?api_key=vJ183E4EG4GYwL4Y"
{}root@dragino-146d78:~#
root@dragino-146d78:~#
```

The output is divided into three sections by red boxes, labeled "First Call", "Second Call", and "Third Call". Each section shows a curl command being run and its corresponding JSON response.

So we success to use LG01 to get commands (downlink data) from ThingSpeak, the curl command is executed in the Linux side, finally, we will have to call **curl** command in Arduino side and fetch the output , parse the command out of the curl output and send out the command to local LoRa network. This is through the **process class** in Arduino and we will show it in the final sketch.

9.4.3 Integrate LoRa, Bridge and Curl

Here we will provide a complete example for our system. The hardware set up is as below:

- ✓ **LoRa Gateway LG01:** Periodically check if there new commands in IoT Server. If there is new command, broadcast the command string to its LoRa network..
- ✓ **LoRa End Node:** LoRa Shied + UNO, listening if there are commands in LoRa network. If there is new command and the node ID match. It will parse and execute the incoming command.



Above is the set up for this test, the LoRa End Node here is with ID 111, and with relay on D4 pin, so , and it will look for command: 111041 or 111040 to switch on/off the relay.

The source code used here are:

- LoRa Shield + UNO : [Client Source Code](#)
- LG01 LoRa Gateway: [Gateway Source Code](#)

In the source code, there is well explaining about the functions and the code. Below is the result for the testing. We can see the result as below:

```

dragino-146d78 at 172.31.255.254 (Arduino Yún)
Checking Talkback from Server
Get Result: {}
Command Length: 2
No new command from server

Checking Talkback from Server
Get Result: {}
Command Length: 2
No new command from server

Checking Talkback from Server
Get Result: {}
Command Length: 2
No new command from server

Checking Talkback from Server
Get Result: {}
Command Length: 2
No new command from server

Checking Talkback from Server
Get Result: {} Gateway get messages from server and broadcast it in LoRa network
Command Length: 2
No new command from server

Checking Talkback from Server
Get Result: {'id':9488590, "command_string": "111041", "position":null, "executed_at": "2017-11-12T10:08:12Z", "command": "111041", "status": "PENDING", "node_id": 111}
Command Length: 129

Get Command String: 111041

COM15 (Arduino/Genuino Mega or Mega 2560)
===== Got Message from Gateway: 66 85 76 69 0
Node ID mismatch, ignore message

===== Got Message from Gateway: 65 110 100 32 104 101 108 108 111 32 98 97 99 107 32 116
Node ID mismatch, ignore message

===== Got Message from Gateway: 66 85 76 69 0
Node ID mismatch, ignore message

===== Got Message from Gateway: 65 110 100 32 104 101 108 108 111 32 98 97 99 107 32 116
Node ID mismatch, ignore message

===== Got Message from Gateway: 66 85 76 69 0
Node ID mismatch, ignore message

===== Got Message from Gateway: 65 110 100 32 104 101 108 108 111 32 98 97 99 107 32 116
Node ID mismatch, ignore message

===== Got Message from Gateway: 66 85 76 69 0
Node ID mismatch, ignore message

===== LoRa Node Get the message and execute 108 108 111 32 98 97 99 107 32 116
the related command.

===== Got Message from Gateway: 49 49 49 48 52 49 0
Node ID match, this message is for us
The pin to be controlled is:4
Set pin to HIGH.

===== Got Message from Gateway: 65 110 100 32 104 101 108 108 111 32 98 97 99 107 32 116
Node ID mismatch, ignore message

```

10. Example: Integrate LoRa with MQTT API

10.1 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 to publish data or subscribe to a channel.

This example will show how to use LG01 to connect to the IoT Server via MQTT.

The server we use here is [ThingSpeak](#). They have the [MQTT API documented here](#). We have already used the ThingSpeak as example in RESTful so we ignore the step of 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.

10.2 Step by Step Uplink Test

In this section, we will try to program LG01 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
- ②: LoRa/MCU part in LG01 get the sensor data from LoRa wireless, and pass the data to Linux side
- ③: Linux part in LG01 send the sensor data to IoT server via MQTT Publish.

We have already tried ① and ② in the above simple LoRa example. Now we will try the step ③ first, after it works as expected, we will integrate these three steps together for a complete uplink example.

10.2.1 Simulate MQTT Publish via Desktop MQTT tool

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

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

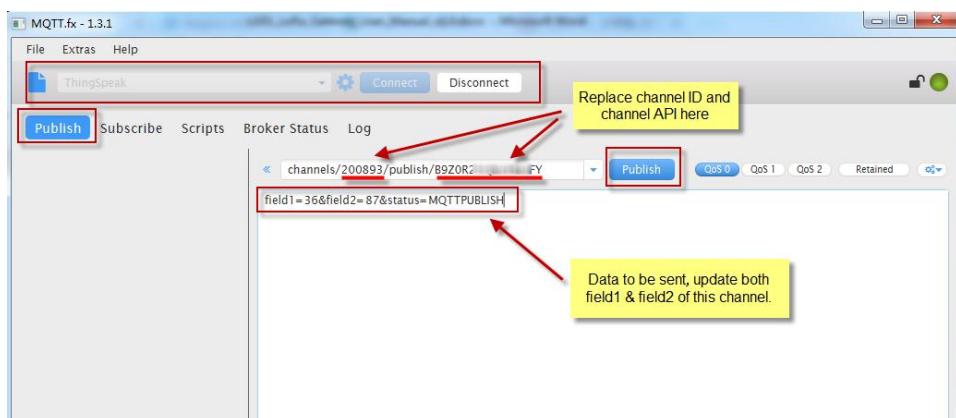
Broker Address: mqtt.thingspeak.com

Broker Port: 1883

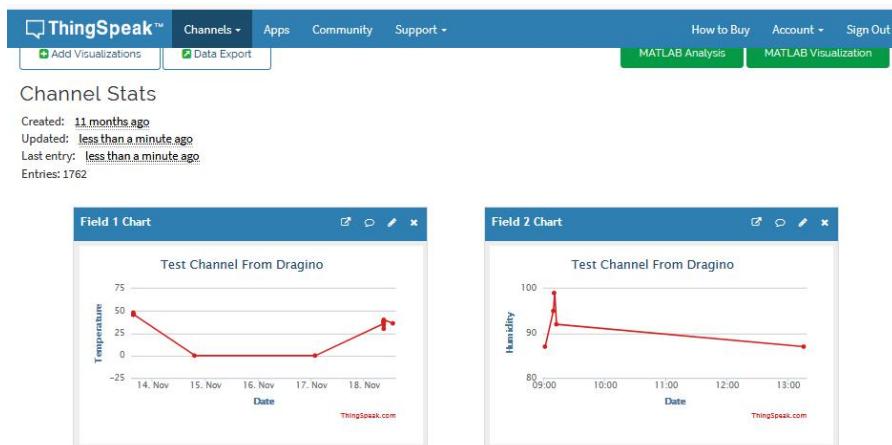
Client ID: User Defined.



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



And we can see the update in the channel:



10.2.2 Try MQTT Publish with LG01 Linux command

First, we need to make sure the LG01 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)

阿里云_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:~#
```

LG01 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 QZXTxxxxxxO2J -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:

A screenshot of the SecureCRT application window. The title bar reads "172.31.255.254 - SecureCRT". The menu bar includes "文件(F)", "编辑(E)", "查看(V)", "选项(O)", "传输(T)", "脚本(S)", "工具(L)", and "帮助(H)". Below the menu is a toolbar with icons for copy, paste, find, and others. The main pane shows a terminal session with the following text:

```
172.31.255.254
root@dragino-146d78:~# mosquitto_pub -h mqtt.thingspeak.com -p 1883 -u dragino -P Q
field1=34&field2=89&status=MQTT PUBLISH"
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 LG01 to uplink data to ThingSpeak, the **mosquitto_pub** command is executed in the Linux side, finally, we will have to call **mosquitto_pub** command with sensor data variable in Arduino side. This is through the **process class** in Arduino and we will show it in the final sketch.

For LG01 new version [4.3.4](#). You just need to configure LG01 Web settings then can realize the function of MQTT.

Select IoT Server
Select the IoT Server type to connect

Select IoT Server

IoT Server	MQTT Server		Select MQTT
Log Debug Info	Level 1		
<input checked="" type="checkbox"/> Show Log in System Log			

Save & Apply **Save** **Reset**

dragino-181b01 Status ▾ Sensor ▾ System ▾ Network ▾ Logout

MQTT Server Settings
Configuration to communicate with MQTT server

Configure MQTT Server

Select Server	ThingSpeak MQTT	Select ThingSpeak Server
User Name [-u]	cheneywang	
Password [-P]	UZ4NGHKJMKS9WR5E	MQTT API Keys
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
Users can fill in arbitrarily	465952	ISUQYWBWESINZE60
12345	The channel ID from IOT Server	 

Login SSH and input the command.

```
This version does not include PKI and PKCS #11 functionality.

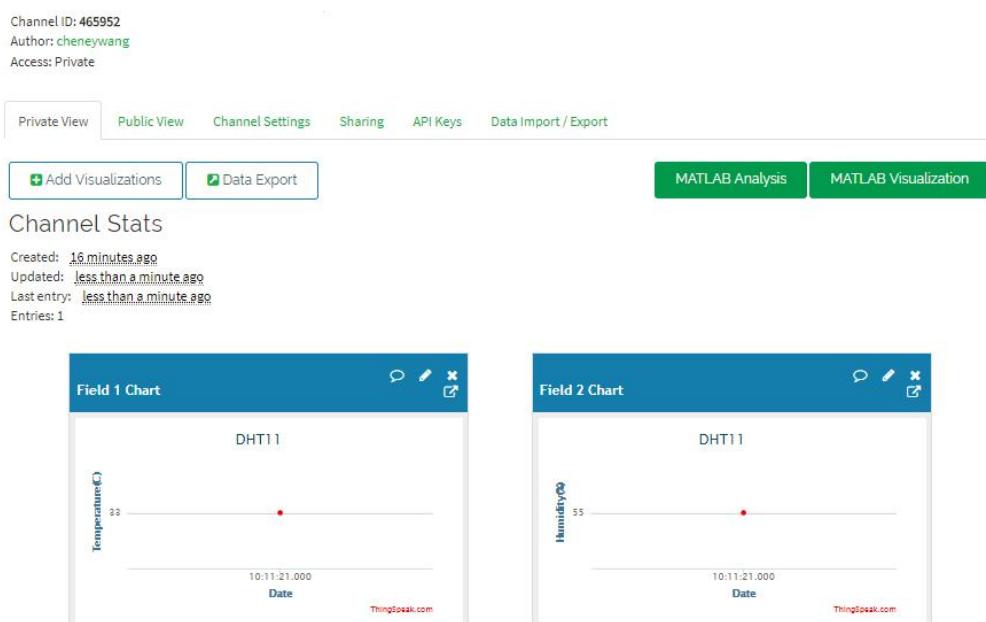
BusyBox v1.23.2 (2017-12-21 22:54:56 CST) built-in shell (ash)

[DRAGINO]
W i F i, L i n u x, M C U, E m b e d d e d
OpenWRT Chaos Calmer 15.05
Version: Dragino-v2 IoT-4.3.3
Build Mon Mar 26 15:13:47 CST 2018
www.dragino.com
-----
root@dragino-181b01:~# store_data 12345 "field1=33&field2=55"
root@dragino-181b01:~# [REDACTED]
Connected to 10.130.2.102      SSH2 - aes128-cbc - hmac-md5 | 80x24
```

The command to update a feed is as below:

store_data 12345 "field1=33&field2=55"

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

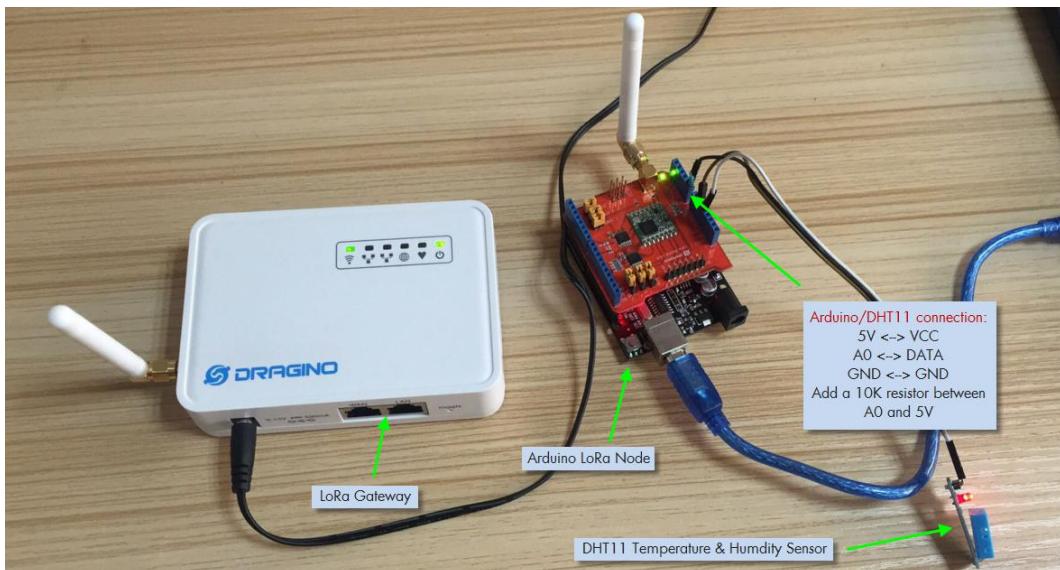


After running this command, we can see the data are updated to ThingSpeak, which has same result as what we did at previous version.

10.2.3 Integrate LoRa, Bridge and mosquitto_pub

Here we will provide a complete example for our system. The hardware set up is as below:

- ✓ **LoRa End Node:** LoRa Shied + UNO + DHT11 Temperature/Humidity sensor. The LoRa End node keeps getting temperature and humidity from the sensor and sends out via LoRa periodically.
- ✓ **LoRa Gateway LG01:** Listening on the LoRa wireless channel, while there is new LoRa packet arrives, parse it and send out to IoT Server.



The source code used here are:

- LoRa Shield + UNO : [Client Source Code\(4.3.3 Version\)](#)
- LG01 LoRa Gateway: [Gateway Source Code\(4.3.3 Version\)](#)

This version of Sketch implements these features:

1. Read the LG01 configuration information from Linux.
2. Receive the LoRa node data and store the data.
3. Send reply after then receive LoRa node data.
4. Sketch will write active content periodically.(Watch dog feature)

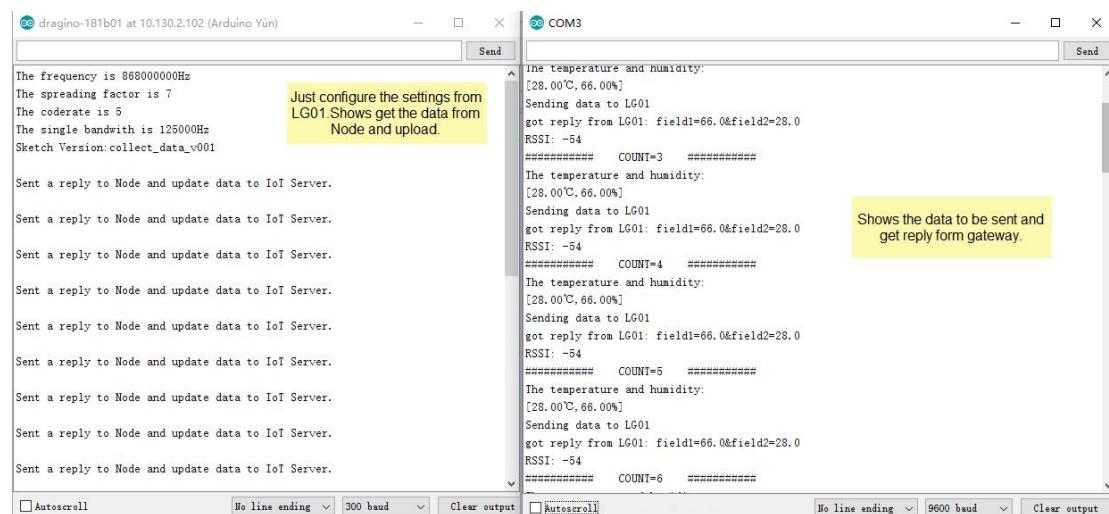
Configure from LG01:

Radio Settings

Radio settings requires MCU side sketch support

TX Frequency	<input type="text" value="9 digits Frequency, etc:868100000"/> ⓘ Gateway's LoRa TX Frequency
RX Frequency	<input type="text" value="868000000"/> ⓘ Gateway's LoRa RX Frequency
Encryption Key	<input type="text" value="Encryption Key"/>
Spreading Factor	SF7 ▾
Transmit Spreading Factor	SF9 ▾
Coding Rate	4/5 ▾
Signal Bandwidth	125 kHz ▾
Preamble Length	8 ⓘ Length range: 6 ~ 65536

In the source code, there is well explaining about the functions and the code. Below is the result for the testing. We can see the



The screenshot shows two serial monitor windows. The left window is titled "dragino-181b01 at 10.130.2.102 (Arduino Yún)" and the right window is titled "COM3". Both windows show the same communication logs:

```

The frequency is 868000000Hz
The spreading factor is 7
The coderate is 5
The single bandwidth is 125000Hz
Sketch Version: collect_data_v001

Sent a reply to Node and update data to IoT Server.
Sent a reply to Node and update data to IoT Server.
Sent a reply to Node and update data to IoT Server.
Sent a reply to Node and update data to IoT Server.
Sent a reply to Node and update data to IoT Server.
Sent a reply to Node and update data to IoT Server.
Sent a reply to Node and update data to IoT Server.
Sent a reply to Node and update data to IoT Server.
Sent a reply to Node and update data to IoT Server.

The temperature and humidity:
[28.00°C, 66.00%]
Sending data to LG01
got reply from LG01: field1=66.0&field2=28.0
RSSI: -54
##### COUNT=3 #####
The temperature and humidity:
[28.00°C, 66.00%]
Sending data to LG01
got reply from LG01: field1=66.0&field2=28.0
RSSI: -54
##### COUNT=4 #####
The temperature and humidity:
[28.00°C, 66.00%]
Sending data to LG01
got reply from LG01: field1=66.0&field2=28.0
RSSI: -54
##### COUNT=5 #####
The temperature and humidity:
[28.00°C, 66.00%]
Sending data to LG01
got reply from LG01: field1=66.0&field2=28.0
RSSI: -54
##### COUNT=6 #####

```

Annotations highlight specific parts of the log:

- A yellow box around the first few lines of the log in the left window contains the text: "Just configure the settings from LG01. Shows get the data from Node and upload."
- A yellow box around the "COUNT" values in the right window contains the text: "Shows the data to be sent and get reply form gateway."

Sketch can write the new version of MCU.

Micro-Controller settings

Configure correct Arduino profile will let you to able upload avr program via Arduino IDE and WiFi

MCU Upload Profile

MCU Part	ATmega328P
	<input checked="" type="checkbox"/> Auto detected by software on boot
Profile	Arduino Uno w/ATmega328P ▾
	<input checked="" type="checkbox"/> Auto detected by software on boot
MCU Version	collect_data_v001
Add Bootloader	<input checked="" type="checkbox"/> <input type="checkbox"/> Add Arduino bootloader while upload
MCU Watchdog	<input type="checkbox"/> <input type="checkbox"/> Reset MCU while sketch hang up

Sketch will write active content periodically.

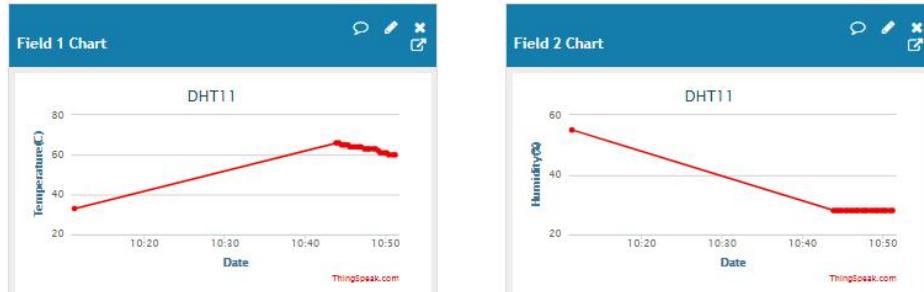
```
root@dragino-181b01:~# cd /tmp/iot/
root@dragino-181b01:/tmp/iot# cat dog
1522724284
root@dragino-181b01:/tmp/iot# cat dog
1522724289
root@dragino-181b01:/tmp/iot# cat dog
1522724291
root@dragino-181b01:/tmp/iot# cat dog
1522724292
root@dragino-181b01:/tmp/iot# cat dog
1522724296
root@dragino-181b01:/tmp/iot# [REDACTED]
Connected to 10.130.2.102      SSH2 - aes128-cbc - hmac-md5 | 80x24 | [REDACTED] [REDACTED] [REDACTED]
```

Watch dog feature

Check the result from IoT Server.

Channel Stats

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



The more : [Through MQTT to upload data](#)

10.3 Step by Step Downlink Test

Now we have two ways to receiving data for downlink.

1. Use [MQTT.fx](#)

About configure follow as 10.2.1 [MQTT_tool](#).

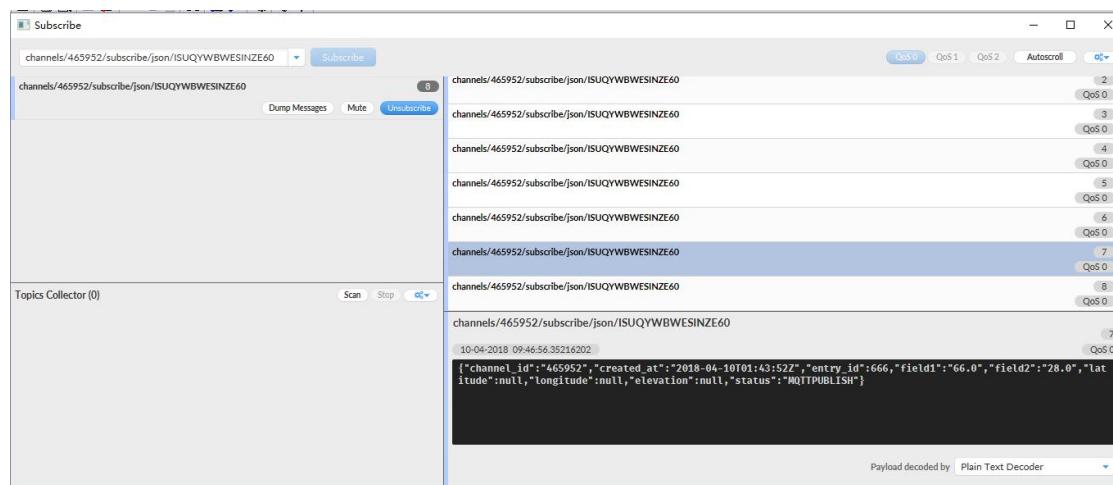
Input command:

```
channels/465952/subscribe/json/ISUQYWBWESINZE60
```

Ex)Channel ID: 465952

```
Write_API_Key:ISUQYWBWESINZE60
```

And you can get these result:



2. Input commands in the LG01 linux console

Open SSH and Login.

```
10.130.2.167 - default - SSH Secure Shell
File Edit View Window Help
Quick Connect Profiles

SSH Secure Shell 3.2.9 (Build 283)
Copyright (c) 2000-2003 SSH Communications Security Corp - http://www.ssh.com/
This copy of SSH Secure Shell is a non-commercial version.
This version does not include PKI and PKCS #11 functionality.

BusyBox v1.23.2 (2017-12-21 22:54:56 CST) built-in shell (ash)

DRAGINO
W i F i, L i n u x, M C U, E m b e d d e d
OpenWRT Chaos Calmer 15.05
Version: Dragino-v2 IoT-4.3.3
build wed feb 14 10:16:48 CST 2018
www.dragino.com
-----
root@dragino-193a1b:~# 
Connected to 10.130.2.167
```

Input command:

```
mosquitto_sub -h mqtt.thingspeak.com -p 1883 -u dragino -P UZ4NGHKJMKS9WR5E -t channels/465952/subscribe/json/ISUQYWBWESINZE60
```

```
root@dragino-193a18:~# mosquitto_sub -h mqtt.thingspeak.com -p 1883 -u dragino -P U24NGHKUMKS9WR5E -t channels/465952/subscribe/json/ISUQYWBMESINZE60
{"channel_id": "465952", "created_at": "2018-04-10T01:20:22Z", "entry_id": 660, "field1": "66.0", "field2": "28.0", "latitude": null, "longitude": null, "elevation": null, "status": "MQTTUPUBLISH"}

{"channel_id": "465952", "created_at": "2018-04-10T01:34:03+00:00", "entry_id": 661, "field1": "65.0", "field2": "28.0", "latitude": null, "longitude": null, "elevation": null, "status": "MQTTUPUBLISH"}
{"channel_id": "465952", "created_at": "2018-04-10T01:38:56+00:00", "entry_id": 662, "field1": "64.0", "field2": "28.0", "latitude": null, "longitude": null, "elevation": null, "status": "MQTTUPUBLISH"}
```

And if your end node send data continuously, it will receive the downlink data continuously.

11. Example: Integrate LoRa with TCP

11.1 What is TCP?

TCP is a connection-oriented transmission control protocol. It is the specified function of the fourth layer of the transport layer in the OSI model of the computer network system. The TCP layer is located above the IP layer and below the application layer. Reliable, pipe-like connections are often required between application layers of different hosts, but the IP layer does not provide such a streaming mechanism, but rather provides unreliable packet switching. TCP is used for communication between applications and needs to send communication requests. After three handshakes, the client and server form a full duplex to send and receive messages to each other.

11.2 Gateway configuration

1. Join the network

Small Enterprise-Campus Network

Internet Access

Access Internet Via	WiFi Client
SSID	dragino-office
Encryption	WPA-WPA2
Password	***** <i>SSID: The name of the wifi used PW: The pw of the wifi used</i>
Way to Get IP	DHCP
Display Net Connection	Domain or IP
<input checked="" type="checkbox"/> Continously Check Net Connection	

2. Click System-> Startup and add:/etc/iot/scripts/tcp_client &

Local Startup

This is the content of /etc/crc.local. Insert your own commands here (in front of 'exit 0') to execute them at the end of the boot process.

```
# Put your custom commands here that should be executed once
# the system init finished. By default this file does nothing.
/etc/iot/scripts/tcp_client &
exit 0
```


3. Enable TCP Server

Select IoT Server

Select the IoT Server type to connect

Select IoT Server

IoT Server	TCP/IP Protocol
Log Debug Info	Disable Debug Info
<input checked="" type="checkbox"/> Show Log in System Log	

4. Configure TCP Server port and address

dragino-18c700 Status Sensor System Network Logout

TCP Client

Communicate with IoT Server through a TCP Client socket

General Settings

Server Address: 10.130.2.157
Server Port: 60000

Save & Apply Save Reset

DRAGINO TECHNOLOGY CO., LIMITED

5. Configure frequency

Radio Settings

Radio settings requires MCU side sketch support

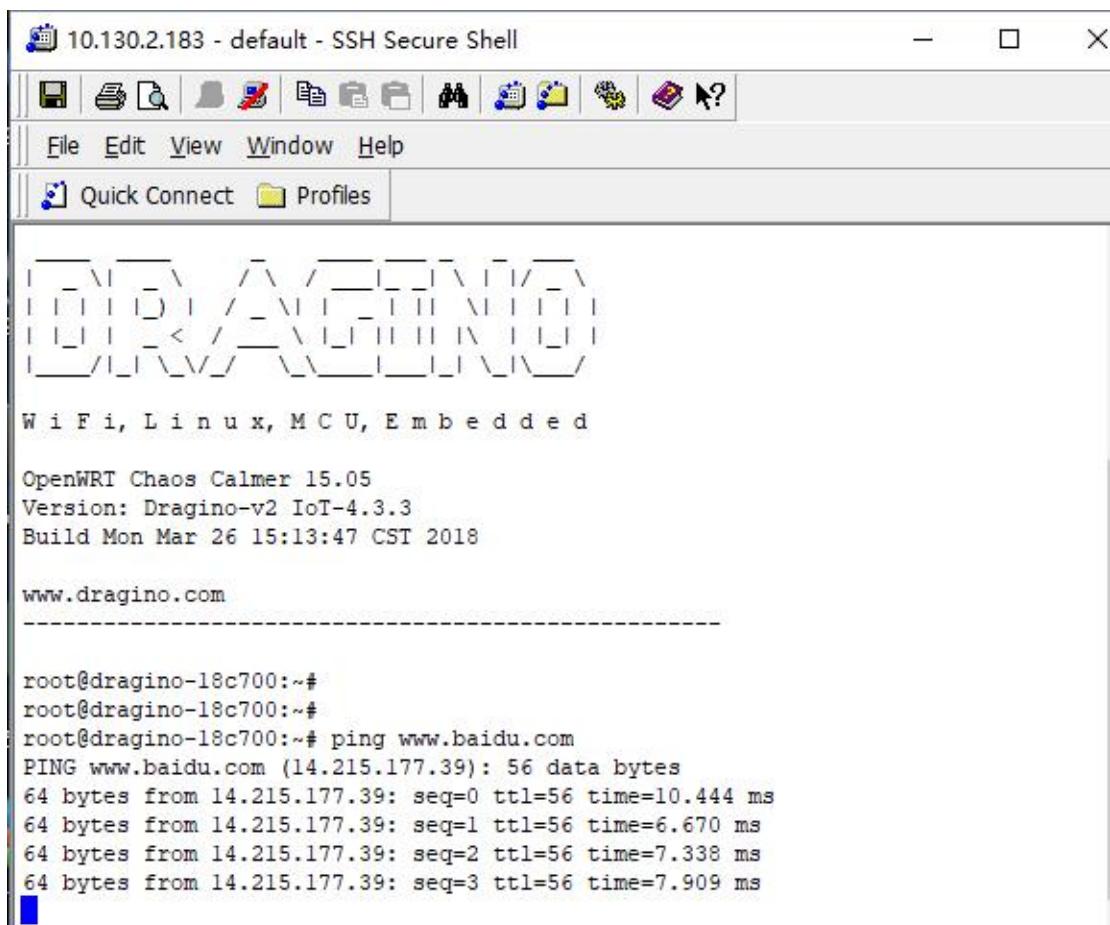
TX Frequency	9 digits Frequency, etc:868100000 <small>Gateway's LoRa TX Frequency</small>
RX Frequency	868300000 <small>Gateway's LoRa RX Frequency</small>
Encryption Key	Encryption Key
Spreading Factor	SF7
Transmit Spreading Factor	SF9
Coding Rate	4/5
Signal Bandwidth	125 kHz
Preamble Length	8 <small>Length range: 6 ~ 65536</small>

For testing used frequency 868 device

11.3 Simulation

Please upgrade the new version [4.3.4](#).

1. we need to make sure the LG01 has internet access. We can log in the SSH and ping an Internet address and see if it get through. As below:



10.130.2.183 - default - SSH Secure Shell

File Edit View Window Help

Quick Connect Profiles

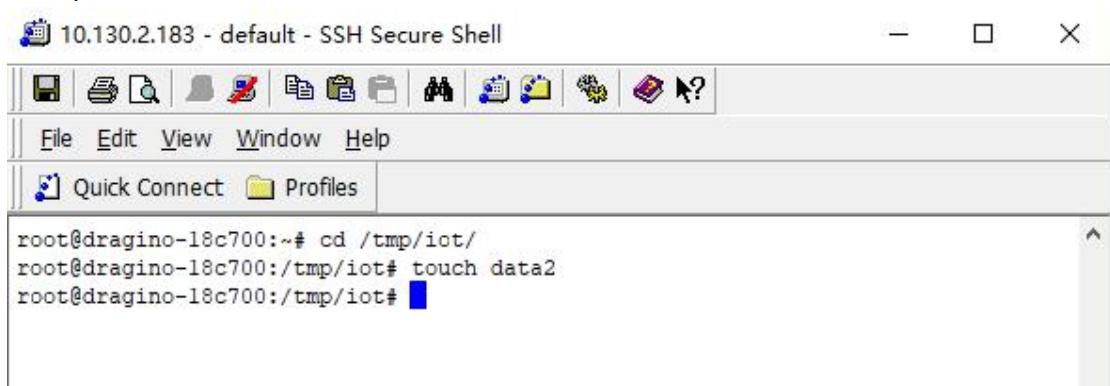
```
Wi-Fi, Linux, MCU, Embedded

OpenWRT Chaos Calmer 15.05
Version: Dragino-v2 IoT-4.3.3
Build Mon Mar 26 15:13:47 CST 2018

www.dragino.com

root@dragino-18c700:~#
root@dragino-18c700:~# ping www.baidu.com
PING www.baidu.com (14.215.177.39): 56 data bytes
64 bytes from 14.215.177.39: seq=0 ttl=56 time=10.444 ms
64 bytes from 14.215.177.39: seq=1 ttl=56 time=6.670 ms
64 bytes from 14.215.177.39: seq=2 ttl=56 time=7.338 ms
64 bytes from 14.215.177.39: seq=3 ttl=56 time=7.909 ms
```

2. Input the command and create a named data2 file.



10.130.2.183 - default - SSH Secure Shell

File Edit View Window Help

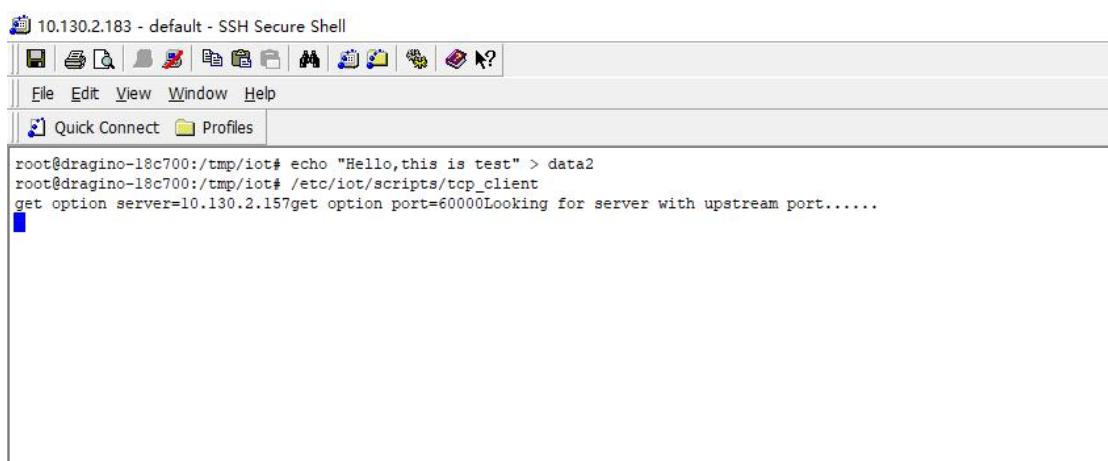
Quick Connect Profiles

```
root@dragino-18c700:~# cd /tmp/iot/
root@dragino-18c700:/tmp/iot# touch data2
root@dragino-18c700:/tmp/iot#
```

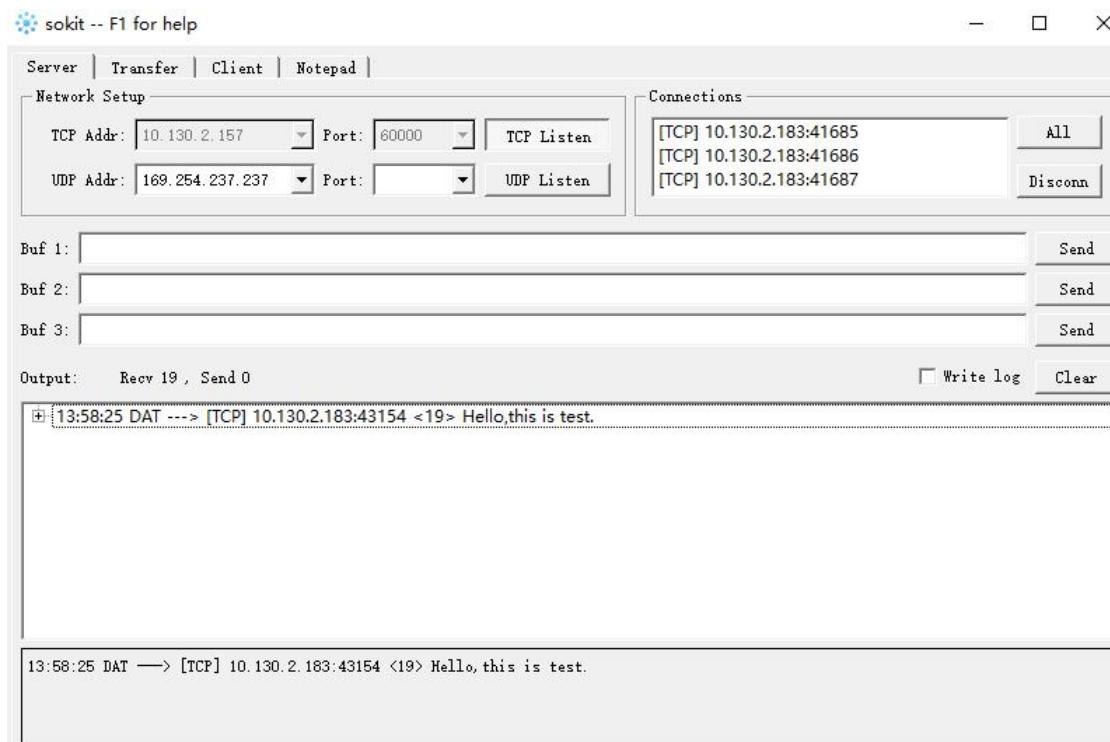
3. Open [Sokit](#) and create TCP Server.



4. Input the command and send the data "Hello,this is test"



4. Check result

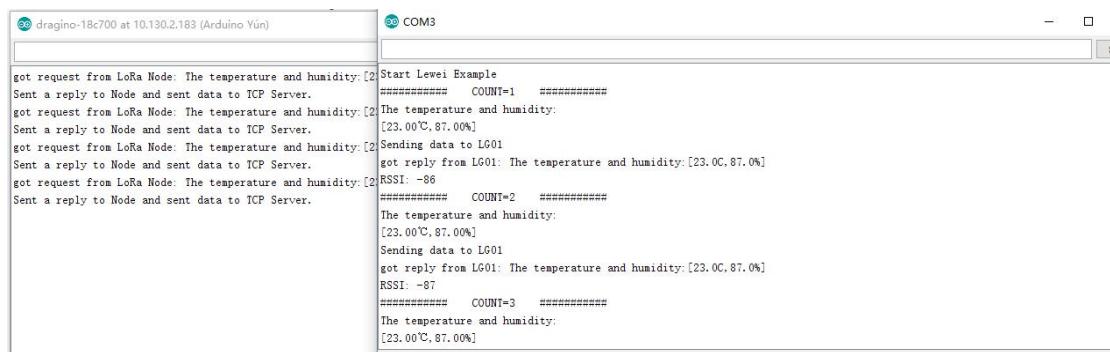


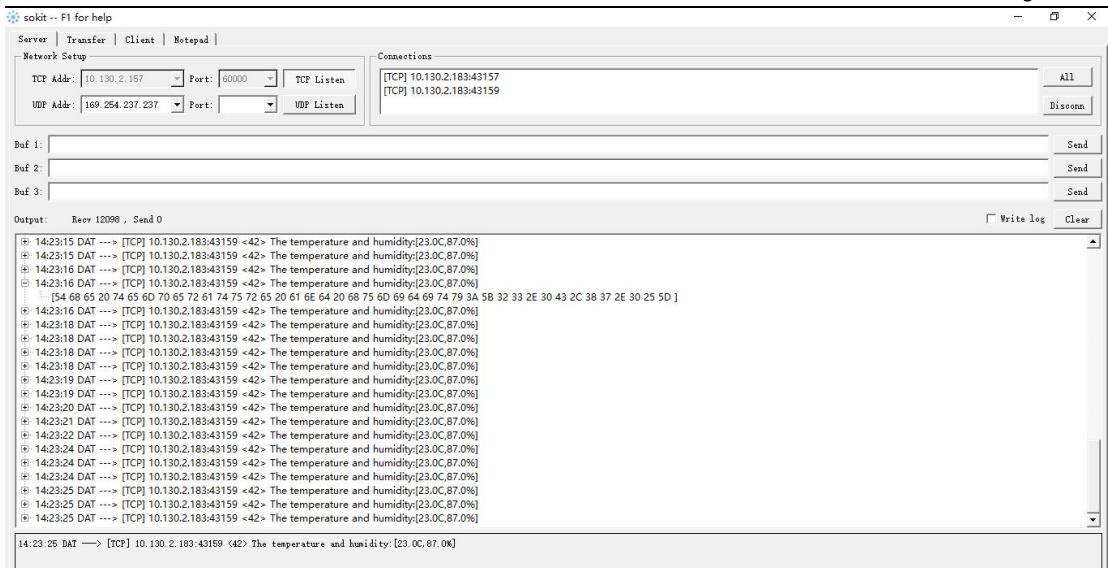
11.4 Send data

Here we will provide a complete example for our system. The hardware set up is as below:

- ✓ **LoRa Gateway LG01:** Periodically check if there new commands in IoT Server. If there is new command, broadcast the command string to its LoRa network.. .
- ✓ **LoRa End Node:** LoRa Shied + UNO, listening if there are commands in LoRa network. If there is new command and the node ID match. It will parse and execute the incoming command.

Code: [TCP Client, TCP Server](#).





The more :[TCP/IP Example](#).

12. Advance Examples

12.1 Example for Connecting to TTN LoRaWAN server

Please check this link for detail: [Connect to TTN](#)

12.2 Multiple Nodes examples

The example shows how the gateway can handle multiple nodes up to several hundreds. The example can be found at [IDE --> File --> Examples --> Dragino --> LoRa --> Concurrent](#)

How it works:

This concurrent client sketch is working together with the concurrent gateway sketch. Before using this sketch, please use the write_client_id sketch to write a client ID in the EEPROM. Client ID is the unique id for each client in the LoRa network. write_gateway_id to gateway is not necessary, if not write, gateway id will be 0xFF.

- When the client boot, it will keep listening for the broadcast message from LoRa Gateway.
- When the gateway sketch boot, it will broadcast a message to set up a LoRa Network. If it gets broadcast message, client will send a join request message to gateway, when the join request message arrive to gateway, the gateway will send back a join-ack message with client id and add this client to the LoRa Network.
- If the client gets its join-ack message for its join request, it will enter the mode to listen the data-request message from gateway. In this mode, if client get a data-request message for this client it will send back a data message to the gateway.
- After client in data_request listening mode, if it has not receive any message from gateway in a timeout, it will go back to the network set up mode to listen the broadcast message.
- Gateway will refresh the LoRa network periodically for adding new client or remove unreachable client.

This example using the polling method between LoRa node and Gateway, it will minimize the LoRa packets transfer on the air and avoid congestion. It is suitable for a not real time LoRa work.

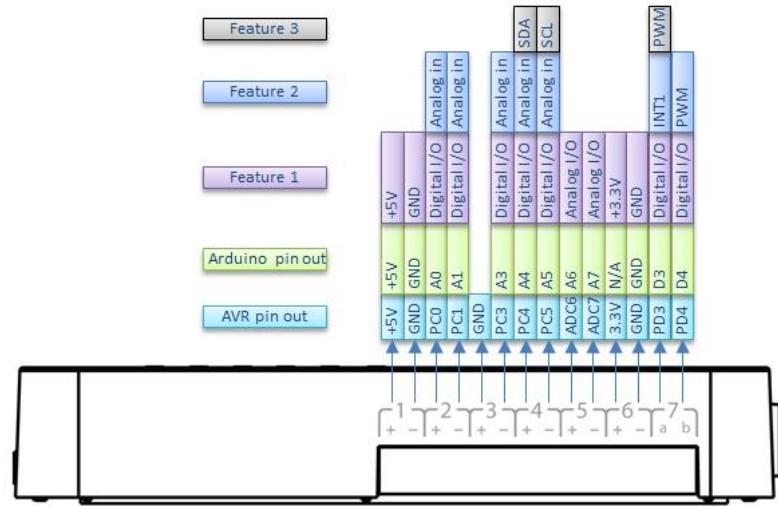
Performance test in a room with 100 nodes and 1 gateway shows:

- a) Gateway require about 1.5 minutes to set up this 100 nodes Network
- b) Gateway takes about 2 minutes to do polling for these 100 nodes.

User can adjust the timing in the sketch from case by case.

12.3 How to use the sensor pin of LG01-S?

The LG01-S has the external sensor pins from the Mega328P MCU, it can connect to external sensor, below is the pin definition for LG01-S:

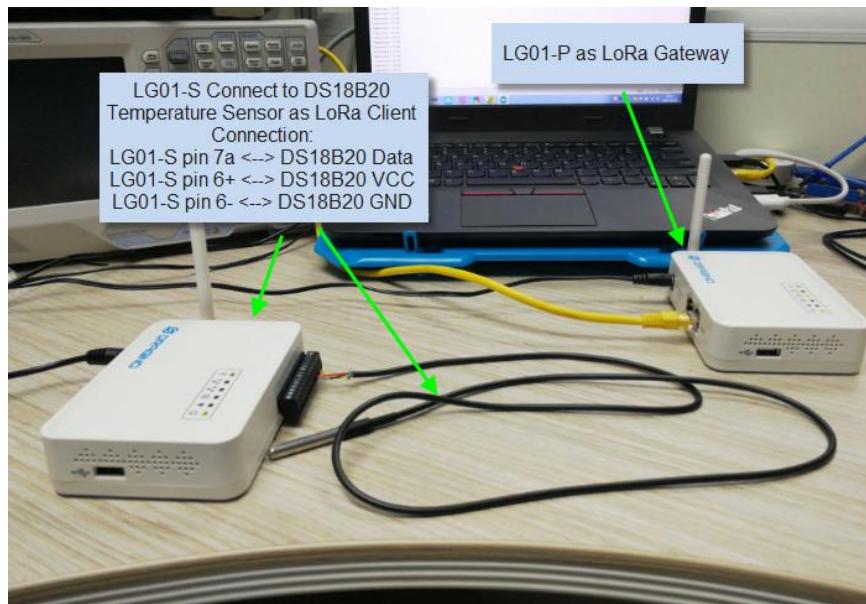


LG01-S Pinout

The program method for these pins are the same as Arduino, **what should notice is that the pins are 3.3v I/O base.**

Here is an example for how to use it with DS18B20 temperature sensor.

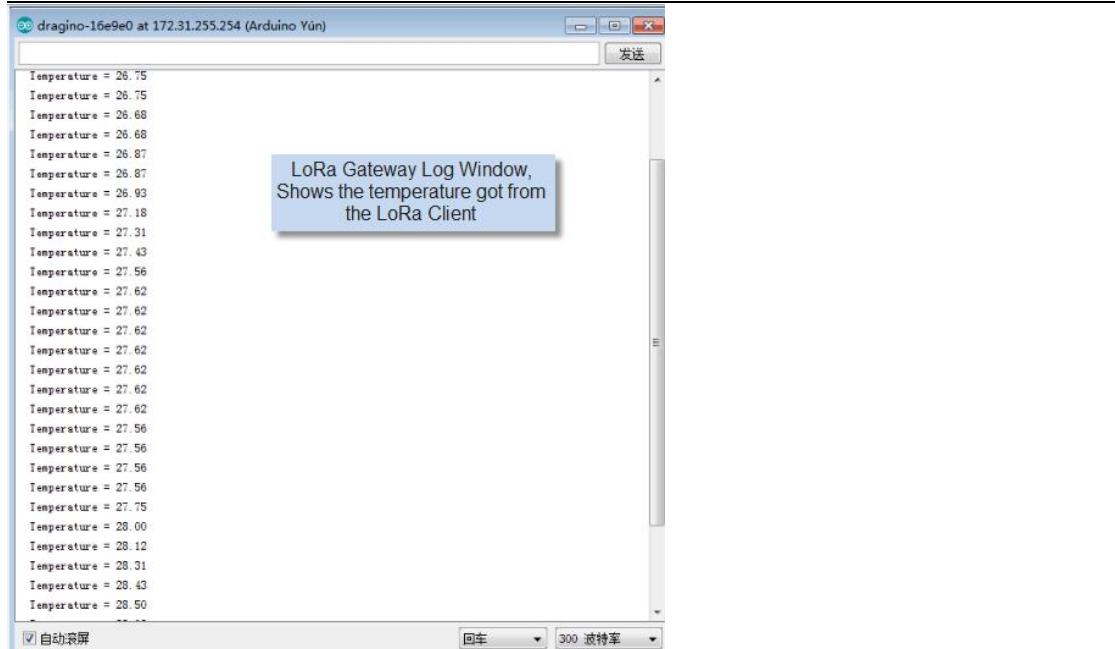
Hardware Setup as below:



Source code is in this link:

- ✓ [Sensor and LoRa client side](#)
- ✓ [Gateway side](#)

Result screen shot:



12.4 More Examples

Dragino keep updating examples in our wiki, more examples please refer the link:
[Dragino Examples Catalog](#)

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 LG01 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	433Mhz
		Band3(LF): 137 ~175Mhz	
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 kind of LoRa devices can the gateway support?

The LoRa part software is running in the Mega328P MCU. And we use the Radiohead Library as examples. If other LoRa devices are running the same Radiohead library, same frequency and same encryption, they should be able to communicate with this gateway.

User can also run other LoRa protocol on the MCU to support other LoRa devices they want.

Here is an Example to Show how to support RN2483: [RN2483 Compatible](#).

13.4 How many nodes can the LG01 support?

The maximum support end-node depends on how the communication (how often) between the end-nodes and gateway. In a lab testing using the [simple LoRa example](#), if the end nodes try to send data to the gateway at every 5 minutes, there will be data lost after the network has 20~30 nodes due to Channel Collision.

If user want to reach more nodes, user can consider using the polling method to ensure that each time will only have a LoRa signal transmit in the frequency. If the gateway uses polling method to get data from the end node, it can support several hundred nodes or more. Examples can see: [Polling example for LoRa](#).

13.5 What kind of Server the LG01 can support?

The Linux side of LG01 is OpenWrt, it is open source and users can develop application over it.

Basically it can support most IoT servers if use the right API. We have examples for how to connect some servers via typical protocol (MQTT,RESTful) for IoT, MQTT or RESTful. From this link: [IoT Server Examples](#).

13.6 Can I make my own firmware for LG01? Where can I find the source code of LG01?

Yes, User can make own firmware for LG01 for branding purpose or add customized application.

The LG01 source code and compile instruction can be found at:

<https://github.com/dragino/openwrt-cc-15.05>

13.7 How to get more examples for this device?

We keep releasing Arduino Examples in the Dragino examples directory under Arduino IDE. If user install the dragino board earlier and we have release new examples. The new ones won't show up in the IDE except the user update the board profile. To update, User can remove the board profile in Arduino board manager and install it again.

13.8 What is the Antenna require for OLG01?

The OLG01 shipped with a small LoRa Antenna inside the box as attached:



User can use this Spring Antenna for testing purpose. For field deployment, user can replace it with high performance outdoor antenna such as fibre-glass epoxy antenna. When choose antenna, please make sure the antenna works with the LoRa frequency for the device.

13.9 More FAQs about general LoRa questions

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

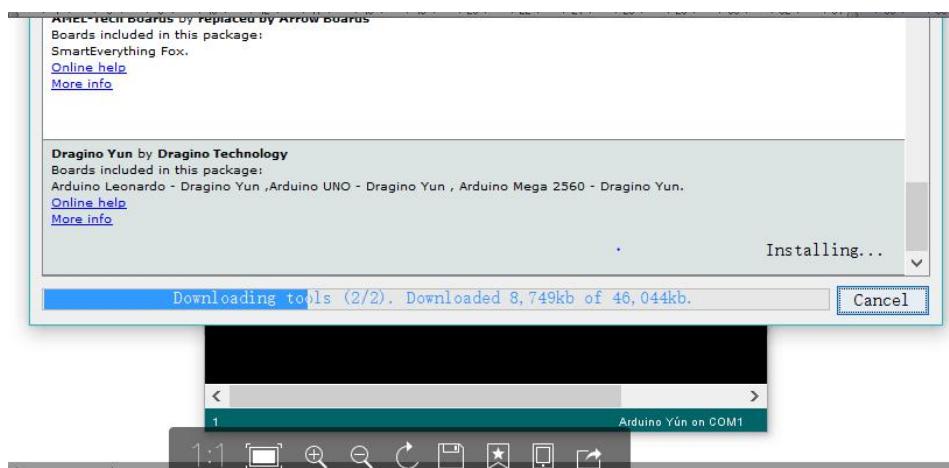
http://wiki.dragino.com/index.php?title=LoRa_Questions

14. Trouble Shooting

14.1 I can't download the Dragino profile in Arduino IDE.

If IDE quite slowly while downloading the Dragino profile in board manager and stuck somewhere.

As show below, it is because your network has slow connection to some packages from Arduino IDE.



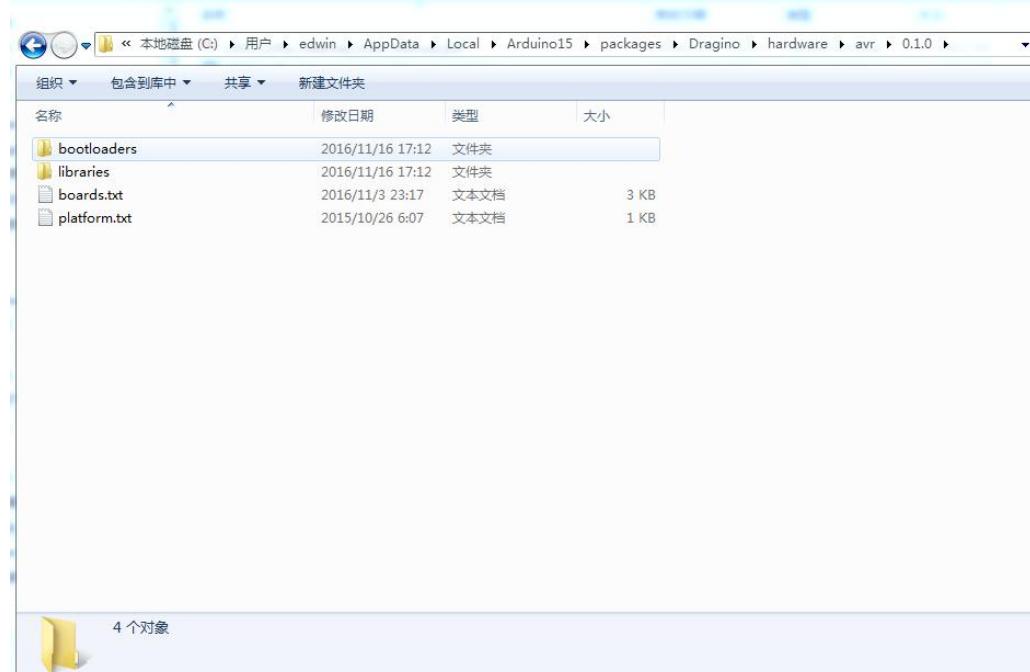
To solve this, user can manually add Dragino profile. Below is the step:

- 1/ Download the profile from <https://github.com/dragino/Arduino-Profile-Examples>
- 2/ Unzip it and put the content under this directory:

C:\Users\xxx\AppData\Local\Arduino15\packages\Dragino\hardware\avr\0.1.0

Note: Different system may have different directory for Arduino15, if you can't see Dragino\hardware\avr\0.1.0, just create it in your Arduino15 directory.

The final directory content should as below.



14.2 Bridge between MCU and Linux module doesn't work.

Some possibilities:

- 1/ You have used the **Serial class** in MCU sketch, like Serial.begin(9600), The bridge library in Mega328P use the same Serial interface. So if you have the Serial code in the sketch. They will conflict and bridge doesn't work.
- 2/ The IDE get mess in the serial setting when you compile other sketch . In this case, you can close the IDE and open it again.

14.3 Arduino IDE doesn't detect LG01

Check below points if this issue happens:

- ✓ The Arduino IDE version is 1.5.4 or later
- ✓ Your PC and Yun LG01 are in the same network.
- ✓ Try to access the LG01 via Web or SSH, then check the IDE again.
- ✓ If above still doesn't work, SSH log in the LG01 and run: **/etc/init.d/avahi-daemon restart** to restart the service so IDE can detect the LG01.

14.4 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
```

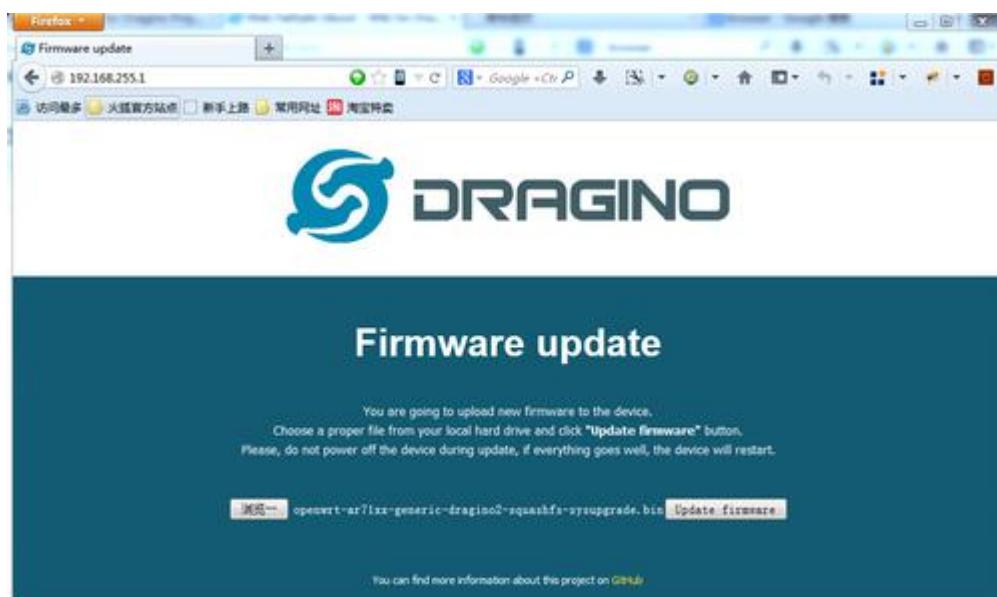
14.5 How to recover the LG01 if firmware crash

LG01 provide 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:

- Use a RJ45 cable to connect the PC to LG01's port directly.
- Set the PC to ip 192.168.255.x, netmask 255.255.255.0
- Pressing the toggle button and power on the device
- All LEDs of the device will blink, release the toggle button after four blinks
- 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.
- Open 192.168.255.1 in web browser
- Select a squashfs-sysupgrade type firmware and update firmware.



14.6 I configured LG01 for WiFi access and lost its IP. What to do now?

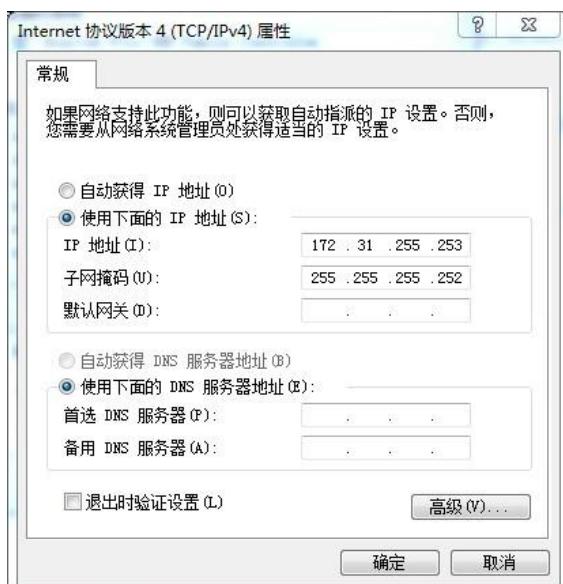
The LG01 has a fall-back ip in its LAN port. This IP is always enabled so user can use fall back ip to access LG01 no matter what the WiFi IP is. The fall back ip is useful for connect and debug the unit.

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

Steps to connect via fall back IP:

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

As below photo:



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

14.7 See GPIO20 issue when uploading.

The error shows:

"Sketch uses 11574 bytes (35%) of program storage space. Maximum is 32256 bytes.

Global variables use 834 bytes (40%) of dynamic memory, leaving 1214 bytes for local variables.

Maximum is 2048 bytes.

Can't export GPIO 20, already exported/busy?: Device or resource busy

avrdude done. Thank you."

This happen when you use Linux machine to compile and upload the sketch, please check your PC settings or change to Windows machine when this happen.

15. Order Info

General Version:

- **LG01P-433**: LoRa Gateway best tune to 433 MHz.
- **LG01P-868**: LoRa Gateway best tuned to 868 MHz.
- **LG01P-915**: LoRa Gateway best tuned to 915 MHz

Screw Terminal Version:

- **LG01S-433**: LoRa Gateway best tune to 433 MHz.
- **LG01S-868**: LoRa Gateway best tuned to 868 MHz.
- **LG01S-915**: LoRa Gateway best tuned to 915 MHz.

Outdoor Version:

- **OLG01-433**: LoRa Gateway best tune to 433 MHz.
- **OLG01-868**: LoRa Gateway best tuned to 868 MHz.
- **OLG01-915**: LoRa Gateway best tuned to 915 MHz.

16. Packing Info

Package Includes:

- ✓ LG01P or LG01S 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

- ✧ LG01 LoRa Gateway official wiki
- ✧ [More examples for LG01 LoRa Gateway](#)
- ✧ Source code for LG01 LoRa Gateway
<https://github.com/dragino/openwrt-cc-15.05>
- ✧ OpenWrt official Wiki
<http://www.openwrt.org/>
- ✧ Arduino Official Site:
<https://www.arduino.cc>
- ✧ Arduino bridge examples:
<https://www.arduino.cc/en/Tutorial/Bridge>
- ✧ Hardware Source Code:
The LG01 includes two parts of hardware:
 - ✓ MS14N Linux Mother Board:
<https://github.com/dragino/motherboard-hardware/tree/master/ms14n>
 - ✓ LoRa Daughter Board LoRa G:
<https://github.com/dragino/Lora/tree/master/LoRa%20G/v1.3>