Connection Instruction to Ubidot

Document Version: 1.0

Firmware Version: LG02_LG08—v5.1.15

For products: LG01-N, OLG01-N,LG02,OLG02

Version	Description	Date
1.0	Release	2019-Feb-14

Contents

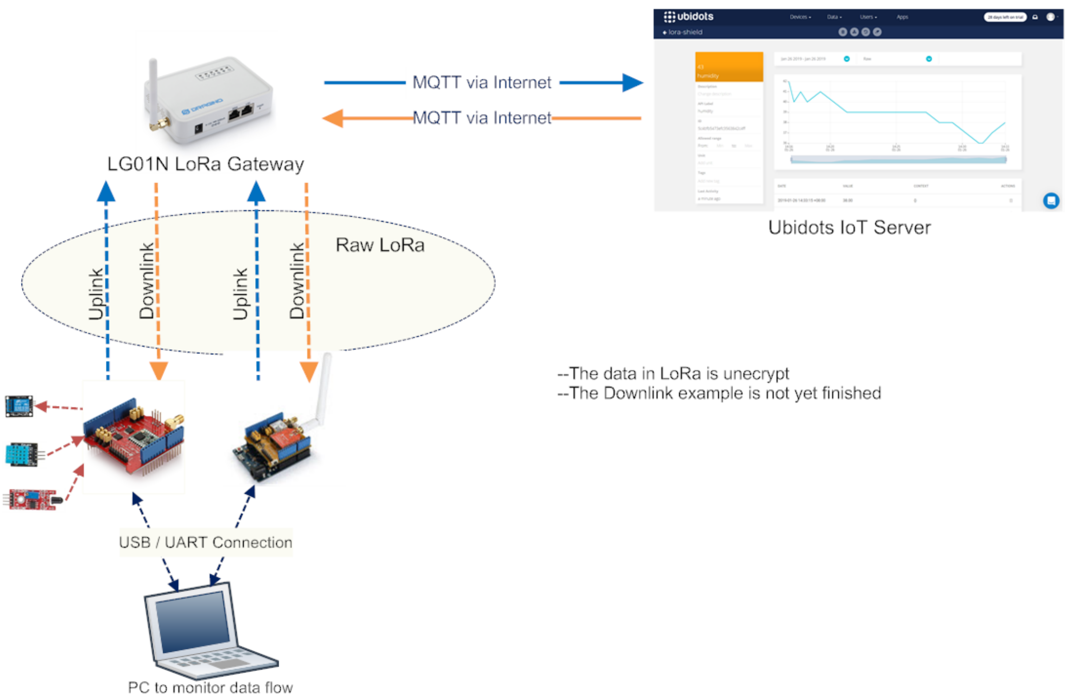
Example : Test MQTT with Ubidot IoT Server

This example describes how to use LG01-N, LoRa Shield & LoRa GPS Shield to set up a LoRa network and connect it to Ubidots IoT Server.

1.1 Typology and Data Flow

The network topology and dataflow for the example is as below:

Topology for Ubidots Connection:



--The data in LoRa is unecrypt
--The Downlink example is not yet finished

In next section we will start to configure for this example.

1.2 Prepare Hardware and Software

In the tutorial, there are two LoRa End Node, they are LoRa Shield + UNO and LoRa/GPS Shield + UNO. Both of them use Arduino UNO as MCU to control the LoRa transceiver.

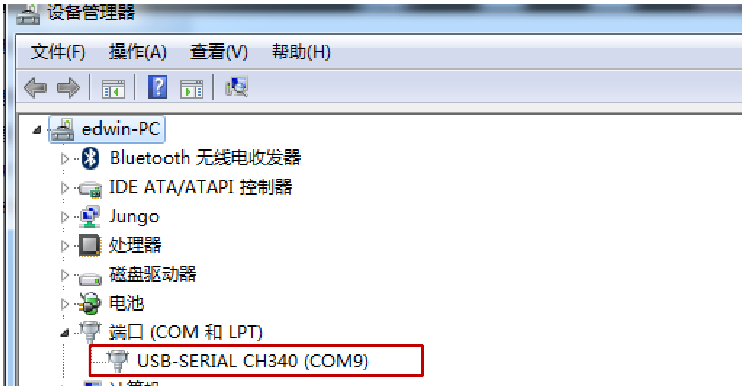
We need to program the Arduino UNO during our testing to support the required functions for end nodes. To finish this, we need to install some software and library first.

1.3 Prepare Software for End Node

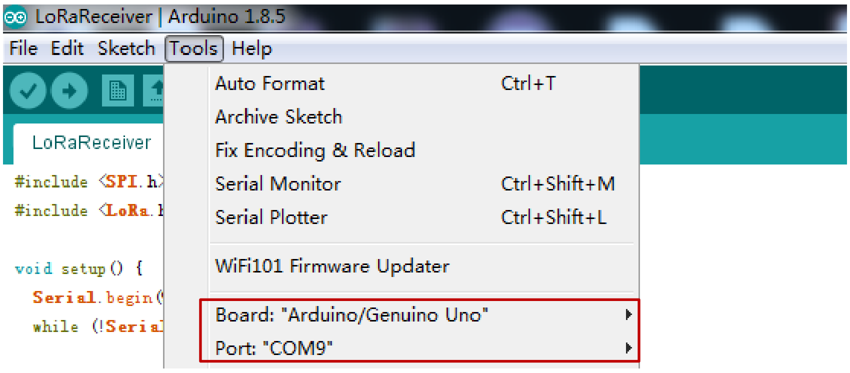
1.3.1 Install Arduino IDE and CH340 driver

First download and install Arduino IDE. This is the tool to program the Arduino UNO.

The Arduino UNO in the kit is clone version and is equipped with CH340 USB to UART chip. We need to install CH340 driver in the PC to let the Arduino IDE program it via USB. If we successful install the driver, a com port will show in the system device manager:

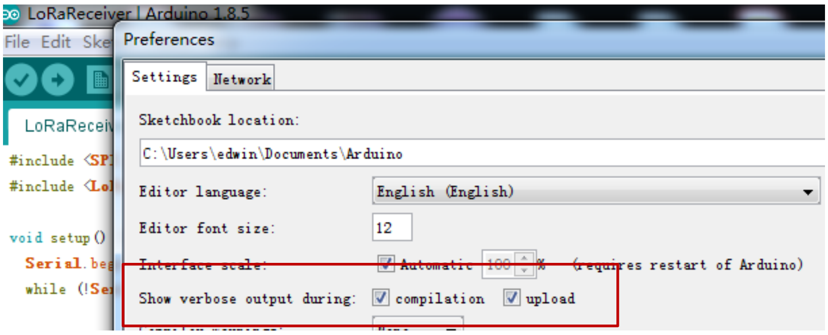


After install the driver, start Arduino and we will be able to use the board Arduino UNO and corresponding COM port to program UNO now.


LoRaReceiver | Arduino 1.8.5
File Edit Sketch Tools Help
LoRaReceiver
Auto Format Ctrl+T
Archive Sketch
Fix Encoding & Reload
Serial Monitor Ctrl+Shift+M
Serial Plotter Ctrl+Shift+L
WiFi101 Firmware Updater
Board: "Arduino/Genuino Uno"
Port: "COM9"

```
#include <SPI.h>
#include <LoRa.

void setup() {
  Serial.begin(
  while (!Seria
```

We can enable compilation and upload in Arduino → File → Preference. This will help us in debug.



```
#include <SP
#include <Lo

void setup()
  Serial.beg
  while (!Se
```

Sketchbook location: C:\Users\edwin\Documents\Arduino

Editor language: English (English)

Editor font size: 12

Interface scale: ☑ Automatic 100 % (requires restart of Arduino)

Show verbose output during: ☑ compilation ☑ upload

1.3.2 Install LoRa Library for Arduino

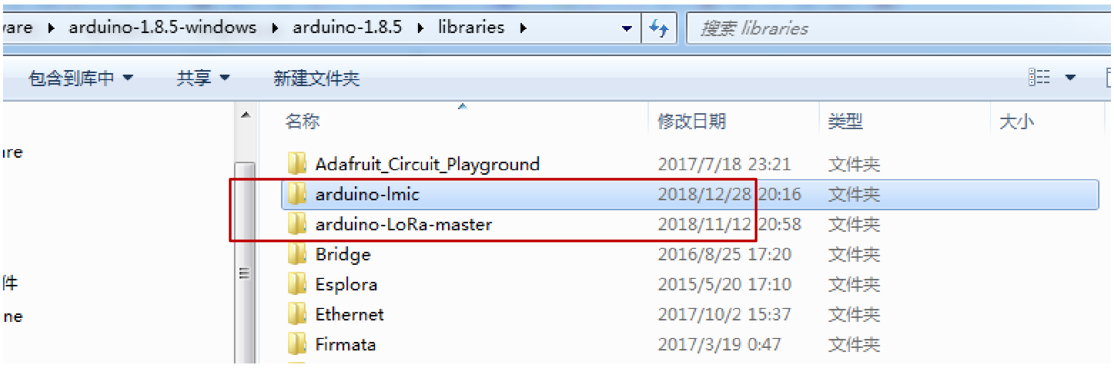
In our examples, we will use two different LoRa libraries for End Node to build different type of LoRa network. They are:

- Arduino-LMIC : LoRaWAN library to configure the End node as a standard LoRaWAN end node.
- LoRa-raw: This is a simple library for LoRa transmit & receive, all data transfer without ID control, encryption. If user wants to develop a LoRa network with private LoRa protocol, he can modify base on this Library.

We also need to install some libraries to connect to different sensors:

- DHTlib: This is the library to use DHT11 temperature & humidity sensor.
- TinyGPS: Library for LoRa GPS Shield to get the GPS data.

Download all above libraries and put them in the Arduino → Libraries directory



名称	修改日期	类型	大小
Adafruit_Circuit_Playground	2017/7/18 23:21	文件夹	
arduino-lmic	2018/12/28 20:16	文件夹	
arduino-LoRa-master	2018/11/12 20:58	文件夹	
Bridge	2016/8/25 17:20	文件夹	
Esplora	2015/5/20 17:10	文件夹	
Ethernet	2017/10/2 15:37	文件夹	
Firmata	2017/3/19 0:47	文件夹	

1.4 Prepare for LG01-N Gateway

In LoRa IoT Kit v2, we use LG01-N as LoRa Gateway. Unlike LG01-P in v1 kit, the LG01-N has its own LoRa utility and not need to program it via Arduino. Since we need to connect to Internet IoT Server, we need to configure the LG01-N to have internet access.

1.4.1 Configure LG01-N for internet connection.

Below steps show how to set up LG01-N to use WiFi for internet access.

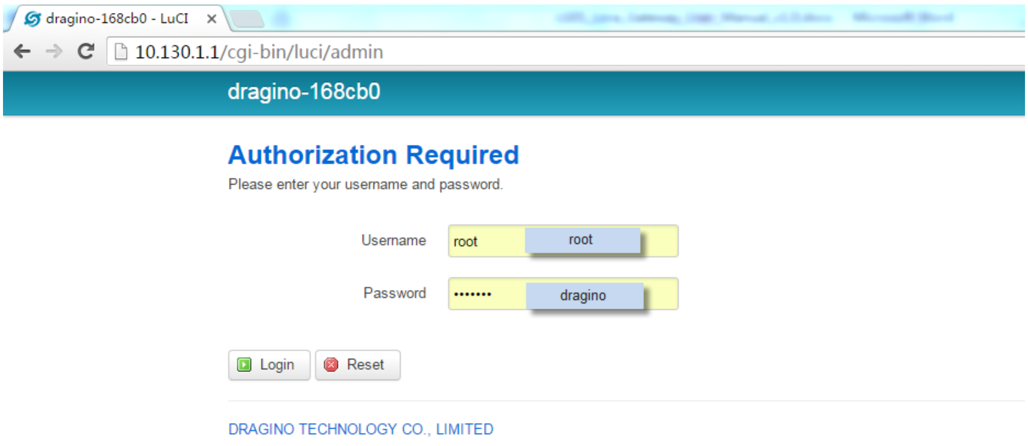
Step1:

Connect PC to LG01-N’s LAN port via RJ45 cable and set up PC Ethernet port to DHCP.
PC will then get IP from LG01-N. The ip range is 10.130.1.xx
Use browser to access the LG01-N via IP 10.130.1.1. (Recommend use Chrome here)

Step2:

Open a browser in the laptop and type
http://10.130.1.1/cgi-bin/luci/admin
User will see the login interface of LG01-N.
The account for Web Login is:

User Name: root
Password: dragino



dragino-168cb0 - LuCI

10.130.1.1/cgi-bin/luci/admin

dragino-168cb0

Authorization Required

Please enter your username and password.

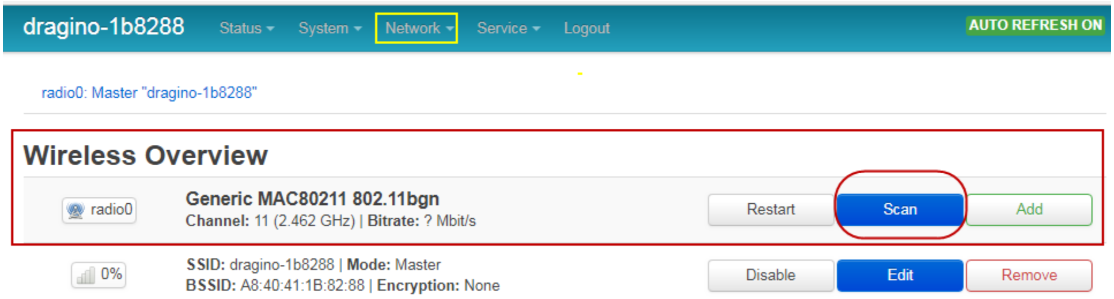
Username: root

Password: •••••••

DRAGINO TECHNOLOGY CO., LIMITED

Step3:

In network -> Wireless, select radio0 interface and scan.



radio0: Master "dragino-1b8288"

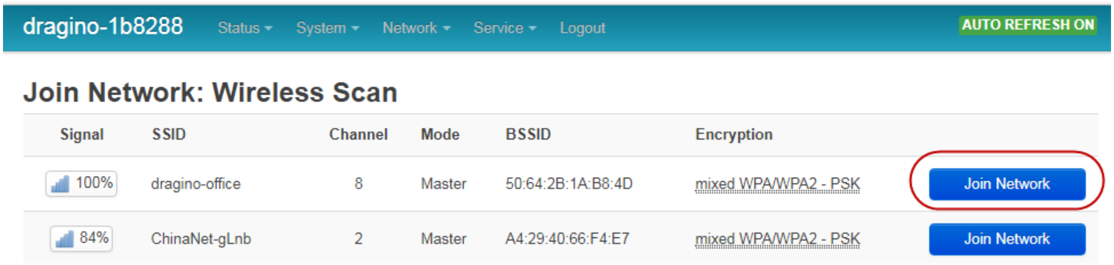
Wireless Overview

radio0 **Generic MAC80211 802.11bgn**
Channel: 11 (2.462 GHz) | **Bitrate:** ? Mbit/s

0% **SSID:** dragino-1b8288 | **Mode:** Master
BSSID: A8:40:41:1B:82:88 | **Encryption:** None

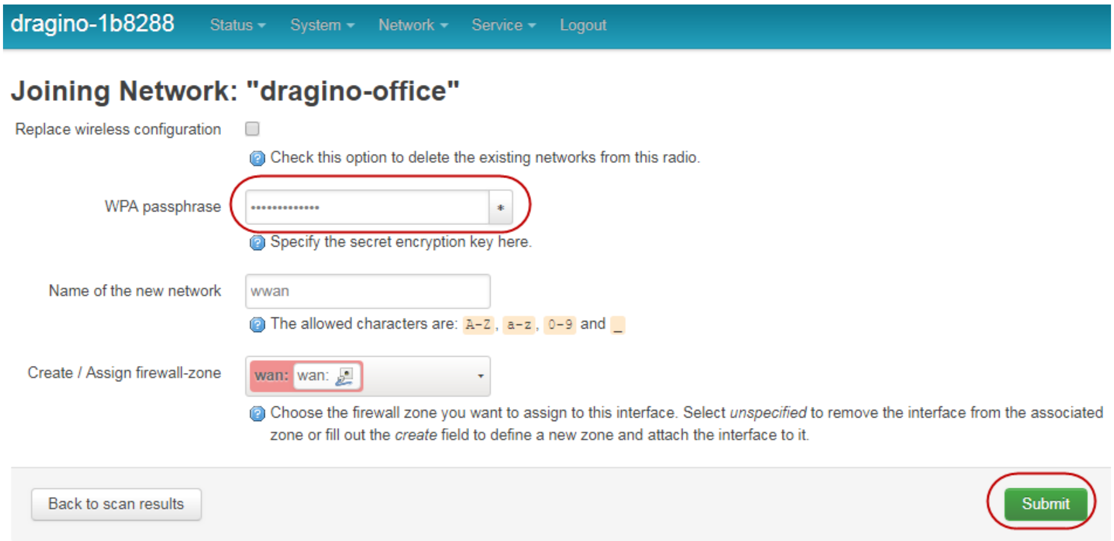
Step4:

Select the wireless AP and join the wifi network:



Join Network: Wireless Scan

Signal	SSID	Channel	Mode	BSSID	Encryption	
100%	dragino-office	8	Master	50:64:2B:1A:B8:4D	mixed WPA/WPA2 - PSK	Join Network
84%	ChinaNet-gLnb	2	Master	A4:29:40:66:F4:E7	mixed WPA/WPA2 - PSK	Join Network



Joining Network: "dragino-office"

Replace wireless configuration □
Check this option to delete the existing networks from this radio.

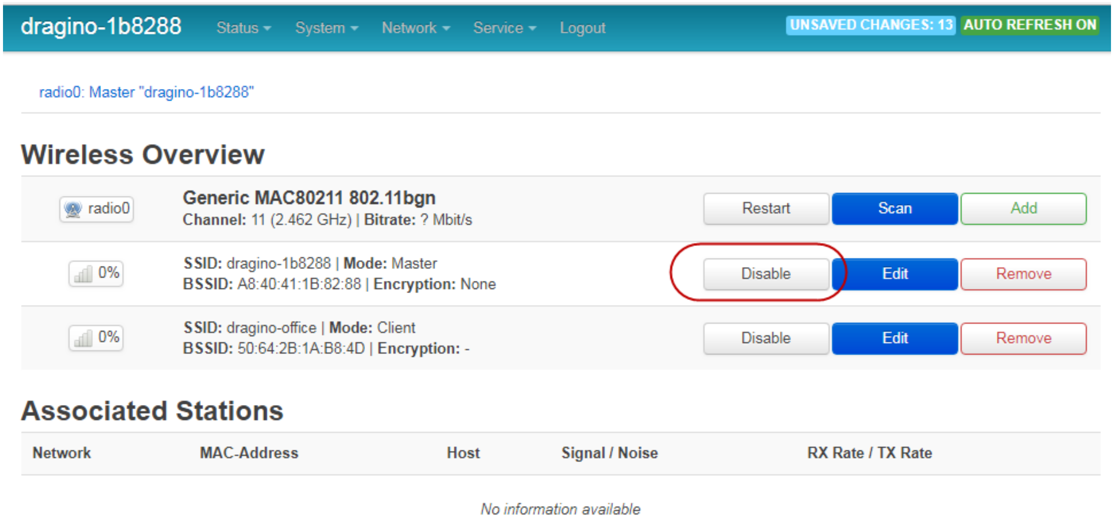
WPA passphrase: ••••••••••••• *
Specify the secret encryption key here.

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

Create / Assign firewall-zone: wan: wan:
Choose the firewall zone you want to assign to this interface. Select *unspecified* to remove the interface from the associated zone or fill out the *create* field to define a new zone and attach the interface to it.

Step5:

In network->wireless page, disable WiFi AP network. Notice: After doing that, you will lose connection if your computer connects to the LG01-N via its WiFi network.



Wireless Overview

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

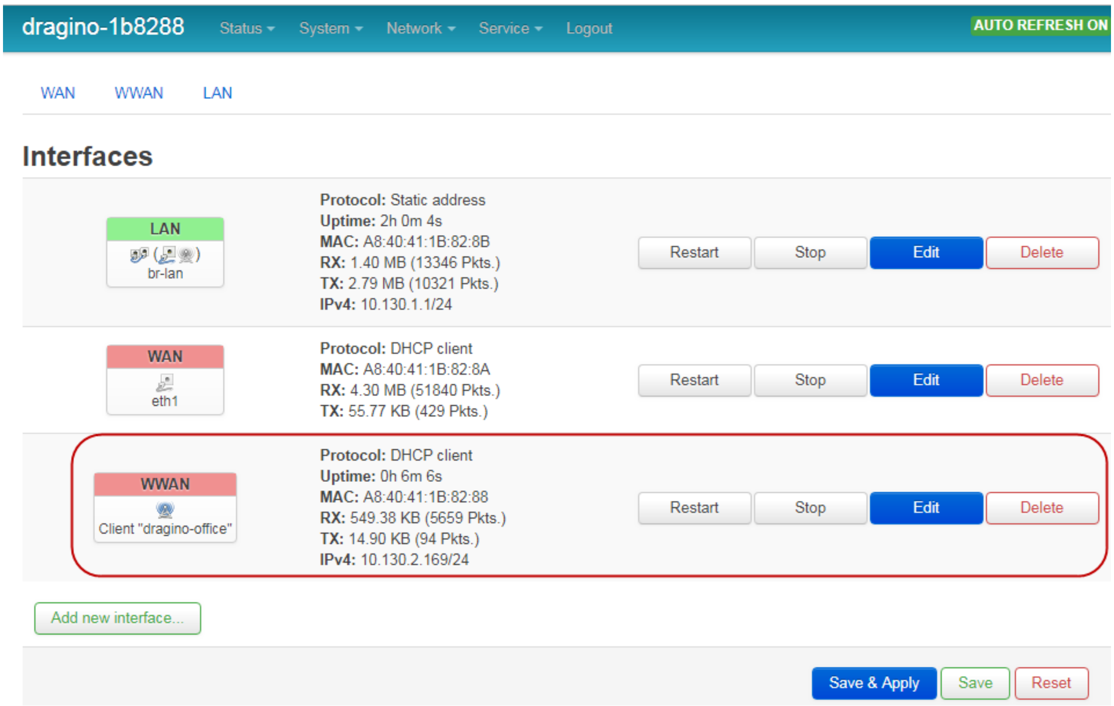
Associated Stations

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

No information available

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

After successful associate, the WiFi network interface can be seen in the same page and see LG01-N get the ip from the uplink router.



Interfaces

LAN
br-lan

Protocol: Static address
Uptime: 2h 0m 4s
MAC: A8:40:41:1B:82:8B
RX: 1.40 MB (13346 Pkts.)
TX: 2.79 MB (10321 Pkts.)
IPv4: 10.130.1.1/24

WAN
eth1

Protocol: DHCP client
MAC: A8:40:41:1B:82:8A
RX: 4.30 MB (51840 Pkts.)
TX: 55.77 KB (429 Pkts.)

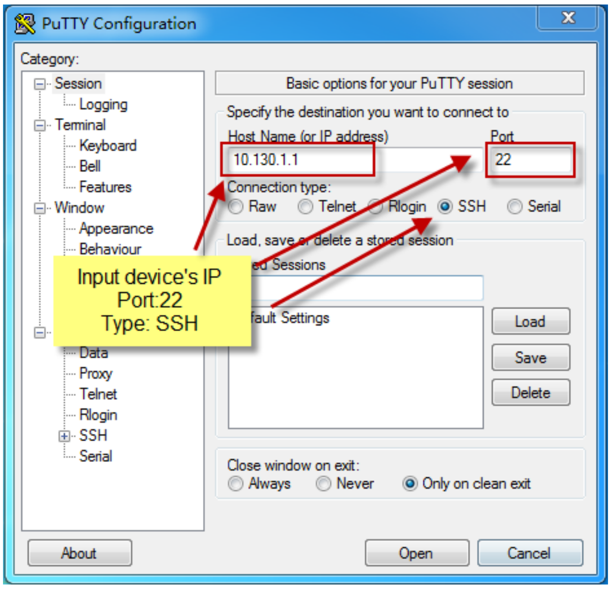
WWAN
Client "dragino-office"

Protocol: DHCP client
Uptime: 0h 6m 6s
MAC: A8:40:41:1B:82:88
RX: 549.38 KB (5659 Pkts.)
TX: 14.90 KB (94 Pkts.)
IPv4: 10.130.2.169/24

1.4.2 Download putty tool to access LG01-N via SSH

It will be helpful to see the LG01-N inside Linux system to understand the data flow and debug.

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



Specify the destination you want to connect to

Host Name (or IP address): 10.130.1.1

Port: 22

Connection type: ○ Raw ○ Telnet ○ Rlogin ● SSH ○ Serial

Close window on exit: ○ Always ○ Never ● Only on clean exit

Input device's IP
Port:22
Type: SSH

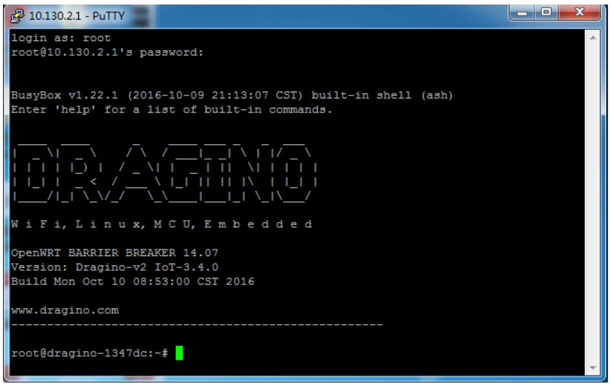
IP address: IP address of LG01-N

Port: 22

User Name: **root**

Password: **dragino** (default)

After log in, you will be in the Linux console and can input commands here.

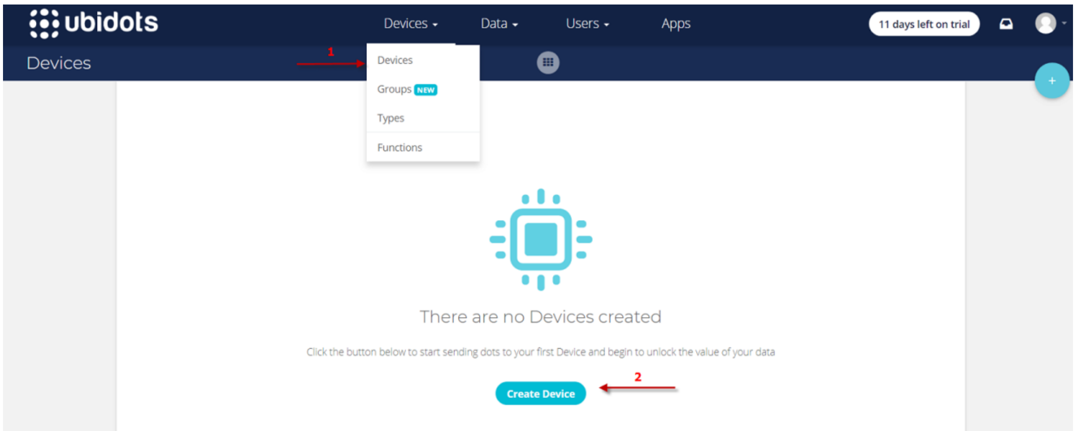


10.130.2.1 - PuTTY

```
login as: root
root@10.130.2.1's password:


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

 ____  ____      _    ____ ___ _   _  ___
|  _ \|  _ \    / \  / ___|_ _| \ | |/ _ \
| | | | |_) |  / _ \| |  _ | ||  \| | | | |
| |_| |  _ <  / ___ \ |_| || || |\  | |_| |
|____/|_| \_\/_/   \_\____|___|_| \_|\___/

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

OpenWRT BARRIER BREAKER 14.07
Version: Dragino-v2 IoT-3.4.0
Build Mon Oct 10 08:53:00 CST 2016

www.dragino.com
----------------------------------------------------

root@dragino-1347dc:~#
```

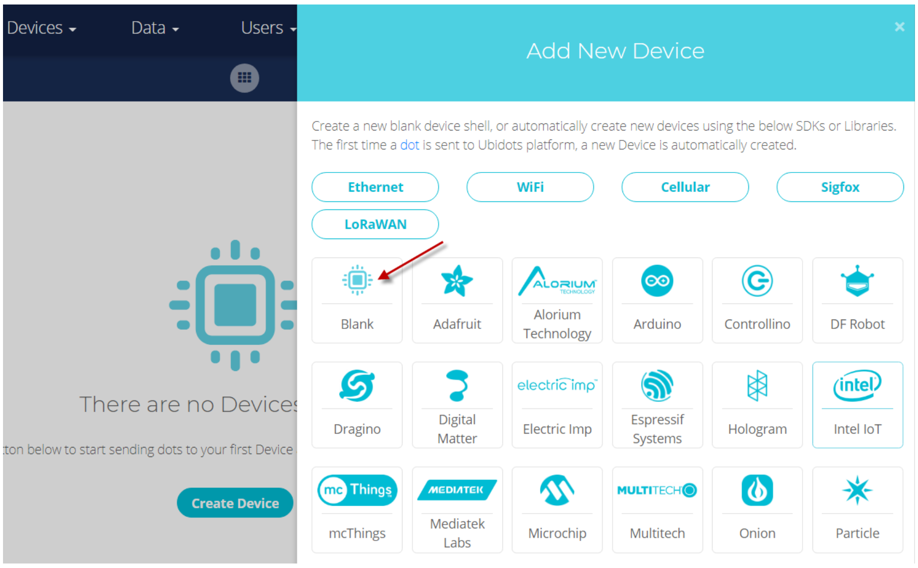
1.5 Create devices in Ubidots

Step 1: Log in Ubidots and create a device.



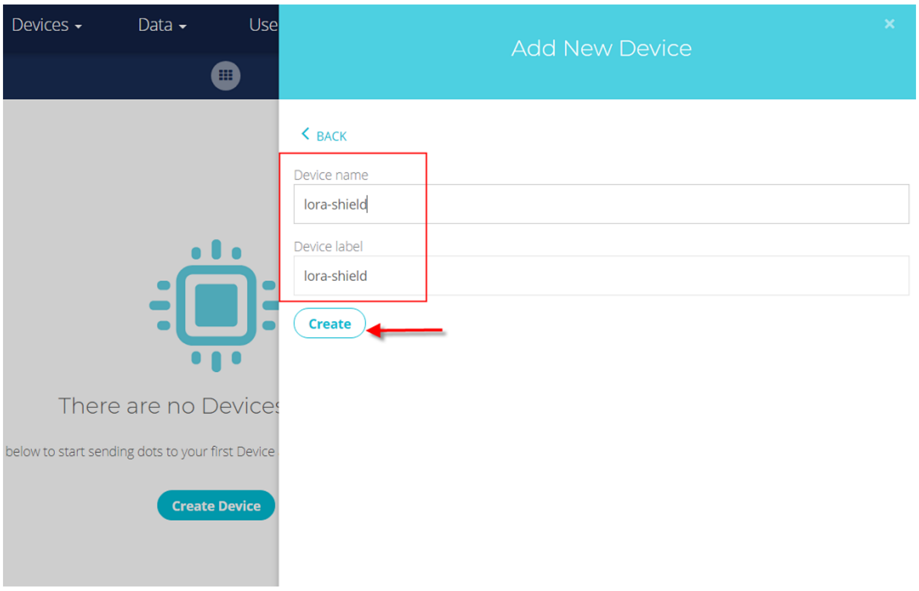
There are no Devices created

Click the button below to start sending dots to your first Device and begin to unlock the value of your data



Add New Device

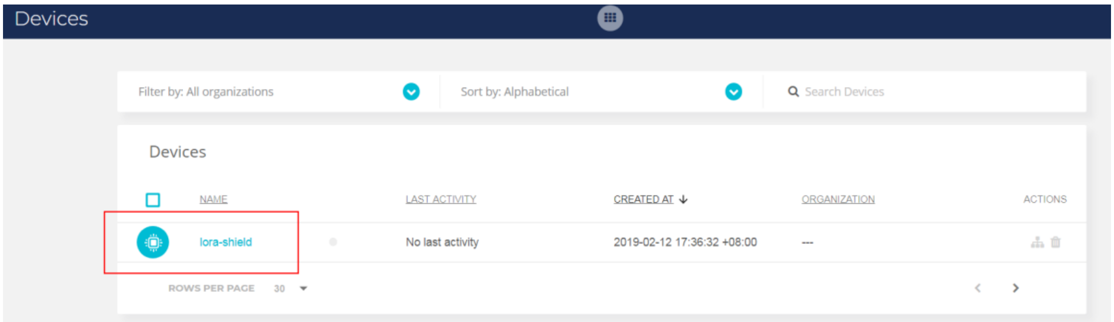
Create a new blank device shell, or automatically create new devices using the below SDKs or Libraries. The first time a dot is sent to Ubidots platform, a new Device is automatically created.



Add New Device

Device name: lora-shield

Device label: lora-shield



Devices

Filter by: All organizations | Sort by: Alphabetical | Search Devices

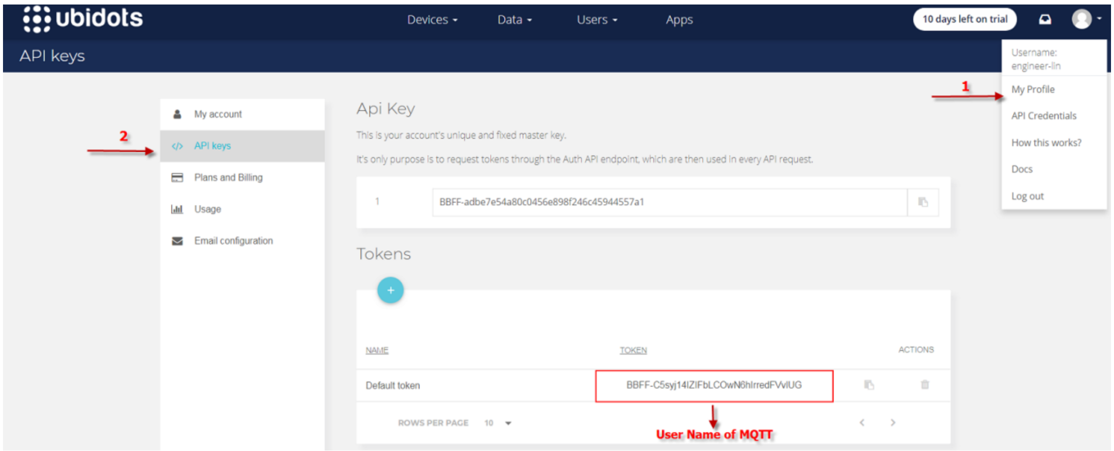
Devices

NAME	LAST ACTIVITY	CREATED AT	ORGANIZATION	ACTIONS
lora-shield	No last activity	2019-02-12 17:36:32 +08:00	---	

ROWS PER PAGE 30

Step 2: Get the TOKEN,API Label .

(1)Go to Account → My profile and get the **TOKEN.**



Api Key

This is your account's unique and fixed master key.

It's only purpose is to request tokens through the Auth API endpoint, which are then used in every API request.

1 BBFF-adbe7e54a80c0456e898f246c45944557a1

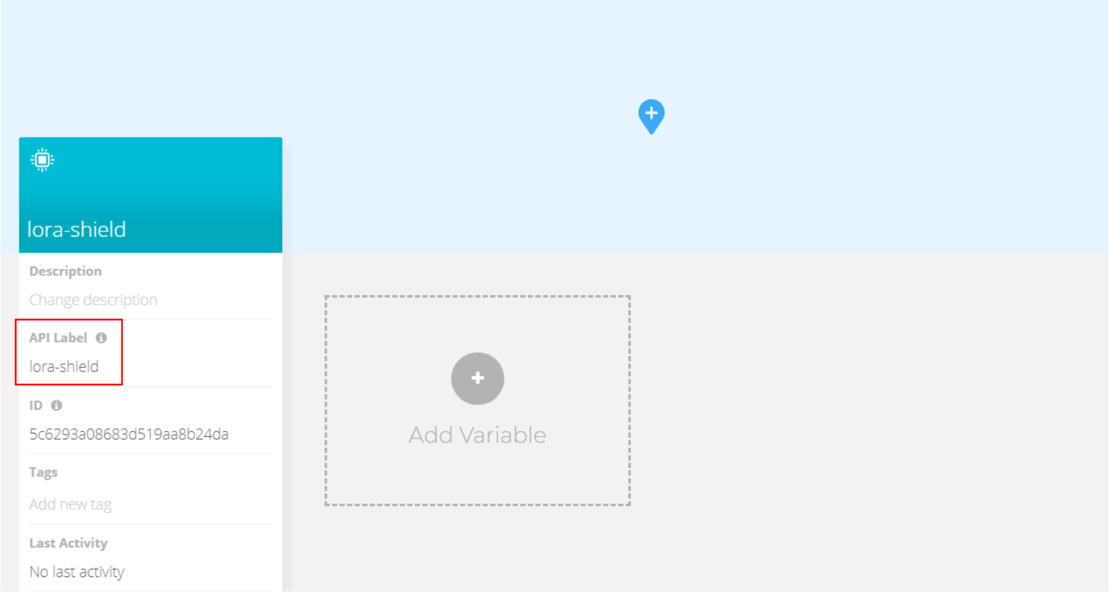
Tokens

NAME	TOKEN	ACTIONS
Default token	BBFF-C5syj14lZIFbLCOwN6hlrredFVvlUG	

ROWS PER PAGE 10

User Name of MQTT

(2)Go to Device → lora-shield and get the **API Label.**



lora-shield

Description
Change description

API Label
lora-shield

ID
5c6293a08683d519aa8b24da

Tags
Add new tag

Last Activity
No last activity

1.6 Simulate MQTT uplink via PC's MQTT tool

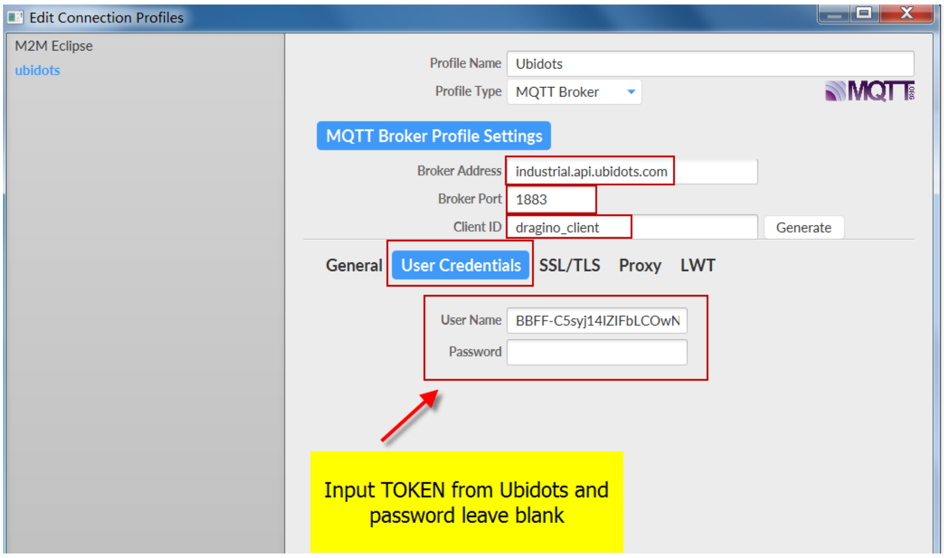
This step is not necessary, it just to help user to understand the MQTT protocol and simulate the MQTT connection to Ubidots. And check if the account info is valid and correct.

In the PC, download and install MQTT.fx. Open MQTT.fx and configure add a new MQTT client, (LoRa GPS-Shield is similar)as below:

Broker Address: industrial.api.ubidots.com

Broker Port: 1883

Client ID: dragino_client



Edit Connection Profiles

M2M Eclipse
ubidots

Profile Name: Ubidots
Profile Type: MQTT Broker

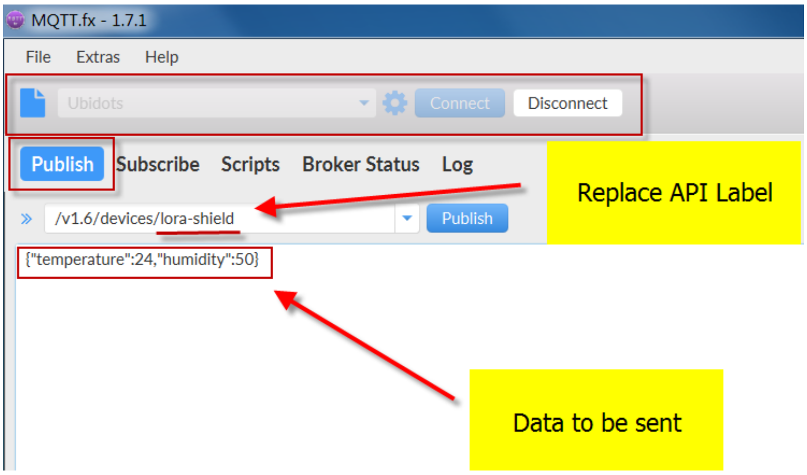
MQTT Broker Profile Settings

Broker Address: industrial.api.ubidots.com
Broker Port: 1883
Client ID: dragino_client

User Name: BBFF-C5syj14IZIFbLCOwN
Password:

Input TOKEN from Ubidots and password leave blank

After add the profile, connect it and publish. Publish MQTT connect it to Ubidots API docs



MQTT.fx - 1.7.1

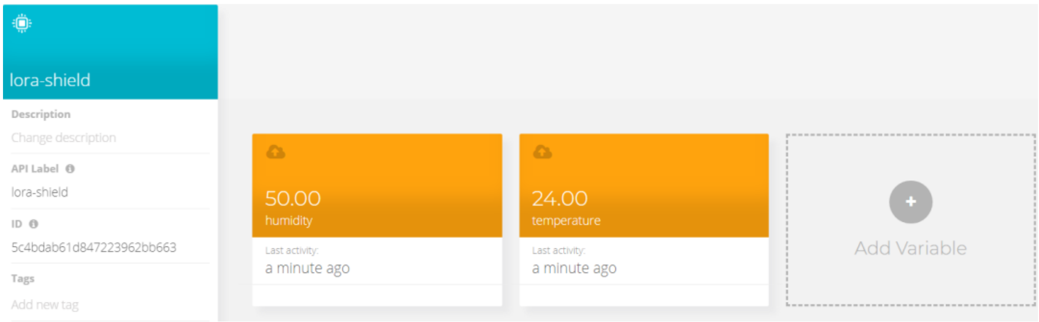
/v1.6/devices/lora-shield

```
{"temperature":24,"humidity":50}
```

Replace API Label

Data to be sent

If update successful, we can see the update in the devices:



lora-shield

Description
Change description

API Label
lora-shield

ID
5c4bdab61d847223962bb663

Tags
Add new tag

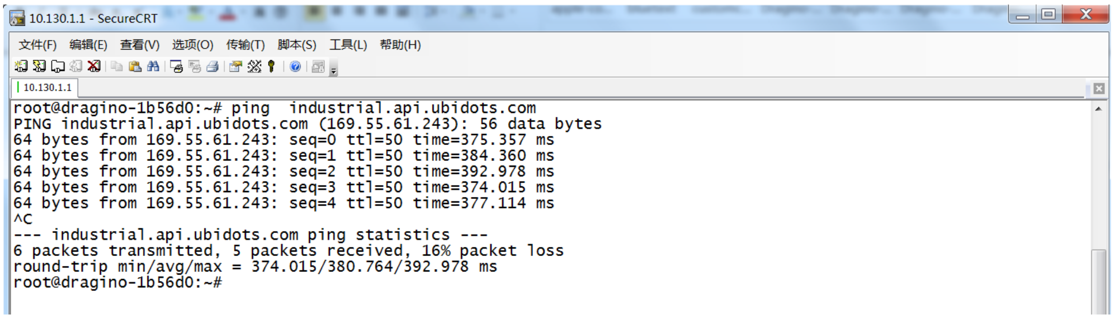
50.00
humidity
Last activity:
a minute ago

24.00
temperature
Last activity:
a minute ago

1.7 Try MQTT Publish with LG01-N Linux command

This step is also not necessary; it is to show the basic command used for MQTT connection and will help for further debug when connection fails.

First, we need to make sure the LG01-N has internet access. We can log in the SSH and ping an Internet address and see if there is reply. As below:



10.130.1.1 - SecureCRT
文件(F) 编辑(E) 查看(V) 选项(O) 传输(T) 脚本(S) 工具(L) 帮助(H)
10.130.1.1

```
root@dragino-1b56d0:~# ping  industrial.api.ubidots.com
PING industrial.api.ubidots.com (169.55.61.243): 56 data bytes
64 bytes from 169.55.61.243: seq=0 ttl=50 time=375.357 ms
64 bytes from 169.55.61.243: seq=1 ttl=50 time=384.360 ms
64 bytes from 169.55.61.243: seq=2 ttl=50 time=392.978 ms
64 bytes from 169.55.61.243: seq=3 ttl=50 time=374.015 ms
64 bytes from 169.55.61.243: seq=4 ttl=50 time=377.114 ms
^C
--- industrial.api.ubidots.com ping statistics ---
6 packets transmitted, 5 packets received, 16% packet loss
round-trip min/avg/max = 374.015/380.764/392.978 ms
root@dragino-1b56d0:~#
```

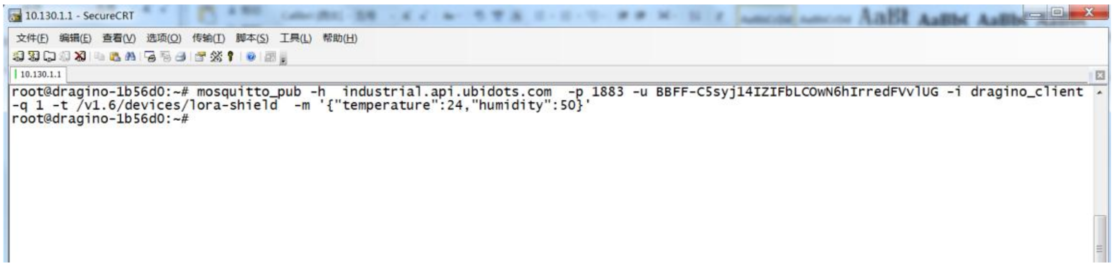
LG01-N has built-in Linux utility **mosquitto_pub**. We can use this command to publish the data to Ubidots.

The command to update a feed is as below:

LoRa-Shield: mosquitto_pub -h industrial.api.ubidots.com -p 1883 -u BBFF-C5syj14IZIFbLCOwN6hIrredFVvlUG -i dragino_client -q 1 -t /v1.6/devices/lora-shield -m '{"temperature":24,"humidity":50}'

LoRa GPS-Shield: mosquitto_pub -h industrial.api.ubidots.com -p 1883 -u BBFF-C5syj14IZIFbLCOwN6hIrredFVvlUG -i dragino_client -q 1 -t /v1.6/devices/lora-shield -m '{"location":{"value": 1, "context":{"lat":37.773, "lng":-122.431}}}'

(Make sure the “” is included, otherwise only one data will be uploaded)
Below is the output window(LoRa GPS-Shield is similar):



10.130.1.1 - SecureCRT
文件(F) 编辑(E) 查看(V) 选项(O) 传输(T) 脚本(S) 工具(L) 帮助(H)
10.130.1.1

```
root@dragino-1b56d0:~# mosquitto_pub -h  industrial.api.ubidots.com  -p 1883 -u BBFF-C5syj14IZIFbLCOwN6hIrredFVvlUG -i dragino_client
-q 1 -t /v1.6/devices/lora-shield  -m '{"temperature":24,"humidity":50}'
root@dragino-1b56d0:~#
```

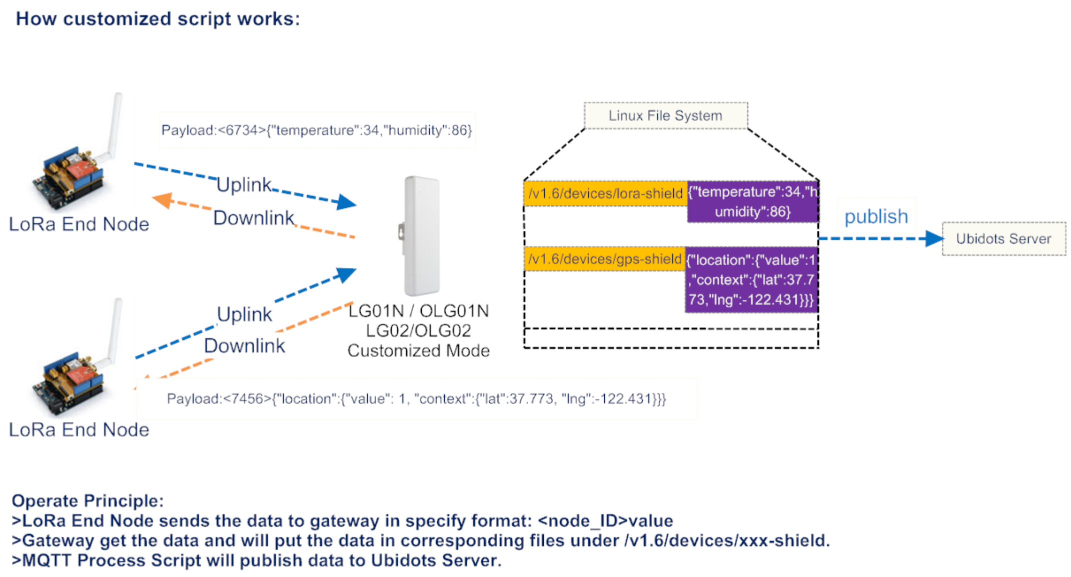
After running this command, we can see the data are updated to Ubidots, which has same result as what we did at mqtt.fx.

So we success to use LG01-N to uplink data to Ubidots, the **mosquitto_pub** command is executed in the Linux side, finally, we will have to call **mosquitto_pub** command while the LoRa sensor data arrive. We will explain how to do that in next step.

1.8 Configure LG01-N Gateway

1.8.1 Publish Logic

In LG01-N (firmware version > LG02_LG08--build-v5.1.1549961114-20190212-1646), there is a built-in script to process the MQTT data. The logic of this flow is as below:

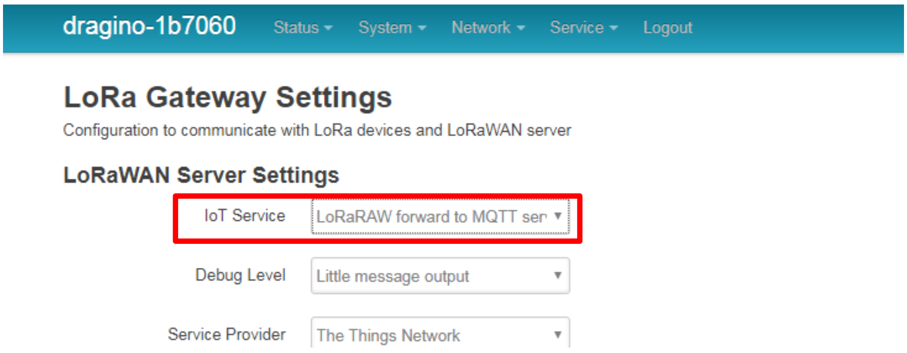


How customized script works:

Payload:<6734>{"temperature":34,"humidity":86}
LoRa End Node
Uplink
Downlink
Uplink
Downlink
LoRa End Node
LG01N / OLG01N
LG02/OLG02
Customized Mode
Payload:<7456>{"location":{"value": 1, "context":{"lat":37.773, "lng":-122.431}}}
Linux File System
/v1.6/devices/lora-shield
{"temperature":34,"humidity":86}
/v1.6/devices/gps-shield
{"location":{"value":1,"context":{"lat":37.773,"lng":-122.431}}}
publish
Ubidots Server

Operate Principle:
>LoRa End Node sends the data to gateway in specify format: <node_ID>value
>Gateway get the data and will put the data in corresponding files under /v1.6/devices/xxx-shield.
>MQTT Process Script will publish data to Ubidots Server.

Step1: Configure LG01-N to act as MQTT mode



LoRa Gateway Settings

Configuration to communicate with LoRa devices and LoRaWAN server

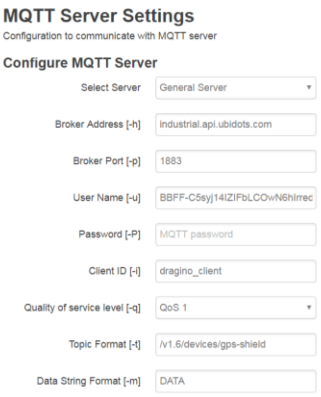
LoRaWAN Server Settings

IoT Service: LoRaRAW forward to MQTT ser

Debug Level: Little message output

Service Provider: The Things Network

Step2: Configure MQTT server info



MQTT Server Settings

Configuration to communicate with MQTT server

Configure MQTT Server

Select Server	General Server
Broker Address [-h]	industrial.api.ubidots.com
Broker Port [-p]	1883
User Name [-u]	BBFF-C5syj14IZIFbLCOwN6hIrrec
Password [-P]	MQTT password
Client ID [-i]	dragino_client
Quality of service level [-q]	QoS 1
Topic Format [-t]	/v1.6/devices/gps-shield
Data String Format [-m]	DATA

In step 2, we have below settings:

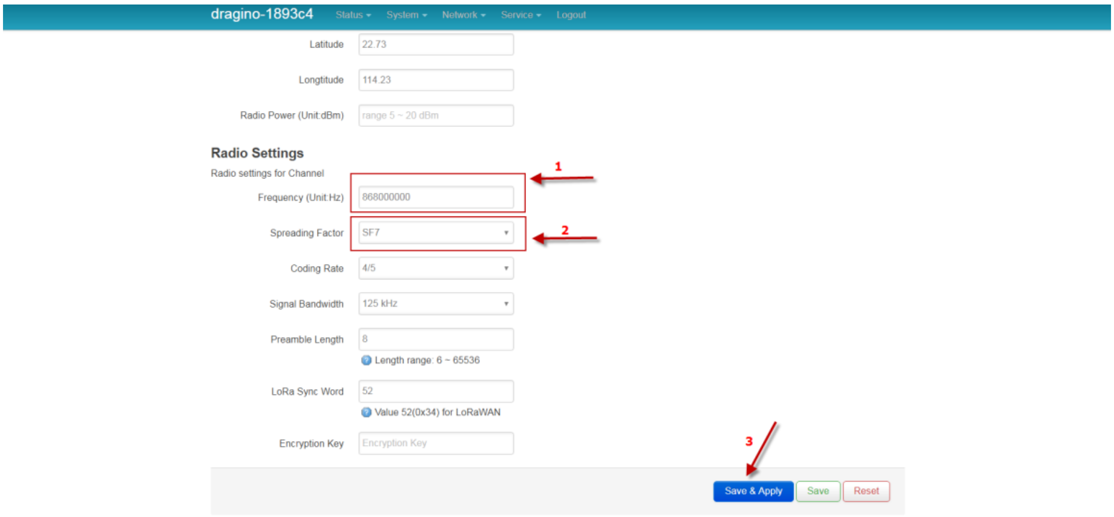
- ✓ Select Server: General Server
- ✓ Broker Address[-h]: industrial.api.ubidots.com
- ✓ Broker Port[-p]: 1883
- ✓ UserName[-u option]: Input TOKEN (user name for MQTT Connection)
- ✓ Password[-P option]: Leave blank
- ✓ Client_ID[-i]: dragino_client (can put any string)
- ✓ Quality of service level[-q]: QoS 1
- ✓ Topic Format[-t]: /v1.6/devices/lora-shield (lora-shield is API Label of devices on the ubidots)
- ✓ Data String Format[-m]: DATA

And we configure this channel:

- ✓ Local Channel ID: 5678
- ✓ Remote Channel ID: Leave blank
- ✓ Write_api_key: Leave blank

1.8.2 Configure LG01-N's Radio frequency

Now we should configure LG01-N's radio parameter to receive the LoRaWAN packets. We are using 868.0Mhz (868000000 Hz) as below:



Latitude: 22.73

Longtitude: 114.23

Radio Power (Unit:dBm): range 5 ~ 20 dBm

Radio Settings

Radio settings for Channel

Frequency (Unit:Hz): 868000000

Spreading Factor: SF7

Coding Rate: 4/5

Signal Bandwidth: 125 kHz

Preamble Length: 8

Length range: 6 ~ 65536

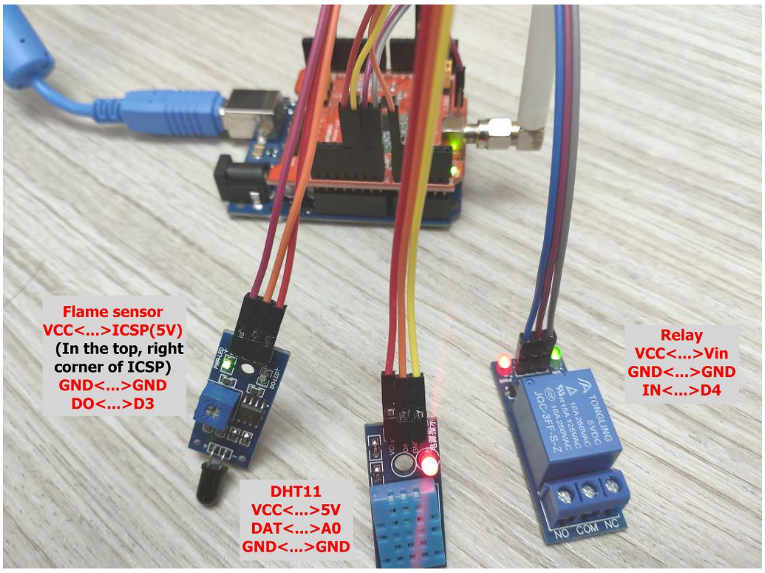
LoRa Sync Word: 52

Value 52(0x34) for LoRaWAN

Encryption Key: Encryption Key

1.9 Create LoRa Shield End Node

1.9.1 Hardware Connection



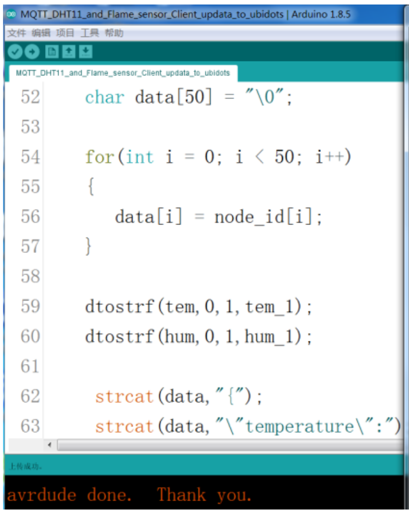
There are three sensors connect to the LoRa Shield + UNO. These sensors are flame sensors, DHT11 (Temperature & Humidity sensor) and Relay. Please use the connection as we show in the photo.

Note: There is a trick above, the relay is connected to VIN. In this case, The UNO can only be power via USB port. If need to power via DC power adapter, please use another 5v pin to power relay.

Upload this sketch to the UNO, this sketch will send temperature and humidity data to gateway at every 60 seconds. If there is a flame detect, it will then immediately send the value to gateway and then upload to the IoT Server.

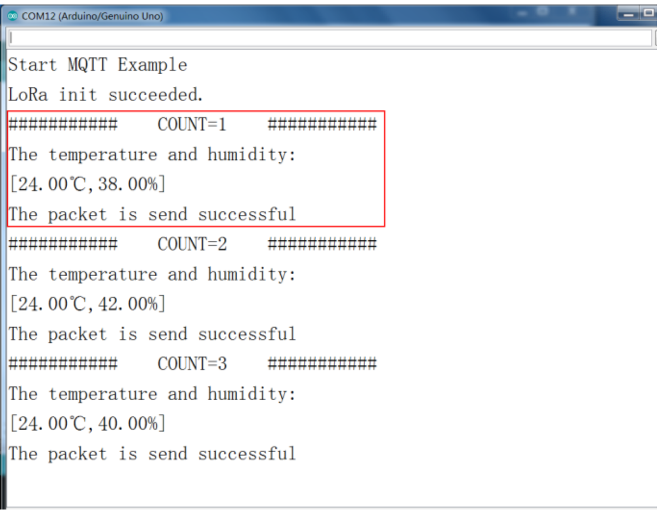
1.9.2 Test with uplink

After we upload the sketch to UNO, we can see below output from Arduino


MQTT_DHT11_and_Flame_sensor_Client_updata_to_ubidots | Arduino 1.8.5
文件 编辑 项目 工具 帮助
MQTT_DHT11_and_Flame_sensor_Client_updata_to_ubidots

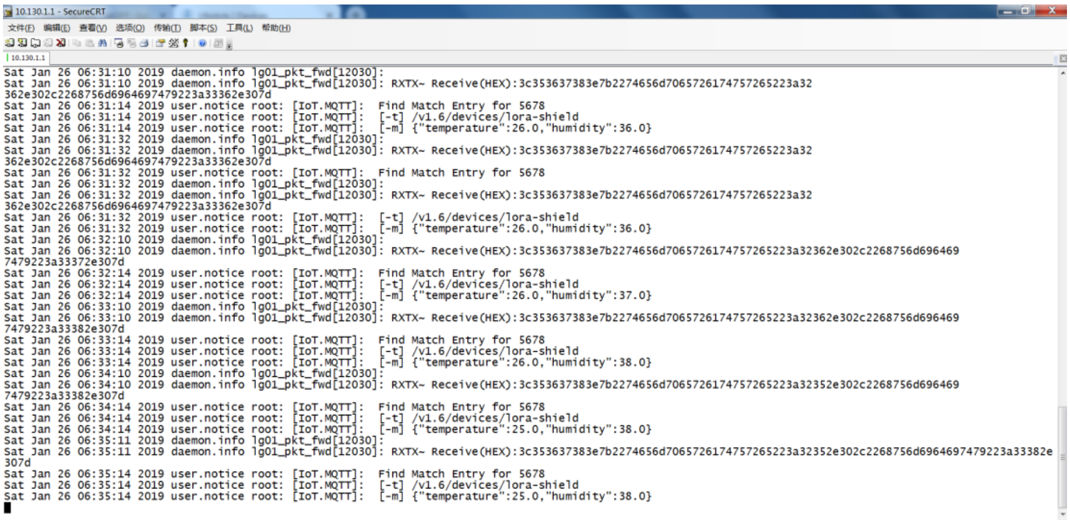
```
  char data[50] = "\0";

  for(int i = 0; i < 50; i++)
  {
     data[i] = node_id[i];
  }

  dtostrf(tem,0,1,tem_1);
  dtostrf(hum,0,1,hum_1);

   strcat(data,"{");
   strcat(data,"\"temperature\":")
```

```
avrdude done.  Thank you.
```



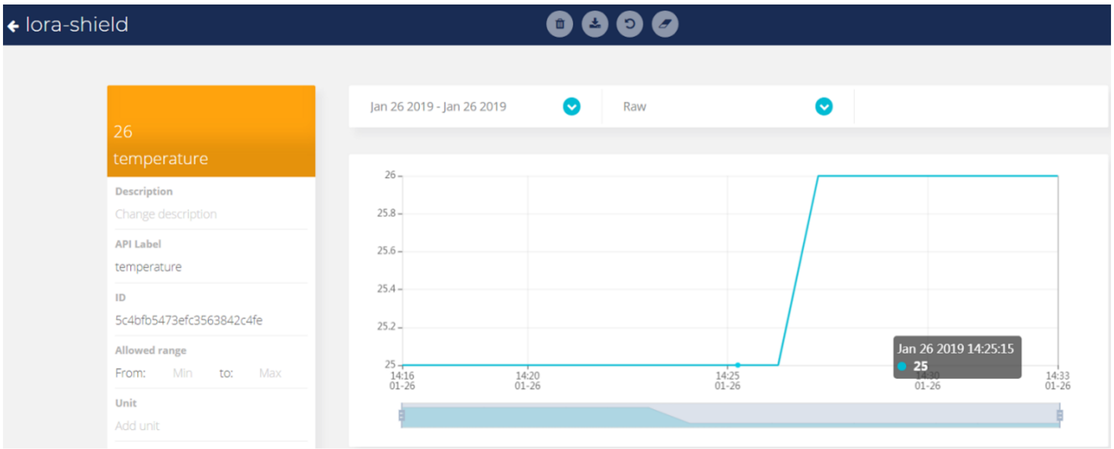
```
Start MQTT Example
LoRa init succeeded.
###########     COUNT=1     ###########
The temperature and humidity:
[24.00℃,38.00%]
The packet is send successful
###########     COUNT=2     ###########
The temperature and humidity:
[24.00℃,42.00%]
The packet is send successful
###########     COUNT=3     ###########
The temperature and humidity:
[24.00℃,40.00%]
The packet is send successful
```

And we can see the logread of gateway as below, means the packet arrive gateway:


10.130.1.1 - SecureCRT
文件(F) 编辑(E) 查看(V) 选项(O) 传输(T) 脚本(S) 工具(L) 帮助(H)
10.130.1.1

```
Sat Jan 26 06:31:10 2019 daemon.info lg01_pkt_fwd[12030]:
Sat Jan 26 06:31:10 2019 daemon.info lg01_pkt_fwd[12030]: RXTX~ Receive(HEX):3c353637383e7b2274656d7065726174757265223a32
362e302c2268756d6964697479223a33362e307d
Sat Jan 26 06:31:14 2019 user.notice root: [IoT.MQTT]:  Find Match Entry for 5678
Sat Jan 26 06:31:14 2019 user.notice root: [IoT.MQTT]:  [-t] /v1.6/devices/lora-shield
Sat Jan 26 06:31:14 2019 user.notice root: [IoT.MQTT]:  [-m] {"temperature":26.0,"humidity":36.0}
Sat Jan 26 06:31:32 2019 daemon.info lg01_pkt_fwd[12030]:
Sat Jan 26 06:31:32 2019 daemon.info lg01_pkt_fwd[12030]: RXTX~ Receive(HEX):3c353637383e7b2274656d7065726174757265223a32
362e302c2268756d6964697479223a33362e307d
Sat Jan 26 06:31:32 2019 user.notice root: [IoT.MQTT]:  Find Match Entry for 5678
Sat Jan 26 06:31:32 2019 daemon.info lg01_pkt_fwd[12030]:
Sat Jan 26 06:31:32 2019 daemon.info lg01_pkt_fwd[12030]: RXTX~ Receive(HEX):3c353637383e7b2274656d7065726174757265223a32
362e302c2268756d6964697479223a33362e307d
Sat Jan 26 06:31:32 2019 user.notice root: [IoT.MQTT]:  [-t] /v1.6/devices/lora-shield
Sat Jan 26 06:31:32 2019 user.notice root: [IoT.MQTT]:  [-m] {"temperature":26.0,"humidity":36.0}
Sat Jan 26 06:32:10 2019 daemon.info lg01_pkt_fwd[12030]:
Sat Jan 26 06:32:10 2019 daemon.info lg01_pkt_fwd[12030]: RXTX~ Receive(HEX):3c353637383e7b2274656d7065726174757265223a32362e302c2268756d696469
7479223a33372e307d
Sat Jan 26 06:32:14 2019 user.notice root: [IoT.MQTT]:  Find Match Entry for 5678
Sat Jan 26 06:32:14 2019 user.notice root: [IoT.MQTT]:  [-t] /v1.6/devices/lora-shield
Sat Jan 26 06:32:14 2019 user.notice root: [IoT.MQTT]:  [-m] {"temperature":26.0,"humidity":37.0}
Sat Jan 26 06:33:10 2019 daemon.info lg01_pkt_fwd[12030]:
Sat Jan 26 06:33:10 2019 daemon.info lg01_pkt_fwd[12030]: RXTX~ Receive(HEX):3c353637383e7b2274656d7065726174757265223a32362e302c2268756d696469
7479223a33382e307d
Sat Jan 26 06:33:14 2019 user.notice root: [IoT.MQTT]:  Find Match Entry for 5678
Sat Jan 26 06:33:14 2019 user.notice root: [IoT.MQTT]:  [-t] /v1.6/devices/lora-shield
Sat Jan 26 06:33:14 2019 user.notice root: [IoT.MQTT]:  [-m] {"temperature":26.0,"humidity":38.0}
Sat Jan 26 06:34:10 2019 daemon.info lg01_pkt_fwd[12030]:
Sat Jan 26 06:34:10 2019 daemon.info lg01_pkt_fwd[12030]: RXTX~ Receive(HEX):3c353637383e7b2274656d7065726174757265223a32352e302c2268756d696469
7479223a33382e307d
Sat Jan 26 06:34:14 2019 user.notice root: [IoT.MQTT]:  Find Match Entry for 5678
Sat Jan 26 06:34:14 2019 user.notice root: [IoT.MQTT]:  [-t] /v1.6/devices/lora-shield
Sat Jan 26 06:34:14 2019 user.notice root: [IoT.MQTT]:  [-m] {"temperature":25.0,"humidity":38.0}
Sat Jan 26 06:35:11 2019 daemon.info lg01_pkt_fwd[12030]:
Sat Jan 26 06:35:11 2019 daemon.info lg01_pkt_fwd[12030]: RXTX~ Receive(HEX):3c353637383e7b2274656d7065726174757265223a32352e302c2268756d6964697479223a33382e
307d
Sat Jan 26 06:35:14 2019 user.notice root: [IoT.MQTT]:  Find Match Entry for 5678
Sat Jan 26 06:35:14 2019 user.notice root: [IoT.MQTT]:  [-t] /v1.6/devices/lora-shield
Sat Jan 26 06:35:14 2019 user.notice root: [IoT.MQTT]:  [-m] {"temperature":25.0,"humidity":38.0}
```

Finally, we can see on the Ubidots:



26
temperature

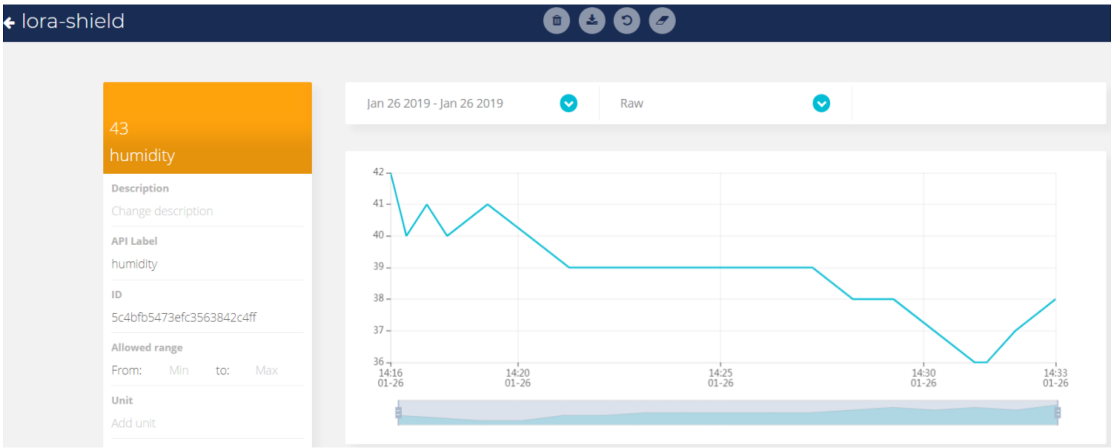
Description
Change description

API Label
temperature

ID
5c4bfb5473efc3563842c4fe

Allowed range
From: Min to: Max

Unit
Add unit



43
humidity

Description
Change description

API Label
humidity

ID
5c4bfb5473efc3563842c4ff

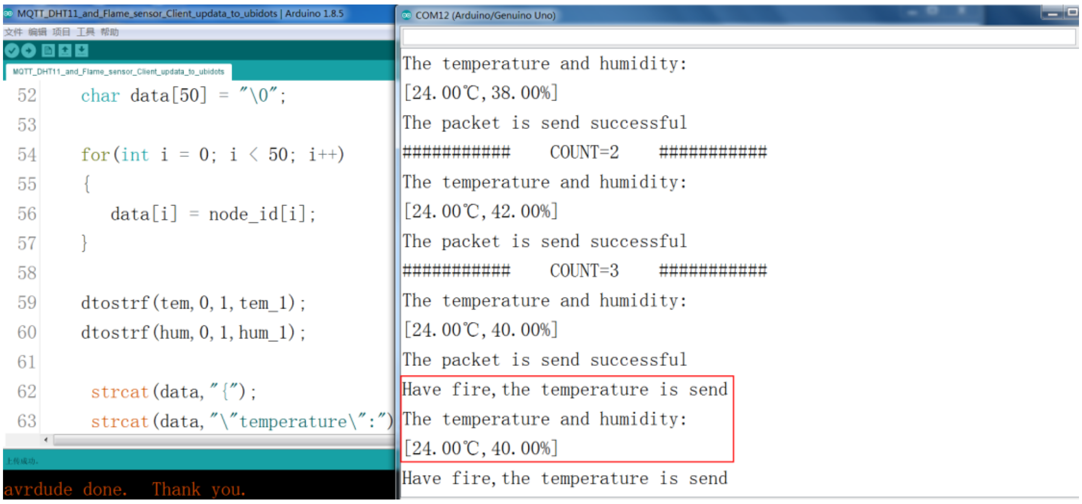
Allowed range
From: Min to: Max

Unit
Add unit

1.9.3 Test with interrupt by flame detect

The DO pin of Flame sensor is high in normal state. When a flame is detected, the DO pin of Flame sensor will become low, then, the UNO generates an external interrupt, and immediately uploads the temperature and humidity to the server.

The DO pin of Flame sensor is low when a flame is detected, and we can see on the Serial Monitor：

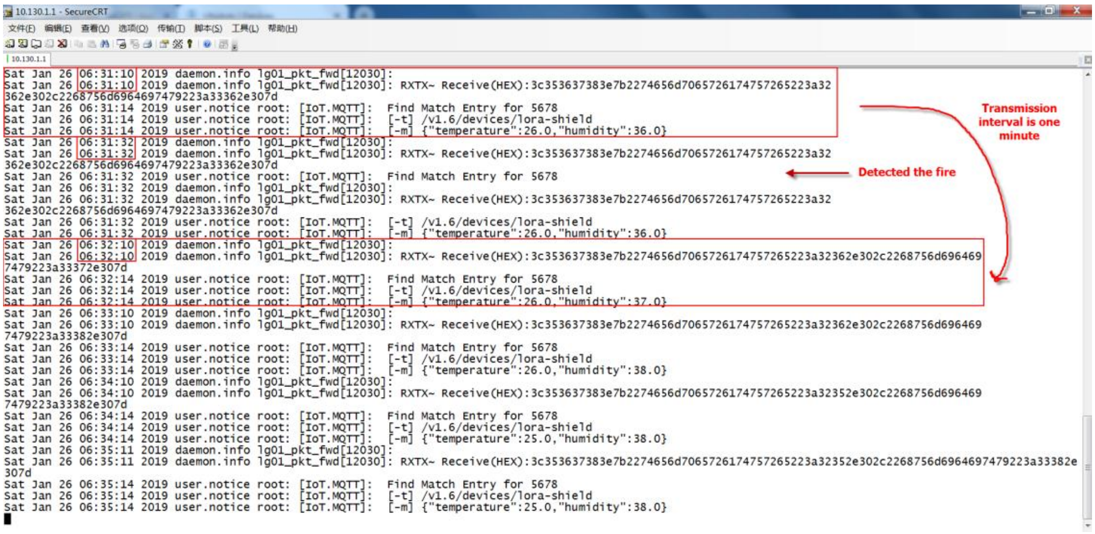

MQTT_DHT11_and_Flame_sensor_Client_updata_to_ubidots | Arduino 1.8.5
文件 编辑 项目 工具 帮助
MQTT_DHT11_and_Flame_sensor_Client_updata_to_ubidots
COM12 (Arduino/Genuino Uno)

```
  char data[50] = "\0";

  for(int i = 0; i < 50; i++)
  {
     data[i] = node_id[i];
  }

  dtostrf(tem,0,1,tem_1);
  dtostrf(hum,0,1,hum_1);

   strcat(data,"{");
   strcat(data,"\"temperature\":")
```

```
avrdude done.  Thank you.
```

```
The temperature and humidity:
[24.00℃,38.00%]
The packet is send successful
###########    COUNT=2    ###########
The temperature and humidity:
[24.00℃,42.00%]
The packet is send successful
###########    COUNT=3    ###########
The temperature and humidity:
[24.00℃,40.00%]
The packet is send successful
Have fire,the temperature is send
The temperature and humidity:
[24.00℃,40.00%]
Have fire,the temperature is send
```

Similarly, we can see the logread of gateway via SSH access:

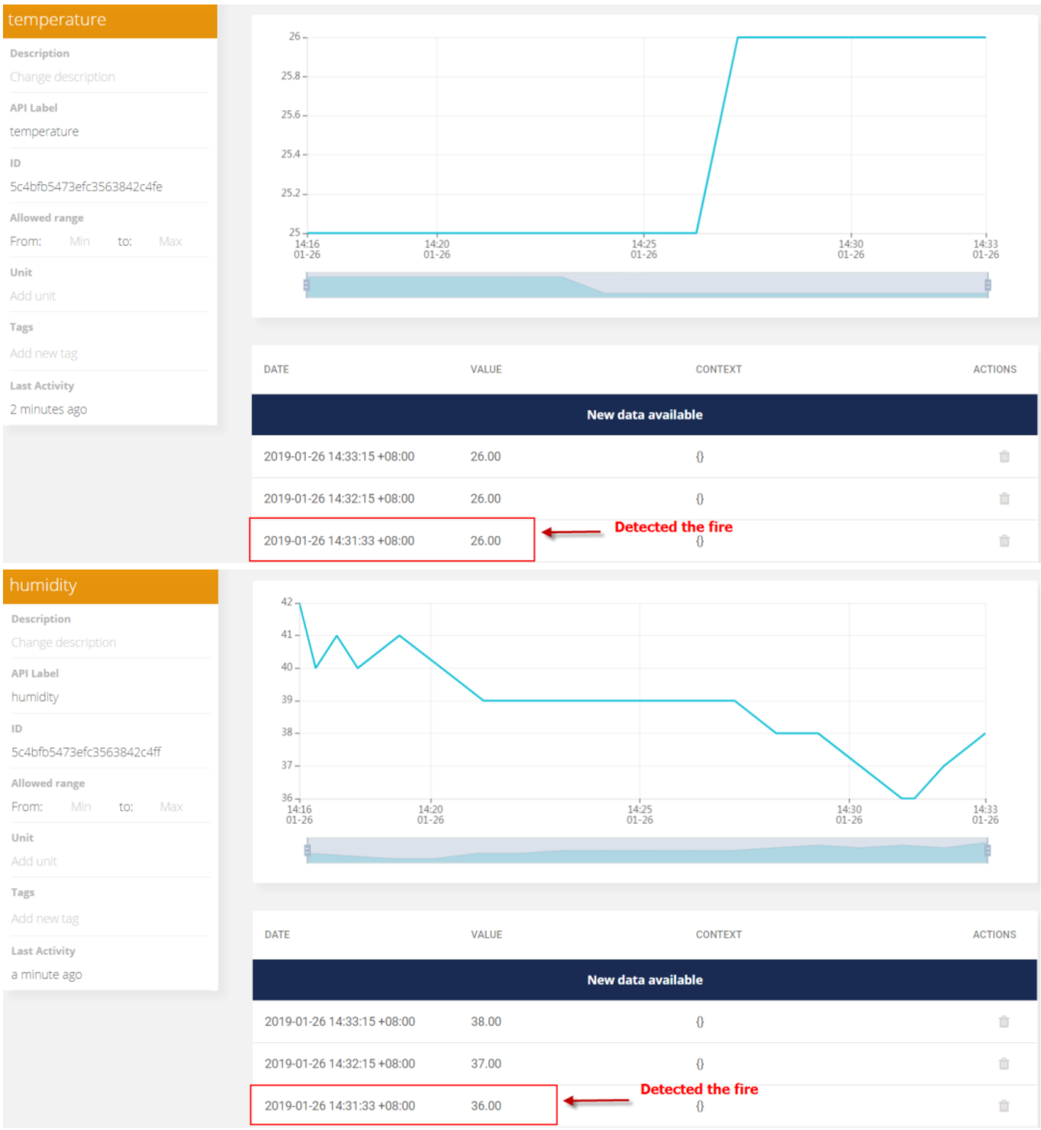

10.130.1.1 - SecureCRT
文件(F) 编辑(E) 查看(V) 选项(O) 传输(T) 脚本(S) 工具(L) 帮助(H)
10.130.1.1

```
Sat Jan 26 06:31:10 2019 daemon.info lg01_pkt_fwd[12030]:
Sat Jan 26 06:31:10 2019 daemon.info lg01_pkt_fwd[12030]: RXTX~ Receive(HEX):3c353637383e7b2274656d706572617475726522 3a32
362e302c2268756d6964697479223a33362e307d
Sat Jan 26 06:31:14 2019 user.notice root: [IoT.MQTT]:  Find Match Entry for 5678
Sat Jan 26 06:31:14 2019 user.notice root: [IoT.MQTT]:  [-t] /v1.6/devices/lora-shield
Sat Jan 26 06:31:14 2019 user.notice root: [IoT.MQTT]:  [-m] {"temperature":26.0,"humidity":36.0}
Sat Jan 26 06:31:32 2019 daemon.info lg01_pkt_fwd[12030]:
Sat Jan 26 06:31:32 2019 daemon.info lg01_pkt_fwd[12030]: RXTX~ Receive(HEX):3c353637383e7b2274656d7065726174757265223a32
362e302c2268756d6964697479223a33362e307d
Sat Jan 26 06:31:32 2019 user.notice root: [IoT.MQTT]:  Find Match Entry for 5678
Sat Jan 26 06:31:32 2019 daemon.info lg01_pkt_fwd[12030]:
Sat Jan 26 06:31:32 2019 daemon.info lg01_pkt_fwd[12030]: RXTX~ Receive(HEX):3c353637383e7b2274656d7065726174757265223a32
362e302c2268756d6964697479223a33362e307d
Sat Jan 26 06:31:32 2019 user.notice root: [IoT.MQTT]:  [-t] /v1.6/devices/lora-shield
Sat Jan 26 06:31:32 2019 user.notice root: [IoT.MQTT]:  [-m] {"temperature":26.0,"humidity":36.0}
Sat Jan 26 06:32:10 2019 daemon.info lg01_pkt_fwd[12030]:
Sat Jan 26 06:32:10 2019 daemon.info lg01_pkt_fwd[12030]: RXTX~ Receive(HEX):3c353637383e7b2274656d7065726174757265223a32362e302c2268756d696469
7479223a33372e307d
Sat Jan 26 06:32:14 2019 user.notice root: [IoT.MQTT]:  Find Match Entry for 5678
Sat Jan 26 06:32:14 2019 user.notice root: [IoT.MQTT]:  [-t] /v1.6/devices/lora-shield
Sat Jan 26 06:32:14 2019 user.notice root: [IoT.MQTT]:  [-m] {"temperature":26.0,"humidity":37.0}
Sat Jan 26 06:33:10 2019 daemon.info lg01_pkt_fwd[12030]:
Sat Jan 26 06:33:10 2019 daemon.info lg01_pkt_fwd[12030]: RXTX~ Receive(HEX):3c353637383e7b2274656d7065726174757265223a32362e302c2268756d696469
7479223a33382e307d
Sat Jan 26 06:33:14 2019 user.notice root: [IoT.MQTT]:  Find Match Entry for 5678
Sat Jan 26 06:33:14 2019 user.notice root: [IoT.MQTT]:  [-t] /v1.6/devices/lora-shield
Sat Jan 26 06:33:14 2019 user.notice root: [IoT.MQTT]:  [-m] {"temperature":26.0,"humidity":38.0}
Sat Jan 26 06:34:10 2019 daemon.info lg01_pkt_fwd[12030]:
Sat Jan 26 06:34:10 2019 daemon.info lg01_pkt_fwd[12030]: RXTX~ Receive(HEX):3c353637383e7b2274656d7065726174757265223a32352e302c2268756d696469
7479223a33382e307d
Sat Jan 26 06:34:14 2019 user.notice root: [IoT.MQTT]:  Find Match Entry for 5678
Sat Jan 26 06:34:14 2019 user.notice root: [IoT.MQTT]:  [-t] /v1.6/devices/lora-shield
Sat Jan 26 06:34:14 2019 user.notice root: [IoT.MQTT]:  [-m] {"temperature":25.0,"humidity":38.0}
Sat Jan 26 06:35:11 2019 daemon.info lg01_pkt_fwd[12030]:
Sat Jan 26 06:35:11 2019 daemon.info lg01_pkt_fwd[12030]: RXTX~ Receive(HEX):3c353637383e7b2274656d7065726174757265223a32352e302c2268756d6964697479223a33382e
307d
Sat Jan 26 06:35:14 2019 user.notice root: [IoT.MQTT]:  Find Match Entry for 5678
Sat Jan 26 06:35:14 2019 user.notice root: [IoT.MQTT]:  [-t] /v1.6/devices/lora-shield
Sat Jan 26 06:35:14 2019 user.notice root: [IoT.MQTT]:  [-m] {"temperature":25.0,"humidity":38.0}
```

Transmission interval is one minute

Detected the fire

Finally, we can see on the Ubidots:



temperature

Description
Change description

API Label
temperature

ID
5c4bfb5473efc3563842c4fe

Allowed range
From: Min to: Max

Unit
Add unit

Tags
Add new tag

Last Activity
2 minutes ago

DATE	VALUE	CONTEXT	ACTIONS
New data available			
2019-01-26 14:33:15 +08:00	26.00	{}	
2019-01-26 14:32:15 +08:00	26.00	{}	
2019-01-26 14:31:33 +08:00	26.00	{}	

Detected the fire

humidity

Description
Change description

API Label
humidity

ID
5c4bfb5473efc3563842c4ff

Allowed range
From: Min to: Max

Unit
Add unit

Tags
Add new tag

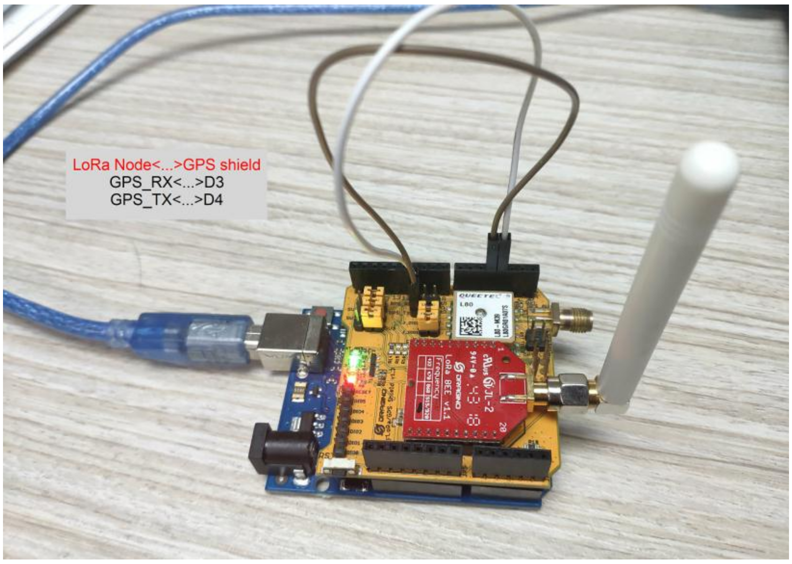
Last Activity
a minute ago

DATE	VALUE	CONTEXT	ACTIONS
New data available			
2019-01-26 14:33:15 +08:00	38.00	{}	
2019-01-26 14:32:15 +08:00	37.00	{}	
2019-01-26 14:31:33 +08:00	36.00	{}	

Detected the fire

1.10 Create LoRa GPS Shield End Node

1.10.1 Hardware Connection

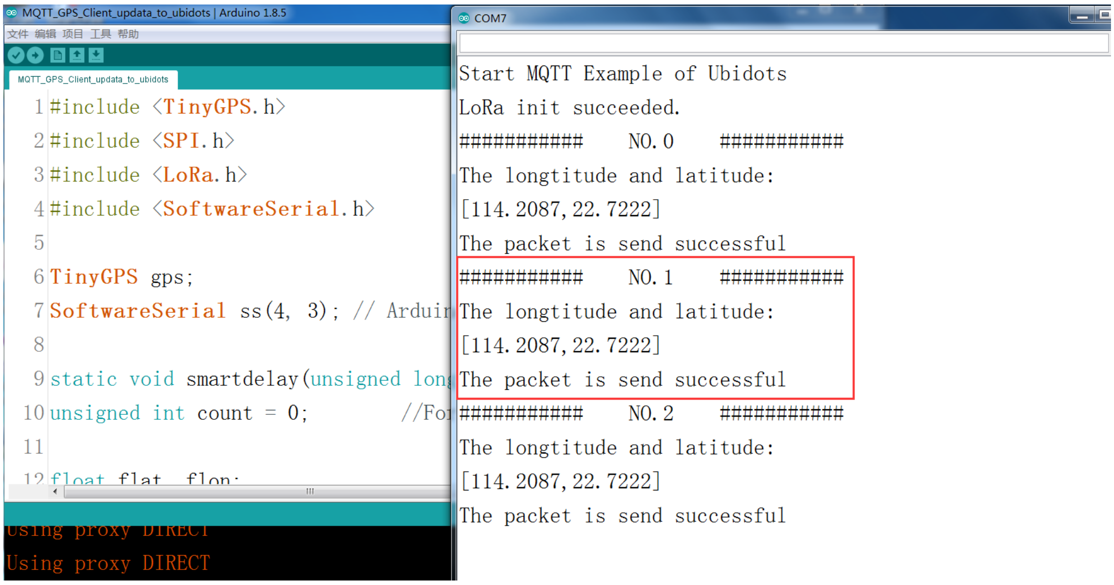


There is LoRa GPS Shield + UNO. Please use the connection as we show in the photo.

Upload this sketch to the UNO, this sketch will send position data to gateway at every 60 seconds.

1.10.2 Test with uplink

After we upload the sketch to UNO, we can see below output from Arduino

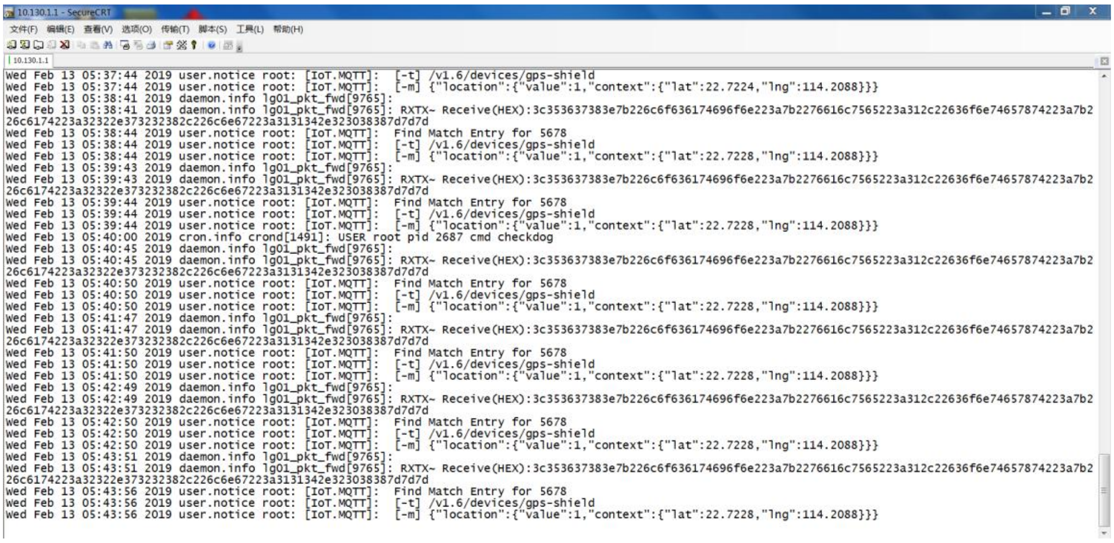

MQTT_GPS_Client_updata_to_ubidots | Arduino 1.8.5
文件 编辑 项目 工具 帮助
MQTT_GPS_Client_updata_to_ubidots
COM7

```
#include <TinyGPS.h>
#include <SPI.h>
#include <LoRa.h>
#include <SoftwareSerial.h>

TinyGPS gps;
SoftwareSerial ss(4, 3); // Arduin

static void smartdelay(unsigned long
unsigned int count = 0;        //Fo

float flat, flon;
```

```
Using proxy DIRECT
Using proxy DIRECT
```

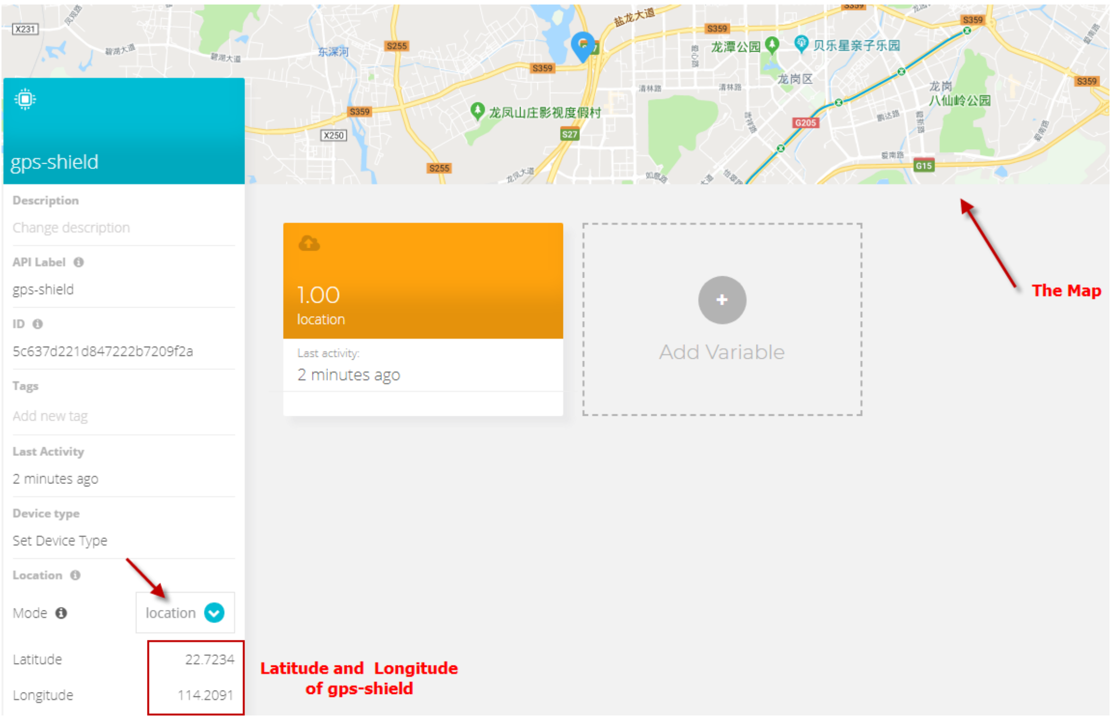
```
Start MQTT Example of Ubidots
LoRa init succeeded.
###########    NO.0    ###########
The longtitude and latitude:
[114.2087,22.7222]
The packet is send successful
###########    NO.1    ###########
The longtitude and latitude:
[114.2087,22.7222]
The packet is send successful
###########    NO.2    ###########
The longtitude and latitude:
[114.2087,22.7222]
The packet is send successful
```

And we can see the logread of gateway as below, means the packet arrive gateway:


10.130.1.1 - SecureCRT
文件(F) 编辑(E) 查看(V) 选项(O) 传输(T) 脚本(S) 工具(L) 帮助(H)
10.130.1.1

```
Wed Feb 13 05:37:44 2019 user.notice root: [IoT.MQTT]:  [-t] /v1.6/devices/gps-shield
Wed Feb 13 05:37:44 2019 user.notice root: [IoT.MQTT]:  [-m] {"location":{"value":1,"context":{"lat":22.7224,"lng":114.2088}}}
Wed Feb 13 05:38:41 2019 daemon.info lg01_pkt_fwd[9765]:
Wed Feb 13 05:38:41 2019 daemon.info lg01_pkt_fwd[9765]: RXTX~ Receive(HEX):3c353637383e7b226c6f636174696f6e223a7b2276616c7565223a312c22636f6e74657874223a7b2
26c6174223a32322e373232382c226c6e67223a3131342e323038387d7d7d
Wed Feb 13 05:38:44 2019 user.notice root: [IoT.MQTT]:  Find Match Entry for 5678
Wed Feb 13 05:38:44 2019 user.notice root: [IoT.MQTT]:  [-t] /v1.6/devices/gps-shield
Wed Feb 13 05:38:44 2019 user.notice root: [IoT.MQTT]:  [-m] {"location":{"value":1,"context":{"lat":22.7228,"lng":114.2088}}}
Wed Feb 13 05:39:43 2019 daemon.info lg01_pkt_fwd[9765]:
Wed Feb 13 05:39:43 2019 daemon.info lg01_pkt_fwd[9765]: RXTX~ Receive(HEX):3c353637383e7b226c6f636174696f6e223a7b2276616c7565223a312c22636f6e74657874223a7b2
26c6174223a32322e373232382c226c6e67223a3131342e323038387d7d7d
Wed Feb 13 05:39:44 2019 user.notice root: [IoT.MQTT]:  Find Match Entry for 5678
Wed Feb 13 05:39:44 2019 user.notice root: [IoT.MQTT]:  [-t] /v1.6/devices/gps-shield
Wed Feb 13 05:39:44 2019 user.notice root: [IoT.MQTT]:  [-m] {"location":{"value":1,"context":{"lat":22.7228,"lng":114.2088}}}
Wed Feb 13 05:40:00 2019 cron.info crond[1491]: USER root pid 2687 cmd checkdog
Wed Feb 13 05:40:45 2019 daemon.info lg01_pkt_fwd[9765]:
Wed Feb 13 05:40:45 2019 daemon.info lg01_pkt_fwd[9765]: RXTX~ Receive(HEX):3c353637383e7b226c6f636174696f6e223a7b2276616c7565223a312c22636f6e74657874223a7b2
26c6174223a32322e373232382c226c6e67223a3131342e323038387d7d7d
Wed Feb 13 05:40:50 2019 user.notice root: [IoT.MQTT]:  Find Match Entry for 5678
Wed Feb 13 05:40:50 2019 user.notice root: [IoT.MQTT]:  [-t] /v1.6/devices/gps-shield
Wed Feb 13 05:40:50 2019 user.notice root: [IoT.MQTT]:  [-m] {"location":{"value":1,"context":{"lat":22.7228,"lng":114.2088}}}
Wed Feb 13 05:41:47 2019 daemon.info lg01_pkt_fwd[9765]:
Wed Feb 13 05:41:47 2019 daemon.info lg01_pkt_fwd[9765]: RXTX~ Receive(HEX):3c353637383e7b226c6f636174696f6e223a7b2276616c7565223a312c22636f6e74657874223a7b2
26c6174223a32322e373232382c226c6e67223a3131342e323038387d7d7d
Wed Feb 13 05:41:50 2019 user.notice root: [IoT.MQTT]:  Find Match Entry for 5678
Wed Feb 13 05:41:50 2019 user.notice root: [IoT.MQTT]:  [-t] /v1.6/devices/gps-shield
Wed Feb 13 05:41:50 2019 user.notice root: [IoT.MQTT]:  [-m] {"location":{"value":1,"context":{"lat":22.7228,"lng":114.2088}}}
Wed Feb 13 05:42:49 2019 daemon.info lg01_pkt_fwd[9765]:
Wed Feb 13 05:42:49 2019 daemon.info lg01_pkt_fwd[9765]: RXTX~ Receive(HEX):3c353637383e7b226c6f636174696f6e223a7b2276616c7565223a312c22636f6e74657874223a7b2
26c6174223a32322e373232382c226c6e67223a3131342e323038387d7d7d
Wed Feb 13 05:42:50 2019 user.notice root: [IoT.MQTT]:  Find Match Entry for 5678
Wed Feb 13 05:42:50 2019 user.notice root: [IoT.MQTT]:  [-t] /v1.6/devices/gps-shield
Wed Feb 13 05:42:50 2019 user.notice root: [IoT.MQTT]:  [-m] {"location":{"value":1,"context":{"lat":22.7228,"lng":114.2088}}}
Wed Feb 13 05:43:51 2019 daemon.info lg01_pkt_fwd[9765]:
Wed Feb 13 05:43:51 2019 daemon.info lg01_pkt_fwd[9765]: RXTX~ Receive(HEX):3c353637383e7b226c6f636174696f6e223a7b2276616c7565223a312c22636f6e74657874223a7b2
26c6174223a32322e373232382c226c6e67223a3131342e323038387d7d7d
Wed Feb 13 05:43:56 2019 user.notice root: [IoT.MQTT]:  Find Match Entry for 5678
Wed Feb 13 05:43:56 2019 user.notice root: [IoT.MQTT]:  [-t] /v1.6/devices/gps-shield
Wed Feb 13 05:43:56 2019 user.notice root: [IoT.MQTT]:  [-m] {"location":{"value":1,"context":{"lat":22.7228,"lng":114.2088}}}
```

Finally, we can see on the Ubidots:



Description
Change description

API Label
gps-shield

ID
5c637d221d847222b7209f2a

Tags
Add new tag

Last Activity
2 minutes ago

Device type
Set Device Type

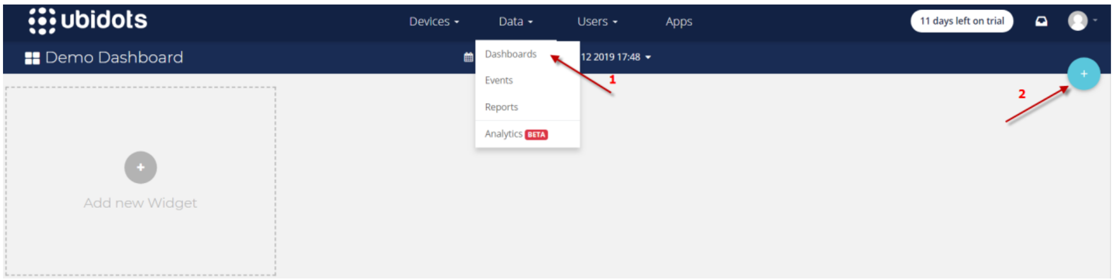
Location

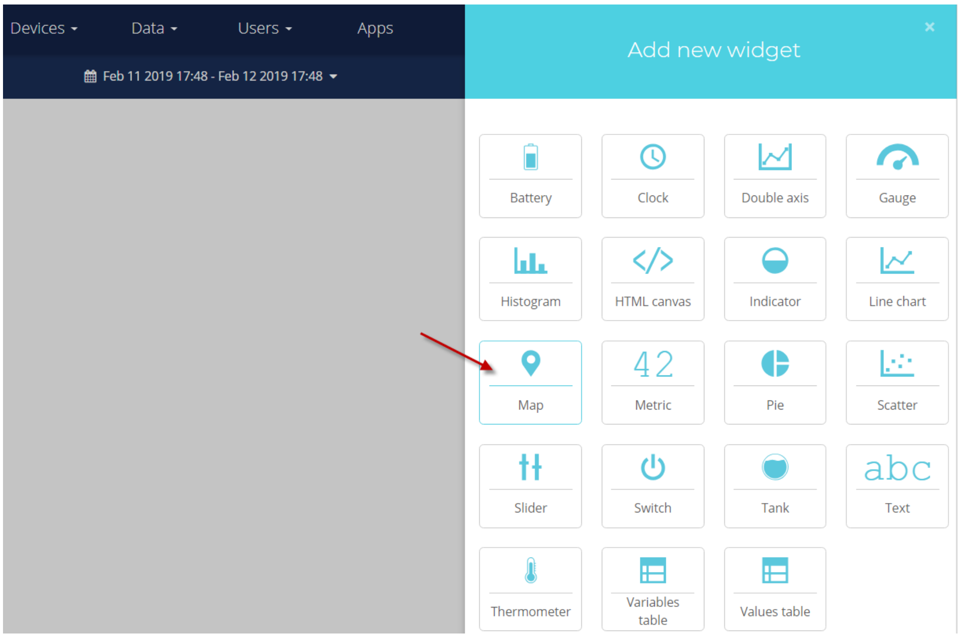
Mode: location

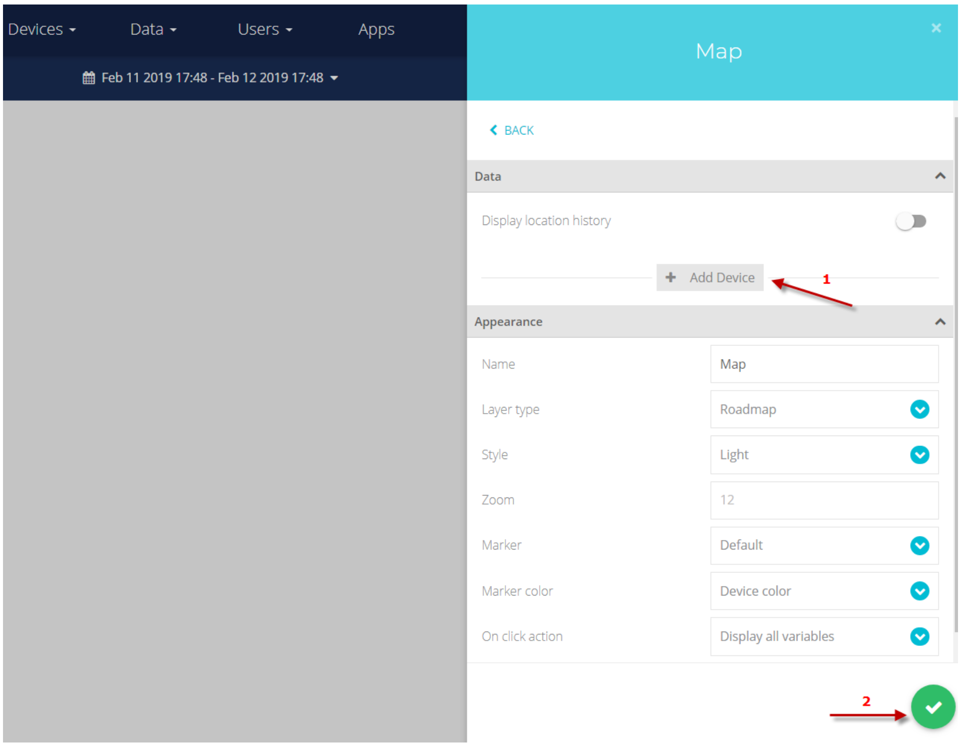
Latitude: 22.7234

Longitude: 114.2091

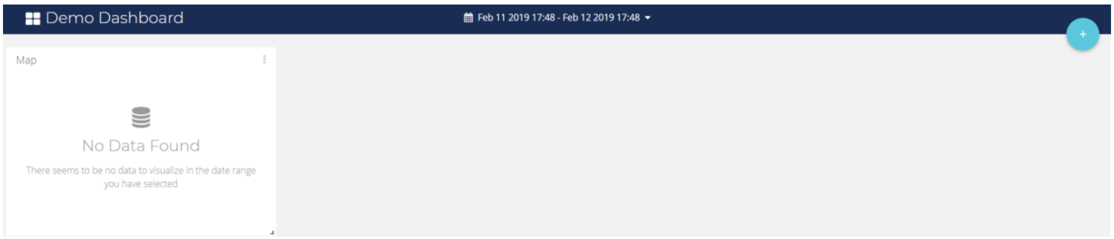
1.10.3 Create Map Widgets in Ubidots







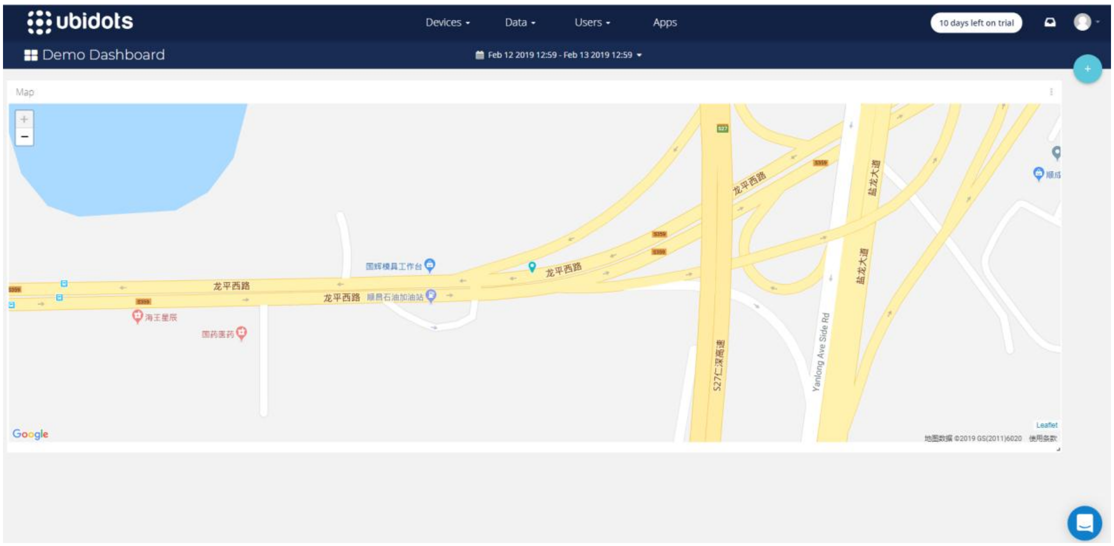
Name	Map
Layer type	Roadmap
Style	Light
Zoom	12
Marker	Default
Marker color	Device color
On click action	Display all variables



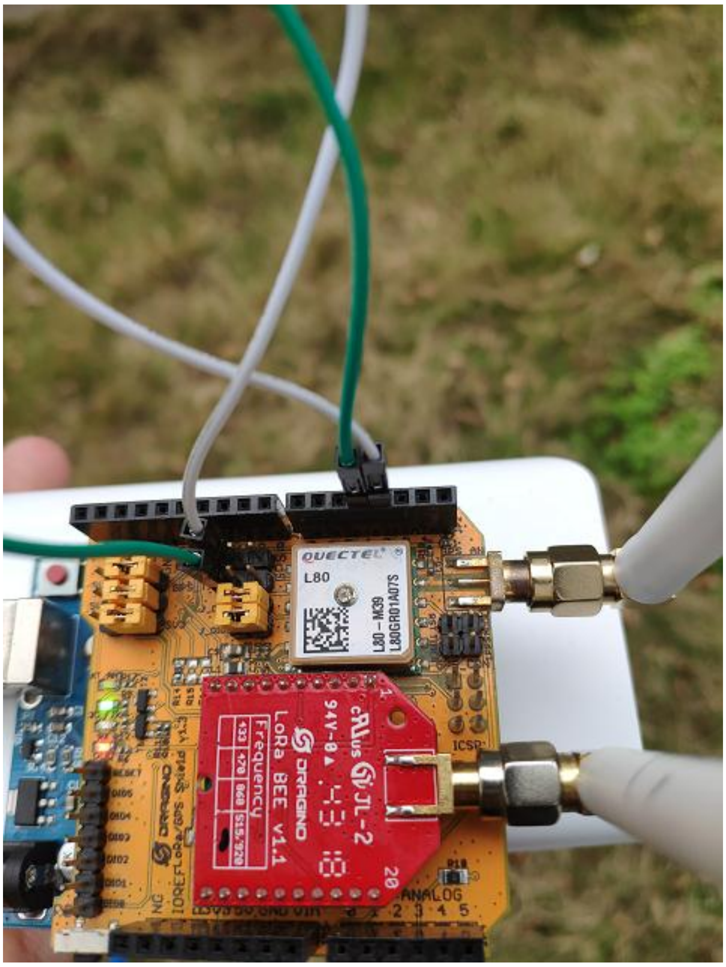
No Data Found

There seems to be no data to visualize in the date range you have selected

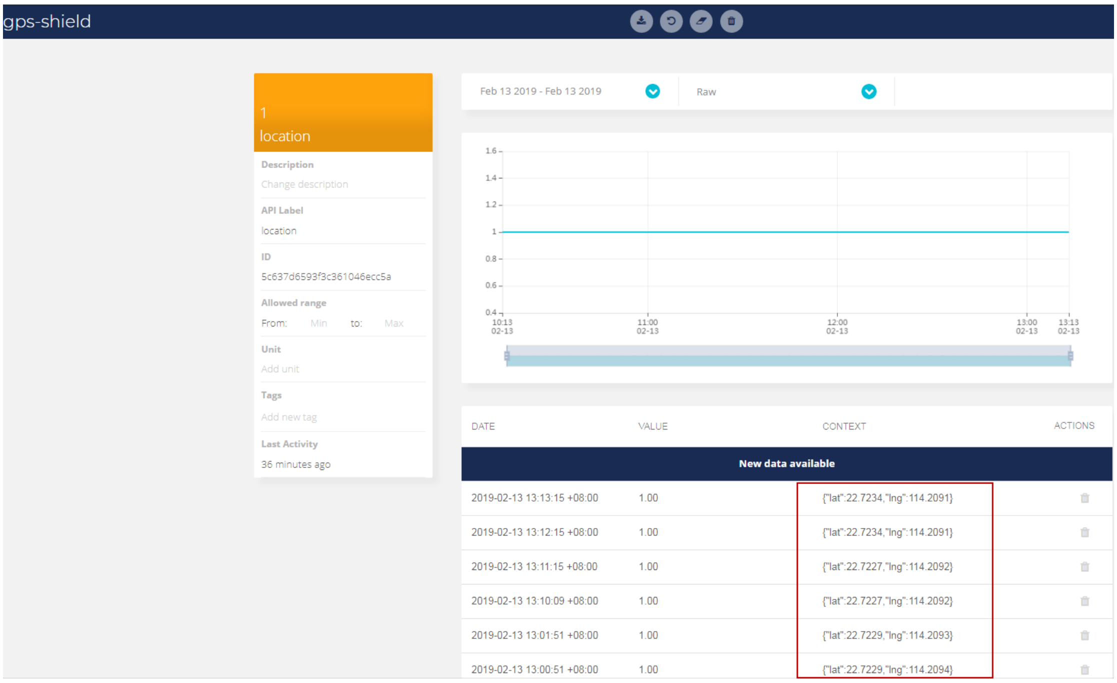
Finally,We can see on the Dashboard when the device is successfully located and successfully published by MQTT:



1.10.4 Moving LoRa GPS-shield outdoors



We can see the position Dashboard when the device is successfully located and successfully published by MQTT:



gps-shield

1
location

Description
Change description

API Label
location

ID
5c637d6593f3c361046ecc5a

Allowed range
From: Min to: Max

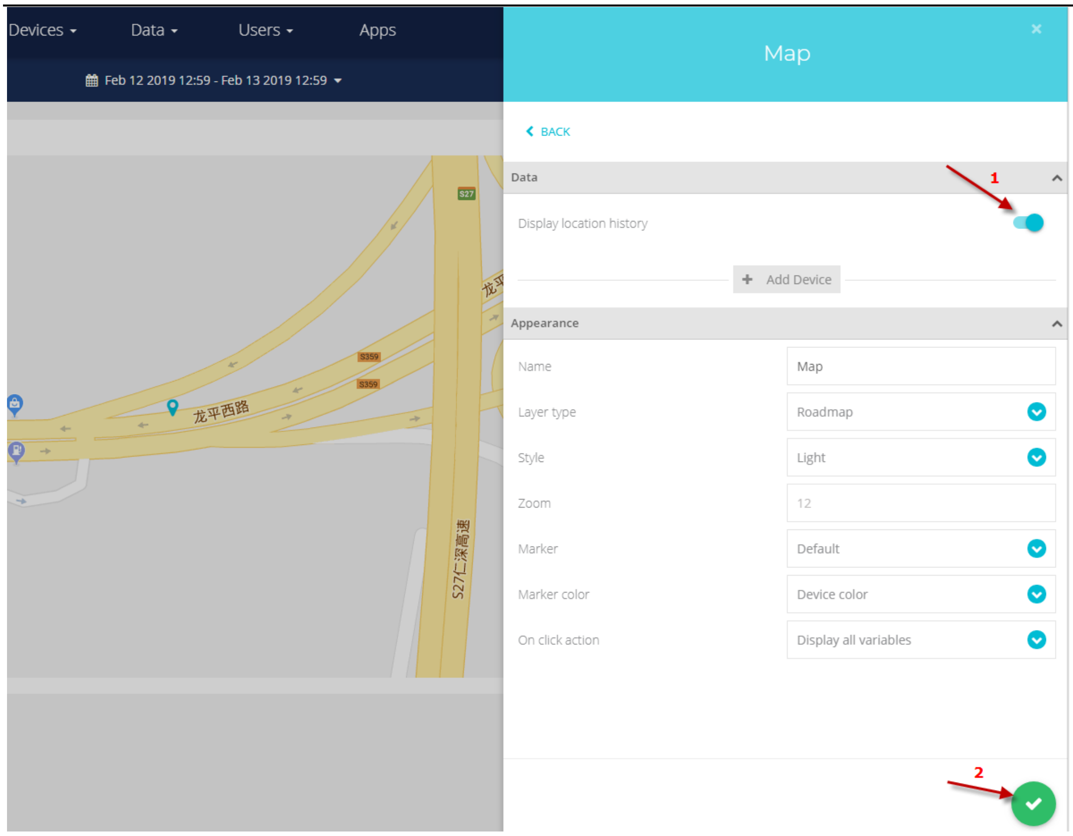
Unit
Add unit

Tags
Add new tag

Last Activity
36 minutes ago

Feb 13 2019 - Feb 13 2019 | Raw

DATE	VALUE	CONTEXT	ACTIONS
New data available			
2019-02-13 13:13:15 +08:00	1.00	{"lat":22.7234,"lng":114.2091}	
2019-02-13 13:12:15 +08:00	1.00	{"lat":22.7234,"lng":114.2091}	
2019-02-13 13:11:15 +08:00	1.00	{"lat":22.7227,"lng":114.2092}	
2019-02-13 13:10:09 +08:00	1.00	{"lat":22.7227,"lng":114.2092}	
2019-02-13 13:01:51 +08:00	1.00	{"lat":22.7229,"lng":114.2093}	
2019-02-13 13:00:51 +08:00	1.00	{"lat":22.7229,"lng":114.2094}	



Data

Display location history

Add Device

Appearance

Name: Map

Layer type: Roadmap

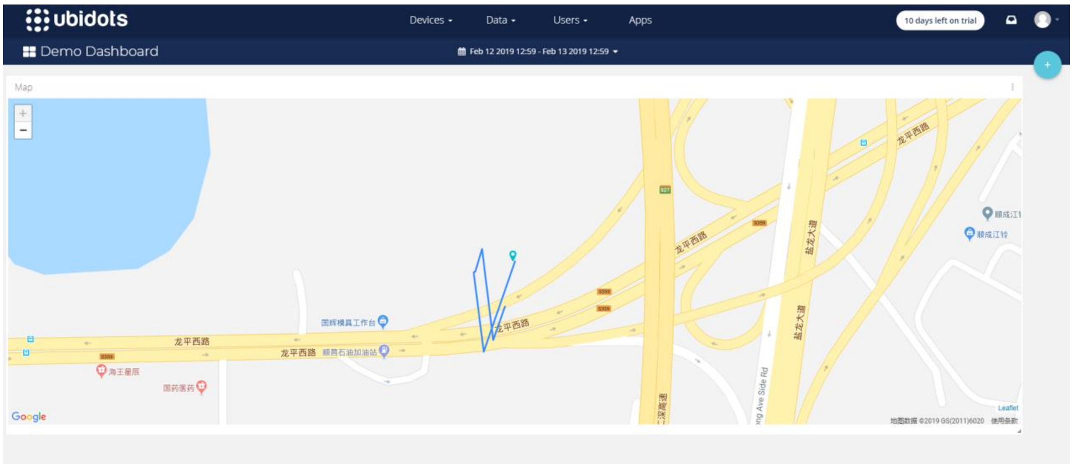
Style: Light

Zoom: 12

Marker: Default

Marker color: Device color

On click action: Display all variables



2 Reference

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

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

✧ Ubidot Server
industrial.ubidots.com