Using an Arduino with the LoRa IoT Development Kit

Aims:

- To connect an Arduino to The Things Network using the LoRa IoT development kit.
- To send sensor data to the server gateway using LoRa technology.

Research

Tutorial url: http://www.dragino.com/downloads/index.php?dir=LoRa_IoT_Kit/v2-Kit/
(Select Single Channel LoRa IoT Kit v2 User Manual_v1.0.5.pdf)

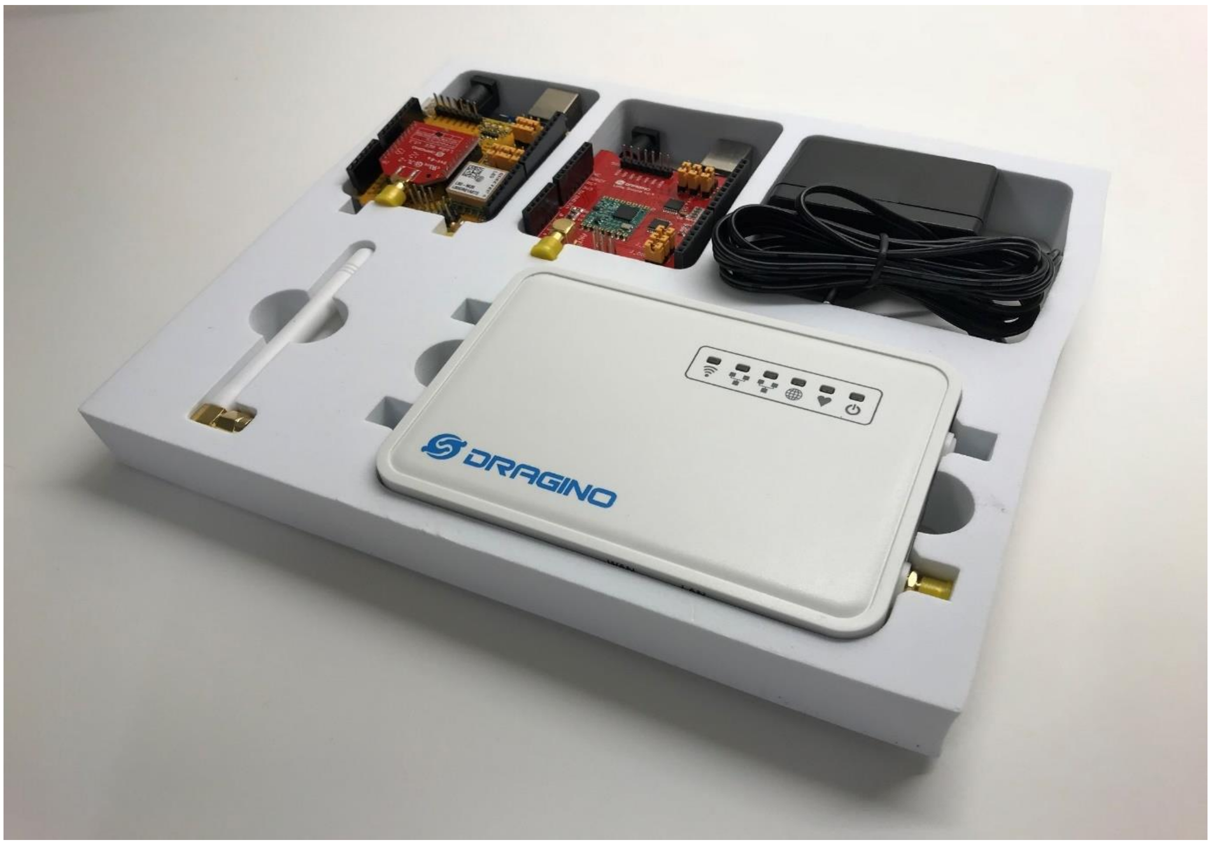
LoRa is a Long-Range radio technology developed by Semtech. It allows Internet of Things sensors to send data to remote locations. The radio network we will be using is the Things Network https://www.thethingsnetwork.org/docs/lorawan/. Use of this system is free.

The Dragino transmitters use RFM96W transceiver modules made by HOPERF. https://www.hoperf.com/modules/lora/RFM95.html. There is a link on the web page to download the datasheet in .PDF format.

Equipment

My shopping list comprised the following items:

• UKIoT Store: LoRa IoT Development Kit https://www.ukiot.store/product/lora-iot-development-kit/	£121.89 (Inc VAT)
• Amazon: Travel Adaptor USA visitor to UK (to power gateway) https://www.amazon.co.uk/s?k=travel+adapter+usa+to+uk&ref=nb_sb_noss_1	£3 – 6
• You will also need a RJ45 ethernet cable. This is not supplied in the kit. https://www.amazon.co.uk/s?k=RJ45+cable&ref=nb_sb_noss	



Already in stock:

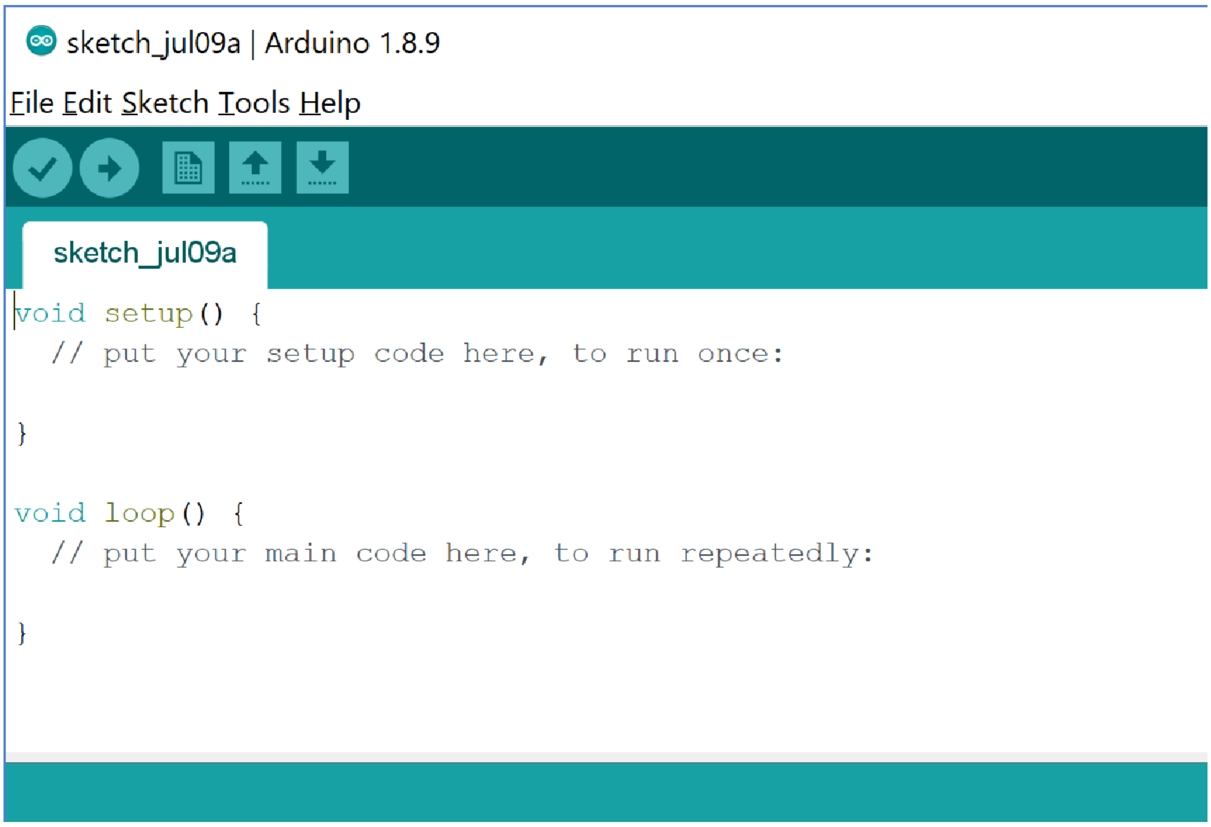
- Solderless breadboard
- Assorted electrical components
- Plastic sandwich box (to contain the electronics)
- Cable ties (to secure the electronics in the sandwich box)

Procedure

These instructions are based on the tutorial referred to above.

In order to program the Arduino, you must first download and install the Arduino IDE (integrated development environment) onto your laptop. This is available here: https://www.arduino.cc/en/Main/Software. From the list on the left of the page, select the 'Windows Installer, for Windows XP and up' option. I am running Windows 10. The file I downloaded was called 'arduino-1.8.9-windows.exe'.

Once downloaded, double click the .exe file and follow the installation instructions. If all goes according to plan, you should see this when you open the IDE.



```
void setup() {
  // put your setup code here, to run once:

}

void loop() {
  // put your main code here, to run repeatedly:

}
```

To quote the tutorial "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 theArduino IDE program it via USB."

"UART stands for Universal Asynchronous Receiver/Transmitter. It's not a communication protocol like SPI and I2C, but a physical circuit in a microcontroller, or a stand-alone IC. A UART's main purpose is to transmit and receive serial data." http://www.circuitbasics.com/basics-uart-communication/

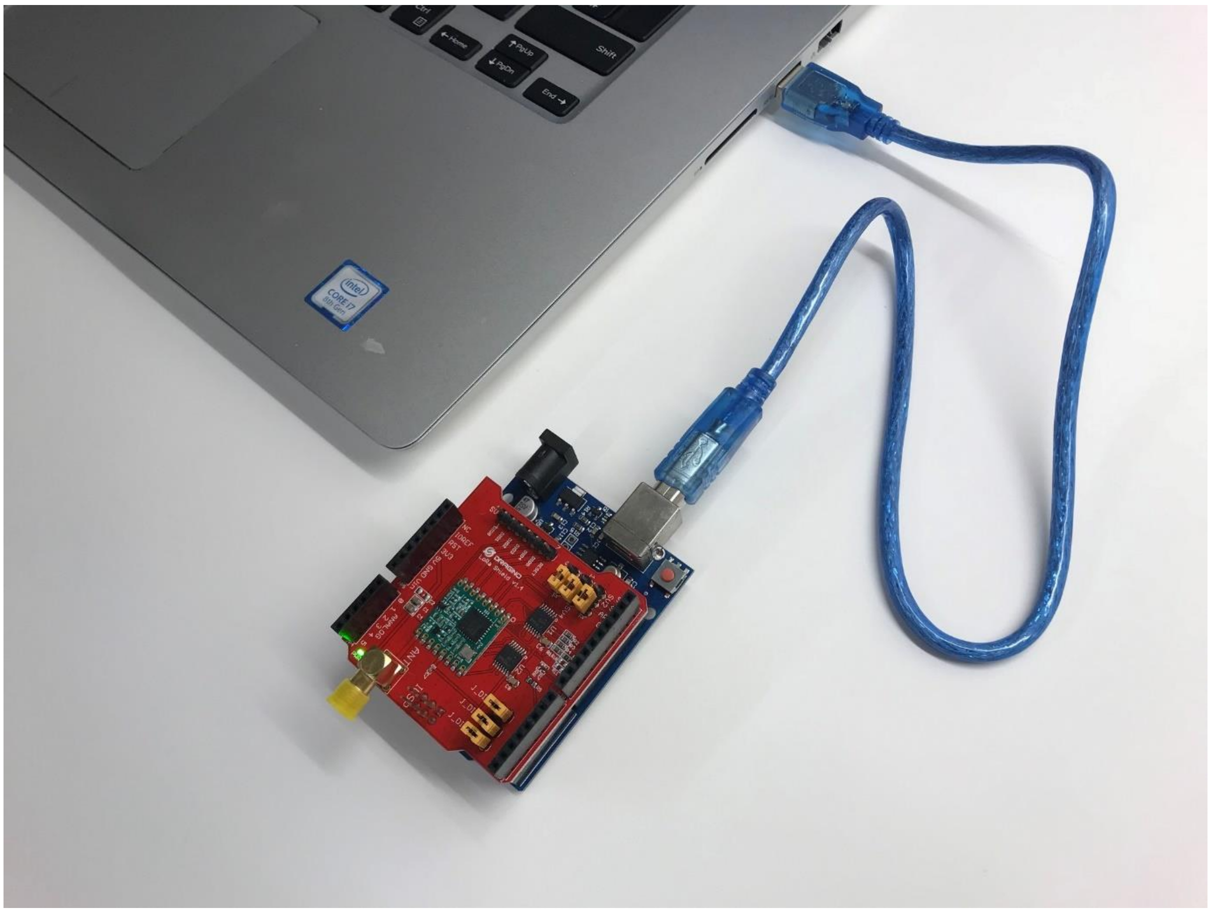
The driver can be downloaded here: https://sparks.gogo.co.nz/ch340.html. The instructions on this website read as follows:

- Download the Windows CH340 Driver
- Unzip the file
- Run the installer which you unzipped
- In the Arduino IDE when the CH340 is connected you will see a COM Port in the Tools > Serial Port menu, the COM number for your device may vary depending on your system.

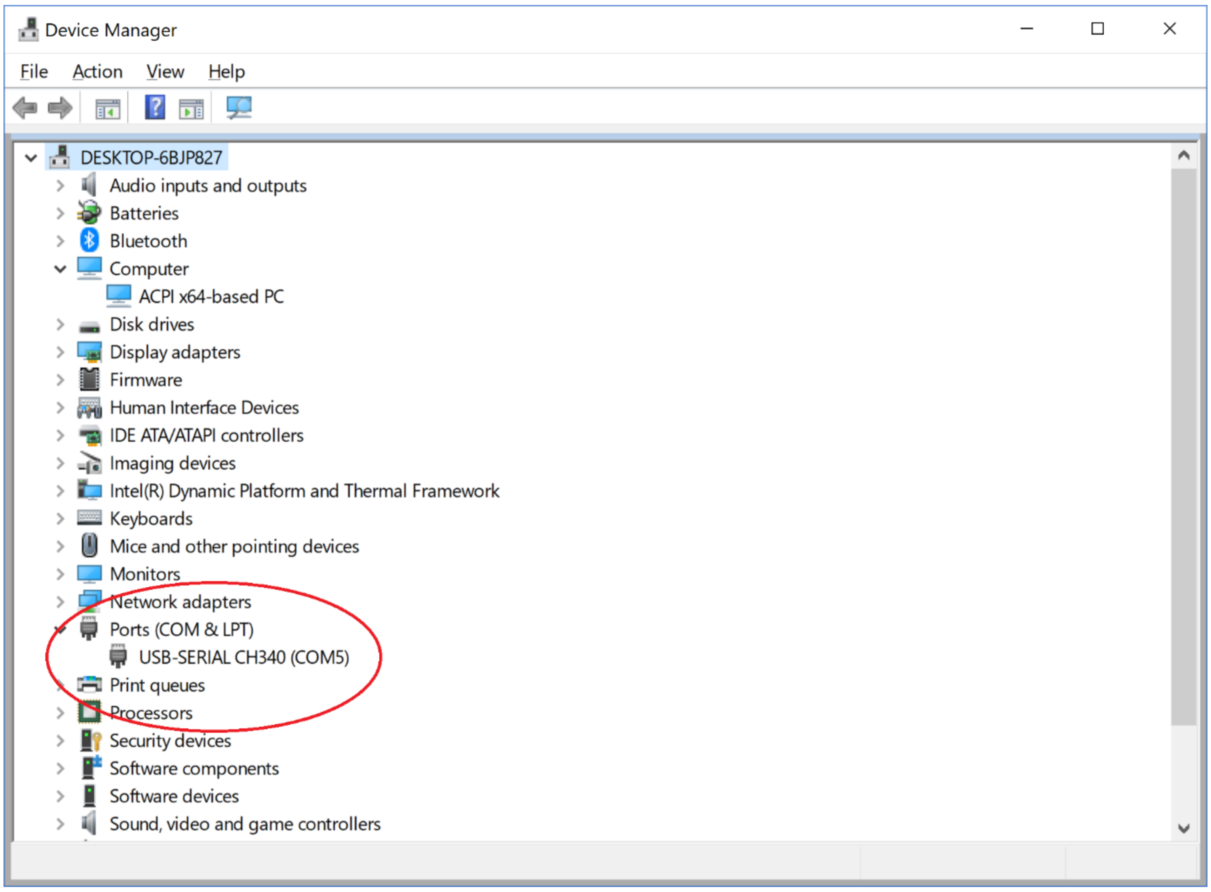
To check this is working, click on the Windows Start button (by default this is at the bottom left part of the desktop screen), and type 'device manager'. Click on the link to the contol panel Device Manager.

All the instructions below will also work with a genuine Arduino Uno, but check the port number in the Device Manager. In my setup, the genuine Arduino uses port 4, while the clone supplied with the development kit uses port 5.

You will not see the COM port unless you have an Arduino plugged in to your computer. I plugged in the Arduino UNO with the LoRa shield initially. My setup looks like this (the cable comes with the kit):



Once you have plugged in the Arduino, look down the list of devices that appear in the Device Manager. The entry you are looking for is USB-SERIAL CH340 (COM*). The * in (COM*) will be a number. This number will vary depending on your computer.

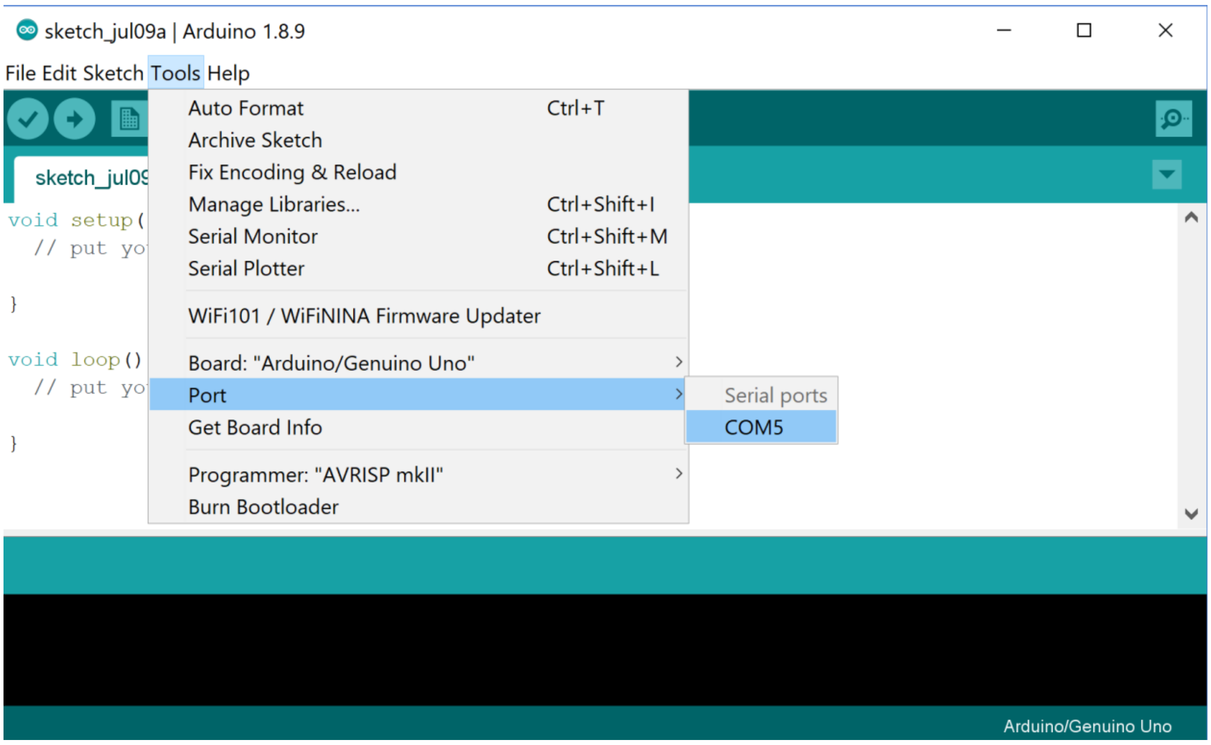


Device Manager

File Action View Help

DESKTOP-6BJP827
- Audio inputs and outputs
- Batteries
- Bluetooth
- Computer
  - ACPI x64-based PC
- Disk drives
- Display adapters
- Firmware
- Human Interface Devices
- IDE ATA/ATAPI controllers
- Imaging devices
- Intel(R) Dynamic Platform and Thermal Framework
- Keyboards
- Mice and other pointing devices
- Monitors
- Network adapters
- Ports (COM & LPT)
  - USB-SERIAL CH340 (COM5)
- Print queues
- Processors
- Security devices
- Software components
- Software devices
- Sound, video and game controllers

Return to the Arduino IDE and click on the ‘Tools’ menu. From the dropdown menu that appears, hover over ‘board’ and select Arduino/Genuino Uno from the ‘Boards Manager’ flyout that appears. If you hover over ‘Port’ (below ‘Board’), you should see a ‘Serial Ports’ flyout with your COM* listed.



sketch_jul09a | Arduino 1.8.9

File Edit Sketch Tools Help

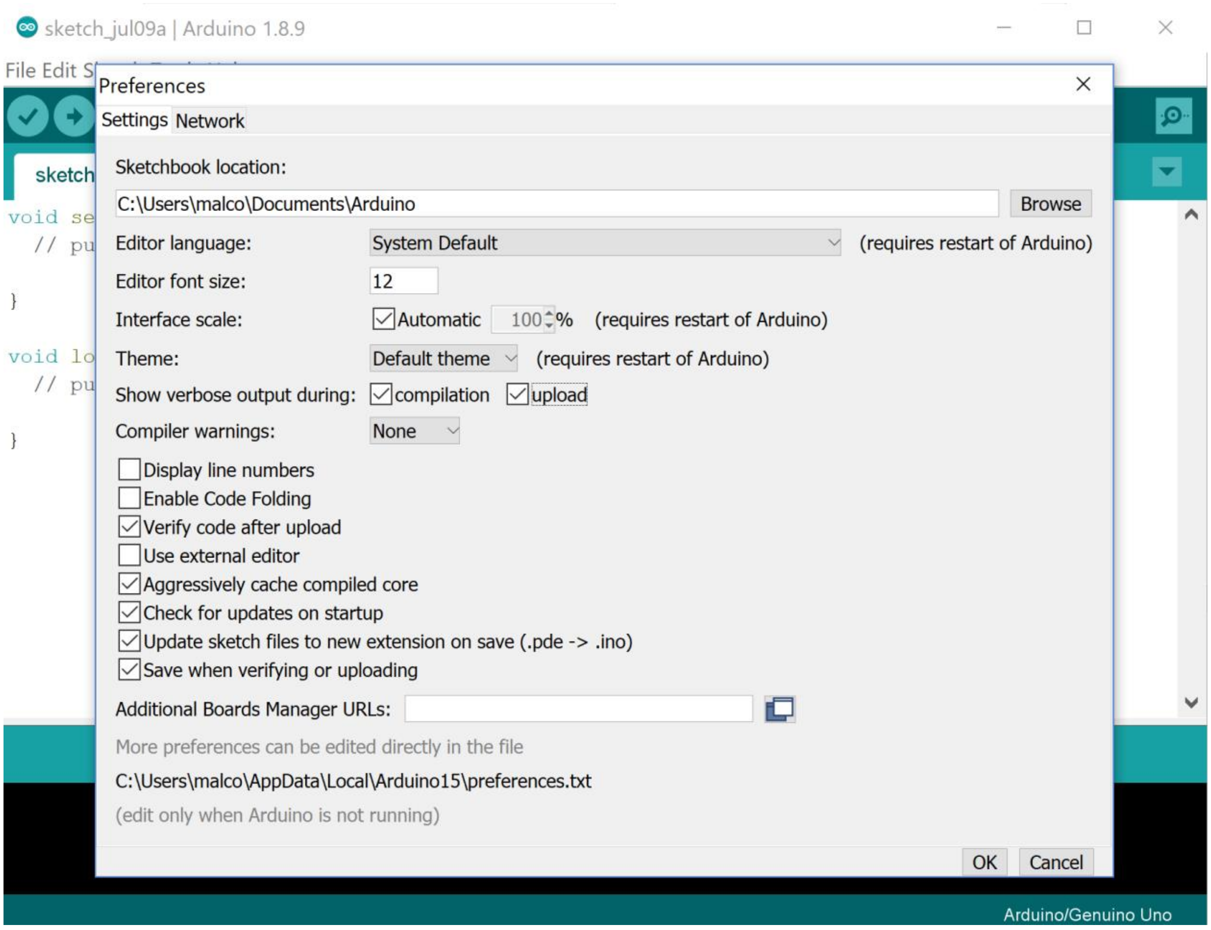
Auto Format	Ctrl+T
Archive Sketch	
Fix Encoding & Reload	
Manage Libraries...	Ctrl+Shift+I
Serial Monitor	Ctrl+Shift+M
Serial Plotter	Ctrl+Shift+L
WiFi101 / WiFiNINA Firmware Updater	
Board: "Arduino/Genuino Uno"	
Port	
Get Board Info	
Programmer: "AVRISP mkII"	
Burn Bootloader	

Serial ports
COM5

```
void setup(
  // put yo

}

void loop()
  // put yo

}
```

Arduino/Genuino Uno

To help with trouble shooting, it may be a good idea to turn on 'verbose output' (so you can read error messages). Select 'File', then 'Preferences' and tick the two check boxes next to 'Show verbose output during:'. Click OK.


sketch_jul09a | Arduino 1.8.9

Preferences

Settings Network

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

Editor language: System Default (requires restart of Arduino)

Editor font size: 12

Interface scale: [✓] Automatic 100 % (requires restart of Arduino)

Theme: Default theme (requires restart of Arduino)

Show verbose output during: [✓] compilation [✓] upload

Compiler warnings: None

- [ ] Display line numbers
- [ ] Enable Code Folding
- [✓] Verify code after upload
- [ ] Use external editor
- [✓] Aggressively cache compiled core
- [✓] Check for updates on startup
- [✓] Update sketch files to new extension on save (.pde -> .ino)
- [✓] Save when verifying or uploading

Additional Boards Manager URLs:

More preferences can be edited directly in the file

C:\Users\malco\AppData\Local\Arduino15\preferences.txt

(edit only when Arduino is not running)

OK Cancel

Arduino/Genuino Uno

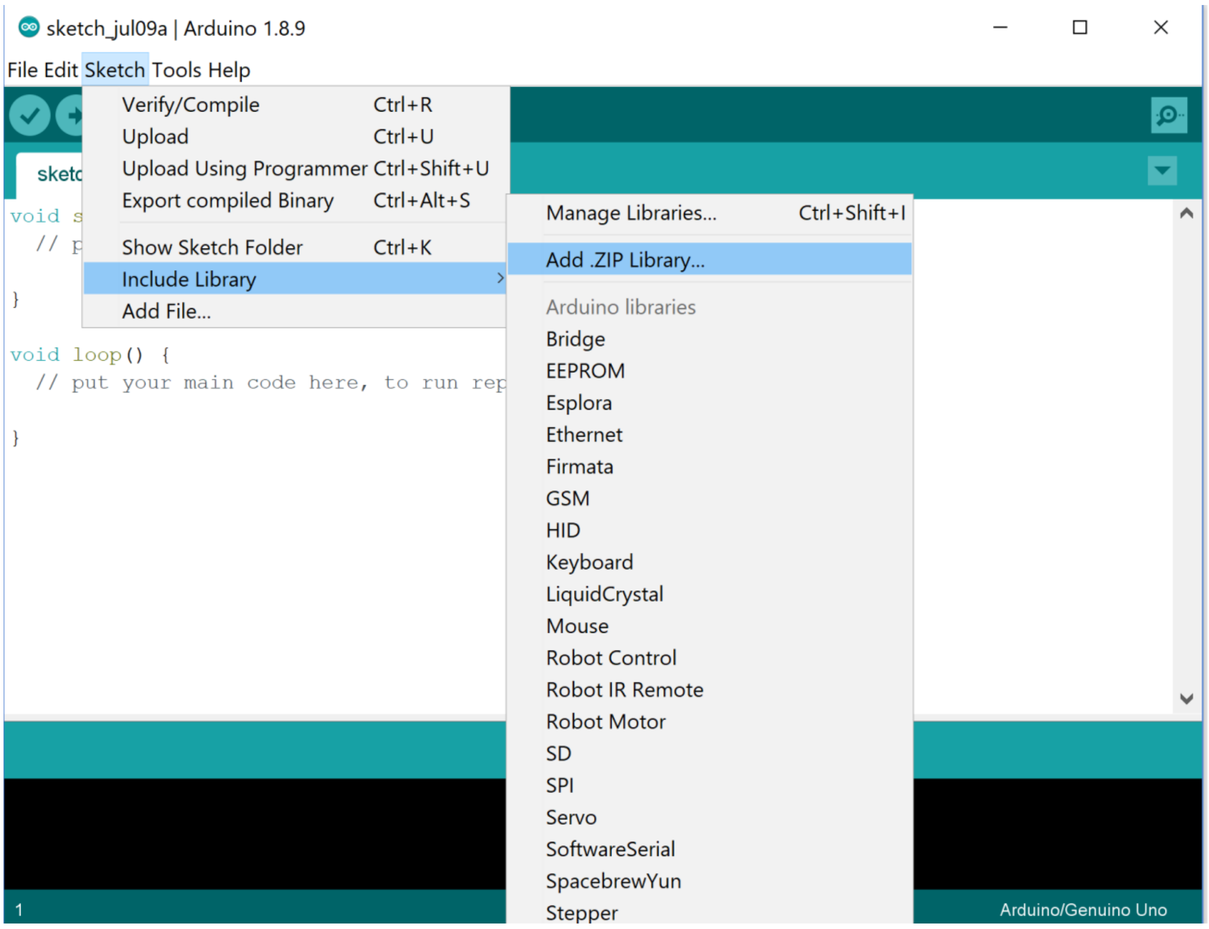
Before we can program the Arduino for use with the Internet of Things, we need to download some libraries. Libraries are blocks of pre-built code that we can use in our scripts.

The Arduino-LMIC library

The first one we will download is the Arduino-LMIC library. This is used to configure the Arduino as a standard LoRaWAN end node. A 'node' is the end part of the Internet of Things that connects to the things, such as sensors.

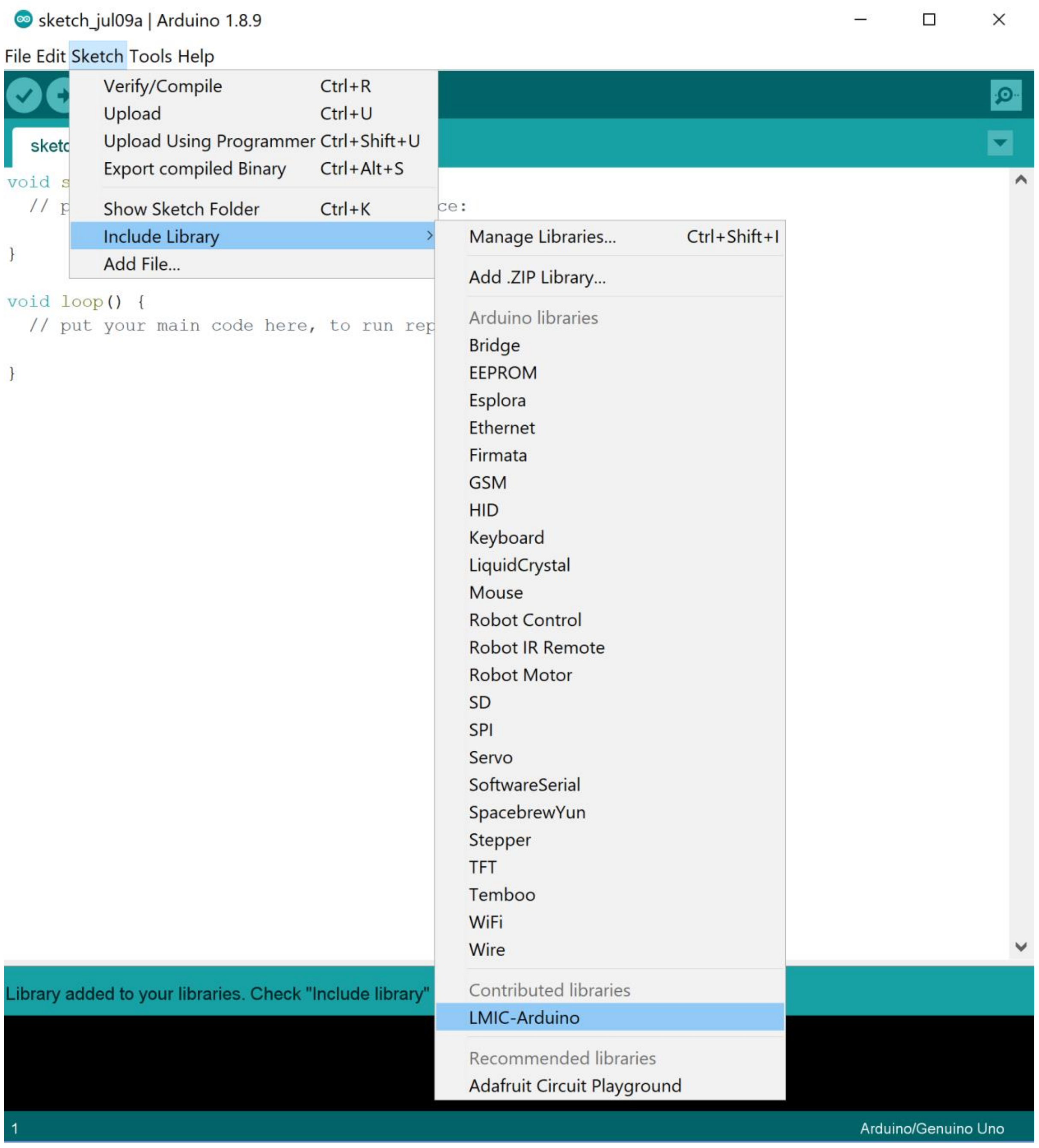
The library is available on GitHub: https://github.com/dragino/arduino-lmic. Once you are on the GitHub page, click 'Clone or download' (the green button), then click 'Download ZIP'. Save the file somewhere you can find it easily. You can read more about the usefulness of this library on the GitHub page.

To install the library in the Arduino IDE, we will use the Arduino Library Manager.



```
void s
  // p

}

void loop() {
  // put your main code here, to run rep

}
```

In the IDE, click ‘Sketch’, then ‘Include Library’. In the flyout that appears, click ‘Add ZIP Library…’, then browse for the ZIP file you just downloaded. Then click on the file, then click ‘Open’. To check the library has been installed, click ‘Sketch’, then ‘Include Library’. In the flyout that appears, in the section headed ‘Contributed libraries’, you should see the library ‘LMIC-Arduino’.



```
void s
  // p                                          ce:

}

void loop() {
  // put your main code here, to run rep

}
```

```
Library added to your libraries. Check "Include library"
```

Add the LoRa send and receive library

We need an Arduino library for sending and receiving data using LoRa radios. Get it here:

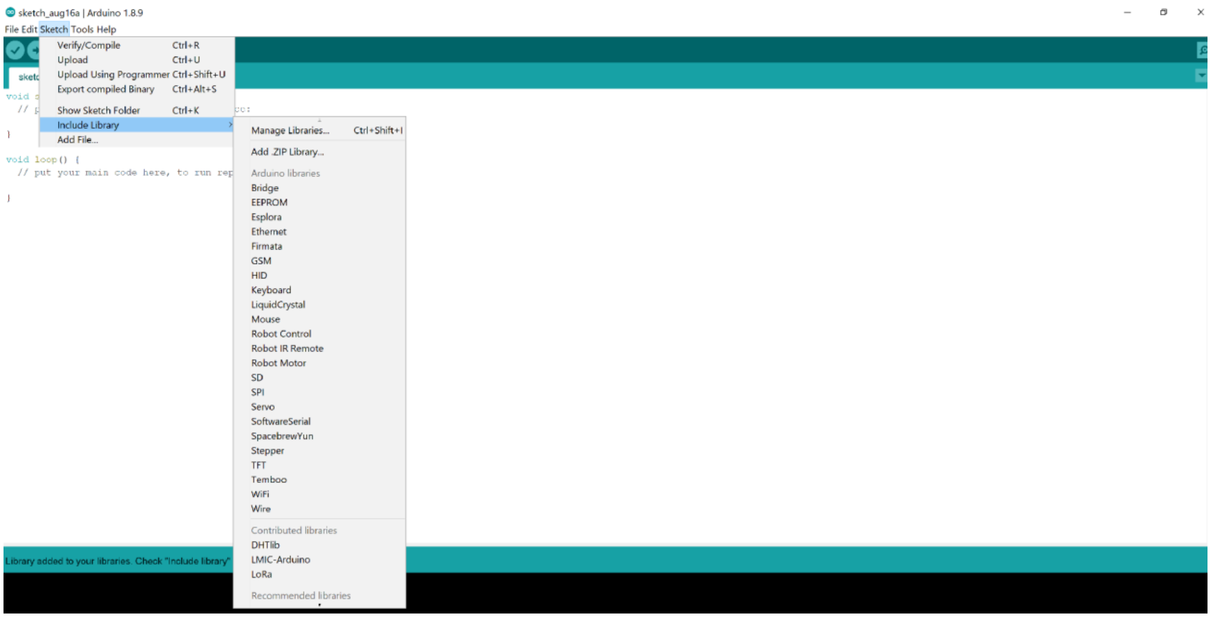
https://github.com/sandeepmistry/arduino-LoRa

Again click ‘Clone or download’ (the green button), then click ‘Download ZIP’. Save the file somewhere you can find it easily.

Make sure your Arduino (with its LoRa shield) is plugged into your computer.

Open the IDE if it is not already open. The procedure is the same as for the installation of the Arduino-LMIC library.

Click ‘Sketch’, then ‘Include Library’. In the flyout that appears, click ‘Add ZIP Library…’, then browse for the ZIP file you just downloaded. Then click on the file, then click ‘Open’. To check the library has been installed, click ‘Sketch’, then ‘Include Library’. In the flyout that appears, in the section headed ‘Contributed libraries’, you should see the library ‘LoRa’.



```
void loop() {
  // put your main code here, to run rep
}
```

Library added to your libraries. Check "Include library"

Add the DHTlib library

This is the library to use DHT11 temperature & humidity sensor. Get it here:

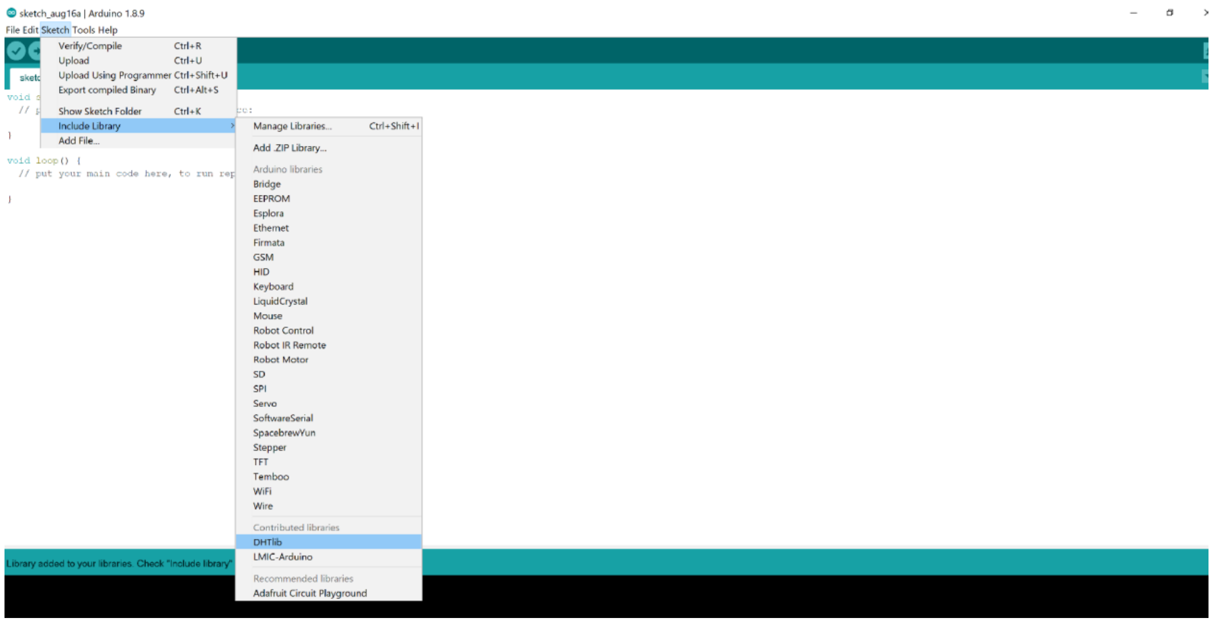
https://github.com/goodcheney/Lora/blob/patch-1/Lora%20Shield/Examples/DHTlib.zip

this time there is no green ‘Clone or download’ button. Instead, click the ‘View raw’ link near the foot of the page. Save the zip file somewhere you can find it.

Make sure your Arduino (with its LoRa shield) is plugged into your computer.

Open the IDE if it is not already open. The procedure is the same as for the installation of the Arduino-LMIC library.

Click ‘Sketch’, then ‘Include Library’. In the flyout that appears, click ‘Add ZIP Library…’, then browse for the ZIP file you just downloaded. Then click on the file, then click ‘Open’. To check the library has been installed, click ‘Sketch’, then ‘Include Library’. In the flyout that appears, in the section headed ‘Contributed libraries’, you should see the library ‘DHTlib’.



```
void loop() {
  // put your main code here, to run rep
}
```

Library added to your libraries. Check "Include library"

Library Locations on your Hard Drive

If you need to edit these libraries, you will need to know where these libraries are.

When you install the Arduino IDE, the included libraries are installed here:

C:\Program Files (x86)\Arduino\libraries

The libraries you downloaded and installed above are called ‘Contributed libraries’ and are stored in your Documents folder:

~\Documents\Arduino\libraries

The LG01-N Gateway

Before we can use this device, we have to connect it to the Internet.

Connect the Dragino gateway to your computer using a RJ45 cable (an ordinary ethernet cable). Connect the Dragino to the power supply. In the UK you will have to use a USA visitor adaptor plug.

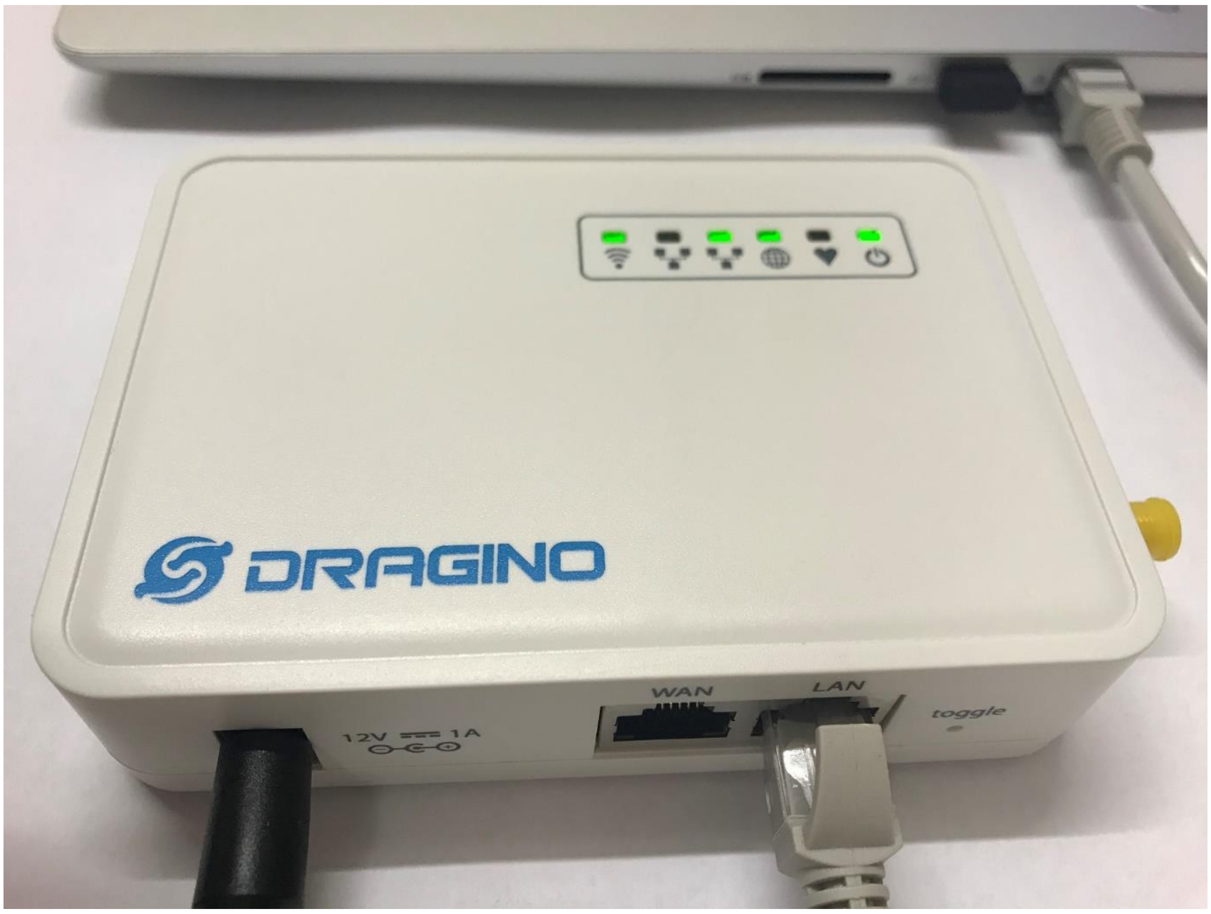
In the Dragino manual (URL above), you are instructed to “set up PC Ethernet port to DHCP”.

“DHCP Stands for Dynamic Host Configuration Protocol. DHCP is a protocol that automatically assigns a unique IP address to each device that connects to a network. With DHCP, there is no need to manually assign IP addresses to new devices. Therefore, no user configuration is necessary to connect to a DCHP-based network. Because of its ease of use and widespread support, DHCP is the default protocol used by most routers and networking equipment.”

https://techterms.com/definition/dhcp

This means your computer is probably already configured for DHCP, so no changes to your computer should be necessary.

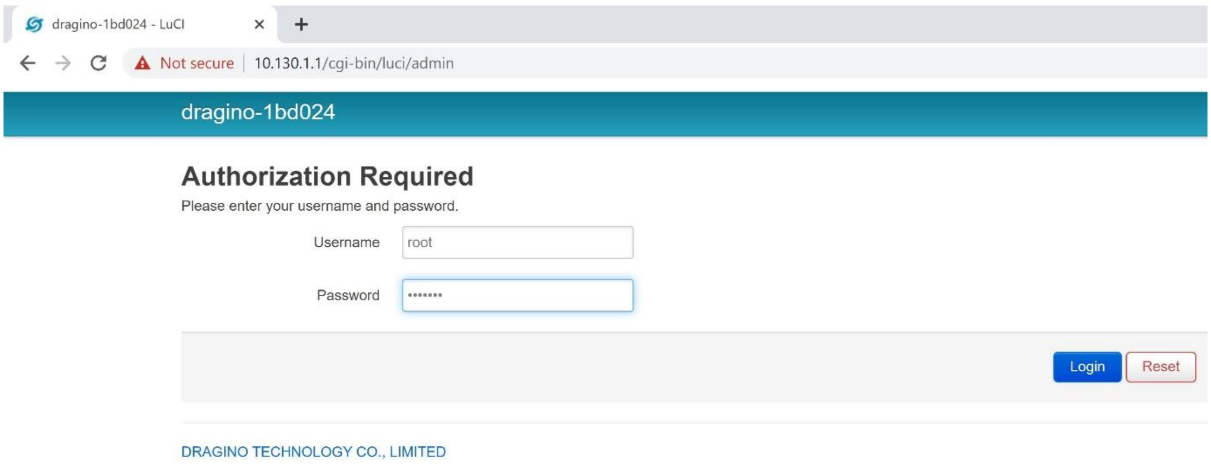
You should see some green lights on the Dragino, for power, www, local network and wireless.



Open your Chrome browser (recommended) and access the Dragino with this URL:

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

You may have to allow the connection before you get to the Dragino login page.


dragino-1bd024 - LuCI
Not secure
10.130.1.1/cgi-bin/luci/admin
dragino-1bd024

Authorization Required

Please enter your username and password.

Username: root

Password: •••••••

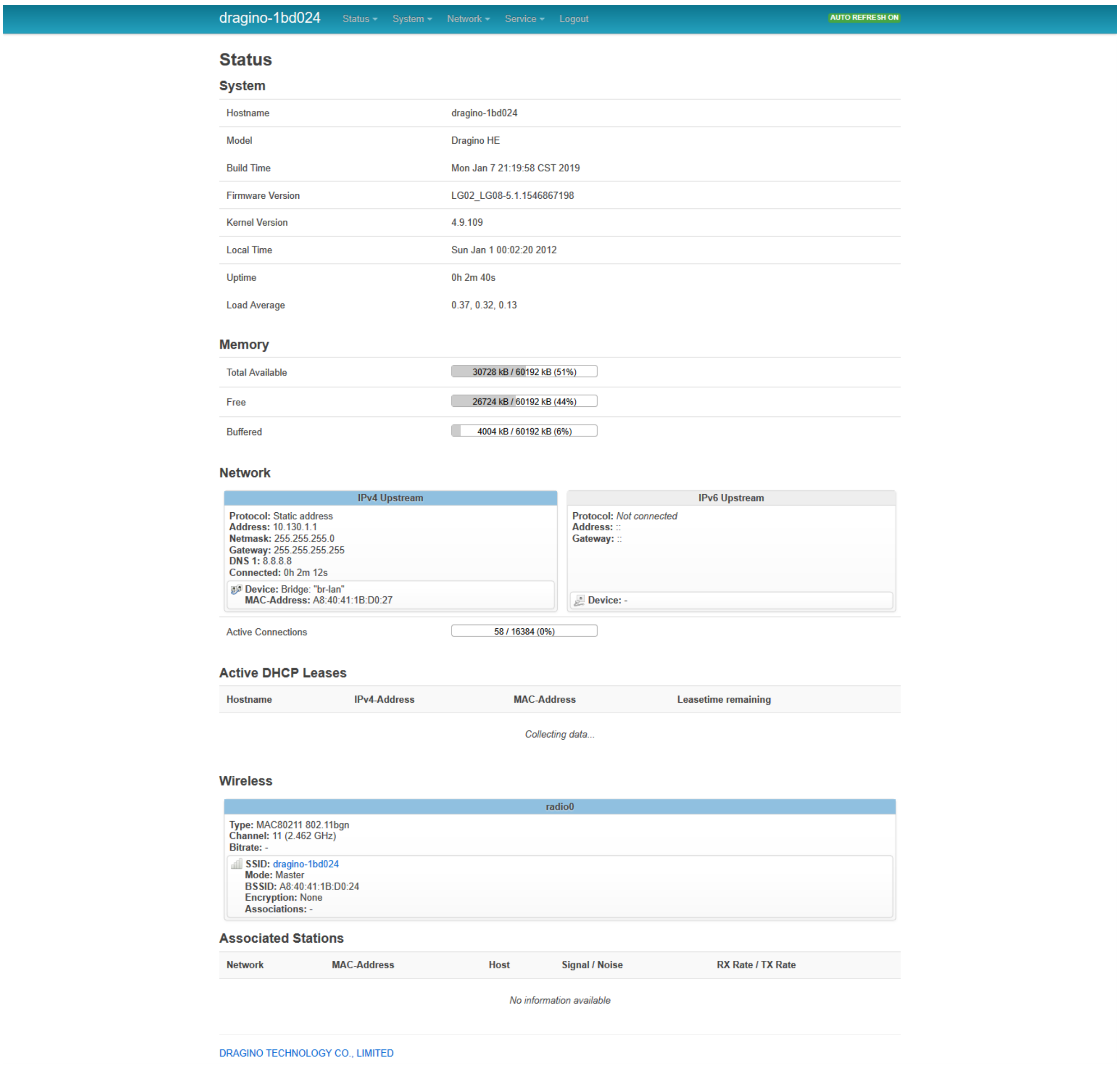
Login
Reset
DRAGINO TECHNOLOGY CO., LIMITED

The account for Web Login is:

User Name: root

Password: dragino

Once logged in, you should see this page.



Status

System

Hostname	dragino-1bd024
Model	Dragino HE
Build Time	Mon Jan 7 21:19:58 CST 2019
Firmware Version	LG02_LG08-5.1.1546867198
Kernel Version	4.9.109
Local Time	Sun Jan 1 00:02:20 2012
Uptime	0h 2m 40s
Load Average	0.37, 0.32, 0.13

Memory

Total Available	30728 kB / 60192 kB (51%)
Free	26724 kB / 60192 kB (44%)
Buffered	4004 kB / 60192 kB (6%)

Network

IPv4 Upstream
Protocol: Static address
Address: 10.130.1.1
Netmask: 255.255.255.0
Gateway: 255.255.255.255
DNS 1: 8.8.8.8
Connected: 0h 2m 12s
Device: Bridge: "br-lan"
MAC-Address: A8:40:41:1B:D0:27

IPv6 Upstream
Protocol: *Not connected*
Address: ::
Gateway: ::
Device: -

Active Connections	58 / 16384 (0%)

Active DHCP Leases

Hostname	IPv4-Address	MAC-Address	Leasetime remaining

Collecting data...

Wireless

radio0
Type: MAC80211 802.11bgn
Channel: 11 (2.462 GHz)
Bitrate: -
SSID: dragino-1bd024
Mode: Master
BSSID: A8:40:41:1B:D0:24
Encryption: None
Associations: -

Associated Stations

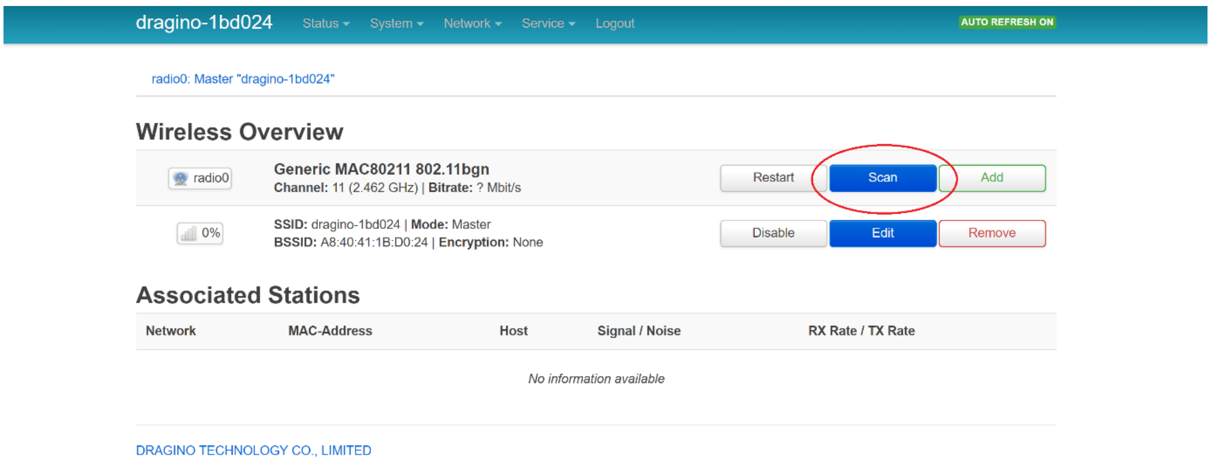
Network	MAC-Address	Host	Signal / Noise	RX Rate / TX Rate

No information available

DRAGINO TECHNOLOGY CO., LIMITED

Hover your mouse over the ‘Network’ tab on the blue bar at the top of the page, and select ‘Wireless’.

You should see this page.



radio0: Master "dragino-1bd024"

Wireless Overview

radio0	**Generic MAC80211 802.11bgn** **Channel:** 11 (2.462 GHz) | **Bitrate:** ? Mbit/s	Restart · Scan · Add
0%	**SSID:** dragino-1bd024 | **Mode:** Master **BSSID:** A8:40:41:1B:D0:24 | **Encryption:** None	Disable · Edit · Remove

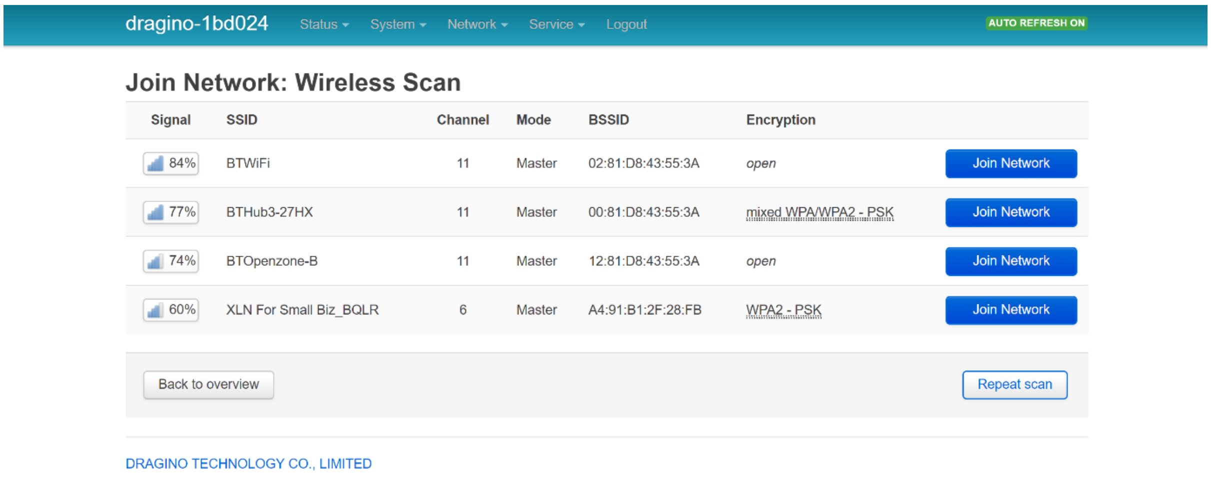
Associated Stations

Network	MAC-Address	Host	Signal / Noise	RX Rate / TX Rate
		No information available		

DRAGINO TECHNOLOGY CO., LIMITED

In the top bar, with the fake 'radio0' button, click 'Scan'.

This should produce a list of available networks.

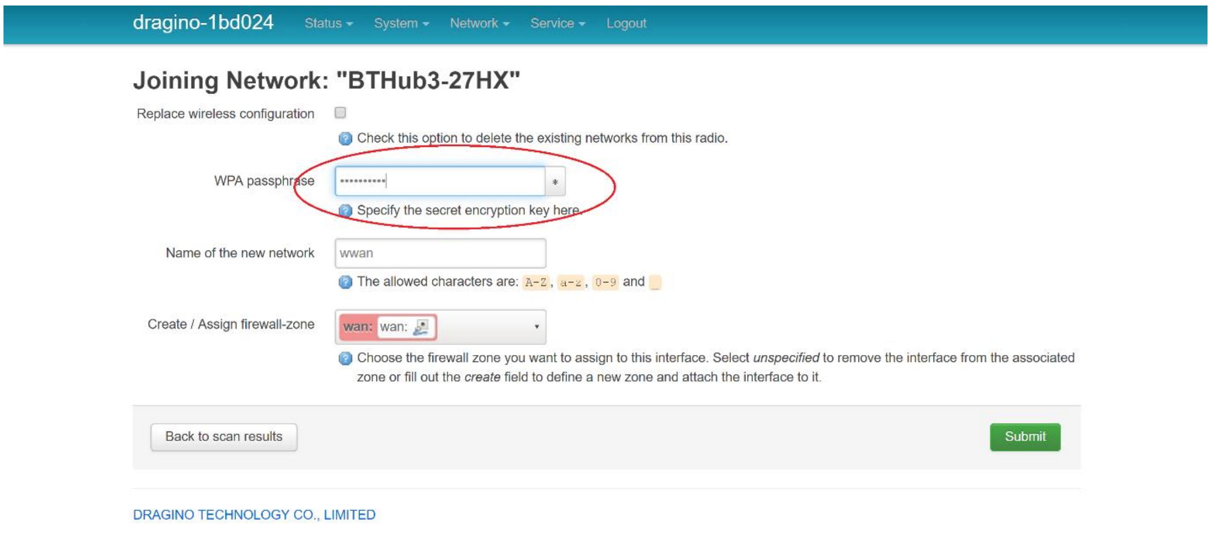


Join Network: Wireless Scan

Signal	SSID	Channel	Mode	BSSID	Encryption	
84%	BTWiFi	11	Master	02:81:D8:43:55:3A	*open*	Join Network
77%	BTHub3-27HX	11	Master	00:81:D8:43:55:3A	mixed WPA/WPA2 - PSK	Join Network
74%	BTOpenzone-B	11	Master	12:81:D8:43:55:3A	*open*	Join Network
60%	XLN For Small Biz_BQLR	6	Master	A4:91:B1:2F:28:FB	WPA2 - PSK	Join Network

DRAGINO TECHNOLOGY CO., LIMITED

Choose you network and click 'Join Network'.



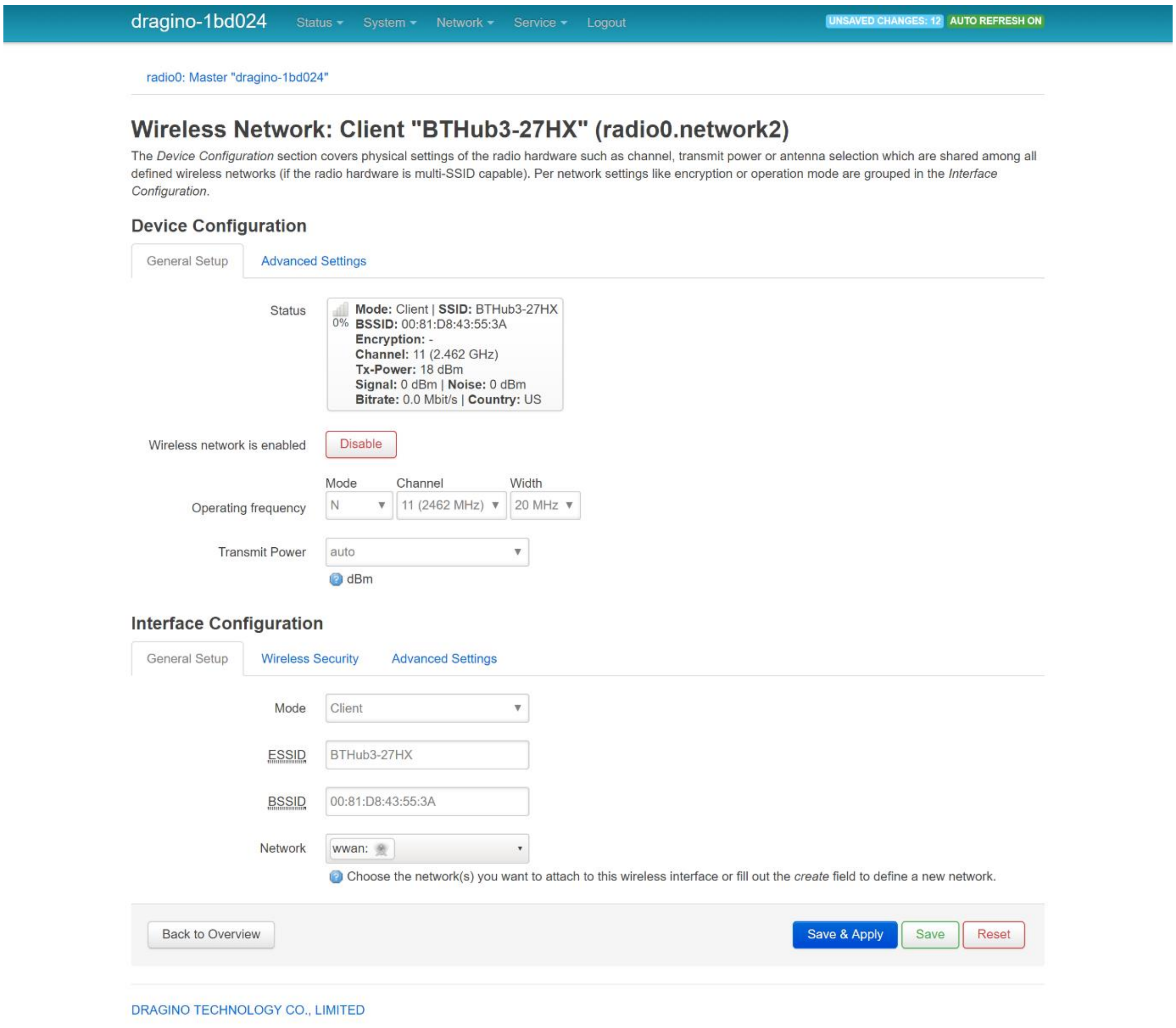
Joining Network: "BTHub3-27HX"

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.

DRAGINO TECHNOLOGY CO., LIMITED

Type your wireless password (wireless key) where is prompts you for the 'WPA passphrase'. Hopefully, the wireless password will be specified somewhere on your Router. Click 'Submit'.

Once logged in, you should see this page:



Wireless Network: Client "BTHub3-27HX" (radio0.network2)

The *Device Configuration* section covers physical settings of the radio hardware such as channel, transmit power or antenna selection which are shared among all defined wireless networks (if the radio hardware is multi-SSID capable). Per network settings like encryption or operation mode are grouped in the *Interface Configuration*.

Device Configuration

General Setup | Advanced Settings

Status: 0% **Mode:** Client | **SSID:** BTHub3-27HX **BSSID:** 00:81:D8:43:55:3A **Encryption:** - **Channel:** 11 (2.462 GHz) **Tx-Power:** 18 dBm **Signal:** 0 dBm | **Noise:** 0 dBm **Bitrate:** 0.0 Mbit/s | **Country:** US

Wireless network is enabled: Disable

Operating frequency: Mode N, Channel 11 (2462 MHz), Width 20 MHz

Transmit Power: auto dBm

Interface Configuration

General Setup | Wireless Security | Advanced Settings

Mode: Client

ESSID: BTHub3-27HX

BSSID: 00:81:D8:43:55:3A

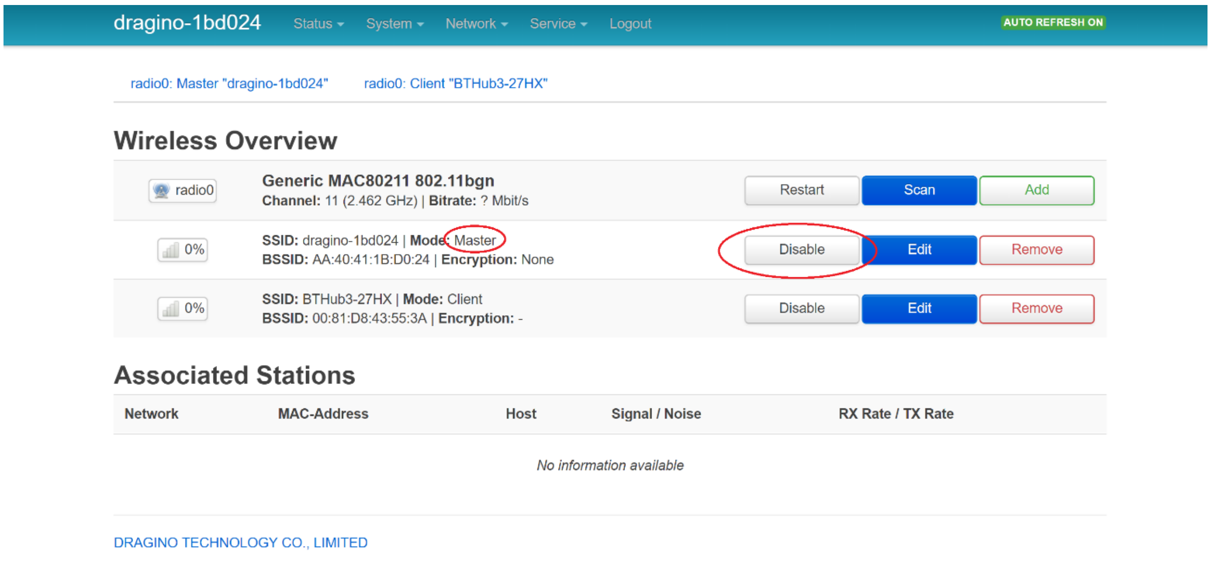
Network: wwan:

Choose the network(s) you want to attach to this wireless interface or fill out the *create* field to define a new network.

Click ‘Save & Apply’.

This will return you to the Wireless Overview page where you should see the network that you have just added.

We now have to disable the original ‘Master’ network, by clicking the ‘Disable’ button.



Wireless Overview

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

0% — **SSID:** dragino-1bd024 | **Mode:** Master
BSSID: AA:40:41:1B:D0:24 | **Encryption:** None

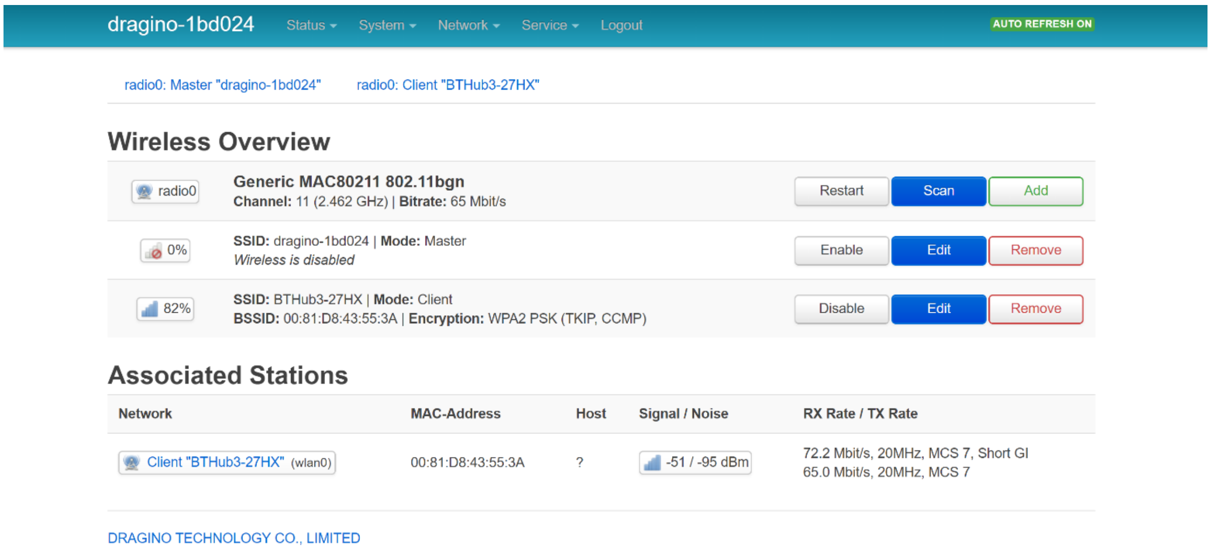
0% — **SSID:** BTHub3-27HX | **Mode:** Client
BSSID: 00:81:D8:43:55:3A | **Encryption:** -

Associated Stations

Network	MAC-Address	Host	Signal / Noise	RX Rate / TX Rate
No information available				

DRAGINO TECHNOLOGY CO., LIMITED

This will reload the Wireless Overview page.



Wireless Overview

radio0 — **Generic MAC80211 802.11bgn**
Channel: 11 (2.462 GHz) | **Bitrate:** 65 Mbit/s

0% — **SSID:** dragino-1bd024 | **Mode:** Master
Wireless is disabled

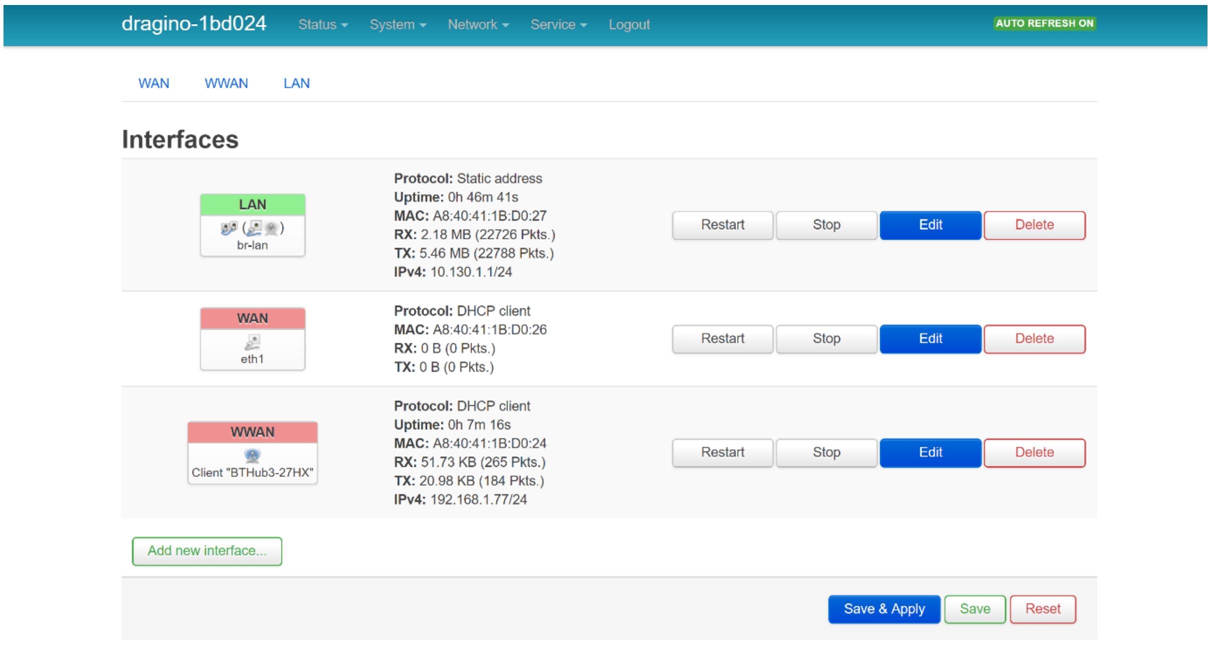
82% — **SSID:** BTHub3-27HX | **Mode:** Client
BSSID: 00:81:D8:43:55:3A | **Encryption:** WPA2 PSK (TKIP, CCMP)

Associated Stations

Network	MAC-Address	Host	Signal / Noise	RX Rate / TX Rate
Client "BTHub3-27HX" (wlan0)	00:81:D8:43:55:3A	?	-51 / -95 dBm	72.2 Mbit/s, 20MHz, MCS 7, Short GI 65.0 Mbit/s, 20MHz, MCS 7

DRAGINO TECHNOLOGY CO., LIMITED

Hover over ‘Network’ on the blue bar and click ‘Interfaces’.



Interfaces

LAN br-lan	**Protocol:** Static address **Uptime:** 0h 46m 41s **MAC:** A8:40:41:1B:D0:27 **RX:** 2.18 MB (22726 Pkts.) **TX:** 5.46 MB (22788 Pkts.) **IPv4:** 10.130.1.1/24	Restart Stop Edit Delete
WAN eth1	**Protocol:** DHCP client **MAC:** A8:40:41:1B:D0:26 **RX:** 0 B (0 Pkts.) **TX:** 0 B (0 Pkts.)	Restart Stop Edit Delete
WWAN Client "BTHub3-27HX"	**Protocol:** DHCP client **Uptime:** 0h 7m 16s **MAC:** A8:40:41:1B:D0:24 **RX:** 51.73 KB (265 Pkts.) **TX:** 20.98 KB (184 Pkts.) **IPv4:** 192.168.1.77/24	Restart Stop Edit Delete

You should see your new network added.

You can now logout of your Dragino gateway.

Logging in to the Dragino LoRa Gateway (LG01-N) using SSH

It will be helpful to be able to access the Linux console inside the Dragino to understand the data flow and to debug. Users can access the Linux console via the SSH protocol.

If you are using a Windows machine and wish to talk to your Dragino and if you have not done so already, then you will have to download PuTTY, http://www.putty.org/. Here you will learn that:

"PuTTY is an SSH and telnet client, developed originally by Simon Tatham for the Windows platform. PuTTY is open source software that is available with source code and is developed and supported by a group of volunteers"

What it doesn't tell you is what PuTTY stands for. Wikipedia again:
"The word PuTTY has no meaning, though 'tty' is sometimes used to refer to the Unix terminals, as an acronym for 'teletype.'"

In case you are wondering what SSH means, here is a definition from Wikipedia:
"Secure Shell (SSH) is a cryptographic network protocol for operating network services securely over an unsecured network. The best known example application is for remote login to computer systems by users. SSH provides a secure channel over an unsecured network in a client-server architecture, connecting an SSH client application with an SSH server."

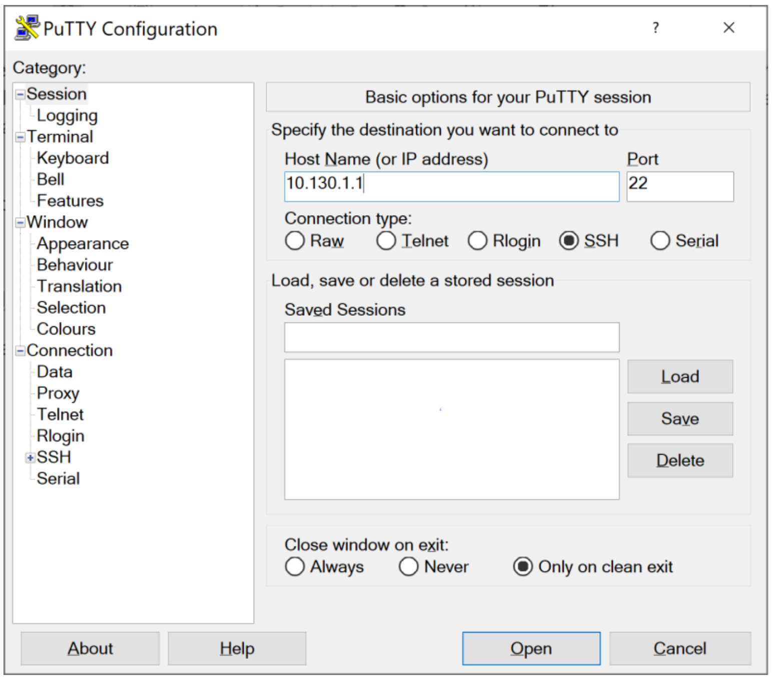
Download and install PuTTY here: https://www.chiark.greenend.org.uk/~sgtatham/putty/latest.html

I downloaded the 32bit option `putty-0.70-installer.msi` , but the 64bit option should be fine on a recent computer.

Logging in to the Dragino remotely requires that both your computer (the one with PuTTY installed) and the Dragino are connected to WiFi through your WiFi router.

If both machines are connected through the same WiFi router, all you need to do is type in the IP address of your Dragino into the 'Host Name (or IP address)' box on the Putty home page.

The IP address for the Dragino is 10.130.1.1



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

Load, save or delete a stored session

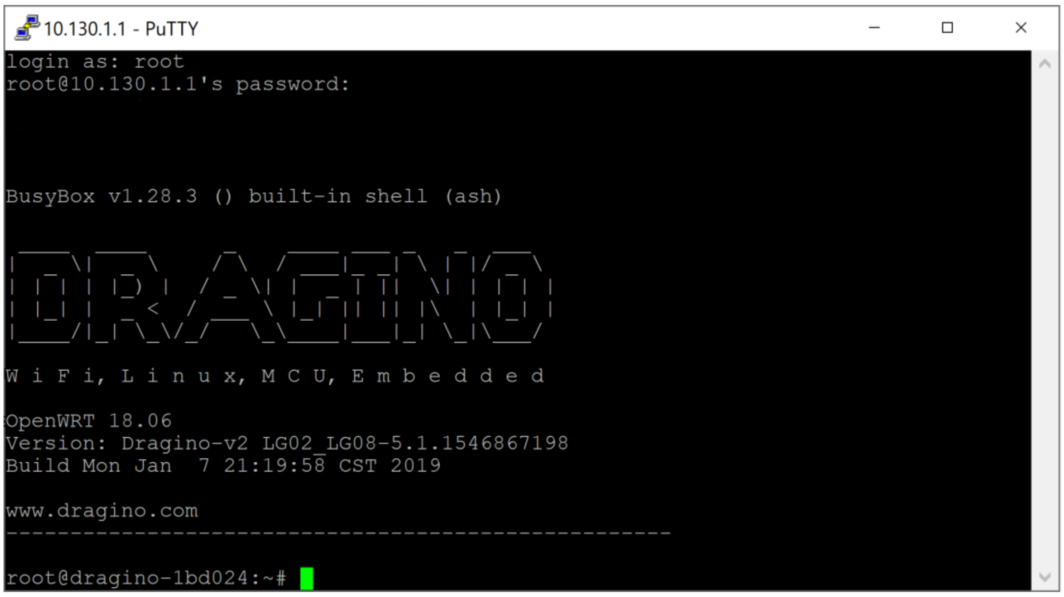
Saved Sessions

Close window on exit: Always Never [✓ Only on clean exit]

Click 'Open' to login to the Dragino. Once you have accessed the Dragino, login with:

User Name: root

Password: dragino



```
10.130.1.1 - PuTTY
login as: root
root@10.130.1.1's password:



BusyBox v1.28.3 () built-in shell (ash)

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

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

OpenWRT 18.06
Version: Dragino-v2 LG02_LG08-5.1.1546867198
Build Mon Jan  7 21:19:58 CST 2019

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

root@dragino-1bd024:~#
```

Test a LoRaWAN Network

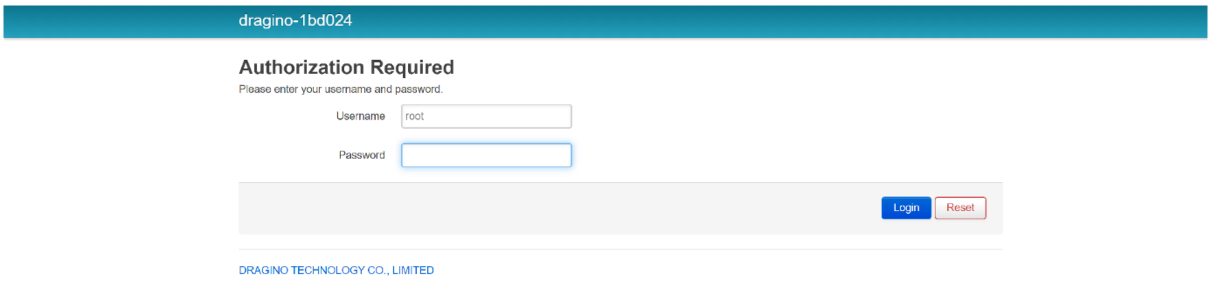
First, you must get the unique ID of your Dragino gateway.

To do this, you must log back in to the Dragino.

Open your Chrome browser (recommended) and access the Dragino with this URL:

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

You may have to allow the connection before you get to the Dragino login page.

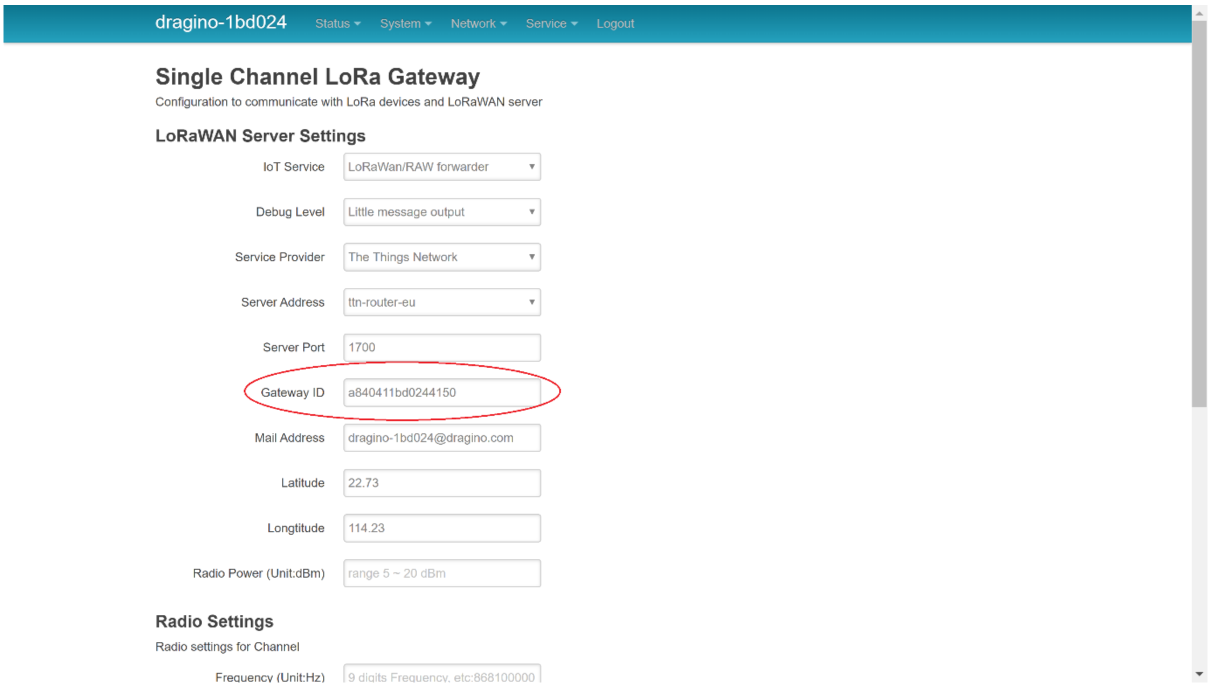


Authorization Required

Please enter your username and password.

Username	root
Password	

Once logged in, click on 'Service' in the blue navigation bar, and then select 'LoRaWan GateWay'.



Single Channel LoRa Gateway

Configuration to communicate with LoRa devices and LoRaWAN server

LoRaWAN Server Settings

IoT Service	LoRaWan/RAW forwarder
Debug Level	Little message output
Service Provider	The Things Network
Server Address	ttn-router-eu
Server Port	1700
Gateway ID	a840411bd0244150
Mail Address	dragino-1bd024@dragino.com
Latitude	22.73
Longtitude	114.23
Radio Power (Unit:dBm)	range 5 ~ 20 dBm

Radio Settings

Radio settings for Channel

Frequency (Unit:Hz)	9 digits Frequency, etc:868100000

In this case, the ID is a840411bd0244150.

Sign up for an account with The Things Network (TTN)

We will be using TTN to send and receive sensor information from the cloud. If you have not already done so, sign up for an account by going to https://www.thethingsnetwork.org/ and clicking the green 'Sign Up' button in the top right.

If you already have an account, then login.

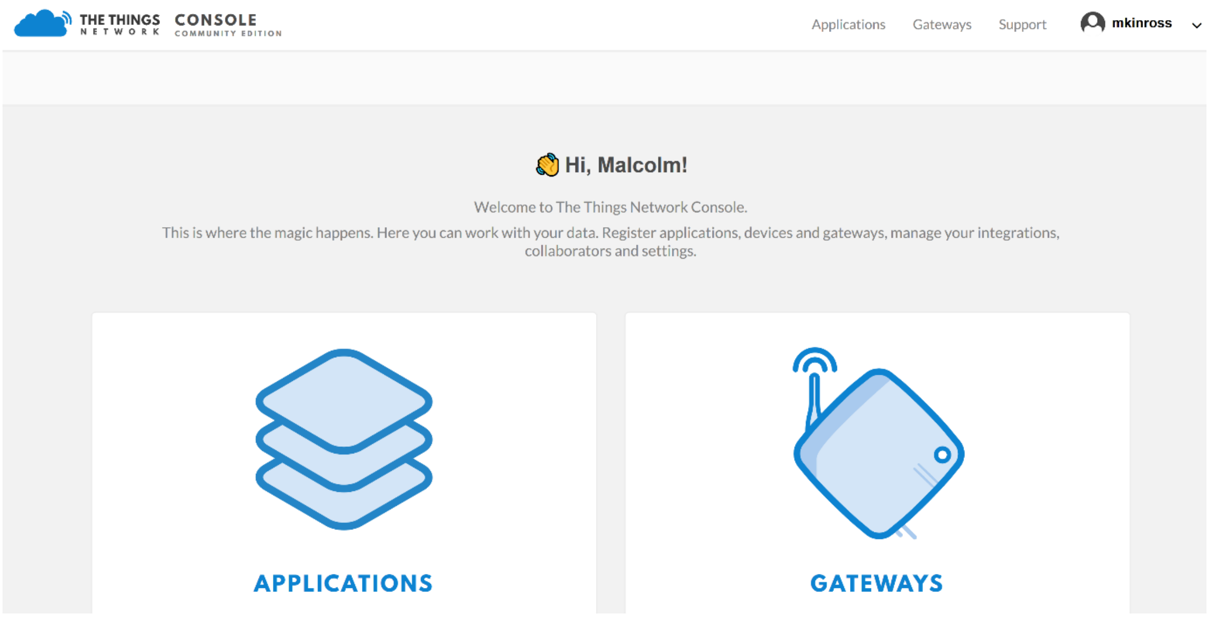
On the TTN home page, you will learn:

"We provide a set of open tools and a global, open network to build your next IoT application at low cost, featuring maximum security and ready to scale.

Through robust end-to-end encryption, a secure and collaborative Internet of Things network is built that spans across many countries around the globe. Now operating thousands of gateways providing coverage to millions of people."

Once you have signed up and logged in, you will need to create a gateway.

Click on this link to get to the TTN console https://console.thethingsnetwork.org/.



Hi, Malcolm!

Welcome to The Things Network Console.

This is where the magic happens. Here you can work with your data. Register applications, devices and gateways, manage your integrations, collaborators and settings.

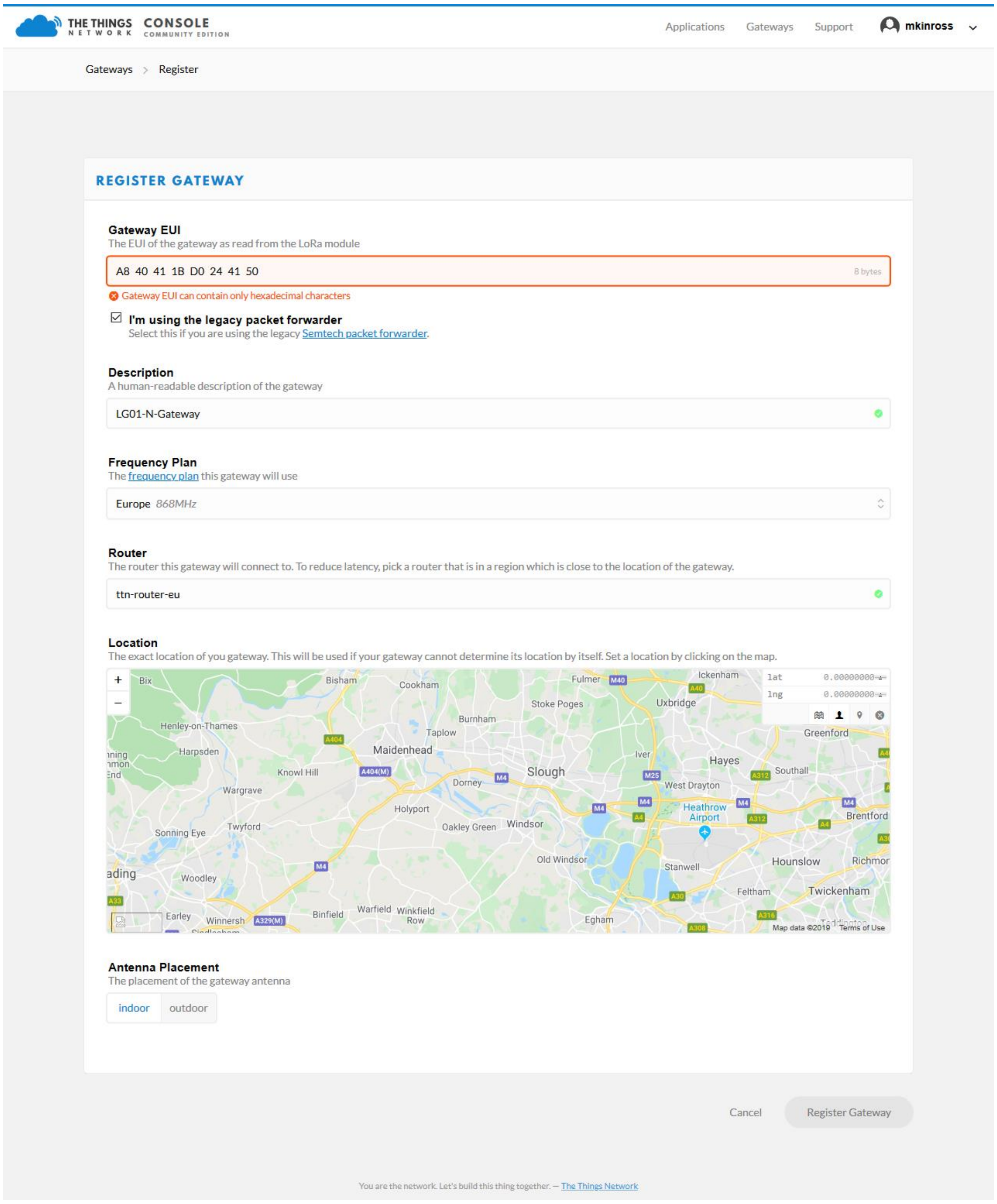
Click on GATEWAYS.

If the next page states "You do not have any gateways", click on the 'Get started by registering one!' link.

Fill in the form:

- Insert the Gateway ID (in this example, it was a840411bd0244150)
- You will be prompted to tick the 'legacy packet forwarder' box
- As you do so, your ID will be converted into hexadecimal numbers
- Type a description (I used 'LG01-N-Gateway')
- Select 'Europe 868MHz' as your frequency plan (assuming you are in Europe)
- The 'Router' box will be filled in for you
- Click in the location map to place a pin at your location
- I selected 'indoor' for Antenna Placement

IMPORTANT: click the orange hexadecimal numbers and the bar will turn from orange to grey. If you don't do this, the 'Register Gateway' button will not work.



REGISTER GATEWAY

Gateway EUI
The EUI of the gateway as read from the LoRa module

A8 40 41 1B D0 24 41 50 — 8 bytes

Gateway EUI can contain only hexadecimal characters

☑ **I'm using the legacy packet forwarder**
Select this if you are using the legacy Semtech packet forwarder.

Description
A human-readable description of the gateway

LG01-N-Gateway

Frequency Plan
The frequency plan this gateway will use

Europe 868MHz

Router
The router this gateway will connect to. To reduce latency, pick a router that is in a region which is close to the location of the gateway.

ttn-router-eu

Location
The exact location of you gateway. This will be used if your gateway cannot determine its location by itself. Set a location by clicking on the map.

lat 0.00000000
lng 0.00000000

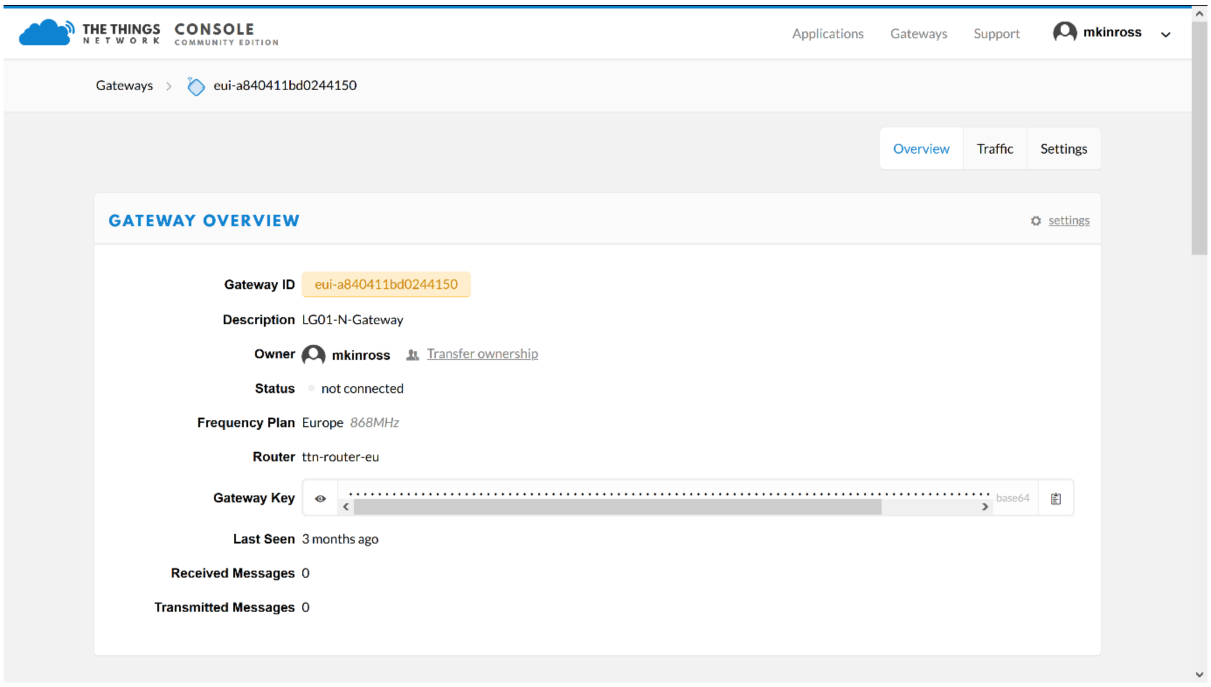
Antenna Placement
The placement of the gateway antenna

indoor | outdoor

Cancel
Register Gateway
You are the network. Let's build this thing together. — The Things Network

Click 'Register Gateway' in the bottom right of the form.

You will then be taken to this page:



Gateway ID eui-a840411bd0244150
Description LG01-N-Gateway
Owner mkinross Transfer ownership
Status not connected
Frequency Plan Europe 868MHz
Router ttn-router-eu
Gateway Key base64
Last Seen 3 months ago
Received Messages 0
Transmitted Messages 0

The status shows ‘not connected’ as we now have to configure the gateway.

Configuring the Gateway to connect to LoRaWAN server

Login to the gateway if you are not already logged in. Make sure your computer is still connected to the Dragino with the ethernet cable. In Chrome, navigate to:

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

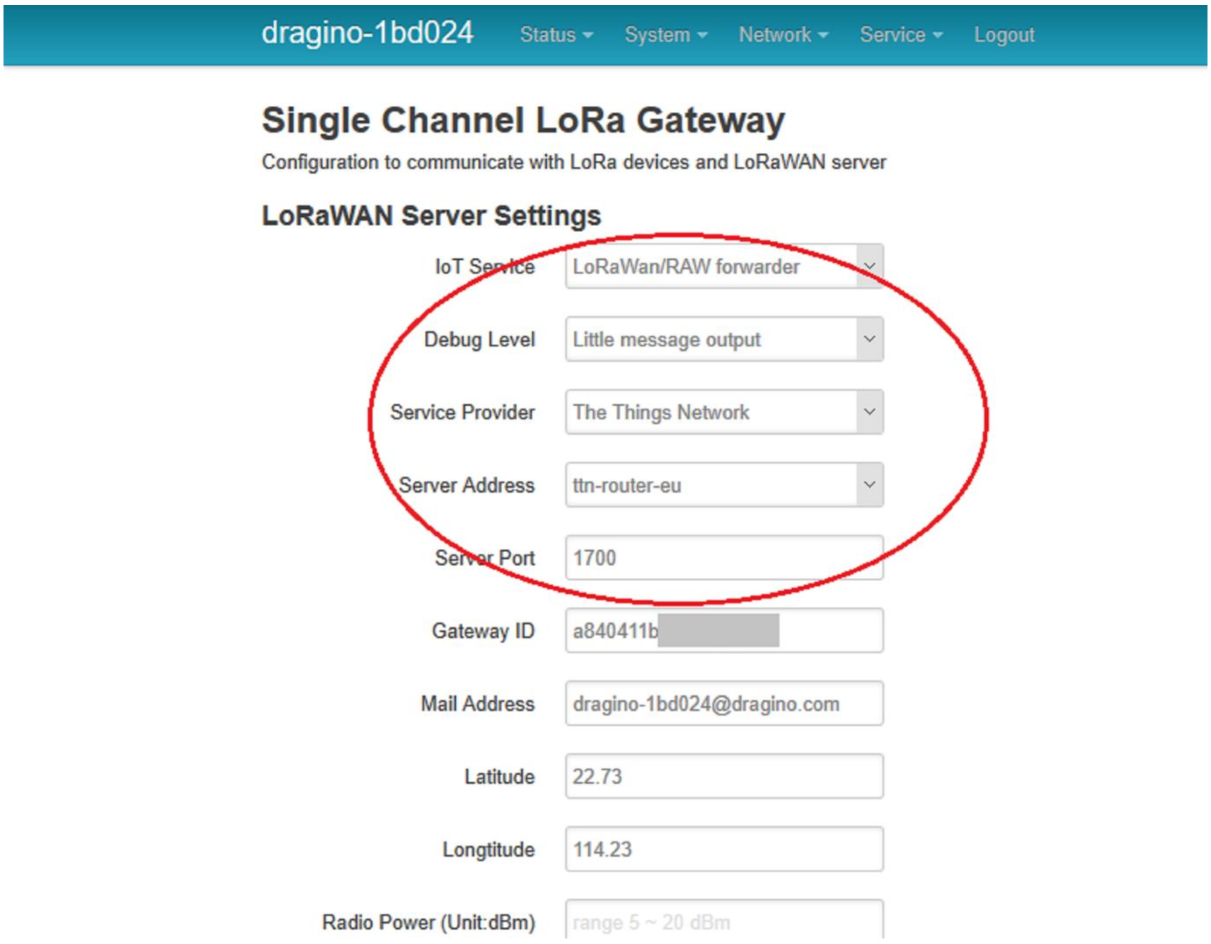
And login with:

User Name: root

Password: dragino

Now open the LoRaWAN server settings page, go to ‘Service’, then ‘LoRaWan GateWay’.

Make sure your settings are as below in the red circle.



Single Channel LoRa Gateway

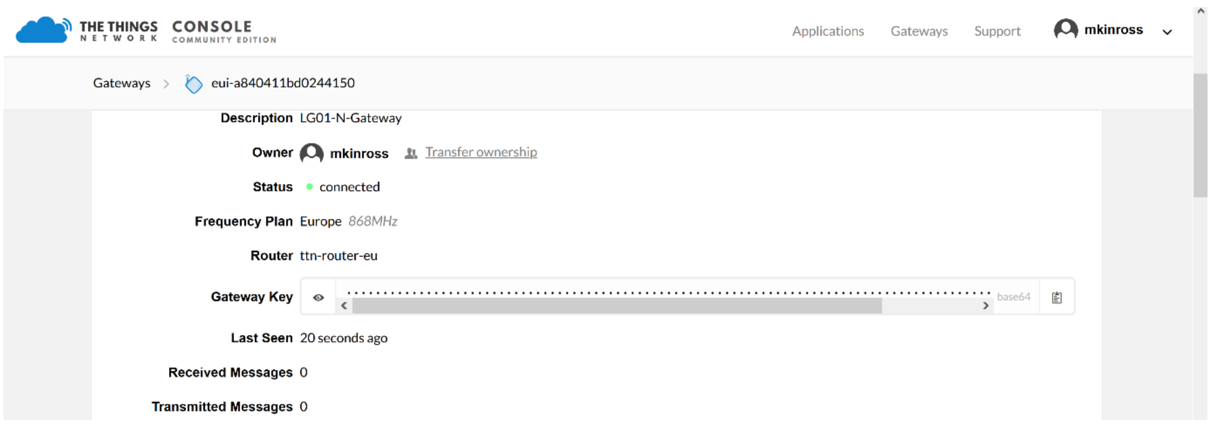
Configuration to communicate with LoRa devices and LoRaWAN server

LoRaWAN Server Settings

IoT Service	LoRaWan/RAW forwarder
Debug Level	Little message output
Service Provider	The Things Network
Server Address	ttn-router-eu
Server Port	1700
Gateway ID	a840411b[redacted]
Mail Address	dragino-1bd024@dragino.com
Latitude	22.73
Longtitude	114.23
Radio Power (Unit:dBm)	range 5 ~ 20 dBm

Then click the blue ‘Save and Apply’ button at the foot of the page.

If you now return to your TTN console, you should see the status as ‘connected’.



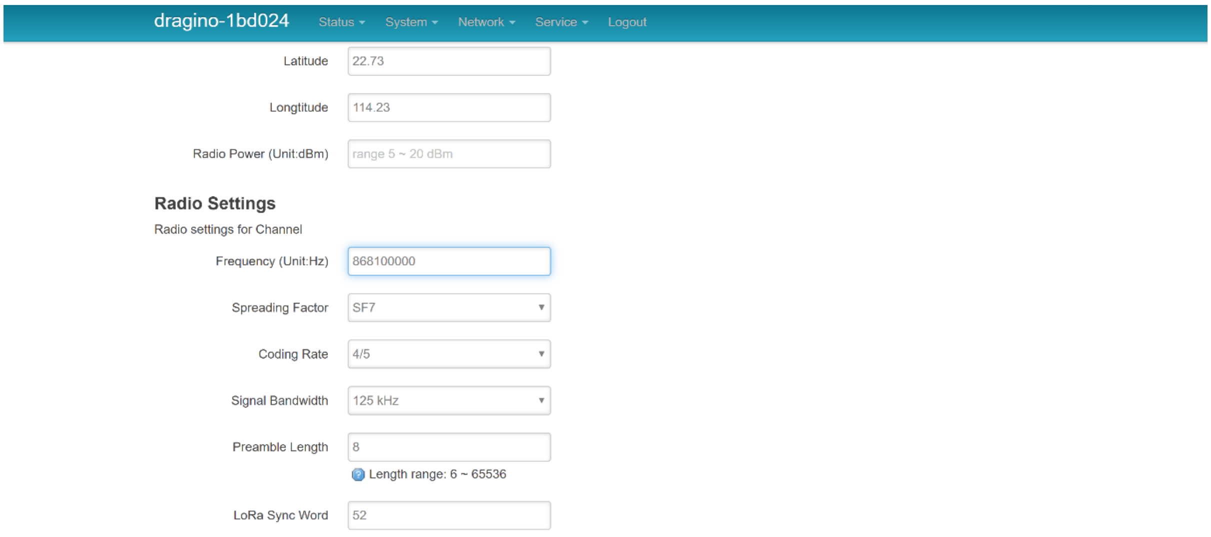
Description	LG01-N-Gateway
Owner	mkinross Transfer ownership
Status	● connected
Frequency Plan	Europe *868MHz*
Router	ttn-router-eu
Gateway Key	·· base64
Last Seen	20 seconds ago
Received Messages	0
Transmitted Messages	0

Configure Radio Settings

Return to your LoRa Gateway settings that you had opened in your Chrome browser.

Scroll down the page to see the radio settings.

If you are in Europe, set the ‘Frequency’ to 868.1MHz. You will have to type the full number, which is 868100000.



Latitude: 22.73
Longtitude: 114.23
Radio Power (Unit:dBm): range 5 ~ 20 dBm

Radio Settings
Radio settings for Channel

Frequency (Unit:Hz): 868100000
Spreading Factor: SF7
Coding Rate: 4/5
Signal Bandwidth: 125 kHz
Preamble Length: 8
Length range: 6 ~ 65536
LoRa Sync Word: 52

As it says in the tutorial:

"This parameters set is for uplink (receive data for LoRa End Node).According to LoRaWAN spec, the downlink radio parameters frequency is defined by network server (TTN). LG01-N will adjust downlink parameters according to info from TTN."

Once you have set the frequency, click the blue 'Save and Apply' button. You should see a 'Configuration has been applied' confirmation message.

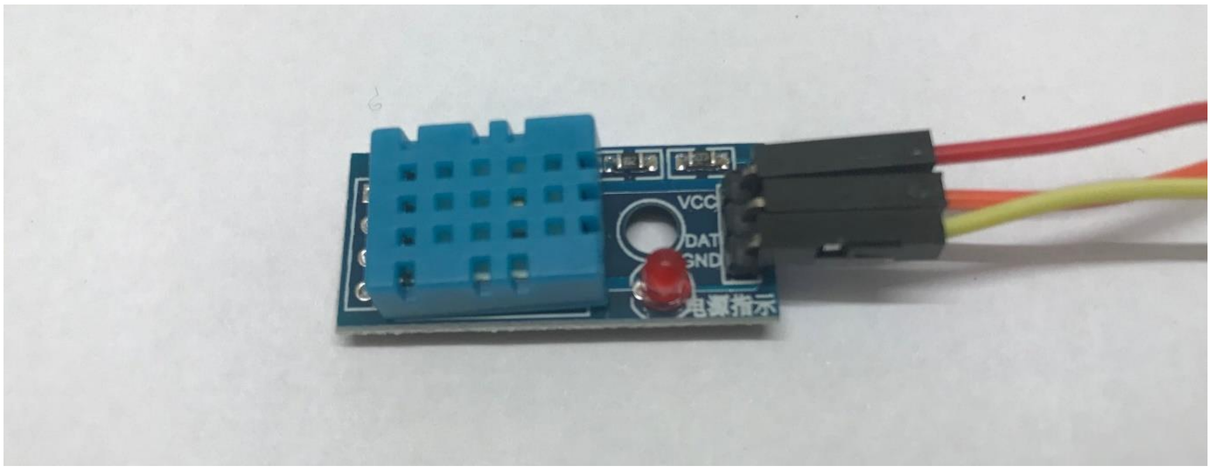
Connecting the Sensors - Create LoRa Shield End Node

We can now create the LoRa end node. The end node is the bit with the sensors connected to a (Arduino) radio transmitter so they can send data over the network.

We shall start by connecting the DHT11 temperature and humidity sensor to the Arduino we set up earlier.

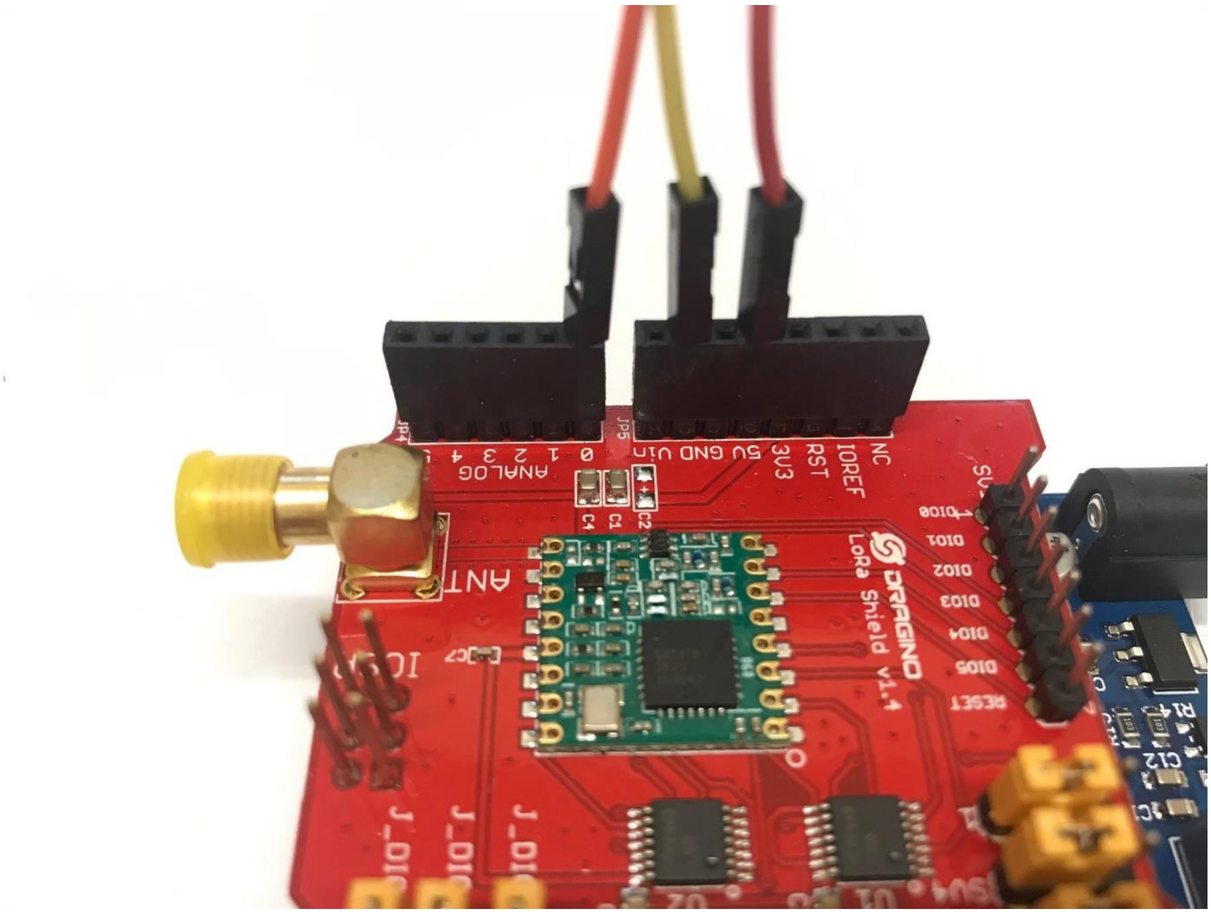
Three male to female jumper wires were used to connect the two. All components can be found in the Development Kit.

Here is a close up of the DHT11.



Here the red wire is connected to VCC (5v input), the orange wire is connected to DAT (sends an analogue signal to the Arduino), and the yellow wire is connected to GND (or ground, which completes the circuit).

At the Arduino end of the wires, it looks like this.



Starting from the left, the orange wire goes to ANALOGUE pin 0, the yellow wire goes to GND (there are two female GND pins, you can use either), and the red wire goes to the 5V input.

You can check everything is working by uploading the sketch below to the Arduino. By the way, 'sketch' is what the Arduino programmers call their programs.

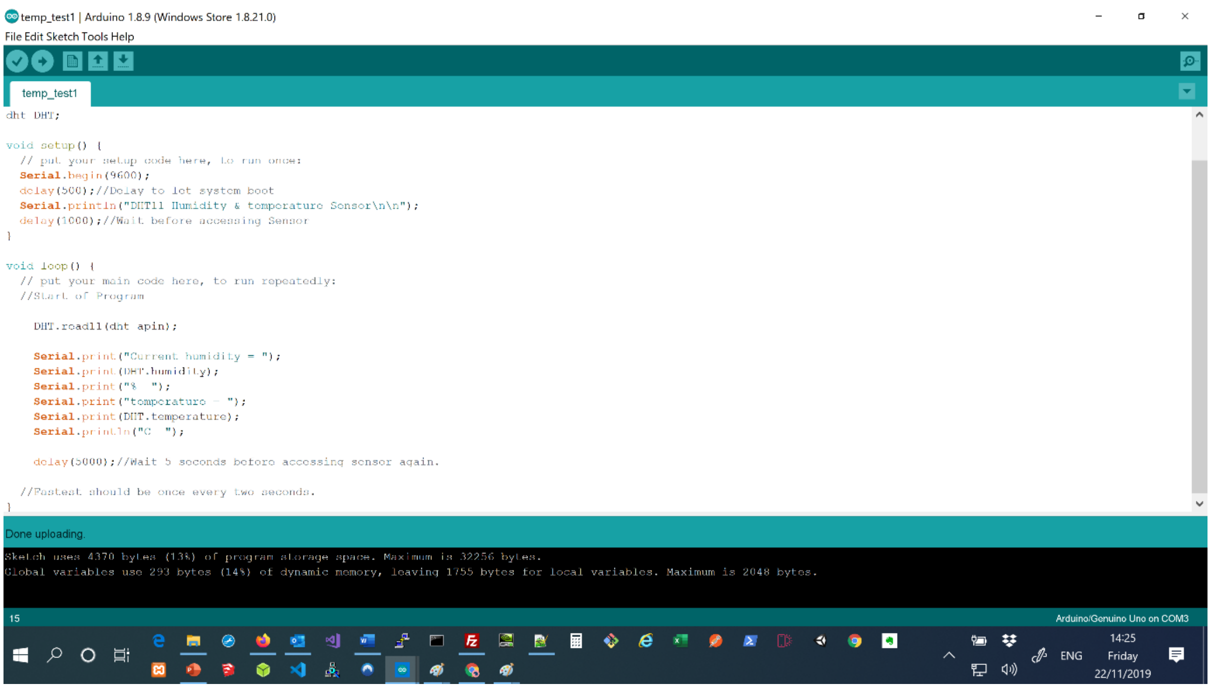
The sketch below was found here http://www.circuitbasics.com/how-to-set-up-the-dht11-humidity-sensor-on-an-arduino/

Sketch to test a LoRaWAN Network:

```
dht DHT; // this accesses the DHTlib that we downloaded earlier
void setup() {
 // put your setup code here, to run once:
 Serial.begin(9600); //the is the rate that data is transmitted
 delay(500);//Delay to let system boot
 Serial.println("DHT11 Humidity & temperature Sensor\n\n");
 delay(1000);//Wait before accessing Sensor
}
void loop() {
 // put your main code here, to run repeatedly:
 //Start of Program
   DHT.read11(dht_apin);
   Serial.print("Current humidity = ");
   Serial.print(DHT.humidity);
   Serial.print("%  ");
   Serial.print("temperature = ");
   Serial.print(DHT.temperature);
   Serial.println("C  ");
   delay(5000);//Wait 5 seconds before accessing sensor again.
 //Fastest should be once every two seconds.
}
```

Copy and paste this code into your Arduino IDE. It should look like the screen shot below.

You can check that the sketch is OK by clicking the green ‘tick’ button (the ‘Verify’ button) at the top left of the IDE. This ‘compiles’ the sketch. You can monitor its progress by reading the message that appears at the foot of the IDE.



```
dht DHT;

void setup() {
  // put your setup code here, to run once:
  Serial.begin(9600);
  delay(500);//Delay to let system boot
  Serial.println("DHT11 Humidity & temperature Sensor\n\n");
  delay(1000);//Wait before accessing Sensor
}

void loop() {
  // put your main code here, to run repeatedly:
  //Start of Program

    DHT.read11(dht_apin);

    Serial.print("Current humidity = ");
    Serial.print(DHT.humidity);
    Serial.print("%  ");
    Serial.print("temperature = ");
    Serial.print(DHT.temperature);
    Serial.println("C  ");

    delay(5000);//Wait 5 seconds before accessing sensor again.

  //Fastest should be once every two seconds.
}
```

Done uploading.

```
Sketch uses 4370 bytes (13%) of program storage space. Maximum is 32256 bytes.
Global variables use 293 bytes (14%) of dynamic memory, leaving 1755 bytes for local variables. Maximum is 2048 bytes.
```

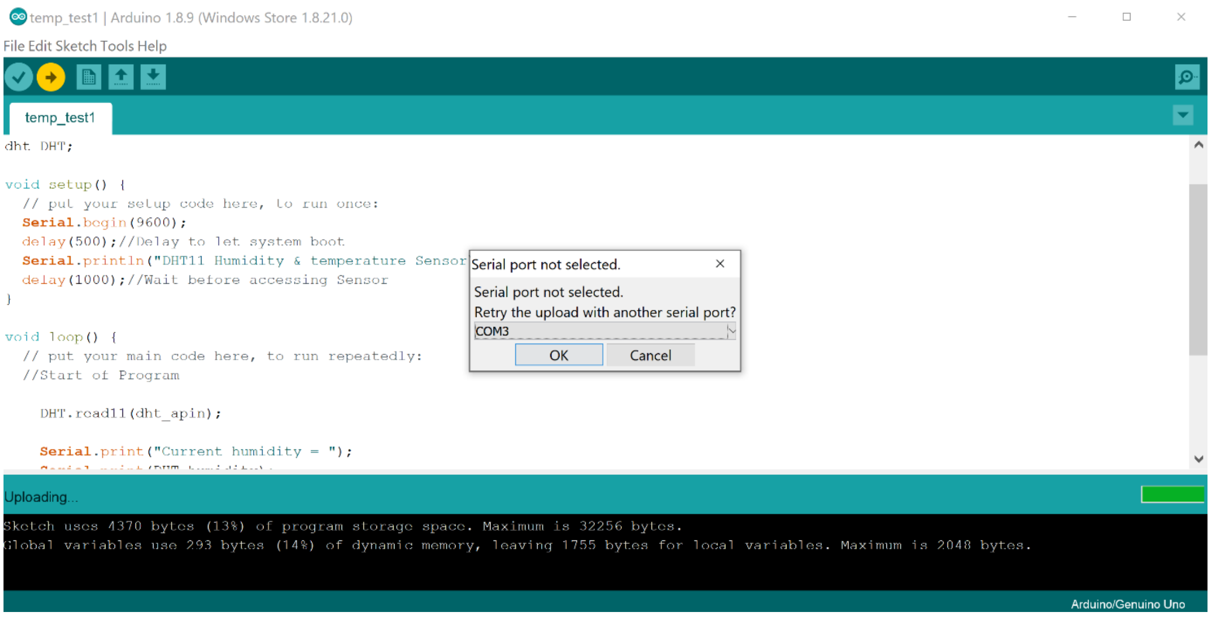
temp_test1 | Arduino 1.8.9 (Windows Store 1.8.21.0)
File Edit Sketch Tools Help
temp_test1
15
Arduino/Genuino Uno on COM3
ENG
14:25
Friday
22/11/2019

When you are ready to upload the sketch to the Arduino, you can click the green right arrow (upload) button at the top left of the IDE. Again, you can monitor progress by reading the message at the foot of the IDE.

Before you upload the script, just make sure the Arduino is plugged into your computer, and that the red power light is showing on the DHT11.

If all is ready, upload the script.

If you see the message below, just click OK.



```
dht DHT;

void setup() {
  // put your setup code here, to run once:
  Serial.begin(9600);
  delay(500);//Delay to let system boot
  Serial.println("DHT11 Humidity & temperature Sensor
  delay(1000);//Wait before accessing Sensor
}

void loop() {
  // put your main code here, to run repeatedly:
  //Start of Program

    DHT.read11(dht_apin);

    Serial.print("Current humidity = ");
```

Serial port not selected.
Retry the upload with another serial port?
COM3

Uploading...

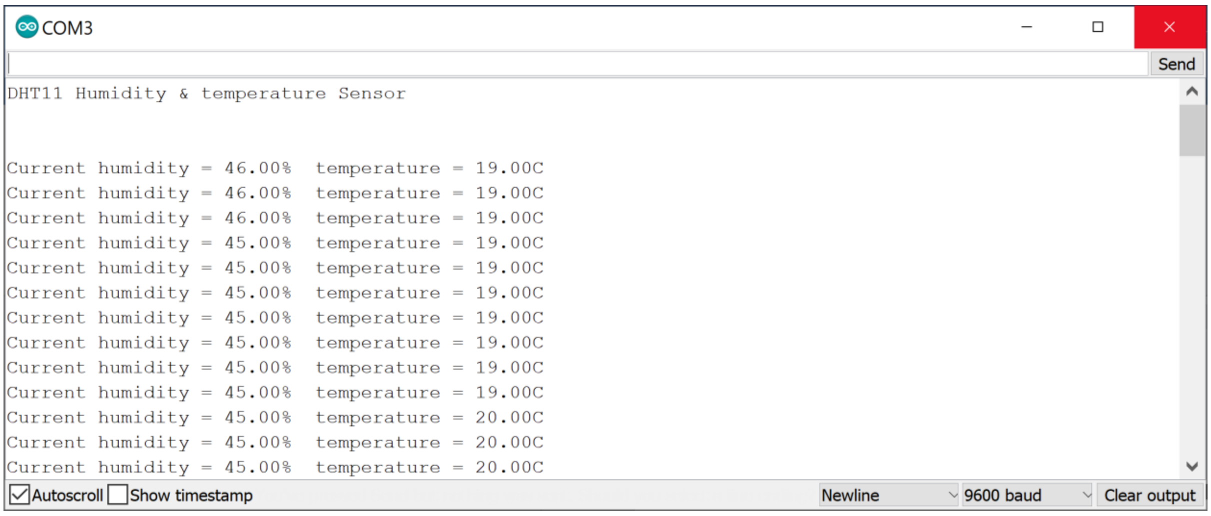
```
Sketch uses 4370 bytes (13%) of program storage space. Maximum is 32256 bytes.
Global variables use 293 bytes (14%) of dynamic memory, leaving 1755 bytes for local variables. Maximum is 2048 bytes.
```

temp_test1 | Arduino 1.8.9 (Windows Store 1.8.21.0)
File Edit Sketch Tools Help
temp_test1
Serial port not selected.
OK
Cancel
Arduino/Genuino Uno

For future reference, you can specify the serial port by going to ‘Tools’ then ‘Port’ and then select the port that has been allocated. How to check your allocated port was discussed near the beginning of this article.

When the message at the foot of the IDE reads ‘done uploading’, you know the process is complete.

Now open the serial monitor. Click the button at the very top right of the IDE. If you mouse over it, it will read Serial Monitor. If all has gone to plan, you will see this popup.



```
DHT11 Humidity & temperature Sensor


Current humidity = 46.00%  temperature = 19.00C
Current humidity = 46.00%  temperature = 19.00C
Current humidity = 46.00%  temperature = 19.00C
Current humidity = 45.00%  temperature = 19.00C
Current humidity = 45.00%  temperature = 19.00C
Current humidity = 45.00%  temperature = 19.00C
Current humidity = 45.00%  temperature = 19.00C
Current humidity = 45.00%  temperature = 19.00C
Current humidity = 45.00%  temperature = 19.00C
Current humidity = 45.00%  temperature = 19.00C
Current humidity = 45.00%  temperature = 20.00C
Current humidity = 45.00%  temperature = 20.00C
Current humidity = 45.00%  temperature = 20.00C
```

You now know the system is working.

Register your device with the things network

Before a device can communicate via The Things Network you need to register it with an application.

We shall use the default Over the Air Activation (OTAA) supported by The Things Network. We now need to register the LoRa Shield + UNO device with its Device EUI (Extended Unique Identifier). This assigns a unique IP address to your device. See here for more details https://community.cisco.com/t5/networking-documents/understanding-ipv6-eui-64-bit-address/ta-p/3116953.

We need to add an application to TTN. Each application you create can have multiple devices associated with it.

Go back to The Things Network console https://console.thethingsnetwork.org. Click on ‘Applications’, then ‘Add Application’.

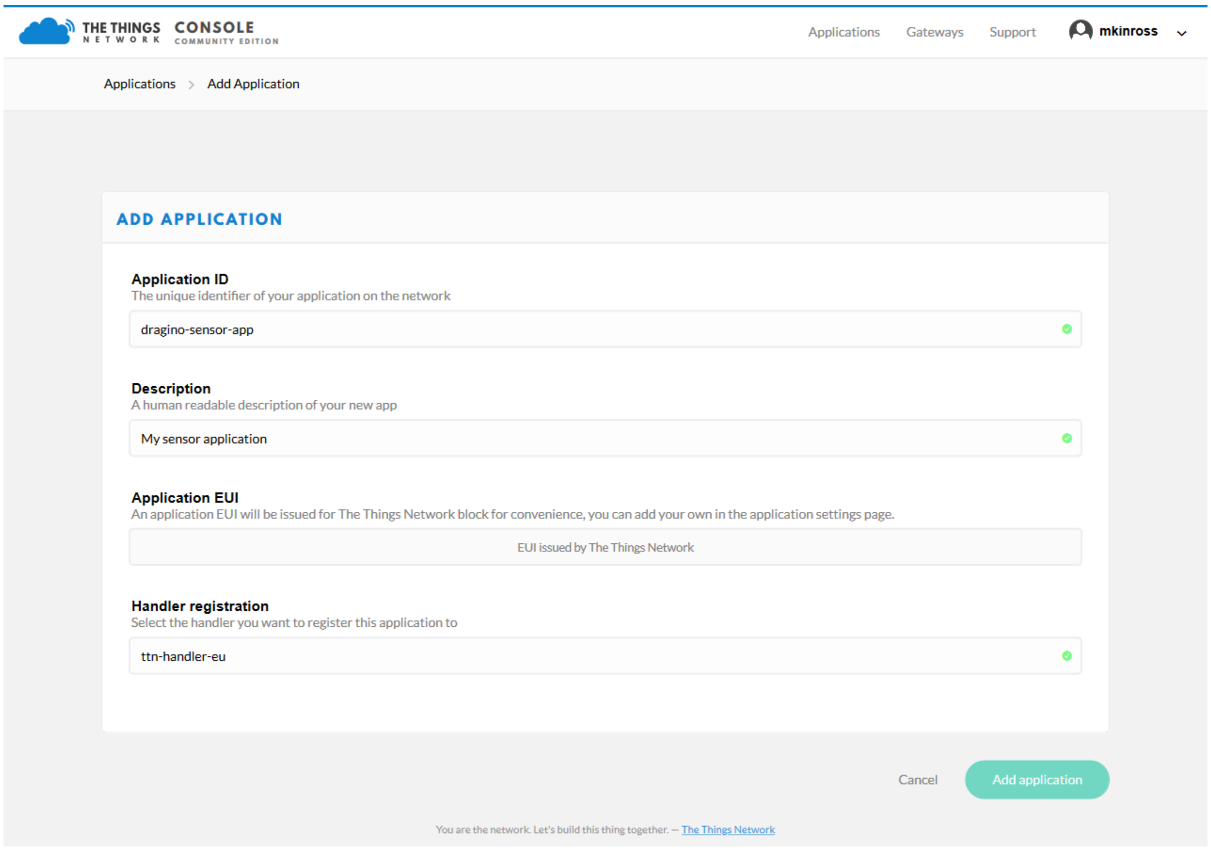
Application ID - You can add anything to the ‘Application ID’ box as long as it is unique.

Description – A human readable description of your device

Application EUI – This identifier will be supplied by the TTN.

Handler Registration – If you are in Europe, this will be ‘ttn-handler-eu’.

(Tutorial page approx. 19)


THE THINGS NETWORK CONSOLE COMMUNITY EDITION
Applications
Gateways
Support
mkinross
Applications > Add Application

ADD APPLICATION

Application ID
The unique identifier of your application on the network
dragino-sensor-app

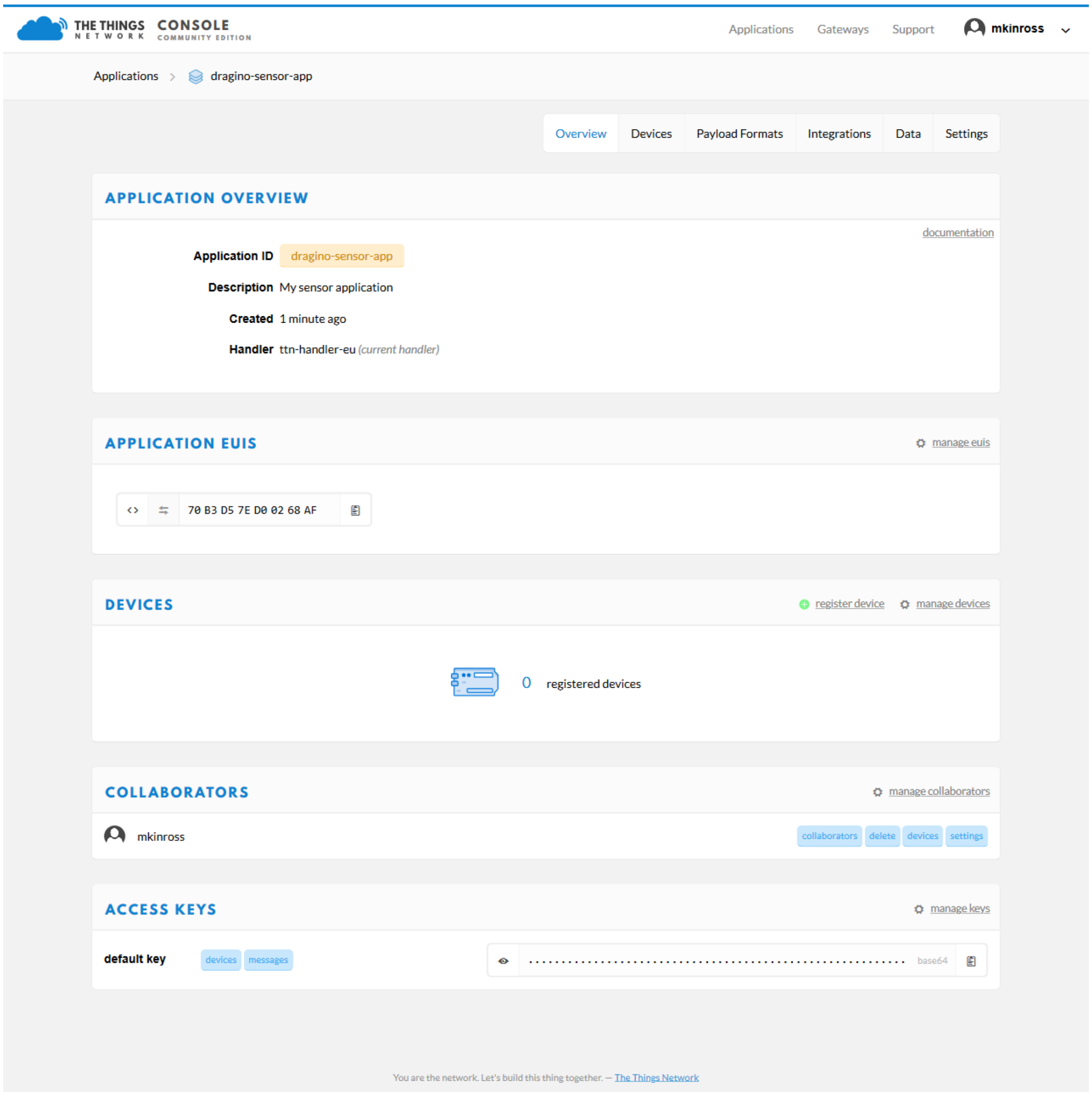
Description
A human readable description of your new app
My sensor application

Application EUI
An application EUI will be issued for The Things Network block for convenience, you can add your own in the application settings page.
EUI issued by The Things Network

Handler registration
Select the handler you want to register this application to
ttn-handler-eu

Cancel
Add application
You are the network. Let's build this thing together. — The Things Network

Click the green ‘Add Application’ button, and you should see this:



APPLICATION OVERVIEW

documentation

Application ID dragino-sensor-app

Description My sensor application

Created 1 minute ago

Handler ttn-handler-eu *(current handler)*

APPLICATION EUIS

manage euis

70 B3 D5 7E D0 02 68 AF

DEVICES

register device manage devices

0 registered devices

COLLABORATORS

manage collaborators

mkinross collaborators delete devices settings

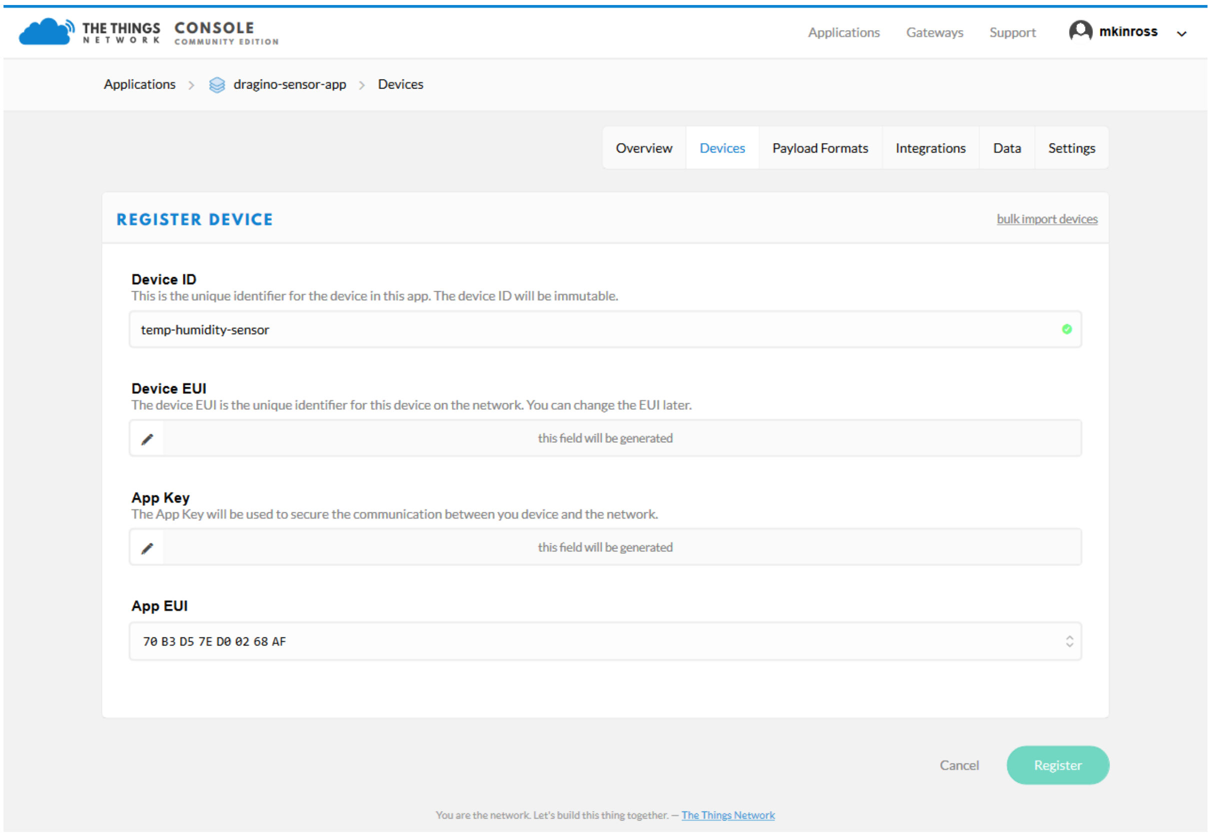
ACCESS KEYS

manage keys

default key devices messages base64

You are the network. Let's build this thing together. – The Things Network

In the DEVICES section, click ‘register device’.



REGISTER DEVICE

Device ID
This is the unique identifier for the device in this app. The device ID will be immutable.
temp-humidity-sensor

Device EUI
The device EUI is the unique identifier for this device on the network. You can change the EUI later.
this field will be generated

App Key
The App Key will be used to secure the communication between you device and the network.
this field will be generated

App EUI
70 B3 D5 7E D0 02 68 AF

You are the network. Let's build this thing together. – The Things Network

Device ID – This must be unique within the application

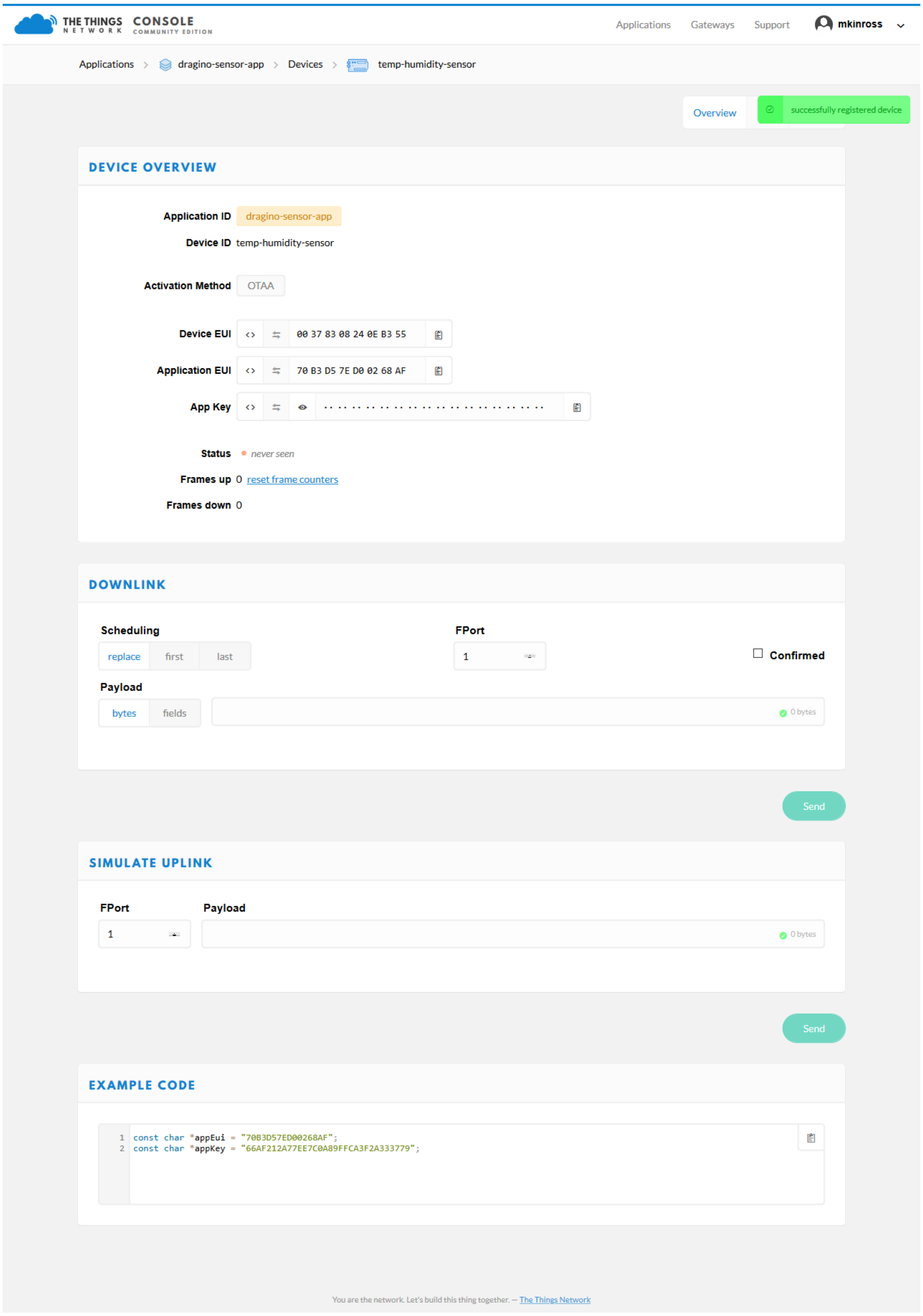
Device EUI – Click the 'squiggle' to the left of the box and it will turn into a pencil, which means it will be generated automatically

App Key – This will be generated automatically too

App EUI – This is the App EUI from the application overview and should be populated automatically

Click the green 'Register' button when you are ready

You will notice that the activation method OTAA (over the air activation) is already selected.



THE THINGS NETWORK CONSOLE COMMUNITY EDITION
Applications
Gateways
Support
mkinross
Applications > dragino-sensor-app > Devices > temp-humidity-sensor
Overview
successfully registered device

DEVICE OVERVIEW

Application ID dragino-sensor-app
Device ID temp-humidity-sensor

Activation Method OTAA

Device EUI 00 37 83 08 24 0E B3 55
Application EUI 70 B3 D5 7E D0 02 68 AF
App Key ································

Status never seen
Frames up 0 reset frame counters
Frames down 0

DOWNLINK

Scheduling: replace | first | last
FPort: 1
Confirmed
Payload: bytes | fields — 0 bytes

Send

SIMULATE UPLINK

FPort: 1
Payload: 0 bytes

Send

EXAMPLE CODE

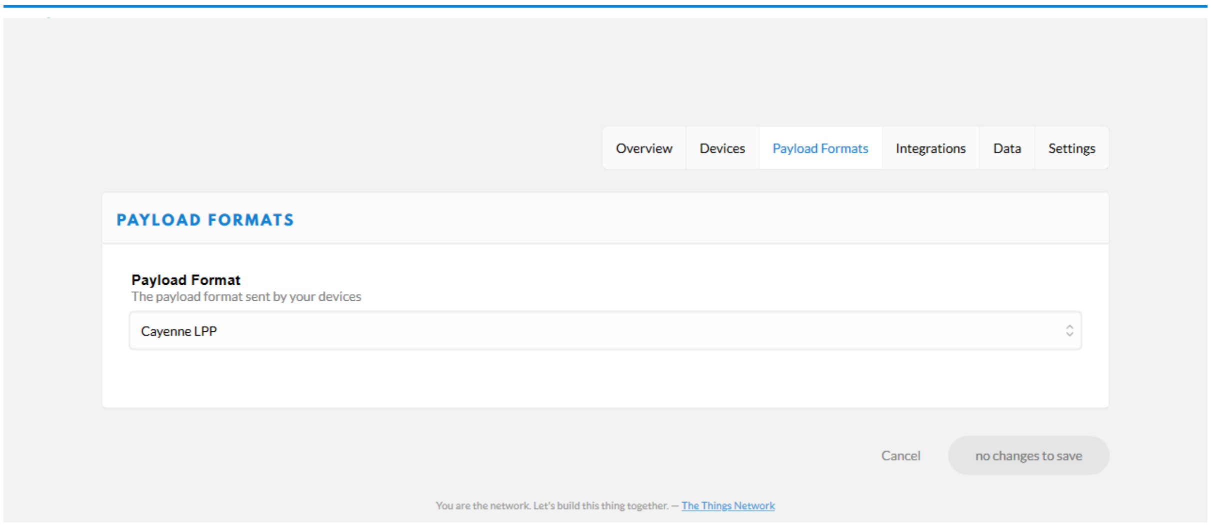
```
const char *appEui = "70B3D57ED00268AF";
const char *appKey = "66AF212A77EE7C0A89FFCA3F2A333779";
```

You are the network. Let's build this thing together. – The Things Network

For this device, set up to use Cayenne payload, so TTN can parse the sensor data properly.

Return to 'Devices' and click on 'Payload Formats'.

Select Cayenne LPP from the dropdown. Click the green ‘Save’ button.



PAYLOAD FORMATS

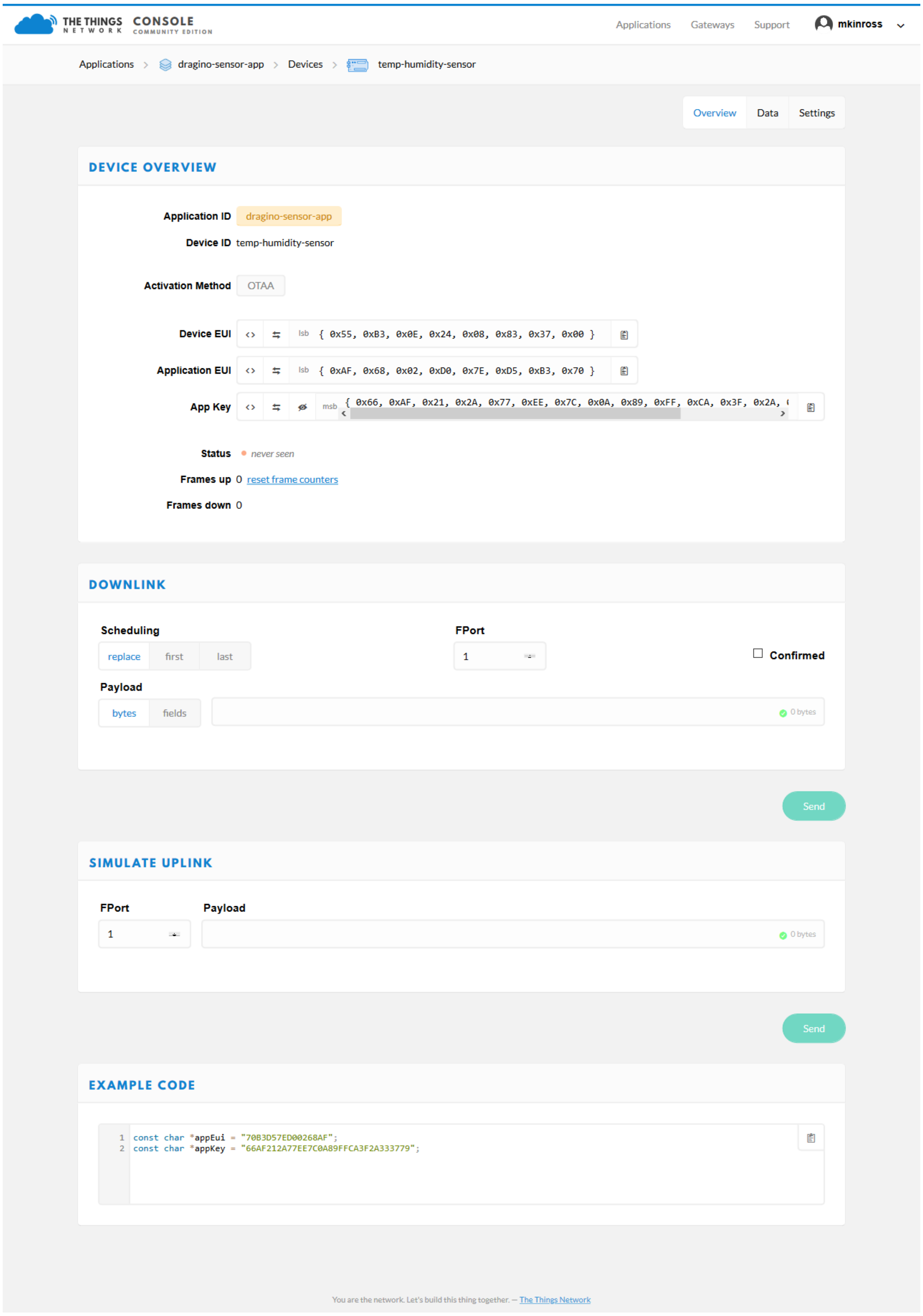
Payload Format
The payload format sent by your devices

Cayenne LPP

Cancel
no changes to save
You are the network. Let's build this thing together. — The Things Network

All the devices within this application will now use the Cayenne LPP payload format.

Your device overview should now look something like the image below.


THE THINGS NETWORK CONSOLE COMMUNITY EDITION
Applications
Gateways
Support
mkinross
Applications > dragino-sensor-app > Devices > temp-humidity-sensor
Overview
Data
Settings

DEVICE OVERVIEW

Application ID	dragino-sensor-app
Device ID	temp-humidity-sensor
Activation Method	OTAA
Device EUI	lsb { 0x55, 0xB3, 0x0E, 0x24, 0x08, 0x83, 0x37, 0x00 }
Application EUI	lsb { 0xAF, 0x68, 0x02, 0xD0, 0x7E, 0xD5, 0xB3, 0x70 }
App Key	msb { 0x66, 0xAF, 0x21, 0x2A, 0x77, 0xEE, 0x7C, 0x0A, 0x89, 0xFF, 0xCA, 0x3F, 0x2A,
Status	● *never seen*
Frames up	0 reset frame counters
Frames down	0

DOWNLINK

Scheduling: replace first last
FPort: 1
☐ Confirmed
Payload: bytes fields — 0 bytes

SIMULATE UPLINK

FPort: 1
Payload: 0 bytes

EXAMPLE CODE

```
const char *appEui = "70B3D57ED00268AF";
const char *appKey = "66AF212A77EE7C0A89FFCA3F2A333779";
```

You are the network. Let's build this thing together. – The Things Network

Do you remember the Arduino LMIC library we downloaded?

(Tutorial page approx. 20)

The Arduino-LMIC library

The first one we will download is the Arduino-LMIC library. This is used to configure the Arduino as a standard LoRaWAN end node. A 'node' is the end part of the Internet of Things that connects to the things, such as sensors.

We now need to open the library to make sure the settings are configured to work with our application.

Make sure that your Arduino and shield are plugged into your computer. Then open the Arduino IDE.

Open your copy of the LMIC library. You will find it with the 'Contributed libraries' which are stored in your Documents folder:

~\Documents\Arduino\libraries

We need to change the Frequency Band to use with LG01-N. The change is in the file

arduino\libraries\arduino-lmic \src\lmic\config.h

In my installation, the path was:

~\Documents\Arduino\libraries\arduino-lmic-master\src\lmic\config.h

You will have to open this in a text editor such as Notepad++, as the Arduino IDE does not recognise this type of file.

All possible options are listed in the code below. What we need to do is uncomment the lines we need so that the code will run. A commented line starts with two forward slashes //. To uncomment the line, we just remove the appropriate //.

```
#ifndef _lmic_config_h_
#define _lmic_config_h_

// In the original LMIC code, these config values were defined on the
// gcc commandline. Since Arduino does not allow easily modifying the
// compiler commandline, use this file instead.
```

```
#define CFG_eu868 1
```
Uncomment this line to use European frequencies which should be same as server

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

```
#define LG02_LG01 1
```
Uncomment this line for LG01 and LG02

Everything below should be OK

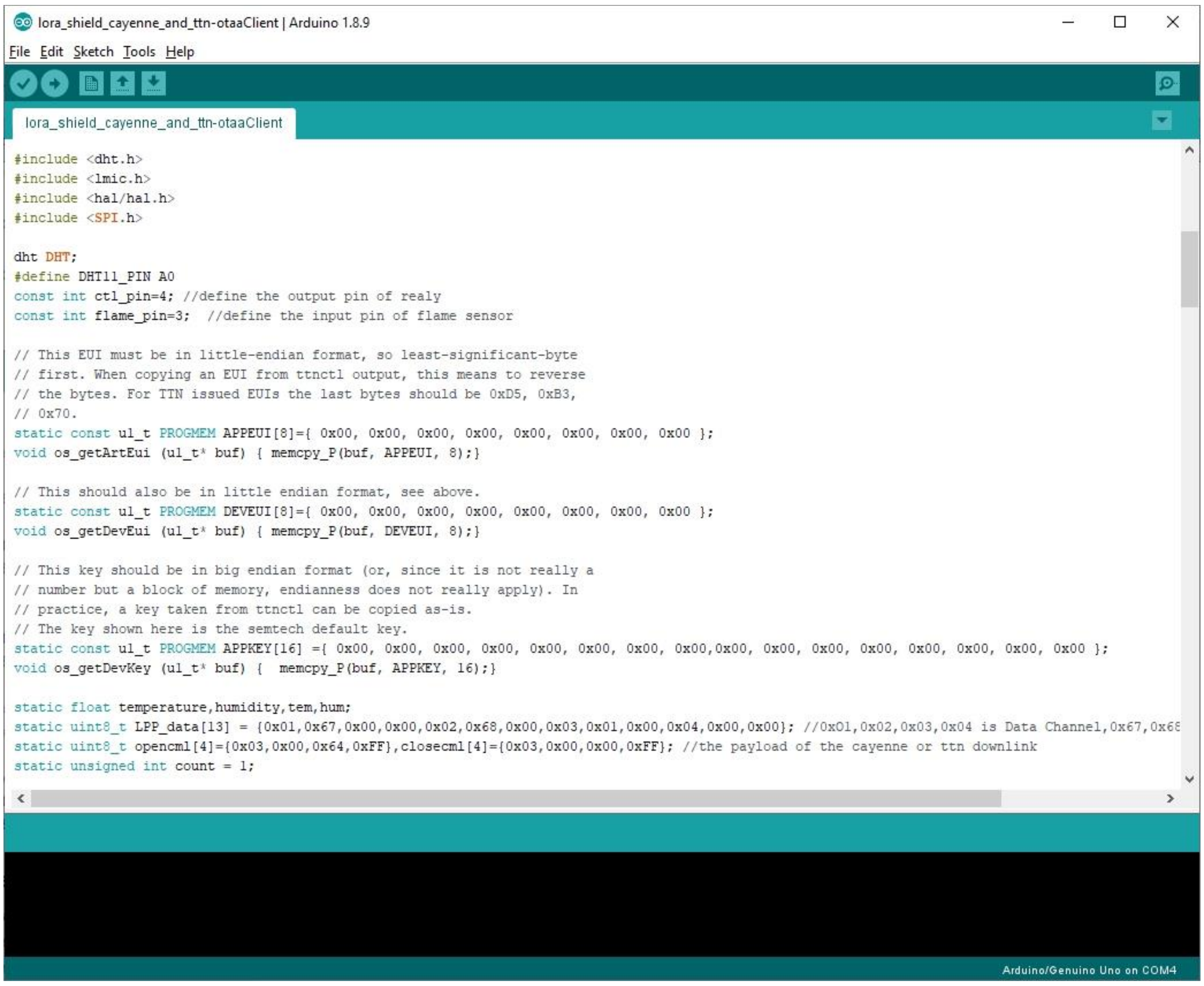
```
//US915: DR_SF10=0, DR_SF9=1, DR_SF8=2, DR_SF7=3, DR_SF8C=4
//        DR_SF12CR=8, DR_SF11CR=9, DR_SF10CR=10, DR_SF9CR=11, DR_SF8CR=12, DR_SF7CR
#if defined(CFG_us915) && defined(LG02_LG01)
// CFG_us915 || CFG_as923
#define LG02_UPFREQ   902320000
#define LG02_DNWFREQ  923300000
#define LG02_RXSF     3      // DR_SF7  For LG01/LG02 Tx
#define LG02_TXSF     8      // DR_SF12CR For LG02/LG02 Rx
#elif defined(CFG_eu868) && defined(LG02_LG01)
// CFG_eu868
//EU868: DR_SF12=0, DR_SF11=1, DR_SF10=2, DR_SF9=3, DR_SF8=4, DR_SF7=5, DR_SF7B=1, DR_FSK, DR_NONE
#define LG02_UPFREQ   868100000
#define LG02_DNWFREQ  869525000
#define LG02_RXSF     5      // DR_SF7 For LG01/LG02 Tx
#define LG02_TXSF     0       // DR_SF12 For LG02/LG02 Rx
```

```
#endif

// Set this to 1 to enable some basic debug output (using printf) about
// RF settings used during transmission and reception. Set to 2 to
// enable more verbose output. Make sure that printf is actually
// configured (e.g. on AVR it is not by default), otherwise using it can
// cause crashing.
#define LMIC_DEBUG_LEVEL 1

// This is the SX1272/SX1273 radio, which is also used on the HopeRF
// RFM92 boards.
//#define CFG_sx1272_radio 1
// This is the SX1276/SX1277/SX1278/SX1279 radio, which is also used on
// the HopeRF RFM95 boards.
#define CFG_sx1276_radio 1

// 16 μs per tick
// LMIC requires ticks to be 15.5μs - 100 μs long
#define US_PER_OSTICK_EXPONENT 4
#define US_PER_OSTICK (1 << US_PER_OSTICK_EXPONENT)
#define OSTICKS_PER_SEC (1000000 / US_PER_OSTICK)


// Enable this to allow using printf() to print to the given serial port
// (or any other Print object). This can be easy for debugging. The
// current implementation only works on AVR, though.
#if defined(LMIC_DEBUG_LEVEL)
#define LMIC_PRINTF_TO Serial
#endif

// Any runtime assertion failures are printed to this serial port (or
// any other Print object). If this is unset, any failures just silently
// halt execution.
#define LMIC_FAILURE_TO Serial

// Uncomment this to disable all code related to joining
//#define DISABLE_JOIN
// Uncomment this to disable all code related to ping
//#define DISABLE_PING
// Uncomment this to disable all code related to beacon tracking.
// Requires ping to be disabled too
//#define DISABLE_BEACONS

// Uncomment these to disable the corresponding MAC commands.
// Class A
//#define DISABLE_MCMD_DCAP_REQ // duty cycle cap
//#define DISABLE_MCMD_DN2P_SET // 2nd DN window param
//#define DISABLE_MCMD_SNCH_REQ // set new channel
// Class B
//#define DISABLE_MCMD_PING_SET // set ping freq, automatically disabled by DISABLE_PING
//#define DISABLE_MCMD_BCNI_ANS // next beacon start, automatical disabled by DISABLE_BEACON

// In LoRaWAN, a gateway applies I/Q inversion on TX, and nodes do the
// same on RX. This ensures that gateways can talk to nodes and vice
// versa, but gateways will not hear other gateways and nodes will not
// hear other nodes. By uncommenting this macro, this inversion is
// disabled and this node can hear other nodes. If two nodes both have
// this macro set, they can talk to each other (but they can no longer
// hear gateways). This should probably only be used when debugging
// and/or when talking to the radio directly (e.g. like in the "raw"
// example).
//#define DISABLE_INVERT_IQ_ON_RX

// This allows choosing between multiple included AES implementations.
// Make sure exactly one of these is uncommented.
//
// This selects the original AES implementation included LMIC. This
// implementation is optimized for speed on 32-bit processors using
// fairly big lookup tables, but it takes up big amounts of flash on the
// AVR architecture.
// #define USE_ORIGINAL_AES
//
// This selects the AES implementation written by Ideetron for their
// own LoRaWAN library. It also uses lookup tables, but smaller
// byte-oriented ones, making it use a lot less flash space (but it is
// also about twice as slow as the original).
#define USE_IDEETRON_AES

#endif // _lmic_config_h_
```

(Tutorial page approx. 21)

Download and open the sketch from Github https://github.com/dragino/Arduino-Profile-Examples/blob/master/libraries/Dragino/examples/IoTServer/Cayenne%20and%20TTN/cayenne%20and%20ttn%20example/lora_shield%20DHT11%20and%20Relay%20examples/lora_shield_cayenne_and_ttn-otaaClient/lora_shield_cayenne_and_ttn-otaaClient.ino.

When you open the sketch in your Arduino IDE, you will see something like this:


lora_shield_cayenne_and_ttn-otaaClient | Arduino 1.8.9
File Edit Sketch Tools Help
lora_shield_cayenne_and_ttn-otaaClient
Arduino/Genuino Uno on COM4

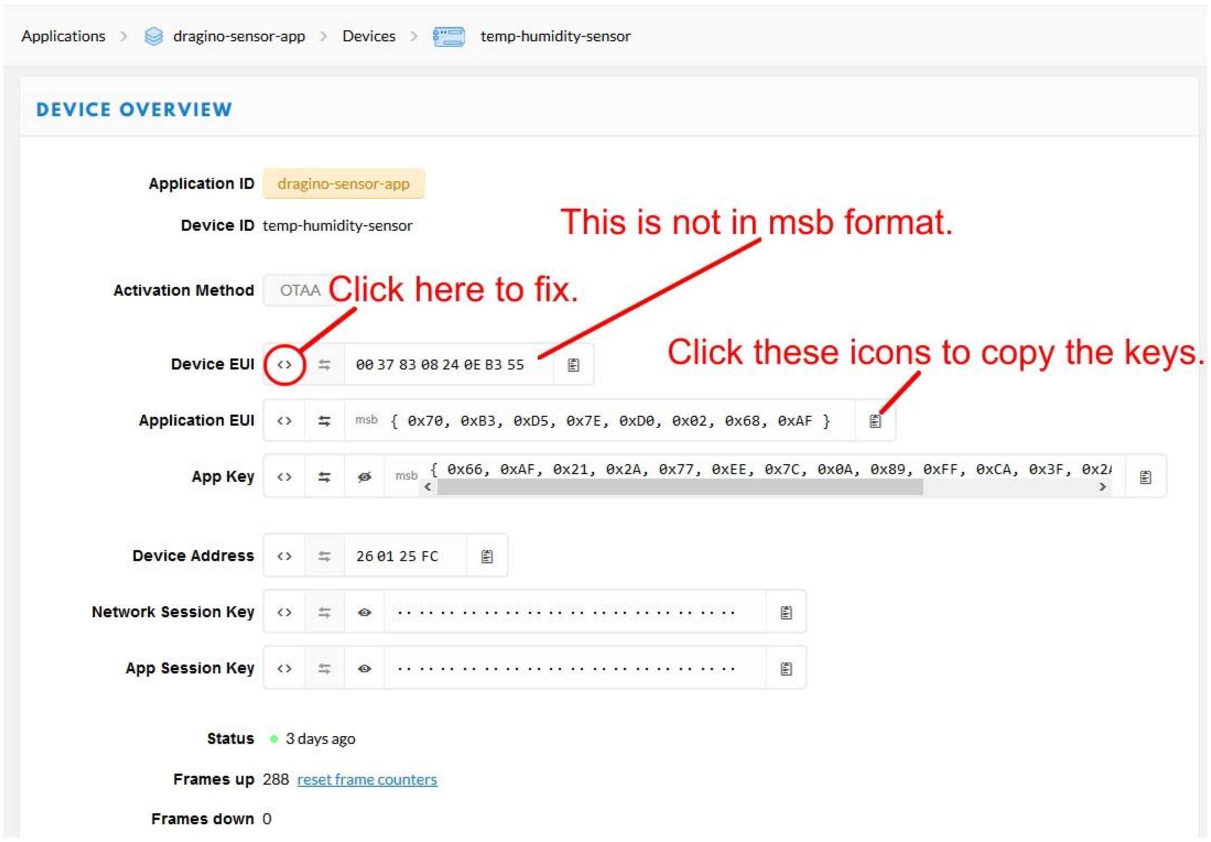
```
#include <dht.h>
#include <lmic.h>
#include <hal/hal.h>
#include <SPI.h>

dht DHT;
#define DHT11_PIN A0
const int ctl_pin=4; //define the output pin of realy
const int flame_pin=3;  //define the input pin of flame sensor

// This EUI must be in little-endian format, so least-significant-byte
// first. When copying an EUI from ttnctl output, this means to reverse
// the bytes. For TTN issued EUIs the last bytes should be 0xD5, 0xB3,
// 0x70.
static const u1_t PROGMEM APPEUI[8]={ 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00 };
void os_getArtEui (u1_t* buf) { memcpy_P(buf, APPEUI, 8);}

// This should also be in little endian format, see above.
static const u1_t PROGMEM DEVEUI[8]={ 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00 };
void os_getDevEui (u1_t* buf) { memcpy_P(buf, DEVEUI, 8);}

// This key should be in big endian format (or, since it is not really a
// number but a block of memory, endianness does not really apply). In
// practice, a key taken from ttnctl can be copied as-is.
// The key shown here is the semtech default key.
static const u1_t PROGMEM APPKEY[16] ={ 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00 };
void os_getDevKey (u1_t* buf) {  memcpy_P(buf, APPKEY, 16);}

static float temperature,humidity,tem,hum;
static uint8_t LPP_data[13] = {0x01,0x67,0x00,0x00,0x02,0x68,0x00,0x03,0x01,0x00,0x04,0x00,0x00}; //0x01,0x02,0x03,0x04 is Data Channel,0x67,0x68
static uint8_t opencml[4]={0x03,0x00,0x64,0xFF},closecml[4]={0x03,0x00,0x00,0xFF}; //the payload of the cayenne or ttn downlink
static unsigned int count = 1;
```

Look for the lines that start ‘static const’, and end with ‘… 0x00, 0x00, 0x00 };’.

We need to modify these keys to match the keys in the TTN console. Get the Device EUI, the Application EUI and the APP Key from your console. Copy the keys and paste them into your sketch.

You can copy the keys by clicking on the icons at the end of the keys.

Make sure all the keys are in ‘msb’ format.



Applications > dragino-sensor-app > Devices > temp-humidity-sensor

DEVICE OVERVIEW

Application ID dragino-sensor-app

Device ID temp-humidity-sensor

Activation Method OTAA

Device EUI 00 37 83 08 24 0E B3 55

Application EUI msb { 0x70, 0xB3, 0xD5, 0x7E, 0xD0, 0x02, 0x68, 0xAF }

App Key msb { 0x66, 0xAF, 0x21, 0x2A, 0x77, 0xEE, 0x7C, 0x0A, 0x89, 0xFF, 0xCA, 0x3F, 0x2A

Device Address 26 01 25 FC

Network Session Key ••••••••••••••••••••••••••••••••

App Session Key ••••••••••••••••••••••••••••••••

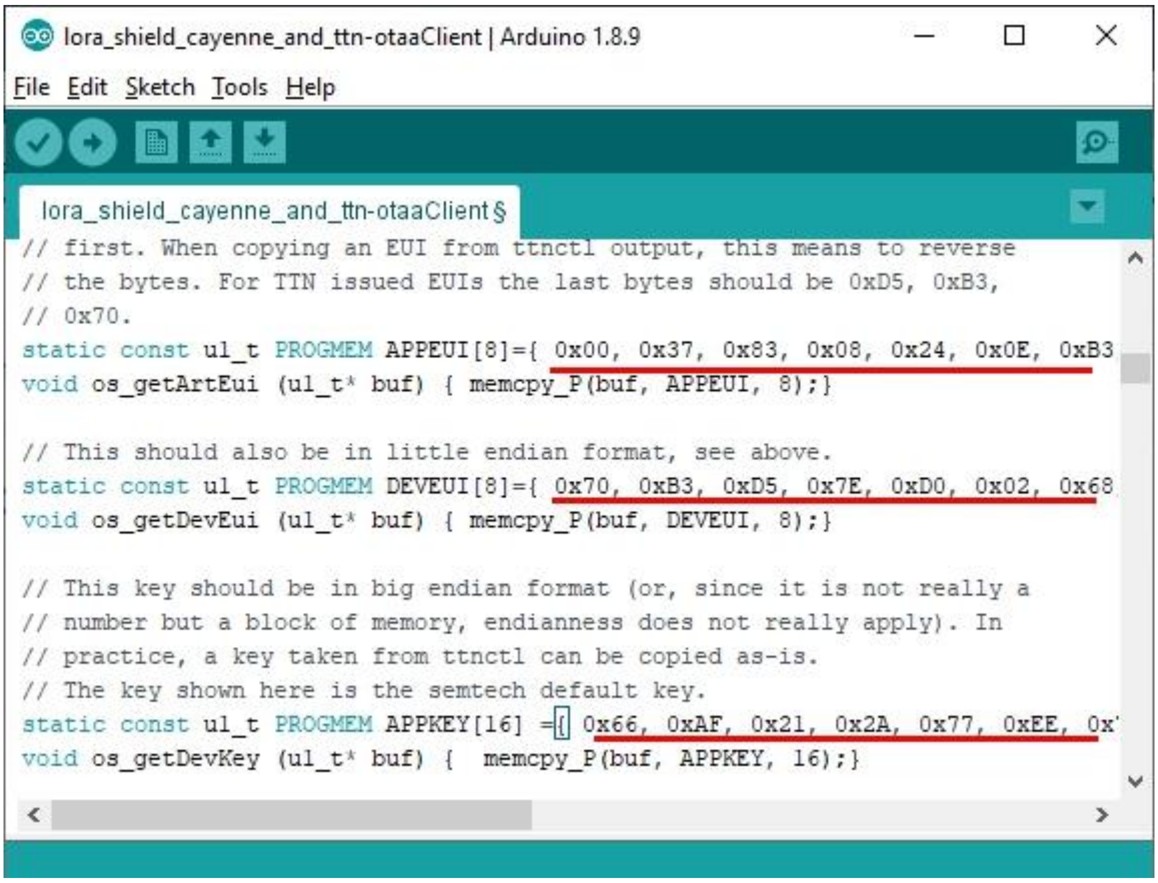
Status 3 days ago

Frames up 288 reset frame counters

Frames down 0

This is not in msb format.
Click here to fix.
Click these icons to copy the keys.

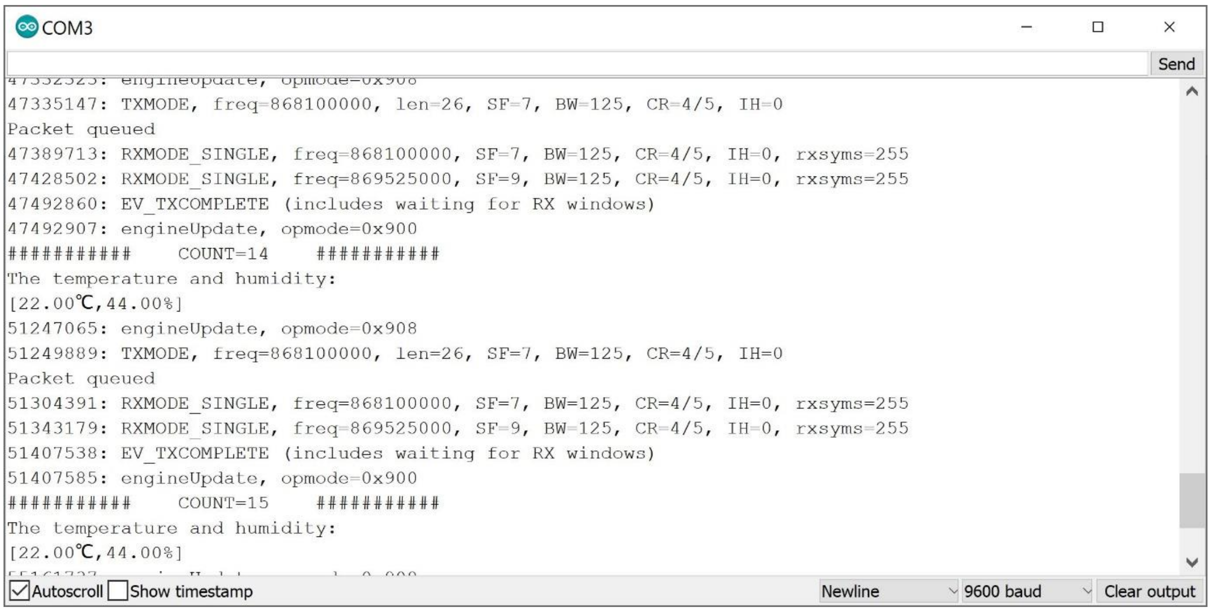
Your sketch should look more like this once you have finished pasting in the keys:



lora_shield_cayenne_and_ttn-otaaClient | Arduino 1.8.9
File Edit Sketch Tools Help
lora_shield_cayenne_and_ttn-otaaClient §

```
// first. When copying an EUI from ttnctl output, this means to reverse
// the bytes. For TTN issued EUIs the last bytes should be 0xD5, 0xB3,
// 0x70.
static const u1_t PROGMEM APPEUI[8]={ 0x00, 0x37, 0x83, 0x08, 0x24, 0x0E, 0xB3
void os_getArtEui (u1_t* buf) { memcpy_P(buf, APPEUI, 8);}

// This should also be in little endian format, see above.
static const u1_t PROGMEM DEVEUI[8]={ 0x70, 0xB3, 0xD5, 0x7E, 0xD0, 0x02, 0x68
void os_getDevEui (u1_t* buf) { memcpy_P(buf, DEVEUI, 8);}

// This key should be in big endian format (or, since it is not really a
// number but a block of memory, endianness does not really apply). In
// practice, a key taken from ttnctl can be copied as-is.
// The key shown here is the semtech default key.
static const u1_t PROGMEM APPKEY[16] ={ 0x66, 0xAF, 0x21, 0x2A, 0x77, 0xEE, 0x
void os_getDevKey (u1_t* buf) {  memcpy_P(buf, APPKEY, 16);}
```

Now upload the code to the Arduino UNO by clicking the green right arrow (upload) button at the top left of the IDE. If you open the serial monitor, you should see something like this.



```
47335147: TXMODE, freq=868100000, len=26, SF=7, BW=125, CR=4/5, IH=0
Packet queued
47389713: RXMODE_SINGLE, freq=868100000, SF=7, BW=125, CR=4/5, IH=0, rxsyms=255
47428502: RXMODE_SINGLE, freq=869525000, SF=9, BW=125, CR=4/5, IH=0, rxsyms=255
47492860: EV_TXCOMPLETE (includes waiting for RX windows)
47492907: engineUpdate, opmode=0x900
###########     COUNT=14     ###########
The temperature and humidity:
[22.00℃,44.00%]
51247065: engineUpdate, opmode=0x908
51249889: TXMODE, freq=868100000, len=26, SF=7, BW=125, CR=4/5, IH=0
Packet queued
51304391: RXMODE_SINGLE, freq=868100000, SF=7, BW=125, CR=4/5, IH=0, rxsyms=255
51343179: RXMODE_SINGLE, freq=869525000, SF=9, BW=125, CR=4/5, IH=0, rxsyms=255
51407538: EV_TXCOMPLETE (includes waiting for RX windows)
51407585: engineUpdate, opmode=0x900
###########     COUNT=15     ###########
The temperature and humidity:
[22.00℃,44.00%]
```

From gateway logread, we can see the data send from end node. (Login to the Gateway to see this.)

Screenshot

Connect the Dragino gateway to your computer using a RJ45 cable (an ordinary ethernet cable).

Open your Chrome browser and go to http://10.130.1.1/cgi-bin/luci/admin

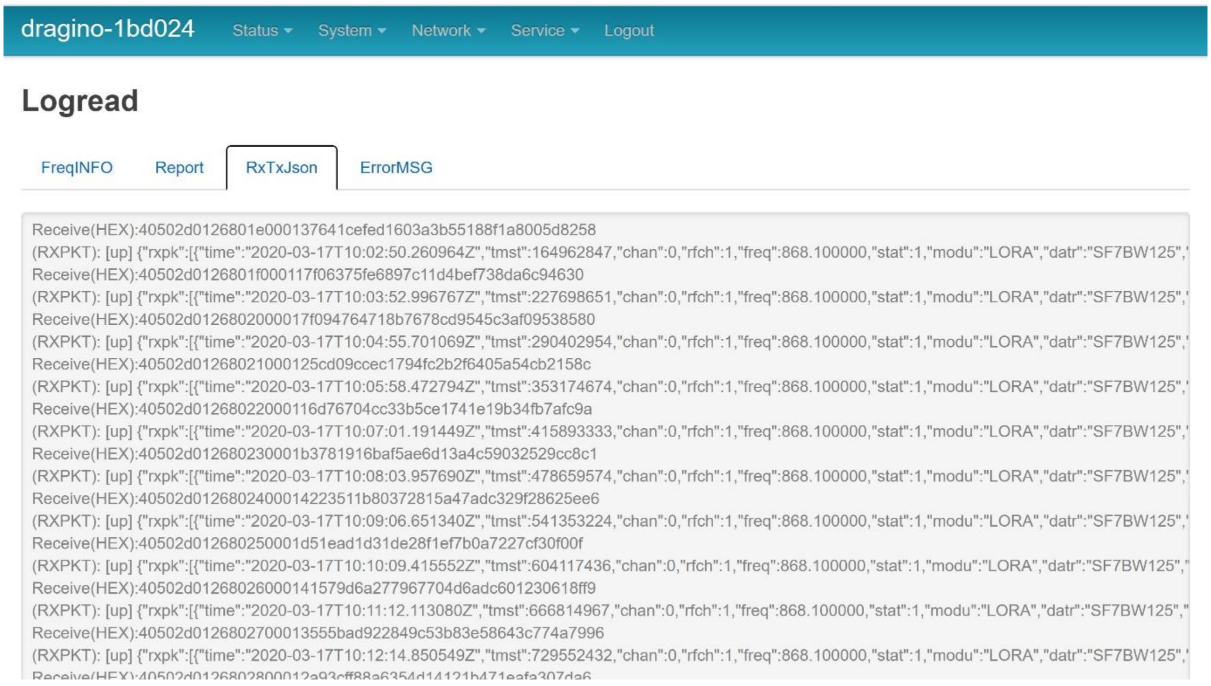
You may have to allow the connection before you get to the Dragino login page.

The account for Web Login is:

User Name: root

Password: dragino

Once logged in, go to ‘Service’ > ‘Logread’ > ‘RxTxJson’, and you should see a page like this:



```
Receive(HEX):40502d0126801e000137641cefed1603a3b55188f1a8005d8258
(RXPKT): [up] {"rxpk":[{"time":"2020-03-17T10:02:50.260964Z","tmst":164962847,"chan":0,"rfch":1,"freq":868.100000,"stat":1,"modu":"LORA","datr":"SF7BW125",'
Receive(HEX):40502d0126801f000117f06375fe6897c11d4bef738da6c94630
(RXPKT): [up] {"rxpk":[{"time":"2020-03-17T10:03:52.996767Z","tmst":227698651,"chan":0,"rfch":1,"freq":868.100000,"stat":1,"modu":"LORA","datr":"SF7BW125",'
Receive(HEX):40502d0126802000017f094764718b7678cd9545c3af09538580
(RXPKT): [up] {"rxpk":[{"time":"2020-03-17T10:04:55.701069Z","tmst":290402954,"chan":0,"rfch":1,"freq":868.100000,"stat":1,"modu":"LORA","datr":"SF7BW125",'
Receive(HEX):40502d01268021000125cd09ccec1794fc2b2f6405a54cb2158c
(RXPKT): [up] {"rxpk":[{"time":"2020-03-17T10:05:58.472794Z","tmst":353174674,"chan":0,"rfch":1,"freq":868.100000,"stat":1,"modu":"LORA","datr":"SF7BW125",'
Receive(HEX):40502d01268022000116d76704cc33b5ce1741e19b34fb7afc9a
(RXPKT): [up] {"rxpk":[{"time":"2020-03-17T10:07:01.191449Z","tmst":415893333,"chan":0,"rfch":1,"freq":868.100000,"stat":1,"modu":"LORA","datr":"SF7BW125",'
Receive(HEX):40502d012680230001b3781916baf5ae6d13a4c59032529cc8c1
(RXPKT): [up] {"rxpk":[{"time":"2020-03-17T10:08:03.957690Z","tmst":478659574,"chan":0,"rfch":1,"freq":868.100000,"stat":1,"modu":"LORA","datr":"SF7BW125",'
Receive(HEX):40502d0126802400014223511b80372815a47adc329f28625ee6
(RXPKT): [up] {"rxpk":[{"time":"2020-03-17T10:09:06.651340Z","tmst":541353224,"chan":0,"rfch":1,"freq":868.100000,"stat":1,"modu":"LORA","datr":"SF7BW125",'
Receive(HEX):40502d012680250001d51ead1d31de28f1ef7b0a7227cf30f00f
(RXPKT): [up] {"rxpk":[{"time":"2020-03-17T10:10:09.415552Z","tmst":604117436,"chan":0,"rfch":1,"freq":868.100000,"stat":1,"modu":"LORA","datr":"SF7BW125","
Receive(HEX):40502d0126802600014157 9d6a277967704d6adc601230618ff9
(RXPKT): [up] {"rxpk":[{"time":"2020-03-17T10:11:12.113080Z","tmst":666814967,"chan":0,"rfch":1,"freq":868.100000,"stat":1,"modu":"LORA","datr":"SF7BW125","
Receive(HEX):40502d0126802700013555bad922849c53b83e58643c774a7996
(RXPKT): [up] {"rxpk":[{"time":"2020-03-17T10:12:14.850549Z","tmst":729552432,"chan":0,"rfch":1,"freq":868.100000,"stat":1,"modu":"LORA","datr":"SF7BW125",'
Receive(HEX):40502d0126802800012a93cff88a6354d14121b471eafa307da6
```

In TTN-Gateway page, we can also see the traffic.

From console – gateways > your gateway > traffic.

Explain the significance of each column.

Screenshot

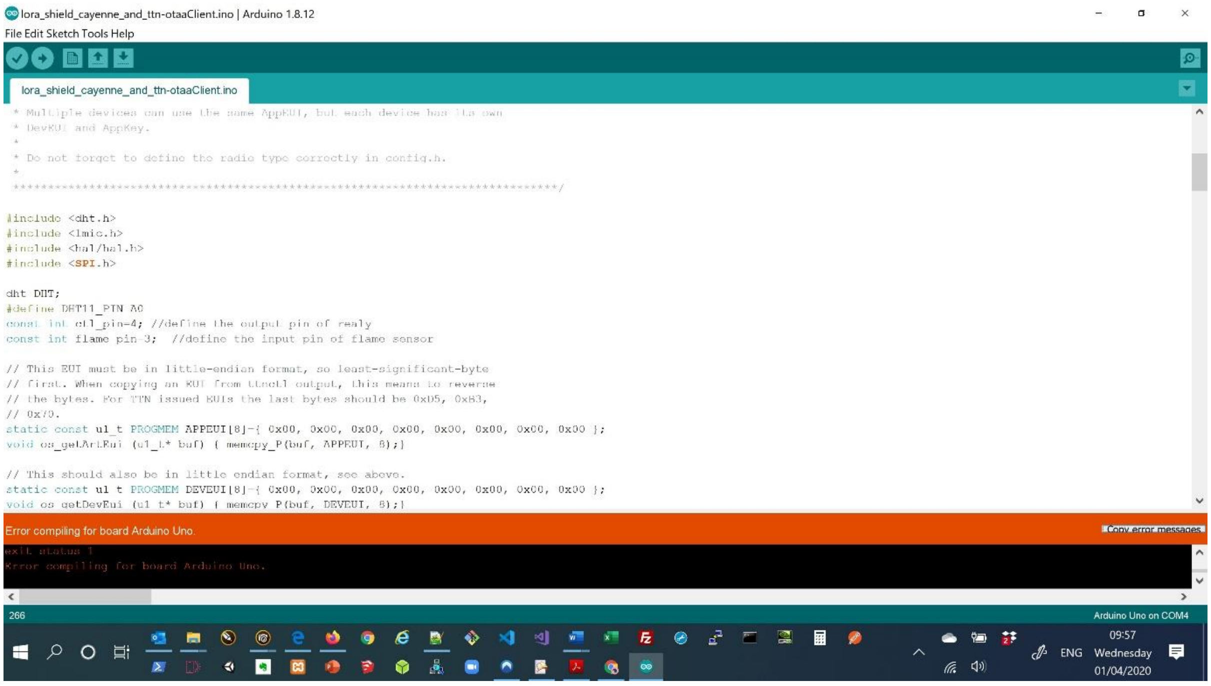
Configure to connect to Mydevices Application Server

(Tutorial page approx. 21)

Problems

Compilation error

If you see a compile error like that below:



lora_shield_cayenne_and_ttn-otaaClient.ino | Arduino 1.8.12

```
 * Multiple devices can use the same AppEUI, but each device has its own
 * DevEUI and AppKey.
 *
 * Do not forget to define the radio type correctly in config.h.
 *
 *******************************************************************************/

#include <dht.h>
#include <lmic.h>
#include <hal/hal.h>
#include <SPI.h>

dht DHT;
#define DHT11_PIN A0
const int ctl_pin=4; //define the output pin of realy
const int flame_pin=3;  //define the input pin of flame sensor

// This EUI must be in little-endian format, so least-significant-byte
// first. When copying an EUI from ttnctl output, this means to reverse
// the bytes. For TTN issued EUIs the last bytes should be 0xD5, 0xB3,
// 0x70.
static const u1_t PROGMEM APPEUI[8]={ 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00 };
void os_getArtEui (u1_t* buf) { memcpy_P(buf, APPEUI, 8);}

// This should also be in little endian format, see above.
static const u1_t PROGMEM DEVEUI[8]={ 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00 };
void os_getDevEui (u1_t* buf) { memcpy_P(buf, DEVEUI, 8);}
```

```
Error compiling for board Arduino Uno.

exit status 1
Error compiling for board Arduino Uno.
```

Then you will need to carry out the following fix:

1) Navigate to C:\Users\yourUserName\Documents\Arduino\libraries\arduino-lmic-master\src\lmic

2) Open the file (oslmic.h) using a decent editor (e.g. notepad++); do not use notepad or word and also do not use the IDE.

3) Search for inline type table_get ## postfix; it should be found on line 230 with the latest version (I downloaded it about 15 minutes ago).

Add the static keyword in front of it so the line looks like:

```
static inline type table_get ## postfix(const type *table, size_t index) { \
```

4) Save the file.

5) Compile the sketch.

https://forum.arduino.cc/index.php?topic=645057.0

OTAA network access problem

The over the air activation system has a intermittent bug which means that occasionally the order to the EUI keys gets reversed.

If you login to your TTN console and view the Gateway page to find that the gateway is receiving join requests, but not sending join replies, check the EUI keys.

If the order of the keys in the gateway traffic page is a reverse of the order in the devices overview page, you will have to manually reverse them before uploading the sketch to the Arduino.

So an EUI key in the device overview might look like:

```
{0xAE, 0xCD, 0x48, 0x55, 0x32, 0x77, 0xCD, 0xAB},
```

But in the sketch, you will have to manually type:

```
{0xAB, 0xCD, 0x77, 0x32, 0x55, 0x48, 0xCD, 0xAE},
```

This only seems to apply to the APPEUI and DEVEUI keys. You can leave the APPKEY as it is.

You can find more information here:

https://wiki.dragino.com/index.php?title=Connect_to_ThingPark

This article to be continued…