



Instituto Politécnico de Castelo Branco
Escola Superior de Tecnologia

ENGENHARIA ELECTROTÉCNICA E DAS TELECOMUNICAÇÕES

R.D.R PROJECT

3ºANO – 1º SEMESTRE

AULAS TEÓRICO-PRÁTICAS

LABORATORY

DOCENTE:

PAULO TORRES

EDIÇÃO REVISTA:

PAULO TORRES

Summary

1	Introduction	Erreur ! Signet non défini.
2	Database	3
2.1	conception	Erreur ! Signet non défini.
2.2	Creation of database	Erreur ! Signet non défini.
3	Python script	Erreur ! Signet non défini.
4	Webpages	Erreur ! Signet non défini.

1 PREAMBLE

LoRa Technology is a wireless modulation for long-range, low-power, low-data-rate applications. By achieving a range of more than 15 kilometers in a suburban environment and more than 2 kilometers in a dense urban environment, LoRa technology solutions target multiple application domains, such as Internet-of-Things (IoT), metering, security, and machine-to-machine (M2M).

This single channel LoRaWAN gateway is a proof-of-concept implementation, that can be used for development and node testing. It is not a replacement for a real multi-channel/multi SF gateway! It supports some LoRaWAN features, but due to its static nature (single channel) it is not fully LoRaWAN compatible (and will never be).

By default, it works with the TTN backend, for testing and development.

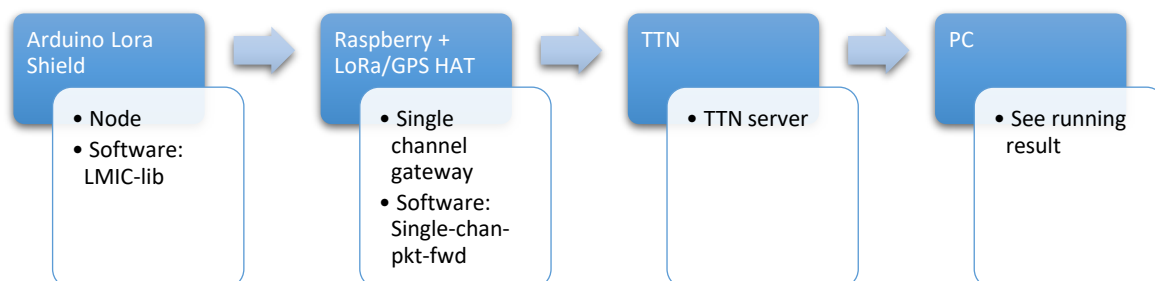
2 INTRODUCTION

In this document, we will talk about:

- How to use LoRa/GPS HAT to set up single channel gateway for TTN network.
- How to use LoRa Shield to set up a LoRa Node.
- How is the communication between the LoRa Node and LoRa Gateway.

We will use the LoRa Shield + Arduino UNO and LoRa/GPS HAT + Raspberry pi 3 to build a single channel LoRaWAN gateway.

3 NETWORK STRUCTURE



The construction of the network is as graphic above.

LoRa Node:

The Arduino UNO will get sensor data and control the LoRa Shield to send this data to the RPi Lora Gateway via LoRa wireless protocol.

LoRa Gateway:

The RPi LoRa Gateway will receive this data and upload it to the TTN network via the Internet.

TTN Server:

The TTN Server will get the data packets from the RPi LoRa Gateway and the data will be stored in the corresponding place, so users can take what they need from the Internet.

PC:

We can use the PC to get the data and check the status of this LoRa Gateway network.

4 BUILD A SINGLE CHANNEL LORAWAN GATEWAY

In this step, we will use the RPi and the LoRa/GPS HAT to build a single channel LoRaWAN Gateway. We should configure the RPi and connect it with the LoRa/GPS HAT.

First, you need to put two jumpers like this:

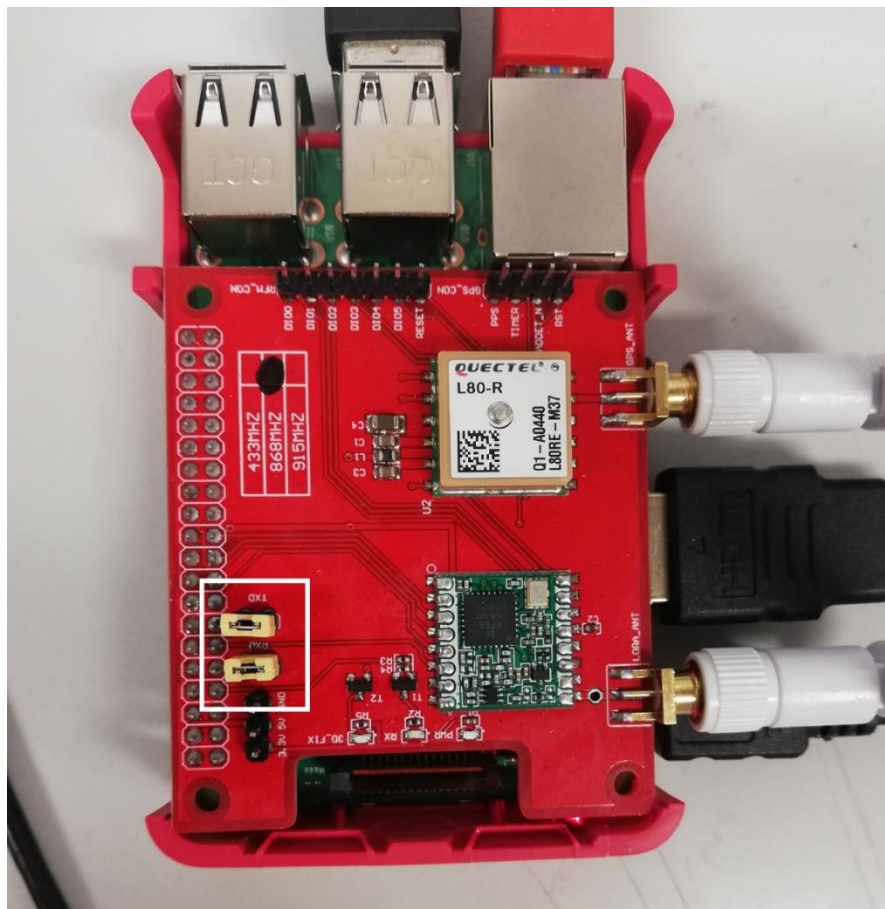


Fig 1: Dragino LoRa HAT on raspberry pi 3

CONFIGURATION

- Connect the Raspberry Pi to the Internet.
- Use `sudo raspi-config` to ensure that SPI can be used on RPi.
- Use `sudo apt-get install wiringpi` to install the GPIO access library written in C for the BCM2835 used in the Raspberry Pi.
- Get the single channel Lora Gateway source code with this command :
`Git clone https://github.com/tftelkamp/single_chan_pkt_fwd.git`
`Sudo chmod 777 single_chan_pkt_fwd`
- Edit the 'main.cpp' to change configuration (look for: "*Configure these values!*").

```
// SX1272 - Raspberry connections
int ssPin = 6;
int dio0 = 7;
int RST = 0;

// Set spreading factor (SF7 - SF12)
sf_t sf = SF7;

// Set center frequency
uint32_t freq = 868100000; // in Mhz! (868.1)

// Set location
float lat=0.0;
float lon=0.0;
int alt=0;

/* Informal status fields */
static char platform[24] = "Single Channel Gateway"; /* platform definition */
static char email[40] = ""; /* used for contact email */
static char description[64] = ""; /* used for free form description */

// define servers
// TODO: use host names and dns
#define SERVER1 "0.0.0.0" // The Things Network: croft.thethings.girotito.nl
// #define SERVER2 "192.168.1.10" // local
#define PORT 1700 // The port on which to send data
```

Fig 2: C++ single channel gateway program

- You just need to `SERVER1`, put the public IP of your router.

Connect the Dragino LoRa/GPS HAT and the RPi, run packet forwarder as root. To do this use this command:

```
make
./single_chan_pkt_fwd
```

Then we can get a Gateway ID and see the running result on the RPi as below picture.

```
pi@pi:~/Desktop/LoRa/single_chan_pkt_fwd $ ./single_chan_pkt_fwd
SX1276 detected, starting.
Gateway ID: b8:27:eb:ff:ff:c2:2b:70
Listening at SF7 on 868.100000 Mhz.
-----
stat update: {"stat":{"time":"2019-12-05 16:22:03 GMT","lati":0.00000,"loni":0.00000,"alt":0,"stat":"Single Channel Gateway","mail":"","desc":""}}
```

Fig 3: single_chan_pkt_fwd program

Keep this ID for the next step.

5 CREATE A GATEWAY ON TTN NETWORK

Before we start this project we must have a TTN account, we can create one at this page: <https://account.thethingsnetwork.org/register>.

After registration, you need to go on *console* part, you will see this page:

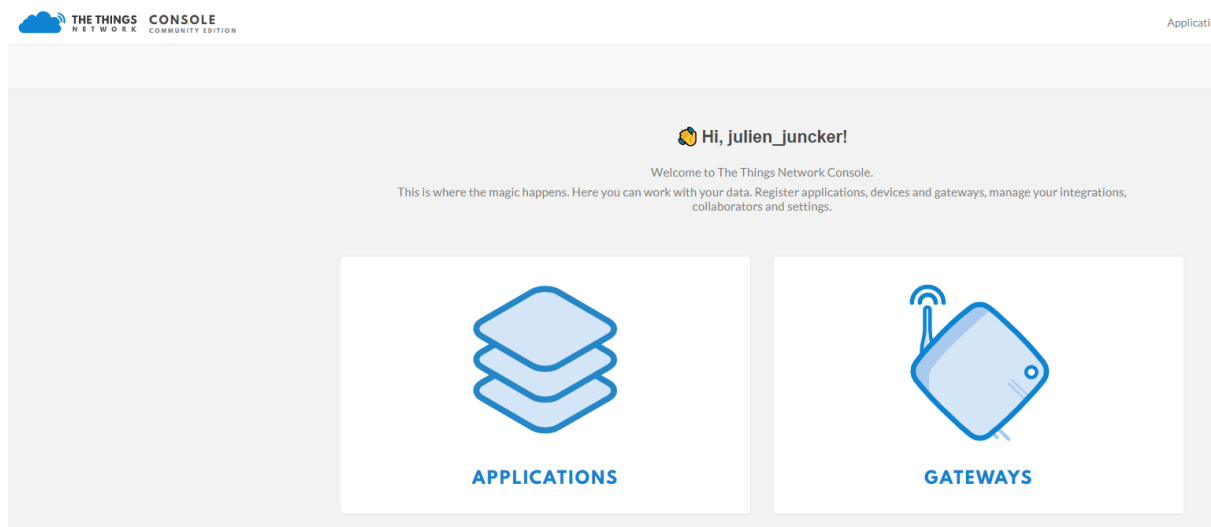


Fig 4: Console view on TTN network

Click on *gateway* and click on *register gateway* link:

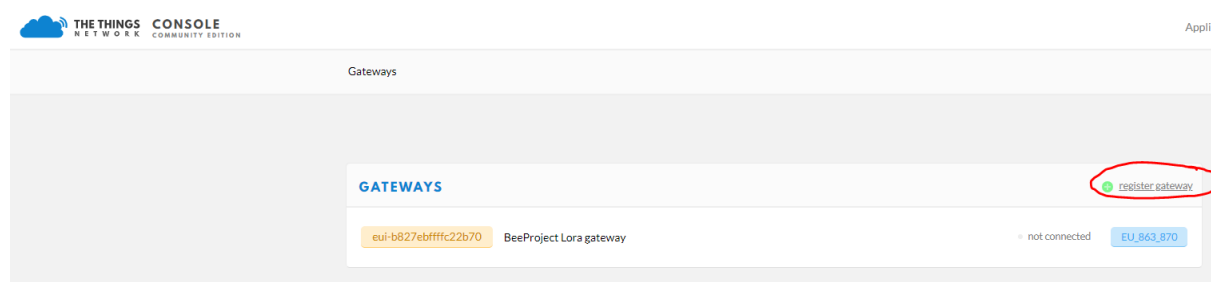


Fig 5: register Gateway button

You will see gateway registering page:

Gateways > Register

REGISTER GATEWAY

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

B8 27 EB FF FF C2 2B 70 8 bytes

☒ **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

Frequency Plan
The [frequency plan](#) this gateway will use

no selection

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.

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.




Fig 6: Register Gateway

- Click on checkbox « I'M USING THE LEGACY PACKET FORWARDER ».
- Fill the *Gateway EUI* label with your own Gateway ID that you take on last part.
- You can put description if you want.
- For *Frequency Plan*, put European frequency, it's *868MHz*.
- For *Router*, put *ttn-router-eu*.
- Select your location on the map and precise if the gateway is *indoor* or *outdoor*.
- When you finish, click on *Register Gateway* button.

Now you can see your private gateway:

GATEWAYS register gateway

eui-b827ebfffc22b70 BeeProject Lora gateway not connected EU_863_870

Fig 7: Private Gateway

6 CREATE APPLICATION FOR LoRa NODE (ARDUINO UNO)

Before connecting LoRa shield on Arduino UNO, you need to register it on TTN server. To do this, go on your console page of your account in TTN:

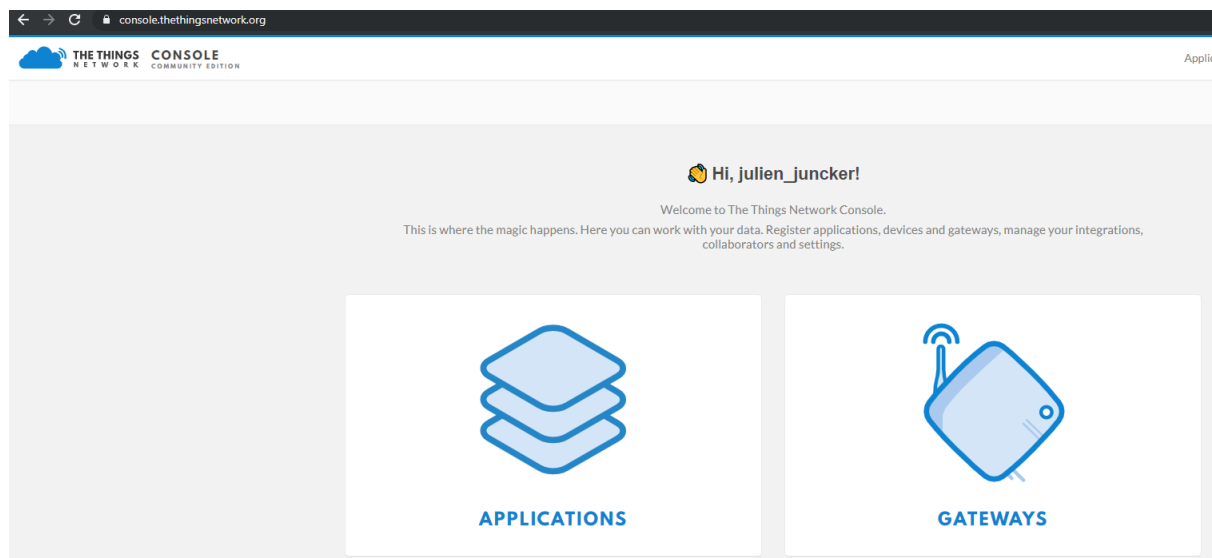


Fig 8: Console page on TTN server

Go to [applications](#) part and click to [add application](#) link:

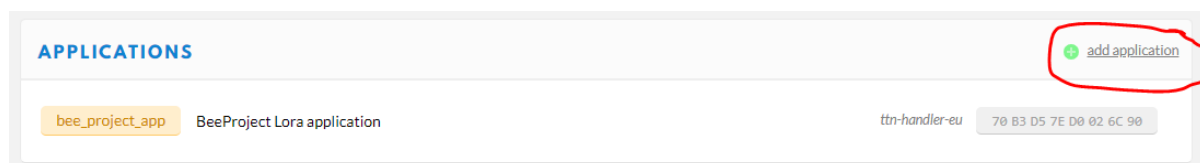


Fig 9: Add application button

ADD APPLICATION

Application ID
The unique identifier of your application on the network

Description
A human readable description of your new app

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.

Handler registration
Select the handler you want to register this application to

Cancel Add application

Fig 10: Add application pages

In this page you need to register all the information of your application:

- Application ID: You can put what you want.
- Description: Put description or not.
- Application EUI: Automatically generation.
- Handler registration: Stay by default [ttn-handler-eu](#).

Click to [Add application](#) button to register new application.

After you create your new application, go on [Devices](#) tab and click to [register device](#):

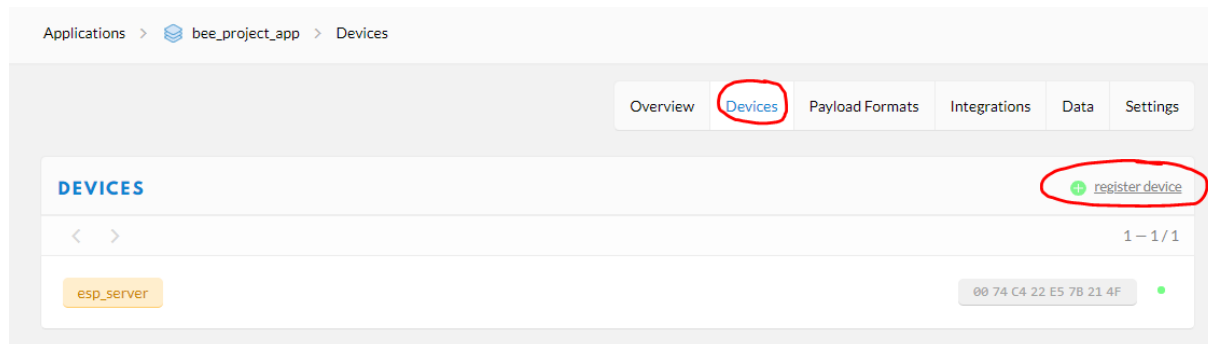


Fig 11: register device button

You will create your link to your Arduino device.

Fig 12: Register device form

In this form, you have:

- Device ID: Put name for your new Device.
- Device EUI: Click the button who is surrounded in red on the last picture to generate random number.
- App Key: Stay this label in blank, automatic generation.

- App EUI: Link your device to the application you created before.

Click to [register](#) button to register new device.

Go back to [Applications -> YOUR_APP](#) and go to [Payload Formats](#) tab and add this code decoder tab:

```
function Decoder(bytes, port) {
  // Decode plain text; for testing only
  return {
    myTestValue: String.fromCharCode.apply(null, bytes)
  };
}
```

Now, you can link your Arduino UNO to TTN server.

7 CONNECT THE LoRA SHIELD ON ARDUINO UNO

Connect the LoRa Shield and Arduino UNO. Put the 868MHZ antenna on it. Connect them to the computer via an USB cable.

Over here, we use the Arduino IDE1.68, we need to install this [Arduino-LMIC library](#). This repository contains the IBM LMIC (LoraMAC-in-C) library, slightly modified to run in the Arduino environment, allowing using the SX1272, SX1276 transceivers and compatible modules.

To install this library, go on [Sketch -> include a library -> manage libraries](#)

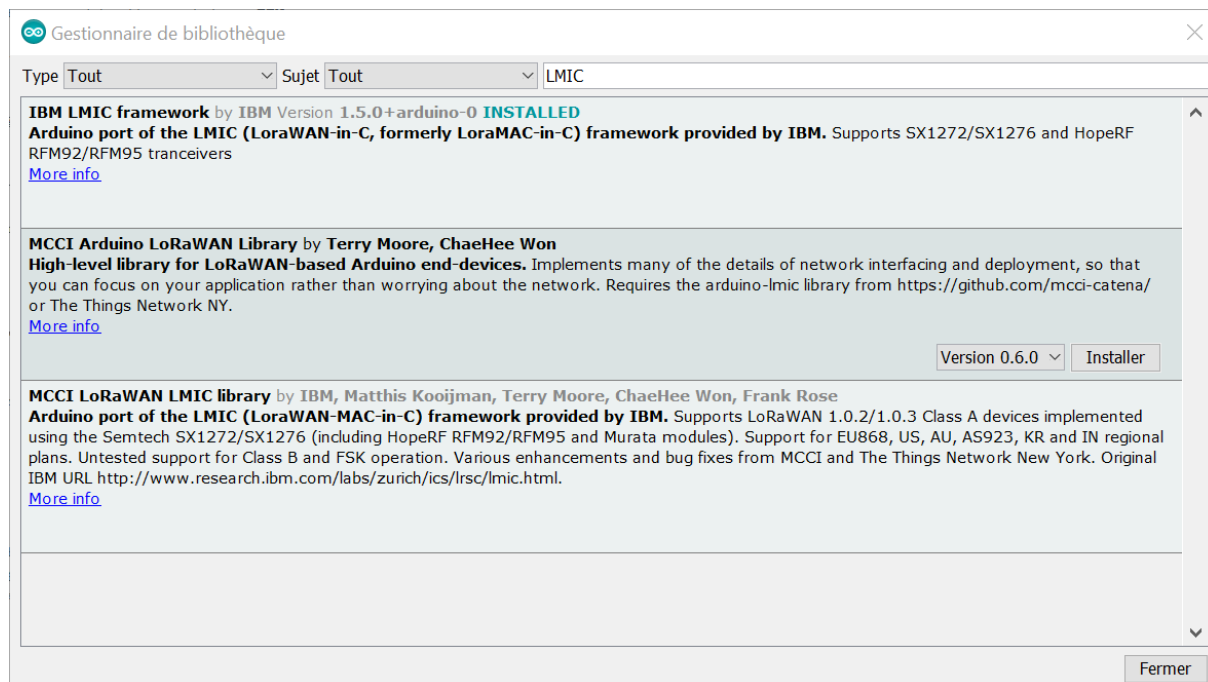


Fig 13: Arduino library manager

Search [LMIC](#) and install [IBM LMIC framework](#).

After do that, we download the ttn sketch from [this link](#).

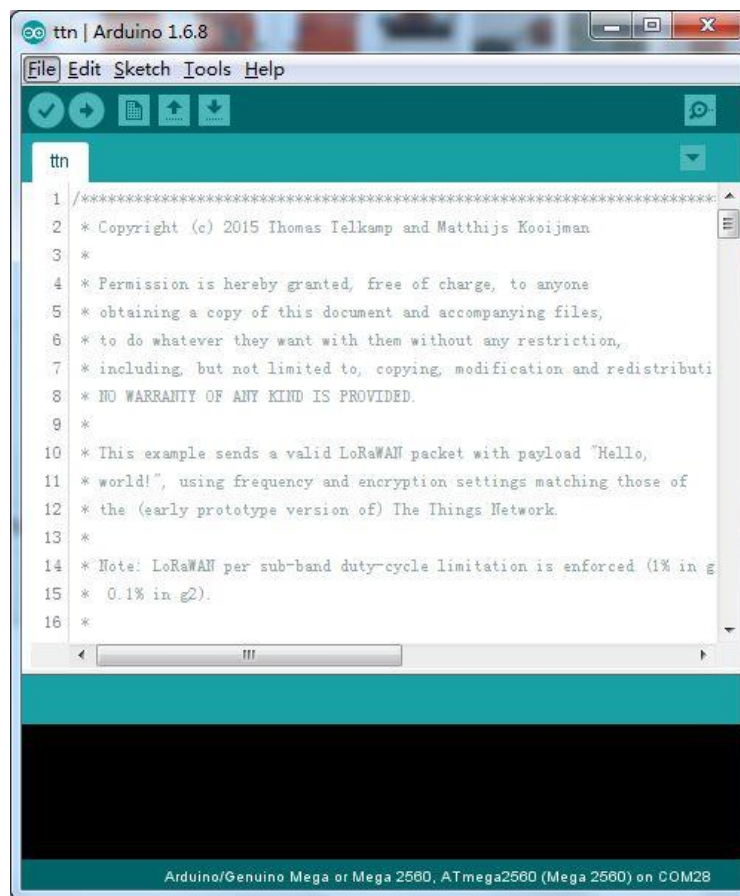


Fig 14: Arduino LoRa program

Open and modify this sketch, change:

- NWKSKEY as your own Network session key
- APPSKEY as your own App session key
- DEVADDR as your node_eui.
- static uint8_t mydata[] = "*hi*";

You can see this information on your application page on TTN website, go to

[Applications](#) -> *YOUR_APP_NAME* -> [Devices](#) -> *YOUR_DEVICE_NAME*

DEVICE OVERVIEW

Application ID bee_project_app

Device ID esp_server

Activation Method ABP

Device EUI <> ⇌ 00 74 C4 22 E5 7B 21 4F 📄

Application EUI <> ⇌ 70 B3 D5 7E D0 02 6C 90 📄

Device Address <> ⇌ 26 01 1E CB 📄

Network Session Key <> ⇌ msb { 0xFC, 0x4E, 0xB1, 0x72, 0x75, 0x8C, 0xB1, 0x60, 0xD5, 0x02, 0xD2, 0x2D, 0x72, 0x72, 0x72, 0x72, 0x72 } 📄

App Session Key <> ⇌ msb { 0x53, 0x70, 0xC2, 0x86, 0xC7, 0x35, 0x61, 0xDB, 0xD1, 0x8B, 0xBF, 0x8F, 0xB8, 0xB8, 0xB8, 0xB8 } 📄

Fig 15: Device overview

You need to put this information as same format you can see on picture. To change the format of this strings, click on button who are underlined in blue on picture.

Also, change the pins:

```
// Pin mapping
const lmic_pinmap lmic_pins = {
    .nss = 10, // Connected to pin D10
    .rxtx = LMIC_UNUSED_PIN, // For placeholder only, Do not connected on
    RFM92/RFM95
    .rst = 9, // Needed on RFM92/RFM95? (probably not)
    .dio = {5, 6, 7}, // Specify pin numbers for DIO0, 1, 2
    // connected to D2, D6, D7
};
```

In your Arduino UNO, put the LoRa shields at the top of your Arduino and wire like that:

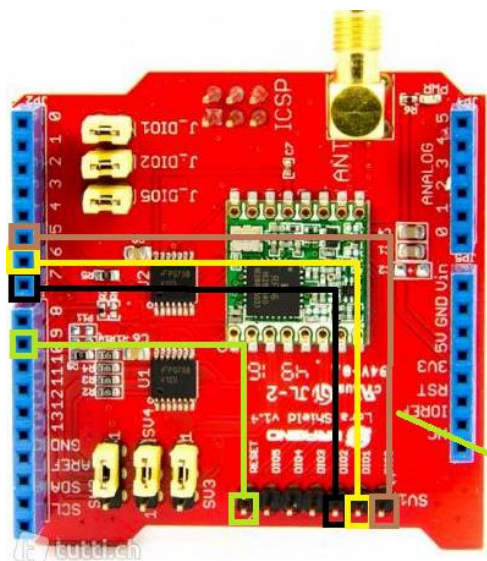


Fig 16: Arduino LoRa HAT wires

Pins:

- D10: reset
- D7: DI02
- D6: DI01
- D5: DI00

After that, put the jumpers as you can see on this picture:

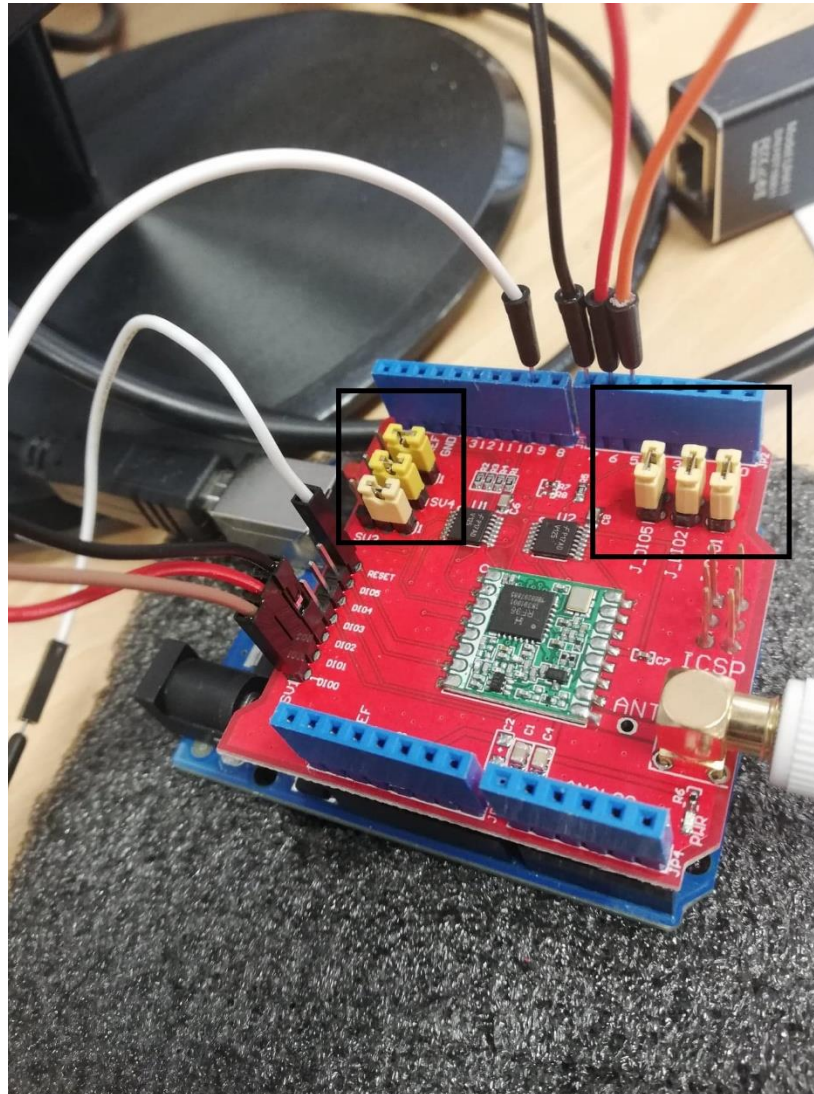


Fig 17: Arduino jumpers

Choose the right port and right board to upload the sketch. Upload the program and if this works, you can see on serial monitor:

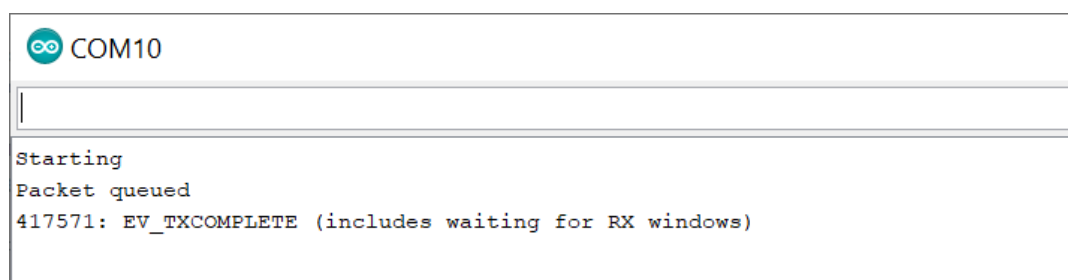


Fig 18: Arduino serial monitor

```

pi@pi:~/Desktop/Lora/single_chan_pkt_fwd $ ./single_chan_pkt_fwd
SX1276 detected, starting.
Gateway ID: b8:27:eb:ff:ff:c2:2b:70
Listening at SF7 on 868.100000 Mhz.
stat update: {"stat":{"time":"2019-12-06 10:46:06 GMT","lati":0.00000,"long":0.00000,"alti":0,"rxnb":0,"rxok":0,"rxfw":0,"ackr":0.0,"dwnb":0,"txnb":0,"pfrm":0,"single_chan_pkt_fwd":{"mail":"","desc":""}}}
stat update: {"stat":{"time":"2019-12-06 10:46:36 GMT","lati":0.00000,"long":0.00000,"alti":0,"rxnb":0,"rxok":0,"rxfw":0,"ackr":0.0,"dwnb":0,"txnb":0,"pfrm":0,"single_chan_pkt_fwd":{"mail":"","desc":""}}}
stat update: {"stat":{"time":"2019-12-06 10:47:06 GMT","lati":0.00000,"long":0.00000,"alti":0,"rxnb":0,"rxok":0,"rxfw":0,"ackr":0.0,"dwnb":0,"txnb":0,"pfrm":0,"single_chan_pkt_fwd":{"mail":"","desc":""}}}
stat update: {"stat":{"time":"2019-12-06 10:47:36 GMT","lati":0.00000,"long":0.00000,"alti":0,"rxnb":0,"rxok":0,"rxfw":0,"ackr":0.0,"dwnb":0,"txnb":0,"pfrm":0,"single_chan_pkt_fwd":{"mail":"","desc":""}}}
stat update: {"stat":{"time":"2019-12-06 10:48:06 GMT","lati":0.00000,"long":0.00000,"alti":0,"rxnb":0,"rxok":0,"rxfw":0,"ackr":0.0,"dwnb":0,"txnb":0,"pfrm":0,"single_chan_pkt_fwd":{"mail":"","desc":""}}}
stat update: {"stat":{"time":"2019-12-06 10:48:36 GMT","lati":0.00000,"long":0.00000,"alti":0,"rxnb":0,"rxok":0,"rxfw":0,"ackr":0.0,"dwnb":0,"txnb":0,"pfrm":0,"single_chan_pkt_fwd":{"mail":"","desc":""}}}
stat update: {"stat":{"time":"2019-12-06 10:49:06 GMT","lati":0.00000,"long":0.00000,"alti":0,"rxnb":0,"rxok":0,"rxfw":0,"ackr":0.0,"dwnb":0,"txnb":0,"pfrm":0,"single_chan_pkt_fwd":{"mail":"","desc":""}}}
stat update: {"stat":{"time":"2019-12-06 10:49:36 GMT","lati":0.00000,"long":0.00000,"alti":0,"rxnb":0,"rxok":0,"rxfw":0,"ackr":0.0,"dwnb":0,"txnb":0,"pfrm":0,"single_chan_pkt_fwd":{"mail":"","desc":""}}}
stat update: {"stat":{"time":"2019-12-06 10:50:06 GMT","lati":0.00000,"long":0.00000,"alti":0,"rxnb":0,"rxok":0,"rxfw":0,"ackr":0.0,"dwnb":0,"txnb":0,"pfrm":0,"single_chan_pkt_fwd":{"mail":"","desc":""}}}
packet RSSI: -52, RSSI: -106, SNR: 10, Length: 26
rxpk update: {"rxpk":{"tmst":3479060350,"chan":0,"rfch":0,"freq":868.100000,"stat":1,"modu":"LORA","data":{"SF7BW125","codr":"4/5","lnt":10,"rsi":52,"size":26,"data":{"QMeASaAAWABNX13IRv5013jt8Ztvo8Mg5A="}}}}}

```

Fig 19: single_chan_pkt_fwd program with packet result

8 TEST THE SOLUTION

Now you can see on *Gateways -> YOUR_GATEWAY -> Traffic*:

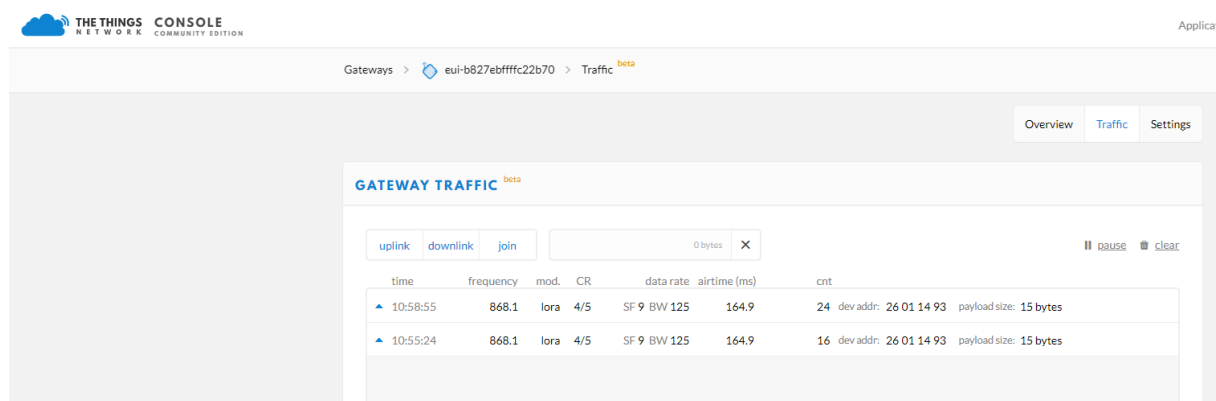


Fig 20: Gateway traffic

And on *Applications -> YOUR_APP -> Devices -> YOUR_DEVICE -> Data*:

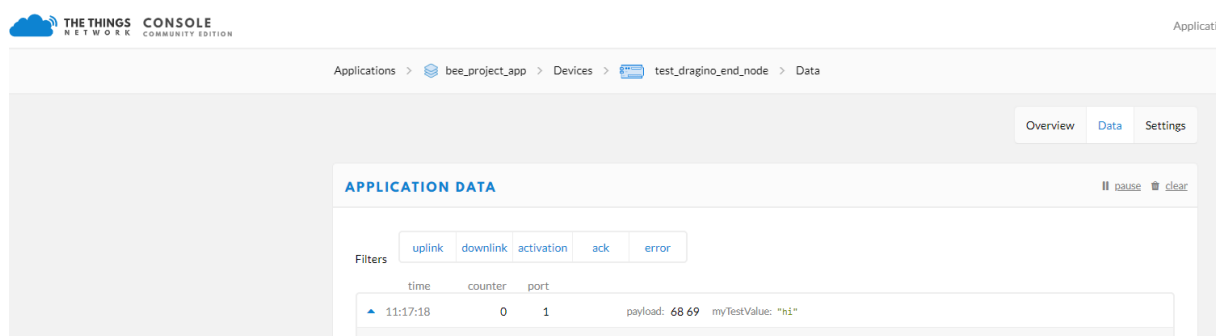


Fig 21: Application data

And on your computer or a server, you need to test the connection with TTN and uplink message. For do that, install ttn python library:

```
Pip install ttn
```

And launch the python script *TTN_subscribe_app.py*. Be careful, you must be in *ROOT* user.

You can see when you receive uplink:



```
You are connected! Press touch "esc" if you want to quit the program
('Received uplink from ', u'test_dragino_end_node')
MSG(myTestValue=u'hi')
```

Fig 22: Receive uplink in bee project MQTT program