R.D.R PROJECT

3ºAno – 1º Semestre

Aulas Teórico-Práticas

laboritory

Docente:

Paulo Torres

Edição revista:

Paulo Torres

**Summary**

[1 Introduction Erreur ! Signet non défini.](#_Toc21962442)

[2 Database 3](#_Toc21962443)

[2.1 conception Erreur ! Signet non défini.](#_Toc21962444)

[2.2 Creation of database Erreur ! Signet non défini.](#_Toc21962445)

[3 Python script Erreur ! Signet non défini.](#_Toc21962446)

[4 Webpages Erreur ! Signet non défini.](#_Toc21962447)



# 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.

# 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.

# 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.

# 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:

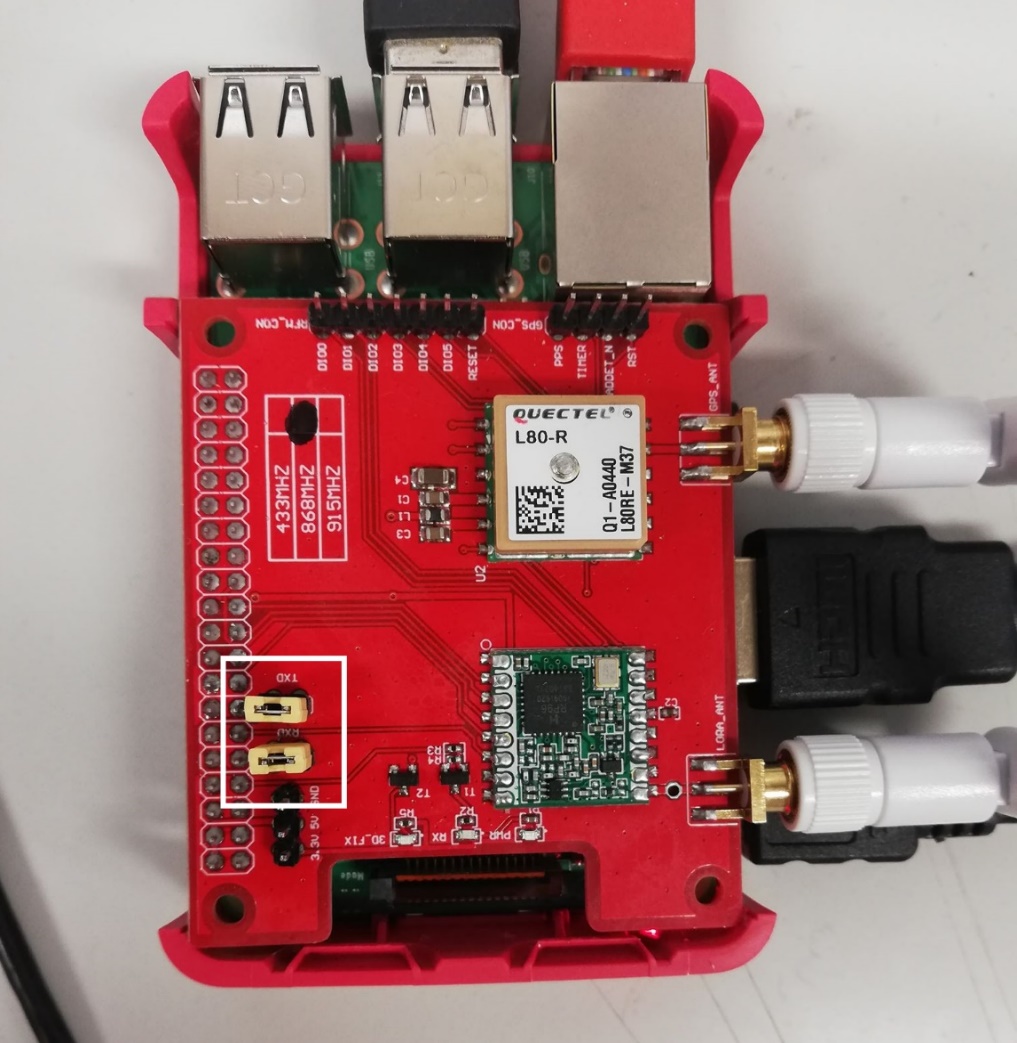
**

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!").



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.

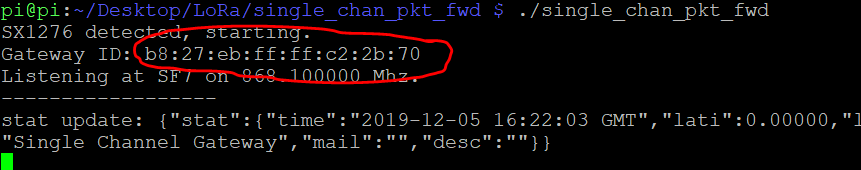


Fig 3: single\_chan\_pkt\_fwd program

Keep this ID for the next step.

# 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:

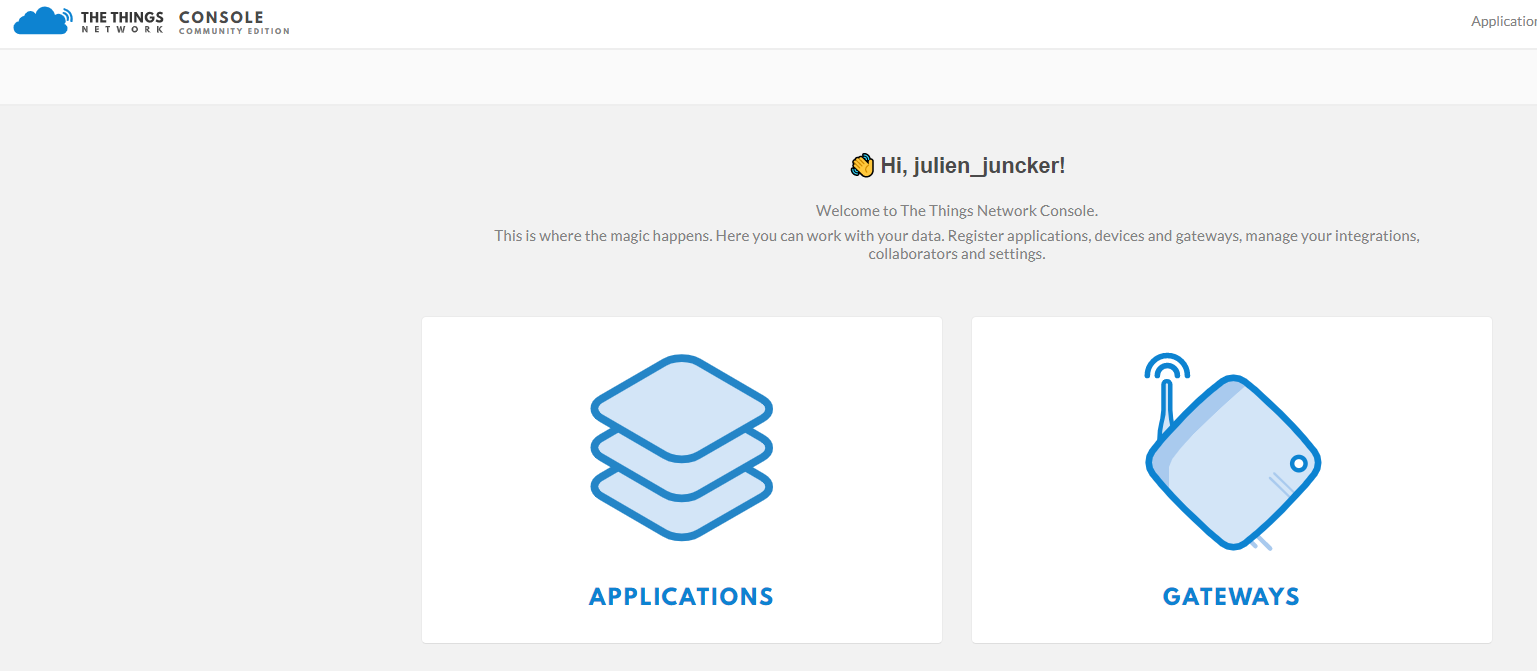


Fig 4: Console view on TTN network

Click on gateway and click on register gateway link:

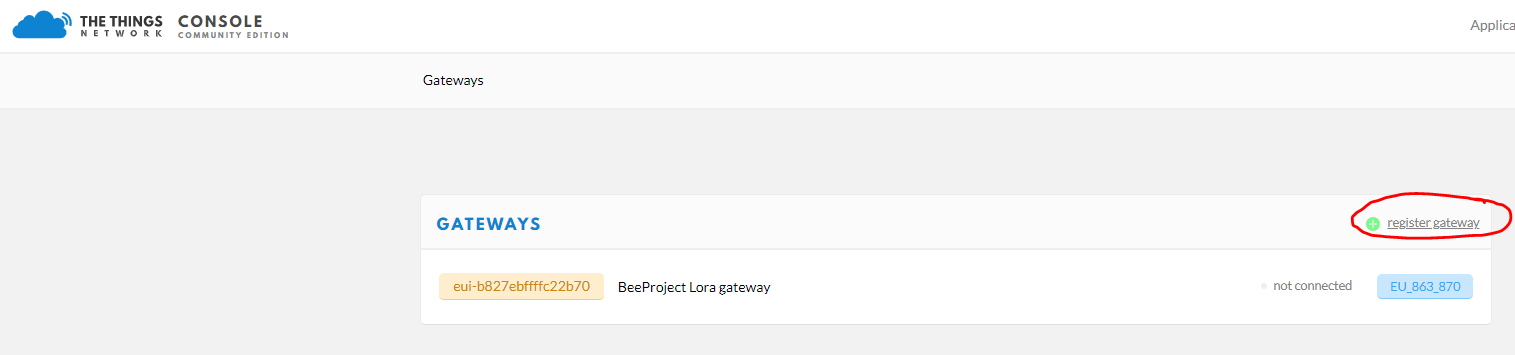


Fig 5: register Gateway button

You will see gateway registering page:

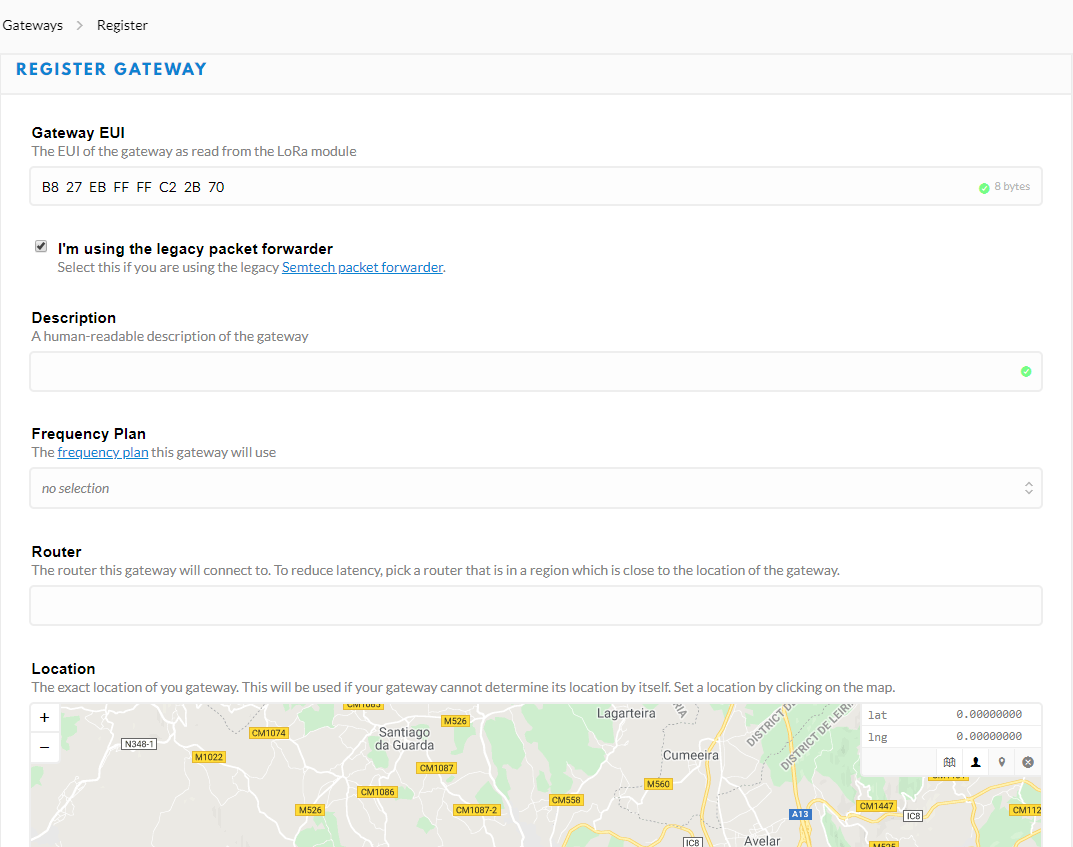


Fig 6: Register Gateway

* 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:

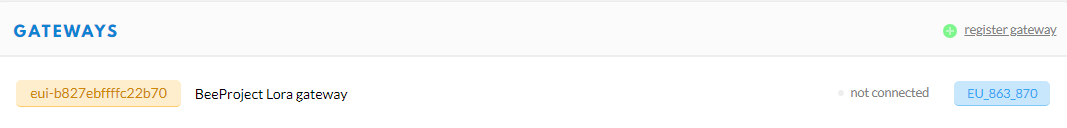


Fig 7: Private Gateway

# 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:

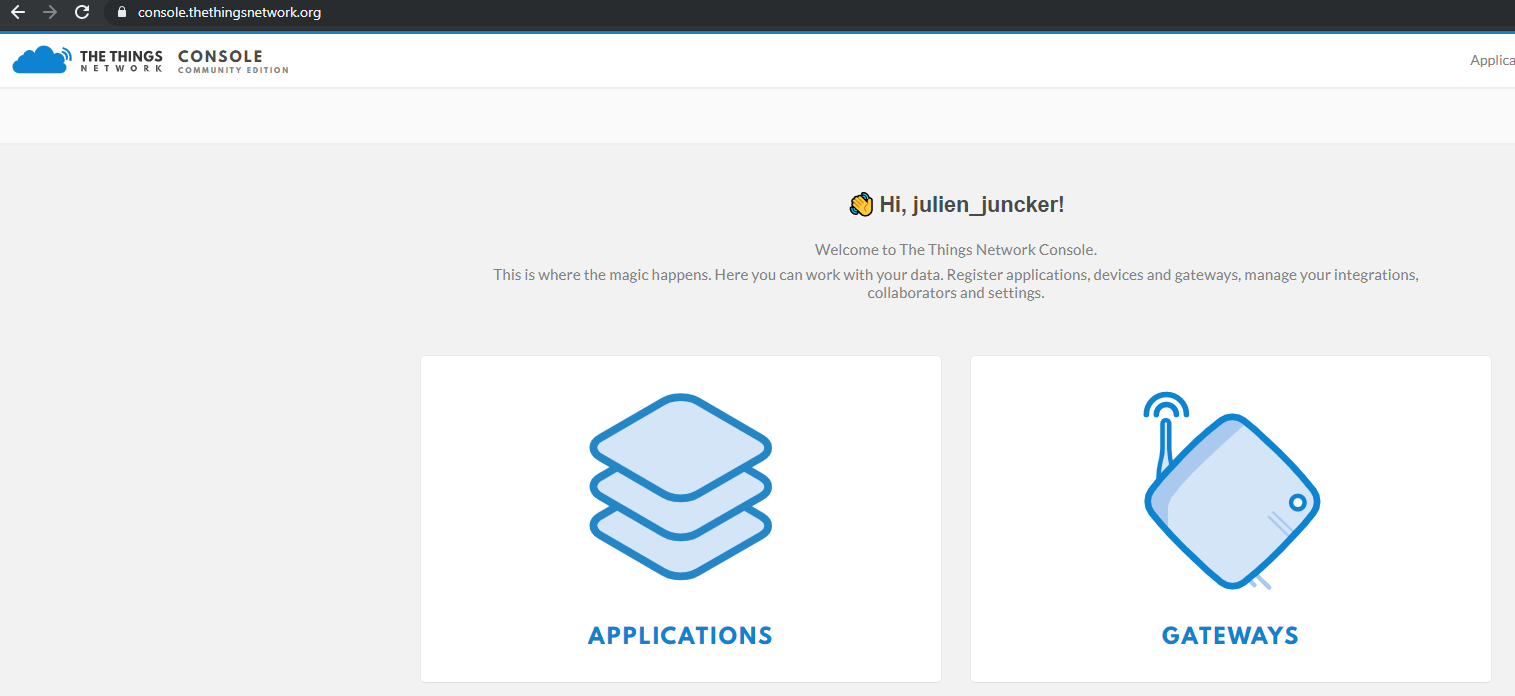


Fig 8: Console page on TTN server

Go to applications part and click to add application link:

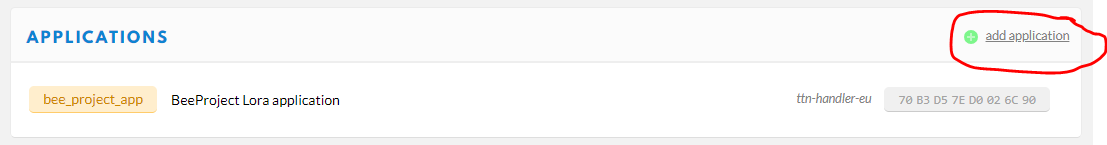


Fig 9: Add application button

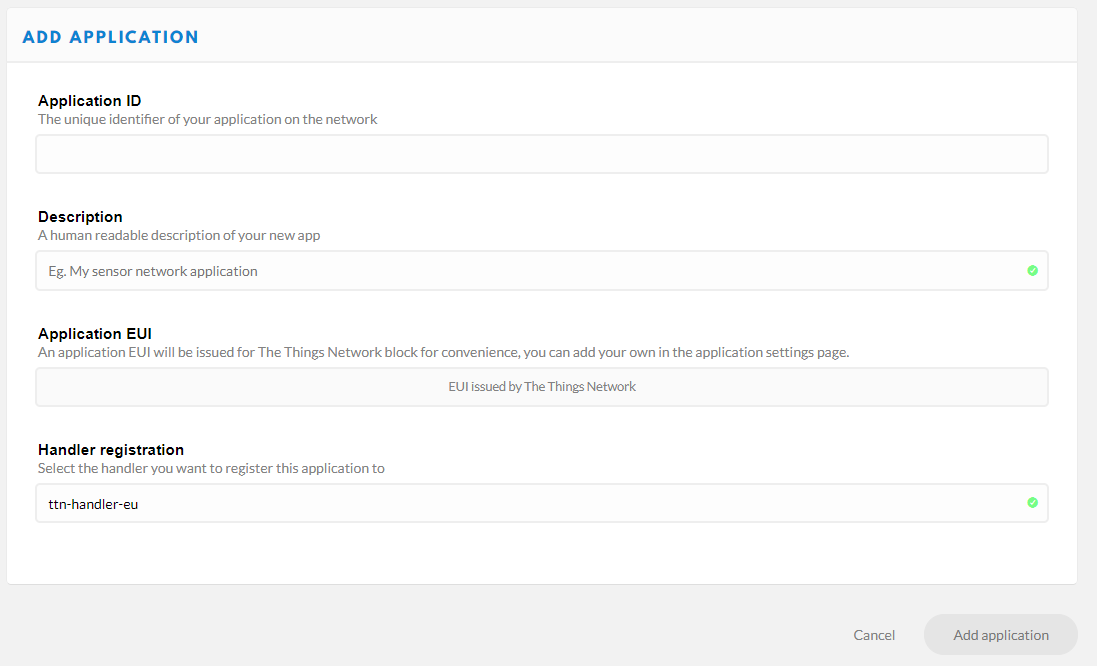


Fig 10: Add application pages

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:

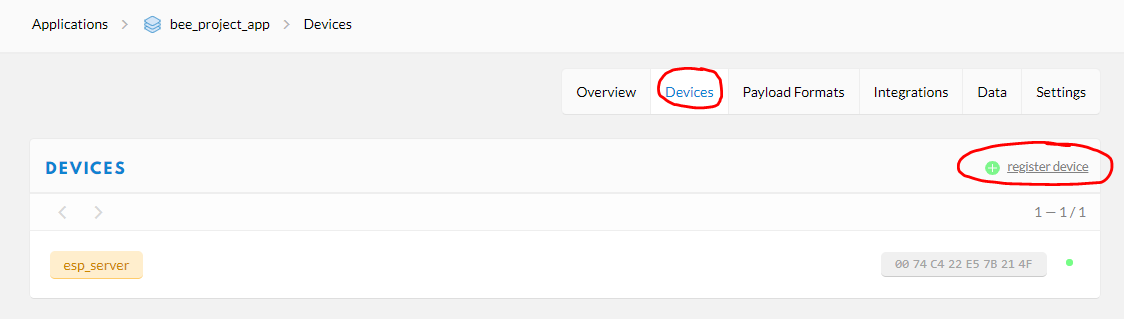


Fig 11: register device button

You will create your link to your Arduino device.

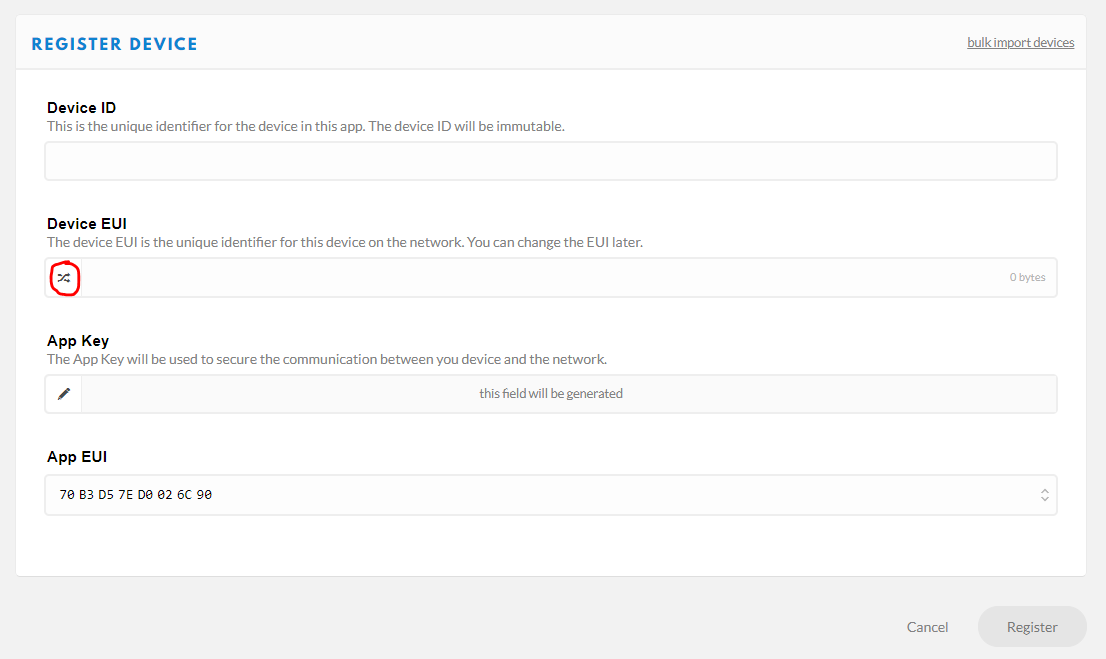


Fig 12: Register device form

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.

# 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](https://github.com/matthijskooijman/arduino-lmic). 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

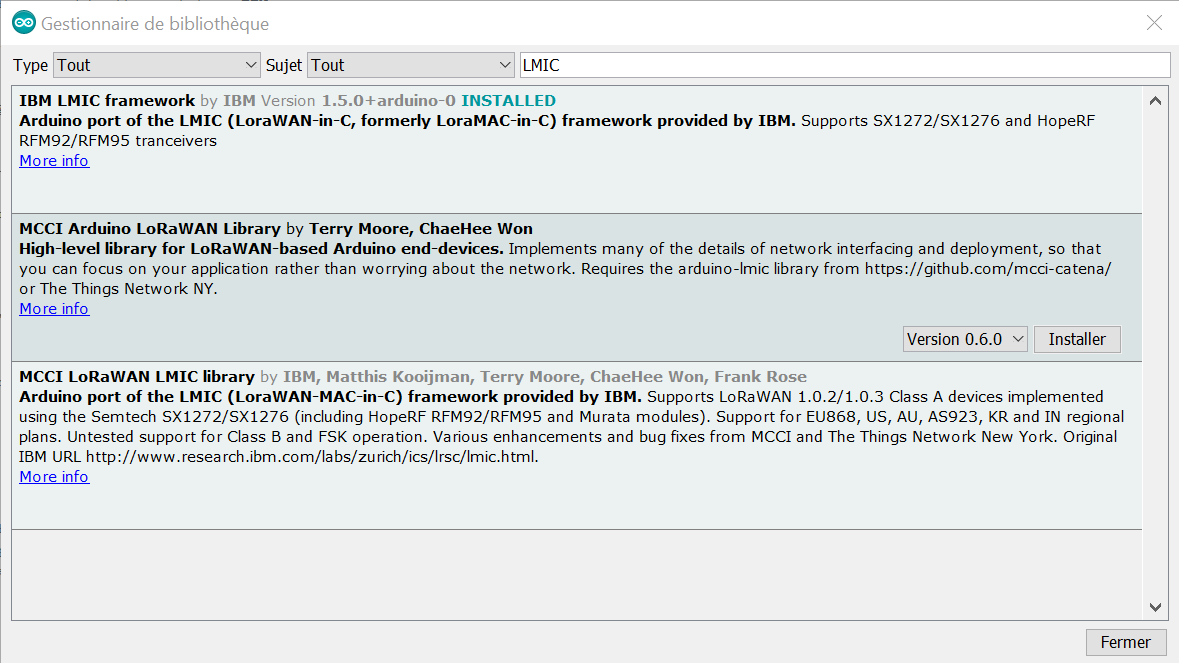


Fig 13: Arduino library manager

Search LMIC and install IBM LMIC framework.

After do that, we download the ttn sketch from [this link](https://github.com/dragino/Lora/tree/master/Lora%20Shield/Examples/lora_shield_ttn).

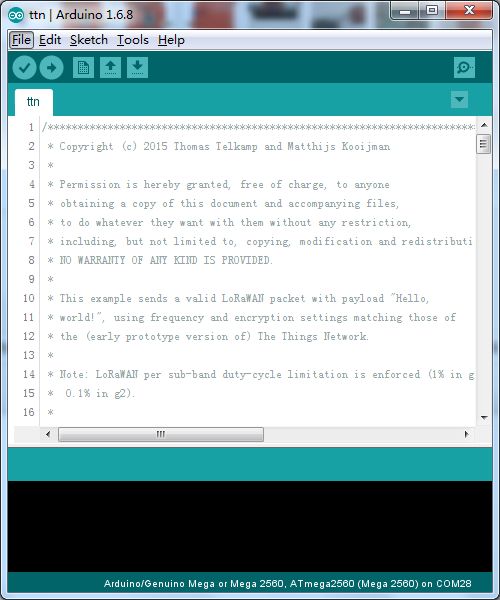


Fig 14: Arduino LoRa program

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



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:

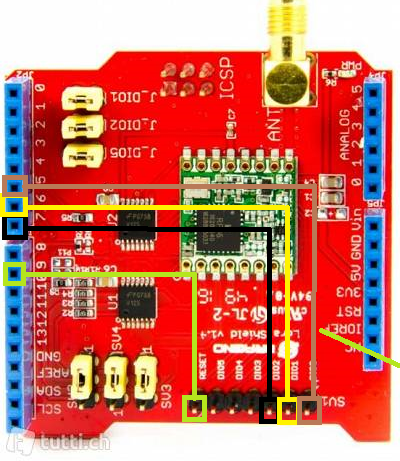


Fig 16: Arduino LoRa HAT wires

Pins:

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

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

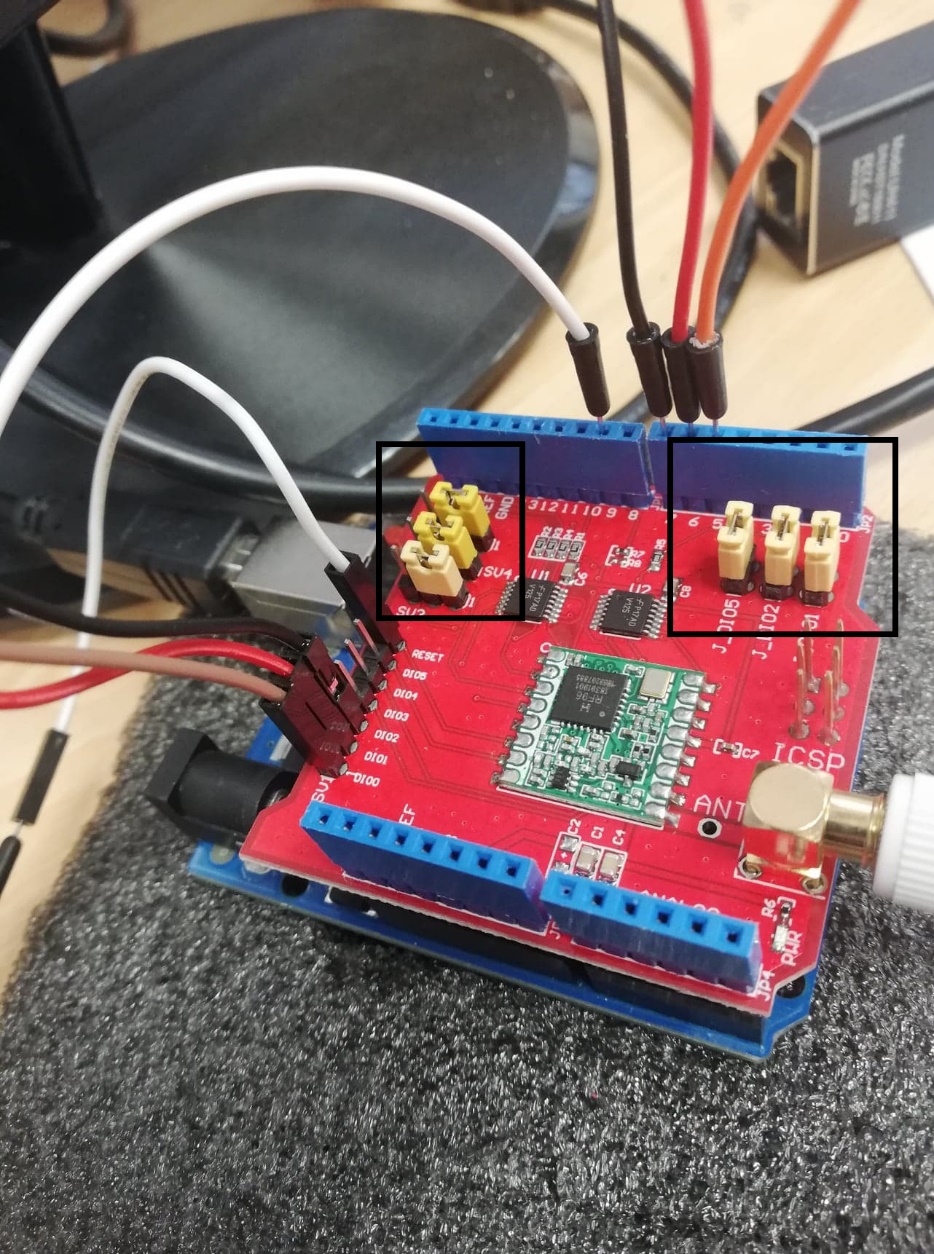


Fig 17: Arduino jumpers

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

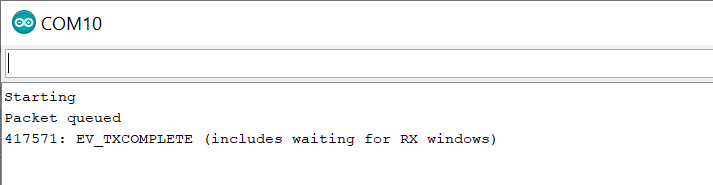


Fig 18: Arduino serial monitor

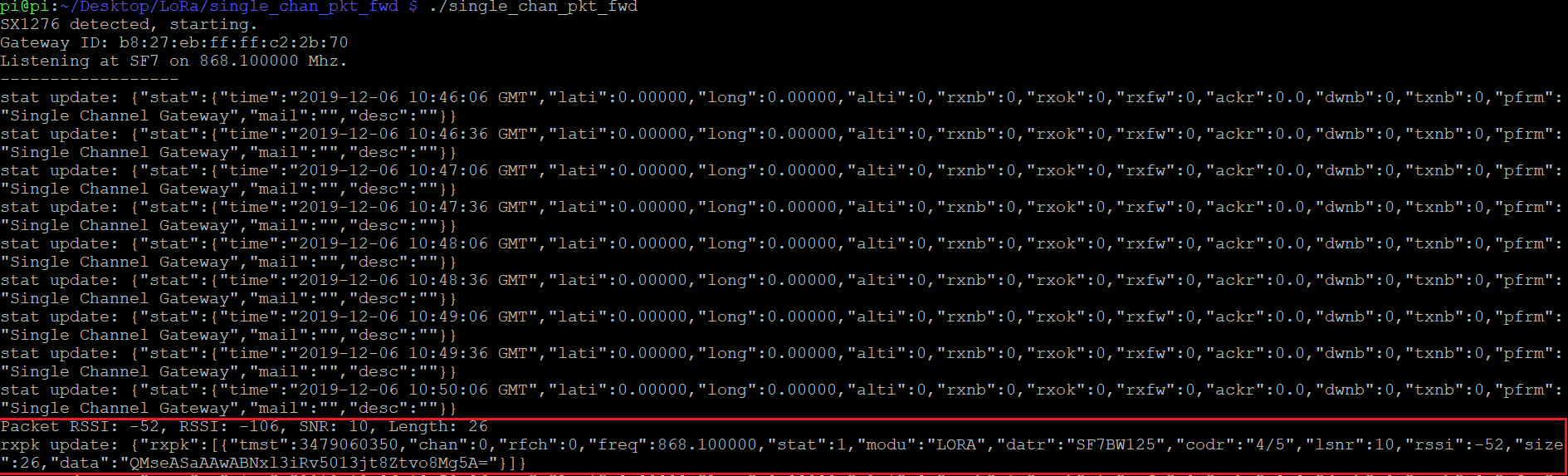


Fig 19: single\_chan\_pkt\_fwd program with packet result

# Test the solution

Now you can see on Gateways -> YOUR\_GATEWAY -> Traffic:

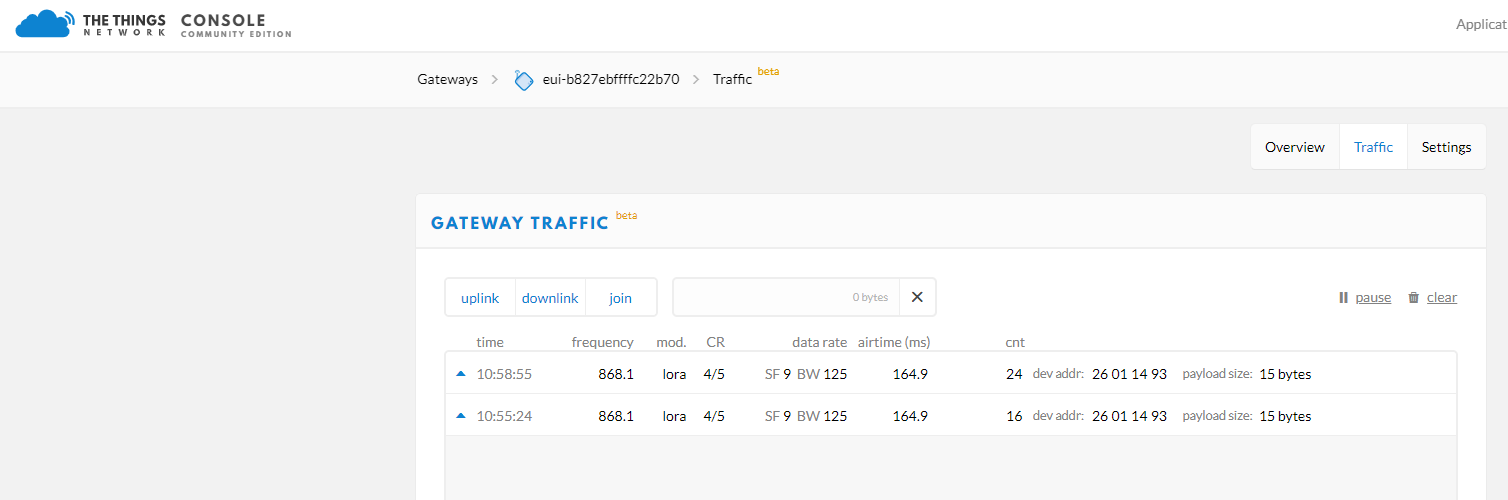


Fig 20: Gateway traffic

And on Applications -> YOUR\_APP -> Devices -> YOUR\_DEVICE -> Data:

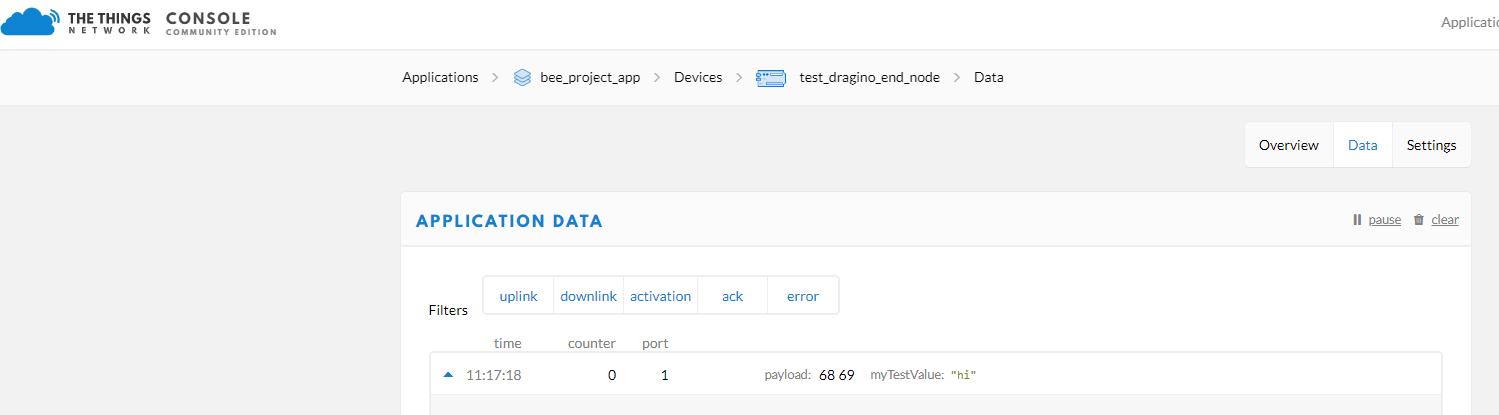


Fig 21: Application data

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:



Fig 22: Receive uplink in bee project MQTT program