Lab3:
LoRaWAN and TTN

Marco Zennaro, PhD
ICTP

Labs

⅓ Ready to use, tested examples

⅓ Exercise based on the examples

⅓ Your imagination → create new applications

Lab alert

The number of variables in the lab settings is huge (computer operating system, firewall, device firmware version, code version, network, etc)

Things will go wrong :-)

Be patient, we will solve all issues!

Found a bug? Let me know! Feedback is welcome.

Hands-on sessions

"Be excellent to each other", asking / helping is OK.

Google error messages to fix issues.

Coping blindly does not lead to new insight.

Reading other people's code helps a lot.

Check Pycom’s documentation.

Our Lab equipment

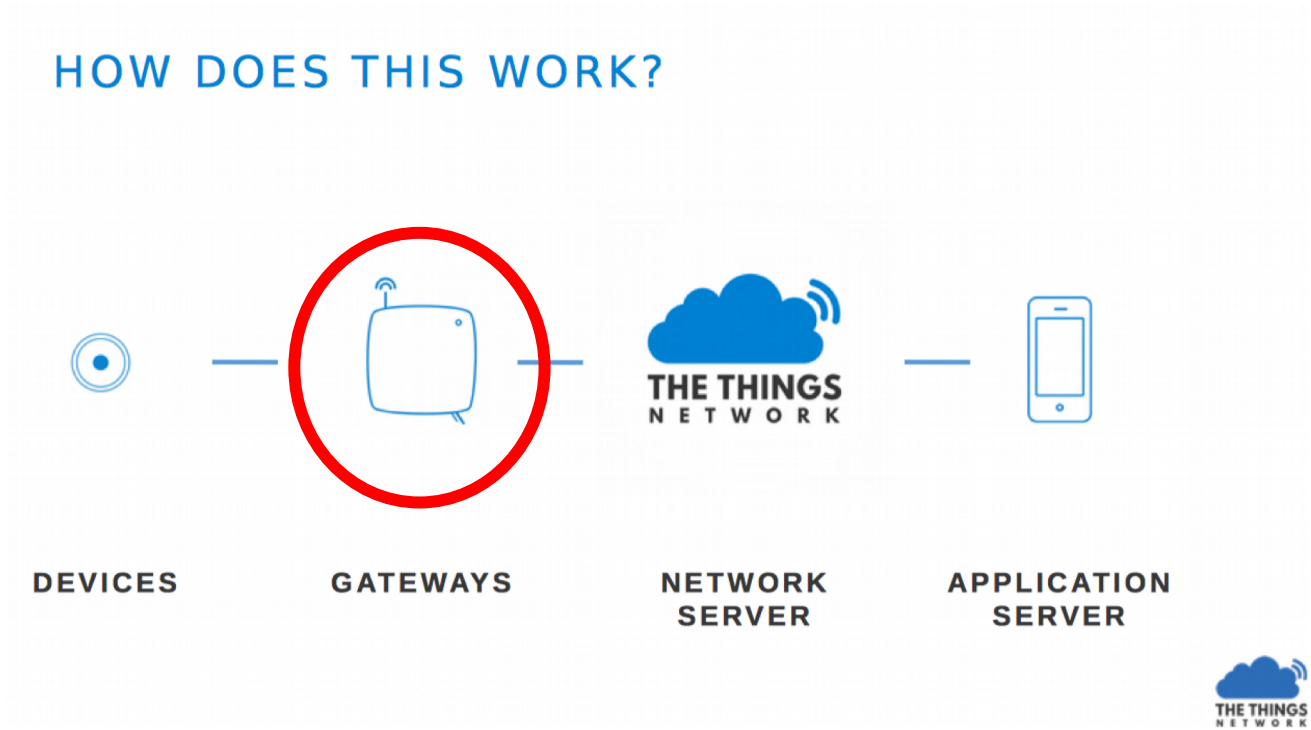
Pycom LoPy 4

PySense

microUSB Cable

RPi with LoRa module

TTN: devices, gateways, servers



Single Channel Gateway

70 euro

RPi+board

No enclosure

Single Channel

Uputronics LoRa expansion board

Raspberry Pi+ LoRa(TM) Expansion Board

Brand: Uputronics
Product Code: HAB-LORA-8N
Availability: 130

£26.39

Ex Tax: £21.99

Available Options

* LoRa(TM) Module Selection CE1

- ○ 434Mhz RFM98W
- ◉ 868Mhz RFM95W
- ○ 915Mhz RFM95W

https://store.uputronics.com

ICTP

Installing a TTN gateway

Install Raspbian Stretch, boot the RPi and login as pi.

Enter:

```
cd /home/pi
```

Installing a TTN gateway: interfaces

You first need to enable the SPI port and SSH access:

```
sudo raspi-config
```

5 →Interfacing Options → Configure connections to peripherals → SPI
5 →Interfacing Options → Configure connections to peripherals → SSH

Installing a TTN gateway: software

Clone the github repository with the packet forwarder code:

Install git: sudo apt-get install git

git clone https://github.com/hallard/single_chan_pkt_fwd

This will create a directory with the necessary code in your home directory.

Installing a TTN gateway: config file

You now have to modify the global_conf.json configuration file to fit the specific uputronics board.

```
cd single_chan_pkt_fwd

nano global_conf.json
```

Installing a TTN gateway: config file

The config file should look like this:

```
{
  "SX127x_conf":
  {
    "freq": 868300000,
    "spread_factor": 7,
    "pin_nss": 11,
    "pin_dio0": 27,
    "pin_rst": 0,
    "pin_led1": 29
  },
```

Installing a TTN gateway: config file

You can add info about the gateway (location, owner, description):

```
"gateway_conf":
 {
   "ref_latitude": 0.0,
   "ref_longitude": 0.0,
   "ref_altitude": 10,

   "name": "SC Gateway",
   "email": "contact@whatever.com",
   "desc": "Dragino Single Channel Gateway on RPI",
```

Installing a TTN gateway: installing

We can now install some additional software:

```
cd /tmp
wget https://lion.drogon.net/wiringpi-2.50-1.deb
sudo dpkg -i wiringpi-2.50-1.deb
```

And we go back to the original directory:

```
cd /home/pi/single_chan_pkt_fwd
```

Installing a TTN gateway: installing

We can now install the software:

```
make
sudo make install
```

The service is active and will start at boot. To check its status you can use:

```
systemctl status single_chan_pkt_fwd
systemctl start single_chan_pkt_fwd
systemctl stop single_chan_pkt_fwd
```

Registering a TTN gateway

To register the gateway in TTN we must obtain the Gateway EUI.
Run:

```
./single_chan_pkt_fwd
```

As an output you will receive a string that contains the Gateway EUI.

Registering a TTN gateway

Gateway Configuration
  SC Gateway (contact@whatever.com)
  Dragino Single Channel Gateway on RPI
  Latitude=0.00000000
  Longitude=0.00000000
  Altitude=10
Trying to detect module with NSS=11 DIO0=27 Reset=0 Led1=29
SX1276 detected, starting.
Gateway ID: b8:27:eb:ff:ff:df:2d:aa
Listening at SF7 on 868.300000 Mhz.

Registering a TTN gateway

Visit https://console.thethingsnetwork.org/

Select Gateways → Register Gateway

Select ““I'm using the legacy packet forwarder”

In Gateway ID enter the value you just obtained as Gateway ID (b8:27:eb:ff:ff:df:2d:aa)

Registering a TTN gateway

Fill in the Description

Select Frequency Plan to 868 for Africa

Click on **Register Gateway**

You just registered a new gateway!

Gateway: exercise

We have three gateways in total.

Register the other two gateways and add your colleagues as collaborators.

Find possible locations for the gateways and find their GPS positions using online services.

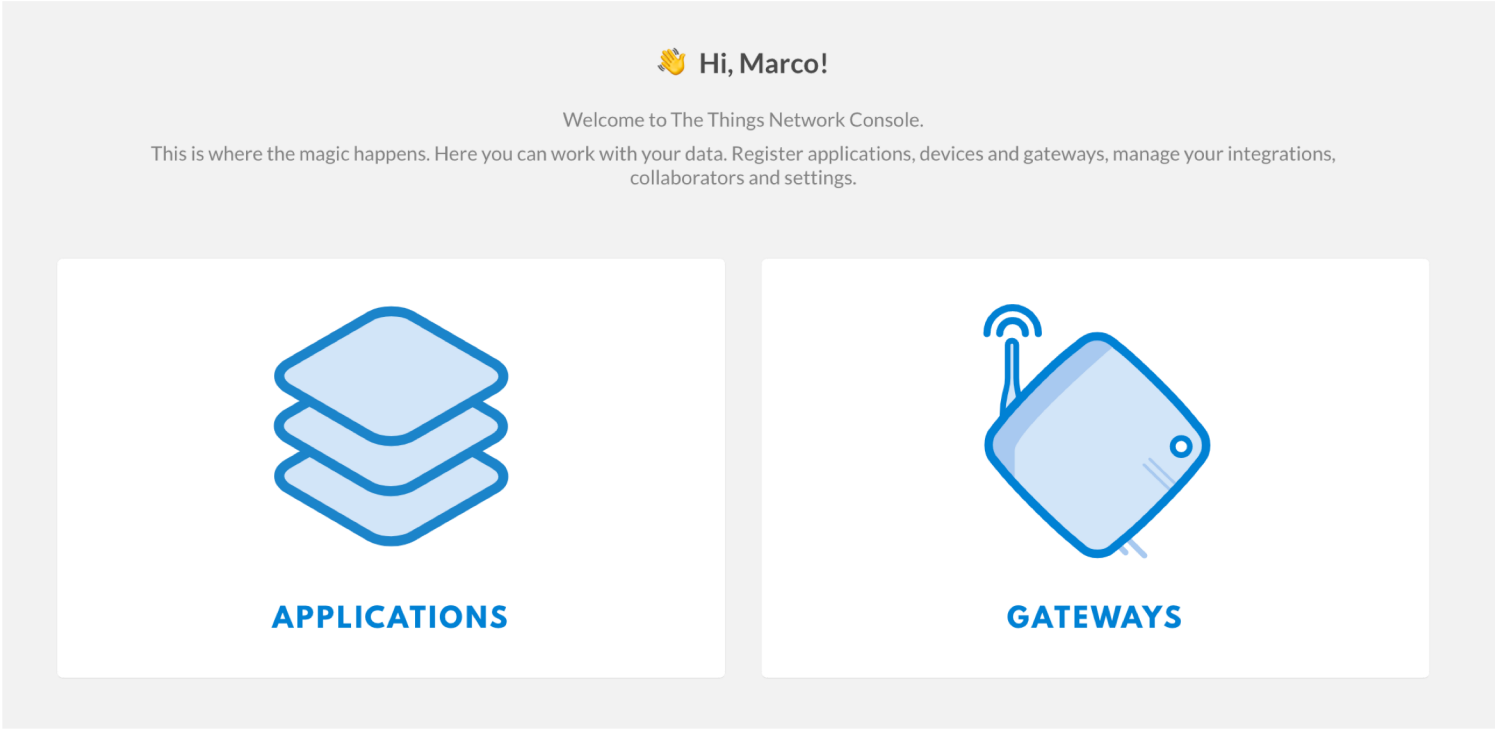
Sending T,H to TTN

TTN: App

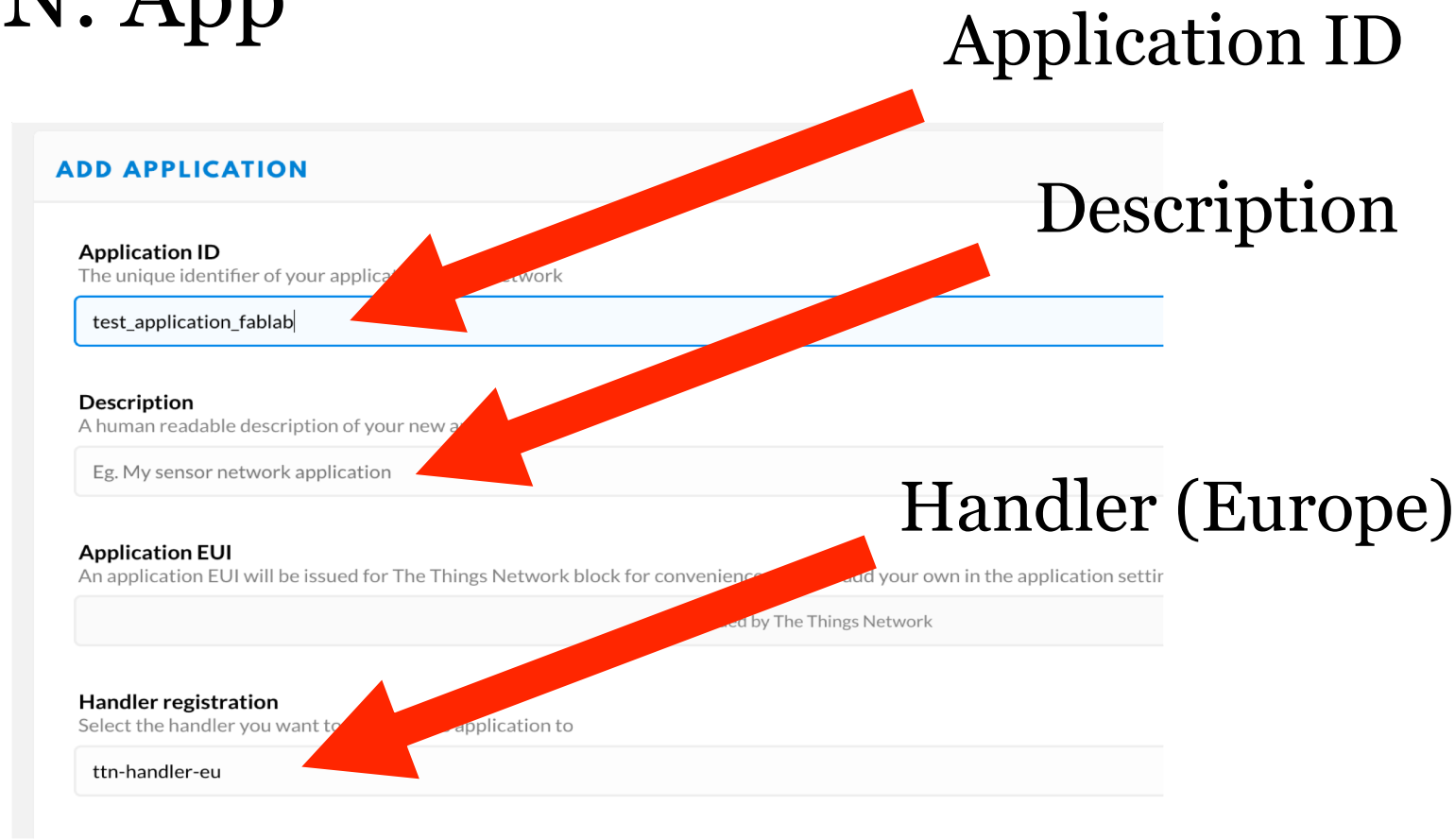
As a first step we must create a TTN application and register our device to it. This is necessary so that data are correctly encrypted.

Create a new application in TTN.

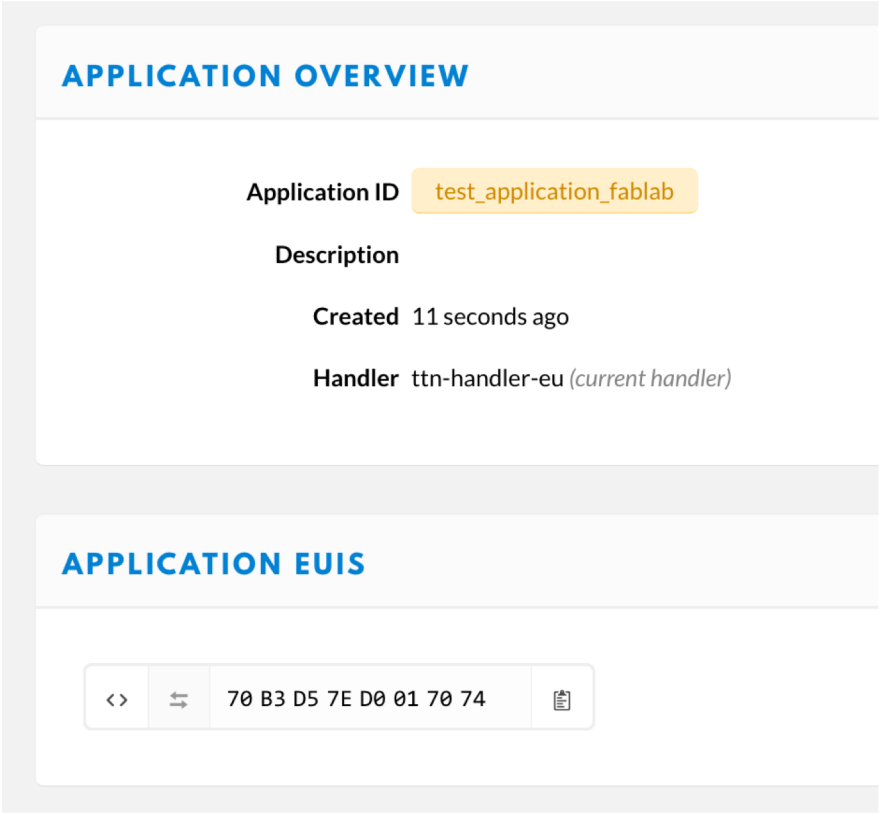
TTN: App


Hi, Marco!
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.
APPLICATIONS
GATEWAYS

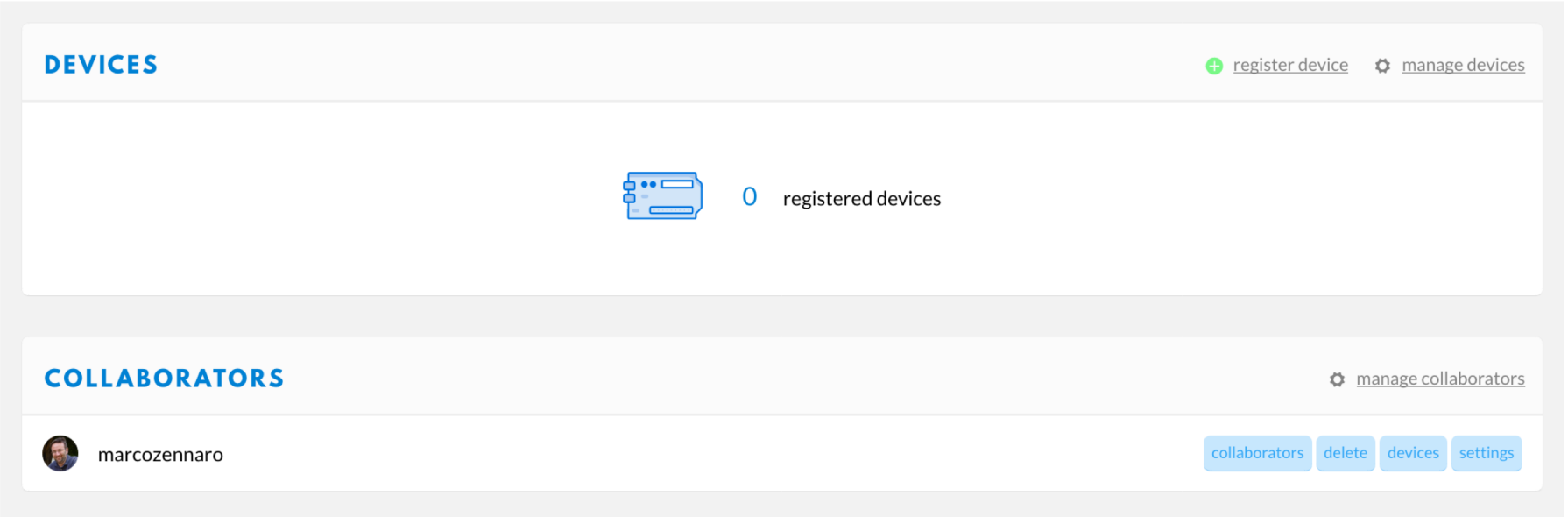
TTN: App


Application ID
Description
Handler (Europe)
ADD APPLICATION
Application ID
The unique identifier of your applica
twork
test_application_fablab
Description
A human readable description of your new a
Eg. My sensor network application
Application EUI
An application EUI will be issued for The Things Network block for convenienc
dd your own in the application settir
ed by The Things Network
Handler registration
Select the handler you want to
application to
ttn-handler-eu

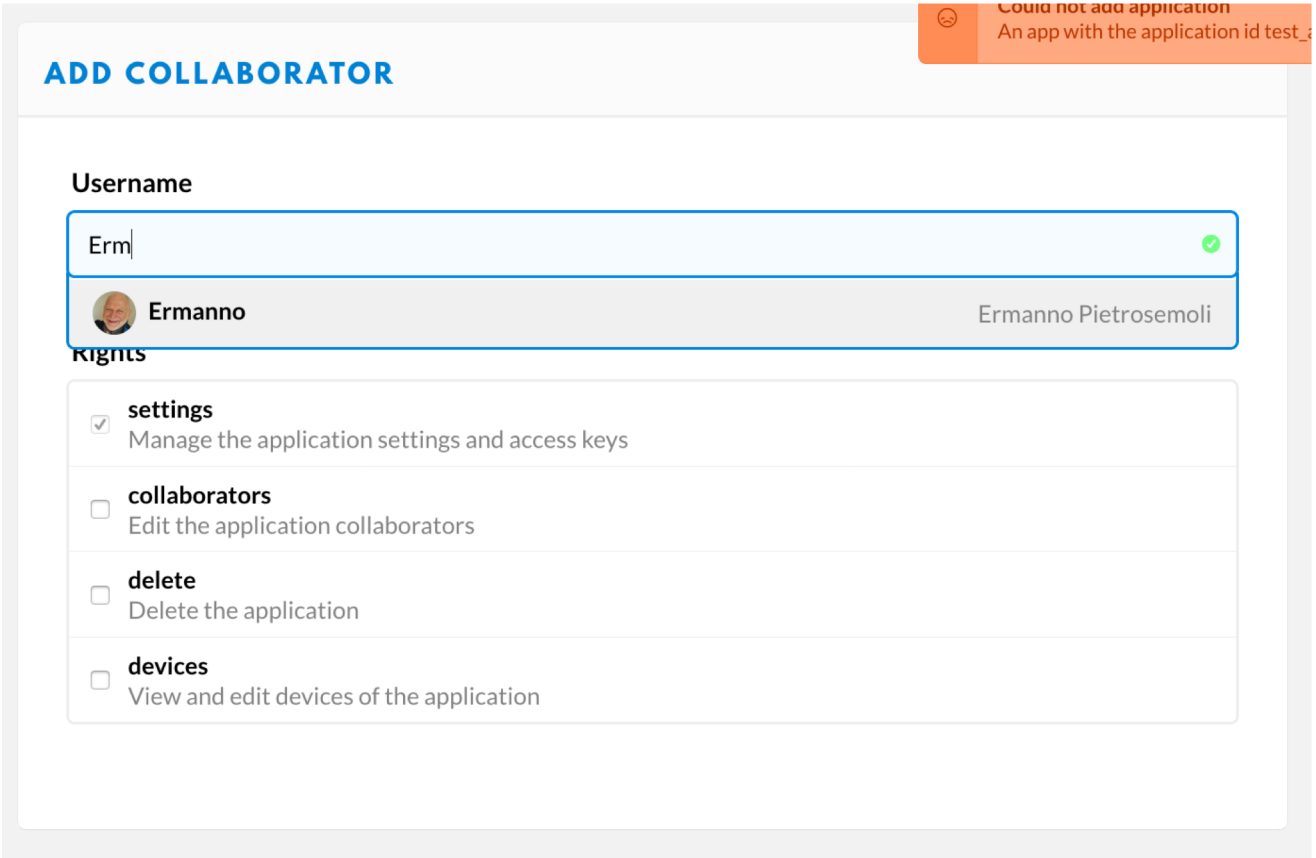
TTN: we have a new App!



TTN: Collaborators



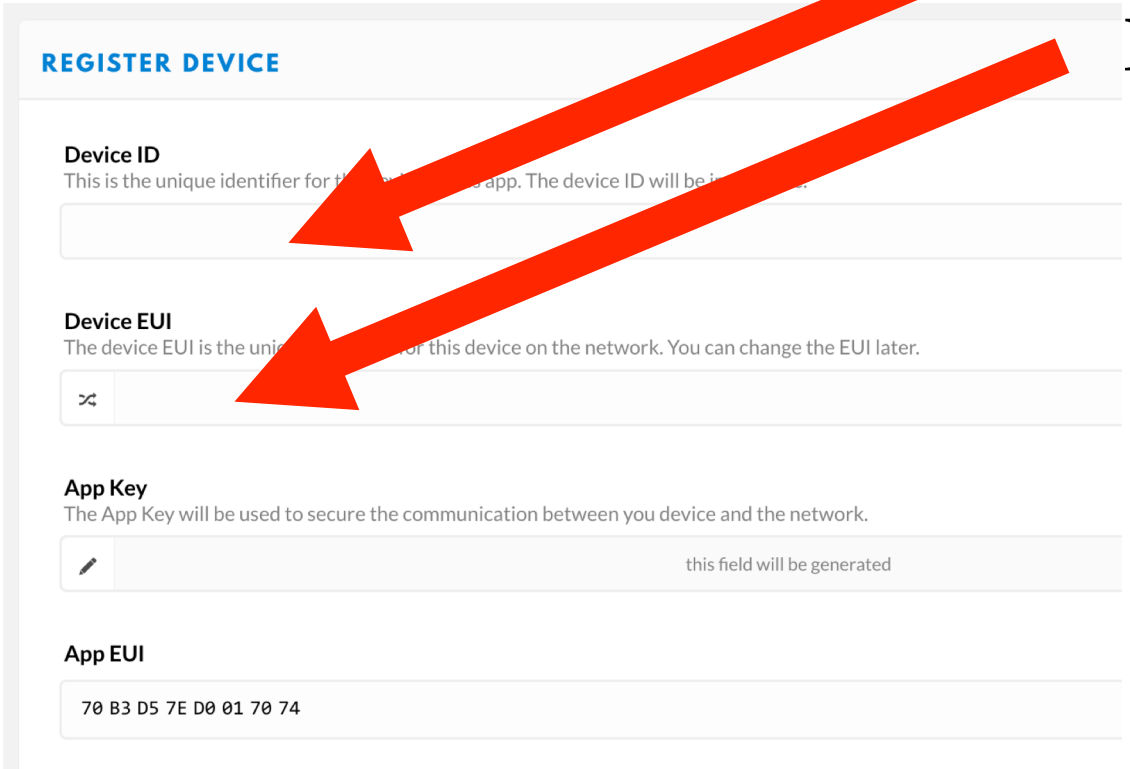
TTN: add a Collaborator to the App



TTN: register a device

Name of Device

Device EUI



Where is the device EUI?

Step 1: Create a device in TTN with the OTAA keys from LGT-92.

Each LGT-92 is shipped with a sticker with the default device EUI as below:

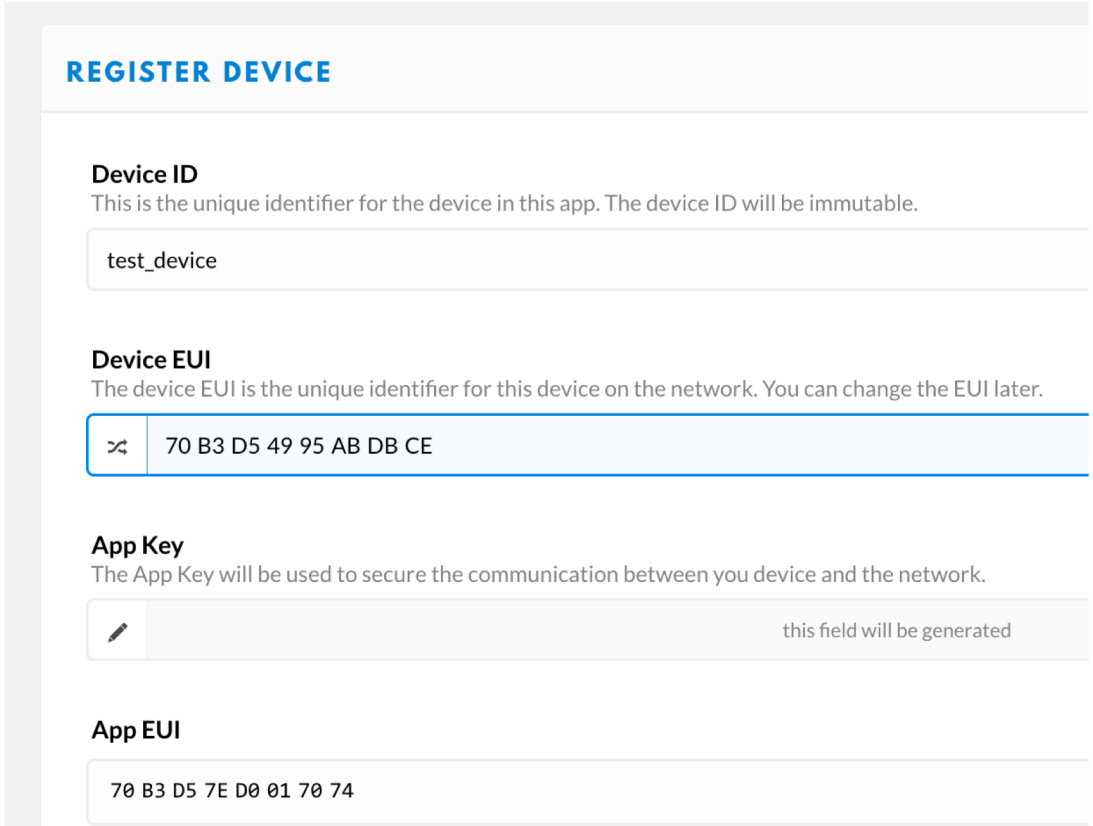
Device EUI for LoPy

To obtain the Device EUI of your LoPy, execute the following code in your REPL console:

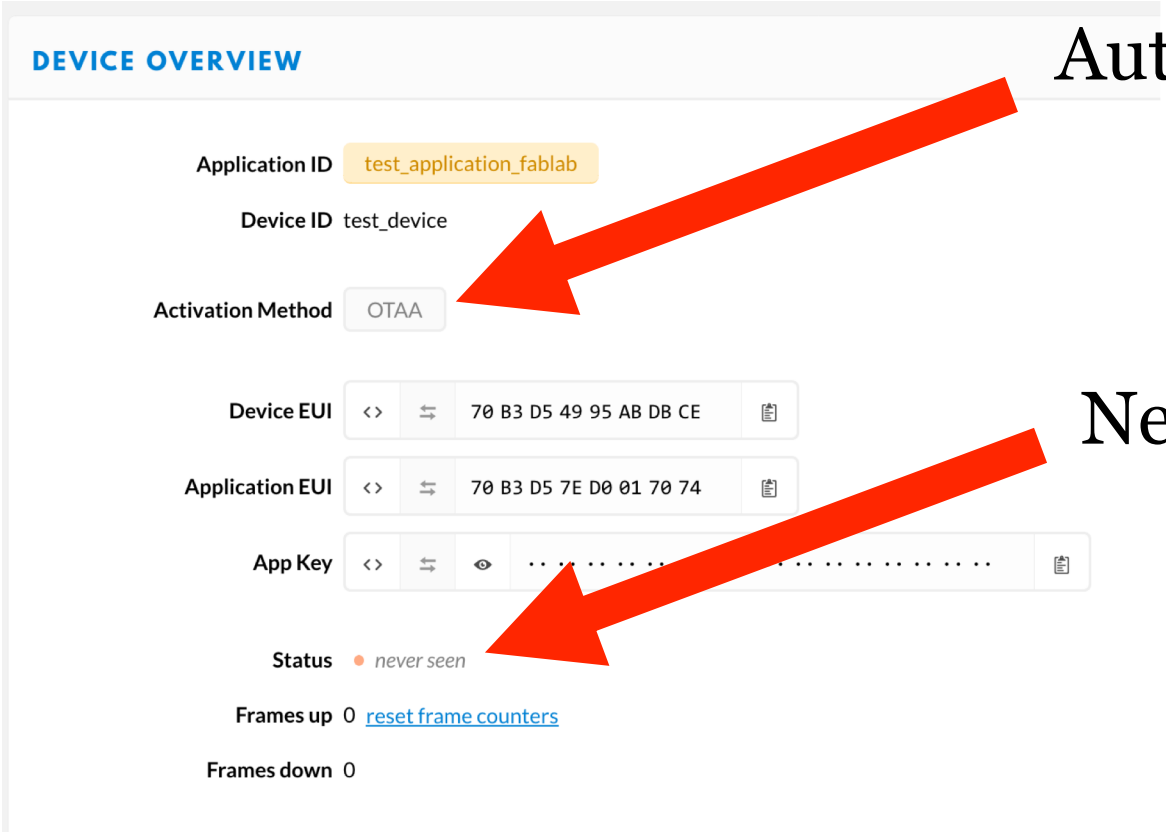
```
from network import LoRa
import binascii
lora = LoRa(mode=LoRa.LORAWAN)
print(binascii.hexlify(lora.mac()).upper().decode('utf-8'))
```

As an output you will receive a string that contains the Device EUI.

TTN: devices


REGISTER DEVICE
Device ID
This is the unique identifier for the device in this app. The device ID will be immutable.
test_device
Device EUI
The device EUI is the unique identifier for this device on the network. You can change the EUI later.
70 B3 D5 49 95 AB DB CE
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 01 70 74

TTN: devices

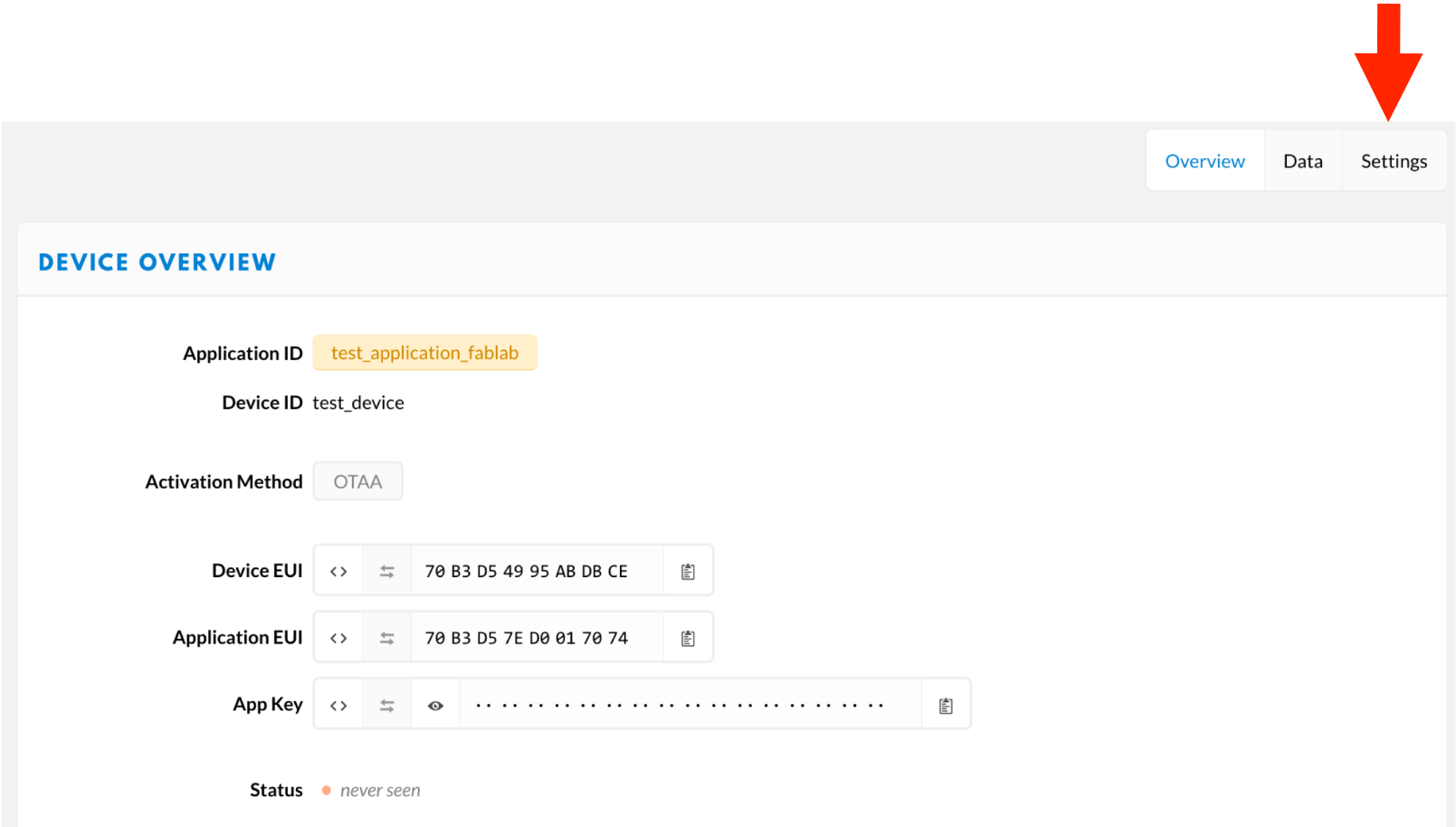


Authentication

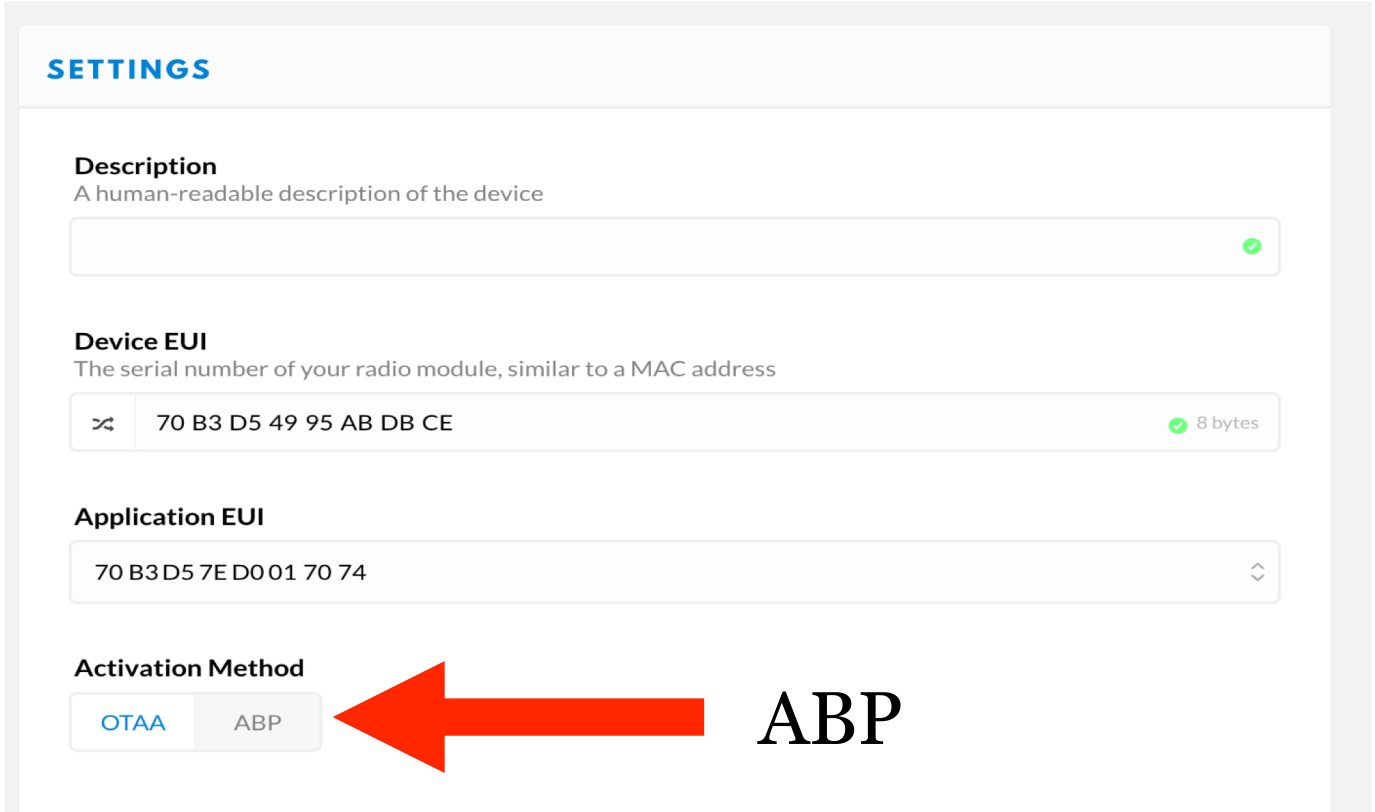
Never seen!

TTN: devices

Settings



TTN: devices


SETTINGS
Description
A human-readable description of the device
Device EUI
The serial number of your radio module, similar to a MAC address
70 B3 D5 49 95 AB DB CE
8 bytes
Application EUI
70 B3 D5 7E D0 01 70 74
Activation Method
OTAA
ABP
ABP

TTN: devices

Activation Method

OTAA | ABP

Device Address

The device address will be assigned by the network server

Network Session Key

Network Session Key will be generated

App Session Key

App Session Key will be generated

TTN: devices

DeviceAdd, NetKey, AppKey

TTN App: first example

Open the example in the Code/LoRa/TTN directory.

This example code sends a short message ”1,2,3” to TTN using ABP authentication.

TTN App: first example

```
dev_addr = struct.unpack(">l",
binascii.unhexlify('260118A2'))[0]

nwk_swkey =
binascii.unhexlify('F913FB6F4E47
169234163839D5A76787')

app_swkey =
binascii.unhexlify('CB4DECE3104
D7B5EB85AFFD8334E45E3')
```

Modify these values with the ones provided by TTN for your application

TTN App: first example

On TTN you should now be able to see the data coming in.

TTN App: T,H

Open the example in the Code/LoRa/TTN+Pysense/pycom directory.

This example code read T and H from the Pysense and sends this information via TTN.

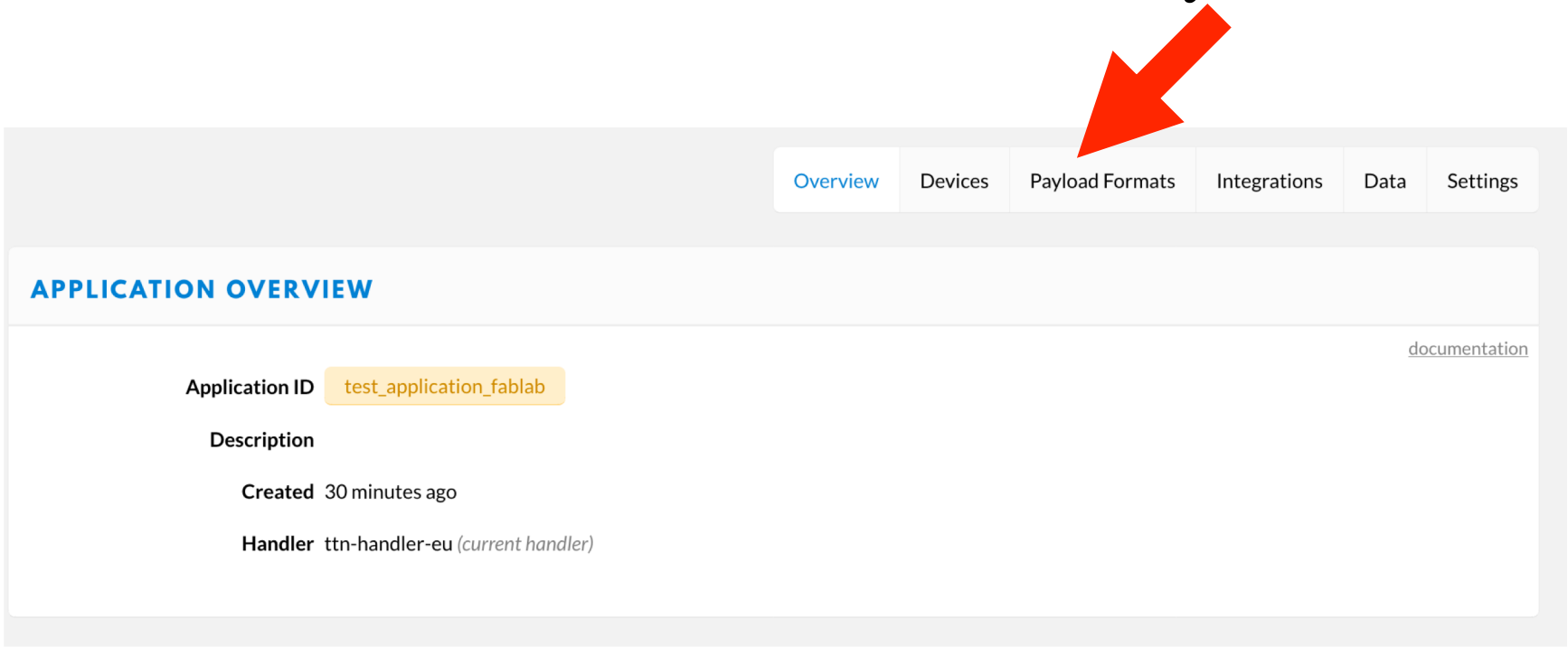
TTN App: T,H example

If your devices are transmitting data properly, all messages received will be seen in TTN.

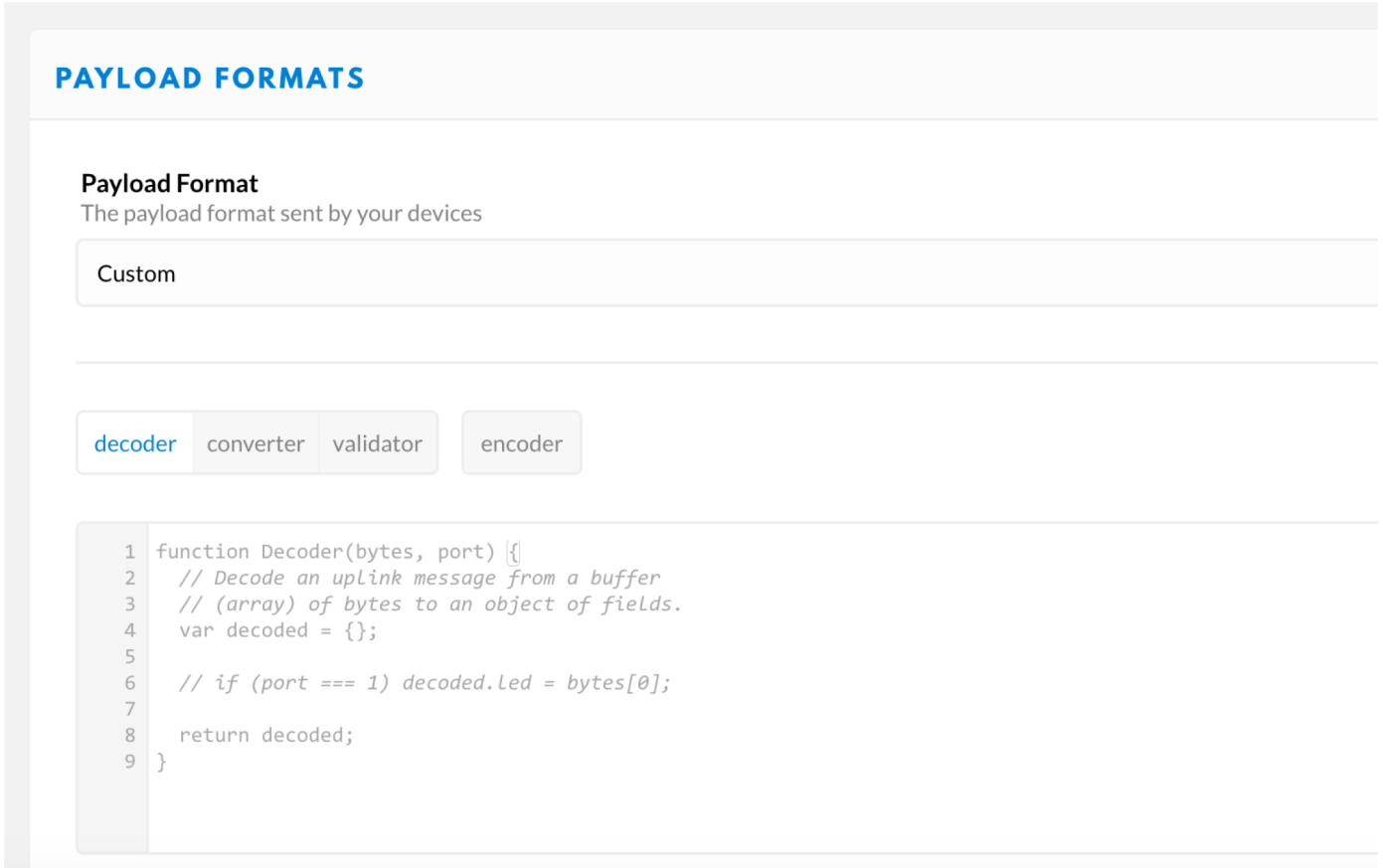
To check the incoming messages from the devices, go to the "Traffic" tab from gateway console.

TTN: payload

Payload format



TTN: payload


PAYLOAD FORMATS
Payload Format
The payload format sent by your devices
Custom
decoder
converter
validator
encoder
function Decoder(bytes, port) {
// Decode an uplink message from a buffer
// (array) of bytes to an object of fields.
var decoded = {};
// if (port === 1) decoded.led = bytes[0];
return decoded;
}

TTN: payload

Open the payload example in the Code/LoRa/TTN+Pysense/ttn-decoder directory.

Copy the decider as payload decoder in TTN.

TTN App: T,H example

On TTN you should now be able to see the data coming in and you should be able to decode the payload.

T,H TTN: Exercises

1) Move in the lab and check the RSSI values as seen by TTN. How far can you go?

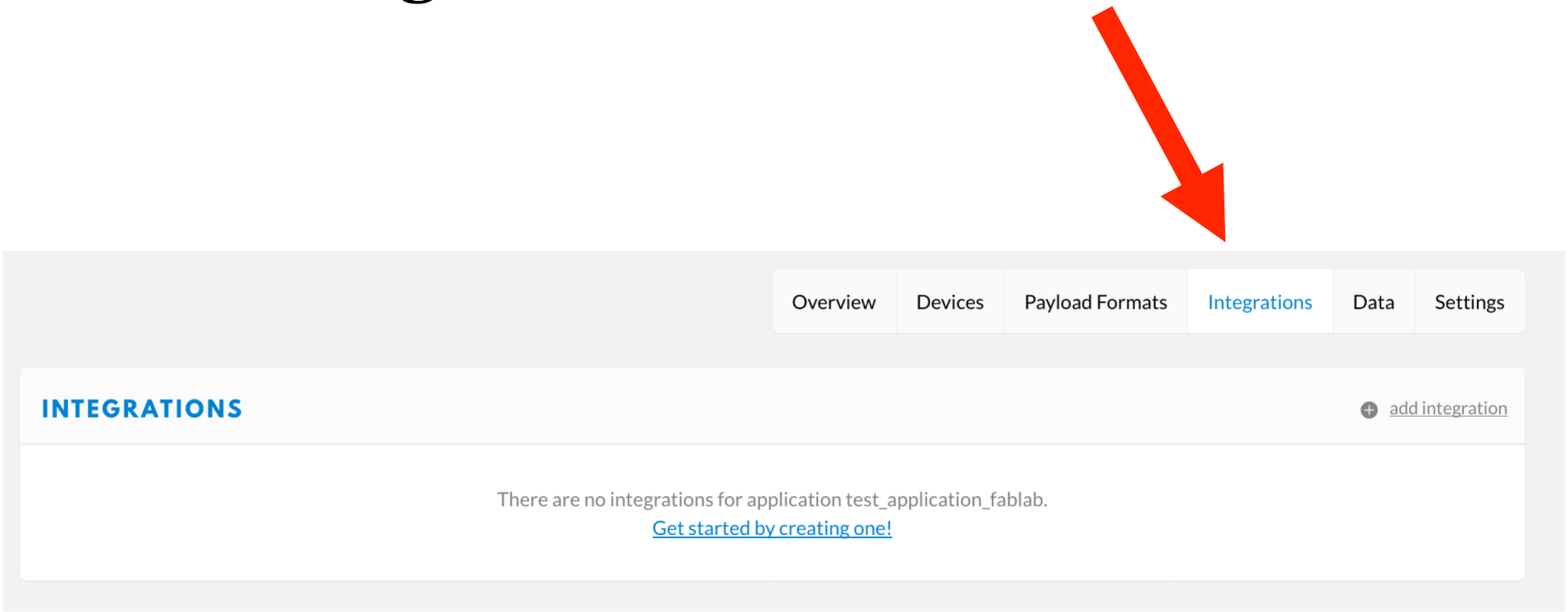
2) Send T,H to TTN and save the values in a log file in the flash memory.

Ubidots Intergration

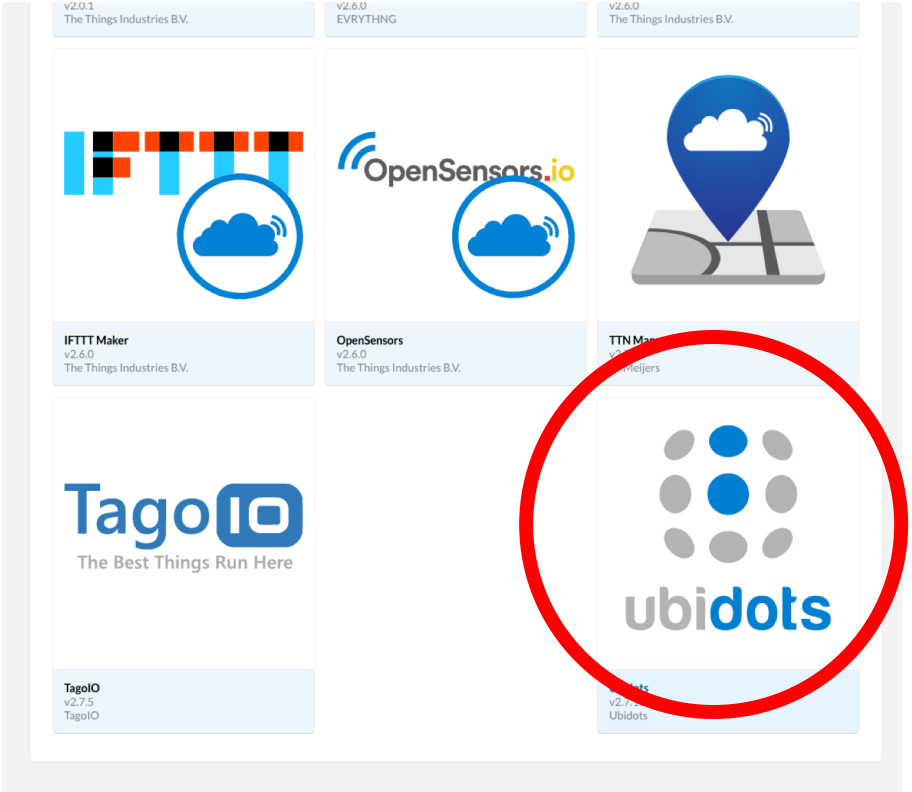
Credit: https://help.ubidots.com/developer-guides/integrate-your-ttn-data-with-ubidots-simple-setup

TTN: integrations

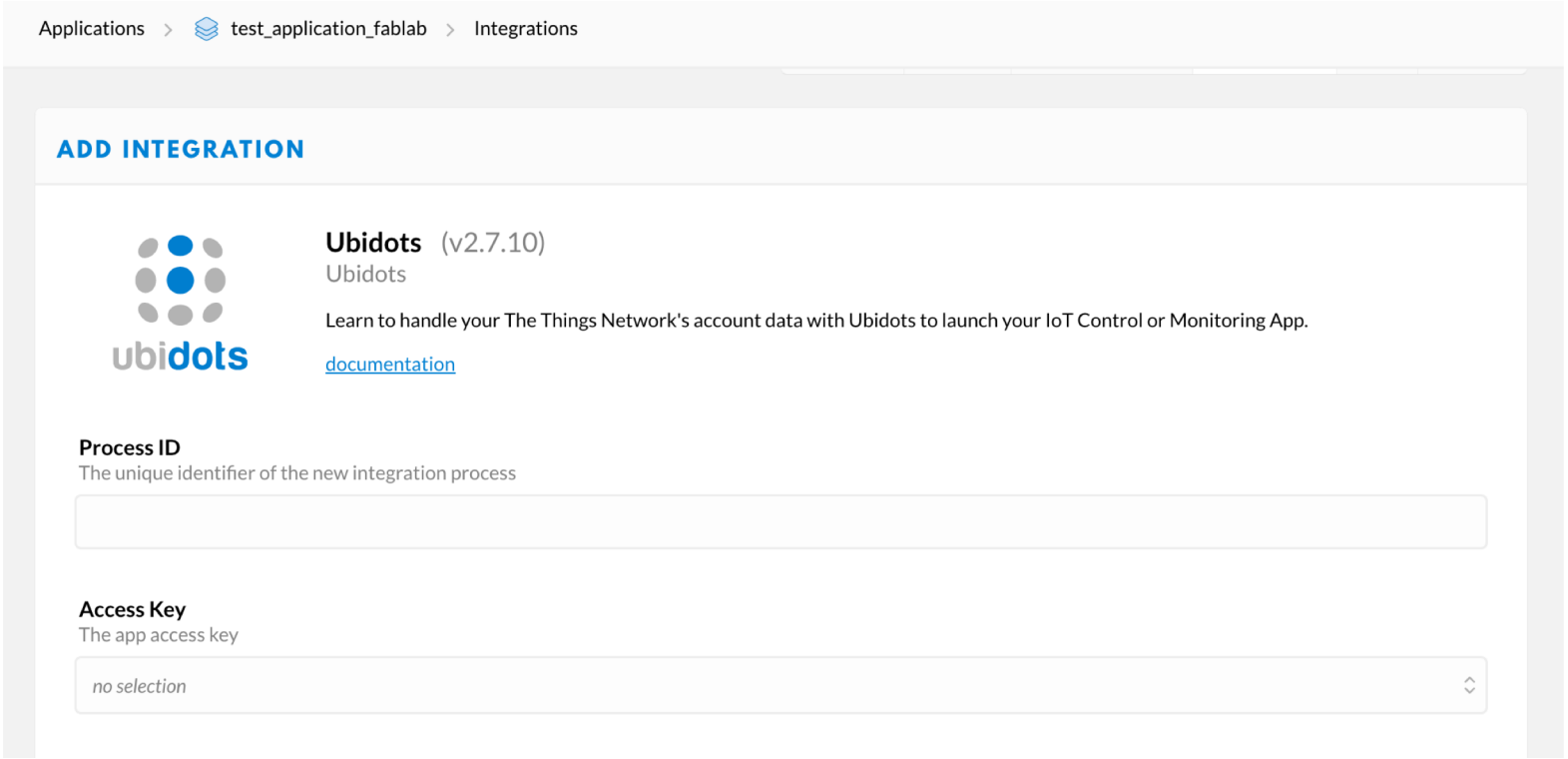
Integrations



TTN: integrations



TTN: integrations



Ubidots Intergration

Select "default key" in the Access Key dropdown menu. The default key represents a "password" that is used to authenticate your application in TTN.

Finally, you have to enter your Ubidots TOKEN where indicated in the TTN user interface.

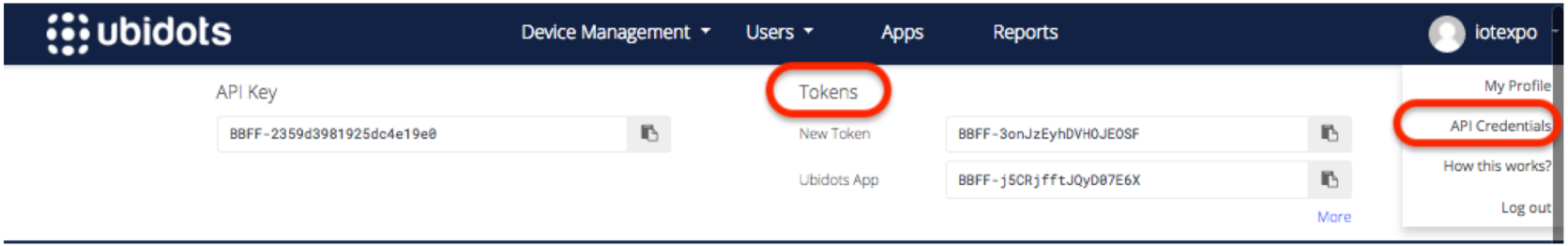
First, you must create an account on Ubidots:

https://industrial.ubidots.com/accounts/signup_industrial/

Ubidots Intergration

Sign into your Ubidots account. Go to you user dropdown and click on API credentials.

Tokens are temporary and revocable keys.



Ubidots Intergration

In the TTN Console enter your Ubidots TOKEN where indicated in the TTN user interface.

You will be able to see your LoRaWAN devices automatically created in your Ubidots account.

This integration will automatically use your DevEUI as the "Device API Label," which is the unique identifier within Ubidots.

Summary

We learned how to create a TTN gateway and register it.

We learned how to send data to TTN.

We visualized data using the Ubidots integration.

Feedback?

Email mzennaro@ictp.it