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

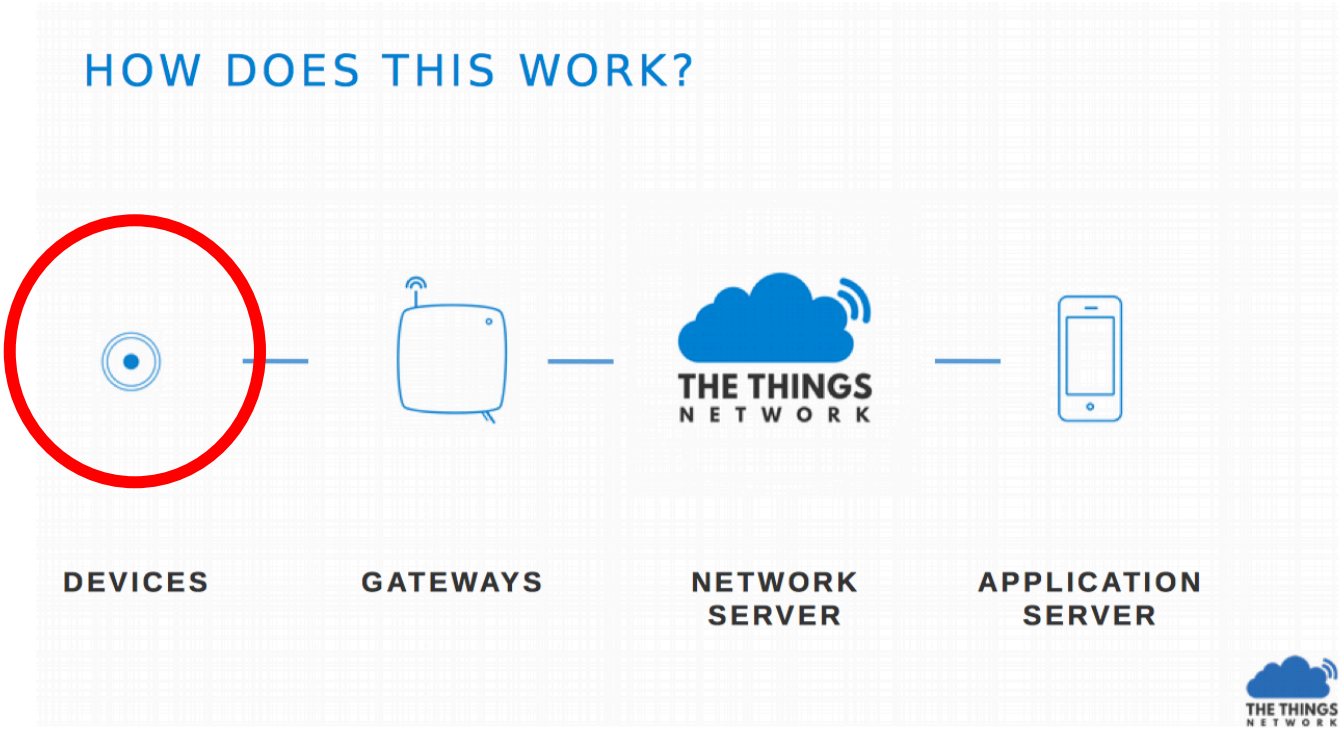
Pycom LoPy 4

PySense

microUSB Cable

LoRaWAN Gateway

TTN: devices, gateways, servers



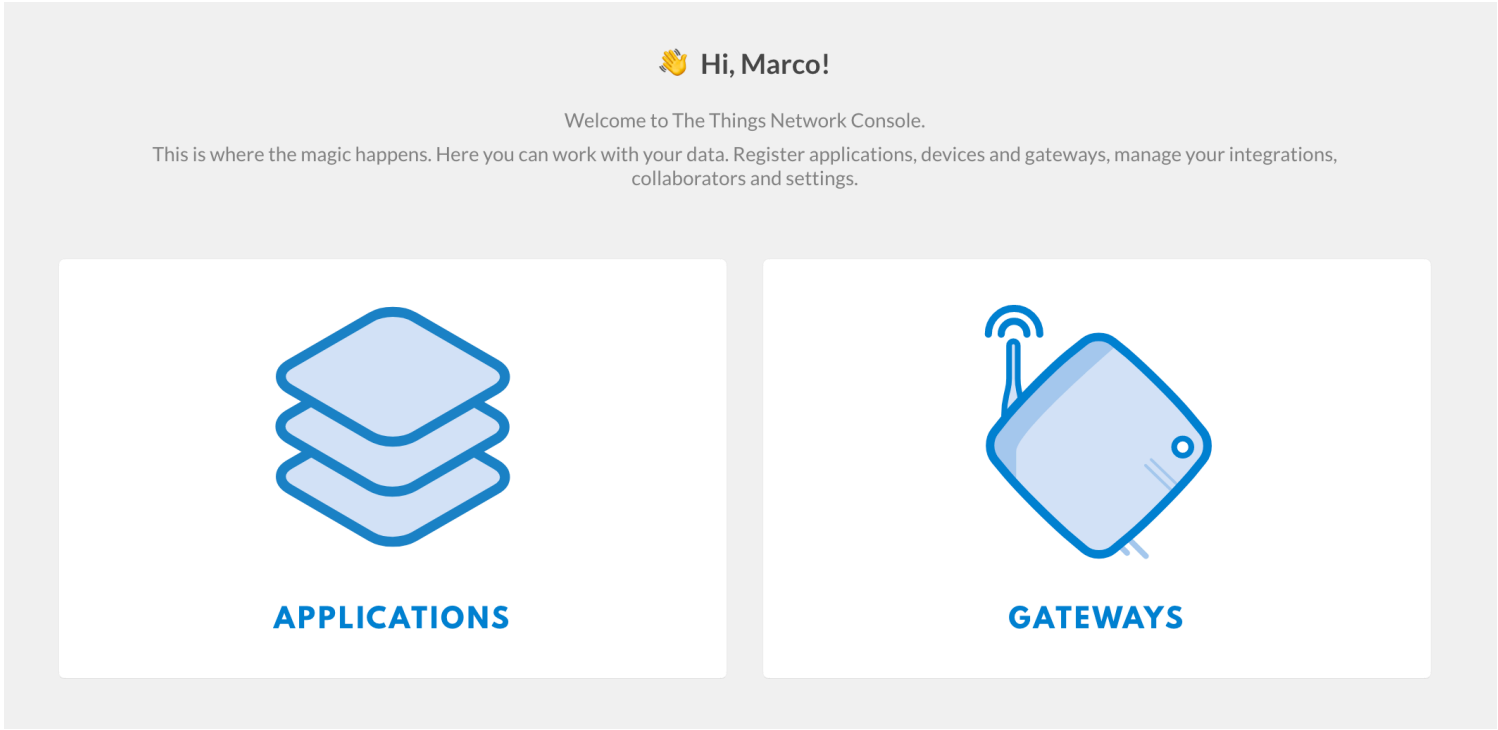
Sending T,H to TTN

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

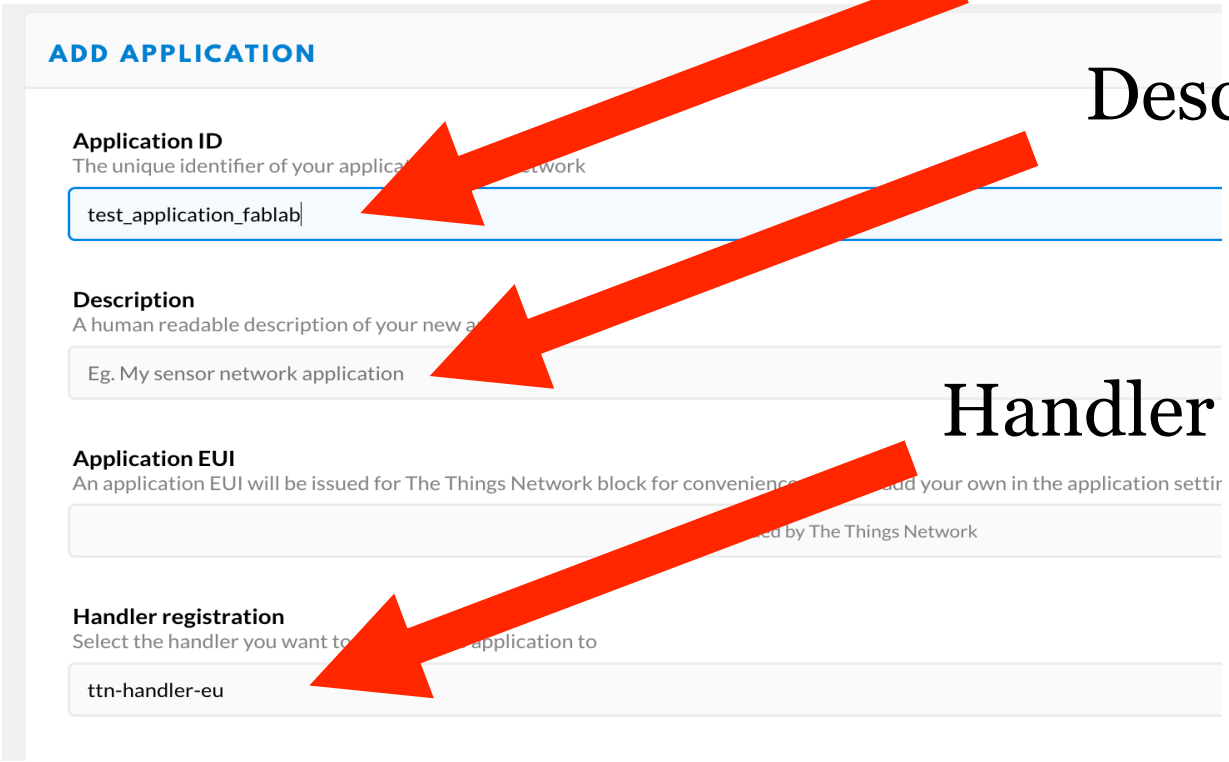

Hi, Marco!
Welcome to The Things Network Console.
This is where the magic happens. Here you can work with your data. Register applications, devices and gateways, manage your integrations, collaborators and settings.
APPLICATIONS
GATEWAYS

TTN: App

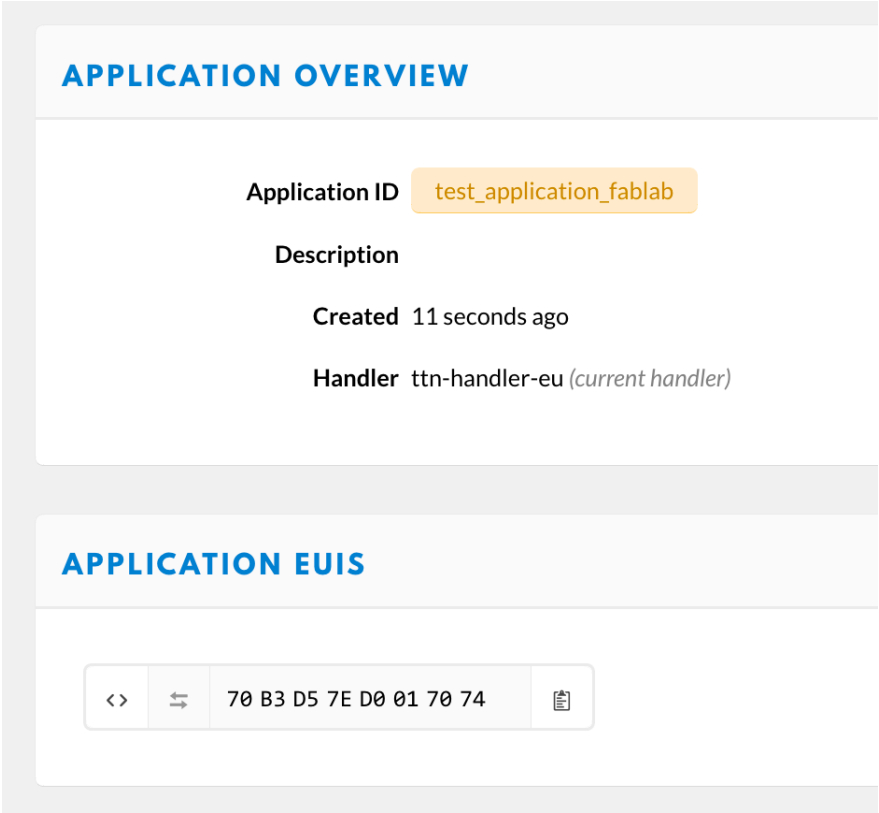
Application ID

Description

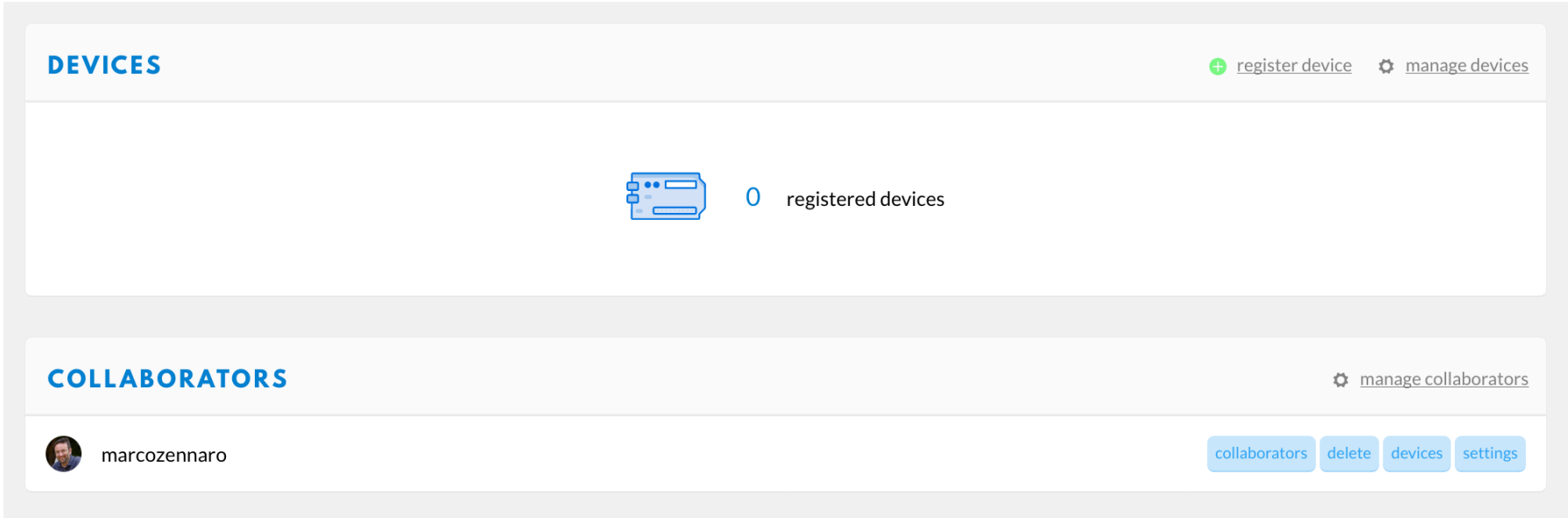
Handler (Europe)



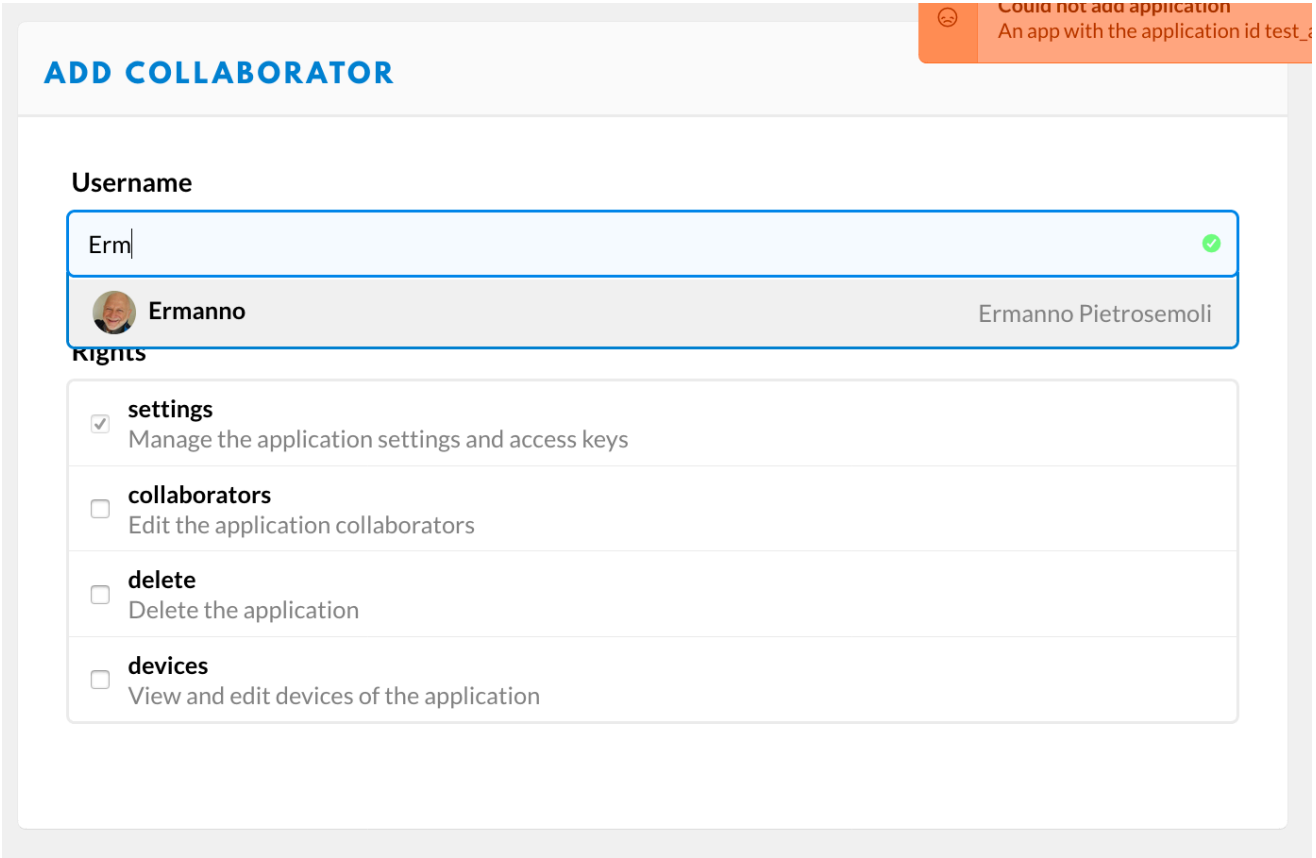
TTN: we have a new App!



TTN: Collaborators



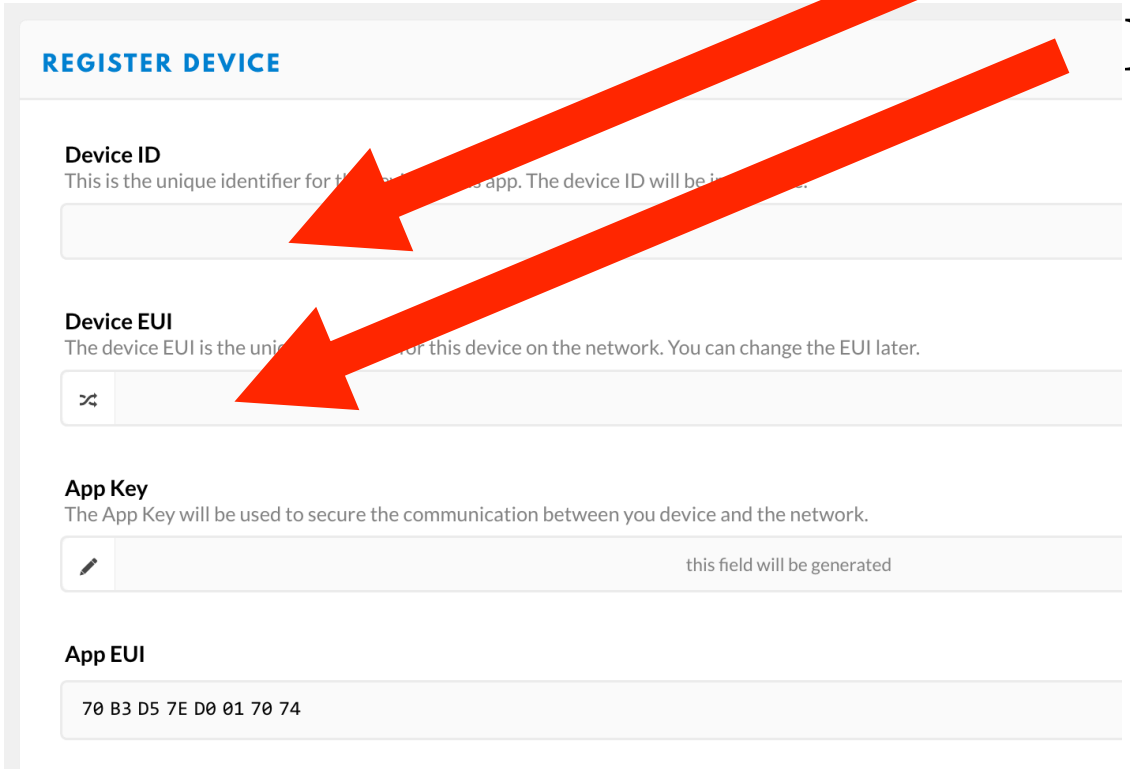
TTN: add a Collaborator to the App



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:

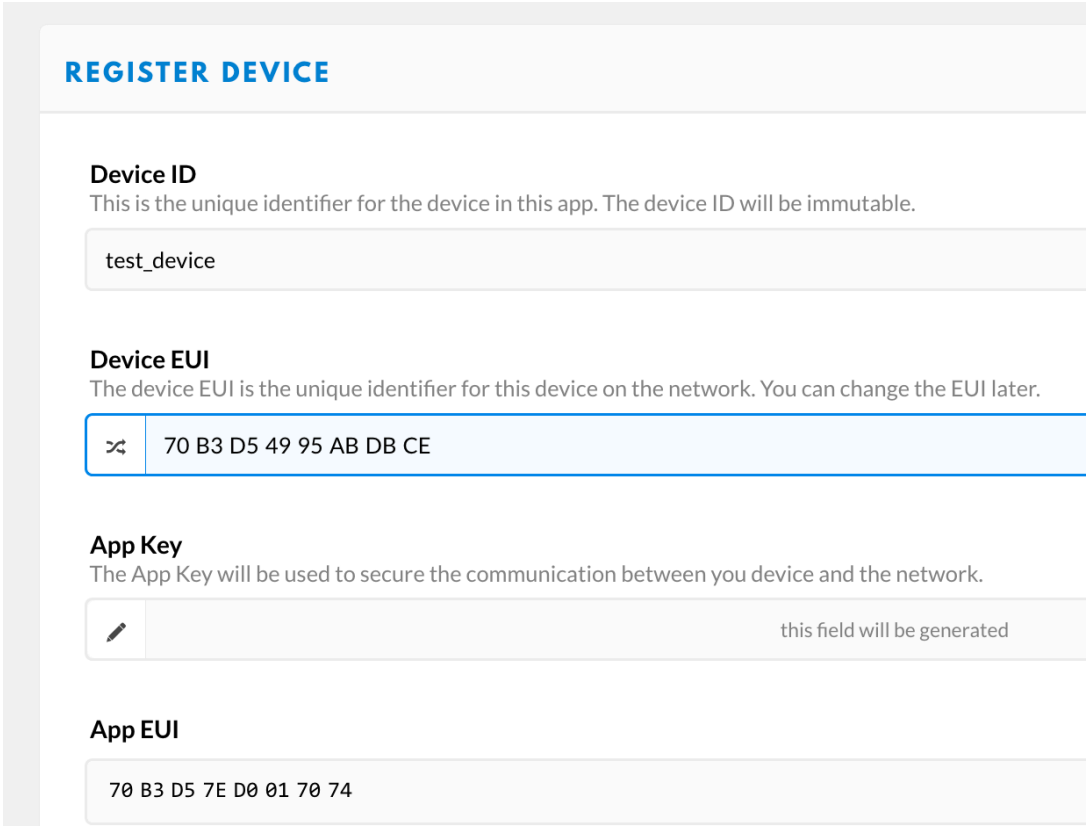
Device EUI for LoPy

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

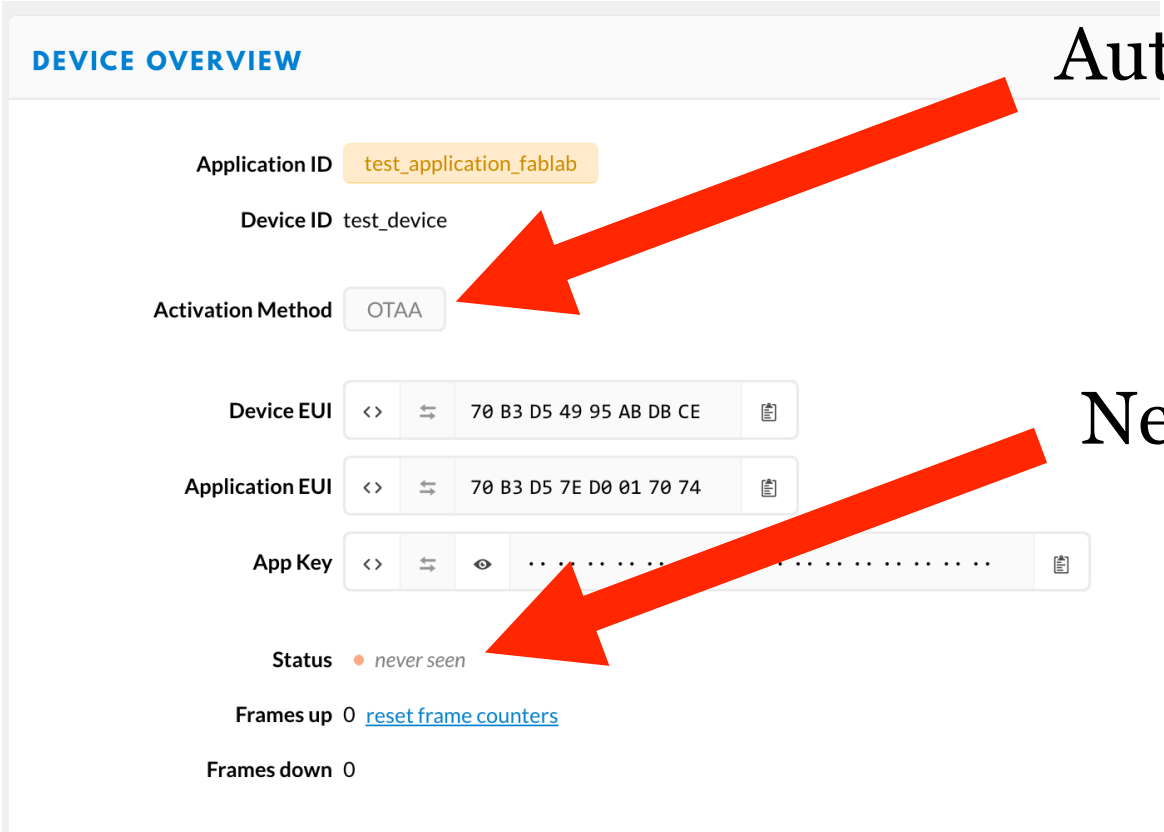
```
from network import LoRa
import binascii
lora = LoRa(mode=LoRa.LORAWAN)
print(binascii.hexlify(lora.mac()).upper().decode('utf-8'))
```

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

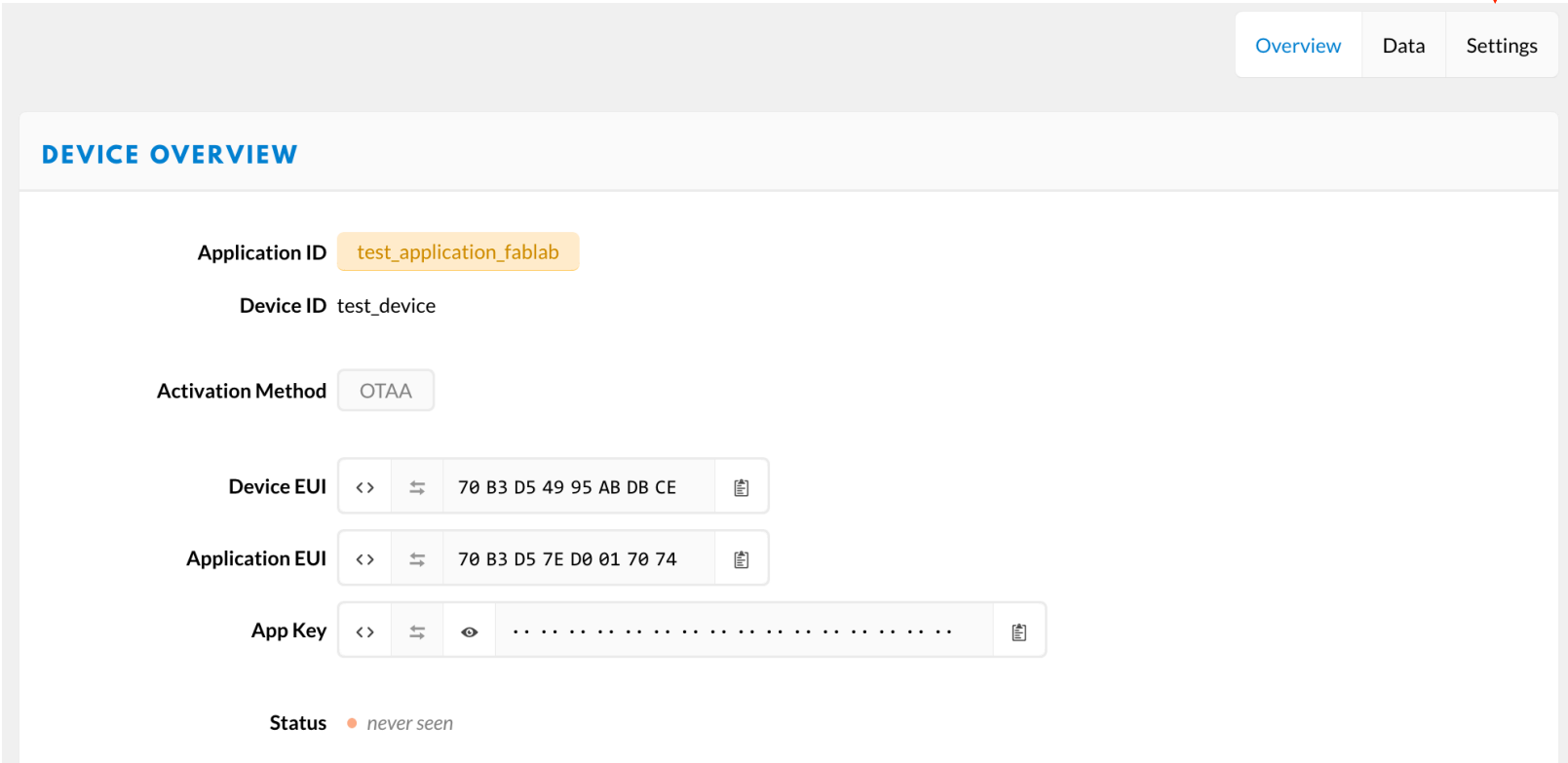


Authentication

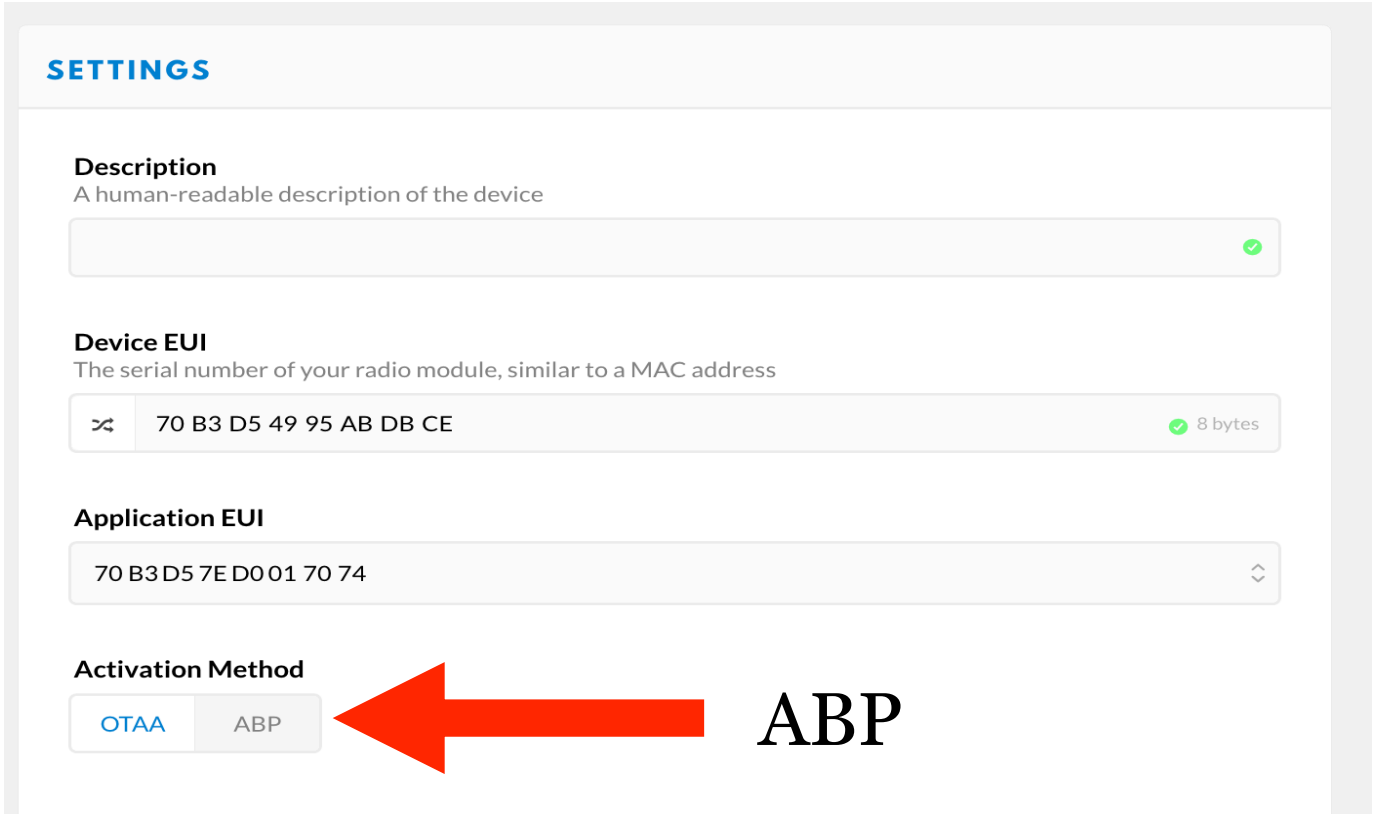
Never seen!

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 reads T and H from the Pysense and sends this information via TTN.

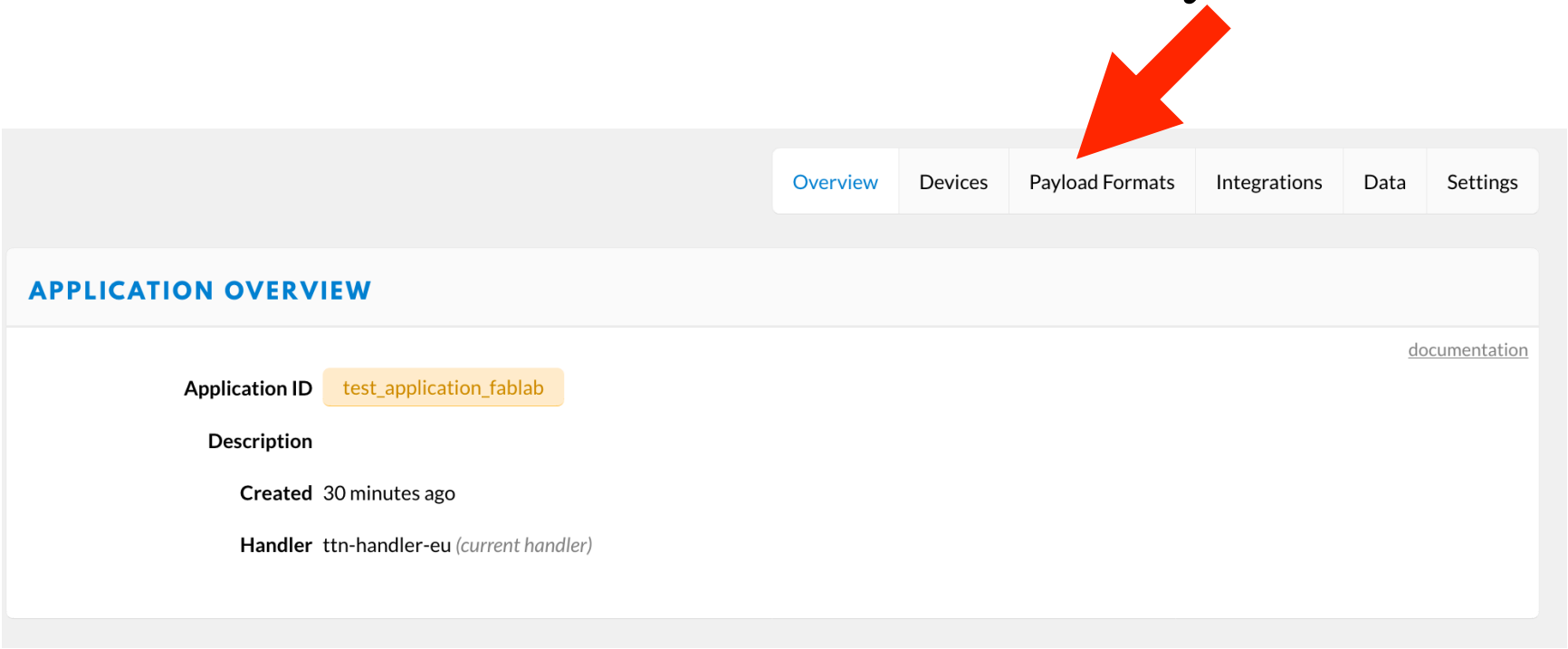
TTN App: T,H example

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

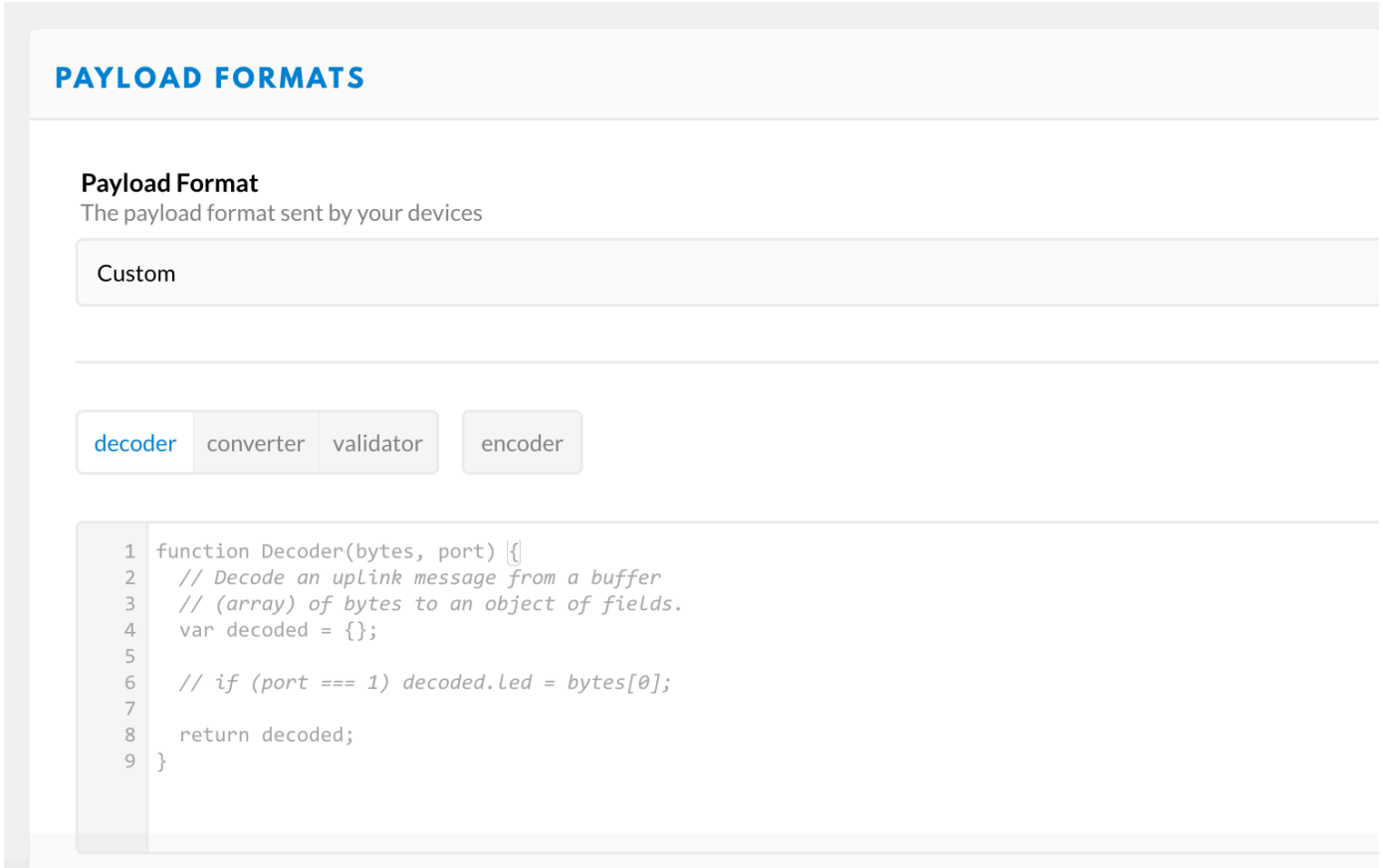
To check the incoming messages from the devices, go to the "Traffic" tab from gateway console.

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 decoder 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?

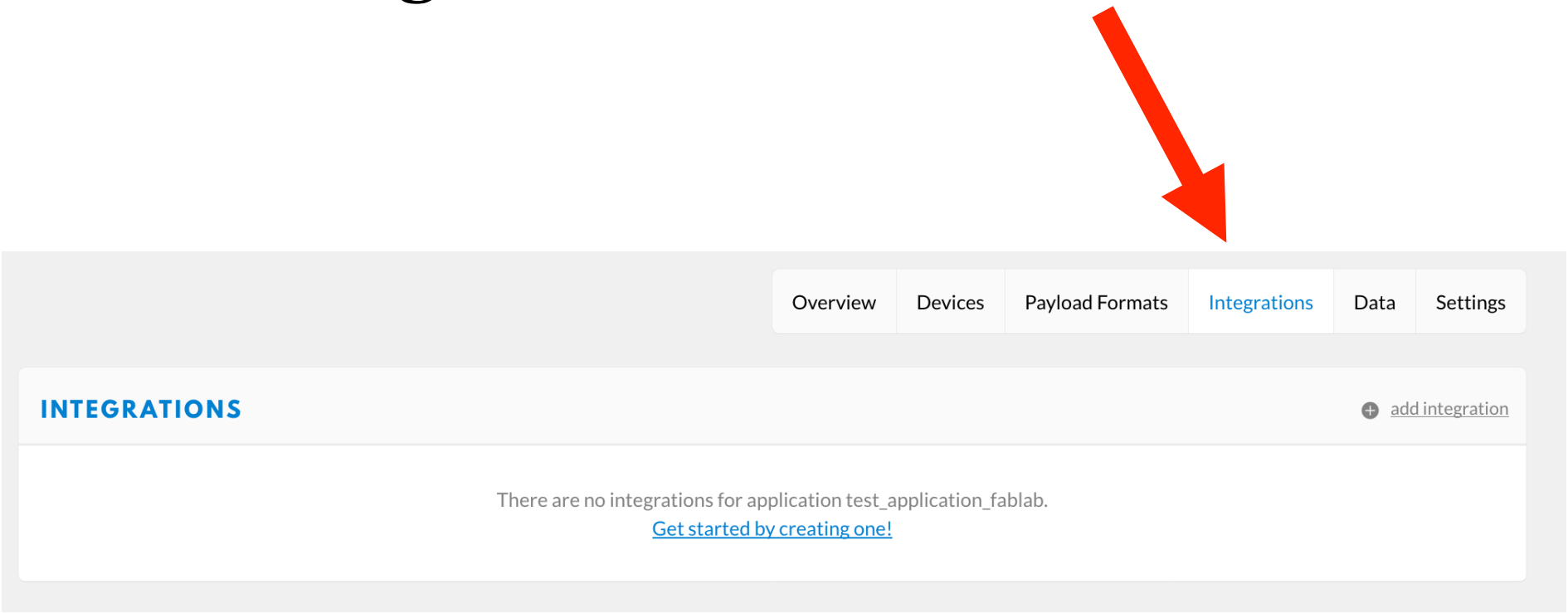
2) Send light readings to TTN (you have to decode the payload).

Ubidots Intergration

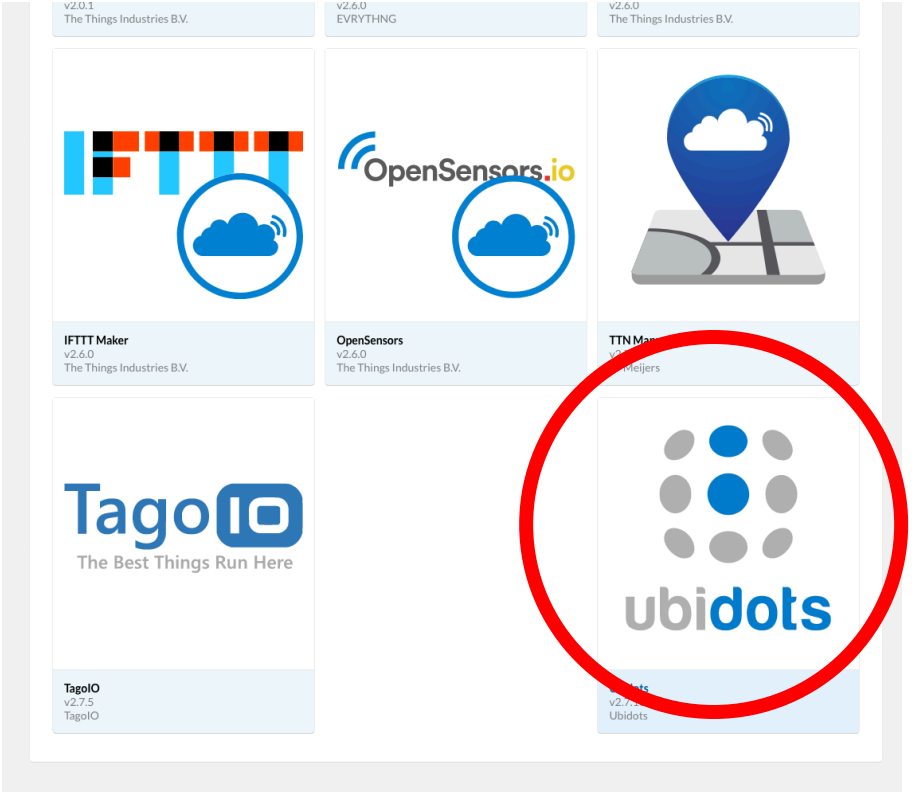
Credit: https://help.ubidots.com/developer-guides/integrate-your-ttn-data-with-ubidots-simple-setup

TTN: integrations

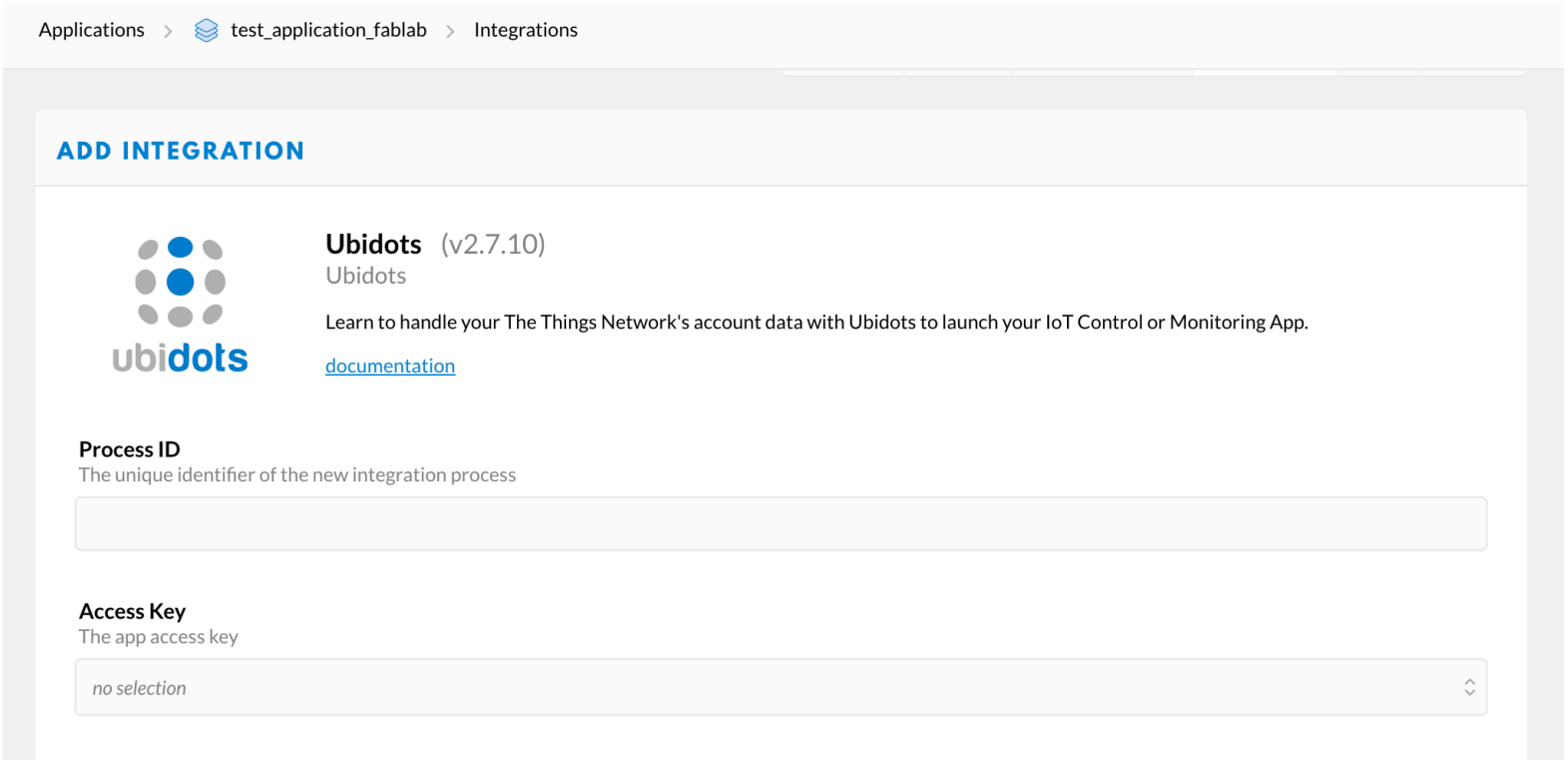
Integrations



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.

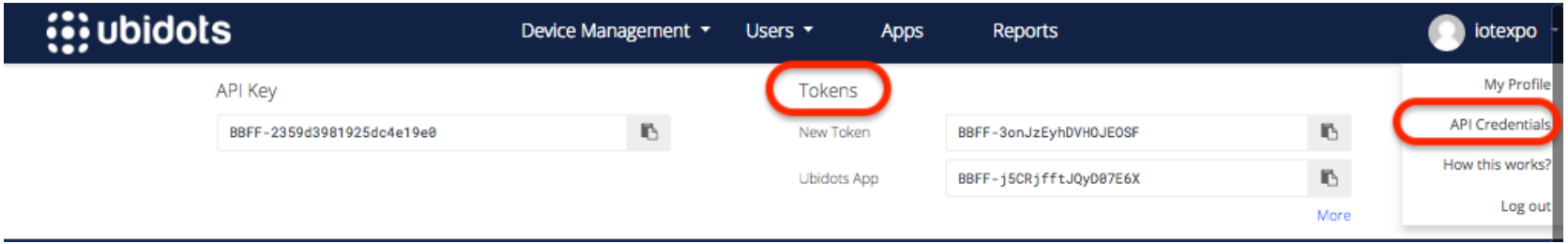
First, you must create an account on Ubidots:

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 send data to TTN.

We visualized data using the Ubidots integration.

Feedback?

Email mzennaro@ictp.it