

Lab3: LoRaWAN and TTN

Marco Zennaro, PhD
ICTP



Labs

- 1/3 Ready to use, tested examples
- 1/3 Exercise based on the examples
- 1/3 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

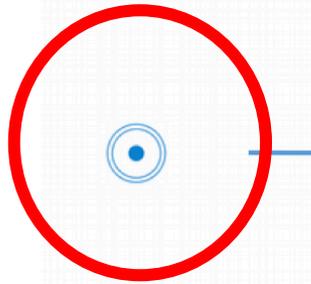
PySense

microUSB Cable

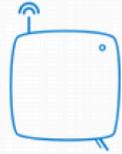
LoRaWAN Gateway

TTN: devices, gateways, servers

HOW DOES THIS WORK?



DEVICES



GATEWAYS



NETWORK
SERVER



APPLICATION
SERVER



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 application on the network

test_application_fablab

Description
A human readable description of your new application

Eg. My sensor network application

Application EUI
An application EUI will be issued for The Things Network block for convenience. You can also add your own in the application settings.

Issued by The Things Network

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

ttn-handler-eu

TTN: we have a new App!

APPLICATION OVERVIEW

Application ID `test_application_fablab`

Description

Created 11 seconds ago

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

APPLICATION EUI64

<> ⇕ 70 B3 D5 7E D0 01 70 74 📄

TTN: Collaborators

DEVICES

[+ register device](#) [manage devices](#)



0 registered devices

COLLABORATORS

[manage collaborators](#)



marcozennaro

[collaborators](#) [delete](#) [devices](#) [settings](#)

TTN: add a Collaborator to the App

ADD COLLABORATOR

Could not add application
An app with the application id test_4

Username

Erm ✓

 **Ermanno** Ermanno Pietrosemoli

Rights

- settings**
Manage the application settings and access keys
- collaborators**
Edit the application collaborators
- delete**
Delete the application
- devices**
View and edit devices of the application

TTN: register a device

Name of Device

Device EUI

REGISTER DEVICE

Device ID
This is the unique identifier for the device on this app. The device ID will be used to identify the device.

Device EUI
The device EUI is the unique identifier for this device on the network. You can change the EUI later.
App Key
The App Key will be used to secure the communication between you device and the network.
App EUI
70 B3 D5 7E D0 01 70 74

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!

DEVICE OVERVIEW

Application ID `test_application_fablab`

Device ID `test_device`

Activation Method `OTAA`

Device EUI `<> ↕ 70 B3 D5 49 95 AB DB CE` 

Application EUI `<> ↕ 70 B3 D5 7E D0 01 70 74` 

App Key `<> ↕ ` 

Status ● *never seen*

Frames up 0 [reset frame counters](#)

Frames down 0

TTN: devices

Settings



Overview Data **Settings**

DEVICE OVERVIEW

Application ID `test_application_fablab`

Device ID `test_device`

Activation Method `OTAA`

Device EUI `<> ⇄ 70 B3 D5 49 95 AB DB CE`

Application EUI `<> ⇄ 70 B3 D5 7E D0 01 70 74`

App Key `<> ⇄ 👁`

Status ● *never seen*



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

 8 bytes

Application EUI

Activation Method

OTAA 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



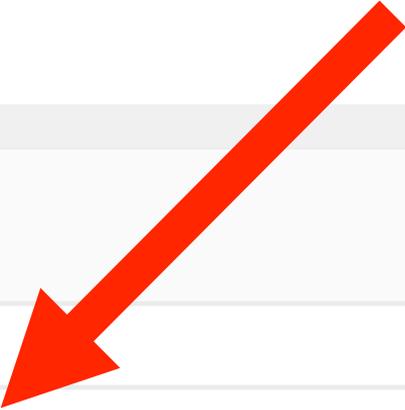
App Session Key will be generated

TTN: devices

DeviceAdd, NetKey, AppKey

EXAMPLE CODE

```
1 const char *devAddr = "26011607";  
2 const char *nwksKey = "09827AA1D4BBDB382859F47A49F6C20B";  
3 const char *appSKey = "6B54FDB99BF4A1E90A768C3B5FAD3F50";
```



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(">I",  
binascii.unhexlify('260118A2'))[0]
```

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

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

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

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



Overview

Devices

Payload Formats

Integrations

Data

Settings

APPLICATION OVERVIEW

[documentation](#)

Application ID `test_application_fablab`

Description

Created 30 minutes ago

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

TTN: payload

PAYLOAD FORMATS

Payload Format

The payload format sent by your devices

Custom

decoder

converter

validator

encoder

```
1 function Decoder(bytes, port) {  
2   // Decode an uplink message from a buffer  
3   // (array) of bytes to an object of fields.  
4   var decoded = {};  
5  
6   // if (port === 1) decoded.lcd = bytes[0];  
7  
8   return decoded;  
9 }
```



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

TTN: integrations

Integrations



Overview Devices Payload Formats **Integrations** Data Settings

INTEGRATIONS

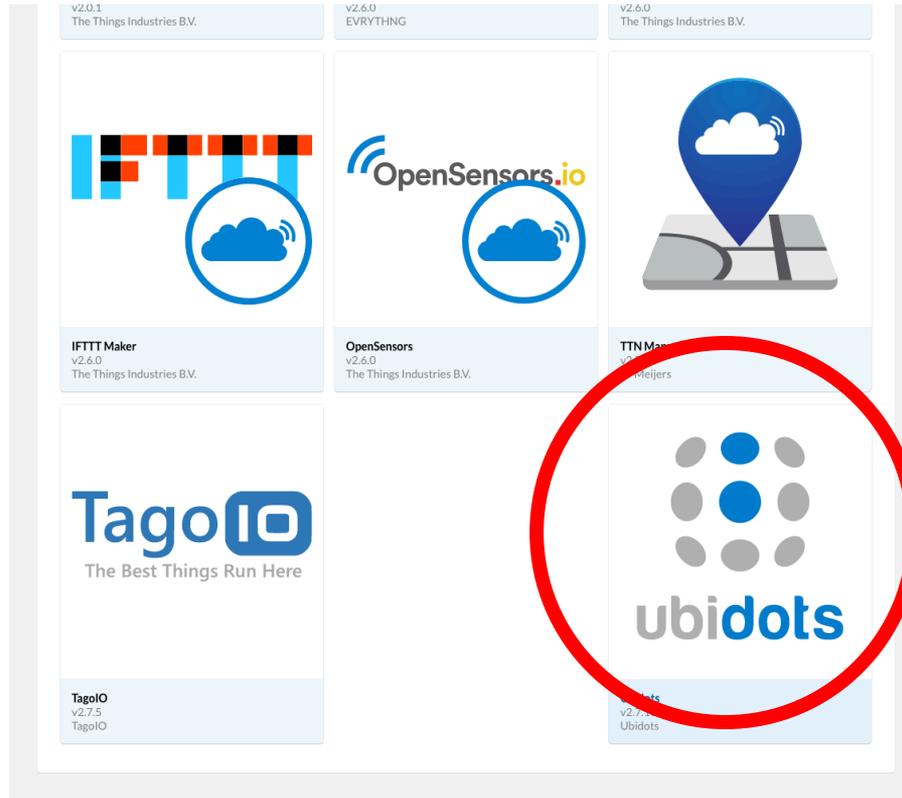
+ [add integration](#)

There are no integrations for application test_application_fablab.

[Get started by creating one!](#)



TTN: integrations



TTN: integrations

Applications > test_application_fablab > Integrations

ADD INTEGRATION



Ubidots (v2.7.10)

Ubidots

Learn to handle your The Things Network's account data with Ubidots to launch your IoT Control or Monitoring App.

[documentation](#)

Process ID

The unique identifier of the new integration process

Access Key

The app access key



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.

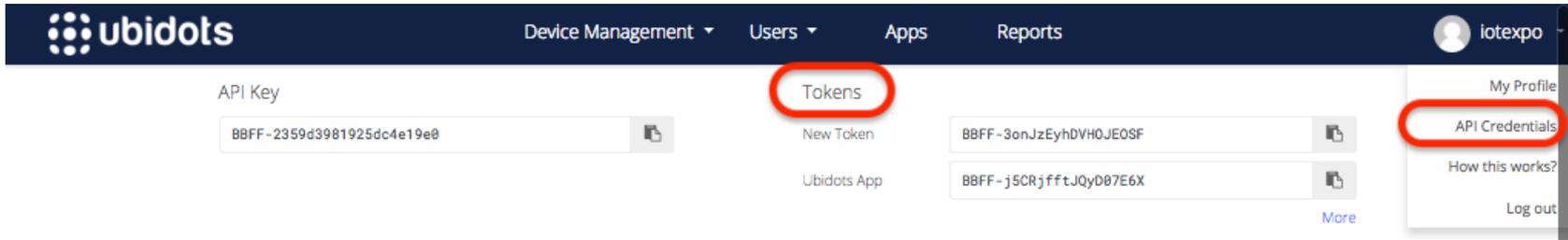
First, you must create an account on Ubidots:

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.



The screenshot displays the Ubidots dashboard interface. At the top, there is a dark blue navigation bar with the Ubidots logo on the left and navigation links for 'Device Management', 'Users', 'Apps', and 'Reports' in the center. On the right side of the navigation bar, the user's profile 'iotexpo' is visible. Below the navigation bar, the main content area is divided into three sections: 'API Key', 'Tokens', and 'API Credentials'. The 'API Key' section shows a text input field containing the key 'BBFF-2359d3981925dc4e19e0'. The 'Tokens' section is highlighted with a red circle and contains a 'New Token' button and a list of tokens, including 'BBFF-3onJzEyhDVH0JE0SF' and 'BBFF-j5CRjfftJQyD07E6X'. The 'API Credentials' section is also highlighted with a red circle and contains a 'Log out' button. A 'More' link is visible at the bottom right of the 'API Credentials' section.

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