

# 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

# Our Lab equipment

Pycom LoPy 4

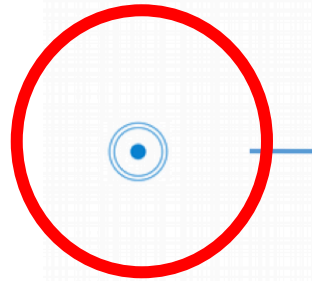
PySense

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.

**Handler registration**  
Select the handler you want to connect 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 EUIS

<>



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

Could not add application  
An app with the application id test\_2

## ADD COLLABORATOR

**Username**



 **Ermanno** Ermanno Pietrosemoli

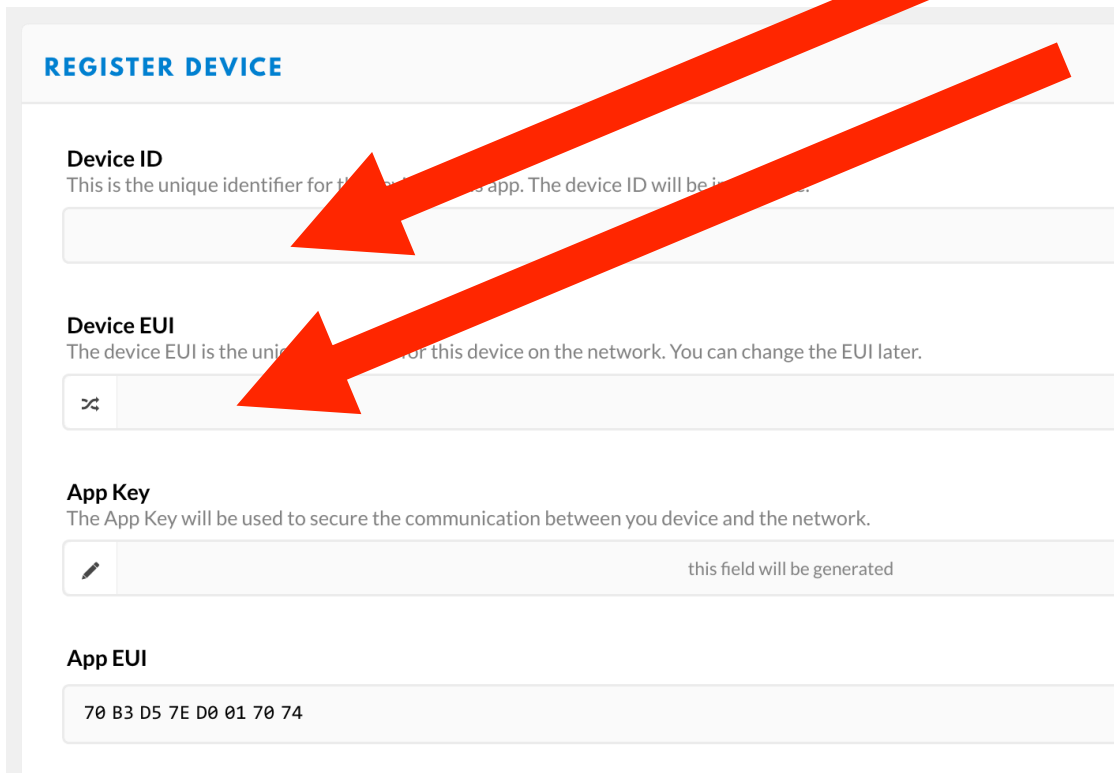
**Rights**

<input checked="" type="checkbox"/>	<b>settings</b> Manage the application settings and access keys
<input type="checkbox"/>	<b>collaborators</b> Edit the application collaborators
<input type="checkbox"/>	<b>delete</b> Delete the application
<input type="checkbox"/>	<b>devices</b> 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 in this app. The device ID will be used to identify the device in the app.

**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

 70 B3 D5 49 95 AB DB CE  8 bytes

**Application EUI**

70 B3D5 7E D001 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

## 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

In TTN's Application, 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

Application ID

test\_application\_fablab

Description

Created

30 minutes ago

Handler

ttn-handler-eu (current handler)

[documentation](#)

# 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 (so you can read Temperature and Humidity).

# T,H TTN: Exercises

- 1) Move in the lab and check the RSSI values as seen by TTN. How far can you go?
- 2) Create one Application for the whole class, and add ALL devices to it.

# Ubidots Intergration

# TTN: integrations

## Integrations



Overview

Devices

Payload Formats

Integrations

Data

Settings

### INTEGRATIONS

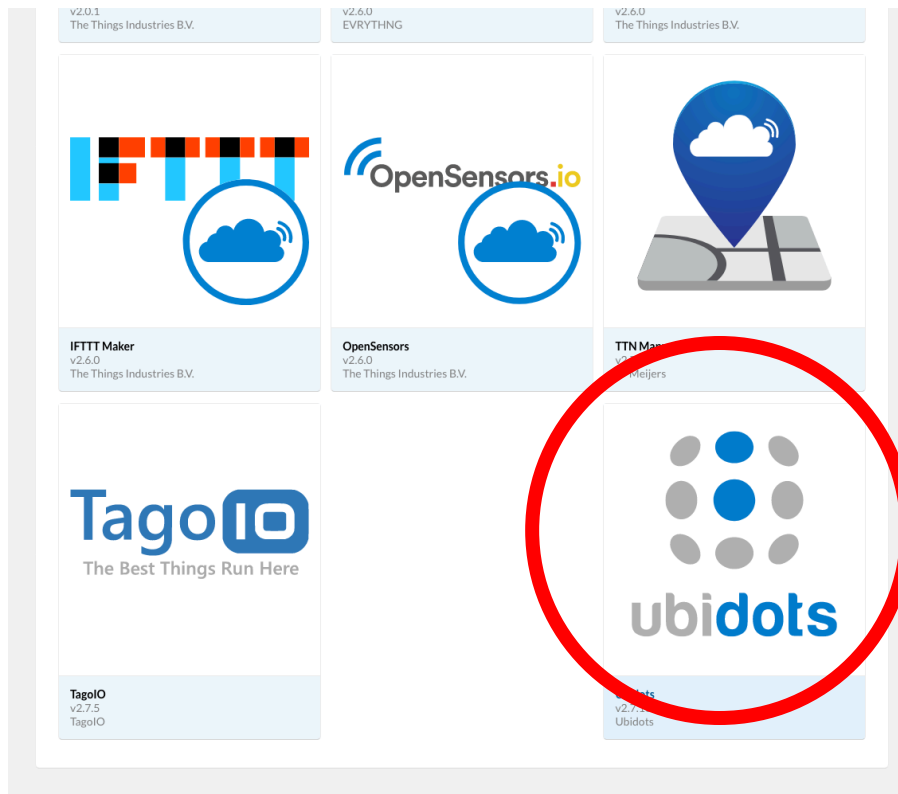
 [add integration](#)

There are no integrations for application test\_application\_fablab.

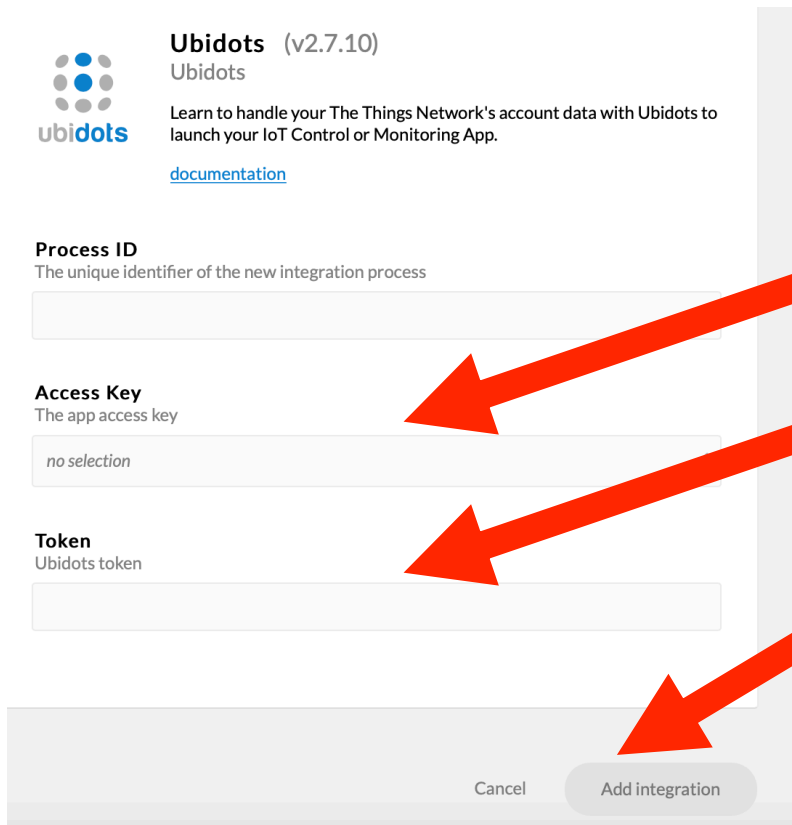
[Get started by creating one!](#)



# TTN: integrations



# TTN: integrations



The screenshot shows a web interface for adding a new integration. At the top, there is a header for 'Ubidots (v2.7.10)' with its logo and a brief description: 'Learn to handle your The Things Network's account data with Ubidots to launch your IoT Control or Monitoring App.' Below this is a link to 'documentation'. The main form consists of three input fields: 'Process ID' (with a description 'The unique identifier of the new integration process'), 'Access Key' (with a description 'The app access key' and a placeholder 'no selection'), and 'Token' (with a description 'Ubidots token'). At the bottom of the form are two buttons: 'Cancel' and '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  
no selection

**Token**  
Ubidots token

Cancel Add integration

default key

Ubidots token

Add  
integration

# 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> 

# Ubidots Intergration



Devices ▾

Data ▾

Users ▾

Apps

30 days left on trial

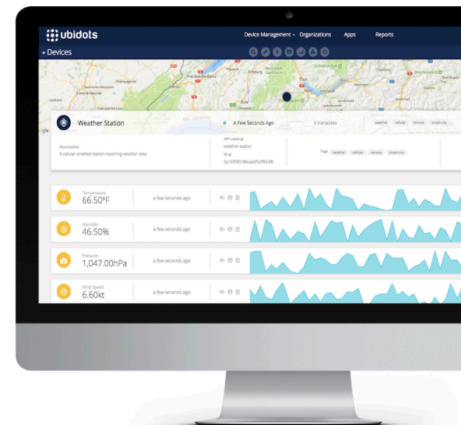


Welcome to Ubidots

Hey,

Welcome to Ubidots

In this onboarding experience we will be walking you through Ubidots, and some of its available tools to optimize your cloud-connected solution.



# Ubidots Intergration



Devices ▾

Data ▾

Users ▾

Apps

30 days left on trial



Welcome to Ubidots

Hey,

Welcome to Ubidots

In this onboarding experience we will be walking you through Ubidots, and some of its available tools to optimize your cloud-connected solution.

Username:  
mzennaro

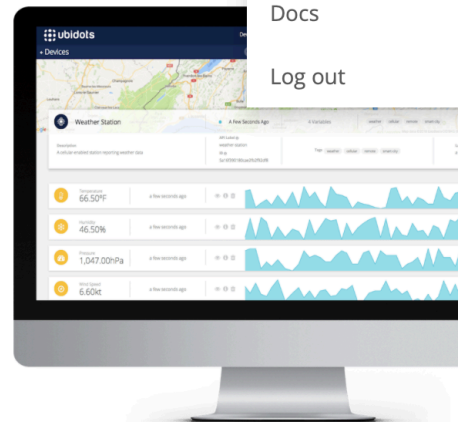
My Profile

API Credentials

How this works?

Docs

Log out



# Ubidots Intergration



Devices ▾

Data ▾

Users ▾

Apps

30 days left on trial



API Key

Click to show



Tokens

Default token

Click to show



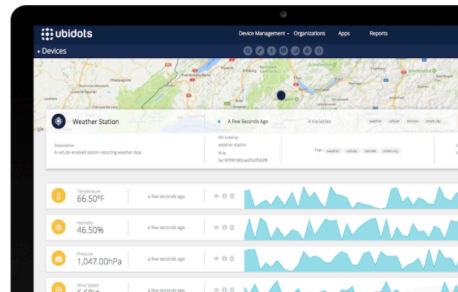
[More](#)

## Welcome to Ubidots

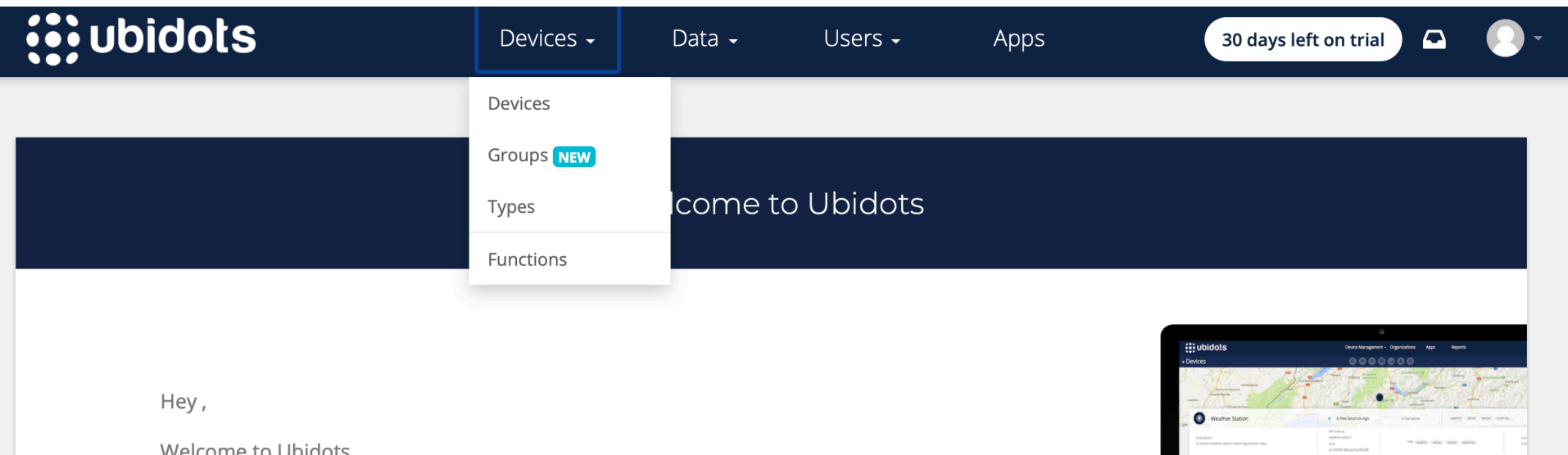
Hey,

Welcome to Ubidots

In this onboarding experience we will be walking you through Ubidots, and some of its available tools to optimize your cloud-connected solution.



# Ubidots Intergration



The screenshot displays the Ubidots web application interface. At the top, a dark blue navigation bar contains the Ubidots logo on the left and several menu items: 'Devices' (highlighted with a blue border and a dropdown arrow), 'Data' (with a dropdown arrow), 'Users' (with a dropdown arrow), and 'Apps'. On the right side of the navigation bar, there is a white pill-shaped button indicating '30 days left on trial', a notification icon, and a user profile icon. A dropdown menu is open under the 'Devices' menu, listing 'Devices', 'Groups' (marked with a blue 'NEW' badge), 'Types', and 'Functions'. Below the navigation bar, a large dark blue banner features the text 'Welcome to Ubidots'. In the bottom left corner, the text 'Hey,' is followed by 'Welcome to Ubidots' on the next line. In the bottom right corner, there is a small inset image showing a mobile device displaying the Ubidots interface, which includes a map and a table of data.

ubidots

Devices ▾ Data ▾ Users ▾ Apps

30 days left on trial

Devices

Groups **NEW**

Types

Functions

Welcome to Ubidots

Hey,

Welcome to Ubidots

# 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](mailto:mzennaro@ictp.it)