

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.

Copинг 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

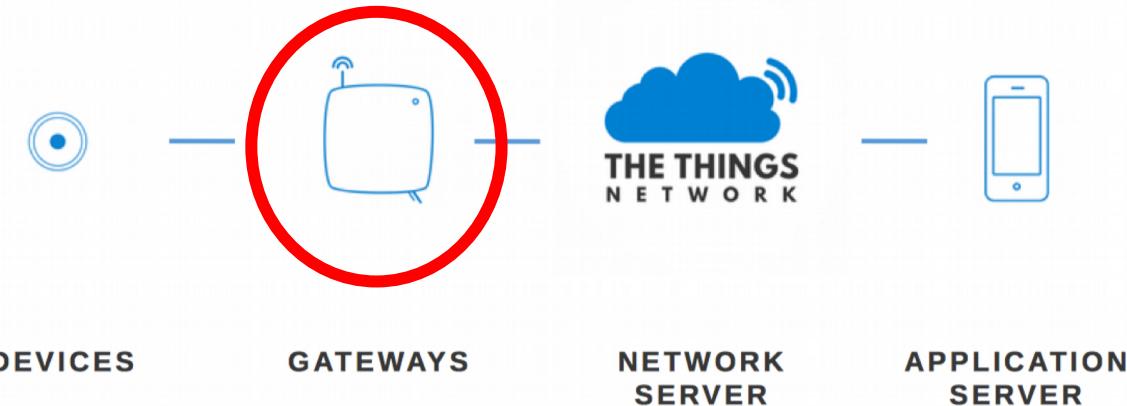
microUSB Cable

RPi with LoRa module



TTN: devices, gateways, servers

HOW DOES THIS WORK?



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.uptronics.com>



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 uptronics 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.

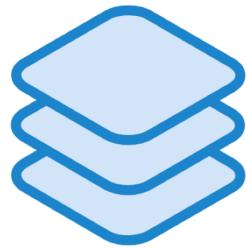
Create a new application in TTN.

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

ADD APPLICATION

Application ID
The unique identifier of your application

Description
A human readable description of your new application

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

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

Description

Handler (Europe)

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

ADD COLLABORATOR

Could not add application
An app with the application id test... is already registered.

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

REGISTER DEVICE

Device ID
This is the unique identifier for the device in the app. The device ID will be generated.

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 your device and the network.
 this field will be generated

App EUI
70 B3 D5 7E D0 01 70 74

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 your device and the network.

 this field will be generated

App EUI

70 B3 D5 7E D0 01 70 74



TTN: devices

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

Authentication

Never seen!

TTN: devices

Settings

A screenshot of the TTN Device Overview page. At the top right, there is a navigation bar with three tabs: "Overview" (highlighted in blue), "Data", and "Settings". A large red arrow points downwards towards the "Settings" tab. Below the navigation bar, the page title "DEVICE OVERVIEW" is displayed in blue capital letters. The main content area contains several device parameters:

- 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:** (redacted)
- 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 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



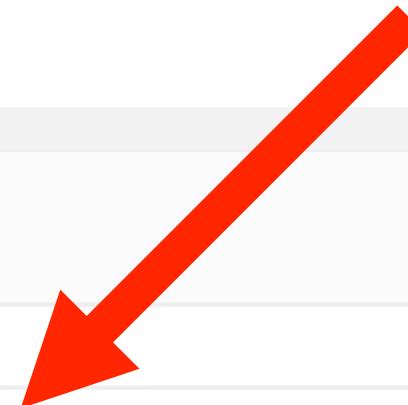
App Session Key will be generated

TTN: devices

DeviceAdd, NetKey, AppKey

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 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

The screenshot shows the TTN application overview page. At the top, there is a navigation bar with tabs: Overview (highlighted in blue), Devices, Payload Formats (with a large red arrow pointing to it), Integrations, Data, and Settings. Below the navigation bar, the section title "APPLICATION OVERVIEW" is displayed in blue. To the right of the title is a link to "documentation". Under the "APPLICATION OVERVIEW" section, there are several details: "Application ID" followed by a yellow button containing the text "test_application_fablab"; "Description" (empty); "Created" followed by the timestamp "30 minutes ago"; and "Handler" followed by the text "ttn-handler-eu (current handler)".

TTN: payload

PAYOUT 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.Led = bytes[0];  
7  
8     return decoded;  
9 }
```

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

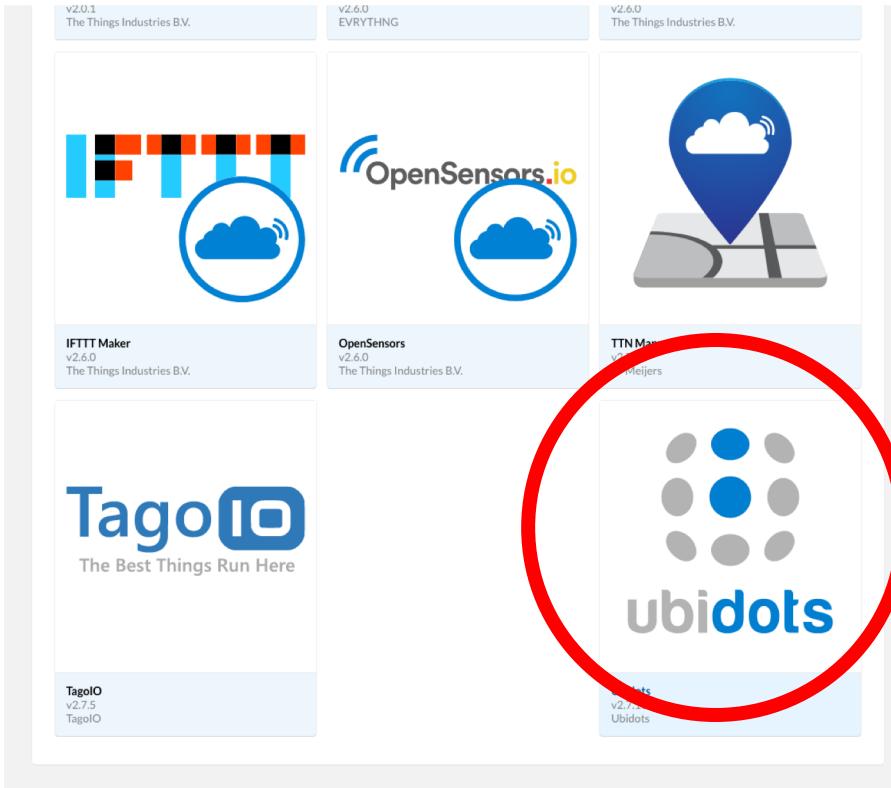
Integrations



The screenshot shows the TTN web interface with the following details:

- Header:** The word "Integrations" is displayed prominently at the top left.
- Navigation Bar:** A horizontal bar with tabs: Overview, Devices, Payload Formats, **Integrations** (which is highlighted in blue), Data, and Settings.
- Section Header:** "INTEGRATIONS" is displayed in bold blue text at the top of the main content area.
- Add Integration:** A button labeled "+ add integration" with a small icon.
- Content Area:** A message states "There are no integrations for application test_application_fablab." followed by a link "[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> 

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 shows the Ubidots dashboard with a dark blue header. The header includes the Ubidots logo, navigation links for Device Management, Users, Apps, and Reports, and a user profile icon labeled "iotexpo". Below the header, there is a section for "API Key" with a value "BBFF-2359d3981925dc4e19e0" and a "Copy" button. To the right of this is a "Tokens" section with a red oval highlighting the "Tokens" link. Below "Tokens" are two entries: "New Token" with value "BBFF-3onJzEyhDVH0JE0SF" and a "Copy" button, and "Ubidots App" with value "BBFF-j5CRjfftJQyD07E6X" and a "Copy" button. At the bottom right of this section is a "More" link. On the far right of the dashboard, there is a sidebar with links for "My Profile", "API Credentials" (which is also highlighted with a red oval), "How this works?", and "Log out".

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