

ESIB IoT Challenge

Welcome to the ESIB IoT Challenge. In this challenge, you will be designing and prototyping the first IoT services based on a LoRaWAN network.

-. Platform

During this challenge, you will benefit from the first experimental platform implementing an end-to-end LoRaWAN solution in Lebanon. The platform consists of the following elements:

- Devices that communicate to one or more gateways via a wireless interface using single hop LoRa and implementing the LoRaWAN protocol. These devices are physically connected to sensors that generate data.
- Gateways or base stations that forward frames between the devices and the network server. Gateways are connected to the network server via IP interfaces.
- A LoRaWAN backend that implements the network server functions and provides frame control and security.
- Applications that enable to visualize and store the sensor data obtained from the devices.

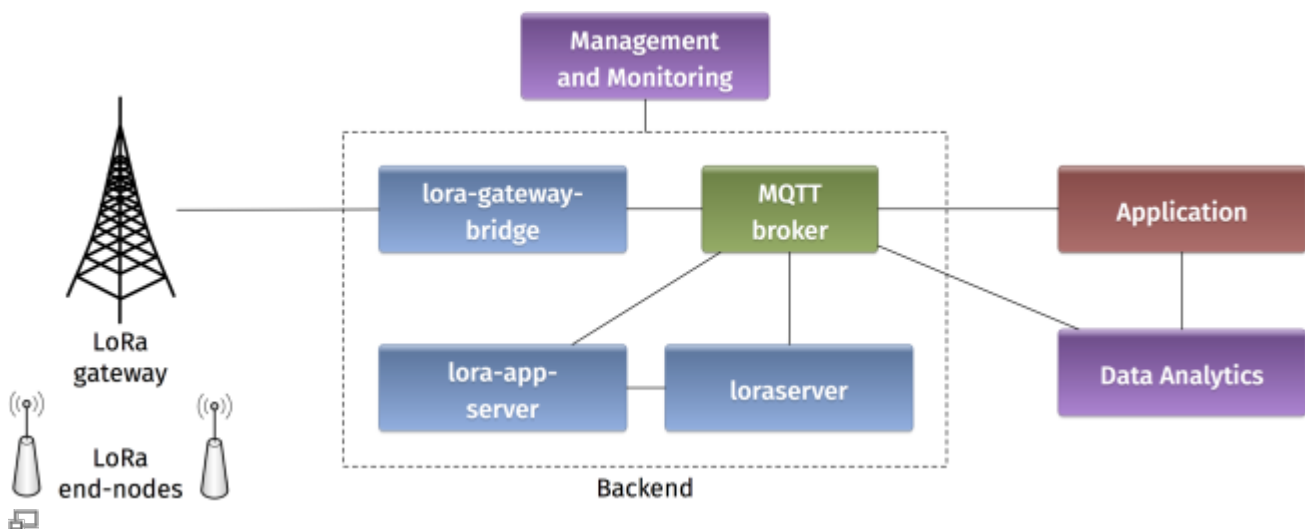


Figure 1. Architecture of the LoRaWAN Platform



- Where is the LoRa modulation implemented on the platform?
- What are the advantages of the LoRa modulation?
- How LoRa is compatible with LPWAN requirements and constraints?
- What is LoRaWAN? What is the difference between LoRaWAN and LoRa?
- Where is LoRaWAN implemented in the platform?
- Where does the IP layer start in the platform? Comment your answer considering the trends in IoT.
- Illustrate the protocol stacks on the LoRaWAN platform.

-. Backend

The Loraserver has a web interface for configuring the applications and devices on the platform.



Figure 5. Loraserver web interface

Start by choosing the application named NTRE - 1617. Then create a new node by providing the following information:

- A unique node name: NTRE - GX
- The node description
- A unique device EUI on 64 bits: Random identifiers can be generated on <https://www.random.org/bytes/>
- The application EUI on 64 bits: 0badde1cafe2deca.
- A unique application key on 128 bits also obtained by random generation.
- In advanced network settings, choose the receive window RX2.

Make sure that the ABP activation button is unchecked, in order to enable OTAA join method.



- What does the application EUI mean? How is it used in LoRaWAN?
- What does the application key mean? How is it used in LoRaWAN security?
- Compare the two join methods used in LoRaWAN by giving the advantages and inconvenients.
- What is the difference between the two receive windows in LoRaWAN? What are they used for?

-. Devices

Devices in the LoRaWAN platform are implemented on Arduino boards with Dragino shields. The combined module as well as the basic configuration steps are presented in [Simple Prototype of LoRa Communications](#).

Start by verifying the installation on your PC of the latest Arduino IDE and place the Arduino LMIC library in the corresponding folder.



- Give the characteristics of the Arduino you are using: number of pins, type of pins, memory sizes, etc.
- Give the main characteristics of the LoRa shield from Dragino (www.dragino.com).
- What type of Antenna are you using? Explain the corresponding characteristics.

Download the example sketch, open it with Arduino IDE and compile it.

You can download the following sketch

test-loraserver-comb-loraserver-dragino.zip

and modify it according to your preferences. Below you can find some commented extracts of the sketch.

Verify that you have the latest Arduino IDE from <https://www.arduino.cc/en/Main/Software> on your computer. Install the board files as noted in <http://support.sodaq.com/sodaq-one/autonomo/getting-started-autonomo/>. Add the following library sodaq_rn2483_2.zip to your Arduino IDE as explained in <https://www.arduino.cc/en/guide/libraries>.

In order to program the LoRaWAN devices, you should verify the installation on your PC of the following software:

- Arduino IDE
- LMIC Library

The pin mapping corresponds to the Dragino electronic schematic:

```
const lmic_pinmap lmic_pins = {  
  .nss = 10,  
  .rxtx = LMIC_UNUSED_PIN,  
  .rst = 9,  
  .dio = {2, 6, 7},  
};
```

The send function is rescheduled TX_INTERVAL seconds after each transmission complete event:

```
case EV_TXCOMPLETE:  
  Serial.println(F("EV_TXCOMPLETE (includes waiting for RX  
windows)"));  
  if(LMIC.dataLen) {  
    // data received in rx slot after tx  
    Serial.print(F("Data Received: "));  
    Serial.write(LMIC.frame+LMIC.dataBeg, LMIC.dataLen);  
    Serial.println();  
  }  
  // Schedule next transmission  
  os_setTimedCallback(&sendjob,  
os_getTime()+sec2osticks(TX_INTERVAL), do_send);  
  break;
```

The send function is initially scheduled here:

```
do_send(&sendjob);
```

The message containing the sensor values is transmitted on one of the radio channels:

```
LMIC_setTxData2(1, (uint8_t*) buffer, message.length(), 0);
```

-. Triggered Message Sending

You can also find another example of sketch to download:

`test-loraserver-moisture-on-move.ino.zip`

. Here the message sending is not periodic but related to an event. For example, an infrared sensor detects a movement and triggers a signal for the device to send a LoRaWAN message. Note also that the join method used in this second sketch is Activation by Personalisation (ABP): the device address, the network session key, and the application session key are directly configured on the device.



- OTAA
- ID
- Security

-. Applications

mqtt-spy is an open source utility intended to help you with monitoring activity on MQTT topics. It has been designed to deal with high volumes of messages, as well as occasional publications. mqtt-spy is a JavaFX application, so it should work on any operating system with an appropriate version of Java 8 installed. A very useful tutorial is available on <https://github.com/eclipse/paho.mqtt-spy/wiki>. You can use mqtt-spy to debug the messages received from the LoRaWAN devices. For this, you should download the software tool from <https://github.com/eclipse/paho.mqtt-spy/wiki>. After starting the application, configure a new connection to the MQTT broker by simply adding the IP address of the broker in the Server URI field. Now you can subscribe to any MQTT topic. If you want to receive all messages arriving at the backend, you can use the generic topic #. You can also limit to the topic including the messages of any specific device:

`application/APPLICATION_ID/node/DEVICE_EUI/rx`.

From:

<http://wiki.lahoud.fr/> - **wikiroute**

Permanent link:

http://wiki.lahoud.fr/doku.php?id=esib_iot_challenge&rev=1495029806

Last update: **2017/05/17 16:03**

