IoT-Leb Hackathon

- Curious to discover the emerging LoRaWAN technology and work on the award-winning scriptr IoT platform?
- Ready to build your first end-to-end IoT chain using cutting-edge technologies?
- Up to test your innovative ideas on a real-world IoT platform?

We invite you on March 12, 2019 to join the first IoT-Leb Hackathon along with Lebanon's most talented hackers, programmers, makers, and entrepreneurs. IoT-Leb Hackathon is part of IoT-Leb 19 conference that will take place at ESIB-USJ, CST Campus, Mar Roukos.

In the evolution towards the Internet of Things, an increasing number of devices equipped with sensors and communication interfaces will interact and collect data, communicate over the internet, and provide valuable monitoring and automation services. IoT-Leb Hackathon is a unique opportunity to get acquainted with the technologies that are set to play an essential role in the IoT landscape. During the Hackathon, you will get exclusive access to the latest IoT technologies.

-. Platform

During this lab, 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.

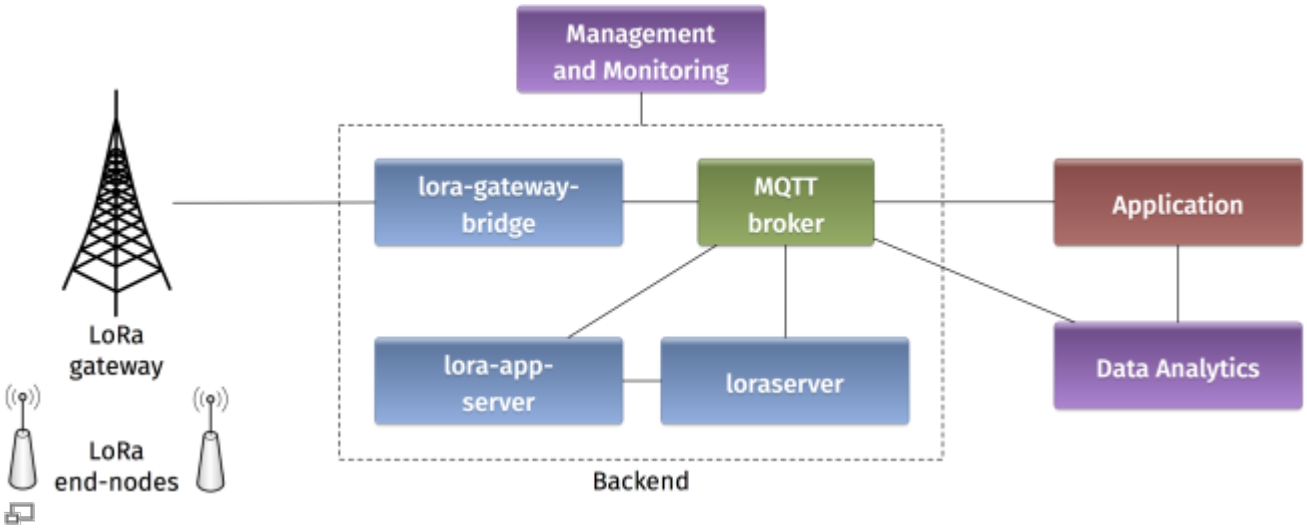


Figure 1. Architecture of the LoRaWAN Platform



Figure 2. Outdoor antenna at ESIB-USJ

-. Backend

In a LoRaWAN network, the devices communicate with a Network Server through the gateway. The backend installed in the platform is based on an open-source LoRaWAN network-server https://www.loraserver.io. A web interface is available for configuring the applications and devices on the platform (https://212.98.ZZ.ZZ:8080).

Start by choosing the application named `hackathon` to create a new device.

You should provide the following information for creating your device:

- A unique device name: `IoTXX` (where XX is your group number)
- The device description
- A unique device EUI on 64 bits
- A unique application key on 128 bits

Make sure to choose `lora-profile` as a `Device-profile` in order to enable OTAA join method.

-. Devices

Devices in the LoRaWAN platform are implemented on TTGO boards with ESP32 system on a chip microcontrollers with integrated Wi-Fi and dual-mode Bluetooth, and an additional LoRa transceiver.

Start by verifying the installation on your PC of the latest Arduino IDE and the presence of the LoRaWAN

LMIC library

. Then, download the example sketch from https://github.com/samerlahoud/iotleb-hacktools and open with Arduino IDE.

Now you should configure your device with the same identifiers `DEVEUI` and `APPKEY` as in the

backend:

```
// This EUI must be in little-endian format, so least-significant-byte
// first. When copying an EUI from loraserver, this means to reverse
// the bytes.
static const u1_t PROGMEM DEVEUI[8]={ 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00};   // LSB mode
void os_getDevEui (u1_t* buf) { memcpy_P(buf, DEVEUI, 8);}

// This key should be in big endian format (or, since it is not really a
// number but a block of memory, endianness does not really apply). In
// practice, a key taken from loraserver can be copied as-is.

static const u1_t PROGMEM APPKEY[16] = { 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00 }; // MSB mode
void os_getDevKey (u1_t* buf) { memcpy_P(buf, APPKEY, 16);}
```



Note that the device EUI should be in little endian format. Hence, the key, `0badde1cafe2deca` should be written as `0xCA, 0xDE, 0xE2, 0xAF, 0x1C, 0xDE, 0xAD, 0x0B` in the Arduino sketch.

The LMIC library (see

documentation

) defines a set of events corresponding to the protocol machine state. These events appear in the `onEvent()` function. For instance, the `EV_TXCOMPLETE` event is called after the completion of a LoRa transmission.

The `do_send` function enables to transmit data over LoRaWAN. After each transmission (`EV_TXCOMPLETE` event) the next one is scheduled in a way to obtain periodic messaging. This period is defined by `TX_INTERVAL` and is subject to duty cycle limitations.

Now you are ready to compile the sketch and upload it to the LoRaWAN device. Connect the device to a USB port on your PC, choose the board type (`TTGO LoRa32` or `T-Beam` from the `ESP32 Arduino` category) and select the corresponding port. Compile and upload!

Open the serial monitor in the Arduino IDE at 115200 baud and analyse the debug messages.

Getting back to the backend, you can monitor some important information related to your device such as the activation status, frame counters, and live data.

-. Capturing Received Messages

mqtt.fx is a 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.

You can use mqtt.fx to debug the messages received from the LoRaWAN devices. After starting the

application, configure a new connection to the MQTT broker by simply adding the IP address of the broker and the user credentials. 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/16/device/DEV_EUI/rx`.

The payload received by the MQTT client is decrypted but encoded in Base64. On the loraserver, we have activated a script that decodes the message before delivery.

-. Sending Downlink Data

If you need to send data to your device, you should publish an encoded message in the corresponding topic `application/16/device/DEV_EUI/tx` as follows:

```
{
    "confirmed": false,                          // whether the payload must
be sent as confirmed data down or not
    "fPort": 10,                                // FPort to use (must be > 0)
    "data": "...."                              // base64 encoded data
(plaintext, will be encrypted by LoRa Server)
}
```



The payload sent by the MQTT client must be encoded in Base64 (you can use for example the online tool at https://www.base64encode.org).

From:
http://wiki.lahoud.fr/ - **wikiroute**

Permanent link:
http://wiki.lahoud.fr/doku.php?id=iot_leb_hackathon&rev=1552295687

Last update: **2019/03/11 10:14**