Deploying an End-to-End LoRaWAN Platform

Starting from September 2016, Saint-Joseph University of Beirut (USJ) will be deploying the first academic LoRa network in Lebanon. The network will support monitoring of micro-climate conditions in vineyards. Here below you can find a detailed description of the experimental platform implementing an end-to-end LoRaWAN solution.

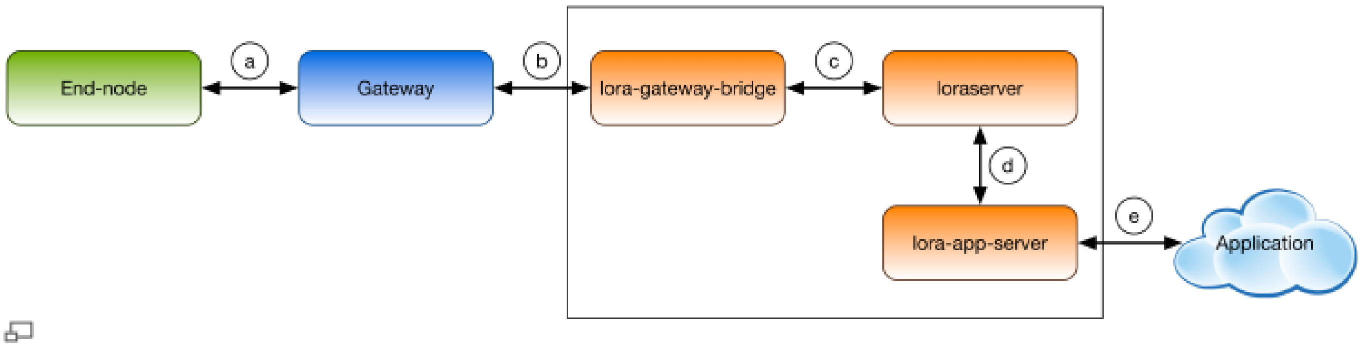


Figure 2. Architecture of the LoRaWAN Platform

-. Devices

-. Autonomo with LoRaBee

For the devices in the LoRaWAN platform, we will use an Autonomo board with a LoRaBee holding the Microchip RN2483 module. According to http://shop.sodaq.com, Autonomo is a matchbox-sized powerhouse which uses the new Atmel Cortex M0+ 32bit micro controller. One advantage of such device is that it can be powered by a smartphone-sized solar panel.

In order to configure the device, you need to install the Arduino IDE from https://www.arduino.cc/en/Main/Software. Then you need to to install the board files as noted in http://support.sodaq.com/sodaq-one/autonomo/getting-started-autonomo/.

sodaq_rn2483_2.zip

-. Arduino with Dragino Shield

-. Gateways

-. Single Channel Gateway

The single channel gateway includes a LoRa transmission module (Dragino Shield) connected to a Raspberry Pi (2 or 3) as shown in Figure 1. Communication between the two modules is done over an SPI interface.

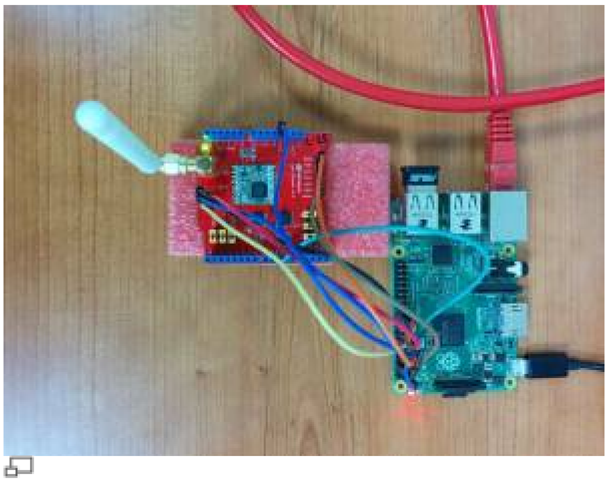


Figure 2. LoRa single channel gateway

In order to assemble the gateway, start by making the wire connections: the connection pins are identified in Figures 2 and 3.

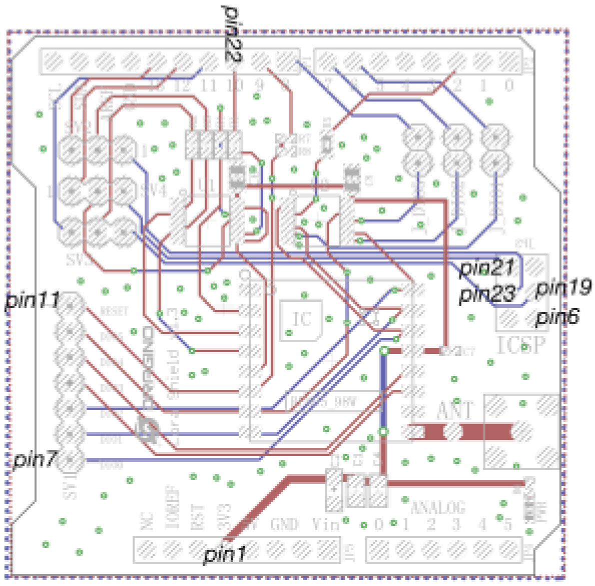
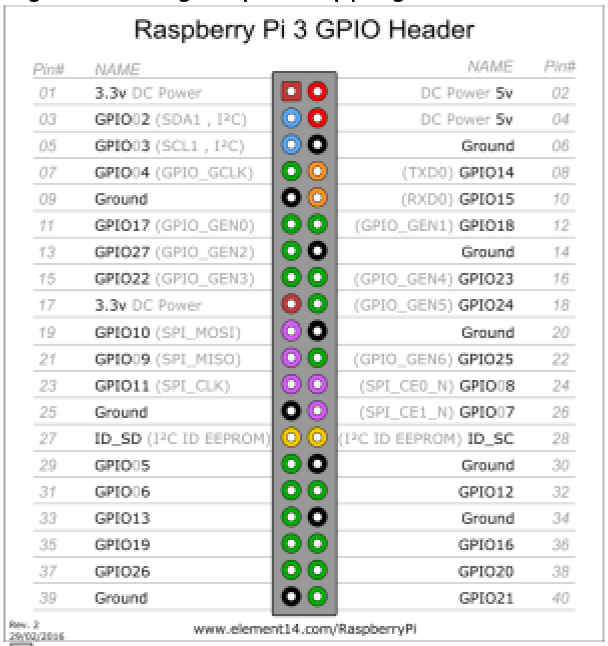


Figure 2. Dragino pin mapping



Raspberry Pi 3 GPIO Header

Pin#	NAME	NAME	Pin#
01	3.3v DC Power	DC Power 5v	02
03	GPIO02 (SDA1 , I²C)	DC Power 5v	04
05	GPIO03 (SCL1 , I²C)	Ground	06
07	GPIO04 (GPIO_GCLK)	(TXD0) GPIO14	08
09	Ground	(RXD0) GPIO15	10
11	GPIO17 (GPIO_GEN0)	(GPIO_GEN1) GPIO18	12
13	GPIO27 (GPIO_GEN2)	Ground	14
15	GPIO22 (GPIO_GEN3)	(GPIO_GEN4) GPIO23	16
17	3.3v DC Power	(GPIO_GEN5) GPIO24	18
19	GPIO10 (SPI_MOSI)	Ground	20
21	GPIO09 (SPI_MISO)	(GPIO_GEN6) GPIO25	22
23	GPIO11 (SPI_CLK)	(SPI_CE0_N) GPIO08	24
25	Ground	(SPI_CE1_N) GPIO07	26
27	ID_SD (I²C ID EEPROM)	(I²C ID EEPROM) ID_SC	28
29	GPIO05	Ground	30
31	GPIO06	GPIO12	32
33	GPIO13	Ground	34
35	GPIO19	GPIO16	36
37	GPIO26	GPIO20	38
39	Ground	GPIO21	40

Rev. 2
29/02/2016

www.element14.com/RaspberryPi

Figure 3. Raspberry pi 3 pins

Connect the Raspberry Pi to the Internet and install the packet forwarding software. The source code of the single channel packet forwarder is available on: https://github.com/samerlahoud/single_chan_pkt_fwd. In order to install it, you need to:

- Enable SPI on the Raspberry Pi using raspi-config
- Download and unzip the source code:

```
wget https://github.com/hallard/single_chan_pkt_fwd/archive/master.zip
unzip master.zip
```

- Install the wiring library:

```
apt-get update
apt-get install wiring
```

Compile the packet forwarder:

```
make all
```

For gcc version 4.6.3, a compilation error results in the following warning `unrecognized command line option '-std=c++11'`. Replace `-std=c++11` by `-std=c++0x` in the Makefile and recompile:

```
CFLAGS = -std=c++0x -c -Wall -I include/
```

Now, you need to configure the single channel packet forwarder. This is done in the `global_conf.json` configuration file. Particularly, you need to choose the channel, the spreading factor, the pins for SPI communication, and the address of the backend server. Note that you can specify multiple backends for testing purposes.

global_config.json

```
{
  "SX127x_conf":
  {
    "freq": 868100000,
    "spread_factor": 7,
    "pin_nss": 6,
    "pin_dio0": 7,
    "pin_rst": 0,
    "pin_led1":4
  },
  "gateway_conf":
  {
    "ref_latitude": 33.86576536772,
    "ref_longitude": 35.56378662935,
    "ref_altitude": 165,

    "name": "ESIB SC Gateway",
    "email": "cimti@usj.edu.lb",
    "desc": "Dragino Single Channel Gateway on RPI",

    "servers":
    [
      {
        "address": "router.eu.thethings.network",
        "port": 1700,
        "enabled": true
      },
      {
        "address": "212.98.137.194",
        "port": 1700,
        "enabled": true
      },
      {
        "address": "172.17.17.129",
        "port": 1700,
        "enabled": false
      }
    ]
  }
}
```

Finally, you can run the packet forwarder as root!

```
nohup ./single_chan_pkt_fwd &
```

-. Kerlink IoT Station

```
# activates eth0 at startup
ETHERNET=yes
# claims dhcp request on eth0
ETHDHCP=yes

# Selector operator APN
GPRSAPN=gprs.touch.com.lb
# Enter pin code if activated
GPRSPIN=0000
# Update /etc/resolv.conf to get dns facilities
GPRSDNS=yes
# PAP authentication
GPRSUSER=
GPRSPASSWORD=

# Bearers priority order
#BEARERS_PRIORITY="eth0,ppp0,eth1"
BEARERS_PRIORITY="ppp0,eth0,eth1"
```

```
./gps-pkt-fwd.sh > /dev/null &
```

```
 3270 root      2548 S    /bin/sh ./gps-pkt-fwd.sh
 3288 root     34908 S    ./gps_pkt_fwd
```

```
/etc/init.d/gprs start

[root@Wirgrid_0b03008c demo_gps_loramote]# /etc/init.d/gprs  status
pppd (pid 5273) is running...
Session: Rx=58, Tx=163
Globals: Rx=1130457, Tx=1195592
Sum:     Rx=1130515, Tx=1195755
[root@Wirgrid_0b03008c demo_gps_loramote]#
```

-. Backend

-. Loraserver

-. The Things Network

-. Applications

-. MQTT spy

-. Emoncms