Low-cost LoRa gateway: A step-by-step tutorial

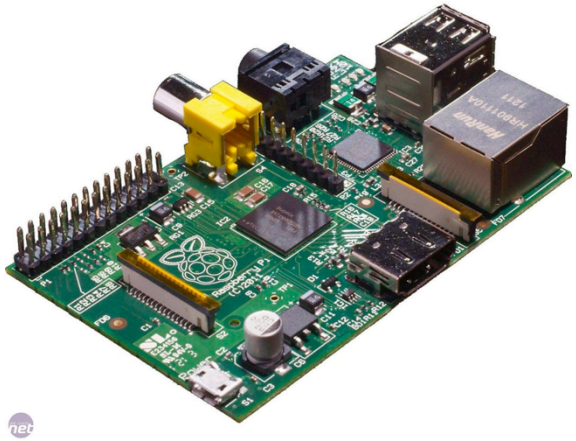
Prof. Congduc Pham
http://www.univ-pau.fr/~cpham
Université de Pau, France

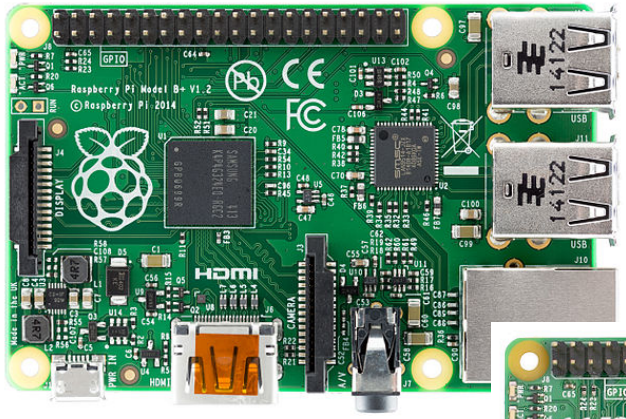
Contents

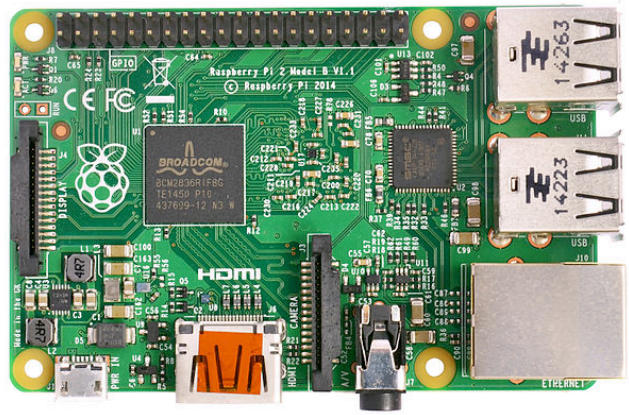
- We will show how to build a low-cost LoRa gateway to collect data from end-devices
- The device part will be shown in a separate tutorial
- The hardware platform is a Raspberry PI. RPI v1&2 have been successfully tested
- Let's get started…

Assembling the hardware

Get the Raspberry

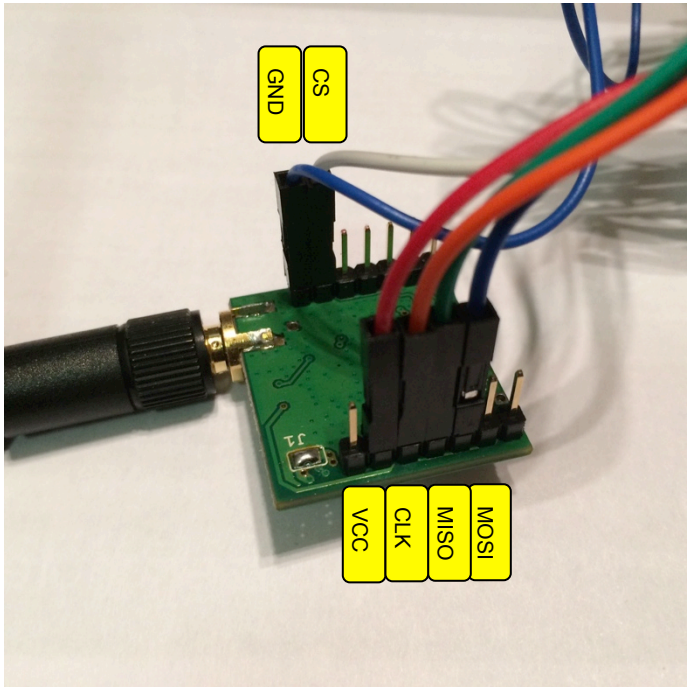


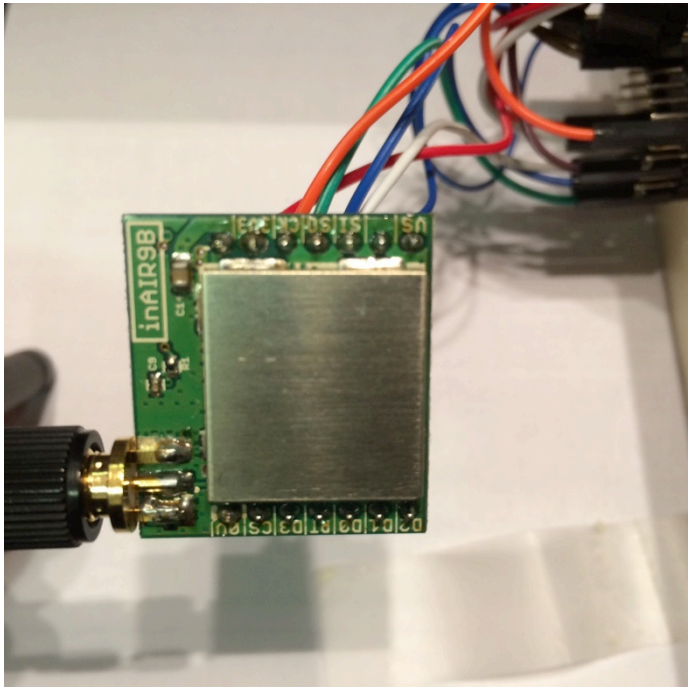




You can use Raspberry 1 model B or B+ or Raspberry 2 model B. The most important usefull feature is the Ethernet interface for easy Internet connection. You can use WiFi to get Internet connection by adding a WiFi USB dongle.

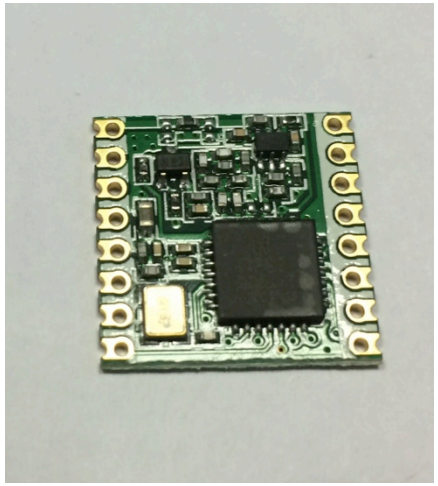
NOW THE RADIO MODULE (1)

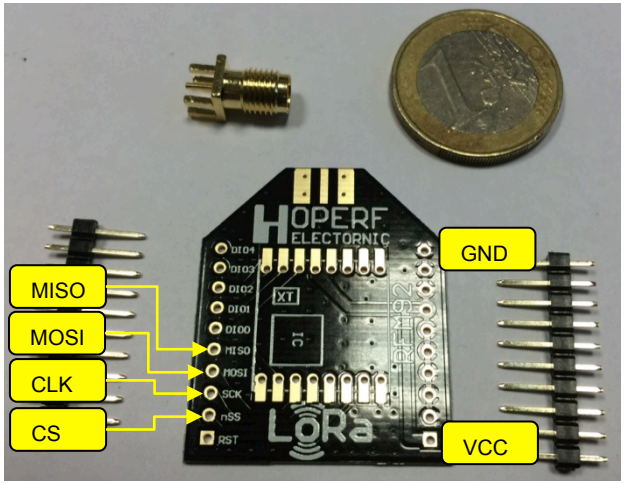




If you go for the inAir9 from Modtronix, then the header pins can come fully assembled. Take the 6mm header pins to have enough length to connect F/F breadboard cables (left). Connect the SPI pins with the F/F cables. Try to use different colors. I use the following colors: MOSI (blue), MISO (green), CS (white), CLK (orange). Then connect also the VCC (red) and the GND (black or any other dark color) of the radio board.

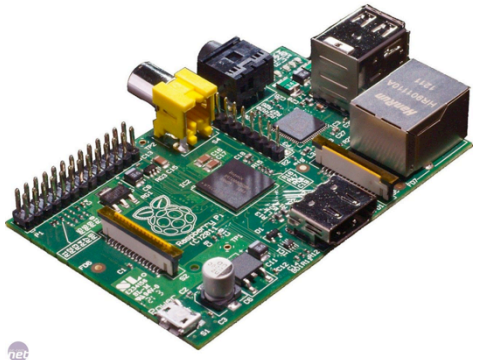
NOW THE RADIO MODULE (2)

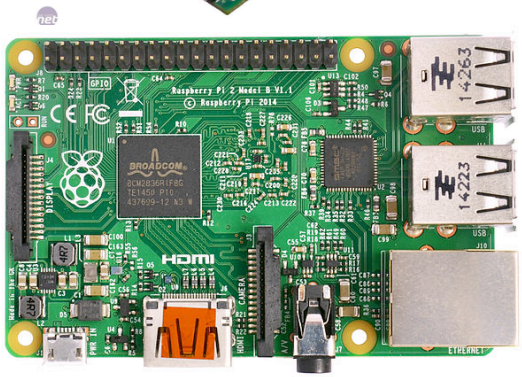


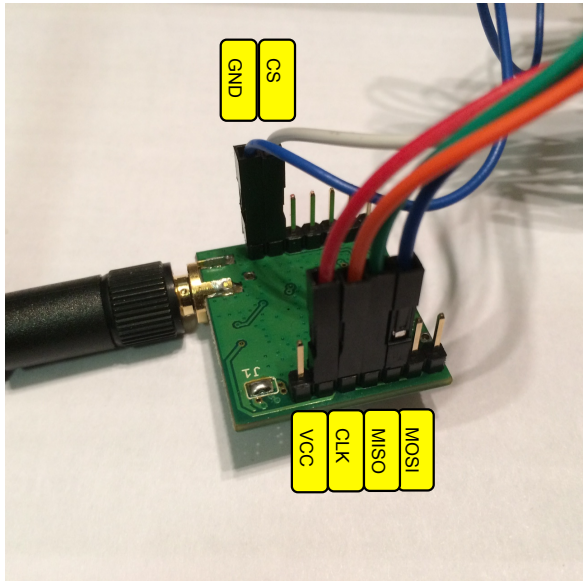


If you take the HopeRF RFM 92W/95W you will need the adaptor breakout and you have to go though some delicate but simple soldering tasks! It is not difficult but you have to trained a bit before! Then, like for the inAir9, use F/F breadboard cable to connect the SPI pins, using different colors as explained previously.

Connecting the radio module (1)



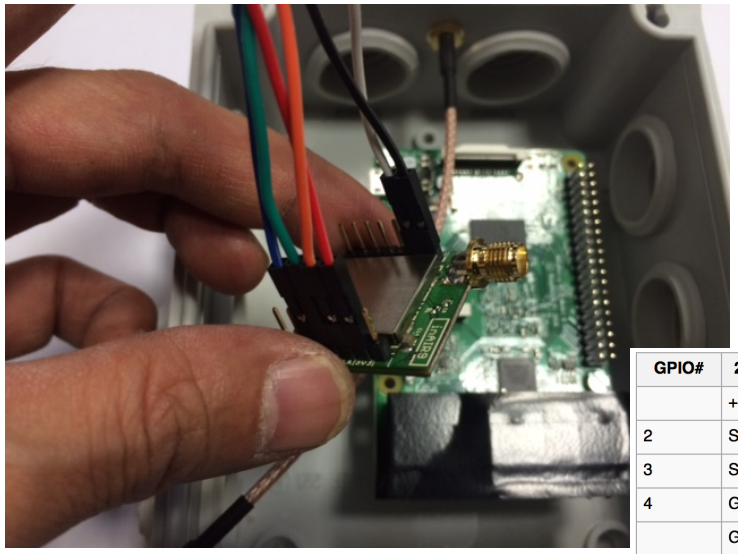


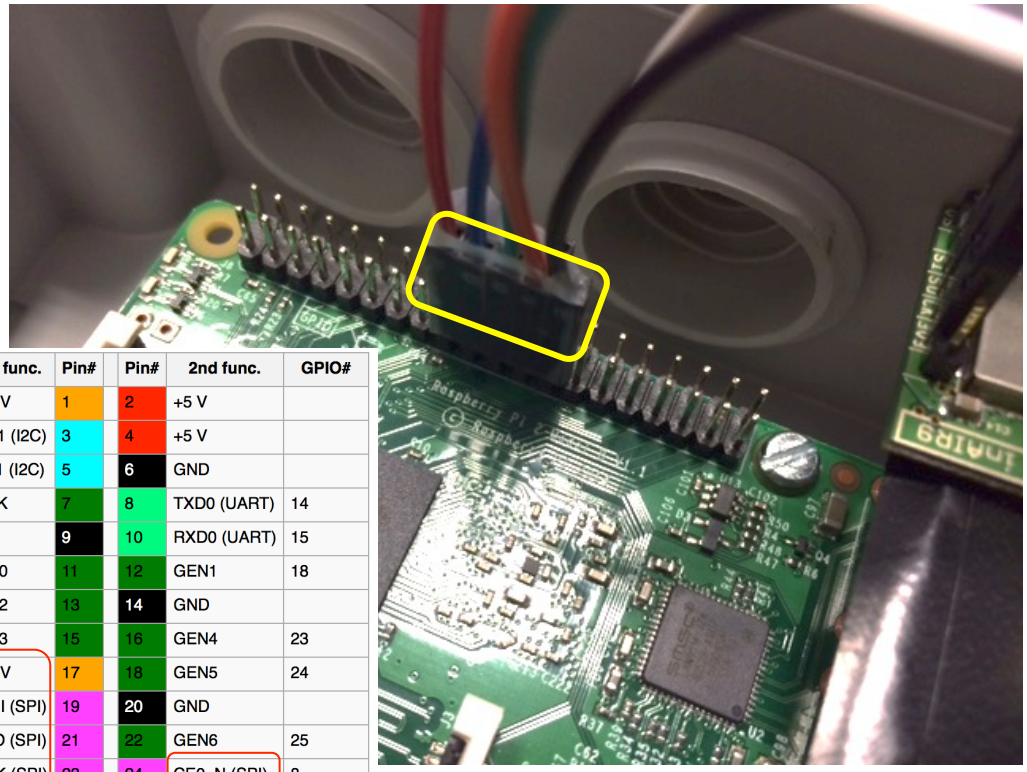


GPIO#	2nd func.	Pin#	Pin#	2nd func.	GPIO#
	+3.3 V	1	2	+5 V	
2	SDA1 (I2C)	3	4	+5 V	
3	SCL1 (I2C)	5	6	GND	
4	GCLK	7	8	TXD0 (UART)	14
	GND	9	10	RXD0 (UART)	15
17	GEN0	11	12	GEN1	18
27	GEN2	13	14	GND	
22	GEN3	15	16	GEN4	23
	+3.3 V	17	18	GEN5	24
10	MOSI (SPI)	19	20	GND	
9	MISO (SPI)	21	22	GEN6	25
11	SCLK (SPI)	23	24	CE0_N (SPI)	8
	GND	25	26	CE1_N (SPI)	7
(RPi 1 Models A and B stop here)					
EEPROM	ID_SD	27	28	ID_SC	EEPROM
5	N/A	29	30	GND	
6	N/A	31	32		12
13	N/A	33	34	GND	
19	N/A	35	36	N/A	16
26	N/A	37	38	Digital IN	20
	GND	39	40	Digital OUT	21

Depending on the model, you can have the « short » or the « long » GPIO interface. However, the SPI pins are at the same location therefore it does not change the way you connect the radio module if you take pin 1 as the reference. Connect the SPI pins (MOSI, MISO, CLK, CS) of the radio to the corresponding pins on the RPI. Note that CS goes to CE0_N on the RPI.

Connecting the radio module (2)



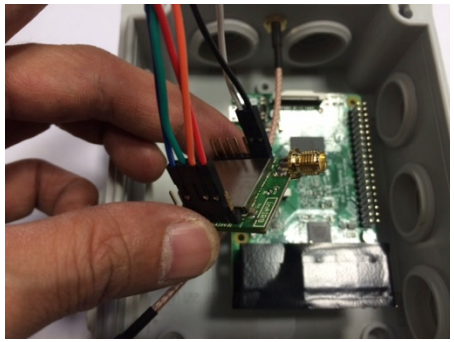


GPIO#	2nd func.	Pin#	Pin#	2nd func.	GPIO#
	+3.3 V	1	2	+5 V	
2	SDA1 (I2C)	3	4	+5 V	
3	SCL1 (I2C)	5	6	GND	
4	GCLK	7	8	TXD0 (UART)	14
	GND	9	10	RXD0 (UART)	15
17	GEN0	11	12	GEN1	18
27	GEN2	13	14	GND	
22	GEN3	15	16	GEN4	23
	+3.3 V	17	18	GEN5	24
10	MOSI (SPI)	19	20	GND	
9	MISO (SPI)	21	22	GEN6	25
11	SCLK (SPI)	23	24	CE0_N (SPI)	8
	GND	25	26	CE1_N (SPI)	7
(RPi 1 Models A and B stop here)					
EEPROM	ID_SD	27	28	ID_SC	EEPROM
5	N/A	29	30	GND	
6	N/A	31	32		12
13	N/A	33	34	GND	
19	N/A	35	36	N/A	16
26	N/A	37	38	Digital IN	20
	GND	39	40	Digital OUT	21

PUT IT IN A BOX

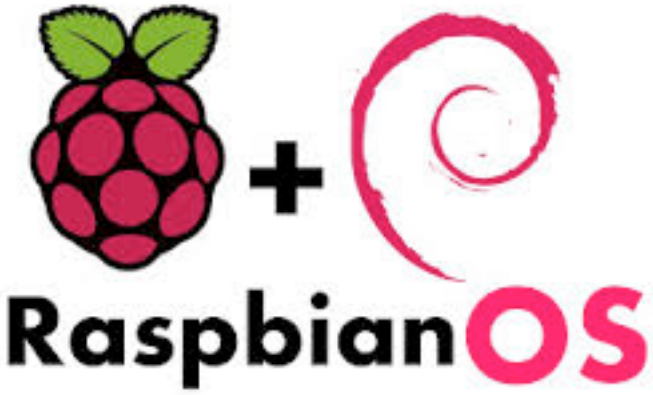




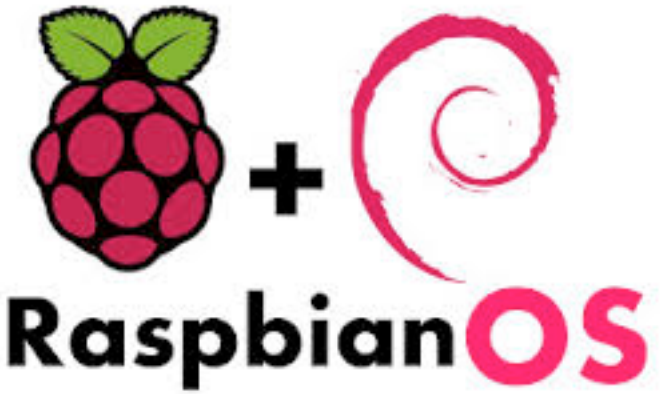



You can have a more integrated version, with a box for outdoor usage and PoE splitter to power the Raspberry with the Ethernet cable. See how we also use a DC-DC converter to get the 5V for the RPI.

Getting, compiling & installing the software

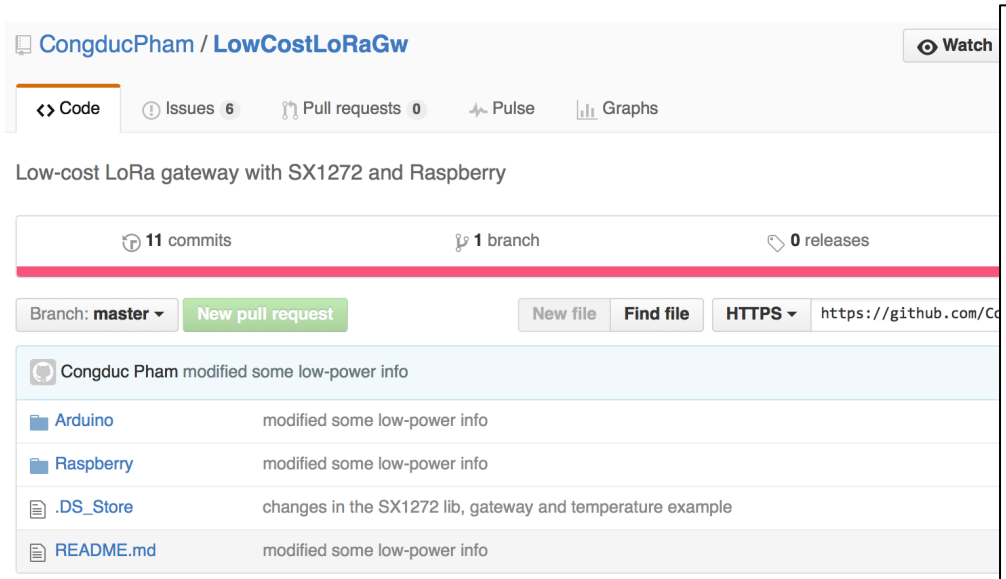


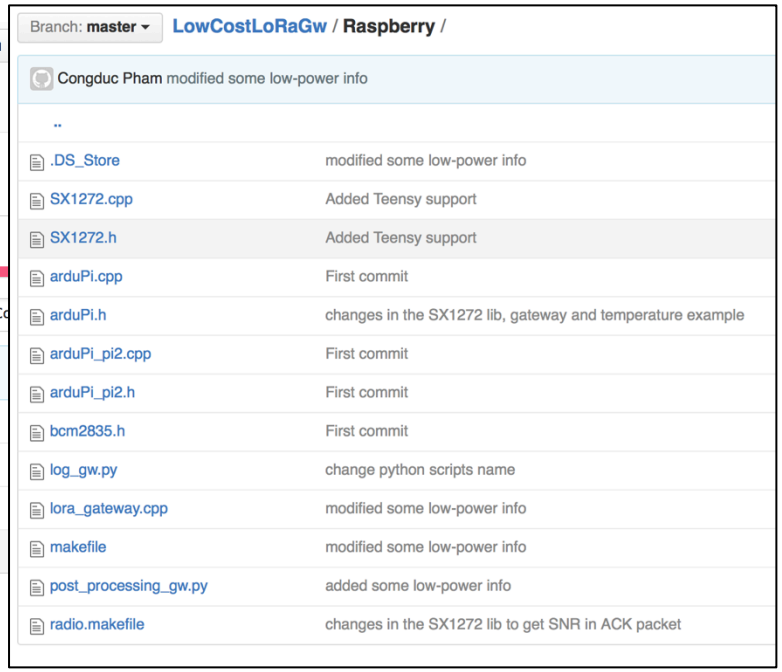
Installing the OS



We use the Raspbian OS. Install it on an SD card. There are many tutorials on the Internet for such procedure. Alternatively, we can provide the full image to burn on the SD card. It's 8GB!

Getting the LoRa specific GW software






```
> mkdir lora_gateway
> git clone https://github.com/CongducPham/LowCostLoRaGw.git
> cp LowCostLoRaGw/Raspberry/* lora_gateway/
```

Log in the RPI (ssh) and create a directory called lora_gateway. Get the LoRa RPI library from our github: https://github.com/CongducPham/LowCostLoRaGw (right) then copy all the files of the github's Raspberry folder into the lora_gateway folder.

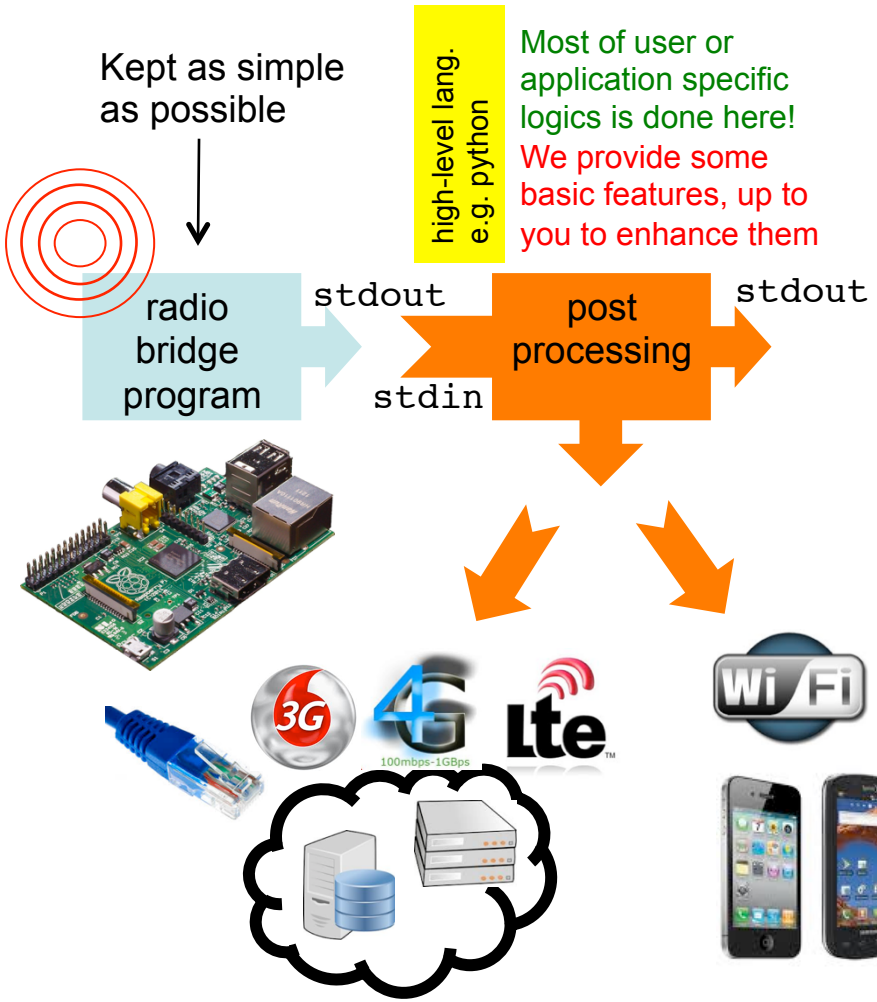
Compiling the GW software

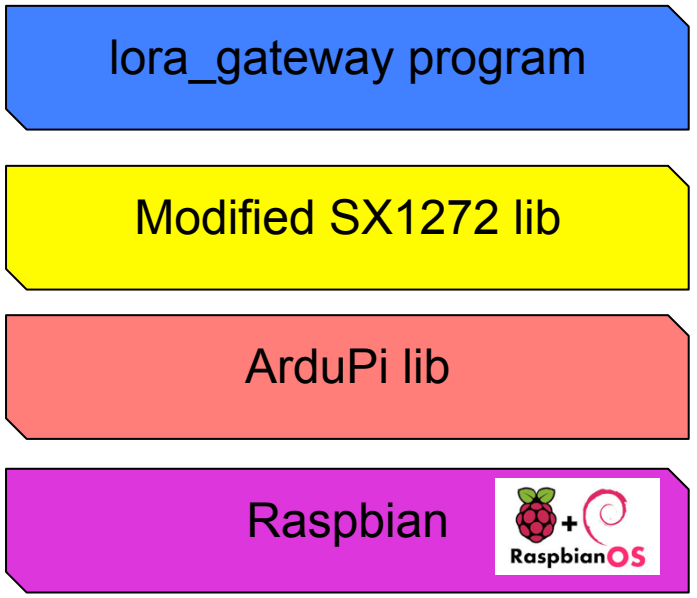
```
> cd lora_gateway
> make lora_gateway
g++ -DRASPBERRY -DIS_RCV_GATEWAY -c lora_gateway.cpp -o lora_gateway.o
g++ -c arduPi.cpp -o arduPi.o
g++ -c SX1272.cpp -o SX1272.o
g++ -lrt -lpthread lora_gateway.o arduPi.o SX1272.o -o lora_gateway
```

If you have a RPI 2, then type:

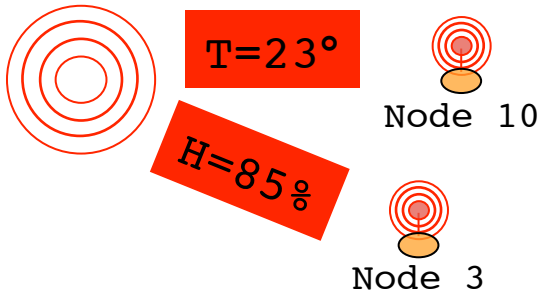
> make lora_gateway_pi2

Our Low-Cost gateway architecture



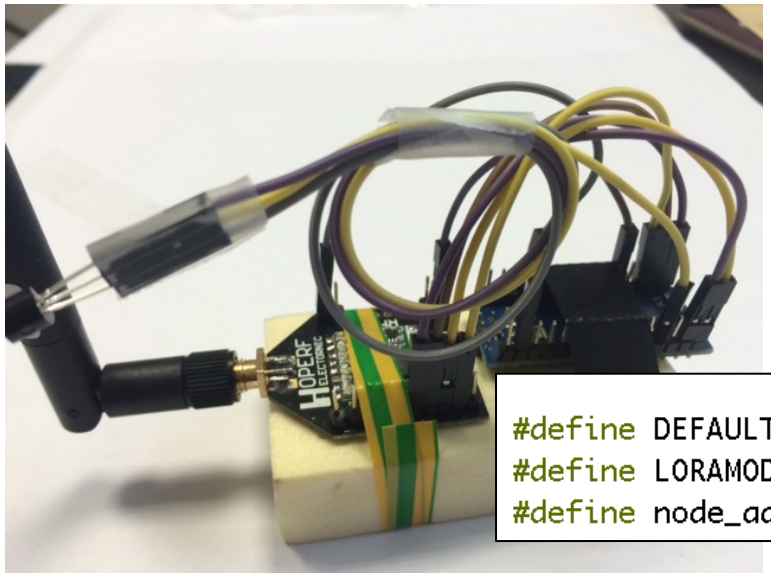


STARTING THE BASIC GATEWAY




```
> sudo ./lora_gateway
Power ON: state 0
LoRa mode: 4
Setting mode: state 0
Channel CH_10_868: state 0
Power M: state 0
Get Preamble Length: state 0
Preamble Length: 8
LoRa addr 1 : state 0
SX1272/76 configured as LR-BS. Waiting RF input for transparent RF-serial bridge

--- rxlora. dst=1 type=0x10 src=10 seq=0 len=5 SNR=9 RSSIpkt=-54
^p1,16,10,0,5,9,-54
T=23°
--- rxlora. dst=1 type=0x10 src=3 seq=0 len=5 SNR=8 RSSIpkt=-54
^p1,16,3,0,5,8,-54
H=85%
```

DEFAULT CONFIGURATION



\!##18.5

```
#define DEFAULT_DEST_ADDR 1
#define LORAMODE  4
#define node_addr 6
```

Device: the default configuration in the Arduino_LoRa_temp example is:
- Send packets to the gateway (one or many if in range)
- LoRa mode 4
- Node short address is 6

Gateway: the default configuration of the gateway is:
- LoRa mode 4
- Node short address is 1

Post-processing received data

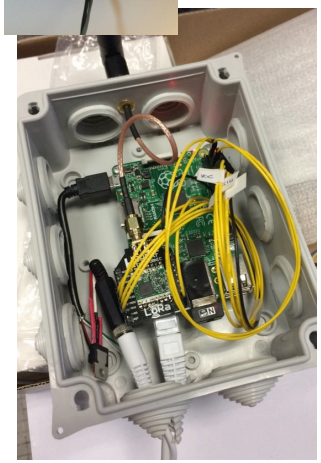
```
> sudo ./lora_gateway | python ./post_processing_gw.py
Power ON: state 0
LoRa mode: 4
Setting mode: state 0
Channel CH_10_868: state 0
Power M: state 0
Get Preamble Length: state 0
Preamble Length: 8
LoRa addr 1 : state 0
SX1272/76 configured as LR-BS. Waiting RF input for transparent RF-serial bridge
--- rxlora. dst=1 type=0x10 src=10 seq=0 len=5 SNR=9 RSSIpkt=-54
^p1,16,10,0,5,9,-54
Rcv ctrl packet info 1,16,10,0,5,9,-54
(dst=1 type=0x10 src=10 seq=0 len=5 SNR=9 RSSI=-5
T=23°
--- rxlora. dst=1 type=0x10 src=3 seq=0 len=5 SNR
^p1,16,3,0,5,8,-54
Rcv ctrl packet info 1,16,3,0,5,8,-54
(dst=1 type=0x10 src=3 seq=0 len=5 SNR=8 RSSI=-54)
H=85%
```

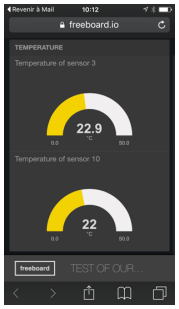
All lines that are not prefixed by specific character sequence are displayed unchanged

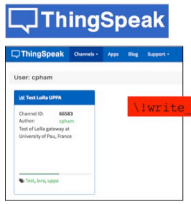
^p provides information on the last received packet: dst, type, src, seq, len, SNR & RSSI

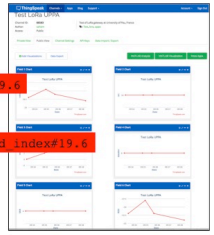
Pre-defined sequences inserted by the gateway or the end-device allow for information exchanged between the gateway and the post-processing program

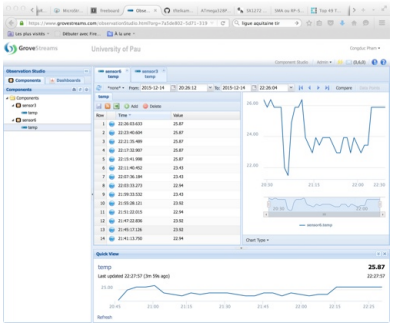
Gateway to cloud





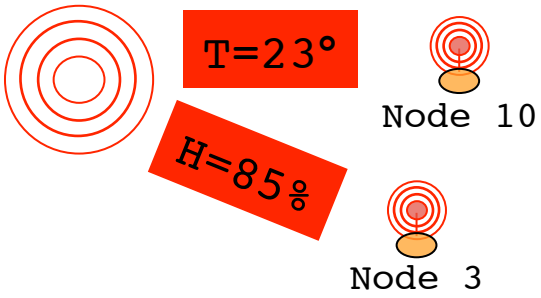






Data received at the gateway can be pushed to IoT clouds. We provide python script examples for many IoT cloud platforms. Most of clouds with REST API can be easily integrated.

LOG RECEIVED MESSAGES USING CLOUD SERVICES



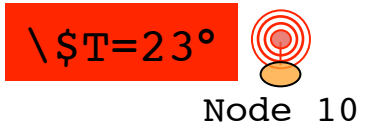

```
> sudo ./lora_gateway | python ./post_processing_gw.py
Power ON: state 0
LoRa mode: 4
Setting mode: state 0
Channel CH_10_868: state 0
Power M: state 0
Get Preamble Length: state 0
Preamble Length: 8
LoRa addr 1 : state 0
SX1272/76 configured as LR-BS. Waiting RF input for transparent RF-serial bridge

--- rxlora. dst=1 type=0x10 src=10 seq=0 len=5 SNR=9 RSSIpkt=-54
Rcv ctrl packet info 1,16,10,0,5,9,-54
(dst=1 type=0x10 src=10 seq=0 len=5 SNR=9 RSSI=-54)
rcv msg to log (\$) on dropbox : T=23°
--- rxlora. dst=1 type=0x10 src=3 seq=0 len=5 SNR=8 RSSIpkt=-54
Rcv ctrl packet info 1,16,3,0,5,8,-54
(dst=1 type=0x10 src=3 seq=0 len=5 SNR=8 RSSI=-54)
rcv msg to log (\&) on firebase : H=85%
```

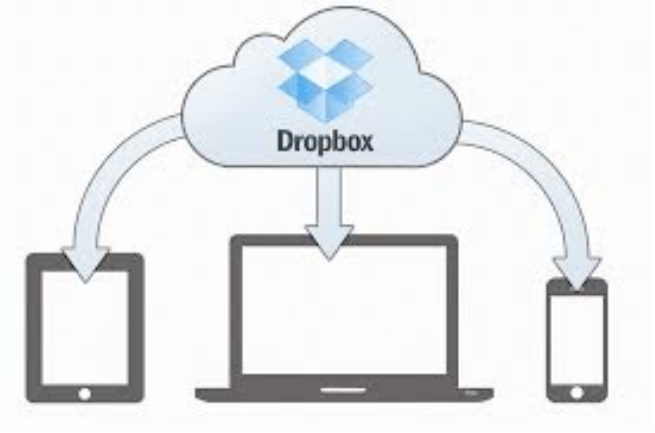
\$ or \& before the data indicates that the data should be logged on a file or server. It is up to the end-device to decide which option

Using Dropbox

- A message starting with '\$' is logged in a file 'telemetry.log' in a folder shared through Dropbox

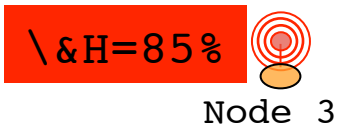
```
(src=10 seq=0 len=5 SNR=9 RSSI=-54) 2015-11-04T10:14:30.328413> T=23°
(src=10 seq=1 len=7 SNR=8 RSSI=-54) 2015-11-04T10:14:37.443350> T=23.2°
(src=10 seq=2 len=5 SNR=8 RSSI=-53) 2015-11-04T10:16:23.343657> T=24°
...
```

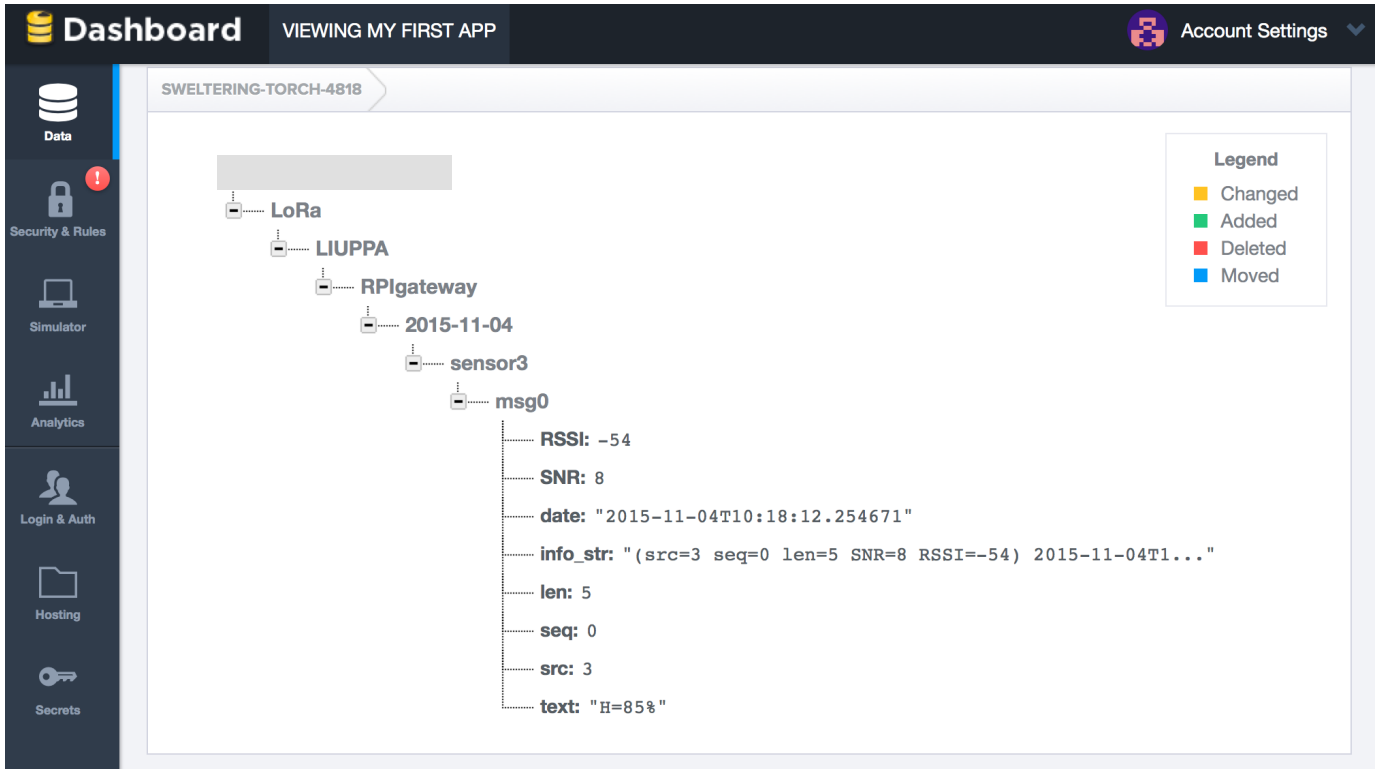




USING Firebase

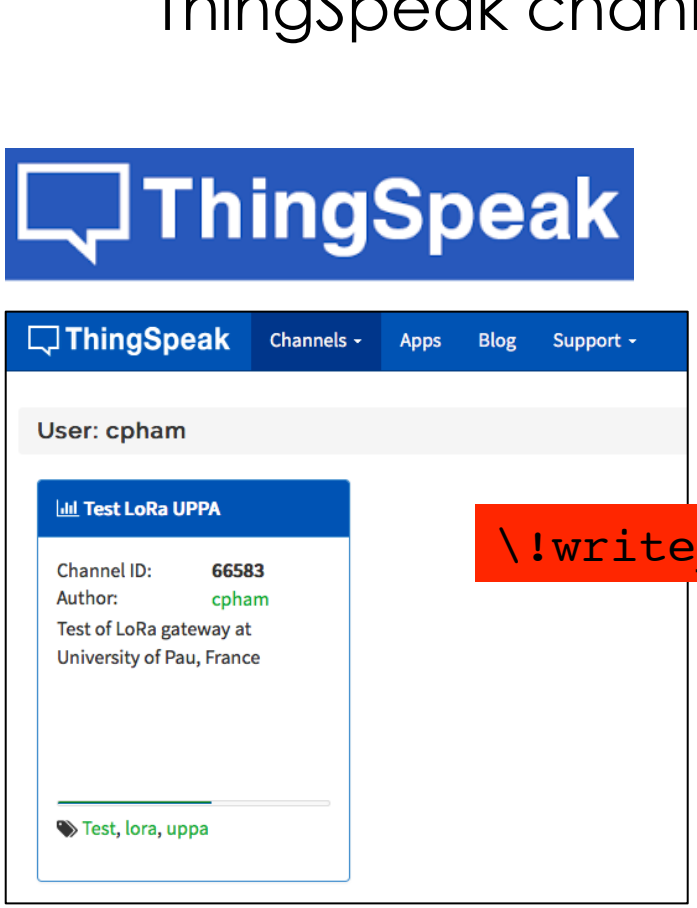
- A message starting with '\&' is logged in a Firebase database





USING ThingSpeak

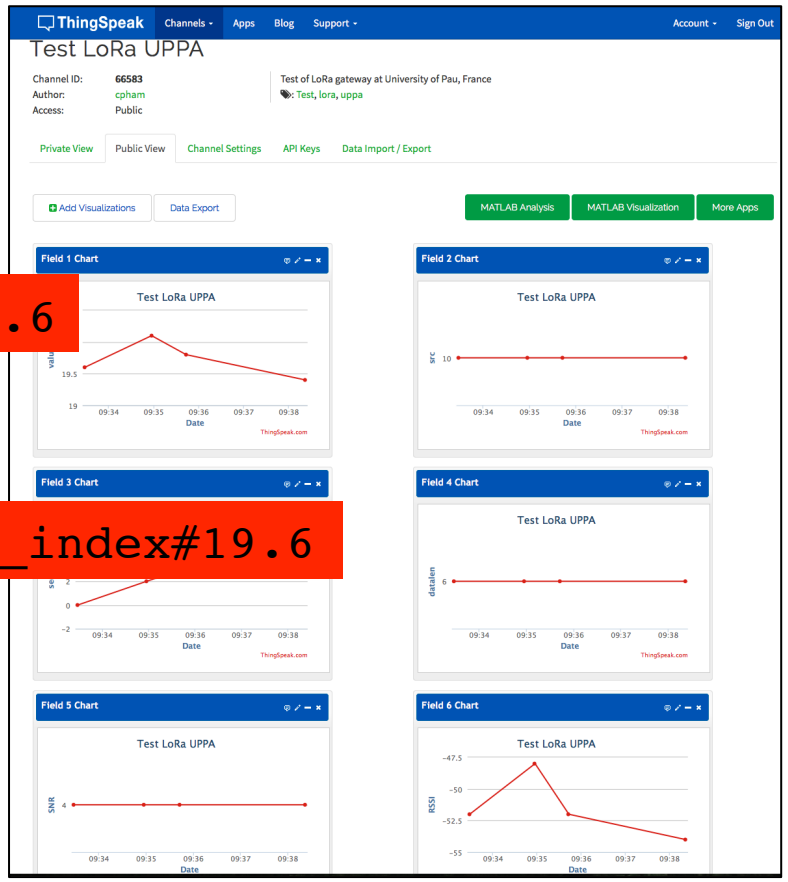
- A message starting with '\!' is logged in a ThingSpeak channel



\!##19.6

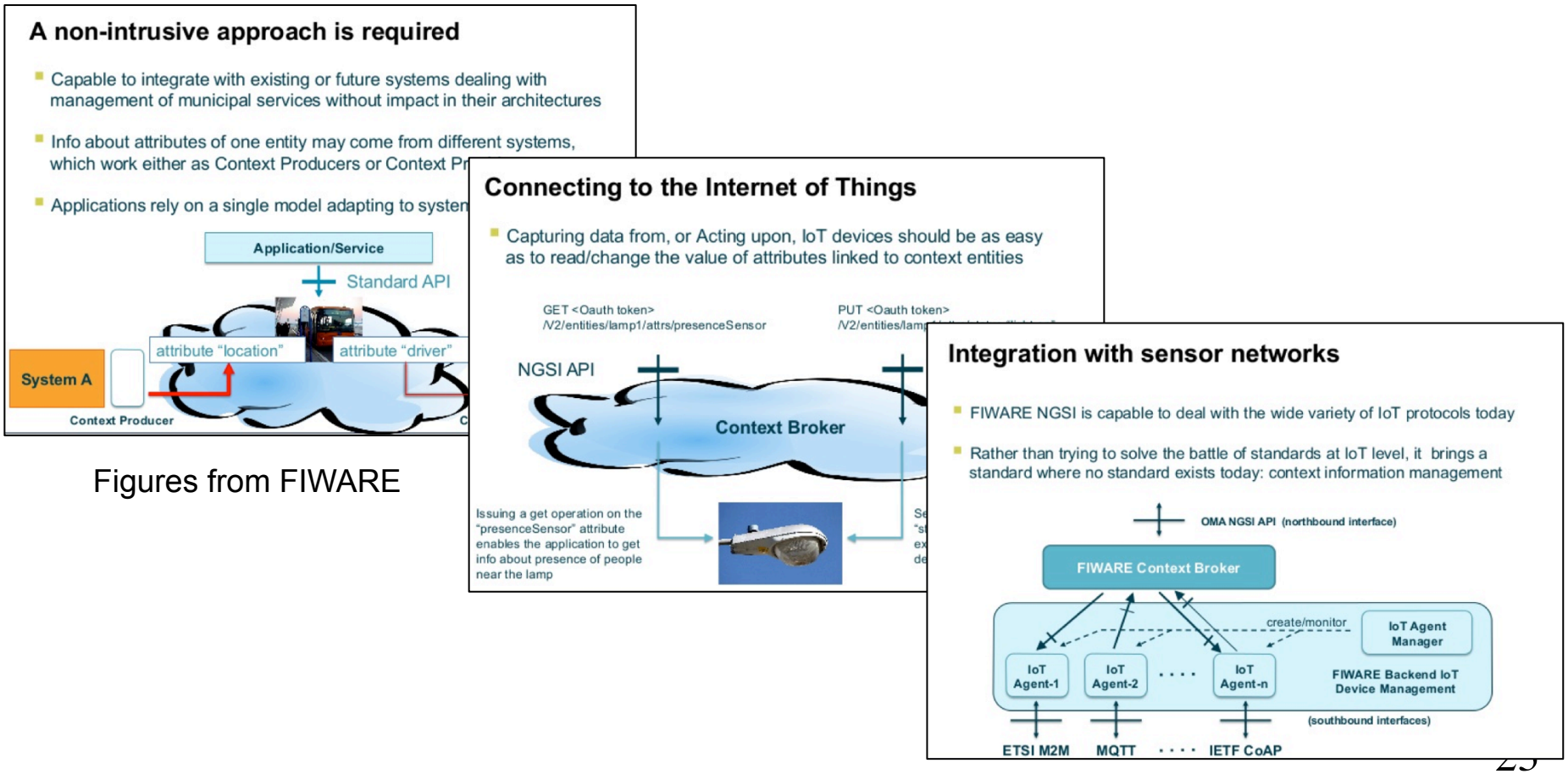
Node 10

\!write_key#field_index#19.6



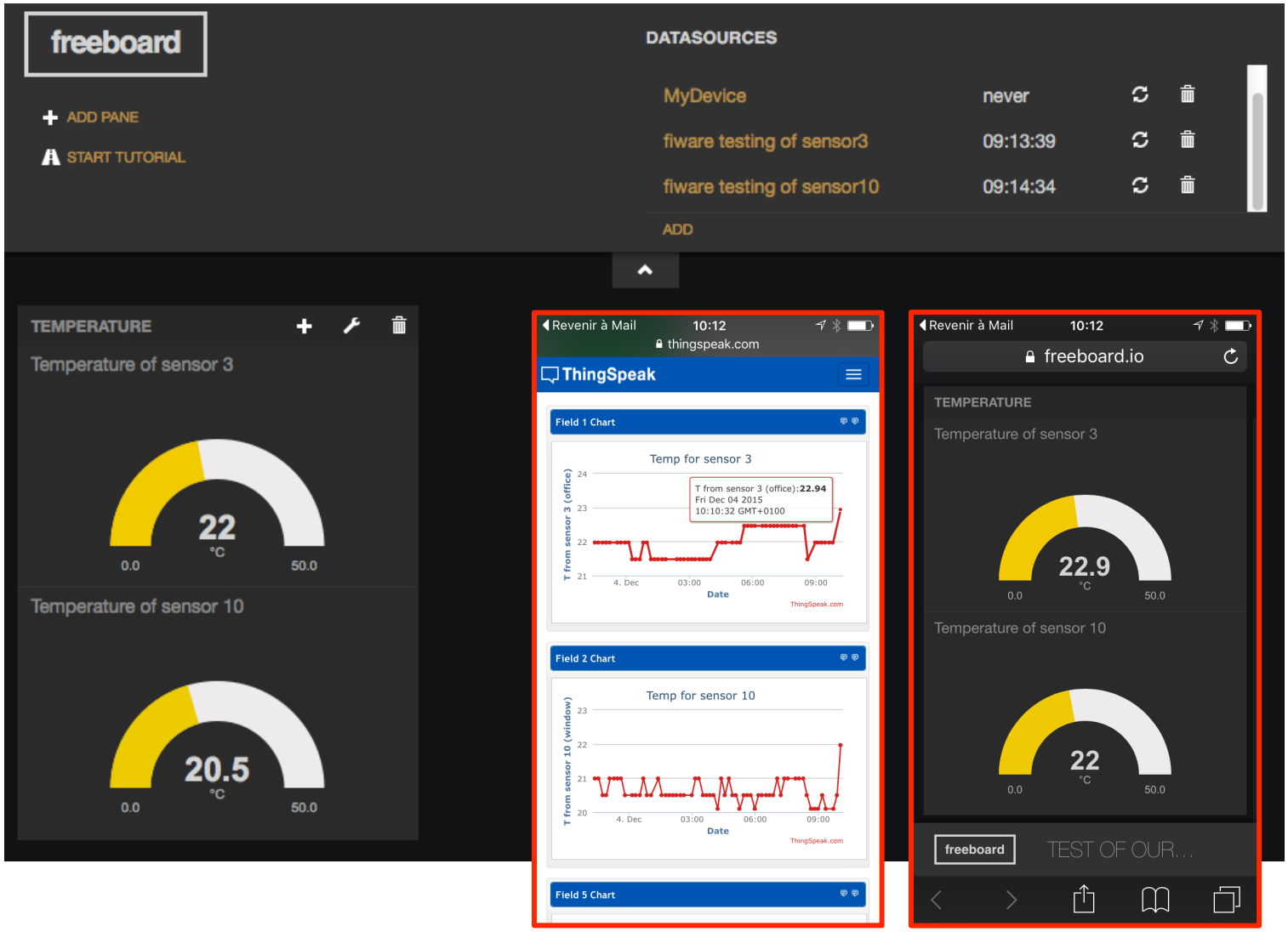
USING FIWARE CLOUD

- FIWARE support has been added with EGM script

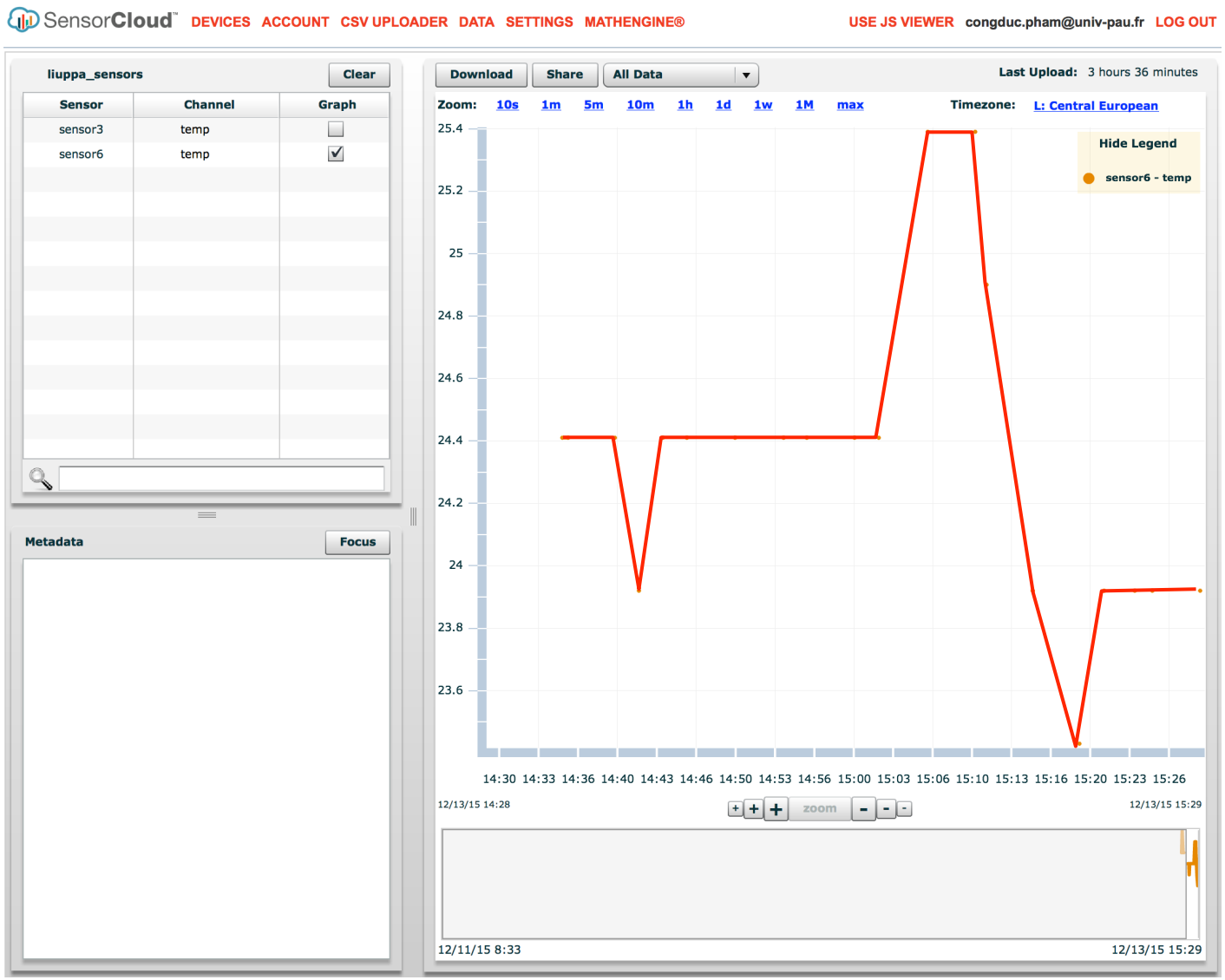


Figures from FIWARE

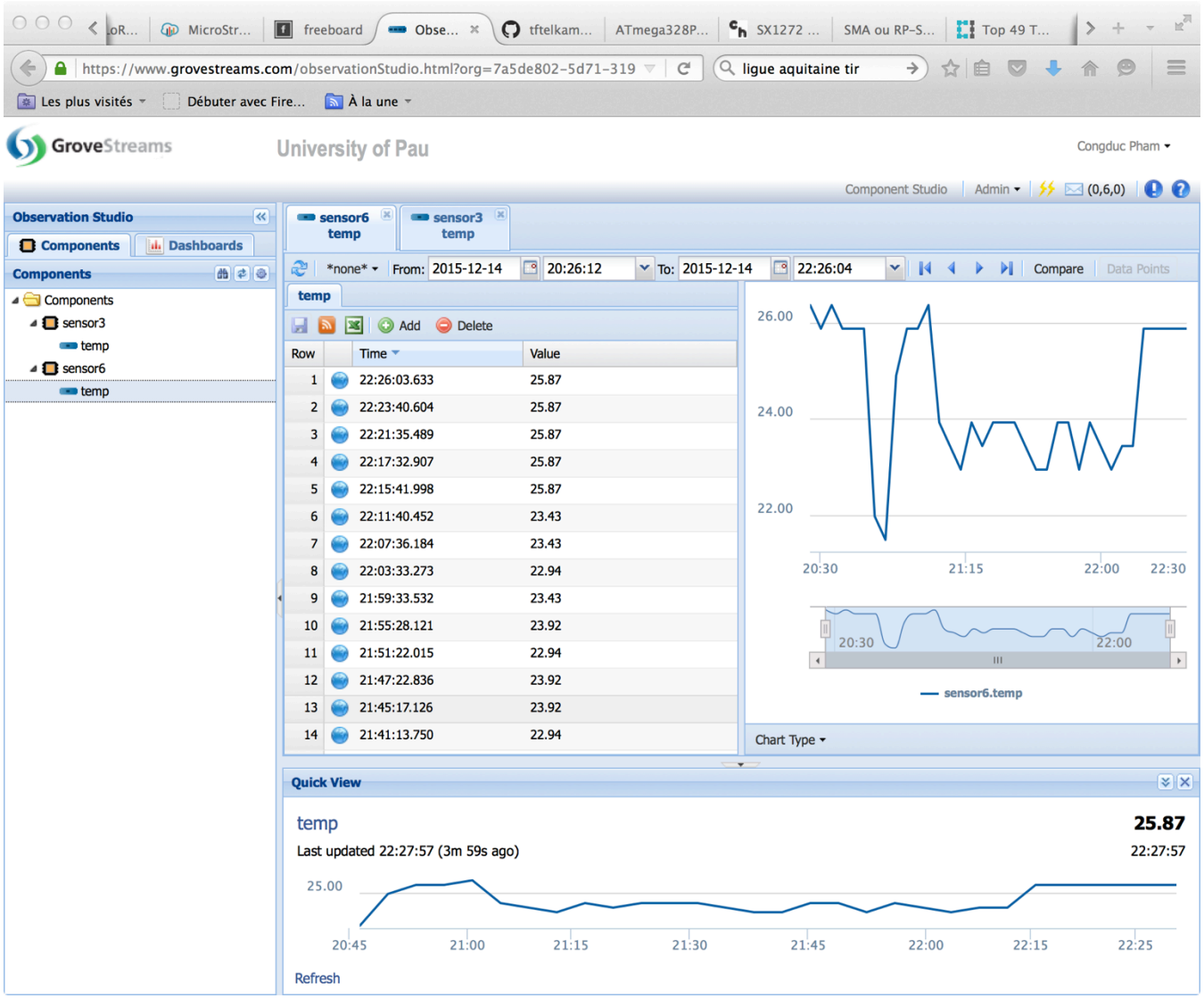
FREEBOARD IOT CLOUD WITH FIWARE



USING SensorCloud™



USING GroveStreams



ADD A NEW CLOUD

```
if (ch=='\\'):
                now = datetime.datetime.now()

                if _validappkey==1:

                                print 'valid app key: accept data'

                                ch=sys.stdin.read(1)


                                if (ch=='$'): #log on Dropbox

                                                data = sys.stdin.readline()
                                                print "rcv msg to log (\$) on dropbox: "+data,
                                                f=open(os.path.expanduser(_telemetrylog_filename),"a")
                                                f.write(info_str+' ')
                                                now = datetime.datetime.now()
                                                f.write(now.isoformat()+'> ')
                                                f.write(data)
                                                f.close()

                                elif (ch=='&' and _firebase==1): #log on Firebase

                                                data = sys.stdin.readline()
                                                print 'rcv msg to log (\&) on firebase: '+data,
                                                now = datetime.datetime.now()
                                                firebase_msg = {
                                                                'dst':dst,
                                                                }
                                                sensor_entry='sensor%d'% (src)
                                                msg_entry='msg%d' % (seq)
                                                date_entry=now.date()
                                                db_entry = _dbpath+date_entry.isoformat()+'/'+sensor_entry
                                                result = _db.put(db_entry, msg_entry, firebase_msg)

                                elif (ch=='!' and _thingspeak==1): #log on ThingSpeak
```

Add your own data prefix or replace the default one with your own cloud parameters