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

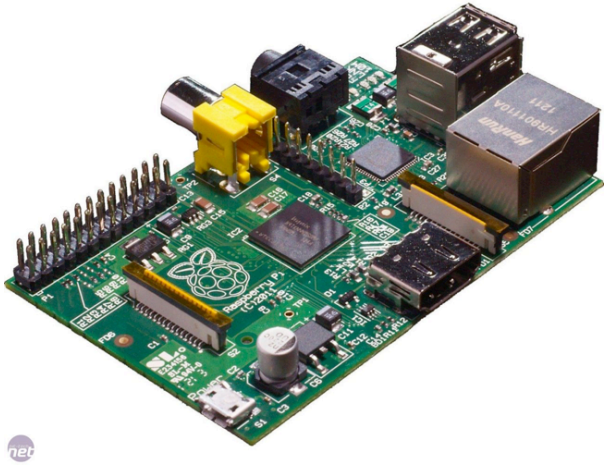
Prof. Congduc Pham
http://www.univ-pau.fr/~cpham
Université de Pau, France

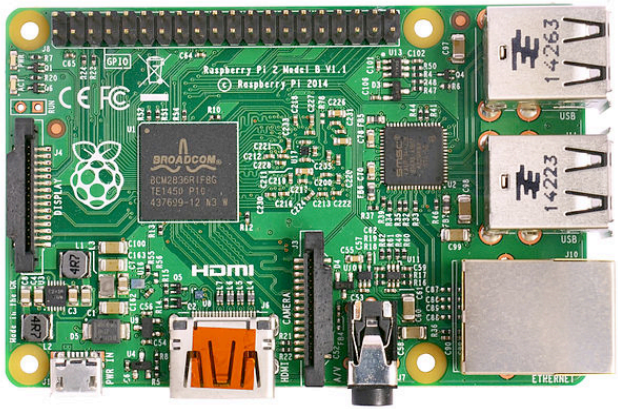
Contents

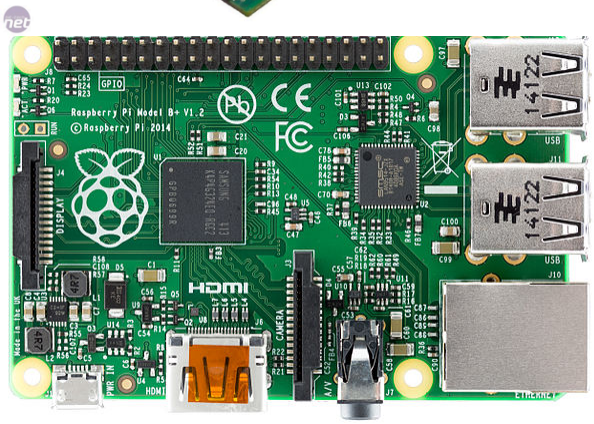
- We will show how to build a low-cost LoRa gateway to collect data from end-devices
- The device part will be shown in a separate tutorial
- The hardware platform is a Raspberry PI. RPI 1B/B+, 2B and 3B have been successfully tested
- Let's get started…

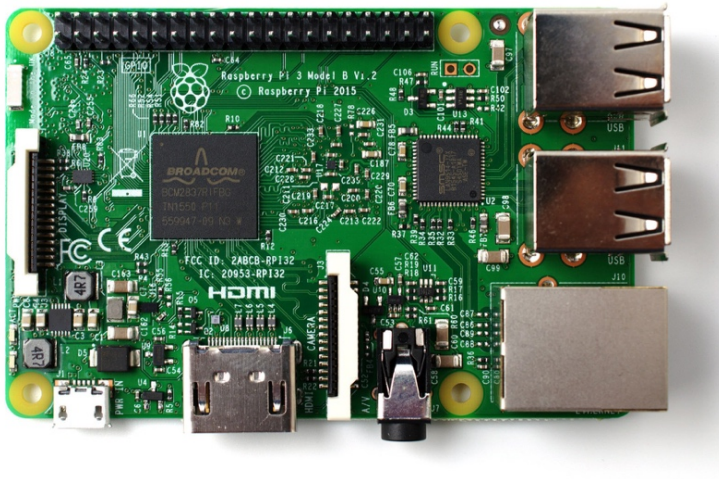
Assembling the hardware

Get the Raspberry



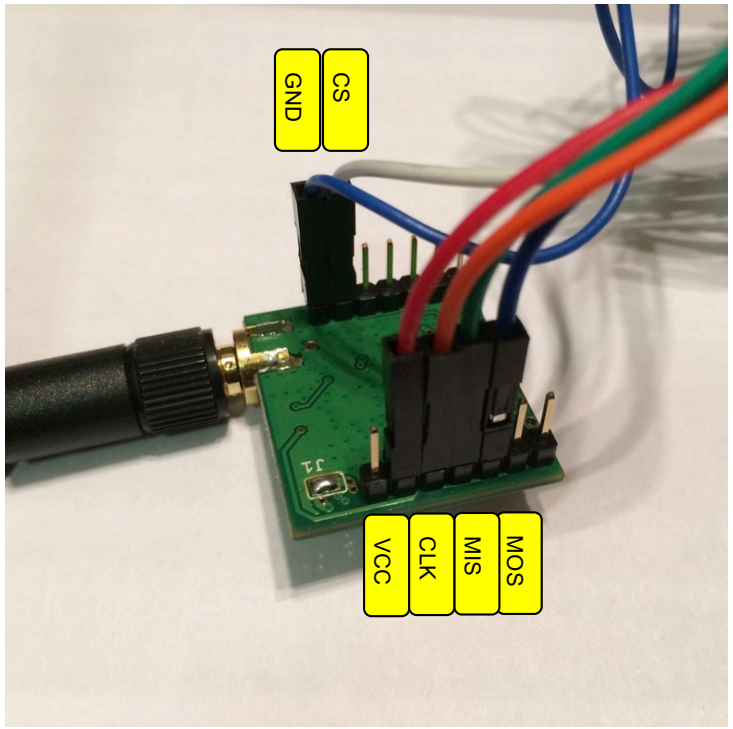


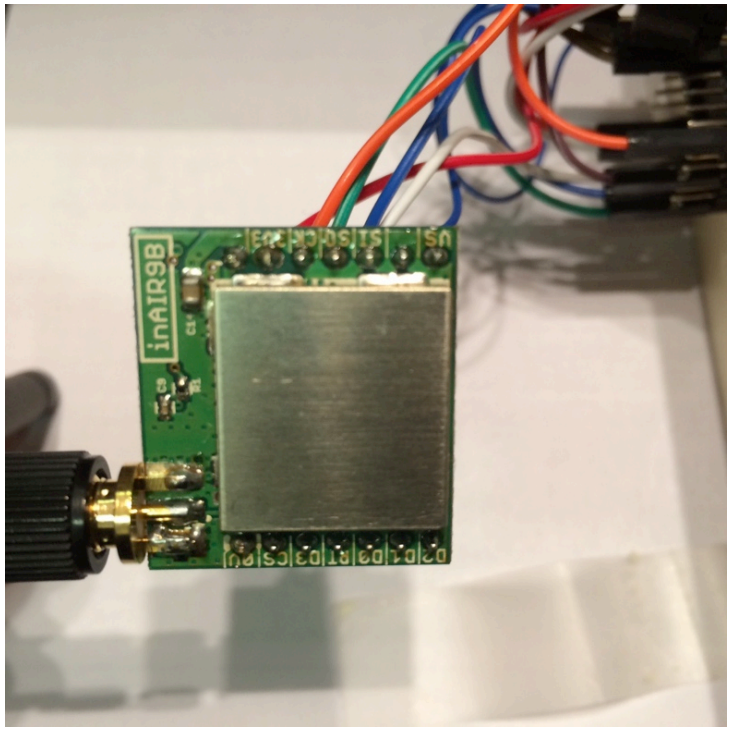




You can use Raspberry 1 model B or B+ or Raspberry 2 model B or Raspberry 3 model B. The most important usefull feature is the Ethernet interface for easy Internet connection. You can use WiFi to get Internet connection by adding a WiFi USB dongle. With the Raspberry 3, WiFi and Bluetooth are embedded on the board.

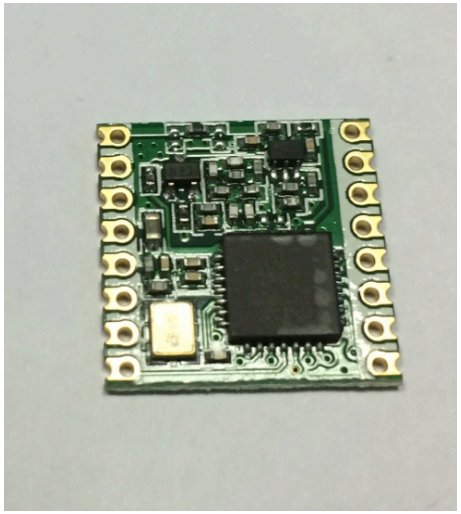
NOW THE RADIO MODULE (1)

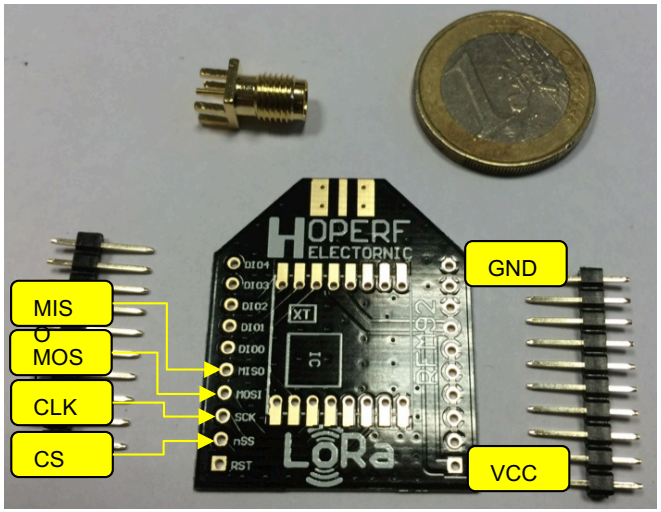




If you go for the inAir9 from Modtronix, then the header pins can come fully assembled. Take the 6mm header pins to have enough length to connect F/F breadboard cables (left). Connect the SPI pins with the F/F cables. Try to use different colors. I use the following colors: MOSI (blue), MISO (green),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

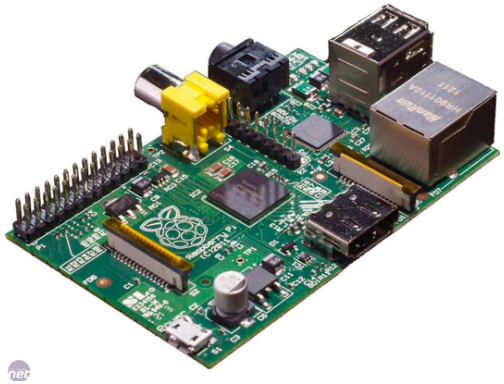
NOW THE RADIO MODULE (2)

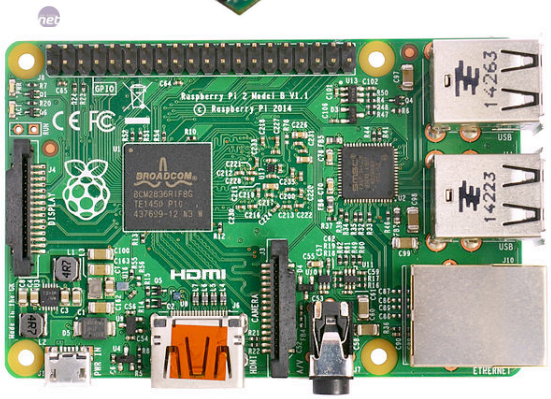


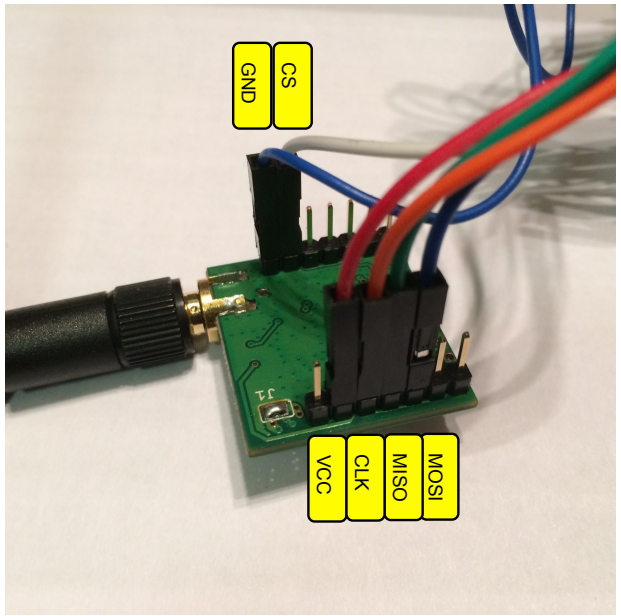


If you take the HopeRF RFM 92W/95W you will need the adaptor breakout and you have to go though some delicate but simple soldering tasks! It is not difficult but you have to trained a bit before! Then, like for the inAir9, use F/F breadboard cable to connect the SPI pins, using different colors as explained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

Connecting the radio module (1)



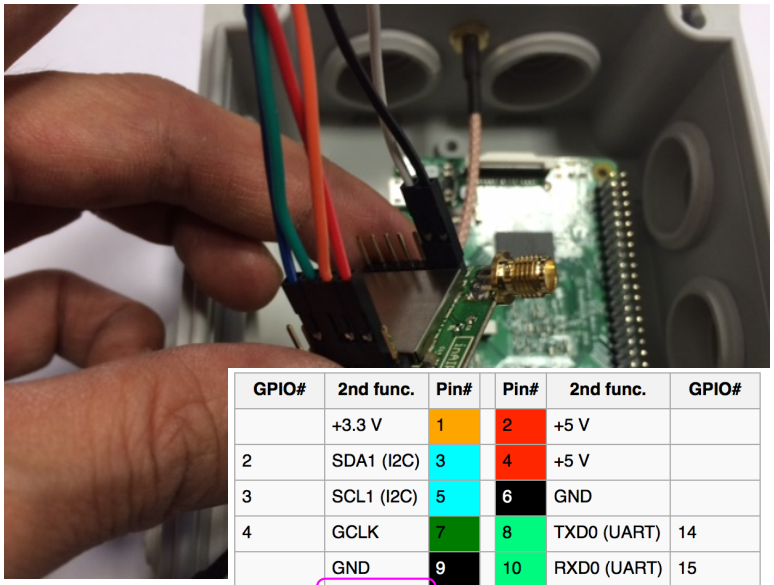




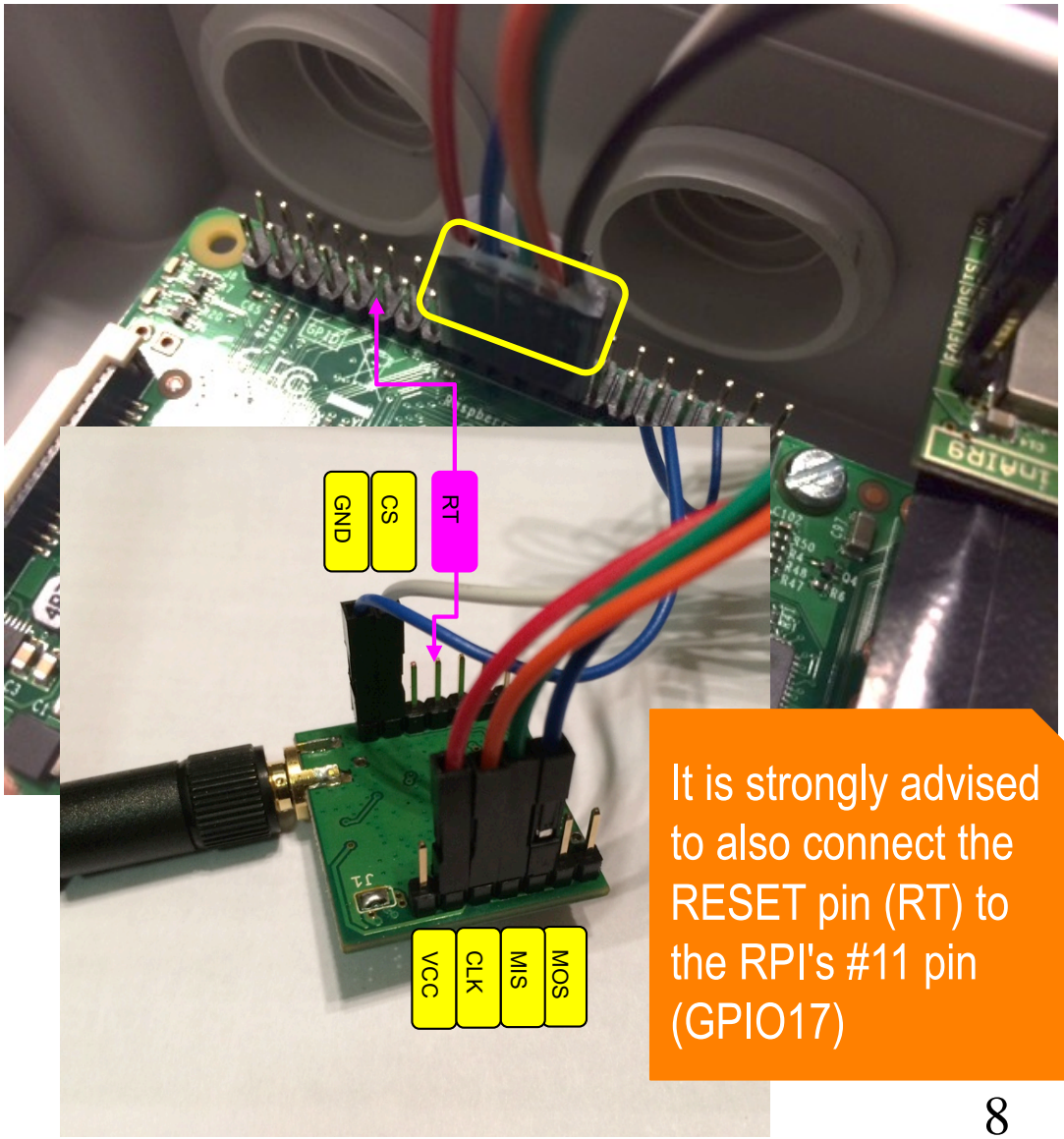
GPIO#	2nd func.	Pin#	Pin#	2nd func.	GPIO#
	+3.3 V	1	2	+5 V	
2	SDA1 (I2C)	3	4	+5 V	
3	SCL1 (I2C)	5	6	GND	
4	GCLK	7	8	TXD0 (UART)	14
	GND	9	10	RXD0 (UART)	15
17	GEN0	11	12	GEN1	18
27	GEN2	13	14	GND	
22	GEN3	15	16	GEN4	23
	+3.3 V	17	18	GEN5	24
10	MOSI (SPI)	19	20	GND	
9	MISO (SPI)	21	22	GEN6	25
11	SCLK (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

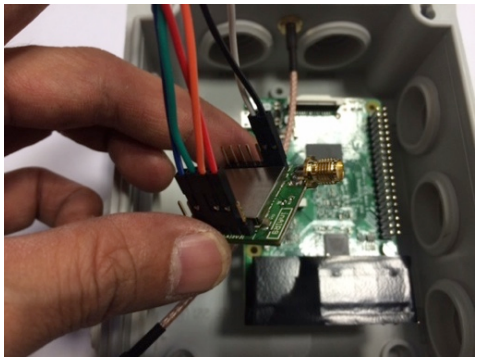


It is strongly advised to also connect the RESET pin (RT) to the RPI's #11 pin (GPIO17)

PUT IT IN A BOX

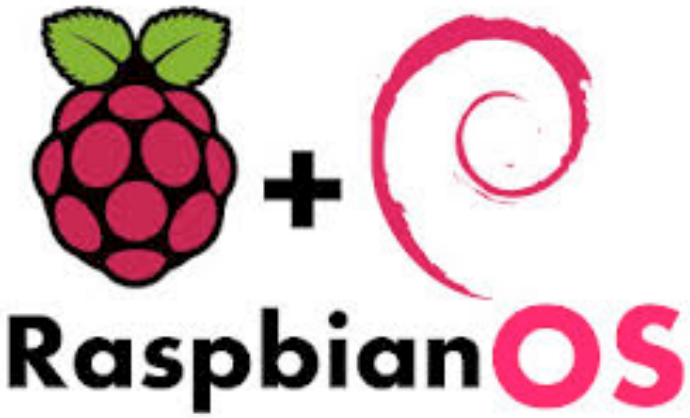




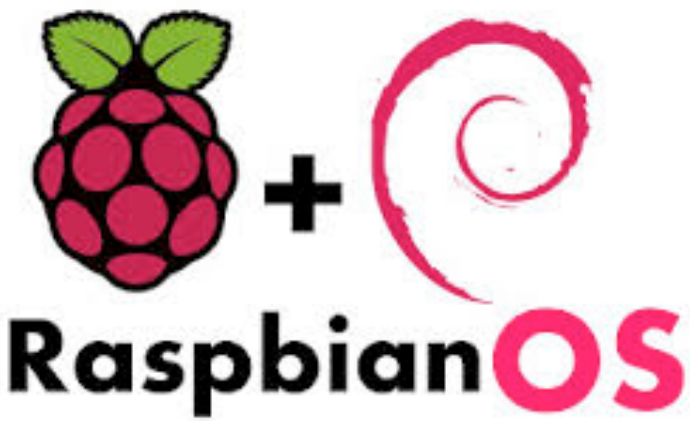


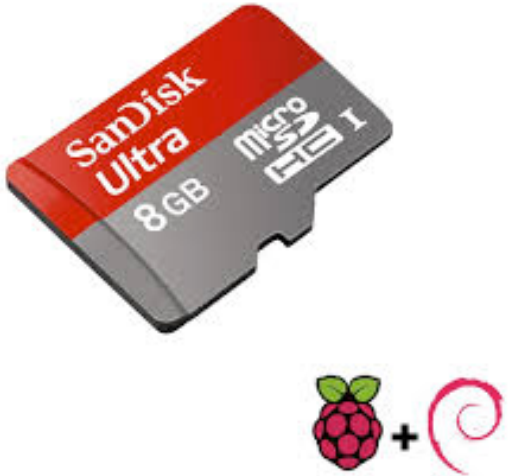

You can have a more integrated version, with a box for outdoor usage and PoE splitter to power the Raspberry with the Ethernet cable. See how we also use a DC-DC converter to get the 5V for the RPI.

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 provide the full image to burn on the SD card. It's 8GB!

http://cpham.perso.univ-pau.fr/LORA/WAZIUP/raspberrypi-jessie-WAZIUP-demo.dmg.zip

Getting the LoRa specific GW software

CongducPham / LowCostLoRaGw — Unwatch 37 | Unstar 95 | Fork 51

Code | Issues 38 | Pull requests 1 | Projects 0 | Wiki | Pulse | Graphs | Settings

Low-cost LoRa gateway with SX1272 and Raspberry — Edit

90 commits | 1 branch | 0 releases | 2 contributors

Branch: master | New pull request | Create new file | Upload files | Find file | Clone or download

Congduc Pham update README — Latest commit 25095e6 2 days ago

Arduino	update README	2 days ago
Raspberry	fix PA_BOOST bug for Raspberry gateway	8 days ago
gw_advanced	update README	2 days ago
gw_full_latest	update README	2 days ago
tutorials	update README	2 days ago
.gitignore	.DS_Store banished	6 months ago
README.md	update README	2 days ago

Branch: master — LowCostLoRaGw / gw_full_latest /

Congduc Pham update README

..	
aes-python-lib/LoRaWAN	add the gw_full_latest folder for easier
downlink	add the gw_full_latest folder for easier
php	add the gw_full_latest folder for easier
rapidjson	add the gw_full_latest folder for easier
scripts	add the gw_full_latest folder for easier
sensors_in_raspi	add the gw_full_latest folder for easier
CloudFireBase.py	update Cloud management with separa
CloudFireBaseAES.py	some more bug fixes
CloudFireBaseLWAES.py	some more bug fixes
CloudGroveStreams.py	update Cloud management with separa
CloudMongoDB.py	update cloud scripts
CloudThingSpeak.py	update Cloud management with separa
MongoDB.py	add the gw_full_latest folder for easier
README-NewCloud.md	update Cloud management with separa
README-advanced.md	update README

```
> 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 & folder of the github's **gw_full_latest folder** into the lora_gateway folder.

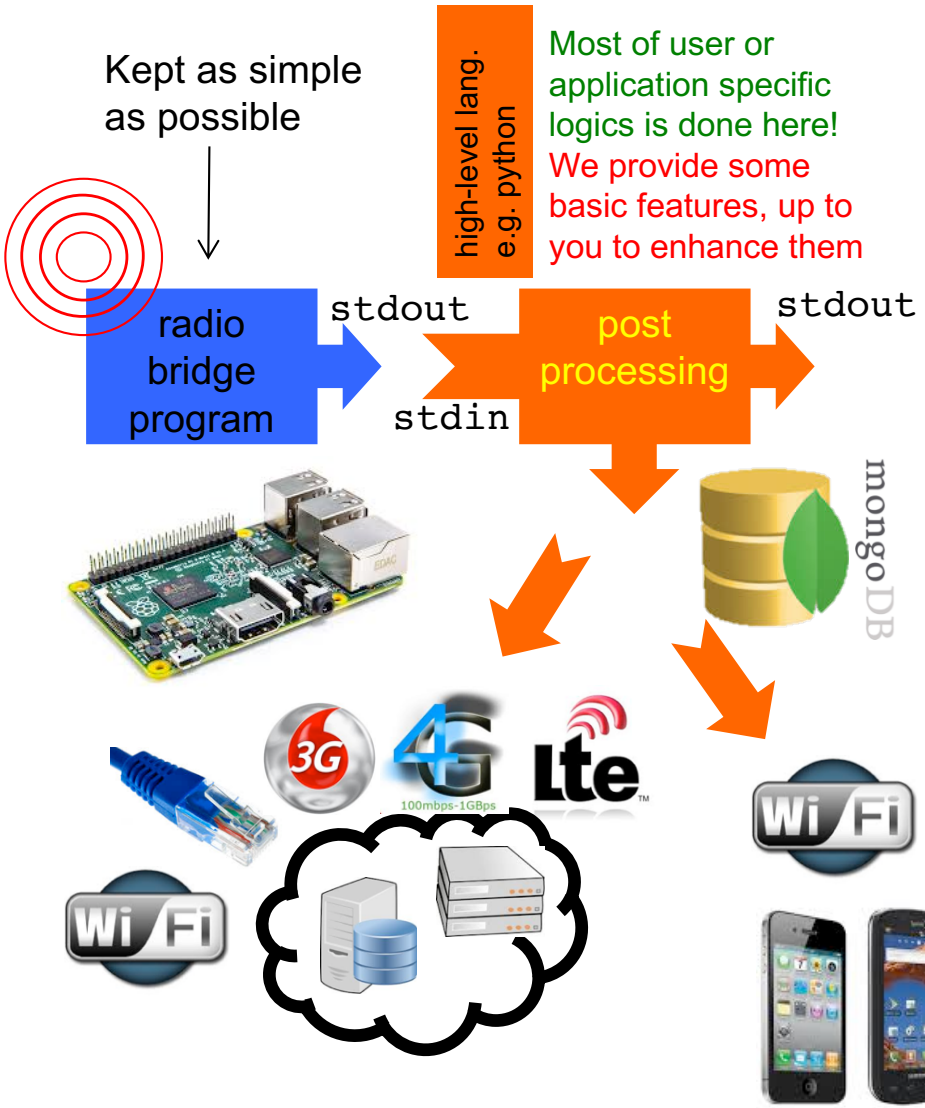
Compiling the GW software

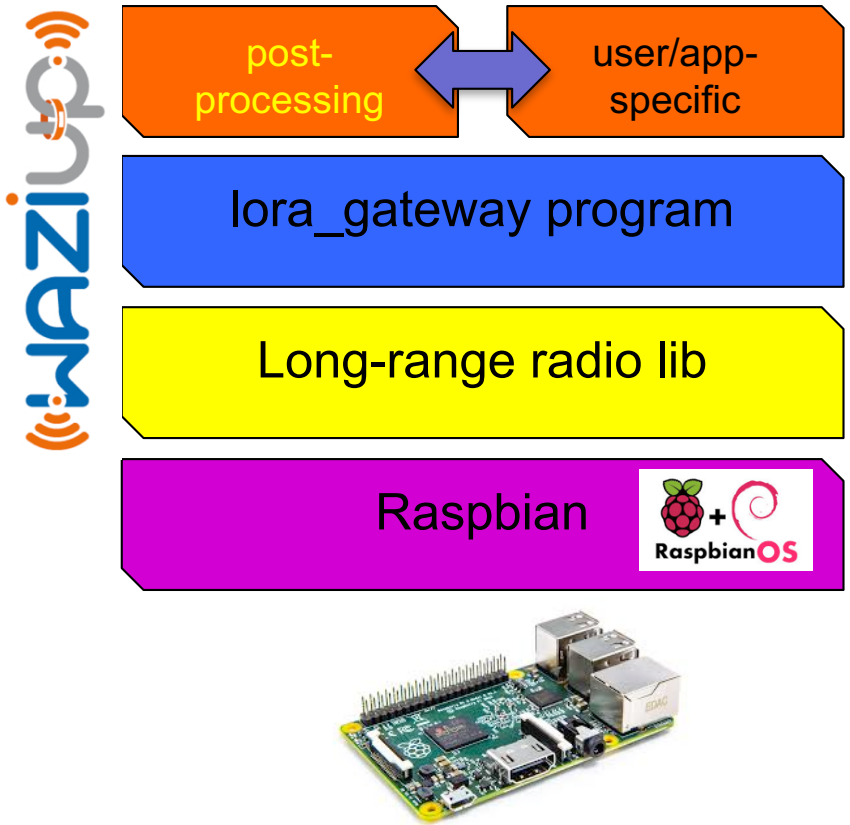
```
> cd lora_gateway
> make lora_gateway
g++ -DRASPBERRY -DIS_RCV_GATEWAY -c lora_gateway.cpp -o lora_gateway.o
g++ -c arduPi.cpp -o arduPi.o
g++ -c SX1272.cpp -o SX1272.o
g++ -lrt -lpthread lora_gateway.o arduPi.o SX1272.o -o lora_gateway
```

If you have a RPI 2 or RPI3, then type:

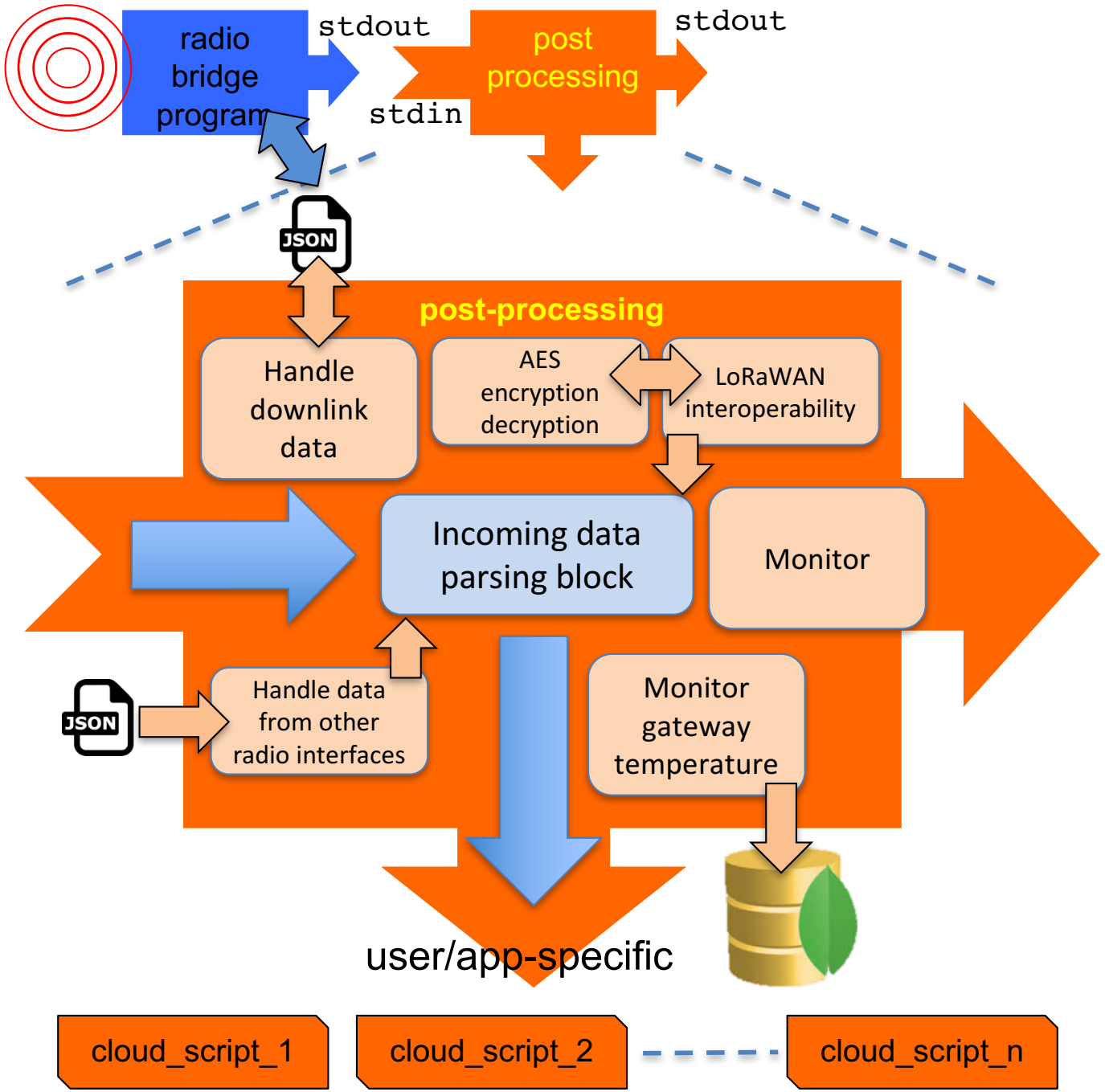
> make lora_gateway_pi2

Our Low-Cost gateway architecture

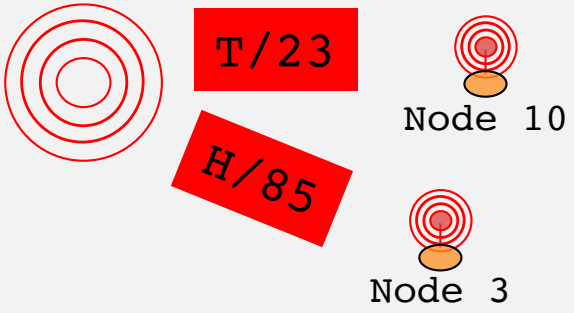




POST-PROCESSING BLOCK



Starting the basic gateway




```
> sudo ./lora_gateway
Power ON: state 0
Default sync word: 0x12
LoRa mode: 1
Setting mode: state 0
Channel CH_10_868: state 0
Set LoRa Power to 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=4 SNR=9 RSSIpkt=-54
^p1,16,10,0,4,9,-54
^r125,5,12
^t2016-02-25T01:51:11.058
T/23
--- rxlora. dst=1 type=0x10 src=3 seq=0 len=4 SNR=8 RSSIpkt=-54
^p1,16,3,0,4,8,-54
^r125,5,12
^t2016-02-25T01:53:13.067
H/85
```

Post-processing received data

```
> sudo ./lora_gateway | python ./post_processing_gw.py
Power ON: state 0
Default sync word: 0x12
LoRa mode: 1
Setting mode: state 0
Channel CH_10_868: state 0
Set LoRa Power to 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(DATA) src=10 seq=0 len=4 SNR=9 RSSIpkt=-54
Rcv ctrl packet info 1,16,10,0,4,9,-54
(dst=1 type=0x10 src=10 seq=0 len=4 SNR=9 RSSI=-54)
rcv ctrl radio info (^r): 125,5,12
splitted in:  [125, 5, 12]
(BW=500 CR=5 SF=12)
rcv timestamp (^t): 2016-02-25T01:53:13.067
got first framing byte
--> got data prefix
T/23
```

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

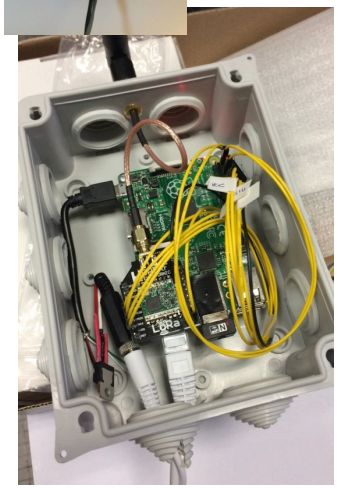
^p provides information on the last received packet: dst, type, src, seq, len, SNR & RSSI

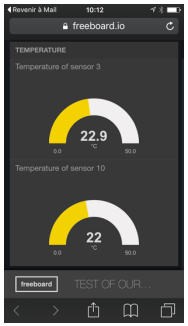
^r provides radio information on the last received packet: bw, cr & sf

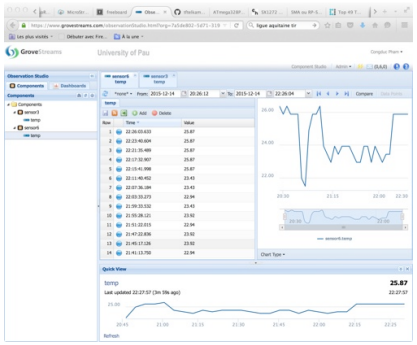
^t provides timestamp information on the last received packet

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

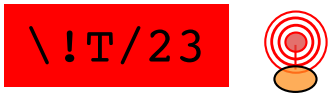






Data received at the gateway can be pushed to IoT clouds. We provide python script examples for many IoT cloud platforms. Most of clouds with REST API can be easily integrated.

LOG RECEIVED MESSAGES USING CLOUD SERVICES



Node 10

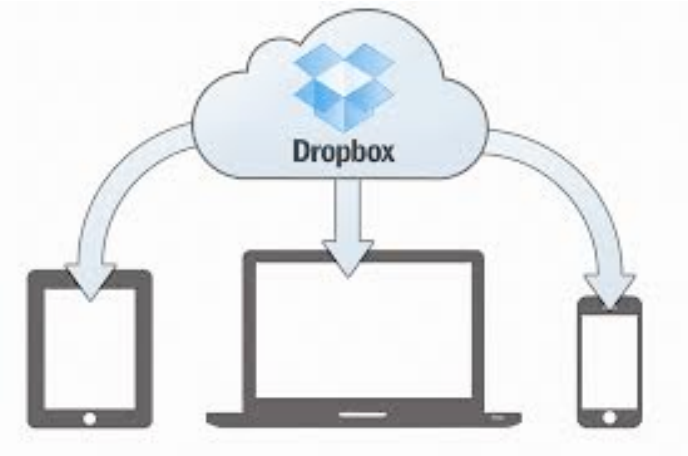
```
SX1272/76 configured as LR-BS. Waiting RF input for transparent RF-serial bridge
--- rxlora. dst=1 type=0x10 src=10 seq=0 len=6 SNR=9 RSSIpkt=-54
Rcv ctrl packet info 1,16,10,0,6,9,-54
(dst=1 type=0x10(DATA) src=10 seq=0 len=6 SNR=9 RSSI=-54)
rcv ctrl radio info (^r): 125,5,12
splitted in:  [125, 5, 12]
(BW=500 CR=5 SF=12)
rcv timestamp (^t): 2016-02-25T01:53:13.067
got first framing byte
--> got data prefix
number of enabled clouds is 1
--> cloud[0]
uploading with python CloudThingSpeak.py
ThingSpeak: uploading
rcv msg to log (\!) on ThingSpeak ( default , 4 ): 23
ThingSpeak: will issue curl cmd
curl -s -k -X POST --data field4=23 https://api.thingspeak.com/[…]
ThingSpeak: returned code from server is 156
--> cloud end
```

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

Using Dropbox

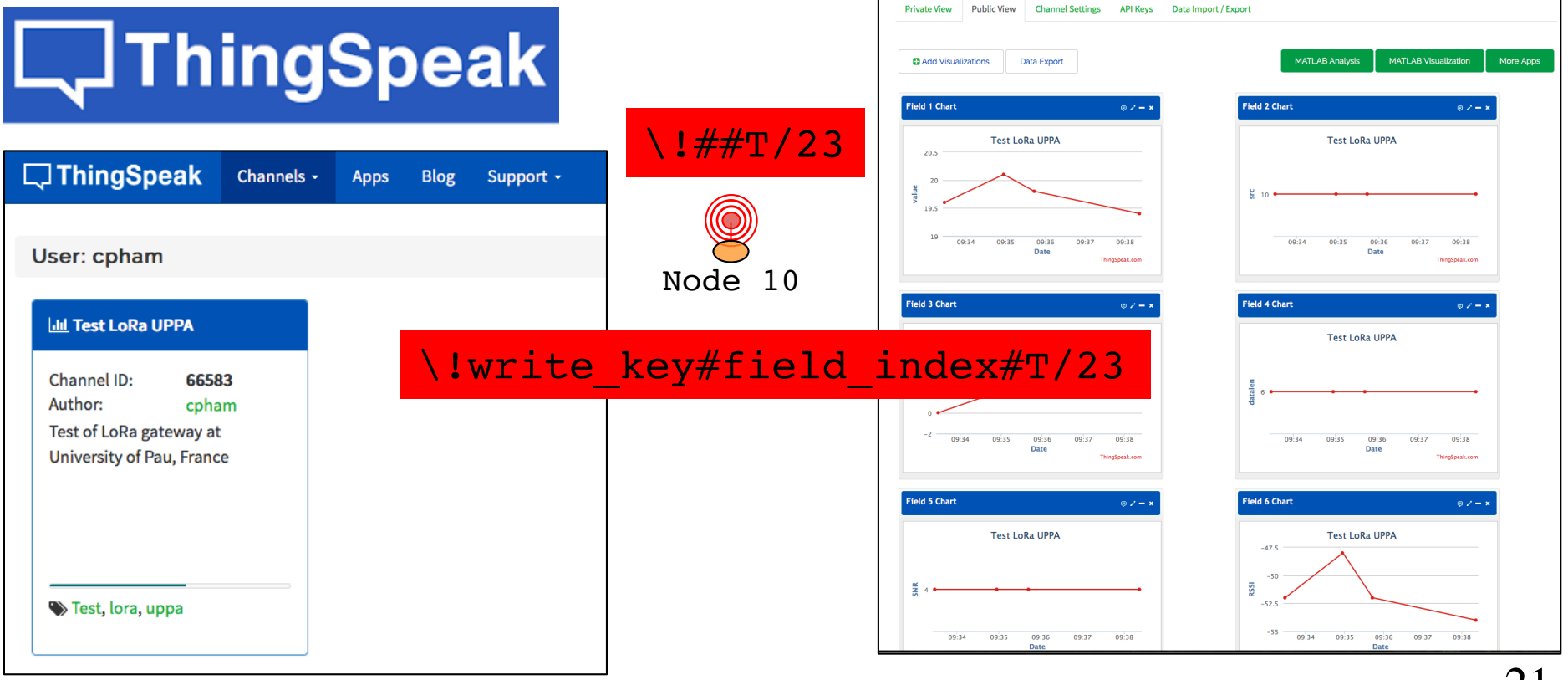
- A message starting with '\$' is logged in a file 'telemetry.log' in a folder shared through Dropbox

```
(src=10 seq=0 len=6 SNR=9 RSSI=-54) 2015-11-04T10:14:30.328413> T/23
(src=10 seq=1 len=8 SNR=8 RSSI=-54) 2015-11-04T10:14:37.443350> T/23.2
(src=10 seq=2 len=6 SNR=8 RSSI=-53) 2015-11-04T10:16:23.343657> T/24
...
```



USING ThingSpeak

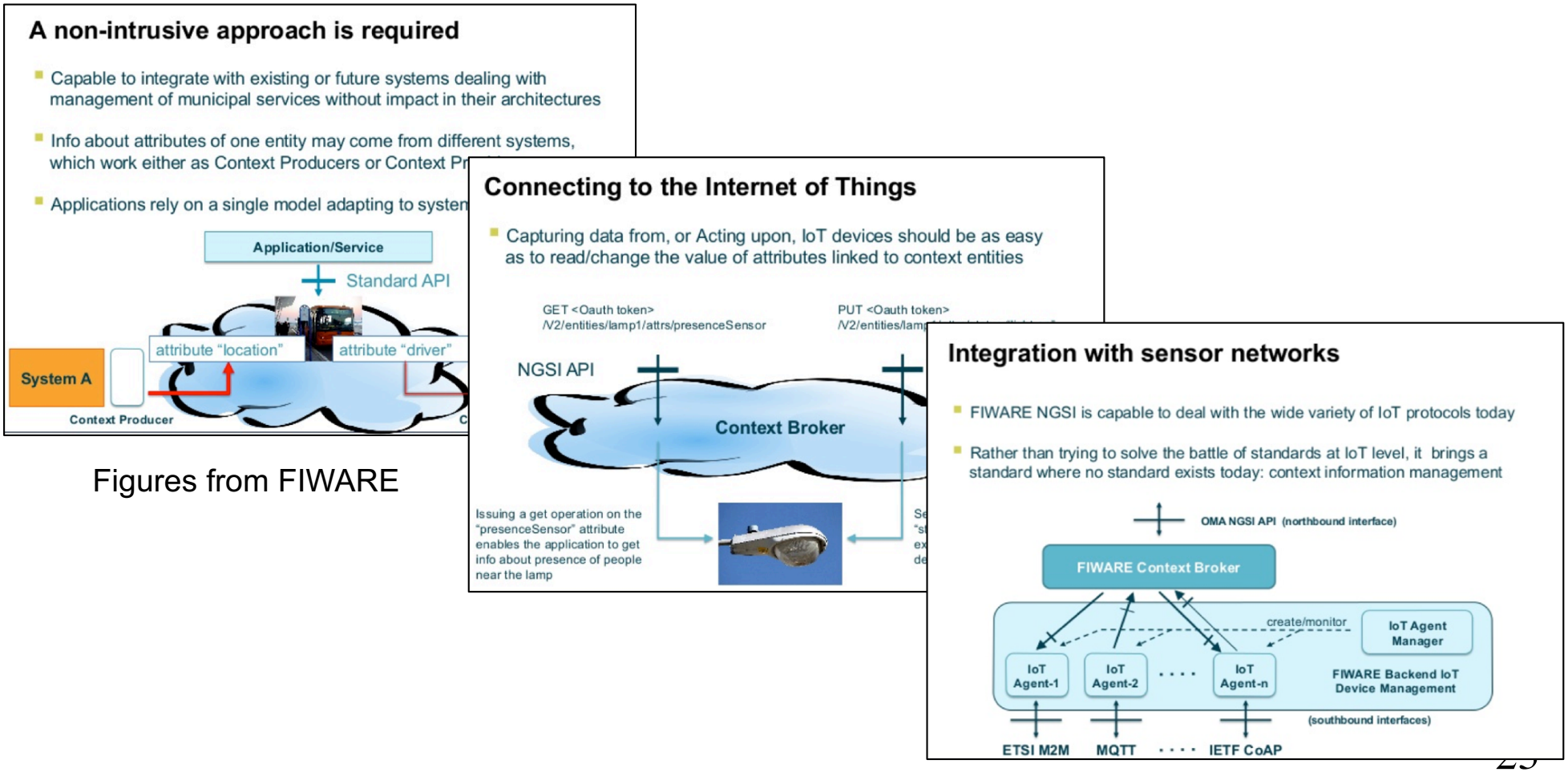
- A message starting with '\!' is uploaded on a cloud, e.g. ThingSpeak



USING Firebase

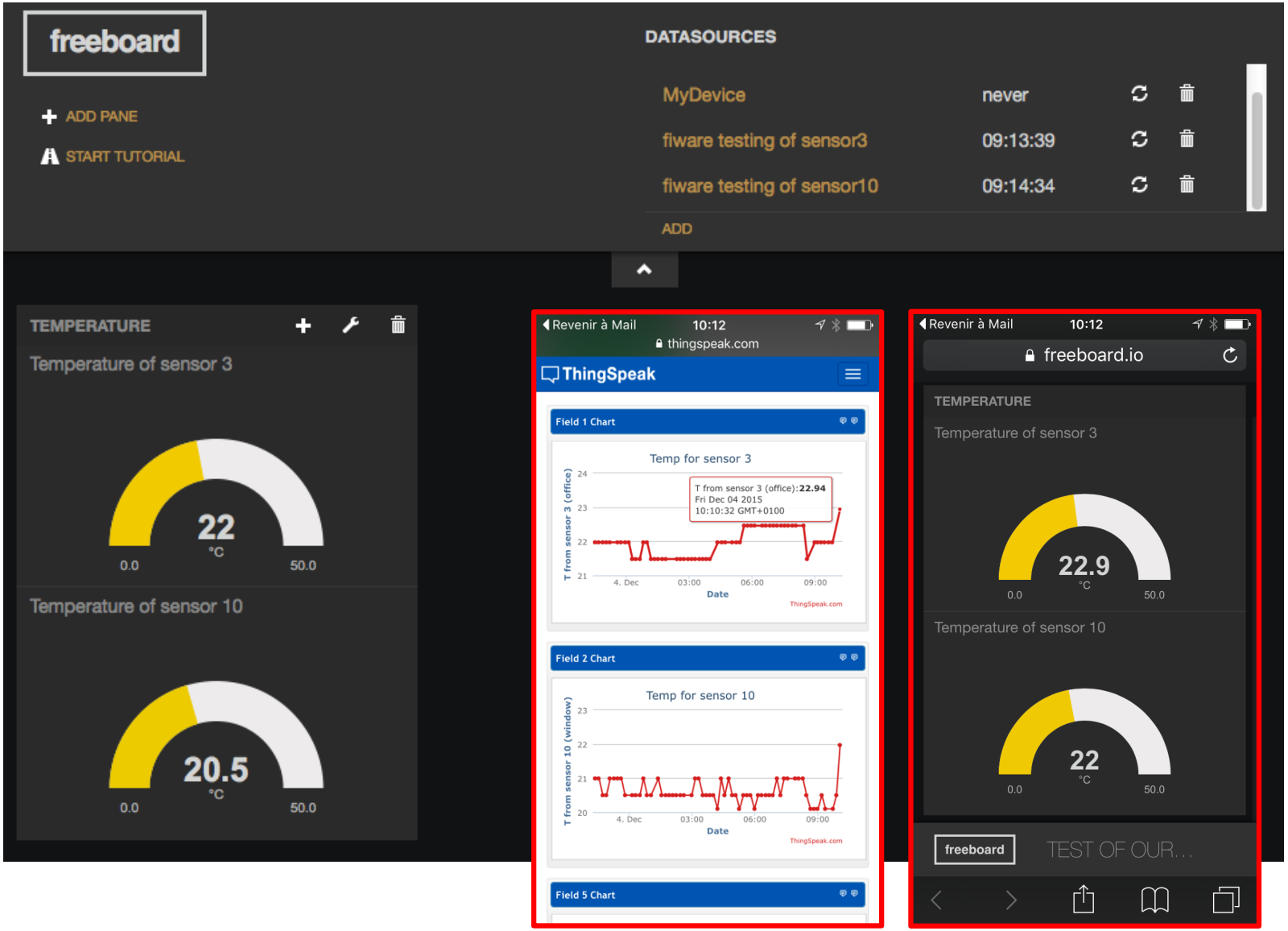
USING FIWARE CLOUD

- FIWARE support has been added with EGM script

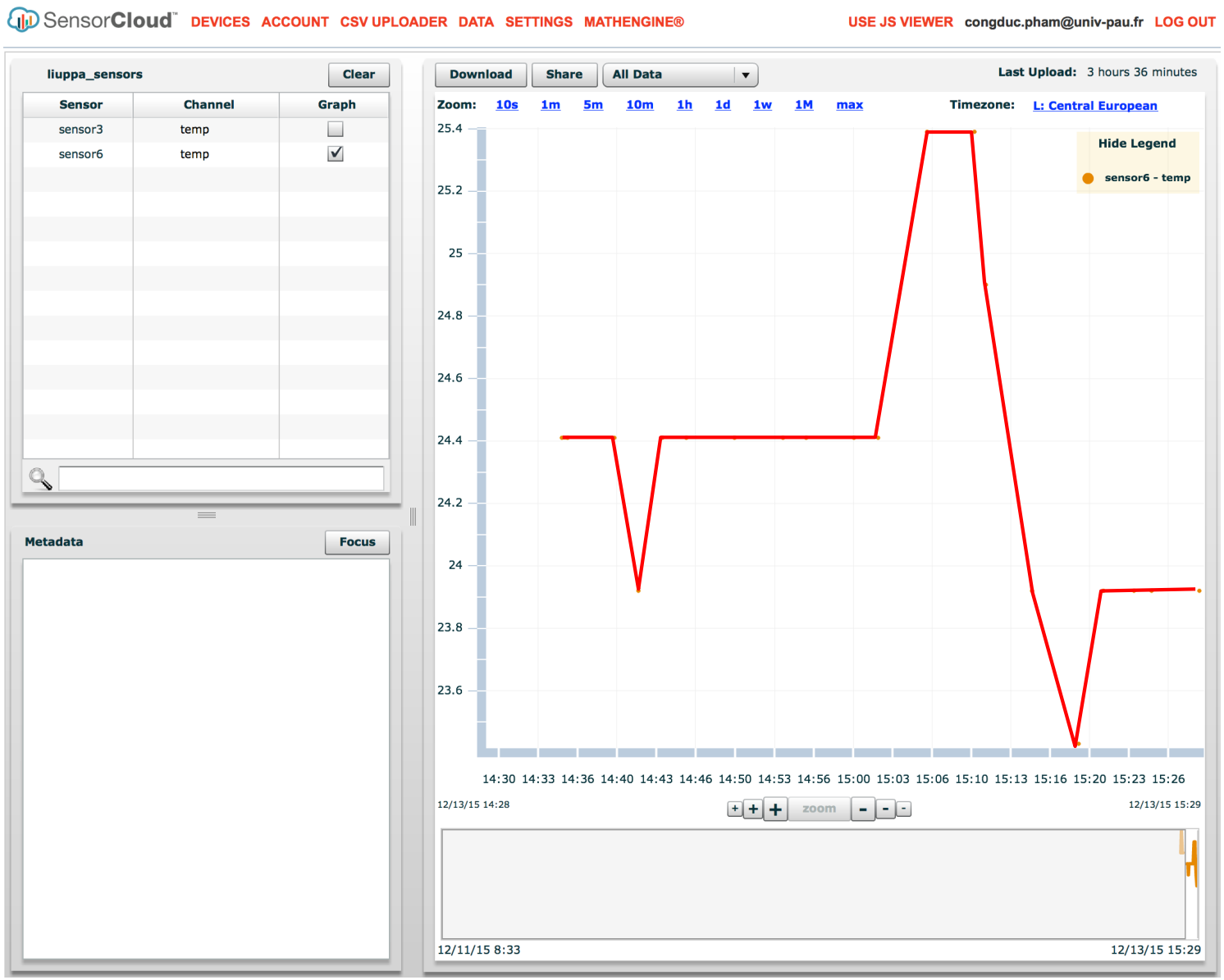


Figures from FIWARE

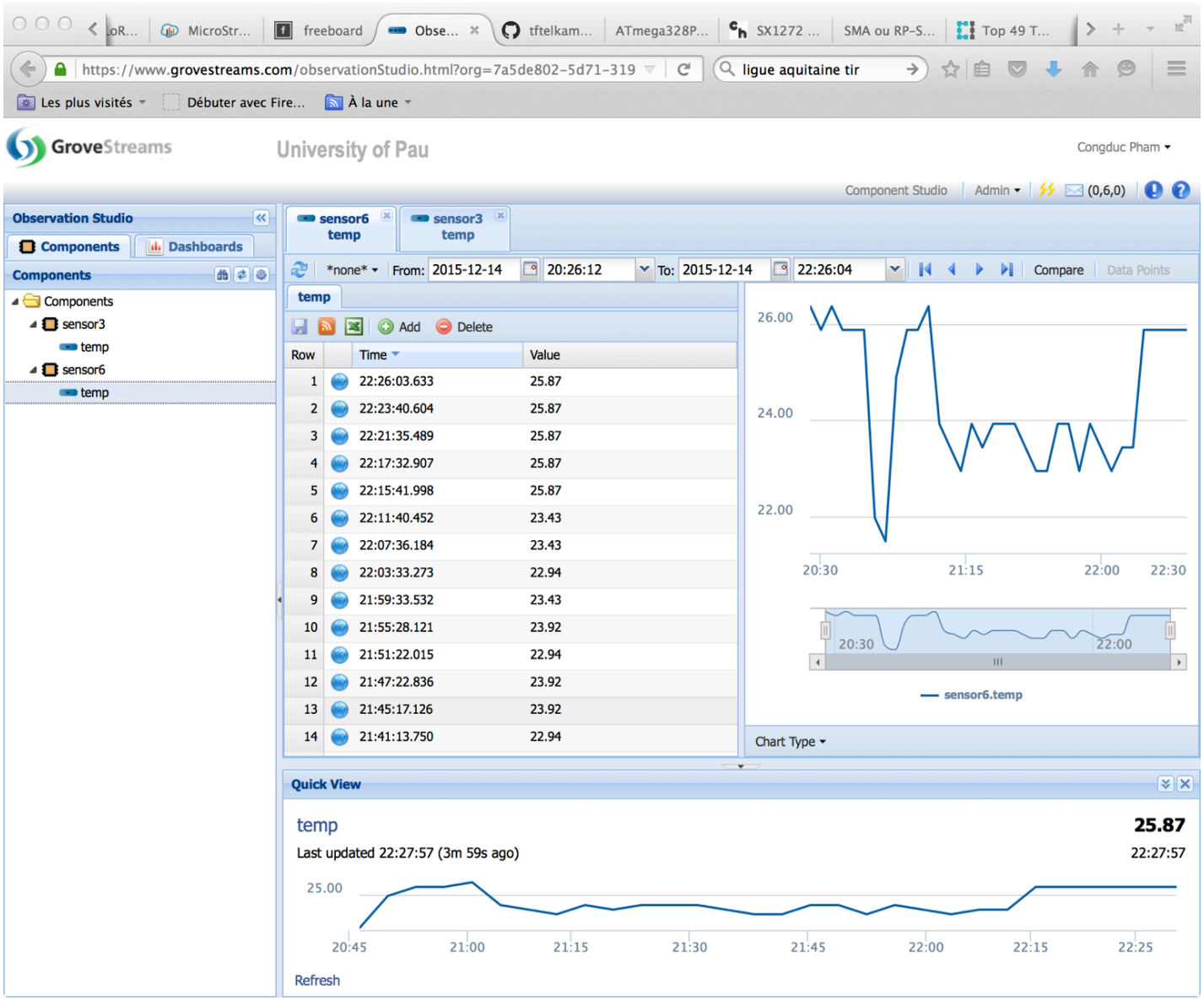
Freeboard IoT cloud with FIWARE



USING SensorCloud™



USING GroveStreams



CLOUDS.JSON


```
{
  "clouds": [
    {
      "notice": "do not remove the MongoDB cloud declaration",
      "name": "Local gateway MongoDB",
      "script": "python CloudMongoDB.py",
      "type": "database",
      "max_months_to_store": 2,
      "enabled": false
    },
    {
      "name": "WAZIUP Orion cloud",
      "script": "python CloudWAZIUP.py",
      "type": "iotcloud",
      "write_key": "",
      "enabled": true
    },
    {
      "name": "ThingSpeak cloud",
      "script": "python CloudThingSpeak.py",
      "type": "iotcloud",
      "write_key": "",
      "enabled": true
    },
    {
      "name": "GroveStreams cloud",
      "script": "python CloudGroveStreams.py",
      "type": "iotcloud",
      "write_key": "",
      "enabled": true
    },
    {
      "name": "Firebase cloud",
      "script": "python CloudFireBase.py",
      "type": "jsoncloud",
      "write_key": "",
      "enabled": true
    },
    {
      "name": "example template",
      "script": "name of your script, preceded by the script launcher",
      "type": "whatever you want FYI",
      "server": "",
      "login": "",
      "password": "",
      "folder": "",
      "write_key": "",
      "enabled": false
    }
  ]
}
```


For each cloud, you have to provide a script and the launcher program (e.g. python)

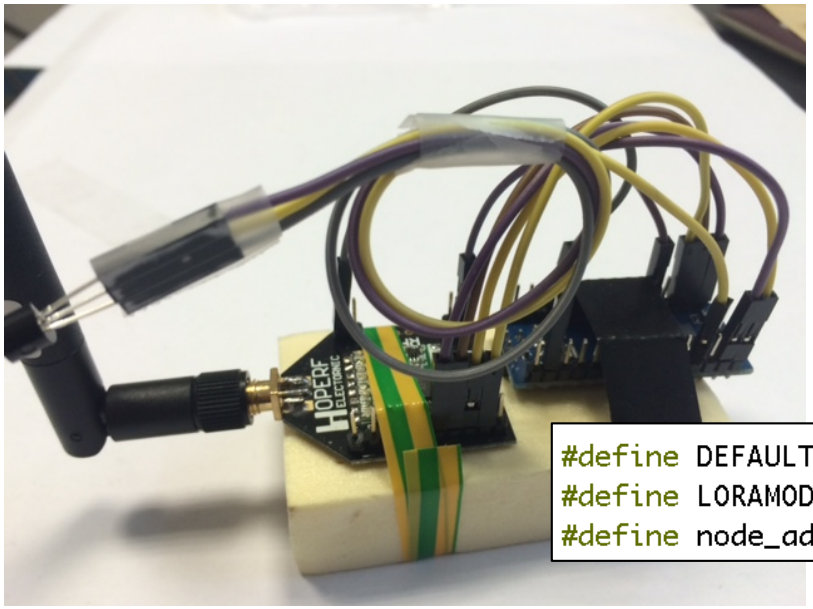
Enabled clouds will be called by the post-processing stage

Write your own cloud script

- Use our templates to write your own cloud script
  - CloudMongoDB.py, CloudThingSpeak.py, CloudFireBase.py, CloudGroveStreams.py, CloudWAZIUP.py
- A cloud script is called with 5 arguments
  - ldata: the received data
    - e.g. #4#TC/21.5 as 1st argument (sys.argv[1] in python)
  - pdata: packet information
    - e.g. "1,16,3,0,10,8,-45" as 2nd argument (sys.argv[2] in python)
    - interpreted as dst,ptype,src,seq,len,SNR,RSSI for the last received packet
  - rdata: the LoRa radio information
    - e.g. "500,5,12" as 3rd argument (sys.argv[3] in python)
    - interpreted as bw,cr,sf for the last received packet
  - tdata: the timestamp information
    - e.g. "2016-10-04T02:03:28.783385" as 4th argument (sys.argv[4] in python)
  - gwid: the gateway id
    - e.g. 00000027EBBEDA21 as 5th argument (sys.argv[5] in python)

These parameters are passed to the script. It is up to the cloud script to use these parameters or not.

DEFAULT CONFIGURATION



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

\!##TC/18.5

The default configuration in the Arduino_LoRa_temp example is:

Send packets to the gateway (one or many if in range)
LoRa mode 1 & Node short address is 6

The default gateway configuration is also LoRa mode 1

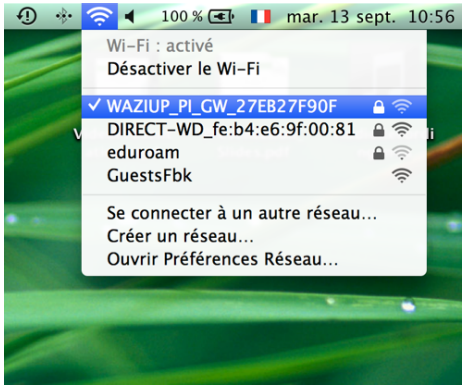
Standalone gateway

SSH TO THE GATEWAY

- If you connected the gateway to your LAN or laptop then the gateway got an IP address. Use this address to connect with SSH to the gateway
- Use ssh pi@RPI_ADDR, where RPI_ADDR is the IP address assigned to the gateway
- Login password is loragateway if you installed from the SD card image

SSH TO THE GATEWAY WITH WIFI

- The gateway is also configured as a WiFi access point with address 192.168.200.1
- Select the WAZIUP_PI_GW_xxxxxxxxxx WiFi
- WiFi password is loragateway
- Then ssh pi@192.168.200.1
- Login password is loragateway

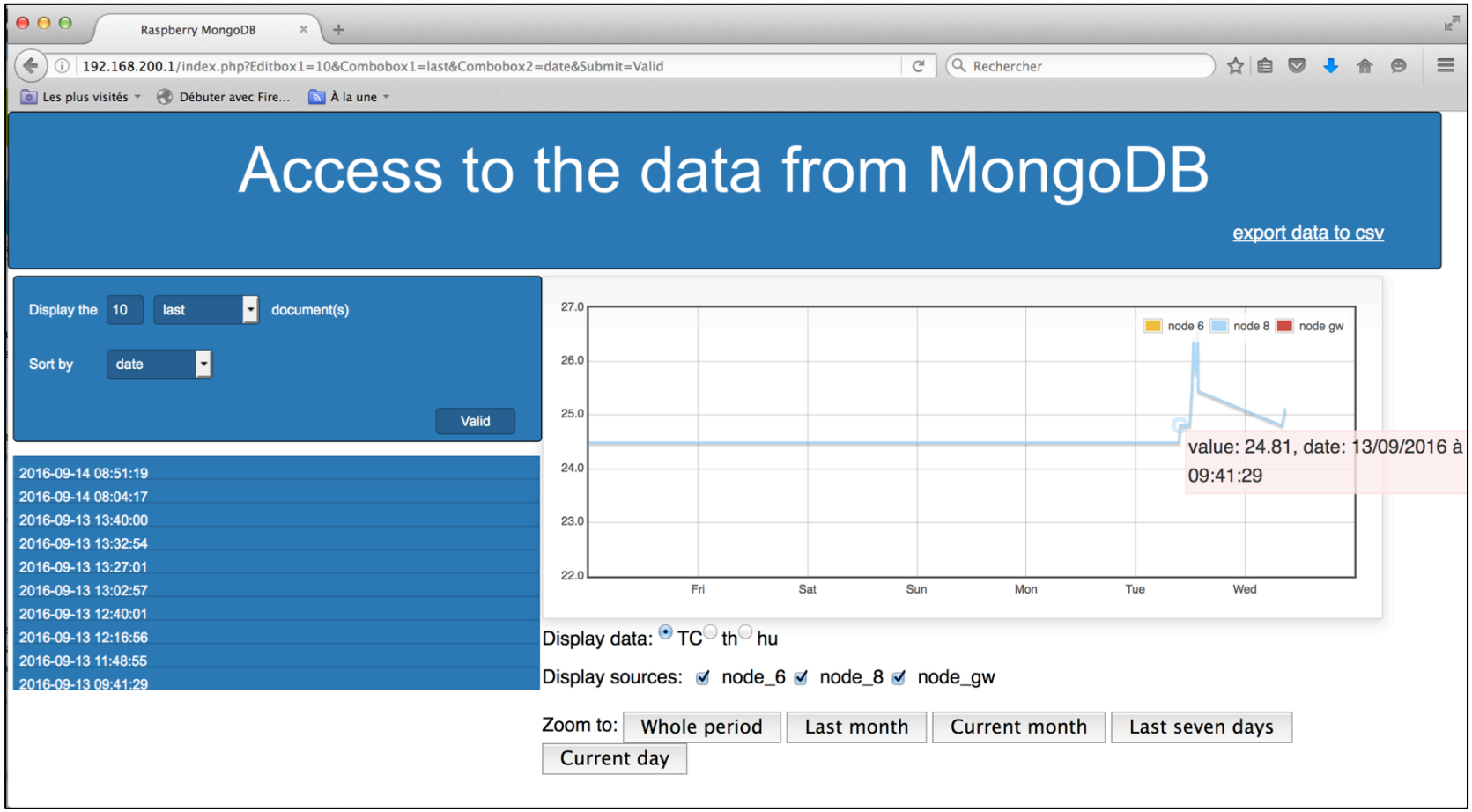



```
cpham — pi@raspberrypi: ~/lora_gateway —
MacBookProRetina-de-Congduc-Pham:~ cpham$ ssh pi@192.168.200.1
pi@192.168.200.1's password:

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Thu Aug  4 17:19:00 2016 from 192.168.200.102
pi@raspberrypi:~ $ cd lora_gateway/
pi@raspberrypi:~/lora_gateway $ ll
total 864
-rw------- 1 pi   pi    44155 Aug  3 16:55 arduPi.cpp
-rw------- 1 pi   pi    16715 Aug  3 16:55 arduPi.h
-rw-r--r-- 1 pi   pi    35164 Aug  3 17:01 arduPi.o
-rw------- 1 pi   pi    43310 Aug  3 16:55 arduPi_pi2.cpp
-rw------- 1 pi   pi    14043 Aug  3 16:55 arduPi_pi2.h
-rw------- 1 pi   pi    77976 Aug  3 16:55 bcm2835.h
```

Connect to the embedded web server

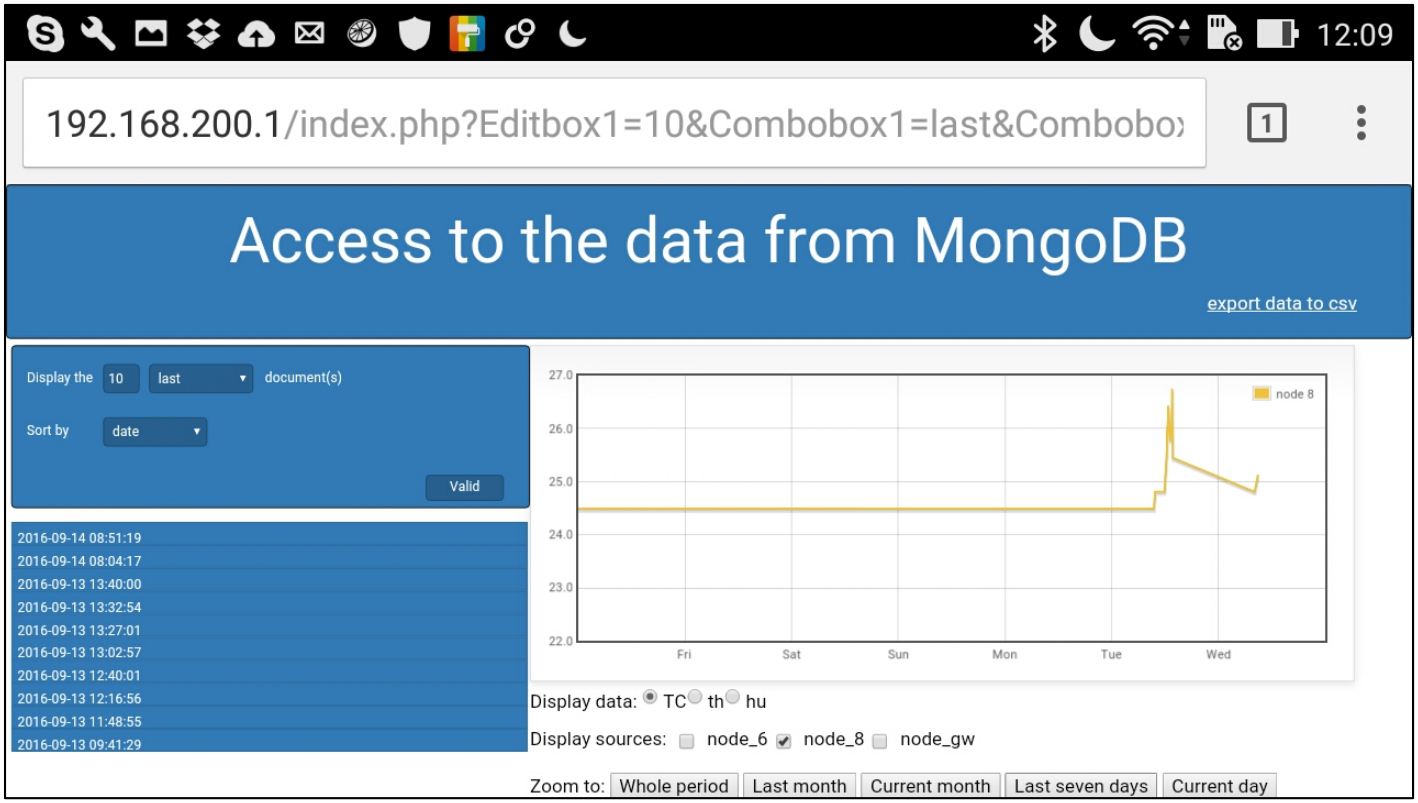
- On the WiFi interface
  - Gateway address is 192.168.200.1
- On the Ethernet interface
  - Gateway address is the IP address assigned by the DHCP server (of your LAN or laptop)
- Choose any of these solutions and open a web browser to enter the gateway IP address in the URL bar
  - http://192.168.200.1

DATA FROM THE LOCAL WEB SERVER



Visualize it on your smartphone!

- Don't forget to join the WAZIUP_PI_GW_xxxxxxxxxx WiFi



Running the gateway without internet access