Low-cost LoRa WaterSense gateway: a step-by-step tutorial

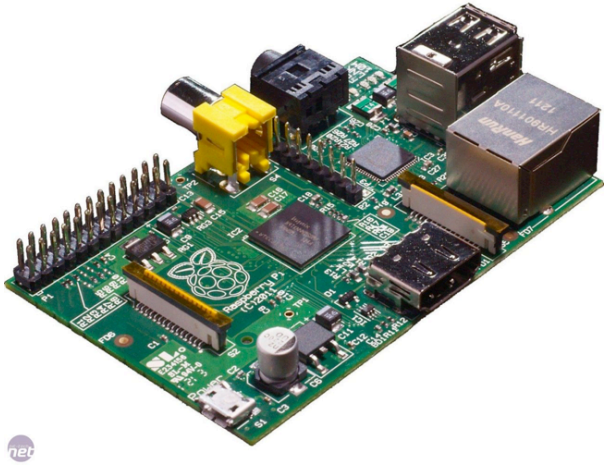
Prof. Congduc Pham
http://www.univ-pau.fr/~cpham
Université de Pau, France

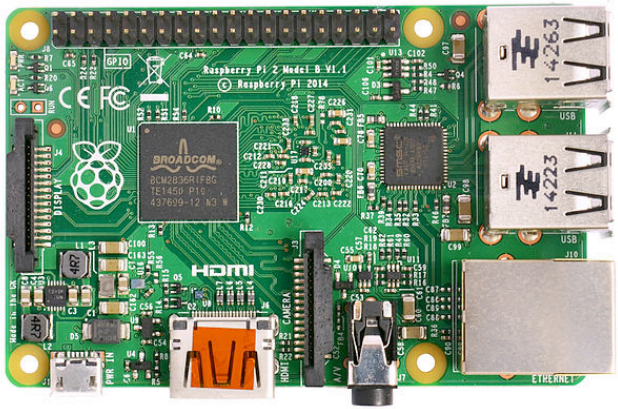
Contents

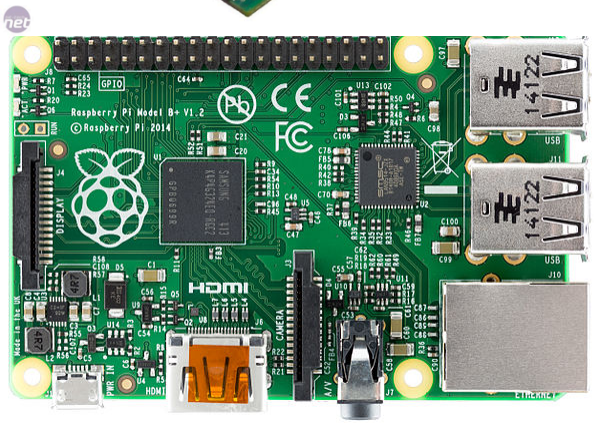
- We will show how to build a low-cost LoRa gateway to collect data from end-devices
- The device part will be shown in a separate tutorial
- The recommended hardware platform is a Raspberry 2B or 3B
- Let's get started…

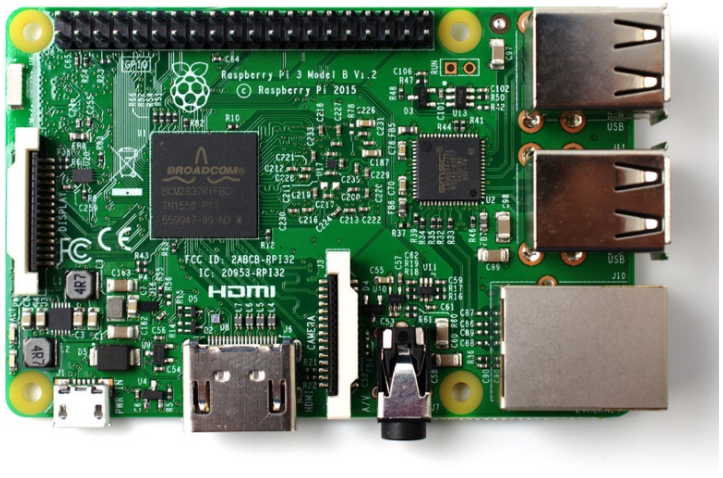
Assembling the hardware

Get the Raspberry



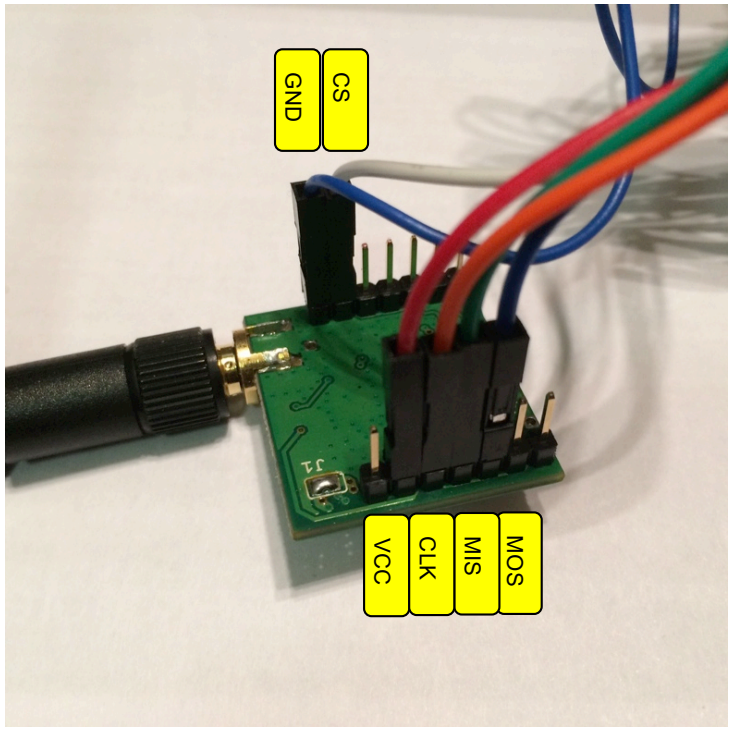


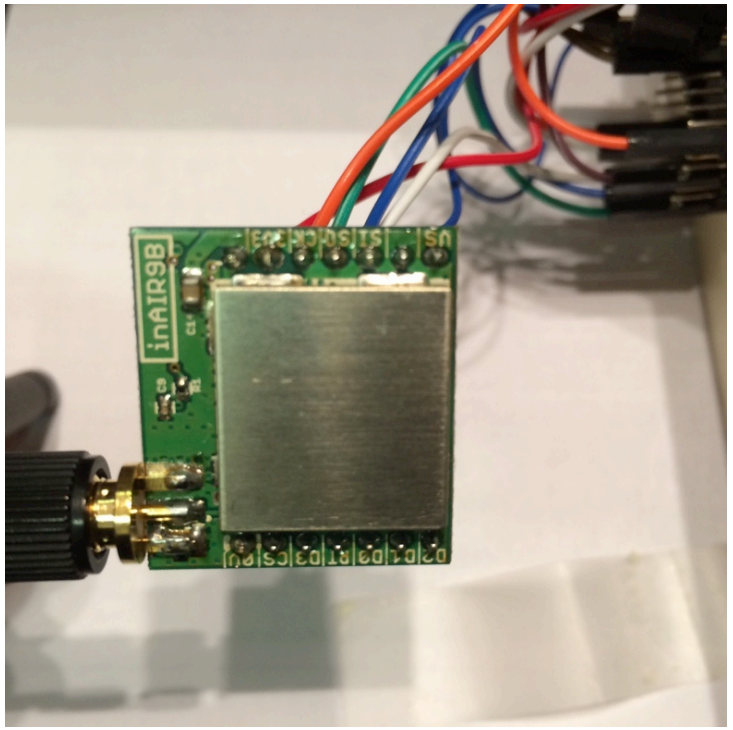




You can use Raspberry 1 model B or B+ or Raspberry 2 model B or Raspberry 3 model B. The most important usefull feature is the Ethernet interface for easy Internet connection. You can use WiFi to get Internet connection by adding a WiFi USB dongle. With the Raspberry 3, WiFi and Bluetooth are embedded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

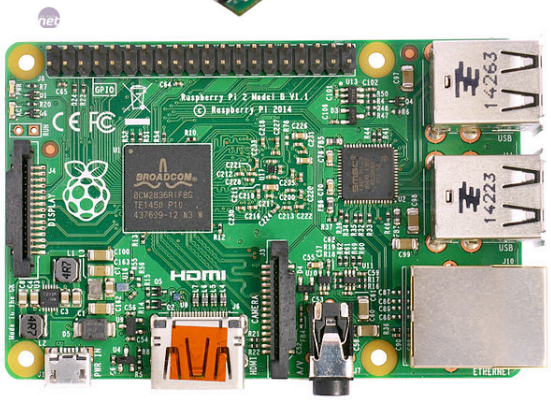
Now the radio module

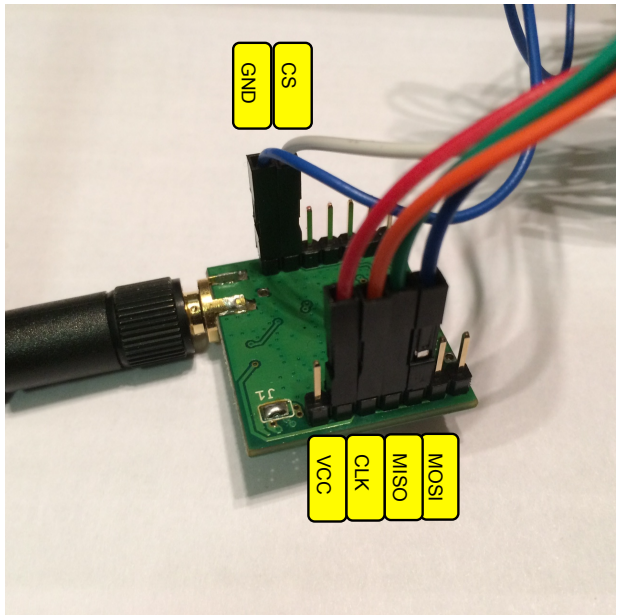




If you go for the inAir (4/9/9B) from Modtronix, then the header pins can come fully assembled. Take the 6mm header pins to have enough length to connect F/F breadboard cables (left). Connect the SPI pins with the F/F cables. Try to use different colors. I use the following colors: MOSI (blue), MISO (green), CS (white), CLK (orange). Then connect also the VCC (red) and the GND (black or any other dark color) of the radio board.

Connecting the radio module (1)

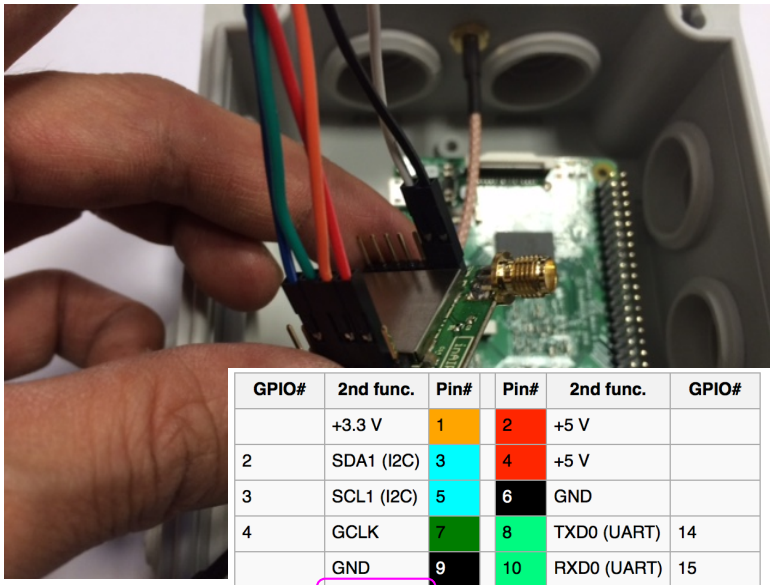




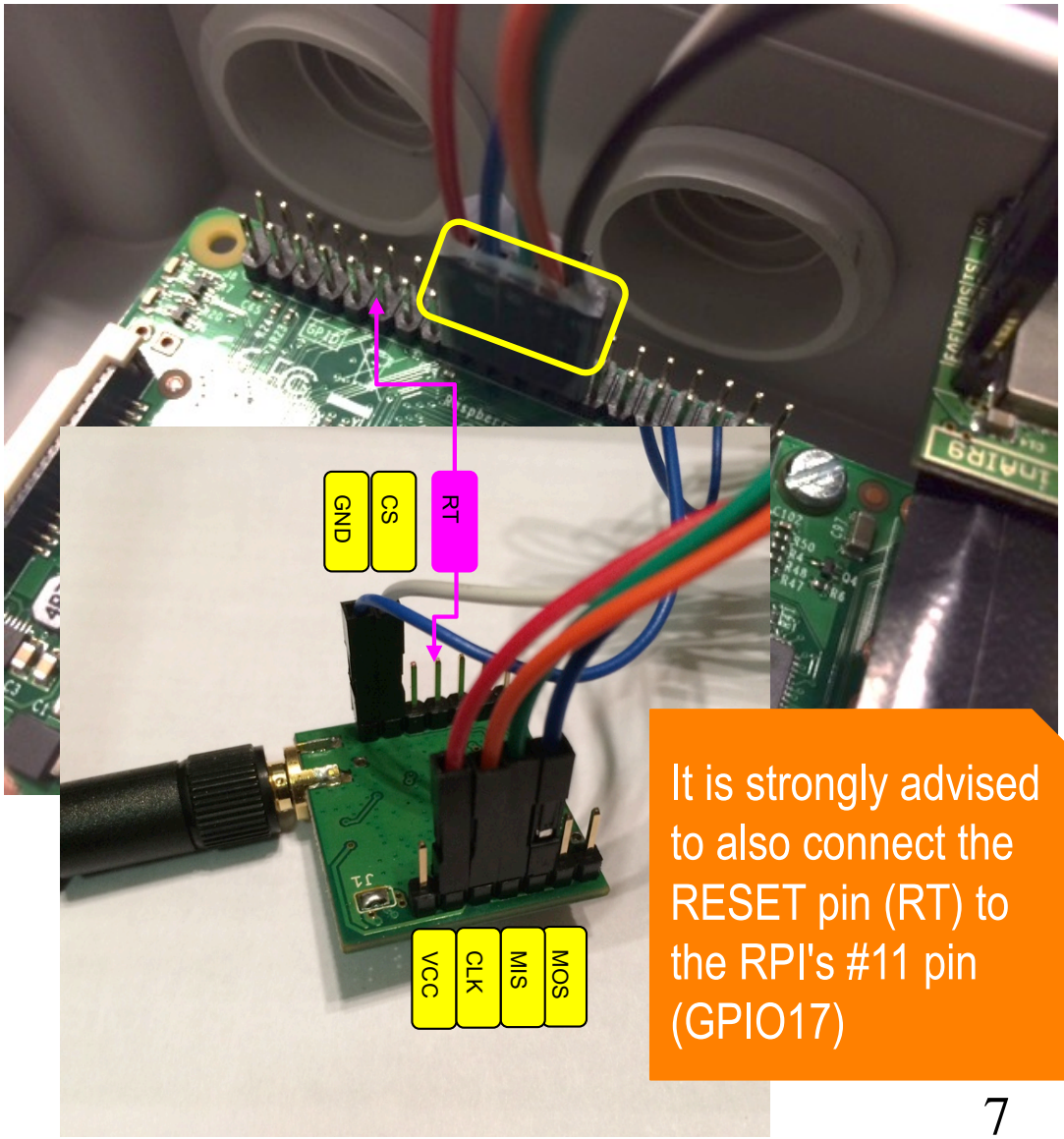
GPIO#	2nd func.	Pin#	Pin#	2nd func.	GPIO#
	+3.3 V	1	2	+5 V	
2	SDA1 (I2C)	3	4	+5 V	
3	SCL1 (I2C)	5	6	GND	
4	GCLK	7	8	TXD0 (UART)	14
	GND	9	10	RXD0 (UART)	15
17	GEN0	11	12	GEN1	18
27	GEN2	13	14	GND	
22	GEN3	15	16	GEN4	23
	+3.3 V	17	18	GEN5	24
10	MOSI (SPI)	19	20	GND	
9	MISO (SPI)	21	22	GEN6	25
11	SCLK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

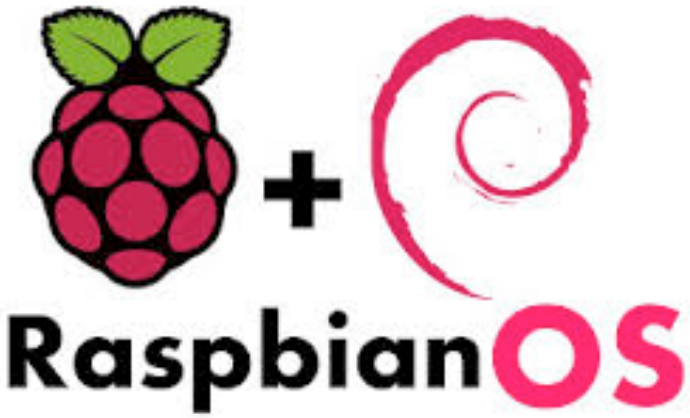


GPIO#	2nd func.	Pin#	Pin#	2nd func.	GPIO#
	+3.3 V	1	2	+5 V	
2	SDA1 (I2C)	3	4	+5 V	
3	SCL1 (I2C)	5	6	GND	
4	GCLK	7	8	TXD0 (UART)	14
	GND	9	10	RXD0 (UART)	15
17	GEN0	11	12	GEN1	18
27	GEN2	13	14	GND	
22	GEN3	15	16	GEN4	23
	+3.3 V	17	18	GEN5	24
10	MOSI (SPI)	19	20	GND	
9	MISO (SPI)	21	22	GEN6	25
11	SCLK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

Getting, compiling & installing the software



Flashing the OS

- An SD card image with a Raspberry Raspbian Jessie version is provided.
- You will need an 8GB SD card. Be careful, some SD cards will not work. This one has been successfully tested. It has to be class 10.

- Look at https://www.raspberrypi.org/documentation/installation/installing-images/ to see the procedure depending on your OS. 7948206080 bytes should be written, otherwise you may have a problem.
- Once flashed, insert the SD card and power-up the Raspberry-based gateway.

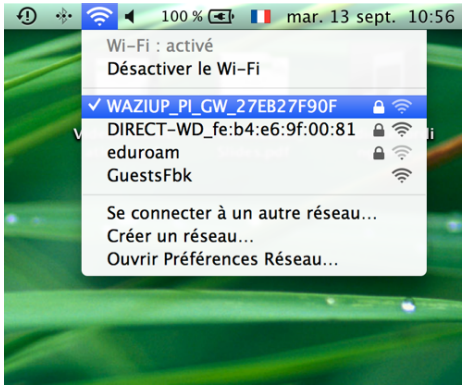
SSH TO THE GATEWAY

- The Raspbian image sets the Raspberry for DHCP on wired Ethernet and as a WiFi access point.
- If you connected the gateway to your LAN or laptop using wired Ethernet then the gateway will be assigned an IP address. Use this address to connect with SSH to the gateway
- You can use Angry IP Scanner (http://angryip.org/) to know the assigned address
- 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
- However, using the built-in WiFi access point is easier as shown in the next slide

SSH TO THE GATEWAY WITH WIFI

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

You can use an Android smartphone or tablet to connect to the gateway with the JuiceSSH app! See next slide.



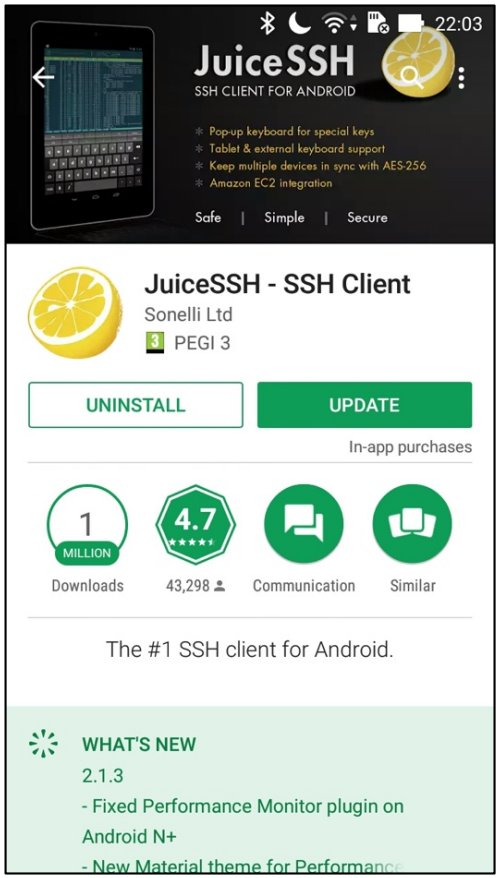

```
cpham — pi@raspberrypi: ~/lora_gateway —
MacBookProRetina-de-Congduc-Pham:~ cpham$ ssh pi@192.168.200.1
pi@192.168.200.1's password:

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
```

Using Android smartphone or tablet

- Use an Android smartphone or tablet with JuiceSSH to connect to the gateway



Gateway's simple command interface

- Once logged on the gateway, you may directly enter in a simple command interface
- This command interface consists in a `cmd.sh` shell script
- **In image versions after May 2017, this script is launched when you log into the gateway with ssh**
- If this happens, select Q and hit RETURN to quit this interface
- You should be in the `lora_gateway` folder

```
=======================================* Gateway 00000027EB5A71F7 *===
0- sudo python start_gw.py                                            +
1- sudo ./lora_gateway --mode 1                                       +
2- sudo ./lora_gateway --mode 1 | python post_processing_gw.py        +
3- ps aux | grep -e start_gw -e lora_gateway -e post_proc -e log_gw   +
4- tail --line=25 ../Dropbox/LoRa-test/post-processing.log            +
5- tail --line=25 -f ../Dropbox/LoRa-test/post-processing.log         +
6- less ../Dropbox/LoRa-test/post-processing.log                      +
-------------------------------------------------* Bluetooth *--+
a- run: sudo hciconfig hci0 piscan                                    +
b- run: sudo python rfcomm-server.py                                  +
c- run: nohup sudo python rfcomm-server.py -bg > rfcomm.log &         +
d- run: ps aux | grep rfcomm                                          +
e- run: tail -f rfcomm.log                                            +
-------------------------------------------------* Connectivity *--+
f- test: ping www.univ-pau.fr                                         +
-------------------------------------------------* Filtering msg *--+
l- List LoRa reception indications                                    +
m- List radio module reset indications                                +
n- List boot indications                                              +
o- List post-processing status                                        +
p- List low-level gateway status                                      +
-------------------------------------------------* Configuration *--+
A- show gateway_conf.json                                             +
B- edit gateway_conf.json                                             +
C- show clouds.json                                                   +
D- edit clouds.json                                                   +
-----------------------------------------------------* Update *--+
U- update to latest version on repository                             +
V- download and install a file                                        +
W- run a command                                                      +
-----------------------------------------------------* kill *--+
K- kill all gateway related processes                                 +
k- kill rfcomm-server process                                         +
R- reboot gateway                                                     +
S- shutdown gateway                                                   +
-----------------------------------------------------------------------+
Q- quit                                                               +
=======================================================================
Enter your choice:

pi@raspberrypi:~/lora_gateway $
```

Default gateway configuration

- The gateway software is **launched** when the Raspberry is powered on
  - `/home/pi/lora_gateway/scripts/start_gw.sh` has been added in `/etc/rc.local`
- The gateway works by default in LoRa mode 1 (BW=125kHz, SF=12) and listen on frequency 433.3MHz (see https://github.com/CongducPham/WaterSense/tree/master/WaterSenseGateway#annexa-lora-mode-and-predefined-channels
- The `gateway_conf.json` file contains the gateway configuration

GATEWAY_CONF.JSON

- The `gateway_conf.json` file contains the gateway configuration

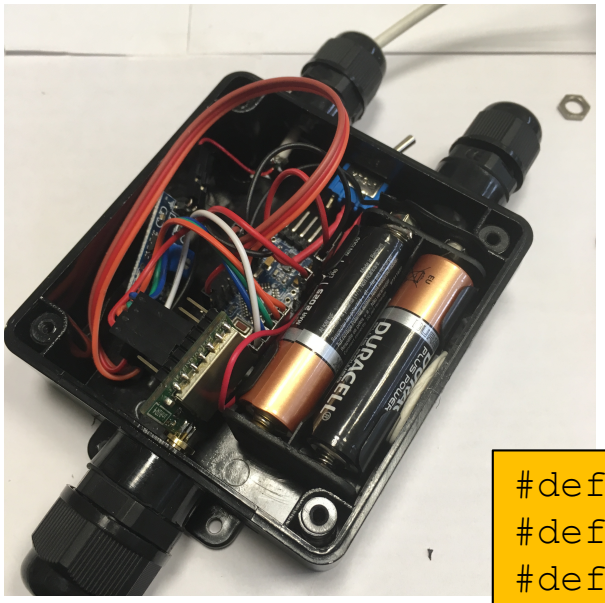
```
{
    "radio_conf" : {
        "mode" : 1,
        "bw" : 500,
        "cr" : 5,
        "sf" : 12,
        "ch" : -1,
        "freq" : -1
    },
    "gateway_conf" : {
        "gateway_ID" : "000000XXXXXXXXXX",
        "ref_latitude" : "my_lat",
        "ref_longitude" : "my_long",
        "wappkey" : false,
        "raw" : false,
        "aes" : false,
        "log_post_processing" : true,
        "log_weekly" : false,
        "dht22" : 0,
        "dht22_mongo": false,
        "downlink" : 0,
        "status" : 600,
        "aux_radio" : 0
    },
```

Set "mode" to -1 if you want to use bw, cr and sf parameters

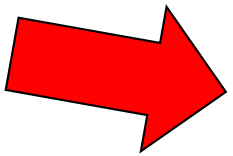
Set "ch" to a channel number if you don't want to use the default channel (which is channel 10 in the 868 band channel 05 in 900 band and channel 00 in the 433 band).

Set "freq" to a frequency, e.g. 433.6, if you want to specify a given frequency, "freq" has priority over "ch"

Default configuration



\!##SM1/530/SM2/756



```
#define DEFAULT_DEST_ADDR 1
#define LORAMODE  1
#define node_addr 2
```

The default configuration in the Arduino_LoRa_SoilHum example is:

- Send packets to the gateway (one or many if in range)
- LoRa mode 1 & Node short address is 2

The default gateway configuration is also LoRa mode 1.

Both are working is 433.3MHz band for Pakistan.

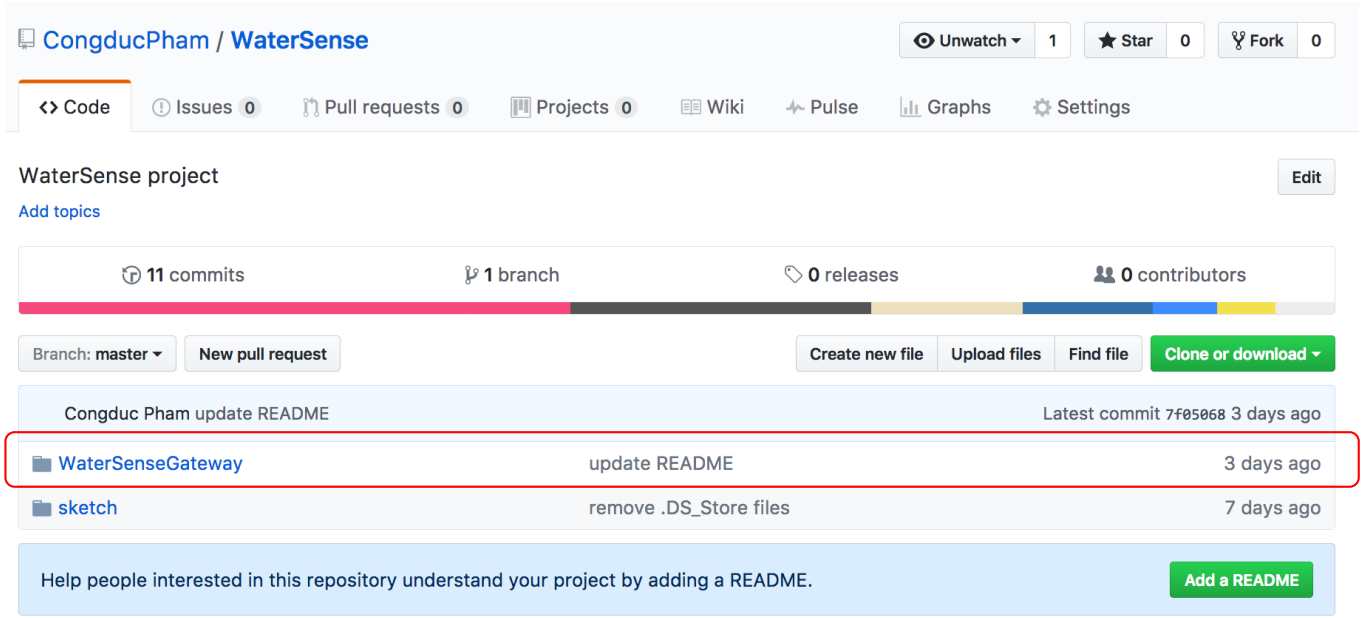
Check that the gateway is running

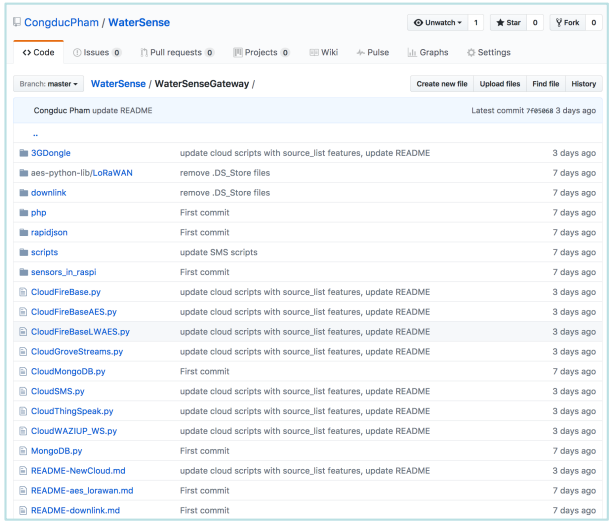
- Use option 3 of the text interface

```
BEGIN OUTPUT
Check for lora_gateway process
################################
root      4119  0.0  0.3   6780  3184 ?        S    10:21   0:00 sudo python start_gw.py
root      4123  0.0  0.5   9228  5180 ?        S    10:21   0:00 python start_gw.py
root      4124  0.0  0.0   1912   364 ?        S    10:21   0:00 sh -c sudo ./lora_gateway --mode 1 --ndl | python p
root      4125  0.0  0.3   6780  3188 ?        S    10:21   0:00 sudo ./lora_gateway --mode 1 --ndl
root      4131 88.5  0.2   3700  2176 ?        R    10:21   3:31 ./lora_gateway --mode 1 --ndl
pi        4176  0.0  0.2   4276  1948 pts/1    S+   10:25   0:00 grep -e start_gw -e lora_gateway -e post_processing
################################
The gateway is running if you see the lora_gateway process
END OUTPUT
Press RETURN/ENTER...
```

- **IMPORTANT NOTICE**: Do not launch a new gateway instance with an existing one as there will be conflict on the SPI bus.

Full update of GW software from github





```
> mkdir lora_gateway
> git clone https://github.com/CongducPham/WaterSense.git
> cp WaterSense/WaterSenseGateway/* 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/WaterSense (right) then copy all the files & folder of the github's **WaterSenseGateway** 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
```

Edit radio.makefile for PABOOST setting. If inAir9B, RFM92W/FM95W, NiceRF1272, uncomment:

`CFLAGS=-DPABOOST`

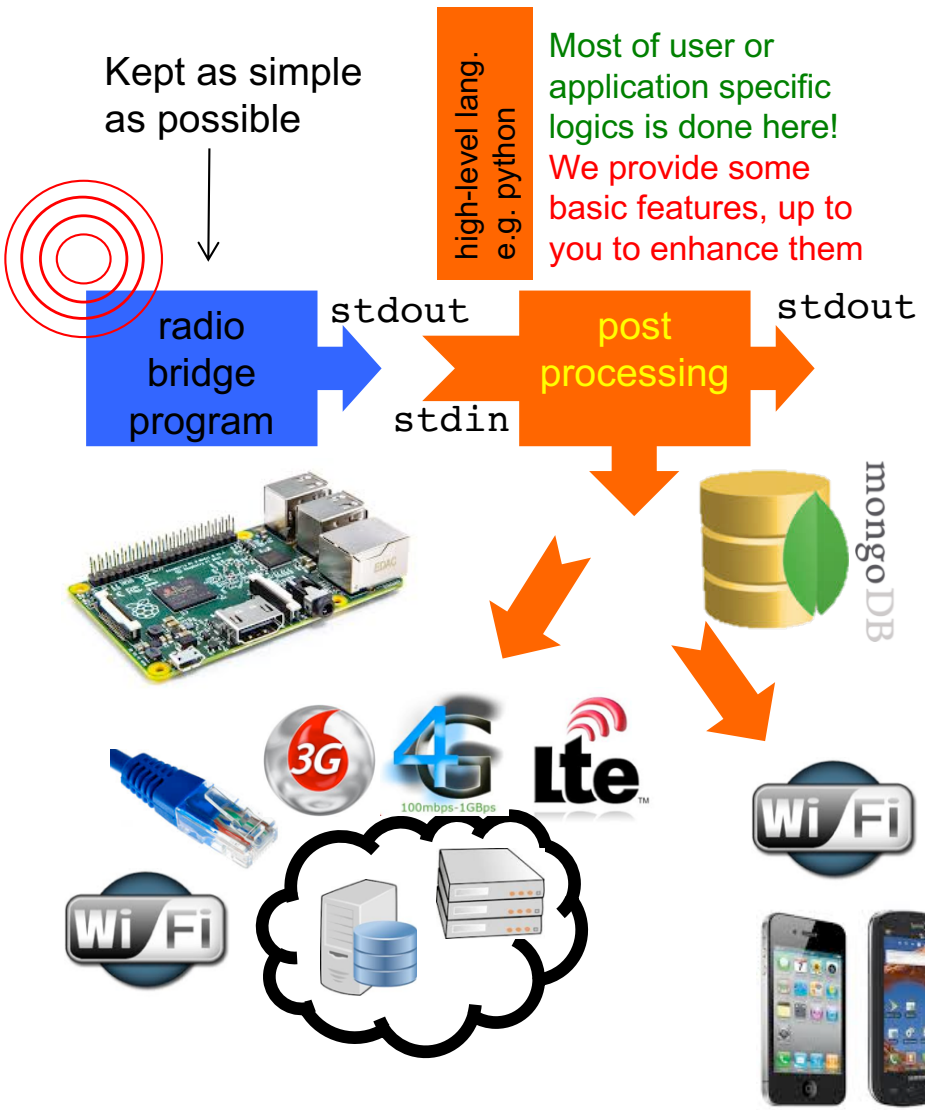
If inAir9/inAir4, Libelium SX1272, leave commented:

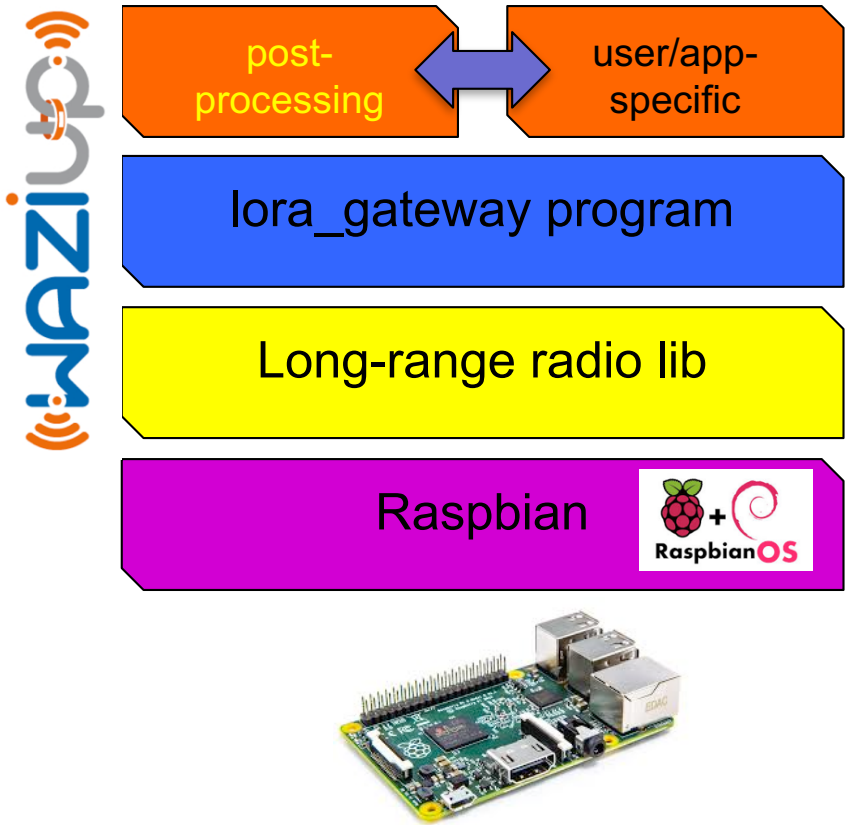
`#CFLAGS=-DPABOOST`

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

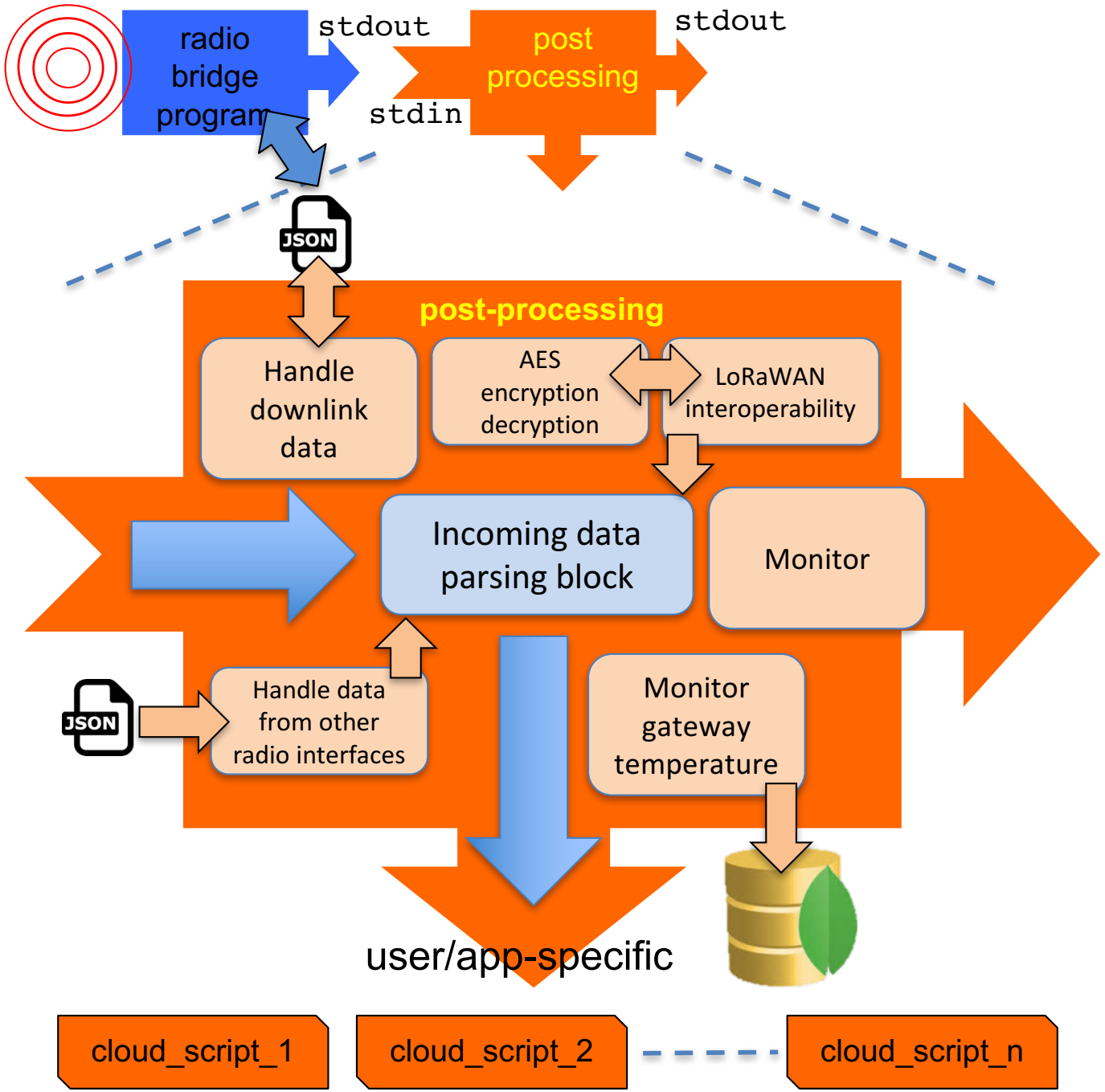
```
>make lora_gateway_pi2
```

Our Low-Cost gateway architecture





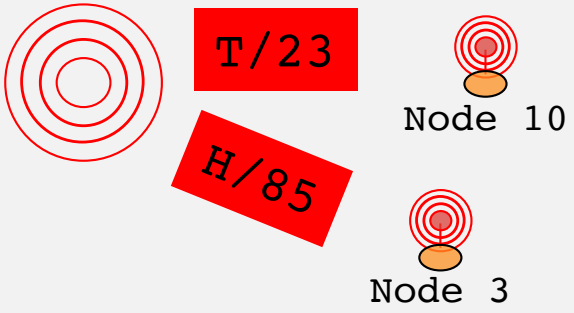
POST-PROCESSING BLOCK



Starting the gateway

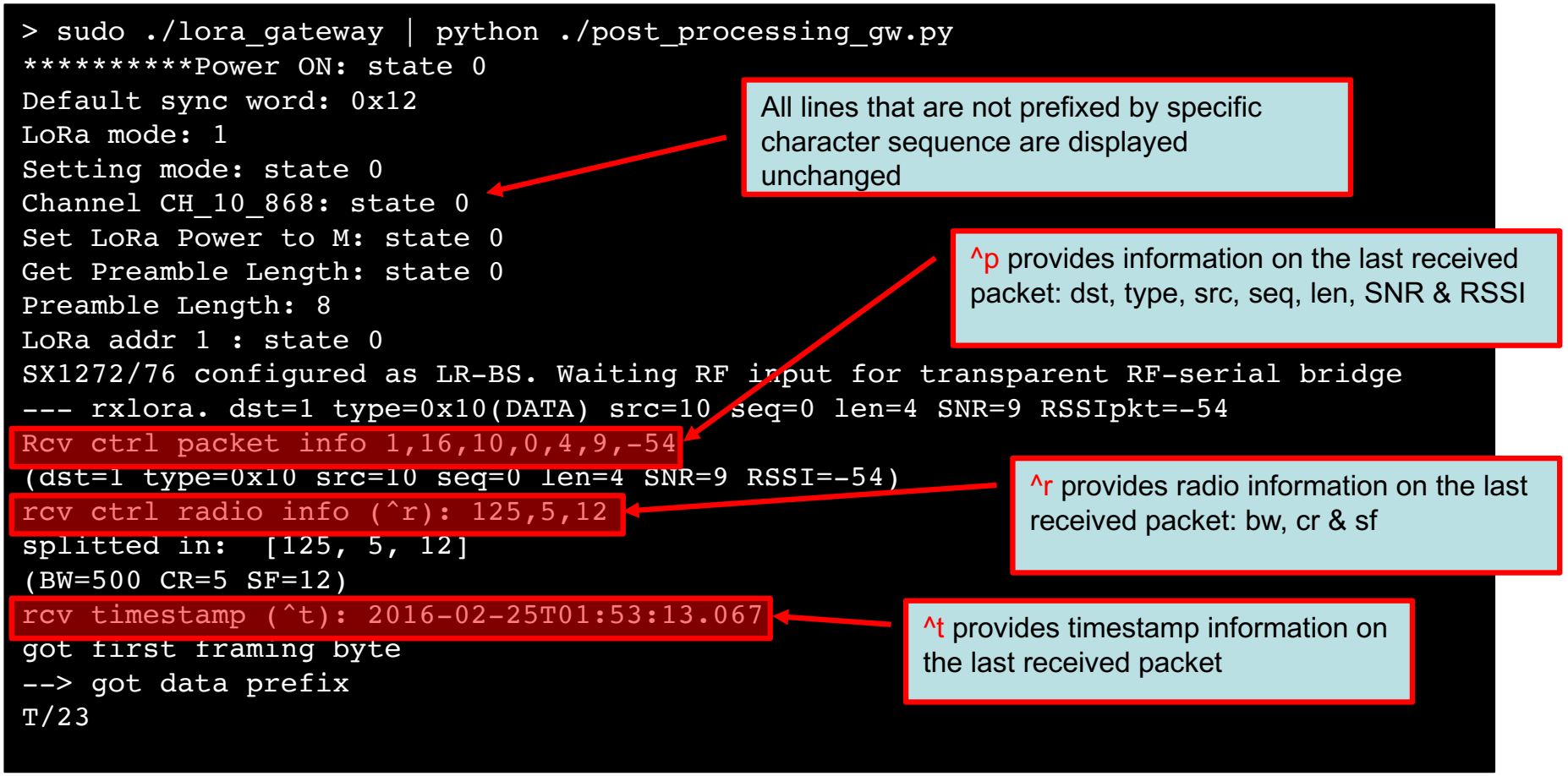
- Remember that the gateway software is **launched** when the Raspberry is powered on
- In the next 4 slides, we will show you some details that is useful to know but that you **DO NOT** need when launching a production gateway
- If you use our SD card image and powered on your Raspberry, then you can use option 3 as explained in slide 16 to see if the gateway is running, then use option K to kill all gateway-related process to test the next 4 slides.

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



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

Log received messages using cloud services

\!T/23

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
```

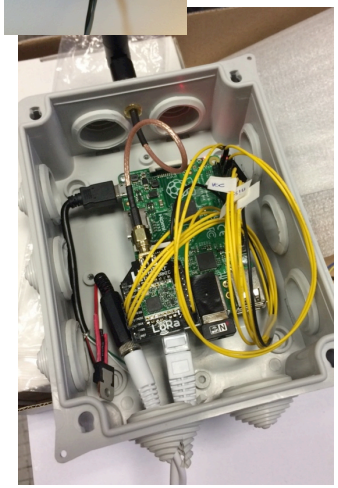
\$ or \! before the data indicates that the data should be logged on a file or a cloud. It is up to the end-device to decide which option

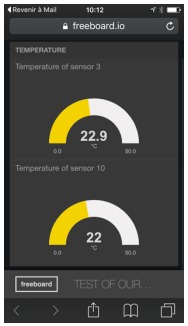
Starting the complete gateway

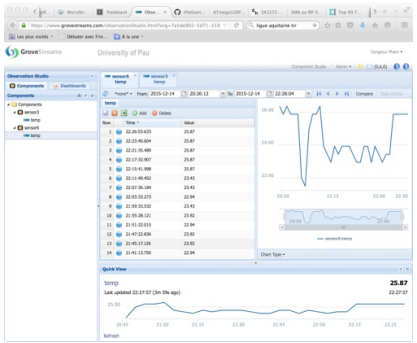
- The complete gateway also logs all output to a post-processing.log file. Use `sudo python start_gw.py`

```
> cd lora_gateway
> sudo python start_gw.py
sudo ./lora_gateway --mode 1 | python post_processing_gw.py | python log_gw.py
Starting thread to report gw status
2017-09-01 12:08:11.751649
post status: gw ON, lat my_lat long my_long
Current working directory: /home/pi/lora_gateway
SX1276 detected, starting.SX1276 LF/HF calibration...
**********Power ON: state 0
Default sync word: 0x12
LoRa mode 1
Setting mode: state 0
Channel CH_10_868: state 0
Set LoRa power dBm to 14
Power: 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
```

Gateway to cloud





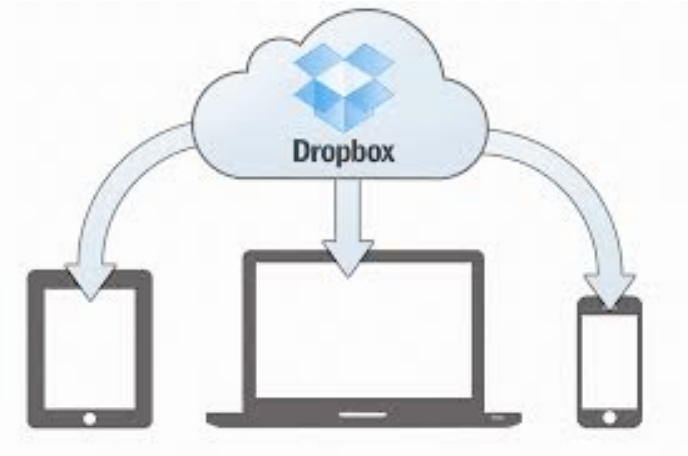


Data received at the gateway can be pushed to IoT clouds. We provide python script examples for many IoT cloud platforms. Most of clouds with REST API can be easily integrated.

Using Dropbox

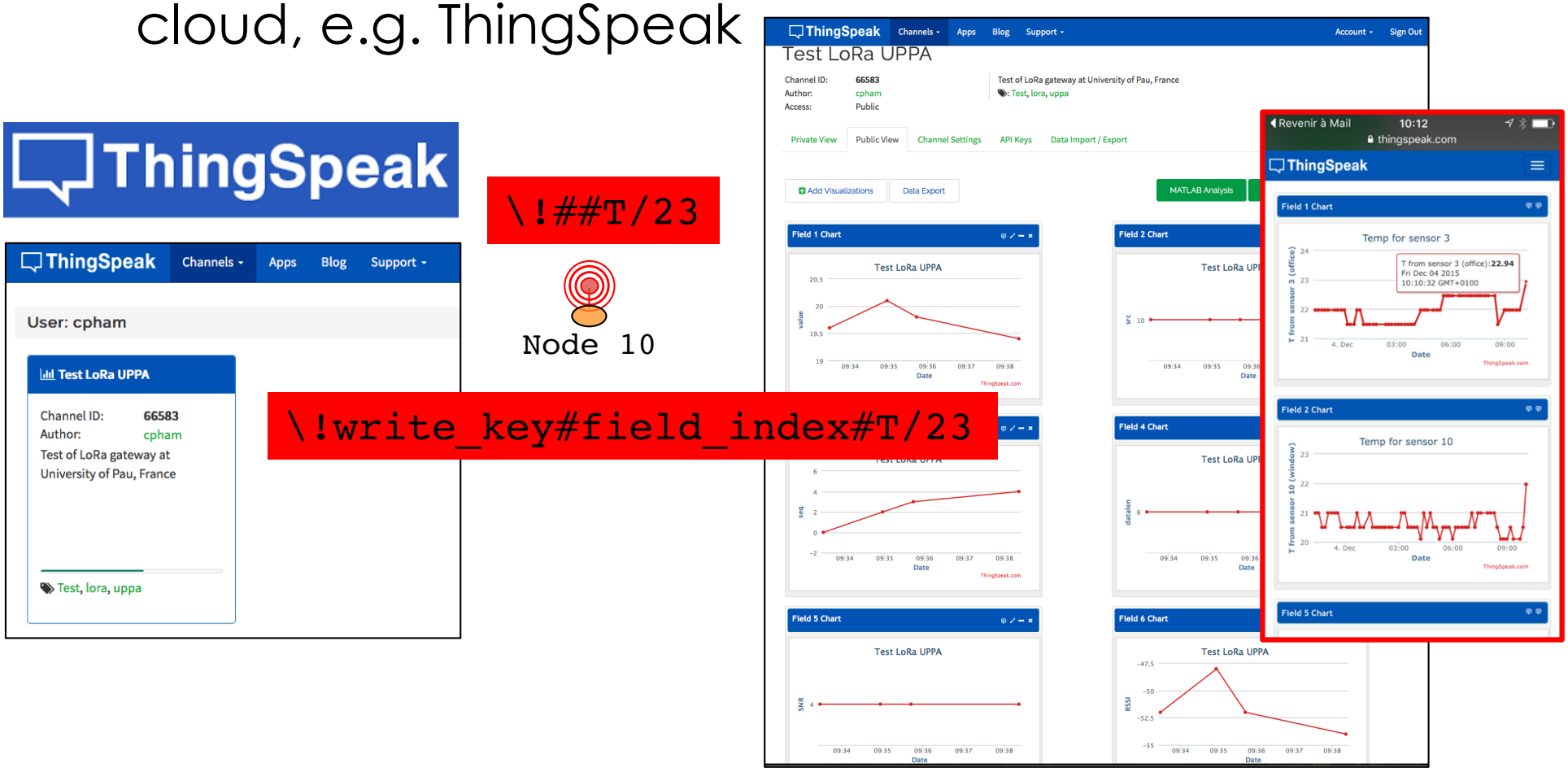
- A message starting with '\$' is logged in a file 'telemetry.log' in a folder shared through Dropbox

```
(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
...
```



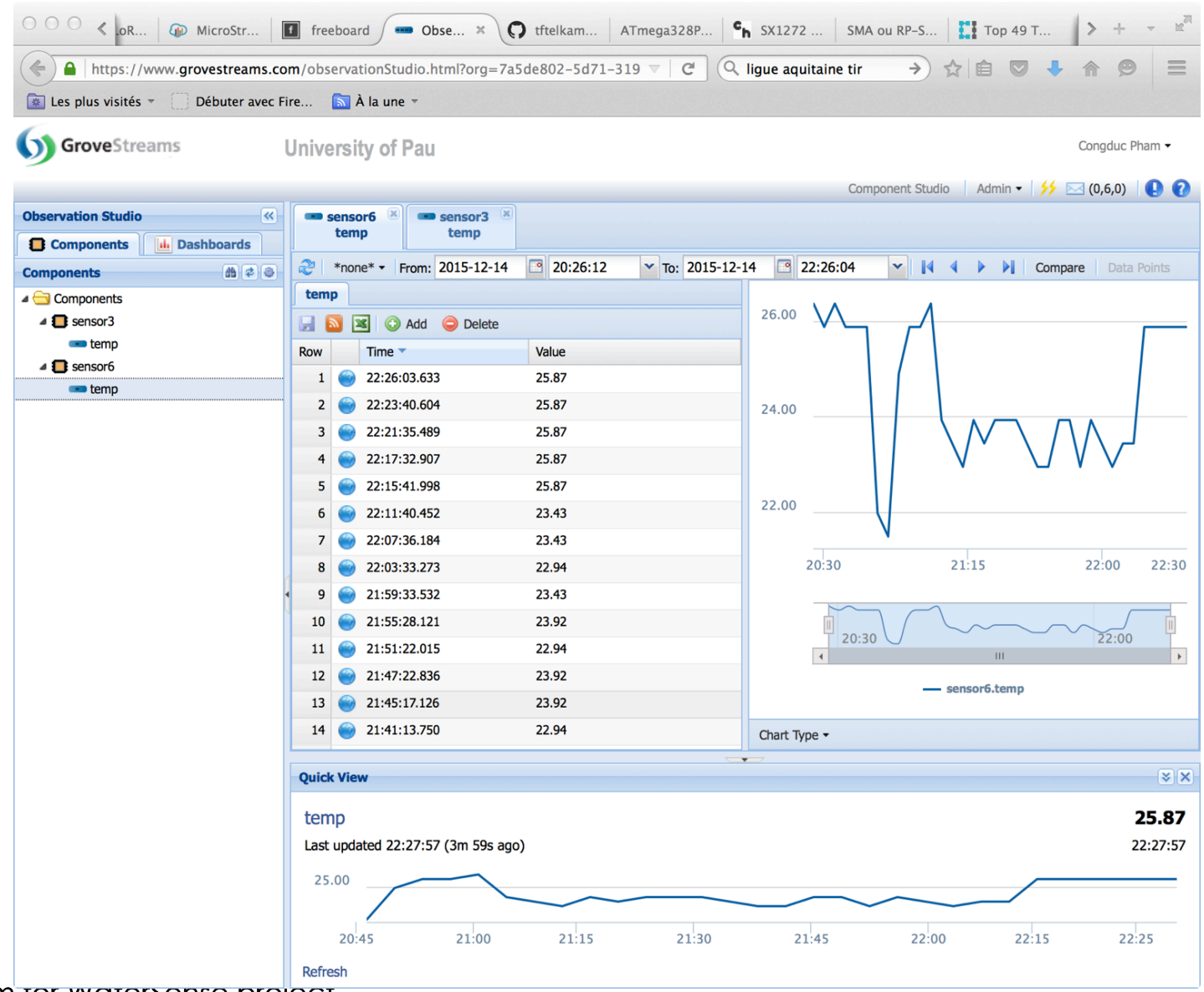
USING ThingSpeak

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



Using Firebase

Using GroveStreams



Gateway configuration

- gateway_conf.json

```
{
  "radio_conf": {
    "mode": 1,
    "bw": 500,
    "cr": 5,
    "sf": 12,
    "ch": -1,
    "freq": 433.3
  },
  "gateway_conf": {
    "gateway_ID": "000000XXXXXXXXXX",
    "ref_latitude": "my_lat",
    "ref_longitude": "my_long",
    "wappkey": false,
    "raw": false,
    "aes": false,
    "log_post_processing": true,
    "log_weekly": false,
    "dht22": 0,
    "dht22_mongo": false,
    "downlink": 0,
    "status": 600,
    "aux_radio": 0
  },
  "alert_conf": {
    "use_mail": false,
    "contact_mail": "the_contact_mail_address_anot
    "mail_from": "a_gmail_address",
    "mail_server": "smtp.gmail.com",
    "mail_passwd": "your_passwd",
    "use_sms": false,
    "pin": "0000",
    "contact_sms": [
      "+33XXXXXXXXX",
      "+33XXXXXXXXX"
    ],
    "gammurc_file": "/home/pi/.gammurc"
  }
}
```

radio_conf section defines the LoRa parameters

gateway_conf section defines additional options for the gateway's post-processing stage

alert_conf section defines alerting methods

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": true
    },
    {
      "name": "WAZIUP Orion cloud",
      "script": "python CloudOrion.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": false
    },
    {
      "name": "Firebase cloud",
      "script": "python CloudFireBase.py",
      "type": "jsoncloud",
      "write_key": "",
      "enabled": false
    },
    {
      "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, CloudOrion.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.

Rebooting the gateway

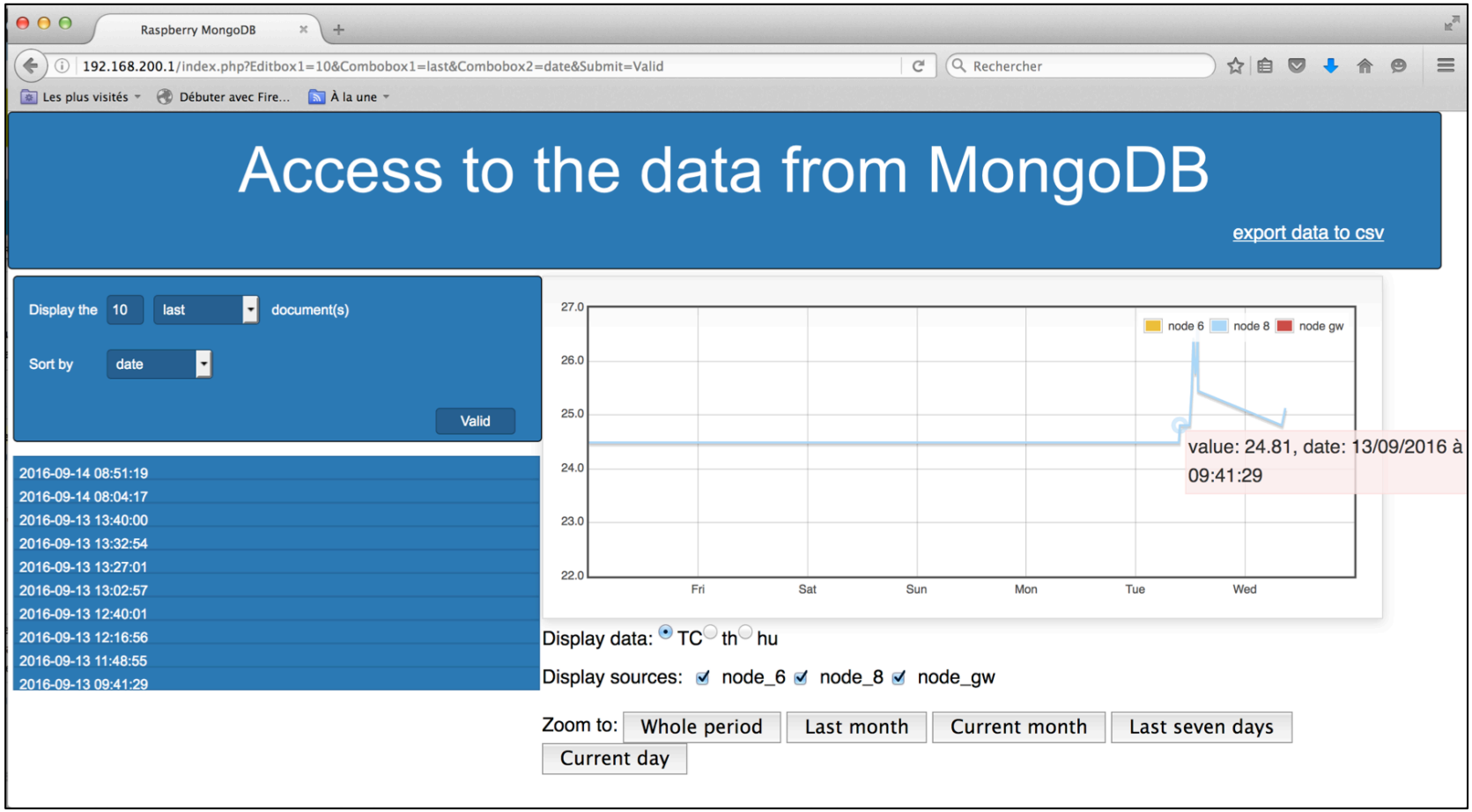
- Your gateway is now updated and configured
- You can now reboot the gateway. To do so, just type:
  - `sudo shutdown -r now`
- Or run `./cmd.sh` and chose option **R**
- Once the gateway has rebooted, check the WiFi SSID which now should meet your gateway's id
- Try to avoid unplugging power cable to shutdown your gateway. Log into the gateway and select option **S** instead.
- Your gateway is now ready to be deployed

Standalone gateway

Connect to the embedded web server

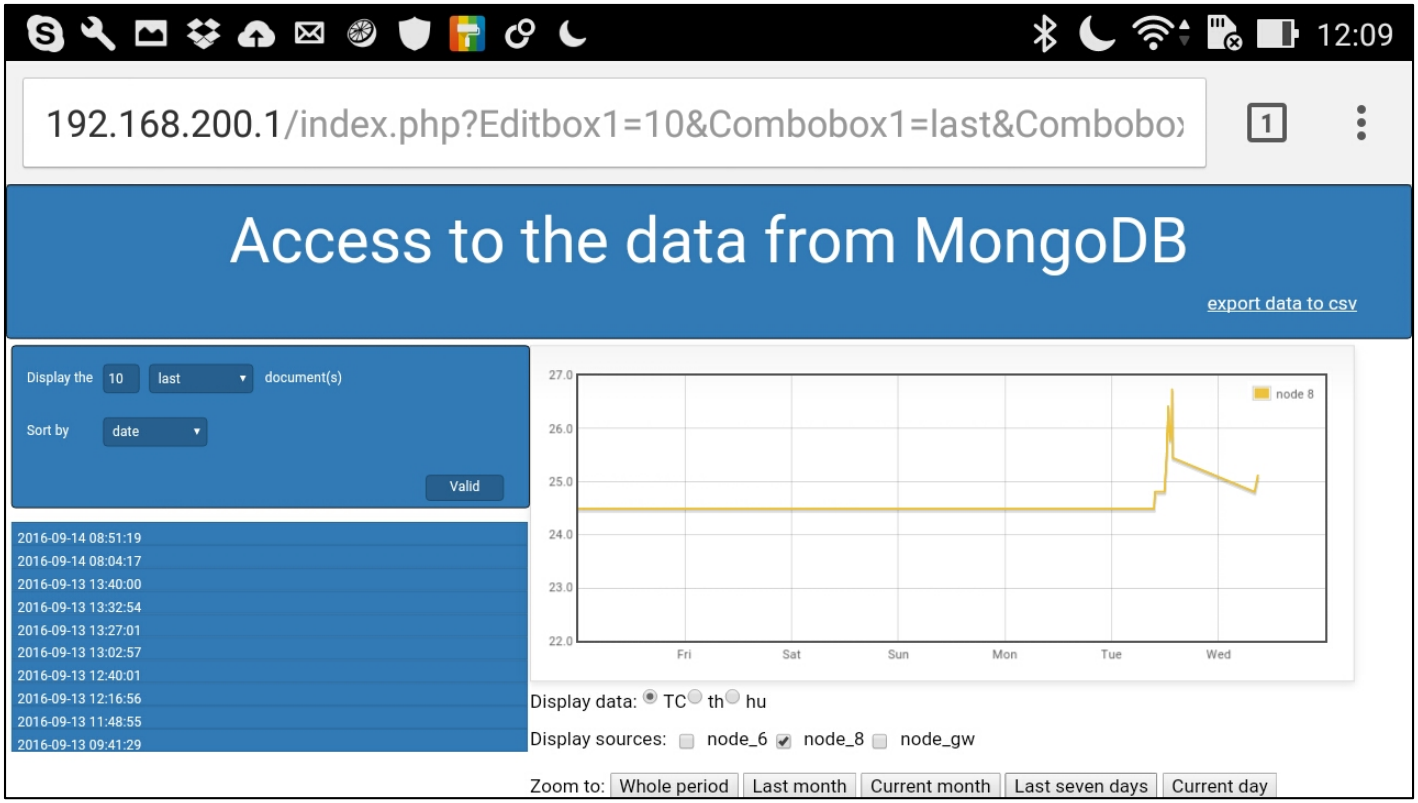
- On the WiFi interface
  - Gateway address is 192.168.200.1
- On the Ethernet interface
  - Gateway address is the IP address assigned by the DHCP server (of your LAN or laptop)
- Choose any of these solutions and open a web browser to enter the gateway IP address in the URL bar
  - http://192.168.200.1

Data from the local web server

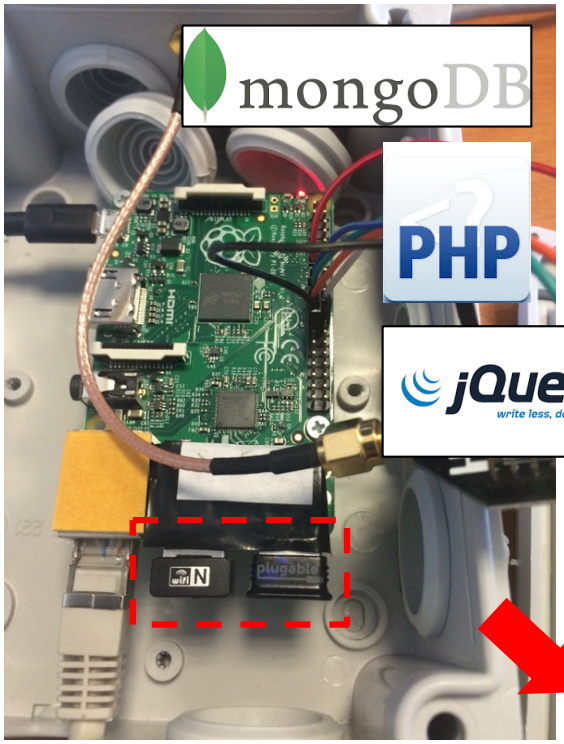


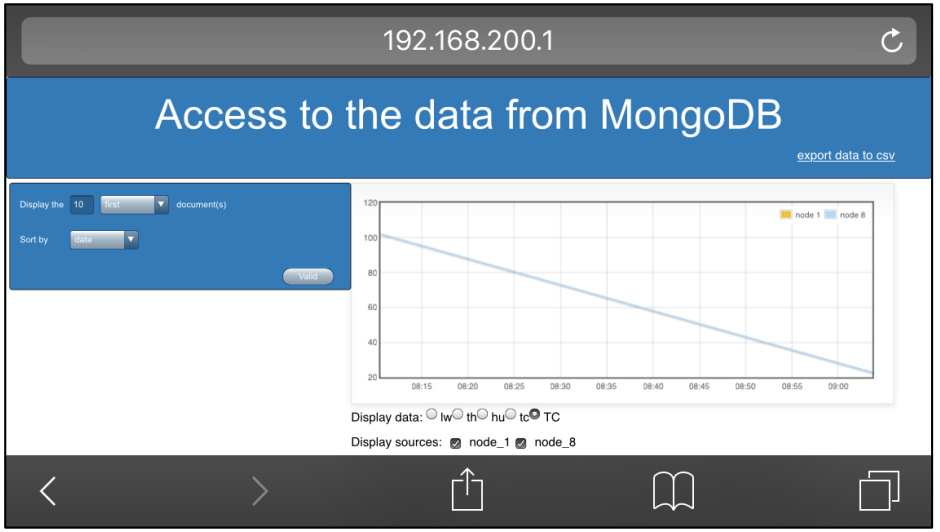
Visualize it on your smartphone!

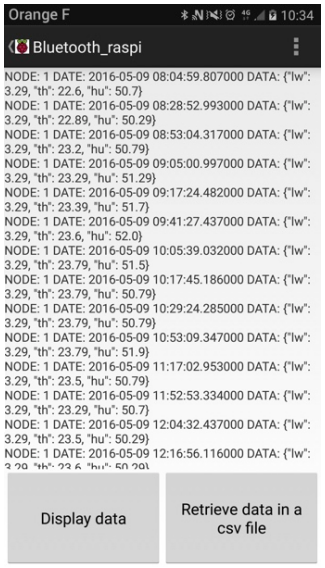
- Don't forget to join the WS_PI_GW_xxxxxxxxxx WiFi

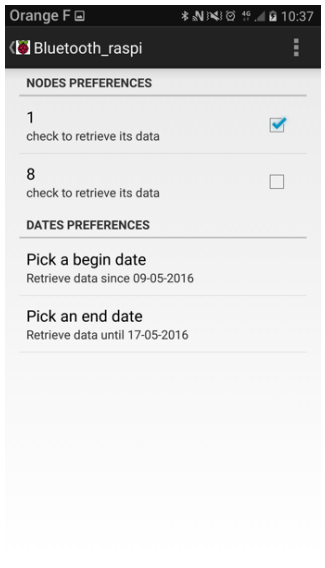


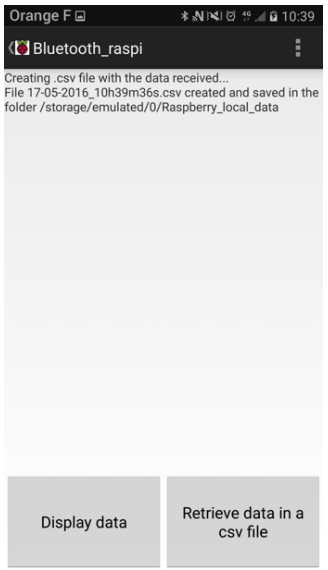
Running the gateway without internet access



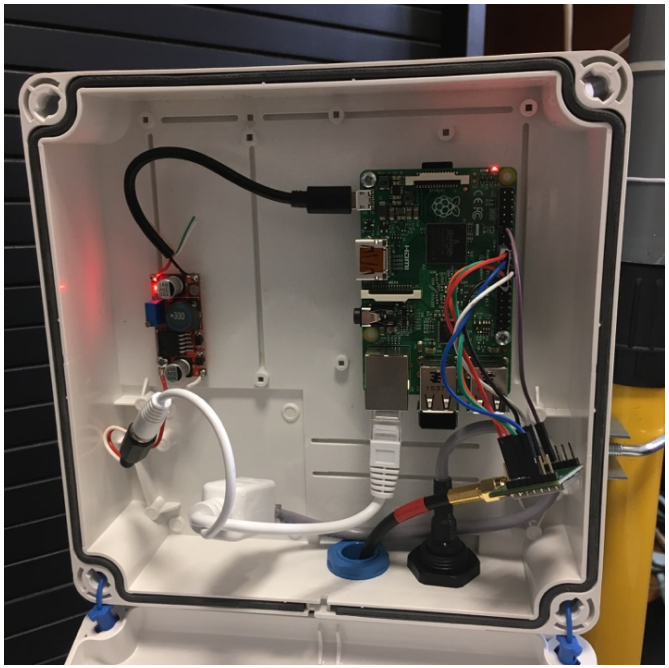








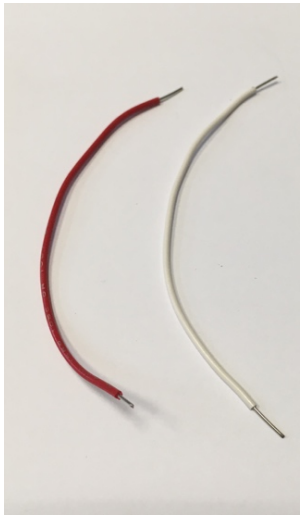
Improving casing and adding PoE to gateway

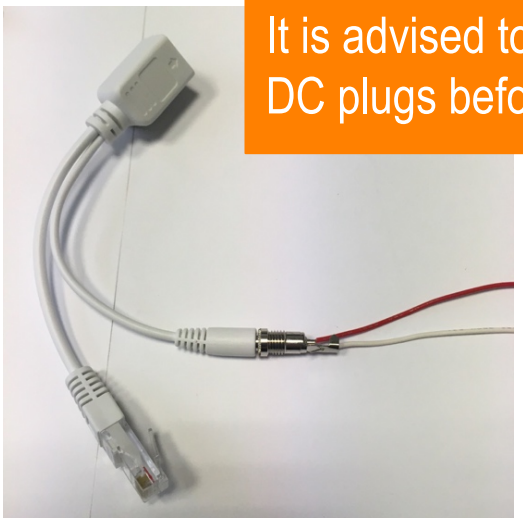


Overview of the parts

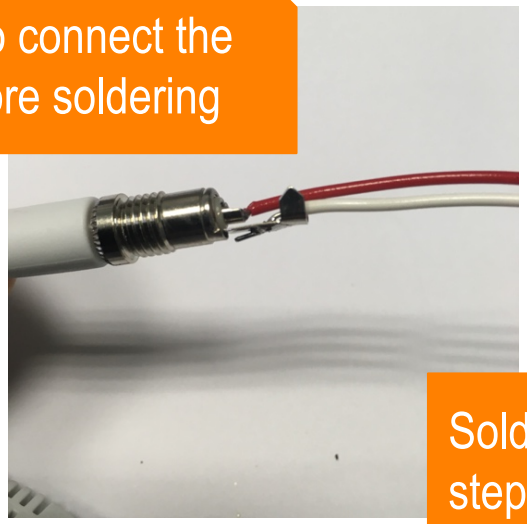
Fixing the Raspberry to the case

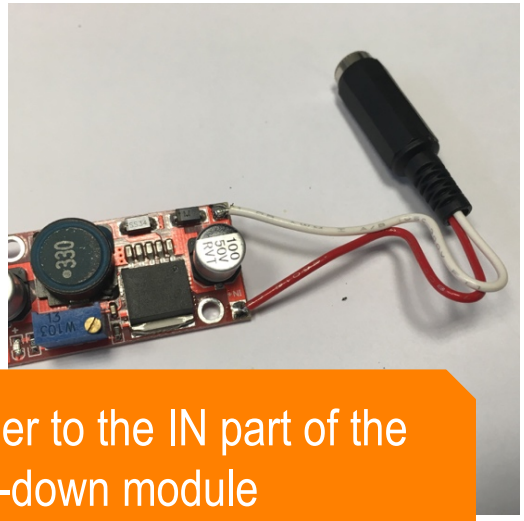
Prepare the DC Step-Down (LM2596)





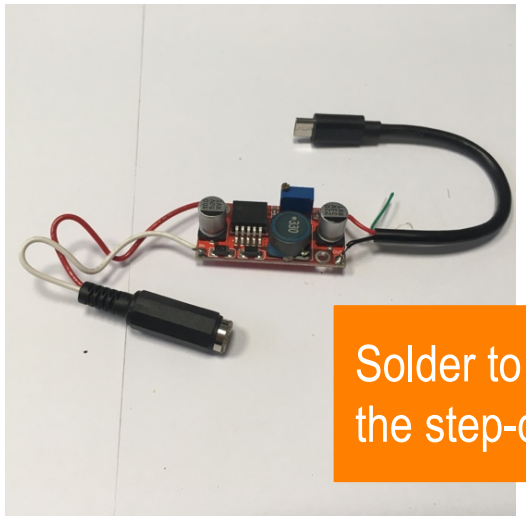
It is advised to connect the DC plugs before soldering





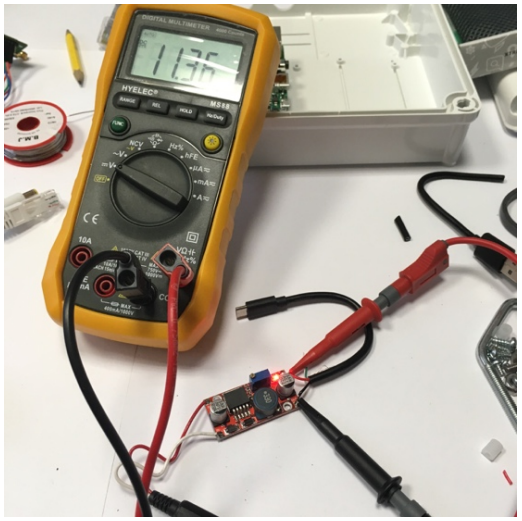
Solder to the IN part of the step-down module

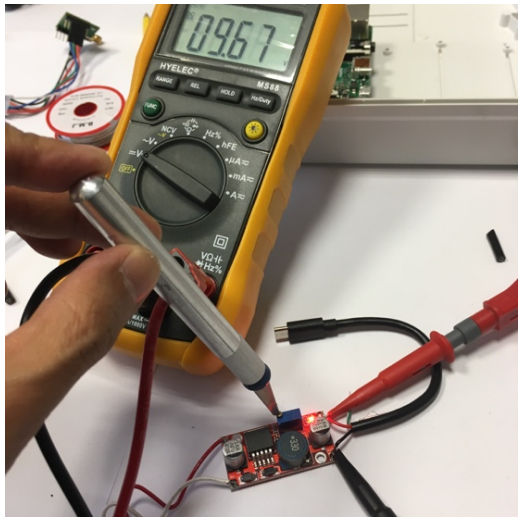
Cut a USB cable, keeping the micro-USB side

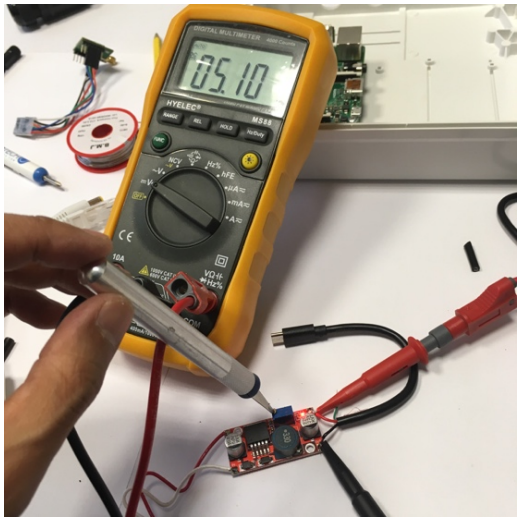


Solder to the OUT part of the step-down module

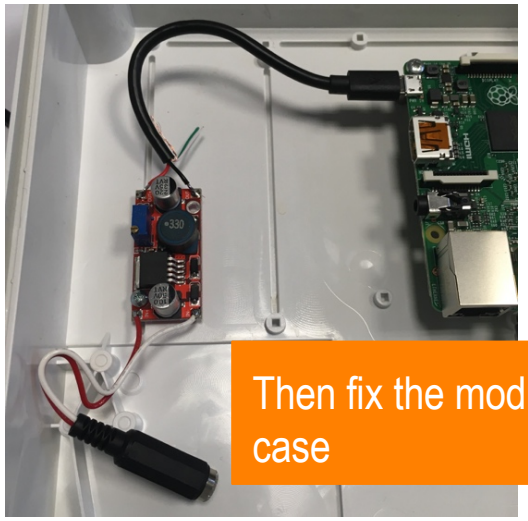
Setting the step-down module





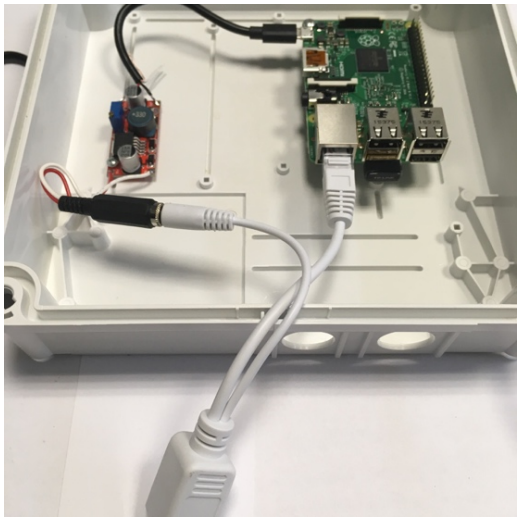


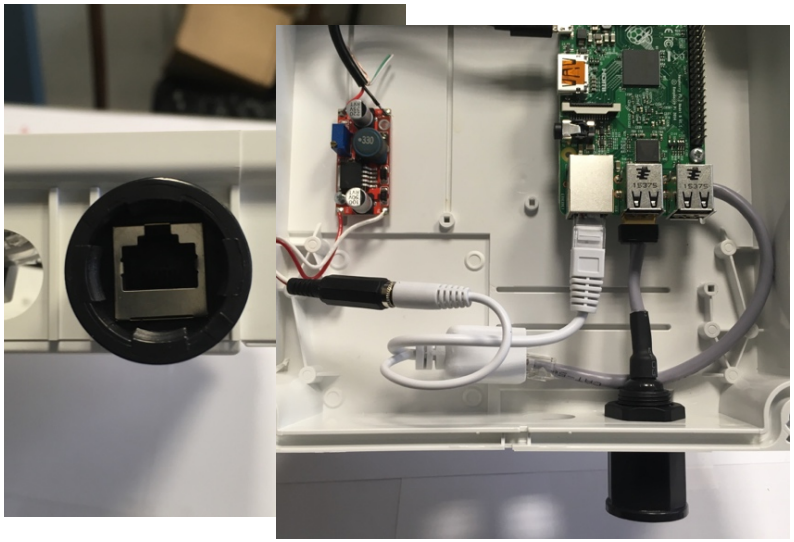
Use for instance a 9v, 12v or 18V AC-DC adaptor, connect to the IN plug, then check the output voltage with a voltmeter and turn the regulation screw until output is about 5.1v.

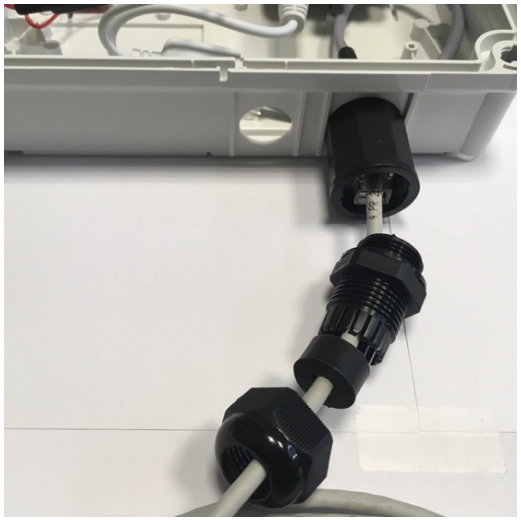
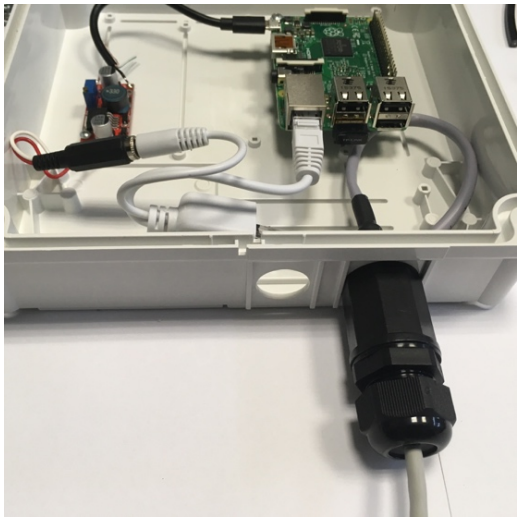


Then fix the module to the case

Installing the PoE injector and water-resistant ethernet plug

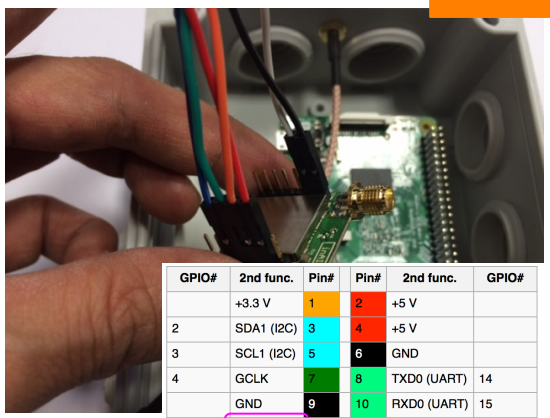




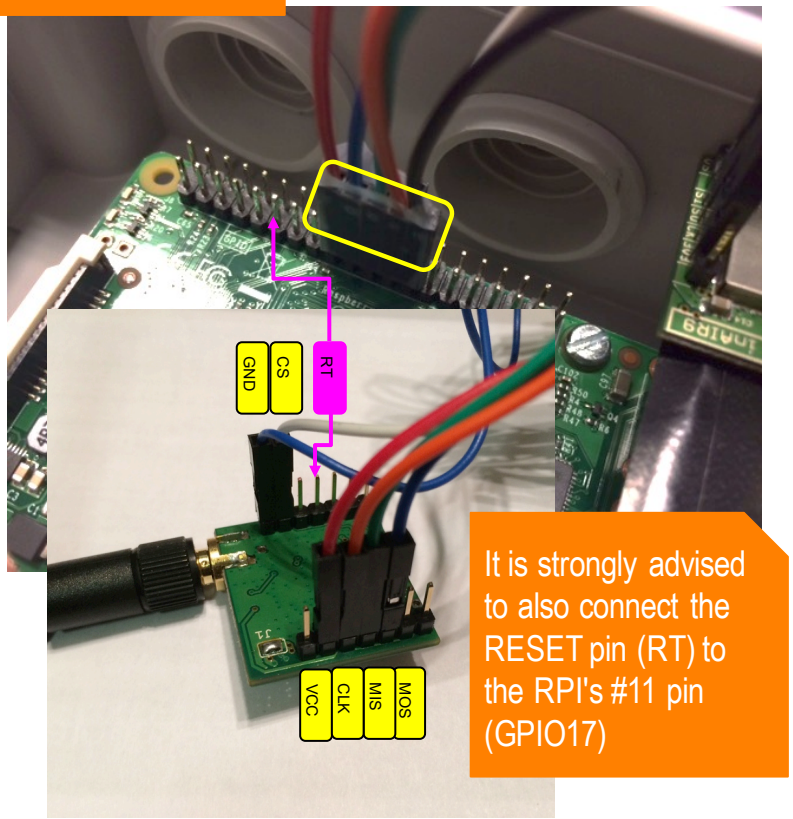



Connect the radio module

Like previously shown

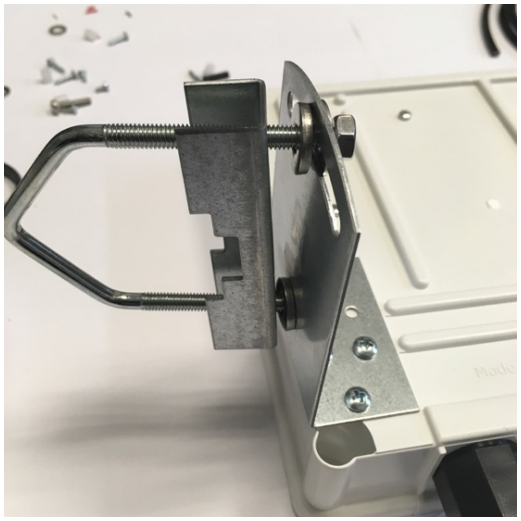


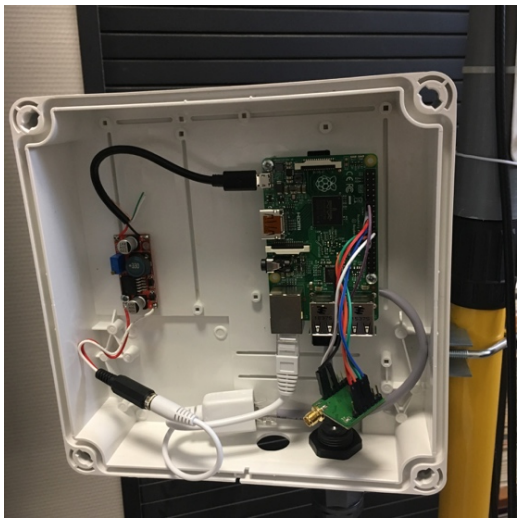
GPIO#	2nd func.	Pin#	Pin#	2nd func.	GPIO#
	+3.3 V	1	2	+5 V	
2	SDA1 (I2C)	3	4	+5 V	
3	SCL1 (I2C)	5	6	GND	
4	GCLK	7	8	TXD0 (UART)	14
	GND	9	10	RXD0 (UART)	15
17	GEN0	11	12	GEN1	18
27	GEN2	13	14	GND	
22	GEN3	15	16	GEN4	23
	+3.3 V	17	18	GEN5	24
10	MOSI (SPI)	19	20	GND	
9	MISO (SPI)	21	22	GEN6	25
11	SCLK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

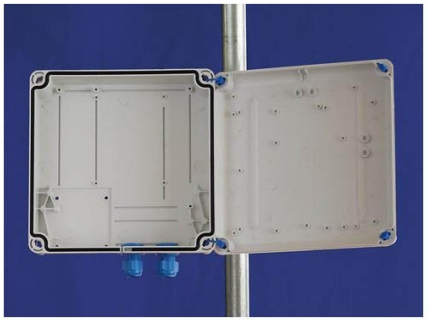
Install fixing parts of the case

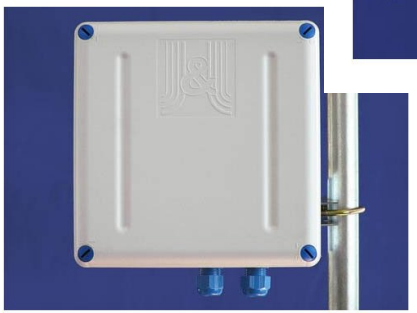




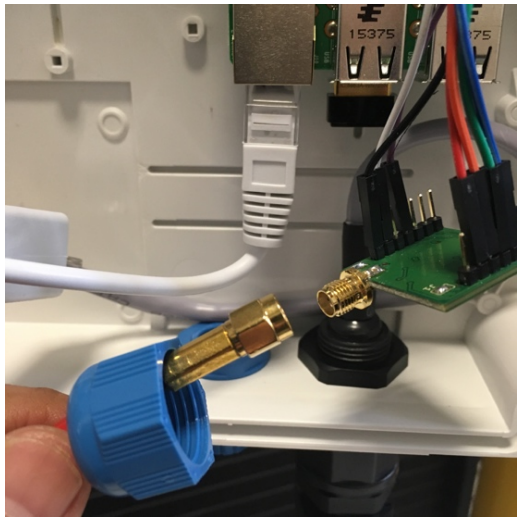
These parts of course depends on the case that you have.

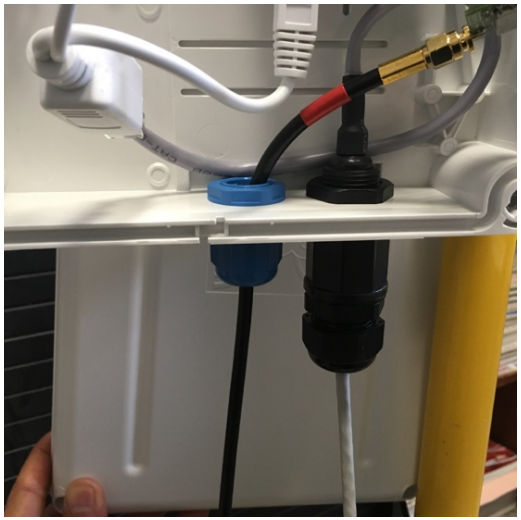
Here we use the GentleBOX JE-200 case from MHzShop.





Fixing the antenna cable





Look at the Antenna tutorial to see how an antenna cable can be made to adapt both the cable length and the antenna connectors

Connecting and powering your gateway