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



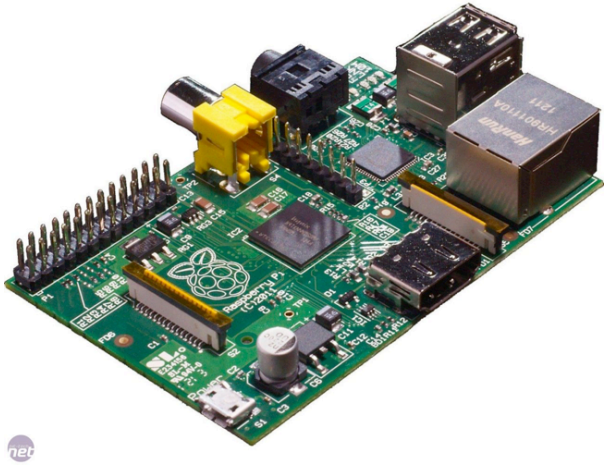
Prof. Congduc Pham
http://www.univ-pau.fr/~cpham
Université de Pau, France

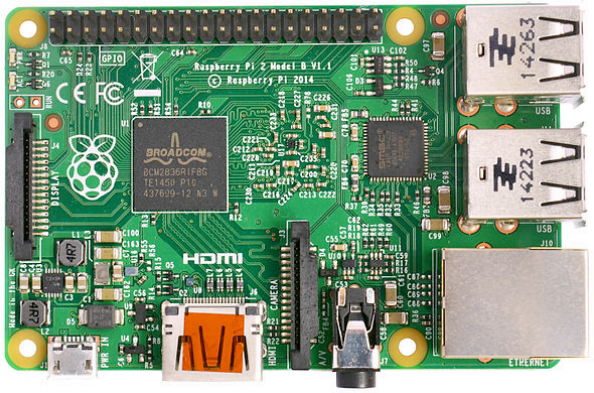
# Contents

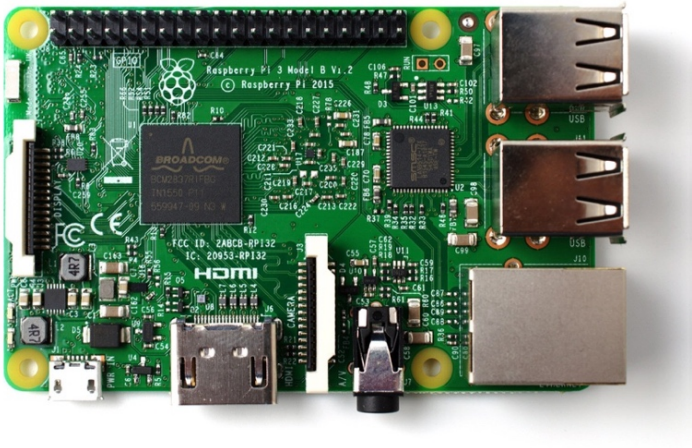
- We will show how to build a low-cost LoRa gateway to collect data from end-devices
- Configuration and update procedures will also be shown
- The device part will be shown in a separate tutorial
- The hardware platform is a Raspberry PI. RPI 1B/B+, 2B and 3B have been successfully tested
- But it is also necessary to read information from
  - https://github.com/CongducPham/LowCostLoRaGw
  - https://github.com/CongducPham/LowCostLoRaGw/tree/master/gw_full_latest
- As there are many issues that are not described here
- Let's get started…

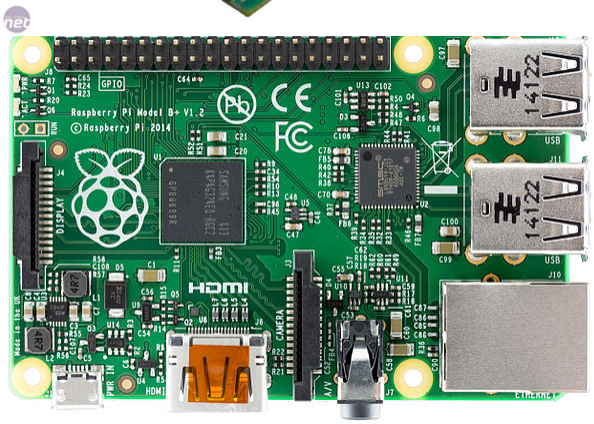
# Assembling the hardware

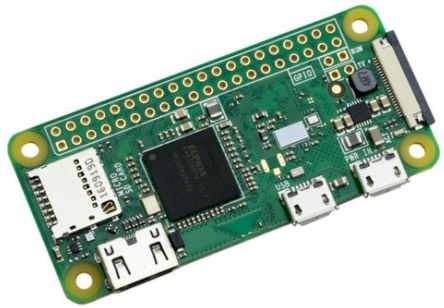
# Get the Raspberry





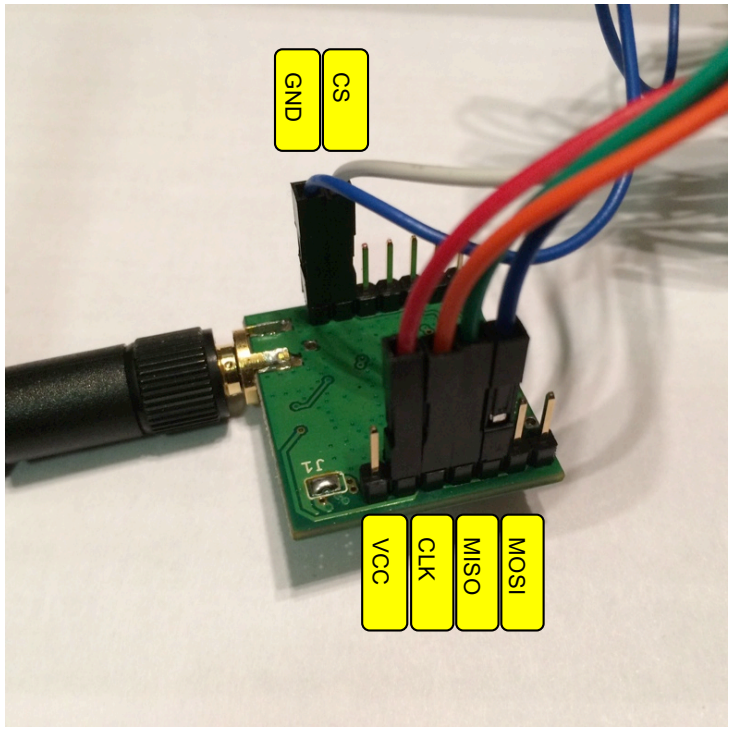


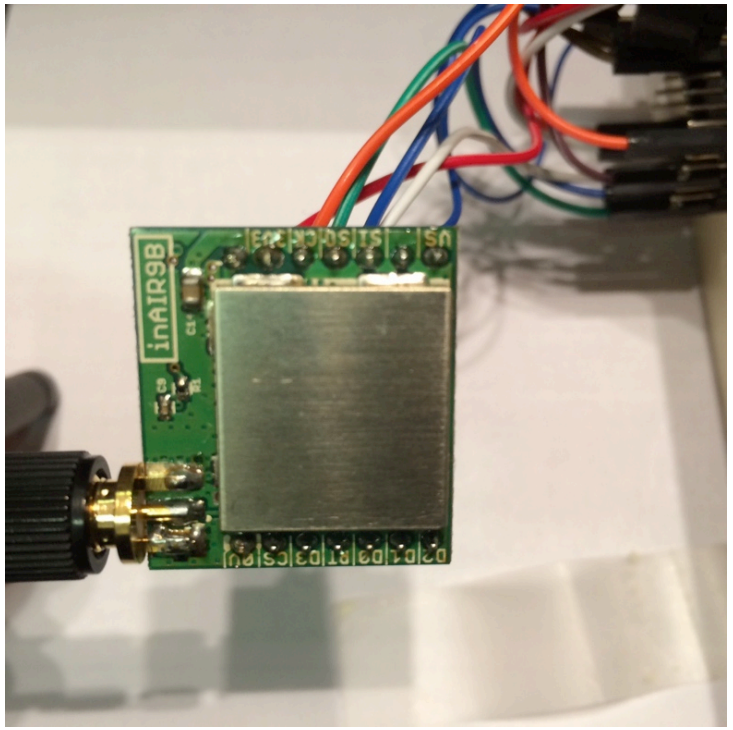




You can use RaspberryPI 1 model B or B+, RaspberryPI 2 model B, RaspberryPI 3 model B and RaspberryPI Zero (W). The most important usefull feature is the Ethernet interface for easy Internet connection. You can add WiFi with a WiFi USB dongle to use access-point features. With the RPI3 & RPI0W, WiFi and Bluetooth are embedded on the board.

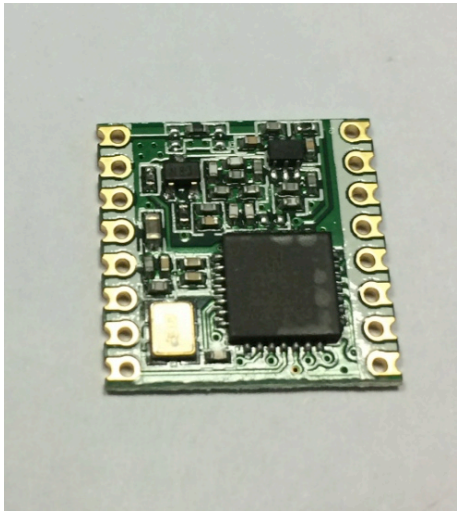
# NOW THE RADIO MODULE (1)

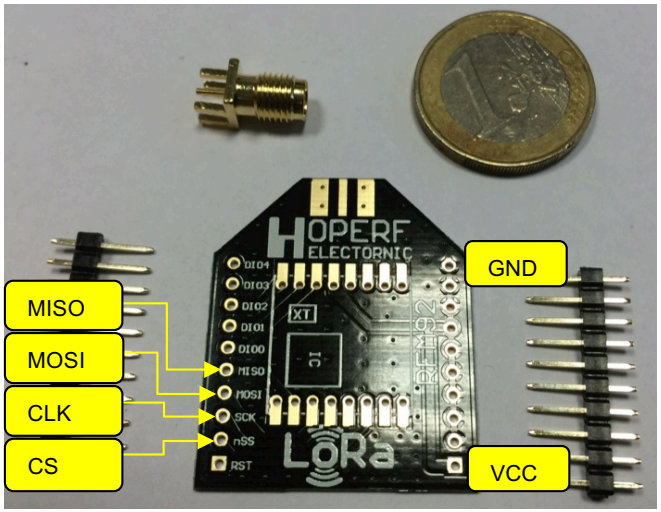



If you go for the inAir (9,9B,4) from Modtronix, the header pins can come fully assembled. Take the 6mm header pins to have enough length to connect F/F breadboard cables (left). Connect the SPI pins with the F/F cables. Try to use different colors. I use the following colors: MOSI (blue), MISO (green), 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)

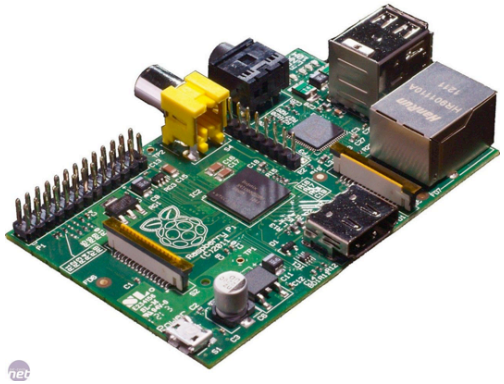


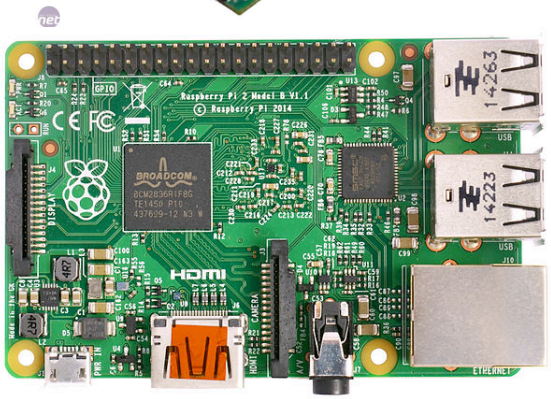


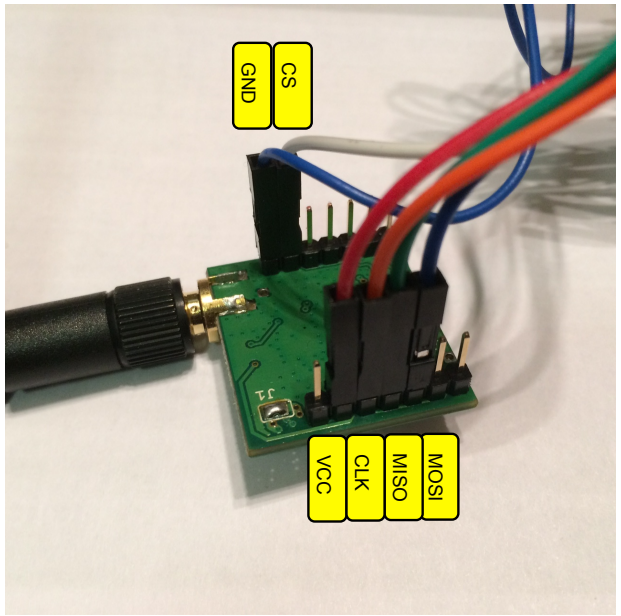


If you take the HopeRF RFM 92W/95W you may need the adaptor breakout and to go though some delicate but simple soldering tasks! It is not difficult but you have to trained a bit before! Then, like for the inAir9, use F/F breadboard cable to connect the SPI pins, using different colors as explained previously.

# Connecting the radio module (1)



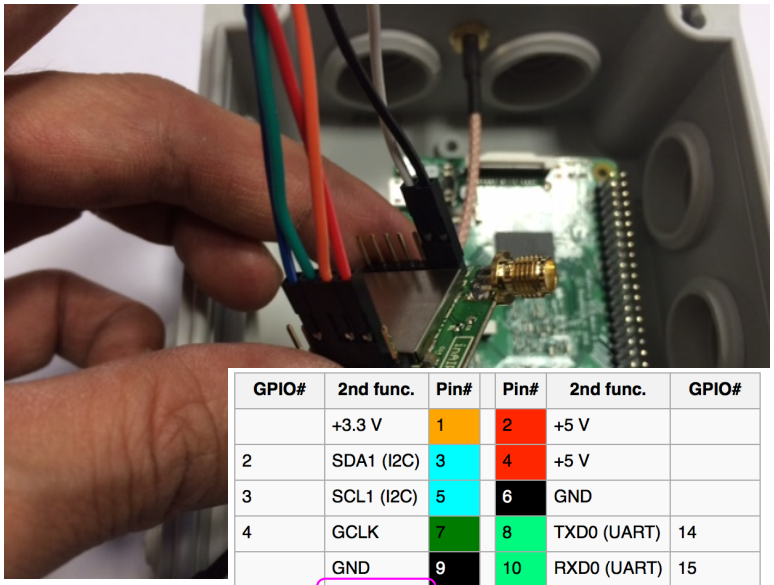




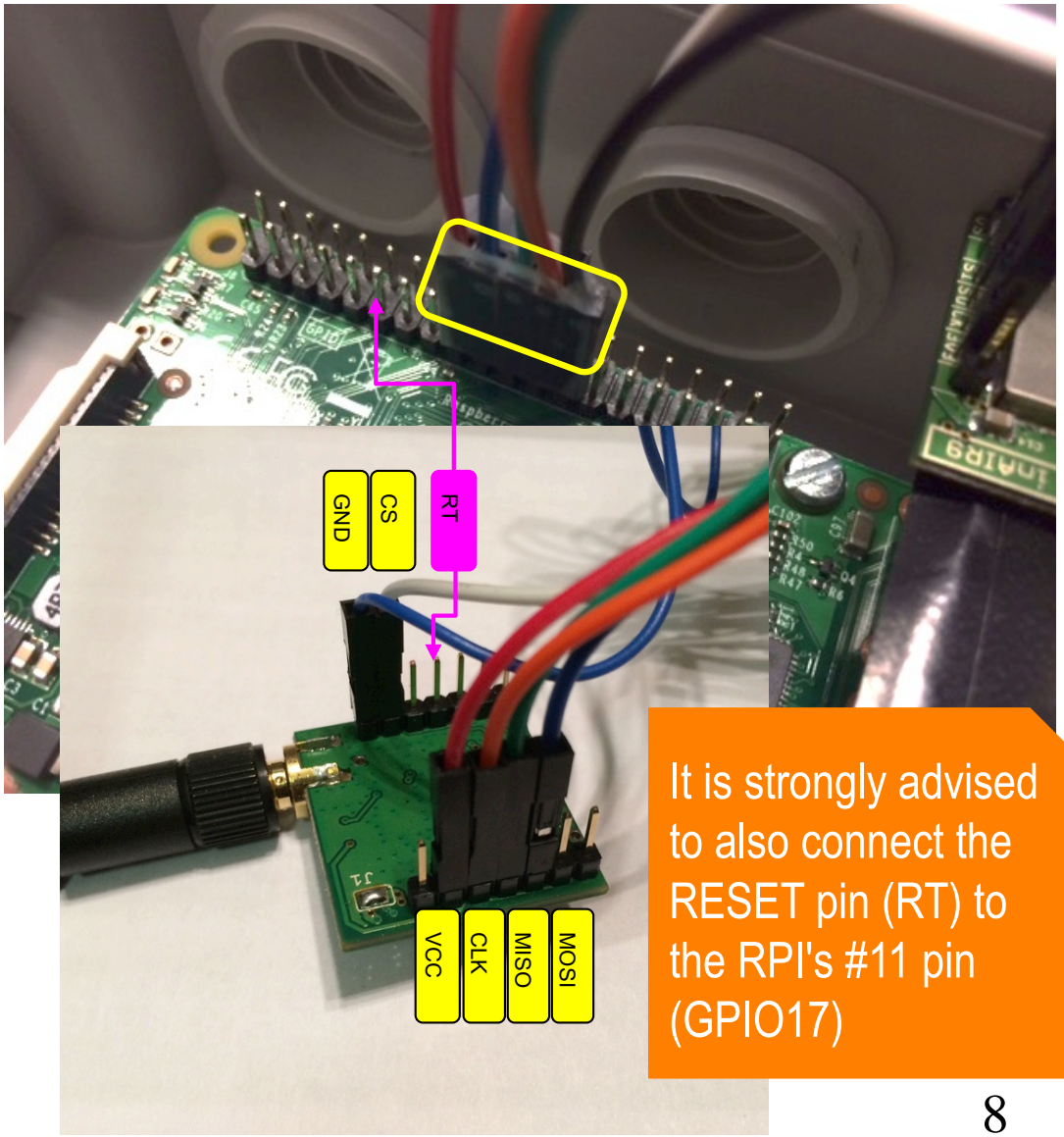

| GPIO# | 2nd func. | Pin# | Pin# | 2nd func. | GPIO# |
|---|---|---|---|---|---|
| | +3.3 V | 1 | 2 | +5 V | |
| 2 | SDA1 (I2C) | 3 | 4 | +5 V | |
| 3 | SCL1 (I2C) | 5 | 6 | GND | |
| 4 | GCLK | 7 | 8 | TXD0 (UART) | 14 |
| | GND | 9 | 10 | RXD0 (UART) | 15 |
| 17 | GEN0 | 11 | 12 | GEN1 | 18 |
| 27 | 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 |

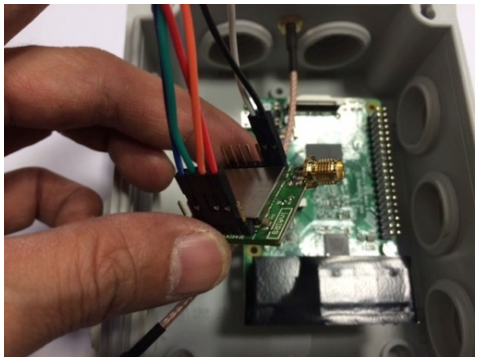



It is strongly advised to also connect the RESET pin (RT) to the RPI's #11 pin (GPIO17)

# Put it in a box (1)

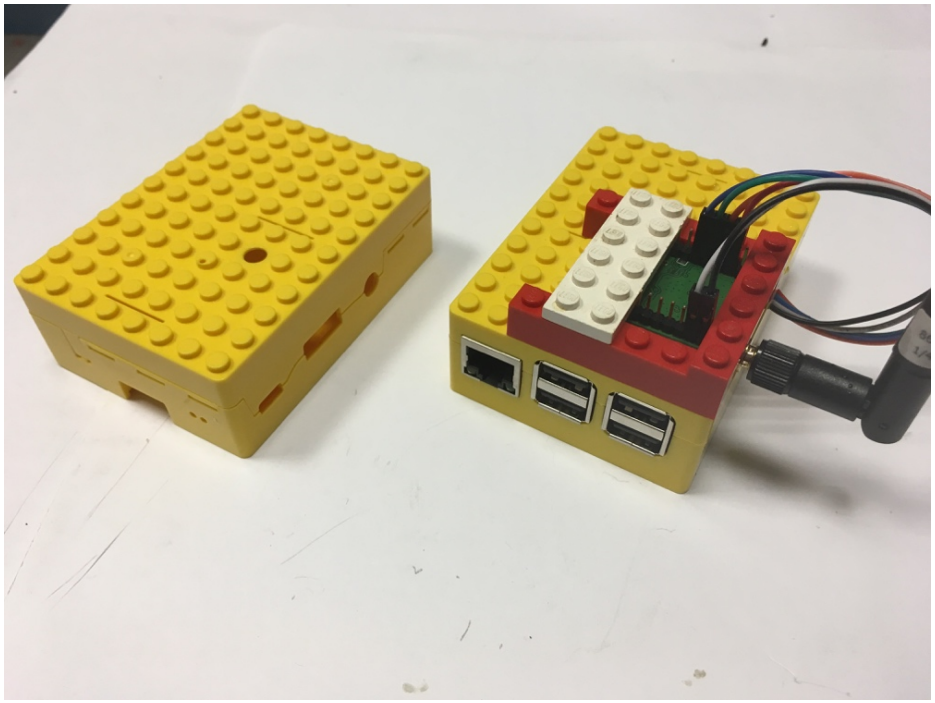


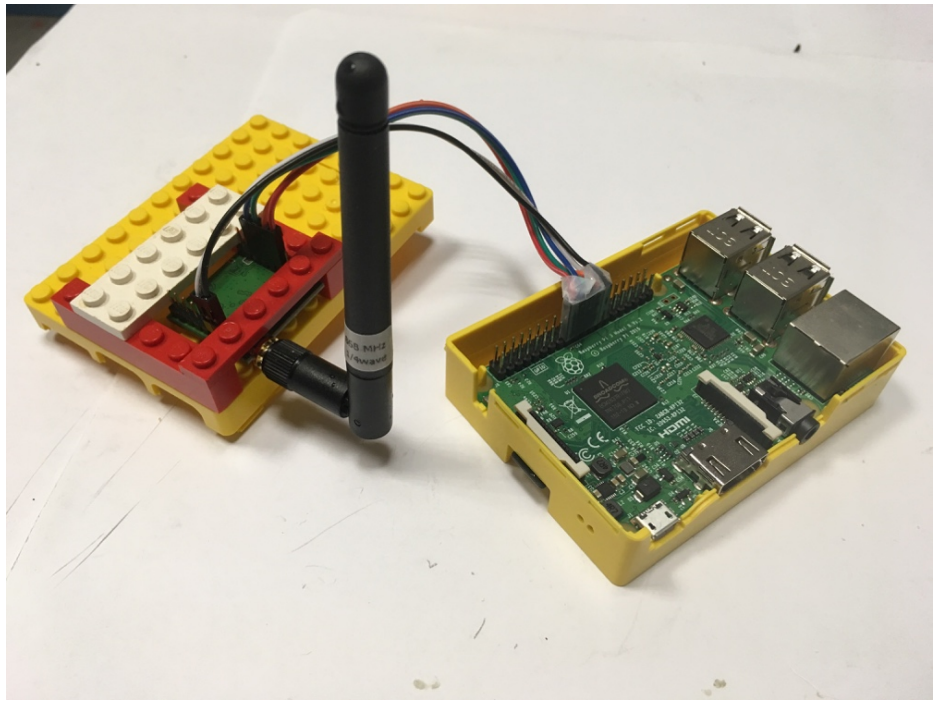





You can have a more integrated version, with a box for outdoor usage and PoE splitter to power the Raspberry with the Ethernet cable. See how we also use a DC-DC converter to get the 5V for the RPI.

# Put it in a box (2)

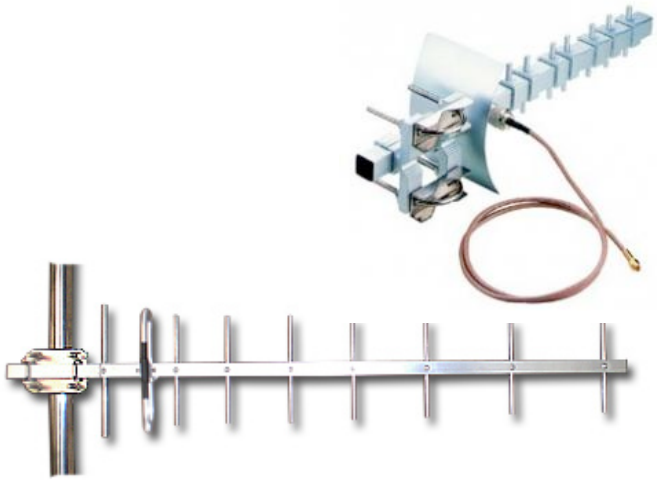




A simple, cheap and funny box is also very suitable for an indoor gateway. Actually, indoor deployment is probably the best option with an outdoor antenna as it will be shown in next slides.

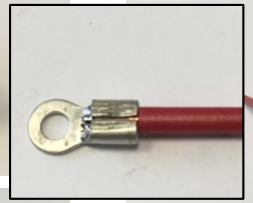
# ANTENNAS FOR GATEWAY

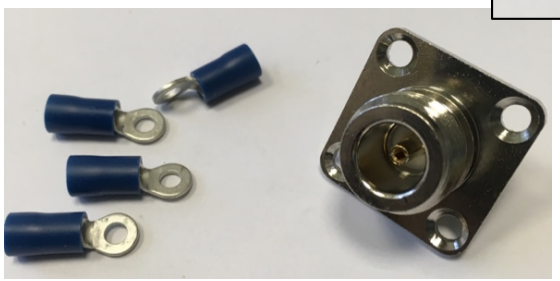
- Antennas for gateways can be placed on a building, at a high location.
- You can easily use ground plane or sleeve dipole antenna. More complex high gain antenna or a directional Yagi antenna can be purchased depending on your budget and whether the device deployment allows it.

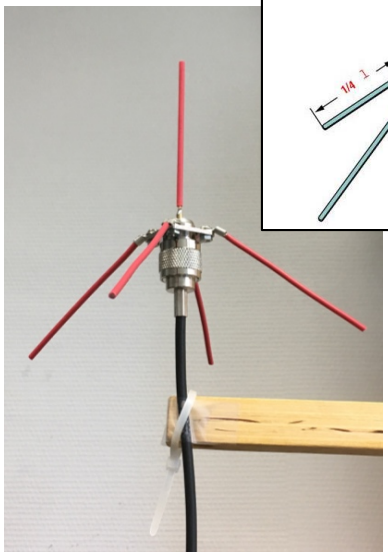
# Simple ¼ wave ground plane antenna

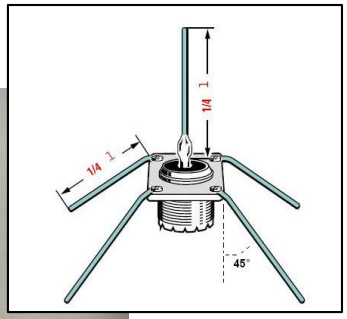
- The ground plane antenna can be made with 5 pieces of ¼ wave wires. ¼ wave in 868MHz is about 8.2cm.







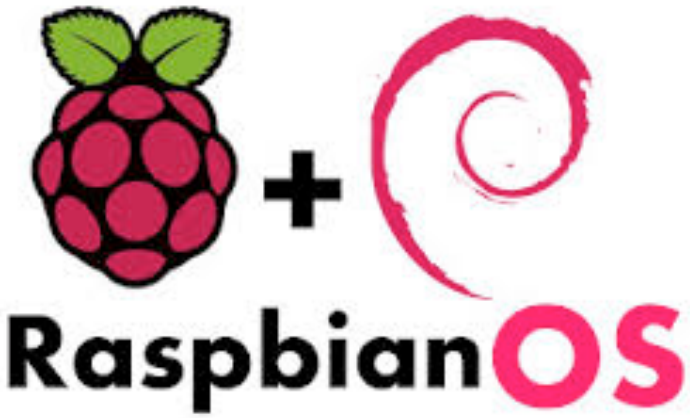





- You can buy a 3m-5m RG58 cable with an SMA-male at one end and a male N-connector at the other end. Or build your own cable.

# DIPOLE AND GROUND PLANE

# Getting, compiling & installing the software



# Flashing the OS

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

- 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.

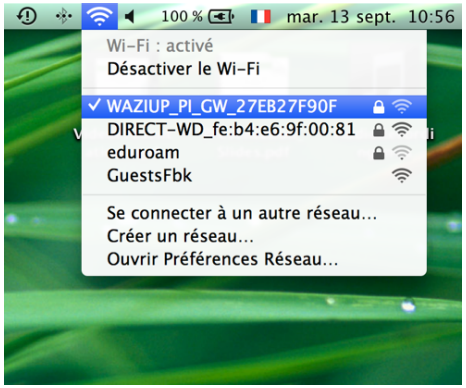
# SSH TO THE GATEWAY

- 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 `WAZIUP_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.



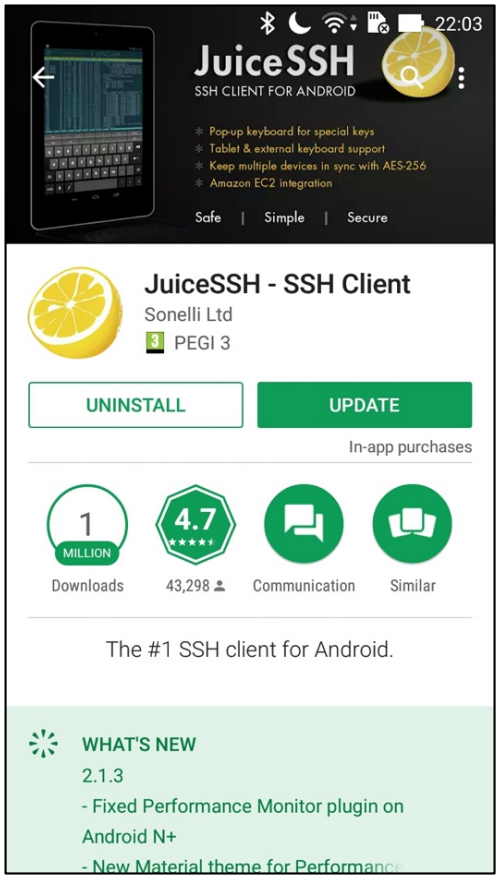

```
cpham — pi@raspberrypi: ~/lora_gateway —
MacBookProRetina-de-Congduc-Pham:~ cpham$ ssh pi@192.168.200.1
pi@192.168.200.1's password:

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

```
pi@raspberrypi:~/lora_gateway $ ./cmd.sh
=======================================* Gateway 00000027EB84C456 *===
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                      +
-------------------------------------------------------* Node-Red *--+
a- start Node-Red                                                     +
b- stop Node-Red                                                      +
c- enable Node-Red at boot                                            +
d- disable Node-Red at boot                                           +
-----------------------------------------------------* Connectivity *--+
f- test: ping www.univ-pau.fr                                         +
g- wifi: configure as WiFi client at next reboot                      +
h- wifi: indicate WiFi SSID and password at next reboot               +
i- wifi: configure as WiFi access point at next reboot                +
-------------------------------------------------* 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:
```

# 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 865.2MHz (see https://github.com/CongducPham/LowCostLoRaGw#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

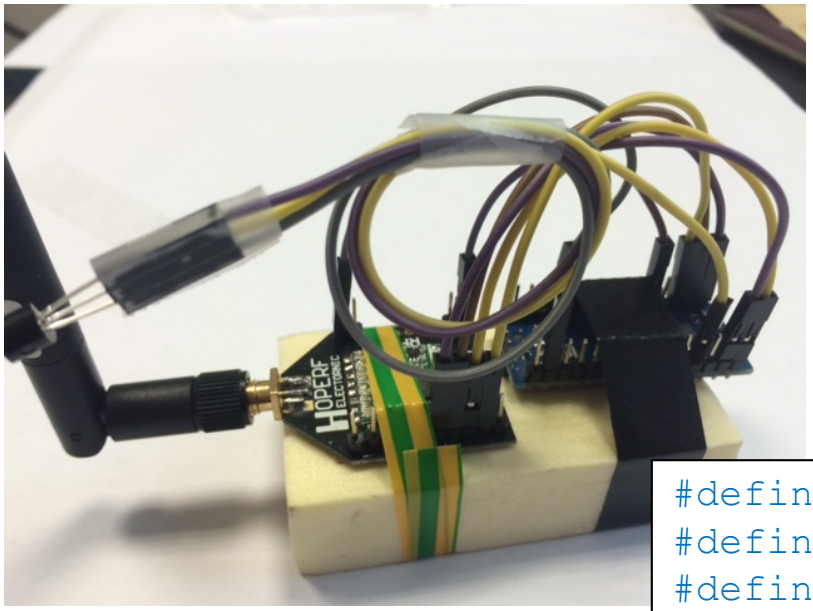
```
{
    "radio_conf" : {
        "mode" : 1,
        "bw" : 500,
        "cr" : 5,
        "sf" : 12,
        "ch" : -1,
        "freq" : -1
    },
    "gateway_conf" : {
        "gateway_ID" : "000000XXXXXXXXXX",
        "ref_latitude" : "my_lat",
        "ref_longitude" : "my_long",
        "wappkey" : 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. 865.2, if you want to specify a given frequency, "freq" has priority over "ch"

# DEFAULT CONFIGURATION



\!##TC/18.5

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

The default configuration in the Arduino_LoRa_Simple_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

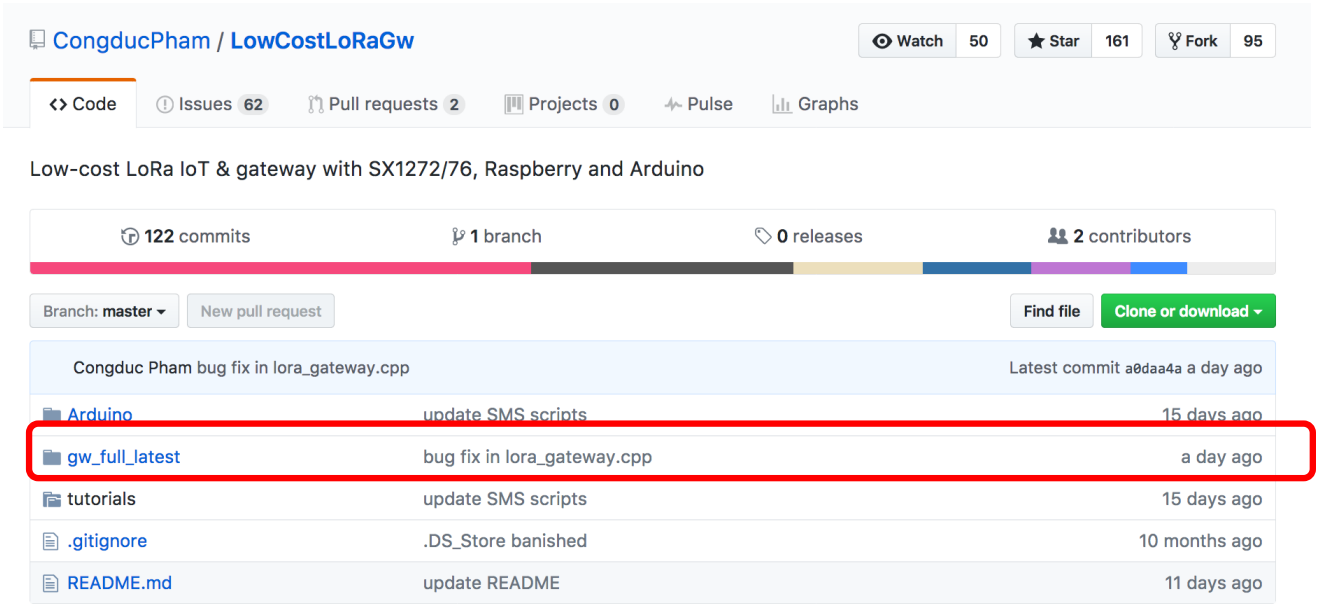
# CHECK THAT THE GATEWAY IS RUNNING

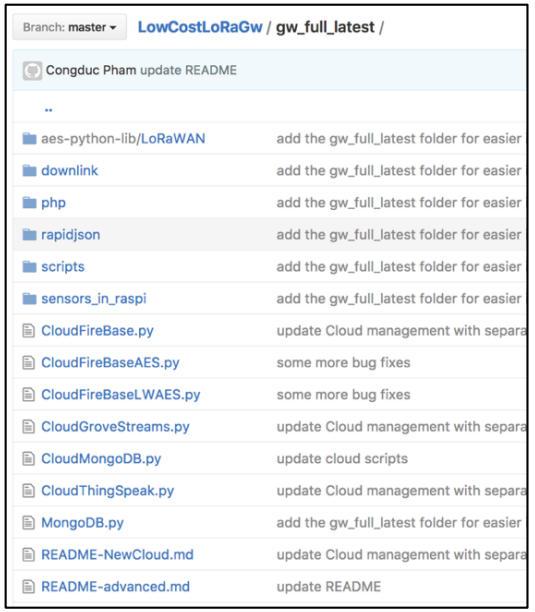
- Use option 3 of the text interface

```
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




The software should be installed in a `lora_gateway` folder. Delete any previous folder.

```
> rm -rf lora_gateway
```

then

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

or

```
> svn checkout https://github.com/CongducPham/LowCostLoRaGw/trunk/gw_full_latest lora_gateway
```

# 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
```

# Use a script to update the gateway

- Alternatively, the gateway can also be updated to the latest version with the `update_gw.sh` script.
- The first step is to get the latest version of the update script

```
> cd
> svn checkout https://github.com/CongducPham/LowCostLoRaGw/trunk/gw_full_latest/scripts
> cd scripts
> ll
total 48
-rw-r--r-- 1 pi pi  3561 May 10 17:31 bashrc.sh
-rwxr-xr-x 1 pi pi 10562 May 10 17:31 config_gw.sh
-rw-r--r-- 1 pi pi   230 May 10 17:31 interfaces_ap
-rwxr-xr-x 1 pi pi    99 May 10 17:31 mnt-dropbox
-rwxr-xr-x 1 pi pi   610 May 10 17:31 mongodb_repair.sh
-rwxr-xr-x 1 pi pi   816 May 10 17:31 start_access_point.sh
-rwxr-xr-x 1 pi pi    57 May 10 17:31 start_gw.sh
-rwxr-xr-x 1 pi pi   673 May 10 17:31 stop_access_point.sh
-rwxr-xr-x 1 pi pi    37 May 10 17:31 unmnt-dropbox
-rwxr-xr-x 1 pi pi  1537 May 10 17:31 update_gw.sh
```

# Updating the gateway software

- It is also possible to get only this script
  - `wget https://raw.githubusercontent.com/CongducPham/LowCostLoRaGw/master/gw_full_latest/scripts/update_gw.sh`
- Then type the following commands
  - `rm -rf lora_gateway`
  - `./update_gw.sh`
- Removing any previous `lora_gateway` folder triggers a full update
- The gateway will obtain the latest distribution from our github repository and will create a new `lora_gateway` folder
- Next periodic updates without deleting the existing `lora_gateway` folder, i.e. preserving existing configuration files, will be presented later on

# CONFIGURING THE GATEWAY (1)

- Go into the `scripts` folder in the newly created `lora_gateway` folder
- Run the `basic_config_gw.sh` script
  - `./basic_config_gw.sh`
- The script will get the hardware address of the gateway to define the gateway'id, using the last 5 bytes of the MAC address
  - `ifconfig`

```
pi@raspberrypi:~ $ ifconfig
eth0      Link encap:Ethernet  HWaddr b8:27:eb:79:5c:47
          inet addr:10.0.13.185  Bcast:10.0.13.255  Mask:255.255.255.0
          inet6 addr: fe80::ba27:ebff:fe79:5c47/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:3500 errors:0 dropped:0 overruns:0 frame:0
```

- It will also compile the low-level gateway software, remember to check compilation options of `radio.makefile`
- If you need advanced configuration, use `config_gw` and follow the instructions as shown in the next slides **otherwise you are all done**

# ADVANCED CONFIGURATION ONLY (1)

```
*********************************************
*** compile lora_gateway executable Y/N  ***
*********************************************
```

Enter Y

```
*****************************************************
*** create log symb link to ~/Dropbox/LoRa-test Y/N  ***
*****************************************************
```

Enter Y

```
***********************************
*** configure hostapd.conf Y/N  ***
***********************************
```

Enter Y

```
*************************************************************
*** configure a newly installed hostapd/dnsmasq package Y/N  ***
*************************************************************
```

Enter N

```
*********************************************
*** configure bluetooth network name Y/N  ***
*********************************************
```

Enter N

```
***********************************
*** install DHT22 support Y/N  ***
***********************************
```

Enter Y

# ADVANCED CONFIGURATION ONLY (2)

```
****************************************
*** edit gateway_conf.json now? Y/N ***
****************************************
```

Enter N

```
************************************
*** activate DHT22 MongoDB Y/N/Q  ***
************************************
```

Enter Q

```
************************************************************
*** edit LoRa data MongoDB local storage option? Y/N  ***
************************************************************
```

Enter N

```
******************************
*** run gateway at boot Y/N ***
******************************
```

Enter Y

```
**********************************************
*** check configuration (recommended) Y/N  ***
**********************************************
```

Enter N

```
********************
*** reboot Y/N  ***
********************
```

Enter N

# Start the command interface

```
> ./cmd.sh
```

As you can see, the gateway id shown by the command interface is now correct

```
pi@raspberrypi:~/lora_gateway $ ./cmd.sh
=======================================* Gateway 00000027EB84C456 *===
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                      +
-------------------------------------------------------* Node-Red *--+
a- start Node-Red                                                     +
b- stop Node-Red                                                      +
c- enable Node-Red at boot                                            +
d- disable Node-Red at boot                                           +
-----------------------------------------------------* Connectivity *--+
f- test: ping www.univ-pau.fr                                         +
g- wifi: configure as WiFi client at next reboot                      +
h- wifi: indicate WiFi SSID and password at next reboot               +
i- wifi: configure as WiFi access point at next reboot                +
-------------------------------------------------* 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:
```

# Periodic update procedure

- You can use option **U** to update from repository and still keep all your configuration files: `gateway_conf.json`, `clouds.json` and `key*`
- This simply call `update_gw.sh`
- You can also install a single file with option **V** that will prompt for a URL
- You can enter a URL that has been provided by some administrator
- Example in the next slide

```
pi@raspberrypi:~/lora_gateway $ ./cmd.sh
=======================================* Gateway 00000027EB84C456 *===
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                      +
-------------------------------------------------------* Node-Red *--+
a- start Node-Red                                                     +
b- stop Node-Red                                                      +
c- enable Node-Red at boot                                            +
d- disable Node-Red at boot                                           +
-----------------------------------------------------* Connectivity *--+
f- test: ping www.univ-pau.fr                                         +
g- wifi: configure as WiFi client at next reboot                      +
h- wifi: indicate WiFi SSID and password at next reboot               +
i- wifi: configure as WiFi access point at next reboot                +
-------------------------------------------------* 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:
```

# Download and install a file (1)

- With option **V**, you can enter an URL that points to a file. The file will be downloaded and installed in the `lora_gateway` folder.

```
Enter your choice:
V
---------------------------------------------------------------------
BEGIN OUTPUT
Download and install a file
Enter the URL of the file:
https://www.dropbox.com/s/mcmg4yeksr340c2/example-install-file.txt
```

```
Download and install a file
Enter the URL of the file:
https://www.dropbox.com/s/mcmg4yeksr340c2/example-install-file.txt
--2017-05-09 22:16:53--  https://www.dropbox.com/s/mcmg4yeksr340c2/example-install-file.txt
Resolving www.dropbox.com (www.dropbox.com)... 162.125.65.1
Connecting to www.dropbox.com (www.dropbox.com)|162.125.65.1|:443... connected.
HTTP request sent, awaiting response... 302 Found
Location: https://dl.dropboxusercontent.com/content_link/Veb5Tx1XY65zpGTJ9ZUYQAuAwhDY9GiEmw9HUxcQXuMh62IneXy7BUp1EF450L0l/file [following]
--2017-05-09 22:16:54--  https://dl.dropboxusercontent.com/content_link/Veb5Tx1XY65zpGTJ9ZUYQAuAwhDY9GiEmw9HUxcQXuMh62IneXy7BUp1EF450L0l/file
Resolving dl.dropboxusercontent.com (dl.dropboxusercontent.com)... 162.125.65.6
Connecting to dl.dropboxusercontent.com (dl.dropboxusercontent.com)|162.125.65.6|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 167 [text/plain]
Saving to: 'example-install-file.txt'

example-install-file.txt            100%[=======================================================================>]     167  --.-KB/s   in 0s

2017-05-09 22:16:55 (17.2 MB/s) - 'example-install-file.txt' saved [167/167]

Done
END OUTPUT
Press RETURN/ENTER...
```

# Download and install a file (2)

- This feature is very usefull for end-users to simply update some files on the gateway.
  - gateway_conf.json and clouds.json
  - radio.makefile
  - ...
- An administrator can write appropriate configuration files for the end-user and generate an URL to this file (with Dropbox for instance)
- The URL can be either be sent by mail or SMS to the end-user.
- The end-user has to simply log into the gateway (using an Android smartphone or tablet connecting to the gateway's WiFi) and select option V to enter the URL.
- The end-user will then just reboot the gateway with option R for the new configuration to run.

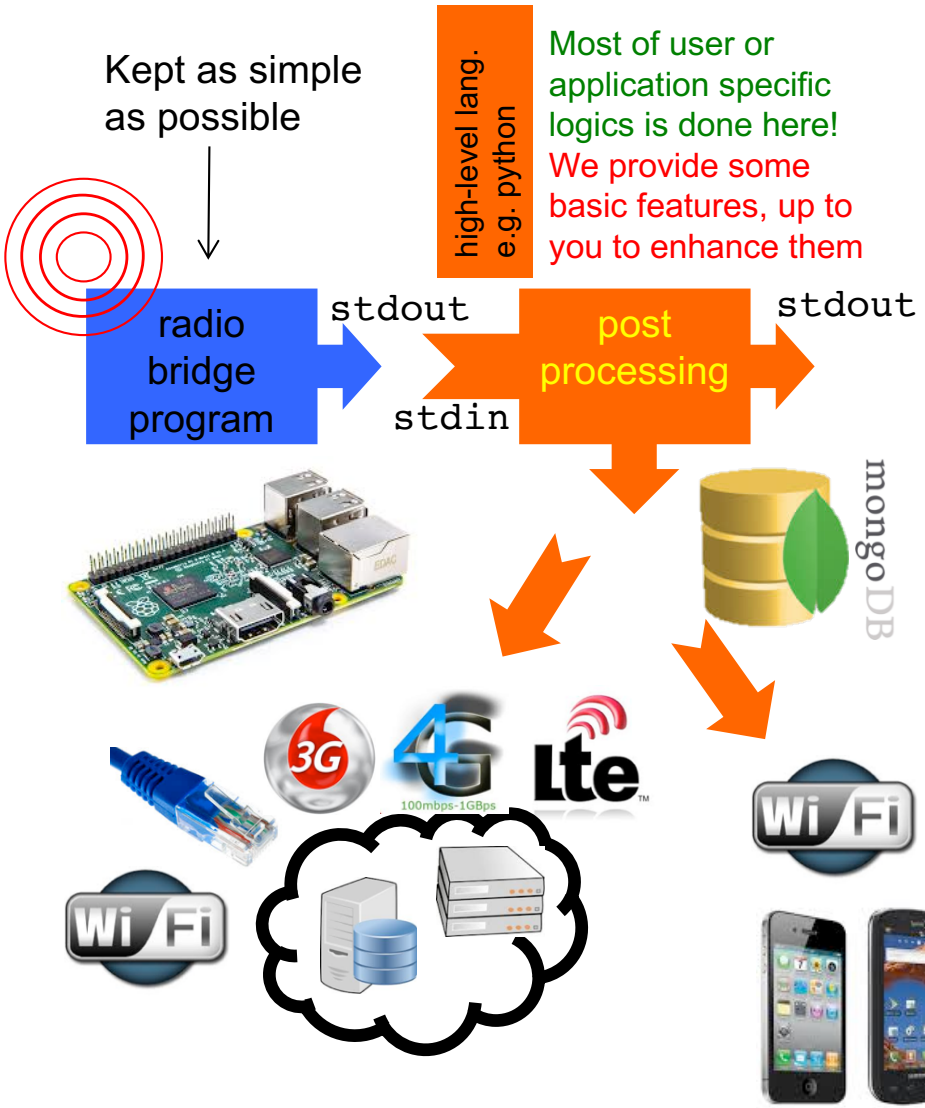
# Download and install a file (3)

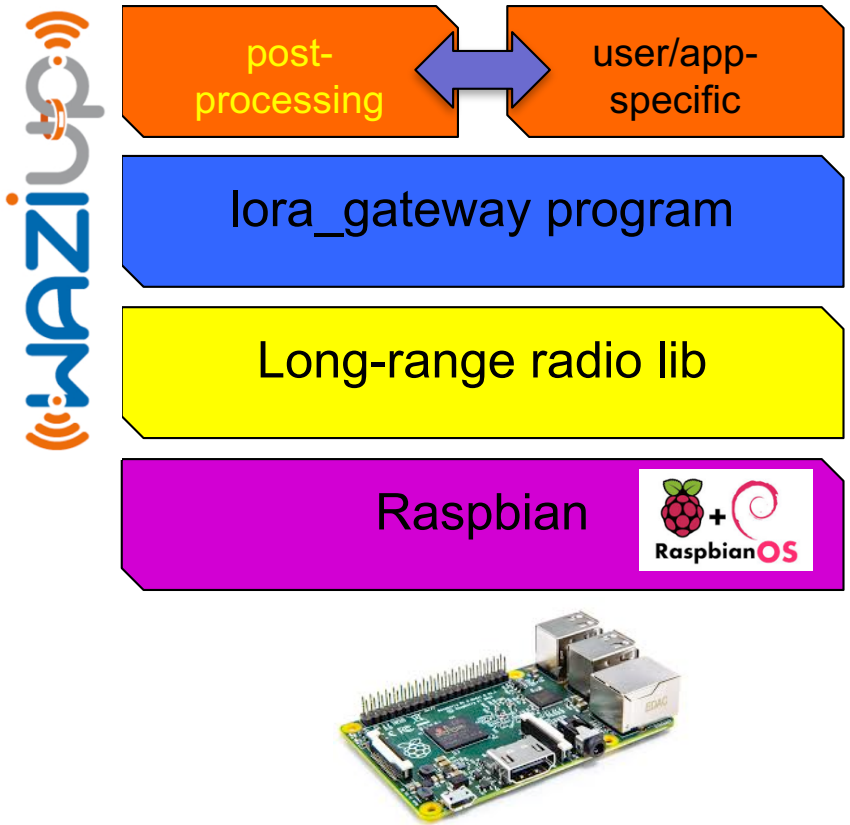
- System files can also be installed with option W that will prompt for a command

```
Enter your choice:
W
---------------------------------------------------------------------
BEGIN OUTPUT
Run a command
Enter the command to run:
sudo wget -O /etc/test.txt https://www.dropbox.com/s/mcmg4yeksr340c2/example-install-file.txt
```

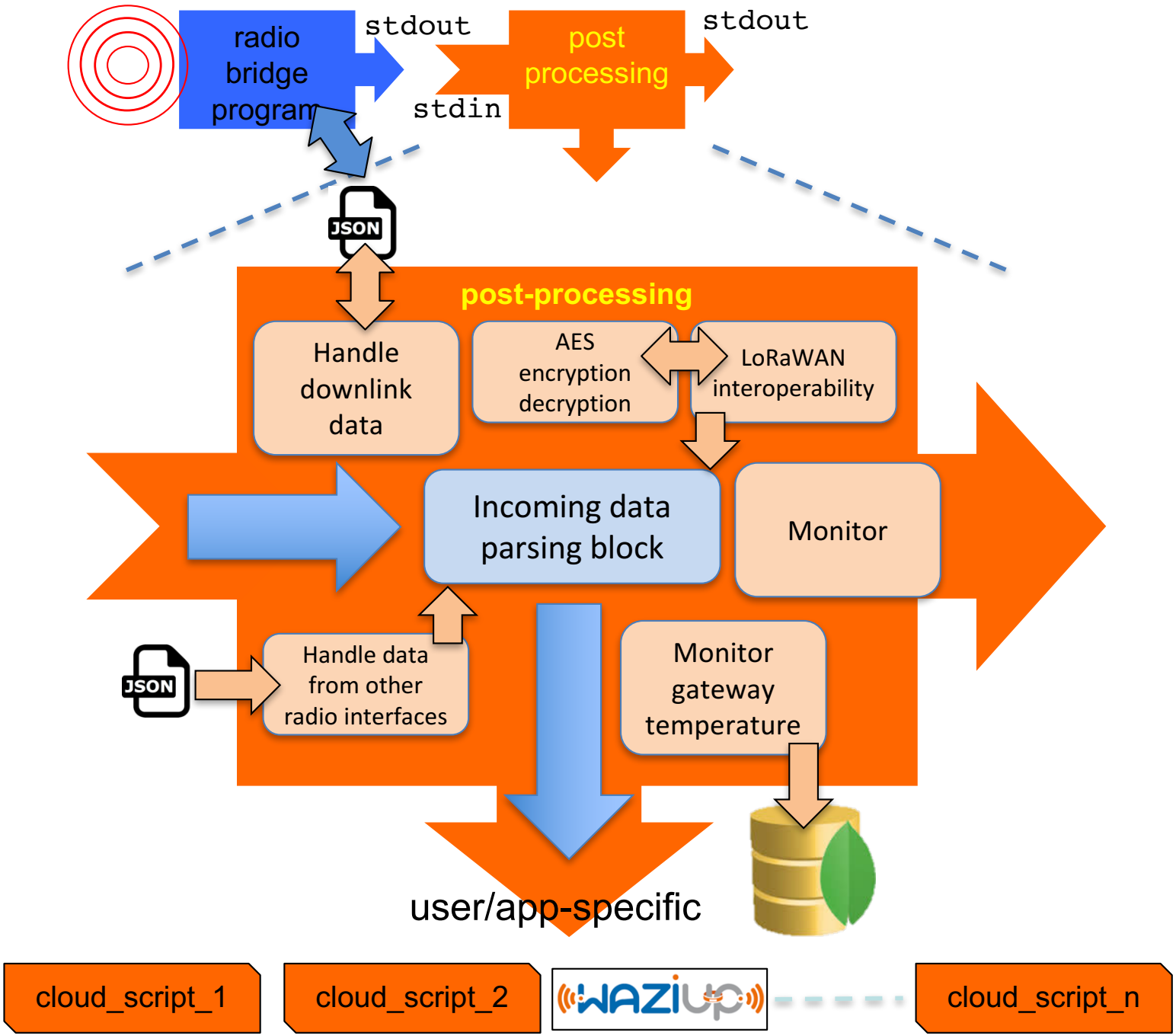
- Here, the previous example file will be installed in `/etc` under the name `test.txt`
- Like previously, the exact command can be sent to the end-user

# Our Low-Cost gateway architecture



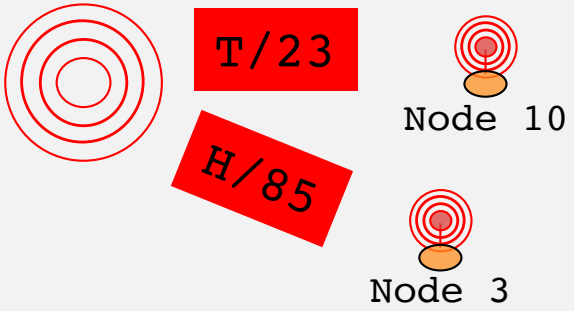
# POST-PROCESSING BLOCK



# 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 21 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

```
> 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

# LOG RECEIVED MESSAGES USING CLOUD SERVICES


```
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
```

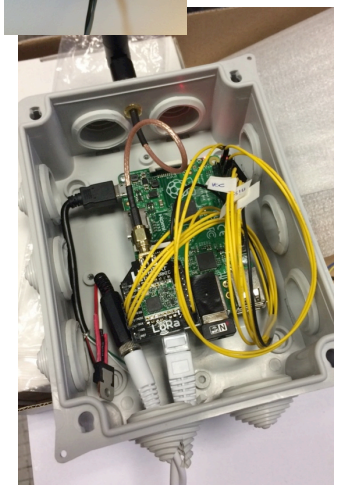
\$ or \! before the data indicates that the data should be logged on a file or a cloud. It is up to the end-device to decide which option

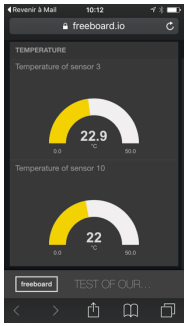
# Starting the complete gateway

- 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

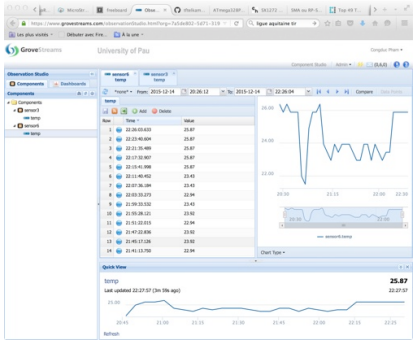





Dropbox


freeboard

FIWARE





GroveStreams

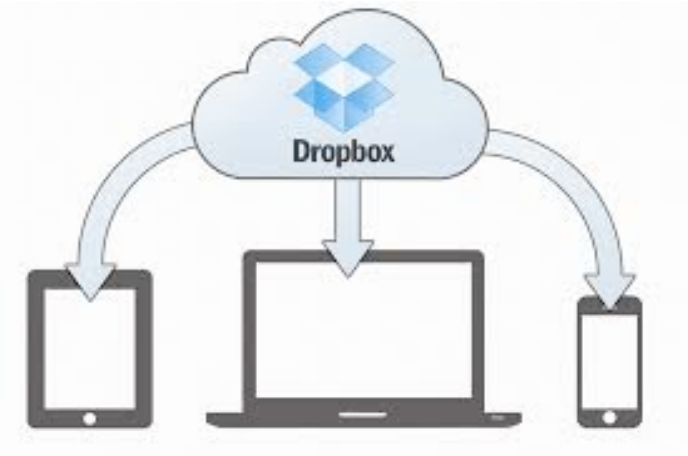
SensorCloud™

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

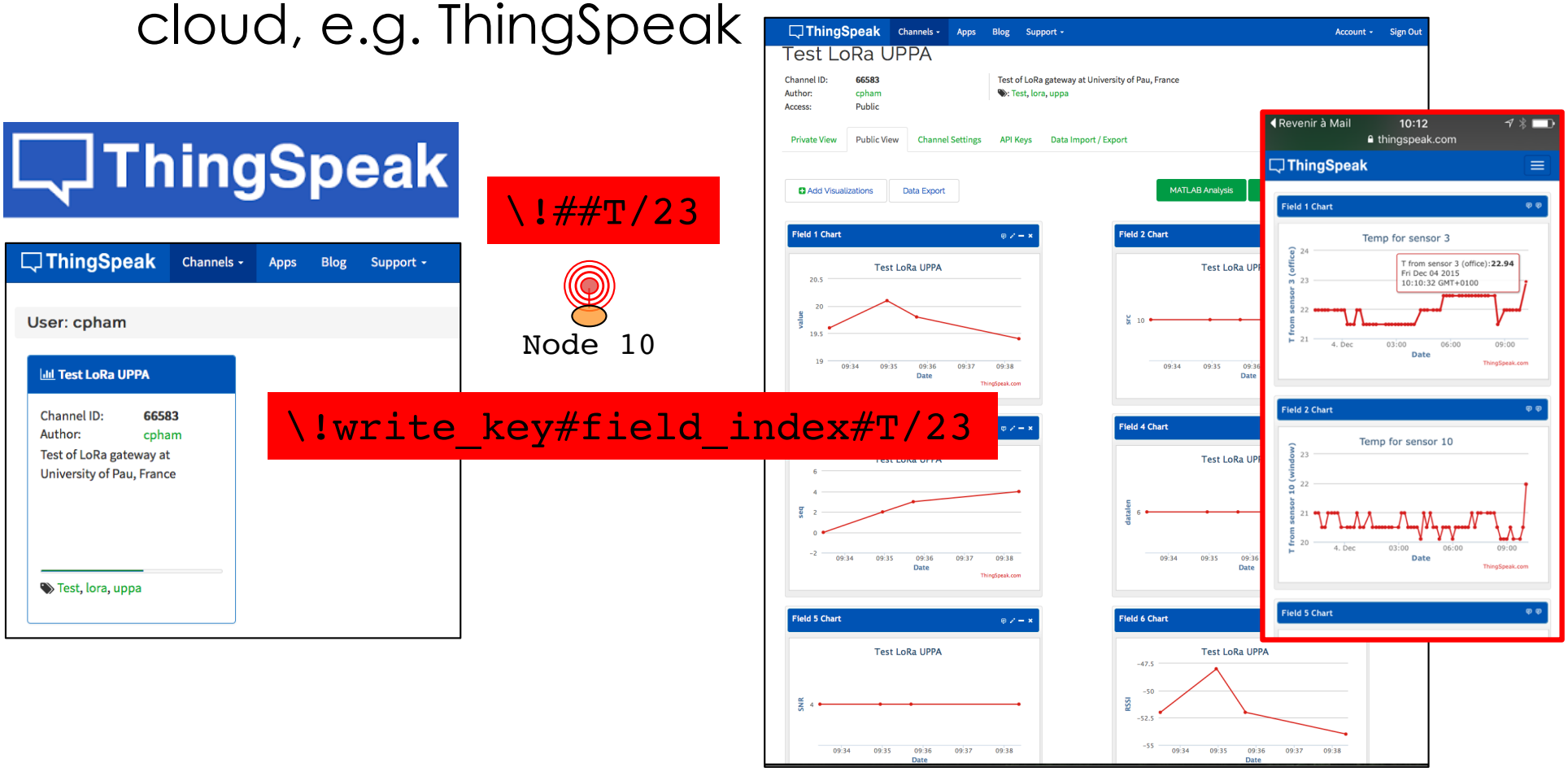
- A message starting with '\$' is logged in a file 'telemetry.log' in a folder shared through Dropbox (if enabled)

```
(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
...
```

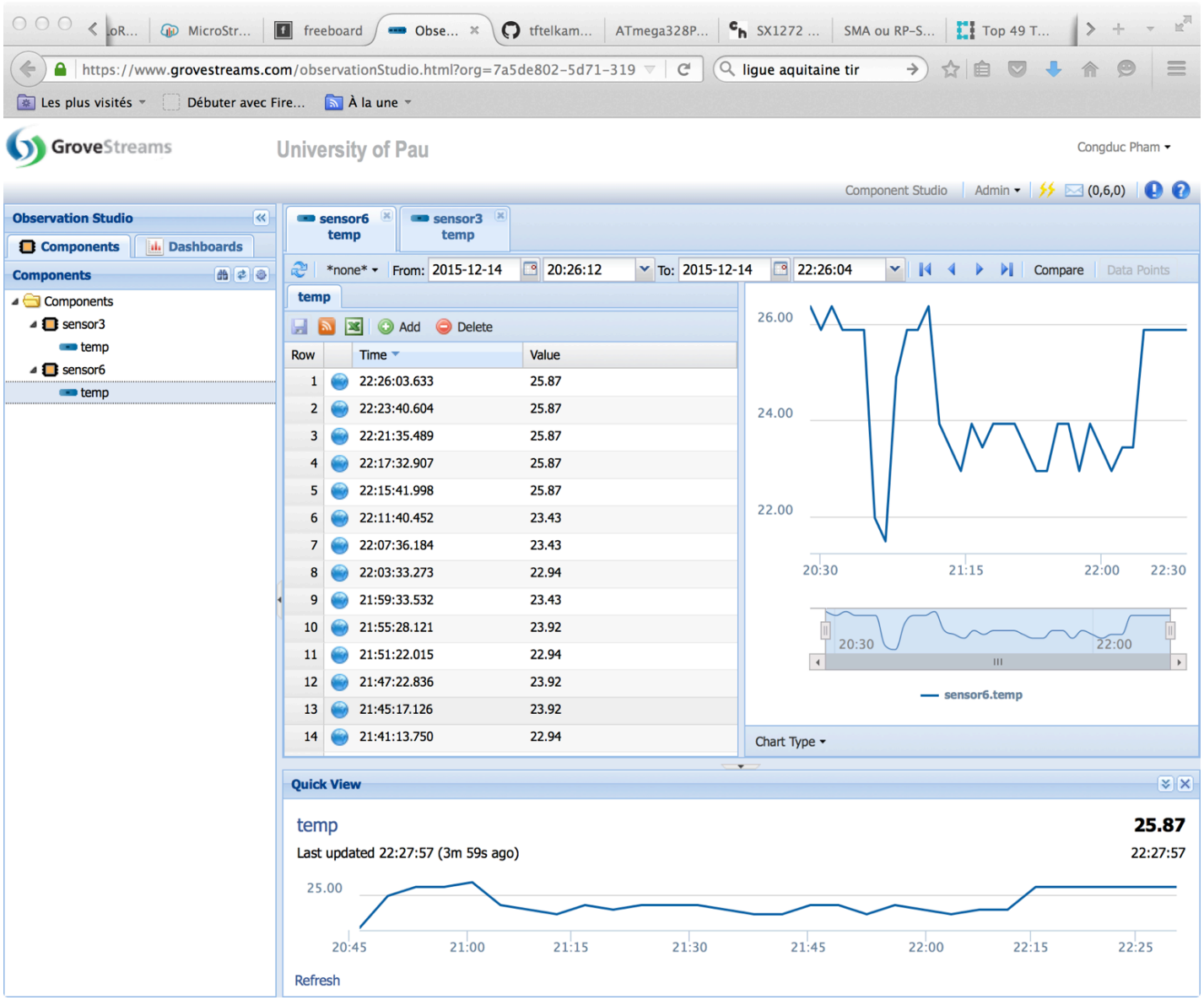

# USING ThingSpeak

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



# USING Firebase

# USING GroveStreams



# Configuring data management

- For WAZIUP, received data from devices will be uploaded to the WAZIUP Orion data platform. Therefore `clouds.json` file should be set as follows:

```
{
    "name":"WAZIUP Orion cloud",
    "script":"python CloudOrion.py",
    "type":"iotcloud",
    "write_key":"",
    "enabled":true
},
```

- Modify `clouds.json` accordingly
- `CloudOrion.py` script will use information from `key_Orion.py` to configure data management for each organization
- Therefore you need to configure this file for each organization/gateway

# KEY_ORION.PY

```
#####################################################
#server: CAUTION must exist
orion_server="http://broker.waziup.io/v2"

#project name
project_name="waziup"

#your organization: CHANGE HERE
#choose one of the following: "DEF", "UPPA", "EGM", "IT21", "CREATENET", "CTIC", "UI", "ISPACE",
"UGB", "WOELAB", "FARMERLINE", "C4A", "PUBD"
organization_name="ORG"

#service tree: CHANGE HERE at your convenience
#should start with /
#service_tree='/LIUPPA/T2I/CPHAM'
service_tree=''

#sensor name: CHANGE HERE but maybe better to leave it as Sensor
#the final name will contain the sensor address
sensor_name="Sensor"

#service path: DO NOT CHANGE HERE
service_path='/'+organization_name+service_tree

#SUMMARY
#the Fiware-Service will be project_name, e.g. "waziup"
#the Fiware-ServicePath will be service_path which is based on both organization_name and
service_tree, e.g. "/UPPA/LIUPPA/T2I/CPHAM"
#the entity name will then be organization_name+"_"+sensor_name+scr_addr, e.g. "UPPA_Sensor2"

source_list=[]
```

You need to change the organization_name.

service_tree is optional

# Editing key_Orion.py


Use `nano` to edit the file:

```
> nano key_Orion.py
```

Then CTRL-O + RETURN to save

CTRL-X to quit

# 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

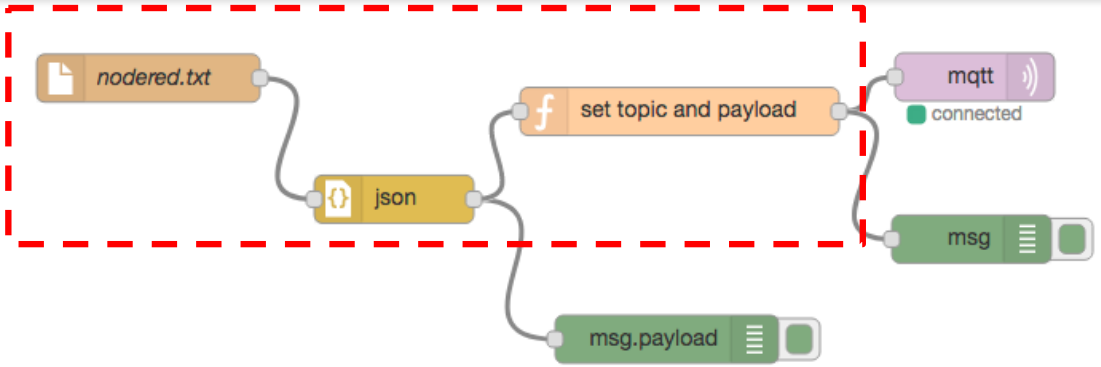
- Use our templates to write your own cloud script
  - CloudMongoDB.py, CloudThingSpeak.py, CloudFireBase.py, CloudGroveStreams.py, CloudOrion.py, CloudNoInternet.py, CloudNodeRed.py
- A cloud script is called with 5 arguments
  - ldata: the received data
    - e.g. #4#TC/21.5 as 1st argument (sys.argv[1] 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.

# Example with Node-Red

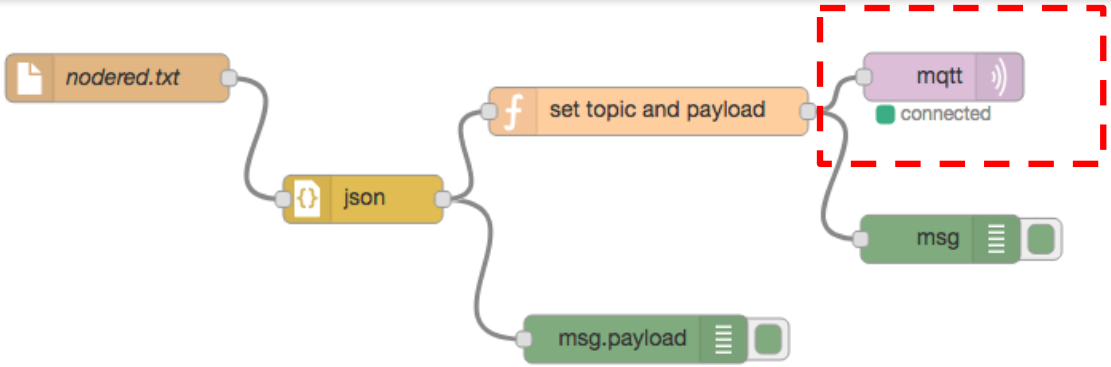
- `CloudNodeRed.py` shows how interface with Node-Red can be simply implemented to benefit from the facility offered by Node-Red
- We use `key_NodeRed.py` to define 3 variables that will be used by `CloudNodeRed.py`
  - `project_name="waziup"`
  - `organization_name="UPPA"`
  - `sensor_name="Sensor"`
- when a device which address is 2 sends "TC/22.5/HU/85" to the gateway, `CloudNodeRed.py` will generate the following json entries in `nodered/nodered.txt` file
  - `{"source":"waziup_UPPA_Sensor2","measure":"TC","value":22.5}`
  - `{"source":"waziup_UPPA_Sensor2","measure":"HU","value":85}`

# Node-Red flow (1)




- The Node-Red flow is composed of a tail node that follows the `nodered/nodered.txt` file for new entries. Each entry will be converted into a json object with a json node. A function node will use the json entry to build a message as follows
  - `msg.topic=msg.payload.source+'/'+msg.payload.measure`
  - `msg.payload=msg.payload.value`
  - `return msg;`

# Node-Red flow (2)




- An MQTT node using the `test.mosquitto.org` broker will receive the messages with the topic defined as `waziup_UPPA_Sensor2/TC` and `waziup_UPPA_Sensor2/HU`
- It will then respectively publish 22.5 and 85 under these topics
- More information on:
  - `https://github.com/CongducPham/LowCostLoRaGw/blob/master/gw_full_latest/README-NodeRed.md`

# Rebooting the gateway

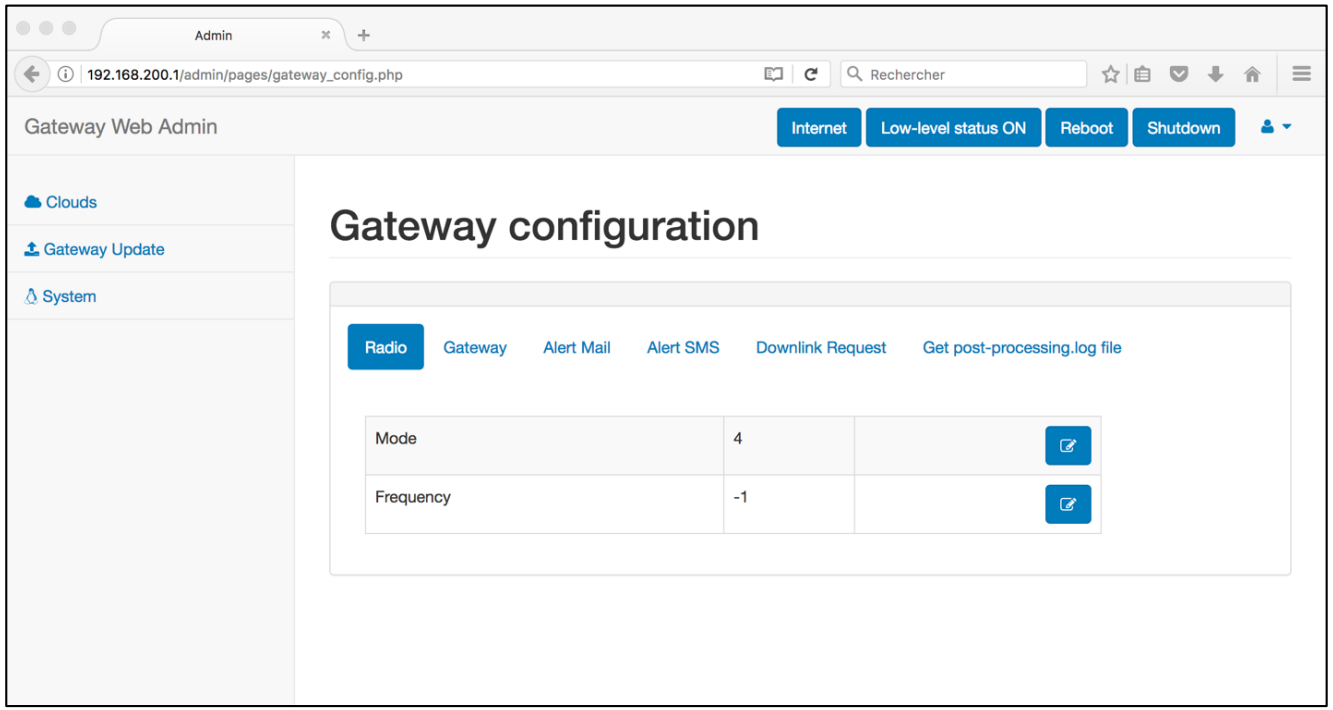
- Your gateway is now updated and configured
- You can now reboot the gateway. To do so, just type:
  - `sudo shutdown -r now`
- Or run `./cmd.sh` and chose option **R**
- Once the gateway has rebooted, check the WiFi SSID which now should meet your gateway's id
- Try to avoid unplugging power cable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

# Gateway web admin interface (1)

- A gateway web admin interface has been added to the latest version
- Note that the SD card image includes the web admin interface installed, so you may skip the installation procedure
- To install the web admin interface, check if you have the `gw_web_admin` folder in your `lora_gateway` folder
- If you don't, then update to the latest version (slide 26)
- Then, go into `gw_web_admin` and run `install.sh`
  - `cd gw_web_admin`
  - `sudo ./install.sh`
- Web admin interface tutorial: https://github.com/CongducPham/tutorials/blob/master/Low-cost-LoRa-GW-web-admin.pdf

# Gateway web admin interface (2)

- http://192.168.200.1/admin
  - Login: admin
  - Password: loragateway



# Web admin features

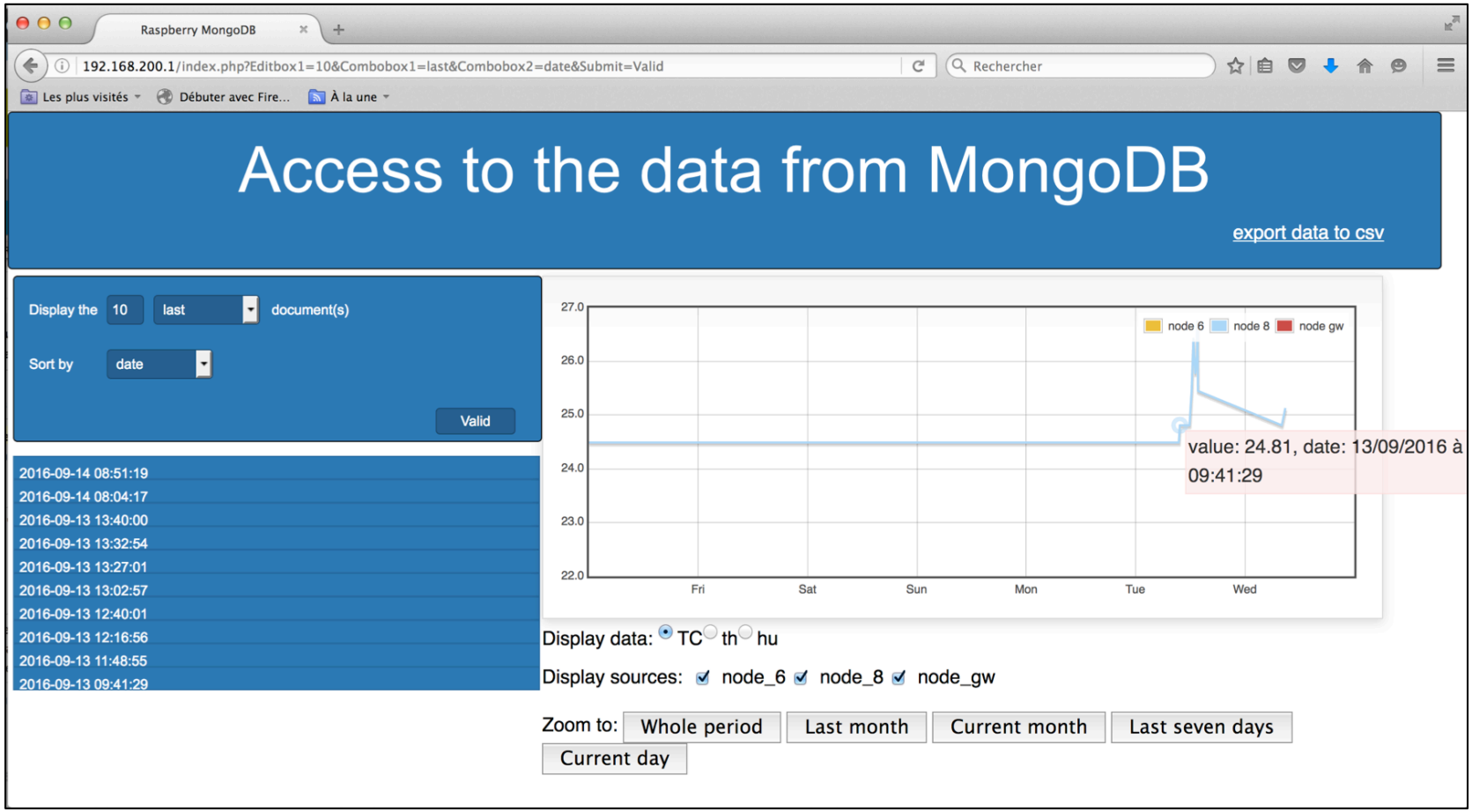
- Currently, you can use the web admin to:
  - Update your gateway with the latest github version and perform the basic configuration procedure. You can preserve your configuration files
  - Configure the gateway as WiFi client to connect to a WiFi network
  - Test Internet connectivity
  - Easily reboot and shutdown your gateway
    - Be carefull, if you shut down the gateway, you need to physically access the gateway to power it it on again
  - Change LoRa mode and frequency
  - Set your gateway id and configure alerting system (mail, SMS)
  - Change the WiFi SSID and password
  - Enable/Disable local AES decryption
  - Enable/Disable ThingSpeak and WAZIUP Orion cloud
  - For ThingSpeak, you can specify a new write key
  - For WAZIUP Orion, you can specify the project name, the organization name and the service tree
    - Fiware-service=project_name
    - sensor_name=organization_name+"_Sensor"
    - Fiware-servicePath='/'+organization_name+service_tree

# Standalone gateway


# Connect to the embedded web server

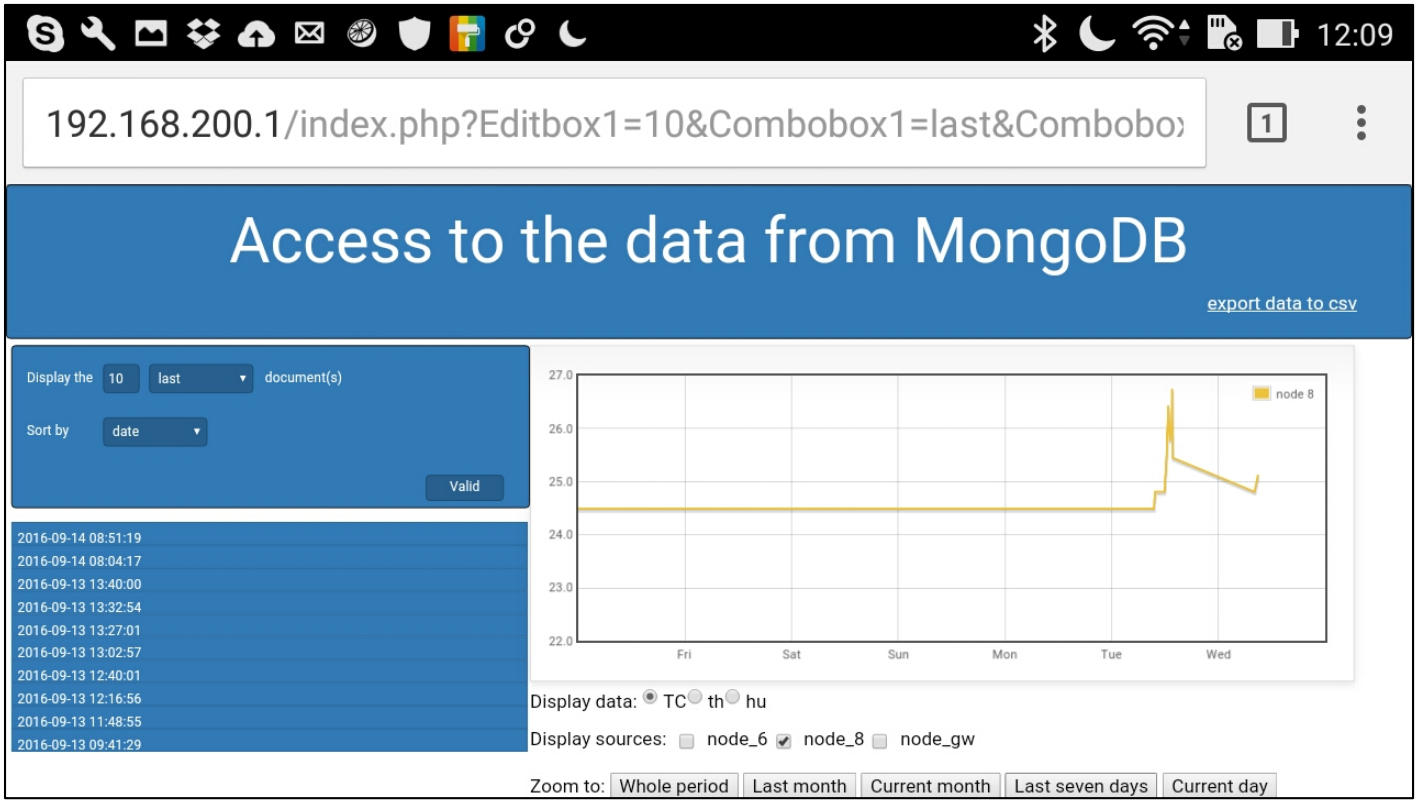
- On the WiFi interface
  - Gateway address is 192.168.200.1
- On the Ethernet interface
  - Gateway address is the IP address assigned by the DHCP server (of your LAN or laptop)
- Choose any of these solutions and open a web browser to enter the gateway IP address in the URL bar
  - http://192.168.200.1

# DATA FROM THE LOCAL WEB SERVER

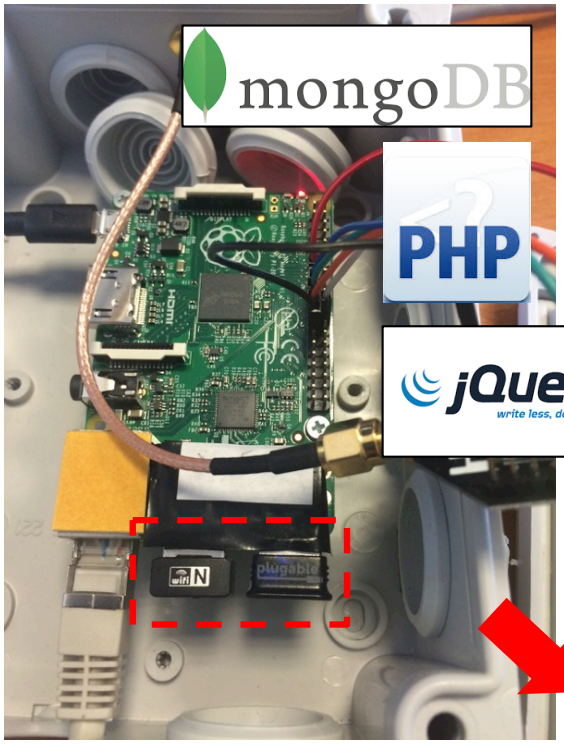


# Visualize it on your smartphone!

- Don't forget to join the WAZIUP_PI_GW_xxxxxxxxxx WiFi

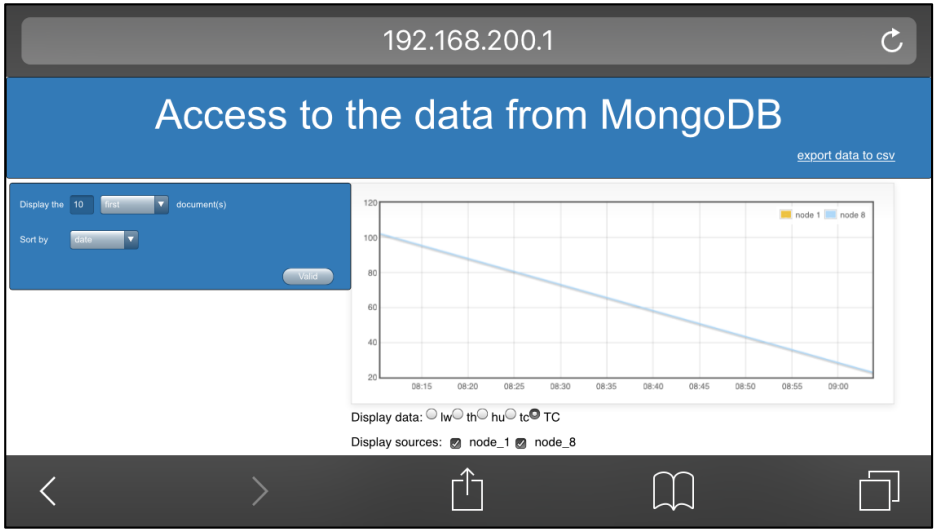


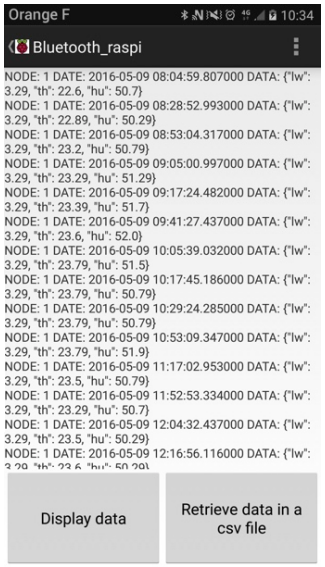
# Running the gateway without internet access

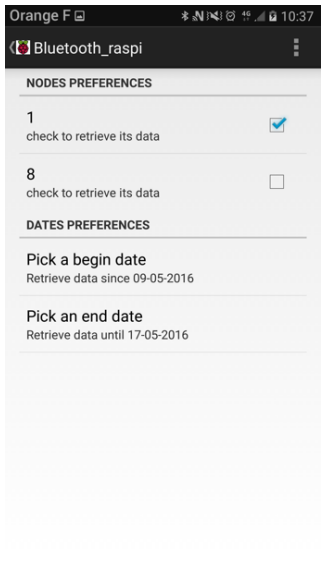


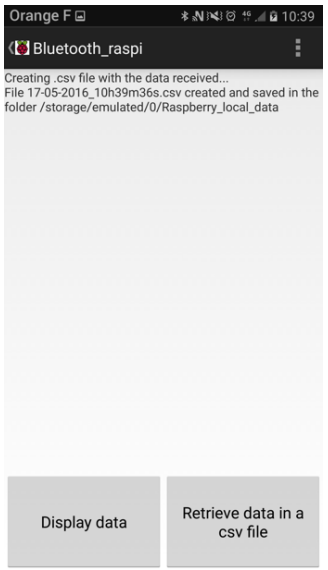




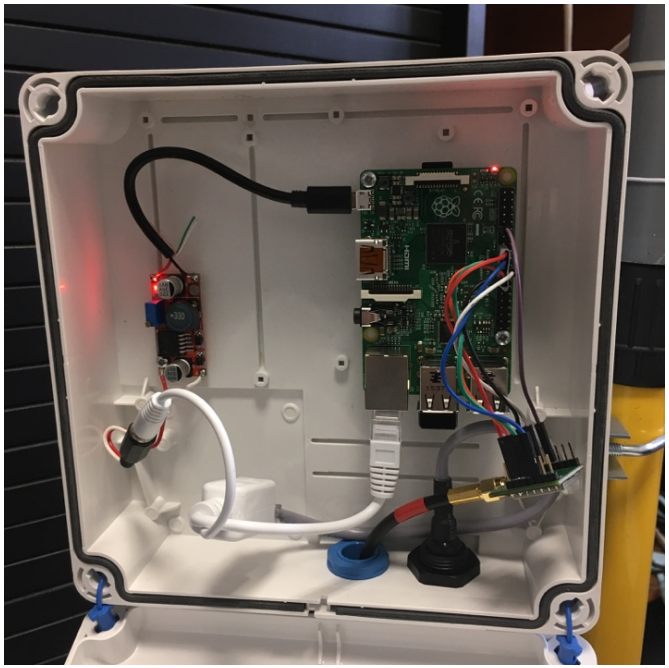


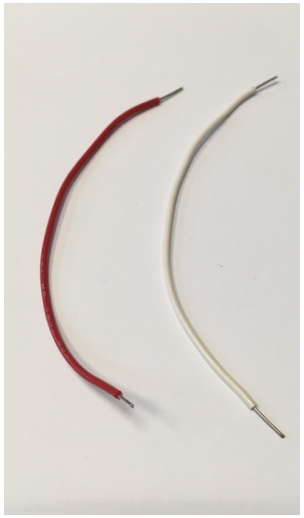
# Improving casing and adding PoE to gateway

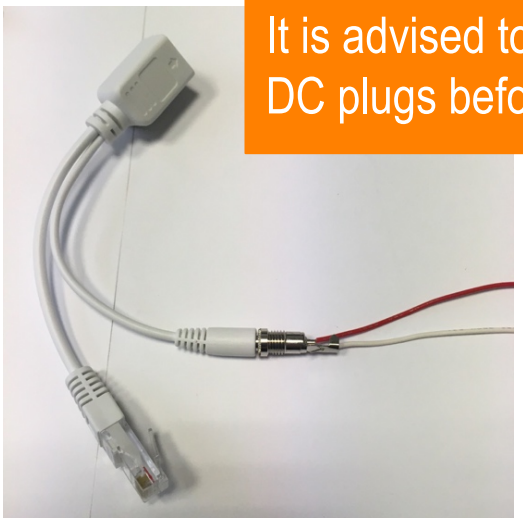


# Overview of the parts

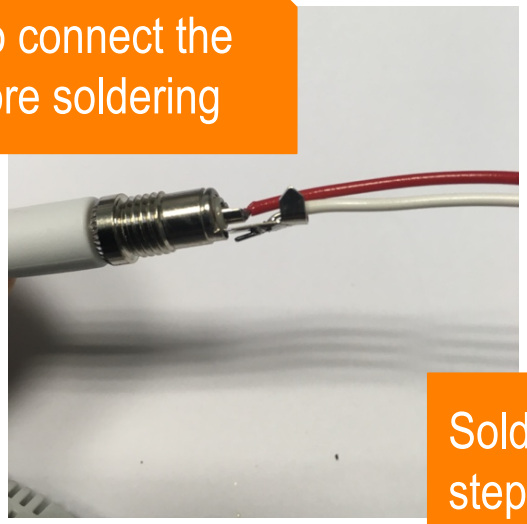
# Fixing the Raspberry to the case

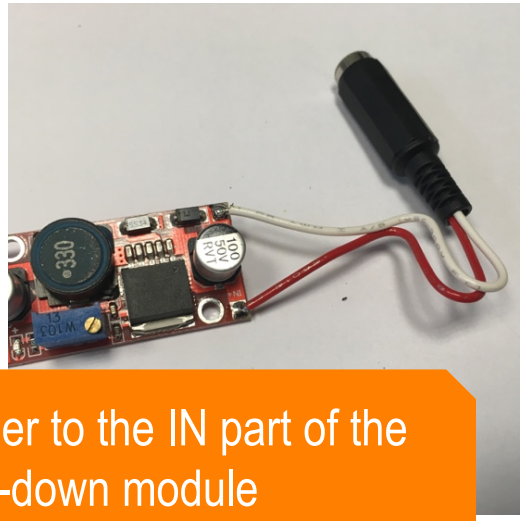
# Prepare the DC Step-Down (LM2596)





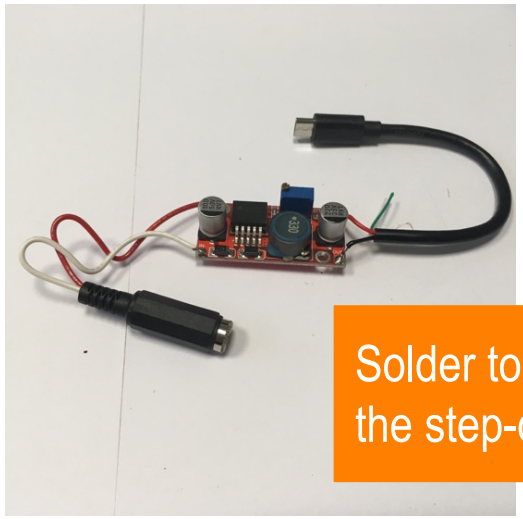
It is advised to connect the DC plugs before soldering





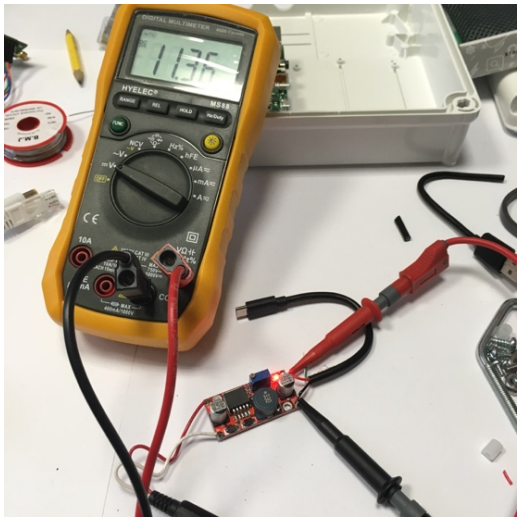
Solder to the IN part of the step-down module

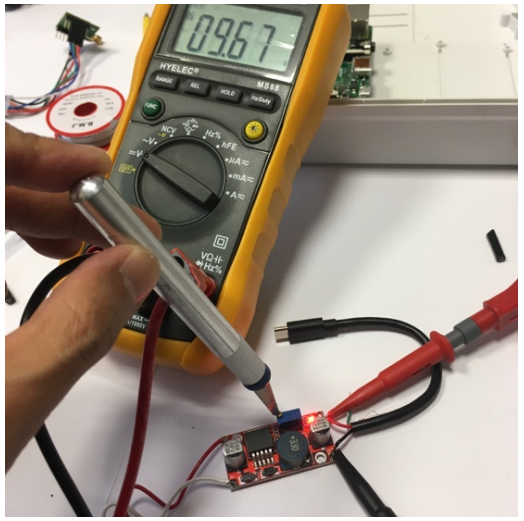
Cut a USB cable, keeping the micro-USB side

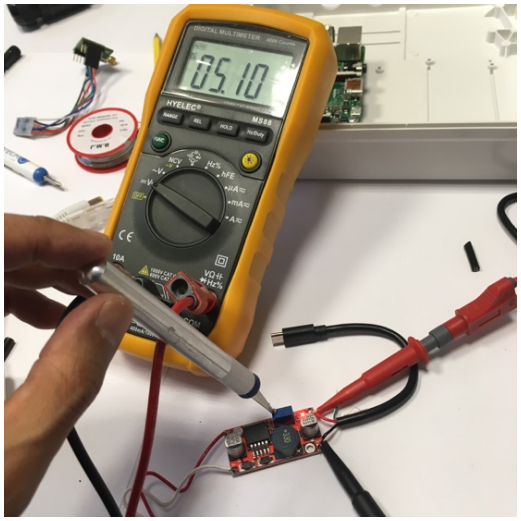


Solder to the OUT part of the step-down module

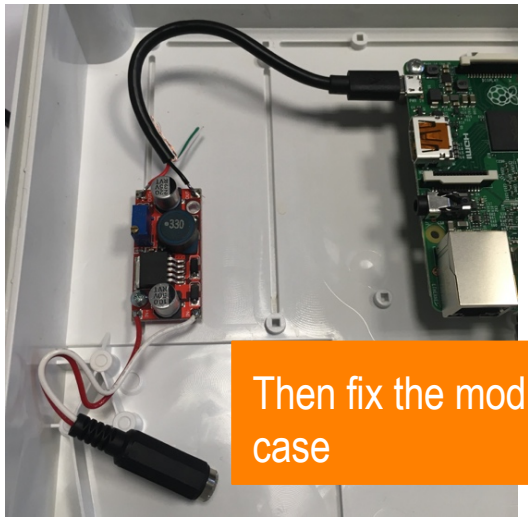
# Setting the step-down module





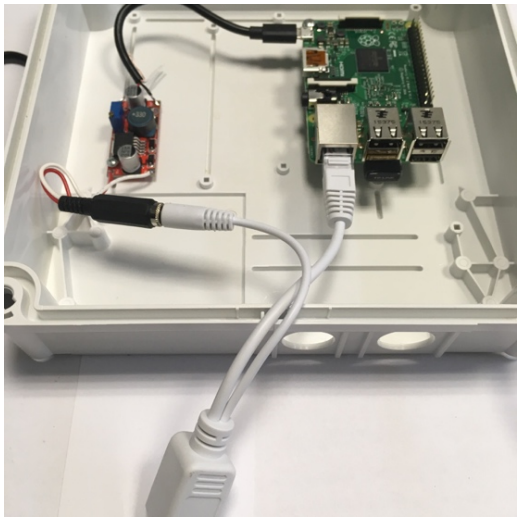


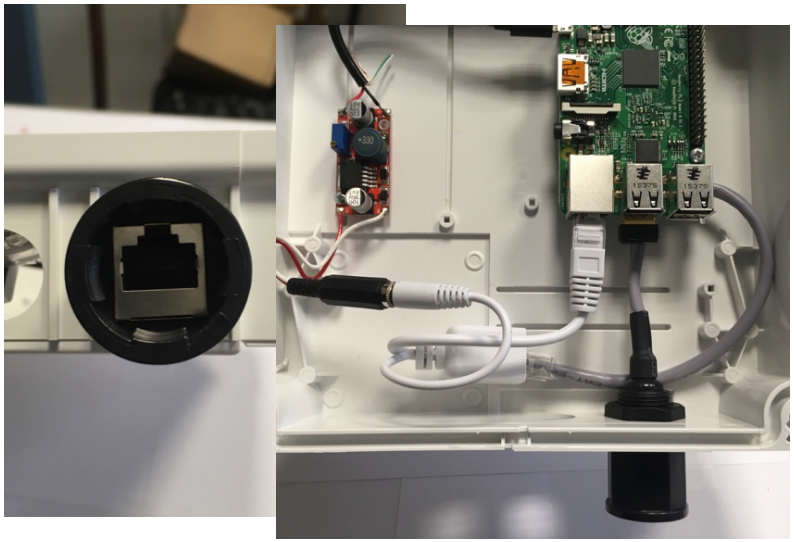
Use for instance a 9v, 12v or 18V AC-DC adaptor, connect to the IN plug, then check the output voltage with a voltmeter and turn the regulation screw until output is about 5.1v.

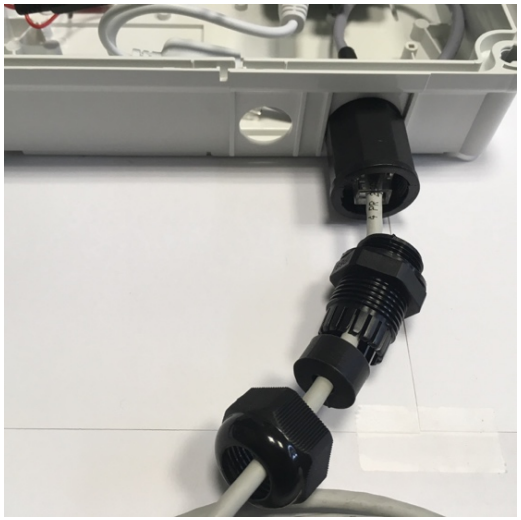


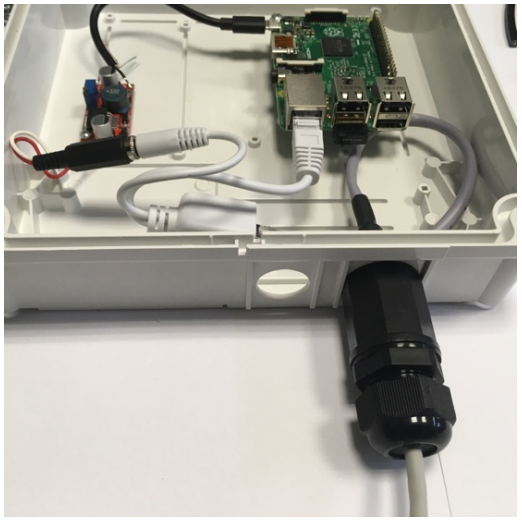
Then fix the module to the case

# Installing the PoE injector and water-resistant ethernet plug



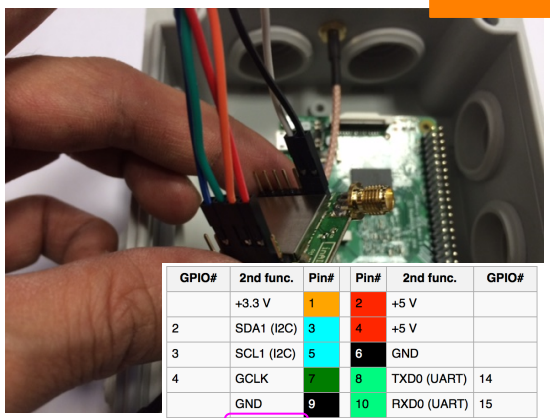




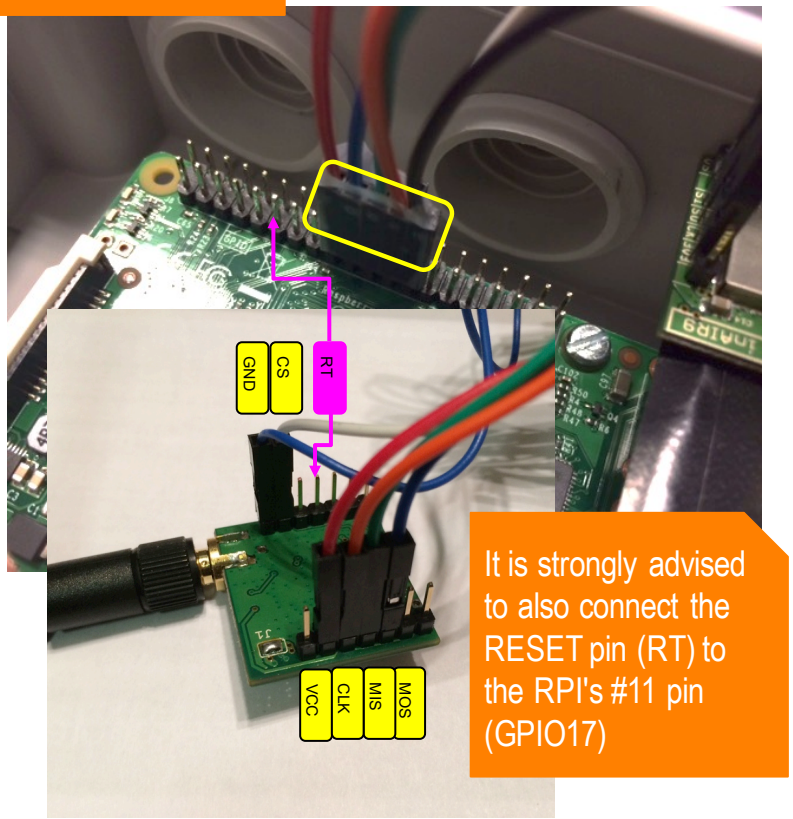


# Connect the radio module

Like previously shown

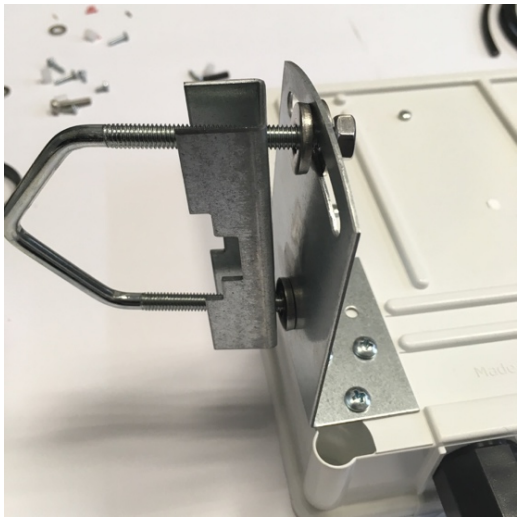


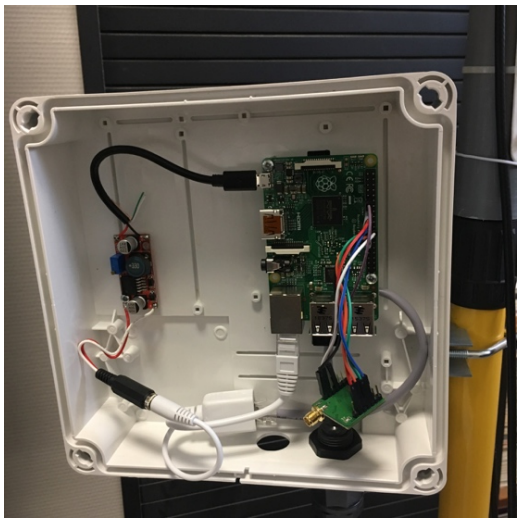
| GPIO# | 2nd func. | Pin# | Pin# | 2nd func. | GPIO# |
|---|---|---|---|---|---|
| | +3.3 V | 1 | 2 | +5 V | |
| 2 | SDA1 (I2C) | 3 | 4 | +5 V | |
| 3 | SCL1 (I2C) | 5 | 6 | GND | |
| 4 | GCLK | 7 | 8 | TXD0 (UART) | 14 |
| | GND | 9 | 10 | RXD0 (UART) | 15 |
| 17 | GEN0 | 11 | 12 | GEN1 | 18 |
| 27 | 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)

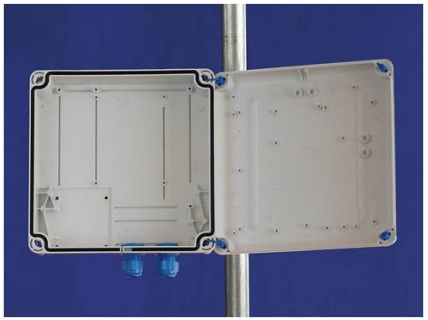
# INSTALL FIXING PARTS OF THE CASE

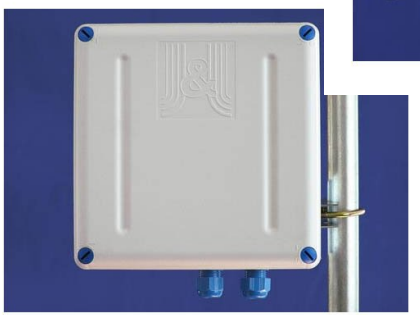





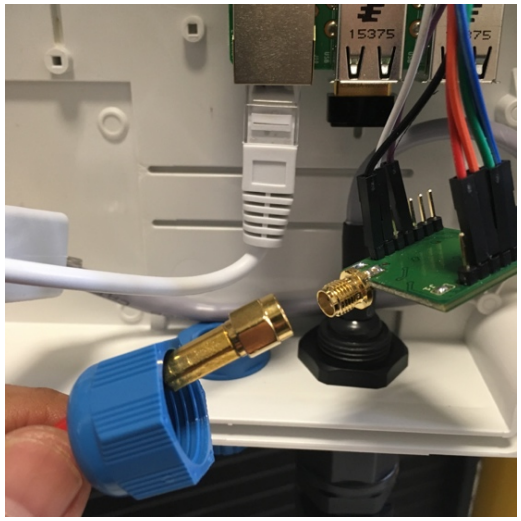
These parts of course depends on the case that you have.

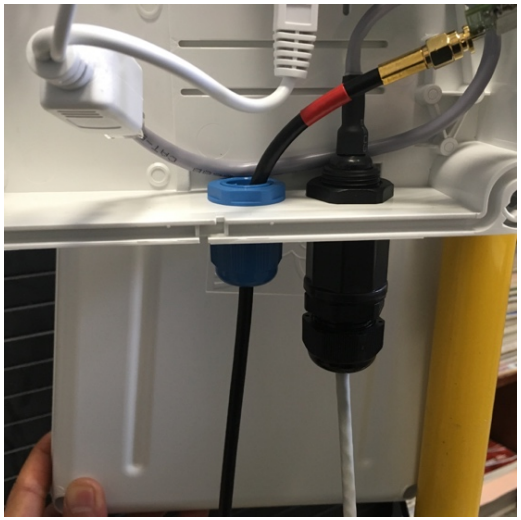
Here we use the GentleBOX JE-200 case from MHzShop.





# 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