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

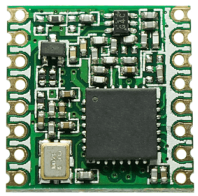
(advanced details on internal architecture)

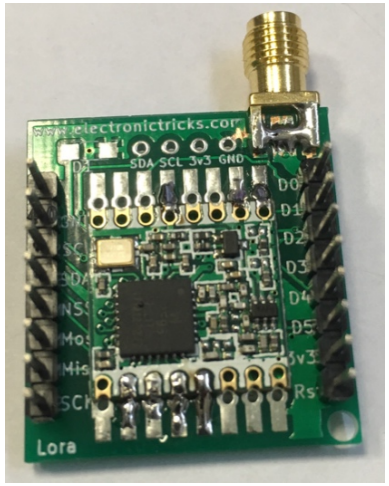
Prof. Congduc Pham
http://www.univ-pau.fr/~cpham
Université de Pau, France

Contents

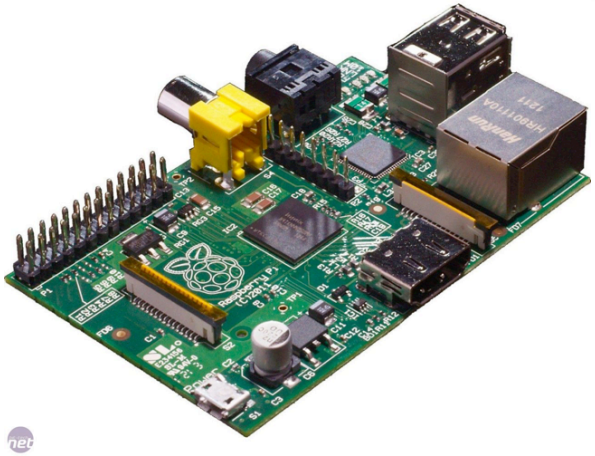
- This tutorial targets user who want to better understand the internal architecture of the low-cost LoRa gateway.
- The hardware platform is a Raspberry PI. RPI0, 1B/B+, 2B, 3B and 4B have been successfully tested
- For end-users, the tutorial on the web admin interface is more adapted
- Here we mainly use ssh to access to more advanced functionalities
- 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 still many issues not described here
- Let's get started…

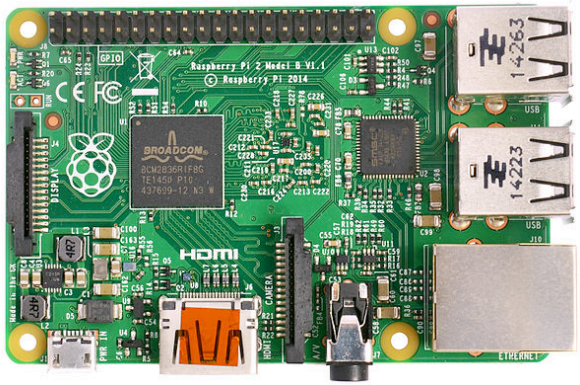
Assembling the hardware

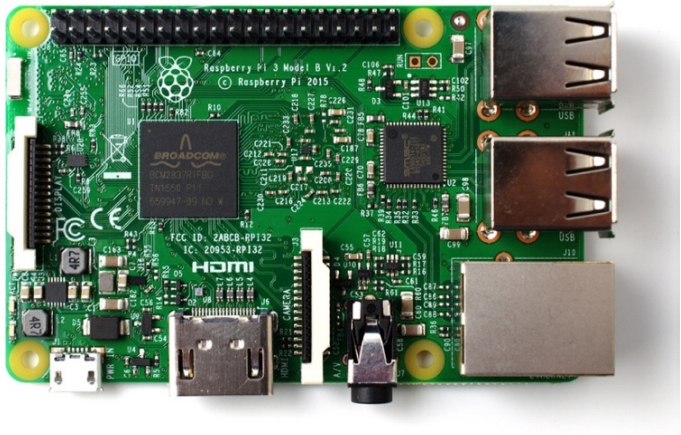


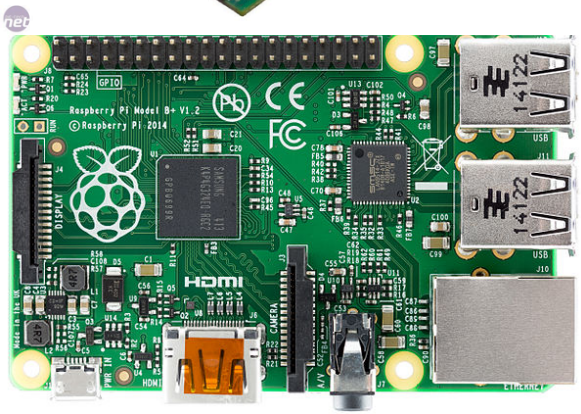


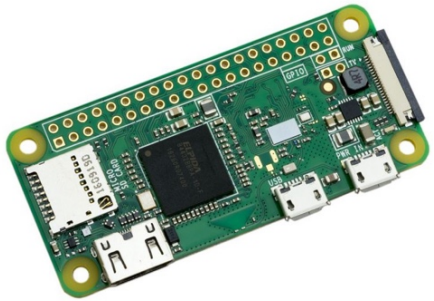
Get the Raspberry





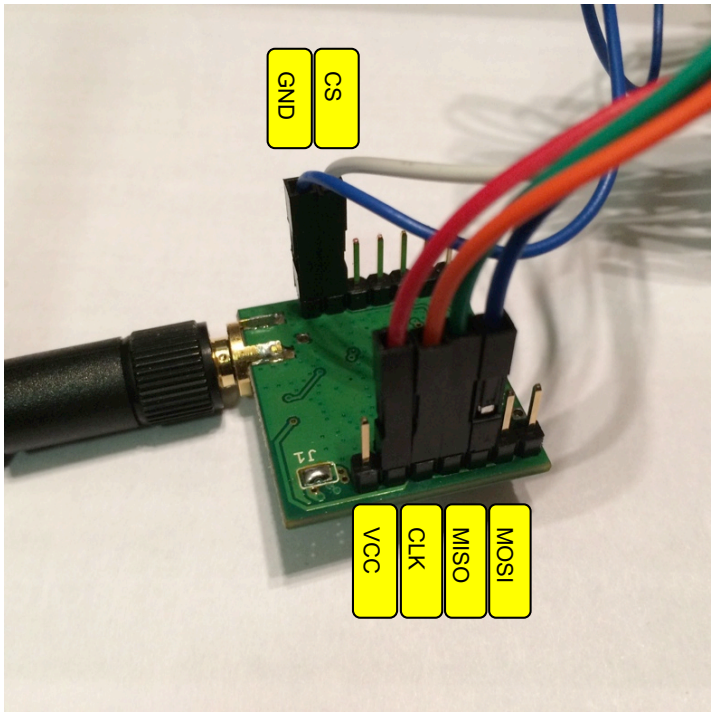


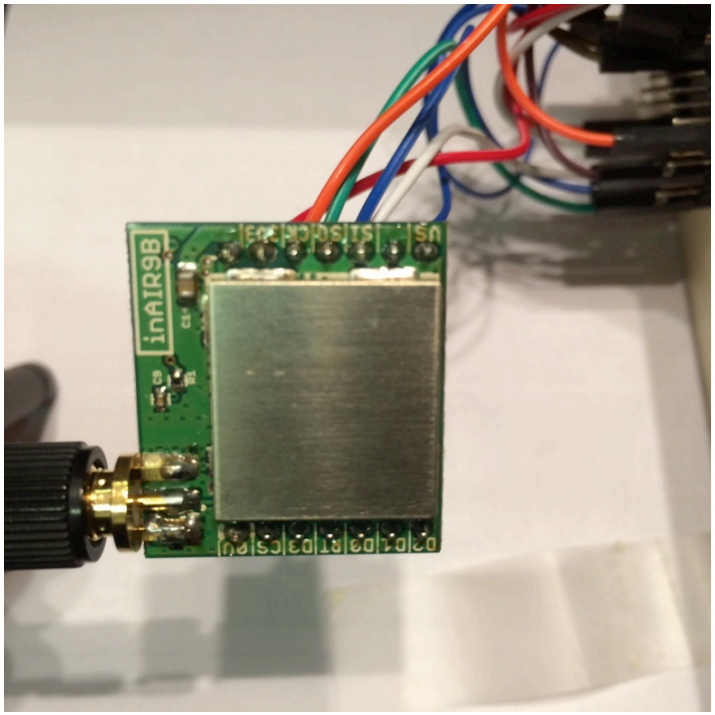




You can use RaspberryPI 1 model B/B+, RaspberryPI 2 model B, RaspberryPI 3 model B/B+, 4B and RaspberryPI Zero (W). The most important usefull feature is the Ethernet interface for easy Internet connection. You can add WiFi with a WiFi USB dongle to use access-point features. With the RPI3/4 & RPI0W, WiFi and Bluetooth are embedded on the board.

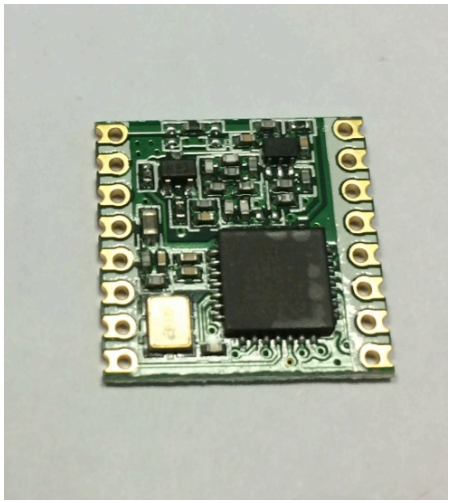
Single-channel GW
The radio module (1)

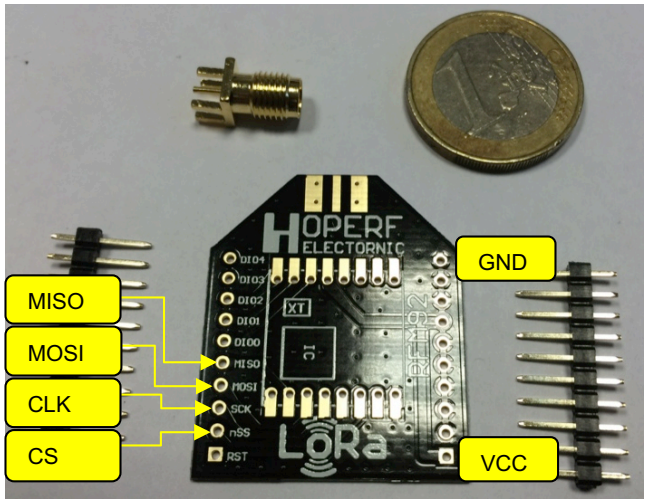




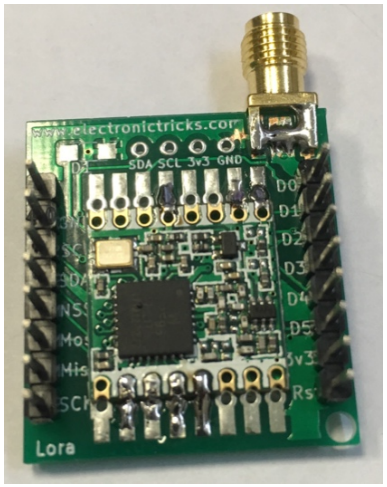
If you go for the inAir (9,9B,4) from Modtronix, the header pins can come fully assembled. Take the 6mm header pins to have enough length to connect F/F breadboard cables (left). Connect the SPI pins with the F/F cables. Try to use different colors. I use the following colors: MOSI (blue), MISO (green),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

Single-channel GW
THE RADIO MODULE (2)





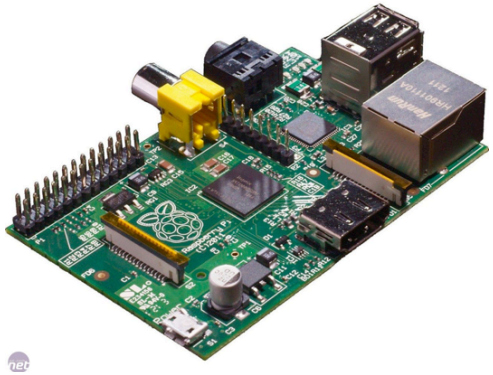
If you take the HopeRF RFM 92W/95W you may need the adaptor breakout and to go though some delicate but simple soldering tasks! It is not difficult but you have to trained a bit before! Then, like for the inAir9, use F/F breadboard cable to connect the SPI pins, using different colors as explained previously.

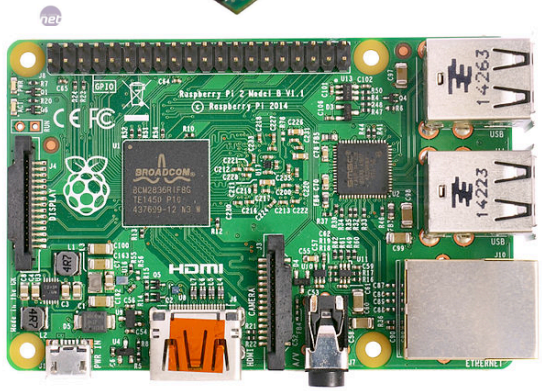


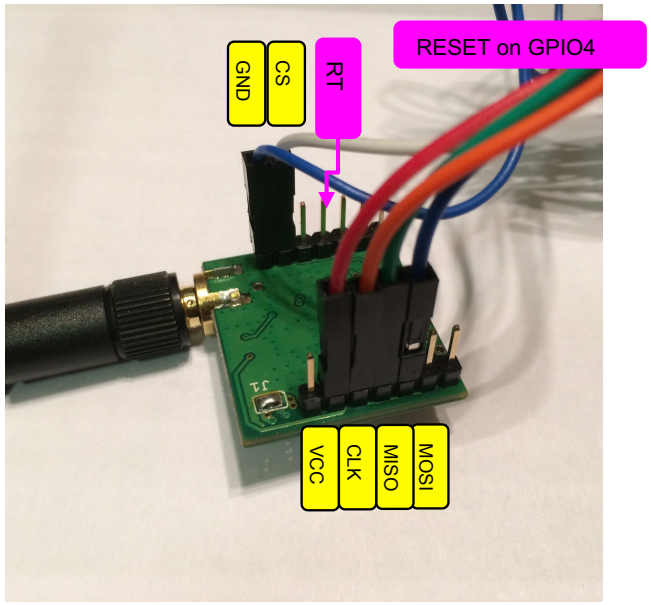
Another breakout from
https://github.com/ccadic/RFM95LORA_Breadboard

Another breakout from Tindie
https://www.tindie.com/products/leonahi/rfm95-lora-breakout-board/

Single-channel GW
Connecting the radio (1)



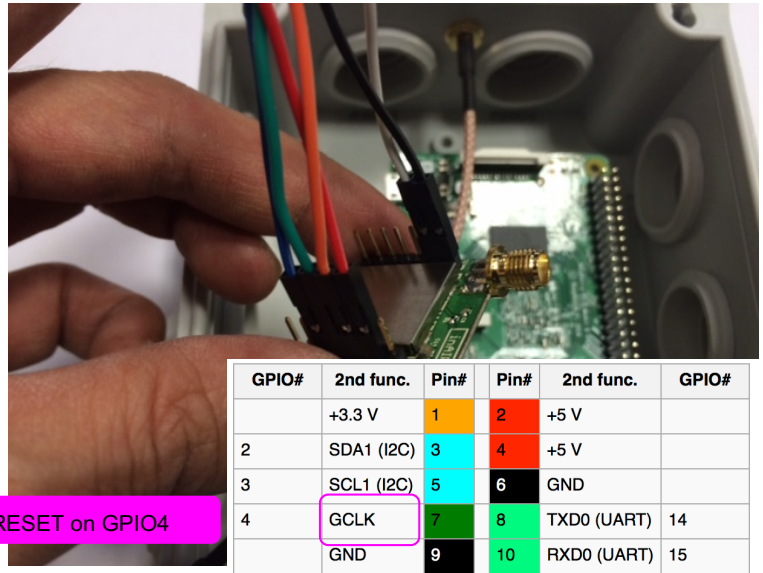




GPIO#	2nd func.	Pin#	Pin#	2nd func.	GPIO#
	+3.3 V	1	2	+5 V	
2	SDA1 (I2C)	3	4	+5 V	
3	SCL1 (I2C)	5	6	GND	
4	GCLK	7	8	TXD0 (UART)	14
	GND	9	10	RXD0 (UART)	15
17	GEN0	11	12	GEN1	18
27	GEN2	13	14	GND	
22	GEN3	15	16	GEN4	23
	+3.3 V	17	18	GEN5	24
10	MOSI (SPI)	19	20	GND	
9	MISO (SPI)	21	22	GEN6	25
11	SCLK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

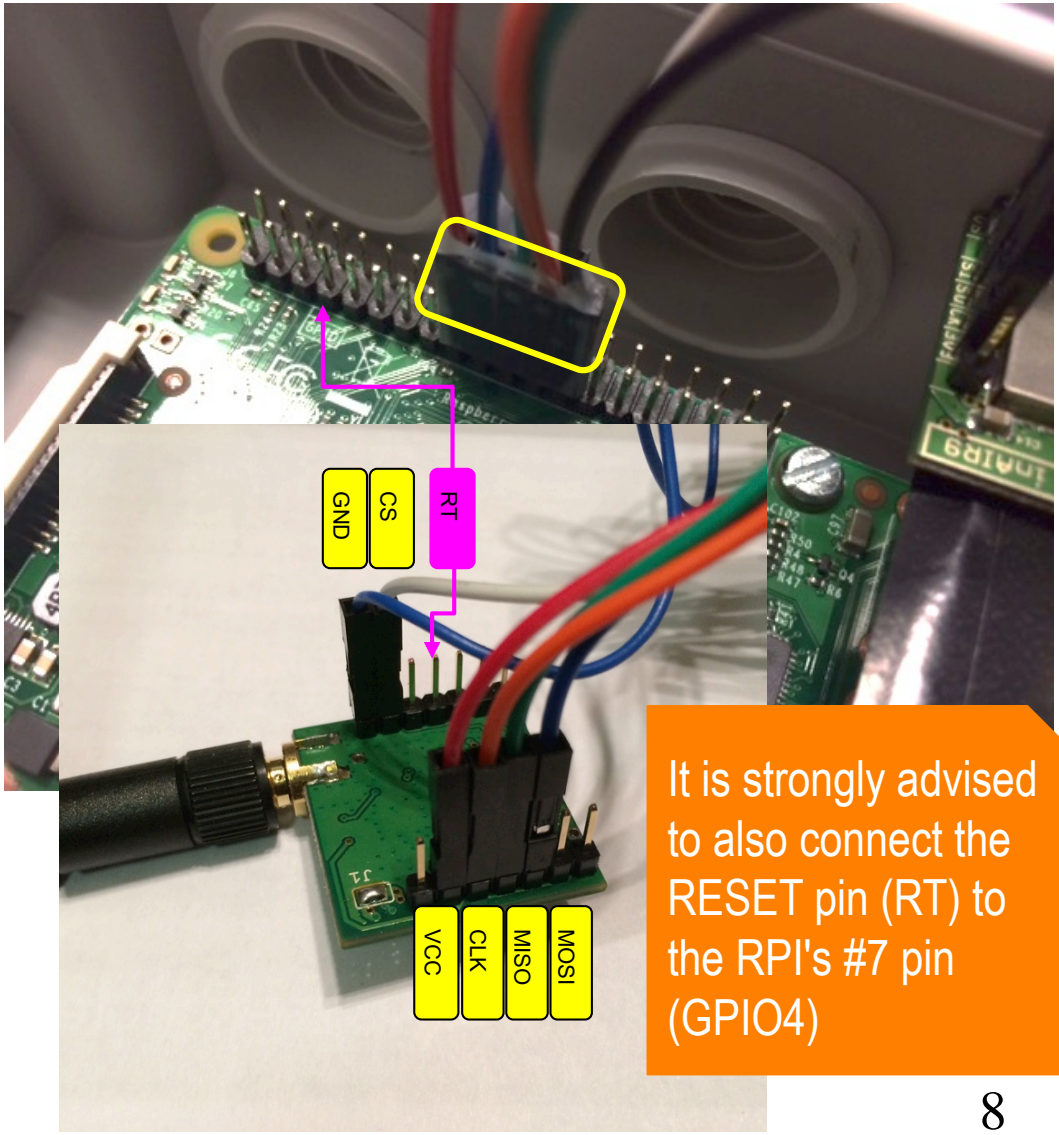
Depending on the model, you can have the « short » or the « long » GPIO interface. However, the SPI pins are at the same location therefore it does not change the way you connect the radio module if you take pin 1 as the reference. Connect the SPI pins (MOSI, MISO, CLK, CS) of the radio to the corresponding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

Single-channel GW
Connecting the radio (2)



RESET on GPIO4

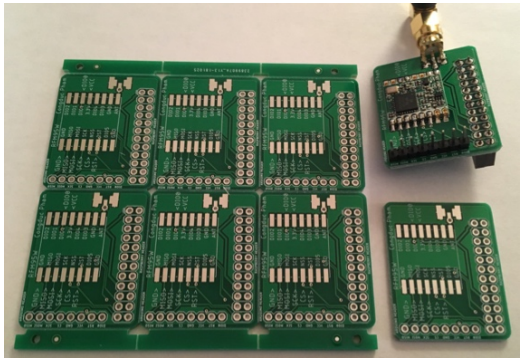
GPIO#	2nd func.	Pin#	Pin#	2nd func.	GPIO#
	+3.3 V	1	2	+5 V	
2	SDA1 (I2C)	3	4	+5 V	
3	SCL1 (I2C)	5	6	GND	
4	GCLK	7	8	TXD0 (UART)	14
	GND	9	10	RXD0 (UART)	15
17	GEN0	11	12	GEN1	18
27	GEN2	13	14	GND	
22	GEN3	15	16	GEN4	23
	+3.3 V	17	18	GEN5	24
10	MOSI (SPI)	19	20	GND	
9	MISO (SPI)	21	22	GEN6	25
11	SCLK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

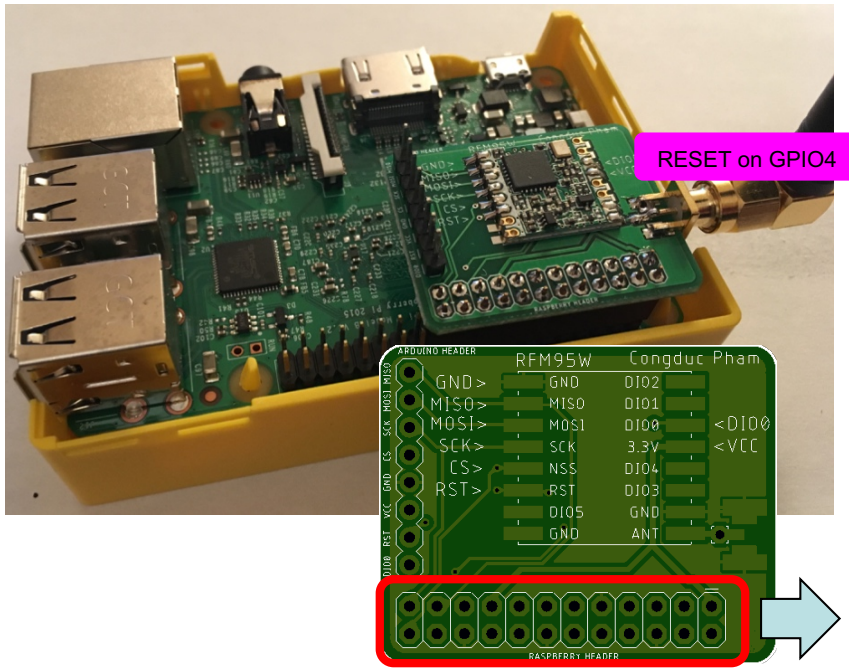


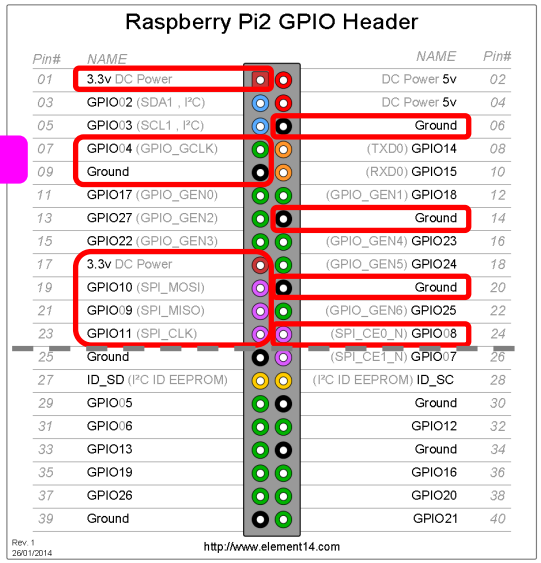
It is strongly advised to also connect the RESET pin (RT) to the RPI's #7 pin (GPIO4)

FREELY AVAILABLE RFM95W BREAKOUT

- We also propose a very simple RFM95W breakout that can be used for gateway and end-device



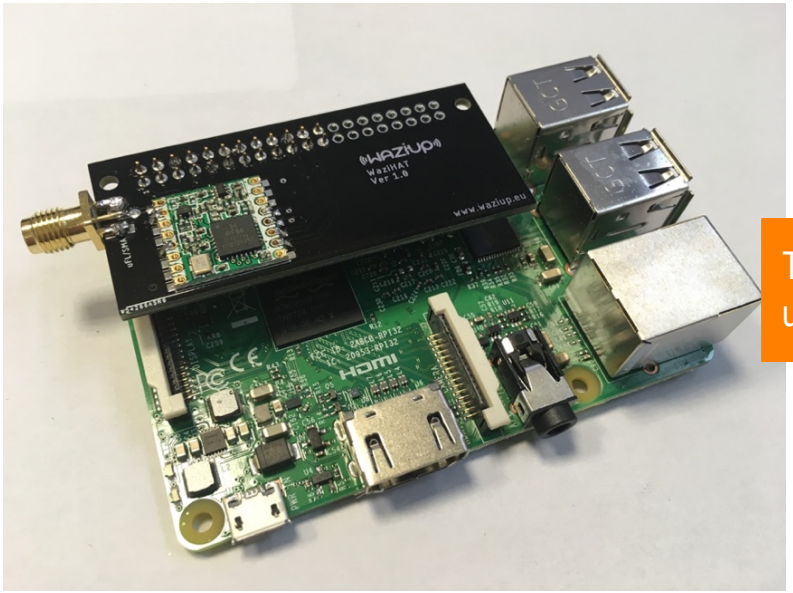


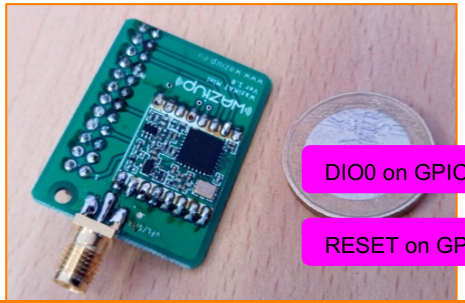


- **The zipped Gerber archive can be freely downloaded from** https://github.com/CongducPham/LowCostLoRaGw

The WaziHat PCB

- WAZIUP has also a simple RFM95W breakout for the gateway
- Contact A. Rahim (arahim"at"fbk.eu) for more information



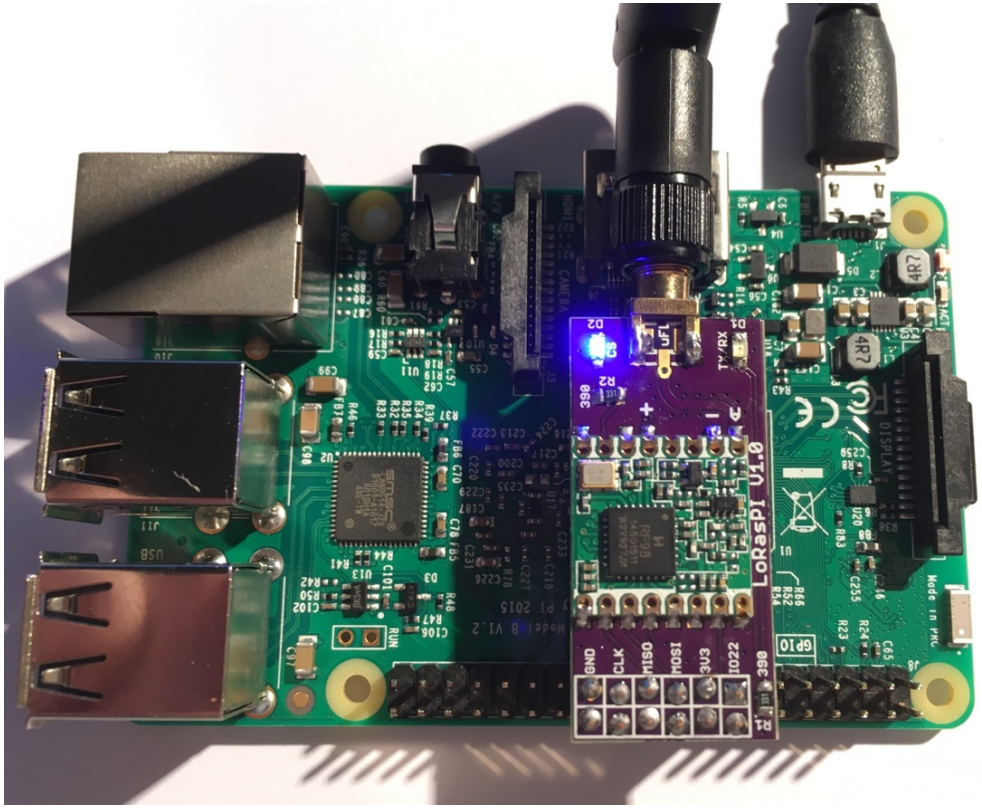


The short WAZIHAT version can be used for end-device as well

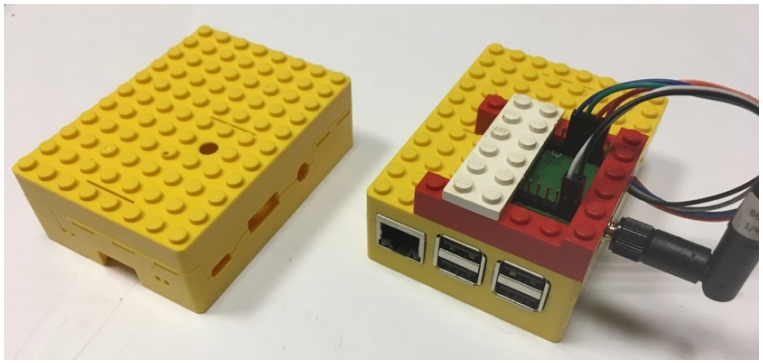
GPIO#	2nd func.	Pin#	Pin#	2nd func.	GPIO#
	+3.3 V	1	2	+5 V	
2	SDA1 (I2C)	3	4	+5 V	
3	SCL1 (I2C)	5	6	GND	
4	GCLK	7	8	TXD0 (UART)	14
	GND	9	10	RXD0 (UART)	15
17	GEN0	11	12	GEN1	18
27	GEN2	13	14	GND	
22	GEN3	15	16	GEN4	23
	+3.3 V	17	18	GEN5	24
10	MOSI (SPI)	19	20	GND	
9	MISO (SPI)	21	22	GEN6	25
11	SCLK (SPI)	23	24	CE0_N (SPI)	8
	GND	25	26	CE1_N (SPI)	7

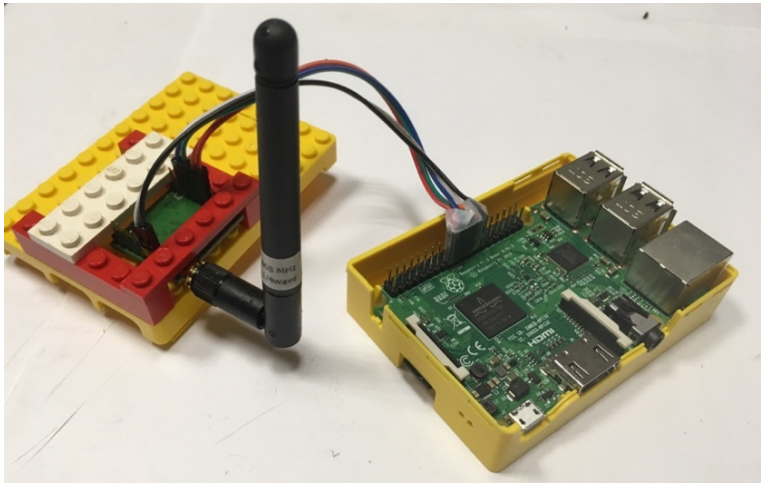
The LoRasPI hat

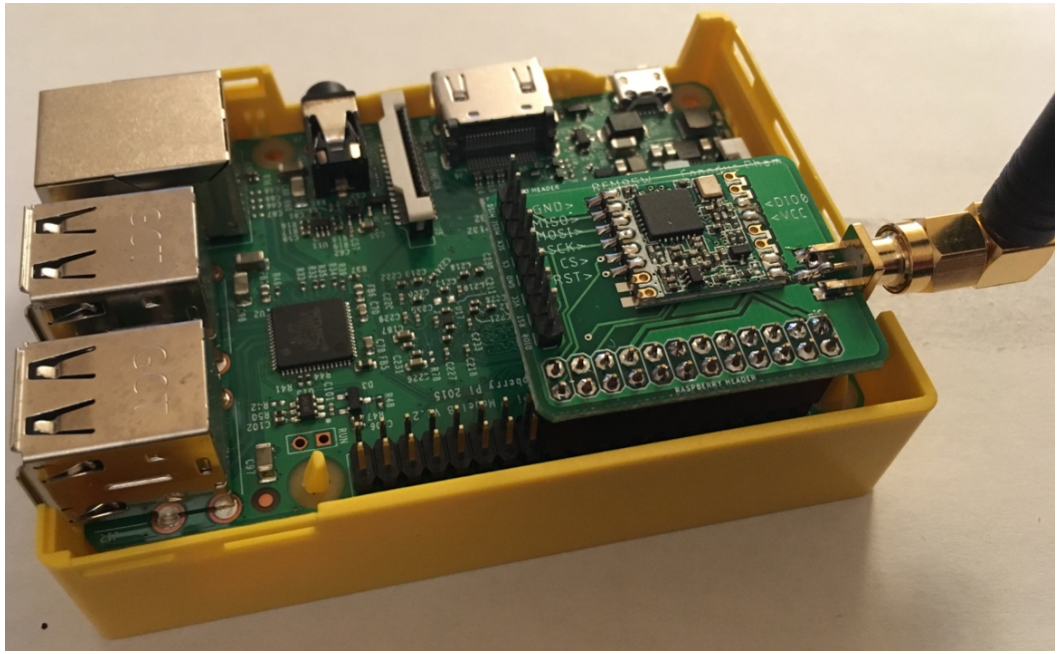
- The LoRasPI hat from https://github.com/hallard/LoRasPI



PUT IT IN A BOX (1)

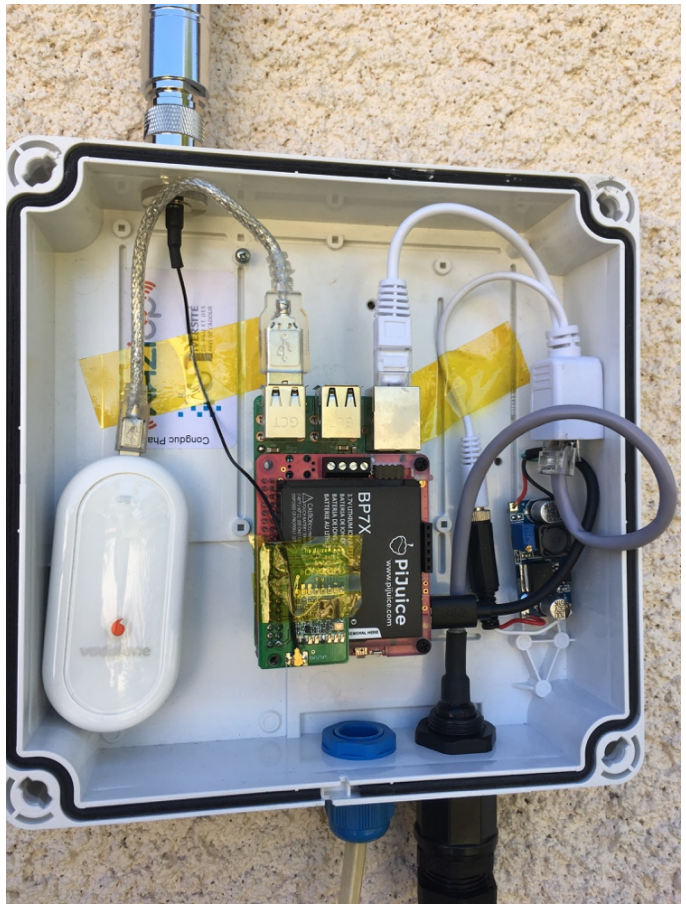






A simple, cheap and funny box is also very suitable for an indoor gateway. Actually, indoor deployment is probably the best option with an outdoor antenna as it will be shown in next slides.

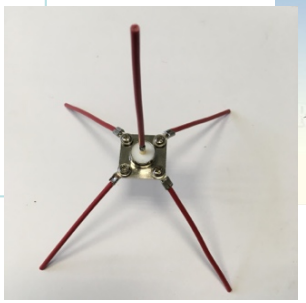
Put it in a box (2)

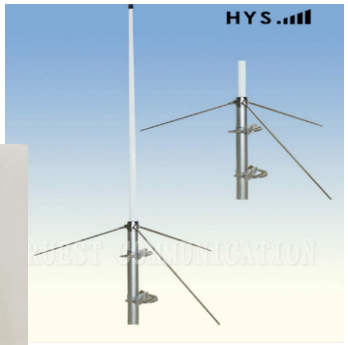


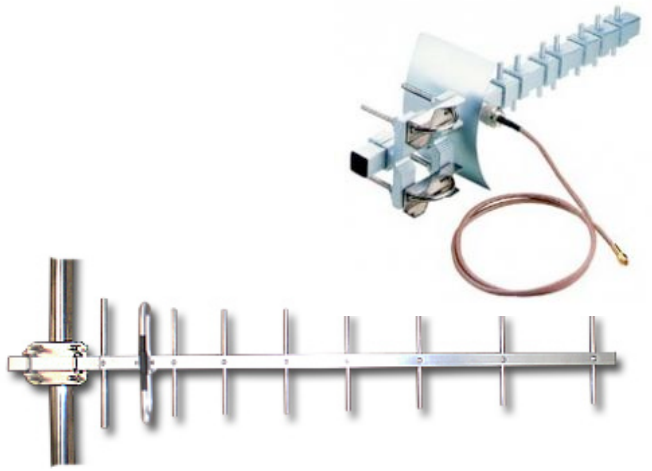
You can have a more integrated version, with a box for outdoor usage and PoE splitter to power the Raspberry with the Ethernet cable. See D-GW-2 course Building an Outdoor Gateway

ANTENNAS FOR GATEWAY

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







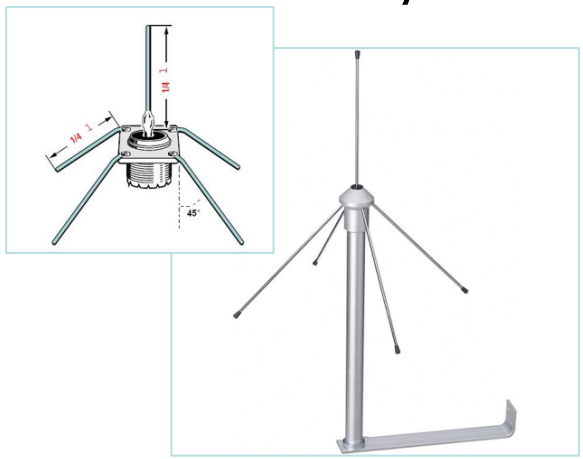
USING A CABLE ANTENNA

- Using an extension coaxial cable between the antenna and the radio module greatly ease the deployment of the gateway **but**:
  - Take a good quality cable (e.g. RG58 minimum) to limit attenuation
  - The antenna cable should not be too long to avoid high attenuation: 2m-5m
  - A simple ¼ wave monopole antenna WILL NOT provide good performance, TAKE a sleeve dipole if you need something compact

Look at the antenna cable tutorial for instructions on how to build your own cable (with adequate connectors) at the correct length.

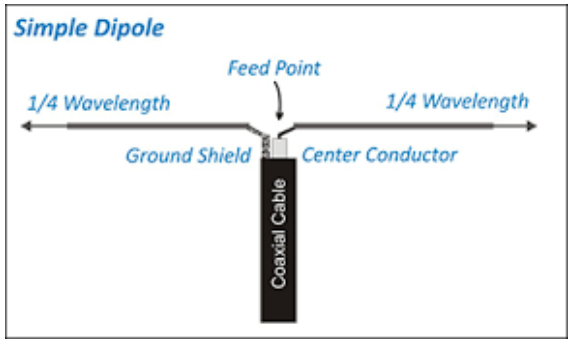
Antenna with a coaxial cable

- At the end of a coaxial cable, it is possible to connect a ground plane antenna (usually ¼ wave) or a ½ wave dipole antenna.



Ground plane

Sleeve dipole



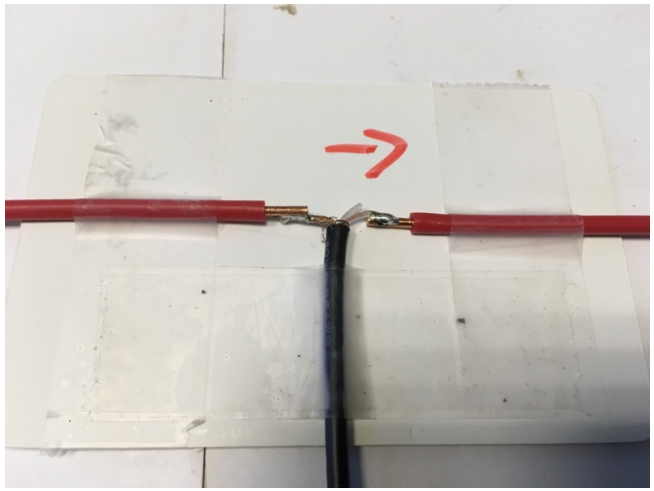
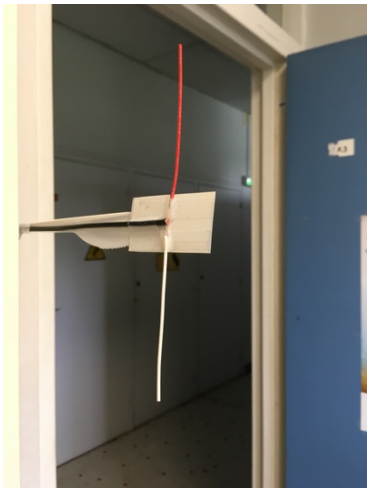
Simple dipole

More complex: collinear, array,…

- Some of them are easy to build (ground plane and simple dipole) and there are many tutorials.

Simple ½ wave dipole antenna

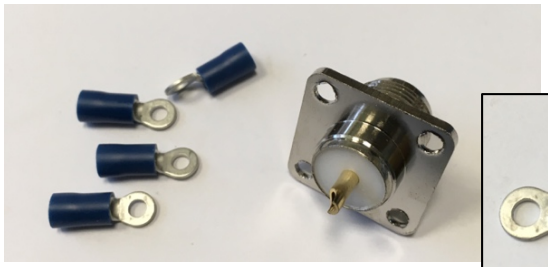
- A very simple dipole can be made with 2 pieces of ¼ wave wires. ¼ wave in 868MHz is about 8.2cm.

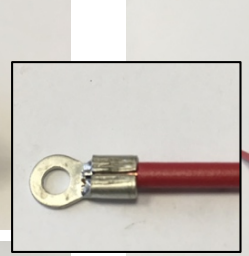



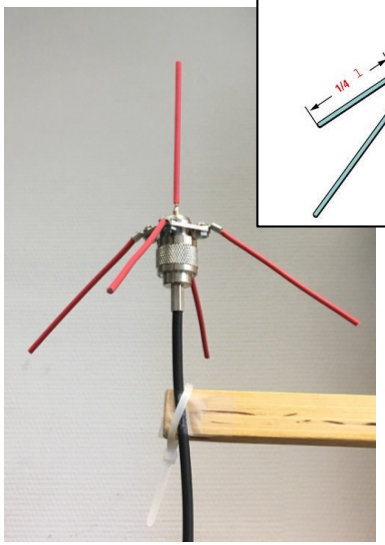

- There is no balun here but it is still better than the ¼ wave monople if a coaxial cable is used
- You can buy a 3m RG58 cable (SMA-m to SMA-f for instance), keep the male side, cut the female side and solder the core conductor and the braid as shown.

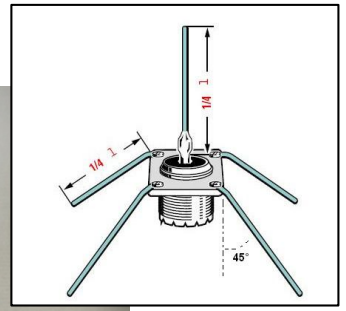
Simple ¼ wave ground plane antenna

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





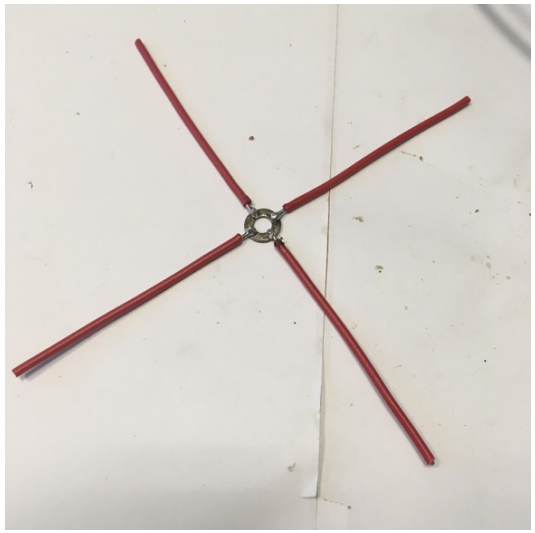
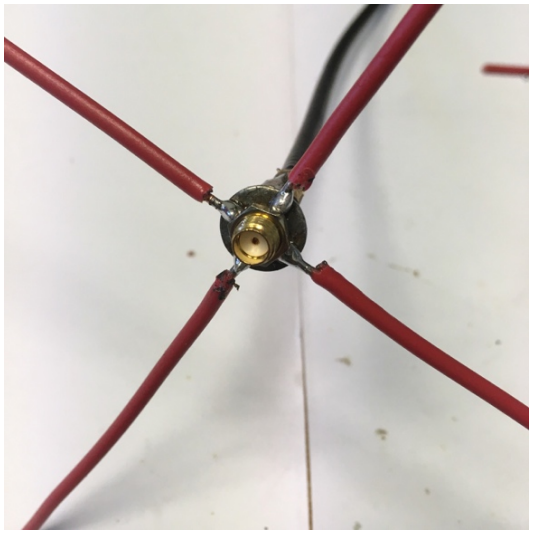
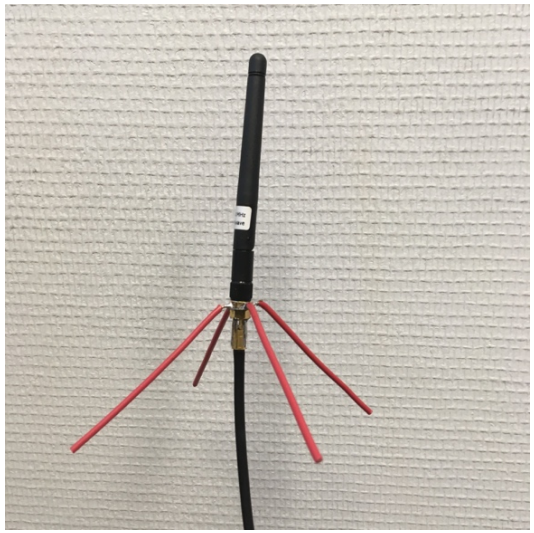




- You can buy a 3m-5m RG58 cable with an SMA-male at one end and a male N-connector at the other end. Or build your own cable.

Even simpler ¼ wave ground plane antenna

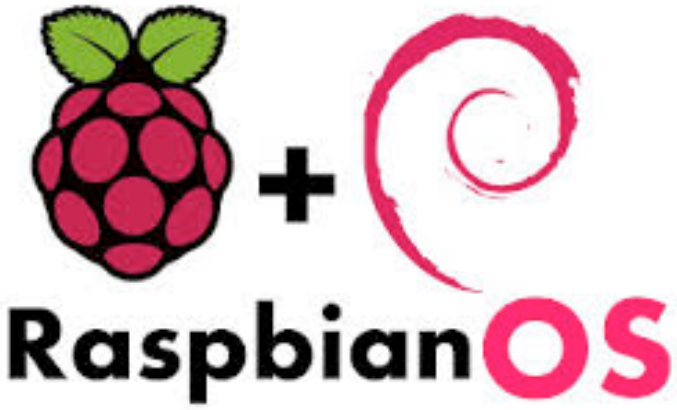
- With an existing SMA-m/SMA-f cable, you can also build a ground plane antenna by adding 4 radiant wires to the ¼ wave monopole.

- This is a cheaper solution for sensing devices

DIPOLE AND GROUND PLANE

Getting, compiling & installing the software



Flashing the OS

http://cpham.perso.univ-pau.fr/LORA/WAZIUP/raspberrypi-buster-WAZIUP-demo.iso.zip

- An SD card image with a Raspberry Raspbian Buster 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.

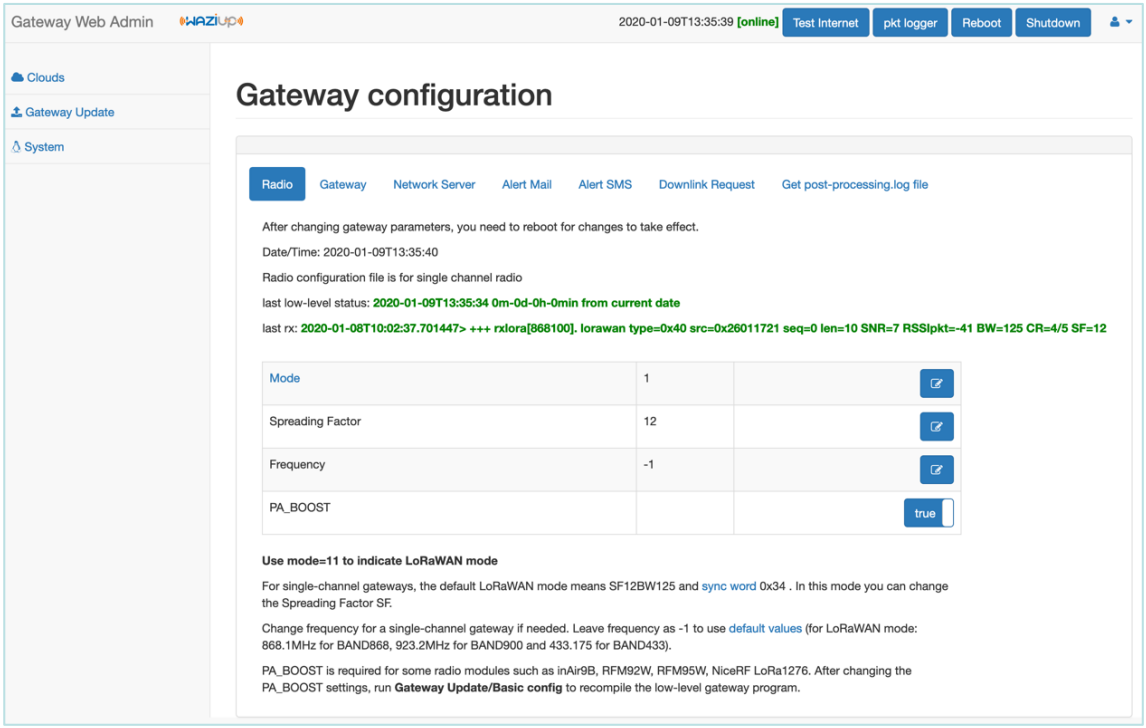
- Look at https://www.raspberrypi.org/documentation/installation/installing-images/ to see the procedure depending on your OS. 7948206080 bytes should be written, otherwise you may have a problem.
- Once flashed, insert the SD card and power-up the Raspberry-based gateway.

Gateway access & configuration interfaces

- There are 2 gateway configuration interfaces
  - A web admin interface
  - A command line interface that needs ssh
- The web interface is sufficient for most users
  - Easy basic configuration and easy update
  - Pre-defined cloud configuration
  - dedicated course: D-GW-4 Gateway Web Admin Interface
- The command line interface has some more options and can easily be extended
- We are going to describe the command line interface and some of the gateway's internal

Gateway web admin interface

- http://192.168.200.1/admin (with WiFi connection)
  - Login: `admin`
  - Password: `loragateway`

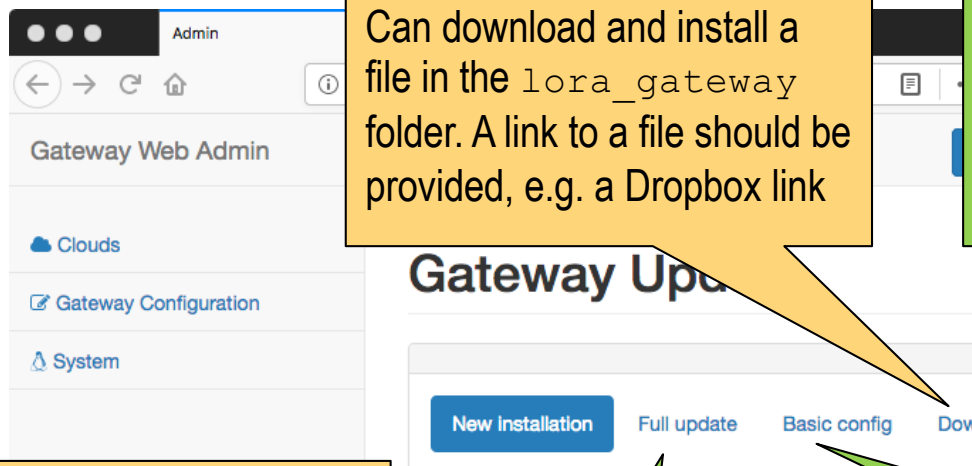


Gateway update

- The gateway must be updated to the latest version
- Internet access for the gateway is necessary
- The update procedure can easily be done with the web admin interface, connect to the gateway WiFi first
- The update steps are
  1. Full Update
  2. Basic Config
  3. Update Web Interface
- We recommend using the web interface for updates

GATEWAY UPDATE PAGE

❑ Gateway update section



Can download and install a file in the `lora_gateway` folder. A link to a file should be provided, e.g. a Dropbox link

Update the web admin interface after an update of the distribution to install the last version of the web admin interface.

It is recommended to run Update web admin right after Full update or New installation
Then reload the page.

3

Install a new gateway by removing the existing `lora_gateway` folder, all existing configuration files will be overwritten.

If you install a new gateway with our SD card image, you can use this option.

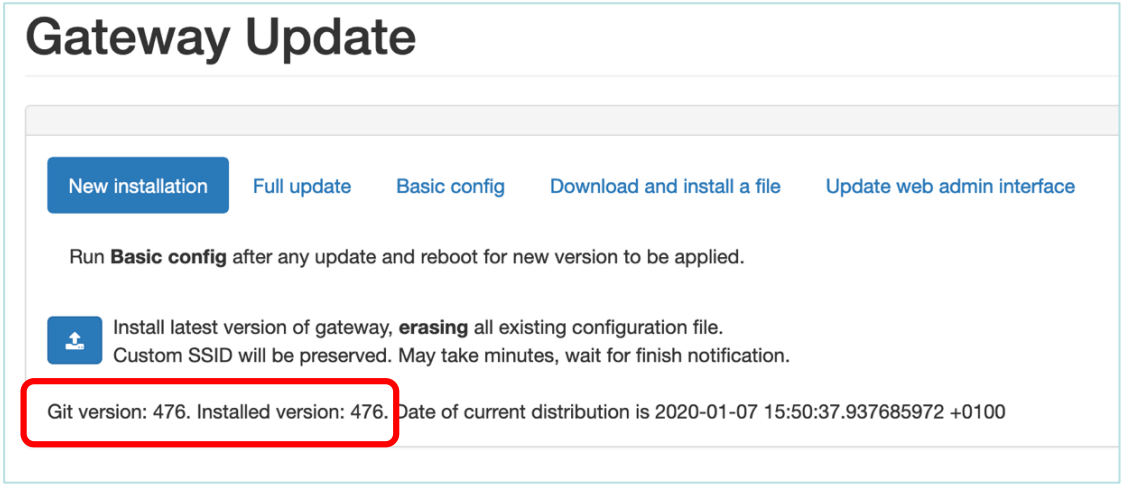
1

Update with latest version on github, all your configuration files will be kept. This is the recommended option.

2

Compile and configure the gateway (to set the gateway id & the WiFi access point SSID). This is also required if you install a new gateway using the provided SD card image. It is recommended to run Basic config right after Full update or New installation

Software version number



- The software version number on github and the installed version number are displayed
- Click on Test Internet to obtain the latest software version number on github

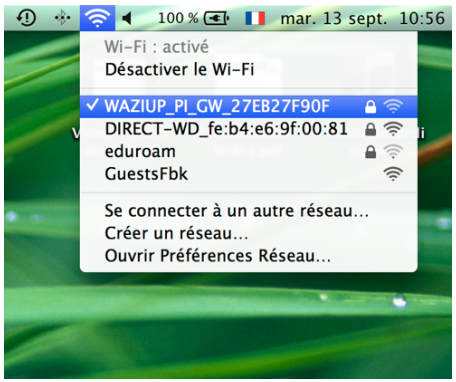
SSH TO THE GATEWAY

- The provided SD 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_xxxxxxxxxxx` WiFi
- WiFi password is `loragateway`
- Then `ssh pi@192.168.200.1`
- Login password is `loragateway`

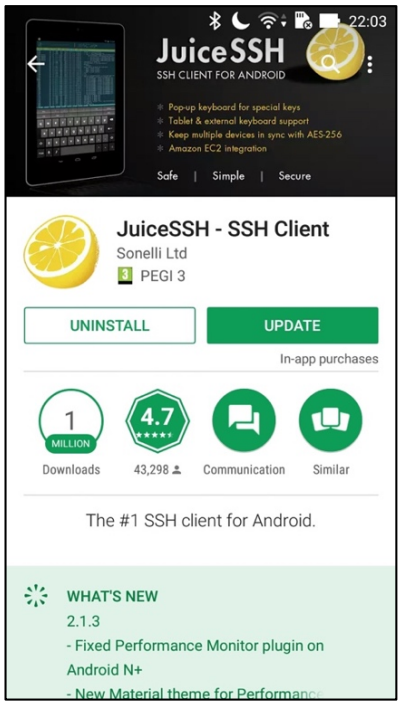
You can use an iOS or Android smartphone or tablet to connect to the gateway with an SSH client app! See next slide.

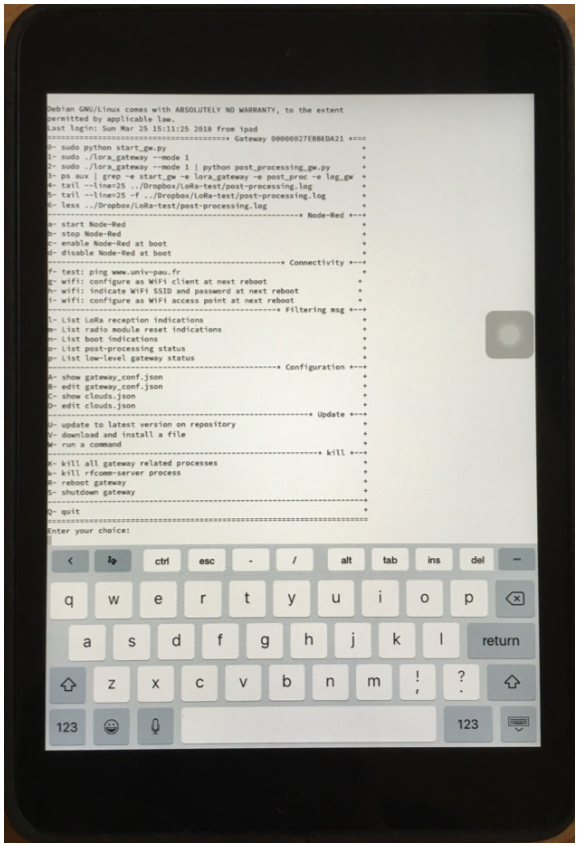



```
cpham — pi@raspberrypi: ~/lora_gateway —
MacBookProRetina-de-Congduc-Pham:~ cpham$ ssh pi@192.168.200.1
pi@192.168.200.1's password:

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 iOS/Android smartphone or tablet

- On iOS we tested Termius
- On Android we tested JuiceSSH



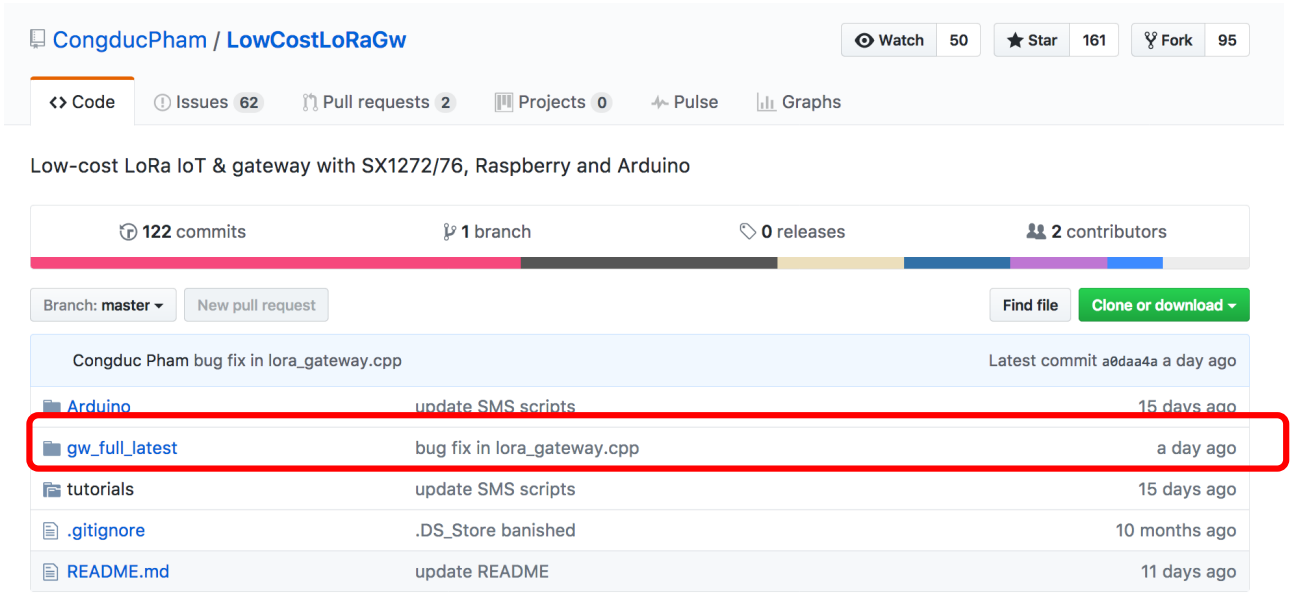


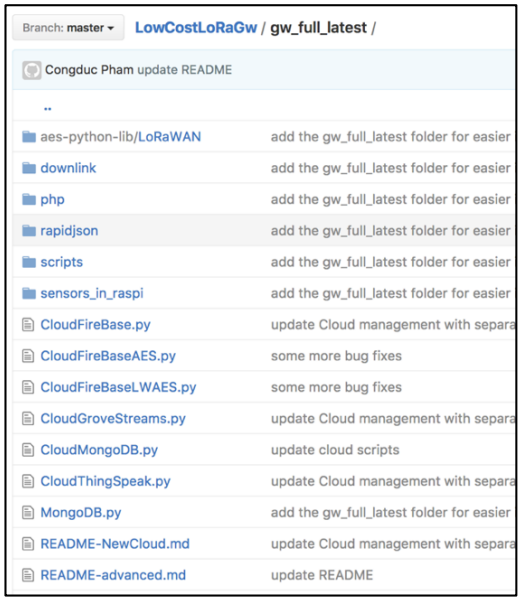
Gateway's simple command interface

- 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                    +
-----------------------------------------------* 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                                                 +
-------------------------------------------------------* ngrok *--+
M- get and install ngrok                                            +
N- ngrok authtoken                                                  +
O- ngrok tcp 22                                                     +
-------------------------------------------------------* 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:
```

MANUAL NEW INSTALL OF GATEWAY DIRECTLY 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
```

MANUAL NEW INSTALL OF GATEWAY USING `update_gw.sh` SCRIPT (1)

- The gateway can also be installed/updated to the latest version with the `update_gw.sh` script (this script is called by the web interface).
- 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
```

Manual new install of gateway using `update_gw.sh` script (2)

- 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 new install
- The gateway will obtain the latest distribution from our github repository and will create a new `lora_gateway` folder
- A full update without deleting the existing `lora_gateway` folder preserves existing configuration files

Manually 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 an RPI 2 or RPI3, then type:

`>make lora_gateway_pi2`

If you have an RPI 4, then type:

`>make lora_gateway_pi4`

GATEWAY STARTUP PROCEDURE

- The gateway software is **launched** when the Raspberry is powered on, i.e. booting
  - `/home/pi/lora_gateway/scripts/start_gw.sh` which starts the gateway has been added in `/etc/rc.local`
  - `start_gw.sh` performs some initial tasks and finally runs `python /home/pi/lora_gateway/start_gw.py`
  - `start_gw.py` parses the configuration files to launch the low-level gateway (`lora_gateway`), the post-processing stage (`post_processing_gw.py`) and the log service
- 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": "000000XXXXXXDEF0",
    "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...
```

- At boot, the gateway software is already launched
- **IMPORTANT NOTICE**: Do not launch a new gateway instance with an existing one as there will be conflict on the SPI bus.

Configuring a new gateway

- 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 6 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 N

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

Gateway's ID

- The gateway's ID is derived from the RPI MAC address, using the 6 bytes

```
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
```

- Here the gateway's ID is 0000B827EB795C47
- This information is stored in `gateway_id.txt` and in `gateway_conf.json`
  - `"gateway_conf": {`
    `"gateway_ID":"0000B827EB795C47",`
    `…`
- `basic_config_gw.sh` also builds the MD5 hash version of the gateway's ID and stores it in `gateway_id.md5`
- The ID is re-created at boot so that a newly installed gateway has the correct ID
- It is recommended to use this default gateway ID

START THE COMMAND LINE 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 0000B827EB795C47 *===
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                    +
-------------------------------------------------* 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                                                 +
-----------------------------------------------------------* ngrok *--+
M- get and install ngrok                                            +
N- ngrok authtoken                                                  +
O- ngrok tcp 22                                                     +
-----------------------------------------------------------* 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:
```

UPDATE PROCEDURE

- You can use option **U** to update from repository and **still keep all your configuration files**: `gateway_conf.json`, `clouds.json`, `radio.makefile` and `key*`
- This simply calls `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

```
pi@raspberrypi:~/lora_gateway $ ./cmd.sh
=====================================* Gateway 00000027EB795C47 *===
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                    +
-------------------------------------------------* 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                                                 +
-----------------------------------------------------------* ngrok *--+
M- get and install ngrok                                            +
N- ngrok authtoken                                                  +
O- ngrok tcp 22                                                     +
-----------------------------------------------------------* 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
```

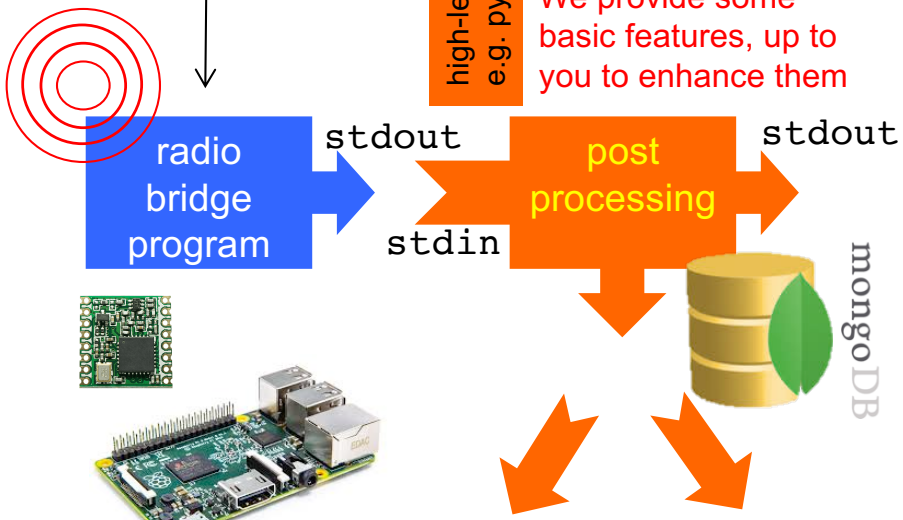
- Here, the previous example file will be installed in `/etc` under the name `test.txt`
- Like previously, the exact command can be sent to the end-user

Our Low-Cost gateway architecture

Kept as simple as possible (for single-channel gateway)

high-level lang. e.g. python

Most of user or application specific logics is done here! We provide some basic features, up to you to enhance them

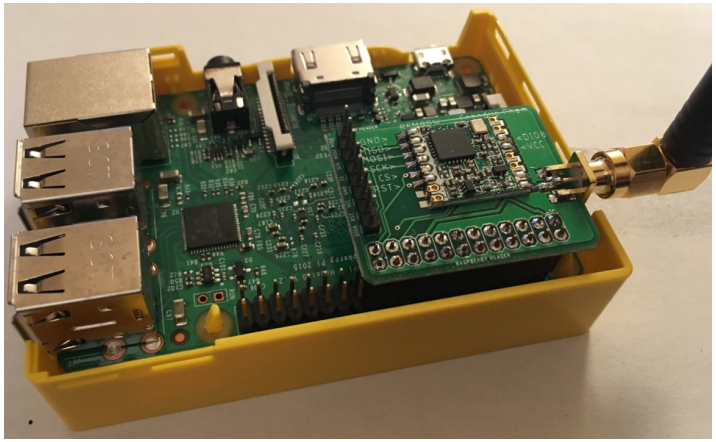


post-processing ⟷ user/app-specific

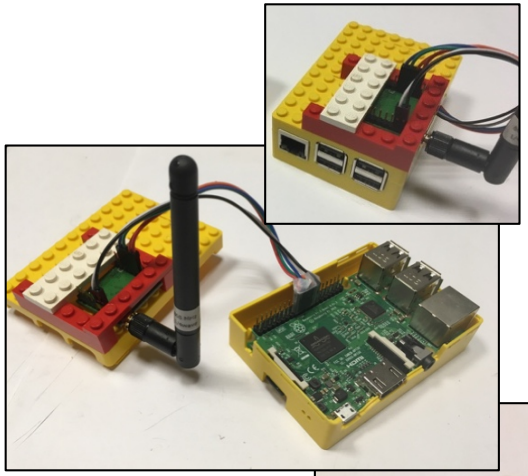
lora_gateway (radio bridge program)

Long-range radio lib

Raspbian + RaspbianOS



Data post-processing stage



RPI 3
WiFi
LoRa

RPI ZeroW
WiFi
LoRa+2G/3G shield

RPI 3
LoRa
WiFi
PoE
2G/3G dongle
DHT22 in-case sensor

radio bridge program

stdout

stdin

post processing

stdout

JSON

post-processing

Handle downlink data

AES encryption decryption

LoRaWAN interoperability

Incoming data parsing block

Periodic task

JSON

Handle data from other radio interfaces

Monitor gateway temperature

user/app-specific

Cloud definition

cloud_script_1

cloud_script_2

cloud_script_n

WAZIUP

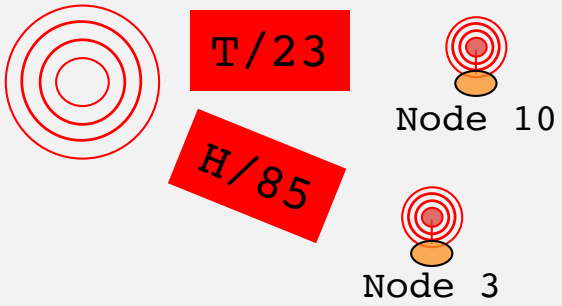
local database

Starting the gateway

- Remember that the gateway software is automatically **launched** when the Raspberry is powered on with our SD card image
- In the next 5 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 39 to see if the gateway is running, then use option K to kill all gateway-related processes to test the next 5 slides.

Starting the basic gateway

- Simply run the low-level `lora_gateway` program
- `sudo ./lora_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
```

ADDING POST-PROCESSING TO 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

UPLOAD 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 full gateway (1)

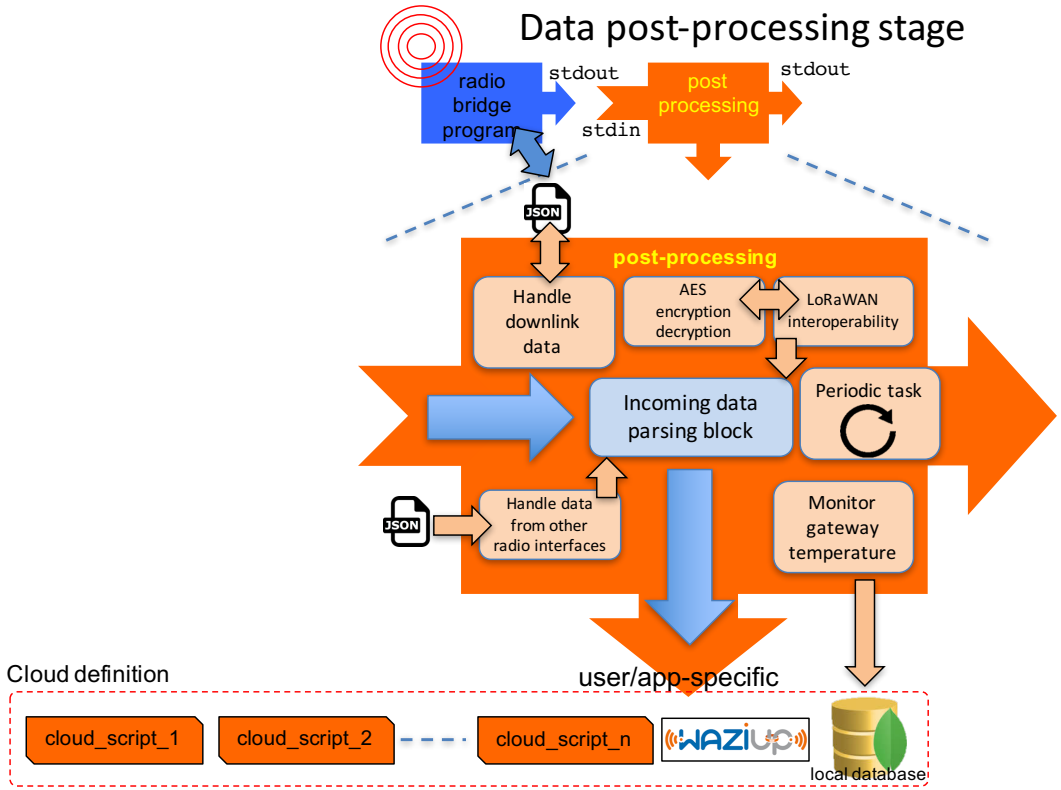
- Simply run `sudo python start_gw.py`
- The full gateway adds logging service of all output to a `post-processing.log` file (with `log_gw.py`)
- LoRa parameters from `gateway_conf.json` are passed to the low-level `lora_gateway` program

```
> 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
```

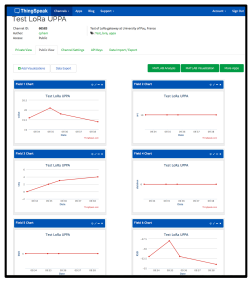
Starting the full gateway (2)

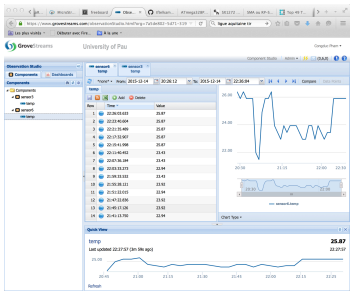
- `python start_gw.py` is actually launch on boot by `/home/pi/lora_gateway/scripts/start_gw.sh`
- `start_gw.sh` starts by launching additional tasks/components depending on the gateway configuration
  - Re-compilation if the RPI model has changed
  - Configuration of Internet access with 3G dongles
  - Configuration of routing rules
  - Starting NodeRed, running MQTT subscription
  - Starting downlink components
  - …
- If you need new functionalities to be added to the gateway on startup, it is the right place to add them

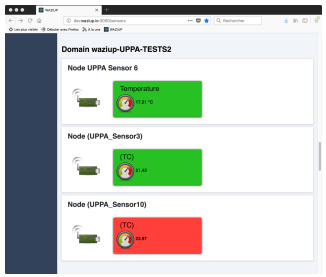
GATEWAY TO CLOUD



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.



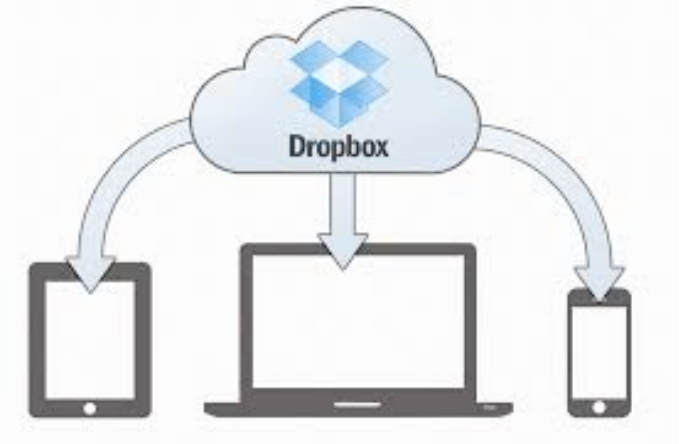




Using Dropbox

- A message starting with '\$' is logged in a file 'telemetry.log' in the `/home/pi/Dropbox/LoRa-test` folder that can 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
...
```



PUSHING TO IoT DATA CLOUDS

- A message starting with '\!' is uploaded on a cloud
- `clouds.json` file defines enabled clouds


```
{
  "clouds": [
    {
      "name": "Local gateway MongoDB",
      "notice": "do not remove the MongoDB cloud declaration, just change en
      "script": "python CloudMongoDB.py",
      "type": "database",
      "max_months_to_store": 2,
      "enabled": false
    },
    {
      "name": "WAZIUP Orion cloud new API",
      "script": "python CloudWAZIUP.py",
      "type": "iotcloud",
      "enabled": true
    },
    {
      "name": "ThingSpeak cloud",
      "script": "python CloudThingSpeak.py",
      "type": "iotcloud",
      "enabled": true
    },
    {
      "name": "NodeRed flow",
      "script": "python CloudNodeRed.py",
      "type": "nodered",
      "enabled": false
    },
    {
      "name": "MQTT cloud",
      "script": "python CloudMQTT.py",
      "type": "MQTT on test.mosquitto.org",
      "enabled": false
    },
    {
      "name": "Firebase cloud",
      "script": "python CloudFireBase.py",
      "type": "jsoncloud",
      "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

Example with WAZIUP cloud

- To use the WAZIUP cloud:

```
{
  "name": "WAZIUPv2",
  "script": "python CloudWAZIUP.py",
  "type": "iotcloud",
  "enabled": false
},
```

- Edit and modify `clouds.json` according to your need
- `CloudWAZIUP.py` script will use information from `key_WAZIUP.py` to configure data management for each organization
- Therefore you need to configure this file for each organization/gateway

KEY_WAZIUP.PY

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

#project name
project_name="waziup"

#your organization: CHANGE HERE
organization_name="ORG"

#service tree: CHANGE HERE at your convenience, can be empty
#should start with -
service_tree='-TESTS'

#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 entity name will then be service_path+"_"+sensor_name+scr_addr, e.g. "UPPA-TESTS_Sensor2"

#use ONLY letters and numbers [A-Za-z0-9] for the username and the password
username="guest"
password="guest"

#here "private" or "public" for the managed sensors
visibility="public"

source_list=[]
```

You MUST change the organization_name.

service_tree is optional

EDITING KEY_WAZIUP.PY

Use `nano` to edit the file:

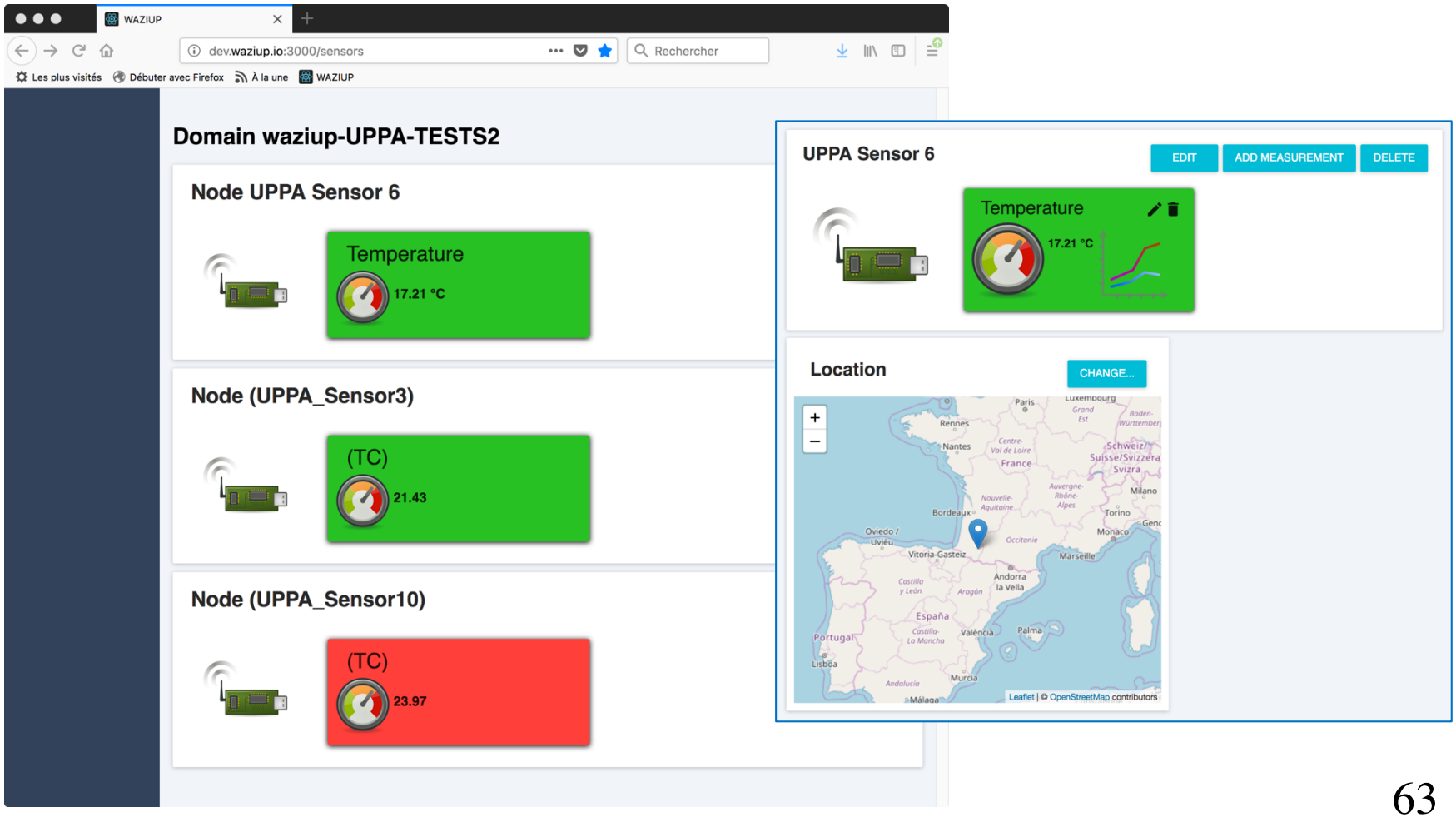
```
> nano key_WAZIUP.py
```

Then CTRL-O + RETURN to save

CTRL-X to quit

The WAZIUP cloud platform

- dashboard.waziup.io

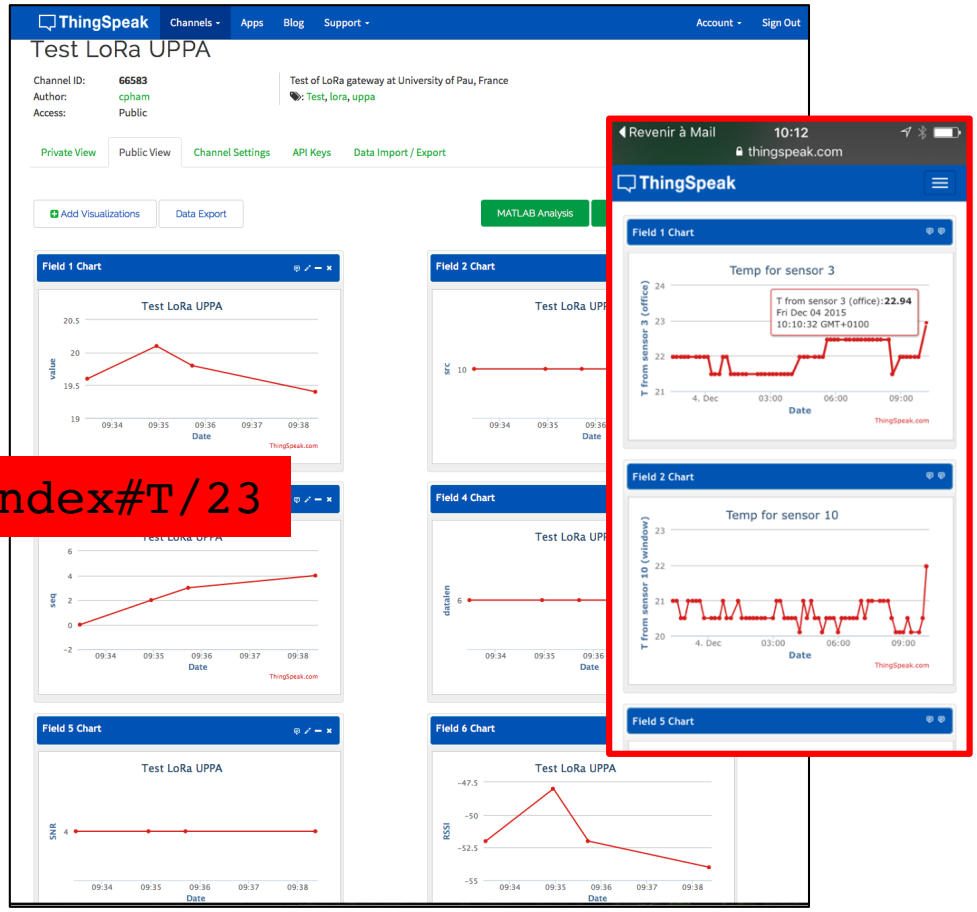


EXAMPLE: ThingSpeak

\!##T/23

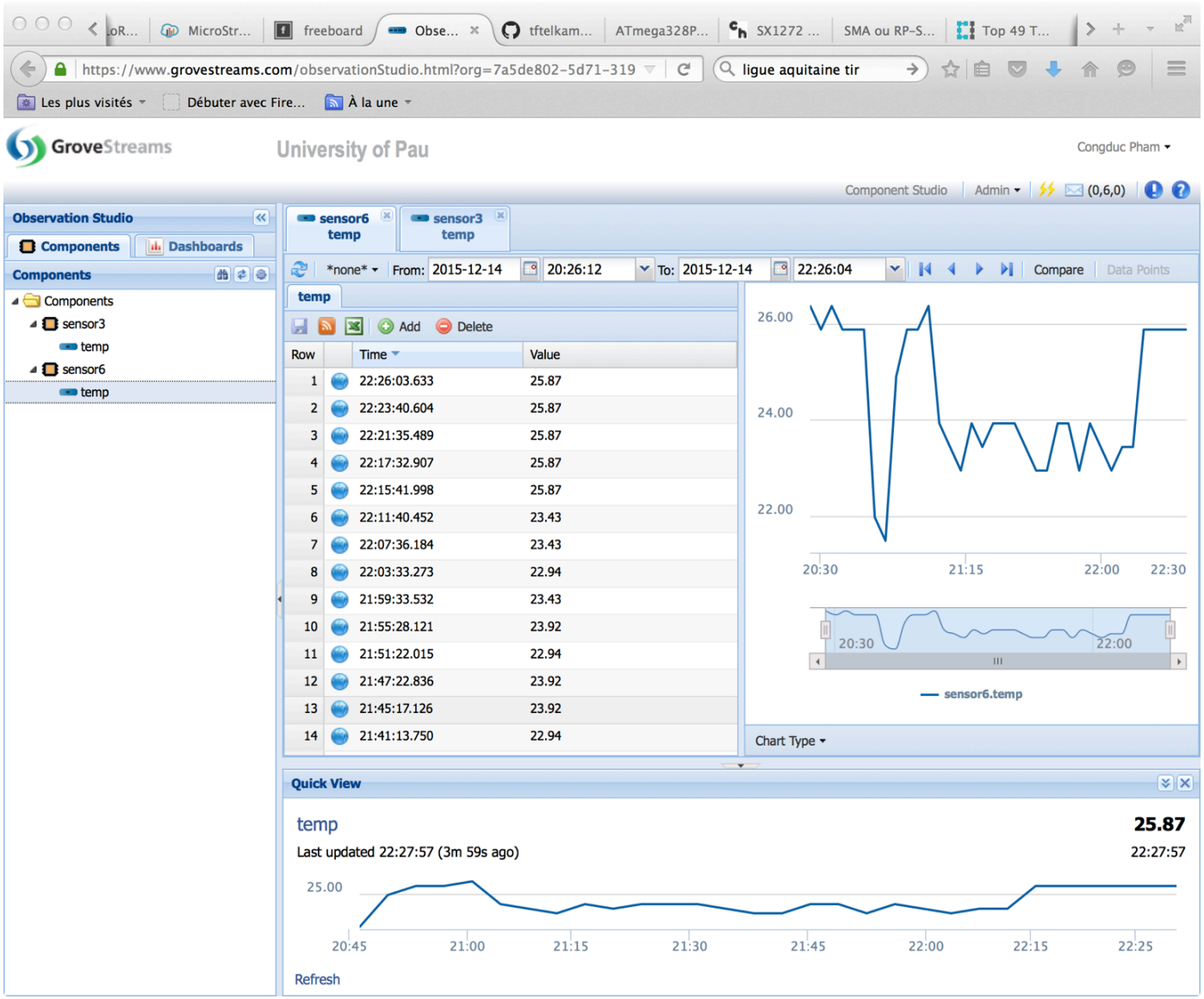
Node 10

\!write_key#field_index#T/23



EXAMPLE: Firebase

EXAMPLE: GroveStreams



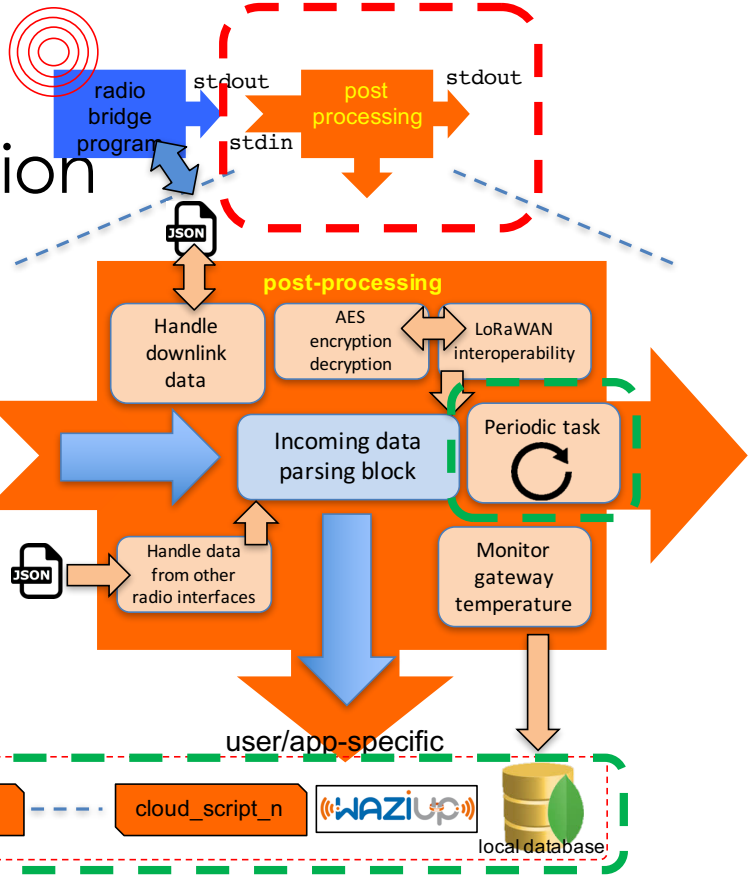
Check log file to see received messages

- Option 5 of command line interface is probably the most useful option to display the `post-processing.log` file
- It uses the Unix `tail -f` command to follow in real time the log file content

```
2018-12-01T14:02:34.745104> --- rxlora. dst=1 type=0x10 src=7 seq=97 len=10 SNR=7 RSSIpkt=-26 BW=125 CR=4/5 SF=12
2018-12-01T14:02:34.745430> 2018-12-01T14:02:34.742386
2018-12-01T14:02:34.745594> rcv ctrl pkt info (^p): 1,16,7,97,10,7,-26
2018-12-01T14:02:34.745742> splitted in:  [1, 16, 7, 97, 10, 7, -26]
2018-12-01T14:02:34.745887> (dst=1 type=0x10(DATA) src=7 seq=97 len=10 SNR=7 RSSI=-26)
2018-12-01T14:02:34.746034> rcv ctrl radio info (^r): 125,5,12
2018-12-01T14:02:34.746185> splitted in:  [125, 5, 12]
2018-12-01T14:02:34.746358> (BW=125 CR=5 SF=12)
2018-12-01T14:02:34.746513> rcv timestamp (^t): 2018-12-01T14:02:34.741
2018-12-01T14:02:34.746663>
2018-12-01T14:02:34.746819> got first framing byte
2018-12-01T14:02:34.746966> --> got LoRa data prefix
2018-12-01T14:02:34.747128> valid app key: accept data
2018-12-01T14:02:34.747313> number of enabled clouds is 1
2018-12-01T14:02:34.747486> --> cloud[0]
2018-12-01T14:02:34.747634> uploading with python CloudThingSpeak.py
2018-12-01T14:02:34.747809> python CloudThingSpeak.py "TC/19.89" "1,16,7,97,19,7,-26" "125,5,12" "2018-12-01T14:02:34+01:00" "0000B827EB3294C8"
2018-12-01T14:02:38.010744> ThingSpeak: uploading (multiple)
2018-12-01T14:02:38.011224> rcv msg to log (\!) on ThingSpeak ( default , default ):
2018-12-01T14:02:38.011552> ThingSpeak: will issue curl cmd
2018-12-01T14:02:38.011872> curl -s -k -X POST --data field1=19.89 https://api.thingspeak.com/update?key=****************
2018-12-01T14:02:38.012200> ThingSpeak: returned code from server is 103
2018-12-01T14:02:38.018844> --> cloud end
```

Customizing/Extending your gateway

- The flexible gateway architecture offers high versatility by customization
- There are 4 alternatives for customization
- The geek way
  - Modify/extend post-processing script
- The "smarter" way
  - Add "cloud" scripts
    - On packet reception
  - Add low frequency periodic tasks
    - Independant from packet reception
  - Add fast frequency statistic-oriented tasks



Add your own cloud script

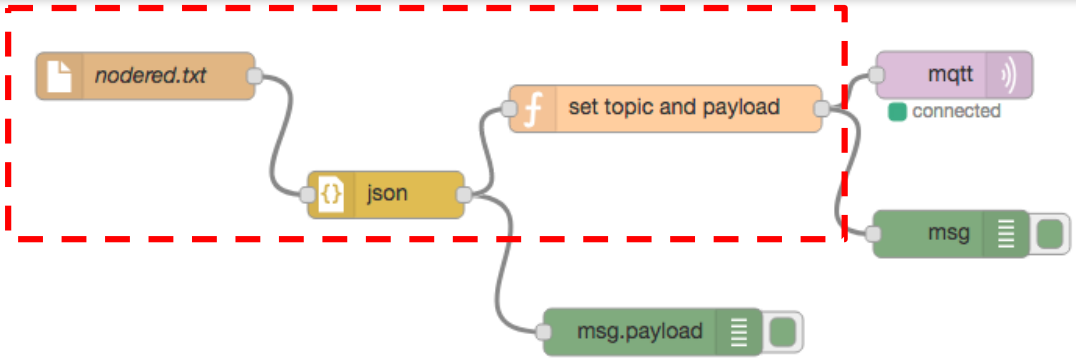
- Use our templates to write your own cloud script
  - `CloudWAZIUP.py, CloudMongoDB.py, CloudThingSpeak.py, CloudGroveStreams.py, CloudNoInternet.py, CloudNodeRed.py, …`
- A cloud script is called with 5 arguments
  - ldata: the received data
    - e.g. #4#TC/21.5 as 1st argument (sys.argv[1] in python)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

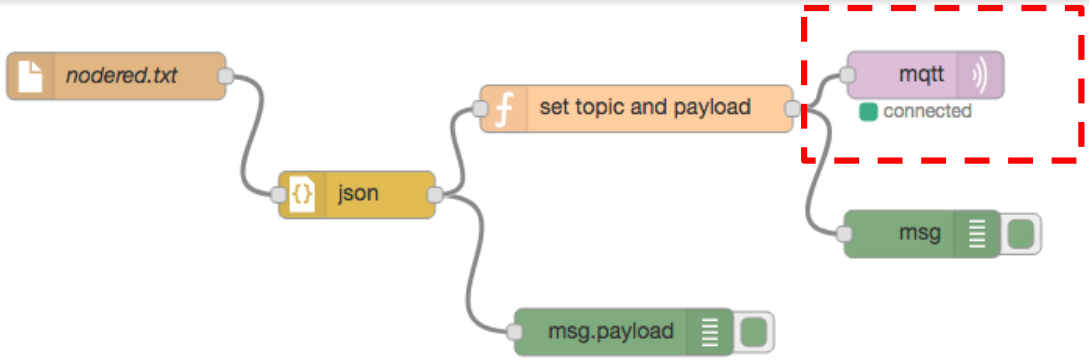
- `CloudNodeRed.py` shows how interface with Node-Red can be simply implemented to benefit from the facility offered by Node-Red
- We use `key_NodeRed.py` to define 3 variables that will be used by `CloudNodeRed.py`
  - `project_name="waziup"`
  - `organization_name="UPPA"`
  - `sensor_name="Sensor"`
- when a device 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}`
- More information on
  - https://github.com/CongducPham/LowCostLoRaGw/blob/master/gw_full_latest/README-NewCloud.md

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;`

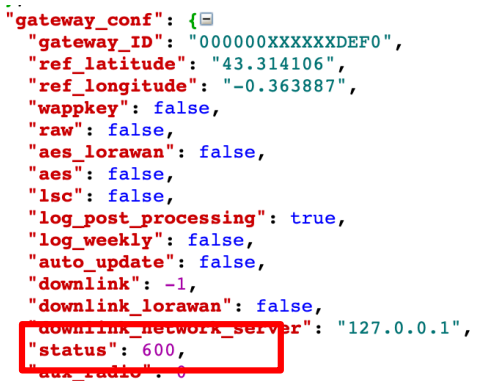
Node-Red flow (2)

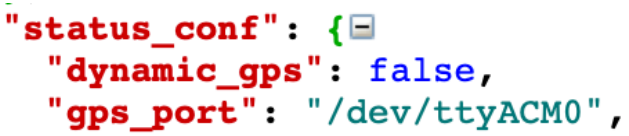


- An MQTT node using the `test.mosquitto.org` broker will receive the messages with the topic defined as `waziup_UPPA_Sensor2/TC` and `waziup_UPPA_Sensor2/HU`
- It will then respectively publish 22.5 and 85 under these topics
- More information on:
  - https://github.com/CongducPham/LowCostLoRaGw/blob/master/gw_full_latest/README-NodeRed.md

Add your own low frequency periodic task

- `post_processing_gw.py` periodically calls `post_status_processing_gw.py` based on the value defined by "status" (in seconds)
- A value of 0 disables periodic status tasks
- You can add your own periodic tasks in `post_status_processing_gw.py`
- For instance, one periodic task is to get the GPS position of the gateway (from a USB GPS module) in a dynamic manner for mobility scenarios
- You can use the `status_conf` section of `gateway_conf.json` to add whatever you need to control your periodic tasks
- `post_status_processing_gw.py` provides examples on how you can add new tasks

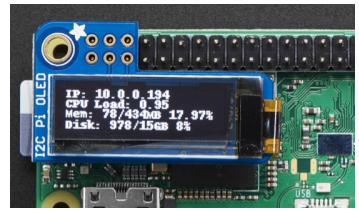




Add your own fast frequency periodic task

- `post_processing_gw.py` periodically calls `sensors_in_raspi/stats.py` based on the value defined by "fast_stats" (in seconds)
- A value of 0 disables fast stats
- You can add your own periodic tasks in `stats.py`
- `stats.py` is mainly based on Adafruit example to display stats on a small OLED screen https://learn.adafruit.com/adafruit-pioled-128x32-mini-oled-for-raspberry-pi/usage
- We added the display of the last received packet's statistics
- You can add other stats but beware that it is a fast frequency stats service so avoid time consuming tasks or printing to Linux stdout

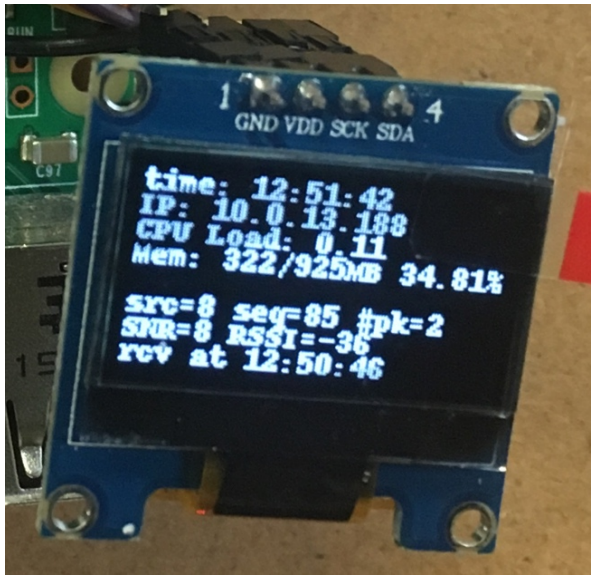

```
"status_conf": {
  "dynamic_gps": false,
  "gps_port": "/dev/ttyACM0",
  "copy_post_processing_log": false,
  "dht22": false,
  "dht22_mongo": false,
  "check_internet_pending": true,
  "fast_stats": 15,
  "cs_status": true
},
```

Connect a small OLED screen to your gateway

- A small I2C OLED screen can be connected to the RPI and it will be driven by the fast frequency stats service
- Just connect 3.3v/5V, GND, SDA, SCL
- You can hot-(un)plug the OLED at any time, very convenient for fast debugging/tests

GPIO#	2nd func	Pin#	Pin#
	+3.3 V	1	2
2	SDA1 (I2C)	3	4
3	SCL1 (I2C)	5	6
4	GCLK	7	8
	GND	9	10
17	GEN0	11	12
27	GEN2	13	14
22	GEN3	15	16
	+3.3 V	17	18
10	MOSI (SPI)	19	20
9	MISO (SPI)	21	22
11	SCLK (SPI)	23	24
	GND	25	26

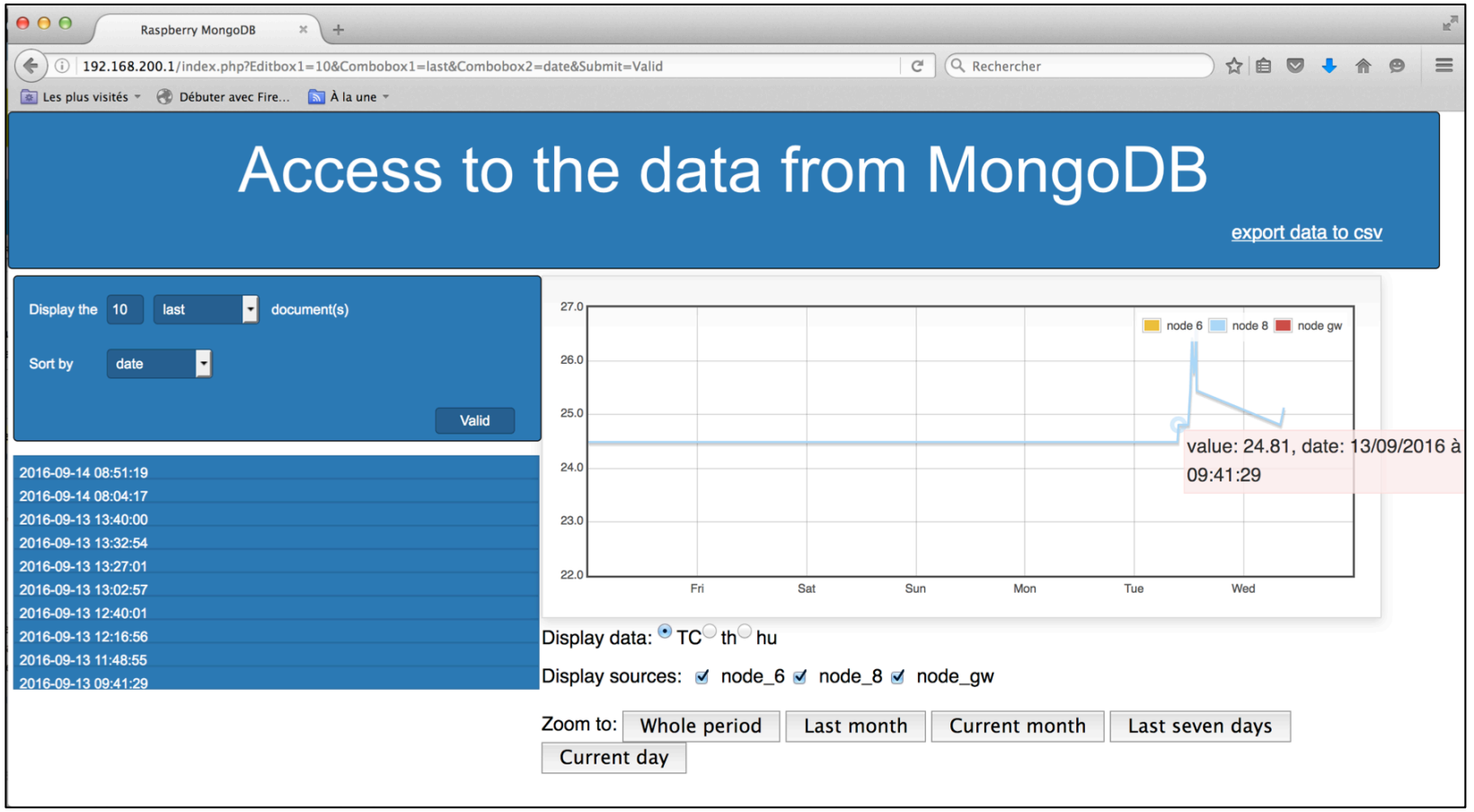


Standalone gateway

Connect to the embedded web data server

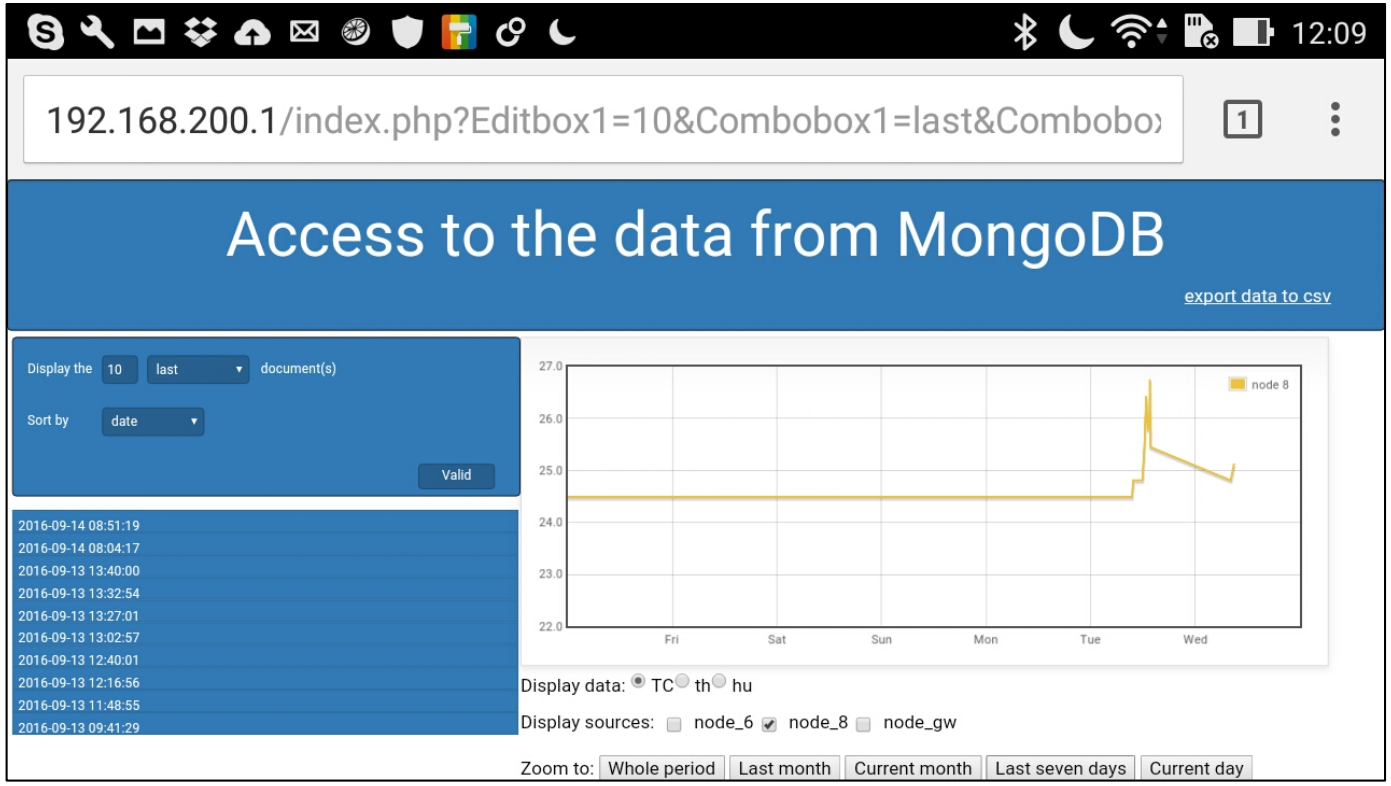
- Received data are also stored on the gateway if `CloudMongoDB.py` is enabled
- On the WiFi interface
  - Gateway address is 192.168.200.1
- On the Ethernet interface
  - Gateway address is the IP address assigned by the DHCP server (of your LAN or laptop)
- Choose any of these solutions and open a web browser to enter the gateway IP address in the URL bar
  - `http://192.168.200.1`

Data from the local web server



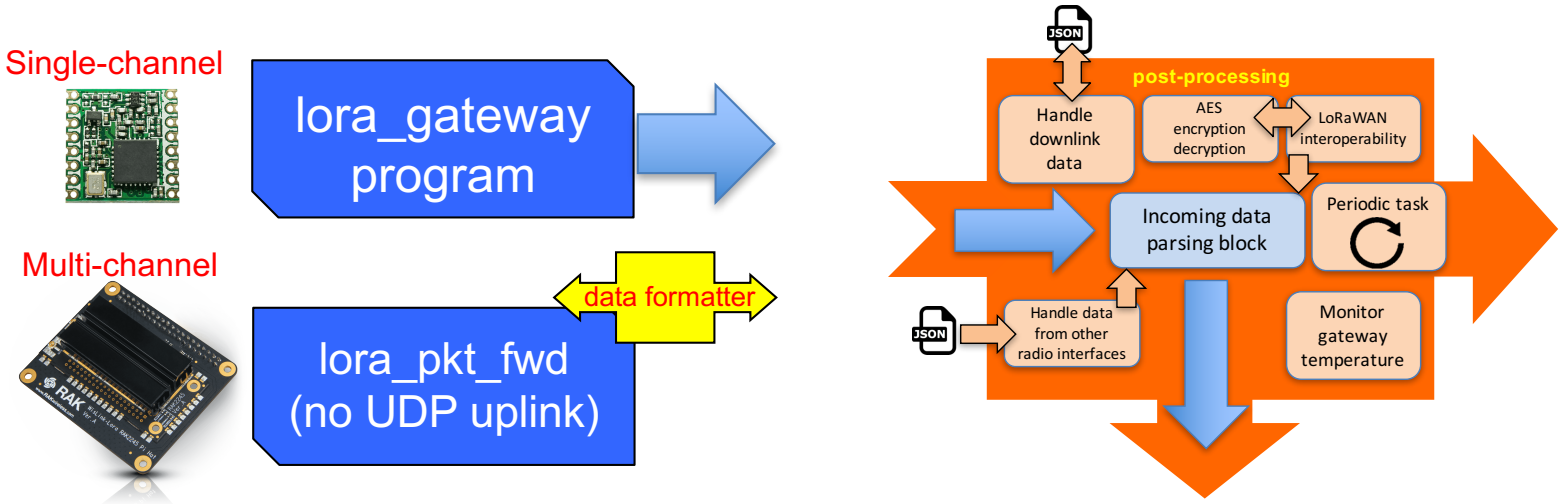
Visualize it on your smartphone!

- Don't forget to join the WAZIUP_PI_GW_xxxxxxxxxx WiFi



Extending with multi-channel support

- Use an SX1301-based concentrator shield for Raspberry
- A data formatter will link the Semtech's packet forwarder (modified, no UDP uplink) to the post-processing stage
- LoRaWAN cloud scripts can push data to LoRaWAN Network Server while other Internet clouds are also supported
- https://github.com/CongducPham/LowCostLoRaGw/blob/master/gw_full_latest/scripts/rak2245-rak831/README.md



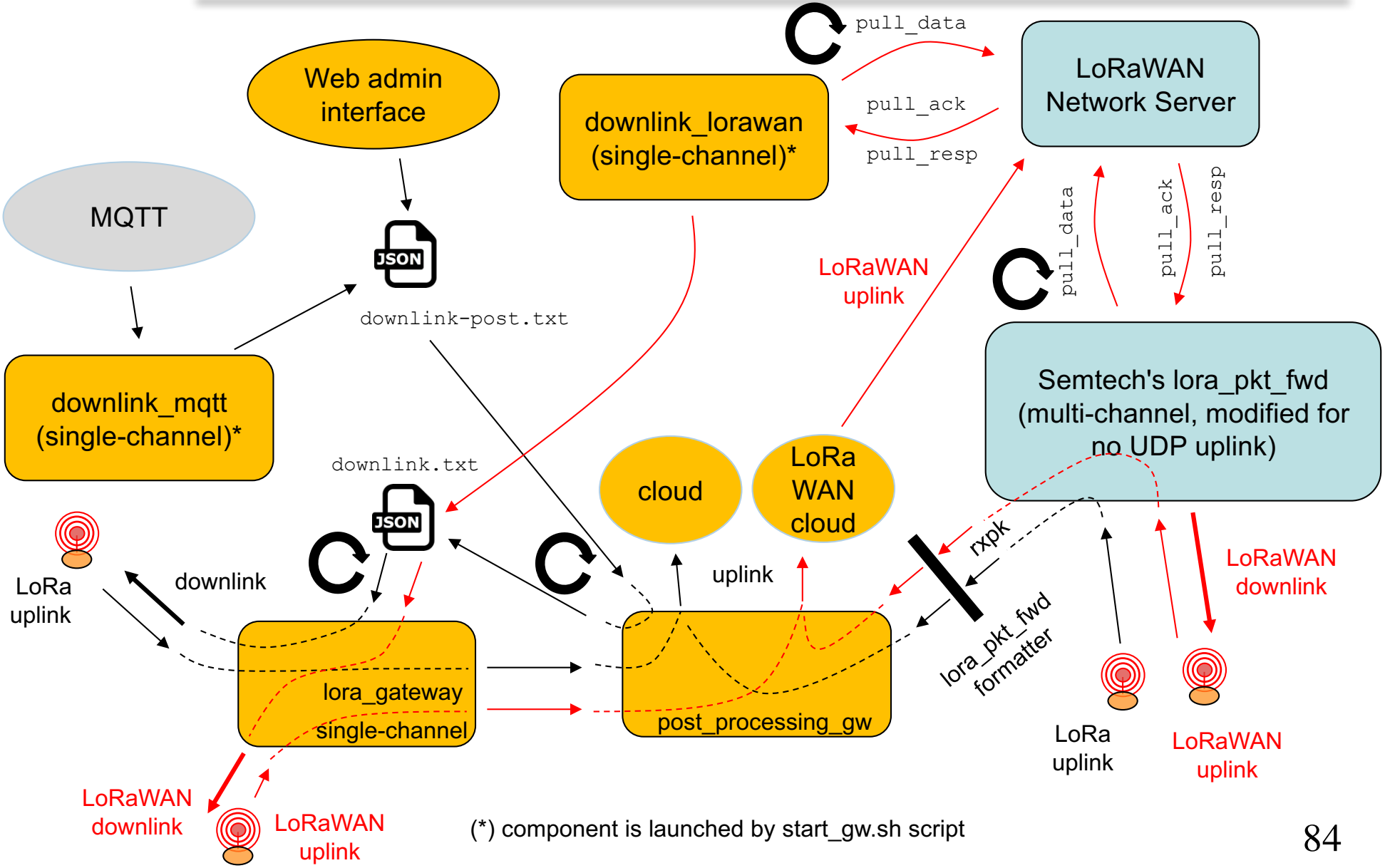
More choices with WAZIUP!

- WAZIUP gateway has an open architecture
  - high-level language scripting, higher-level of flexibility
  - easy customization according to the application and deployment needs,
  - any "cloud" systems can be used to receive uplink data from devices
  - can also handle no-internet scenarios
- The software can be used to build
  - Low-cost single channel gateways for small scale, adhoc LoRa/LoRaWAN deployments
  - Low-cost multi-channel gateways for medium to large scale, adhoc LoRa/LoRaWAN deployments
  - Low-cost multi-channel gateways for large-scale, both adhoc and public LoRaWAN deployments

Downlink support

- Downlink is needed for
  - Actuation at device
  - Over-The-Air-Activation (OTAA) of device
- Multi-channel gateway based on modified Semtech's `lora_pkt_fwd` inherits from LoRaWAN downlink feature
  - `lora_pkt_fwd` periodically polls Network Server and transmit downlink packet
- For single-channel gateway, `lora_gateway` provides a limited LoRaWAN downlink support
  - A separate software component periodically polls Network Server and writes downlink messages to a file
  - `lora_gateway` reads the file and transmit the downlink packet
  - Join-request used in OTAA must use the same frequency and LoRa setting (i.e. LoRa datarate) than data uplink messages

GLOBAL PICTURE WITH DOWNLINK SUPPORT



(*) component is launched by start_gw.sh script