

LOW-COST LORA WATERSENSE GATEWAY: A STEP-BY-STEP TUTORIAL



PROF. CONGDUC PHAM
[HTTP://WWW.UNIV-PAU.FR/~CPHAM](http://www.univ-pau.fr/~cpham)
UNIVERSITÉ DE PAU, FRANCE



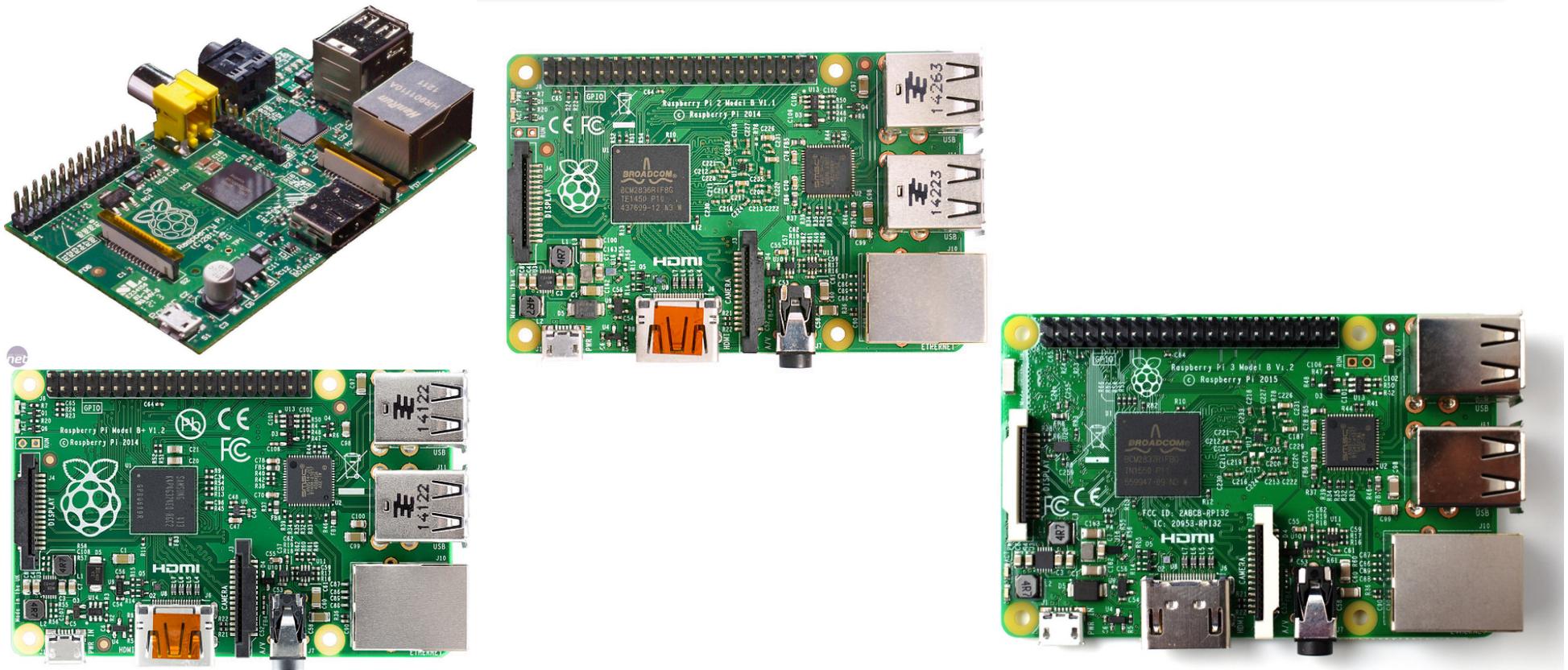
CONTENTS

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

ASSEMBLING THE HARDWARE

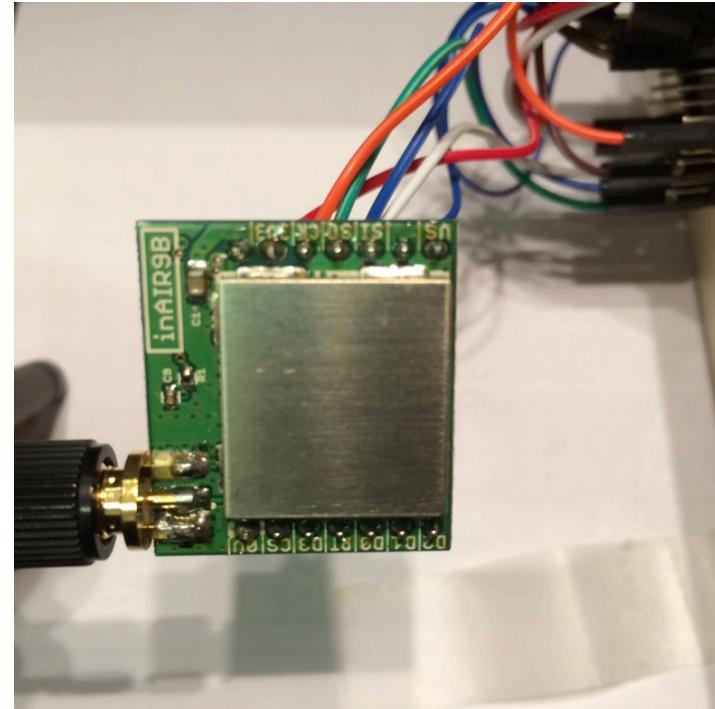
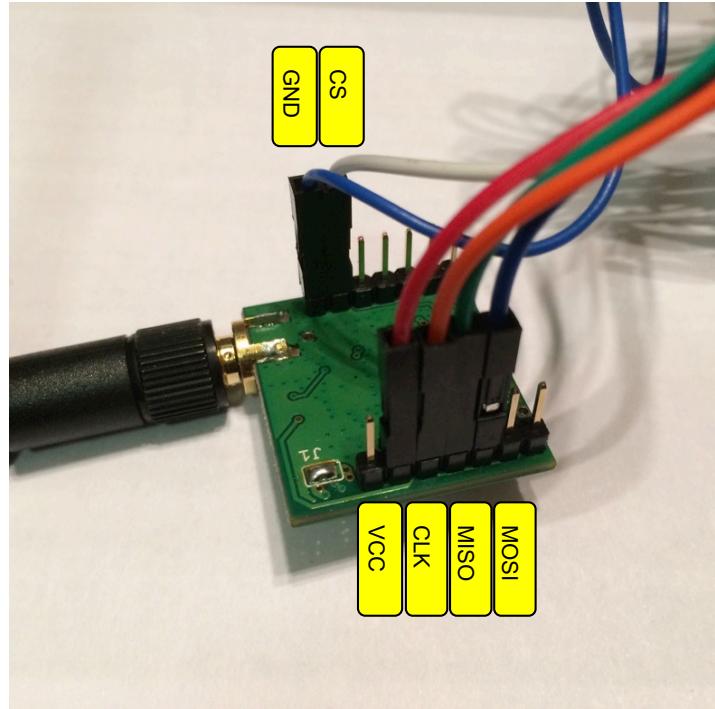


GET THE RASPBERRY



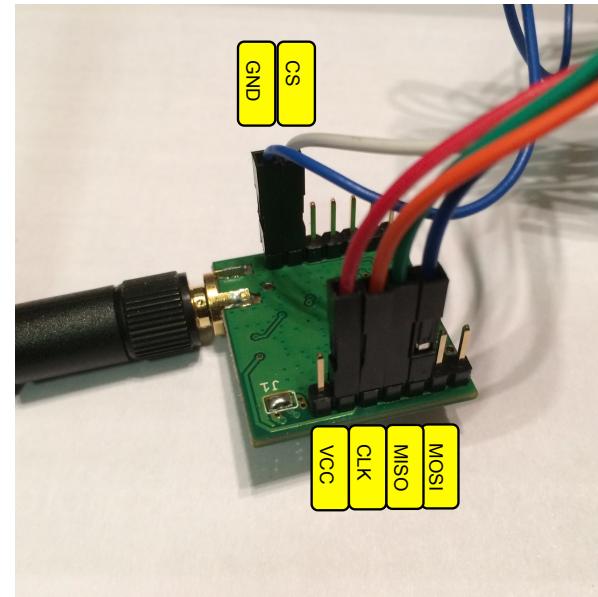
You can use Raspberry 1 model B or B+ or Raspberry 2 model B or Raspberry 3 model B.
The most important useful feature is the Ethernet interface for easy Internet connection.
You can use WiFi to get Internet connection by adding a WiFi USB dongle. With the
Raspberry 3, WiFi and Bluetooth are embedded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

NOW THE RADIO MODULE



If you go for the inAir (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.

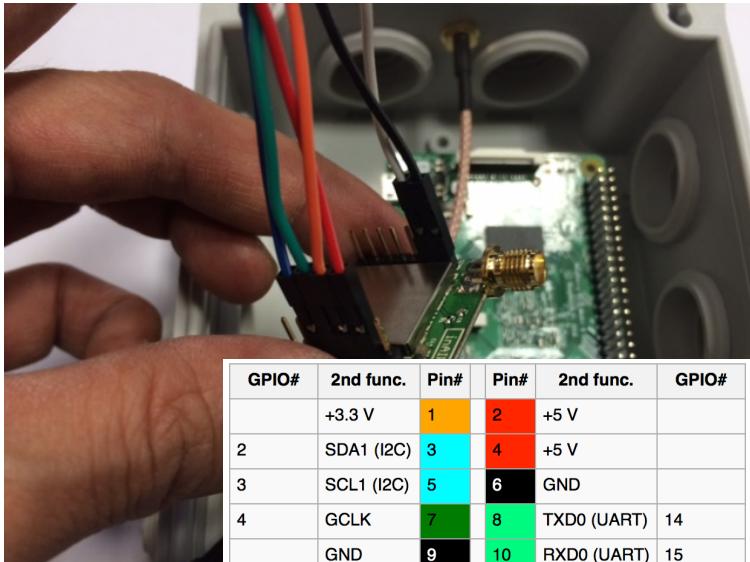
CONNECTING THE RADIO MODULE (1)



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.

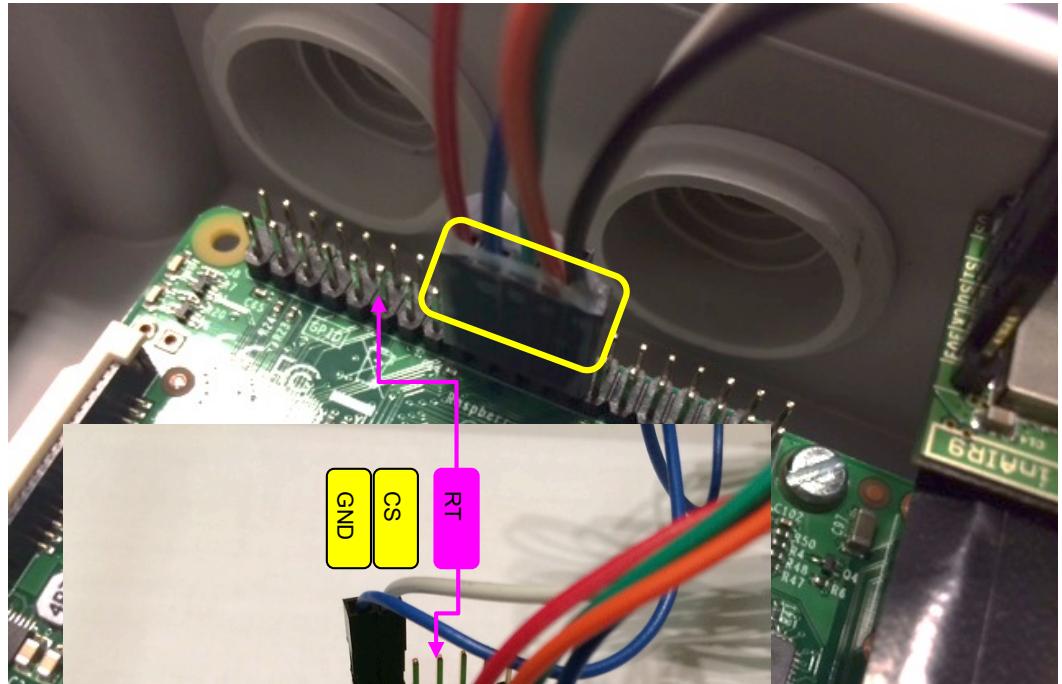
CONNECTING THE RADIO MODULE (2)



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	TXDO (UART)	14
	GND	9	10	RXD0 (UART)	15
17	GEN0	11	12	GEN1	18
27	GEN2	13	14	GND	
22	GEN3	15	16	GEN4	23
	+3.3 V	17	18	GEN5	24
10	MOSI (SPI)	19	20	GND	
9	MISO (SPI)	21	22	GEN6	25
11	SCLK (SPI)	23	24	CE0_N (SPI)	8
	GND	25	26	CE1_N (SPI)	7

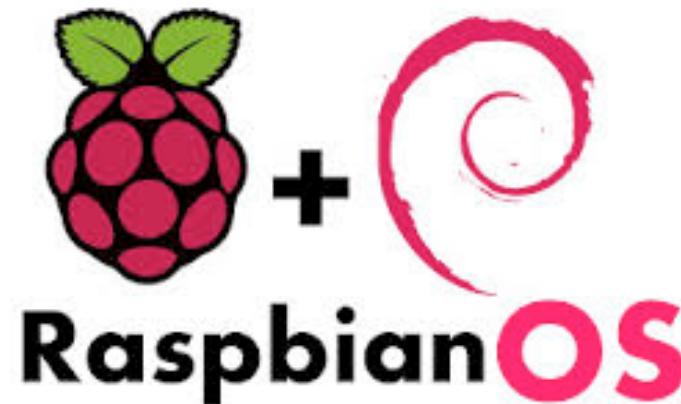
(RPI 1 Models A and B stop here)

EEPROM	ID_SD	27	28	ID_SC	EEPROM
5	N/A	29	30	GND	
6	N/A	31	32		12
13	N/A	33	34	GND	
19	N/A	35	36	N/A	16
26	N/A	37	38	Digital IN	20
	GND	39	40	Digital OUT	21



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

GETTING, COMPILING & INSTALLING THE SOFTWARE



FLASHING THE OS

- An SD card image with a Raspberry Raspbian Jessie version is provided.
- You will need an 8GB SD card. Be careful, some SD cards will not work. This one has been successfully tested. It has to be class 10.
- Look at
<https://www.raspberrypi.org/documentation/installation/installing-images/> to see the procedure depending on your OS. 7948206080 bytes should be written, otherwise you may have a problem.
- Once flashed, insert the SD card and power-up the Raspberry-based gateway.

SSH TO THE GATEWAY

- ❑ The Raspbian image sets the Raspberry for DHCP on wired Ethernet and as a WiFi access point.
- ❑ If you connected the gateway to your LAN or laptop using wired Ethernet then the gateway will be assigned an IP address. Use this address to connect with SSH to the gateway
- ❑ You can use Angry IP Scanner (<http://angryip.org/>) to know the assigned address
- ❑ Use `ssh pi@rpi_addr`, where `rpi_addr` is the IP address assigned to the gateway
- ❑ Login password is `loragateway` if you installed from the SD card image
- ❑ However, using the built-in WiFi access point is easier as shown in the next slide

SSH TO THE GATEWAY WITH WiFi

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

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



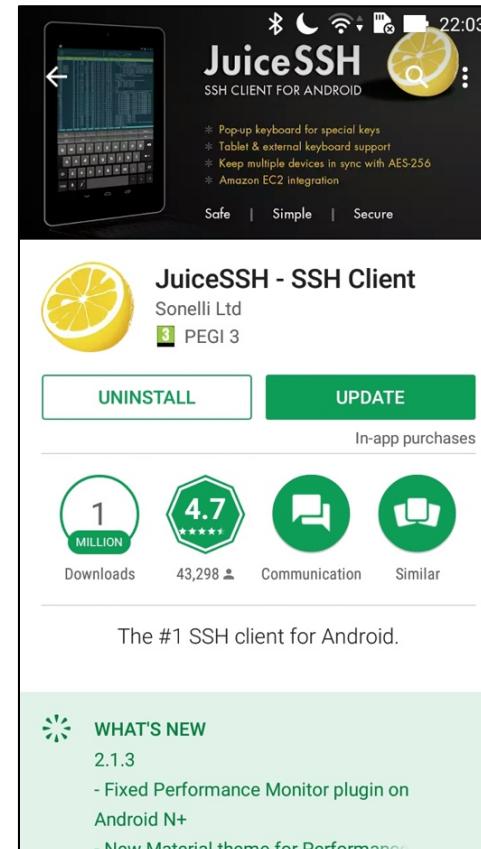
```
MacBookProRetina-de-Congduc-Pham:~ cpham$ ssh pi@192.168.200.1
pi@192.168.200.1's password:

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

By Congduc Pham for WaterSense project

```
=====* Gateway 0000027EB5A71F7 ====
0- sudo python start_gw.py +
1- sudo ./lora_gateway --mode 1 +
2- sudo ./lora_gateway --mode 1 | python post_processing_gw.py +
3- ps aux | grep -e start_gw -e lora_gateway -e post_proc -e log_gw +
4- tail --line=25 ../Dropbox/LoRa-test/post-processing.log +
5- tail --line=25 -f ../Dropbox/LoRa-test/post-processing.log +
6- less ../Dropbox/LoRa-test/post-processing.log +
-----* Bluetooth *---+
a- run: sudo hciconfig hci0 pscan +
b- run: sudo python rfcomm-server.py +
c- run: nohup sudo python rfcomm-server.py -bg > rfcomm.log &
d- run: ps aux | grep rfcomm +
e- run: tail -f rfcomm.log +
-----* Connectivity *---+
f- test: ping www.univ-pau.fr +
-----* Filtering msg *---+
l- List LoRa reception indications +
m- List radio module reset indications +
n- List boot indications +
o- List post-processing status +
p- List low-level gateway status +
-----* Configuration *---+
A- show gateway_conf.json +
B- edit gateway_conf.json +
C- show clouds.json +
D- edit clouds.json +
-----* Update *---+
U- update to latest version on repository +
V- download and install a file +
W- run a command +
-----* kill *---+
K- kill all gateway related processes +
k- kill rfcomm-server process +
R- reboot gateway +
S- shutdown gateway +
-----+
Q- quit +
=====+
Enter your choice:

```

```
pi@raspberrypi:~/lora_gateway $ 
```

DEFAULT GATEWAY CONFIGURATION

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

GATEWAY_CONF.JSON

- The gateway_conf.json file contains the gateway configuration

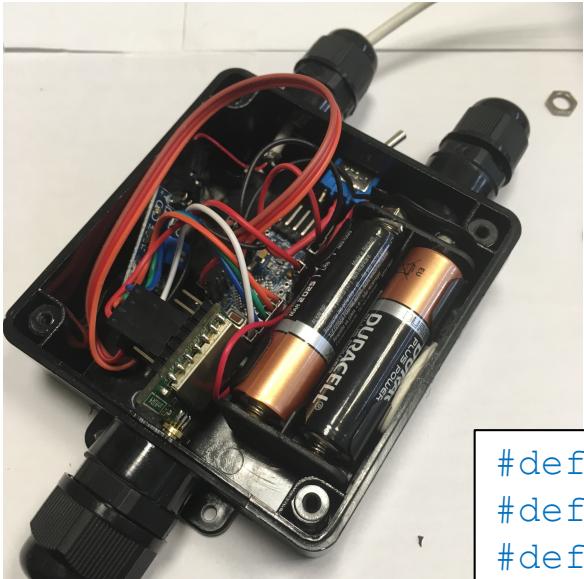
```
{  
    "radio_conf" : {  
        "mode" : 1,  
        "bw" : 500,  
        "cr" : 5,  
        "sf" : 12,  
        "ch" : -1,  
        "freq" : -1  
    },  
    "gateway_conf" : {  
        "gateway_ID" : "000000XXXXXXXXXX",  
        "ref_latitude" : "my_lat",  
        "ref_longitude" :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

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

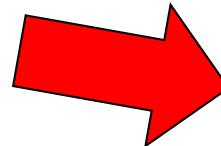
Set "ch" to a channel number if you don't want to use the default channel (which is channel 10 in the 868 band channel 05 in 900 band and channel 00 in the 433 band).

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

DEFAULT CONFIGURATION



\!##SM1/530/SM2/756



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



The default configuration in the Arduino_LoRa_SoilHum example is:

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

The default gateway configuration is also LoRa mode 1.

Both are working in 433.3MHz band for Pakistan.

CHECK THAT THE GATEWAY IS RUNNING

- Use option 3 of the text interface

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

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

FULL UPDATE OF GW SOFTWARE FROM GITHUB

CongducPham / WaterSense

Code Issues 0 Pull requests 0 Projects 0 Wiki Pulse Graphs Settings

WaterSense project Add topics

11 commits 1 branch 0 releases 0 contributors

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

Congduc Pham update README Latest commit 7f05068 3 days ago

WaterSenseGateway update README 3 days ago
sketch remove .DS_Store files 7 days ago

Help people interested in this repository understand your project by adding a README. Add a README

CongducPham / WaterSense

Code Issues 0 Pull requests 0 Projects 0 Wiki Pulse Graphs Settings

Branch: master WaterSense / WaterSenseGateway / Create new file Upload files Find file History

Congduc Pham update README Latest commit 7f05068 3 days ago

3GDongle update cloud scripts with source_list features, update README 3 days ago
aes-python-lib/LoRaWAN remove_.DS_Store files 7 days ago
downlink remove_.DS_Store files 7 days ago
php First commit 7 days ago
rapidjson First commit 7 days ago
scripts update SMS scripts 7 days ago
sensors_in_rasp Pi First commit 7 days ago
CloudFireBase.py update cloud scripts with source_list features, update README 3 days ago
CloudFireBaseAES.py update cloud scripts with source_list features, update README 3 days ago
CloudFireBaseLWAES.py update cloud scripts with source_list features, update README 3 days ago
CloudGroveStreams.py update cloud scripts with source_list features, update README 3 days ago
CloudMongoDB.py First commit 7 days ago
CloudSMS.py update cloud scripts with source_list features, update README 3 days ago
CloudThingSpeak.py update cloud scripts with source_list features, update README 3 days ago
CloudWAZIUP_WS.py update cloud scripts with source_list features, update README 3 days ago
MongoDB.py First commit 7 days ago
README-NewCloud.md update cloud scripts with source_list features, update README 3 days ago
README-aes_lowrwan.md First commit 7 days ago
README-downlink.md First commit 7 days ago

```
> mkdir lora_gateway  
> git clone https://github.com/CongducPham/WaterSense.git  
> cp WaterSense/WaterSenseGateway/* lora_gateway/
```

Log in the RPI (ssh) and create a directory called `lora_gateway`. Get the LoRa RPI library from our github: <https://github.com/CongducPham/WaterSense> (right) then copy all the files & folder of the github's **WaterSenseGateway** into the `lora_gateway` folder.

COMPILING THE GW SOFTWARE

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

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

CFLAGS=-DPABOOST

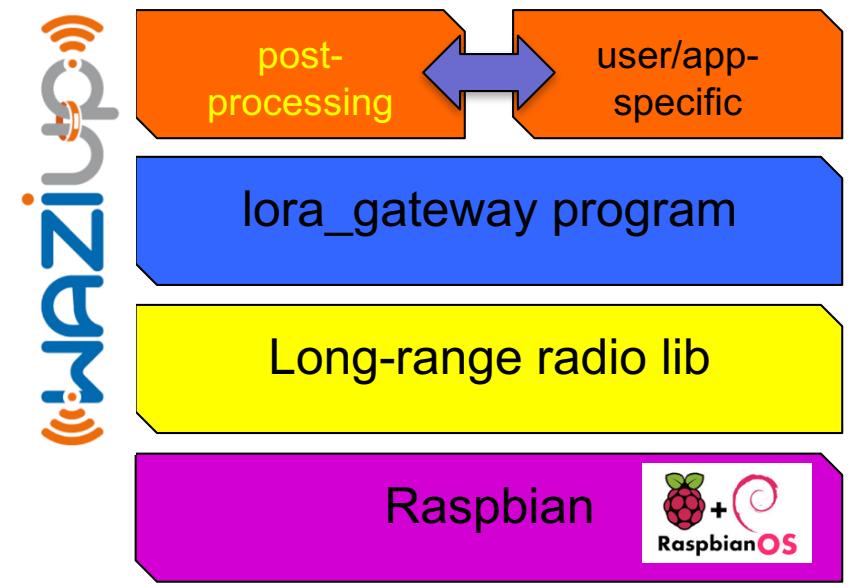
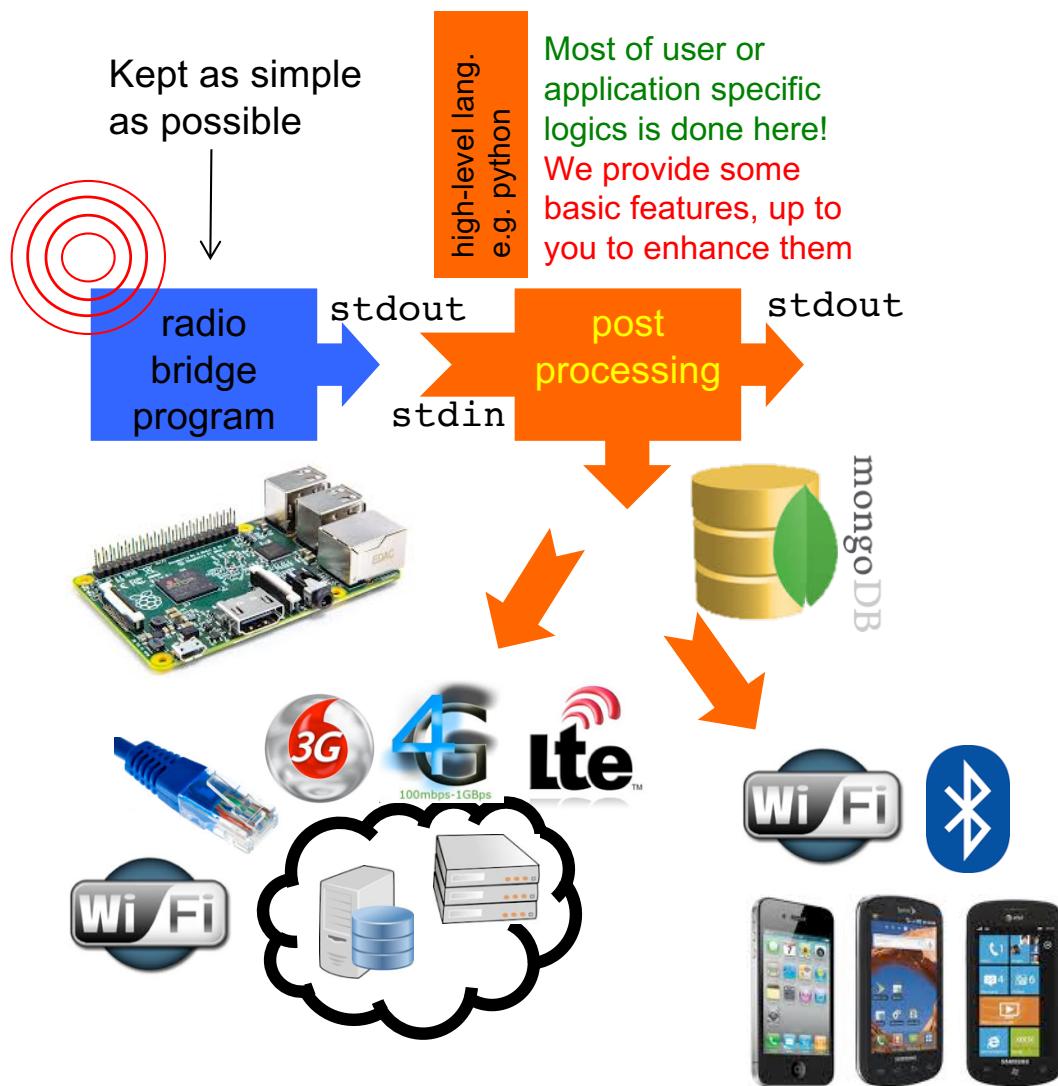
If inAir9/inAir4, Libelium SX1272, leave commented:

#CFLAGS=-DPABOOST

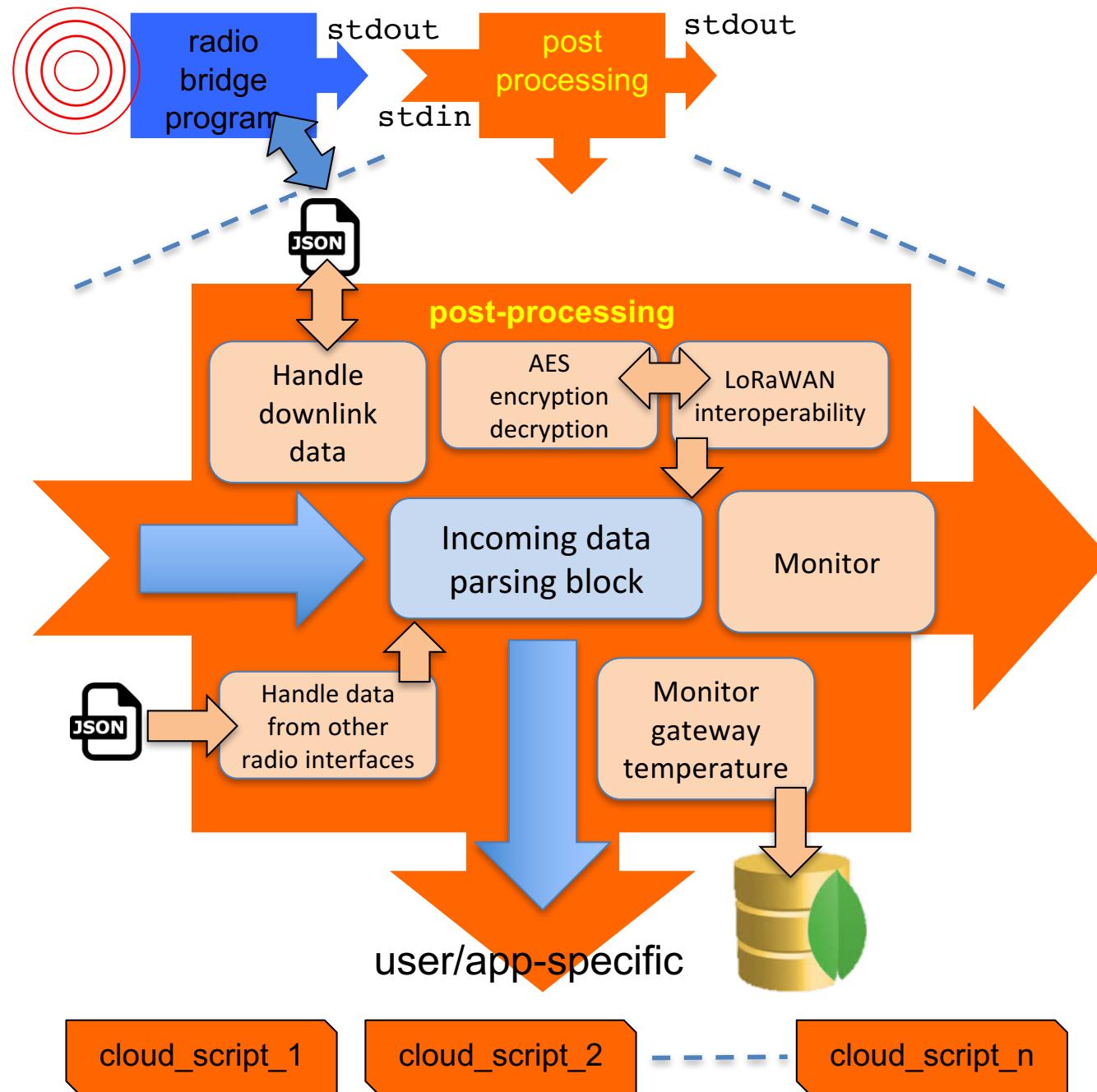
If you have a RPI 2 or RPI3, then type:

```
> make lora_gateway_pi2
```

OUR LOW-COST GATEWAY ARCHITECTURE



POST-PROCESSING BLOCK

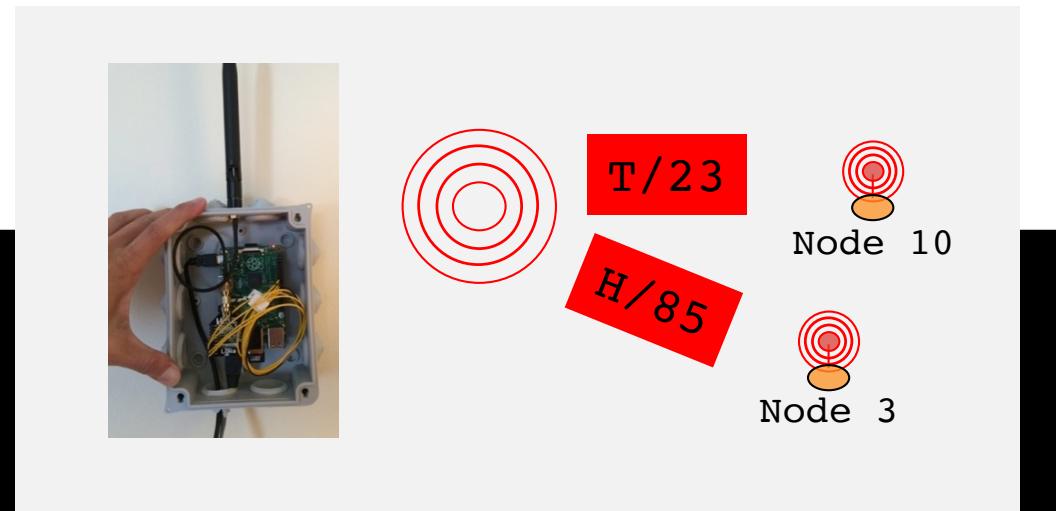


STARTING THE GATEWAY

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

STARTING THE BASIC GATEWAY

```
> sudo ./lora_gateway
Power ON: state 0
Default sync word: 0x12
LoRa mode: 1
Setting mode: state 0
Channel CH_10_868: state 0
Set LoRa Power to M: state 0
Get Preamble Length: state 0
Preamble Length: 8
LoRa addr 1 : state 0
SX1272/76 configured as LR-BS. Waiting RF input for transparent RF-serial bridge
--- rxlora. dst=1 type=0x10 src=10 seq=0 len=4 SNR=9 RSSIpkt=-54
^p1,16,10,0,4,9,-54
^r125,5,12
^t2016-02-25T01:51:11.058
T/23
--- rxlora. dst=1 type=0x10 src=3 seq=0 len=4 SNR=8 RSSIpkt=-54
^p1,16,3,0,4,8,-54
^r125,5,12
^t2016-02-25T01:53:13.067
H/85
```



POST-PROCESSING RECEIVED DATA

```
> 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



\!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
--> qot data prefix
number of enabled clouds is 1
--> cloud[0]
uploading with python CloudThingSpeak.py
ThingSpeak: uploading
rcv msg to log (\!) on ThingSpeak ( default , 4 ): 23
ThingSpeak: will issue curl cmd
curl -s -k -X POST --data field4=23 https://api.thingspeak.com/...
ThingSpeak: returned code from server is 156
--> cloud end
```

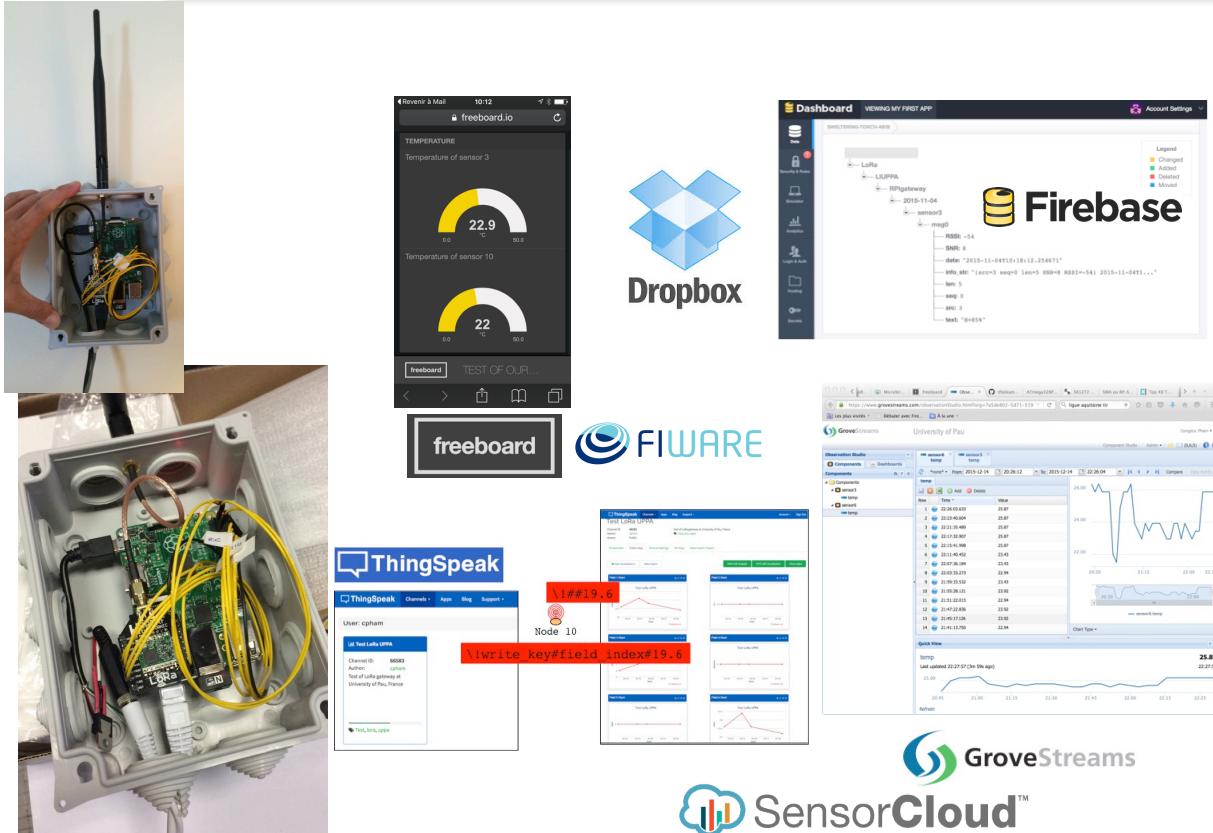
\\$ or \! before the data indicates that the data should be logged on a file or a cloud. It is up to the end-device to decide which option

STARTING THE COMPLETE GATEWAY

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

```
> cd lora_gateway
> sudo python start_gw.py
sudo ./lora_gateway --mode 1 | python post_processing_gw.py | python log_gw.py
Starting thread to report gw status
2017-09-01 12:08:11.751649
post status: gw ON, lat my_lat long my_long
Current working directory: /home/pi/lora_gateway
SX1276 detected, starting SX1276 LF/HF calibration...
*****Power ON: state 0
Default sync word: 0x12
LoRa mode 1
Setting mode: state 0
Channel CH_10_868: state 0
Set LoRa power dBm to 14
Power: state 0
Get Preamble Length: state 0
Preamble Length: 8
LoRa addr 1: state 0
SX1272/76 configured as LR-BS. Waiting RF input for transparent RF-serial bridge
```

GATEWAY TO CLOUD



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

USING

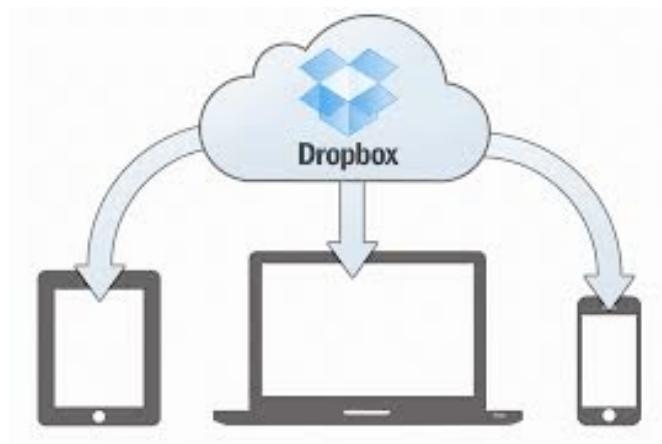


- A message starting with '\\$' is logged in a file 'telemetry.log' in a folder shared through Dropbox

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

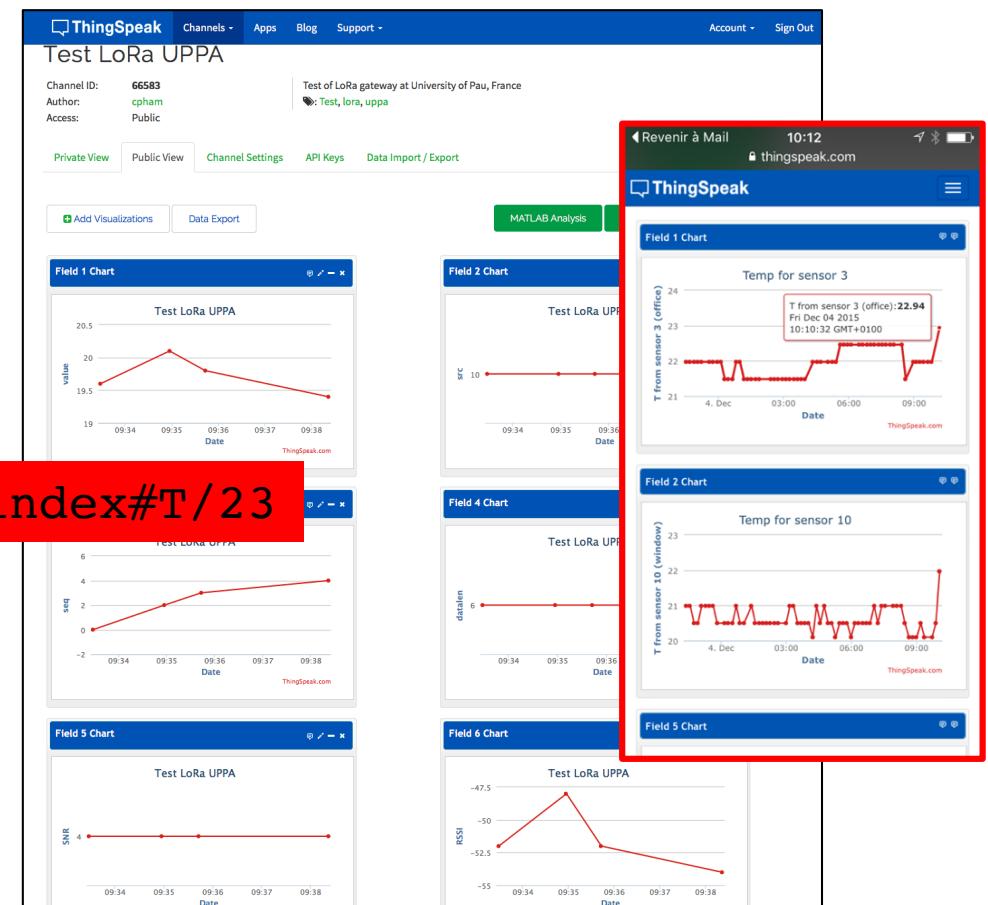
\\$T/23

Node 10

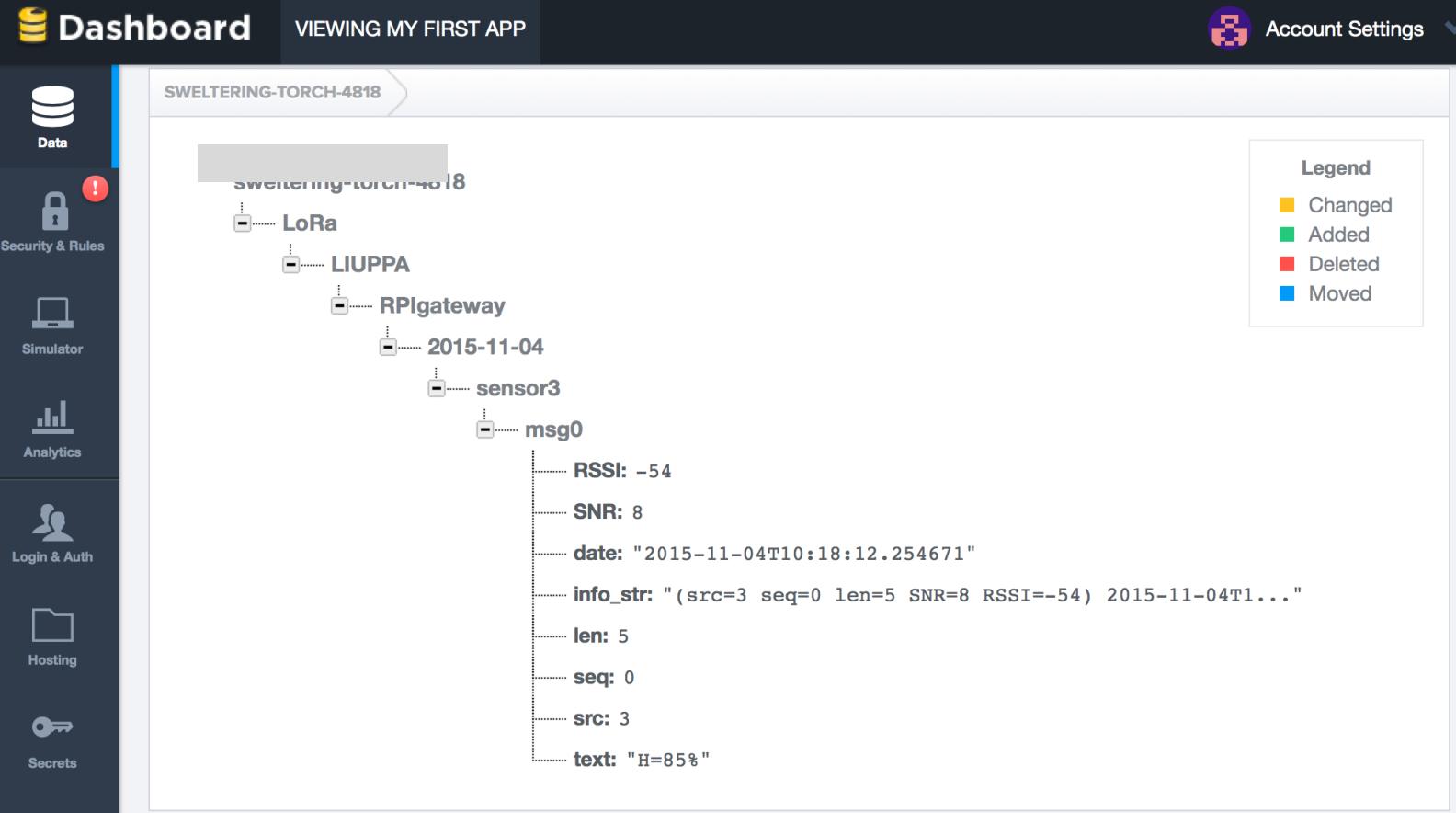


USING ThingSpeak

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



USING Firebase

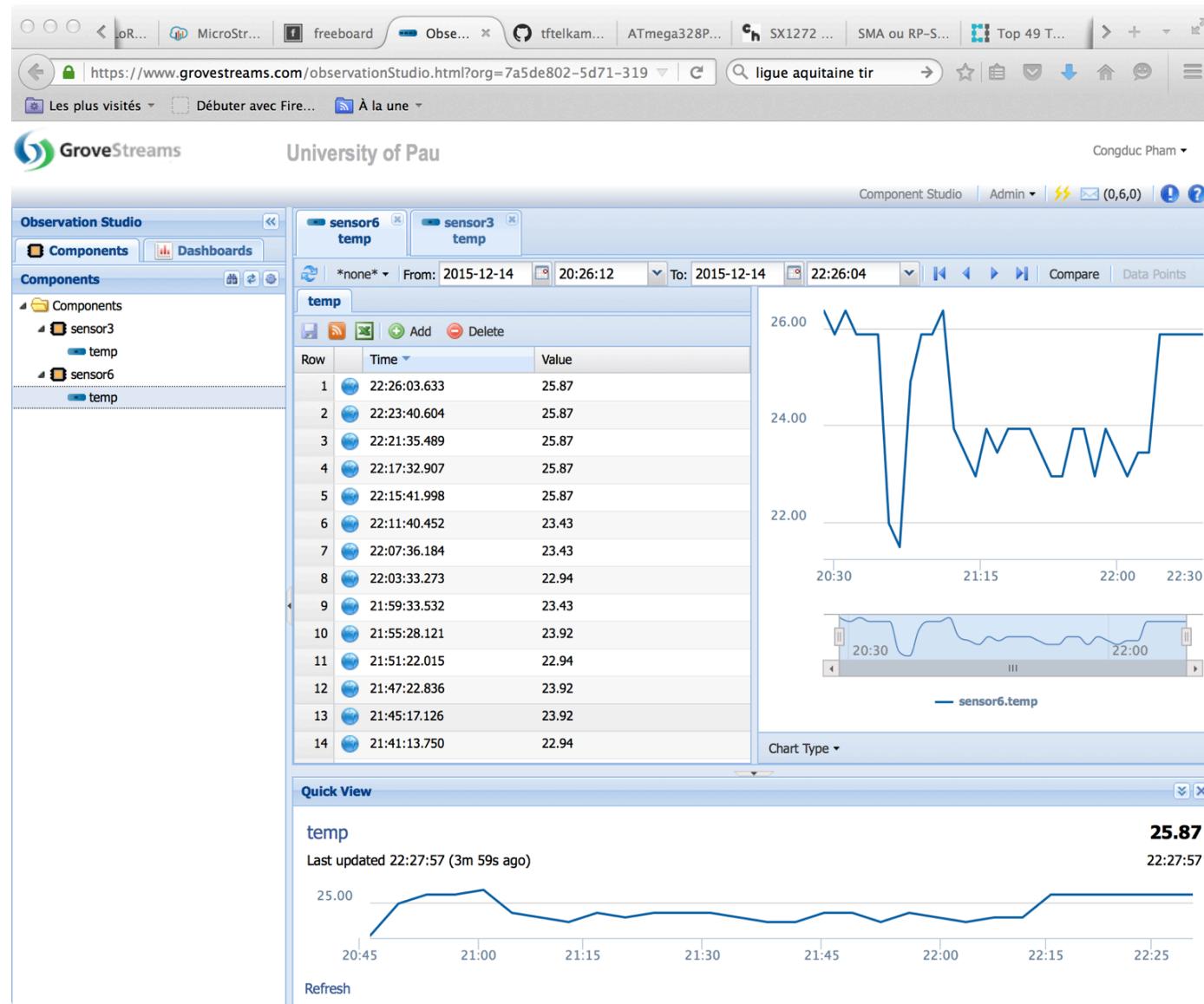


The screenshot shows the Firebase Realtime Database dashboard for a project named "SWELTERING-TORCH-4818". The left sidebar contains navigation links for Dashboard, Data (selected), Security & Rules, Simulator, Analytics, Login & Auth, Hosting, and Secrets. The main panel displays a tree structure of data under the "LoRa" node, specifically for the "LIUPPA" device. The data is organized by date ("2015-11-04") and then by sensor ("sensor3"). A message named "msg0" is expanded, showing its contents:

- RSSI: -54
- SNR: 8
- date: "2015-11-04T10:18:12.254671"
- info_str: "(src=3 seq=0 len=5 SNR=8 RSSI=-54) 2015-11-04T1..."
- len: 5
- seq: 0
- src: 3
- text: "H=85%"

A legend in the top right corner identifies the color coding for changes: yellow for Changed, green for Added, red for Deleted, and blue for Moved.

USING GroveStreams



By Congduc Pham for watersense project

GATEWAY CONFIGURATION

gateway.conf.json

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

radio_conf section defines the LoRa parameters

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

alert_conf section defines alerting methods

CLOUDS.JSON

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

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

Enabled clouds will be called by the post-processing stage

WRITE YOUR OWN CLOUD SCRIPT

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

These parameters are passed to the script. It is up to the cloud script to use these parameters or not.
By Congduc Pham for waterSense project

REBOOTING THE GATEWAY

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

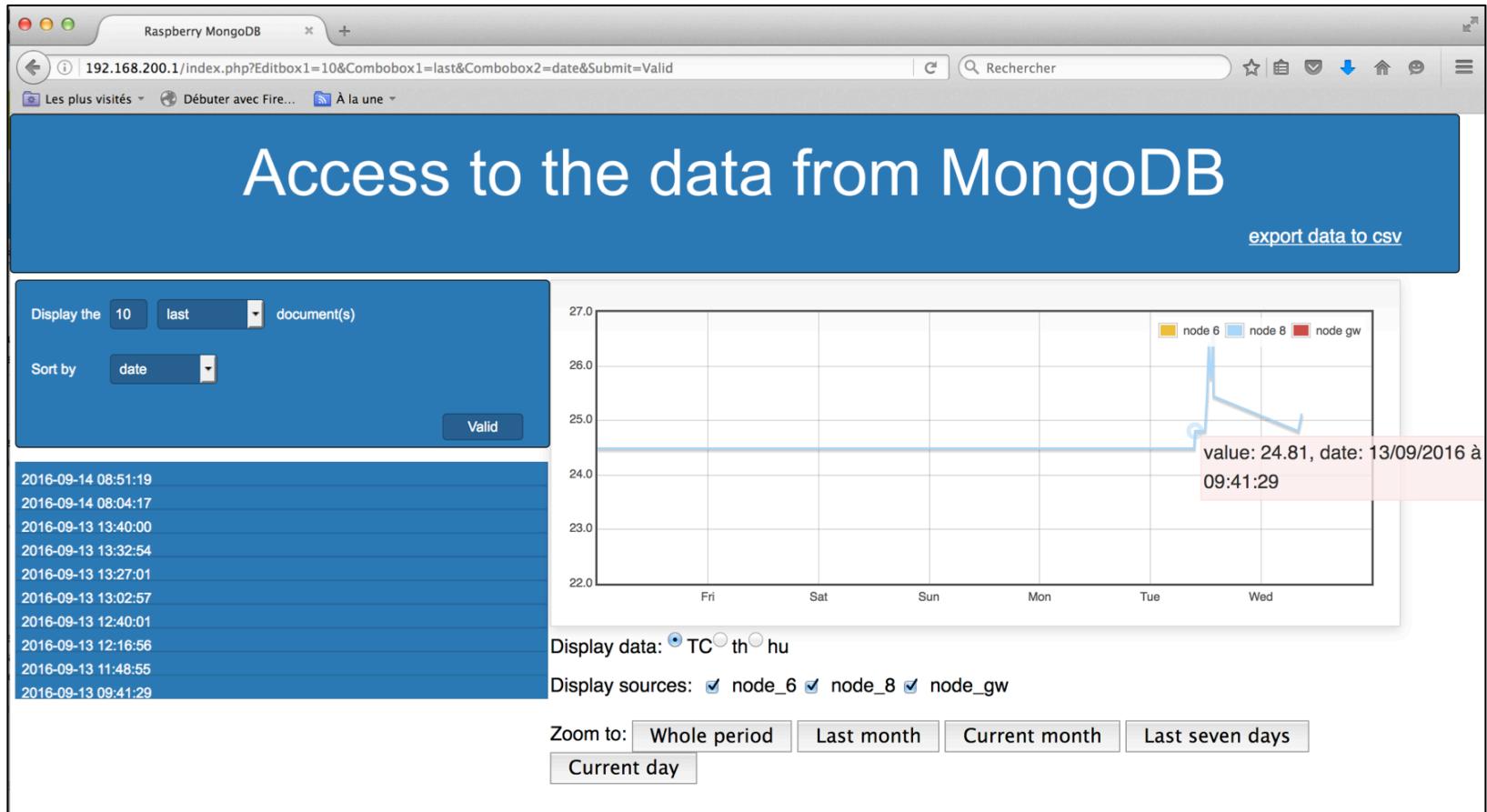
STANDALONE GATEWAY



CONNECT TO THE EMBEDDED WEB SERVER

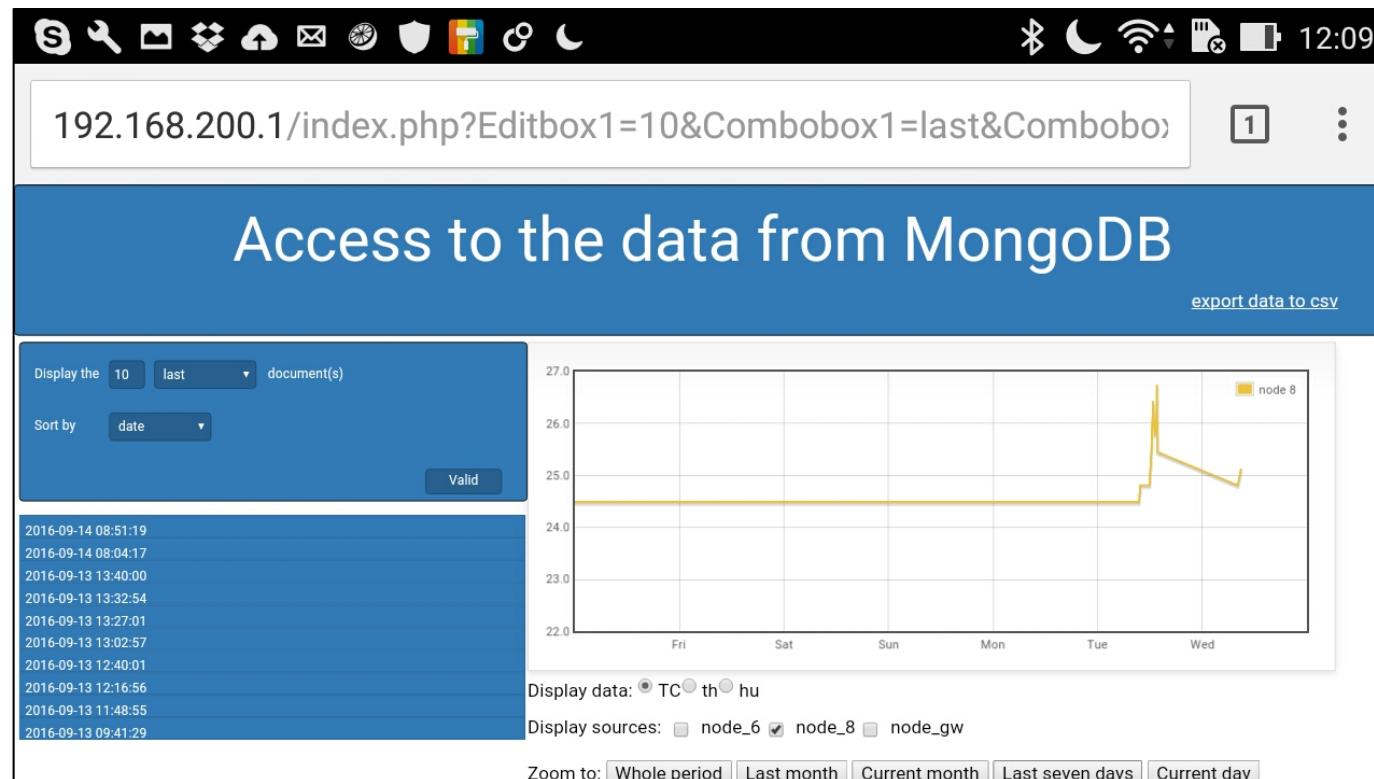
- On the WiFi interface
 - Gateway address is 192.168.200.1
- On the Ethernet interface
 - Gateway address is the IP address assigned by the DHCP server (of your LAN or laptop)
- Choose any of these solutions and open a web browser to enter the gateway IP address in the URL bar
 - <http://192.168.200.1>

DATA FROM THE LOCAL WEB SERVER



VISUALIZE IT ON YOUR SMARTPHONE!

- Don't forget to join the WS_PI_GW_xxxxxxxxxxx WiFi



RUNNING THE GATEWAY WITHOUT INTERNET ACCESS

The diagram illustrates the hardware and software components of the gateway. On the left, a Raspberry Pi is shown with various sensors and a WiFi module. Icons for MongoDB, PHP, and jQuery are displayed above the Pi, with a red arrow pointing from the Pi to a WiFi icon. A red dashed box highlights the WiFi module, and a red arrow points from this box to a Bluetooth icon.

Three screenshots of a mobile phone interface are shown below. The top screenshot shows a list of data documents with a graph of sensor values over time. The middle screenshot shows the 'NODES PREFERENCES' and 'DATES PREFERENCES' sections. The bottom screenshot shows a message indicating a CSV file has been created and saved.

A screenshot of a web browser window titled 'Access to the data from MongoDB' is shown. It displays a table of data and a line graph. Below the graph, there are options to 'Display data' or 'Retrieve data in a csv file'.

By Congduc Pham for WaterSense project

40

IMPROVING CASING AND ADDING PoE TO GATEWAY



By Congduc Pham for WaterSense project

OVERVIEW OF THE PARTS

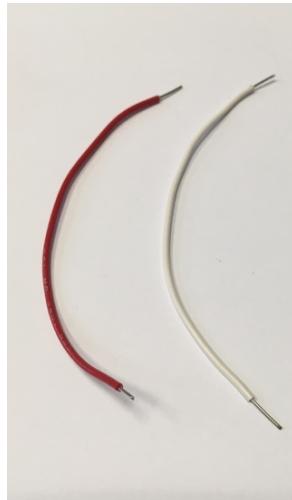


FIXING THE RASPBERRY TO THE CASE

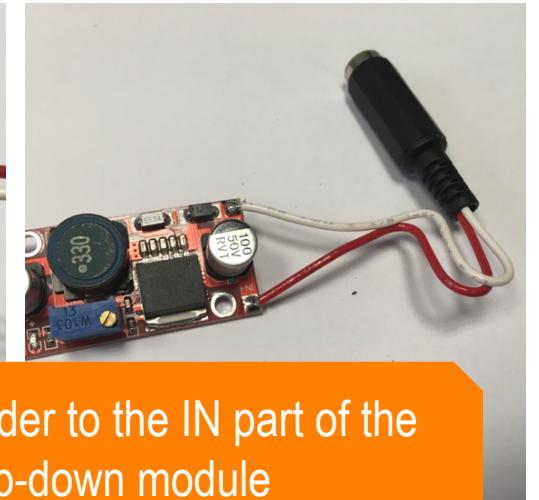
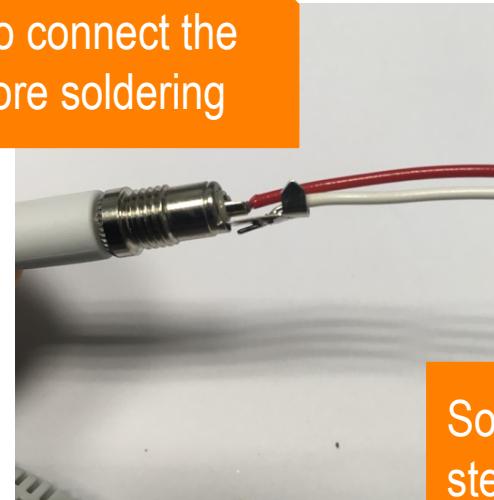


By Congduc Pham for WaterSense project

PREPARE THE DC STEP-DOWN (LM2596)



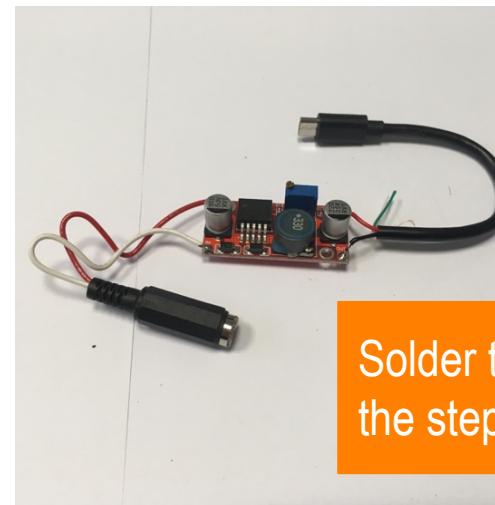
It is advised to connect the DC plugs before soldering



Solder to the IN part of the step-down module

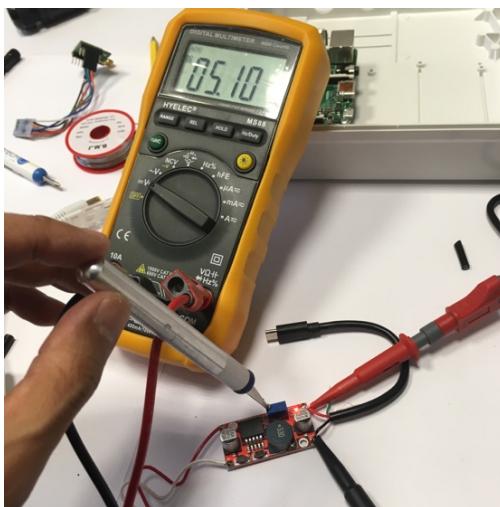
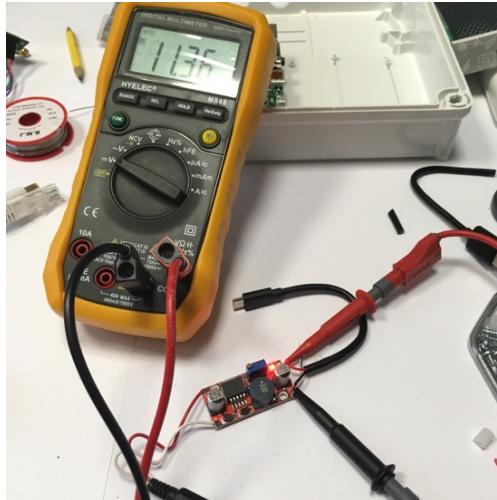


Cut a USB cable, keeping the micro-USB side



Solder to the OUT part of the step-down module

SETTING THE STEP-DOWN MODULE

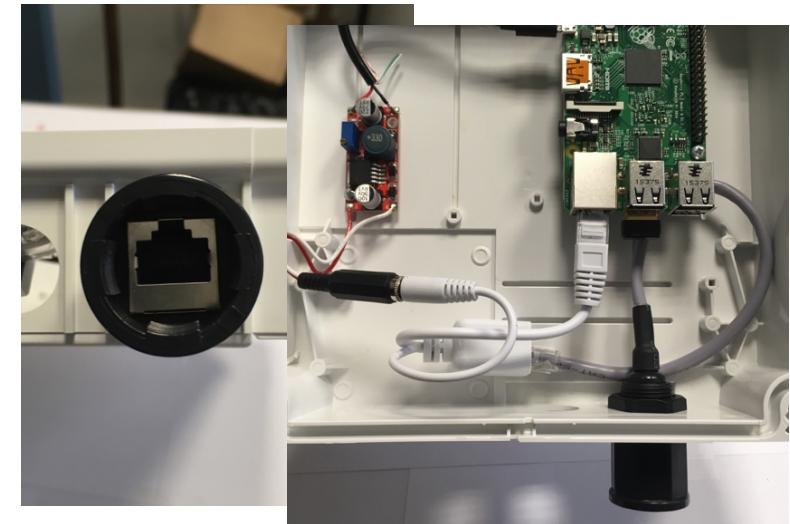
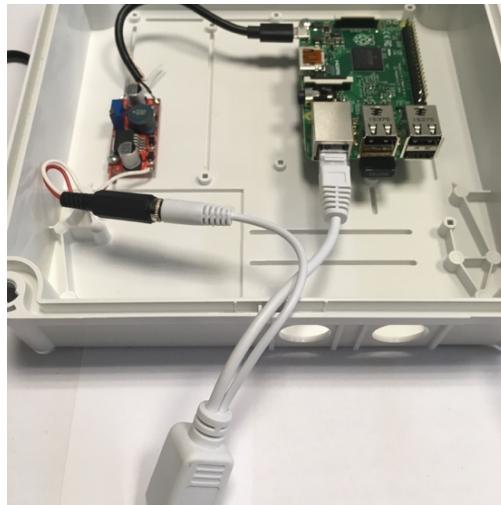


Use for instance a 9v, 12v or 18V AC-DC adaptor, connect to the IN plug, then check the output voltage with a voltmeter and turn the regulation screw until output is about 5.1v.

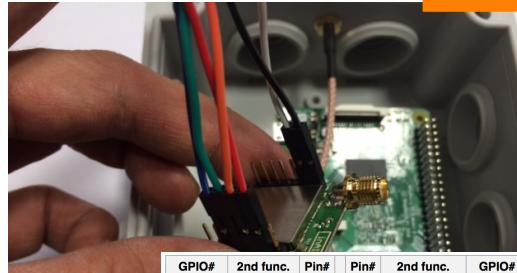


Then fix the module to the case

INSTALLING THE POE INJECTOR AND WATER-RESISTANT ETHERNET PLUG

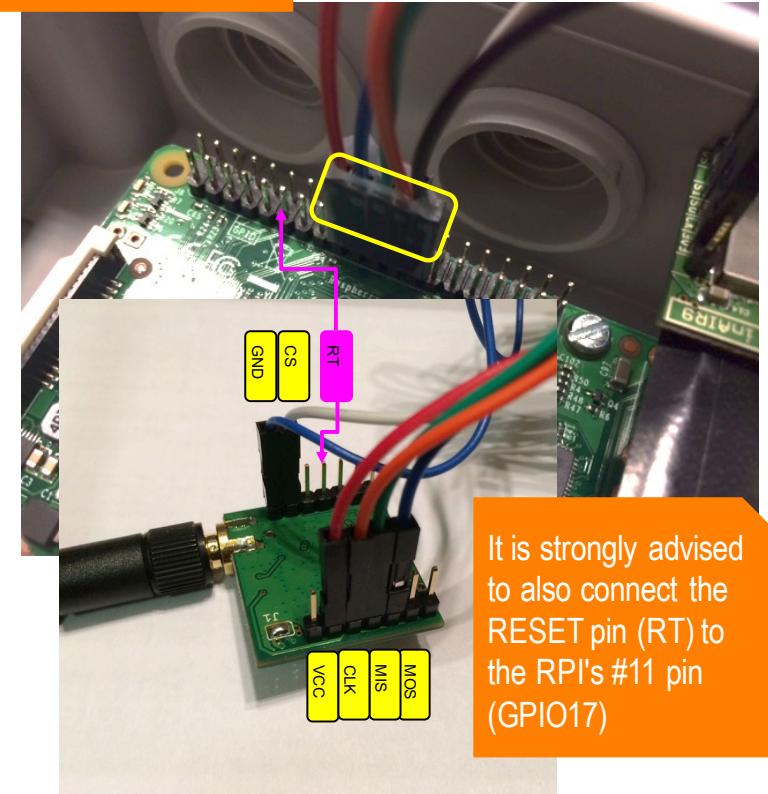


CONNECT THE RADIO MODULE



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	TXDD (UART)	14
	GND	9	10	RXDO (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

Like previously shown



INSTALL FIXING PARTS OF THE CASE

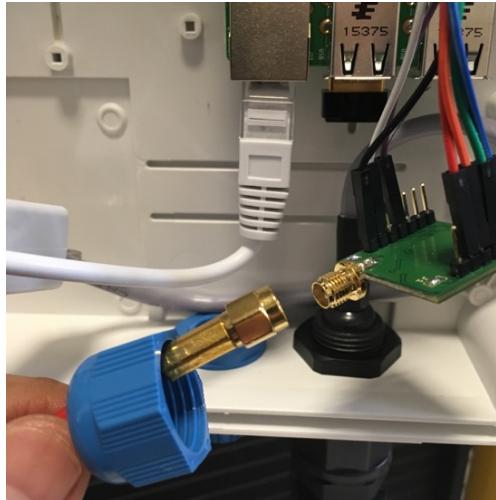


These parts of course depends on the case that you have.

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

