Raspberry PI

- CRÉER SA STATION MÉTÉO À L’AIDE DU RASPBERRY PI ET DE SON ÉCRAN 3.5’’

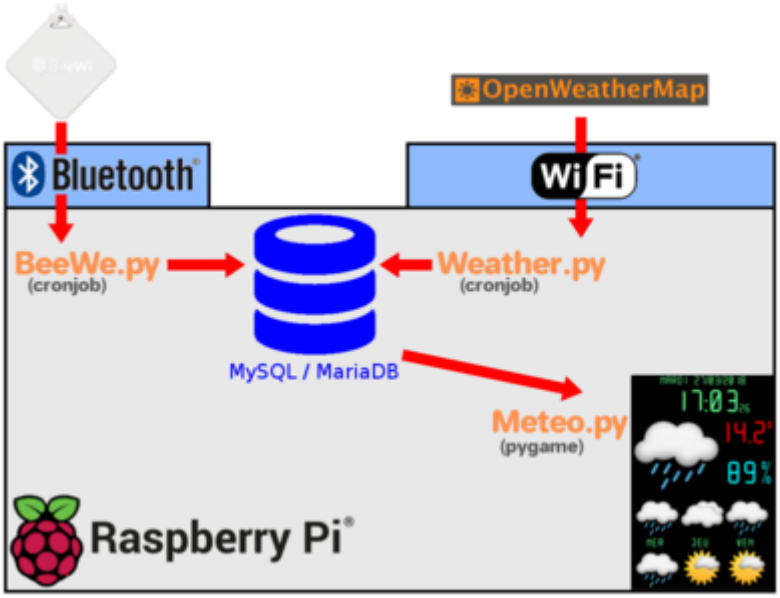
CRÉER SA STATION MÉTÉO À L'AIDE DU RASPBERRY PI ET DE SON ÉCRAN 3.5''

ÉTAPE 1 : INSTALLER LE RASPBERRY PI ET SON ÉCRAN LCD 3.5''

Nous allons nous baser sur cet écran, disponible sur Amazon :
https://www.amazon.fr/gp/product/B07ZQCM57G?pf_rd_p=3369e5a6-6989-43dc-ad01-b2b5ee1dcd12&pf_rd_r=V9EJBB7VE4VDETNKZ50Y ainsi que le raspberry pi 4.
Lors de ce tutoriel, nous allons supposer que votre Raspberry Pi ainsi que son écran LCD 3.5'' sont déjà correctement installés.
Si tel n'est pas le cas, voici tout de même l'installation simplifiée que vous pourrez retrouver ICI de manière plus détaillée :

```
sudo su
rm -rf LCD-show
git clone https://github.com/goodtft/LCD-show.git
chmod -R 755 LCD-show
cd LCD-show/
./MHS35-show 90
```

ÉTAPE 2 : RÉFLÉCHIR SON ARCHITECTURE



Pour une station météo, les données essentielles sont : les prévisions météorologiques ainsi que la température et l'humidité extérieure.Les prévisions météorologiques seront téléchargées du site OpenWeatherMap. La température extérieure sera mesurée par un capteur Bluetooth BeeWi-BBW200. Ce capteur a été choisi, car la communauté open source a développé une librairie d'accès python et que c'est un produit français facile à trouver. Toutes les données seront stockées dans une base de données MySQL/MariaDB (la base de données utilisée sera MariaDB, mais certains packages continuent à porter le nom MySQL).

Le capteur BeeWi peut être acheté ICI

ÉTAPE 3 : CRÉER LE PROJET

La première chose à faire est de créer le répertoire qui accueillera notre projet :

```
1
2
```

Créons maintenant notre fichier de configuration *meteo.ini* :

```
1
```

Copier, coller dedans le contenu ci-dessous :

```
1
2
3
4
```

```
password = meteo

location = localhost



[sensor_0]

type = BeeWi

UUID = a8b3fb43-4834-4051-89d0-3de95cddd318

MAC_ADDRESS = XX:XX:XX:XX:XX:XX

#A modifier dès que nous aurons récupéré l'adresse MAC du capteur bluetooth

retry = 5



[openweathermap]

url = http://api.openweathermap.org/data/2.5/forecast

APPID = XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX

#A modifier dès que nous aurons récupéré l'APPID du capteur bluetooth

id = XXXXXXX
```

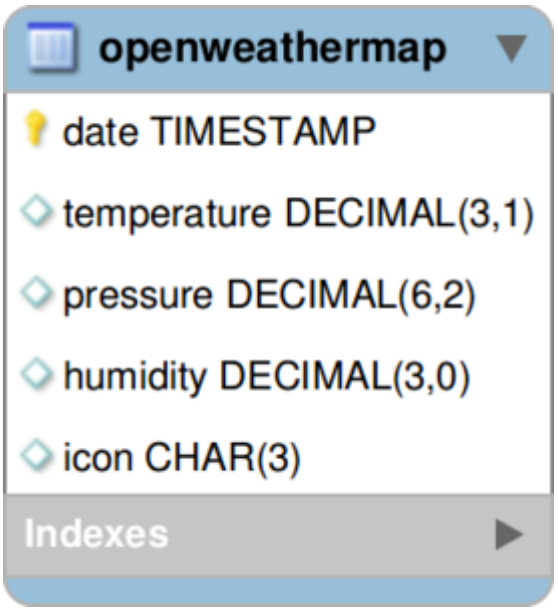
Un certain nombre de paquets sera utilisé pour réaliser ce projet qu'il faut installer :

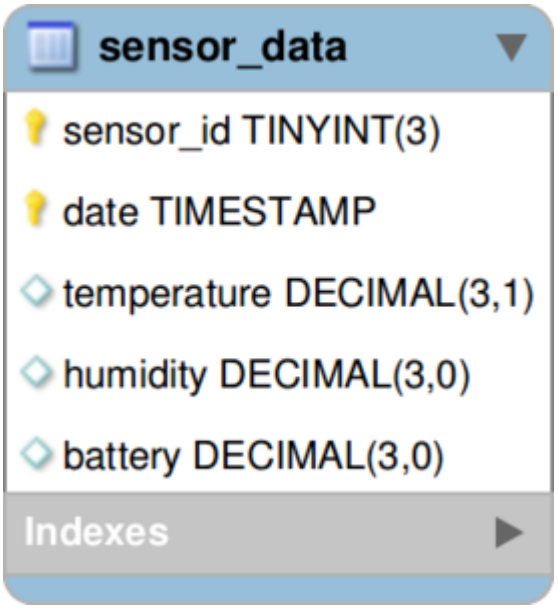
```
1
2
3
4
```

ÉTAPE 4 : CONFIGURER MYSQL/MARIADB

Pour stocker les informations, nous allons créer 2 tables : **`openweathermap`** et **`sensor_data`** qui nous permettront de stocker les prévisions météo OpenWeatherMap et le relevé des températures.

Voici à quoi elles ressembleront une fois créées :





Pour la création des tables, arrêter le service SQL :

```
1
```

Création du script de changement de mot de passe **`meteo.passwd.sql`** :

```
1
```

Puis copier, coller le contenu suivant et sauvegarder :

```
1
```

```

```

Configuration du mot de passe root pour MySQL/MariaDB :

```
1
```

Création du script de création de la base de données, des tables ainsi que de l'utilisateur **`meteo.database.sql`** :

```
1
```

Puis copier, coller le contenu suivant et sauvegarder :

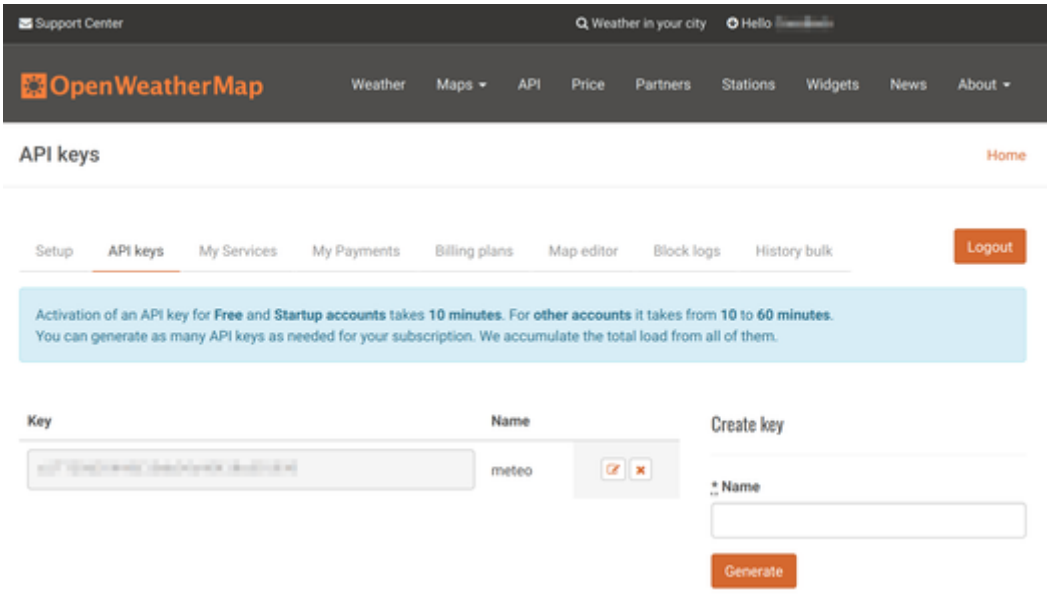
```
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
```

```
pressure decimal(6,2) DEFAULT NULL,

humidity decimal(3,0) DEFAULT NULL,

icon char(3) DEFAULT NULL,

PRIMARY KEY (date)

);

CREATE TABLE meteo.sensor_data (

sensor_id tinyint(3) unsigned NOT NULL,

date timestamp NOT NULL DEFAULT CURRENT_TIMESTAMP,

temperature decimal(3,1) DEFAULT NULL,

humidity decimal(3,0) DEFAULT NULL,

battery decimal(3,0) DEFAULT NULL,

PRIMARY KEY (sensor_id,date)

);
```

Exécutons maintenant le script dans la base de données MySQL/MariaDB :

```
1
```

ÉTAPE 5 : ACCÉDER AUX INFORMATIONS OPENWEATHERMAP



Les données qui nous intéresseront pour ce projet sont :

- la température (list[0].main.temp) ;
- la pression atmosphérique (list[0].main.pressure) ;
- l'humidité (list[0].main.humidity) ;
- l'icône de prévision météo (list[0].weather[0].icon) ;
- la date et l'heure de la prévision (list[0].dt_txt).

Téléchargeons la liste de villes :

```
1
```

Extrayons le fichier et récupérons le numéro de ville (id) :

```
1
```

```
2
```

Par exemple, l'id de Vernon dans le 27 est : 6453910

```
1
2
3
4
5
6
7
8
9
```

Pour pouvoir utiliser le service OpenWeatherMap, il faut nous enregistrer pour récupérer une clef d'accès APPID :

https://home.openweathermap.org/users/sign_up.

Avant d'aller plus loin, on teste que tout fonctionne bien en tapant l'URL dans son navigateur :

http://api.openweathermap.org/data/2.5/forecast?id=xxxxx&APPID=xxxxxxxxxxxxxxxxxxxxxx&units=metric

Il faut bien sûr saisir l'id et l'APPID correctement.

Dans le fichier de configuration meteo.ini, positionner l'id et l'APPID OpenWeatherMap. Editer le fichier de configuration meteo.ini :

```
1
```

Puis copier, coller le contenu suivant et sauvegarder :

```
1
2
3
4
5
```

```
units = metric
```

ÉTAPE 6 : CODER L'ACCÈS À OPENWEATHERMAP

Maintenant que nous avons accès aux données météo, nous allons créer un script récupérant ces informations pour les stocker dans notre base de données Weather.py :

```
1
```

Puis copier, coller le contenu suivant et sauvegarder :

```
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
```

```
#! /usr/bin/env python3

import configparser, json, datetime

import urllib.request

import mysql.connector



# Lecture du fichier de configuration

settings = configparser.ConfigParser()

settings._interpolation = configparser.ExtendedInterpolation()

settings.read('/home/pi/meteo/meteo.ini')



# Construction de l'url OpenWeatherMap

url_openweathermap =
'http://api.openweathermap.org/data/2.5/forecast?id=2976879&APPID=45b086df1d0050022426ab4ce5f2e9

#+ '?id=' + settings.get('openweathermap', 'id') \

#+ '&APPID=' + settings.get('openweathermap', 'APPID') \

#+ '&units=' + settings.get('openweathermap', 'units')



try:



# Récupération des prévisions météo et décodage JSON

webURL = urllib.request.urlopen(url_openweathermap)

data = webURL.read()

encoding = webURL.info().get_content_charset('utf-8')

infos=json.loads(data.decode(encoding))
```

```
except:


print("error reading url: "+url_openweathermap);

exit(1)


# Connexion à la base de données à l'aide des paramètres de configuration


cnx = mysql.connector.connect(user=settings.get('mysql', 'user'),

database=settings.get('mysql', 'database'),

password=settings.get('mysql', 'password'),

host=settings.get('mysql', 'location'))

cursor = cnx.cursor()


# Pour chaque prévision météo, on écrit ou remplace l'information en base


for item in infos["list"]:

date=item["dt_txt"]

icon=item["weather"][0]["icon"]

temperature=item["main"]["temp"]

humidity=item["main"]["humidity"]

pressure=item["main"]["pressure"]

cursor.execute('REPLACE INTO openweathermap VALUES (%s,%s,%s,%s,%s);',

(date,str(temperature),str(pressure),str(humidity),icon))
```

```
# validation et fermeture de la connexion à la base de données



cnx.commit()

cursor.close()

cnx.close()
```

Notez bien que la requête SQL est un REPLACE et non pas un INSERT.
Cette requête propre à MySQL/MariaDB permet de remplacer la ligne si elle existe déjà, l'insérer sinon.

Testons notre programme :

```
chmod +x Weather.py

./Weather.py

mysql -umeteo -pmeteo

MariaDB [(none)]> USE meteo;
MariaDB [meteo]> select * FROM openweathermap;

+---------------------+-------------+----------+----------+------+
| date                | temperature | pressure | humidity | icon |
+---------------------+-------------+----------+----------+------+
| 2018-05-30 12:00:00 |        22.7 |  1013.44 |       84 | 02d  |
| 2018-05-30 15:00:00 |        24.0 |  1012.43 |       74 | 01d  |
| 2018-05-30 18:00:00 |        23.3 |  1012.83 |       66 | 01d  |
| 2018-05-30 21:00:00 |        19.1 |  1013.47 |       73 | 03n  |
...
+---------------------+-------------+----------+----------+------+

40 rows in set (0,01 sec)
MariaDB [meteo]> exit
```

ÉTAPE 7 : ACCÉDER AUX INFORMATIONS DU CAPTEUR BEEWI-BBW200

Tout d'abord, il nous faut récupérer l'adresse Bluetooth du capteur :

```
bluetoothctl
[NEW] Controller XX:XX:XX:XX:XX:XX my-linux [default]
[bluetooth]# scan on
Discovery started
[CHG] Controller XX:XX:XX:XX:XX:XX Discovering: yes
```

```
[NEW] Device F0:C7:00:00:00:00 BeeWi SmartClim
[bluetooth]# exit
```

Dans cet exemple, l'adresse Bluetooth de mon capteur est F0:C7:00:00:00:00 (il s'agit d'une adresse factice).

Nous allons maintenant télécharger la librairie d'accès au capteur :

```
cd ~
git clone https://github.com/enrimilan/BeeWi-BBW200-Reader
cd BeeWi-BBW200-Reader/python/Reader/src
```

Insérer l'adresse Bluetooth dans le fichier Constants.py :

```
1
```

Puis copier, coller le contenu suivant et sauvegarder :

```
1
2
```

Test :

```
python3 Main.py
raw data:
00 a4 00 02 38 07 00 00 06 40
temperature: 16.4°C
humidity: 56%
battery: 64%
```

Copions la partie python dont nous avons besoin pour notre projet :

```
1
2
```

Le principal problème de cette librairie est que l'adresse Bluetooth ainsi que l'identifiant de l'appareil sont codés « en dur » dans le fichier utils/Constants.py. Nous voulons pouvoir configurer ces informations dans notre fichier meteo.ini. Modifions donc la fonction se chargeant de la lecture du capteur pour lui passer ces informations : reader/GattSensorReader.py :

```
1
```

Puis copier, coller le contenu suivant et sauvegarder :

```
nano ~/Reader/GattSensorReader.py
```

```
1
2
3
4
5
6
7
8
9
10
11
12
```

Il faut maintenant donner l'adresse Bluetooth de notre capteur à notre fichier de configuration meteo.ini :

```
import pygatt
```

```
1
```

```
def readRawData(self,MAC_ADDRESS,CHARACTERISTIC_UUID):
```

Puis copier, coller le contenu suivant *(en ayant pris compte la bonne adresse bluetooth du capteur)* et sauvegarder :

```
if(MAC_ADDRESS ==    ):
```

```
1
2
3
4
5
```

```
adapter = pygatt.GATTToolBackend()
```

ÉTAPE 8 : CODER L'ACCÈS AU CAPTEUR BEEWI-BBW200

```
UUID = a8b3fb43-4834-4051-89d0-3de95cddd318
```

Maintenant que nous avons accès aux données du capteur BeeWi, nous allons créer un script récupérant ces informations pour les stocker dans notre base de données BeeWi.py :

```
MAC_ADDRESS = 78:C7:  :  :  :
return ''.join('{:02x}'.format(x) for x in value)
```

```
1
```

Puis copier, coller le contenu suivant et sauvegarder :

```
nano ~/BeeWi.py
```

```
1
2
3
```

```
settings = configparser.ConfigParser()

settings._interpolation = configparser.ExtendedInterpolation()

settings.read('/home/pi/meteo/meteo.ini')



# Connexion à la base de données à l'aide des paramètres de configuration

cnx = mysql.connector.connect(user=settings.get('mysql', 'user'),

database=settings.get('mysql', 'database'),

password=settings.get('mysql', 'password'),

host=settings.get('mysql', 'location'))

cursor = cnx.cursor()



# Parfois, il est difficile de connecter le capteur, on fait donc plusieurs tentatives

try_count=int(settings.get('sensor_0', 'retry'));

while try_count>0 :

try:

sensorReader = GattSensorReader()

rawData = sensorReader.readRawData(settings.get('sensor_0', 'MAC_ADDRESS'),settings.get(
'sensor_0', 'UUID'))

sensorData = ParseUtils.parseSensorData(rawData.split(" "))

try_count=0
```

```
print(time.strftime("%c") + " : Tentative "+str(try_count))

try_count-=1

if (try_count==0):

print(time.strftime("%c") +" Lecture du capteur impossible")



# validation et fermeture de la connexion à la base de données

cnx.commit()

cursor.close()

cnx.close()
```

Testons notre programme :

```
chmod +x BeeWi.py
./BeeWi.py

mysql -umeteo -pmeteo
MariaDB [(none)]> USE meteo;
MariaDB [meteo]> SELECT * FROM sensor_data;
```

```
+-----------+---------------------+-------------+----------+---------+
| sensor_id | date                | temperature | humidity | battery |
+-----------+---------------------+-------------+----------+---------+
|     1     | 2018-06-14 18:04:00 |    23.8     |    65    |   100   |
+-----------+---------------------+-------------+----------+---------+

1 row in set (0,01 sec)
MariaDB [meteo]> exit
```

ÉTAPE 9 : PRÉPARER L'AFFICHAGE

Réfléchissons maintenant à l'affichage des données. Quelle orientation pour l'écran LCD ? Quelle police de caractères ? Quelles icônes à afficher ?

Pour la police de caractères, j'ai créé la mienne en suivant le tutoriel « Créer votre fonte symbole » de *Linux Pratique n°101*. Ne pas oublier les symboles ° et %.

Pour les icônes, j'ai trouvé des icônes libres d'utilisation que j'ai redimensionnées : https://d3stroy.deviantart.com/art/SILq-Weather-Icons-356609017.

Téléchargement des ressources :

```
1
2
```

ÉTAPE 10 : CODER L'AFFICHAGE DES DONNÉES MÉTÉORLOGIQUES

Tout est prêt, il ne nous reste plus qu'à coder le programme principal qui se chargera de l'affichage sur l'écran LCD. Ici, la librairie graphique qui a été retenue est pygame tout simplement, car je la maîtrise. Attention de bien commenter le code pour pouvoir le maintenir facilement. Créer le fichier Meteo.py :

```
1
```

Puis copier, coller le contenu suivant et sauvegarder :

```
nano ./Meteo.py
```

```
#! /usr/bin/env python3
```

```
seconds_since_last_refesh = refresh_rate
```

```
# Paramétrage MySQL

mysql_config={

'user': settings.get('mysql', 'user'),

'password': settings.get('mysql', 'password'),

'database': settings.get('mysql', 'database'),

'host': settings.get('mysql', 'location')



}# Connexion à MySQL

cnx = mysql.connector.connect(**mysql_config)



# Initialisation de l’affichage

pygame.init()

pygame.mouse.set_visible(0)

screen = pygame.display.set_mode(dimension,fullscreen,32)

font = pygame.font.Font(font_file, 20)

font_big = pygame.font.Font(font_file, 54)



# Le programme ne s'arrête jamais

while True:

now = datetime.datetime.now()



# Mise à jour de l'affichage des données ?

if(seconds_since_last_refesh >= refresh_rate) :

forecast_3d_icons=[]

forecast_icon=[]
```

```
txt_temperature="--.-"

txt_humidity="---"

cursor = cnx.cursor()



# Température et humidité extérieures

cursor.execute(sql_sensor)

for (date, temperature, humidity, battery) in cursor:

txt_temperature = "%.1f"%float(temperature) +"°"

txt_humidity = "%.0f"%int(humidity) + "%"



# Date du jour à minuit

midnight_date=now.strftime('%Y-%m-%d 00:00:00');



# prochaine prévision météo

cursor.execute(sql_forecast.replace('{DATE}',now.strftime('%Y-%m-%d %H:%M:%S')))

for (icon,date) in cursor:

forecast_icon.append(icon)



# prévision météo sur les 3 jours à venir

cursor.execute(sql_forecast_3d.replace('{DATE}',midnight_date))

for (icon_3d,date) in cursor:

forecast_3d_icons.append(icon_3d)

seconds_since_last_refesh=0

cursor.close()

cnx.commit()
```

```
# On efface l'écran pour le nouvel affichage

screen.fill((0, 0, 0))


# Affichage de l'image de prévision météo

for icon in forecast_icon :


# image_weather = pygame.image.load(img_dir+icon+'.png')

image_weather = pygame.image.load('/home/pi/meteo/img/'+icon+'.png')

screen.blit(image_weather, (-6, 60))


# Affichage de la date en haut

label = font.render(settings.get('translation','days').split(',')[datetime.datetime.today().
weekday()]+now.strftime(' %d/%m/%Y'), 1, color_green_lcd)

screen.blit(label, (dimension[0]/2-label.get_width()/2, 2))


# Affichage de l'heure (les secondes seront plus petites)

label = font_big.render(now.strftime('%H:%M'), 1, color_green_lcd)

screen.blit(label, (90, 30))

label = font.render(now.strftime('%S'), 1, color_green_lcd)

screen.blit(label, (240, 64))


# Affichage de la température

label = font_big.render(txt_temperature, 1, color_red_lcd)

screen.blit(label, (dimension[0]-label.get_width()-4, 100))
```

```
# Affichage de l'humidité

label = font_big.render(txt_humidity, 1, color_blue_lcd)

screen.blit(label, (dimension[0]-label.get_width(), 190))



# Affichage des prévisions météo sur les 3 prochains jours

index=0

for icon in forecast_3d_icons :

image_weather = pygame.image.load('/home/pi/meteo/img/'+icon+'_1.png')

screen.blit(image_weather, ((index-(index%2))/2*100+5, 260+(index%2)*116))

if (index%2)==0 :

label = font.render(settings.get('translation','days').split(',')[(datetime.datetime.today().
weekday()+1+int(index/2))%7][:3], 1, color_green_lcd)

screen.blit(label, (index/2*100+34, 364))

index+=1



# Affichage et attente 1s pour le prochain affichage

pygame.display.flip()

seconds_since_last_refesh+=1

time.sleep(1)

cnx.close()
```

Testons notre programme :

```
chmod +x Meteo.py
./Meteo.py
```

Tout fonctionne, il ne nous reste plus qu'à automatiser le démarrage de notre station météorologique.

Nous allons collecter les données périodiquement grâce au service crontab : la lecture du capteur de température toutes les 5 minutes et les prévisions météorologiques toutes les 15 minutes.

Crééons le fichier de configuration meteo.crontab :

```
1
```

Puis copier, coller le contenu suivant et sauvegarder :

```
1
2
```

On ajoute les lignes à la crontab avec la commande suivante :

```
1
```

Il ne nous reste plus qu'à désactiver l'économiseur d'écran et à lancer automatiquement notre affichage des données météorologiques. On crée le fichier meteo.autostart :

```
1
```

Puis copier, coller le contenu suivant et sauvegarder :

```
1
2
3
4
```

On ajoute ces commandes au démarrage de LXDE :

```
1
```

> Notre station météorologique est désormais opérationnelle. À vous de rajouter ce dont vous avez besoin…
> Je citerai comme exemple : un capteur de température intérieure, le support d'autres capteurs, l'affichage

des phases de la lune, l’affichage des marées...
Au niveau du code, vous pourrez aussi retirer toutes les constantes de localisation d’élément pour les mettre dans le fichier de configuration.