

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 :

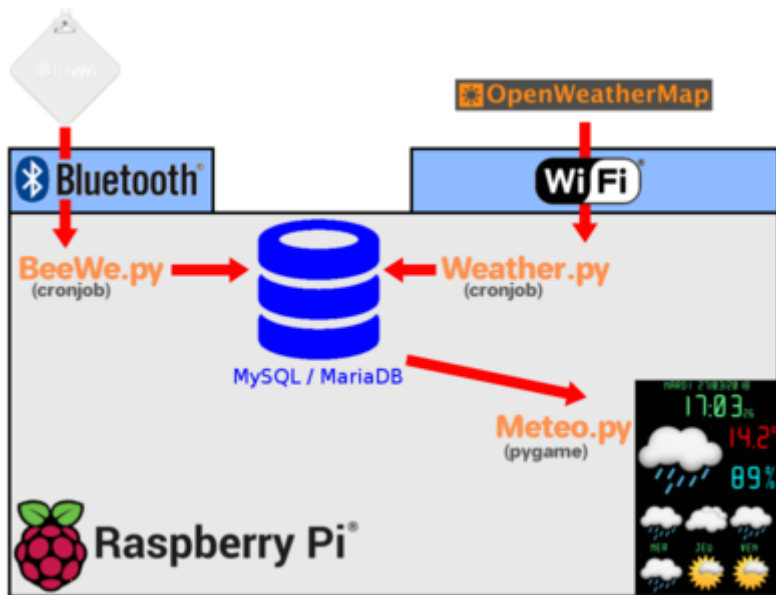
https://www.amazon.fr/gp/product/B07ZQCM57G?pf_rd_p=3369e5a6-6989-43dc-ad01-b2b5ee1dcd12&pf_rd_r=V9EJBB7VE4VDETNKZ50Y ainsi que le raspberry pi 4.

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 mkdir -p ~/meteo
```

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

```
1 [DEFAULT]
```

Copier, coller dedans le contenu ci-dessous :

```
1 [DEFAULT]
```

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

```

password = meteo

location = localhost

[sensor_0]

type = BeelWi

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 = XXXXXXXX

```

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

```

1
2
3
4

```

ÉTAPE 4 : CONFIGURER MYSQL/MARIADB

```
apt -y install mariadb-server-10.0 mariadb-client-10.0 python3-mysqldb.connector
```

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 :

openweathermap	
🔑	date TIMESTAMP
💎	temperature DECIMAL(3,1)
💎	pressure DECIMAL(6,2)
💎	humidity DECIMAL(3,0)
💎	icon CHAR(3)
Indexes ▶	

sensor_data	
🔑	sensor_id TINYINT(3)
🔑	date TIMESTAMP
💎	temperature DECIMAL(3,1)
💎	humidity DECIMAL(3,0)
💎	battery DECIMAL(3,0)
Indexes ▶	

Pour la création des tables, arrêter le service SQL :

```
systemctl stop mysql
```

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

```
mysql -u root -h /dev/null -e "ALTER USER 'root'@'localhost' IDENTIFIED BY 'root';" > /etc/mysql/mysql.conf.d/meteo.passwd.sql
```

Puis copier, coller le contenu suivant et sauvegarder :

```
mysql -u root -h /dev/null -e "CREATE DATABASE IF NOT EXISTS openweathermap; CREATE TABLE IF NOT EXISTS openweathermap.sensor_data (sensor_id TINYINT(3) UNSIGNED NOT NULL, date TIMESTAMP NOT NULL, temperature DECIMAL(3,1) NOT NULL, humidity DECIMAL(3,0) NOT NULL, battery DECIMAL(3,0) NOT NULL, icon CHAR(3) NOT NULL);" > /etc/mysql/mysql.conf.d/meteo.passwd.sql
```

Configuration du mot de passe root pour MySQL/MariaDB :

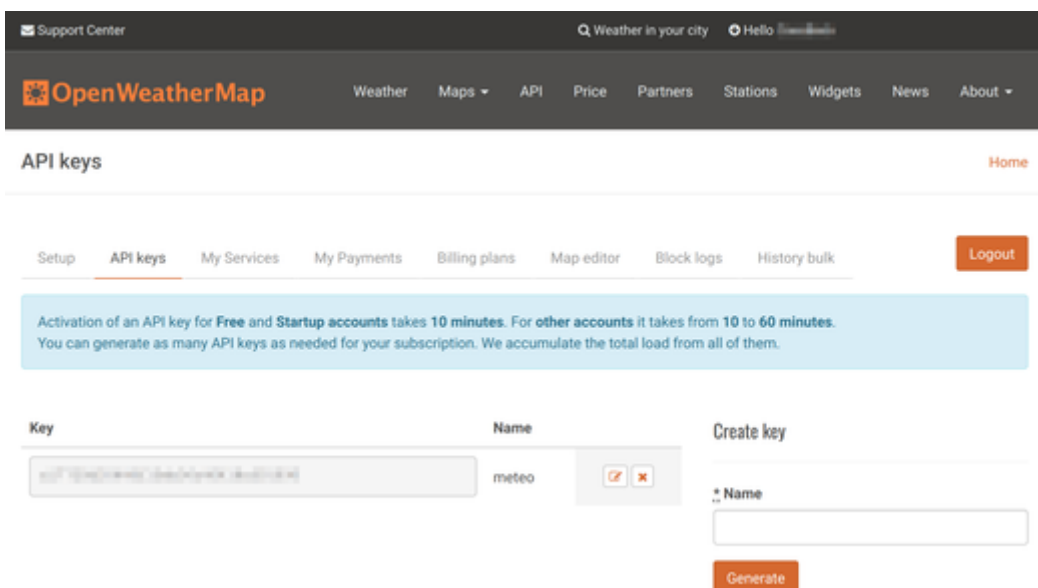
Création du script de création de la base de données, des tables ainsi que de l'utilisateur `meteo.database.sql` :

Puis copier, coller le contenu suivant et sauvegarder :

battery decimal(3,0) DEFAULT NULL

DDTMADV KEY (censor id date)

—



api.openweathermap.org/data/2.5/forecast?id=6453910	
JSON	Raw Data Headers
Save	Copy
cod:	"200"
message:	0.0433
cnt:	40
list:	
0:	
dt:	1528221600
main:	
temp:	19.85
temp_min:	19.85
temp_max:	22.08
pressure:	1010.1
sea_level:	1023.94
grnd_level:	1010.1
humidity:	76
temp_kf:	-2.23
weather:	
0:	
id:	803
main:	"Clouds"
description:	"broken clouds"
icon:	"04d"
clouds:	
all:	56
wind:	
speed:	5.56
deg:	39.0006
rain:	{}
sys:	
pod:	"d"
dt_txt:	"2018-06-05 18:00:00"
city:	
id:	6453910
name:	"Vernon"
coord:	
lat:	49.0833
lon:	1.4833
country:	"FR"

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) :

wget http://back.openweathermap.org/sample/city-list.json.gz

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 :

```
"id": "6453910",  
https://home.openweathermap.org/users/sign\_up.
```

```
"name": "Vernon"
```

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

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

```
"coord": {  
  "lon": 10.083328,  
  "lat": 49.083328
```

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

```
"id": "6453910",  
"name": "Vernon",  
"count": "FR",  
"coord": {  
  "lon": 10.083328,  
  "lat": 49.083328
```

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

```
"lat": 49.083328
```

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 :

```
nano ~/weather.py
```

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

32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54

```
#!/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=45b086df1d0050022426ab4ce5f2e9c'

#+ '?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()
```

the 1990s, the number of people in the UK who are aged 65 and over has increased by 1.5 million, and the number of people aged 75 and over has increased by 1 million (Office for National Statistics 2000). The number of people aged 65 and over is projected to increase to 10.5 million by 2026, and the number of people aged 75 and over to 6.5 million (Office for National Statistics 2000).

There is a growing awareness of the need to develop strategies to meet the needs of older people, and to ensure that they are able to live independently and actively in their own homes for as long as possible. This has led to a number of initiatives, including the development of age-friendly communities, and the establishment of local authority services to support older people. The aim of this paper is to review the current state of research on the needs of older people, and to identify areas where further research is needed.

The paper is organized as follows. First, we review the current state of research on the needs of older people, and identify areas where further research is needed. Second, we discuss the implications of this research for policy and practice. Finally, we conclude with some thoughts on the future of research on the needs of older people.

The first section of the paper reviews the current state of research on the needs of older people. This includes a review of the literature on the physical, psychological, and social needs of older people, and on the factors that influence these needs. The second section discusses the implications of this research for policy and practice.

The third section concludes with some thoughts on the future of research on the needs of older people. This includes a discussion of the need for further research on the physical, psychological, and social needs of older people, and on the factors that influence these needs. It also discusses the need for research on the development of strategies to meet the needs of older people, and on the establishment of local authority services to support older people.

The paper is organized as follows. First, we review the current state of research on the needs of older people, and identify areas where further research is needed. Second, we discuss the implications of this research for policy and practice. Finally, we conclude with some thoughts on the future of research on the needs of older people.

The first section of the paper reviews the current state of research on the needs of older people. This includes a review of the literature on the physical, psychological, and social needs of older people, and on the factors that influence these needs. The second section discusses the implications of this research for policy and practice.

The third section concludes with some thoughts on the future of research on the needs of older people. This includes a discussion of the need for further research on the physical, psychological, and social needs of older people, and on the factors that influence these needs. It also discusses the need for research on the development of strategies to meet the needs of older people, and on the establishment of local authority services to support older people.

The paper is organized as follows. First, we review the current state of research on the needs of older people, and identify areas where further research is needed. Second, we discuss the implications of this research for policy and practice. Finally, we conclude with some thoughts on the future of research on the needs of older people.

The first section of the paper reviews the current state of research on the needs of older people. This includes a review of the literature on the physical, psychological, and social needs of older people, and on the factors that influence these needs. The second section discusses the implications of this research for policy and practice.

The third section concludes with some thoughts on the future of research on the needs of older people. This includes a discussion of the need for further research on the physical, psychological, and social needs of older people, and on the factors that influence these needs. It also discusses the need for research on the development of strategies to meet the needs of older people, and on the establishment of local authority services to support older people.

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 = "00:00:00:00:00:00"
```

```
return ''.join('%02x' % format(x) for x in value)
```

1

Puis copier, coller le contenu suivant et sauvegarder :

```
nano - /BeeWi2.py
```

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
32
33
34
35
36
37
38

```
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 :



ÉTAPE 10 : CODER L'AFFICHAGE DES DONNÉES MÉTÉOROLOGIQUES

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 :

```
#!/usr/bin/env python3
# Meteo.py
# Programme principal de l'application météo
```


52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102

103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141

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

```
seconds_since_last_refresh = 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_refresh >= 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_refresh=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_refresh+=1

time.sleep(1)

cnx.close()
```

the 1990s, the number of people in the UK who are employed in the public sector has increased by 1.5 million, from 2.5 million in 1980 to 4 million in 1995. The public sector has also become an important employer of women, with 55% of public sector employees being women in 1995, compared with 45% in 1980.

There are a number of reasons why the public sector has become an important employer of women. One reason is that the public sector has a high proportion of jobs that are traditionally held by women, such as teaching, nursing, and social work. Another reason is that the public sector has a high proportion of jobs that are part-time or flexible, which are more likely to be held by women. A third reason is that the public sector has a high proportion of jobs that are in the service sector, which is also a sector that is traditionally held by women.

The public sector has also become an important employer of women because of the increasing demand for public services. As the population ages, there is a need for more social care services, which are provided by the public sector. There is also a need for more health care services, which are also provided by the public sector. This has led to an increase in the number of people employed in the public sector, and a corresponding increase in the number of women employed in the public sector.

The public sector has also become an important employer of women because of the increasing demand for public services. As the population ages, there is a need for more social care services, which are provided by the public sector. There is also a need for more health care services, which are also provided by the public sector. This has led to an increase in the number of people employed in the public sector, and a corresponding increase in the number of women employed in the public sector.

The public sector has also become an important employer of women because of the increasing demand for public services. As the population ages, there is a need for more social care services, which are provided by the public sector. There is also a need for more health care services, which are also provided by the public sector. This has led to an increase in the number of people employed in the public sector, and a corresponding increase in the number of women employed in the public sector.

The public sector has also become an important employer of women because of the increasing demand for public services. As the population ages, there is a need for more social care services, which are provided by the public sector. There is also a need for more health care services, which are also provided by the public sector. This has led to an increase in the number of people employed in the public sector, and a corresponding increase in the number of women employed in the public sector.

The public sector has also become an important employer of women because of the increasing demand for public services. As the population ages, there is a need for more social care services, which are provided by the public sector. There is also a need for more health care services, which are also provided by the public sector. This has led to an increase in the number of people employed in the public sector, and a corresponding increase in the number of women employed in the public sector.

The public sector has also become an important employer of women because of the increasing demand for public services. As the population ages, there is a need for more social care services, which are provided by the public sector. There is also a need for more health care services, which are also provided by the public sector. This has led to an increase in the number of people employed in the public sector, and a corresponding increase in the number of women employed in the public sector.

The public sector has also become an important employer of women because of the increasing demand for public services. As the population ages, there is a need for more social care services, which are provided by the public sector. There is also a need for more health care services, which are also provided by the public sector. This has led to an increase in the number of people employed in the public sector, and a corresponding increase in the number of women employed in the public sector.

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.

Créons 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
```

```
@xset -dpms
```

```
/home/pi/meteo/Meteo.py
```

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.

Revision #5

Created 2026-04-24 07:42:39 UTC by Nico là

Updated 2026-04-24 08:10:59 UTC by Nico là