
RAPPORT DE STAGE AU CENTRE PROVINCIAL METEOROLOGIQUE

DU 01 JUILLET AU 31 AOUT 2021



Ministère de l'Équipement, du Transport, de la Logistique et de l'Eau
DIRECTION GÉNÉRALE DE LA MÉTÉOROLOGIE



Réalisé par : ASMAA TOUNSI
Encadré par : MERYEM AMEUR
RABIA MERROUCHI

REMERCIEMENT :

Je tiens en premier lieu à remercier vivement les directeurs de la Direction de la Météorologie National

Mes remerciements vont également à monsieur MERROUCHI Rabia, pour le partage de son expertise, son encadrement, ses conseils et ses remarques pertinentes.

Je voudrais aussi remercier le personnel du CPM de Casablanca, pour son soutien technique, sa disponibilité, ses conseils généraux ainsi que l'intérêt spécial qui nous a accordé tout au long du stage.

Je profite de cette occasion pour remercier l'administration de l'École Marocaine Des Sciences De L'Ingénieur, madame AMEUR Meryem pour son suivi tout au long de notre stage et le service de formation de la Direction de la Météorologie National pour leur assistance.

Et finalement, je voudrais exprimer mon reconnaissance et gratitude à toute personne ayant contribué de près ou de loin à la réussite de mon stage.

Table des Matières

I.	Remercîment.....	2
II.	Table des Matières.....	3
III.	Listes des figures.....	6
IV.	Listes des tableaux.....	8
V.	Listes des abréviations.....	9
VI.	Introduction générale.....	10
VII.	Chapitre 1 : La DMN et la monographie du CPM de Casablanca.....	11
	1. La DMN et ses directions régionales.....	11
	2. Présentation générale du CPM de ANFA.....	11
	3. Organigramme du CPM.....	12
	4. Les valeurs extrêmes enregistrées au CPM.....	12
VIII.	Chapitre 2 : Travail effectuée.....	13
	→ Partie 1 : Contexte général de projet.....	13
	1. Etude et analyse de l'existant.....	13
	→ Partie 2 : Etude technique du projet.....	14
	1. Identification des besoins.....	14
	2. Capteurs.....	14
	○ Bme280.....	14
	○ DHT22.....	14
	3. Raspberry Pi.....	15
	4. Montage du circuit et configuration.....	16
	○ Raspberry Pi – Bme280.....	16
	• Configuration de l'interface I2C.....	16
	• Circuit.....	18

○ Raspberry Pi – DHT22.....	18
• Circuit.....	19
• Installation de la bibliothèque Adafruit DHT.....	19
IX. Chapitre 3 : Déroulement du projet.....	20
→ Partie 1 : Partie préparation et création.....	20
1. Préparation de l’environnement sous Raspbian	20
○ Installation de Apache web server.....	20
○ Mise en place de PHP.....	20
○ Mise en place de MYSQL.....	20
○ Mise en Place de PHPMYADMIN.....	21
○ Installation de Python.....	22
2. Création d’un projet Django.....	22
○ Configuration de l’environnement virtuel.....	23
○ Installation de Django et Configuration du projet.....	23
→ Partie 2 : Partie réalisation et Commandes	25
1. Création de l’application MeteoApp.....	25
2. Création de la bdd meteo(models+migration)	25
3. Script du capteur DHT22.....	28
○ Récupérations des valeurs.....	28
○ Insertion dans la bdd.....	29
○ Sélection et insertion des maxs et mins.....	29
4. Urls & Views.....	30
○ Ajouts des views.....	30
○ Liaison des views avec leurs Urls.....	32

→	Partie 3 : Interface.....	33
	1. Valeurs instantanées.....	33
	2. Historique.....	34
	3. Message.....	33
→	Partie 4 : Partie logiciel.....	36
X.	Conclusion générale.....	37
XI.	Bibliographie.....	38

Liste Des Figures

Figure 1 : organigramme du CPM.....	12
Figure 2 : capteur BME280.....	14
Figure 3 : capteur DHT22.....	15
Figure 4 : Raspberry Pi.....	15
Figure 5 : Interface Options.....	16
Figure 6 : Activation I2C.....	17
Figure 7 : Schéma du montage Raspberry Pi et capteur Bme280.....	18
Figure 8 : Schéma du montage Raspberry Pi et capteur DHT22.....	19
Figure 9 : Login PhpMyAdmin.....	22
Figure 10 : Interface PhpMyAdmin.....	22
Figure 11 : Création myenv.....	23
Figure 12 : Activation myenv.....	23
Figure 13 : Adresses autorisées.....	23
Figure 14 : Configuration static.....	24
Figure 15 : Création MeteoApp et ajout de static et templates.....	25
Figure 16 : Diagramme de classe du projet.....	25
Figure 17 : Connexion base de données.....	26
Figure 18 : Class Valeurs.....	26
Figure 19 : Class Min_Temp.....	26
Figure 20 : Class Max_Temp.....	27
Figure 21 : Class Min_Hum.....	27
Figure 22 : Class Max_Hum.....	27
Figure 23 : Class Min_Pres.....	27
Figure 24 : Class Max_Pres.....	27
Figure 25 : Migration de models.....	28
Figure 26 : Script DHT22.....	28

Figure 27 : Fonction insertIntoDatabase.....	29
Figure 28 : Fonctions max_temp et min_temp.....	29
Figure 29 : Action de l'exécution du script.....	29
Figure 30 : View fonction valeurs_page.....	30
Figure 32 : View fonction temperature_page.....	31
Figure 32 : View fonction metar_page.....	31
Figure 33 : Urls.....	32
Figure 34 : Interface Valeurs Instantanées 1.....	33
Figure 35 : Interface Valeurs Instantanées 2.....	33
Figure 36 : Interface Température 1.....	34
Figure 37 : Interface Température 2.....	34
Figure 38 : Interface Metar 1.....	35
Figure 39 : Interface Metar 2.....	35

Liste Des Tableaux

Tableau 1 : connexion Broche Bme280 avec Raspberry Pi.....	17
Tableau 2 : connexion Broche DHT22 avec Raspberry Pi.....	19

Liste Des Abréviation

DMN	Direction de la Météorologie Nationale
CPM	Centre Provincial de Casablanca
DESC	Description
BDD	Base de données

INTRODUCTION GENERALE :

Dans le cadre de la formation des ingénieurs, l'École Marocaine Des Sciences De L'Ingénieur exige à ses élèves ingénieurs d'effectuer un stage d'observation à la fin de leur troisième année.

Ce stage permet aux élèves ingénieurs d'agrémenter leurs notions théoriques par la pratique, de s'accoutumer et se familiariser avec le monde professionnel et d'enrichir le savoir-faire et l'esprit technique. Durant ce stage, l'élève ingénieur est appelé à mettre en œuvre les capacités acquises pendant sa formation, à combler ses lacunes et à approfondir ses connaissances, tout ceci entant que futur ingénieur et la responsabilité que ce poste exige en termes de décision, de créativité et de développement.

Ce stage a été réalisé dans le domaine Météo plus particulièrement, la conception et réalisation d'une application web de visualisation de données mesurer par une station automatique de mesures météorologique.

Le présent rapport contient les éléments suivants :

- L'Etude technique du projet.
- Préparation et création.
- La réalisation.

De ce fait, je présente ce rapport qui contient toutes mes activités durant mon stage, les différentes tâches effectuées, les informations théoriques et pratiques relatives à chaque partie du rapport.

Chapitre 1 : La DMN et la monographie du CPM de Casablanca

1. La DMN et ses directions régionales :

La Direction de la Météorologie Nationale « DMN » est un établissement d'état sous la tutelle du ministère de l'équipement, du Transport, de la Logistique et de l'Eau. Afin d'assurer une bonne expertise et répondre aux attentes de ses différents partenaires, la DMN a choisi d'axer une politique de régionalisation de ses services sur le territoire marocain. Ainsi, six directions météorologiques régionales ont été créées permettant l'élaboration et la mise à disposition de prestations plus adaptées à la région.

2. Présentation générale du CPM de Casablanca :

Le Centre Provincial de Casablanca fait partie de la direction Régionale de Météorologie Centre- Ouest DRMCO dont celle-ci comporte 9 autres centres provinciaux (Benslimane-ANFA – Bengrir – El Jadida – Essaouira – Mohammedia - Casa Nouacer – Safi – Settat). Ce centre a été créé en 1911. Il se situe au. Le siège du CPM de Casablanca est composé de :

- Bureau de chef.
- Cellule d'Observation.
- Cellule de Climatologie.
- Cellule de Maintenance.

Le CPM de Casablanca se trouve dans le centre de la ville, il est exposé aux développements des nuages cumuliformes pendant la période d'été.

3. Organigramme du CPM :

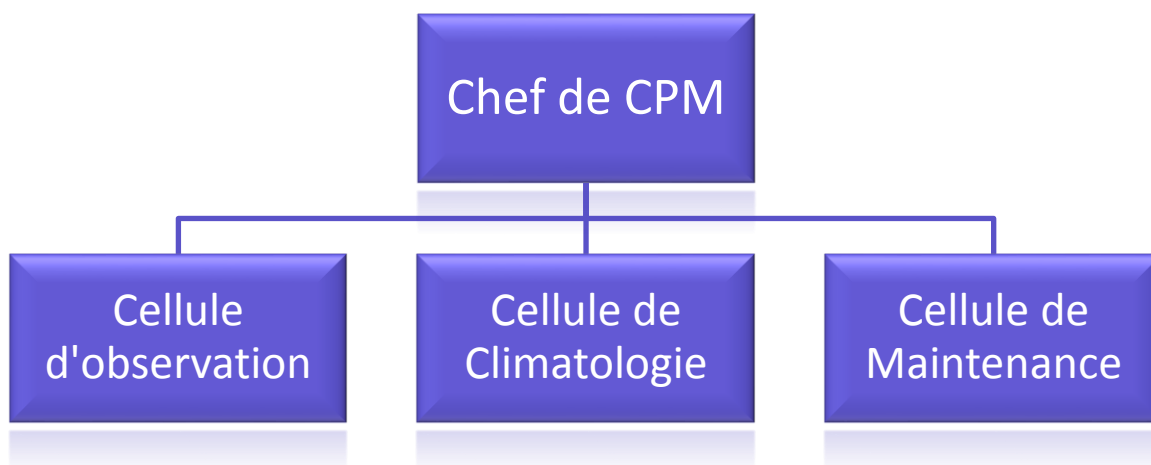


Figure 1 : organigramme du CPM

4. Les valeurs extrêmes enregistrées au CPM :

La température maximale : +46.1°C le 19/07/1967

La température minimale : -02.7°C le 03/01/1932

Relevées quotidiennes de précipitations : 177. 9 mm le 29/11/2010

Relevées mensuelles de précipitations : 296.4 mm novembre 2010

La pointe maximale est de 155 km/h observée le 28/02/1965 d'une direction ouest.

Chapitre 2 : Travail effectuée :

Partie 1 : Contexte général de projet

1. Etude et Analyse de l'existant :

- Système de télémessure (data logger type Thiès climat) est une ancienne machine fabriquée en 1998 dotée d'une liaison de communication série type RS422 avec une station de travail tournant sous système d'exploitation Windows XP, une application de supervision fonctionnant seulement sur cette version d'exploitation (pas de mise à jour pour les nouveaux systèmes d'exploitation).
- Manque de port série dans les nouvelles stations de travaux.
- Ancienne interface Graphique de supervision non adaptée aux différents environnements.
- Manque de fonctionnalités d'exploitation (calcul à la main)

Partie 2 : Etude technique du projet

1. Identification des besoins :

Une application web de supervision s'adaptant aux différents types de machines et systèmes d'exploitation, s'avère la mieux convenante.

2. Capteurs :

○ Bme280

Le Bme280 est un capteur qui permet de mesurer une température, une pression barométrique et le pourcentage d'humidité. Ce capteur est idéal pour tous les projets climatiques ou environnementaux et très modulable par sa connexion en I2C ou SPI. Techniquement il est caractérisé par :

- Tension d'alimentation : 3 à 5V
- Plage de température : -40 à +85°C
- Plage d'humidité : 0-100%
- Plage de pression : 300-1100 hPa
- Dimensions : 19.0mm x 18.0mm x 3.0mm
- Broche VIN : tension d'alimentation comprise entre 3 et 5V
- Broche GND : masse
- Broche SCL : broche d'horloge du bus I2C
- Broche SDA : broche de donnée du bus I2C

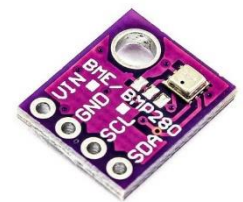


Figure 2 : capteur Bme280

○ DHT22

Le DHT22 est un capteur numérique de base, à faible coût permettant de mesurer de manière efficace la température et l'humidité de l'air ambiant grâce à sa combinaison de deux en un d'un capteur humidité capacitif et d'une thermistance.

Techniquement il est caractérisé par :

- Tension d'alimentation : 3 à 5V
- Plage de température : -40 à +80°C
- Humidité de : 0 à 100% RH

- Dimension : 15.1*25*7.7mm
- Broche 1 : tension d'alimentation
comprise entre 3 et 5V
- Broche2 : entrée des données
- Broche3 : non connecté
- Broche 4 : Masse



Figure 3 : capteur DHT22

3. Raspberry Pi :

La Raspberry Pi est un nano-ordinateur à faible coût qui se branche sur un écran d'ordinateur ou un téléviseur avec clavier et souris standard. C'est un petit appareil capable qui permet aux personnes de tous âges d'explorer l'informatique et d'apprendre à programmer dans des langages comme Scratch et Python.

Techniquement elle est caractérisée par :

- Carte mère Raspberry Pi 4
- Processeur : Broadcom BCM2711, quad-core Cortex-A72 (ARM v8)
64-bit SoC @ 1.5GHz
- RAM : 8 Go LPDDR4
- GPU : VideoCore VI prenant en charge OpenGL ES 3.0, décodage HEVC 4K à 60 i/s
- Connexion sans fil : Bluetooth 5.0, Wi-Fi 802.11b/g/n/ac
- Connexion filaire : Gigabit Ethernet (RJ45)
- Lecteur de carte micro-SD (stockage non fourni)
- Port caméra CSI pour connecter la caméra Raspberry Pi
- Port d'affichage DSI pour connecter l'écran tactile Raspberry Pi
- Audio : AV 3.5 mm
- Ports : 2 x USB 3.0 / 2 x USB 2.0 / 1 x USB-C (alimentation seulement) / 1 x GPIO 40 pin / 1 x port quadripôle Audio/Vidéo composite / 2 x micro-HDMI
- Alimentation : 5V DC via un connecteur USB-C (minimum 3A), 5V DC via un entête GPIO (minimum 3A), compatible Power over Ethernet (PoE) (nécessite un HAT pour PoE)



Figure 4 : Raspberry Pi

4. Montage circuit et configuration :

- Raspberry Pi + Bme280 :
 - Configuration de l'interface I2C :

Avant d'entamer le montage du circuit Raspberry Pi + Bme280 il faut avant tout activer l'interface I2C dans le Raspberry pi :

→ Etape 1 : Activation de l'interface I2C :

A partir du terminal on commence par exécuter la commande suivante : **Sudo raspi-config**

Cette commande lancera l'utilitaire raspi-config, puis on sélectionne **Interface Options** :

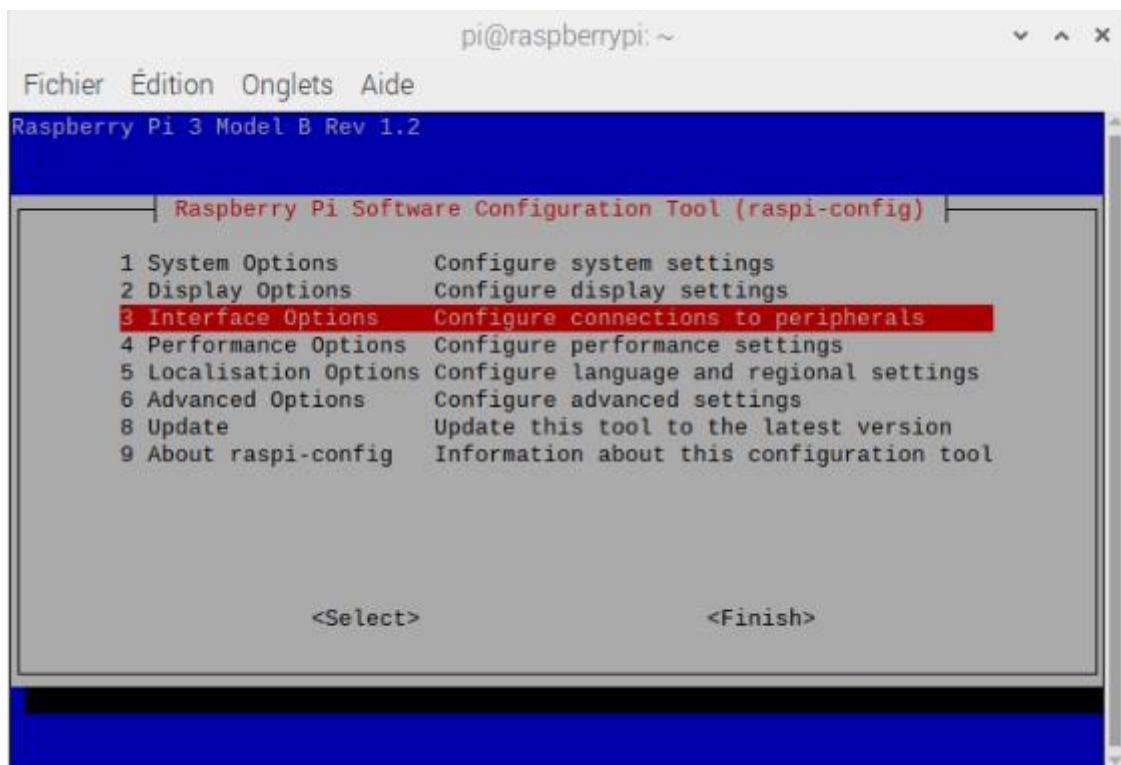
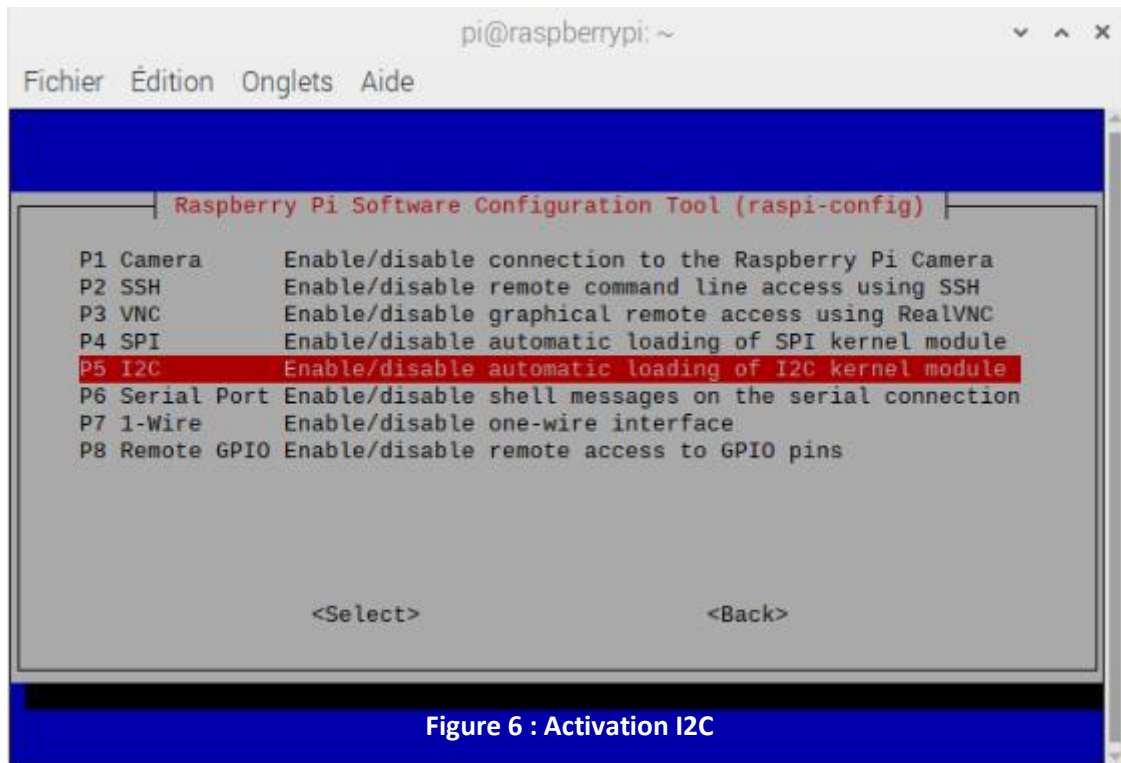


Figure 5 : Interface Options

On active ensuite l'option I2c :



Après la Raspberry pi va se redémarrer et l'interface I2C sera activé.

→ Etape 2 : Installation des utilitaires :

Pour aider au débogage et permettre à l'interface d'être utilisée dans Python, nous pouvons installer « python-smbus » et « i2c-tools » en exécutant les commandes suivantes :

Sudo apt-get update

Sudo apt-get install -y python-smbus i2c-tools

→ Etape 3 : Arrêt :

Arrêt de la Raspberry Pi en exécutant la commande : **Sudo halt**

Puis on attend dix secondes, On débranche l'alimentation de la Raspberry Pi et on est maintenant prêt à connecter notre matériel I2C.

- Circuit :

Le tableau ci-dessous montre comment le module est connecté à l'en-tête GPIO du Raspberry Pi (P1) :

Broches Bme280	Desc	Broches en-tête GPIO
VCC	3.3V	P1-01
GND	Masse	P1-06
SCL	I2C SCL	P1-05
ADD	I2C SDA	P1-03

Tableau 1 : connexion Broche Bme280 avec Raspberry Pi

Voici un schéma d'une configuration de maquette. On connecte les quatre broches du module directement au Pi, On aura besoin que de quatre fils femelle-femelle.

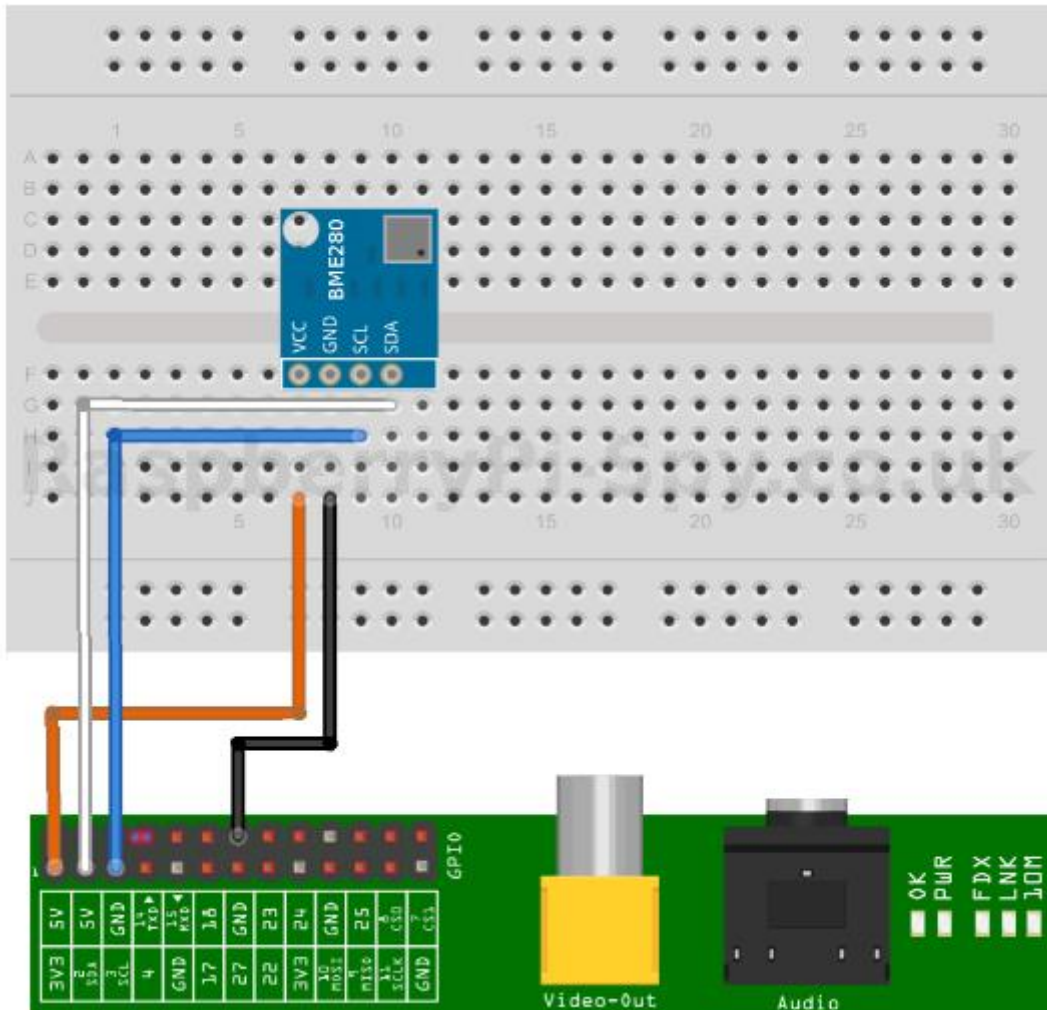


Figure 7 : Schéma du montage Raspberry Pi et capteur Bme280

Il nous reste maintenant que de télécharger le script BME280 en exécutant la commande :

Wget -O bme280.py <http://bit.ly/bme280py>

- Raspberry Pi + DHT22 :
- Circuit :

Le tableau ci-dessous montre comment le module est connecté à l'en-tête GPIO du Raspberry Pi :

Broche DHT22	Broches en-tête GPIO
Pin 1	3v3
Pin2	GPIO4
Pin4	GND

Tableau 2 : connexion Broche DHT22 avec Raspberry Pi

Sans oublier une résistance de 10K entre Pin1 et Pin2 du capteur DHT22.

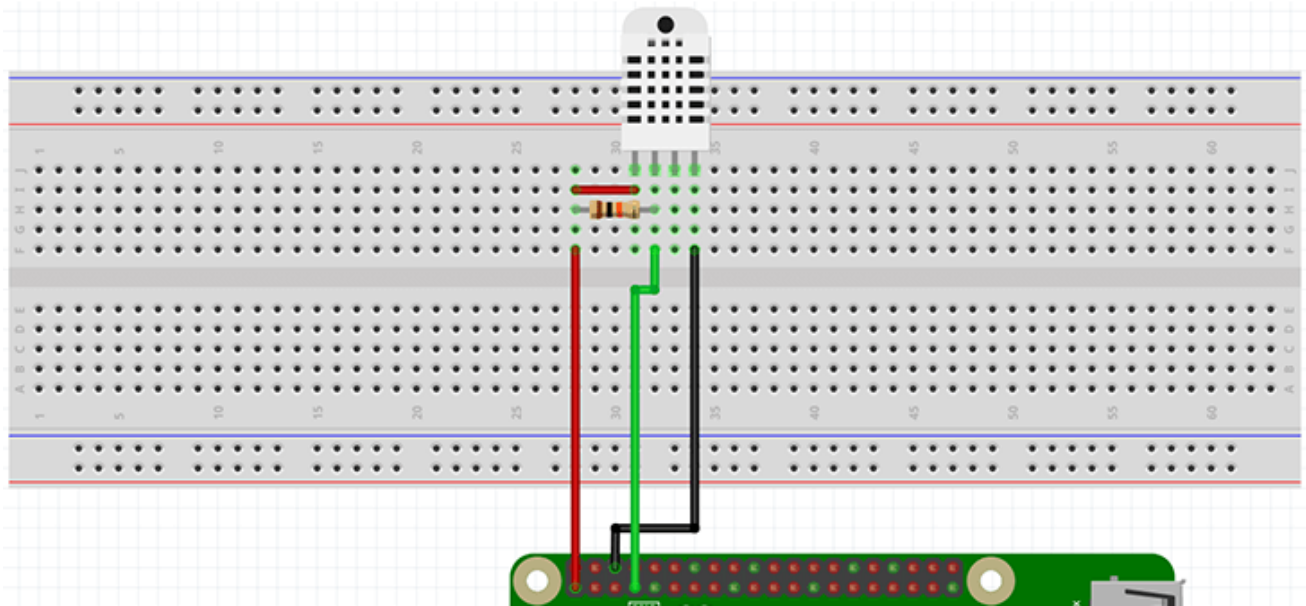


Figure 8 : Schéma du montage Raspberry Pi et capteur DHT22

- Installation de la bibliothèque Adafruit DHT :

Avant notre code python, On doit télécharger et installer la bibliothèque DHT dans notre Raspberry Pi. Dans le terminal on exécute les commandes suivantes :

Git clone https://github.com/adafruit/Adafruit_Python_DHT.git

Cd Adafruit_Python_DHT

Sudo apt-get update

Sudo apt-get install build essential python-dev

Sudo python setup.py install

Il nous reste maintenant qu'à redémarrer notre Raspberry afin de récupérer le pilote Adafruit .

Chapitre 3 : Déroulement du projet :

Partie 1 : Partie préparation et création

1. Préparation de l'environnement sous Raspbian :

- Installation de Apache web server :

Apache est un des serveurs web les plus populaires disponibles pour la Raspberry Pi, il peut servir des fichiers HTML via les protocoles Web HTTP et HTTPS.

Avant d'installer Apache sur notre Raspberry Pi, nous devons d'abord nous assurer que la liste des packages est à jour en exécutant les deux commandes suivantes :

Sudo apt-get update

Sudo apt-get upgrade

Maintenant on peut entamer l'installation d'apache2 avec la commande suivante :

Sudo apt install apache2 -y

Il nous reste qu'ajouter l'utilisateur pi au groupe www-data, puis donner la possession de tous les fichiers et dossiers du répertoire /var/www/html au groupe www-data à travers les deux commandes suivantes :

Sudo usermod -a -G www-data pi

Sudo chown -R -f www-data /var/www/html

- Mise en place de PHP :

PHP est un des langages les plus utilisés pour créer des sites dynamiques. La manière la plus classique et la plus simple d'installer PHP est de l'installer sous forme d'un module Apache. Pour cela, il suffit d'installer le module libapache2-mod-php :

Sudo apt install php7.3 libapache2-mod-php7.3 php7.3-mbstring php7.3-mysql php7.3-curl php7.3-gd php7.3-zip -y

Maintenant après l'exécution de cette commande, PHP est installé sur notre Raspberry PI

- Mise en place de MySQL :

MySQL est l'un des systèmes de gestion base de données relationnelles les plus populaires au monde qui permet de stocker et de gérer facilement de grandes quantités de données. C'est l'une des technologies qui aident à piloter le WEB moderne.

Avant de commencer à installer MySQL sur notre Raspberry Pi, nous devons d'abord mettre à jour notre liste de packages et tous les packages installés. En exécutant les deux commandes suivantes :

Sudo apt-get update

Sudo apt-get upgrade

L'étape suivante consiste à installer le logiciel du serveur MySQL sur notre Raspberry Pi en effectuant la commande suivante : **sudo apt install mariadb-server**

Nous devons maintenant lancer le processus de sécurisation MySQL :

Sudo mysql_secure_installation

- Mise en place de PHPMYADMIN :

PHPMYADMIN est un outil qui a été conçu pour permettre une administration facile de MySQL.

Pour installer le package PHPMYADMIN sur notre Raspberry Pi, nous devons exécuter la commande suivante : **sudo apt install phpmyadmin**

Une fois le processus d'installation de PHPMYADMIN terminé, il reste à créer un nouvel utilisateur afin d'accéder à des tables de données dans PHPMYADMIN. Pour ce, nous devons nous connecter à l'interface de ligne de commande MySQL en utilisant l'utilisateur « root » : **sudo mysql -u root -p**

Puis on exécute la commande suivante : **GRANT ALL PRIVILEGES ON *.* TO 'username'@'localhost' IDENTIFIED BY 'password' WITH GRANT OPTION ;**

→ Configuration d'Apache pour PHPMYADMIN :

Avant de pouvoir charger l'interface PHPMYADMIN sur notre Raspberry Pi, nous devons apporter quelques modifications à la configuration d'Apache.

Pour commencer, nous devons éditer le fichier « Apache2.conf » :

sudo nano /etc/apache2/apache2.conf

Puis nous devons ajouter la ligne suivante au bas de ce fichier :

Include /etc/phpmyadmin/apache.conf

Et finalement redémarrer le service Apache sur notre Raspberry Pi :

sudo service apache2 restart

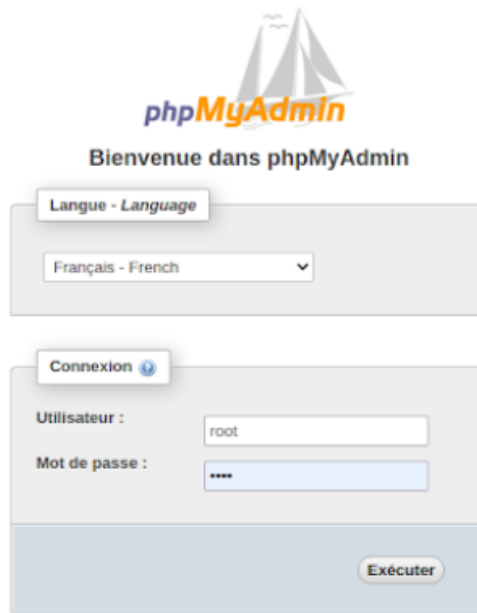


Figure 9 : Login PhpMyAdmin

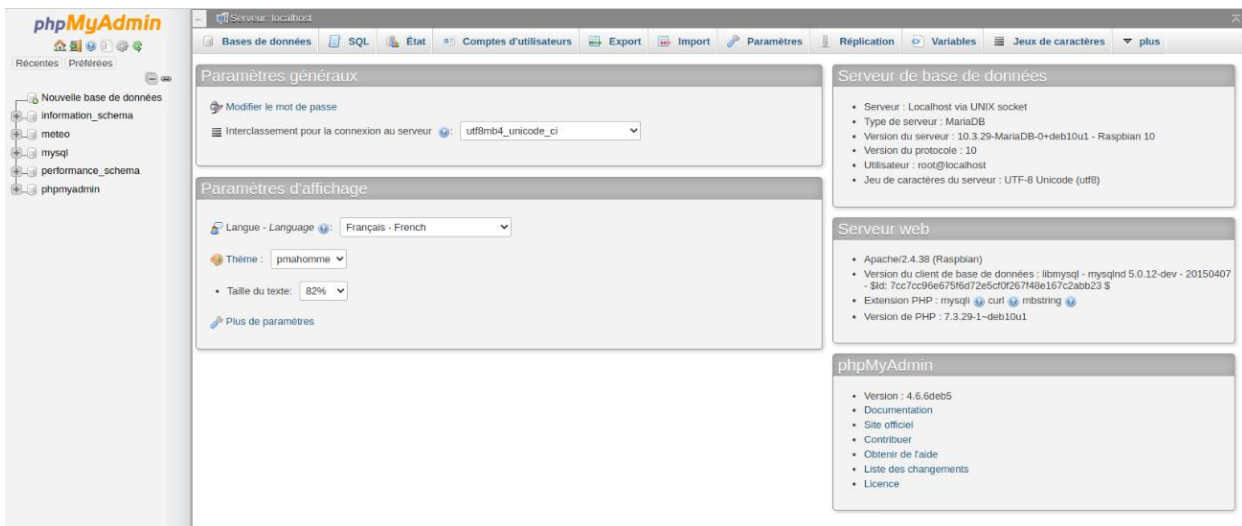


Figure 10 : Interface PhpMyAdmin

- Installation de Python :

Python est un langage de programmation interprété, multi-paradigme et multiplateformes. Il favorise la programmation impérative structurée, fonctionnelle et orientée objet. Pour installer Python il faut exécuter les commandes suivantes :

sudo apt-get update

sudo apt-get install python3-pip apache2 libapache2-mod-wsgi-py3

→ Configurer un environnement virtuel Python :

L'environnement virtuel Python est un environnement qui permet de séparer un projet des outils du système et de tout autre projet Python sur lequel nous pourrions travailler.

Nous devons installer la virtualenv commande pour créer ces environnements . Nous pouvons obtenir ceci en utilisant pip : **sudo pip3 install virtualenv**

2. Création d'un projet Django :

- Configuration de l'environnement virtuel :

Une fois virtualenv est installé , nous pouvons commencer à former notre projet .Pour commencer on doit crée un repertoire dans lequel on souhaite conserver notre projet :

```
pi@raspberrypi:~/Desktop $ mkdir meteo
```

Puis on se deplace dans le repertoire afin de crée un environnement virtuel Python :

```
pi@raspberrypi:~/Desktop $ cd meteo
```

```
pi@raspberrypi:~/Desktop/meteo $ virtualenv myenv
```

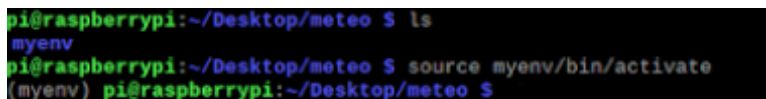


```
pi@raspberrypi:~/Desktop $ cd meteo
pi@raspberrypi:~/Desktop/meteo $ ls
myenv
```

Figure 11 : Création myenv

Il nous reste qu'installer Django dans notre environnement , mais avant tout il faut activer l'environnement virtuel en executant :

```
pi@raspberrypi:~/Desktop/meteo $ source myenv/bin/activate
```



```
pi@raspberrypi:~/Desktop/meteo $ ls
myenv
pi@raspberrypi:~/Desktop/meteo $ source myenv/bin/activate
(myenv) pi@raspberrypi:~/Desktop/meteo $
```

Figure 12 : Activation myenv

- Installation de Django et Configuration du projet :

Après la configuration de l'environnement virtuel de notre projet il nous reste qu'installer Django :

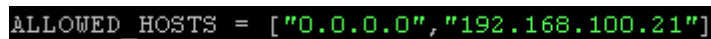
```
(myenv) pi@raspberrypi:~/Desktop/meteo $ pip3 install django
```

Maintenant tout est prêt , il nous faut que crée notre projet Django avec la commande suivante :

```
(myenv) pi@raspberrypi:~/Desktop/meteo $ django-admin.py startproject meteo
```

Après la creation de notre projet Django , on ajuste les parametres dans le fichier settings.py :

- On commence par saisir les adresses autorisées dans la ligne 29 du fichier :



```
ALLOWED_HOSTS = [ "0.0.0.0", "192.168.100.21" ]
```

Figure 13 : Les adresses autorisées

- En bas du fichier , on ajoute une ligne pour la configuration du repertoire static de notre projet sans oublier « import os » en haut du fichier :

```
# Static files (CSS, JavaScript, Images)
# https://docs.djangoproject.com/en/3.2/howto/static-files/

STATIC_URL = '/static/'
STATICFILES_DIRS = (
    os.path.join(BASE_DIR, 'static'),
)
```

Figure 14 : Configuration static

Partie 2: Partie réalisation et commandes

1. Création de l'application MeteoApp :

Maintenant apres la configuration de notre environnement virtuel et de notre projet Django il nous reste la creation de notre application MeteoApp pour cela on execute la commande suivante :

(myenv) **pi@raspberrypi:~/Desktop/meteo/meteo \$ python3 manage.py startapp MeteoApp**

Sans oubliant la creation des deux repertoires static et templates .

```
pi@raspberrypi:~/Desktop/meteo/meteo $ ls
db.sqlite3  manage.py  meteo  MeteoApp  static  templates
```

Figure 15 : Création MeteoApp et ajout de static et templates

2. Création de la bdd meteo (models+migration) :

Afin de stocker les valeurs récupères des capteurs on doit crée un base de données pour notre projet :

- Schema conceptuel de notre bdd :

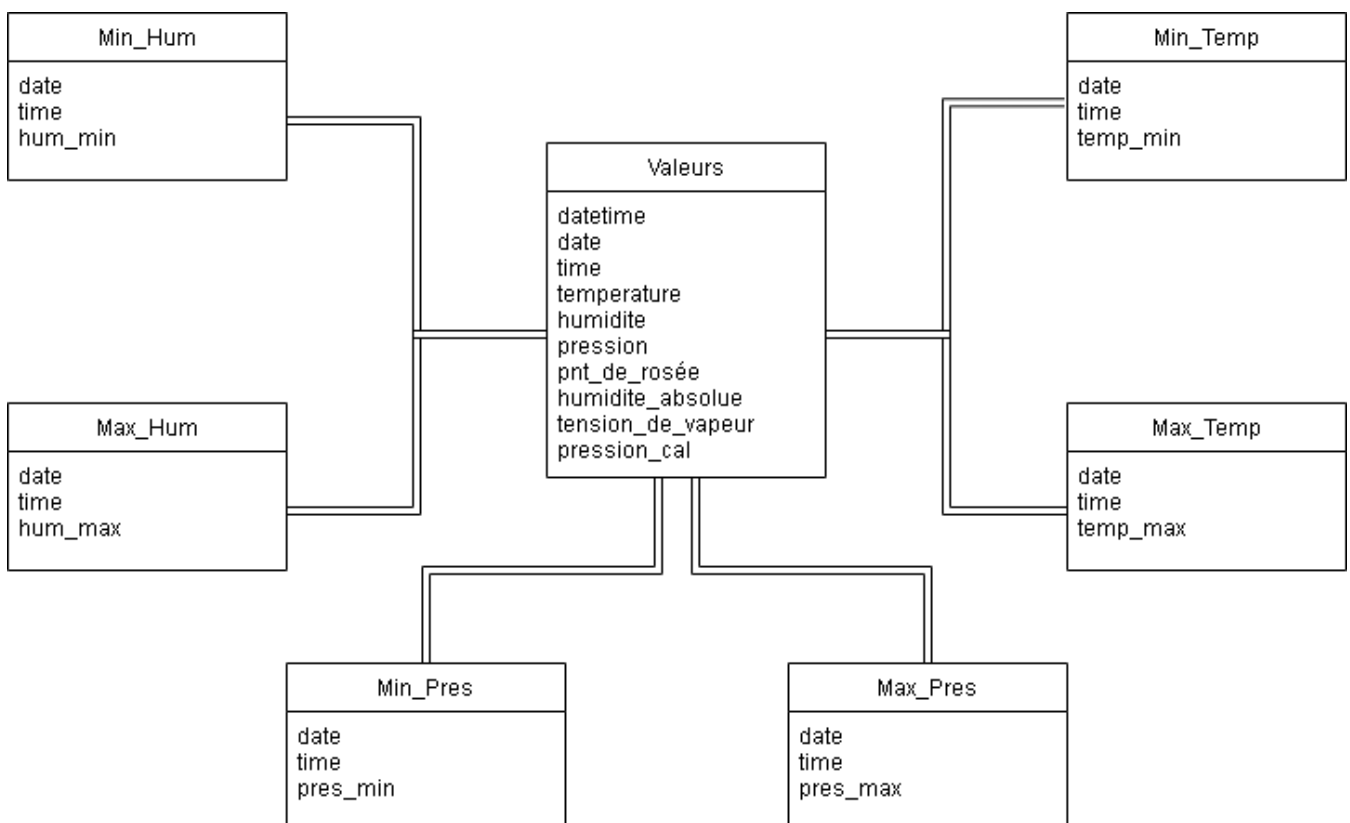


Figure 16 : Diagramme de classe du projet

- Django CRUD :

Une fois le projet créé, on modifie les paramètres dans 'settings.py'. Django est configuré pour utiliser SQLite par défaut, mais comme on utilise MySQL, on doit configurer le type de base de données.

```
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.mysql',
        'NAME': 'meteo',
        'USER': 'root',
        'PASSWORD': 'root',
        'HOST': 'localhost',
        'PORT': '3306'
    }
}
```

Figure 17 : Connexion base de données

- Models :

Voici les classes utiliser pour le projet :

- Valeurs :

```
class Valeurs(models.Model):
    datetime=models.DateTimeField(null=True)
    date=models.DateField(null=True)
    time=models.TimeField(null=True)
    temperature=models.FloatField()
    humidite=models.FloatField()
    pression=models.FloatField()
    pnt_de_rosée=models.FloatField()
    humidite_absolue=models.FloatField()
    tension_de_vapeur=models.FloatField()
    pression_cal=models.FloatField()

    def __str__(self):
        return self.datetime
```

Figure 18 : class Valeurs

- Min_Temp :

```
class Min_Temp(models.Model):
    date=models.DateField(null=True)
    time=models.TimeField(null=True)
    temp_min=models.FloatField(null=True)

    def __str__(self):
        return self.date
```

Figure 19 : Class Min_Temp

- Max_Temp :

```
class Max_Temp(models.Model):
    date=models.DateField(null=True)
    time=models.TimeField(null=True)
    temp_max=models.FloatField(null=True)

    def __str__(self):
        return self.date
```

Figure 20 : Class Max_Temp

- Min_Hum :

```
class Min_Hum(models.Model):
    date=models.DateField(null=True)
    time=models.TimeField(null=True)
    hum_min=models.FloatField(null=True)

    def __str__(self):
        return self.date
```

Figure 21 : Class Min_Hum

- Max_Hum :

```
class Max_Hum(models.Model):
    date=models.DateField(null=True)
    time=models.TimeField(null=True)
    hum_max=models.FloatField(null=True)

    def __str__(self):
        return self.date
```

Figure 22 : Class Max_Hum

- Min_Pres :

```
class Min_Pres(models.Model):
    date=models.DateField(null=True)
    time=models.TimeField(null=True)
    pres_min=models.FloatField(null=True)

    def __str__(self):
        return self.date
```

Figure 23 : Class Min_Pres

- Max_Pres :

```
class Max_Pres(models.Model):
    date=models.DateField(null=True)
    time=models.TimeField(null=True)
    pres_max=models.FloatField(null=True)

    def __str__(self):
        return self.date
```

Figure 24 : Class Max_Pres

Et en utilisant la migration, on obtient les tables dans notre base de données en exécutant la commande suivante : **(python3 manage.py makemigrations & python3 manage.py migrate)**.

Table	Action	Lignes	Type	Interclassement	Taille	Perte
auth_group	Afficher Structure Rechercher Insérer Vider Supprimer	0	InnoDB	utf8mb4_general_ci	32 Kio	-
auth_group_permissions	Afficher Structure Rechercher Insérer Vider Supprimer	0	InnoDB	utf8mb4_general_ci	48 Kio	-
auth_permission	Afficher Structure Rechercher Insérer Vider Supprimer	52	InnoDB	utf8mb4_general_ci	32 Kio	-
auth_user	Afficher Structure Rechercher Insérer Vider Supprimer	0	InnoDB	utf8mb4_general_ci	48 Kio	-
auth_user_groups	Afficher Structure Rechercher Insérer Vider Supprimer	0	InnoDB	utf8mb4_general_ci	48 Kio	-
auth_user_permissions	Afficher Structure Rechercher Insérer Vider Supprimer	0	InnoDB	utf8mb4_general_ci	48 Kio	-
django_admin_log	Afficher Structure Rechercher Insérer Vider Supprimer	0	InnoDB	utf8mb4_general_ci	48 Kio	-
django_content_type	Afficher Structure Rechercher Insérer Vider Supprimer	13	InnoDB	utf8mb4_general_ci	32 Kio	-
django_migrations	Afficher Structure Rechercher Insérer Vider Supprimer	19	InnoDB	utf8mb4_general_ci	16 Kio	-
django_session	Afficher Structure Rechercher Insérer Vider Supprimer	0	InnoDB	utf8mb4_general_ci	32 Kio	-
MeteoApp_max_hum	Afficher Structure Rechercher Insérer Vider Supprimer	40	InnoDB	utf8mb4_general_ci	16 Kio	-
MeteoApp_max_pres	Afficher Structure Rechercher Insérer Vider Supprimer	50	InnoDB	utf8mb4_general_ci	16 Kio	-
MeteoApp_max_temp	Afficher Structure Rechercher Insérer Vider Supprimer	25	InnoDB	utf8mb4_general_ci	16 Kio	-
MeteoApp_min_hum	Afficher Structure Rechercher Insérer Vider Supprimer	40	InnoDB	utf8mb4_general_ci	16 Kio	-
MeteoApp_min_pres	Afficher Structure Rechercher Insérer Vider Supprimer	40	InnoDB	utf8mb4_general_ci	16 Kio	-
MeteoApp_min_temp	Afficher Structure Rechercher Insérer Vider Supprimer	40	InnoDB	utf8mb4_general_ci	16 Kio	-
MeteoApp_valeurs	Afficher Structure Rechercher Insérer Vider Supprimer	49	InnoDB	utf8mb4_general_ci	16 Kio	-
17 tables	Somme	391	InnoDB	utf8mb4_general_ci	488 Kio	0 o

Figure 25 : Migration de models

3. Script du capteur DHT22 :

- Récupération des valeurs :

On a deux types de valeurs, valeurs mesurer et calculer ces dernier on les recuperes a traver une boucle dans un script python ou on import les bibliotheques des capteurs bme280 et Adafruit_DHT du capteur dht22 , la bibliotheque datetime afin d'insérer la date et temps actuelle des valeurs mesurées sans oublier aussi la bibliotheque math qu'on utilisera lors du calcul des valeurs .

```
try:
    while 1:
        #boucle infinie de la lecture des capteurs avec enregistrement sur bdd
        #temps = time.strftime("%d-%m-%Y %H:%M:%S")
        datetime=time.strftime("%Y-%m-%d")
        tim=time.strftime('%H:%M:%S')
        with open('/sys/class/thermal/thermal_zone0/temp', 'r') as ftemp:
            CPU_temp = int(ftemp.read()) / 1000
            #bloc :lecture des capteurs
            tempe,pres,humid = bme280.readBME280All()
            humi, temp = dht.read_retry(dht.DHT22, 4)
            humid=round(humi,2)
            temperature = round(temp,2)
            pression = round(pres,2)
            #bloc: calcul du point de rosée
            td = round((243.04*(log(humi/100)+((17.625*temp)/(243.04+temp)))/(17.625-log(humi/100)-((17.625*temp)/(243.04+temp))),2)
            #bloc: calcul de la Pression au niveau de la mer (QNH)
            QNH = round((1013.25*math.pow(math.pow((pres/1013.25), -(287.053*-0.0065)/9.80665) - (56*-0.0065)/288.15, -(9.80665/(287.053*-0.0065)))-1.29 ,2)
            #Script valeurs
            #bloc: calcul du de la Tension de vapeur & humidité absolue
            ai=7.45
            bi=235
            z1=(ai*temp)/(bi+temp)
            es=6.1*exp(z1*2.3025851)
            e=es*humid/100
            z2=e/6.1
            z3=0.434292289*log(z2)
            dru=e*100
            tau=(235+z3)/(7.45-z3)*100
            tau=floor(tau)/100
            feu=(216.7*e)/(273.15+temp)*100
            humabs=round(feu)/100
            presvap=floor(dru)/100
```

Valeurs mesurées :
Température, humidité et pression

Valeurs calculées :
Pression Nv Mer, tension de vapeur
et humidité absolue

Figure 26 : Script DHT22

- Insertion dans la bdd :

L'insertion des valeurs mesurées et calculées se fait aussi a traver le meme script python , premierement on doit importer **mysql connector** afin de lier et connecter notre base de donnée a notre script puis on

implémente une fonction **insertIntoDatabase** dans la quelle on se connecte a notre base de donnée et on prepare notre sql requette qui s'exutera lors de l'execution de la fonction **insertIntoDatabase**

```
def insertIntoDatabase(zeit,date,time,temperature,humidi,pression,td,humabs,presvap,QNH):
    try:
        db = mysql.connector.connect(host="localhost",user="root",password="root",database="meteo")
        cur = db.cursor()
        mySql_insert_query = """INSERT INTO MeteoApp_valeurs(datetime,date,time,temperature,humidite,pression,pnt_de_rosée,humidite_absolue,tension_de_vapeur,
        pression_cal) VALUES(%s,%s,%s,%s,%s,%s,%s,%s,%s,%s)"""
        record = (zeit,date,time,temperature,humidi,pression,td,humabs,presvap,QNH)
        cur.execute(mySql_insert_query, record)
        db.commit()
        print("Record inserted successfully into MeteoApp_valeurs table")
    except mysql.connector.Error as error:
        print("Failed to insert into MySQL table {}".format(error))
```

Figure 27 : Fonction insertIntoDatabase

- Sélection et insertion des maxs et mins :

Pour la sélection et l'insertion elle se fait aussi a travers les fonctions max et min dans notre script , dans ces fonctions on se connecte a notre base de donnée et on prepare notre sql requette qui s'exutera lors de l'execution des fonctions .

Exemple fonction max_temp() et min_temp() :

```
def max_temp():
    db = mysql.connector.connect(host="localhost",user="root",password="root",database="meteo")
    cur = db.cursor()
    mySql_max_temp_query = """INSERT INTO MeteoApp_max_temp(time,temp_max) SELECT time,MAX(temperature) from MeteoApp_valeurs where ( date = CURRENT_DATE() and
    time >= '06:00:00' ) or ( date = CURRENT_DATE()+1 and time <= '06:00:00' );"""
    cur.execute(mySql_max_temp_query)
    db.commit()

def min_temp():
    db = mysql.connector.connect(host="localhost",user="root",password="root",database="meteo")
    cur = db.cursor()
    mySql_min_temp_query = """INSERT INTO MeteoApp_min_temp(date,time,temp_min) SELECT date,time,MIN(temperature) from MeteoApp_valeurs where
    ( date = CURRENT_DATE()-1 and time >= '18:00:00' ) or ( date = CURRENT_DATE() and time <= '06:00:00' ); """
    cur.execute(mySql_min_temp_query)
    db.commit()
```

Figure 28 : Fonctions max_temp et min_temp

Meme structure pour les fonctionn max_hum , min_hum , max_pres et min_pres .

L'execution de se script se fera automatiquement par la commande **crontab -e** :

```
# Notice that tasks will be started based on the cron's system
# daemon's notion of time and timezones.
#
# Output of the crontab jobs(including errors) is sent through
# email to the user the crontab file belongs to (unless redirected).
#
# For example, you can run a backup of all your user accounts
# at 5 a.m every week with:
# 0 5 * * 1 tar -zcf /var/backups/home.tgz /home/
#
# For more information see the manual pages of crontab(5) and cron(8)
#
# m h dom mon dow   command
0 0 * * * /usr/bin/python3 /home/pi/Desktop/meteo/meteo/MeteoApp/DHT22.py
```

Figure 29 : Action de l'exécution du script

4. Urls et Views :

- Ajout des views :
 - valeurs-page :

```
def valeurs_page(request):  
    last_values=Valeurs.objects.last()  
    last_value_max_temp=Max_Temp.objects.last()  
    last_value_min_temp=Min_Temp.objects.last()  
    last_value_max_hum=Max_Hum.objects.last()  
    last_value_min_hum=Min_Hum.objects.last()  
    last_value_max_pres=Max_Pres.objects.last()  
    last_value_min_pres=Min_Pres.objects.last()  
  
    context={  
        'last_values':last_values,  
        'last_value_max_temp':last_value_max_temp,  
        'last_value_min_temp':last_value_min_temp,  
        'last_value_max_hum':last_value_max_hum,  
        'last_value_min_hum':last_value_min_hum,  
        'last_value_max_pres':last_value_max_pres,  
        'last_value_min_pres':last_value_min_pres  
    }  
    return render(request,"valeurs.html",context)
```

Figure 30 : View fonction valeurs_page

Dans cette fonction on récupère la dernière ligne de notre base de données meteo (last_values), dernières valeurs max et min de temperature , humidite et pression afin de les afficher dans notre template « valeurs.html ».

- view d' historiques :

Pour les pages d'historiques , on récupère la dernière valeur instantanée (last_val) , max et min de chaque historique , toutes les valeurs de la base de donée (date,time et la valeur de la page) order by « - id » , un filtre pour filtrer les valeurs et labels et data pour notre chart :

Exemple temperature_page (meme structure pour les autres pages de l'historique) :

```
def temperature_page(request):
    last_temp=Valeurs.objects.last()
    last_temp_max=Max_Temp.objects.last()
    last_temp_min=Min_Temp.objects.last()
    temp=Valeurs.objects.all().order_by('-id')

    myFilter=TempFilter(request.GET, queryset=temp)
    temp = myFilter.qs

    labels = []
    data = []
    queryset = Valeurs.objects.order_by('-date')
    for val in queryset:
        labels.append(val.datetime.strftime("%Y-%m-%d %H:%M"))
        data.append(val.temperature)
    context={
        'temp':temp,
        'last_temp':last_temp,
        'labels':labels,
        'data':data,
        'last_temp_max':last_temp_max,
        'last_temp_min':last_temp_min,
        'myFilter':myFilter
    }
    return render(request,"temperature.html",context)
```

- view message : **Figure 31 : View fonction temperature_page**

Pour les pages de messages , on récupère seulement la dernière ligne des valeurs instantanées de notre base de donnée meteo (last_values).

Exemple metar_page (meme structure pour synop_page) :

```
def metar_page(request):
    last_values=Valeurs.objects.last()
    context={
        'last_values':last_values
    }
    return render(request,"metar.html",context)
```

Figure 32 : View fonction metar_page

- Liaison des views avec leurs url :

```
from django.urls import path
from . import views

urlpatterns = [
    path('', views.valeurs_page, name="valeurs"),
    path('temperature/', views.temperature_page, name="temperature"),
    path('humidite/', views.humidite_page, name="humidite"),
    path('pression/', views.pression_page, name="pression"),
    path('point_de_rosée/', views.point_de_rose_page, name="point_de_rose"),
    path('humidite_absolue/', views.humidite_absolue_page, name="humidite_absolue"),
    path('tension_de_vapeur/', views.tension_de_vapeur_page, name="tension_de_vapeur"),
    path('pression_nv_mer/', views.pression_nv_mer_page, name="pression_nv_mer"),
    path('metar/', views.metar_page, name="Metar"),
    path('synop/', views.synop_page, name="Synop"),
    path('speci/', views.speci_page, name="Speci"),
]
```

Figure 33 : Urls

Partie 3 : Interfaces

1. Valeurs Instantanées :

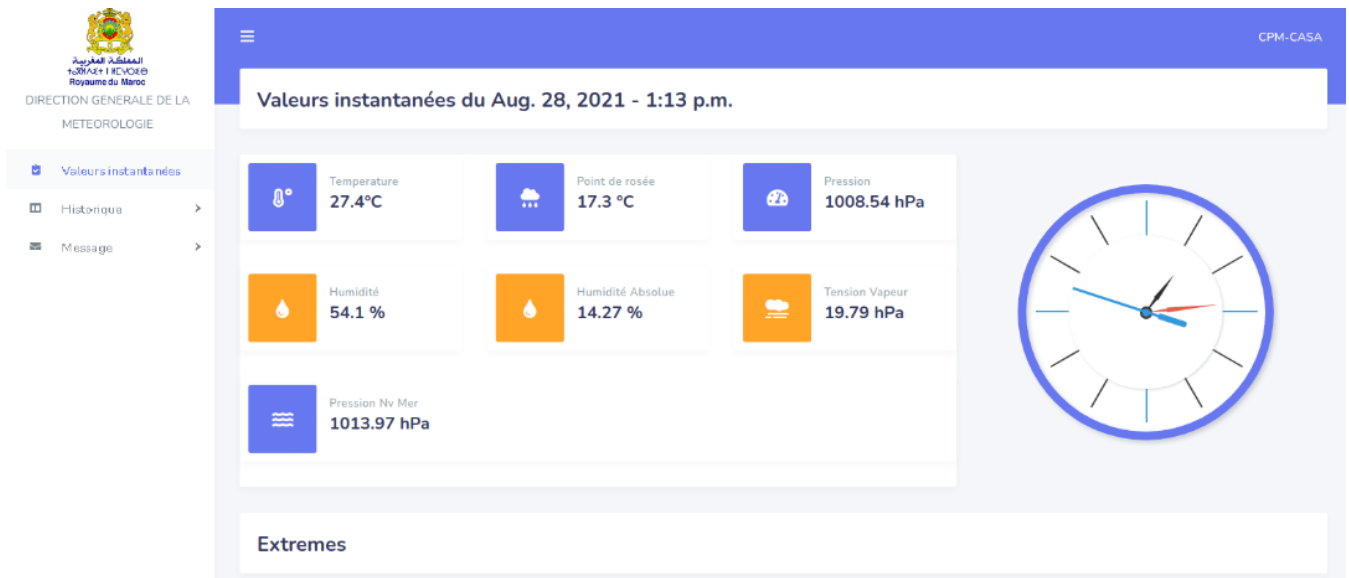


Figure 34 : Interface Valeurs Instantanées 1

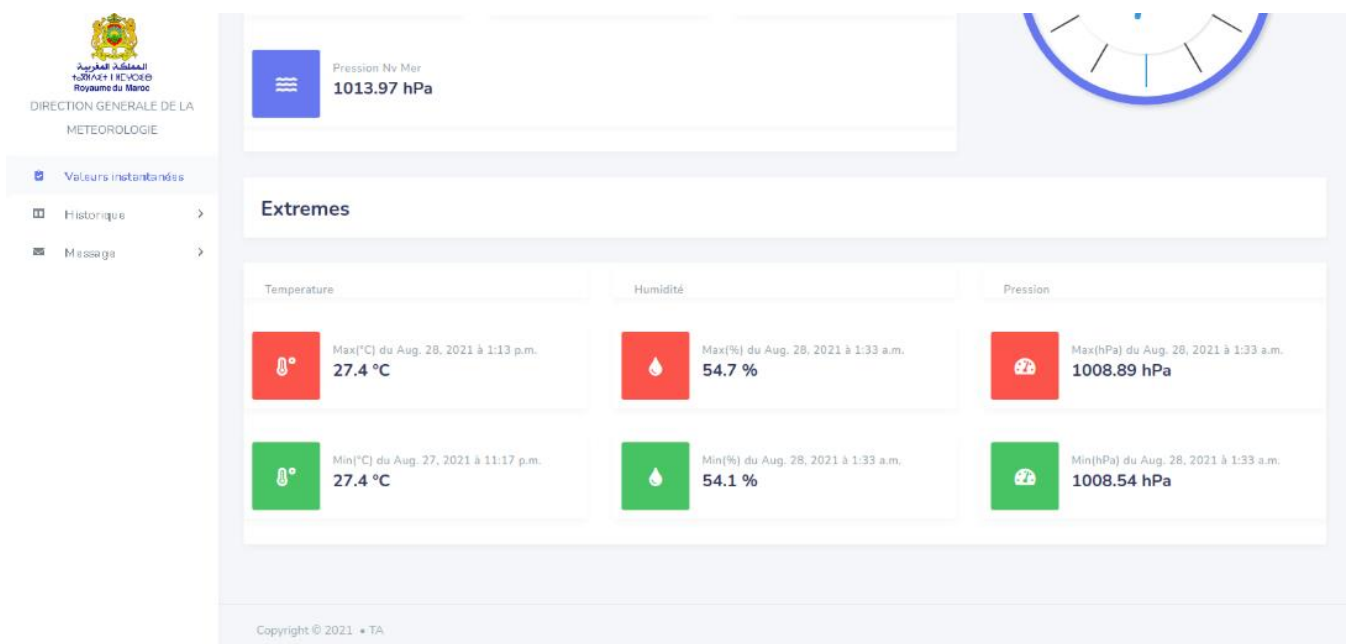


Figure 35 : Interface Valeurs Instantanées 2

La page d'accueil (valeurs instantanées) de notre application web se compose de :

- Une nav bar a gauche qui permet a l'utilisateur de passer d'une page a une autre par un simple clique
- L'entete de la page (Valeurs instantanées de +date +time de la recuperation des valeurs)
- Un div qui contient l'affichage des valeurs instantanées
- Un div qui contient une horloge
- Un div qui contient l'affichage des valeurs mins et maxs

- Footer

2. Historique : « Exemple température , même structure pour les autres pages de l'historique »

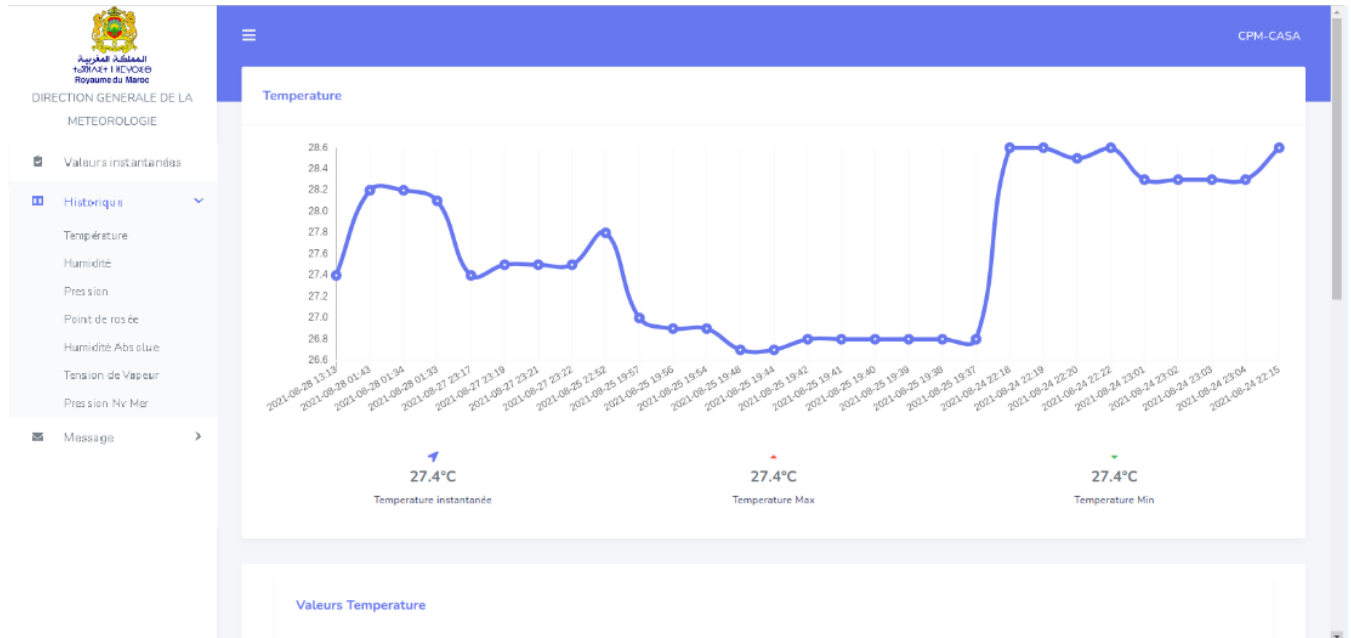


Figure 36 : Interface Température 1

Figure 37 displays the 'Interface Température 2' showing a table of temperature history data. The table lists temperature values (Y-axis, ranging from 26.6 to 28.6) against time (X-axis, showing dates from 2021-08-28 13:19 to 2021-08-24 22:15). The table shows a significant peak in temperature around 2021-08-24 22:15, reaching approximately 28.5°C. Below the graph, three summary values are displayed: 27.4°C for 'Temperature instantanée', 27.4°C for 'Temperature Max', and 27.4°C for 'Temperature Min'. The interface includes a sidebar with navigation options like 'Valeurs instantanées', 'Historique', and 'Message'.

Date	Time	Temperature(°C)
Aug. 28, 2021	1:13 p.m.	27.4
Aug. 28, 2021	1:43 a.m.	28.2
Aug. 28, 2021	1:34 a.m.	28.2
Aug. 28, 2021	1:33 a.m.	28.1
Aug. 27, 2021	11:22 p.m.	27.5
Aug. 27, 2021	11:21 p.m.	27.5
Aug. 27, 2021	11:19 p.m.	27.5
Aug. 27, 2021	11:17 p.m.	27.4

Figure 37 : Interface Température 2

Les pages d'historiques de notre application web visualise les anciennes valeurs en fonction du temps et date . C'est pages ce compose de :

- Graffe d'evolution des valeurs en fonction de datetime

- Filtre des valeurs du tableau en fonction de la date
- Tableau des anciennes valeurs

3. Message : « Example metar », même structure pour les autres pages de l'historique »

The screenshot shows the 'Message Metar' interface. On the left is a sidebar with the logo of the Direction Générale de la Météorologie du Maroc and a menu with options: Valeurs instantanées, Historique, Message (selected), Météar, Synop, and Speci. The main content area has a blue header with 'CPM-CASA'. Below the header, there's a 'Message Metar' section with a green 'Données' button. A large text area is empty. At the bottom, there are input fields for 'IP', 'Login', and 'Mot de passe', followed by 'Envoyer' and 'Annuler' buttons. The footer says 'Copyright © 2021 • TA'.

Figure 38 : Interface Metar 1

This screenshot shows the 'Message Metar' interface with a sample METAR message displayed in the text area. The message is: 'SAMC45 GMMC Aug. 31, 2021 12:16 p.m. 00 METAR GMMC Aug. 31, 2021 12:16 p.m. 00Z... 26.5/17.92 Q1014.26 NOSIG ='. The rest of the interface, including the sidebar, header, and footer, is identical to Figure 38.

Figure 39 : Interface Metar 2

Les pages messages de notre application web permettent d'envoyer un message spécifique par FTP , ce message est recupere en cliquant sur le button donnée qui declenche en fonction par methode onclick cette dernière remplit l'innerText de la page par les données recuperer de notre base de donnée

Partie 4 : Partie Logiciel



Visual Studio Code est un éditeur de code source qui peut être utilisé avec une variété de langages de programmation, notamment Java, JavaScript, Go, Node.js et C++



Python est un langage de programmation interprété, multi paradigme et multiplateformes. Il favorise la programmation impérative structurée, fonctionnelle et orientée objet.



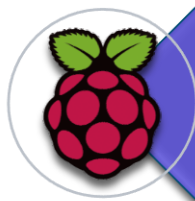
Django est un cadre de développement web open source en Python. Il a pour but de rendre le développement web 2.0 simple et rapide.



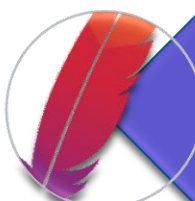
PhpMyAdmin (PMA) est une application Web de gestion pour les systèmes de gestion de base de données MySQL et MariaDB, réalisée principalement en PHP



PuTTY est un émulateur de terminal doublé d'un client pour les protocoles SSH, Telnet, rlogin, et TCP brut



Raspberry Pi OS (anciennement nommé **Raspbian**¹) est un système d'exploitation libre et gratuit basé sur Debian optimisé pour fonctionner sur les différents Raspberry Pi.



Apache HTTP Server (**Apache**) est un serveur HTTP créé et maintenu au sein de la fondation Apache

Conclusion générale :

Le stage d'observation au CPM de Casablanca, était une expérience incontournable et très utile pour ma carrière en tant qu'élèves ingénieurs en informatique.

En fait, ce stage a servi de champ d'application et de mise en œuvre des connaissances théoriques apprises lors de nos trois premières années à

L'École Marocaine Des Sciences De L'Ingénieur tout en étant confronté aux difficultés réelles du monde du travail.

Ce stage a été très enrichissant, car il m'a permis de découvrir en premier lieu le travail, d'acquérir l'esprit du travail en équipe et de développer nos compétences d'expression et de communication.

Finalement, ce stage n'est qu'une initiation pour affranchir et relever de nouveaux défis dans le monde professionnel.

Je tiens une autre fois à exprimer mes profondes gratitude à l'ensemble du personnel de CPM Casa Anfa.

Bibliographie :

[1] : <https://www.instructables.com/Raspberry-Pi-Tutorial-How-to-Use-the-DHT-22/>

[2] : <https://www.raspberrypi-spy.co.uk/2016/07/using-bme280-i2c-temperature-pressure-sensor- in-python/>