



Ensemble Scolaire
Pradeau-La Sède



Lycée Pradeau la Sède - St Pierre
24 Avenue d'Azereix,
65000 Tarbes, France

Projet WebTV

Alamar Valérian - Dossier Personnel
Acquisitions
2020-2021



N'PY
3 bis Avenue Jean Prat
65100 Lourdes

Sommaire

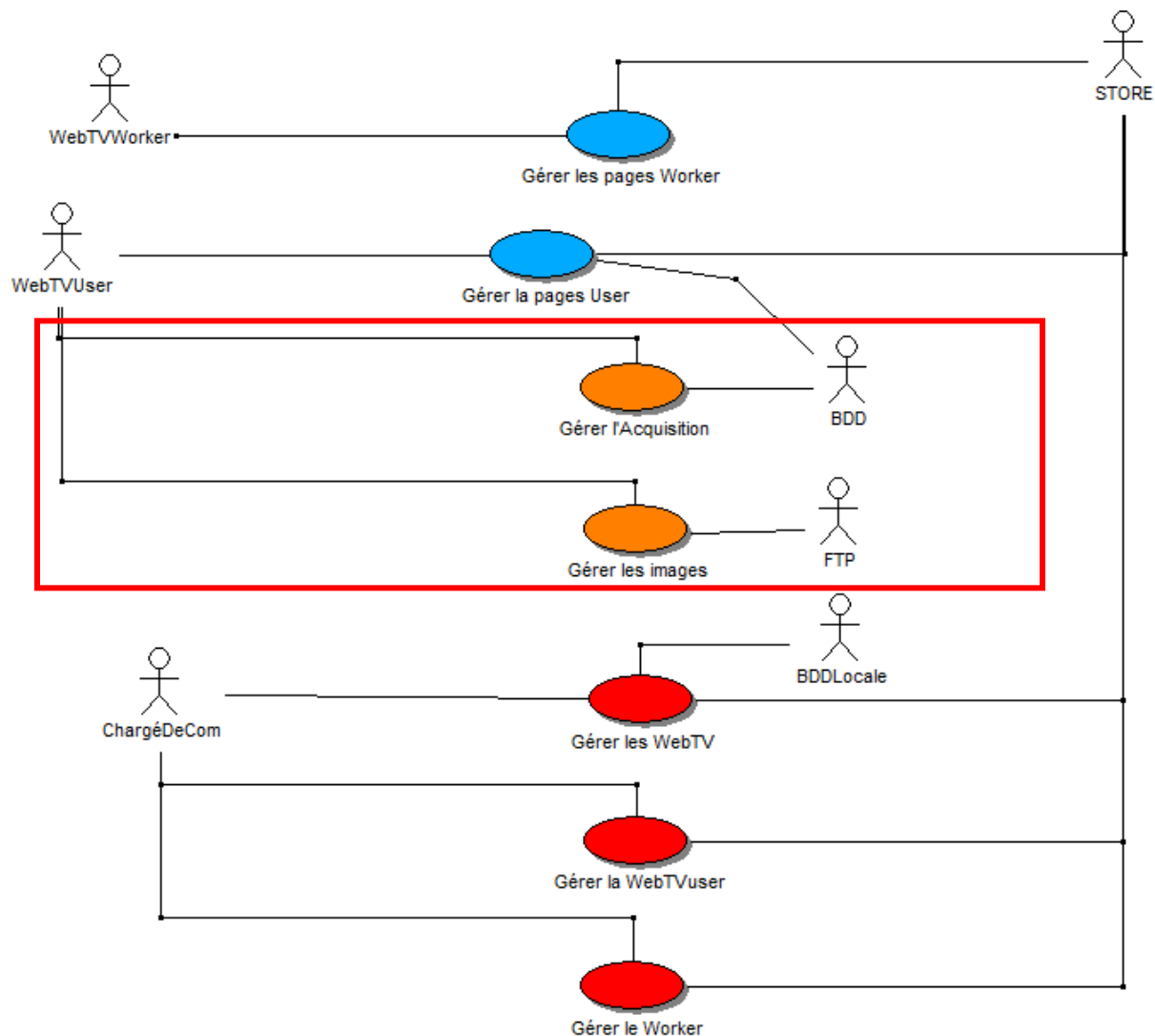
I – INTRODUCTION	3
1- L’OBJECTIF DE LA PARTIE ACQUISITIONS	3
2- DIAGRAMME DE DEPLOIEMENT	4
3- DIAGRAMME DES CAS D’UTILISATION	5
4- DIGRAMME DE CLASSES	5
5- QU’EST-CE QU’UNE RASPBERRY PI ?	6
6- PLANIFICATION DU TRAVAIL	6
II - LA BASE DE DONNEES	8
1- PRESENTATION DE LAMP	8
2- PRESENTATION DE LA BASE DE DONNEES	8
3- ENVOI DES DONNEES VERS LA BDD	9
III – LE SERVEUR FTP	10
1- PRESENTATION DU PROTOCOLE FTP	10
2- RECUPERATION DES IMAGES SUR LA RASPBERRY	10
IV – ACQUISITIONS	12
1- CAPTEUR DE TEMPERATURE	12
1. FONCTIONNEMENT DU CAPTEUR	12
2. MONTAGE DU CAPTEUR	12
3. CODE D’ACQUISITION DE LA TEMPERATURE	13
2- CAPTEUR D’HUMIDITE	14
1. FONCTIONNEMENT DU CAPTEUR	14
2. MONTAGE DU CAPTEUR	16
3- PROBLEMES RENCONTRES	17
4- LA TACHE CRON	17
5- EXECUTABLE ACQUISITIONS	18
V - MANUEL D’INSTALLATION ACQUISITIONS	19
I – MATERIEL ET LOGICIELS REQUIS	19
II – CONFIGURATION DE LA RASPBERRY	19
II – INSTALLATION DES LOGICIELS ET LIBRAIRIES	20
1. INSTALLATION DE LAMP	20
2. INSTALLATION DES LIBRAIRIES	21
3. INSTALLATION DE LA BASE DE DONNEES	21
VI : CONCLUSION	22
VII : ANNEXES	23
1- DOCUMENTATION TECHNIQUE	23
1- FICHES DE TEST UNITAIRE	26

I – Introduction

1- L'objectif de la partie Acquisitions

Le client, NP'Y, souhaite installer dans la station Cauterets des bornes d'information appelées WebTV qui sont des systèmes embarqués composés d'une raspberry PI 3 équipée d'un écran et de différents capteurs, un de température, un d'humidité et un anémomètre ainsi qu'une base de données intégrée sous LAMP.

Le but de ces WebTV est de permettre aux clients de la station de ski d'avoir des informations en temps réel sur la température, l'humidité et la vitesse du vent avec la gestion des acquisitions ainsi que des images issues de caméras de la station avec la gestion des images à l'aide d'un site Web embarqué qui va afficher toutes les données sur l'écran de la WebTV.



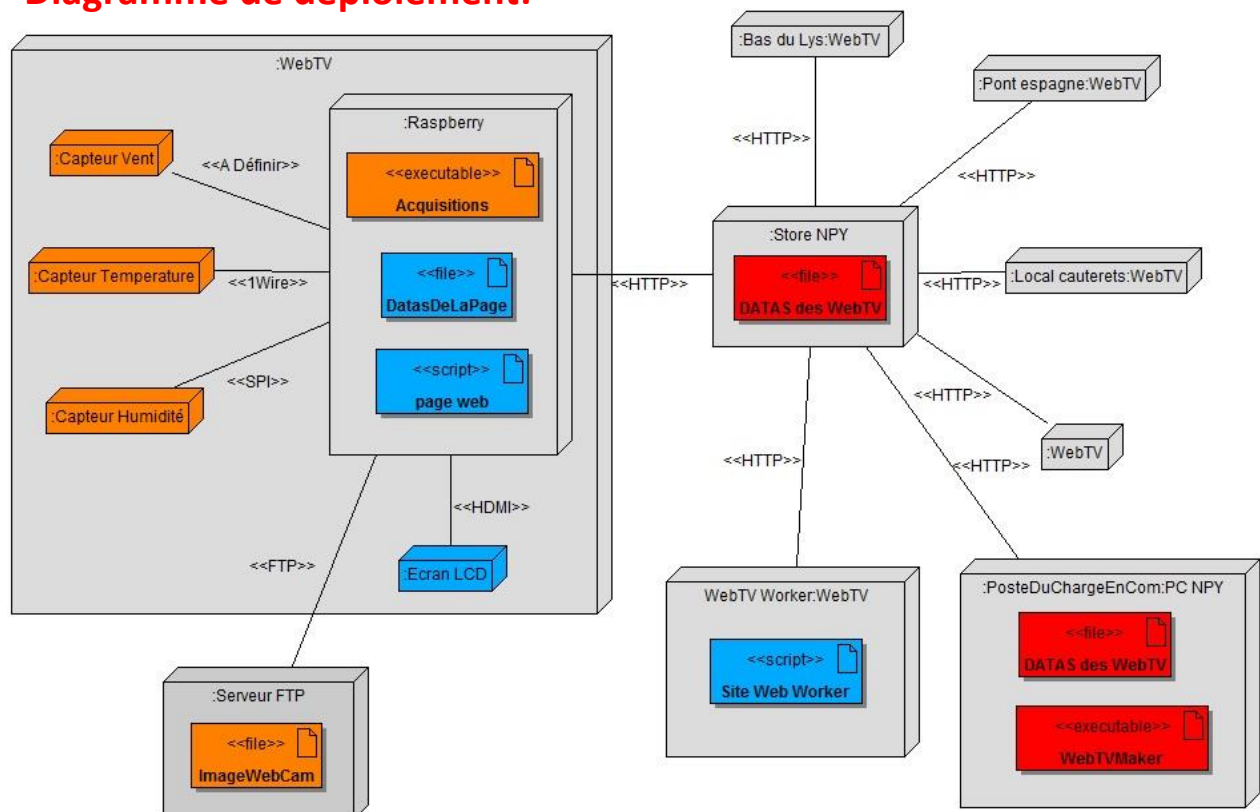
2- Diagramme de déploiement

Afin de situer ma partie dans le projet, je vais utiliser un diagramme de déploiement qui permet de montrer l'architecture matérielle d'un système et le déploiement des composants logiciels sur ces entités matérielles.

Mon but dans ce projet est de gérer les différentes acquisitions qui correspondent aux parties colorées en **orange** sur le diagramme, d'une part avec les différents capteurs à l'aide de l'exécutable Acquisitions développé sous python qui va récupérer les données du capteur de température avec le protocole 1-wire et les données du capteur d'humidité avec le protocole SPI.

Dans la partie Acquisitions il y'a également une gestion des acquisitions depuis un serveur FTP situé au service central de N'PY à Lourdes qui contient des images provenant des caméras de la station, pour cela j'ai développé une application FTP qui va récupérer les images depuis ce serveur.

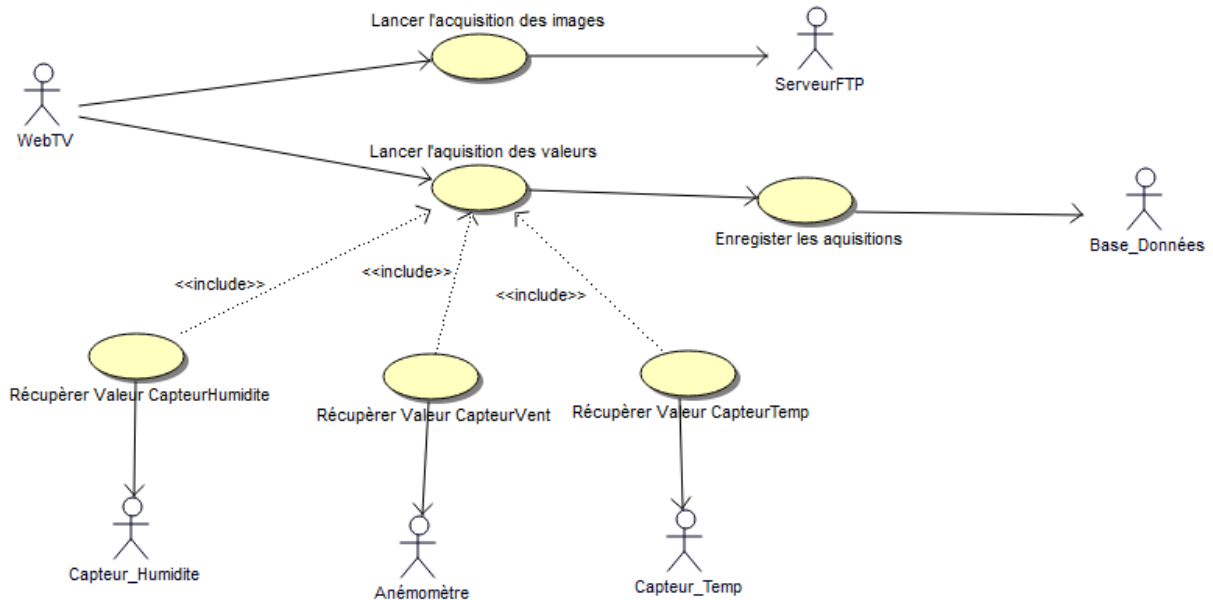
Diagramme de déploiement:



3- Diagramme des cas d'utilisation

Comme indiqué dans le diagramme des cas d'utilisation ci-dessous, qui permet de représenter les fonctionnalités et le contexte d'utilisation du système, les valeurs des capteurs sont récupérées par l'application Acquisitions et sont enregistrées dans une base de données locale sous LAMP.

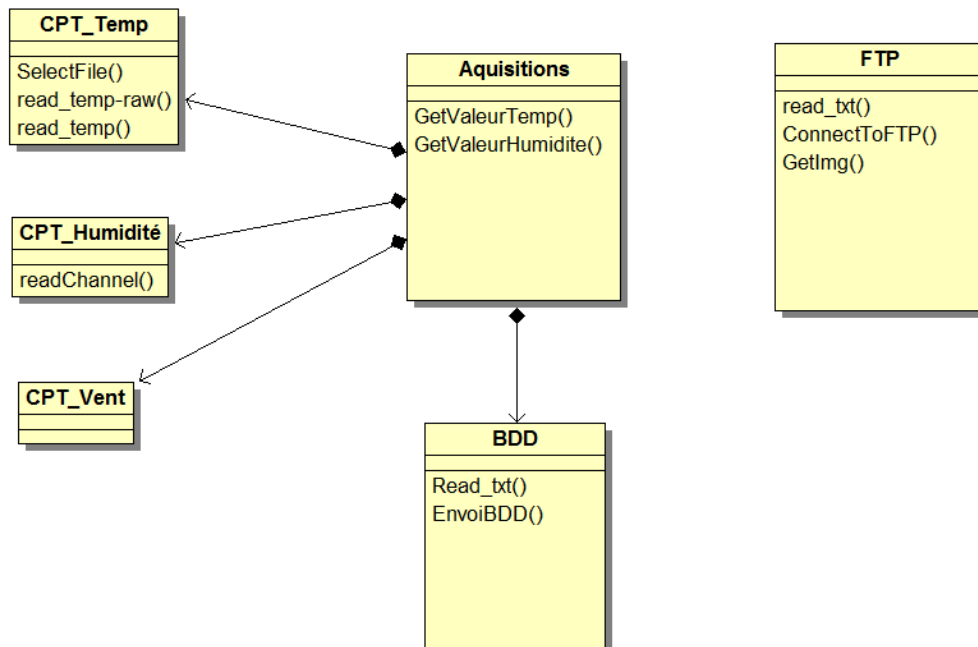
Diagramme des cas d'utilisation:



Il y'a également la partie acquisition des images qui enregistre les images du serveur FTP en local sur la raspberry. Les acquisitions sont lancées automatiquement par une tâche cron, toutes les 30min pour les acquisitions des capteurs et toutes les 15 min pour les images.

4- Diagramme de classes

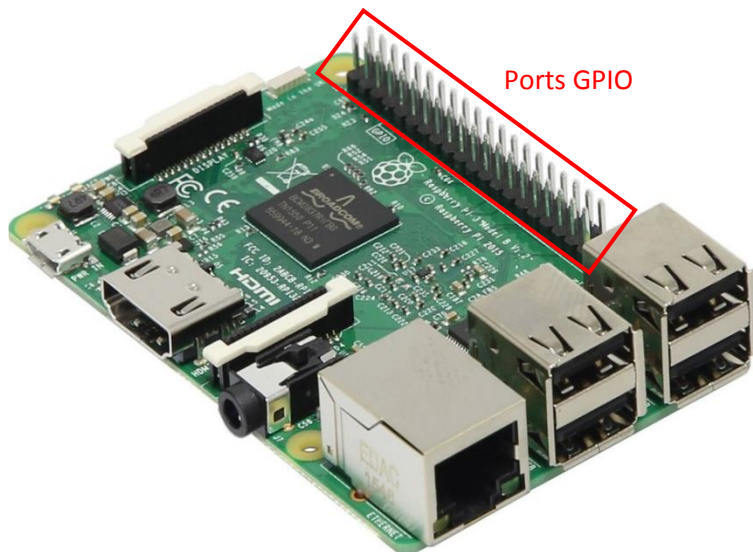
Afin de montrer plus en détail la structure du code d'acquisition, je vais utiliser un diagramme de classe qui permet de montrer l'architecture logicielle du projet ainsi que les relations entre les classes.



L'exécutable acquisition est la classe centrale avec des agrégations par valeur aux classes capteurs et à la classe BDD. Les classes CPT_Temp et CPT_Humidite permettent de récupérer les valeurs des différents capteurs et la classe BDD permet de se connecter et d'envoyer les données des capteurs vers la base de données. Il y'a également une classe FTP qui permet de se connecter et récupérer les images depuis le serveur.

5- Qu'est-ce qu'une raspberry PI ?

La raspberry PI est le support qui a été choisi par le client pour gérer les WebTV, c'est un nano-ordinateur de petite taille avec un prix moindre qui utilise généralement le système d'exploitation Linux, dans notre cas la raspberry utilise la distribution Debian de linux qui supporte nativement la raspberry PI.



3V3 Power	1	2	5V Power
GPI02 SD/L1 I2C	3	4	5V Power
GPI03 SCL1 I2C	5	6	Ground
GPI04 I-wire	7	8	GPI014 UART0_TXD
Ground	9	10	GPI015 UART0_RXD
GPI017	11	12	GPI018 PCM_CLK
GPI027	13	14	Ground
GPI022	15	16	GPI023
3V3 Power	17	18	GPI024
GPI010 SPI0_MOSI	19	20	Ground
GPI09 SPI0_MISO	21	22	GPI025
GPI011 SPI0_SCLK	23	24	GPI08 SPI0_CE0_N
Ground	25	26	GPI07 SPI0_CE1_N
ID_SD I2C ID EEPROM	27	28	ID_SC I2C ID EEPROM
GPI05	29	30	Ground
GPI06	31	32	GPI012
GPI013	33	34	Ground
GPI019	35	36	GPI016
GPI026	37	38	GPI020
Ground	39	40	GPI021

Elle dispose de 40 pins dont 27 pins GPIO (General Purpose Input/Output) qui sont des ports d'entrées-sorties utilisés sur un grand nombre de microcontrôleurs. Ils permettent de communiquer avec des composants électroniques et circuits externes comme des capteurs dans notre cas. Les autres pins GPIO correspondent à des pins d'alimentation et ground afin d'alimenter les différents périphériques connectés à la Raspberry.

6- Planification du travail

Afin de travailler de manière efficace, nous avons utilisé la méthode SCRUM qui permet d'organiser et planifier le travail par étapes comme expliqué dans le dossier général et je vais présenter les différentes US (Stories Utilisateur) qui m'ont été assignées à chaque sprint.

Sprint 1 : Lors de ce sprint l'objectif de mon US1 était dans un premier temps de configurer la Raspberry avec l'installation des différentes entités logicielles avec Debian, le système d'exploitation puis l'installation de LAMP et la création d'une base de données.

☐ US1 - Mise en place RaspBerry Terminé

| Configuration de la raspberry

- ☑ Installation de Debian Valerian Alamar
- ☑ Installation de LAMP Valerian Alamar
- ☑ Création d'une Base de données Valerian Alamar

Dans un second temps, lors du sprint 1, dans mon US2, j'ai effectué l'installation du serveur FTP sur un PC afin de tester l'Application FTP que j'ai développé ensuite afin de récupérer des images depuis ce serveur.

☐ US2 - FTP Terminé

| Gestion du serveur FTP

- ☑ Installation d'un serveur FTP Valerian Alamar
- ☑ Code d'acquisition FTP Valerian Alamar

Sprint 2 : Ensuite, lors du second sprint, j'ai eu un seul US dans lequel je me suis occupé entièrement du capteur de température avec le câblage et le codage de la classe pour ce capteur nommée « **CPT_Temp** ».

C'est également dans ce sprint que j'ai effectué la classe pour la base de données nommée « **BDD** » afin d'envoyer les valeurs des différents capteurs vers la base de données

☐ US1 - Capteur température Terminé

| Mise en place du capteur de température.

- ☑ Câblage du capteur de température Valerian Alamar
- ☑ Code d'acquisition de température Valerian Alamar
- ☑ Code d'envoi à la BDD Valerian Alamar

Puis, lors du sprint final, deux US m'ont été assignés, un pour le capteur d'humidité et un pour l'anémomètre. Actuellement, le câblage du capteur d'humidité a été effectué mais je n'ai pas pu valider le code car je n'ai pas pu tester s'il était fonctionnel à cause d'un composant non compatible avec le protocole utilisé par ce capteur. Cependant, pour le 2^{ème} US, je n'ai pas eu le temps de m'occuper de l'anémomètre.

☐ US1- Capteur d'humidité En cours

| Mise en place du capteur d'humidité

- ☑ Câblage du capteur d'humidité Valerian Alamar
- ☐ Codage d'acquisition du capteur d'humidité Valerian Alamar

☐ US2- Capteur Anémomètre En cours

| Mise en place du capteur d'anémomètre

- ☐ Câblage du capteur Anémomètre Valerian Alamar
- ☐ Codage d'acquisition du capteur Anémomètre Valerian Alamar

II - La base de données

1- Présentation de LAMP

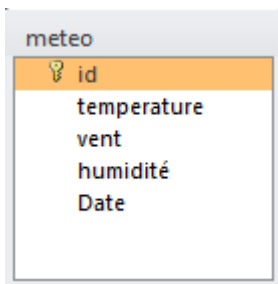
LAMP est un ensemble d'entités logicielles permettant de créer des serveurs de site web. Il y'a 4 entités différentes dans ce pack : Linux qui est le système d'exploitation sur lequel est installé le serveur, Apache est un serveur WEB http qui permet d'interpréter le langage de codage du site web, MySQL est le système de gestion de bases de données avec une interface intégrée qui permet de créer et gérer des bases de données et PHP qui est un langage de codage Web dynamique qui permet d'interagir avec la base de données dans la structure HTML du site.

2- Présentation de la base de données

La base de données est nommée « npy » et contient une seule table, nommée météo avec 5 champs :

- l'ID qui est la clé primaire de la table
- La température, la vitesse du vent et l'humidité qui correspondent aux champs sur lesquels les valeurs des capteurs sont enregistrées en type float.
- La Date de type « datetime » qui correspond à l'heure et la date où les mesures de l'enregistrement ont été effectuées.

Ces valeurs sont ensuite affichées sur l'écran de la WebTV à l'aide d'un site Web intégré réalisé par un autre membre du projet.



#	Nom	Type	Interclassement	Attributs	Null	Valeur par défaut	Commentaires	Extra
<input type="checkbox"/> 1	id	int(11)			Non	Aucune		AUTO_INCREMENT
<input type="checkbox"/> 2	temperature	float			Non	Aucune		
<input type="checkbox"/> 3	vent	float			Non	Aucune		
<input type="checkbox"/> 4	humidite	float			Non	Aucune		
<input type="checkbox"/> 5	date	datetime			Non	current_timestamp()		

3- Envoi des données vers la BDD

Puis, afin de se connecter à la base de données locale j'ai créé une classe nommée « **BDD** » qui contient 2 méthodes : Premièrement « **Read_txt** » qui permet de lire les informations de connexion à la base de données spécifiées sur un fichier texte :

```
def read_txt(self):
    #Ouverture du fichier Info_Connexion.txt et récupération des informations dans une variable
    txt = open("Info_connexion.txt", "r")
    liste_txt = txt.readlines()
    IP = liste_txt[9].replace('IP_BDD : ','')
    user = liste_txt[10].replace('user : ','')
    pswd = liste_txt[11].replace('password : ','')
    DB = liste_txt[12].replace('DB : ','')
    return IP,user,pswd,DB
```

Info_connexion.txt

```
IP_BDD : localhost
user : root
password : WebTV
DB : npy
```

Ensuite, la méthode « **EnvoiBDD** » va utiliser ces informations pour effectuer la connexion à la base de données puis va envoyer les valeurs des capteurs vers la base de données à l'aide d'une requête SQL, pour cela je récupère les valeurs des capteurs depuis les différentes classes dans l'application Acquisitions et j'appelle la méthode « **EnvoiBDD** » avec pour paramètre les valeurs de température et d'humidité.

```
def EnvoiBDD(self,Temp,Humidite):
    retr_values = self.read_txt()
    liste = list(retr_values)
    IP_BDD = retr_values[0]
    User_BDD = retr_values[1]
    pswd_BDD = retr_values[2]
    DB_BDD = retr_values[3]
    #Connexion à la BDD
    bdd = mysql.connector.connect(host=IP_BDD,user=User_BDD,password="WebTV",database=DB_BDD)
    curseur = bdd.cursor()
    #Requête d'envoi vers la BDD
    curseur.execute("INSERT INTO meteo (temperature, humidite, vent)
                    VALUES ('+Temp+', '+Humidite+',0);")
    bdd.commit()
    bdd.close()
```

Dans cette classe j'ai utilisé la librairie **mysql.connector** afin d'effectuer la connexion à la base de données

III – Le Serveur FTP

1- Présentation du protocole FTP

FTP (File Transfer Protocol) est un protocole de communication permettant de partager et gérer différents types de fichiers entre des ordinateurs et périphériques à l'aide d'une connexion en TCP/IP.

FTP fonctionne sur un modèle client-serveur dans lequel le client envoie des requêtes auxquelles réagit le serveur. Le serveur est situé sur un ordinateur sur lequel est installée une application serveur FTP. Pour accéder à ce serveur, on utilise un logiciel client FTP qui va communiquer avec l'application serveur. Le protocole FTP utilise 2 ports différents : le port 21 pour gérer la connexion entre les deux parties et le port 20 pour l'échange de données.

Sur l'application serveur, on peut définir des utilisateurs avec différentes autorisations comme télécharger, envoyer ou encore gérer les fichiers contenus sur le serveur depuis un périphérique distant. L'utilisateur peut accéder au serveur FTP à l'aide de l'adresse IP de l'ordinateur hébergeur ainsi que les identifiants de l'utilisateur définis sur l'application serveur.

2- Récupération des images sur la raspberry

L'application d'Acquisition des images utilise la librairie « **ftplib** » qui permet d'implémenter le côté client du protocole FTP et ainsi communiquer avec un serveur FTP à l'aide d'un programme Python.

Il y'a 3 méthodes dans la classe « **FTP** », la première nommé « **read_txt** » qui est la même que pour la classe pour la base de données, elle permet de récupérer les informations de connexion au serveur FTP depuis un fichier texte.

```
def read_txt(self):  
    #Ouverture du fichier Info_Connexion.txt et récupération des informations dans une variable  
    txt = open("Info_connexion.txt", "r")  
    liste_txt = txt.readlines()  
    IP = liste_txt[2].replace('IP_FTP : ', '')  
    user = liste_txt[3].replace('User : ', '')  
    pswd = liste_txt[4].replace('password : ', '')  
    DestFile = liste_txt[5].replace('Fichier Destination : ', '')  
    return IP,user,pswd, DestFile
```

Info_connexion.txt

```
IP_FTP : "10.0.134.14"  
User : "WebTV"  
password : "WebTV"  
Fichier Destination : "Camera"
```

Ensuite, la 2^{ème} méthode nommée « **ConnectToFTP** » permet d'effectuer la connexion au serveur FTP à l'aide des informations récupérées par la méthode précédente

```
def ConnectToFTP(self):
    retr_values = self.read_txt()
    liste = list(retr_values)
    IP_FTP = retr_values[0]
    User_FTP = retr_values[1]
    pswd_FTP = retr_values[2]
    DestFile = retr_values[3]
    ftp = FTP()
    ftp.login(user=User_FTP, passwd=pswd_FTP)
    ftp.cwd(DestFile)
    return ftp
```

Puis la 3^{ème} méthode, « **GetImg** » permet de récupérer les images contenues dans le serveur FTP. En l'absence d'informations concernant le fonctionnement du serveur FTP, nous sommes partis du principe que les images avaient pour nom « img x.jpg », j'ai donc fait une variable nommée « nb » qui s'incrémente à chaque tour de boucle

A chaque tour de boucle la commande « RETR » est envoyée vers le serveur FTP qui va comprendre qu'il doit envoyer un fichier vers le client dont le nom correspond avec la variable « filename ».

```
def GetImg(self,ftp):
    nb=0
    while True:
        nb = int(nb)
        nb = nb+1
        nb = str(nb)
        #on définit le nom du fichier qu'on veut recuperer
        filename = 'img'+ nb + '.jpg'
        #envoi d'une trame au serveur FTP pour lui indiquer le fichier qu'on veut recuperer
        with open(filename, 'wb') as fp:
            ftp.retrbinary('RETR '+ filename, fp.write)
    ftp.quit()
```

IV – Acquisitions

1- Capteur de température

1. Fonctionnement du capteur

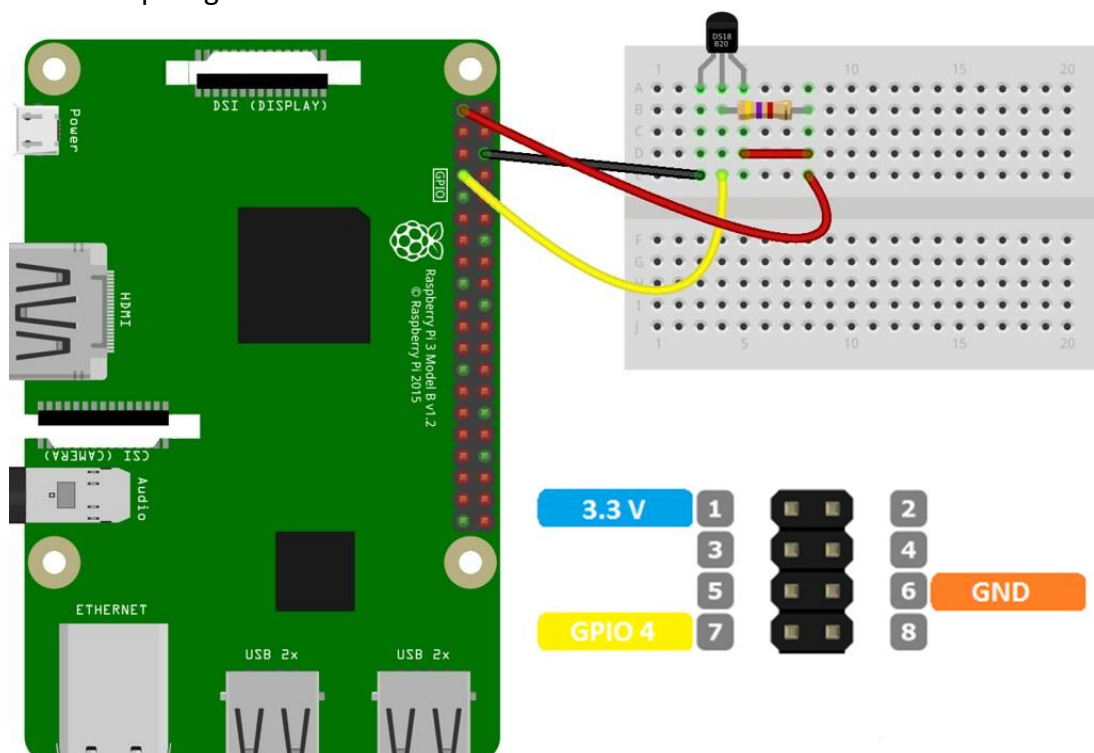
Le 1^{er} capteur est un capteur de température nommé **DS1822+** par la marque **Maxim Integrated** qui fonctionne à l'aide du protocole 1wire.



Le protocole 1wire a été conçu par **Dallas Semiconductor** et fonctionne en suivant le principe de master/slave, La raspberry est le master qui contrôle le bus et interroge le slave, ici le capteur de température qui va envoyer une trame vers un fichier spécifique sur la raspberry quand le master l'interroge.

2. Montage du capteur

Le port GPIO 4 est le port par défaut pour le protocole 1-wire, c'est donc sur ce port que la broche données du capteur est branché, le capteur est alimenté sur le port 1 de la raspberry en 3,3V et il est également relié à un port ground.



On peut aussi noter la présence d'une résistance de 5.7 KOhm entre le port GPIO et le port d'alimentation car étant donné qu'il y'a une seule sonde il n'y a pas besoin d'une tension élevée pour tout faire fonctionner.

3. Code d'acquisition de la température

Afin de récupérer les valeurs du capteur j'ai créé une classe nommée « **CPT-Temp** » avec 3 méthodes : « **SelectFile** » qui récupère l'emplacement du fichier où le capteur envoie les données, « **read_temp_raw** » qui lit ce fichier et « **read_temp** » qui retourne la valeur de température en type float.

Juste avant la déclaration de ma classe j'active les modules 1Wire à l'aide de librairie « **os** » qui est installée de base avec Python 3, ce qui permet d'entrer une commande dans le terminal lors du lancement de l'application, ces commandes permettent d'activer les modules 1-wire de la raspberry.

```
#execute les commandes dans le terminal
os.system('sudo modprobe w1-gpio')
os.system('sudo modprobe w1-therm')
```

La première méthode, « **SelectFile** », est celle qui permet de sélectionner l'emplacement de la trame contenant la température que le capteur envoie à chaque fois qu'il est interrogé par la raspberry.

```
def Selectfile():
    base_dir = '/sys/bus/w1/devices/'
    device_folder = glob.glob(base_dir + '*0*')[0]
    device_file = device_folder + '/w1_slave'
    return device_file
```

Contenu du fichier "w1_Slave"

```
67 01 4b 46 7f ff 09 10 3b : crc=3b YES
67 01 4b 46 7f ff 09 10 3b t=22437
```

Cette trame est située dans le dossier « w1_Slave », dans le répertoire sys / bus/ w1/ **devices**/ «numéro de série du capteur ».

Le numéro de série du capteur est unique donc au lieu de spécifier un chemin spécifique pour accéder au dossier j'ai utilisé la librairie **Glob** qui va ouvrir le dossier situé sur le 1^{er} emplacement dans le dossier **devices** qui correspond au numéro de série du capteur afin que le code puisse fonctionner sur toutes le WebTV étant donné qu'elles auront toutes un capteur de température et donc des numéros de série différents.

Ensuite, la fonction **read_temp_raw()** va lire et récupérer la trame où est contenue la température dans le fichier « w1_Slave ».

```
def read_temp_raw(self):
    #on ouvre le fichier "w1_Slave" dont le chemin est defini dans la variable device_file
    f = open(CPT_TEMP.Selectfile(), 'r')
    #on attribue le contenu du fichier a la variable "lines"
    lines = f.readlines()
    f.close()
    return lines
```

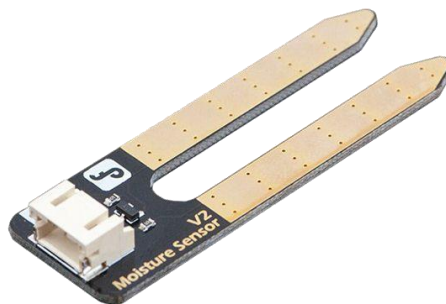
Ensuite dans la fonction **read_temp()** j'appelle la méthode précédente afin de récupérer la trame et le code va isoler la valeur de température de la trame et la convertir en valeur de type float.

```
def read_temp(self):
    l = CPT_TEMP()
    lines = l.read_temp_raw()
    #si le CRC est incorrect on attend 0.2sec et on lit la prochaine trame jusqu'a que le CRC soit correct
    while lines[0].strip()[-3:] != 'YES':
        time.sleep(0.2)
        lines = CPT_TEMP.read_temp_raw()
        #on cherche dans la 2eme ligne l'endroit ou est marque "t="
    equals_pos = lines[1].find('t=')
    if equals_pos != -1:
        temp_string = lines[1][equals_pos+2:]
        #on convertit la valeur contenue dans la trame en float
        temp_c = float(temp_string) / 1000.0
        temp_s = str(temp_c)
        return temp_s
```

2- Capteur d'humidité

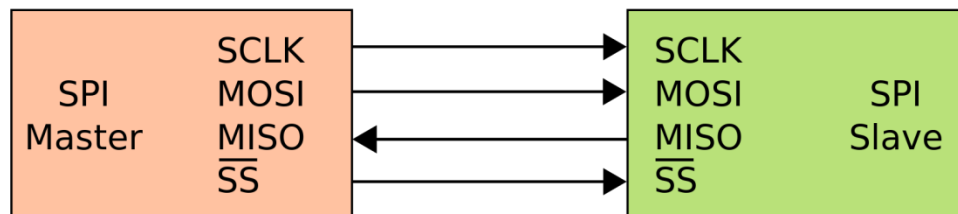
1. Fonctionnement du capteur

Le second capteur est un capteur d'humidité analogique nommé **SEN0114** par la marque **DF Robot** qui fonctionne à l'aide du protocole SPI.

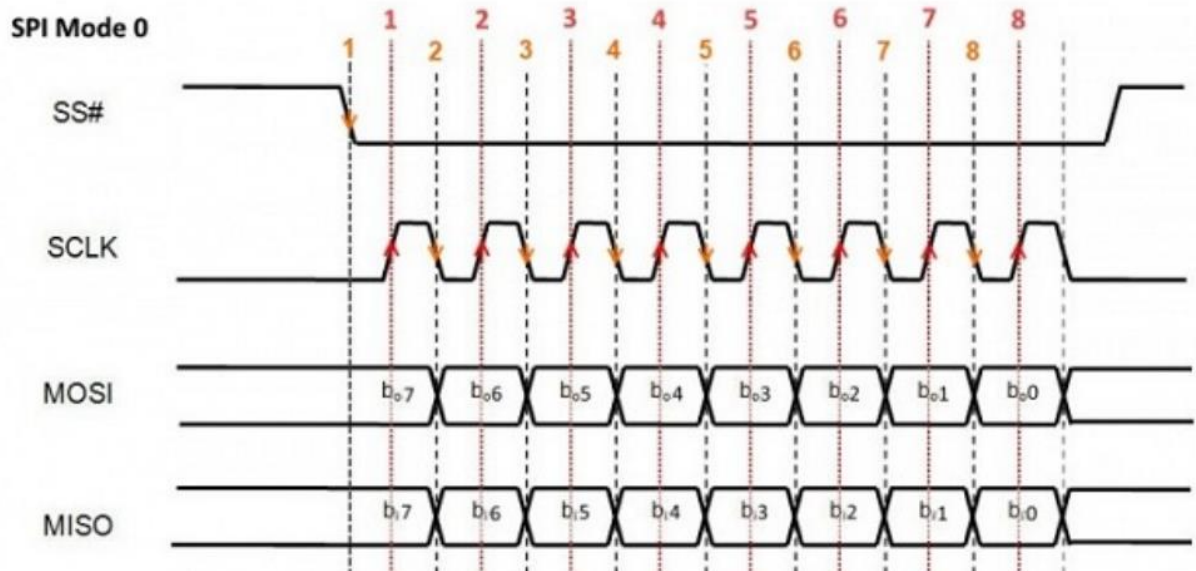


Le protocole **SPI** (Serial Peripheral Interface) a été conçu par Motorola en 1980 et suit également un principe master/slave en full duplex. Tout comme le précédent protocole, le master est la raspberry et le slave est le capteur. Le bus SPI utilise 4 signaux logiques : **SCLK** (*Serial Clock*) qui correspond à l'horloge du maître. A chaque tour d'horloge, un bit est transmis et donc un octet tous les 8 tours d'horloge. Il y'a aussi les signaux **MOSI** (*Master Out Slave Input*) qui correspond aux requêtes envoyées par le maître et **MISO** (*Master Input Slave Output*) qui correspond aux réponses renvoyées par l'esclave.

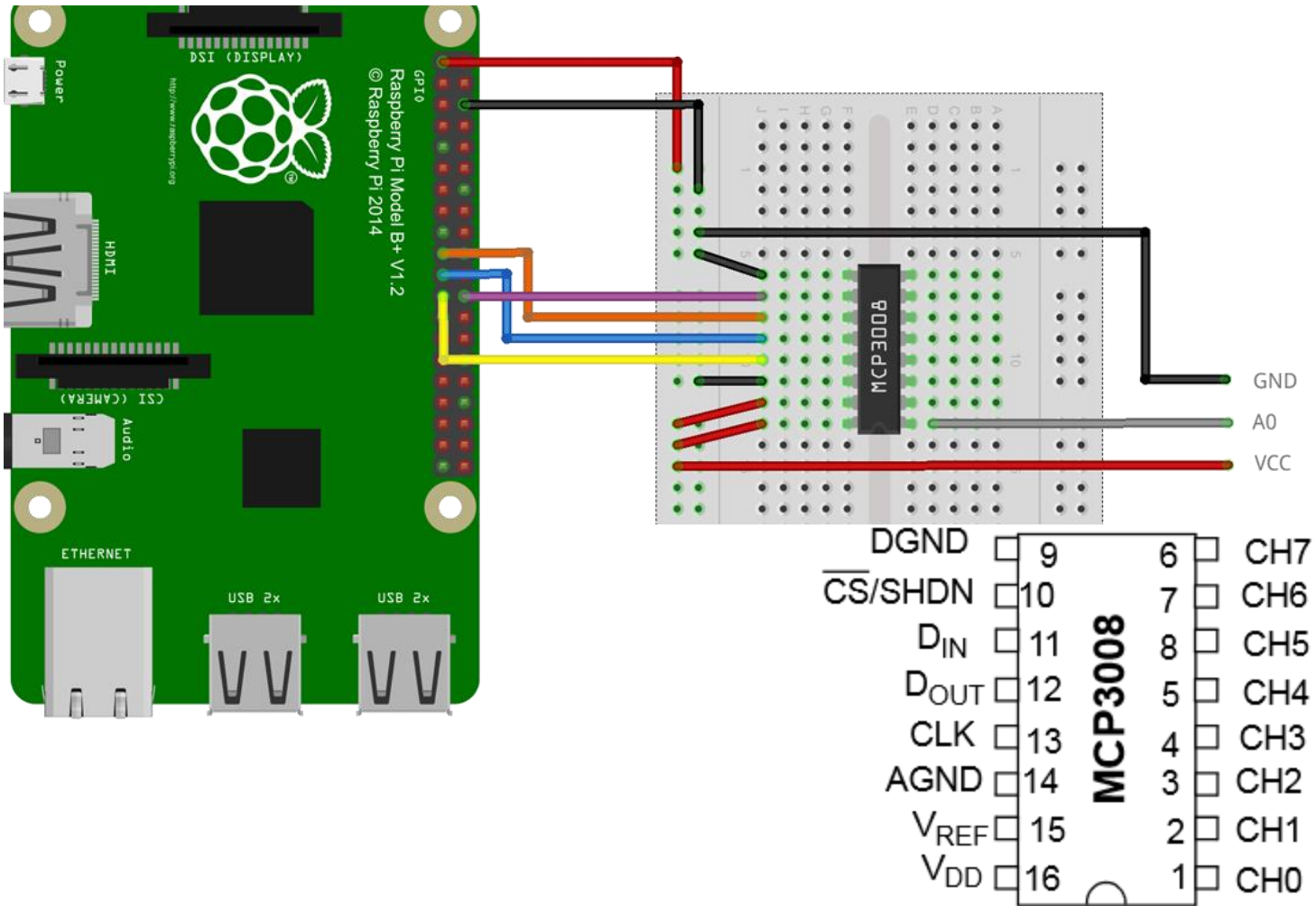
Le quatrième signal est le signal **SS** (*Slave Select*) qui permet au master de sélectionner le slave avec lequel il veut communiquer.



Il y'a 4 modes de transmission SPI, dans mon code j'ai utilisé la mode 0 dans lequel la communication commence par l'envoi d'un signal **Slave Select** a 0 afin d'indiquer le début de la communication puis le master va envoyer 8 signal d'horloge, à chaque front montant d'horloge, le master envoie un bit d'instruction et à chaque front descendant, l'esclave renvoie un bit correspondant à l'instruction vers le maître puis la communication se termine par le passage à 1 du signal de **Slave Select**.



2. Montage du capteur



Le tableau ci-dessous présente les branchements entre la Raspberry, le convertisseur et le capteur : CLK correspond au signal d'horloge, CD/SHDN correspond au signal de sélection de l'esclave et CH0 correspond au channel sur lequel le capteur envoie ses données.

Raspberry PI	MCP3008	SEN0114
Pin 1 (3.3V)	Pin 16 (VDD)	VCC
Pin 1 (3.3V)	Pin 15 (VREF)	
Pin 6 (GND)	Pin 14 (AGND)	GND
Pin 23 (DCLK)	Pin 13 (CLK)	
Pin 21 (MISO)	Pin 12 (D out)	
Pin 19 (MOSI)	Pin 11 (D in)	
Pin 24 (CE0)	Pin 10 (CD/SHDN)	
Pin 6 (GND)	Pin 9 (DGND)	
	Pin 1 (CH 0)	A0

3- Problèmes rencontrés

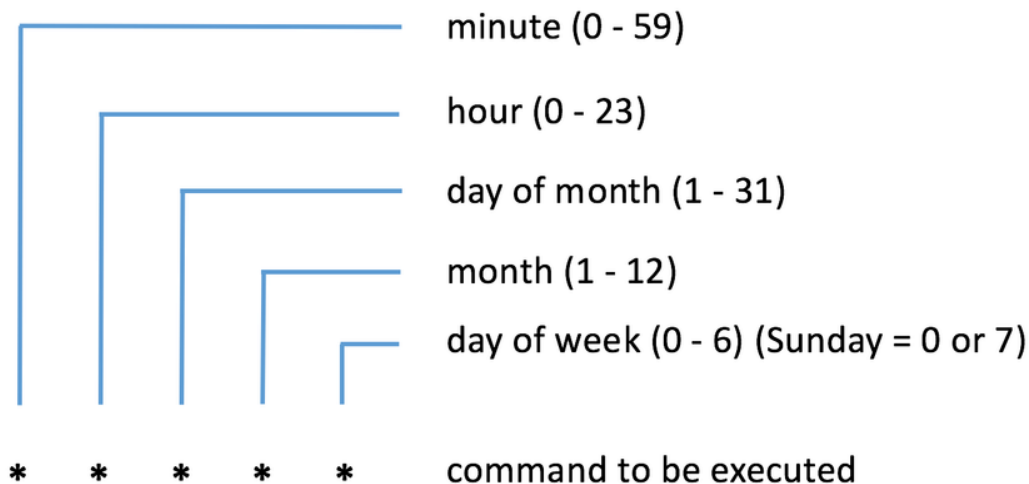
Afin de récupérer la valeur d'humidité, j'ai créé la classe « **CPT-Humidité** » qui utilise la librairie « **Spidev** » qui permet de communiquer avec des périphériques SPI.

Cette classe contient une méthode nommée « **readChannel** » qui permet de retourner la valeur d'humidité mais comme expliqué dans la partie planification, je n'avais pas à disposition de convertisseur analogique compatible avec le protocole SPI donc je n'ai pas pu vérifier si le code fonctionne mais j'ai tout de même inclus cette partie afin d'expliquer le fonctionnement et les branchements du capteur.

4- La tâche Cron

Afin d'effectuer les acquisitions de manière automatique, j'ai utilisé une tâche Cron afin de lancer automatiquement l'exécutable acquisition toutes les 30 minutes pour récupérer les valeurs des capteurs et toutes les 15min pour les images.

Cron est un programme permettant d'exécuter automatiquement des scripts à une date et une heure prédéfinie à l'aide des commandes à entrer dans un fichier lu par le logiciel avec les paramètres de temps suivi de la commande à exécuter.



Afin de configurer les tâches Cron il faut accéder au fichier de configuration nommé crontab avec la commande **crontab -e** et insérer les commandes suivantes pour les deux acquisitions différentes dans ce fichier.

Images : `*/15 * * * * /usr/bin/python3 /home/pi/Desktop/WebTV/FTP.py`

Capteurs : `*/30 * * * * /usr/bin/python3 /home/pi/Desktop/WebTV/Acquisitions.py`

5- Exécutable Acquisitions

L'exécutable Acquisition et la classe centrale avec 2 méthodes pour récupérer les valeurs du capteur de température et du capteur d'humidité et qui les envoie ensuite vers la base de données en appelant la méthode « **EnvoiBDD** » de la classe BDD avec pour paramètre les valeurs des capteurs.

La première ligne commentée permet de rendre la classe acquisition exécutable par la tâche Cron.

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

```
from BDD import BDD
from CPT_Temp import CPT_TEMP
from CPT_Humidite import CPT_HUMIDITE
```

```
bdd = BDD()
temp = CPT_TEMP()
humidite = CPT_Humidite()
```

```
class Acquisition:
```

```
    def GetValeurTemp():
        Val_Temp = temp.read_temp()
        return Val_Temp

    def GetValeurHumidite():
        Val_Humidite = humidite.readChannel(0)
        return Val_Humidite
```

```
bdd.EnvoiBDD(Acquisition.GetValeurTemp(),Acquisition.GetValeurHumidite())
```

V - Manuel D'installation Acquisitions

I – Matériel et logiciels requis

Entités matérielles :

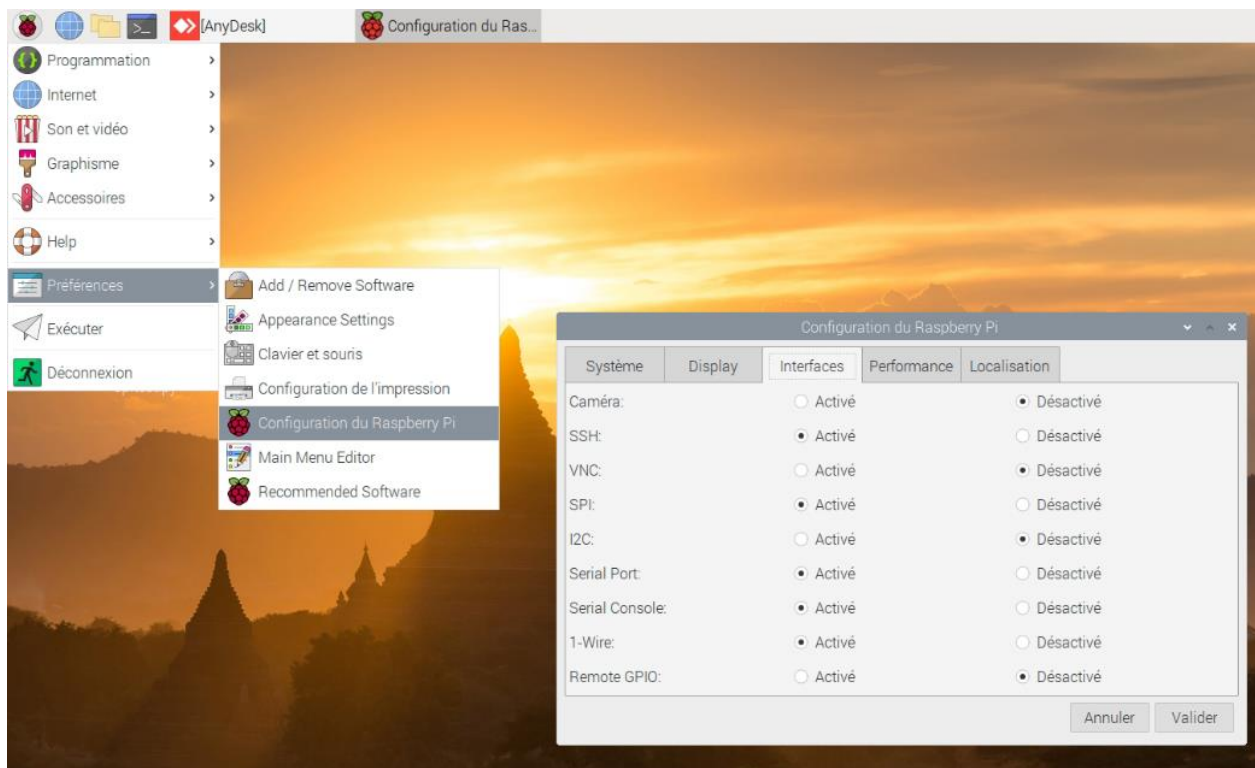
- Une raspberry PI
- Un capteur de température DS1822+, marque Maxim Integrated
- Un capteur d'humidité SEN0114, marque DF Robot
- Un convertisseur Analogique – Numérique MCP3008

Entités logicielles :

- Exécutable Acquisitions
- Classes : BDD.py, FTP, CPT_Temp.py, CPT_Humidite.py
- un Fichier texte nommé «infos_connexion.txt » avec les informations de connexion au serveur FTP et à la base de données
- La base de données « npy »
- Le site web embarqué
- Python 3

II – Configuration de la raspberry

Ouvrir la Configuration du Raspberry PI et activer les interfaces 1-Wire, SPI et SSH



Les exécutables FTP et Acquisitions doivent être présents être dans le répertoire home/pi/Desktop/WebTV afin que la tâche Cron fonctionne

II – Installation des logiciels et librairies

Avant l'installation des logiciels, il faut mettre à jour la Raspberry à l'aide des commandes suivantes à entrer sue le terminal de linux

```
sudo apt-get update  
sudo apt-get upgrade
```

Python est installé de base avec linux Debian, il suffit juste d'installer la base de données.

1. Installation de LAMP

Ouvrir le terminal de linux et entrer les commandes suivantes pour installer Apache :

```
sudo apt install apache2  
sudo chown -R pi:www-data /var/www/html/  
sudo chmod -R 770 /var/www/html/
```

Toujours dans le terminal entrer la commande suivante pour installer PHP

```
sudo apt install php php-mbstring
```

Entrer la commande suivante pour installer MySQL

```
sudo apt install mariadb-server php-mysql
```

Puis créer un utilisateur avec les commandes :

```
CREATE USER 'root'@'localhost' IDENTIFIED BY 'password';  
GRANT ALL PRIVILEGES ON *.* TO 'root'@'localhost' WITH GRANT  
OPTION;
```

Afin que le programme d'acquisitions fonctionne, il faudra modifier le fichier "Infos_Connexion.txt" et remplacer les champs « user » et « password » avec l'utilisateur et le mot de passe définis avec la commande précédente

Pour finir, installer PHPmyAdmin avec la commande :

```
sudo apt install phpmyadmin
```

2. Installation des librairies

La librairie « ftplib » pour la partie ftp est présente de base sur la raspberry, il faut installer les autres :

« Mysql connector » pour la base de données :

```
Sudo pip install mysql-connector-python
```

« Spidev » pour le support du protocole SPI pour le capteur de température

```
Sudo pip install spidev
```

« Glob » afin de sélectionner le fichier ou est contenu la trame du capteur de température

```
pip install glob2
```

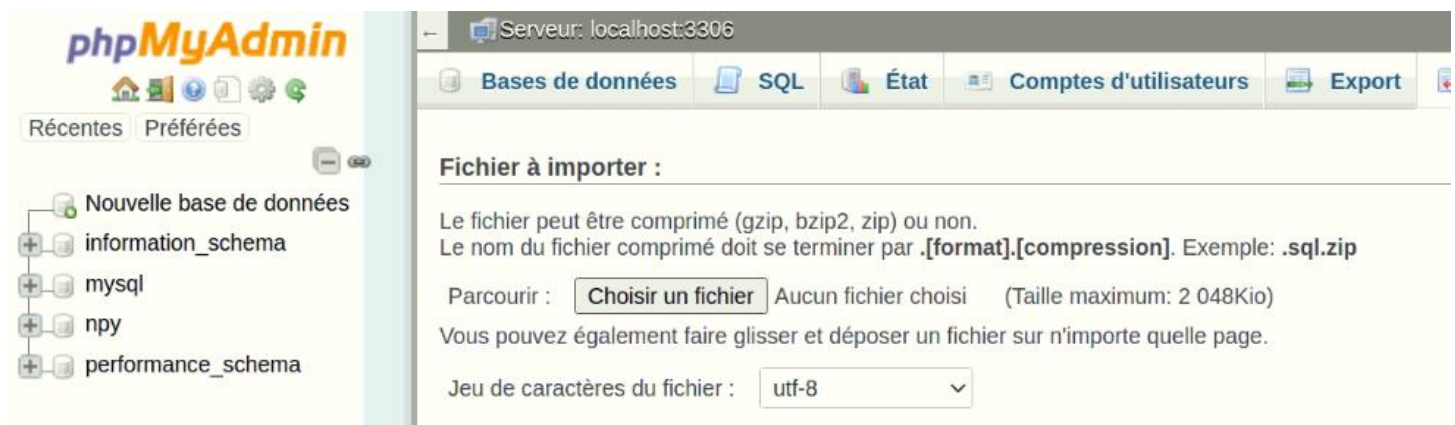
Puis « os » afin d'activer des commandes sur le terminal depuis python

```
pip install os-sys
```

3. Installation de la base de données

Afin d'installer la base de données, il faut accéder a PHPmyAdmin en utilisant l'URL

« <http://localhost/phpmyadmin/> » sur le navigateur de la raspberry et se connecter avec l'utilisateur et l'identifiant définis lors de la création de l'utilisateur lors de l'installation de MySQL



Appuyer sur choisir un fichier et sélectionner la base de données « npy.sql » qui est fournie avec les exécutables d'acquisition puis appuyer sur exécuter en bas de la page.

VI : Conclusion

Pour conclure sur ce projet, cela m'a été énormément bénéfique et m'a apporté de nouvelles connaissances et compétences sur le domaine de l'informatique avec :

- L'utilisation de la méthode Scrum qui nous donne un aperçu des moyens de planifications qui peuvent être utilisés en entreprise pour la gestion d'un projet.
- Cela m'a donné également beaucoup d'expérience sur les Raspberry et le système d'exploitation Linux et ses nombreuses utilisations grâce aux ports GPIO qui permettent de communiquer avec un grand nombre de périphériques à signaux numériques (ou analogiques à l'aide d'un convertisseur) avec de nombreux protocoles compatibles.
- L'utilisation du langage python tout au long du projet qui m'a permis de découvrir les possibilités de ce langage ainsi qu'une meilleure maîtrise de ce dernier.
- La découverte de nouveaux protocoles de communication et la manière dont ils fonctionnent.

Ce projet m'a également permis de mieux appréhender le travail en équipe et ses avantages, avec l'exemple de la méthode Scrum qui permet de coordonner tous les membres qui travaillent sur le projet ou encore l'utilisation de Github qui permet de mettre facilement à disposition toutes sortes de codes entre tous les membres de l'équipe.

D'un point de vue personnel j'ai apprécié travailler sur ce premier projet concret qui a été enrichissant sur de nombreux points. Cependant, je n'ai pas pu finir totalement ma partie du projet mais les fonctionnalités centrales sont fonctionnelles et je compte terminer la partie du capteur d'humidité dans les prochains jours.

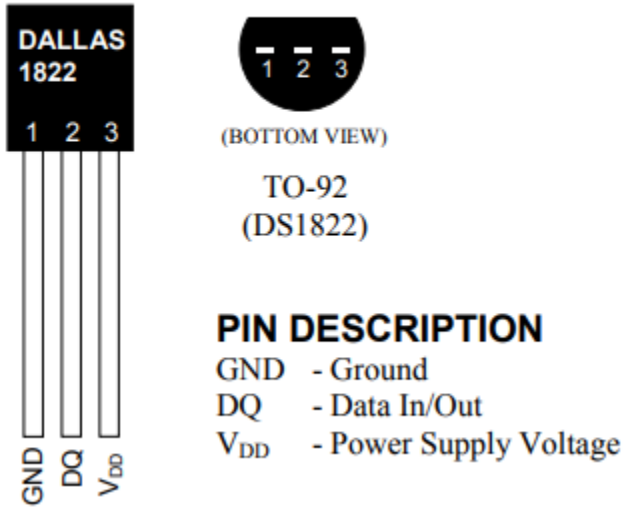
VII : Annexes

1- Documentation Technique

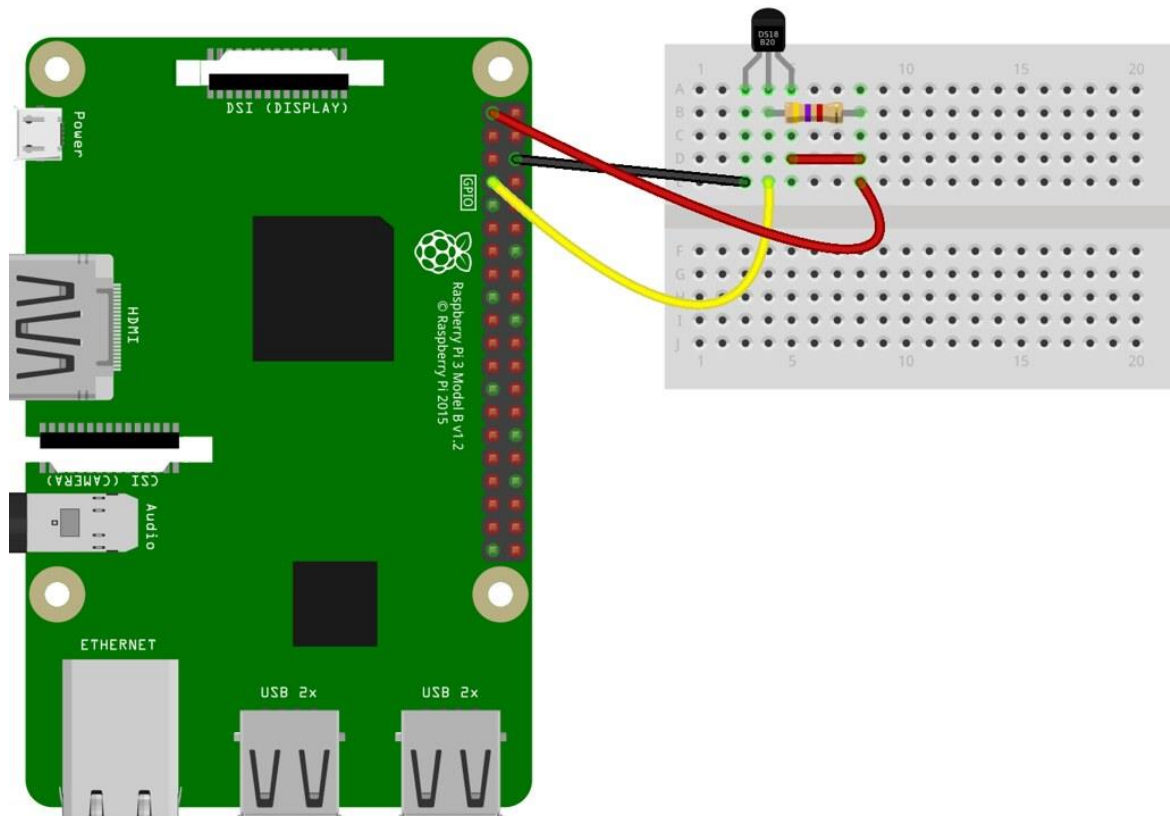
Capteur Température :

<https://4donline.ihs.com/images/VipMasterIC/IC/MAXM/MAXM-S-A0000633411/MAXM-S-A0000773938-1.pdf?hkey=52A5661711E402568146F3353EA87419> La documentation technique du capteur afin de savoir à quoi correspond chaque PIN du capteur.

PIN ASSIGNMENT

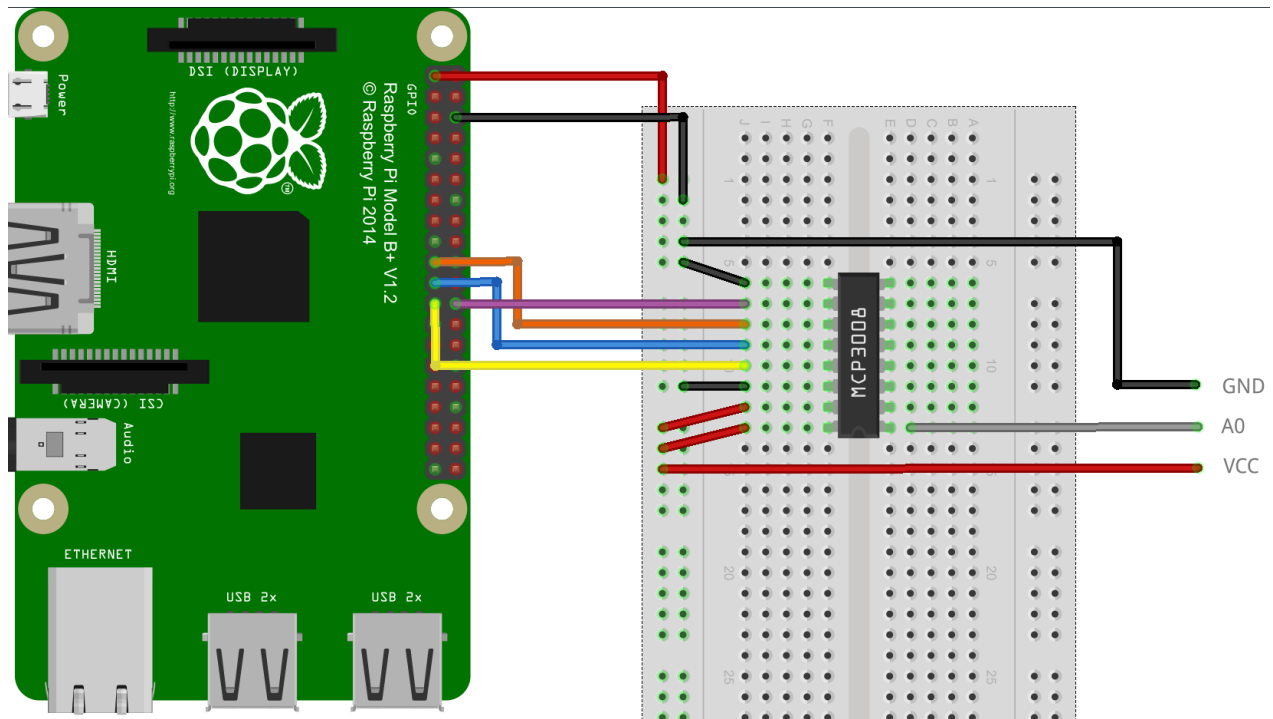


<https://bigl.es/ds18b20-temperature-sensor-with-python-raspberry-pi/> Pour le branchement du capteur



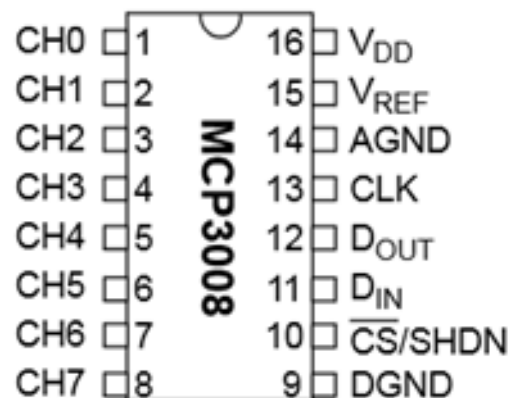
Capteur humidité :

<https://tutorials-raspberrypi.com/measuring-soil-moisture-with-raspberry-pi/> pour le branchement du capteur



The connections to the MCP3008 are as follows:

RaspberryPi	MCP3008
Pin 1 (3.3V)	Pin 16 (VDD)
Pin 1 (3.3V)	Pin 15 (VREF)
Pin 6 (GND)	Pin 14 (AGND)
Pin 23 (SCLK)	Pin 13 (CLK)
Pin 21 (MISO)	Pin 12 (DOUT)
Pin 19 (MOSI)	Pin 11 (DIN)
Pin 24 (CE0)	Pin 10 (CS/SHDN)
Pin 6 (GND)	Pin 9 (DGND)



Base de données et FTP :

<https://raspberrypi.fr/installer-serveur-web-raspberry-lamp/> pour l'installation de lamp et de la base de données

<https://pythonprogramming.net/ftp-transfers-python-ftplib/> pour la conception du code d'acquisition des images depuis le serveur FTP

```
def grabFile():  
  
    filename = 'example.txt'  
  
    localfile = open(filename, 'wb')  
    ftp.retrbinary('RETR ' + filename, localfile.write, 1024)  
  
    ftp.quit()  
    localfile.close()
```

<https://www.computerhope.com/issues/ch001721.htm> pour la récupération des informations de connexion depuis le fichier texte

```
myfile = open("lorem.txt", "rt") # open lorem.txt for reading text  
contents = myfile.read()         # read the entire file to string  
myfile.close()                   # close the file  
print(contents)                   # print string contents
```





<https://dev.mysql.com/doc/connector-python/en/connector-python-example-connecting.html> Pour la conception du code d'envoi à la base de données avec la librairie « **MySQL-connector** » qui gère la connexion à la base de données.

```
1  import mysql.connector  
2  
3  cnx = mysql.connector.connect(user='scott', password='password',  
4                                host='127.0.0.1',  
5                                database='employees')  
6  cnx.close()
```

1- Fiches de test unitaire





TEST N° : 1		AUTEUR : Alamar Valerian		
IDENTIFICATION DU TEST				
DATE	OBJECTIF		NIVEAU	
17/05/21	Test de la récupération des informations de connexion à la BDD		TU	x
			TI	
			TV	

DESCRIPTION DU TEST A REALISER	
CONTEXTE	
Environnement du test	Cible du test
Python 3.0, Linux Debian	CU : Récupérer les informations de connexion à la BDD
SCENARIO DU TEST	
Exécution de l'application TestUnitaire Choix 1	
RESULTAT(S) ATTENDU(S)	
Récupération des informations de connexion à la base de données	

REALISATION DU TEST	
RESULTAT(S) OBTENU(S)	
<pre> Quel est votre choix ? 1 ('localhost\n', 'root\n', 'WebTV\n', 'npy') </pre>	
EVALUATION DU RESULTAT OBTENU	
Satisfaisant	Erreur
	Critique  Majeure  Mineure 





TEST N° : 2		AUTEUR : Alamar Valerian		
IDENTIFICATION DU TEST				
DATE	OBJECTIF		NIVEAU	
17/05/21	Test de la connexion à la BDD		TU	x
			TI	
			TV	

DESCRIPTION DU TEST A REALISER	
CONTEXTE	
Environnement du test	Cible du test
Python 3.0, Linux Debian	CU : Connexion à la BDD
SCENARIO DU TEST	
Exécution de l'application TestUnitaire Choix 2	
RESULTAT(S) ATTENDU(S)	
Récupération des informations de connexion à la base de données	

REALISATION DU TEST	
RESULTAT(S) OBTENU(S)	
<pre> Quel est votre choix ? 2 Connexion Ok <mysql.connector.connection.MySQLConnection object at 0x762e97b0> </pre>	
EVALUATION DU RESULTAT OBTENU	
Satisfaisant	Erreur
	Critique  Majeure  Mineure 

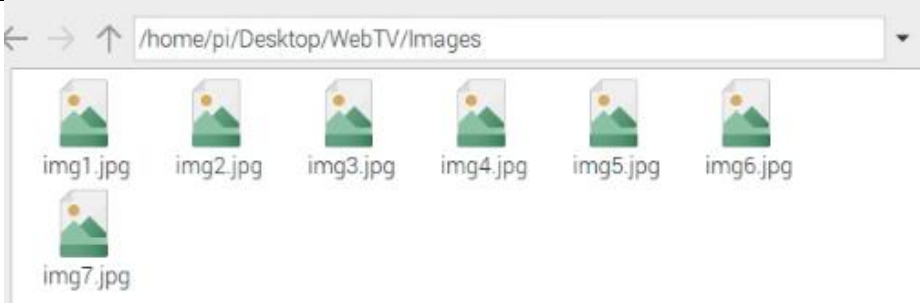




TEST N° : 3		AUTEUR : Alamar Valerian		
IDENTIFICATION DU TEST				
DATE	OBJECTIF		NIVEAU	
17/05/21	Test de l'envoi des données des capteurs vers la base de données		TU	x
			TI	
			TV	

DESCRIPTION DU TEST A REALISER	
CONTEXTE	
Environnement du test	Cible du test
Python 3.0, Linux Debian	CU : Envoi des données des capteurs vers la BDD
SCENARIO DU TEST	
Exécution de l'application TestUnitaire Choix 3	
RESULTAT(S) ATTENDU(S)	
Un enregistrement contenant les valeurs des capteurs et la date à laquelle l'enregistrement a été fait est ajouté à la table météo de la base de données	

REALISATION DU TEST				
RESULTAT(S) OBTENU(S)				
id	temperature	vent	humidite	date
121	10	0	10	2021-05-26 15:35:55
EVALUATION DU RESULTAT OBTENU				
Satisfaisant	Erreur			
	Critique 	Majeure 	Mineure 	





TEST N° : 4		AUTEUR : Alamar Valerian		
IDENTIFICATION DU TEST				
DATE	OBJECTIF		NIVEAU	
17/05/21	Test de la classe FTP		TU	x
			TI	
			TV	

DESCRIPTION DU TEST A REALISER	
CONTEXTE	
Environnement du test	Cible du test
Python 3.0, Linux Debian	CU : récupération des images depuis le serveur FTP
SCENARIO DU TEST	
Exécution de l'application TestUnitaire Choix 4	
RESULTAT(S) ATTENDU(S)	
Les images contenues dans le serveur FTP sont envoyées vers la raspberry	

REALISATION DU TEST	
RESULTAT(S) OBTENU(S)	
	
EVALUATION DU RESULTAT OBTENU	
Satisfaisant	Erreur
	Critique  Majeure  Mineure 





TEST N° : 5		AUTEUR : Alamar Valerian		
IDENTIFICATION DU TEST				
DATE	OBJECTIF		NIVEAU	
17/05/21	Test de la récupération de l'emplacement de la trame du capteur de température		TU	x
			TI	
			TV	

DESCRIPTION DU TEST A REALISER	
CONTEXTE	
Environnement du test	Cible du test
Python 3.0, Linux Debian	CU : Récupérer l'emplacement ou est envoyée la trame du capteur température
SCENARIO DU TEST	
Exécution de l'application TestUnitaire Choix 5	
RESULTAT(S) ATTENDU(S)	
L'emplacement du fichier contenant la trame du capteur de température est retournée	

REALISATION DU TEST	
RESULTAT(S) OBTENU(S)	
<pre> Quel est votre choix ? 5 /sys/bus/w1/devices/00-7c0000000000/w1_slave </pre>	
EVALUATION DU RESULTAT OBTENU	
Satisfaisant	Erreur
	Critique  Majeure  Mineure 

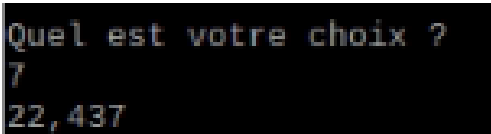



TEST N° : 6		AUTEUR : Alamar Valerian		
IDENTIFICATION DU TEST				
DATE	OBJECTIF		NIVEAU	
17/05/21	Test de la récupération de la trame de température		TU	x
			TI	
			TV	

DESCRIPTION DU TEST A REALISER	
CONTEXTE	
Environnement du test	Cible du test
Python 3.0, Linux Debian	CU : récupération de la trame de température
SCENARIO DU TEST	
Exécution de l'application TestUnitaire Choix 6	
RESULTAT(S) ATTENDU(S)	
La trame contenant la température est récupérée	

REALISATION DU TEST	
RESULTAT(S) OBTENU(S)	
<pre> Quel est votre choix ? 6 67 01 4b 46 4f ff 09 10 3b : crc=3b YES 67 01 4b 46 4f ff 09 10 3b t=22437 </pre>	
EVALUATION DU RESULTAT OBTENU	
Satisfaisant	Erreur
	Critique  Majeure  Mineure 

TEST N° : 7		AUTEUR : Alamar Valerian		
IDENTIFICATION DU TEST				
DATE	OBJECTIF		NIVEAU	
17/05/21	Test de la récupération de la valeur de température		TU	x
			TI	
			TV	

DESCRIPTION DU TEST A REALISER	
CONTEXTE	
Environnement du test	Cible du test
Python 3.0, Linux Debian	CU : récupération de la valeur de température
SCENARIO DU TEST	
Exécution de l'application TestUnitaire Choix 7	
RESULTAT(S) ATTENDU(S)	
La valeur de température est récupérée	

REALISATION DU TEST	
RESULTAT(S) OBTENU(S)	
	
EVALUATION DU RESULTAT OBTENU	
Satisfaisant	Erreur
	Critique  Majeure  Mineure 