## 0 - Table des matières

[0 - Table des matières 1](#_Toc39423494)

[1 – Introduction 2](#_Toc39423495)

[2 – Mon Rôle Dans Le Projet 2](#_Toc39423496)

[3 – Notre Réaction au Covid-19 5](#_Toc39423497)

[4 - Installation Physique Raspberry 6](#_Toc39423498)

[5 – Installation Logiciel Raspberry 8](#_Toc39423499)

[6 - Configuration Raspberry 10](#_Toc39423500)

[6.1 - Configuration du SSH 10](#_Toc39423501)

[7 – Installation Logiciels et Modules 11](#_Toc39423502)

[7.1 – Installation de Python 11](#_Toc39423503)

[7.2 – Installation du module MySQL 11](#_Toc39423504)

[7.3 – Installation du module Raspiomix 15](#_Toc39423505)

[8 – Présentation Du Code 16](#_Toc39423506)

[8.1 – Code général 16](#_Toc39423507)

[8.2 – Calcul Température 16](#_Toc39423508)

[8.3 – Partie Buzzer 16](#_Toc39423509)

[8.4 – Envoi Email 19](#_Toc39423510)

[8.5 – Affichage Graphique Données 20](#_Toc39423511)

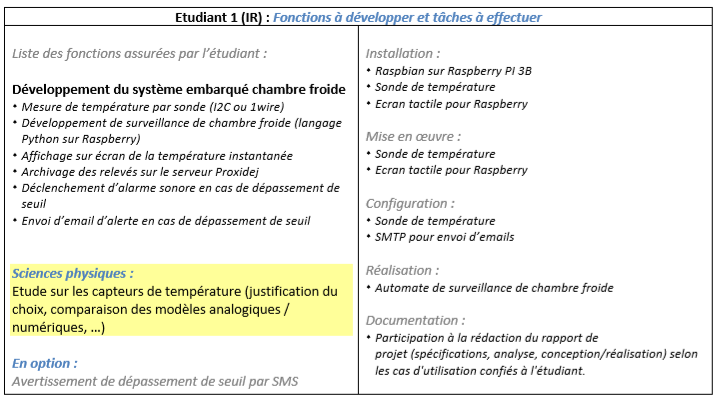
[8.6 – Envoi des données sur la base de données 21](#_Toc39423512)

## 1 – Introduction

Ce document permet de montrer le travail que j’ai réalisé au sein du projet ainsi que spécifier les choix, analyses personnelles m’impliquant dans ce projet.

Nous verrons donc en premier lieu des spécifications et analyses de mon rôle au sein du projet et en seconde partie ce que j’ai et comment je l’ai réalisé.

## 2 – Mon Rôle Dans Le Projet

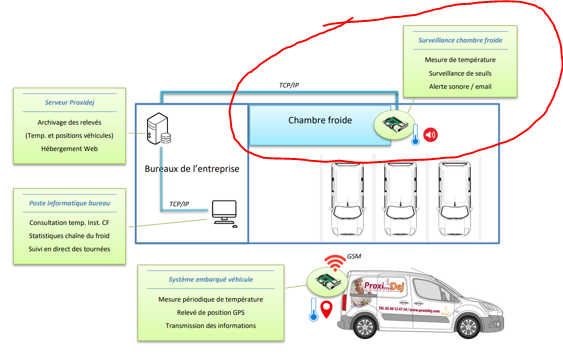


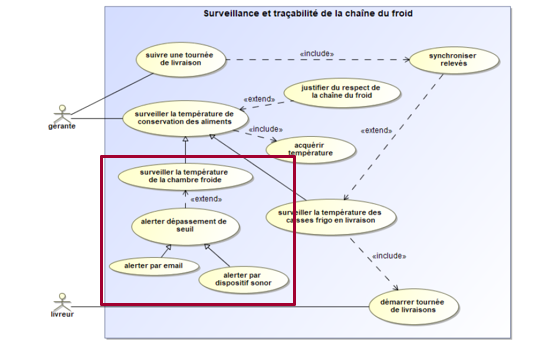
Au sein de ce projet mon rôle sera celui de réaliser sur un Raspberry un système automatique de surveillance de chambre froide.

Pour que le projet soit complet il faut que ce système puisse, en premier lieu, mesurer la température de la chambre froide ainsi qu’envoyer les données actuelles sur une base de données MariaDB pour pouvoir effectuer la traçabilité de la chaine du froid au sein de la chambre froide.

Ce système devra prendre en compte aussi l’affichage en temps réelles des récentes températures de la chambre froide et prendre en compte les dépassements des seuils autorisés en envoyant un mail au/à la gérant/e de la chambre froide tout en émettant un signal sonore pour prévenir les personnes autour de la chambre froide.

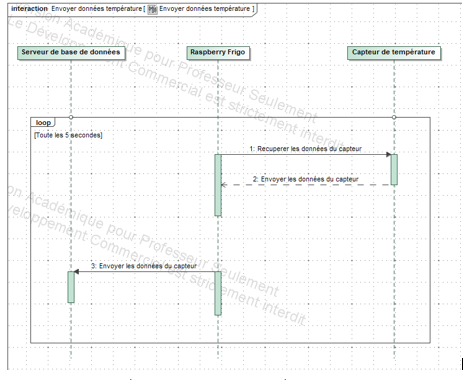
Voici quelques analyses de ma partie perso dans ce projet :



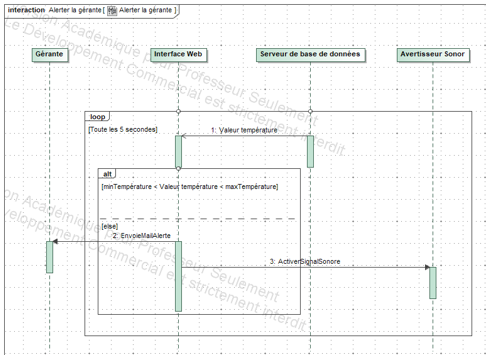


Et ensuite des diagrammes de séquence indiquant de quelle manière coder le programme qui m’a été attribué à réaliser :

En premier lieu comment récupérer la température avec la sonde :



Ensuite sur comment faire fonctionner le buzzer :

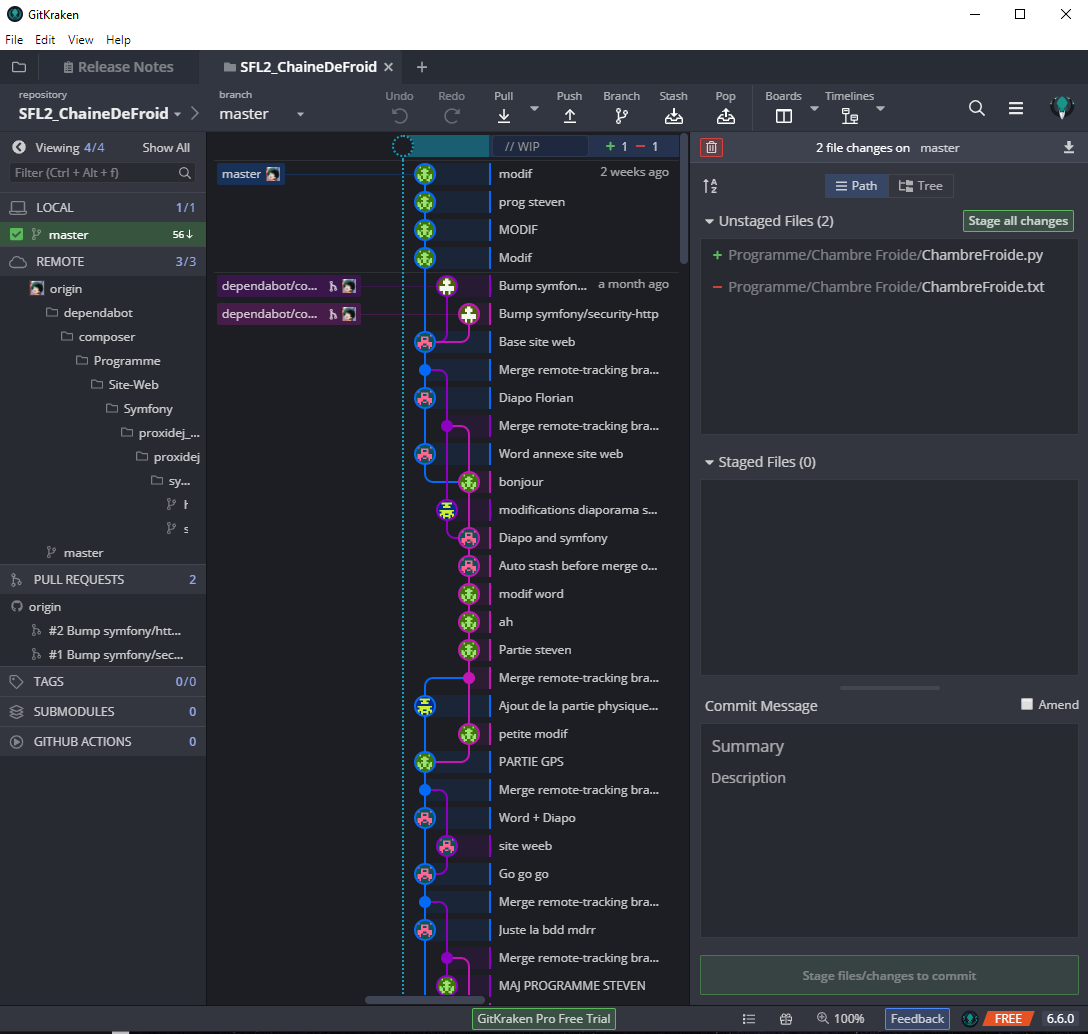


## 3 – Notre Réaction au Covid-19

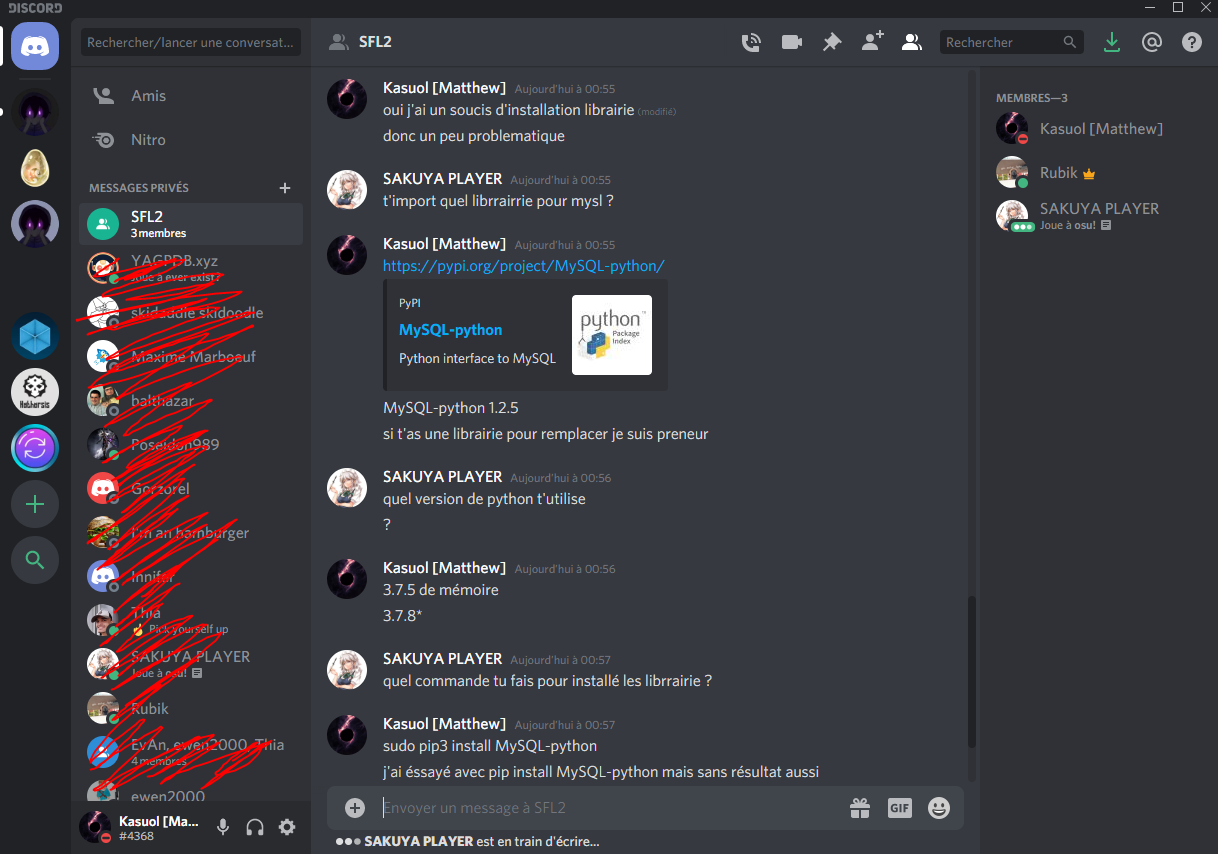
Au moment où ce texte est écrit nous sommes en pleine période de confinement depuis plusieurs semaines, afin que le projet soit toujours maintenu et réalisé nous avons dû mettre, avec mon groupe, des moyens de communications unique a ce genre de situation.

En premier lieu nous continuons à faire avancer le projet depuis chez nous en ligne en partageant grâce à Github nos avancées sur celui-ci

Github nous permet de toujours mettre le projet général a jour et se tenir informer sur les derniers ajouts réalisés par mes collègues et moi-même.

Nous avons utilisé GitKraken en tant que logiciel pour travailler sur notre ensemble Github. 

Pour permettre une communication textuelle ou vocale plus simplifié nous avons décidé de mettre en place un groupe Discord (application VoIP : Voix sur Ip) nous permettant de partager des informations sur le projet ou de s’entraider

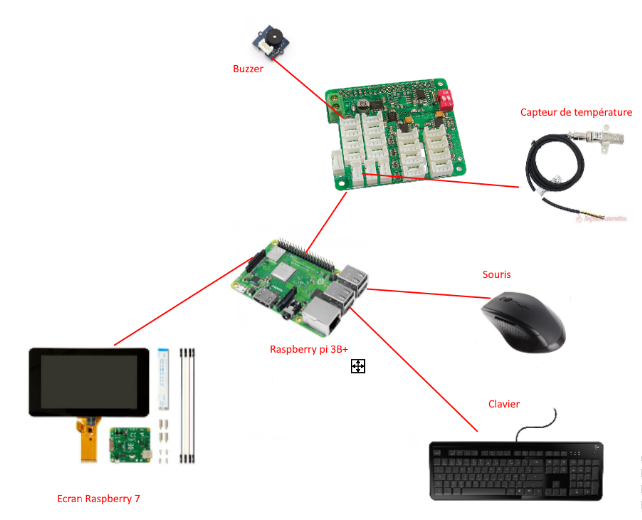


## 4 - Installation Physique Raspberry

Dans cette partie je vais détailler l’installation générale de la Raspberry ainsi que tout ce qui y est connecté.

Pour voir en détail les caractéristiques de chaque périphérique je vous invite à regarder la partie commune ou tout y est décrit.

Voici en premier lieu un plan simplifié de l’installation générale :



Et en réel :

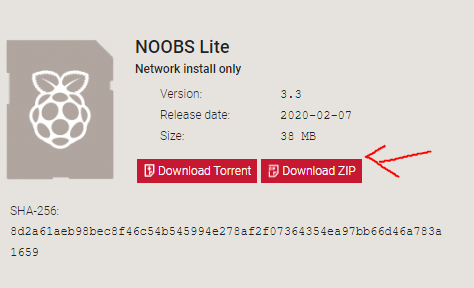
Une image contenant clavier, ordinateur, machine à écrire, table

Description générée automatiquement

## 5 – Installation Logiciel Raspberry

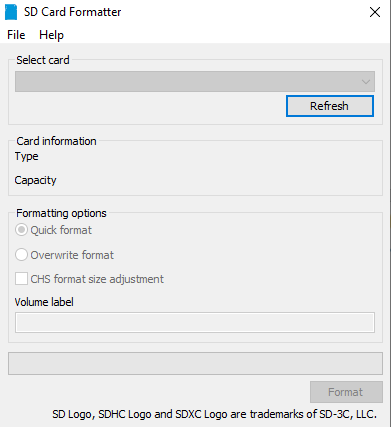
En premier lieu on cherche à télécharger NOOBS un installeur de système d’exploitation Raspberry.

On décide de télécharger la version Lite car elle pèse moins lourd à installer sur la carte SD Et elle nous permettra d’installer directement la dernière version de Raspbian a l’aide d’une connexion internet (dans notre cas : une connexion Wifi).

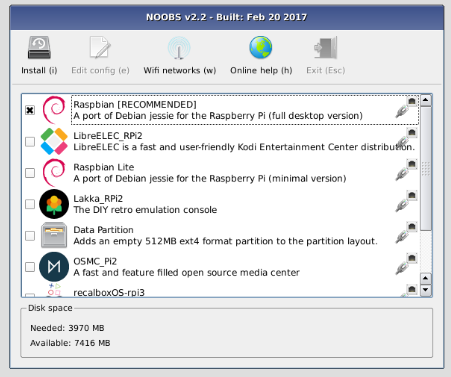


Il nous suffit juste d’utiliser un formateur de carte SD afin de réinitialiser la carte que nous allons utiliser et la mettre au format FAT32 pour ensuite décompresser l’archive NOOBS que nous venons de télécharger à l’intérieur de la carte SD.

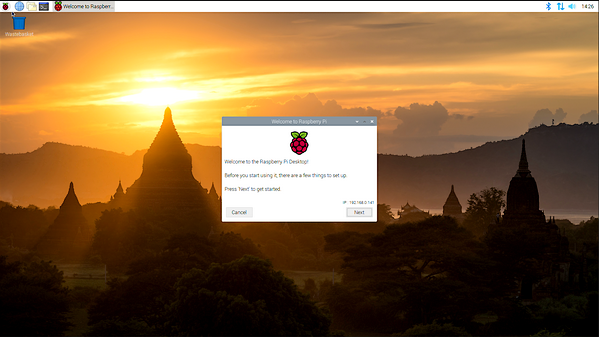
Pour formatter la Carte SD j’ai utilisé un logiciel appelé « SD Card Formatter »



On rentre ensuite la carte SD avec NOOBS à l’intérieur et nous arrivons sur cette interface :



Je confirme le choix sur Raspbian et appuie sur install, après une installation et quelques configurations de base comme l’utilisateur et le mot de passe



Notre Raspberry est donc bien installé et prête à être utilisée !

## 6 - Configuration Raspberry

### 6.1 - Configuration du SSH

J’ai commencé par activer le SSH sur la Raspberry afin de compléter ce rapport avec des captures d’écran et afin de faciliter mon travail en ayant pas à installer une deuxième station de travail chez moi avec un écran, souris clavier...

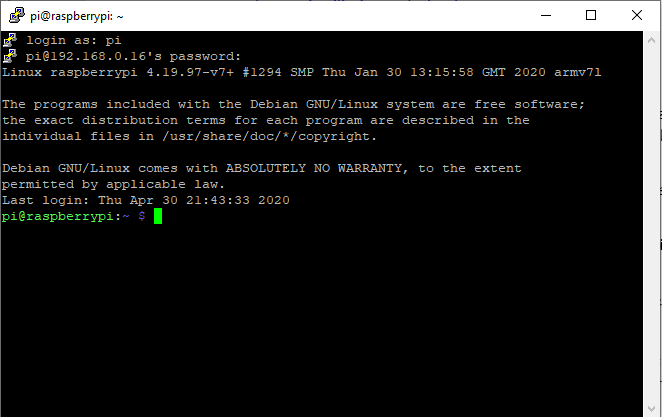
Avec La Version de Raspbian que j’ai installé, l’interface graphique nous permet d’activer directement cette option dans les paramètres.

Nous allons donc dans **Préférences > Configuration du Raspberry Pi > Interfaces > SSH > Activé.**

Le Raspberry est connecté sur le même réseau que mon ordinateur principal en wifi, par conséquent j’utilises Putty afin de me connecter sur le Raspberry. Je rentre tous les paramètres du Raspberry :

(On obtient l’IP en rentrant dans le terminal sur le Raspberry pi la commande : **hostname -I**

**Adresse IP : 192.168.0.16 | Port : 22 (SSH)**



**Login : Pi | Password : Erperr@6**

Nous avons donc bien configuré le SSH et il est prêt à être utilisé.

**A NOTER que l’adresse IP sera changé par une adresse IP FIXE pour le client**

## 7 – Installation Logiciels et Modules

### 7.1 – Installation de Python

Par défaut **Python** est déjà installé sur la Raspberry avec un IDE s’appelant **Thonny Python IDE**, il correspond à la version **3.7.3** de **Python**, **version qui nous convient pour les modules à installer par la suite**, il ne sera donc pas nécessaire d’installer une version différente de Python

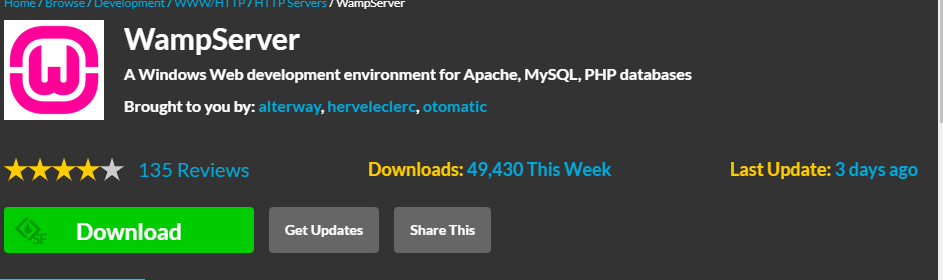
### 7.2 – Installation du module MySQL

Due aux conditions actuelles liées au Covid-19 il nous est très compliqué d’accéder à une base de données avec lequel chacun des membres de notre groupe puissent travailler

Par conséquent J’ai décidé de créer une base de données à domicile, locale sur mon ordinateur principal ou tous les tests seront effectuer, cette base de données MariaDB possède la même structure que la base de données réalisé au sein du Campus Saint Félix La Salle, par conséquent il n’y aura qu’à modifier l’adresse IP auxquels les données sont envoyés pour pouvoir réaliser les opérations sur la base de données finales

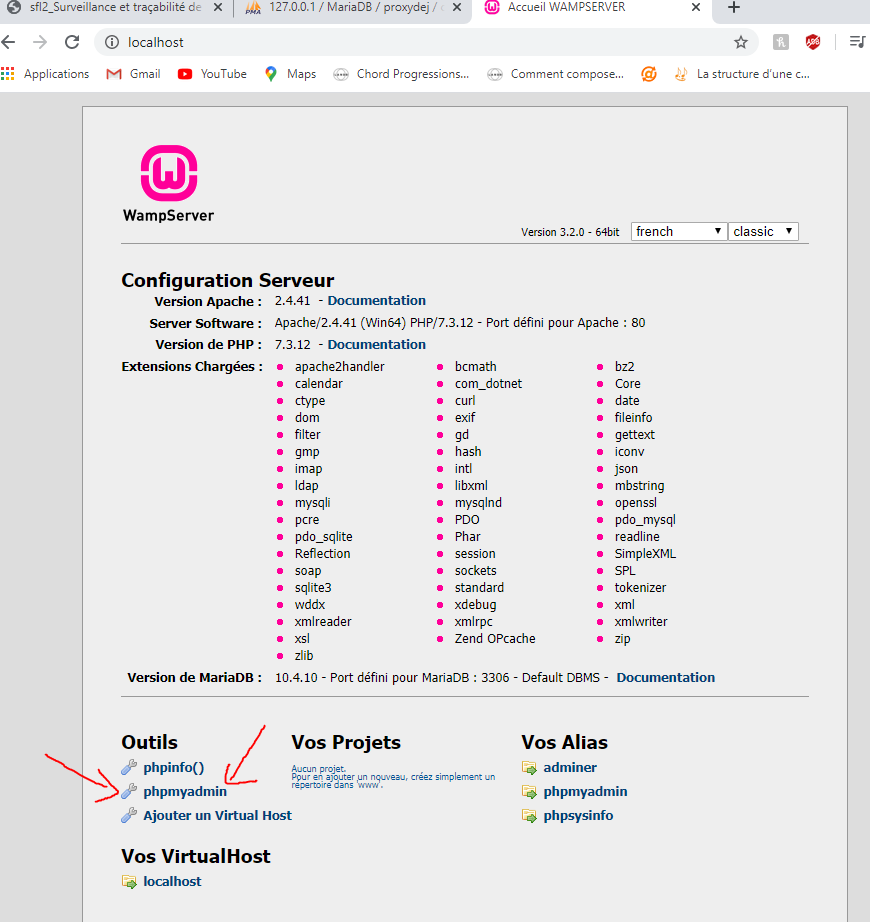
**Ci-dessous la réalisation de l’installation de la base de données :**

En premier je dois installer sur mon pc principal WampServer qui est une plateforme de développement web nous permettant de faire fonctionner localement des scripts php.

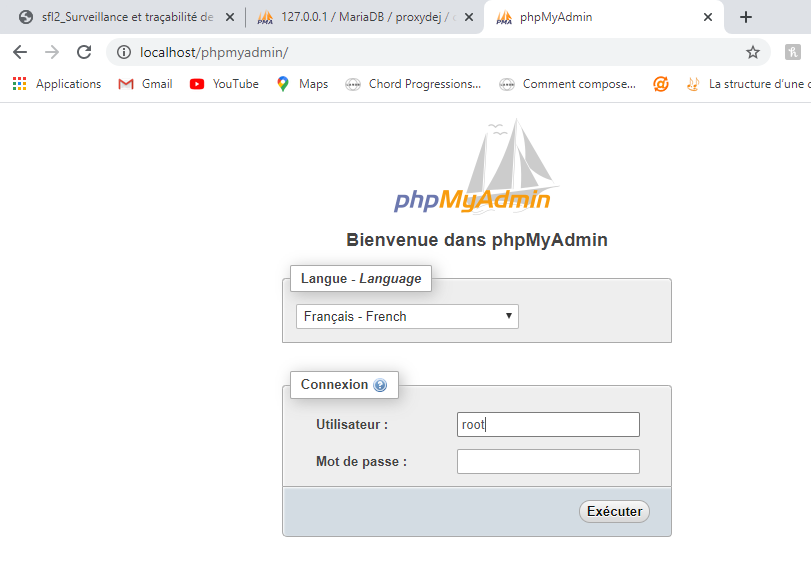


Une fois installer on lance WampServer et mettons 127.0.0.1 dans la barre de recherche internet correspondant à notre localhost.

On clique par la suite sur phpMyAdmin afin de créer sa base de données



On se connecte avec l’utilisateur root et sans mot de passe.

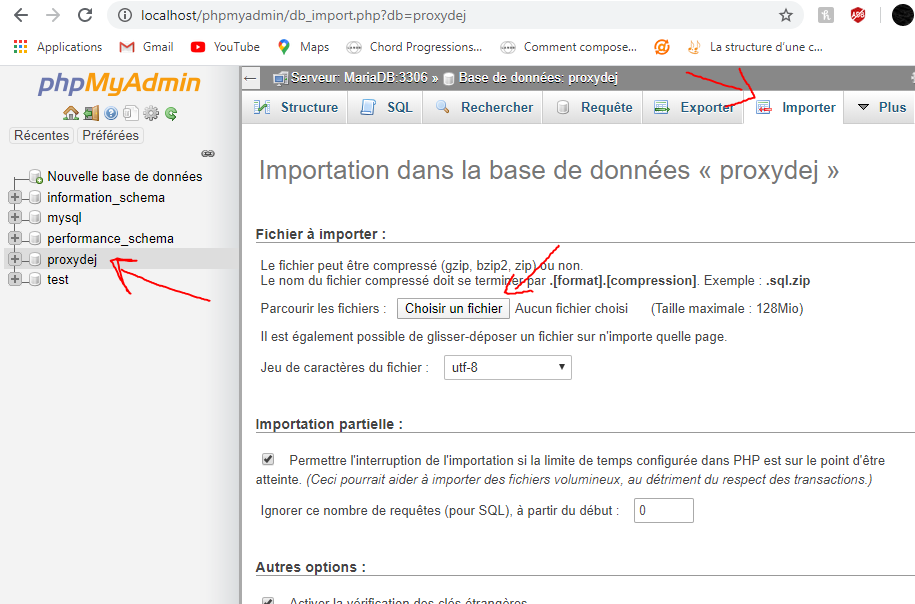


Une fois connecter on crée une base de données que l’on appelle proxydej, qu’on utilisera par la suite pour se connecter dessus avec python

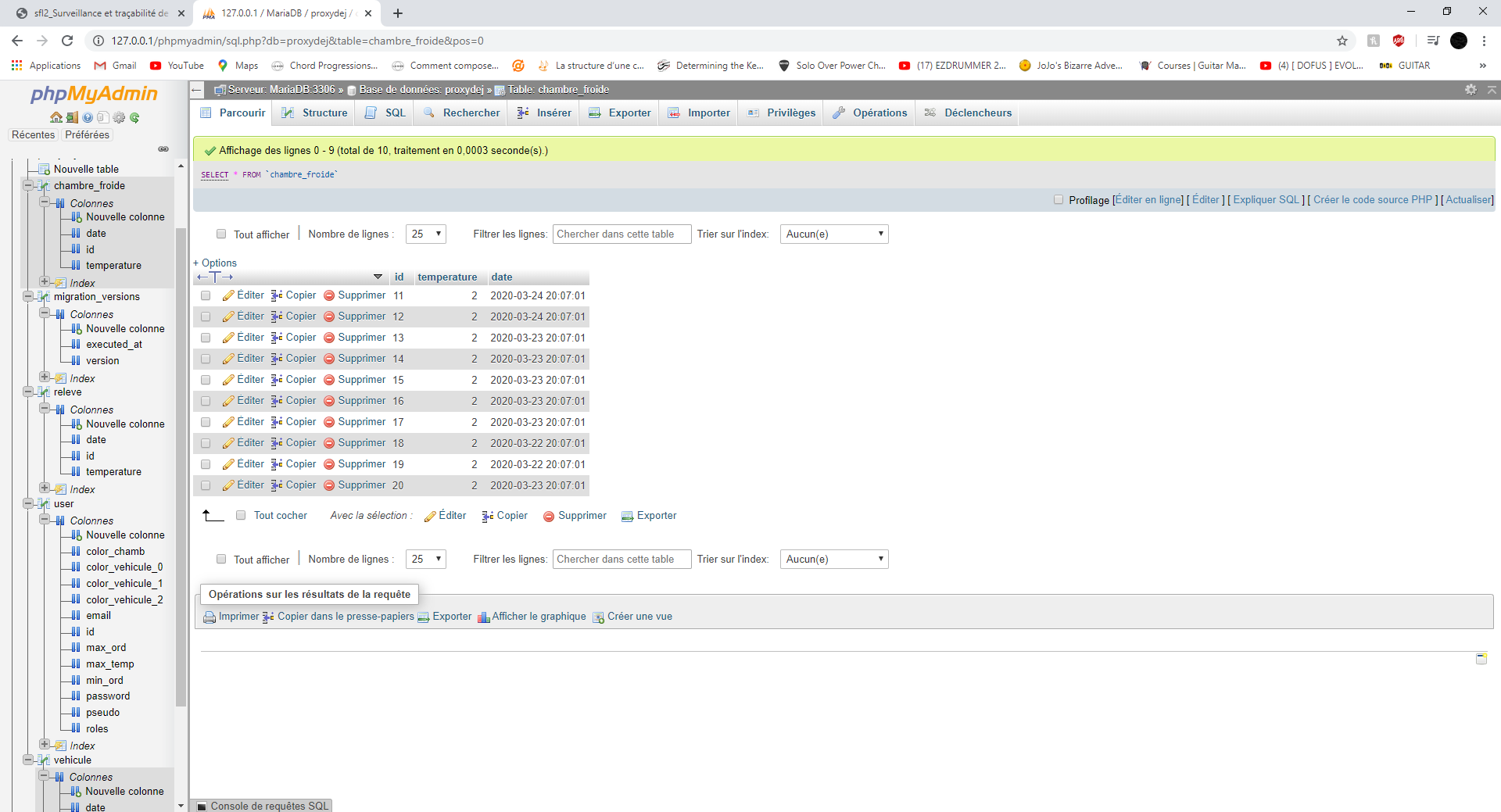
Pour toutes les tables et ce qui constitue la base de données en général j’ai repris la même structure que celle crée par Florian



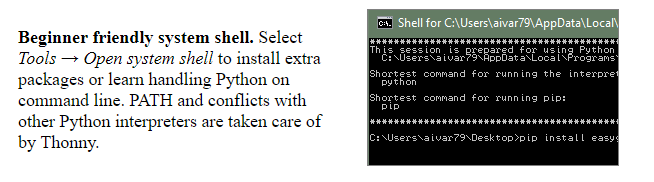
Pour pouvoir l’appliquer à ma base de données locale je dois sélectionner ma base de données, cliquer sur importer, et lier le fichier dans les fichiers à importer.



Après un court instant la base de données mariaDB a bien été copiée et je peux travailler localement dessus



Pour pouvoir réaliser des opérations via python sur cette base de données nous devons en complément installer le module nécessaire à la communication entre mon Raspberry et la base de données :

Pour cela **Thonny Python IDE** nous est très pratique, car il possède une interface dédiée à l’installation de modules externes

Avant cela, nous devons télécharger sur notre Raspberry **le MySQL Python package** pour que le module que nous allons installer fonctionne.

Pour cela nous utilisons la commande dans le terminal : **sudo pip3 install MySQL-Python**

### 7.3 – Installation du module Raspiomix

Nous utilisons un RaspioMix+ avec notre Raspberry afin de connecter les capteurs à l’aide de connecteur spécifiques appelés Grove nous permettant de faciliter la gestion des capteurs et pouvoir les manipuler sans les endommager

On installe les deux dépendances nécessaires pour que la librairie python Raspiomix fonctionne :

**Rpi.GPIO et smbus :** que l’on obtient en faisant : **sudo pip install RPi.GPIO smbus**

Ensuite il nous suffit de récupérer la libraire python raspiomix sur le site :

[**https://raw.githubusercontent.com/hugokernel/RaspiOMix/master/src/python/raspiomix.py**](https://raw.githubusercontent.com/hugokernel/RaspiOMix/master/src/python/raspiomix.py)

Et l’inclure dans un fichier que l’on appellera **raspiomix.py**, ou l’on acédera dans le code à l’aide de : **from raspiomix import Raspiomix**

**Pour récupérer la température de notre sonde nous aurons besoin d’utiliser le bus de données I2C**

**Malheureusement la sonde de température que nous utilisons n’utilises qu’un système Grove, non conçu pour cet usage a la base, n’étant pas i2C nous ne pouvons pas récupérer des données**

## 8 – Présentation Du Code

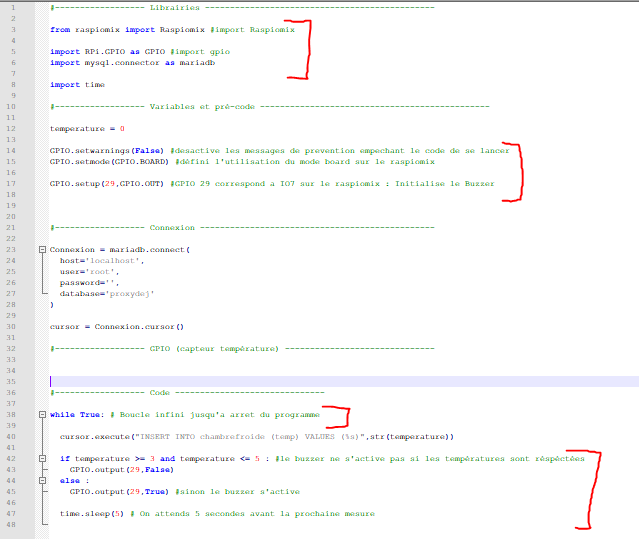
### 8.1 – Code général

### 8.2 – Calcul Température

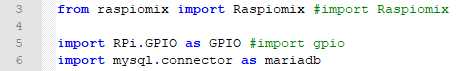
**Malheureusement le calcul de température par bus I2C est compliqué car j’ai resoudé la sonde de base pour la mettre sur un ensemble Grove, cependant il ne semble pas être compatible i2c, nous ne pouvons pas récupérer de données, des recherches et solutions sont en cours de réalisation**

### 8.3 – Partie Buzzer

La Partie du code Buzzer est fonctionnel et correspond à cette partie du programme :

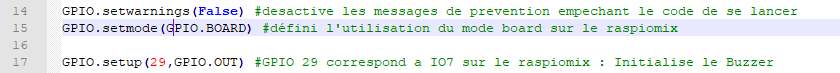


**Librairies :**

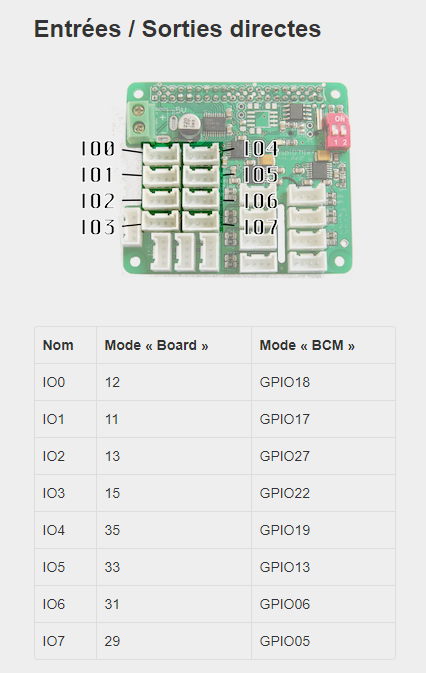


* La ligne 3 permet de récupérer la librairie du fichier Raspiomix.py situé sur le bureau de la Raspberry
* La ligne 5 permet d’utiliser la librairie précédemment installée **RPi.GPIO** ayant pour but de contrôler les sorties et entrées de la raspiomix, de définir leurs états.

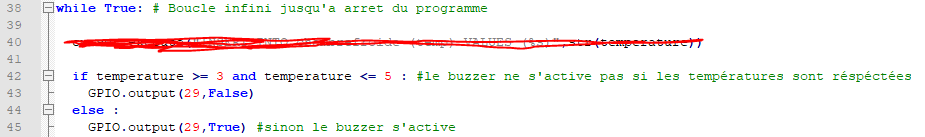
**Variables et pré-code :**



* La ligne 14 permet de désactiver les messages de prévention, lors du démarrage du programme un message empêche le programme de démarrer avec un avertissement, cette ligne permet de supprimer cet avertissement
* Le Raspberry Pi autorise deux numérotations : celle de la sérigraphie du connecteur de la carte (GPIO.BOARD), ou la numérotation électronique de la puce (GPIO.BCM), La ligne 15 nous permet de choisir le mode BOARD.
* La ligne 17 permet d’initialiser le buzzer en **mode Sortie** (émet un son) au **BOARD** **29** qui correspond à **IO7** selon la documentation raspiomix :



**Le Code :**



* La ligne 38 fait que le reste du code se répétera jusqu’à l’arrêt du programme
* La ligne 42 à 43 vérifie si la température actuelle correspond à ce qui est demandé, dans ce cas le buzzer ne fait aucun bruit
* Dans le cas contraire la ligne 44 à 45 permet d’activer le buzzer si les conditions ne sont pas respectées

### 8.4 – Envoi Email

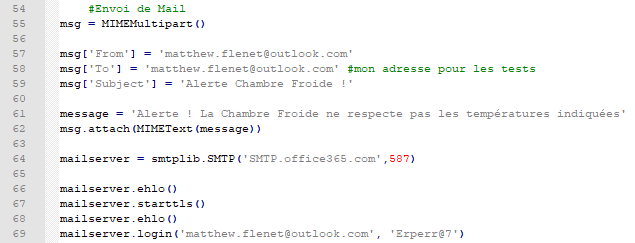
Voici la partie du code correspondant à l’envoi d’email :

**Librairies :**



* La ligne 18 et 19 est l’import de variantes de la libraires Email.mime qui permet de créer des messages à envoyer par mail
* La ligne 20 est l’import de la bibliotheque inclue dans l’installation de python de smtlib, permettant par la suite de créer une connexion SMTP

**Variables et pré-code :**

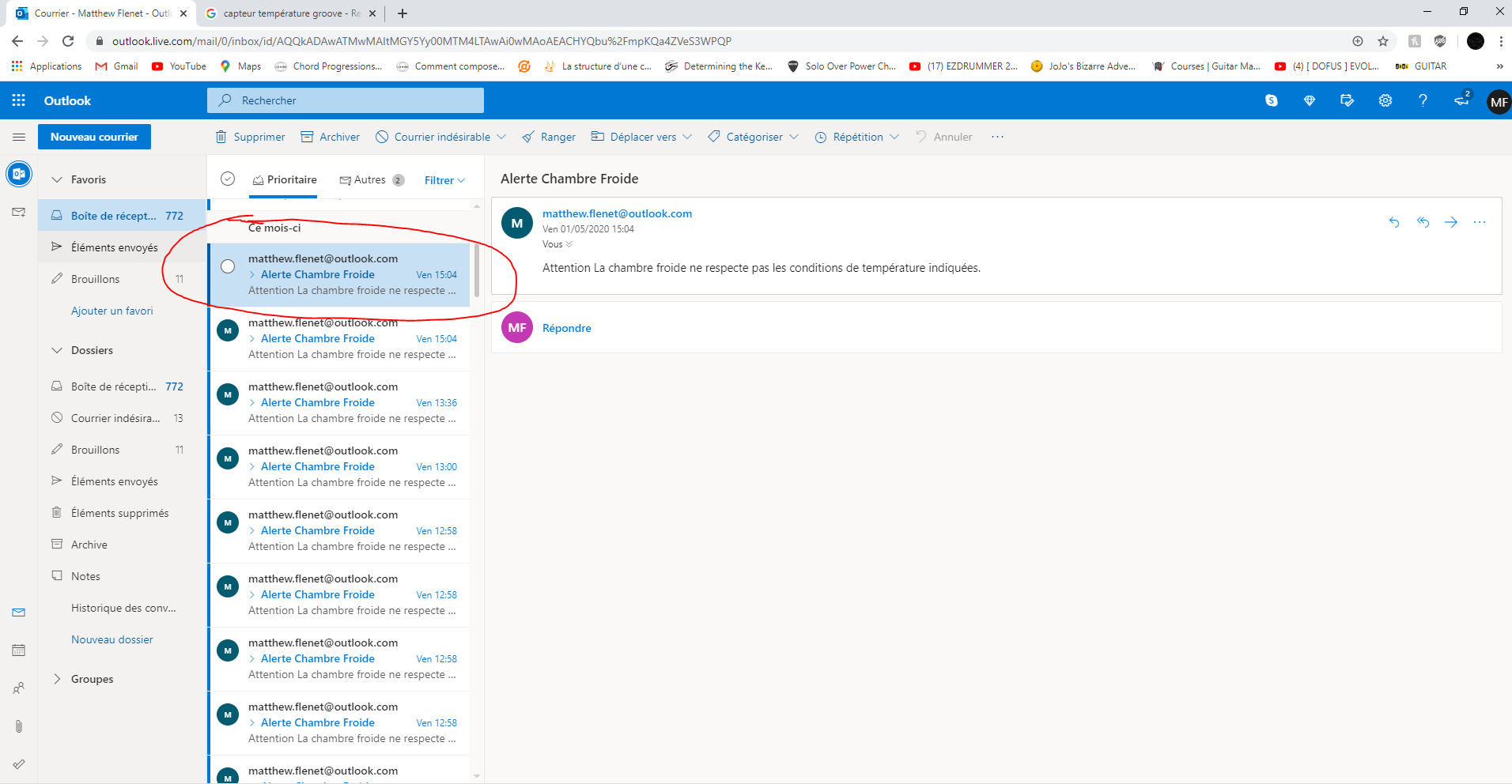


* La ligne 55 permet de créer la structure du message à envoyer, l’enveloppe
* La ligne 57 à 59 permet de définir l’expéditeur, le récepteur du mail et le sujet du mail à envoyer
* La ligne 61 permet de définir le message en lui-même qui sera envoyé à l’adresse mail choisie
* La Ligne 62 permet de mettre le message dans sa structure, son enveloppe
* La ligne 64 permet la connexion au serveur SMTP office car nous utilisions la messagerie Outlook pour envoyer notre message
* Les lignes 66 à 68 permettent la connexion au serveur SMTP choisi et l’encryptage des données a envoyé afin que si quelqu’un les réceptionne ils ne puissent pas les déchiffrer
* La ligne 69 permet de se connecter à l’adresse email qui va être utilisé afin d’envoyer le message

**Le Code :**



* Si les températures réelles ne correspondent pas à celles voulues un email est envoyé, ici a [matthew.flenet@outylook.com](mailto:matthew.flenet@outylook.com) (adresse à changer avec celle du/de la gérant/e) en envoyant le message sous forme de chaine de caractère

Ci-dessous un exemple de la partie du code en action lorsque les températures indiquées ne sont pas respectées :

### 8.5 – Affichage Graphique Données

Pour l’affichage des donnés on fait simplement un affichage des températures récentes dans le terminal, et on affichera seulement ce terminal une fois le programme lancé.

**Librairies :**



* Les seules librairies utilisées sont déjà inclues dans Python et ne demandent pas d’installation nécessaire, nous utiliserons seulement datetime qui n’est pas exclusif à l’affichage graphique des données

**Variables et pré-code :**



* La ligne 71 permet de créer un raccourci du moment actuel qui est reformaté sous la forme jour-mois-année Heure Minute secondes

**Le Code :**



* La ligne 89 permet d’afficher à chaque mesure de température, le moment de la mesure et la température actuelle

Ci-dessous un exemple d’affichage des dernières températures récupérées :

Une image contenant texte

Description générée automatiquement

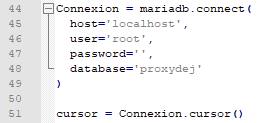
### 8.6 – Envoi des données sur la base de données

**Librairies :**



* La ligne 15 permet l’importation du connecteur python MySQL, dans ce cas ci on va utiliser les commandes pour mariaDB

**Variables et pré-code :**



* La ligne 44 à 49 permet de créer une connexion sur la base de données proxydej, avec l’utilisateur root et son mot de passe en localhost.
* La ligne 51 permet de créer un curseur, qui va effectuer une action, ici le curseur exécute la variable connexion qui permet de se connecter à notre base de données

**Le Code :**



* La ligne 82 permet donc d’utiliser le curseur afin de rentrer une commande mariaDB ou il sera mis dans la table chambre froide, la température actuelle calculée par la sonde de température.