Compil des tutos cours 2020 pour réaliser une Station météo

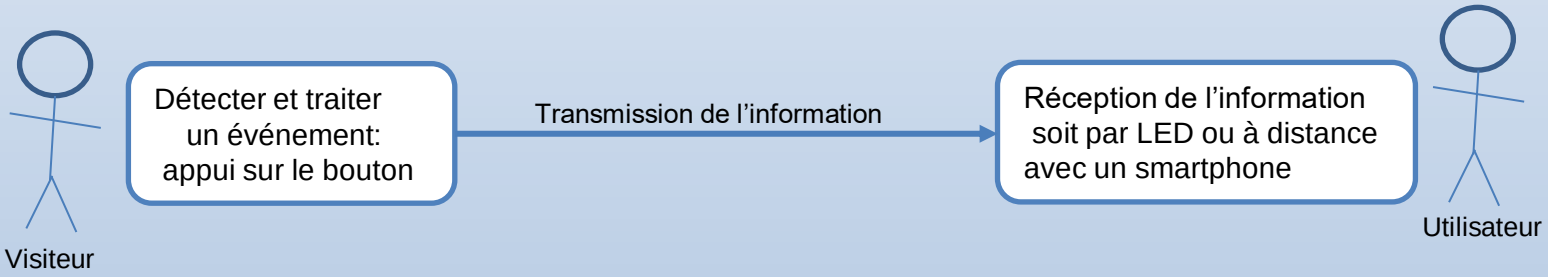
1. Demarche projet
2. Présentation des composants utilisés
3. Présentation Station météo
4. Installation des librairies
5. Ajout de la sonde DS18B20
6. Arborescence
7. Architecture de la Station météo
8. Codage Station météo
    - sonde.py
    - bme280.py
    - meteo.py
    - meteo_web.py
    - meteo_web_config.conf
    - meteo_web.html
9. Diagramme de fonctionnement
10. Annexe LibreOffice calc

Démarche projet

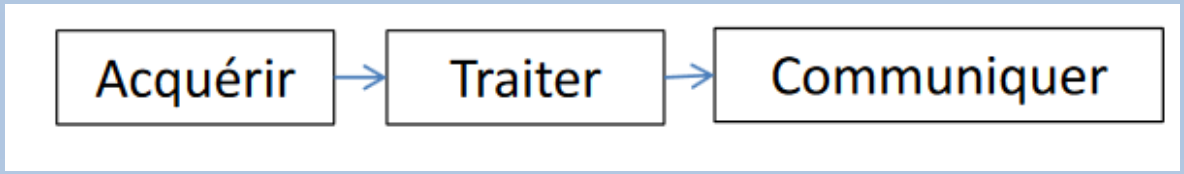
Description par un texte

Quand l'utilisateur du système n'est pas dans sa maison, il veut être averti de la présence de quelqu'un à son domicile qui actionne la sonnette.

Description par un schéma



Description sous forme fonctionnelle



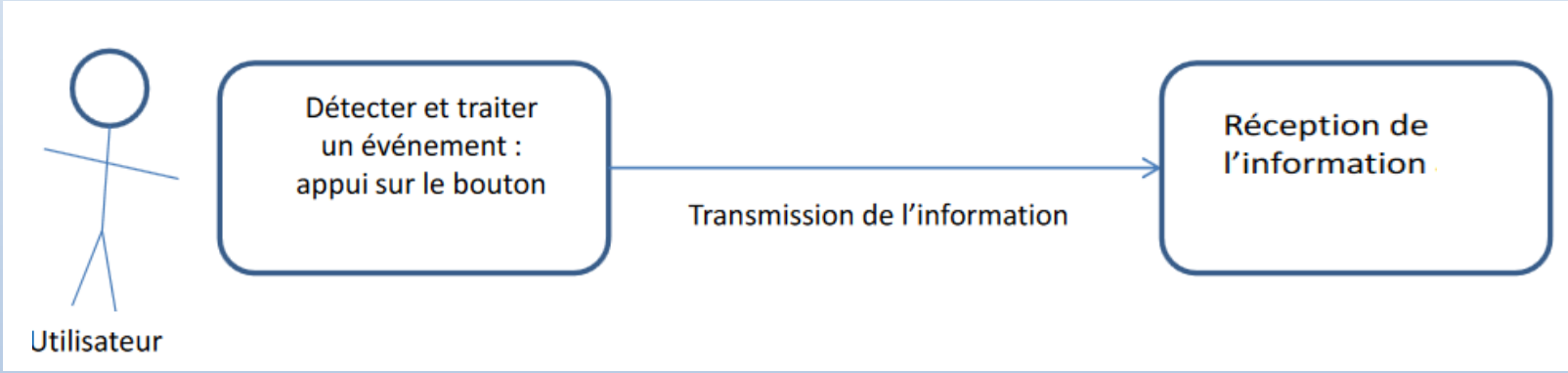
Démarche projet

Exemple TP 1 allumer LED à partir d'un bouton

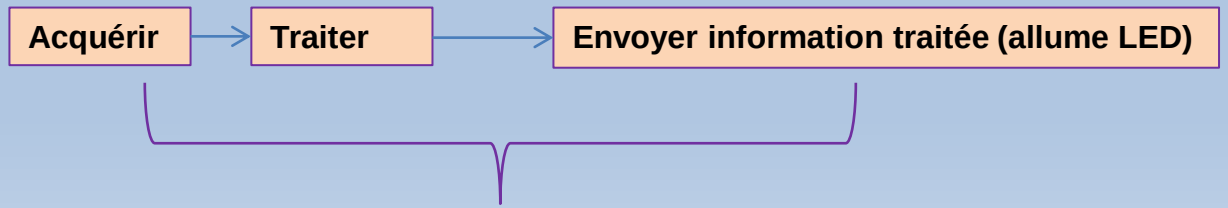
. Description textuelle du système

Quand l'utilisateur appui sur un bouton cela allume 3 LED de couleur successivement.

. Description par un schéma



. Description fonctionnelle



Architecture fonctionnelle

Démarche projet

. Description sous forme d'organigramme

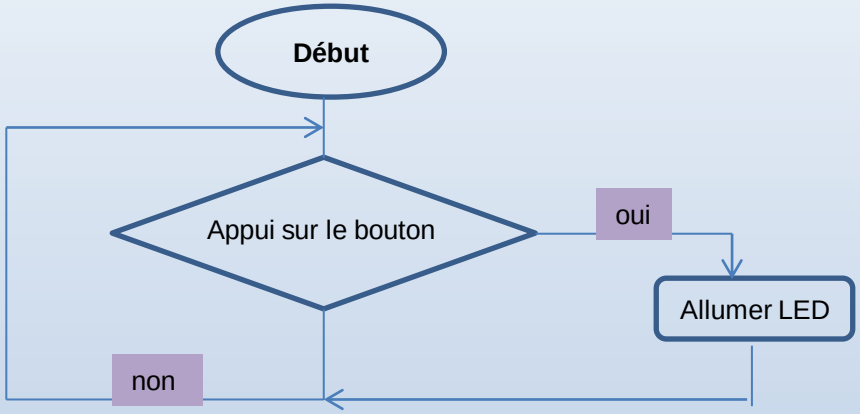
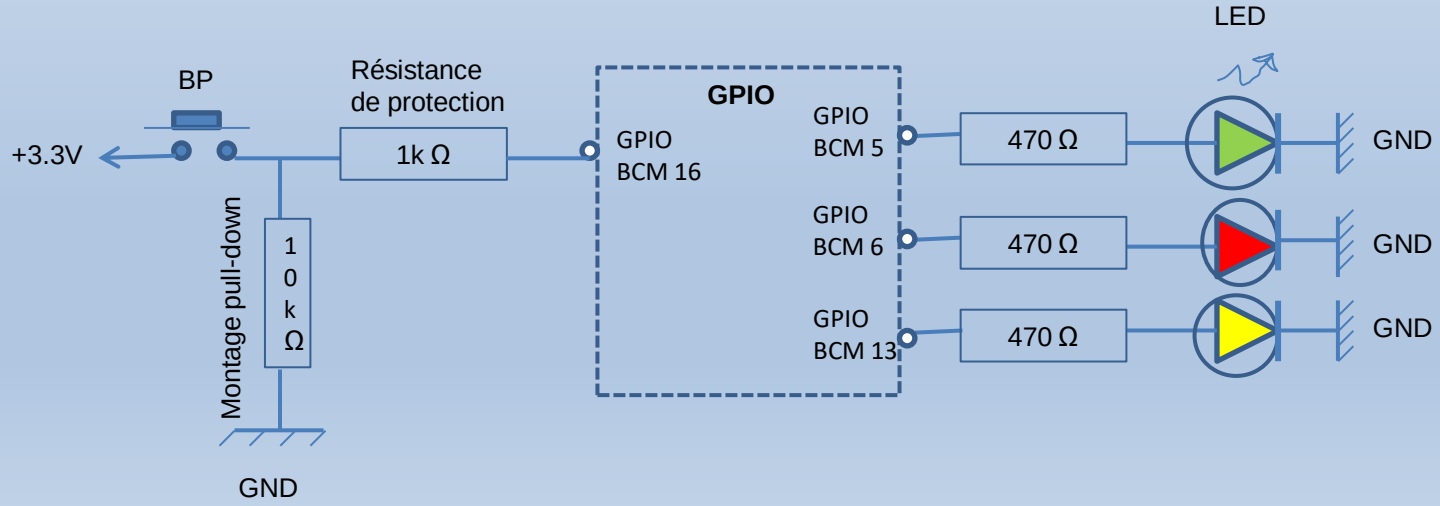


Schéma électrique



Démarche projet

Besoin matériels

Raspberry Pi

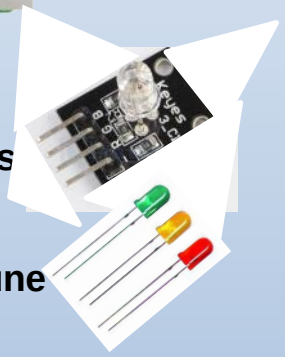
GPIO Pi

Carte µSD avec l'OS Raspbian

Module LED 3 couleurs

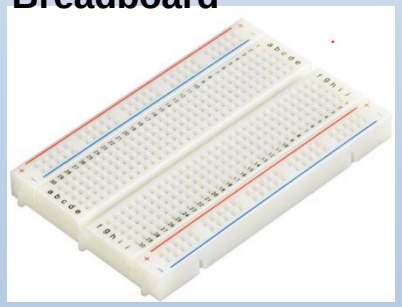
ou

LED Rouge, Verte, Jaune



Résistance

Breadboard



Module Bouton Poussoir

ou

Bouton Poussoir

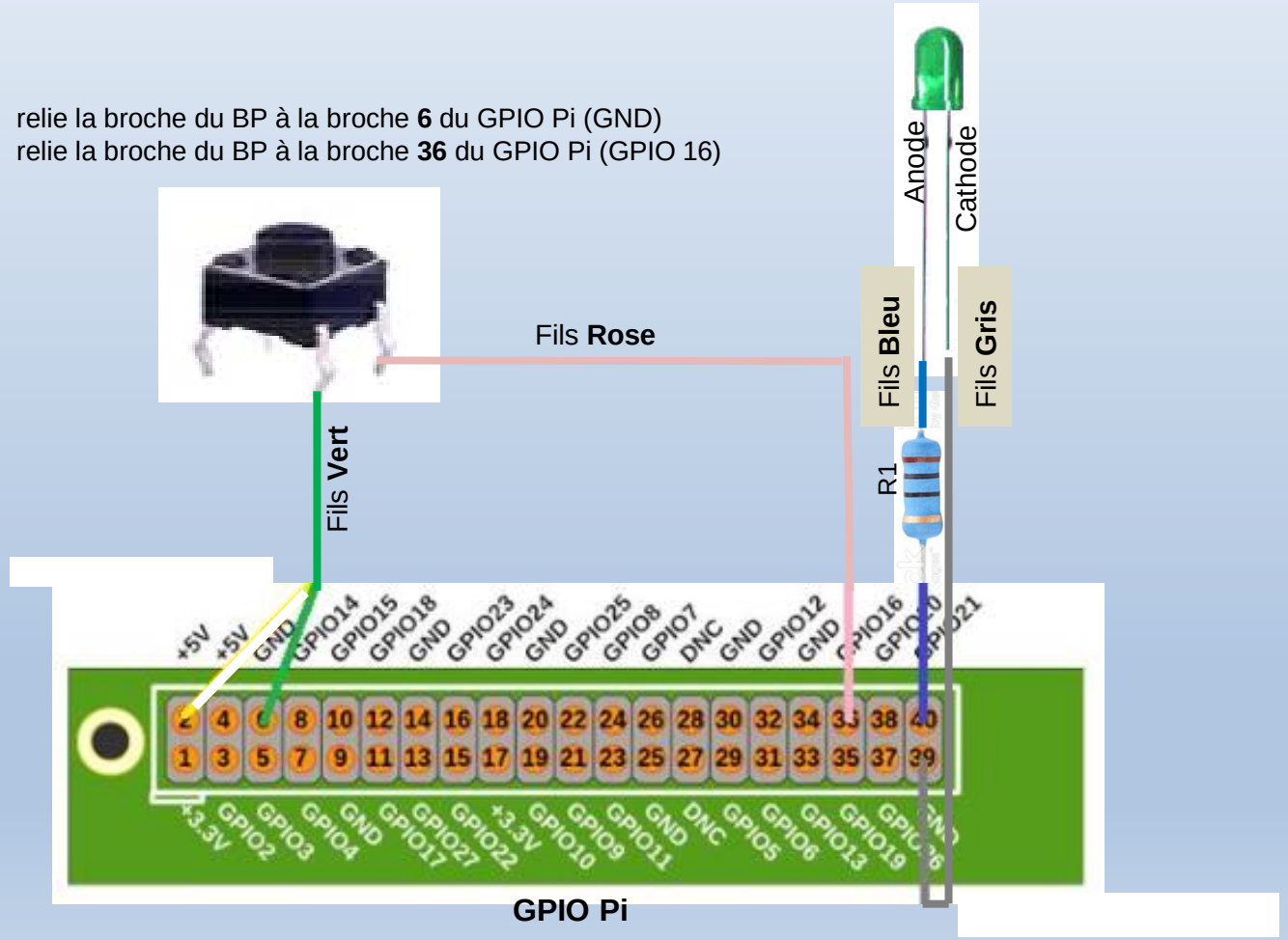
Fils : Bleu, Gris, Jaune, Vert, Rose

Démarche projet

Câblage de la maquette

Fils gris : relie la Cathode de la LED à la broche **39** du GPIO Pi (GND)
Fils bleu : relie l' Anode de la LED à la broche **40** du GPIO Pi à travers la résistance R1

Fils vert : relie la broche du BP à la broche **6** du GPIO Pi (GND)
Fils rose : relie la broche du BP à la broche **36** du GPIO Pi (GPIO 16)



GPIO Pi

Démarche projet

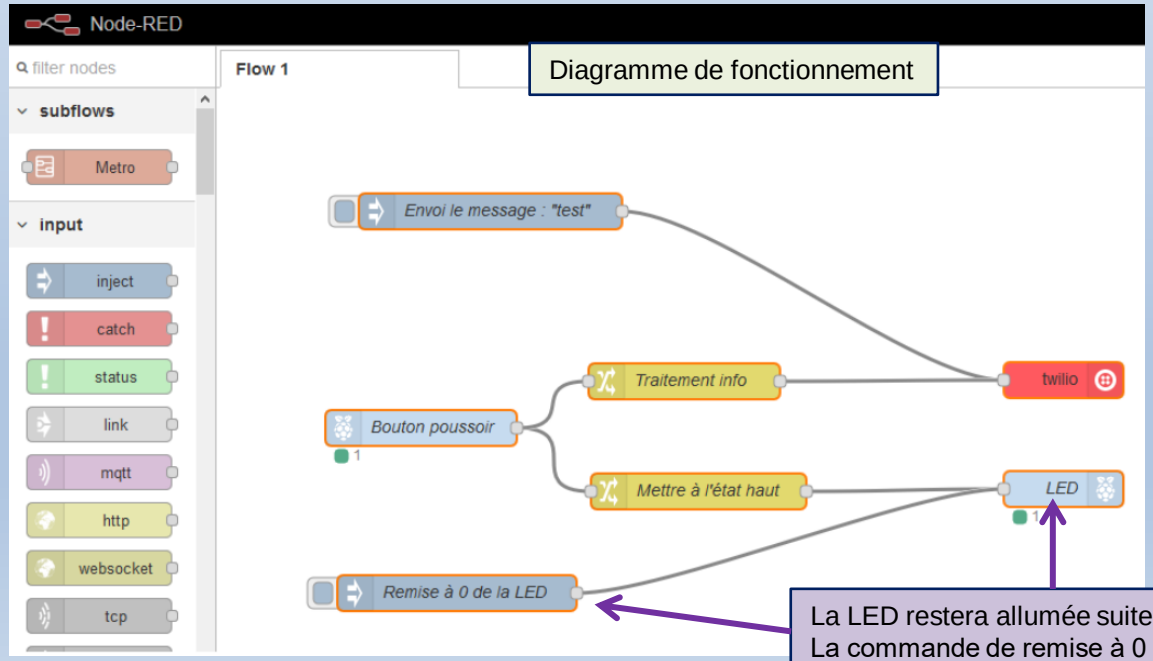
Partie Software

Choisir un langage

Programmer avec Python

Programmer avec Node-RED

Langage graphique de très haut niveau. Cf figure ci-dessous
Adapté pour les **iot**



Présentation des composants utilisés

A. Matériels

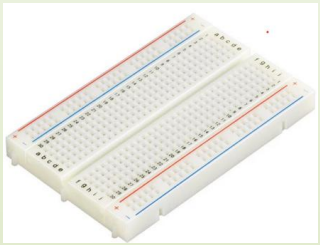
Raspberry Pi

GPIO Pi

Carte µSD avec l'OS Raspbian

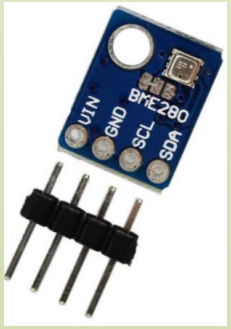
T Cobbler 40 pts

Breadboard



Bouton Poussoir

BME280



18B20

Buzzer Piezo

Fils de liaison

Câble en nappe

Résistances

4.7 KΩ

220Ω

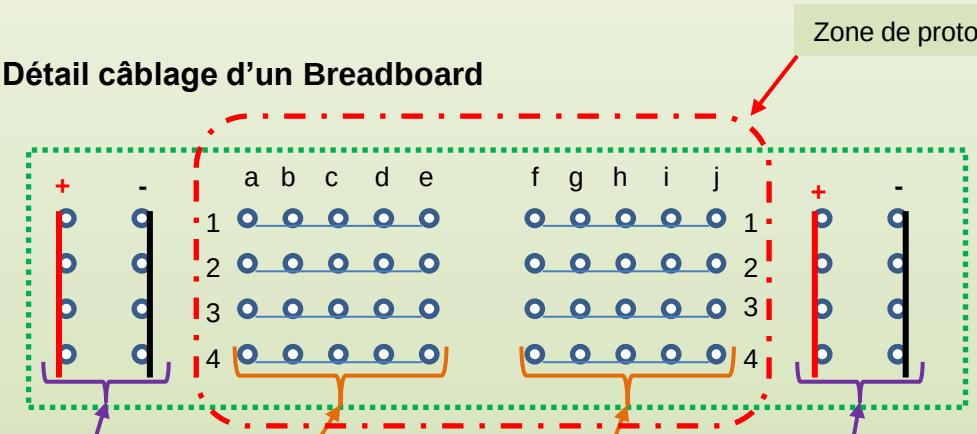
LED Rouge, Verte, Jaune

Présentation des composants utilisés

BREADBOARD

Le Breadboard permet de construire vos réalisations électrique sans faire de soudure (**prototypage**).
Ci-après, le détail électrique d'un Breadboard.

Détail câblage d'un Breadboard



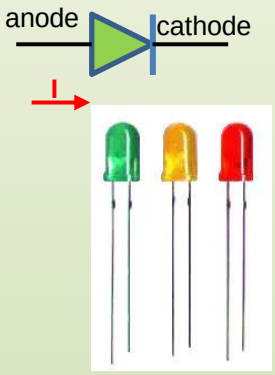
Les 5 trous horizontaux sont reliés électriquement via une bande métallique interne

Les colonnes verticales qui se situent de chaque côté du breadboard sont utilisées pour les alimentations
(+ et -)

Présentation des composants utilisés

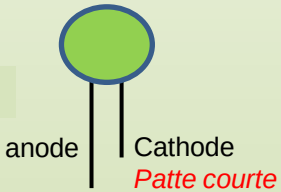
LED ou DEL (light-emitting diode ou diode électroluminescente

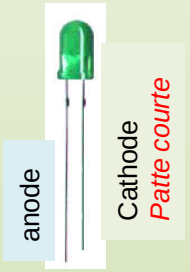
Une LED est un composant qui convertit l'énergie électrique en énergie lumineuse.
C'est un composant **polarisé,** c'est-à-dire que le courant électrique ne passe que dans un sens. Anode ➔Cathode
(moyen mnémotechnique sens du triangle de la base vers le sommet)



Représentation schématique

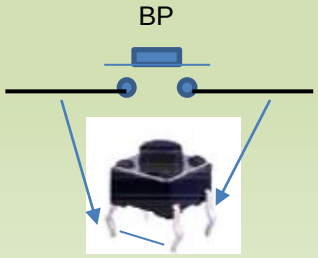
Représentation physique





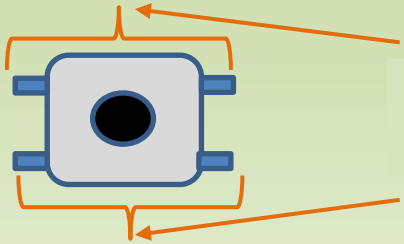
Interrupteur (bouton poussoir)

Un interrupteur interrompt le passage du courant électrique lorsqu'il est en position « ouvert »
Il existe plusieurs types d'interrupteurs (nous utilisons un bouton poussoir)



Représentation schématique

Représentation physique



Les 2 pattes sont reliées électriquement ensemble

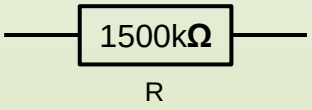
Présentation des composants utilisés

Résistance

Une résistance électrique est un composant qui a la propriété à s'opposer au passage d'un courant électrique (**i)** sous une tension électrique donnée.

Elle est identifiée par la lettre **R** et son unité de mesure est la lettre grecque *oméga* (symbole : **Ω**).

Représentation schématique



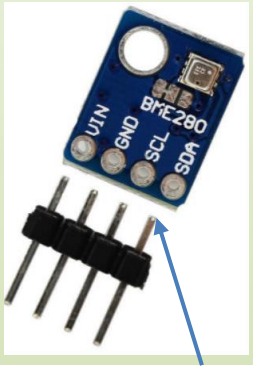
Représentation physique

Les bandes de couleurs permettent de connaître la valeur d'un résistance

Présentation des composants utilisés

Capteur BM280

Le module BME280 de Bosch est un module pour station météo accessible, fiable et très précis.
Le degré d'humidité est mesuré avec une marge de +/- 3% ;
La pression atmosphérique, avec une précision de +/- 1hPa
La température communiquée est précise au degré près.
Le capteur est soudé sur un circuit imprimé intégrant un régulateur 3,3 V et un registre à décalage.
Il peut communiquer avec une Raspberry via un bus i2c

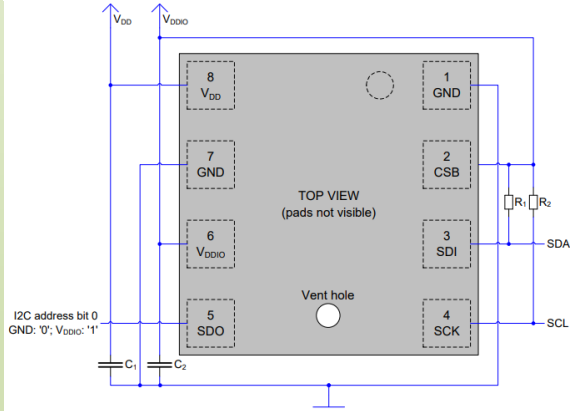


Souder les 4 broches courtes du connecteur au capteur

Après soudure

Nom	Fonction	GPIO
VIN	Alim	3.3V
GND	Masse	GND
SCL	Horloge	GPIO SCL
SDA	Données	GPIO SDA
@ i2c	**0x76**	

Diagramme de connexion du bus i2C

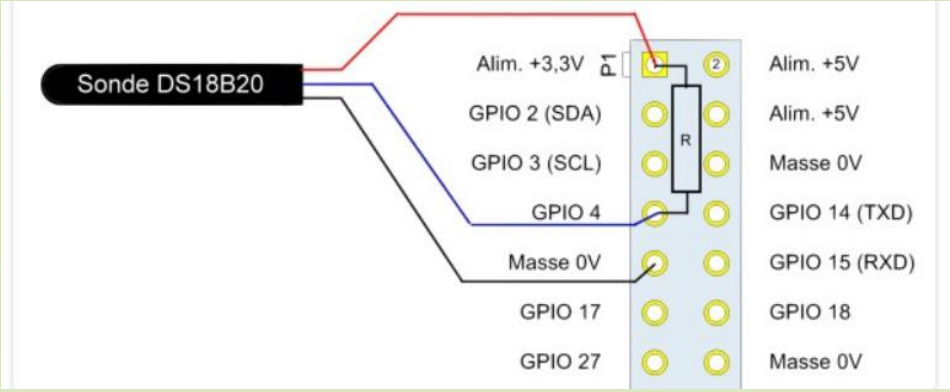


Présentation des composants utilisés

Capteur DS18B20

Caractéristiques:
- Capteur de température DS18B20
- Protocole: 1-Wire
- Plage de mesure: -55°C à +125°C
- Précision: ± 0,5°C
- Résolution : 9 - 12 bit
- 3 broches (Signal / + / -) au pas de 2,54 mm
- Dimensions: 30 x 20 mm

La Sonde DS18B20 se connecte sur la broche GPIO4 de la Raspberry avec une résistance de 4.7kΩ entre GPIO4 et l'alimentation.



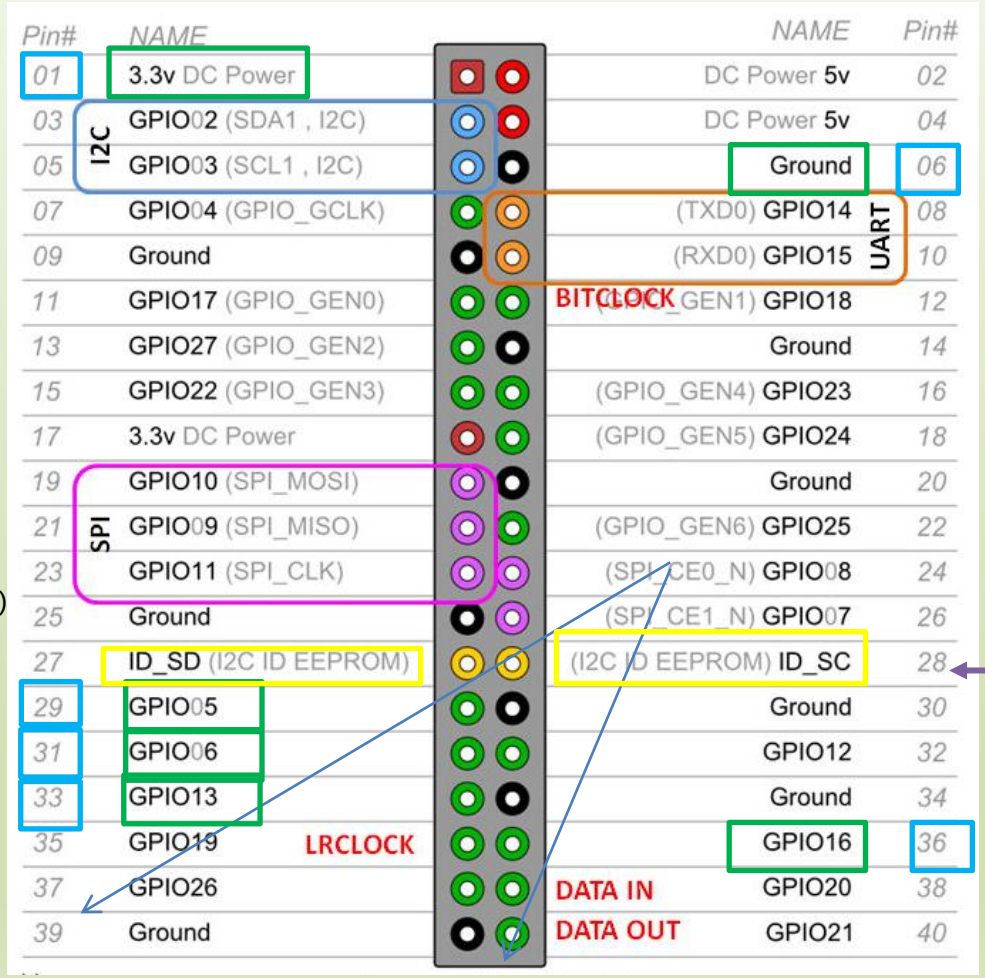
Présentation des composants utilisés

Détail port GPIO

Il existe 2 numérotations des broches:

numérotations graphique:
GPIO.BOARD

numérotations électrique:
GPIO.BCM (recommandé)

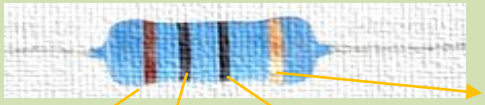


Ne pas utiliser les broches: ID_SD & ID_SC

Présentation des composants utilisés

Code couleur résistance

Couleur	Valeur 1ére Bande	Valeur 2éme Bande	Valeur 3éme Bande	Mnémotechnique
Noir	0	0	$10^0 = 1$	Ne
Marron	1	1	$10^1 = 10$	Manger
Rouge	2	2	$10^2 = 100$	Rien
Orange	3	3	$10^3 = 1\,000$	Ou
Jaune	4	4	$10^4 = 10\,000$	Jeûner
Vert	5	5	$10^5 = 100\,000$	Voilà
Bleu	6	6	$10^6 = 1\,000\,000$	Bien
Violet	7	7		Votre
Gris	8	8		Grande
Blanc	9	9		Bêtise



4éme Bande dorée
Précision

1ére Bande

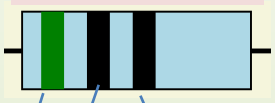
2éme Bande

3éme Bande
Coeff de multiplication

Présentation des composants utilisés

Exemples réalisés à partir de l'utilitaire: **code_couleur.py**

50Ω 50*1



Vert **5**
Noir **0**
Noir $10^{0} = 1$

1 000Ω ou **1k Ω** 10*100

Marron **1**
Noir **0**
Rouge $10^{2} = 100$

10000Ω ou **10k Ω** 10*1 000

Marron **1**
Noir **0**
Orange $10^{3} = 1\,000$

220Ω 22*10

Rouge **2**
Rouge **2**
Marron $10^{1} = 10$

2 200Ω ou **2.2k Ω** 10*100

Rouge **2**
Rouge **2**
Rouge $10^{2} = 100$

100 000Ω ou **100k Ω** 10*10 000

Marron **1**
Noir **0**
Jaune $10^{4} = 10\,000$

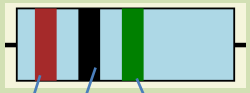
470Ω 47*10

Jaune **4**
Violet **7**
Marron $10^{1} = 10$

4 700Ω ou 4.7k **Ω** 47*100

Jaune **4**
Violet **7**
Rouge $10^{2} = 100$

1 000 000Ω ou **1MΩ** 10*100 000



Marron **1**
Noir **0**
Vert $10^{5} = 100\,000$

Présentation des composants utilisés

B. Logiciel **Python**

Récapitulatif sur les bibliothèques et les objets utilisés

gpiozero Cette bibliothèque utilise la numérotation des broches **BCM**

Ex:

```
>>>from gpiozero import LED
>>>led = LED(17)                                  # GPIO17  broche 11
>>>led.on()                                        # allume Led
>>>led.off()                                       # éteint LED
<gpiozero.LED object on pin GPIO17, active_high=True, is_active=False>
```

Ex:

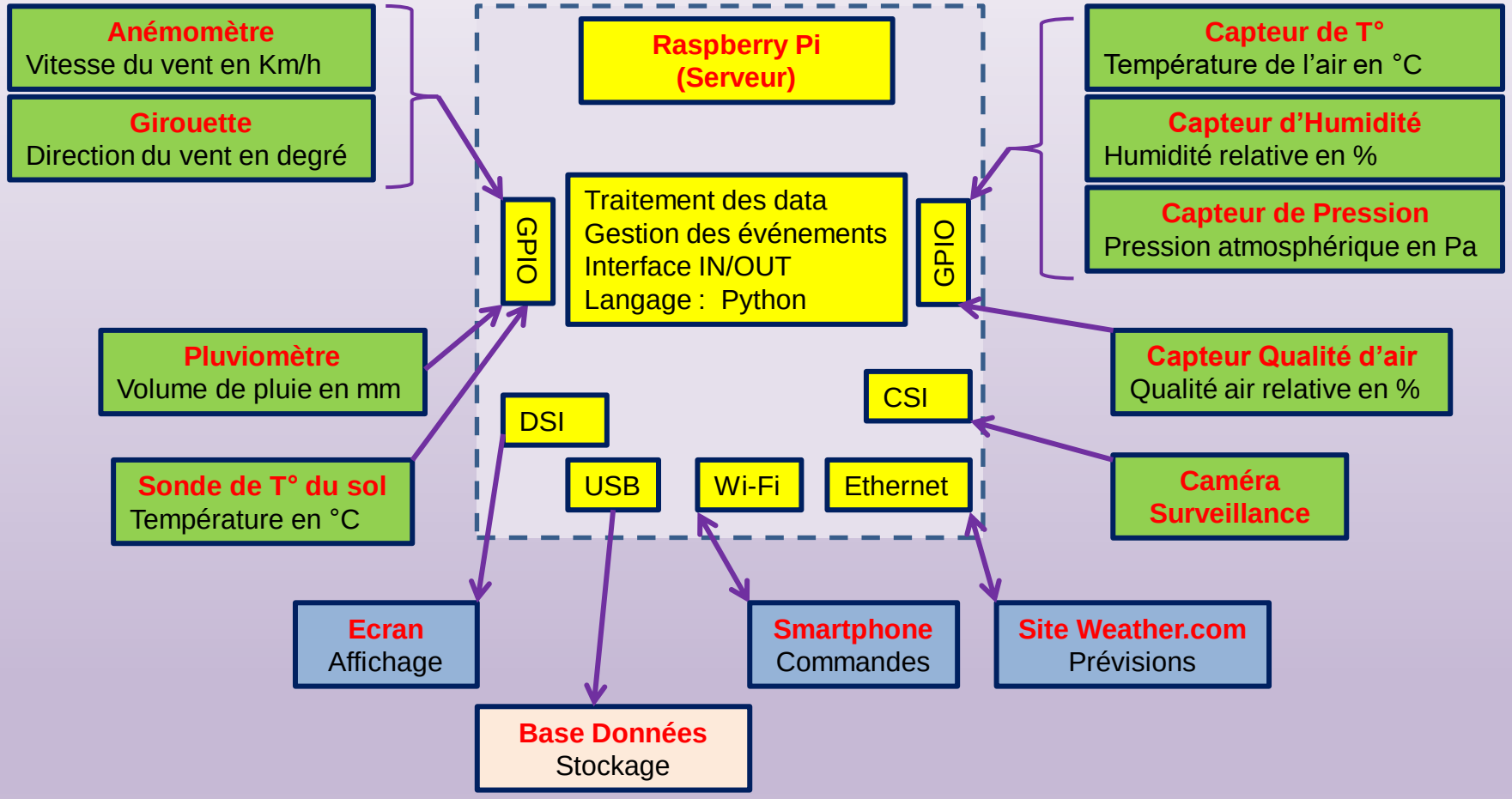
```
>>>from gpiozero import Button
>>>bp = Button(2)                                 # GPIO2
>>>bp
<gpiozero.Button object on pin GPIO2, pull_up=True, is_active=False>
```

De cette bibliothèque nous utiliserons les '**Objets**':

Button, Buzzer, LED, LEDBoard, PWMLED, RGBLED

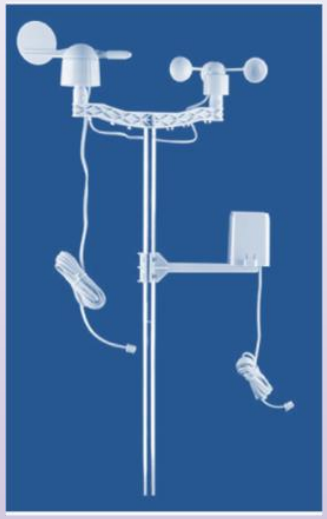
Présentation Station météo

1. Synoptique d'une station météo complète



Présentation Station météo

2. Matériels



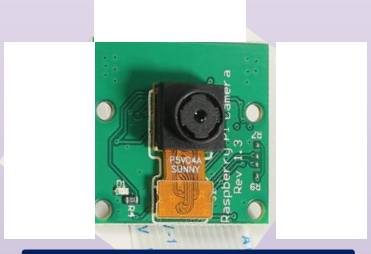
Anémomètre/Girouette

Capteur de T°/Humidité/Pression

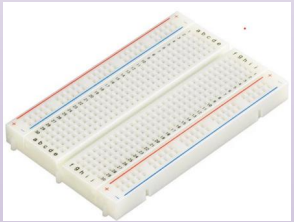
Capteur de T°/Humidité/Pression/Qualité air

Capteur de T° sol

Pluviomètre

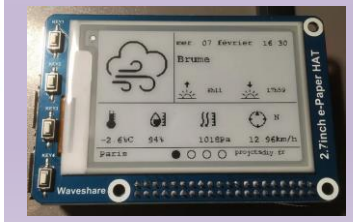


Caméra surveillance



Breadboard

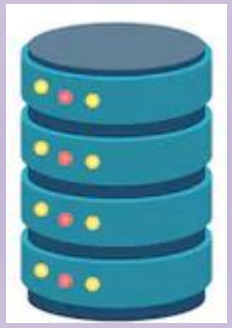
Raspberry Pi



Ecran

Smartphone

Site Weather.com



Base Données

Station météo

3. Méthode pour la réalisation

La réalisation s'effectuera en intégrant capteur par capteur

BME 280 (T° ; Pression ; Humidité)

DS18B20 (sonde de T° du sol)

Anémomètre / Girouette (*Plus tard*)

Pluviomètre (*Plus tard*)

Affichage prévision avec site '*Weather.com*' (*Plus tard)*

Application sur smartphone

Affichage sur un écran OLED ou E-Paper (*Plus tard)*

Station météo

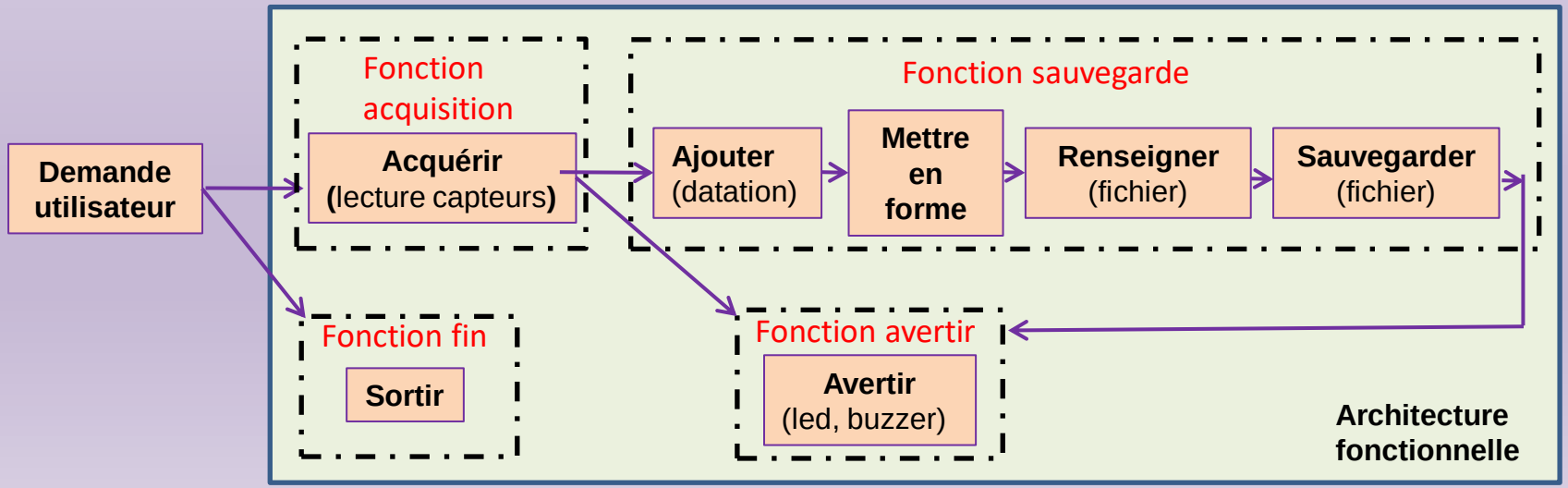
Etape 1: intégration du capteur **BME280 (*T°, Pression & Humidité*)**

. Description textuelle du système

Lorsque l'utilisateur appuie sur un bouton, le programme doit:

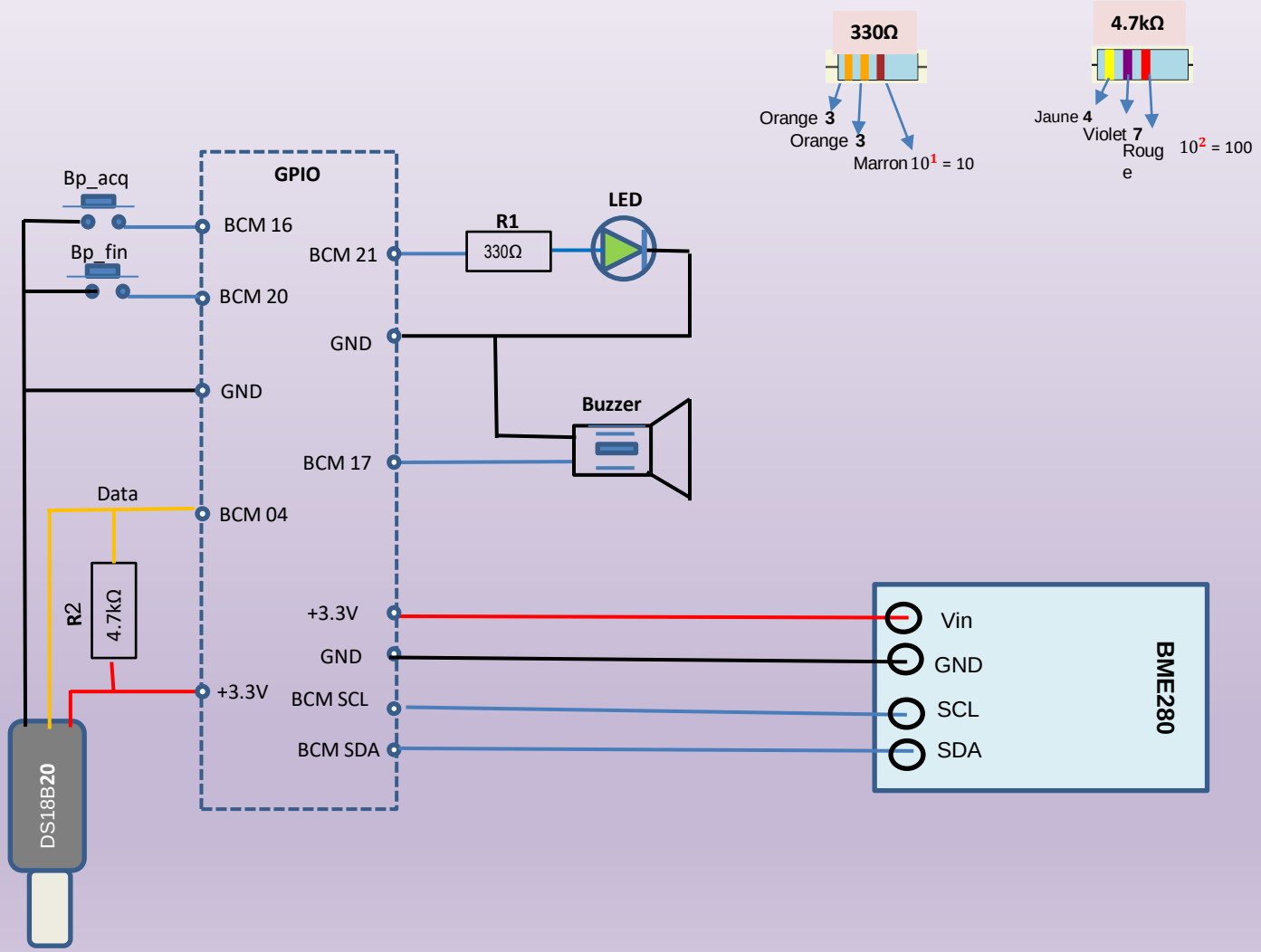
. acquérir les données fournies par le capteur
. dater la mesure
. mettre en forme le résultat
. renseigner le résultat dans un fichier.
. sauvegarder le fichier
. allumer une led pendant l'acquisition
. actionner un buzzer pour signaler la fin de l'acquisition
. arrêter le programme

. Description fonctionnelle



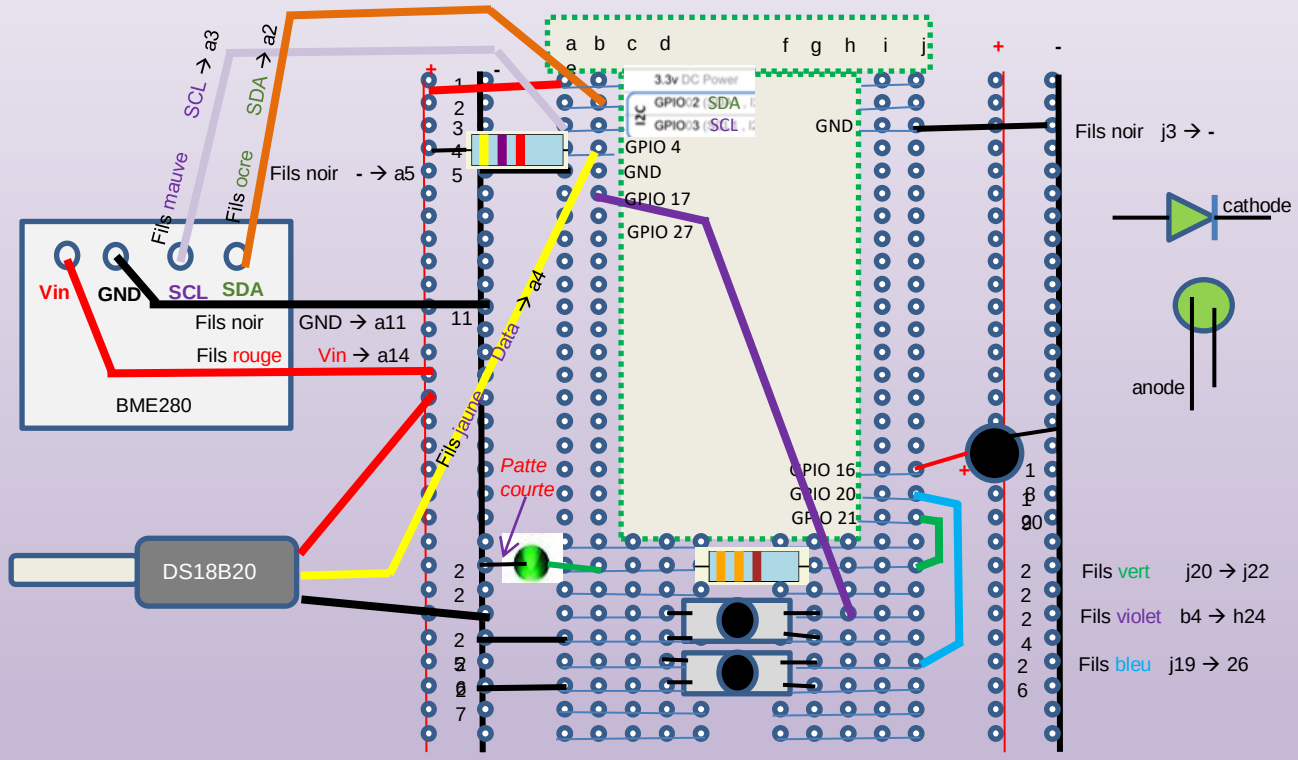
Station météo

Schéma électrique avec les 2 capteurs



Station météo

Câblage



Ajout des librairies

Programme meteo.py

```
from gpiozero import Button, LED, TonalBuzzer
from gpiozero.tones import Tone

from openpyxl import load_Workbook

import locale
import bme280, sonde
import datetime, time
```

Programme sonde.py

```
import glob
```

Ajout des bibliothèques suivantes:

gpiozero

Button pour les boutons poussoirs
utilisation de la fonction: **.when_pressed =**

LED pour led
utilisation des fonctions: **.on()** & **.off()**

TonalBuzzer pour le son du buzzer
utilisation des fonctions: **.play()** & **.stop()**

gpiozero.tones

Tone pour la tonalité du buzzer
utilisation de la fonction: **Tone(midi=)**

openpyxl

load_Workbook pour le classeur LibreOffice calc
utilisation des fonctions: **load_workbook(‘**@ du fichier**.xls**
.save(‘@ du fichier**.xlsx ‘)**

locale utilisation de la fonction : **locale.setlocale(locale.LC_TIME,’’)**

bme280 utilisation de la fonction: **bme280.readBME280All()**

sonde utilisation de la fonction **sonde.acq_sonde()**

datetime

datetime pour la date
utilisation des fonctions: **datetime.now()**
.strftime(‘%a %d %B %Y’)

time

time pour le temps
utilisation des fonctions: **time.strftime(‘%H %M %S’)**
time.sleep()

glob utilisation de la fonction **glob.glob((**@ du fichier **28*/w1_slave)**

Ajout des librairies

Procédure d'installation de la bibliothèque openpyxl

En mode « Console »

```
Nouveau bureau : masterCIGpi:1 (192.168.1.28:1)
pi@masterCIGpi:~ $ pip3 install openpyxl
```

Procédure d'installation de la bibliothèque smbus avec les outils liés au bus i2c

En mode « Console »

```
Nouveau bureau : masterCIGpi:1 (192.168.1.28:1)
pi@masterCIGpi:~ $ sudo apt-get install -y python-smbus python3-smbus
```

Bibliothèque I2C

```
Nouveau bureau : masterCIGpi:1 (192.168.1.28:1)
pi@masterCIGpi:~ $ sudo apt-get install -y python-dev python3-dev
```

Outils développement

```
Nouveau bureau : masterCIGpi:1 (192.168.1.28:1)
pi@masterCIGpi:~ $ sudo apt-get install -y i2c-tools
```

Installation des outils de diagnostique

Ajout des librairies

Procédure d'installation de la bibliothèque bme280

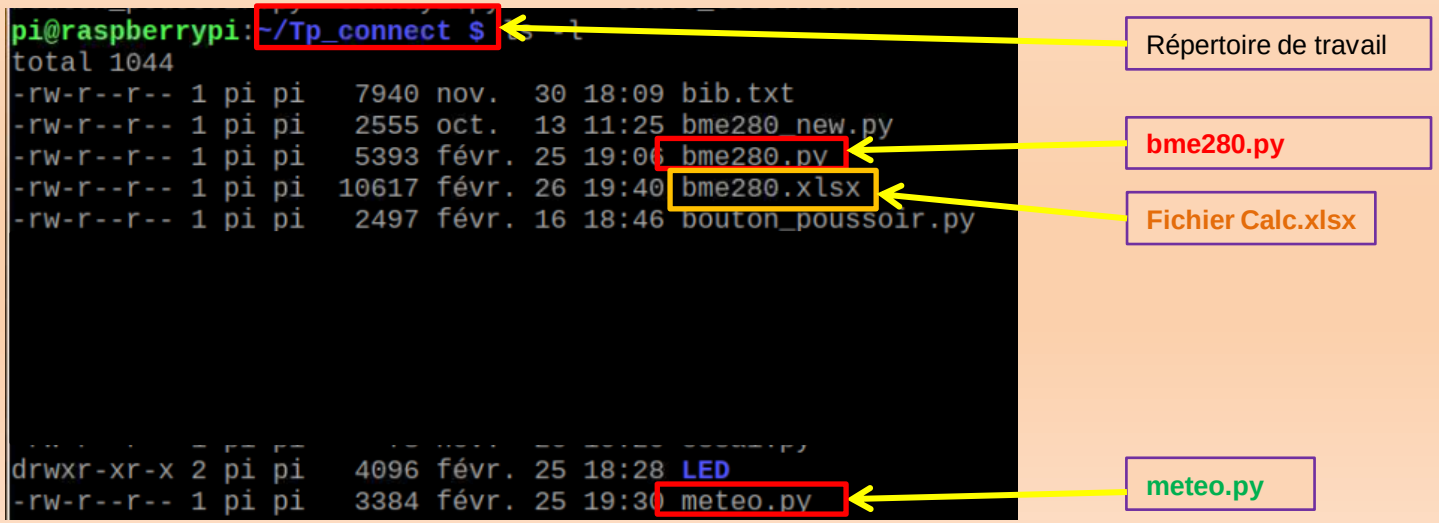
En mode « Console »

```
Nouveau bureau : masterCIGpi:1 (192.168.1.28:1)
pi@masterCIGpi:~ $ pip3 install bme280
```

Installation du logiciel bme280.py

Copier le logiciel bme280.py

Nota : Pour que**: import bme280** fonctionne:
il faut que ce fichier soit dans le même répertoire que le programme **meteo.py**



Ajout de la sonde DS18B20

Intégration du capteur **DS18B20 (sonde de *T*°)**

Changer le nom du fichier *sonde_ds18b20 en sonde* en utilisant la commande **mv** (move)
(plus pratique pour import)

Ajout de la sonde DS18B20

Une fois le câblage de la sonde fait

```
Nouveau bureau : masterCIGpi:1 (192.168.1.28:1)
pi@masterCIGpi:~ $ sudo lsmod | more
```

Vérifier si les fichiers suivants sont présents:

w1_gpio (protocole **1-wire** est déjà géré par la PI)

w1_therm (composants de la famille *28h* reconnus **DS18B20**)

Cas des modules non installés:

Utilisation de la commande modprobe: ← permet de charger ou décharger des modules.

```
Nouveau bureau : masterCIGpi:1 (192.168.1.28:1)
pi@masterCIGpi:~ $ sudo modprobe w1-gpio
pi@masterCIGpi:~ $ sudo modprobe w1-therm
pi@masterCIGpi:~ $ sudo lsmod | grep w1
```

Active le protocole **1-Wire**

Gestion des composants 1-Wire famille ***28h***

Vérification présence des 2 modules **w1**

Résultat

```
w1_gpio                 3171  0
w1_therm                7330  0
wire                   32947  2 w1_gpio,w1_therm
```

Ajout des librairies

Installation de Cherrypy

```
Nouveau bureau : masterCIGpi:1 (192.168.1.28:1)
pi@masterCIGpi:~ $ Sudo apt-get update
```

```
Nouveau bureau : masterCIGpi:1 (192.168.1.28:1)
pi@masterCIGpi:~ $ Sudo apt-get upgrade
```

```
Nouveau bureau : masterCIGpi:1 (192.168.1.28:1)
pi@masterCIGpi:~ $ Sudo pip3 install cherrypy
```

Installation de matplotlib

```
Nouveau bureau : masterCIGpi:1 (192.168.1.28:1)
pi@masterCIGpi:~ $ sudo apt  install python3-matplotlib
```

Ajout des librairies

Programme meteo_web.py

```
from matplotlib import matplot

Import cherrypy, os
Import  meteo
import time
```

Ajout des bibliothèques suivantes:

cherrypy permet de construire un site Web
utilisation des fonctions: **cherrypy.session.get()**
index.exposed
cherrypy.config.update()
cherrypy.quickstart()

os permet d'utiliser des fonctions système: **open(), close()**
os.getcwd()

meteo utilisation de la fonction **sonde.acq_sonde()**

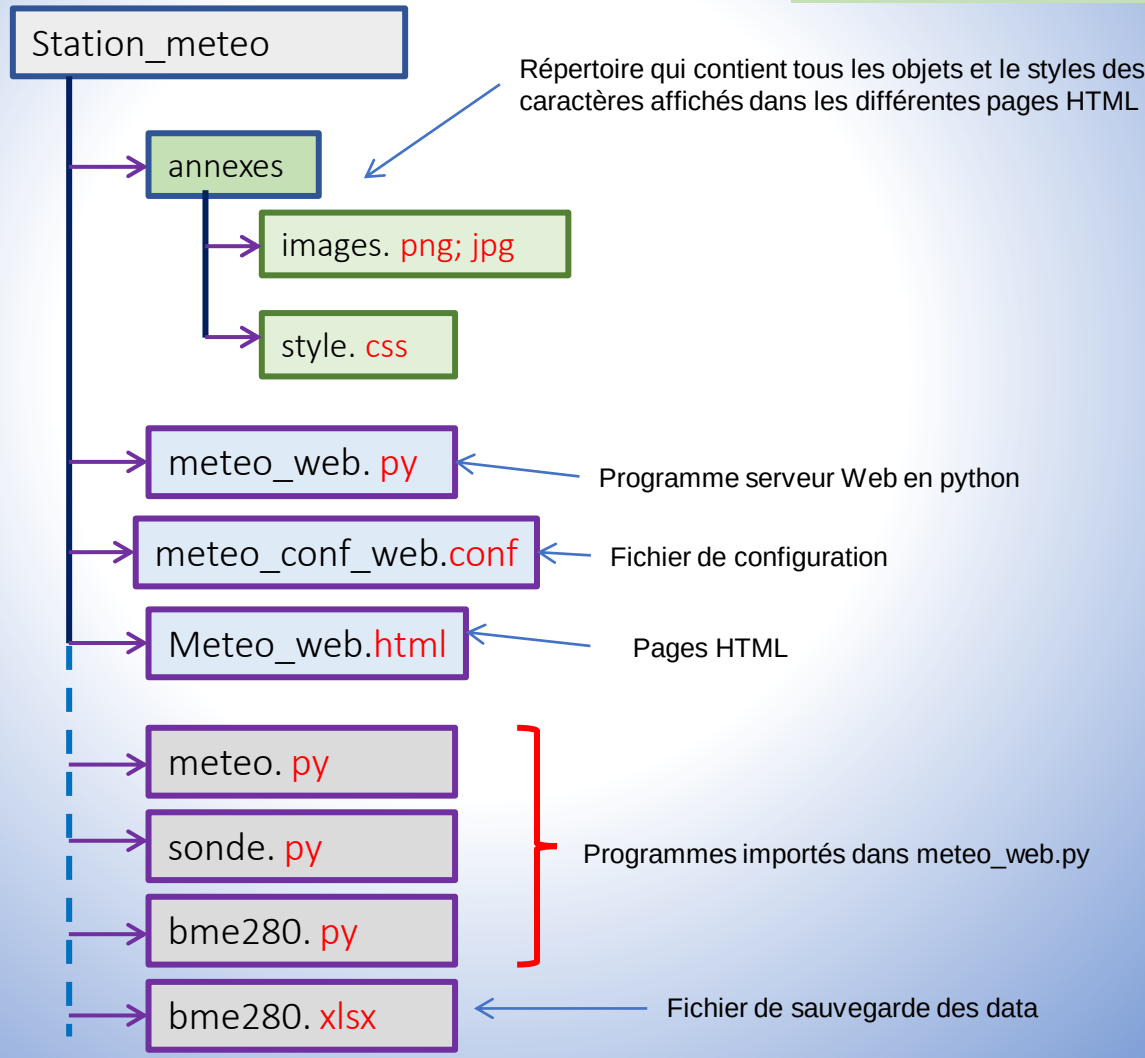
time
time utilisation de la fonction: **time.sleep()**

matplotlib permet de construire un graphique

Arborescence

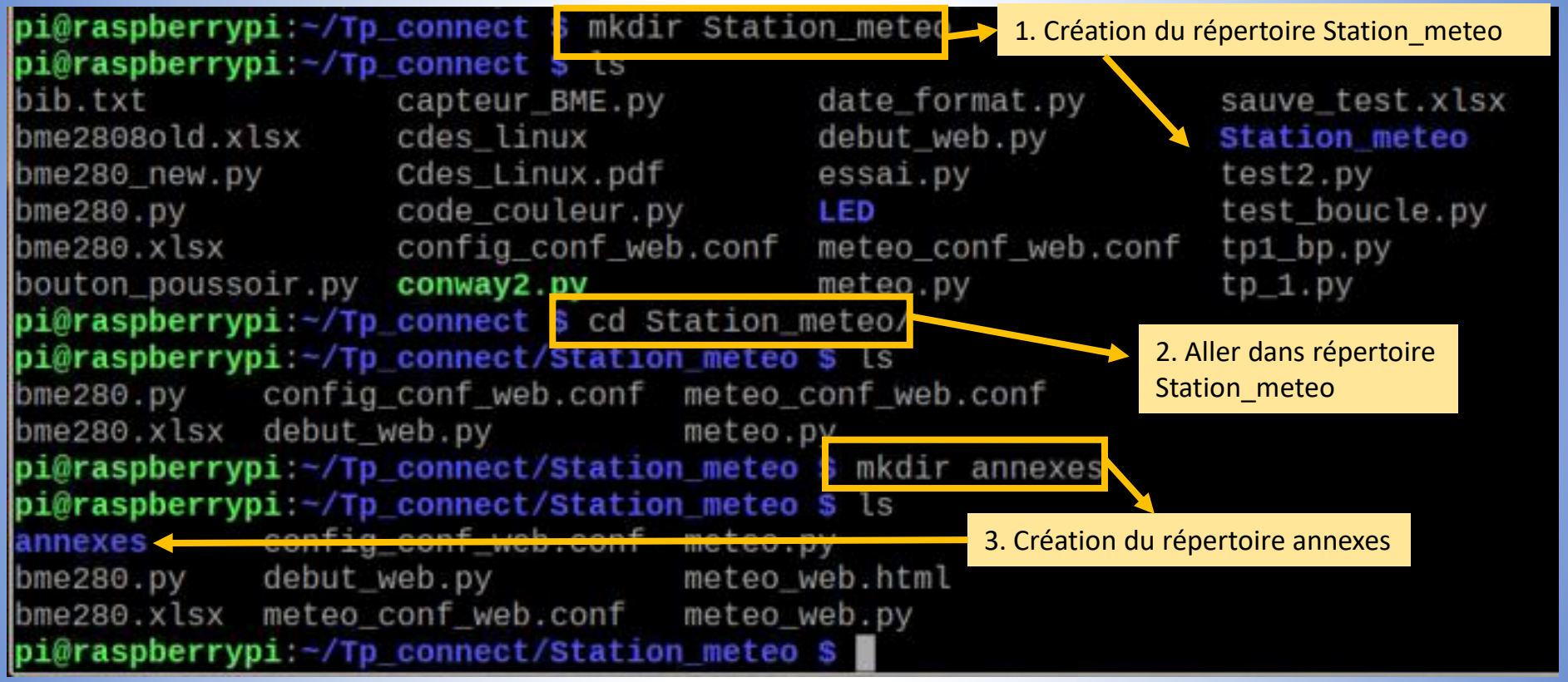
Constitutif d'un serveur :
fichier de configuration,
programme principal python,
pages HTML,
répertoire styles

Arborescence du serveur WEB Station météo



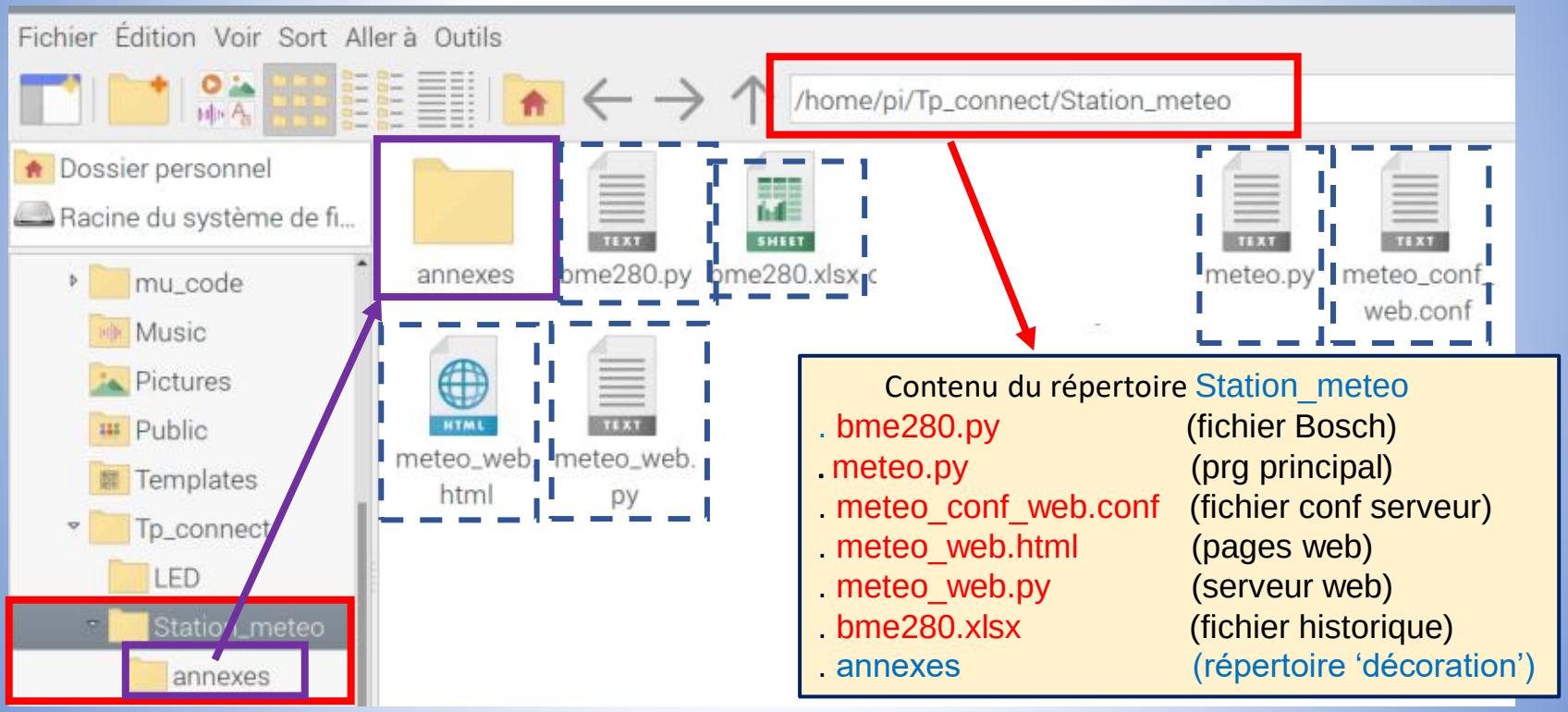
Arborescence

Mode Console



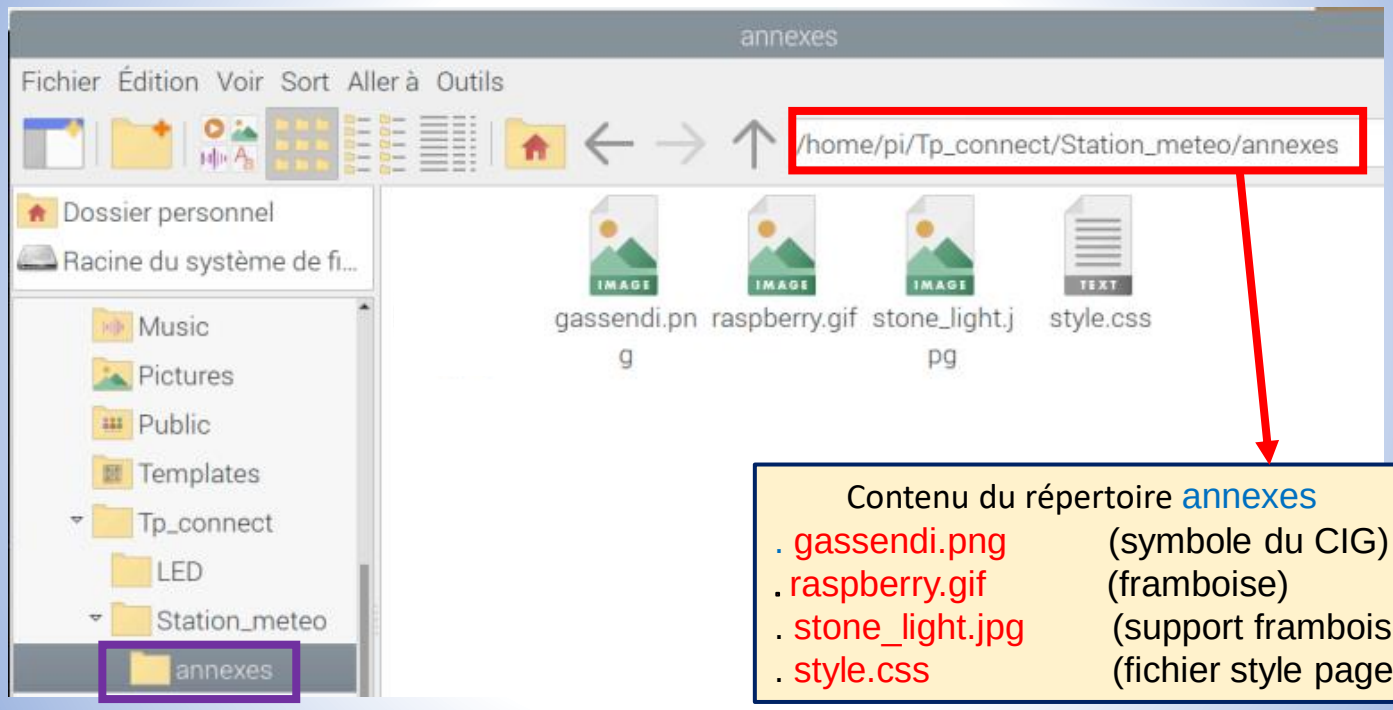
Arborescence

Mode Graphique



Arborescence

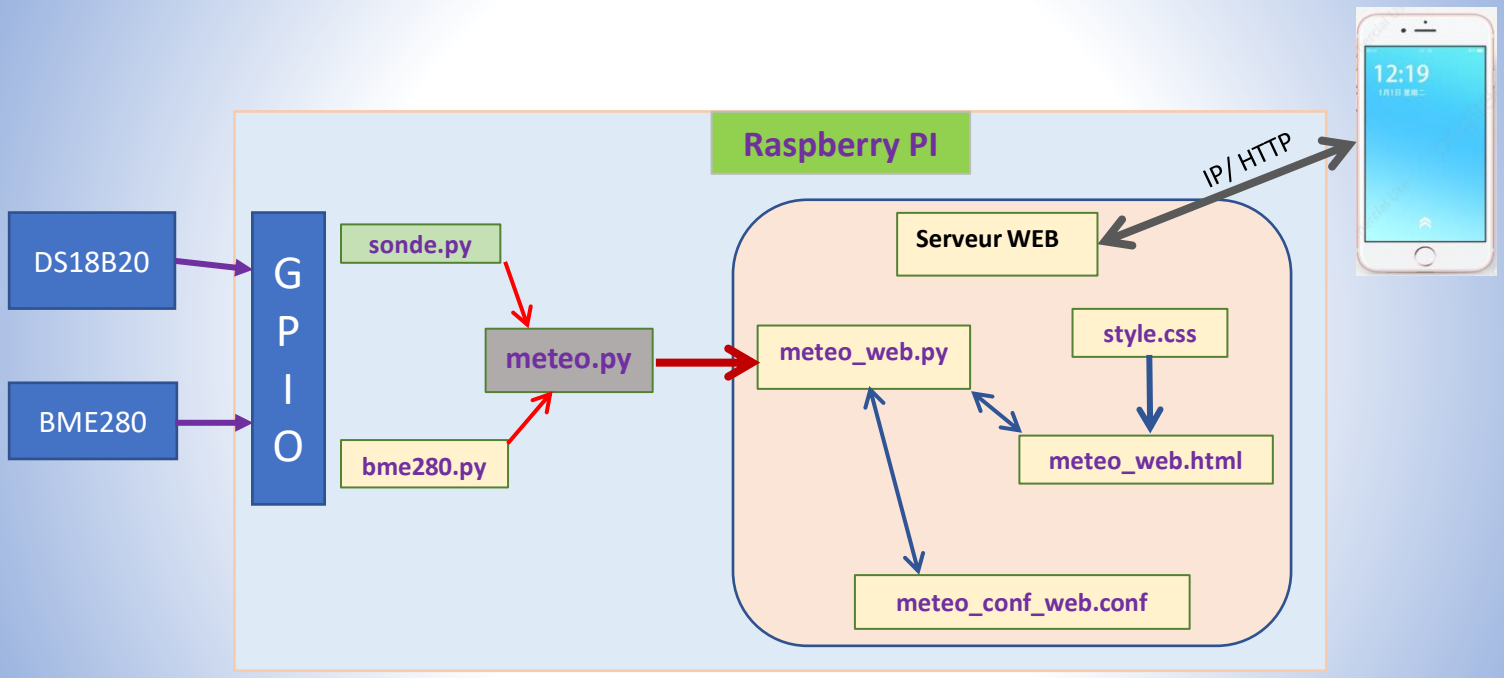
Mode Graphique



Contenu du répertoire annexes

. gassendi.png (symbole du CIG)
. raspberry.gif (framboise)
. stone_light.jpg (support framboise)
. style.css (fichier style pages)

Architecture de la station météo



Codage station météo

Programme sonde.py

```
#       lecture des données de la sonde DS18B20
import glob

def lire_fichier(emplacement):
    fiche = open(emplacement)
    contenu = fiche.read()
    fiche.close()
    return contenu

def extraire_temp(contenu):

    seconde_ligne = contenu.split(''/n'')[1]
    val_temp = seconde_ligne.split('' '')[9]
    return float(val_temp[2:]) / 1000

def acq_sonde():
    repertoire_capteur= glob.glob(''/sys/bus/w1/devices/28*/w1_slave'')

    if len(repertoire_capteur) > 0:
       contenu_fichier = lire_fichier(repertoire_capteur[0])
       temp_ext = extraire_temp(contenu_fichier)

     else:
        print('Sonde non détectée. Vérifier le branchement')
        temp_ext = 666

    return temp_ext
```

glob bibliothèque qui permet de rechercher un fichier dans une arborescence

Fonction qui récupère les datas de la sonde

Fonction qui extrait la température parmi les datas de la sonde

saute la 1ere ligne
récupération de la température case 9
suppression du ''t=''

Fonction qui recherche le fichier w1_slave de la sonde

Appel de la fonction lire_fichier()

Appel de la fonction extraire_temp()

Valeur de la température affichée sur la page Web

Codage station météo

Programme **bme280.py**

\# ***lecture des données du capteur BME280***

Programme à télécharger chez le constructeur **BOSCH**

L'adresse officielle du datasheet:

https://www.bosch-sensortec.com/bst/products/all_products/bme280

Ce programme utilise la bibliothèque **smbus** qui gère la liaison **i2c**

L'adresse i2c du composant est : **0x76**

La récupération de la *Température*, *Pression, Humidité* par le programme **meteo.py** est faite par l'appel de la fonction

readBME280All()

Codage station météo

Programme meteo.py

Programme qui récupère les données provenant des programme « **sonde.py** & **bme280.py** »
et après ajout de la **datation** envoi le tout au programme **meteo_web.py** (serveur Web)

```
from gpiozero import Button, LED, TonalBuzzer
from gpiozero.tones import Tone

from openpyxl import load_Workbook

import locale
import datetime, time

import bme280, sonde
```

Import de toutes les bibliothèques utiles

Import de la date format France

Import des 2 programmes capteurs

```
bp_acq = Button(16)
bp_fin   = Button(20)

led_verte= LED(21)

buz = TonalBuzzer(17)
```

Affectation des composants sur le GPIO

```
index, t_acq =0, 3
```

Variables globales

Codage station météo

Programme meteo.py

Fonction acquisition

```
def acquisition():
    global index

    print(" Le bouton poussoir acquisition a été pressé: ")
    print("La durée de l'acquisition est d'environ:", t_acq,"s")

    temperature,pression,humidite = bme280.readBME280All()
    print("Temp : ", temperature, "A°C \t P : ", pression, "hPa \t HR : ", humidite, "%")

    temperatur_ext = sonde.acq_sonde()
    print("T° sonde :", temperatur_ext)
```

Fonction qui récupère les datas des 2 capteurs

Appel de la fonction **readBME280All()** du programme **bme280.py**

Appel de la fonction **sonde.acq_sonde()** du programme **sonde.py**

```
    avertir()
```

Appel de la fonction **avertir()** (allume LED et active le buzzer)

Appel de la fonction **sauvegarder(t., p. , h. , i. , t.)** (enregistrement dans LibreOffice calc)

```
    index = index + 1
    l=sauvegarder(temperature,pression,humidite,index,temperatur_ext)
    return l
```

La fonction **acquisition** renvoie la variable **l** au serveur web qui contient les valeurs :
- temperature, pression , humidite (bme280)
- index (n° de l'acquisition)
- temperatur_ext (sonde DS18B20)

retournées par la fonction **sauvegarder** à travers la variable **ligne**

Codage station météo

Programme meteo.py

Fonctions avertir

```
#   avertissement  lumineux
def avertir():
    led_verte.on()
    time_sleep(3)
    led_verte.off()
```

Début d'acquisition

```
#   avertissement  sonore   (old Mac Donald)
def son_finacq(numero):
    couplet_1 =["G4", "G4", "G4", "D4", "E4", "E4", "D4"]
    couplet_2 =["B4", "B4", "A4", "A4", "G4"]
    couplet_3 =["D4", "G4", "G4", "G4", "D4", "E4", "E4", "D4"]

    musique = [couplet_1, couplet_2, couplet_3, couplet_2]

    for partition in musique:
        for note in partition:
            buz.play(note)
            time.sleep(0.4)
            buz.stop()
            time.sleep(0.1)
        time.sleep(0.2)

    print("L'acquisition numéro {0} est terminée".format(numero))
```

Insertion dans un texte de la valeur de l'index

Fin d'acquisition

Codage station météo

Programme meteo.py

Fonction sauvegarder

Récupération des data capteurs

```
#    fonction qui ajoute la datation aux data formatées puis sauvegarde sous LibreOffice calc.
def sauvegarder(t, p, h, i, e):
    locale.setlocale(locale.LC_TIME,' ')
    dat = datetime.datetime.now()

    form_date = dat.strftime(''%a %d %B %Y'')
    horaire = time.strftime(''%Hh %Mmn  % Ss'')

    form_t = round(t, 1)
    form_p = int(p)
    form_h = int(h)
    form_i = str(' enr n°') + str(i)
    form_e = round(e, 1)

    wb = load_workbook('home/pi/Tp_connect/Station_meteo/bme280.xlsx')
    fiche = wb['data']

    ligne = (form_i, form_date, horaire, form_t, form_p, form_h, form_e)
    fiche.append(ligne)
    time.sleep(1)

    wb = save('home/pi/Tp_connect/Station_meteo/bme280.xlsx')
    time.sleep(10)

    son_finacq(i)

    return ligne
```

Récupération de la date système format France

Mise en forme de la date système (jour, n°du jour, mois, année, h , mn, s)

Mise en forme des data capteurs

Ouvrir le fichier à l'@ indiquée & onglet 'data'

Ecriture d'une ligne de 7 colonnes)

Sauvegarde du fichier

Appel de la fonction **son_finacq(i)**
Avertissement sonore de fin d'acquisition

ligne renvoie tous les paramètres formatés au serveur Web

Ajouter (datation)

Mise en forme des paramètres

Fonction sauvegarde

Renseigner (fichier)

Sauvegarder (fichier)

Codage station météo

Programme meteo.py

Fonction finir

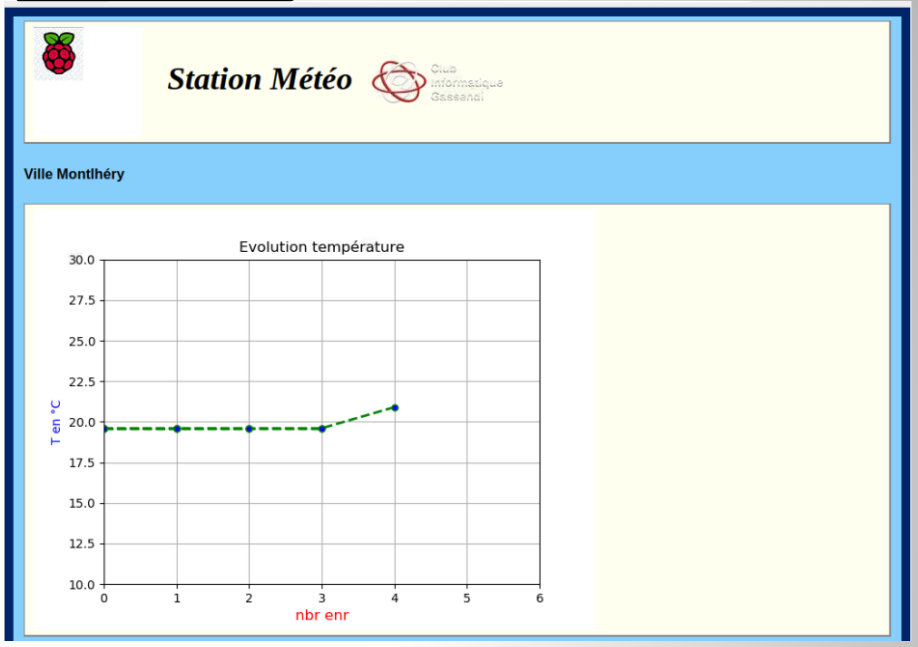
```
#    fonction de fin de programme
def fin():
    print('’Fin du programme d’acquisition”)
    print('’Pour sortir définitivement du programme, taper ‘CTRL F2’ ”)
    quit()
```


```
#          programme principal  demande utilisateur
#       permet de tester  meteo.py sans le serveur Web
bp_acq.when_pressed = acquisition

bp_fin.when_pressed = fin
```

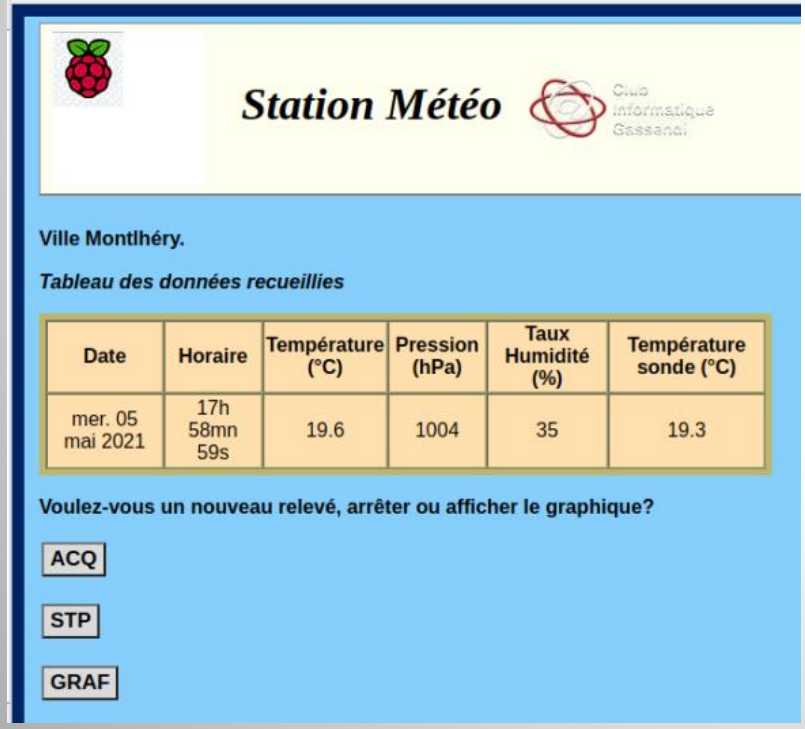
Présentation des pages Web

Page d'accueil

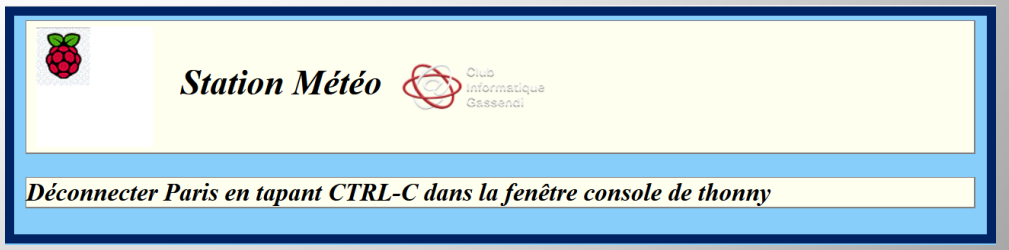
Page graphique



Page d'affichage



Page de sortie

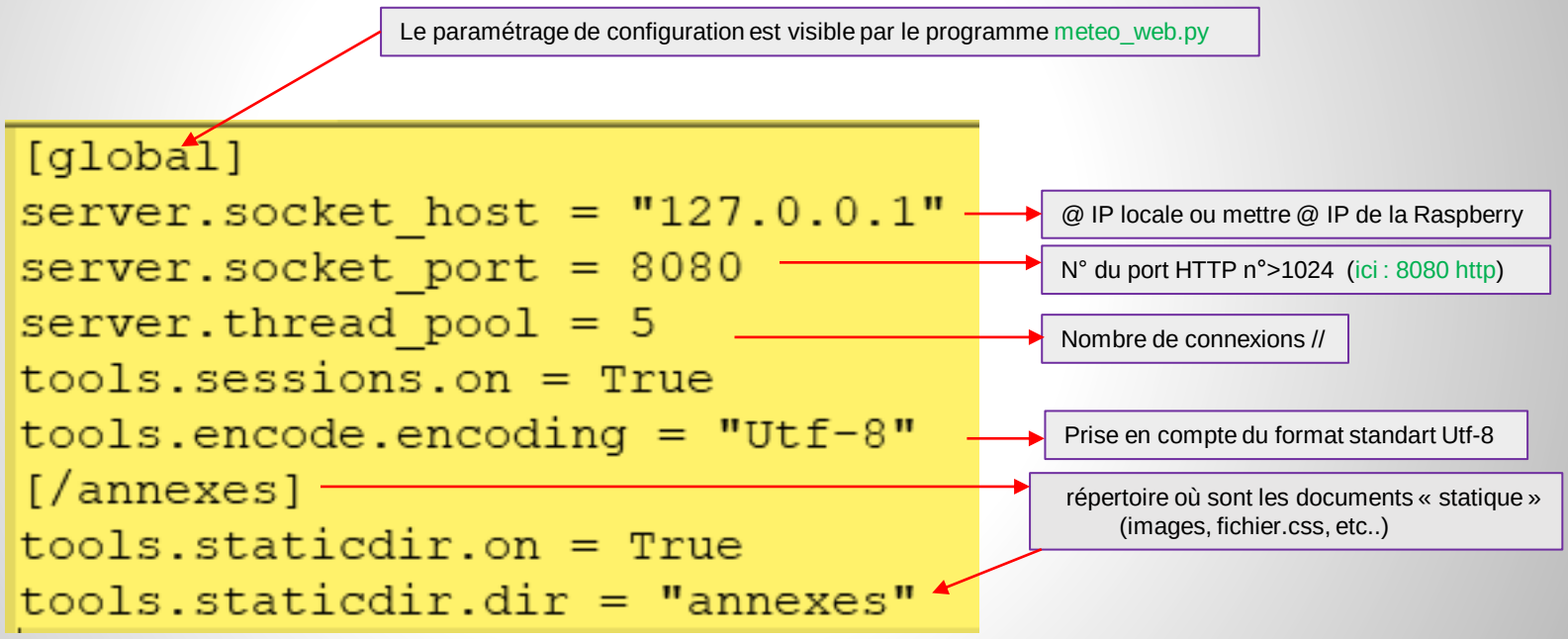


Codage station météo

Programme meteo_conf_web.conf

Nombre de connexions //

Fichier texte qui configure le serveur Web



Fichier de configuration appelé au démarrage par le programme meteo_web.py pour lancer le *serveur Web*

Codage station météo

Diagramme du lancement du serveur Web

La commande suivante consiste à s'assurer que le répertoire racine du site est bien le répertoire courant

meteo_web.py (serveur web)

tools.staticdir.root → @ de gestion répertoire racine

os.getcwd() → récupère le répertoire courant

```
cherrypy.config.update({"tools.staticdir.root": os.getcwd()})
```

Dictionnaire de configuration de cherrypy

```
cherrypy.quickstart(Pracine(), config="meteo_conf_web.conf")
```

Démarrage rapide du serveur

Appel de l'application

Appel du fichier de configuration

meteo_conf_web.conf

```
[global]
server.socket_host = "127.0.0.1"
server.socket_port = 8080
server.thread_pool = 5
tools.sessions.on = True
tools.encode.encoding = "utf-8"
[/annexes]
tools.staticdir.on = True
tools.staticdir.dir = "annexes"
```

Résultat après démarrage

```
[09/Mar/2021:17:58:36] ENGINE Listening for SIGTERM.
[09/Mar/2021:17:58:36] ENGINE Bus STARTING
[09/Mar/2021:17:58:36] ENGINE Set handler for console events.
CherryPy Checker:
dir is a relative path and no root provided.
section: [/annexes]
root: None
dir: 'annexes'

[09/Mar/2021:17:58:36] ENGINE Started monitor thread 'Autoreloader'.
[09/Mar/2021:17:58:36] ENGINE Serving on http://127.0.0.1:8080
[09/Mar/2021:17:58:36] ENGINE Bus STARTED
```

@ à entrer sur le navigateur

Le serveur est actif

Codage station météo

Programme meteo_web.py

```
import cherrypy, os
import datetime, time

import meteo

from matplotlib import pyplot

class Glob(object):

    motifs = ''meteo_web.html''

    dico_motifs = {}

    abscis = []
    ordon  = []

    index = 0
```

Bibliothèques qui gère le serveur Web

Programme qui récupère les data capteurs

Bibliothèque qui permet de tracer des graphiques

@ du fichier page Web

meteo_web.html

```
[*enTete*]
<html>
<!-- Balise
<head>
<meta conte
<link rel=s
</head>
<body>
<h1><img sr
{0}
</body>
</html>
##########
```

Codage station météo

Programme meteo_web.py

fonction qui met dans le dictionnaire **dico_motifs** toutes les pages html à afficher

```
def chargermotifs():
    global phrase, label

    fiche = open(Glob.motifs, "r", encoding="Utf8")

---------- Traitement du fichier HTML pour extraire les titres des pages (étiquette dico) ----------

    try:

        for ligne in fiche:
            if ligne[:2] == "[*":
                label = ligne[2:]
                label = label[:-1].strip()
                label = label[:-2]
                phrase =""
            else:

                if ligne[:5] == "#####:
                    Glob.dico_motifs[label] = phrase
                else:
                    phrase += ligne

    finally:
        fiche.close()
```

lecture du fichier HTML

test de gestion I/O du fichier **try -- finally**

Si le 2 premiers caractères sont **[*** **alors** étiquette trouvée

suppression [* ➔ label vaut **enTete*]**

suppression *] ➔ label vaut **enTete**

Sinon traitement des autres lignes

Si la ligne contient plus de **5 #** **alors** écrire la page

Glob.dico_motifs**[enTete]** contient le texte de la page

Sinon passer ligne suivante

fermeture du fichier HTML

[*enTete*]

enTete*]

enTete

##########

meteo_web.html

```
[*enTete*]
<html>
<!-- Balise
<head>
<meta conte
<link rel=s
</head>
<body>
<h1><img sr
{0}
</body>
</html>
##########
```

Codage station météo

Programme meteo_web.py

```
def misenpg(page):
    return Glob.dico_motifs["enTete"].format(page)
```

fonction qui ajoute un en-tête

```
class Pracine(object):
    def index(self):
        nom = cherrypy.session.get("nom", "")
        if nom:
            acces = cherrypy.session["acces"]
            if acces == "Fin":
                return misenpg(Glob.dico_motifs["pageFin"])
            else:
                return misenpg(Glob.dico_motifs["pageGo"].format(nom))
        else:
            return misenpg(Glob.dico_motifs["pageAccueil"])
    index.exposed = True
```

fonction qui gère les pages Web et ajoute un en-tête

Validation de la fonction

meteo_web.html

```
[*enTete*]
<html>
<!-- Balises HEAD: Entête de la page HTML -->
<head>
<meta content="text/html; charset=utf-8" http-equiv="Content-Type">
<link rel=stylesheet type=text/css media=screen href="/annexes/style.css">
</head>
<body>
<h1><img src="/annexes/raspberry.gif"> Station Météo
    <img src="/annexes/gassendi.png"></h1>
{0}
</body>
```

Fichier style

en-tête

PageAccueil

Codage station météo

Programme meteo_web.py

Page Accueil renseignée

meteo_web.html

```
[*pageAccueil*]
<form action="/identification" method=GET>
<h4>Veuillez SVP entrer votre ville :</h4>
<table>
<tr><td>Votre ville :</td><td><input name="nom"></td></tr>
</table>
<input type=submit class="button" name="acces" value="Go">
<input type=submit class="button" name="acces" value="Fin">
</form>
```

nom = Paris

acces : transmet le choix **Go** ou **Fin**

Fonction qui selon le choix renseigne la page Web adéquate

```
def identification(self, acces="", nom=""):
    cherrypy.session["nom"] = nom
    cherrypy.session["acces"] = acces
    i,da, he, t, p, h, e = 0, "01/Janu/2021", "12h00", 0, 0, 0, 0
    if acces == "Fin":
        return misenpg(Glob.dico_motifs["pageFin"].format(nom))
    else:
        return misenpg(Glob.dico_motifs["pageGo"].format(nom, i, da, he, t, p, h, e))
identification.exposed = True
```

Prise en cpte de la ville

Prise en cpte des Cdes

Si Fin **alors** **renvoi ville**

Sinon
renvoi ville & data tableau

Affichage : **Page Fin**

Affichage : **PageGo**

Codage station météo

Programme meteo_web.html

Data renvoyés par la fonction identification

```
return misenpg(Glob.dico_motifs["pageGo"].format(nom, i, da, he, t, p, h, e))
```

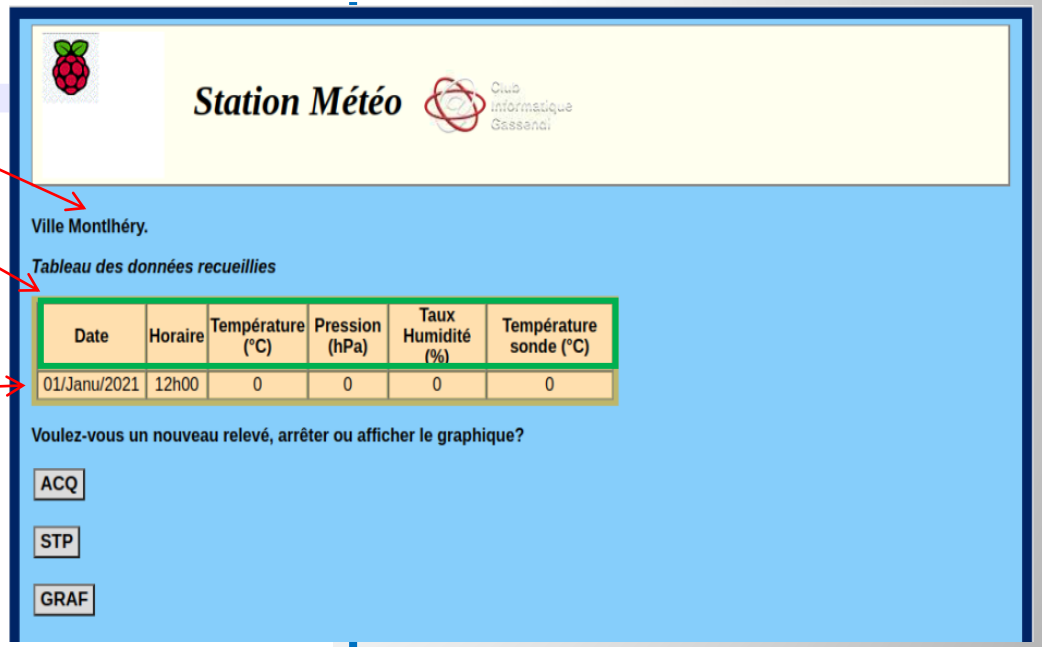
0 1 2 3 4 5 6 7

```
"01/Janu/2021", "12h00", 0, 0, 0, 0
```

meteo_web.html

```
[*pageGo*]
<h3> Ville {0}.
<h3><i>Tableau des données recueillies</i></h3>
<table border="1" cellpadding="1" cellspacing="1" style="width:60%;">
<thead>
<tr>
<!--<th scope="col">n°Acq</th></!-->
<th scope="col">Date</th>
<th scope="col">Horaire</th>
<th scope="col">Température (°C)</th>
<th scope="col">Pression (hPa)</th>
<th scope="col">Taux Humidité (%)</th>
<th scope="col">Température sonde (°C)</th>
</tr>
</thead>
<tbody>
<tr>
<!--<td>{1}</td></!-->
<td>{2}</td>
<td>{3}</td>
<td>{4}</td>
<td>{5}</td>
<td>{6}</td>
<td>{7}</td>

</tbody>
</table>
<h3> Voulez-vous un nouveau relevé, arrêter ou afficher le graphique?</h3>
<form action="/acquerir" method=GET>
<h3> <input type=submit class="button" name="acces" value="ACQ"></h3>
<h3> <input type=submit class="button" name="acces" value="STP"></h3>
<h3><input type=submit class="button" name="acces" value="GRAF"></h3>
</form>
```

PageGo



Codage station météo

Programme meteo_web.html

Data renvoyés par la fonction identification

```
if acces == "Fin":
    return misenpg(Glob.dico_motifs["pageFin"].format(nom))
```

meteo_web.html

```
[*pageFin*]
<h2> Déconnecter {0} en tapant CTRL-C dans la fenêtre console de thonny </h2>
"""""""""""
```

PageFin

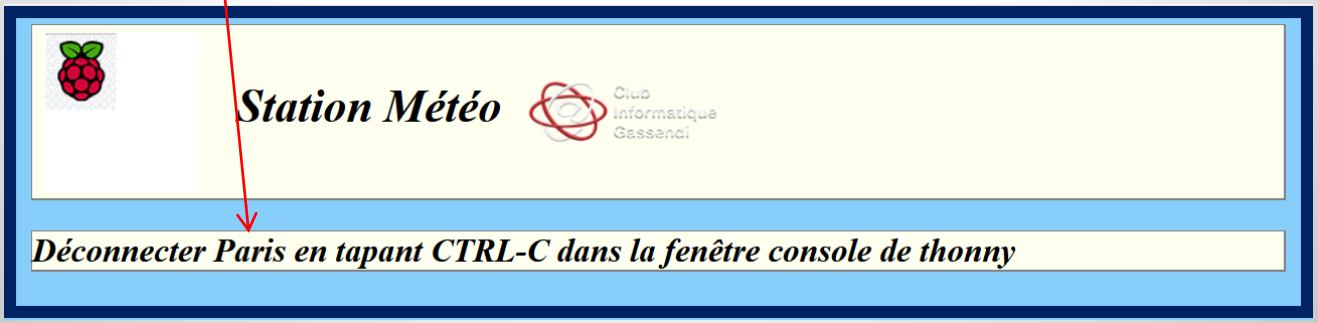
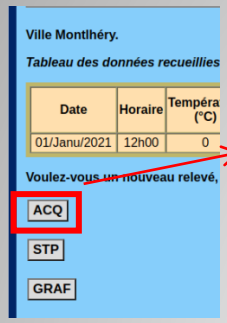


Diagramme de fonctionnement ACQ Affichage Web



appui sur bouton ''ACQ''

meteo_web.py

accès à la fonction **acquerir**

```
def acquerir(self, acces="", nom=""):
    nom = cherrypy.session["nom"]
    cherrypy.session["acces"] = acces

    if acces == "ACQ":
        i = 1
        data_bme = meteo.acquisition()

        journe = data_bme[1]
        horair = data_bme[2]
        temper = data_bme[3]
        pressi = data_bme[4]
        humidi = data_bme[5]
        texter = data_bme[6]
return misenpg(Glob.dico_motifs["pageGo"].format(nom, i,
        journe, horair, temper, pressi, humidi, texter))
```

meteo.py

accès à la fonction **acquisition**

```
def acquisition():
    global index
    index = index + 1
    l=sauvegarder(temperature,pression,humidite,index,temperatur_ext)
return l
```

```
def sauvegarder(t,p,h,i,e):
# mise en forme de la date et de l'horaire
    locale.setlocale(locale.LC_TIME,'')

    dat = datetime.datetime.now()
    form_date = dat.strftime("%a %d %B %Y")

    horaire = time.strftime('%Hh %Mmn %Ss')
ligne = (form_i, form_date, horaire, form_t, form_p, form_h, form_e)

return ligne
```

meteo_web.html

```
[*pageGo*]
<h3> Ville {0}.
<h3><i>Tableau des données recueillies</i></h3>
<table border="1" cellpadding="1" cellspacing="1" style="width:60%;">
<thead>
<tr>
<!--<th scope="col">n°Acq</th>-->
<th scope="col">Date</th>
<th scope="col">Horaire</th>
<th scope="col">Température (°C)</th>
<th scope="col">Pression (hPa)</th>
<th scope="col">Taux Humidité (%)</th>
<th scope="col">Température sonde (°C)</th>
</tr>
</thead>
<tbody>
<tr>
<!--<td>{1}</td>-->
<td>{2}</td>
<td>{3}</td>
<td>{4}</td>
<td>{5}</td>
<td>{6}</td>
<td>{7}</td>

</tbody>
</table>
<h3> Voulez-vous un nouveau relevé, arrêter ou afficher le graphique?</h3>
<form action="/acquerir" method=GET>
<h3> <input type=submit class="button" name="acces" value="ACQ"></h3>
<h3> <input type=submit class="button" name="acces" value="STP"></h3>
<h3><input type=submit class="button" name="acces" value="GRAF"></h3>
</form>
```

Ville Montlhéry.

Tableau des données recueillies

Date	Horaire	Température (°C)	Pression (hPa)	Taux Humidité (%)	Température sonde (°C)
mer. 05 mai 2021	17h 58mn 59s	19.6	1004	35	19.3

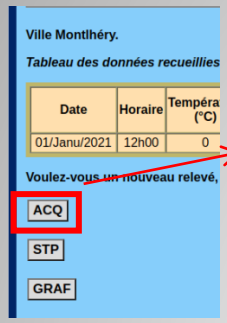
Voulez-vous un nouveau relevé, arrêter ou afficher le graphique?

ACQ

STP

GRAF

Diagramme de fonctionnement ACQ
Enregistrement dans fichier .xlsx



appui sur bouton ''ACQ''

meteo_web.py

accès à la fonction **acquerir**

meteo.py

accès à la fonction **sauvegarder**

bme.xlsx

Écriture dans 'calc'

```
def acquerir(self, acces="", nom=""):
    nom = cherrypy.session["nom"]
    cherrypy.session["acces"] = acces

    if acces == "ACQ":
        i = 1
        data_bme = meteo.acquisition()
```

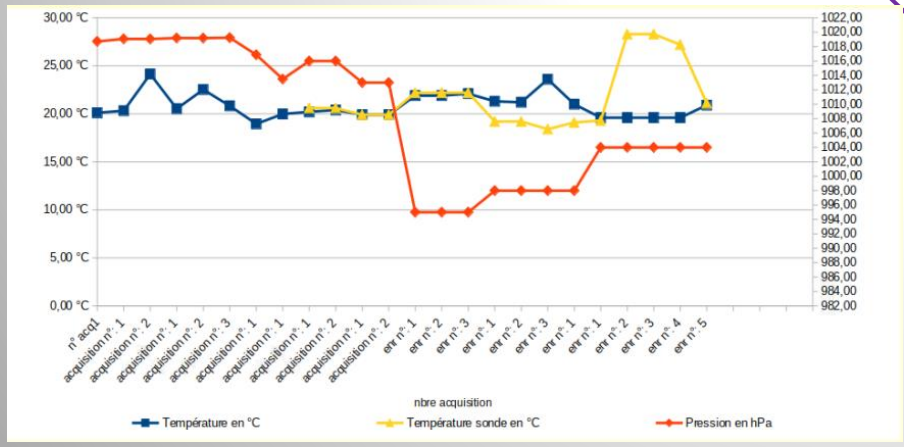
```
def acquisition():
    global index
    index = index + 1
    l=sauvegarder(temperature,pression,humidite,index,temperatur_ext)
    return l
```

```
def sauvegarder(t,p,h,i,e):
# mise en forme de la date et de l'horaire
    locale.setlocale(locale.LC_TIME,'')

    dat = datetime.datetime.now()
    form_date = dat.strftime("%a %d %B %Y")

    horaire = time.strftime('%Hh %Mmn %Ss')
    wb = load_workbook('/home/pi/Tp_connect/Station_meteo/bme280.xlsx')
    fiche = wb['data']
    ligne = (form_i, form_date, horaire, form_t, form_p, form_h, form_e)
    fiche.append(ligne)
    time.sleep(1)
    wb.save('/home/pi/Tp_connect/Station_meteo/bme280.xlsx')
    return ligne
```

bme.xlsx

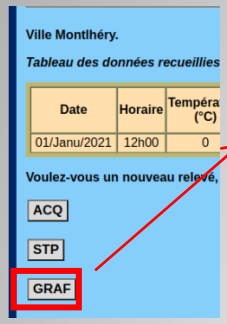
N° acquisition	Date	Heure	Température en °C	Pression en hPa	Taux d'Humidité en %	Température sonde en °C
enr n°: 1	mer. 05 mai 2021	17h 58mn 59s	19.60 °C	1004 hPa	35 %	19.30 °C
enr n°: 2	mer. 05 mai 2021	18h 00mn 14s	19.60 °C	1004 hPa	35 %	28.30 °C
enr n°: 3	mer. 05 mai 2021	18h 01mn 12s	19.60 °C	1004 hPa	35 %	28.30 °C
enr n°: 4	mer. 05 mai 2021	18h 01mn 57s	19.60 °C	1004 hPa	35 %	27.20 °C
enr n°: 5	mer. 05 mai 2021	18h 03mn 18s	20.90 °C	1004 hPa	82 %	21.10 °C

data



A chaque acquisition
Ouverture de l'onglet '**data**' du fichier **.xlsx**
Écriture de la ligne avec les data
Fermeture du fichier

Diagramme de fonctionnement GRAF



appui sur bouton ‘’GRAF’’

meteo_web.py

accès à la fonction **acquerir**

meteo.py

accès à la fonction **acquisition**

```
def acquerir(self, acces="", nom=""):
    nom = cherrypy.session["nom"]
    cherrypy.session["acces"] = acces
    if acces == "ACQ":
        i = 1
        data_bme = meteo.acquisition()
        journe = data_bme[1]
        horair = data_bme[2]
        temper = data_bme[3]
        pressi = data_bme[4]
        humidi = data_bme[5]
        texter = data_bme[6]

        time.sleep(1)
        Glob.abscis.append(Glob.index)
        Glob.ordon.append(temper)
        afficher(Glob.abscis,Glob.ordon)
        Glob.index = Glob.index + 1

    elif acces == "GRAF":
        return misenpg(Glob.dico_motif["pageAffiche"].format(nom))
```

```
def acquisition():
    global index
    index = index + 1
    l=sauvegarder(temperature,pression,humidite,index,temperatur_ext)
    return l

    ligne = (form_i, form_date, horaire, form_t, form_p, form_h, form_e)
    return ligne
```

A chaque acquisition un point est crée avec **index** en abscisse et **temp** en ordonnée

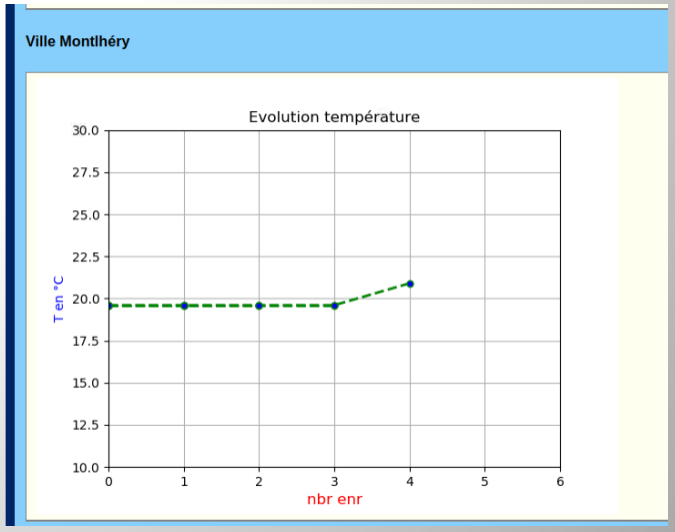
meteo_web.html

```
[*pageAffiche*]
<h3> Ville {0} </h3>
<h1><img src="/annexes/graph3.png"></h1>
"""
```

```
def afficher(x, y):
    color = 'green'

    pyplot.plot(x, y, color, linestyle = 'dashed', linewidth=2, marker='o', markerfacecolor='blue', markersize=5)
    pyplot.xlim(0,6)
    pyplot.xlabel('nbr enr', color = 'red', fontsize = 12)
    pyplot.ylim(10,30)
    pyplot.ylabel('T en °C', color = 'blue', fontsize = 10)
    pyplot.grid(b=True)
    pyplot.title('Evolution température')
    pyplot.savefig('/home/pi/Tp_connect/Station_meteo/annexes/graph3.png')
```

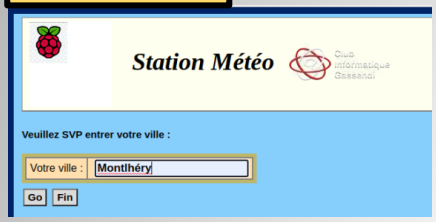
fonction qui crée et affiche les relevés de température sur un graphe

Enregistrement de chaque pt

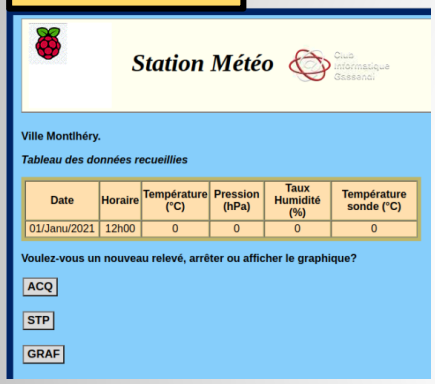


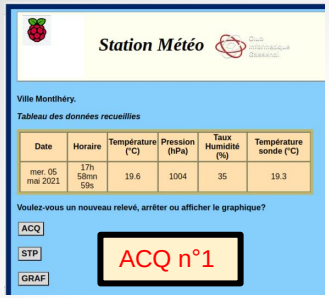
Aperçu des pages station météo

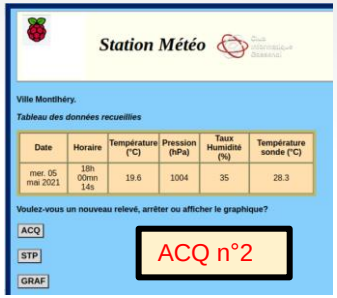
Page d'accueil

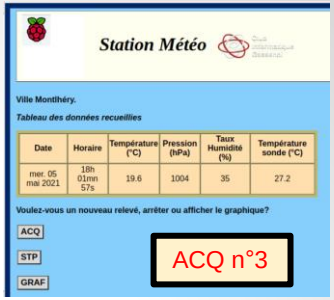


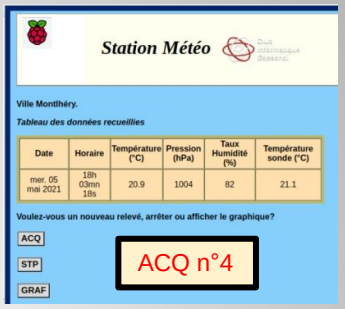
Page de travail

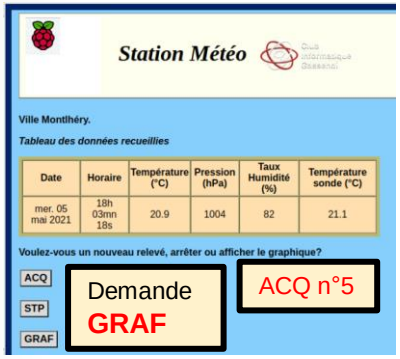




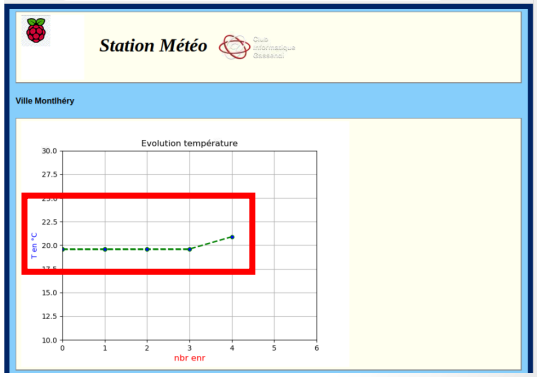


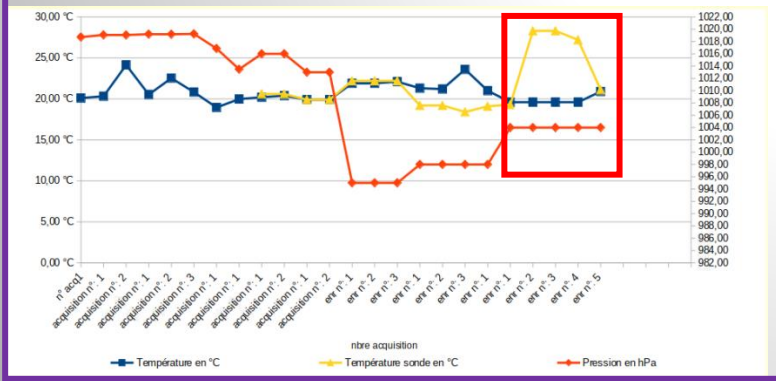






Page Affiche





enr n°: 1	mer. 05 mai 2021	17h 58mn 59s	19,60 °C	1004 hPa	35 %	19,30 °C
enr n°: 2	mer. 05 mai 2021	18h 00mn 14s	19,60 °C	1004 hPa	35 %	28,30 °C
enr n°: 3	mer. 05 mai 2021	18h 01mn 12s	19,60 °C	1004 hPa	35 %	28,30 °C
enr n°: 4	mer. 05 mai 2021	18h 01mn 57s	19,60 °C	1004 hPa	35 %	27,20 °C
enr n°: 5	mer. 05 mai 2021	18h 03mn 18s	20,90 °C	1004 hPa	82 %	21,10 °C

Enregistrement LibreOffice Calc
fichier **bme280.xlsx**

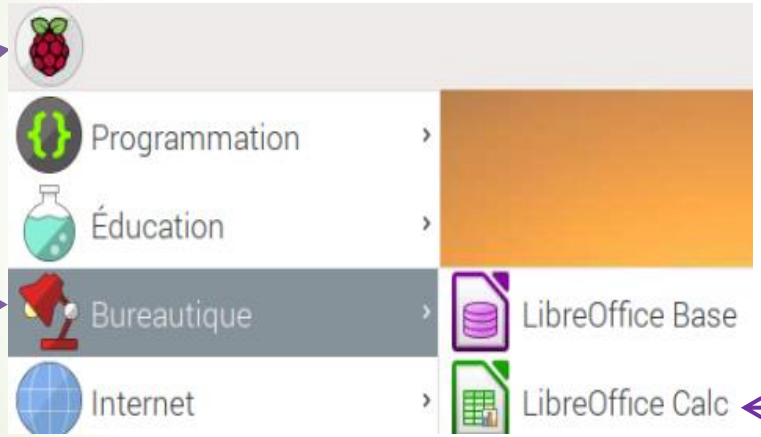
Annexe LibreOffice calc

Créer un fichier tableur (*Workbook*) avec LibreOffice calc: **bme280.xlsx**

En mode « Graphique »

1. Cliquez **Menu**

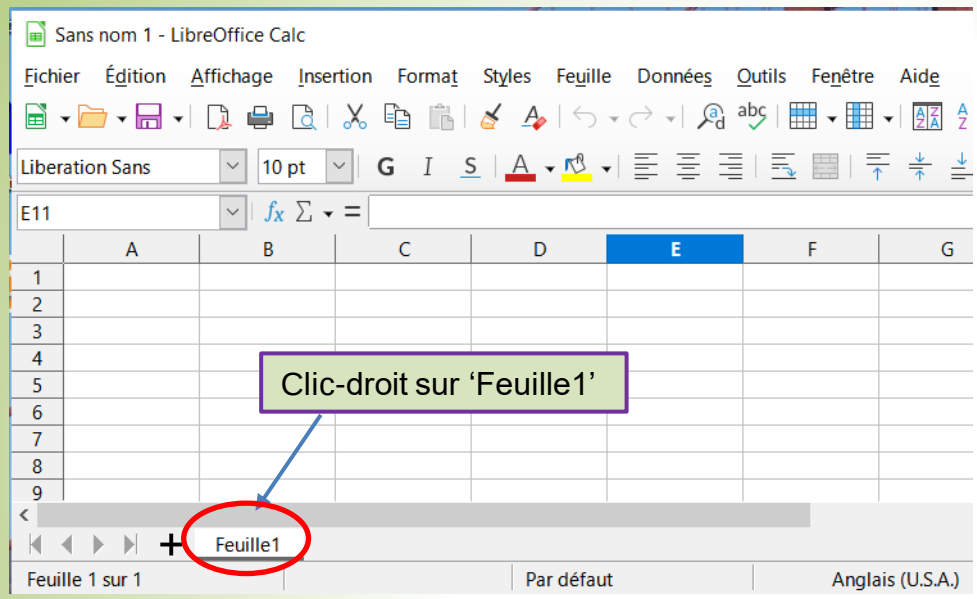
2. Choisir **Bureautique**

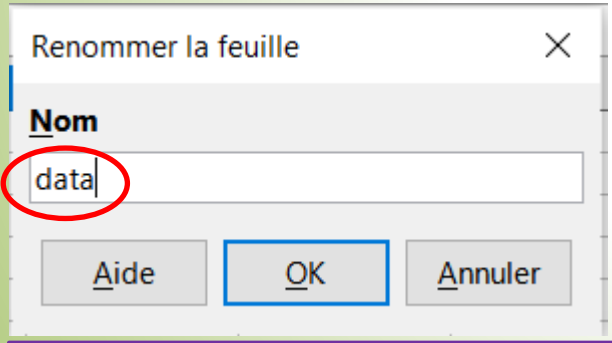


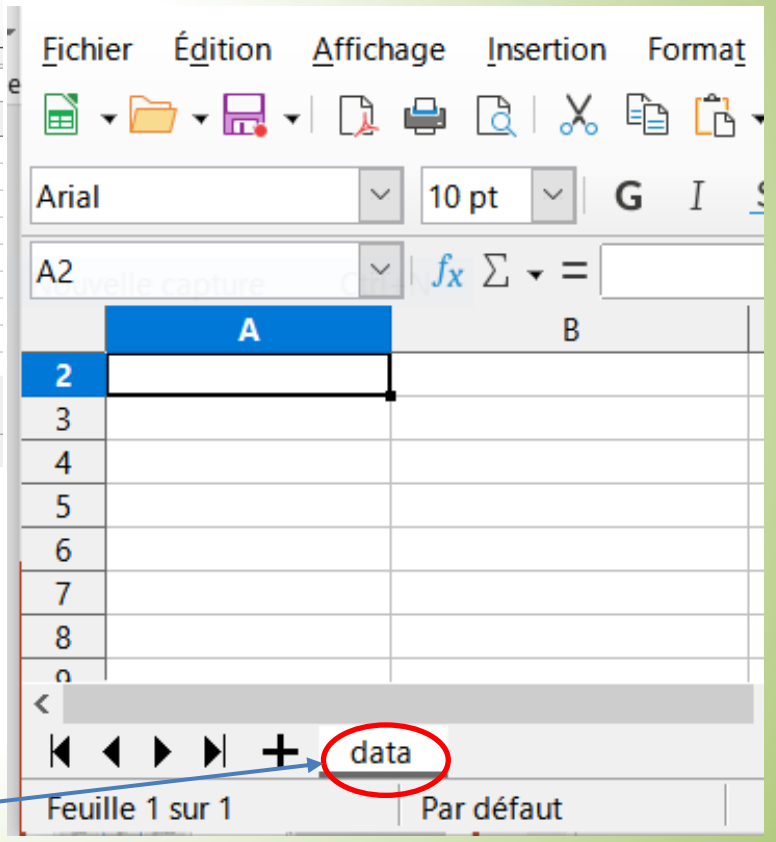
Ouvrir **LibreOffice** Calc

Annexe LibreOffice calc

Changer le nom de l'onglet



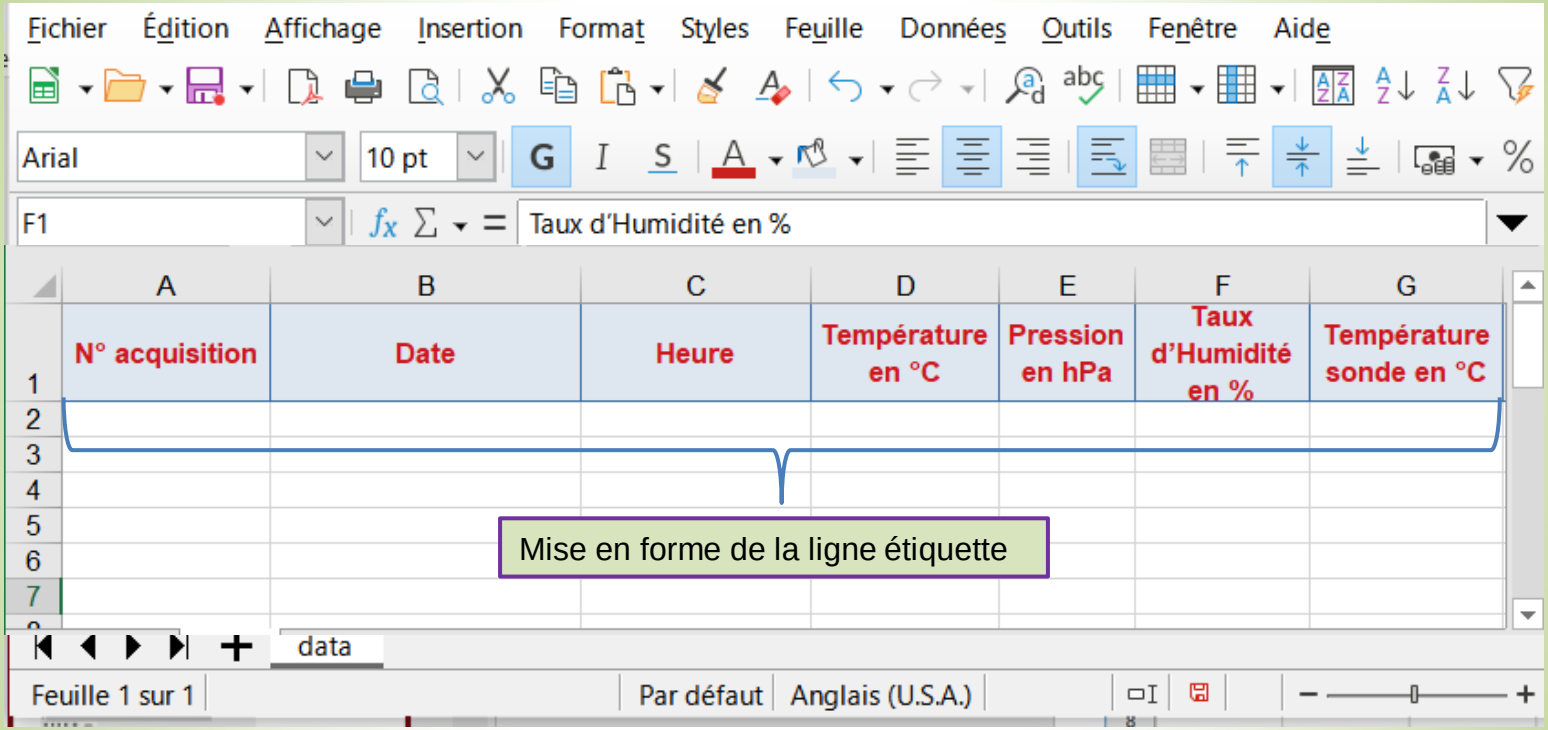




Renommer la feuille: Feuille1 ➔ data

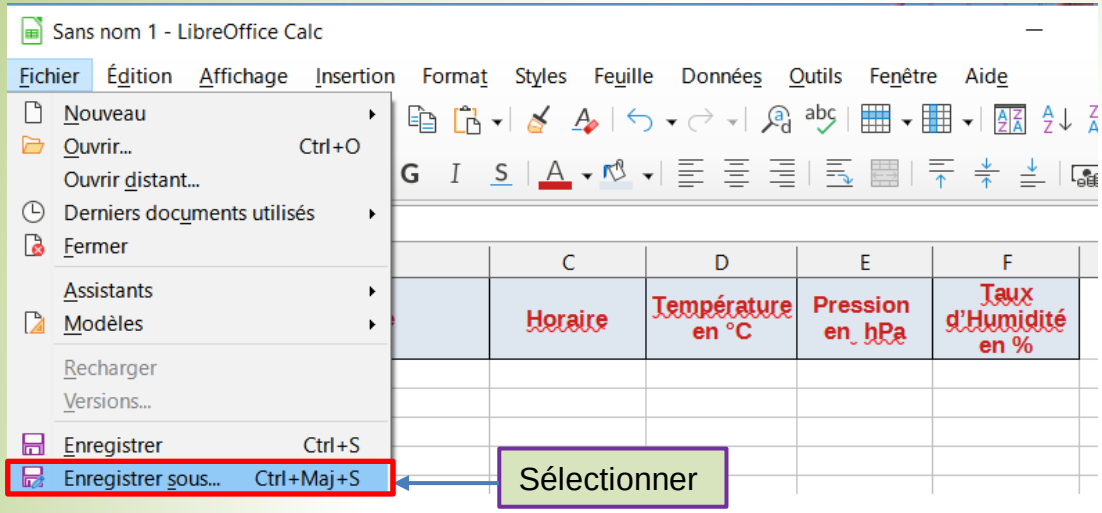
Annexe LibreOffice calc

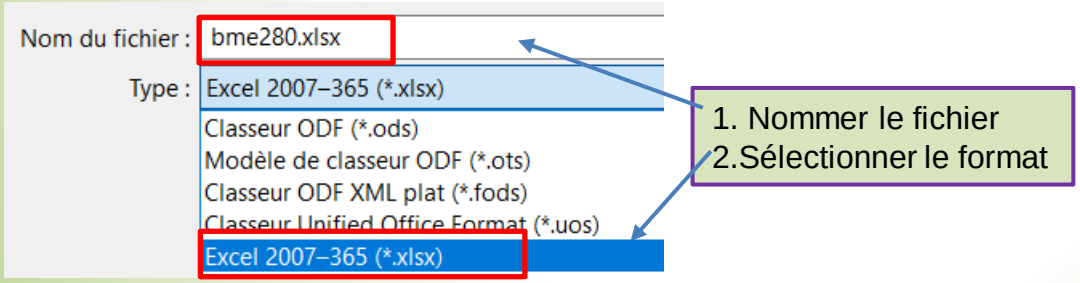
Initialiser les colonnes de la 1ére ligne

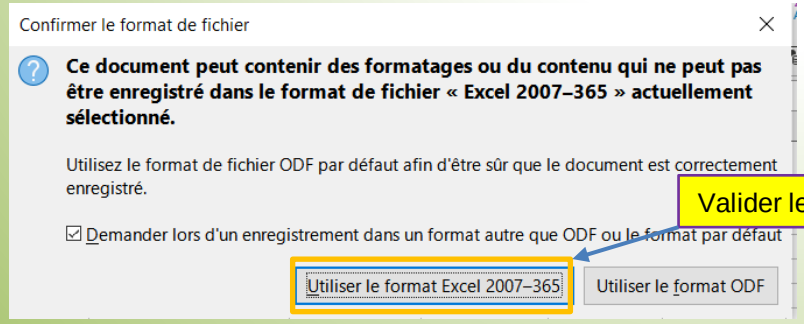


Annexe LibreOffice calc

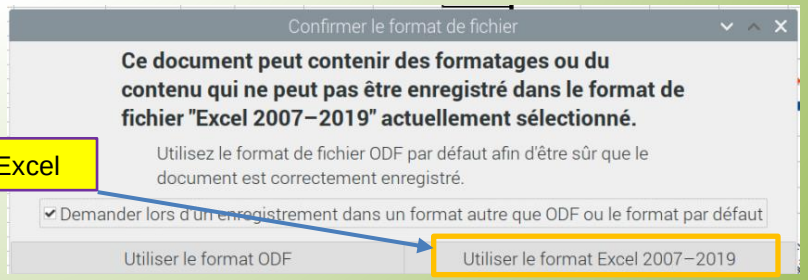
Nommer le fichier sous: bme280 au format xlsx







Valider le format Excel



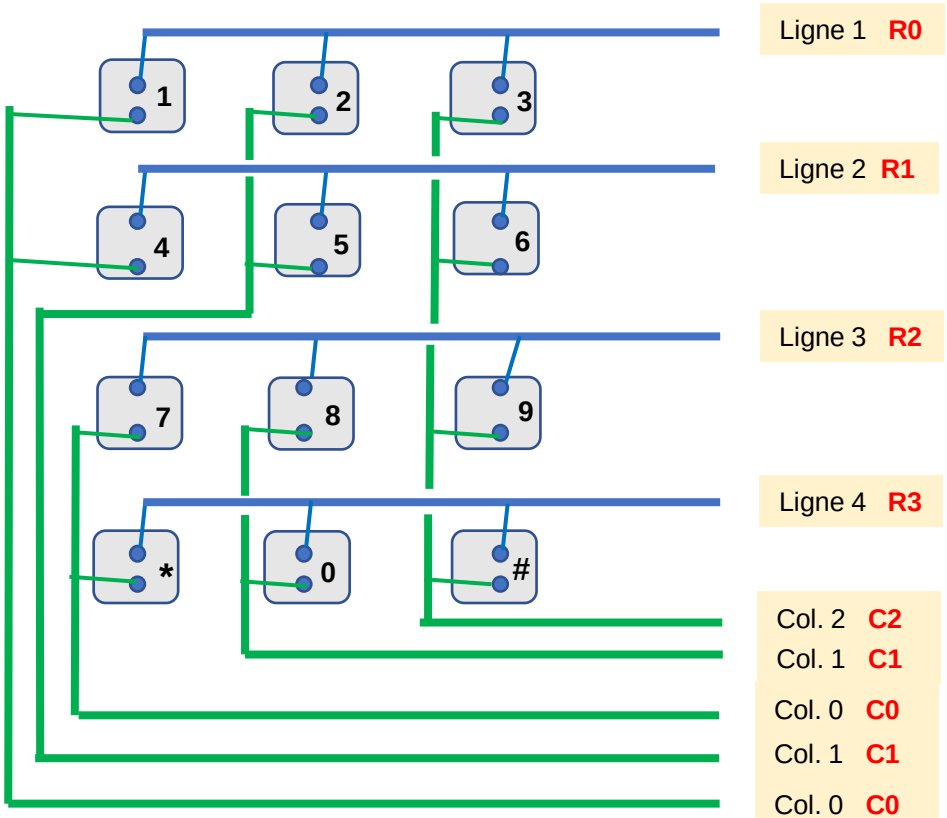
Présentation composants

Clavier 12 touches

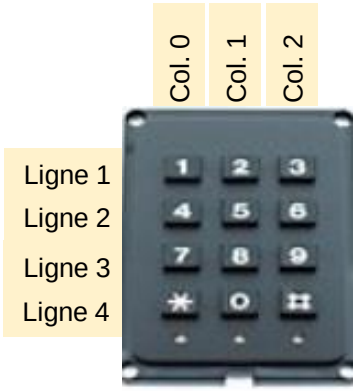
Un clavier est composé de **n** boutons poussoir disposés sous forme matricielle (**m** lignes de **p** colonnes)
La pression sur la touche met en relation la ligne et la colonne correspondante.

Le clavier que nous utiliserons est constitué de 4 lignes de 3 colonnes (12 touches)

Représentation schématique



Représentation physique



Clavier KB 12M