

# Revue de Projet Finale

Projet Luminaire

Maxence x  
BTS SN – Lycée x

# Sommaire :

- Présentation du projet
- Présentation de mon contrat
- Diagramme de Gantt
- Diagramme des exigences
- Diagramme de cas d'utilisation
- Diagramme de classe prt.1
- Diagramme de classe prt.2
- Tous les fichiers utilisés
- Diagramme de séquence
- Diagramme de thread
- Page de connexion et BDD
- Customisation des fenêtres
- Image et Canvas
- Update automatique des images et labels
- Les fichiers textes
- Envoi de mail en cas de panne batterie
- Envoi de donnée sur TTN
- Calcul tension, courant et charge batterie
- Actualiser la barre de progression
- Allumage progressif avec un slider
- Afficher l'évolution de la batterie
- Montage et simulation
- Mise en place du système et démonstration
- Questions



**1**

# **Présentation du projet**

- Un lampadaire solaire dit “SmartLight” de l’entreprise Fonroche
- Lampadaire 100% solaire ne nécessitant pas de raccordement EDF ni de creuser des tranchées ou d’armoire électrique
- Puissance du luminaire de 60W et puissance photovoltaïque de 190W
- Éclairage automatique
- Écran tactile sur le luminaire pour la maintenance



# Les techniciens et leur contrat



**Romain x**

PyQt

Gestion d'une application  
distante avec connexion à une  
BDD



**Maxence x**

Tkinter

Gestion du luminaire via un  
écran tactile connecté à une  
Raspberry



**Kanouni x**

Electronique

Gestion du luminaire avec des  
sondes et des modules



**Faris x**

Lora, TTN et BDD

Récupération des informations  
du luminaire et stocker les  
données sur une BDD



**2**

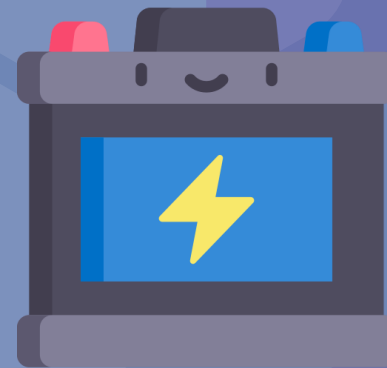
## **Présentation de mon contrat**



Création d'une fenêtre  
graphique pour la maintenance  
du luminaire



Création d'un fenêtre qui  
s'actualise toute seule



Récupération de la tension de la  
batterie est courant dans les  
LEDs



Interaction avec la fenêtre  
graphique via un écran tactile



Et tout ça sur une  
Raspberry Pi...





























**3**

## **Diagramme de Gantt**



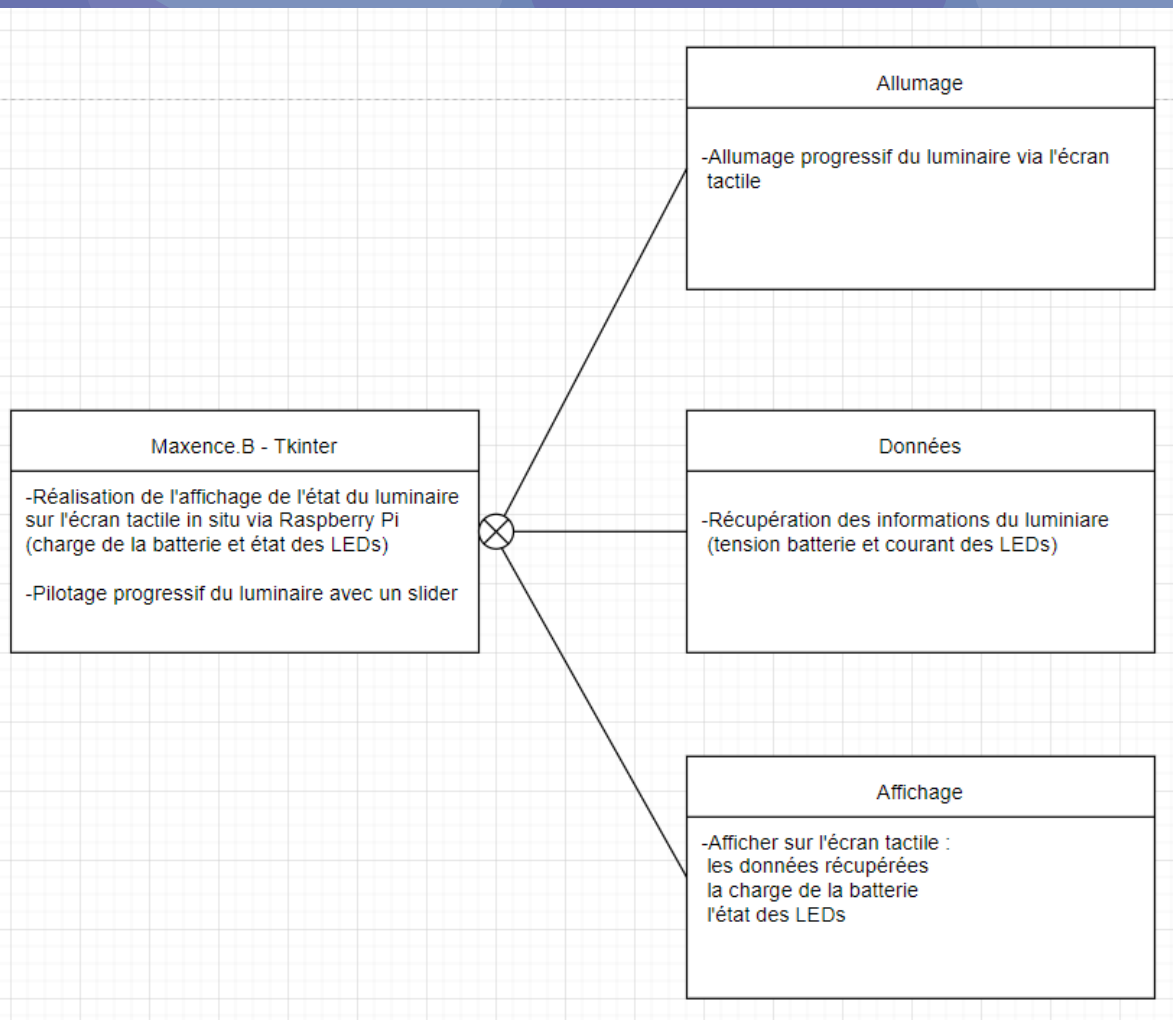
	!	PI...	Nom de tâche	Durée	Début	Fin
1			<u>Contrat Tkinter</u>	54 jours	04/01/2022	15/06/2022
2			<u>Diagramme de Gantt</u>	6 h	04/01/2022	04/01/2022
3			<u>Diagramme utilisation</u>	2 h	04/01/2022	04/01/2022
4			<u>Diagramme des exigences</u>	2 h	05/01/2022	05/01/2022
5			<u>Prise en main raspberry pi 4</u>	8 h	05/01/2022	07/01/2022
6			<u>Bon de commande</u>	2 h	07/01/2022	07/01/2022
7			<u>Revue 1</u>	1 jour	21/01/2022	21/01/2022
8			<u>Prise en main de Python</u>	8 h	25/01/2022	25/01/2022
9			<u>Prise en main OOP Python</u>	10 h	26/01/2022	28/01/2022
10			<u>Premiers pas avec Tkinter</u>	2 h	28/01/2022	28/01/2022
11			<u>1er schéma de ma fenêtre graphique</u>	40 min	28/01/2022	28/01/2022
12			<u>Réalisation de ma 1ère fenêtre en OOP</u>	5 h	28/01/2022	01/02/2022
13			<u>Prise en main du module YI-40</u>	2 h	01/02/2022	01/02/2022
14			<u>Prise en main du PCF8591</u>	3 h	01/02/2022	01/02/2022
15			<u>Recherche sur le bus I2C</u>	4 h	01/02/2022	02/02/2022
16			<u>Essais du bus I2C</u>	1 h	02/02/2022	02/02/2022
17			<u>Ajout d'un slider et d'une progressbar</u>	2 h	02/02/2022	02/02/2022
18			<u>Controler des LED avec mon slider</u>	4 h	02/02/2022	04/02/2022
19			<u>Revue 2</u>	1 jour	05/04/2022	05/04/2022
20			<u>Schéma de ma page connexion</u>	40 min	06/04/2022	06/04/2022
21			<u>Réalisation de ma fenêtre connexion</u>	10 h	06/04/2022	08/04/2022
22			<u>Réalisation d'une fenêtre en cas de mauvais mot de passe</u>	1 h	08/04/2022	08/04/2022
23			<u>Réalisation d'une fenêtre en cas d'aucune donnée rentré</u>	1 h	08/04/2022	08/04/2022
24			<u>Réalisation d'une fenêtre en cas de donnée correct</u>	1 h	08/04/2022	08/04/2022
25			<u>Lien entre ma fenêtre connexion et ma fenêtre principale</u>	30 min	08/04/2022	08/04/2022
26			<u>Récupération de la tension de la batterie</u>	2 h	08/04/2022	26/04/2022
27			<u>Récupératio du courant des LEDs</u>	2 h	26/04/2022	26/04/2022
28			<u>Calcul et affichage de la tension</u>	3 h	26/04/2022	26/04/2022
29			<u>Calcul et affichage du courant</u>	3 h	26/04/2022	27/04/2022

30		Calcul et affichage du % de la batterie	3 h	27/04/2022	27/04/2022
31		Mise en place de photo dynamiques	6 h	27/04/2022	29/04/2022
32		Mise en place de labels dynamiques	6 h	29/04/2022	29/04/2022
33		Mise en place de la progressbar dynamique	4 h	29/04/2022	03/05/2022
34		Intégration du programme de Faris dans la fenêtre	2 h	03/05/2022	03/05/2022
35		Crée et écrire dans un fichier log la date et l'heure pour laquelle la batterie est à plat	1 h	03/05/2022	03/05/2022
36		Réalisation d'un système qui envoie un mail lorsque la batterie est vide	6 h	03/05/2022	04/05/2022
37		Envoi dans le mail le fichier log	30 min	04/05/2022	04/05/2022
38		Réalisation d'une fenêtre qui dit qu'on envoie un mail	1 h	04/05/2022	04/05/2022
39		Mise en place de 3 Threads Timer	4 h	04/05/2022	06/05/2022
40		Recherche sur la bibliothèque matplotlib	5 h	06/05/2022	06/05/2022
41		Schéma de ma fenêtre de l'évolution du % de batterie	40 min	06/05/2022	06/05/2022
42		Création d'un fichier data.txt pour écrire le % de batterie	2 h	06/05/2022	10/05/2022
43		Réalisation de ma fenêtre avec l'évolution de la batterie avec data.txt	10 h	10/05/2022	11/05/2022
44		Réalisation d'un diagramme entre les fichiers	2 h	11/05/2022	11/05/2022
45		Réalisation d'un diagramme de séquence	6 h	11/05/2022	13/05/2022
46		Réalisation diagramme de classe	6 h	13/05/2022	17/05/2022
47		Rédaction du rapport	2 jours	17/05/2022	20/05/2022
48	 	Revue 3	1 jour	31/05/2022	31/05/2022
49		Peaufinement	5 h	01/06/2022	01/06/2022
50		Intégration et essais	1 jour	01/06/2022	03/06/2022
51		Maquette fonctionnelle	1 jour	03/06/2022	07/06/2022
52		Rendu Dossier	1 jour	07/06/2022	08/06/2022
53	 	Revue Finale	1 jour	15/06/2022	15/06/2022



**4**

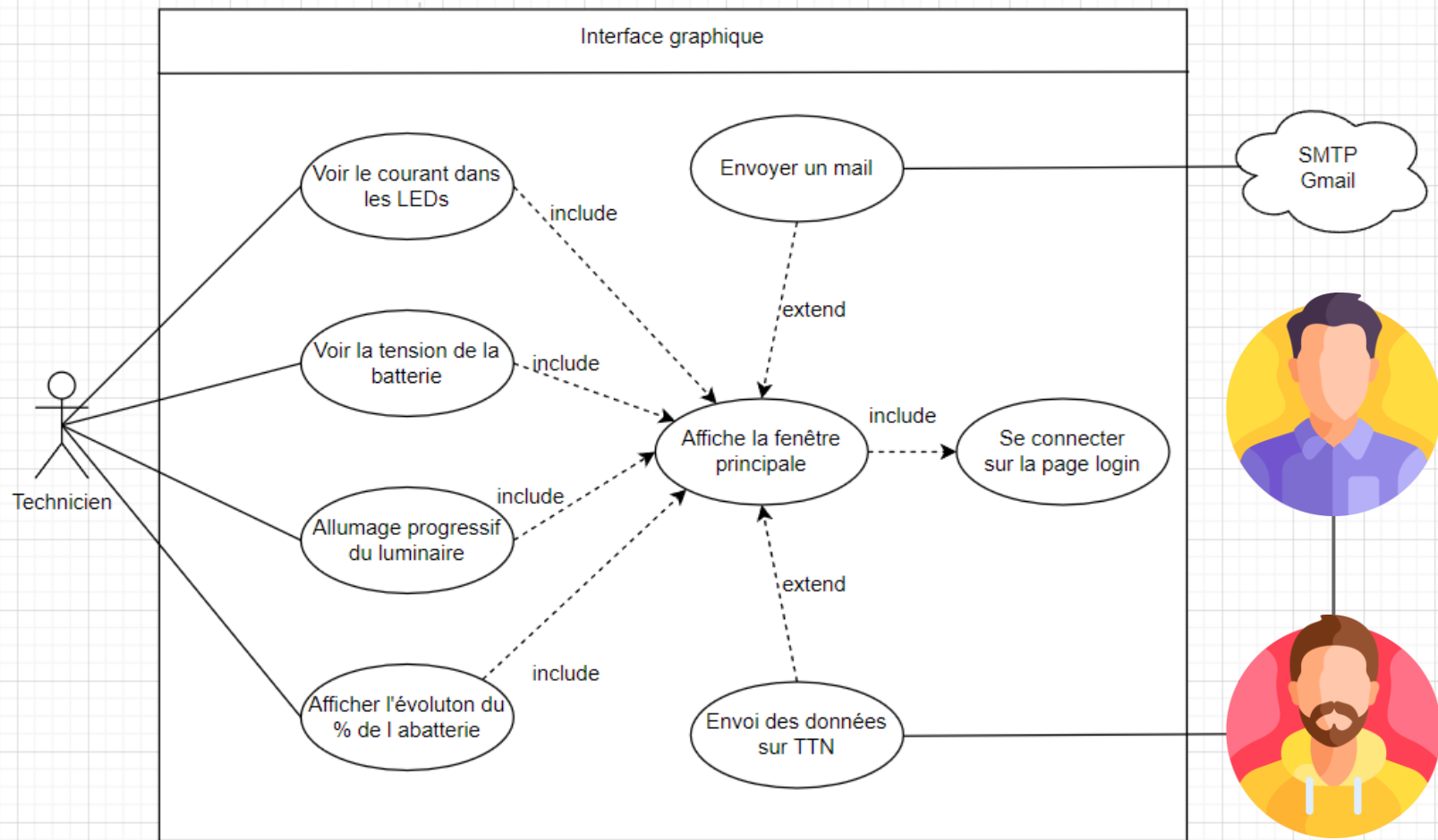
## **Diagramme des exigences**





**5**

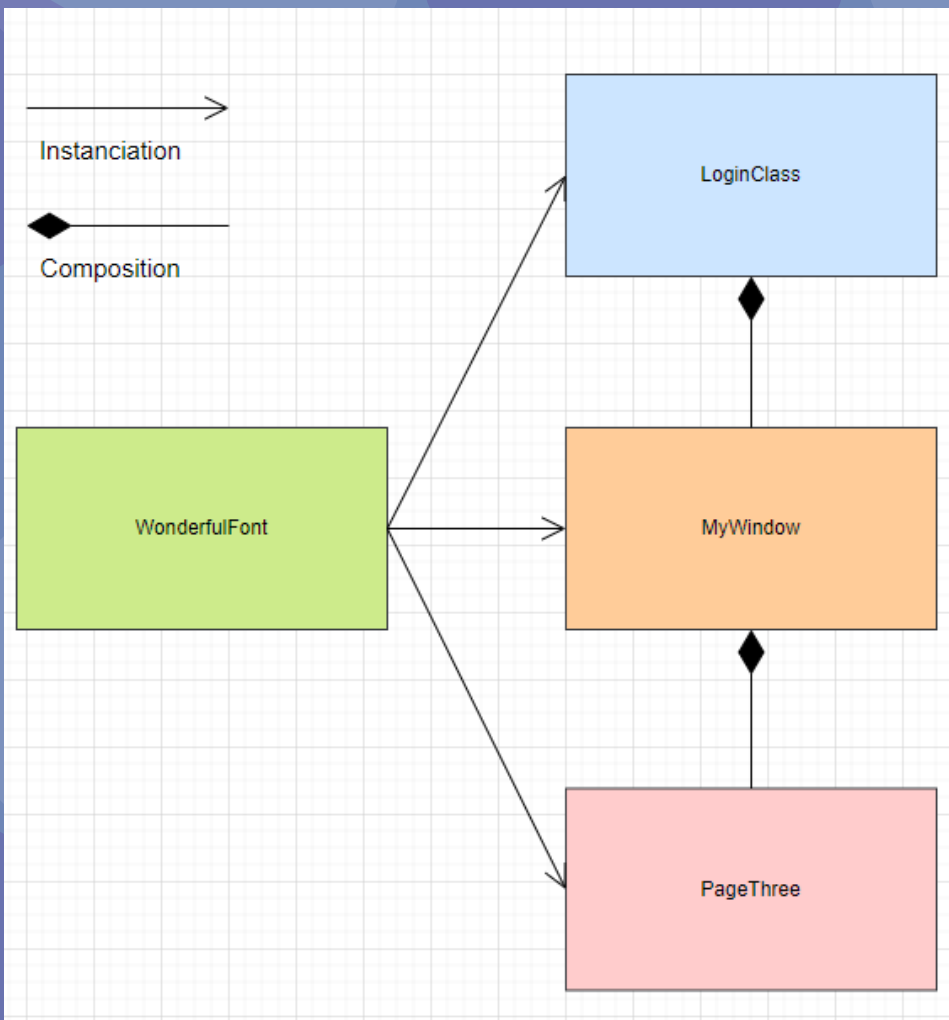
## **Diagramme de cas d'utilisation**





6

## Diagramme de classe prt.1







**7**

## **Diagramme de classe prt.2**

## LoginClass (Tk)

```

- obj_font WonderfulFont ()

- close_button (button)
- canvas_logo (canvas)
- canvas_login (canvas, image)
- canvas_password (canvas, image)
- entry_login (entry)
- entry_password (entry)
- button_login (button)

- time ()

+ time (current_time, clock)

+ win_warn (frame_warn_0, frame_warn_a, frame_warn_b,
            frame_warn_c, frame_warn_d, close_button_warning,
            lab_warn, canvas_warn, warn_txt, ok_choice_warn)
+ select_warn (choice, destroy)

+ win_error (frame_error_0, frame_error_a, frame_error_b,
            frame_error_c, frame_error_d, close_button_warning,
            lab_error, canvas_error, error_txt, ok_choice_error)
+ select_error (choice, destroy)

+ win_info (frame_info_0, frame_info_a, frame_info_b,
            frame_info_c, frame_info_d, close_button_info,
            lab_info, canvas_info, info_txt, ok_choice_info)
+ select_info (choice, destroy, obj_mywin MyWindow())

+ logpass (username, password, win_warn(), win_error(), win_info())

```

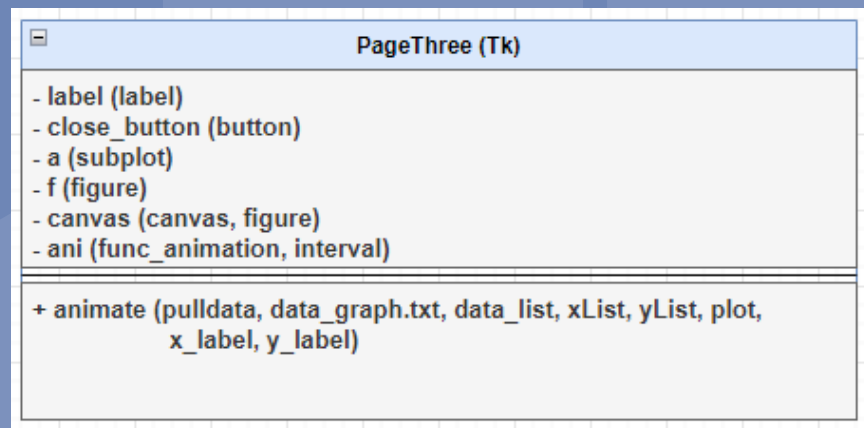
## WonderfulFont (container)

```

- font_1 (font)
- font_2 (font)
- font_3 (font)
- font_4 (font)

- color_1 (color)
- color_2 (color)
- color_3 (color)
- color_4 (color)
- color_5 (color)
- color_6 (color)
- color_7 (color)
- color_8 (color)
- color_9 (color)
- color_10 (color)
- color_11 (color)
- color_12 (color)
- color_13 (color)
- color_14 (color)
- color_15 (color)
- color_16 (color)
- color_17 (color)
- color_18 (color)
- color_19 (color)

```



## MyWindow (Toplevel)

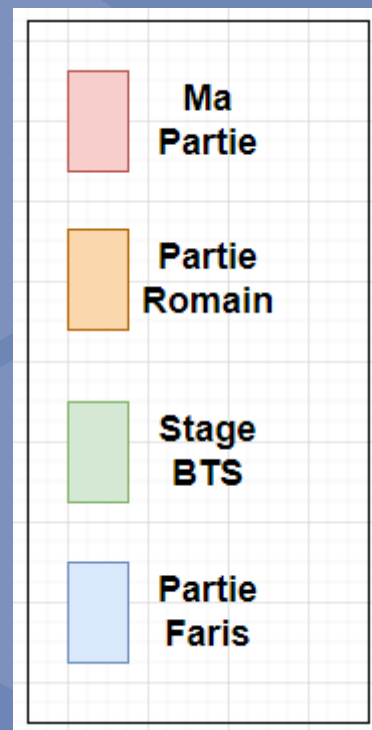
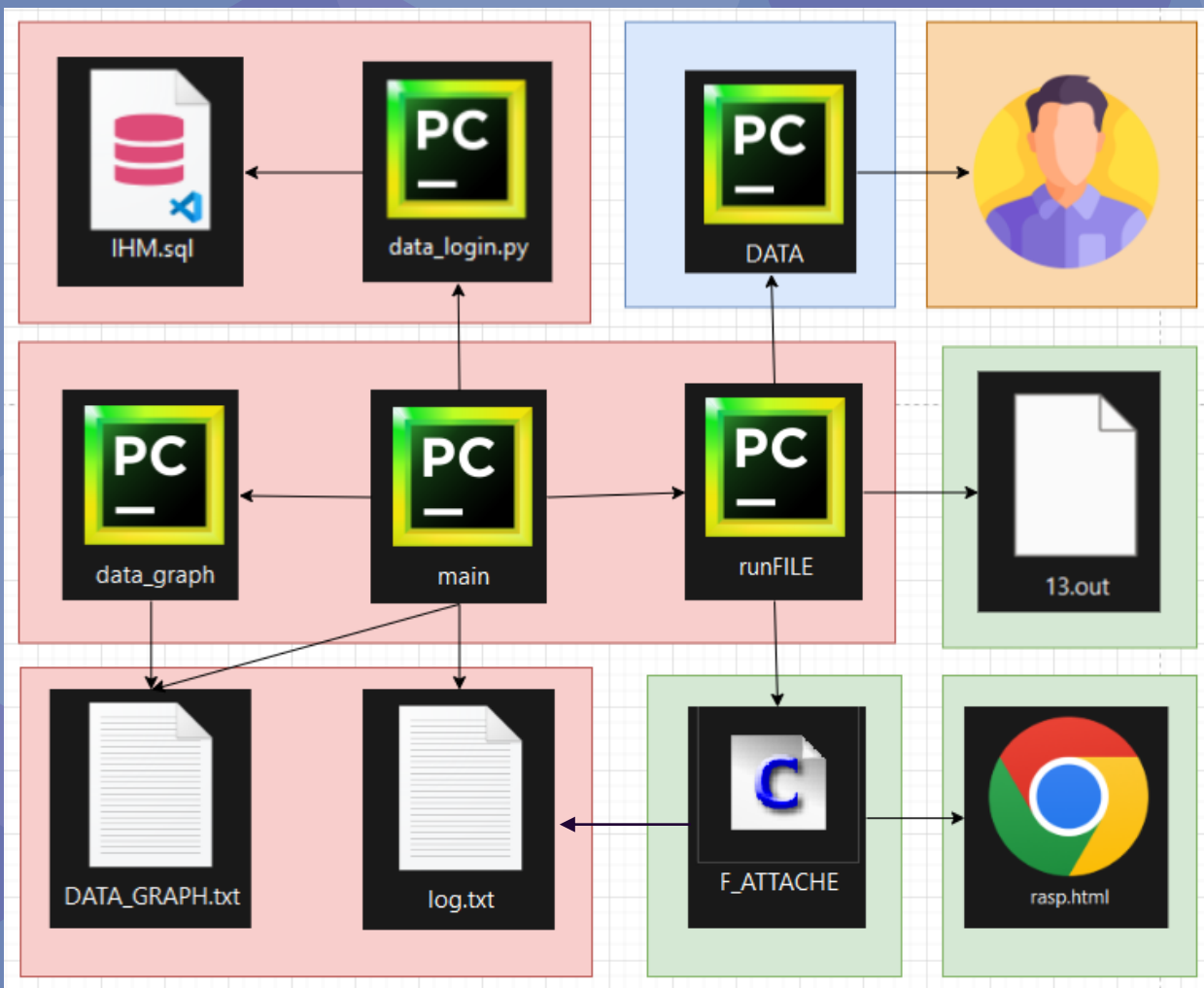
- obj\_font WonderfulFont ()
- canvas\_logo (canvas, logo)
- slider (scale)
- bar\_lab (label)
- bar\_style (style())
- bar (progressbar)
- close\_button (button)
- canvas\_bat (canvas, image)
- canvas\_sli (canvas, image)
- graph\_button (button)
- canvas\_sli (canvas)
- canvas\_bat (canvas)
- thread\_graph (thread, timer)
- thread\_data\_bdd (thread, timer)
- thread\_mail (thread, timer)
- bus (smbus)
- adress (smbus)
- blue\_led (pwmlcd)
- file\_graph (file)
- read\_AIN0 ()
- read\_AIN1 ()
- value\_bat ()
- status ()
- image\_bulb ()
- image\_bar ()
- time ()

- + write\_graph\_data (n, file\_graph)
- + update\_data (BigData())
- + win\_dead\_bat (canvas\_dead\_bat, dead\_bat\_txt,  
ok\_choice\_dead\_bat)
- + select\_dead\_bat (choice, destroy)
- + send\_mail\_problem (current\_time\_full, charge\_integer, file\_log,  
mail\_sent())
- + image\_bar (canvas)
- + image\_bulb (canvas)
- + value\_bat (difference\_max, difference\_range, charge,  
charge\_integer, bar\_update)
- + status (blue\_led, status\_lab)
- + button\_close (gpiocleanup, exit(), destroy())
- + button\_clicked (data\_graph(), obj\_graph PageThree())
- + changevalueled (value, led\_value, blue\_led)
- + read\_ain0 (read\_0, value\_tension, tension\_lab)
- + read\_ain1 (read\_1, value\_current, current\_lab)
- + time (current\_time, clock)



8

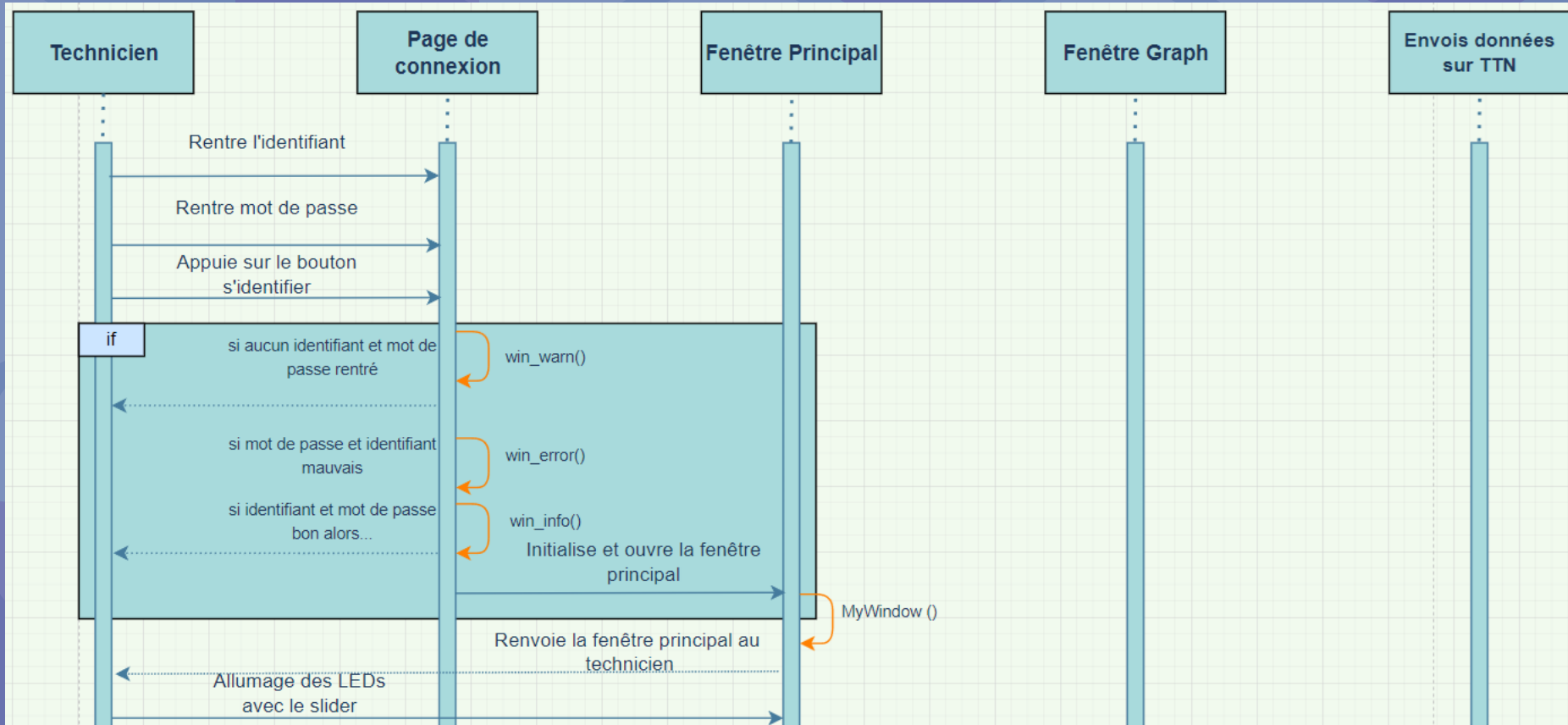
**Tous les fichiers utilisés**



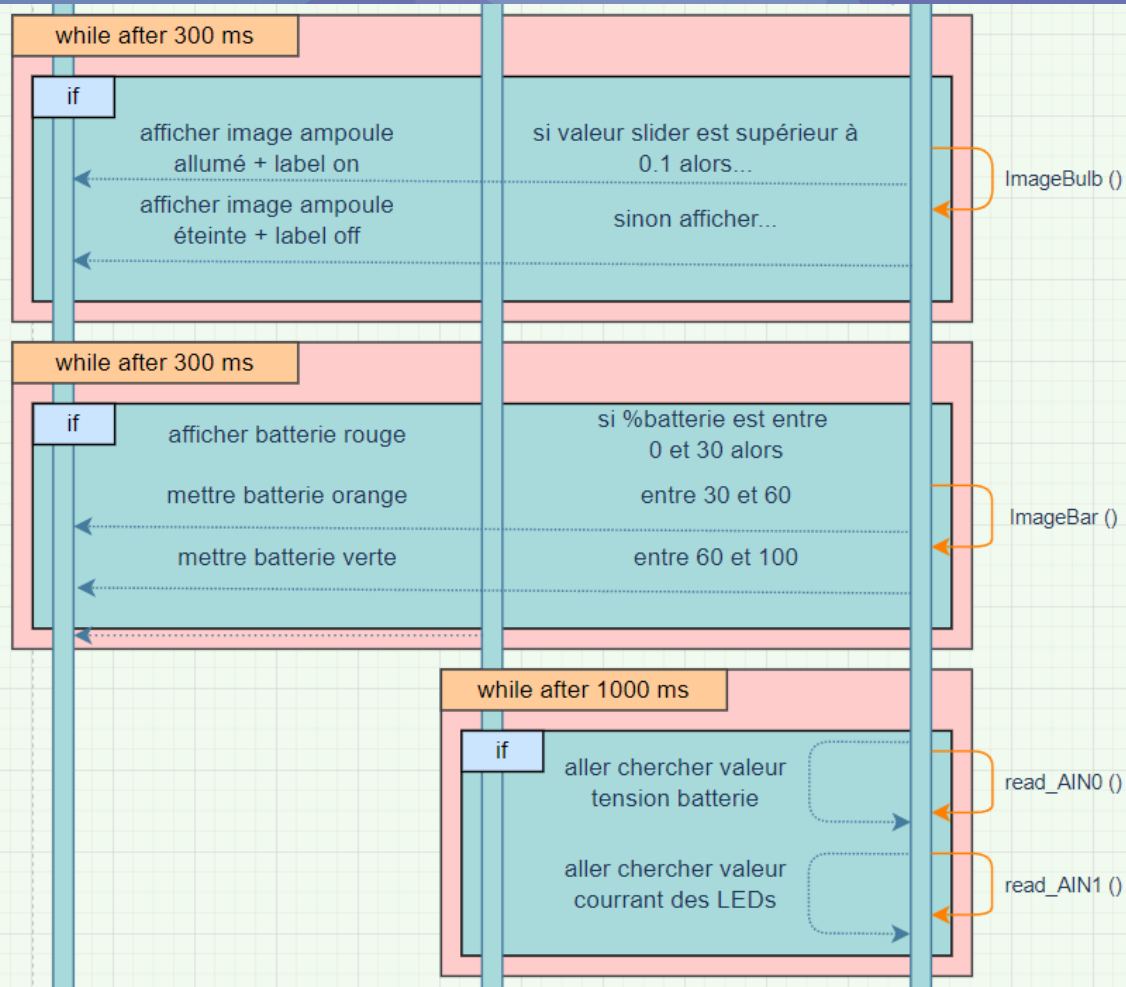


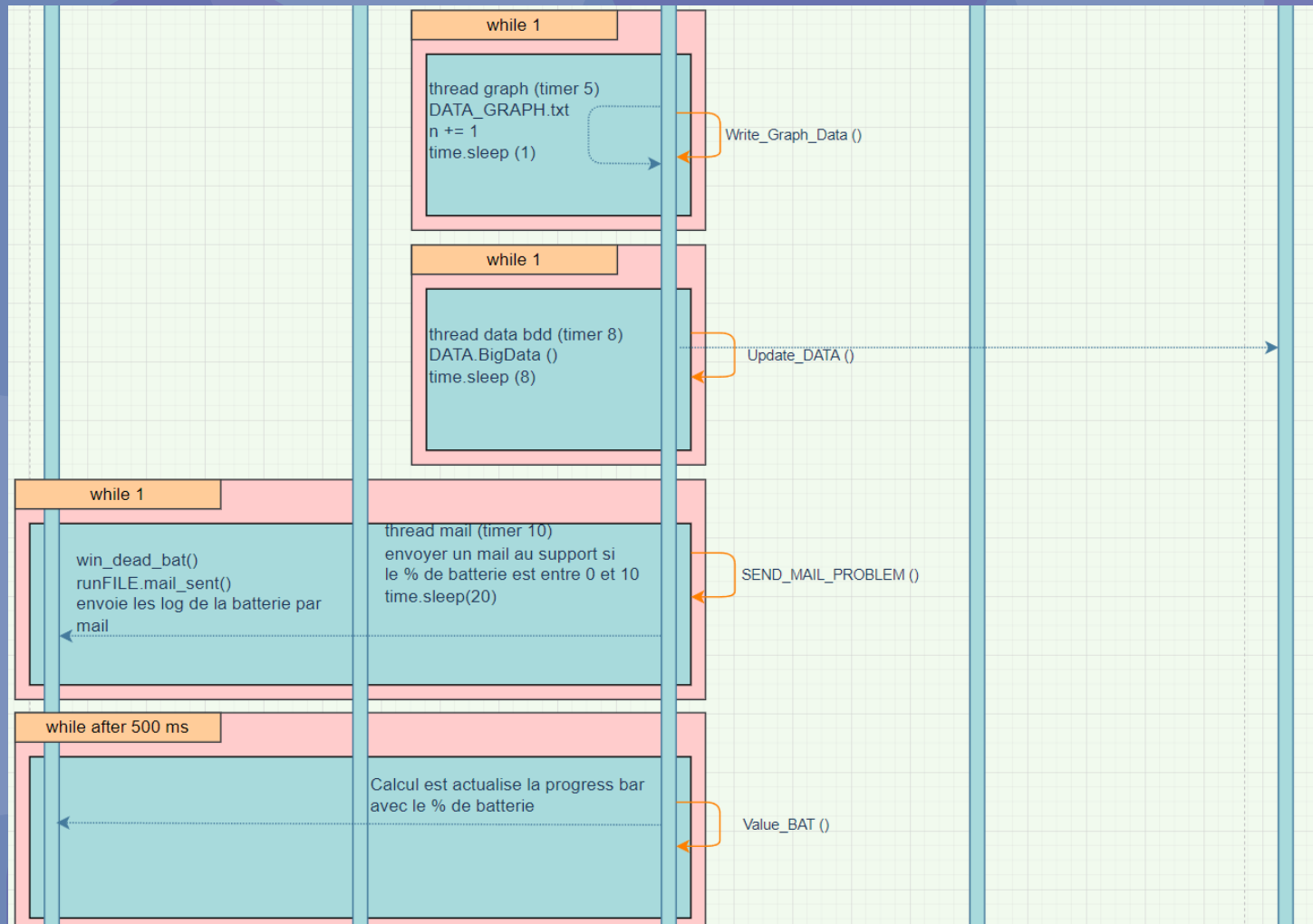
9

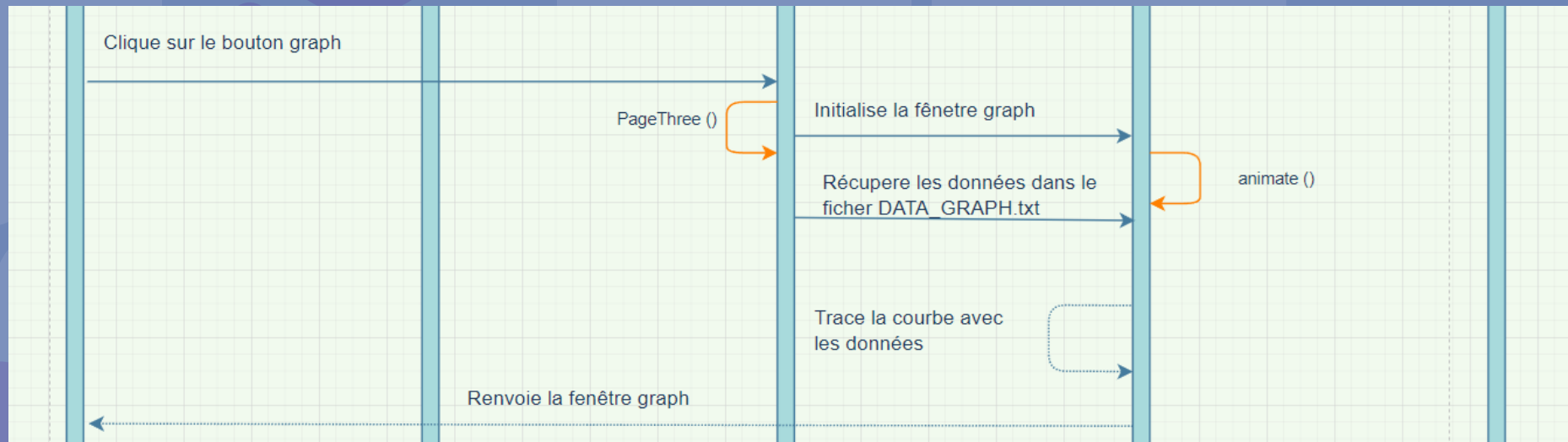
## Diagramme de séquence







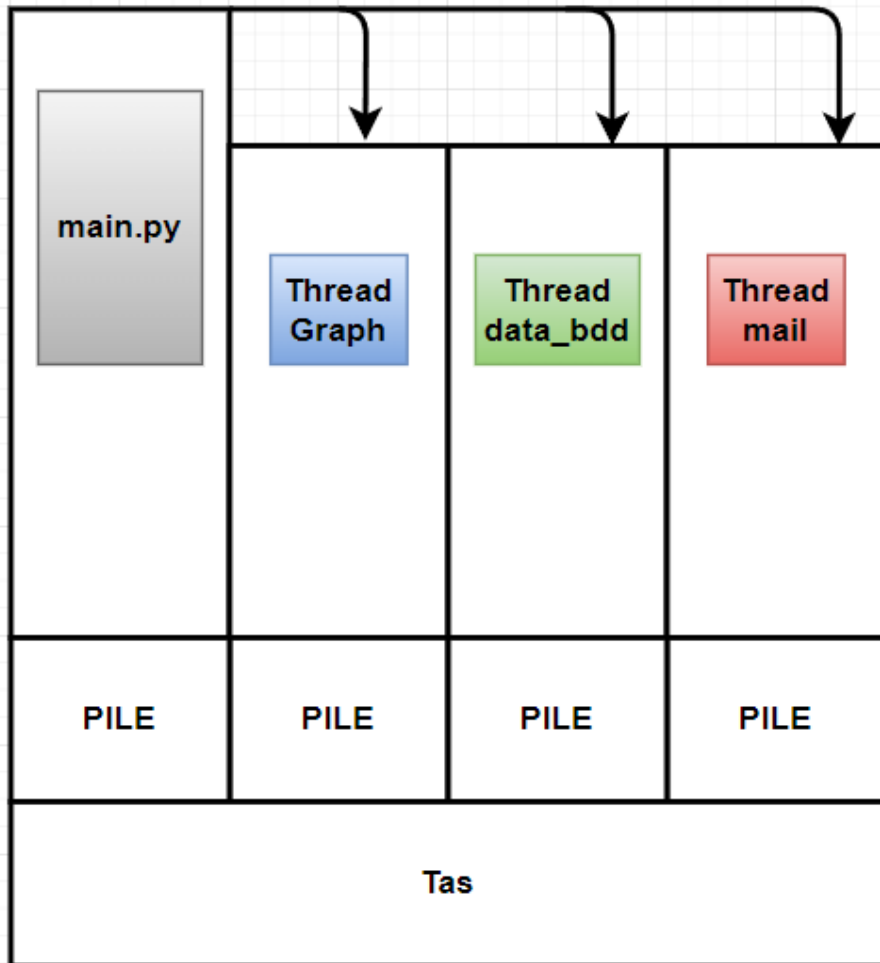






10

## Diagramme de thread





**11**

**Page de connexion**



Le technicien doit rentrer un identifiant et un mot de passe pour pouvoir accéder à la page principale

Voici à quoi ressemble ma page connexion :

Page de connexion

X



Identifiant :

tech



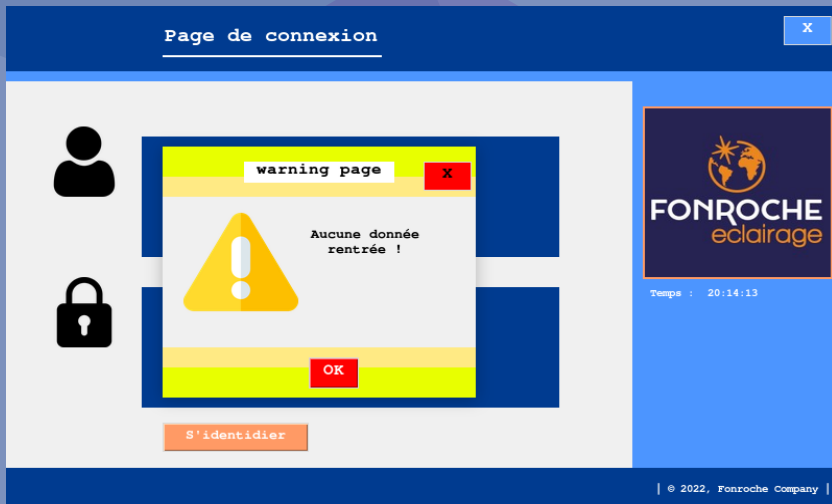
Mot de passe :

\*\*\*

S'identifier

  
FONROCHE  
éclairage  
Temps : 22:47:28

© 2022, Fonroche Company



```
if (self.username == "" and self.password == ""):  
    self.win_warn()
```



```
elif myresult == None:  
    self.win_error()
```





```
else:  
    self.win_info()
```

```
def select_info(self, choice):  
    if choice == "OK":  
        self.info.destroy()  
        self.obj_mywin = MyWindow(self)
```

```
def logpass(self):  
    self.username = self.entry_login.get()  
    self.password = self.entry_password.get()  
  
    mydb = mysql.connector.connect(**data_login.config)  
    mycursor = mydb.cursor(dictionary=True)  
    mycursor.execute("select * from login_table where identifiant = '"+self.username+"' and mot_de_passe = '"+self.password+"';")  
    myresult = mycursor.fetchone()
```

```
import mysql.connector  
  
config = {  
    'user': 'maintenance',  
    'password': 'raspberry',  
    'host': '127.0.0.1',  
    'database': 'IHM'  
}
```



**12**

**Base de donnée locale**

Fichier Édition Onglets Aide

```
mxe@raspberrypi:~ $ mysql -u maintenance -p
```

```
Enter password:
```

```
Welcome to the MariaDB monitor.  Commands end with ; or \g.
```

```
Your MariaDB connection id is 30
```

```
Server version: 10.5.15-MariaDB-0+deb11u1 Raspbian 11
```

```
Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.
```

```
Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.
```

```
MariaDB [(none)]> use IHM
```

```
Reading table information for completion of table and column names
```

```
You can turn off this feature to get a quicker startup with -A
```

```
Database changed
```

```
MariaDB [IHM]> show tables;
```

```
+-----+
```

```
| Tables_in_IHM |
```

```
+-----+
```

```
| login_table   |
```

```
+-----+
```

```
1 row in set (0.001 sec)
```

```
MariaDB [IHM]> select * from login_table;
```

```
+----+-----+-----+
```

```
| id | identifiant | mot_de_passe |
```

```
+----+-----+-----+
```

```
|  1 | tech       | 123          |
```

```
+----+-----+-----+
```

```
1 row in set (0.001 sec)
```

```
MariaDB [IHM]> 
```



**13**

## **Customisation des fenêtres**

```
self.frame_a = tk.Frame(self, height=500, width=800, bg=self.obj_font.color_6)
self.frame_a.place(x=0, y=0)
self.frame_b = tk.Frame(self, height=500, width=220, bg=self.obj_font.color_9)
self.frame_b.place(x=600, y=0)
self.frame_c = tk.Frame(self, height=70, width=800, bg=self.obj_font.color_10)
self.frame_c.place(x=0, y=0)
self.frame_d = tk.Frame(self, height=120, width=400, bg=self.obj_font.color_10)
self.frame_d.place(x=130, y=130)
self.frame_e = tk.Frame(self, height=120, width=400, bg=self.obj_font.color_10)
self.frame_e.place(x=130, y=280)
```

```
self.obj_font = WonderfulFont()
```

```
class WonderfulFont:
    def __init__(self):

        self.font_1 = ("Courier", 15, "bold")
        self.font_2 = ("Courier", 12, "bold")
        self.font_3 = ("Courier", 8, "bold")
        self.font_4 = ("Courier", 10, "bold")

        self.color_1 = "black"
        self.color_2 = "white"
        self.color_3 = "#FFC2A1"
        self.color_4 = "#FF9A65"
        self.color_5 = "#D1D1E0"
        self.color_6 = "#F0F0F0"
        self.color_7 = "#D7D7D7"
        self.color_8 = "#4CFF70"
        self.color_9 = "#4C95FF"
        self.color_10 = "#003B90"
        self.color_11 = "#E8FF00"
        self.color_12 = "#FFEA84"
        self.color_13 = "#FF0000"
        self.color_14 = "#FF8F8F"
        self.color_15 = "#002EFF"
        self.color_16 = "#758EFF"
        self.color_17 = "#7E7E7E"
        self.color_18 = "#000000"
        self.color_19 = "#9CA4C8"
```



**14**

## **Image et Canvas**



```
self.canvas_logo = tk.Canvas(self, width=260, height=85, bg=self.obj_font.color_4, highlightthickness=0)
self.canvas_logo.place(x=15, y=15)
self.image_logo = ImageTk.PhotoImage(Image.open("logo.jpg"))
self.canvas_logo.create_image(0, 0, anchor=tk.NW, image=self.image_logo)
```

FRAMES  $\neq$  CANVAS





**15**

**Update automatique des images  
et labels**



**FONROCHE**  
éclairage

Courant LED : 10.51 mA

Tension BAT : 4.41 V

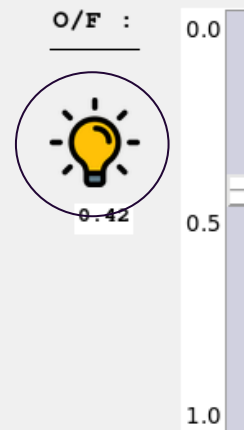
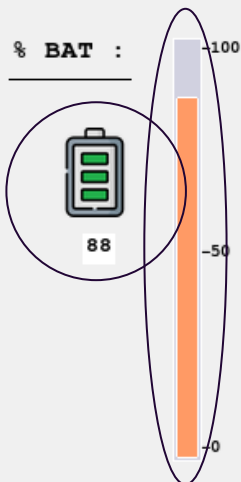
Etat BAT : ON

Temps Actuel 16:37:15

Dernière Ouverture : 16:37:02

## Informations Luminaire

X



Évolution Batterie

```
self.canvas_sli = tk.Canvas(self, width=60, height=60, bg=self.obj_font.color_6, highlightthickness=0)
self.canvas_sli.place(x=640, y=155)
```

```
def Image_Bulb(self):
    if self.blue_led.value == 0.00:

        self.open_image_sli = (Image.open("bulb_off.png"))
        self.resize_sli = self.open_image_sli.resize((55, 55), Image.ANTIALIAS)
        self.new_image_sli = ImageTk.PhotoImage(self.resize_sli)
        self.canvas_sli.create_image(0, 0, anchor=tk.NW, image=self.new_image_sli)

    elif self.blue_led.value ≥ 0.01:

        self.open_image_sli = (Image.open("bulb_on.png"))
        self.resize_sli = self.open_image_sli.resize((55, 55), Image.ANTIALIAS)
        self.new_image_sli = ImageTk.PhotoImage(self.resize_sli)
        self.canvas_sli.create_image(0, 0, anchor=tk.NW, image=self.new_image_sli)

    else:

        self.open_image_sli = (Image.open("bulb_default.png"))
        self.resize_sli = self.open_image_sli.resize((55, 55), Image.ANTIALIAS)
        self.new_image_sli = ImageTk.PhotoImage(self.resize_sli)
        self.canvas_sli.create_image(0, 0, anchor=tk.NW, image=self.new_image_sli)

    self.value_sli.config(text=self.blue_led.value)
    self.value_sli.after(500, self.Image_Bulb)
```

La fonction after () pour  
update mes labels et  
canvas



**16**

**Les fichiers textes pour écrire et  
stocker des données**



DATA\_GRAPH.txt



log.txt



[\\*link](#)

DATA\_GRAPH.txt :

```
def Write_Graph_Data(self):  
    n = 0  
    while 1:  
        n += 1  
        print("In Graph X = ", n, " AND Y = ", self.charge_integer)  
        self.file_graph.write(str(n) + "," + str(self.charge_integer) + "\n")  
        self.file_graph.flush()  
        time.sleep(1)
```



DATA\_GRAPH.txt - Bloc-notes

Fichier

Modifier

Affichage

1,62

2,62

3,63

4,62

5,62

6,62



**17**

**Envoi de mail en cas de panne  
batterie**


## log.txt :

```
def SEND_MAIL_PROBLEM(self):  
    self.current_time_full = strftime("%d/%m/%Y, %H:%M:%S")  
  
    while 1:  
        print("Scan Status Battery")  
        time.sleep(20)  
        if 0 ≤ self.charge_integer ≤ 10:  
            self.win_dead_bat()  
  
            self.file_log = open("log.txt", "w")  
            self.file_log.write(self.current_time_full)  
            self.file_log.write(" Batterie = ")  
            self.file_log.write(str(self.charge_integer))  
            self.file_log.write(" %")  
            self.file_log.close()  
  
            runFILE.mail_sent()  
            print("MAIL SENT TO SUPPORT")
```





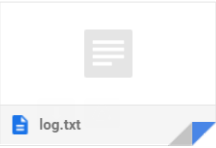
!!! ⚠️ ALERTE ROUGE ⚠️ !!! ➡️ Boîte de réception x

 **raspberry\_de\_maxence** <raspberry.localhost@gmail.com>  
À moi, Destinataires ▼

La **BATTERIE** est bientôt à plat !

🔋 🔋 🔋 🔋 🔋 🔋 🔋 🔋 🔋 🔋

[Notre Site](#)



← Répondre   ← Répondre à tous   → Transférer

log.txt

Rechercher dans les messages

22/05/2022, 00:58:47 Batterie = 5 %


Nouveau message

Boîte de réception

Messagerie suivie


En attente

Messages envoyés

 **Raspberry.Localhost** [Home](#) [Nos Produits](#) [Contact](#)

# Raspberry Pi

Your tiny, dual-display, desktop computer



## Nos Produits

Notre boîtier a été conçu par les meilleurs ingénieurs, totalement créé avec une imprimante 3D, cette petite boîte vous permettra de faire bien des choses...



**18**

**Envoi de donnée sur TTN**

```
def Update_Data(self):  
    while 1:  
        DATA.BigData()  
        print("data send to bdd")
```



Partie de Faris

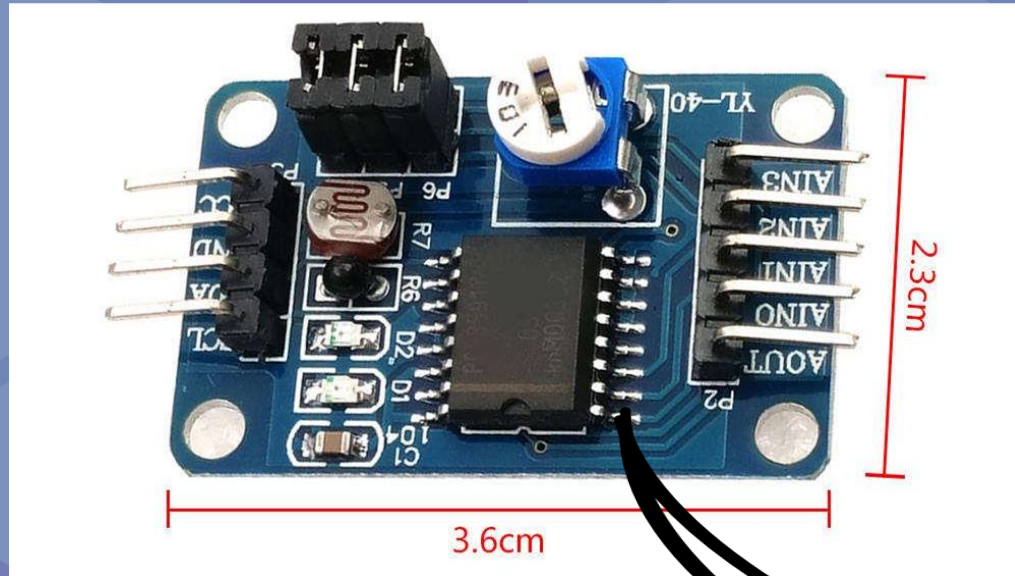
```
import serial  
import time  
  
def BigData():  
    print("DATA file is OK, data sent")  
    ser = serial.Serial(port='/dev/ttyUSB0', baudrate=9600, bytesize=8, parity='N', stopbits=1, timeout=1)  
    cmd="AT+MODE=LWOTAA"  
    ser.write(cmd.encode())  
    print(ser.read(40))  
    cmd="AT+DR=EU868"  
    ser.write(cmd.encode())  
    print(ser.read(40))  
    cmd="AT+KEY=APPKEY,\"BE3D4D9951757E75A75F9D00D2B8BC5E\""  
    ser.write(cmd.encode())  
    print(ser.read(40))  
    cmd="AT+JOIN"  
    ser.write(cmd.encode())  
    print(ser.read(40))  
    time.sleep(8)  
    cmd="AT+MSGHEX=\"4641524953\""  
    ser.write(cmd.encode())  
    msg=ser.read(64)  
    print(msg)
```



**19**

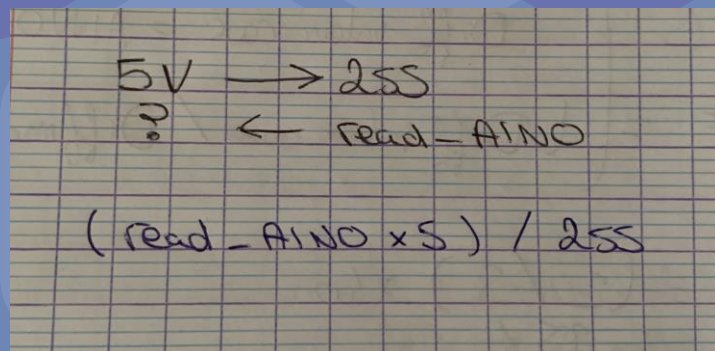
**Calcul tension, courant et % de la charge de la batterie**

## Module YL-40

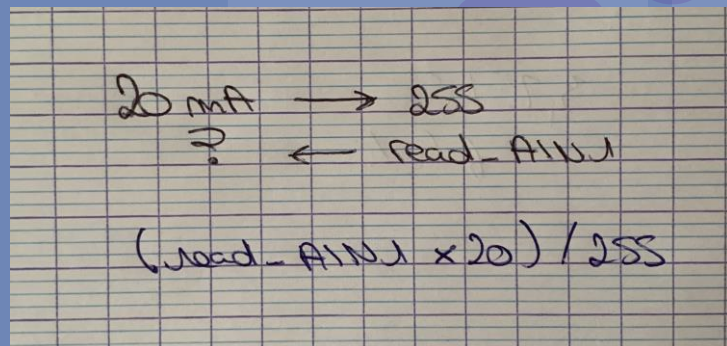


## Calcul tension et courant :

```
def read_AIN0(self):  
    self.bus.write_byte(self.address, 0x40)  
    self.read_0 = self.bus.read_byte(self.address)  
    print("AIN0 = ", self.read_0, "\n")  
  
    self.value_tension = (self.read_0 * 5) / 255  
    self.tension_lab.config(text=round(self.value_tension, 2))  
    print("Data on AIN0 = ", self.value_tension, "\n")  
    self.tension_lab.after(300, self.read_AIN0)  
  
def read_AIN1(self):  
    self.bus.write_byte(self.address, 0x41)  
    self.read_1 = self.bus.read_byte(self.address)  
    print("AIN1 = ", self.read_1, "\n")  
  
    self.value_current = (self.read_1 * 20) / 255  
    self.current_lab.config(text=round(self.value_current, 2))  
    print("Data on AIN1 = ", self.value_current, "\n")  
    self.current_lab.after(300, self.read_AIN1)
```



Handwritten diagram on graph paper showing the calculation for AIN0. It starts with "5V" pointing to "255", then a question mark points to "Read-AIN0". Below this, the formula  $(\text{Read-AIN0} \times 5) / 255$  is written.



Handwritten diagram on graph paper showing the calculation for AIN1. It starts with "20mA" pointing to "255", then a question mark points to "Read-AIN1". Below this, the formula  $(\text{Read-AIN1} \times 20) / 255$  is written.

## Calcul charge batterie :

```
print("Tension pour calcul BAT = ", self.value_tension)
self.difference_max = 5 - 0
self.difference_range = self.value_tension - 0
self.charge = (self.difference_range / self.difference_max)*100
self.charge_integer = int(self.charge)
print("Value BAT in % = ", self.charge_integer)
```

charge batterie = (tension de la batterie / différence de voltage) \* 100



20

Actualiser la barre de progression  
&  
Allumage progressif avec un  
slider



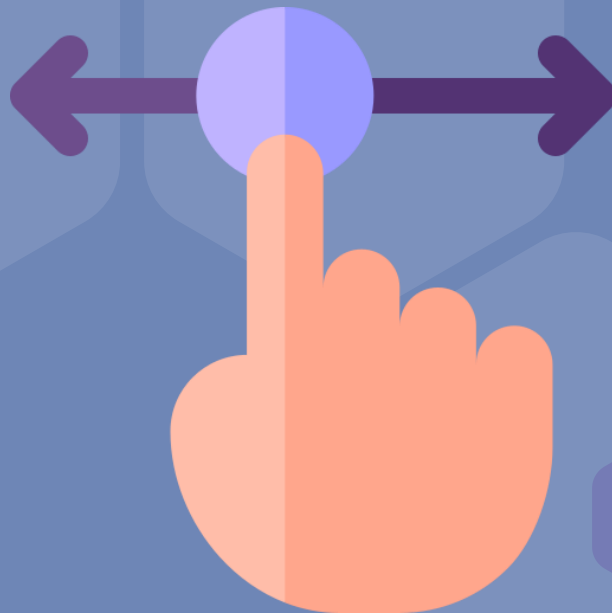
## Barre de progression :

```
self.bar.config(value=self.charge_integer)
self.bar.after(500, self.Image_Bar)
```

## Allumage avec slider :

```
self.blue_led = PWMLED(18)
```

```
def ChangeValueLED(self, value):
    self.led_value = float(value)
    print(value, self.led_value, self.blue_led)
    self.blue_led.value = self.led_value
```





**21**

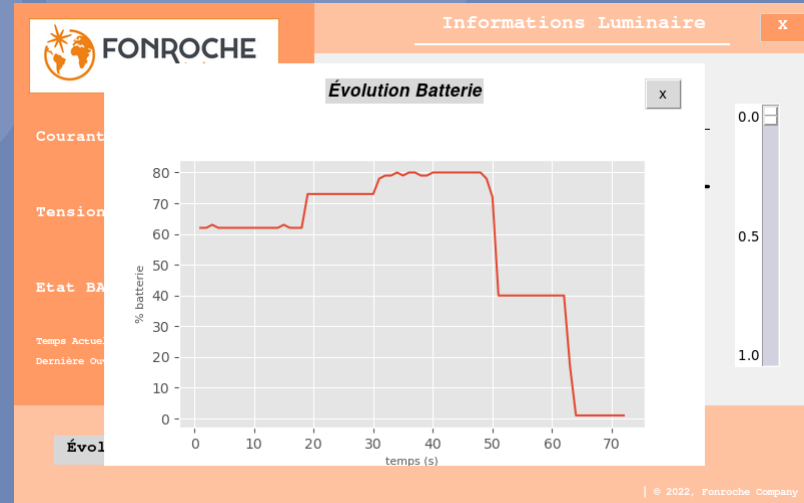
**Afficher l'évolution de la batterie**

Temps Actuel : 16:37:15

Dernière Ouverture : 16:37:02

## Évolution Batterie

```
def Button_Clicked(self):  
    data_graph.animate()  
    self.obj_graph = PageThree()
```

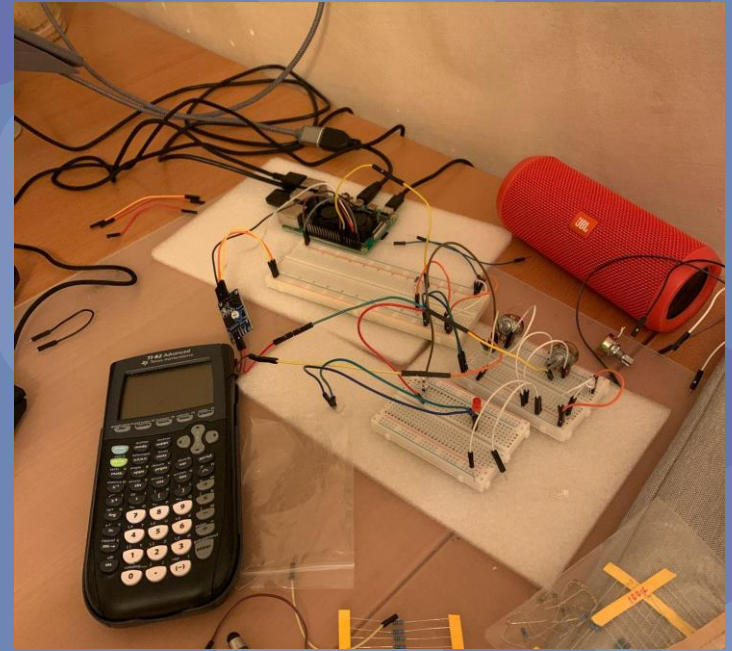
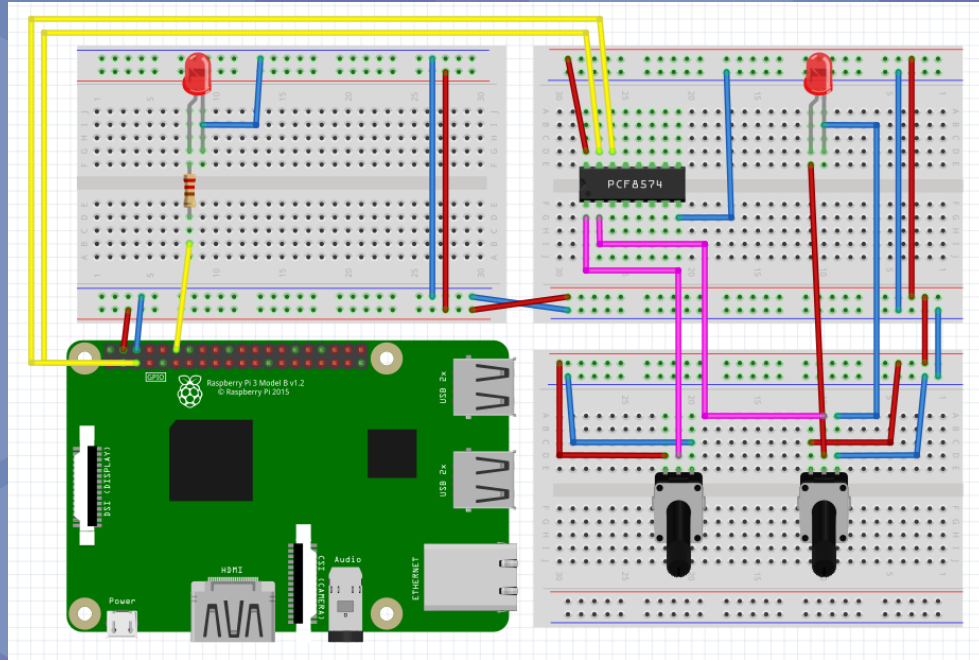


```
def animate():  
    print("data_graph file is open")  
    pullData = open("DATA_GRAPH.txt", "r").read()  
    dataList = pullData.split("\n")  
    xList = []  
    yList = []  
    for eachLine in dataList:  
        if len(eachLine) > 1:  
            x, y = eachLine.split(",")  
            xList.append(int(x))  
            yList.append(int(y))  
  
    a.clear()  
    a.plot(xList, yList)  
    a.set_xlabel("temps (s)", size=8)  
    a.set_ylabel("% batterie", size=8)
```



22

## Montage et simulation



- Simulation de la batterie avec un potentiomètre
- Simulation du courant dans la LED avec un potentiomètre
- Simulation de l'éclairage avec une LED



**23**

**Mise en place du système et  
démonstration**

# Des Questions ?

