INSTITUT d'OPTIQUE GRADUATE SCHOOL ParisTech | LEnsE Laboratoire d'Enseignement Expérimental

Projet Harpe Laser

Rapport technique

Jiaqi Chen

Alyssa Mayeux

Cyprien Lanneau

Yunze Xie

Projet d'Ingénierie Électronique

pour le Traitement de l'Information

Institut d'Optique / 1A

Date de rédaction : 27/03/2023-17/05/2023

Sommaire

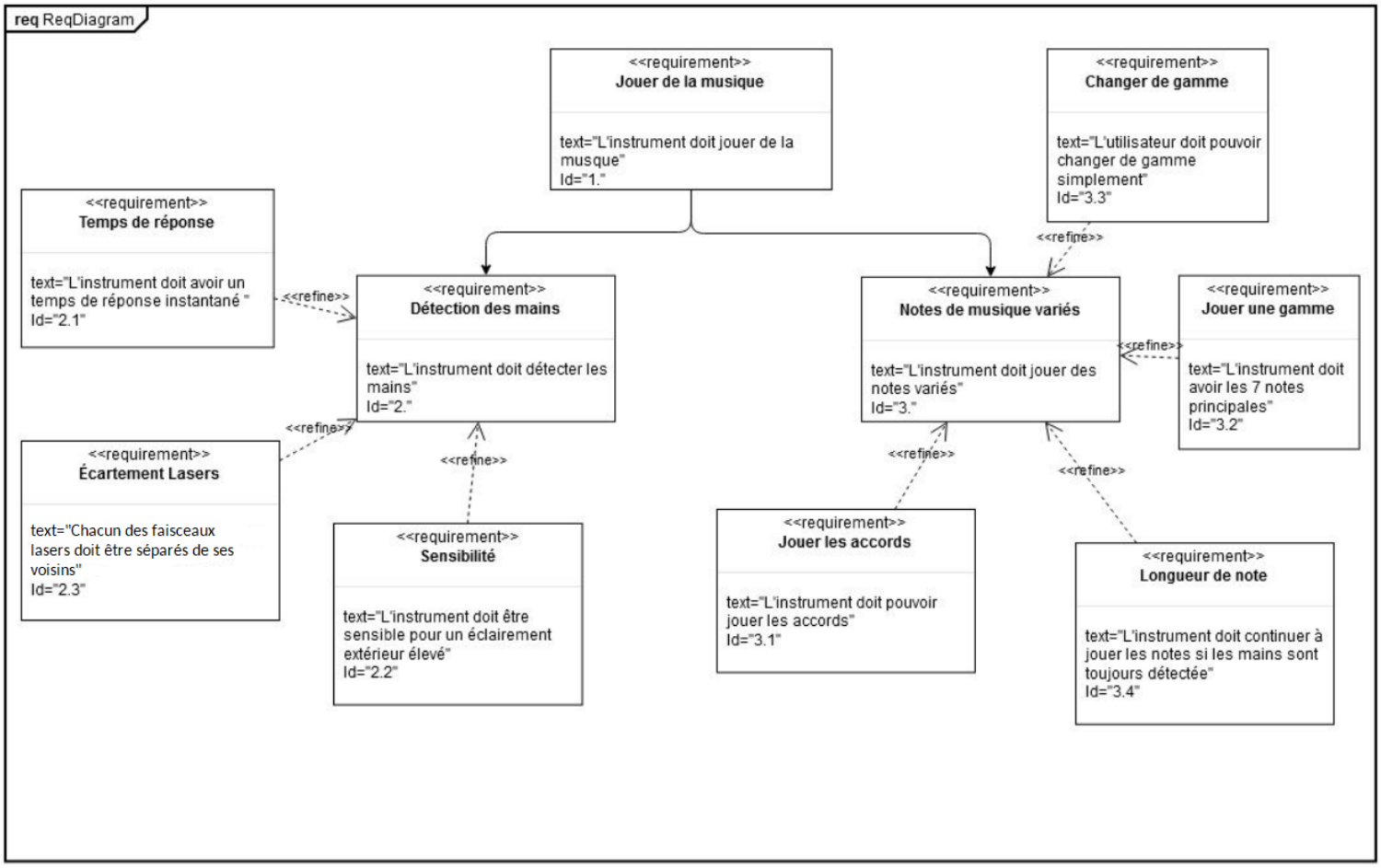
I - Introduction

1.1 Présentation

Notre projet consiste à réaliser une harpe laser, un instrument de musique où l'émission du son est contrôlée par les mains du musicien qui coupent les faisceaux lasers. Chaque faisceau est associé à une note de musique, ce qui permet la création de mélodies.

1.2 Cahier des charges

Notre instrument doit vérifier les fonctions suivantes:

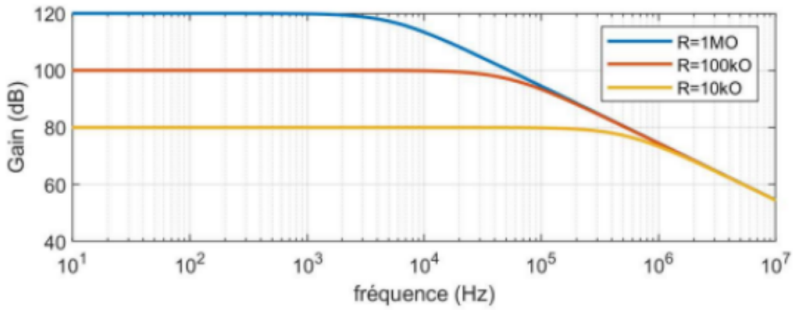


II - Conception et Réalisation

Notre système est constitué d'une première partie électronique & optique permettant de détecter les doigts du musicien coupant les faisceaux lasers de l'instrument et d'une deuxième partie numérique permettant de produire une note associée à chaque faisceau. Une troisième partie permet de finaliser la harpe laser à l'aide d'une structure découpée à la découpeuse laser.

2.1 Electronique & optique

Cette partie constitue la dimension physique de l'instrument. Pour la réaliser, on utilise plusieurs systèmes de photodétection montés en parallèle (schéma ci-dessous). On vérifie bien que le gain de chaque détecteur est assez élevé pour pouvoir détecter ou non chaque faisceau laser. On s'arrange donc pour avoir 3,3V en sortie lorsqu'un faisceau est détecté pour pouvoir transmettre l'information numérique à la carte Nucleo et on fixe le gain avec la résistance (ici $1M\Omega$). En effet, plus la résistance choisie est élevée, plus le gain du photodétecteur est élevé.



Résultat analytique du gain pour un système de photodétection

On relie chaque sortie des photodétecteurs à une entrée numérique de la carte Nucleo qui traduira directement la tension 0V en '0' et la tension 3,3V en '1'. On réalise le système schématisé ci-dessous en utilisant la source de tension de la carte Nucleo pour réaliser la photodétection.

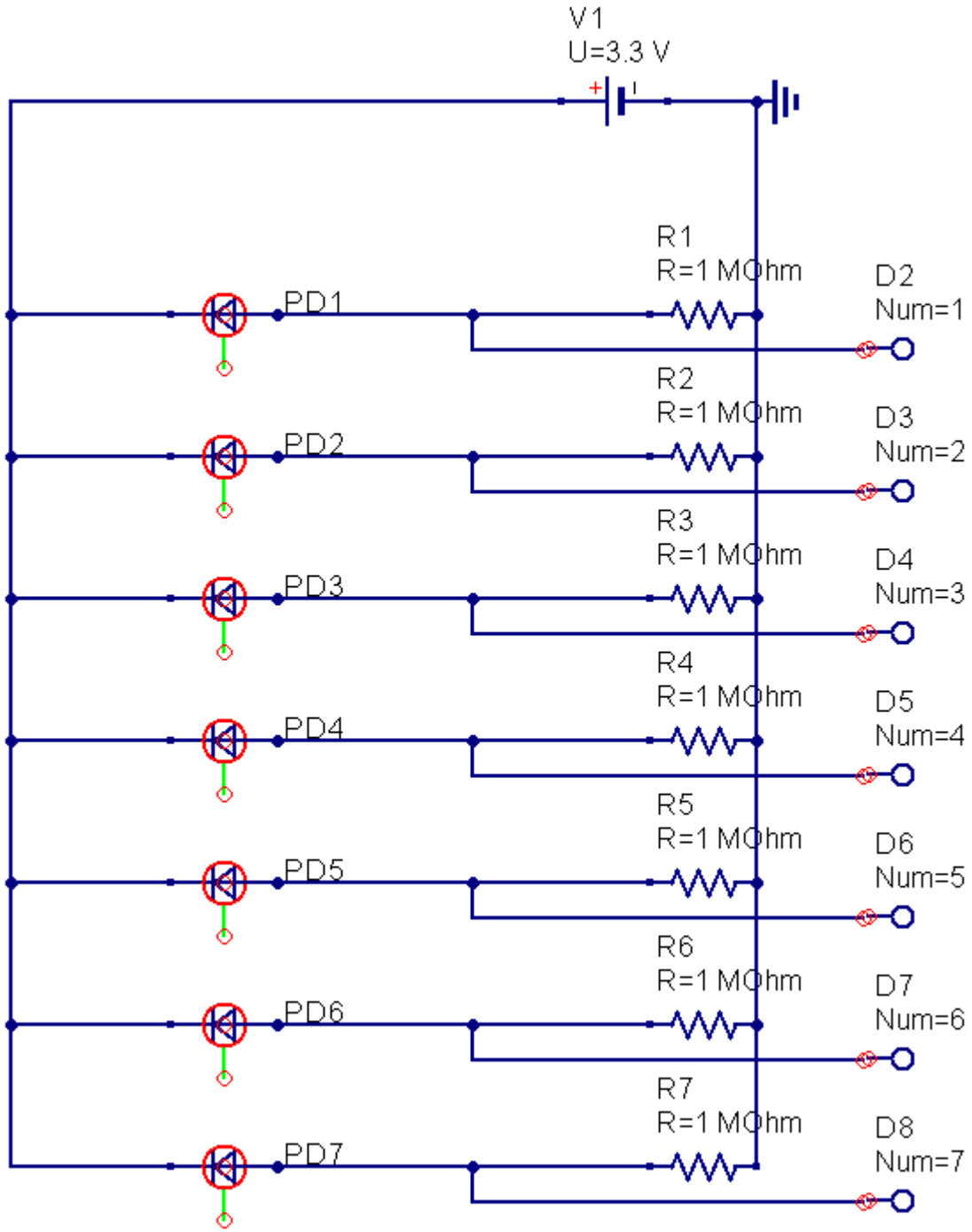


Schéma de la partie électronique

2.2 Numérique

La partie numérique est constituée d'une partie *Mbed* permettant de transmettre les informations numériques de la carte Nucleo à l'ordinateur et d'une partie *python* permettant de convertir chaque information numérique en une note de musique.

Mbed

Pour la partie *Mbed*, commentée en annexe, on déclare les entrées associées chacune à un système de photodétection puis on crée un tableau de longueur 8. Les 7 premières cases du tableau sont associées chacune à une photodiode et servent à transmettre à *python* le numéro des faisceaux coupés par les doigts du musicien. En effet, si le n-ème faisceau est coupé, la n-ème case du tableau prendra la valeur n, sinon elle prend la valeur 0. La 8-ème case du tableau permet de transmettre la valeur d'un bouton. Celui-ci a une entrée analogique pouvant prendre 3 valeurs. Son rôle est de permettre de changer de gamme musicale avec le code *python*. La 8-ème case prend alors la valeur 0, 8 ou 9 selon la valeur du potentiomètre.

Pour réaliser ce bouton, on fait un montage simple avec un potentiomètre associé à la source de tension de la nucléo.

Le tableau précédemment décrit est ensuite affiché, ce qui permet de le transmettre à l'ordinateur et donc à *python*. Il est actualisé toutes les 30 ms de façon à avoir une détection instantanée des faisceaux lasers à l'échelle du musicien.

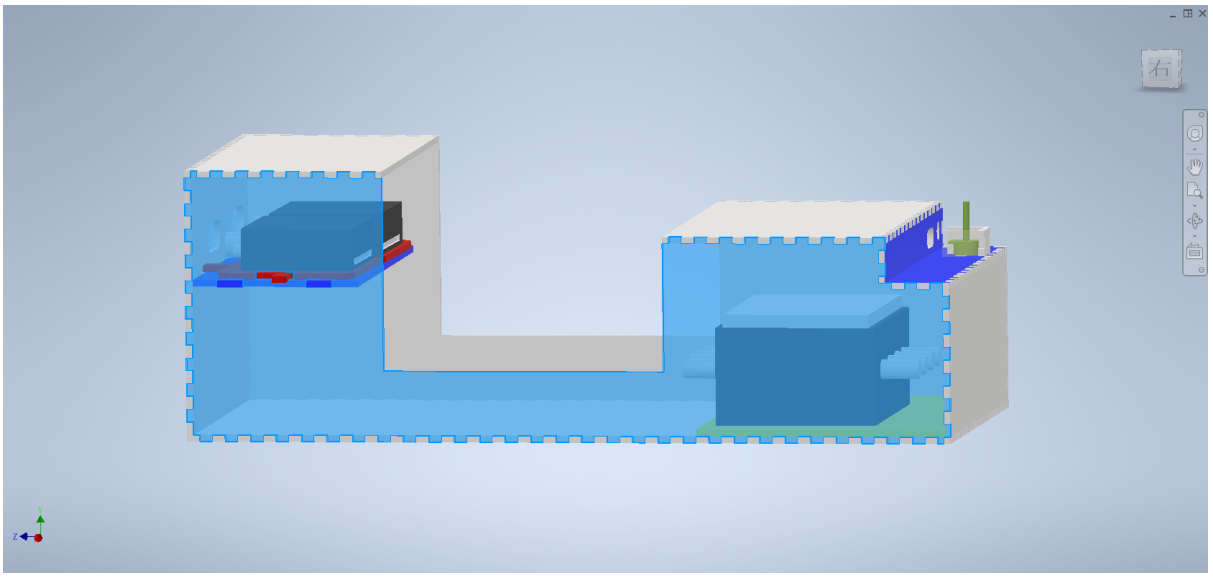
Python

La partie *python* utilise le module *pyserial* pour récupérer les informations affichées par la carte Nucleo. Il est important d'associer chaque photodiode à des chiffres différents puisque python ne voit qu'une suite de chiffres et non pas un tableau. On recrée sur python un tableau de longueur 8, une première boucle sert à synchroniser les informations de *mbed* avec le tableau *python* puis une deuxième boucle permet de lire les informations de *mbed* et enfin une dernière boucle permet de remplir le tableau *python*.

On utilise le module *pygame.mixer* pour jouer les notes de musiques (enregistrées préalablement dans l'ordinateur) en fonction des valeurs directement lu avec *pyserial*. Ce module permet de jouer plusieurs notes à la fois, ce qui est l'un de nos critères. Une fois rempli, le tableau nous permet de connaître la gamme de la note de musique et de ne pas avoir de répétition d'une note alors que celle-ci n'a pas encore fini d'être jouée une première fois.

Rq: Il faut fermer le port de lecture de la Nucleo après chaque utilisation avec la commande `ser.close()`.

2.3 Réalisation 3D



Réalisation 3D du support de la harpe laser

On réalise un support 3D pour la harpe laser avec le logiciel *Autdesk Inventor*. L'idée était de camoufler les éléments tels que les boîtiers laser et le breadboard afin que ce soit plus esthétique mais aussi de pouvoir ensuite utiliser de la poudre de craie sans salir l'un ou l'autre. La poudre de craie permet de visualiser les faisceaux lasers et donc de vraiment jouer de la harpe laser. La partie supérieure au-dessus du breadboard est également là pour ne pas exposer les photodiodes à la lumière afin de pouvoir y jouer dans une pièce éclairée voire à l'extérieur.

III - Conclusion

En conclusion, la harpe laser fonctionne bien : toutes les photodiodes renvoient les bonnes valeurs, permettant ainsi de jouer les notes correspondantes. Par ailleurs, nous pouvons effectivement jouer de deux à sept notes en même temps, comme nous l'espérions. De plus, nous disposons de trois gammes couvrant les notes aigües et les graves qui peuvent être modifiées sans avoir à relancer le programme : il suffit de tourner le bouton pendant la mélodie pour changer de hauteur.

La harpe fonctionne uniquement à l'intérieur quand il n'y a pas trop de luminosité. Nous n'avons pas pu tester notre support mais nous espérions qu'il permette de jouer également de la harpe dans un cadre moins sombre.

Finalement, bien qu'il y ait encore des axes d'amélioration du projet, le produit final est fonctionnel et répond à notre cahier des charges.

Annexes

Python

Installation

`Spyder`

On installe les modules appropriés en ouvrant Anaconda Powershell Prompt (Anaconda 3) dans le répertoire de anaconda:

```
pip install pygame
```

```
install pyserial
```

On installe aussi une bibliothèque de son constituant nos différentes notes de musique dans le même fichier que le fichier python.

Code Python

```
import serial #importer les modules serial et pygame
import pygame

pygame.mixer.init()#initialiser le module mixer

ser = serial.Serial('COM6',112500,timeout = 1)
print(ser) #lire le nom de la liaison série de la nucléo
i = 0


tab = [0,0,0,0,0,0,0,0] #recréation d'un tableau de longueur 8
pour garder en mémoire les états des photodiodes

#indenter la première valeur sortie avec la valeur de la gamme, on
renvoie ensuite les différentes notes de musique
while(True):
    test = True
    while(test):
        res = ser.read()#sort b'a' avec a la valeur transmise
        res = str(res)
        res = res[2] #récupère le chiffre transmis par la nucléo
        if res == '7':
            test = False

#Lire la liaison série de la nucléo
    while(True):
        print(res)
        res = ser.read()
        res = str(res)
        res = res[2]


#émettre la note correspondante en fonction des valeurs sorties
par pyserial avec le module pygame et la bibliothèque de son, la
comparaison entre res et tab permet d'éviter la répétition de la
même note
        if (tab[0] == '8') and (res == '1') and (res!= tab[1]) :
     pygame.mixer.Channel(0).play(pygame.mixer.Sound("son/do.mp3"))

        elif (tab[0] == '8') and (res == '2') and (res!= tab[2]):
     pygame.mixer.Channel(1).play(pygame.mixer.Sound("son/re.mp3"))

        elif (tab[0] == '8') and (res == '3') and (res!= tab[3]):
     pygame.mixer.Channel(2).play(pygame.mixer.Sound("son/mi.mp3"))

        elif (tab[0] == '8') and (res == '4') and (res!= tab[4]):
```

```
    pygame.mixer.Channel(3).play(pygame.mixer.Sound("son/fa.mp3"))

        elif (tab[0] == '8') and (res == '5') and (res!= tab[5]):
    pygame.mixer.Channel(4).play(pygame.mixer.Sound("son/so.mp3"))

        elif (tab[0] == '8') and (res == '6') and (res!= tab[6]):
    pygame.mixer.Channel(5).play(pygame.mixer.Sound("son/la.mp3"))


        elif (tab[0] == '8') and (res == '7') and (res!= tab[7]):
    pygame.mixer.Channel(6).play(pygame.mixer.Sound("son/si.mp3"))

#changement de gamme : basse
        elif (tab[0] == '0') and (res == '1') and (res!= tab[1]) :
 pygame.mixer.Channel(0).play(pygame.mixer.Sound("son/do_bas.mp3"))

        elif (tab[0] == '0') and (res == '2') and (res!= tab[2]):
pygame.mixer.Channel(1).play(pygame.mixer.Sound("son/re_bas.mp3"))

        elif (tab[0] == '0') and (res == '3') and (res!= tab[3]):
 pygame.mixer.Channel(2).play(pygame.mixer.Sound("son/mi_bas.mp3"))

        elif (tab[0] == '0') and (res == '4') and (res!= tab[4]):
pygame.mixer.Channel(3).play(pygame.mixer.Sound("son/fa_bas.mp3"))

        elif (tab[0] == '0') and (res == '5') and (res!= tab[5]):
pygame.mixer.Channel(4).play(pygame.mixer.Sound("son/so_bas.mp3"))

        elif (tab[0] == '0') and (res == '6') and (res!= tab[6]):
pygame.mixer.Channel(5).play(pygame.mixer.Sound("son/la_bas.mp3"))

        elif (tab[0] == '0') and (res == '7') and (res!= tab[7]):
pygame.mixer.Channel(6).play(pygame.mixer.Sound("son/si_bas.mp3"))

#changement de gamme : haute
        elif (tab[0] == '9') and (res == '1') and (res!= tab[1]) :
pygame.mixer.Channel(0).play(pygame.mixer.Sound("son/do_haut.mp3")
)
        elif (tab[0] == '9') and (res == '2') and (res!= tab[2]):
pygame.mixer.Channel(1).play(pygame.mixer.Sound("son/re_haut.mp3")
)
        elif (tab[0] == '9') and (res == '3') and (res!= tab[3]):
pygame.mixer.Channel(2).play(pygame.mixer.Sound("son/mi_haut.mp3")
)
        elif (tab[0] == '9') and (res == '4') and (res!= tab[4]):
pygame.mixer.Channel(3).play(pygame.mixer.Sound("son/fa_haut.mp3")
)
```

```
        elif (tab[0] == '9') and (res == '5') and (res!= tab[5]):
pygame.mixer.Channel(4).play(pygame.mixer.Sound("son/so_haut.mp3")
)
        elif (tab[0] == '9') and (res == '6') and (res!= tab[6]):
pygame.mixer.Channel(5).play(pygame.mixer.Sound("son/la_haut.mp3")
)
        elif (tab[0] == '9') and (res == '7') and (res!= tab[7]):
pygame.mixer.Channel(6).play(pygame.mixer.Sound("son/si_haut.mp3")
)


 #créer un tableau de longueur 8 avec les informations données par
la carte nucléo
        tab[i] = res
        i = i + 1
        if i == 8:
            i = 0

#ser.close() #commande à entrer après l'utilisation pour reset la
liaison ordinateur-nucléo
```

Mbed

```
/* mbed Microcontroller Library
 * Copyright (c) 2019 ARM Limited
 * SPDX-License-Identifier: Apache-2.0
 */
#include "mbed.h"
// inputs and outputs configuration

//Déclaration des entrées
DigitalIn photodiode1 (D2);
DigitalIn photodiode2 (D3);
DigitalIn photodiode3 (D4);
DigitalIn photodiode4 (D5);
DigitalIn photodiode5 (D6);
DigitalIn photodiode6 (D7);
DigitalIn photodiode7 (D8);
AnalogIn Bouton(A0);

int main()
{
//Déclaration des variables
    float bouton;
    int tab[8];
    for (int i = 0; i < 10; i++){  // création d'un tableau
            tab[i] = 0;
    }
    while (1)
    {
//Permet d'associer le n-ème faisceau coupé à la n-ème case du
tableau qui prendra la valeur n, sinon elle prend la valeur 0
        tab[0] = 1-(photodiode1*1);
        tab[1] = 2-(photodiode2*2);
        tab[2] = 3-(photodiode3*3);
        tab[3] = 4-(photodiode4*4);
        tab[4] = 5-(photodiode5*5);
        tab[5] = 6-(photodiode6*6);
        tab[6] = 7-(photodiode7*7);
```

```
//permet de changer de gamme à l'aide du bouton : 3 gammes
disponibles
        bouton = Bouton.read();
        bouton = bouton *3;

        if ((bouton >= 0) and (bouton < 1)){
            tab[7] = 0;
        }
        if ((bouton >= 1) and (bouton < 2)){
            tab[7] = 8;
        }
        if ((bouton >= 2) and (bouton < 3)){
            tab[7] = 9;
        }

//renvoie les valeurs contenues dans le tableau pour qu'elles
soient transmises à python
        printf("%d",tab[0]);
        printf("%d",tab[1]);
        printf("%d",tab[2]);
        printf("%d",tab[3]);
        printf("%d",tab[4]);
        printf("%d",tab[5]);
        printf("%d",tab[6]);
        printf("%d",tab[7]);
        wait_us(30000); //30ms d'attente entre chaque boucle


    }
}
```