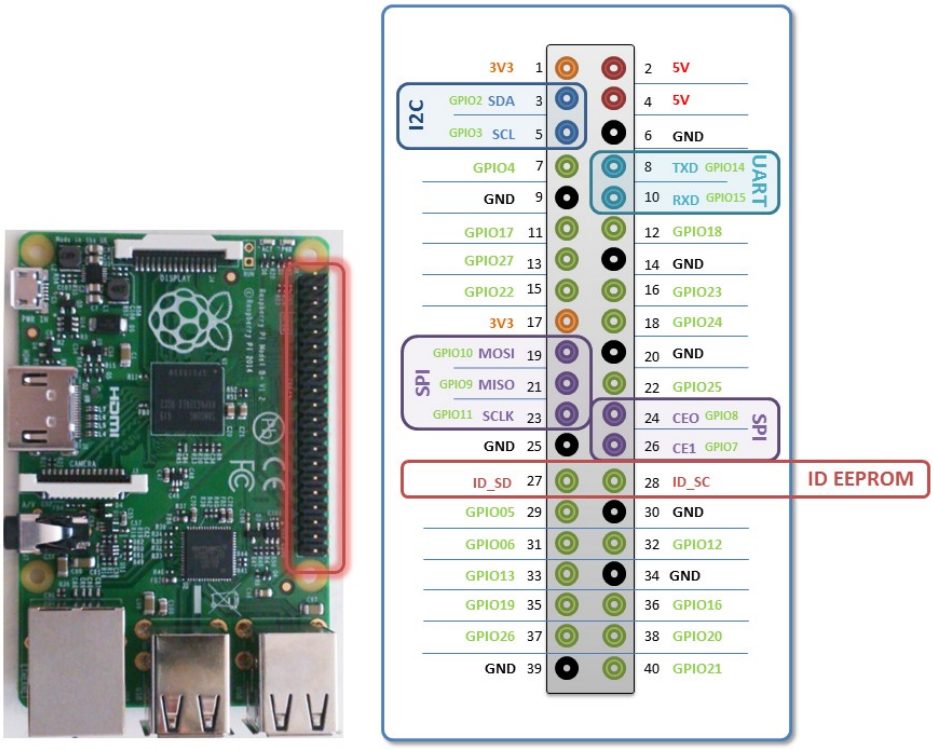
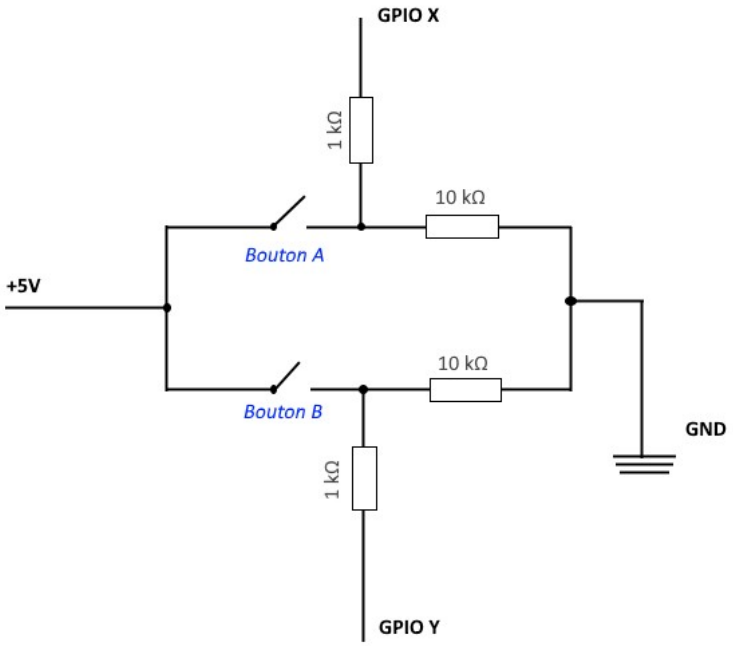
Partie I



[I-1] : 'Pin-mapping' des broches d'entrée/sortie GPIO

Partie II



[II-1] : Schéma électronique détaillé pour deux bouton connectés à deux GPIO différents

Partie III

```
Initialisation :
GPIO.setmode(GPIO.BCM)   #Cette ligne permet l'utilisation de la
                         numérotation BCM pour l'appel des
                         différentes broches GPIO

GPIO.setup(X,GPIO.IN)    #Cette instruction permet de définir le
                         GPIO X (numéro d'identification BCM),
                         comme une entrée, chargé de capter un
                         signal binaire.

Fonctions :

GPIO.input(X)            #Agit comme une condition, renvoie
                         'True' dès lors qu'un signal est
                         perçut,'False' sinon.
```

***[III-1]** : Éléments utilisé depuis le module GPIO*

```
time.sleep(secondes)  #la fonction sleep() du module time
                      prend en argument un temps en
                      secondes (entier ou décimal) et
                      permet d'arrêter l'exécution du
                      programme pendant ce laps de temps.
```

***[III-2]** : Fonction sleep() utilisée dans notre programme*

```
#Fonction permettant de créer un tableau de largeur x et
de hauteur y

def creer_t(x,y):
    tabl = []
    for i in range(y):
        tabl.append(["  " for p in range(x)])
    return(tabl)

#Fonction permettant d'afficher un tableau à l'écran
précédé d'un titre donné en paramètre

def afficher(tabl,titre='null'):
    os.system("clear")
    if titre!='null':
        print('==================== '+titre+'
====================\n')
    for i in tabl:
        print(" ".join(i))
```

***[III-3]** : Fonctions de création/affichage de tableau*

```
#Fonction de déplacement prenant en argument t, le
tableau concerné par le deplacement ; xi, et yi les
coordonnées initiales du caractère à déplacer c, et la
position à atteindre (x,y).

def deplacement(t,xi,yi,x,y,c):
    if t[yi][xi] == c + c:
        t[yi][xi]="  "
    if t[y][x] == "  ":
        t[y][x]= c + c
    return(t)
```

***[III-4]** : Fonction de déplacement d'un caractère*

```
#Création du tableau 'quitter' avant la boucle infinie

quitter=src.creer_t(2,2)
quitter[0][1]=' OUI'
quitter[1][1]=' NON'

#Requete pour fermer le programe, n'est pas satisfaite si le
bouton close (GPIO 24) n'est pas enclenché.

if GPIO.input(24): #Vaut False par défaut
        Quitter[0][0]='>>' #on insere un module deplaçable
        mx=modulex     #on definit la position de ce module
        my=moduley
        modulex=moduley=0
        while GPIO.input(23)!=True:#boucle → attente validation
            src.afficher(quitter,fermer)
            if GPIO.input(17) and moduley>0:
               #Si on pousse le joystick vers le haut...
quitter=src.deplacement(quitter,modulex,moduley,modulex,moduley
-1,">")
                moduley -=1
            elif GPIO.input(27) and moduley<1:
               #et vers le bas ...
quitter=src.deplacement(quitter,modulex,moduley,modulex,moduley
+1,">")
                moduley+=1
            time.sleep(0.1)    #on evite de rafraichir l'ecran
                               trop vite...
        if moduley==0:
            sys.exit(0) #on ferme le programme
        else:
            modulex=mx
            moduley=my
            quitter[1][0]='  '
```

***[III-5]** : Cette partie du code nous permet de fermer le programme*

Liens

[L-1] : Liens du cours sur la modularité que nous avons suivis sur le cours Python du site OpenClassrooms :

https://openclassrooms.com/courses/apprenez-a-programmer-en-python/pas-a-pas-vers-la-modularite-2-2

[L-2] : Lien vers le forum américain traitant de la fonction os.system(‘clear’) :

http://stackoverflow.com/questions/4810537/how-to-clear-the-screen-in-python

[L-3] : Liens du cours sur la portée des variables de référence du site OpenClassrooms :

https://openclassrooms.com/courses/apprenez-a-programmer-en-python/portee-des-variables-et-references