Chapitre N°6.) Simplifier l'accès aux entrées/sorties avec gpiozero

Ce que vous allez apprendre :

- ✓ Piloter les broches d'entrée-sortie du Raspberry Pi
- ✓ Découvrir la simplicité du module gpio zero p
- ✓ Le mot clé global

I. Présentation de la bibliothèque gpiozero

Des GPIO (General Purpose Input/Output) sont disponibles sur le connecteur de la Raspberry Pi. Ce qui permet de l'interfacer avec des circuits électroniques.
Vous avez vu que ces broches peuvent être configurées soit en entrée (binaire) soit en sortie.
Vous avez également codé à l'aide des commandes python setup, output et input : c'est une tâche ardue et pour faire clignoter par exemple une led, cela demande quatre lignes.

Mais une nouvelle bibliothèque, appelée GPIO Zero, simplifie tout ça de manière significative :

Avec gpiozero	Avec RPi.GPIO

`from gpiozero import LED`
`led=LED(21)`
`led.on()`
`led.off()`
	`import RPi.GPIO as GPIO Led=21`
	`GPIO.setup(Led,GPIO.OUT)`
	`GPIO.output(Led,GPIO.HIGH)`
	`GPIO.output(Led,GPIO.LOW)`

Dans cette partie vous allez utiliser la bibliothèque ***gpiozero***

La documentation complète se trouve à http://gpiozero.readthedocs.io/en/stable/

Les éléments pris en charge bien sûr des leds et BP mais pas uniquement : capteur de ligne, détecteur de mouvements, photorésistance, ultra-son, led, RGB,PWM, buzzer, moteur, servo-moteur, convertisseurs CAN, SPI, de nombreuses cartes d'extension…

II. Retour sur les leds avec gpiozero

1. Le code

Rien de plus simple :

```
from gpiozero import LED
led = LED(21)
led.on()
```

Sans essayer le programme, que fait-il ? Devinez comment on éteint une led ?

Avec un peu de connaissance en anglais vous savez que 'on' c'est allumé et 'off' éteint.
En plus d'avoir moins de lignes, le code est bien plus compréhensif.

2. Explications

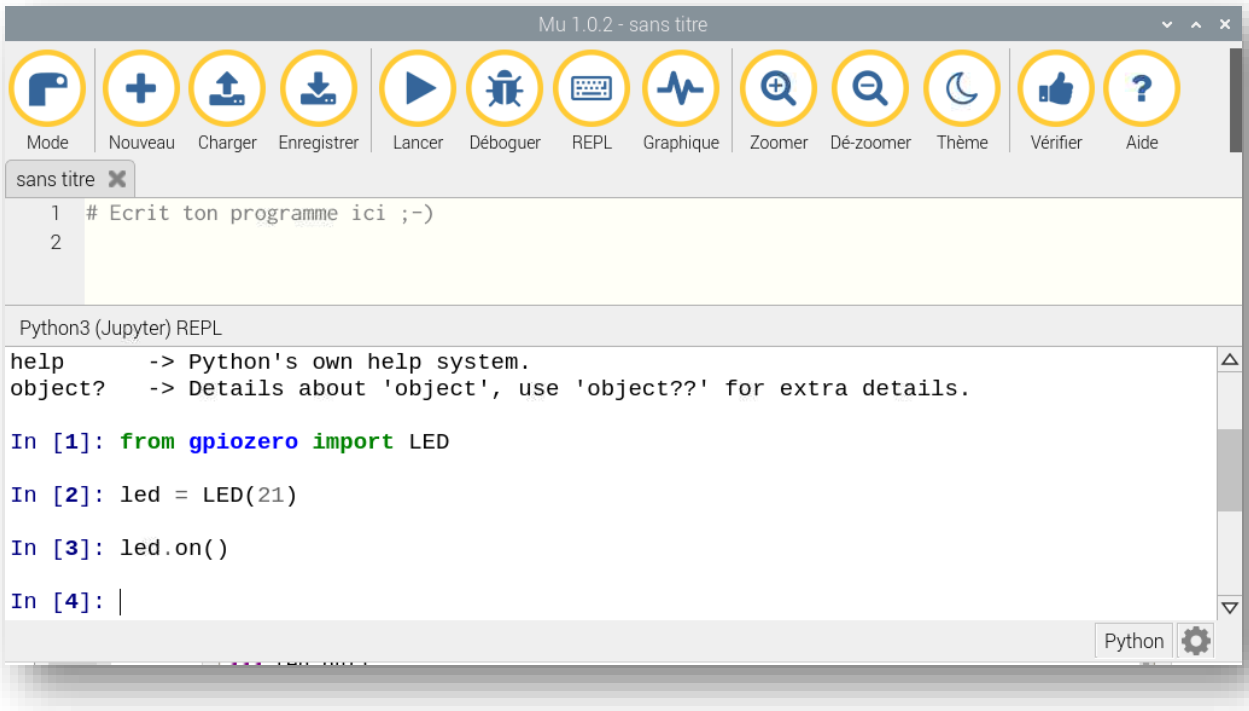
1. La première ligne du programme correspond à l'importation de la bibliothèque gpiozero dans Python.
2. La deuxième ligne crée un objet DEL, avec comme argument le numéro de broche (entre parenthèses)
3. La troisième ligne indique à la broche de s'activer (c'est de la Programmation Orientée Objet, on() est une méthode de la classe LED)

3. Mode intéractif

Malheureusement il n'existe plus idle dans la dernière version de rasbian.

Nous allons alors utiliser le mode intéractif de mu nommé REPL (ou Jupyter notebook) :

Mettez une led sur le port **21** et sa cathode à la masse, puis lancez mu (-> ->) et entrez les trois lignes du code un par un en observant le led :



Toujours dans le terminal du bas : *led.off()*

4. En mode programme

Ouvre un environnement de programmation soient Geany, mu ou Thonny et entrez le programme. Exécutez -le.

Si vous avez ajouté *led.off()*, la led n'a pas le temps de s'allumer qu'elle est déjà éteinte, sinon elle est toujours allumée.

Quelle est la commande python pour faire une pause dans le programme ? Modifiez le pour qu'au bout de 3 secondes, la led s'éteigne.

5. Led clignotante : le blink

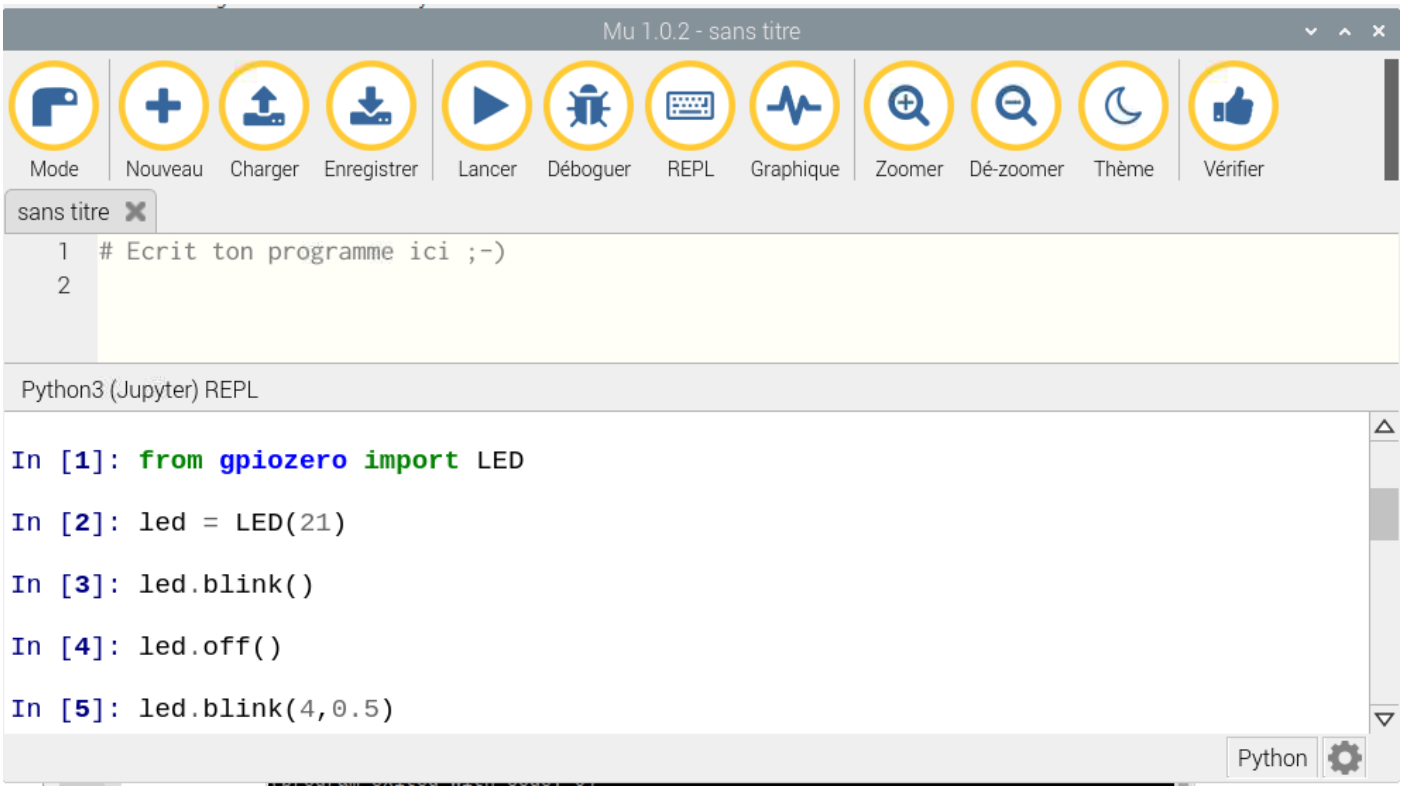
Souvenez-vous de l'un des tous-premiers code Arduino *blink* permettant de tester une carte ?
Et bien avec gpio zero c'est simplement :

```
led.blink()
```

Ouvrez *mu* et dans le notebook Jupyter (REPL) testez les commandes suivantes et donnez leurs

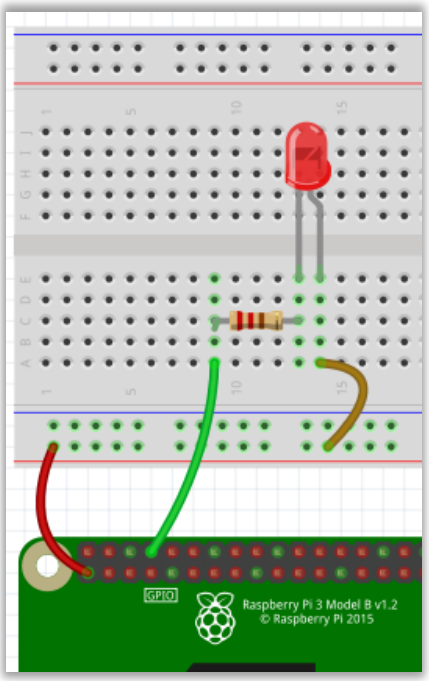
significations :

Code Python	Signification
`from gpiozero import LED`	
`led = LED(21)`	
`led.blink()`	
`led.off()`	
`led.blink(4,0.5)`	
`led.blink(0.5,4)`	
`led.blink(0.5,0.5)`	
`led.blink(n=3)`	
`led.toggle()`	
`led.toggle()`	
`print(led.is_lit)`	
`led.toggle()`	
`print(led.is_lit)`	
`led.blink(n=5,background=False)`	



6. Anode à la masse

Ajoutez une seconde led comme ci-dessous



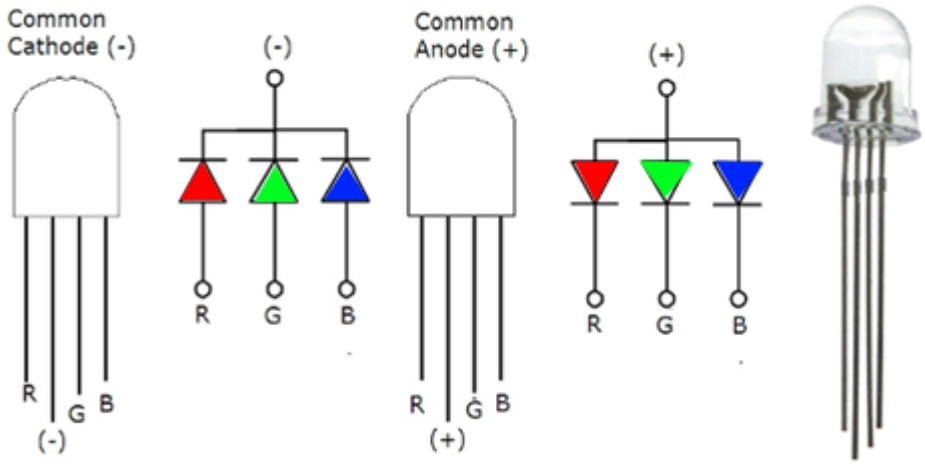
Essayez d'allumer et d'éteindre la led, quelles fonctions allez-vous utiliser ? Remarques ?

La définition de la LED est `gpiozero.LED(pin, active_high=True, initial_value=False)`

En modifiant un des paramètres, faites en sorte que `led.on()` allume bien la led

7. La Led RGB

Pour simplifier, une led RGB est un ensemble de trois leds dont une des deux pattes est soudée :



Dans les deux cas la patte la plus longue est la patte commune.

Ainsi pour avoir une couleur rouge il faut :

Cathode commune = Cathode à la Masse	Anode commune = Anode à 3.3V
Broche R à 3.3 V Broche G à la masse Broche B à la masse	Broche R à la masse Broche G à 3.3 V Broche B à 3.3V

La superposition de couleurs est possible mais le résultat est souvent décevant.

Sans effectuez le câblage quelle est le résultat de ces lignes.
Quelles couleurs sont allumées ?

```
from gpiozero import RGBLED

led = RGBLED(2, 3, 4)
led.color = (1, 1, 0)
```

Quelle est la différence avec le code précédent ?

```
from gpiozero import RGBLED
from colorzero import Color

led = RGBLED(2, 3, 4)
led.color = Color('yellow')
```

III. Maintenant au tour des BP (chouette !!!)

1. Détection appuis

Mettez sur un port au choix, un bouton-poussoir en pull-down .

Trois méthodes de gestion des boutons

Code 1

```
from gpiozero import Button

bp = Button(xx)
while True:
    if bp.is_pressed:
        print('bp on')
        break
    else:
        print('bp off')

print('Fin du programme')
```

Code 2

```
from gpiozero import Button
bp = Button(xx)

bp.wait_for_press()
print('bp on')
bp.wait_for_release()
print('bp off')
```

Code 3

```
# -*- coding: utf-8 -*-
from gpiozero import Button
bp = Button(xx)

def MessageOn():
    print('bp on')

def MessageOff():
    print('bp off')

#programme principal
bp.when_pressed = MessageOn
#ATTENTION fonction sans parenthèse
bp.when_released = MessageOff

print("Appuyez sur le bouton")
input("Ou sur Entrée pour terminer le programme")
```

Testez ces trois méthodes et complétez le tableau de comparaison suivant (réponde par oui ou non)

Critère	Code 1	Code 2	Code 3
Bloque le programme et attend	OUI - NON	OUI - NON	OUI - NON

L'attente de l'appui se fait en arrière-plan du programme	OUI - NON	OUI - NON	OUI - NON
Ressemble à la programmation sans gpiozero (Arduino)	OUI - NON	OUI - NON	OUI - NON

2. Avec une led

Rajoutez une led : elle doit s'allumer lors d'un appui et s'éteint sur un relâchement. Utilisez les trois méthodes vues ci-dessus.

IV. Retour sur le jeu précédent en plus facile

1. Eléments de python

a. Global devant une variable

Sans le tester, prédisez l'affichage du programme suivant

```
compteur1 = 10
compteur2 = 10

def Unefonction():
    global compteur1
    compteur1 = 100
    compteur2 = 10

print("Avant l'appel de la fonction")
print("La valeur de compteur1 est", compteur1)
print("La valeur de compteur2 est", compteur2,"\n")

#appel de la fonction
Unefonction()
print("Après l'appel de la fonction")
print("La valeur de compteur1 est", compteur1)
print("La valeur de compteur2 est", compteur2)
```

Testez le et indiquez le rôle du mot de clé *global*

2. Retour sur le cahier des charges de l'exercice "qui est le plus rapide"

Ce jeu se joue à deux : une led s'allume et le premier qui a appuyé sur son bouton a gagné.

- Phase I : au début la led clignote cinq fois puis reste allumé pendant 2 secondes. Ensuite, le programme attend aléatoirement entre 1 et 3 secondes
- Phase II : la led s'allume et le jeu commence. Le programme attend l'appui sur un des deux boutons.

- Phase III : la led clignotera une fois si c'est le joueur de gauche qui a été le plus rapide ou deux fois si c'est celui de droite.

Quand le jeu est terminé, la led s'éteint.

Complétez le programme suivant (là où il y a des commentaires) :

```
# -*- coding: utf-8 -*-
from gpiozero import Button,LED
import time
import random


def Gauche():
    #arret est une variable globale
    #on éteint la led
    #on la fait clignoter une seule fois
    print("C'est le joueur de gauche qui gagne")
    arret = True


def Droite():
    #arret est une variable globale
    #on éteint la led
    #on la fait clignoter une seule fois
    print("C'est le joueur de droite qui gagne")
    arret = True

bpG = Button(20)
bpD = Button(16)
led = LED(21)
arret = False

#5 clignotements pas en arrière plan
# la led est éteinte
#on dort pendant 2 secondes
time.sleep(random.uniform(1,3))  #uniform renvoi un réel
#la led est allumée -> qui sera le plus rapide ?

bpG.when_pressed = # ???
bpD.when_pressed = #???


while arret==False:
    time.sleep(0.1)

print("Fin du programme")
```

V. Parcours 1 : un petit jeu pour terminer l'exploration des gpio

1. Quelle couleur vient de s'allumer ?

Pour ce jeu, il faut une led RGB et trois boutons associés aux trois couleurs : Rouge, Vert et Bleu.
De façon aléatoire, le programme choisit une des trois couleurs et le joueur doit appuyer le plus rapidement possible sur le bon bouton. Simple non ?

2. Notions python utilisées pour cette première version

a. Présentation des chaînes, listes et tuples

- Les **chaines de caractères** ne sont pas modifiables (non mutable) :

```
>>> chaine='Roméo et Juliette'
```

- Les **listes** sont des sortes de tableaux pouvant contenir tous les types de données se trouvant entre crochets[], ils sont mutables :

```
liste =[ ' jambon ' , ' fromage ' , ' miel ' , ' confiture ' , ' chocolat ']
liste2=[1,2,3]
liste3=['toto',2,3.5,True,['b',liste2]]
```

- Les **tuples** ≈ listes non mutables, que l'on ne peut pas modifier, les données sont entre parenthèses ()

```
>>> tuple=('a','b','c','d')
```

- Les **dictionnaires**, mutables, portent bien leurs noms. Ils associent à un élément, un autre élément. Les éléments sont entre accolades

```
dicoAnglais={'computer': 'pc', 'mouse': 'souris', ' keyboard ': ' clavier '}
```

Appelez le professeur

b. Liste et aléatoire

Soit le code suivant :

```
rouge_rgb = (1,0,0) # couleur RGB rouge vert bleu
vert_rgb = (0,1,0)
bleu_rgb = (0,0,1)
couleurs_rgb = [rouge_rgb,vert_rgb,bleu_rgb]
CouleurHasard = random.choice(couleurs_rgb)
```

Sans tester le code, quelle valeur va prendre CouleurHasard ? Comment pourra-t-on l'utiliser avec une ledRGB ?

3. Détail du jeu

Etape N°1 : la ledRGB clignotera 5 fois (temps allumé=0.5, temps éteint=0.2) pas en arrière plan mais bloquant
Etape N°2 : la led RGB allumera une couleur des trois couleurs, choisie au hasard
Etape N°3 : si le bouton correspondant à la couleur est appuyé alors le joueur aura gagné et la couleur verte sera allumée. Sinon c'est la rouge.

Implémentez cette première version

4. Deuxième version avec affichage de la vitesse

Inspirez-vous du code suivant pour afficher la vitesse à laquelle le joueur aura appuyé sur le bon bouton

```
debut = time.clock()
time.sleep(2)
fin = time.clock()
print(fin - debut)
```

5. Dernière version avec une liste des boutons – extension -

Expliquez le programme suivant

```
ListeDesBoutons = [Button(2), Button(3), Button(4)]

for CouleurATrouver in CouleursJeu:
    BoutonAppuye = False
    while BoutonAppuye == False:
        for UnBouton in LesBoutons:
            if UnBouton.is_pressed:
                BoutonAppuye = True
```

Qu'affiche ce programme ?

```
ListeDesBoutons = [Button(2), Button(3), Button(4)]
print(LesBoutons.index(UnBouton))
```

En utilisant les deux notions vues précédemment, modifiez votre code

VI. Parcours 2 : jeu Simon

1. Présentation du jeu Simon

Le jeu éclaire une des quatre couleurs. Le joueur doit alors appuyer sur la touche de la couleur qui vient de s'allumer.

Le jeu répète la même couleur, puis ajoute au hasard une nouvelle couleur. Le joueur doit reproduire cette nouvelle séquence. Chaque fois que le joueur reproduit correctement la séquence, le jeu ajoute une nouvelle couleur.

2. Notions python utilisées

a. Présentation des chaînes, listes et tuples

- Les **chaines de caractères** ne sont pas modifiables (non mutable) :

```
>>> chaine='Roméo et Juliette'
```

- Les **listes** sont des sortes de tableaux pouvant contenir tous les types de données se trouvant entre crochets[], ils sont mutables :

```
liste =[ ' jambon ' , ' fromage ' , ' miel ' , ' confiture ' , ' chocolat ']
liste2=[1,2,3]
liste3=['toto',2,3.5,True,['b',liste2]]
```

- Les **tuples** ≈ listes non mutables, les données sont entre parenthèses ()

```
>>> tuple=('a','b','c','d')
```

- Les **dictionnaires**, mutables, portent bien leurs noms. Ils associent à un élément, un autre élément. Les éléments sont entre accolades

```
dicoAnglais={'computer': 'pc', 'mouse': 'souris', ' keyboard ': ' clavier '}
```

Appelez le professeur

b. La method append

Cette method permet d'ajouter un element à la fin de la liste.
Soit le code :

```
nombres =[]
nombres.append(5)
print(nombres)
nombres.append(6)
print(nombres)
```

Sans le tester, quel est l'affichage ? Complétez le pour ajouter deux autres nombres

Dans le programme suivant la liste a été initialisée avec des valeurs :

```
nombres =[2,3,4]
nombres.append(5)
print(nombres)
nombres.append(6)
print(nombres)
```

c. Une list avec des tuples

Un tuple ressemble à une liste mais non mutable, c'est-à-dire qu'il est fixe, qu'on ne peut pas ajouter des valeurs.

Les données sont entre parenthèses ()

```
>>> tuple=('a','b','c','d')
```

Testez le programme suivant , quelle est l'erreur ?

```
Untuple=('a','b','c','d')
Untuple.append('2')
```

d. For … in

Exemple

```
liste=['Cédric','Loïc','Joelle','Pierre']
for nom in liste:
    print('La longueur de '+nom+' est '+str(len(nom)))#mettre des espaces
```

L'affichage est :
La longueur de Cédric est 6
La longueur de Loïc est 4
La longueur de Joelle est 6
La longueur de Pierre est 6

e. Exercice : avec les leds ou boutons poussoirs

Créez une liste de trois leds sur les ports 20,21 et 22 puis en utilisant une boucle for .. in, allumez chacune d'elle pendant une seconde. Il n'est pas nécessaire de faire le montage. Faites valider par le professeur.

Expliquez le programme suivant

```
ListeDesBoutons = [Button(2), Button(3), Button(4)]

for CouleurATrouver in CouleursJeu:
    BoutonAppuye = False
    while BoutonAppuye == False:
        for UnBouton in LesBoutons:
            if UnBouton.is_pressed:
                BoutonAppuye = True
```

f. Un programme de test

Soit le programme :

```
1. import random
2.
3. liste = ["choix 1","choix 2","choix 3"]
4. print(liste)
5. print(liste[0])
6. print(random.choice(liste))
7.
8. unTuple = (1,2,3)
9. unAutreTuple =(2,3,4)
10.  uneListe =[unTuple,unAutreTuple]
11.  print(uneListe)
12.  print(uneListe[0])
13.  print(random.choice(uneListe))
14.
```

```
15. unDictionnaire = {23:(1,2,3),25:(4,5,6),26:(7,8,9)}
16. print(unDictionnaire[25])
17.
```

Expliquez toutes les lignes et l'affichage obtenu.

3. Présentation du jeu : rappel

Le jeu éclaire une des quatre couleurs. Le joueur doit alors appuyer sur la touche de la couleur qui vient de s'allumer.

Le jeu répète la même couleur, puis ajoute au hasard une nouvelle couleur. Le joueur doit reproduire cette nouvelle séquence. Chaque fois que le joueur reproduit correctement la séquence, le jeu ajoute une nouvelle couleur.

La plateforme « Raspberry Pi » est composée de 3 BP et d'une RGB.

4. Cahier des charges :

- Les leds seront allumées pendant 1 seconde
- La couleur Bleu de la RGB indiquera le début imminent du jeu
- Si la séquence est correcte, c'est la led Verte qui s'allumera
- Sinon, c'est la Rouge qui clignotera pendant 3 secondes et le jeu redémarrera

5. Conseils d'implémentation :

- Mettez en place des fonctions
- Définissez :
  - Une liste pour les trois boutons
  - Une liste des couleurs qui doivent être affichées au fur et à mesure du jeu
  - Ajoutez à cette liste, les couleurs RGB : rouge_rgb =(1,0,0) vert_rgb =(0,1,0) et bleu_rgb=(0,0,1) au hasard
- Utilisez :
  - Deux *for ..in ..* : une pour les couleurs à afficher par la led rgb, une autre pour détecter l'appui sur un des bouton
  - Une variable booléenne BP_appuye
  - Un *break* permet de sortir de la boucle

6. Code du jeu

Proposez un algorithme au professeur

Implémentez ce jeu avec une Rpi