Dans toute la suite, on va travailler avec le mode « Board » qui implique la numérotation suivante des brochages :

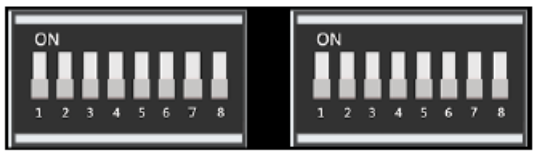
ASSIGNMENT OF THE GPIO PINS ACCORDING TO GPIO.BOARD SCHEME

GPIO-Board Number:	Used sensors and modules:
1	3.3V
2	5.0V
3	I2C, SDA1 (Licht Sensor, LCD Display, 7 Segment Display)
4	5.0V
5	I2C, SCL1 (Light Sensor, LCD Display, 7 Segment Display)
6	Ground
7	DHT11 Sensor
8	TXD0
9	Ground
10	RXD0
11	Touch Sensor
12	Buzzer
13	Button matrix (ROW1), Vibration motor
14	Ground
15	Button matrix (ROW2), Tilt sensor
16	Motion sensor
17	3.3V
18	Sonic sensor
19	SPI
20	Ground
21	SPI
22	Servo2, Button matrix (COL1), Left Button
23	SPI
24	RFID Modul
25	Ground
26	LED-MATRIX
27	ID_SD (I2C, EEPROM(Electrically Erasable Programmable Read-only Memory))
28	ID_SC
29	Stepper Motor (STEP1), Button matrix (ROW3)
30	Ground
31	Stepper Motor (STEP2), Button matrix (ROW4)
32	Ultrasonic sensor (Echo)
33	Stepper Motor (STEP3), Buttonmatrix (COL4), Down Button
34	Ground
35	Stepper Motor (STEP4), Buttonmatrix (COL3), Right Button
36	Ultrasonic sensor (TRIG)
37	Servo1, Button matrix (COL2), Up Button
38	Infrared sensor
39	Ground
40	Relais

On utilisera pour cela la ligne de code suivante :

```
GPIO.setmode(GPIO.BOARD)
```

Pour l'ensemble des applications qui suivent, il sera nécessaire de relier carte Raspberry aux différents composants à l'aide de switches :



On donne ci-dessous les lignes de commandes qui seront utilisées pour lire ou écrire les données numériques des capteurs et actionneurs de la mallette :

```
import RPi.GPIO as GPIO# importation de la librairie GPIO
GPIO.setmode(GPIO.BOARD) # Mode GPIO.BOARD

#configuration en mode sortie d'un actionneur (relai, buzzer, vibreur,….)
GPIO.setup(N°pin, GPIO.OUT)
#sortie mise à l'état haut
GPIO.output(N°pin, GPIO.HIGH)
#sortie mise à l'état bas
GPIO.output(N°pin, GPIO.LOW)


#configuration en mode entrée d'un récepteur (bouton, capteur numérique)
GPIO.setup(N°pin, GPIO.IN)
#lecture d'une entrée (valeur 0 ou 1 lue)
GPIO.input(N°pin)


GPIO.cleanup()#afin de lâcher la liaison
```

Rq 1 : Vous travaillerez sous sypder et avec une « ctrl +c » vous pourrez forcer l'arrêt de vos programmes dans la fenêtre console.

Rq 2 : Avec la librairie time, vous pourrez mesurer le temps depuis un instant de départ :

```
import time
t_debut = time.time()
t=time.time()-t_debut
```

Exemple 1 : Elaboration de la sonnerie d'un réveil

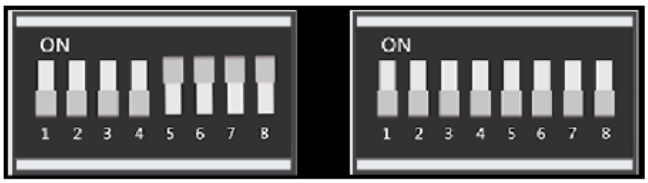
On donne le code ci-dessous permettant de mettre en marche un buzzer pendant 0,5s.

```
import RPi.GPIO as GPIO
import time
GPIO.setmode(GPIO.BOARD)

buzzer = 12
GPIO.setup(buzzer, GPIO.OUT)
GPIO.output(buzzer, GPIO.HIGH)
time.sleep(0.5)
GPIO.output(buzzer, GPIO.LOW)
GPIO.cleanup()
```

En vous inspirant du programme ci-dessus et en utilisant une boucle While, élaborer un programme permettant de réaliser une sonnerie fonctionnant par alternance (une seconde de sonnerie puis une seconde de silence) et cela pendant moins de 10s.

Exemple 2 : Commande manuel d'un buzzer

Voici la configuration des Switches pour cet exercice :



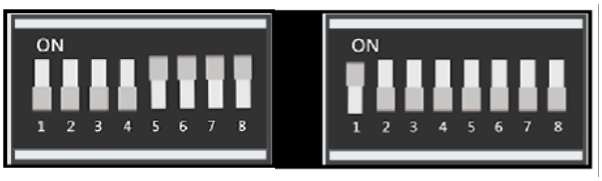
On donne le code ci-dessous permettant de faire fonctionner un buzzer lorsqu'on appuie sur le bouton « up » de la mallette

```
import RPi.GPIO as GPIO
import time
GPIO.setmode(GPIO.BOARD)
#affectation des numéros de pins
bouton = 37#up bouton
buzzer = 12

#initialisation du temps
t_debut=time.time()

# bouton fixé comme une entrée
GPIO.setup(bouton, GPIO.IN)
# buzzer fixé comme une sortie
GPIO.setup(buzzer, GPIO.OUT)

t=time.time()-t_debut
while t<10:
    #si on appuie sur le bouton
    if(GPIO.input(bouton) == 0):
            GPIO.output(buzzer,
GPIO.HIGH)
    else:
    # si on n'appuie pas sur le bouton
            GPIO.output(buzzer,
GPIO.LOW)
    t=time.time()-t_debut
GPIO.cleanup()
```

Modifier ce code pour que le buzzer n'émette aucun son lorsqu'on appuie sur le bouton « up ».

Exemple 3 : Commande manuelle d'un buzzer et d'un vibreur

Voici la configuration des Switches pour cet exercice :



On donne ci-dessous le code permettant de faire vibrer le vibreur présent sur la mallette pendant 2s.

```
import RPi.GPIO as GPIO
import time
# Mode Board
GPIO.setmode(GPIO.BOARD)

# affectation du capteur
vibration = 13

# vibreur en mode sortie
GPIO.setup(vibration, GPIO.OUT)

# Activation du vibreur
GPIO.output(vibration, GPIO.HIGH)
# temps de vibration
time.sleep(2)
# arrêt du vibreur
GPIO.output(vibration, GPIO.LOW)
# GPIO relaché
GPIO.cleanup()
```

Modifier ce code afin de pouvoir activer le vibreur à l'aide du bouton « up » et le buzzer à l'aide du bouton « left » pendant 10s:

Exemple 4 : Capteur de bruit

Matériels supplémentaires : tournevis

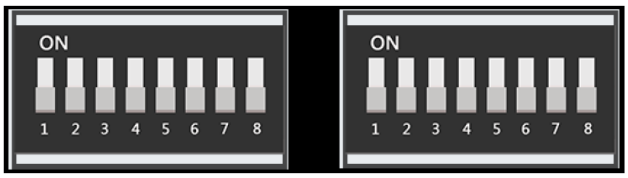
Le capteur (pin 18) étudié ici est un capteur de son acoustique. Ce n'est pas un microphone, il détecte la présence d'un son en délivrant un état bas.

La sensibilité de ce capteur se règle à l'aide du potentiomètre (capteur sensible, potentiomètre à droite).

Proposer un programme permettant d'actionner le buzzer pendant une seconde si un bruit est détecté (et le toute pendant 10s).

Exemple 5 : Détecteur de mouvement

Voici la configuration des Switches pour cet exercice :



On donne ci-dessous le code permettant de faire fonctionner le capteur de mouvement présent sur la mallette :

```
import RPi.GPIO as GPIO
import time

# affectation capteur
motion_pin = 16

# Mode BOARD
GPIO.setmode(GPIO.BOARD)
# Capteur déclarée en entrée
GPIO.setup(motion_pin, GPIO.IN)

#initialisation du temps
t_debut=time.time()

t=time.time()-t_debut
while t<10:
    if(GPIO.input(motion_pin) == 0):
        print ("pas de mouvement...")
    elif(GPIO.input(motion_pin) == 1):
        print ("Mouvement détecté!")
    time.sleep(0.1)
    t=time.time()-t_debut
GPIO.cleanup()
```

Proposer un code permettant de déclencher le buzzer lorsqu'un mouvement est détecté.