Micro-essai : Utilisation de l'accéléromètre intégré de la carte BBC Micro:bit

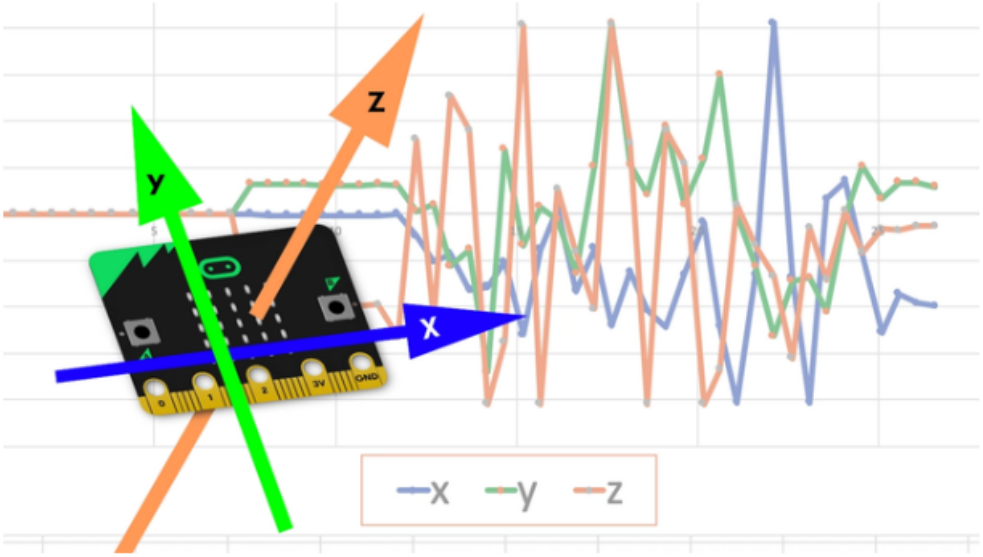
(Merci à Sandrine Gaudin pour son travail qui m'a permis de réaliser l'essai!)

1/ Présentation du capteur accéléromètre intégré à la carte BBC Micro:bit

> - La carte BBC micro:bit possède un accéléromètre 3 axes intégré.
> Il est capable de mesurer des accélérations dans la gamme $[-2g; 2g]$ et il affiche les valeurs en " $milli$ " g.
> Il est sensible à l'accélération de la pesanteur.
> - Je ne trouve pas très clair les informations officielles fournies et je n'ai pas trouvé d'informations sur la précision du capteur pour le moment. Voici un extrait de la notice :
> `Hold the micro:bit flat with the LEDs uppermost. You should see that the X and Y accelerations are near to zero, and the Z acceleration is close to -1024. This tells you that gravity is acting downwards relative to the micro:bit. Flip the board over so the LEDs are nearest the floor. The Z value should become positive at +1024 milli-g. If you shake the micro:bit vigorously enough, you'll see that the accelerations go up to ±2048 milli-g. That's because this accelerometer is set to measure a maximum of ±2048 milli-g: the true number might be higher than that.`

In [1]:
```
from IPython import display
from base64 import b64decode
base64_acc1="iVBORw0KGgoAAAANSUhEUgAAAzMAAAFnCAIAAAAdSYApAAAAAXNSR0IArs4c6QAAAARnQU1BAACxjv
display.Image(b64decode(base64_acc1),width=700)
```

Out[1]:



> **Pour pouvoir utiliser l'accéléromètre intégré :**
>
> - Il faut alimenter la carte en mode embarqué : Sandrine et moi-même avons trouvé un module d'alimentation plus pratique que le branchement de pile via un module

> `MI-power` se vissant sur la carte et l'alimentant par pile bouton.
> - Il faut pouvoir stocker les données en mode embarqué dans la mémoire flash de la carte ($256\ ko$).

In [2]:
```
from IPython import display
from base64 import b64decode
base64_acc2="iVBORw0KGgoAAAANSUhEUgAAA2EAAAHNCAIAAAAYL+TUAAAAAXNSR0IArs4c6QAAAARnQU1BAACxjw
display.Image(b64decode(base64_acc2),width=300)
```

Out[2]:

> Voici la méthode permettant de lire les valeurs de l'accélération :
> `microbit.accelerometer.get_x()`
> `microbit.accelerometer.get_y()`
> `microbit.accelerometer.get_z()`
>
> La méthode « get_x » (resp. « get_y », « get_z ») retourne l'accélération sur l'axe des x (resp. y, z) en mg, sous la forme d'un entier positif ou négatif, en fonction de la direction de l'accélération.

2/Programmation du micro-contrôleur

J'ai réalisé un programme en Micro-Python permettant de récupérer les données dans un fichier `csv` stocké dans la mémoire flash de la carte.
Deux points à préciser :

> - Je ne relève les mesures que deux secondes après le lâcher de la masse.
> - Il ne faut pas être trop exigeant en nombre de points car il n'y a que $16ko$ de mémoire vive et on la sature assez facilement lors de l'execution de la boucle `for` qui remplit les listes!

In [3]:
```
# import microbit as m
# while True:
#     if m.button_a.is_pressed():
#         m.display.clear()
#         m.display.scroll("GO")
#         t0 = m.running_time()                    #initialisation du temps
#         t = []                                   #création du liste t vide
#         acc = []                                 #création du liste acc vide
#         m.sleep(2000)                            #attente de 2 secondes
#
#         for i in range(200):                     #enregistrement de 200 points
#
#             tn = m.running_time()-t0
#             yn = m.accelerometer.get_y()
#             t.append(tn)
#             acc.append(yn)
```

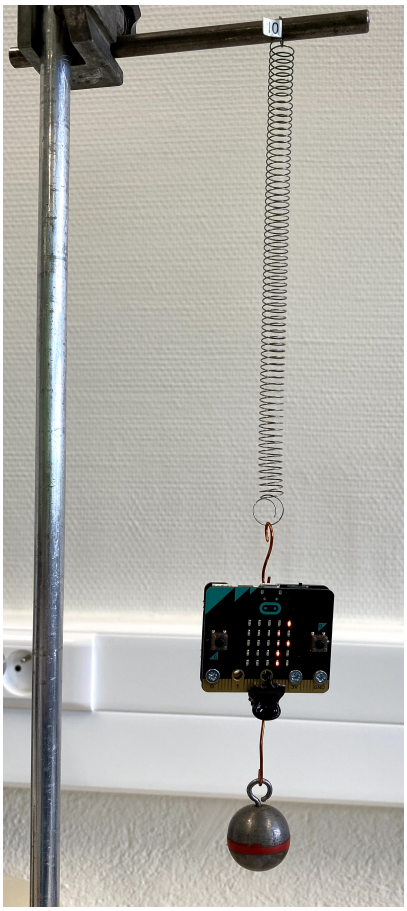
```
#             m.sleep(20)                           #attente de 20 ms entre chaque mesure
#
#         with open('ACC.csv', 'w') as f:           #ouverture du fichier csv
#                                                   #et écriture des couples (t,acc)
#             for i in range(len(t)):
#                 f.write(str(t[i])+";"+str(acc[i])+"\n")
#         m.display.scroll('FIN')
#     else:
#         m.display.scroll("A")
```

3/Montage expérimental

> - A la demande d'Isabelle Quinot, Sandrine Gaudin et moi-même avons réalisé les essais sur l'oscillation verticale masse-ressort.
> Voici le montage proposé :

In [4]:
```
from IPython import display
from base64 import b64decode
base64_acc3="/9j/4AAQSkZJRgABAQAAAQABAAD/4RVkRXhpZgAATU0AKgAAAAgADAEPAAIAAAAGAAAAngEQAAIAAA
display.Image(b64decode(base64_acc3),width=200)
```

Out[4]:



4/Traitement des données avec Python

> - **Récupération des données issues du csv enregistré sur la carte BBC Micro:bit**

In [5]:
```
import csv
def readCSV(file,sep,n):
    with open(file,"r") as f:
        read=csv.reader(f,delimiter=sep)
        col=[]
```

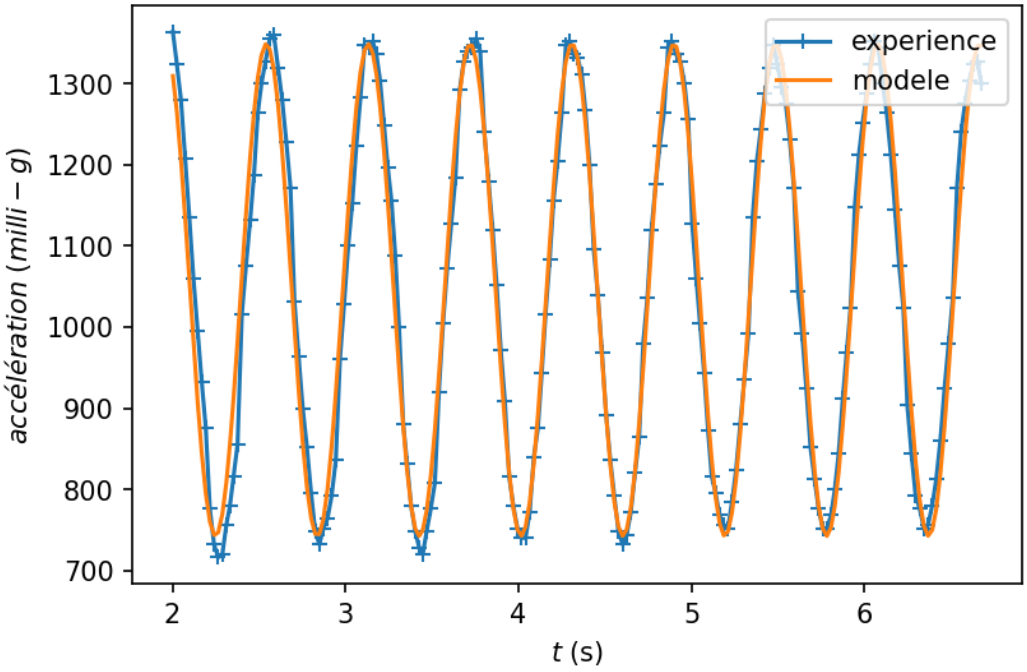
```
        for row in read :
            try:
                col.append(float(row[n].replace(",","")))
            except:
                pass
    return col

#listes des valeurs experimentales
tms=readCSV("ACC_mvt_95,6_k10.csv",";",0) #temps en ms
#print (t)

acc=readCSV("ACC_mvt_95,6_k10.csv",";",1) #accélération verticale
#print(acc)
```

> - **Modélisation de l'accélération en fonction du temps et tracés des courbes expérimentale et modèle**

In [7]:

```
import numpy as np
import matplotlib.pyplot as plt
import scipy as sp
from scipy.optimize import curve_fit

# Initialisation des paramètres du modele:
a=350
w=10
phi=-1
b=1050.
p0=[a,w,phi,b]

#definition du modele
N=len(tms)
ntms=np.array(tms)                #temps en ms
nt=ntms/1000                      #temps en s
nacc_modele=np.zeros_like(nt)
def func(nt,a,w,phi,b):           #modèle
    for i in range(N):
        nacc_modele[i]=a*np.cos(w*nt[i]+phi)+b
    return nacc_modele

#optimisation
(a_opt,w_opt,phi_opt,b_opt),_=curve_fit(func,nt,acc,p0,
                                bounds=[(100, 0,-2,0),(400,20,0,2000)])

#tracés des courbes experimentale et modèle
plt.figure(dpi=150)
plt.plot(nt,acc,'-+',label='experience')
plt.plot(nt,func(nt,a_opt,w_opt,phi_opt,b_opt),label='modele')
plt.xlabel('$t$ (s)')
plt.ylabel("$accélération$ ($milli-g$)")
plt.title(r"modelisation $acc_{{th}}=a\cdot\cos(\omega t+\phi)+b$ avec"
          r"$a$ = {:.2f} SI, $\omega$={:.2f} SI, $\phi$={:.2f} SI,"
          r"$b$={:.2f} SI".format(a_opt,w_opt,phi_opt,b_opt), fontsize=10)
plt.legend(loc = 'upper right')
plt.show()
```

modelisation $acc_{th} = a \cdot \cos(\omega t + \phi) + b$ avec$a$ = 303.37 SI, ω=10.67 SI, ϕ=-2.00 SI,b=1045.77 SI



- **Exploitation : détermination de la constante de raideur du ressort**

L'experience a été réalisée avec une masse totale $M_t = 95,6g$ (masse de la carte microbit +masse sphérique).
À l'aide du modèle, on détermine $\omega = 10,67\ rad.\ s^{-1}$.
On en déduit $k = \omega^2 \times M_t = 10,67^2 \times 0,0956 = 10,9\ N.m^{-1}$ "à priori" en accord avec le constructeur ($k_{constructeur} = 10\ N.m^{-1}$).