TP - Loi de Hook

Matériel

- ◇ Acquisition : Micro-contrôleur MicroBit et connecteur vers une plaque de prototypage électronique avec fils adaptés.
- ◇ Module HX711.
- ◇ Jauge de contrainte.
- ◇ Ressort (raideur d'une centaine de N/m ou inférieur), jeu de masselottes (de 50 g à 1 kg), serre-joint.
- ◇ Règles.

Utilisation du matériel

- ☠ Le micro-contrôleur utilisé est extrêmement sensible aux court-circuits. Il faudra être à 200% sûr de votre montage avant de le brancher à une source de courant.
- **Remettre en l'état avant de quitter la salle (éteindre les appareils, débrancher les fils et les remettre aux bons emplacements (paillasse derrière la votre), ranger les composants utilisés).**

Documentation

FIGURE 1 – Fiche technique : https://cdn.sparkfun.com/datasheets/Sensors/ForceFlex/hx711_english.pdf et fiche de présentation générale : https://www.sparkfun.com/products/13879 du HX711.
Le site https://python.microbit.org/v/2 vous permet de coder en micropython les instructions à envoyer au micro-contrôleur. Les informations sur micropython et sur les broches du micro-contrôleur sont disponibles sur le site https://microbit-micropython.readthedocs.io/fr/latest/pin.html. Enfin pour ajuster un modèle de signal sinusoïdal sous python et en extraire les paramètres comme la période des oscillations, l'utilisation de scipy.optimize.curve_fit() sera nécessaire.

Au programme

Notions et contenus	**Capacités exigibles**
Deuxième loi de Newton.	Mettre en oeuvre un protocole expérimental permettant d'étudier une loi de force par exemple à l'aide d'un microcontrôleur.
Nature et méthode	**Capacités exigibles**
Quantifier une action.	Utiliser un dynamomètre.
—*contexte différent*—	Choisir le capteur en fonction de ses caractéristiques (linéarité, sensibilité, gamme de fonctionnement, temps de réponse), et du type de mesures à effectuer.
—*contexte différent*—	Étalonner une chaîne de mesure si nécessaire.

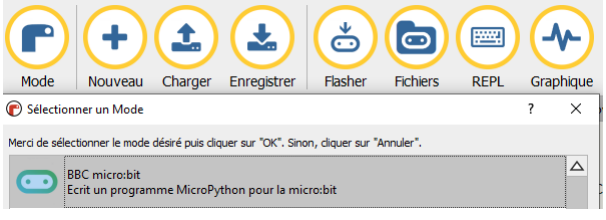
Objectif

Le but de ce TP est de retrouver expérimentalement la loi de Hook et d'utiliser celle-ci pour construire un dynamomètre ou un capteur de position à l'aide d'un micro-contrôleur.

1 Micro-contrôleur

Un micro-contrôleur est un dispositif électronique programmable. Son architecture est semblable à celle des smartphones, tablettes et ordinateurs dans le sens qu'il contient les mêmes éléments électroniques de base : un processeur, une mémoire programmable, une mémoire tampon et des entrées et sorties numériques (ou analogiques grâce à un CAN intégré). Les micro-contrôleurs sont souvent utilisés pour enregistrer ou traiter des données en temps réel dans des systèmes embarqués, leurs points forts étant d'être peu onéreux et peu consommateurs d'énergie.
Le micro-contrôleur utilisé dans ce TP est appelé "micro :bit" (https://microbit.org/). Il possède plusieurs capteurs ou émetteurs intégrés (comme votre smartphone) : antenne bluetooth et radio, microphone, mini-enceinte, accéléromètre et magnétomètre.
Nous utiliserons l'éditeur "mu-editor" pour coder en micro-python les instructions souhaitées, les envoyer au micro-contrôleur et visualiser les données transmises au PC par ce dernier (pour les versions antérieures à Windows 10 téléchargez le driver disponible ici).
Cet éditeur est assez sommaire, voici un bref aperçu des onglets utilisés :

- — Mode d'utilisation : choisir BBC micro :bit,
- — Nouveau : génère une nouvelle page de code,
- — Flasher : permet de compiler le code édité sur le micro-contrôleur,
- — Fichier : permet de visualiser les fichiers sur le micro-contrôleur et sur l'ordinateur (par défaut le fichier utilisé par mu-editor sur l'ordinateur est placé dans "C :\Users\votre nom d'utilisateur\mu_code\")



— REPL (Read Evaluate Print Loop) : console permettant de programmer directement sur le micro-contrôleur,

— Graphique : affiche les données envoyées (s'il y en a !) par le micro-contrôleur.

L'icône en bas à droite permet de vérifier si le micro-contrôleur est bien connecté.

Commencez par connecter le micro-contrôleur à l'ordinateur. Ouvrez l'éditeur "mu-editor" et essayez un de ces deux codes :

```
from microbit import *
from math import sqrt
while True:
    (x, y, z) = accelerometer.get_values()
    print((x, y, z, sqrt(x**2 + y**2 + z**2)))
    sleep(50) # attend 50 milisecondes
```

Ouvrez l'onglet graphique pour visualiser les données.

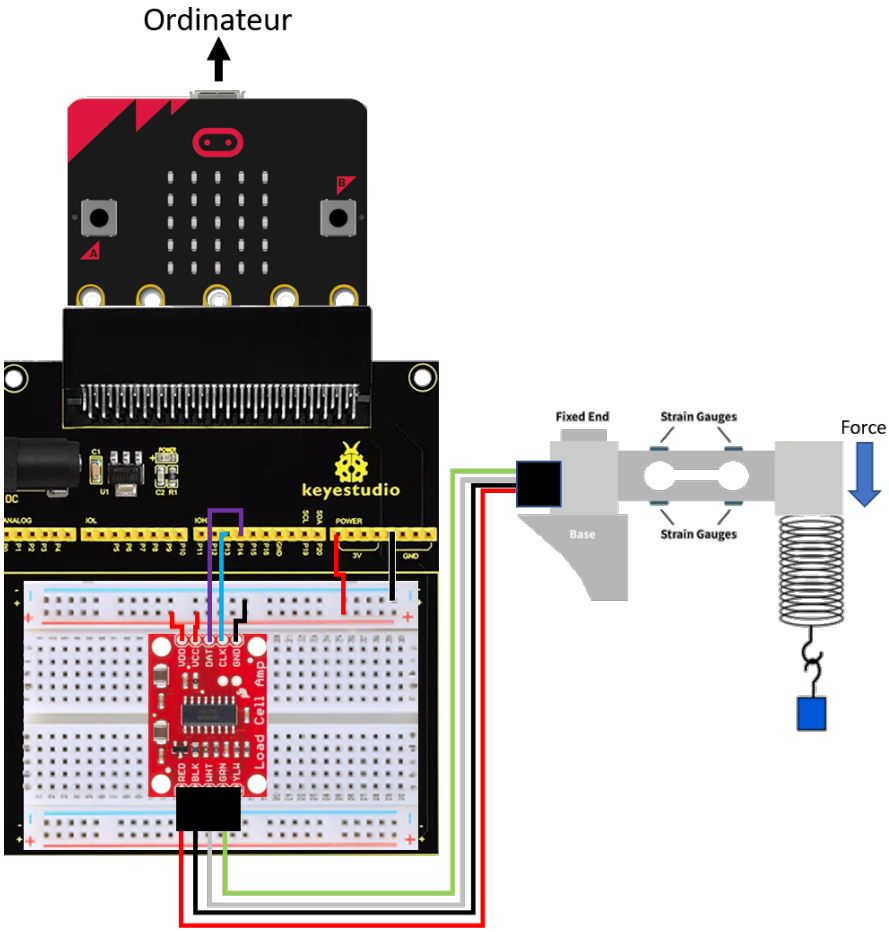
```
from microbit import *
import music
notes = ['c4:1', 'e', 'g', 'c5', 'e5']
while True:
    if button_a.was_pressed():
        display.scroll("Bonjour")
    if button_b.was_pressed():
        music.play(notes)
```

2 Jauge de contrainte

Débrancher le micro-contrôleur de l'ordinateur et mettre en place le montage suivant. **Appeler le professeur avant de connecter le micro-contrôleur à l'ordinateur.** Fixez une extrémité du capteur de force sur la table à l'aide d'un serre-joint (ne serrez pas la pâte blanche, uniquement la partie métallique) et placez l'autre extrémité dans le vide pour y suspendre un ressort et une masse.

— La broche 13 (SCK) du micro-contrôleur est à relier avec l'entrée "CLK" du HX711 (signal d'horloge),

— La broche 14 (MISO) du micro-contrôleur est à relier avec l'entrée "DAT" du HX711 (données).



Un capteur de force est constitué de quatre jauges de contrainte. Ce sont des fils ou des feuillets conducteurs dont la géométrie est choisie pour optimiser l'effet de variation de résistivité lorsqu'ils sont soumis à des déformations. Ces jauges peuvent être vues comme des résistances variables contrôlées par déformation. Les variations de résistivités sont si faibles que ces résistances sont souvent assemblées par quatre dans un montage électrique appelé pont de Wheatstone, permettant d'obtenir un signal électrique mesurable traduisant la variation de résistivité de ces jauges. Le module HX711 couplé avec ce capteur de force correspond à un pont de Wheatstone dont la valeur de tension de sortie est convertie en signal numérique avant d'être transmise au micro-contrôleur (par le protocole de communication "SPI"). Le code que doit compiler le micro-contrôleur pour lire et afficher sur la console mu-editor les valeurs transmises par le HX711 s'écrit :

```
from microbit import *
spi.init(baudrate=1000000, bits=8, mode=0, sclk=pin13, mosi=pin15, miso=pin14)
tare = 0
while True:
    a = spi.read(6)
    bit1 = ord(a[0:1].decode())
    bit2 = ord(a[1:2].decode())
    # bit1 = (bit1+2**6) #décallage pour les valeurs négatives
    value = int( bit1 << 8 ) + int( bit2 ) - tare
    if button_a.was_pressed():
        tare = int( bit1 << 8 ) + int( bit2 )
    print((value,))
    sleep(100)
```

Compiler sur le micro-contrôleur le code pour afficher la valeur émise par le module HX711. Appuyez sur le bouton A du micro-contrôleur puis afficher les valeurs transmises dans l'onglet graphique de la console mu-editor. Appuyez sur la jauge de déformation et constatez le changement de valeur mesurée.
Quel est le rôle du bouton A ? Dans quelle condition l'utiliser ?

3 Loi de Hook

En modifiant la masse m suspendue au ressort, la longueur du ressort à l'équilibre ℓ_e varie : $\ell_e = \ell_0 + \frac{mg}{k}$, et la force mesurée par la jauge varie : $F = k(\ell_e - \ell_0) = mg$.

Effectuez une série de mesures avec le montage précédent permettant de relier, pour différentes masses : la longueur du ressort, la valeur mesurée par le capteur de contrainte et la masse.
Utilisez la console Spyder pour y rentrer les valeurs que vous allez mesurer.
Utilisez la bibliothèque numpy pour ajuster un modèle affine à vos données (cf exemple) et en déduire la raideur du ressort.
Ajuster un modèle affine à vos données pour en déduire la courbe d'étalonnage du capteur, permettant de relier la masse ou la longueur du ressort à une valeur affichée par la console mu-editor. Vous pourrez modifier le code à transmettre au micro-contrôleur pour utiliser cet étalonnage.
En déduire que le capteur de force devient un capteur de position.

Le code ci-dessous permet de retrouver la valeur de pente (ici "3") et l'ordonnée (ici "-2") d'une fonction affine :

```
X = np.array([0,1,2,3,4,5,6,7,8,9,10])
Y = -2 + 3 * np.array([0,1,2,3,4,5,6,7,8,9,10])
fit = np.polyfit(X,Y,2) # fit est une liste de valeurs estimées, ici fit = [pente, ordonnée]
fit_func = np.poly1d(fit)
plt.figure()
plt.plot(X, Y, '+', label = "données")
plt.plot(X, fit_func(X), label = "modèle")
plt.legend()
plt.show()
print(fit[0],fit[1])
```

4 Mesure de g

Utilisez le montage précédent en dynamique : mettre en mouvement la masse pour percevoir une oscillation avec le capteur de force. Sachant que le capteur ne peut mesurer que des points espacés de 100 ms, prendre une masse élevée pour avoir la période d'oscillation la plus faible possible.
En utilisant la courbe obtenue avec le capteur de force et la mesure de la raideur du ressort, proposer et mettre en place un protocole permettant d'en déduire l'accélération de la pesanteur g.