Evaluation 3 : Une expérience avec Arduino et un accéléromètre

CHEN Virginie
GOGRITCHIANI Lasha

5 janvier 2023

Introduction

Dans les premiers TP réalisés au cours de cette UE de physique expérimentale, nous avons utilisés l'application phyphox qui nous permettait d'exploiter le capteur d'accélération contenu dans notre smartphone. Notre but cette fois ci est d'utiliser un accéléromètre et d'en exploiter les données nous même à l'aide d'Arduino. Pour, finalement, réaliser des expériences à l'aide de ce capteur.

Capturer les données

Tout d'abord, il nous faut pour ce TP un capteur MPU-6050 qui permet de mesurer l'accélération sur trois axes : x, y, z. Ensuite, il faut procéder au montage comme sur la figure 1 ci-dessous. Pour finalement lire les données qui sont enregistrées à l'aide de notre MPU-6050, il va nous falloir un code Arduino. Pour écrire ce code, nous avons besoin de plusieurs bibliothèques. La bibliothèque Adafruit_MPU6050.h nous permet de communiquer avec le MPU-6050.

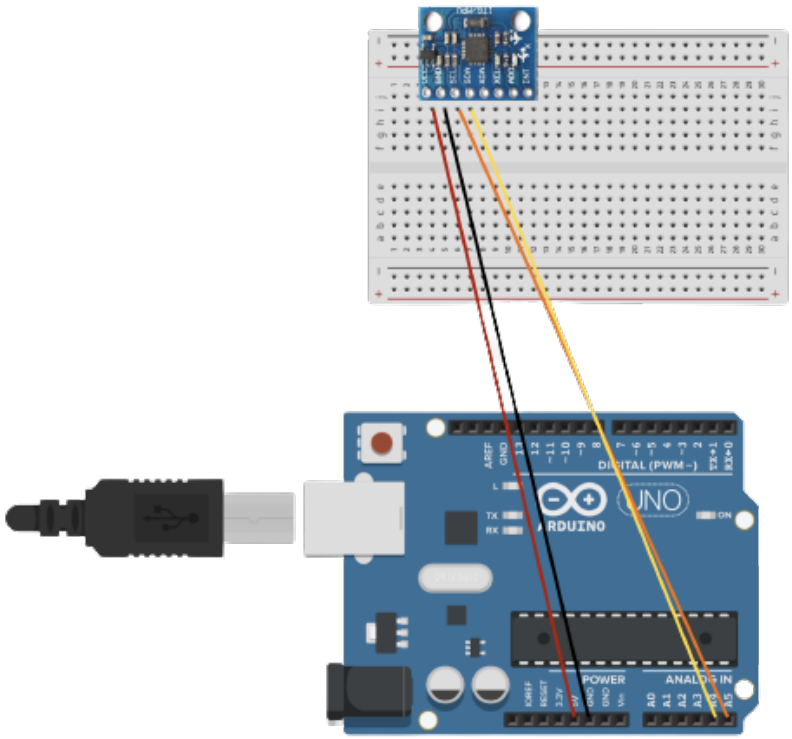

FIGURE 1 – Schéma du montage réalisé.

Installation des bibliothèques

Ouvrir l'application Arduino IDE. Suivre le chemin suivant : **Sketch** > **Include Library** > **Manage Libraries**.

Taper "adafruit mpu6050" dans la barre de recherche et installer la bibliothèque de la Figure 2.

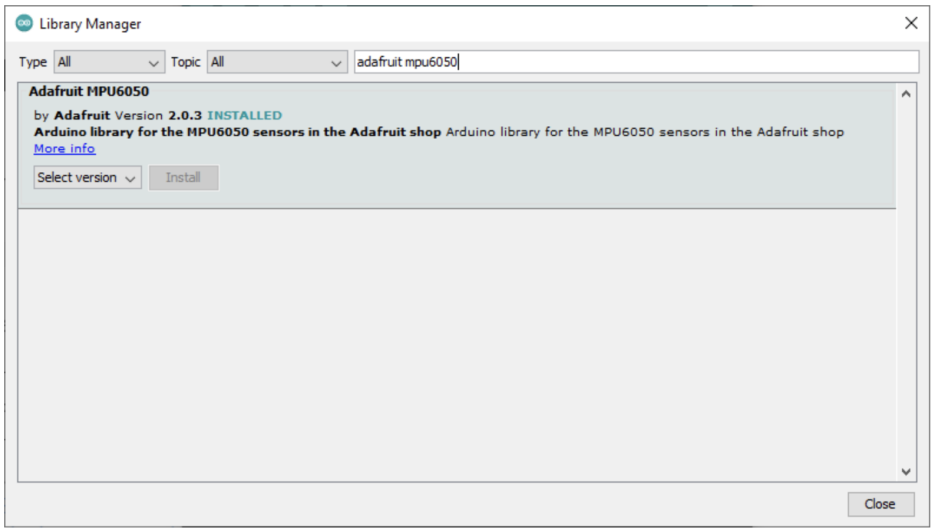

FIGURE 2 – Bibliothèque "adafruit mpu6050"

A nouveau, rechercher "Adafruit Unified Sensor". Défiler vers le bas pour trouver la bibliothèque et installer-la, voir Figure 3.

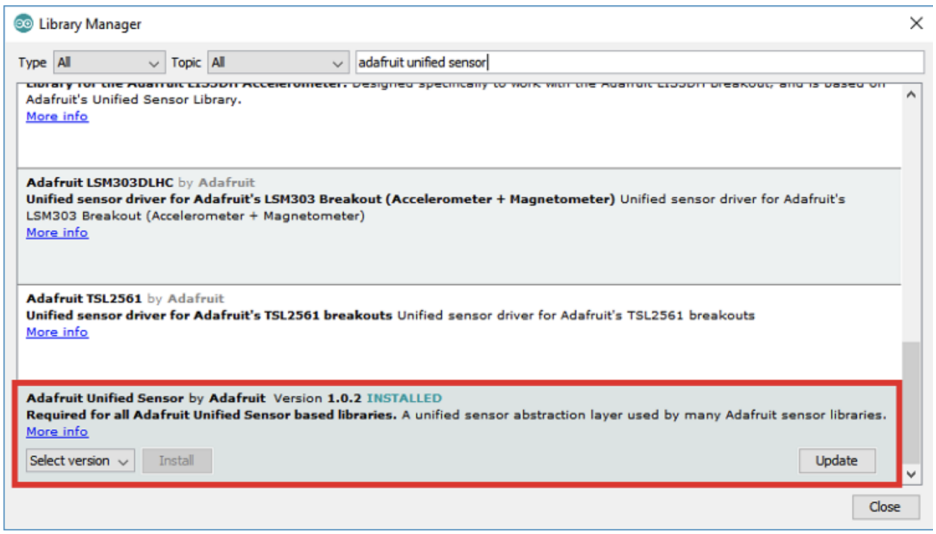

FIGURE 3 – Bibliothèque "adafruit mpu6050"

Finalement, rechercher la bibliothèque “Adafruit Bus IO” et l’installer. Cf Figure 4.

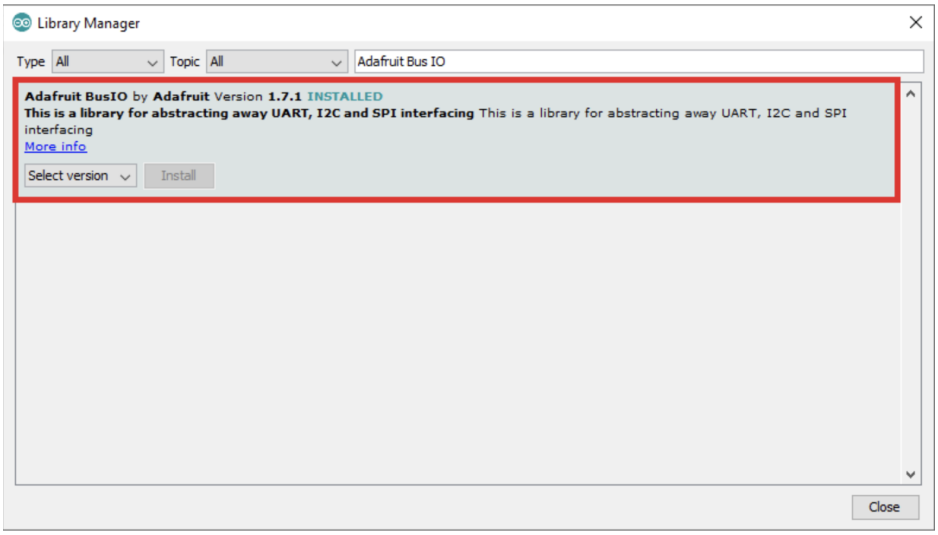

FIGURE 4 – Bibliothèque "adafruit mpu6050"

Code Arduino pour le capteur d’accélération

```
#include <Adafruit_MPU6050.h>
#include <Adafruit_Sensor.h>
#include <Wire.h>

Adafruit_MPU6050 mpu;
double t0; double t1;
int run = 1;
void setup(void) {

  Serial.begin(115200);

  while (!Serial)
    delay(10); // will pause Zero, Leonardo, etc until serial console opens

  Serial.println("Adafruit MPU6050 test!");

  // Try to initialize!
  if (!mpu.begin()) {
    Serial.println("Failed to find MPU6050 chip");
    while (1) {
      delay(10);
    }
  }
  Serial.println("MPU6050 Found!");

  mpu.setAccelerometerRange(MPU6050_RANGE_8_G);
  Serial.print("Accelerometer range set to: ");
  switch (mpu.getAccelerometerRange()) {
  case MPU6050_RANGE_2_G:
    Serial.println("+-2G");
    break;
  case MPU6050_RANGE_4_G:
    Serial.println("+-4G");
    break;
  case MPU6050_RANGE_8_G:
    Serial.println("+-8G");
    break;
  case MPU6050_RANGE_16_G:
    Serial.println("+-16G");
    break;
  }
  mpu.setGyroRange(MPU6050_RANGE_500_DEG);
  Serial.print("Gyro range set to: ");
  switch (mpu.getGyroRange()) {
  case MPU6050_RANGE_250_DEG:
```

```
    Serial.println("+- 250 deg/s");
    break;
  case MPU6050_RANGE_500_DEG:
    Serial.println("+- 500 deg/s");
    break;
  case MPU6050_RANGE_1000_DEG:
    Serial.println("+- 1000 deg/s");
    break;
  case MPU6050_RANGE_2000_DEG:
    Serial.println("+- 2000 deg/s");
    break;
  }

  mpu.setFilterBandwidth(MPU6050_BAND_5_HZ);
  Serial.print("Filter bandwidth set to: ");
  switch (mpu.getFilterBandwidth()) {
  case MPU6050_BAND_260_HZ:
    Serial.println("260 Hz");
    break;
  case MPU6050_BAND_184_HZ:
    Serial.println("184 Hz");
    break;
  case MPU6050_BAND_94_HZ:
    Serial.println("94 Hz");
    break;
  case MPU6050_BAND_44_HZ:
    Serial.println("44 Hz");
    break;
  case MPU6050_BAND_21_HZ:
    Serial.println("21 Hz");
    break;
  case MPU6050_BAND_10_HZ:
    Serial.println("10 Hz");
    break;
  case MPU6050_BAND_5_HZ:
    Serial.println("5 Hz");
    break;
  }

  Serial.println(" ");
  Serial.println("Temps (s),aX (m/s^2),aY (m/s^2),aZ (m/s^2)");

  delay(100);
}

void loop() {
  /* Get new sensor events with the readings */

  /* Print out the values */
  while (Serial.read() != '\n') {}

    t0 = millis() / 1000.0;
    run = 1;
    while (run == 1) {
    if (Serial.read() == '\n') {
      run = 0;
    }

    sensors_event_t a, g, temp;
    mpu.getEvent(&a, &g, &temp);

    t1 = millis() / 1000.0 - t0;
    Serial.print(t1);
    Serial.print(",");
    Serial.print(a.acceleration.x);
    Serial.print(",");
    Serial.print(a.acceleration.y);
    Serial.print(",");
    Serial.print(a.acceleration.z);
    Serial.println("");
    delay(500);

  }
}
```

Fiche expérience 1

Accélération dans l'ascenseur

Le but de l'expérience est de déterminer la hauteur d'un bâtiment à l'aide de notre nouvel accéléromètre.

Matériel

- L'accéléromètre conçu à l'aide d'Arduino
- Un ascenseur

Protocole

voir TP1

Résultat brut

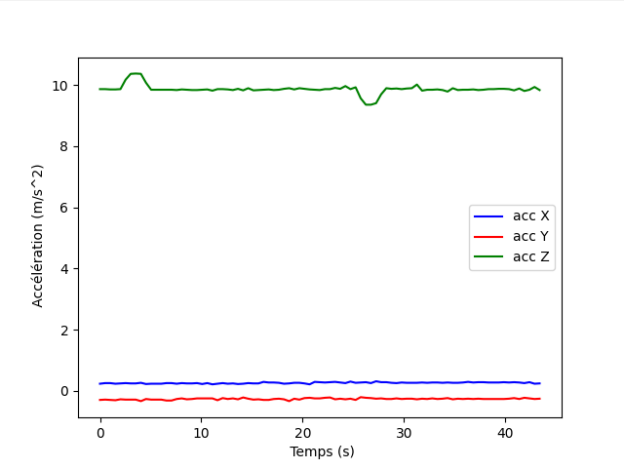

FIGURE 5 – Graphique des données brutes mesurées sur chaque axe

Analyse

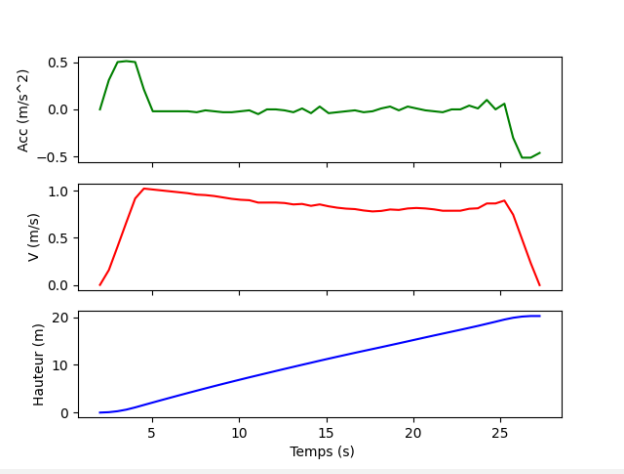

FIGURE 6 – Graphique des données analysées sur chaque axe

On observe que les données entre 2 secondes et 28 secondes correspondent à la montée. Nous allons donc extraire les données seulement comprises entre ce laps de temps. On effectue ensuite une double intégration pour avoir la vitesse verticale puis la hauteur.

Code Python

```
import matplotlib.pyplot as plt # Pour tracer des courbes
import pandas as pd             # Pour lire des fichiers
import numpy as np              # Pour le calcul numerique

data = pd.read_csv("ascenseur.txt")       # lit le fichier du format .txt contenant les donnees
data.to_csv('ascenseur.csv', index=None)  # convertit .txt au .csv

# extraction des donnees brutes
tBrut = data["Temps (s)"]
accXBrut = data["aX (m/s^2)"]
accYBrut = data["aY (m/s^2)"]
accZBrut = data["aZ (m/s^2)"]

# affichage des donnees brutes
# plt.plot(tBrut, accXBrut, "-b", label="acc X")
# plt.plot(tBrut, accYBrut, "-r", label="acc Y")
# plt.plot(tBrut, accZBrut, "-g", label="acc Z")
# plt.legend(loc="best")
# plt.xlabel(" Temps (s)")
# plt.ylabel(" Acceleration (m/s^2)")
# plt.show()

# on extrait les donnees entre les temps t = 2 et t = 28
# pour ne garder que les mesures pendant la montee

min_index = None # variable pour stocker l'indice du premier tBrut >= 2s
max_index = None # variable pour stocker l'indice du dernier tBrut <= 28s

for i, temps in enumerate(tBrut):                    # trouver min_index et max_index
    if 2 <= temps <= 28:
        if min_index is None or i < min_index:
            min_index = i
        if max_index is None or i > max_index:
            max_index = i


t = np.array(tBrut[min_index:max_index])
Az = np.array(accZBrut[min_index:max_index])    # extrait les donnees entre 2 et 28s
Az = Az - Az.mean()                             # Soustrait la valeur moyenne, donc g

N = len(Az)             # taille des tableaux
Vz = np.zeros(N)        # prepare le tableau des vitesses
Z = np.zeros(N)         # prepare le tableau des hauteurs

for i in range (1,N) :              # parcourt tout le tableau
    dt = t[i] - t[i-1]              # calcule le dt
    Vz[i] = Vz[i-1] + Az[i]*dt      # integre Az pour obtenir vitesse
    Z[i] = Z[i-1] + Vz[i]*dt        # integre la vitesse pour obtenir distance

fig, axs = plt.subplots(nrows=3, sharex = True)

axs[0].plot(t,Az, "-g", label = "Acc")
axs[0].set_ylabel('Acc (m/s^2)')

axs[1].plot(t,Vz, "-r")
axs[1].set_ylabel('V (m/s)')

axs[2].plot(t,Z, "-b")
axs[2].set_ylabel('Hauteur (m)')

plt.xlabel(" Temps (s)")
plt.show()
```

Fiche expérience 2

Mesurer l'inclinaison

Le but de l'expérience est de mesurer l'angle lorsque le capteur est incliné et immobile.

Matériel

- L'accéléromètre conçu à l'aide d'arduino
- Une pile de livre

Protocole

Placer la pile de livre devant vous. A l'aide de celle-ci, inclinez l'accéléromètre. S'assurer que l'accéléromètre est immobile. Lancez l'enregistrement des données et attendre une petite minute. Arreter l'enregistrement.

Resultat brut

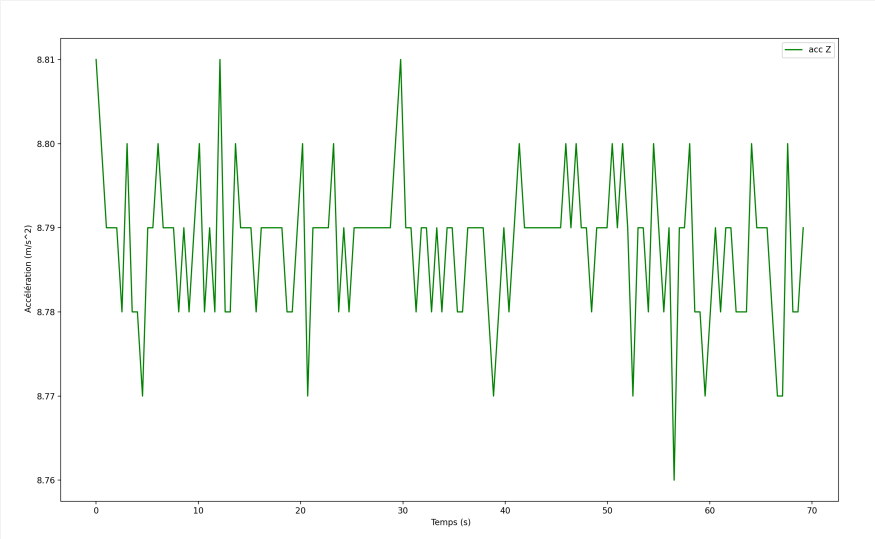

FIGURE 7 – Graphique des données brutes mesurées sur l'axe z

On observe sur ce graphique des données brutes une légère variation. Pour un résultat plus fiable, nous avons décidé de mesurer les données pendant une minute et prendre la moyenne de toutes les mesures effectuées.

Analyse

Un accéléromètre mesure une accélération, lorsqu'un objet est statique, la seule accélération possible à mesurer est la gravité. Ainsi, on peut déterminer l'angle dans lequel se trouve l'objet. Pour cela, on se place dans le repère représenté dans la figure 8

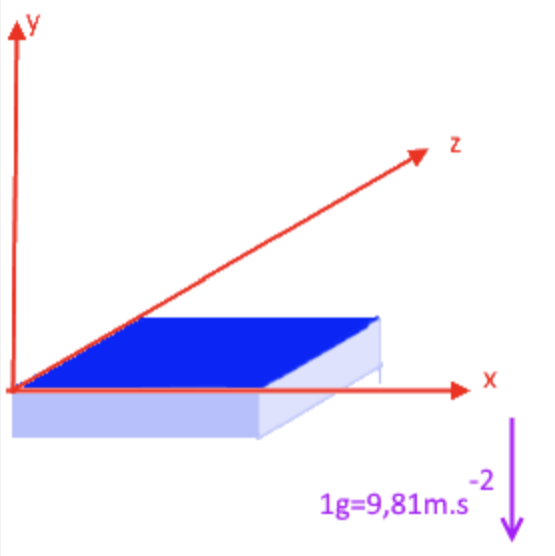

FIGURE 8 – Repere dans lequel on se place pour l'analyse

Pour simplifier le problème nous allons nous placer dans le repère du plan OxOy. On incline notre capteur et on obtient le schéma suivant.

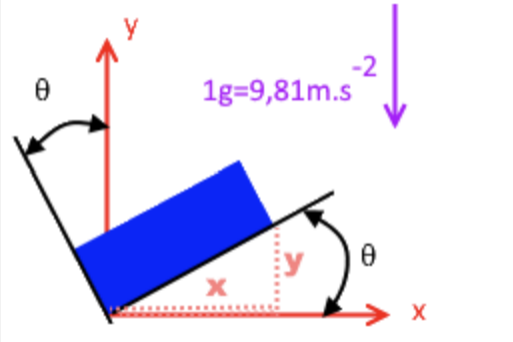

FIGURE 9 – Schéma de l'objet dans le plan OxOy

A l'aide de projection, on trouve la relation

$$\theta = \arctan(x/y)$$

On se base donc ensuite sur cette relation pour trouver l'angle de notre inclinaison. Dans notre cas, durant l'expérience le capteur se trouvait dans le plan OzOz. Donc on on mesure l'inclinaison selon z.
Dans l'espace, il faut considérer les trois axes. Pour l'angle en Z, il faudra utiliser l'accélération en Z et le module des accélérations en X et Y. Ce qui donne :

$$AngleX = \arctan(\frac{Z}{\sqrt{X^2 + Y^2}})$$

Code Python

```
import matplotlib.pyplot as plt # Pour tracer des courbes
import pandas as pd             # Pour lire des fichiers
import numpy as np              # Pour le calcul numerique

data = pd.read_csv("inclinaison.txt")       # lit le fichier du format .txt contenant les donnees
data.to_csv('inclinaison.csv', index=None)  # convertit .txt au .csv

tBrut = data["Temps (s)"]
```

```python
accXBrut = data["aX (m/s^2)"]
accYBrut = data["aY (m/s^2)"]
accZBrut = data["aZ (m/s^2)"]

x = accXBrut.mean()
y = accYBrut.mean()
z = accZBrut.mean()

angleZ = np.arctan(z/np.sqrt(x**2 + y**2))  # calcule de l'incliaison selon Z en rad

# convertit les valeurs d'angles en degrees
angleZdeg = np.degrees(angleZ)

print(f'Inclinasion selon l\'axe oZ {angleZ:.3f} rad, soit {angleZdeg:.3f} degre')

# afficher les donnees bruts
plt.plot(tBrut, accZBrut, "-g", label="acc Z")
plt.legend(loc="best")
plt.xlabel(" Temps (s)")
plt.ylabel(" Acceleration (m/s^2)")
plt.show()
```

```
In [1]: runfile('/Users/lasha/Documents/00-github/Arduino/MPU6050/gog_chen_projet/analyze_inclinaison.py', wdir='/Users/
lasha/Documents/00-github/Arduino/MPU6050/gog_chen_projet')
Inclinasion selon l'axe oZ 1.204 rad, soit 68.961 degré

In [2]:
```

FIGURE 10 – Affichage de la réponse sur la console