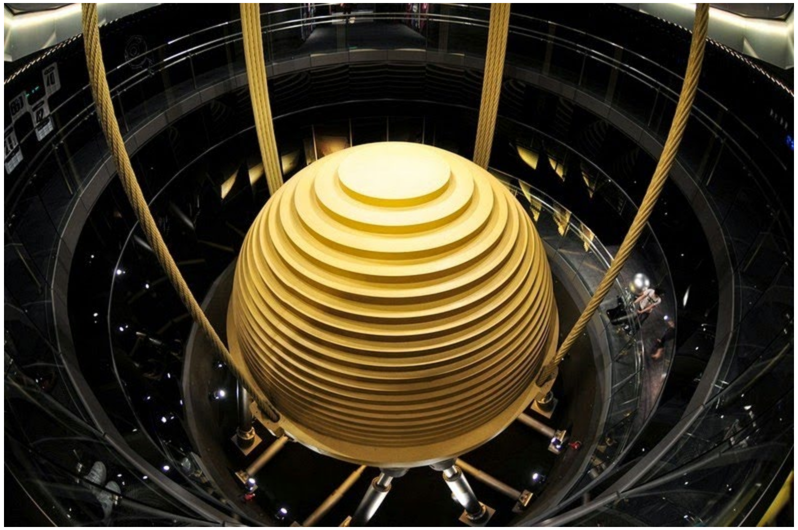
Stabilisation d'immeubles à l'aide d'amortisseurs à masse accordée

Sommaire

Un exemple: Cas de la tour Taipei 101 à Taiwan



$$h = 500m$$
$$m_{TOUR} = 7 \cdot 10^{8}\ kg$$
$$m_{TMD} = 6.6 \cdot 10^{5}\ kg$$
$$r_{TMD} = 2.7m$$

Pendule -> 4 cables d'acier de 11.5 m

Oscillations atténuées de 30 à 40%

Résiste à des bourrasques de plus de 200 km/h

Problématisation et objectifs du TIPE

Moyens de réduction des oscillations des immeubles → Variés, Domaine très vaste

↓

Recentrer le sujet sur la réduction des oscillations dues aux bourrasques

Comment réduire les oscillations des immeubles à l'aide d'amortisseurs à masse accordée?

2 axes d'analyse:

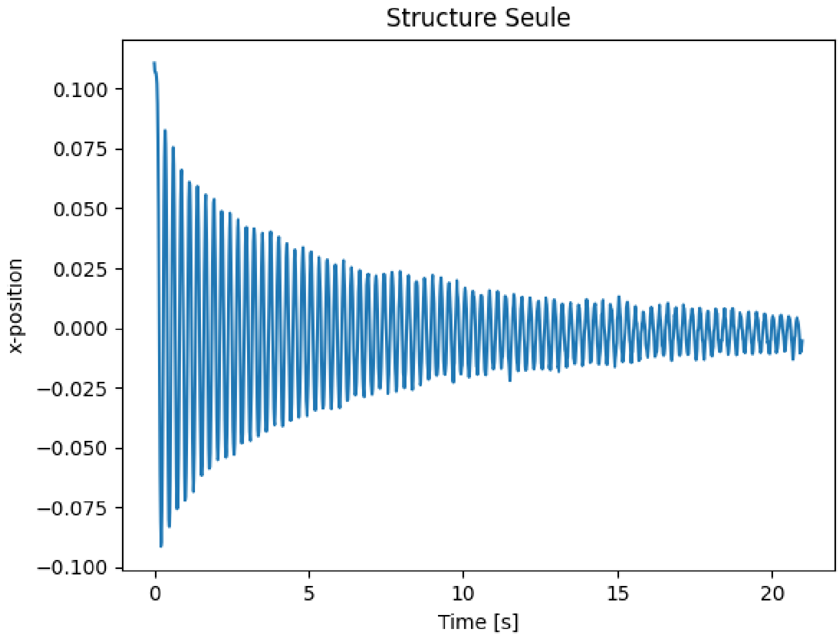
- Axe pratique avec la création d'un prototype de structure oscillante
- Axe théorique avec l'étude physique d'une structure oscillante munie d'un pendule

1. Construction de notre propre structure oscillante

Maquette

$$h = 14\ cm$$
$$L = 12.5\ cm$$
$$l = 10\ cm$$
$$m = 150\ g$$

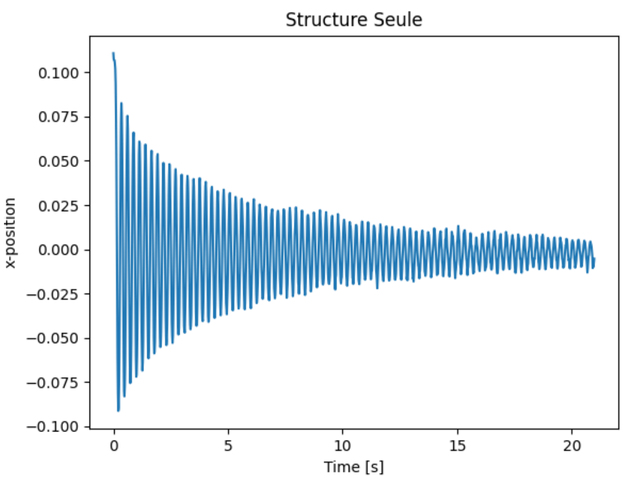
En effectuant un pointage vidéo de la structure en oscillations libre, on obtient la courbe de position suivante:



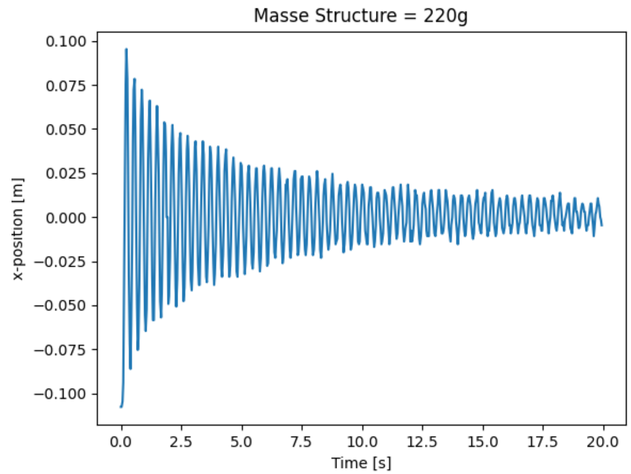
$$f_0 = 4\ Hz$$

Ajustement de la fréquence propre de la structure oscillante

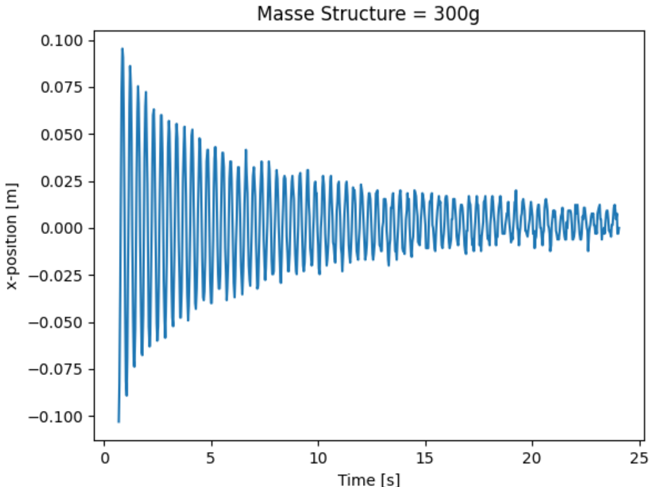
Si on ajuste la masse du plateau de la structure oscillante, on obtient des fréquences d'oscillations différentes.



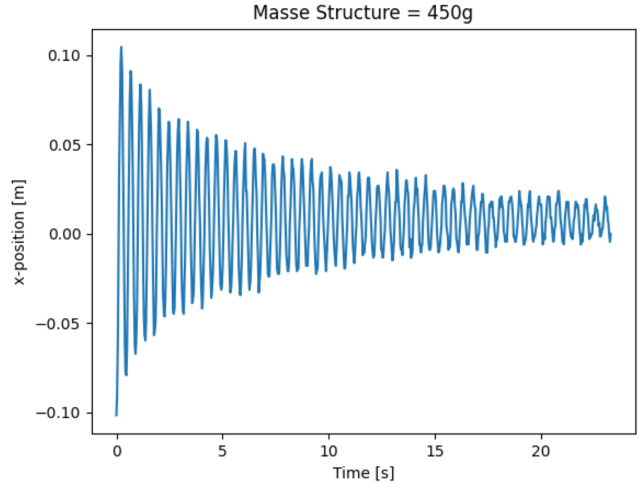
f = 4Hz



f = 3Hz



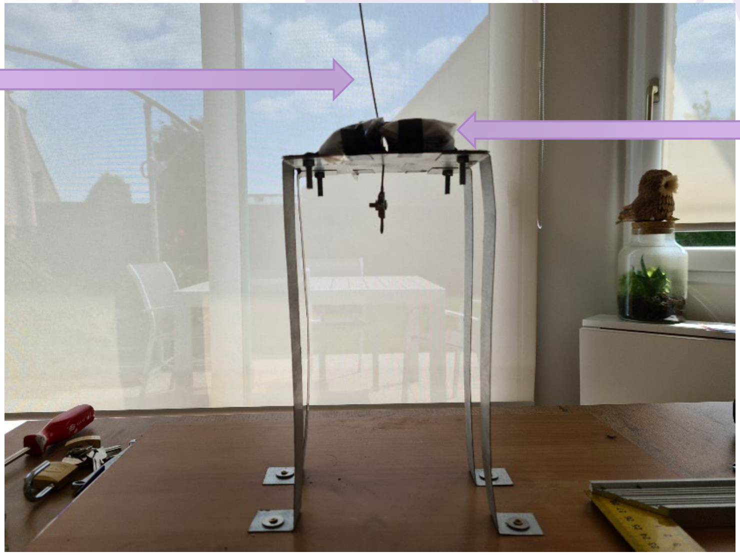
f = 2.6Hz



f = 2.2Hz

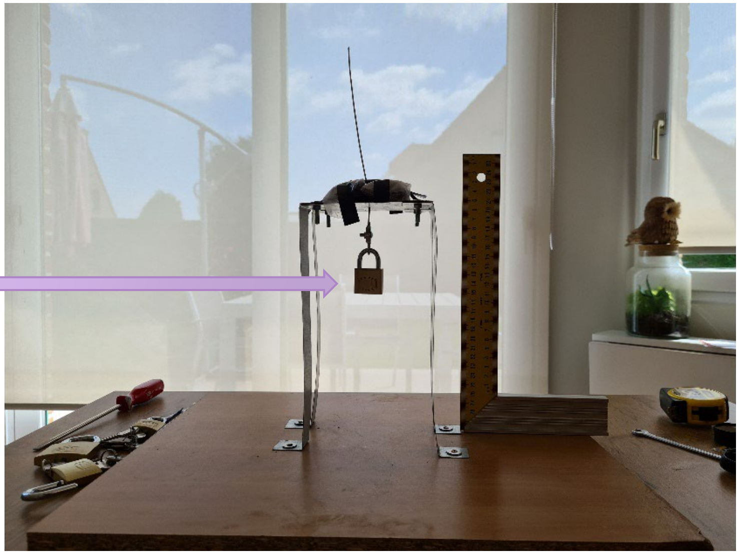
Finalisation de la maquette

Fil de fer tressé de longueur ajustable



Lestage de 300g

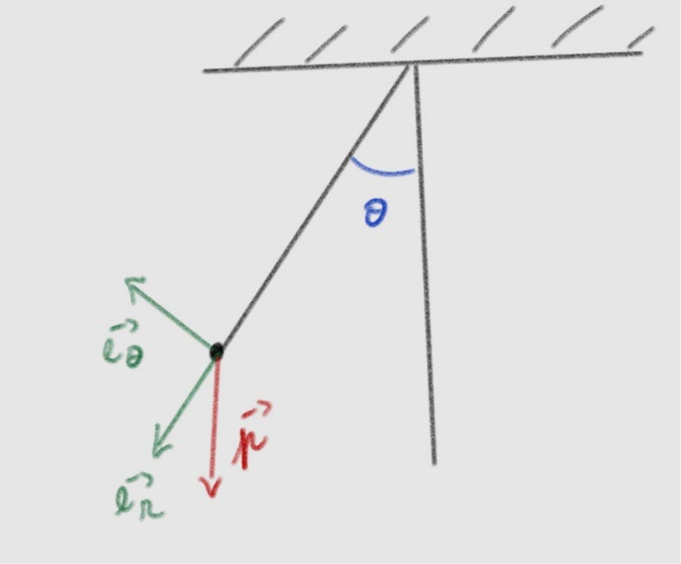
Masse de 66g



2. Calcul des paramètres idéaux du TMD

Méthode 1 – Approche naïve

On étudie ici le comportement isolé du TMD modélisé ici comme un pendule simple de masse ponctuelle, muni d'un fil de masse négligeable de longueur inextensible.
On néglige les forces de frottements



On veut que le TMD et la structure oscillante soient en résonance.
Le TMD doit donc être de même fréquence propre que la structure.

$$\omega_1 = \omega_0$$

Méthode 1 – Approche naïve

$ml\ddot{\theta} = -mg\sin(\theta)$ Par PFD projeté sur $\overrightarrow{e_\theta}$ appliqué au pendule

$\ddot{\theta} + \frac{g}{l}\theta = 0$ Par approximation des petits angles

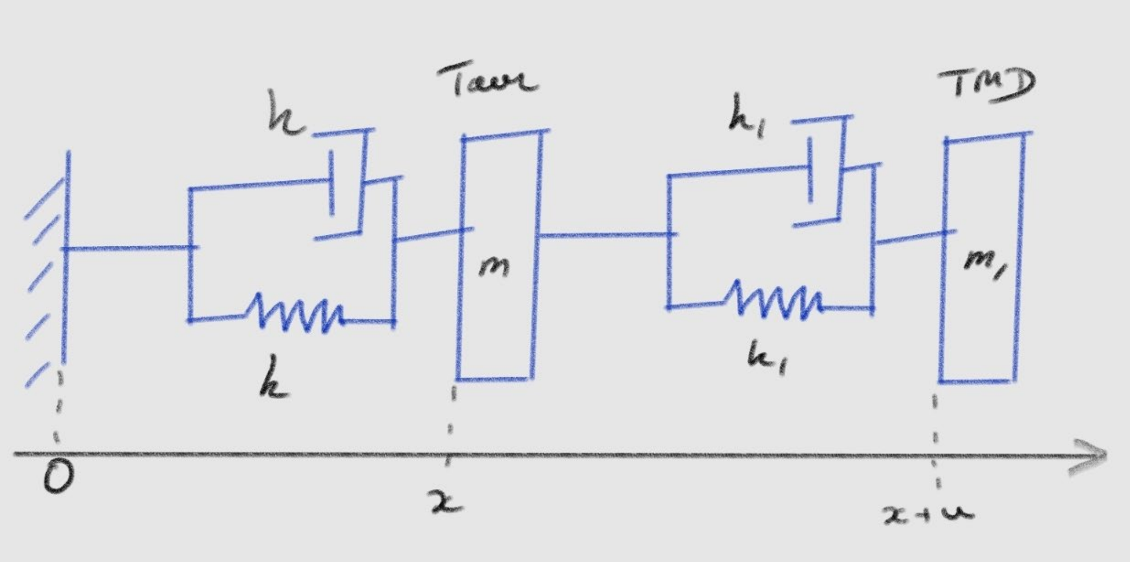
D'où $\omega_1^2 = \frac{g}{l}$

$l = \dfrac{g}{4 \times f_0^2 \times \pi^2}$ ⇒ $l = 1{,}8\ cm$ pour $f_0 = 4Hz$

On remarque qu'un pendule de 1.8 cm est compliqué à mettre en place expérimentalement. On veut donc changer la fréquence de notre structure oscillante. D'où le choix précedent de lester notre structure

Méthode 2 – Approche Précise

On modélise ici le building muni d'un TMD comme 2 ressorts en séries munis d'un coefficient de frottement fluide



<u>Bilan des forces</u>

$$\overrightarrow{K_1} = kx\overrightarrow{e_x}$$

$$\overrightarrow{K_2} = k_1(x+u)\overrightarrow{e_x}$$

$$\overrightarrow{F_1} = -h\dot{x}\overrightarrow{e_x}$$

$$\overrightarrow{F_2} = -h_1(\dot{x} + \dot{u})\overrightarrow{e_x}$$

$$\overrightarrow{F_{ie}} = -m_1\ddot{x}\overrightarrow{e_x}$$

$$\overrightarrow{f_0} = f_0(t)\overrightarrow{e_x}$$

Mise en Equation

En appliquant le PFD sur le système {Tour + TMD} puis sur le système {TMD},
On obtient le couple d'équations différentielles suivant:

$$\begin{cases} m\ddot{x} + m_1(\ddot{x} + \ddot{u}) = -kx + f_0(t) \\ m_1(\ddot{x} + \ddot{u}) = -k_1 u - h_1 \dot{u} \end{cases}$$

On peut simplifier ce système en posant:

$$\alpha = \frac{m_1}{m} \qquad \eta_1 = \frac{h_1}{2\sqrt{k_1 m_1}} \qquad \omega_0^2 = \frac{k}{m} \qquad \omega_1^2 = \frac{k_1}{m_1} \qquad a_0(t) = f_0(t)$$

$$\Rightarrow \quad \begin{matrix} (1+\alpha)\ddot{x} + \alpha\ddot{u} + \omega_0^2 = a_0(t) \\ \ddot{x} + \ddot{u} + 2\eta_1\omega_1\dot{u} + \omega_1^2 u = 0 \end{matrix}$$

Fonctions de transferts

Les équations différentielles précédentes n'étant pas solvables facilement, on passe en complexe et pose deux fonctions de transferts bien choisies

On pose: $\beta = \frac{\omega_1}{\omega_0}$ $\quad \underline{B} = -\omega^2 \underline{X}$ $\quad H_1 = \frac{\underline{U}}{\underline{X}}$ $\quad \underline{H_2} = \frac{\underline{B}}{\underline{A_0}}$

$$z = \frac{\omega}{\omega_0}$$

$$H_1(z) = \frac{z^2}{-z^2 + 2i\eta_1\beta z + \beta^2}$$

$$H_2(z) = \frac{z^2}{(1 + \alpha + \alpha H_1)z^2 - 1}$$

Etude qualitative de la réduction d'amplitude (frottements négligés)

$$\eta_1 = 0$$

$$H_1(z) = \frac{z^2}{-z^2 + \beta^2}$$

$$H_2(z) = \frac{z^2}{(1 + \alpha + H_1)z^2 - 1}$$

$$G = |H_2|$$

On veut minimiser l'accélération de la tour pour une excitation extérieure donnée.

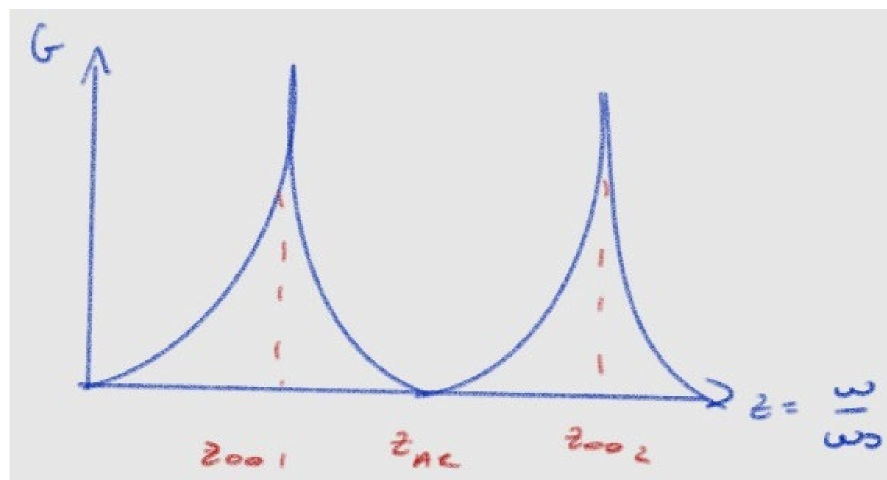
On veut donc minimiser G

$$G(z) = 0$$

$\Leftrightarrow (1 + \alpha + \alpha H_1)z^2 - 1 \rightarrow +\infty$

Or $H_1 \rightarrow +\infty \Leftrightarrow z = \beta$

D'où $\boxed{z_{AR} = \beta}$



Etude qualitative de l'énergie dissipée (frottements importants)

$\eta_1 \rightarrow +\infty$ Le TMD subi des forces de frottements infinies

$$H_1(z) \sim \frac{z}{-2i\eta_1\beta} \qquad H_2(z) \sim \frac{z^2}{z^2(1+\alpha)-1}$$

$$W = -h_1(U+X)$$

$$W = h_1 \frac{A_0}{z^2{\omega_0}^2} H_2(H_1+1)$$

$$W = \frac{h_1 A_0}{{\omega_0}^2} \frac{1}{z^2(1+\alpha)-1}\left(1 - \frac{z}{2i\eta_1\beta}\right)$$

$$|W| \rightarrow +\infty \Leftrightarrow z \rightarrow +\infty \text{ OU } z^2(1+\alpha)-1=0$$

D' où z_∞ tel qu'on dissipe le mieux l'énergie:

$$\boxed{z_\infty = \frac{1}{\sqrt{1+\alpha}}}$$

Résolution des paramètres idéaux du TMD

On se place dans la cas η_1 quelconque, on utilise les résultats précedents

$$z_{AR} = \beta$$
$$z_\infty = \frac{1}{\sqrt{1+\alpha}}$$

On désire garder le meilleur des deux analyses précedentes:
- Amplitude des oscillations minimisée
- Dissipation de l'énergie maximisée

On pose donc $z = z_{AR} = z_\infty$

D'où le choix de poser $\beta = \frac{1}{\sqrt{1+\alpha}}$

$$\frac{\omega_1}{\omega_0} = \frac{1}{\sqrt{1+\frac{m_1}{m}}}$$

$$\frac{g}{l} = \frac{4\pi^2 {f_0}^2}{1+\frac{m_1}{m}}$$

Résultats

A l'aide des 2 méthodes étudiées on a déduit des valeurs de la longueur du pendule l pour une masse de celui-ci fixée à $66g$

METHODE 1

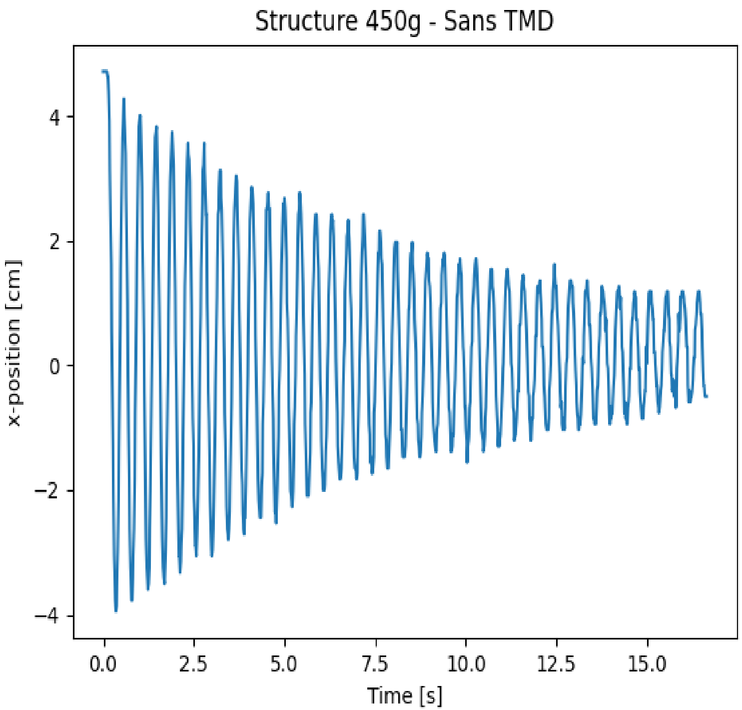
$$l = \frac{g}{4\pi^2 {f_0}^2}$$

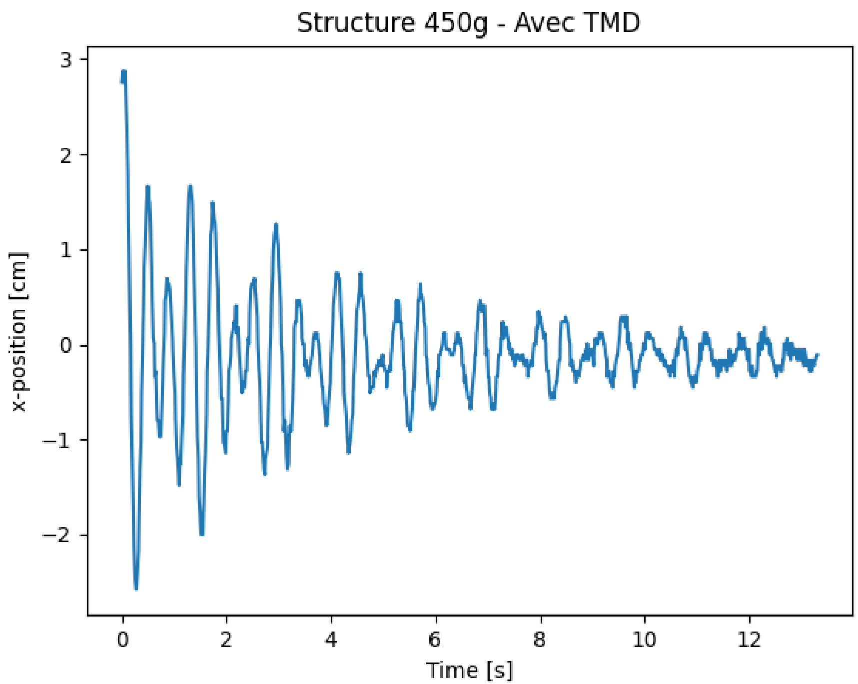
METHODE 2

$$l = \frac{g(1 + \frac{m_1}{m})}{4\pi^2 {f_0}^2}$$

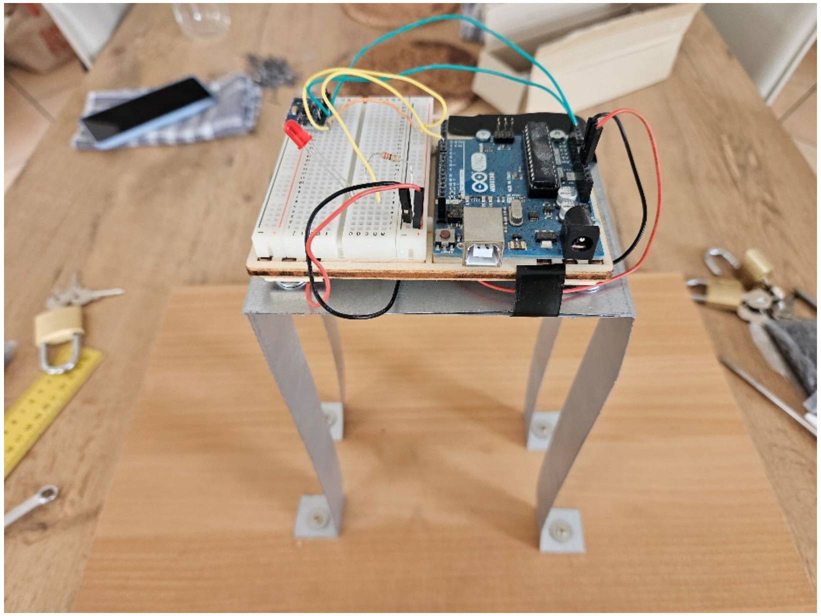
Masse Structure (g)	150	220	300	450
Fréquence propre tour (Hz)	4.0	3.0	2.6	2.2
Longueur Pendule (cm) Méthode 1	1.58	2.81	4.05	5.23
Longueur Pendule (cm) Méthode 2	2.28	3.66	4.94	6.00

Vérification expérimentale à l'aide de la maquette





3. Automatisation des mesures et du calcul à l'aide d'un Arduino

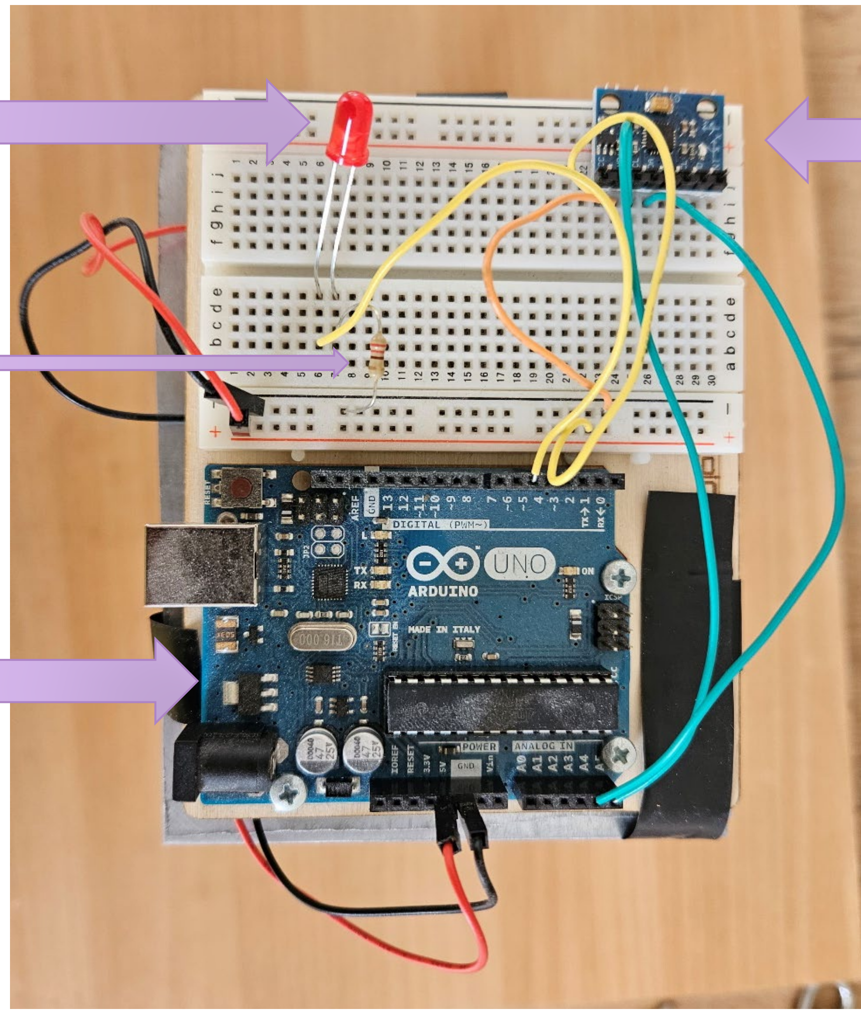


Maquette munie de l'Arduino

LED

Résistance

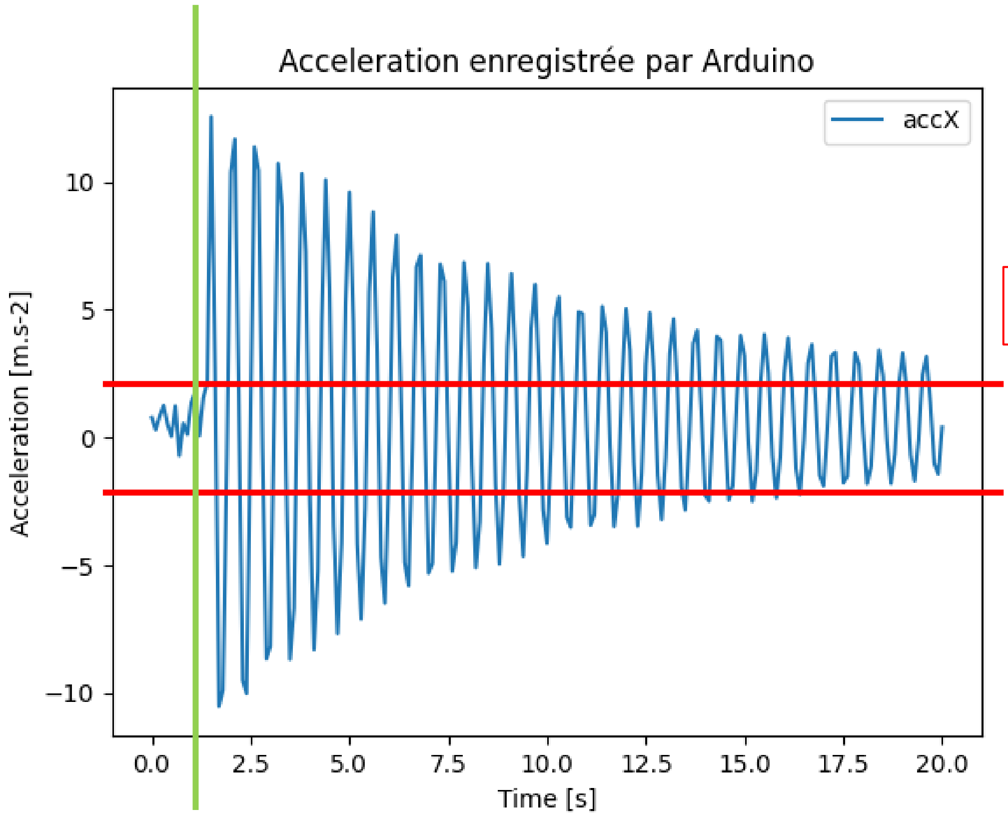
Carte Arduino UNO



Accéléromètre

$$m_a = 183g$$

Résultats



$$m_{TOUR} = m_{plateau} + m_{Arduino}$$
$$m_{TOUR} = 333g$$

Seuil défini

Début du suivi des changements de signes

Nbr changements de signes : 71

Temps mesuré: 18.6s

Fréquence calculée: 1.9Hz

Longueur optimale calculée: 8.7cm

Axes d’approfondissement

- Considération d’autres forces perturbatrices comme les ondes sismiques
  (-> expérimentalement, nécessiterait une table vibrante)

- Améliorer l’algorithme permettant de calculer la période du signal reçu depuis l’Arduino

Annexe

Principe de l'algorithme embarqué

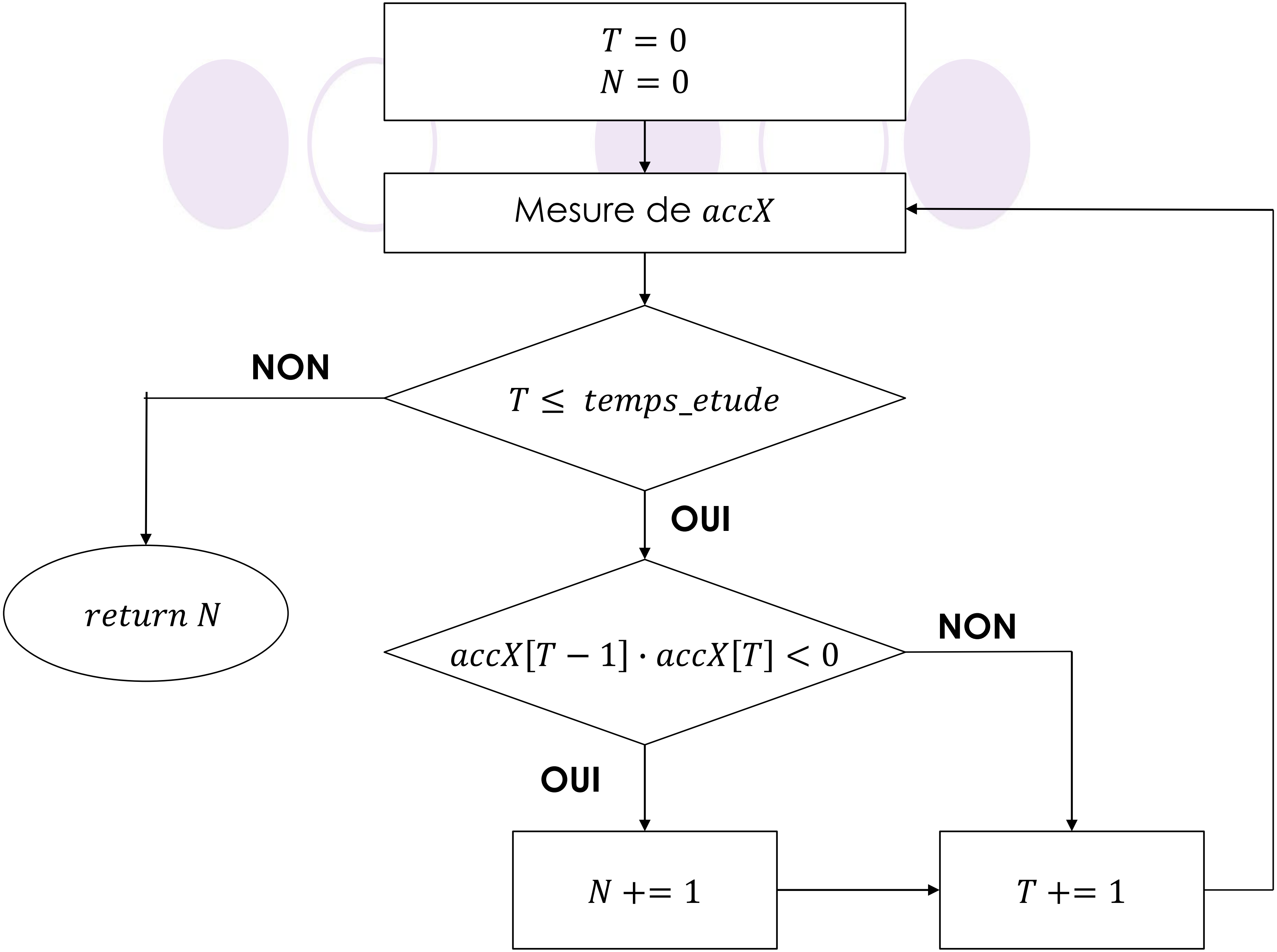
On fait osciller la structure munie de l'Arduino à sa fréquence propre

↓

L'Arduino mesure l'acceleration en temps réel et compte le nombre de changements de signes pendant un temps d'étude donné

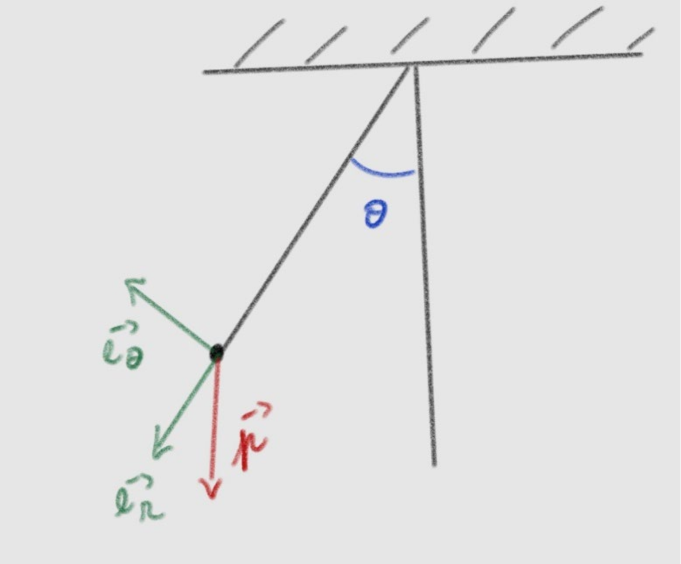
↓

A partir de la formule déterminée précédemment, l'Arduino calcule la longueur idéale du pendule



Algorigramme du code embarqué dans l'Arduino

Calcul pulsation propre pendule



Bilan des forces

$$\vec{p} = mg(\cos(\theta)\,\overrightarrow{e_r} - \sin(\theta)\,\overrightarrow{e_\theta})$$

$$Par\ PFD\ projeté\ sur\ \overrightarrow{e_\theta}\ \text{on obtient} \qquad \ddot{\theta} + \frac{g}{l}\sin\theta = 0$$

$$Par\ approximation\ des\ petits\ angles \qquad \ddot{\theta} + \frac{g}{l}\theta = 0$$

$$On\ pose\ \omega_1 = \sqrt{\frac{g}{l}}$$

$$En\ résolvant\ l'équation\ différentielle \qquad \theta = A\cos\omega_1 t + B\sin\omega_1 t$$

$$Or\ \theta(0) = A = \theta_0 \quad ; \quad \dot{\theta}(0) = \omega_1 B = \dot{\theta}_0$$

$$\theta(t) = \theta_0\cos\omega_1 t + \frac{\dot{\theta}_0}{\omega_1}\sin\omega_1 t$$

$$\omega_1 = \sqrt{\frac{g}{l}}$$

Code Arduino - 1

```
//Libraries
#include <Wire.h>// Bibliothèque standart qui permet l'interaction avec les élements de l'Arduino -
https://www.arduino.cc/en/reference/wire
#include <Adafruit_MPU6050.h>// Bibliothèque permettant l'interaction avec notre acceleromètre, le MPU-6050 -
https://github.com/adafruit/Adafruit_MPU6050
#include <Adafruit_Sensor.h>//https://github.com/adafruit/Adafruit_Sensor


// Définition des constantes initiales mutables
float accX = 0;
bool key = false;
int nbr = 0;
int unite_time = 0;
float seuil = 2.0;


//Objects
Adafruit_MPU6050 mpu;
void setup() {

  //Initialisation du MPU-6050
  Serial.begin(9600);
  Serial.println(F("Initialize System"));
  if (!mpu.begin(0x68)) { // Change address if needed
    Serial.println("Failed to find MPU6050 chip");
    while (1) {
     delay(10);
   }
  }
  mpu.setAccelerometerRange(MPU6050_RANGE_16_G);
  mpu.setGyroRange(MPU6050_RANGE_250_DEG);
  mpu.setFilterBandwidth(MPU6050_BAND_21_HZ);
}
```

Code Arduino - 2

```
void loop() {

  readMPU();

  // Initialisation des constantes du système
  const float m_structure = 240;
  const float m_TMD = 66;
  const float pi = 3.14;
  const float g = 9.81;

 // Calcul de la période puis de la longueur idéale du pendule effectué une fois que l'on a mesuré l'acceleration
pendant 20 sec
 if (unite_time > 200){
   digitalWrite(5, HIGH);
   float N = nbr;
   float T = 2*20/N;
   float omega1_carre = 4*pi*pi/(T*T*(1 + m_TMD/m_structure));
   float longueur = g/omega1_carre;
   Serial.print("Nbr Chgmt Signes = ");
   Serial.print(nbr);
   Serial.print("\nPeriode =");
   Serial.print(T);
   Serial.print("\nLongueur Pendule = ");
   Serial.print(longueur);
   while(1);
 }

  // Permet de régler la fréquence d'échantillonage fe = 1/(delay/1000)
  delay(50); // décalage en ms entre deux mesures
}
```

Code Arduino - 3

```
void readMPU( ) { /* function readMPU */

   // Read accelerometer data
   sensors_event_t a, g, temp;
   mpu.getEvent(&a, &g, &temp);
   float accX2 = a.acceleration.x;
   float accY = a.acceleration.y;
   float accZ = a.acceleration.z;
   float norme_acc = sqrt(accX2*accX2 + accY*accY + accZ*accZ);

   // Print accelerometer data -> data reçu ensuite par PuTTy
   Serial.print(unite_time/10.0);
   Serial.print(";");
   Serial.print(accX2);
   Serial.print(";");
   Serial.print(accY);
   Serial.print(";");
   Serial.print(accZ);
   Serial.print(";");
   Serial.print(norme_acc);
   Serial.print("\n");

  if (norme_acc > seuil){
    key = true;
  }

  // Permet de compter le nombre de changements de signes et donc ensuite la période
  if (key = true){
    if (accX2 * accX < 0){
      nbr++;
    }
    unite_time++;
    accX = accX2;
  }
}
```

Code Python – Pymeca_to_graph - 1

```python
import matplotlib.pyplot as plt
import csv

# Nom fichier csv à étudier et titre de la courbe qui sera obtenue
fichier = 'data/avec tmd.csv'
nom_courbe = 'Structure 450g - Avec TMD'


# Formatage des données du fichier csv pour que celles-ci soient utilsables par python
def replace(j):
    ans = ""
    if j == "":
        return 0
    for i in j:
        if i != ',':
            ans += i
        else:
            ans += '.'
    print(type(ans))
    return float(ans)
```

Code Python – Pymeca_to_graph - 2

```python
# Ouverture du fichier csv qui pointe la position de la Structure en fonction du temps
with open(fichier, newline='') as f1:
    reader = csv.reader(f1, dialect='excel-tab')
    T1 = []
    X1 = []
    for row in reader:
        T1.append(replace(row[0]))
        X1.append(100*(replace(row[1])))


# Création Courbe x(t)
plt.plot(T1, X1)
plt.xlabel('Time [s]')
plt.ylabel('x-position [cm]')
plt.title(nom_courbe)
plt.show()
```

Code Python – Arduino_to_graph - 1

```python
import matplotlib.pyplot as plt
import csv

# Nom fichier csv à étudier et titre de la courbe qui sera obtenue
fichier = 'data/putty_exp_nn.csv'
nom_courbe = 'Acceleration enregistrée par Arduino'

# Formatage des données du fichier csv obtenu depuis l'arduino pour que celles-ci soient exploitables par python
def refactor(row):
    answer = []
    sous_chaine = ""
    for i in row[0]:
        if i == ";":
            answer.append(float(sous_chaine))
            sous_chaine = ""
        else:
            sous_chaine += i
    return answer
```

Code Python – Arduino_to_graph - 2

```python
# Ouverture du fichier csv qui pointe l'acceleration de la structure munie de l'aruino en fonction du temps
with open(fichier, newline='') as f1:
    reader = csv.reader(f1, dialect='excel-tab')
    finish = False
    start = 0
    T = []
    X = []
    for row in reader:
        if row[0][0] == "N" or finish:
            finish = True
            print(row[0])
        elif start <= 1:
            start += 1
        else:
            new_row = refactor(row)
            T.append(new_row[0])
            X.append(new_row[1])


# Création Courbe x(t)
plt.plot(T, X, label="accX")
plt.xlabel('Time [s]')
plt.ylabel('Acceleration [m.s-2]')
plt.legend()
plt.title(nom_courbe)
plt.show()
```