La stabilisation de caméra

Dans quelles mesures l’analyse d’un système de stabilisation mécanique asservie permet-elle d’améliorer la stabilité du système ?

Sommaire:

I. Présentation et modélisation

1. Le système d'étude
2. Modélisation du stabilisateur
3. Etude du servomoteur

II. Stabilisation

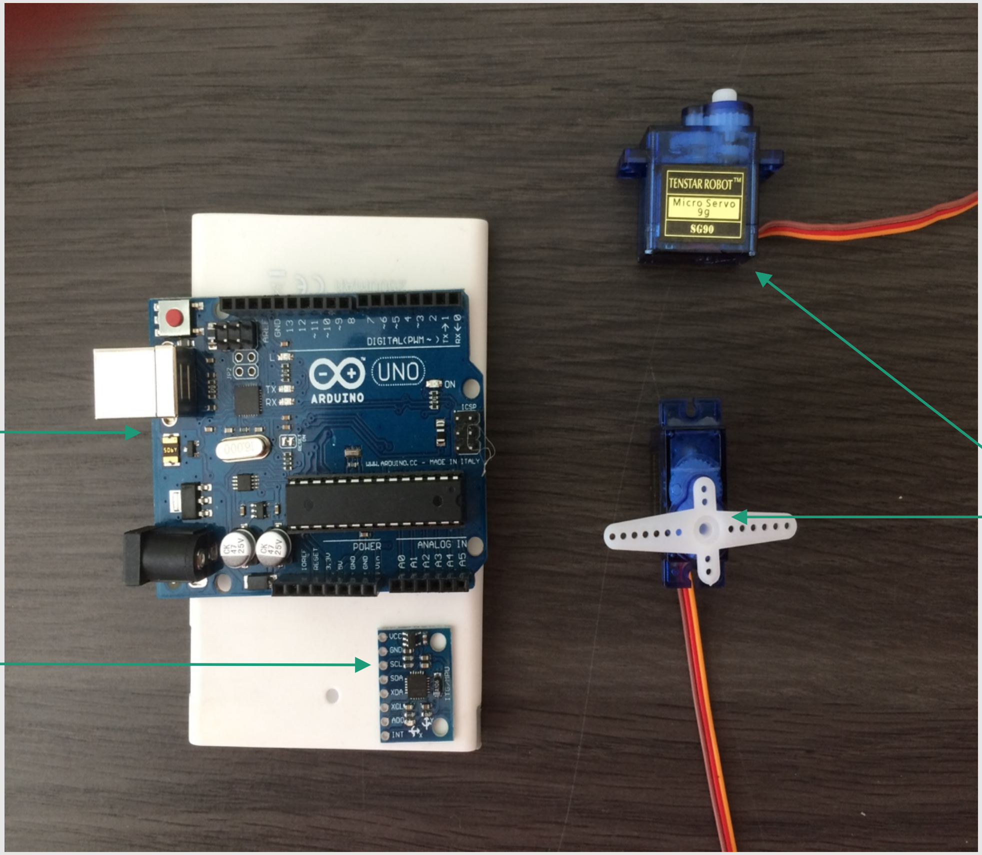
1. Le référentiel galiléen
2. Mise en œuvre algorithmique et analyse des données
3. Expérience et prévision en référentiel non galiléen

III. Améliorations

1. Atténuation du bruit
2. Amélioration en référentiel non galiléen

I. Présentation et modélisation

1. Le système d’étude

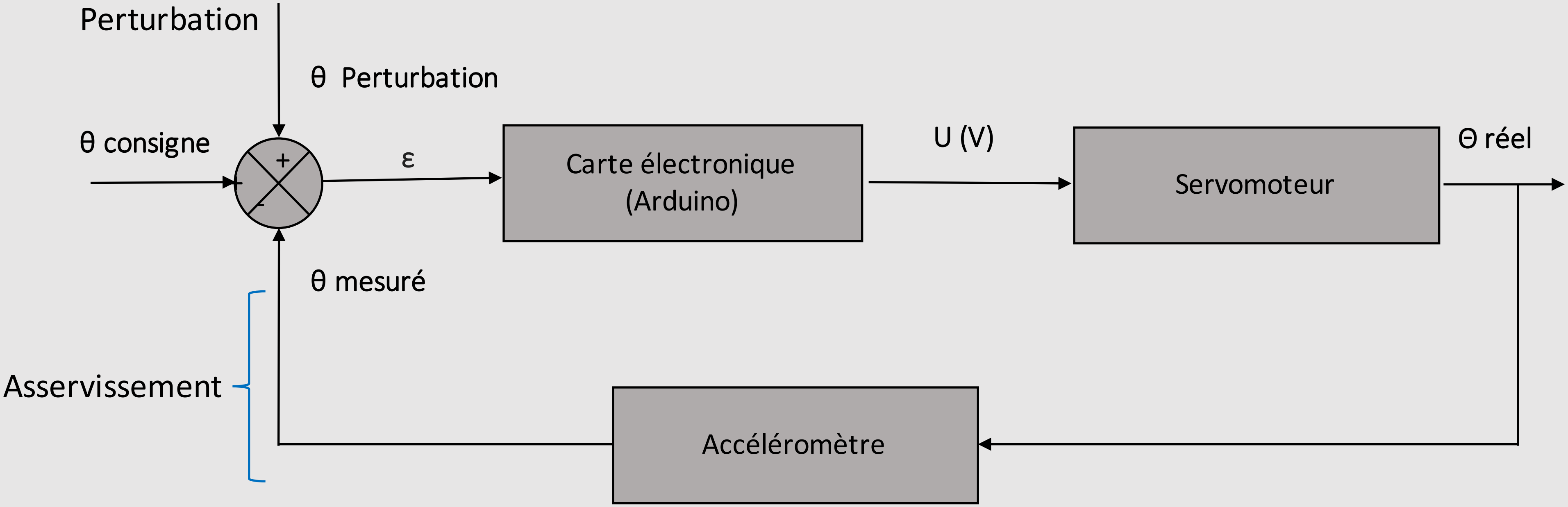

UNO
ARDUINO
TENSTAR ROBOT™
Micro Servo 9g
SG90

Carte Arduino Uno

Accéléromètre
MPU6050

Servomoteur SG90

2. Modélisation du stabilisateur



Perturbation

θ Perturbation

θ consigne

ε

Carte électronique (Arduino)

U (V)

Servomoteur

Θ réel

θ mesuré

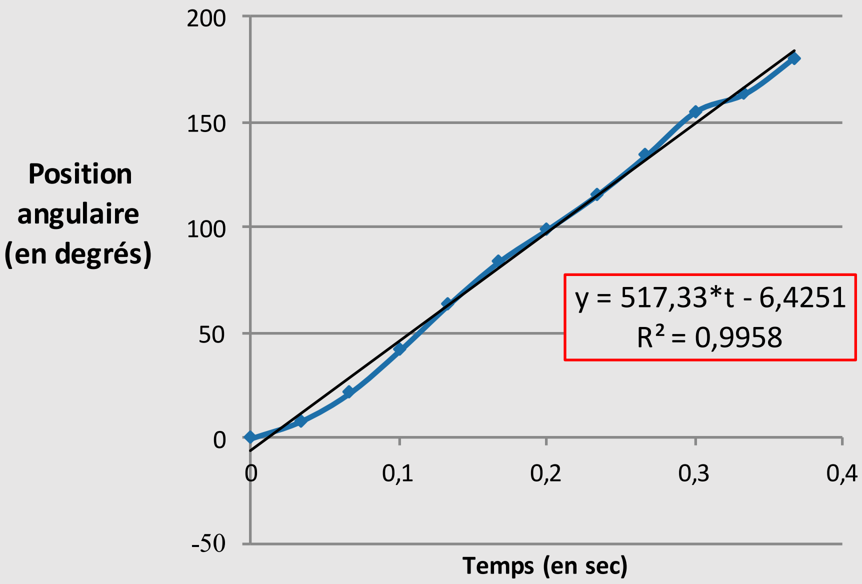
Asservissement

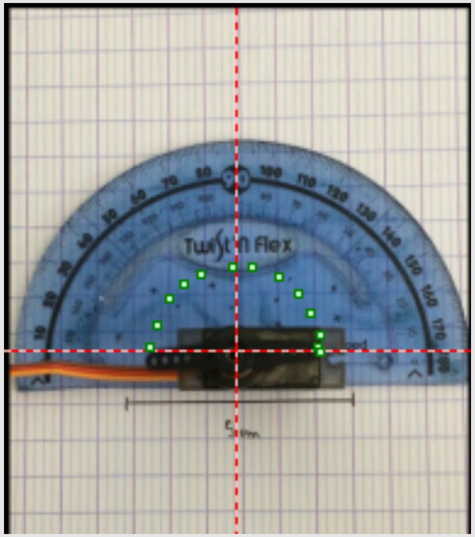
Accéléromètre

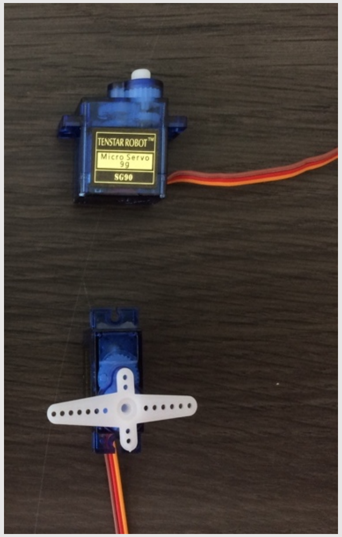
But: θ mesuré = θ perturbation → ε = θ consigne

3. Etude du servomoteur

Réponse du servomoteur à un échelon de 180 degrés




Twist'n Flex
5cm

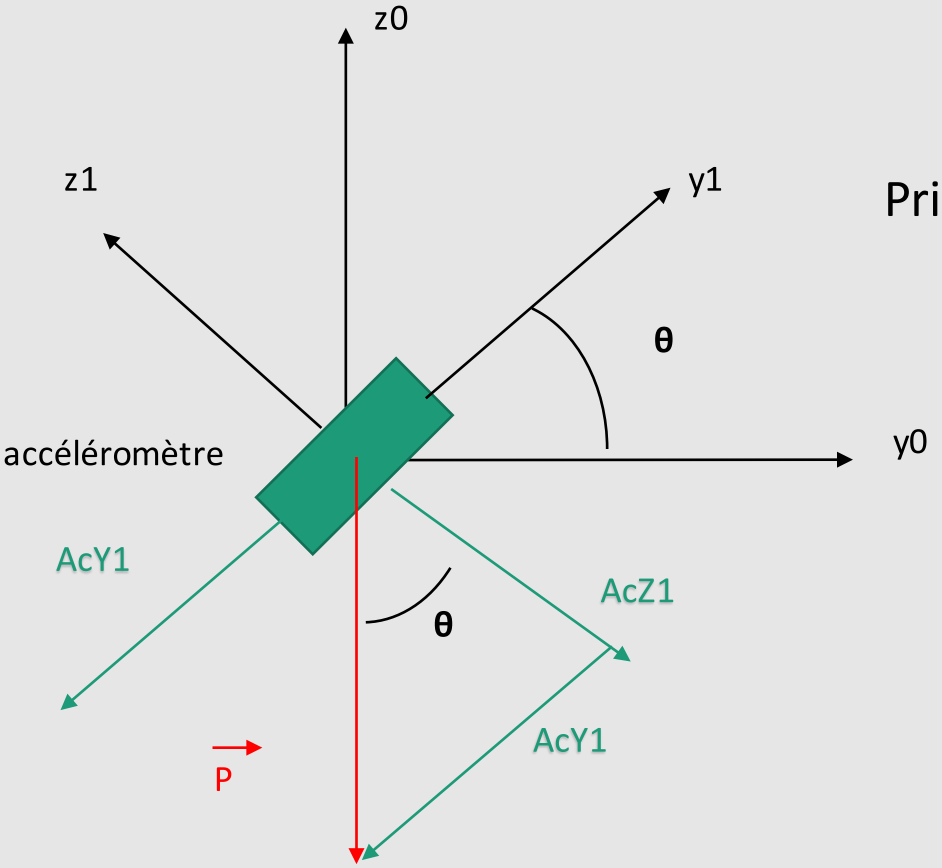

TENSTAR ROBOT
Micro Servo 9g
SG90

Temps de réponse à 5% =(0,33 ± 0,01) s

Vitesse de rotation moyenne = (8,8 ± 0,8) rad.s^{-1}

II. Stabilisation

1. Le référentiel galiléen


z0
z1
y1
θ
y0
accéléromètre
AcY1
AcZ1
θ
AcY1
$\vec{P}$

Principe fondamental de la dynamique :

$$\boxed{\theta = \tan^{-1}\frac{-AcY1}{AcZ1}}$$

2. Mise en œuvre algorithmique et analyse des données

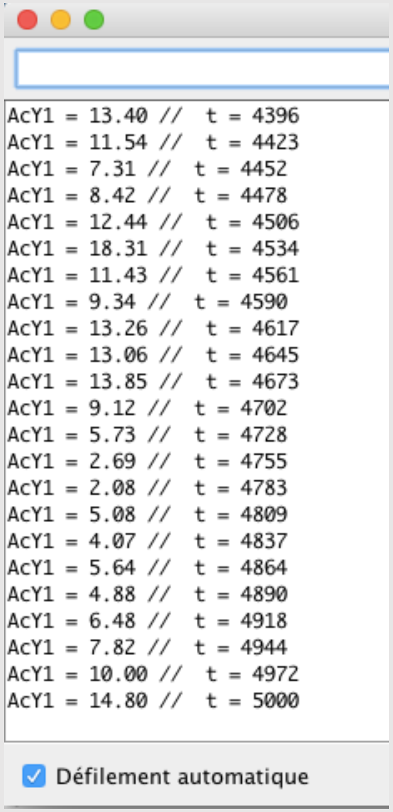
1
- Lien avec l'accéléromètre

2
- Lecture des accélérations
- Transformation en angle

3
- Demande de mouvement du servomoteur

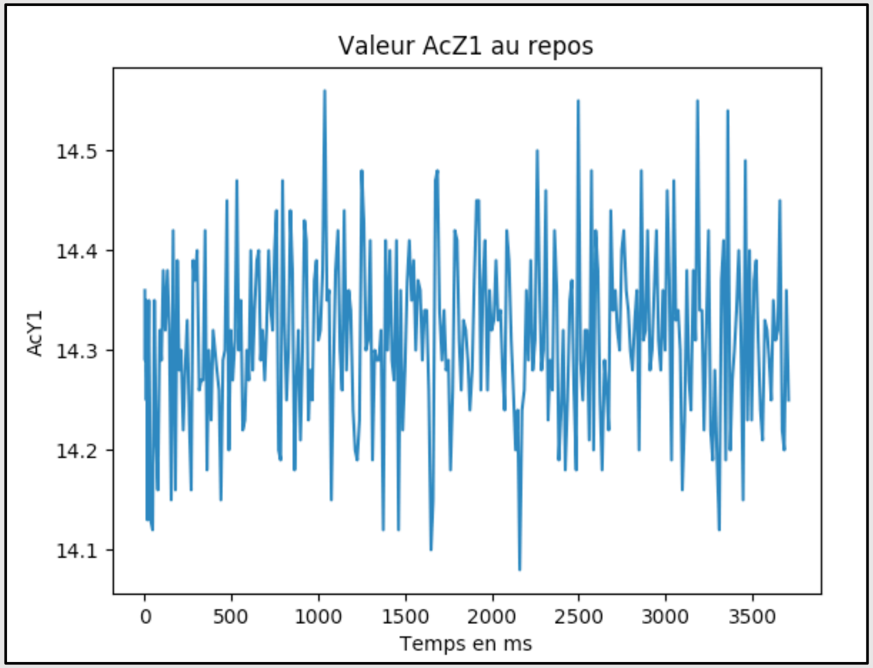
4
- Lien Arduino-Python

5
- Analyse des données



```
AcY1 = 13.40 //  t = 4396
AcY1 = 11.54 //  t = 4423
AcY1 = 7.31 //  t = 4452
AcY1 = 8.42 //  t = 4478
AcY1 = 12.44 //  t = 4506
AcY1 = 18.31 //  t = 4534
AcY1 = 11.43 //  t = 4561
AcY1 = 9.34 //  t = 4590
AcY1 = 13.26 //  t = 4617
AcY1 = 13.06 //  t = 4645
AcY1 = 13.85 //  t = 4673
AcY1 = 9.12 //  t = 4702
AcY1 = 5.73 //  t = 4728
AcY1 = 2.69 //  t = 4755
AcY1 = 2.08 //  t = 4783
AcY1 = 5.08 //  t = 4809
AcY1 = 4.07 //  t = 4837
AcY1 = 5.64 //  t = 4864
AcY1 = 4.88 //  t = 4890
AcY1 = 6.48 //  t = 4918
AcY1 = 7.82 //  t = 4944
AcY1 = 10.00 //  t = 4972
AcY1 = 14.80 //  t = 5000
```

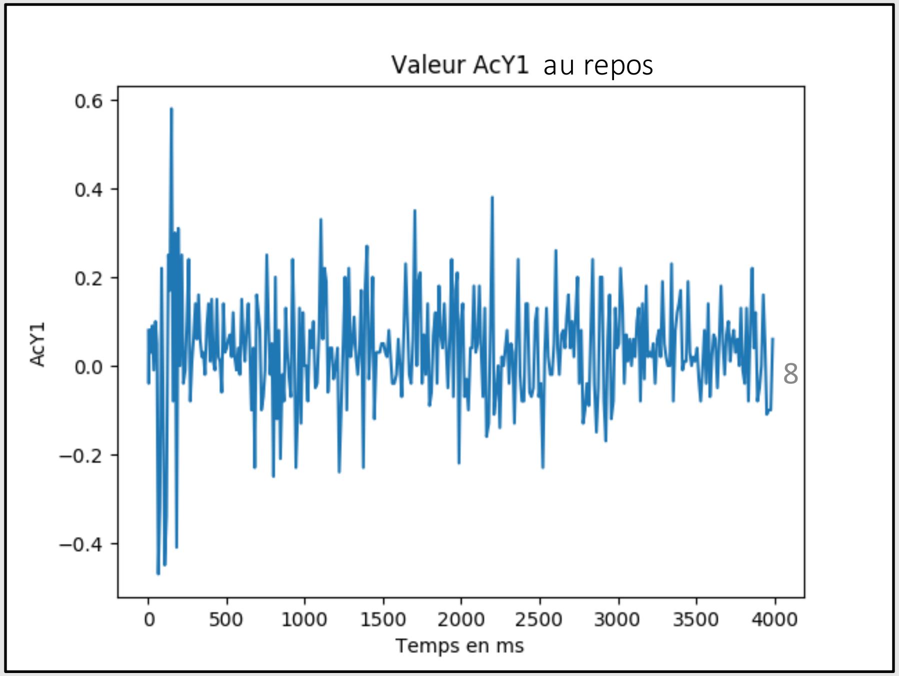
☑ Défilement automatique



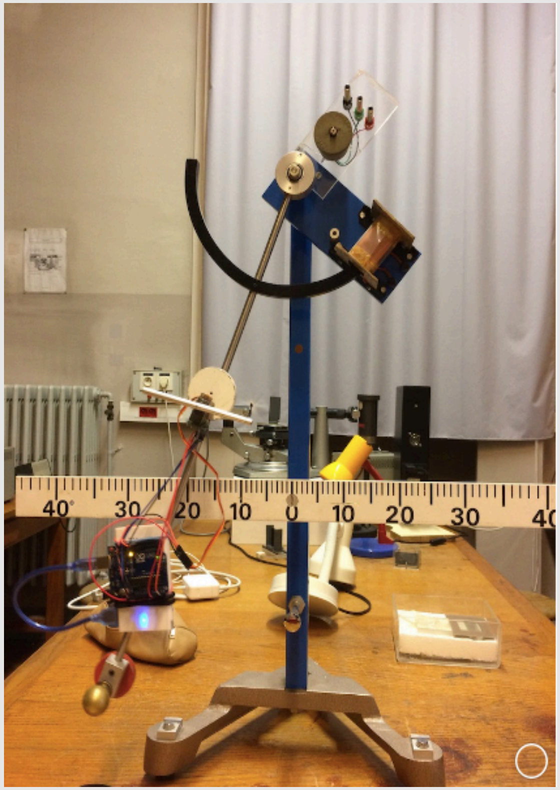
Accélération au repos

AcZ1 ≠ g au repos

Période d’échantillonnage: 10 ms

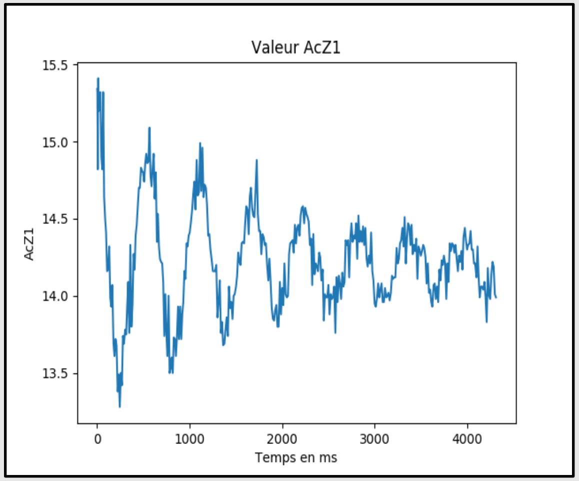


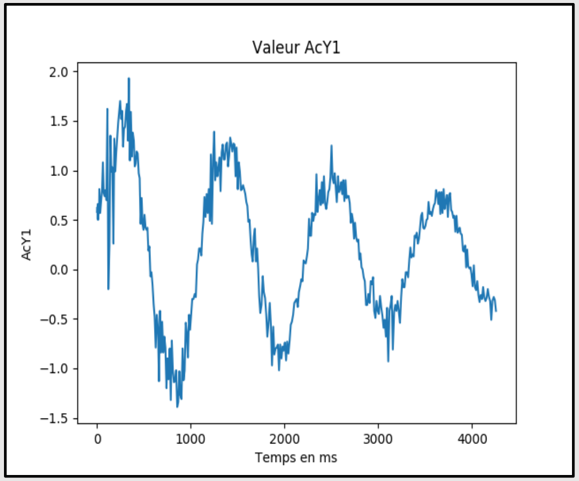
Bruit



Conditions initiales :

$$\begin{cases} \theta\ (t=0) = 24.5\ \pm 0.5\, deg \\ \dot{\theta}(t=0) = 0\ \mathrm{deg.}\, s^{-1} \\ l = 21\ cm \end{cases}$$





Analogie Période:

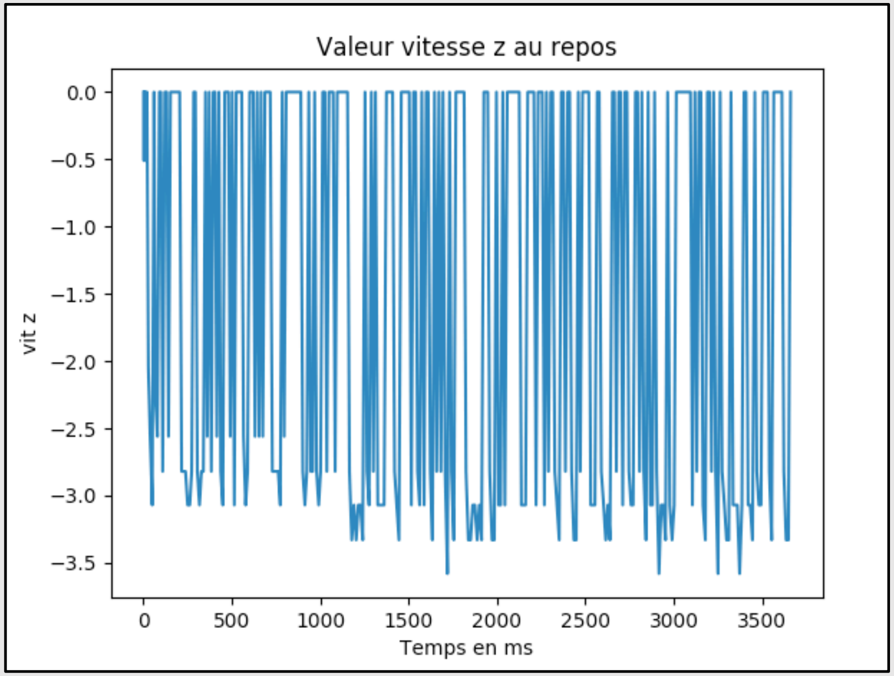
$$T_{exp} = 1200 \pm 50\ ms$$

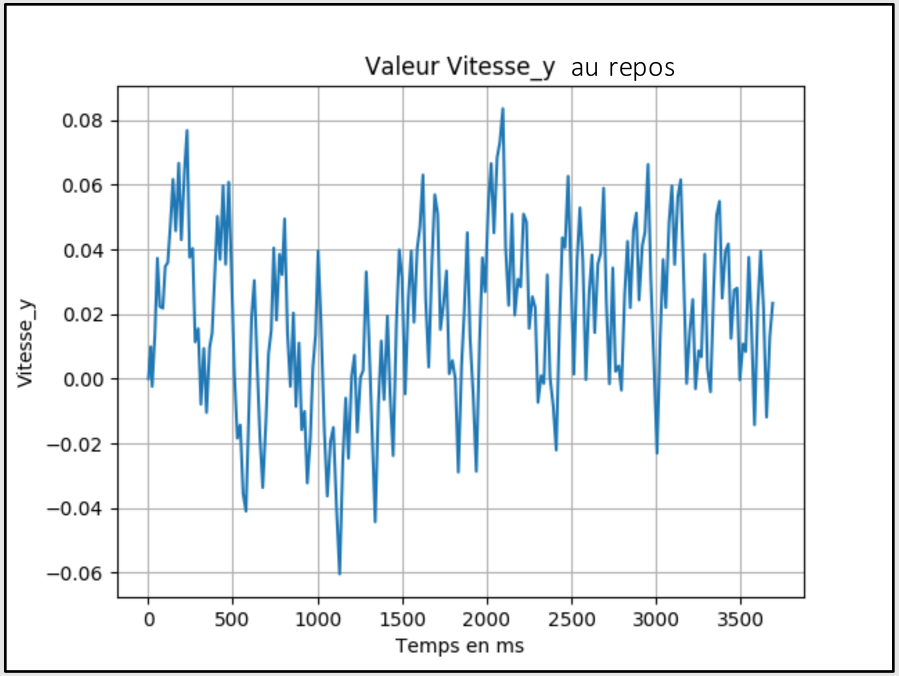
$$T_{\ theorique} = \frac{2\pi}{\omega_0} = 2\pi\sqrt{\frac{l}{g}} = 910\ ms$$

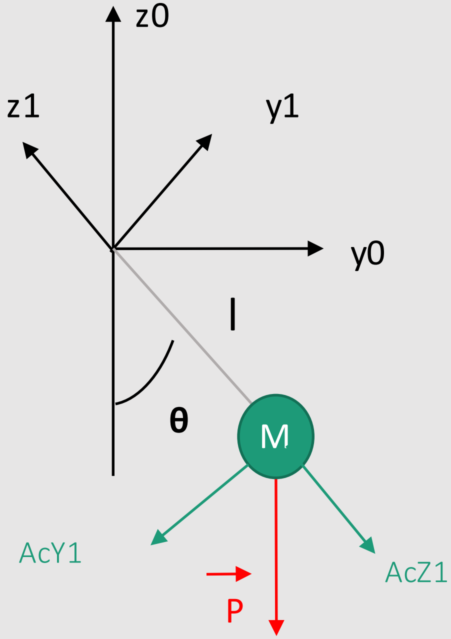
$$\frac{T_{\ théorique}}{T_{exp}} = 76\ \pm 2\ \ \%$$

Intégration par la méthode des rectangles:

$$vitesse(t) = \sum_{ti \in [t, t+\Delta t]} acc(ti) * \Delta t$$ avec Δt la durée d'acquisition $\sim 10\ ms$

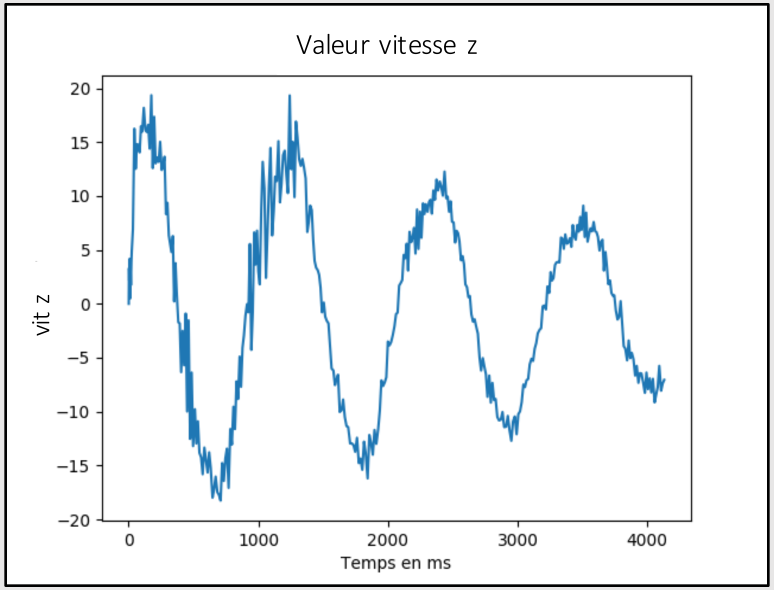


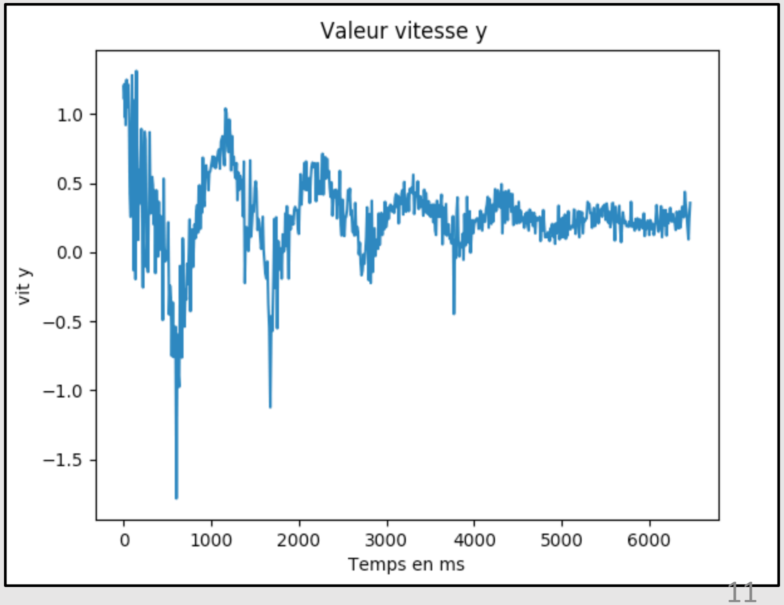



z0
z1
y1
y0
l
θ
M
AcY1
AcZ1
P

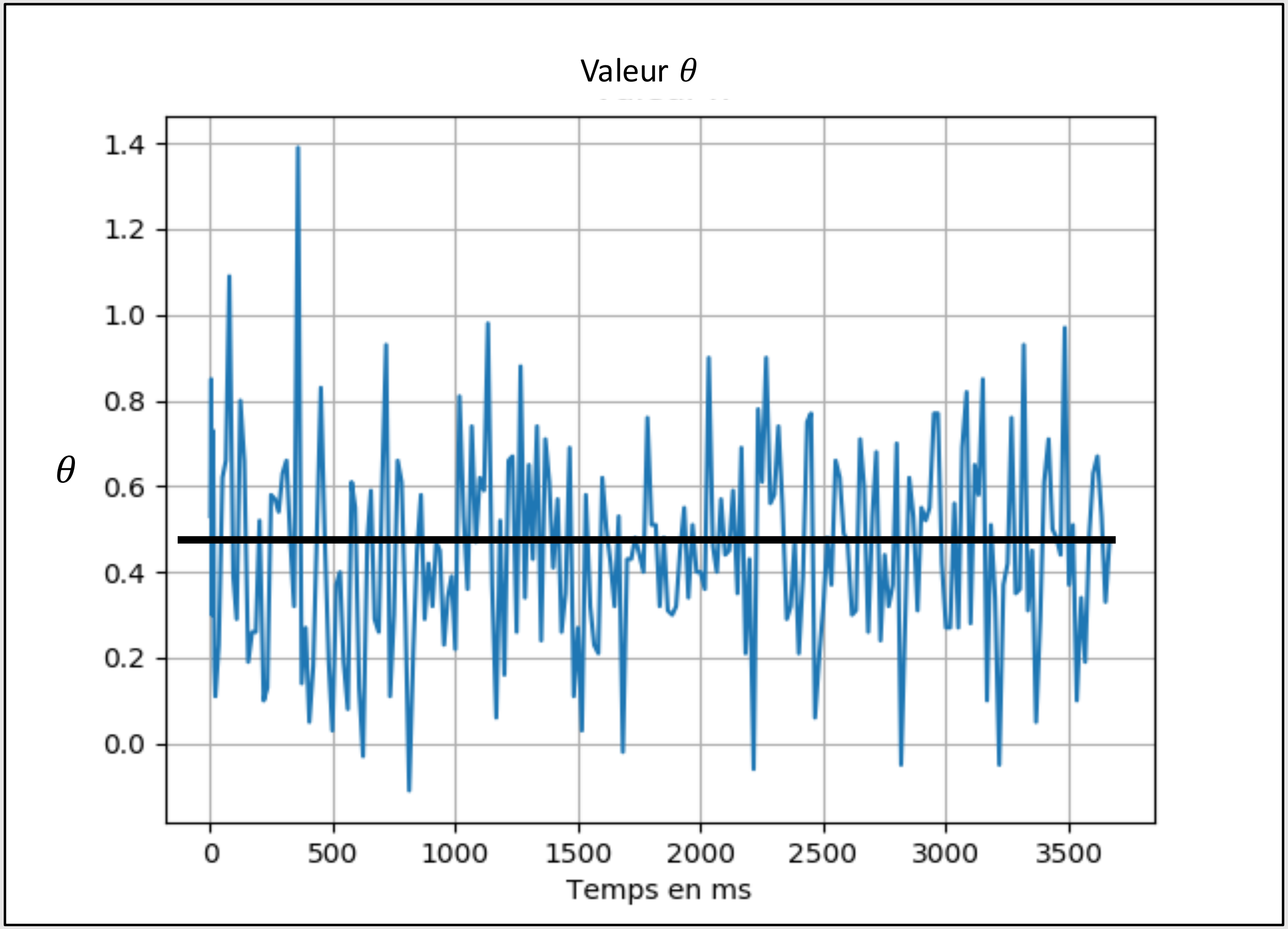
Conditions initiales :

$$\begin{cases} \theta\,(t=0) = 24.5 \pm 0.5\ deg \\ \dot{\theta}(t=0) = 0\ \text{deg}.\,s^{-1} \\ l = 21\ cm \end{cases}$$


Valeur vitesse z

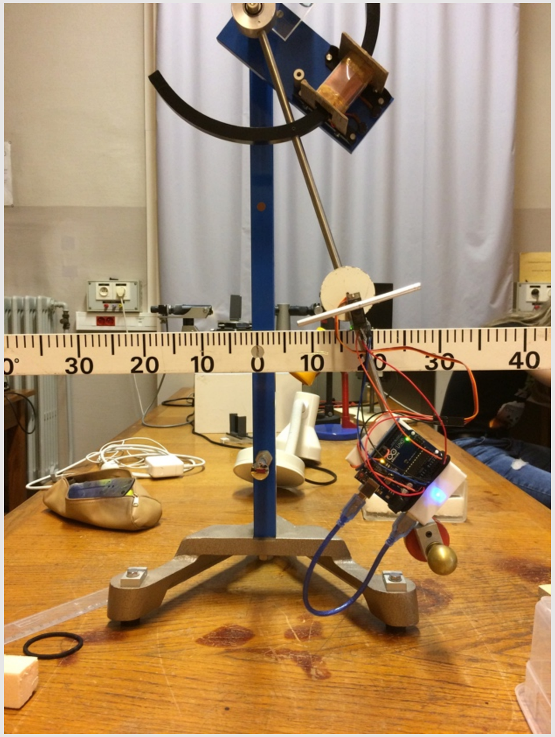

Valeur vitesse y

Angle au repos



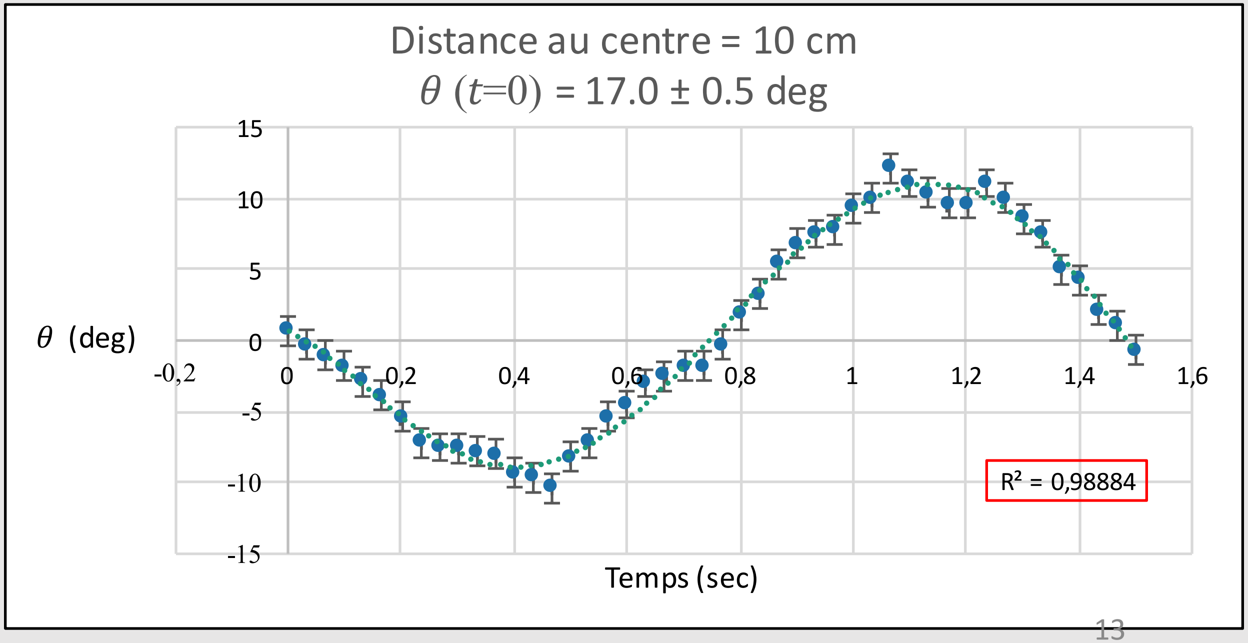
Valeur moyenne = 0.47 deg

3. Expérience et prévision en référentiel non galiléen



Conditions initiales :
$$\begin{cases} \theta\,(t=0) = 17.0 \pm 0.5\; deg \\ \dot{\theta}(t=0) = 0 \text{ deg}.\,s^{-1} \\ l = 10\; cm \end{cases}$$

θ < Perturbation = 17.0 ±0.5 deg



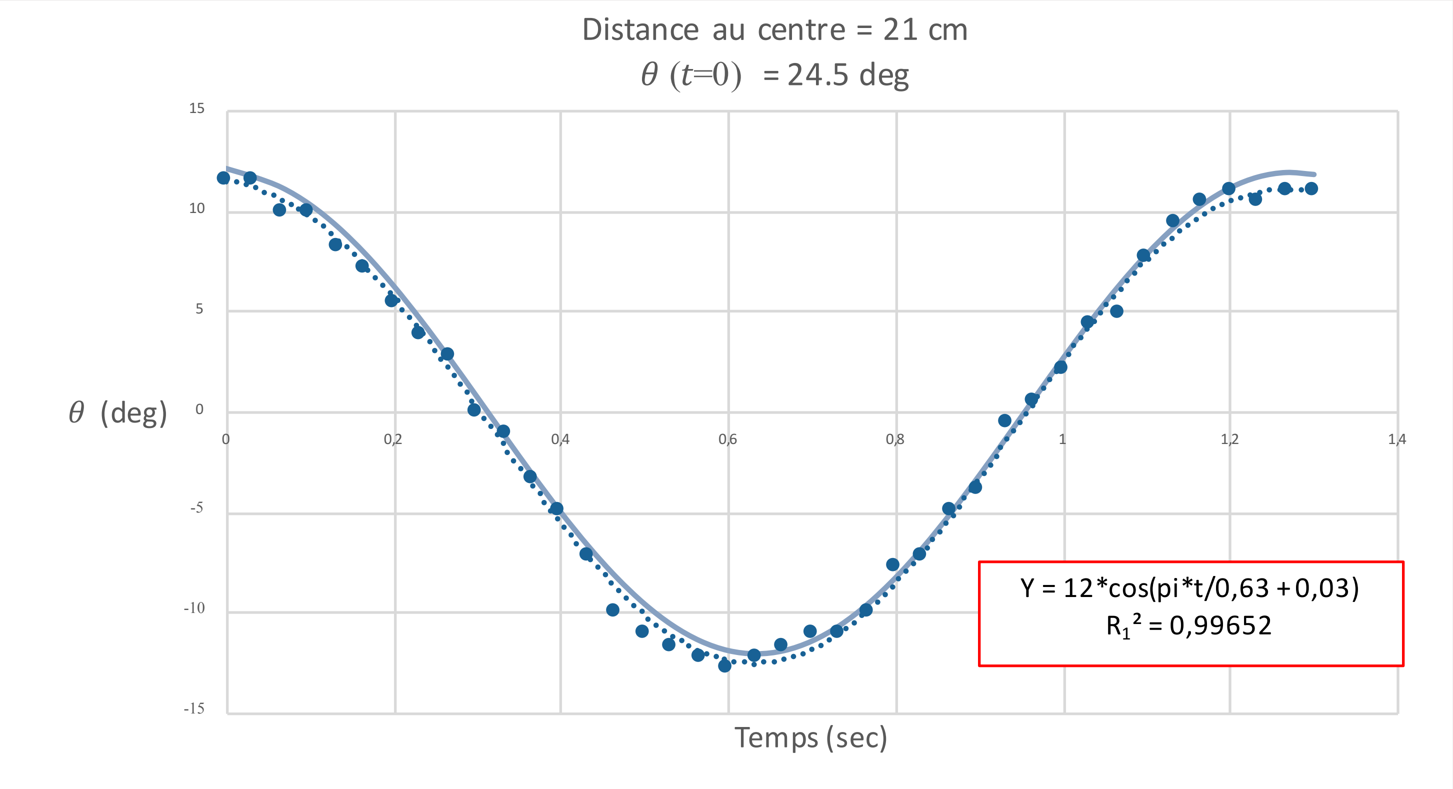
Distance au centre = 10 cm
$\theta\ (t{=}0)$ = 17.0 ± 0.5 deg

Autre pointage:

$$\begin{cases} \theta\ (t = 0) = 24.5 \pm 0.5\ deg \\ \dot{\theta}(t = 0) = 0\ \text{deg}.s^{-1} \\ l = 21\ cm \end{cases}$$

Mouvement sinusoïdal ⇒ Logique car perturbation de la forme $\theta(t) = \theta_0 \cos(\ \omega_0 t\ )$

⇒ Approximation des petits angles



Distance au centre = 21 cm

$\theta\ (t=0)$ = 24.5 deg

Y = 12*cos(pi*t/0,63 +0,03)

$R_1{}^2$ = 0,99652

Accélération considérée en référentiel non galiléen

$$-Ay = g\sin(\theta) - l\ddot{\theta}$$

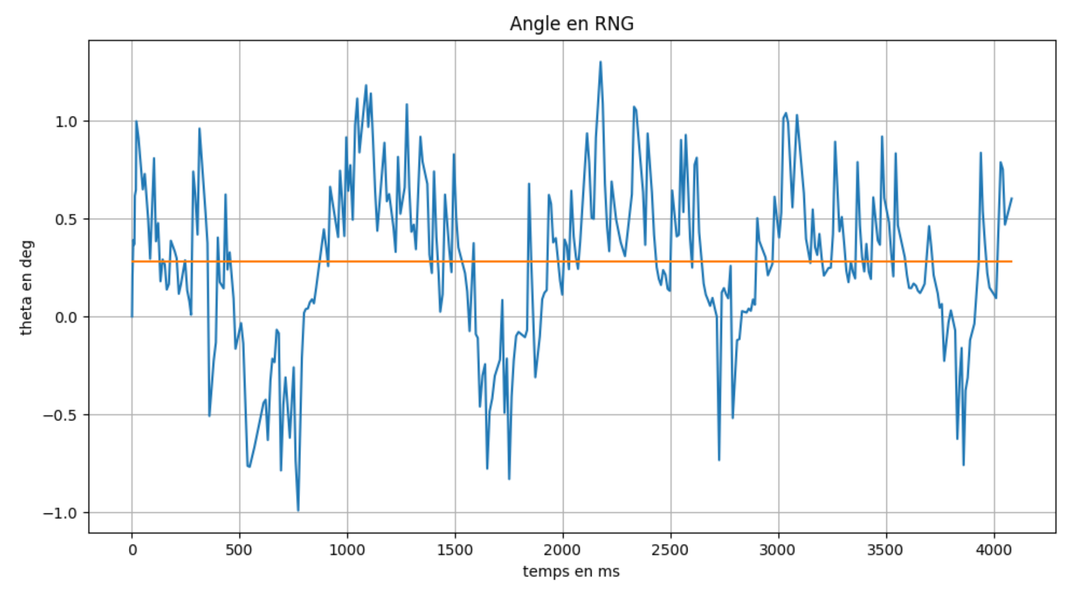
$$Az = g\cos(\theta) + l\dot{\theta}^2$$

$$\text{avec } \theta\ (t) = 12\cos(\frac{\pi}{0.63}t + 0.03)$$

$$\theta_{\text{ mesurée}} = \tan^{-1} \frac{-Ay + l\ddot{\theta}}{Az - l\dot{\theta}^2}$$

Amplitude beaucoup plus faible

Besoin de correction du bruit



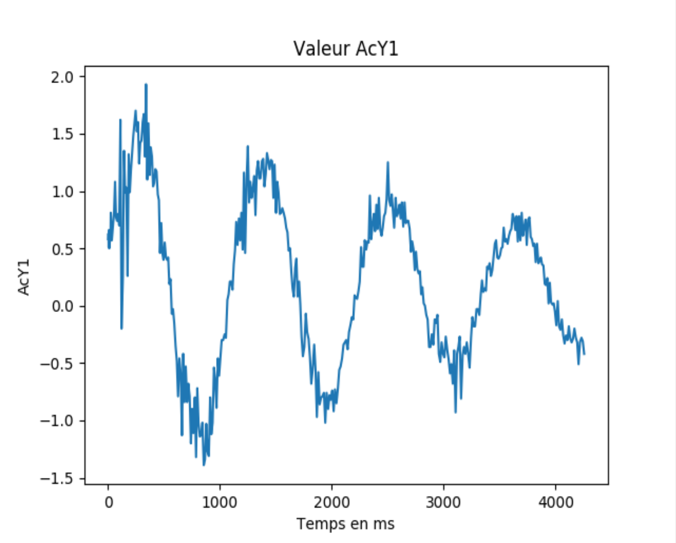
III. Améliorations

1. Considération du bruit

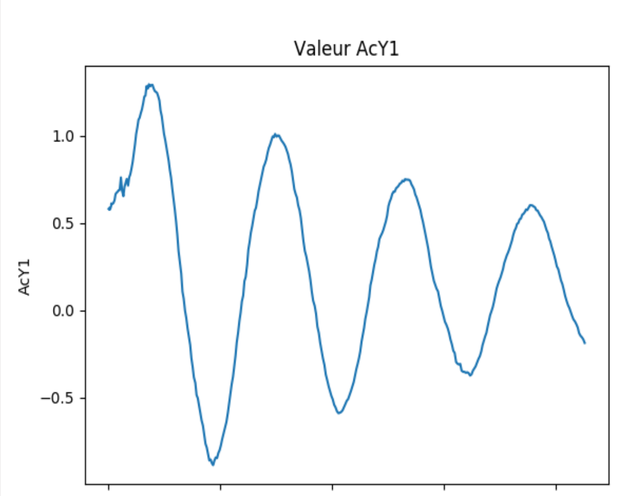
Equation filtre passe-bas: $\frac{ds}{dt} + \omega_0{}^2\, s = \omega_0{}^2\, e$

$$\Leftrightarrow \boxed{\frac{s[n] - s[n-1]}{dt} + \omega_0{}^2\, s[n] = e[n]\omega_0{}^2}$$

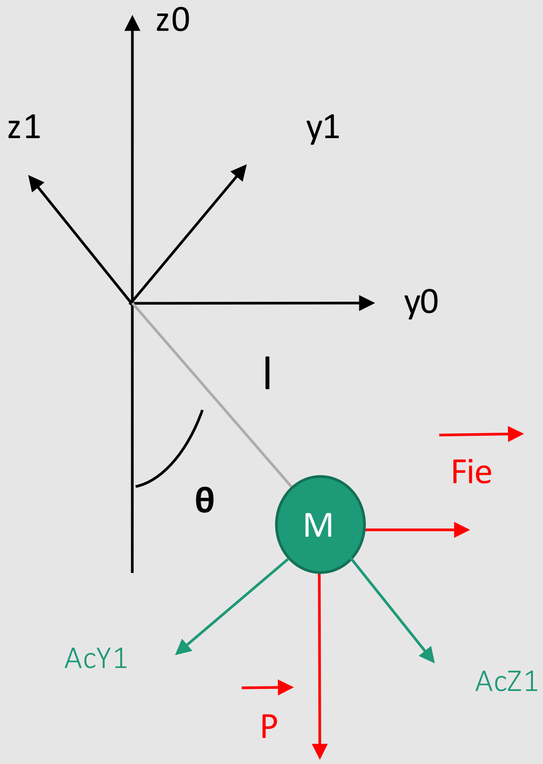
$\omega_0 = 2\pi f_{coup}$ avec $f_{coup} \sim 2Hz$: la fréquence de la marche de l'homme



Filtre P-B



2. Amélioration en référentiel non galiléen



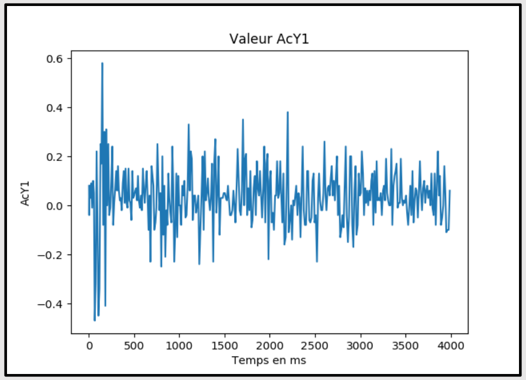
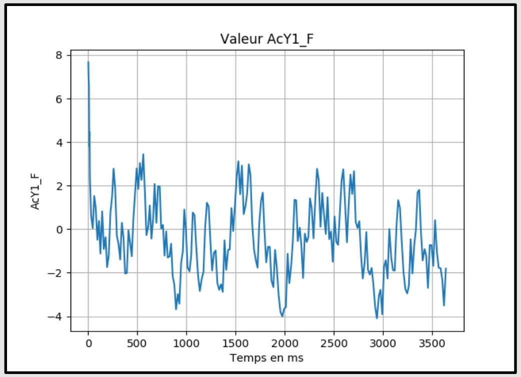
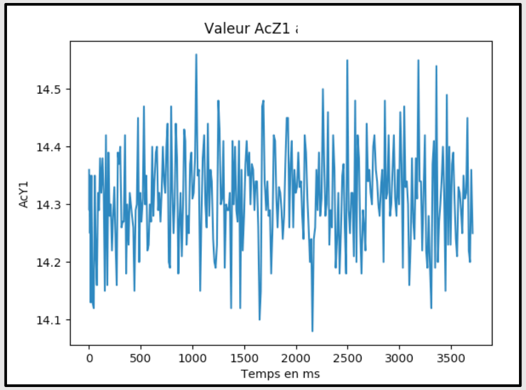
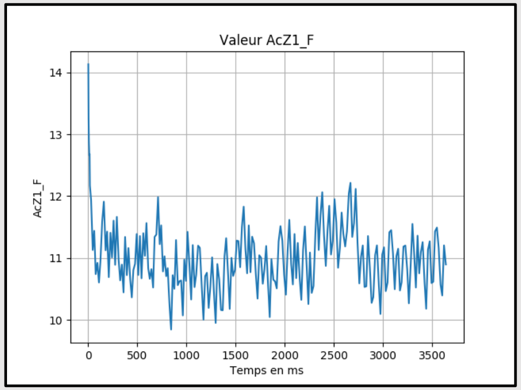
Accéléromètre en rotation sur le pendule

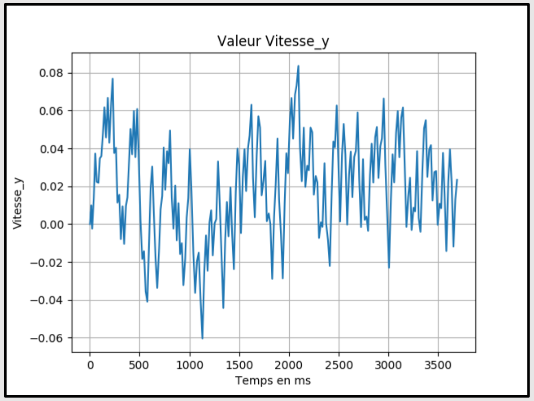
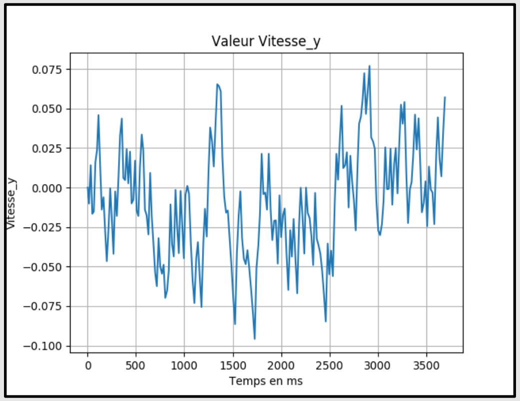
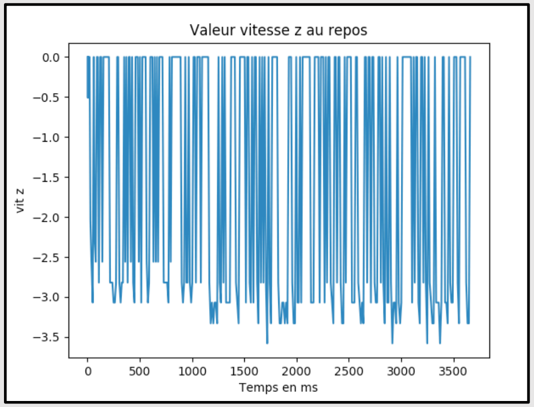
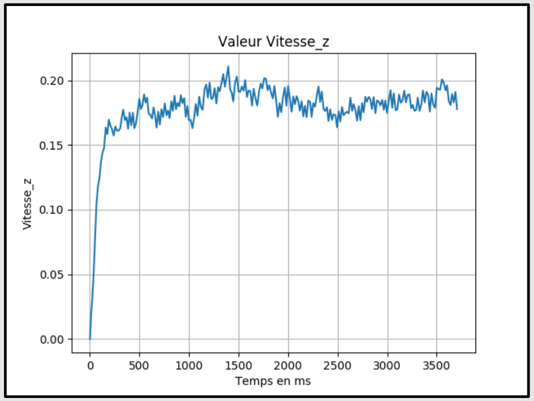
Principe fondamental de la dynamique sur M:

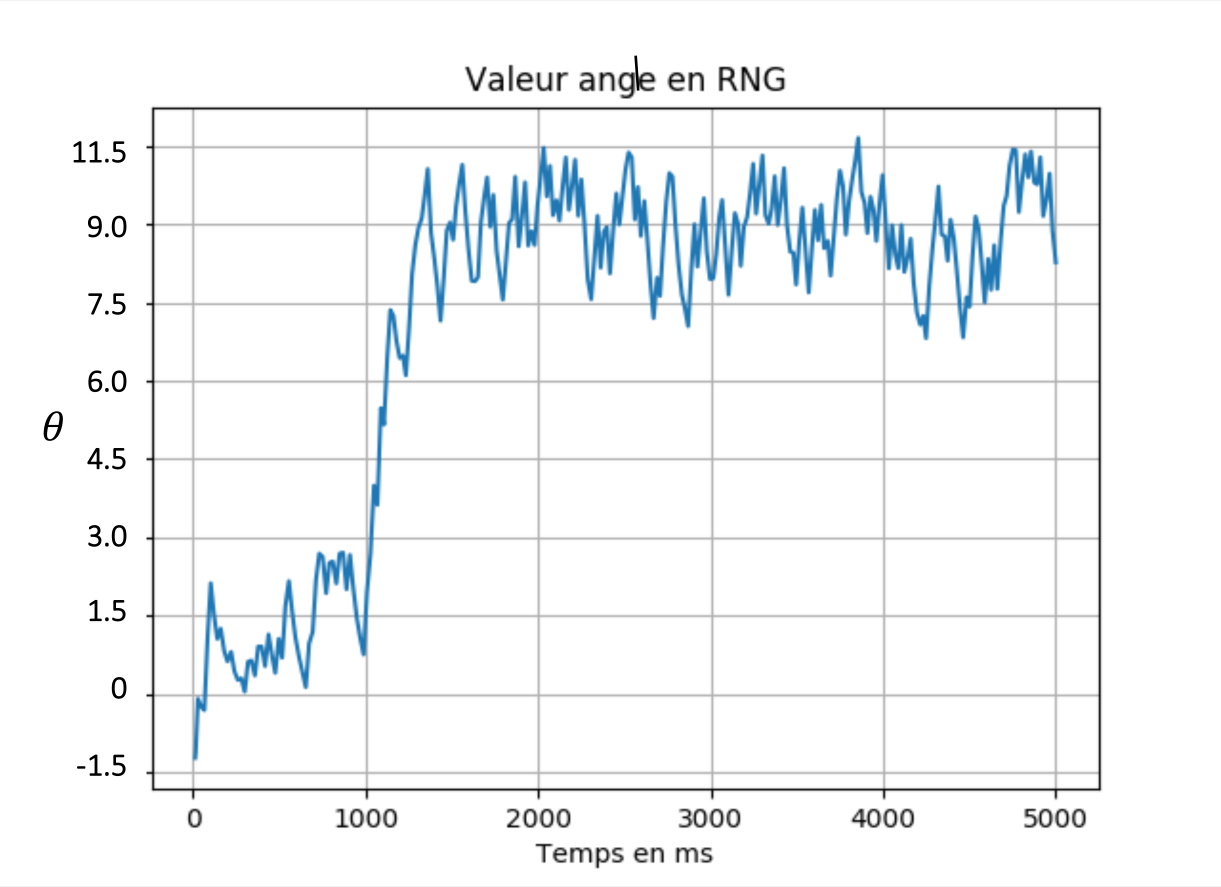
$$\boxed{\theta = \tanh^{-1} \frac{AcY1 + l\,\omega_0\,\theta_0 \sin(\omega_0 t)}{AcZ1 - l\,\omega_0^2\,\theta_0 \cos(\omega_0 t)}}$$

Avec $\theta(t) = \theta_0 \cos(\omega_0 t)$

Conditions initiales : $\begin{cases} \theta\ (t=0) = 24.5 \pm 0.5\ deg \\ l = 21.0 \pm 0.1 cm \end{cases}$

	Accélération en référentiel galiléen	Accélération en référentiel non-galiléen
Accélération selon Y1 ➡ Belle correction		
Accélération selon Z1 ➡ Plus proche de g		

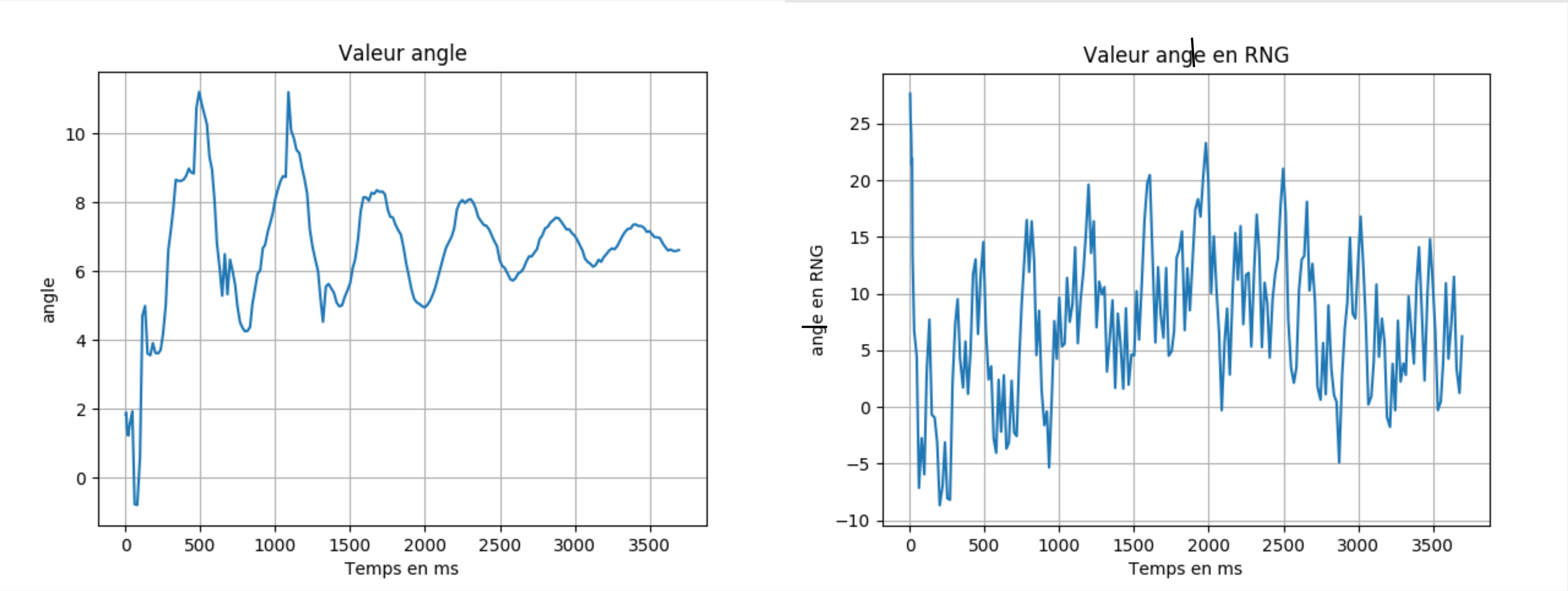
	Vitesse en référentiel galiléen	Vitesse en référentiel non galiléen
Vitesse selon Y1	 	
Vitesse selon Z1	 	



Stable au début
A t = 1000 ms : le système diverge

Désynchronisation à partir d'une seconde
Instabilité du système ➡ approximation des petits angles fausse

➡ En remplaçant l$\dot{\theta}^2$ par $\dfrac{(Vitesse\ z)^2}{l}$



Système en RNG instable

Conclusion

- Référentiel non galiléen → système plus instable

- Améliorations possibles:

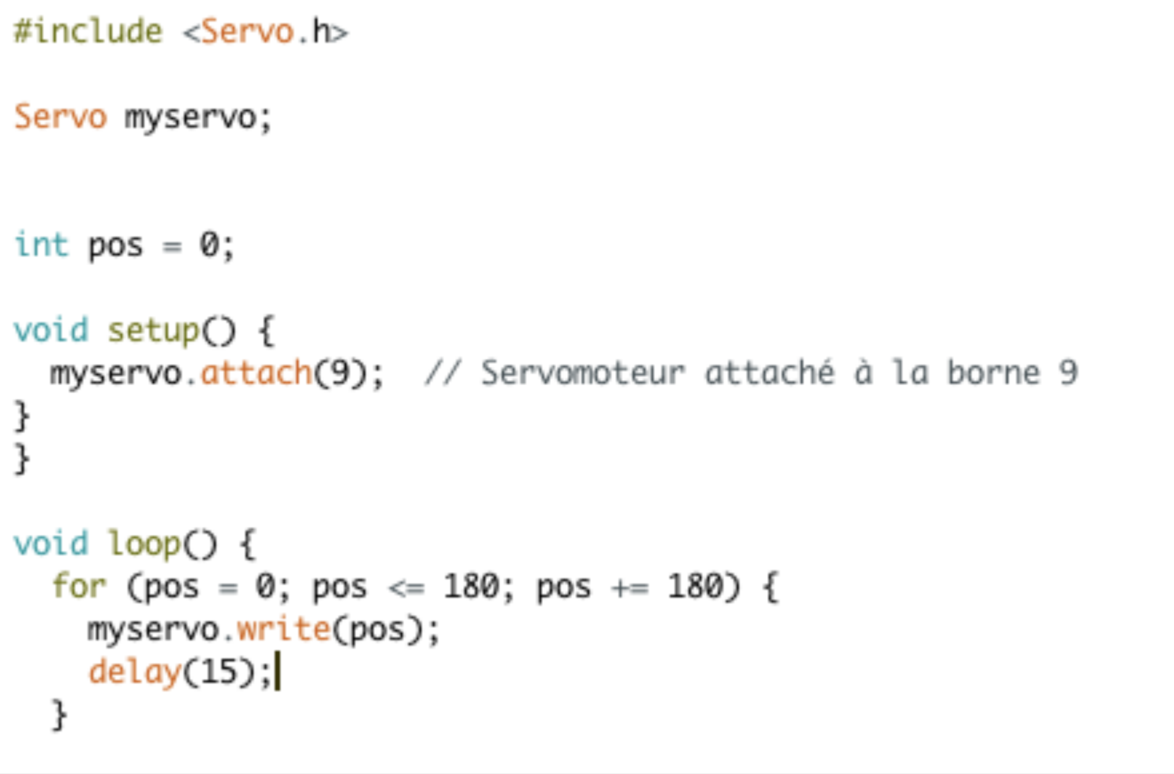
  - Se placer en étude quasi-statique

  - Augmenter la bande passante de stabilité en rajoutant un gyromètre pour la stabilité en HF et BF

Annexe

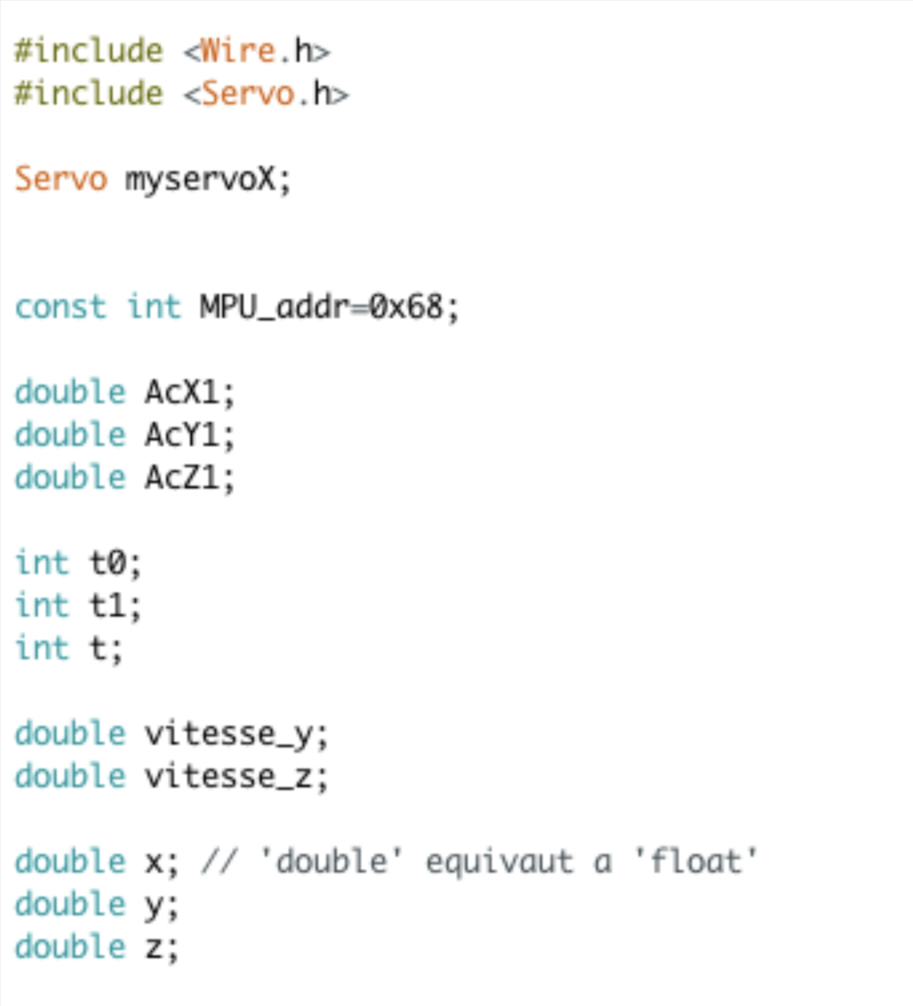
- Programmes sur Arduino

Réponse du moteur à un échelon de 180 degrés



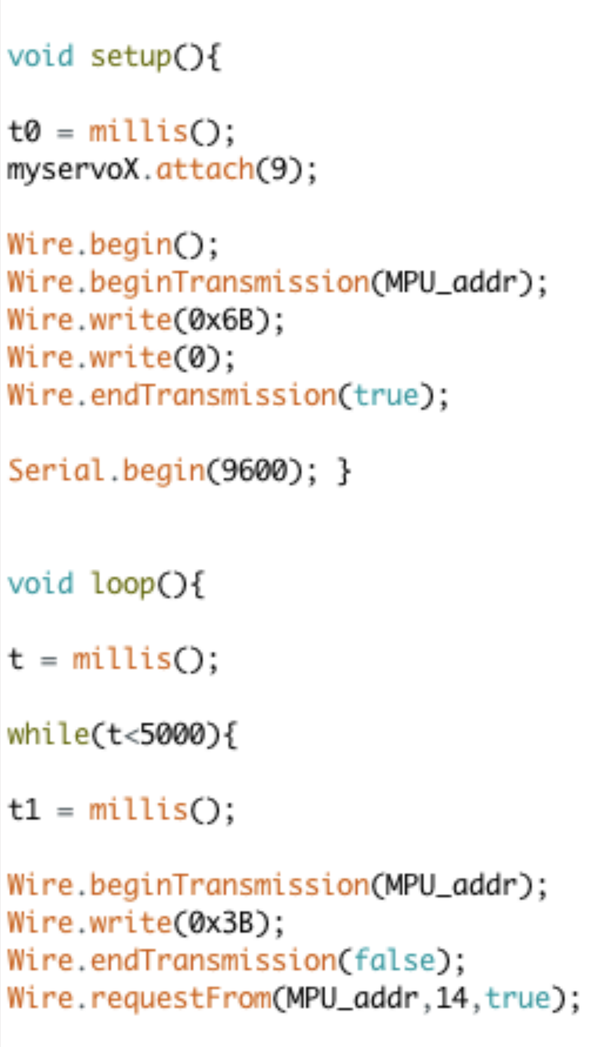
```
#include <Servo.h>

Servo myservo;


int pos = 0;

void setup() {
  myservo.attach(9);  // Servomoteur attaché à la borne 9
}
}

void loop() {
  for (pos = 0; pos <= 180; pos += 180) {
    myservo.write(pos);
    delay(15);
  }
```

- Programmes sur Arduino



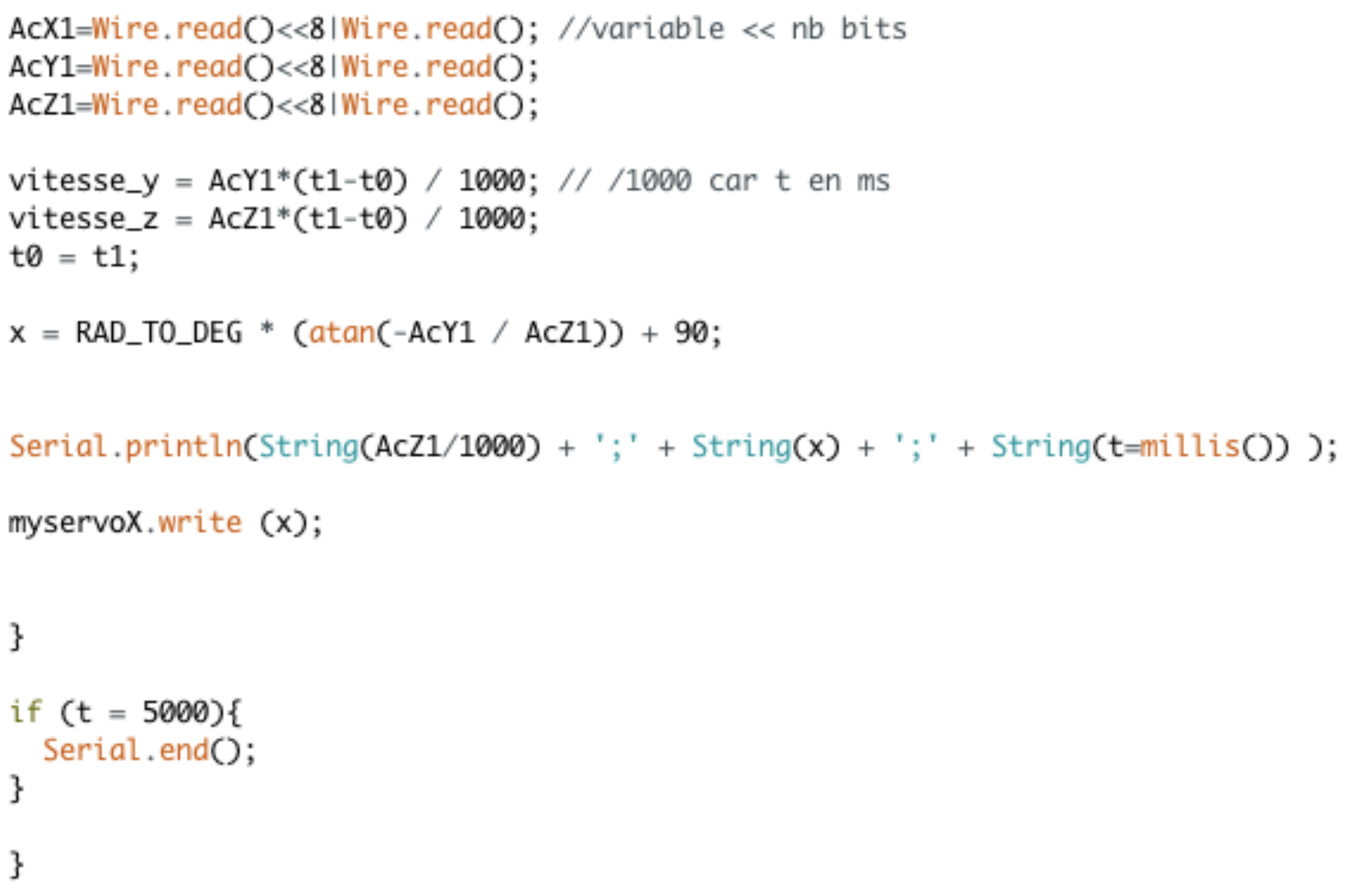
```
#include <Wire.h>
#include <Servo.h>

Servo myservoX;


const int MPU_addr=0x68;

double AcX1;
double AcY1;
double AcZ1;

int t0;
int t1;
int t;

double vitesse_y;
double vitesse_z;

double x; // 'double' equivaut a 'float'
double y;
double z;
```

- Programmes sur Arduino



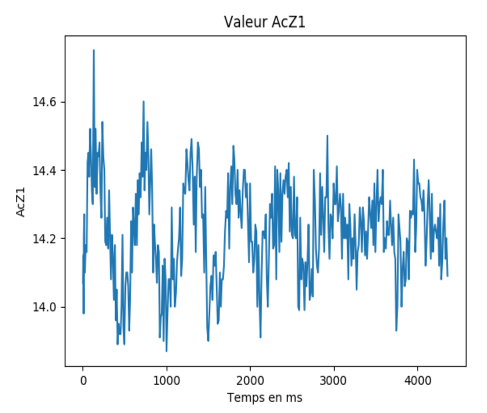
```
void setup(){

t0 = millis();
myservoX.attach(9);

Wire.begin();
Wire.beginTransmission(MPU_addr);
Wire.write(0x6B);
Wire.write(0);
Wire.endTransmission(true);

Serial.begin(9600); }


void loop(){

t = millis();

while(t<5000){

t1 = millis();

Wire.beginTransmission(MPU_addr);
Wire.write(0x3B);
Wire.endTransmission(false);
Wire.requestFrom(MPU_addr,14,true);
```

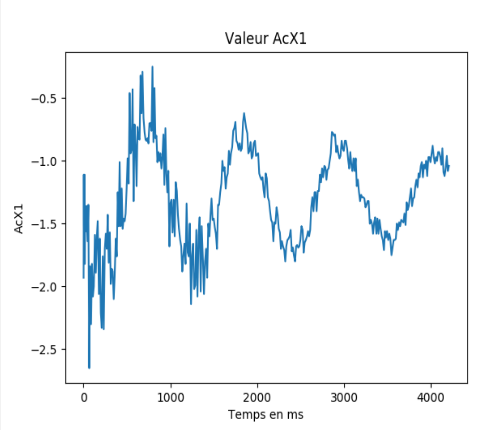
- Programmes sur Arduino



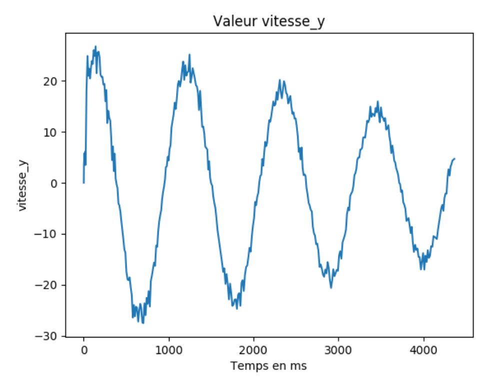
```
AcX1=Wire.read()<<8|Wire.read(); //variable << nb bits
AcY1=Wire.read()<<8|Wire.read();
AcZ1=Wire.read()<<8|Wire.read();

vitesse_y = AcY1*(t1-t0) / 1000; // /1000 car t en ms
vitesse_z = AcZ1*(t1-t0) / 1000;
t0 = t1;

x = RAD_TO_DEG * (atan(-AcY1 / AcZ1)) + 90;


Serial.println(String(AcZ1/1000) + ';' + String(x) + ';' + String(t=millis()) );

myservoX.write (x);


}

if (t = 5000){
  Serial.end();
}

}
```

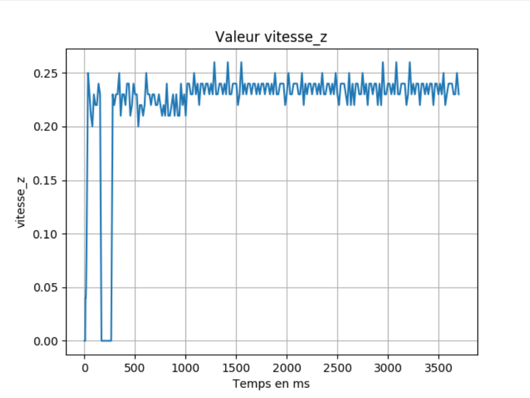
- Autres études





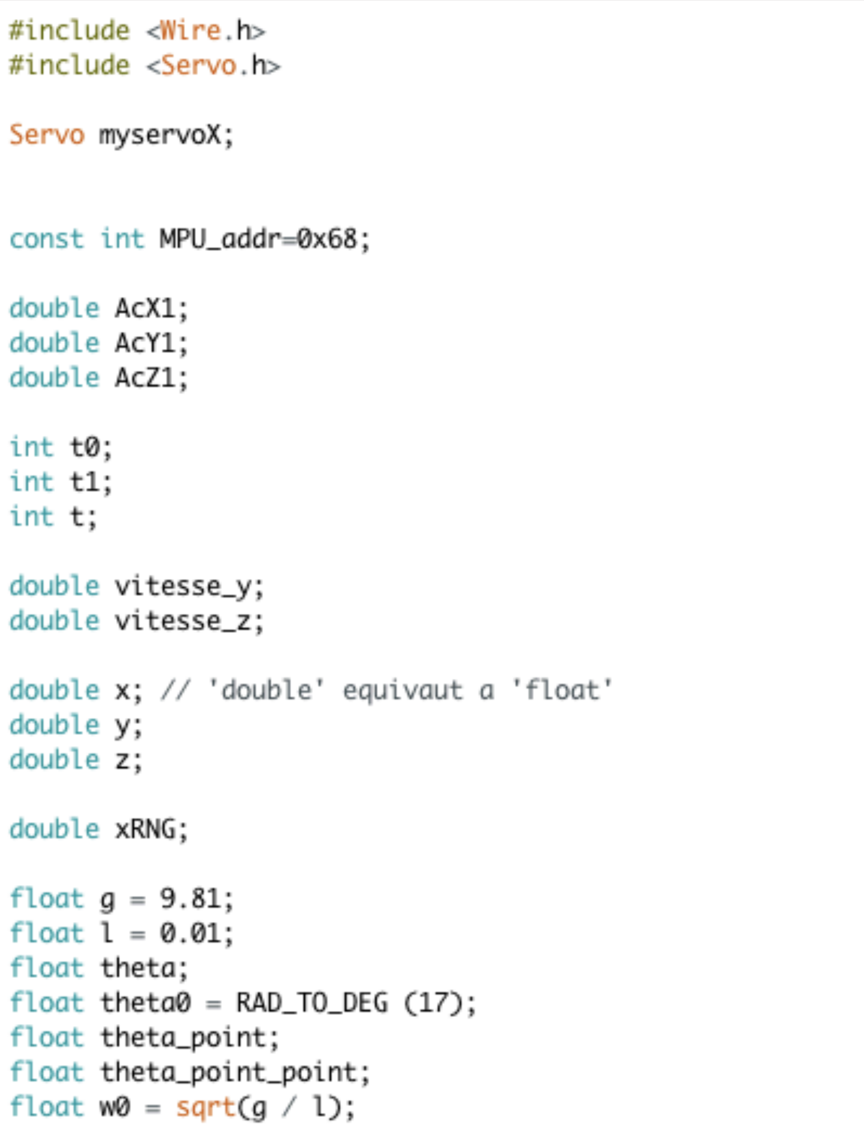
$$\begin{cases} \theta\ (t=0) = 17.0 \pm 0.5\, deg \\ \dot{\theta}(t=0) = 0\ \mathrm{deg}.s^{-1} \\ l = 10\ cm \end{cases}$$





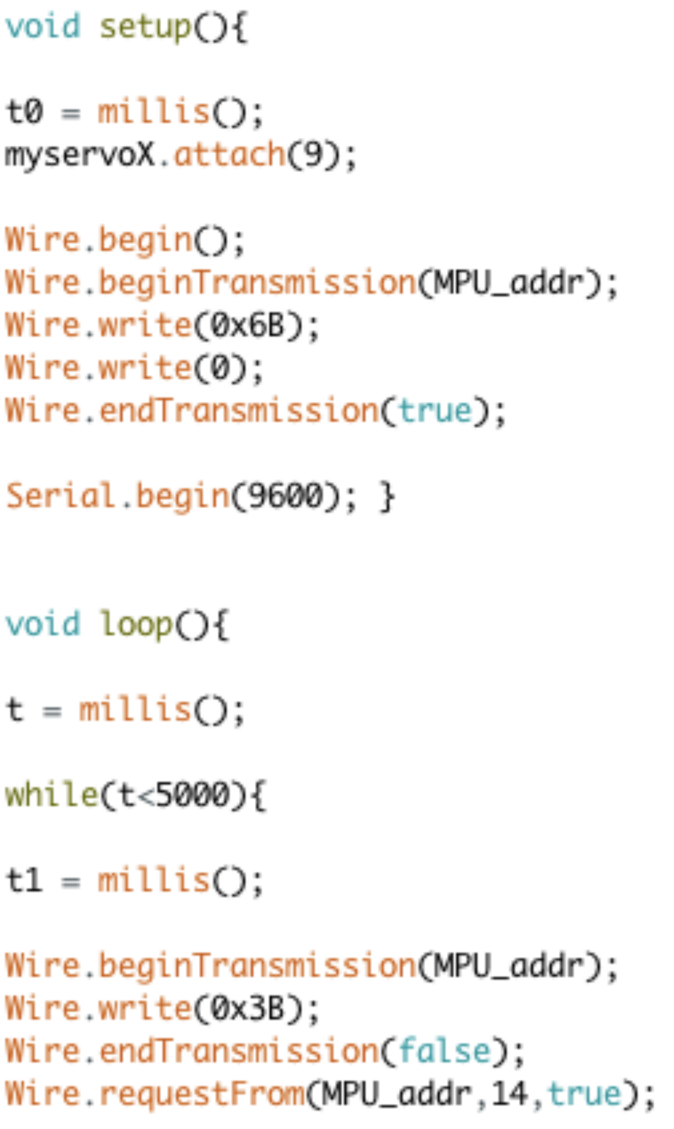
$$\begin{cases} \theta\ (t=0) = 17.0 \pm 0.5\, deg \\ \dot{\theta}(t=0) = 0\ \mathrm{deg}.s^{-1} \\ l = 10\ cm \end{cases}$$

- Programmes sur Arduino



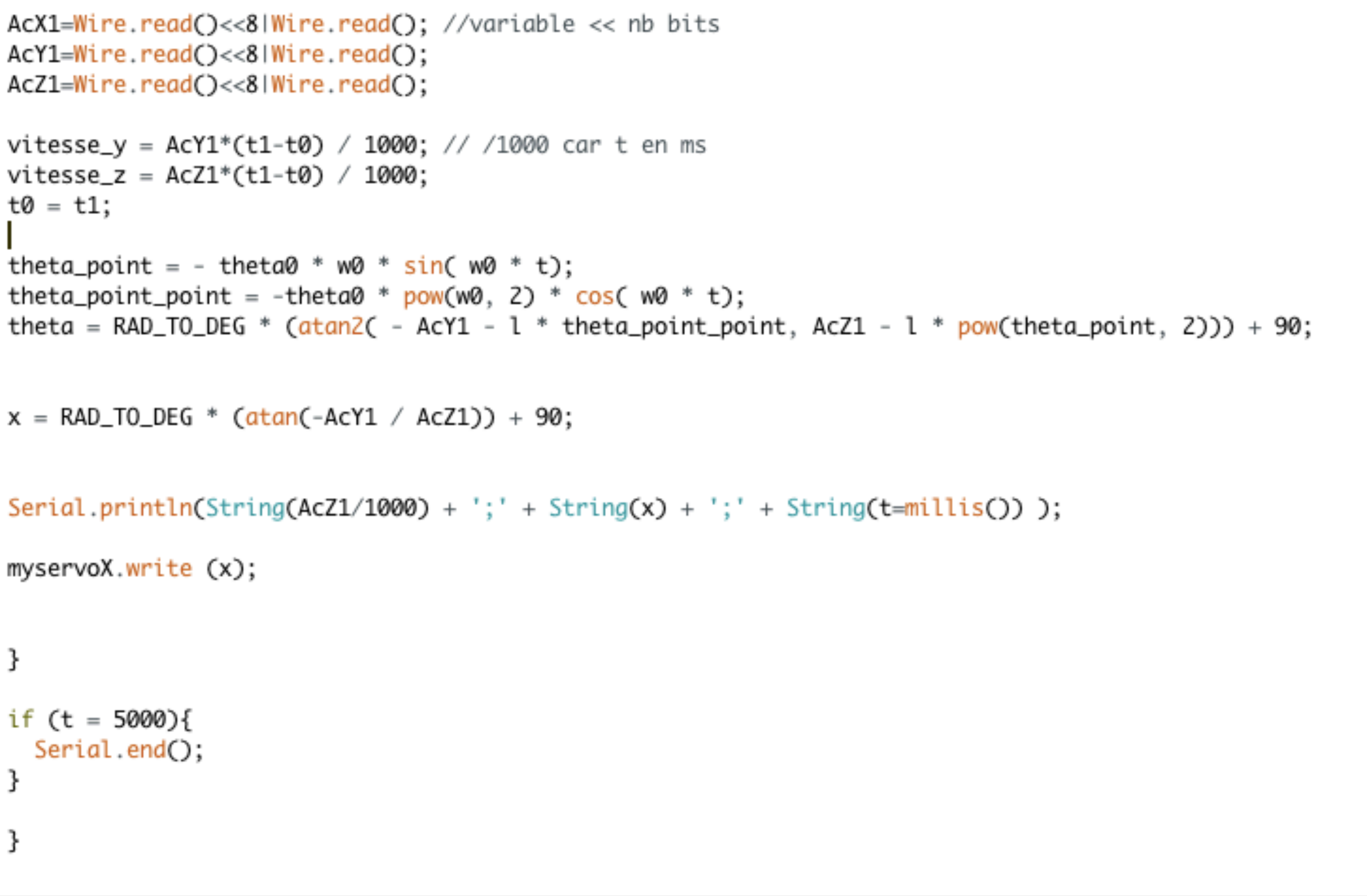
```
#include <Wire.h>
#include <Servo.h>

Servo myservoX;


const int MPU_addr=0x68;

double AcX1;
double AcY1;
double AcZ1;

int t0;
int t1;
int t;

double vitesse_y;
double vitesse_z;

double x; // 'double' equivaut a 'float'
double y;
double z;

double xRNG;

float g = 9.81;
float l = 0.01;
float theta;
float theta0 = RAD_TO_DEG (17);
float theta_point;
float theta_point_point;
float w0 = sqrt(g / l);
```

- Programmes sur Arduino



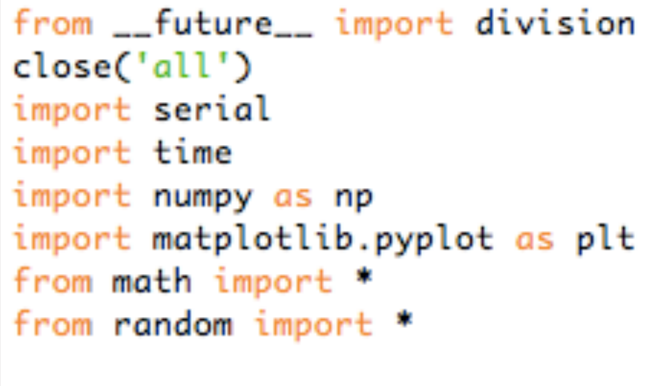
```
void setup(){

t0 = millis();
myservoX.attach(9);

Wire.begin();
Wire.beginTransmission(MPU_addr);
Wire.write(0x6B);
Wire.write(0);
Wire.endTransmission(true);

Serial.begin(9600); }


void loop(){

t = millis();

while(t<5000){

t1 = millis();

Wire.beginTransmission(MPU_addr);
Wire.write(0x3B);
Wire.endTransmission(false);
Wire.requestFrom(MPU_addr,14,true);
```

- Programmes sur Python

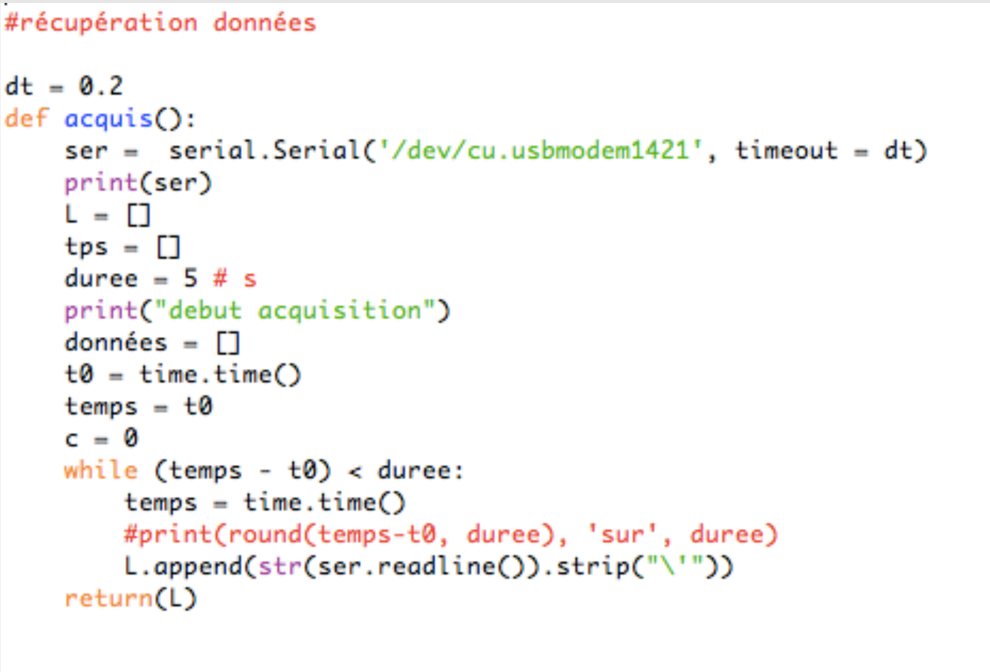


```
AcX1=Wire.read()<<8|Wire.read(); //variable << nb bits
AcY1=Wire.read()<<8|Wire.read();
AcZ1=Wire.read()<<8|Wire.read();

vitesse_y = AcY1*(t1-t0) / 1000; // /1000 car t en ms
vitesse_z = AcZ1*(t1-t0) / 1000;
t0 = t1;

theta_point = - theta0 * w0 * sin( w0 * t);
theta_point_point = -theta0 * pow(w0, 2) * cos( w0 * t);
theta = RAD_TO_DEG * (atan2( - AcY1 - l * theta_point_point, AcZ1 - l * pow(theta_point, 2))) + 90;


x = RAD_TO_DEG * (atan(-AcY1 / AcZ1)) + 90;


Serial.println(String(AcZ1/1000) + ';' + String(x) + ';' + String(t=millis()) );

myservoX.write (x);


}

if (t = 5000){
  Serial.end();
}

}
```

- Programmes sur Python

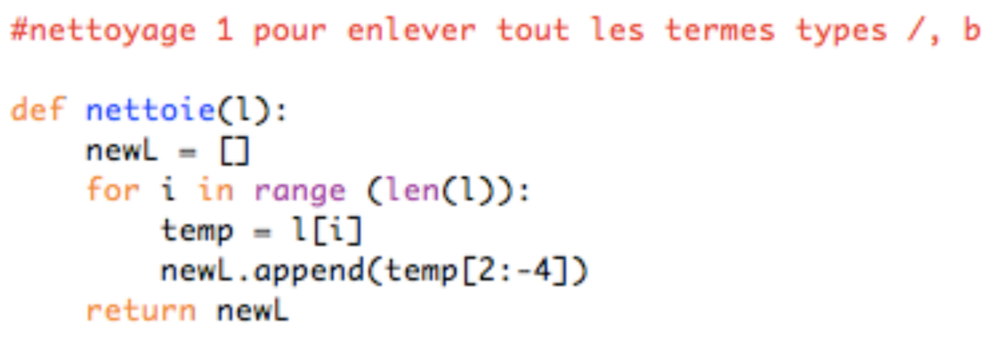


```
from __future__ import division
close('all')
import serial
import time
import numpy as np
import matplotlib.pyplot as plt
from math import *
from random import *
```

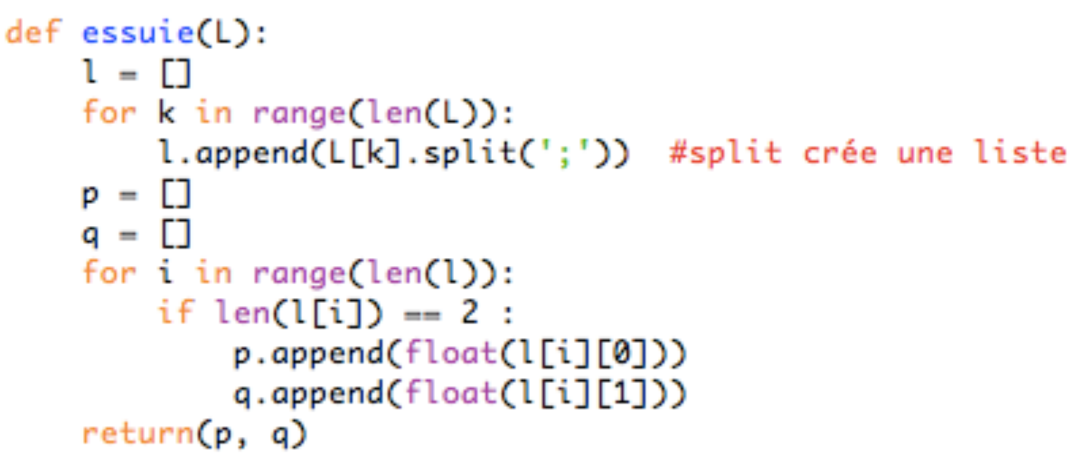


```
#récupération données

dt = 0.2
def acquis():
    ser =  serial.Serial('/dev/cu.usbmodem1421', timeout = dt)
    print(ser)
    L = []
    tps = []
    duree = 5 # s
    print("debut acquisition")
    données = []
    t0 = time.time()
    temps = t0
    c = 0
    while (temps - t0) < duree:
        temps = time.time()
        #print(round(temps-t0, duree), 'sur', duree)
        L.append(str(ser.readline()).strip("\'"))
    return(L)
```

- Programmes sur Python

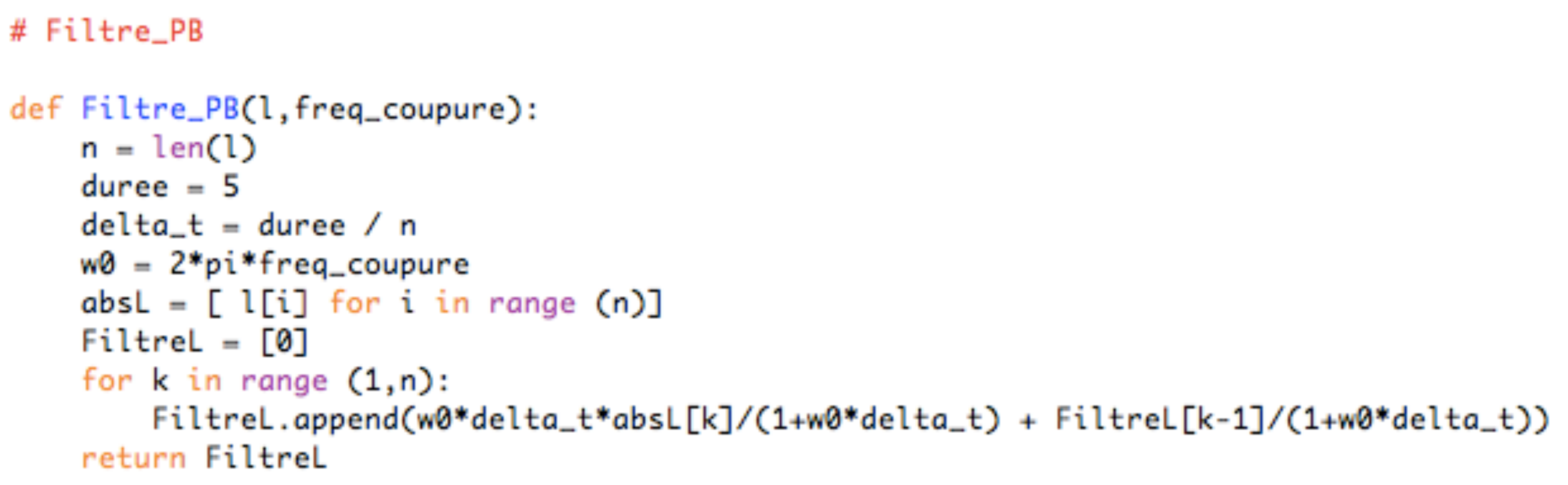


```
#nettoyage 1 pour enlever tout les termes types /, b

def nettoie(l):
    newL = []
    for i in range (len(l)):
        temp = l[i]
        newL.append(temp[2:-4])
    return newL
```

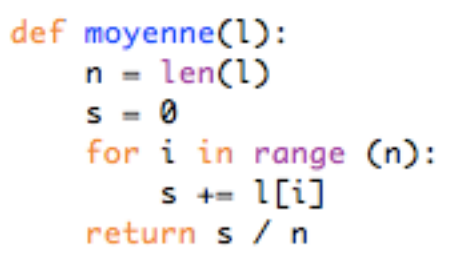


```
def essuie(L):
    l = []
    for k in range(len(L)):
        l.append(L[k].split(';'))  #split crée une liste
    p = []
    q = []
    for i in range(len(l)):
        if len(l[i]) == 2 :
            p.append(float(l[i][0]))
            q.append(float(l[i][1]))
    return(p, q)
```

- Programmes sur Python

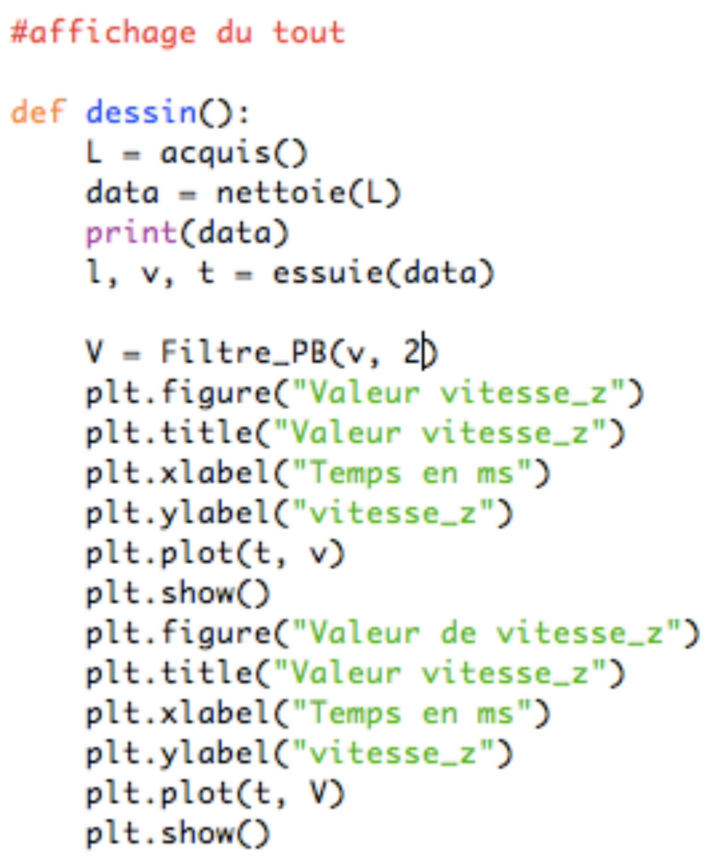


```
# Filtre_PB

def Filtre_PB(l,freq_coupure):
    n = len(l)
    duree = 5
    delta_t = duree / n
    w0 = 2*pi*freq_coupure
    absL = [ l[i] for i in range (n)]
    FiltreL = [0]
    for k in range (1,n):
        FiltreL.append(w0*delta_t*absL[k]/(1+w0*delta_t) + FiltreL[k-1]/(1+w0*delta_t))
    return FiltreL
```



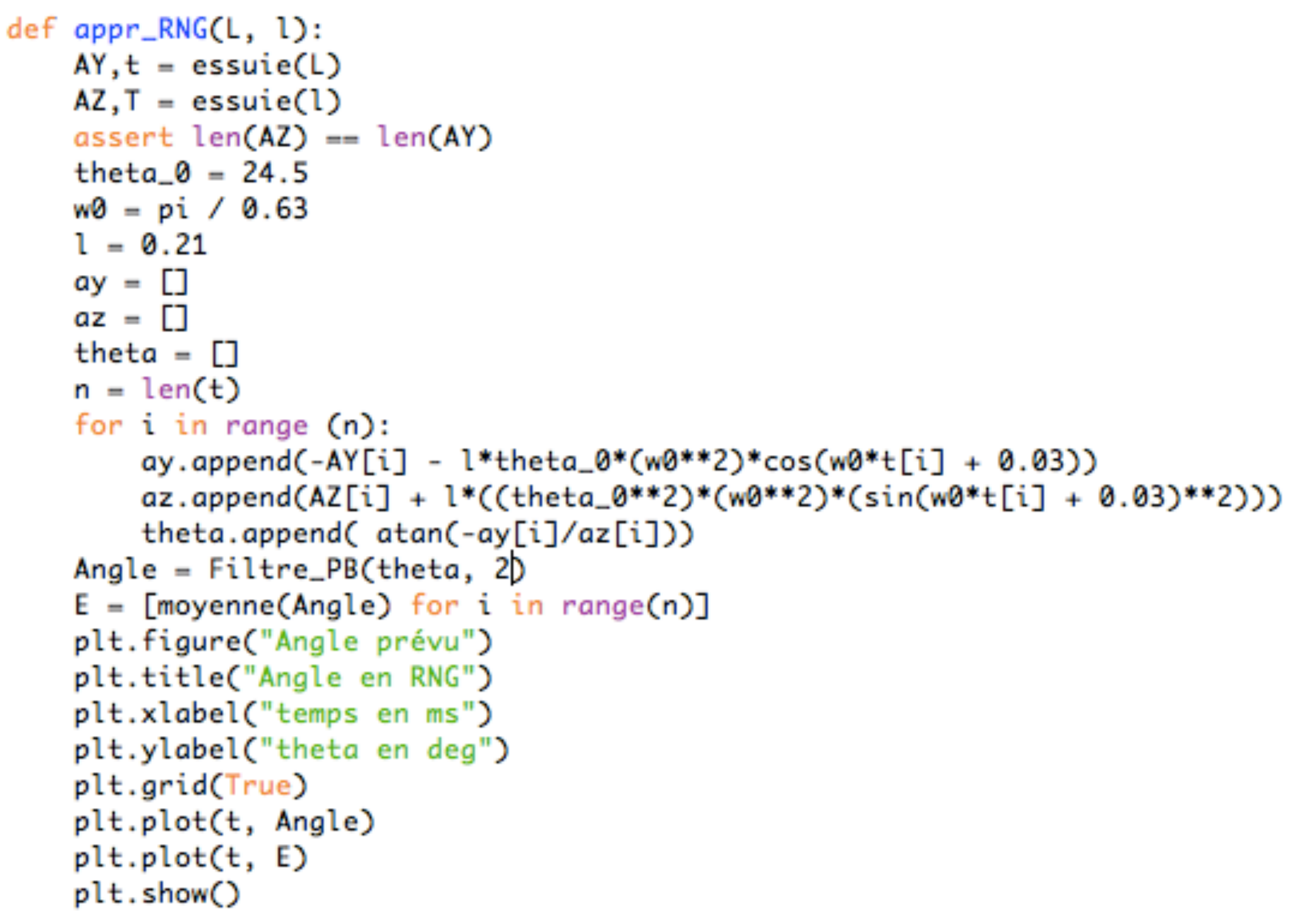
```
def moyenne(l):
    n = len(l)
    s = 0
    for i in range (n):
        s += l[i]
    return s / n
```

- Programmes sur Python



```
#affichage du tout

def dessin():
    L = acquis()
    data = nettoie(L)
    print(data)
    l, v, t = essuie(data)

    V = Filtre_PB(v, 2)
    plt.figure("Valeur vitesse_z")
    plt.title("Valeur vitesse_z")
    plt.xlabel("Temps en ms")
    plt.ylabel("vitesse_z")
    plt.plot(t, v)
    plt.show()
    plt.figure("Valeur de vitesse_z")
    plt.title("Valeur vitesse_z")
    plt.xlabel("Temps en ms")
    plt.ylabel("vitesse_z")
    plt.plot(t, V)
    plt.show()
```

- Programmes sur Python



```
def appr_RNG(L, l):
    AY,t = essuie(L)
    AZ,T = essuie(l)
    assert len(AZ) == len(AY)
    theta_0 = 24.5
    w0 = pi / 0.63
    l = 0.21
    ay = []
    az = []
    theta = []
    n = len(t)
    for i in range (n):
        ay.append(-AY[i] - l*theta_0*(w0**2)*cos(w0*t[i] + 0.03))
        az.append(AZ[i] + l*((theta_0**2)*(w0**2)*(sin(w0*t[i] + 0.03)**2)))
        theta.append( atan(-ay[i]/az[i]))
    Angle = Filtre_PB(theta, 2)
    E = [moyenne(Angle) for i in range(n)]
    plt.figure("Angle prévu")
    plt.title("Angle en RNG")
    plt.xlabel("temps en ms")
    plt.ylabel("theta en deg")
    plt.grid(True)
    plt.plot(t, Angle)
    plt.plot(t, E)
    plt.show()
```