Lycée André Chamson

ANNEXES DU MÉMOIRE

LA LÉGENDE DES TOURBILLONS

Mémoire préparé sous la direction de M. Magnoux

Présenté et soutenu par Ylan Hernández-Motte et Jules Péchard

29° Édition des Olympiades de Physique

Année scolaire 2021/2022

SOMMAIRE

Annexe 1 : Le dispositif expérimental

Figure 1: L'agitateur magnétique

Figure 2: L'alimentation

Figure 3: L'objet

Figure 4: Le barreau

Figure 5: Le trépied

Figure 6: Le dispositif de chronométrage

Figure 7: La photorésistance et le laser

Figure 8: Mesure de la distance

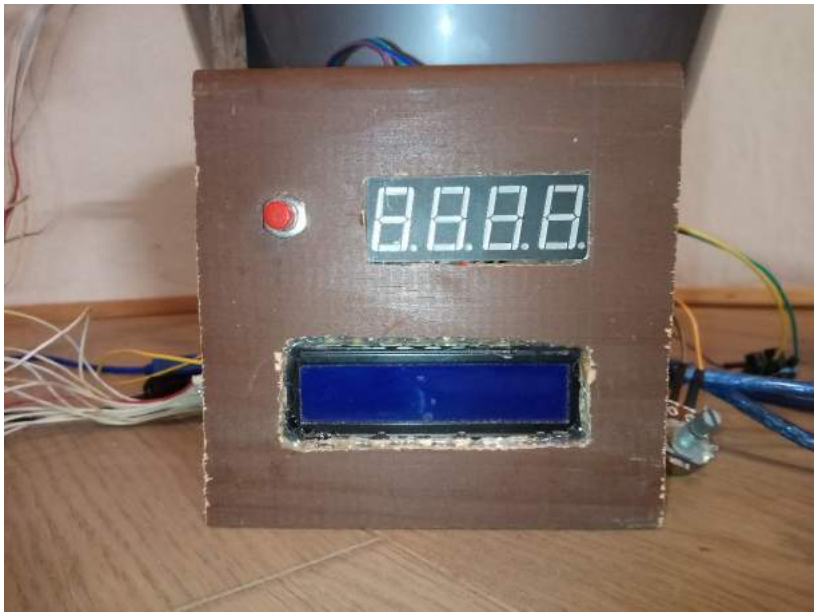

Figure 9: Les afficheurs

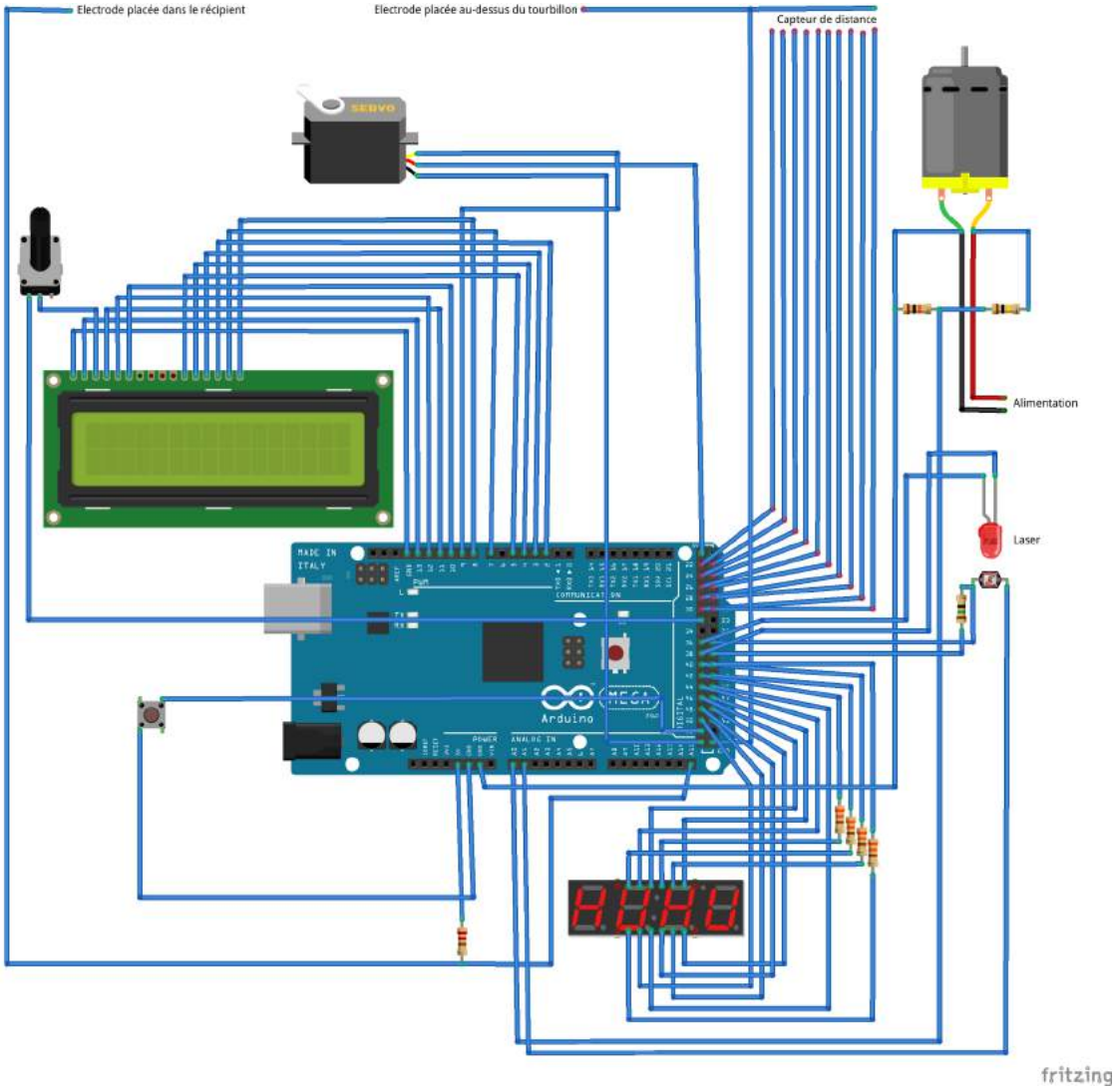

Figure 10: Schéma de câblage Arduino

Annexe 2 : Programme Arduino

```
#include "SevSeg.h"
#include "LiquidCrystal.h"  //Ajout des bibliothèques
#include "Servo.h"
// Création des objets sevseg, ldc et monServo
SevSeg sevseg;
LiquidCrystal lcd(11,10,5,4,3,2);
Servo monServo;
//Variables de l'afficheur LCD
char message1[16] = "";
char message2[16] = "";
//Variable du capteur de distance
int d = 0;
//Variables du chronomètre
int etatBouton;
int capteur;
float departChrono;
float tempsChrono;
//Variables du capteur de vitesse
int valeur1;
int valeur2;
float tempsVitesse;
int vitesse;
unsigned long tempsPassage[50];
//Variables du voltmètre
const int pinVoltmetre = 0;
float vout = 0.0;
float vin = 0.0;
float R1 = 100000.0;
float R2 = 10000.0;
int voltlue = 0;
float tensionMoyenne = 0;
float tension[100];
float somme;
char affichageTension[4];
void setup(){
   //Initialisation des broches
  pinMode(pinVoltmetre, INPUT);  //Broche voltmètre
  pinMode(A1, INPUT);  //Broche photorésitance
  pinMode(36, OUTPUT);
  digitalWrite(36, HIGH);  //Alimentation laser
  pinMode(38, OUTPUT);
  digitalWrite(38, LOW);
  pinMode(37, OUTPUT);
  digitalWrite(37, HIGH);  //Alimentation photorésistance
  pinMode(39, OUTPUT);
  digitalWrite(39, LOW);
  for (int i=22; i<=31; i++){
    pinMode(i, INPUT_PULLUP);  //Initialisation automatisée des broches du capteur
  }
  monServo.attach(9);  // Broche servomoteur
  pinMode(52, INPUT_PULLUP);  //Broche bouton
  pinMode(8, OUTPUT);
  digitalWrite(8, LOW);
  pinMode(7, OUTPUT);
```

```
  digitalWrite(7, HIGH);  //Alimentation écran LCD
  pinMode(13, OUTPUT);
  digitalWrite(13, LOW);
  pinMode(12, OUTPUT);
  digitalWrite(12, HIGH);
  pinMode(32, OUTPUT);
  digitalWrite(32, LOW);

  byte numDigits = 4;
  byte digitPins[] = {41, 43, 40, 42};  //Broches afficheur 7 segments
  byte segmentPins[] = {44, 47, 50, 49, 48, 46, 51, 45};
  //Initialisation des affichages
  Serial.begin(9600);  //Initialisation de la liaison série
  Serial.println("Distance;Vitesse;Tension;Temps d'aspiration");  //Envoie du messa
  lcd.begin(16, 2);  //Initialisation de l'afficheur LCD
  bool resistorsOnSegments = true;
  bool updateWithDelaysIn = true;   //Initialisation de l'afficheur 7 segments
  byte hardwareConfig = COMMON_CATHODE;
  sevseg.begin(hardwareConfig, numDigits, digitPins, segmentPins, resistorsOnSegmer
  sevseg.setBrightness(90);
  monServo.write(3);  //Initialisation de la position du servomoteur}
  void loop(){
etatBouton = digitalRead(52);  //Récupération de la valeur du bouton
    if (etatBouton == LOW){  //Test du bouton
     while (etatBouton == LOW){
        etatBouton = digitalRead(52);  //Anti-rebond
      }
      delay(200);
      monServo.write(30);  //Ouverture du servomoteur
      departChrono = millis();  //Début du chronomètre
      capteur = analogRead(A15);  //Récupération de la valeur du capteur
      //Chronométrage
      while (capteur > 940 && etatBouton == HIGH){
        capteur = analogRead(A15);
        etatBouton = digitalRead(52);
        tempsChrono = (millis() - departChrono) / 1000;
        sevseg.setNumberF(tempsChrono,2);
        sevseg.refreshDisplay();
      }
      monServo.write(3);  //Fermeture du servomoteur
      delay(200);
      while (etatBouton == LOW){
        etatBouton = digitalRead(52);  //Attente de confirmation par le bouton
        sevseg.refreshDisplay();  //Affichage sur l'afficheur 7 segments
      }
      sevseg.blank(); //Supression de l'affichage sur l'afficheur 7 segments
      Serial.print(d);
      Serial.print(";");
      Serial.print(vitesse);
      Serial.print(";");   //Envoi des données dans la liaison série
      Serial.print(tensionMoyenne);
      Serial.print(";");
      Serial.println(tempsChrono,2);
      delay(200);
```

```
    }
  for(int j=22; j<=31; j++){
    if (digitalRead(j)==LOW){   //Mesure de la distance
      if ((j-20) != d){
          d = j-20;
          lcd.setCursor(0,1);  //Affichage de la distance
          lcd.write(message2);
        }
      }
    }

  //Mesure de la vitesse de rotation
  valeur2 = analogRead(A1);
//Récupération de la valeur de la photorésistance
  if(valeur2>900){
    if(valeur1<900){
      for (int k=0; k<=48; k++){
        tempsPassage[k]=tempsPassage[k+1];
      }
//Moyenne de la vitesse de rotation
      tempsPassage[49]=millis();
      tempsVitesse = (tempsPassage[49] - tempsPassage[0])/49;
      tempsVitesse = tempsVitesse / 60000;
      vitesse = 1/tempsVitesse;
      sprintf(message1,"v=%dtrs/min  ",vitesse);
      lcd.home();
      lcd.setCursor(0,0);
      lcd.write(message1);  //Affichage de la vitesse de rotation
      lcd.setCursor(0,1);
      lcd.write(message2);
    }
  }
  valeur1=valeur2;
  //Mesure de la tension
  voltlue = analogRead(pinVoltmetre);
  vout = (voltlue * 5.0) / 1024.0;
  vin = vout / (R2/(R1+R2));
  for (int k=0; k<=98; k++){
    tension[k]=tension[k+1];
  }
  tension[99]=vin;
  somme = 0;
  for (int k=0; k<=99; k++){
    somme=somme+tension[k];
  }
  tensionMoyenne = somme/100.00;
  dtostrf(tensionMoyenne, 4, 2, affichageTension);
//Conversion du résultat en chaîne de charactères
  sprintf(message2,"U=%sV  d=%dcm ", affichageTension, d);
//Ajout de la chaine de charactères dans le message
}
```

Annexe 3 : Description du mouvement

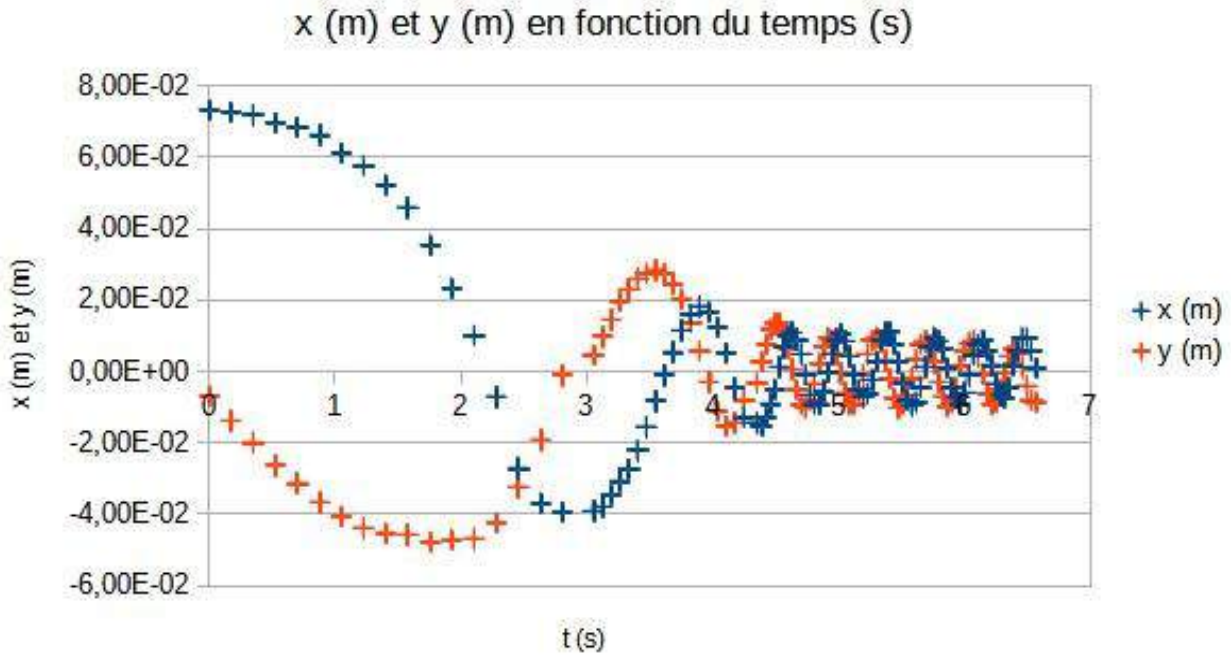

Figure 11

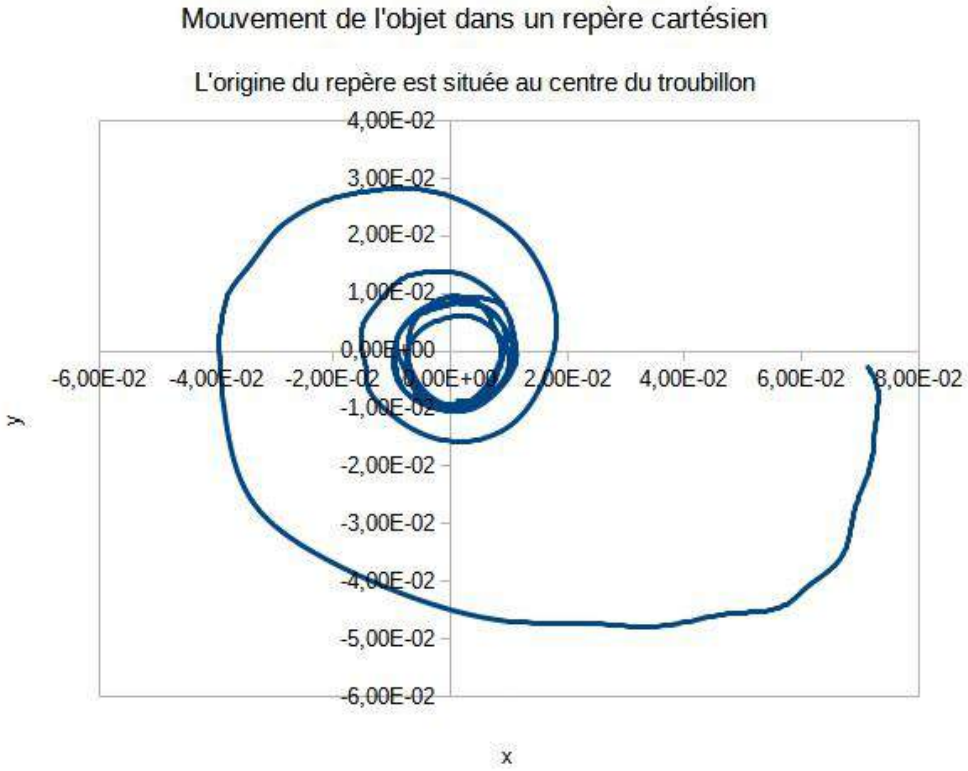

Figure 12

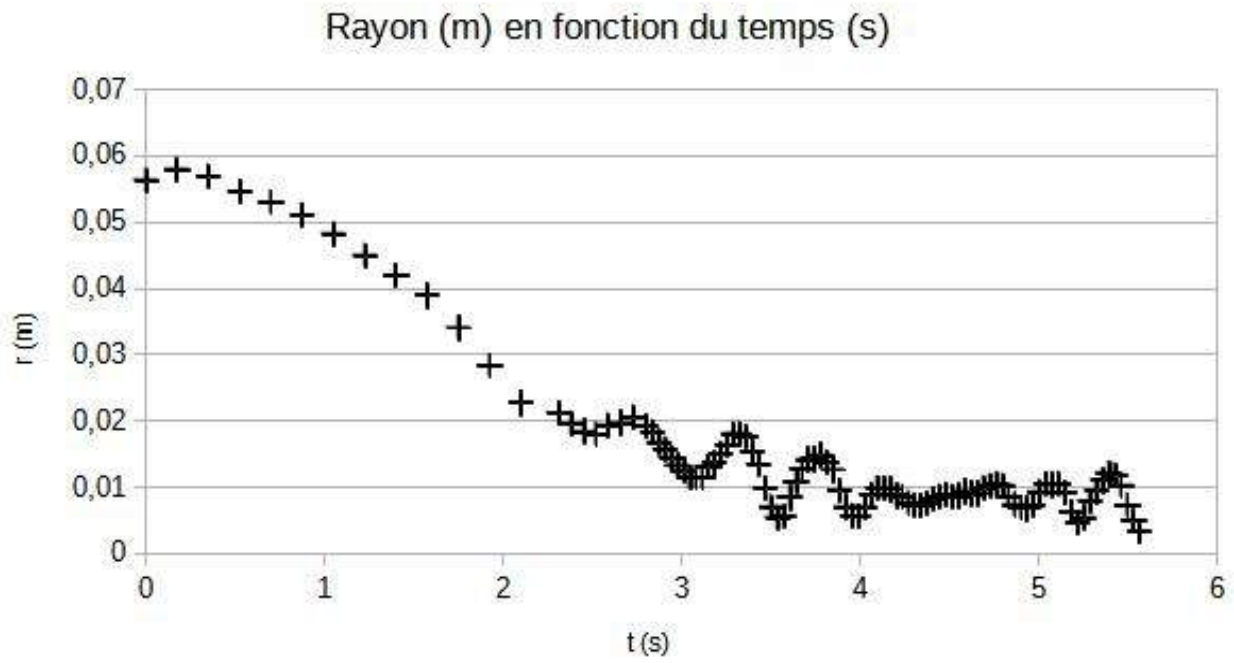

Figure 13

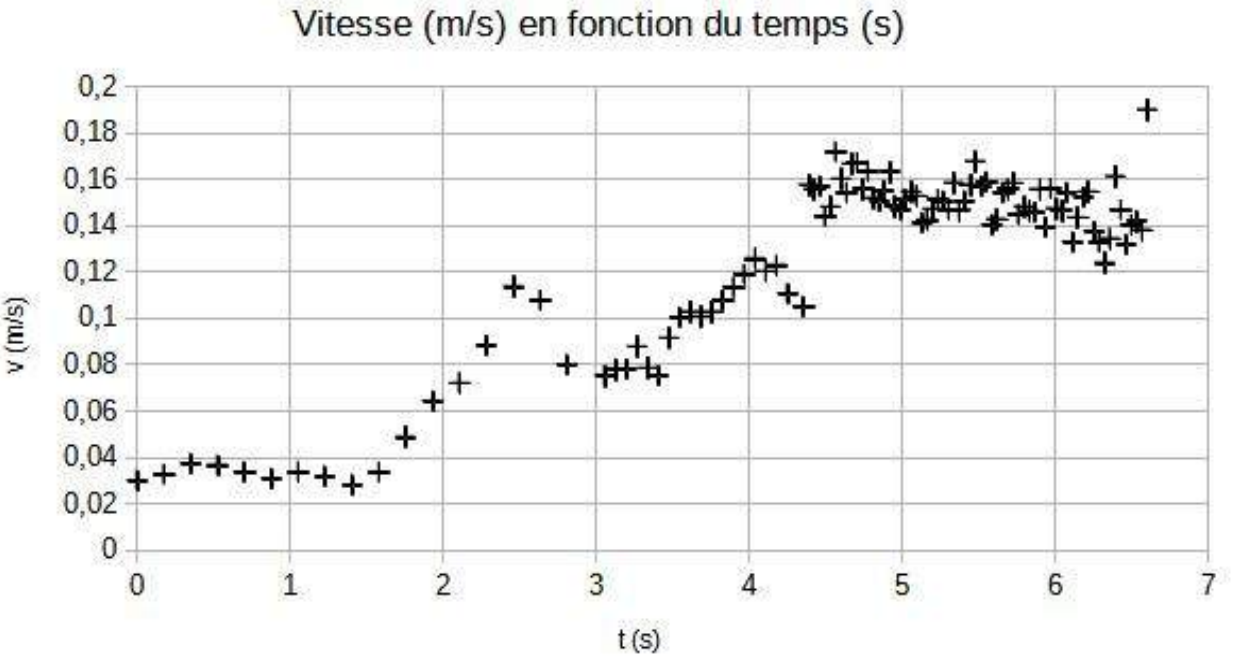

Figure 14

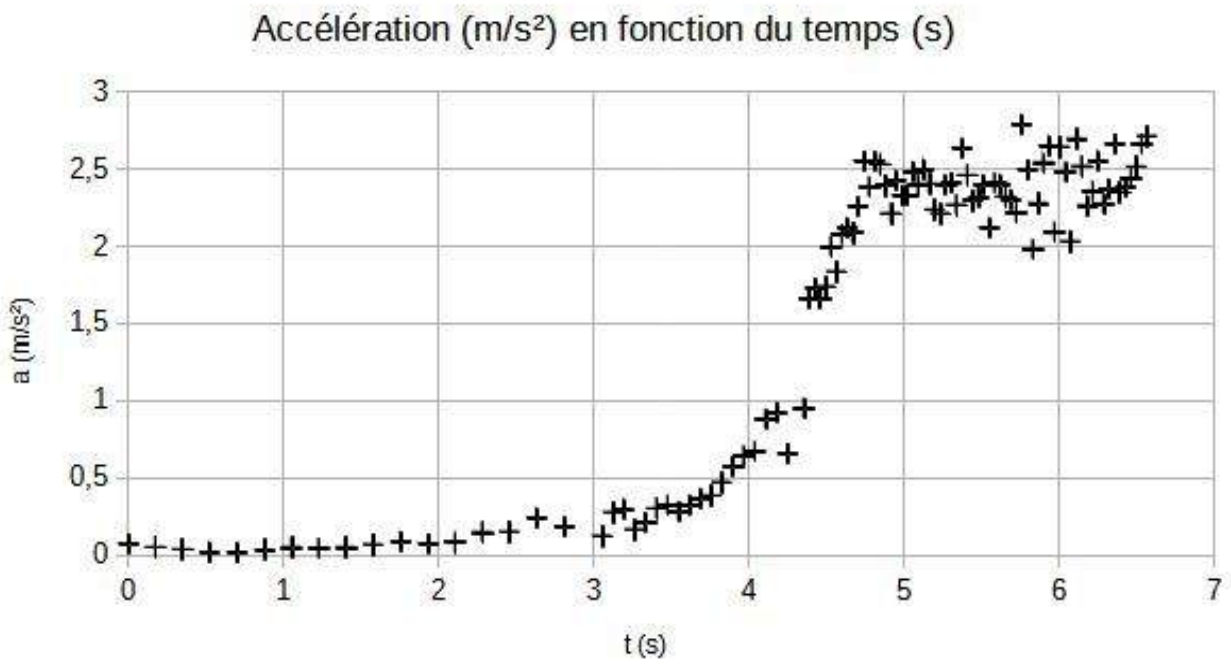

Figure 15

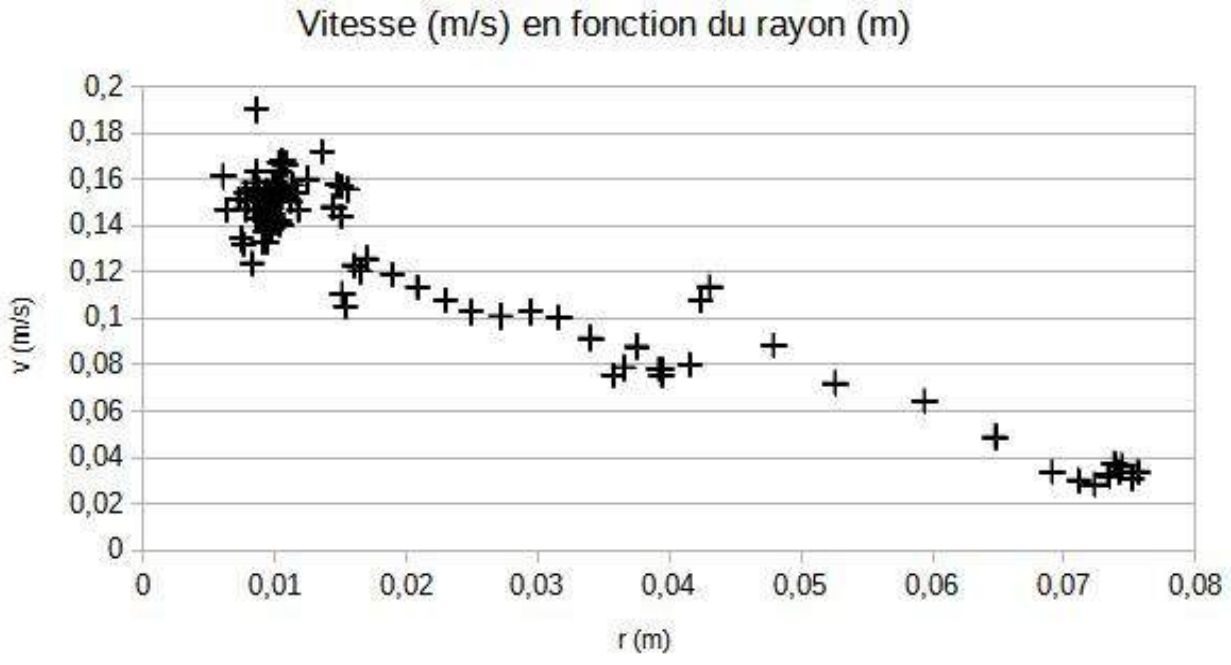

Figure 16

Annexe 4 : Code python

```python
import numpy as np
import scipy.linalg
from mpl_toolkits.mplot3d import Axes3D
import matplotlib.pyplot as plt
# On rentre les coordonnées x,y,z de nos points ( vitesse du barreau,distance de lâché, temps d'aspiration)
v = []
d = []
z = []
data = np.c_[v,d,z]
mn = np.min(data, axis=0)
mx = np.max(data, axis=0)
X,Y = np.meshgrid(np.linspace(mn[0], mx[0], 20), np.linspace(mn[1], mx[1], 20))
XX = X.flatten()
YY = Y.flatten()

order = int(input("rentrer le degré choisi : ")) # 1 :linéaire , 2 : quadratique , 3 : cubique
if order == 1:
    # Approximation linéaire
    A = np.c_[data[:,0], data[:,1], np.ones(data.shape[0])]
    C,_,_,_ = scipy.linalg.lstsq(A, data[:,2]) # coefficients

    # Créer la surface
    Z = C[0]*X + C[1]*Y + C[2]


    # Coefficient de Pearson sous forme matricielle
    z_vals = np.array([]) # rentrer valeur du temps d'aspiration expérimental
    t_vals = Z
    my_rho = np.corrcoef(z_vals, t_vals)
    somme=0
    k=np.array(np.corrcoef(z_vals, t_vals))
    taille_matrice =(int(np.size(k)**0.5))
    i=0
    while i<taille_matrice:
        j=0
        while j<taille_matrice:
            somme=somme+my_rho[i,j]
            j=j+1
        i=i+1
    somme=somme-taille_matrice
    somme=somme/(taille_matrice**2-taille_matrice)
    print(somme) #afficher coefficient de Pearson
    print(C[0],"*v+",C[1],"*d+",C[2]) # afficher l'equation
```

```python
elif order == 2:
    # Approximation quadratique
    A = np.c_[np.ones(data.shape[0]), data[:,:2], np.prod(data[:,:2], axis=1), data[:,:2]**2]
    C,_,_,_ = scipy.linalg.lstsq(A, data[:,2])

    # Créer la surface
    Z = np.dot(np.c_[np.ones(XX.shape), XX, YY, XX*YY, XX**2, YY**2], C).reshape(X.shape)

    #Coefficient de Pearson sous forme matricielle
    z_vals = np.array([])# rentrer valeur du temps d'aspiration expérimental
    t_vals = Z
    my_rho = np.corrcoef(z_vals, t_vals)
    somme=0
    k=np.array(np.corrcoef(z_vals, t_vals))
    taille_matrice =(int(np.size(k)**0.5))
    i=0
    while i<taille_matrice:
        j=0
        while j<taille_matrice:
            somme=somme+my_rho[i,j]
            j=j+1
        i=i+1
    somme=somme-taille_matrice
    somme=somme/(taille_matrice**2-taille_matrice)
    print(somme)# Afficher coefficient de Pearson
    print(C[4],"*v**2+",C[5],"*d**2+",C[3],"*v*d+",C[1],"*v+",C[2],"*d+",C[0]) #afficher l'equation

elif order == 3:
   # Approximation cubique
    A = np.c_[np.ones(data.shape[0]), data[:,:2], data[:,0]**2, np.prod(data[:,:2], axis=1), \
              data[:,1]**2, data[:,0]**3, np.prod(np.c_[data[:,0]**2,data[:,1]],axis=1), \
              np.prod(np.c_[data[:,0],data[:,1]**2],axis=1), data[:,2]**3]
    C,_,_,_ = scipy.linalg.lstsq(A, data[:,2])
    # créer la surface
    Z = np.dot(np.c_[np.ones(XX.shape), XX, YY, XX**2, XX*YY, YY**2, XX**3, XX**2*YY, XX*YY**2, YY**3], C).reshape(X.shape)

    #Coefficient de Pearson sous forme matricielle
    z_vals = np.array([])# rentrer valeur du temps d'aspiration expérimental
    t_vals = Z
    my_rho = np.corrcoef(z_vals, t_vals)
    somme=0
    k=np.array(np.corrcoef(z_vals, t_vals))
    taille_matrice =(int(np.size(k)**0.5))
    i=0
    while i<taille_matrice:
       j=0
       while j<taille_matrice:
           somme=somme+my_rho[i,j]
           j=j+1
       i=i+1
    somme=somme-taille_matrice
    somme=somme/(taille_matrice**2-taille_matrice)
    print(somme)#Afficher coefficient de Pearson
    print(C[6],"*v**3+",C[9],"*d**3+",C[7],"*v**2*d+",C[8],"*v*d**2+",C[3],"*v**2+",C[5],"*d**2+",C[4],"*v*d+",C[1],"*v+",C[2],"*d+",C[0]
```

```python
# Afficher les points et la surface de tendance.
fig = plt.figure()
ax = fig.gca(projection='3d')
ax.plot_surface(X, Y, Z, rstride=1, cstride=1, alpha=0.2)
ax.scatter(data[:,0], data[:,1], data[:,2], c='r', s=50)
plt.xlabel('Vitesse(trs/min)')
plt.ylabel('Distance(cm)')
ax.set_zlabel('tau(s)')
ax.axis('auto')
ax.axis('tight')
plt.show()
```

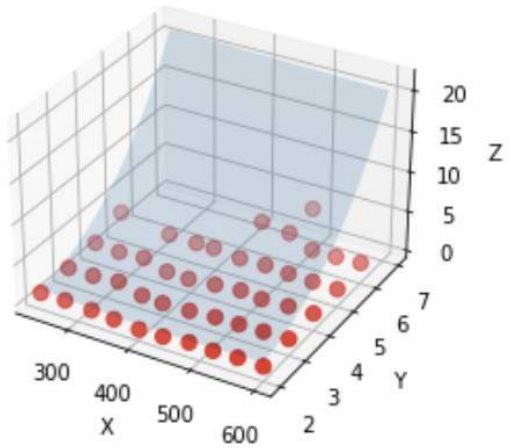

Figure 17: Exemple d'une mauvaise approximation du temps d'aspiration

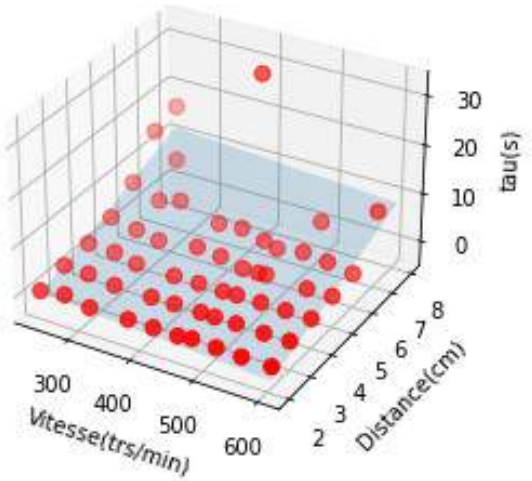

Figure 18: Courbe pour η = 0,2197 Pa.s

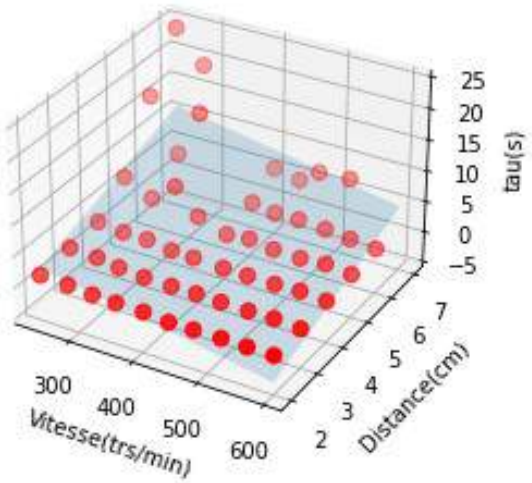

Figure 19: Courbe pour η = 0,1650 Pa.s

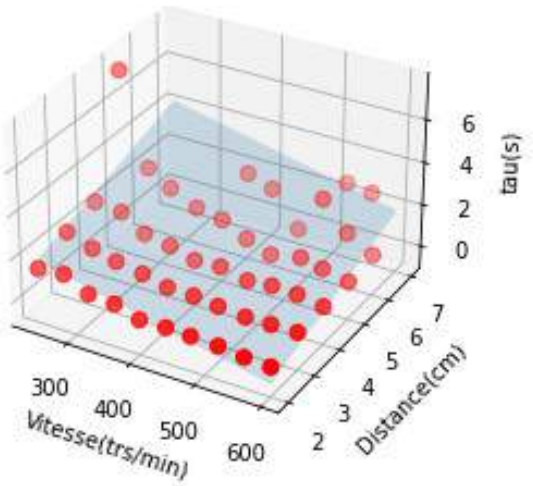

Figure 20: Courbe pour η = 0,1104 Pa.s

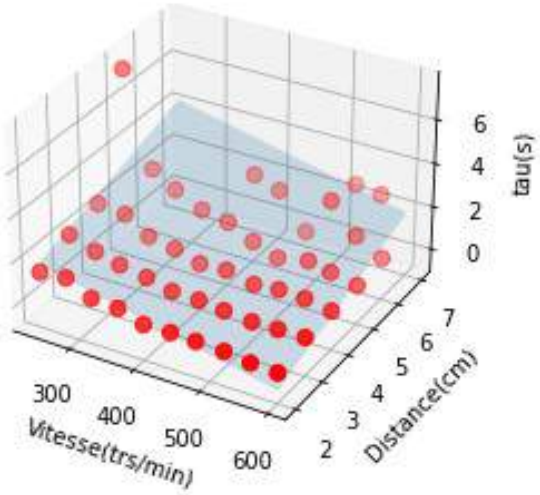

Figure 21: Courbe pour η = 0,05568 Pa.s

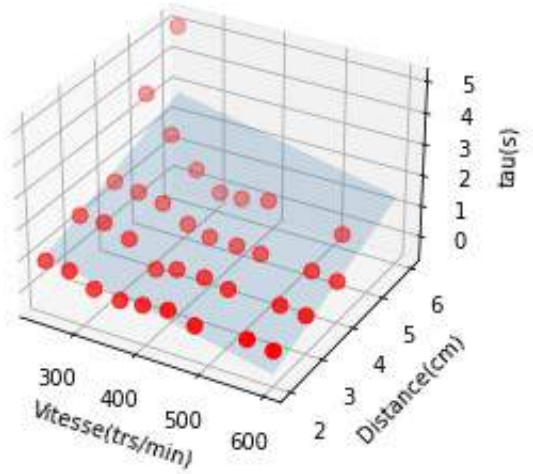

Figure 22: Courbe pour η = 0,001005 Pa.s