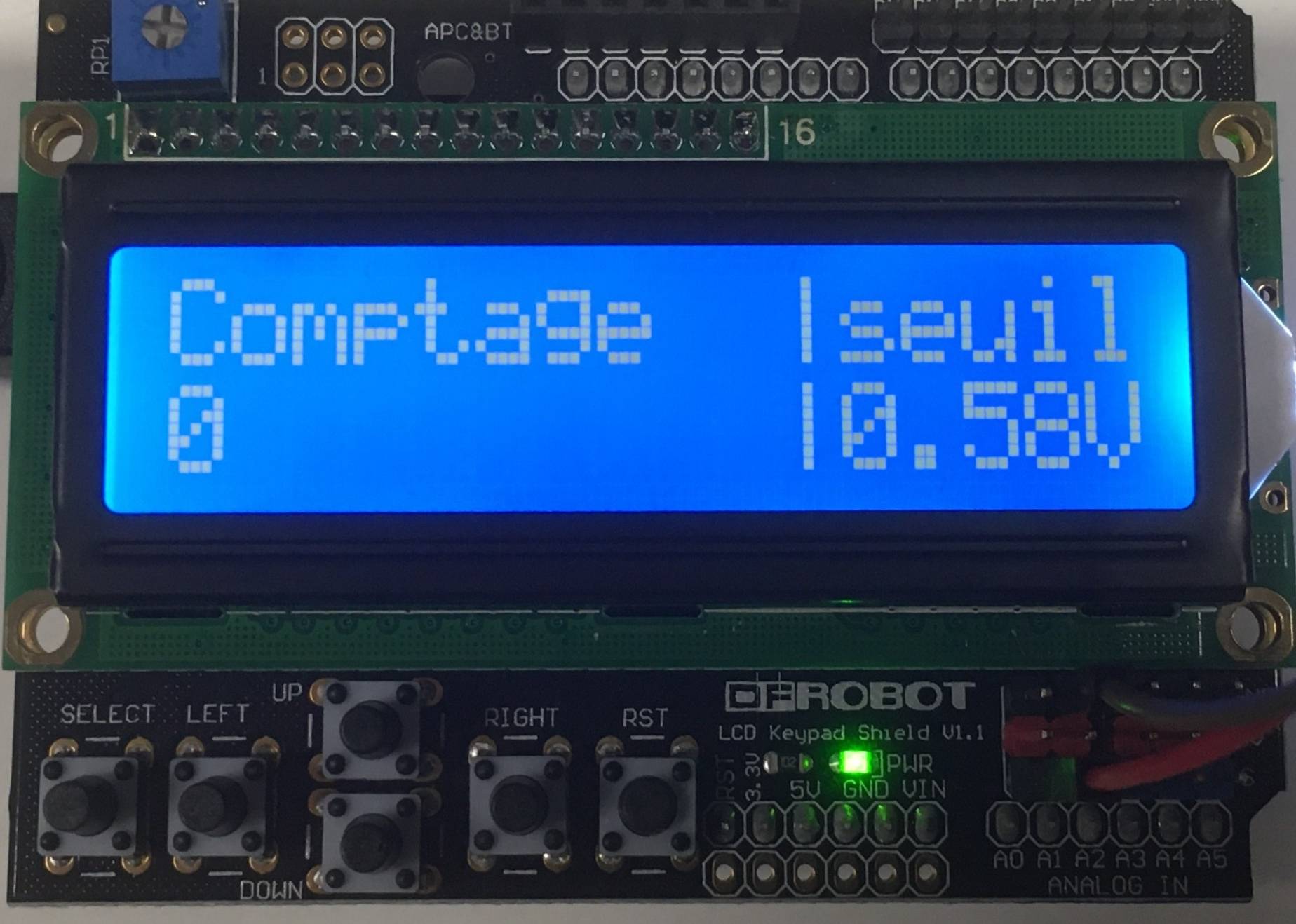
**COMPTEUR DE DEFILEMENT DE FRANGES/ANNEAUX**

**MICHELSON ARDUINO**

Nous utilisons une carte Arduino avec un shield LCD DFROBOT.



**LEFT** : Pour remettre le compteur à zéro

**RIGHT** : consultation de la tension mesurée (utile pour estimer la valeur du seuil

**UP/DOWN** : réglage du seuil à zéro

Un photodétecteur (« cible » DIDALAB par exemple) est connecté à la première entrée analogique (entrée A1 car AO est déjà utilisée pour récupérer les valeurs envoyées par les boutons du curseur du shield).

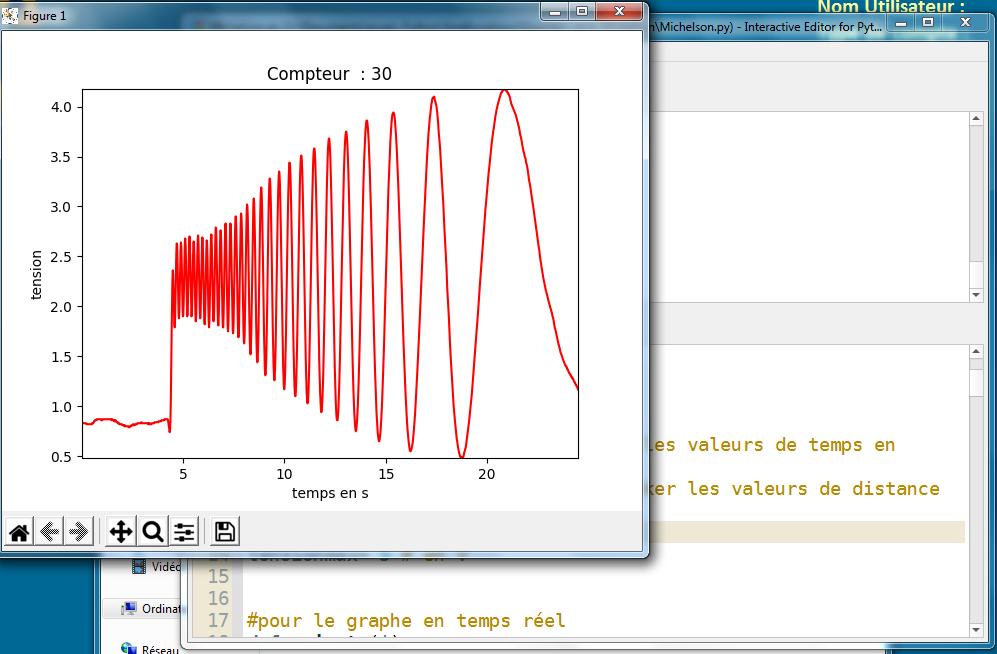
RIGHT : consultation de la tension mesurée (utile pour estimer la valeur du seuil

UP/DOWN : réglage du seuil à zéro

LEFT : Pour remettre le compteur à zéro

Le réglage du seuil de détection dépend du photodétecteur et de la luminosité. Idéalement, il faudrait qu’il corresponde à la valeur moyenne de la tension mesurée lors de l’alternance des franges ou anneaux.

Pour le régler efficacement, on peut consulter la tension mesurée avec le bouton **RIGHT**, ou faire une acquisition avec la carte Sysam.



Il est aussi possible de lancer le script Python

**Michelson.py** (avec un IDLE comme Pyzo par

exemple) pour une acquisition en temps réel.

**Code Arduino téléversé sur la carte :**

|  |
| --- |
| #include <EEPROM.h>  #include <LiquidCrystal.h>  LiquidCrystal lcd(8,9, 4, 5, 6, 7);  int sensor = A1; // broche pour détection du capteur  int etatSensor ; // état du capteur (haut ou bas)  int valeur\_bouton;  int seuil ;  float seuil\_tension ;  int mesure;  float mesure\_tension;  bool etat\_old= false ; //  bool etat\_new = false; // les états vont changer à chaque chaque modiication de la valeu lue par le capteur (haut/5V ou bas/0V)  int compt=0; // comptage initialisé à 0  long temps; //mesure du temps pour l'acquisition  void setup()  {    Serial.begin(9600); // pour le moniteur série  lcd.begin(16,2);  int seuil;    temps = millis();  }    void loop()  {  valeur\_bouton = analogRead(A0);  Serial.println(valeur\_bouton);  if (valeur\_bouton ==205 or valeur\_bouton ==203) {  seuil+=1;  EEPROM.put(0, seuil);  delay(10);  }  if (valeur\_bouton == 405 or valeur\_bouton == 402) {  seuil-=1;  EEPROM.put(0, seuil);  delay(10);  }  EEPROM.get(0,seuil);    if (valeur\_bouton == 619 or valeur\_bouton == 622) {  lcd.clear();  compt=0;  delay(100);  }  temps = millis();  mesure = analogRead(sensor);  mesure\_tension = mesure\*5.0/1023;  seuil\_tension = seuil \*5.0/1023;  if (valeur\_bouton == 0) {  lcd.clear();  lcd.setCursor(0,0);  lcd.print(" Valeur tension ");  lcd.setCursor(5,1);  lcd.print(mesure\_tension);  lcd.print(" V");  delay(1000);  lcd.clear();  }  lcd.setCursor(0,0);  lcd.print("Comptage |seuil");  lcd.setCursor(0,1);  lcd.print(compt/2);  lcd.setCursor(10,1);  lcd.print("|");  lcd.setCursor(11,1);  lcd.print(seuil\_tension);  lcd.setCursor(15,1);  lcd.print("V");  if (mesure\_tension> seuil\_tension){  etat\_new = true;  }  else {  etat\_new =false;  }  if (etat\_old != etat\_new) {  etat\_old = etat\_new;  compt = compt + 1;  // Serial.print(" comptage ");  // Serial.println(compt);  //lcd.clear();  }    Serial.print(temps);  Serial.print ("\t");  Serial.println(mesure\_tension);  //Serial.println(valeur\_bouton);  delay(10);  } |

**Script Python pour acquisition en temps réel :**

|  |
| --- |
| #importation des modules  import serial  import serial.tools.list\_ports # pour la communication avec le port série  import matplotlib.pyplot as plt # pour le tracé de graphe  from matplotlib import animation # pour la figure animée  import time # gestion du temps  #initialisation des listes  liste\_temps=[] # liste pour stocker les valeurs de temps en partant de t=0  liste\_tension = [] # liste pour stocker les valeurs de distance  t\_acquisition = 1000  tensionmax= 5 # en V  #pour le graphe en temps réel  def animate(i):  line1 = Data.readline()  print (line1)  # on retire les caractères d'espacement en début et fin de chaîne  listeDonnees = line1.strip()  # on sépare les informations reçues séparées par les espaces et on stocke ces informations dans une liste pour chacune de lignes  listeDonnees = line1.split()  print (listeDonnees)  if len(listeDonnees) == 2 : # parfois des lignes de données vides peuvent être envoyées, il faut les "écarter"  try :  tension = float(listeDonnees[1].decode()) # après consulation des données, nous choisissons le 4ème élément de listeDonnees  temps = (float(listeDonnees[0].decode()))/1000.0 # après consulation des données, nous choisissons le 2ème élément de listeDonnees  while temps <= t\_acquisition:  liste\_tension.append(tension)  print("tension = %f"%(tension)) # affichage de la valeur de la distance  liste\_temps.append(temps)  print("temps mesuré = %f"%(temps), " s") # affichage de la valeur du temps absolu  line.set\_data(liste\_temps,liste\_tension)  return line,  except:  pass  # Fonction pour la récupération des données série venant de la carte Arduino  def recup\_port\_Arduino() :  ports = list(serial.tools.list\_ports.comports())  for p in ports:  if 'Arduino' in p.description :  mData = serial.Serial(p.device,9600)  print(mData.is\_open) # Affiche et vérifie que le port est ouvert  print(mData.name) # Affiche le nom du port  return mData  def moyenne(liste):  somme=0  for i in range (len (liste)) :  somme += liste[i]  return somme/(len(liste))  def comptage (liste):  moy = moyenne (liste)  etat\_old = False  etat\_new = False  compt = 0  for i in range (len(liste)):  if (liste[i]> moy) :  etat\_new = True  else :  etat\_new = False  if etat\_old != etat\_new :  etat\_old = etat\_new  compt = compt + 1  return compt/2  Data =recup\_port\_Arduino() #récupération des données  # Création figure  fig=plt.figure()  line, = plt.plot([],[])  plt.xlim(0, t\_acquisition)  plt.ylim(0,tensionmax)  plt.xlabel('temps en s')  plt.ylabel('tension')  plt.grid()  #Animation  ani = animation.FuncAnimation(fig, animate, frames=20000, interval=20,repeat=False)  plt.show()  plt.close(fig)  Data.close() # pour arrêter la lecture des données série  #comptage en parcourant les listes  compteur = int(comptage(liste\_tension))  texte = "Compteur : " +str(compteur)  plt.title(texte) # titre du graphique  plt.plot(liste\_temps,liste\_tension, color ='r')  plt.xlabel('temps en s')  plt.ylabel('tension')  plt.xlim (min(liste\_temps),max(liste\_temps)) #limtes pour les axes avec les valeurs extrêmes de I et de U  plt.ylim(min(liste\_tension),max(liste\_tension))  plt.show() #afficher le graphique (ne rien mettre dans la parenthèse)  #Ecriture dans un fichier txt  lines=['t\ttension\n'] #première ligne du fichier txt  for i in range (len (liste\_tension)):  line = str(liste\_temps[i]) +'\t'+ str(liste\_tension[i])+'\n'  lines.append(line)  fichier = open('U:\Documents\essais Python\Améliorations\Données série Michelson\data.txt', 'w').writelines(lines) #création d'un nouveau fichier texte |