# 8 ANNEXES

## 8.1 Arduino RGB

```
/* Broches */
const int LED_R = 8;
const int LED_G = 9;
const int LED_B = 10;

void setup(){
  // Initialise les broches
  pinMode(LED_R, OUTPUT);
  pinMode(LED_G, OUTPUT);
  pinMode(LED_B, OUTPUT);
}

void loop(){
  // 1 seconde par couleur
  digitalWrite(LED_R, HIGH);
  delay(1000);
  digitalWrite(LED_R, LOW);
  digitalWrite(LED_G, HIGH);
  delay(1000);
  digitalWrite(LED_G, LOW);
  digitalWrite(LED_B, HIGH);
  delay(1000);
  digitalWrite(LED_B, LOW);
}
```

## 8.2 Arduino JCMB

```
/* Broches */
const int LED_R = 8;
const int LED_G = 9;
const int LED_B = 10;

void setup(){
  // Initialise les broches
  pinMode(LED_R, OUTPUT);
  pinMode(LED_G, OUTPUT);
  pinMode(LED_B, OUTPUT);
}

void loop(){
  digitalWrite(LED_R, HIGH);
  digitalWrite(LED_G, HIGH);
  delay(1000);
  digitalWrite(LED_R, LOW);
  digitalWrite(LED_B, HIGH);
  delay(1000);
  digitalWrite(LED_G, LOW);
  digitalWrite(LED_R, HIGH);
  delay(1000);
  digitalWrite(LED_G, HIGH);
  delay(1000);
  digitalWrite(LED_R, LOW);
  digitalWrite(LED_G, LOW);
  digitalWrite(LED_B, LOW);}
```

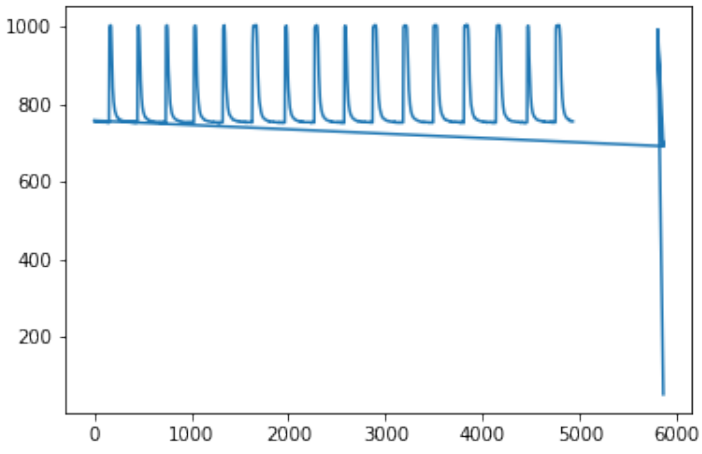
## 8.3 Arduino intensité d'un canal

```
/* Broches */
const int LED_R = 8;
const int LED_G = 9;
const int LED_B = 10;

void setup(){
  // Initialise les broches
  pinMode(LED_R, OUTPUT);
  pinMode(LED_G, OUTPUT);
  pinMode(LED_B, OUTPUT);
}

void loop(){
  analogWrite(LED_G, 0);
  delay(1000);
  analogWrite(LED_G, 75);
  delay(1000);
  analogWrite(LED_G, 125);
  delay(1000);
  analogWrite(LED_G, 250);
  delay(1000);
}
```

## 8.4 La photorésistance

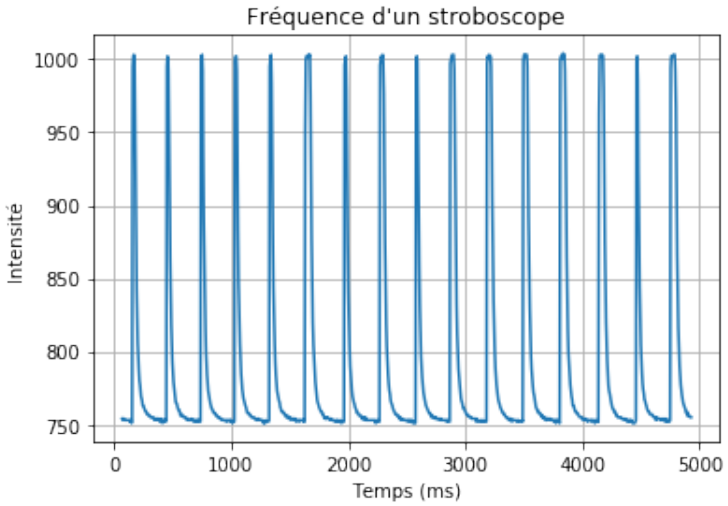
### Le code arduino

```
const int broche = A0;

void setup() {
  Serial.begin(19200);
}

void loop() {
  int valeur = analogRead(broche);
  Serial.print(valeur);
  Serial.print("\t");
  Serial.println(millis());
}
```



On ne conserve que les valeurs exploitables. On rappelle qu'il est possible de n'utiliser qu'une partie des valeurs d'une liste, soit L une liste alors on écrit : `L[debut, fin]`. Dans notre exemple on peut essayer de démarrer à partir de la valeur d'index 50 en gardant toutes les valeurs suivantes, ce qui donne : `temps[50: ]`. À vous d'ajuster en fonction de ce que vous obtenez.

```
1 # attention les deux listes doivent contenir le même nombre de valeurs.
2 plt.plot(temps[50:], mesure[50:])
3
4 plt.title("Fréquence d'un stroboscope")
5 plt.ylabel('Intensité')
6 plt.xlabel('Temps (ms)')
7 plt.grid()
8 plt.show()
```



### 8.4.1 Le bouton poussoir

```
// Etat en cours de l'automate
int etat;
// Etat à mémoriser
int oldEtat;

//Les états possibles de l'automate
const int WAIT = 2;
const int START = 1;
const int STOP  = 0;

// Les broches utilisées
//capteur
const int broche = A0;
//bouton poussoir
const int BP = 3;

void setup() {
  //initialisation des variables
  oldEtat = LOW;
  etat = WAIT;
  //config E/S
  pinMode(BP, INPUT);
  //liaison série
  Serial.begin(19200);
}

void loop() {
  //Lecture du bouton
  int etatBP = digitalRead(BP);
  //gestion des états
  if(oldEtat == LOW && etatBP == HIGH){
    if (etat == WAIT)
    {
      etat = START;
    }
    else if (etat == STOP)
    {
      etat = START;
    }
    else if (etat == START)
    {
      etat = STOP;
    }
  }

  //Traitement des états
  if(etat == START){
    int valeur = analogRead(broche);
    Serial.print(valeur);
    Serial.print("\t");
    Serial.println(millis());
  }
  else if(etat == STOP)
    Serial.println("-1\t -1");

  oldEtat = etatBP;
  delay(10);
}
```

## 8.5 Le projet : vitesse du son

### 8.5.1 Le code Arduino

```
// Déclaration des variables globales : broches
int trigg = 8;
int echo = 9;

void setup() {
  pinMode(trigg, OUTPUT);          // Configuration des broches
  digitalWrite(trigg, LOW);        // La broche TRIGGER doit être à LOW au repos
  pinMode(echo, INPUT);            // La broche ECHO en entrée
  Serial.begin(9600);              // Démarrage de la liaison série
}

void loop() {

  digitalWrite(trigg, HIGH);       // Lance une mesure de distance en envoyant
  delayMicroseconds(10);           // Une impulsion HIGH de 10 microsecondes
  digitalWrite(trigg, LOW);        // Fin d'émission
  int temps = pulseIn(echo, HIGH); // Mesure temps émission-reception

  Serial.print(temps);
  Serial.print("\t");              // on ajoute une tabulation et la
  Serial.println("-1");            // la valeur -1
  delay(500);
}
```

Dans cet exemple seule la valeur temps nous intéresse. Mais pour uniformiser le code nous ajoutons une tabulation et la valeur $-1$. Cela permet de réutiliser les codes Python précédents. En effet coté Python on attend toujours deux informations (deux grandeurs physiques). C'est un choix de programmation que nous avons fait dès le départ mais qui est cohérent avec les besoins que nous avons en physique-chimie qui se traduit souvent par l'étude d'une grandeur en fonction du temps. Il suffira d'ignorer la lecture de la valeur $-1$ coté Python. En informatique il est courant d'utiliser la valeur $-1$ pour indiquer soit une erreur soit une valeur à ignorer.

### 8.5.2 Le code Python

Penser à écrire une fonction pour calculer le temps moyen sur un nombre entier $n$ de valeurs renvoyées par le capteur ultrason.

```
def temps_moyen(n, serial_port):
    """
    Cette fonction calcule une moyenne des temps ultrasons
    reçus sur n valeurs

    n           -> <int>    : Nombre de valeurs à lire
    serial_port -> <serial> : Port série
    """
    i = 0
    t_somme = 0
    serial_port.flushInput()
    while i < n:
        val = serial_port.readline().split()
        try:
            t = float(val[0])       # lecture temps ultrason
            t_somme += t            # la somme des temps ultrasons
            i = i + 1               # ajoute 1 à la variable comptant les mesures
        except:
            pass
    return t_somme/n
```

```
# les imports
import serial
import matplotlib.pyplot as plt
```

```
# ouverture du port série et synchronisation des données entre arduino et Python.
serial_port = serial.Serial( port = "/dev/ttyACM0", baudrate =9600)

temps = []      # une liste pour les mesures de temps
distances = []  # une liste pour les mesures de distance
dist = 0        # juste pour entrer dans la boucle
nb_t = 10       # le nombre de mesure temps

while dist != -1:
    dist = float(input("Entrez votre mesure : "))
    if dist != -1:
        distances.append(dist)
        tm = temps_moyen(nb_t, serial_port)
        temps.append(tm)

# fermeture du port série
serial_port.close()
```

### 8.5.3 Comment faire les mesures ?

- Téléverser le programme Arduino dans la mémoire de la carte.
- Ouvrir un moniteur série et vérifier que les mesures de temps s'affichent. Faire varier la distance entre l'objet et le capteur HC-SR04 pour observer que les valeurs temps augmentent si la distance augmente.
- Valider les cellules contenant le code Python concernant cet exercice.
- Placer correctement votre capteur HC-SR04 à une distance déterminée de votre objet.
- Normalement vous devez avoir la possibilité de saisir cette distance dans une boite de dialogue qui c'est ouverte sur le Notebook.

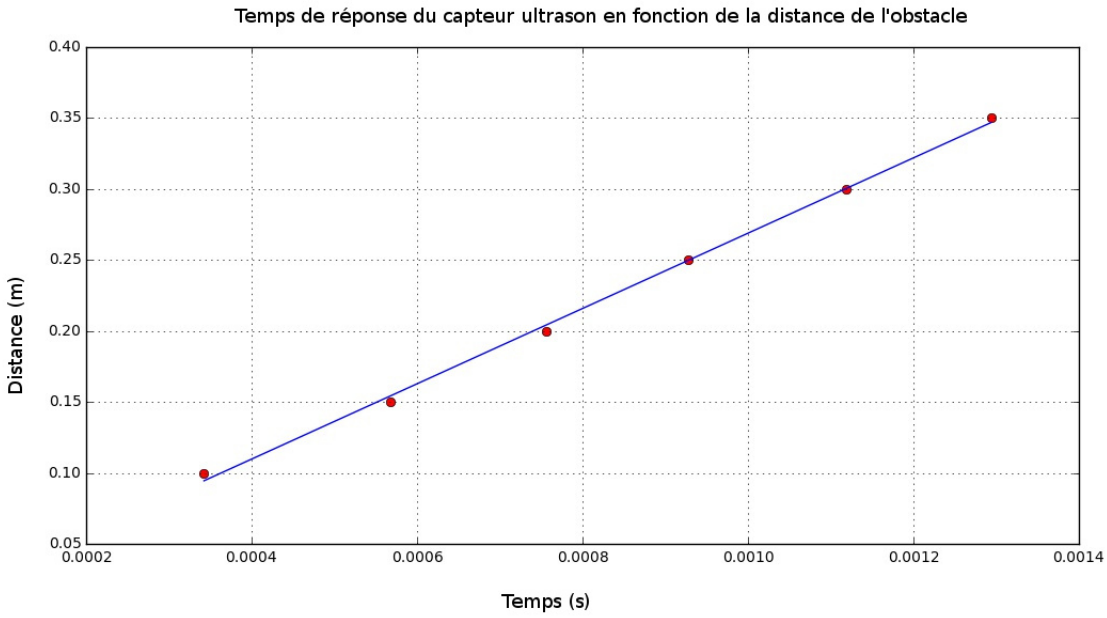
```
# fermeture du port série
serial_port.close()
```

```
Entrez votre mesure : 0.1
Entrez votre mesure : 0.15
Entrez votre mesure : 0.20
Entrez votre mesure : 0.25

Entrez votre mesure :
```

- Valider avec la touche *Enter*, modifier la distance capteur-objet, entrer la valeur, etc...

L'affichage du graphique : distances en fonction du temps, je vous laisse ajouter les commentaires : title, xlabel, ylabel

```
plt.figure(figsize=(12,6))
plt.plot(mesures, distances, 'ro')
plt.grid()
plt.show()
```




## 8.6 La chute libre

### 8.6.1 Le code Arduino

```
1  // Déclaration des variables globales : broches
2  int trigg = 8;
3  int echo = 9;
4
5
6  void setup() {
7
8    pinMode(trigg, OUTPUT);           // Configuration des broches
9    digitalWrite(trigg, LOW);         // La broche TRIGGER doit être à LOW au repos
10   pinMode(echo, INPUT);             // La broche ECHO en entrée
11
12   Serial.begin(19200);              // Démarrage de la liaison série
13 }
14
15 void loop() {
16
17   digitalWrite(trigg, HIGH);        // Lance une mesure de distance en envoyant
18   delayMicroseconds(10);            // Une impulsion HIGH de 10 microsecondes
19   digitalWrite(trigg, LOW);         // Fin d'émission
20
21   int temps_echo = pulseIn(echo, HIGH); // Mesure temps émission-reception
22
23   Serial.print(temps_echo);
24   Serial.print("\t");
25   Serial.println(millis());
26   delay(1);
27 }
```

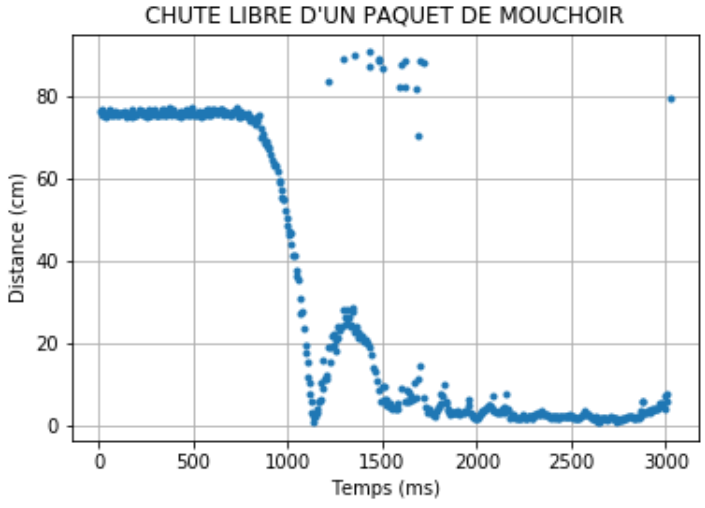
### 8.6.2 Le code Python

Les mesures ont été faites avec un paquet de mouchoirs en papier accroché à une ficelle d'environ 80 cm. J'ai lancé le code python puis j'ai compté jusqu'à deux avant de lacher le paquet bien à la verticale du capteur ultrason.

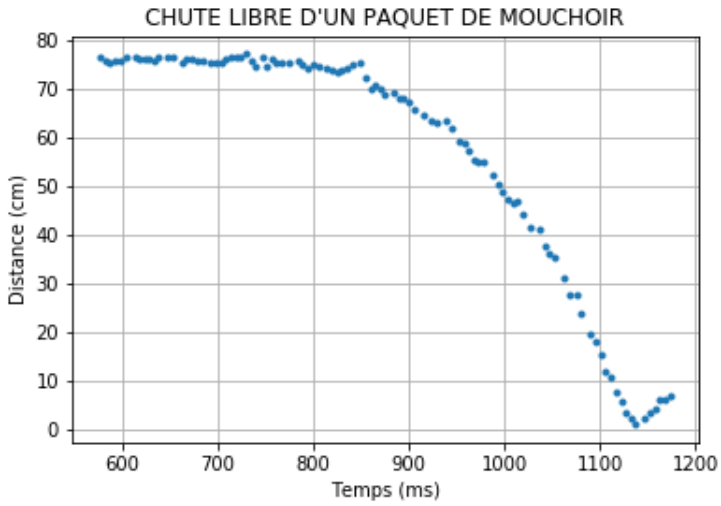
```
# les imports
import serial
import time
import matplotlib.pyplot as plt
```

```
# ouverture du port série
serial_port = serial.Serial( port = "/dev/ttyACM1", baudrate =19200)
serial_port.setDTR(False)
time.sleep(0.1)
serial_port.setDTR(True)
serial_port.flushInput()

# les mesures
mesure_dist = []
temps = []
duree = 3000                                    # durée d'acquisition
end = False

while end == False or temps[-1] - temps[0] <= duree:
    val = serial_port.readline().split()        # lecture des données
    try:
        d = float(val[0])/58                    # distance en cm
        t = float(val[1])                       # temps écoulé en ms
        if d > 1 and d < 100:                   # filtrage des valeurs aberrantes
            mesure_dist.append(d)               # entre 1cm et 1m
            temps.append(t)
            end = True
    except:
        pass
# fermeture du port série
serial_port.close()
```

```
plt.plot(temps, mesure_dist, ".")
plt.title("CHUTE LIBRE D'UN PAQUET DE MOUCHOIR")
plt.ylabel('Distance (cm)')
plt.xlabel('Temps (ms)')
plt.grid()
plt.show()
```

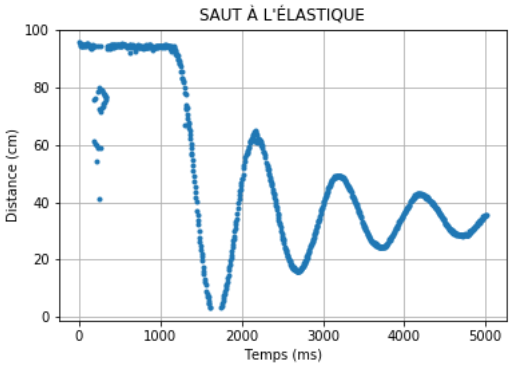


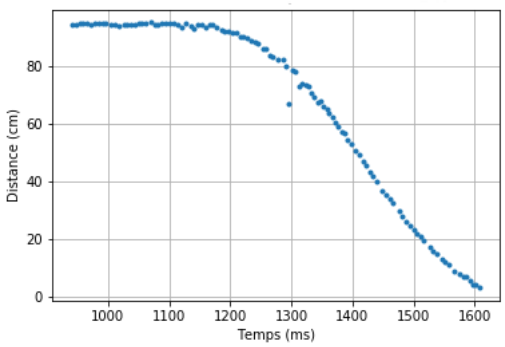
```
plt.plot(temps[100:200], mesure_dist[100:200], ".")
plt.title("CHUTE LIBRE D'UN PAQUET DE MOUCHOIR")
plt.ylabel('Distance (cm)')
plt.xlabel('Temps (ms)')
plt.grid()
plt.show()
```




## 8.7 Le saut à l'élastique

Avec un capteur ultrason il est possible il également possible d'étudier les différentes phases d'un saut à l'élastique.




Les codes Arduino et Python pour obtenir ces résultats sont exactement les mêmes que pour la chute libre. Il peut donc être intéressant de créer une fonction Python que les élèves pourraient utiliser à partir d'un module.

```
# ouverture du port série
serial_port = serial.Serial( port = "/dev/ttyACM0", baudrate =19200)
serial_port.setDTR(False)
time.sleep(0.1)
serial_port.setDTR(True)
serial_port.flushInput()

temps, mesure_dist = distance_ultrason(5000, serial_port)

# fermeture du port série
serial_port.close()
```

```python
def distance_ultrason(duree, serial_port, inf = 1, sup = 100):
    """
    Renvoie respectivement une liste temps (dates d'acquisition)
    et une liste distance en (cm) mesurée par le capteur ultrason
    durant une durée donnée et dans un intervalle de distance
    fixée par défaut entre 1cm et 1m

    duree       -> <float>  : durée de l'acquisition en (ms)
    serial_port -> <serial> : port série ouvert à la communication
    inf         -> <float>  : distance minimum d'acquisition en (cm)
    sup         -> <float>  : distance maximum d'acquisition en (cm)
    """
    mesure = []
    temps = []
    end = False
    while end == False or temps[-1] - temps[0] <= duree:
        val = serial_port.readline().split()    # lecture des données
        try:
            d = float(val[1])/58                # distance en cm
            t = float(val[0])                   # temps écoulé en ms
            if d > inf and d < sup:             # filtrage des valeurs aberrantes
                mesure.append(d)
                temps.append(t)
                end = True
        except:
            pass
    return temps, mesure
```