Tests de traction et loi de Hooke

Prenez en note tout élément pouvant figurer dans un compte-rendu de TP :
mesures, calculs d'incertitude, observations (schémas) et interprétations, méthodes expérimentales...

Objectifs :

— Vérifier expérimentalement la loi de Hooke de l'élasticité sur un fil et sa limite de validité.

— Mettre en évidence une déformation plastique et un seuil de rupture.

Capacités expérimentales exigibles :

- Mettre en œuvre un protocole expérimental permettant d'étudier une loi de force.
- Mettre en œuvre un microcontrôleur lors d'un test de traction.

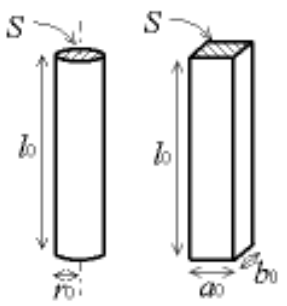
I. Introduction

Déformation élastique ou plastique, rupture

L'*élasticité* est la propriété d'un matériau solide à retrouver sa forme d'origine après avoir été déformé par des forces qui lui sont appliquées. Un matériau élastique retrouve sa forme et sa taille initiales quand ces forces ne s'exercent plus, jusqu'à une certaine limite de la valeur de ces forces.
L'*élasticité linéaire* concerne les petites déformations proportionnelles à la sollicitation. Dans cette gamme, l'allongement est proportionnel à la force dans le cas d'un étirement (ou d'une compression), selon le *module de Young*, et l'angle est proportionnel au « couple » [1] dans le cas d'une torsion, selon le *module de Coulomb*.
Pour certains matériaux (cahoutchoucs élastiques), l'élasticité peut parfois devenir *non linéaire* aux grandes déformations.
Pour d'autres (la plupart des matériaux cristallins notamment), la *fracture* intervient après une phase de *déformation plastique* plus ou moins longue. La théorie de la *plasticité* traite des déformations irréversibles indépendantes du temps. Certains matériaux, dits *fragiles*, cassent dans le mode de déformation élastique si la sollicitation est trop forte. Pour les matériaux dits *ductiles*, une augmentation suffisante de la sollicitation entraîne une déformation définitive sans rupture ; à l'arrêt de la sollicitation, la pièce reste déformée. C'est par exemple le cas d'une petite cuillère qui a été tordue : on ne pourra jamais la redresser pour lui faire reprendre sa forme initiale.
Par ailleurs, notons que si une contrainte même inférieure à la limite d'élasticité est maintenue dans le temps (déformation lente ou comportement statique), le matériau peut éventuellement se déformer par *fluage* (déformation irréversible différée). De façon générale, le critère cinétique intervient lui aussi dans le comportement du matériau soumis à des sollicitations.

Test de traction

Un essai de traction est une expérience qui permet d'obtenir des informations sur le comportement élastique, le comportement plastique et le degré de résistance à la rupture d'un matériau, lorsqu'il est soumis à une sollicitation uniaxiale.
Prenons le cas de la traction ou de la compression d'une pièce cylindrique ou parallélépipédique selon son axe. La traction-compression correspond à des forces s'exerçant perpendiculairement aux sections de ces pièces ; elle est dite uniaxiale car les côtés de la pièce ne sont pas contraints, toutes les forces sont sur un même axe.



En prenant des pièces de différentes dimensions, on remarque que :

— **pour une force donnée**, l'allongement $\Delta\ell$ est proportionnel à la longueur initiale ℓ_0 du cylindre ;

— **pour une longueur donnée**, la force de traction ou compression est proportionnelle à l'aire S de la section du cylindre.

1. c'est-à-dire au moment dynamique résultant.

Si l'on veut caractériser le matériau en faisant abstraction de la forme de la pièce et de ses dimensions, on définit donc :

- l'*allongement relatif* ou *déformation* (en anglais *strain*)

$$\varepsilon = \frac{\Delta \ell}{\ell_0} = \frac{\ell - \ell_0}{\ell_0}$$

  sans dimension, parfois exprimé parfois en %.

- la *contrainte* (en anglais : *stress*) est la force par unité de surface de section,

$$\sigma = \frac{F}{S}$$

  Elle est homogène à une pression, et généralement exprimée en mégapascal (MPa, du fait des valeurs énormes mises en jeu). Elle est parfois aussi notée R.

Loi de Hooke

La loi élastique s'écrit alors :

$$\boxed{\sigma = E\varepsilon}$$

où E est le *module de Young*, qui est une caractéristique du matériau, généralement exprimé en gigapascals (GPa).

La courbe de traction dite « conventionnelle [2] » est obtenue en traçant la contrainte en fonction de l'allongement relatif.

Pour un matériau ductile, on obtient typiquement l'allure ci-contre (Fig. 1).

Dans un premier temps, la déformation est élastique donc la courbe est une droite linéaire, de pente E.
À partir d'un certain allongement, la courbe s'infléchit : c'est le début de la déformation plastique. La transition peut être franche (rupture de pente), ce qui permet de déterminer facilement la *limite d'élasticité* R_e. Lorsque la rupture de pente n'est pas franche (cas des matériaux très ductiles), on définit la *limite d'élasticité conventionnelle* $R_{e_{0,2}}$ comme étant la contrainte donnant 0,2 % de déformation résiduelle.
La courbe de traction présente ensuite un maximum qui détermine la *résistance à la traction conventionnelle* R_m. L'allongement plastique à ce point est appelé allongement sous charge maximale et est noté A_g ; c'est la déformation résiduelle maximale que l'on peut imposer. On définit également l'allongement total sous charge maximale, A_{gt}, qui inclut la déformation élastique.

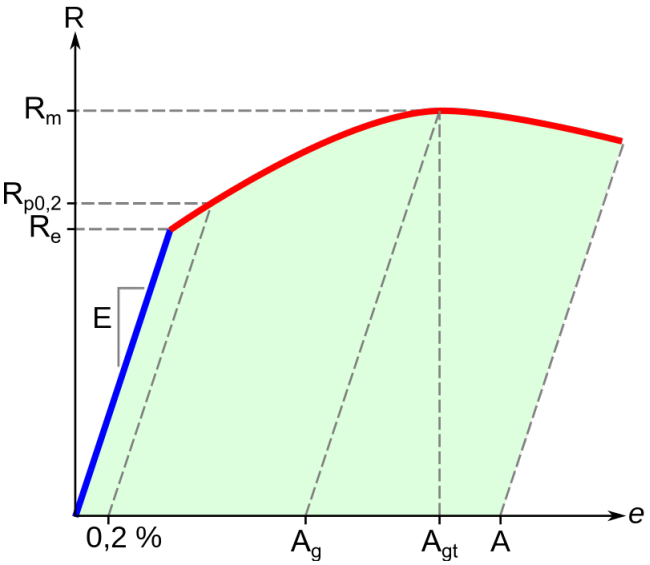


FIGURE 1 – Courbe de traction conventionnelle typique d'un matériau ductile (Par Cdang - Travail personnel, from File :Courbe contrainte vs deformation.png. https ://-commons.wikimedia.org/w/index.php ?curid=21883751).

À partir de ce point, la déformation est concentrée dans une zone, c'est la striction (« étranglement »). La force enregistrée diminue, puisque la section diminue dans la zone de striction. La rupture a ensuite lieu dans la zone de striction.

2. En toute rigueur on a définit ici la déformation et la contrainte dites *nominales*, c'est-à-dire à partir des dimensions initiales ℓ_0 et S avant traction. Le calcul de la déformation et de la contrainte dites *vraies*, c'est-à-dire prenant en compte l'évolution continue de la longueur et de la section au cours de l'essai donne lieu à la courbe dite « rationnelle », un peu différente. Comme nous travaillons ici sur des fils, ce calcul est hors de portée et on se restreindra à la courbe conventionnelle.

Pour un matériau fragile, la rupture survient en fin de domaine élastique. L'allongement plastique à la rupture est nul ou très faible. On ne peut déduire de la courbe que le module de Young E, et la résistance à la traction R_m.

II. Dispositif expérimental

Schéma complet

Une plate-forme de traction permet d'étirer un fil entre l'extrémité mobile d'actionneur linéaire et un support fixe muni d'un capteur de force (cf Fig. 2).

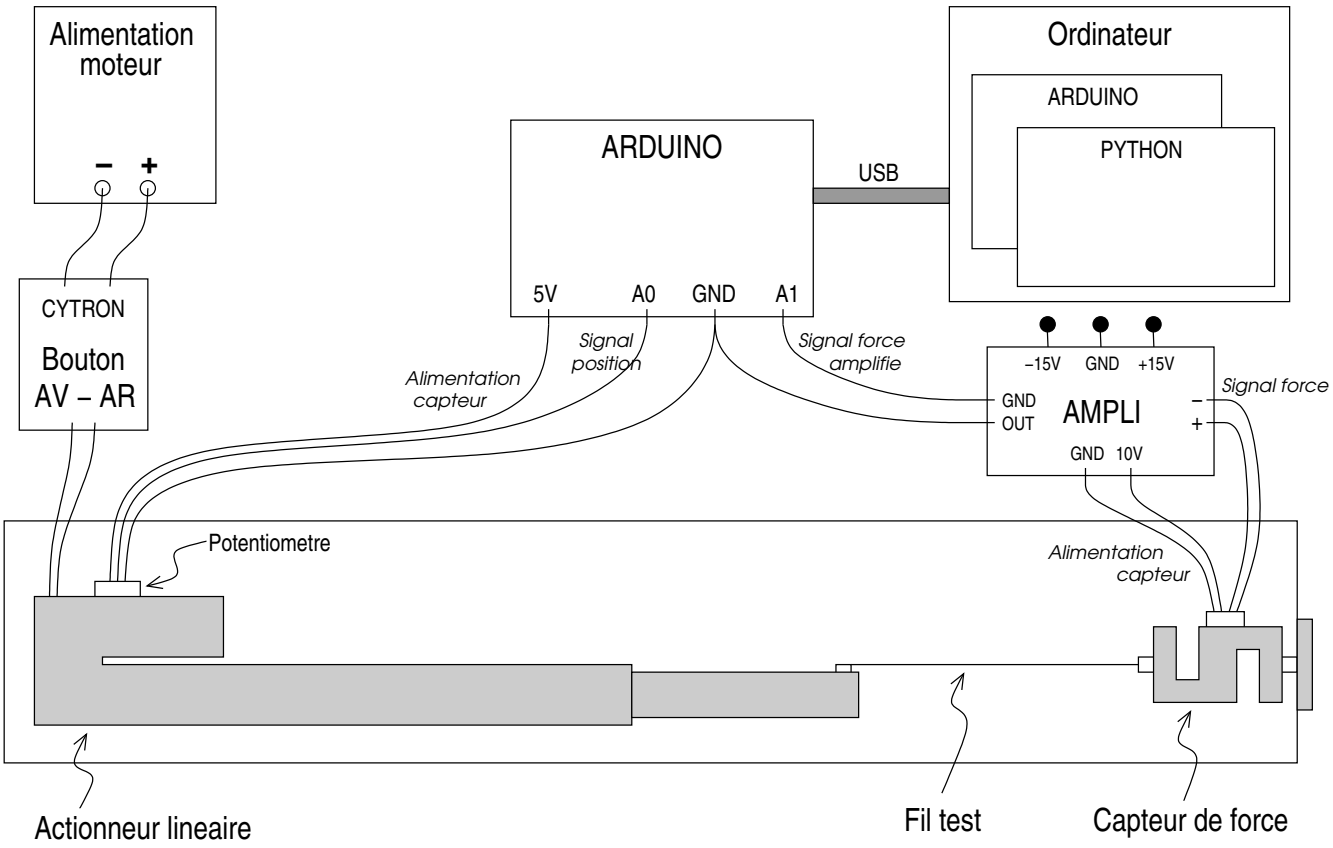


FIGURE 2 – Schéma complet du dispositif expérimental.

- Le moteur de l'actionneur est piloté par une alimentation en tension continue réglable suivie d'une plaquette de commande CYTRON [3] permettant d'inverser le sens du déplacement (via le signe de la tension appliquée au moteur). À charge constante, la vitesse de déplacement est proportionnelle à la tension appliquée.
- La position du bras mobile sur une plage de 25 cm est captée via un *potentiomètre* interne, qui fourni une tension variable en relation affine.
- La force est mesurée via un capteur de force, qui transforme la déformation élastique d'une *jauge de déformation* [4] en une tension proportionnelle via à un pont de Wheatstone intégré (cf documentation fournie). Ce signal très faible nécessite d'être amplifié par un amplificateur différentiel conçu sur mesure [5], qui permet aussi d'alimenter indirectement le pont de Wheatstone du capteur (10V). Cet amplificateur doit être alimenté (comme un ALI) avec une alimentation -15V , 0V , $+15\text{V}$.

3. Cet élément aurait pu être remplacé par un simple système de commutation via interrupteurs. Mais cette plaquette a été choisie pour plus de confort et de polyvalence, permettant si besoin de piloter le moteur via un microcontroleur.

4. Il s'agit d'un piézorésistor. La *piézorésistance* est le changement de résistance d'un résistor dû à une contrainte mécanique, qui peut modifier à la fois la géométrie et la conductivité du matériau.

5. Le signal en entrée de la carte ARDUINO doit être suffisamment grand mais ne doit pas dépasser 5V.

- La carte ARDUINO sert de carte d'acquisition : elle réalise l'échantillonnage et la conversion Analogique-Numérique des signaux des capteurs (entrées A0 et A1 0-5V, conversion sur 10 bits donc 1024 niveaux de quantification), et la transmission périodique vers l'ordinateur via le port série pour le traitement des données. On utilise aussi sa tension d'alimentation de 5V pour alimenter en dérivation les deux capteurs [6].
- L'ordinateur permet de traiter les données. La carte ARDUINO est ici utilisée en « mode Maître-Esclave » c'est-à-dire directement pilotée via le port série par un script Python qui recueille et traite les données. Les scripts Arduino et Python sont fournis. Seul le script Python devra être modifié selon vos besoins.

Matériel et accessoires

Pour réaliser les tests, on dispose de différents types de fil (matière et largeur), ainsi que d'outils qui permettront de préparer et fixer les échantillons (image ci-contre).

On portera une attention particulière sur la façon de fixer le fil : celui-ci doit être coincé suffisamment fermement pour ne pas glisser au cours de la traction (ce qui fausserait la mesure de son allongement), mais doit être le moins possible écrasé par le serrage (ce qui fausserait la courbe de traction en facilitant la rupture). Pour cela on utilise des joints de plomberie en fibre au contact direct du fil, et on enroule le fil autour de la tige (sur sa partie lisse) sur un demi-tour. Le fil est serré par un écrou et une paire de rondelles en acier.

III. Manipulations

- ***MANIP 1 : Étalonnage du potentiomètre***
  - Allumer l'alimentation stabilisée continue du moteur et la régler en générateur de tension. Fixer une tension relativement basse mais suffisante pour que le moteur se déplace lorsqu'on actionne les boutons AV-AR [a].
  - À l'aide du réglet et du voltmètre (connections GND et CURSEUR), étalonner le potentiomètre pour établir la relation *position-tension*.

a. La plaquette de commande CYTRON fonctionne normalement pour des tensions entre 5V et 30V. En pratique 4V convient.

- ***MANIP 2 : Étalonnage de la jauge de contrainte***

À l'aide du dynamomètre 50N, et du voltmètre, étalonner la jauge pour établir la relation *force-tension*.

On procède maintenant en deux temps, pour séparer les difficultés :

— un premier essai sur la console d'acquisition FOXY pour vérifier que tout fonctionne bien, et avoir un premier jeu de données exploitable ;

— d'autres essais suivis d'une exploitation complète avec le système d'acquisition ARDUINO-PYTHON.

6. En cas de signal de force trop faible, on peut aussi augmenter cette tension d'alimentation plutôt que de jouer sur le gain du filtre.

• ***MANIP 3 : Premier test de traction avec la console Foxy***

- Brancher les sorties des 2 capteurs sur 2 voies de la Foxy ;
- Fixer un échantillon de fil, si possible non tordu ;
- Mesurer sa longueur initiale et son diamètre ;
- Effectuer une acquisition jusqu'à rupture ;
- Sauvegarder le tableau des valeurs au format .csv [a] pour une exploitation ultérieure sous Python.

a. Depuis Atelier Scientifique, il faut d'abord copier-coller les valeurs dans Excel ou LibreOffice puis exporter en .csv

• ***MANIP 4 : Tests de traction avec ARDUINO-PYTHON***

- Connecter les sorties des deux capteurs aux entrées A0 et A1 de la carte ARDUINO. Téléverser le programme depuis l'application ARDUINO, après avoir vérifié que le port série a été correctement défini (cf annexe).
- À l'aide du script de pilotage Python `traction_acquisition.py`, réaliser d'autres tests de traction pour différents types de fils. Le script sauvegarde automatiquement les données à chaque expérience [a]

a. Changer de nom à chaque fois pour ne pas écraser vos précédentes données.

Q1. Les entrées analogiques de la carte ARDUINO étant conçues pour recevoir des tensions sur l'intervalle $[0\,\mathrm{V}, 5\,\mathrm{V}]$ et les convertir sur 1024 niveaux (10 bits), quelle relation doit-on appliquer sur ce signal numérique enregistré pour reconstruire la valeur de tension du capteur ?

• ***MANIP 5 : Traitement des données avec PYTHON***

Traiter les séries de données précédemment enregistrées avec le script de traitement Python `traction_traitement.py` que vous aurez préalablement complété en implémentant les deux étalonnages précédents dans le script.

- Tracer la courbe de traction conventionnelle $\sigma = f(\varepsilon)$. Le matériau est-il ductile ou fragile ?
- Déterminer les grandeurs caractéristiques E et R_m, et si le matériau est ductile, R_e et A_g.

On prendra soin de relever la vitesse de déplacement du bras de l'actionneur et la tension d'alimentation associée. Si le temps le permet on testera l'effet de ce paramètre. Dans un premier temps il est préférable de procéder avec une vitesse aussi petite que possible.
On n'oubliera pas d'évaluer les incertitudes.

IV. ANNEXE : Acquisition des données avec ARDUINO-PYTHON

IV.1. Carte Arduino

Arduino est la marque d'une plateforme de prototypage open-source qui permet aux utilisateurs de créer des objets électroniques interactifs à partir de cartes électroniques matériellement libres sur lesquelles se trouve un *microcontrôleur*[7]. Le microcontrôleur peut être programmé pour analyser et produire des signaux électriques, de manière à effectuer des tâches très diverses. Le langage utilisé est le C^{++} avec des bibliothèques associées. Un logiciel d'environnement de programmation open-source est fourni pour cela (Arduino IDE). La carte Arduino peut être utilisée pour construire des objets interactifs indépendants (prototypage rapide), ou bien peut être connecté à un ordinateur pour communiquer avec ses logiciels. Les différentes versions des Arduino fonctionnent sous le même principe général, décrit succinctement ci-dessous.

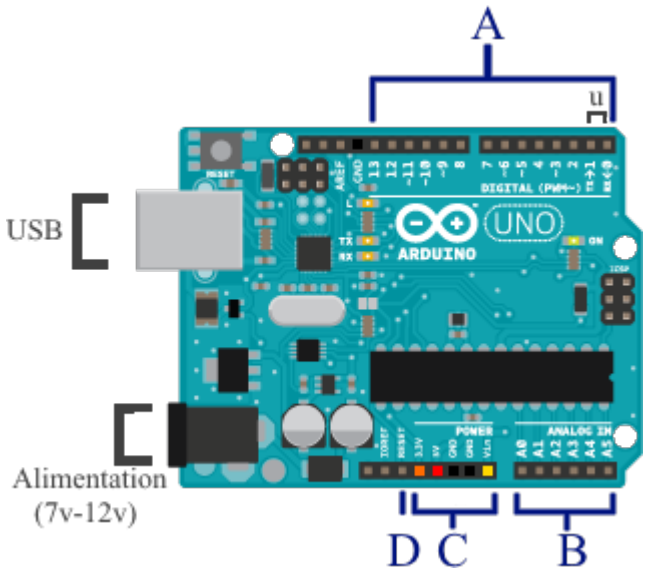


FIGURE 3 – Carte Arduino UNO (Par Sad-Cloud – Travail personnel, CC BY-SA 4.0, https ://commons.wikimedia.org/w/index.php ?curid=37290368).

- A : 14 broches dites *numériques* c'est-à-dire à fonctionnement binaire (1 ou 0, HIGH ou LOW en anglais) ; elles offrent en sortie du 5V et acceptent en entrée du 5V sur le même principe. Fonctions `digitalWrite()` et `digitalRead()`, et pour les ports PWM [a] `analogWrite()`.
- B : 6 broches dites *analogiques* (`A0-A5`), reçoivent des tensions entre 0V et 5V, converties en entiers naturels par le CAN sur 10 bits, pour des fréquences de 10kHz maximum. Fonction `analogRead()`.
- C : les différentes broches d'**alimentation en sortie** (où la carte sert de générateur de tension pour un élément externe) : 5V, 3,3V, la masse (GND), et `Vin` reliée à l'alimentation de la carte (7 V-12 V).

a. Pulse Width Modulation, ou Modulation de Largeur d'Impulsions, technique couramment utilisée pour synthétiser des signaux pseudo analogiques à l'aide de circuits à fonctionnement binaire.

Lorsqu'elle est reliée à l'ordinateur par le port USB, la carte est alimentée par ce biais en 5V.

Toute erreur de polarité (signe de la tension appliquée) ou dépassement de la valeur nominale (5V) peut conduire à la **destruction de la carte** ! On prendra donc soin de s'assurer que les signaux appliqués sont bien dans l'intervalle attendu.

IV.2. Environnement de programmation Arduino

L'interface de programmation du microcontrôleur s'ouvre (Fig. 4) lorsque l'on clique sur un fichier de type `programme.ino` (ci-après appelé *sketch*), après avoir branché la carte à l'ordinateur via le port USB[8]. On commence par vérifier le choix du type de carte Arduino (ici UNO en ce qui nous concerne) dans le menu

`Outils -> Type de carte`

ainsi que le port proposé pour la connection :

`Outils -> Port série`

7. Un microcontrôleur est un circuit intégré qui rassemble les éléments essentiels d'un ordinateur : processeur, mémoires (mémoire morte et mémoire vive), unités périphériques et interfaces d'entrées-sorties. Il constitue une alternative plus accessible par rapport aux microprocesseurs polyvalents utilisés dans les ordinateurs personnels, avec de nombreuses applications notamment dans la domotique, la robotique et plus généralement les systèmes embarqués.

8. Au branchement, la diode `ON` située à côté du nom `UNO` doit s'allumer.

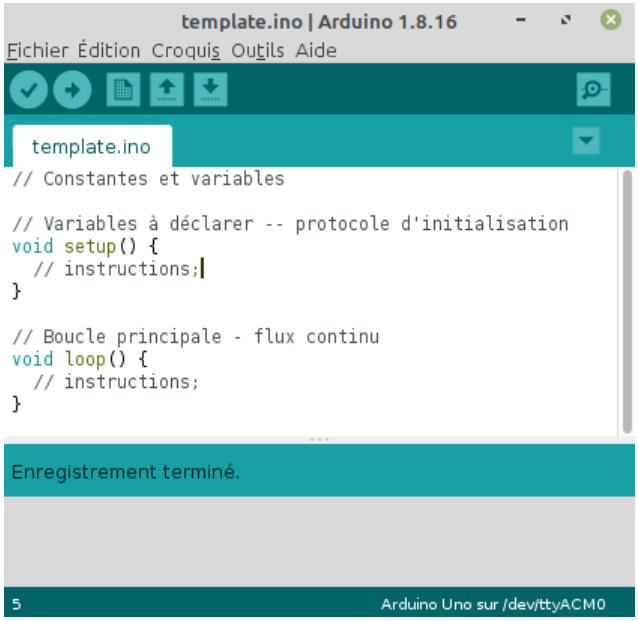


FIGURE 4 – Interface Arduino IDE et structure type d'un programme.

 : Bouton pour compiler et téléverser le sketch dans le microcontrôleur.

Pour tester le bon fonctionnement de la carte on peut ouvrir puis téléverser le sketch `Blink`, que l'on trouvera ici :
`Fichiers -> Exemples -> 01.Basics -> Blink`
et qui a pour effet de faire clignoter la LED `L`.

La structure générale d'un sketch Arduino est illustrée ci-contre. Après la définition éventuelle de quelques constantes,

- une première procédure d'initialisation `setup()` est exécutée une seule fois ;
- une seconde procédure `loop()` est exécutée en boucle indéfiniment, c'est-à-dire jusqu'à l'arrêt de la carte, ou le téléversement d'un nouveau sketch.

IV.3. Programmes d'acquisition ARDUINO-PYTHON

Étant donné que le langage enseigné en CPGE est Python et non C^{++}, nous optons pour une utilisation limitée de ce dernier. La carte Arduino sera utilisée en tant que simple carte d'acquisition, et tout le traitement des données sera dévolu à Python. Pour cela on utilise la carte en mode *Maître-Esclave* à l'aide de la bibliothèque `pySerial`. Le code Python est séparé en deux scripts disjoints, l'un pour l'acquisition et la sauvegarde des données brutes, l'autre pour le post-traitement.

a. *Installation du code C^{++} dans le microcontrôleur (sketch)*

On commencera donc par téléverser, à l'aide de l'interface Arduino IDE, le programme du microcontrôleur ci-dessous (`lecture_capteurs.ino`, à ne pas modifier[9]), qui demande à la carte de lire et transférer périodiquement les valeurs de tension issues des deux capteurs sur deux entrées analogiques.

```
/*
  Lecture AnalogPin A0 et A1, pour traitement par Python
*/

// Constantes et variables
const int analogPinA0 = 0; // Choix de la voie d'acquisition (à adapter)
const int analogPinA1 = 1; // Choix de la voie d'acquisition (à adapter)
int capteur_Position = 0; // variable de stockage entière
int capteur_Force = 0; // variable de stockage entière

// Variables à déclarer – protocole d'initialisation
void setup() {
  pinMode(analogPinA0,INPUT); // Définition de la voix d'entrée
  pinMode(analogPinA1,INPUT); // Définition de la voix d'entrée
  // Ouverture d'une connection série pour transmettre les valeurs avec choix du taux de transfert (baud rate) :
  // Supported baud rates 300, 600, 1200, 2400, 4800, 9600, 14400, 19200, 28800, 31250, ...
  Serial.begin(19200); // Attend la connexion du port
  while(!Serial); // Attend la connexion du port
}

```

9. sauf éventuellement pour modifier le débit d'échange avec l'ordinateur (baud rate, à ajuster aussi dans le script Python) si l'échantillonnage est mauvais.

```
// Boucle principale – flux continu
void loop() {
  // read the values on AnalogPin A0 and A1 and store in a variables
  capteur_Position = analogRead(analogPinA0);
  capteur_Force = analogRead(analogPinA1);
  // send the date in millisecond out the serial port
  Serial.print(millis()); // envoi de la date en ms
  Serial.print("\t"); // Ajout d'une tabulation
  // send the 10-bit sensor values out the serial port
  Serial.print(capteur_Position);
  Serial.print("\t"); // Ajout d'une tabulation
  Serial.println(capteur_Force);
  // Une éventuelle temporisation si le débit est trop rapide (peu probable)
  delay(1); // attendre 1 ms
}
```

On peut remarquer que les données sont envoyées sous la forme d'un triplet d'entiers naturels (entre 0 et 1023)

date position force

séparés par des tabulations et suivis d'un retour à la ligne (`println`, ou `EOL` pour *End Of Line* en abrégé informatique), avec une périodicité de l'ordre d'une milliseconde (temporisation `delay(1);`), à condition que les opérations de lecture et d'écriture soient beaucoup plus rapides que cela... à vérifier donc.

b. Programme Python de pilotage de l'acquisition

Puis on exécute le code Python ci-dessous (script `traction_acquisition.py` à compléter) pour acquérir puis sauvegarder les données mesurées brutes (numériques). Pour ce faire on ouvrira un environnement de programmation au choix, par exemple en passant par la suite Anaconda (utiliser le logiciel Spyder, ou iPython ou Jupyter si vous savez vous en servir, après avoir sélectionné l'environnement *envCPGE* pour avoir accès à la bibliothèque pySerial). A priori ce script ne nécessite pas d'être modifié, si ce n'est pour adapter éventuellement le nombre de points d'acquisition, ou le débit d'échange avec la carte ARDUINO (baud rate).

```
import numpy as np # module de calcul numérique
import os # module de dialogue avec le système d'exploitation
import matplotlib.pyplot as plt # module graphique
import serial # gestion du port série
import time # module de gestion du temps

# Définition du nom de l'expérience pour les sauvegardes : 1 mot, pas de caratères spéciaux
name_exp = input("Entrer le nom de l'experience (ex: nylon\_25kg): ")
namefig = name_exp + '_brut.png'
namefile = name_exp + '_brut.txt'
# Paramètres initiaux de l'échantillon avant étirement (1ère évaluation)
l0 = float(input("Entrer la longueur initiale de l'echantillon (mm) : "))
d0 = float(input("Entrer le diametre initial de l'echantillon (mm) : "))

# Connexion de l'ordinateur au port série, même débit que dans le code Arduino
# Supported baud rates 300, 600, 1200, 2400, 4800, 9600, 14400, 19200, 28800, 31250, 384
serial_port = serial.Serial(port = 'COM9', baudrate =19200) # Ouverture du port
    # port sous LINUX : '/dev/ttyACM0' (ou /dev/ttyACM1, ou /dev/ttyUSB0...)
    # port sous WINDOWS : 'COM9' (ou 'COM0, 1, 2, 3,... choisir a priori le plus élevé)

serial_port.setDTR(False) # Initialisation de la carte Arduino
time.sleep(1) # temps de stabilisation - temporisation
serial_port.setDTR(True) # Activation de la carte
serial_port.reset_input_buffer() # Vide le tampon de stockage existant

# Réception des mesures
N=2000 # nombre d'acquisitions
instants = np.zeros(N) # Tableau des instants, type float64 par défaut.
```

```
positions = np.zeros(N) # Tableau d'acquisition des valeurs de position
forces = np.zeros(N) # Tableau d'acquisition des valeurs de force
try:
    for i in range(N) :
        mesures = serial_port.readline().split() # lit une ligne avec EOL + séparation
        instants[i] = int(mesures[0])
        positions[i] = int(mesures[1])
        forces[i] = int(mesures[2])
except:
    print("defaut de fonctionnement")
    pass

# Fermeture du port série
serial_port.close()

# Sauvegarde des valeurs dans un fichier texte
file=open(namefile,'w') # Ouverture en lecture et écriture ('w' : write)
# 1ères lignes : écriture d'un en-tête
file.write("Experience : \t"+ name_exp +"\n")
file.write("Longueur initiale echantillon (mm) : \t"+ str(l0) +"\n")
file.write("Diametre initial echantillon (mm) : \t"+ str(d0) +"\n")
file.write("Instant \t Position \t Force \n")
# lignes suivantes : valeurs numériques séparées par des tabulations et EOL en fin
for i in range(N):
    file.write(str(instants[i])+'\t'+str(positions[i])+'\t'+str(forces[i])+'\n')
file.close()

# Graphe de contrôle
plt.plot(positions,forces,'-b')
plt.xlabel("Positions (numerique 10 bits)")
plt.ylabel("Forces (numerique 10 bits)")
plt.title("Traction sur un fil de "+name_exp+' - Signaux bruts')
plt.grid()
plt.savefig(namefig)
plt.show()

# Pour gérer le répertoire courant depuis Python (emplacement sauvegardes) :
# obtenir le nom du répertoire courant : os.getcwd()
# changer le répertoire courant : os.chdir('chemin_de_mon_repertoire')
```

On notera l'importance de la bonne dénomination du port affecté à la communication avec la carte Arduino, qui dépend du système d'exploitation et du poste utilisé. Elle doit correspondre au choix effectué dans l'interface Arduino IDE, menu :
`Outils -> Port série`
En cas de doute sur le port utilisé sous LINUX ou MAC, taper l'instruction suivante dans un terminal avant puis après avoir branché la carte :
`python -m serial.tools.list_ports`

On note aussi qu'au cours de la sauvegarde, on a pris soin d'inclure dans l'entête les informations nécessaires au traitement et à l'exploitation ultérieurs de ces données brutes. De manière générale il est toujours prudent de sauvegarder les données brutes avant traitement pour parer à une éventuelle erreur d'étalonnage ou de calcul. C'est pourquoi le code de traitement a été placé dans un second script.

c. Programme Python de traitement

Le traitement des données brutes est fait à l'aide du script `traction_traitement.py` ci-dessous, à compléter.

```
import numpy as np # module de calcul numérique
import os # module de dialogue avec le système d'exploitation
```

```
import matplotlib.pyplot as plt # module graphique

# Définition du nom de l'expérience pour les sauvegardes
name_exp = input("Entrer le nom de l'experience : ")
namefile = name_exp + '_brut.txt'
namefig = name_exp + '.png'

# Lecture des données brutes stockées dans le fichier texte "namefile"
donnees_brutes = np.loadtxt(namefile,delimiter='\t',skiprows=4)
# Rmq : ici on n'importe pas les 4 1ères lignes (entête)
nb_mes = np.shape(donnees_brutes)[0] # nombre de mesures
# Extraction des colonnes
instants = donnees_brutes[:,0]
positions = donnees_brutes[:,1]
forces = donnees_brutes[:,2]

# Conversion des valeurs mesurées
### PARTIE À MODIFIER ###
deformations = positions
contraintes = forces
### FIN PARTIE À MODIFIER ###

# Graphe
plt.plot(deformations,contraintes,'-b')
plt.xlabel("Deformation (%)")
plt.ylabel("Contrainte (MPa)")
plt.title("Traction sur un fil de "+name_exp)
plt.grid()
plt.savefig(namefig)
plt.show()
```

La partie à compléter concerne la traduction des valeurs brutes (entiers entre 0 et 1023) en terme de position ou déplacement puis de déformation d'une part, et de force puis contrainte d'autre part. Son écriture est spécifique à chaque poste expérimental puisque cela repose sur les étalonnages des capteurs [10].

10. Chaque dispositif expérimental diffère des autres par les valeurs exactes des tensions d'alimentation des capteurs et les valeurs réelles des composants qui les constituent.