

# ACQUISITION ET TRAITEMENT DES DONNÉES

TP 2

## Chaine d'acquisition



UNIVERSITÉ  
BOURGOGNE  
EUROPE



POLYTECH  
DIJON



[charles.meunier@u-bourgogne.fr](mailto:charles.meunier@u-bourgogne.fr)

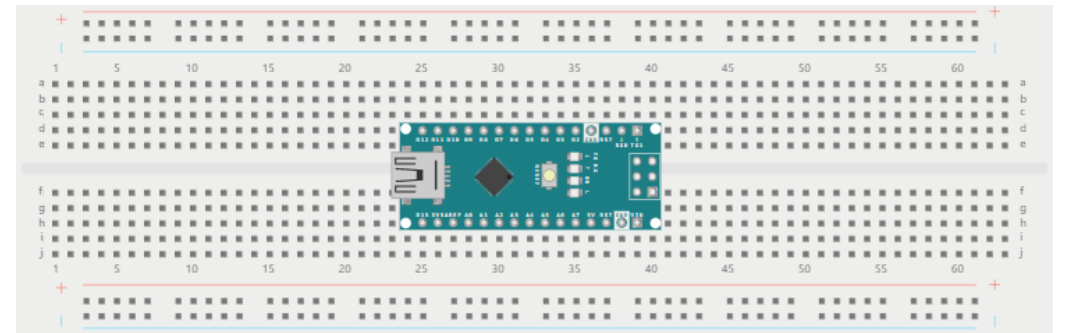
## OBJECTIFS



- Acquérir des données numériques en provenance d'un capteur
- Transférer les données vers un ordinateur
- Traiter et afficher les données reçues

## ARDUINO IDE

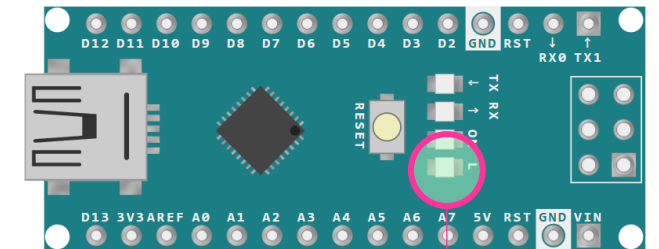
- Prenez un Arduino Nano ainsi qu'un kit de composants (breadboard, câbles, résistances, led, ...)
- Placez l'Arduino sur la breadboard comme sur le schéma ci-contre :
- Démarrez l'environnement de développement Arduino IDE installé sur les postes de la salle de TP.



**Note** : vous pouvez installer Arduino IDE sur votre ordinateur personnel depuis ce lien : <https://www.arduino.cc/en/software>

La carte Arduino Nano intègre une LED branchée sur la broche **BUILTIN\_LED**.

- Ecrivez un programme faisant clignoter la LED de la carte Arduino à une fréquence de 1Hz
- Connectez la carte Arduino à l'ordinateur à l'aide d'un câble USB (notez le port COM utilisé)
- Compilez et transférez le programme sur la carte Arduino
- Vérifiez que la LED clignote bien



LED embarquée

## QU'EST-CE DONC QUE CECI ?

Un capteur vous a été fourni pour ce TP.

- De quel type de capteur s'agit-il ?
- A partir de sa documentation (disponible en ligne) expliquez de manière détaillée comment le faire fonctionner.
- Qu'est-ce qu'un chronogramme ?
- Représentez les signaux des broches **echo** et **trigger** sur un chronogramme.

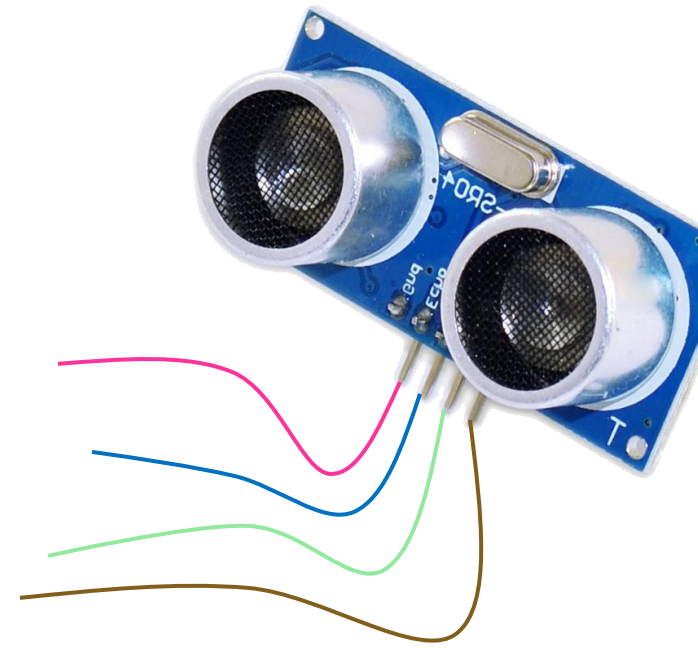
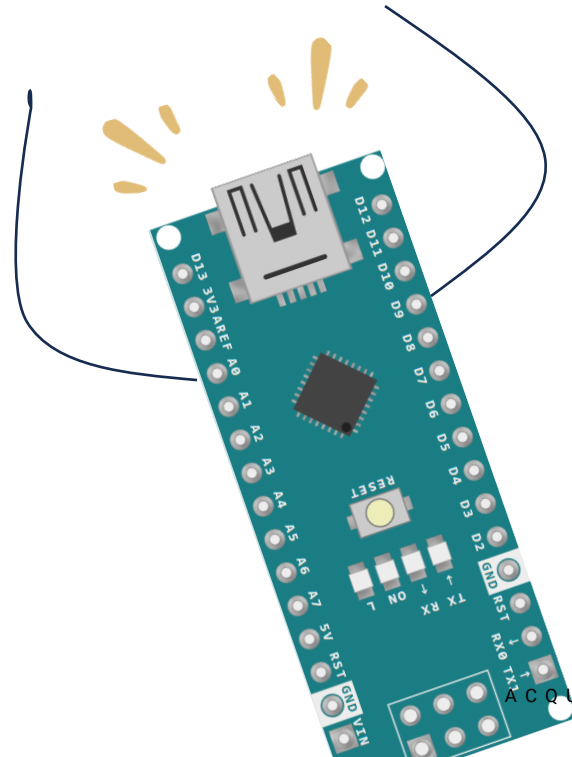


## ACQUISITION

- Débranchez le câble reliant l'Arduino à l'ordinateur (coté PC seulement)
- Placez le capteur sur la breadboard et connectez-le à l'Arduino
- Faites vérifier votre montage par un enseignant
- Affichez sur le terminal de la voie série l'information renvoyée par le capteur

### TIPS

Le terminal de la voie série est accessible via :  
*Tools > Serial Monitor*



# PYTHON

- Quittez le logiciel Arduino IDE une fois que votre programme fonctionne correctement.
- Dans le dossier de votre TP, créez un dossier "python" et ouvrez-le avec Visual Studio Code.
- Créez un fichier program.py
- Pour ne pas rompre la tradition, écrivez un programme qui affiche "Hello World" dans la console.
- Exécutez le en cliquant sur le triangle qui doit se situer en haut à droite de la fenêtre de VS Code.

## TIPS

La fonction `print("QueLqueChose")` permet...  
d'afficher quelque chose.

# MATPLOTLIB

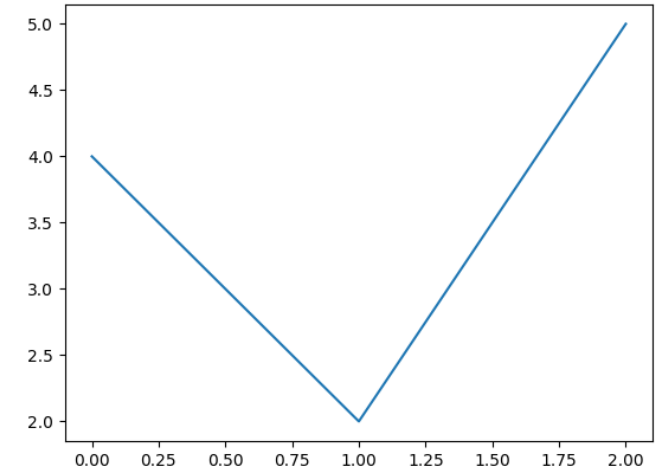
La bibliothèque Python Matplotlib permet d'afficher des courbes :

```
import matplotlib.pyplot as plt

xCoordinates = [0, 1, 2] # Abscisses des points de la courbe
yCoordinates = [4, 2, 5] # Ordonnées des points de la courbe

plt.plot(xCoordinates, yCoordinates) # Tracé de la courbe

plt.show() # Affichage de la fenêtre
```



➤ Ecrivez un programme qui affiche une sinusoïde.

## TIPS

*La commande `pip install nom_bibliotheque` permet d'installer des bibliothèques qui ne seraient pas présentes sur votre poste.*

# PYSERIAL

La bibliothèque Python `PySerial` permet de recevoir et de transmettre des données via la voie série :

```
import serial
try:
    serialConnection = serial.Serial("COM 5", 9600) # Ouverture de la liaison série sur le port COM 5
except serial.SerialException:
    print("Impossible d'ouvrir le port série") # Quitte en affichant un erreur si une erreur se produit
    exit()

bytes = serialConnection.readline() # Lit tous les caractères jusqu'à trouver un retour à la ligne
string = bytes.decode() # Convertit les octets lus en une chaîne de caractères
print(string) # Affiche la donnée lue
```

- Écrivez un programme qui affiche les valeurs transmises par la carte Arduino (n'oubliez pas de brancher le câble USB entre le PC et l'Arduino).

## COMBO

- Ecrivez un programme qui affichera en temps réel les variations de distance mesurées par le capteur. La courbe affichée ne présentera que les 100 dernières valeurs mesurées.

### TIPS

```
plt.plot(xCoordinates, yCoordinates)      # Crée une courbe à partir des coordonnées fournies
plt.gca().line[0].set_xdata(xValues)     # Met à jour les abscisses des points de la courbe
plt.gca().line[0].set_ydata(yValues)     # Met à jour les coordonnées des points de la courbe

plt.gca().relim()                        # Recalcule les bornes du graphique
plt.gca().autoscale_view()               # Recalcule l'échelle du graphique

plt.pause(0.01)                          # Réalise une pause permettant le rafraichissement du graphique tout en
# permettant des interactions avec l'application
```