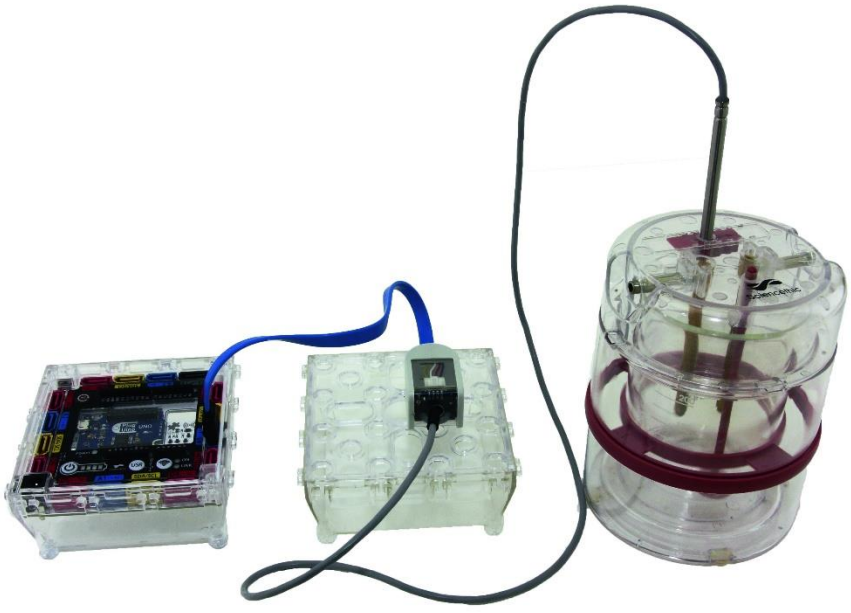
Niveau : seconde	Ondes et signaux partie 3 : Signaux et capteurs
Capteurs électriques.	*Produire et utiliser une courbe d'étalonnage reliant la résistance d'un système avec une grandeur d'intérêt (température, pression, intensité lumineuse, etc.).* *Utiliser un dispositif avec microcontrôleur et capteur.*

Dossier TP Plug'Uino® : Réalisation d'un thermomètre électronique à l'aide d'une CTN à partir d'un étalonnage simple

Programmation en langage Python



1. Résumé de l'activité

Cette activité expérimentale a pour but de réaliser la courbe d'étalonnage d'une sonde de mesure de température CTN, et de réaliser ainsi un thermomètre numérique.

2. Thème du programme abordé

Niveau : seconde	Ondes et signaux partie 3 : Signaux et capteurs
Capteurs électriques.	*Utiliser un dispositif avec microcontrôleur et capteur.*

3. Matériel mis en œuvre

Une interface Plug'Uino®Uno Réf. 650 003 ;
Un capteur de température CTN Plug'Uino® Réf. 651 054;
Un calorimètre Réf. 005 025;
Un thermomètre numérique Réf. 310 007;
Un tableur-grapheur.

4. Présentation de l'expérience réalisée

— On verse de l'eau chaude dans un calorimètre. On y plonge la sonde de température CTN Plug'Uino®, ainsi qu'un thermomètre de référence.
— À l'aide du script Python, on saisit la valeur de température donnée par le thermomètre de référence et on enregistre la mesure envoyée par l'entrée analogique sur laquelle est branché le capteur de température CTN. On fait varier la température de l'eau en rajoutant progressivement de la glace pilée et en homogénéisant bien le mélange et on relève, en validant les demandes du script Python, les valeurs pour différentes températures.

Remarque : il est aussi possible de procéder plus rapidement en préparant des récipients contenant de l'eau à différentes températures.

— En fin de script Python après la dernière mesure, les mesures sont affichées dans la console et il est possible d'afficher un tracé T=f(mesure capteur)

— En copiant ces mesures, on entre les valeurs obtenues dans un tableur-grapheur afin de visualiser la courbe d'étalonnage$T = f(mesurescapteur)$.

— On réalise alors une courbe de tendance permettant d'obtenir la relation mathématique$T = f(mesurescapteur)$.

— On saisit cette fonction dans le script Python de façon à ce qu'il puisse afficher directement la température mesurée par la sonde CTN.

5. Programme Python

Le programme Python fonctionnel, permettant de relever la valeur mesurée par le microcontrôleur et délivré par la sonde de température CTN est le suivant :

```
""" Sciencéthic étalonnage d'une sonde de température CTN branché sur la broche
 analogique A0 avec une LED branchée sur la broche digitale D2"""

#------------------------------Attention ------------------------------------------------
#
# A partir de l'IDE Arduino, téléverser dans la carte Arduino l'exemple "StandardFirmata"
# accessible via :Menu Fichier / Exemples / Firmata / StandardFirmata.
#
#-----------------------------------------------------------------------------------------


from pyfirmata import Arduino, util
import matplotlib.pyplot as plt
import time                     #importation des bibliothèques

#------------------------------ paramétrage de la communication-----------------------------------------
com=input('Saisir le port de communication COM1, COM2, COM3 .... :')
port = Arduino(com)             # la variable port contient le port de communication
util.Iterator(port).start()     # itérateur permettant de ne pas saturer la liaison série
A0 = port.get_pin('a:0:o')      # A1 désigne la broche analogique 1 (a:1) utilisée en sortie (:o)
time.sleep(0.5)                 # délai d'attente pour la mise en place de la communication
port.digital[2].write(1)        # écriture sur le port digital 2 (Ok pour mesure si LED allumée)

#------------------------------ mesures ----------------------------------------------------------------
mesures_capteur=[]              # liste des mesures données par le capteur
temperatures=[]                 # liste des températures
poursuite=True                  # initialisation de la variable poursuite des mesures
while poursuite:                # tant qu'une mesure est à effectuer
    temperatures.append(float(input(' valeur de la température : ')))  # ajout de la température dans la liste
    m=A0.read()                 # lecture de la valeur sur la broche A1
    mesures_capteur.append(m)   # ajout de la valeur dans la liste
    print('mesure : ',m)        # affichage de la valeur mesurée
    poursuite=input('Déclencher une nouvelle mesure o/n ? ')
    if poursuite=='n'or poursuite=='N':
        poursuite=False
print('liste des mesures : ',mesures_capteur) # affichage de la liste des mesures
port.digital[2].write(0)        # écriture sur le port digital 2 (fin de mesure si LED éteinte)
port.exit()                     # fermeture de la communication sur le port

#------------------------------ réqultats des mesures -------------------------------------------------
print('températures : ',temperatures)
print('mesures du capteur : ', mesures_capteur)

#------------------------------ tracé ------------------------------------------------------------------
trace=input('voulez-vous effectuer un tracé o/n : ')
if trace=='o':
    plt.plot(mesures_capteur,temperatures,'+')
    plt.xlabel('mesures capteur')
    plt.ylabel('température (°C)')
    plt.show()
```

ATTENTION :

- Pour le pilotage en langage Python, il est indispensable de mettre en place de la connexion du microcontrôleur et la configuration initiale Python.
- Au début de l'exécution du script Python saisir le port de communication sur lequel est branché le microcontrôleur via la connexion USB. (ce port est accessible sur l'IDE Arduino menu outils/port)

Valeurs obtenues dans les listes Python :

temperatures = [4.4, 7.6, 11.5, 16.6, 26.2, 29.2, 40.2, 46.3, 50.2, 58.5, 69.4]

mesures_capteur = [0.2893, 0.3275, 0.3587, 0.4203, 0.5132, 0.5386, 0.6325, 0.6794, 0.6872, 0.7283, 0.7869]

6. Résultats obtenus

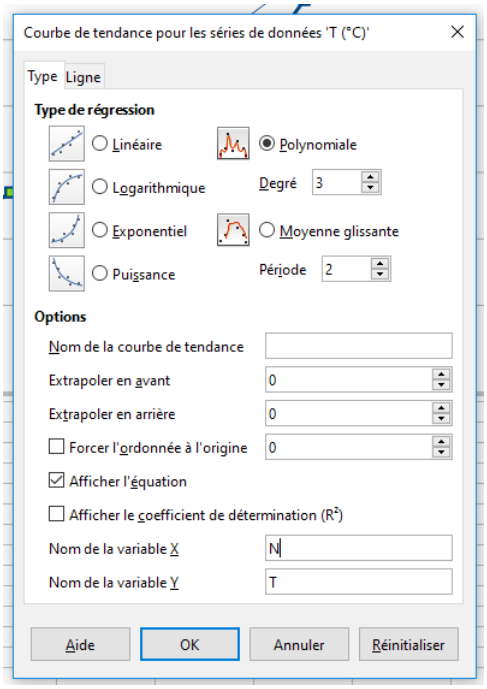
La courbe d'étalonnage obtenue est la suivante (et disponible dans le fichier traitement_temperature_simple.ods)

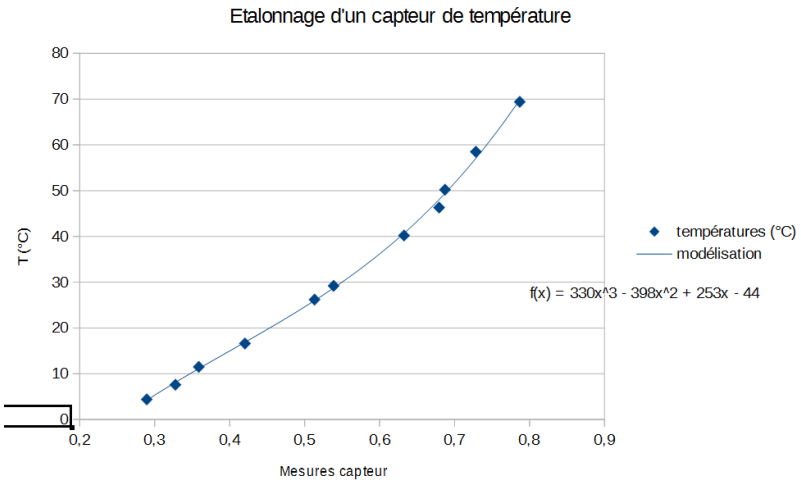
Exploitation sous LibreOffice :

On modélise la courbe obtenue en double-cliquant sur le graphique, puis en faisant clic-droit sur la courbe de données, et « insérer une courbe de tendance » :

On sélectionne une courbe de tendance polynomiale de degré 3, et on demande d'afficher l'équation comme ci-contre.

La courbe de tendance ainsi que son équation sont alors affichées :





Il suffit alors de modifier le programme Python pour y rentrer la modélisation obtenue

script téléchargeable : *temperature_avec_modele.py*

```python
""" Sciencéthic mesure d'une température avec une sonde de température CTN branché sur la broche
 analogique A0 avec une LED branchée sur la broche digitale D2"""

#------------------------------Attention ---------------------------------------------
#
# A partir de l'IDE Arduino, téléverser dans la carte Arduino l'exemple "StandardFirmata"
# accessible via :Menu Fichier / Exemples / Firmata / StandardFirmata.
#
#-------------------------------------------------------------------------------------


from pyfirmata import Arduino, util
import matplotlib.pyplot as plt
import time                     #importation des bibliothèques

#------------------------------ paramétrage de la communication---------------------------------
com=input('Saisir le port de communication COM1, COM2, COM3 .... :')
port = Arduino(com)             # la variable port contient le port de communication
util.Iterator(port).start()     # itérateur permettant de ne pas saturer la liaison série
A0 = port.get_pin('a:0:o')      # A1 désigne la broche analogique 1 (a:1) utilisée en sortie (:o)
time.sleep(0.5)                 # délai d'attente pour la mise en place de la communication
port.digital[2].write(1)        # écriture sur le port digital 2 (Ok pour mesure si LED allumée)

#------------------------------ mesures ----------------------------------------------------

poursuite=True                  # initialisation de la variable poursuite des mesures
while poursuite:                # tant qu'une mesure est à effectuer
    m=A0.read()                 # lecture de la valeur sur la broche A0
    temperature=330*m*m*m-398*m*m+253*m-44   # calcul de la temperature à partir du modèle
    print('température =',round(temperature,2),'°C')        # affichage de la valeur mesurée
    poursuite=input('Déclencher une nouvelle mesure o/n ? ')
    if poursuite=='n'or poursuite=='N':
        poursuite=False

port.digital[2].write(0)        # écriture sur le port digital 2 (fin de mesure si LED éteinte)
port.exit()                     # fermeture de la communication sur le port
```

Les mesures sont alors affichées ainsi :

```
In [11]: (executing lines 1 to 36 of "temperature_avec_modele.py")
Saisir le port de communication COM1, COM2, COM3 .... :COM3
température = 34.06 °C
Déclencher une nouvelle mesure o/n ? o
température = 31.53 °C
Déclencher une nouvelle mesure o/n ? o
température = 26.34 °C
Déclencher une nouvelle mesure o/n ? o
température = 21.75 °C
Déclencher une nouvelle mesure o/n ? n
```

7. Exploitation

7.1. Programmation

En fonction des capacités des différents élèves on peut leur demander diverses choses :

- à un niveau **d'initiation**, on peut enlever certains commentaires du programme et demander aux élèves à quoi correspond cette ligne ;
- à un niveau **avancé** on peut par exemple enlever la ligne 29 du script et demander aux élèves de rentrer l'équation obtenue lors de la phase d'étalonnage ;
- à un niveau **expert** on peut demander aux élèves de programmer complètement le microcontrôleur.

7.2. Résultats des mesures

On peut discuter avec les élèves de la mesure de température affichée en fonction de celle mesurée par un thermomètre : l'écart est-il grand ? Varie-t-il en fonction de la plage de température sur laquelle on travaille ? Les mesures réalisées à partir du microcontrôleur sont-elles compatibles avec les grandeurs mesurées au thermomètre ?