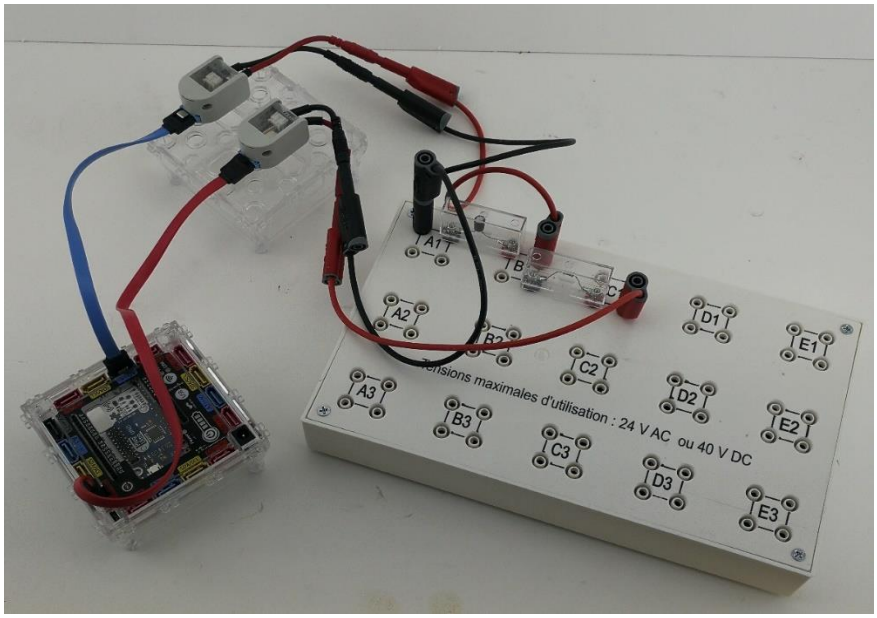
| Niveau : Tale Spé PC | Ondes et signaux : Etudier la dynamique d'un système électrique |
|---|---|
| Modèle du circuit RC série : charge d'un condensateur par une source idéale de tension, décharge d'un condensateur, temps caractéristique. | Étudier la réponse d'un dispositif modélisé par un dipôle RC.<br>Déterminer le temps caractéristique d'un dipôle RC à l'aide d'un microcontrôleur, d'une carte d'acquisition ou d'un oscilloscope.<br><br>Réaliser un montage électrique pour étudier la charge et la décharge d'un condensateur dans un circuit RC. |

# Dossier TP Plug'Uino® :

# Etude de la charge et décharge d'un condensateur

# Programmation en langage Python sur une carte Arduino



## 1. Résumé de l'activité

Cette activité expérimentale a pour but de :

- Mesurer et enregistrer à l'aide d'un microcontrôleur, l'évolution en fonction du temps de la tension aux bornes d'un condensateur lors de sa charge et de sa décharge.
- Déterminer le temps caractéristique d'un dipôle RC à l'aide d'un microcontrôleur.

## 2. Thème du programme abordé

| **Niveau : Tale** | **Ondes et signaux : Etudier la dynamique d'un système électrique** |
|---|---|
| Modèle du circuit RC série : charge d'un condensateur par une source idéale de tension, décharge d'un condensateur, temps caractéristique. | Étudier la réponse d'un dispositif modélisé par un dipôle RC.<br>Déterminer le temps caractéristique d'un dipôle RC à l'aide d'un microcontrôleur, d'une carte d'acquisition ou d'un oscilloscope.<br><br>Réaliser un montage électrique pour étudier la charge et la décharge d'un condensateur dans un circuit RC. |

## 3. Matériel mis en œuvre

1 interface Plug'Uino® Réf. 650 003

2 modules de connexion Plug'Uino®, SATA -Connecteurs 2 fiches bananes femelle Réf. 651 056

(Le module intègre une résistance de 1 kOhms qui s'ajoute en série dans le circuit RC)

2 cordons SATA 25 cm Réf. 655 013

1 boitier de montage Plug'Uino® Réf. 656 016

1 Platine de montage électronique Réf. 000 017

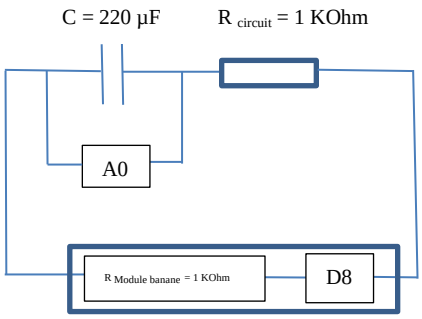
1 Boitier dipôle Condensateur 220 µF

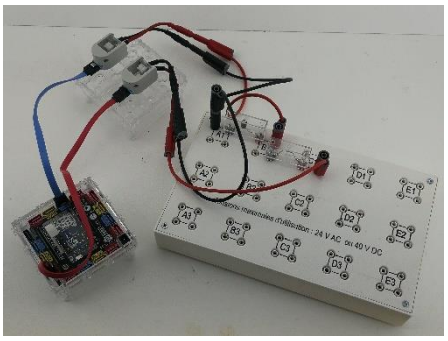
1 Boitier dipôle Résistance 1 kOhms

## 4. Présentation de l'expérience réalisée

Le montage comporte en série :

Un condensateur C = 220 µF, une résistance $R_{circuit}$= 1 KOhm, et la résistance interne de 1 KOhm du module Plug'Uino® de connexion SATA-Bananes qui alimente le circuit (connecté à la sortie D8).
À l'aide de la sortie numérique D8 du microcontrôleur, on applique une tension de 5 V aux bornes du circuit RC. On peut schématiser le montage comme ci-dessous :



Simultanément on mesure sur l'entrée analogique A0 du microcontrôleur, l'évolution de la tension aux bornes du condensateur (capacité), à l'aide du deuxième module de connexion banane Plug'Uino®.

Le programme calcule le temps caractéristique du circuit RC.

## 5. Programme Python

Le programme Python fonctionnel, permettant d'enregistrer et d'afficher le graphique de la tension est donné ci-après.

Ce programme est téléchargeable sur notre site au format «.py»

(mot de passe pour ouvrir le fichier téléchargé : SCIENCETHIC).

```
# /********** Charge et décharge d'un condensateur ***************
#  *
#  * On mesure la tension aux bornes d'un condensateur pendant qu'il
#  * se charge, puis pendant qu'il se décharge.  La mesure de la tension
#  * est prise toutes les 10 ms.
#  *
#  *
#  ************************************************************/

from pyfirmata2Ext import Arduino, util
import matplotlib.pyplot as plt # Importation de la bibliothéque néccéssaire pour tracer des graphiques
import time

#importation des bibliothèques

#------------------------------ paramétrage de la communication----------------------------------------------
port = Arduino(Arduino.AUTODETECT)              # 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:i')     # A0 désigne la broche analogique 0 (a:0) utilisée en entrée (:i)
port.digital[8].write(1)       # écriture sur le port digital 8
time.sleep(0.5)                # délai d'attente pour la mise en place de la communication




etat = 0# 0 au début, 1 pendant la charge, 2 pendant la décharge, 3 quand c'est fini

previousMillis = 0 # variable qui retient les anciennes valeurs en ms

interval = 10  # Nombre de millisecondes entre chaque mesure
```

```
# temps associer à la mesure
tempsdecharge=[]
tempsdedecharge=[]
TempsCaract=[]

# liste des tensions
tensionsdecharge=[]
tensionsdedecharge=[]
TensionCaract=[]

# Dans un premier temps, on s'assure que le condensateur est complètement décharg

port.digital[8].write(0) # Mise à l'état bas (0 V) de la sortie D8

time.sleep(2)      #Délai d'attente de 2000 ms



#Définition de "tempsCaract"comme la valeur du temps caractéristique
tempsCaract=[]


startms = 0



while etat < 3 :


    if etat == 0 :
        etat = 1 # nouvel état:charge du condensateur
        startms = time.time()*1000 # On enregistre le temps de départ en ms
        port.digital[8].write(0) # Mise à l'état bas (0 V) de la sortie D8
```

```
    currentMillis = time.time()*1000 - startms
    tension = 0.0#Définition de "tension" comme un nombre à virgule flottante

     #Définition de "TensionCaract"comme la valeur de la tension à t = Temps caractéristique


    if currentMillis - previousMillis >= interval :

        # Prendre une mesure à chaque intervalle de temps définit par la variable "interval" (interval = 10 ms)

        previousMillis = currentMillis

        if etat == 1 :   # Charge du condensateur avec une tension de 5 V

            port.digital[8].write(1)  # On alimente le circuit avec une tension de 5 V (délivrée par le microcontrôleur sur D8 )

            tension = 6.2*a0.read()  #Conversion de la valeur numérisée de A0 en une valeur de tension en Volts
            #(on fait x6.2 au lieux de 5 pour compenser le retard de communication entre la carte et l'ordinateur)



            tensionsdecharge.append(tension) # ajout de la tension dans le tableau tensionsdecharge
            tempsdecharge.append(currentMillis) # ajout du temps dans le tableau tempsdecharge

            if tension < 4.7 : # Si le condensateur n'est pas complètement chargé on rentre dans la condition ci-dessous

                if tension > 3.0 and tension < 3.2 : #Si la tension aux bornes du condensateur atteint 63% de la valeur max (5 V) alors...


                    #Enregistre le temps caractéristique dans le tableau TempsCaract
                    TensionCaract.append(0) # Courbe
                    TensionCaract.append(tension)# Courbe
                    TempsCaract.append(currentMillis-1/1000.0)# Courbe
                    TempsCaract.append(currentMillis)# Courbe
                    tempsCaract.append(currentMillis)# Légende de la Courbe


             else : # Sinon, si le condensateur est complètement chargé, on change d'état: décharge du condensateur

                 etat = 2

                 port.digital[8].write(0)  # On coupe l'alimentation du circuit, en mettant la sortie D8 à l'état bas (0 V)

         if etat == 2 : # décharge

             tension = 6.2*a0.read() #Conversion de la valeur numérisée de A0 en une valeur de tension en Volts.
             #(on fait x6.2 au lieux de 5 pour compenser le retard de communication entre la carte et l'ordinateur)
             tensionsdedecharge.append(tension)# ajout de la tension dans le tableau tensionsdedecharge
             tempsdedecharge.append(currentMillis)# ajout du temps dans le tableau tempsdedecharge
             if tension < 0.01 : # complètement déchargé


                 etat = 3  # on change d'état:  tout est terminé



port.exit()
```

```
##################################################################################################
#                  Tracé du graphique
##################################################################################################

 # pour réaliser des tracés
import matplotlib.pyplot as plt


fig, ax = plt.subplots()
ax.plot(tempsdecharge,tensionsdecharge,'r', label='Charge du Condensateur') # Courbe de Charge en rouge
ax.plot(TempsCaract,TensionCaract,'g', label='Temps Caractéristique = %s ms' %(round(min(tempsCaract),1)))# Courbe du Temps Caratéristique en vert
ax.plot(tempsdedecharge,tensionsdedecharge,'b', label='Décharge du Condensateur') # Courbe de Décharge en bleu

legend = ax.legend(loc='upper center', shadow=False, fontsize='x-large')
plt.ylabel('Tension(V)') #Légende
plt.xlabel('Temps(s)') #Légende
plt.show()

print("\nL exécution du programme s est terminée sans erreur ! \n\n")
```

Pour utiliser ce programme il faut commencer par installer la bibliothèque Python « pyFirmata2Ext » :

1er Etape :

Ouvrer le dossier pyFirmata2Ext.

2ème Etape :

Copier le contenue de pyFirmata2Ext dans :

C:\Users\NON_DE_LA_SESSION\AppData\Roaming\Python\Python37\site-packages

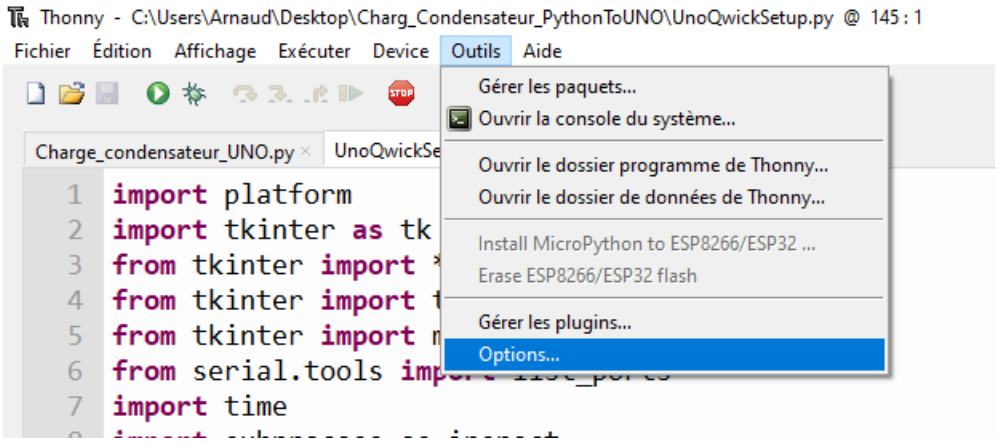
Ensuite avant de lancer le programme nous devons exécuter le programme « UnoQwickSetup.py »

1er Etape :

Ouvrer le programme UnoQwickSetup.py (avec Thonny).

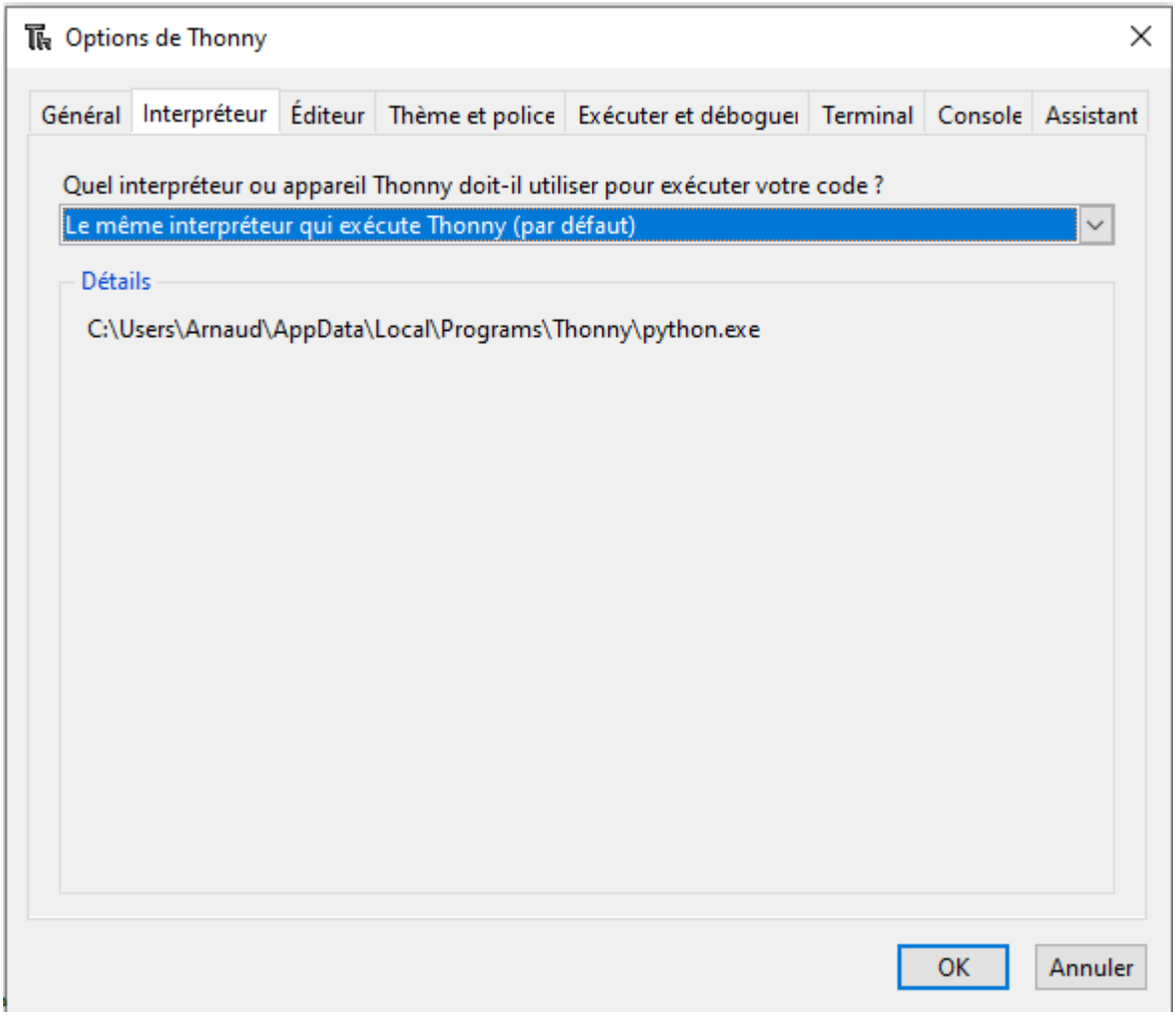
2ème Etape :

Aller dans l'onglet Outils, puis Options…



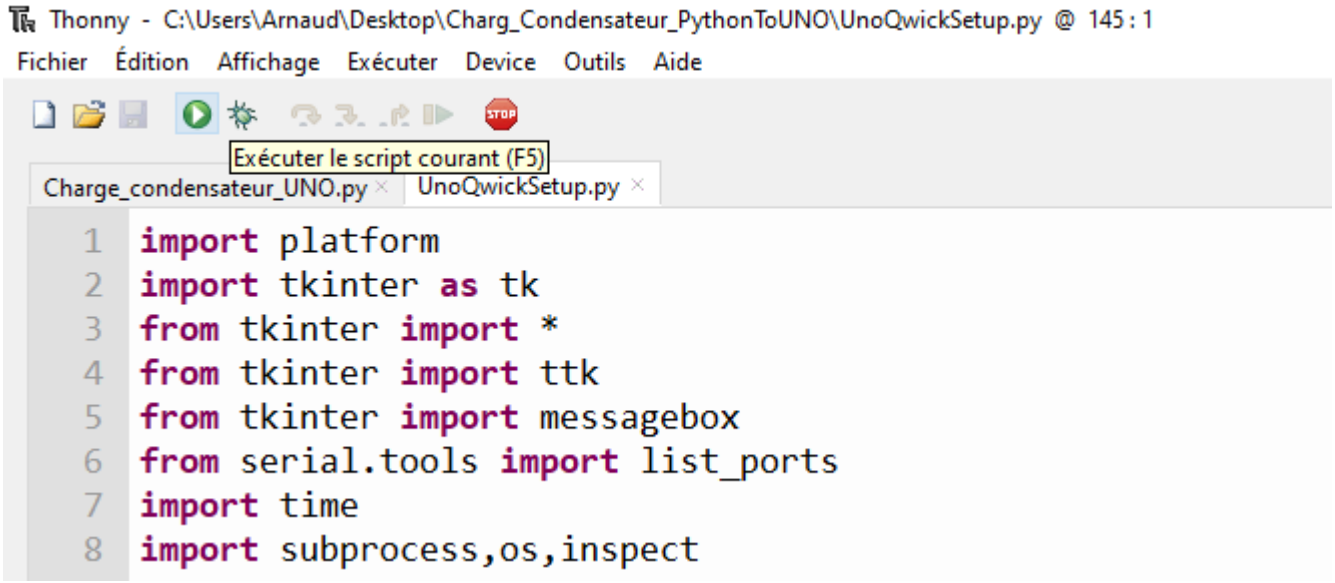
3ème Etape :

Aller dans l'onglet Interpréteur et sélectionné dans (Quel interpréteur ou appareil Thonny doit 'il utiliser pour exécuter votre code ? : « Le même interpréteur qui exécute Thonny(par défaut) »



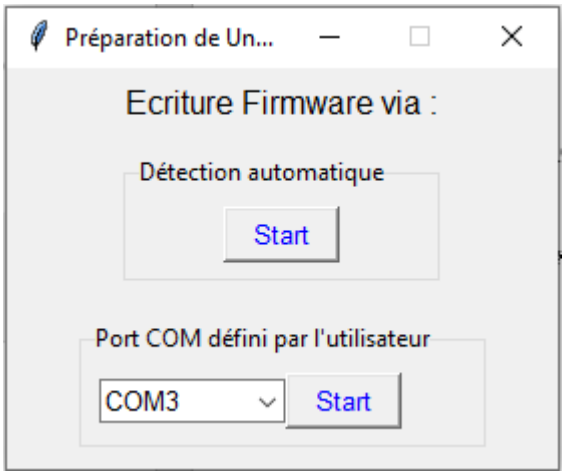

4ème Etape :

Lancer le code avec la flèche verte ou faisant F5 :

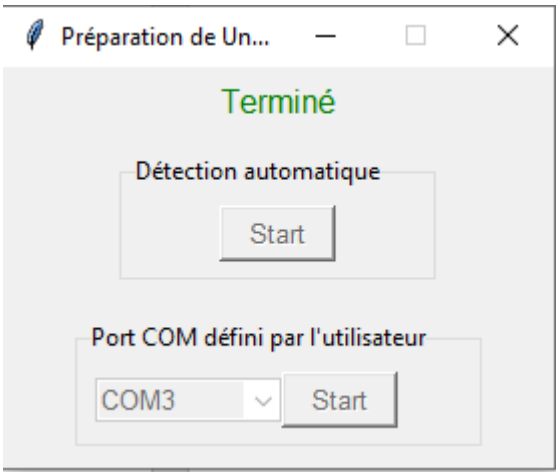



5ème Etape :

Une fenêtre s'ouvre, faite la détection automatique ou renseigner manuellement le port com sur le quel est connecté votre carte.

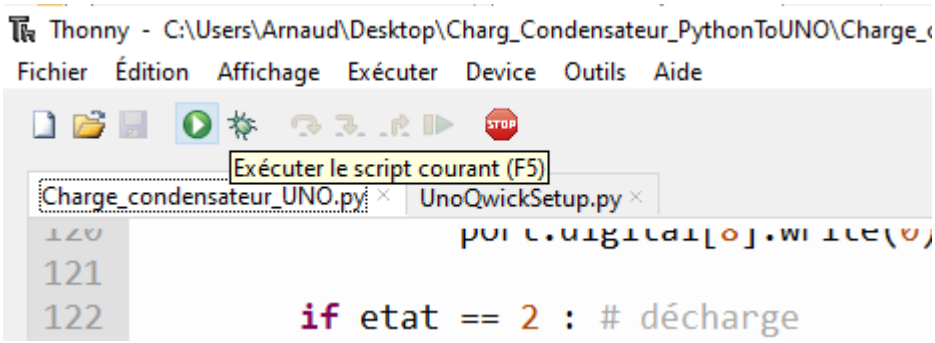


Puis faite Start et attendez que la séquence soit terminée.




Maintenant le code de la charge du condensateur est prêt à être exécuté.

Ouvrer le programme « Charge_condensateur_UNO.py » et exécutez le en cliquant sur la flèche verte ou en faisant F5.



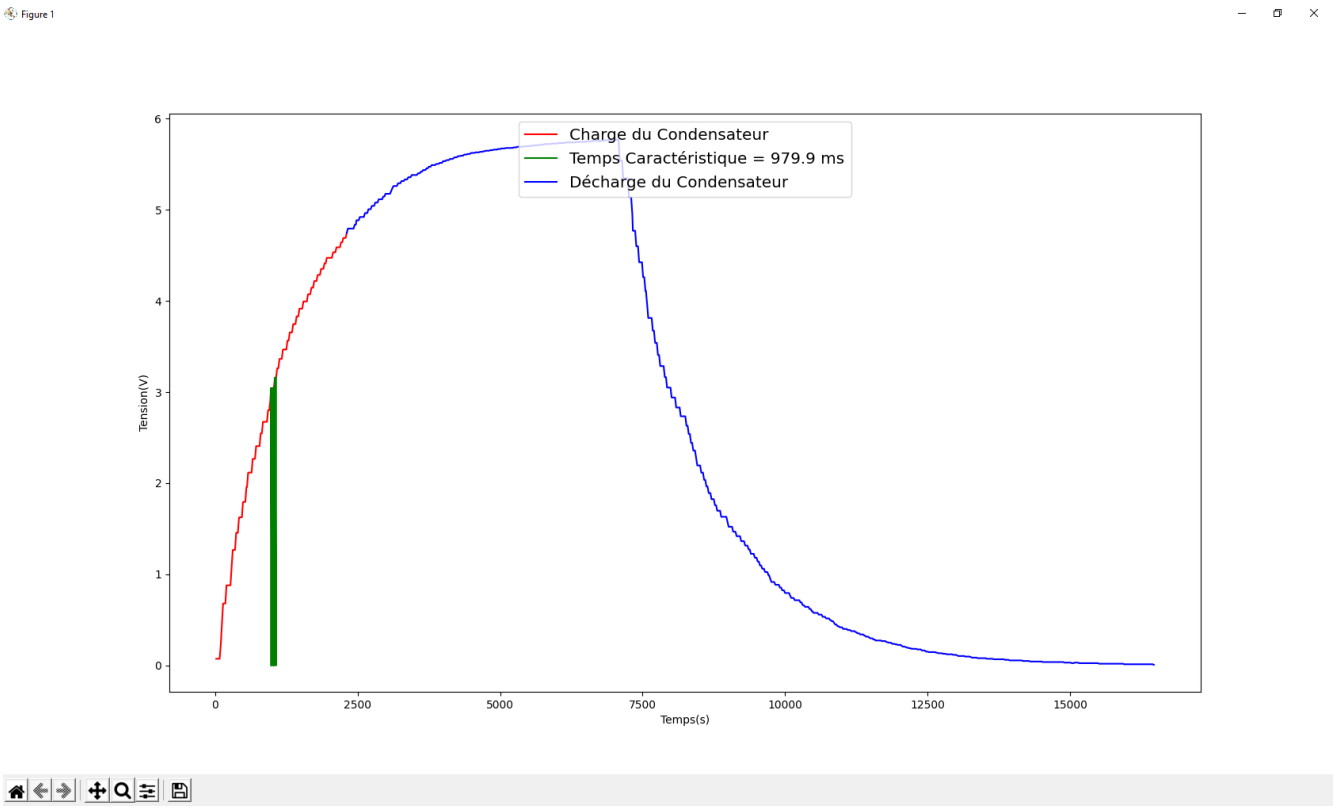

Attention le programme va mettre du temps avant d'afficher le graphique.

## 6. Résultats obtenus

Pour ce circuit RC, R = (1+1) KOhm, C= 220 µF, la valeur théorique du temps caractéristique est t = 440 ms.

Nous trouvons expérimentalement t = 979 ms, soit un écart de 122.5%.

Cet écart est inacceptable.



En supprimant la résistance de 1 KOhm du circuit, il reste donc uniquement la résistance de 1 kOhms du module connecteurs bananes Plug'Uino.
La valeur théorique du temps caractéristique est t = 0,00022x 1 000 = 220 ms
Nous trouvons expérimentalement t = 639 ms, soit un écart de 86.36 %
Cet écart est plus petit mais toujours inacceptable.

Pourquoi obtient-on des résultats erronés ?
Les courbes ont l'air correcte mais on peut observer que la charge est tracée en partie avec la couleur de la décharge. La charge est trop rapide par rapport à la vitesse de lecture ce qui entraine un retard et donne des valeurs erronées. En temps normal la carte Arduino est capable d'effectuer se programme sans souci particulier (voir Etude de la charge et décharge d'un condensateur – Langage Arduino). Ce qui change dans ce cas, c'est que le programme est codé en langage Python et que l'on fait appel à une bibliothèque pour l'interpréter sur une carte qui utilise le langage Arduino. Ce processus prend du temps et cela cause un retard.

### 7. Conclusion

Si on veut réaliser ce TP avec un langage python il faut utiliser une carte utilisant le même langage.
Example : voir Etude de la charge et décharge d'un condensateur – Langage Python