Acquisition point par point

1. Introduction

Dans certaines expériences, l'acquisition peut se faire point par point, c'est-à-dire sans utiliser l'échantillonneur du CAN pour cadencer les mesures. L'acquisition peut être déclenchée par l'utilisateur, ou bien à intervalle de temps régulier (intervalle de l'ordre de la seconde ou plus).

Dans l'exemple ci-dessous, un pont diviseur constitué d'une résistance de 47 K et d'une thermistance de 47 K est alimenté par l'alimentation +5 V du CAN SysamSP5. La tension aux bornes de la résistance est lue sur l'entrée EA0. Pendant l'acquisition, la thermistance est légèrement chauffée (à la main) pour faire varier la tension mesurée.

2. Lecture directe

La première méthode consiste à utiliser la lecture directe des entrées. Les mesures sont faites de manière régulière, en effectuant un échantillonnage (approximatif) avec l'horloge de l'ordinateur. On commence par ouvrir l'interface et configurer l'entrée EA0 :

```
import pycan.main as pycan
import matplotlib.pyplot as plt
import numpy
import time

sys = pycan.Sysam("SP5")
sys.config_entrees([0],[10])
```

Choix de la période d'échantillonnage, du nombre de points et création des tableaux :

```
te = 5.0
ne = 30
listeEA0 = numpy.zeros(ne,dtype=float)
listeTemps = numpy.zeros(ne,dtype=float)
```

La commande suivante permet d'activer la lecture directe de l'entrée EA0 :

```
sys.activer_lecture([0])
```

Boucle d'acquisition :

```
for k in range(ne):
    tensions=sys.lire() # lecture directe
    listeTemps[k] = k*te # instant voulu
    listeEA0[k] = tensions[0]
    print("t = %f, u0 = %f"%(listeTemps[k],listeEA0[k]))
```

```
    t_clock_new = time.clock()
    t_reel = t_reel + t_clock_new-t_clock # instant réel
    t_clock = t_clock_new
    print("t reel = %f"%t_reel)
    time.sleep((k+1)*te-t_reel) # calage sur le temps réel à chaque itération
```

Remarquer la fonction `time.sleep`, qui effectue une attente de manière à effectuer l'acquisition suivante à l'instant voulu (à peu près). Dans cet exemple, le temps voulu est stocké, mais on pourrait aussi stocker le temps réel. On vérifie avec les valeurs affichées que le décalage entre le temps voulu et le temps réel reste très faible. Cependant, l'ordinateur n'est pas un système temps réel et donc la régularité de l'échantillonnage ne peut être garantie, bien qu'elle soit très probable pour des périodes aussi grandes. En conséquence, il est préférable de stocker le temps réel calculé dans la boucle.

Enfin on désactive la lecture directe puis on ferme l'interface :

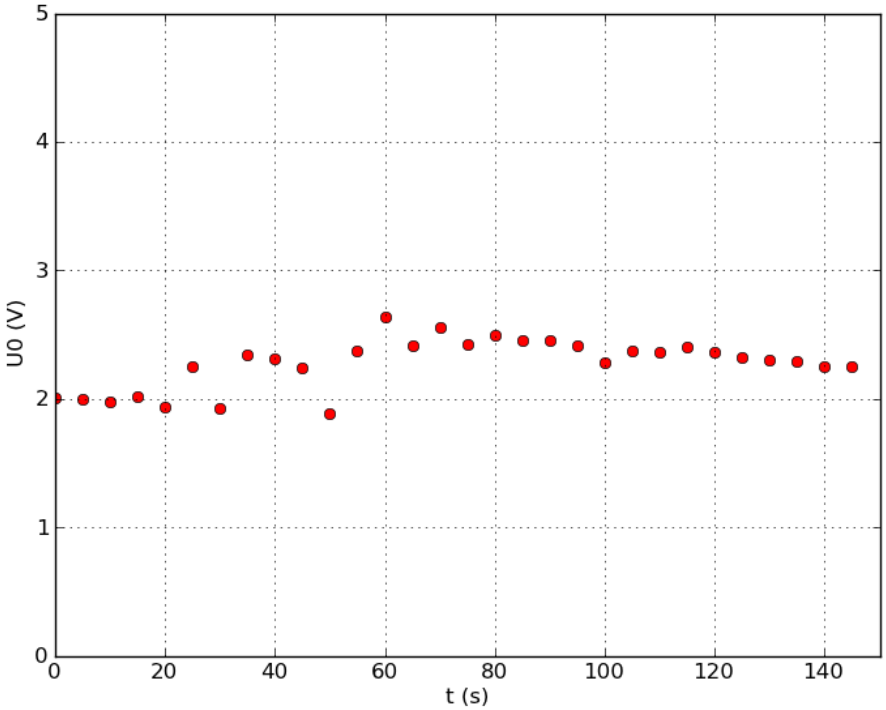
```
sys.desactiver_lecture()
sys.fermer()
```

Sauvegarde des données pour une utilisation ultérieure :

```
numpy.savetxt('entreeDirecte-data-1.txt',[listeTemps,listeEA0])
```

Tracé de la tension en fonction du temps :

```
plt.figure()
plt.plot(listeTemps,listeEA0,'ro')
plt.axis([0,ne*te,0.0,5.0])
plt.xlabel("t (s)")
plt.ylabel("U0 (V)")
plt.grid()
plt.show()
```



Les points ne semblent pas suivre une courbe régulière. Cela est dû au fait que la tension mesurée est très fluctuante. Dans ce cas, la lecture directe n'est pas une bonne méthode.

3. Lecture par paquet échantillonné

Dans le cas où la tension mesurée fluctue fortement, il faut effectuer l'acquisition de paquets échantillonnés afin de calculer la valeur moyenne. Cette méthode offre en plus l'avantage de fournir un écart type, c'est-à-dire une incertidude pour les mesures (incertitude liée aux fluctuations).

Le début est identique à l'exemple ci-dessus :

```
te = 5.0 # période d'échantillonnage en secondes
ne = 40 # nombre d'échantillons
listeEA0 = numpy.zeros(ne,dtype=float)
listeTemps = numpy.zeros(ne,dtype=float)
listeEcarts = numpy.zeros(ne,dtype=float)
sys = pycan.Sysam("SP5")
sys.config_entrees([0],[10])
```

On fixe la durée des paquets d'acquisition et le nombre de points :

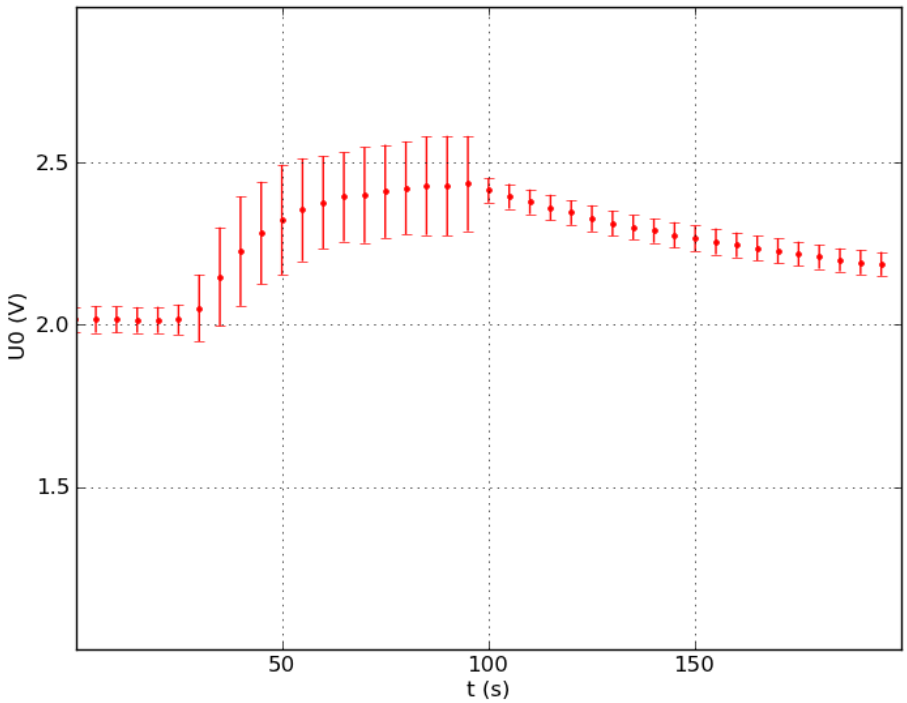
```
ta = te/10.0 # durée de l'acquisition
na = 1000 # nombre de points de l'acquisition
tea = ta/na # période d'échantillonnage de l'acquisition
sys.config_echantillon(tea*1e6,na)
```

Dans la boucle de mesure, on calcule la moyenne des valeurs du paquet et l'écart type :

```
t_clock = time.clock()
t_reel = 0.0
for k in range(ne):
    sys.acquerir()
    tensions=sys.entrees()
    moyenne = numpy.mean(tensions[0])
    ecart = numpy.std(tensions[0])
    listeTemps[k] = k*te
    listeEA0[k] = moyenne
    listeEcarts[k] = ecart
    print("t = %f, u0 = %f, ecart = %f"%(listeTemps[k],listeEA0[k],listeEcarts[k]))
    t_clock_new = time.clock()
    t_reel = t_reel + t_clock_new-t_clock
    t_clock = t_clock_new
    print("t reel = %f"%t_reel)
    time.sleep((k+1)*te-t_reel)
sys.fermer()
numpy.savetxt('entreePointParPoint-data-1.txt',[listeTemps,listeEA0,listeEcarts])
```

On trace la tension en fonction du temps avec des barres d'incertitude :

```
plt.figure()
plt.plot(listeTemps,listeEA0,marker='.',linestyle='',color='r')
plt.errorbar(listeTemps,listeEA0,yerr=listeEcarts,xerr=None,fmt=None,ecolor='r')
plt.axis([0,ne*te,1.0,3.0])
plt.xlabel("t (s)")
plt.ylabel("U0 (V)")
plt.grid()
plt.show()
```



On voit à présent que la tension est très fluctuante lorsque la thermistance est tenue entre les doigts pour la chauffer.