# Utilisation des sorties

## 1. Génération d'un signal échantillonné sur une sortie

La carte SysamPCI possède un convertisseur N/A (sortie SA1). La centrale SysamSP5 possède deux convertisseurs N/A (sorties SA1 et SA2).

On commence par charger les modules et ouvrir l'interface :

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

sys = pycan.Sysam("SP5")
```

Pour configurer une sortie, il faut tout d'abord créer un tableau numpy (ndarray) contenant les valeurs des échantillons en volts. Dans le cas d'un signal périodique, on peut se contenter d'échantillonner une période (voir le document Synthèse numérique d'un signal périodique pour l'inconvénient de cette méthode). Voici par exemple la génération d'une sinusoïde d'amplitude 5 volts :

```
ns = 1000 # nombre de points
v1 = 5.0*numpy.sin(2*numpy.pi*numpy.arange(ns)/ns)
```

La configuration d'une sortie se fait en précisant la période d'échantillonnage en microsecondes, les échantillons, et le nombre de répétitions des échantillons fournis. Si l'on veut une répétition périodique sans fin, on choisit ce nombre à -1 :

```
tes = 1.0e-5 # période d'échantillonnage en secondes
sys.config_sortie(1,tes*1e6,v1,-1)
```

On déclenche la sortie SA1 au moment souhaité :

```
sys.declencher_sorties(1,0)
```

Sur SysamPCI, la fonction précédente est bloquante et l'émission de la sortie périodique est stoppée par un appui sur la touche ESC. Avec la SysamSP5, la fonction précédente retourne immédiatement. Il faut placer une fonction d'attente avant de stopper la sortie :

```
time.sleep(10) # on laisse la sortie émettre pendant 10 secondes
sys.stopper_sorties(1,0)
sys.fermer()
```

On peut aussi utiliser la fonction `matplotlib.pyplot.show` pour bloquer l'exécution, ce qui laisse l'émission se faire jusqu'à fermeture de la fenêtre.

Remarque : la période d'échantillonnage effective est nécessairement multiple de la période d'échantillonnage de base, qui est de $0.2\ \mu$s sur SysamSP5. Pour générer un signal de fréquence arbitraire, il faut l'échantillonner sur plusieurs périodes, comme expliqué dans le document Synthèse numérique d'un signal périodique.

## 2. Utilisation des entrées et sorties (SysamSP5)

Sur la SysamSP5, il est possible de déclencher les sorties, puis une acquisition.

Dans l'exemple suivant, les sorties génèrent deux sinusoïdes déphasées avec une période de 100 ms. Les tensions des deux sorties sont acquises sur les voies EA0 et EA1. Un signal TTL de période 10 ms est acquis sur la voie EA2. Les périodes d'échantillonnage des entrées et sorties sont choisies identiques (10 microsecondes).

Configuration des entrées :

```
sys=pycan.Sysam("SP5")
sys.config_entrees([0,1,2],[10,10,10])
ne=20000
te=1e-5
sys.config_echantillon(te*1e6,ne)
```

Calcul des échantillons pour les sorties et configuration des sorties :

```
ns=10000
tes=te
v1 = 5.0*numpy.cos(2*numpy.pi*numpy.arange(ns)/ns)
v2 = 5.0*numpy.cos(2*numpy.pi*numpy.arange(ns)/ns + numpy.pi/2)
sys.config_sortie(1,tes*1e6,v1,-1)
sys.config_sortie(2,tes*1e6,v2,-1)
```

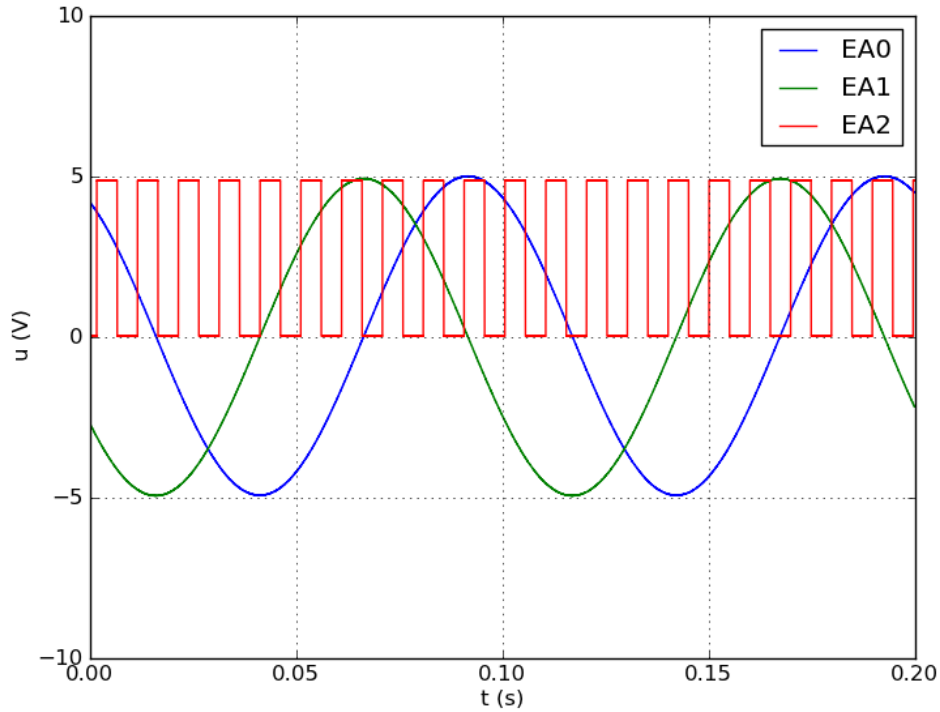
On déclenche les sorties, puis on attend 0.5 secondes avant de déclencher l'acquisition des entrées :

```
sys.declencher_sorties(1,1)
time.sleep(0.5)
sys.acquerir()
```

Récupération des données et tracé des courbes :

```
t=sys.temps()
u=sys.entrees()
sys.fermer()
plt.figure()
plt.plot(t[0],u[0],label="EA0")
plt.plot(t[1],u[1],label="EA1")
```

```
plt.plot(t[2],u[2],label="EA2")
plt.axis([0,te*ne,-10.0,10.0])
plt.xlabel("t (s)")
plt.ylabel("u (V)")
plt.grid()
plt.legend()
plt.show()
```




## 3. Utilisation simultanée des entrées et sorties

Sur SysamPCI, la méthode précédente est impossible car l'appel de la fonction `declencher_sorties(` est bloquant.

On peut faire une utilisation simultanée des entrées et sorties avec la fonction `acquerir_avec_sortie` qui fonctionne aussi sur SysamSP5.

Dans l'exemple suivant, une rampe est générée sur chaque sortie. Les tensions délivrées par les deux sorties sont acquises sur les voies EA0 et EA1. Un signal TTL de période 10 ms est acquis sur la voie EA2.

On génère les tableaux des échantillons pour les sorties :

```
sys=pycan.Sysam("SP5")
T=0.1
ne=10000
te=T/ne
v1 = numpy.zeros(ne,numpy.double)
v2 = numpy.zeros(ne,numpy.double)
for k in range(ne-2): # rampe  de 0 à 5.0 V, montante pour SA1, descendante pour SA2
    v1[k] = k*5.0/(ne-1)
```

```
    v2[k] = 5.0-v1[k]
```

Les deux dernières valeurs des tableaux sont laissées nulles de manière à annuler les tensions des sorties à la fin de l'acquisition.

On configure l'acquisition :

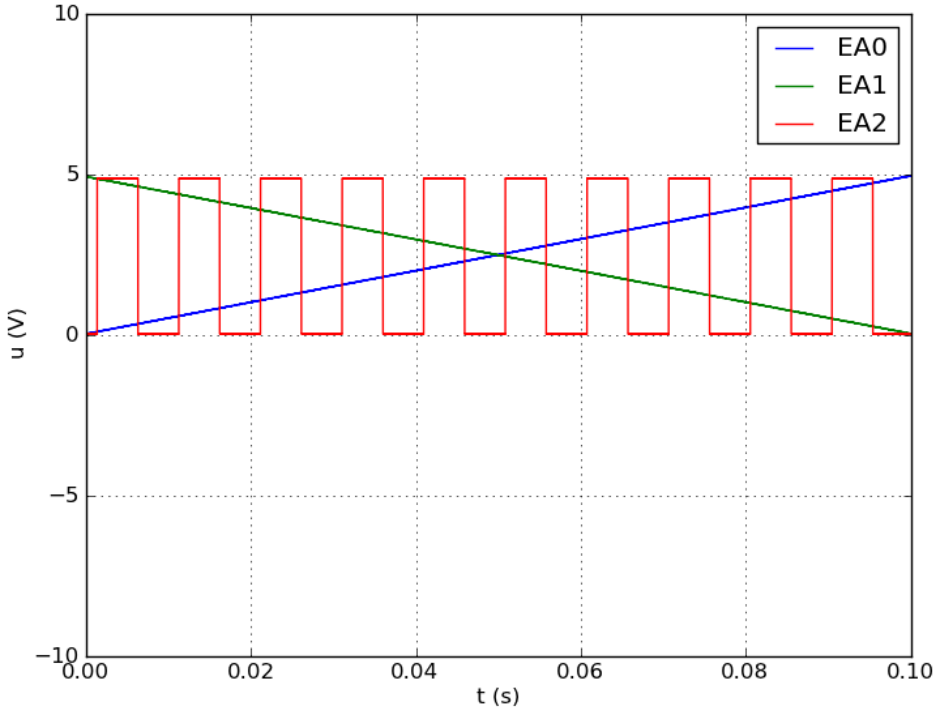
```
sys.config_entrees([0,1,2],[10,10,10])
sys.config_echantillon(te*1e6,ne)
```

Puis on déclenche l'acquisition et les sorties simultanément. La période d'échantillonnage des sorties est identique à celle des entrées, de manière à assurer la synchronisation.

```
sys.acquerir_avec_sorties(v1,v2)
```

Enfin on récupère les données et on trace les courbes :

```
t=sys.temps()
u=sys.entrees()
sys.fermer()
plt.figure()
plt.plot(t[0],u[0],label="EA0")
plt.plot(t[1],u[1],label="EA1")
plt.plot(t[2],u[2],label="EA2")
plt.xlabel("t (s)")
plt.ylabel("u (V)")
plt.axis([0,ne*te,-10.0,10.0])
plt.grid()
plt.legend()
plt.show()
```


EA0
EA1
EA2
10
5
0
−5
−10
u (V)
0.00
0.02
0.04
0.06
0.08
0.10
t (s)