CAN Eurosmart : interface pour Python

1. Installation

Ce document présente une interface Python de la centrale SysamSP5 de Eurosmart pour Python.
La dernière version de l'interface (version 5.0.x) fonctionne en Python 32 bits et 64 bits mais ne contient pas les fonctions complémentaires de pilotage de la sortie audio. L'interface pour Sysam PCI n'est plus implémentée. Cette nouvelle version est constituée d'un exécutable 32 bits, qui utilise le pilote Eurosmart 32 bits et qui implémente un serveur HTTP. La partie Python de l'interface communique avec ce serveur par des requêtes HTTP.
Cette nouvelle version fait appel au module requests. Il est donc conseillé d'avoir une version à jour de ce module.
Distribution par PyPI (pypi.org/project/pycanum/). Pour l'installer, ouvrir une console et exécuter les commandes suivantes dans le répertoire Scripts de la distribution (après avoir exécuté `activate.bat`) :

```
pip install --upgrade pip
pip install pycanum
```

Pour faire une mise à jour :

```
pip install --upgrade pycanum
```

Version 5.0.1 : correction d'un bug dans la configuration de la sortie 2.
Version 5.0.0 : première version fonctionnant en Python 64 bits.
Pour installer le pilote de la centrale SysamSP5, télécharger le fichier zip sur Eurosmart et exécuter le programme d'installation. Remarque : sous windows 10 64 bits, on exécute DPInst64.exe mais le pilote installé est en 32 bits. La DLL SP5.dll se trouve dans le dossier `c:\Windows\SysWOW64`, qui contient les DLL 32 bits sur un windows 64 bits.
Le serveur qui gère les requêtes HTTP est implémenté dans l'exécutable nommé `sysamhttp.exe`.
Si ce serveur ne répond plus, il doit être arrêté (CTRL ALT SUPR puis Gestionnaire des tâches).
Scripts de démonstration :

- ▷ Acquisition sur une voie et analyse spectrale
- ▷ Acquisition sur deux voies et analyse spectrale
- ▷ Acquisition avec déclenchement
- ▷ Synthèse d'un signal périodique sur une sortie
- ▷ Acquisition avec utilisation simultanée d'une sortie
- ▷ Acquisition d'un signal, filtrage puis restitution en sortie
- ▷ Boucle de filtrage d'un signal
- ▷ Acquisition et tracé en parallèle

- Acquisition en mode permanent avec tracé du signal
- Acquisition en continu avec tracé du signal et de son spectre

Pages de documentation avec exemples :

- Enregistrement d'un signal
- Déclenchement de l'acquisition
- Acquisition point par point
- Utilisation des sorties
- Mesure de fréquence
- Diagramme de Bode
- Caractéristique d'un dipôle
- Filtrage numérique d'un signal
- Acquisition et traitement en parallèle
- Acquisition en mode permanent

Ancienne version pour Python 3.9 32 bits : pycanum-4.2.7-cp39-cp39-win32.whl.

2. Fonctions de l'interface

2.a. Ouverture d'une liaison avec le CAN

L'interface se trouve dans le module `pycanum.main`. Elle est constituée d'une classe `Sysam`. L'ouverture d'une liaison avec un convertisseur se fait en construisant un objet de la classe `Sysam`.

```
sys = Sysam(nom)
```

Constructeur : ouverture de la liaison avec le CAN

- `nom (string)` : nom du convertisseur (SP5 ou PCI)

```
Sysam.fermer()
```

Fermeture de la liaison
Attention : un seul objet de la classe Sysam peut être ouvert dans un programme Python. Il faut donc fermer la liaison avant de créer un nouvel objet Sysam.

2.b. Configuration des entrées analogiques

```
Sysam.config_entrees(voies,calibres,diff=[])
```

Sélection des entrées analogiques et configuration du calibre (gain de l'amplificateur d'entrée).

- `voies (list)` : liste des entrées analogiques à sélectionner, numérotées de 0 à 7.

- ▷ `calibres (list)` : liste des valeurs absolues maximales des tensions (en volts), une pour chaque voie sélectionnée.
- ▷ `diff (list)` : argument optionnel, liste des voies en mode différentiel.

Sur la carte SysamPCI, la fréquence d'échantillonnage maximale diminue lorsqu'on augmente le nombre de voies sélectionnées. Sur la centrale SysamSP5, il faut sélectionner les entrées 0,1,2 et 3 pour bénéficier de la fréquence d'échantillonnage maximale (10 MHz).
Sur SysamPCI, la sélection d'une seule voie en mode différentiel les place toutes en mode différentiel. Les tensions sont alors mesurées entre les entrées 0 et 4, 1 et 5, 2 et 6, 3 et 7. Sur SysamSP5, chaque canal (0-4,1-5,2-6,3-7) peut être placé en mode différentiel indépendamment des autres. Pour placer le premier et le deuxième canal en mode différentiel, il faut affecter [0,1] au dernier argument (diff).

2.c. Utilisation des entrées analogiques avec échantillonnage

Les fonctions décrites ci-dessous permettent d'effectuer des acquisitions échantillonnées sur les entrées analogiques.

```
Sysam.config_echantillon(techant,nbpoints)
```

Configuration de la période d'échantillonnage et du nombre de points à acquérir.

- ▷ `techant` : période d'échantillonnage en microsecondes
- ▷ `nbpoints` : nombre de points à acquérir

La période d'échantillonnage effectivement utilisée est multiple de la période d'horloge du convertisseur. Le nombre de points est limité par la capacité mémoire de la carte.
Le convertisseur de la carte Sysam effectue une numérisation en 12 bits mais l'interface permet de réduire le nombre de bits utilisés pour le calcul de la tension (les bits les moins significatifs sont éliminés par un décalage à droite) :

```
Sysam.config_quantification(nbits)
```

Configuration du nombre de bits de la quantification.

- ▷ `nbits` : nombre de bits utilisés pour la quantification (inférieur ou égal à 12).

```
Sysam.config_trigger(voie,seuil,montant=1,pretrigger=0,pretriggerSoup
```

Configuration d'un déclenchement sur une des voies de l'acquisition.

- ▷ `voie (integer)` : voie sur laquelle se fait le déclenchement (une des voies configurées).
- ▷ `seuil (float)` : seuil du déclenchement en volts.
- ▷ `montant (integer)` : 1 pour un déclenchement sur front montant (par défaut), 0 pour un déclenchement sur front descendant.
- ▷ `pretrigger (integer)` : option pour SysamSP5, nombre de points enregistrés avant la condition de déclenchement.

- ▷ `pretriggerSouple (integer)` : option pour SysamSP5. 1 si le déclenchement doit se faire lorsque le nombre de points de pretrigger n'est pas atteint.
- ▷ `hystereris (integer)` : option pour SysamSP5. Hystérésis du déclenchement.

Attention : la valeur `pretrigger=0` ne conduit pas au déclenchement attendu. Il faut choisir au minimum `pretrigger=1`.

```
Sysam.config_trigger_externe(pretrigger=0,pretriggerSouple=0)
```

Configuration d'un déclenchement sur une voie externe TTL (entrée SYNCHRO EXT).

- ▷ `pretrigger (integer)` : option pour SysamSP5, nombre de points enregistrés avant la condition de déclenchement.
- ▷ `pretriggerSouple (integer)` : option pour SysamSP5. 1 si le déclenchement doit se faire lorsque le nombre de points de pretrigger n'est pas atteint.

```
Sysam.desactiver_trigger()
```

Désactivation du déclenchement par un signal (voie acquise ou externe).

```
Sysam.acquerir()
```

Acquisition des tensions sur les entrées configurées. La fonction retourne lorsque l'acquisition est terminée.

```
temps = Sysam.temps(reduction=1)
```

Récupération des temps de la dernière acquisisition.

- ▷ `reduction (integer)` : facteur de réduction de la fréquence d'échantillonnage.
- ▷ `temps (ndarray)` : tableau ndarray (numpy). Chaque ligne du tableau fournit les temps (en s) échantillonnés de la voie correspondante.

Remarque : sur SysamPCI, les temps des différentes voies sont décalés.

```
tensions = Sysam.entrees(reduction=1)
```

Récupération des tensions de la dernière acquisition.

- ▷ `reduction (integer)` : facteur de réduction de la fréquence d'échantillonnage.
- ▷ `tensions (ndarray)` : tableau ndaray (numpy). Chaque ligne du tableau fournit les tensions (en V) de la voie correspondante.

Un facteur de réduction égal à 2 permet d'obtenir un échantillon sur 2.
Les temps et les tensions de la première voie sélectionnée sont `temps[0]` et `tensions[0]`.

2.d. Utilisation des sorties analogiques avec échantillonnage

Les fonctions décrites ci-dessous permettent d'émettre un signal échantillonné sur la sortie SA1 (SysamPCI), ou sur les deux sorties SA1 et SA2 (SysamSP5). Pour une utilisation simultanée des entrées, voir le paragraphe suivant.

```
Sysam.config_sortie(nsortie,techant,tensions,repetition=0)
```

Configuration d'une sortie

- ▷ `nsortie (integer)` : numéro de la sortie (1 ou 2)
- ▷ `techant (float)` : période d'échantillonnage en microsecondes
- ▷ `tensions (ndarray)` : tableau ndarray (numpy) fournissant le signal échantillonné à appliquer sur la sortie (valeurs en volts).
- ▷ `repetition (integer)` : nombre de répétitions du signal, -1 pour une répétition périodique.

Sur la carte SysamSP5, l'argument `repetition` est égal à -1 ou 0. Pour répéter 2 fois ou plus un signal donné, il faut donc programmer cette répétition dans le signal échantillonné fourni.

```
Sysam.declencher_sorties(s1,s2)
```

Déclenchement d'une ou deux sorties. Les deux sorties (SysamSP5) sont déclenchées simultanément. Pour qu'elles soient synchrones, il faut néanmoins que les périodes d'échantillonnage soient identiques. Sur SysamSP5, la fonction retourne immédiatement. Sur SysamPCI, elle retourne lorsque l'émission est terminée ou lorsque l'utilisateur presse la touche ESC.

- ▷ `s1 (integer)` : 1 pour déclencher la sortie 1, 0 sinon.
- ▷ `s2 (integer)` : 1 pour déclencher la sortie 2, 0 sinon.

Remarque : sur SysamSP5, la fonction retourne immédiatement ; il faut donc éventuellement prévoir une mise en attente avant d'effectuer d'autres opérations.

```
Sysam.stopper_sorties(s1,s2)
```

Stopper une émission périodique sur SysamSP5. Sur SysamPCI, il faut appuyer sur la touche ESC pour stopper l'émission.

- ▷ `s1 (integer)` : 1 pour stopper la sortie 1, 0 sinon.
- ▷ `s2 (integer)` : 1 pour stopper la sortie 2, 0 sinon.

Sur SysamSP5, il faut stopper les sorties avant de les configurer (si elles sont déjà en fonctionnement). Ne pas le faire peut provoquer des erreurs dans le remplissage de la mémoire du convertisseur numérique-analogique.
Sur SysamSP5, il est possible de déclencher une acquisition lorsque les sorties sont en fonctionnement. Par exemple, si les sorties sont en fonctionnement périodique, une acquisition peut être déclenchée à tout moment. Dans ce cas, il faudra toutefois que le nombre de points total utilisés par les entrées et les sorties n'excède par 0x3FFFF, soit 262142. Dans le cas contraire, une erreur est déclenchée à l'appel de la fonction `Sysam.acquerir`.

2.e. Utilisation simultanée des entrées et sorties

Sur SysamSP5, il est possible de déclencher les sorties puis l'acquisition après (mais pas l'inverse). Sur SysamPCI, cela est impossible avec les fonctions décrites plus haut. La fonction suivante permet d'effectuer simultanément une acquisition et une émission des sorties :

```
Sysam.acquerir_avec_sorties(tensions1,tensions2)
```

Effectuer l'acquisition avec une utilisation simultanée et synchrone des sorties. La période d'échantillonnage des sorties est la même que pour les entrées.

- ▷ `tensions1 (ndarray)` : tableau ndarray (numpy) fournissant le signal échantillonné à appliquer sur la sortie 1 (en volts).
- ▷ `tensions2 (ndarray)` : tableau ndarray (numpy) fournissant le signal échantillonné à appliquer sur la sortie 2 (en volts).

Remarque : le nombre de points total utilisable pour les entrées et sorties est limité par la mémoire RAM du SysamSP5. Il est de 0x3FFFF, soit 262142.

2.f. Lecture des entrées sans échantillonnage

La lecture directe des entrées (sans passer par la mémoire du CAN) peut être utilisée lorsque la cadence d'acquisition est très lente (moins de un échantillon par seconde). Dans ce cas, on utilisera l'horloge interne du PC pour déclencher des lectures périodiques (fonction time.sleep).

```
Sysam.activer_lecture(voies)
```

Activer la lecture directe sur une ou plusieurs voies, qui doivent avoir été configurées au préalable avec `config_entrees`.

- ▷ `voies (list)` : liste des voies.

```
tensions = Sysam.lire()
```

Lecture directe des tensions sur les entrées.

- ▷ `tensions (list)` : liste des tensions lues.

```
Sysam.desactiver_lecture()
```

Désactivation de la lecture directe.

2.g. Écriture directe sur les sorties

```
Sysam.ecrire(s1,tension1,s2,tension2)
```

Appliquer une tension constante sur une ou deux sorties

- ▷ `s1 (integer)` : 1 pour changer la tension sur la sortie 1, 0 sinon.
- ▷ `tension1 (float)` : tension à appliquer sur la sortie 1 en volts.
- ▷ `s2 (integer)` : 1 pour changer la tension sur la sortie 2, 0 sinon.
- ▷ `tension2 (float)` : tension à appliquer sur la sortie 2 en volts.

2.h. Acquisition en mode parallèle (SysamSP5)

Lorsque le temps d'acquisition est long (supérieur à quelques secondes), il peut être utile de l'effectuer sur une tâche d'exécution parallèle (thread) afin d'effectuer un traitement des données simultanément. Les fonctions suivantes permettent d'effectuer une acquisition en mode parallèle, sur le CAN SysamSP5 seulement. Pour le CAN SysamPCI, le pilote ne permet pas ce mode d'acquisition.

```
Sysam.lancer()
```

Lancer l'acquisition en mode parallèle. La fonction retourne dès que l'acquisition est lancée.

```
Sysam.lancer_avec_sorties(tensions1,tensions2)
```

Lancer l'acquisition en mode parallèle, avec utilisation simultanée des sorties.

- ▷ `tensions1 (ndarray)` : tableau ndarray (numpy) fournissant le signal échantillonné à appliquer sur la sortie 1 (en volts).
- ▷ `tensions2 (ndarray)` : tableau ndarray (numpy) fournissant le signal échantillonné à appliquer sur la sortie 2 (en volts).

```
A = Sysam.paquet(premier,reduction=1)
```

Lire le paquet de points déjà acquis lors d'une acquisition en mode parallèle.

- ▷ `premier (integer)` : indice du premier point à récupérer, ou -1 pour obtenir un paquet lors d'une acquisition permanente en boucle.
- ▷ `reduction (integer)` : facteur de réduction de la fréquence d'échantillonnage.
- ▷ `A (ndarray)` : tableau ndarray (numpy) contenant les données. Si N est le nombre de voies acquises, les N premières lignes sont les instants de ces N voies, les N lignes suivantes sont les tensions numérisées, et les N dernières lignes sont les tensions numérisées et filtrées.

2.i. Filtrage numérique (SysamSP5)

Les tensions lues sur les entrées analogiques sont filtrées pendant l'acquisition. La relation de récurrence appliquée est la suivante :

$$a_0 y_n = \sum_{k=0}^{N-1} b_k x_{n-k} - \sum_{k=1}^{M-1} a_k y_{n-k} \tag{1}$$

Pour définir un filtre à réponse impulsionnelle finie (RIF), on doit définir les coefficients a_k. Pour définir un filtre à réponse impulsionnelle infinie (RII), il faut aussi définir les coefficients b_k.

```
Sysam.config_filtre(a,b)
```

Définition des coefficients de filtrage

- ▷ `a (list)` : liste (ou tableau ndarray) définissant les coefficients a.
- ▷ `b (list)` : liste (ou tableau ndarray) définissant les coefficients b.

Si cette fonction n'est pas appelée, le filtre par défaut est défini par $a_0 = 1, b_0 = 1$ (filtre identité).
Pour obtenir les valeurs des tensions filtrées, on utilise la fonction suivante :

```
tensions = Sysam.entrees_filtrees(reduction=1)
```

Récupération des tensions filtrées de la dernière acquisition.

- ▷ `reduction (integer)` : facteur de réduction de la fréquence d'échantillonnage.
- ▷ `tensions (ndarray)` : tableau ndaray (numpy). Chaque ligne du tableau fournit les tensions (en V) de la voie correspondante.

Les tensions non filtrées sont toujours accessibles avec la fonction `Sysam.entrees`.
La fonction `Sysam.paquet` décrite ci-dessus permet de récupérer les signaux filtrés avec les signaux non filtrés.
Une application typique de ces fonctions est le filtre anti-repliement numérique : on numérise avec un suréchantillonnage pour effectuer le filtrage numérique, puis on réduit la fréquence d'échantillonnage au moment de récupérer les tensions.

2.j. Acquisition en mode permanent (Sysam SP5)

Dans l'acquisition échantillonnée en mode permanent, la mémoire interne de la carte Sysam SP5 n'est pas utilisée. Les valeurs des tensions sont transmises au fur et à mesure de l'acquisition (avec un tampon interne à la carte de seulement 256 valeurs).
L'intérêt du mode permanent est de permettre l'acquisition d'un nombre de points supérieur à la capacité mémoire de la carte, par exemple plusieurs millions de points. En revanche, ce mode d'acquisition ne fonctionne pas si la fréquence d'échantillonnage est trop élevée (le maximum est entre 500 kHz et 1 MHz). Il faut faire des tests avec des signaux déterminés pour évaluer la limite exacte.
Le mode permanent permet aussi d'obtenir un flux continu d'échantillons, par exemple pour traiter un signal audio en temps réel.

```
Sysam.config_echantillon_permanent(techant,nbpoints)
```

Configuration de la période d'échantillonnage et du nombre de points à acquérir.

- ▷ `techant` : période d'échantillonnage en microsecondes
- ▷ `nbpoints` : nombre de points à acquérir (ou taille des paquets)

```
Sysam.acquerir_permanent()
```

Acquisition des tensions sur les entrées configurées, en mode permament. La fonction retourne lorsque l'acquisition est terminée.

```
Sysam.lancer_permanent(repetition=0)
```

Lancer l'acquisition en mode parallèle et mode permanent. La fonction retourne dès que l'acquisition est lancée.

▷ `repetition (integer)` : 1 pour que l'acquisition soit répétée dans fin.

Lorsque l'argument `repetition=1` est adopté, l'acquisition se fait en boucle et les paquets lus ont la taille définie par `config_echantillon_permanent`. Les paquets sont récupérés avec la fonction `paquet` en choisissant `premier=-1`. Les paquets contiennent les instants, les tensions non filtrées, et les tensions filtrées. Les paquets sont en fait stockés dans un tampon circulaire comportant 16 paquets. Lorsqu'aucun paquet n'est disponible, la fonction `paquet` renvoie un tableau vide. Une application de ce mode d'acquisition est le traitement d'un flux audio.

2.k. Ports logiques (Sysam SP5)

La centrale Sysam SP5 possède deux ports logiques de 8 bits chacun, accessibles par les connecteurs Sub-D.
Le port B fonctionne en TTL (tensions de 0 à 5 V). Il est constitué de deux paquets de 4 bits, chacun pouvant être programmé en entrée ou en sortie. Ce port bénéficie d'une protection contre les surtensions (max 12V). Le boitier BOLOGIC vendu par Eurosmart permet d'y accéder avec des douilles bananes. La fonction suivante permet de configurer le port B.

```
Sysam.portB_config(bits,etat)
```

Configuration d'un des paquets de 4 bits du port B en entrée ou en sortie.

▷ `bits (integer)` : 0 pour configurer les bits 0 à 4, 1 pour configurer les bits 4 à 7.

▷ `etat (integer)` : 0 pour configurer en entrée, 1 en sortie.

La fonction suivante permet d'écrire sur un bit configuré en écriture :

```
Sysam.portB_ecrire(bit,etat)
```

Écriture sur un bit du port B

▷ `bit (integer)` : numéro du bit (de 0 à 7).

▷ `etat (integer)` : 0 ou 1

La fonction suivante permet de lire sur un bit configuré en lecture :

```
b=Sysam.portB_lire(bit)
```

Lire un bit du port B

▷ `bit (integer)` : numéro du bit (de 0 à 7).

▷ `b (integer)` : valeur du bit (0 ou 1).

Remarque importante : les bits configurés en sortie se mettent automatiquement au niveau haut lorsqu'on ferme l'interface. Pour éviter cela, il suffit de configurer les bits en entrée avant de fermer l'interface.
Le port C fonctionne en CMOS (tensions de 0 à 3.3 V) et ne possède pas de protection contre les surtensions. Son utilisation est donc réservée à l'interfaçage avec des circuits CMOS. Chacun des 8 bits est programmable en entrée ou en sortie.
La fonction suivante configure le port C

```
Sysam.portC_config(bit,etat)
```

Configuration d'un des 8 bits du port C en entrée ou en sortie.

▷ `bit (integer)` : numéro du bit à configurer (0 à 7).

▷ `etat (integer)` : 0 pour configurer en entrée, 1 en sortie.

La fonction suivante permet d'écrire sur un bit configuré en écriture :

```
Sysam.portC_ecrire(bit,etat)
```

Écriture sur un bit du port C

▷ `bit (integer)` : numéro du bit (de 0 à 7).

▷ `etat (integer)` : 0 ou 1

La fonction suivante permet de lire sur un bit configuré en lecture :

```
b=Sysam.portC_lire(bit)
```

Lire un bit du port C

▷ `bit (integer)` : numéro du bit (de 0 à 7).

▷ `b (integer)` : valeur du bit (0 ou 1).

2.l. Compteur et chronomètre (Sysam SP5)

La centrale Sysam SP5 possède un compteur 48 bits avec une horloge de période 10 nanosecondes. L'entrée CHRONO peut être utilisée pour cela, ou bien une des entrées du port B accessibles par le connecteur PORT LOGIQUE. Ces entrées doivent recevoir un signal TTL (0/5 V).
L'utilisation en compteur consiste à compter le nombre de fronts sur une durée déterminée. Une application est la détermination de la fréquence d'un signal TTL.

```
Sysam.config_compteur(entree,front_montant,front_descend,hysteresis,du
```

Configuration du compteur

▷ `entree (integer)` : Numéro de l'entrée utilisée (0 à 7 pour le port B), ou `pycan.ENTREE_CHRO` (valeur 16) pour l'entrée CHRONO.

- ▷ `front_montant (integer)` : 1 si les fronts montants sont comptés, 0 sinon.
- ▷ `front_descend (integer)` : 1 si les fronts descendants sont comptés, 0 sinon.
- ▷ `hysteresis (float)` : durée de l'hystérésis en microsecondes. Après un front compté, le compteur reste inactif pendant cette durée.
- ▷ `duree (float)` : durée de fonctionnement du compteur, en microsecondes.

La période de base de l'horloge étant de 10 ns, les durées effectives sont multiples de 10 ns.

```
Sysam.compteur()
```

Déclenchement du compteur.

```
n=Sysam.lire_compteur()
```

Attente de la fin du comptage et renvoi du nombre de fronts comptés.

- ▷ `n (integer)` : nombre de fronts comptés sous forme un entier 64 bits `numpy.uint64`.

L'utilisation en chronomètre consiste à compter le nombre de tops d'horloge entre deux fronts du signal TTL.

```
Sysam.config_chrono(entree,front_montant,front_descend,hysteresis)
```

Configuration du chronomètre

- ▷ `entree (integer)` : Numéro de l'entrée utilisée (0 à 7 pour le port B), ou `pycan.ENTREE_CHRO` (valeur 16) pour l'entrée CHRONO.
- ▷ `front_debut (integer)` : front déclenchant le chronomètre (0 pour descendant, 1 pour montant).
- ▷ `front_fin (integer)` : front d'arrêt du chronomètre, (0 pour descendant, 1 pour montant).
- ▷ `hysteresis (float)` : durée de l'hystérésis en microsecondes. Après le front de démarrage, le détecteur de front reste inactif pendant cette durée.

```
Sysam.chrono()
```

Déclenchement du chronomètre.

```
n=Sysam.lire_chrono()
```

Attente de la fin du chronométrage et renvoi du nombre de tops d'horloge.

- ▷ `n (integer)` : nombre de tops d'horloges (10 ns) sous forme un entier 64 bits `numpy.uint64`.

Attention : l'entrée CHRONO ne fonctionne que sur front montant. Pour déclencher aussi sur front descendant, il faut utiliser une des entrées du port logique B.

2.m. Synthèse de signaux périodiques sur la sortie audio (non disponible dans la version 5)

Ces fonctions complémentaires permettent de générer deux signaux périodiques sur la sortie audio stéréo. La synthèse d'un signal se fait avec une table de 256 échantillons, par la méthode de l'accumulateur de phase. La fréquence d'échantillonnage par défaut est $44\ kHz$, mais il est possible de l'augmenter jusqu'à environ $300\ kHz$.
La méthode de synthèse directe par table (DDS) permet d'ajuster la fréquence avec une très grande précision. La précision en fréquence est :

$$\Delta f = \frac{f_e}{2^{32}}$$

Par exemple, pour une fréquence d'échantillonnage de $44\,\mathrm{kHz}$, la précision est de 10^{-5} Hz.
L'amplitude de la sortie audio lorsque le volume est au maximum est d'environ $1{,}6\,\mathrm{V}$.

```
audio_table_start(ch1,ch2,f1,f2,fs=44000,fpb=256)
```

Démarrage de la génération de deux signaux périodiques sur la sortie audio

- ▷ `ch1 (ndarray)` : table de 256 flottants compris entre -1.0 et 1.0 pour la voie 1
- ▷ `ch2 (ndarray)` : table de 256 flottants compris entre -1.0 et 1.0 pour la voie 2
- ▷ `f1 (float)` : fréquence pour la voie 1
- ▷ `f2 (float)` : fréquence pour la voie 2
- ▷ `fs (int)` : fréquence d'échantillonnage en Hz
- ▷ `fpb (int)` : nombre d'échantillons envoyés au contrôleur audio par trame

Les valeurs des tables supérieures à 1.0 ou inférieures à -1.0 sont écrêtées. La synchronisation des deux voies n'est garantie que si les deux fréquences sont identiques.

```
audio_table_stop(seconds)
```

Arrêt de la génération des signaux.

- ▷ `seconds` : délai en secondes avant l'arrêt

```
(table_1,table_2) = Sysam.audio_harmonics_start(a1,p1,a2,p2,f1,f2,nor
```

Génération de deux signaux périodiques définis par des harmoniques

- ▷ `a1 (float list)` : amplitudes des harmoniques pour la voie 1
- ▷ `p1 (float list)` : phases des harmoniques pour la voie 1
- ▷ `a2 (float list)` : amplitudes des harmoniques pour la voie 2
- ▷ `p2 (float list)` : phases des harmoniques pour la voie 2
- ▷ `f1 (float)` : fréquence pour la voie 1
- ▷ `f2 (float)` : fréquence pour la voie 2
- ▷ `norm (boolean)` : True pour normaliser les tables (valeur max ou min ramenée à 1 ou -1)

- ▷ `gain (float)` : coefficient multiplicateur appliqué aux tables (à utiliser avec norm=True)
- ▷ `fs (int)` : fréquence d'échantillonnage en Hz
- ▷ `fpb (int)` : nombre d'échantillons envoyés au contrôleur audio par trame
- ▷ `table_1 (ndarray)` : table de la voie 1
- ▷ `table_2 (ndarray)` : table de la voie 2

2.n. Sortie audio d'un signal (non disponible dans la version 5)

Les fonctions suivantes permettent de diriger un ou deux signaux numériques vers la sortie audio.
Les échantillons à envoyer sur une des deux voies de la sortie audio sont stockés dans un tampon circulaire. Les données sont écrites dans le tampon par blocs. La classe `RingBuffer` contient ce tampon circulaire.

```
buffer=RingBuffer(nblocks_power,samples_per_block)
```

Création d'un tampon.

- ▷ `nblocks_power (int)` : puissance de 2 du nombre de blocs, par exemple 6 pour 32 blocs.
- ▷ `samples_per_block (int)` : nombre d'échantillons par bloc.

Le tampon doit être détruit à la fin du script, afin de libérer l'espace mémoire associé :

```
RingBuffer.delete()
```

Destruction du tampon.
La fonction suivante permet d'écrire un bloc de données dans le tampon :

```
RingBuffer.write(data)
```

Écriture d'un bloc d'échantillons dans le tampon.

- ▷ `data (ndarray)` : tableau contenant le bloc d'échantillons, qui doit avoir la longueur `sample_per_block` défini au moment de la création du tampon.

La fonction suivante déclenche l'écriture d'un flux sur la sortie audio :

```
output_stream_start(sample_rate,ring_buffer_1,ring_buffer_2)
```

Configuration de la sortie audio.

- ▷ `sample_rate (int)` : fréquence d'échantillonnage en Hz.
- ▷ `ring_buffer_1 (RingBuffer)` : tampon circulaire pour la voie 1.
- ▷ `ring_buffer_2 (RingBuffer)` : tampon circulaire pour la voie 2 ou `None`.

Si l'on veut diriger le même flux de données sur les deux voies, il faut attribuer `None` à `ring_buffer_2`. Il ne faut pas attribuer le même tampon aux deux voies, car un tampon ne peut subir qu'une seule lecture.
La fonction suivante permet de stopper le flux audio :

```
output_stream_stop(seconds)
```

▷ `seconds (int)` : nombre de secondes avant l'arrêt.

Remarque importante : l'écriture des blocs dans le tampon doit se faire à la même fréquence que la sortie audio. Si N est la taille d'un bloc et f_e la fréquence d'échantillonnage, il faut faire cette écriture environ toutes les N/f_e secondes. Le tampon circulaire a néanmoins la faculté d'absorber des variations dans le rythme d'écriture, d'autant mieux que le nombre de blocs est élevé.