TP Instrument

October 16, 2024

1 Introduction

L'objectif de l'interfaçage d'un instrument est d'envoyer des instructions depuis l'ordinateur afin que celui-ci effectue des opérations normalement faites manuellement.

1.1 Liste du matériel

- Ordinateur avec installé : anaconda, VISA, pyvisa
- Oscilloscope (Rigol, DS1052E)
- Générateur (Rigol, DG1022)
- Décade de capacités, décades de résistances, bobine 9mH, cables bananes et adaptateurs BNC bananes, bouchons 50 Ohms.

1.2 Connexion

Il existe plusieurs type de connexion entre un ordinateur et un instrument. Citons le GPIB, le RS232, l'USB et l'ethernet.

Le GPIB et le RS232 sont des interfaces de communications qui sont obsolètes, mais se trouvent encore fréquemment sur des appareils. Ce sont des interfaces simples, puisqu'on échange des mots d'octets entre les deux appareils. Le RS232 est le port série (COM sous windows).

L'USB est beaucoup plus complexe car il ne définit par précisément ce qui peut être échangé. Pour l'USB, il est nécessaire d'avoir un driver pour chaque mode de communications. Comme mode de communication, citons l'émulation d'un port série (c'est par exemple ce que l'on retrouve sur les arduinos). Pour les instruments de mesure, une norme appelée USBTMC est utilisée par plusieurs fabricants (USB Test and Measurement Class). Certains fabricants peuvent cependant avoir développé leur propre interface USB.

Enfin, de plus en plus de d'instruments sont connectés par ethernet. Les choses sont en général beaucoup plus simples que l'USB car il n'y a pas besoin de drivers spécifiques. Des normes de communications existent pour les instrument, en particulier le VXI-11.

Le VXI-11 et l'USBTMC ont pour but de remplacer le GPIB. Ces protocoles permettent d'échanger des instructions selon une norme précise, le SCPI.

1.3 SCPI

Le SCPI (Standard Commands for Programmable Instruments) est en quelque sorte la grammaire que l'on va utiliser. Par exemple, pour changer l'échelle verticale d'un oscilloscope on peut envoyer

le chaîne : “CHANNEL1:SCALE 10”. Pour connaître l'échelle verticale on va envoyer “CHANNEL1:SCALE?”. L'instrument va alors renvoyer une chaîne correspondant à l'échelle. Dans cet exemple, les mots précis dépendent de la marque de l'instrument. Ce que le SCPI défini est la syntaxe et quelques instructions de base.

Les instructions sont regroupées sous forme d'un arbre. Chaque “branche” est séparée par un “:”. Les arguments sont séparés de l'instruction par une espace et sont séparés entre eux par des virgules. Il est aussi possible de transférer des données binaires. Lorsque l'instruction attend une réponse, alors elle est suivie d'un “?”.

Pour connaître l'ensemble des commandes il faut se référer à la partie de la documentation de l'instrument appelée souvent Programmers' User Guide.

Les instructions de base d'un instrument sont entre autre :

- `*IDN?` qui renvoie une chaîne d'identification unique de l'appareil. Cette chaîne contient : `<manufacturer>,<model number>,<serial number>,<software revision>`
- `*RST` qui fait un reset de l'appareil.

Un dernier point important : le SCPI n'est pas sensible à la casse. De plus chaque instruction possède une version courte et une version longue. Celles-ci sont en général distinguées par la casse. Par exemple “CHANnel1:SCALe” signifie que l'on peut utiliser indépendamment CHAN ou CHANNEL et SCAL ou SCALE.

1.4 VISA

Sous Windows, il est possible d'utiliser un driver VISA qui fournit une interface commune, quelque soit le mode de connexion ou le protocole bas niveau de communication avec l'instrument. La librairie VISA est une librairie propriétaire gratuite fournie par National Instruments. Elle peut être utilisée sous Python avec le module pyvisa (qu'il faut installer en plus de VISA).

Le logiciel NI-MAX est installé en même temps que VISA. Il permet entre autre de lister l'ensemble des appareils connecté à l'ordinateur et d'envoyer des instructions directement à l'instrument.

2 Connexion

1. Préparation : Manuellement, générez un signal sinusoïdal de fréquence 20 kHz, d'amplitude 5 Vpp et d'offset 0 V sur le channel 1 de votre générateur de fonction arbitraire. Mesurez votre signal à l'aide de l'oscilloscope. Vous utiliserez la fonction “trig” sur un front descendant.
2. Identifiez votre générateur de fréquence arbitraire, en utilisant les commandes de pyvisa :

```
import pyvisa
rm = pyvisa.ResourceManager()
rm.list_resources()
```

Les commandes suivantes vous permettront de créer une connection avec votre instrument et de la tester

```
instrument = rm.open_resource(chaine_d_identification, query_delay=0.001)
instrument.query("*IDN?")
```

L'argument optionel `query_delay` est indispensable pour le GBF.

3. Créez deux variables (`scope` et `gbf`) correspondant à chacun de vos deux instruments
4. Ecrivez une fonction, que l'on testera sur l'oscilloscope, qui renvoie un dictionnaire avec le nom de la compagnie, le nom du modèle, et le numéro de série.

3 Interfaçage du générateur de fonction arbitraire

1. Regardez dans la documentation l'instruction `APPLy`. Programmez une sinusoïde de fréquence 3 kHz, d'amplitude 2 Vpp et avec un offset de 0.5 V.

Il existe des commandes simples permettant de changer un paramètre unique, par exemple :

```
gbf.write('FREQ 1542')
gbf.query('FREQ?')
```

Ces fonctions utilisent des chaînes de charactères.

2. Écrivez deux fonctions python `set_frequency` et `get_frequency` qui utilisent des nombres.
3. Testez votre fonction `set_frequency` pour régler la fréquence du signal à 1.123 MHz. Qu'observez vous à l'oscilloscope ? Pourquoi ?

Nous rappelons la définition de la puissance en unité dBm:

$P(dBm) = 10\log(P(mW)) = 20\log(\frac{V_{rms}}{\sqrt{0.001R}}))$

où par convention P est la puissance électrique dissipée dans une résistance de $R = 50$ Ohm. Nous rappelons aussi que $V_{pp} = 2\sqrt{2}V_{rms}$

4. À la main, avec la bouton utility réglez le GBF pour qu'il ait une impédance de 50 Ohm. Régler l'amplitude du signal sur 0 dBm. Quelle est l'amplitude Vpp du signal que vous mesurez sur l'oscilloscope ? Est-ce cohérent ? Pourquoi ?
5. Ecrivez une fonction python `set_amplitude` qui permet de changer l'amplitude du signal généré. Elle acceptera aussi l'unité Vpp, Vrms, dBm (l'unité par défaut sera Vpp). Il faudra d'abord utiliser une commande pour changer l'unité et ensuite utiliser une commande pour ecrire la valeur de l'amplitude. Entre les deux commandes, on mettra une pause de 100 ms (`from time import sleep`)
6. Ecrivez une fonction qui retourne un dictionnaire avec la forme, la fréquence, l'amplitude et l'offset du signal actuellement généré sur le channel 1. On utilisera la commande `APPL?` .
7. *(si vous avez le temps)* Ecrivez une fonction qui permet de faire un saut de fréquence de δf. Cette fonction renvera une exception si la fréquence finale n'est pas comprise entre 1 Hz et 5 MHz.

4 Interfaçage de l'oscilloscope

Comme pour le GBF, il y a un certain nombre de commandes que vous pouvez trouver dans la documentation : DS1000DE_ProgrammingGuide_EN.pdf

Par exemple la commande `:TIMEBASE:SCALE` permet de connaître ou de programmer l'échelle horizontale.

ATTENTION : les commandes de cet oscilloscope commencent par un `:` .

1. Essayez cette commande.

La difficulté particulière pour l'oscilloscope est de récupérer la courbe affichée sur celui-ci. La commande `:WAVEFORM:DATA?` va renvoyer les donner sous formes binaires. Qu'est-ce que cela signifie ?

L'échange de donnée entre l'oscilloscope et l'ordinateur se fait par l'échange d'une chaîne d'octet. Lorsque l'on veut envoyer un mot (par exemple `*IDN?`), celui ci va être encodé avec la table ASCII (en.wikipedia.org/wiki/ASCII). Dans ce cas la suite d'octet sera (en décimale) `42, 73, 68, 78, 63`. De la même façon un nombre écrit en décimal peut être encodé en ASCII : par exemple `12345` donnera `49, 50, 51, 52, 53`. Cependant, un nombre peut être codé plus efficacement : en effet on peut écrire $12345 = 48 \times 256 + 57$, et décider d'envoyer les deux octets `48` et`57`. C'est le codage binaire, codage utilisé par notre oscilloscope. La différence est la même que celle qu'il y a entre `np.savetxt` et `np.save`.

La librairie `pyvisa` est capable de décoder automatiquement les données binaires envoyées par l'oscilloscope. Dans notre cas, les ordonnées des points sont des entiers entre 0 et 256 et sont donc codées sur un seul octet. Cela correspond au type de donnée 's' (short). On pourra récupérer les données à l'aide de la commande

```
scope.query_binary_values(':WAVEFORM:DATA?', datatype='s')
```

2. Utilisez cette commande et affichez la courbe avec matplotlib.

On remarque que les valeurs sont à l'envers (le haut correspond au bas) et qu'il faut les mettre à l'échelle. Cette opération nécéssite de connaître plusieurs paramètres de l'oscilloscope et prend du temps à coder. Nous ne le ferons pas.

5 Librairie tpmontrouge

Une librairie a été écrite afin de faciliter l'usage des instruments de la plateforme expérimentale de montrouge (particulièrement la préparation à l'agrégation). L'idée est d'avoir une interface pour chaque type d'instrument indépendamment de la marque et du modèle.

La création des objets se fait de la façon suivante.

[40]:
```
from tpmontrouge.instrument import Scope,GBF, get_first_instrument

scope = get_first_instrument(Scope) # renvoie le premier instrument de type␣
↪Scope qu'il trouve
gbf = get_first_instrument(GBF)
```

Si il y a une erreur avec le GBF, ou si il n'est pas trouvé automatiquement, utilisez la commande suivante :

[ ]:
```
from tpmontrouge.instrument.gbf.rigol.generic import Rigol
```

```
gbf = Rigol(rm.open_resource(chaine_d_identification, query_delay=0.001))
```

Le changement d'une valeur peut se faire de deux façon différences : soit à l'aide de méthodes soit d'une `property`. Par exemple :

```
gbf.set_frequency(10E3)
print(gbf.get_frequency())

gbf.frequency = 10E3
print(gbf.frequency)
```

Pour le GBF, nous avons défini les propriétés suivante : `amplitude`, `frequency`, `offset`, `function`

Pour l'oscilloscope, nous avons plusieurs méthodes `autoset()`, `start_acquisition()`, `stop_acquisition()` et beaucoup de propriétés. Voici un exemple :

[99]:
```
print('Scale', scope.horizontal.scale, 's/div')
sleep(0.01)
print('Offset', scope.horizontal.offset, 's')
sleep(0.01)
scope.horizontal.scale = 1.0e-4 # s/div
sleep(0.01)
scope.horizontal.offset = 0.0E-3 # s
sleep(0.01)

scope.channel[1].coupling = 'AC'
sleep(0.01)
scope.channel[1].scale = 1.0
sleep(1)
wfm = scope.channel[1].get_waveform()
```

```
Scale 0.0001 s/div
Offset 0.0 s
```

Cette dernière commande renvoie un objet qui contient toute les données de la trace dont `wfm.x_data` et `wfm.y_data`.

3. Utilisez la librairie tpmontrouge pour programmer une sinusoïde de fréquence 150 kHz, d'amplitude 2 Vpp et avec un offset de 0.5 V. Initialisez ensuite l'oscilloscope et récupérez une trace

6 Ajustement (fit) d'une sinusoïde avec un modèle

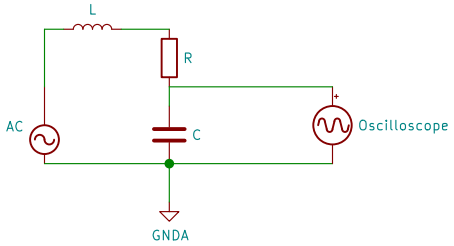
L'idée ici est de récupérer l'amplitude et la phase d'une sinusoïde par une fit. Au lieu de fitter par quelque chose comme $a\sin(\omega t + \phi)$ nous allons fitter par $a\sin(\omega t) + b\cos(\omega t)$ et calculer ensuite l'amplitude et la phase comme étant la norme et l'argument du nombre complexe $a + ib$. On rajoutera un offset.

1. Définissez une fonction de fit. Ajuster la sinusoïde que vous avez récupérée avec l'oscilloscope. Affichez l'amplitude et la phase de la sinusoïde. On créera une fonction `fit_sinusoid(wfm,`

`freq)` qui renverra le nombre $a + ib$.

7 Diagramme de Bode

Nous avons maintenant tous les outils pour faire un diagramme de Bode. Nous proposons de monter le circuit suivant:



On prendra typiquement $C = 500$ pF et $R = 100\ \Omega$

L'idée consiste à mesurer l'amplitude complexe des deux sinusoïdes (entrée et sortie) à l'aide de deux ajustements et de calculer ensuite la réponse complese pour en déduire le gain et le déphasage.

1. Faire un nouveau notebook qui contient tout ce qui est nécéssaire pour prendre les points du diagramme de Bode. On enregistrera dans un fichier les points de mesures de la réponse complexe du circuit. Avec un autre notebook, lire et afficher les données.

   On pourra utliser la fonction autoset à chaque nouvelle fréquence (et attendre quelques secondes avant d'enregistrer les données).