automatisation des tests et des mesures

programmation de l'équipement d'essai pour qu'il fasse ce que vous voulez

Stuart Cording (Elektor)

Les essais peuvent être monotones et répétitifs, et la recherche de défauts est particulièrement difficile. Alors pourquoi ne pas utiliser les interfaces de communication des équipements de test et de mesure pour faciliter les choses ? Nous explorons ici des outils qui peuvent facilement être contrôlés à l'aide de Python, sans avoir à investir dans des licences ou des logiciels coûteux.

Figure 1. Deux des alimentations CC de la série EL-R sont dotées d'une interface série (RS-232/COM virtuel via USB) pour faciliter l'automatisation des tests. (Source : Aim-TTi)

Pour la plupart des développeurs, les équipements de test et de mesure sont posés sur l'établi, attendant d'effectuer une tâche. Mais si vous retournez la plupart de ces appareils, vous trouverez souvent une interface de communication à l'arrière. Associée au logiciel approprié, elle permet de contrôler les mesures et de rassembler les résultats en vue d'une analyse ultérieure. Cela peut s'avérer très utile pour rechercher des événements et des défaillances sporadiques ou pour tester une application en fonction de divers paramètres du système. C'est également de cette manière que les tests de fin de ligne de production et le regroupement des composants peuvent être automatisés.

Contrôle à distance des blocs d'alimentation des laboratoires

Un bon point de départ est l'alimentation électrique. La plupart des applications nécessitent un cycle d'alimentation pour effectuer une réinitialisation. Dans le cadre de tests plus avancés, l'application cible sera exploitée aux extrêmes de la spécification de l'alimentation d'entrée autorisée, et il peut être nécessaire d'effectuer des tests de surtension. Cela est courant dans l'automobile, où des impulsions allant jusqu'à 87 V doivent être supportées pendant 400 ms (ISO 7637-2). Un autre mode de défaillance courant lié aux conditions d'alimentation est une tension qui augmente ou diminue lentement. Dans de telles conditions, les circuits tombent souvent dans une condition de *brown-out* dont ils ne peuvent pas se remettre. Enfin, pour les systèmes de test automatisés, il est utile de disposer d'un équipement qui peut être correctement configuré à la fois pour la tension et le courant maximal, en particulier lorsque la même installation est utilisée pour plusieurs produits différents.

Les prix des alimentations de laboratoire ont considérablement baissé ces dernières années. Pour compenser l'afflux d'équipements bon marché, les marques plus connues élargissent généralement leur gamme de caractéristiques, car elles ne veulent ou ne peuvent pas rivaliser uniquement sur le plan des prix. On ne sait pas si c'est le cas d'Aim and Thurlby Thandar Instruments (Aim-TTi) à Cambridge, au Royaume-Uni, mais l'entreprise tient sa promesse d'offrir un « meilleur rapport qualité-prix ». Ils proposent une gamme étendue d'alimentations CC, mais leur série EL-R d'entrée de gamme (**figure 1**) vaut la peine d'être examinée de plus près, surtout si l'on caresse l'idée d'une première installation de test automatisée.

Ces alimentations utilisent une régulation linéaire à faible bruit pour fournir des sorties simples, doubles et triples. Elles sont dépourvues de ventilateur, s'appuyant sur un refroidissement par convection, et leur puissance varie de 30 W à 130 W. Dotés d'un ou deux écrans LED rouges et de commandes analogiques, certains modèles comprennent également des bornes de détection à distance. Deux modèles intéressants sont le EL302P à sortie unique, avec RS-232, et le EL302P-USB, avec USB, tous deux des unités de 60 W avec des sorties de 0 V à 30 V et 0 A à 2 A.

Les unités sont livrées avec des pilotes logiciels pour l'interface. En outre, l'utilitaire « PSU Sequencer » est fourni sur le site web du fournisseur [1], permettant aux paramètres de tension/courant configurés par l'utilisateur d'être échelonnés manuellement ou automatiquement (**figure 2**). En outre, des séquences préparées à l'avance peuvent être importées à partir d'une feuille de calcul.

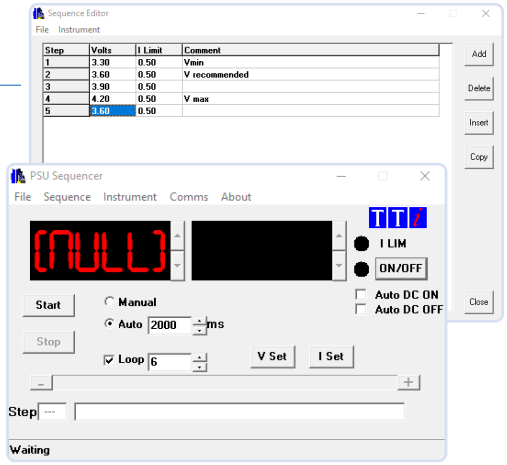

Figure 2. Le logiciel PSU Sequencer d'Aim-TTi permet d'effectuer des cycles simples et répétitifs de réglages de tension et de courant. (Source : Aim-TTi)

Exemple 1 : script Python simple pour acquérir la chaîne d'identification de l'unité PSU à l'aide de pySerial

```
import serial
ser = serial.Serial('/dev/ttyUSB0', 9600, timeout=0)
                                    # open serial port
print(ser.name)                      # confirm port was really used
ser.write(b'*IDN?')                  # request PSU's ID strings
response = ser.readline()            # collect response
print(response)                      # output PSU ID string
ser.close()                          # close port
```

Python pour le contrôle de la puissance

Le développement de votre propre logiciel de contrôle, adapté à vos besoins de test, est également simple. L'interface RS-232 matérielle fonctionne entre 600 et 9 600 bauds, et l'interface USB apparaît comme un port série virtuel. Les commandes sont toutes répertoriées dans le manuel d'instructions [2] et, avec un peu d'organisation, il est assez simple de développer une bibliothèque qui transforme toutes les commandes en fonctions ou méthodes clairement nommées. Une option consiste à utiliser un Arduino et un émetteur-récepteur RS-232 pour contrôler l'alimentation électrique. Cette solution présente l'avantage de permettre l'intégration d'autres fonctions de test, telles que la commutation de signaux à l'aide de relais ou la capture de signaux analogiques et numériques.

Alternativement, Python fournit le module pySerial [3]. Avec un script simple (**exemple 1**), l'interface de commande peut être implémentée comme un module Python, et le contrôle automatisé peut être mis en œuvre. En lisant la documentation, il est également possible d'implémenter un pont TCP/IP vers série [4]. Cette fonction est définie dans le mémo expérimental RFC2217 [5] et permet à un PC distant de configurer une interface série et de mettre en œuvre la communication.

Détermination de la portée des signaux

Les oscilloscopes peuvent également être contrôlés à distance. Grâce à leur large éventail de capacités, allant de la capture de signaux analogiques et numériques aux FFT, ils peuvent être utilisés pour divers tests automatisés. Par exemple, certaines défaillances sont difficiles à détecter en raison d'une série complexe d'événements qui doivent se produire, dans une séquence spécifique, pour les déclencher. Une fois que vous avez déterminé comment déclencher la défaillance, l'étape suivante consiste à configurer l'oscilloscope pour capturer les signaux pertinents qui aideront à déterminer la cause ultime.

Les cartes telles qu'Arduino et Raspberry Pi sont idéales pour le développement rapide d'un ensemble de tests qui peut déclencher la défaillance de manière répétée, en utilisant des sorties analogiques et des signaux numériques, parfois complétés par un relais ou un MOSFET si nécessaire. Ils peuvent également fournir des signaux de déclenchement synchronisés avec précision pour un oscilloscope afin de s'assurer que la bonne partie des signaux pertinents est sauvegardée pour l'analyse.

De nombreux oscilloscopes proposent des interfaces USB et LAN, mais certains n'offrent qu'un support pour un logiciel propriétaire ou un accès à un serveur web pour la configuration. Cependant, il existe une spécification de classe USB pour le test et la mesure, connue sous le nom d'USBTMC [6]. À l'instar des dispositifs de stockage, tels que les clés USB, et des dispositifs d'interface homme-machine (IHM), tels que votre souris et votre clavier, cette classe fournit des commandes prédéfinies adaptées à l'interface avec les équipements de test et de mesure.

Des appareils tels que le B&K Precision 2567B [7], un oscilloscope à signaux mixtes (MSO) de 200 MHz et 2-GSa/s, prennent en charge cette interface. Avec quatre voies analogiques, un port numérique à 16 voies, un générateur de formes d'ondes arbitraires intégré de 50 MHz et des déclencheurs avancés, il est simple à configurer à l'aide de son grand écran tactile capacitif de 10,1 pouces (**figure 3**). Mais c'est tout aussi facile via l'USB.

Configuration à distance

Là encore, Python est le langage de programmation préféré grâce au projet USBTMC [8] d'Alex Forencich, qui est hébergé sur GitHub [9].

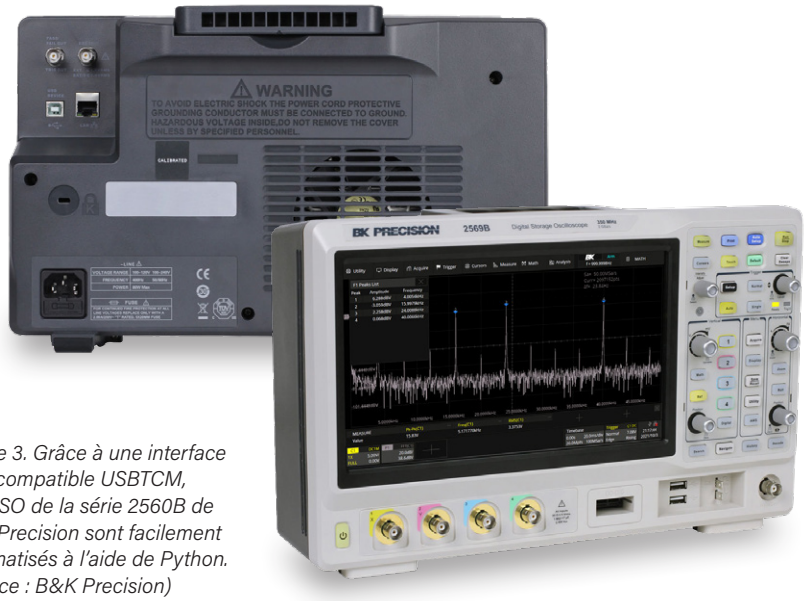

Figure 3. Grâce à une interface USB compatible USBTCM, les MSO de la série 2560B de B&K Precision sont facilement automatisés à l'aide de Python. (Source : B&K Precision)

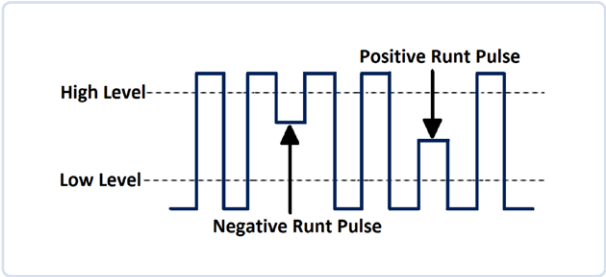

*Figure 4. Le **runt trigger** est un mécanisme astucieux qui permet de détecter les impulsions qui ne montent ou ne descendent pas complètement. (Source : B&K Precision)*

Figure 5. Le Moku:Lab, de la taille d'une petite assiette, est constitué de 12 instruments en un seul, et peut être configuré à l'aide d'une API Python. (Source : Liquid Instruments)

Le module fonctionne sous Linux, mais peut nécessiter de modifier les permissions de manière appropriée. Sous Windows, *PyUSB* [10] et *Libusb* [11] doivent être installés au préalable.

Les premières étapes sont relativement simples. Avant de commencer, il faut connaître les identités du vendeur (VID) et du produit (PID), deux valeurs de référence spécifiques à l'USB qui permettent d'identifier un produit USB. Sous Linux, il est facile d'obtenir ces informations à partir de la ligne de commande en utilisant `lsusb` une fois que le périphérique est connecté. La liste qui en résulte fournit les détails nécessaires. Sous Windows, ces valeurs peuvent être déterminées via le gestionnaire de périphériques et les propriétés du périphérique. Dans les deux cas, les valeurs fournies sont des valeurs hexadécimales de 16 bits.

Il ne reste plus qu'à importer le module `usbtmc` dans votre code Python et à utiliser l'interface de programmation disponible (API). Cela revient à imprimer du texte à l'écran et à évaluer les données saisies au clavier. L'USBTMC n'est en fait qu'une enveloppe permettant de communiquer avec des équipements de test et de mesure performants. Les commandes de contrôle sont des chaînes ASCII avec diverses options, comme décrit dans le manuel de programmation de l'oscilloscope [12]. Si elles sont disponibles, les chaînes de ressources VISA [13] peuvent également être utilisées.

Le VID et le PID sont propres au produit, et non à l'unité individuelle. Ainsi, si deux ou plusieurs appareils identiques sont connectés, ils peuvent être adressés indépendamment à l'aide d'un troisième paramètre, leur numéro de série. Celui-ci figure généralement sur une étiquette ou peut être obtenu à l'aide de la commande `*IDN?` (**exemple 2**).

De nos jours, il n'y a pas grand-chose qui différencie les oscilloscopes au-delà de la bande passante, de la fréquence d'échantillonnage et de la quantité de mémoire. Mais de temps en temps, une petite caractéristique intéressante apparaît. La série B&K dispose d'une fonction déclenchement *runt* (« avorton ») (**figure 4**) qui se déclenche lorsqu'un signal franchit un seuil (par exemple, un niveau élevé) mais pas l'autre (par exemple, un niveau bas). En matière d'analyse des défaillances, tout est bon à prendre.

Équipement de test sans tête

Red Pitaya a innové dans le domaine des équipements de test, en remettant en question la nécessité pour un outil d'avoir son propre écran. Grâce à un puissant FPGA et à une interface Ethernet, votre PC ou votre ordinateur portable constitue l'interface utilisateur.

Liquid Instruments a adopté la même approche avec son Moku:Lab [14], un laboratoire de la taille et de la forme d'une assiette. Il offre une gamme d'entrées et de sorties analogiques et est conçu comme un laboratoire dans une boîte.

Quatre connecteurs BNC sont encastrés dans la face avant (**figure 5**). La paire de droite fournit les sorties analogiques, supportant un taux d'échantillonnage de 1 GSa/s par canal avec une résolution de 16 bits et une bande passante (-3 dB) de >300 MHz. La paire de gauche correspond aux entrées analogiques, avec une bande passante (-3 dB) de 200 MHz en 50 Ω et un taux d'échantillonnage de 500 MSa/s par voie à une résolution de 12 bits. La base de temps interne offre une précision meilleure que 500 ppb. Une entrée de déclenchement est également fournie, ainsi que des connecteurs permettant la synchronisation de plusieurs unités. La connectivité filaire est assurée par un port Ethernet et une interface USB, et un second port d'alimentation USB est disponible pour charger une tablette. Enfin, il y a un emplacement pour carte SD et une entrée d'alimentation en courant continu.

Bien que l'appareil dispose d'interfaces filaires, il est réellement conçu pour être utilisé en conjonction avec un iPad en Wi-Fi (802.11 b/g/n) et l'application correspondante. Douze instruments peuvent être sélectionnés via l'interface utilisateur, allant du familier, comme l'oscilloscope et l'analyseur de spectre, à l'éclectique, comme le contrôleur PID et la boîte de verrouillage laser. L'outil fonctionne également comme un enregistreur de données. La taille de votre carte SD est la limite pour le stockage des données jusqu'à une acquisition de 100 kSa/s, tandis que la capacité de la mémoire interne la définit pour des taux allant jusqu'à 1 MSa/s.

Bien entendu, comme tous les outils modernes, le Moku:Lab propose une API Python et un support MATLAB et LabVIEW.

Exemple 2 : utilisation de USBTMC dans Python pour contrôler un oscilloscope B&K Precision

```
import usbtmc
instr = usbtmc.Instrument(<VID>, <PID>, 'ABC123456')
# or, without serial number
instr = usbtmc.Instrument(<VID>, <PID>)
print(instr.ask("*IDN?"))
# returns 'BK Precision,2567B-MSO,ABC123456,5.0.1.3.9R3'
```

Appuyé par de nombreux exemples (**example 3**), le dispositif peut aussi être rapidement intégré dans un harnais de test automatisé. Il serait bien adapté aux tests de masse des dispositifs RF et au tri de composants.

Gagner du temps, améliorer la précision

Les équipements d'essai sont les yeux de l'ingénieur, car ils montrent ce qui se passe dans les systèmes complexes. Mais ils ont leurs limites. Par définition, les événements sporadiques sont difficiles à repérer, et une solution ne peut être proposée que pour une cause. Les interfaces de programmation supportées par les API sur les équipements de test et de mesure sont plus courantes aujourd'hui, allant du plus simple au plus raffiné. Et, de plus en plus, c'est le langage Python, facile à apprendre, qui devient le langage de programmation de prédilection. Si votre problème de mesure devient difficile à déclencher, nécessite des changements consécutifs pour compléter une seule série de tests, ou est simplement ennuyeux et répétitif (conduisant à des erreurs de l'opérateur), l'automatisation n'est pas aussi complexe que vous l'avez peut-être pensé.

VF : Maxime Valens — 230046-04

Des questions, des commentaires ?

Envoyez un courriel à l'auteur (stuart.cording@elektor.com) ou contactez Elektor (redaction@elektor.fr).

Exemple 3 : création d'une forme d'onde arbitraire sur le Moku:Lab en Python

```
"""pymoku example: Arbitrary waveform generator
 (c) 2019 Liquid Instruments Pty. Ltd.
 (shortened version for Elektor)
"""
 from pymoku import Moku
 from pymoku.instruments import ArbitraryWaveGen
 import numpy as np
# generate a signal the the Arb Waveform Gen should generate on the output
 t = np.linspace(0, 1, 100)  # Evaluate our waveform at 100 points
# Simple square wave (can also use scipy.signal)
 sq_wave = np.array([-1.0 if x < 0.5 else 1.0 for x in t])
# More interesting waveform. Note that we have to normalize this waveform
 # to the range [-1, 1]
 not_sq = np.zeros(len(t))
 for h in np.arange(1, 15, 2):
     not_sq += (4 / (np.pi * h)) * np.cos(2 * np.pi * h * t)
not_sq = not_sq / max(not_sq)
# Connect to your Moku by its device name
 m = Moku.get_by_name('Moku')
# Prepare the ArbitraryWaveGen instrument
 i = m.deploy_or_connect(ArbitraryWaveGen)
try:
     # Load the waveforms to the device.
     i.write_lut(1, not_sq)
     i.write_lut(2, sq_wave)
    # Configure on-device linear interpolation
     i.gen_waveform(1, period=1e-6, amplitude=1, interpolation=True)
     i.gen_waveform(2, period=1e-6, amplitude=2, interpolation=False)
 finally:
     m.close()
```

LIENS

[1] PSU Sequencer, Aim-TTi: http://bit.ly/40jGiCQ
[2] EL-R Series DC Power Supplies, Aim-TTi: http://bit.ly/3kWZmXk
[3] Module pySerial: http://bit.ly/3YgSJgZ
[4] Exemple de pont TCP/IP pySerial: http://bit.ly/3kTDKem
[5] G. Clark, « RFC 2217 Telnet Com Port Control Option », Cisco Systems, Inc, octobre 1997.: http://bit.ly/3jcQGM4
[6] « Universal Serial Bus Test and Measurement Class Specification (USBTMC) », USB Implementers Forum, Inc. avril 2003: http://bit.ly/3YbJY7L
[7] Modèle 2567B-MSO, B&K Precision: http://bit.ly/3kN3rNy
[8] A. Forencich,« Python USBTMC », juillet 2014: http://bit.ly/3HMLeZN
[9] A. Forencich, python-usbtmc, Projet GitHub: http://bit.ly/3wK4i4r
[10] Logiciel PyUSB: http://bit.ly/3WRFW3I
[11] Logiciel libusb: http://bit.ly/3wLEvIY
[12] « Programming Manual 2560B Series », B&K Precision, septembre 2022: http://bit.ly/3WRDYjs
[13] « VISA Resource Syntax and Examples », National Instruments Corp., mai 2022: http://bit.ly/3XK3okh
[14] Moku:Lab, Liquid Instruments: http://bit.ly/3WT6bXb