Automatisering van testen en meten

testapparatuur programmeren om je te dienen

Stuart Cording (Elektor)

Testen kan eentonig en steeds weer hetzelfde zijn, en het opsporen van fouten is een buitengewone uitdaging. Waarom dan geen gebruik maken van de communicatie-interfaces op test- en meetapparatuur om het gemakkelijker te maken? We verkennen diverse apparaten die op eenvoudige wijze met Python bestuurd kunnen worden, zonder de noodzaak van dure software of licenties.

Figuur 1. Twee modellen uit de EL-R serie gelijkspanningsvoedingen bieden een seriële interface (RS-232/virtuele COM via USB) ter ondersteuning van testautomatisering (bron: Aim-TTi).

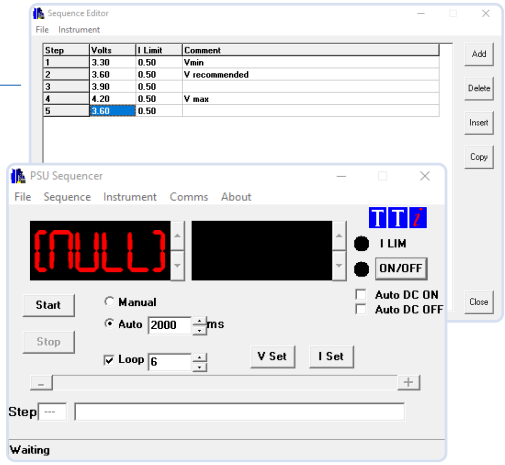
Voor de meeste ontwikkelaars staat test- en meetapparatuur op de werkbank te wachten om een taak uit te voeren. Maar draai de meeste van die apparaten eens om, en je vindt vaak een communicatie-interface aan de achterkant. In combinatie met de juiste software krijg je de mogelijkheid om metingen te besturen en resultaten te verzamelen voor latere analyse. Dit kan bijzonder nuttig zijn bij het zoeken naar sporadische gebeurtenissen en storingen of het testen van een toepassing met verschillende systeemparameters. Dit is ook de manier waarop het testen aan het einde van de productielijn en het sorteren van componenten aan de hand van prestaties kan worden geautomatiseerd.

Afstandsbediening van labvoedingen

Een goed uitgangspunt is de voeding. Voor de meeste toepassingen is het nodig om de voeding te herstarten om een harde reset uit te voeren. Bij meer geavanceerde tests zal de applicatie in kwestie tot aan de uiterste grenzen van de toegestane ingangsspecificatie werken, en kan het nodig zijn om overspanningstests uit te voeren. Dit is gebruikelijk in de auto-industrie, waar pulsen tot 87 V moeten worden verdragen gedurende 400 ms (ISO 7637-2). Een andere veel voorkomende storing in samenhang met de voedingsspanning is een langzaam stijgende of dalende spanning. Onder dergelijke omstandigheden komen schakelingen vaak in een brown-out toestand terecht waarvan ze zich niet meer kunnen herstellen. Ten slotte helpt het voor geautomatiseerde testsystemen om apparatuur te hebben die correct kan worden ingesteld voor zowel spanning als maximale stroom, vooral wanneer dezelfde opstelling wordt gebruikt voor verschillende producten.

Laboratoriumvoedingen zijn de laatste jaren sterk in prijs gedaald. Om de toestroom van goedkope apparatuur te compenseren, breiden de bekendere merken meestal het scala aan mogelijkheden uit, en zijn niet bereid of in staat om alleen op prijs te concurreren. Het is onbekend of dit het geval is bij Aim and Thurlby Thandar Instruments (Aim-TTi) in Cambridge (Groot-Brittannië), maar ze maken hun belofte van 'meetbae meerwaarde' wel waar. Ze bieden een uitgebreide serie gelijkstroomvoedingen, en hun instapmodel uit de EL-R serie (**figuur 1**) is een nadere kennismaking waard, vooral als je speelt met het idee van je eerste geautomatiseerde testopstelling. Deze voedingen gebruiken een ruisarme lineaire regeling voor enkele, dubbele en drievoudige uitgangen. Ze hebben geen ventilator maar vertrouwen op convectiekoeling, en variëren in vermogen van 30 tot 130 W. Met één of twee rode LED-displays en analoge regeling; sommige modellen zijn ook uitgerust met remote sense-terminals. Twee interessante modellen zijn de EL302P met één uitgang, met RS-232, en de EL302P-USB met USB, beide 60 W bij 0...30 V en 0...2 A.

De apparaten worden geleverd met softwaredrivers voor de interface. Daarnaast is op de



Figuur 2. De PSU Sequencer-software van Aim-TTi biedt eenvoudige, repeterende cycli van spannings- en stroominstellingen (bron: Aim-TTi).

Listing 1. Eenvoudig Python-script om een PSU ID-string te verkrijgen met 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
```

website van de leverancier het hulpprogramma 'PSU Sequencer' beschikbaar [1], waarmee door de gebruiker geconfigureerde spanning/stroom-instellingen handmatig of automatisch kunnen worden doorlopen (**figuur 2**). Bovendien kunnen vooraf voorbereide meetreeksen uit een spreadsheet worden geïmporteerd.

Python voor voedingsregeling

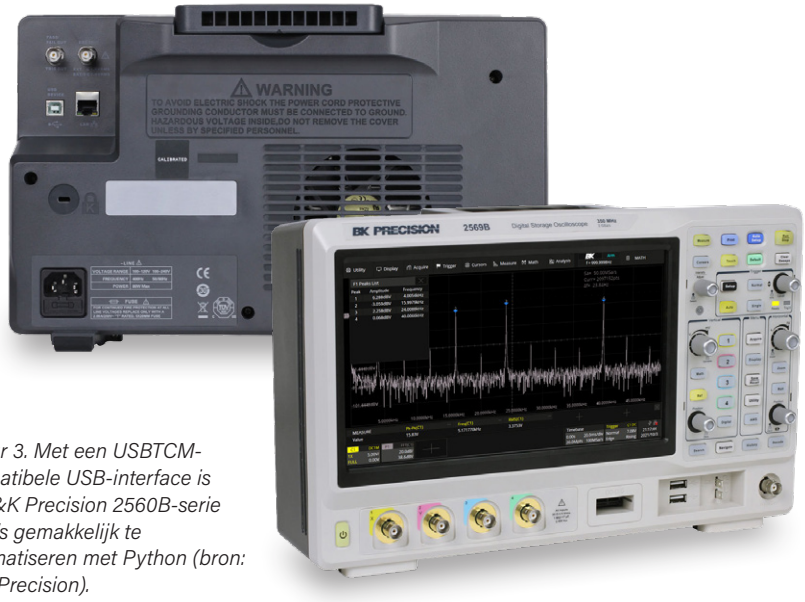
Het ontwikkelen van je eigen besturingssoftware, afgestemd op jouw testbehoeften, is ook eenvoudig. De hardware RS-232 interface werkt van 600 tot 9.600 baud, en de USB-interface verschijnt als een virtuele COM-poort. De commando's staan allemaal in de handleiding [2], en met een beetje planning is het eenvoudig genoeg om een bibliotheek te ontwikkelen die alle commando's omzet in duidelijk benoemde functies of methoden. Een optie is om een Arduino en een RS-232 transceiver te gebruiken om de voeding te regelen. Dit heeft het voordeel dat ook andere testfuncties kunnen worden geïntegreerd, zoals het schakelen van signalen met behulp van relais of het inlezen van analoge en digitale signalen. Als alternatief biedt Python de module pySerial [3]. Met wat eenvoudige codering (**listing 1**) kan de commando-interface worden geïmplementeerd als een Python-module en kan geautomatiseerde besturing worden gerealiseerd. Met wat leeswerk in de documentatie is het ook mogelijk om een TCP/IP-naar-serieel verbinding te implementeren [4]. Deze functionaliteit is gedefinieerd in de experimentele RFC2217 [5] memo en staat een PC op afstand toe om een seriële interface te configureren en communicatie te implementeren.

Signalen onderzoeken

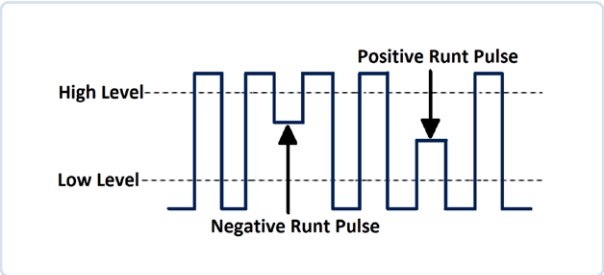
Oscilloscopen kunnen ook op afstand bediend worden. Met hun uitgebreide mogelijkheden, van opname van analoge en digitale signalen tot FFT's, kunnen ze gebruikt worden voor diverse geautomatiseerde tests. Sommige storingen zijn bijvoorbeeld moeilijk te vinden door een complexe reeks gebeurtenissen die in een specifieke volgorde moeten plaatsvinden om ze te veroorzaken. Als je team eenmaal heeft bepaald hoe de storing wordt veroorzaakt, is de volgende fase het configureren van de oscilloscoop om relevante signalen op te vangen die helpen de primaire oorzaak vast te stellen.

Boards als Arduino en Raspberry Pi zijn geweldig voor het snel ontwikkelen van een testopstelling die een storing herhaaldelijk kan opwekken, met behulp van analoge uitgangen en digitale signalen, en indien nodig aangevuld met een relais of MOSFET. Ze kunnen ook nauwkeurigee triggersignalen leveren voor een oscilloscoop, zodat het juiste deel van de relevante signalen wordt opgeslagen voor latere analyse.

Veel oscilloscopen hebben USB- en LAN-interfaces, maar sommige bieden alleen ondersteuning voor eigen software of toegang tot een webserver voor configuratie. Er is echter een USB-klassespecificatie voor testen en meten beschikbaar, die bekend staat als USBTMC [6]. Net als opslagapparatuur zoals USB-drives, en human-interface devices (HUD) zoals je muis en toetsenbord, biedt deze klasse voorgedefinieerde commando's die geschikt zijn voor de interfacing met test- en meetapparatuur. Apparaten zoals de B&K Precision 2567B [7], een 200-MHz, 2-GSa/s mixed signal oscilloscoop (MSO), ondersteunen deze interface. Met vier analoge kanalen, een 16-kanaals digitale poort, een ingebouwde 50-MHz arbitraire golfvormgenerator en geavanceerde triggers is hij eenvoudig te configureren met behulp van zijn grote 10,1" capacitieve touchscreen (**figuur 3**). Maar het gaat net zo gemakkelijk via USB.



Figuur 3. Met een USBTCM-compatibele USB-interface is de B&K Precision 2560B-serie MSO's gemakkelijk te automatiseren met Python (bron: B&K Precision).



Figuur 4. De runt-trigger is een slim mechanisme om pulsen te vinden die niet volledig stijgen of dalen (bron: B&K Precision).

Figuur 5. De schijfvormige Moku:Lab is een twaalf-in-één instrument, en kan geconfigureerd worden met behulp van een Python API (bron: Liquid Instruments).

Configuratie op afstand

Ook hier is Python de aangewezen programmeertaal, dankzij het USBTMC-project [8] van Alex Forencich, dat gehost wordt op GitHub [9]. De module draait onder Linux, maar kan vragen dat de machtigingen op de juiste manier worden aangepast. Onder Windows moeten eerst *PyUSB* [10] en *libusb* [11] worden geïnstalleerd.

De eerste stappen zijn relatief eenvoudig. Alvorens te beginnen zijn de vendor- (VID) en product-ID (PID) nodig, twee USB-specifieke referentiewaarden die een USB-product identificeren. Onder Linux gaat dat eenvoudig via de opdrachtregel met `lsusb` zodra het apparaat is aangesloten. De resulterende lijst geeft de benodigde details. In Windows kunnen deze waarden worden bepaald via Apparaatbeheer en de eigenschappen van het apparaat. In beide gevallen zijn de resulterende waarden 16-bit hexadecimaal

Alles wat dan nog gedaan moet worden is het importeren van de *usbtmc*-module in je Python-code en het gebruiken van de beschikbare programmeerinterface (API). Het lijkt veel op het printen van tekst naar het scherm en het evalueren van toetsenbordinvoer. USBTMC is eigenlijk gewoon een wrapper om te communiceren met geschikte test- en meetapparatuur. De besturingscommando's zijn ASCII-strings met diverse opties, zoals beschreven in de programmeerhandleiding van de oscilloscoop [12]. Indien beschikbaar kunnen ook VISA resource-strings [13] gebruikt worden.

De VID en PID zijn uniek voor het product, niet voor het individuele apparaat. Als er dus twee of meer van hetzelfde apparaat zijn aangesloten, kunnen ze onafhankelijk van elkaar worden aangesproken met een derde parameter, het serienummer. Dit staat meestal op een etiket of kan worden verkregen met het `*IDN?` commando (**listing 2**).

Tegenwoordig onderscheiden oscilloscopen zich niet veel meer van elkaar afgezien in bandbreedte, samplingsnelheid en geheugendiepte. Maar af en toe duikt er een kleine voorziening op die interessant is. De B&K-serie heeft een runt-trigger (**figuur 4**), een mogelijkheid die een triggerpuls levert wanneer een signaal de ene drempelwaarde (bijvoorbeeld hoog) passeert, maar de andere (bijvoorbeeld laag) niet. Bij storingsanalyse kan elk detail behulpzaam zijn.

Testapparatuur zonder display

Red Pitaya doorbrak de standaard op het gebied van testapparatuur en vroeg zich af of een meetapparaat wel een eigen display nodig heeft. Dankzij een krachtige FPGA en een Ethernet-interface levert je PC of laptop de gebruikersinterface. Liquid Instruments heeft dezelfde aanpak gekozen met zijn Moku:Lab [14], een veelzijdig apparaat met vijf tot twaalf instrumenten in één schijfvormige behuizing. Het biedt een reeks analoge in- en uitgangen en is ontworpen als een klein laboratorium.

Aan de voorzijde zijn vier BNC-connectoren aangebracht (**figuur 5**). Het rechter tweetal levert de analoge uitgangen, met een samplingsnelheid van 1 GSa/s per kanaal bij 16bit-resolutie en een bandbreedte (–3 dB) van >300 MHz. Het linker tweetal vormt de analoge ingangen, met een bandbreedte (–3 dB) van 200 MHz in 50 Ω en een samplingsnelheid van 500 MSa/s per kanaal bij 12bit-resolutie. De interne tijdbasis biedt een nauwkeurigheid beter dan 500 ppb. Een triggeringang is ook aanwezig, samen met aansluitingen die synchronisatie van meerdere units mogelijk maken. Bedrade connectiviteit wordt geboden via een Ethernetpoort en een USB-interface, en een tweede USB-voedingspoort is beschikbaar om een tablet op te laden. Tenslotte zijn er een SD-kaartslot en een gelijkstroomingang.

Hoewel het apparaat bedrade interfaces heeft, is het eigenlijk ontworpen om te worden gebruikt in combinatie met een iPad via WiFi (802.11 b/g/n) en de bijbehorende app. Via de gebruikersinterface kunnen twaalf instrumenten worden geselecteerd, variërend van het bekende, zoals oscilloscoop en spectrumanalyzer, tot het exotische, zoals PID-regelaar en laser lock box. Het instrument functioneert ook als datalogger. De grootte van je SD-kaart is de grens voor de gegevensopslag bij maximaal 100 kSa/s bemonsteringssnelheid, terwijl de interne geheugencapaciteit het instrument geschikt maakt voor snelheden tot 1 MSa/s.

Natuurlijk biedt de Moku:Lab, zoals alle moderne instrumenten, een Python API en ondersteuning voor MATLAB en LabVIEW. Ondersteund door talrijke voorbeelden

Listing 2. USBTMC in Python gebruiken om een B&K Precision-oscilloscoop aan te sturen.

```
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'
```

(**listing 3**) kan het apparaat ook snel geïntegreerd worden in een geautomatiseerde testopstelling. Het is waarschijnlijk bijzonder geschikt voor het op grote schaal testen van RF-apparaten en het sorteren van componenten.

Tijd besparen, nauwkeurigheid verbeteren

Testapparatuur vormt de ogen van de technicus, die laten zien wat er in complexe systemen gebeurt. Maar ze hebben hun beperkingen. Sporadische gebeurtenissen zijn per definitie moeilijk op te sporen, en een oplossing kan alleen worden verzonnen als een mogelijke oorzaak bekend is. API-ondersteunde programmeerinterfaces op test- en meetapparatuur worden tegenwoordig steeds gebruikelijker, variërend van eenvoudig tot verfijnd. En steeds vaker is het eenvoudig te leren Python de programmeertaal bij uitstek. Als je meetprobleem moeilijk te triggeren is, steeds weer wijzigingen nodig heeft om een enkele testronde te voltooien, of gewoon saai en repetitief is (wat leidt tot bedieningsfouten), dan is automatisering niet zo ingewikkeld als je misschien dacht.

vertaling: Hans Adams — 230046-03

Vragen of opmerkingen?

Hebt u technische vragen of opmerkingen naar aanleiding van dit artikel? Stuur een e-mail naar de auteur via stuart.cording@elektor.com of naar de redactie van Elektor via redactie@elektor.nl.

Listing 3. Een willekeurige golfvorm genereren met de Moku:Lab in 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()
```

WEBLINKS

[1] PSU Sequencer, Aim-TTi: http://bit.ly/40jGiCQ
[2] EL-R serie gelijkspanningsvoedingen, Aim-TTi: http://bit.ly/3kWZmXk
[3] pySerial-module: http://bit.ly/3YgSJgZ
[4] Voorbeeld pySerial TCP/IP Bridge: http://bit.ly/3kTDKem
[5] G. Clark, "RFC 2217 Telnet Com Port Control Option," Cisco Systems, Inc., oktober 1997: http://bit.ly/3jcQGM4
[6] "Universal Serial Bus Test and Measurement Class Specification (USBTMC)," USB Implementers Forum, Inc., april 2003.: http://bit.ly/3YbJY7L
[7] Model 2567B-MSO, B&K Precision: http://bit.ly/3kN3rNy
[8] A. Forencich, "Python USBTMC," juli 2014: http://bit.ly/3HMLeZN
[9] A. Forencich, python-usbtmc, GitHub Project: http://bit.ly/3wK4i4r
[10] PyUSB-software: http://bit.ly/3WRFW3l
[11] libusb-software: http://bit.ly/3wLEvIY
[12] "Programming Manual 2560B Series", B&K Precision, september 2022: http://bit.ly/3WRDYjs
[13] "VISA Resource Syntax and Examples", National Instruments Corp., mei 2022: http://bit.ly/3XK3okh
[14] Moku:Lab, Liquid Instruments: http://bit.ly/3WT6bXb