

eevidtron episode 002

Controlling Measurement Equipment with
Python and SCPI via VXI-11 and USBTMC

GPIB and SCPI

- GPIB
 - General Purpose Interface Bus
 - Introduced by HP in 1972
 - Standardized as IEEE-488 in 1975
 - Bit-parallel interface
- SCPI
 - Standard Commands for Programmable Instruments
 - Common syntax and structure for commands
 - Also introduced generic commands

VXI-11 and LXI

- VXI-11
 - Provides a virtual GPIB interface over RPC and TCP/IP
 - Can be accessed from Python using PyVXI11
<https://pypi.python.org/pypi/PyVXI11>
- LXI
 - LAN eXtensions for Instrumentation
 - Introduced in 2005 by Agilent and VTI Instruments
 - LXI uses VXI-11 for device discovery
 - Most LXI Devices support full VXI-11, but it is optional

USBTMC

- USB Device Class for Test and Measurement devices
- Can be accessed with usbtmc Linux kernel driver
- I wrote simple class with PyVXI11-like API
 - see Video description for GitHub link
- Windows users: Simply use VISA using PyVISA:
<https://pyvisa.readthedocs.org/>

Udev rules for USBTMC

/etc/udev/rules.d/90-usbtmc.rules

```
# make all USBTMC devices world read- and writable
KERNEL=="usbtmc*", MODE="0666"

# special names for known devices
DEVPATH=="/devices/pci0000:00/0000:00:12.2/usb1/1-2/1-2.3/1-2.3:1.0/usb/usbtmc*", SYMLINK+="usbtmc_fgen"
DEVPATH=="/devices/pci0000:00/0000:00:12.2/usb1/1-2/1-2.4/1-2.4:1.0/usb/usbtmc*", SYMLINK+="usbtmc_scope"
```

Hint for creating udev rules files:

Run “`sudo udevadm monitor --property`” and plug in the device. All relevant information about the device will be printed to the terminal. Alternatively type “`cat /var/log/udev`” to view details on devices that have been present during boot up.

VISA and VISA Addresses

- VISA
 - Virtual Instrument Software Architecture
 - Widely used API for communicating with instruments
 - NI-VISA is a VISA implementation by National Instruments, providing bindings to GPIB, VXI-11, and many other interfaces
 - Unfortunately there is no 64-bit Linux version of NI-VISA
- VISA Addresses for VXI-11
 - E.g. “TCPIP::192.168.1.20::INSTR”
 - `Vxi11Device('192.168.1.20', 'inst0')`
 - E.g. “TCPIP::192.168.1.20::foobar::INSTR”
 - `Vxi11Device('192.168.1.20', 'foobar')`

Demo: Query Device Identification

VXI-11 (demo_idn_vxi11.py)

```
# boilerplate code
from __future__ import division
from __future__ import print_function

# import PyVXI11
from vx11Device import Vxi11Device

# connect to instrument
dev = Vxi11Device('192.168.1.20', 'inst0')

# query and output identification string
answer = dev.ask('*IDN?').strip()
print(answer)
```

USBTCMC (demo_idn_usbtmc.py)

```
# boilerplate code
from __future__ import division
from __future__ import print_function

# import UsbtmcDevice
from usbtmcDevice import UsbtmcDevice

# connect to instrument
dev = UsbtmcDevice('/dev/usbtmc0')

# query and output identification string
answer = dev.ask('*IDN?').strip()
print(answer)
```

SCPI Command Syntax

- Common Command Examples:
 - `*RST`
 - `*RCL 3`
- Common Query Examples:
 - `*IDN?`
 - `*TST?`
- Instrument Control Command Examples:
 - `:OUTPUT:TIMER 10,5,1,60`
 - `:OUTPUT:TIMER:STATE ON`
- Instrument Control Query Examples:
 - `:OUTPUT:TIMER? 10`
 - `:OUTPUT:TIMER:STATE?`

Programming Manuals

- Instruments with SCPI support usually come with a “Programming Manual” with descriptions of all SCPI commands the device supports
- Example descriptions of SCPI commands and queries:

```
SYSTem:LANGuage:TYPE {EN|CH}  
SYSTem:LANGuage:TYPE?
```

```
[SOURce:]CURRent[:LEVel][:IMMediate][:AMPLitude] {<current>|MINimum|MAXimum}  
[SOURce:]CURRent[:LEVel][:IMMediate][:AMPLitude]? [MINimum|MAXimum]
```

- All commands are case insensitive, leading colons are optional.
- Each mnemonic has a short (e.g. LANG) and a long (e.g. LANGUAGE) form. Choose either.
- Parts in [. . .] are optional
- Parts in { . . . | . . . } are alternatives
- Parts in [. . . | . . .] are optional alternatives
- Parts in < . . . > are values to fill in

PyVXI-11, PyVISA, UsbtmcDevice API

- Creation of device handle is different for each library:
 - `dev = Vxi11Device('192.168.1.20', 'inst0')`
 - `dev = visa.instrument('TCPIP::192.168.1.20::INSTR')`
 - `dev = UsbtmcDevice('/dev/usbtmc0')`
- Use `dev.write(command)` to send a command
- Use `answer = dev.read()` to read responses
 - Only use this when there actually is something to read
- Use `answer = dev.ask(command)` for queries
 - This is a convenience function that simply combines `write()` and `read()`
- Each library has many additional methods, but they are rarely needed.

Python Code Review

- Example #1: A simple SCPI shell
 - `shell.py`
- Example #2: Sampling RIGOL DG1000 built-in functions
 - `dg1000_builtins.py`
- Example #3: Decoding a PAL video signal
 - `paldecode.py`
- Libraries
 - `usbtmcDevice.py`, `rigol_ds2000.py`