2026 ASEE ANNUAL CONFERENCE June 21–24, 2026 | Charlotte Convention Center, NC
Host Sponsor UNC Charlotte
ASEE AMERICAN SOCIETY FOR ENGINEERING EDUCATION
Paper ID #52278

WIP: Automated Current-Voltage Device Characterization Using Networked Instruments in an Instrumentation Lab

Mr. Bradley Lane Kicklighter P.E., University of Southern Indiana

Brad holds a BS in Electrical Engineering from Rose-Hulman Institute of Technology (1989) and an MS in Electrical and Computer Engineering from Purdue University (2001).

His past work experience includes eleven years at Delphi (formerly Delco Electronics) as an Advanced Project Engineer, eleven years at Whirlpool Corporation as a Lead Engineer/Solution Architect, and three years at Ivy Tech Community College as an Instructor/Program Chair Pre-Engineering. Since 2015, he has been employed at the University of Southern Indiana as a Clinical Associate Professor of Engineering.

He holds three patents, has served as an IEEE section officer since 2004, and has been a Licensed Professional Engineer in the State of Indiana since 2005.

Brad is the current chair of the ASEE Instrumentation Division.

WIP: Automated Current-Voltage Device Characterization Using Networked Instruments in an Instrumentation Lab

Abstract

In the Fundamentals of Instrumentation course, junior-level Manufacturing Engineering Technology students gain practical experience with DC and AC circuits, sensors, actuators, digital logic, filters, and discrete semiconductors. A key lab exercise focuses on Ohm's Law and device characterization, where students measure the current-voltage (I–V) behavior of a resistor and an incandescent light bulb to determine linearity.

Traditionally, this lab required students to manually adjust voltages and record corresponding currents—a process that was both time-consuming and prone to error, particularly for nonlinear devices requiring fine voltage steps. To improve efficiency and allow students to focus more on data analysis, the lab was automated using a Python-based solution developed by the author.

The Python program uses the PyVISA library to control Ethernet-connected Keysight instruments (a power supply and a digital multimeter) using the Virtual Instrument Software Architecture (VISA) and Standard Commands for Programmable Instruments (SCPI) protocols. While similar labs in the literature often rely on USB-connected devices and LabVIEW, this implementation demonstrates the flexibility and effectiveness of a Python-based, networked approach.

The automated system was deployed for the first time during the Fall 2025 semester with a class of sixteen students. The results indicated improved efficiency, more accurate data collection, and greater student engagement with data interpretation and analysis.

This work in progress paper shows how automation using open-source tools can enhance the educational value of traditional electronics labs, and it offers a scalable model for modernizing instrumentation instruction in engineering technology programs.

Introduction

Fundamentals of Instrumentation, offered in the fall semester, is a junior-level course required for the Manufacturing Engineering Technology program at the University of Southern Indiana. It covers the basics of instrumentation, resistance, capacitance, inductance, impedance, voltage, current, power, DC circuit analysis, AC circuit analysis, RC/RL circuits, op-amp circuits, low-pass/high-pass passive filters, combinational digital logic circuits, diodes, bipolar junction transistors, wire sizing, and circuit protection devices.

It is important for students to understand the difference between linear and nonlinear devices. When Ohm's Law is introduced, students learn that devices are characterized by their current-voltage (I-V) behavior. A range of voltages is applied to the device, and the current at each voltage is measured. The data are plotted with voltage on the horizontal axis and current on the vertical axis. Resistors follow Ohm's Law and produce a linear plot that passes through the origin. Devices whose plots are not linear are considered nonlinear. Incandescent light bulbs and diodes are two such nonlinear devices.

Incandescent light bulbs with tungsten filaments are nonlinear because their resistance depends on temperature and the properties of tungsten [1][2]. As the voltage increases, the filament heats up, and its resistance increases. The current-voltage characteristic of tungsten-filament incandescent light bulbs follows a power-law equation [3]:

$$i = \text{sgn}(v)a|v|^b$$

where $a > 0$ and $0 < b < 1$.

If the voltages are positive, the equation simplifies to:

$$i = av^b$$

In the past, the Ohm's Law lab required students to manually adjust the voltage and record the corresponding current. This process was time-consuming and prone to error, especially for non-linear devices that required fine voltage steps. The author observed that students did not devote enough time or thought to plotting and analyzing the data to determine linearity. To improve efficiency and allow students to focus more on data analysis, the lab was automated using a Python-based solution developed by the author. The measurement automation for this lab uses networked instruments (a power supply and a digital multimeter).

Academic Use of Data Acquisition (DAQ) in the Literature

A review of the literature on automated measurements in academic settings identifies a range of communication methods, programming languages, and measurement types.

Communication methods include IEEE-488 [4]; Peripheral Component Interconnect (PCI) Bus [5]; and Universal Serial Bus (USB) [6], [7], [8], [9], [10], [11], and [12]. USB is the most widely used communication method.

Programming languages include LabWindows [4]; LabVIEW [6], [7], [8], [9], [10], and [12]; and Python [11]. LabVIEW is the most used programming language.

There is a wide range of measurement types. These include frequency response [6], [11], and [5]; voltage [8], [9], [4], and [5]; and I-V characteristic [7]. The most common types are frequency response and voltage.

Instrument Network

The classroom for Fundamentals of Instrumentation accommodates up to 16 students and has 8 sets of instruments (students work in pairs during labs), as shown in Figure 1. Each student has a Windows 11 desktop computer, and there are four Keysight instruments between each pair of computers:

- EDU36311A Smart Bench Essentials DC Power Supply

- EDU34450A 5½-Digit Dual-Display Digital Multimeter (DMM)
- EDUX1052G Digital Storage Oscilloscope (DSO)
- EDU33212A Waveform Generator

Figure 1: The classroom has 16 desktop computers and 8 sets of instruments.

To enable automated measurements in the Ohm's Law lab, the author networked the instruments. Each instrument set and each pair of computers is connected via USB-to-Ethernet adapters, Ethernet patch cables, and an 8-port Ethernet switch (see Figures 2, 3a, and 3b).

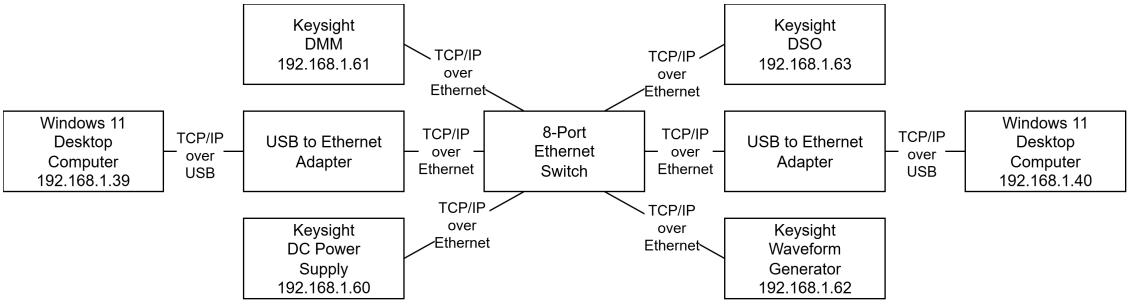
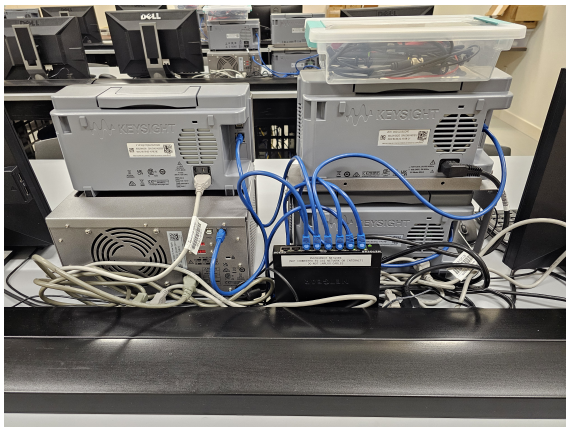


Figure 2: The instrument network comprises two Windows 11 desktop computers, two USB-to-Ethernet adapters, a DC power supply, a digital multimeter (DMM), a waveform generator, a digital storage oscilloscope (DSO), and an 8-port Ethernet switch. The static Internet Protocol (IP) addresses for each device are listed. Communication is Transmission Control Protocol/Internet Protocol (TCP/IP) over USB and Ethernet.

(a) Front view of computers and instruments.



(b) Rear view of instruments and Ethernet switch.

Figure 3: A set of instruments is shared between two students who each have a desktop computer that can control the instruments.

Since both computers are networked to the set of instruments, either student may access the instruments from their computer, though not simultaneously (an instrument can be controlled by only one computer at a time). This arrangement provides flexibility and avoids the need to change connections (unplugging and plugging cables) between the instruments and computers. If USB connections were used between the computers and instruments without the need to unplug and replug cables, four USB A/B switches would be required for each set of computers, and four USB ports per computer would be necessary (a more complex setup than the Ethernet approach).

The instrument network is separate from the university network, and static Internet Protocol (IP) addresses are assigned to desktop computers and instruments (see Figure 2). Communication over the instrument network is Transmission Control Protocol/Internet Protocol (TCP/IP).

I-V Characterization Python Program

The author wrote a Python program to control the power supply and DMM and to perform I-V characterization of a device. The PyVISA and PyVISA-py open-source Python packages are used to control the instruments [13]. These packages implement the Virtual Instrument Software Architecture (VISA) [14]. The instruments support the Standard Commands for Programmable Instruments (SCPI) [15]. SCPI commands for instrument control are sent via the VISA interface.

PyVISA is available on PyPI.org [16] and GitHub.com [17]. PyVISA-py is available on PyPI.org [18] and GitHub.com [19].

Program Development

To control instruments using a Python program, VISA drivers are required. Although Keysight recommends installing Keysight IO Libraries, the author found that this large installation package was unnecessary if PyVISA-py is installed instead. “PyVISA-py is a pure Python backend for the

PyVISA library that implements the Virtual Instrument Software Architecture (VISA) standard, allowing control of test equipment via Serial, USB, GPIB, and Ethernet without requiring proprietary drivers like NI-VISA" [19].

Before communicating with the instruments via Ethernet, it is recommended to configure static IP addresses on the instruments. For example, the power supply used by the program described here is assigned the address "192.168.1.60" (all power supplies in the room are assigned the same IP address). The resource path used by PyVISA-py to communicate with the power supply is: "`TCPIP::192.168.1.60::INSTR`".

Identify the SCPI commands that the instruments recognize to execute the commands and queries needed by the program. This information may be in the instrument user manual or in a separate "programming" manual. Note that SCPI commands vary somewhat between manufacturers and models.

The general flow of an instrument control program begins by creating a resource manager and instrument resources, then executing commands and queries, and finally closing communication and performing cleanup.

The program must instantiate a resource manager object : `rm = pyvisa.ResourceManager('@py')` (where `'@py'` directs the resource manager to use the PyVISA-py backend). Next, resources are opened to control the power supply and DMM: `ps = rm.open_resource(PS_PATH)` and `dmm = rm.open_resource(DMM_PATH)` (where `PS_PATH` and `DMM_PATH` are resource paths containing the IP addresses of the instruments).

Now, SCPI commands can be sent using the `.write()` method, and information such as readings, can be retrieved using the `.query()` method. For example, writing `ps.write('outp ON, (@2)')` to the power supply turns on channel 2 and `ps.write('volt 10, (@2)')` sets it to 10 V. Sending `dmm.query('meas:prim:volt:dc?')` to the DMM reads the voltage, and `dmm.query('meas:sec:curr:dc?')` reads the current (each query returns a string that must be converted to a floating point number).

After commanding and querying the instruments, it is good practice to return the instruments to local control using the SCPI command `syst:loc`, then close the resources and the resource manager using each object's `.close()` method (close the resource manager after closing all the resources).

Program Behavior

The program performs a DC voltage sweep from 0.1 V to 12.0 V in 0.1 V increments. The user may override the start, stop, and step voltage parameters. The power supply is set to output the specified voltage, and the DMM is set to measure the voltage across the device and the current through it. Once the sweep is complete, the voltage and current data are written to a comma-separated value (CSV) file named with a date/time stamp (i-v_data_YYYYMMDD-HHMMSS.csv). The user can open the CSV file in Microsoft Excel to plot and analyze the data.

See the Appendix for the program's source code.

Methodology

The Fundamentals of Instrumentation course is a lecture/lab course in which students are assessed through exercises, quizzes, labs, and exams. Lab 2: Ohm's Law has the following objectives:

1. Become familiar with the use of the DC power supply.
2. Become familiar with the use of the digital multimeter (DMM).
3. Know how to measure current and voltage in a DC circuit.
4. Apply and plot Ohm's Law.
5. Recognize non-linear devices.

Students work in pairs and submit one lab write-up per pair. The instruments used in the lab are the Keysight EDU36311A Smart Bench Essentials DC Power Supply and the Keysight EDU34450A 5½-Digit Dual-Display Digital Multimeter. The devices the students characterize in the lab are a 1 kΩ resistor and a 12 V incandescent light bulb.

The students perform the following steps in the lab: 1. The students use the DMM to measure the resistances of the resistor and the light bulb. 2. The power supply and DMM are connected to each device as shown in Figure 4; Figure 5 shows the setup. 3. The I-V characterization Python program is used to characterize each device with the default parameters. XY scatter plots are created in Microsoft Excel from the resistor and light bulb data. A trend line is added to each plot (linear for the resistor and power law for the light bulb), along with the equation and R^2 value (see Figure 6 for an example of the light bulb plot). 4. Students answer the following questions: Does the percent difference between the measured and calculated resistances indicate that Ohm's Law matches reality? Does the plot indicate that the device is linear? Why? 5. Each pair of students completes and submits a lab write-up that includes initial device resistance measurements, answers to the questions, and plots of the I-V characteristics with trend lines. The Microsoft Excel spreadsheets are also submitted as supporting evidence of their work.

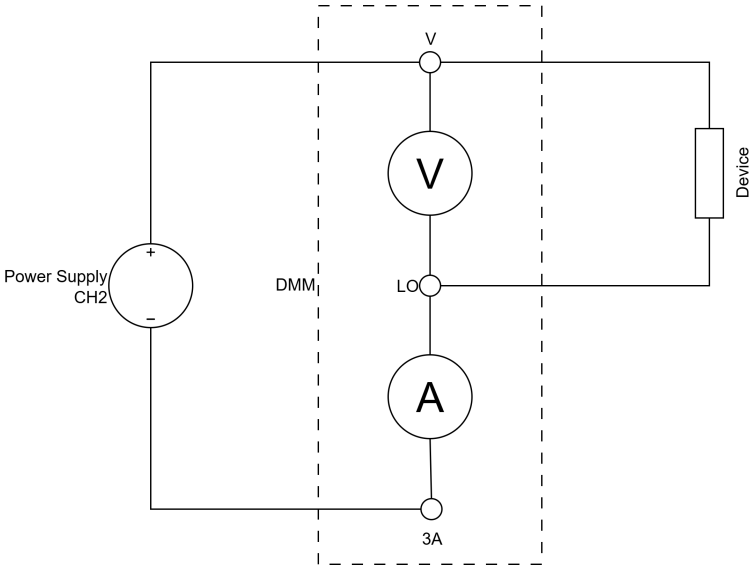


Figure 4: Circuit diagram showing the power supply and the digital multimeter connected to the device being characterized.

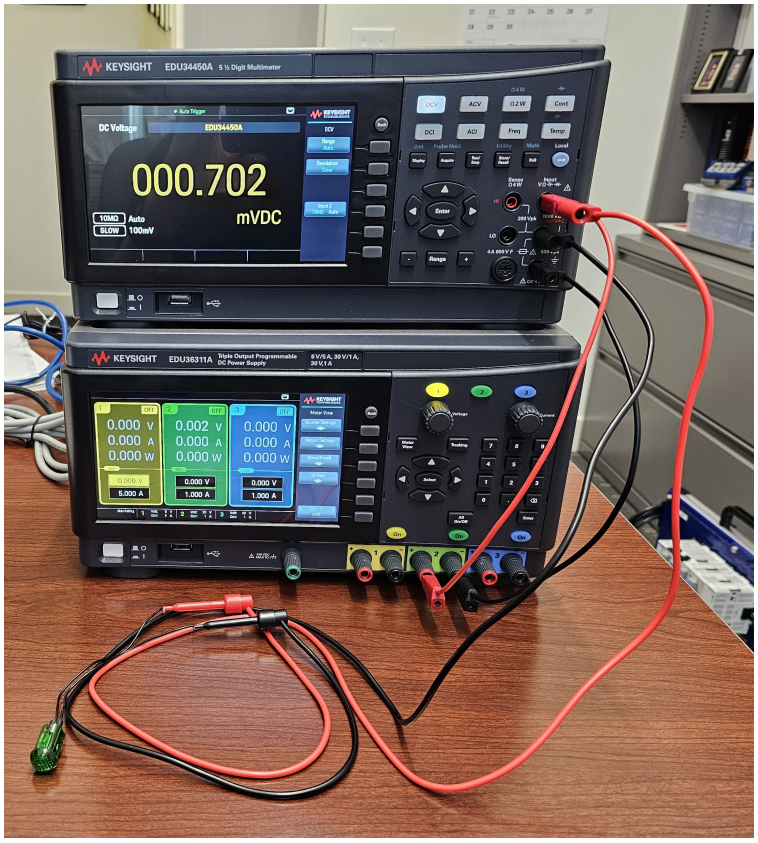


Figure 5: This picture shows how to interconnect the power supply, digital multimeter, and the device to be characterized. The light bulb is included in this setup.

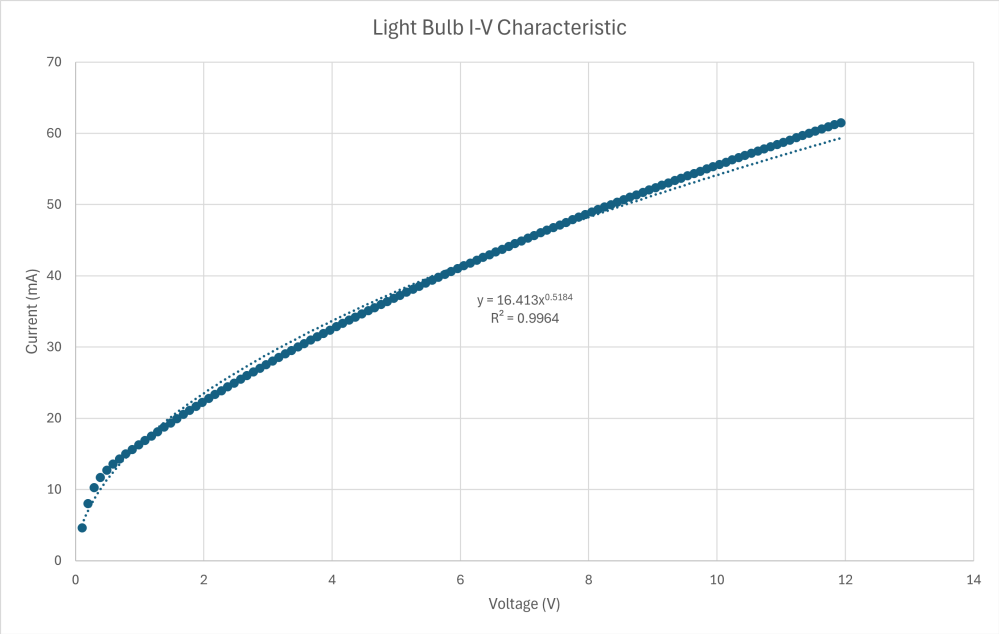


Figure 6: Plot of the incandescent light bulb I-V characteristic with a power-law curve fit.

Results

In Fall 2025, there were sixteen (16) students in the course and eight (8) lab groups. This was the first time the I-V Characterization Python program and networked instruments were used. With only eight sets of lab rubric scores for automated device characterization, it is not possible to provide a statistical comparison of results between manual and automated measurements. However, after hearing about the previous version of Lab 2, the students indicated that they appreciated what the Python program could do for them. They noted that data collection did not take long and appreciated not having to write it down manually.

In the previous version of Lab 2, students measured ten data points for the resistor (1 V steps) and 23 for the light bulb (0.5 V steps). All data was manually recorded. While 10 data points were sufficient for the resistor, there were not enough near the beginning of the light bulb plot, where resistance was changing more rapidly. In the current version of Lab 2 using the I-V Characterization Python program, 120 data points are measured (0.1 V steps) and automatically recorded.

The author observed that students were better able to distinguish between linear and nonlinear behavior when trend lines were added to the plots.

Conclusions and Future Work

While more data are needed for Lab 2 to make a statistical comparison between manual and automated measurements, the following can be concluded.

Open-source tools such as Python, PyVISA, and PyVISA-py can be useful in the lab for automating tedious, error-prone measurements. More detailed data can be collected, freeing up time for

students to devote more thought to their analysis, and the collected data is free of recording errors. This approach can be applied to other measurement types, and other classrooms can network their equipment. It was applied to automatic screen capture from the digital storage oscilloscope used in later labs in Fundamentals of Instrumentation in Fall 2025.

References

[1] P. Moon, *The Scientific Basis of Illuminating Engineering*, 1st ed. New York, NY: McGraw-Hill Book Company, Inc., 1936.

[2] D. Van Horn, "Mathematical and Physical Bases for Incandescent Lamp Exponents," *Illuminating Engineering*, vol. 60, no. 4, pp. 196–202, 1965.

[3] J. Martinez and D. Krug, "Voltage-Current Characteristic of Incandescent Lightbulbs: Measurement and Analysis," Tech. Rep., Jul. 2013. [Online]. Available: https://www.researchgate.net/publication/251567174_Voltage-Current_Characteristic_of_Incandescent_Lightbulbs_Measurement_and_Analysis

[4] D. Ilić and K. Poljančić, "Computer-Controlled Experiments in the Course "Measurements in Electrical Engineering"," in *Proceedings of the 12th IMEKO TC4 International Symposium, Part 2*. Zagreb, Coatia: IMEKO, Sep. 2002, pp. 539–542.

[5] M. Parten, "Using Virtual Instruments In A Measurements Laboratory," in *2003 Annual Conference*. ASEE, Jun. 2003, pp. 8.1265.1–8.1265.13. [Online]. Available: https://peer.asee.org/11527

[6] M. G. Guvench and M. Ye, "Automated Bode-Magnitude and Bode-Phase Frequency Response Testing of Analog Systems and Electronic Circuits Using Standard USB-Interfaced Test Instruments," in *2015 ASEE Annual Conference & Exposition*. ASEE, Jun. 2015, pp. 26.271.1–26.271.12. [Online]. Available: https://peer.asee.org/23610

[7] ——, "Automated Measurement of Power MOSFET Device Characteristics Using USB Interfaced Power Supplies," in *2016 ASEE Annual Conference & Exposition*. ASEE, Jun. 2016. [Online]. Available: https://peer.asee.org/26354

[8] D. R. Loker and S. A. Strom, "Automated Test & Measurement System for a Power Supply and Control Board," in *2016 ASEE Annual Conference & Exposition*. ASEE, Jun. 2016. [Online]. Available: https://peer.asee.org/26358

[9] ——, "Innovative Laboratory Projects for a Measurements and Instrumentation Course," in *2019 ASEE Annual Conference & Exposition*. ASEE, Jun. 2019. [Online]. Available: https://peer.asee.org/32967

[10] G. E. Meyer and Y. Ge, "Instrumentation and Controls Instruction for Agricultural and Biological Engineering Students," in *2016 ASEE Annual Conference & Exposition*. ASEE, Jun. 2016. [Online]. Available: https://peer.asee.org/25752

[11] C. K. Frederickson, "Introducing Automation for Data Acquisition and Analysis in a Sophomore-level Electronics Course," in *2023 ASEE Annual Conference & Exposition*. ASEE, Jun. 2023. [Online]. Available: https://peer.asee.org/43832

[12] J. W. Dyer, D. Sandmann, and C. E. Davis, "Measurement and Automation: Experiential Learning Opportunity," in *2014 ASEE Annual Conference & Exposition*. ASEE, Jun. 2014, pp. 24.891.1–24.891.15. [Online]. Available: https://peer.asee.org/22824

[13] H. E. Grecco, M. C. Dartiailh, G. Thalhammer-Thurner, T. Bronger, and F. Bauer, "PyVISA: The Python instrumentation package," *Journal of Open Source Software*, vol. 8, no. 84, p. 5304, Apr. 2023. [Online]. Available: https://joss.theoj.org/papers/10.21105/joss.05304

[14] "Virtual instrument software architecture," *Wikipedia*, Feb. 2025. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Virtual_instrument_software_architecture&oldid=1273908358

[15] "Standard Commands for Programmable Instruments," *Wikipedia*, Dec. 2025. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Standard_Commands_for_Programmable_Instruments&oldid=1325325224

[16] "PyVISA: Python VISA bindings for GPIB, RS232, TCPIP and USB instruments." [Online]. Available: https://pypi.org/project/PyVISA/

[17] H. E. Grecco, M. C. Dartiailh, G. Thalhammer-Thurner, T. Bronger, and F. Bauer, "PyVISA: The Python instrumentation package," Apr. 2023. [Online]. Available: https://github.com/pyvisa/pyvisa

[18] "PyVISA-py: Pure Python implementation of a VISA library." [Online]. Available: https://pypi.org/project/PyVISA-py/

[19] "Pyvisa/pyvisa-py," pyvisa, Jan. 2026. [Online]. Available: https://github.com/pyvisa/pyvisa-py

Appendix

I-V Characterization Python program source code:

```
"""
 I-V Characterization
 This program controls a Keysight power supply and digital multimeter
 (DMM) over Ethernet using PyVISA and PyVISA-Py to characterize a
 device (voltage vs current).
 Power Supply: Keysight EDU36311A (192.168.1.60)
 DMM: Keysight EDU34450A (192.168.1.61)
 The user can use default values or enter the starting voltage, the
 ending voltage, and the voltage increment for the voltage sweep.
 The power supply is commanded to a voltage level and the DMM is
 queried for the voltage across the device and the current going
 through it.
 The I-V data is collected and written to a CSV file at the end of
 the voltage sweep.
 The CSV file is saved in the same folder as the program using the
 current date and time as the last part of the file name.
 """

__author__ = "XXXXXXXXXXX"
__version__ = "2025.09.10"

import time
import csv
import pathlib
import pyvisa

# Function to get DC sweep parameter from user and limit values
def get_sweep_param(prompt):
    value = float(input(prompt))
    if value < 0.0:
        print('Parameter clamped to 0.0')
        value = 0.0
    elif value > 30.0:
        print('Parameter clamped to 30.0')
        value = 30.0
    return value

print('I-V Characterization')
print('====================')

# SCPI commands
```

```
PS_ON = 'outp ON, (@2)' # Turn on CH2 power supply output
PS_OFF = 'outp OFF, (@2)' # Turn off CH2 power supply output
PS_0_V = 'volt 0, (@2)' # Set CH2 power supply output to 0 V
DMM_MEAS_V = 'meas:prim:volt:dc?' # Measure DC voltage (primary)
DMM_MEAS_I = 'meas:sec:curr:dc?' # Measure DC current (secondary)
DMM_SEC_DISP_OFF = 'disp:wind2 OFF' # Turn off secondary display
LOCAL_CONT = 'syst:loc' # Return instrument to local control

# Time between setting voltage and making measurements
SETTLING_TIME = 0.2

# Instrument resource paths and DC sweep parameters
PS_PATH = 'TCPIP::192.168.1.60::INSTR'
DMM_PATH = 'TCPIP::192.168.1.61::INSTR'
START_V = 0.1
STOP_V = 12.0
INC_V = 0.1

# PyVISA resource manager using PyVISA-Py as the VISA backend
print('Starting VISA resource manager')
try:
    rm = pyvisa.ResourceManager('@py')
except BaseException as e:
    print(f'Cannot start VISA resource manager: {e}')
    exit()

# Keysight EDU36311A Power Supply resource
print('Opening power supply')
try:
    ps = rm.open_resource(PS_PATH)
    print(ps.query('*IDN?').strip())
except BaseException as e:
    print(f"Cannot connect to instrument: {e}")
    exit()

# Keysight EDU34450A DMM resource
print('Opening DMM')
try:
    dmm = rm.open_resource(DMM_PATH)
    print(dmm.query('*IDN?').strip())
except BaseException as e:
    print(f"Cannot connect to instrument: {e}")
    ps.close()
    exit()
```

```
# Set power supply output to 0 V and turn it on
print('Initializing power supply')
ps.write(PS_0_V)
ps.write(PS_ON)

# Query the user for the sweep parameters
print(f'Current DC sweep parameters: Start {START_V} V, Stop {STOP_V}\
 V, Increment {INC_V} V')
resp = input('Do you want to enter new parameters (y/n)?: ').lower()
if resp == 'y':
    temp_start_v = get_sweep_param('Start: ')
    temp_stop_v = get_sweep_param('Stop: ')
    temp_inc_v = get_sweep_param('Increment: ')
    if temp_start_v < temp_stop_v and temp_inc_v < temp_stop_v:
        START_V = temp_start_v
        STOP_V = temp_stop_v
        INC_V = temp_inc_v
    else:
        print('One or more invalid parameters entered; default values used')

# List of voltages and currents with header
data = [['Voltage (V)', 'Current (A)']]

# Perform DC voltage sweep
curr_v = START_V
print(f'Running voltage sweep from {START_V:.1f} V to {STOP_V:.1f} V\
 by {INC_V:.1f} V increment')
# Step voltage from start_v to stop_v in inc_v
while curr_v <= STOP_V:
    ps.write(f'volt {str(curr_v)}, (@2)')
    time.sleep(SETTLING_TIME)
    dmm_v = float(dmm.query(DMM_MEAS_V))
    dmm_i = -float(dmm.query(DMM_MEAS_I))
    data.append([dmm_v, dmm_i])
    print(f"PS V: {curr_v:.1f} V  DMM V: {dmm_v} V  DMM I: {dmm_i} A")
    curr_v = curr_v + INC_V
print('DC sweep done')

# Set power supply output to 0 V, turn it off, and turn off DMM
# secondary display
print('Turning off power supply output and DMM secondary display')
ps.write(PS_0_V)
ps.write(PS_OFF)
dmm.write(DMM_SEC_DISP_OFF)
```

```
# Set power supply and DMM to local control
print('Returning instruments to local control')
ps.write(LOCAL_CONT)
dmm.write(LOCAL_CONT)

# Close resources and manager
print('Closing resources and resource manager')
ps.close()
dmm.close()
rm.close()

# Write V-I data to CSV file in program folder
data_file = (str(pathlib.Path(__file__).parent) +
    "\\" +
    "i-v_data_" +
    time.strftime("%Y%m%d-%H%M%S") +
    ".csv")
print(f'Writing CSV file: {data_file}')
with open(data_file, 'w', newline='') as csvfile:
    csvwriter = csv.writer(csvfile)
    csvwriter.writerows(data)

# Debug
#print(data)

input('Press enter to exit program')
```