Development of Open-Source Python Program for Lab Automation

Project Report

submitted in the partial fulfilment of the requirement of the award of the degree of Bachelors in Science in Physics

Submitted By

Swaraj V

Reg.No.35219054
Department of Physics
Cochin University of Science and Technology

Under the Supervision of

Dr. Vineeth Mohanan P

Assistant Professor, Department of Physics, CUSAT

DEPARTMENT OF PHYSICS

COCHIN UNIVERSITY OF SCIENCE AND TECHNOLOGY
COCHIN-682022, KERALA, INDIA
July 2022

Division for Research in Advanced Materials
(DREAM LAB)
Department of Physics
Cochin University of Science and Technology
Cochin-682022, Kerala, India

08-07-2022

Certificate

This is to certify that the project report entitled **Development of Open-Source Python Program for Lab Automation** is the bonafide work done by **SWARAJ V** (Reg No: 35219054), in the Department of Physics, Cochin University of Science and Technology, under the guidance of Dr.Vineeth Mohanan P for partial fulfilment of the requirements for the award of the Bachelor of Science degree in Physics.

Prof.Titus K Mathew
Head of Department
Department of Physics
CUSAT

Dr.Vineeth Mohanan P
Supervising Guide
Assistant Professor
Department of Physics
CUSAT

Declaration

I hereby declare that the project report entitled **Development of Open Source Python Program for Lab Automation**, submitted in partial fulfilment of the requirements for the award of the degree, Bachelor of Science in Physics is an authentic record of work done by me under the guidance **Dr.Vineeth Mohanan P**, Assistant Professor, Department of Physics, Cochin University of Science and Technology and has not been submitted to any institute for the award of any other degree.

SWARAJ V
IM.Sc.Physics
Department of Physics,
CUSAT, Kochi

Kochi
08/07/2022

Acknowledgement

I would like to express my gratitude for the support I have received from my friends, teachers, and mentors—Ph.D. scholars from the Physics Department at CUSAT. I am grateful to my guide Dr. Vineeth Mohanan P, whose guidance and advice helped me get through every step of completing my research. My profound appreciation goes out to Dr. Riju C. Isaac, our course coordinator, for assisting us in different ways. I want to extend my gratitude to Professor Titus K. Mathew, Head of the Physics Department, for providing us with a favorable environment for conducting experiments. I'm grateful for everything I have and want to express my sincere thanks to my friends, teachers, and family.

Mr. Swaraj V

Contents

Abstract

In this project we have implemented an open-source python program to automate lab equipment for performing experiments. This program is used to control a Keithley source measuring unit[1] for performing I-V characteristics of silicon diode and LED and also visualize the data after completion of the experiment. The main motivation of this project is to replace expensive and proprietary software such as LabVIEW used in research laboratories with simple, cost-free and popular programming language Python. We expect this to be a small yet effective step towards cutting down the cost of performing research work.

1 Introduction

Conventionally lab automation of instruments is possible only when a visual programming software like LabVIEW[2] is available. These visual programming software though very intuitive and easily to use, they are expensive and has annual subscriptions between 50,000 - 280,000 Rs. A vast majority of researchers are not aware of this programming language and cannot afford it. By developing a low CPU intensive open-source Python program every one will get the opportunity to perform lab automation provided suitable hardware are available. Since one has access to the source code, anyone with reasonable programming skills can make their customisation in the software and add more utilities. In this project, we developed an open source Python program to automate Keithley 2400 source measure unit (SMU) and to perform I-V Characteristics measurement of a silicon diode and an LED and then plot the resulting data.

2 Hardware

The project made use of both hardware and software components listed below.
Hardware: The hardware for lab automation consists of a computer with the required drivers installed, a Keithley 2400 SMU, a GPIB interface, a silicon diode, and a LED (Light Emitting Diode).
Software: The software section consists of programming language python, different python packages and libraries, Keithley's GPIB to USB adapter drivers, and National Instruments visa library manager.

2.1 Keithley 2400 SMU

The Keithley 2400 SMU is the combination of a high impedance digital multi-meter, a precision voltage source, and a precision current source. It can function simultaneously as a source and a measuring unit (for example, in an I-V measurement, the SMU can source voltage and measure current).

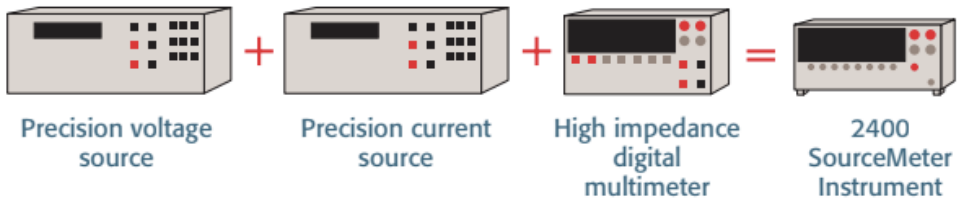


Figure 1: Keithely's SMU as a combination of precision voltage source, current source, high impedance digital multimeter.

Depending on the requirement, this instrument can function as a voltage source, a current source, a voltage metre, a current metre, or an ohm-meter. This SMU is a precision, low noise instrument.

(a) (b)

Figure 2: The front (a) and back panels (b) of Keithley 2400. The Front panel has various control buttons for configuring the equipment and the back panel has power, communication ports etc.[1]

The Keithley 2400 features a source voltage range of 5 μV to 210 V, a measure voltage range of 1 μV to 211 V, and a source current range of 50 pA to 1.05 A and measure current from 10 pA to 1.055 A. The controls, display output, on/off indicator, source measure ports, are all located on the front panel and ports for IEEE-488, RS-232, line input, and interlock digital input and output are located in the rear panel.

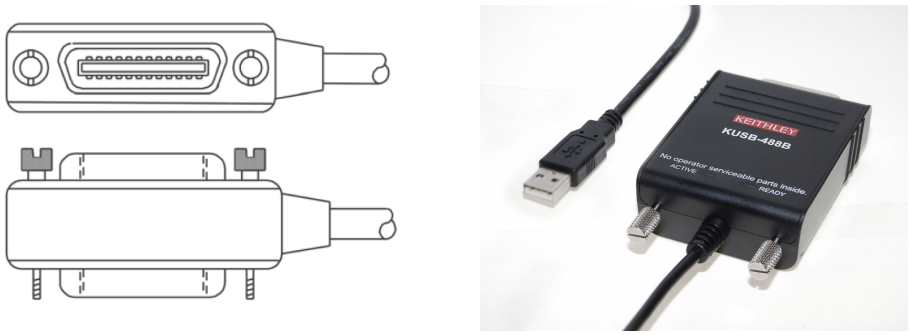
2.2 GPIB Interface

General purpose interface bus (GPIB [3]) is an interface connection necessary to establish communication between computer and the measuring instrument. GPIB includes IEEE-488 and IEEE-488.2 protocols. It enables a handshake communication with many instruments connected to a PC configured to GPIB address[1]. In IEEE-488 the instrument connected in the bus can be categorised into three ie, **controller**, **listener** and **talker**

Controller- It is the controlling device, usually it is a PC which sends commands to Listener. **Listener**-They are the instruments which listens the commands form Controller, usually

[1]It contains information about the channel number GPIB interface where instrument is connected

they are the instruments connected to a GPIB bus (SMU).
Talker-They are the intermediate devices (interface) which channels the connection between listeners and controllers, they exchanges all the instructions to and for controller and listener (GPIB interface equipment).



(a) (b)

Figure 3: (a) An illustrative figure of GPIB connector, (b) GPIB to USB converter from Keithley equipment (KUSB)[1]

3 Software

The software components includes Python, all of the python packages and libraries, Keithley drivers, and NI GPIB libraries.

3.1 Python and Python packages

The required packages are then installed using pip after installing Python 3.7, These are the installed Python libraries[4].
PyVISA-This Python library enables bus system and PC connectivity. We can control any type of measurement device, regardless of interface, thanks to the PyVISA[5] package. It supports ethernet, GPIB, RS232, and USB.
NumPy-It is one of the core Python packages for scientific computation. The Numpy library allows users to perform any mathematical, logical, basic statistical operation, random simulation, etc.
matplotlib.pyplot-matplotlib.pyplot python library enables plotting. It follows MATLAB-like plotting in python.
Time-This default python library can perform various time related functions.
Os- Os python module gives more functionality to the operating system like saving and exporting files.
CSV- CSV (Comma Separated Values) python module exports the python list[2] or string[3] into csv file.

[2]lists are the python data types which stores multiple data types/information in one string/one variable

[3]string are arrays of bytes representing unicode characters

3.2 Drivers

The software known as drivers is what links hardware to the operating system of a computer or other device. Here, KUSB-488B [6] and NI VISA Library[4] are the two drivers that we utilise to connect our system (a PC) to GPIB and, from there, to the SMU (like Keithley 2400). It is a keithley driver for GPIB to USB converter devices, it can be downloaded from keithley's official website. IEEE-488.2 [7][5] configuration and has a data transfer rate of 1.5 megabytes per second.

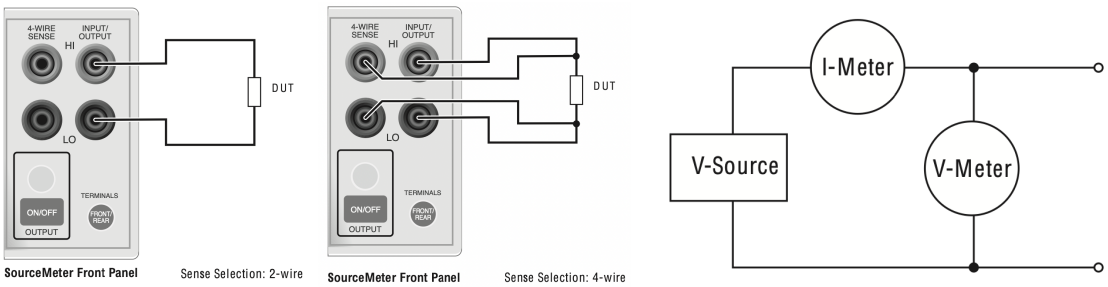
NI VISA is a standard for instrumental communication with systems containing GPIB, Serial, Ethernet, and USB interfaces. The NI-VISA Library is a package from National Instruments for VISA implementation. Here the diode/sample is connected to the probes, voltage is applied and Current is measured (or vice versa). How a source measuring device operates or what is the principle behind source measuring?

4 Measurements using SMU

There are two sensing methods for Source Measure i.e., two wire local sense connection and four wire remote sense connections

4.1 Two Wire Local Sensing Connections

These type of connection is used only when error contributed by an IR drop is not significant in the experiment i.e., in two wire configuration for measuring or sourcing current, current



(a) (b) (c)

Figure 4: SMU measurement connection, (a) Local sensing, (b) Remote sensing, (c) An illustration of current measurement sourcing voltage[1]

is in series to the circuit. Since the current is constant in series circuit the error contribution will be neglected. (Here we are using this sensing method for measuring current).

[4]It is a collection of National Instruments' VISA Driver

[5]IEEE 488.2 defines standard formats, protocols, and common commands for IEEE 488 programmable instrumentation

4.2 Four Wire Remote Sensing Connections

When measuring or sourcing voltage error associated with IR drop in the test lead is visible. There fore four leads are connected across the device under test (DUT). Here constant voltage (also resistance) source or accurate voltage measure must be maintained. so, in I-V Measurement, current is the measured Quantity. and two wire local sense connection must be used.

4.3 Setting up SMU for GPIB communication

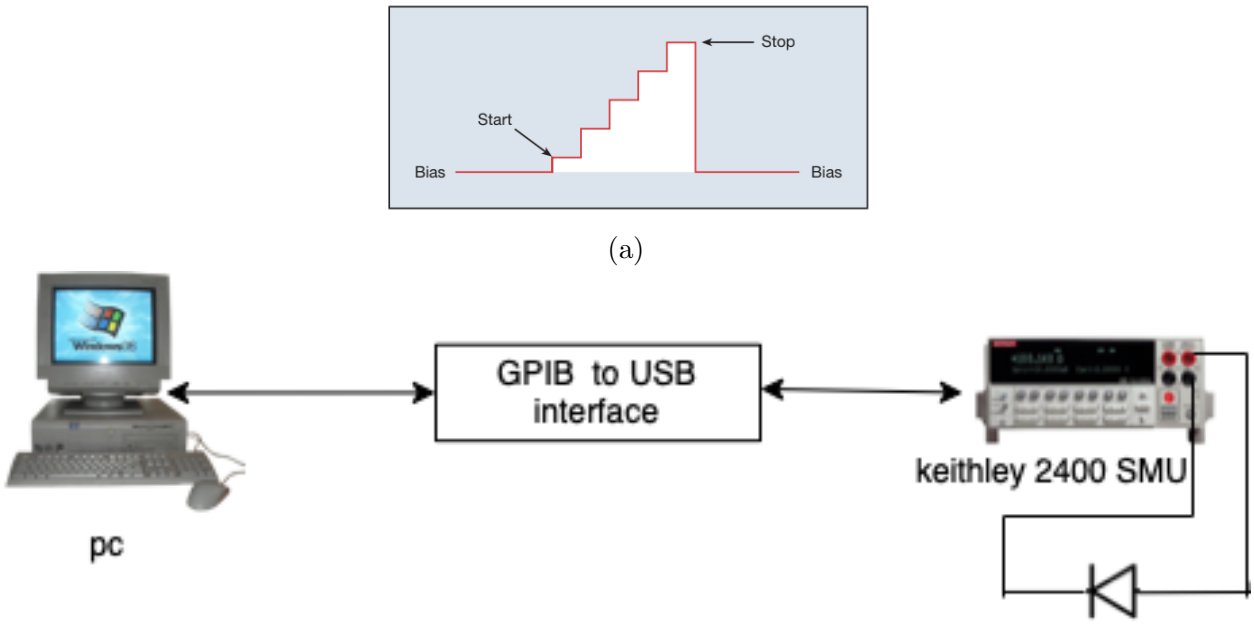


Figure 5: (a) Schematic illustration of Linear Staircase Sweep that we plan to implement. (b) An illustration of diode measurement connection with PC

The following steps are followed for establishing communication between computer and the SMU via GPIB device.

- A Keithley driver, KUSB-488B is downloaded from their official website and installed
- NI-VISA library is downloaded from national instruments' official website and installed
- PC is connected to Keithley 2400 SMU through GPIB to USB adapter
- Upon connecting the GPIB to our system, a device manager notification is displayed. Also a green indicator will glow if the connection is established successfully.
- A test diode (DUT) is connected to the SMU. Here the P and N regions are connected to the HI and LO terminals of the SMU as shown in Figure **??**

4.4 Python Coding

Once the necessary hardware and software are installed, then the system is ready for being computer controlled.

The instrument must now be given a handshake command so that, in the event that it is functional, it will respond with the name and address of the instrument.

The following python code will output the list of all devices connected via GPIB or there ports

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

The python output console will display the following message stating the GPIB address of the available instrument ie,

$'GPIB0::26::INSTR'$

The code below sends a query to the instrument's GPIB address, which is 26, and expects a response about the instrument connected to channel 26. If this procedure is successful, we can assume that the instrument and computer are connected properly.

```
import pyvisa
rm = pyvisa.ResourceManager()
rm.list_resources()
my_instrument = rm.open_resource('GPIB0::26::INSTR')
print ( my_instrument.query('*IDN?'))
```

The output will be the following string:

$KEITHLEY INSTRUMENTS INC., MODEL 2400, 4067084, C32 Oct 4201014 : 20 : 11/A02/U/K$

To configure the instrument, the GPIB interface's settings are cleared, the terminals are set to the front, the source is set to voltage, the compliance level is written, and the compliance range is set to auto to prevent instrument damage, and the output is turned on. (gave a initial voltage of 10 V, maximum voltage 20 V, current compliance value 10e-3 and number of steps 100).

```
#CONFIG INSTRUMENT and compliance setting
keithley.write("*RST") #resetting GPIB
time.sleep(0.5)
keithley.write(":ROUT:TERM FRON") #terminal set to front
keithley.write(":SOUR:FUNC:MODE VOLT") # source is set to voltage
keithley.write(":SENS:CURR:PROT:LEV " + str(CurrentCompliance))
keithley.write(":SENS:CURR:RANGE:AUTO 1")

PROTECTION = input("Enter Compliance Value : ")
PROTECTION = float(PROTECTION)
PROTECTION_FUNCTION = ":SENS:CURR:PROT "+str(PROTECTION)
keithley.write(PROTECTION_FUNCTION)
#CurrentCompliance = PROTECTION
```

Once the compliance current, voltage limits are set, then one can enter the voltage sweep parameters via the following code.

```
#initial voltage/ starting voltage
INITIAL_VOLTAGE = input("Enter Initial Voltage : ")
INITIAL_VOLTAGE =float(INITIAL_VOLTAGE)
INITIAL_VOLTAGE_FUNCTION = ":SOUR:VOLT:LEV "+str(INITIAL_VOLTAGE)
keithley.write(INITIAL_VOLTAGE_FUNCTION)

#maximum voltage
MAXIMUM_VOLTAGE = input("Enter Maximum Voltage : ")
MAXIMUM_VOLTAGE =float(MAXIMUM_VOLTAGE)
VOLTAGE_RANGE_FUNCTION = ":SOUR:VOLT:RANG "+str(MAXIMUM_VOLTAGE)
keithley.write(VOLTAGE_RANGE_FUNCTION)

start = INITIAL_VOLTAGE
stop = MAXIMUM_VOLTAGE
NUMBER_POINTS = input( "Enter Number of Points : " )
keithley.write(":OUTP ON") #output is turned on
```

Then, with the specified step size, voltage is looped (for loop) between the initial and final voltage ranges. These values are appended in 2 empty lists.

```
#for loop
Voltage=[]
Current = []
for V in np.linspace(start,stop,num=int(NUMBER_POINTS), endpoint=True):
    keithley.write(":SOUR:VOLT " + str(V))
    time.sleep(0.1)
    data = keithley.query(":READ?")
    answer = data.split(',')
    I = eval( answer.pop(1) )*1e3
    Current.append( I )
    V = eval( answer.pop(0) )
    Voltage.append(V)
```

Now, measurement values are added to those previous empty lists, filled into an array and printed into csv format.

then they are plotted.

```
x=np.array(Voltage)
y=np.array(Current)
z = np.array([x,y])
z = z.transpose()
comp_name = input('Enter Component Name :: ')
filename = f'I_V_Observation_{comp_name}.csv'

np.savetxt(filename,z, delimiter=',', header='V,C'])
```

```
keithley.write(":OUTP OFF")
keithley.write("SYSTEM:KEY 23") # go to local control
keithley.close()

plt.plot(Voltage,Current)
plt.title("Diode I–V Characteristics")
plt.xlabel("voltage ( V )",color='green')
plt.ylabel("Current ( mA )",color='green')
plt.show()
```

5 Experiment and Observations

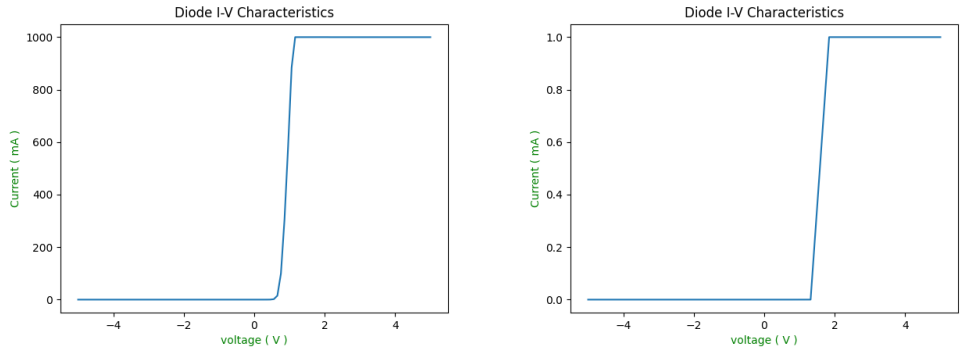
In the experiments we set the voltage scan range between -5 V to +5V and the current compliance to 1 mA for LED and 1 A for the silicon diode. The SMU will not apply a current value more than the current compliance limit.

Diode Characteristics

In IV characteristics of the diode is indicated in the following figures. Until a knee voltage value, the forward current is very low, upon crossing that limit the forward current increased rapidly (this is because the applied voltage overcomes the barrier potential of P-N junction). Knee voltage is measured around 0.53 V. When the applied voltage goes to negative direction, we observe very low current equal to the reverse saturation limit.

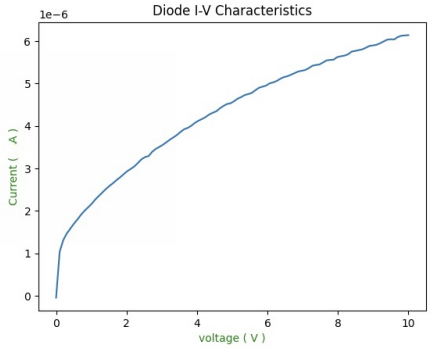
The next device we tested was an LED. Here the P and N regions of the diode are connected to the HI and LO of the SMU. The voltage in the range [-5V, 5V] is applied across the terminal. It showed similar result to that of Si diode, i.e. until the voltage reached the knee voltage, the current is zero, upon reaching the knee voltage the holes will get enough barrier potential and crosses the depletion region and mixes with electrons. The energy thus formed is radiated as light.

In the figure(c) Si diode is connected in reverse biased conditions. On increasing the reverse voltage only a slight increase in current is seen (in micro ampere).



(a)

(b)



(c)

Figure 6: Diode IV characteristics, (a) Si diode (current compliance is set to 1 A, minimum voltage= 0 V , maximum voltage = 5 V), (b) LED (current compliance is set to 1 mA, minimum voltage = -5 V, maximum voltage = 5 V), (c) Reverse biased Si diode, current compliance is set to 1 mA, minimum current = 0 V, maximum current = 10 V)

6 Result and Discussion

In this project, we developed a python based open source code for automating Keithley's source measure unit, a very commonly used lab equipment. Using this we can command and control the operations of SMU via a desktop and perform IV characteristics of a Si diode and an LED. This code can be changed and personalised according to our requirements. Further improvements of this basic software are necessary and we are working towards building graphical user interface to this program.

**

A Appendix

The Final Code

```
#importing all the necessary python library
import pyvisa
import numpy as np
import matplotlib.pyplot as plt
import time
import os
import csv

#establishing connection with keithley 2400 ( 26 th channel)
rm = pyvisa.ResourceManager()
instr = rm.list_resources()
keithley = rm.open_resource('GPIB0::26::INSTR')

#CONFIG INSTRUMENT
keithley.write("*RST") #resetting GPIB
time.sleep(0.5)
keithley.write(":ROUT:TERM FRON") #terminal set to front
keithley.write(":SOUR:FUNC:MODE VOLT") # source is set to voltage
keithley.write(":SENS:CURR:PROT:LEV " + str(CurrentCompliance))
keithley.write(":SENS:CURR:RANGE:AUTO 1")   #compliance range

#compliance
PROTECTION = input("Enter Compliance Value : ")
PROTECTION =float(PROTECTION)
PROTECTION_FUNCTION = ":SENS:CURR:PROT "+str(PROTECTION)
keithley.write(PROTECTION_FUNCTION)
#CurrentCompliance = PROTECTION

#initial voltage/ starting voltage
INITIAL_VOLTAGE = input("Enter Initial Voltage : ")
INITIAL_VOLTAGE =float(INITIAL_VOLTAGE)
INITIAL_VOLTAGE_FUNCTION = ":SOUR:VOLT:LEV "+str(INITIAL_VOLTAGE)
keithley.write(INITIAL_VOLTAGE_FUNCTION)
#maximum voltage
MAXIMUM_VOLTAGE = input("Enter Maximum Voltage : ")
MAXIMUM_VOLTAGE =float(MAXIMUM_VOLTAGE)
VOLTAGE_RANGE_FUNCTION = ":SOUR:VOLT:RANG "+str(MAXIMUM_VOLTAGE)

start = INITIAL_VOLTAGE
```

```
stop = MAXIMUM_VOLTAGE
NUMBER_POINTS = input( "Enter Number of Points : " )

keithley.write(VOLTAGE_RANGE_FUNCTION)

keithley.write(":OUTP ON") #output is turned on
#for loop
Voltage=[]
Current = []
for V in np.linspace(start, stop, num=int(NUMBER_POINTS), endpoint=True):
    keithley.write(":SOUR:VOLT " + str(V))
    time.sleep(0.1)
    data = keithley.query(":READ?")
    answer = data.split(',')
    I = eval( answer.pop(1) )*1e3
    Current.append( I )
    V = eval( answer.pop(0) )
    Voltage.append(V)

x=np.array(Voltage)
y=np.array(Current)
z = np.array([x,y])
z = z.transpose()

comp_name = input('Enter Component Name :: ')
filename = f'I_V_Observation_{comp_name}.csv'
np.savetxt(filename,z, delimiter=',', header='V,C'])
keithley.write(":OUTP OFF")
keithley.write("SYSTEM:KEY 23") # go to local control
keithley.close()
plt.plot(Voltage,Current)
plt.title("Diode I-V Characteristics")
plt.xlabel("voltage ( V )",color='green')
plt.ylabel("Current ( mA )",color='green')
plt.show()
```

References

(1) Keithley Series 2400 SourceMeter User's Manual, https://download.tek.com/manual/2400S-900-01_K-Sep2011_User.pdf.

(2) Instruments, N. LabVIEW Documentation, https://www.ni.com/docs/en-US/bundle/labview/page/lvhelp/labview_help.html.

(3) Electronics notes what is GPIB, https://www.electronics-notes.com/articles/test-methods/gpib-ieee-488-bus/what-is-gpib-ieee488.php.

(4) DataFlair Python Libraries, https://data-flair.training/blogs/python-libraries.

(5) Authors, P. PyVISA: Control your instruments with Python, https://pyvisa.readthedocs.io/en/latest/#pyvisa-control-your-instruments-with-python.

(6) Keithley tektronicx, t. IEEE-488 (GPIB) Interface Solutions, https://www.farnell.com/datasheets/1760368.pdf?_ga=2.78872785.450713776.1499842582-1530823304.1499842582.

(7) Mueller, J. E. In *6th IEEE Conference Record., Instrumentation and Measurement Technology Conference*, 1989, pp 66–70.