Article **Journal of Computational Systems and ICTs**
June 2020 Vol.6 No.17 23-29

# Introduction to the automatization of measurement equipment with Python-GPIB

# Introducción a la automatización de equipos de medición con Python-GPIB

MEDINA-BRISEÑO, Pablo*†, MOLINAR-SOLIS, Jesus Ezequiel, CHAVEZ-VELARDE, Juan Jose and BRACAMONTES-DEL TORO, Humberto

*Instituto Tecnológico de Ciudad Guzmán, Posgrado de Electrónica. Avenida Tecnológico #100 Mpio. de, 49100 Jal.*

ID 1st Author: *Pablo, Medina-Briseño*

ID 1st Coauthor: *Jesus Ezequiel, Molinar-Solis*

ID 2nd Coauthor: *Juan Jose, Chavez-Velarde*

ID 3rd Coauthor: *Humberto, Bracamontes-Del Toro*

**DOI:** 10.35429/JCSI.2020.17.7.23.29 Received February 01, 2020; Accepted June 10, 2020

**Abstract**

This article provides a quick introduction to the automation of measurements using a public domain language such as Python. We bring references for the used libraries and the basic commands for manipulating the GPIB port. Some examples are provided.

**GPIB-USB, automatic measurement system, GPIB port, Python**

**Resumen**

Este artículo brinda una rápida introducción para la automatización de mediciones utilizando un lenguaje de dominio público como es Python. Se hace referencia a las librerías utilizadas y los comandos básicos para la manipulación del puerto GPIB. Se brindan algunos ejemplos de uso.

**GPIB-USB, Mediciones automaticas, Puerto GPIB, Python**

**Citation**: MEDINA-BRISEÑO, Pablo, MOLINAR-SOLIS, Jesus Ezequiel, CHAVEZ-VELARDE, Juan Jose and BRACAMONTES-DEL TORO, Humberto. Introduction to the automatization of measurement equipment with Python-GPIB. Journal of Computational Systems and ICTs. 2020. 6-17: 23-29.

* Author Correspondence (E-Mail: pablobrimed@gmail.com)
† Researcher contributing as first Author.

 

## Introdution

In electronic characterization and research laboratories, it is very common to require automation of measurement and test equipment, such as; multimeters, signal generating sources, oscilloscopes, among others. In such a way that, in order to obtain these data, the user can work easily and quickly with electronic equipment.

The GPIB [1] or General Purpose Interface Bus, (Figure1 bus), supports the communication standard: IEEE 488.1, IEEE 488.2, which allows you to connect up to 15 devices and be able to control them with your bus of communication through the PC, Figure 2. There are commercial cards that allow the connection between the PC's USB port and the GPIB bus, such as the Keysight 82377B and ADLINK 3488A card. However, this work uses the PROLOGIX GPIB-USB card as it represents one of the cheapest on the market.

The Python programming environment facilitates automation because it is intuitive programming and it has many libraries that make it a versatile and very useful environment, in addition to being free [2], it is possible to use other languages, such as C, C ++, Visual BASIC, Labview among others, but Python offers many advantages over the previous ones, ease, versatility, friendly and free environment

The SCPI (Standard Commands for Programmable Instrumentation), are standard commands for the programming of the instruments, the advantage is that they replace the programming drivers that each instrument had, in such a way that several of them had to be installed. That is why they were adopted by the most recognized and important brands in the manufacture of measuring equipment such as:

- Agilent: Agilent Technologies
- Cec: Capital Equipment Corporation
- Iotech: IOTech hardware.
- Keithley: Keithley
- cc: Measurement Computing Corporation
- Ni: National Instruments.

These commands are important, since they are the ones that manipulate the instruments in particular. They comply with a coding standard and such commands are different between brands.


*a)* *b)*

**Figure 1** GPIB bus, a) GPIB connector available in measurement equipment. b) PROLOGIX business card – USB

**Figure 2** GPIB cable, female / male at both terminals

## Python installation and library for GPIB

Python is a language that has gained territory in the programming world, it is intuitive, easy and dynamic. The Python community has managed to make a series of libraries, in such a way that it becomes a very complete language. In this work, it is important to install some of them, with which we will achieve the connection between instruments.

The Python IDLE (Integrated DeveLopment Environment for Python) is a graphical environment for elemental development, it allows you to edit and run programs. For its installation we have to go to the official Python page [3]. The latest version is 3.7.4 for 64-bit Windows [4] or choose for 32-bit Windows [5]. The Python “pip” or Preferred Installation Program [6] is a tool that allows you to install, reinstall, or uninstall PyPI packages.

ISSN-2444-5002

This program executes the installation commands for all the libraries. If it has one, it can be updated by executing the command "-m pip install –upgrade pip", from the Windows command interpreter (CDM) or command prompt, Figure 3.

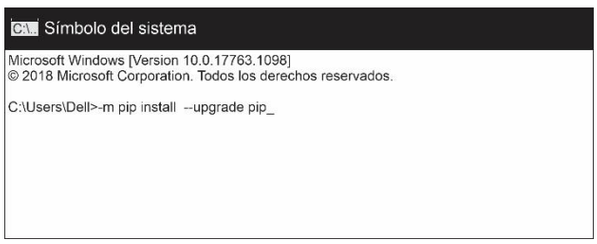


**Figure 3** Python pip update

It is essential to Install the necessary Python libraries. These will allow us to communicate with the instrument, so that we can operate and control it. Such libraries are:

| | |
|---|---|
| Pyvisa | [7] |
| Pyvisa-py | [7] |
| Pymeasure | [8] |
| pyUSB | [9] |
| pyserial | [10] |
| pyinstruments | [11] |

Libraries are easily installed from CDM, with the command pip install Figure 4.

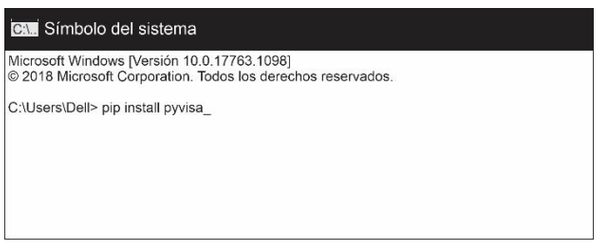


**Figure 4** Command to install pyvisa

We list below the commands to install the libraries using pip.

Pip install pyvisa-py

Pip install pymeasure

Pip install pyUSB

Pip install pyserial

Pip install pyinstruments

To do the first communication tests, the instruments must be connected, there are two ways to connect the equipment, star and serial. In Figure 5, reference is made to the serial connection with the device to be measured or characterized (DUT).

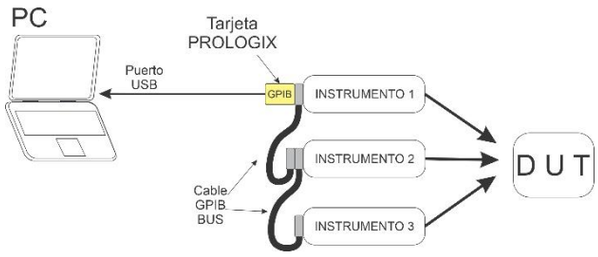


**Figure 5** Conventional way of connecting equipment to the device to be measured

## Installation of prologix gpib configurator

The Prologix GPIB Configurator application [12] is a user interface to test the communication of the CPU with the equipment, Figure 6. This can execute the SCPI commands of the instruments directly, which is recommended for preliminary tests. The interface shows us crucial data such as the "COM" port number of the PC to which the equipment was connected (shaded area). It also shows the GPIB address, which is a decimal number that each instrument has assigned to establish communication. In this way, this instruction "++ addr" must be executed followed by the address number every time an instruction is given to a particular computer. Such address in some cases is displayed when the instrument is turned on and others can be programmed from the options menu of the same equipment. In this document we will use a 6 1/2 digit Agilent 34410A equipment, which shows us the number 5 as the address.

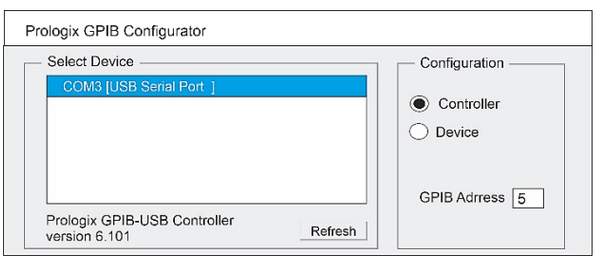


**Figure 6** Prologix GPIB Configurator, user interface

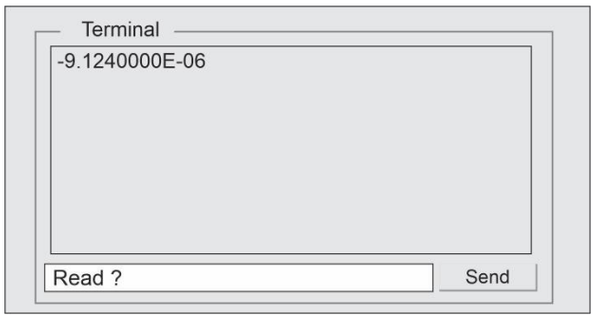


**Figure 7** Prologix GPIB Configurator Terminal

At this point we are ready to test the SCPI commands of the measuring instruments, in the dialog box of Figure 7, we write the command READ ?. This is a SCPI command, whose function is to read the data displayed on the multimeter screen. When sending the instruction, it returns the reading shown on the instrument's display, this reading is displayed in the "Terminal" part, in this case the data 9.12400000E-06. In this way, it is possible to verify the communication between the PC, the GPIB card and the instrument.

It is important to know the commands of the PROLOGIX GPIB – USB card [13]. These commands are particular, like: ++ addr; ++ auto; ++ clr; ++ read, among others. They have important functions, which help define your settings.

**SCPI commands**

The SCPI language is based on the IEEE 488.2 standard, which is defined for the commands and their syntax. The main advantage of SCPI is that it reduces the time used to create an application, which allows controlling the different computers. Since prior to its appearance, the programmers directly used the drivers of the measuring instruments. SCPIs have two types of commands, these are: Common commands, those that come preceded with an "*" as an example, * CLS; * OPC; * RST; * IDN ?, among others. And the specific commands, those that use the instruments in particular for their operation, automation and control. The latter depend on the purpose and characteristics of each team, they are made up of 3 to 4 letters and are combined with others to perform specific actions. Example:

- MEASure:VOLTage:DC?
- TRIGger:MODe NORMal
- CONFigure:VOLTage:DC (@2)

These commands are explained in the user manuals of each instrument, where some even have programming examples.

**Establishing Python and gpib communication**

To establish communication between the PROLOGIX GPIB-USB and the equipment using Python, the following steps are listed:

1.- The PROLOGIX GPIB-USB card, the measuring instrument and the PC are connected.

2.- In the Python IDLE the visa library is imported >>> import visa

3.- Establish a variable "rm" that calls the communication resources and prints them. >>> rm = visa.ResourceManager () >>> print (rm.list_resources ())

For this case, the result it returns is as follows:

***('ASRLCOM3::INSTR',)***

With the complete command ('ASRLCOM3 :: INSTR',) to establish communication between the program and the instrument, we start with the coding of the program.

The following code is an example that allows us to read the voltage values of a multimeter.

```
P1  import visa
    rm=visa.ResourceManager()
    rm.list_resources()
P2  inst=rm.open_resource('ASRLCOM3::I
    NSTR')
P3  print(inst.write("++addr 5"))
P4  print(inst.write("CONF:VOLT:DC 10,
    0.003"))
    for i in range(5):
P5  print(inst.query("READ?"))
    time.sleep(1)
    print()
P6  rm.close()
```

P1.- Visa libraries are imported to use all their resources.

P2.- A “inst” variable is created that keeps the communication command open ('ASRLCOM3 :: INSTR').

P3.- The input / output of the instrument is addressed “++ addr” followed by the number assigned to the equipment, in this case 5.

Q4.-Using the corresponding SCPI command, the equipment is configured to measure voltage, direct current, maximum value 10V. with 0.003V resolution.

Q5.- The measured value is requested; it prints it in the Python Shell.

P6.- Close the communication port.

**Programming of several equipment**

To program several devices with the PROLOGIX GPIB-USB, it must be connected through the communication cables as shown in Figure 2, and Figure 6. It is necessary to remember that each instrument will have a different address.

| | |
|---|---|
| P1 | import visa |
| | import time |
| | Import matplotlib.pyplot as plt |
| | Import numpy |
| | Import pandas as pd |
| | rm=visa.ResourceManager(‘@py’) |
| | rm.list_resources() |
| P2 | inst=rm.open_resource ('ASRLCOM3::INSTR') |
| P3 | vin2=0 |
| P4 | vfin2=3 |
| P5 | pasos=10 |
| P6 | var1=(vfin2-vin2)/pasos |
| P7 | x=[] |
| P8 | y=[] |
| P9 | print(inst.write("++auto 1")) |
| P10 | print(inst.write("++read")) |
| P11 | archivo=open(“mediciones.csv”,”w”) |
| P12 | for i in range (pasos): |
| P13 | v2=vin2+i*var1 |
| P14 | print(inst.write("++addr 4")) |
| P15 | print(inst.write("OUTP ON")) |
| P16 | print(inst.write("CURR .125")) |
| P17 | print(inst.write('SOUR:VOLT {}'.format(v2))) |
| P18 | print("voltaje"+str(v2)) |
| P19 | x.append(v2) |
| P20 | time.sleep(1) |
| P21 | print(inst.write("++addr 5")) |
| P22 | print(inst.write("CONF:CURR:DC AUTO")) |
| P23 | num=(inst.query('READ? ' ) |
| P24 | print("medicion"+str(num)) |
| P25 | y.append(num) |
| P26 | archivo.write(" , "+str(num)+str(v2)) |
| P27 | print(inst.write("++addr 4")) |
| P28 | print(inst.write("OUTP OFF")) |
| P29 | rm.close() |
| P30 | archivo.close() |
| P31 | plt.xlabel('BARRIDO DE VOLTAJE. -0V. A 2.0V.') |
| P32 | plt.ylabel('CORRIENTE MEDIDA -DIODO-') |
| P33 | plt.plot(x,y,".") |
| P34 | plt.grid(True) |
| P35 | plt.show() |

**Table 1**

P1.- The following libraries are imported:

Visa: To establish communication with the instrument.

Time: With the function time.sleep (). We handle waiting times in the execution of the program.

Mathplotlib: Your resources help us generate the graphs with which we visualize the measurements.

Numpy: Fundamental parcel for handling scientific data.

Pandas: with this library we can manipulate number tables and time series.

P2.- A “inst” variable is created that keeps the communication command open ('ASRLCOM3 :: INSTR').

P3.- Variables are declared to be able to calculate the values for the DC voltage sweep, which is from 0V to 3V. The variable "vin2" starts the sweep at 0V.

P4.- This variable vfin2, defines until which value the sweep ends, which in this case is at 3V.

P5.- The steps is the number of voltage increments for the sweep, here the voltage interval is divided into 10 steps.

Q6.- The formula “var1 = (vfin2-vin2) / steps” calculates the value of the voltage increments that the instrument will put on its terminals.

ISSN-2444-5002

Q7.- “x” is given a list of empty spaces to be filled with assigned data, in this case, the 0V voltage sweep. at 3v.

Q8.- “y” is given a list of empty spaces to be filled with assigned data, in this case, the measurement values.

Q9.- The command "++ auto 1", enables or disables the "read-after-write". This enables reading to be done after sending an instruction.

P10.- The “++ read” command allows us to read data from the instrument in use.

P11.- In this programming line, a variable called “file” is declared, it contains a command called open, and within this, the creation of a csv file is declared, which will be named measurements, the “w” stands for write, that is to say; We create a measurements.csv file, which we open to write to it.

P12.- The for cycle makes a series of iterations on the assigned variable “i”, and iterates the number of times the variable steps has been designated. For this example, there are 10 iterations. Once the iterations are finished, the cycle stops.

P13.- In the formula, v2 = vin2 + i * var1, the variable v2, receives a value, and changes according to var1, which marks the increase in voltage.

P14.- Communication is opened with the instrument that has address 4 "++ addr 4", in our case it is a voltage source AGILENT E3645A.

P15.- This SCPI command causes the source output to be enabled.

Q16.- We assign a current limit of 0.125A.

P17.- The instrument is asked to put the voltage value of the variable v2 at its terminals.

P18.- Converts the value of the variable "v2" to a data of string form and prints it in the Python Shell. That is the voltage value that the instrument displays on its screen.

P19.- This command causes the values generated by the cycle for the sweep to be added one by one to the variable x, in the empty list defined in P7.

Q20.- A pause of one second of time is assigned.

P21.- Communication is established with the instrument that has address 5 “++ addr 5”, in this case it is an AGILENT 34401A multimeter.

P22.- The multimeter is configured to measure DC current, in auto scale.

P23.- The variable “num” is declared. This variable obtains the value of the measurement that is requested from the instrument using the “READ?” Command.

P24.- The measured value of the variable "num" is converted into a string value which is printed in the Python Shell.

P25.- This command causes the values measured by the instrument to be added one by one to the variable “y”, in the empty list that was defined in P8.

P26.- Write the data acquired by the variable "num" and "v2" in the file measured.csv that was declared in P11, write it in different columns.

P27.- Communication is opened to the address ++ addr 4 which is the voltage source.

P28.- The source is asked to be disabled after finishing the cycle. This in order that no voltage value remains on the device at the end of the measurement.

Q29.- the variable “rm” declared to open the port, is asked to close the communication port.

P30.- The file “measurements.csv” that was opened to write data in P11 is closed.

P31.- Plt.xlabel, allows us to put text on the x axis to denote references, in this case; VOLTAGE SWEEP. -0V. At 2.0V.

P32.- Plt.ylabel, allows us to put text on the y-axis to denote references, in this case; 'CURRENT MEASURE -DIODE-

Q33.- Python graphs the data stored in “x” and “y”, and it is asked that, for each corresponding value of x, and a point be put, to visualize the graph.

ISSN-2444-5002

P34.- A maya or grid is requested

**Storage of data and graphics**

Most of the time when making measurements, the data should be saved for analysis. There are different ways to save them, here are some options. Plain text file .txt, file .csv (comma separate values) comma separated values, and even the use of a database such as MySQL or ACCES among others.

Generating a graph of measurements with Python is possible using the matplotlib [14], and numpy [15] libraries. With these libraries you can make graphs of the measurements for the visualization of the data.

**Conclusions**

Python's versatility makes it a powerful and freely accessible tool that in this case makes it easy to use the GPIB port for synchronization and manipulation of measuring instruments. This paper presents a quick introduction that serves as a start for those who want to make more complex measurements using various instruments.

**References**

[1]http://prologix.biz/?gclid=CjwKCAjw29vsBRAuEiwA9s0BzFFgwD7sBRmjJxGBIxVy9cNmP__uJhYyXmAddR9KypBsUuUzuuonRoCCOQQAvD_BwE

[2]https://github.com/pyvisa/pyvisa/blob/master/LICENSE

[3]https://www.python.org/.

[4]https://www.python.org/ftp/python/3.7.4/python-3.7.4-amd64.exe

[5]https://www.python.org/ftp/python/3.7.4/python-3.7.4.exe

[6] https://pip.pypa.io/en/stable/installing/#do-i-need-to-install-pip

[7] https://pyvisa.readthedocs.io/en/latest/

[8] https://pymeasure.readthedocs.io/en/latest/pymeasure

[9] https://pypi.org/project/pyusb/

[10] https://pypi.org/project/pyserial/

[11] https://pypi.org/project/pyinstruments/

[12]http://www.thegleam.com/ke5fx/gpib/readme.htm#prologix

[13]http://prologix.biz/downloads/PrologixGpibUsbManual-4.2.pdf

[14] https://matplotlib.org/

[15] https://numpy.org/