Syllabus of Data Acquisition & Instrumentation laboratories

In those sessions, you will be interfacing electronic instruments in order to take measurements and acquires data in various way. Three concepts are important :

1. Python is the language used to implement the solutions,
2. Visa is the standard API helping the communication
3. SCPI is the Standard command understood by most of the instruments.

Introduction

Python

Python is the programming language used as it's part of your studying path. Moreover, it's fully documented and has a good compatibility with other API to process and display the collected data.

Visa

To communicate with the laboratory's instrument, we will use Visa to help us. Virtual Instrument Software Architecture (Visa) is an Application Programming Interface (API). As an industry standard, it is possible to use the same API for all the top bench instruments such as Keysight Technologies, Rohde & Schwarz, Rigol and Tektronix.**?** Another advantage is the compatibility of the API with the different communication protocols such as USB, LAN-Ethernet, GPIB, RS232, ...

SCPI

*The Standard Commands for Programmable Instruments (SCPI; often pronounced "skippy") defines a standard for syntax and commands to use in controlling programmable test and measurement devices, such as automatic test equipment and electronic test equipment.***?**

Computer and Raspberry Pi setup

To be global, free and more into computer science, the use of Python is the most suitable approach to use VISA. MATLAB and LabVIEW, being commercial software, the access is restricted to a paying license. Industrial is more of the use of remote instrumentation or automated one, we will interface a Raspberry Pi (Unix system) or your computer to control the instruments. To do it, it is important to verify if both systems are updated (your computer or the raspberry)

To correctly configure your computer, we need to use an SSH connection to access remotely the Raspberry Pi, a good way to do it is by the mean of Putty [1] [2]

1. Link to Putty, https ://www.chiark.greenend.org.uk/ sgtatham/putty/latest.html
2. it's possible to forward the graphical view of the Raspberry Pi but it won't be of use and might damage the Raspberry Pi

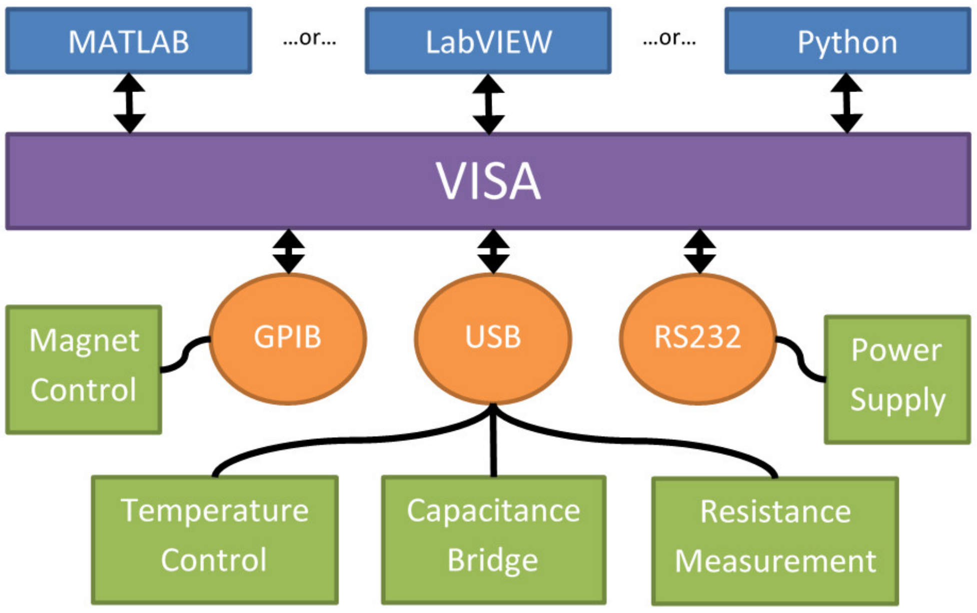


FIGURE 1 – Visual stack of interaction between VISA, the instruments and the program

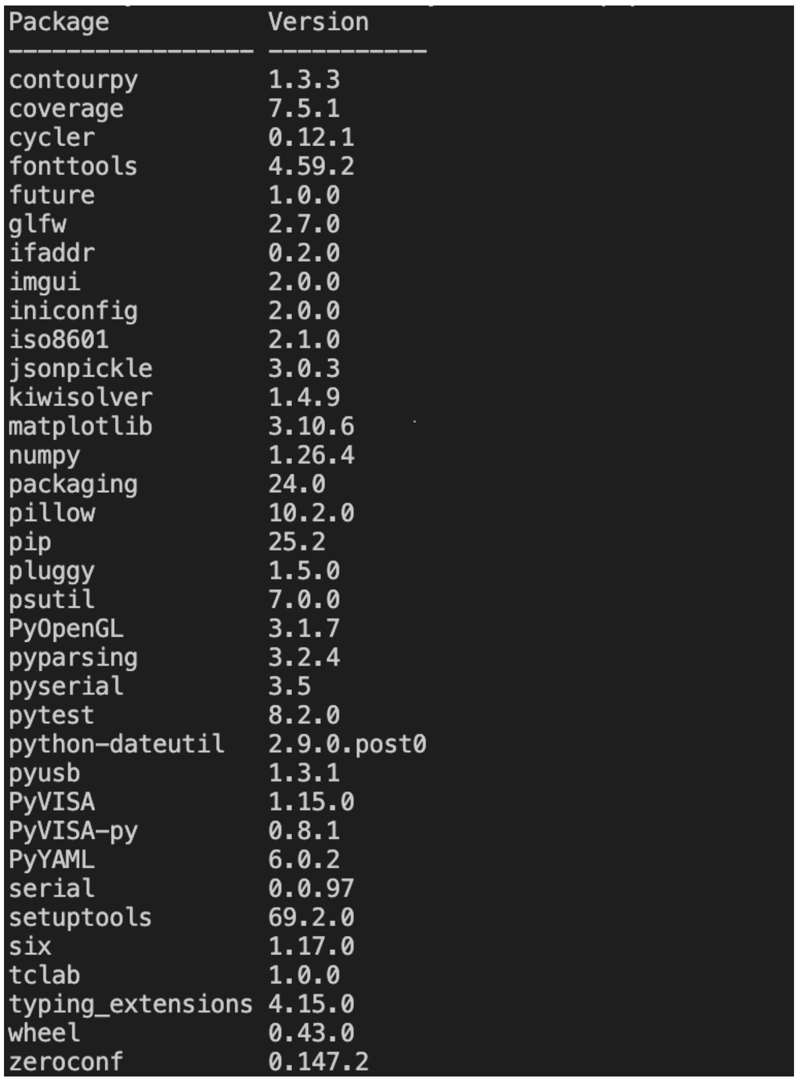
To configure your computer or the Raspberry Pi, be sure to have installed **Python 3** and the following packages :

— PyVisa-py (this is limited but it allows you to avoid installing NI-VISA packages with legal commitment)
— serial
— zeroconf
— pyserial
— Numpy
— MatplotLib
— pyusb
— psutil
— If your computer advises you of some extra packages, do not hesitate to install them
— usbutils (for archlinux or enable to use the lusb)

On your computer or the Raspberry Pi, it's recommended to always execute the code or the environment, respectively as admin or as superuser (ex. SUDO) to see the USB connection and interact with the instruments.

For Windows and Mac users, Ni-Visa must be installed : *https://www.ni.com/fr/support/downloads/drivers/download.ni-visa.html#570633*

Here is an example of all the packages installed on a MAC where the program works fine.



```
Package           Version
----------------- -----------
contourpy         1.3.3
coverage          7.5.1
cycler            0.12.1
fonttools         4.59.2
future            1.0.0
glfw              2.7.0
ifaddr            0.2.0
imgui             2.0.0
iniconfig         2.0.0
iso8601           2.1.0
jsonpickle        3.0.3
kiwisolver        1.4.9
matplotlib        3.10.6
numpy             1.26.4
packaging         24.0
pillow            10.2.0
pip               25.2
pluggy            1.5.0
psutil            7.0.0
PyOpenGL          3.1.7
pyparsing         3.2.4
pyserial          3.5
pytest            8.2.0
python-dateutil   2.9.0.post0
pyusb             1.3.1
PyVISA            1.15.0
PyVISA-py         0.8.1
PyYAML            6.0.2
serial            0.0.97
setuptools        69.2.0
six               1.17.0
tclab             1.0.0
typing_extensions 4.15.0
wheel             0.43.0
zeroconf          0.147.2
```

Figure 2 – Package list on a working MAC

Session 1 - the function generator - Rigol DG1022

The programming reference manual is available at this address : *https://beyondmeasure.rigoltech.com/acton/attachment/1579/f-0038/0/-/-/-/-/file.pdf*. Make sure your Rigol is connected to your usb.

1. As your instrument has a unique ID[3], let's find out what is its number with the following code. Notice that this code should always be used to start the Visa connection

   Excute :

   ```python
   import pyvisa
   rm = pyvisa.ResourceManager('@py')
   L=rm.list_resources()
   print(L)
   ```

   Answer of a DG1022 : `('USB0::0x1AB1::0x0588::DG1D114803911::INSTR')`

   Hint : sometimes the code won't synchronize correctly with the instrument, in order to give time to the instrument to update don't hesitate to add a delay with *time.sleep()*

2. To start create a simple code that will open and close Channel 1 ten times with a delay of one second between the steps (import time to help you). To communicate correctly, it is important to open the resource manager with the ID of your device and create your instrument in the code. To do this step, use the function `open_resources(ID)` on the variable of your resource manager and store it in a variable that will correspond to your instrument (example : `osci = rm.open_resource(ID)`). Do not forget to close the communication with your instrument and the resource manager after the execution of your code[4].

3. Challenge, make it automatic to connect one function generator and one oscilloscope with the right ID and variable. 1 bonus point out of 20 will be provided if you can do it before the last session, only to the first one to achieve it

4. For all the functions used by the resource manager, you can get more information on this web page *https://pyvisa.readthedocs.io/en/latest/api/resources.html*

3. based on the programming Manual, explore the functionalities in order to create a sine of 5V on channel 1 and a square of 3 volt on channel 2. To visualize the generated waves, you can use the oscilloscope and BNC wires in the lab.

4. On a breadboard, use a capacitor to flatten an exponential wave generated by the generator (find the right information in the manual). Watch your signal on the oscilloscope.

5. To be complete, create a square signal on channel 1 that has
— a frequency of 1250 Hz
— an offset of 2.3V
— a duty-cycle of 60%
— an amplitude changing every 2 seconds form 2.3V to 4.6V

6. Record all the information of exercise 5 in a CSV file with a timeline and show it with a graph

7. **Challenge :** Use a song as an arbitrary waveform and make it generated by the generator into the speaker. (ask the speaker to the professors)

Session 2 - oscilloscope

Exercise 1 - Connect to the oscilloscope

To start using the oscilloscope, let's copy some functionalities and try to understand them by executing[5] :

```python
import pyvisa

rm = pyvisa.ResourceManager()
Osci = rm.open_resource('your ressource')
# To select a single channel (1 or 2) permanently you can use this line
# it allows you to not write CHAN2 at every call
Osci.write(':MEAS:SOUR:CHAN2')
data = []
data.append(Osci.query(':MEAS:Item? VPP'))
data.append(Osci.query(':MEAS:Item? VAVG'))
data.append(Osci.query(':MEAS:Item? PER'))
data.append(Osci.query(':MEAS:SOUR?'))
data.append(Osci.query(':CHAN1:SCAL?'))
data.append(Osci.query(':FUNC:WREC:FEND?'))
Osci.write(':MATH:FFT:SOUR CHAN1')
Osci.write(':MATH:OPER FFT')
Osci.write(':MATH:DISP ON')
data.append(Osci.query(':LAN:VISA?'))
# Warning to Measure 'MAth' should be off
# either it might not work, try it yourself! do you get data 8?
Osci.write(':MATH:OPER FFT')
Osci.write(':MATH:DISP OFF')
#measure the frequency of the signal
Osci.write(':MEAS:COUN:SOUR CHAN1')
data.append(Osci.query(':MEAS:COUN:VAL?'))
print(data)
```

Exercise 2 - oscilloscope

1. Create a program that can interface the oscilloscope and plot V_{RMS} as a function of time.

   hints, to help you :

   (a) Use numpy

   (b) use Use MatplotLib[6]

5. *https://www.rigol.eu/Public/Uploads/uploadfile/files/ftp/DS/%E6%89%8B%E5%86%8C/DS1000Z/EN/DS1000Z_ProgrammingGuide_EN.pdf*

6. Some people may encounter an error of "non interactive plot", in this case, you can install pyQT5 (or higher) with the line pip install pyqt5

2. To measure an accurate value of the V_{RMS} take several values of it (10 then 100), plot all the value and the average of its. Don't forget to leave some time to the data to be stored (500ms per scan).

3. Let's start reducing the value of waiting time. Compute the minimum waiting time and with it the sampling rate of V_{RMS} on the oscilloscope.

Exercise 3 - full wave measurement

Try this code that allows you to read what you see on the oscilloscope. How does it perform ?

```
import pyvisa
import numpy as n
import matplotlib.pyplot as mp
#Create a resource manager
```

```python
resources = pyvisa.ResourceManager()
#Open the Rigol by name. (Change this to the string for your instrument)
osc = resources.open_resource('USB0::0x1AB1::0x04CE::DS1ZC212301108::INSTR')
#Return the Rigol's ID string to tell us it's there
print(osc.query('*IDN?'))

# reading procedure initialisation
osc.write(':WAVeform:FORMat ASCII') #easy to read information but slower
osc.write(':WAVeform:Mode NORMAL')   #see programming book
L1 = osc.query(':WAVeform:PREamble?') #to know everything
print(L1)
osc.write(':WAVform:SOURce Channel1')
D1 = osc.query(':WAVEFORM:DATA?')
D2 = D1.split(',')

Lgth = n.size(D2)

mp.plot(n.linspace(0,1,Lgth-1),D2[1:Lgth])
```

What code could help you get a better plot of your data ? Print D1 or D2, this might help you to find the answer. Hint : the ascii format is not convenient to plot your data , try to cast it into an usable one

Exercise 4 - phase measurement

Based on two identical sinusoidal signals but having a difference in phase, perform a delay and phase measurement. Does it correspond to the signal generated by the rigol generator ? To go further, you can generate the signal in Python with the DG1022.

Exercise 5 - Automated code

Create a generic code that generates a sinus with the DG1022 and measure it with the DS1104. The measured value must be plotted with Matplotlib, in an understandable voltage and time scale. The legend of the plot should have the frequency, VPP, Vmin and Vmax. At the same time, the values generated and measured have to be stored in a TXT or CSV file for later use.

Session 3 - Arduino and Power Supply

Exercise 1 - Arduino

In this session you will interface the Arduino with python and the serial protocol. Previously the instruments were using the VISA standard. However, the Arduino does not implement it natively with a set of functions. The official web site of Arduino gives you some hints on how to communicate with the Arduino Uno in serial[7] and with the help of python[8] :

Now you see how a simple code works. Create your own set of functions in the Arduino to call them and perform a measurement or an action when using the serial port. your set should have at least the following function available :

1. An analog Read
2. A Digital Read
3. A PWM write
4. An analog write
5. A digital Write
6. LED on/off
7. Blink LED13 (built in LED)

Exercise 2 - All Together !

Gather all your knowledge to validate some information from the datasheet of the LM741 (*https://www.ti.com/lit/ds/symlink/lm741.pdf*) :

- Supply the circuit with the right potential, try some different values within the possible range.
- Generate a sinus and a triangle signal as input with the help of the wave generator.
- Measure with the oscilloscope the input and output.
- Define a protocol of measure in order to make it repeatable.

Exercise 3 - All Together ! bis

Gather all your knowledge to reproduce the I-V curve of the N-mos transistor BS170. Its datasheet is available here (*https://docs.rs-online.com/7137/0900766b81538660.pdf*) :

- Polarize the circuit with the right potential, try some different values within the possible range.
- Generate a sinus and a triangle signal as input with the help of the wave generator.
- Measure with the oscilloscope the input and output.
- Define a protocol of measure in order to make it repeatable.
- Trace the graphic of the I-V curve and save the image

7. To use the serial communication, you can install pyserial (pip install pyserial)
8. *https://projecthub.arduino.cc/ansh2919/serial-communication-between-python-and-arduino-663756*

Challenge

(1 point, only available during the class period but open to everybody and after completion of exercises 2 and 3)

Control the Arduino with your computer and measure the value of the temperature[9] given by the LM35D on an analog pin, display the temperature on a homemade LED VU meter. As the value is not straight forward perform the adjustment, in both way, with python on the raw data and with the Arduino before sending the computed data with a new function. Can you measure it with the oscilloscope at the same time using Pyvisa ? If yes, the three values should be display on the same "continuous" time graphic with matplotlib and stored in a text file. Is there any differences between the curves ? Is there any error ?

Challenge bis - Power Supply

If you finished the challenge, you can ask to start using the power supply for 20 minutes (ask the professor to access the power supply). Here are some functionalities and try to understand them by executing them[10] :

How does it work ?

```
import pyvisa

rm = pyvisa.ResourceManager()
L=rm.list_resources()
LL = '{}'.format(*L)
print(LL)
IV = rm.open_resource(LL)



IV.write(':OUTP:OVP:CLEar ch2') #bonne pratique
IV.write(':OUTP:OVP:VAL ch2,2.2')
#IV.write(':OUTP:OVP: CH2,on')  #marche pas
OVP = IV.query(':OUTP:OVP:val? ch2')
print('OVP3: '+OVP)
OUT = IV.query(':OUTP? ALL')
print('CH1 (ON/OFF),CH2,CH3 :' + OUT)
OUT = IV.query(':OUTP? ch2')
print("CH2 :" + OUT)
IV.write(':OUTP ch2,ON')
IV.write(':OUTP ON')
IV.write(':APPL 2.1,0.3,up')   #apply potential , then current with  precision of 0.1A
VOLT1 = IV.query(':MEAS? CH1')
```

9. Heat the sensor with your hand or cool it down by opening the window.
10. *https://download.rigol.com/en/Manual/DC%20Power%26Loads/DP900/DP900_ProgrammingGuide_en.pdf*

```
print('Volt CH1 :'+ VOLT1 +' V')
AMP1 = IV.query(':MEAS:CURR? CH1')
print('AMP CH1 :'+ AMP1 +' A')
POW1 = IV.query(':MEAS:POWE? CH1')
print('POW CH1 :'+ POW1 +' W')

ALL1 = IV.query(':MEAS:ALL? CH1')
print('ALL CH1 :'+ ALL1 +' V,A,W')

IV.write(':OUTP CH1,OFF')
IV.write('INST CH2')
IV.write(':APPL 2.1,0.3,up')
```