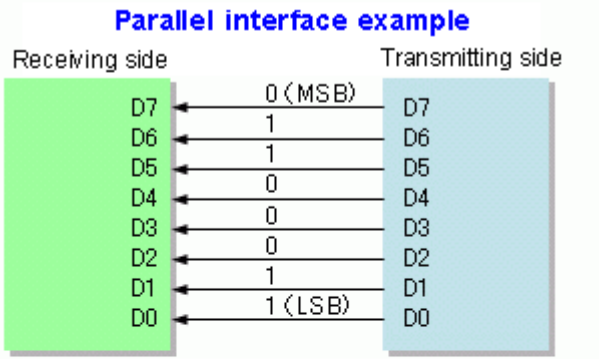
Laboratorio 5, Verano 2022

Cómo comunicarse con un instrumento con Python

Martín Drechsler

(basado en clase de Nico N.)

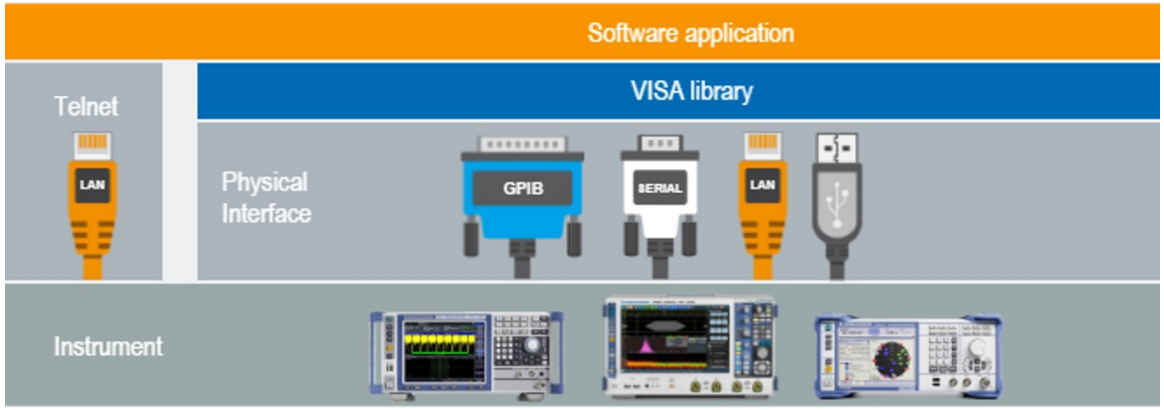
Tipos de comunicación



Interface	GPIB	LAN	USB	RS232
				
Bandwidth	1.8Mbit/s (488.1)	12.5 Mbit/s 125 Mbits/s (GB LAN)	60 Mbit/s (Hi-Speed) 120 Mbit/s (USB 3.0)	28.8 kB/s
Latency (informative) *Note1	300 µs	250 µs	1000 µs (USB) 125 µs (Hi-Speed)	~100 ms

Ej: Lock-In SR830: GPIB

Comunicándonos con Python



PyVISA

~~VISA~~

Virtual **I**nstrument
Software **A**rchitecture

Usando el paquete **PyVisa** (https://pyvisa.readthedocs.io/en/latest/) podremos controlar casi todo lo que querramos

Otros paquetes más específicos: *PySerial*, *PyUSB*, *linux-GPIB*...

Ejemplo en Python en MUY pocas líneas

```
In [1]: import pyvisa

In [2]: rm = pyvisa.ResourceManager()

In [3]: rm.list_resources()

Out[3]: ('USB0::0x0699::0x0368::C033910::INSTR',
         'ASRL3::INSTR',
         'ASRL4::INSTR',
         'ASRL5::INSTR',
         'ASRL6::INSTR')

In [4]: osc = rm.open_resource('USB0::0x0699::0x0368::C033910::INSTR')
```

osc es un **objeto** (o pueden pensarlo como una variable) que representa a la comunicación con el instrumento

¿Qué comandos entiende VISA?

write: le enviamos un mensaje al instrumento

```
In [5]: osc.write('hola')
Out[5]: 6
```

read: si el instrumento mandó un mensaje, lo leemos
(si no mandó nada, obtendremos un timeout)

```
In [6]: osc.read()
```

¿Qué comandos entiende VISA?

write: le enviamos un mensaje al instrumento

```
In [5]: osc.write('hola')
Out[5]: 6
```

read: si el instrumento mandó un mensaje, lo leemos
(si no mandó nada, obtendremos un timeout)

```
In [6]: osc.read()
VisaIOError: VI_ERROR_TMO (-1073807339): Timeout expired before
operation completed.
```

¿Qué comandos entiende VISA?

write: le enviamos un mensaje al instrumento

```
In [5]: osc.write('hola')
Out[5]: 6
```

read: si el instrumento mandó un mensaje, lo leemos
(si no mandó nada, obtendremos un timeout)

```
In [6]: osc.read()

VisaIOError: VI_ERROR_TMO (-1073807339): Timeout expired before
operation completed.
```

query: enviar mensaje y leer respuesta luego

```
In [7]: osc.query('*IDN?')
Out[7]: 'TEKTRONIX,TBS 1052B,C033910,CF:91.1CT FV:v4.06\n'
```

¿Qué comandos entiende VISA?

Leyendo valores

- **query**:

```
In [16]: osc.query_ascii_values('HOR:SCA?')
Out[16]: [5e-06]
```

Dependiendo la configuración del instrumento, podemos especificar el encoding:

- **query_ascii_values**
- **query_binary_values**

¿Qué comandos entiende el instrumento?

Leer manual!

Extra:¿Qué son las clases?

Clases y objetos

```
class Auto:
```

Clase

Auto

```
In [40]: type(Auto)
Out[40]: type
```

Objeto/instancia de la clase

AutoNico

```
In [41]: AutoNico = Auto(4, 'Azul')

In [42]: type(AutoNico)
Out[42]: __main__.Auto
```

Atributos

- Color
- Ruedas
- Marca
- Precio

```
In [43]: AutoNico.ruedas
Out[43]: 4
```

Métodos/funciones

- Decir la hora
- Pintar el auto
- Agregar una rueda
- Hablarle a otro auto
- Prender/apagar
-

```
In [44]: AutoNico.SetRuedas(5)

In [45]: AutoNico.GetRuedas()
Out[45]: 5
```

Una clase es un nuevo tipo de objetos (int, float, bool) cuyas propiedades las definimos nosotros

Ejemplo (¿muy?) abstracto

Definiendo una clase

```
10  class ClasePrueba:
11
12      def __init__(self, input1, input2):
13          self.atributo1 = input1
14          self.atributo2 = input2
15
16      def metodo1(self, inputmetodo1):
17          self.atributo1 = self.atributo1 + inputmetodo1
18          return self.atributo1
19
20      def metodo2(self):
21          print('Hola')
```

Instanciando (creando un objeto de) una clase

```
In [52]: objetoprueba = ClasePrueba(1, 2)

In [53]: objetoprueba.atributo1
Out[53]: 1

In [54]: objetoprueba.metodo1(3)
Out[54]: 4

In [55]: objetoprueba.metodo2()
Hola
```

Obligatorio:
método `__init__(self, argumentos)`
para inicializar el objeto de la clase y asignarle propiedades o ejecutar funciones