Adquisición, Control y Procesado

5.1 Instrumentación básica

5.2 Sistemas de interfaz con computador: tarjetas DAQ

5.3 Sistemas de interfaz con computador: buses de instrumentación

5.4 Herramientas software de control y procesado

Bloque V: Adquisición, Control y Procesado

Introducción

Tarjetas de adquisición de datos

Instrumentos controlados por computador

Herramientas software

Adquisición, Control y Procesado

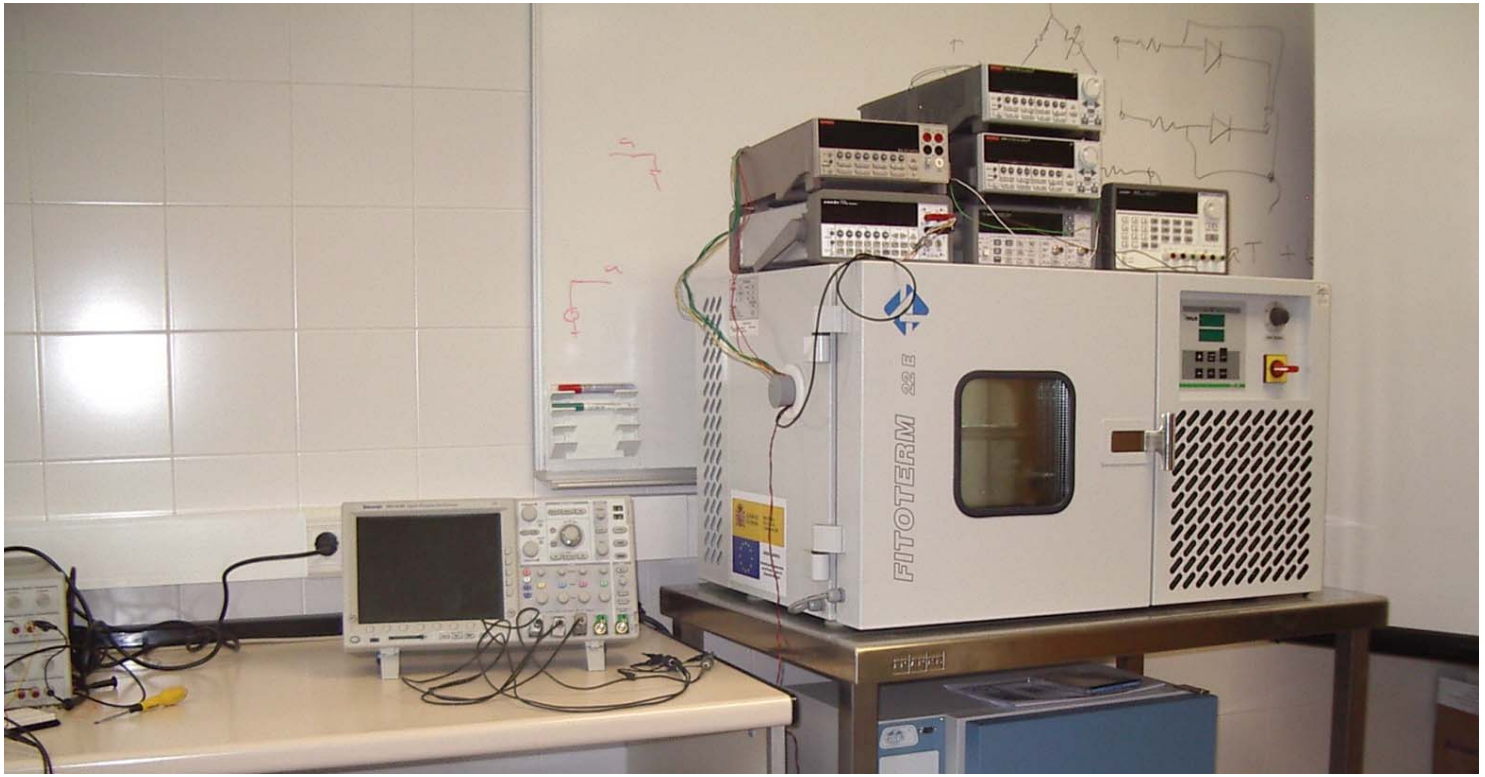
5.1 Instrumentación básica

5.2 Sistemas de interfaz con computador: tarjetas DAQ

5.3 Sistemas de interfaz con computador: buses de instrumentación

5.4 Herramientas software de control y procesado

5.1. Instrumentación básica

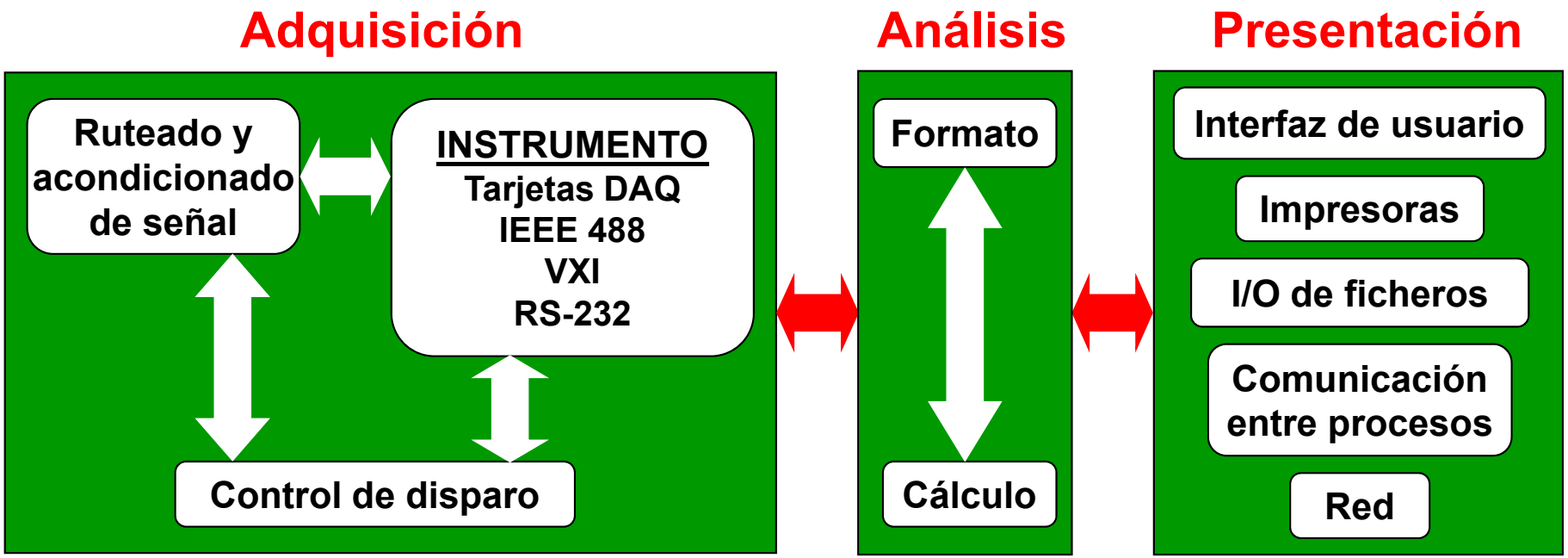


5.1.1 Introducción

Instrumentación inteligente

Dispone de un procesador capaz de realizar determinados procesados de datos y generar señales de control

Modelo de un sistema de instrumentación inteligente:
Standard Architecture for Measurement for Instrumentation (SAMI):



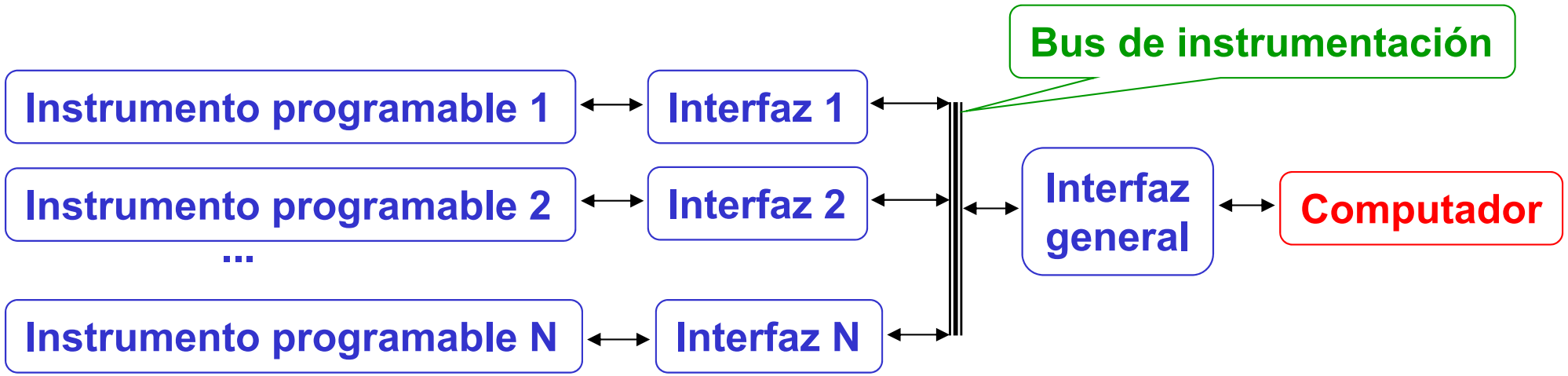
5.1.1 Introducción

Durante el procesado puede ser necesario modificar las operaciones:

PROGRAMABILIDAD

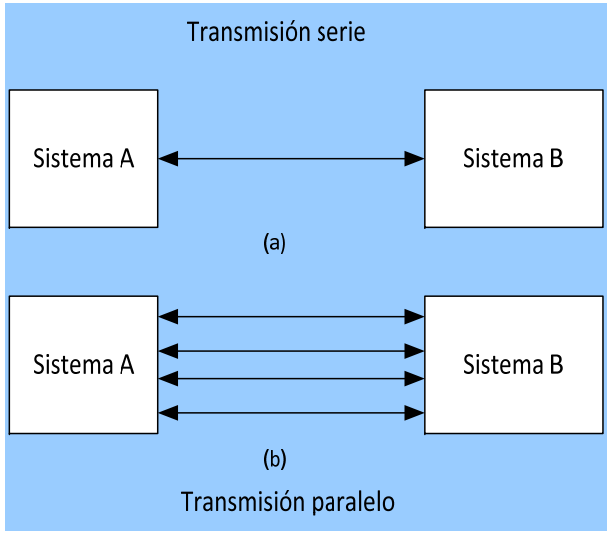
Interés por la posibilidad de modificar el programa desde un computador

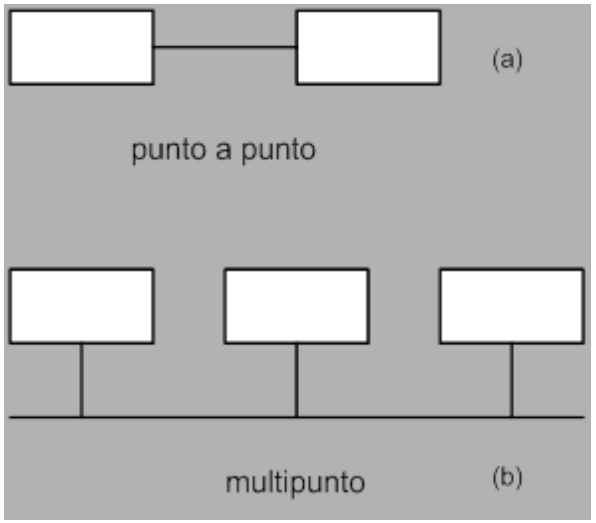
Interfaz de comunicaciones

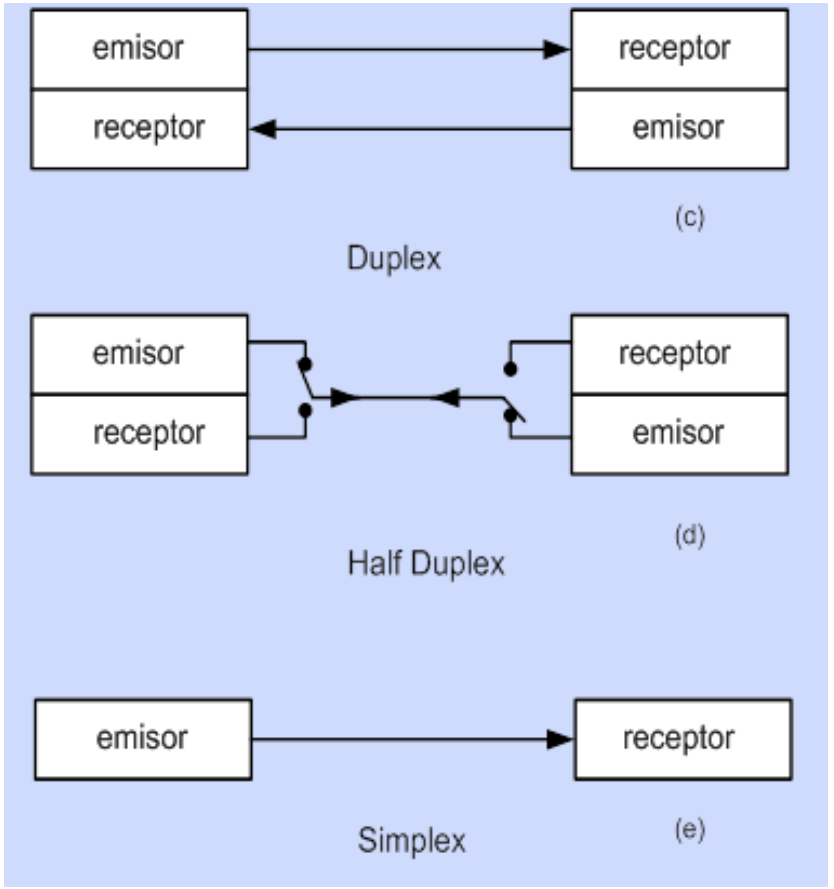


5.1.2 Transmisión de datos

Tipos de transmisión de datos







Full Duplex (USA) = Duplex

Half Duplex (USA) = Simplex

Adquisición, Control y Procesado

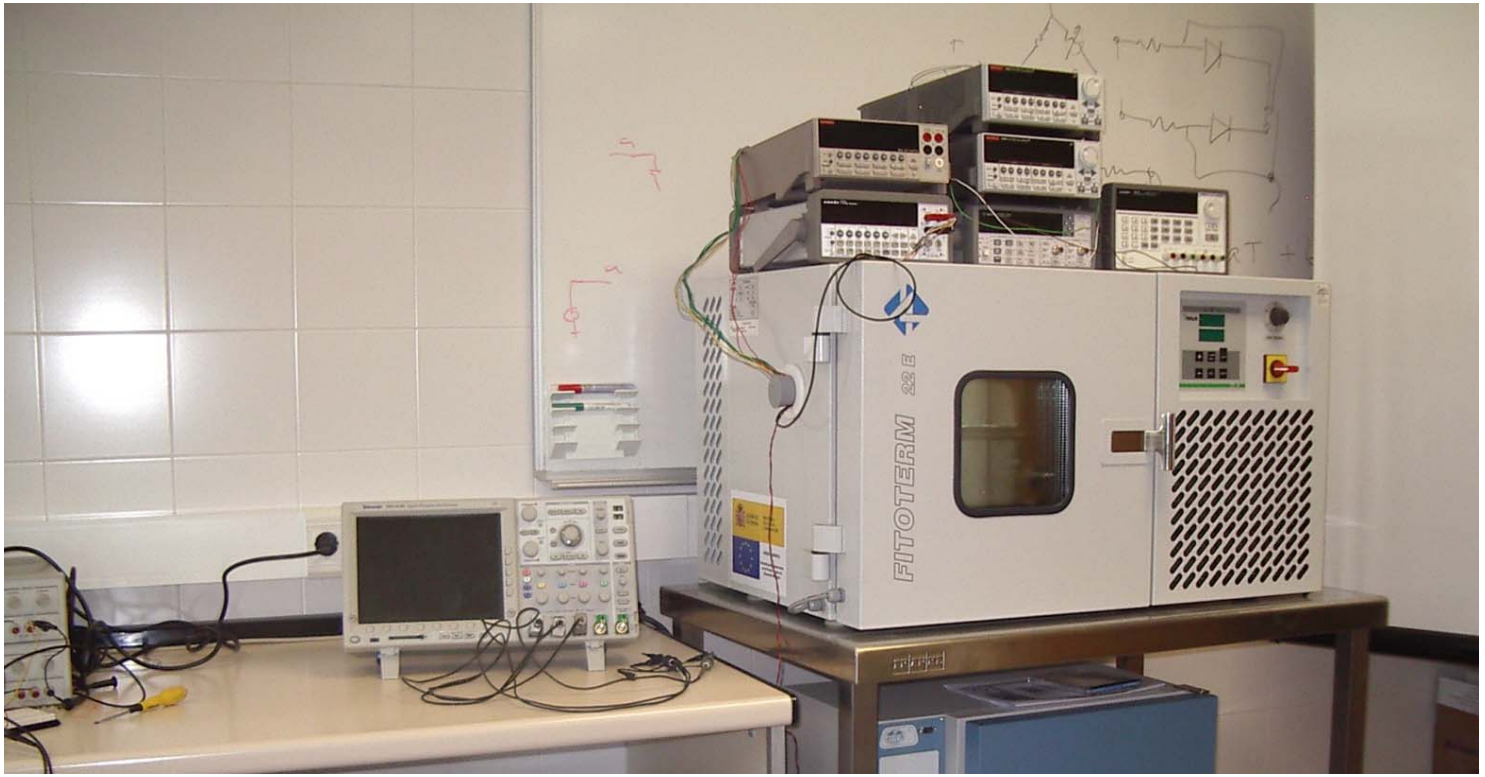
5.1 Instrumentación básica

5.2 Sistemas de interfaz con computador: tarjetas DAQ

5.3 Sistemas de interfaz con computador: buses de instrumentación

5.4 Herramientas software de control y procesado

5.2. Tarjetas DAQ



5.2.1 Tarjetas de adquisición de datos

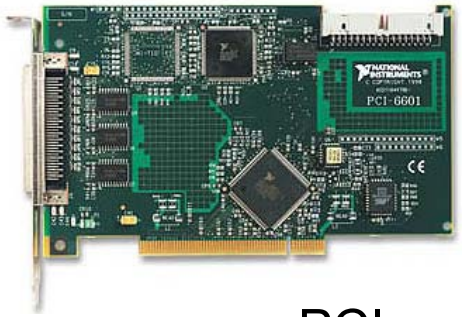
Data Acquisition Systems (DAQ o DAS)

- Sistemas de adquisición de señales eléctricas integrados en computador
- Permiten generar y adquirir señales analógicas y digitales
- Conexión mediante bus PC (PCI), USB
- Prestaciones:
    - 12 a 24 bits
    - 10 kS/s a 2 MS/s

5.2.1 Tarjetas de adquisición de datos

Formatos

Ethernet



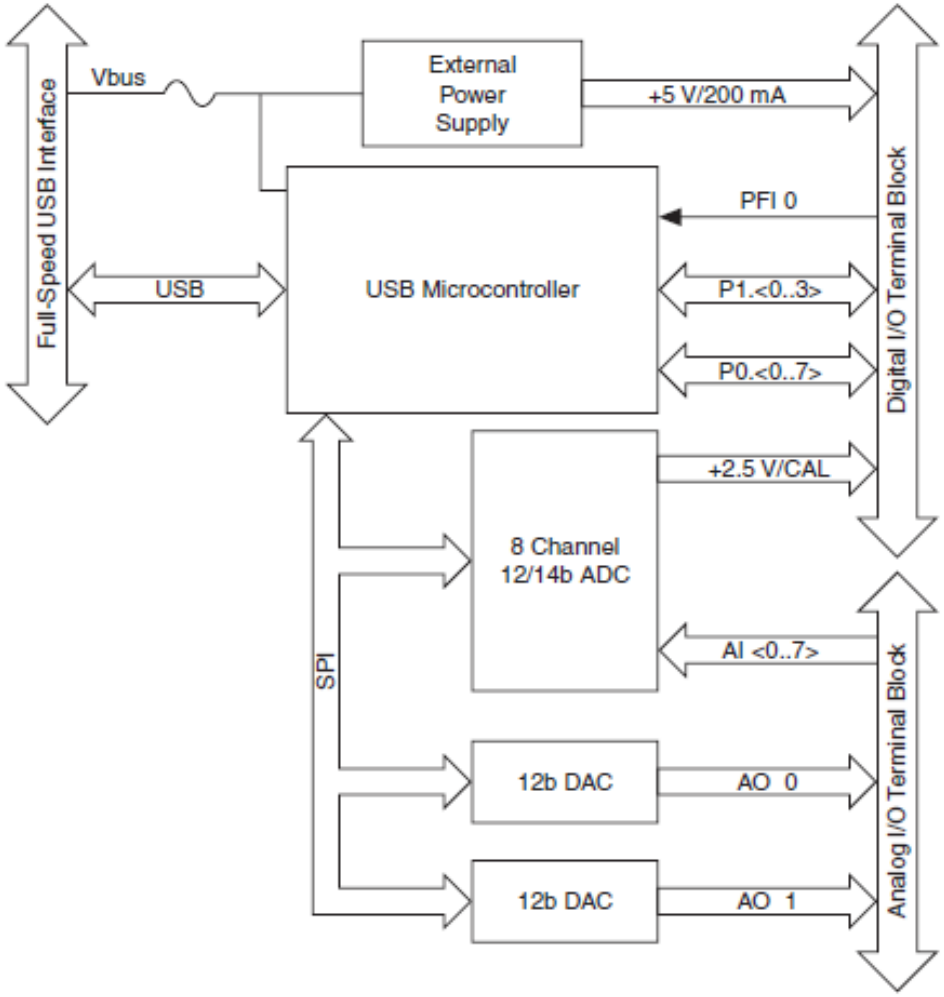
PCI

USB

PXI

5.2.2 DAQ: Bloques básicos

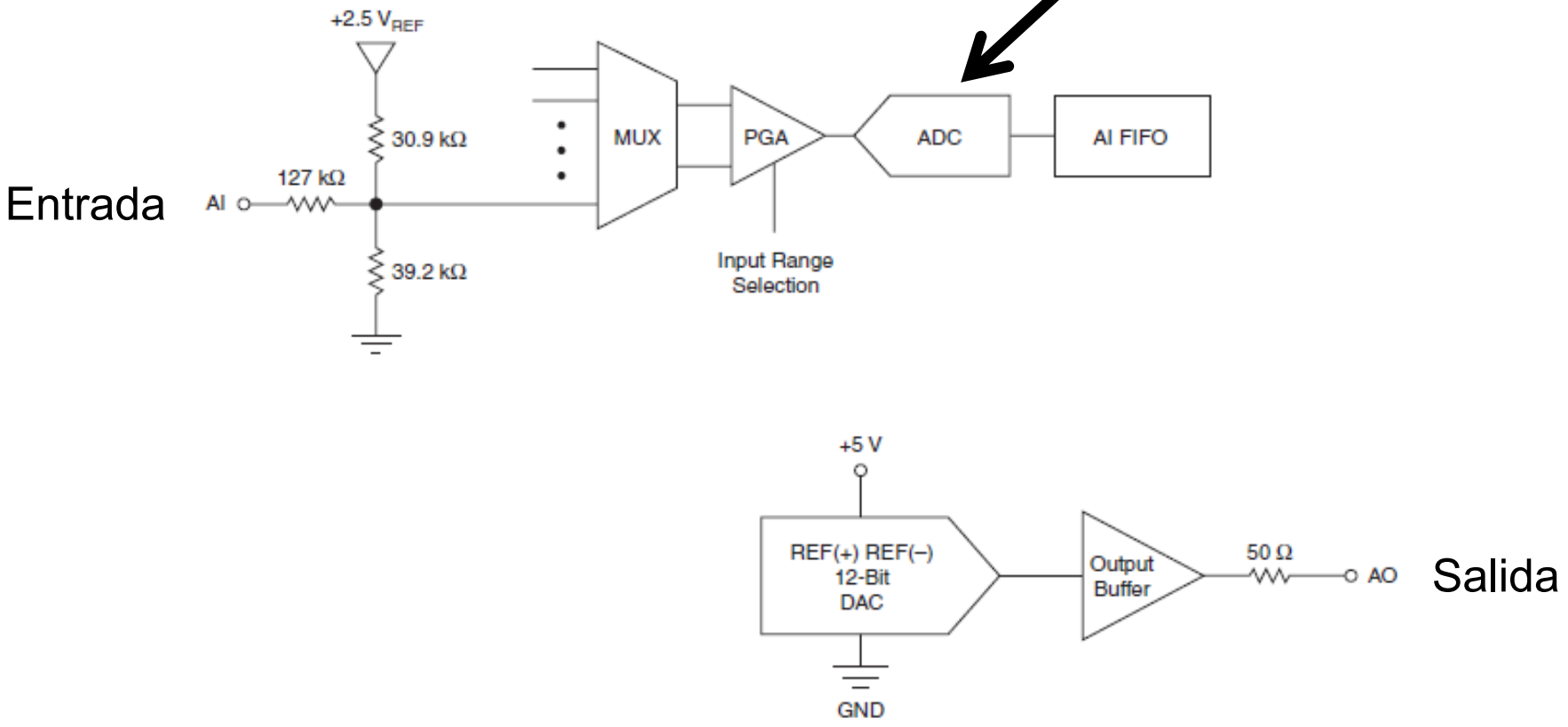
Arquitectura interna



5.2.3 DAQ: Módulos

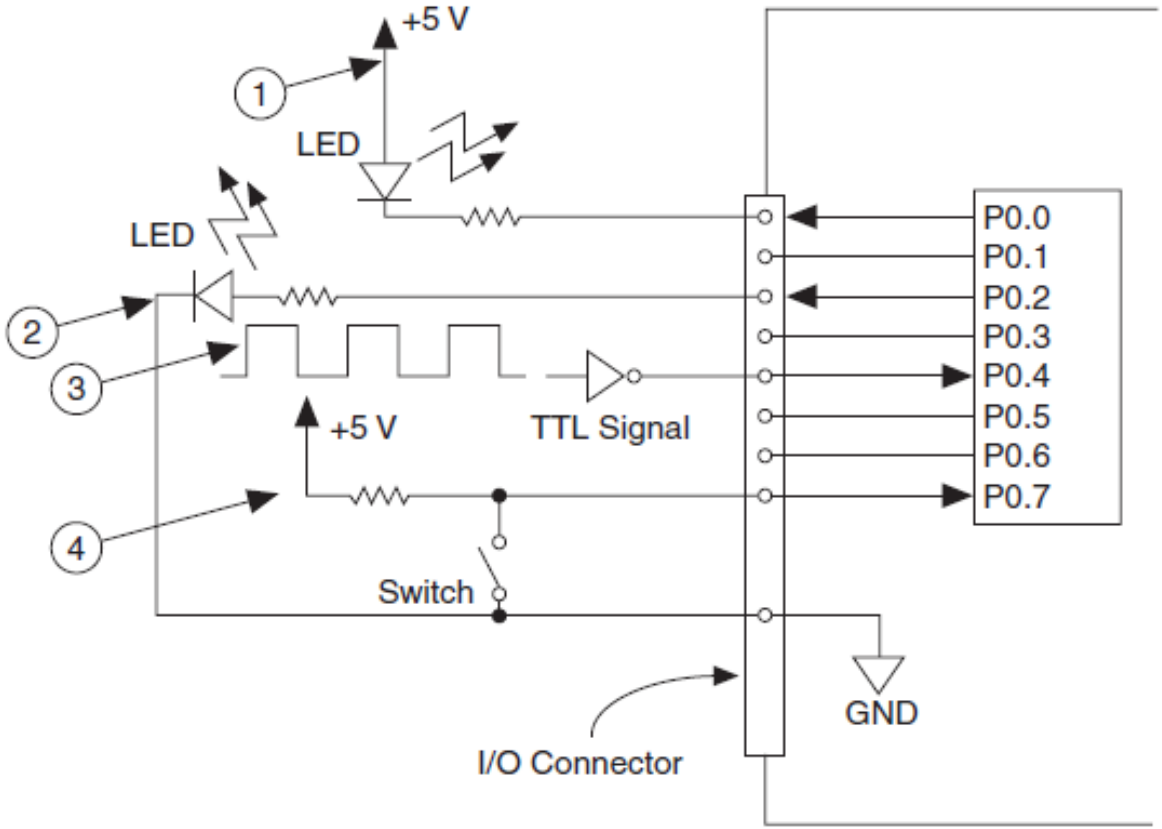
Módulo analógico

Un único ADC multiplexado



5.2.3 DAQ: Módulos

Módulo digital



1 P0.0 configured as an open collector digital output driving an LED
2 P0.2 configured as an active drive digital output driving an LED
3 P0.4 configured as a digital input receiving a TTL signal from a gated invertor
4 P0.7 configured as a digital input receiving a 0 V or 5 V signal from a switch

5.2.3 DAQ: Módulos

Otras características:

- Salida de alimentación
- Salida de referencia
- Entrada de sincronismo (trigger)
- Contadores
- Temporizadores
- Lectura directa de sensores (temperatura)
- …

Adquisición, Control y Procesado

5.1 Instrumentación básica

5.2 Sistemas de interfaz con computador: tarjetas DAQ

5.3 Sistemas de interfaz con computador: buses de instrumentación

5.4 Herramientas software de control y procesado

5.3 Buses de instrumentación

5.3.1 Buses

Buses de propósito general:

- Recommended Standard 232 (RS-232)
- Universal Serial Bus (USB)
- Ethernet
- Peripheral Component Interconnect (PCI)

5.3.2 Tipos de bus

Bus serie

- **Transmisión síncrona**: utiliza línea de transmisión de datos y línea de reloj común para el sincronismo →Comunicación
- **Transmisión asíncrona**: utiliza sólo línea de transmisión de datos; una palabra debe contener bits de inicio y de fin →Instrumentación
- **Velocidad de transmisión**: número de bits por segundo (b/s) o baudios (transiciones de estado). Términos equivalentes en instrumentación

- ❑ RS-232: Recommended Standard 232, punto a punto 1 a 1 señales no diferenciales (20 kb/s, 5 m)
- ❑ RS-422: multipunto 1 a n señales diferenciales (10 Mb/s, 1500 m)
- ❑ RS-485: multipunto n a n señales diferenciales (10 Mb/s, 1200 m)
- ❑ USB (3.0): Universal Serial Bus, alta velocidad (5 Gb/s, 5m) multipunto n a n

5.3.2 Tipos de bus

Bus paralelo

- **Transmisión síncrona**

- ❑ PCI: Peripheral Component Interconnect, 533 MB/s, 64 bits
- ❑ PXI: PCI eXtensions for Instrumentation, 250 MB/s, 32 bits
- ❑ Centronics: El puerto paralelo del PC, 1 MB/s, 8 bits
- ❑ IEEE-488 (GPIB): 8 bits (8 MB/s, 20 m)
- ❑ VXI: VME eXtensions for Instrumentation, 160 MB/s, 32 bits

5.3.3 Bus GPIB

El bus GPIB

Específico para instrumentación

- En 1965, Hewlett-Packard diseña el HP Interface Bus (HP-IB), para conectar sus instrumentos programables a un computador
- En 1975 el Institute of Electrical and Electronics Engineers establece el IEEE Standard 488
- En 1987 se definen el IEEE Standard 488.1 y 488.2
- *En 1990 se establece la norma SCPI (Standard Commands for Programmable Instruments), que determina la estructura de los comandos usados para la comunicación entre los diversos elementos del sistema de instrumentación*

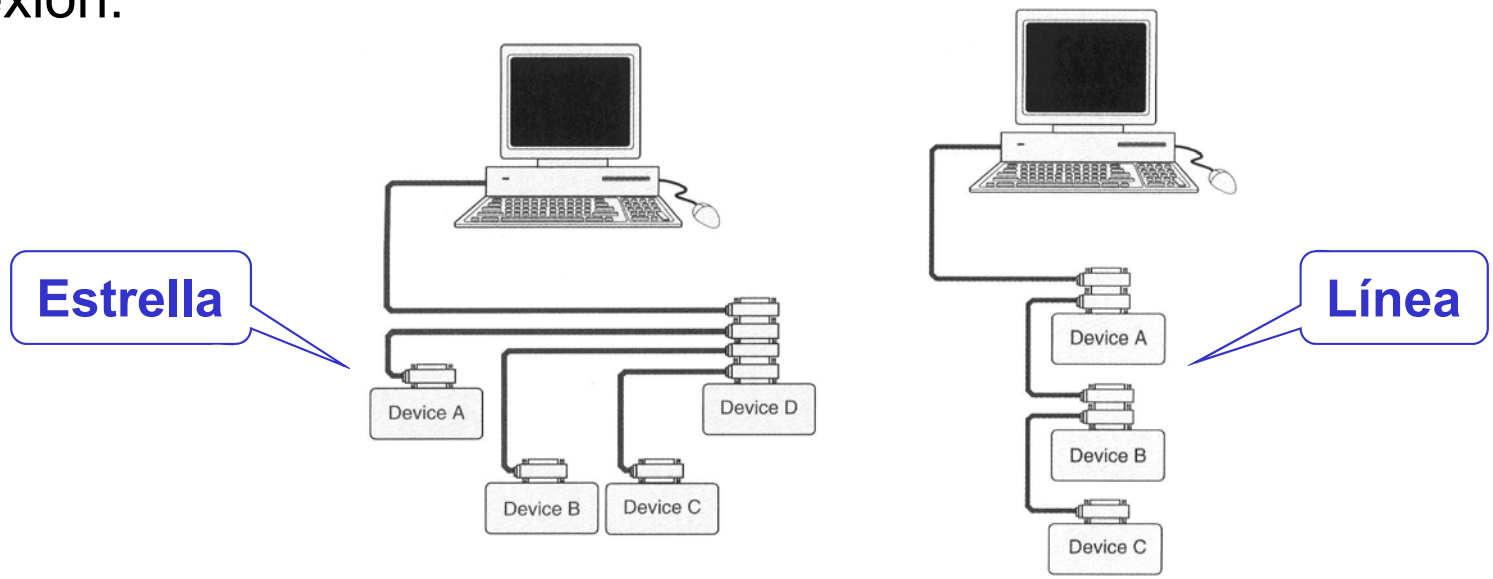
El nombre más común para este estándar en comunicaciones entre instrumentos es GPIB (*General Purpose Interface Bus*)

5.3.3 Bus GPIB

Tipos de instrumentos

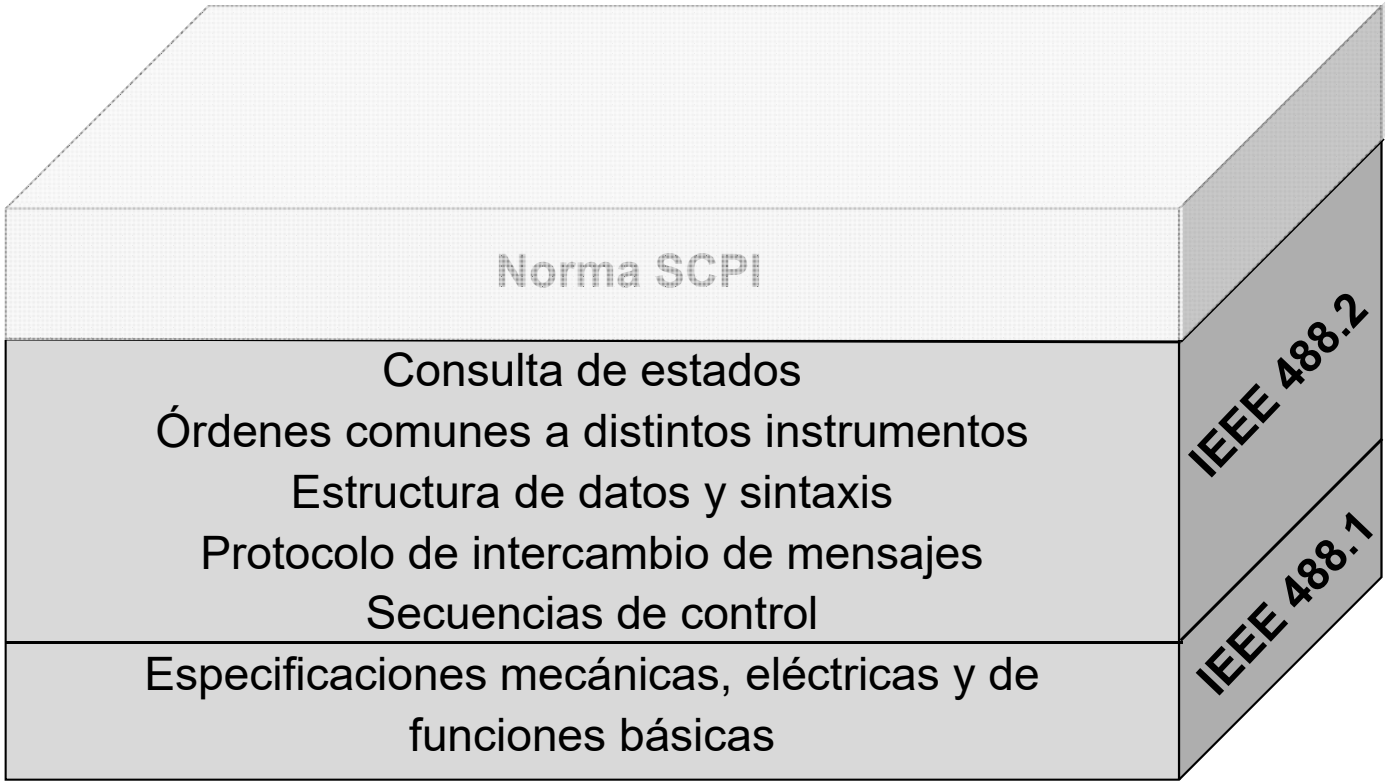
- Talker: Genera mensajes y datos a otros instrumentos
- Listener: Ejecuta los mensajes enviados por otros instrumentos
- Controller: Gestiona el flujo de información por el bus GPIB. Secuencia las operaciones coordinadas de los diferentes instrumentos

Interconexión:



5.3.3 Bus GPIB

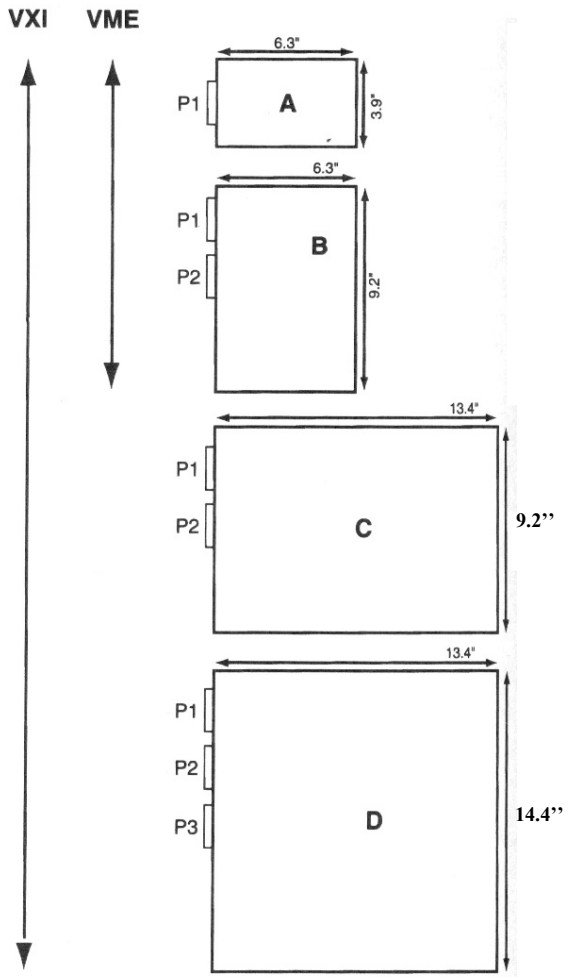
GPIB: relación entre las normas



5.3.4 VXIbus

VXIbus

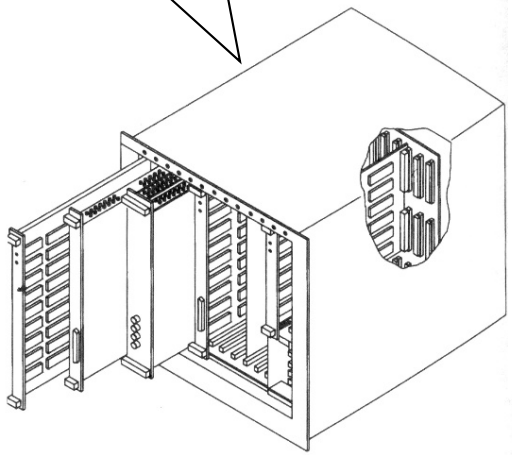
- Denominado VME eXtensions for Instrumentation bus, el bus de instrumentación VXIbus es una evolución del bus VME de computador (Motorola) para proporcionar mejores prestaciones que el GPIB
- Desarrollado a partir de un programa de estandarización del ejército norteamericano
- En la actualidad sus características se encuentran definidas por un conjunto de normas, IEEE 1155
- Los instrumentos son tarjetas conectables al bus VXI (no instrumentos independientes)
- Una de las tarjetas puede ser un *embedded PC*



VXI Card Sizes

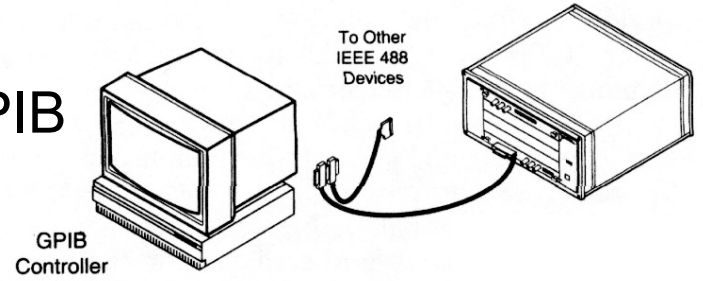
5.3.4 VXIbus

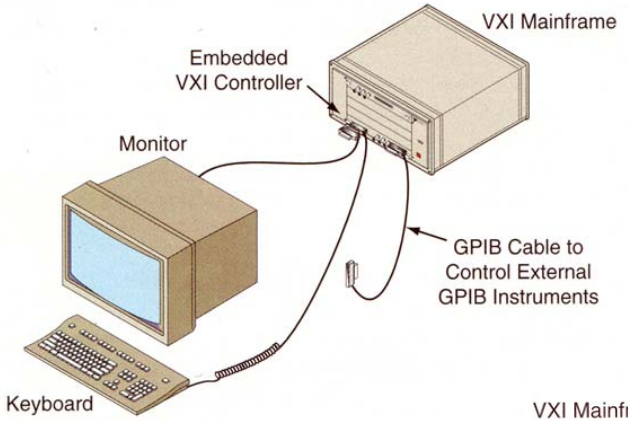
Sistema VXI



Opciones de uso

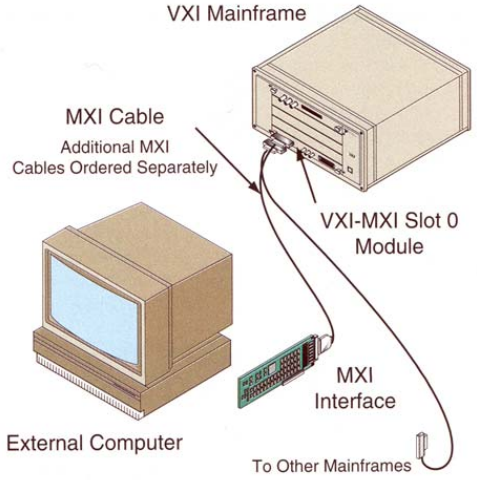
Conectado a GPIB



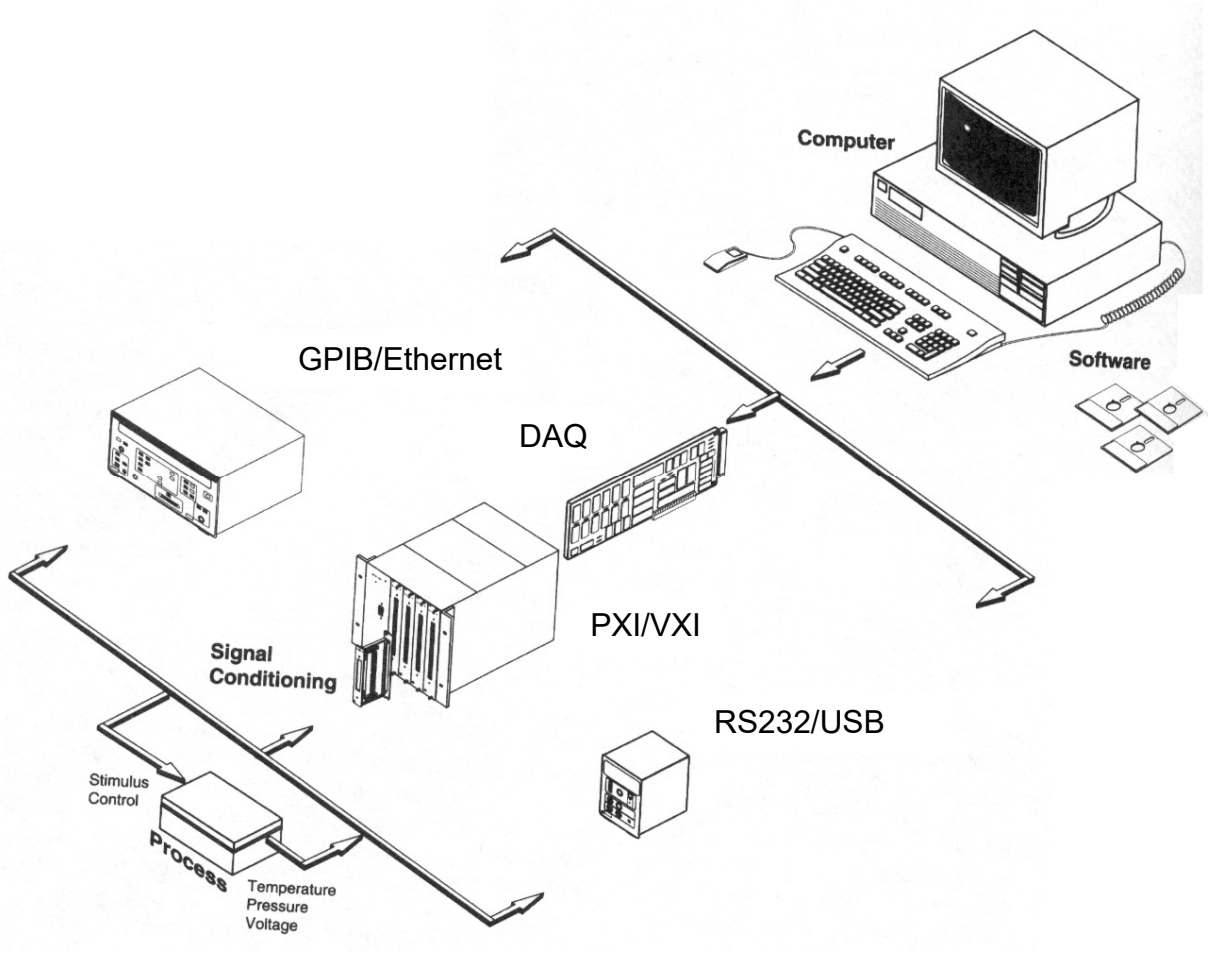


Con PC embebido

Con conexión MXI
a PC estándar



5.3.5 Sistema genérico de adquisición de datos



Adquisición, Control y Procesado

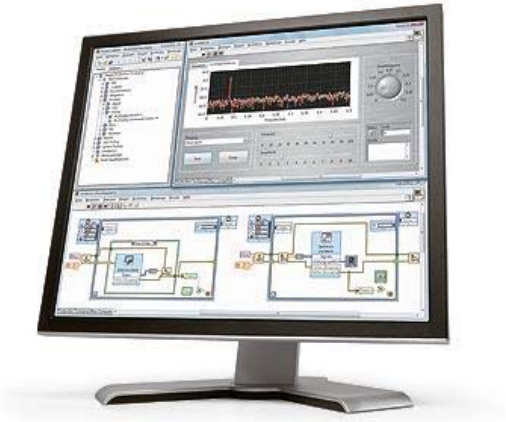
5.1 Instrumentación básica

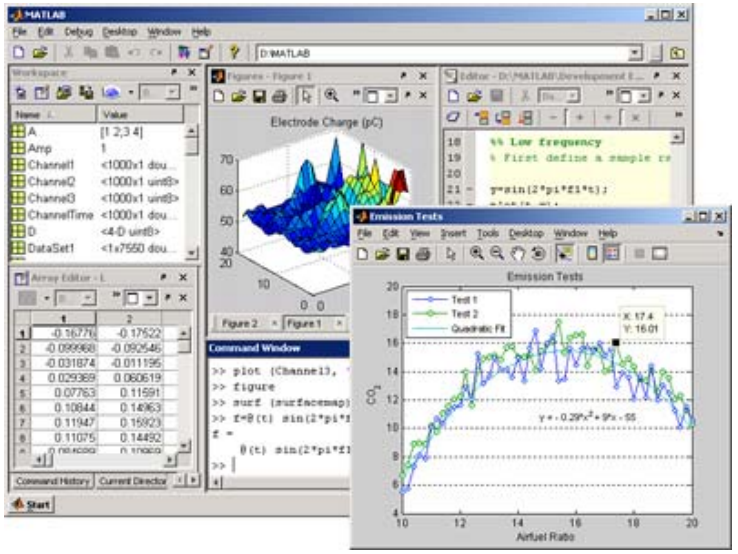
5.2 Sistemas de interfaz con computador: tarjetas DAQ

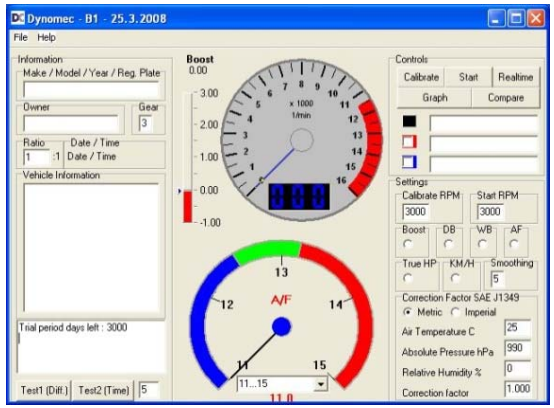
5.3 Sistemas de interfaz con computador: buses de instrumentación

5.4 Herramientas software de control y procesado

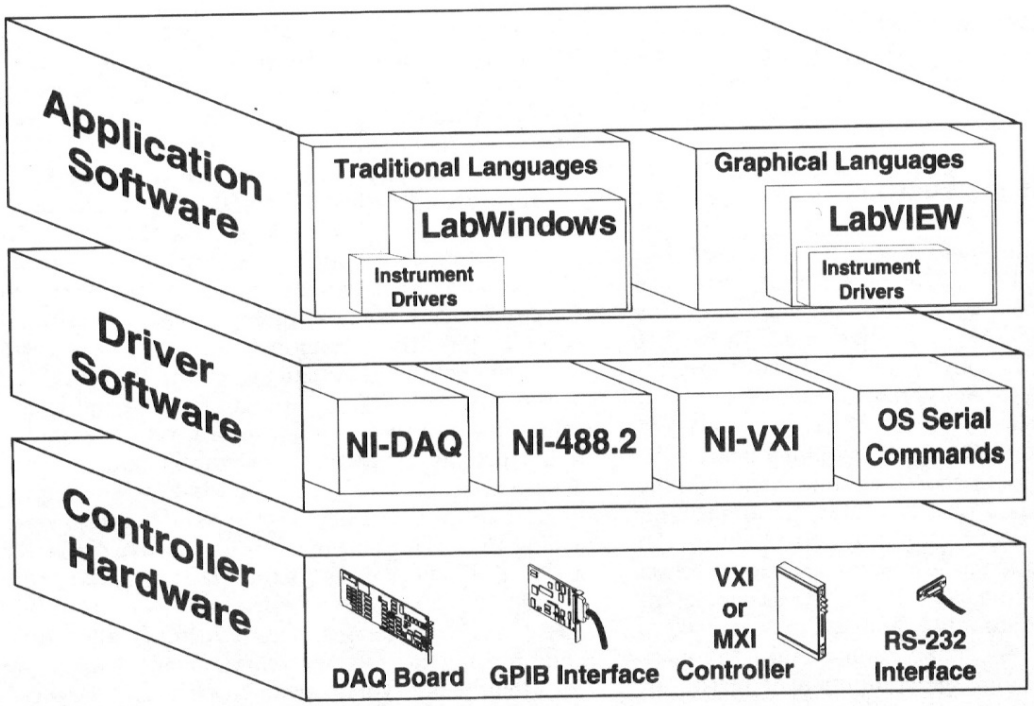
5.4 Software de control y procesado







5.4.1 Instrumentación virtual



Capa de software y/o hardware que se añade a un computador de propósito general de manera que el usuario puede interactuar con éste como si fuese un instrumento electrónico

5.4.2 Programación de instrumentos

- Soporte físico (bus de instrumentación)

  GPIB RS232 VXI USB ETHERNET

- Instrucciones y comandos

  - Instrucciones específicas al bus

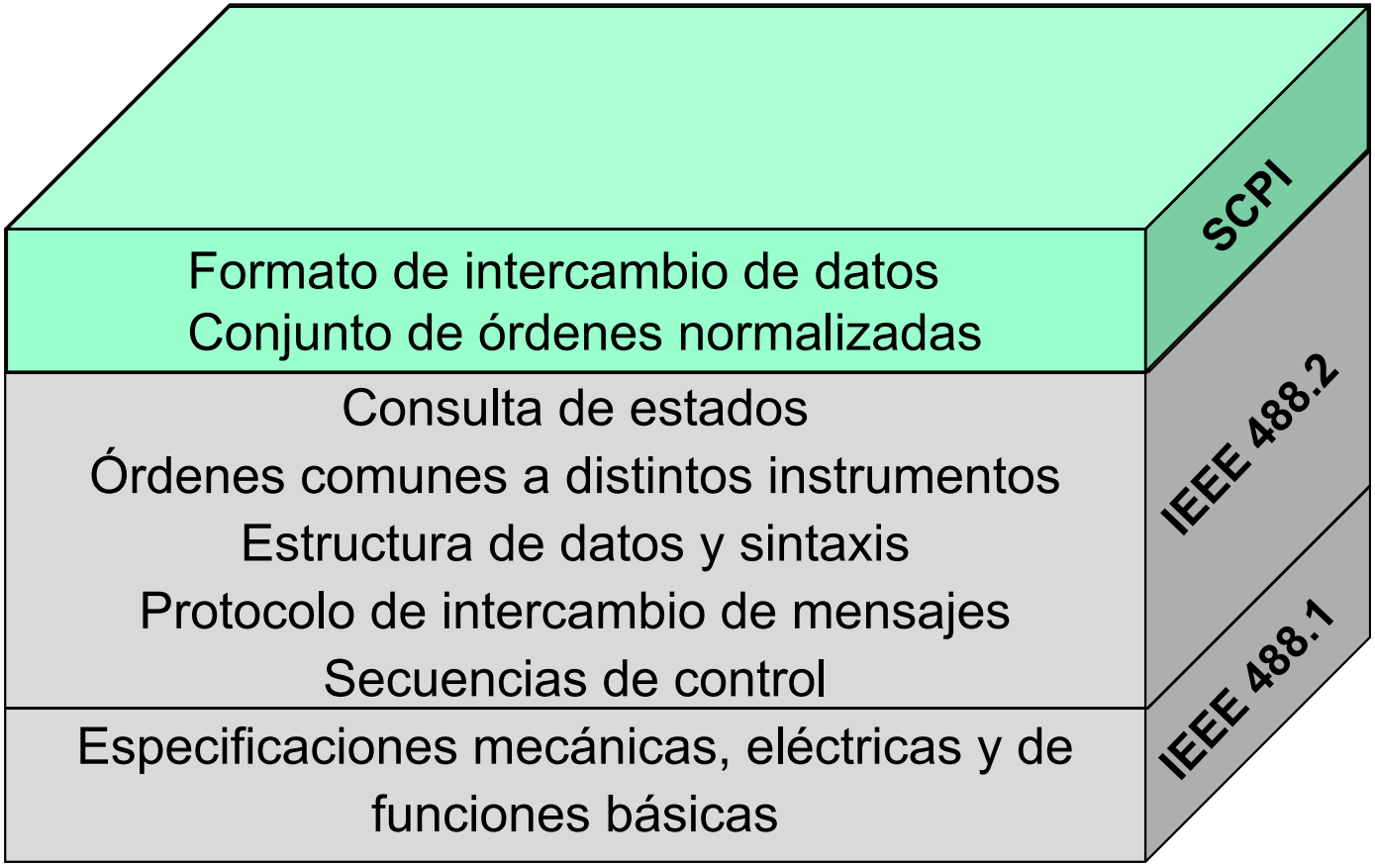
  - Conjunto de instrucciones independientes del bus

    Norma SCPI

    (**S**tandard **C**ommands for **P**rogrammable **I**nstruments)

5.4.2 Programación de instrumentos

Norma SCPI



5.4.2 Programación de instrumentos

SCPI

- Unificar los múltiples lenguajes de programación y sintaxis existentes
- Simplificar la programación de equipos sofisticados y complejos

Promocionar un lenguaje y sintaxis comunes

Adoptado por la mayoría de fabricantes de instrumentos programables (Agilent (HP), Tektronix, Keithley, Fluke…)

5.4.2 Programación de instrumentos

Qué hace SCPI

- Especifica la estructura de comandos y sintaxis
- Incluye conjuntos de comandos comunes
- Comandos con formato **ASCII**, fácilmente comprensibles

Qué NO hace SCPI

- Una definición del soporte físico de la comunicación (IEEE488.2, RS232…)
- El conjunto de comandos del instrumento. Sólo un conjunto básico que cada instrumento debe soportar, común para instrumentos de la misma clase (DMMs)

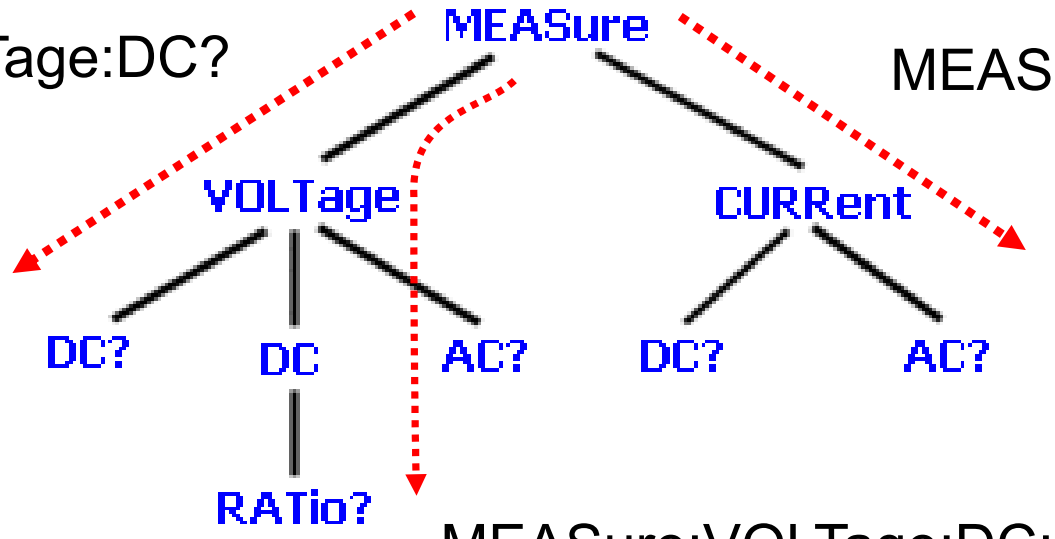
ASCII: American Standard Code for Information Interchange

5.4.2 Programación de instrumentos

Cómo se construye un comando/pregunta:

Recorrer el árbol desde el nodo raíz hacia abajo, delimitando los nodos con dos puntos (:)

MEASure:VOLTage:DC?

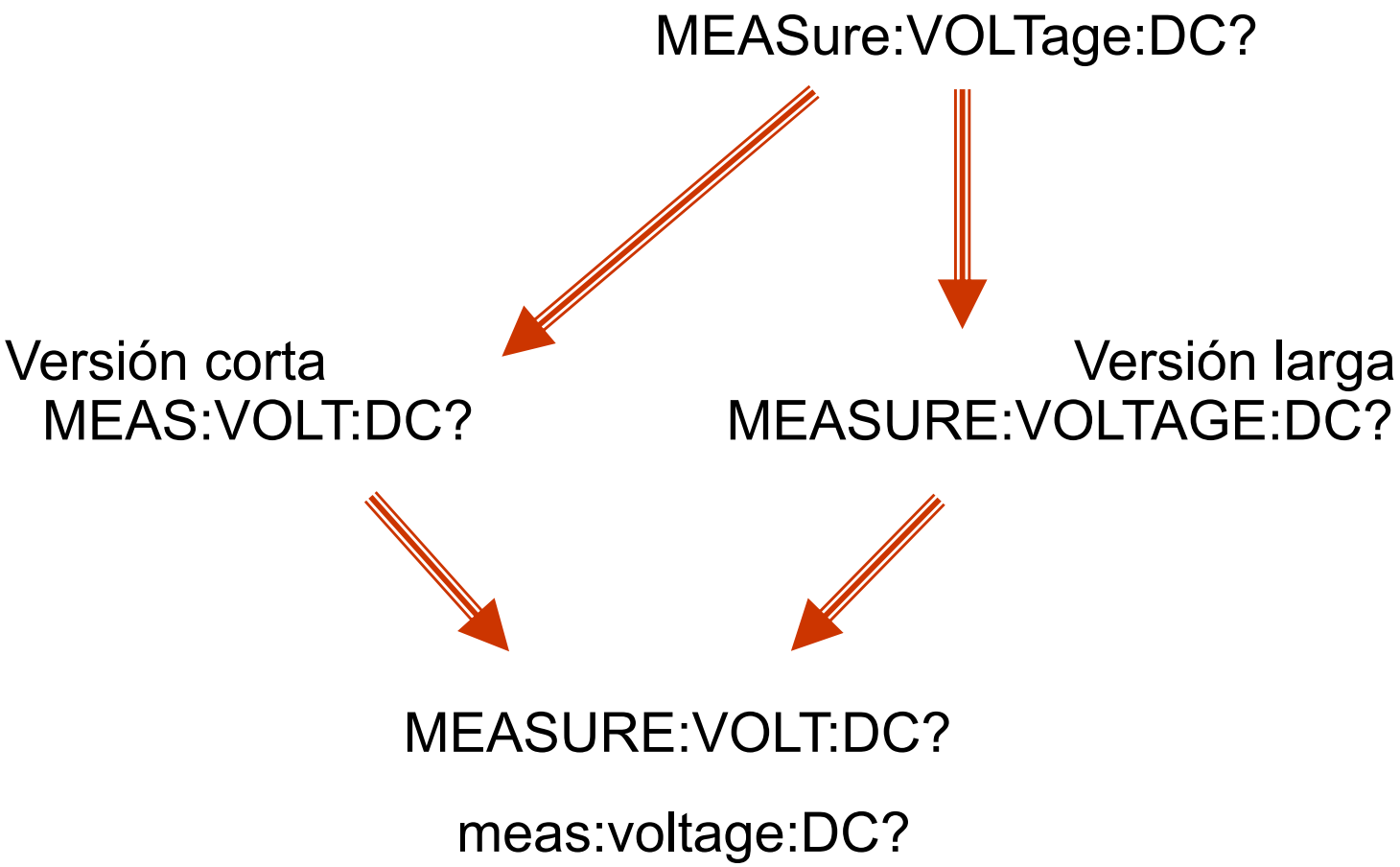


MEASure:CURRent:AC?

MEASure:VOLTage:DC:RATio?

5.4.2 Programación de instrumentos

Opciones



5.4.2 Programación de instrumentos

Sintaxis

Comando <:header>[<space><argument>[<comma><argument>…]]

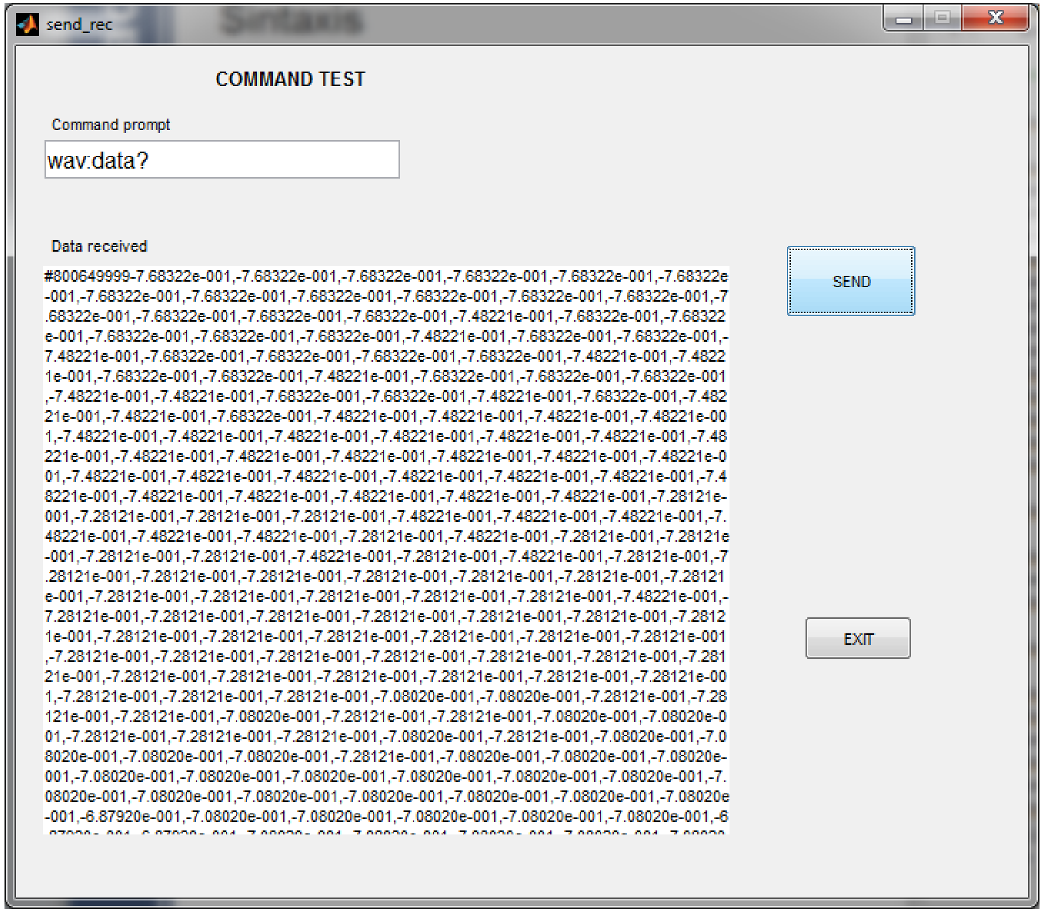
:CHANnel<n>:SCALe <scale>[sufix] → :CHAN1:SCAL 10mV

Pregunta <:header?>[<space><argument>[<comma><argument>…]

:MEASure:FREQuency? [<source>] → :MEAS:FREQ? CH2

5.4.3 Ejemplos prácticos

Vamos a practicar…



5.4.3 Ejemplos prácticos

Vamos a practicar…
con el osciloscopio

*idn?

AUToscale

meas:freq chan1

meas:freq? chan1

meas:vamp? chan1

chan1:coup ac

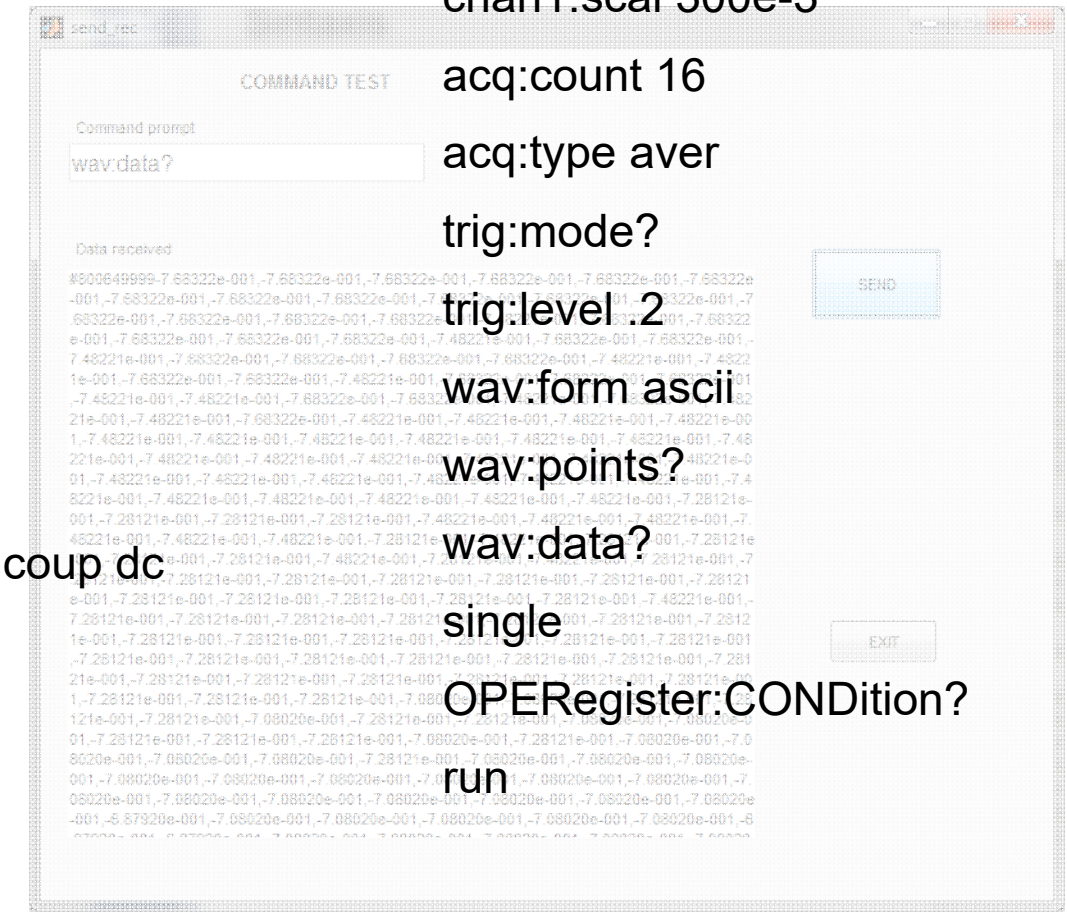
chan1:disp 0

chan1:disp 1;:chan1:coup dc

tim:scal 200e-6

tim:pos 300e-6

tim:pos 0



chan1:scal 300e-3

acq:count 16

acq:type aver

trig:mode?

trig:level .2

wav:form ascii

wav:points?

wav:data?

single

OPERegister:CONDition?

run

5.4.3 Ejemplos prácticos

Vamos a practicar…
con el generador

wgen:outp 0

wgen:outp 1

wgen:func sin

wgen:volt 2

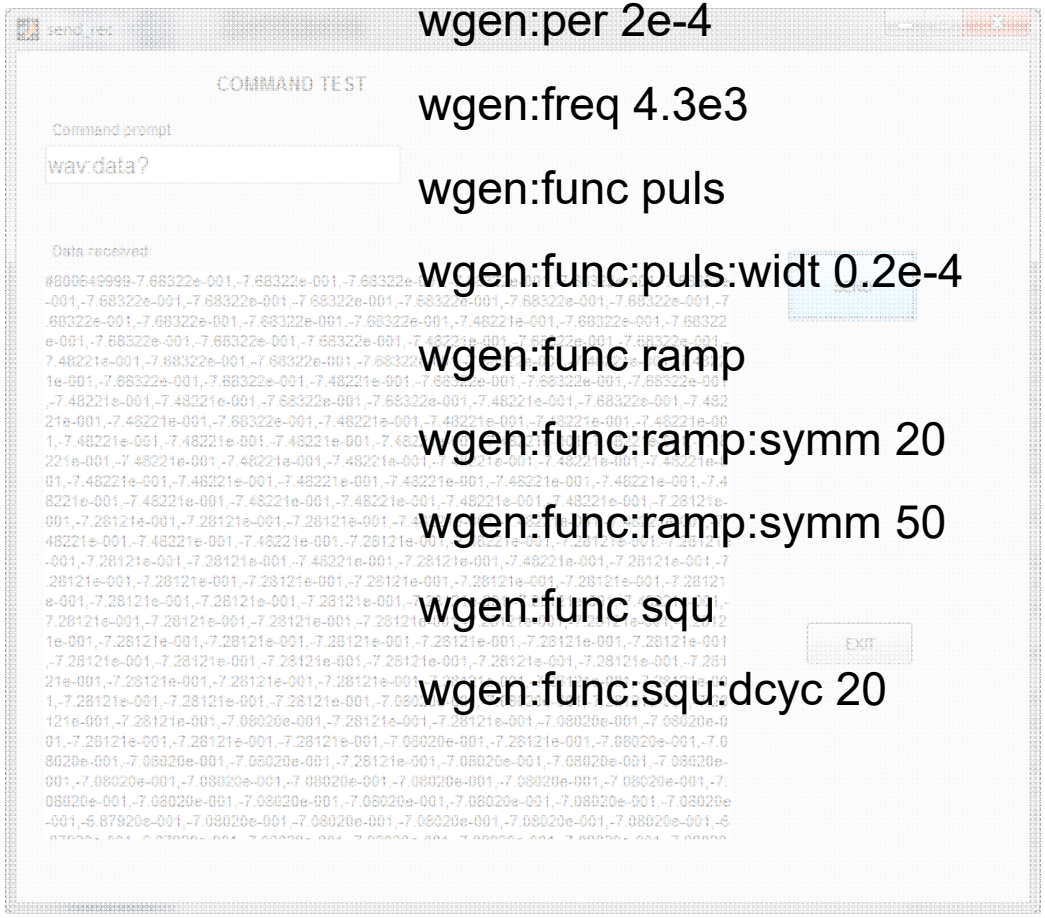
wgen:volt:high 1.5

wgen:volt:low .3

wgen:volt:offs -.5

wgen:volt:offs 0

wgen:volt 1.5



wgen:per 2e-4

wgen:freq 4.3e3

wgen:func puls

wgen:func:puls:widt 0.2e-4

wgen:func ramp

wgen:func:ramp:symm 20

wgen:func:ramp:symm 50

wgen:func squ

wgen:func:squ:dcyc 20

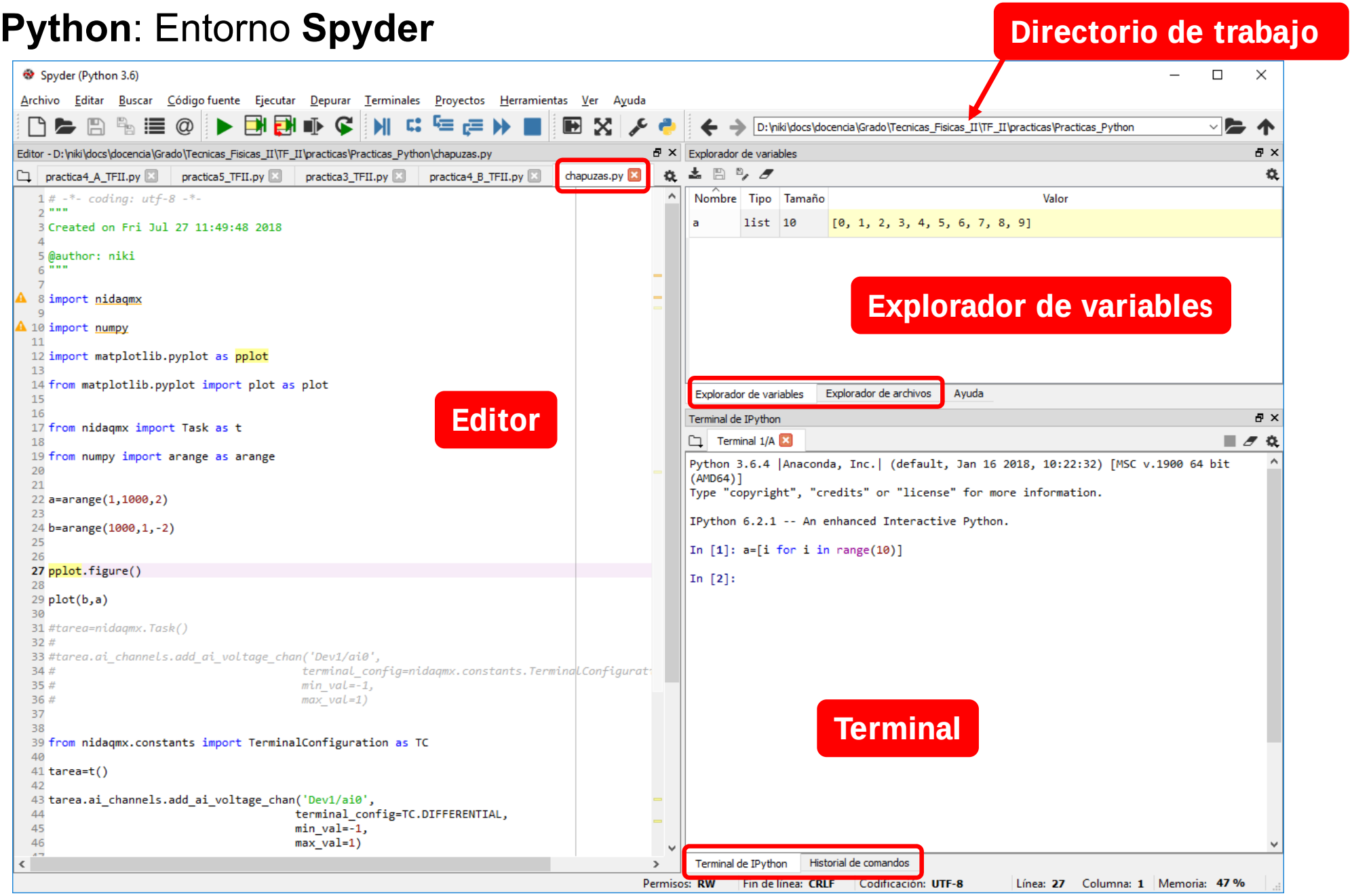
5.4.4 Control de instrumentos desde PC

Python:

- Lenguaje de programación interpretado - Multiplataforma
- Programación por comandos
- Código abierto
- Múltiples tipos de comandos y *Packages*

5.4.4 Control de instrumentos desde PC

Python: Entorno **Spyder**



5.4.4 Control de instrumentos desde PC

Python: Tipos de elementos (I)

- Números

```
a=23
b=15.75
c=8
d=2
```

Operaciones aritméticas

```
Res=a+b
Res=a-b
Res=a*b
Res=a/b
Res=a**2
Res=a//c
Res=a%c
```

Operaciones con bits

```
a|c
a&c
a>>d
a<<d
bin(a<<d)

bin(a)
hex(a)
oct(a)
```

Operaciones lógicas

```
a or c
a and c
not a
a==c
a>c
a<c
a>=c
a<=c
a!=c
Res=a//c
Res=a%c
```

5.4.4 Control de instrumentos desde PC

Python: Tipos de elementos (II)

- Cadenas de caracteres

```
a='hola mundo '
b='adiós '
c=75.5
```

```
a+b
a+b+str(c)

a[0:3]

a='c:\directorio1\directorio2'
a
a='c:\\directorio1\\directorio2'
a
a=r'c:\\directorio1\\directorio2'
a
a='hola\nadiós'
print(a)
a=R'hola\nadios'
print(a)
```

5.4.4 Control de instrumentos desde PC

Python: Tipos de elementos (III)

- Listas

  Arreglos de elementos cuyo valor puede ser alterado. Accesibles a través de su posición

```
a=[1,2,3,‘hola mundo‘]
b=[4,5,6,7,8]
a[0]
a[3]
a[3][1]
a[0]=‘adiós’
a[1]=a[1]+2
```

```
c=a+b
c.append(a)
len(c)
c[9]
c[9]3]
c[9][3][3]
```

5.4.4 Control de instrumentos desde PC

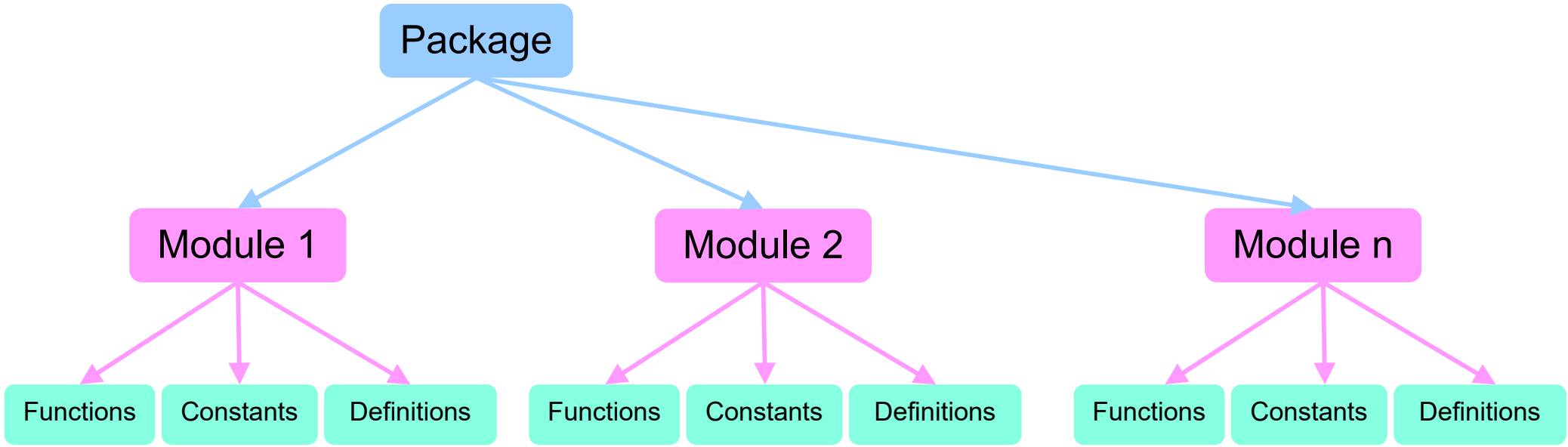
Python: Librerías de funciones (*packages*)

- *Package*: Conjunto de *módulos* con funciones, constantes, etc., con un tema común (representación gráfica, matemáticas).
  - *numpy* – Paquete matemático básico
  - *scipy* – Paquete matemático avanzado
  - *matplotlib* – Representación gráfica
  - *time* – Temporización, relojes, etc.
  - *visa* – Control de instrumentación
  - *nidaqmx* – Gestión de tarjetas de adquisición de datos
- *Módulos* - Subconjuntos de una librería con una unidad:
  - *scipy* – fftpack, integrate, interpolate…
  - *matplotlib* – colours, pyplot, contour…

5.4.4 Control de instrumentos desde PC

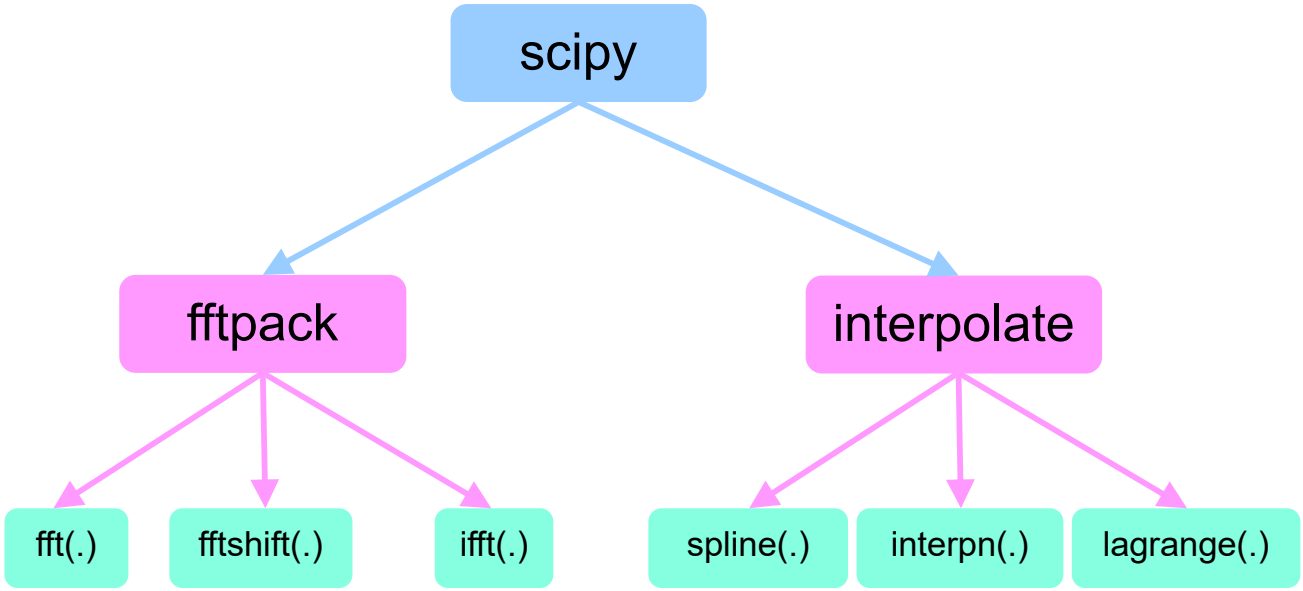
Python: Librerías de funciones (*packages*)

ESTRUCTURA:



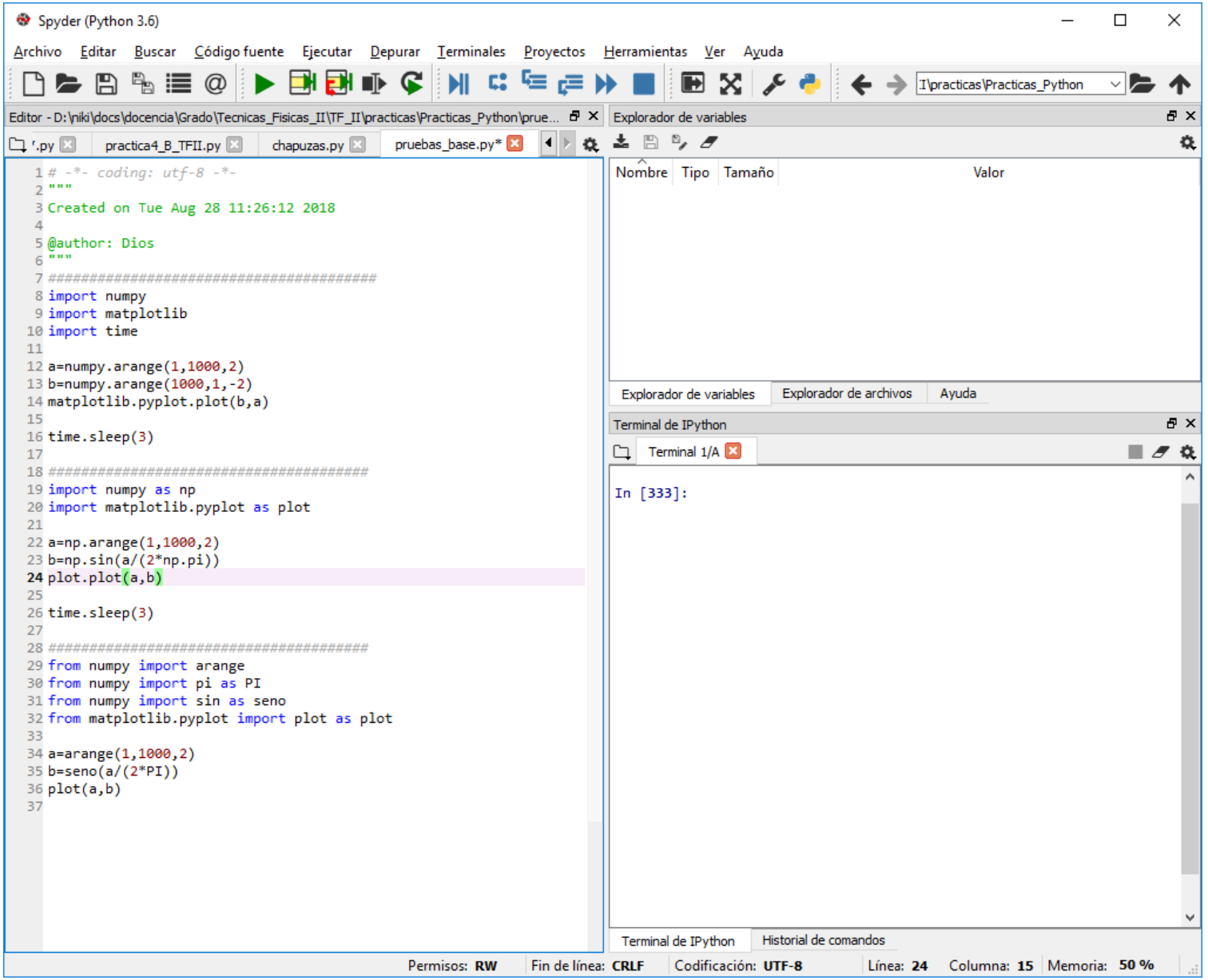
5.4.4 Control de instrumentos desde PC

Python: Librerías de funciones



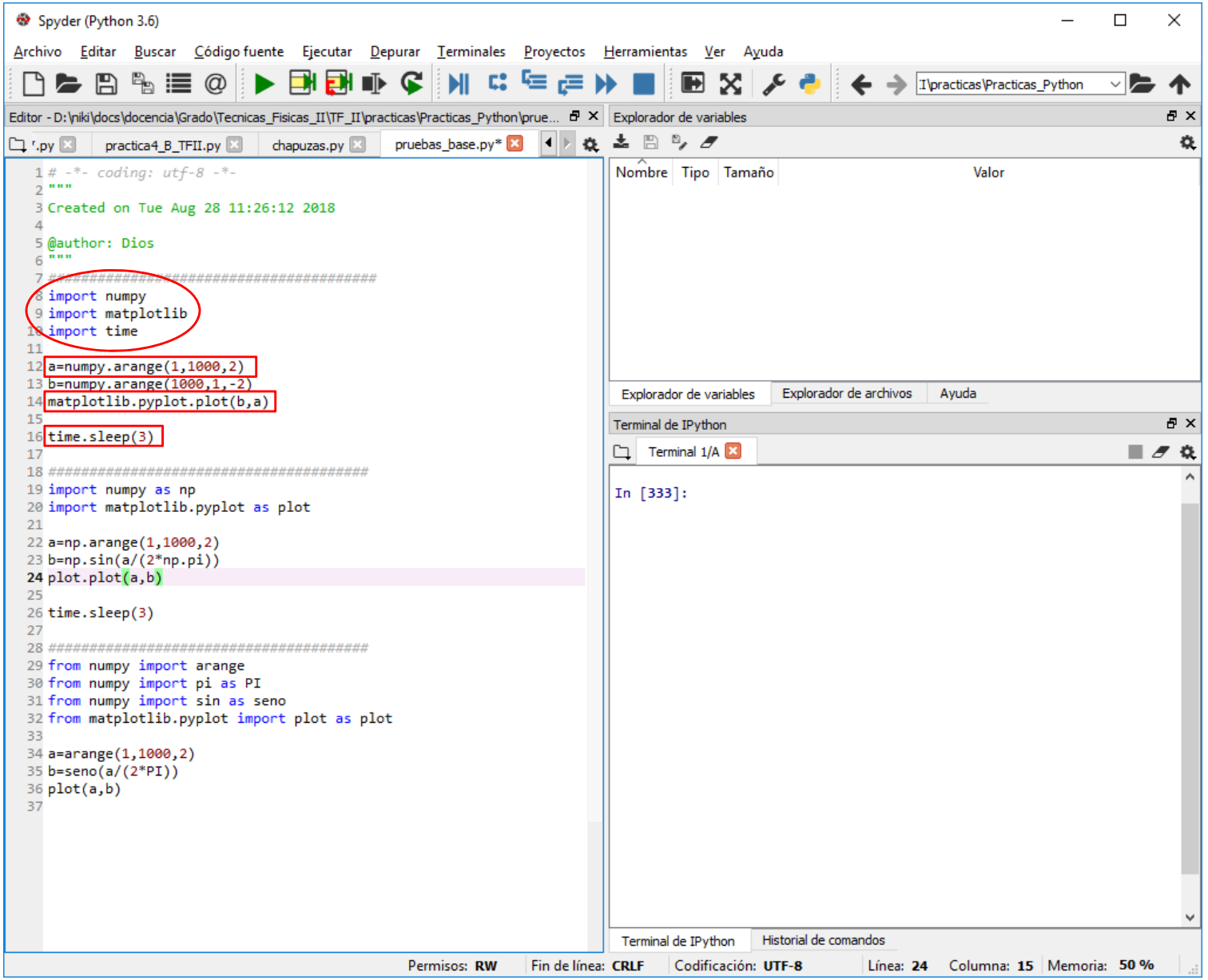
5.4.5 Operadores y funciones matemáticas

Python: Importación y uso de librerías



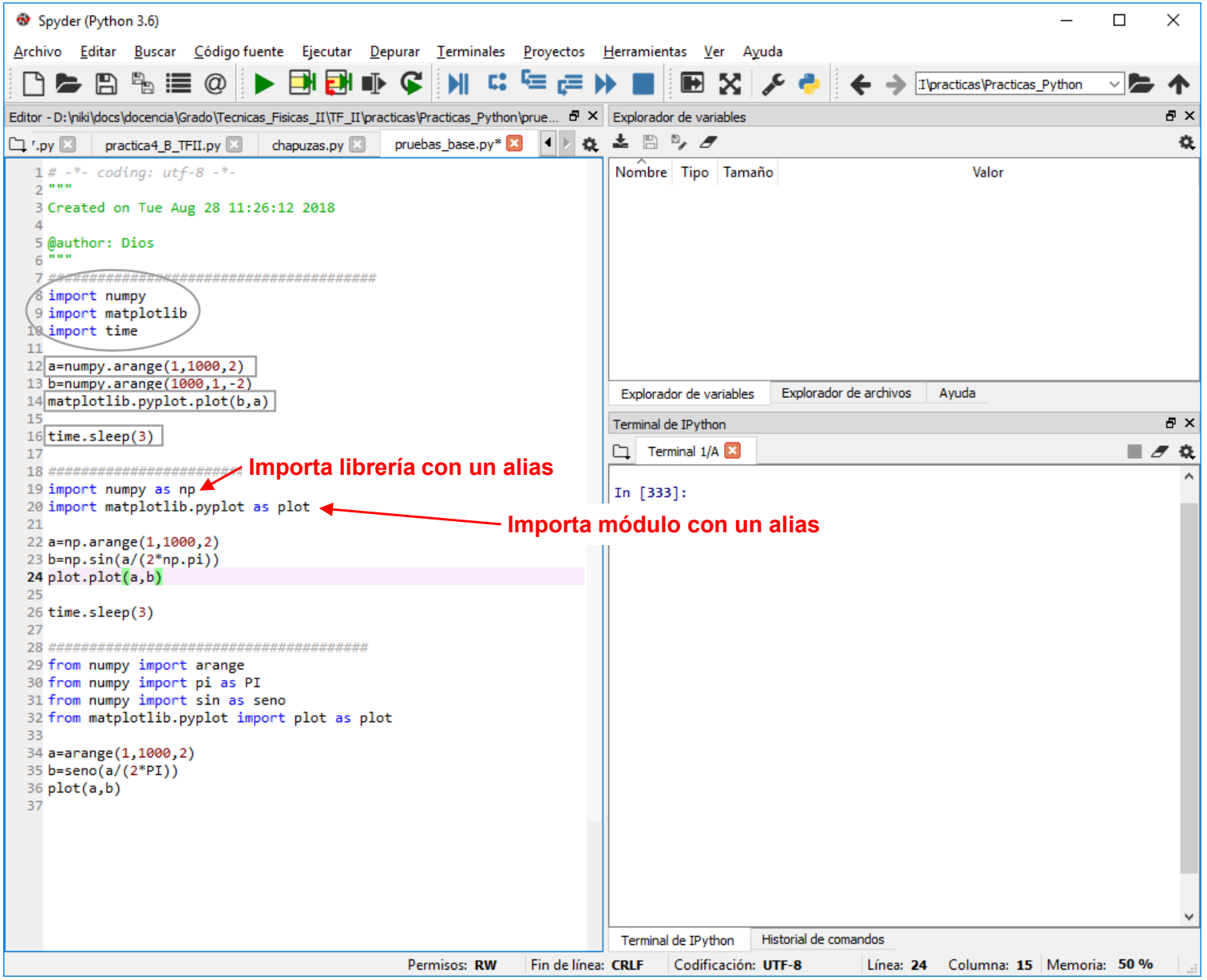
5.4.5 Operadores y funciones matemáticas

Python: Importación y uso de librerías



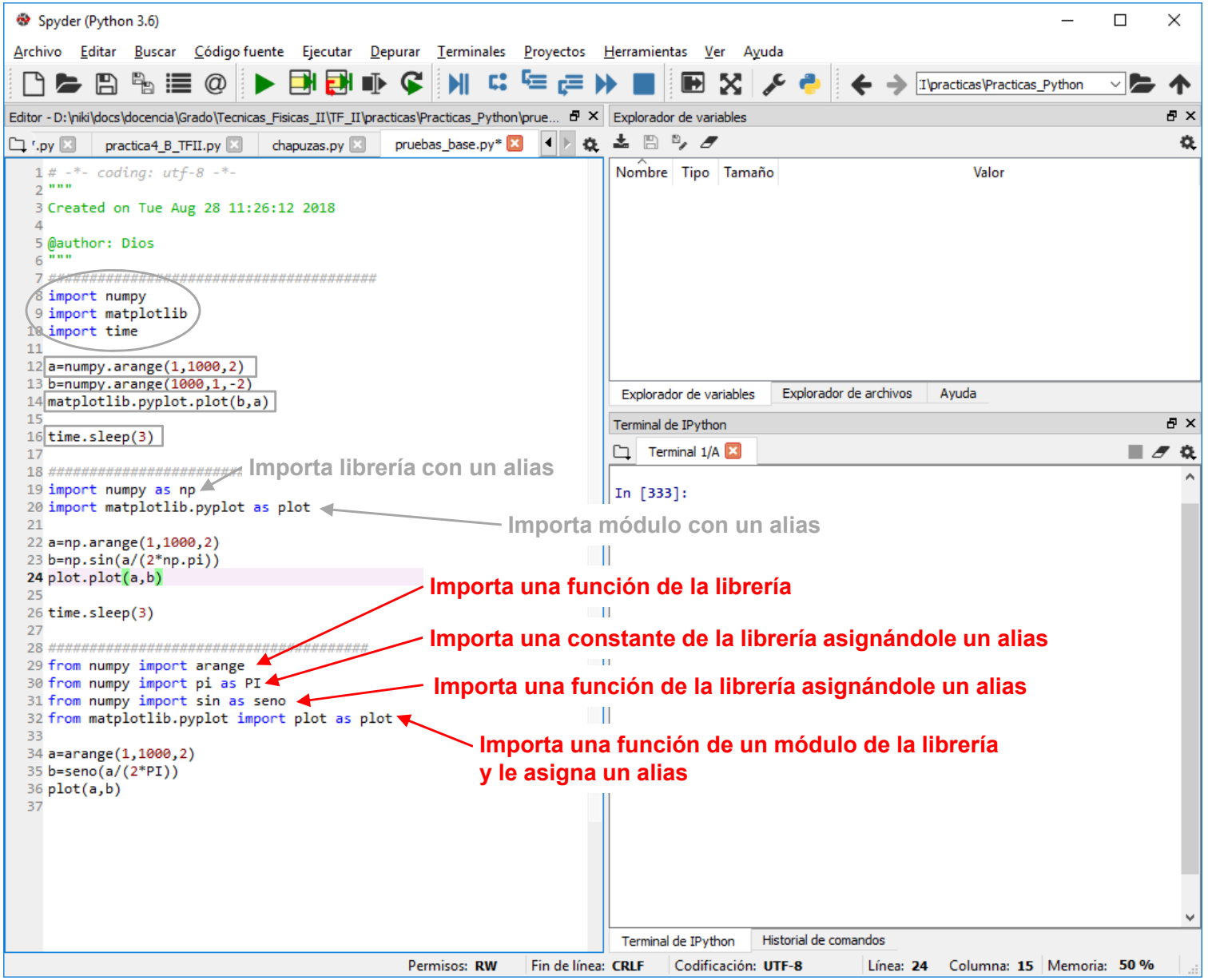
5.4.5 Operadores y funciones matemáticas

Python: Importación y uso de librerías



5.4.5 Operadores y funciones matemáticas

Python: Importación y uso de librerías




```
# -*- coding: utf-8 -*-
"""
Created on Tue Aug 28 11:26:12 2018

@author: Dios
"""
#######################################
import numpy
import matplotlib
import time

a=numpy.arange(1,1000,2)
b=numpy.arange(1000,1,-2)
matplotlib.pyplot.plot(b,a)

time.sleep(3)

######################## Importa librería con un alias
import numpy as np
import matplotlib.pyplot as plot

a=np.arange(1,1000,2)
b=np.sin(a/(2*np.pi))
plot.plot(a,b)

time.sleep(3)

#######################################
from numpy import arange
from numpy import pi as PI
from numpy import sin as seno
from matplotlib.pyplot import plot as plot

a=arange(1,1000,2)
b=seno(a/(2*PI))
plot(a,b)
```

5.4.5 Funciones

```
7  ########################################
8  import numpy as np
9
10 pasos=100
11 a=np.arange(1000)
12 b=np.zeros(2000)
13 c=np.log10(0.707)
14 f=np.logspace(np.log10(100),np.log10(5000),pasos)
15
16 b[0:1000]=a
17
18 np.max(b)
19 np.argmax(b)
20
```

```
7  ########################################
8  import matplotlib.pyplot as plt
9
10 import numpy as np
11 a=np.arange(1000)
12 b=np.sin(a/(2*np.pi))
13
14 plt.plot(a,b)
15 plt.figure()
16 plt.plot(b,a)
17
18 #plt.figure()
19 fig,ax1=plt.subplots()
20 ax1.semilogy(a,b)
21
22 ax2=ax1.twinx()
23 ax2.plot(a,b,color='red')
24
25 plt.figure()
26 plt.step(a,b)
27
28
29 import numpy.fft as FFT
30 fourier=FFT.fft(b)
31 plt.figure()
32 plt.plot(fourier)
33
34 fourier2=FFT.fftshift(fourier)
35 plt.plot(fourier2,color='red')
```

5.4.6 Control del flujo de programa

```
########################################
import matplotlib.pyplot as plt
import time
import numpy as np

a=np.arange(1000)
b=np.sin(a/(2*np.pi))
c=np.zeros(len(a))


########## BUCLE FOR ####################

for i in range(len(a)):
    c[i]=np.cos(a[i]/(np.pi*2))
    print(c[i])
    time.sleep(0.02)

plt.plot(a,b,a,c)

########################################

########## BUCLE WHILE #################

c=np.zeros(len(a))
i=0

t1=time.time()
while i<=len(a)/2:
    c[i]=np.cos(a[i]/(np.pi*2))
    i+=1 #i=i+1
    plt.clf()
    plt.plot(a[0:i],c[0:i])
    plt.pause(0.01)
else:
    print(time.time()-t1)

plt.figure()
plt.plot(a[0:int(len(a)/2)],c[0:int(len(a)/2)])

########################################
```

```
########################################
import matplotlib.pyplot as plt
import time
import numpy as np

a=np.arange(1000)
c=np.zeros(int(len(a)/10))

########## CONDICIONAL IF #################

i=0 #PROBAR CON DIFERENTES VALORES DE i: 0, 1, 4

t1=time.time()

if i==0:
    while i<len(a)/10:
        c[i]=np.cos(a[i]/(np.pi*2))
        plt.clf()
        plt.plot(a[0:i],c[0:i])
        plt.pause(0.01)
        i+=1 #i=i+1
    else:
        print(time.time()-t1)

elif not i%2:
    i=0
    while i<len(a)/10:
        c[i]=np.tan(a[i]/(np.pi*2))
        plt.clf()
        plt.plot(a[0:i],c[0:i])
        plt.pause(0.01)
        i+=1 #i=i+1
    else:
        print(time.time()-t1)

else:
        print('Vale pues, co\n\n')



plt.figure()
plt.plot(a[0:int(len(a)/10)],c)

########################################
```

5.4.6 Control del flujo de programa

```
7  ########################################
8  import matplotlib.pyplot as plt
9  import time
10 import numpy as np
11
12 a=np.arange(1000)
13 b=np.sin(a/(2*np.pi))
14 c=np.zeros(len(a))
15
16
17 ########## BUCLE FOR ####################
18
19 for i in range(len(a)):
20     c[i]=np.cos(a[i]/(np.pi*2))
21     print(c[i])
22     time.sleep(0.02)
23
24 plt.plot(a,b,a,c)
25
26 ########################################
27
28 ########## BUCLE WHILE #################
29
30 c=np.zeros(len(a))
31 i=0
32
33 t1=time.time()
34 while i<=len(a)/2:
35     c[i]=np.cos(a[i]/(np.pi*2))
36     i+=1 #i=i+1
37     plt.clf()
38     plt.plot(a[0:i],c[0:i])
39     plt.pause(0.01)
40 else:
41     print(time.time()-t1)
42
43 plt.figure()
44 plt.plot(a[0:int(len(a)/2)],c[0:int(len(a)/2)])
45
46 ########################################
```

```
7  ########################################
8  import matplotlib.pyplot as plt
9  import time
10 import numpy as np
11
12 a=np.arange(1000)
13 c=np.zeros(int(len(a)/10))
14
15 ########## CONDICIONAL IF ################
16
17 i=0 #PROBAR CON DIFERENTES VALORES DE i: 0, 1, 4
18
19 t1=time.time()
20
21 if i==0:
22     while i<len(a)/10:
23         c[i]=np.cos(a[i]/(np.pi*2))
24         plt.clf()
25         plt.plot(a[0:i],c[0:i])
26         plt.pause(0.01)
27         i+=1 #i=i+1
28     else:
29         print(time.time()-t1)
30
31 elif not i%2:
32     i=0
33     while i<len(a)/10:
34         c[i]=np.tan(a[i]/(np.pi*2))
35         plt.clf()
36         plt.plot(a[0:i],c[0:i])
37         plt.pause(0.01)
38         i+=1 #i=i+1
39     else:
40         print(time.time()-t1)
41
42 else:
43         print('Vale pues, co\n\n')
44
45
46
47 plt.figure()
48 plt.plot(a[0:int(len(a)/10)],c)
49
50 ########################################
51
```

5.4.6 Control del flujo de programa

```
#########################################
import matplotlib.pyplot as plt
import time
import numpy as np

a=np.arange(1000)
b=np.sin(a/(2*np.pi))
c=np.zeros(len(a))


########## BUCLE FOR ####################

for i in range(len(a)):
    c[i]=np.cos(a[i]/(np.pi*2))
    print(c[i])
    time.sleep(0.02)

plt.plot(a,b,a,c)

#########################################

########## BUCLE WHILE #################

c=np.zeros(len(a))
i=0

t1=time.time()
while i<=len(a)/2:
    c[i]=np.cos(a[i]/(np.pi*2))
    i+=1 #i=i+1
    plt.clf()
    plt.plot(a[0:i],c[0:i])
    plt.pause(0.01)
else:
    print(time.time()-t1)

plt.figure()
plt.plot(a[0:int(len(a)/2)],c[0:int(len(a)/2)])

#########################################
```

```
#########################################
import matplotlib.pyplot as plt
import time
import numpy as np

a=np.arange(1000)
c=np.zeros(int(len(a)/10))

########## CONDICIONAL IF #################

i=0 #PROBAR CON DIFERENTES VALORES DE i: 0, 1, 4

t1=time.time()

if i==0:
    while i<len(a)/10:
        c[i]=np.cos(a[i]/(np.pi*2))
        plt.clf()
        plt.plot(a[0:i],c[0:i])
        plt.pause(0.01)
        i+=1 #i=i+1
    else:
        print(time.time()-t1)

elif not i%2:
    i=0
    while i<len(a)/10:
        c[i]=np.tan(a[i]/(np.pi*2))
        plt.clf()
        plt.plot(a[0:i],c[0:i])
        plt.pause(0.01)
        i+=1 #i=i+1
    else:
        print(time.time()-t1)

else:
        print('Vale pues, co\n\n')



plt.figure()
plt.plot(a[0:int(len(a)/10)],c)

#########################################
```

5.4.6 Control del flujo de programa

```
########################################
import matplotlib.pyplot as plt
import time
import numpy as np

a=np.arange(1000)
b=np.sin(a/(2*np.pi))
c=np.zeros(len(a))


########## BUCLE FOR ###################

for i in range(len(a)):
    c[i]=np.cos(a[i]/(np.pi*2))
    print(c[i])
    time.sleep(0.02)

plt.plot(a,b,a,c)

########################################

########## BUCLE WHILE #################

c=np.zeros(len(a))
i=0

t1=time.time()
while i<=len(a)/2:
    c[i]=np.cos(a[i]/(np.pi*2))
    i+=1 #i=i+1
    plt.clf()
    plt.plot(a[0:i],c[0:i])
    plt.pause(0.01)
else:
    print(time.time()-t1)

plt.figure()
plt.plot(a[0:int(len(a)/2)],c[0:int(len(a)/2)])

########################################
```

```
########################################
import matplotlib.pyplot as plt
import time
import numpy as np

a=np.arange(1000)
c=np.zeros(int(len(a)/10))

########## CONDICIONAL IF #################

i=0 #PROBAR CON DIFERENTES VALORES DE i: 0, 1, 4

t1=time.time()

if i==0:
    while i<len(a)/10:
        c[i]=np.cos(a[i]/(np.pi*2))
        plt.clf()
        plt.plot(a[0:i],c[0:i])
        plt.pause(0.01)
        i+=1 #i=i+1
    else:
        print(time.time()-t1)

elif not i%2:
    i=0
    while i<len(a)/10:
        c[i]=np.tan(a[i]/(np.pi*2))
        plt.clf()
        plt.plot(a[0:i],c[0:i])
        plt.pause(0.01)
        i+=1 #i=i+1
    else:
        print(time.time()-t1)

else:
        print('Vale pues, co\n\n')



plt.figure()
plt.plot(a[0:int(len(a)/10)],c)

########################################

```

5.4.7 Librerías de funciones

Control de instrumentos

```
7  ########################################
8  import visa
9
10 resources=visa.ResourceManager()
11
12 resources.list_resources()
13
14 instrumento=resources.open_resource('USB0::0x0957::0x179B::MY51250761::INSTR')
15
16
17
18 instrumento.write('wgen:outp 1')
19
20 instrumento.write('wgen:func sin;volt 1;volt:offs 0')
21
22 instrumento.write('wgen:freq '+str(freq[0]))
23
24
25 vamp=float(instrumento.query('meas:vamp? chan2'))
26
27 for i in range(pasos):
28     instrumento.write('wgen:freq '+str(freq[i]))
29     medida_o[i]=float(instrumento.query('meas:vamp? chan2'))
30
31 instrumento.close() # cierra la sesion
32
33
```

5.4.7 Librerías de funciones

Control de instrumentos

```
7  ########################################
8  import visa
9
10 resources=visa.ResourceManager()
11
12 resources.list_resources()
13
14 instrumento=resources.open_resource('USB0::0x0957::0x179B::MY51250761::INSTR')
15
16
17
18 instrumento.write('wgen:outp 1')
19
20 instrumento.write('wgen:func sin;volt 1;volt:offs 0')
21
22 instrumento.write('wgen:freq '+str(freq[0]))
23
24
25 vamp=float(instrumento.query('meas:vamp? chan2'))
26
27 for i in range(pasos):
28     instrumento.write('wgen:freq '+str(freq[i]))
29     medida_o[i]=float(instrumento.query('meas:vamp? chan2'))
30
31 instrumento.close() # cierra la sesion
32
33
```

5.4.9 Librerías de funciones

Adquisición de datos (DAQ) → **NO usan SCPI**

```
7  ########################################
8  import nidaqmx
9
10 #from nidaqmx import Task as t
11
12
13 tarea=nidaqmx.Task()
14
15 tarea.ai_channels.add_ai_voltage_chan('Dev1/ai0',
16                                        terminal_config=nidaqmx.constants.TerminalConfiguration.DIFFERENTIAL,
17                                        min_val=-1,
18                                        max_val=1)
19
20
21 #from nidaqmx.constants import TerminalConfiguration as TC
22 #
23 #tarea=t()
24 #
25 #tarea.ai_channels.add_ai_voltage_chan('Dev1/ai0',
26 #                                       terminal_config=TC.DIFFERENTIAL,
27 #                                       min_val=-1,
28 #                                       max_val=1)
29
30 valores=tarea.read(number_of_samples_per_channel=1000)
31
32 tarea.stop()
33
34 tarea.close()
35
36
```

5.4.9 Librerías de funciones

Adquisición de datos (DAQ) → NO usan SCPI

```
7  ########################################
8  import nidaqmx
9
10 #from nidaqmx import Task as t
11
12
13 tarea=nidaqmx.Task()
14
15 tarea.ai_channels.add_ai_voltage_chan('Dev1/ai0',
16                                       terminal_config=nidaqmx.constants.TerminalConfiguration.DIFFERENTIAL,
17                                       min_val=-1,
18                                       max_val=1)
19
20
21 #from nidaqmx.constants import TerminalConfiguration as TC
22 #
23 #tarea=t()
24 #
25 #tarea.ai_channels.add_ai_voltage_chan('Dev1/ai0',
26 #                                      terminal_config=TC.DIFFERENTIAL,
27 #                                      min_val=-1,
28 #                                      max_val=1)
29
30 valores=tarea.read(number_of_samples_per_channel=1000)
31
32 tarea.stop()
33
34 tarea.close()
35
36
```

5.4.9 Librerías de funciones

Adquisición de datos (DAQ) ⟶ NO usan SCPI

```
7  ########################################
8  import nidaqmx
9
10 from nidaqmx import Task as t
11
12
13 #tarea=nidaqmx.Task()
14 #
15 #tarea.ai_channels.add_ai_voltage_chan('Dev1/ai0',
16 #                                      terminal_config=nidaqmx.constants.TerminalConfiguration.DIFFERENTIAL,
17 #                                      min_val=-1,
18 #                                      max_val=1)
19
20
21 from nidaqmx.constants import TerminalConfiguration as TC
22
23 tarea=t()
24
25 tarea.ai_channels.add_ai_voltage_chan('Dev1/ai0',
26                                       terminal_config=TC.DIFFERENTIAL,
27                                       min_val=-1,
28                                       max_val=1)
29
30 valores=tarea.read(number_of_samples_per_channel=1000)
31
32 tarea.stop()
33
34 tarea.close()
35
36
37
```