Test Instrument Automation / Control

My Tricks and Recommendations

You know my passion for test instruments…

Bertrand Zauhar, VE2ZAZ
ve2zaz@rac.ca
October 2012

Today's Program on Instrument Control

- The need for test intrument automation/control,
- The available electrical interfaces,
- A closer look at GPIB,
- An overview of a typical test automation cycle,
- The available software environments,
- The Required documentation,
- A suggested approach: Python / Qt4
- Instrument Control with the Raspberry Pi

Why Automating Instrument Control?

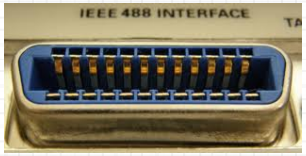
- Simplifies repetitive or complex tasks (several measurements or several instruments),
- Enables unattended activities (overnight, long term)
- Allows to embed post-measurement processing (averaging calculation, plotting, etc),
- Provides error-free capture of data,
- Guarantees evenly time-distributed samples.

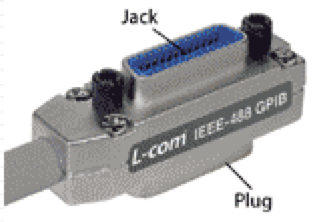
One measurement? Easy.
How about...
1000 measurements ?
Averaging, Std Deviation?
Long term drift?

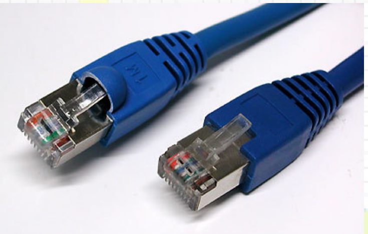
What Electrical Interface?

Faster

- GPIB (HP-IB)
  - 1-Controller : N-Device(s)
  - 8-bit parallel bus
  - Expensive interface and cables
- USB
  - 1-Controller : 1-Device
  - Low cost interface and cables
- Ethernet
  - 1-Controller : 1-Device
  - Low cost interface and cables





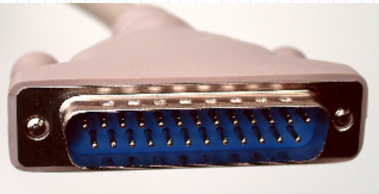


What Electrical Interface? (cont'ed)

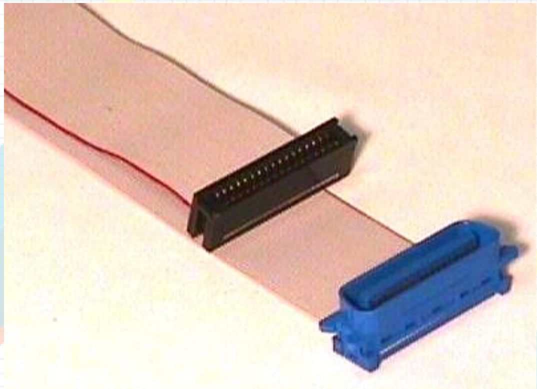
Slower

- Serial
  - 1-Controller : 1-Device
  - Low cost interface and cables

- Parallel (Printer)
  - 1-Controller : 1-Device
  - Low cost interface and cables

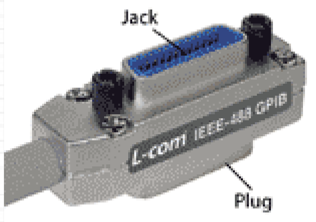


- Parallel (Custom)
  - 1-Controller : 1-Device
  - Low cost interface and cables
  - Complex interface and cabling?



What's GPIB?

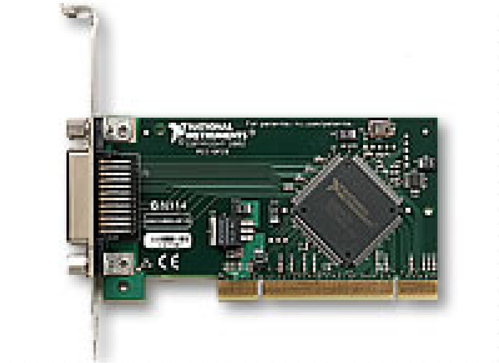
- Omnipresent, most test instruments offer it as option, was (and still is) standard on many,
- Other naming: HP-IB, IEEE-488,
- Still in use despite USB and Ethernet,



- 8-bit parallel bus, 3 handshake lines, five management lines,
- Truly designed with instrument control in mind (Trigger, Device Clear, Service Request),
  - Pros: Reliable, rugged, fast, adopted on a large scale,
  - Cons: Bulky, expensive,will surely become "passé", but when?

GPIB Controller– What solutions?

- PC Interface card
  - PCI expensive (>100$), ISA low cost (~20$)
  - Drivers required (OS-Dependent)



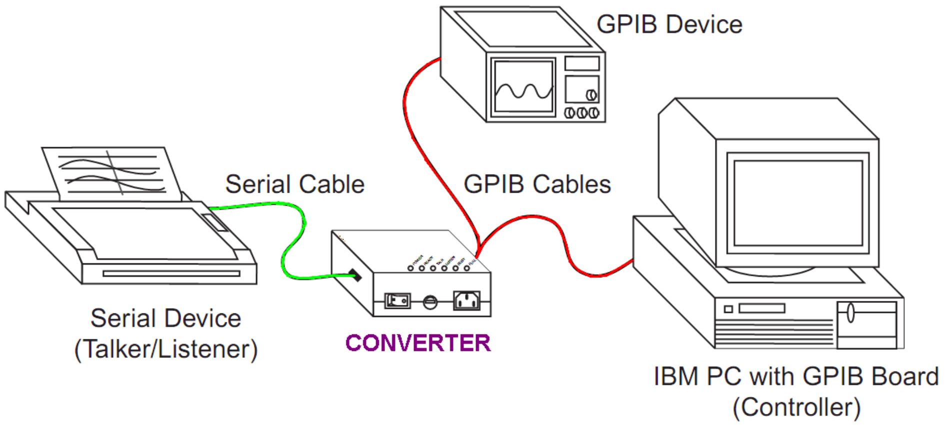
- USB-GPIB dongle
  - Expensive (>100$),
  - Drivers may be required

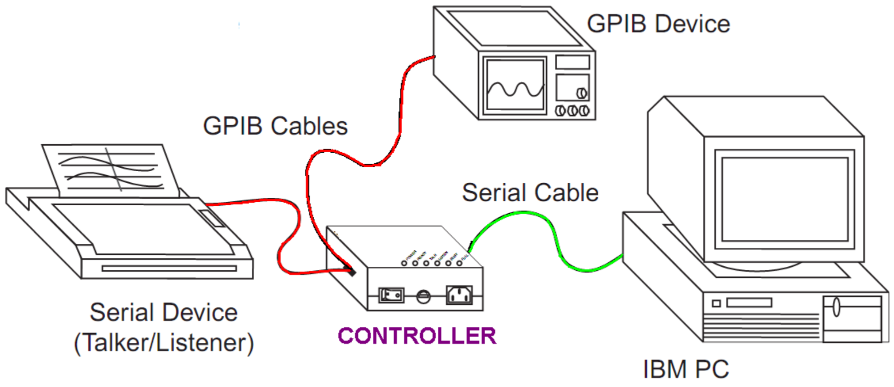
- Serial-GPIB Interface
  - Somewhat expensive (<100$)
  - Not limited by drivers (OS-independent)

- Ethernet-GPIB
  - Expensive (>100$), not common
  - Not limited by drivers (OS-independent)

Controller or Converter?

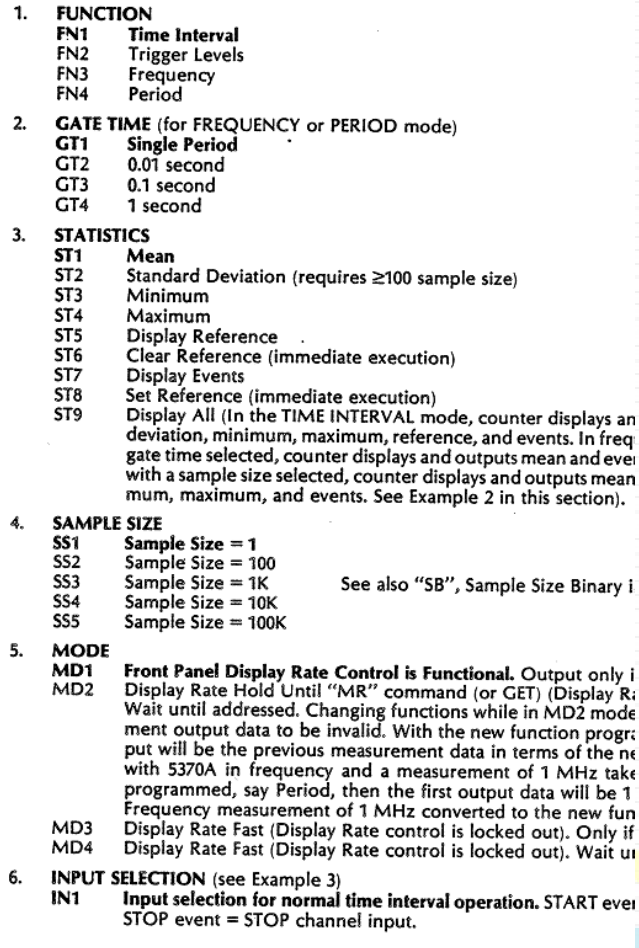
- Converter
  - Between GPIB controller and serial device (dumb),
  - Instrument or PC is controller, printer is slave,
  - Little or no data buffering,
  - Not useful for automation.
- Controller
  - Required for automation,
  - Controller is the master, all test instruments are slaves,
  - Has built in data buffering .





What Documentation to Move Forward?

- GPIB (or other) controller manual required for PC-to-interface command syntax. Mostly Free.
- User Manual for each instrument to control. Needed for Interface-to Instrument command syntax. Available online in .PDF for most instruments. Mostly Free.
- Built-in Help in most programming environments.
- Programming Language documentation and manuals also available online. Free.



1. **FUNCTION**
FN1 **Time Interval**
FN2 Trigger Levels
FN3 Frequency
FN4 Period

2. **GATE TIME** (for FREQUENCY or PERIOD mode)
GT1 **Single Period**
GT2 0.01 second
GT3 0.1 second
GT4 1 second

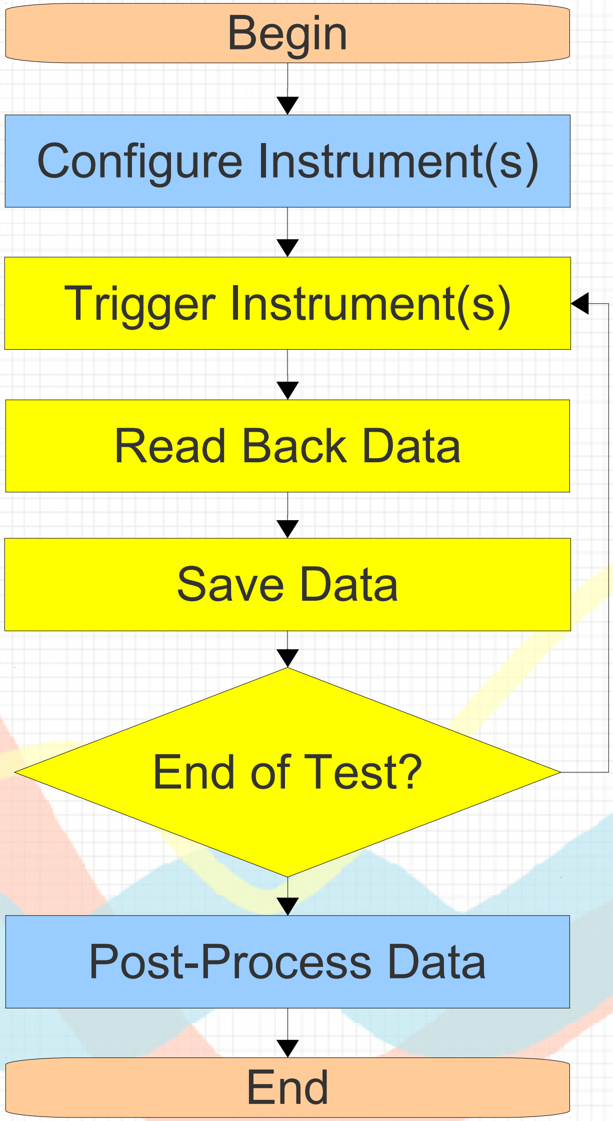
3. **STATISTICS**
ST1 **Mean**
ST2 Standard Deviation (requires ≥100 sample size)
ST3 Minimum
ST4 Maximum
ST5 Display Reference
ST6 Clear Reference (immediate execution)
ST7 Display Events
ST8 Set Reference (immediate execution)
ST9 Display All (In the TIME INTERVAL mode, counter displays an deviation, minimum, maximum, reference, and events. In freq gate time selected, counter displays and outputs mean and eve with a sample size selected, counter displays and outputs mean mum, maximum, and events. See Example 2 in this section).

4. **SAMPLE SIZE**
SS1 **Sample Size = 1**
SS2 Sample Size = 100
SS3 Sample Size = 1K See also "SB", Sample Size Binary i
SS4 Sample Size = 10K
SS5 Sample Size = 100K

5. **MODE**
MD1 **Front Panel Display Rate Control is Functional.** Output only i
MD2 Display Rate Hold Until "MR" command (or GET) (Display Ra Wait until addressed. Changing functions while in MD2 mode ment output data to be invalid. With the new function progra put will be the previous measurement data in terms of the ne with 5370A in frequency and a measurement of 1 MHz take programmed, say Period, then the first output data will be 1 Frequency measurement of 1 MHz converted to the new fun
MD3 Display Rate Fast (Display Rate control is locked out). Only if
MD4 Display Rate Fast (Display Rate control is locked out). Wait u

6. **INPUT SELECTION** (see Example 3)
IN1 **Input selection for normal time interval operation.** START eve STOP event = STOP channel input.

Typical Test Automation Cycle



Begin

Configure Instrument(s)

Trigger Instrument(s)

Read Back Data

Save Data

End of Test?

Post-Process Data

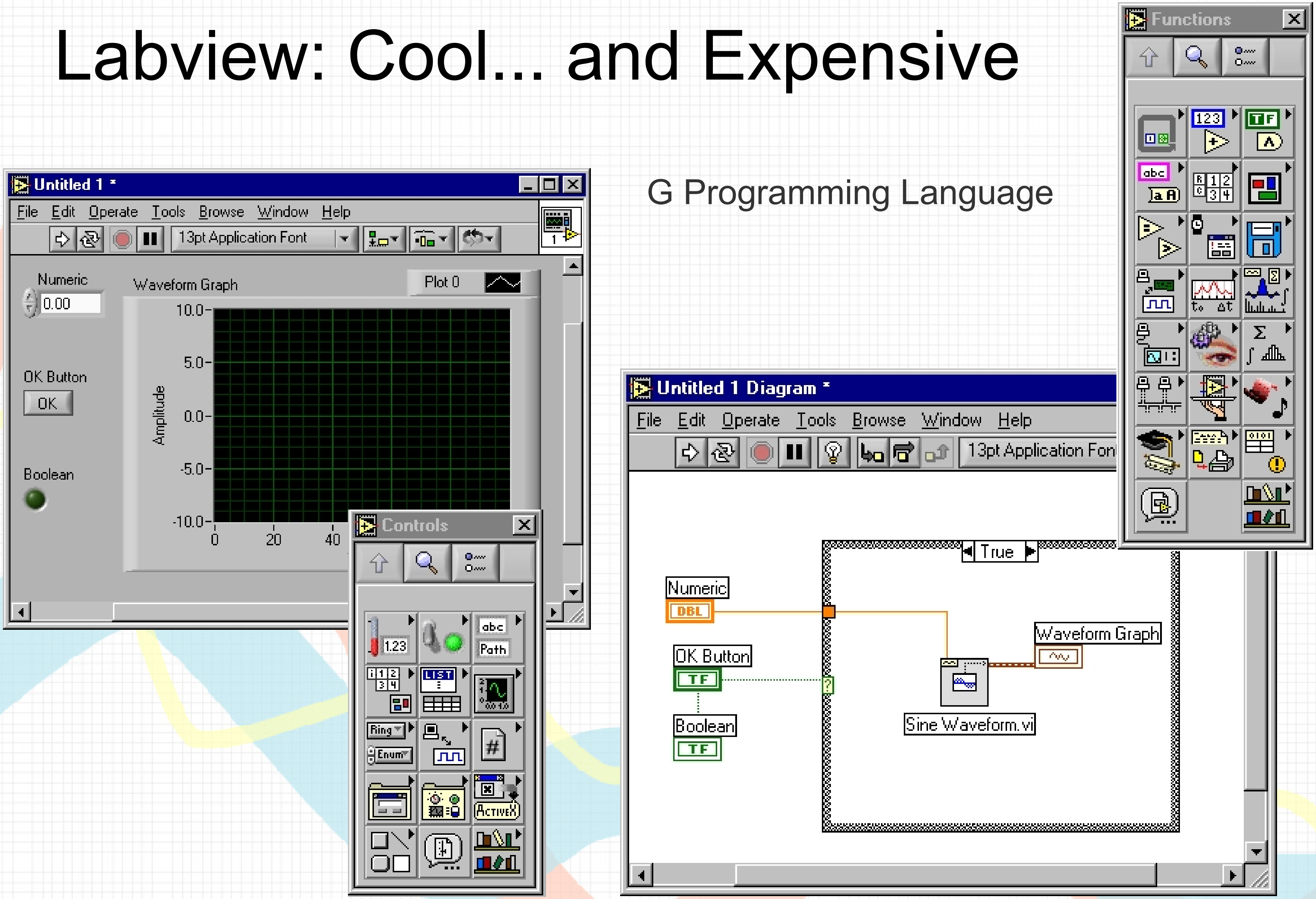
End

What Environments for Control?

- NI Labview (GUI) / LabWindows (C++)
  - Simple and very expensive. Windows and Linux.
- High Level Programming Languages with/without GUI
  - C,Pascal, Basic, Python, Fortran, etc
  - Free and somewhat more complex. Windows, Linux, Mac
- Agilent Vee (GUI) (HP-Vee obsolete)
  - Not mainstream, expensive. For Windows
- Matlab
  - Very complex (overkill) and very expensive...
- Clairsoft TestPad Development Studio v1.00
  - For Windows. Free. Worth investigating! In Ottawa!
  - Uses VISA layer. Has GUI controls. Basic.

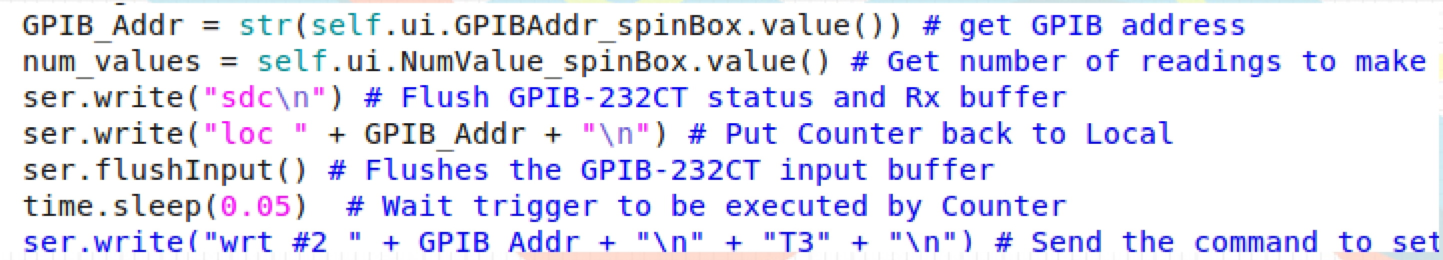
Labview: Cool… and Expensive

G Programming Language



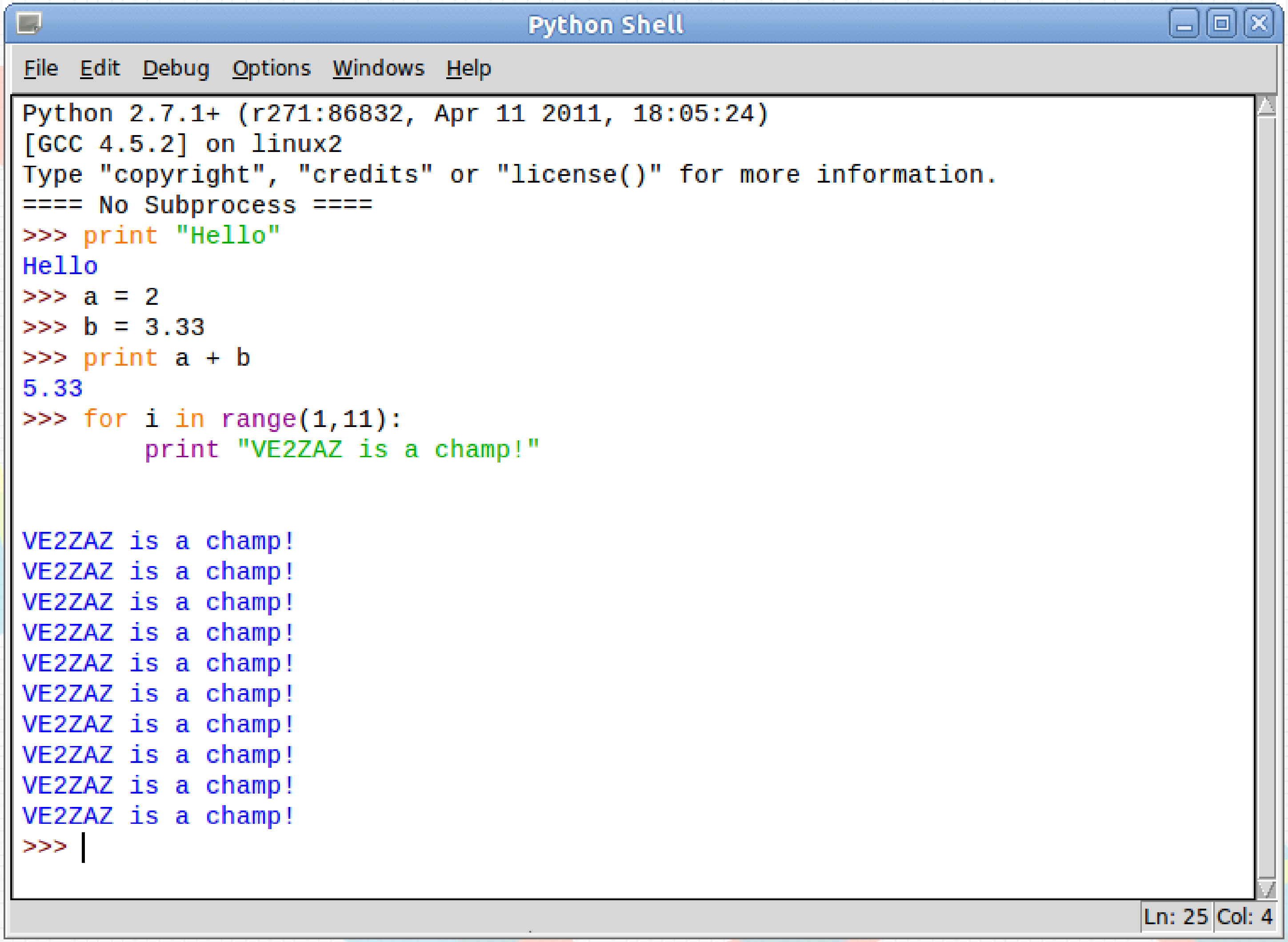
I now use Python. Why?

- Python is a scripting language.
  - Not compiled like C, Pascal, V-Basic...
  - Interpreted in real-time by Python engine.
  - Comes pre-installed into most Linux distros. Also available for Windows and Mac. → Easily Portable
  - Rather simple syntax. Reminds of old 1980’s Basic. Yet is powerful.
  - Complete set of instructions.
  - Libraries readily available: GPIB, serial port, plotting, Ethernet, FTP, HTTP, etc...
  - Free!



```
GPIB_Addr = str(self.ui.GPIBAddr_spinBox.value()) # get GPIB address
num_values = self.ui.NumValue_spinBox.value() # Get number of readings to make
ser.write("sdc\n") # Flush GPIB-232CT status and Rx buffer
ser.write("loc " + GPIB_Addr + "\n") # Put Counter back to Local
ser.flushInput() # Flushes the GPIB-232CT input buffer
time.sleep(0.05)  # Wait trigger to be executed by Counter
ser.write("wrt #2 " + GPIB_Addr + "\n" + "T3" + "\n") # Send the command to set
```

The Python Shell

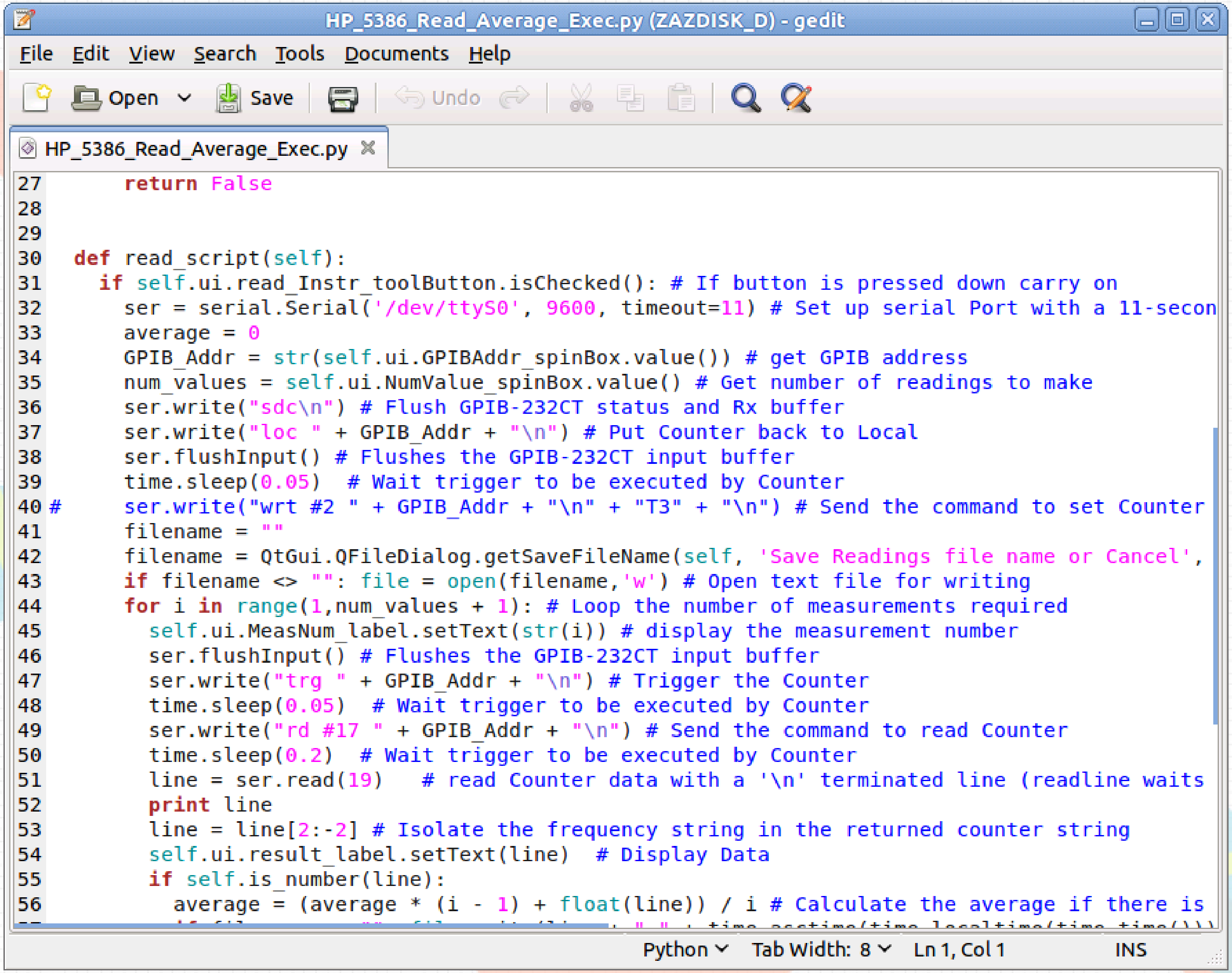


```
Python 2.7.1+ (r271:86832, Apr 11 2011, 18:05:24)
[GCC 4.5.2] on linux2
Type "copyright", "credits" or "license()" for more information.
==== No Subprocess ====
>>> print "Hello"
Hello
>>> a = 2
>>> b = 3.33
>>> print a + b
5.33
>>> for i in range(1,11):
        print "VE2ZAZ is a champ!"


VE2ZAZ is a champ!
VE2ZAZ is a champ!
VE2ZAZ is a champ!
VE2ZAZ is a champ!
VE2ZAZ is a champ!
VE2ZAZ is a champ!
VE2ZAZ is a champ!
VE2ZAZ is a champ!
VE2ZAZ is a champ!
VE2ZAZ is a champ!
>>> 
```

Reminds us of 1980’s Basic on C64, VIC-20 and TRS-80!...

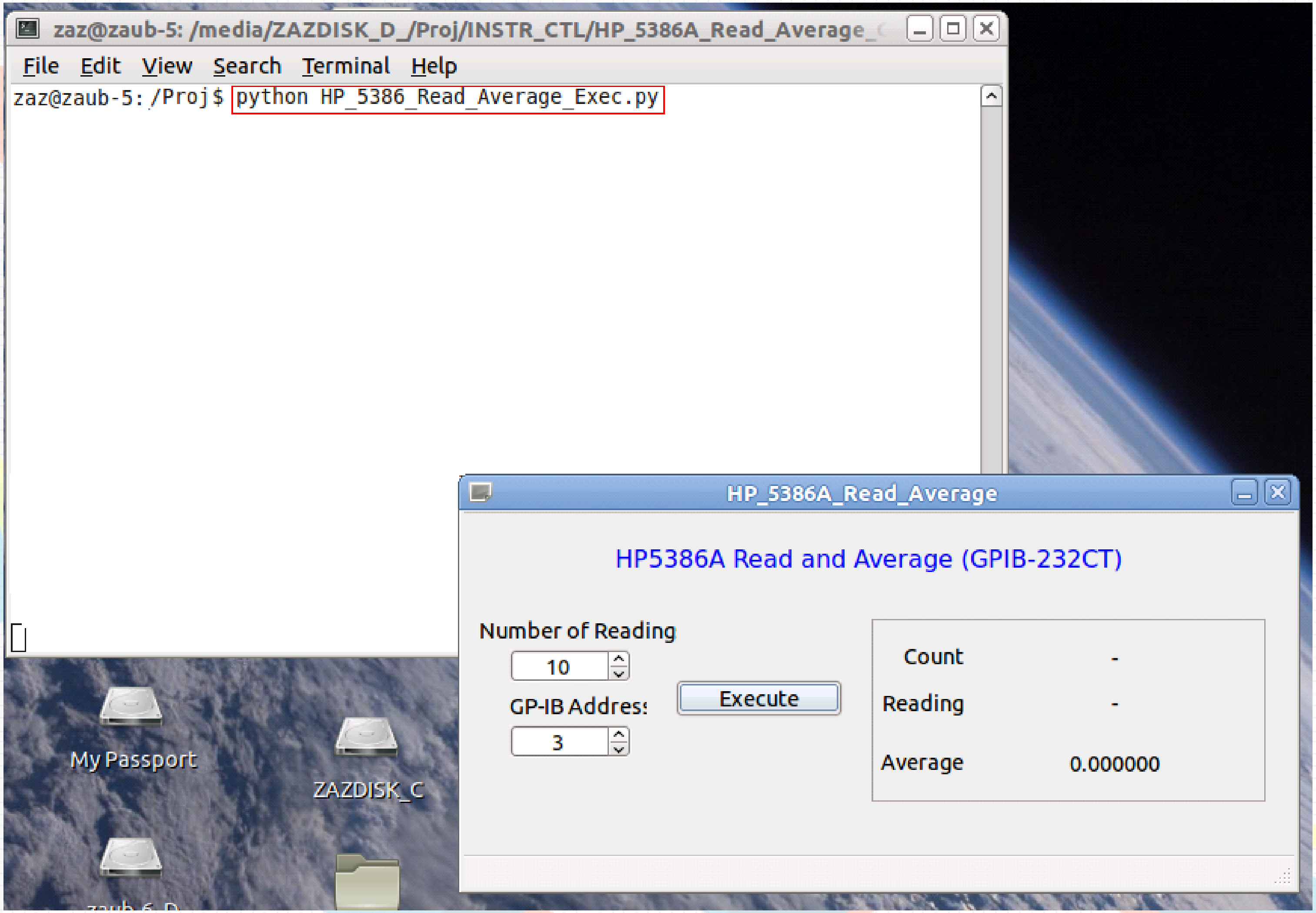
The Python Script...



```python
   return False


 def read_script(self):
  if self.ui.read_Instr_toolButton.isChecked(): # If button is pressed down carry on
   ser = serial.Serial('/dev/ttyS0', 9600, timeout=11) # Set up serial Port with a 11-secon
   average = 0
   GPIB_Addr = str(self.ui.GPIBAddr_spinBox.value()) # get GPIB address
   num_values = self.ui.NumValue_spinBox.value() # Get number of readings to make
   ser.write("sdc\n") # Flush GPIB-232CT status and Rx buffer
   ser.write("loc " + GPIB_Addr + "\n") # Put Counter back to Local
   ser.flushInput() # Flushes the GPIB-232CT input buffer
   time.sleep(0.05)  # Wait trigger to be executed by Counter
#  ser.write("wrt #2 " + GPIB_Addr + "\n" + "T3" + "\n") # Send the command to set Counter
   filename = ""
   filename = QtGui.QFileDialog.getSaveFileName(self, 'Save Readings file name or Cancel',
   if filename <> "": file = open(filename,'w') # Open text file for writing
   for i in range(1,num_values + 1): # Loop the number of measurements required
    self.ui.MeasNum_label.setText(str(i)) # display the measurement number
    ser.flushInput() # Flushes the GPIB-232CT input buffer
    ser.write("trg " + GPIB_Addr + "\n") # Trigger the Counter
    time.sleep(0.05)  # Wait trigger to be executed by Counter
    ser.write("rd #17 " + GPIB_Addr + "\n") # Send the command to read Counter
    time.sleep(0.2)  # Wait trigger to be executed by Counter
    line = ser.read(19)   # read Counter data with a '\n' terminated line (readline waits
    print line
    line = line[2:-2] # Isolate the frequency string in the returned counter string
    self.ui.result_label.setText(line)  # Display Data
    if self.is_number(line):
     average = (average * (i - 1) + float(line)) / i # Calculate the average if there is
```

Python Invoked From Command Line



zaz@zaub-5: /media/ZAZDISK_D_/Proj/INSTR_CTL/HP_5386A_Read_Average_
File Edit View Search Terminal Help
HP_5386A_Read_Average
Execute
My Passport
ZAZDISK_C

```
zaz@zaub-5: /Proj$ python HP_5386_Read_Average_Exec.py
```

HP5386A Read and Average (GPIB-232CT)

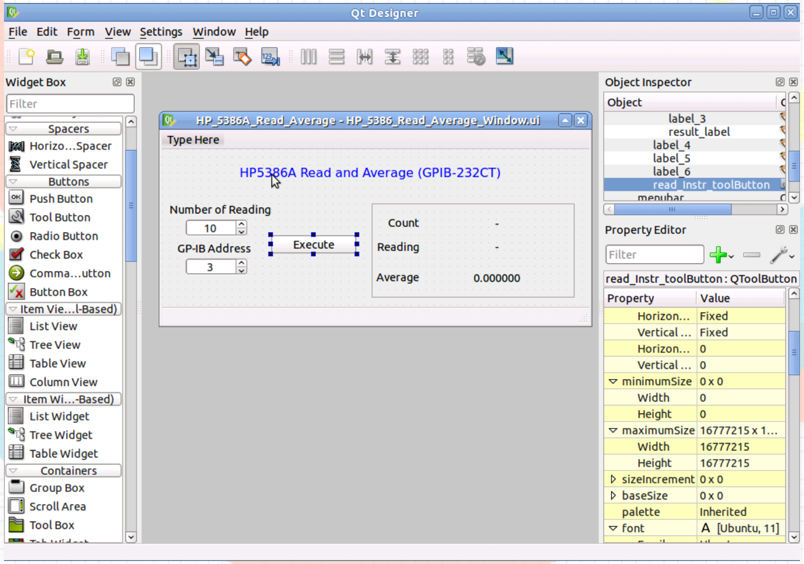
Number of Reading: 10

GP-IB Addres: 3

```
Count      -
Reading    -
Average    0.000000
```

Create your Windows with QT4 Designer

- Member of the QT family of S/W development tools
- Cross-platform GUI layout and forms builder. Allows to design and build widgets and dialogs using on-screen forms.
- Forms created with Qt4 Designer are fully-functional, and they can be previewed.



HP5386A Read and Average (GPIB-232CT)

Number of Reading: 10

GP-IB Address: 3

Count -

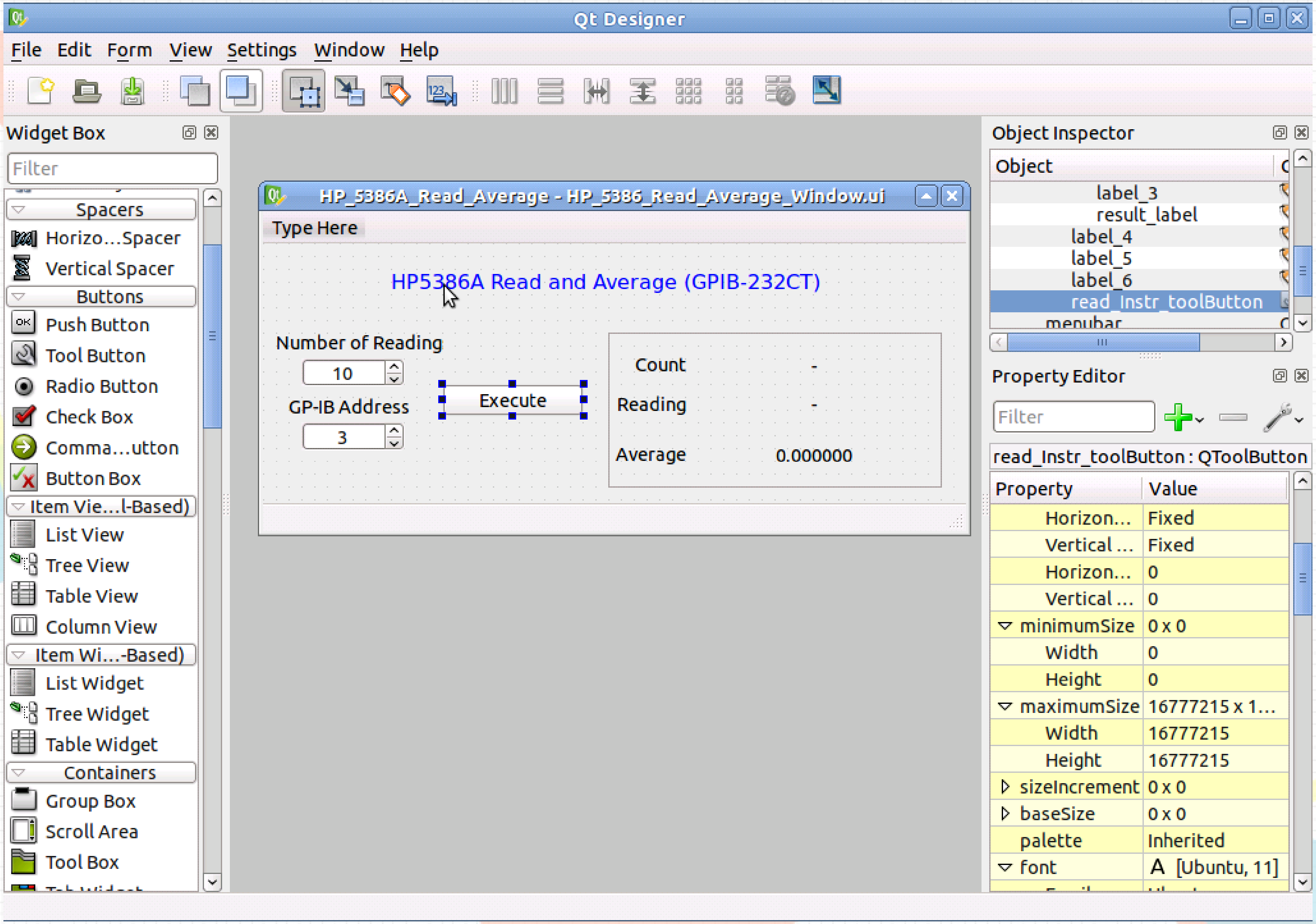
Reading -

Average 0.000000

read_Instr_toolButton : QToolButton

Property	Value
Horizon...	Fixed
Vertical ...	Fixed
Horizon...	0
Vertical ...	0
▽ minimumSize	0 x 0
Width	0
Height	0
▽ maximumSize	16777215 x 1...
Width	16777215
Height	16777215
▷ sizeIncrement	0 x 0
▷ baseSize	0 x 0
palette	Inherited
▽ font	A [Ubuntu, 11]

Create a Form, Place Widgets



HP5386A Read and Average (GPIB-232CT)

Number of Reading: 10

GP-IB Address: 3

Execute

Count -

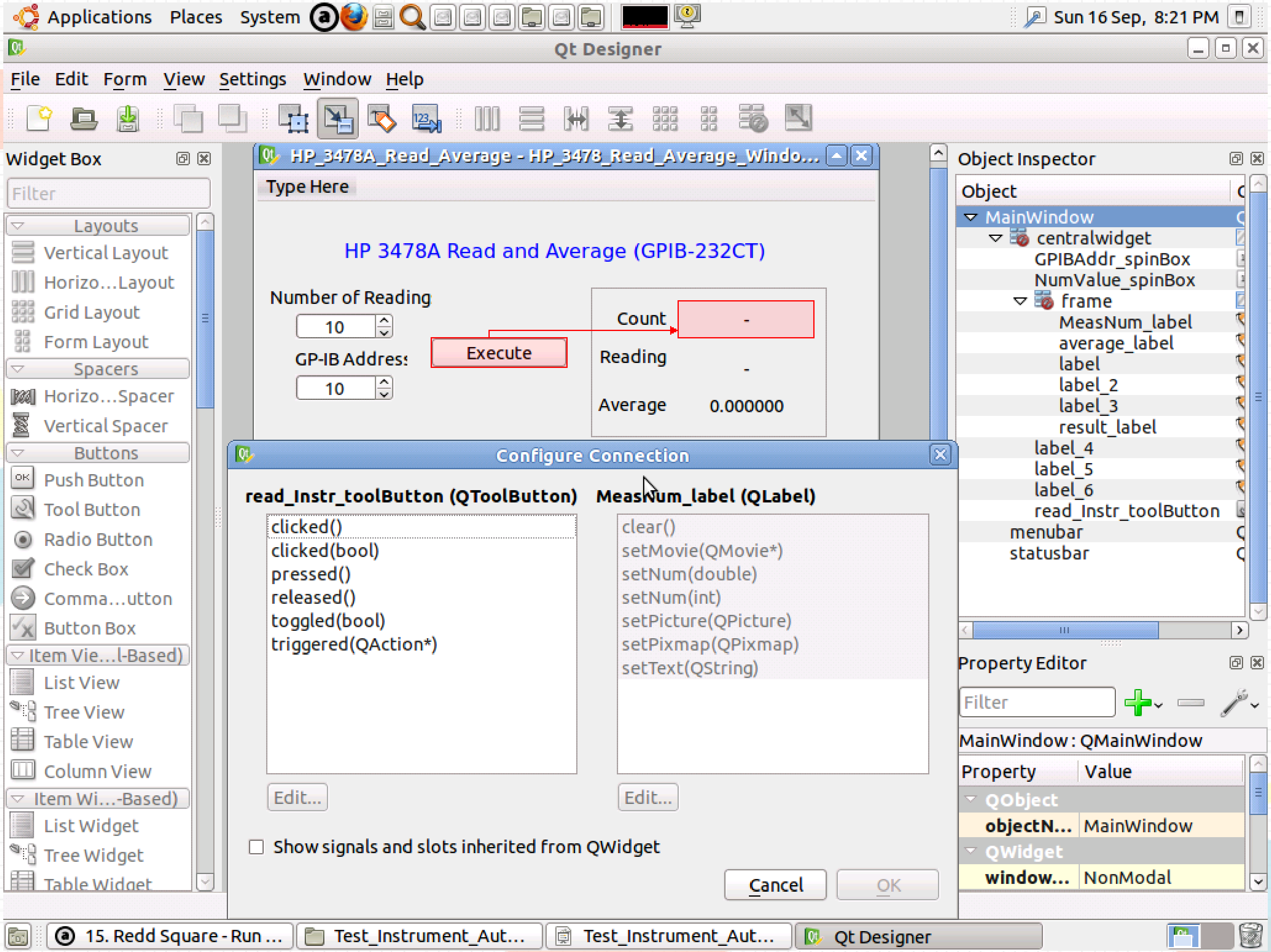
Reading -

Average 0.000000

read_Instr_toolButton : QToolButton

Property	Value
Horizon…	Fixed
Vertical …	Fixed
Horizon…	0
Vertical …	0
minimumSize	0 x 0
Width	0
Height	0
maximumSize	16777215 x 1…
Width	16777215
Height	16777215
sizeIncrement	0 x 0
baseSize	0 x 0
palette	Inherited
font	A [Ubuntu, 11]

Associate Events to Actions



Configure Connection

read_Instr_toolButton (QToolButton)	MeasNum_label (QLabel)
clicked()	clear()
clicked(bool)	setMovie(QMovie*)
pressed()	setNum(double)
released()	setNum(int)
toggled(bool)	setPicture(QPicture)
triggered(QAction*)	setPixmap(QPixmap)
	setText(QString)

Link the QT4 Window Design Into Your Python Script

- Use the PyQt4 set of Python bindings to integrate Qt4 designs into Python. Free.
  - Use “pyuic4” command to translate the Qt4 .ui (XML) window design file into a .py Python script file.
  - “Include” the resulting window .py script file into your .py code.
  - Will create the window at execution time.
- In your Python script, you refer to window widgets by their name for updates, display or refreshes.

Automation with the Raspberry Pi ver-B

- CPU powerful enough to fulfill any automation task.
- Raspbian “wheezy” distro is close enough to Debian and Ubuntu
  - A complete compatibility of the Python test code to the Raspberry Pi.
- Ideal for long term testing (low power, stable platform, independent from any PC)
- Remote desktop control ideal (VNC, SSH, etc)
- Serial Port device definition may differ: “ttyUSB0” vs. “ttyS0”

References

- Serial – GPIB Controllers
  - National Instruments GPIB-232CT, GPIB-232CT-A
  - IOTech Micro488EX
- Labview – National Instruments
  - http://www.ni.com/labview
- Python
  - http://www.python.org/
- PySerial
  - pyserial.sourceforge.net
- Qt4 Designer
  - http://doc.qt.digia.com/4.5/designer-manual.html
  - Py
- Raspberry Pi
  - http://www.raspberrypi.org/
- Clairsoft Test Automation & Measurement software
  - www.clairsoft.com/

Thanks!