Pierre-François Duc & Benjamin Schmidt

# *LabGui*

or automating measurements with Python

**Manual measurements**

**LabView**

**Python**

RTech www.facebook.com/RTechMcGill www.hep.physics.mcgill.ca/RTech

# *Automating your experiment with LabGui*

## May 19, Tuesday, noon

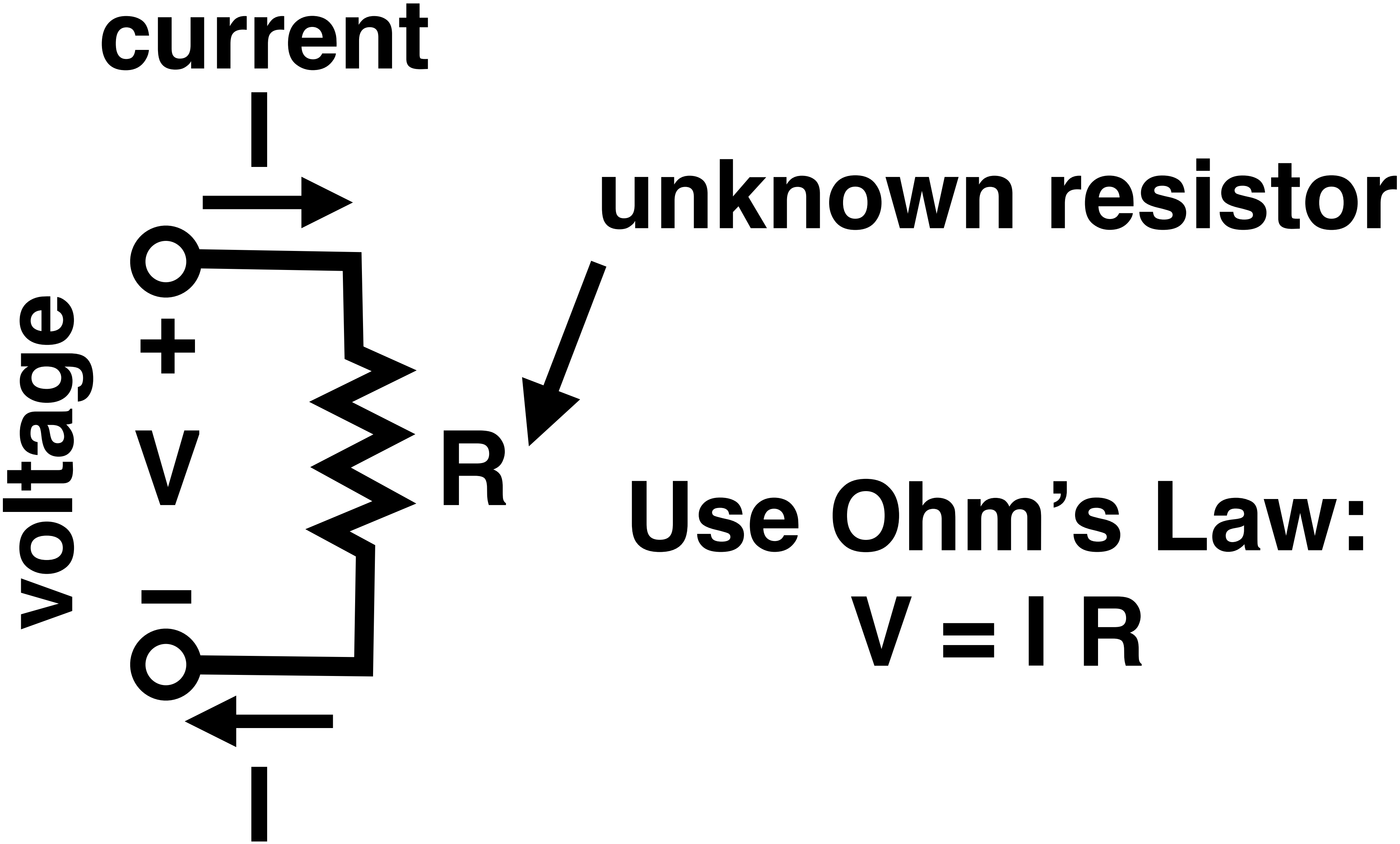
**3600 University, Rutherford Physics Building, room 103**

**We have funds to help you cover transportation costs. If interested contact us at:**

**rtech@physics.mcgill.ca**

**https://www.facebook.com/RTechMcGill**

# Determine resistance



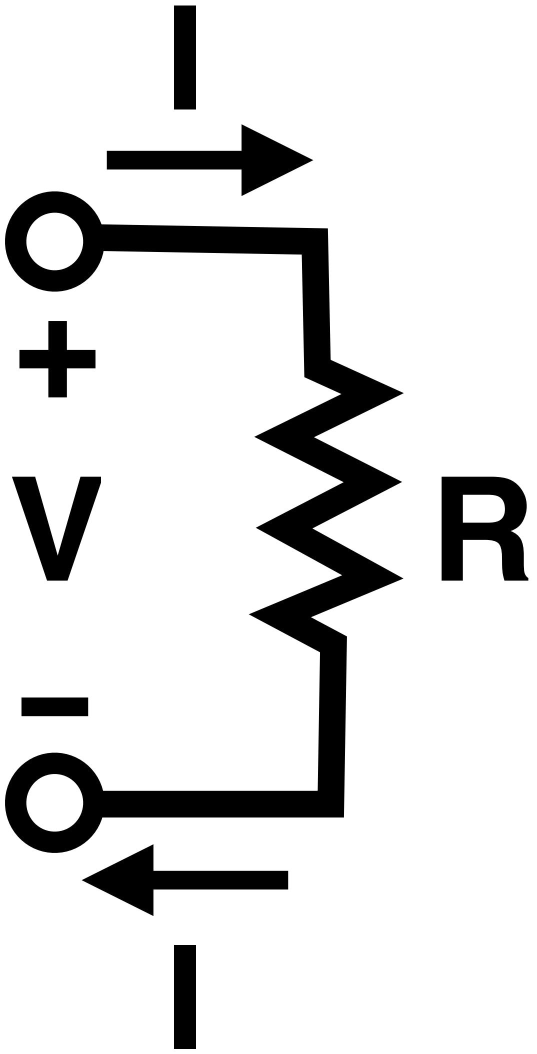
Use Ohm’s Law:

$V = I R$

# Ohm’s Law example: manual

Start measurement #1

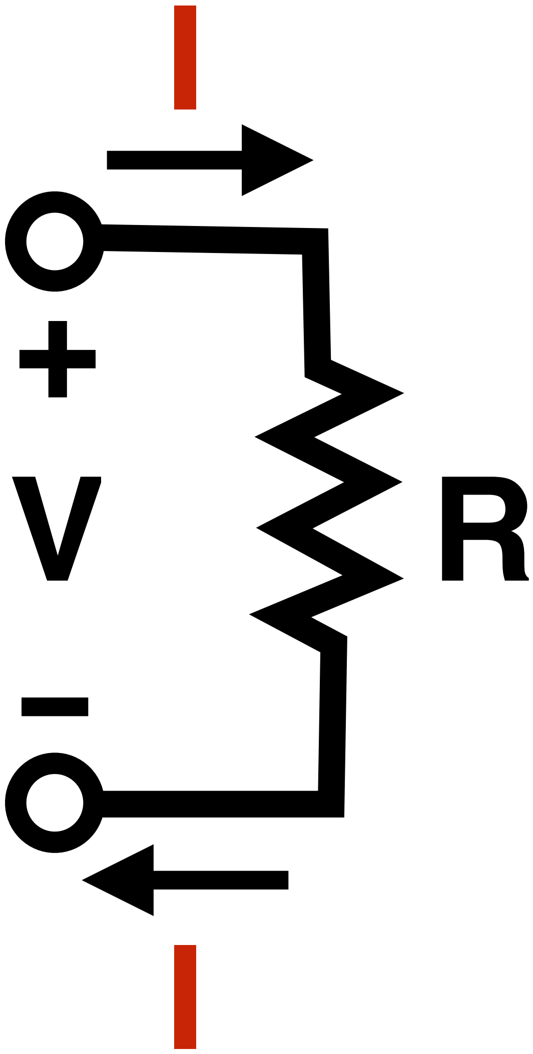
| # | I | V |
|---|---|---|
| 1 | | |



# Ohm’s Law example: manual

Set current value I to 1A

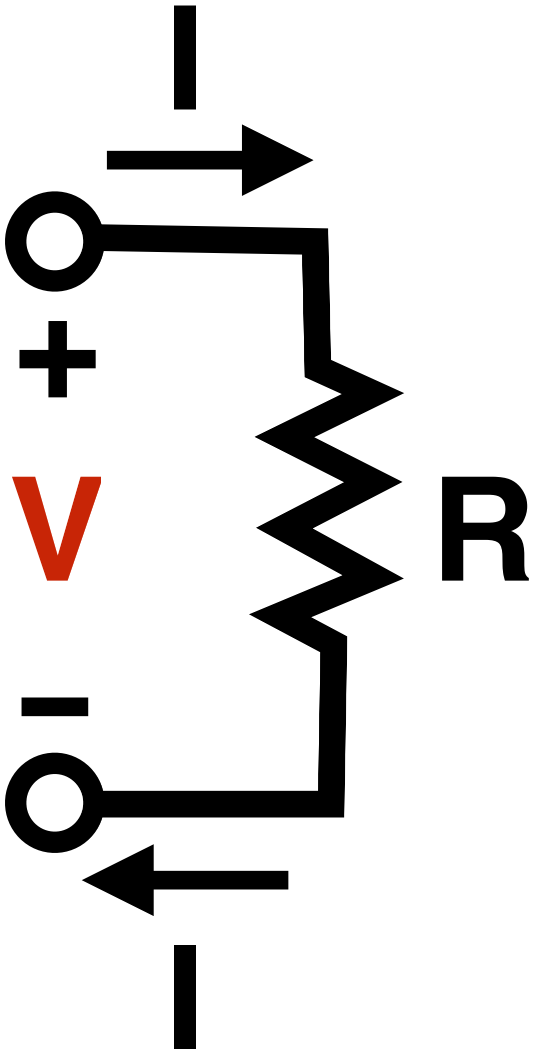
| # | I | V |
|---|---|---|
| 1 | 1.0 |  |



# Ohm’s Law example: manual

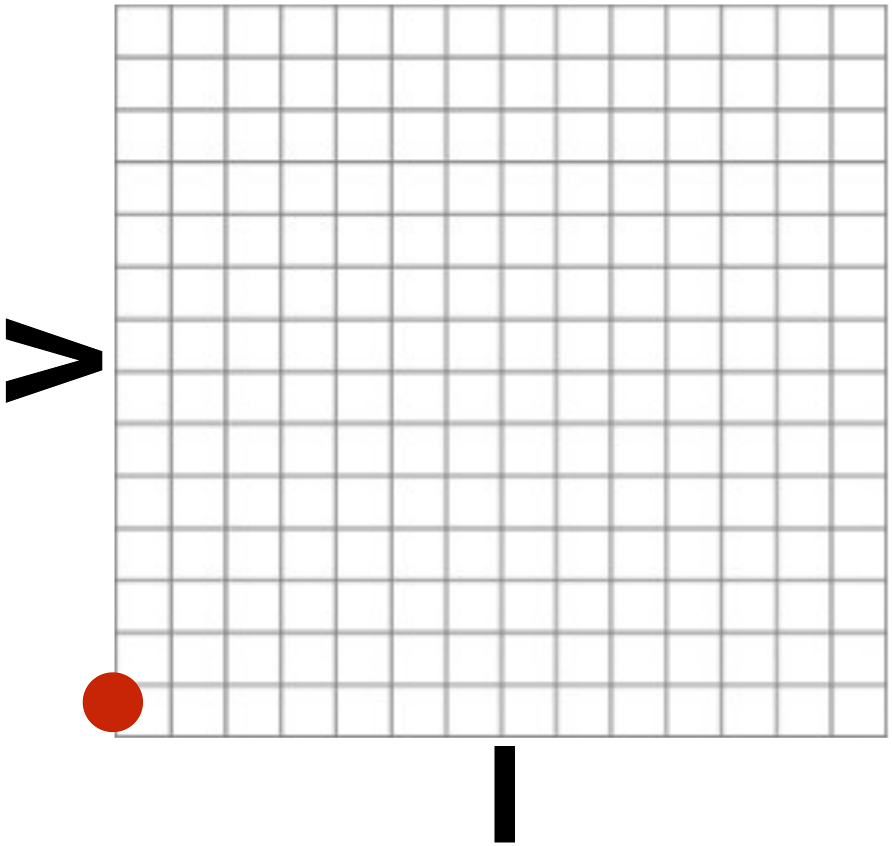
Measure voltage **V**

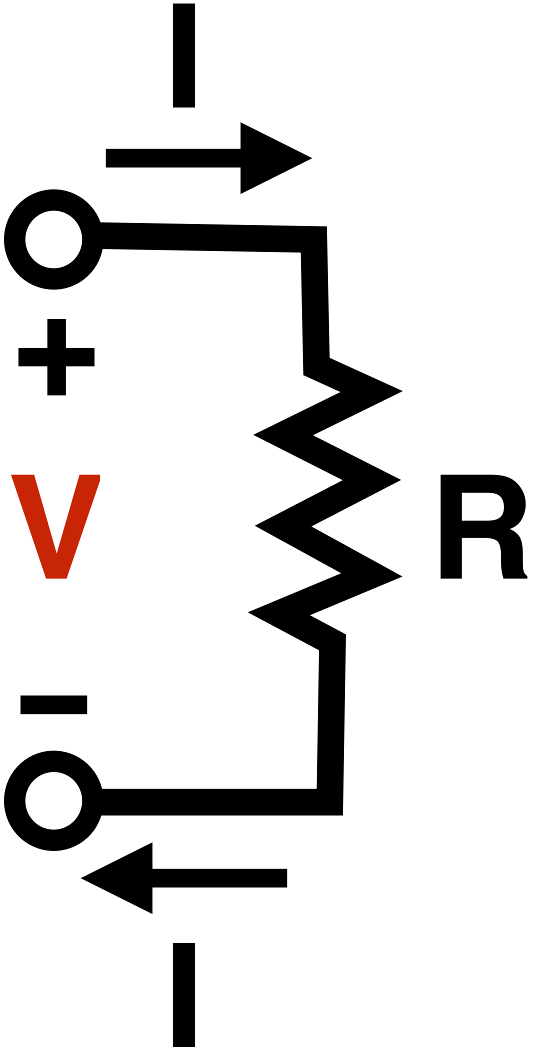
| # | I | V |
|---|---|---|
| 1 | 1.0 | 10.3 |



# Ohm’s Law example: manual

Plot your first point

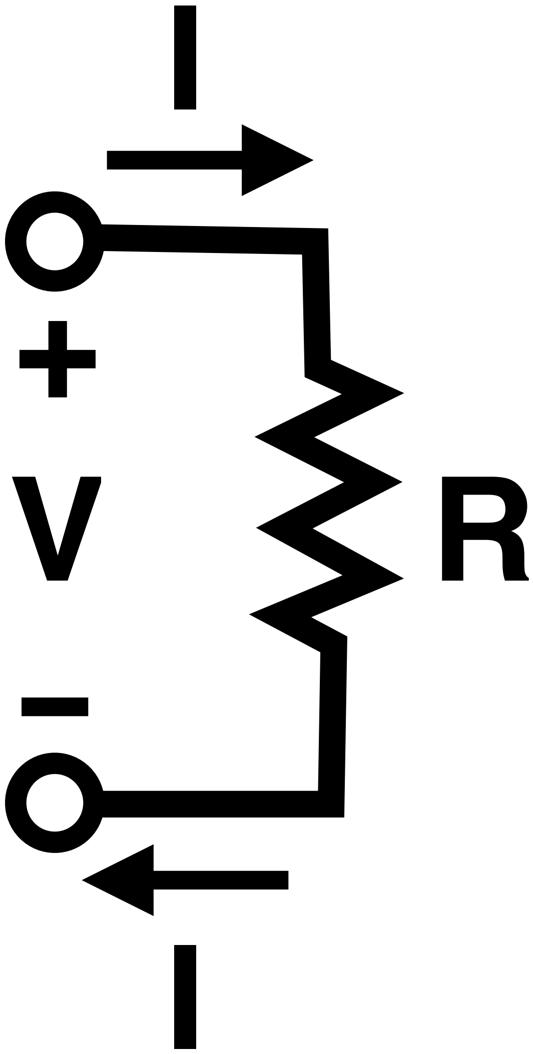




# Ohm’s Law example: manual

Start measurement #2

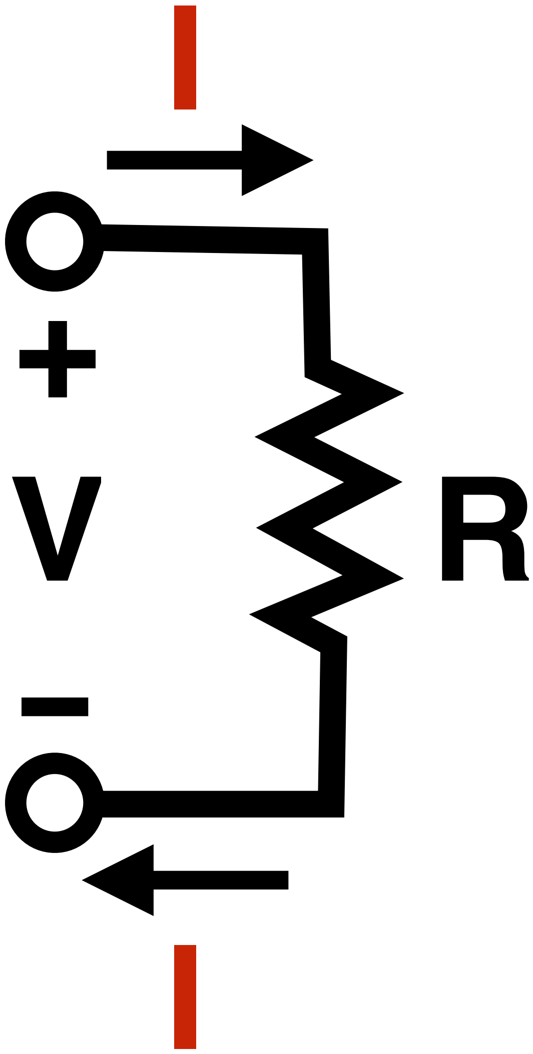
| # | I | V |
|---|---|---|
| 1 | 1.0 | 10.3 |
| 2 | | |



# Ohm’s Law example: manual

Set current value I to 2A

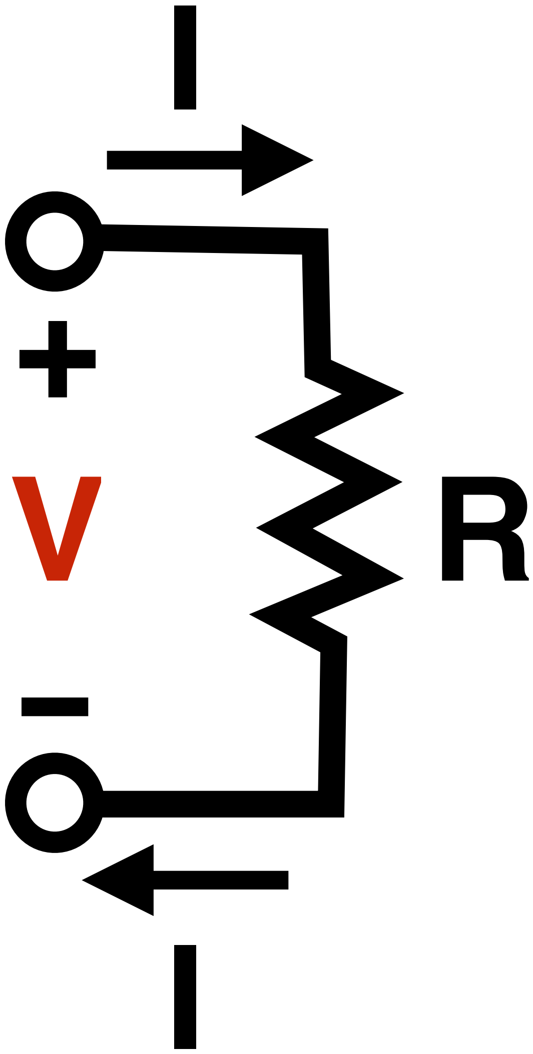
| # | I | V |
|---|---|---|
| 1 | 1.0 | 10.3 |
| 2 | 2.0 | |



# Ohm’s Law example: manual

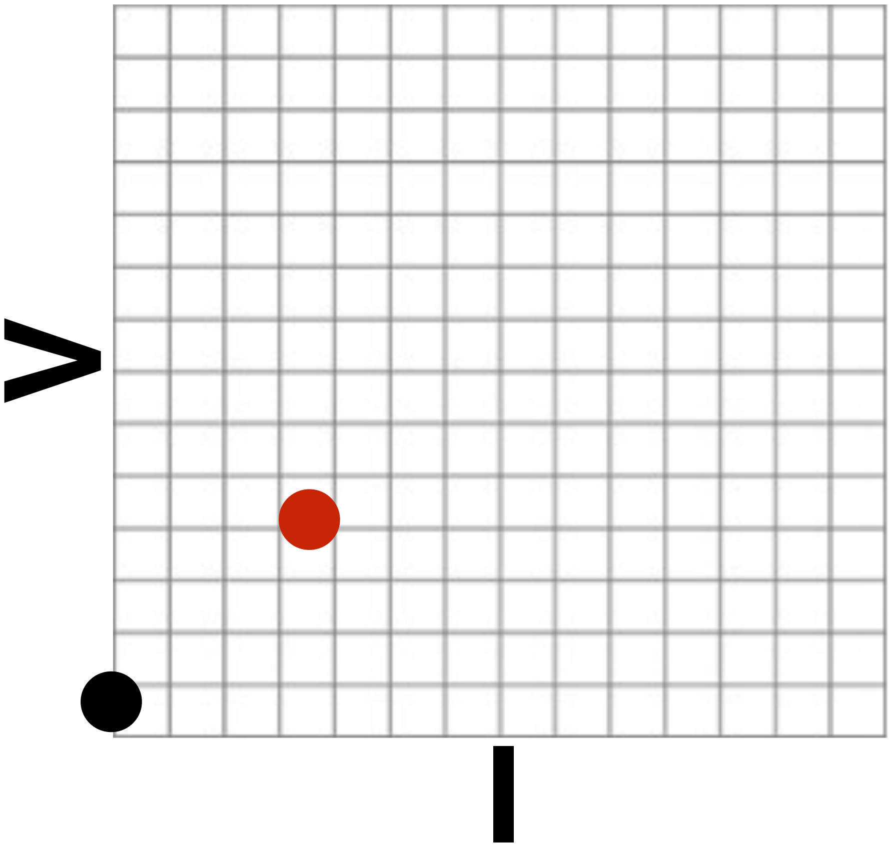
Measure voltage V

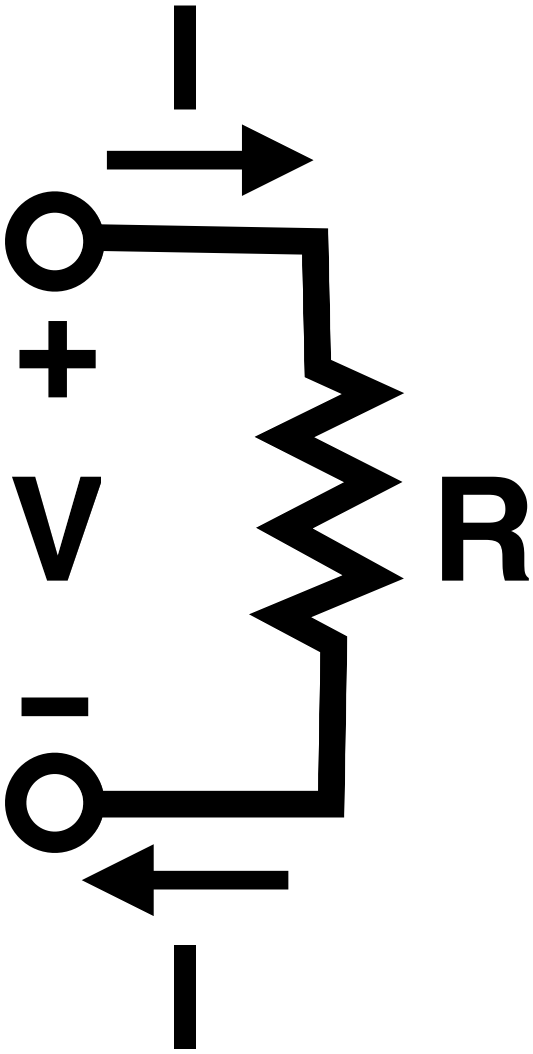
| # | I | V |
|---|---|---|
| 1 | 1.0 | 10.3 |
| 2 | 2.0 | 21.3 |



# Ohm’s Law example: manual

Plot your second point





# Ohm’s Law example: manual

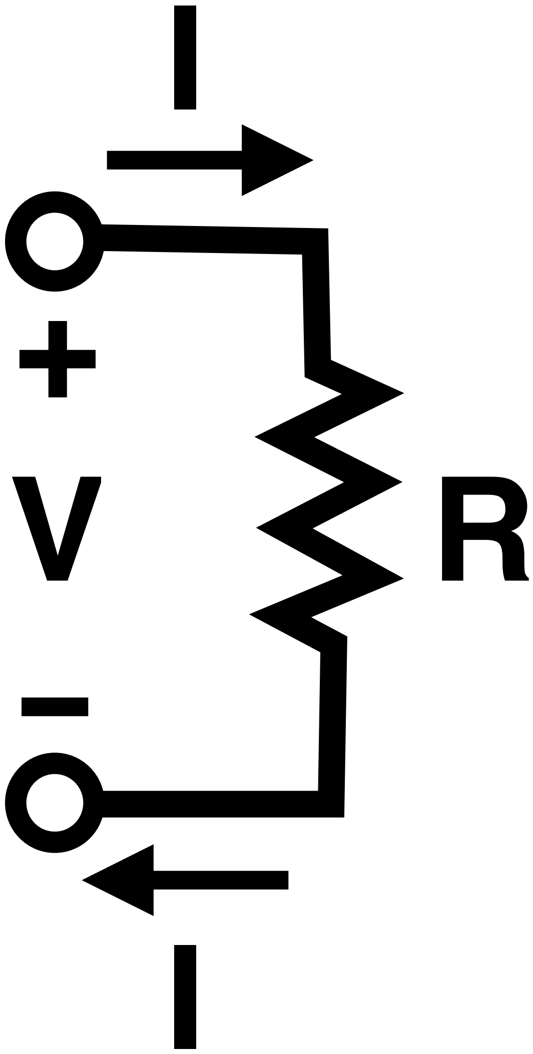
How many measurements
you say one needs to do
to reduce statistical uncertainty?

**so, continue…**

# Ohm’s Law example: manual

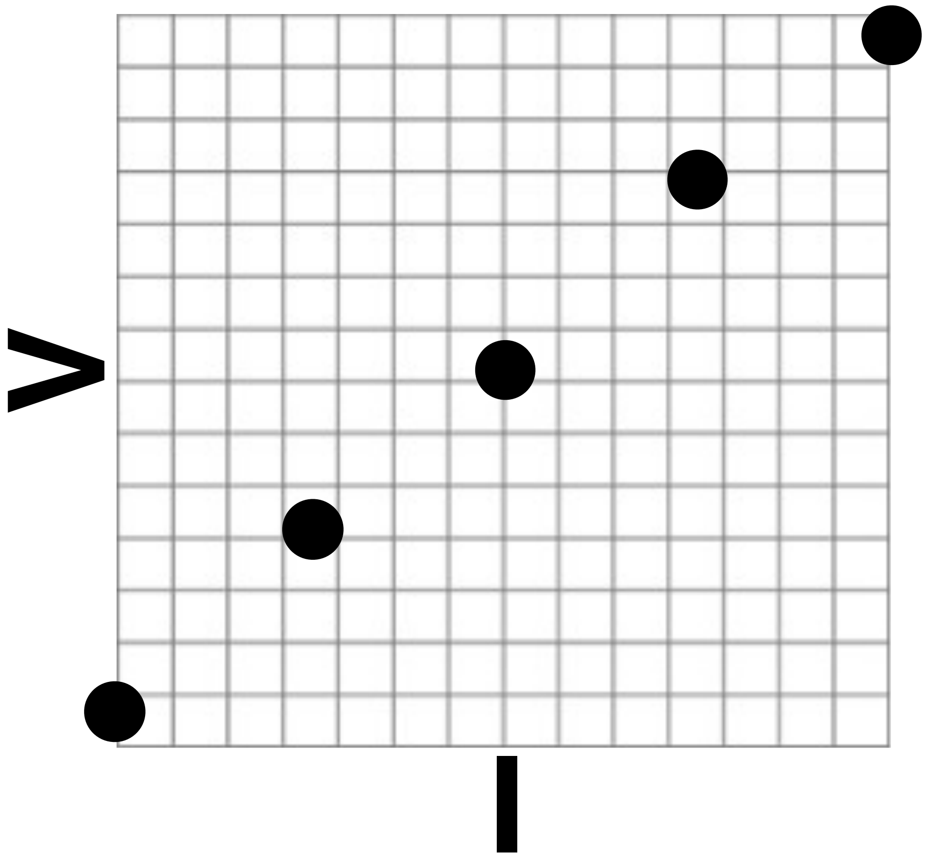
And here is your data

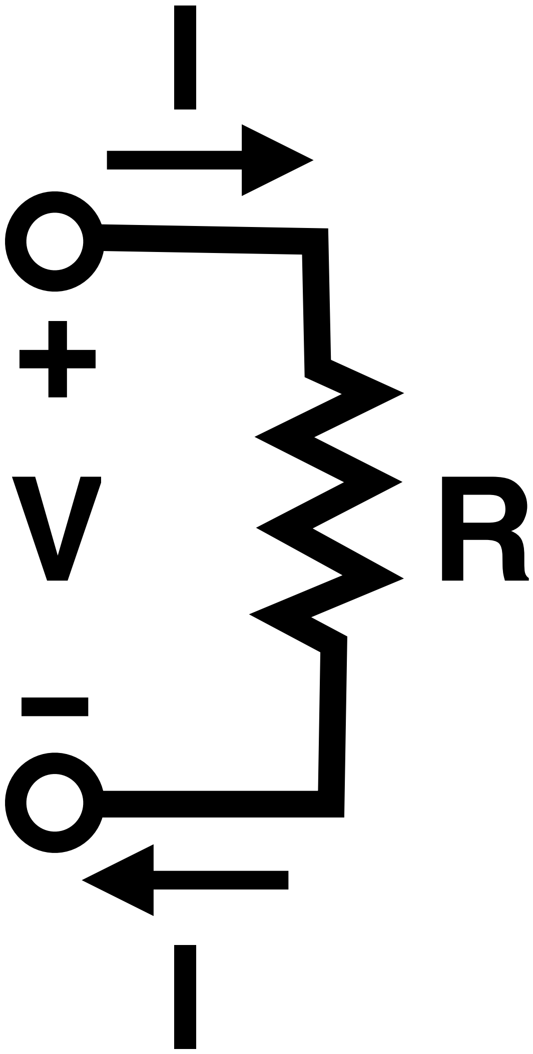
| # | I | V |
|---|---|---|
| 1 | 1.0 | 10.3 |
| 2 | 2.0 | 21.3 |
| 3 | 3.0 | 29.9 |
| 4 | 4.0 | 39.5 |
| 5 | 5.0 | 51.2 |



# Ohm’s Law example: manual

And here is your plot



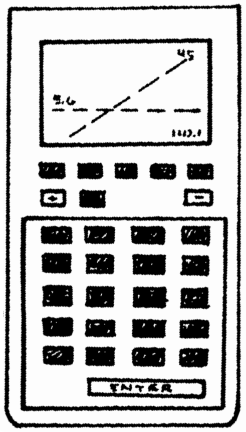


# Ohm’s Law example: manual

Linear regression **V = I R**

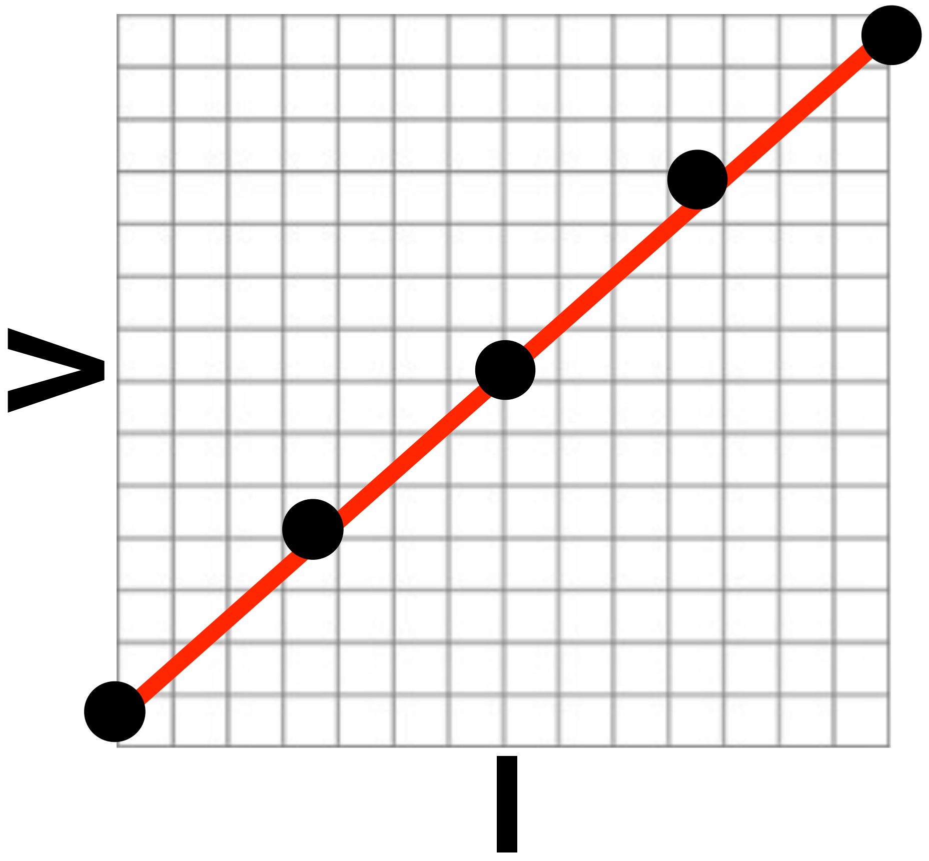
$$\mathbf{y} = \mathbf{X}\beta$$

$$\mathbf{y} = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{pmatrix}, \quad \mathbf{X} = \begin{pmatrix} \mathbf{x}_1^{\mathrm{T}} \\ \mathbf{x}_2^{\mathrm{T}} \\ \vdots \\ \mathbf{x}_n^{\mathrm{T}} \end{pmatrix} = \begin{pmatrix} x_{11} & \cdots & x_{1p} \\ x_{21} & \cdots & x_{2p} \\ \vdots & \ddots & \vdots \\ x_{n1} & \cdots & x_{np} \end{pmatrix}, \quad \boldsymbol{\beta} = \begin{pmatrix} \beta_1 \\ \beta_2 \\ \vdots \\ \beta_p \end{pmatrix}$$



# Ohm’s Law example: manual

And… Here you are!



$$V = I\,R$$

$$\boxed{R=10.12}$$

# Ohm’s Law example: manual

Is it a job for a student?

# Ohm’s Law example: manual

Only two options left

Manual measurements

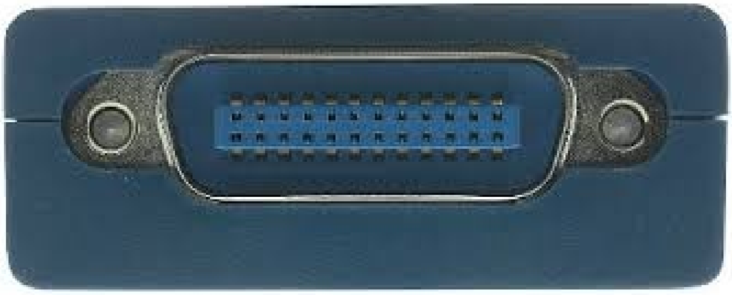
LabView

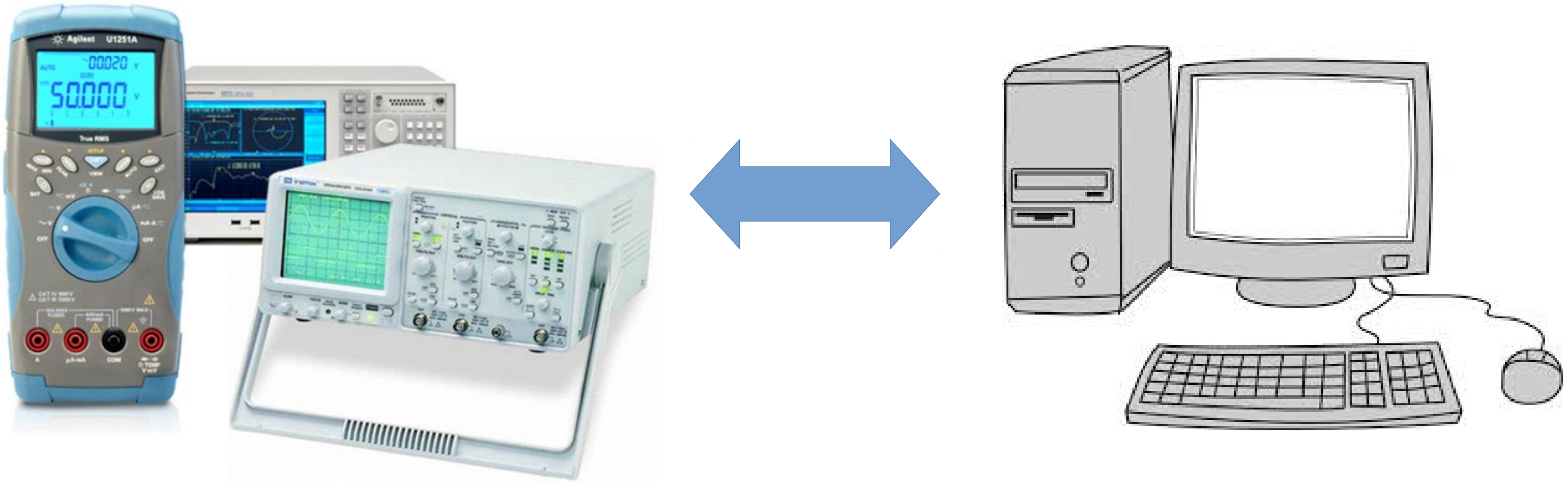
Python / LabGui

# Communicate with the device

Ethernet

GPIB





RS232

USB

# Ohm’s Law example

**LabView** or **LabGui**

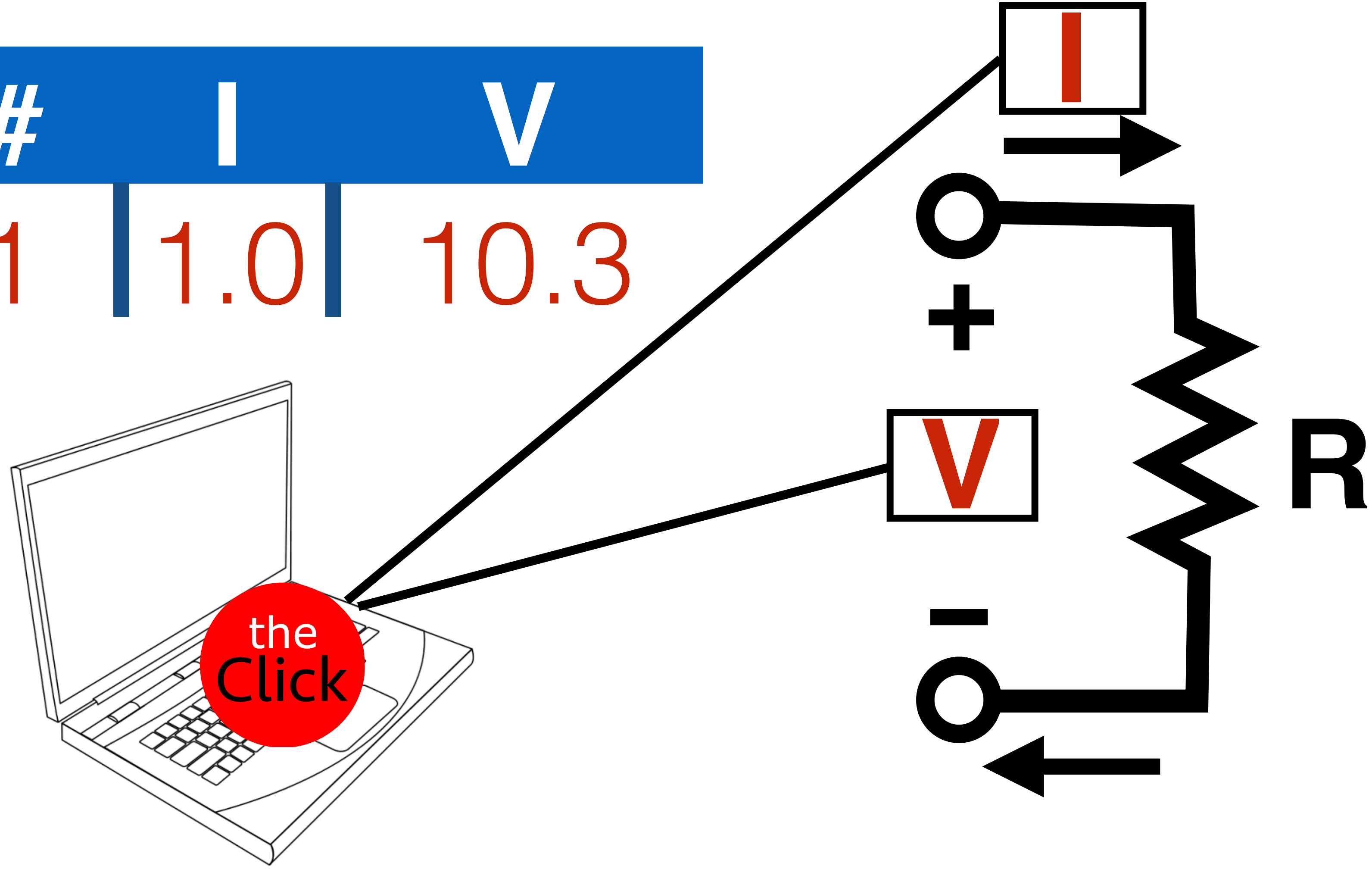
Let’s see how

**LabGui** performs

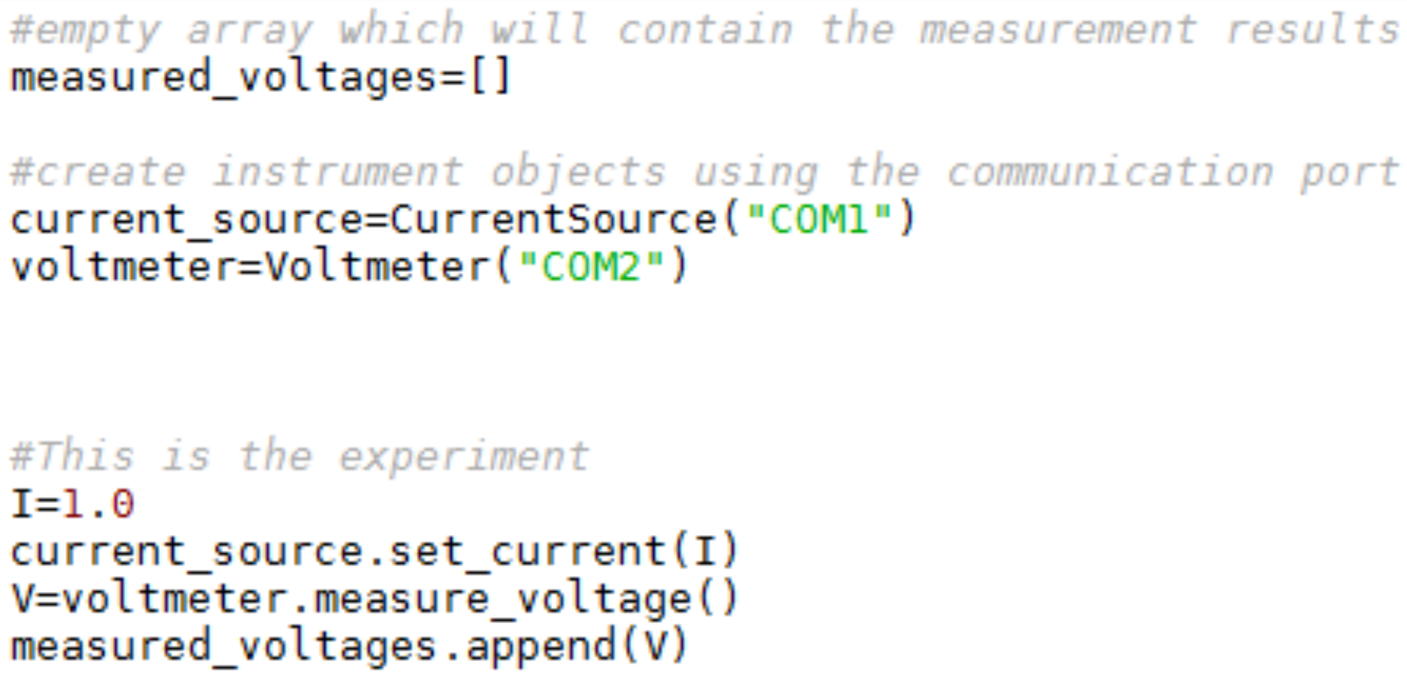
# Ohm’s Law example: LabGui

Set & Measure with a **Click**

| # | I | V |
|---|---|---|
| 1 | 1.0 | 10.3 |



# Ohm’s Law example: LabGui

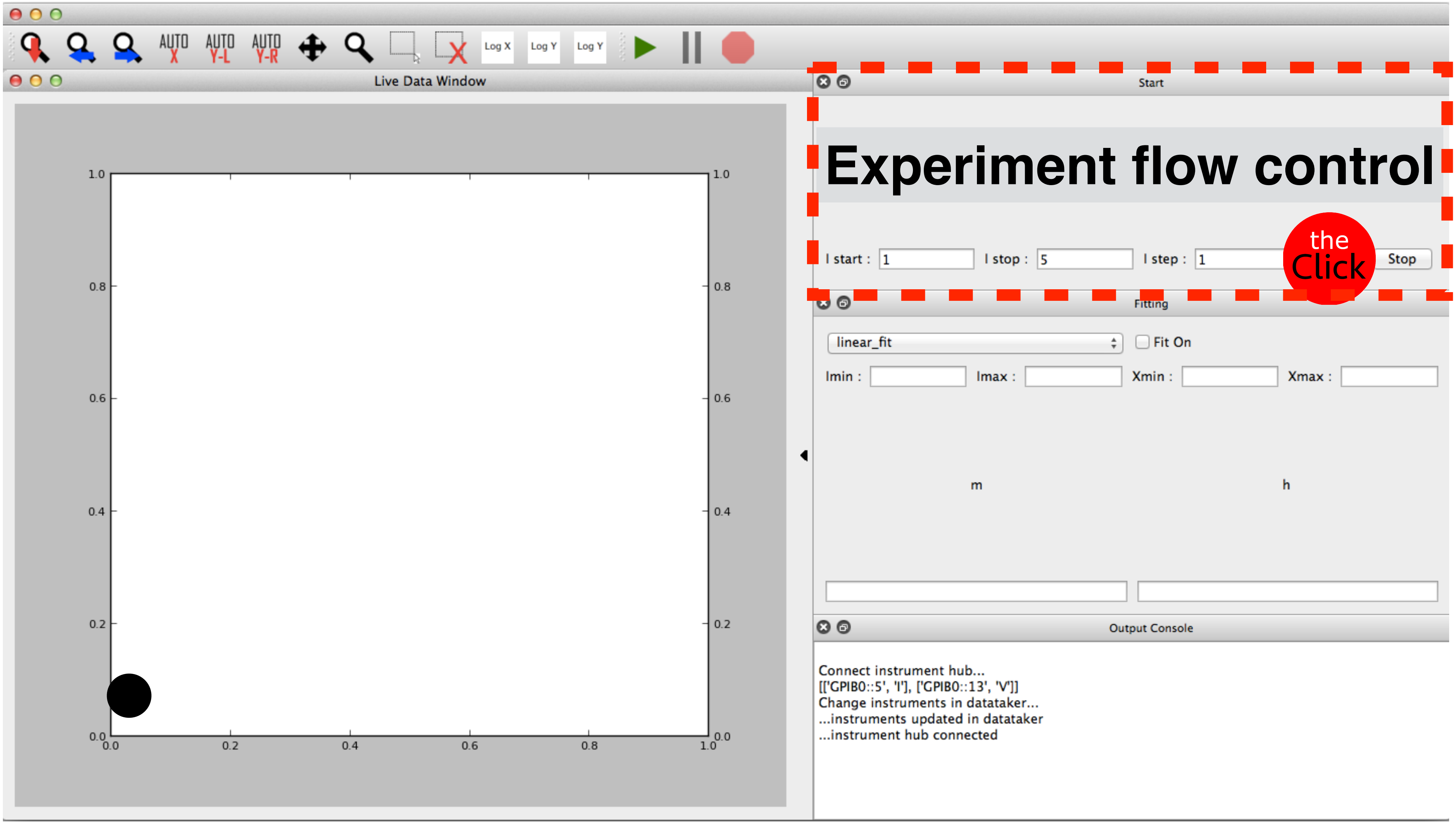


```
#empty array which will contain the measurement results
measured_voltages=[]

#create instrument objects using the communication port
current_source=CurrentSource("COM1")
voltmeter=Voltmeter("COM2")



#This is the experiment
I=1.0
current_source.set_current(I)
V=voltmeter.measure_voltage()
measured_voltages.append(V)
```

#code to perform one measurement

# Ohm’s Law example: LabGui

## Set & Measure with a Click



Experiment flow control

I start : 1 I stop : 5 I step : 1

the Click

linear_fit □ Fit On

Imin : Imax : Xmin : Xmax :

```
Connect instrument hub...
[['GPIB0::5', 'I'], ['GPIB0::13', 'V']]
Change instruments in datataker...
...instruments updated in datataker
...instrument hub connected
```

# Ohm’s Law example: LabGui

Is that hard to automate?

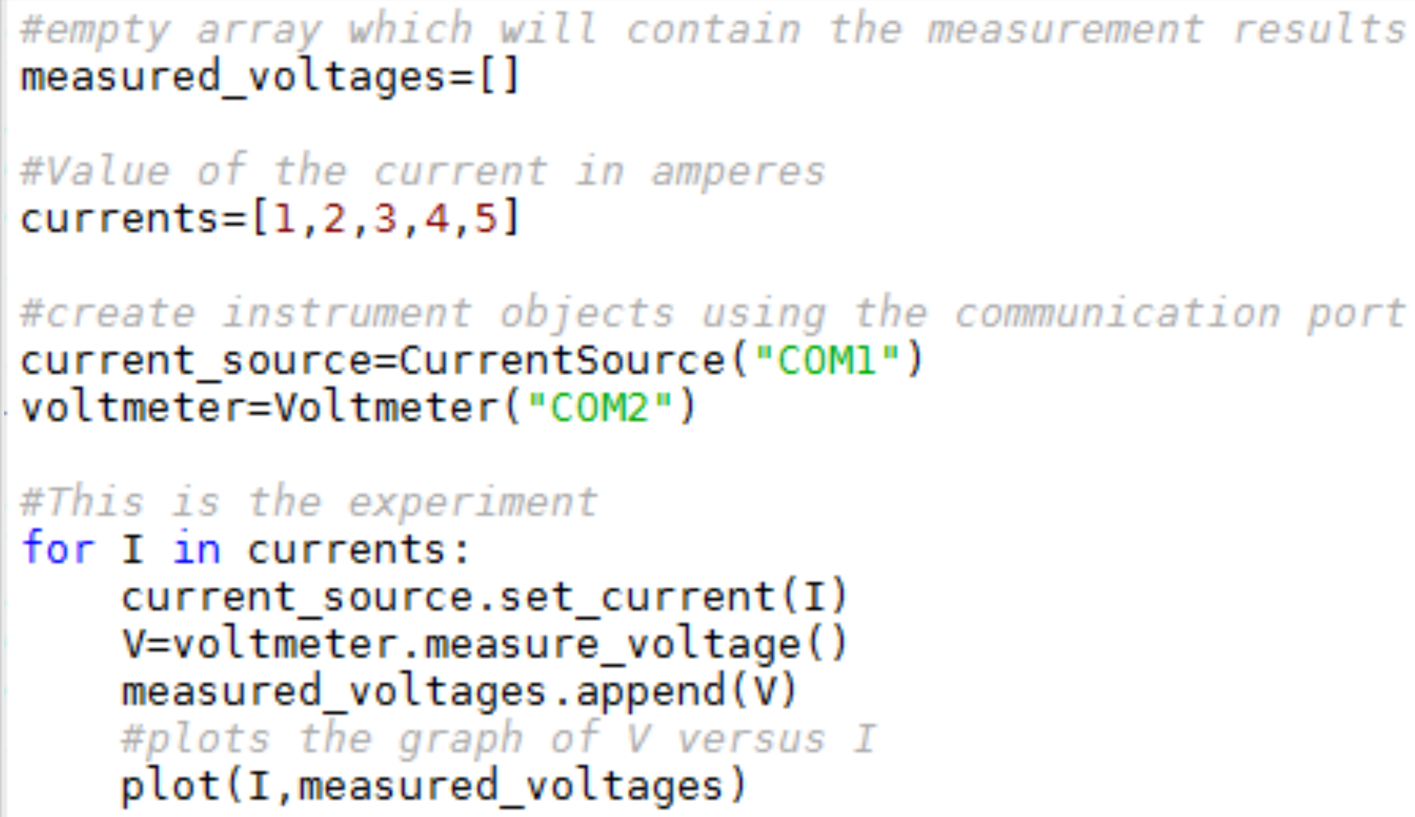
# Ohm’s Law example: LabGui

Is that hard to automate?

Nope, with **Python** it’s easy!

Just add a **FOR** loop.

# Ohm’s Law example: LabGui

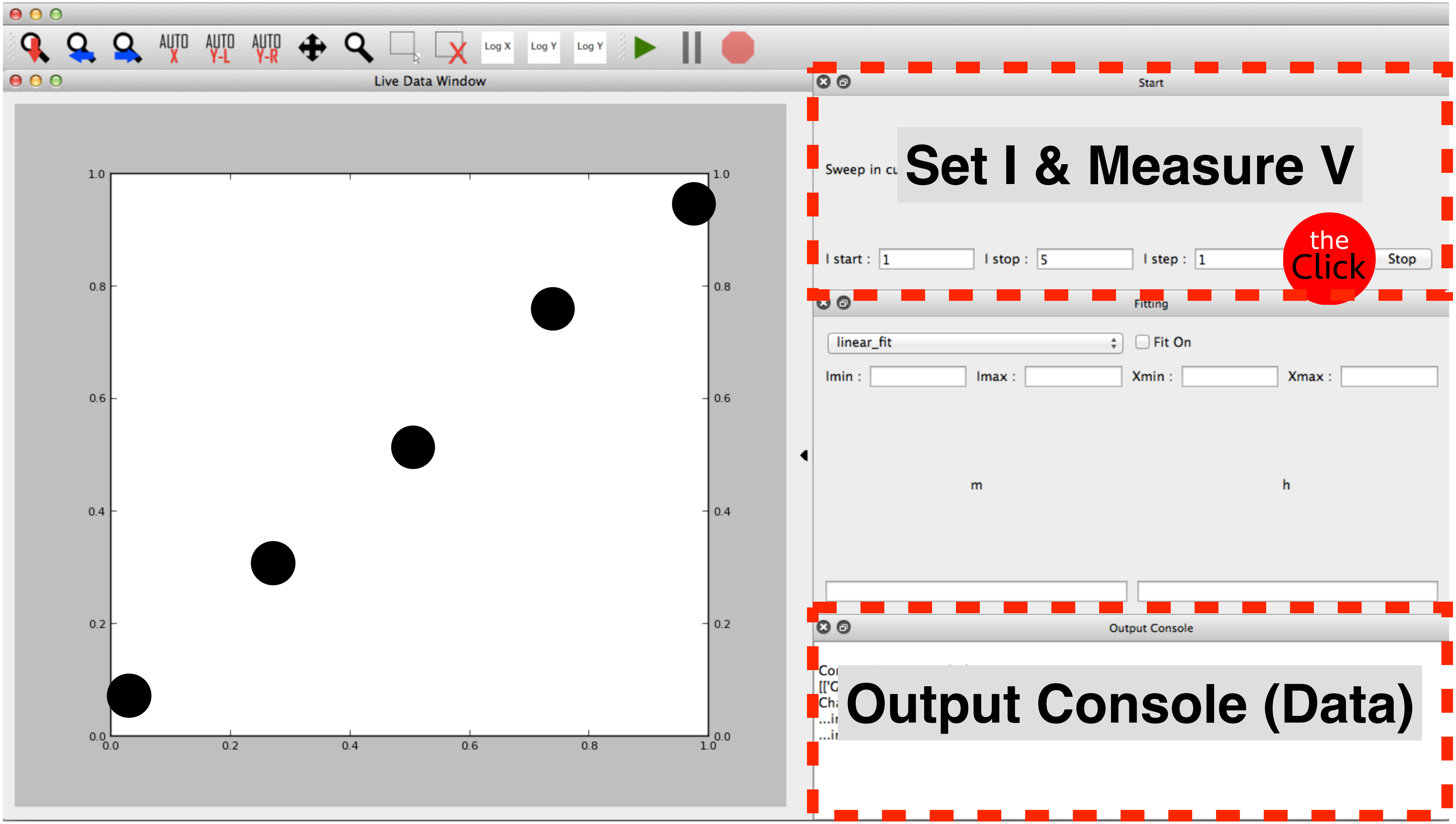


```
#empty array which will contain the measurement results
measured_voltages=[]

#Value of the current in amperes
currents=[1,2,3,4,5]

#create instrument objects using the communication port
current_source=CurrentSource("COM1")
voltmeter=Voltmeter("COM2")

#This is the experiment
for I in currents:
    current_source.set_current(I)
    V=voltmeter.measure_voltage()
    measured_voltages.append(V)
    #plots the graph of V versus I
    plot(I,measured_voltages)
```

#code to perform five measurements

# Ohm’s Law example: LabGui

5 experimental points at once



Sweep in cu

I start : 1 I stop : 5 I step : 1

linear_fit ☐ Fit On

Imin : Imax : Xmin : Xmax :

Co
[['C
Ch
...i
...i

Set I & Measure V

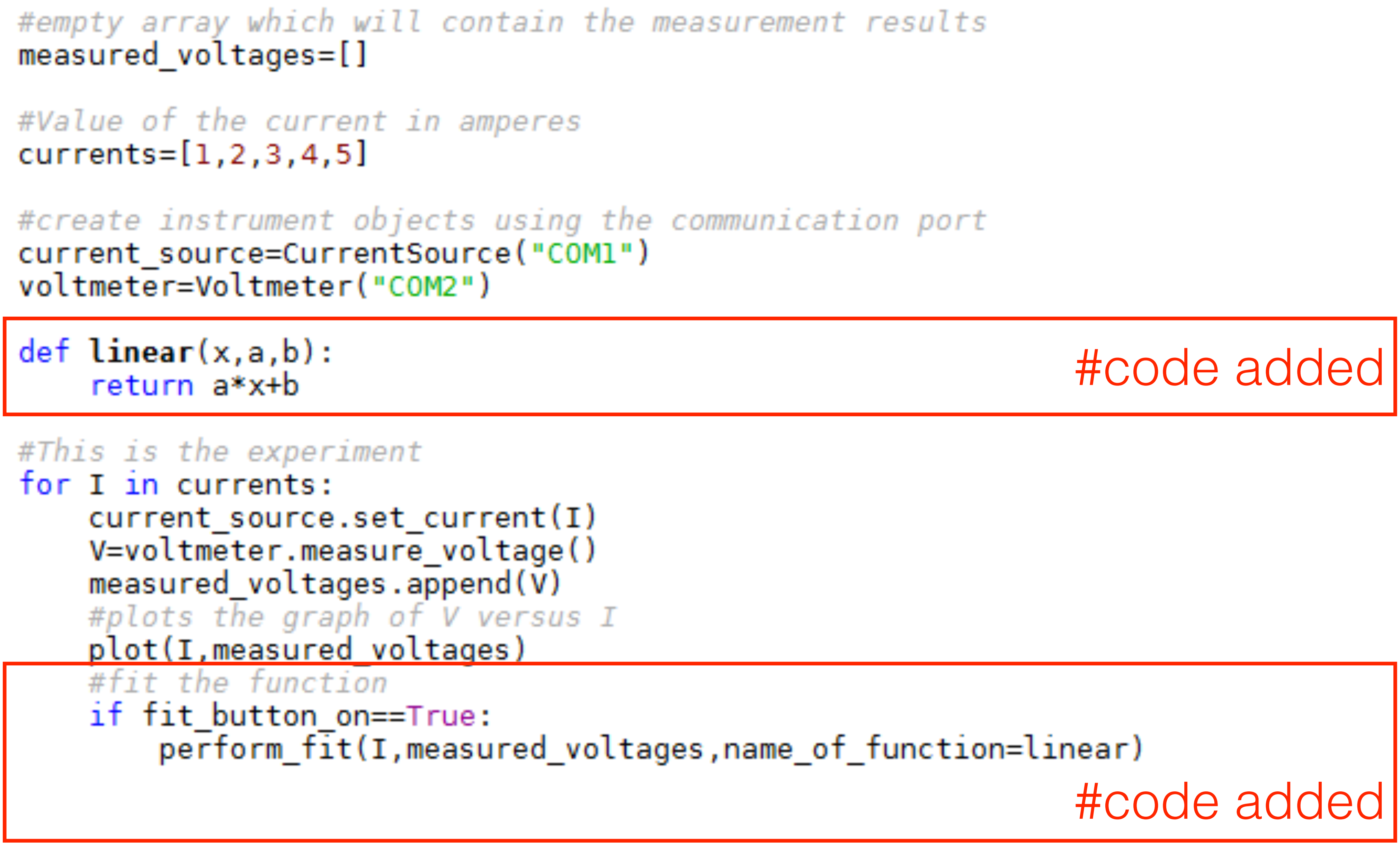
the Click

Output Console (Data)

# Ohm’s Law example: LabGui

What about fitting?

# Ohm’s Law example: LabGui



```
#empty array which will contain the measurement results
measured_voltages=[]

#Value of the current in amperes
currents=[1,2,3,4,5]

#create instrument objects using the communication port
current_source=CurrentSource("COM1")
voltmeter=Voltmeter("COM2")

def linear(x,a,b):
    return a*x+b
```

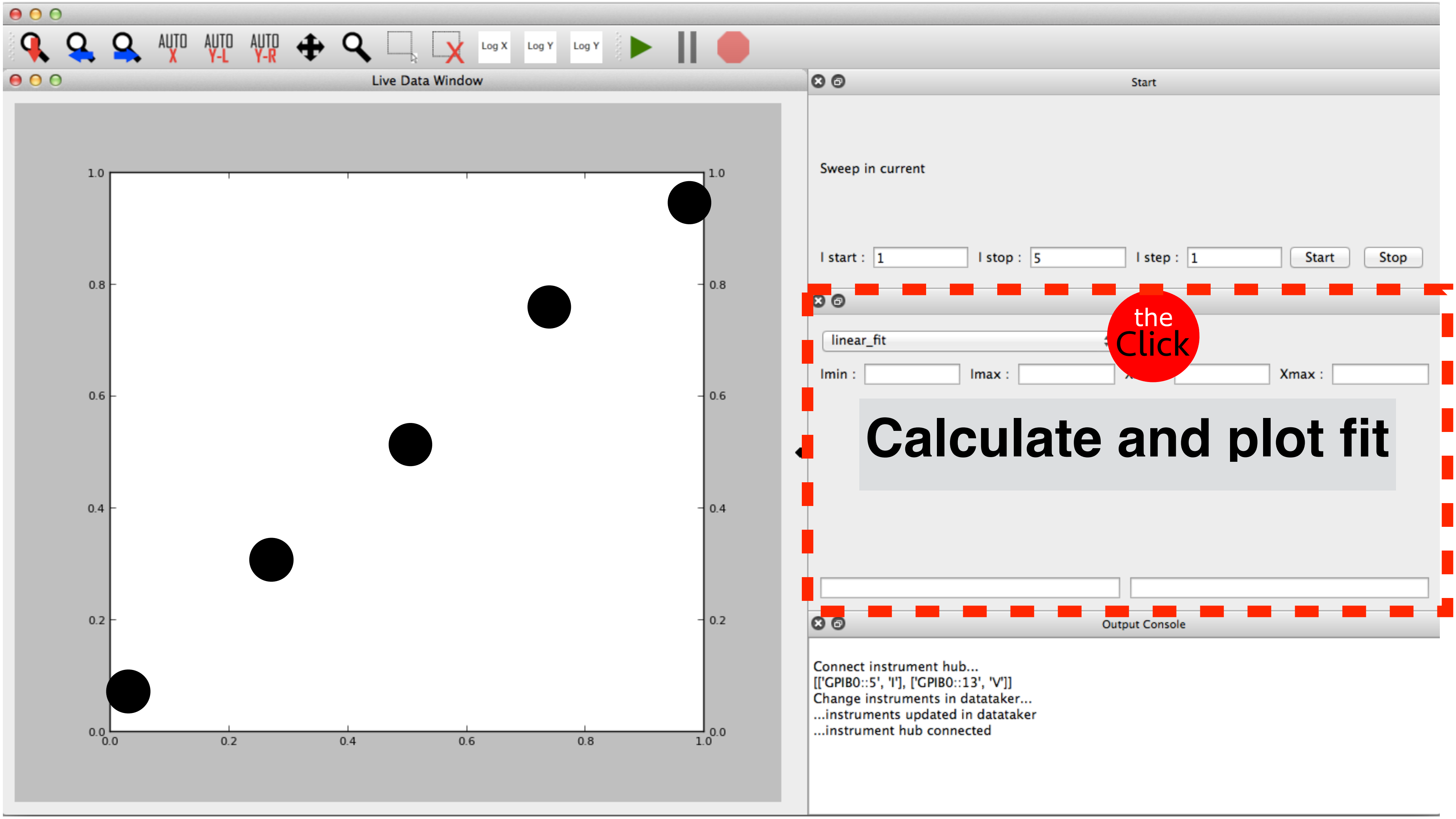
#code added

```
#This is the experiment
for I in currents:
    current_source.set_current(I)
    V=voltmeter.measure_voltage()
    measured_voltages.append(V)
    #plots the graph of V versus I
    plot(I,measured_voltages)
    #fit the function
    if fit_button_on==True:
        perform_fit(I,measured_voltages,name_of_function=linear)
```

#code added

# Ohm’s Law example: LabGui

Calculate and plot fit



Start

Sweep in current

I start : 1 I stop : 5 I step : 1

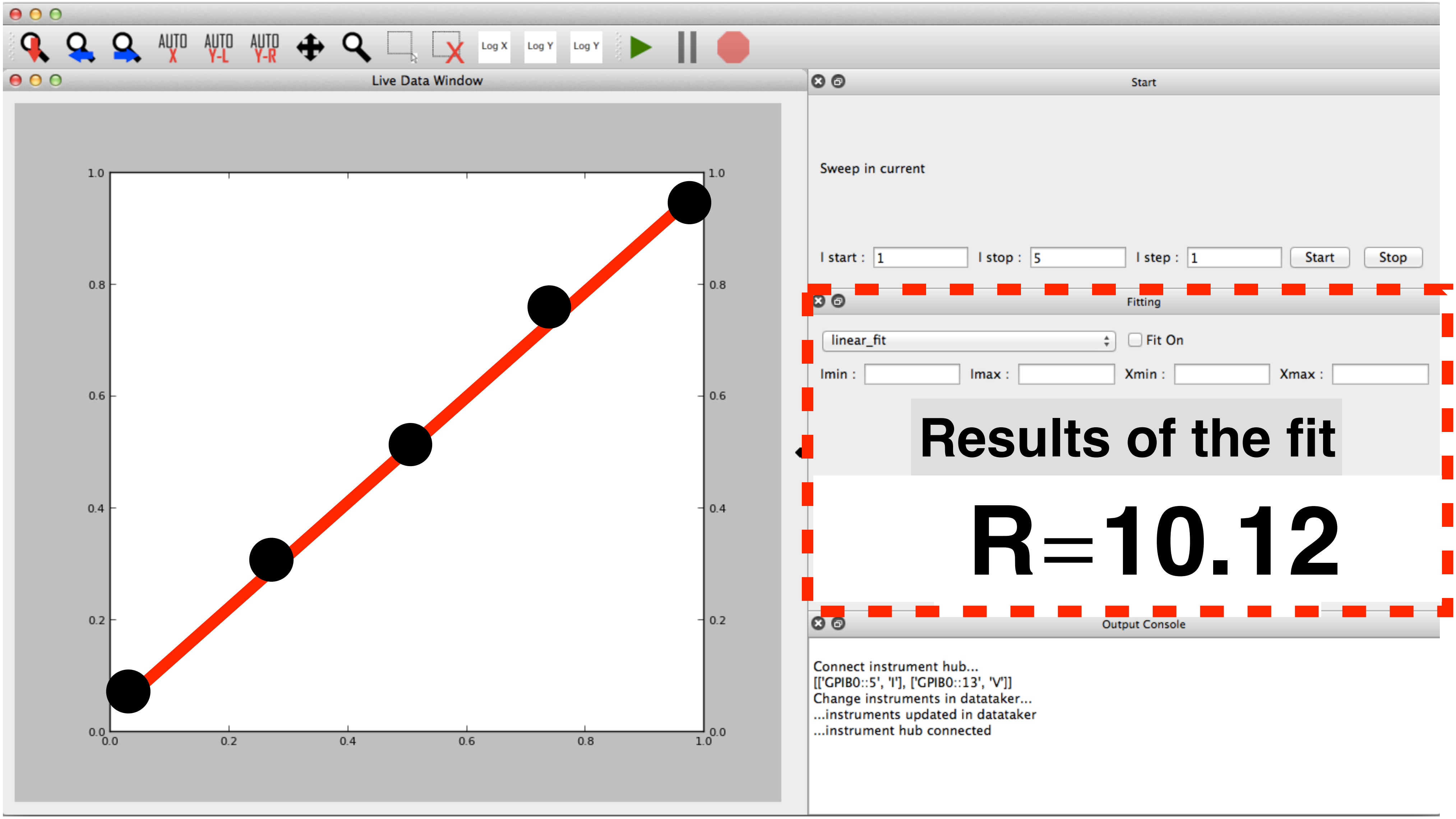
linear_fit

the Click

```
Connect instrument hub...
[['GPIB0::5', 'I'], ['GPIB0::13', 'V']]
Change instruments in datataker...
...instruments updated in datataker
...instrument hub connected
```

# Ohm’s Law example: LabGui

## Find fitting parameters



Sweep in current

I start : 1 I stop : 5 I step : 1 Start Stop

linear_fit □ Fit On

Imin : Imax : Xmin : Xmax :

Results of the fit

R=10.12

```
Connect instrument hub...
[['GPIB0::5', 'I'], ['GPIB0::13', 'V']]
Change instruments in datataker...
...instruments updated in datataker
...instrument hub connected
```

# Ohm’s Law example: LabGui

Ok, what about **non-linear** fitting?

# Ohm’s Law example: LabGui

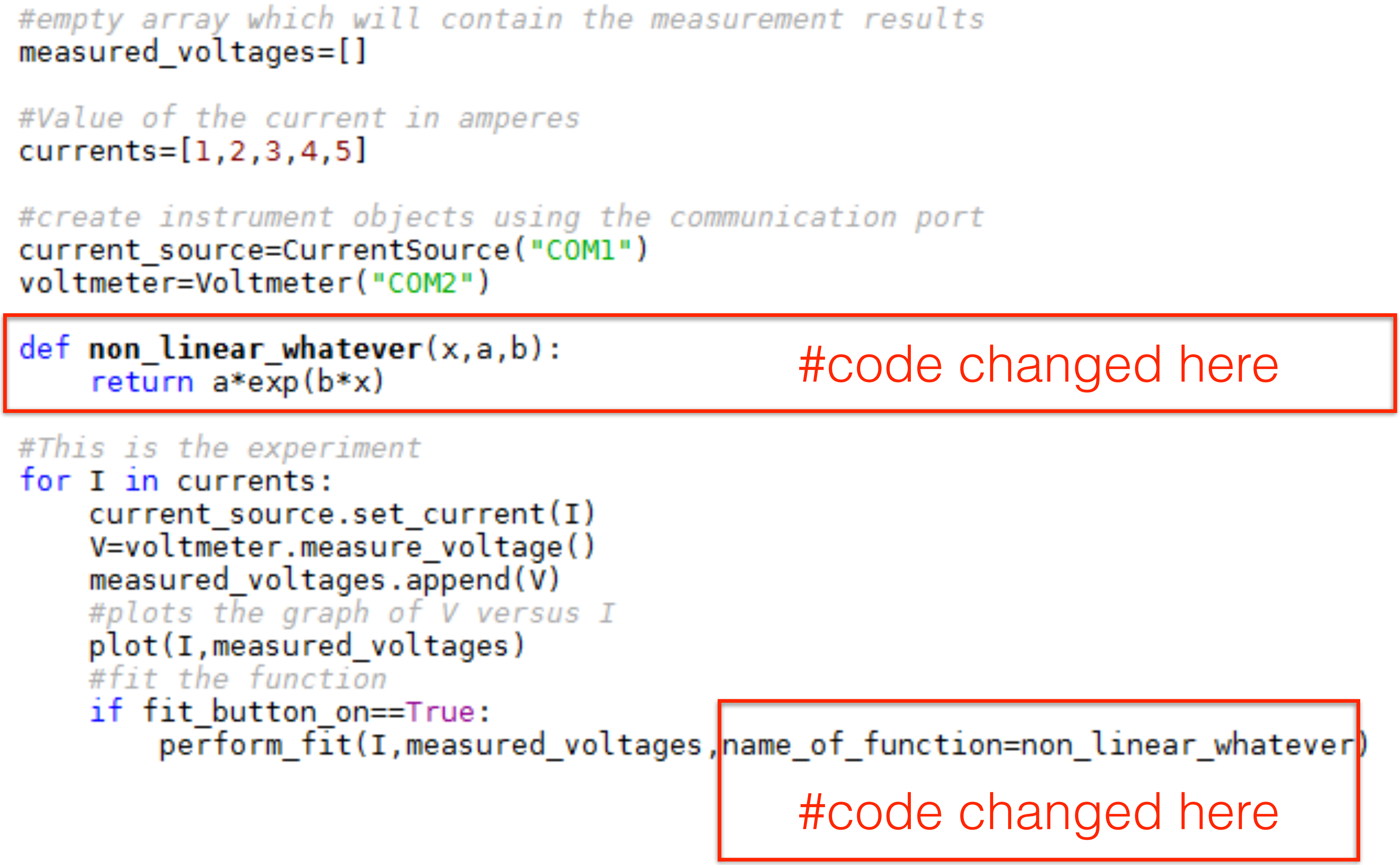
Ok, what about **non-linear** fitting?

Any function!

This is **Python**

YOU control everything

# Ohm’s Law example: LabGui



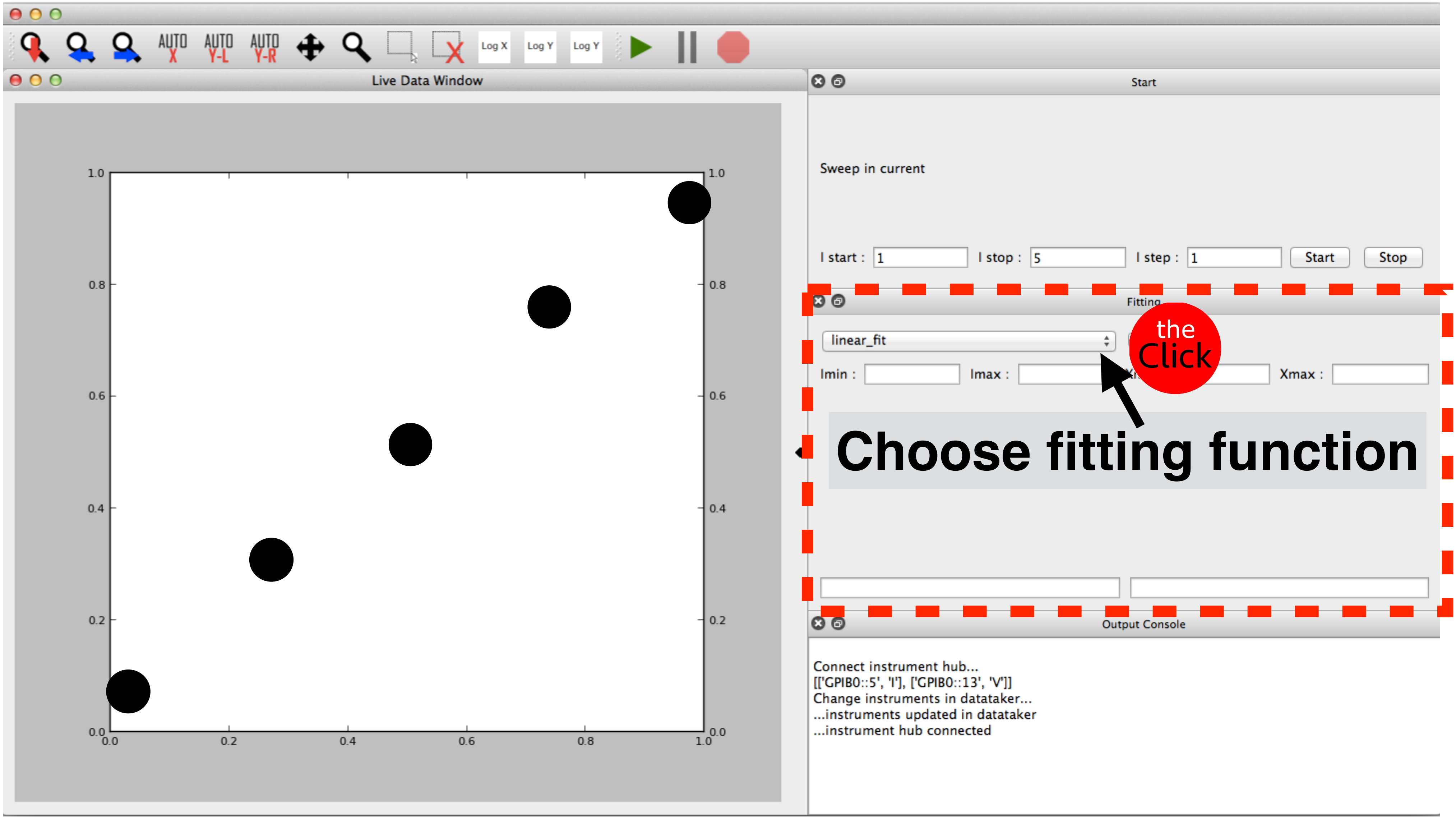
```
#empty array which will contain the measurement results
measured_voltages=[]

#Value of the current in amperes
currents=[1,2,3,4,5]

#create instrument objects using the communication port
current_source=CurrentSource("COM1")
voltmeter=Voltmeter("COM2")

def non_linear_whatever(x,a,b):
    return a*exp(b*x)

#This is the experiment
for I in currents:
    current_source.set_current(I)
    V=voltmeter.measure_voltage()
    measured_voltages.append(V)
    #plots the graph of V versus I
    plot(I,measured_voltages)
    #fit the function
    if fit_button_on==True:
        perform_fit(I,measured_voltages,name_of_function=non_linear_whatever)
```

#code changed here

#code changed here

# Ohm’s Law example: LabGui

Choose a fitting function



Sweep in current

I start : 1 I stop : 5 I step : 1 Start Stop

linear_fit

Imin : Imax : Xmax :

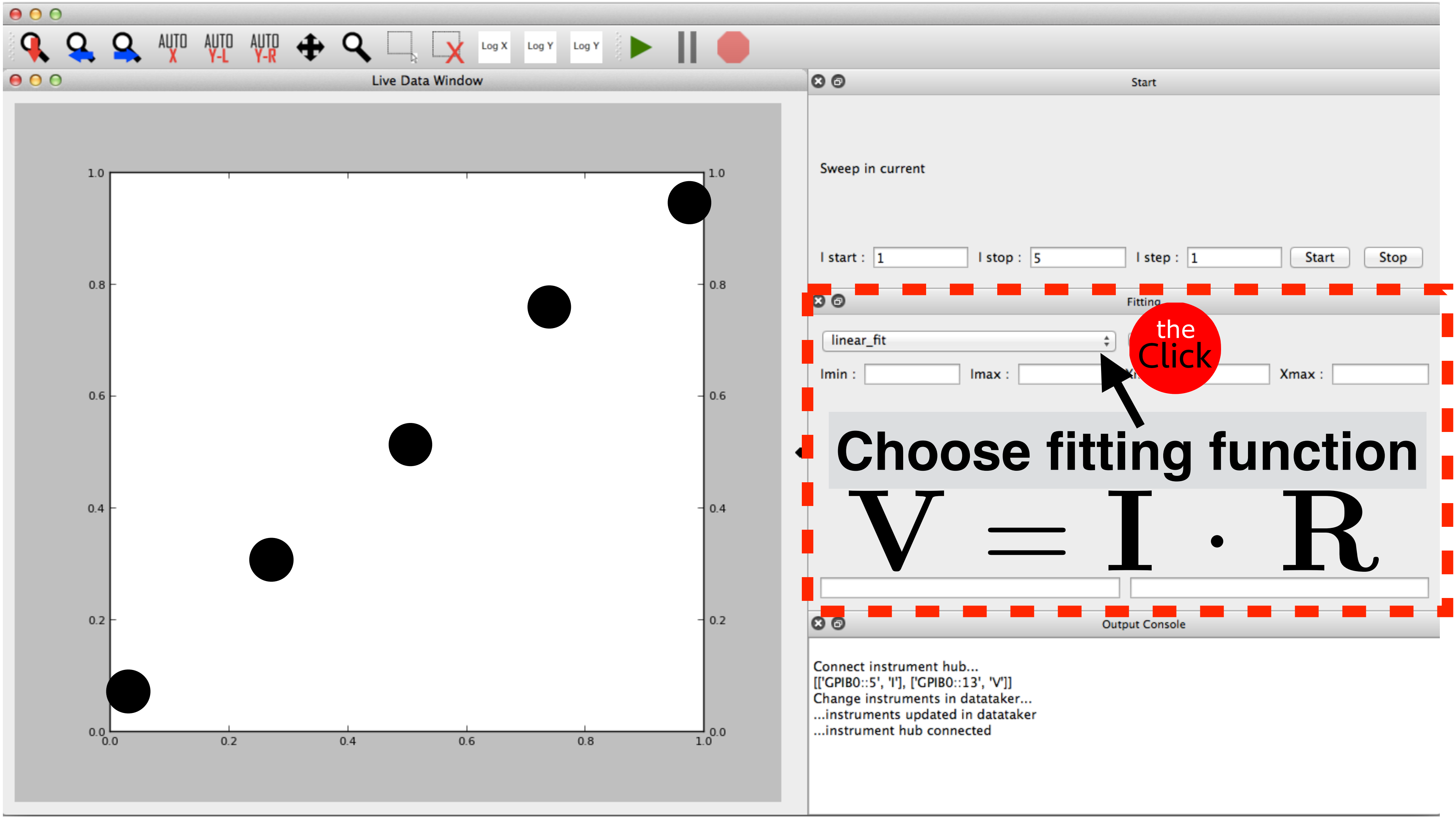
```
Connect instrument hub...
[['GPIB0::5', 'I'], ['GPIB0::13', 'V']]
Change instruments in datataker...
...instruments updated in datataker
...instrument hub connected
```

the Click

Choose fitting function

# Ohm’s Law example: LabGui

## Choose a fitting function



Sweep in current

I start : 1 I stop : 5 I step : 1 Start Stop

linear_fit

Imin : Imax : Xmax :

the Click

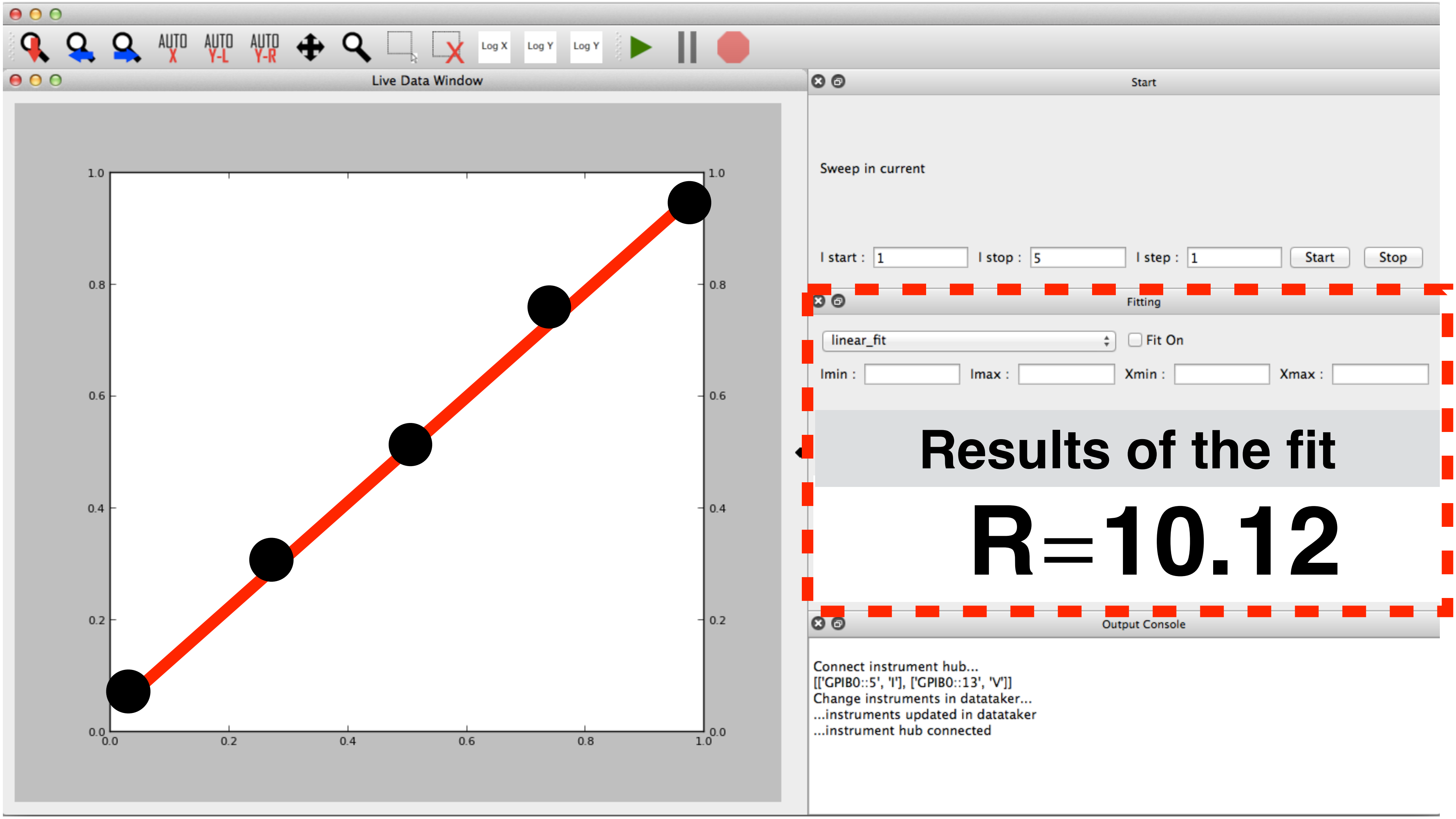
Choose fitting function

$$V = I \cdot R$$

```
Connect instrument hub...
[['GPIB0::5', 'I'], ['GPIB0::13', 'V']]
Change instruments in datataker...
...instruments updated in datataker
...instrument hub connected
```

# Ohm's Law example: LabGui

Find fitting parameters



Sweep in current

I start : 1 I stop : 5 I step : 1 Start Stop

linear_fit ☐ Fit On

Imin : Imax : Xmin : Xmax :

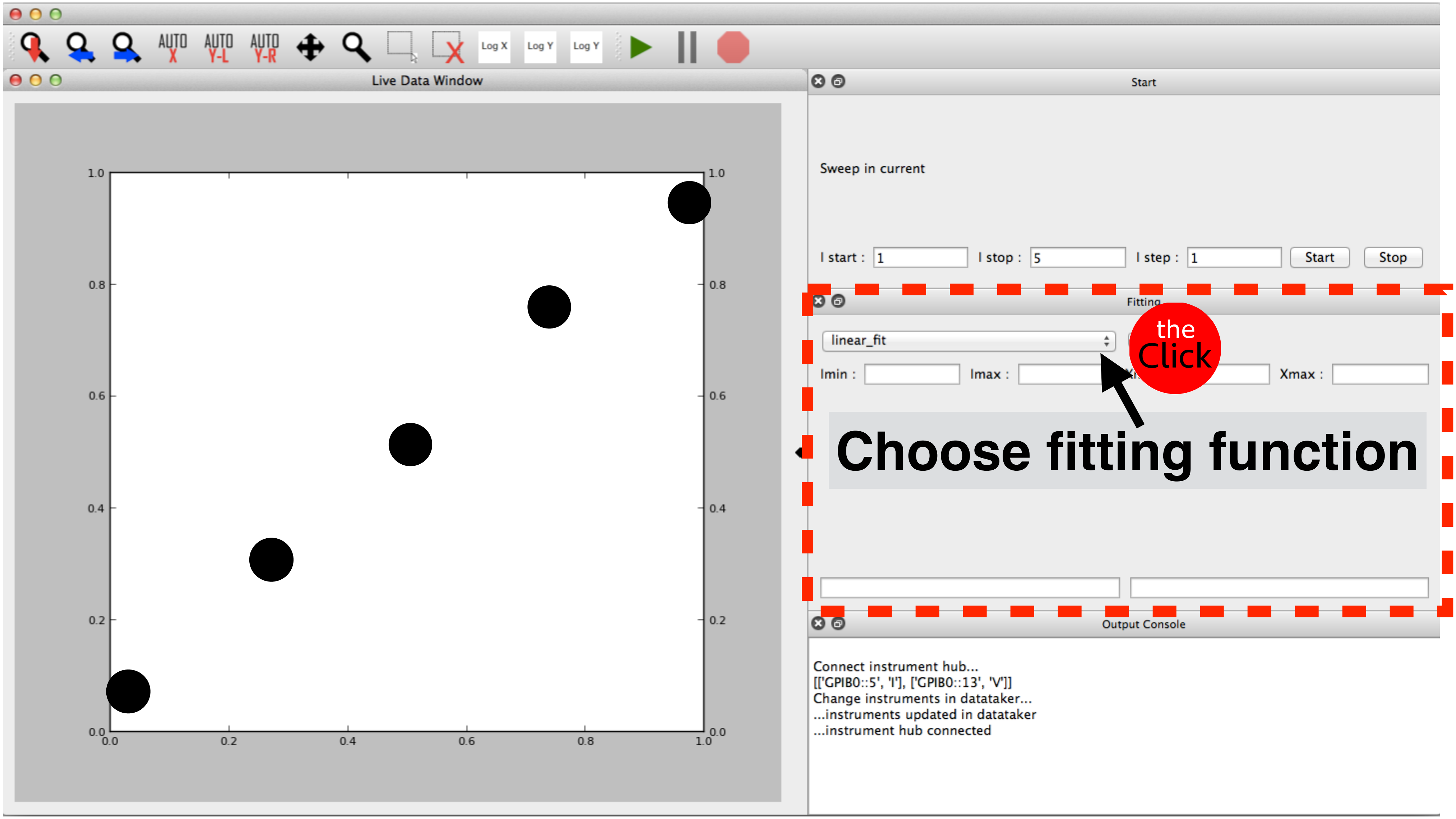
Results of the fit

R=10.12

```
Connect instrument hub...
[['GPIB0::5', 'I'], ['GPIB0::13', 'V']]
Change instruments in datataker...
...instruments updated in datataker
...instrument hub connected
```

# Ohm’s Law example: LabGui

## Choose another fitting function



Sweep in current

I start : 1 I stop : 5 I step : 1

linear_fit

Imin : Imax : Xmax :

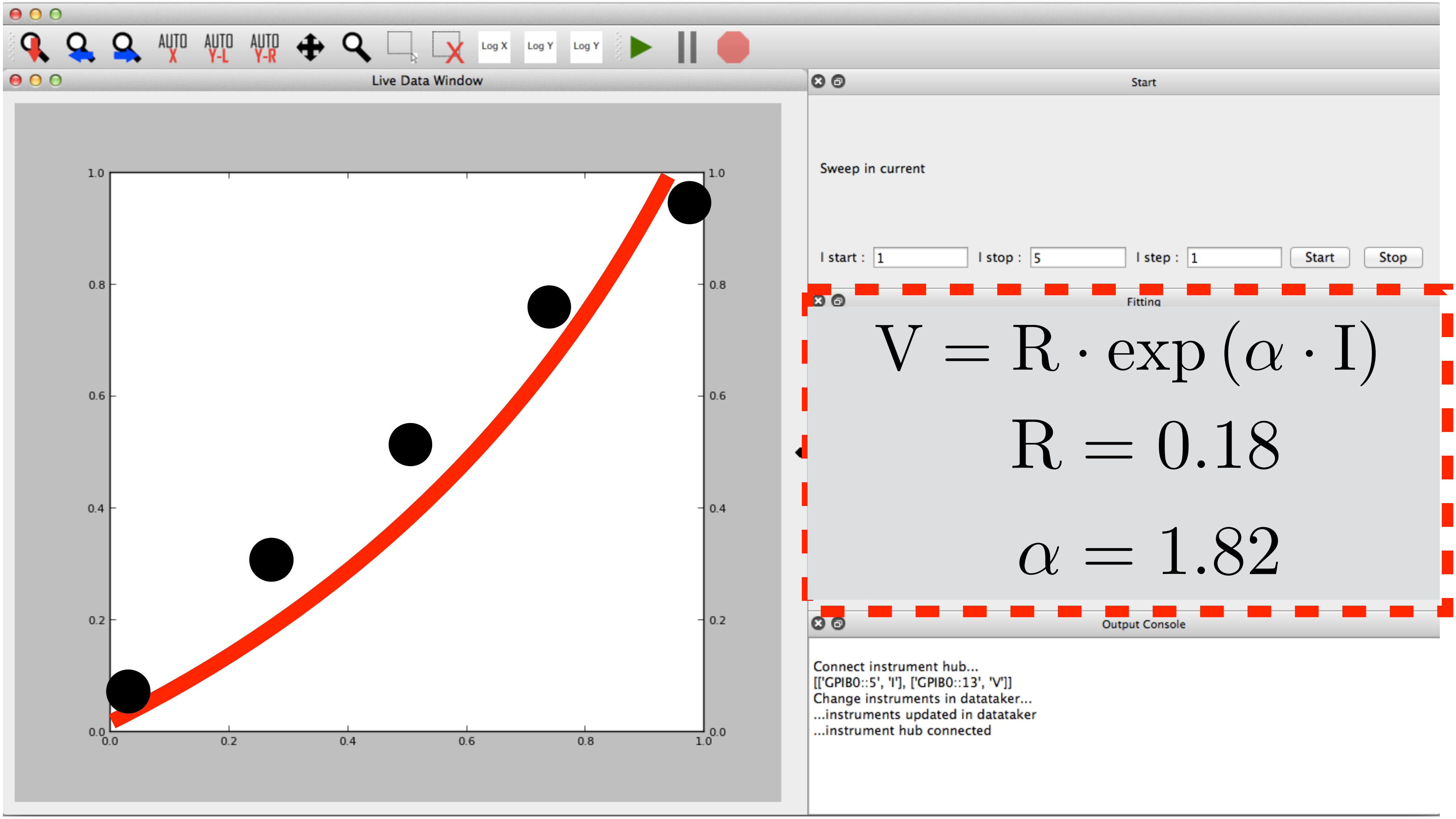
the Click

Choose fitting function

```
Connect instrument hub...
[['GPIB0::5', 'I'], ['GPIB0::13', 'V']]
Change instruments in datataker...
...instruments updated in datataker
...instrument hub connected
```

# Ohm’s Law example: LabGui

Calculate and plot new fit



Sweep in current

I start : 1    I stop : 5    I step : 1

$$V = R \cdot \exp\left(\alpha \cdot I\right)$$

$$R = 0.18$$

$$\alpha = 1.82$$

```
Connect instrument hub...
[['GPIB0::5', 'I'], ['GPIB0::13', 'V']]
Change instruments in datataker...
...instruments updated in datataker
...instrument hub connected
```

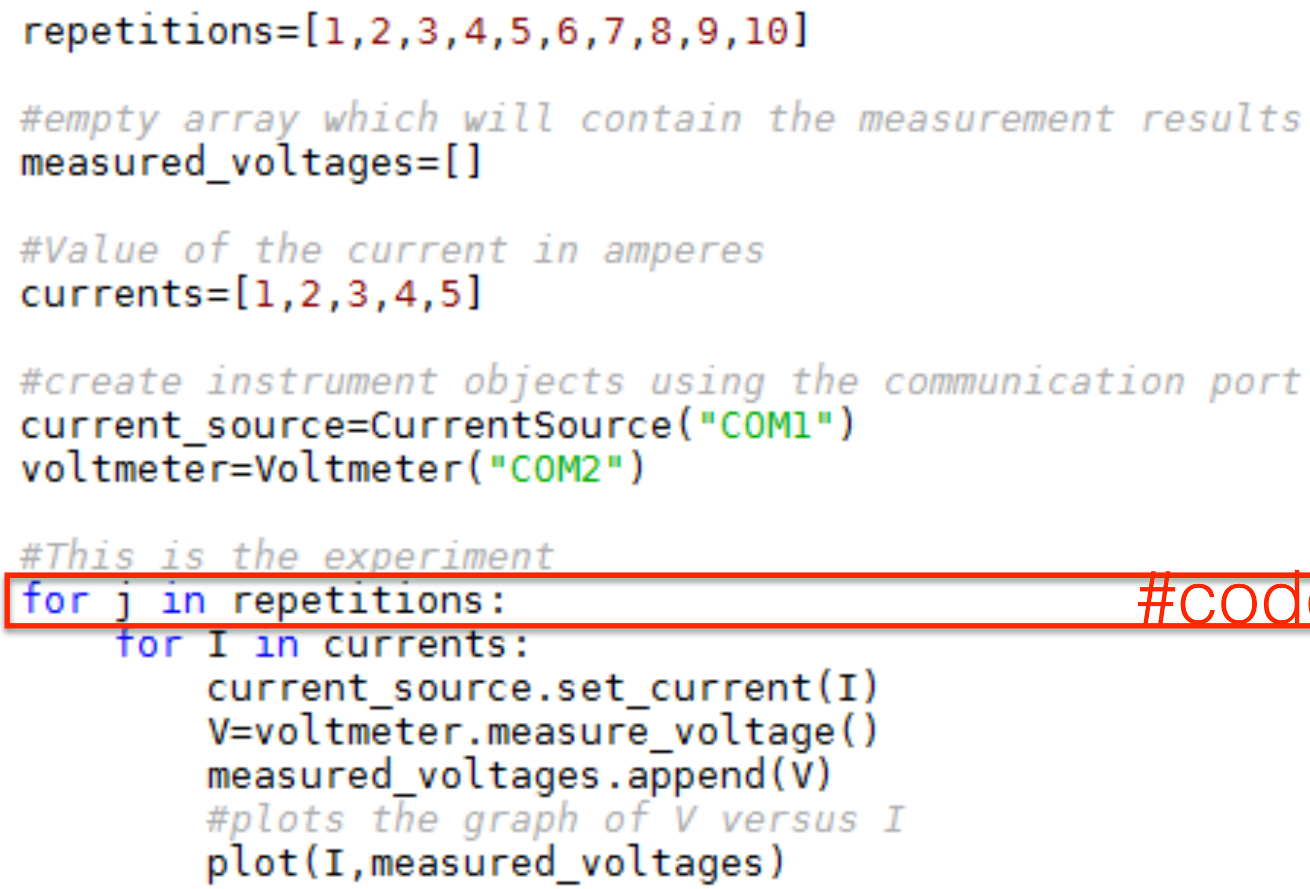
# Ohm’s Law example: LabGui

Want more statistics?

# Ohm’s Law example: LabGui

Want more statistics?
Just add another **FOR** loop

# Ohm’s Law example: LabGui



```
repetitions=[1,2,3,4,5,6,7,8,9,10]

#empty array which will contain the measurement results
measured_voltages=[]

#Value of the current in amperes
currents=[1,2,3,4,5]

#create instrument objects using the communication port
current_source=CurrentSource("COM1")
voltmeter=Voltmeter("COM2")

#This is the experiment
for j in repetitions:
    for I in currents:
        current_source.set_current(I)
        V=voltmeter.measure_voltage()
        measured_voltages.append(V)
        #plots the graph of V versus I
        plot(I,measured_voltages)
```

#code changed **only here**

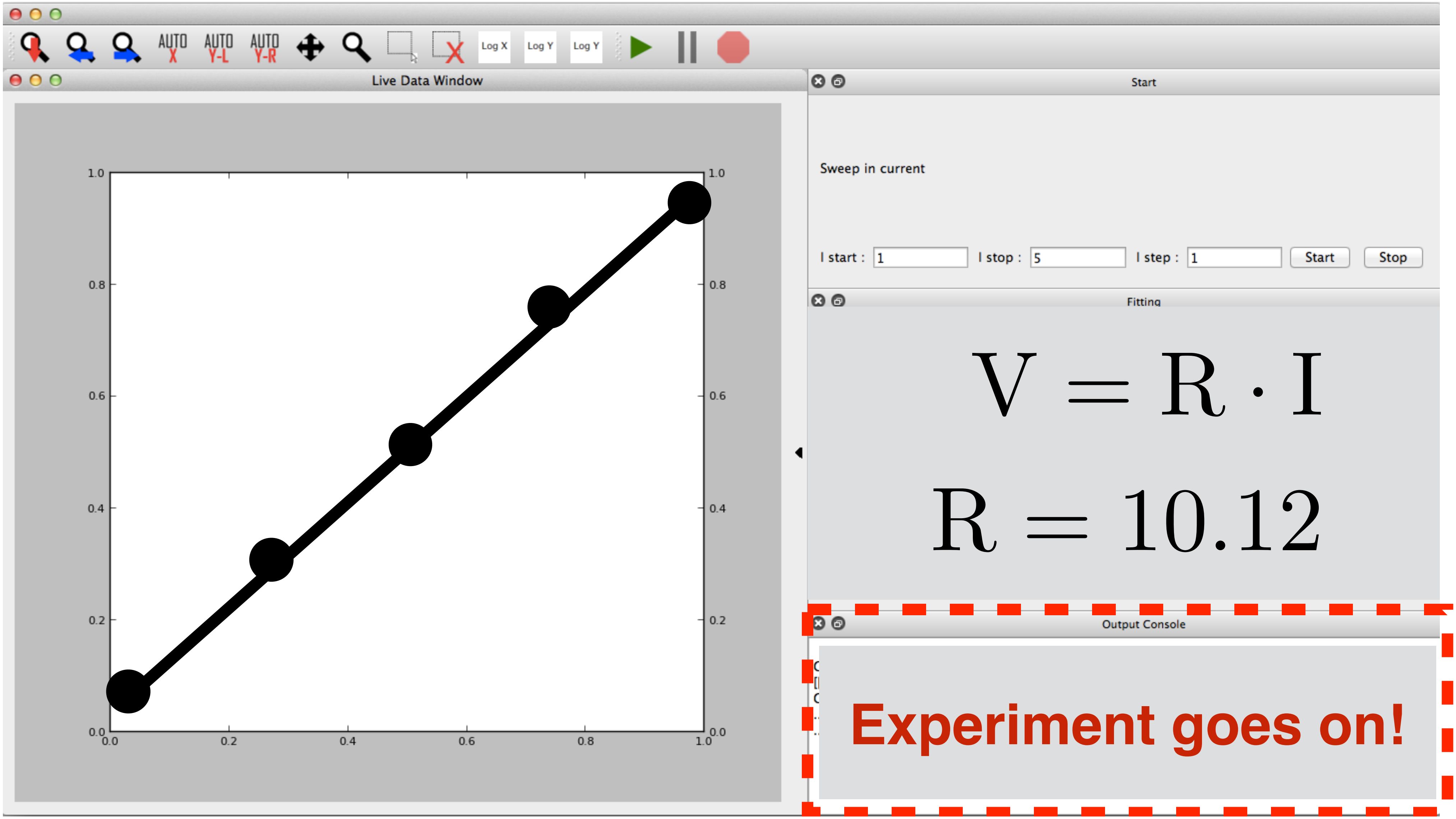
# Ohm’s Law example: LabGui

Want more statistics?
Just add another **FOR** loop

**Bonus:**
Play with fits of already taken data
while
experiment goes on in the bckg

# Ohm’s Law example: LabGui

Meanwhile in the background



Sweep in current

I start : 1 I stop : 5 I step : 1

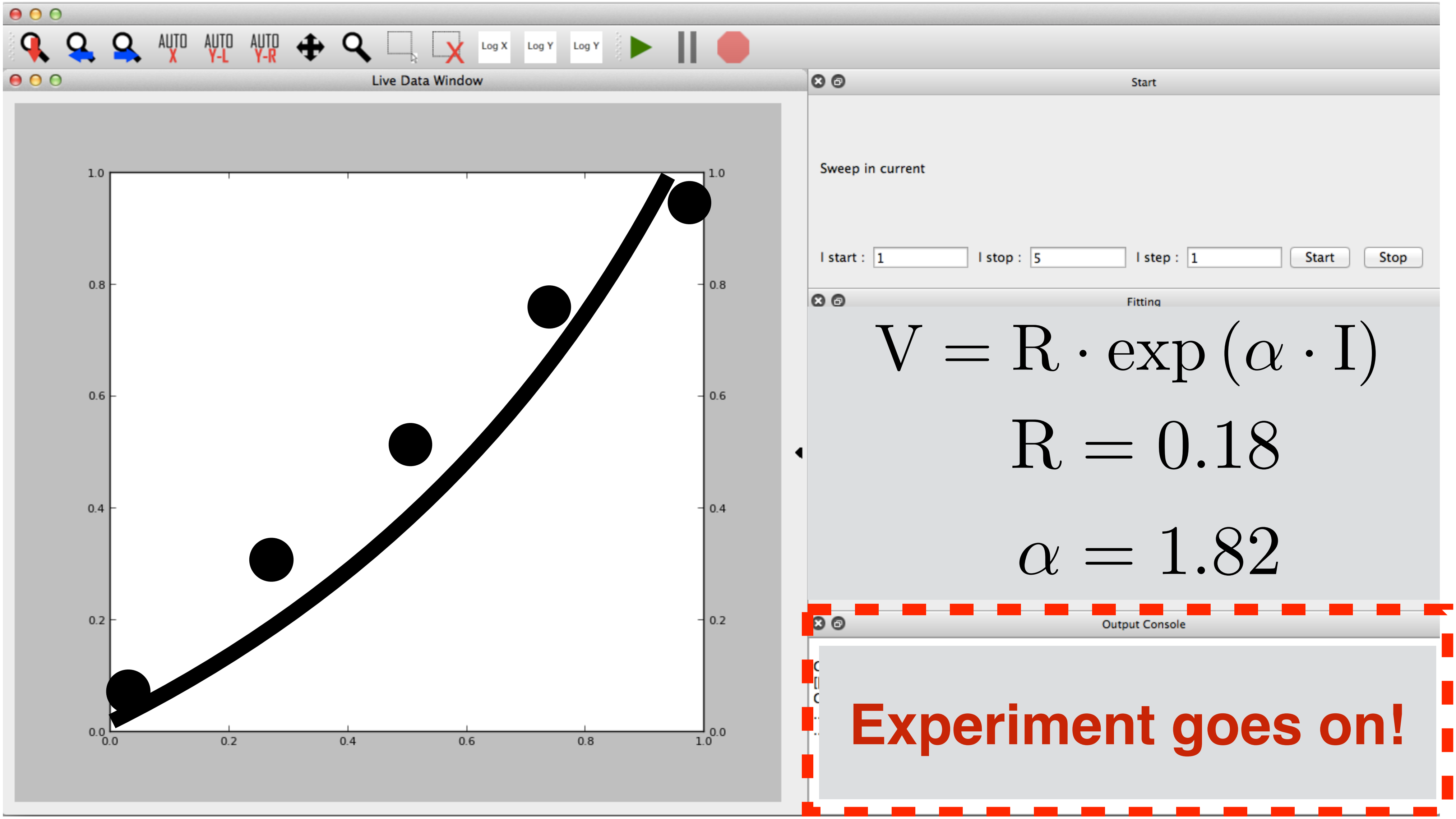
$$V = R \cdot I$$

$$R = 10.12$$

Experiment goes on!

# Ohm's Law example: LabGui

Meanwhile in the background



Sweep in current

I start : 1 I stop : 5 I step : 1

$$V = R \cdot \exp\left(\alpha \cdot I\right)$$

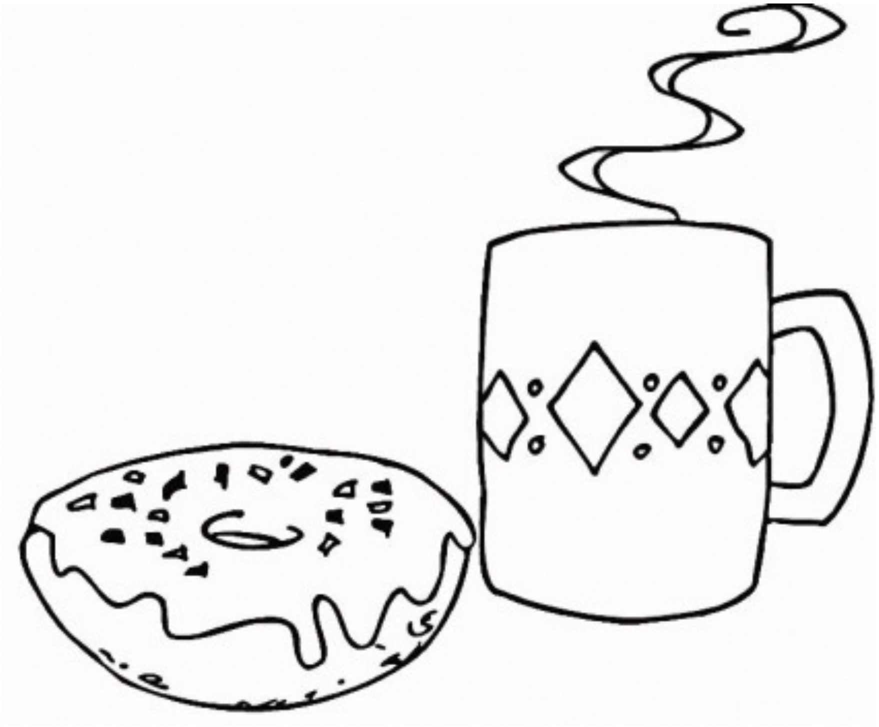
$$R = 0.18$$

$$\alpha = 1.82$$

Experiment goes on!

# Ohm’s Law example: LabGui

Go grab a coffee
while your experiment is going on!

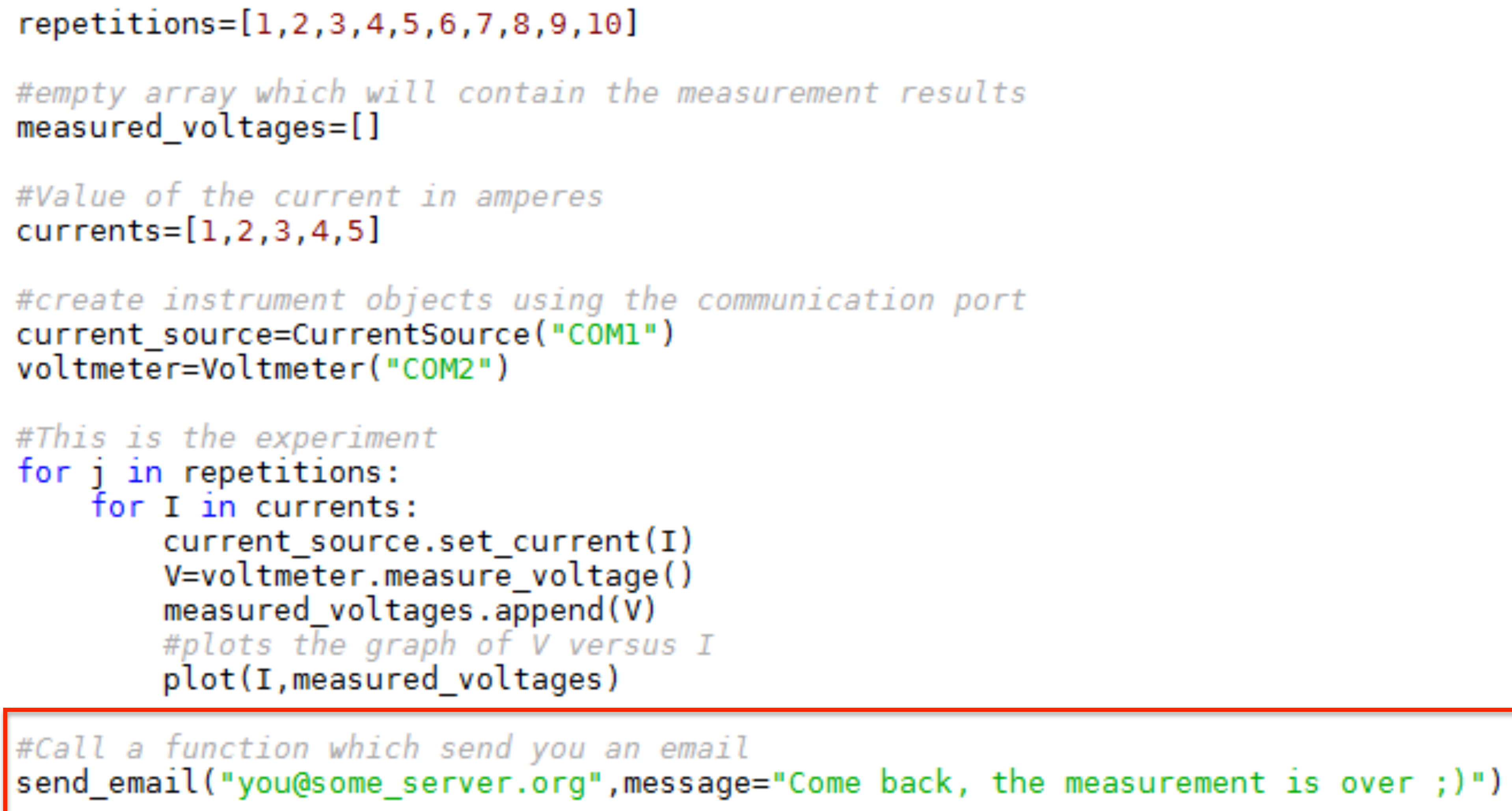


# Ohm’s Law example: LabGui

Go grab a coffee
while your experiment is going on!

**Another bonus:**
**LabGui** will send you an email, if something happens!

# Ohm’s Law example: LabGui



```
repetitions=[1,2,3,4,5,6,7,8,9,10]

#empty array which will contain the measurement results
measured_voltages=[]

#Value of the current in amperes
currents=[1,2,3,4,5]

#create instrument objects using the communication port
current_source=CurrentSource("COM1")
voltmeter=Voltmeter("COM2")

#This is the experiment
for j in repetitions:
    for I in currents:
        current_source.set_current(I)
        V=voltmeter.measure_voltage()
        measured_voltages.append(V)
        #plots the graph of V versus I
        plot(I,measured_voltages)

#Call a function which send you an email
send_email("you@some_server.org",message="Come back, the measurement is over ;)")
```

#all the code you need to send an email

# Ohm’s Law example

Why should I use **LabGui**?

Let’s compare the key features

# Compare LabView & LabGui

VS

**LabView**

**Python / LabGui**

#1

**ALL** devices speak

**ONE** language

# Compare LabView & LabGui

| | | |
|---|---|---|
| ALL devices speak ONE language | ✓ | ✓ |

# #2
# Project’s complexity

# #2
# Project’s complexity

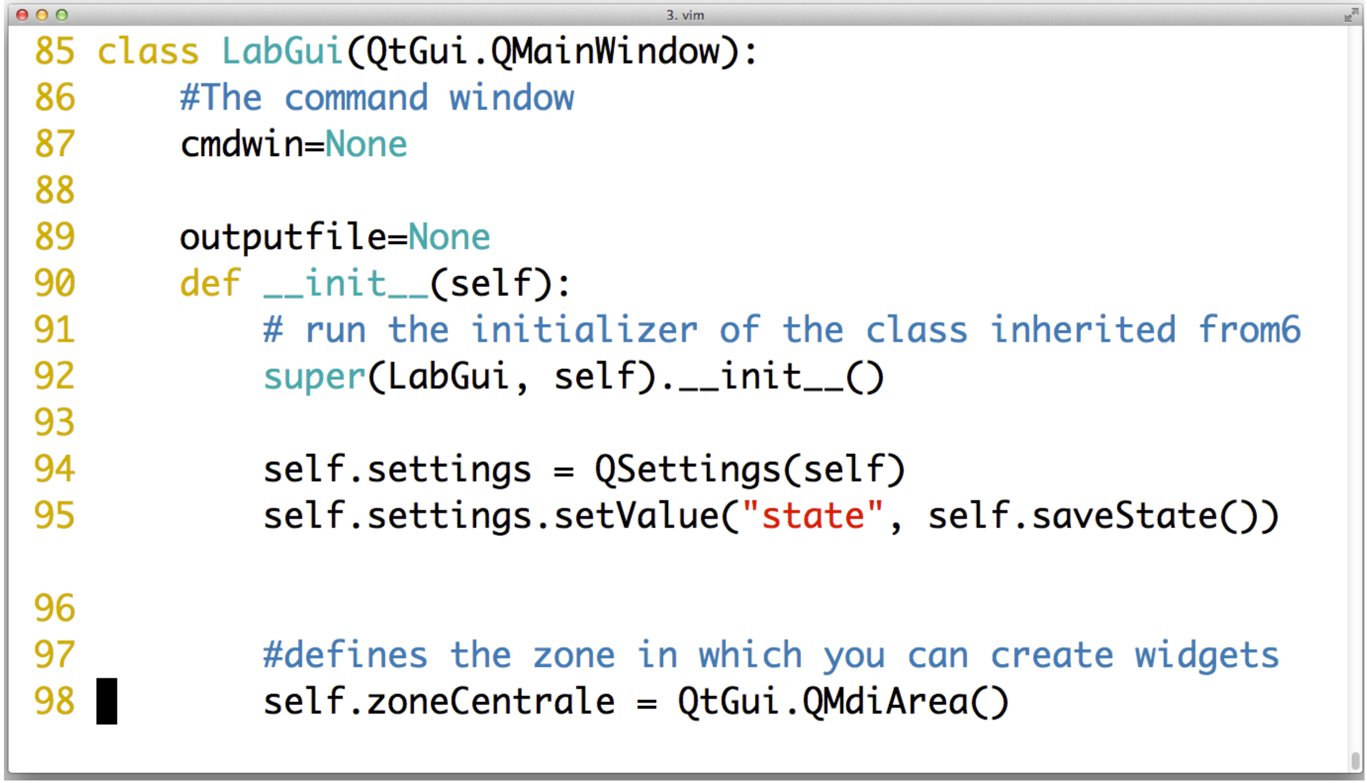
Source of complications: **PL** vs **GPL**

**LabGui** = **P**rogramming **L**anguage

**LabView** = **G**raphical **P**rogramming **L**anguage

# LabGui

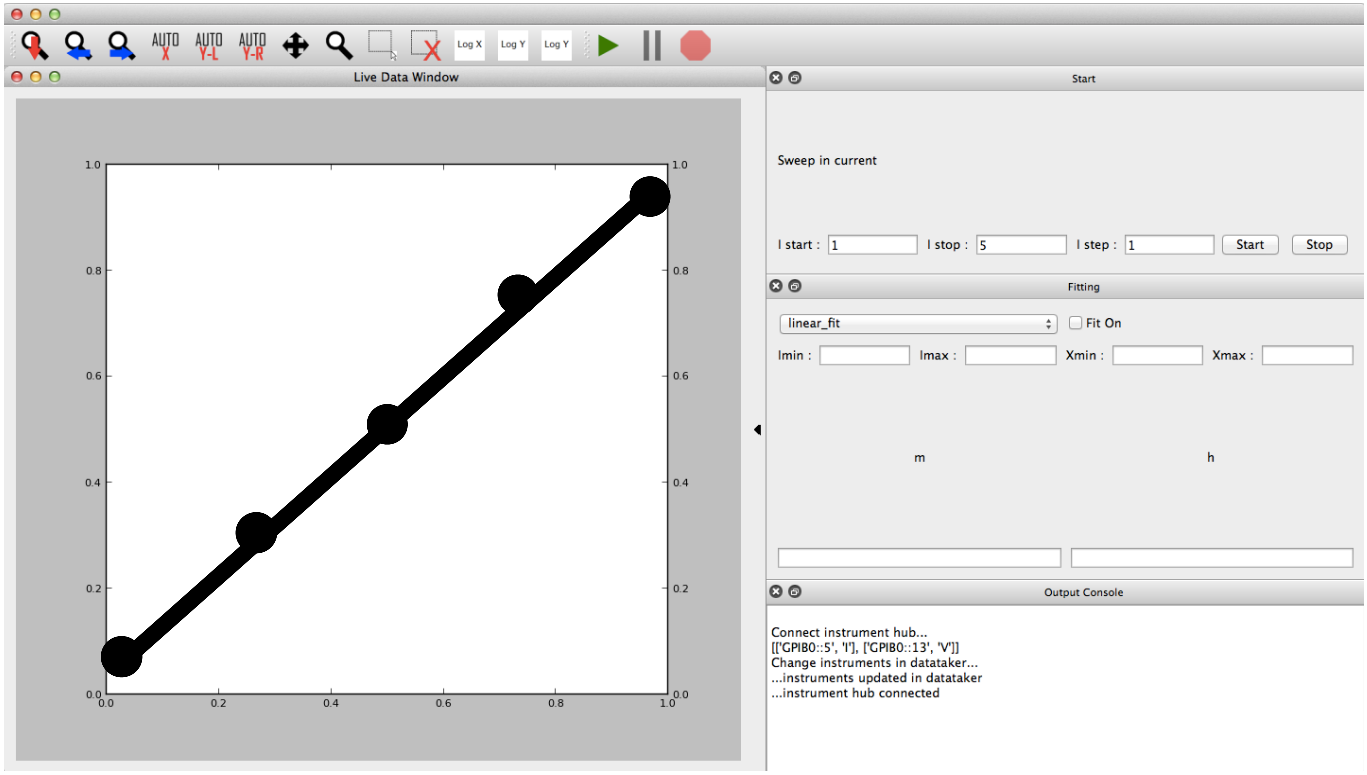
Code →



3. vim

```python
class LabGui(QtGui.QMainWindow):
    #The command window
    cmdwin=None

    outputfile=None
    def __init__(self):
        # run the initializer of the class inherited from6
        super(LabGui, self).__init__()

        self.settings = QSettings(self)
        self.settings.setValue("state", self.saveState())


        #defines the zone in which you can create widgets
        self.zoneCentrale = QtGui.QMdiArea()
```

Interface→



Sweep in current

I start : 1 I stop : 5 I step : 1

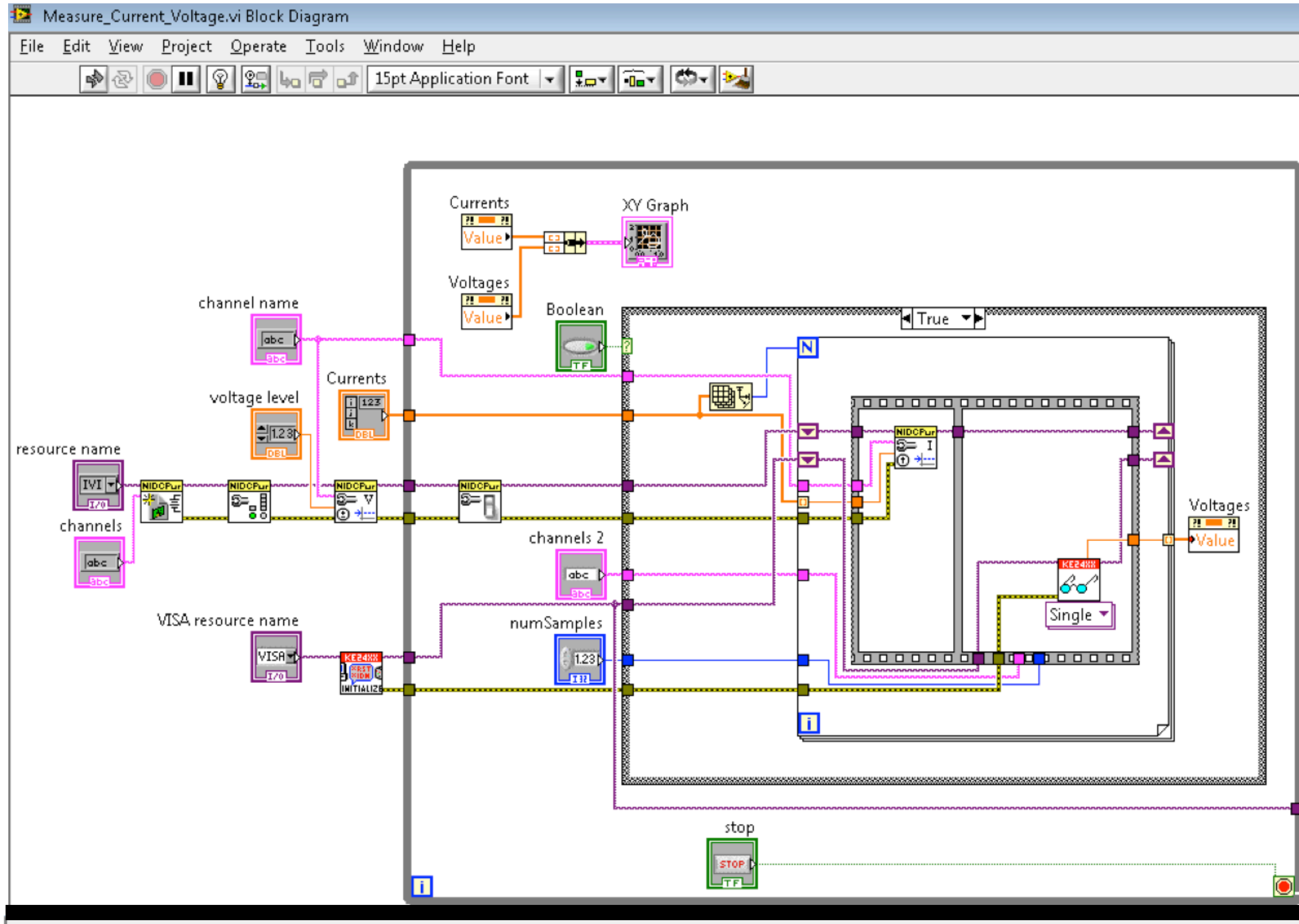
linear_fit ☐ Fit On

Imin : Imax : Xmin : Xmax :

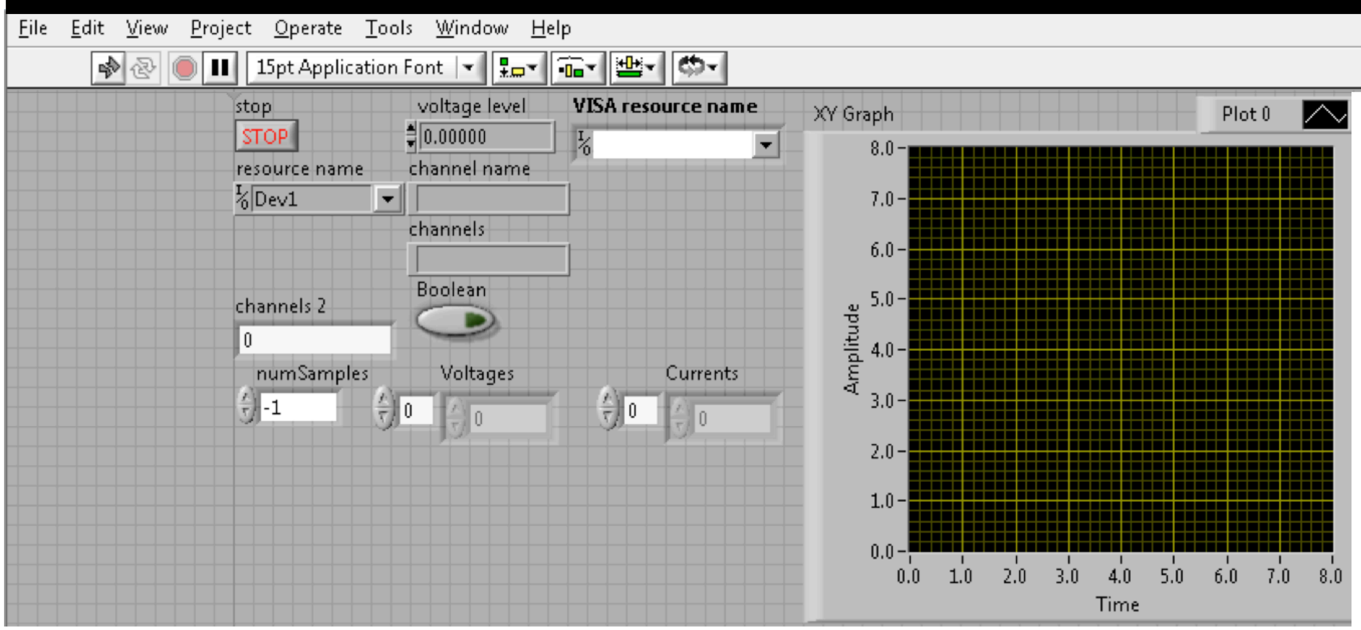
```
Connect instrument hub...
[['GPIB0::5', 'I'], ['GPIB0::13', 'V']]
Change instruments in datataker...
...instruments updated in datataker
...instrument hub connected
```

# LabView

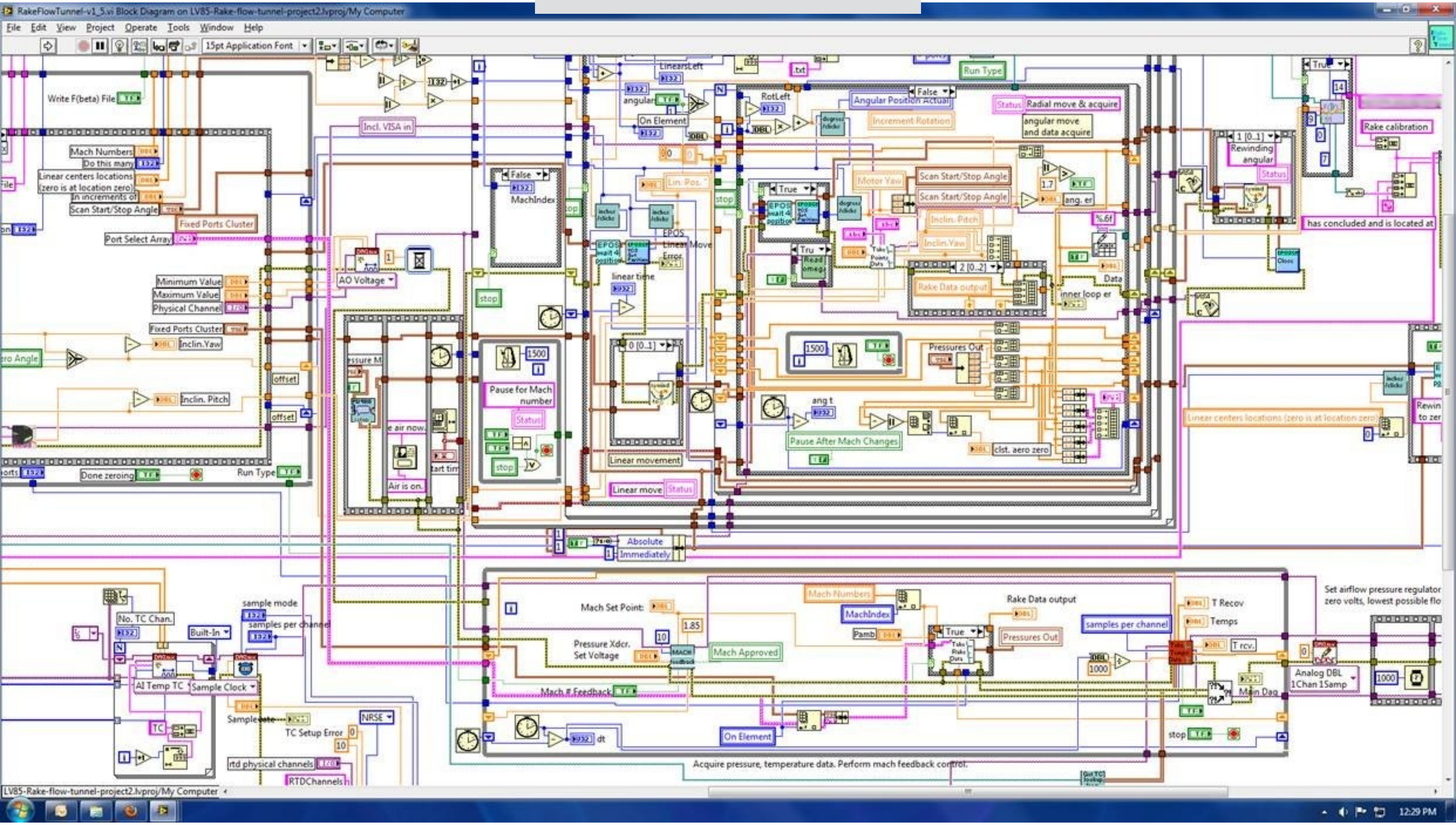
Code →



Interface ▸



# LabView



Actually that diagram looks benign compared to some of the stuff that we use in our lab.

# PL vs GPL

“LabView makes:

- the easy things easier and
- the hard things harder.”

# PL vs GPL

“LabView makes:

- the easy things easier and
- the hard things harder.”

- The learning curve of **GPL** is lower, but so is the productivity

# PL vs GPL

“LabView makes:

- the easy things easier and
- the hard things harder.”

- The learning curve of **GPL** is lower, but so is the productivity

- **GPL** has less convenient and standard VERSION CONTROL

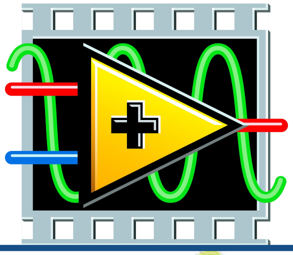
# PL vs GPL

"LabView makes:

- the easy things easier and
- the hard things harder."

- The learning curve of **GPL** is lower, but so is the productivity
- **GPL** has less convenient and standard VERSION CONTROL
- It's cumbersome and slow to use **GPL** over SSH and/or on old computers

# Compare LabView & LabGui

| |  | |
|---|---|---|
| ALL devices speak ONE language | ✓ | ✓ |
| Language | GPL | PL |
| Complex projects easy | ✗ | ✓ |
| Version control | ✗ | ✓ |

#3

# Proprietary vs Open Source software

# Proprietary vs Open Source

- **Full control** of the program:
  you know, what is inside

LabView

LabGui

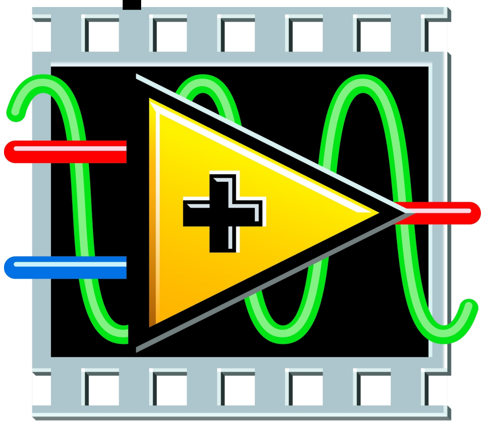
# Proprietary vs Open Source

- **Full control** of the program:
  you know, what is inside
- **It’s free**

LabView

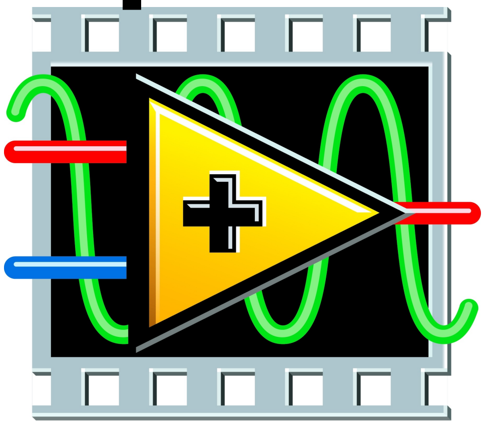
LabGui

# Proprietary vs Open Source



- Use **any text editor** to modify your code

# Proprietary vs Open Source



- Use **any text editor** to modify your code no need to fight with your colleagues for the computer, where the latest version of LabView is installed

# LabView compatibility issues

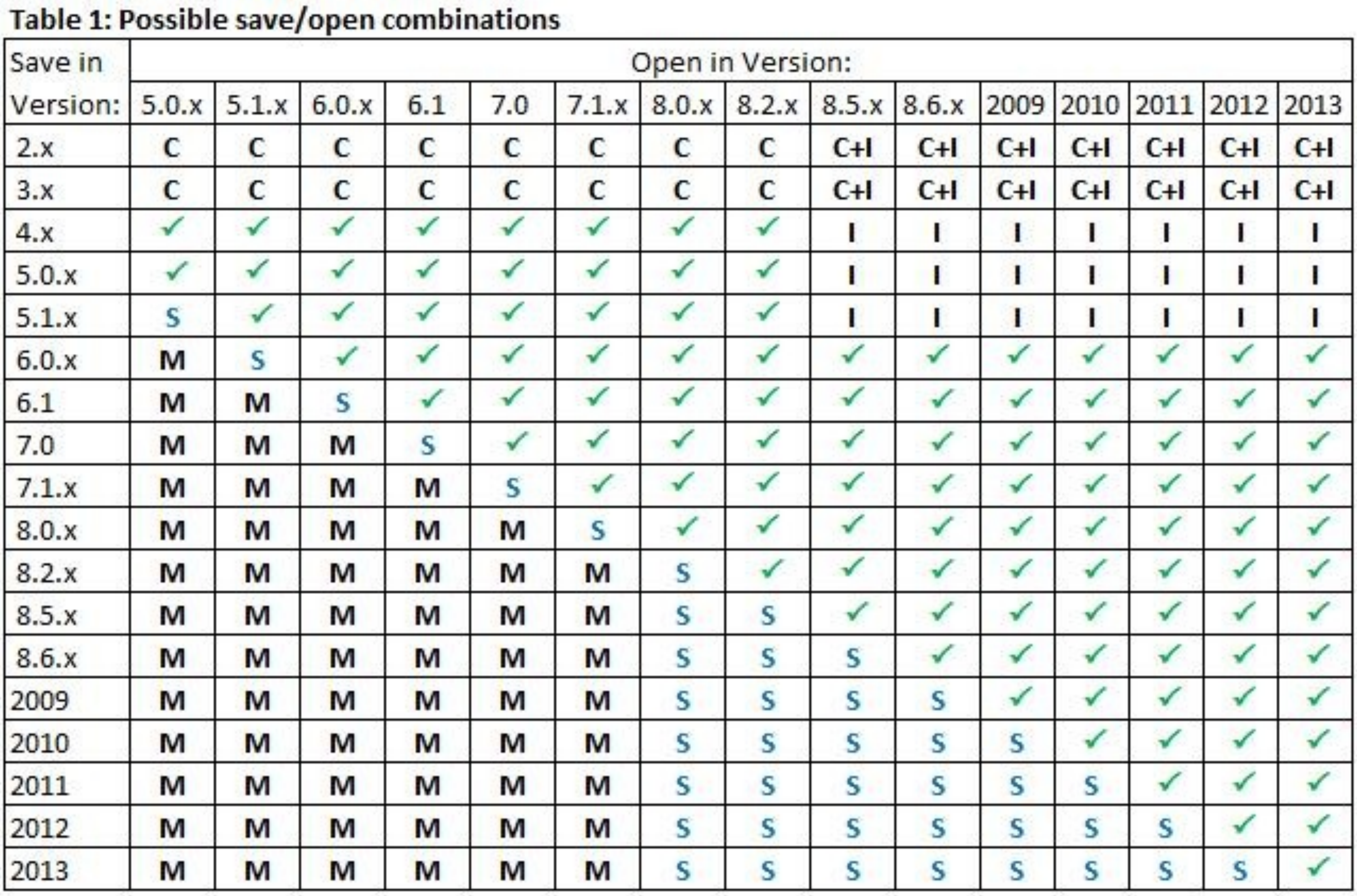


Table 1: Possible save/open combinations

| Save in | Open in Version: | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Version: | 5.0.x | 5.1.x | 6.0.x | 6.1 | 7.0 | 7.1.x | 8.0.x | 8.2.x | 8.5.x | 8.6.x | 2009 | 2010 | 2011 | 2012 | 2013 |
| 2.x | C | C | C | C | C | C | C | C | C+I | C+I | C+I | C+I | C+I | C+I | C+I |
| 3.x | C | C | C | C | C | C | C | C | C+I | C+I | C+I | C+I | C+I | C+I | C+I |
| 4.x | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | I | I | I | I | I | I | I |
| 5.0.x | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | I | I | I | I | I | I | I |
| 5.1.x | S | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | I | I | I | I | I | I | I |
| 6.0.x | M | S | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| 6.1 | M | M | S | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| 7.0 | M | M | M | S | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| 7.1.x | M | M | M | M | S | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| 8.0.x | M | M | M | M | M | S | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| 8.2.x | M | M | M | M | M | M | S | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| 8.5.x | M | M | M | M | M | M | S | S | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| 8.6.x | M | M | M | M | M | M | S | S | S | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| 2009 | M | M | M | M | M | M | S | S | S | S | ✓ | ✓ | ✓ | ✓ | ✓ |
| 2010 | M | M | M | M | M | M | S | S | S | S | S | ✓ | ✓ | ✓ | ✓ |
| 2011 | M | M | M | M | M | M | S | S | S | S | S | S | ✓ | ✓ | ✓ |
| 2012 | M | M | M | M | M | M | S | S | S | S | S | S | S | ✓ | ✓ |
| 2013 | M | M | M | M | M | M | S | S | S | S | S | S | S | S | ✓ |

# LabView compatibility issues

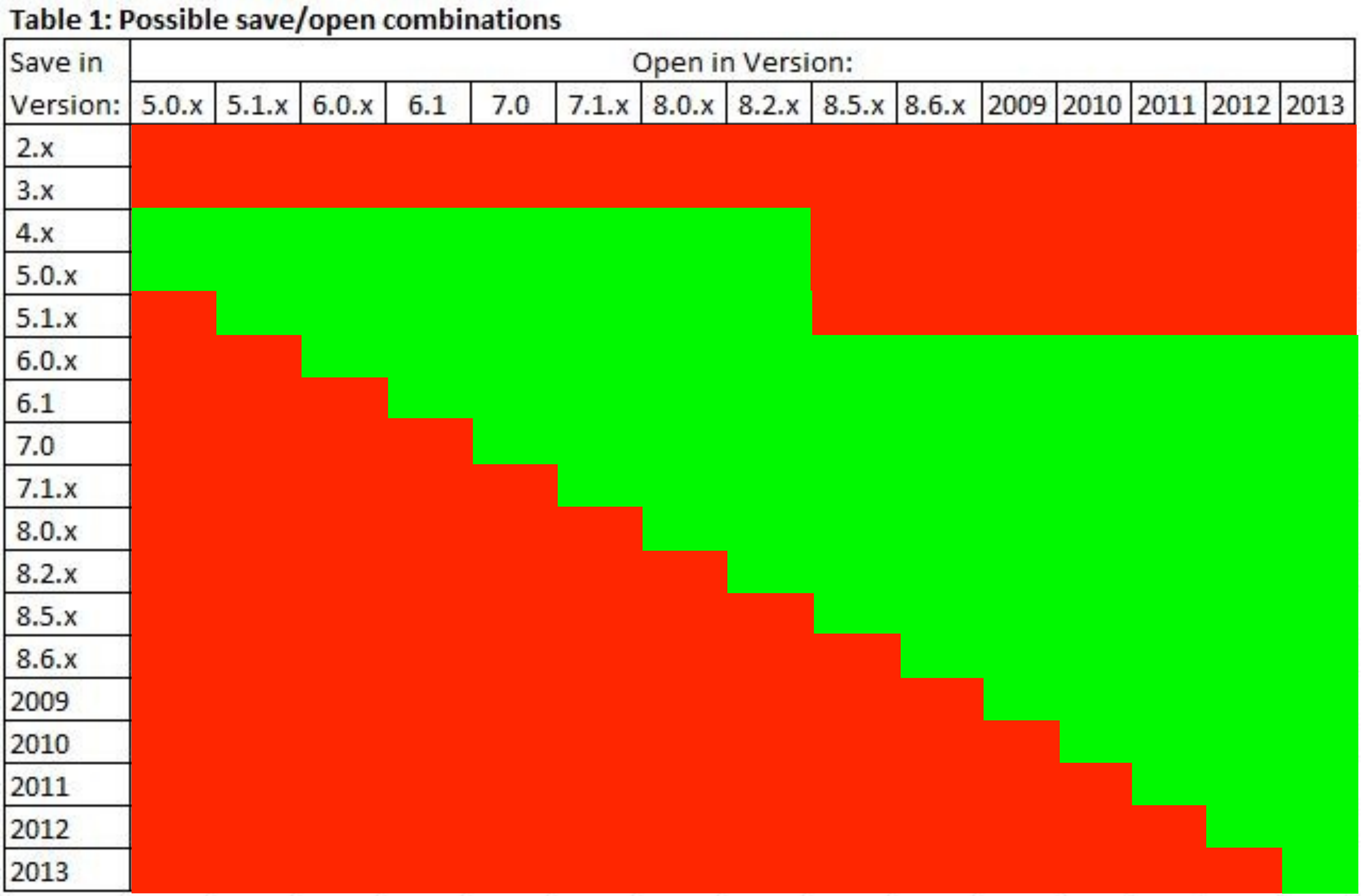


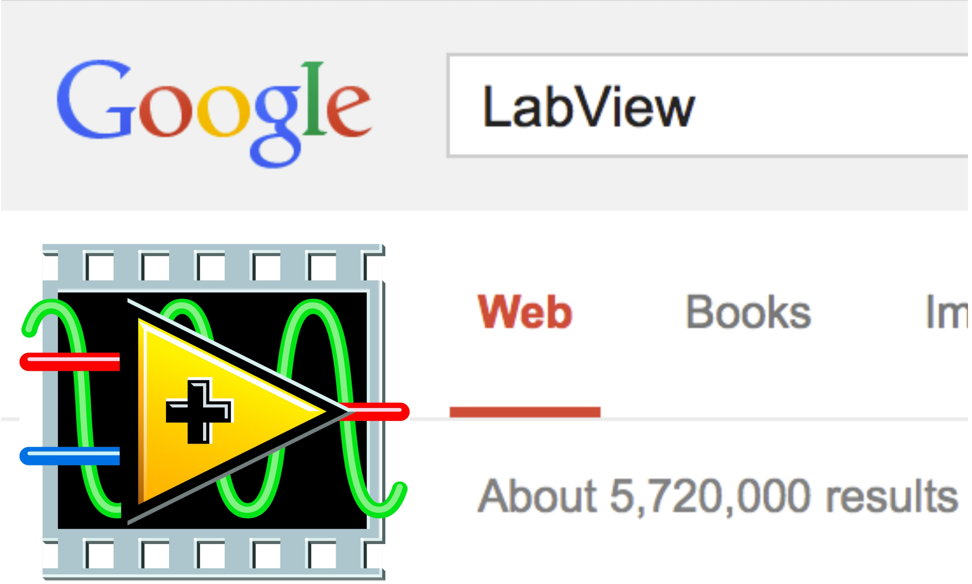
Table 1: Possible save/open combinations

| Save in | Open in Version: | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Version: | 5.0.x | 5.1.x | 6.0.x | 6.1 | 7.0 | 7.1.x | 8.0.x | 8.2.x | 8.5.x | 8.6.x | 2009 | 2010 | 2011 | 2012 | 2013 |
| 2.x | 🟥 | 🟥 | 🟥 | 🟥 | 🟥 | 🟥 | 🟥 | 🟥 | 🟥 | 🟥 | 🟥 | 🟥 | 🟥 | 🟥 | 🟥 |
| 3.x | 🟥 | 🟥 | 🟥 | 🟥 | 🟥 | 🟥 | 🟥 | 🟥 | 🟥 | 🟥 | 🟥 | 🟥 | 🟥 | 🟥 | 🟥 |
| 4.x | 🟩 | 🟩 | 🟩 | 🟩 | 🟩 | 🟩 | 🟩 | 🟩 | 🟥 | 🟥 | 🟥 | 🟥 | 🟥 | 🟥 | 🟥 |
| 5.0.x | 🟩 | 🟩 | 🟩 | 🟩 | 🟩 | 🟩 | 🟩 | 🟩 | 🟥 | 🟥 | 🟥 | 🟥 | 🟥 | 🟥 | 🟥 |
| 5.1.x | 🟥 | 🟩 | 🟩 | 🟩 | 🟩 | 🟩 | 🟩 | 🟩 | 🟥 | 🟥 | 🟥 | 🟥 | 🟥 | 🟥 | 🟥 |
| 6.0.x | 🟥 | 🟥 | 🟩 | 🟩 | 🟩 | 🟩 | 🟩 | 🟩 | 🟩 | 🟩 | 🟩 | 🟩 | 🟩 | 🟩 | 🟩 |
| 6.1 | 🟥 | 🟥 | 🟥 | 🟩 | 🟩 | 🟩 | 🟩 | 🟩 | 🟩 | 🟩 | 🟩 | 🟩 | 🟩 | 🟩 | 🟩 |
| 7.0 | 🟥 | 🟥 | 🟥 | 🟥 | 🟩 | 🟩 | 🟩 | 🟩 | 🟩 | 🟩 | 🟩 | 🟩 | 🟩 | 🟩 | 🟩 |
| 7.1.x | 🟥 | 🟥 | 🟥 | 🟥 | 🟥 | 🟩 | 🟩 | 🟩 | 🟩 | 🟩 | 🟩 | 🟩 | 🟩 | 🟩 | 🟩 |
| 8.0.x | 🟥 | 🟥 | 🟥 | 🟥 | 🟥 | 🟥 | 🟩 | 🟩 | 🟩 | 🟩 | 🟩 | 🟩 | 🟩 | 🟩 | 🟩 |
| 8.2.x | 🟥 | 🟥 | 🟥 | 🟥 | 🟥 | 🟥 | 🟥 | 🟩 | 🟩 | 🟩 | 🟩 | 🟩 | 🟩 | 🟩 | 🟩 |
| 8.5.x | 🟥 | 🟥 | 🟥 | 🟥 | 🟥 | 🟥 | 🟥 | 🟥 | 🟩 | 🟩 | 🟩 | 🟩 | 🟩 | 🟩 | 🟩 |
| 8.6.x | 🟥 | 🟥 | 🟥 | 🟥 | 🟥 | 🟥 | 🟥 | 🟥 | 🟥 | 🟩 | 🟩 | 🟩 | 🟩 | 🟩 | 🟩 |
| 2009 | 🟥 | 🟥 | 🟥 | 🟥 | 🟥 | 🟥 | 🟥 | 🟥 | 🟥 | 🟥 | 🟩 | 🟩 | 🟩 | 🟩 | 🟩 |
| 2010 | 🟥 | 🟥 | 🟥 | 🟥 | 🟥 | 🟥 | 🟥 | 🟥 | 🟥 | 🟥 | 🟥 | 🟩 | 🟩 | 🟩 | 🟩 |
| 2011 | 🟥 | 🟥 | 🟥 | 🟥 | 🟥 | 🟥 | 🟥 | 🟥 | 🟥 | 🟥 | 🟥 | 🟥 | 🟩 | 🟩 | 🟩 |
| 2012 | 🟥 | 🟥 | 🟥 | 🟥 | 🟥 | 🟥 | 🟥 | 🟥 | 🟥 | 🟥 | 🟥 | 🟥 | 🟥 | 🟩 | 🟩 |
| 2013 | 🟥 | 🟥 | 🟥 | 🟥 | 🟥 | 🟥 | 🟥 | 🟥 | 🟥 | 🟥 | 🟥 | 🟥 | 🟥 | 🟥 | 🟩 |

# Proprietary vs Open Source

Use all the power and support:

- **HUGE Python community**

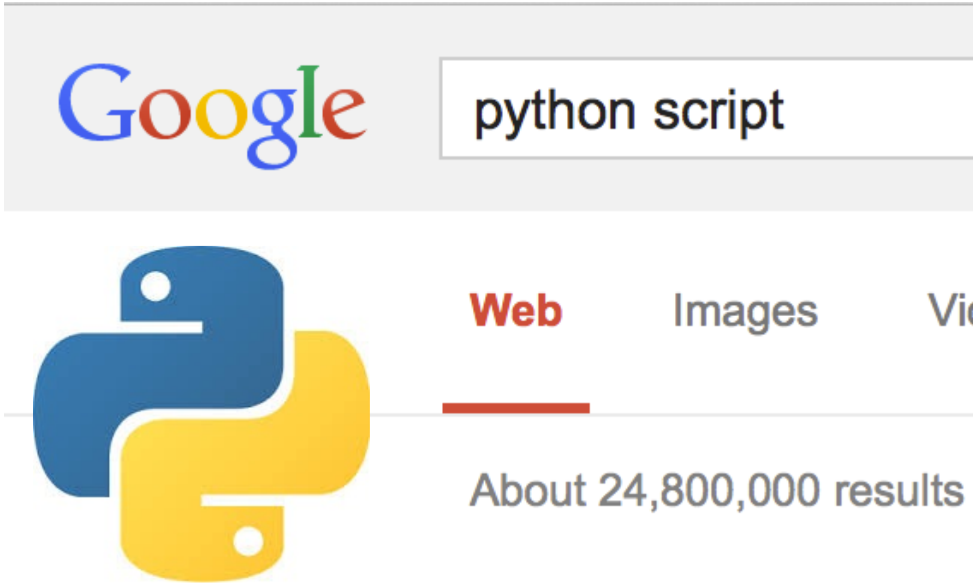


Google

LabView

About 5,720,000 results

**6 millions**



Google

python script

About 24,800,000 results

**25 millions**

# Proprietary vs Open Source

Use all the power and support:

- **HUGE Python community**
- **Growing scientific community**

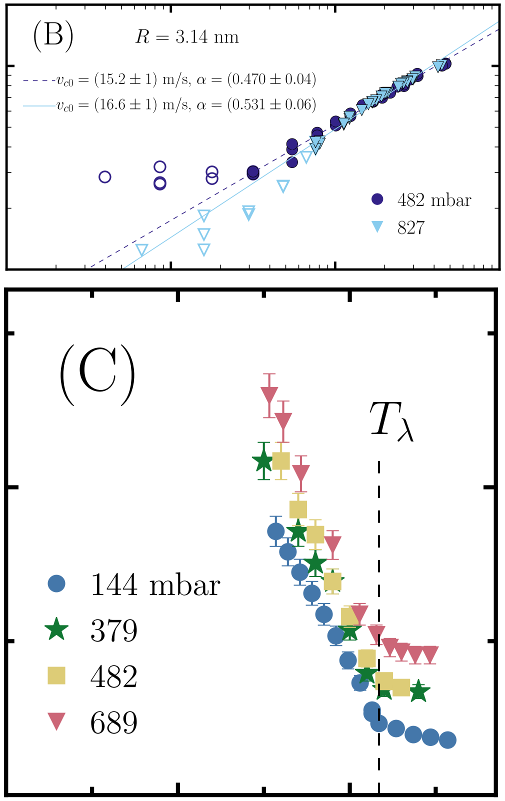
# Proprietary vs Open Source

Use all the power and support:

- **HUGE Python community**
- **Growing scientific community**

## List of papers that used **LabGui**

- B. A. Schmidt, K. Bennaceur, S. Bilodeau, K. W. West, L. N. Pfeiffer, G. Gervais, “5/2 Fractional Quantum Hall Effect in the Corbino Geometry”, http://arxiv.org/abs/1503.07775
- P-F Duc, M.Savard, M. Petrescu, B. Rosenow, A. Del Maestro, and G. Gervais, “Critical Flow and Dissipation in a Quasi-One-Dimensional Superfluid”, http://arxiv.org/abs/1412.5124, accepted in Science Advances
- V. Tayari, N. Hemsworth, I. Fakih, A. Favron, E. Gaufres, G. Gervais, R. Martel, T. Szkopek, “Two-Dimensional Magnetotransport in a Black Phosphorus Naked Quantum Well”, http://arxiv.org/abs/1412.0259, accepted in Nature Communications



# Summary

# LabView & LabGui

# Summary LabView & LabGui

| | | |
|---|---|---|
| ALL devices speak ONE language | ✓ | ✓ |
| Language | GPL | PL |
| Complex projects easy | ✗ | ✓ |
| Version control | ✗ | ✓ |
| Accessibility | License | Open Source |
| Easy compatibility | ✗ | ✓ |
| Full control | ✗ | ✓ |

# Download LabGui

https://bitbucket.org/RTechMcGill/labgui/src

RTech www.facebook.com/RTechMcGill www.hep.physics.mcgill.ca/RTech

# *Automating your experiment with LabGui*

## May 19, Tuesday, noon

**3600 University, Rutherford Physics Building, room 103**

**We have funds to help you cover transportation costs. If interested contact us at:**

**rtech@physics.mcgill.ca**

**https://www.facebook.com/RTechMcGill**

www.hep.physics.mcgill.ca/RTech

facebook.com/RTechMcGill

# RTech

Video created by:
Pierre-François Duc & Igor Kozlov

with contributions by:
Julien Lhermitte
Anna Mkrtchyan
Hélène Seiler