# Example of BG51 data acquisition in micropython

bg51 test program

# 1 Introduction

This document shows how the LoPy, using micropython, can interface the bg51 radiation sensor through the Grove expansion board.

## 1.1 Basic requirements of the acquisition program

The program implement these basic functions:

1) the LoPy  acquire the pulses generated by the bg51 sensor as quickly as possible

2) the reading of new impulses causes the production of sound "ticks", obtained activating a buzzer for a very short time

3) using the counts of pulses in a minute, we calculate the value of the Radiation dose equivalent rate in mSv/h (see graph on bg51 documetation)

4) periodically, the REPL terminal display must display the number of pulses per minute and the Radiation value dose equivalent rate

## 1.2 Parameters in the bg51 documentation

the BG51 technical data document:

https://www.teviso.com/file/pdf/bg51-data-specification.pdf

provides useful information for the development of the program

### 1.2.1 duration of the pulses generated by the sensor

Electrical characteristics (page 2):

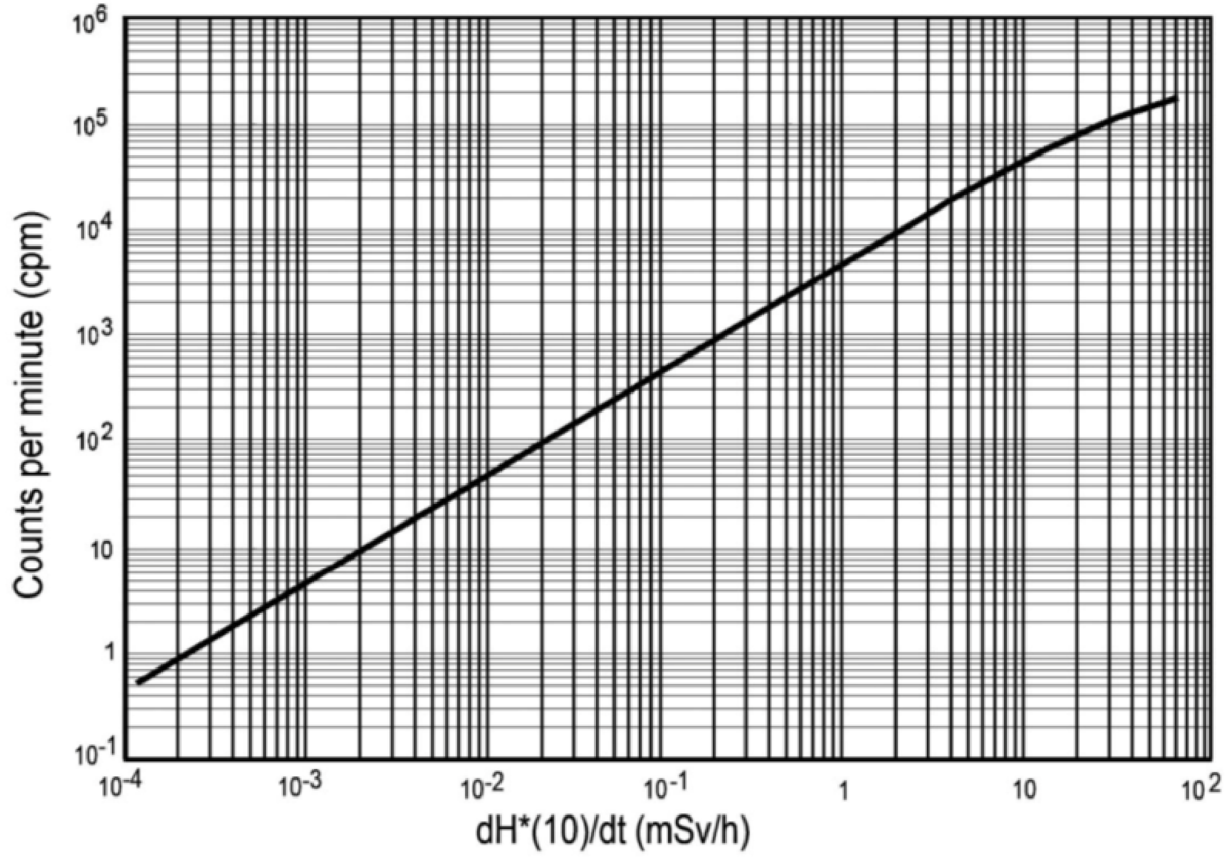
**Output pulse width: 50 µs to 200 µs (LOW→HIGH→LOW)**

### 1.2.2 BG51 Sensor Linearity (page 2)

On page 2 there is a graph with axes in logarithmic scale that relates the number of pulses per minute with Radiation dose equivalent rate for Cs-

137 and Co-60 (mSv / h).

The graph data will be used to construct the conversion function between pulses and dose Radiation.

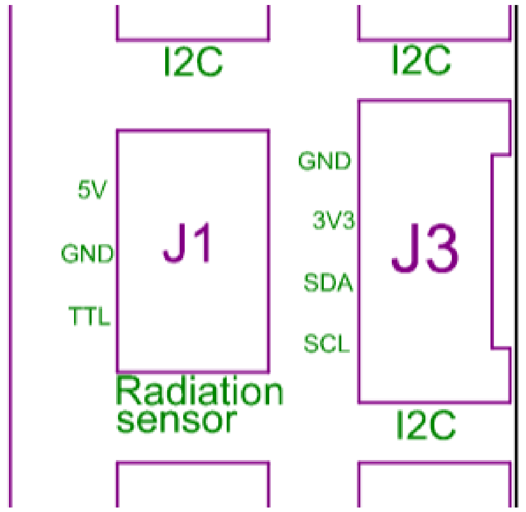


## 1.3 How to read the BG51 pulses with LoPy

The sensor must be connected to the J1 connector on the Grove board.

**Its output signal is connected to the LoPy pin labeled P17.**

In the program, pin P17 must be set as digital input.



### 1.3.1 pin P17 set as digital input

from:

https://docs.pycom.io/chapter/firmwareapi/pycom/machine/Pin.html

we can see how to set the pin as digital input.

## bg51 test program

```
from machine import Pin
….
# radiation sensor to P17 input Pin
sigBg51 = Pin("P17", mode=Pin.IN, pull=None)
```

In this case, these settings are made for pin P17

1. Pin.IN: the pin is input
2. pull=None: no pull up or down resistor.

Note:

If there is nothing connected to the pin and the program reads the state of the pin, the pull-up resistor guarantee that the read will it be high (pulled to VCC).

The pull-down instead, guarantees that the value read is low (to ground).

In our case, the input is set to None, because it is directly connected to the sensor digital output

## 1.4 Define an interrupt to count the BG51 pulses

In general, an interrupt is a signal to the processor indicating an event that needs immediate attention.

An interrupt alerts the processor to a high-priority condition requiring the stop the execution of the current flow of program instructions.

The processor handles the interrupt in this way:

1. suspend its current activities,
2. save its state,
3. execute a function called **interrupt service routine (ISR)** to deal with the event.
4. at the end of ISR, restore the status and continue with the previously interrupted code

To manage an input pin via Interrupt, see:

https://docs.pycom.io/chapter/firmwareapi/pycom/machine/Pin.html

## pin.callback(trigger, handler=None, arg=None)

Set a callback to be triggered when the input level at the pin changes.

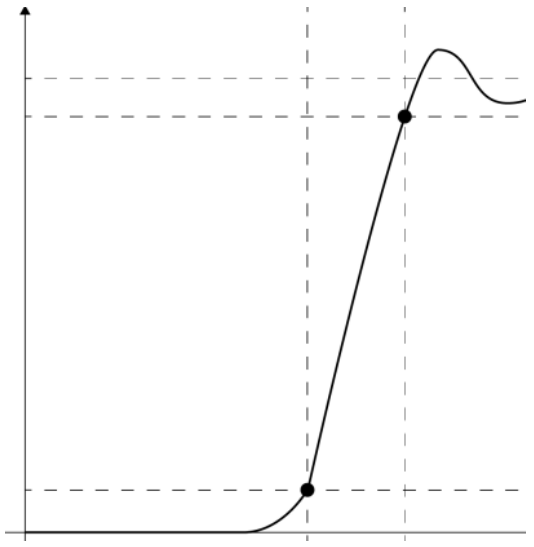
- trigger is the type of event that triggers the callback. Possible values are:
    - interrupt on falling edge.
    - interrupt on rising edge.
    - interrupt on low level.
    - interrupt on high level.

The values can be OR-ed together, for instance trigger=Pin.IRQ_FALLING | Pin.IRQ_RISING

- **handler is the function to be called when the event happens. This function will receive one argument. Set handler to None to disable it.**
- arg is an optional argument to pass to the callback. If left empty or set to None, the function will receive the Pin object that triggered it.

**In our case:**

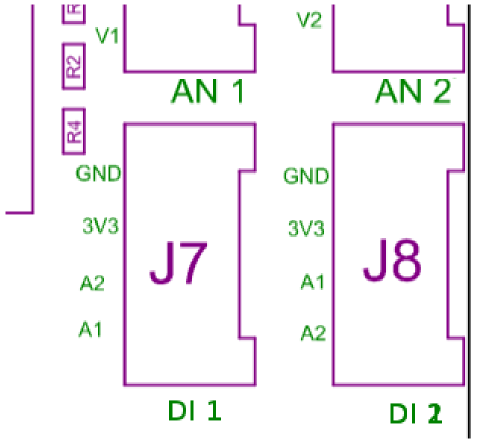
```
# BG51 rise signal interrupt routine.
def riseBg51(pin):
    global counter
    counter = counter + 1
...
# the rising edge of input sigBg51
# generate an Interrupt, managed by riseBg51
sigBg51.callback(Pin.IRQ_RISING, riseBg51)
```



## 1.5 Buzzer management

The buzzer is used to generate ticks when we read the pulses from the radiation sensor.

The buzzer is controlled by a digital output and must therefore be connected to one of the Grove board's digital I/O connectors (J7 or J8)



| Connector | LoPy I/O Pin |
|---|---|
| J7 | 'P12' |
| J8 | 'P11' |

from:

https://docs.pycom.io/chapter/firmwareapi/pycom/machine/Pin.html

we can see how to set the pin as digital output.

```
from machine import Pin

. . .
# buzzer to digital connector J8
#  J7 connector: to I/O Pin 'P12'
buz = Pin('P11', mode=Pin.OUT)          #  J8 connector: to I/O Pin 'P11'
```

Code example to generate a ‘tick’

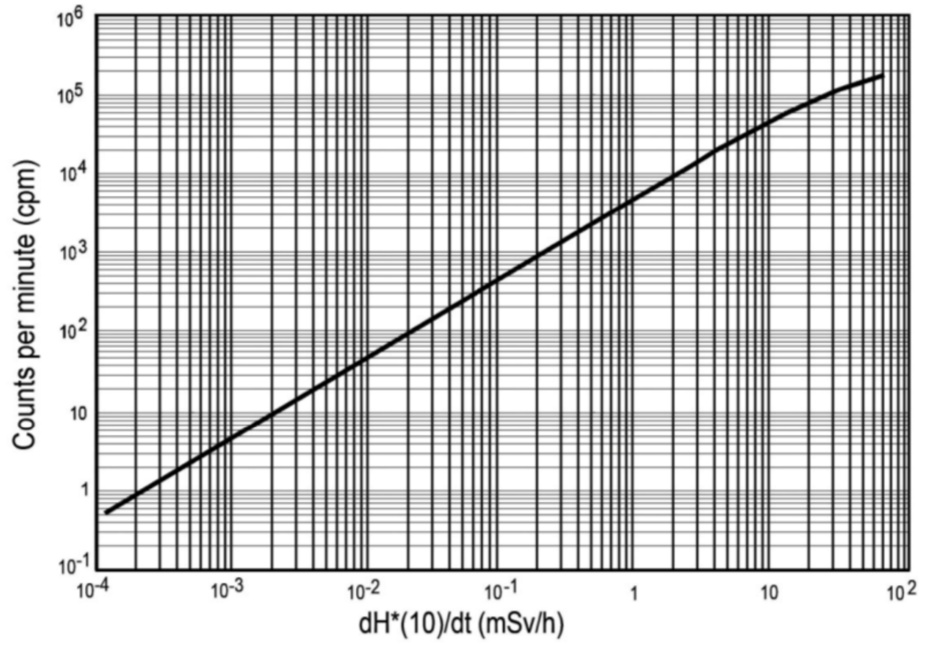
```
        buz.value(1)                                # buzzer on
        for i in range(0,10):                       # loop to make a fast delay
            pass                                    #    no operation
        buz.value(0)                                # buzzer off
```

## 1.6 Conversion of Pulses per minute to mSv/h

the algorithm implemented in the conversion that uses a Lookup table.

In this software, the table is made using two lists of sorted values:

1. pls_min which contains the pulses per minute
2. mSv_h contains the corresponding values of mSv/h



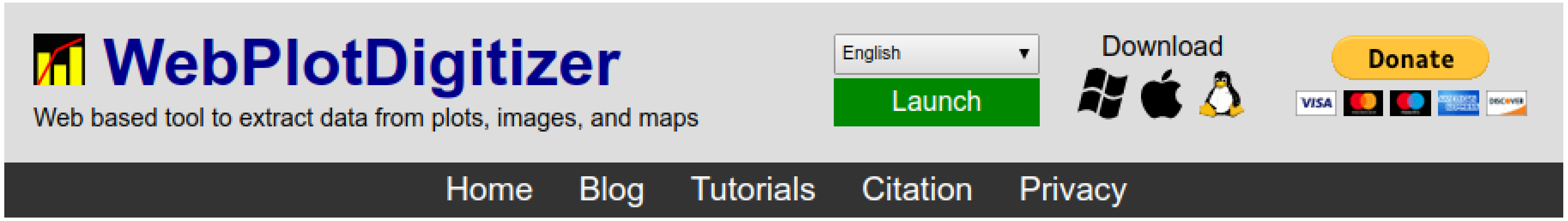
The values of the two lists have been obtained from the graph provided in the sensor documentation

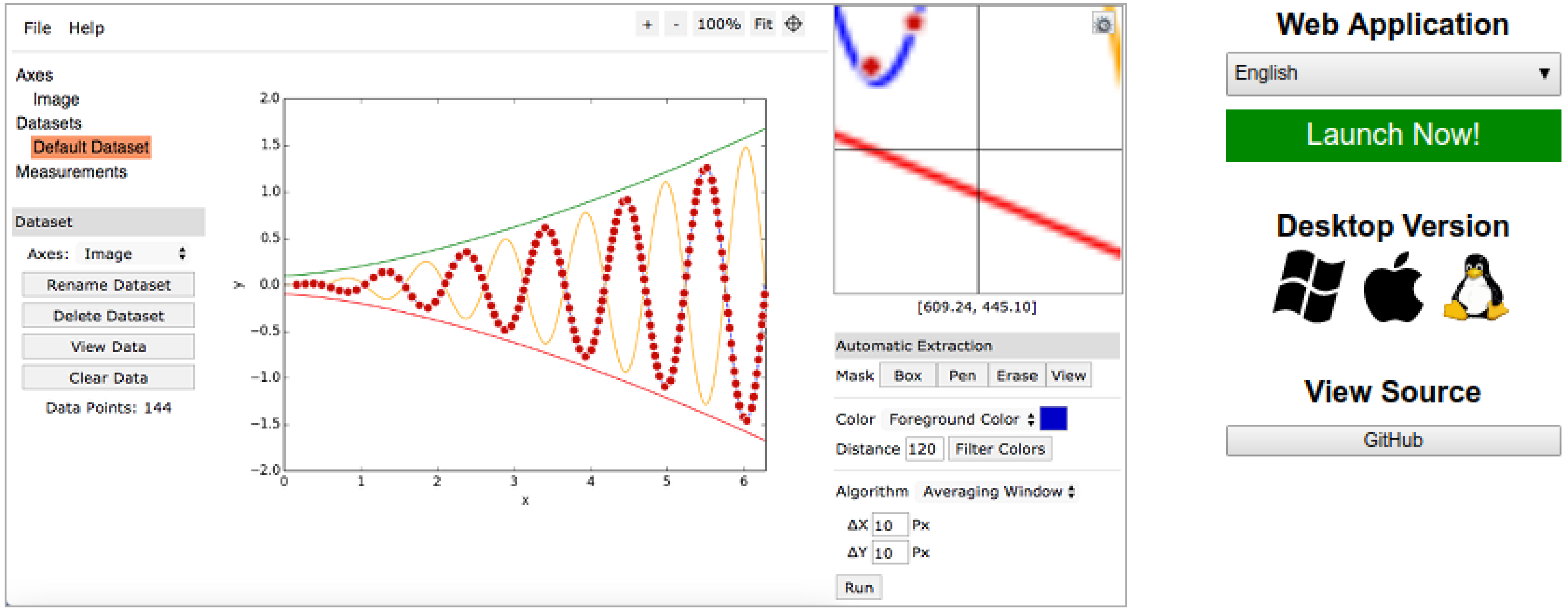
The conversion algorithm from ***pulses*** to mSv/h is represented here in pseudocode:

```
Step1:
        in pls_min list, use binary search to find the index element where:
            min( (pulses - pls_min[index]) >= 0)
Step2:
        use the equation of the line passing from these two Points:
          1.    A = (pls_min[index],  mSv_h[index])
          2.    B = (pls_min[index+1],  mSv_h[index+1])
        to calculate the mSv/h from pulses value
```

## 2 Software tool to extract the underlying numerical data from graphs

**https://automeris.io/WebPlotDigitizer/**





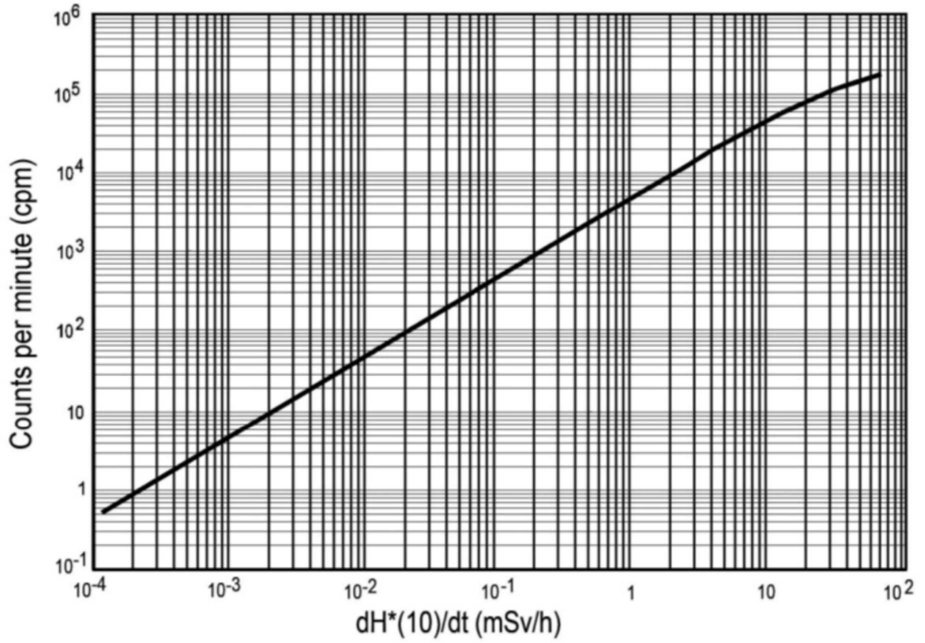
WebPlotDigitizer is a semi-automated tool that help extract data from graphs:

- Works with a wide variety of charts (XY, bar, polar, ternary, maps etc.)
- Automatic extraction algorithms make it easy to extract a large number of data points
- Free to use, opensource and cross-platform (web and desktop)

## 2.1 First step: pre-process the graph

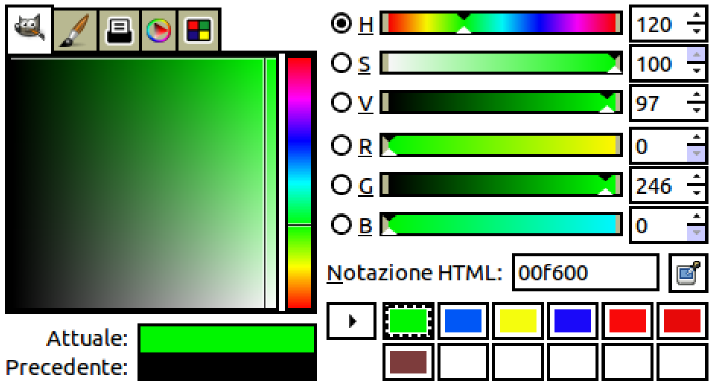
Using the original graph from sensor documentation, we need to highlight the sensor curve using another color

To do this, we use for example Gimp.



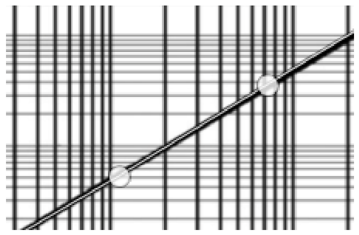
In Gimp:

1. File → Open
2. choose the graph image file (example:bg51graph.png)
3. Color: change the foreground color (for example pass to Green)

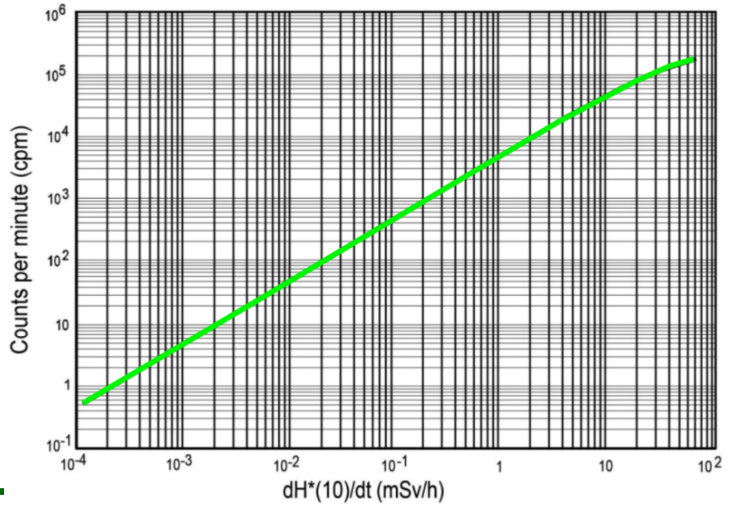


H 120
S 100
V 97
R 0
G 246
B 0
Notazione HTML: 00f600
Attuale:
Precedente:

4. In main menu, choose Instruments → paths
5. Using mouse track a path on sensor curve



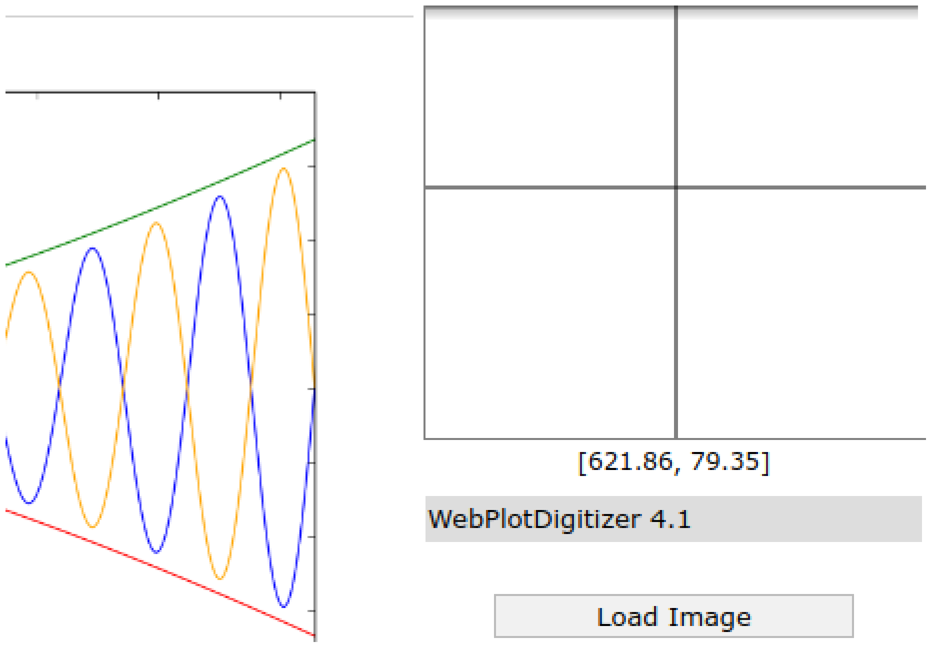
6. Use the command path-stroke to draw the line on path with the foreground color (green)
7. Save the result as a new image (example:bg51graph2.png)

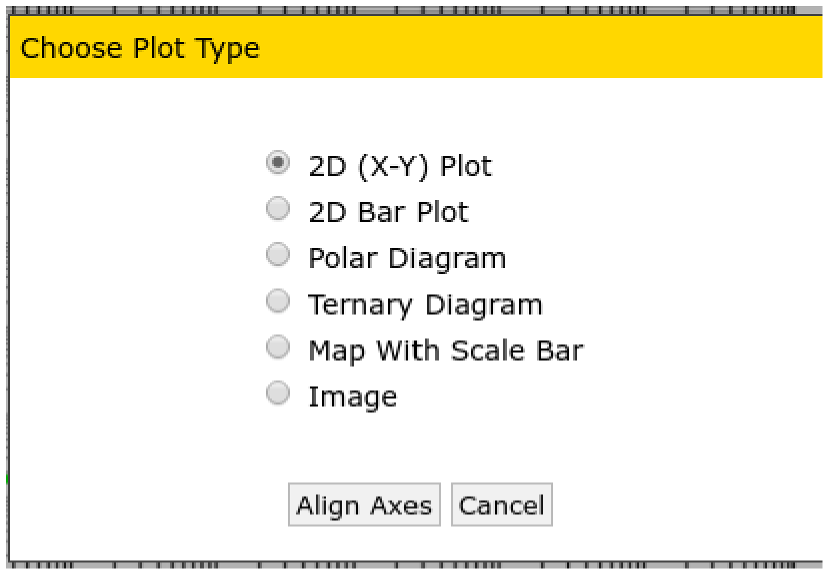


## 2.2 Second step: use WebPlotDigitizer to generate a CSV file

In main page, press button “Launch now”

Load Image file (bg51graph2.png)

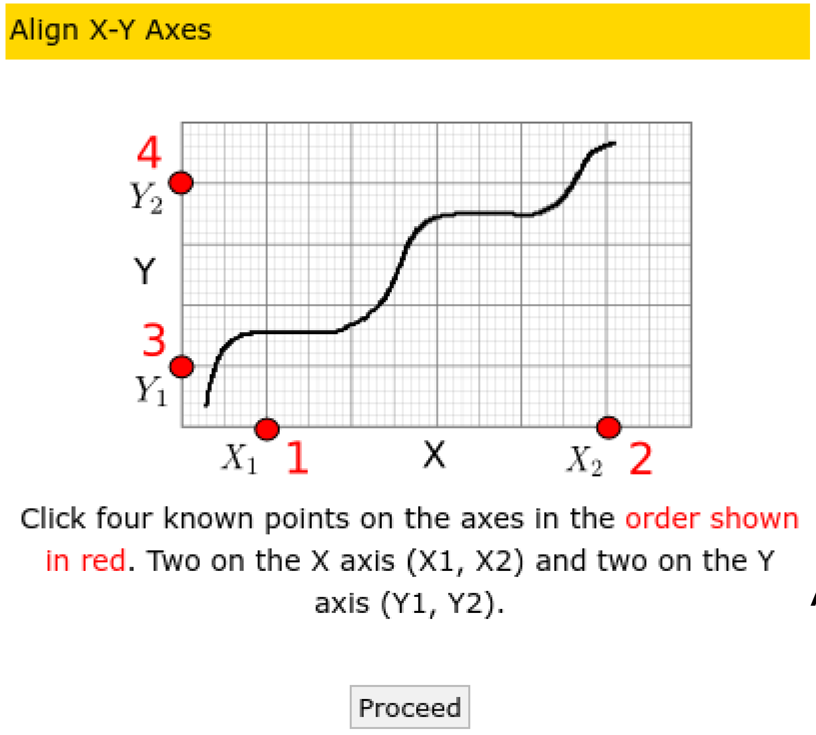



Choose Plot Type

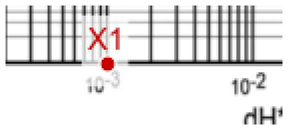
[✓ 2D (X-Y) Plot]
2D Bar Plot
Polar Diagram
Ternary Diagram
Map With Scale Bar
Image

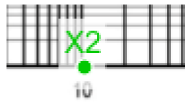
Choose 2D Plot - Align Plot

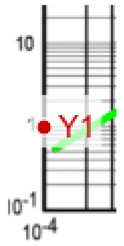
# bg51 test program


Align X-Y Axes

Click four known points on the axes in the order shown in red. Two on the X axis (X1, X2) and two on the Y axis (Y1, Y2).



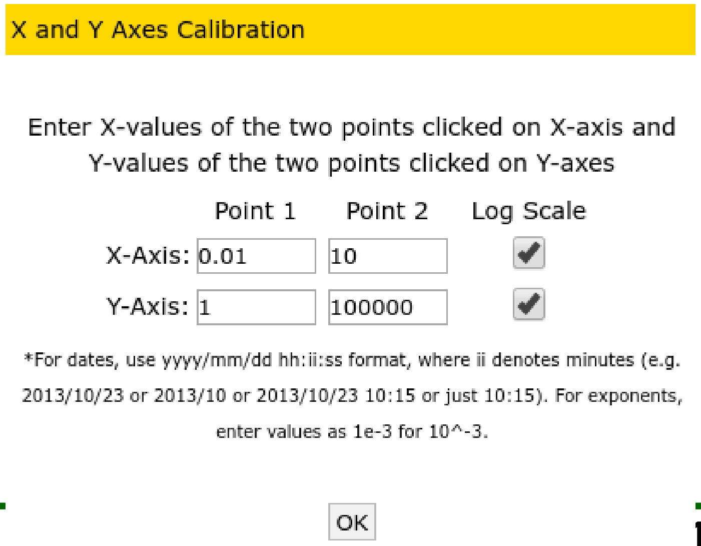




At the end, press Complete

Complete!

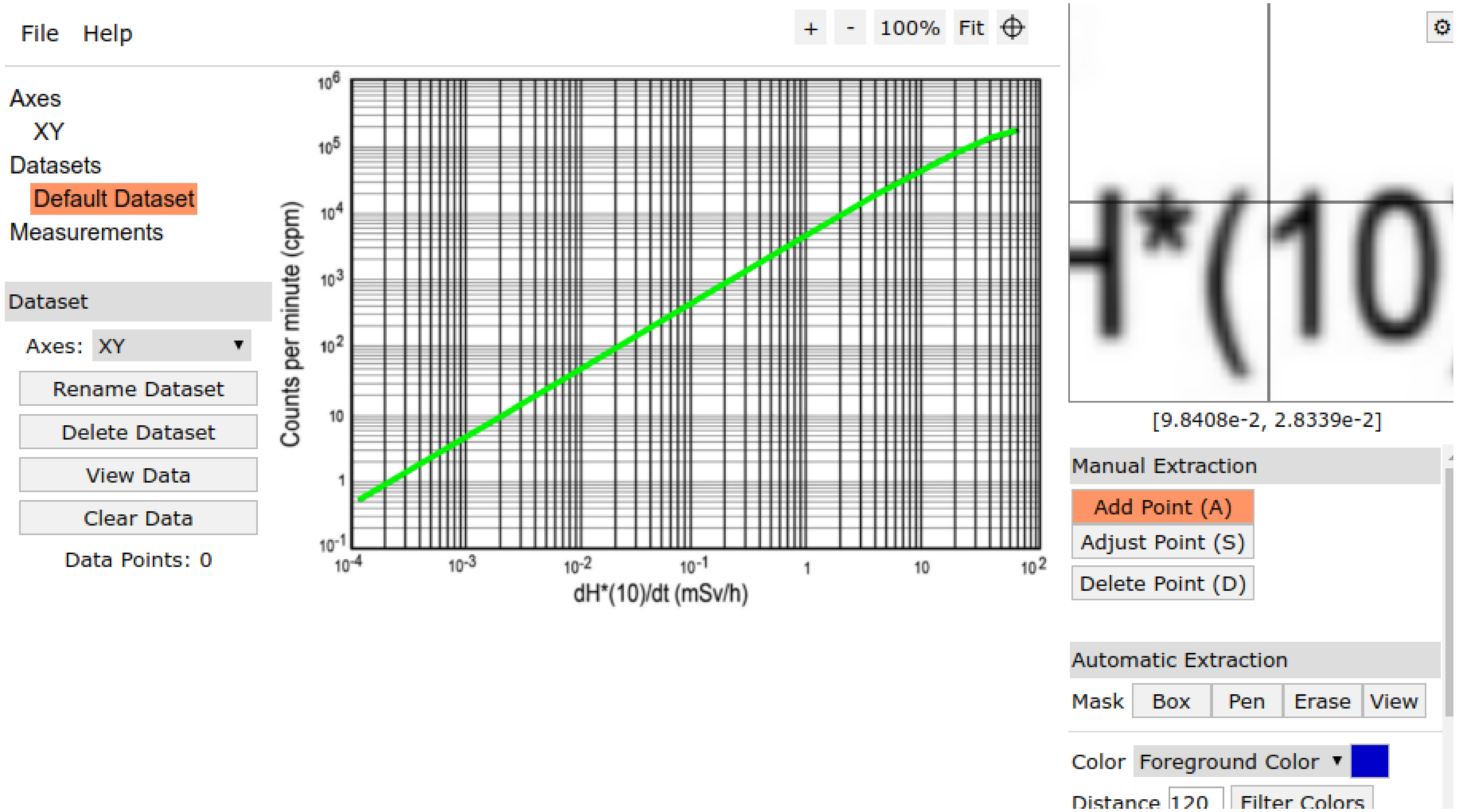
Set the marker values


X and Y Axes Calibration

Enter X-values of the two points clicked on X-axis and Y-values of the two points clicked on Y-axes

| | Point 1 | Point 2 | Log Scale |
|---|---|---|---|
| X-Axis: | 0.01 | 10 | ✓ |
| Y-Axis: | 1 | 100000 | ✓ |

*For dates, use yyyy/mm/dd hh:ii:ss format, where ii denotes minutes (e.g. 2013/10/23 or 2013/10 or 2013/10/23 10:15 or just 10:15). For exponents, enter values as 1e-3 for 10^-3.



Axes: XY

Data Points: 0

[9.8408e-2, 2.8339e-2]

Color: Foreground Color

Distance: 120

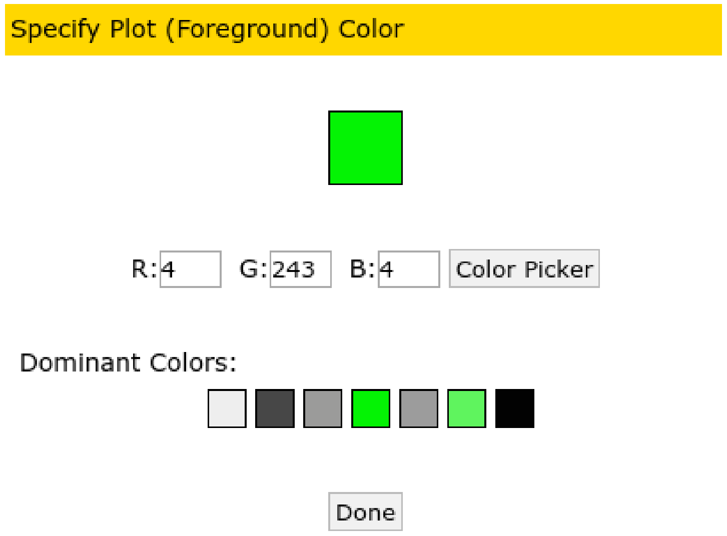
Select Automatic Extraction by color.

## bg51 test program

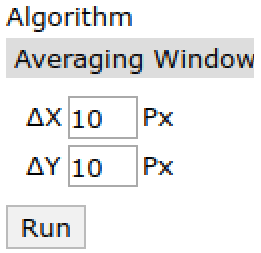
Press on Foreground Color box and select the color of data curve (Green)

Press Done

and press Run (at bottom of page)


Specify Plot (Foreground) Color

R: 4 G: 243 B: 4

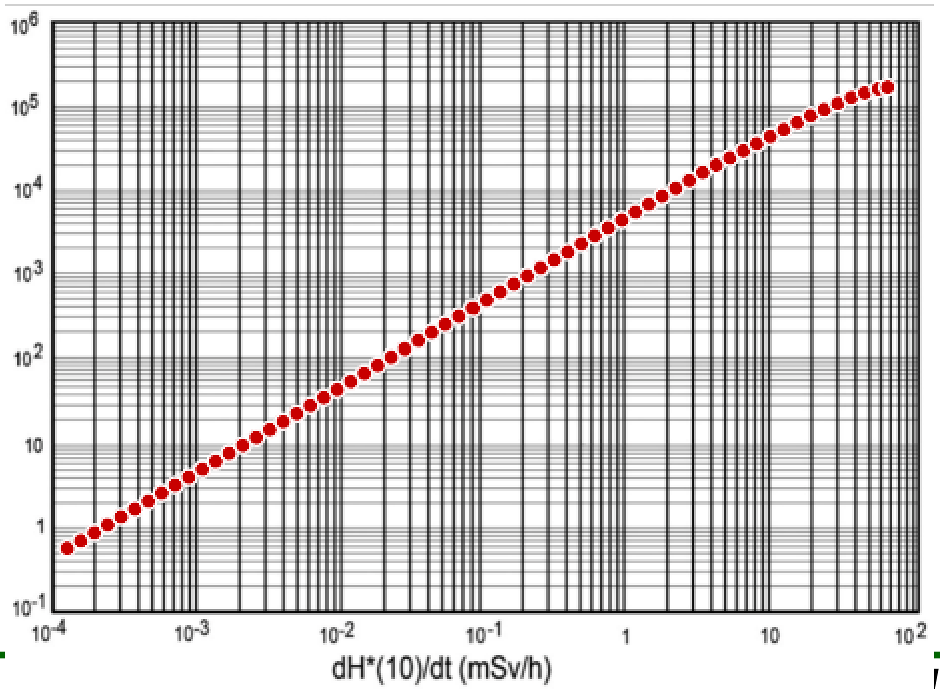

Algorithm

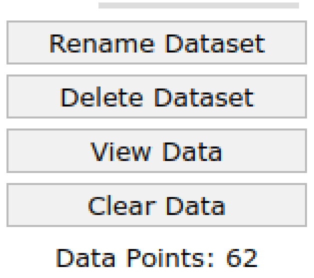
Averaging Window

ΔX 10 Px

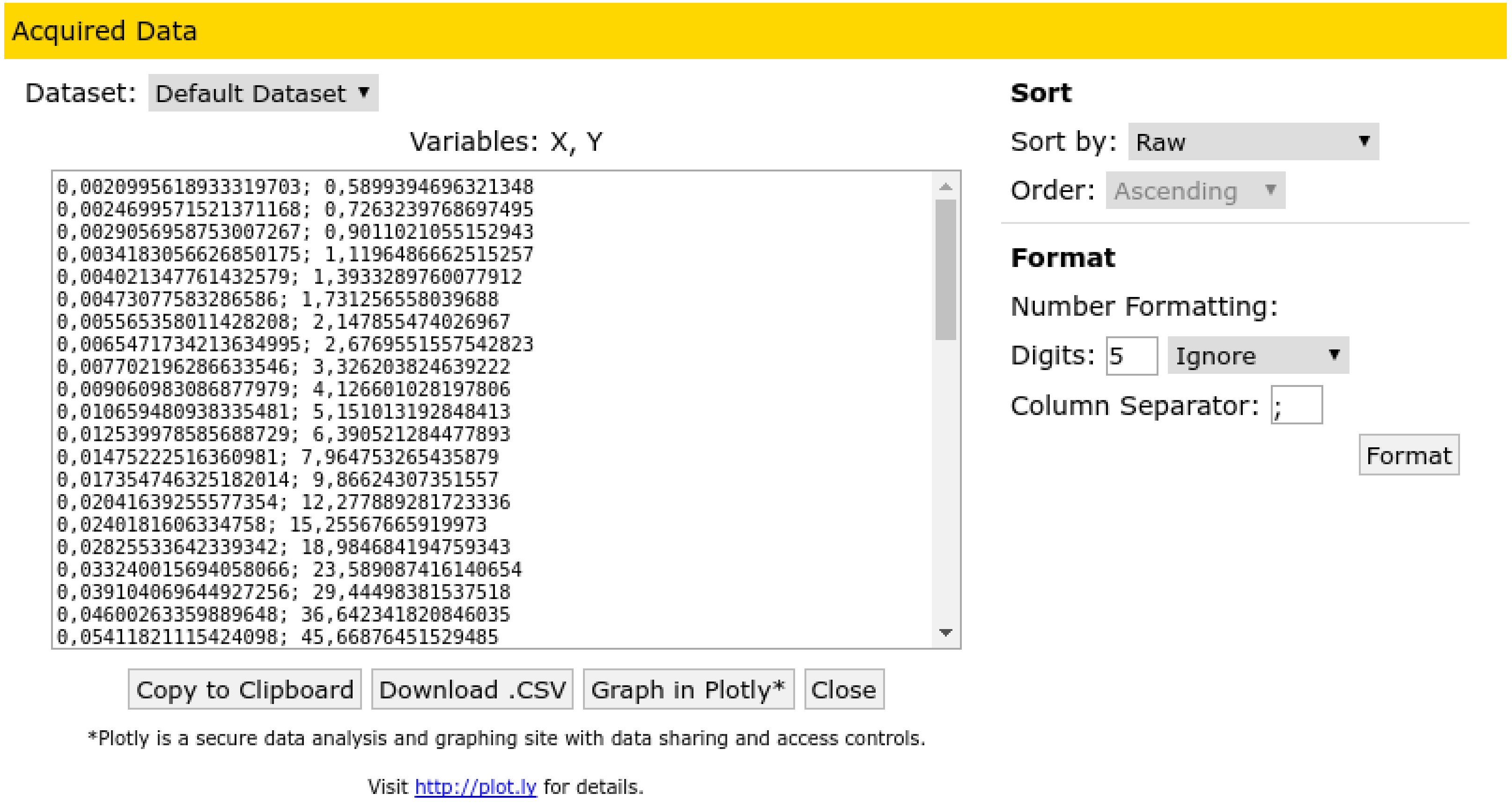
ΔY 10 Px

The web application generates a series of points on image chart




Data Points: 62

On right, press “View Data”



Variables: X, Y

```
0,0020995618933319703; 0,5899394696321348
0,0024699571521371168; 0,7263239768697495
0,0029056958753007267; 0,9011021055152943
0,0034183056626850175; 1,1196486662515257
0,004021347761432579; 1,3933289760077912
0,00473077583286586; 1,731256558039688
0,005565358011428208; 2,147855474026967
0,0065471734213634995; 2,6769551557542823
0,007702196286633546; 3,326203824639222
0,009060983086877979; 4,126601028197806
0,010659480938335481; 5,151013192848413
0,012539978585688729; 6,390521284477893
0,01475222516360981; 7,964753265435879
0,017354746325182014; 9,86624307351557
0,02041639255577354; 12,277889281723336
0,0240181606334758; 15,25567665919973
0,02825533642339342; 18,984684194759343
0,033240015694058066; 23,589087416140654
0,039104069644927256; 29,44498381537518
0,04600263359889648; 36,642341820846035
0,05411821115424098; 45,66876451529485
```

*Plotly is a secure data analysis and graphing site with data sharing and access controls.

Visit http://plot.ly for details.

Press Download .CSV to save the dataset in a file for further elaborations.