Control Software user guide

Author:
André Silva

Student Number:
up201403977

July 19, 2018

Contents

1 Brief introduction to the Software

This sofware was built as a way to provide users with a better and easier control of the testing processes of triboelectric materials. It allows the users to choose a single resistor to activate and test the material, or choose multiple resistors and

2 Prerequisites for correct usage

2.1 Operating system

The software was built on a Windows platform and wasn't tested on other operating systems. However it's is already known that the oscilloscope interface will not work on Linux systems synce, at the time of writting, the Tektonix API is different.

2.2 Arduino

This software uses the Firmata protocol to control the arduino's pins, allowing the activation or deactivation of the relays. It was built in an option to send the protocol to the arduino, only supporting arduino *Nano* and *Uno*. However, it should be possible to use other types of arduinos to control the relays, as long as they have the Firmata protocol.

2.3 Oscilloscope

The oscilloscope connection was built upon the *Tektronix* API, using open source python libraries. By default, and at the time of writting, there is only support for models described on the *Tektronix* programmer guide. It is important to say that there is a limited number of models for which the API works [inserir referencia ao manual da tektronix].

2.4 Necessary installations

2.4.1 Using script

In this first version of the program it's needed to have some programs installed since the software is dependent on them.

Python 3 modules:

- matplotlib - https://matplotlib.org/index.html
- numpy - http://www.numpy.org/
- pyserial - https://pypi.python.org/pypi/pyserial
- pyfirmata - https://pypi.python.org/pypi/pyFirmata
- pyvisa - https://pyvisa.readthedocs.io/en/stable/getting.html

It is also necessary to install the National Insruments backend:

- https: //pyvisa.readthedocs.io/en/stable/getting_nivisa.html#getting-nivisa

2.4.2 Executable file

It is possible to run this program on other computer without further installation bu just running the executable file. However, this file has around 1 Gb of size in memory (static compilation, which includes all the necessary libraries in it).

3 Home page

At startup the software shows a Home windows with two boxes. One will change the window to the manual interface and the other to the automatic interface (Figure 1).

In this early version of the user guide and the control software it is only guarenteed functionality on the manual version, and even that one is prone to have bugs since it wasn't properly tested.

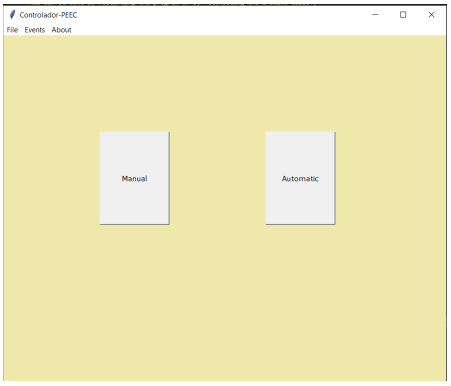


Figure 1: Home page, seen after the start of the program.

At the top of the window we can see a menubar with some menus (Figure 2). The file menu drops down two options (Figure 2). The first one - "Settings" - allows the user to change some simple definitons, while the second one - "pins" - allows the user to freely declare the pin numbers and the resistors associated with it.

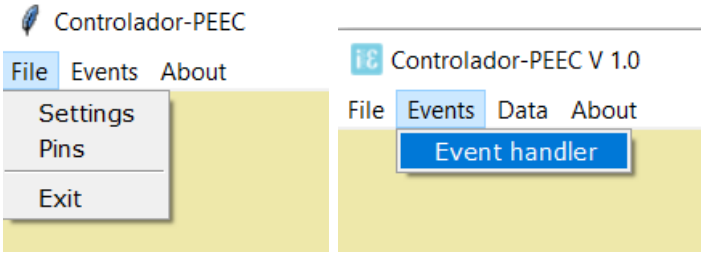


Figure 2: Menu bar options. Left: "File" dropdown menu; Right: "Events dropdown menu.

The second option - "Events" - drops down (Figure 2) one option - "Event handler" - which opens a interface to declare autonomous testing sequences, which will be discussed in a later stage (Section 5.1).

Now that the basic layout of the main page and the menus was presented, we can prepare our program for testing.

In the next two subsection we will further explain the required preparations for the usage of the program. It's important to note that all the displayed values are the default ones.

3.1 The "Settings" Menu

In this menu we can change some of the fundamental settings of our program (Figure 4).

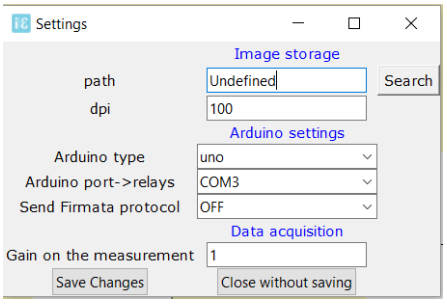


Figure 3: Settings menu.

In the first group (Image Storage) we can set the **path** in which all the results will be saved and the **dpi** of all the saved images (not yet supported) By hitting the "Search" button we can browse to the correct path on a windows explorer window.

The second group (Arduino Settings) allows the user to choose the arduino type from the supported ones (at the momento *Nano* and *Uno*) on the *Arduino type* dropdown menu. On the Arduino Port the user defines the **serial port** that holds the arduino connection. The last option is a toggle to turn on/off the dispatch of the Firmata protocol to the Arduino at the specified serial port.

The last group (Data Acquisition) is the value of the gain used in the data acquisition of the current. This is not yet used for data analysis but it is recorded on the associated file.

3.1.1 "Pins" interface

In this interface we can specify the Arduino pins used and the resistors associated with each pin. There are three options (Figure 4): One to add new pairs, one to edit the pairs and the other one to delete them.

To **Add** a value we must simply write on the text box "pin number: resistor value" and then click on the **Add** button, making sure that the pin number isn't being used and that there is no repeated resistor.

To **Edit** a value we must first select it, which will put its associated values on the text box. After changing the desired parameter it's only needed to click on the **Edit**

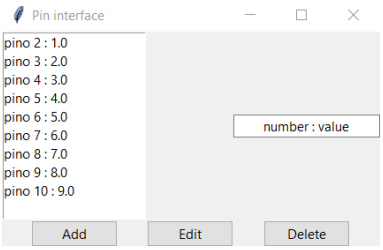


Figure 4: Pin interface menu.

button.

To **Delete** a pair we just have to select it and then click on the **Delete** button.

After closing this window the *Manual interface* (which will be covered in the next section) is updated with the new values.

4 Manual Interface

In this page it's possible to activate a single pin and colect the data from the oscilloscope. This data will also be displayed on the graph, but will only be saved as a text file, with information related to the acquisition.

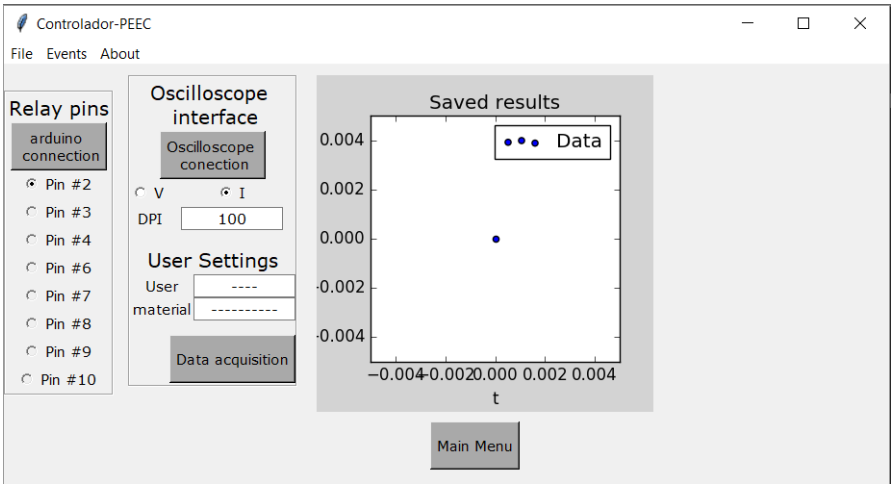


Figure 5: Manual page.

Before starting the process it's needed to initialize the **arduino connection**, on the left side of the window. The displayed pins are the ones previously inserted on the

"Pins interface". It's also needed to initialize the **oscilloscope connection**. These actions assume that the user has set up both instruments with the correct settings.

4.1 Data acquisition

To start this process we choose the desired pin on the list inside the **Relay pins** box. In the **Oscilloscope interface** box we choose the parameter under study ("I" for current and "V" for voltage) and the DPI for the graphs saved (not yet supported).

Under the **User Settings** it's specified the name of the **user** and the material under study.

After choosing all the desired settings the last stage is hitting the **Data acquisition** button. The data will be displayed on the graph, but it isn't saved as an image.

5 Automatic Interface

In this window we can select the events that will be processed, make the connections to the instruments and start the data acquisition.

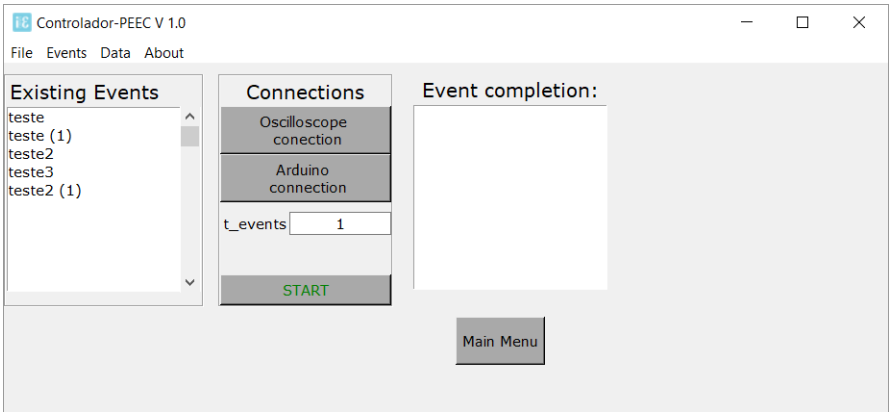


Figure 6: Automatic page.

In the left side we can select the events that will run when the "START" button is pressed and, by double clicking them, the edit event interface will open. On the DPI entry we can specify the DPI of the saved images. On the "t_events" we specify the time interval between two consecutive events which, by default, is one second.

On the far rigth we see a white box, with the label *Event completion*. In here, during run time we will see the event name and the number of pins already done out of all the pins inside the event (Figure 7.

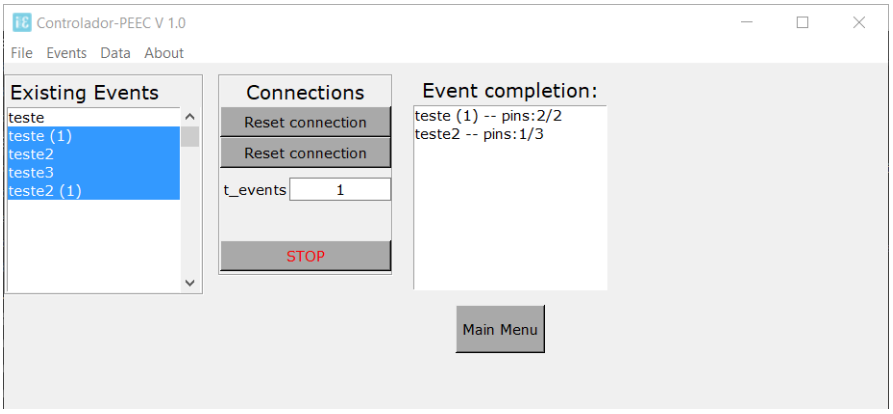
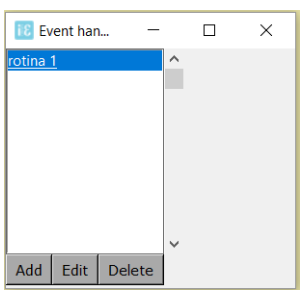


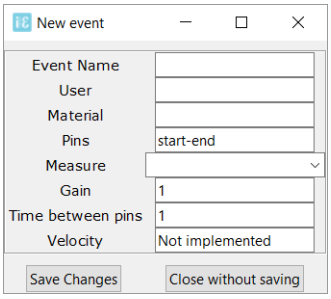
Figure 7: Automatic page during the execution of a group of events.

5.1 Events creation

After clicking on the "Event handler" button (Fig 1) the event window will appear (Figure 8-a). In here the user will have three different choices: *add*, *edit* or *delete*.



(a) Window for event handling.



(b) Window to create and edit events.

Figure 8: Event interface menu.

The *add* button will open a window (Fig 8-b) in which we can customize the event, with the material under study, the researcher's name, the pins that will activate, the type of measure (this parameter should be **I** or **V**), the gain used on the measure, the interval of time that we have between two consecutive pin activations and lastly

the speed of the motor. The pins that will be activated during the event can be defined in two different ways:

- start-end: all the pins between "start" and "end", including the start and end pin, are used. To repeat the pin activation we have to use the syntax *start-end[number of activations]*. For example, if we want to use the pins 2 to 4 with 2 activations each we should input 2-4[2].
- pin.pin: We manually define each pin that will activate, by separating them with dots. Like before we can user the "[]" identifier to activate multiple times the desired pins.

The *edit* button will open a window, equal to the *add* window but, with every parameter showing the selected event information.

The *delete* button will, as the name shows, delete the chosen event.

During the creation of the event, the user should abstain from using "—" or ":" in the descriptions, wince those characters are used to parse the information. A built in check was built for every entry.

Lastly we should note that double clicking on a event the *edit* interface for that specific event will open and, if the new event name is equal to any of the others then it's name will be changed to the format "event name(number of last match)".

5.2 Data acquisition

After choosing the desired events and making the necessary connections we can then click on the "Start" button to start the acquisition. In the first place it's necessary to say that if we have more than 1 event choosen first we will parse them. If two events have the same user, material, speeds and different measures (one from current and the other from voltage) the outputs will be placed on the same folder. Likewise, if we have different speeds but same material and user it will also be placed with them. This process will only occur if **all** the events are selected at the same time.

6 Motor control

This section was not implemented and was left for future work. However, the the data analysis already supports different motor speeds. To properly implement this feature one imply has to change the *event* class [1] (on the *event_handler.py* to accept

[1] On the file we can find "TODO" comments with the necessary changes and recommendations

a range of velocities (at the moment we can only choose a singular value) and afterwards create the control program.

7 Data treatment

The Data treatment is done on a separate window, which contains a dynamical graph, which is updated after the analysis. Before we can start the process we have to create the threshold value for each file, either by doing them one by one **or** by selecting them all at the same time.

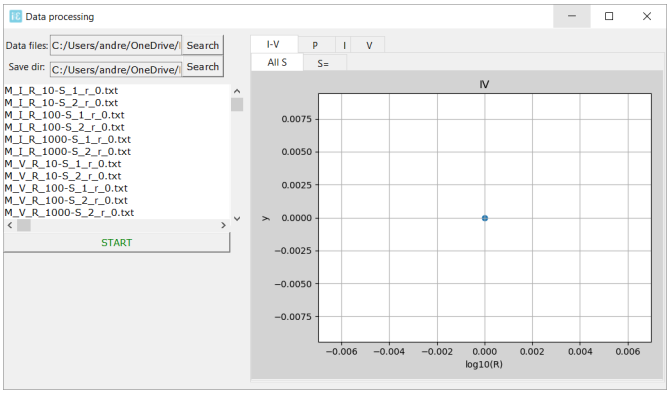


Figure 9: Data treatment window.

1. On top of the page we see a search button for the directory in which the files are saved. After choosing it we will see a list of all the files inside the specified directory. Immediatly bellow this box we find the search button for the directory in which the results will be saved
2. On the graph we can see four different tabs and, by clicking on them, we will only see the specified part of the analysis. On each tab we have a collection of other tabs, in which we can see the data related to the specified motor velocity.
3. The "LOAD" button opens a new interface, in which we will choose the threshold that will be utilized during the analysis. After doing this setup the button will be changed for other;
4. The "START" button will appear after the "load" one and start the automatic analysis. If we select (with the mouse) files then only the highlighted files will be used. Otherwise, if we click on none, then the analysis will run through all of them.

It's important to note that this setup will only work with files created from within this program, since on the first lines of each file we can find acquisition related data. It can also parse trough files using different motor speeds and group the relevant data together.

7.1 Choice of threshold

In this window (Figure 10 we can see a text box in which we will input the threshold value (if it's desired a negative peak then you should input a negativa threshold and,later on we will use the absolute value of that peak). If we desire to run through all the file manually we must insert the value, click on the "Set threshold" and afterwards on the "Next file". When we arrive at the last file we will see a box warning us of that fact.

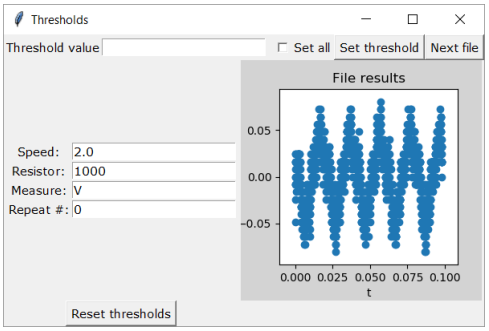


Figure 10: Threshold definition window.

On the other hand, if we check the "Set all" box then all the files will be associated with the value at the time and lock the option to change files. Now we should take note of some aspects:

- While setting the threshold one by one:
  - Until we change the file we can freely change the value by clicking again the "Set threshold" button.
  - At any point we can input a number and set it as the global threshold.
- If we used a global threshold and desire to change it to a file-by-file we must hit the "Reset thresholds" button.

We should also take note that we have a graph in which we can visualize the data and some text boxes in which we can see the relevant information about the file at

hand. In this system we do not have a check on the material between files so, one should be carefull in the choice of used files.

7.2 Automatic analysis

After creating the thresholds for all the files in use we can now proceed to the data analysis part. We implemented a simple threshold based system that calculates the average of all the points above the passed threshold value. If we used the "repeat pin" functionality those points will also be considered.

In order to filter out the other points that make up one peak we say that around a maximum, in a window of 6 points for each side we can't have peaks (If we have a greater peak in this region then the peak value is changed for that value).

We should also note that if we have missing values they will be taken as "NaN" (not a number) and thus not appear on the graphs. On the graphs we can also see them individually (if we have different speeds on the files) and also save them individually.

7.3 Outputs

On the selected folder for saving outputs we will see the appearance of a new folder, called "data", with six files:

1. I.txt;
2. V.txt;
3. P.txt;
4. I.png;
5. V.png;
6. P.png

The text files will store the data points for the current, tension and power graphs. In the first two cases, we will save the average value as well as all the values taken into account for the calculation of the average value.

This way we can have a better controll of the system, with the ability to manually check if we have big deviations between values which shifts the true value of the peak.

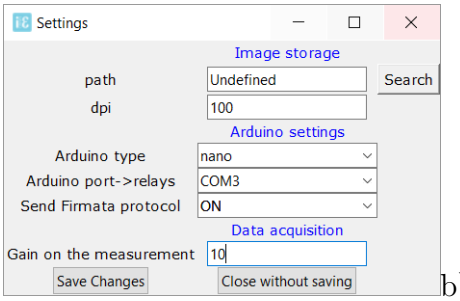
The *png* files are the previously seen (on the data treatment window) graphs, saved with the desired dpi.

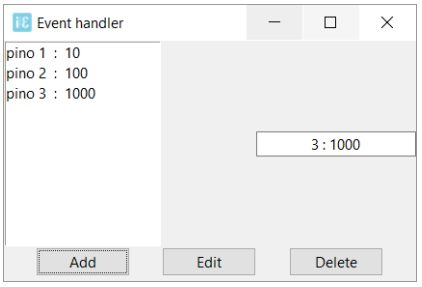
8 Quick tutorial for setup and use

For this tutorial we will assume that a user, called **User** wants to test a material **Tribo**, using a gain of 10 on the current measurement, one **arduino Nano** connected to **COM3**, with the pins **1 to 3** connected to the resistors of 10,100 and 1000 Ω,respectively.

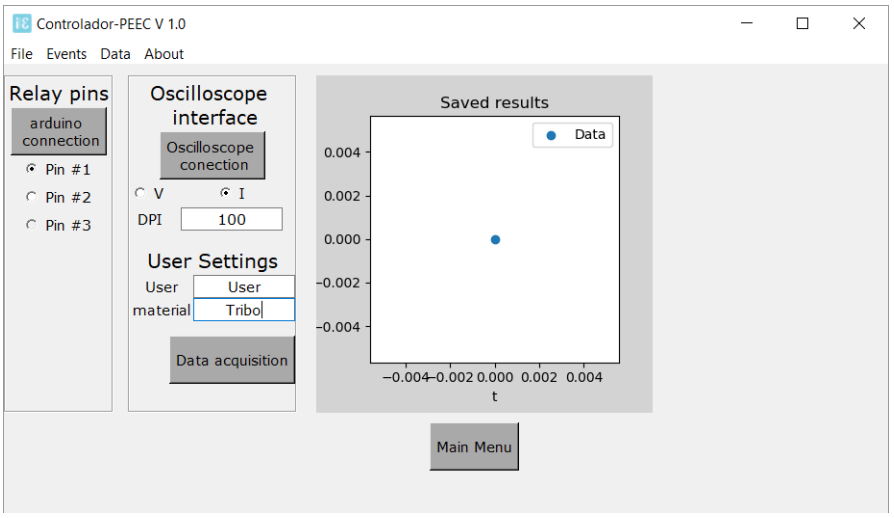
8.1 Manual process

1. Input the correct settings on the *settings menu* (Figure -a). Note: if the path option is undefined then all the relevant data will be saved on the code's directory. If it's the first time using the arduino we should also set the *Send Firmata protocol* to **ON** and, after this first analysis, change that option to **OFF**. Afterwards hit the "Save changes" button and open the pins interface (Figure - b) , inputing the desired pins.

a) 

b) 

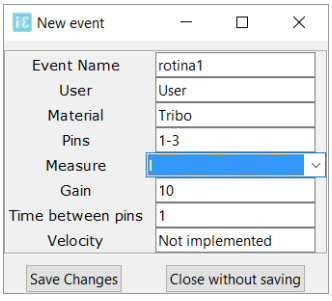
2. Afterwards, on the main menu, we can fill in the *User* and *Material fields*, press the connection buttons for both the oscilloscope and the arduino and specify the measure that we will undertake as well as the pin that we wish to activate. If we are measuring voltage the gain that will considered is of 1, since we use no amplifier on this study. If all the devices are properly connected and all the steps were followed we should see no error message. In this stage is important to take attention to the erro messages and the command line that opens with the program. If one of the connections is not completed with success the button will not revert to it's old state and it's necessary to click on the "Reset connection" and try again after solving the problem.



3. The last step is to press the *Data acquisiton* button.

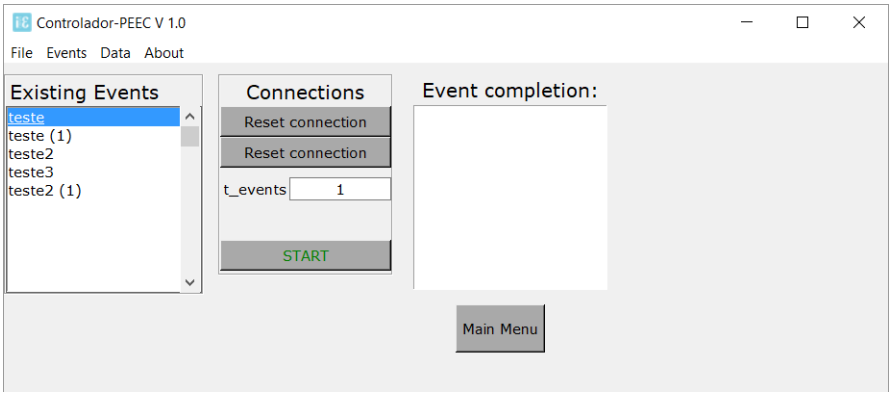
8.2 Automatic process

1. The first two steps are equal to the ones from the *Manual process* from the previous subsection.
2. We can then navigate to the new event window on the *Event Handler* interface and create an event specific for our needs and press the "Save changes" button when finished.



3. It's important to note that in the event's pins we can define them in two different ways: start-end, in which all the pins between those two values are used , and pin.pin, in which the pins that will be used are separareted by dots(".").

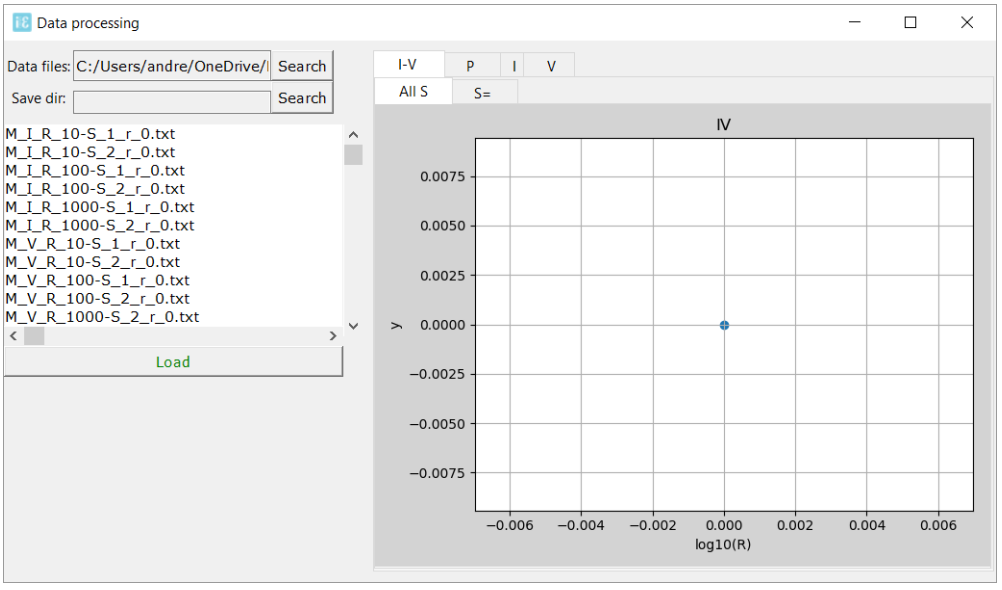
4. On the automatic page we have to, in the first place, to start both connections, by pressing the respective buttons. Afterwards, we select the desired event and press the "START" button. In this case, since we only have one event the *t_events* parameter is not relevant.



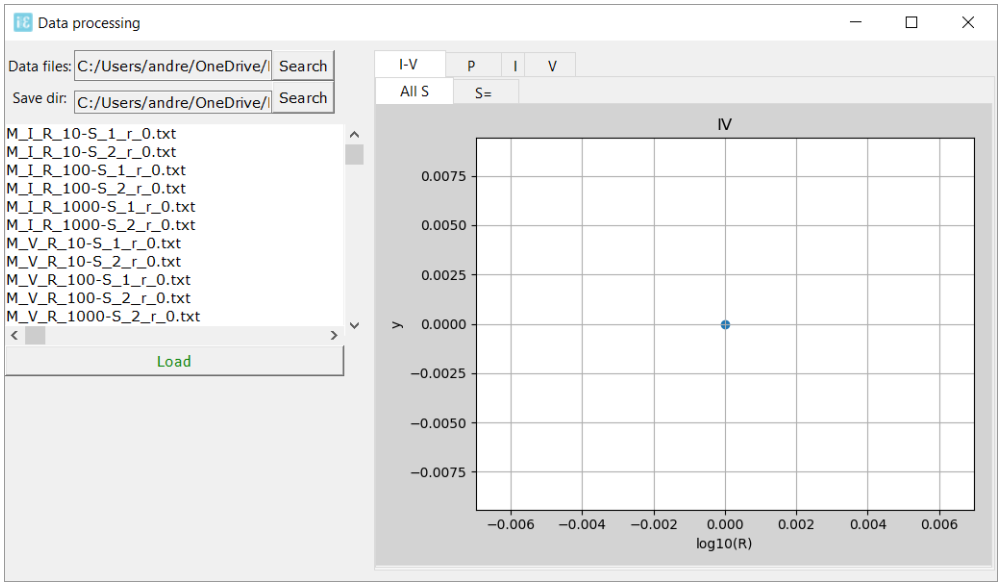
8.3 Data analysis

After the data acquisition we can now analyse the data to study the material under study.

1. We start by hitting the "Search" button to find the directory in which our files are stored.

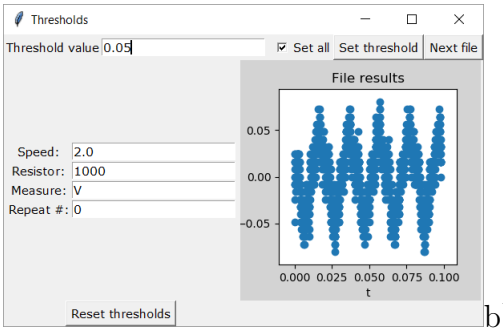


2. Afterwards we choose the path in which we want to save our data and press the **START** button.

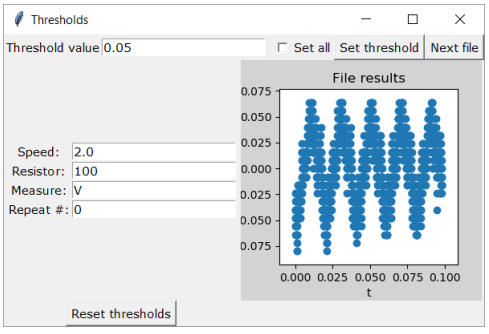


3. We can then click on the **Load** button to open the threshold input window [2]. If we wish to put the same threshold for all the values we input the number, check the "Set all" Checkbox and hit the "Set thresholds" (Figure - a). Otherwise, we have to input each value one by one (Figure - b). We should remember that the input value is the actual value that we want the average to stop counting and not a percentage of the maximum value.

a)

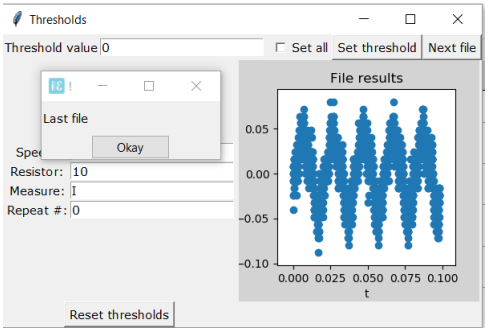


b)

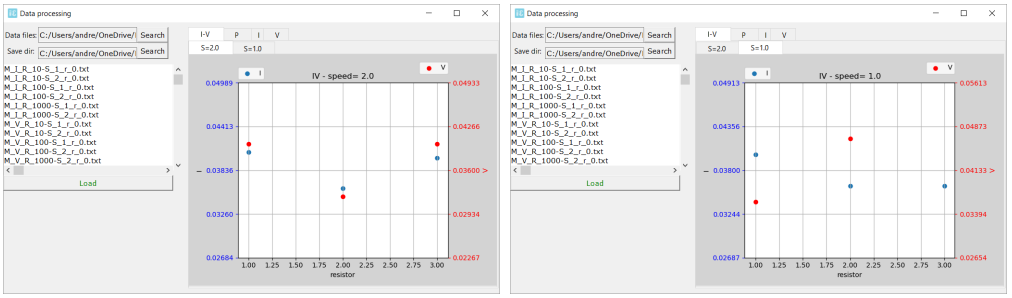


4. When we arrive at the last item we get a message and can close the window;

[2] All the data on this report was simulated or noise.

a) 

5. Afterwards we can hit the "Start" button and the graphs are filled with the data.



9 Comments on design choices

On the later stages of this project some details regarding the used libraries and some coding practices became aparent:

1. The tkinter library is not thread-safe, i.e., we can't launch a new window from another thread. This particularity lead to the usage of virtual events to open the warning messages from the main thread;
2. We used text files to store information and later on, when the program booted up, load it. This could be avoided if we used a library that can convert python objects to a byte stream, see *pickle*, *jsonpickle*, among other. This method would allow to store the information on a class and directly it save it on disk and, when needed, load it;
3. In aesthetic terms the Tkinter is not the best available, giving the program an "older look";
4. I could have modularized the program in a far superior way, allowing for easier future extensions.

5. The usage of tkinter's built in directory browser was also not the best choice since on windows 10 every time we do it we get a warning message related to an untested functionality. This warning is in no way prejudicial to our software but it fills the command line with "junk"

10 Settings reset

It is possible to reset the settings to the factory default by deleting the hidden file *settings-peec.txt* on the directory of the software. On the next startup of the program it will be recriated with default values for all the variables. This action will not delete all the current saved events, since they are saved in another file.