User Manual v5.0

OTBiolab

Index

Introduction

OTBiolab is a software developed by OT Bioelettronica, completely free and downloadable at:

https://otbioelettronica.it/en/software/

It allows to acquire signals from OT Bioelettronica devices, review and process them. It has been designed for .NET Core and Windows 10 operating system with 64-bit processor resolution and use WPF technology to design the graphical interface.

For a best experience we recommend a 1920x1080 resolution monitor.

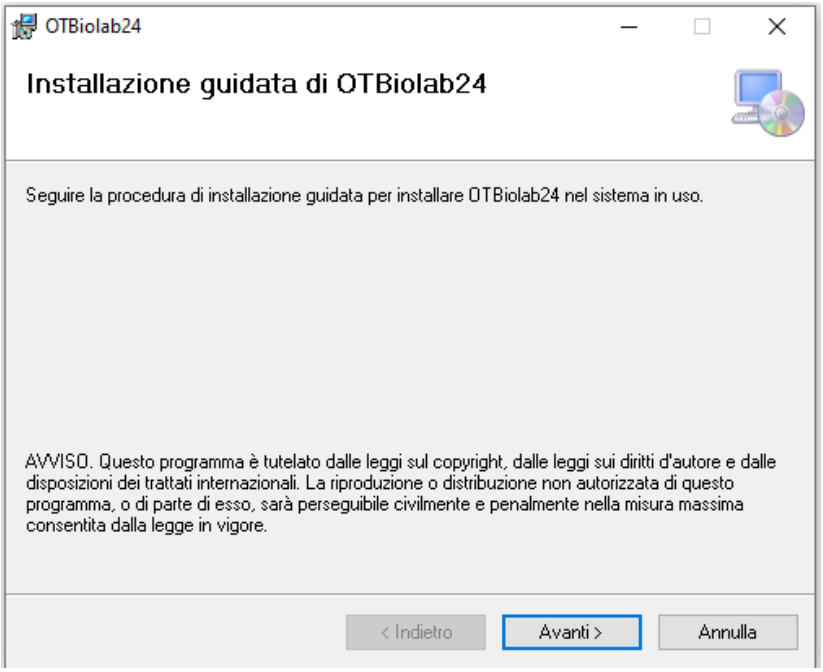
PC Requirements

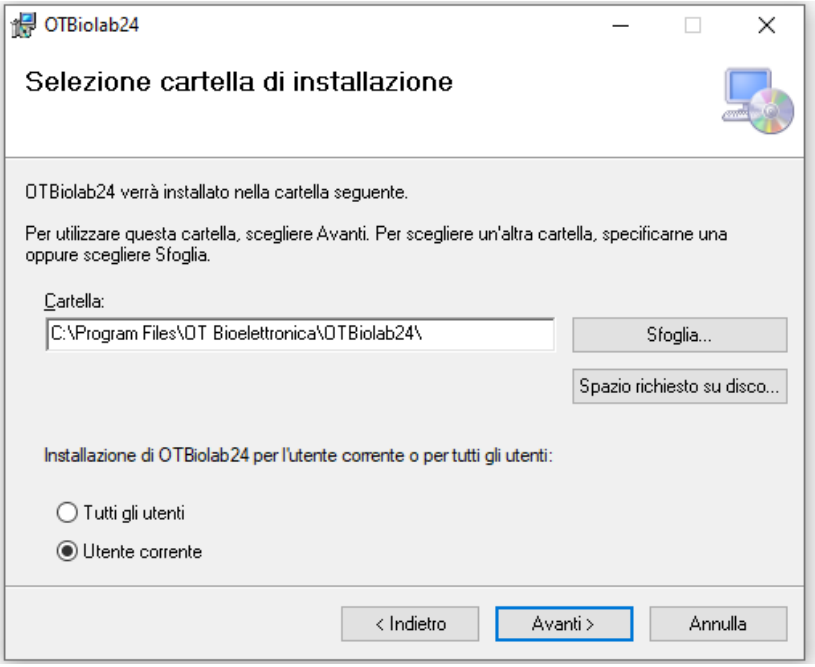
Operative system: Windows 10
Processor: Intel i3 5th generation.
Port: USB2.0 and Ethernet
Hard Disk: 1 GByte for software installation, additional space is required in order to store data and interact with the database, or the python features
Minimum RAM: 4 GBytes

The minimal PC requirements listed allows the proper acquisition of all 256 channels of Quattrocento equipment (with max sampling frequency 10kHz). The increase of the number of visualized channels can produce a visualization latency or an intermittent visualization.

OTBiolab installation

OTBiolab installation procedure requires few minutes and allows to install the software. Download the installer and run it. The start-up window will appear, showing the software is under copyright law and then asking the user to select the installation folder.

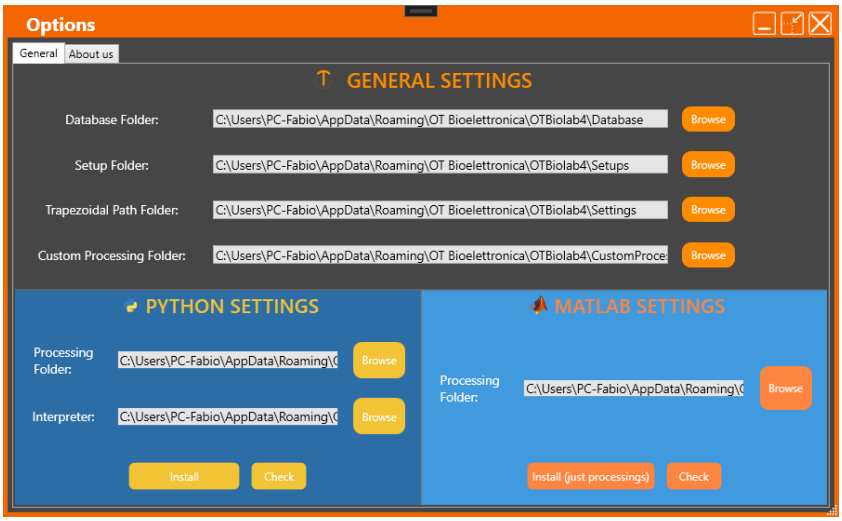




To confirm the installation the user must allow the software to apply changes to the computer, otherwise the procedure is reverted and OTBiolab will not be installed.

Additional Component

The new OTBiolab4 provides the possibility to create, run and import custom codes written in Python or MATLAB. In the default installation the necessary components are not automatically installed and require a specific procedure to start using it. The following page is the view dedicated to these procedures.

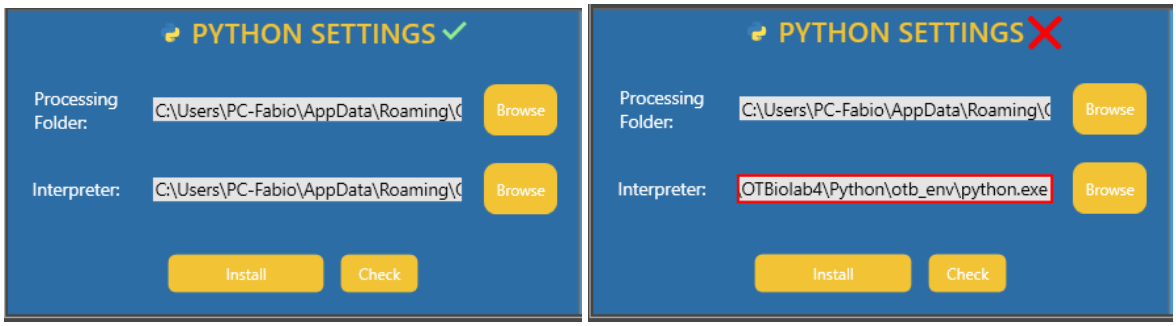


Let's focus on python and MATLAB installation procedures.

Python

In the python settings the user will be able to start and check the necessary external components to run custom codes. To run python in a computer an interpreter must be locally installed and must be pointed through the “Interpreter” path. If you already have installed python in your computer, click the “Browse” button and search for your “python.exe” file. If you don’t have python, the installation procedure, that can be executed through a bat file contained in the software hidden files executed with “Install” button, will download all the python components, included “python.exe”, and a first list of default python processing created by OT Bioelettronica. The default path of interpreter and of processing folder are reported in the dedicated textbox and can be changed through “browse” button.

With the “Check” button the software will verify if the interpreter path exists and if the processing folder is valid. If the check is overcome a green tick will appear near “PYTHON SETTINGS” title, otherwise a red cross will be showed, and the involved path problem will be highlighted through a red box. Below an explicative image of both situations:



So for a proper installation, click the “Install” button and verify the components with the “Check” button. If something goes wrong a dedicated message will be showed which advise the user to run the installation with administrator permission.

ATTENTION: this installation procedure is totally automatic, for this reason a huge amount of components are installed, pay attention because it will require 1.8 GByte of free space. The installed environment can also be used and updated with “Miniconda3” software if installed in the computer.

Matlab

In the MATLAB settings subsection, the user can find a dedicated textbox for the default MATLAB processing, an install button and a check button. The textbox contains the path to the folder which should contain the default MATLAB processing. For MATLAB, the installation procedure is simpler compared to python because just the default processing developed by OT Bioelettronica are downloaded in a dedicated folder. The “Check” button just verify that the default MATLAB processing folder exists and is a valid folder. The result of the check will be showed in the same way of python (green tick or red cross).

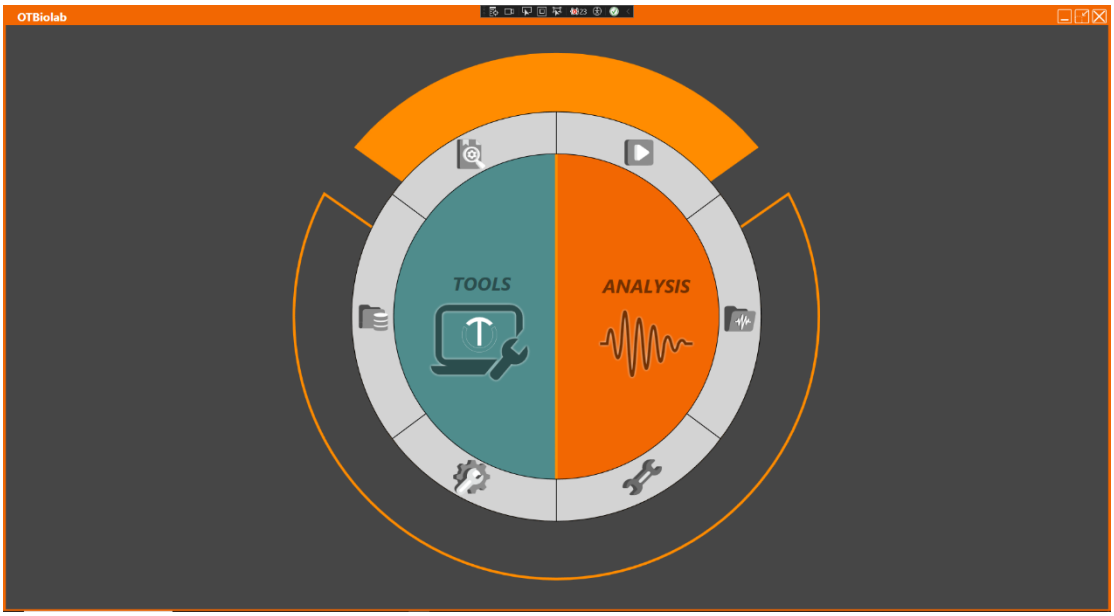
ATTENTION: the installation procedure is really simple and just processings are installed, this mean that if the user hasn’t MATLAB already installed in the computer, the software will not be able to run MATLAB processing. The check doesn’t verify the installation of MATLAB, so be careful to this aspect.

Main Interfaces

This chapter described the main windows the user will encounter during a classic utilization of OTBiolab.

Main Menu

This is the main menu. It's divided in two sub-sections: **tools** and **analysis**.



Tools contains three different buttons:

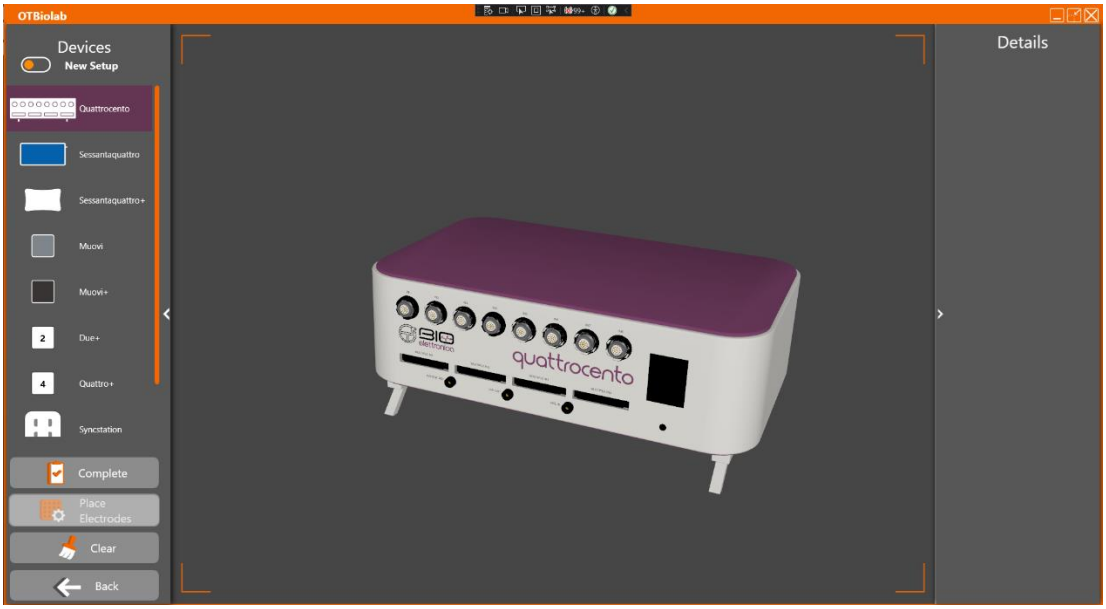
- **Database Management**: it takes the user to the database window where he can interact with the database to organize his work. He can create new subjects, new studies and place the data in a tree structure in order to easily access file and handle large amount of data and information.
- **Community tools**: this window contains many tools created by the EMG community that are free to use. Most of these tools can be used with Matlab or Python.
- **Option/Support window**: contains OTBiolab configuration parameters such as specific path for features.

Analysis contains three different buttons:

- **Realtime**: this window is specific to the realtime acquisition, where a user can select a setup, visualize signals from the device and start a recording.
- **Offline**: through this window it can be possible to open a recorded file, visualize the signal, run processing and do other kind of operation.
- **Setup**: this is where the user can create a device setup. Each setup specifies the device configuration during the realtime acquisition (number of channels, sample frequency and other hardware parameters)

Setup

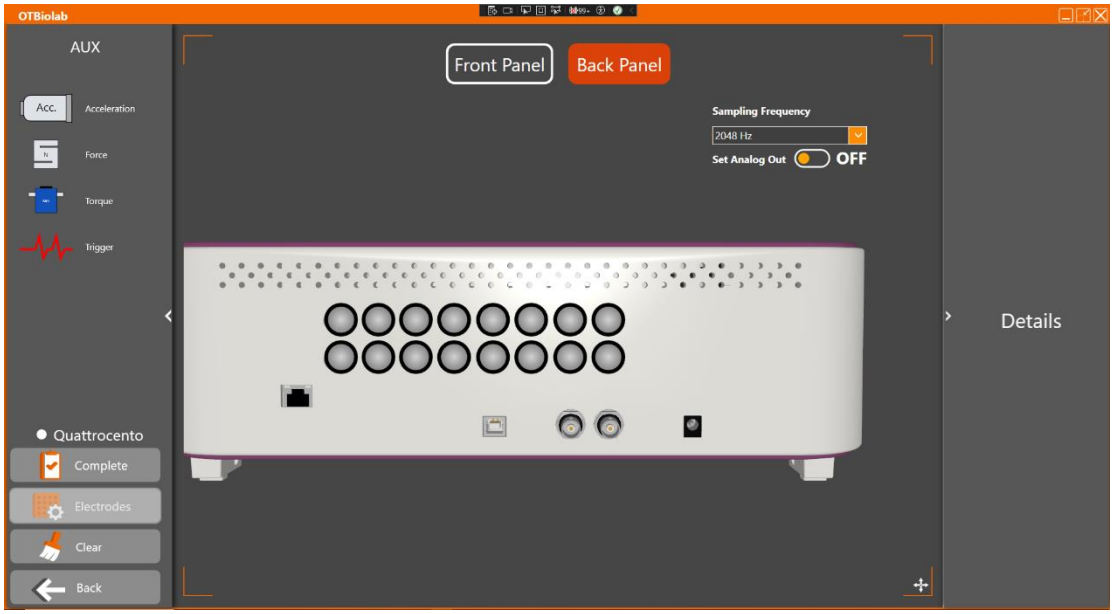
The setup window contains a 3D visualization of each device. From the left side menu, it's possible to select the device for which we want to create a setup and see its 3D model. It can be possible also to move the model in a free visualization mode and zoom in and out for details.



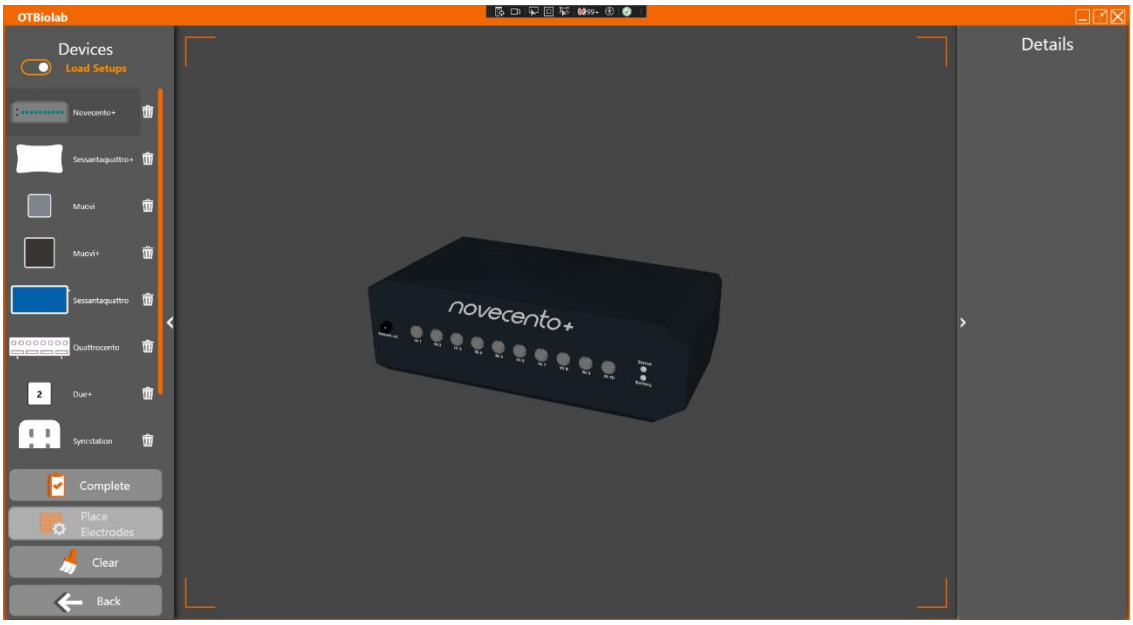
When the device is selected the device aside list switch to the adapter list of available adapters for that specific device. In the following picture a list of Quattrocento adapters is displayed.

Using mouse drag and drop the user can place the adapter in the corresponding slot and then fill it with the sensor in the right-side menu. In case the device has a back panel, the user can access it with the dedicated button and move between the two figures. The 3D visualization automatically rotates to visualize it.

The back panel button visualizes the AUX connectors and switch the left side adapter list with the AUX sensor list. The user can configure an AUX connector in the same way on another slot: using mouse drag and drop.



OTBiolab allows to create and save multiple setups. To see and pick one of them the user can use the toggle button and switch to *"Load Setups"*. In this way the left list of device switch to the list of saved setups.



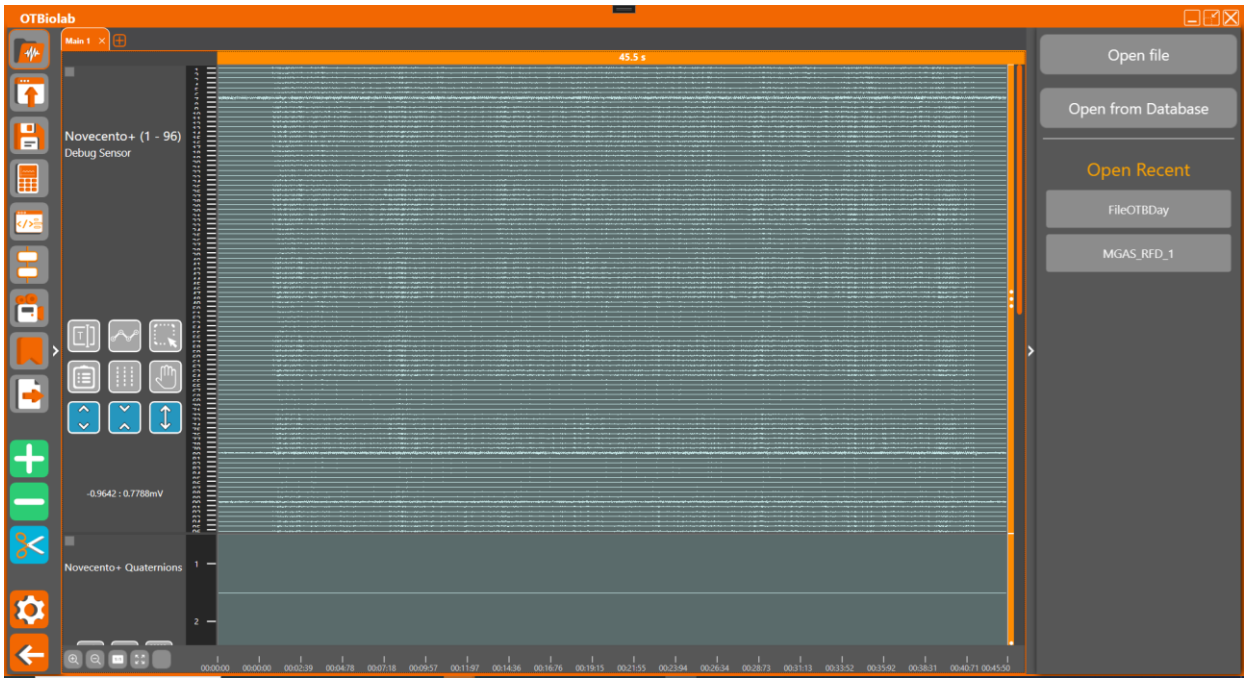
A saved setup can be selected to see the current configuration.

The setup window provides a set of buttons to access specific feature:

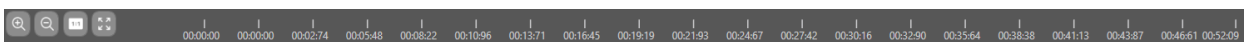
- **Complete**: confirm the setup completion and add to the list of saved setups
- **Place Electrodes**: open a new window to specify muscles where the electrodes are placed (currently unavailable)
- **Clear**: remove all adapters and configuration from the visualized setup
- **Go Back**: use this button to go back to Main Menu.

Offline

The offline window is used to open file, visualize signals, run processing and much more. It's composed by two side vertical menus and a central part dedicated to the soundtrack, a graphical container where the tracks are displayed. On the left side a vertical menu contains different buttons to access additional features and open sub-menus. On the right size the space is dedicated to those sub-menus. Based on the selected buttons on the left a different sub-menu is opened on the right, containing many other buttons or controls depending on the selected functionality.



To navigate the data in time it's possible to use a set for dedicated buttons placed in the bottom left part of the soundtrack.



These buttons are used in this order to: increase the horizontal scale (zoom in), decrease the horizontal scale (zoom out), zoom to a specific selection of time and fit the entire track length in the window. Right to this section there is an horizzontal time bar that display the current visualized range of time a include a scrollbar to move along it. A set of white vertical lines are used as time separator for the x-axis.

The left side menu of the Offline window is composed by a set of buttons that enable features. The buttons are:

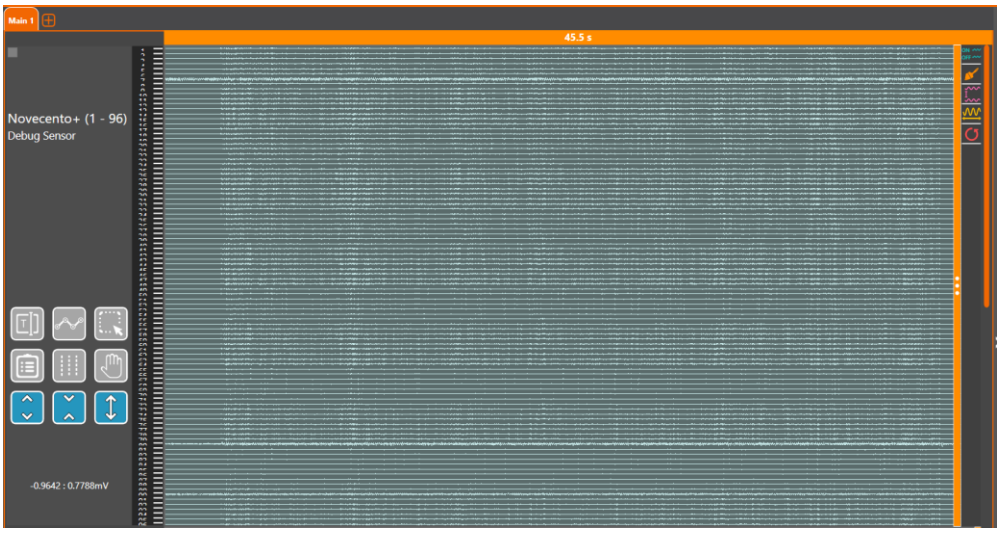
- **Open File**:
  Use this button to open the right-side menu where you can select to open a new file, open a file from the database or open a recent file.
- **Import File**:
  Use this button to open a dialog window and select a file to import. Differently form the open file, the import, will keep visualized current file without overwriting the soundtrack content.
- **Save**:
  Use to save the file on disk.
- **Processing**:
  Use this button to open a right-side menu containing the OT Bioelettronica processing suite. Check future chapter for the entire list of processing and its description.
- **Python Processing**:
  Use this button to open a right-side menu containing the OT Bioelettronica python processing suite. This is where you can also import a custom processing or implement a new one.
- **Graph Editor**:
  This button opens the Graph Editor, a GUI specialized in python processing pipeline.
- **Video**:
  Open the Video window.
- **Marker**:
  Open the Marker window.
- **Export**:
  This button open a right side menu where you can select the export format (Matlab .mat, Csv .csv, Python .npy)
- **Group Tracks**:
  This button is used to group together different track into a Grouped track.
- **Ungroup Tracks**:
  This button is used to separate a grouped track into single tracks.
- **Edit Tracks**:
  This is a toggle button that can be turned on or off. If this function is activated, a scissor image will substitute the mouse arrow, and the user will be able to remove track pieces from the visualization.
- **Settings**:
  Open the setting right-side menu.
- **Go Back**:
  Use this button to go back to the main menu.

Track

A **track** is the graphical representation of an acquired channel (or more than one). It shows information about the name of the channels, the acquisition configuration such as sample frequency and conversion factor, and it loads the data from file to display them. A track that contains multiple channels such as all channels of a matrix is called **grouped track**.

A set of buttons is associated to each track and are used to:

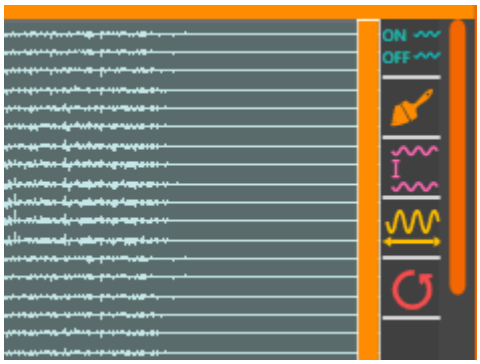
- **Set title**: change the track title.
- **Show points**: data are visualized as interpolation of point. Enable this to see values as red dots.
- **Manage selections**: open a little menu to handle the track selections.
- **Display information**: this buttons open a sub-windows where all the information related to the track are displayed. Such as hardware configuration used during the acquisition, source track conversion factor, etc.
- **Show vertical lines**: display a set of vertical lines.
- **Open the free hand modes**: open the track in a new window where it's possible to zoom in and out the values and move along the track using mouse wheel and other shortcut.
- **Vertical Zoom In:** increase track vertical scale.
- **Vertical Zoom Out:** decrease track vertical scale.
- **Autoscale:** Load max/min value of the visualized data and set them as vertical scale.



In the track right part there is an additional vertical menu with another set of features, more complex than the previous and that allows a deeper kind of personalization.

The buttons are:

- **Choose channel to plot:** turn on and off channels in the grouped track. This feature make sense if the grouped track is the representation of a matrix and the user wants to remove specific channels from the visualization.
- **Choose color:** open a window that allows the user to change color of track data and background.
- **Set vertical range:** specify the min/max values for the vertical scale. It's also possible to remove the vertical offset in case of a grouped and visualize all the track on the same level.
- **Shift:** using a slider it's possible to shift to the right or left the track adding zeroes sample to the data. The file containing the original data is not modified.
- **Reset**: reset changes to default and display the original track.



Realtime

The realtime window structure is very similar to the offline ones but it contains a slightly different set of buttons and features.

This window is meant to be used selecting a setup of the device we want to acquire with and then starting the acquisition. The start button sends a start command to the device and then the software begins to receive data packets and distribute them to the tracks. The data are drowned with a specific time range and overwrite the old ones on the window.

The realtime buttons in the realtime menu are:

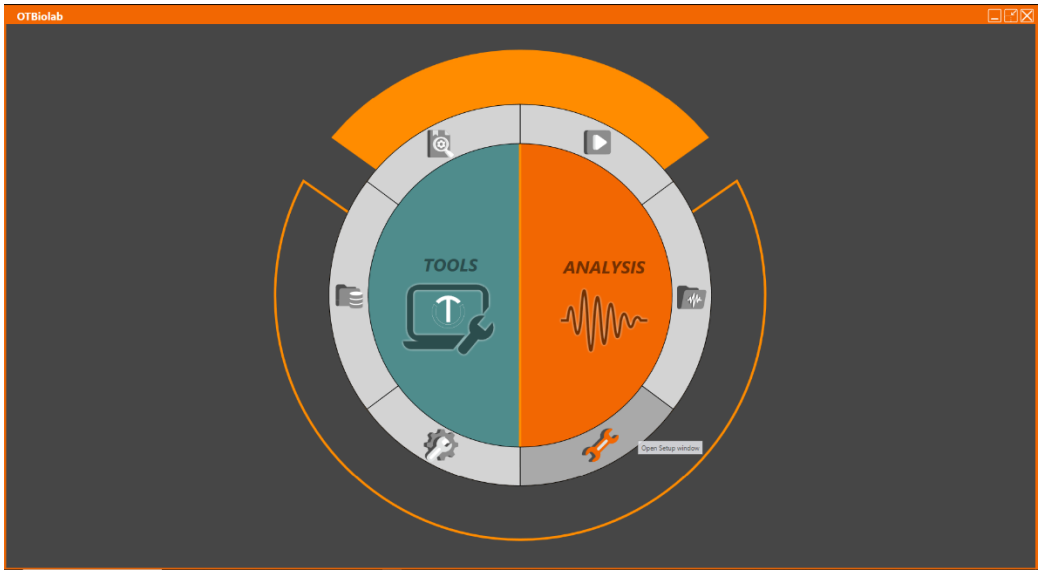
- **Processing**:
  Use this button to open a right-side menu containing the OT Bioelettronica processing suite that can be used during a realtime acquisition.
- **Video**:
  Open the video window.
- **Marker**:
  Open the marker editor.
- **Activation Map**:
  Open the activation maps processing window.
- **Trapezoidal Feedback**:
  Open the trapezoidal feedback window.
- **Setup**:
  Open a right-side menu where to select the setup to use for a realtime acquisition.
- **3D Inertial Visualization**:
  Open a window specific to the 3D visualization of devices with quaternions in order to see the direction and orientation of the device.
- **Start**:
  Use this button to start and stop the acquisition. A setup must be selected in order to do that.
- **Pause**:
  Use this button to pause the acquisition and freeze the visualization.
- **Record**:
  Use this button start and stop recording data on a file.
- **Settings**:
  Open the setting right side menu.
- **Go Back**:
  Use this button to go back to the main menu.

Use cases

This chapter describes a series of use cases that contains the main features of the software. Those cases are examples of main functionalities and what can be done using OTBiolab.

Create a new setup

This step is necessary in order to acquire signals from an OT Bioelettronica device. Creating a setup means specify the configuration of the device before starting an acquisition. The setup window can be opened from the right-bottom button of the main menu.



Then the setup window opens and from the left side is possible to select the device. Each device has different configuration, please refer to the device user manual for additional details. As example the Quattrocento will be used for this use cases.

Using the mouse it's possible to drag and drop an adapter into a connector slot. If the adapter can be placed in that specific connector the slot turns green, otherwise the red colour appears and it's not possible to insert it there.

By clicking on the connected adapter, a right-side menu opens, allowing to insert a sensor. This is done as previously using the mouse drag and drop.

The user can proceed this way and configure all the connectors he needs and when it's done, click on **Setup Complete** button to save the setup.

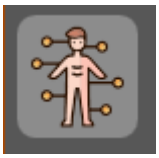
Run an acquisition

One of the software main goals is to acquire, save and visualize data from OT Bioelettronica devices. Supposing that the user already has created a setup (refer to previous use case) it's now time to start the acquisition.

From the main menu click on the top-right button to enter the realtime window.

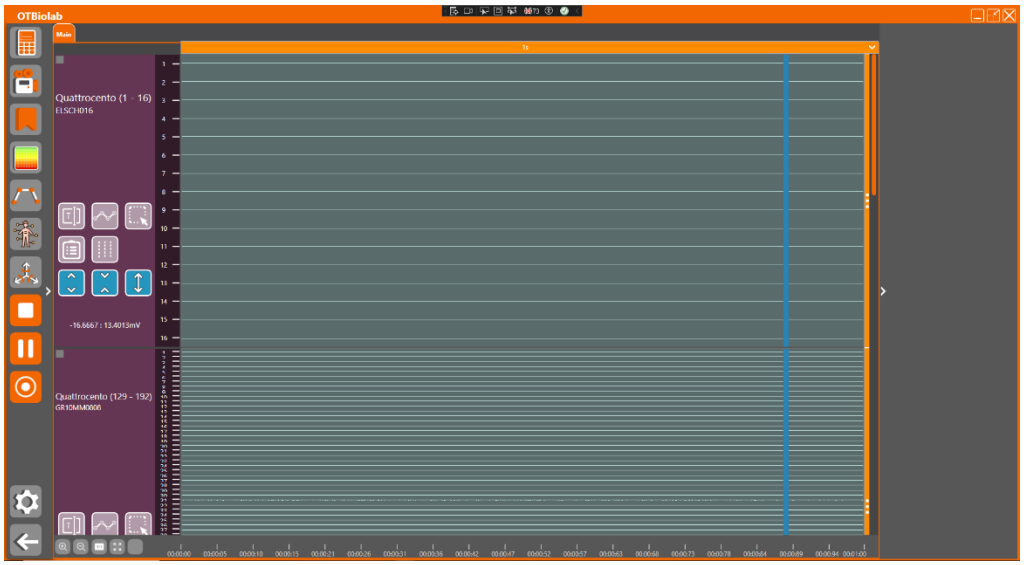
As described in the related chapter the window contains a vertical menu with many options and features. What we need to start an acquisition is to select the setup we want to use.

Of course, we consider the device to be turned on and connected to the computer. To select a setup, click on the following button to open a right-side menu.

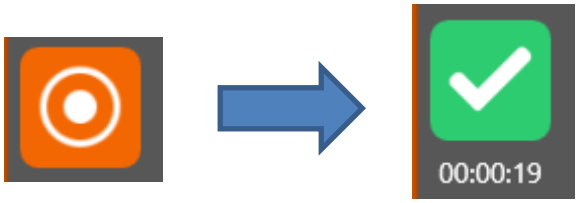


The entire list of saved setups is shown, and the user can pick one of them. A recap of the configuration is displayed to summarize the setup information and details.

Then, click on the orange **Start Button** from the left side menu to run the acquisition. Based on the setup the tracks are created and displayed in the centre part of the window. For each connector a specific grouped track or single track is created, this difference depends on the number of channels contained in the connected sensor. It could be possible that additional control channels of the device are shown too.



The start button is substituted by the **Stop Button** and it's now possible to record data on a file while the acquisition run. Click on the record button to start the recording. The icon turns green, starts blinking and a timer is displayed.

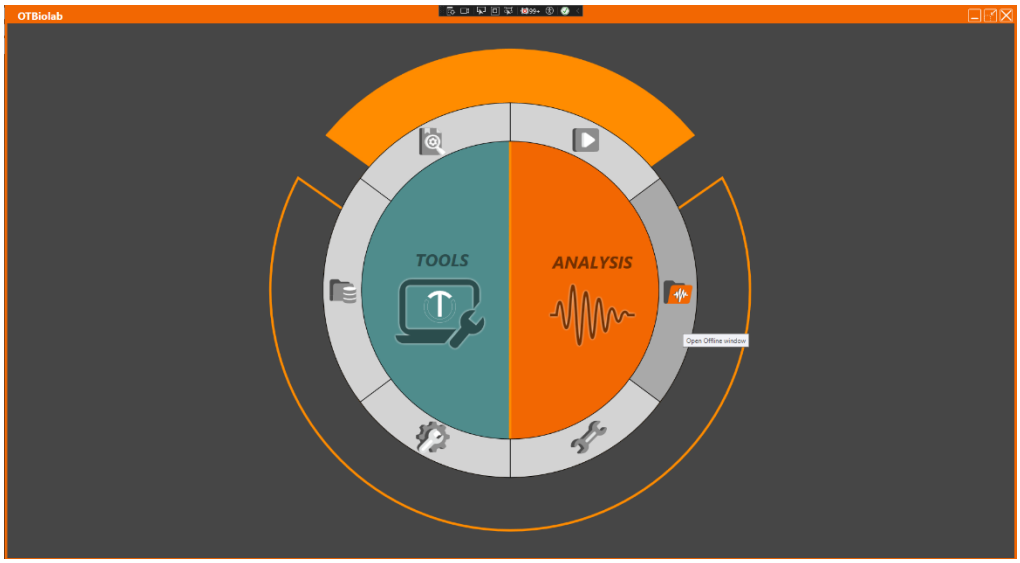


When the recording is complete, click on the green tick icon to save a file.

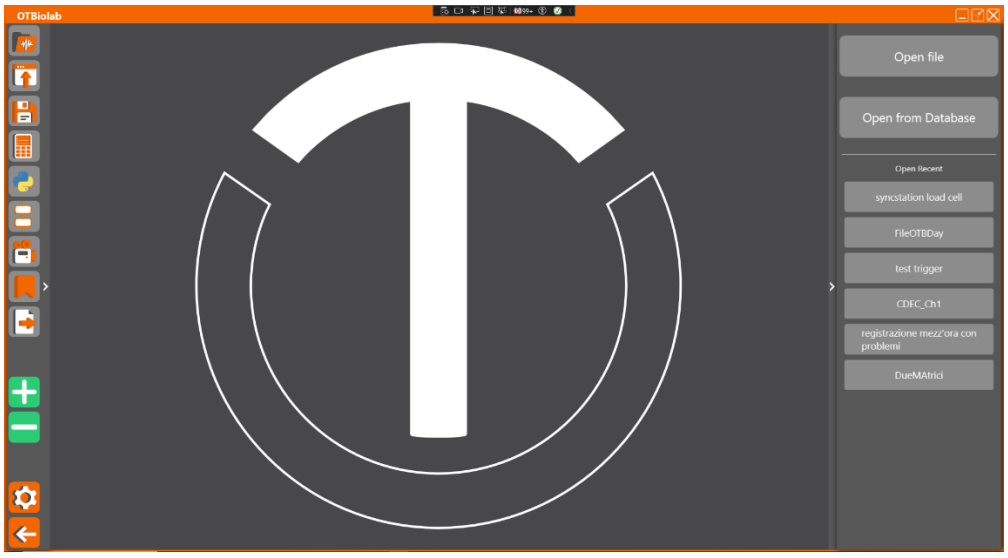
Review a file

Using OTBiolab is possible to open a .otb4/.otb+ file, visualize the recorded data and use a large set of tools to analyse, extract information or export them. Each use is specific to the user's goal and this use cases cannot cover all of the different feature but to provide an example of file opening.

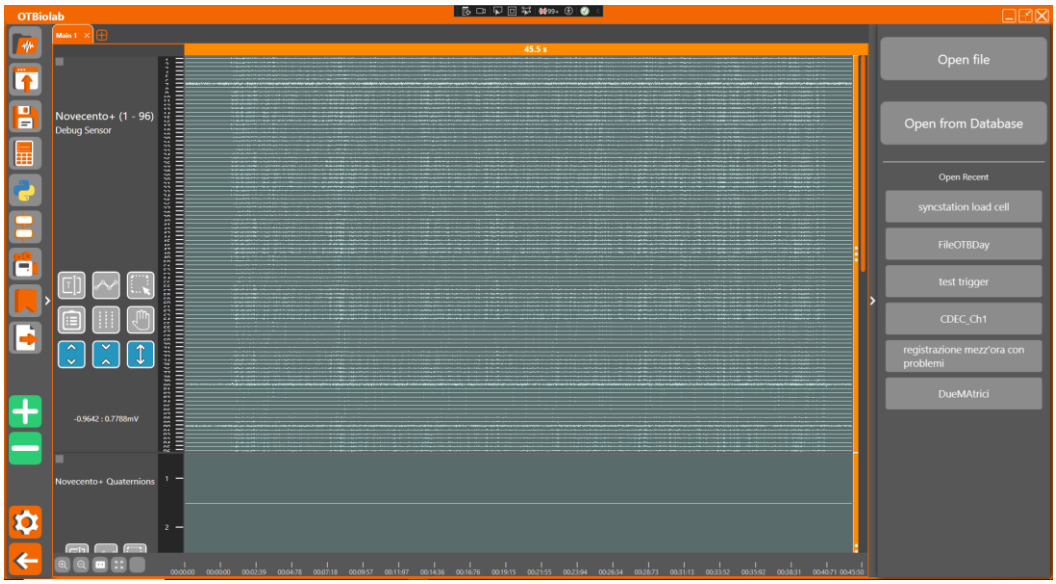
From the main menu click on the centre-right button to enter the offline window.



Click on the top-left button to open the right-side menu dedicated to files. It contains two buttons used to open and import and also the list of recent files, to give quick access to most recent data.



Click on **Open File** button and select a file using the dialog window. Then the software will load the data and create the list of tracks.

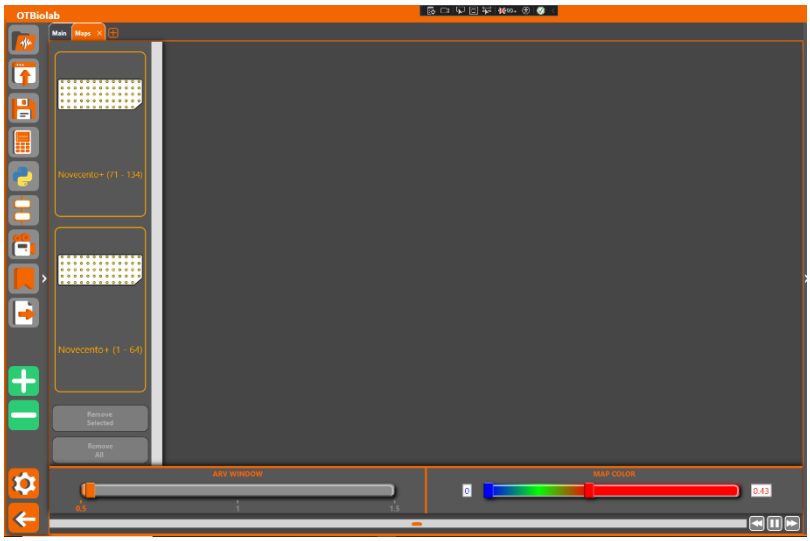


Using the following buttons is possible to change the horizontal scale of the file and set the favourite level of zoom. Then with a horizontal scrollbar the user can move along the time and look at the signals.

A vertical scrollbar is provided too to move along the tracks.
In case of grouped tracks is possible to separate them using the dedicated *"-"* green buttons. Viceversa it's possible also to group many single tracks into a grouped track using the *"+"* button.

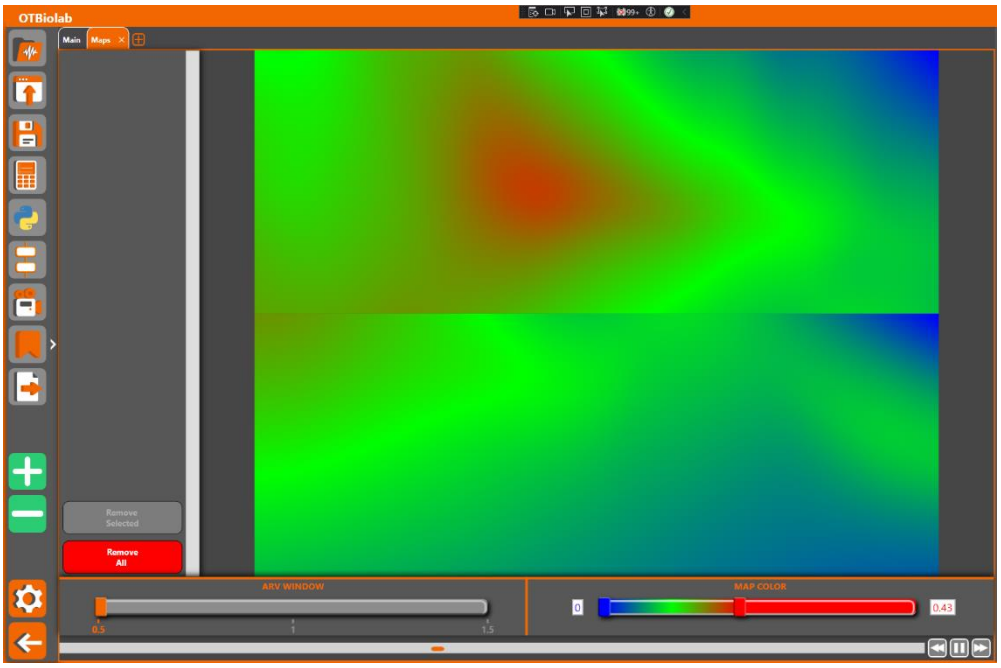
Activation Map

The activation map is a tool included in OTBiolab that allows to visualize the signals as a heat map, respecting the grid shape and place many grids side by side to replicate the acquisition.



The activation map window can be found in the offline window inside the Processing tab called EMG and it's composed by:

1) **A list of tracks**: the selected tracks are displayed here together with their grids (if possible) as buttons to be inserted in the centre part. With mouse drag and drop is possible to place them in the middle. Only group track can be visualized in this tool for obvious reason.
2) **The ARV slider**: this slider is used to determine the time window of the ARV calculation.
3) **The map colour slider**: this slider is used to set the minimum (blue) and maximum (red) values that will be used to display the heat map. Adjust the values accordingly (max, min values of the data)
4) **The heat maps**: the activation maps representation. Is possible to place multiple grid side by side in row-column visualization.
5) **Horizontal scrollbar**: used to move along time.



In order to prevent very quick value changes due to EMG data characteristic, the activation map doesn't display raw data, but the Average Rectified Value (ARV) is calculated before passing them to the graphical interface. The ARV value is calculated with a time window that can be changed using the slider and goes from 0.5s to 1.5s

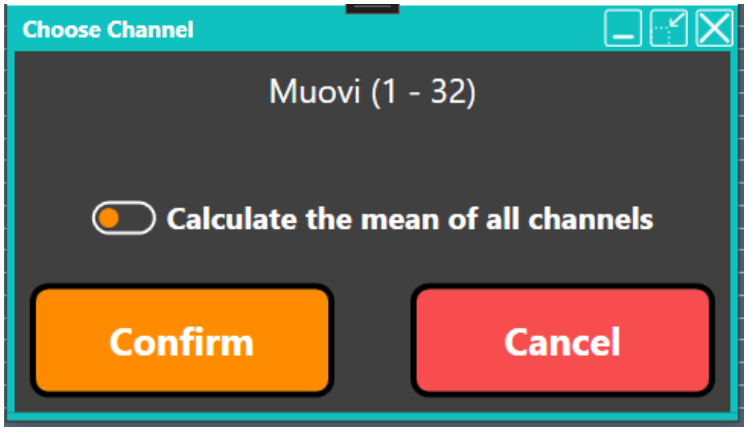
A realtime version of the activation map is available from the realtime window, by clicking on the following button. In this case the ARV is not calculated on a specific time window but on each single packet of device data.

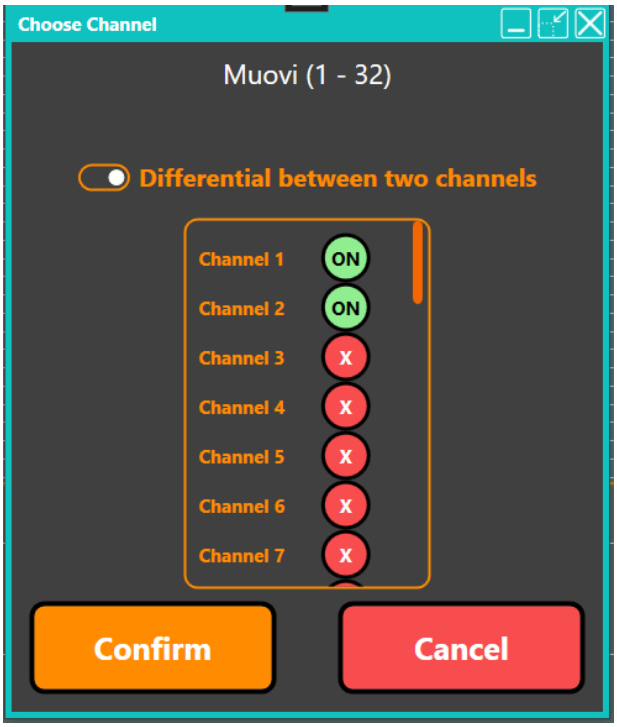
Trapezoidal Biofeedback

The Trapezoidal biofeedback is a specific tool used by many researchers out there in the previous OT Bioelettronica software and for this reason it has been implemented in OTBiolab too. The main purpose is to define a specific path and, based on the subject maximum voluntary contraction (MVC), make the subject to follows it during an acquisition. In order to do that, a specific window has been implemented and it's accessible from the realtime window, using the following button:

It's possible to run the trapezoidal biofeedback on a grouped track (a track that contains multiple channels) and in this case an additional information is requested.
If so, the software asks the user to specify whether he prefer to calculate a mean of all channels and using it to follow the path. Otherwise, he can specify to calculate the differential between two particular channels. With this solution, at each time instant, the software will calculate the difference between values and this difference will be used to follow the path.

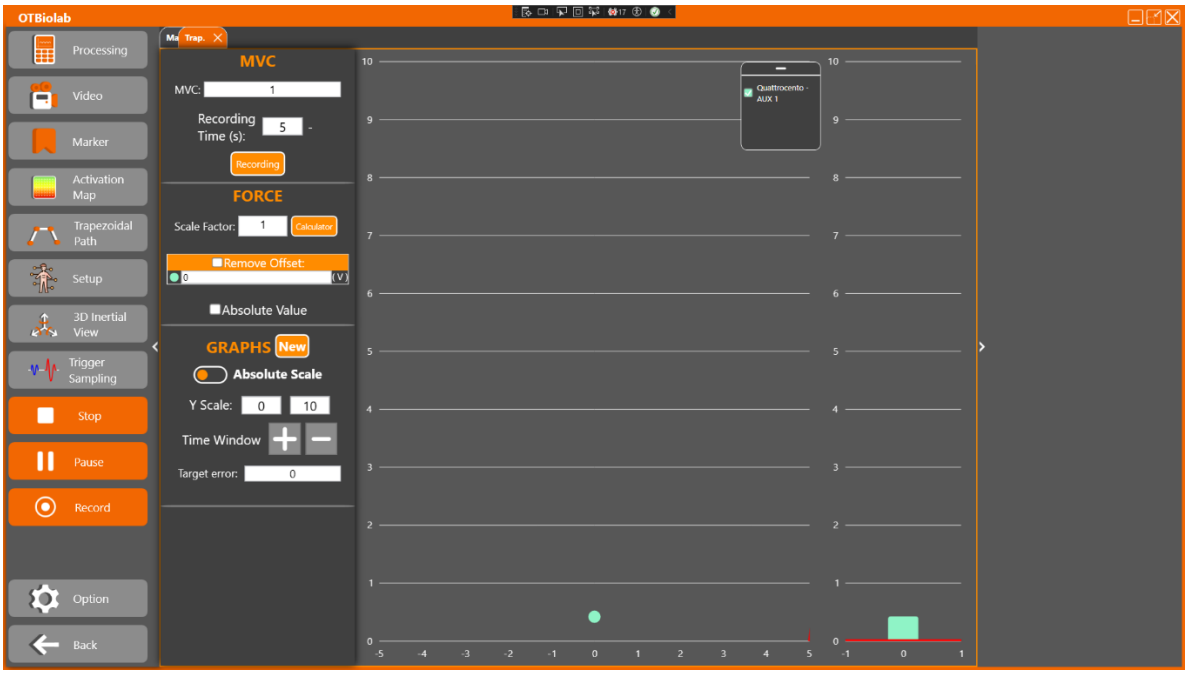
This choice is possible by clicking on the related toggle button.



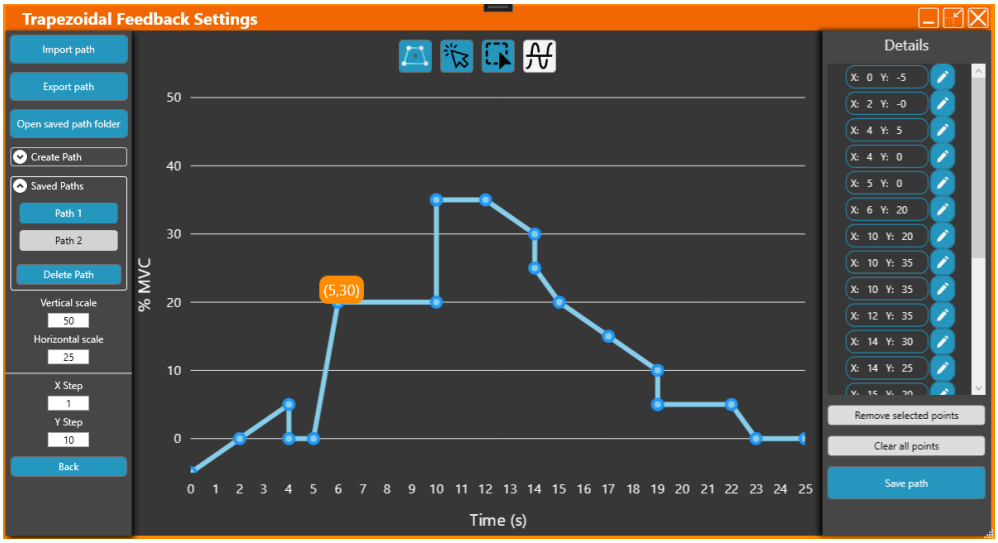


The displayed windows contains different controls that allows to add a **scale factor** for the data (very important if the customer use a load cell), **record the MVC** or write it manually, enable or disable the visualization of a specific channel (multiple channels can be visualized in the trapezoidal feedback), **zoom**

horizontal scale, set **relative or absolute** scale, open the **setting** to create a path and start or stop the **protocol**.



Creating the trapezoidal path is one the most important feature in the trapezoidal biofeedback and to do that a separate window has been developed. Using the **Settings** button it's possible to open it.



Through this graphical interface a path can be defined and saved along with the others and chosen from the user for the trapezoidal feedback. To create a path, it's possible to insert point by point or sinusoid from the left menu. It's possible to manually edit each single point values (x, y) from the right side list and also to swap position of two different point by clicking and dragging them.

Using the related button is possible also to manually click on the graph to insert a point.

When the settings window is closed the path will appears in the trapezoidal feedback window graph.

Absolute scale

The absolute scale displays the data of the track in the graph, the same values displayed in the real-time plotter window. In this mode it's possible to record the MVC, specifying the timer duration.

Relative scale

The relative scale changes the vertical scale of the graph and adapt the values accordingly. This scale set the MVC as the value 100 of the graph and it's used to display % of MVC in a 0-100% vertical scale.

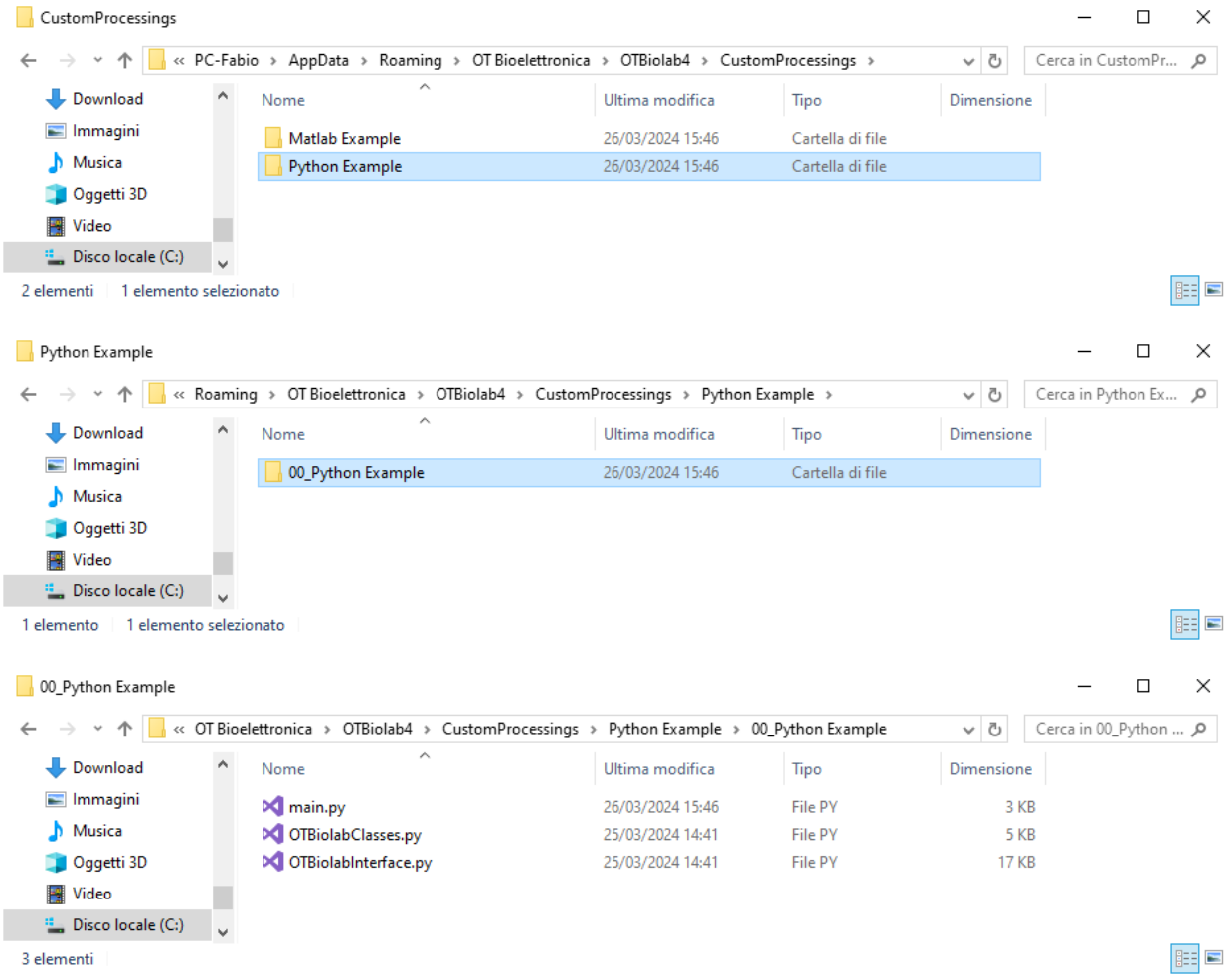
Start a protocol

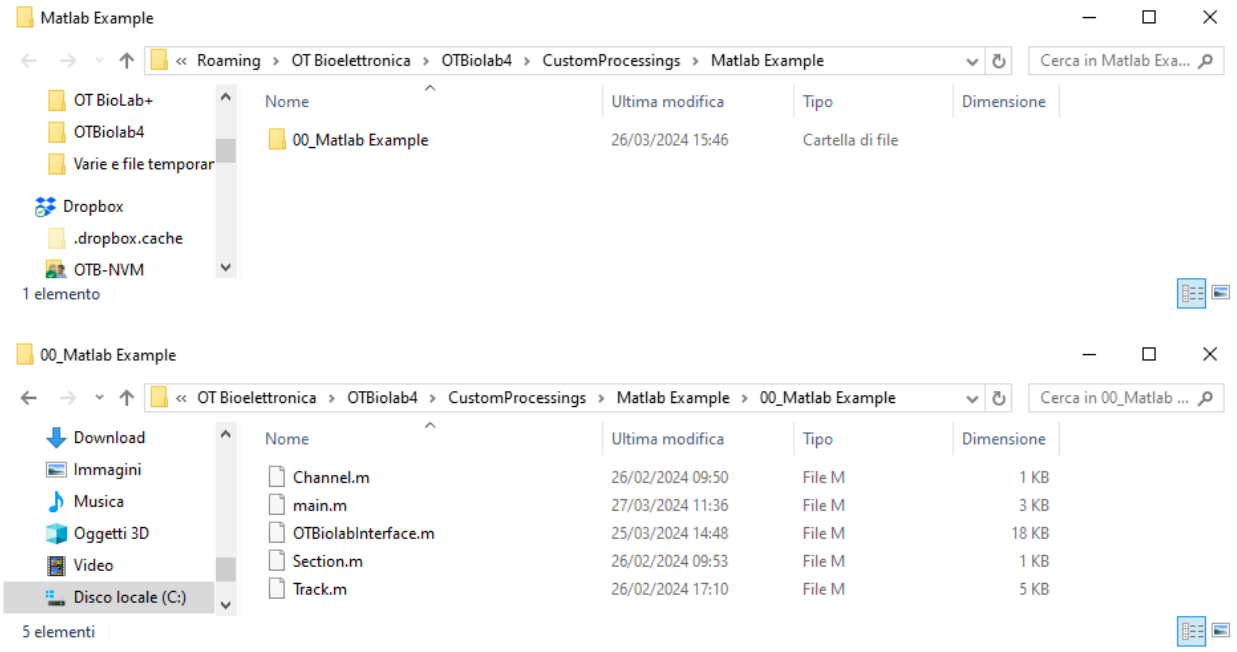
The trapezoidal protocol is the intended use of the Trapezoidal feedback and can be accessed following these steps:

1) Create a path using the setting window.
2) Write or record the MVC, this is mandatory in order to have the correct vertical scale.
3) Switch to Relative scale.
4) Start protocol.

Custom Processings

The software allows to run MATLAB and python script in an easy and intuitive way. From the point of view of software, a custom processing is a simple folder system with files written in python or MATLAB containing different code lines. The main folder of a custom processing gives the title to it, and inside of it there are other folders containing the different step of the processing. Every step can be written either only in MATLAB or only in python, but inside of a processing there can be different steps, this mean that a processing is a chain of steps, where these can be all python steps, all MATLAB steps or python and MATLAB steps. This structure is mandatory to import a processing inside the software, and every step needs to contain a “main.py” file for python steps, or a “main.m” for MATLAB steps. In the next chapter called “Graphic Editor” we’ll also see how this structure can be used to merge different processing by chaining the steps of each one inside a “merged processing”. Here a resume images of processing structure:





As you can see from the images, the “00_Python Example”/”00_Matlab Example” steps are identified by a number which represents the execution order, in case the processing contains different steps. Give a look to the files contained in the step folder, the “main” is the file that will be mainly updated by the user, it’s the script that contains the main elaboration. The other files provide different functions to interface the script with OTBiolab software and will be fully explained in the next sub-chapter.

Additional script files

The main feature of the custom processing is the possibility to run your custom scripts by selecting data from OTBiolab software and plot the results and visualize them in OTBiolab. To create this bridge an interface library has been developed and added to the custom processing, if the user wants this type of feature it needs to be integrate in its own script. This library has been built with different files and from a first point of view seems to be different between MATLAB and python steps, but the functionalities provided are exactly the same and can be generally grouped in “Classes” and “Interface functions”.

1. **CLASSES:** the classes used by this library are included in OTBClasses.py file and Channel, Section and Track files for MATLAB. The programming language is different, but the concepts are exactly the same, indeed in the OTBClasses.py file the same classes are defined and included. This explanation is meant for all those who want to update the classes and insert new methods or properties to create more complex algorithm, actually the only purpose of these is just to have a simple container to access the data.
    a. *Channel:* this class represents the data contained by each channel of the track. Inside of it there is a simple property which contains a list of double values.
    b. *Section:* this class represents the section selected in the track of the software. When you decide to select just one or multiple pieces of a track, this is translated in section class, one for each selection the user did previously. Inside the section class there are a *start* double value, representing the beginning in seconds of the selection, an *end* double value, representing the end in seconds of the selection and a list of *channel* objects which contain the data.

c. *Track:* this is the more complex class compared to the other and represents a python/MATLAB copy of the track used inside OTBiolab. Inside of it are defined, a *title* field, a string representing the track title of the software, a *unit of measurement* field, a string representing the unit of measurement of the initial track, a *frequency* field, a double value representing the sampling frequency, a *number of channels* field, an int value representing the number of channels included in the tracks, a *device* field, a string representing the device name used to acquire the signals, and finally a List of *Section* to access the data. There also different methods defined that can be used to interact with the main software after the processing execution.
SaveData: this method requires a path and a track index to create a file ".data" containing the data with a unique name. This file format will be highly explained in the chapter "Run selected processing".
SaveText: this function takes a path, a track index number and a channel title string as input. It creates a ".text" file where the data contained in the track are saved in a text format by adding for each channel the channel title specified as input. The file will be saved to the patch with a unique name created by track index number. This file format will be highly explained in the chapter "Run selected processing".
GetDataFromSections: this function takes as input the index of a section and returns a matrix containing the data contained in that section.
Plot: this function creates one plot for each section contained in the track.

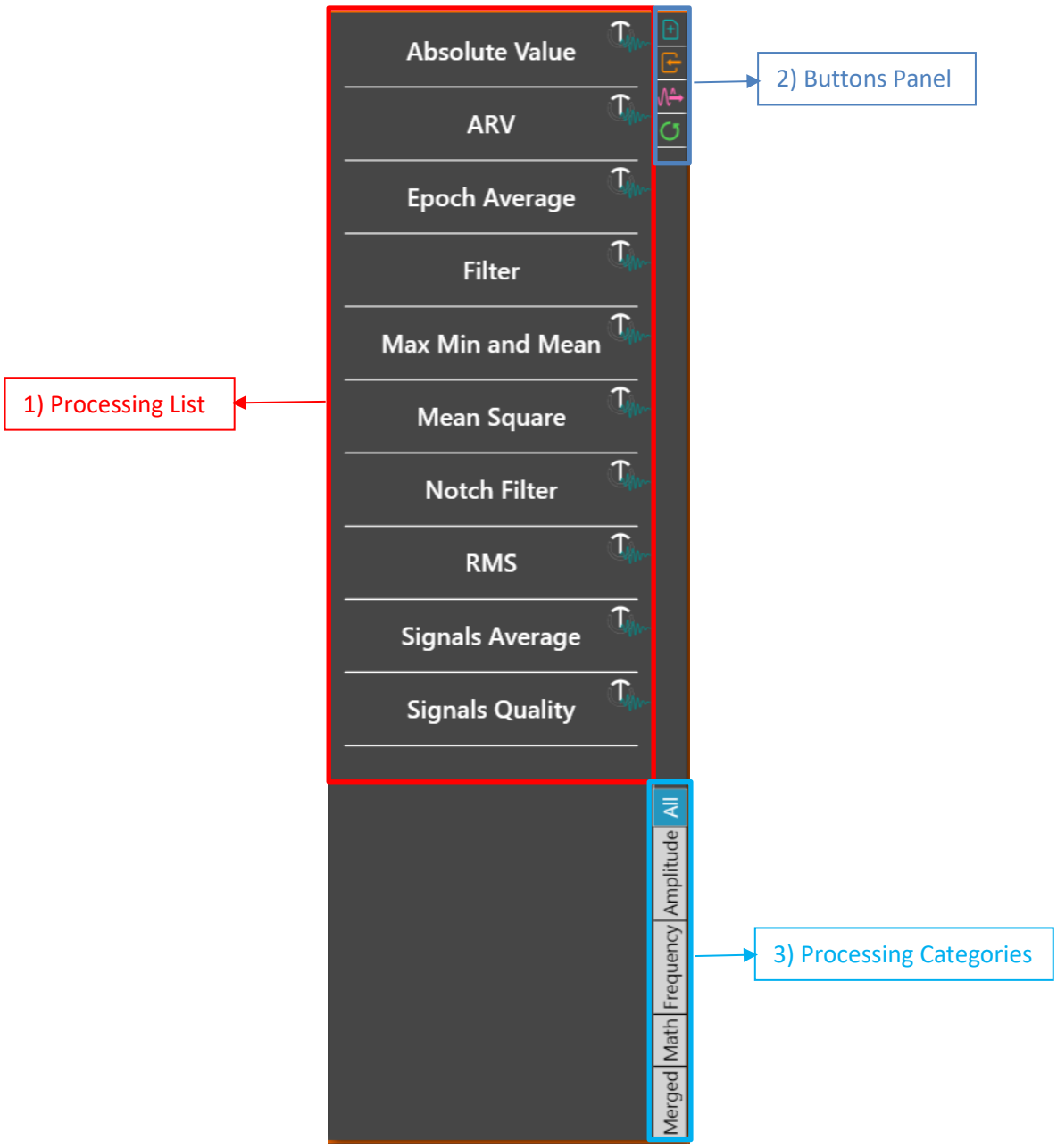
2. **INTERFACE:** the interface part is managed through different functions which use the classes to communicate with the main software. There are different functions inside of it, and the main are explained below:
   a. *OpeningOTBPlusFile*: this function receives a path to otb+ file and returns the list of tracks loaded by this file. The track type is defined in the Class explanation.
   b. *OpeningOTB4File*: this function receives a path to otb4 file and returns the list of tracks loaded by this file. The track type is defined in the Class explanation.
   c. *LoadDataFromXXFolder*: this function is used to read all the ".data" files contained in the Common Folder and returns the list of tracks, one for each file, as output.
   d. *WriteDataInXXFolder*: this function gets a list of tracks as input and call for each one the method SaveData explained before. Each track will create a ".data" file inside the common folder that can be used to another processing or used by the software to plot the data.
   e. *WriteMessageInXXFolder*: this function gets a list of tracks as input and call for each one the method SaveText explained before. Each track will create a ".text" file inside the common folder that will be used to create a message box in the software.

ATTENTION: apart the main the other files are totally different between python and MATLAB and need to be modified with extreme attention, because it can influence the proper script work. These files are automatically generated by the software when a new processing is created but the only mandatory file in each step is the "main" file.

Custom Processing Panel

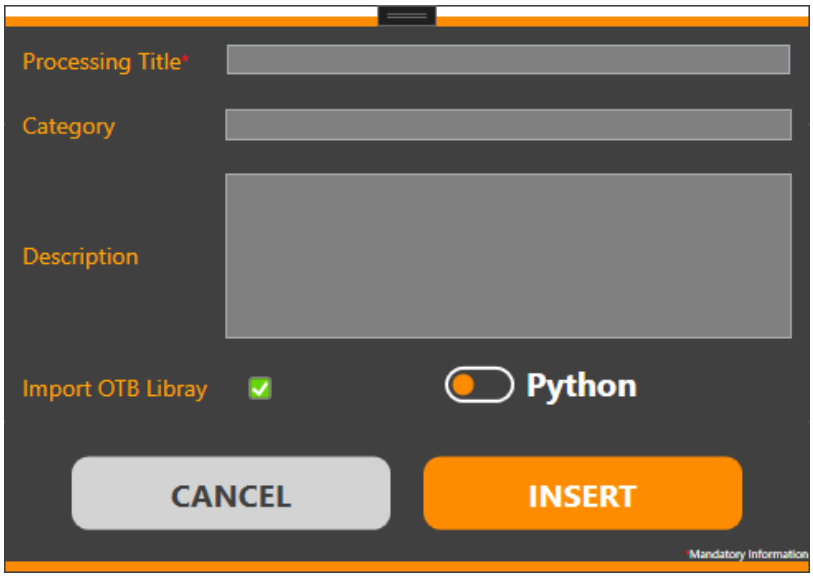
To open the Custom Processing menu the user needs to click the button called “Custom Processing”:

This will open the list of the current custom processing saved in the software in the right-side menu. If this list is empty means that you missed to install the necessary components to run python/MATLAB script, follow the “Additional Components” chapter to see how to handle it. If the user has installed both python and matlab tools, he should be able to find the default python/matlab processing installed during the procedure.



The processing list is filled with the processing actually found in the software folders specified in the "Settings" View (see chapter "Additional Components"), this includes the default Matlab/Python processing, and the new processing created or imported by the user. Each processing is classified with a category, that can be used to filter the processing list by clicking in the "Processing Categories" bar. Each category is a button which selects the processing under that specific category. The "Buttons Panel" allows user to interact and update the processing list, below a full explanation of the different buttons:

1. **Create New Script ( ):** With this button the user can create a new processing to add in list. This feature needs to be used when the user wants to create its own script with its own code. We will see the entire workflow in the next chapter. The window related to new script tool is the following:



The processing title is the only mandatory field, the others are set default if the user doesn't digit nothing. The category is the same field used by the "Processing Categories" panel. If this category is the first one, it will be added to the list automatically. If no category is specified in this window, the "CUSTOM" category is used as default. The description is a simple descriptive field of the processing, if it's not added, the "–"character will be used. The "Import OTB Library" checkbox is simply used to create a basic template of the new processing, if this is toggled a basic structure template with a default elaboration is added to your processing, if not a total empty main file will be used. The last toggle is used to select if the user wants a python or a MATLAB processing. When the "Insert" button is clicked the folder representing the new processing is created, and all the necessary files are moved in it with a dedicated step folder and a main file with a different template depending on "Import OTB Library" checkbox. The generated templates are quite important to use all the functionalities provided by the main software, indeed the software shows different information of the scripts that need to be placed in specific code blocks to be found and captured. This information is the *CATEGORY,* the *DESCRIPTION* and the *PARAMETERS*. Here the images and the explanation of these blocks:

When the processing is loaded and in the main file there is this type of block (left python, right MATLAB), everything that will be written inside the limiters and under the "DESCRIPTION" word will be used to fill the description in the Selected Processing Panel.

```
''' CATEGORY
'''
```

```
%{
CATEGORY
%}
```

When the processing is loaded and in the main file there is this type of block (left python, right MATLAB), everything that will be written inside the limiters and under the "CATEGORY" word will be used to fill the category in the Selected Processing Panel. If the category has not been found between the processing contained in the processing list, a new category will be added. This category can be used to filters the processing list as described in "Custom Processing Panel" chapter.

```
############################################### PARAMETERS ##########################################################

#var1=...
#var2=...
#var3=...

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

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% PARAMETERS %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%var1=...
%var2=...
%var3=...

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

When the processing is loaded and in the main file there is this type of block (upper python, lower matlab), all the lines that are not comments will be parsed to find parameters and variables to be used and showed in the Selected Processing Panel. This block is overwritten in the code if the parameters are changed from the software interface and will be substituted in the software if the code is changed from the editor and reload. The variables that can be parsed and recognized are int, double, float, string, char and bool.

ATTENTION: The collections are not recognized as type, so if you write a list in the parameters block it will be recognized as string.

2. **Import Processing ( ):** With this button you can import an existing processing to your list and use it like the other processing. The software does different checks about the processing folder structure, that needs to be exactly the same to the one described in the introduction, and the presence of main files in the different steps of the folder. The best solution to add your code in the custom processing list is creating a new processing and moving your files inside the step 0 folder by renaming the main file as "main.py" or "main.m" depending on the script language.

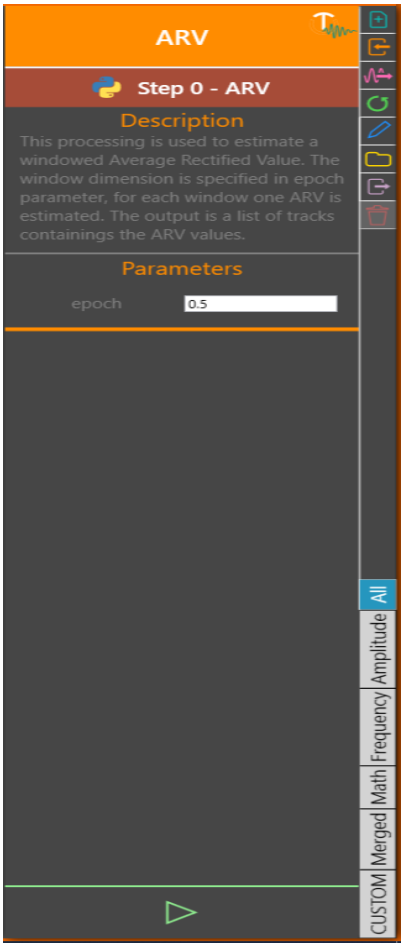
3. **Export Signal ( ):** the export signal functionality it's a really important feature for all the users that want to develop their own code. As we'll see in the next chapters, the software provides a simple text editor to visualize the custom processing codes and update them with a lot of simplification. Obviously, this editor is not a true IDE like MATLAB Ide, Spyder or PyCharm, since it doesn't provide debugs instruments or other mandatory functionalities for code development. For this reason, OTBiolab provides to user the possibility to develop its own codes but using the

signals and the data visualized in the software. The communication between main software and custom scripts is done through specific files created before the run event, and that need to be loaded by the script to have the data to elaborate, this allows to run the custom processing on the signals visualized in the software, and to visualize subsequently the results of the elaboration. The support files generated by the software in the custom processing folder contain all the functions to load and write these data, and the data loading and writing are already written in the main template, in this way, if the processing is created and opened with its dedicated IDE, it will work if the proper data file will be found in the proper folder. This can be done through the Export Signal function, just select the tracks that you want to use in the processing, click the button and the files containing data will be moved in the proper folder, now the new custom processing can be opened and, in the main file with the support functions of the template, the user will be able to run the processing with OTBiolab data in any IDE.

4. **Refresh Scripts ( ):** since the custom processing are simply folders with files inside of them, the user can also manage them through operating system or other software, with this button the software reloads the processing from the proper folders, in this the external updates can be transposed to the software.

Selected processing panel

From the processing list you can choose one of the custom processing and it will be opened in the right-side menu. Let's see an example:



In the processing panel there will be a list of steps contained in the processing (in this case just one), with a description of it (when the user creates a new processing, the description of the windows is used as description of the first step, see Chapter Custom Processing Panel), and a list of parameters that can be changed by the user. In the button panel new buttons that manage the selected processing will be added, below a complete description of these ones is reported:

1. **Open script editor ():** As described in the previous section, the custom processing are simply folders which contain matlab/python script. With this button the user can open a simple text editor to visualize and modify the codes. It will be fully explained in the next chapters.
2. **Open project folder ():** this button opens the processing folder in a folder dialog window. This can be useful if the user wants to manage directly with the files contained in the processing. Remember that after changes you need to reload the processing with the "Refresh scripts" button previously described.
3. **Export processing ():** this function will create a copy of the processing folder in a destination chosen by the user through a folder dialog window. It can be useful to update or create a new processing starting from an existing one without touching the original.
4. **Delete processing ():** this button is used to delete a processing and remove it from the processing list. For this function a distinction between custom processing is necessary. The processing installed with the "Install" function previously described, are called "Default Custom Processing" and these cannot be deleted or updated in the software. Indeed, in this type of processing this button is disabled and cannot be clicked. The processing created by the user are called "User Custom Processing" and these cannot be deleted and update through OTBiolab. To distinguish the two types of processing a symbol in the high part of processing button has been inserted.

(Default) (User)

Other differences are in the editor page, but these will be fully explained in the dedicated chapter.

Run selected processing

In the final part of the processing panel the start button can be used to run the processing (chain of steps) in the tracks selected by the user in the plotter view. The software and the scripts communicate each other through specific files created before the processing is executed and by reading the results in other files when that processing is concluded. This file exchange is done in a specific shared folder located at "C:\Users\%CURRENTUSER%\AppData\Roaming\OT Bioelettronica\OTBiolab4\FileToProcess" folder. When the processing is started different files are generated with different extension:

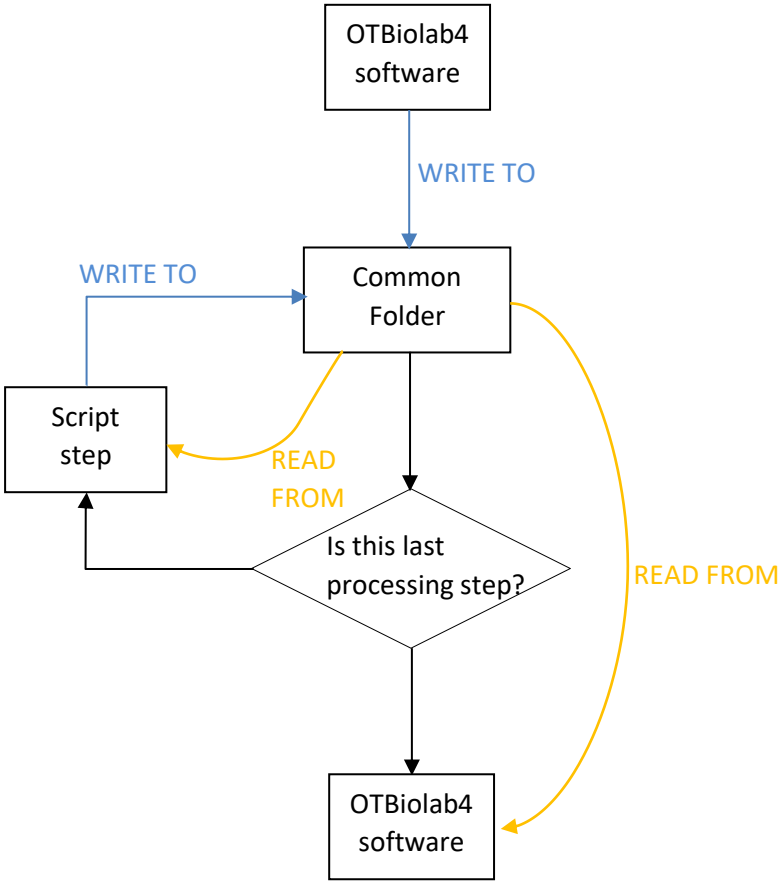
1. **.data files:** these are binary files which contain the data and all the information related to a specific track. Indeed, for each track with at least one selection one file of this type is created.

   The first part of the file is a sort of header that contains, track title, sampling frequency, number of channels, unit of measurement, after the header there are data divided in sections depending on how many selections the user has done before running the processing. For each selection the

start time and end time of it are written in the file, and after these the data are saved in double format, this means that from the script point of view there is no need of conversion. This section logic is iterated for all the sections of the user. The title is associated with a number to create a different file for each selected track.

2. **.common file:** this file is created to resume the files path involved in the processing. It contains a list of otb+/otb4 that are currently opened when the run button has been clicked.

These files are necessary to communicate with the scripts, and from the script point of view the user can decide to directly open the files by using the .common file or create a copy of the tracks in the code by using the .data files. If the scripts want to plot the results in OTBiolab needs to create files in a similar format, in this way the software will check the same "FileToProcessFolder" and will show the results in different way depending on the extension of the files. The files can be ".data" like before, in this way the software will create a track for each file and will add this on the current tab, or ".text" files, which are file that can be created as result file, and which are visualized in a message box of the software. So, by cleaning the "FileToProcessFolder" and exchanging ".data" files the processing can be chained to multiple steps and obtaining a result viewable in OTBiolab software. Here a schematic resume of this process:



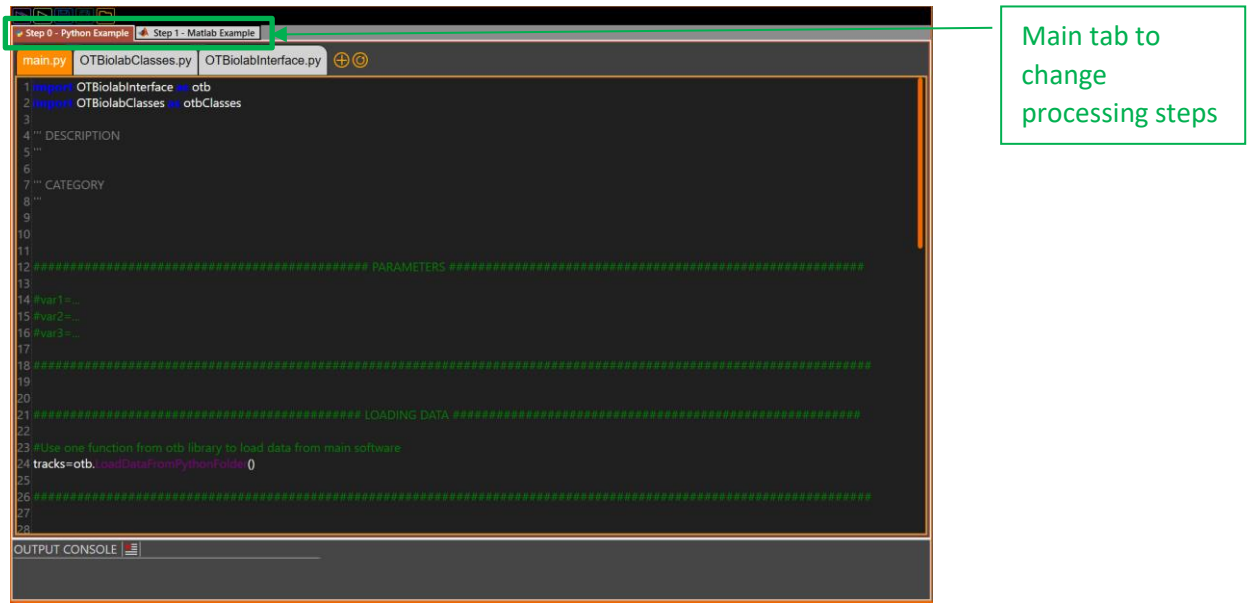
As can be seen in the figure, the software starts the processing by writing the selected track in ".data" format inside the common folder and then the different script steps are executed by repeating the operation of reading from common folder the data and overwrite the results to common folder until the steps are all executed. At the end the software does one last reading from the common folder and translated the .data files in tracks which will be showed in the proper page. The software in the last step,

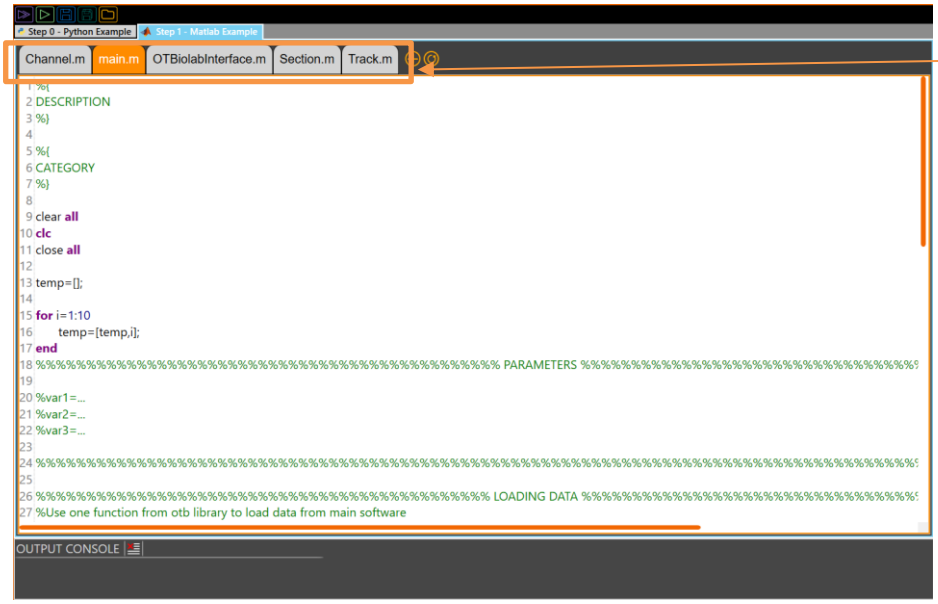
will check for the files contained in the common folder by reading them and depending on the data format it will be manage them in different way. Here a list of the possible files:

1. **.data files:** same as the previous description. If the software finds this type of files inside common folder at the end of the processing, it will translate these data in tracks that will be showed in a dedicate page.
2. **.text files:** this file is a binary file which can contain processing results. If the software finds this type of file, it will create a message box that contains the results written in .text file like a message string.

Code Editor

As described in the "Selected processing panel" chapter, through the button "Open scripts editor" the user can open a text editor to visualize and modify the codes contained inside the processing folder. The editor will automatically recognize if the files in the steps are python or MATLAB and a view with dedicated buttons, syntax highlights and functionalities will be opened. Here the images of editor opened for MATLAB and python steps:





Secondary tab to change visualized file

We are visualizing a User Custom Processing that involves two steps, the first one written in python and the second one written in MATLAB. As we can see, the theme is different in the two steps, python code has a dark theme while the MATLAB code has a white one. Furthermore, the correspondent syntax, like comments and keywords, is loaded by the editor to highlight specific text pieces like comment and keywords. The editor provides a "Main Tab" control to visualize the codes of the different step. Then a "Secondary Tab" control is provided to switch the codes and moving inside the step processing codes contained in the correspondent folder. Below there is the "Output Console" where different messages are showed in different colours:

1. **Green Messages (Software information):** This message represents a normal information of the software, like the scripts saving, or what script is running at the moment.
2. **Yellow Messages (Warning):** This message represents a software warning that it's not blocker. This type of message needs to be investigated and for the proper processing execution is better if no warnings are showed. The classic message of this type is the warning that tells user that the script cannot be saved because it's a "Default Custom Processing". Indeed, the default processing cannot be modified and for this reason cannot be saved by the software.
3. **Red Messages (Errors):** This message represents a potential error found during the current script execution. This message, despite the others, is related to script working and not to OTBiolab. If a red message is visualized the script is failed, the message will provide you all the necessary information to understand the script problem and solve it.
4. **White Messages (Output):** the software get all the output generated by the script, like "print" functions or the standard channel output of a matlab script. This output is represented through this white message.

Near the Secondary Tab, there are two buttons with different purposes. The first one () is used to create a new file called "Untitled" of extension .py if the focused step is Python or .m if it's MATLAB. The file will be opened with a "*" in the name tab, that means that it's never saved, so the first time that the user run the step or the file, a file dialog will be opened to confirm the name and the path of this new file. The second one () needs to refresh the step content. In a previous chapter we described the processing as a chain of steps represented by folders, so the user can manage them also with other

software or just with operative system by moving files or changing folder names and so on. This button is necessary to reload the content if this is changed out of the software. This can also be used if the user wants to remove some files in the processing, he can open the processing folder, remove the files he wants and refresh the content with this button.

Editor button panel

In the upper part of the editor there is a panel containing different buttons that allow the users to save and run codes. Here the buttons functionalities:

- **Run all steps ():** this button run all the steps starting from 0 in increasing order. In each step, the code focused on the Secondary Tab will be executed. Before each execution the code is saved (if possible, remember that Default Processing cannot be modified/saved) and then executed. For each executed code, a message containing saving information and current code title is showed.
- **Run single code ():** this button executes just the code actually visualized, so a precise script contained in the step actually pointed from Main Tab. The script executed is chosen by the user, it will be the main code just if the user is visualizing the main code. Remember that before execution the code is automatically saved, and the info message will be showed in output console.
- **Save ():** this button can be used to save the specific file actually visualized. This will generate a message about the success or failure of the saving.
- **Save all ():** this button repeats the Save operation but for all the files contained in all the steps of the processing. A dedicated message will be showed in the console for each file saving.
- **Open Folder ():** this represents the redundant functionality of opening the processing folder. Can be useful if the user wants to change some file name or add other files in a file explorer window and reload the processing content with the “reload” button previously described.

ATTENTION: if the user update or add some parameters in the code editor and save the modified files, to find the same updates in the Processing Panel List, the “refresh scripts” button explained in the chapter “Custom Processing Panel” need to be clicked.

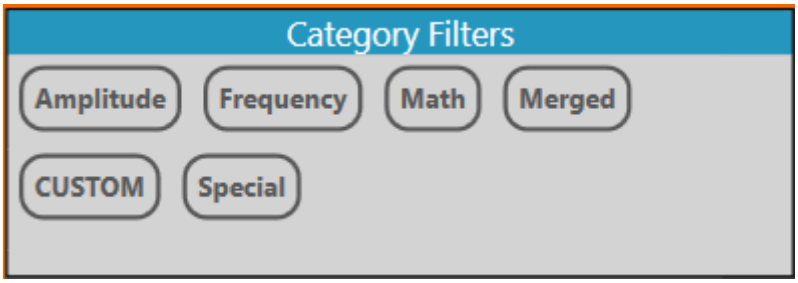
Graph Editor

The graph editor functionality can be used just in combination with custom processing developed in MATLAB/Python codes, this means that at least the installation procedure of python components or MATLAB components or both, showed in the “Additional Components” chapter, must be completed. Once that the python/MATLAB components have been installed, the graph editor functionality can be used. The base concept of this tool is the folder structure of the processing and the step chain contained in it, indeed since the processing execution can be seen as a cycle operation of read-elaborate-overwrite in the same common folder, the graph editor allows user to merge the steps of different processing by creating a new one with the initial steps chained one after the other. To open the graph editor, the user needs to click the following button of the left panel menu:

A new view will be opened and added in the main docking of the platform:

The main components of this page are the processing list in the left, the graph in the middle, the manage panel below and a details section that is not visible, but it will be highly explained in the next chapters.

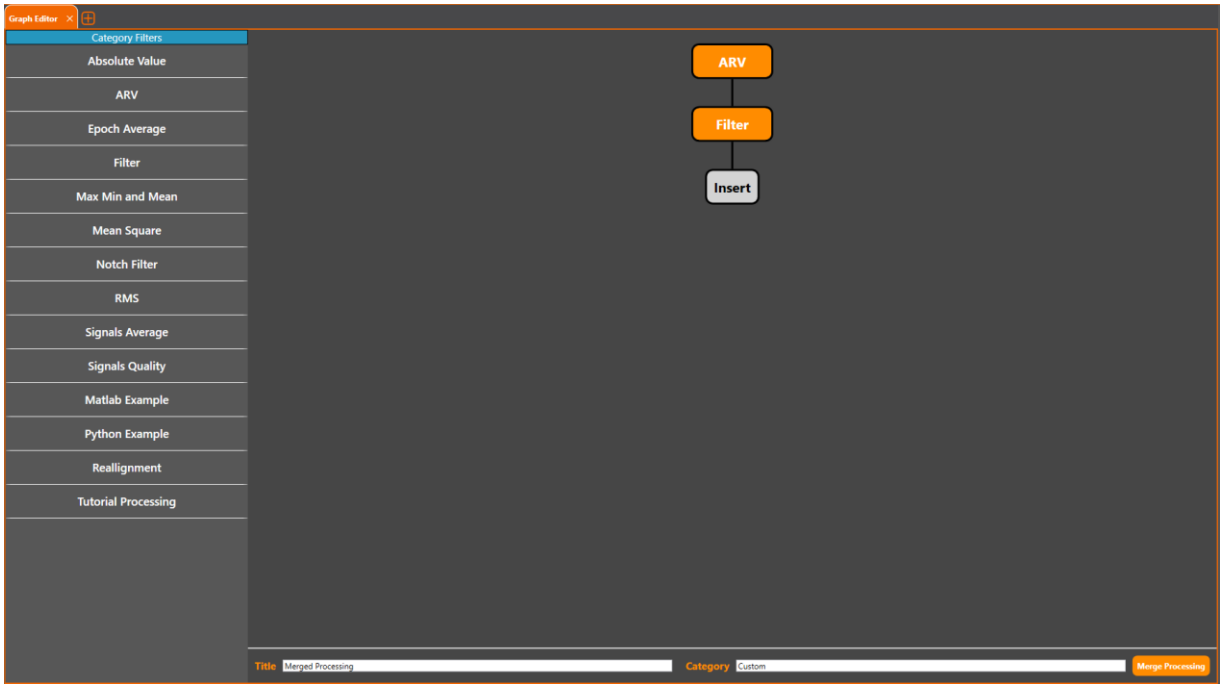
Processing list: it contains all the custom processing, both the default ones added with the installation procedure and the ones created, added, or imported by the user. In the higher part of the list there is a field called “Category Filters” that can be expanded by moving over it with the mouse, like in the image below.



The options contained in it are the single categories found in the processing list and can be used to filters the processing list. The options inside this container can be activated or deactivated, when no categories are activated all the processing will be showed, when one category is activated all the processing which share that specific category will be showed. The user can select more categories and all the processing that share at least one of the activated categories. The processing inside the list is draggable elements, this means that the user can click on a processing and drag it to the graph in the centre.

Processing graph: As can be seen in the image there is a graph with a single node called “Insert” placed in the centre of the view. The “Insert” node can interact with the dragged processing, and if the user leave the mouse when the processing has been dragged over the node, it will be inserted in the graph as first node, like showed in the figure with the ARV processing.

As can be seen in the image a new “Insert” node appear after the “ARV” node and can be filled in the same way with another processing. Let’s suppose for example a filter processing.



In this way the user can obtain a chain of processing by dragging in the “Insert” node the desired element from the list. The “Signals Quality” processing has been obtained in this way and added to the default processing folder.

Manage panel: The below part of the page can be used to conclude the merge of the processing. By inserting the title and the category of the new processing when the user clicks the Merge Processing button the software will get all the processing in each node and in the insertion order of the graph and it will create a new processing folder that will be filled with all the step folders found inside the involved processing.

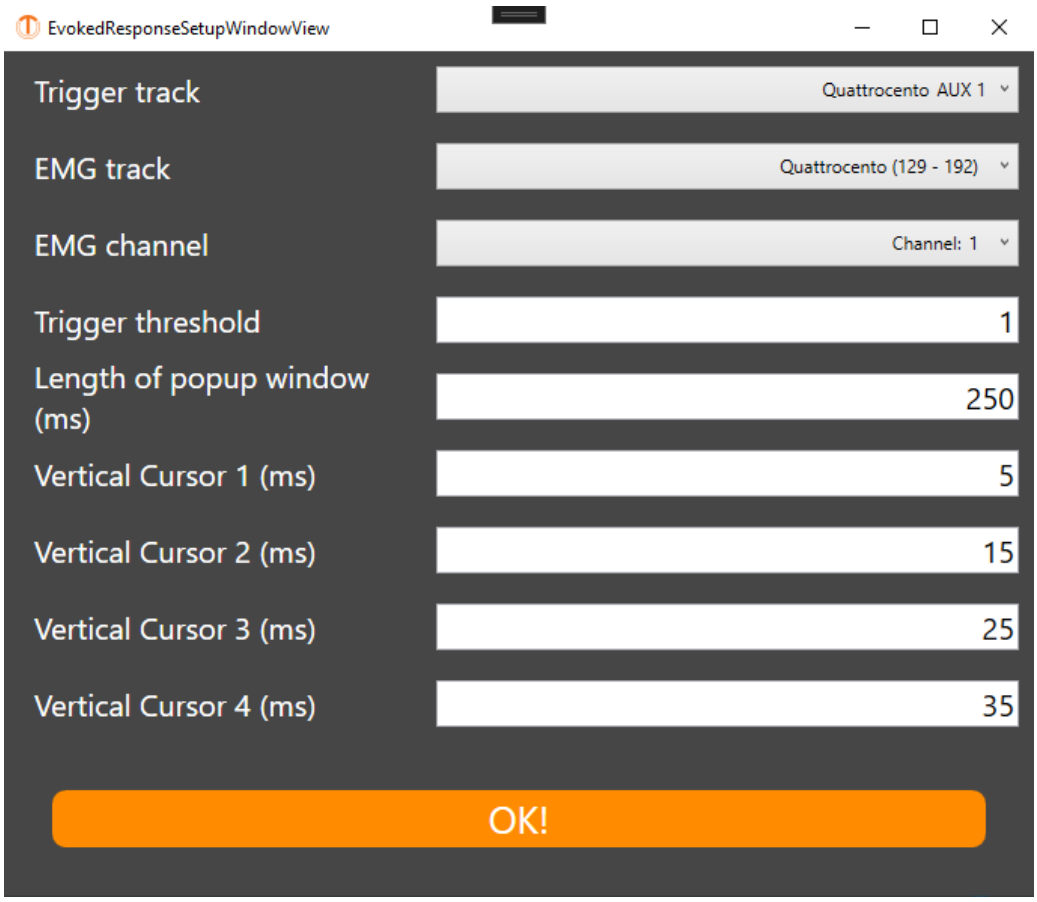
Node details panel: There is one last page section that need to be discussed and that is initially hidden in the software. Each node can be opened and expanded to visualize what are the steps inside of it, to do it the user can just move over the desired node until the writing “Expand” is visible. If the user clicks with mouse on it, this will open a new panel near the graph editor which contains all the steps inside that processing and the node title will be substituted with the “Expanded” writing. This means that the graph editor can be used to explore the content of a processing, let’s assume that we have a merged processing, and we want to see its steps without opening the proper folder, the user can just drag it in the insert node, click on it and check the Node details panel. Let’s see an example with the Signals Quality processing.

The right panel will contain all the steps included in the expanded processing node under the shape of a graph. Each step will have its own description and will be executed in the exact same order of the graph. So, if the user creates a merged processing with a lot of steps, to check them in a graphical visualization he can move the merged processing just created in the “Insert” node and expand it as explained before.

Trigger Sampling

The purpose of this feature is to refresh a particular channel, visualized in a separate window, every time a stimulation occurred. In order to access this feature the user should navigate to the realtime window, start an acquisition (check related chapter of the manual) and click on the Trigger Sampling button.

The first window that appears is the configuration window that allows to setup all of the parameters needed by the Trigger Sampling mode.



In details:

1) **Trigger track**: the list is filled only with single tracks (AUX channel for example). The selected track is used to refresh the evoked response plot when the threshold is reached.
2) **EMG track**: the list is filled with EMG track (grids, matrices, arrays, bipolar). Select the track you want to take the data from.
3) **EMG channel**: select the channel of the grid you want to see in the evoked response plot.
4) **Trigger threshold**: value used to refresh the plot. This value is expressed in the same unit of measurement of the Trigger track. So, if it's in volt, write the value in Volt.

5) **Length of popup window**: specify the plot x-axis time. This value is used to bufferize the data, consider that data comes from device every 0.0625 second and contains fsamp/16 samples circa.
6) **Vertical cursors**: insert vertical lines where to create areas and calculate the Peak-to-peak.

Once the user has decided each parameter and clicked “OK!”, the software opens the Trigger Sampling window.

#STIM ID	1-2 (mV)	3-4 (mV)	Cursor 1 (ms)	Cursor 2 (ms)	Cursor 3 (ms)	Cursor 4 (ms)
1	1.664	1.648	0.005	0.015	0.025	0.035
2	1.512	1.475	0.005	0.015	0.025	0.035
3	0.243	0.264	0.005	0.015	0.025	0.035
4	1.643	1.748	0.005	0.015	0.025	0.035
5	1.624	1.64	0.005	0.015	0.025	0.035
6	0.264	0.289	0.005	0.015	0.025	0.035
7	1.608	1.629	0.023	0.08	0.15	0.199
8	1.618	1.603	0.023	0.08	0.15	0.199
9	1.605	1.639	0.023	0.08	0.15	0.199
10	1.605	1.639	0.023	0.08	0.15	0.199
11	1.569	1.881	0.023	0.08	0.15	0.199

The screenshot was taken without a real setup and the signal display a channel with no sensor attached. There is no useful information about the EMG channel but it’s good enough to describe the window and its parts.

The top part contains a plot of the data with the same time visualization specified in the setting window (length of popup window). Vertical scale instead is automatically adapted based on the data.

When the Trigger Track exceed the threshold, the data of the selected EMG track are displayed in this new window and a new record is added in the table.
The plot contains four different vertical bars (two blue and two red ones), that are used to determine two separate areas. Using the mouse is possible to select and more each separate line. These areas are used to calculate the peak-to-peak value of the data and place this information in a table record.

The table columns are:

1) STIM ID: stimulation ID used to identify the record of the table
2) 1-2 (mV): peak to peak value of the data inside blue zone
3) 3-4 (mV): peak to peak value of the data inside red zone

4) Cursor 1 (ms): time position of cursor 1
5) Cursor 2 (ms): time position of cursor 2
6) Cursor 3 (ms): time position of cursor 3
7) Cursor 4 (ms): time position of cursor 4

In the top right part there are three buttons:

1) **Save button**: can be used to save the plot as a picture
2) **Left/right arrows**: can be used to navigate the table and visualize old plots.

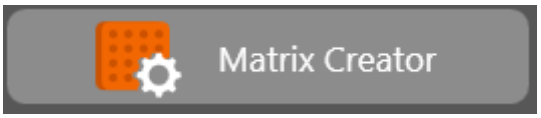
Navigation between old plots is also possible by selecting a record of the table with the mouse left click.

One last button can be used to export the table as a csv file.

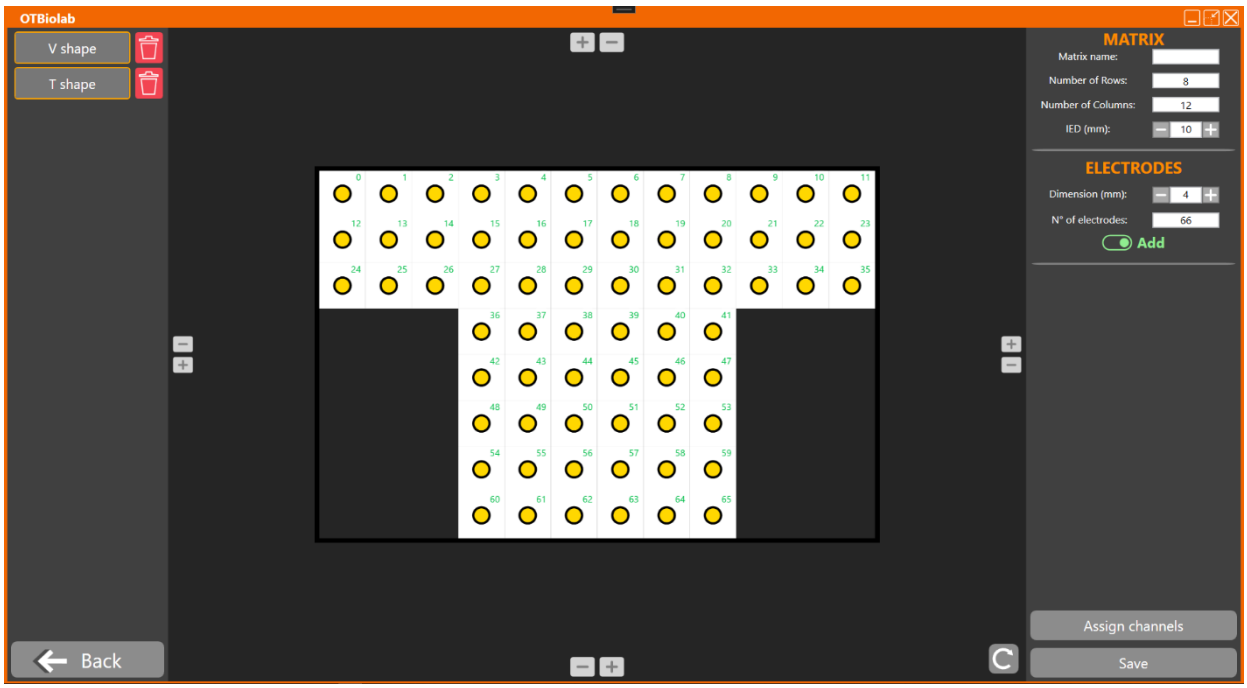
Matrix Creator

The Matrix Creator tool is a feature designed for users who need to create custom matrices for signal acquisition. These matrices, composed of rows and columns of electrodes, can be tailored with varying shapes, sizes, and configurations, enabling precise customization for different applications.

To access the Matrix Creator, navigate through the main menu and enter the setup section of the software. From there, select the *Matrix Creator* button.



This action opens a new window where the tool is displayed, with a default matrix at the centre of the screen. The interface is intuitive, offering a variety of options to modify and customize the matrix according to the user's needs.



Surrounding the default matrix are buttons marked with + and - symbols on each side. These controls allow users to increase or decrease the number of rows and columns in the matrix. As the matrix is adjusted, users can further refine the design by adding or removing individual channels. This is done simply by clicking on specific cells within the matrix.

By clicking the rotate button, users can view the backside of the matrix, where they can manually assign channel indices to each electrode. This manual assignment provides an extra layer of control, allowing for meticulous configuration of the matrix. However, for users who prefer automation, there is also a button that automatically assigns channel indices, streamlining the process and ensuring consistent channel numbering.

Beyond configuring the shape and channel assignment, the Matrix Creator offers additional customization options. Users can adjust the size of the electrodes and the interelectrodic distance (IED), which is the spacing between electrodes. Once all adjustments are made, users can assign a unique name to the matrix, making it easy to identify and manage among multiple configurations.

On the left side of the Matrix Creator window, there is a list of previously saved matrices, each represented as a button. By clicking on one of these buttons, the corresponding matrix is loaded into the main editor window, allowing for quick access and further modifications if needed. This feature is particularly useful for users managing multiple matrices, as it simplifies the process of organizing, retrieving, and modifying existing configurations.

Once the matrix is fully customized and saved, it can be seamlessly integrated into the setup tool. The Matrix Creator is designed to work with the setup tool, ensuring that the custom matrices are accurately reflected in the device configuration.

Signal Processing

OTBiolab allows to process any kind of signals managed by the software itself. The processing tools are available in the offline window and all the processing are accessible under the Processing menu.
There are processing plugins that can be only applied to single tracks and other processing can be applied both to a single track or to a grouped track. Each processing can run on a selected range using the manual selection or can be applied to the entire selected track.
All the processing tools do not change the recorded signals but generate new tracks under the last existing track.

Absolute value

The plugin, when applied, simply rectify the signals.
Practically, it returns a new non-negative signal without regarding to its sign:
$Abs\_value = |s(t)|$
Where s(t) is the original signal.

Average Rectified Value - ARV

The estimation of ARV (AARV) is obtained for the epoch i using the following formula:

$$A_{ARv}(i) = \frac{1000}{N}\sum_{k=1}^{N}|x_k|[\mu v]$$

where:
'N' is the number of samples per epoch
'x_k' is the amplitude of the signal at the input of the amplifier in mV.

'1000' allows to obtain the result in µV.

Epoch Average

The estimation of Epoch Average is the mean value of each epoch. Graphically, it is represented by a point for each epoch.

Mean Square

The estimation of Mean Square is obtained using the following formula:

$$\text{Mean Square} = \frac{1}{N}\sum_{k=1}^{N}x_k^2\ [mv]$$

where:

'N' is the number of samples

'x_k' is the amplitude of the signal at the input of the amplifier in mV.

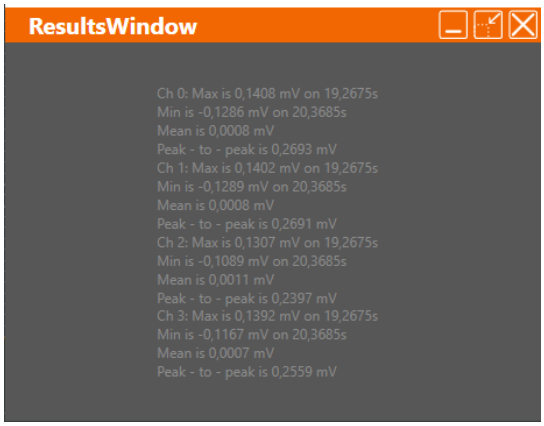
Max Min Mean

This processing computes max, min, mean and peak to peak value of the selected signal, and displays a window in which these values are. Also, you can click on "Set as scale" and the min and max value are set as track vertical range.

In particular it works comparing all the subsequent points of the selected track (memorized in a vector), finding the maximum or the minimum one. The mean is simply obtained as follow:

$$mean = \frac{\sum_{i=1}^{n} x_i}{length}$$

Where *n* is the number of elements in the vector.



Root Mean Square - RMS

The estimation of RMS (ARMS) is obtained for the epoch i using the following formula:

$$A_{RMS}(i) = 1000 \sqrt{\frac{1}{N} \sum_{k=1}^{N} x_k^2} \, [\mu V]$$

where:

- 'N' is the number of samples per epoch.
- 'x_k' is the amplitude of the signal at the input of the amplifier in mV.
- '1000' allow to obtain the result in µV.

Signal-Noise Ratio - SNR

The estimation of Signal to Noise Ratio is computed as the Ratio between the highest RMS value of the track and the lowest one.

Filtering

This processing tool can be applied both to single tracks and multiple tracks. When running it, a request of filter type and corner frequencies is displayed. It is possible to choose between different type of filters, whose expressions are represented below:

Low pass

$$H(s) = \frac{K\omega_0^2}{s^2 + \left(\frac{\omega_0}{Q}\right)s + \omega_0^2}$$

Band pass

$$H(s) = \frac{K\left(\frac{\omega_0}{Q}\right)s}{s^2 + \left(\frac{\omega_0}{Q}\right)s + \omega_0^2}$$

High pass

$$H(s) = \frac{Ks^2}{s^2 + \left(\frac{\omega_0}{Q}\right)s + \omega_0^2}$$

Stop band

$$H(s) = \frac{K(s^2 + \omega_0^2)}{s^2 + \left(\frac{\omega_0}{Q}\right)s + \omega_0^2}$$

Moreover, the corner frequencies are differently asked depending on the filter type selected. The signals generated by the filtering process are added as single or multiple tracksin the review window leaving unaltered the starting signals. All filter types are II order Butterworth filters.

Median Frequency - MDF

The estimation of MDF (fMDF) is obtained by comparing the total signal spectrum intensity divided by two with the cumulative intensity (i.e. all the intensity values for frequencies lower and including the focal intensity). The lowest frequency retrieving the cumulative intensity bigger than the half total intensity is fMDF.

Mean Frequency - MNF

The estimation of MNF (f_{MNF}) is obtained for the epoch i using the following formula:

$$f_{MNF}(i) = \frac{\sum_{k=0}^{n} I_k \cdot f_k}{\sum_{k=0}^{n} I_k}[Hz]$$

where:

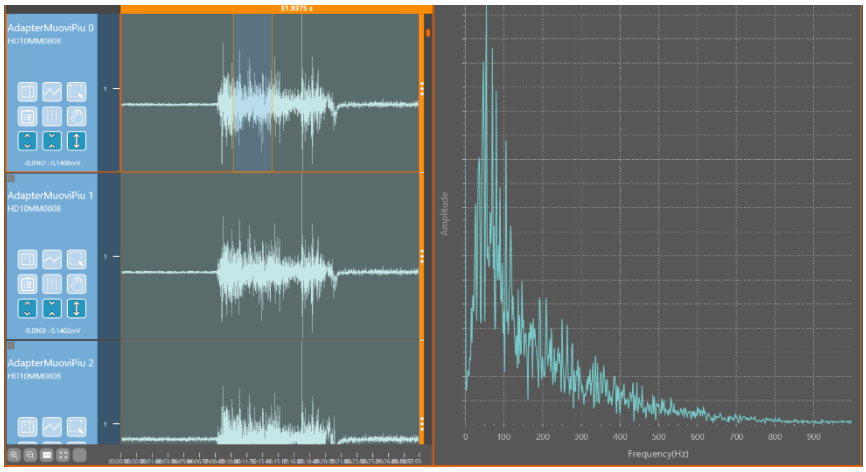
'n' is the number of frequency bins in the spectrum
'f_k' is the frequency of spectrum at bin k of n
'I_k' is the Intensity of spectrum at bin k of n

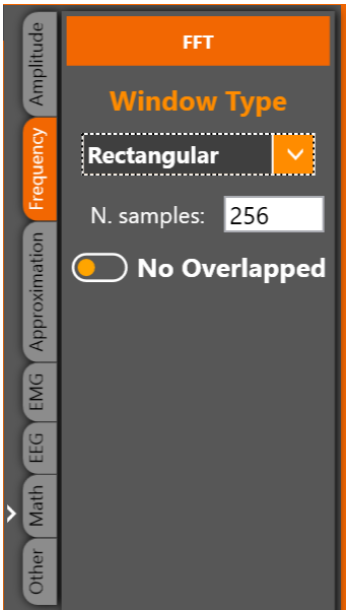
Fast Fourier Transform - FFT

The FFT can be estimated from single tracks on the desired number of epochs. After selecting the track, a window will appear for selecting the number of samples that will be used to obtain an averaged FFT to reduce the noise in the estimated power spectrum. A minimum of 256 samples for the averaged FFT is required. The maximum number of samples is equal to the length of the data. From this menu the user can select the window functions to apply to the time domain data for estimating the FFT (rectangular, hanning, hamming, blackman). Samples used to obtain the averaged FFT can be overlapped (50 % of overlap - Welch's method) or no overlapped (Barlett's method) selecting the options that appears on the bottom-right of the window menu.

After selecting these options, a new window will display the result of the algorithm. Details about the FFT algorithm can be found at http://sourceforge.net/projects/kissfft.

After the FFT has been estimated, keeping the left mouse button pressed a rectangular window can be drawn to indicate the area that the user wants to zoom in. When a right click on the FFT zoom out will be performed.

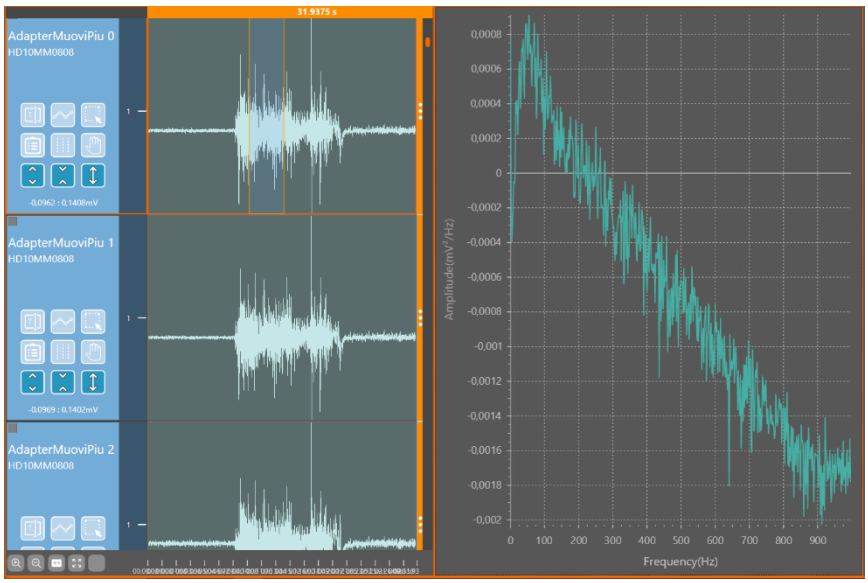




Power Spectral Density - PSD

The power spectral density (PSD) can be estimated from single or multiple tracks on the desired number of epochs. As in the FFT plugin, after the selection of the track, a window will appear in which you can choose the window type (rectangular, hanning, hamming and blackmann) and the number of samples, from a minimum of 256, that will be used to obtain an averaged FFT to reduce the noise (see FFT plugin for details), the window functions and the method (Welch or Bartlett's method).

After selecting options of this menu two plots will show the PSD of the signal in two different units: dB/Hz and mV2/Hz. As an example, a PSD signal is shown in logarithmic scale.



Average

The Average i calculated as:

$$Av = \frac{1}{lenght(x[i])}\sum_{i=1}^{n} x_i$$

Where:
x_i is a component of the vector *x[i]* of *n* elements (in fact the operation is integrated in a for cycle)

Coefficient of Variation - CoV

The coefficient of variation is calculated as:

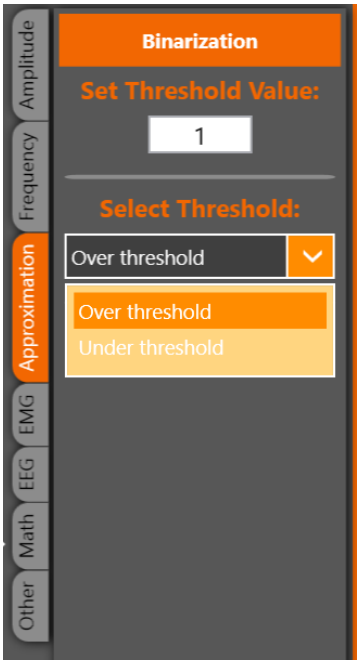
$$CV = \frac{\sigma}{\mu} \cdot 100$$

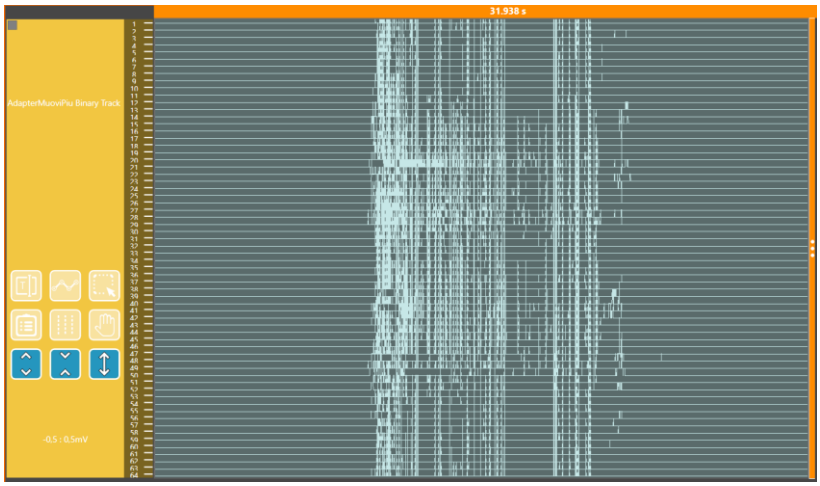
Where:
σ is the standard deviation
μ is the mean

Binarization

The processing Create Binary Track allows you to set a threshold: all the values above this threshold will be set to 1, the remaining value (which are value lesser than the threshold) are set to zero, or vice versa, depending on the ThresholdParameters you choose.

It could be useful for highlighting interesting parts of the signal or eliminating minor parts of the signal.





Linearization and Parabolic

The linear and parabolic approximations are two methods that use an affine function to approximate the original signal in a simplified way. They are used in the finite difference method for solving complicate equation, also in this case of EMG signal acquisition.

Standard Deviation

The Standard Deviation is calculated as:

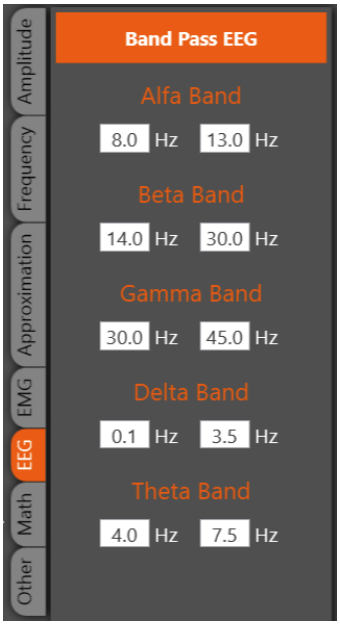
$$SD = \sqrt{\frac{\sum|x-\mu|^2}{N}}$$

Where
X is a generic element of the vector
μ is the mean
N it the length of the vector

It's a statistic measure of the amount of variation or dispersion of all the vector's values resulting from the acquisition.

Band Pass EEG

This processing will extract from each signal (each channel) the characteristic 5 bands of the EEG signals. Each of them has a specific frequency range, as you can see in the figure on the right, and it's related to a different functional-brain status.



In particular, the alfa band it's the oscillatory component ad it's the ruling frequency in the human EEG. Its power spectrum is represented by the following formula:

$$P(\Omega) = \frac{1}{2\pi} \lim_{T\to\infty} \left| \frac{1}{2T} \int_{-T}^{+T} x(t) e^{-j\Omega t} dt \right|^2 .$$

Calibration

This processing is used to multiply the data for a specific value and to remove the offset. Both these choices are optional, and the user can select one or both.

Convolution

This processing performs convolution between two signals in order to recognize the highest cross correlation between them and how the shape of one is modified by the other. Once it recognizes it, it will shift in time the second signal in order to align them.

$$(f * g)(t) := \int_{-\infty}^{\infty} f(\tau) g(t-\tau) d\tau$$

Where:
f(t) is the first signal
g(t) is the second signal
τ is a generic time constant

For doing this process it is necessary to select two tracks because of the convolution's definition, otherwise it will not work.

Derivative

Performs derivative of a signal. The first derivative of a signal is the rate of change of amplitude with the time, which is interpreted as the slope of the tangent to the signal at each point.

Envelope

Extract the envelope from a single track and normalize it with respect to the envelope maximum peak. The tool has been designed for EMG signals, the resulting signal is obtained by rectifying the desired signal and applying a II order Butterworth low pass filter with 5 Hz corner frequency to the rectified signal.

The frequency response of the II order Butterworth filter is defined as follows:

$$|G(\omega)| = \frac{1}{\sqrt{1+\omega^{2n}}}$$

Where:
ω is the angular frequency
n is the order, in this case n=2

Instead, the first order low-pass filter eliminates everything under the frequency of 5Hz, whose Laplace transform is:

$$H(s) = \frac{\omega_0}{(s+\omega_0)}$$

Integral

Performs the integral of a signal which is the continuous analog of a sum. A definite integral of a function can be represented as the signed area of the region bounded by its graph and the horizontal axis.

Sum

This processing tool can be applied to a multiple track or to, at least, a pair of single tracks. Simply it calculates, sample by sample, the sum between signals. When it is applied to a multiple track, it calculates the sum between all channels associated to the selected sensor and retrieve a new single track containing the sum. A typical application is the use with signals from an electrode array; the processing calculates the signals virtually obtained with multiple inter-electrodic distance.

Activate Instant Detection - AID

This processing tool estimates the muscle on–off timing, based on a physical model of muscle activation rather than on a phenomenological one. Muscle activity is recognized based on the presence of MUAPs in the surface EMG signal.

The surface EMG signal is the sum of the MUAP trains. A basic function can be assumed to roughly describe the shape of a generic MUAP detected on the skin surface and the train is indicated with **MUAPT(t)**.

The EMG signal **s(t)** can therefore be expressed as:

$$\boldsymbol{s(t) = \sum_j MUAPT_j(t) + n(t) = \sum_j \sum_i k_j \cdot f\left(\frac{t-\theta_{ij}}{\alpha_j}\right) + n(t)} \qquad \textbf{(1)}$$

where:
j indicates specific MU
k_j *is an amplitude factor*
θ_{ij} *are the occurrence times of the MUAPs of the MU*
α_j *is the scaling factor*
$n(t)$ *is the additive noise*

In the case $\boldsymbol{\alpha_j = \alpha_0}$, $\forall j$ and n(t) being white noise, one single event is repeating in the signal with a known shape and unknown amplitudes and times of occurrence. In this case, the matched filter (CWT) is the optimal filter for detecting the presence of a MUAP.

$$\boldsymbol{CWT(a,\tau) = \frac{1}{\sqrt{a}} \int_{-\infty}^{+\infty} s(t) \cdot w * \left(\frac{t-\tau}{a}\right) dt} \quad \textbf{(2)}$$

where:
s(t) is the signal to be analysed
w(t) is a prototype function (mother wavelet)
τ is a translation index
a is a scale parameter related to the frequency content

The CWT acts as a filter whose impulse response h(t) is equal to w(-t).

The event detection is applied to the filter (CWT) output and an amplitude threshold function η(t) is defined as $\boldsymbol{\eta(t) = \max\{CWT(a,t)\}}$, based on the noise level.

The amplitude M of the maximum of the test function η(t) within the interval (0 < t < T_{noise}), initial period in which the EMG activity is not present, is used to set the threshold value as:

$$\boldsymbol{M = max\{\eta(t)\}} \quad \textbf{(3)}$$

$$\boldsymbol{th = \alpha \cdot M} \quad \textbf{where } \alpha\textbf{>1} \quad \textbf{(4)}$$

The revealed vector is set at one for the time periods during which η(t) exceeds the threshold and zero, otherwise. If the SNR is high, the choice of the value $\boldsymbol{\alpha}$ is not critical, due to the great amplitude difference between EMG signal and noise. If the SNR is low, a low value of $\boldsymbol{\alpha}$ becomes a conservative choice.

After the detection process, events identified and separated by a temporal distance smaller than 125 ms are considered as belonging to the same contraction and merged. This value (125 ms) corresponds to a global muscular firing rate of eight pulses per second (pps), which is arbitrarily assumed as the lowest effective muscle activity. After the merging of activity bursts closer than 125 ms, the detected events shorter than 5 ms are attributed to either isolated MUAPs or noise related spikes and disregarded.

The algorithm is described in the paper: “A Fast and Reliable Technique for Muscle Activity Detection from Surface EMG Signals. A.Merlo, D. Farina & R. Merletti, 2003, IEEE Transaction on Biomedical Engineering, VOL. 50, NO. 3.”

When the Activation Instant Detection plug-in is selected, the window shown in figure 10.3 appears. The Initial noise section and the Alpha value correspond to the initial interval and the alpha parameter of the formulas (3) and (4). While the Minimum size of an inactivity should be greater than 125ms and the Minimum size of an activity should be between 5 - 40ms.

Find Zeroes

This plug-in produces a square waveform with a value equal to 1 where there are zeroes and a value equal to 0 where it has found signal. To identify zero value, must find a zero and compare it with the next

10 samples if they are all zero then score. On the raw signal it is difficult to identify 10 samples equal to zero while on the already processed signal it is more performing.

Normalization

The Normalization rescale all the values in a track given a specific max and min value.

$$Norm(x_i) = (x_{i-}x_{\max}) \,/\, (x_{max} - x_{min})$$

Where:
x_i is a component of the vector *x[i]* of *n* elements (in fact the operation is integrated in a for cycle)

Time Shift

This processing will shift the signal in time, and it is applicable only on a single track. When you run this processing, it will ask you how many samples you want to shift the signal. If the number is negative, it will move the beginning of the signal forward in time, delaying it. If it will be positive, it will remove the first part of the signal.

Decomposition (coming soon)

EEG Impedance Check

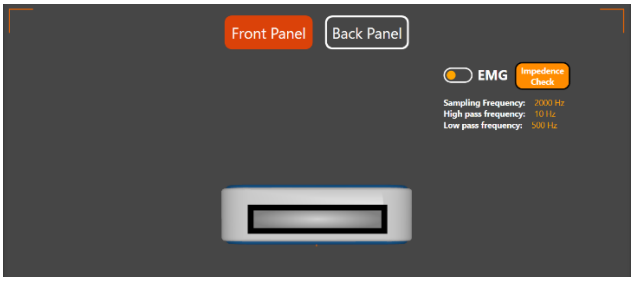
The EEG impedance check is a feature used to shows the electrode-skin impedance and let the user understand if the electrode contact is good enough to allow reliable recording. On the headset diagram the different electrodes positioned on the head are represented by a circle. For each one the electrode-skin impedance value is calculated using the "Lead-Off Detection" function present in the Sessantaquattro and Sessantaquattro+ ADC converter and a colour code is associated to this value which varies between red (not good) and green (low impedance).

Red: greater than 80 kohm

Orange: 25-80 kohm

Green: less than 25 kohm

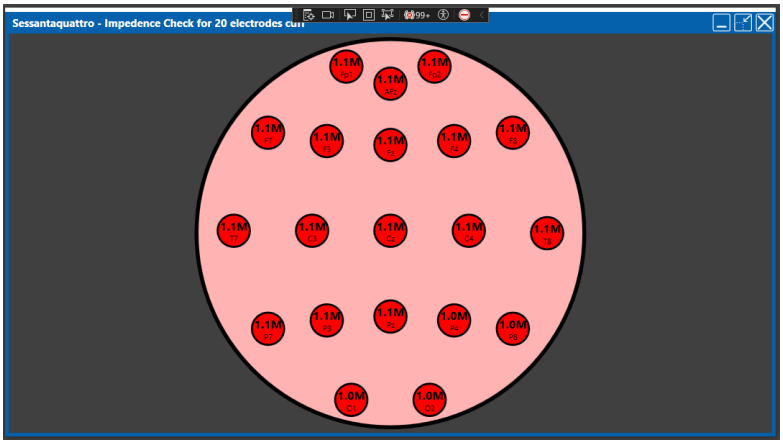
This feature can be used from the setup window, after selecting any device that allows to use it, an orange button appears.

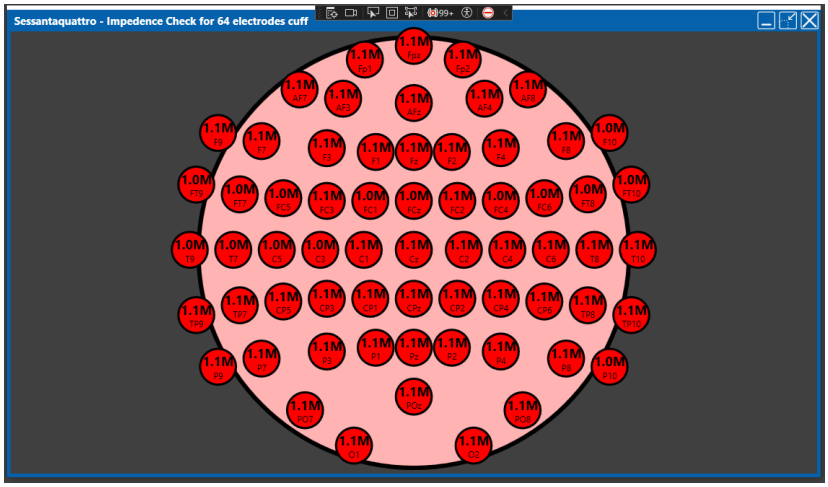


When it's clicked a new window appears and the visualization can be different depending on the adapter and sensor inserted.

1) Default visualization: show the pin out of the device itself and the impedance value for each channel.

2) EEG Cap visualization: show the impedance in a graphical representation of the EEG cap, based on the number of channels of the cap itself.





Troubleshooting