|  |
| --- |
|  |
| OpenViBE User Manual |
| Daniel Elstob |
| 14011862@hope.ac.uk |
|  |
|  |

|  |
| --- |
|  |

Contents

[1. Introduction 3](#_Toc441836395)

[2. Acquisition Server 4](#_Toc441836396)

[3. OpenViBE Designer 5](#_Toc441836397)

[4. Acquisition.xml 6](#_Toc441836398)

[5. CSP\_Training.xml 8](#_Toc441836399)

[6. LDA\_Classify.xml 10](#_Toc441836400)

[7. Analysis.xml 13](#_Toc441836401)

[8. RealTime\_v1.xml 18](#_Toc441836402)

[9. RealTime\_v2.xml 22](#_Toc441836403)

[Figure 1 Default Acquisition Server 4](#_Toc441836335)

[Figure 2 Emotiv EPOC Acquisition Server 4](#_Toc441836336)

[Figure 3 Driver Properties 4](#_Toc441836337)

[Figure 4 OpenViBE Designer 5](#_Toc441836338)

[Figure 5 Designer Controls 5](#_Toc441836339)

[Figure 6 Acquisition.xml 6](#_Toc441836340)

[Figure 7 Acquisition Client Settings 6](#_Toc441836341)

[Figure 8 Generic Stream Writer Settings 6](#_Toc441836342)

[Figure 9 Acquisition.xml GUI 7](#_Toc441836343)

[Figure 10 CSP\_Training.xml 8](#_Toc441836344)

[Figure 11 Generic Stream Reader Settings 8](#_Toc441836345)

[Figure 12 CSP Spatial Filter Trainer Settings 9](#_Toc441836346)

[Figure 13 CSP Spatial Filter Training Success Message 9](#_Toc441836347)

[Figure 14 LDA\_Classify.xml 10](#_Toc441836348)

[Figure 15 Generic Stream Reader Settings 11](#_Toc441836349)

[Figure 16 CSP Spatial Filter Settings 11](#_Toc441836350)

[Figure 17 Classification Trainer Settings 12](#_Toc441836351)

[Figure 18 Acquisition1.ov LDA Results 12](#_Toc441836352)

[Figure 19 Analysis.xml 13](#_Toc441836353)

[Figure 20 Generic Stream Reader Settings 14](#_Toc441836354)

[Figure 21 CSP Spatial Filter Settings 14](#_Toc441836355)

[Figure 22 Classifier Processor Settings 14](#_Toc441836356)

[Figure 23 Python Scripting 15](#_Toc441836357)

[Figure 24 Analysis GUI 15](#_Toc441836358)

[Figure 25 Direction of Motor Imagery Strength 16](#_Toc441836359)

[Figure 26 Overall Performance Metrics 16](#_Toc441836360)

[Figure 27 Opening COM1 using Python Script 16](#_Toc441836361)

[Figure 28 Python Scripting Box Extracting Stimulation Codes and Printing 16](#_Toc441836362)

[Figure 29 Arduino Serial Receiving the Characters 17](#_Toc441836363)

[Figure 30 Realtime\_v1.xml 18](#_Toc441836364)

[Figure 31 Acquisition Server Client 18](#_Toc441836365)

[Figure 32 Generic Stream Writer 19](#_Toc441836366)

[Figure 33 Graz Motor Imagery Settings 19](#_Toc441836367)

[Figure 34 CSP Spatial Filter Settings 20](#_Toc441836368)

[Figure 35 Classifier Processor Settings 20](#_Toc441836369)

[Figure 36 Python Scripting Settings 20](#_Toc441836370)

[Figure 37 Real-time One 21](#_Toc441836371)

[Figure 38 Opening COM1 using Python Script 21](#_Toc441836372)

[Figure 39 Python Scripting Box Extracting Stimulation Codes and Printing 21](#_Toc441836373)

[Figure 40 Arduino Serial Receiving the Characters 21](#_Toc441836374)

[Figure 41 Realtime\_v2.xml 22](#_Toc441836375)

[Figure 42 Acquisition Server Client 23](#_Toc441836376)

[Figure 43 Generic Stream Writer 23](#_Toc441836377)

[Figure 44 Graz Motor Imagery Settings 23](#_Toc441836378)

[Figure 45 CSP Spatial Filter Settings 24](#_Toc441836379)

[Figure 46 Classifier Processor Settings 24](#_Toc441836380)

[Figure 47 Python Scripting Settings 24](#_Toc441836381)

[Figure 48 Real-time Two 25](#_Toc441836382)

[Figure 49 Opening COM1 using Python Script 25](#_Toc441836383)

[Figure 50 Python Scripting Box Extracting Stimulation Codes and Printing 25](#_Toc441836384)

[Figure 51 Arduino Serial Receiving the Characters 26](#_Toc441836385)

# Introduction

For the purpose of quickly understanding how the OpenViBE software has been used in this thesis and how to allow for the user to test the scenarios or setup a live acquisition for themselves, this document has been created. For more in depth documentation of the software and its usability or for any support visit <http://openvibe.inria.fr/documentation-index/> that will provide overviews of functionality, designer features or box descriptions.

This manual will show a brief overview of the software and how to use the designed scenarios for the thesis. The next section will follow the following structure:

* How to get a live connection using the Acquisition server in 2.
* A brief description of the Designer and its controls in 3.
* Setup Acquisiton.xml
* Setup CSP\_Training.xml
* Setup LDA\_Classify.xml
* Setup Analysis.xml
* Setup Realtime\_v1.xml
* Setup Realtime\_v2.xml

# Acquisition Server

When loading up the Acquisition Server you should see Figure 1 for the default setup. Change the settings to match Figure 2.

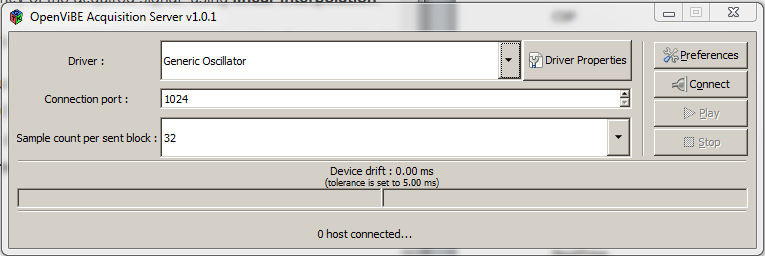


Figure Default Acquisition Server

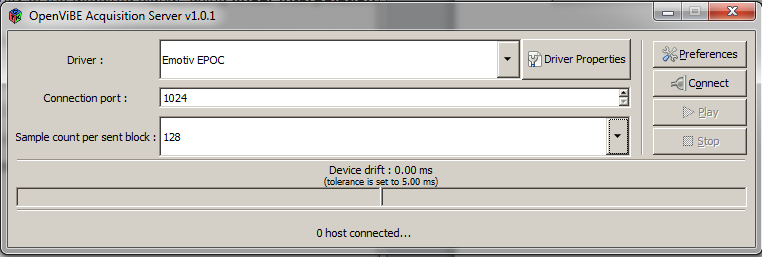


Figure 2 Emotiv EPOC Acquisition Server

Click the ‘Driver Properties’ box which will open Figure 3, and set the path to the Research SDK which is required in order to acquire the EEG signals from the headset.

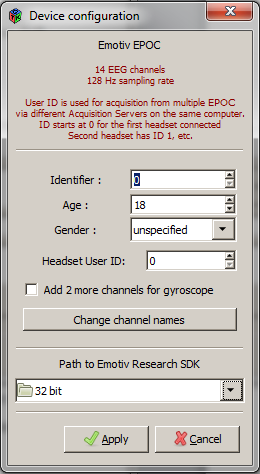


Figure Driver Properties

# OpenViBE Designer

When one of the XML files has been loaded into the OpenViBE Designer (Figure 4) and ensuring the file paths to specific boxes are correct the controls located at the top of the Designer can be used.

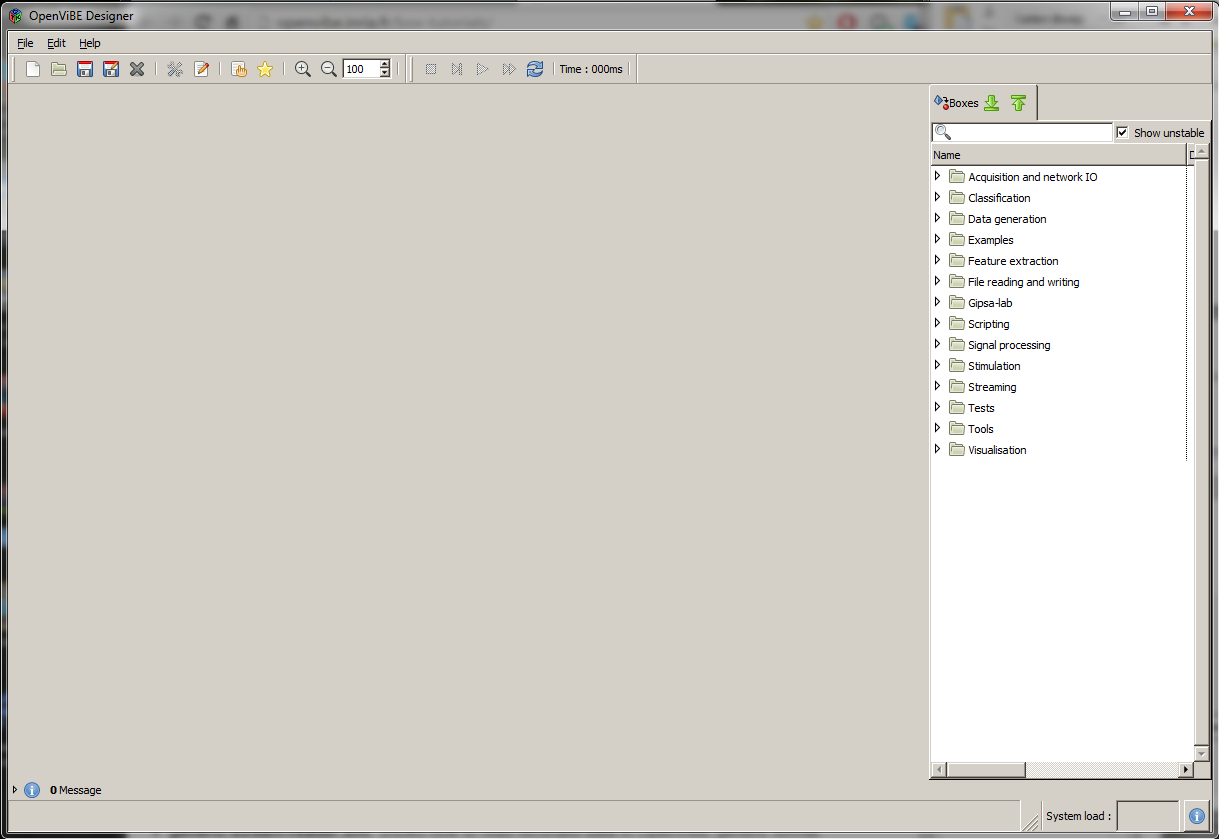


Figure OpenViBE Designer

Figure 5 shows a close up view of the controls. These are **Stop**, **Skip**, **Play**, **Fast-forward** and **Replay**.

* **Stop** will stop the currently playing scenario.
* **Skip** will go to the end of the currently playing scenario. Useful if the user want to see the end result of a scenario such as Analysis.xml.
* **Play** will play the currently playing scenario.
* **Fast-forward** will speed up the currently playing scenario and can resume normal speed by pressing **Play.**
* **Replay** will replay the currently playing scenario from the start.



Figure Designer Controls

# Acquisition.xml

After loading in the Acquisition.xml (Figure 6) certain boxes may need to be changed which can be done by double clicking the box and its setting should appear. By default some of these boxes should path to the directory the .xml file is saved to (Figure 7, 8) however if not simply change the path to their current directory.

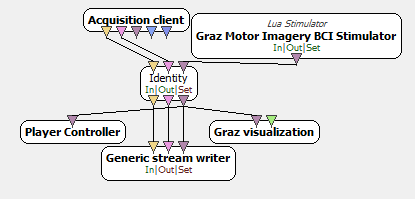


Figure Acquisition.xml

Figure 7 shows the acquisition sever port which is the connection port from section 2, if that value was change it must also be changed here.

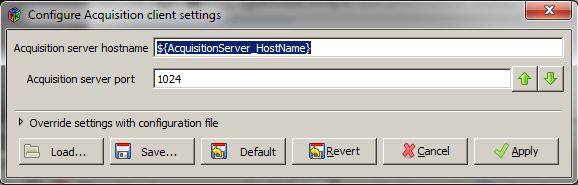


Figure Acquisition Client Settings

The generic stream writer (Figure 8) is the file directory to save the acquired EEG signals.

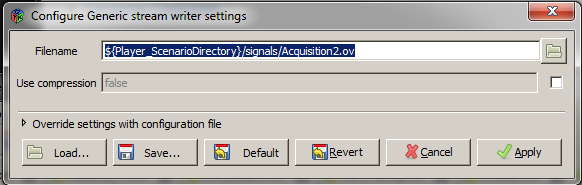


Figure Generic Stream Writer Settings

The Acquistision.xml will present the user will a GUI (Figure 9) that every four seconds will go through the process of a just the black background, followed by a green cross to indicate a new cue is going to appear and finally the cue in the form of a red arrow that points either left or right. When the cue appears the user must imagine movement of that particular direction until the next green cross appears. This ensures that there will be enough samples per cue for the duration of the acquisition. The more sample the system has for further classification the better the result for accuracy.

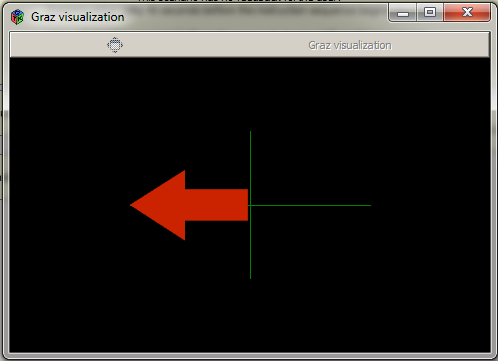


Figure Acquisition.xml GUI

# CSP\_Training.xml

After loading in the CSP\_Training.xml (Figure 10) certain boxes may need to be changed which can be done by double clicking the box and its setting should appear. By default some of these boxes should path to the directory the .xml file is saved to however if not simply change the path to their current directory.

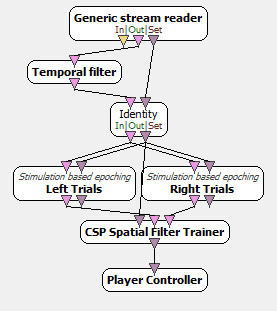


Figure CSP\_Training.xml

Figure 11 shows the generic stream reader which reads in data from the signals file directory used to save the acquired EEG signals. This will need to be changed to the current directory the folder is in.

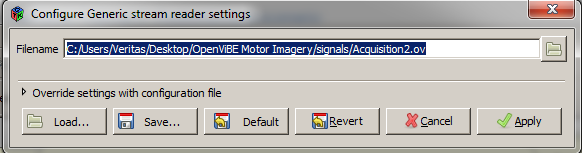


Figure Generic Stream Reader Settings

Figure 12 is the CSP Spatial Filter Trainer settings that showed the folder path which will save the trained CSP filter to.

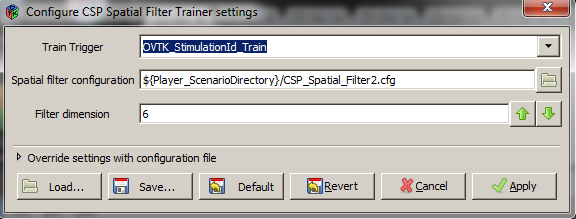


Figure CSP Spatial Filter Trainer Settings

This scenario can be sped up using the controls in in section 3. Once the scenario has finished the CSP spatial filter is saved to a configuration file that can be used in scenarios to overwrite the default. The scenario will also leave a message (Figure 13) telling the user that it has succeeded and extra information such as number of samples and the size of the sample set.

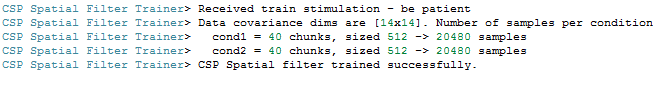


Figure CSP Spatial Filter Training Success Message

# LDA\_Classify.xml

The LDA\_Classify.xml scenario (Figure 14) is used to train the LDA classifier to detect and recognise the left and right arm movements. Similar to previous scenarios this will require setting changes.

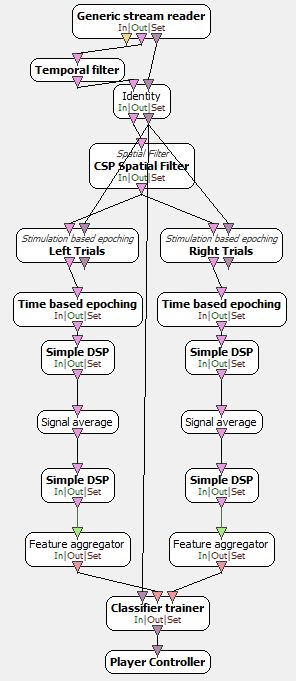


Figure LDA\_Classify.xml

Figure 15 shows the generic stream reader which reads in data from the signals file directory used to save the acquired EEG signals. This will need to be changed to the current directory the folder is in.

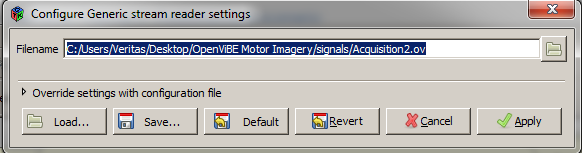


Figure Generic Stream Reader Settings

Figure 16 shows the CSP Spatial Filter Settings which will use the recently created CSP configuration file to overwrite the default settings. This will need to be changed to the current directory the folder is in.

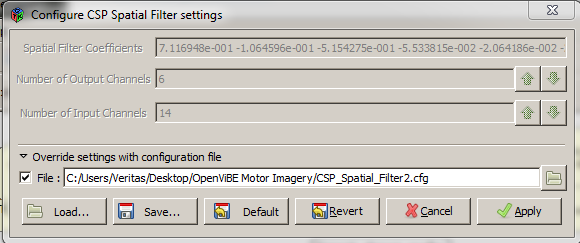


Figure CSP Spatial Filter Settings

Figure 17 shows the Classification Trainer Settings which will save a classification configuration file that will alter be used to overwrite the default settings in classifier processor settings. This will by default save to the current directory the folder is in.

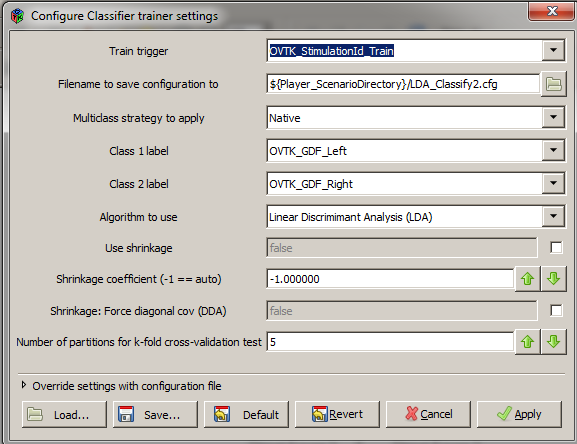


Figure Classification Trainer Settings

Once the training is completed a message in the command line and OpenViBE message box will give information about the cross validation and the accuracy test as shown in the Figure 18. This accuracy gives an estimate of how the LDA Classifier will perform but does not guarantee it for practical usage.

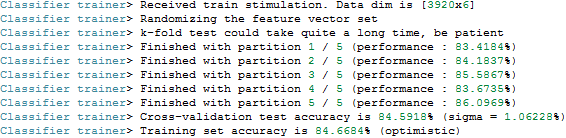


Figure Acquisition1.ov LDA Results

# Analysis.xml

For the purpose of examining and analysing the results from the previous scenario’s training and how well the BCI system will perform using performance metric of accuracy, Analysis.xml an offline visual replay of what has been recorded and trained in the previous three scenarios can be used (Figure 19). It will also send data to the COM ports for prosthetic control.

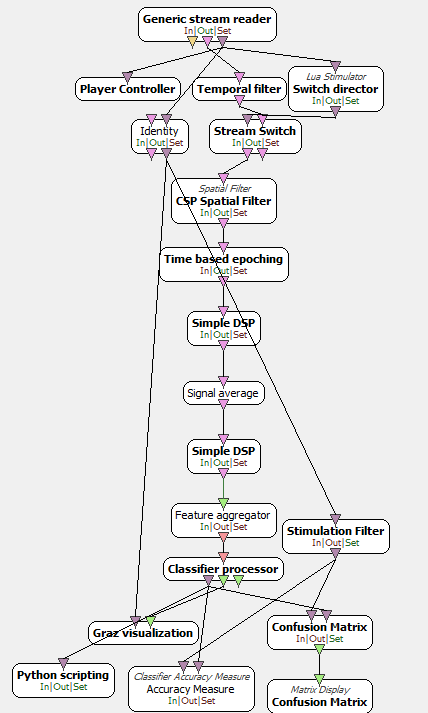


Figure Analysis.xml

Figure 20 shows the Generic Stream Reader Settings which reads in data from the signals file directory used to save the acquired EEG signals. This will need to be changed to the current directory the folder is in.

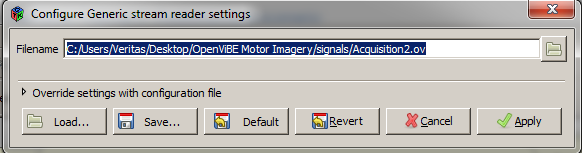


Figure Generic Stream Reader Settings

Figure 21 shows the CSP Spatial Filter Settings which will use the recently created CSP configuration file to overwrite the default settings. This will need to be changed to the current directory the folder is in.

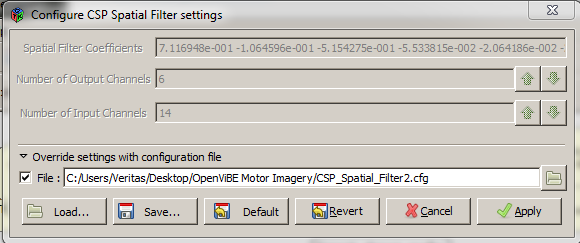


Figure CSP Spatial Filter Settings

Figure 22 shows the Classifier Processor Settings which will use the recently created CSP configuration file to overwrite the default settings. This will need to be changed to the current directory the folder is in.

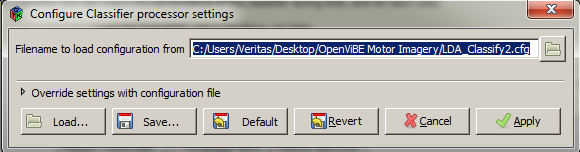


Figure Classifier Processor Settings

Figure 23 shows the Python Scripting Settings which will use the recently created CSP configuration file to overwrite the default settings. This will need to be changed to the current directory the folder is in.

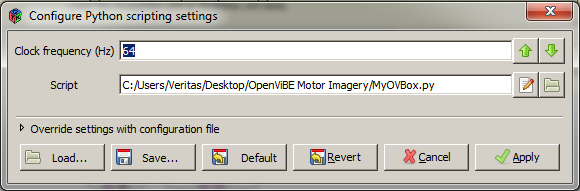


Figure Python Scripting

Using this scenario will present the user with Figure 24 that will replay the acquisistion.xml but will now show how the user performed (Figure 25). At the end of the replay Figure 26 will show the overall accuracy of the system.

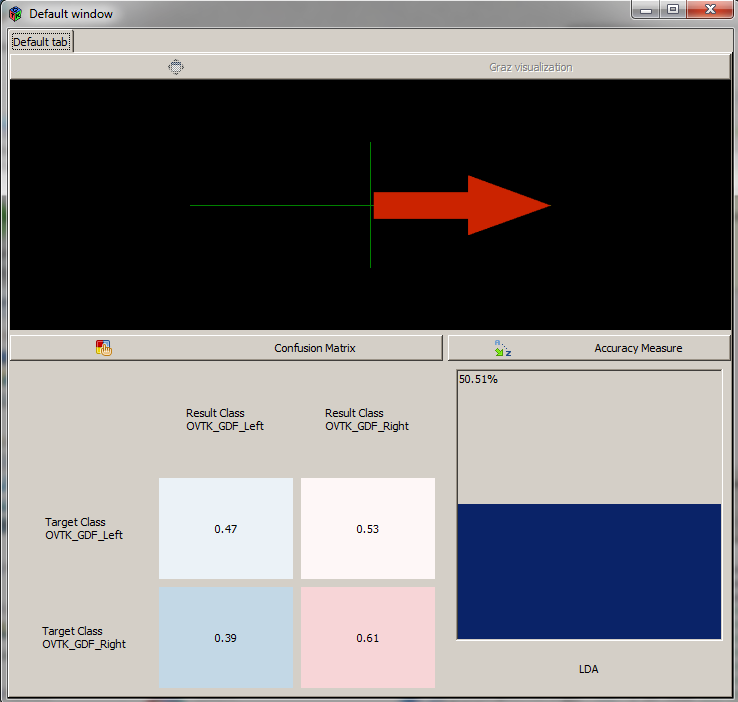


Figure 24 Analysis GUI

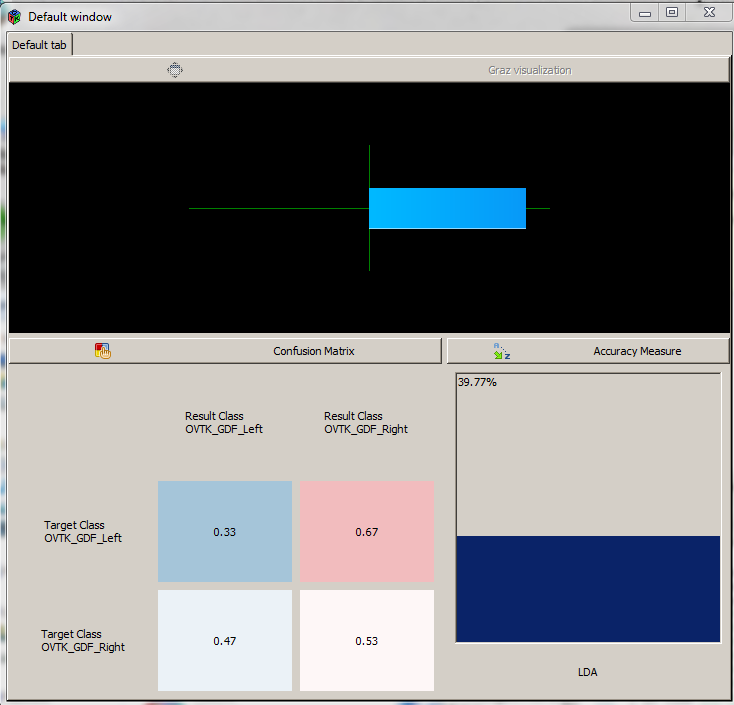


Figure 25 Direction of Motor Imagery Strength

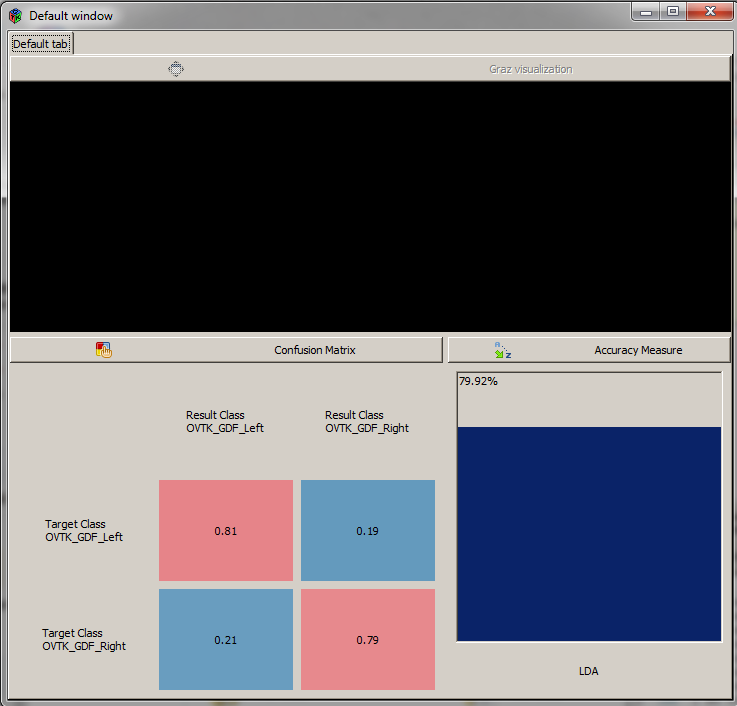


Figure 26 Overall Performance Metrics

The Python Scripting box will open COM1 (Figure 27) and send data (Figure 28) to any COM port on the same Baud rate which in this case will be COM6 at 9600 for the Arduino (Figure 29).



Figure 27 Opening COM1 using Python Script



Figure 28 Python Scripting Box Extracting Stimulation Codes and Printing

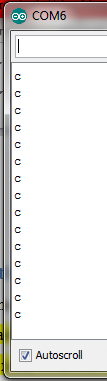


Figure 29 Arduino Serial Receiving the Characters

# RealTime\_v1.xml

Realtime\_v1.xml (Figure 30) will be a more advanced version of Acquisition.xml and will show feed back to the user in real-time. It will also send data to the COM ports for prosthetic control.

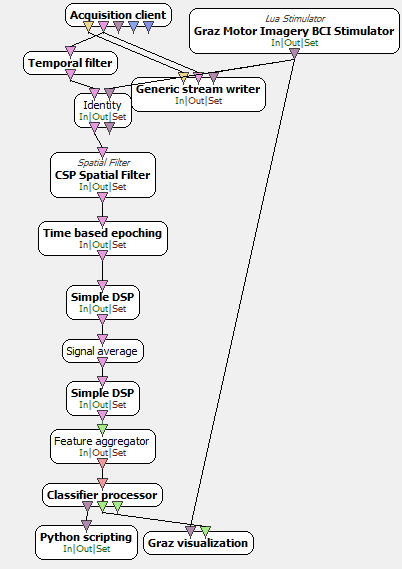


Figure Realtime\_v1.xml

Figure 31 will again use a live connection, shown is the acquisition sever port which is the connection port from section 2, if that value was change it must also be changed here.

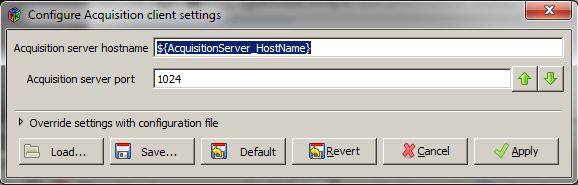


Figure 31 Acquisition Server Client

The generic stream writer (Figure 32) is the file directory to save the real-time acquired EEG signals.

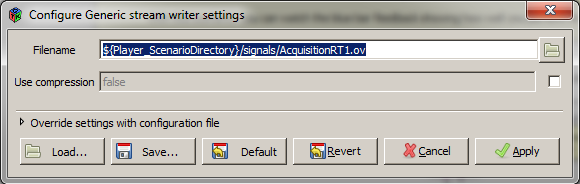


Figure 32 Generic Stream Writer

The Graz Stimulation setting (Figure 33) will need to be changed to show the user the cues hence it needs the file path to Graz\_Stimulator.lua .

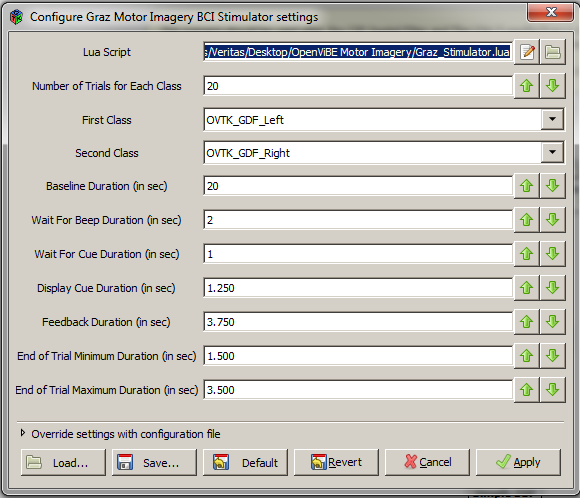


Figure Graz Motor Imagery Settings

Figure 34 shows the CSP Spatial Filter Settings which will use the recently created CSP configuration file to overwrite the default settings. This will need to be changed to the current directory the folder is in.

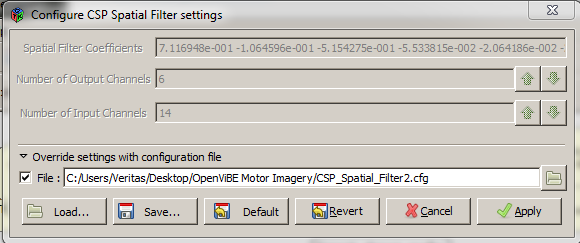


Figure CSP Spatial Filter Settings

Figure 35 shows the Classifier Processor Settings which will use the recently created CSP configuration file to overwrite the default settings. This will need to be changed to the current directory the folder is in.

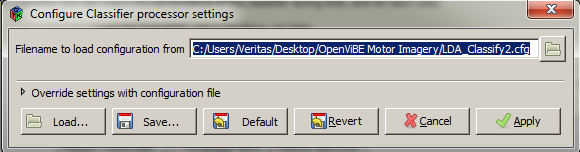


Figure Classifier Processor Settings

Figure 36 shows the Python Scripting Settings which will use the recently created CSP configuration file to overwrite the default settings. This will need to be changed to the current directory the folder is in.

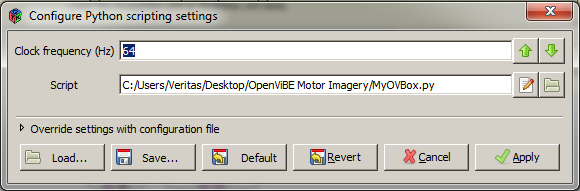


Figure Python Scripting Settings

When run Figure 37 will appear which is the same a Figure 9 but will show feedback in a blue horizontal line similar to the Analysis.xml

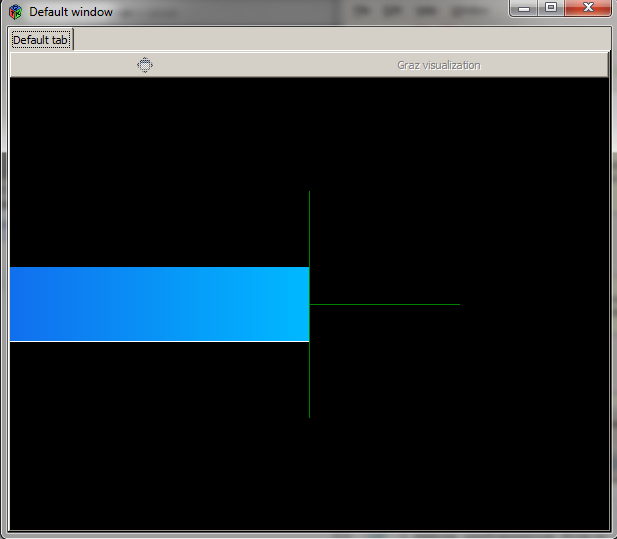


Figure 37 Real-time One

The Python Scripting box will open COM1 (Figure 38) and send data (Figure 39) to any COM port on the same Baud rate which in this case will be COM6 at 9600 for the Arduino (Figure 40).



Figure 38 Opening COM1 using Python Script



Figure 39 Python Scripting Box Extracting Stimulation Codes and Printing

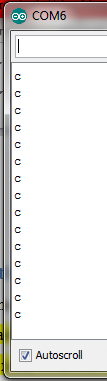


Figure 40 Arduino Serial Receiving the Characters

# RealTime\_v2.xml

Realtime\_v2.xml (Figure 41) will be a more advanced version of Acquisition.xml and Analysis.xml will show feed back to the user in real-time and the overall analysis accuracy. It will also send data to the COM ports for prosthetic control.

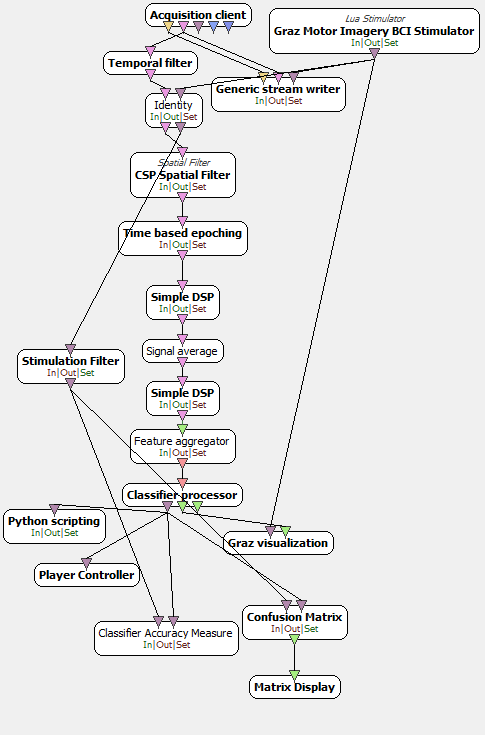


Figure Realtime\_v2.xml

Figure 42 will again use a live connection, shown is the acquisition sever port which is the connection port from section 2, if that value was change it must also be changed here.

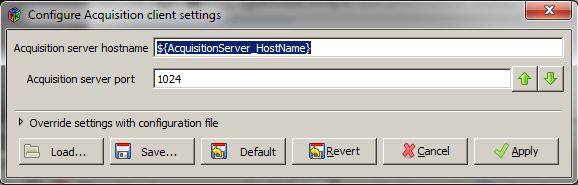


Figure 42 Acquisition Server Client

The generic stream writer (Figure 43) is the file directory to save the real-time acquired EEG signals.

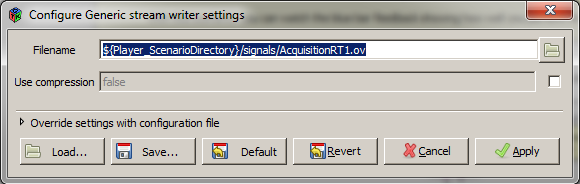


Figure 43 Generic Stream Writer

The Graz Stimulation setting (Figure 44) will need to be changed to show the user the cues hence it needs the file path to Graz\_Stimulator.lua .

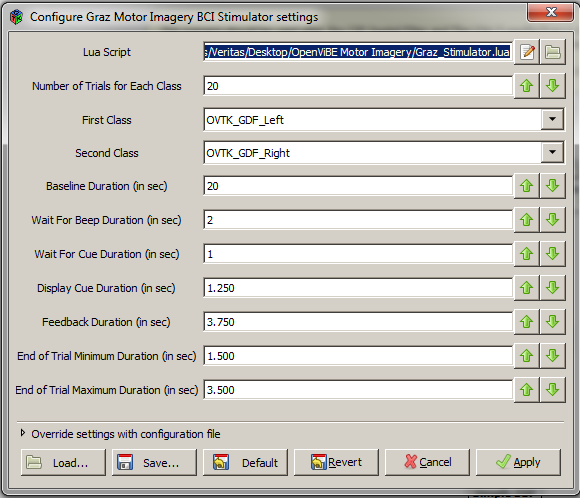


Figure 44 Graz Motor Imagery Settings

Figure 45 shows the CSP Spatial Filter Settings which will use the recently created CSP configuration file to overwrite the default settings. This will need to be changed to the current directory the folder is in.

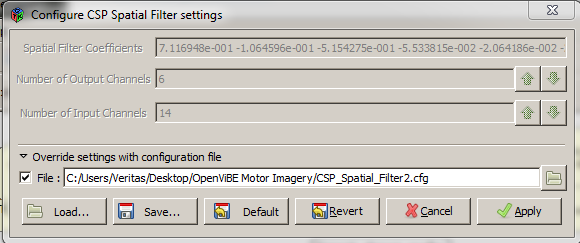


Figure 45 CSP Spatial Filter Settings

Figure 46 shows the Classifier Processor Settings which will use the recently created CSP configuration file to overwrite the default settings. This will need to be changed to the current directory the folder is in.

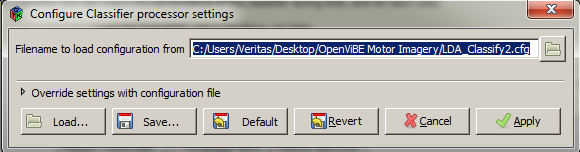


Figure 46 Classifier Processor Settings

Figure 47 shows the Python Scripting Settings which will use the recently created CSP configuration file to overwrite the default settings. This will need to be changed to the current directory the folder is in.

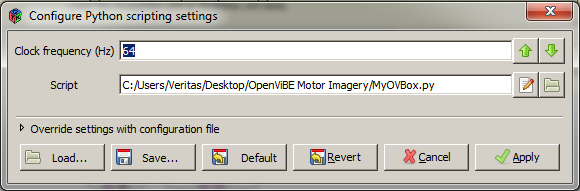


Figure 47 Python Scripting Settings

When run Figure 48 will appear resembling Analysis.xml however rather than taking and displaying data from the acquired file it will show the live feedback.

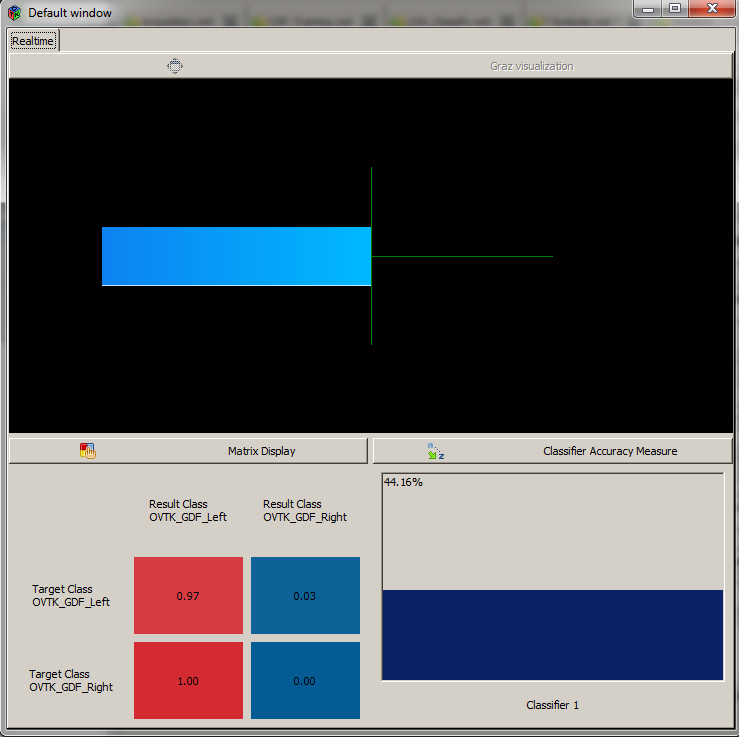


Figure Real-time Two

The Python Scripting box will open COM1 (Figure 49) and send data (Figure 50) to any COM port on the same Baud rate which in this case will be COM6 at 9600 for the Arduino (Figure 51).



Figure 49 Opening COM1 using Python Script



Figure 50 Python Scripting Box Extracting Stimulation Codes and Printing

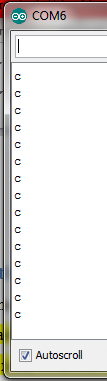


Figure 51 Arduino Serial Receiving the Characters