OpenBCI & OpenVIBE for P300 speller paradigm: a quick starting guide

VICTORIA PETERSON
Version 1.0, March 2017

Instituto de Investigación en Señales, Sistemas e Inteligencia Computacional, UNL, CONICET, FICH, Ruta Nac. 168, km 472.4, 3000, Santa Fe, Argentina

Instituto de Matemática Aplicada del Litoral IMAL, CONICET-UNL, Predio CCT-CONICET-Santa Fe, Ruta Nac. 168, km. 472, 3000, Santa Fe, Argentina

1 Introduction

This tutorial aims to provide to beginners the essential information for using OpenBCI together with OpenVIBE in P300 speller paradigm. I am not an expert on this matter. I just want to share with everyone my whole experience about make the OpenBCI system to work together with OpenVIBE in both Linux (Ubuntu 16.04, 64 bits) and Windows 7 (64 bits) Operating Systems (OS). It is expected that new users receive a quick and comprehensive (I hope) guide before start working with OpenBCI.

If you find this tutorial useful or do you have any suggestion, doubt or query, just email me to vpeterson@sinc.unl.edu.ar or victoriapeterson09@gmail.com

1.1 OpenBCI

OpenBCI is an open-source Brain Computer Interface platform that provides to anyone the necessary tools for acquiring their own biomedical signals. It is a versatile, compact and affordable bio-sensing systems that can be used to sample electrical brain activity (EEG), muscle activity (EMG), heart rate (EKG), and much more. OpenBCI offers a variety of acquisition boards (Ganglion Board Kit (4-channel), Cyton Biosensing Board Kit (8-channel), Cyton Biosensing Board & Daisy (16-channel) and R&D Kit (16-channel): Cyton, Daisy & Accessories) with wireless communication (based on the RFDuino radio module) to any computer, mobile device or tablet compatible with Bluetooth Low Energy (BLE).

Two type of 3D-printable EEG headsets (UltraCortex Mark III & UltraCortex Mark IV) had been developed by the OpenBCI people to be used with any of the boards to get research-grade EEG recordings. The headsets are provided with dry electrodes in predefined location (8 or 16).

The OpenBCI's default software tool is the OpenBCI GUI. It is possible to visualize, record, and stream data from any of the OpenBCI boards. It can be launched as a standalone application or launched from Processing (a Java-based programming language).

For more information about OpenBCI please visit `http://openbci.com/`.

1.2 OpenVIBE

OpenVIBE is a free and open source software that permits to design, test and use brain-computer interfaces for both real-time and off-line analysis of the EEG data. It has many capabilities from signal processing and machine learning framework. It does not require of any previous programming knowledge from the user, since it is "box-algorithm" designed. OpenVIBE contains two applications: System Acquisition Server (SAS) and Scenario Designer (SD). The last version of OpenVIBE allows OpenBCI hardware as a driver. The SD is provided with different example scenarios from different BCI paradigms that can be used as a first sight.

Please, if you are new in OpenVIBE go to http://openvibe.inria.fr/start/ and take a view.

1.3 What do you need

In order to acquire your own brain waves, you would need:

✓ a PC with either Linux or Windows,
✓ MATLAB® software appropriately installed,
✓ an OpenBCI board (either of them) with its USB dongle,
✓ a battery pack (4 AA batteries),
✓ an OpenBCI 3D headset or any standard electrode-cap or just the electrodes,
✓ conductive pasta (if you are using cup-electrodes),
✓ cotton and alcohol.

2 Get OpenBCI software running on your OS

2.1 On Linux (tested on Ubuntu 16.04, 64 bits)

To get OpenBCI GUI running on Linux, follow the following steps [1]:

1. Download the .zip file from http://openbci.com/index.php/downloads and extract it into your home folder.

2. Download the latest version of the jSSC library from https://code.google.com/archive/p/java-simple-serial-connector/downloads

3. Install a Java Runtime Environment — a program that runs programs written in Java — if you do not already have one. You can do this from the software center if you are on Ubuntu, or in the terminal:

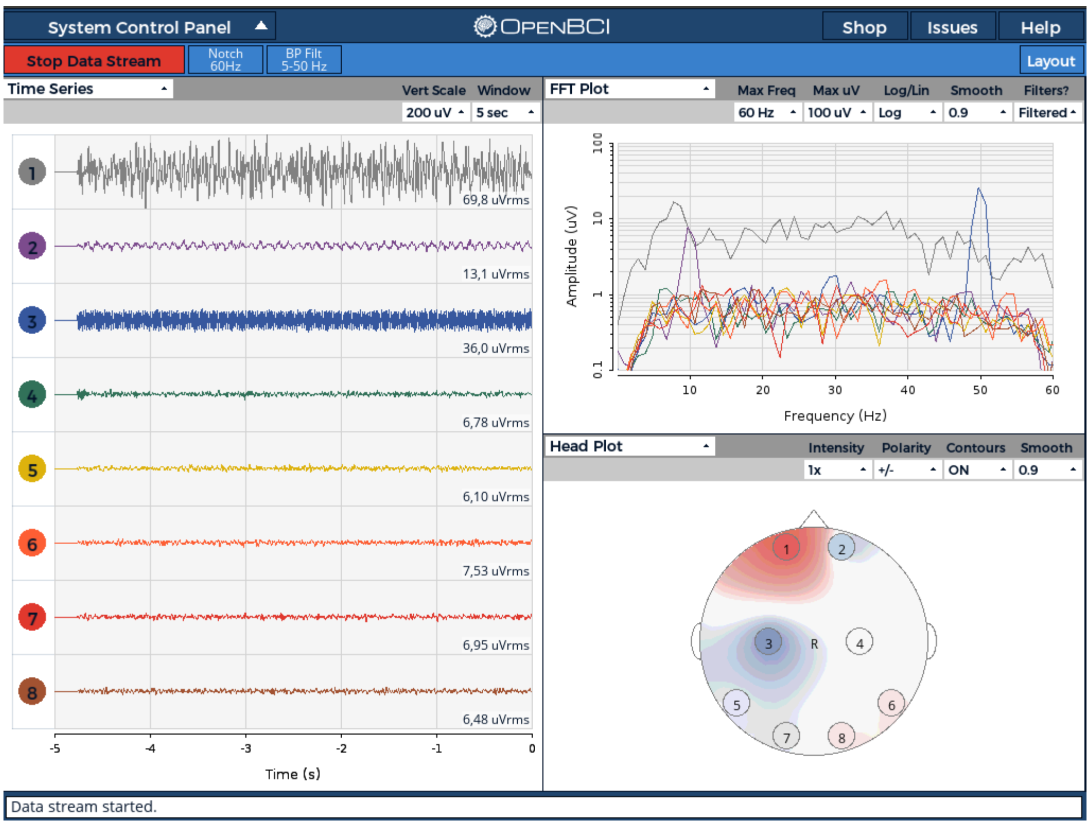
   ```
   sudo apt-get install default-jre
   ```

4. Open the OpenBCI app folder and go to the `lib` folder. Replace the file `jssc.jar` (this is the out-of-date library) with the file `jssc.jar` provided in the folder `jSSC-2.x.x-Release` that you just downloaded and extracted.

5. Start the OpenBCI application with sufficient privileges to access the serial port. Assuming that you have extracted the OpenBCI software into your home folder, this can be done with the command: `cd ~/application.linux64 && sudo bash OpenBCI_GUI` or `cd ~/OpenBCI_GUI_200_LINUX64 && sudo bash OpenBCI_GUI` depending on the version you have downloaded.

6. Select SYNTHETIC and press the START SYSTEM button. You should now see some brain waves like in Figure 1.

[1] extracted from "Getting Started with OpenBCI" http://www.autodidacts.io/getting-started-with-openbci-a-tutorial-on-testing-troubleshooting-and-recording-ekg/



Vert Scale: 200 uV; Window: 5 sec

Max Freq: 60 Hz; Max uV: 100 uV; Log/Lin: Log; Smooth: 0.9; Filters?: Filtered

Intensity: 1x; Polarity: +/-; Contours: ON; Smooth: 0.9

```
Data stream started.
```

Figure 1: OpenBCI GUI with synthetics data stream.

2.2 On Windows (tested on Windows 7, 64 bits)

1. Upgrade the graphics chip drivers for your machine.
2. Follow the instructions for downloading/installing processing from https://processing.org/download/?processing.
3. Copy and paste the 'controlP5' and 'gwoptics' libraries from `OpenBCI_Processing-master/libraries` into your `Documents/Processing/libraries directory` (if `/libraries` doesn't exist, then create it)
4. Copy and paste `OpenBCI_GUI` from `OpenBCI_Processing-master` into your `Documents/Processing` directory
5. You should now we able to start running the OpenBCI GUI by opening any of the .pde files in the folder `/OpenBCI_GUI`.

3 Get familiar with OpenBCI

The information given in this section is based upon the instructions given in http://docs.openbci.com/Tutorials/01-Cyton_Getting%20Started_Guide.

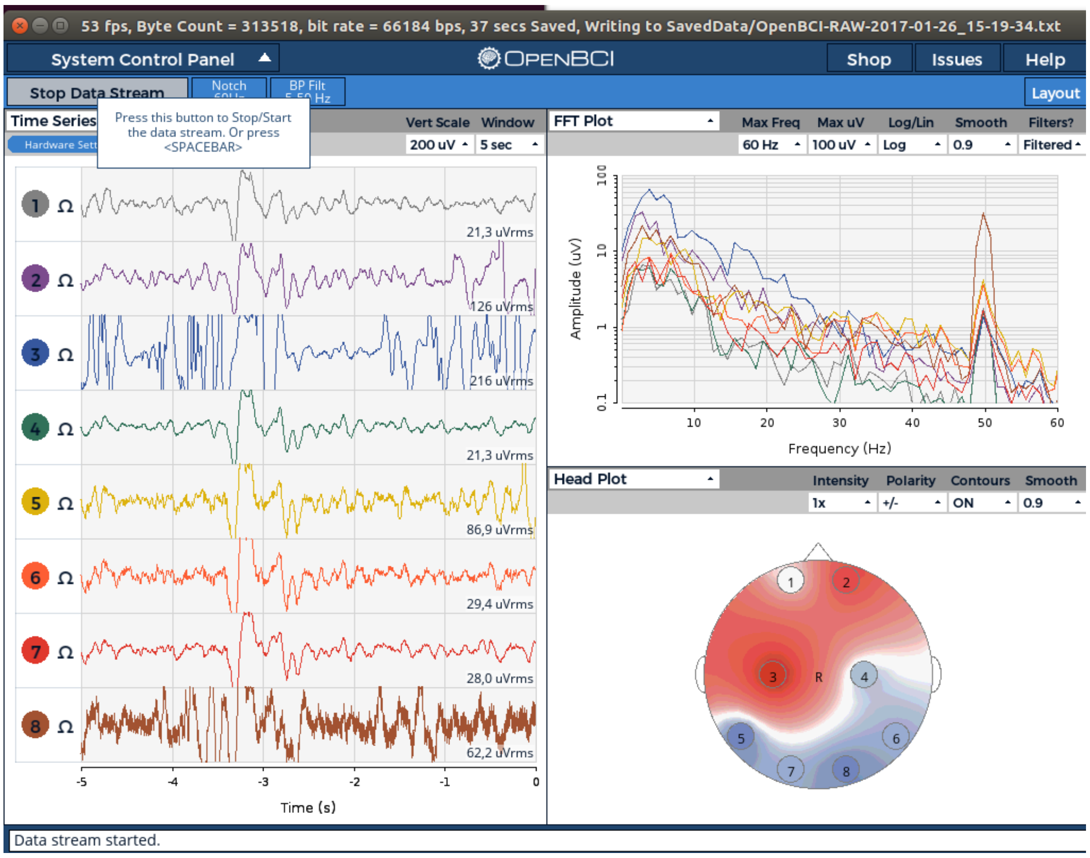
3.1 Hardware

- Make sure your FTDI drivers are installed and up-to-date. If you need to downloaded the drivers go to http://www.ftdichip.com/Drivers/VCP.htm. If you are using Linux the drivers are most surely already installed.

- Plug in your OpenBCI USB Dongle. A blue LED should light up.
- Plug in your 6V AA battery pack to the OpenBCI board. If you are not using the battery pack provided with the OpenBCI, *Be careful, never supply voltages over the specify ranges appearing in back of the board.*
- Switch the board to PC setting. A blue LED should light up.

3.2 Software

- Lunch the OpenBCI GUI.
- Select LIVE (from Cython or Ganglion)
- Select the USB Dongle port. In Linux they appear as `/dev/tty*` (I used /dev/tty/USB0) and in Windows they appear as `COM#`
- Press STAR SYSTEM (the communication should be established in about 5 seconds)
- Press START DATA STREAM. You should see same waves on the channel plots section. Since no electrodes are yet connected you may see the word RAILED on the right hand size. To check that the connection is well established, gently rub your fingers over the pins. You should be able to see something as shown in Figure 2.



```
53 fps, Byte Count = 313518, bit rate = 66184 bps, 37 secs Saved, Writing to SavedData/OpenBCI-RAW-2017-01-26_15-19-34.txt
```

Press this button to Stop/Start the data stream. Or press <SPACEBAR>

Vert Scale: 200 uV; Window: 5 sec

Max Freq: 60 Hz; Max uV: 100 uV; Log/Lin: Log; Smooth: 0.9; Filters?: Filtered

Intensity: 1x; Polarity: +/-; Contours: ON; Smooth: 0.9

```
Data stream started.
```

Figure 2: Display obtained by finger rubbing the board's pins.

TIP 1: Cleaning the electrodes

It is highly recommended that you clean up the electrodes after each session since the conductive paste may damage them. To do that, sink the electrodes in hot (but not boiling) water for about three seconds three or four times. This process melts away and leaves the electrodes clean. Don't leave the electrodes in water for a long time, since doing so may damage the wire connection to the head. After that, wipe them off with a soft tissue or paper towel. Finally, you should "disinfect" the electrodes by wiping them with an alcohol impregnated cotton ball.

4 Get OpenVIBE running on your OS

Depending on which OS do you have, please download the OpenVIBE Software from `http://openvibe.inria.fr/downloads/`. The following installation steps can also be found at `http://openvibe.inria.fr/build-instructions/`.

4.1 On Linux

1. Unzip the folder.
2. Install the third-party dependencies by running the provided `linux-install_dependencies` script inside the `scripts/` directory:

   `cd ~/openvibe-1.3.0-src/scripts && sudo bash linux-install_dependencies`.

   This may take some time. After the dependency installation, OpenVIBE is ready to be built.
3. Run `linux-build` script inside the `scripts` directory

   `cd ~/openvibe-1.3.0-src/scripts && sudo bash linux-build`.

   At this point a folder called `dist` will appear on your *openvibe* directory.
4. `openvibe-[application-name].sh` will launch the application you want:

   Acquisition server

   `cd ~/openvibe-1.3.0-src/dist && sudo bash openvibe-acquisition-server.sh`

   Designer

   `cd ~/openvibe-1.3.0-src/dist && sudo bash openvibe-designer.sh`.

4.2 On Windows

1. Execute the openvibe setup file that you have just downloaded.
2. Follow the installation steps
3. Once the installation is finished look for openvibe on the Windows Search Tool. You are going to find the following two programs: openvibe acquisition server and openvibe designer.

5 Electrode placement and OpenBCI connection

OpenBCI assumes the N position (the one closer to the board). By following the numbers and the colors of the channels in OpenBCI GUI it is:

Channel	Board name	Position	Color
1	N1P	Fp1	grey
2	N2P	Fp2	violet
3	N3P	C3	blue
4	N4P	C4	green
5	N5P	P7	yellow
6	N6P	P8	orange
7	N7P	O1	red
8	N8P	O2	brown

If you are using the Daisy module (i.e. 16 channels, 125 sps) the others eight channels follow the same order but are indexed from 9 to 16. This information is described in the Figure 3.

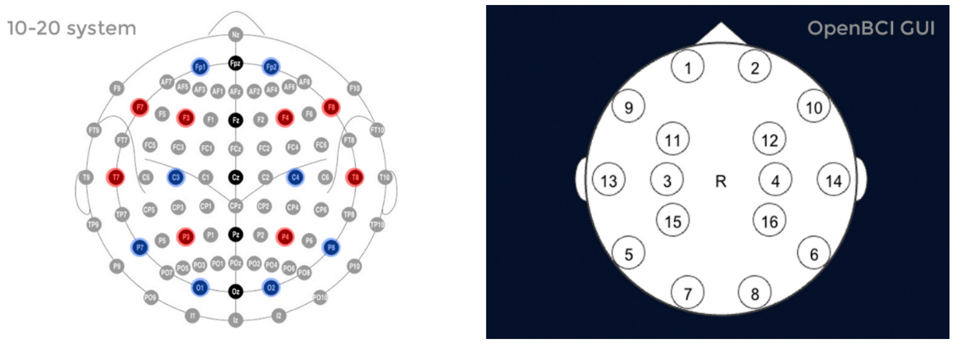


Figure 3: Electrode placement and ID number for 16 channels

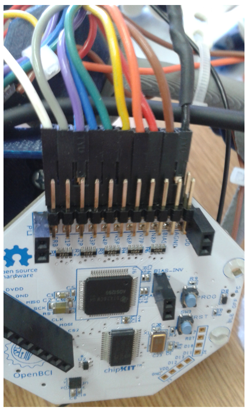


Figure 4: Example of electrode-board connection.

Figure 4 shows how the wires are ordered and plugged in to the board. The white electrode/wire is generally connect to the SRB2 pin (the bottom SRB pin). The SRB2 pin is the default "reference pin" for your OpenBCI input channels. The black electrode/wire is connect into the bottom BIAS pin. The BIAS pin is similar to the ground pin of common EEG systems, but it uses destructive interference waveform techniques to eliminate the "common mode noise" of all of the active channels.

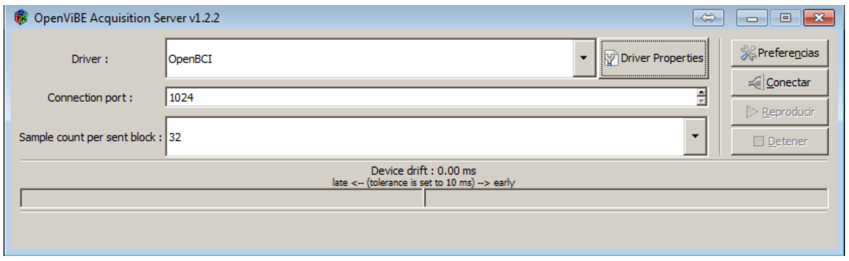
For more information about this section please go to `http://docs.openbci.com/headware/01-Ultracortex-Mark-III-Nova-Revised`.

6 P300 speller with OpenVIBE and OpenBCI

The last version of OpenVIBE Signal Acquisition Server (SAS) admits the OpenBCI driver as it is described in `http://openvibe.inria.fr/drivers-openbci/`. Thus, in the following subsection the main configurations for using OpenVIBE SAS with OpenBCI driver are described.

6.1 OpenVIBE Acquisition Server Configuration

1. Start the SAS.
2. Select OpenBCI in `Driver` option.



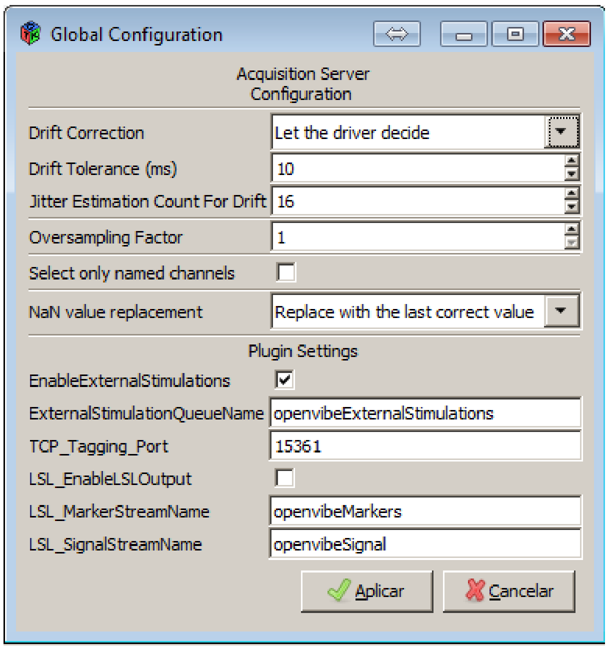
OpenViBE Acquisition Server v1.2.2

Driver :	OpenBCI
Connection port :	1024
Sample count per sent block :	32

Device drift : 0.00 ms
late <-- (tolerance is set to 10 ms) --> early

Figure 5

3. Click on `Preferences` to change the `drift tolerance` to 10ms and make sure that `EnableExternalStimulation` is *enable*.



Global Configuration

Acquisition Server Configuration

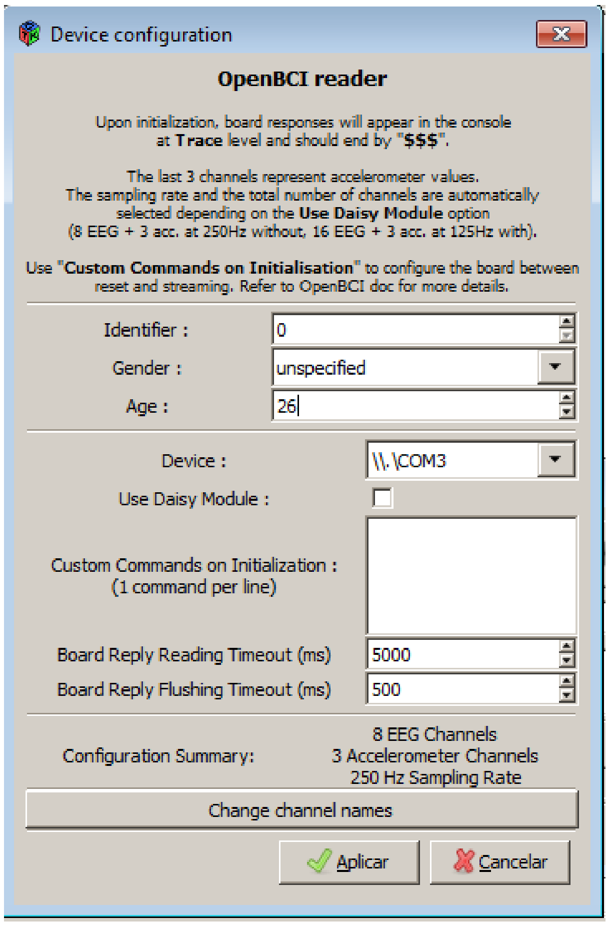
Drift Correction	Let the driver decide
Drift Tolerance (ms)	10
Jitter Estimation Count For Drift	16
Oversampling Factor	1
Select only named channels	☐
NaN value replacement	Replace with the last correct value

Plugin Settings

EnableExternalStimulations	✓
ExternalStimulationQueueName	openvibeExternalStimulations
TCP_Tagging_Port	15361
LSL_EnableLSLOutput	☐
LSL_MarkerStreamName	openvibeMarkers
LSL_SignalStreamName	openvibeSignal

Figure 6

4. Click on `Driver Properties` and set the `port number` in which your OBCI board is connected to. If you are using the Daisy module click this option. You can also change the channel names and save the configuration that you are setting for further uses.



Device configuration

OpenBCI reader

Upon initialization, board responses will appear in the console at **Trace** level and should end by "**$$$**".

The last 3 channels represent accelerometer values.
The sampling rate and the total number of channels are automatically selected depending on the **Use Daisy Module** option
(8 EEG + 3 acc. at 250Hz without, 16 EEG + 3 acc. at 125Hz with).

Use "**Custom Commands on Initialisation**" to configure the board between reset and streaming. Refer to OpenBCI doc for more details.

Identifier :	0
Gender :	unspecified
Age :	26
Device :	\\.\COM3
Use Daisy Module :	☐
Custom Commands on Initialization : (1 command per line)	
Board Reply Reading Timeout (ms)	5000
Board Reply Flushing Timeout (ms)	500
Configuration Summary:	8 EEG Channels 3 Accelerometer Channels 250 Hz Sampling Rate

Figure 7

TIP 2: Proper communication between SAS and Designer

For a proper communication between the SAS and the Designer, once you have selected the driver (our case OpenBCI) press CONNECT. If everything is OK you will see **CONNECTED!**. To send the data to the Designer press **PLAY** and go to the Designer window and press the **PLAY button** to run the scenario you want. If everything is OK in the SAS window you will see how many *host* are connected (e.g. one host connected) and in the Designer window the desired output (e.g. EEG visualization) will be hold on. For any drawback in communication go to `http://openvibe.inria.fr/acquisition-server/`.

IT IS REALLY IMPORTANT TO FIRST BEGIN THE DATA STREAM AND THEN RUN THE SCENARIO IN OPENVIBE

6.2 OpenVIBE Designer

OpenVIBE provides a number of scenarios that can be used as a first try. The P300 speller example located in `~/openvibe-x.x.x-src/dist/share/openvibe/scenarios/bci-examples/p300-speller` presents five scenarios:

- **Scenario 0:** Signal Monitoring
- **Scenario 1:** Acquisition
- **Scenario 2:** Train Classifier
- **Scenario 3:** Online
- **Scenario 4:** Replay

You can use any of these scenarios just in the way the are, or you can change some configurations by double clicking on the corresponding boxes. The configuration parameters can be saved to be used in future by clicking in the *save* button on the configuration windows.

It is highly recommended to check the quality of your signals before starting to acquire them. So, run the **Scenario 0** and if everything is OK (i.e. you are able to see *eye blinks, jaw clenching* and *alpha waves*) go to **Scenario 1**.

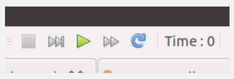
In the **Scenario 1** the acquired signal are save in ov extension (OpenVIBE file). The file will be saved in the same folder were the scenario is being run. If you want to change this, double click the *Generic Stream Writer* box and browse your folder. At the end of the name-file in the configuration window you will see the following: `[$core{date}-$core{time}]`. This "code" is used to save the file with the exact date and time in which the signals were recorded. If you do not care about this kind of information, just throw away this line!

These calibration signals can then be used for training purposed. A simple training classifier is implemented in **Scenario 2**. Run this scenario and in the end you will see how accurate are your brain waves !

Once you classifier is trained, you can now go to **Scenario 3** and run the P300 speller in on-line session. The latest scenario, **Scenario 4** look for a quickly replay of the on-line session.

TIP 3: Fast Training

The training procedure can take a long time. For a faster training procedure run the scenario with the "fast running button" located on the right side of play button.



7 Working with OV files in Matlab

It is possible to convert you *ov* files to *mat* files in order to use them for off-line analysis in Matlab. OpenVIBE provides a toolbox which can be download from `http://openvibe.inria.fr/converting-ov-files-to-matlab/`. This toolbox contains a main function called `convert_ov2mat`. By using this function you should be able to convert any .ov file to a .mat file, which contains the following variables on it:

- channelsNames: a cell array of size $1 \times C$, where C is the number of electrodes used in acquisition. This variable contains the name of the channels (the ones that have been load in the SAS),
- samples: a $N \times C$ matrix containing the recorded EEG data,
- sampleTime: a $N \times 1$ vector containing the time information,
- sampleFreq: an *int* indicating the sample frequency used during acquisition. Note that sampleFreq=1/(sampleTime(2)-sampleTime(1)),
- stims: a $M \times 3$ matrix containing the stimulation information.

It is highly probably that you may want to extract EEG segment (named trials or epochs) from the sample data. In order to do this, it is required to know the exact moment in which a Target or Non-Target event has occurred. The second column of the *stims* matrix contains the stimulation codes that OpenVIBE uses to label the different events that occur during the experiment. In `http://openvibe.inria.fr/stimulation-codes/` you can found the "meaning" of any OpenVIBE stimulation code. In Table 1 the most important stimulation codes are shown. With this information and with some help from INRIA people (see `http://openvibe.inria.fr/forum/` for any further doubt about OpenVIBE), a Matlab script for extracting the EEG trials (see TIP 4) was implemented. You can found this function in the code-folder named `Extract_ov2mat_Trials`.

Now, you are able to acquire your own EEG signals and work with them!.

TIP 4: Extract EEG trials

```
target_ind=stims(:,2)==33285;
nontarget_ind=stims(:,2)==33286;
ind=logical(target_ind+nontarget_ind);
CantTrials=sum(ind);

startIdx = floor(stims(ind,1) * samplingFreq) + 1;
stopIdx = startIdx + floor(opt.window * samplingFreq) - 1;


label=double(ind);
label(nontarget_ind)=2;
label(target_ind)=1;
label=label(label~=0);

for i=1:CantTrials
    epoch=signals(startIdx(i):stopIdx(i),:);
end
```

Stimulation type	**Code**
`OVTK_StimulationId_BaselineStart`	32775
`OVTK_StimulationId_BaselineStop`	32776
`OVTK_StimulationId_Beep`	33282
`OVTK_StimulationId_Button1_Pressed`	32786
`OVTK_StimulationId_DoubleBeep`	33283
`OVTK_StimulationId_DoubleBeep`	33284
`OVTK_StimulationId_ExperimentStart`	32769
`OVTK_StimulationId_ExperimentStop`	32770
`OVTK_StimulationId_Label_00 to 1F`	33024-55
`OVTK_StimulationId_LabelEnd`	33279
`OVTK_StimulationId_LabelStart`	33024
`OVTK_StimulationId_NonTarget`	33286
`OVTK_StimulationId_Number_00 to 1F`	0-31
`OVTK_StimulationId_NumberEnd`	255
`OVTK_StimulationId_NumberStart`	0
`OVTK_StimulationId_Reset`	33288
`OVTK_StimulationId_RestStart`	32777
`OVTK_StimulationId_RestStop`	32778
`OVTK_StimulationId_SegmentStart`	32772
`OVTK_StimulationId_SegmentStop`	32774
`OVTK_StimulationId_Target`	33285
`OVTK_StimulationId_Train`	33281
`OVTK_StimulationId_TrainCompleted`	33287
`OVTK_StimulationId_TrialStart`	32773
`OVTK_StimulationId_TrialSttop`	32774
`OVTK_StimulationId_VisualSteadyStateStimulationStart`	32784
`OVTK_StimulationId_VisualSteadyStateStimulationStop`	32785
`OVTK_StimulationId_VisualStimulationStart`	32779
`OVTK_StimulationId_VisualStimulationStop`	32780

Table 1: OpenVIBE Stimulation Codes, modified from `http://openvibe.inria.fr/stimulation-codes/`