# WS: BRAIN-COMPUTER INTERFACE USING OPENVIBE, AN OPEN-SOURCE SOFTWARE PLATFORM – PART 1

Arthur Desbois & Marie-Constance Corsi

ARAMIS team, Paris Brain Institute

# OUTLINE

- **PART 1**: Intro to BCI (approx. 30 minutes)
  - What is a BCI system? Examples of clinical applications
  - Designing a BCI system: goals, methods, paradigms
  - OpenViBE: an open source BCI framework
- **PART 2**: My first BCI experiment using OpenViBE – a hands-on tutorial (approx. 2 hours)
  - Details on Motor Imagery paradigm, and features of interest
  - Scenario 1 – data acquisition
  - Scenario 2 – signal processing
  - Scenario 3 – classification
- **PART 3**: Going further (interactive choice!) & concluding remarks (approx. 30 minutes)
  - C++ Algo Box development for OpenViBE
  - Python/Matlab scripting
  - Current works and perspectives on BCI

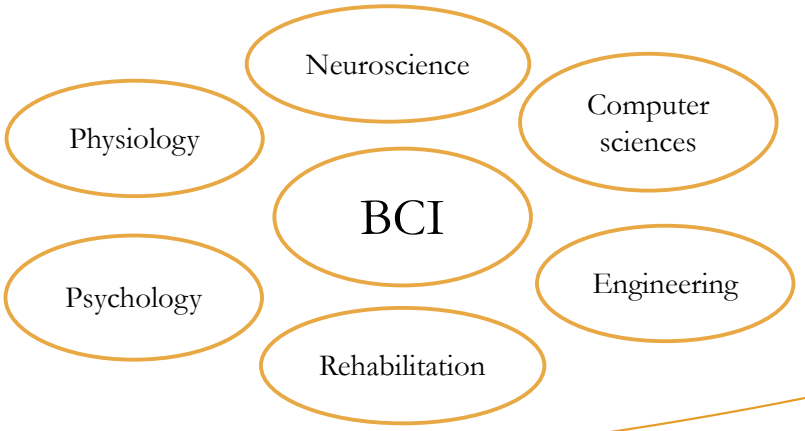

**Reminder: you need to have OpenViBE installed!**

http://openvibe.inria.fr/downloads/

# INTRODUCTION TO BCI SYSTEMS

# WHAT IS A BCI ?

# CONTEXT




Today

1970

1st implanted BCI

2000

~10k publications
3 research groups

2005

> 30k publications
6-8 research groups

2010

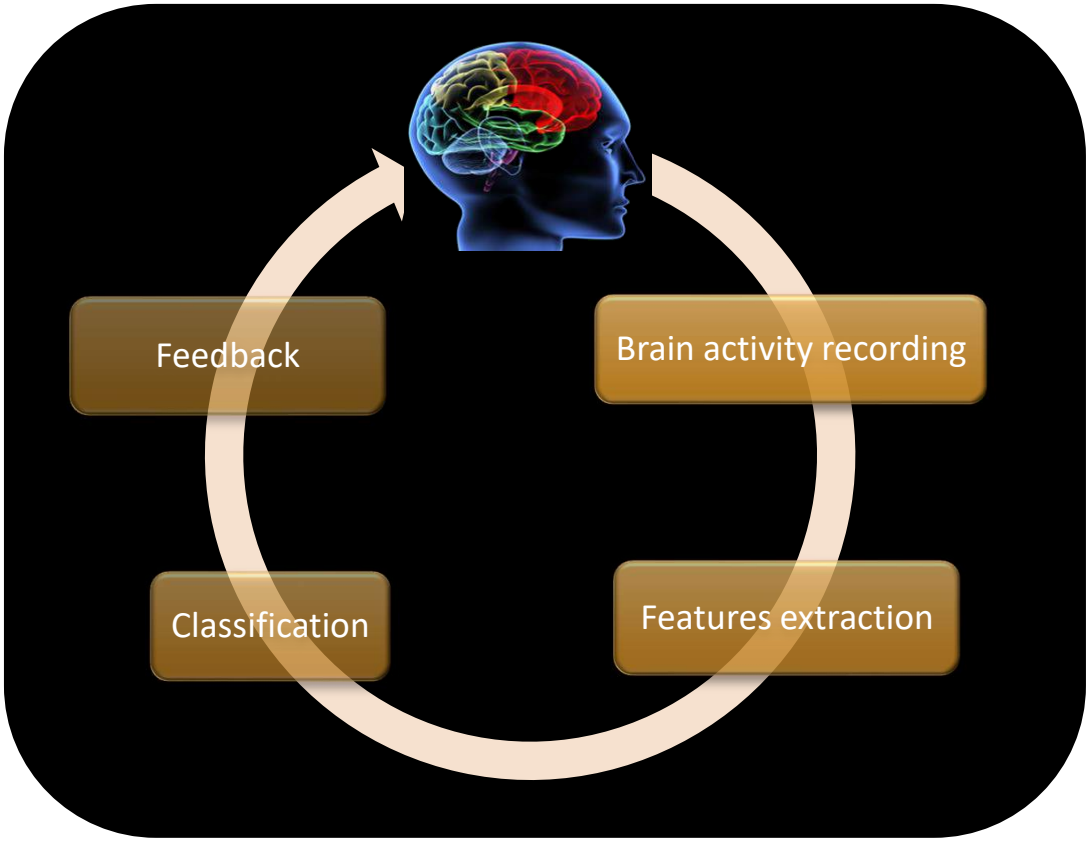
> 50k publications
> 100 research groups

Commercialization of devices oriented to general public

Emotiv, US

2015

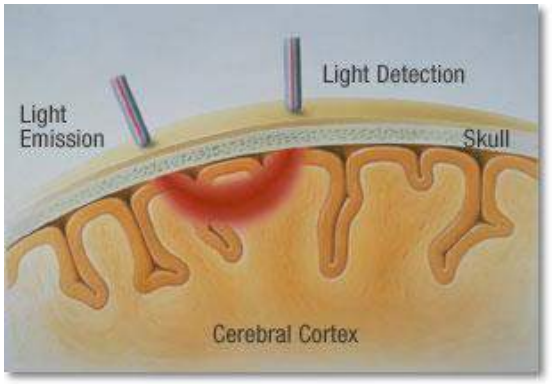
# BEHIND THE MAGIC…



# TOOLS TO CAPTURE BRAIN ACTIVITY (NON INVASIVELY)

Electroencephalography (EEG)
(Mak et al, 2012)

Magnetoencephalography (MEG)
(Mellinger et al, 2007)



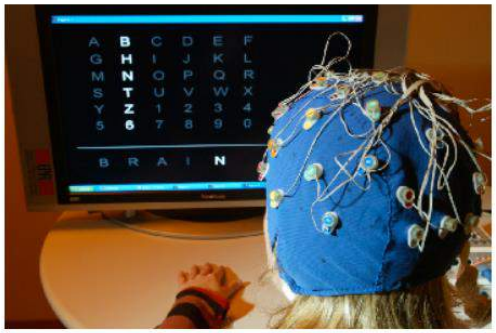

Near Infrared Spectroscopy (NIRS)
(Fazli et al, 2012)

Functional MRI
(Sitaram et al, 2009)

# CLINICAL APPLICATIONS

- Control
  - Prosthesis (Fifer et al, 2014)
  - Wheelchair (Carlson & Millan, 2013)
  - Quadcopter (LaFleur et al, 2013)

- Communication
  - Verbal & nonverbal communication (Jin et al, 2012; Hwang et al, 2012; Kashihara, 2014)
  - Silent talk (Naci et al, 2013)




BCI & communication

- Neurological disorders treatment
  - Stroke (Prasad et al, 2010)
  - Spinal cord injury (King et al, 2013)
  - Consciousness (Chatelle et al, 2012)
  - Psychiatric disorders (Arns et al, 2017)

# OPENVIBE - EXAMPLES OF RESEARCH AND CLINICAL APPLICATIONS

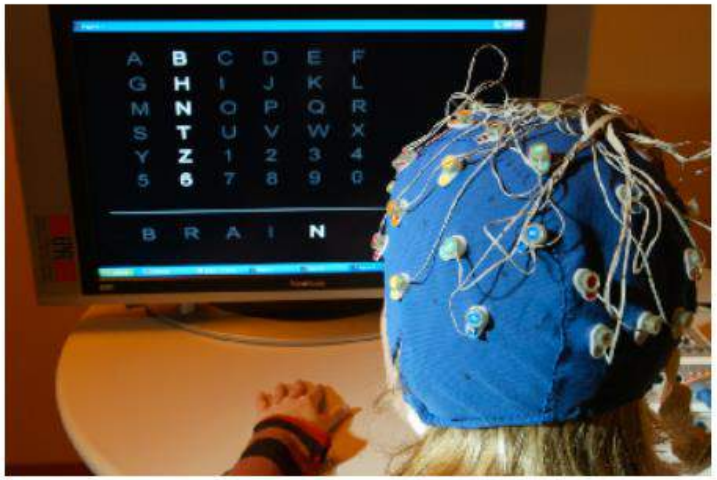
- OpenViBE for:
  - Robotic device control
  - Stroke rehabilitation
  - Better monitoring general anesthesia

- Involved laboratories
  - LORIA team (Nancy, France)
  - Hybrid team (Rennes, France)
  - Potioc team (Bordeaux, France)
  - ARAMIS team (Paris, France)

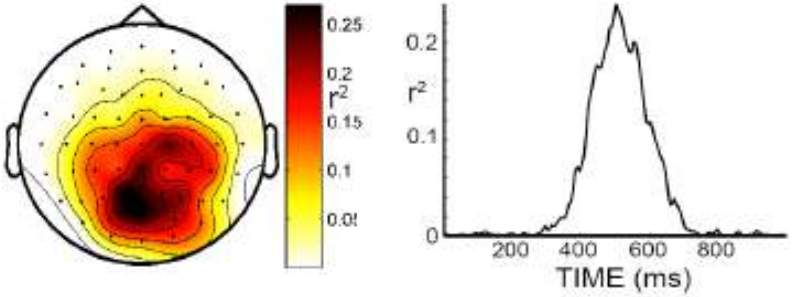
LORIA projects
Courtesy of S. Rimbert

ARAMIS projects
Courtesy of T. Venot

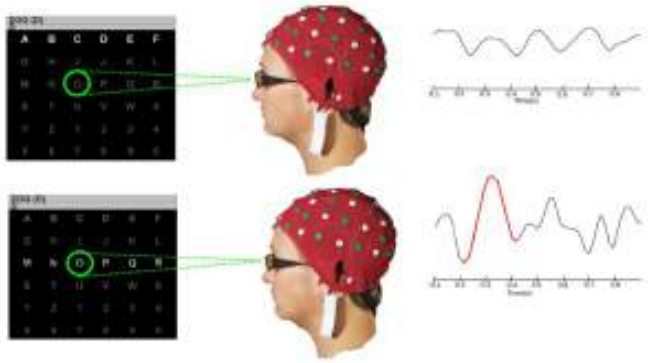
# DIFFERENT TYPES OF BCI – P300 SPELLER




P300 Speller

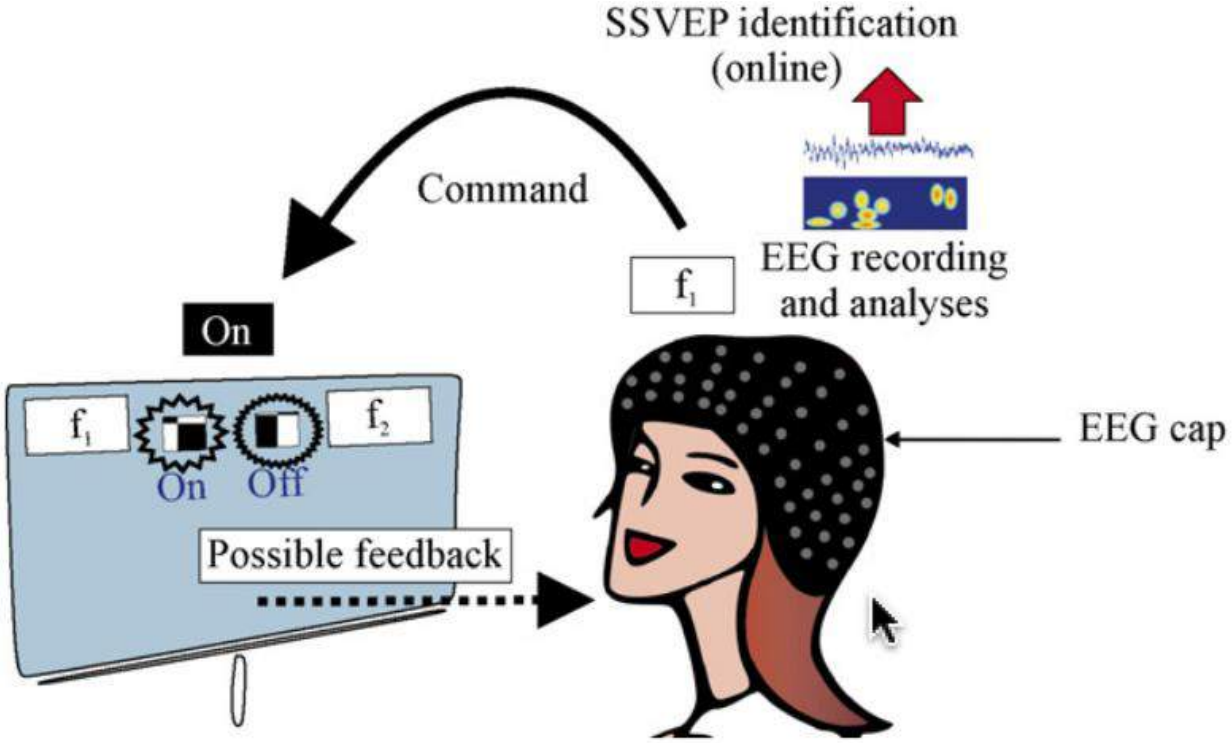



Illustrations from BCI2000 website



(Lotte et al, 2015)

# DIFFERENT TYPES OF BCI – VISUAL EVOKED POTENTIAL (VEPS)




(Vialatte et al, 2010)

# OPENVIBE

## INTRODUCTION - AN OPEN-SOURCE BCI FRAMEWORK

# OPENVIBE

- Open-source software platform - http://openvibe.inria.fr/
  - Design, test & use BCIs
  - Generic system for realtime EEG acquisition, processing and visualization
- Key features - http://openvibe.inria.fr/features/
  - Modularity, flexibility
  - Aimed for different types of users
  - Portability, cross-platform
  - Compatibility with EEG hardware, + VR integration
  - Compatibility with Python, MATLAB, LUA
  - BCI paradigms available as demos (P300, MI, Neurofeedback…)

# OTHER BCI SYSTEMS & TOOLBOXES

- BCI2000
  - C++ based system for BCI research
  - Not open source, but sources & executables available for free (non-profit & educ. purposes)
  - Real-time BCI through 4 stages : acquisition, processing, user interface, operator/visualization interface

- BioSig
  - Matlab / Octave toolbox

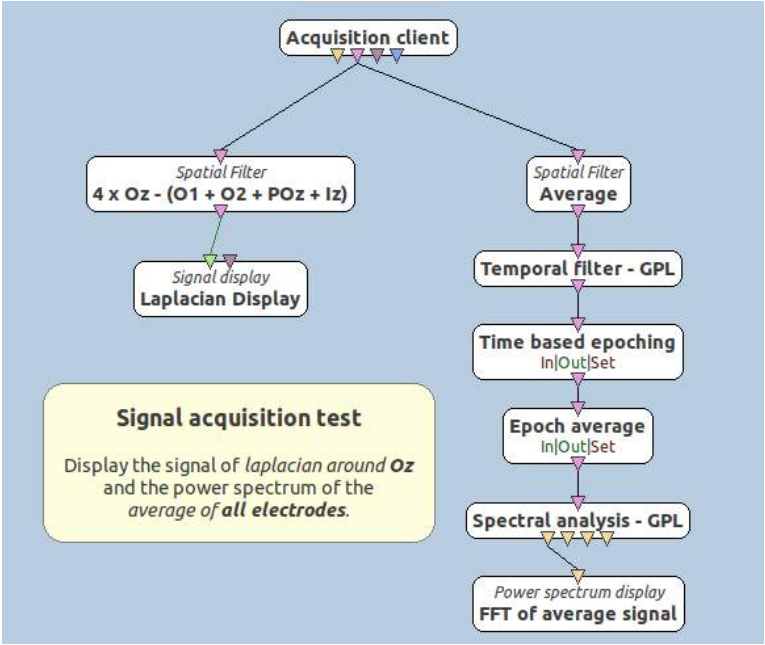
- BCI++
  - C/C++ based framework for designing BCI experiments.
  - Not open-source

# OPENVIBE - EEG HARDWARE COMPATIBILITY

- Any EEG device can be supported, through the development of a C++ driver
- Already supported :
  - Brain Products devices (Brainamp series, VAmp…)
  - Brainmaster (Atlatis, Discovery)
  - EGI (Netamps 300)
  - Micromed devices (via SystemPlus Evolution software)
  - OpenEEG
  - Neurosky
  - …
- Full list on http://openvibe.inria.fr/supported-hardware/

# OPENVIBE - MODULARITY

- No need to be a programmer to design a BCI system!
  - Processing boxes
  - Links for different types of data flow & info
  - Easy prototyping, flexibility for experimenting



# OPENVIBE: OPEN SOURCE & COMMUNITY

- Contributions are possible… and more than welcome!
  - Many processing boxes have been integrated following user submissions
- Community - http://openvibe.inria.fr/forum/
  - Discussions about usage, ongoing developments, issues…
- License - http://openvibe.inria.fr/license/

  Fully AGPL-3 (http://www.gnu.org/licenses/agpl-3.0.html)

# OPENVIBE: OPEN SOURCE & COMMUNITY

- Existing tutorials
  - Level 1 - beginners
    - My first OpenViBE setup
    - Choosing my BCI paradigm
    - Using a (new) hardware with OpenViBE

- Level 2 – more advanced
  - Troubleshooting OpenViBE scenarios
  - How to do repeatable experiments with OpenViBE
  - Using Python with OpenViBE
  - Using Matlab with OpenViBE
- Level 3 – OpenViBE black belt
  - How to make a box plugin library

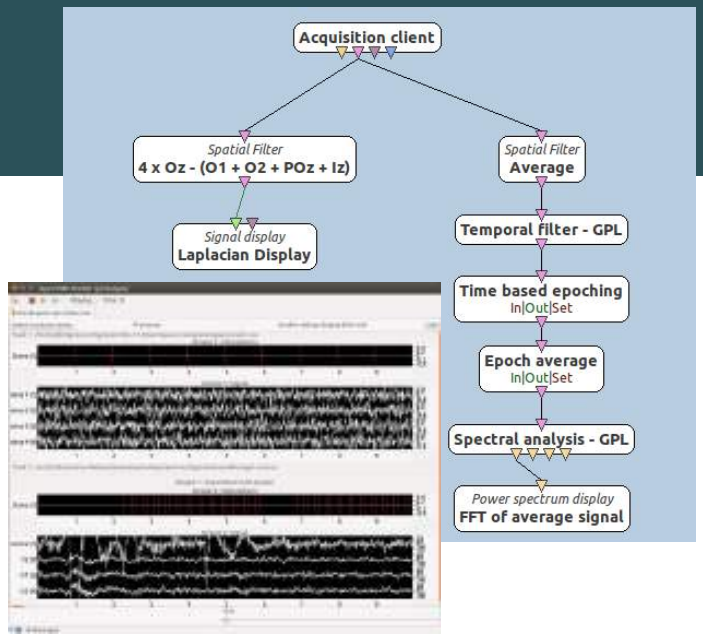
# OPENVIBE

## TECHNICAL CONCEPTS

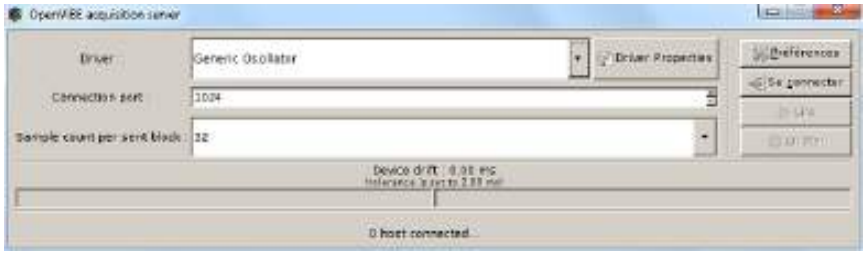
# OPENVIBE – IN DEPTH

- OpenViBE is made of two principal applications :

- **OpenViBE Designer**
  - For creating, modifying and using BCI scenarios
  - Fully graphical interface, data visualization
  - Processing boxes to link and parametrize

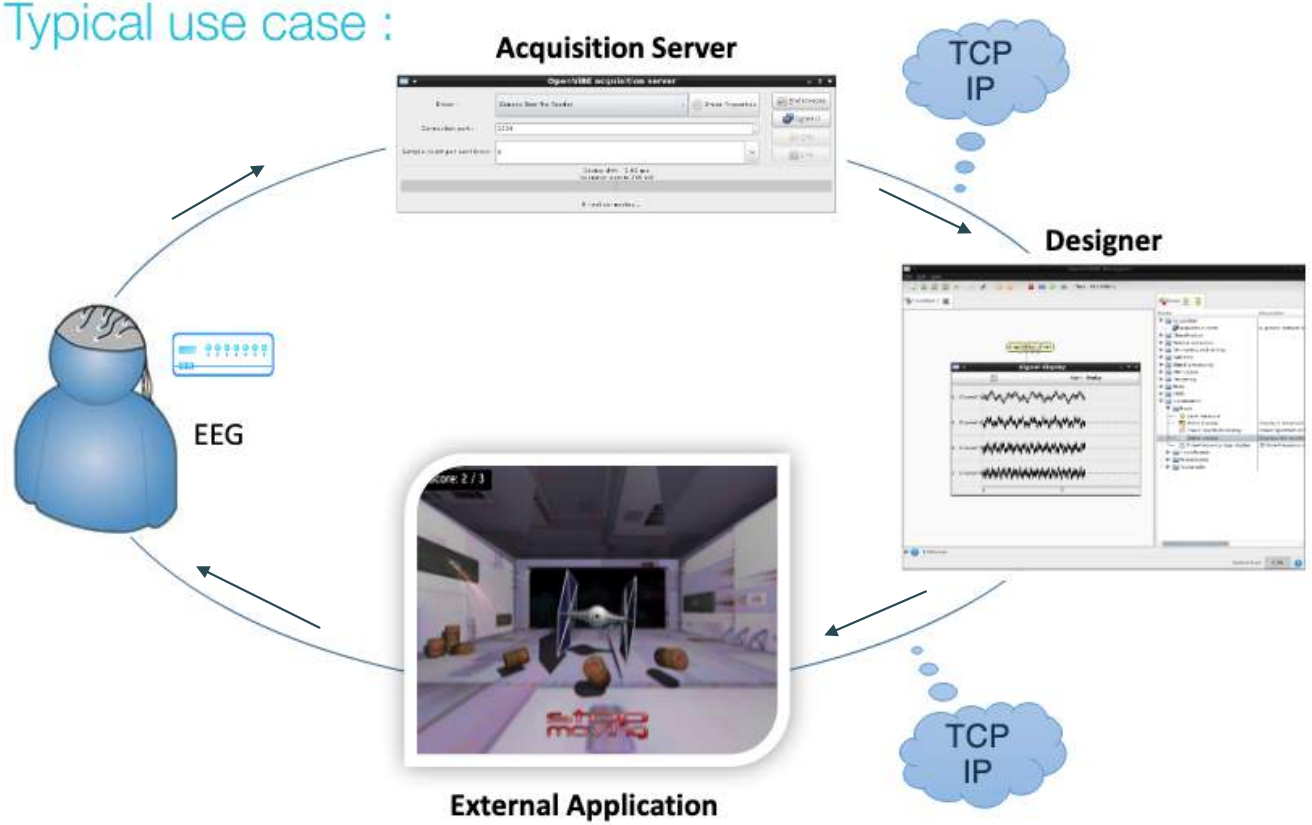



- **OpenViBE Acquisition Server**
  - Acquires EEG & bio signals from the hardware
  - Translates signals to a common format
  - Transmits data to connected apps, such as the Designer, over a local network (or on the same computer)



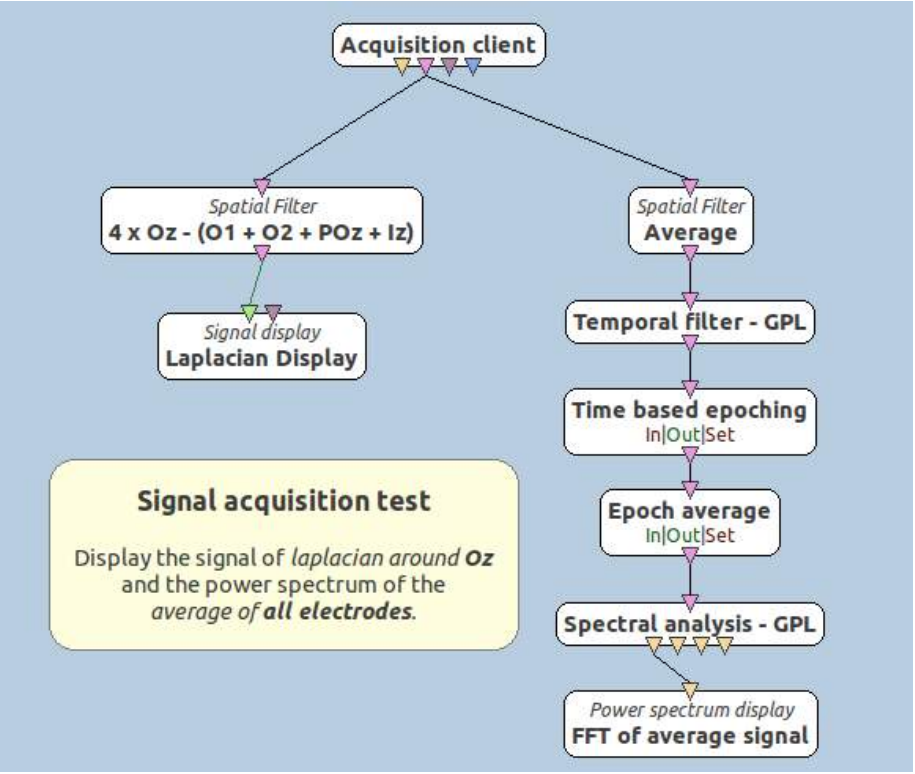
# OPENVIBE – TYPICAL USECASE

- Typical use case :



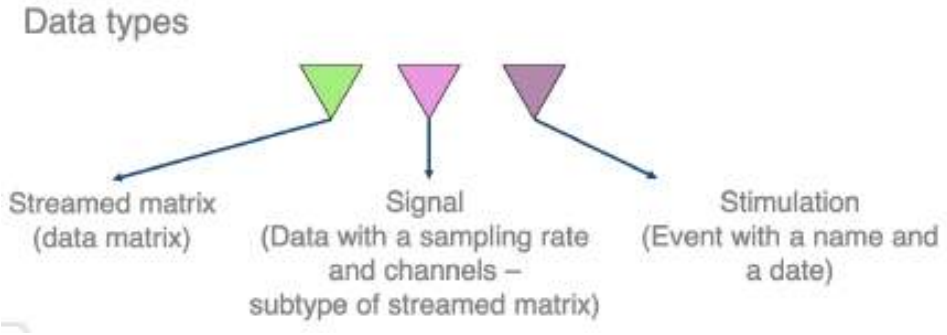
# OPENVIBE – PLUGINS & « BOXES »

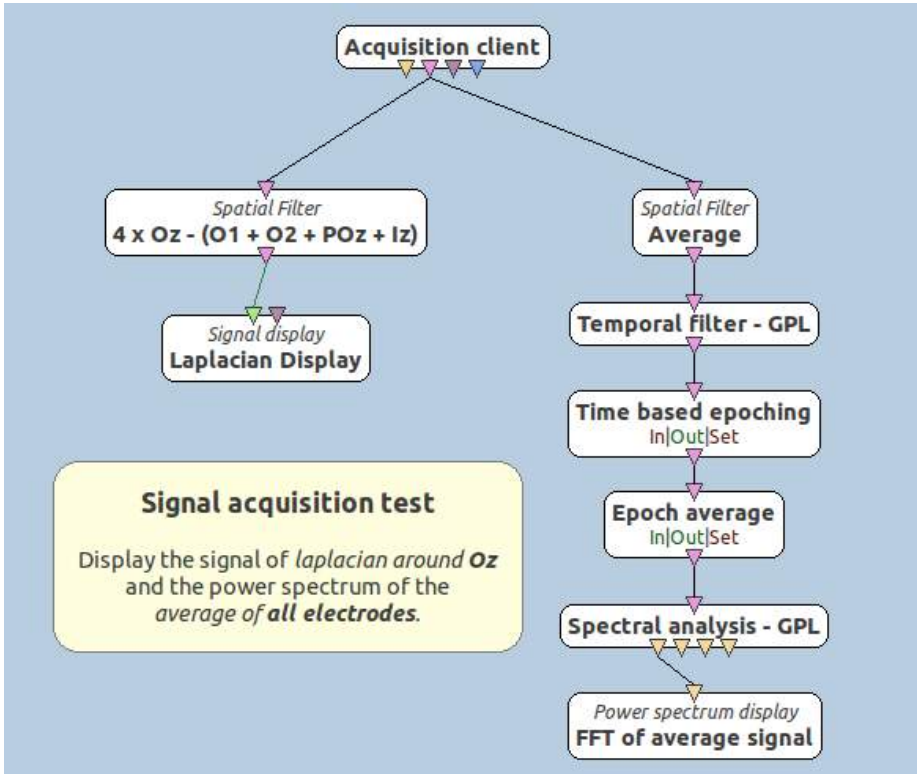
- **Processing boxes**
  - I/O (file and stream reading/writing)
  - Signal Processing (filtering, spectral analysis…)
  - Classification
  - Visualization (topography, time signals, spectra…)
  - Scripting (LUA, Matlab, Python…)
  - Scenario/experiment management
- **Different links btw. boxes**
  - Signal streams (multi-channel signal, matrix,…)
  - Experiment stimulations



# OPENVIBE – PLUGINS & « BOXES »

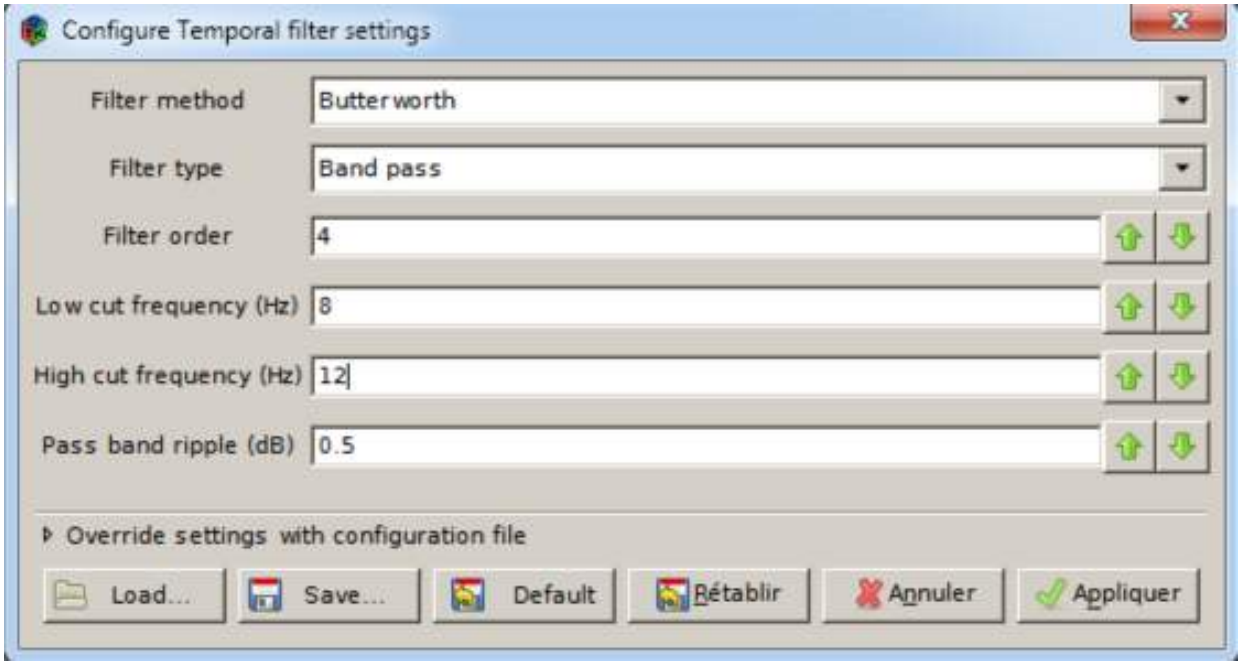
- **Data Streams**

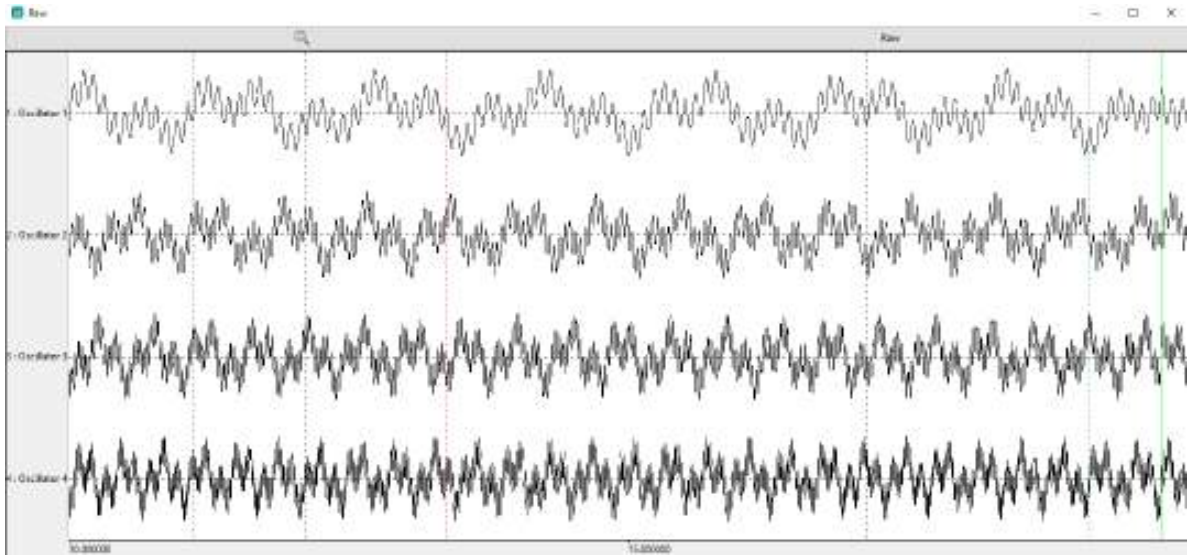
# OPENVIBE – PLUGINS & « BOXES »

- **Box parameters**



# OPENVIBE – STIMULATION CONCEPT (+ GRAPH)

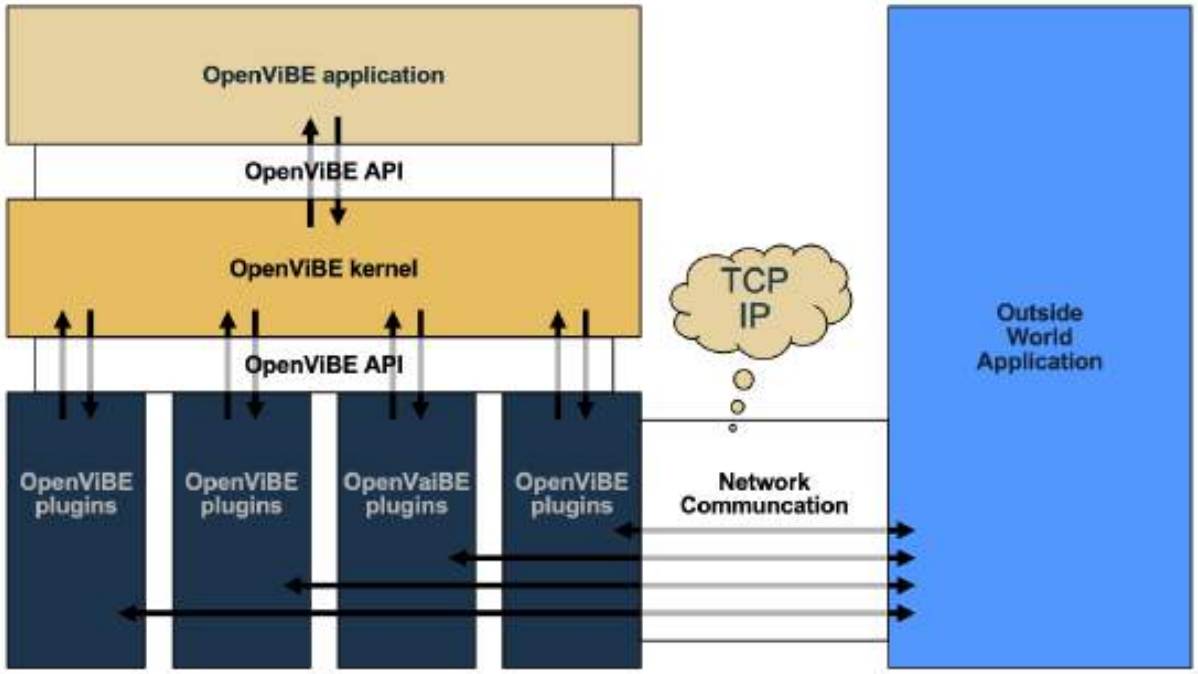
- **“Stimulations” = events**
  - Management of particular sections or useful events in a BCI experiment
  - *Experiment Start/Stop*
  - *Button pressed*
  - *New trial…*
- **Synchronized with the acquired signal**
- **More info in Part 2!**



# CHAPTER 1 – Q&A

BCI Motor Imagery with OpenViBE in X-Men: First Class

# OPENVIBE – PLUGINS & « BOXES »



# WS: BRAIN-COMPUTER INTERFACE USING OPENVIBE, AN OPEN-SOURCE SOFTWARE PLATFORM – PART 2

Arthur Desbois & Marie-Constance Corsi

ARAMIS team, Paris Brain Institute

# CHAPTER 1

## PREREQUISITES BEFORE PERFORMING A MI-BCI EXPERIMENT

# DIFFERENT TYPES OF BCI

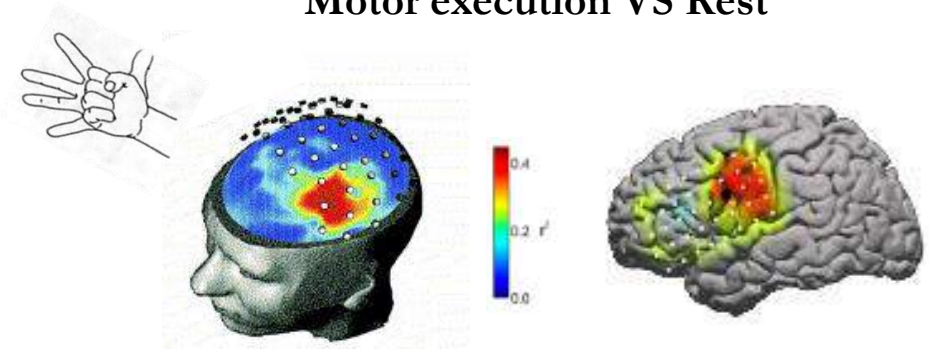
## Underlying idea

Taking advantage of a neurophysiological phenomenon to establish a communication between the brain and the computer

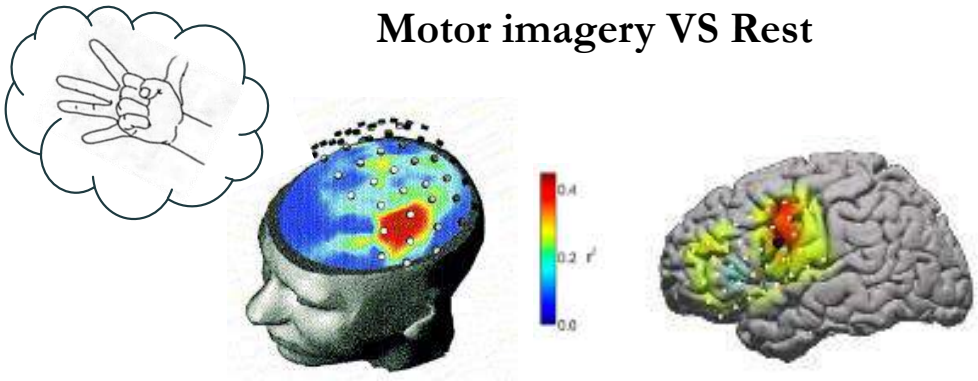
## Illustration with Motor imagery-based BCI

# MOTOR IMAGERY – OBSERVATIONS

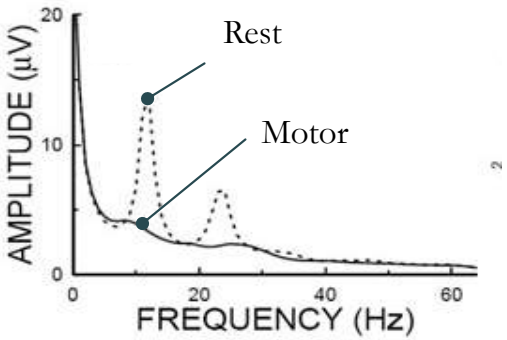
**Motor execution VS Rest**



**Motor imagery VS Rest**



Power decrease

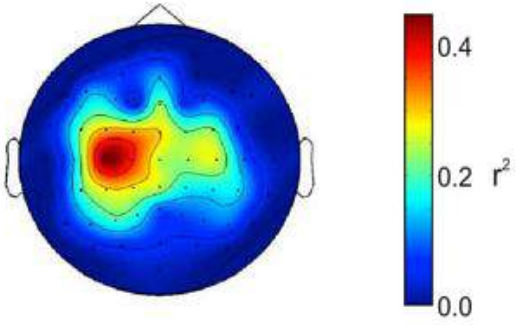



Desynchronization effect
(Pfurtscheller et al, 1999)

# MOTOR IMAGERY – MU-BETA RHYTHM

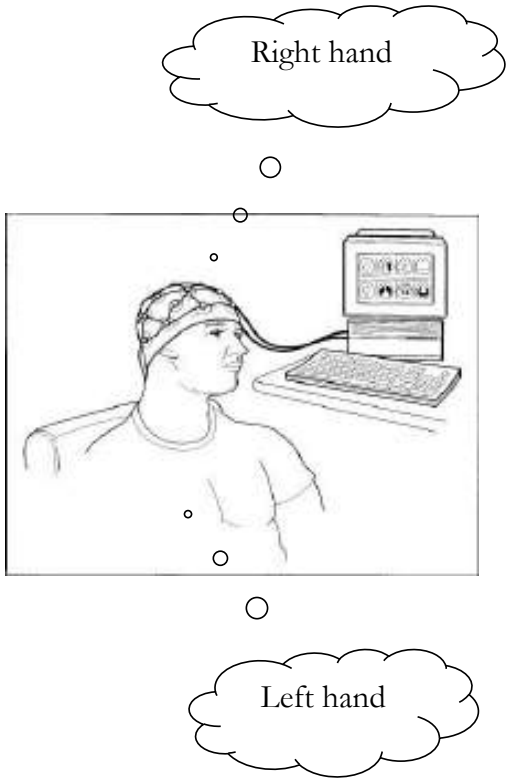
- Behavioral properties
  - Movement / preparation for movement : Event-related desynchronization (ERD) (Pfurtscheller, G, Lopes da Silva, FH, 1999)
  - With relaxation/post-movement period : ERS

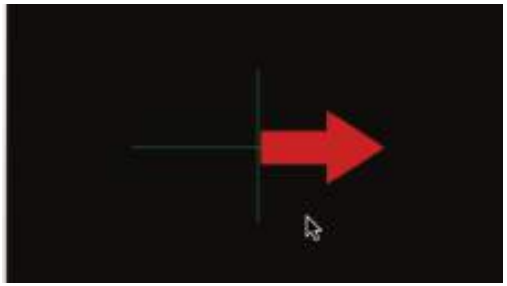
- Why using it in BCI ?
  - Mu/Beta activity modulation by motor-imagery, a way to communicate
  - Use of power spectra
  - To establish this communication :
    - Spatial selection
    - Frequency selection

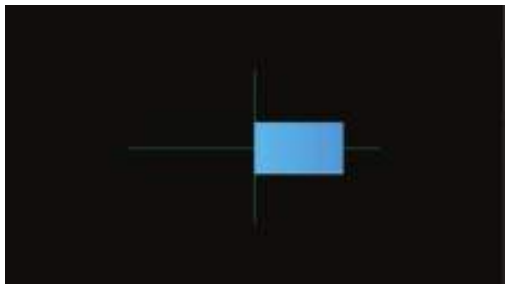


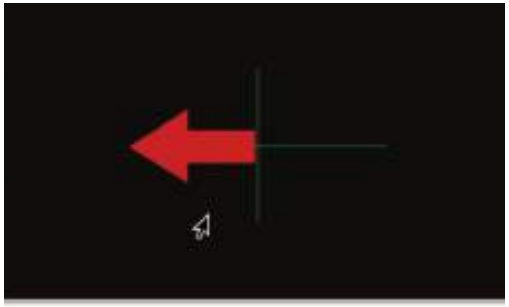

Illustrations from BCI2000 website

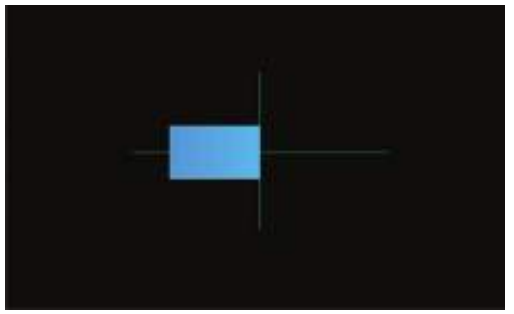
# MOTOR IMAGERY – IN PRACTICE



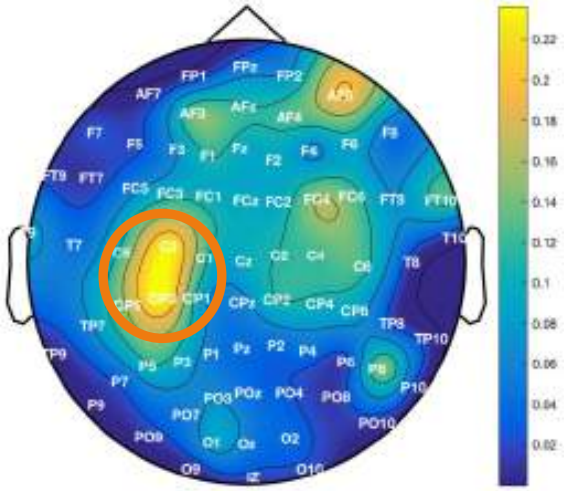






# MOTOR IMAGERY – IN PRACTICE

@ 10Hz




$r^2$

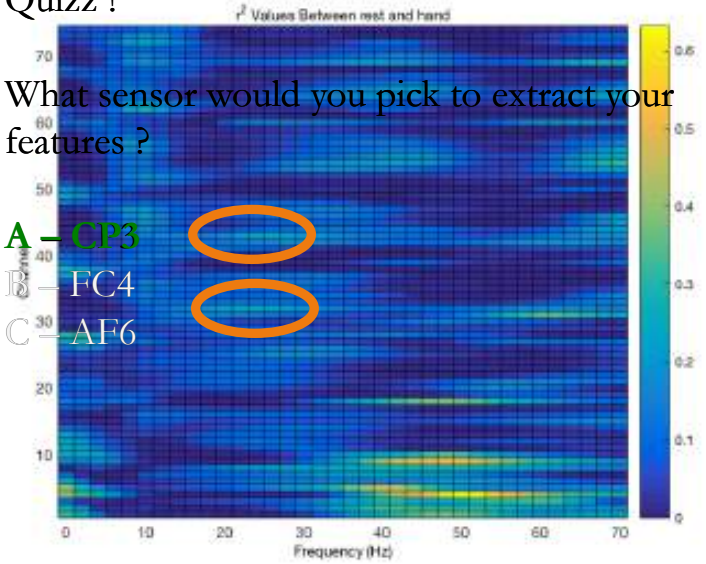
Quizz !

What sensor would you pick to extract your features ?

A – CP3

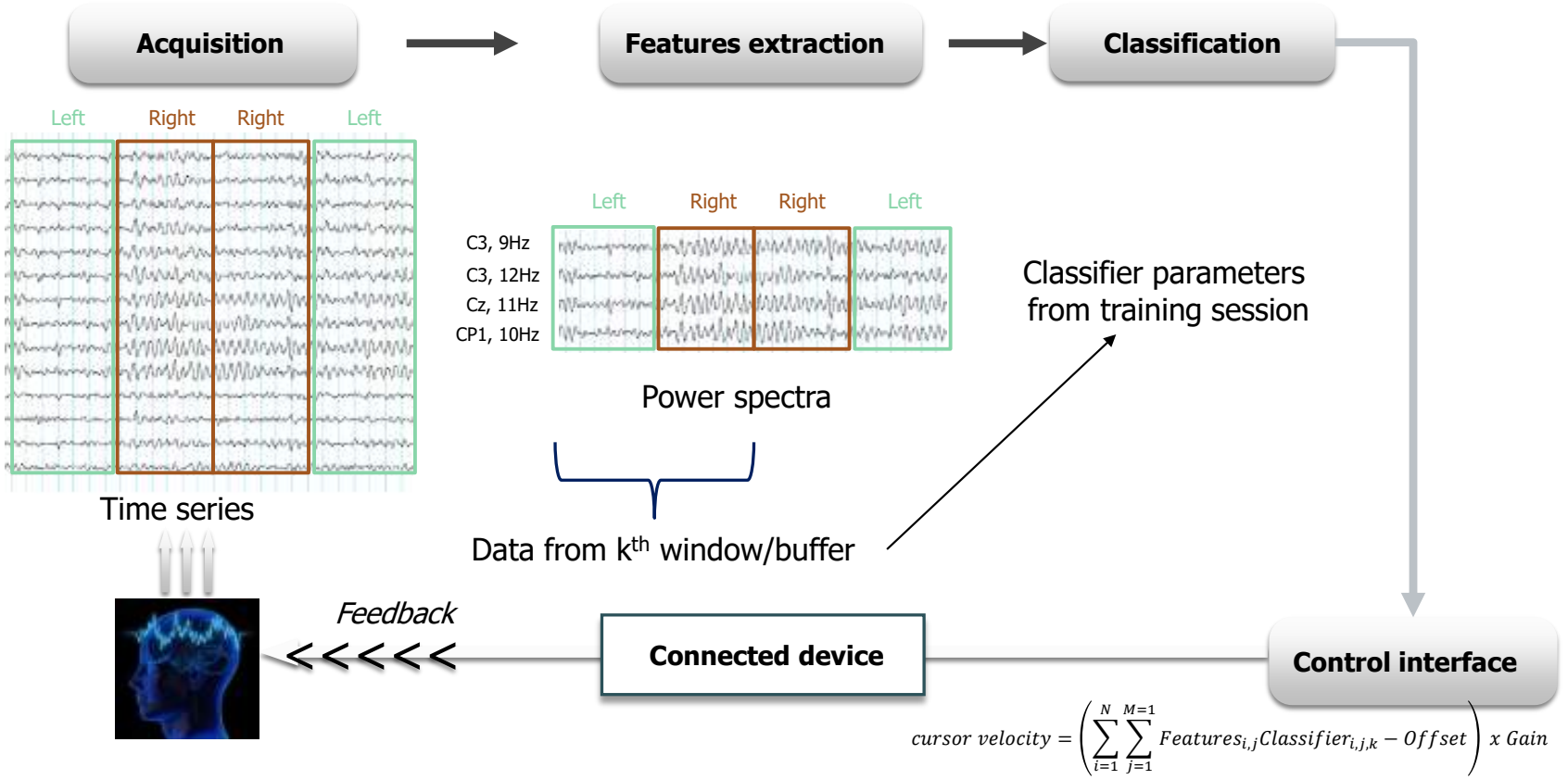
B – FC4

C – AF6

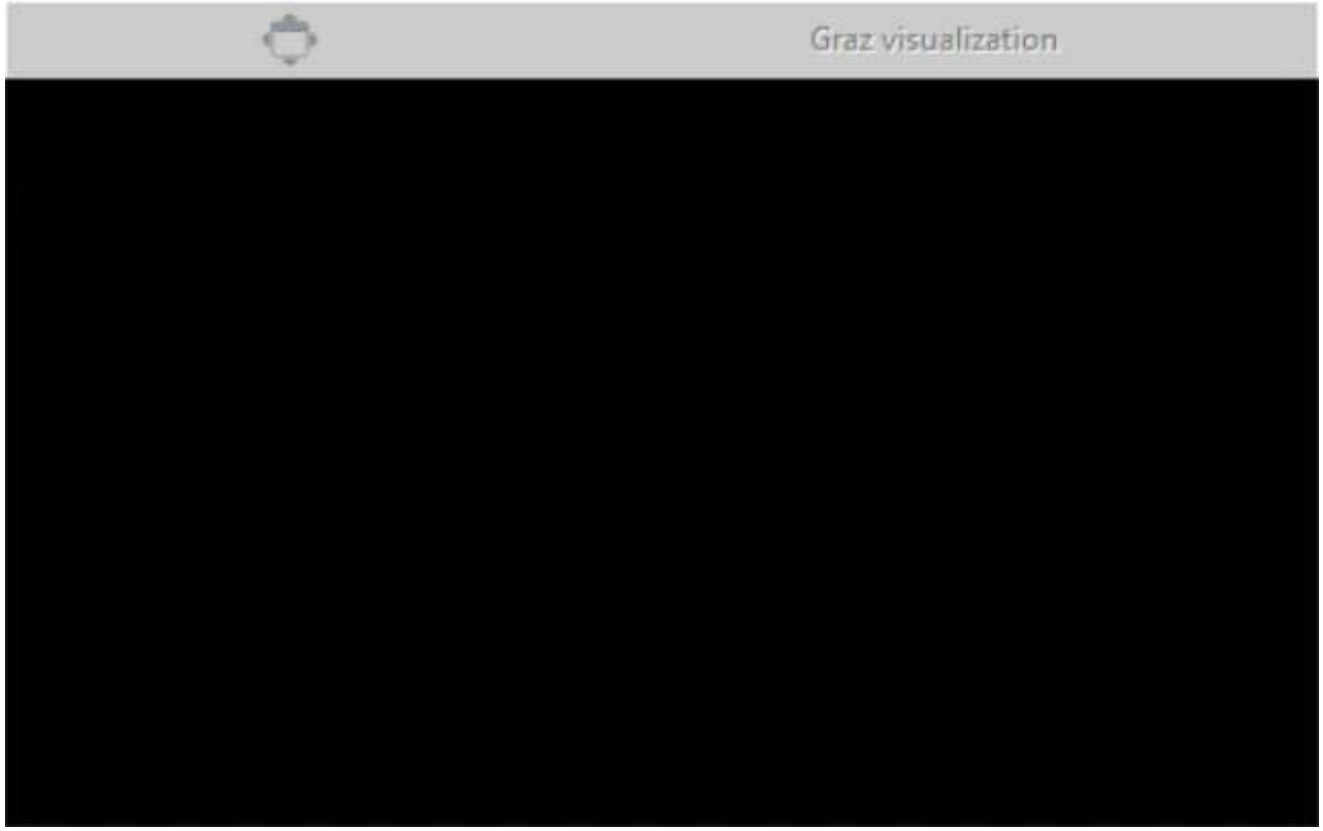



$r^2$

# MOTOR IMAGERY – IN PRACTICE



# MOTOR IMAGERY – IN PRACTICE



# CHAPTER 2

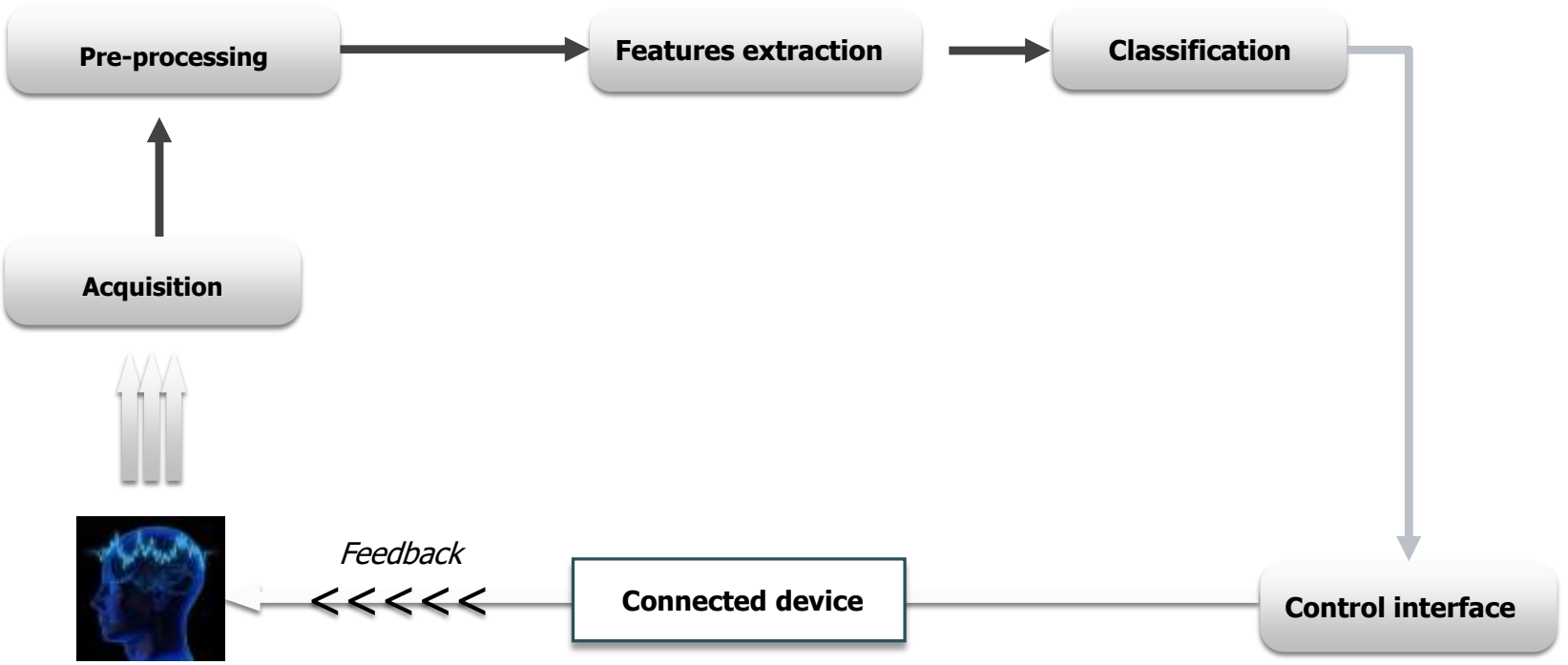
## BCI-CUTTINGEEG PROTOCOL SET-UP

# RESOURCES

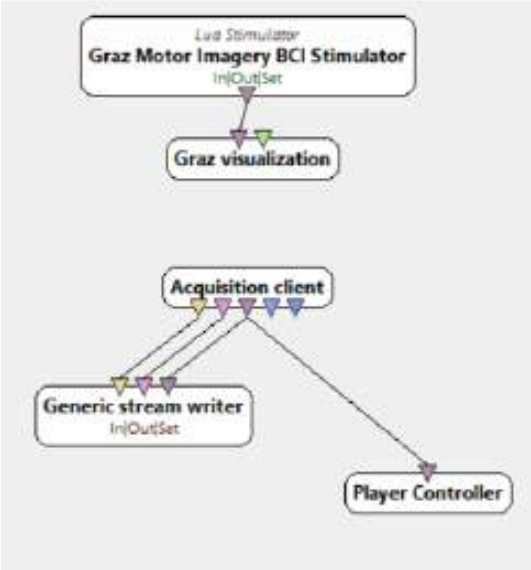
- GitHub repo:

  https://github.com/BCI-NET/BCI-OpenViBE-CuttingEEG2021/

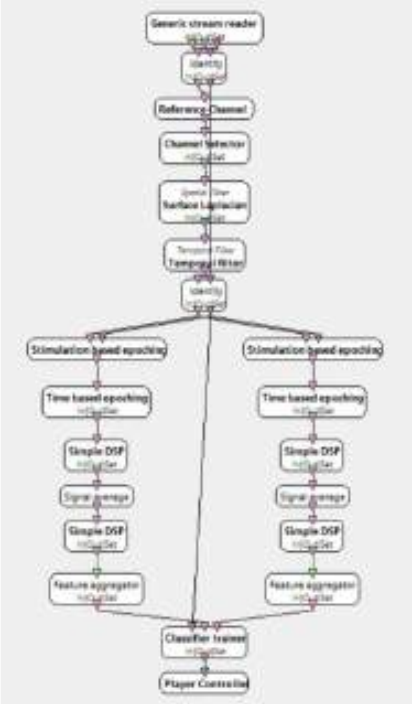
# AIM OF THE PART



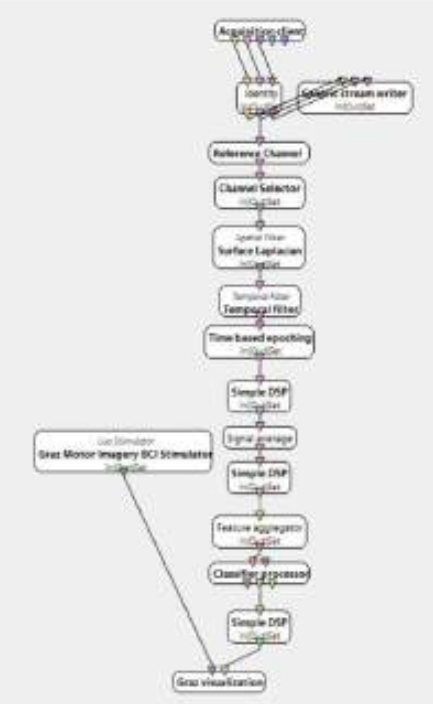
# AIM OF THE PART




1- Data acquisition

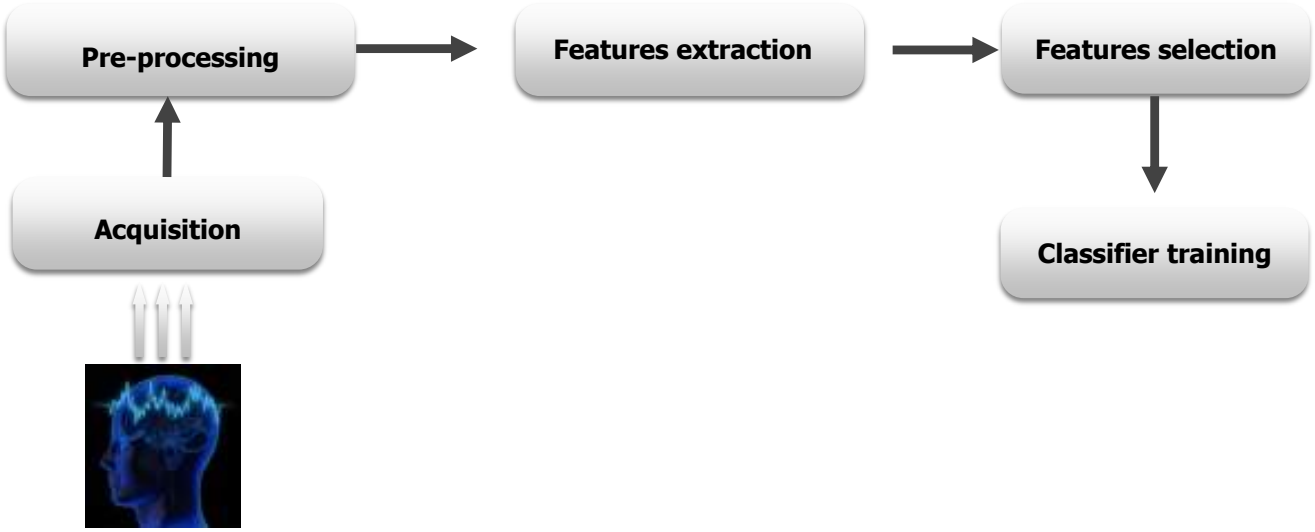


2- Features extraction & Classification



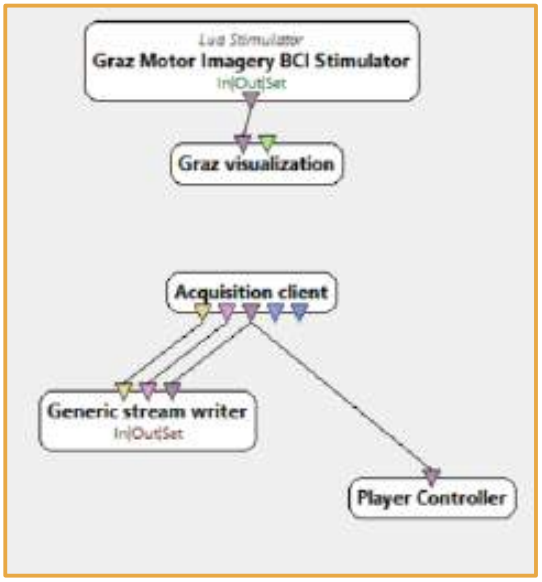
3- Online feedback

# AIM 1 – BUILDING THE CLASSIFIER…




*Adapted from (X. Navarro & F. Grosselin)*

# SC 1 – DATA ACQUISITION




**1- Data acquisition**

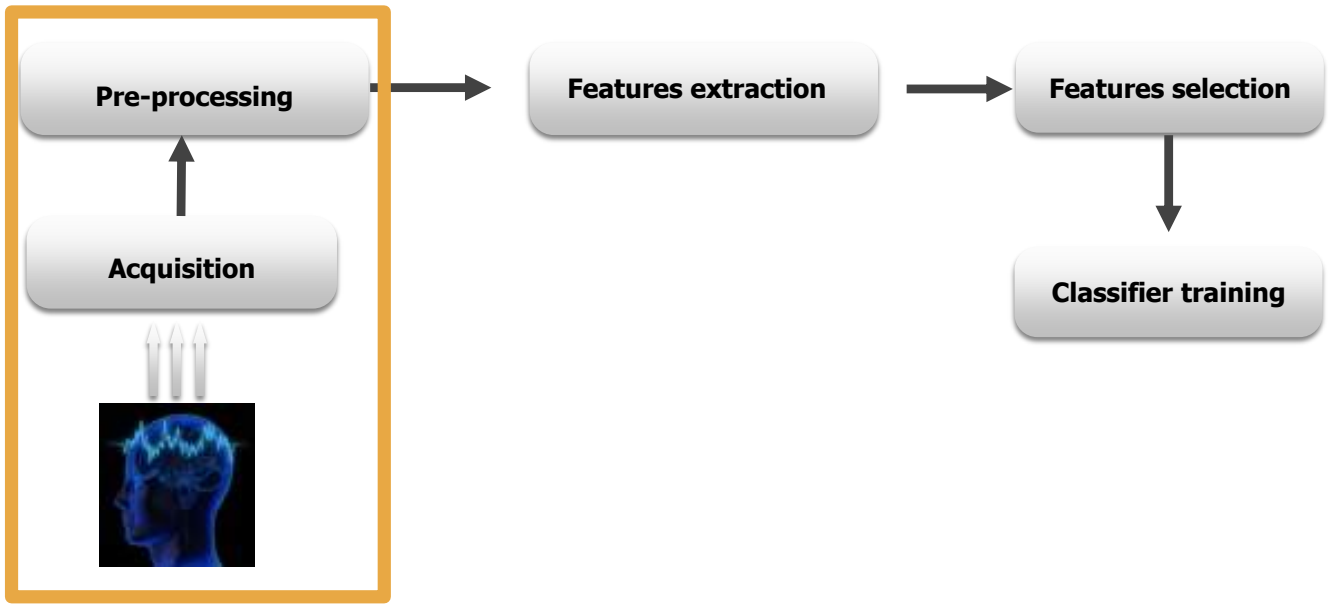


2- Features extraction & Classification



3- Online feedback

# SC 1 – DATA ACQUISITION




*Adapted from (X. Navarro & F. Grosselin)*

# SC1 – DATA ACQUISITION

- **Sub-chapters:**
  - Warm-up, first steps: simple I/O
  - Warm-up 2: acquisition server
  - BCI acquisition protocol, Stimulations

# SC1 - STEP 1 – WARMUP ! SIMPLE I/O AND DISPLAY

- **Sub-chapters:**
  - Warm-up, first steps: simple I/O
  - Warm-up 2: acquisition server
  - BCI acquisition protocol, Stimulations

# SC1 - STEP 1 – WARMUP ! SIMPLE I/O AND DISPLAY

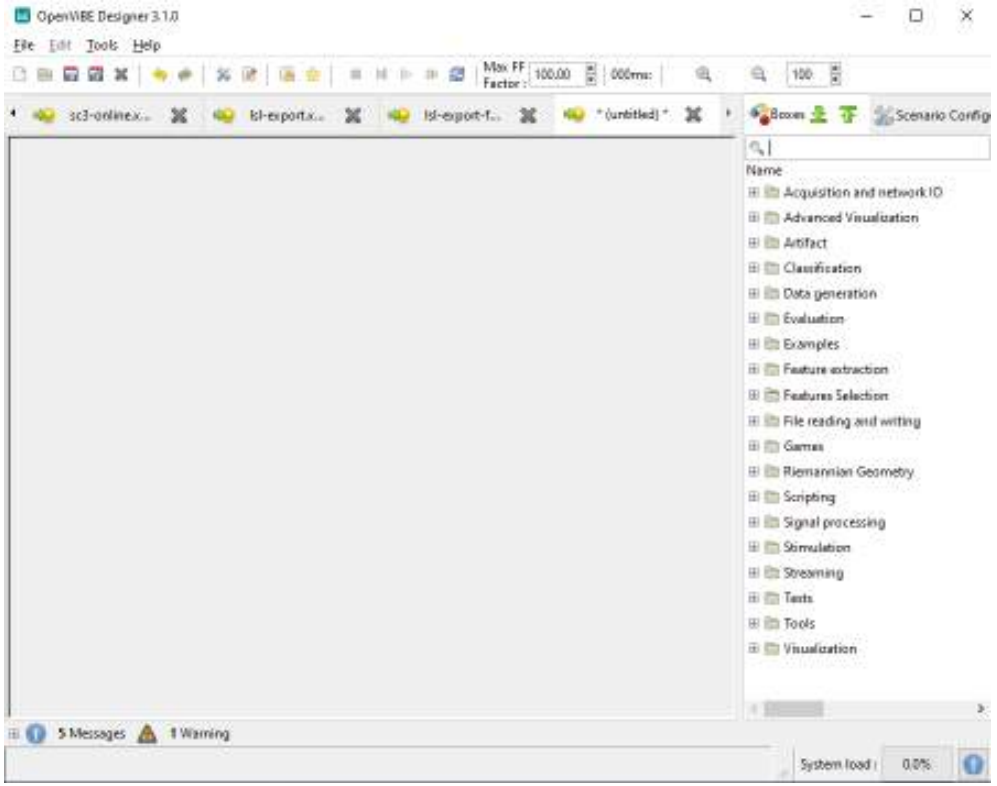
- **Goal : load & display file :**

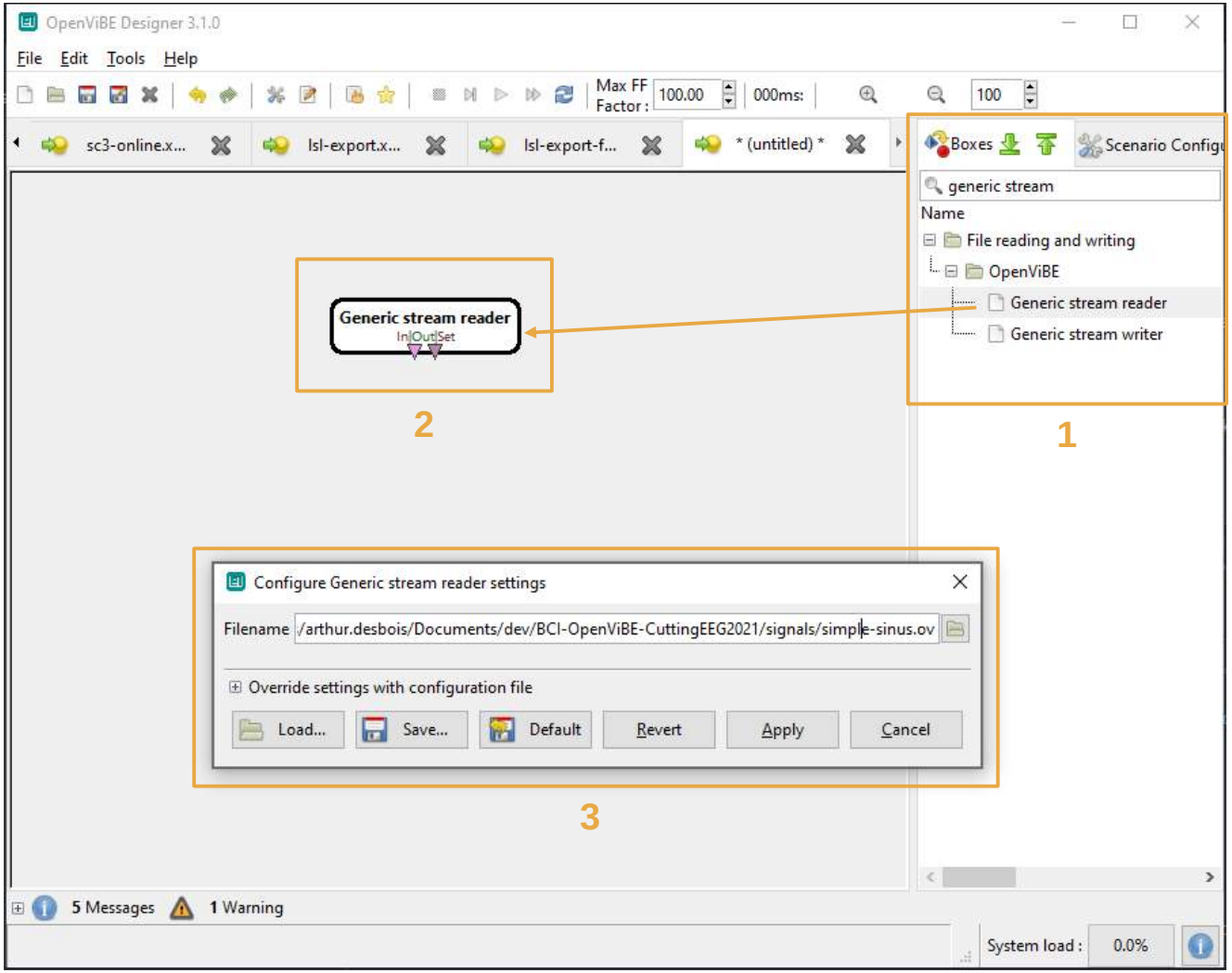
  signals/simple-sinus.ov

  (in Github repo)

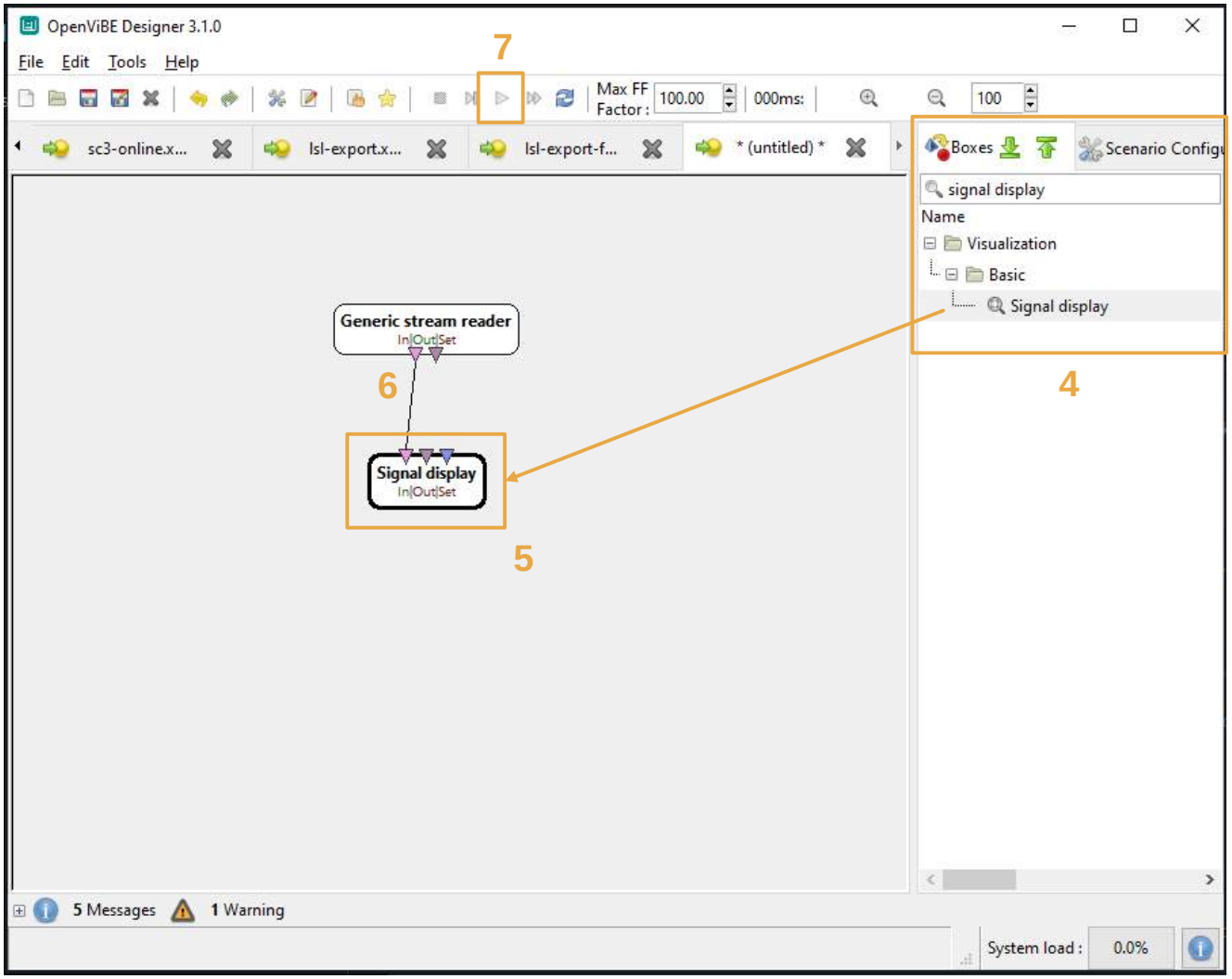
- **0- Launch OpenViBE Designer :**

  *openvibe-designer.cmd*






- 1- Search for “generic stream reader” in boxes list
- 2- Drag & drop to designer window
- 3- Set filename with browser




- 4- Search for “signal display” in boxes list
- 5- Drag & drop to designer window
- 6- Drag & drop connection between boxes, using the “signal” stream type
- 7- Press Play!

# SC1 - STEP 2 – WARMUP ! ACQ SERVER

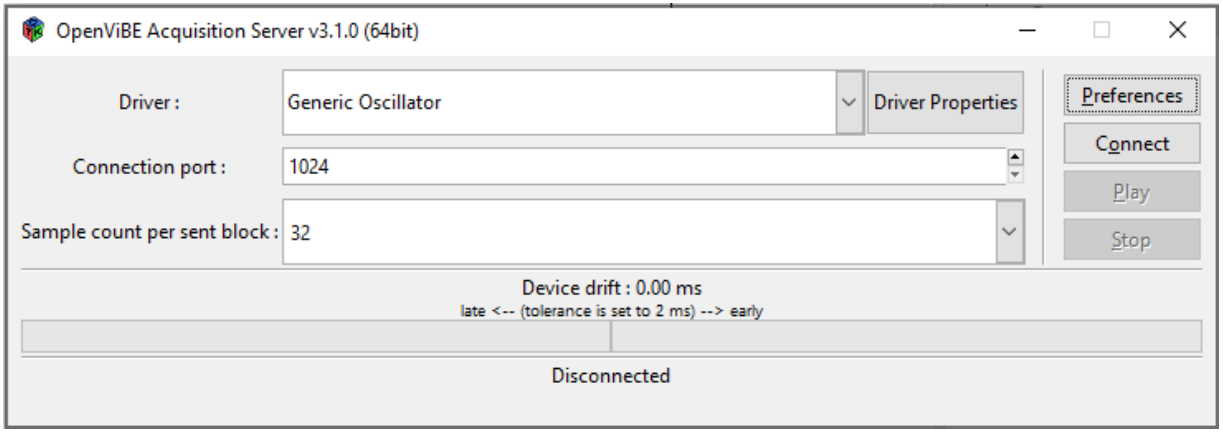
- **Sub-chapters:**
  - Warm-up, first steps: simple I/O
  - Warm-up 2: acquisition server
  - BCI acquisition protocol, Stimulations

# SC1 - STEP 2 – WARMUP ! ACQ SERVER

- **Goal : use acquisition server/client** with a simple sine generator

- **0- Launch OpenViBE Acquisition server :**

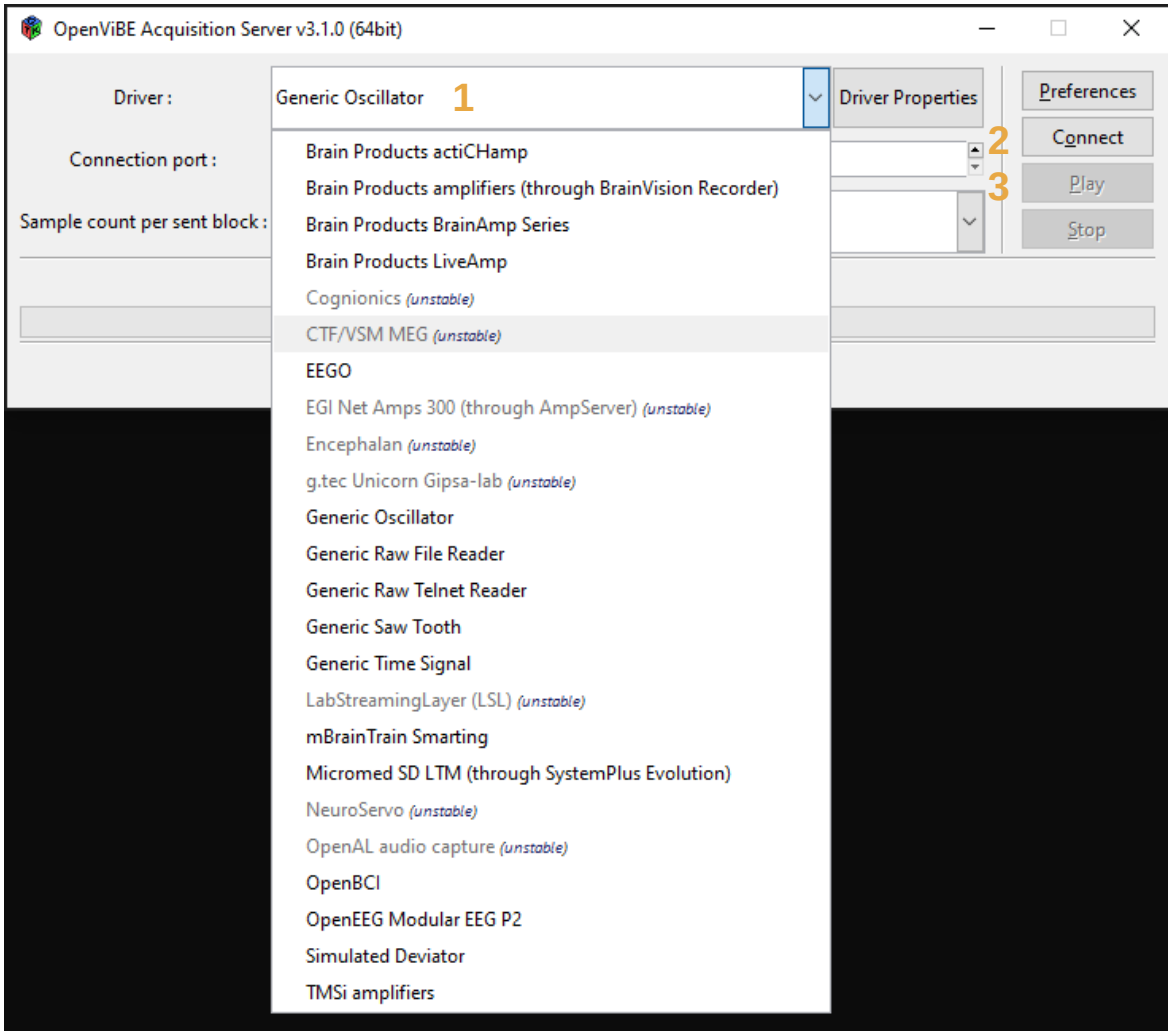
  *openvibe-acquisition-server.cmd*

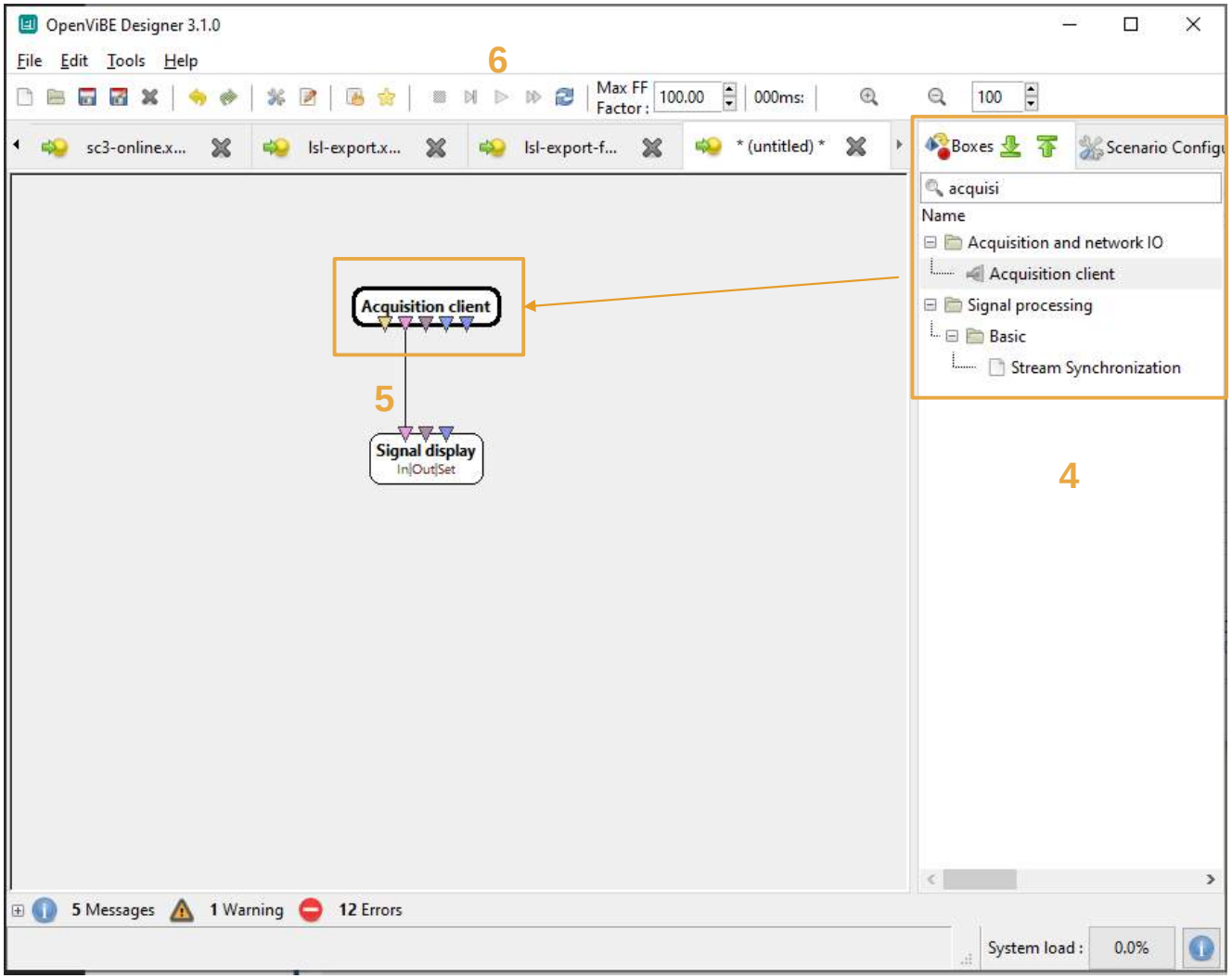


- **1- In the “Driver” list, select “Generic Oscillator”**
  In this list, you’ll find all the drivers for OpenViBE’s supported EEG hardware

- **2- Click “Connect”**

- **3- Click “Play”**






- 4- In the Designer, look for the "Acquisition client" Box and add it to your scenario
- 5- Connect it to the Signal Display Box
- 6- Play the scenario

# SC1 - STEP 3 – PROTOCOL MANAGEMENT & STIMULATIONS…

- **Sub-chapters:**
  - Warm-up, first steps: simple I/O
  - Warm-up 2: acquisition server
  - BCI acquisition protocol, Stimulations

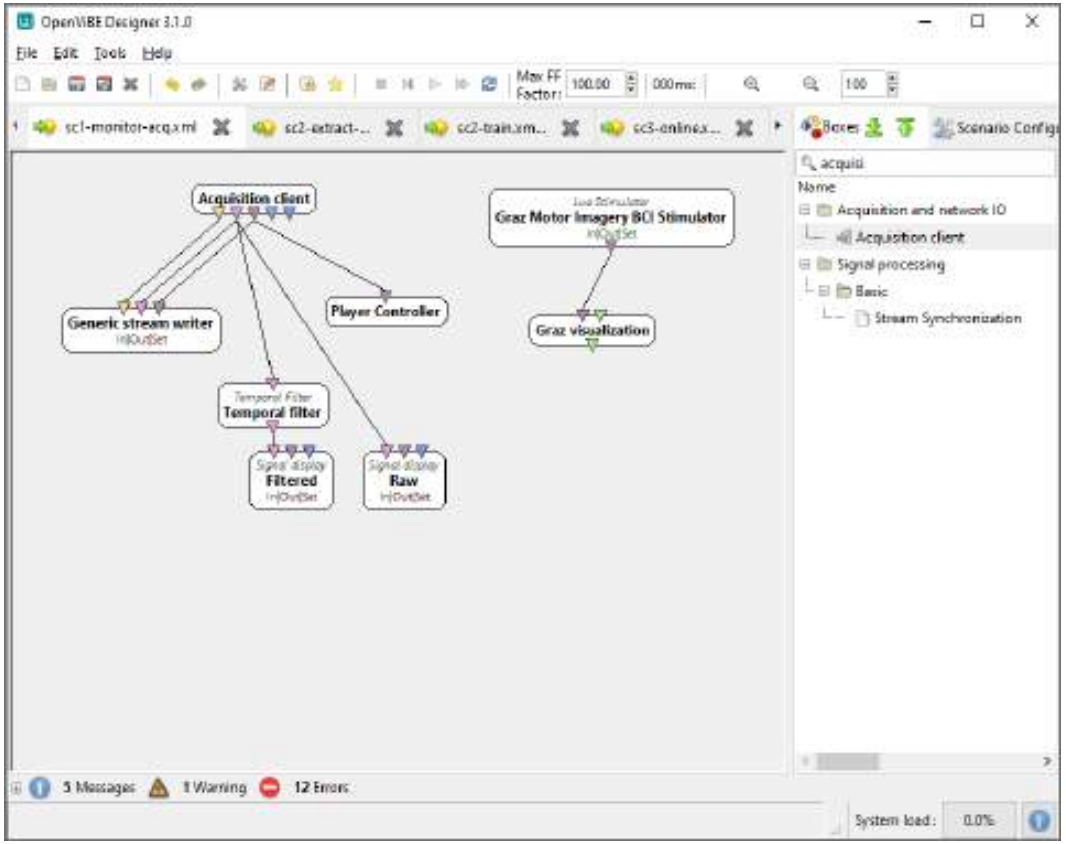
# SC1 - STEP 3 – PROTOCOL MANAGEMENT & STIMULATIONS…

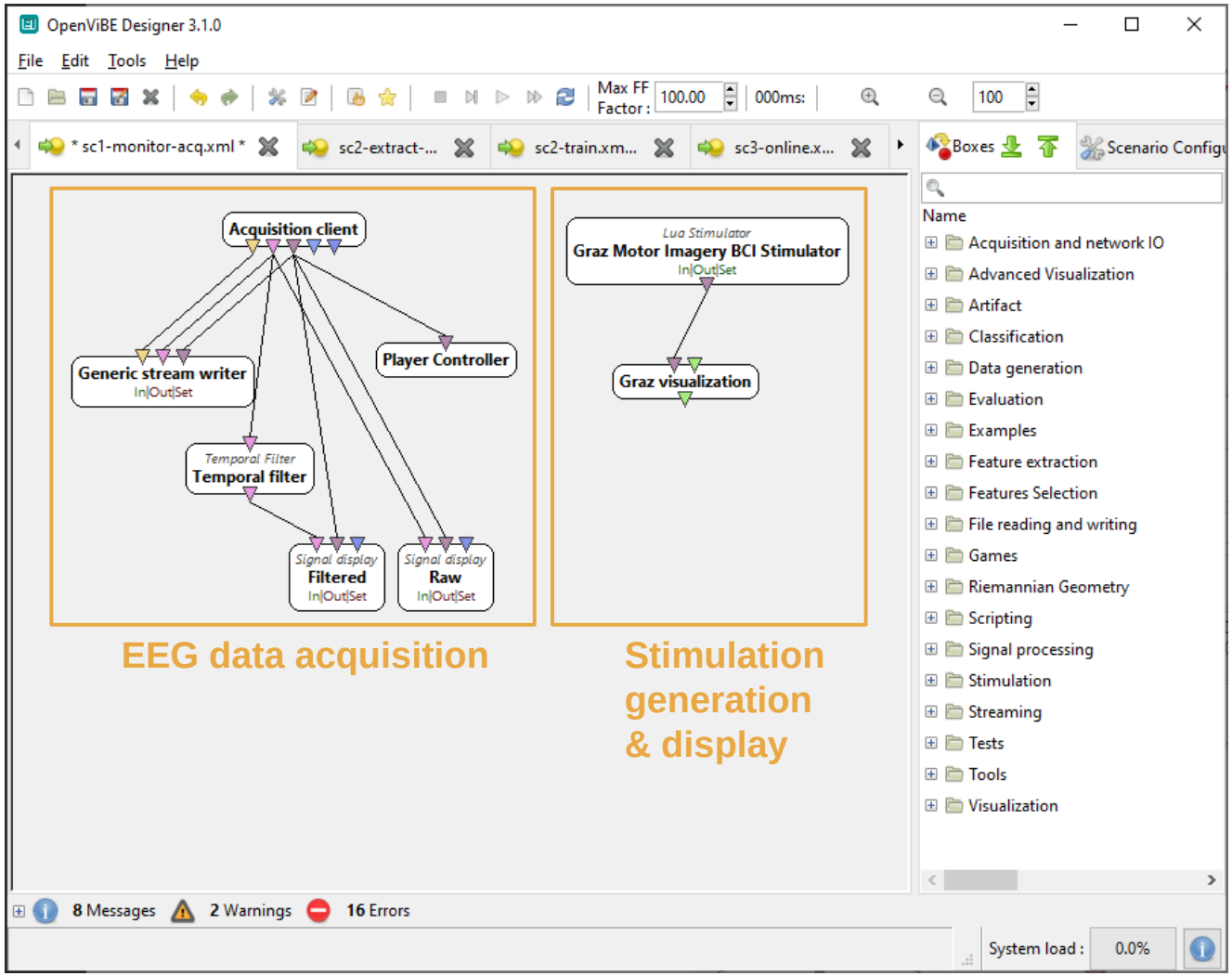
- Goal : understand data acquisition & synchro. with stimulations

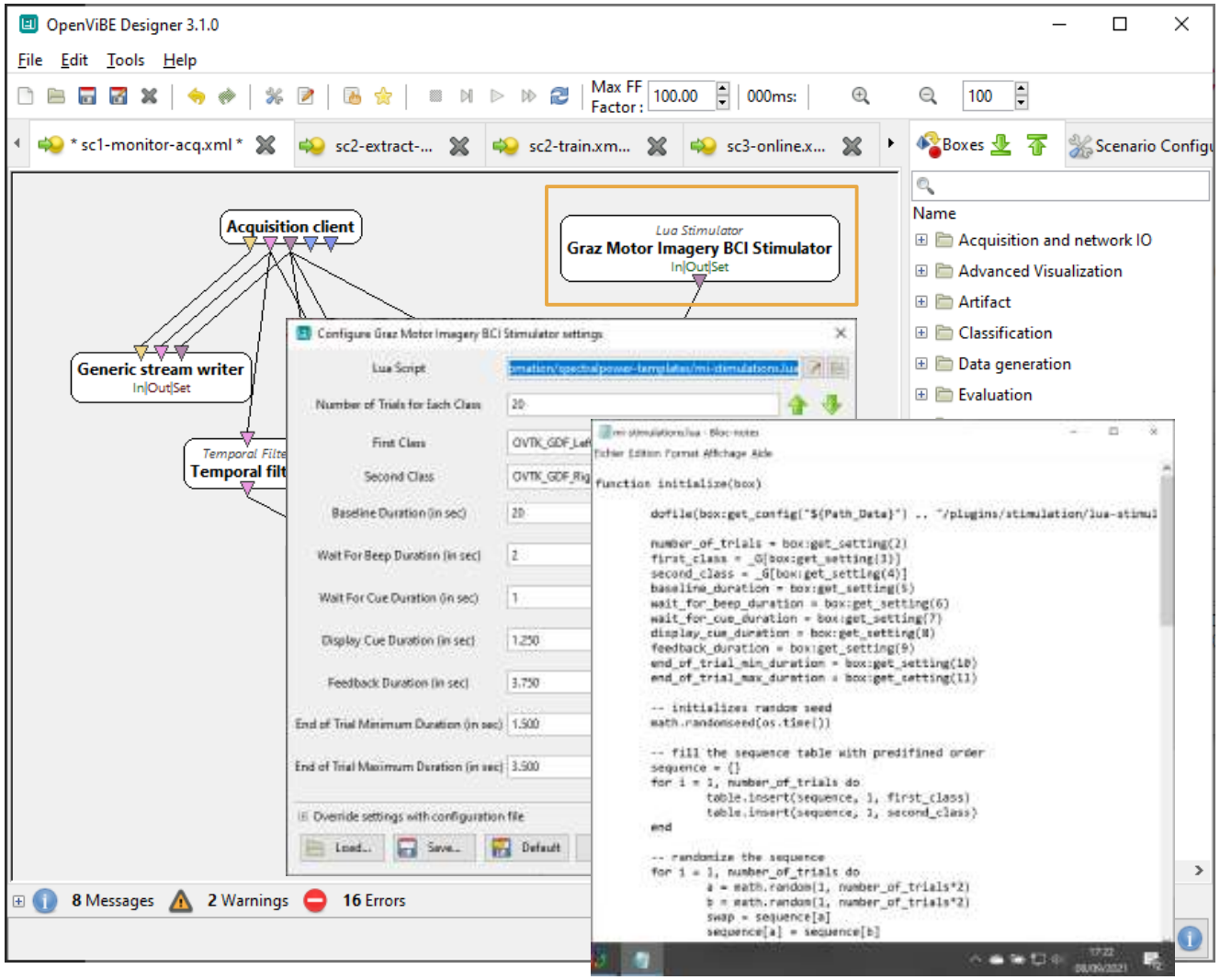
- Load scenario:

  **BCI-OpenViBE-CuttingEEG2021/ scenarios/sc1-monitor-acq.xml**

  (github)



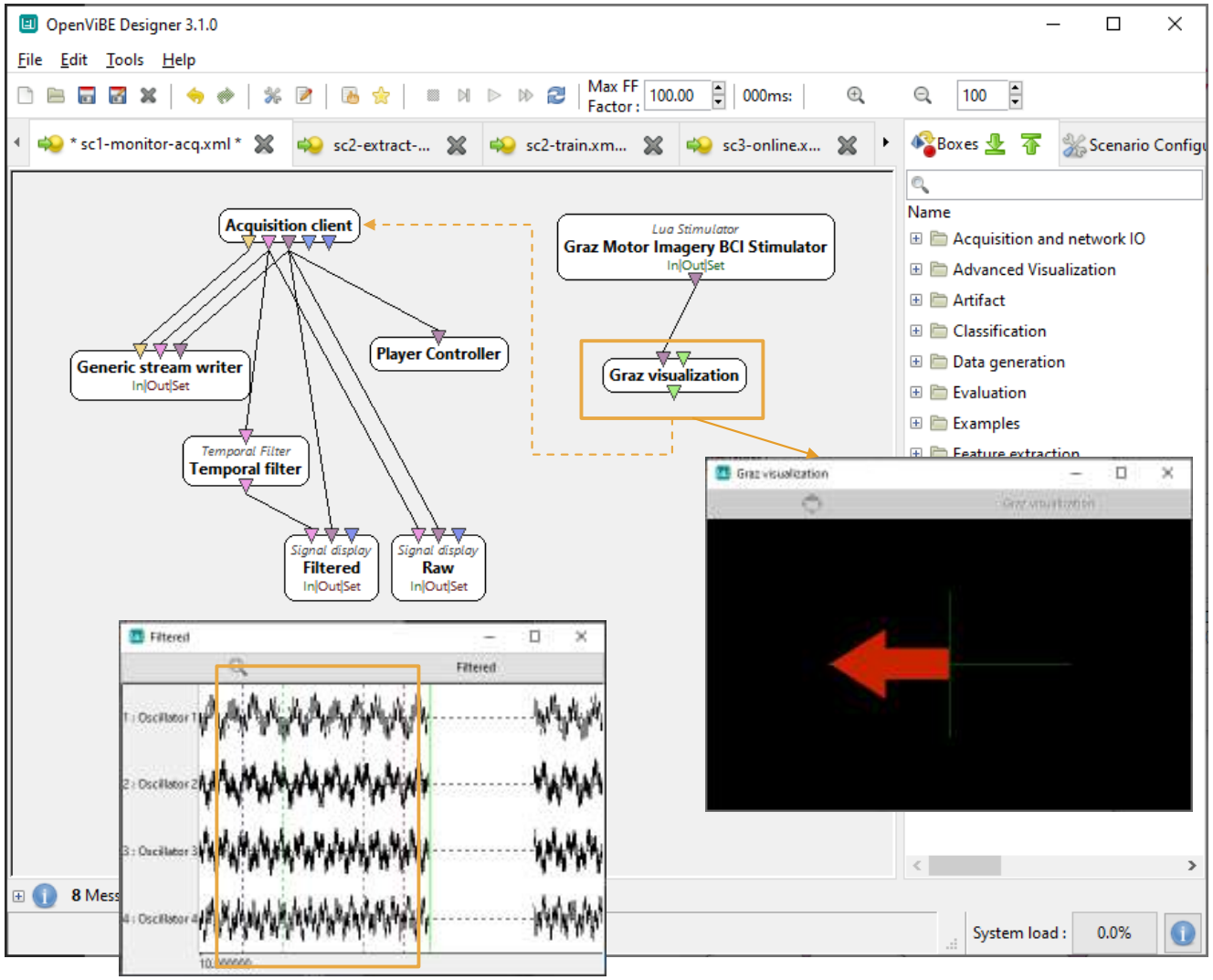

OpenViBE Designer 3.1.0
File Edit Tools Help
Max FF Factor : 100.00
000ms:
100
* sc1-monitor-acq.xml *
sc2-extract-...
sc2-train.xm...
sc3-online.x...
Acquisition client
Generic stream writer
In|Out|Set
Player Controller
Temporal Filter
Temporal filter
Signal display
Filtered
In|Out|Set
Signal display
Raw
In|Out|Set
Lua Stimulator
Graz Motor Imagery BCI Stimulator
In|Out|Set
Graz visualization
EEG data acquisition
Stimulation generation & display
Boxes
Scenario Configu
Name
Acquisition and network IO
Advanced Visualization
Artifact
Classification
Data generation
Evaluation
Examples
Feature extraction
Features Selection
File reading and writing
Games
Riemannian Geometry
Scripting
Signal processing
Stimulation
Streaming
Tests
Tools
Visualization
8 Messages
2 Warnings
16 Errors
System load :
0.0%




- **LUA Stimulator Box**
  (LUA = scripting language)

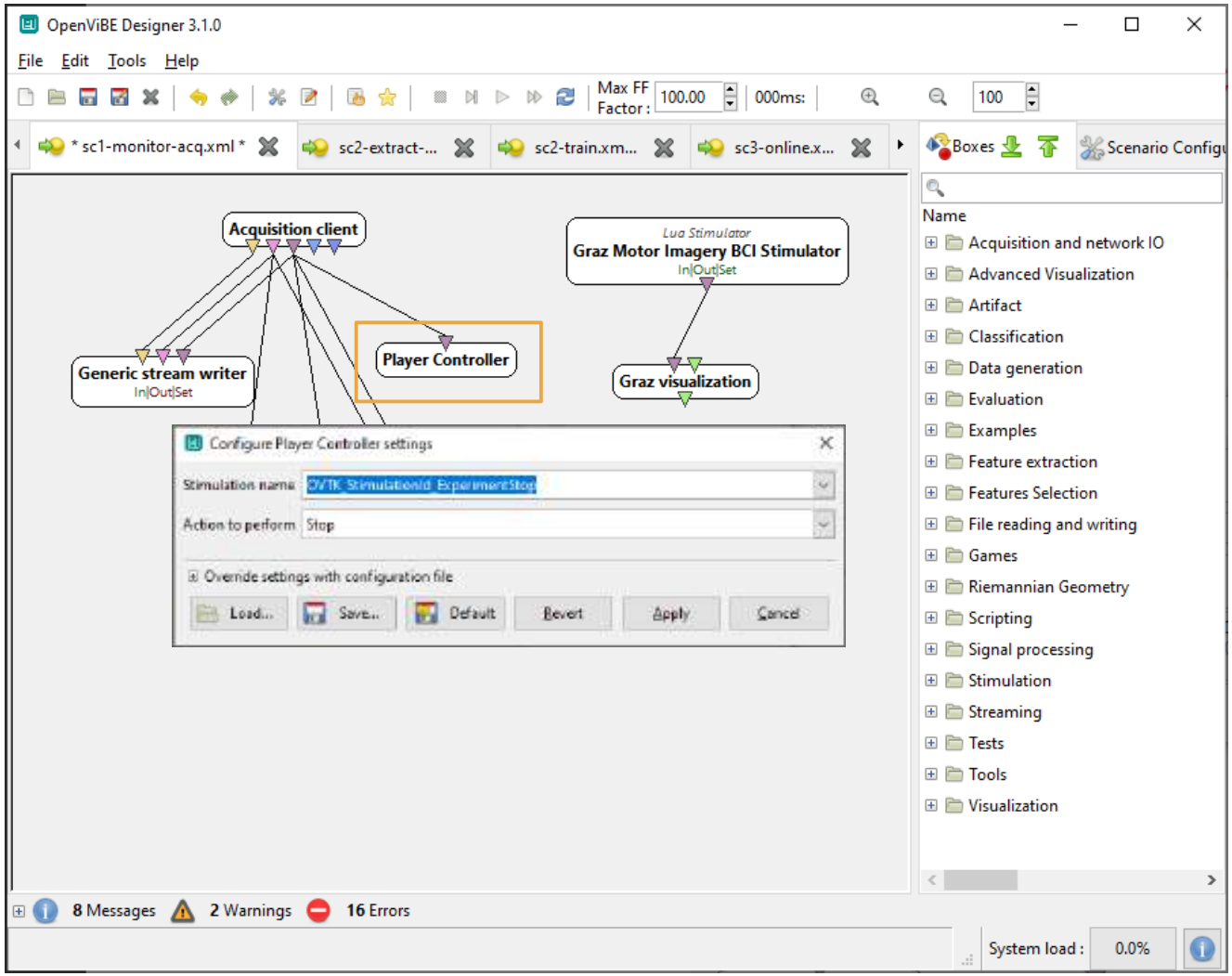
- **Experiment example:**
  Different stimulation/event codes at different times.
  Useful for signal segmentation ("epoching")

- **Stimulation label examples**:
  - Experiment Start/Stop
  - Trial Start/Stop
  - LEFT / RIGHT
  - Button pressed
  - etc

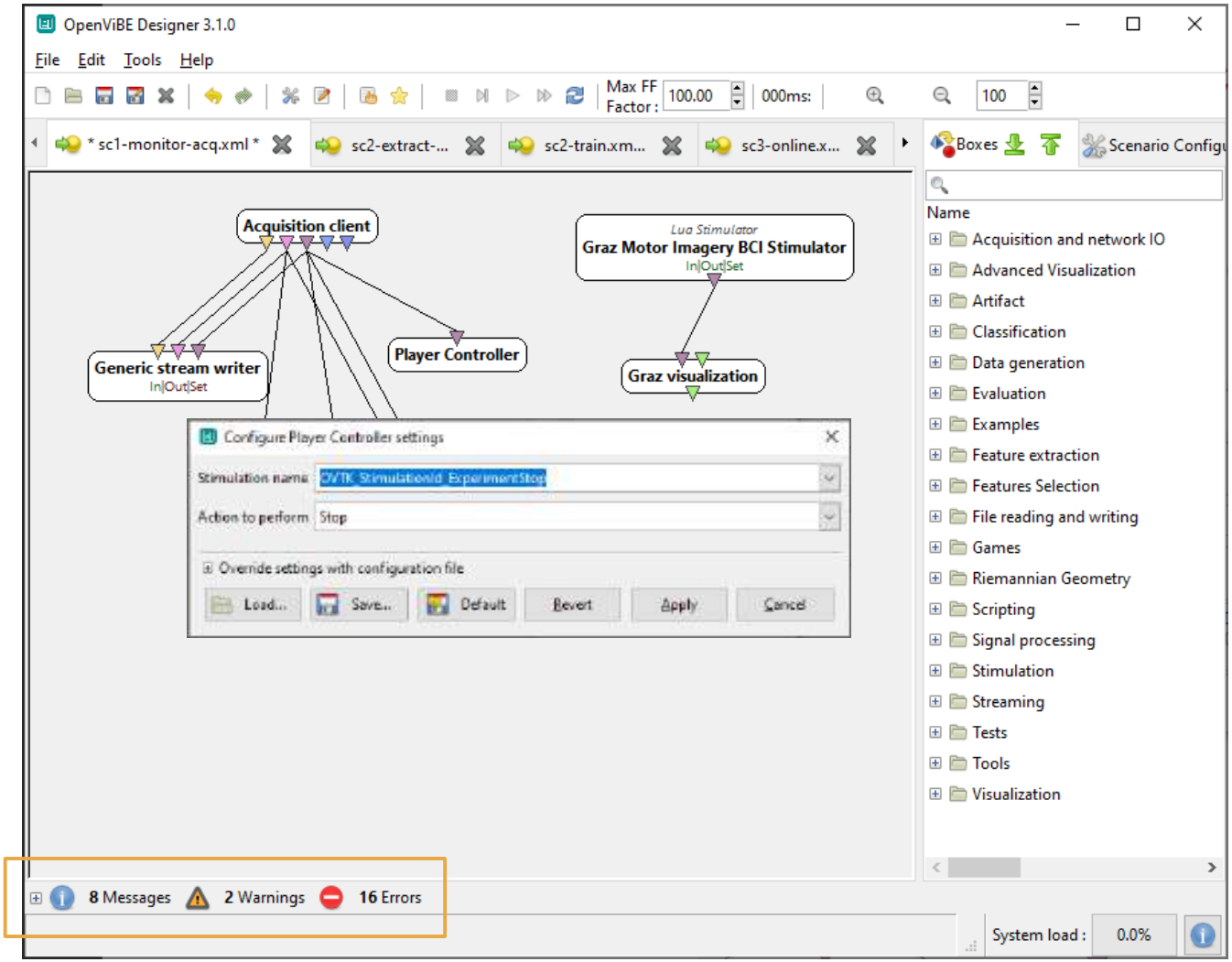



- **"Graz Visualization"** (specific for "Graz Protocol", based on Box "Display Cue Image")


- Displays specified image upon receiving specified stimulation code
- Transmits the **stimulation code & time** to the Acquisition Server
- **The stimulations are received by the ACQ Client, synchronized with the signal**




- **"Player Controller"**
  Orchestrates the experiment course, by applying an action upon receiving a stimulation




- Check the embedded log console if anything goes wrong!

# SC1 – FINAL WORDS / Q&A

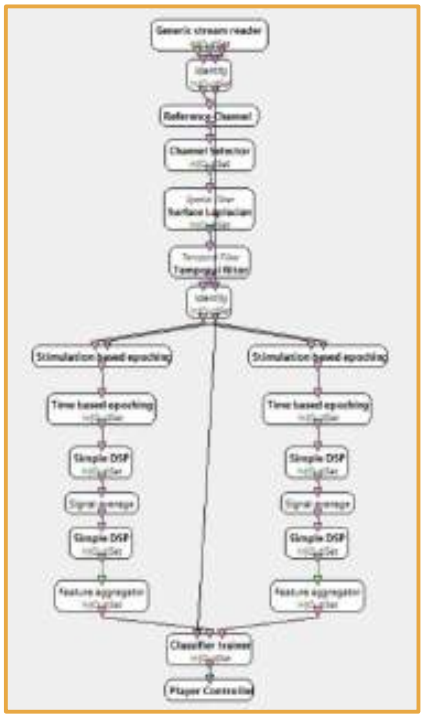
- **Recap:**
  - Simple I/O operations (signal loading, writing)
  - Signal Display
  - Acquisition Server / Client concept
  - Stimulations
  - Typical BCI protocol acquisition scenario

- **Note: from now on, we will only be using pre-generated/recorded signals**

# SC2 – CLASSIFIER TRAINING




1- Data acquisition

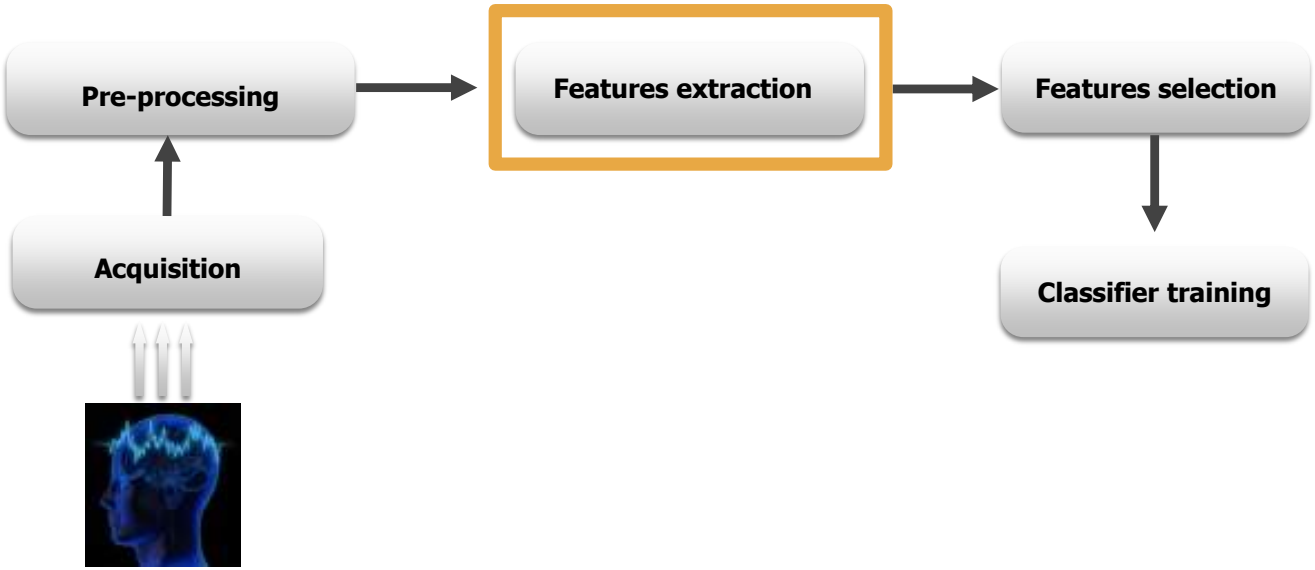



**2- Features extraction & Classification**



3- Online feedback

# SC2 – CLASSIFIER TRAINING

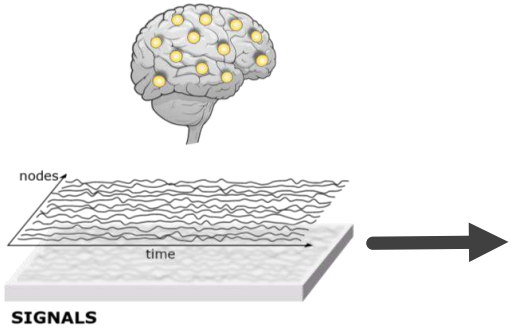



*Adapted from (X. Navarro & F. Grosselin)*

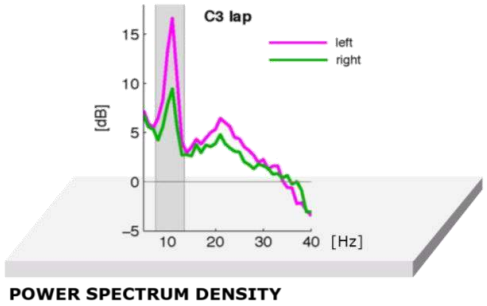
# STEP 2 – SIGNAL PROCESSING & VISUALIZATION

## Features to extract (recap)

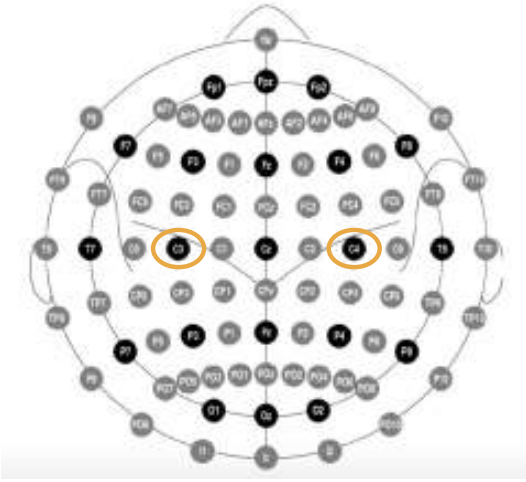
- Power spectra
- Sensorimotor area
- Mu: 8-12Hz &/OR Beta: 14-29Hz




(Gonzalez-Astudillo et al, 2020)




(Maeder et al., 2012)



# SC2 – CLASSIFIER TRAINING

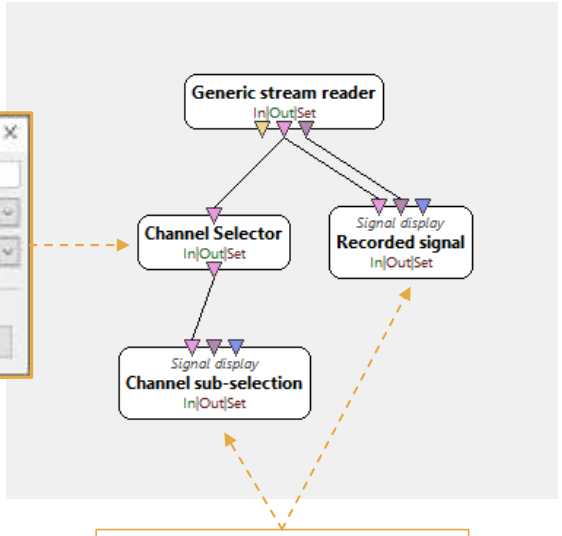
- **Sub-chapters:**
  - Channel Selection
  - Basic Signal Processing / Spectral analysis
  - Topography visualization
  - Feature Selection + Spatial Filtering
  - Classifier training

# SC2 – STEP 1 - CHANNEL SELECTION

- **Sub-chapters:**
  - Channel Selection
  - Basic Signal Processing / Spectral analysis
  - Topography visualization
  - Feature Selection + Spatial Filtering
  - Classifier training

# SC2 – STEP 1 - CHANNEL SELECTION

- **OpenViBE** can manage multiple signal streams in parallel
- In our example, **1 channel = 1 EEG electrode**
- We will load a pre-recorded signal file, that used 11 labeled electrodes and consider only a sub-set of those electrodes.


Note:
Right click + Rename can be useful when designing a scenario…

Display windows will use « renamed » label

- 1- Import a "Generic stream reader" box, and set the filename to:

  *<openvibe-3.1.0-64bit>\share\openvibe\scenarios\signals* ***bci-motor-imagery.ov***

- (optional – have a look at the recorded signals with "**Signal Display**")

- 2- Import a "**Channel Selector**" & link the boxes
- 3- Set the selection to :

  C3;C4;FC3;FC4;C5;C1;
  C2;C6;CP3;CP4

- 4- Display the output

# SC2 – STEP 2 – SIGNAL PROCESSING & SPECTRAL ANALYSIS

- **Sub-chapters:**
  - Channel Selection
  - Basic Signal Processing / Spectral analysis
  - Topography visualization
  - Feature Selection + Spatial Filtering
  - Classifier training

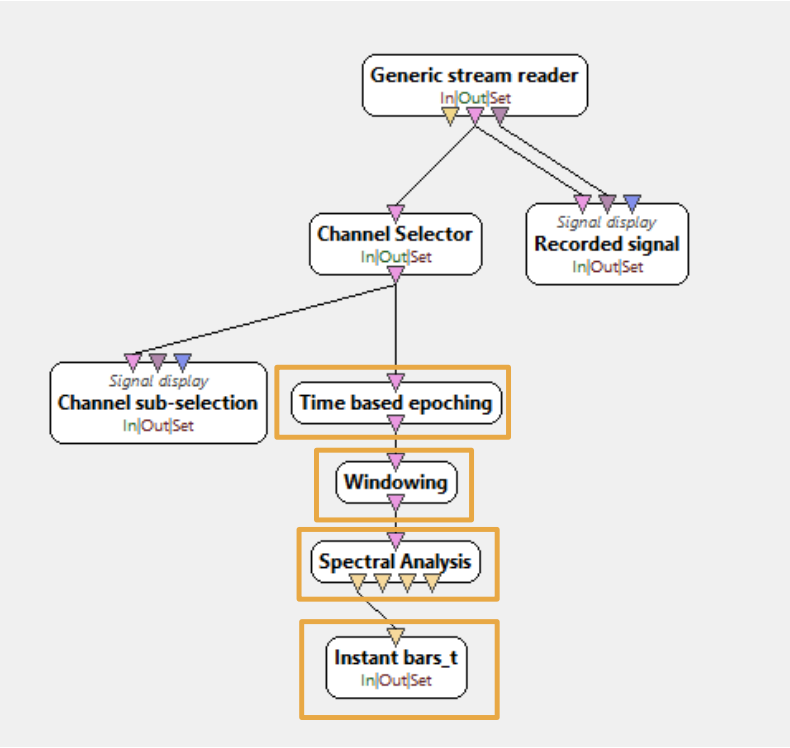
# SC2 – STEP 2 – SIGNAL PROCESSING & SPECTRAL ANALYSIS

- **Goals:**

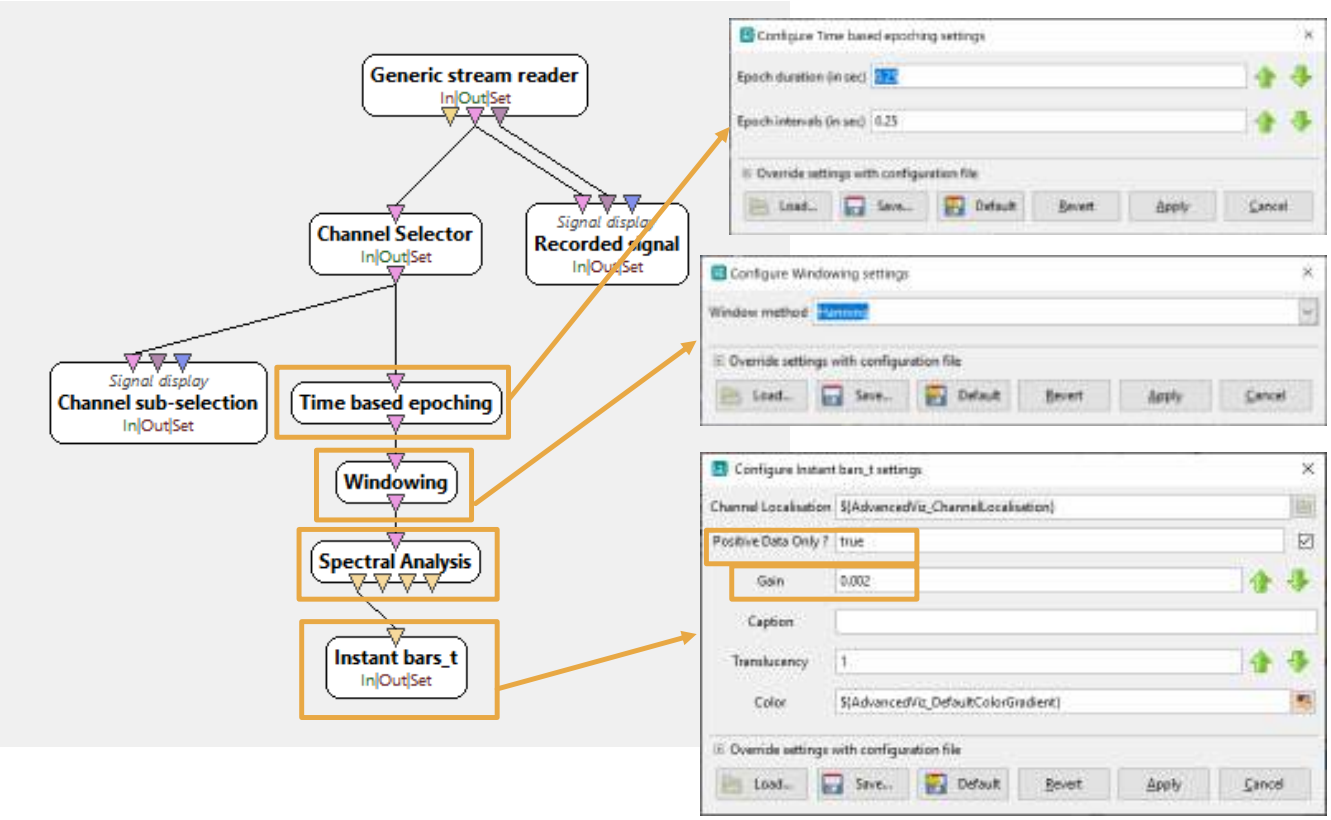
  Discover & use filtering, epoching, windowing tools

  Compute & display spectra for each channel/electrodes:

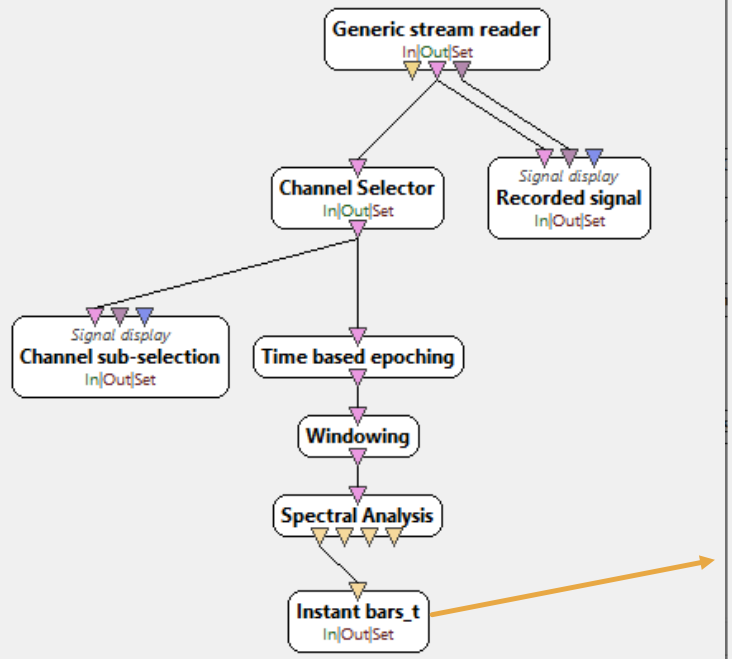
  - for the whole band
  - for different frequency bands (alpha, beta, etc)

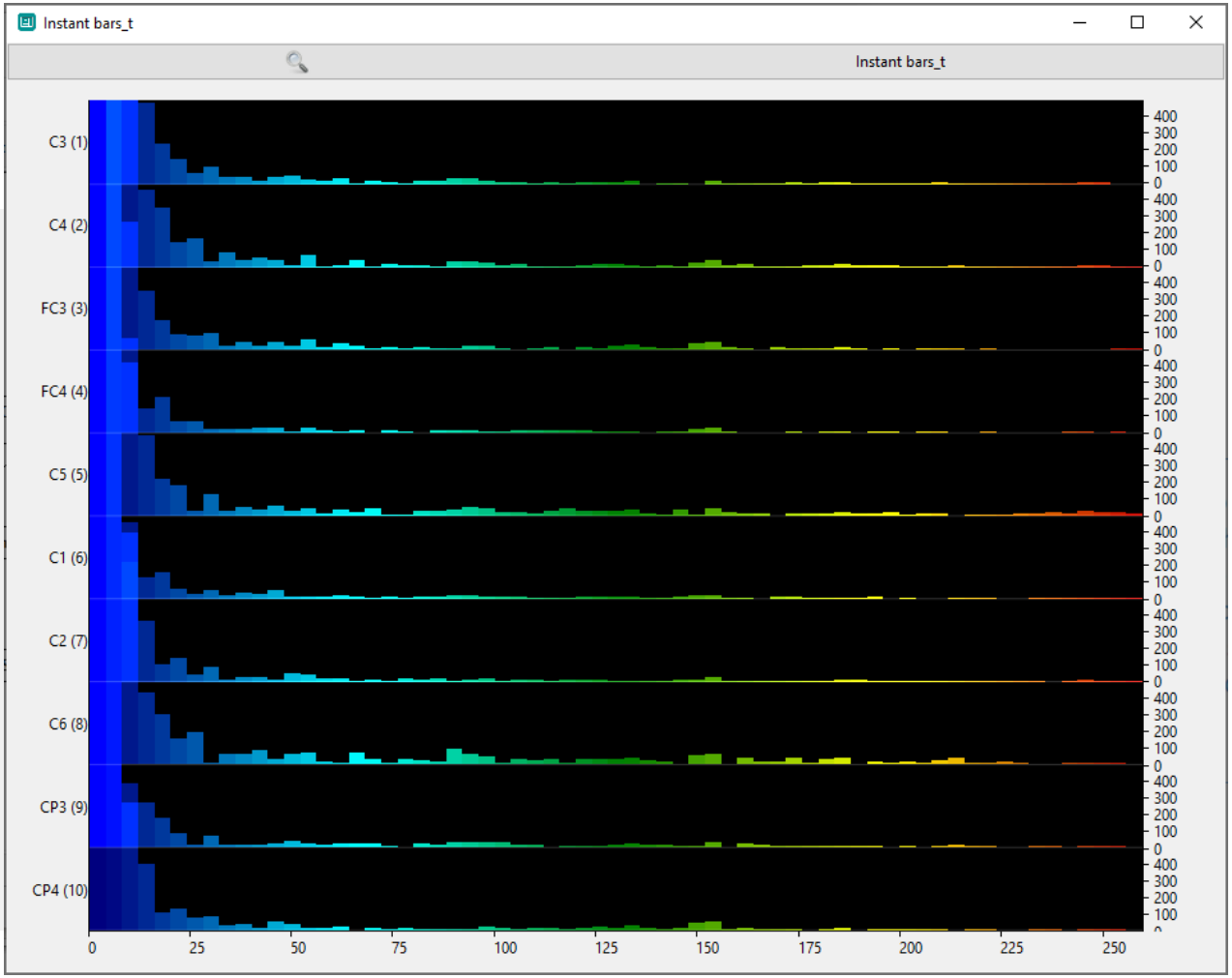



- 1- Import:
  "**Time Based Epoching**",
  "**Windowing**",
  "**Spectral Analysis**",
  "**Instant bars**"

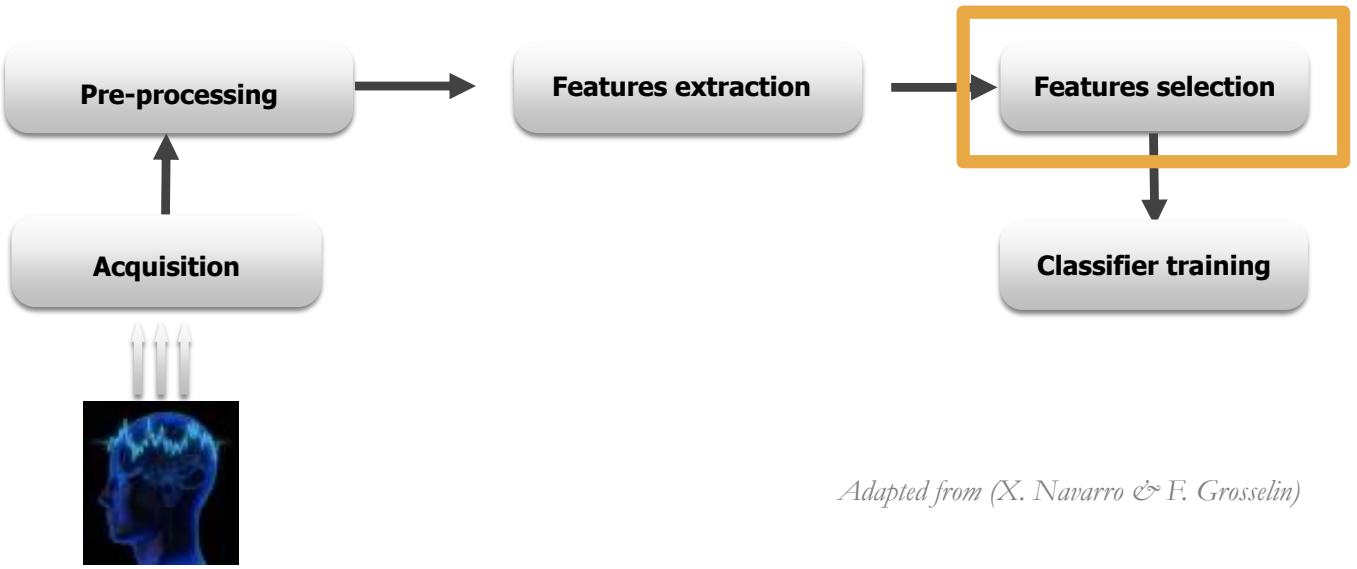



- 1- Import:
  "**Time Based Epoching**",
  "**Windowing**",
  "**Spectral Analysis**",
  "**Instant bars**"
- 2- Set preferred settings for Epoching and windowing (ex: 0,25s, no overlap, Hann Window)
- Note : Spectral Analysis has 4 outputs (see parameters for details).
- 3- Set gain for "instant bars" ! No automatic scaling…


Generic stream reader
In|Out|Set
Channel Selector
In|Out|Set
Signal display
Recorded signal
In|Out|Set
Signal display
Channel sub-selection
In|Out|Set
Time based epoching
Windowing
Spectral Analysis
Instant bars_t
In|Out|Set

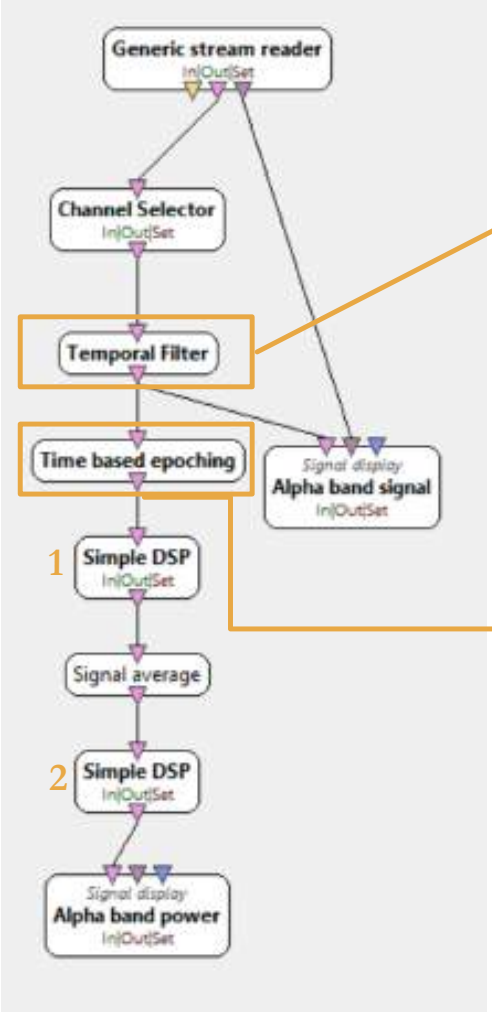

Instant bars_t
Instant bars_t
C3 (1)
C4 (2)
FC3 (3)
FC4 (4)
C5 (5)
C1 (6)
C2 (7)
C6 (8)
CP3 (9)
CP4 (10)
0
25
50
75
100
125
150
175
200
225
250
400
300
200
100
0

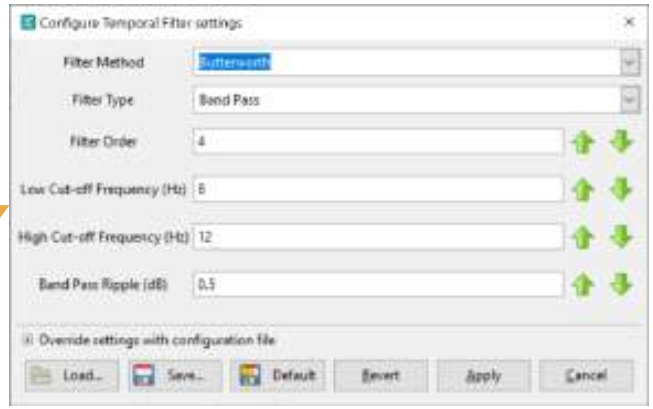
- We now have access to a view of the full band spectrum
- However, the feature we need for training the classifier is the **spectral power in the alpha/beta band**

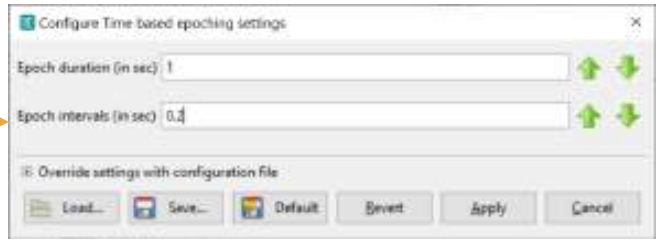



Manually or automatically

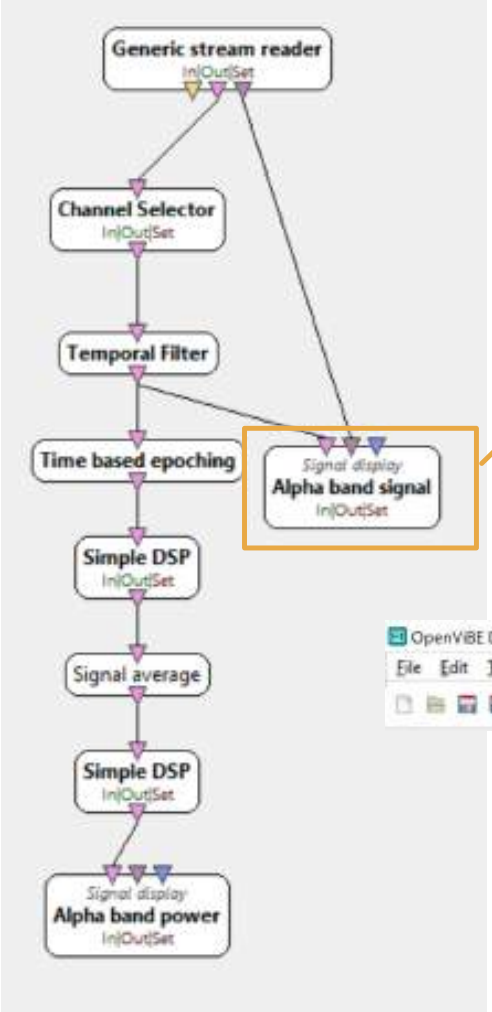
*Adapted from (X. Navarro & F. Grosselin)*

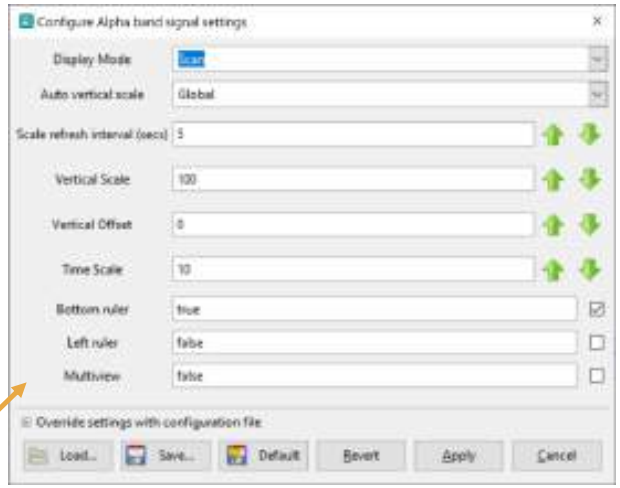


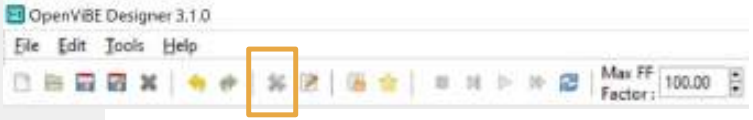



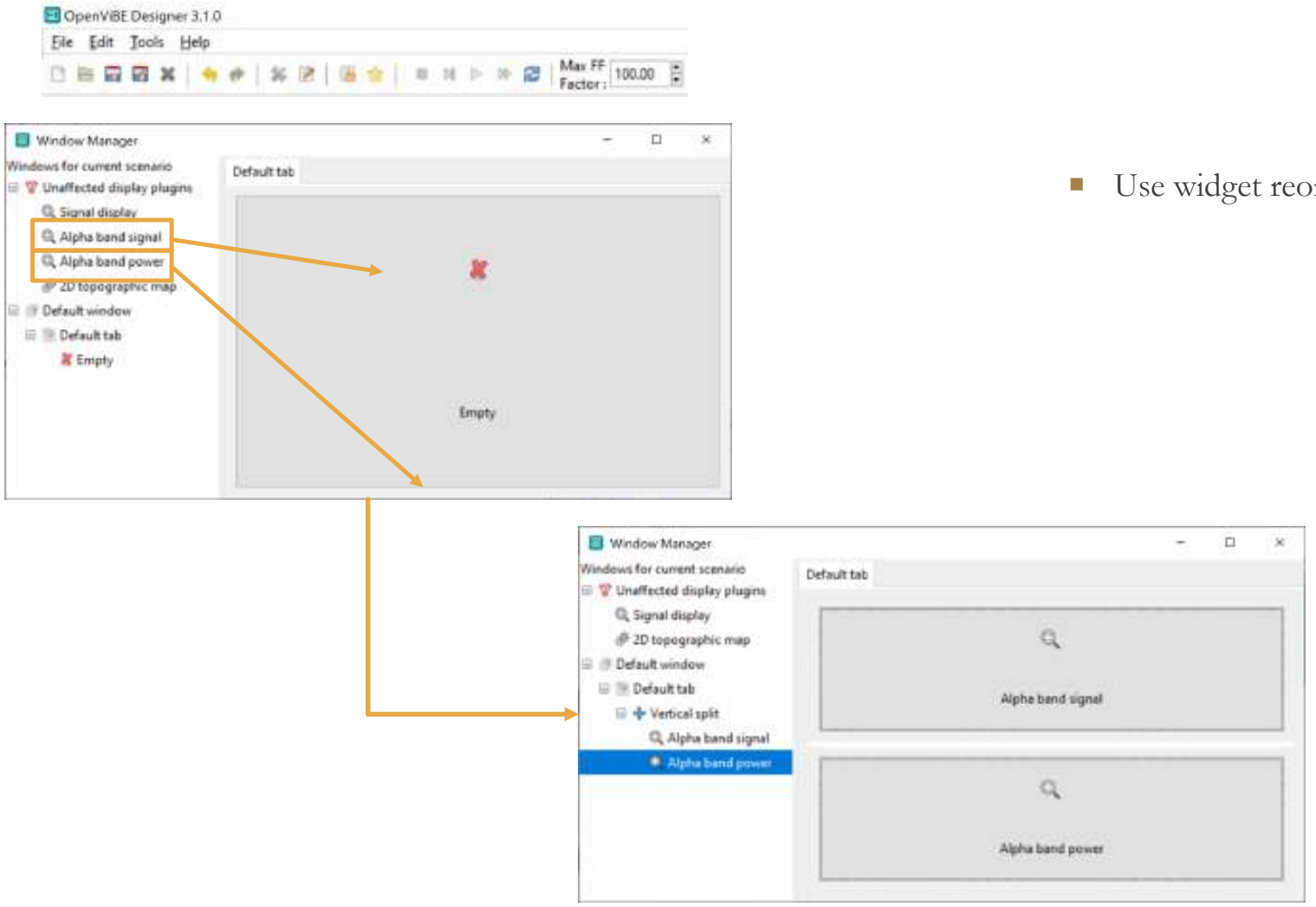

- "**Temporal Filter**", set to [8;12] Hz band
- Signal processing chain:
  - "**Time based epoching**" of 1s, every 0.2s (overlapping windows of signal for power computation)
  - **DSP 1** formula: x*x
  - **Average** ( = of $x^2$ across a window of 1s)
  - **DSP 2** formula: Log10(1+x)
  - => log10(1+avg($x^2$))
- Add **Displays** and rename them
  - Set time scales & vertical scaling



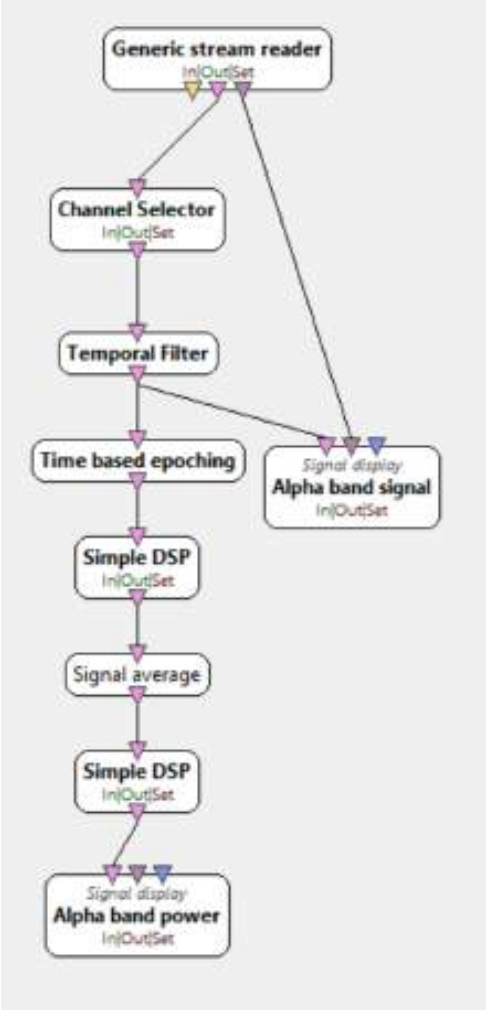



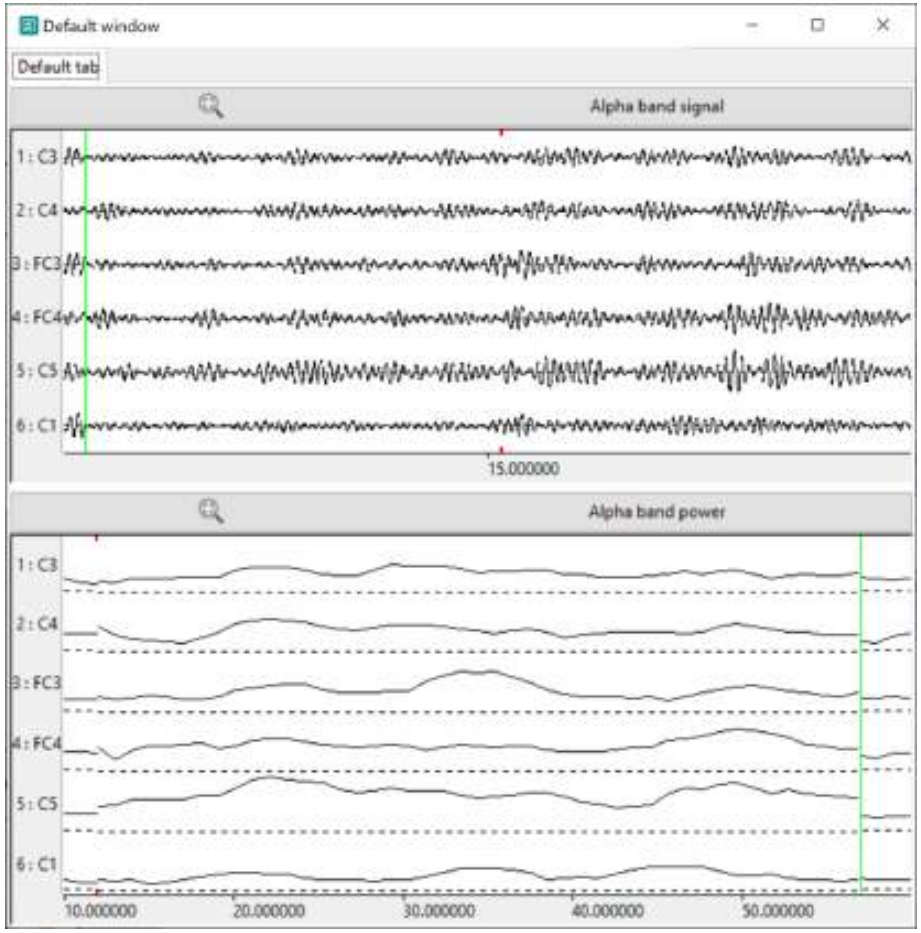



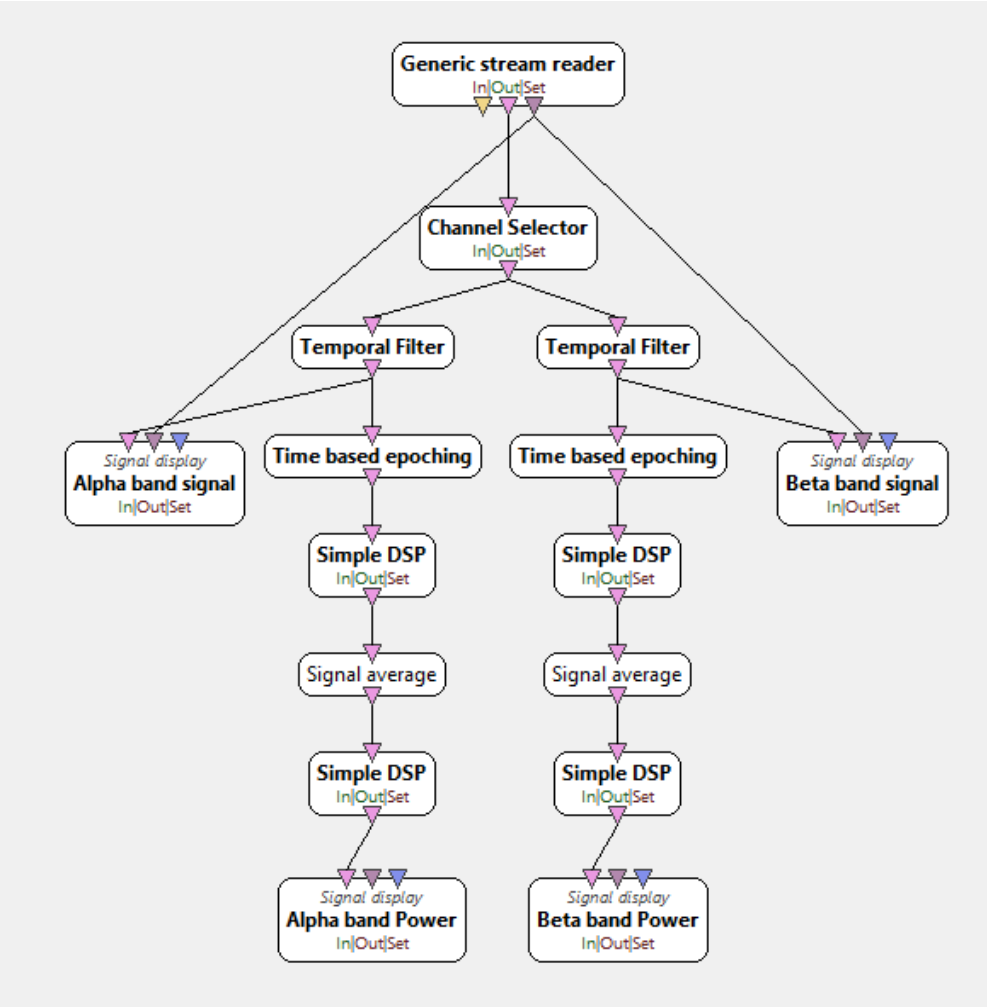

- Add **Displays** and rename them
  - Set time scales : 10 (seconds) for alpha band signal display, 50 for alpha band power (since we have 5 times more data to display)
  - Set vertical scaling to "global"
- Use widget reordering tool for ease of use!



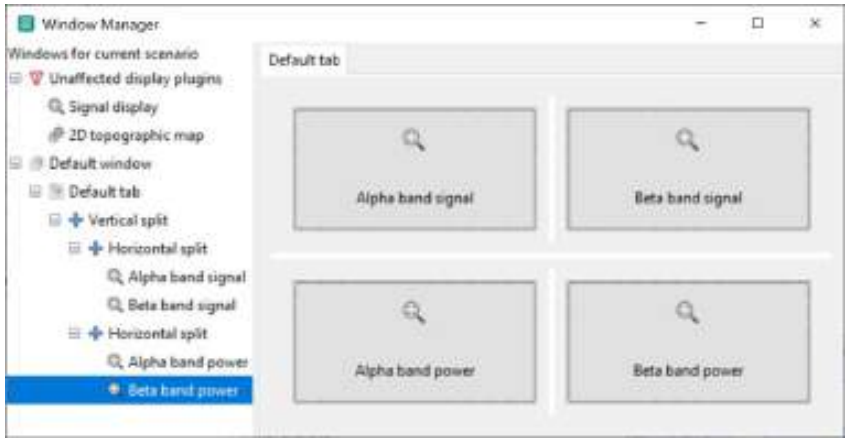

- Use widget reordering tool for ease of use!

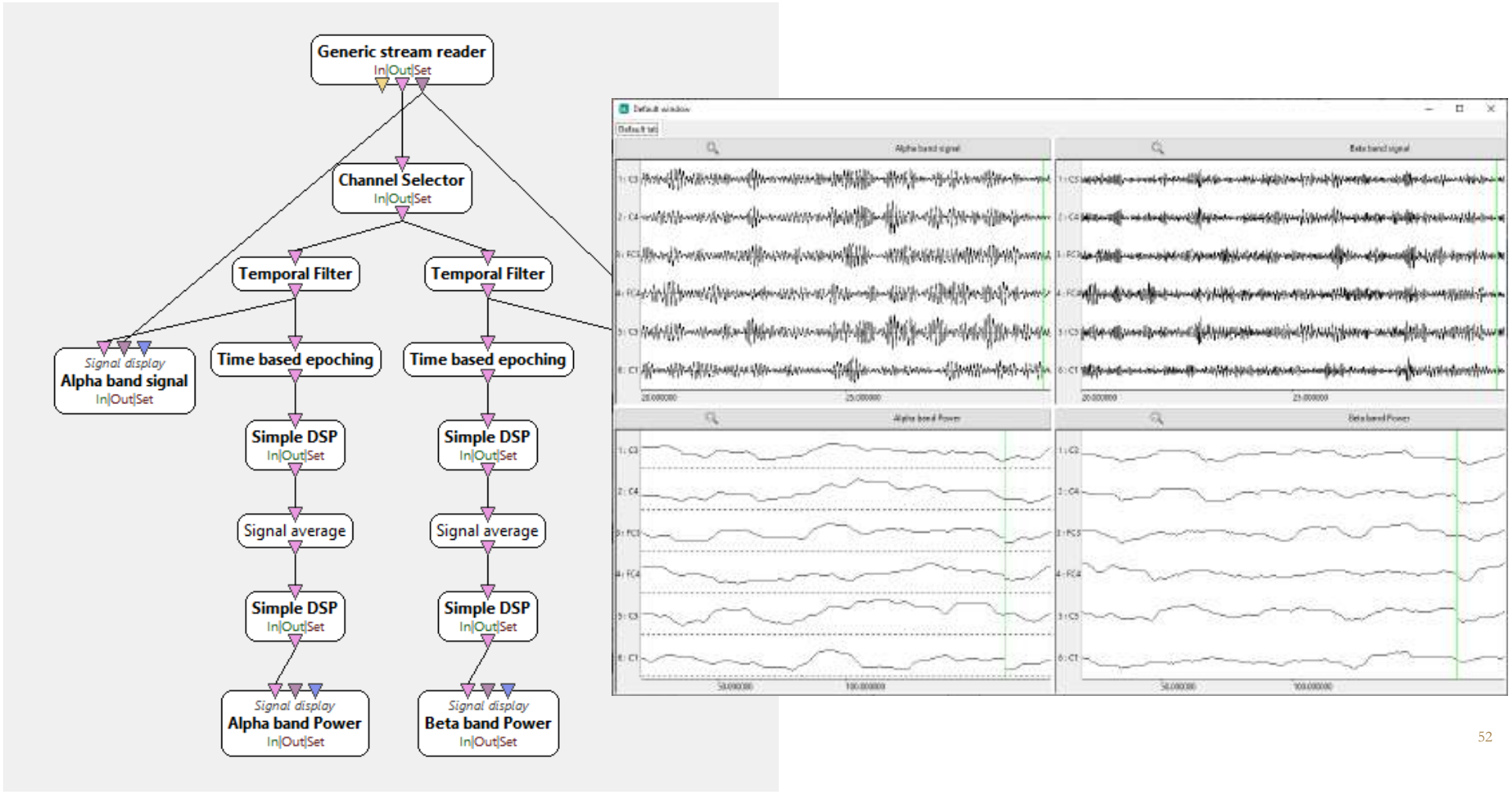

Generic stream reader
In|Out|Set
Channel Selector
In|Out|Set
Temporal Filter
In|Out|Set
Time based epoching
In|Out|Set
Signal display
Alpha band signal
In|Out|Set
Simple DSP
In|Out|Set
Signal average
In|Out|Set
Simple DSP
In|Out|Set
Signal display
Alpha band power
In|Out|Set

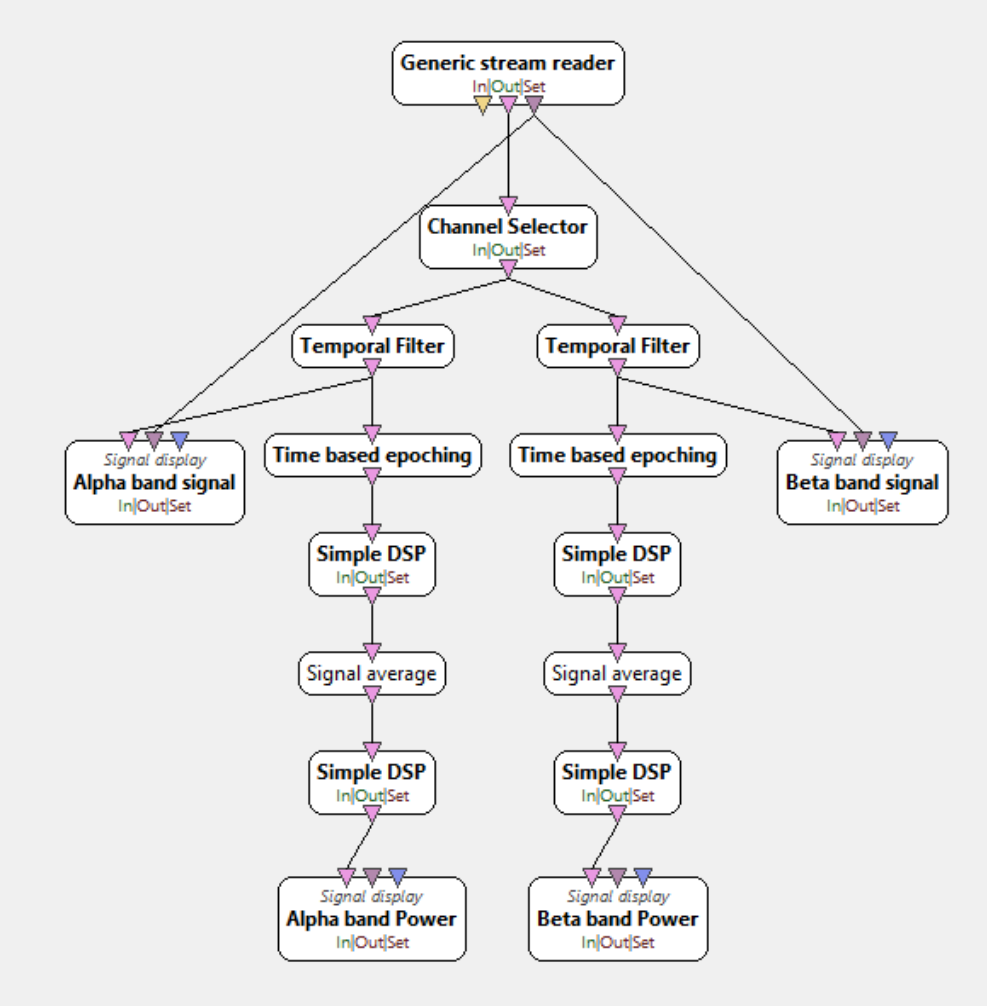

Default window
Default tab
Alpha band signal
1 : C3
2 : C4
3 : FC3
4 : FC4
5 : C5
6 : C1
15.000000
Alpha band power
1 : C3
2 : C4
3 : FC3
4 : FC4
5 : C5
6 : C1
10.000000
20.000000
30.000000
40.000000
50.000000




- Repeat for **Beta Band** [12;24]Hz
- Don't hesitate to copy/paste boxes…!




Generic stream reader
In|Out|Set
Channel Selector
In|Out|Set
Temporal Filter
Temporal Filter
Signal display
Alpha band signal
In|Out|Set
Time based epoching
Time based epoching
Simple DSP
In|Out|Set
Simple DSP
In|Out|Set
Signal average
Signal average
Simple DSP
In|Out|Set
Simple DSP
In|Out|Set
Signal display
Alpha band Power
In|Out|Set
Signal display
Beta band Power
In|Out|Set




- Scenario is available on the github repo:

BCI-OpenViBE-CuttingEEG2021/
scenarios/sc2-spectralAnalysis.xml

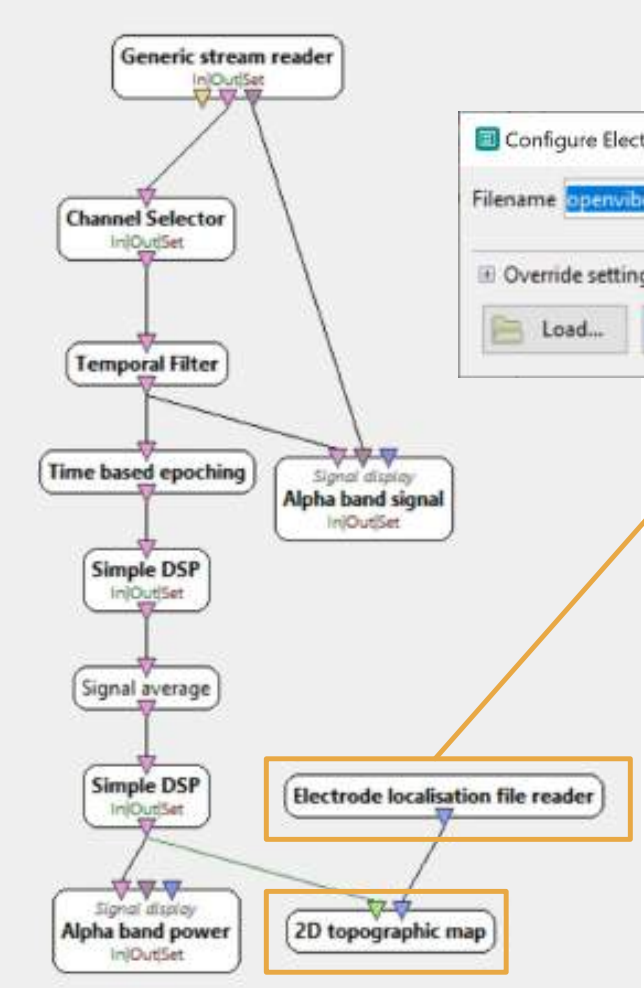
# SC2 – STEP 3 – VISUALIZING BRAIN TOPOGRAPHY

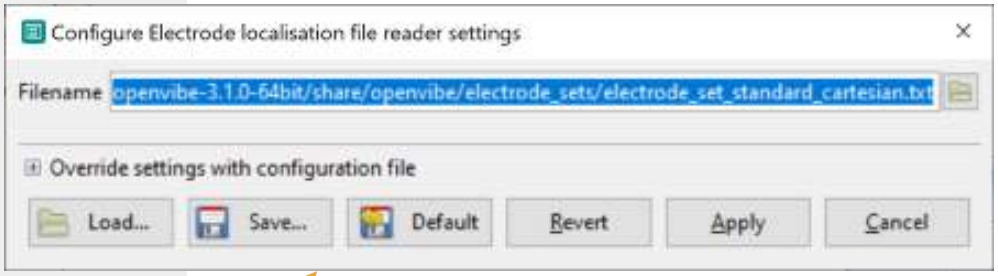
- **Sub-chapters:**
  - Channel Selection
  - Basic Signal Processing / Spectral analysis
  - Topography visualization
  - Feature Selection + Spatial Filtering
  - Classifier training

# SC2 – STEP 3 – VISUALIZING BRAIN TOPOGRAPHY

- **Goals:**

- Load electrode localization data
- Topography viewer for a specific band

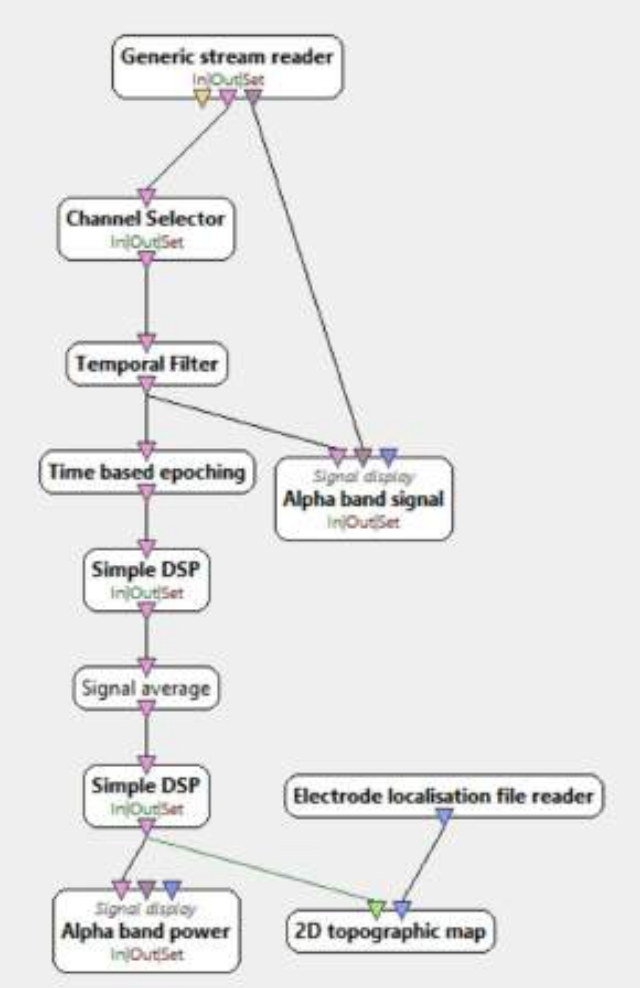


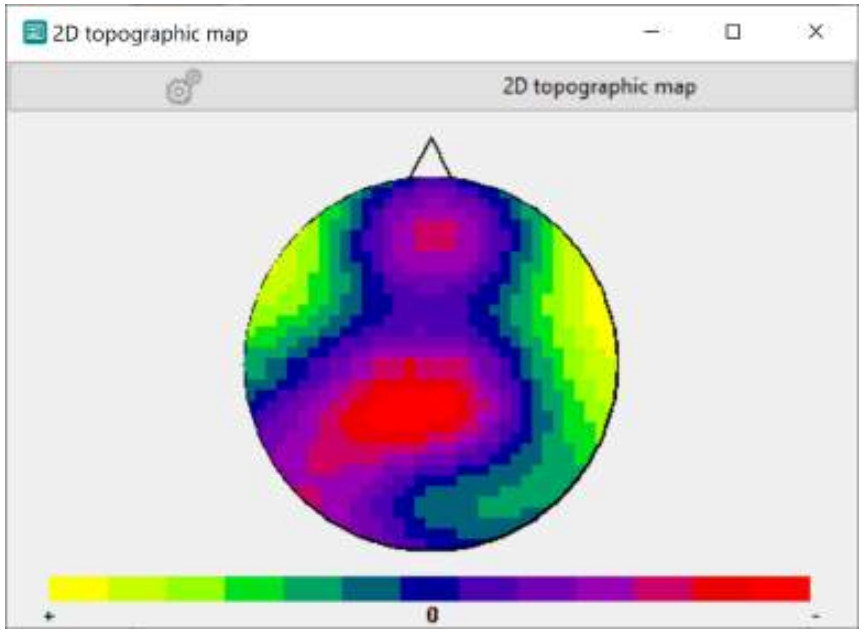




- Using the previous scenario computing the spectral power (in band alpha only):
- Add "**Electrode localization file reader**" and check the file location

**<openvibe-3,1,0-64bits>/ share/openvibe/electrode_sets/electrode_set_standard_cartesian.txt**

- Add "**2D topographic map**"
- Connect boxes and play the scenario


Generic stream reader
In|Out|Set
Channel Selector
In|Out|Set
Temporal Filter
Time based epoching
Signal display
Alpha band signal
In|Out|Set
Simple DSP
In|Out|Set
Signal average
Simple DSP
In|Out|Set
Electrode localisation file reader
Signal display
Alpha band power
In|Out|Set
2D topographic map


2D topographic map
2D topographic map
0

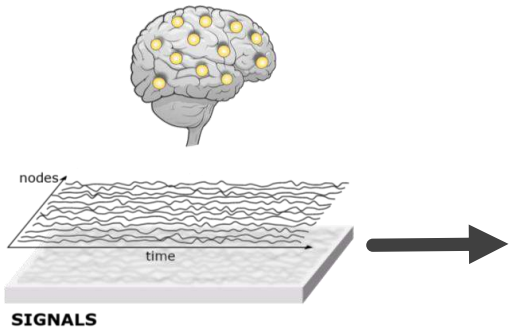
# SC2 – STEP 4 – FEATURE SELECTION

- **Sub-chapters:**
  - Channel Selection
  - Basic Signal Processing / Spectral analysis
  - Topography visualization
  - Feature Selection + Spatial Filtering
  - Classifier training

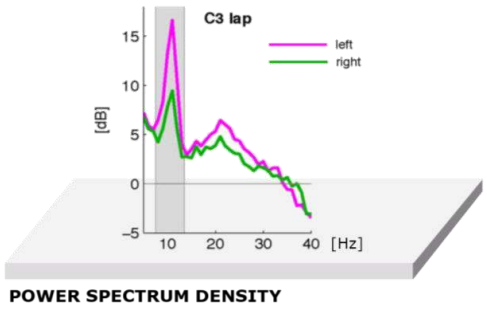
# SC2 – STEP 4 – FEATURE SELECTION

## Features to extract (recap)

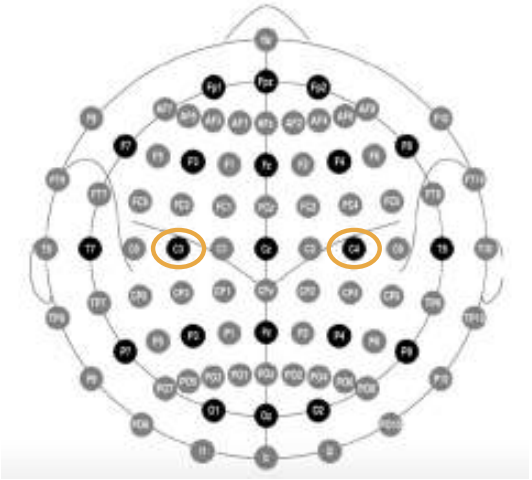
- Power spectra
- Sensorimotor area
- Mu: 8-12Hz &/OR Beta: 14-29Hz




(Gonzalez-Astudillo et al, 2020)



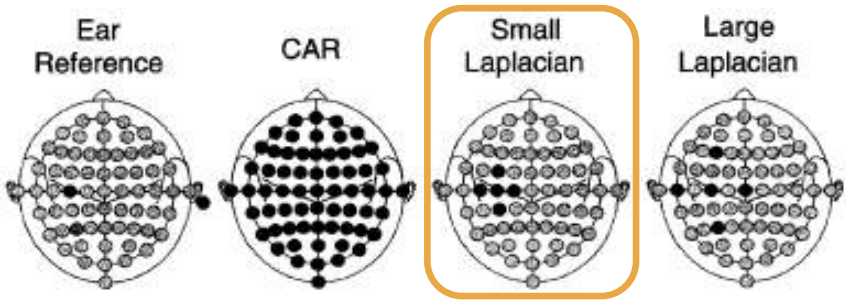

(Maeder et al., 2012)



# SC2 – STEP 4 – FEATURE SELECTION – SPATIAL FILTERING

- **Spatial Filtering**
  - Common Average Reference (CAR)
  - Surface Laplacian – used here

$C3' = 4*C3 - FC3 - C5 - C1 - CP3$
$C4' = 4*C4 - FC4 - C2 - C6 - CP4$




Spatial filtering
(McFarland et al, 1997)

- OV box: **Spatial Filter**
- **10 inputs => 2 outputs**

  C3' = 4*C3 - FC3 - C5 - C1 - CP3

  C4' = 4*C4 - FC4 - C2 - C6 - CP4
- **Our list of channels:**

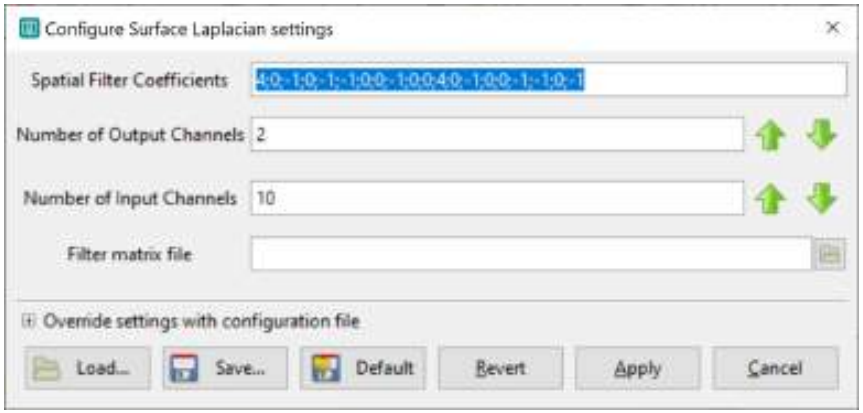
  C3; C4; FC3; FC4; C5; C1; C2; C6; CP3; CP4

C3' = 4 ; 0 ; -1 ; 0 ; -1; -1; 0; 0; -1 ; 0

C4' = 0 ; 4 ; 0 ; -1 ; 0 ; 0 ; -1; -1; 0 ; -1


- **Serialize formulae in box parameters**

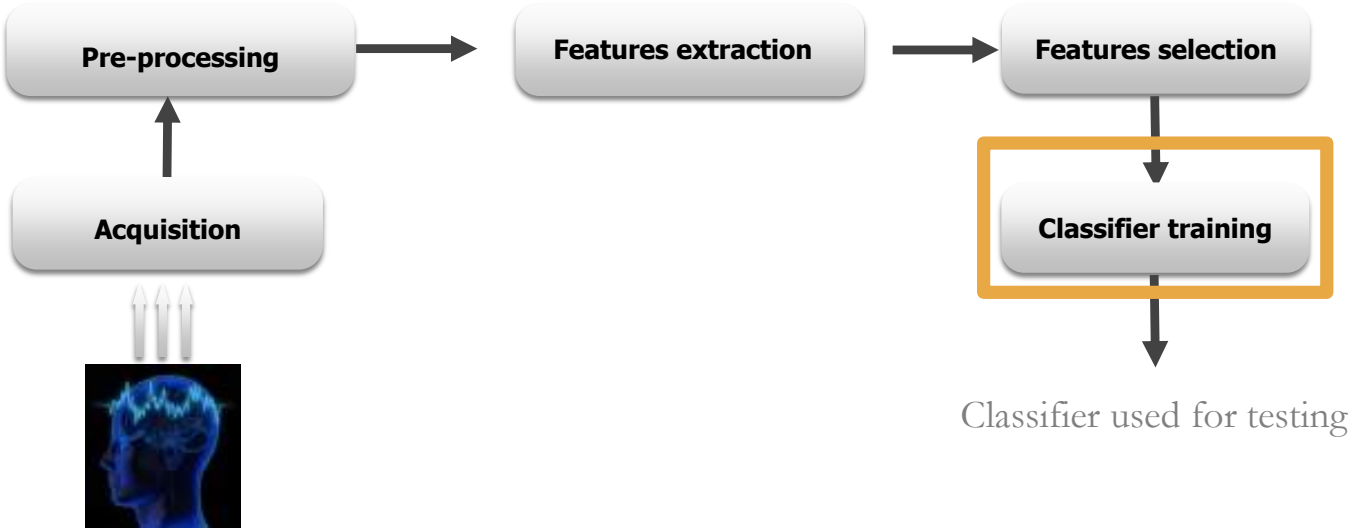
4;0;-1;0;-1;-1;0;0;-1;0;0;4;0;-1;0;0;-1;-1;0;-1



# SC2 – STEP 5 – CLASSIFIER TRAINING

- **Sub-chapters:**
  - Channel Selection
  - Basic Signal Processing / Spectral analysis
  - Topography visualization
  - Feature Selection + Spatial Filtering
  - Classifier training

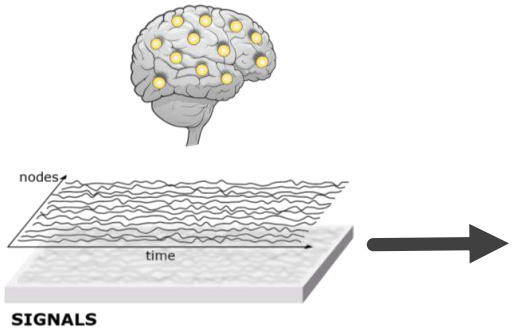
# SC2 – STEP 5 – CLASSIFIER TRAINING



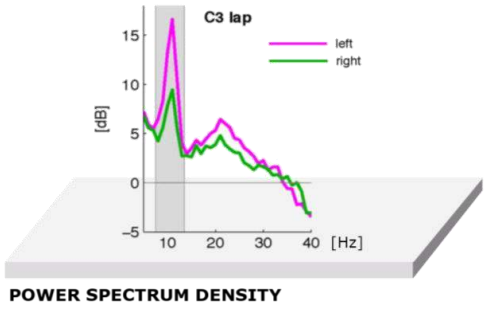

*Adapted from (X. Navarro & F. Grosselin)*

# SC2 – STEP 5 – CLASSIFIER TRAINING

- **Goal reminder:**
- We want to train a classifier to distinguish between mental states, based on instantaneous spectral power in bands alpha and/or beta




(Gonzalez-Astudillo et al, 2020)




(Maeder et al., 2012)

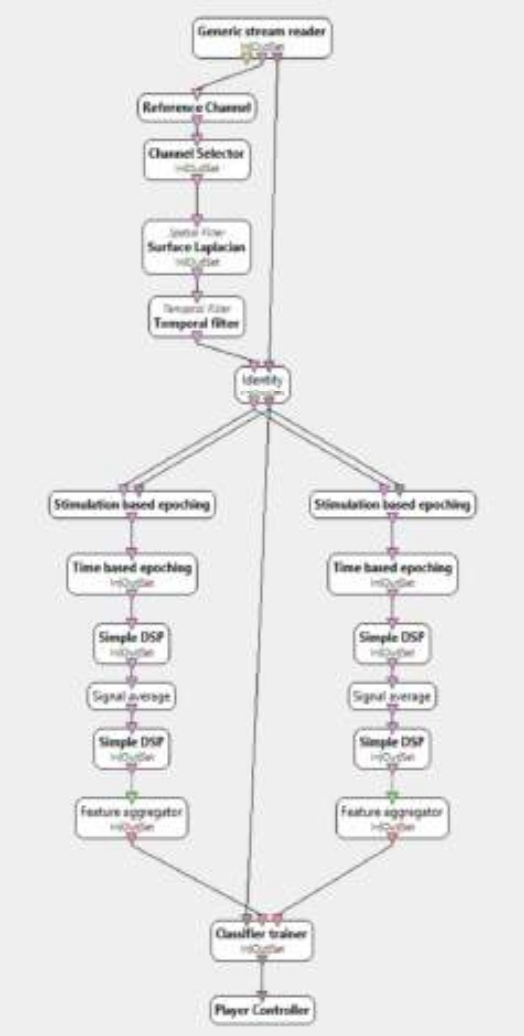
# SC2 – STEP 5 – CLASSIFIER TRAINING

- **Load the full scenario so we can analyze it step by step…**

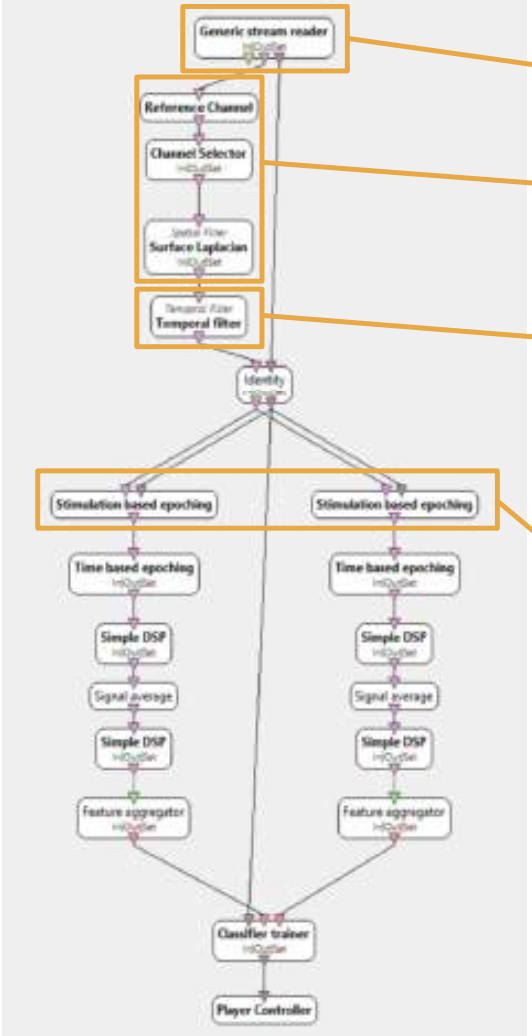
<OpenViBE_install_path>\
share\openvibe\scenarios\bci-examples\motor-imagery\
**motor-imagery-bci-2-classifier-trainer.xml**

# SC2 – STEP 5 – CLASSIFIER TRAINING

- **Goal: putting it all together in a BCI scenario…**
  - **Load** pre-recorded signal file
  - Select **sub-set of electrodes**
  - Surface Laplacian
  - Filter to the **frequency band** of interest
  - Stimulation-based Epoching: **split between conditions** (LEFT vs RIGHT…)
  - Epoching & Signal processing: compute a "**classifier feature**" based on **spectral power**
  - **Train the classifier!**

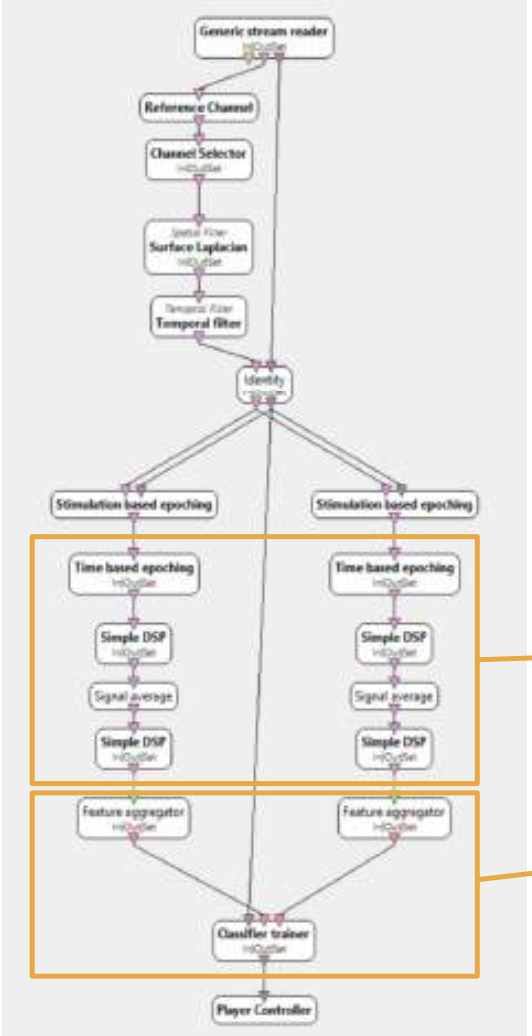



- **Load** pre-recorded signal file
- Select **sub-set of electrodes,** set **Reference Channel + apply spatial filtering**
- Filter to the **frequency band** of interest
- Stimulation-based Epoching: **split between conditions** (LEFT vs RIGHT…)
- Epoching & Signal processing: compute a "**classifier feature**" based on **spectral power**
- **Train the classifier!**

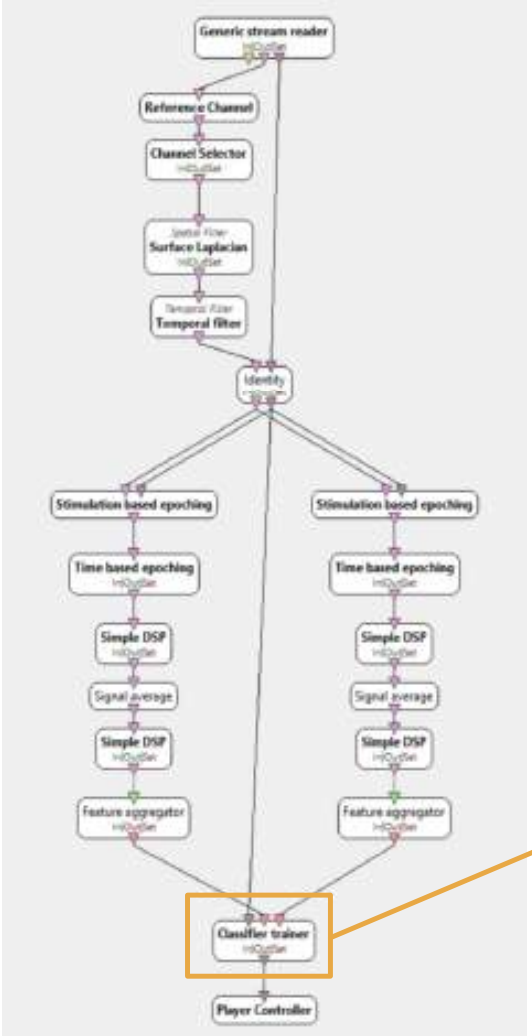


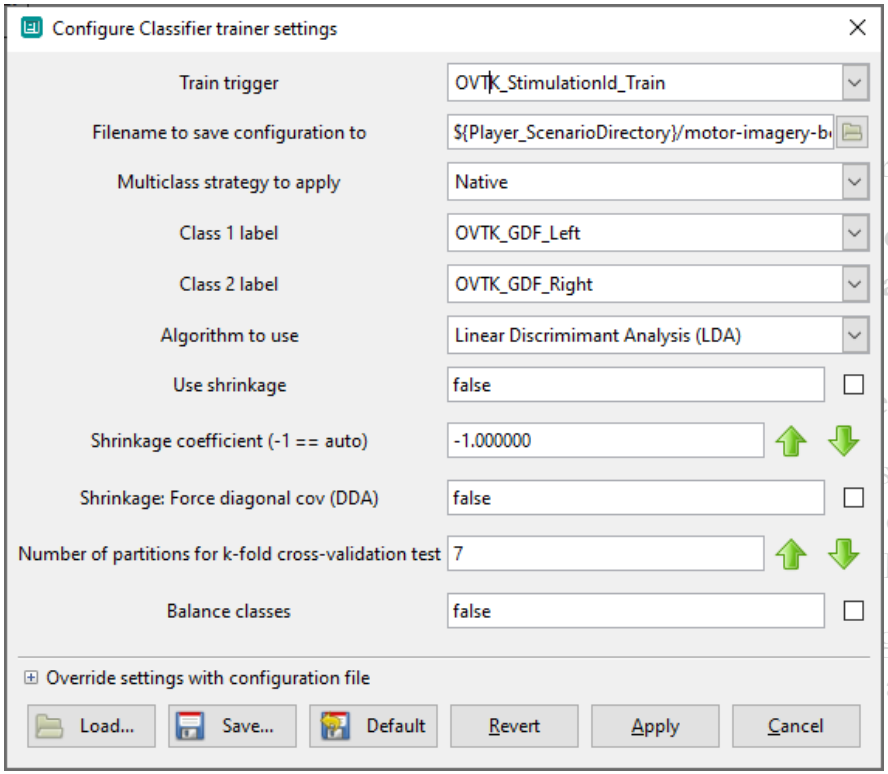


- **Load** pre-recorded signal file
- Select **sub-set of electrodes,** set **Reference Channel + apply spatial filtering**
- Filter to the **frequency band** of interest
- Stimulation-based Epoching: **split between conditions** (LEFT vs RIGHT…)
- Epoching & Signal processing: compute a "**classifier feature**" based on **spectral power**
- **Train the classifier!**




- **Load** pre-recorded signal file
- Select **sub-set of electrodes,** set **Reference Channel + apply spatial filtering**
- Filter to the **frequency band** of interest
- Stimulation-based Epoching: **split between conditions** (LEFT vs RIGHT…)
- Epoching & Signal processing: compute a "**classifier feature**" based on **spectral power**
- **Train the classifier!**




rded signal file

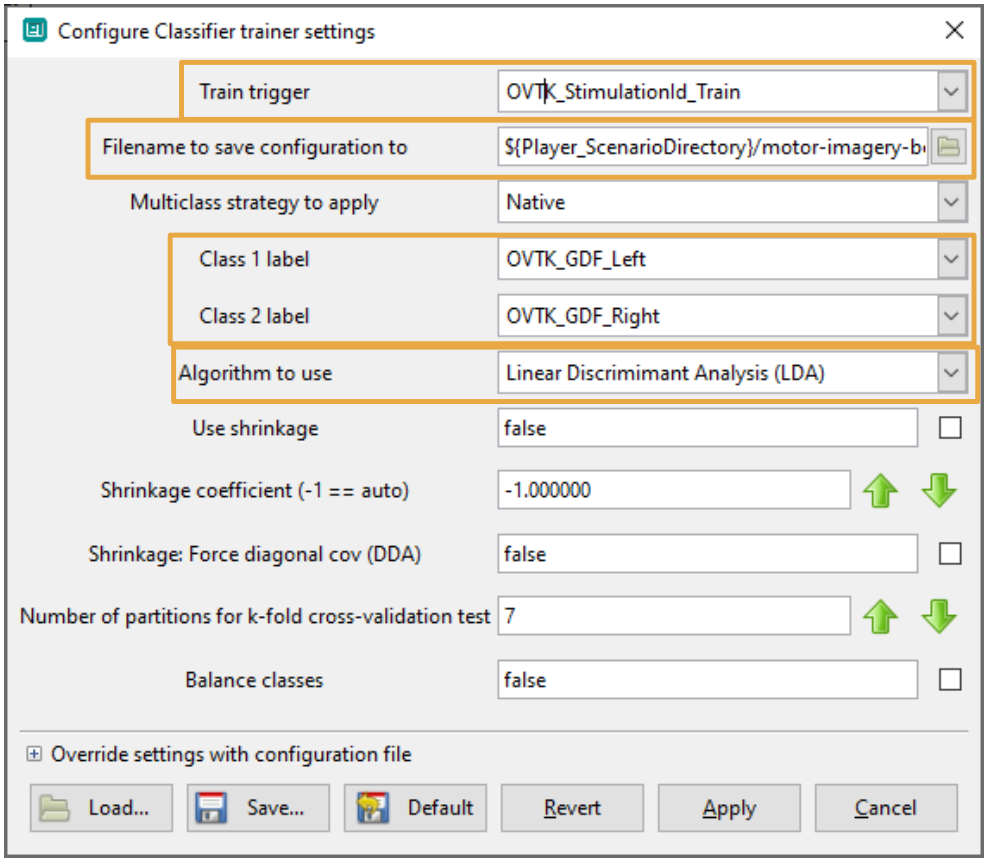
of **electrodes,** set
annel + **apply spatial**

**equency band** of interest

sed Epoching:
**conditions**
HT…)

gnal processing: compute
**ature**" based on **spectral**

- **Train the classifier!**




- **"Train trigger"** stimulation should be set in the acquisition LUA script, at the end of the experiment
- Set the **path/filename** for the training "weights" (results)
- Check that the classifier classes are correctly labeled
- Select the classifying algorithm (LDA, SVM…) and set the parameters

```
[ INF ] At time 462.008 sec <Box algorithm::(0x02e67945, 0x5ea8d309) aka Classifier trainer> Received train stimulation. Data dim is [1764x2]
[ INF ] At time 462.008 sec <Box algorithm::(0x02e67945, 0x5ea8d309) aka Classifier trainer> For information, we have 931 feature vector(s) for input 1
[ INF ] At time 462.008 sec <Box algorithm::(0x02e67945, 0x5ea8d309) aka Classifier trainer> For information, we have 833 feature vector(s) for input 2
[ INF ] At time 462.008 sec <Box algorithm::(0x02e67945, 0x5ea8d309) aka Classifier trainer> k-fold test could take quite a long time, be patient
[ INF ] At time 462.008 sec <Box algorithm::(0x02e67945, 0x5ea8d309) aka Classifier trainer> Finished with partition 1 / 7 (performance : 75.000000%)
[ INF ] At time 462.008 sec <Box algorithm::(0x02e67945, 0x5ea8d309) aka Classifier trainer> Finished with partition 2 / 7 (performance : 74.206349%)
[ INF ] At time 462.008 sec <Box algorithm::(0x02e67945, 0x5ea8d309) aka Classifier trainer> Finished with partition 3 / 7 (performance : 59.126984%)
[ INF ] At time 462.008 sec <Box algorithm::(0x02e67945, 0x5ea8d309) aka Classifier trainer> Finished with partition 4 / 7 (performance : 88.492063%)
[ INF ] At time 462.008 sec <Box algorithm::(0x02e67945, 0x5ea8d309) aka Classifier trainer> Finished with partition 5 / 7 (performance : 75.000000%)
[ INF ] At time 462.008 sec <Box algorithm::(0x02e67945, 0x5ea8d309) aka Classifier trainer> Finished with partition 6 / 7 (performance : 88.095238%)
[ INF ] At time 462.008 sec <Box algorithm::(0x02e67945, 0x5ea8d309) aka Classifier trainer> Finished with partition 7 / 7 (performance : 82.539683%)
[ INF ] At time 462.008 sec <Box algorithm::(0x02e67945, 0x5ea8d309) aka Classifier trainer> Cross-validation test accuracy is 77.494331% (sigma = 9.406674%)
[ INF ] At time 462.008 sec <Box algorithm::(0x02e67945, 0x5ea8d309) aka Classifier trainer>   Cls vs cls      1     2
[ INF ] At time 462.008 sec <Box algorithm::(0x02e67945, 0x5ea8d309) aka Classifier trainer>   Target  1:   82.7  17.3 %, 931 examples
[ INF ] At time 462.008 sec <Box algorithm::(0x02e67945, 0x5ea8d309) aka Classifier trainer>   Target  2:   28.3  71.7 %, 833 examples
[ INF ] At time 462.008 sec <Box algorithm::(0x02e67945, 0x5ea8d309) aka Classifier trainer> Training set accuracy is 80.045351% (optimistic)
[ INF ] At time 462.008 sec <Box algorithm::(0x02e67945, 0x5ea8d309) aka Classifier trainer>   Cls vs cls      1     2
[ INF ] At time 462.008 sec <Box algorithm::(0x02e67945, 0x5ea8d309) aka Classifier trainer>   Target  1:   85.2  14.8 %, 931 examples
[ INF ] At time 462.008 sec <Box algorithm::(0x02e67945, 0x5ea8d309) aka Classifier trainer>   Target  2:   25.7  74.3 %, 833 examples
```

# SC2 – FINAL WORDS / Q&A

- **Recap:**
  - Signal processing / Channel selection / Filtering / Epoching
  - Spectral Analysis, spectrum display
  - Brain topography
  - Feature selection
  - Classifier training

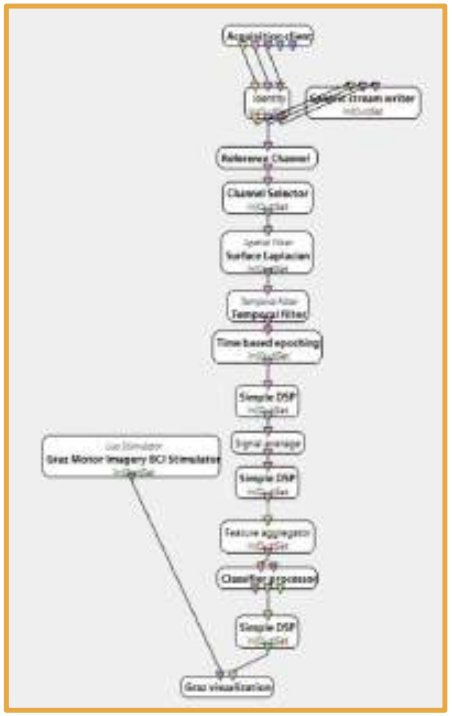
# SC3 – ONLINE CLASSIFICATION & FEEDBACK




1- Training data acquisition

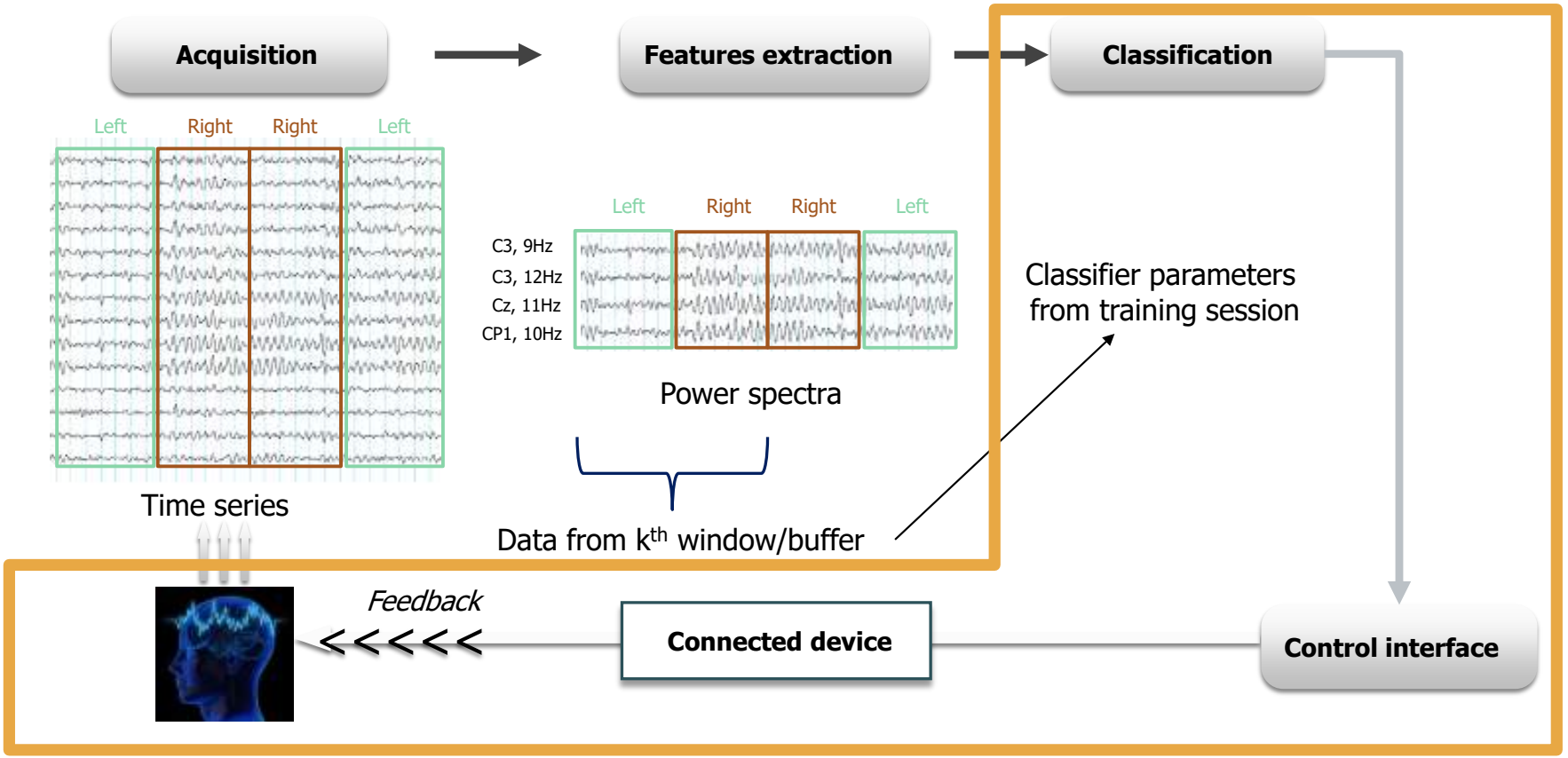


2- Features extraction & Classification




**3- Online feedback**

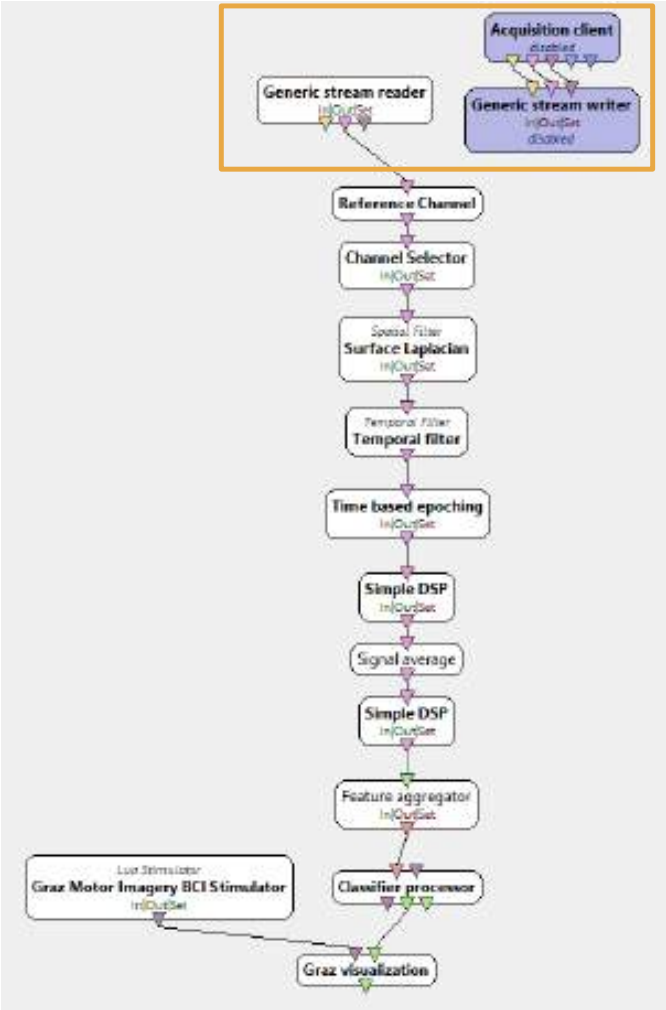
# SC3 – ONLINE CLASSIFICATION



# SC3 – ONLINE CLASSIFICATION

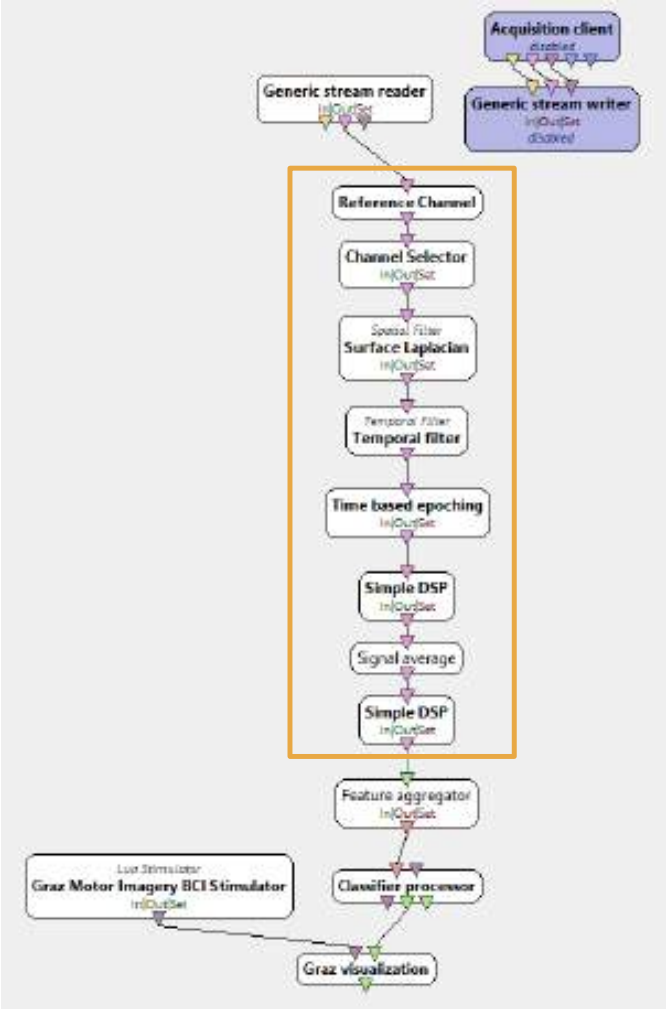
- **Goals:**
  - Acquire EEG data in real-time, while providing "tasks" to the subject
  - Classify his/her mental states (also in real-time) using the classifier trained in SC2
  - (almost) everything we need is already there!
- **Let's load the scenario and analyze it…**

  <OpenViBE_install_path>\share\openvibe\scenarios\bci-examples\motor-imagery**motor-imagery-bci-3-online.xml**




- In a real BCI experiment, we should of course use the Acquisition Client (and record the incoming signals for future replays and studies)

- Here, for the sake of an already long workshop, we'll use the same signal file as before…



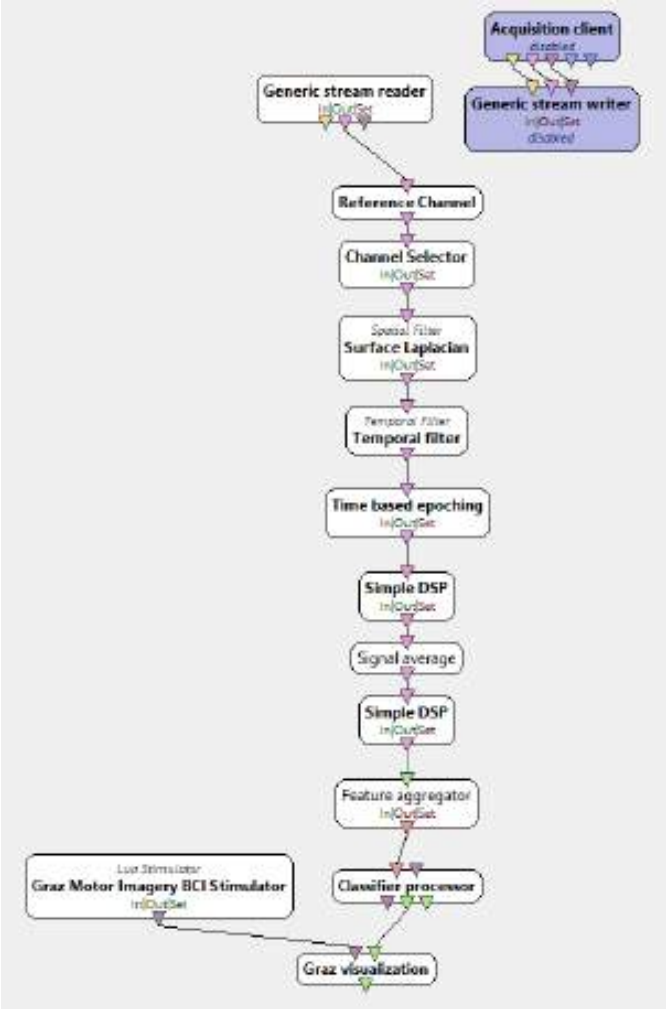

- We want the classifier to have the same type of data used for training
- (It's actually the other way around: train your classifier with the same data type you use in the online step)

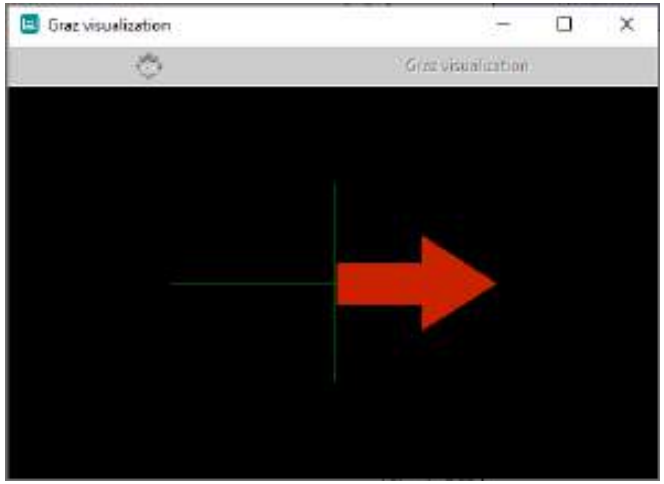
- The whole selection, filtering, epoching & Signal Processing chain is the same as in SC2 !

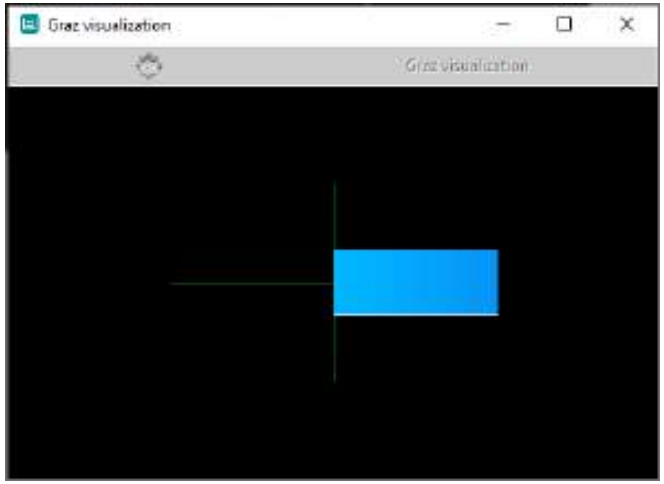

- Use the "**Classifier Processor**" box, loading the correct configuration file
(= the training step output)

- Use the same LUA script & Graz Visualization boxes as for the SC1 (acquisition) to generate stimulations & "tasks"
- The "**streamed matrix**" stream from **Classifier processor** to **Graz Visualization** corresponds to the "classification **accuracy**"


- **Notes** regarding the difference between this simulation and an actual BCI/MI protocol
- In a real BCI experiment, the user would receive as visual inputs:
  - first, an arrow towards the side on which to perform Motor Imagery task,
  - then a continuous feedback of the classifier's performance (in separating MI from rest)
- As in SC1, the Graz visualization box is used both to update the display, and to propagate the stimulations to the ACQ server for synchronization with the signal stream.
- Here, we use pre-recorded signals, so this stimulation sync. is not done…


Acquisition client
Generic stream reader
Generic stream writer
Reference Channel
Channel Selector
Surface Laplacian
Temporal filter
Time based epoching
Simple DSP
Signal average
Simple DSP
Feature aggregator
Graz Motor Imagery BCI Stimulator
Classifier processor
Graz visualization


Graz visualization
Graz visualization


Graz visualization
Graz visualization

# SC3 – FINAL WORDS / Q&A

- **Recap:**
  - Online classification pipeline
  - Example of real-time visual feedback

# CHAPTER 2 – Q&A

BCI Motor Imagery with OpenViBE in X-Men: First Class

# CONCLUDING REMARKS & PERSPECTIVES

-

# OPENVIBE


ARAMIS LAB BRAIN DATA SCIENCE
Institut du Cerveau
Inria
Inserm
cnrs
SORBONNE UNIVERSITÉ

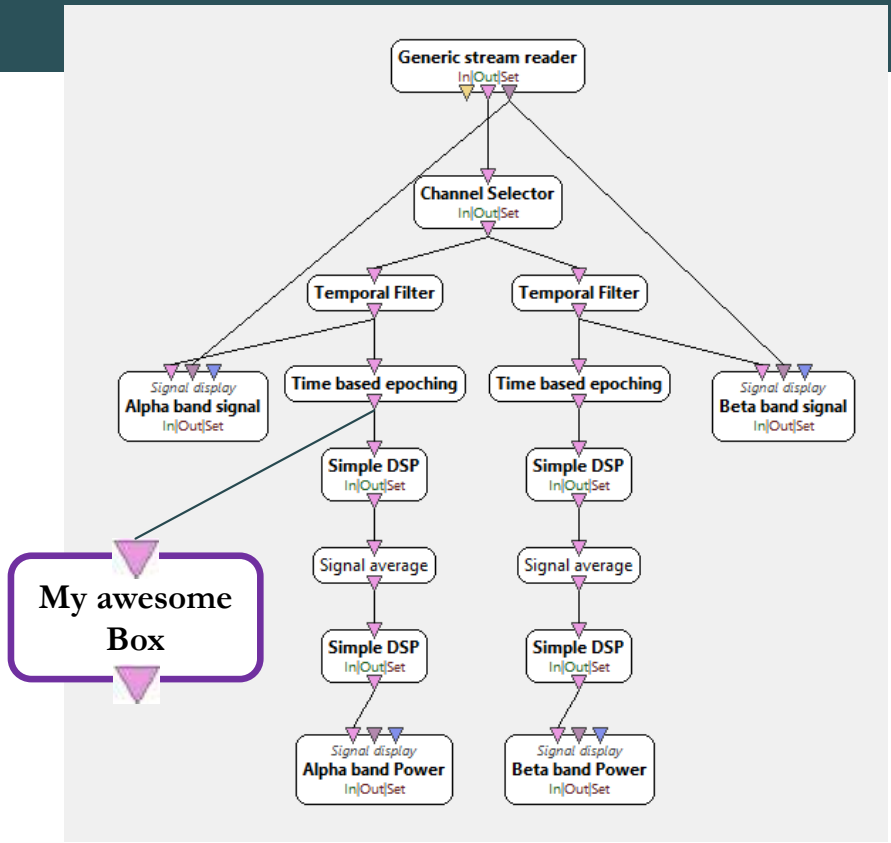
# PROTOTYPING, DESIGNING, TUTORIALS

- As seen in Chapter 2, with OpenViBE you can easily prototype and design BCI protocols and experiments

- Lots of **scenario examples and templates** are already available in the install!
  <openvibe-3.1.0-64bit>\share\openvibe\scenarios\bci-examples
- Wanting to use a particular box? **Tutorial scenarios** are there for you:
  <openvibe-3.1.0-64bit>\share\openvibe\scenarios\box-tutorials
- Check the general documentation for a great amount of info:
  http://openvibe.inria.fr/documentation-index/

# BOX DEVELOPMENT

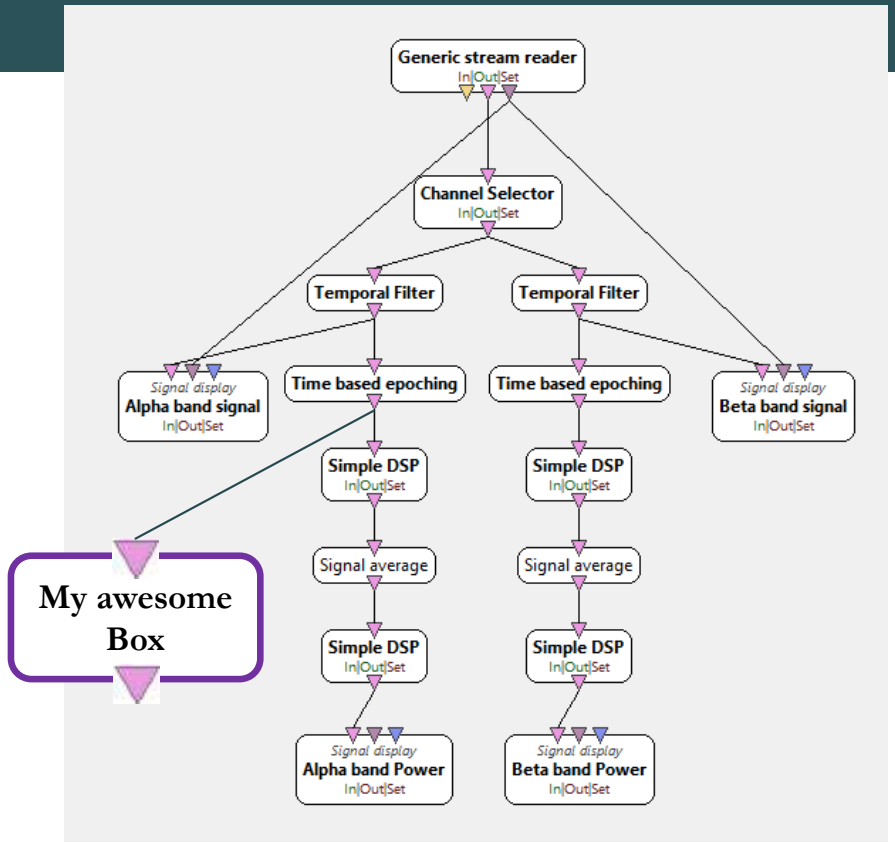
- So, you want to develop a **new processing box**?



# BOX DEVELOPMENT

- So, you want to develop a **new processing box**?

- First step:
  check if an existing box has what you need…
  … or if you can do what you want using a combination of existing boxes.



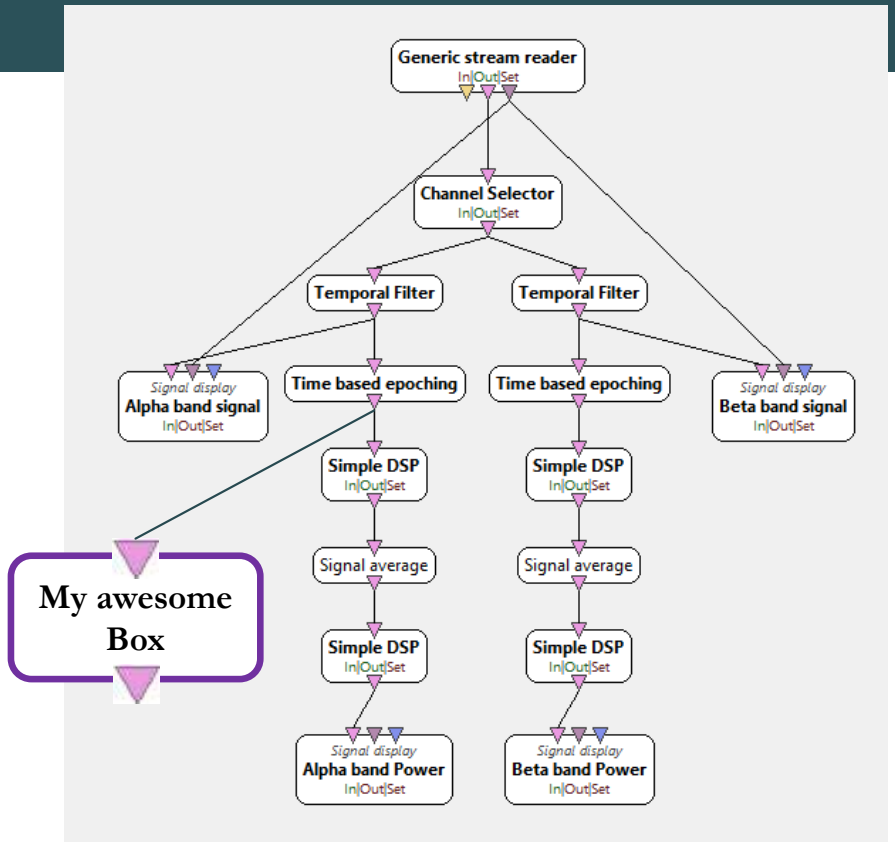
# BOX DEVELOPMENT

- So, you want to develop a **new processing box**?

- **First step**:
  check if an existing box has what you need…
  … or if you can do what you want using a combination of existing boxes.

- If not – then:
  Do you want a quick&flexible prototype?
  → box calling **Python/Matlab scripts**

  … or a fine-tuned optimized algorithm?
  → **C++ Box & Algorithm** classes



# BOX DEVELOPMENT – PYTHON/MATLAB

- **Using Python/Matlab scripts in OpenViBE scenarios**

Use cases:

- Need for a quick proof-of-concept (e.g. signal processing)
- Don't want/need to code in C++
- Python/Matlab implementation is already perfect
- Need specific libraries (numpy, scikit-learn…)

http://openvibe.inria.fr/tutorial-using-matlab-with-openvibe/

http://openvibe.inria.fr/tutorial-using-python-with-openvibe/

- Great Python tutorial: (courtesy of MENSIA)
  - http://openvibe.inria.fr/openvibe/wp-content/uploads/2016/06/Quick-prototyping-in-OpenViBE-with-Python.pdf

# BOX DEVELOPMENT – C++

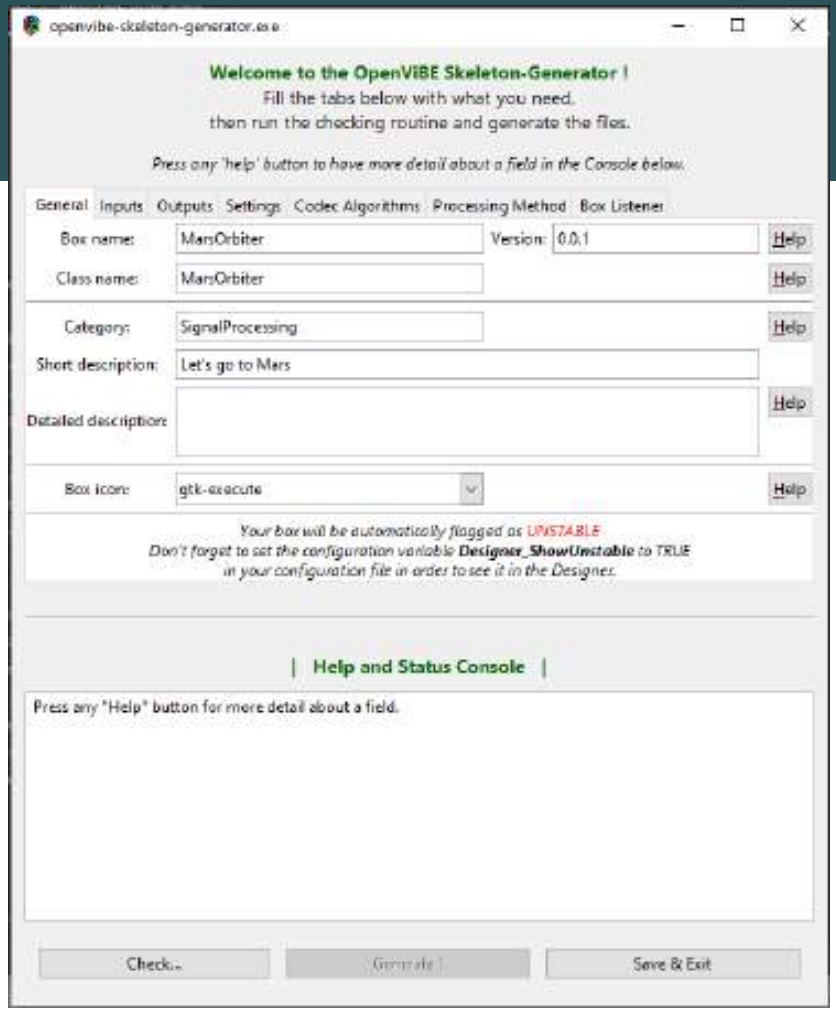
- **Developing C++ OpenViBE boxes**
- Use cases:
  - Need speed!
  - Complete integration with OpenViBE, contribution to the open-source project

- http://openvibe.inria.fr/build-instructions/
- **2016 Tutorial**: http://openvibe.inria.fr/openvibe/wp-content/uploads/2016/06/jl_hacking_boxes_2016.pdf

# BOX DEVELOPMENT – C++

- **Skeleton generator**
  Simplest, fastest, go-to solution for beginners…
  openvibe-skeleton-generator.cmd
- GUI helping with creating the bare minimum a box needs, with given inputs/outputs, parameters, etc.

  All the "OpenViBE glue" is here!
  You "only" need to add your specific code.
- http://openvibe.inria.fr/tutorial-1-implementing-a-signal-processing-box/
- Tip: take inspiration from existing boxes…!

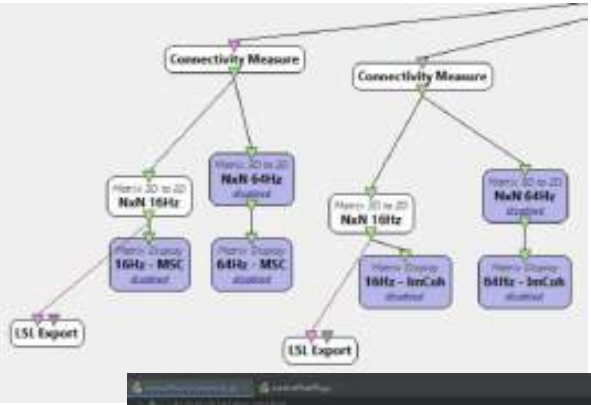


# EXTERNAL INTERFACES

- Examples: interface w/ virtual reality products, video games, external data visualization toolboxes…
- Various ways exist to stream data/events between OpenViBE and external apps.
  - VRPN
  - TCP/IP
  - LSL (Lab Streaming Layer)
  - Python/Matlab boxes

- Demo using LSL to visualize Connectivity/Adjacency Matrices using an external Python script
- Code and example scenarios:

  https://github.com/AsteroidShrub/openVibe-Lsl-Demo

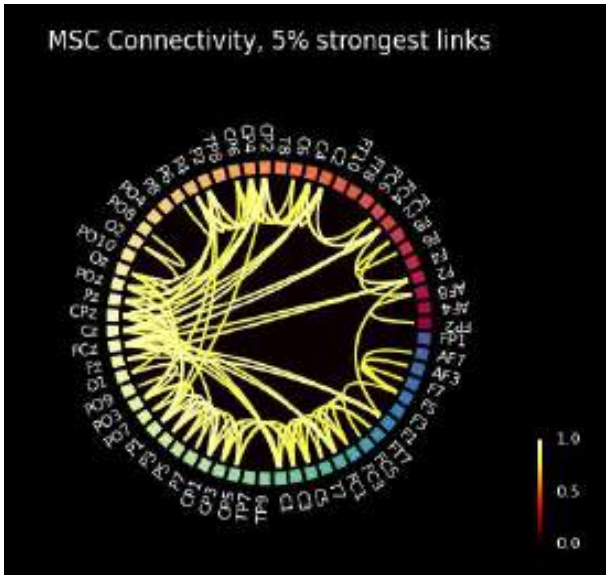


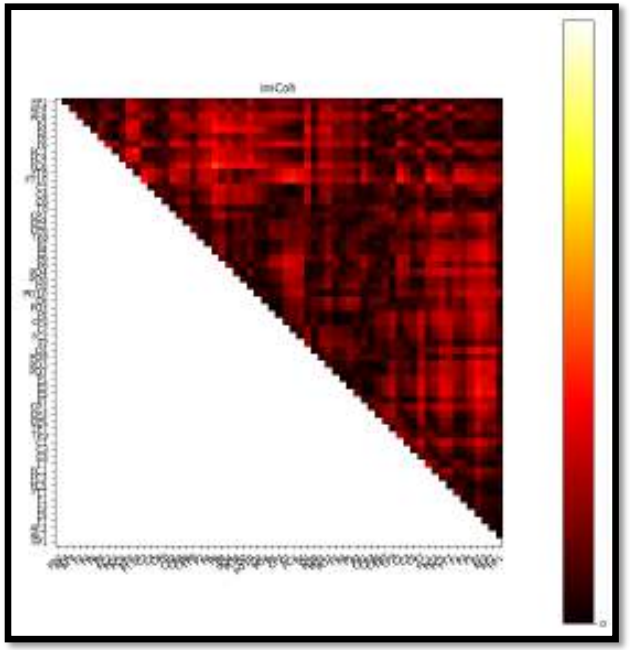

- OpenViBE "LSL Export" Box

Connectivity Measurement Box

- + Python scripting (using pylsl)

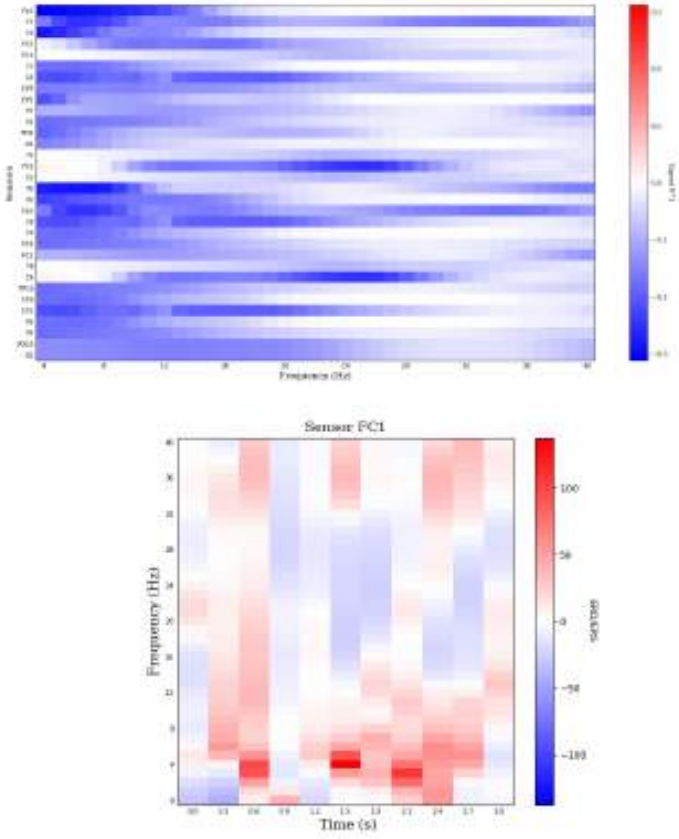
  ➔ external matrices analysis/plotting



# ONGOING PROJECT: BCI PIPELINE AUTOMATION

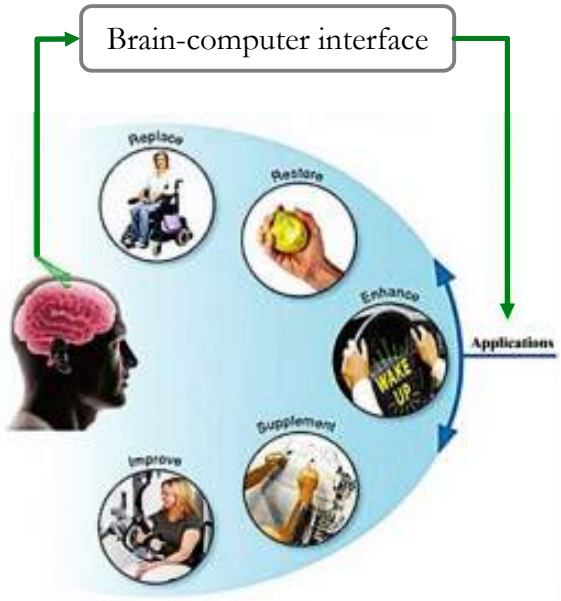
- **Goals:**
  - **GUI for automatic generation of scenarios**, in a unified & robust pipeline framework (acquisition / feature-extraction / training / online)
    Scenarios & parameters automatically generated depending on user's preferences, from template scenarios
  - **GUI with data viz for feature selection**, with automatic scenario update after selection
    ex: $R^2$ map from Spectral power in a set of frequency bands
    ex: Node Strength based on connectivity



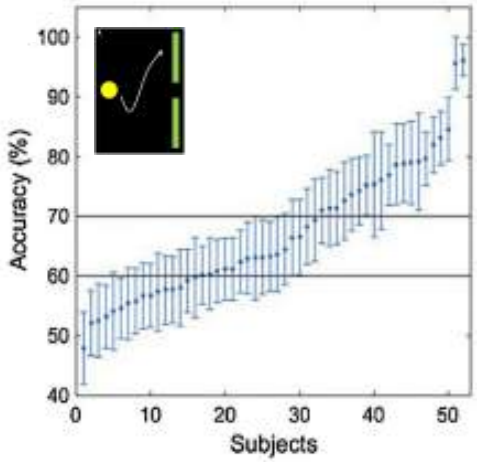
# CONCLUDING REMARKS & PERSPECTIVES
-
# BCI RESEARCH

# BCI INEFFICIENCY CHALLENGE

## Great potential




## Poor usability




(Ahn & Jun, 2015)

**Problem** : Current BCIs fail to detect the mental intentions in ~30% of users

# BCI INEFFICIENCY CHALLENGE – STATE-OF-THE-ART

- Machine-centered approaches
  - Signal conditioning (Ang et al, 2012)
  - Classification algorithms (Lotte et al, 2018)

⇒ Rely on EEG signals

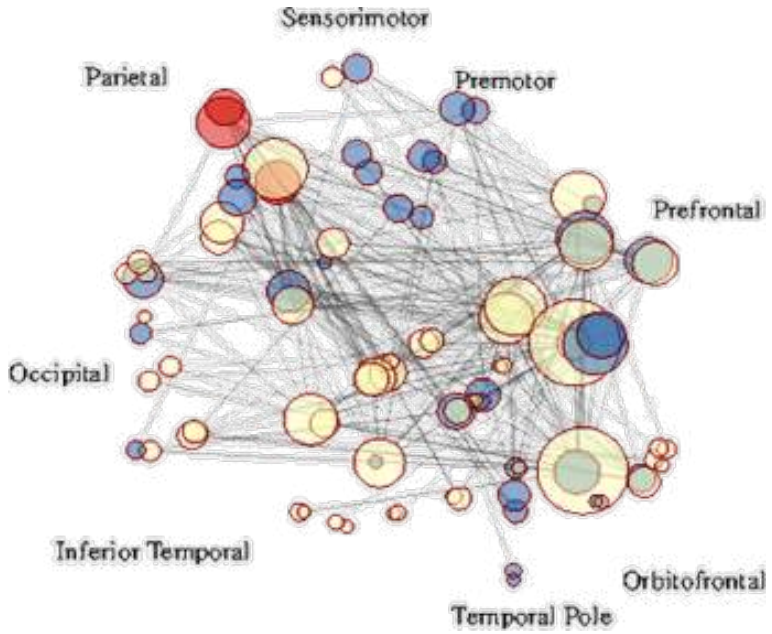
- User-centered approaches
  - Search for neurophysiological patterns (Blankertz et al, 2010)
  - Human factors (Jeunet et al, 2015)

⇒ Lack of reliable markers

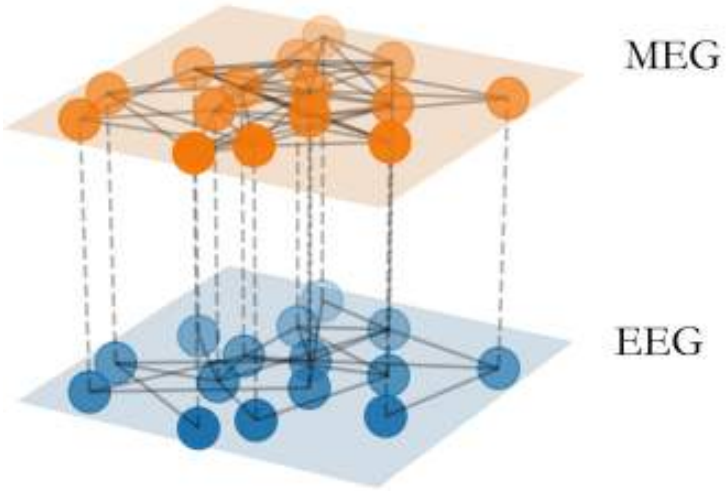
⇒Neural mechanisms underlying BCI learning **poorly understood**

⇒Do not consider the **interconnected** nature of the brain functioning

# BCI INEFFICIENCY CHALLENGE – NETWORK APPROACH



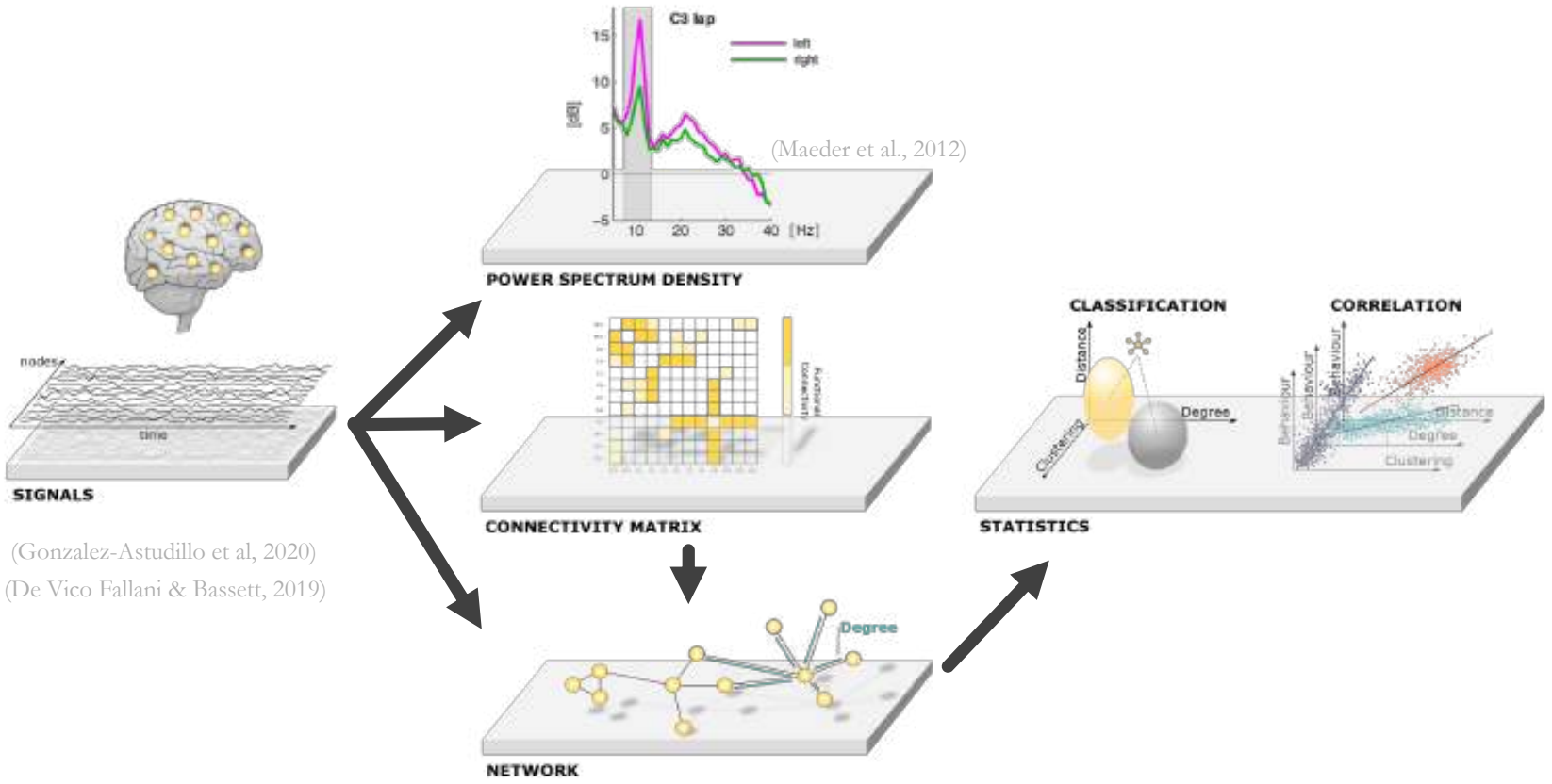

(Varela et al, 1999)






Use of multimodal brain networks to identify alternative features & BCI learning patterns

# NETWORK METRICS FOR MENTAL STATES CHARACTERIZATION

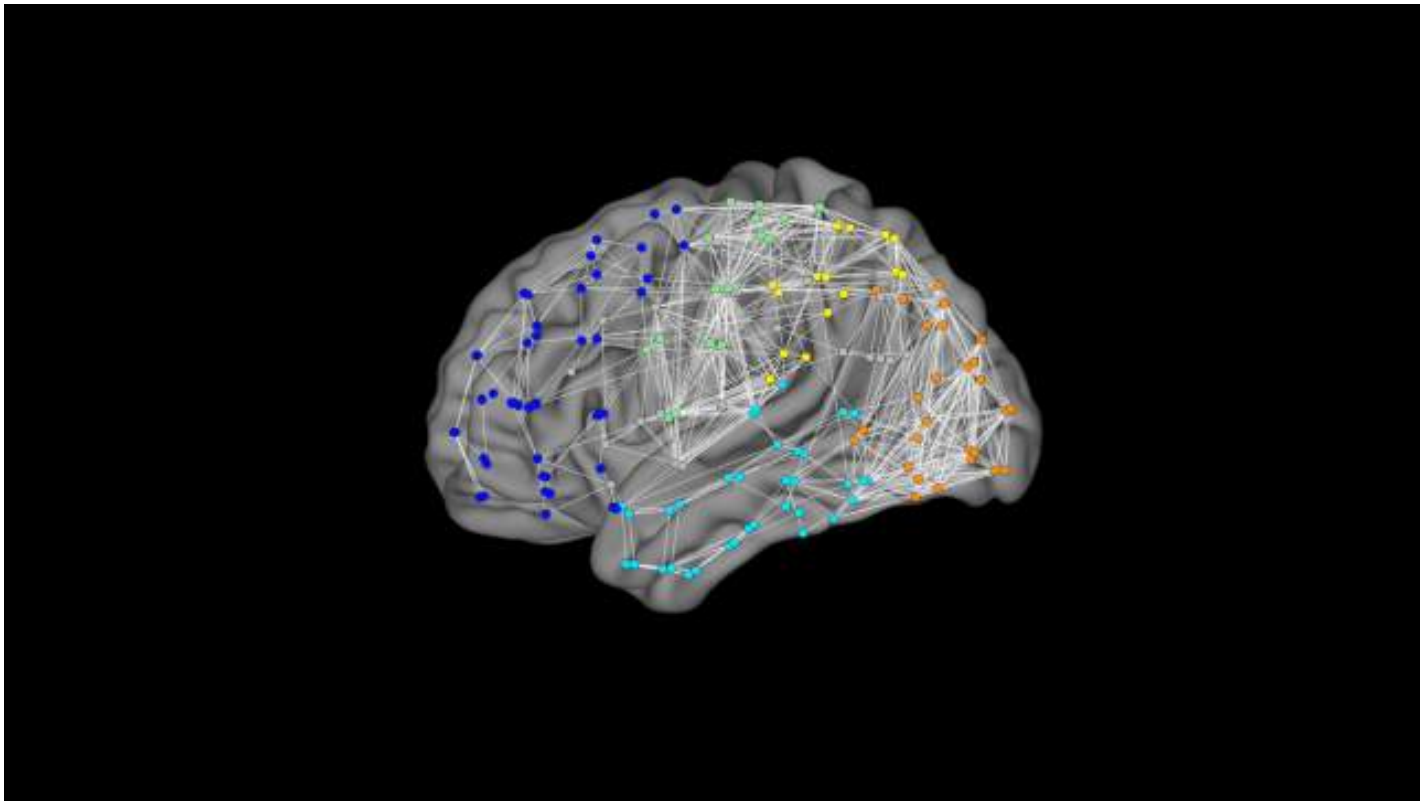



(Maeder et al., 2012)

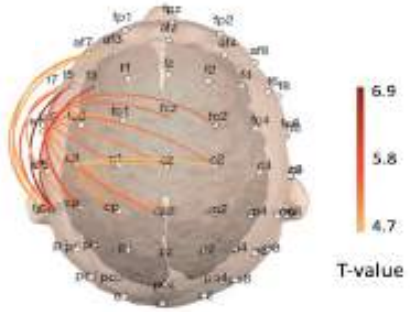
(Gonzalez-Astudillo et al, 2020)

(De Vico Fallani & Bassett, 2019)

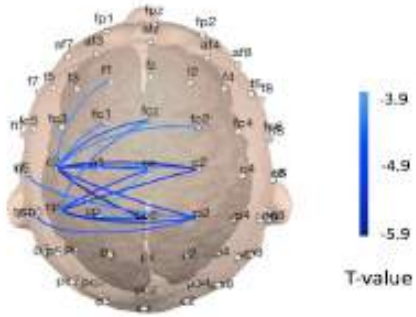
# NETWORK METRICS FOR MENTAL STATES CHARACTERIZATION



# BRAIN CONNECTIVITY CHANGES IN MI-BCI

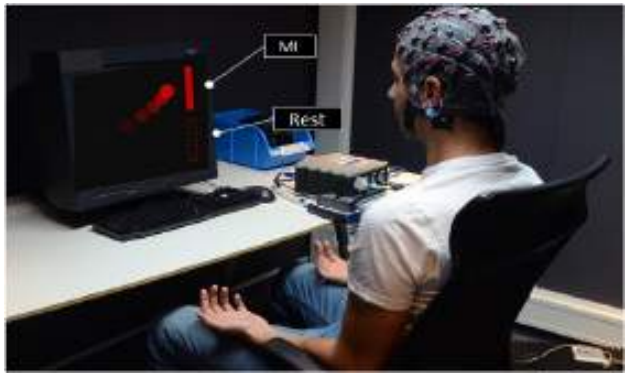



Amplitude synchronization



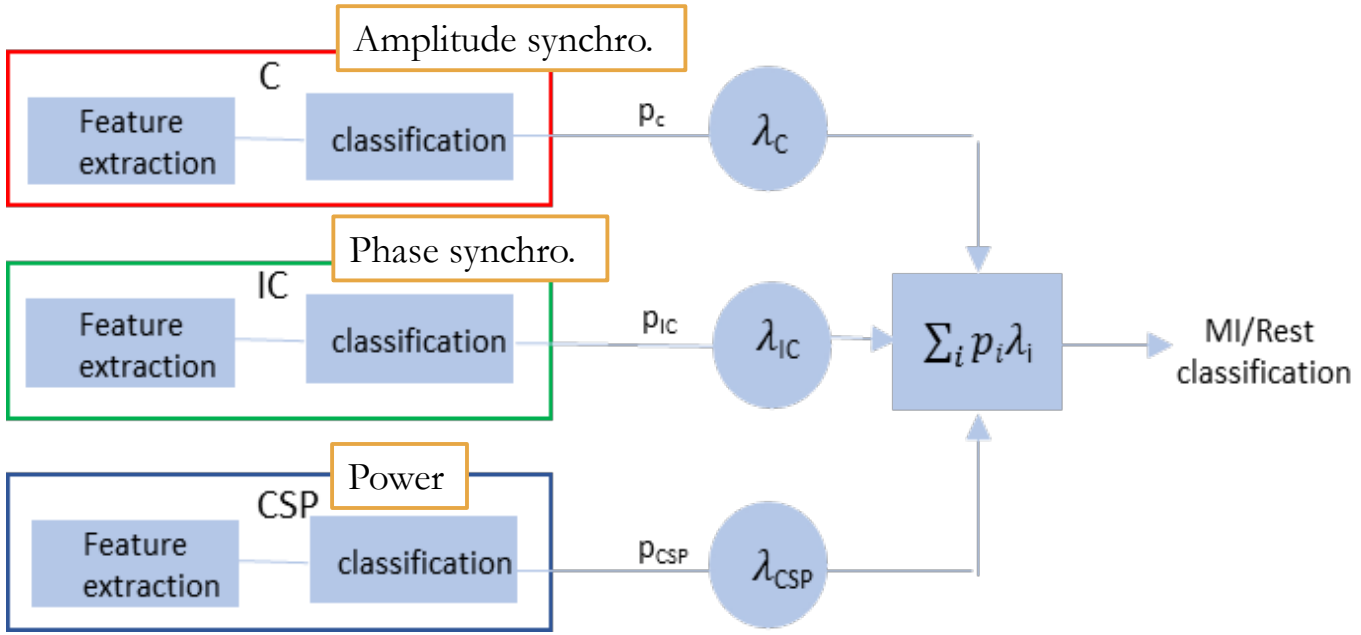

Phase synchronization

Motor imagery
VS
Resting state




(Cattai et al, IEEE TNSRE, 2021)

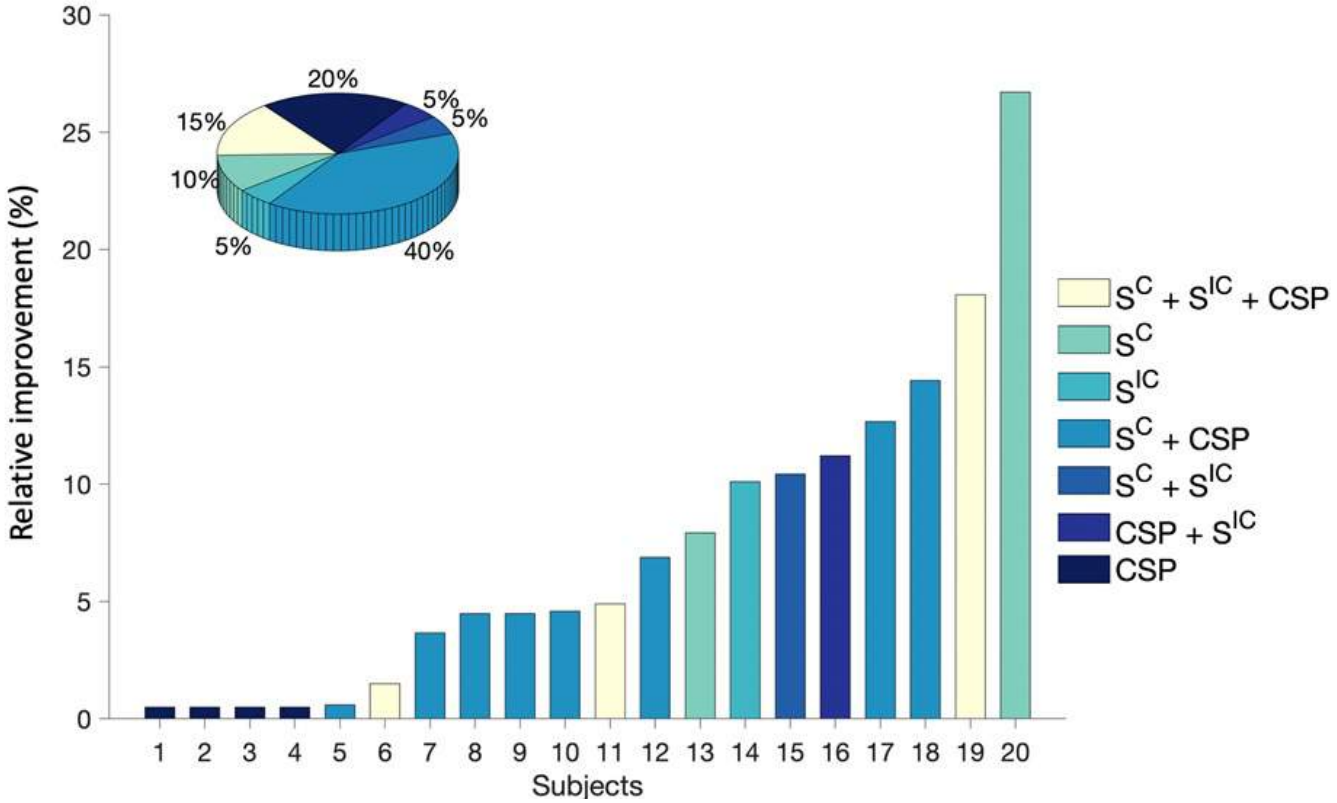
# FUSING INFORMATION TO IMPROVE THE CLASSIFICATION






(Cattai et al, IEEE TNSRE, 2021)

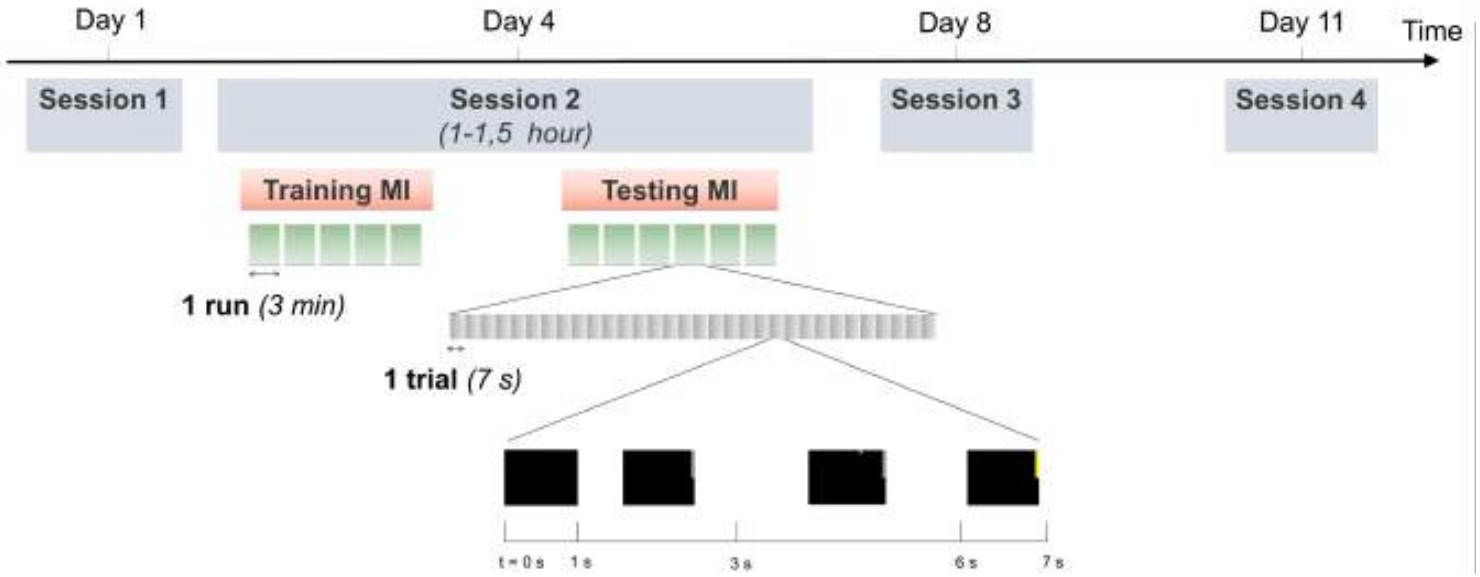
# FUSING INFORMATION TO IMPROVE THE CLASSIFICATION



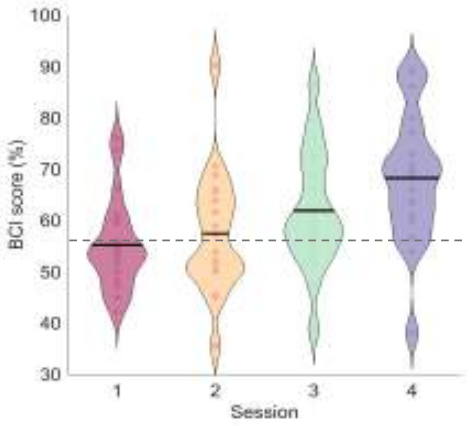

(Cattai et al, IEEE TNSRE, 2021)

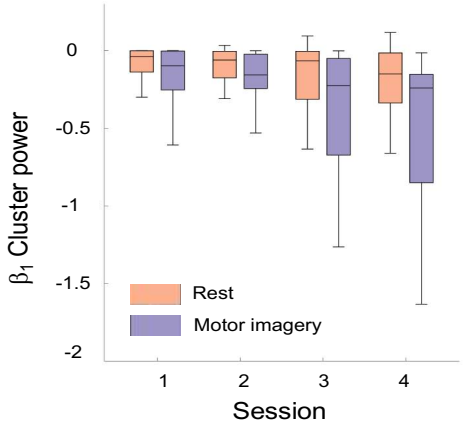
# HOW DO WE LEARN TO CONTROL A BCI ?

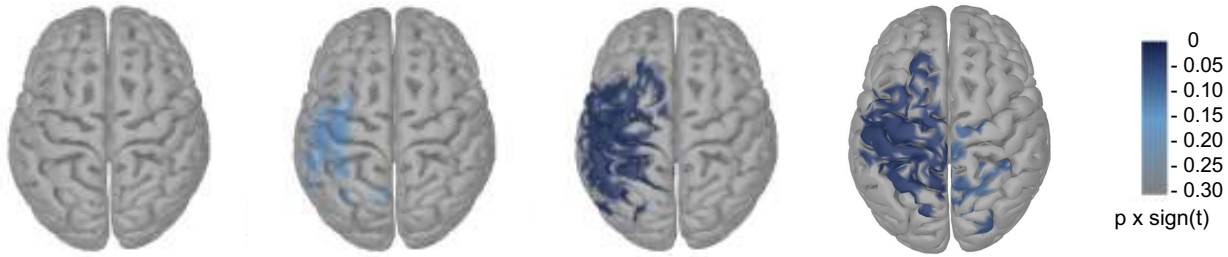
**NETBCI project**



# REINFORCEMENT OF MOTOR-RELATED ACTIVITY

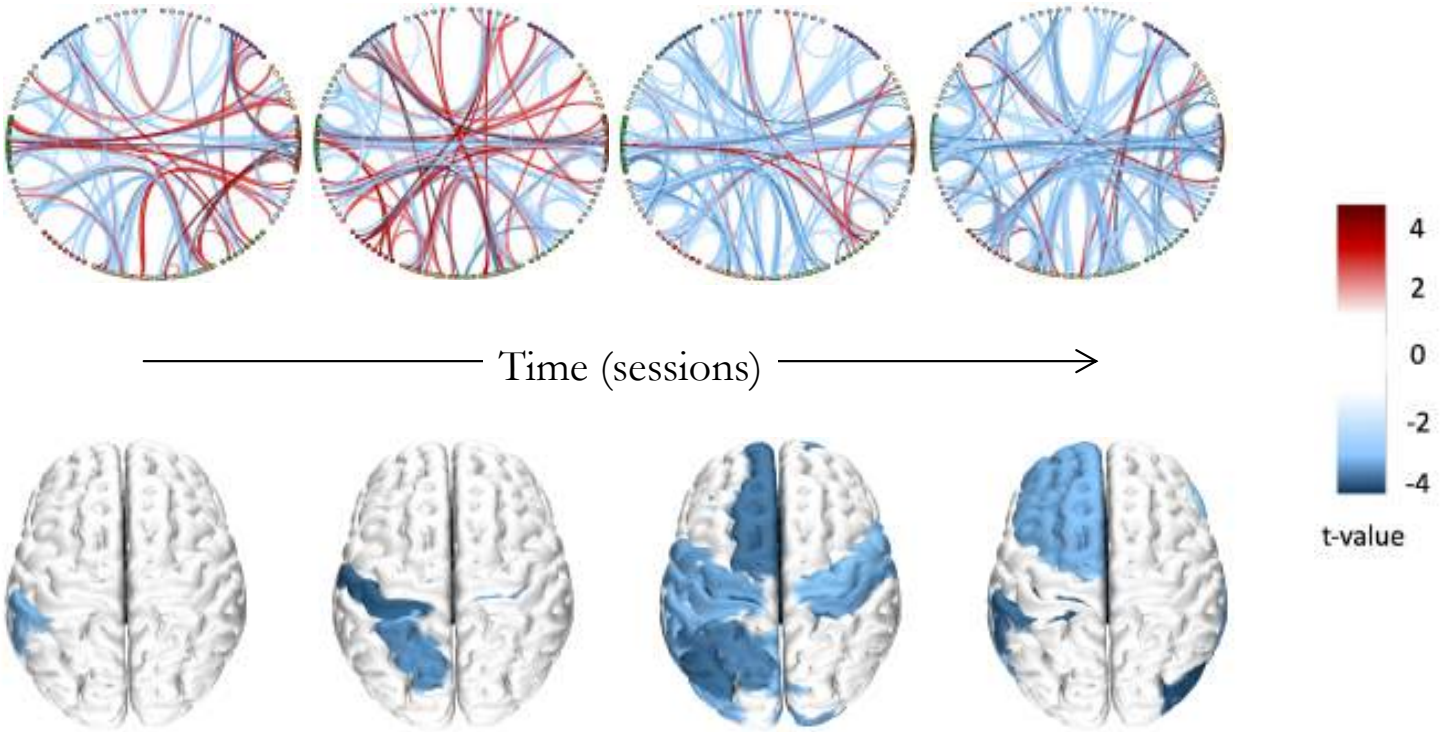


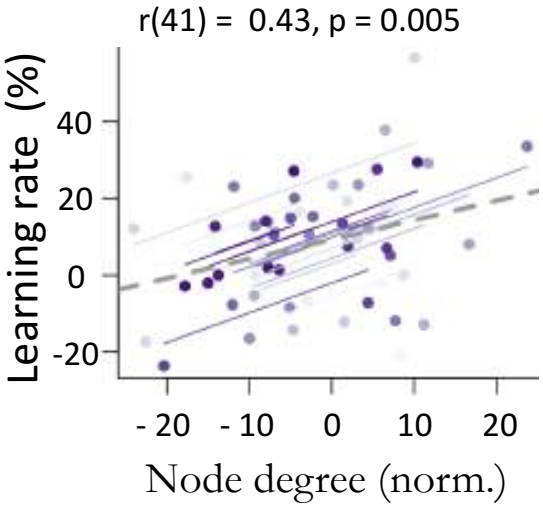

(Corsi et al, 2020)

# FUNCTIONAL DISCONNECTION OF ASSOCIATIVE AREAS




(Corsi et al, 2020)

# NODE STRENGTH PREDICTS BCI LEARNING RATE

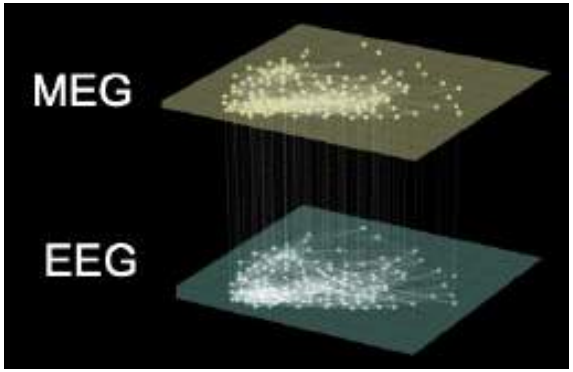



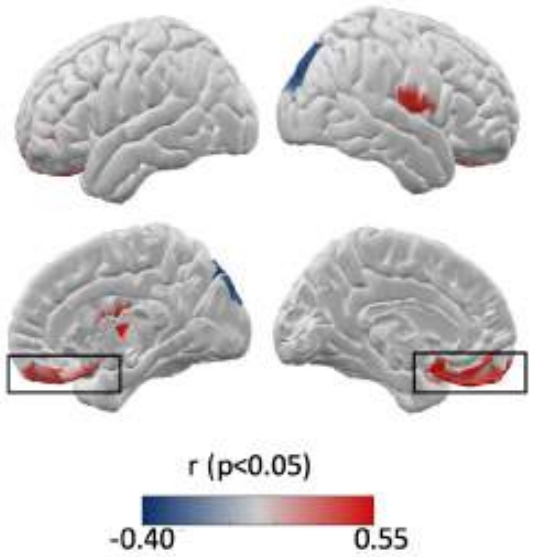

The *reserve* effect

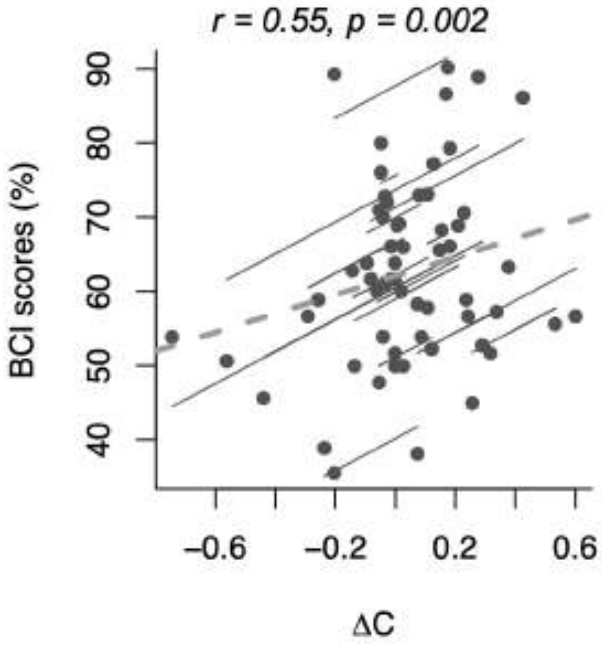
Higher connectivity → higher *potential* to disconnect (learning)

(Corsi et al, 2020)

# MULTIPLEX CORENESS ASSOCIATED WITH BCI PERFORMANCE



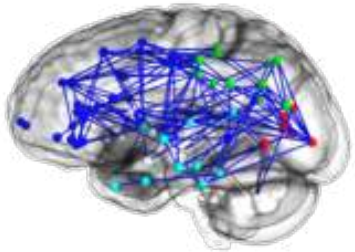



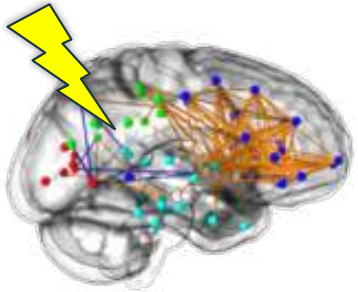



(Corsi et al, 2021)

# STROKE – CORTICAL REORGANIZATION



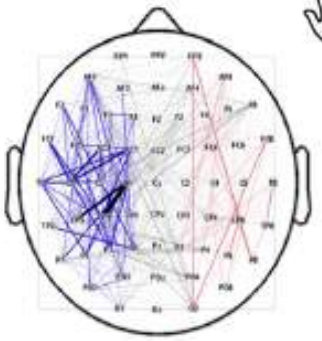
Disability

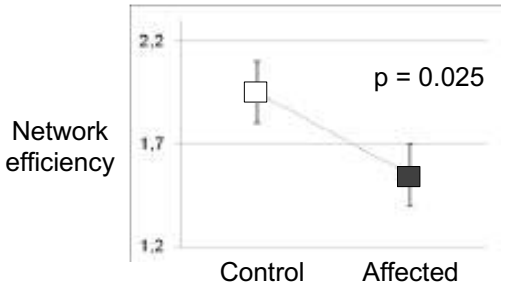


Motor Imagery

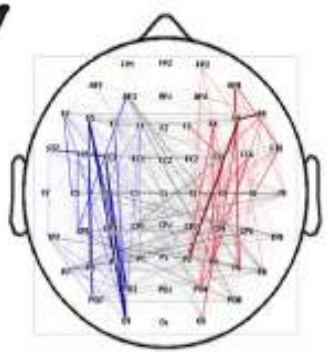
# STROKE – INTER-HEMISPHERIC CONNECTIVITY & EFFICIENCY

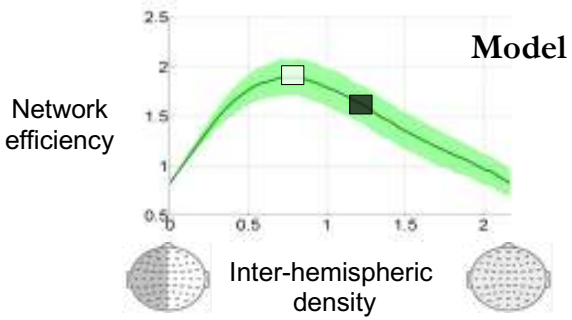
**Control**





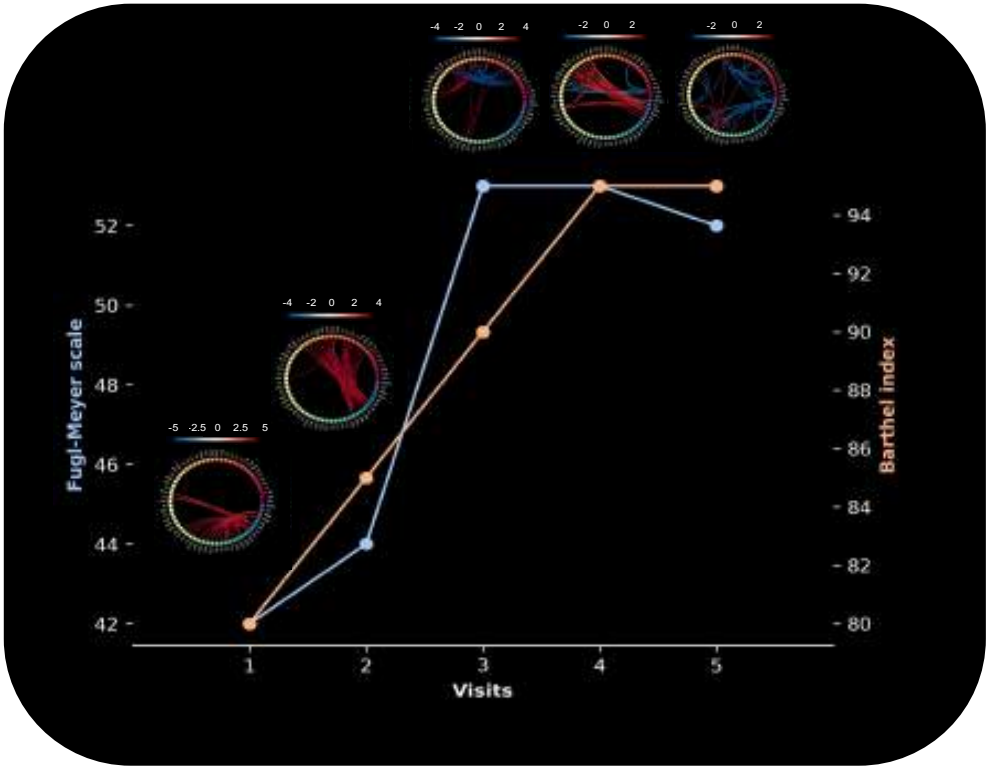

**Affected**





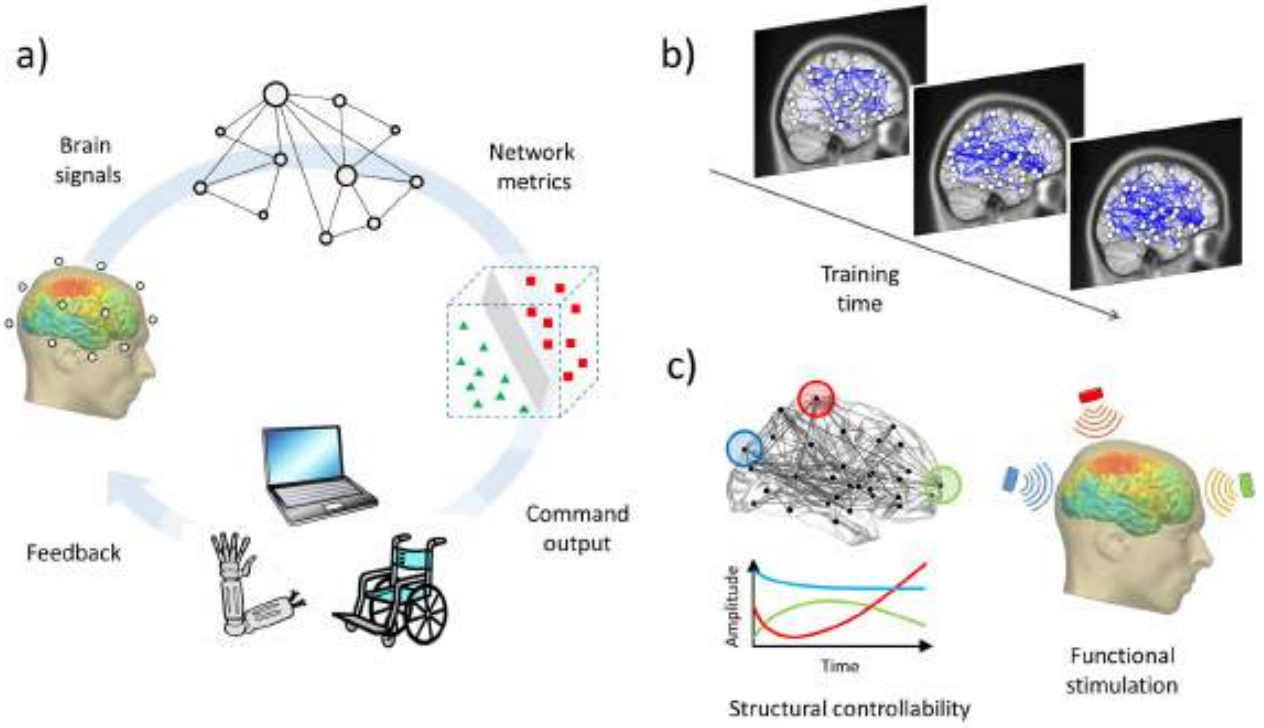

(De Vico Fallani et al, 2013)

# STROKE – SEARCH FOR ALTERNATIVE FEATURES




Neurophysiological patterns of stroke recovery over 1 year (ongoing project w/ AP-HP)

# NEW PERSPECTIVES FOR OPTIMIZING BCIS




(De Vico Fallani & Bassett, 2019)

# TAKE HOME MESSAGES

- BCI
  - Promising tool for clinical applications
  - Multidisciplinary domain
  - Growing interest in the last few years with the AI

- BCI learning & inter-subject variability
  - Improving the classifier / signal processing
  - Improving instructions
  - Finding (new) subject-related predictors

- Groups & events
  - International: BCI society, international society
  - Cybathlons: competitions to promote BCI and to test the finest algorithms with **end users !**
  - In France: CORTICO, French association to promote BCI

# TO GO FURTHER…

- Python tools – with many tutorials
  - Performing online experiments : OpenViBE, an Inria software
  - Open datasets to test algorithms & check their replicability: MOABB
  - M/EEG data analysis : MNE-Python
  - Classification tools : Scikit-learn

- Available demos (available soon)
  - Visualize E/MEG data
  - Data extraction (ERD/S)
  - Classification