OpenViBE tutorial

Basics, concepts, step-by-step use of the software

David Ojeda, PhD
Mensia Technologies

Summary

- Training objectives:
  - Understand the main concepts
  - First hands on the Graphical User Interface
  - First signal acquisition
  - Understand tutorial scenarios
  - Create your own scenarios

- Schedule:
  - General architecture, installation, file tree
  - Reading a pre-recorded file, EEG signal filtering
  - Handling the Visualization widgets
  - Datastream structures and manipulation
  - Computing band powers and typical frequency bands (alpha, beta, delta, theta)
  - Spectral Analysis
  - Acquisition server and real time applications

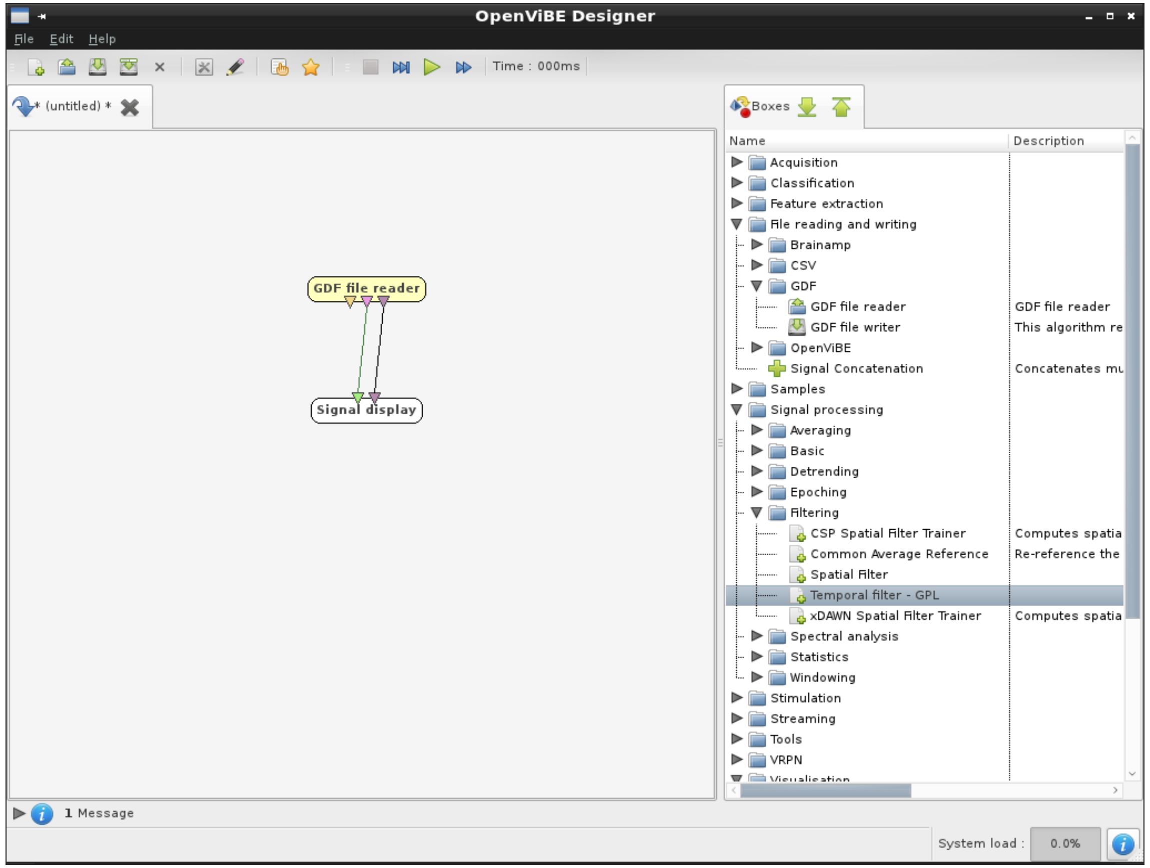
General Architecture, Installation tree

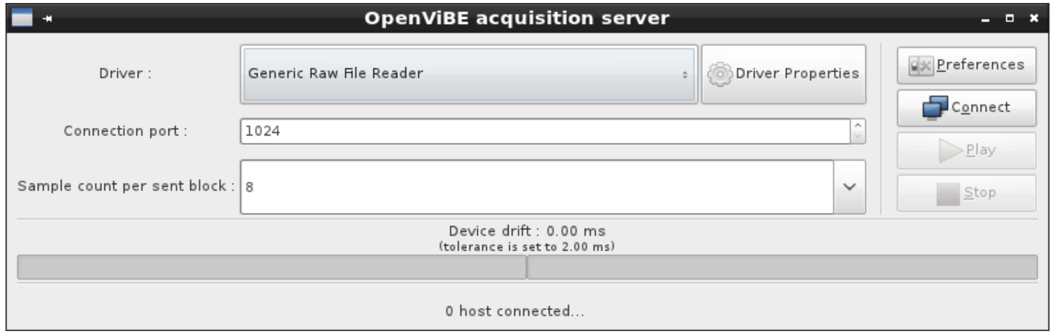
OpenViBE

- OpenViBE is made of two principal applications

- The OpenViBE designer
  - For creating, modifying and using BCI scenarios
  - Graphical programming language
  - Aggregation of interconnected boxes

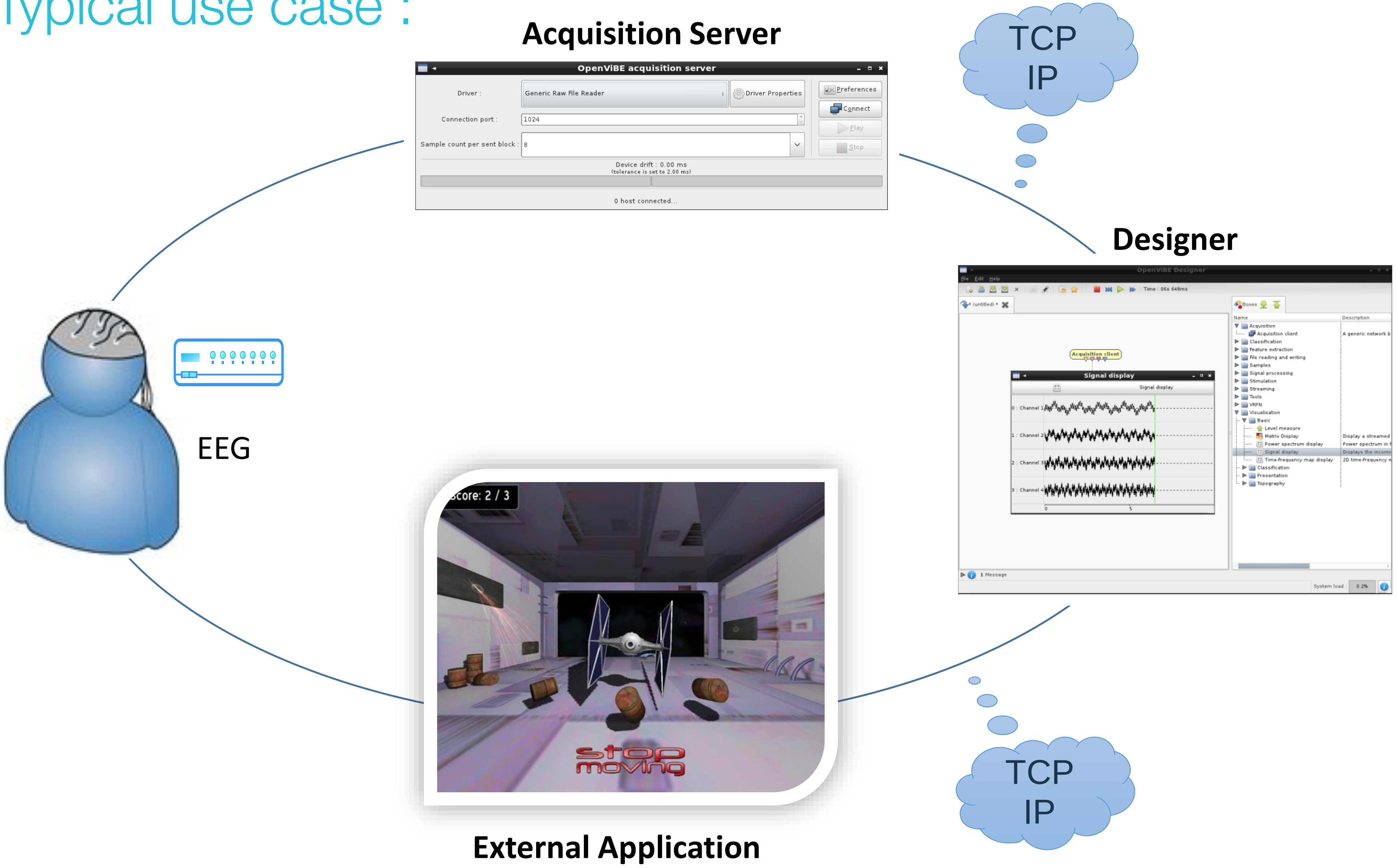
- The OpenViBE acquisition server
  - Acquires brain signals from the device
  - Translates signals from many possible devices in a common format
  - Sends the data to the applications connected (e.g. over a local network) such as the designer





OpenViBE

- Typical use case :



OpenViBE tree structure

- openvibe acquisition server
- openvibe designer

or

- openvibe
  - openvibe-acquisition-server.cmd
  - openvibe-designer.cmd

Main applications :
Acquisition Server or Designer

- openvibe
  - share
    - openvibe
      - scenarios
        - bci-examples
        - box-tutorials
        - convert
        - signals

Sample scenarios :
Basic tutorials (*box-tutorials*) and advanced scenarios (*bci-examples*), plus signal files and format conversion scenarios

- openvibe
  - openvibe-ssvep-mind-shooter-shooter.cmd
  - openvibe-ssvep-mind-shooter-trainer.cmd
  - openvibe-vr-demo-handball.cmd
  - openvibe-vr-demo-tie-fighter.cmd

2D/3D Demo applications for advanced scenarios :
Handball, tie-fighter, ssvep shooter

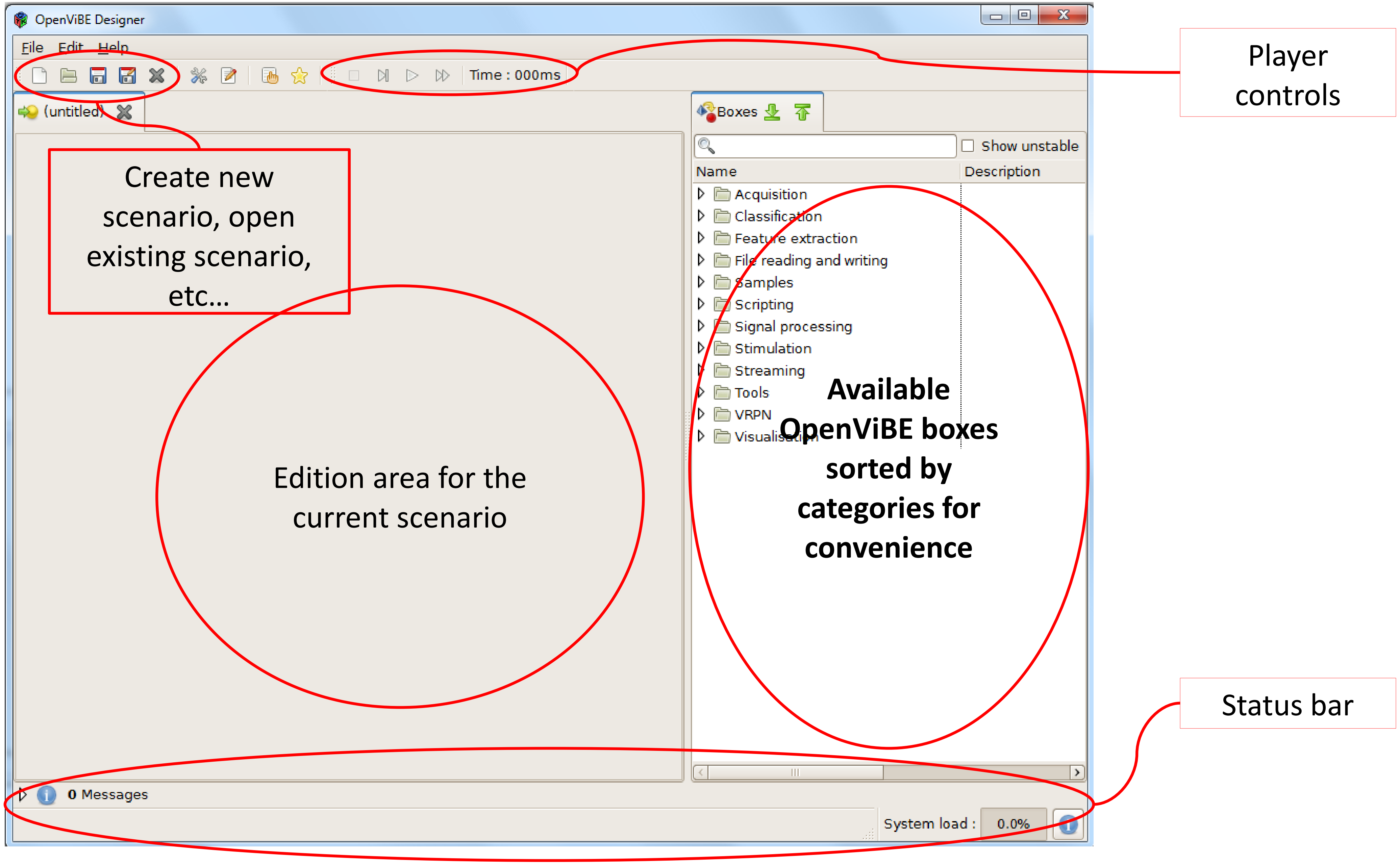
- openvibe
  - openvibe-id-generator.cmd
  - openvibe-plugin-inspector.cmd
  - openvibe-skeleton-generator.cmd

Developers and contributors tools

Start the Designer :

- **From Start menu (Start → OpenViBE → OpenViBE designer)**
- **From the file explorer, directly execute *openvibe-designer.cmd* in the OpenViBE folder**

Discovering the GUI

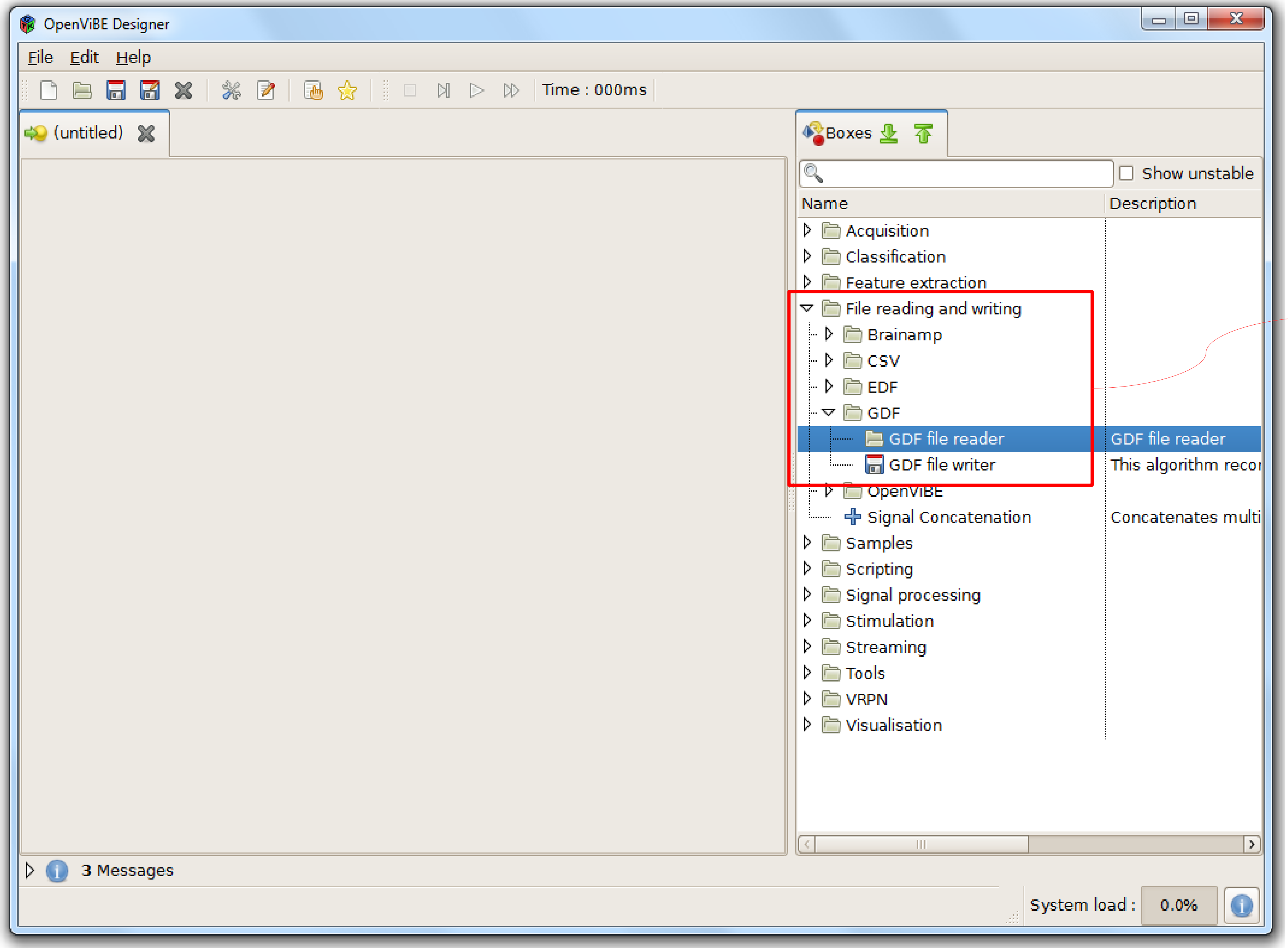


Reading a pre-recorded file

1. Select reading / display signal boxes
2. Connect input and outputs
3. Configure boxes
4. Play

Reading a pre-recorded file

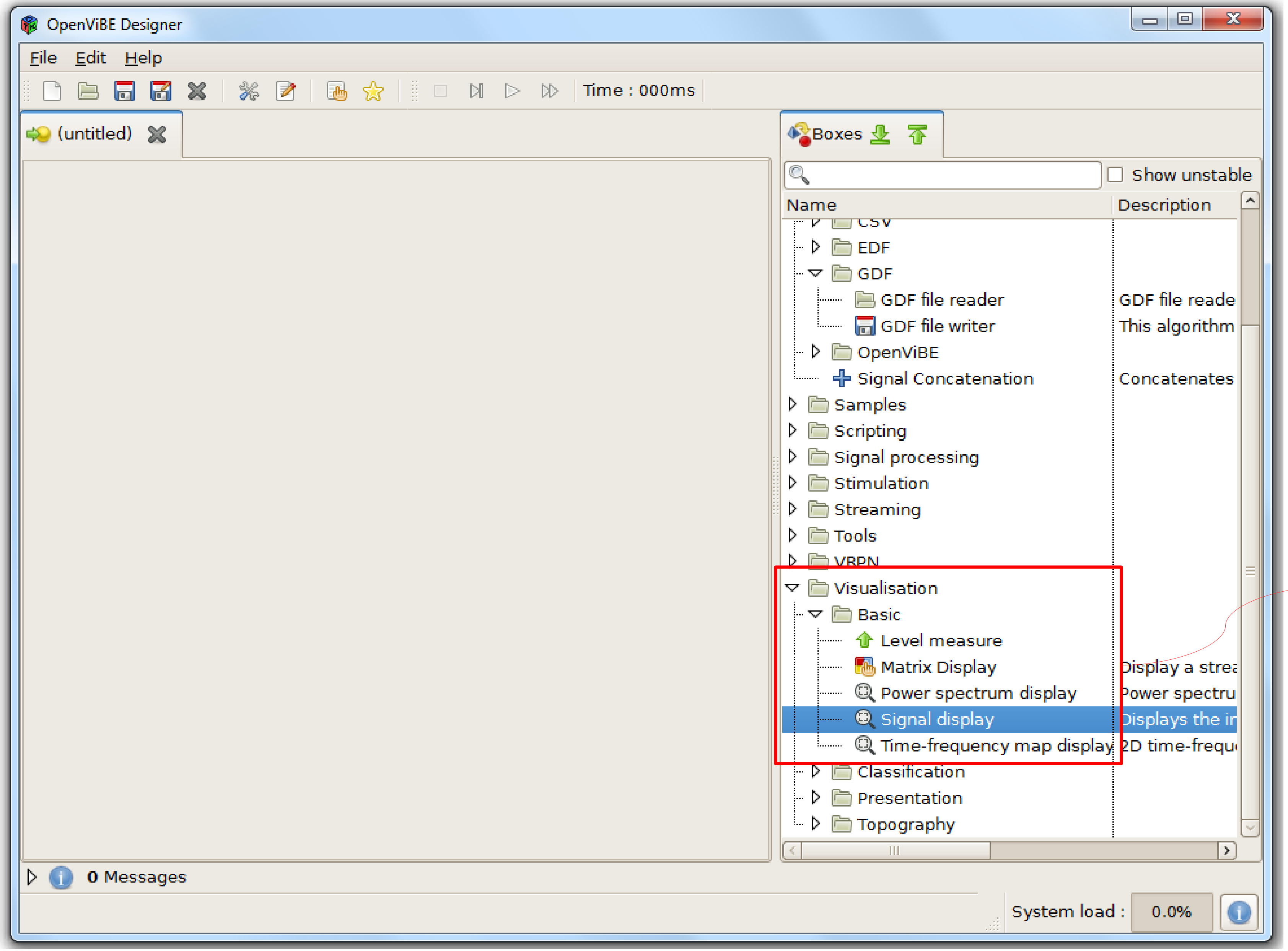
Step 1: add reader and display boxes



File reading and writing boxes

Reading a pre-recorded file

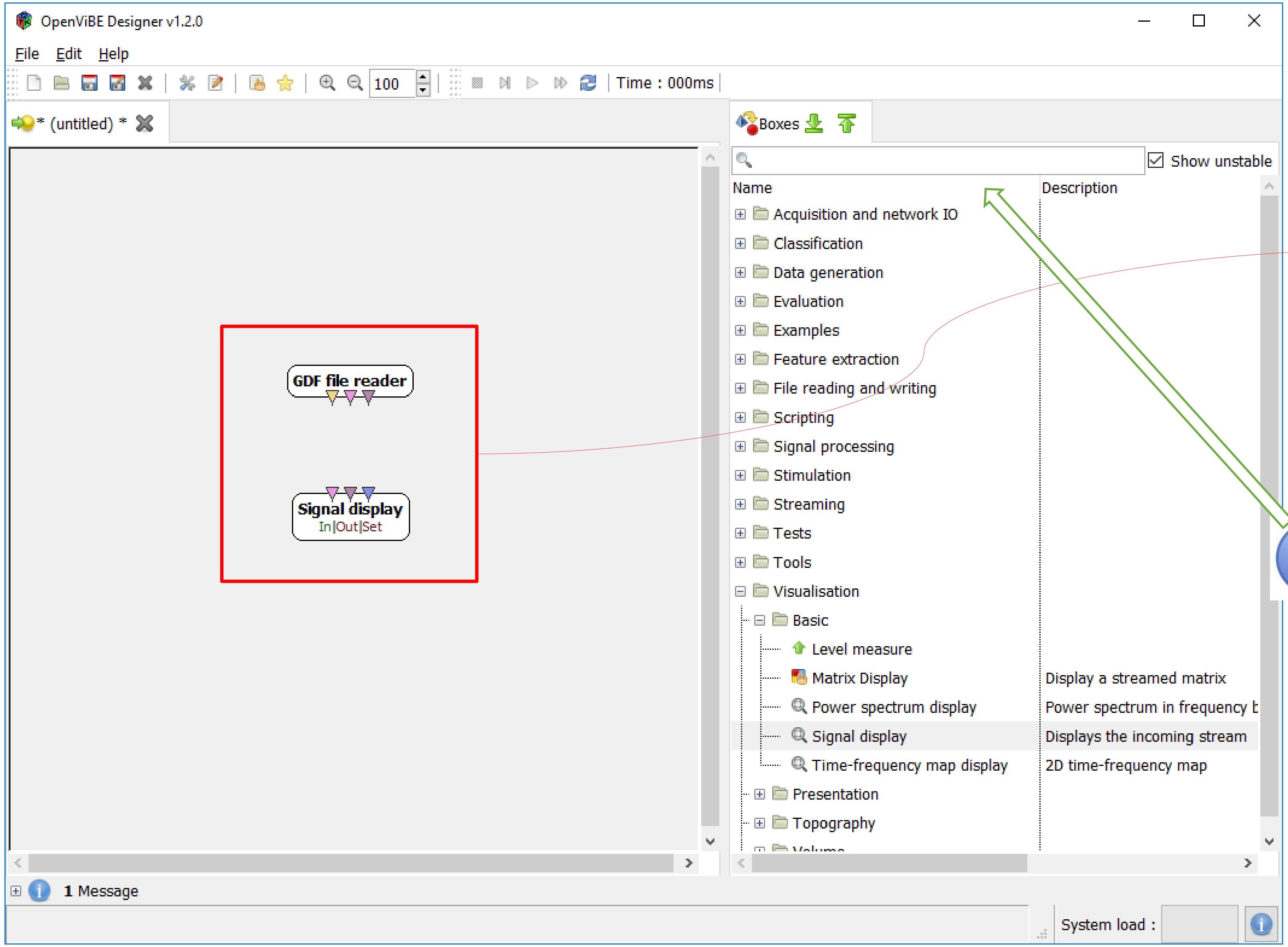
Step 1: add reader and display boxes



Basic visualisation boxes

Reading a pre-recorded file

Step 1: add reader and display boxes

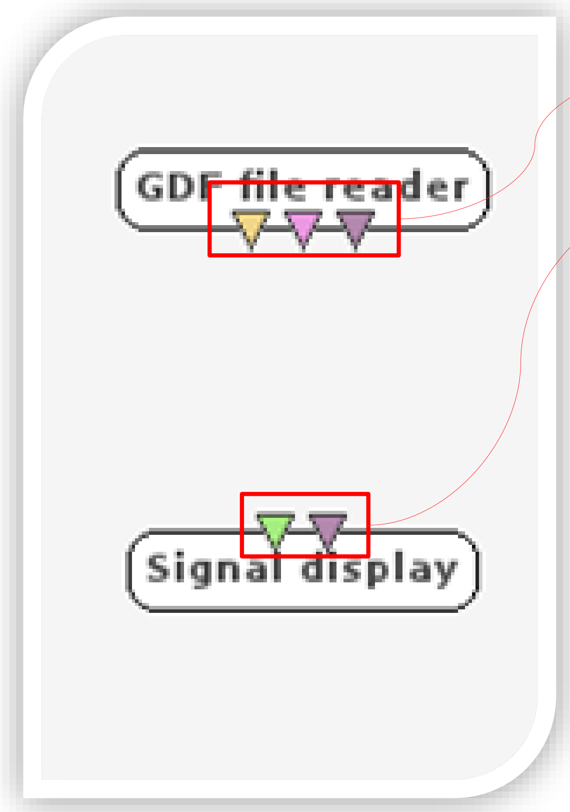


Drag & Drop the *GDF File Reader* and *Signal Display* boxes in the edition area

You can also look for a specific box from the search bar if you know its name and do not remember the category

Reading a pre-recorded file

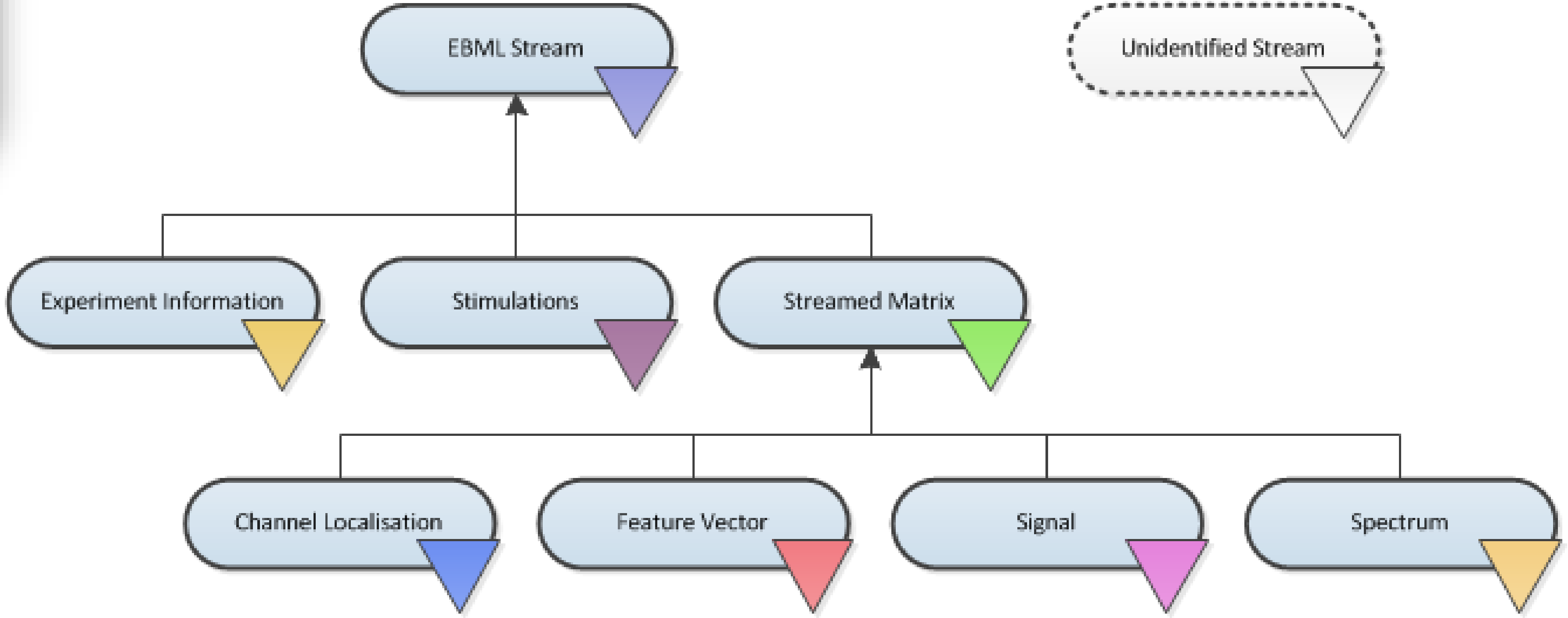
Step 2: connect inputs and outputs



Boxes outputs

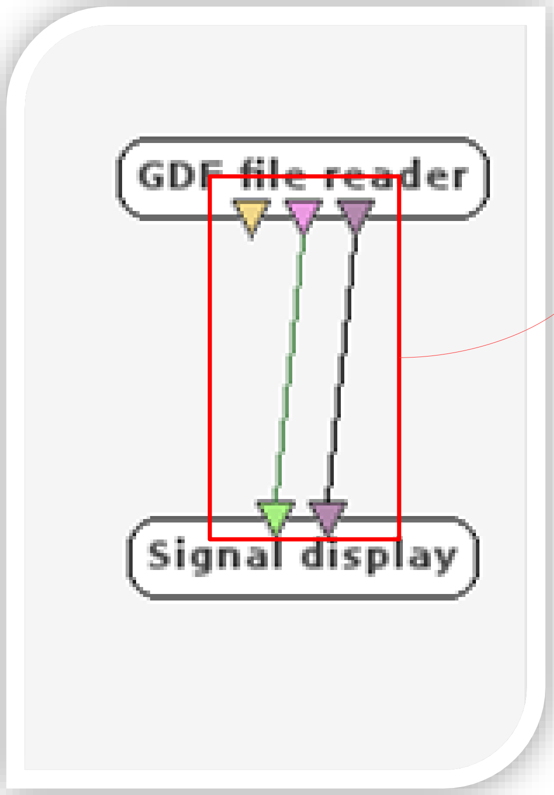
Boxes inputs

I/O types are organized in a hierarchy and coded with specific colors as follows :



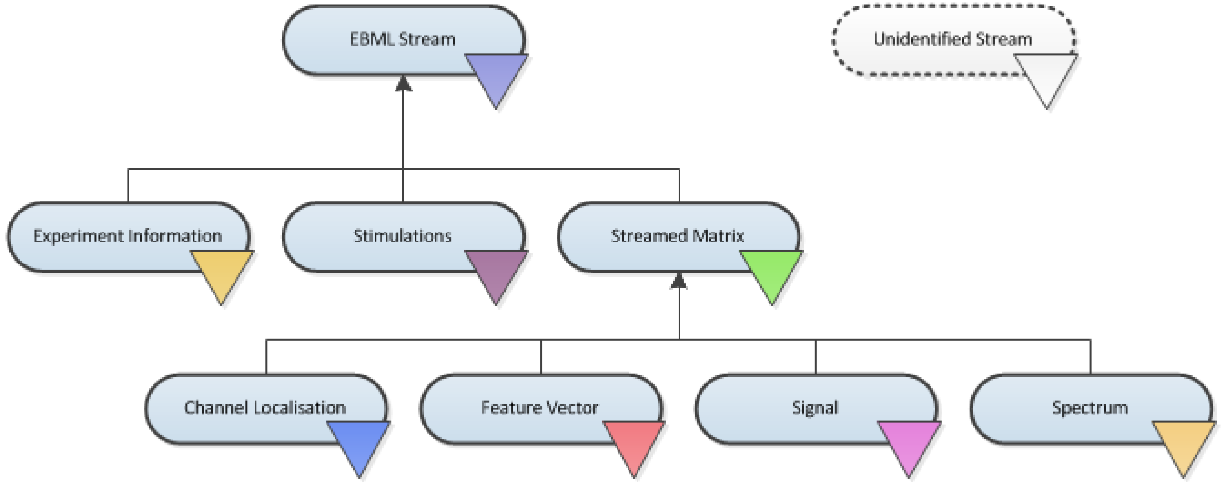
Reading a pre-recorded file

Step 2: connect inputs and outputs



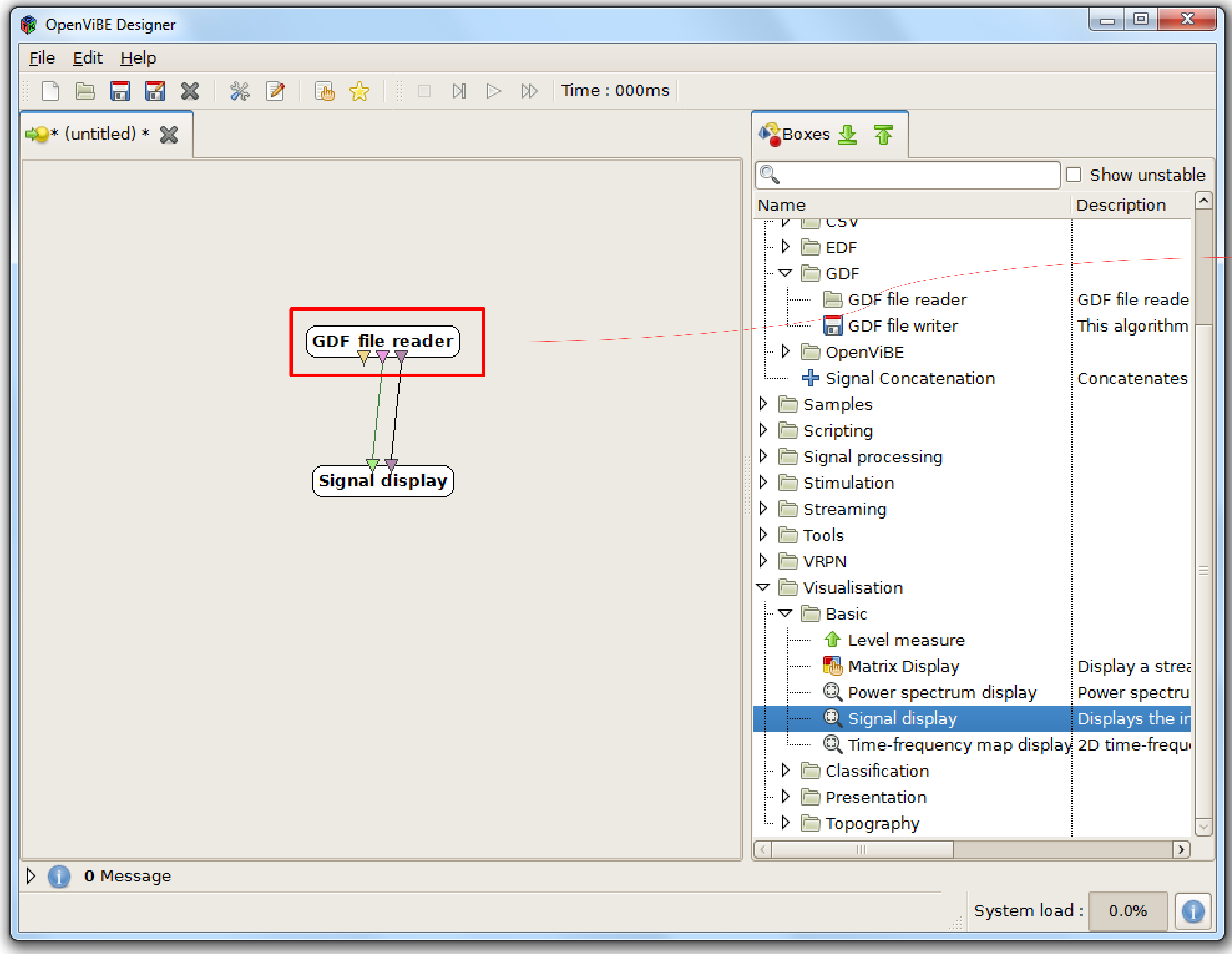
Connect the inputs of the *signal display* box to the outputs of the ***GDF file reader box*** **by clicking an input and dragging / releasing on the corresponding output**

- Inputs can receive connections from any outputs with compatible type (same type or derived type)
- You can connect an output to multiple inputs but you can't connect multiple outputs to a single input.



Reading a pre-recorded file

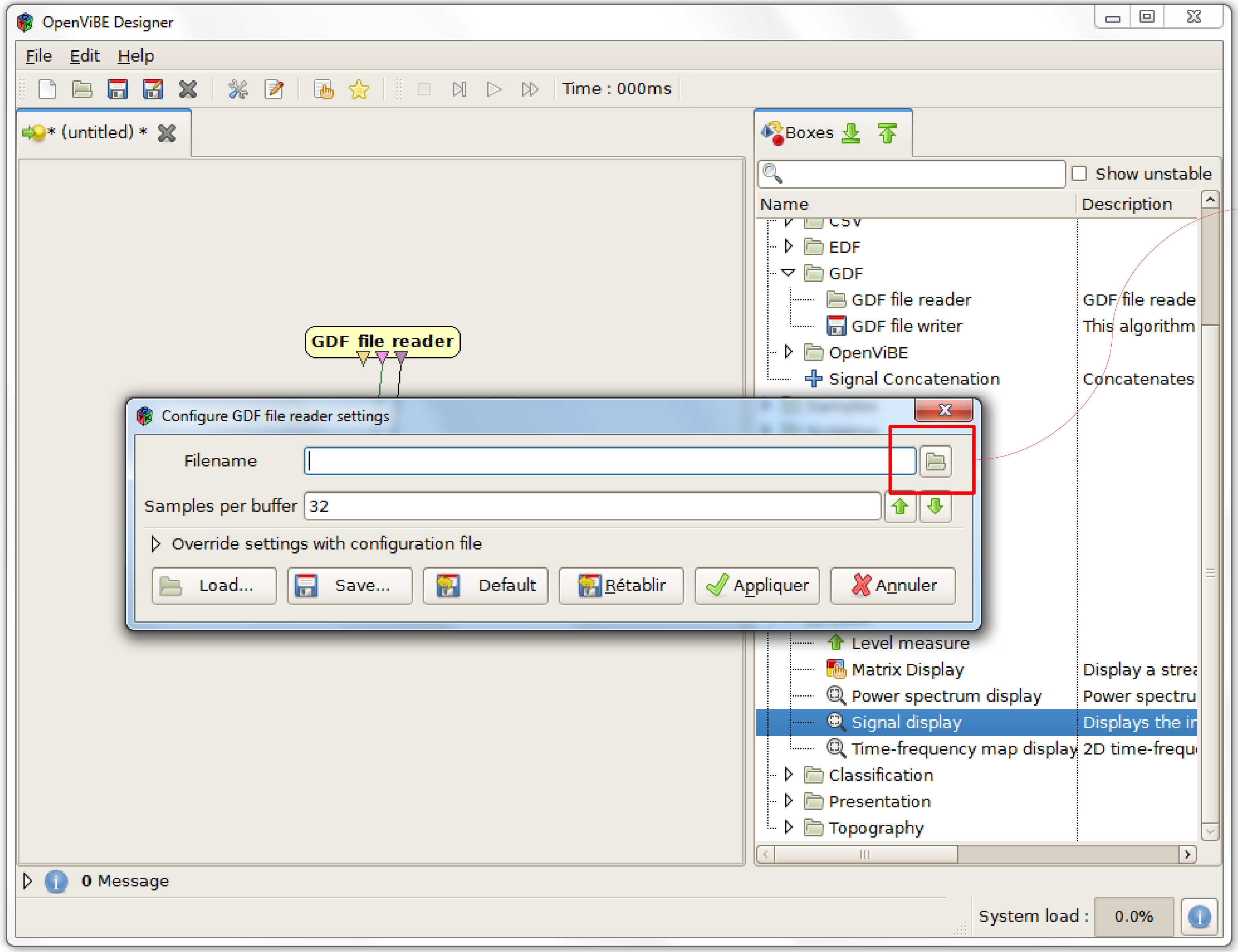
Step 3: configure boxes



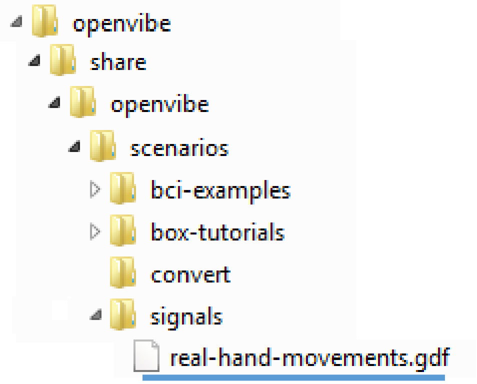
Double click on the *GDF file reader* box to open its configuration dialog

Reading a pre-recorded file

Step 3: configure boxes



Select a GDF file

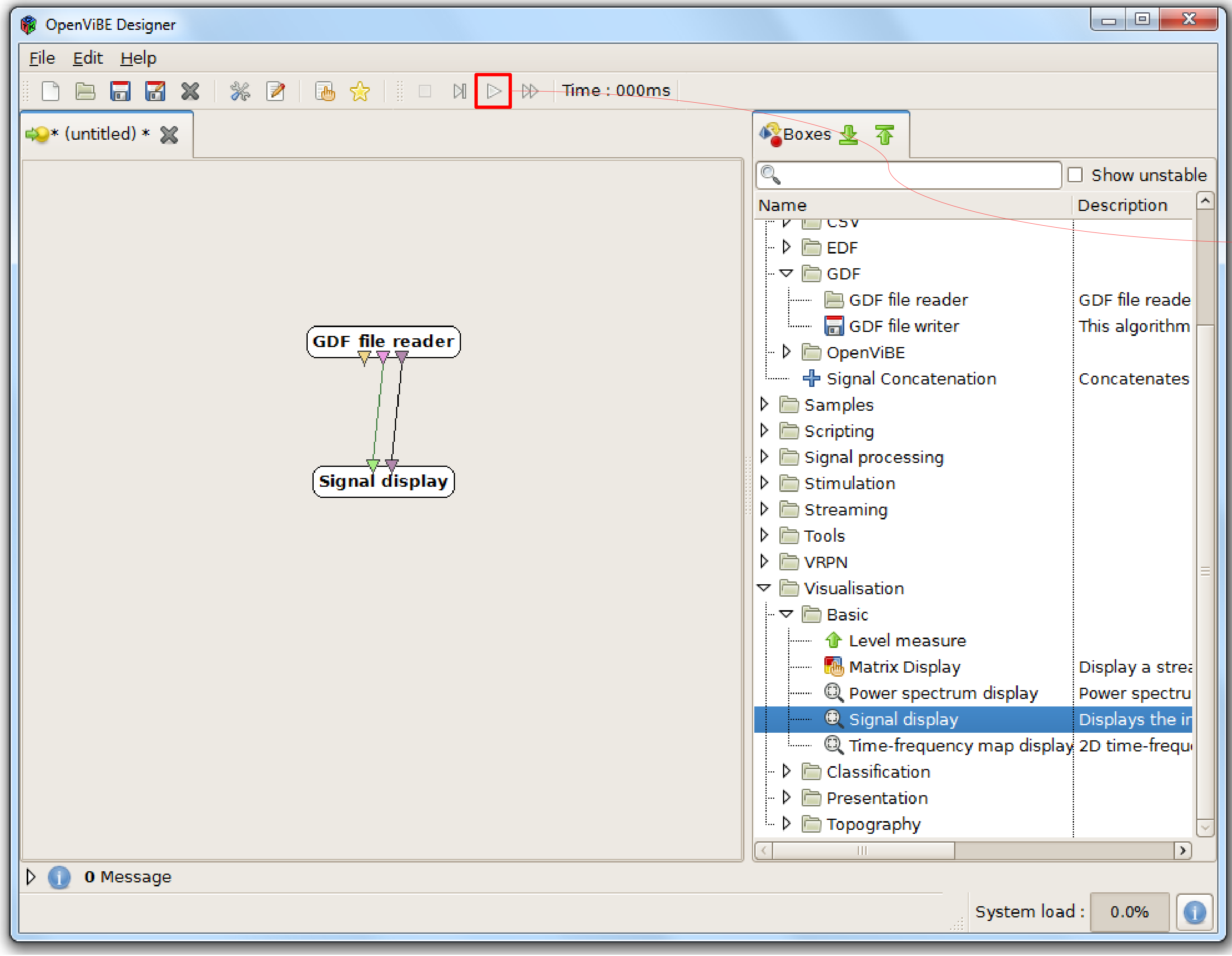


Then click on *Apply*

This file contains 9 minutes of signal sampled at 512 Hz. The box will split these signals in blocks of 32 samples, which represents a 16th of second per block

Reading a pre-recorded file

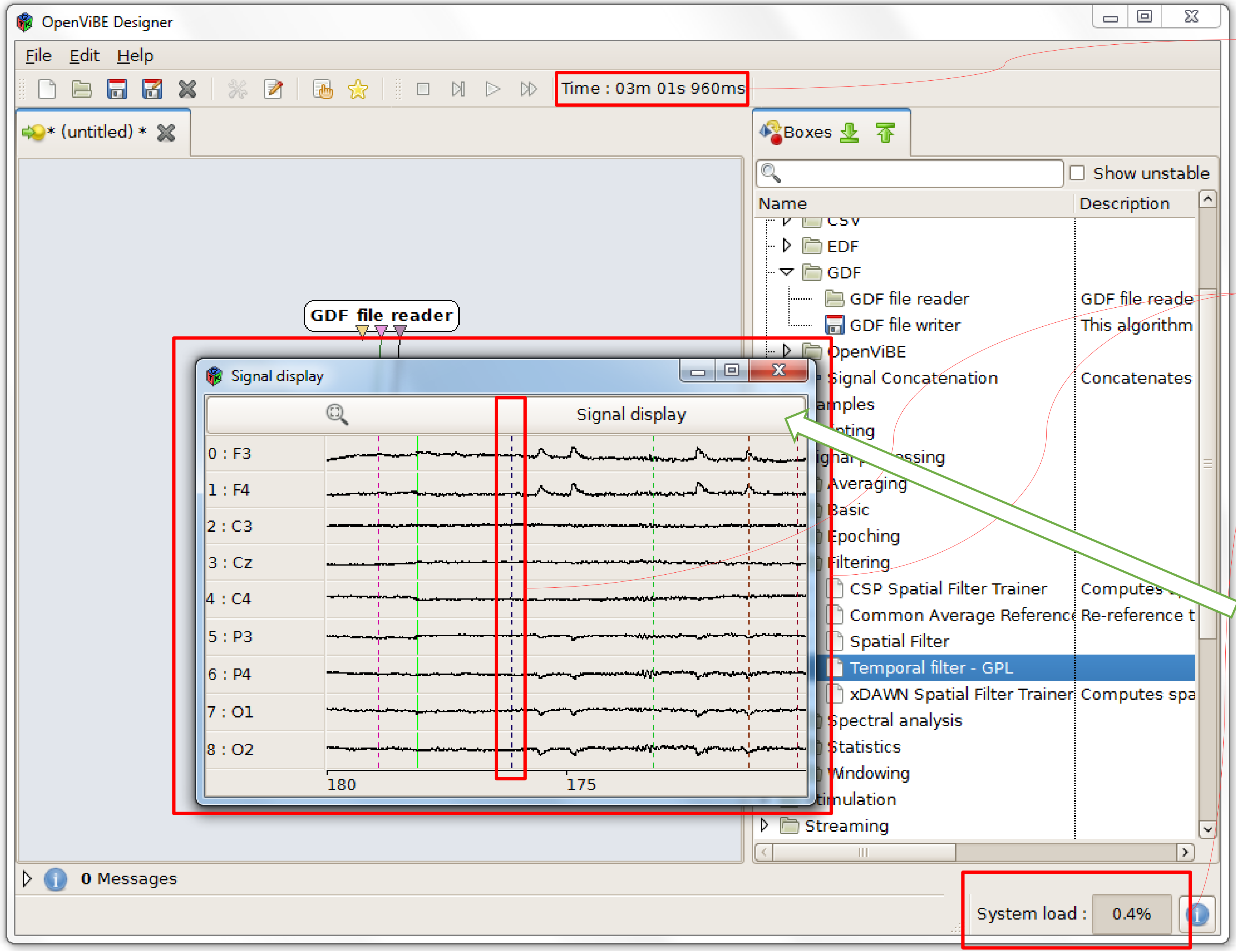
Step 4: test, play, explore scenario



Start the scenario clicking the *Play* button

Reading a pre-recorded file

Step 4: test, play, explore scenario



Current time

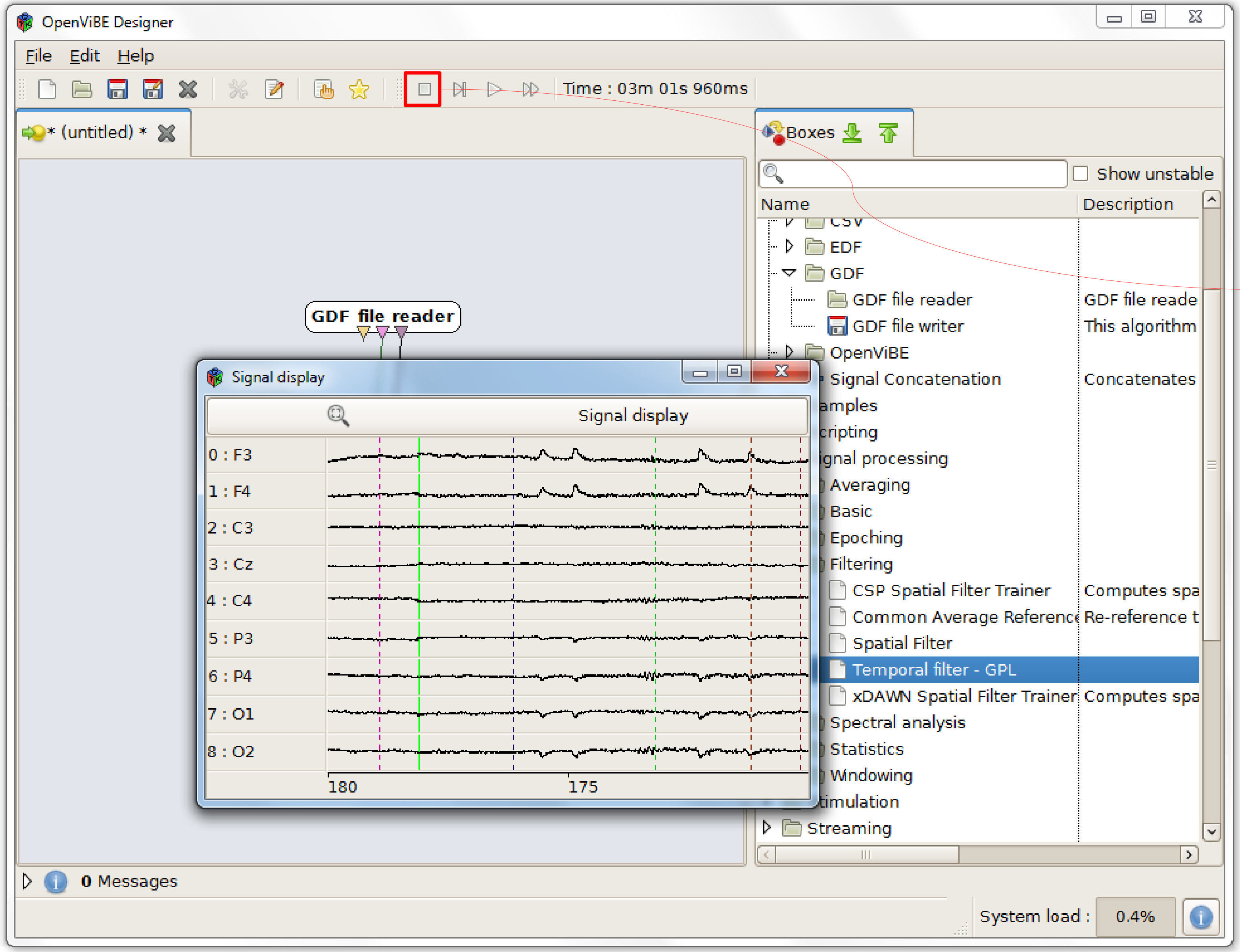
The file is read and displayed in real-time: signal and stimulations (event).

Processor load

Options and information are available when clicking on the signal display toolbar (scales, channel selection, color codes, etc.)

Reading a pre-recorded file

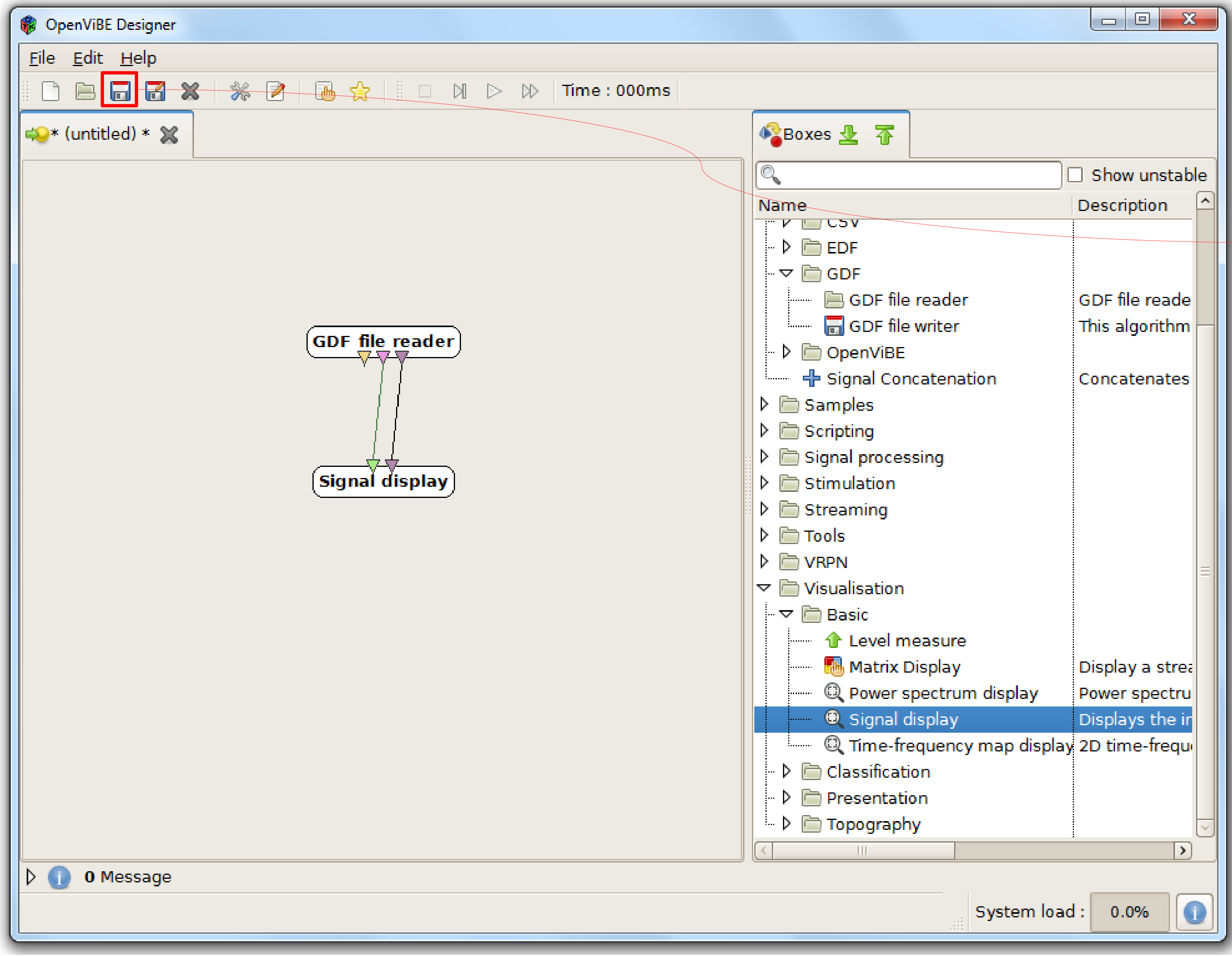
Step 4: test, play, explore scenario



Stop the scenario by clicking the *Stop* button

Reading a pre-recorded file

Step 4: test, play, explore scenario



You can save the scenario on the disk for later use

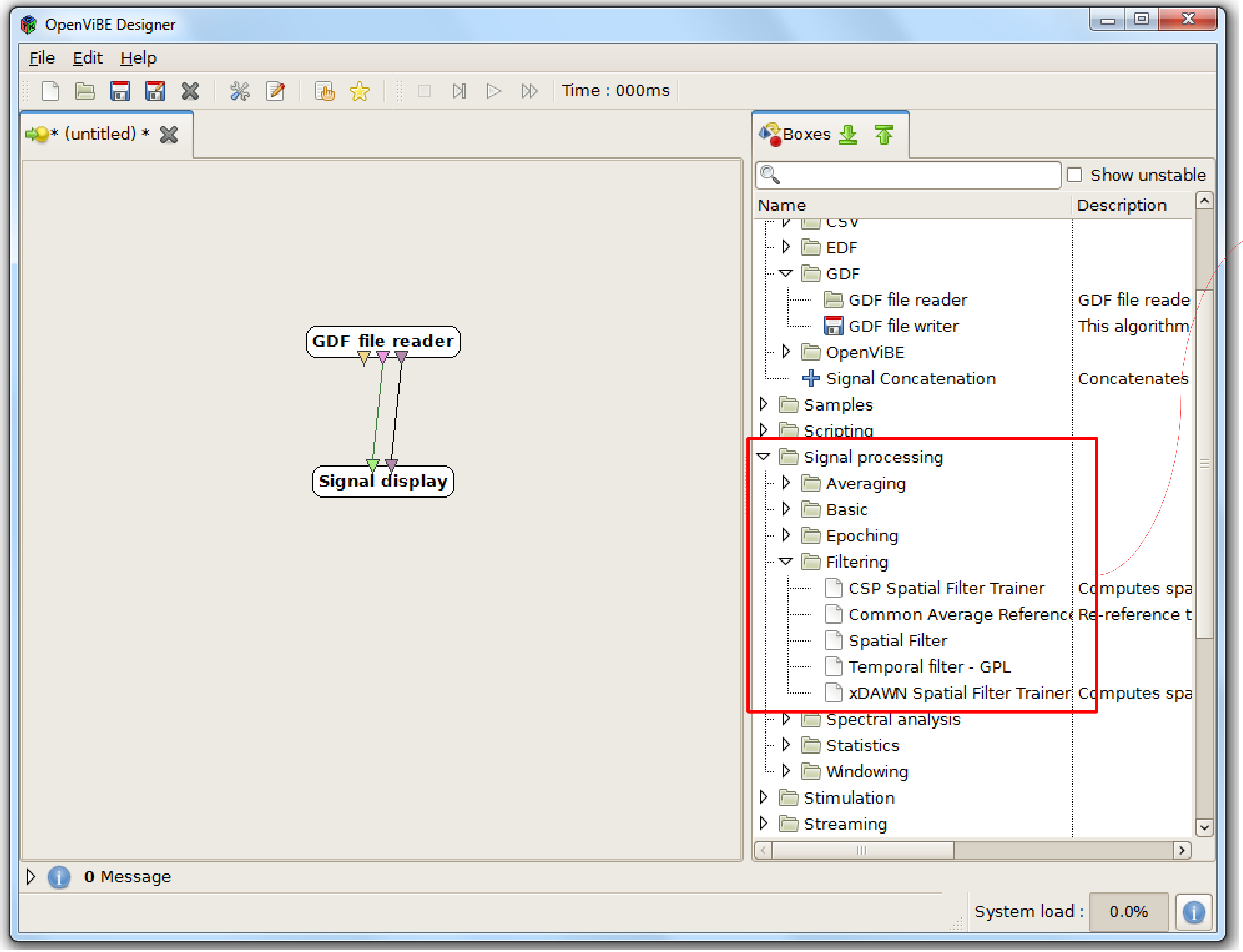
Good practice: save your scenarios on a regular basis

EEG Signal Filtering

Displaying alpha band power

EEG Signal Filtering

Step 1: add temporal filter box



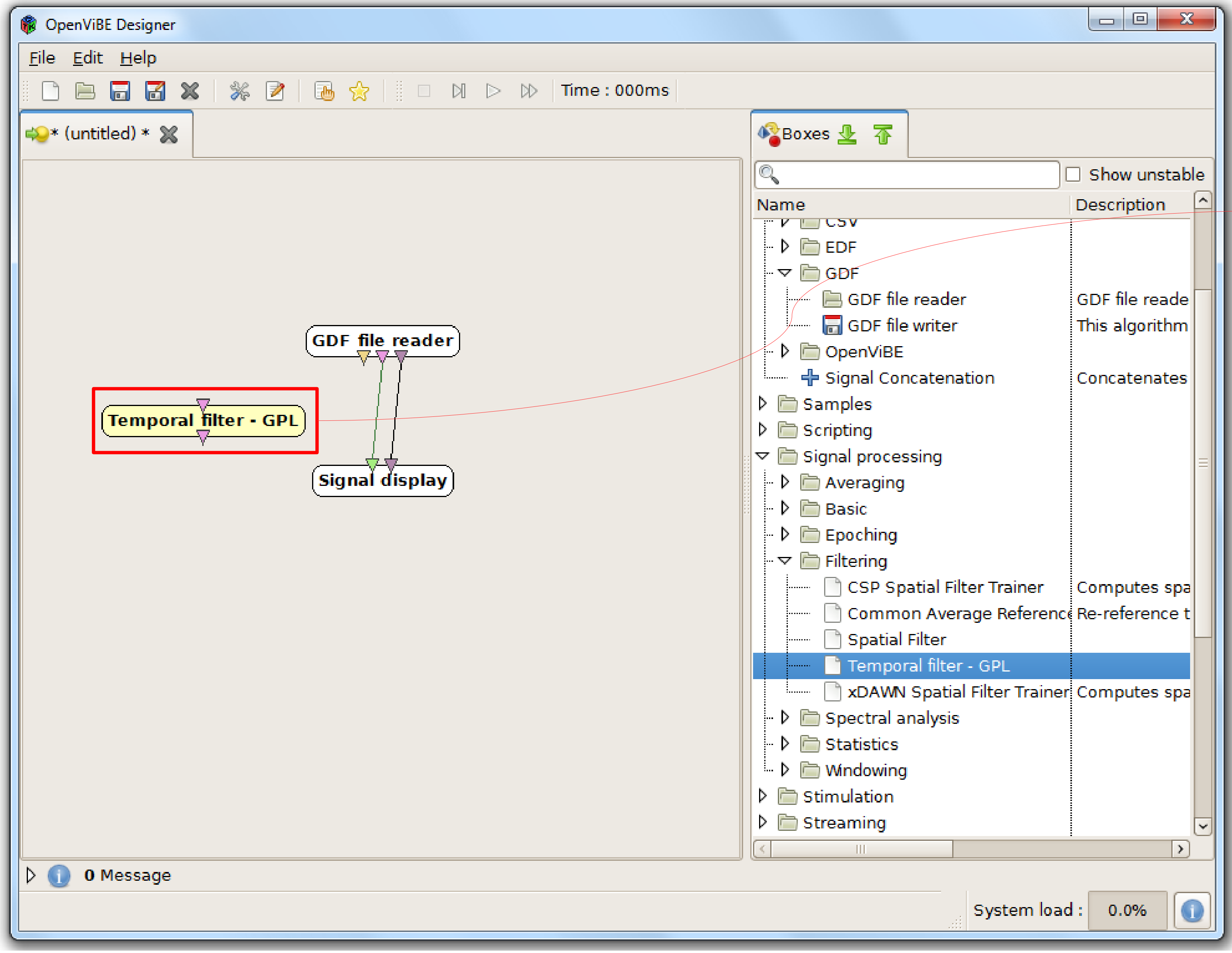
Categories

> *signal processing*

> *Filtering*

EEG Signal Filtering

Step 1: add temporal filter box

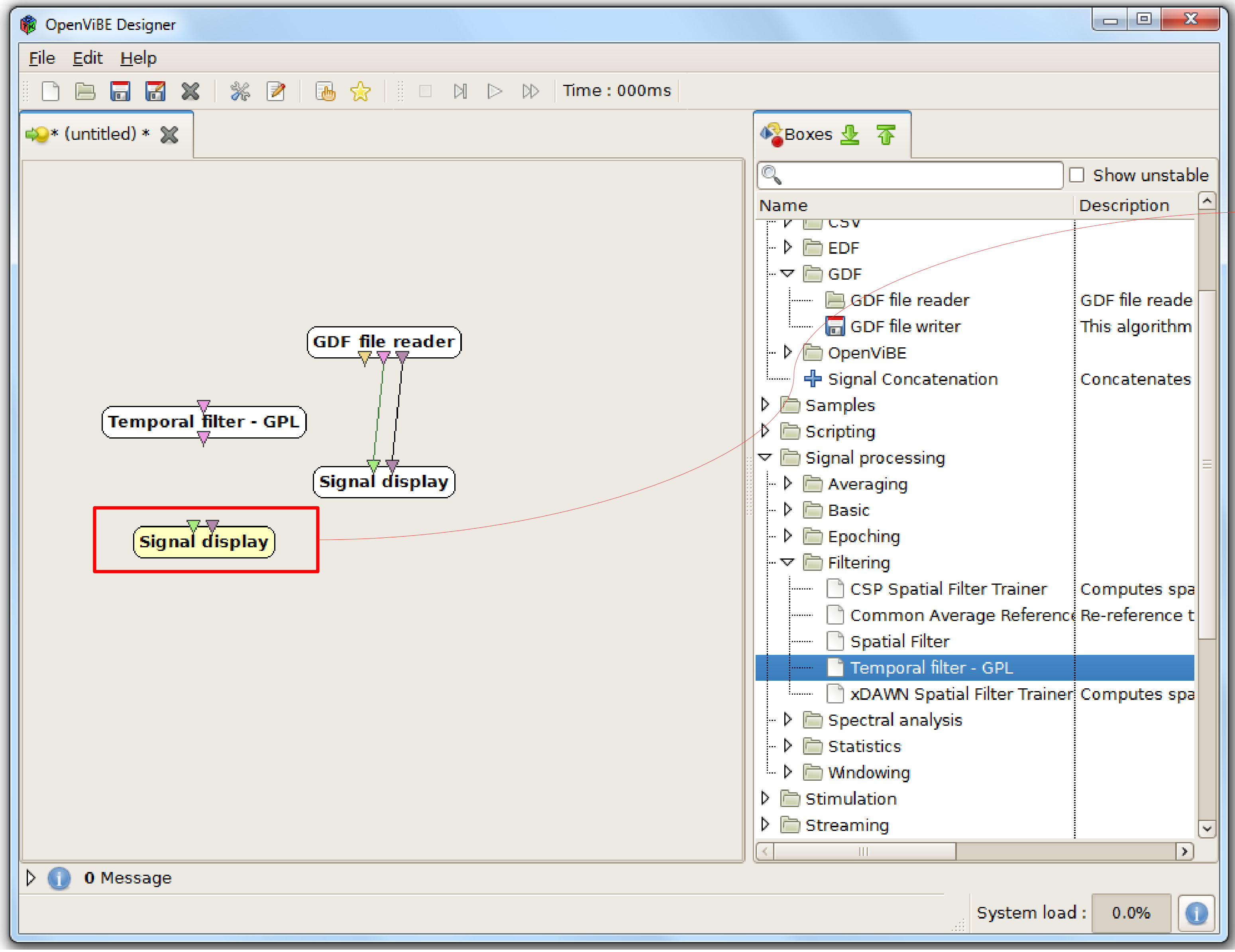


Drag & Drop a *Temporal Filter* box

You can access online documentation of individual boxes by selecting box of interest and pressing F1.
This requires a connection to the Internet.

EEG Signal Filtering

Step 2: add display box



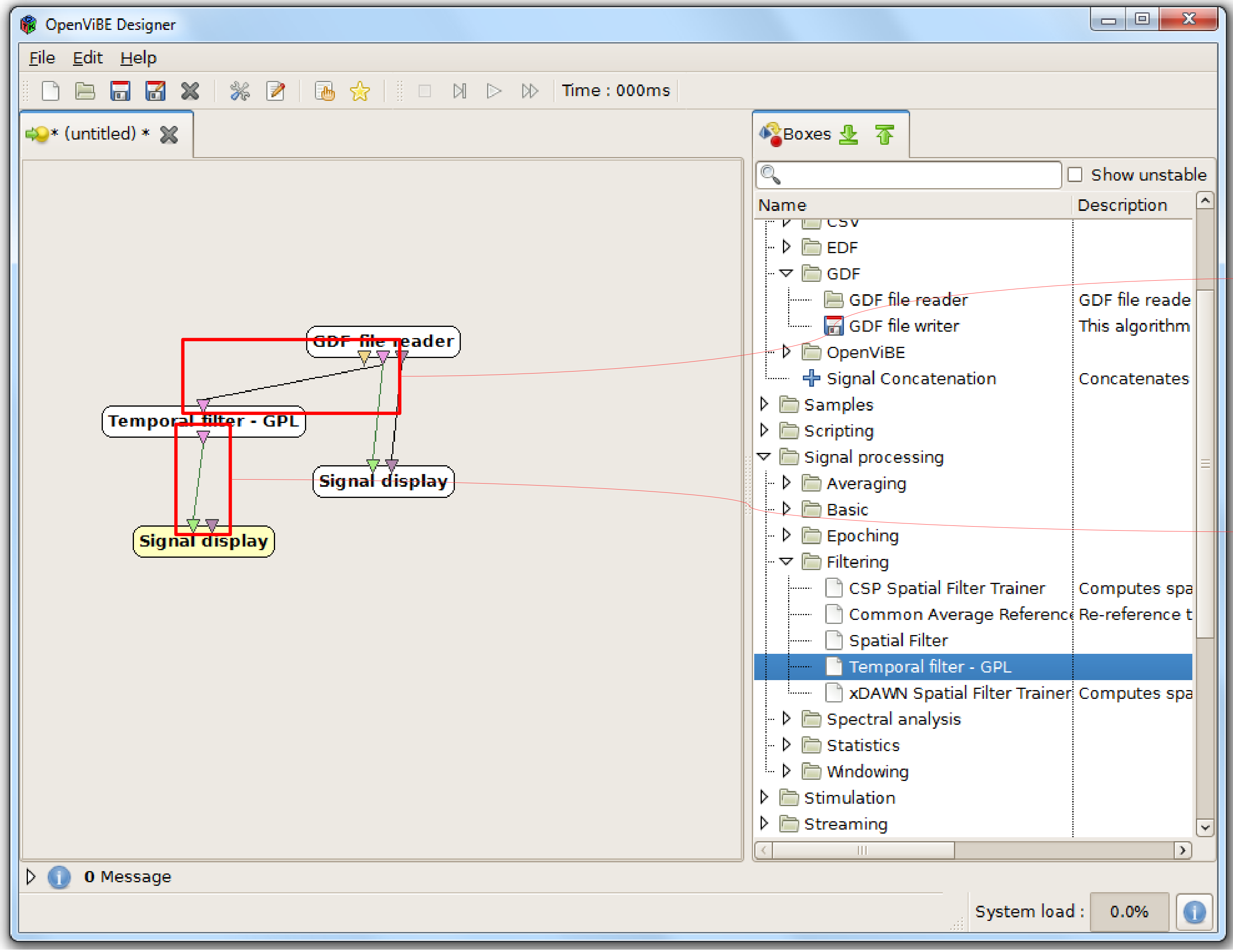
Copy / Paste a new *Signal Display* box

Copy / Paste can be done through three different pathways :

- *Edit* Menu
- Right click
- Ctrl-c/Ctrl-x/Ctrl-v

EEG Signal Filtering

Step 3: connect inputs and outputs

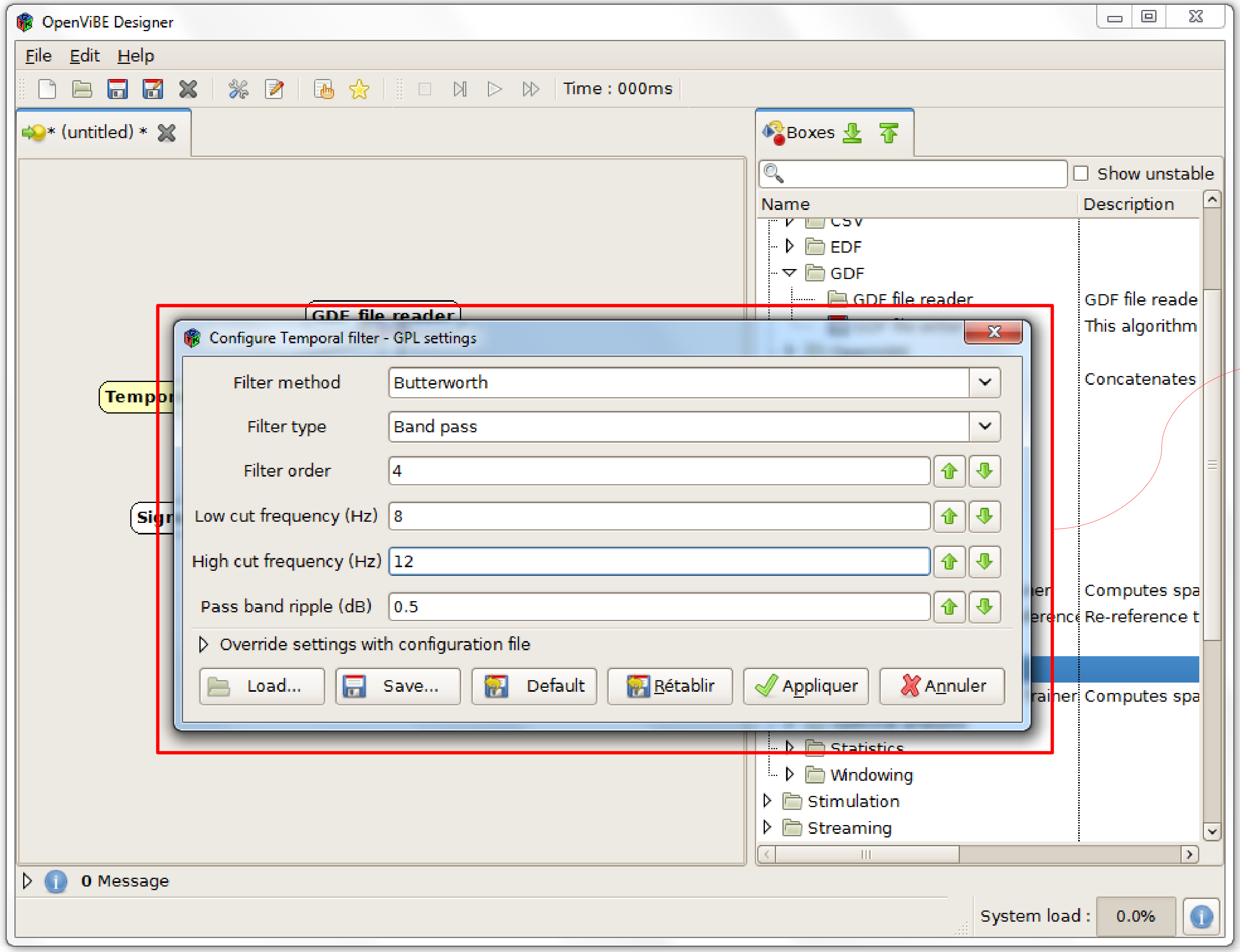


Connect the input of the *Temporal Filter* box to the output of the *GDF file reader* box

Connect the output of the *Temporal Filter* box to the input of the *Signal Display* box

EEG Signal Filtering

Step 4: configure boxes



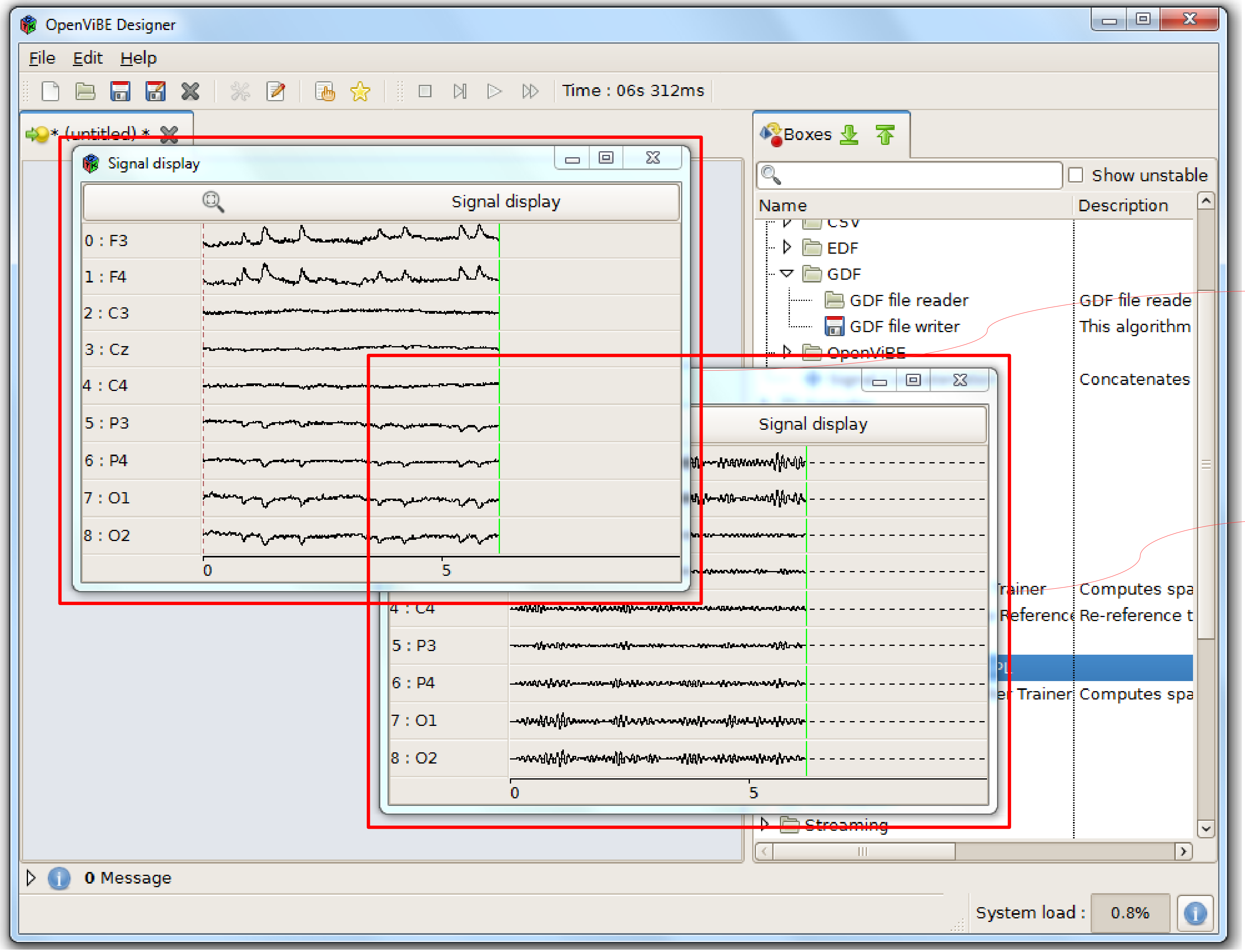
Configure the *Temporal Filter* box:

- Band-pass
- Low-cut : 8 Hz
- High-cut : 12 Hz

The signal is filtered in the alpha band (8 to 12 Hz)

EEG Signal Filtering

Step 5: play scenario

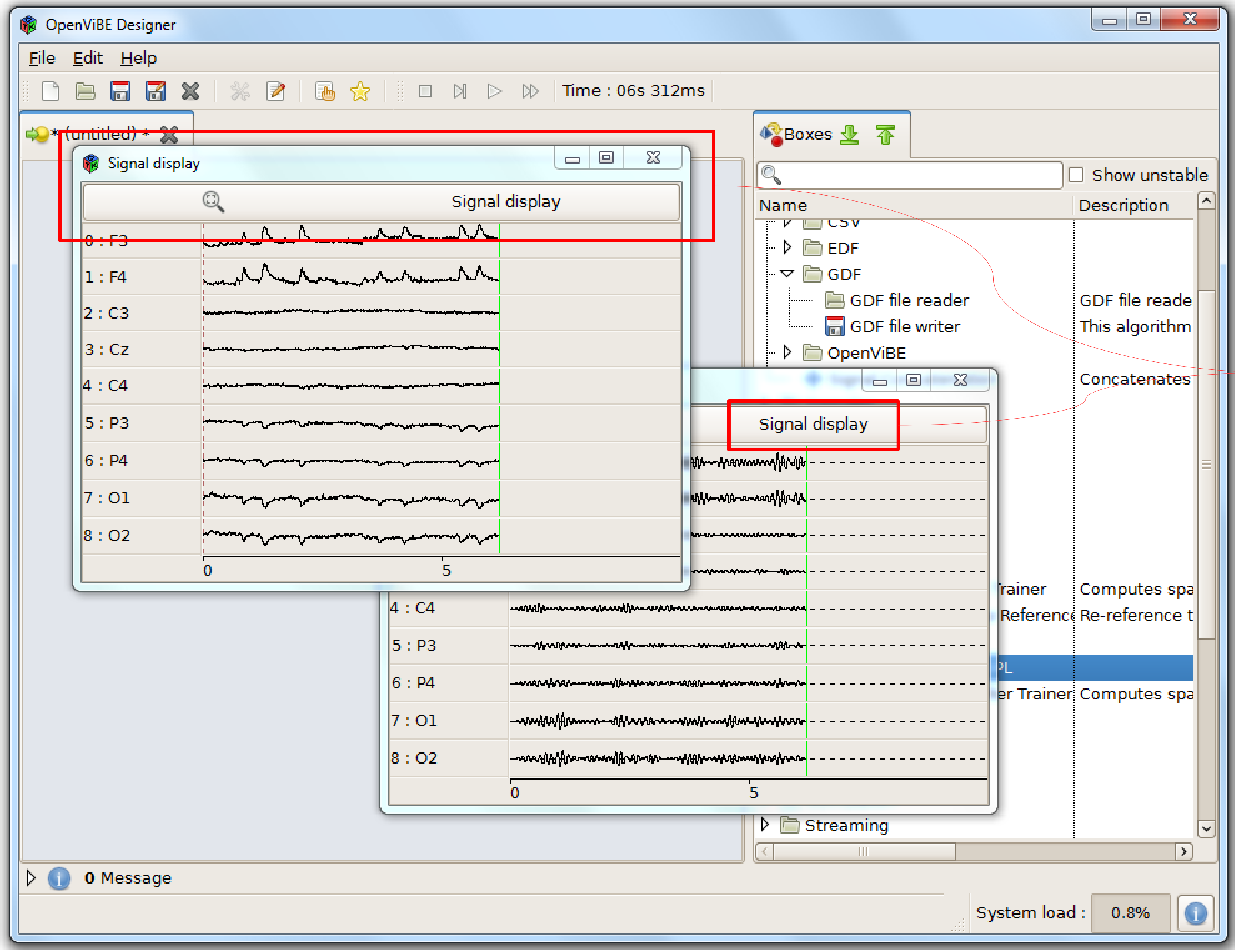


The raw signal (unfiltered) is still displayed

The filtered signal is displayed in a new window

EEG Signal Filtering

Step 5: play scenario

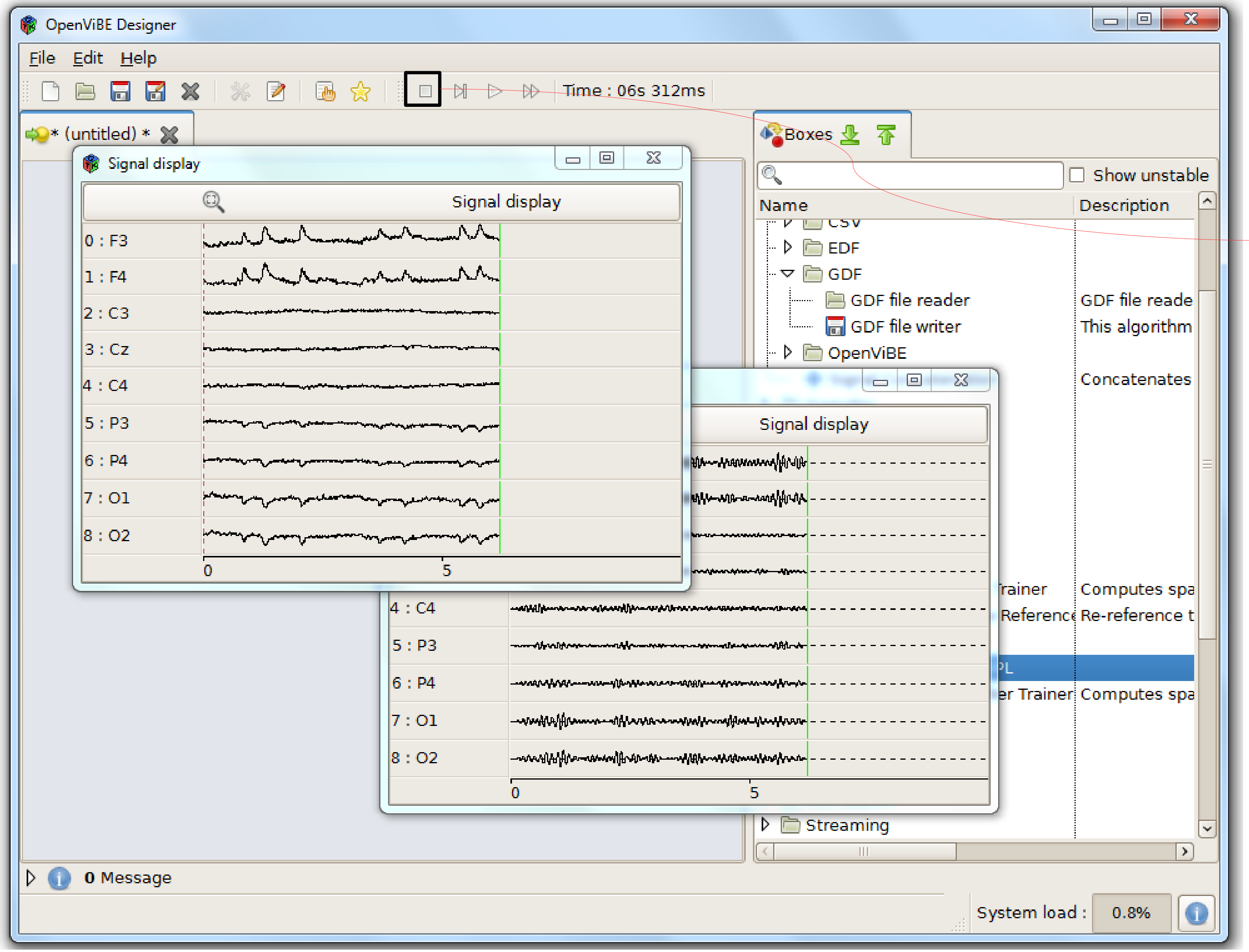


The windows have the same title…

Their (automatic) placement is not very convenient…

EEG Signal Filtering

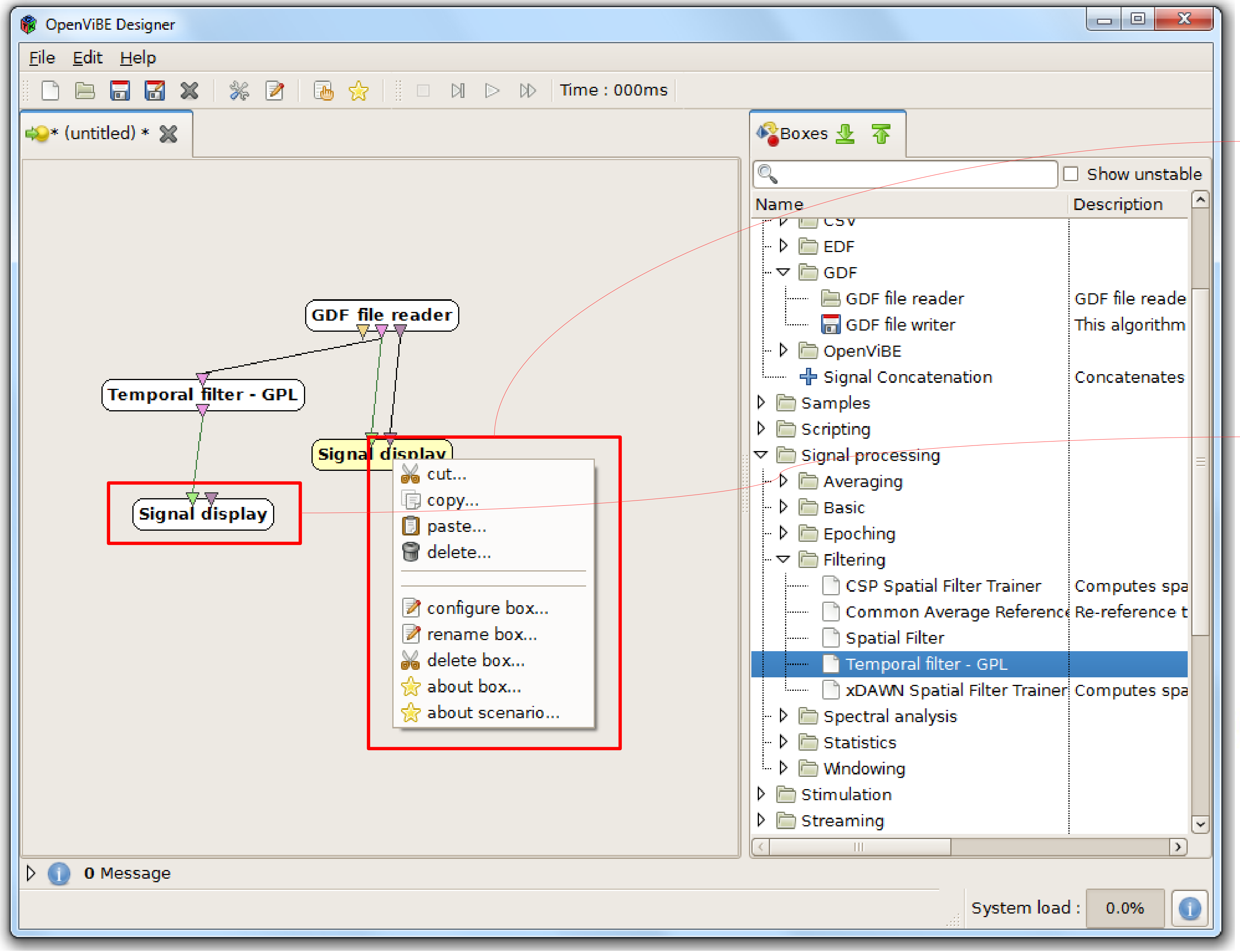
Step 5: play scenario



Stop the scenario by clicking the *Stop* button

Handling visualisation widgets

Handling visualisation widgets

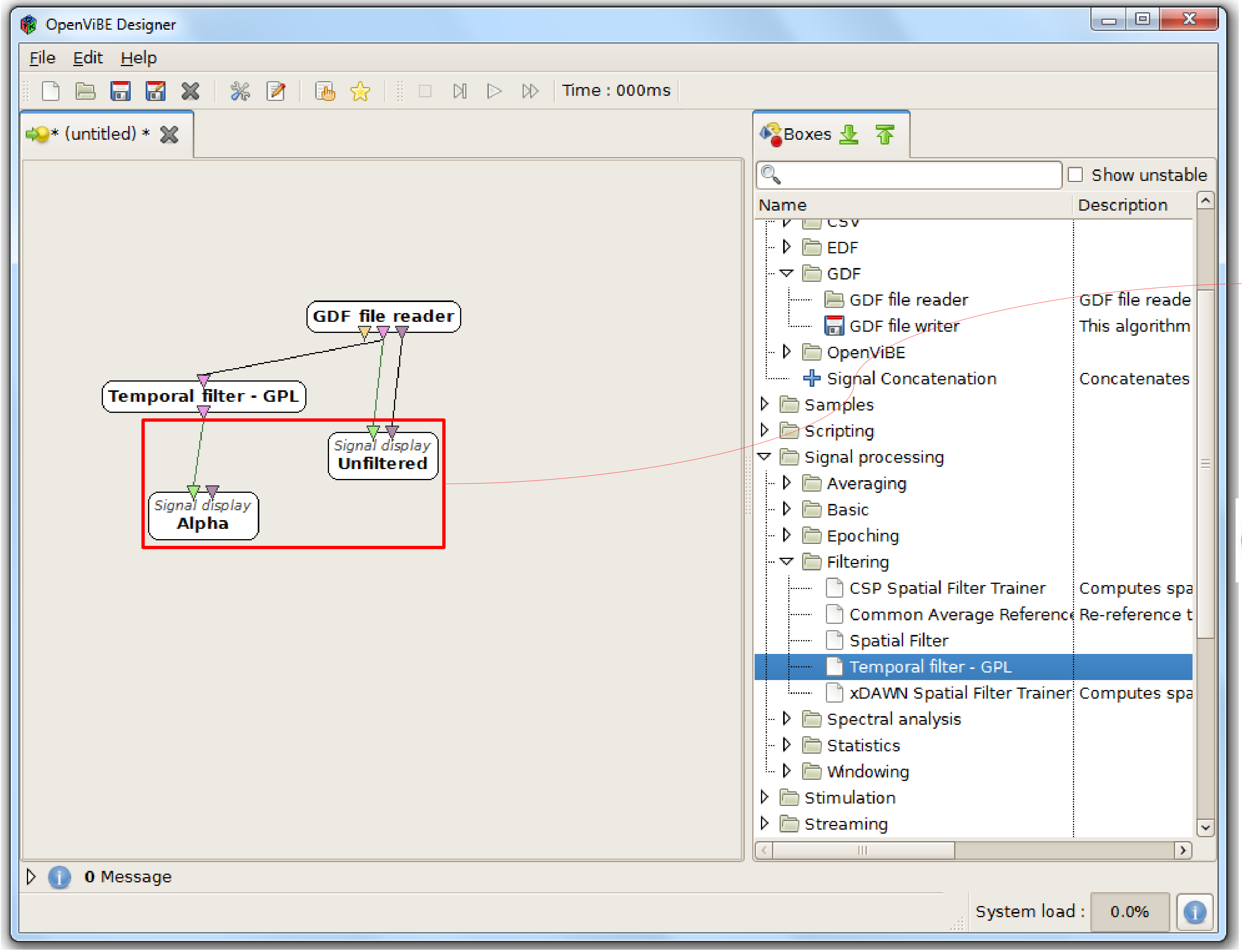


Right click on the visualisation box that receives raw signals and rename it to “unfiltered” using the *rename box* option

Rename the other visualisation box to “alpha ”

You can rename a selected box by pressing F2

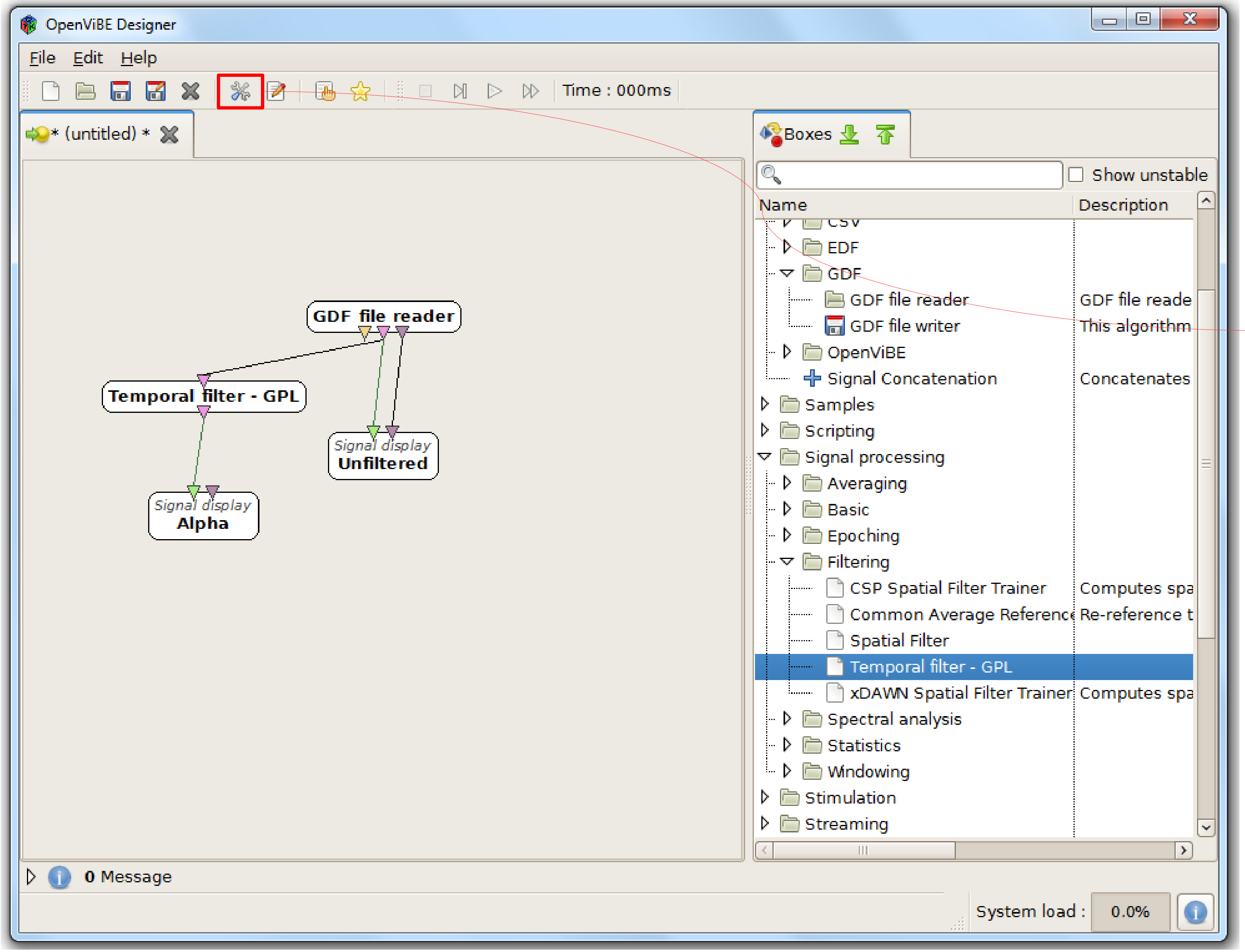
Handling visualisation widgets



We now have « unfiltered » and « alpha » visualisation boxes

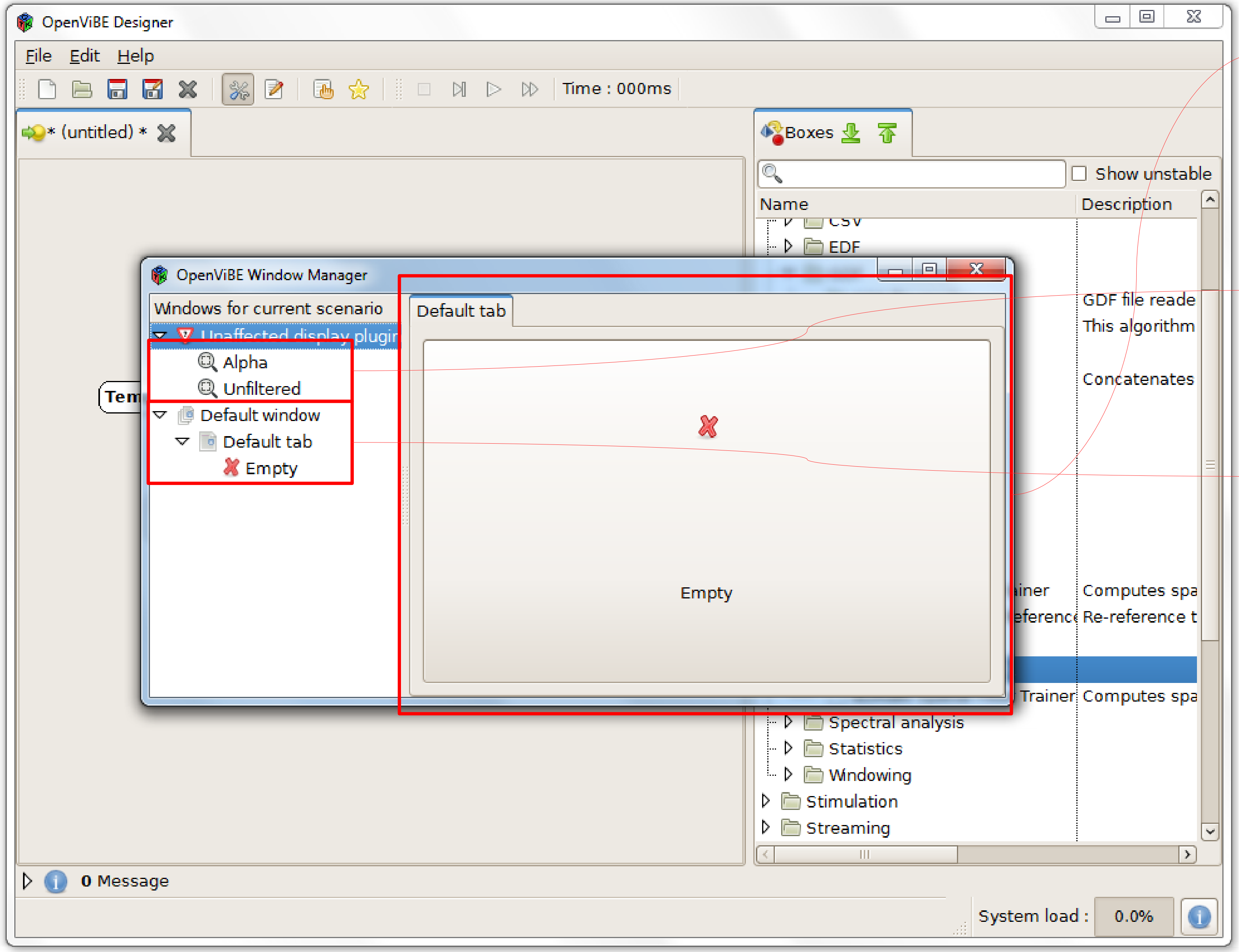
The original name of the box is still visible on top of the new name

Handling visualisation widgets



Now open the « Window Manager »

Handling visualisation widgets

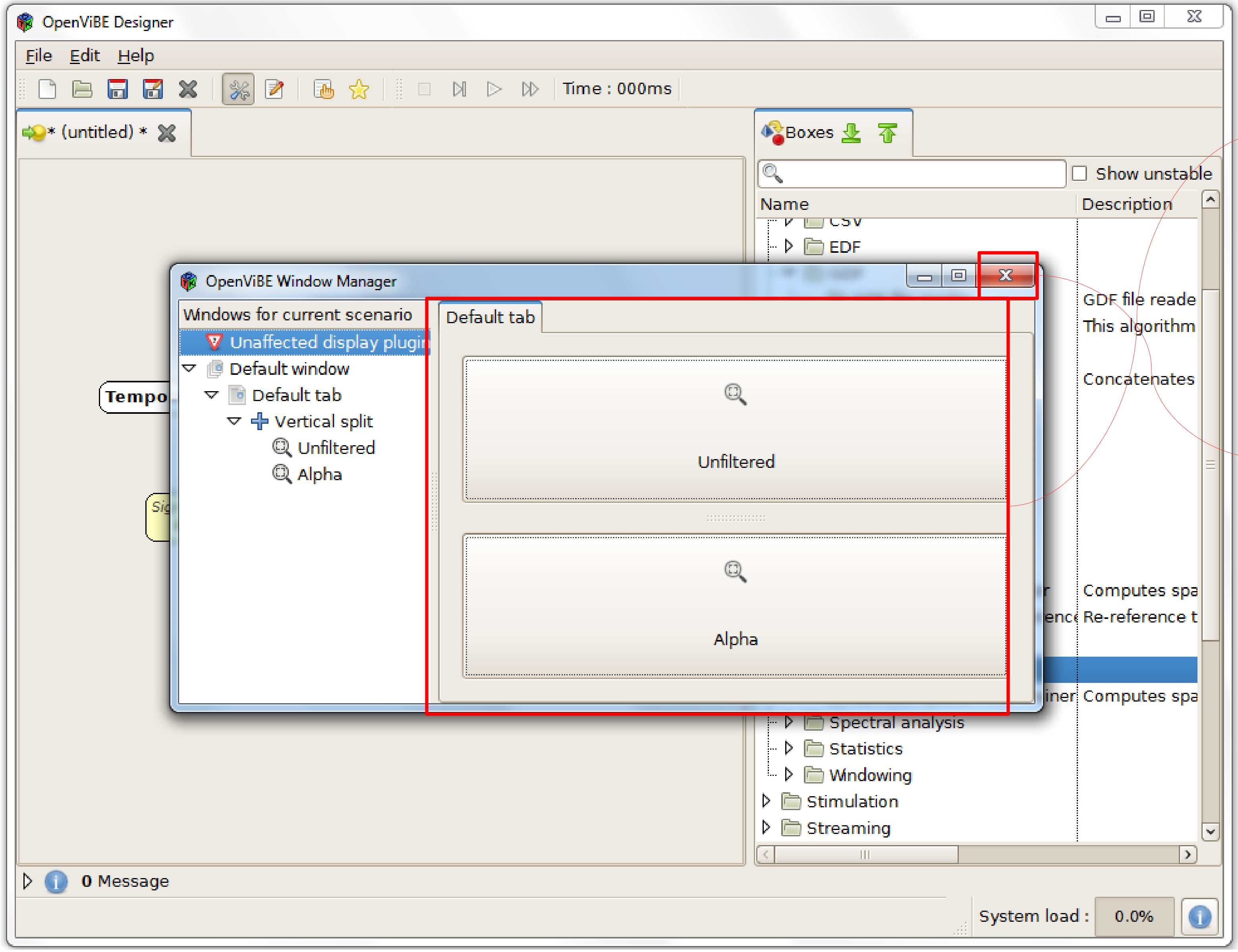


Widget arrangement area

List of unconfigured widgets

Treeview of configured widgets

Handling visualisation widgets

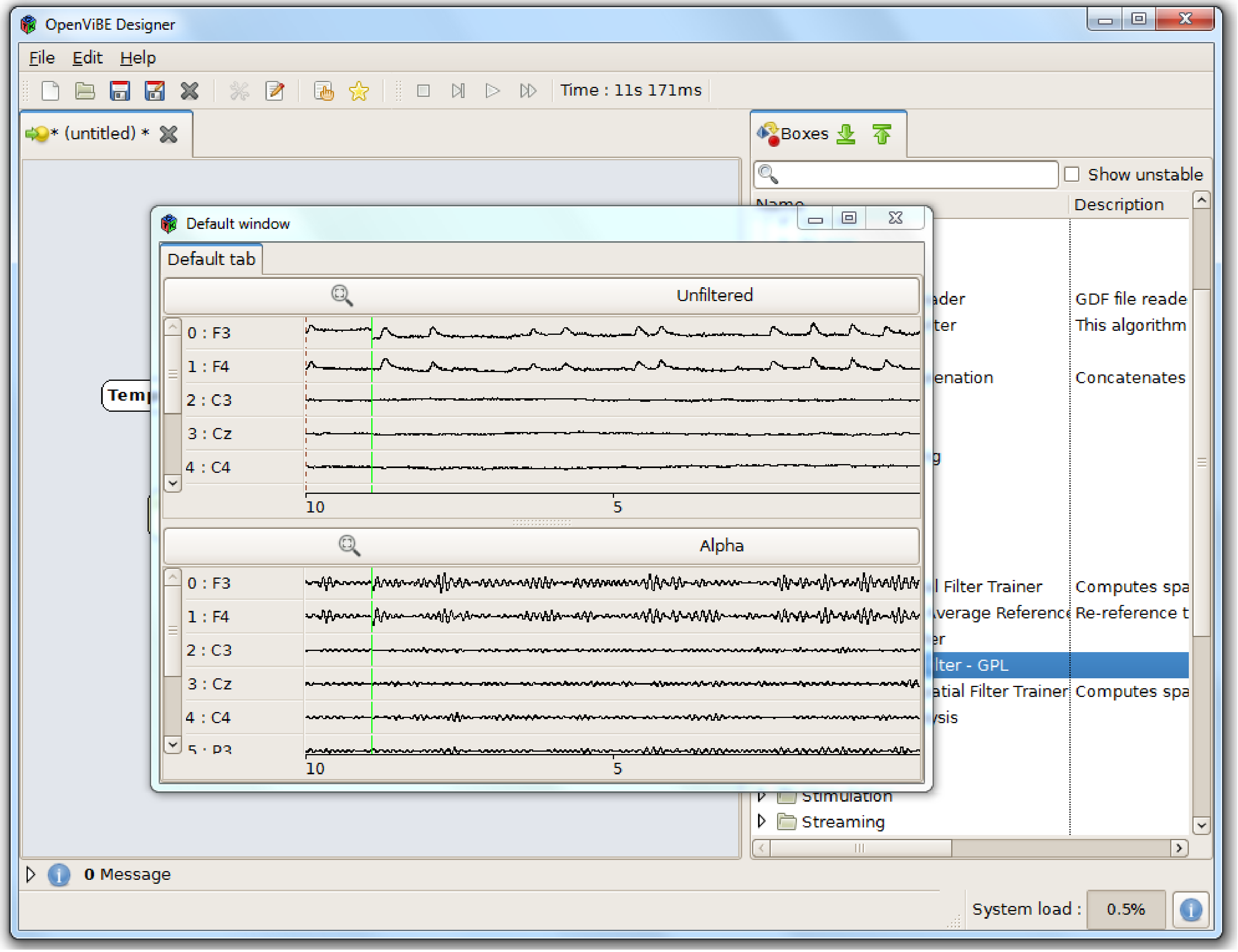


Drag & Drop the widgets in the arrangement area to place the *alpha* widget under the *unfiltered* widget

Close the *Window Manager* to apply changes.

You can create tabs and windows and name them at will. Just right click on the tree view for more actions.

Handling visualisation widgets



Start the scenario…

The widgets are now arranged in a more convenient way !

Datastream structure

Chunks and epoching

Simulated Time and box scheduling

- OpenViBE uses simulated time
  - The data is acquired and timed as a continuous flow
  - No data is lost as long as the acquisition process can sustain the real-time constraint
  - The data is processed by the Designer as fast as possible to reach real-time,

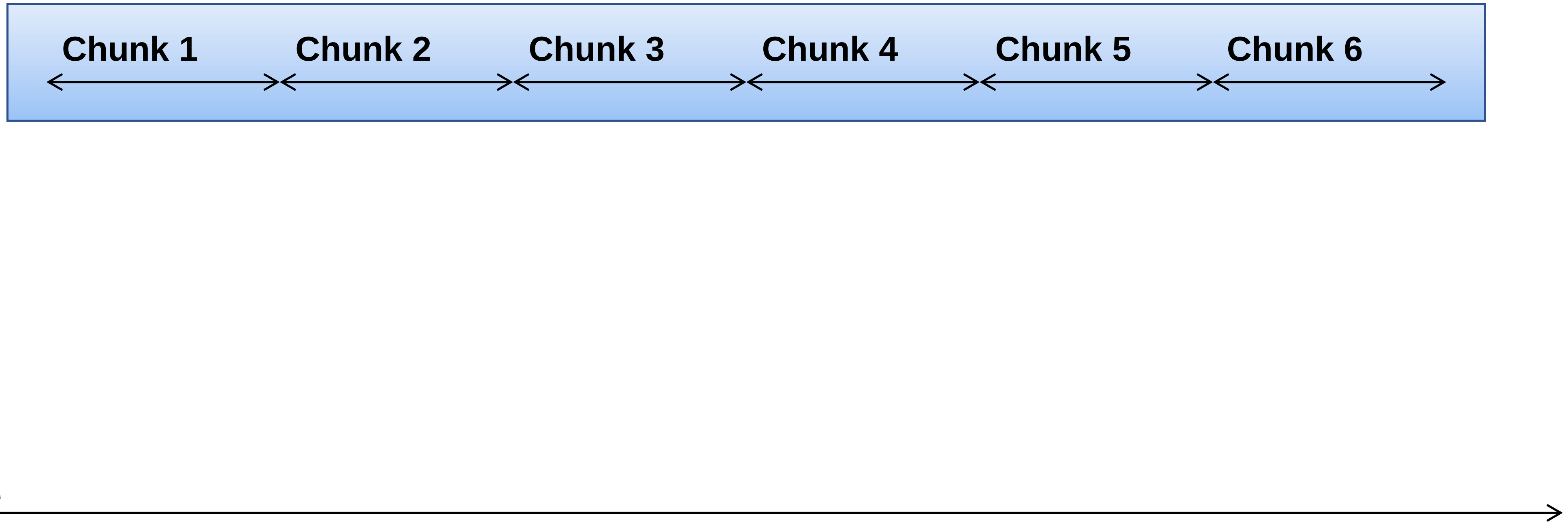
    i.e. **all the processing must be completed before a new block of data is received**

- The datastream is split in successive *chunks*
  - Each chunk as a *start* and *end* time reflecting the time period it represents
  - These *start* and *end* times are used by subsequent boxes for inter stream synchronization

- The OpenViBE kernel schedules the box processing
  - An order of execution is defined by the *scheduler*
  - Chunks are sent from boxes to boxes, and can pile up in the box input, waiting for the scheduler to call a step of process.

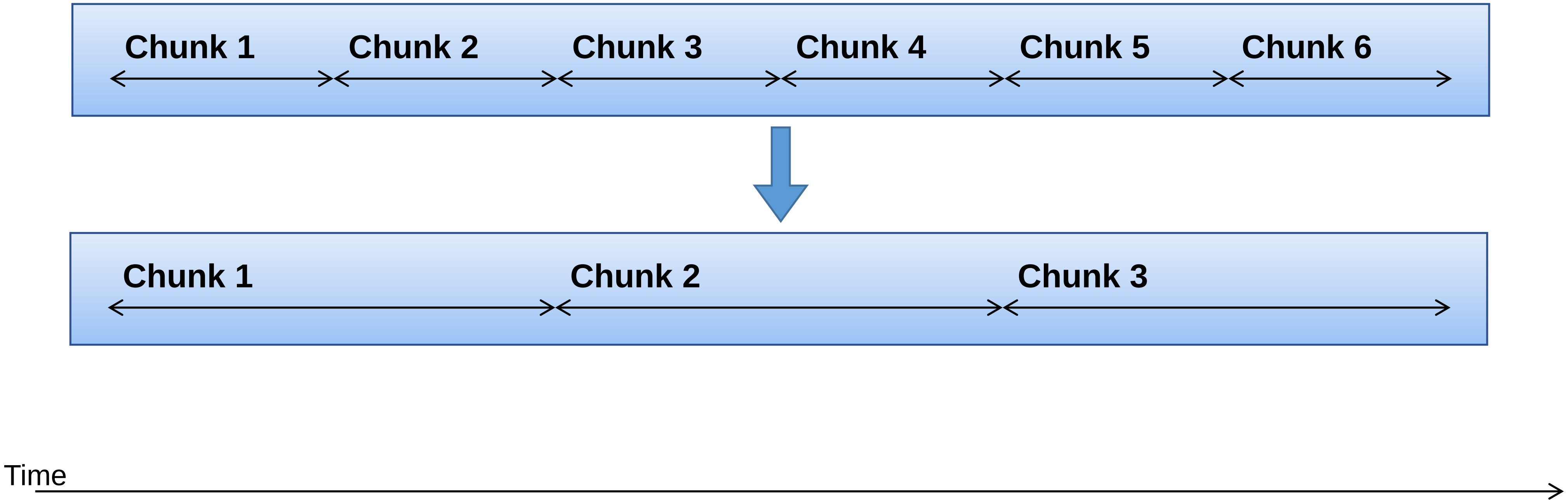
Manipulating the datastream structure

- The datastream is splitted in successive *chunks*
  - Each chunk as a *start* and *end* time reflecting the time period it represents
  - These *start* and *end* times are used by subsequent boxes for inter stream synchronization



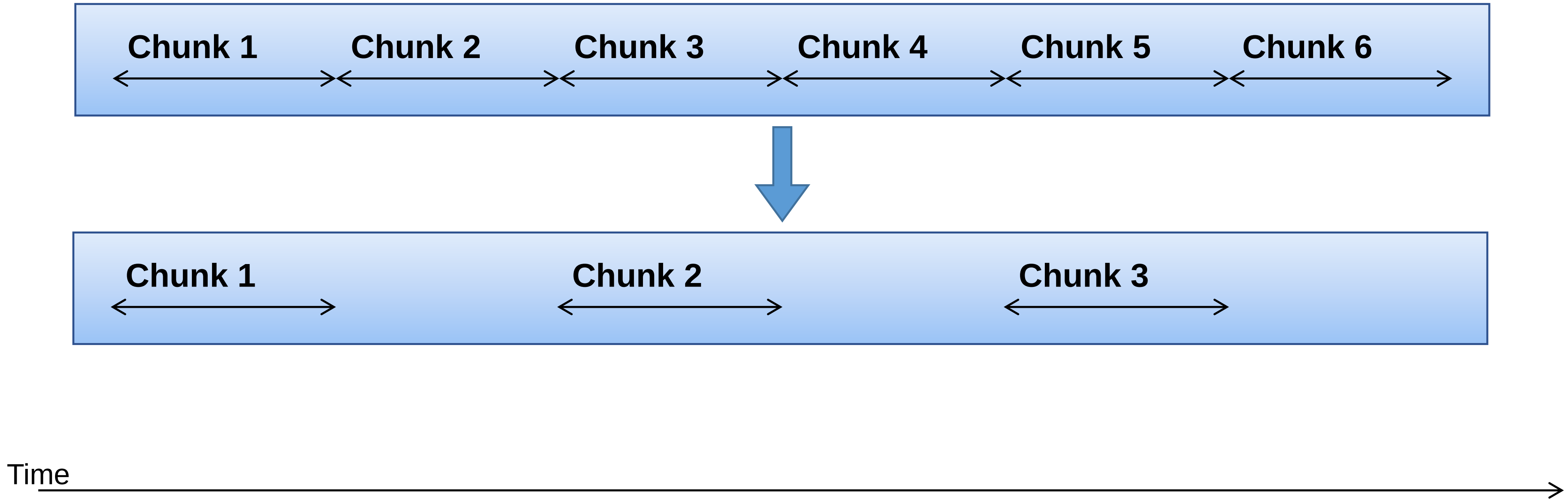
Manipulating the datastream structure

- The stream structure can be modified using epochers, e.g :
  - Chunk resize



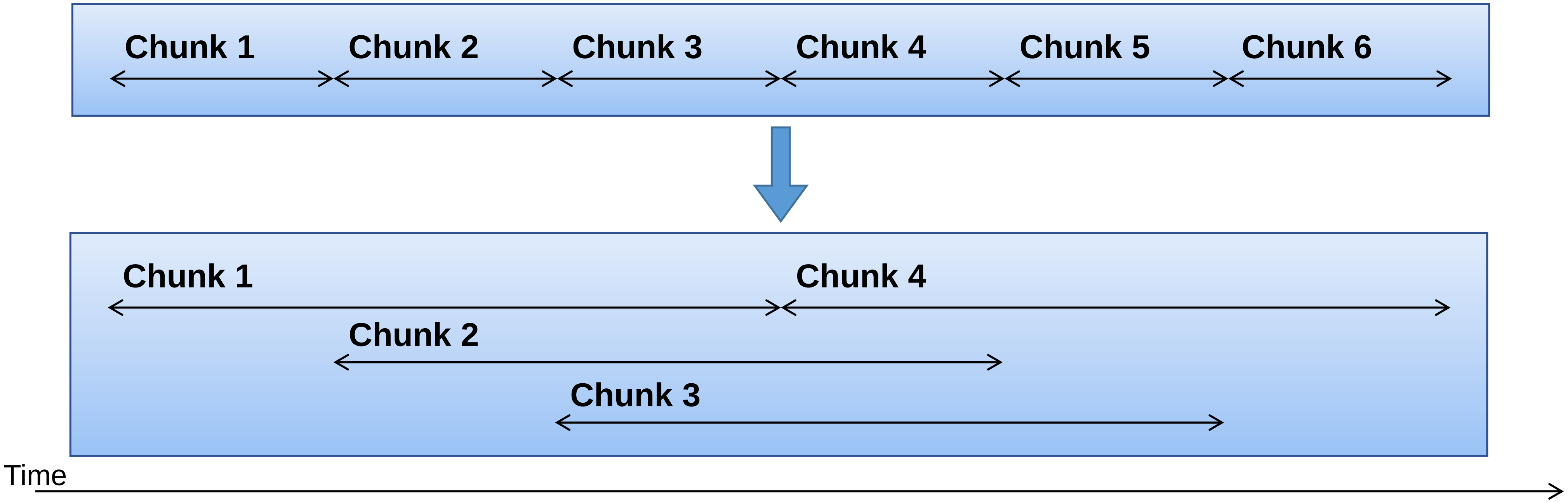
Manipulating the datastream structure

- The stream structure can be modified using epochers, e.g :
  - Chunk resize
  - Regular chunk selection



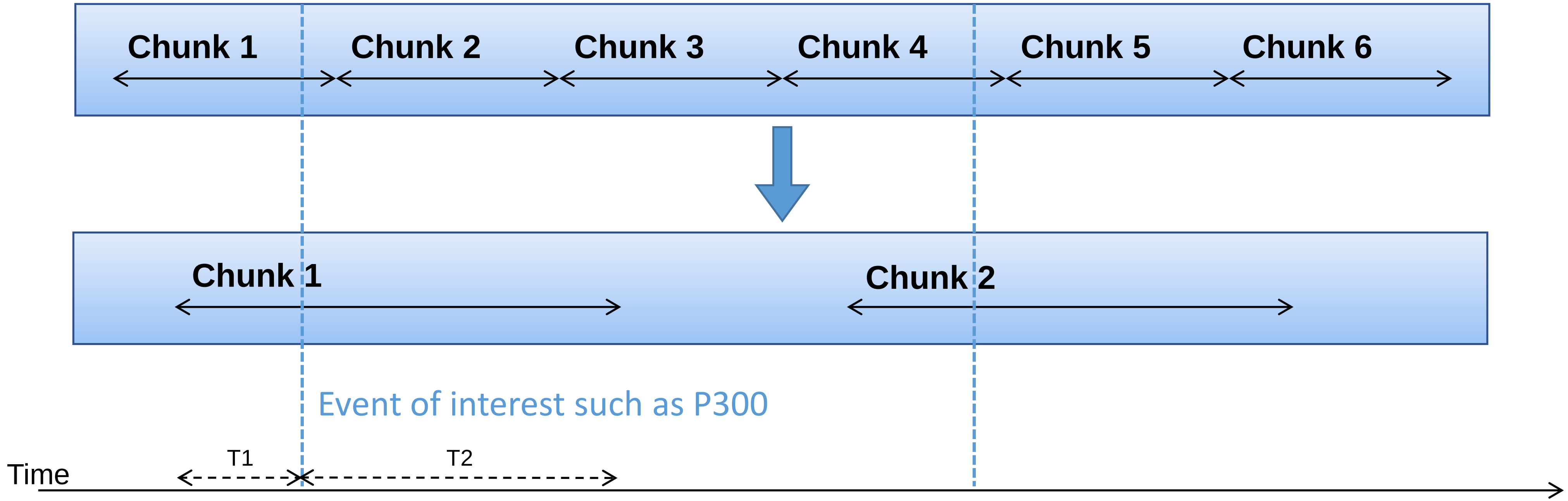
Manipulating the datastream structure

- The stream structure can be modified using epochers, e.g :
  - Chunk resize
  - Regular chunk selection
  - Overlapping chunks (e.g. for moving averages)



Manipulating the datastream structure

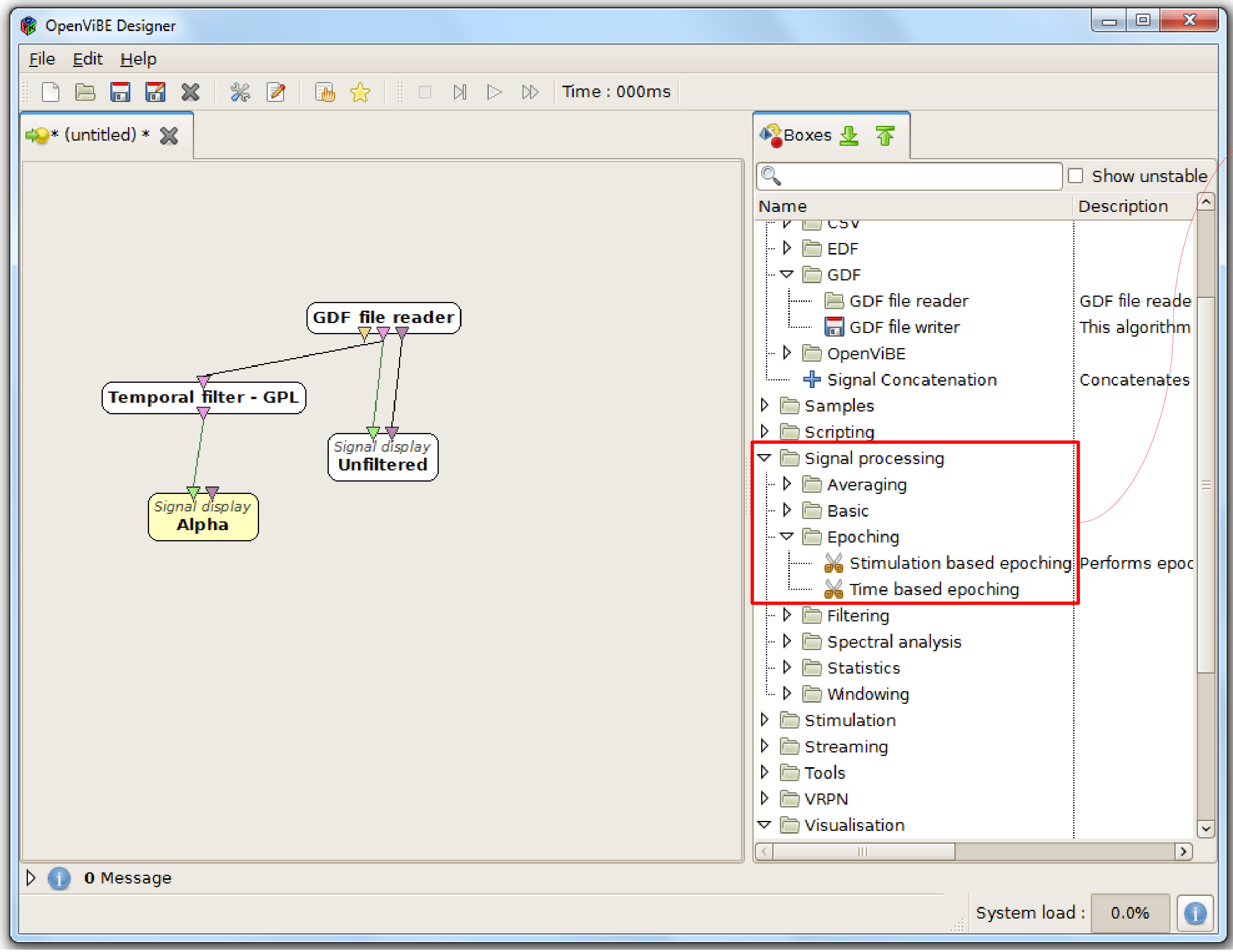
- The stream structure can be modified using epochers, e.g :
  - Chunk resize
  - Regular chunk selection
  - Overlapping chunks (e.g. for moving averages)
  - Signal selection around events of interest



Resizing epochs

Time-based epoching use case 1

Resizing epochs

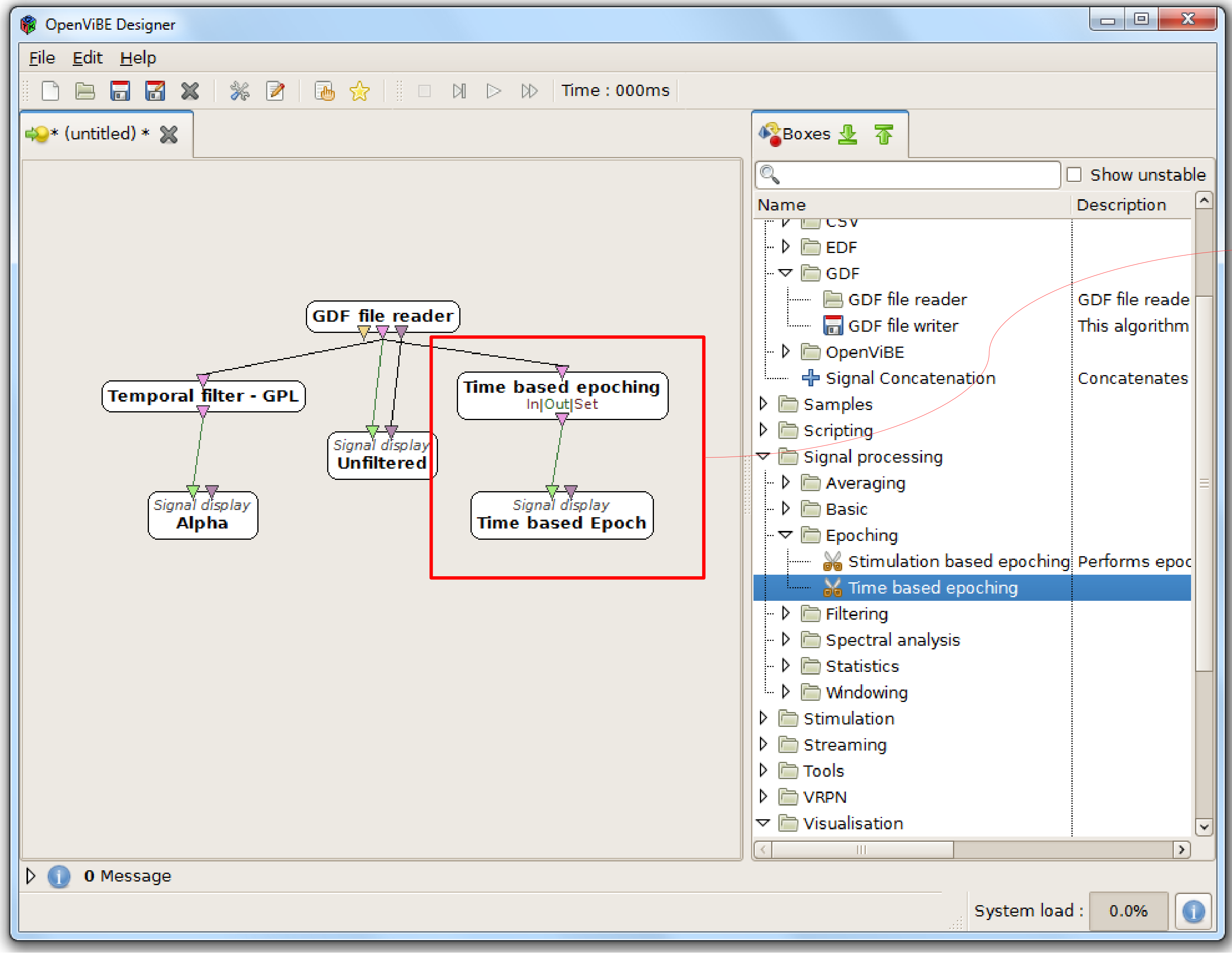


Category
signal processing, epoching

The *Time based epoching* box allows to resize the epochs of the datastream.

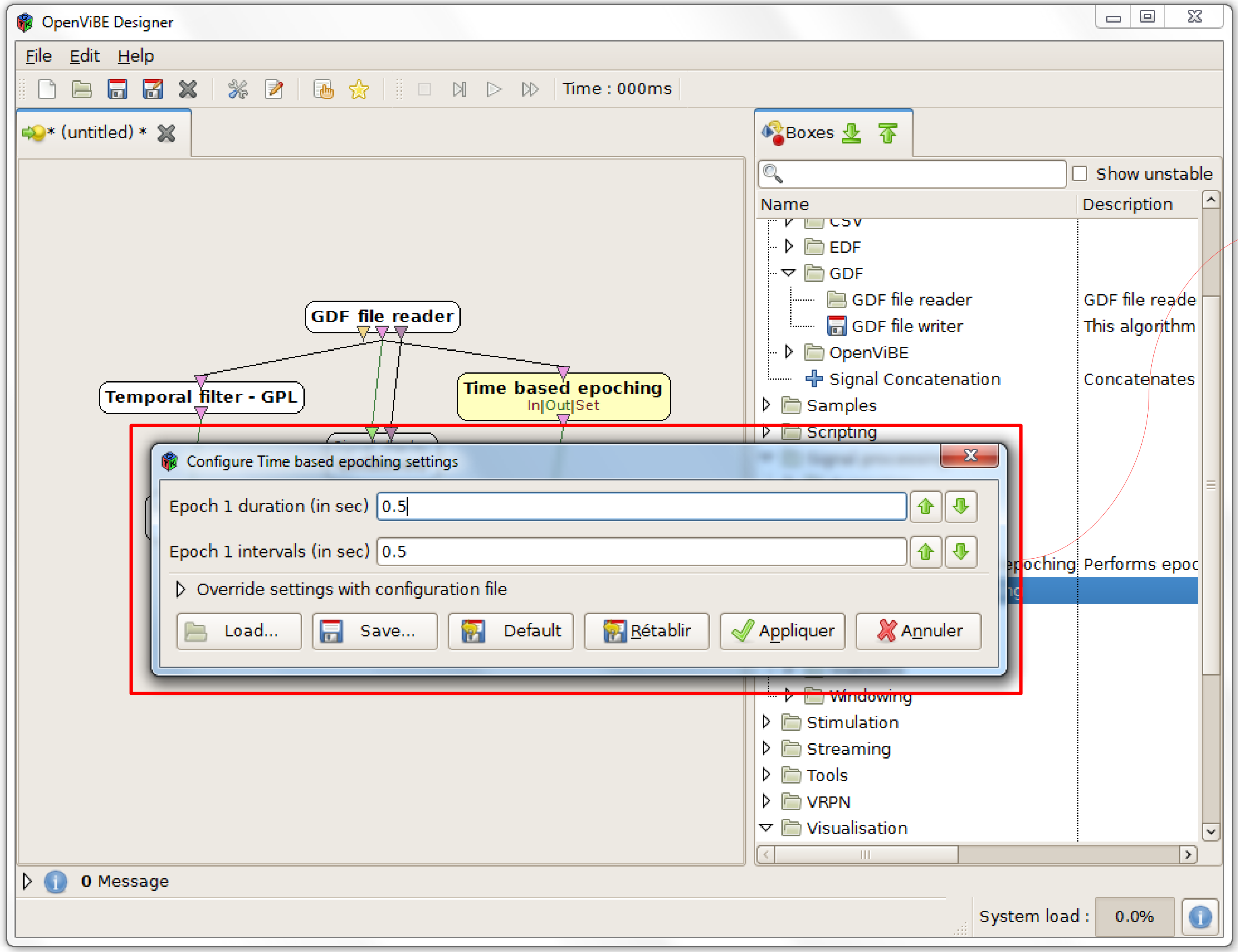
The *Stimulation based epoching* box allows to select signal around an event of interest (also called *Stimulation* in OpenViBE).

Resizing epochs

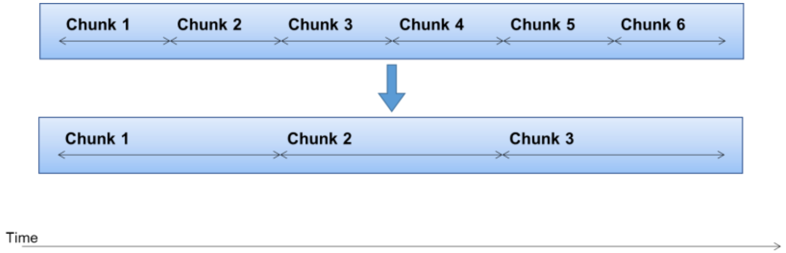


Drag & drop a *Time based epoching* box and a new *Signal Display* box. Then rename the *Signal Display* box in a convenient way.

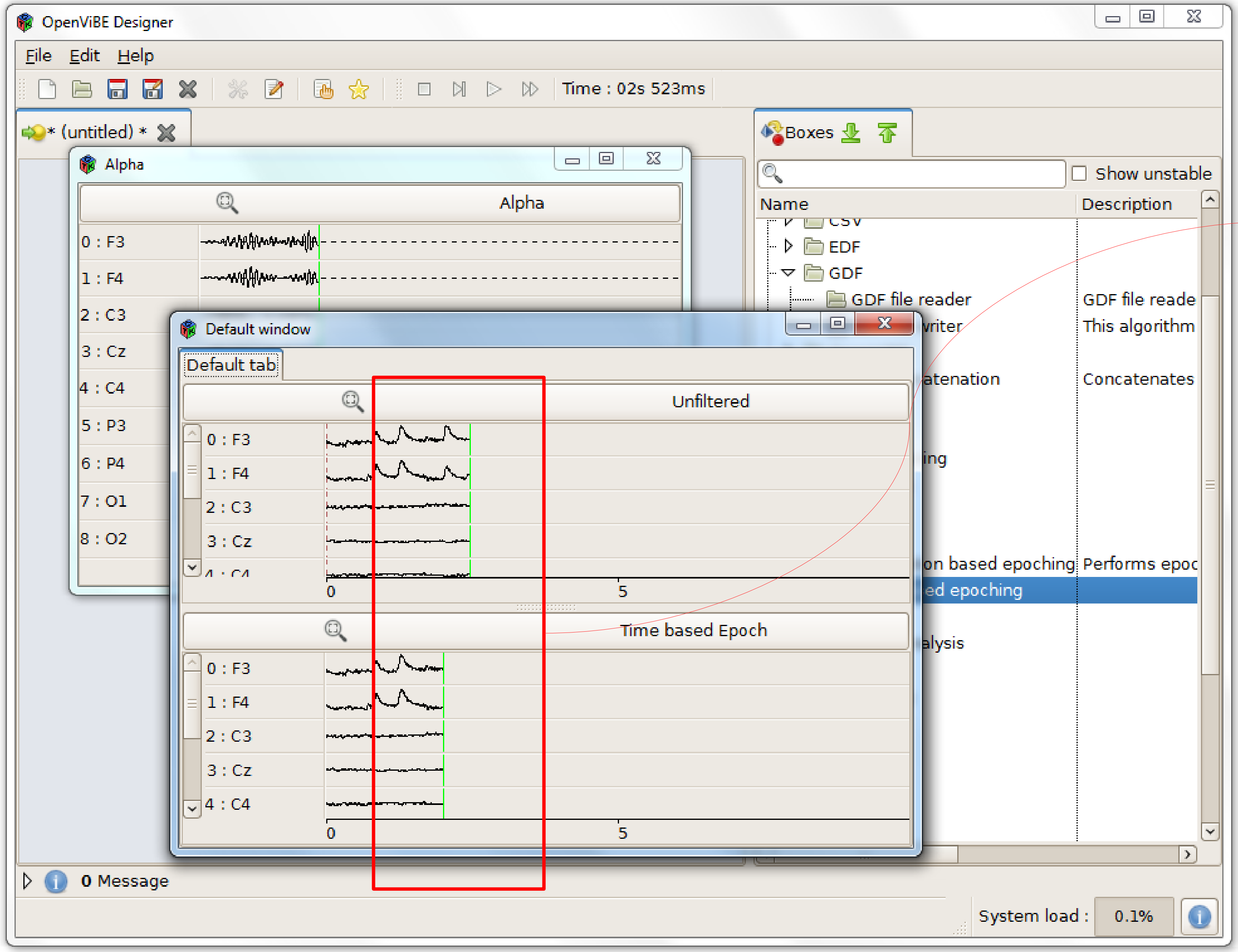
Resizing epochs



Configure the *Time based epoching* box to build epochs of half a second every half a second.



Resizing epochs



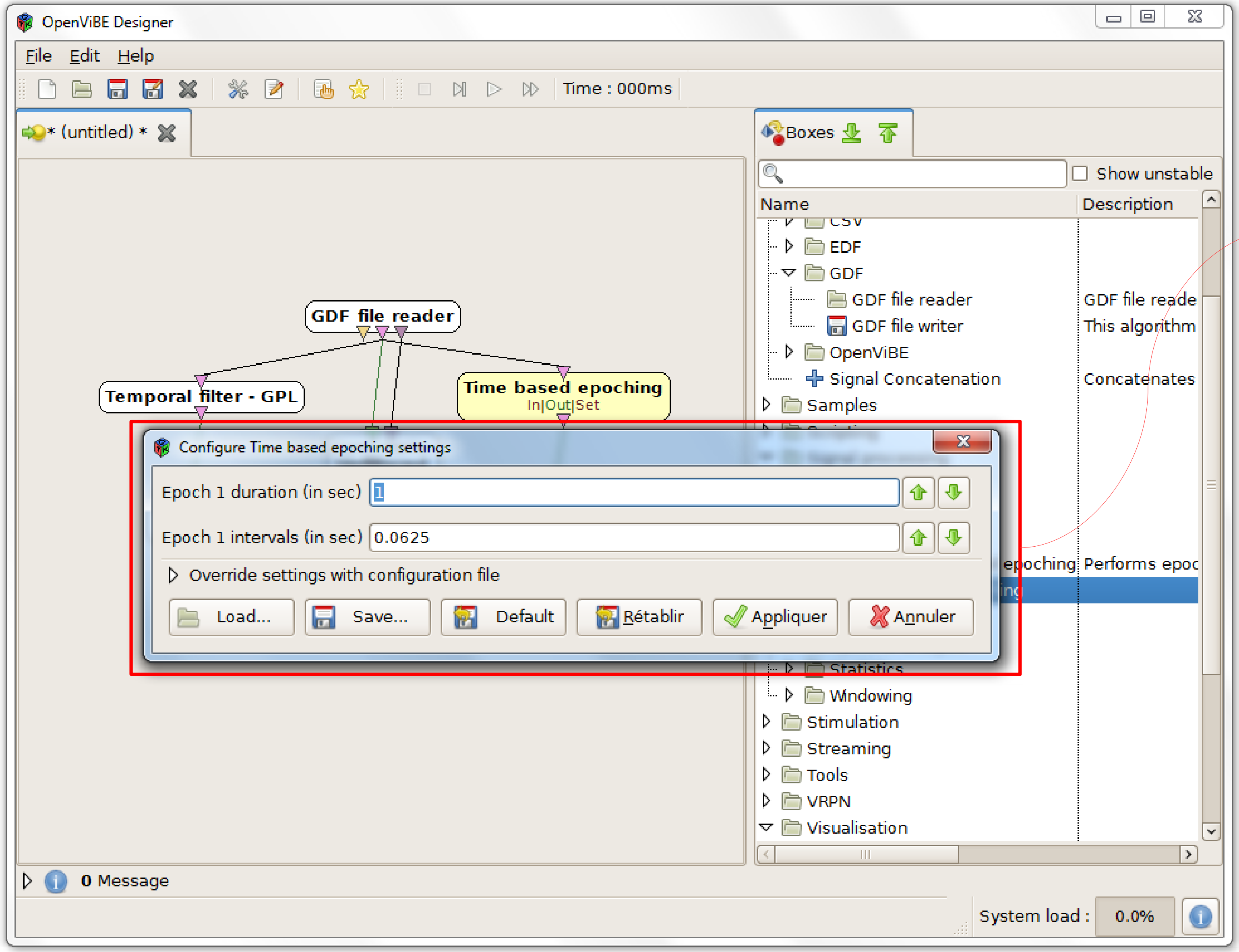
The refresh rate is now lower in the new widget than in the first because the epochs cover a bigger amount of time. The content however is visually identical.

With a sampling rate of 512 Hz, half a second represents 256 samples, thus exactly 8 chunks of 32 samples each, as we configured in the file reading box.

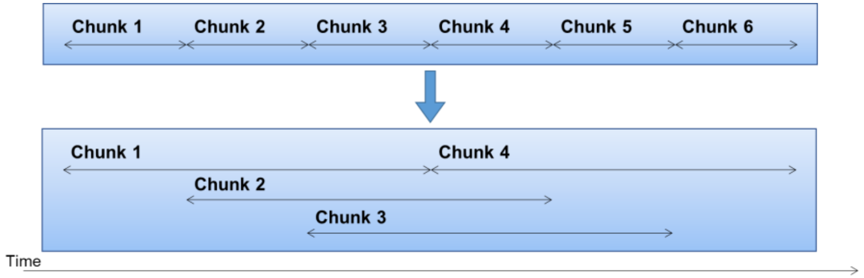
Overlapping epochs

Time-based epoching use case 2

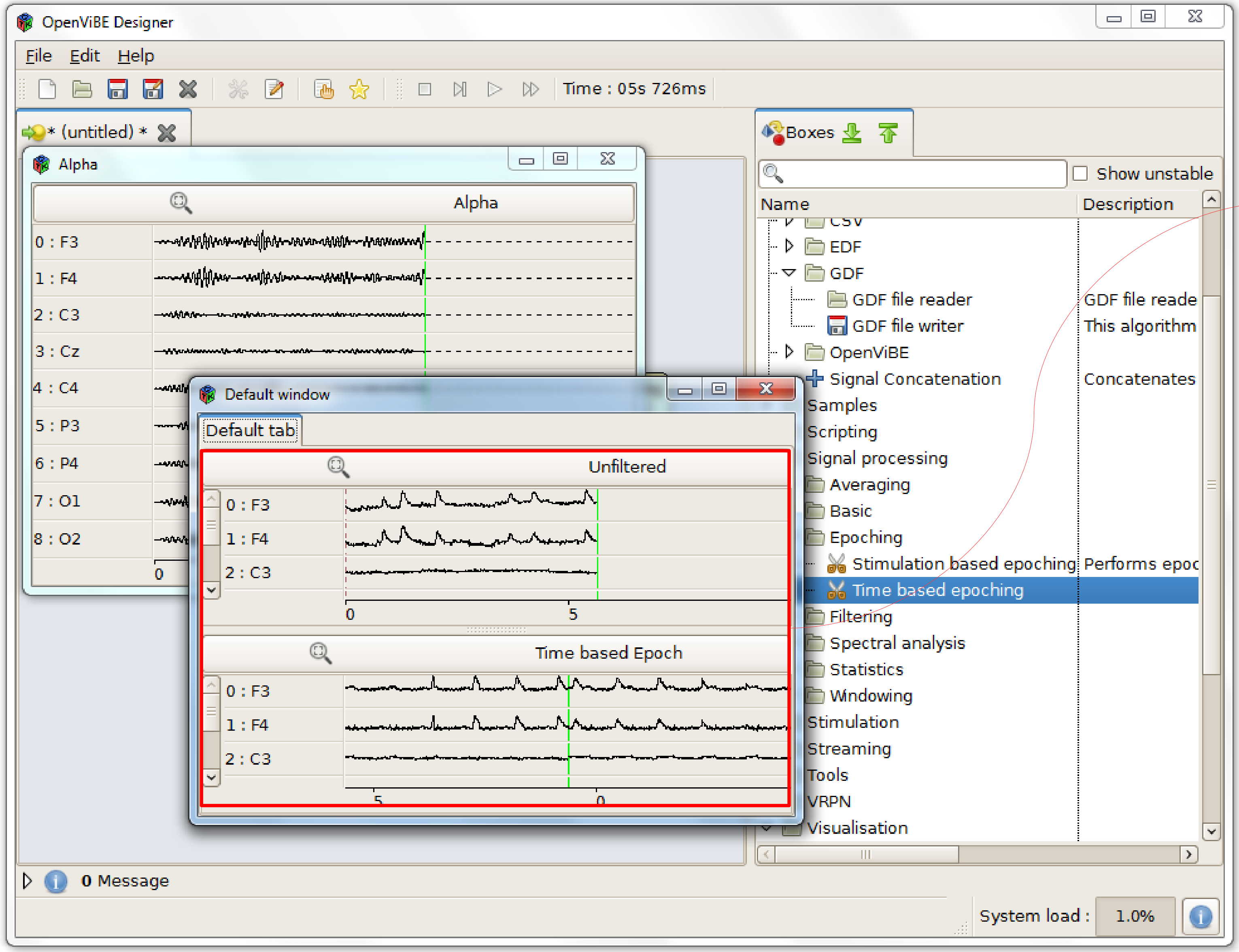
Overlapping epochs



Edit the *Time Based Epoching* box settings to generate epochs of 1 second every 16th of second (0.0625sec)

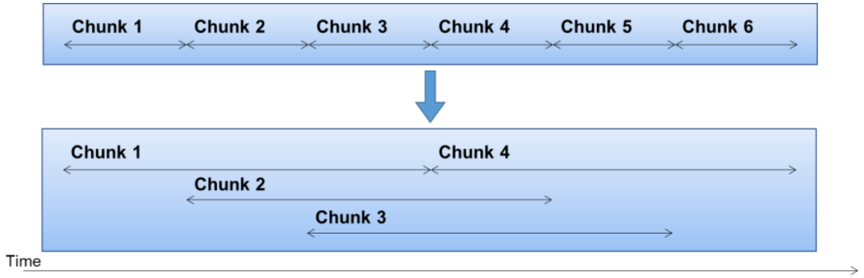


Overlapping epochs



The epoched signal appears to to have16 times more samples with duplicated samples. Visualisation of this is not very useful…

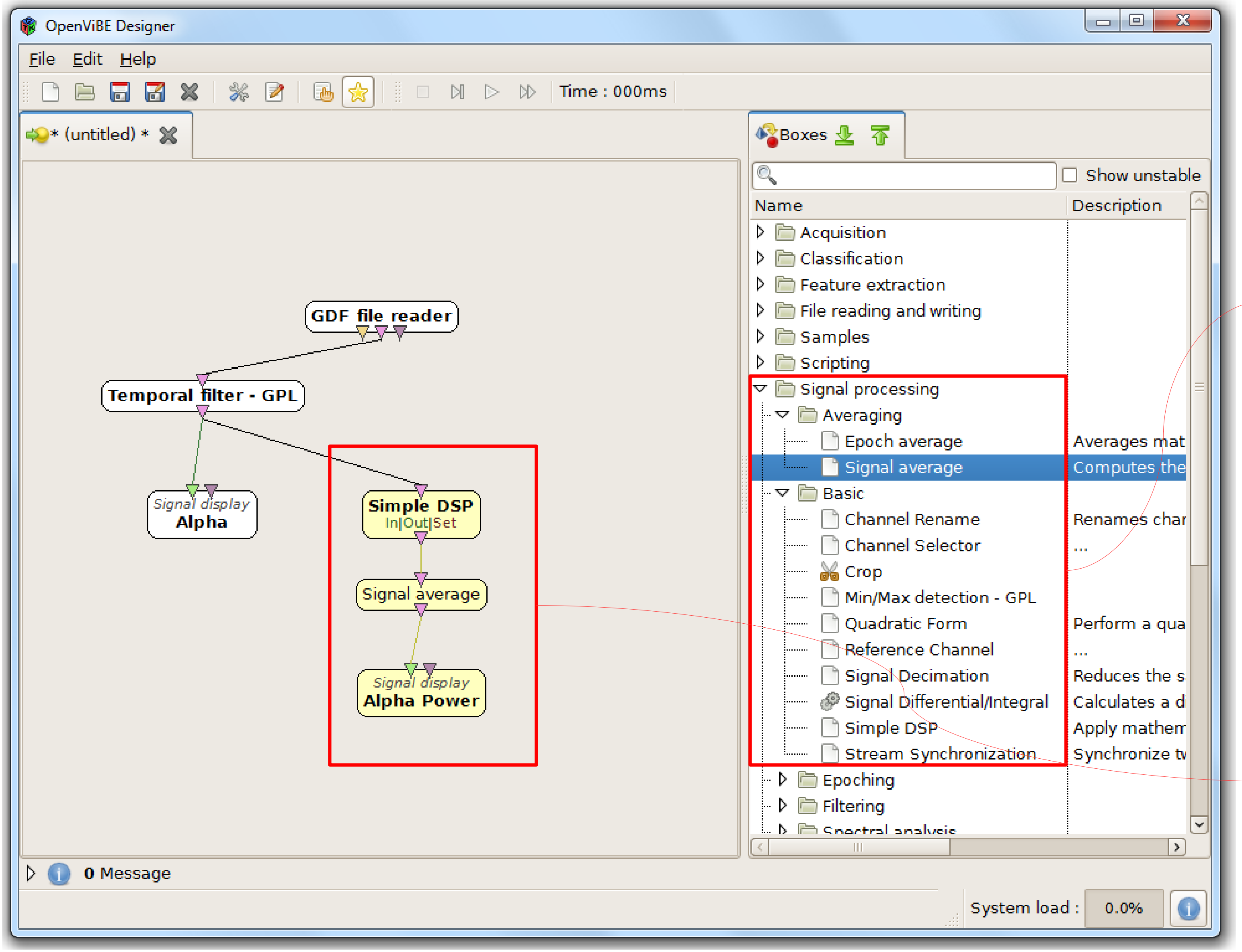
… however, we can now process these epochs to grab interesting information !



Computing band powers

1. Band-pass filter signal
2. Calculate square of signal

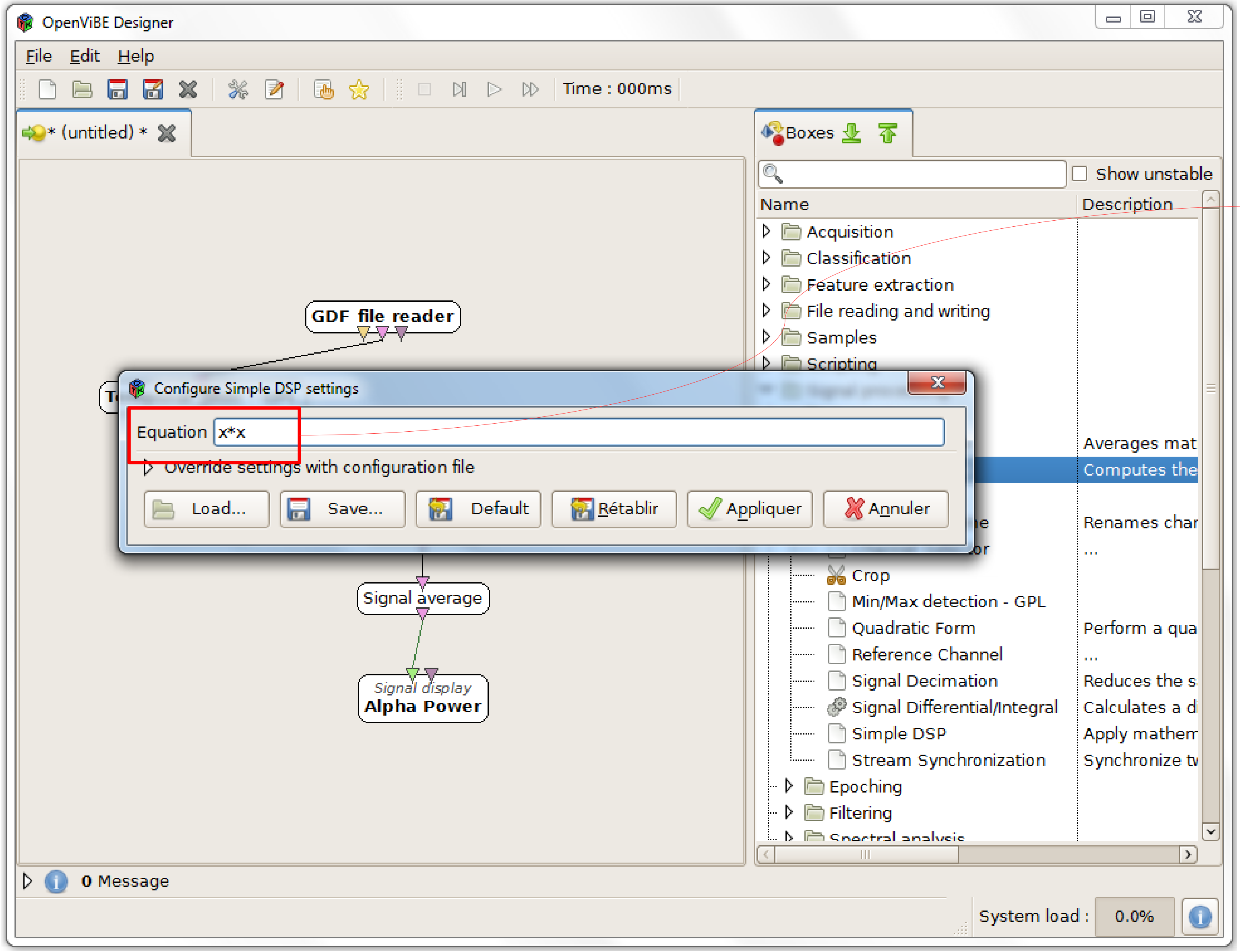
Computing band powers



Remove the *unfiltered* pipeline then drag & drop a *Simple DSP* box and a *Signal Average* box (found in Signal Processing / Basic and Signal Processing / Averaging respectively).

Connect them to the*Temporal Filter* box and add a new *Signal Display* box.

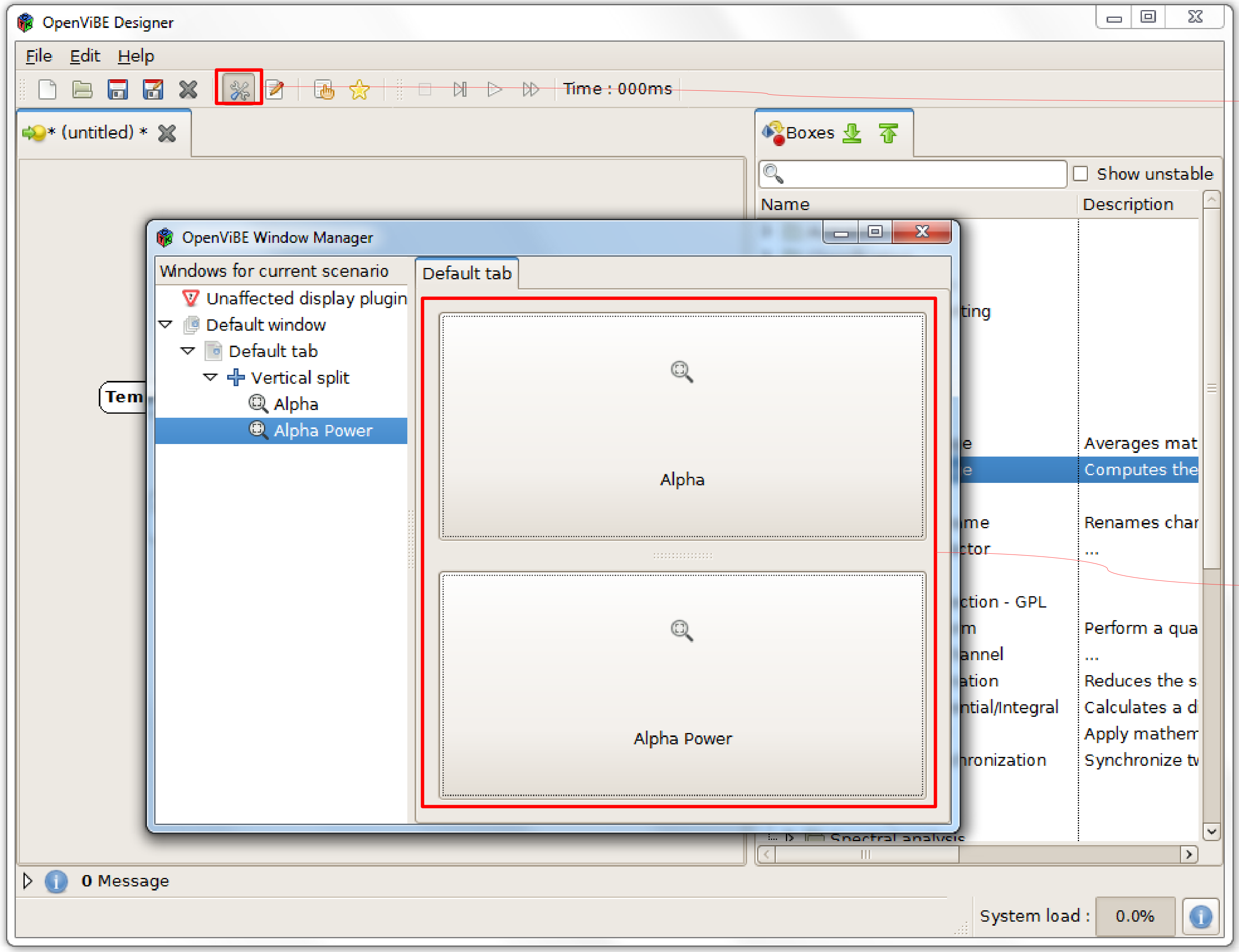
Computing band powers



Configure the *Simple DSP* box so that it computes the square of each sample.

The power of a signal in a given frequency band can be computed as the average of the square of the samples in the given frequency band. The *Simple DSP* box computes the square of the samples and the *Signal Average* box computes the average of the values for each chunk.

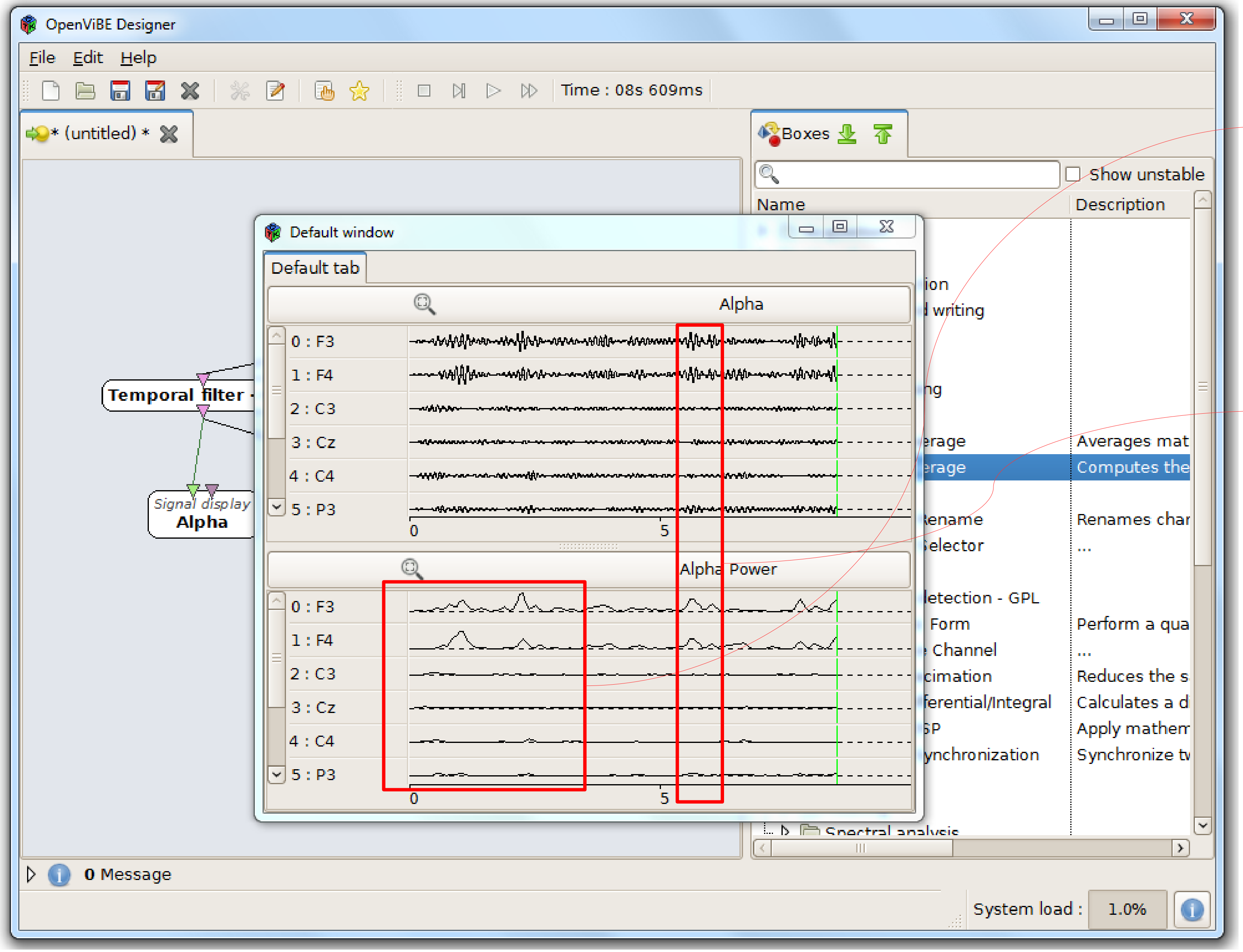
Computing band powers



Open the *Window Manager*

Arrange the widgets so that the *Alpha* widget is on top of the *Alpha Power* widget.
Then close the *Window Manager*.

Computing band powers



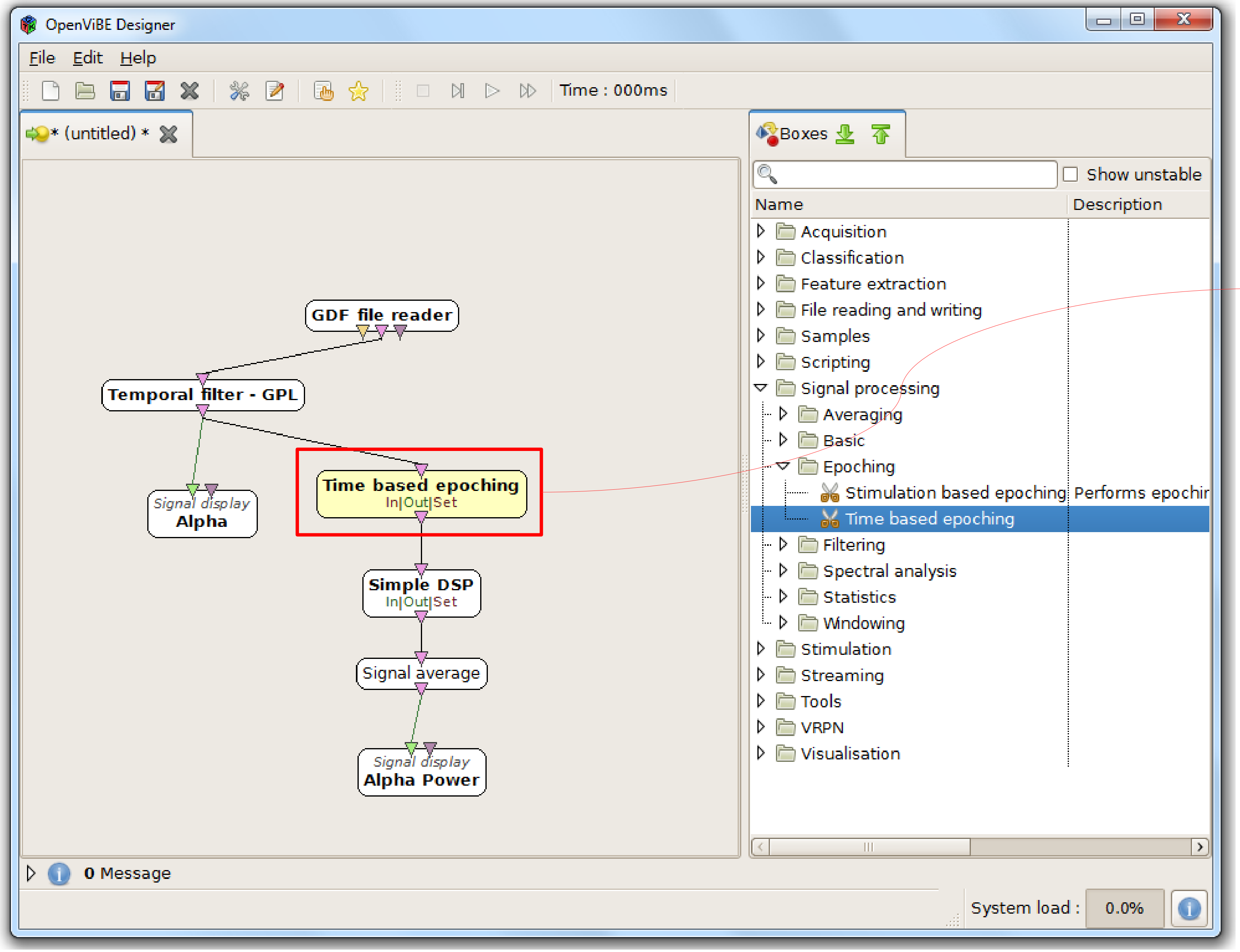
The bottom widget shows the alpha band power

A burst of activity in the alpha band translates into a peak in the alpha power that can be easily visualized.

Computing band powers on larger epochs

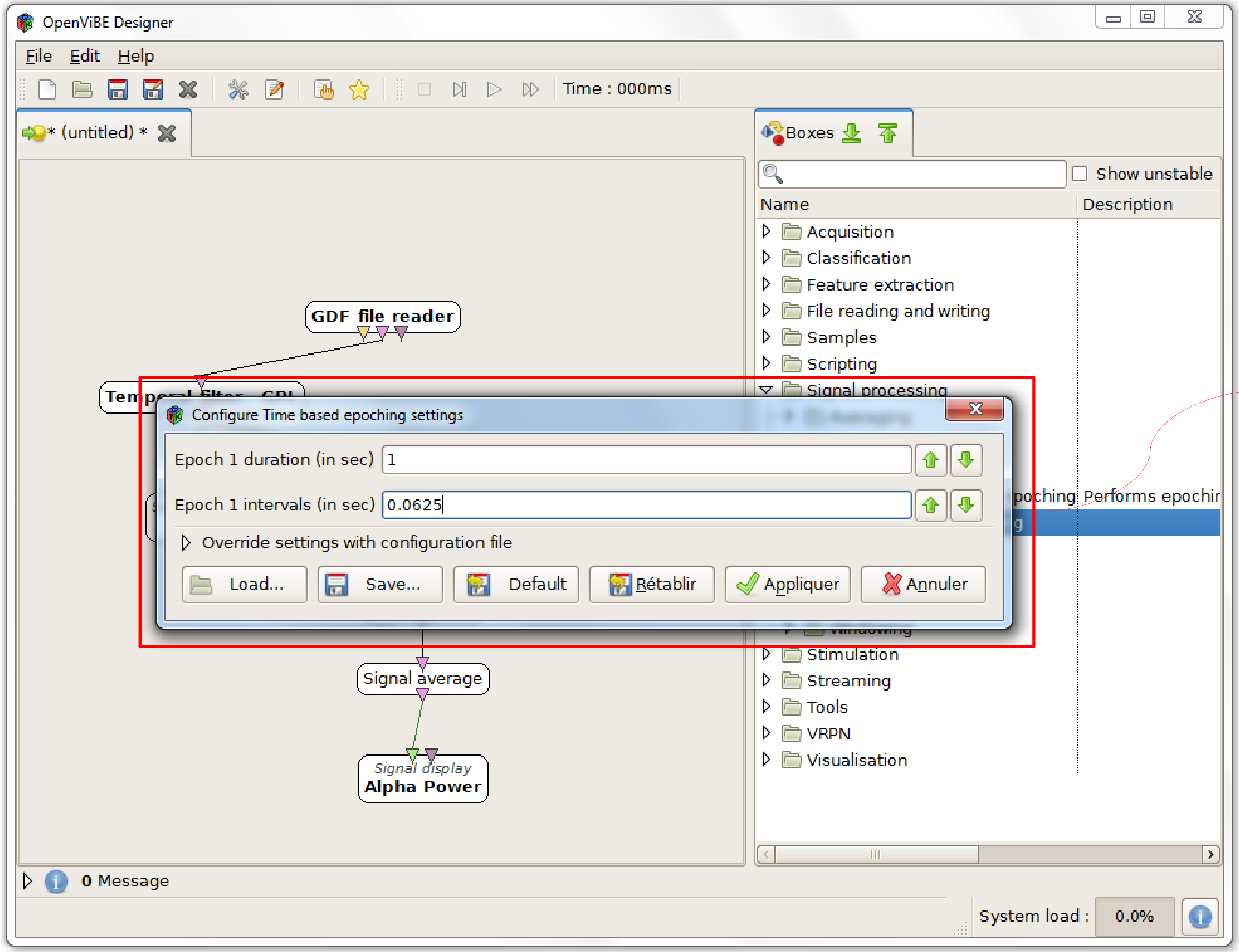
1. Band-pass filter signal
2. Overlapping time-based epoching
3. Calculate square of signal

Computing band powers on larger epochs



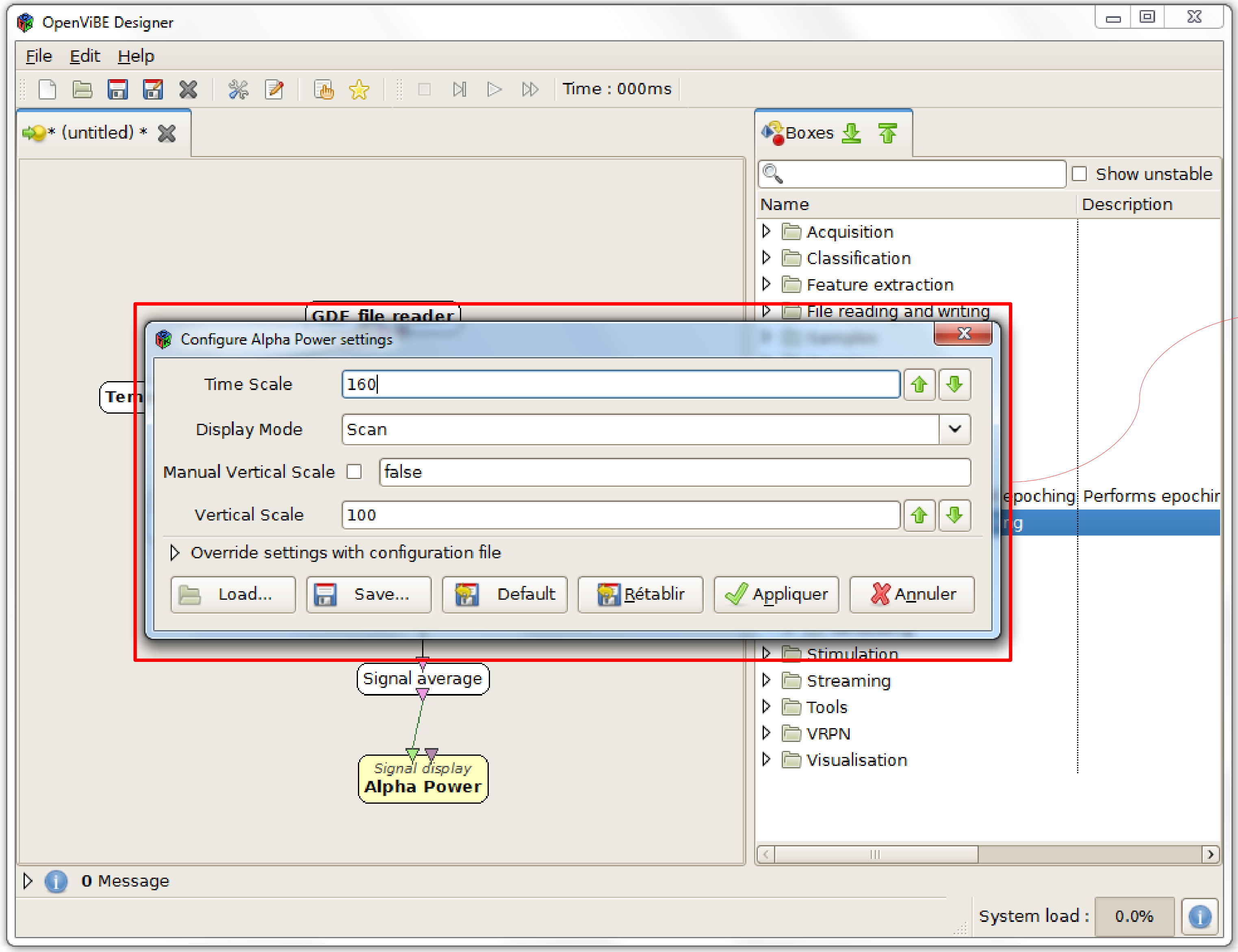
Add a *Time Based Epoching* box before the *Simple DSP* box.

Computing band powers on larger epochs



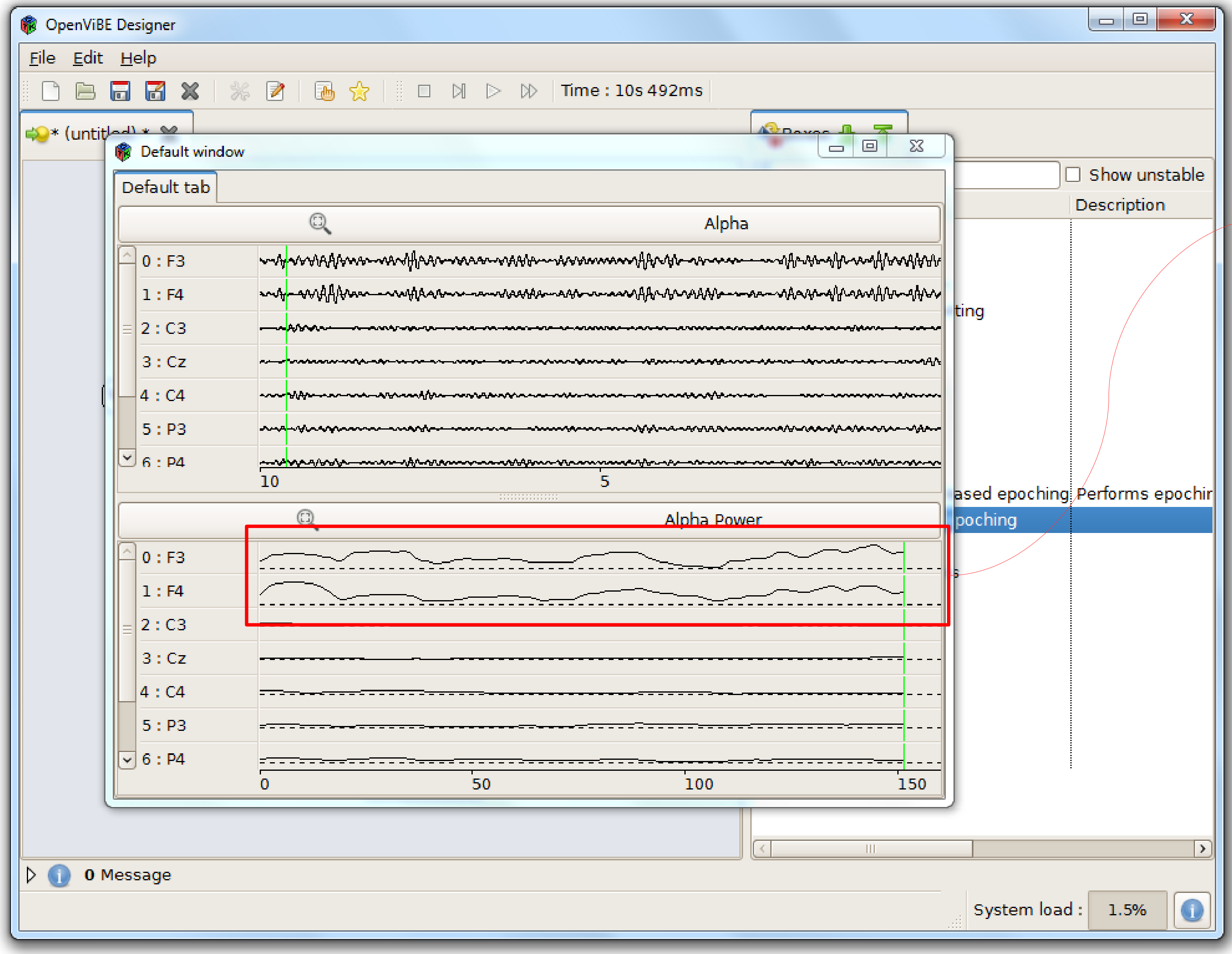
Edit the *Time Based Epoching* box settings to generate epochs of 1 second every 16th of second (that is 0.0625 sec)

Computing band powers on larger epochs



Configure the *Alpha* visualization box so that it shows 160 seconds of signal.

Computing band powers on larger epochs

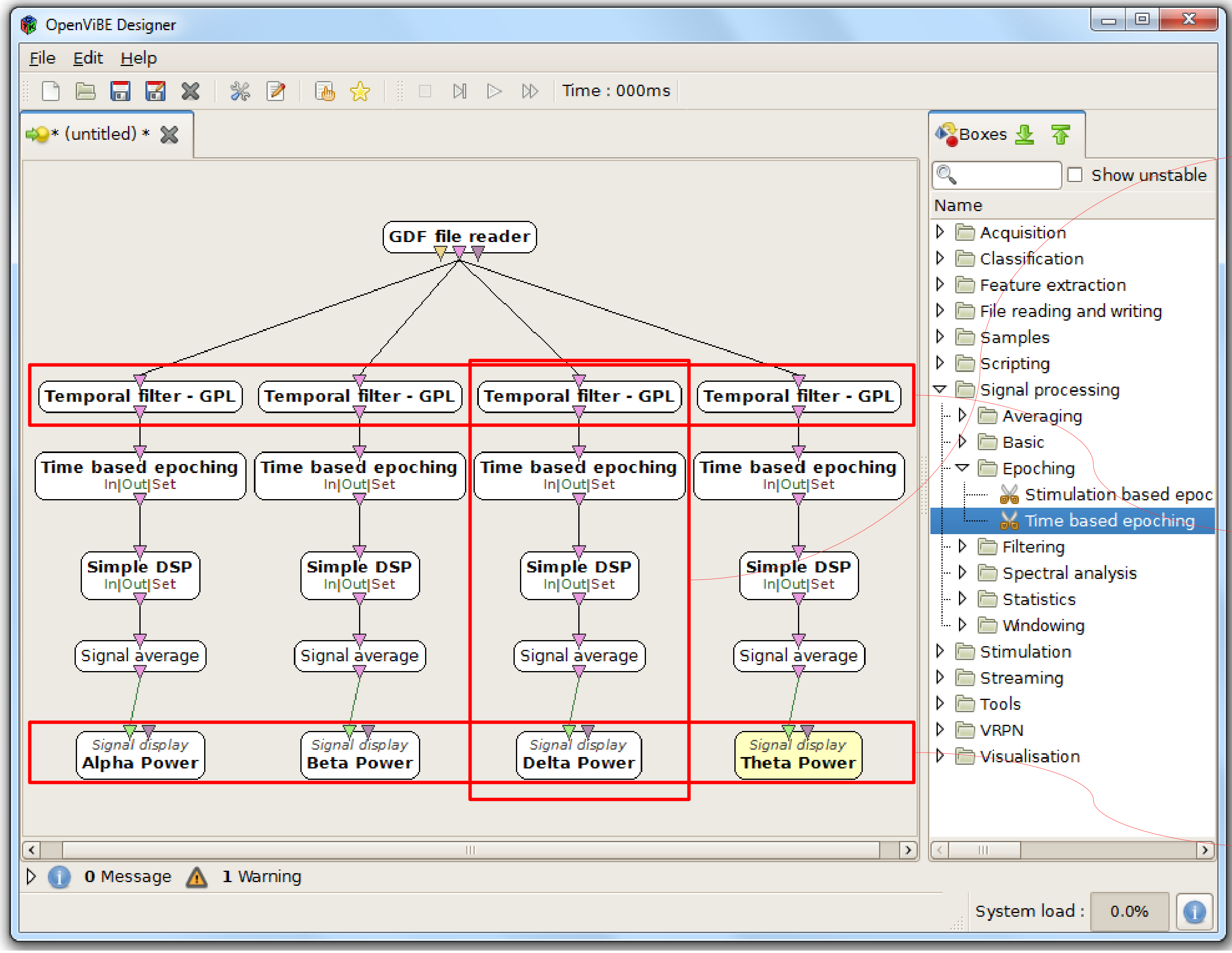


The power in the *alpha* band is extracted from larger epochs, thus is smoother than the previous one.

As soon as non continuous epochs are concerned, the *Signal Display* is usually unable to display time scales correctly.

Alpha, beta, delta and theta band powers

Alpha, beta, delta and theta band powers



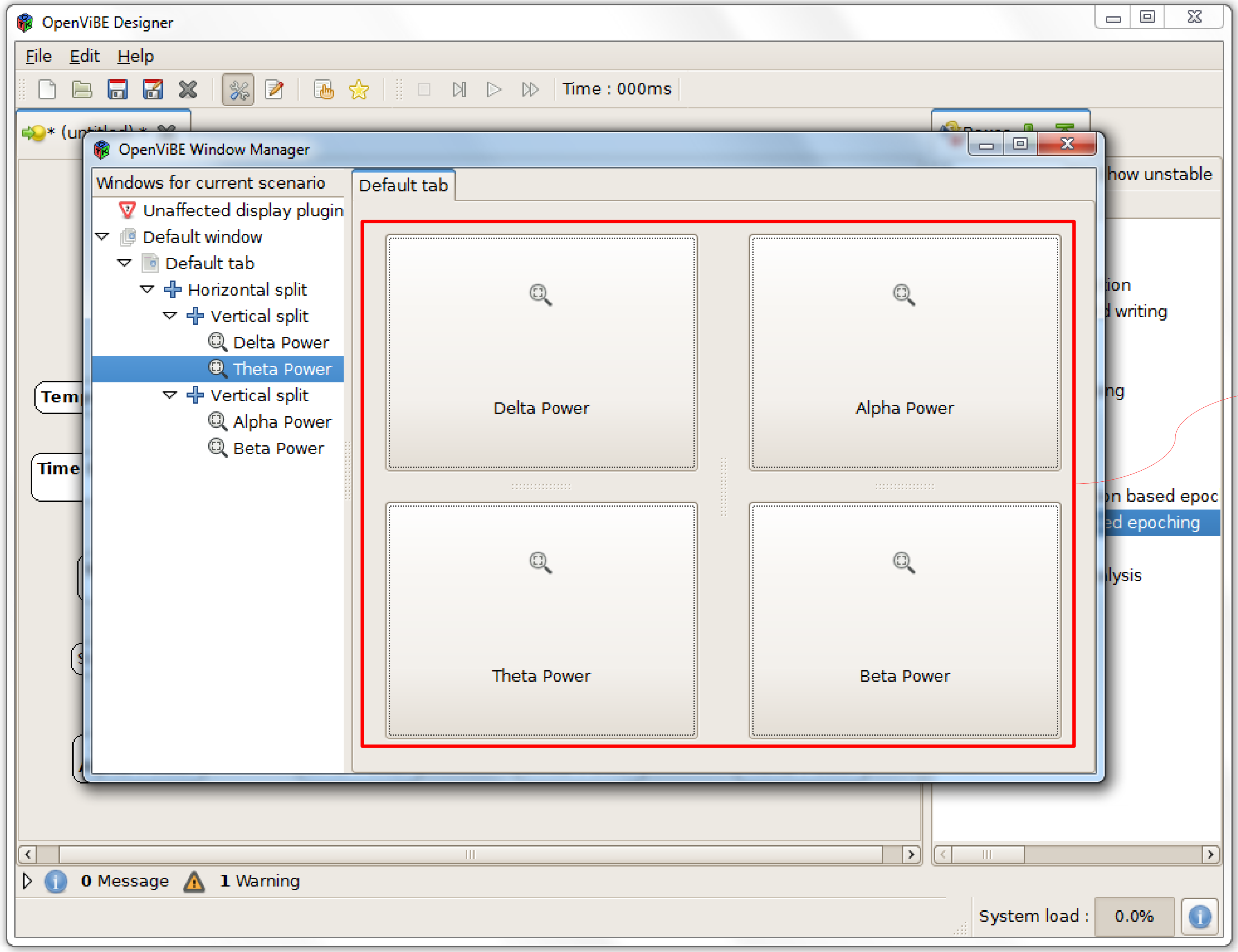
Remove the *Alpha* visualization box. Then copy & paste the entire signal processing pipeline 3 times.

Configure the *Temporal Filter* boxes as follows :

- Alpha : [8,12] Hz
- Beta : [12,24] Hz
- Delta : [1,4] Hz
- Theta : [4,8] Hz

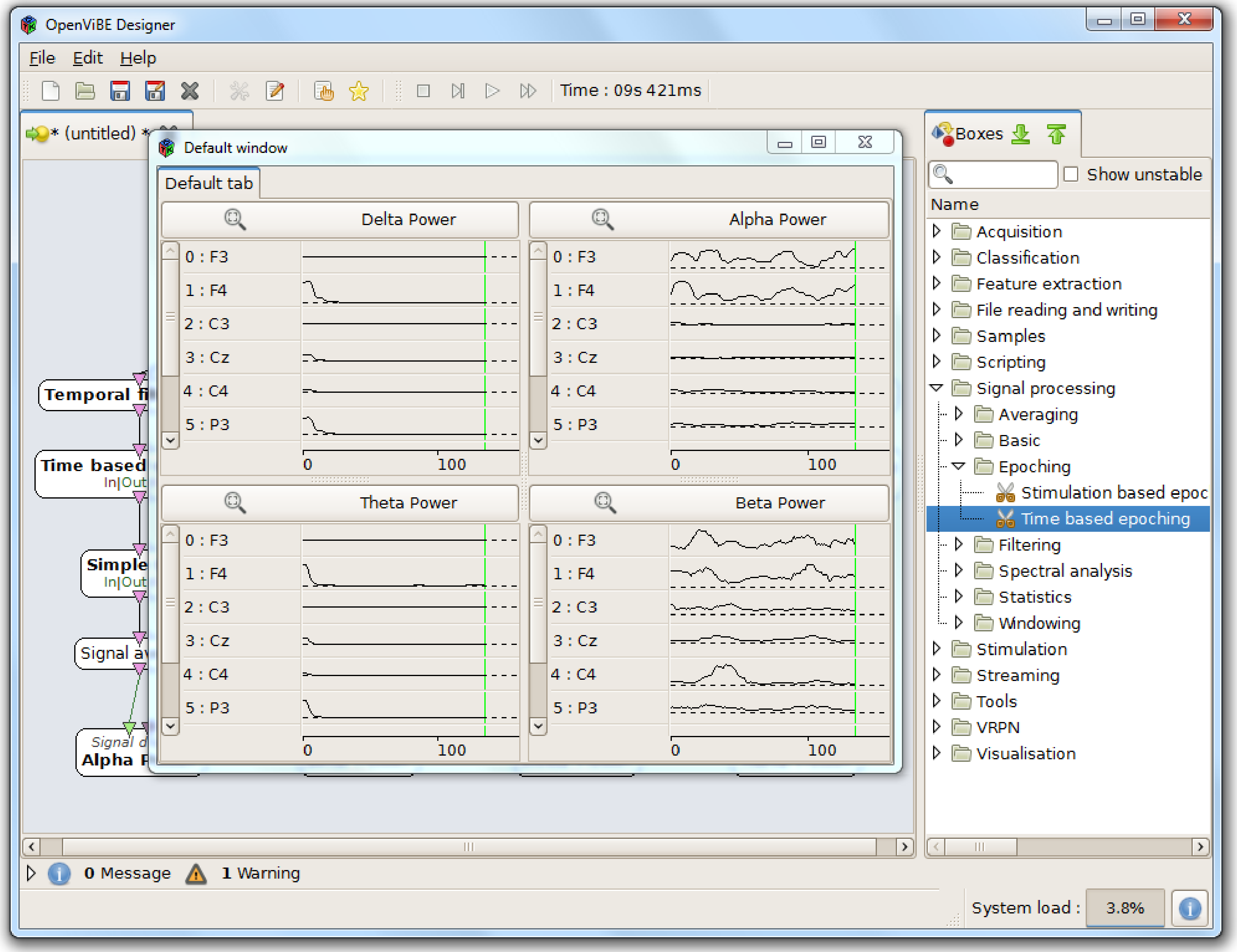
Rename the *Signal Display* boxes accordingly.

Alpha, beta, delta and theta band powers



Arrange the widgets in a convenient way using the *Window Manager.*

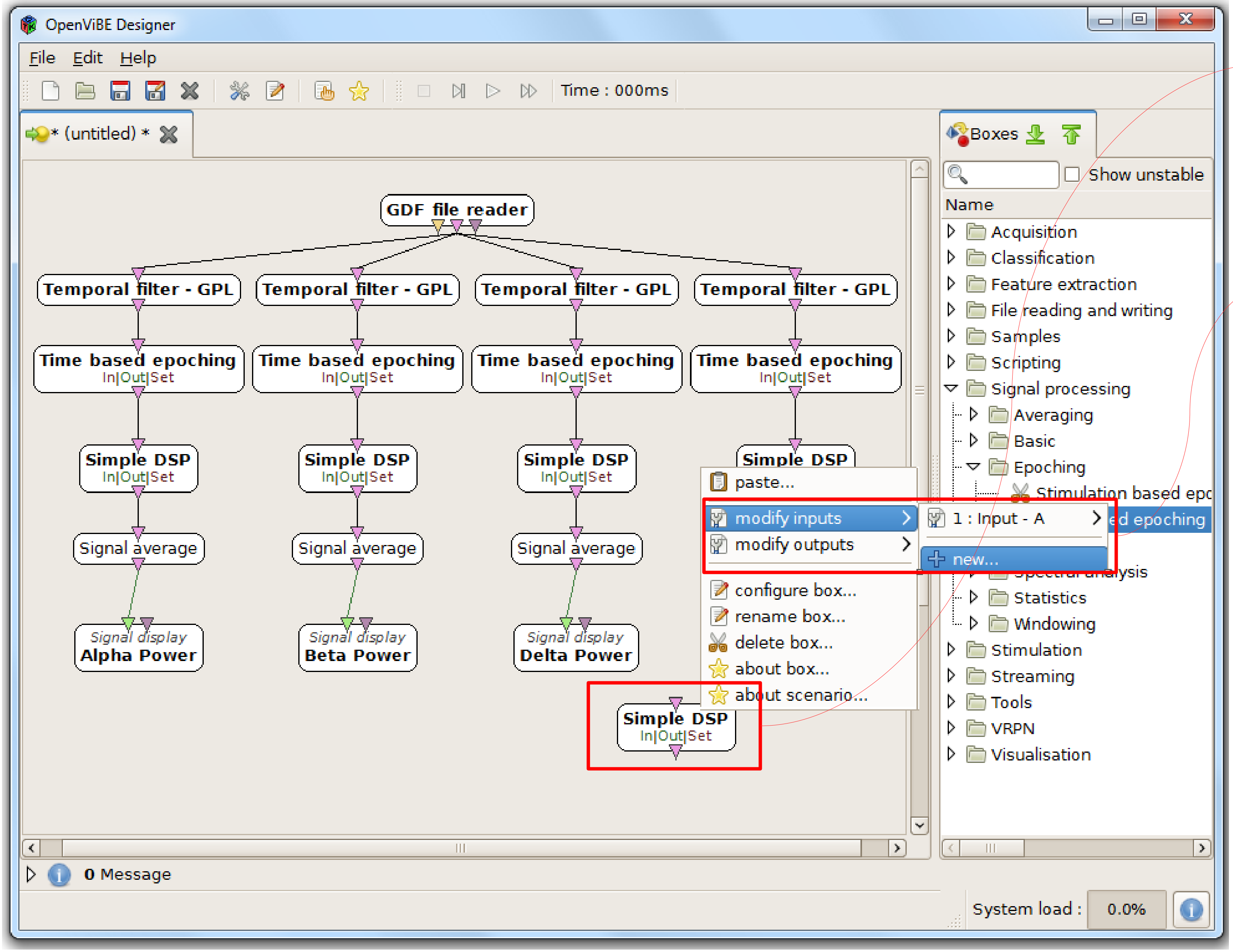
Alpha, beta, delta and theta band powers



The 4 powers are computed and displayed in real time

These frequency bands are related to specific brain activity. Alpha is involved in attention and relaxation for instance. Beta is involved in sensori motor processes (as for instance real or imagined hand or foot movements)

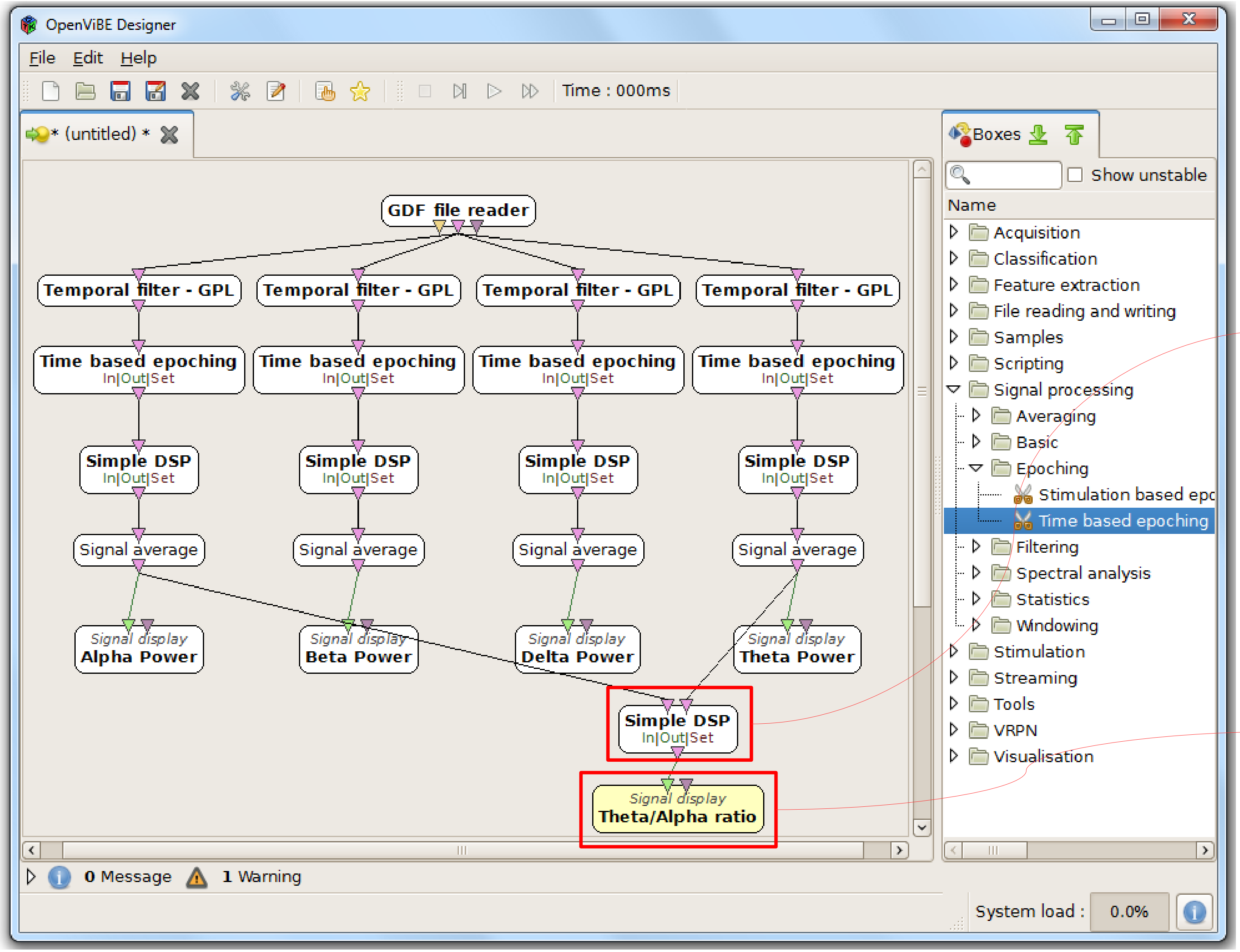
Alpha, beta, delta and theta band powers



Add a new *Simple DSP* box.

Right click on the box and add a new Input of type *Signal*.

Alpha, beta, delta and theta band powers



Connect the *Simple DSP* as follows :

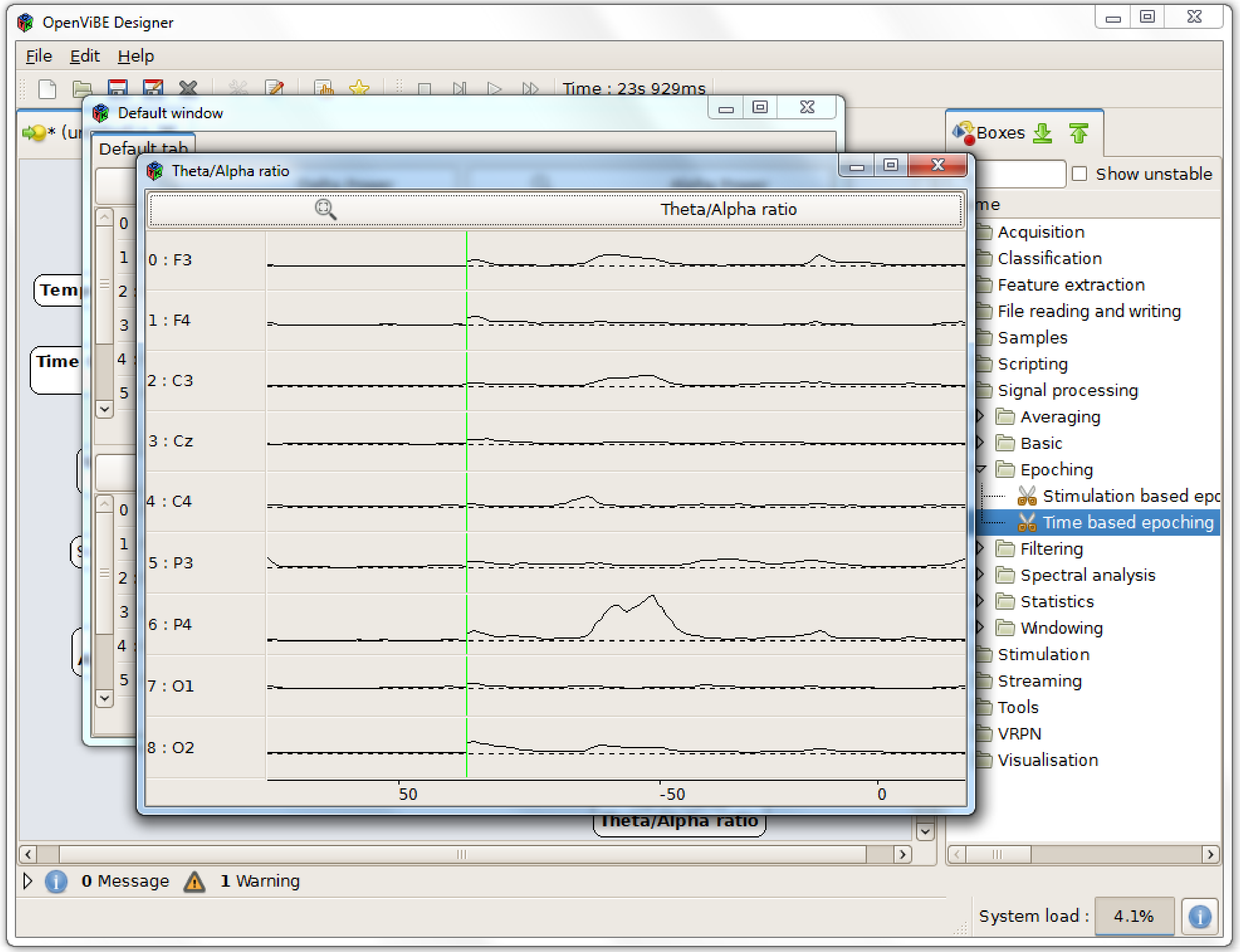
- Input A to alpha
- Input B to theta

Configure the box settings with the following equation A/B.

This will compute Alpha / Theta power ratio

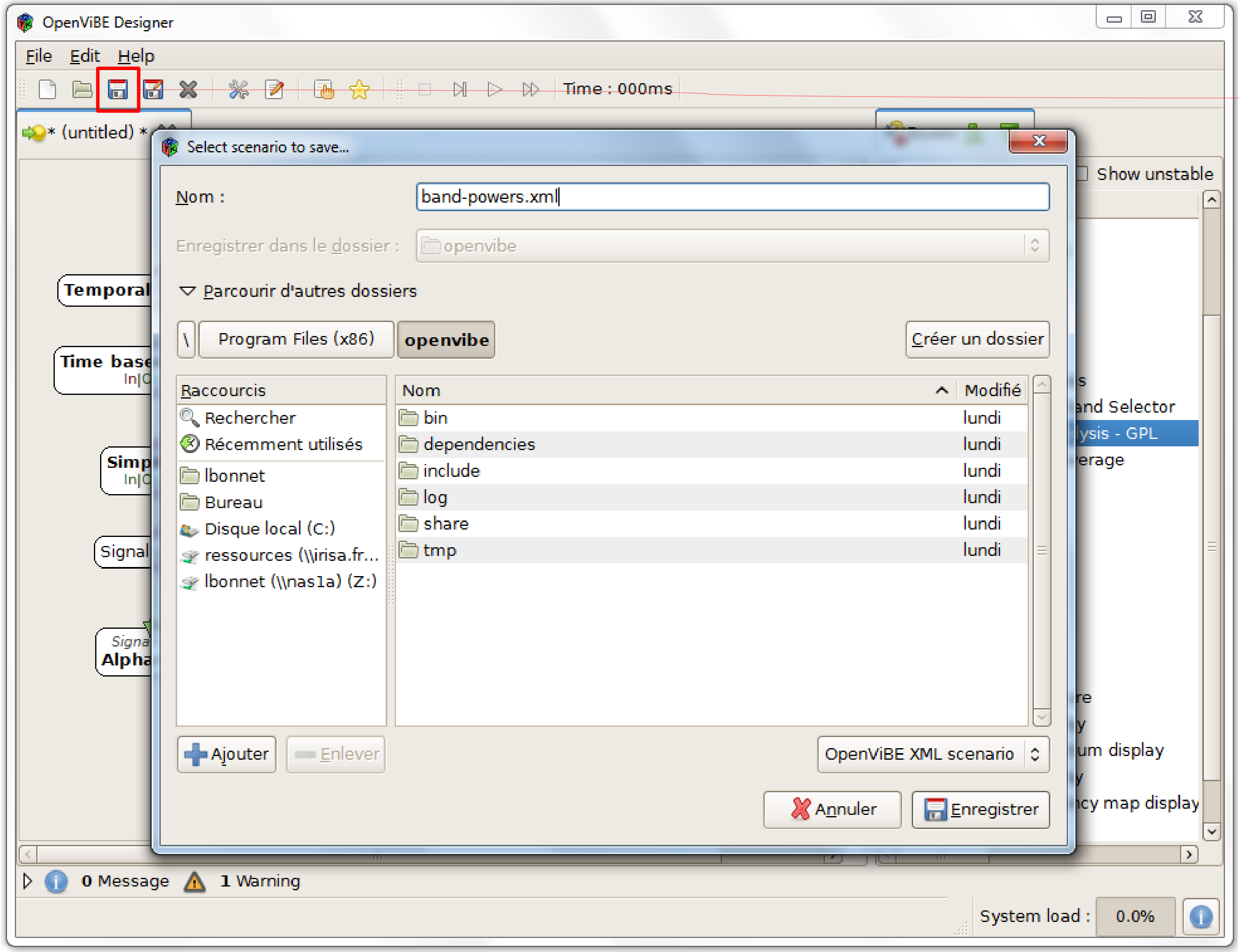
Then add a new *Signal Display* box

Alpha, beta, delta and theta band powers



Different band powers can be used at the same time. For example, most of the neurofeedback protocols are based on band powers and ratios.

Alpha, beta, delta and theta band powers



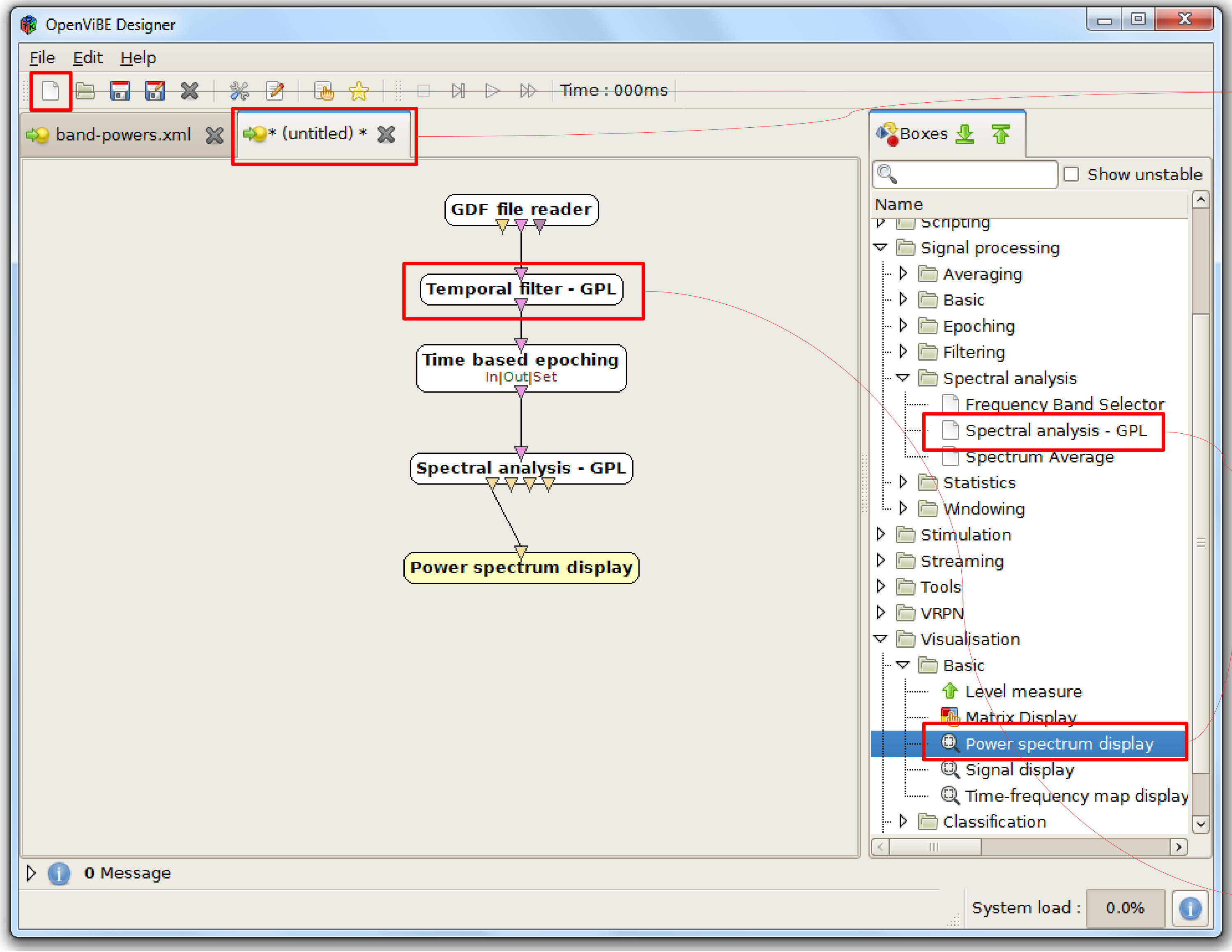
Now save the scenario.

OpenViBE saves the scenarios in a dedicated XML file format describing the boxes involved, their graphical location, their settings and connections etc...

Computing a complete spectrum of the signal

1. Filter to band of interest (optional)
2. Overlapping time-based epoching
3. Spectral analysis box
4. Average

Computing a complete spectrum of the signal

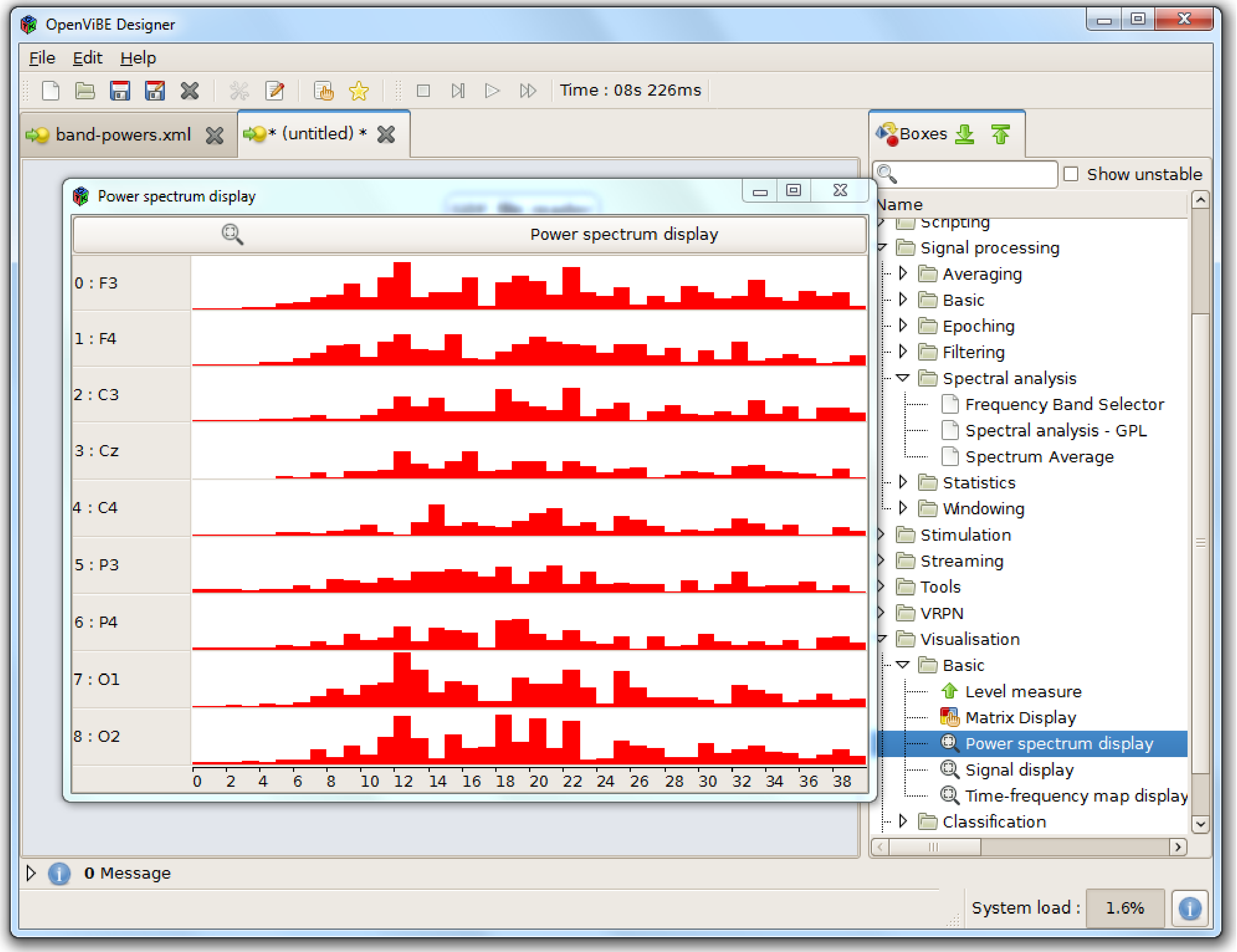


Create a new scenario (this creates a new tab)

Copy and paste the *GDF File Reader*, a *Temporal Filter* and a *Time Based Epoching* from the previous scenario. Then add a *Spectral Analysis* box and a *Power Spectrum Display* box and connect them to the rest of the pipeline

Configure the filter :
- Type : High-pass
- Low-cut : 1 Hz

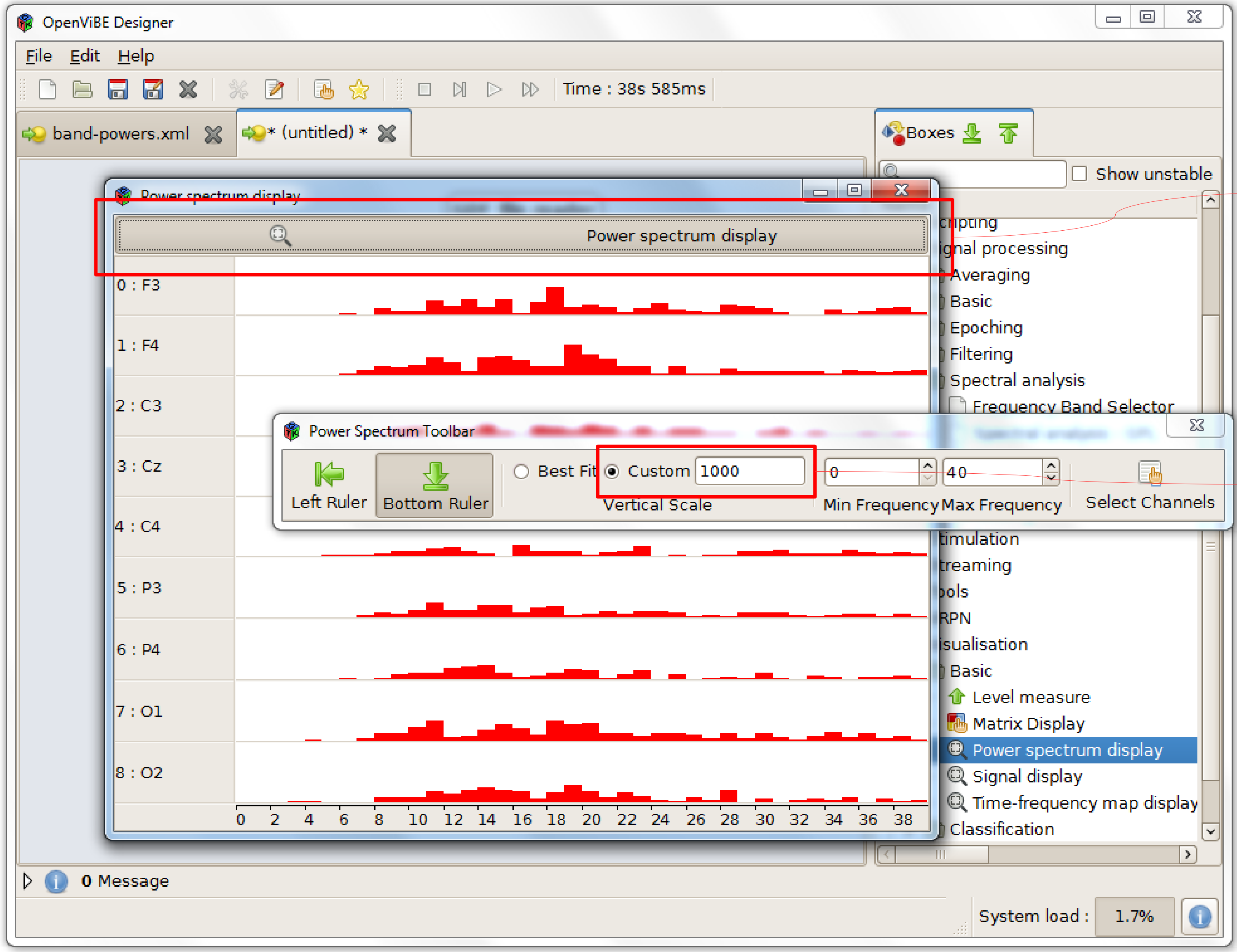
Computing a complete spectrum of the signal



The power spectrum is computed in real time. The amplitude of each band between 0 and 40 Hz is shown independently

Default settings of the *Power Spectrum Display* automatically update the datascale to fit the widget height. This is sometimes not convenient and can be changed.

Computing a complete spectrum of the signal



Click on the title bar of the *Power Spectrum Display* widget to configure the visualisation settings

Change the vertical scale to *custom* and set the scale to 1000

You can also change the Min and Max frequencies to display in this toolbar

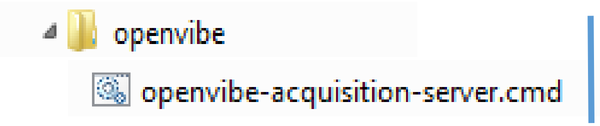
Using the acquisition server

OpenViBE

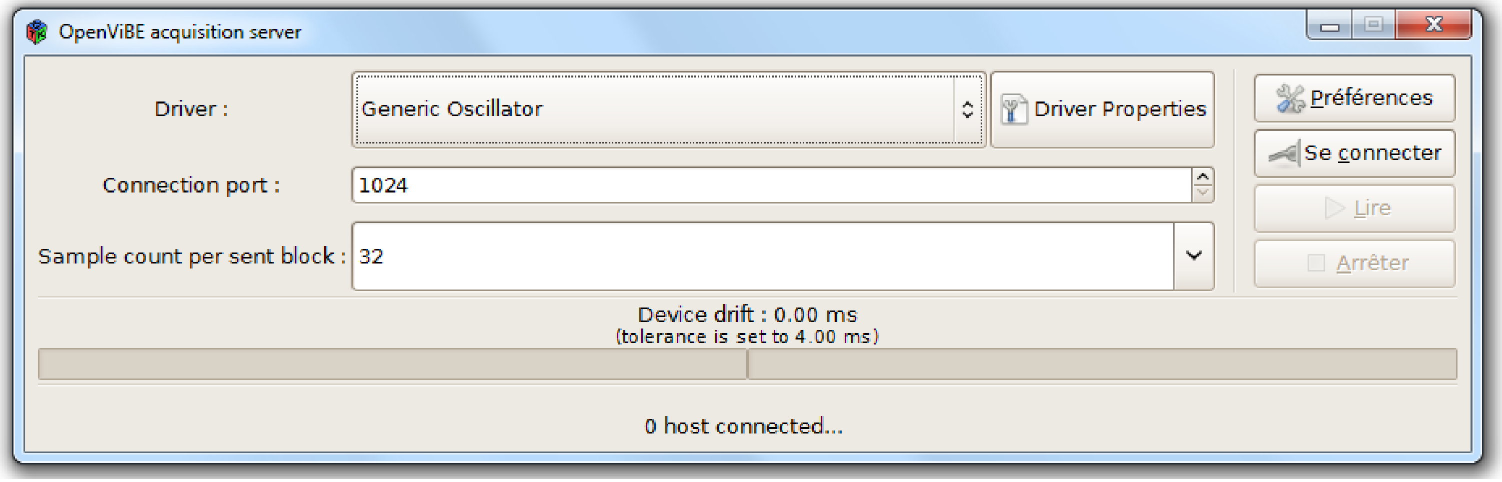
Using the acquisition server

In order to start the acquisition server, proceed as follows :

In the Windows shortcut list :
(Start → OpenViBE → OpenViBE Acquisition Server)



Directly start the *openvibe-acquisition-server.cmd* script in the OpenViBE installation folder

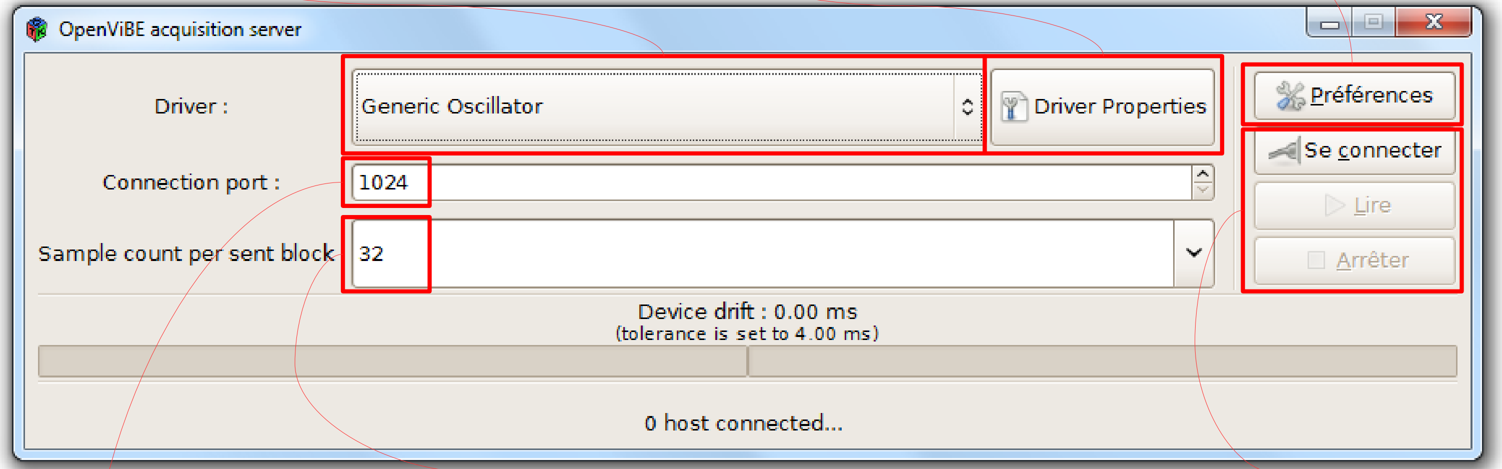


Using the acquisition server

Drop down list with all supported acquisition devices

Configuration of the driver for a specific acquisition device

Configuration of the acquisition server itself



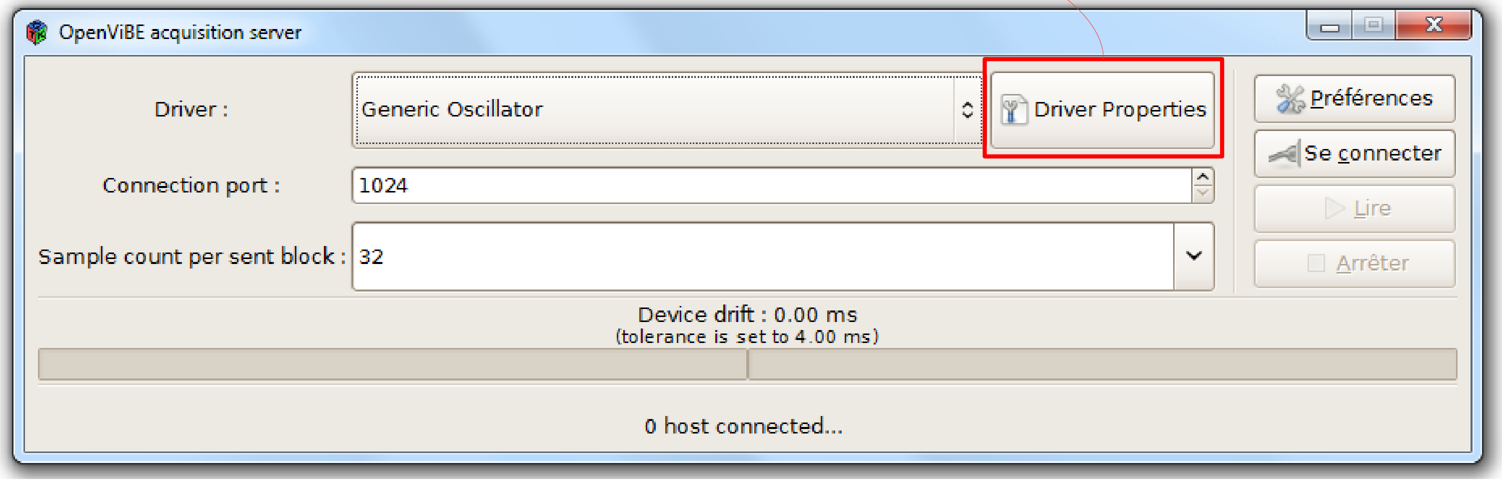
TCP port to use so that other applications can receive the acquired data

Sample count per chunk of data

Control buttons to connect / disconnect the device and start / stop the acquisition process

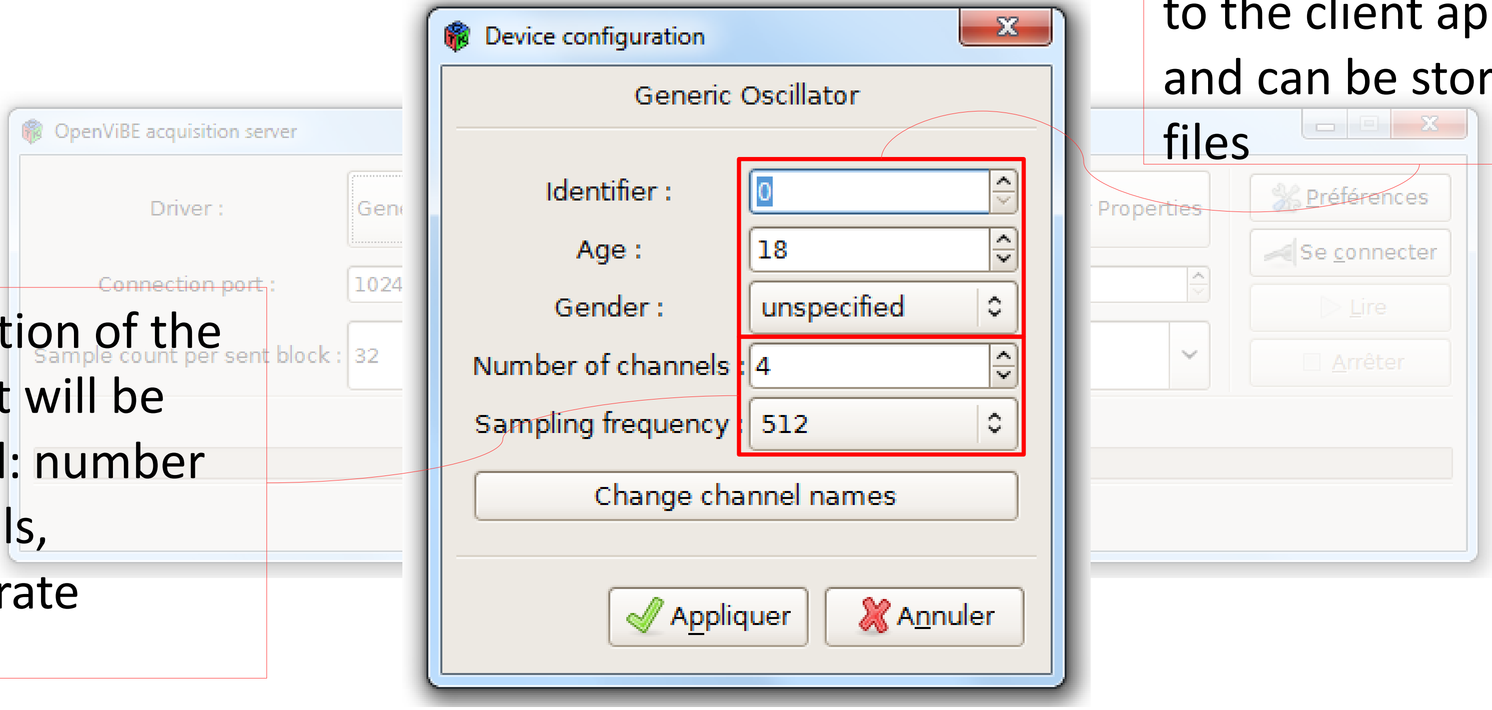
Using the acquisition server

Let's configure the
Generic Oscillator



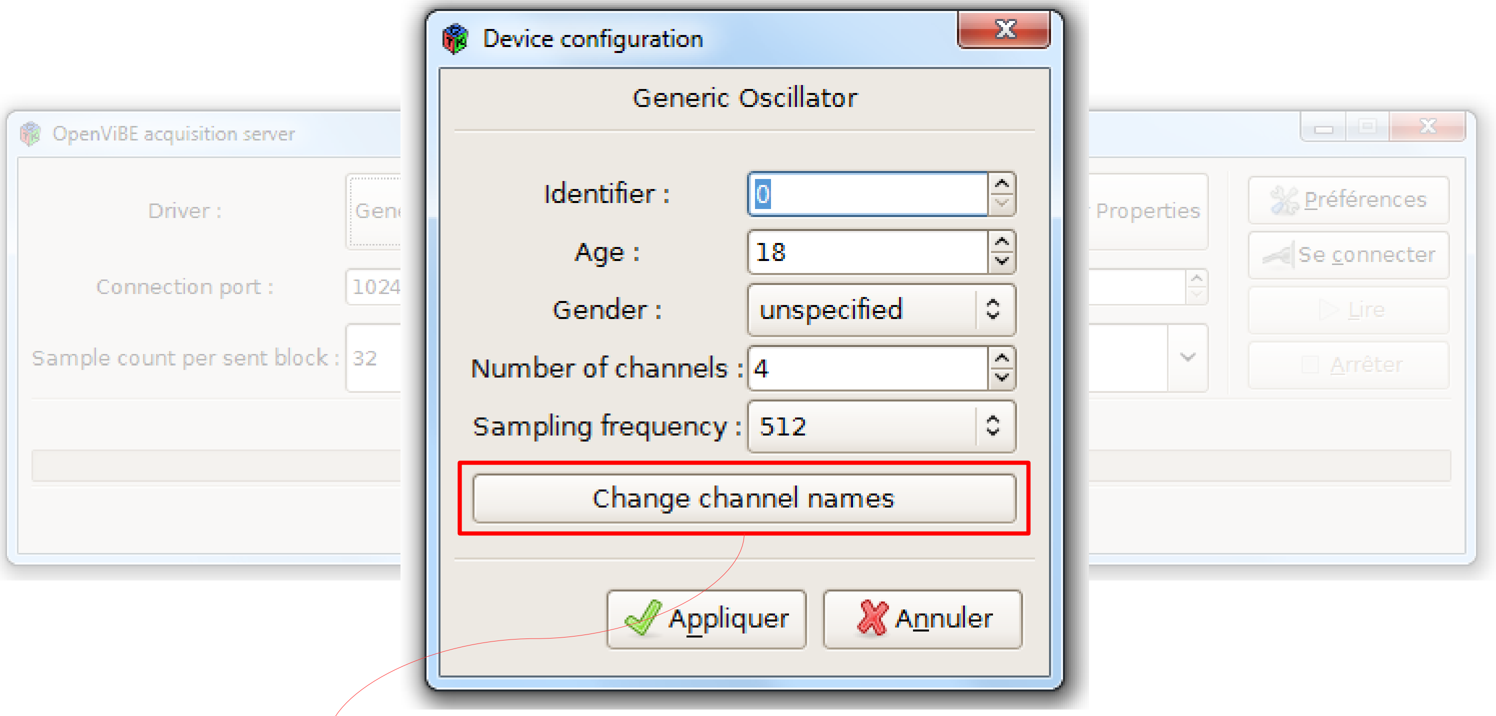
Using the acquisition server

Id information about the acquisition. These information will be sent to the client application and can be stored in files



Configuration of the signal that will be generated: number of channels, sampling rate

Using the acquisition server

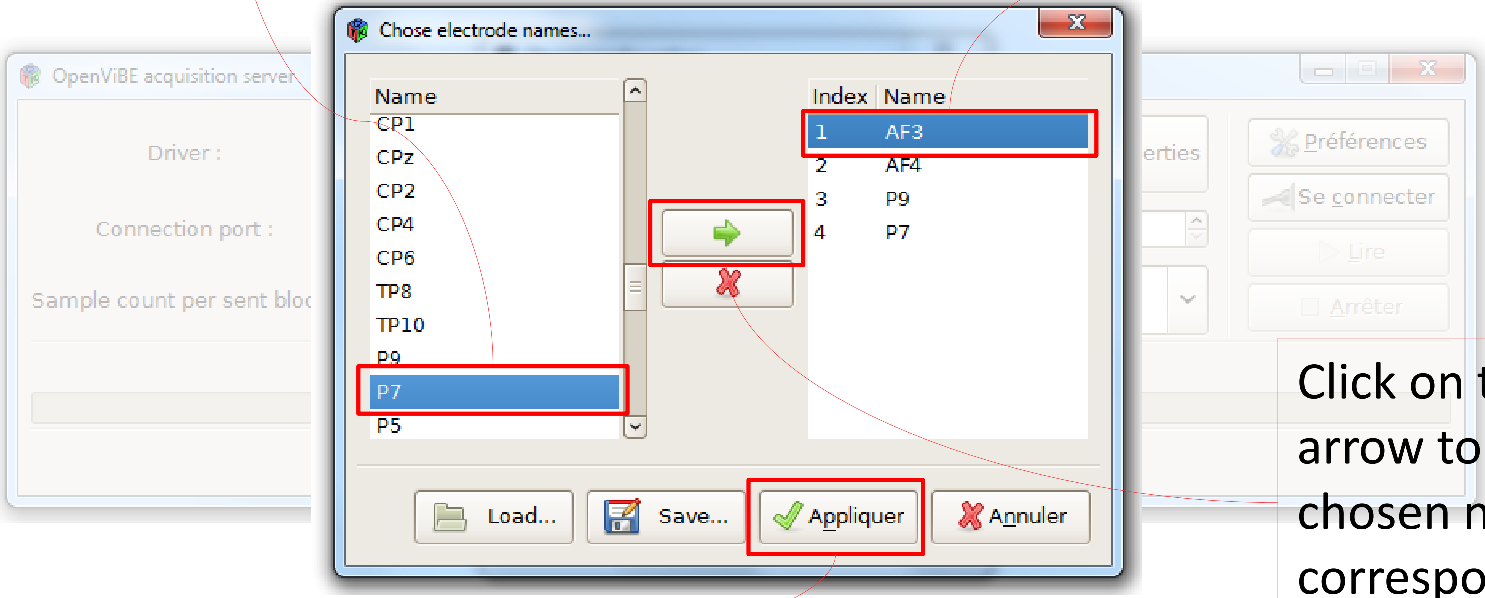


Let's change the channel names

Using the acquisition server

Select a name in the left panel

Select a channel in the right panel

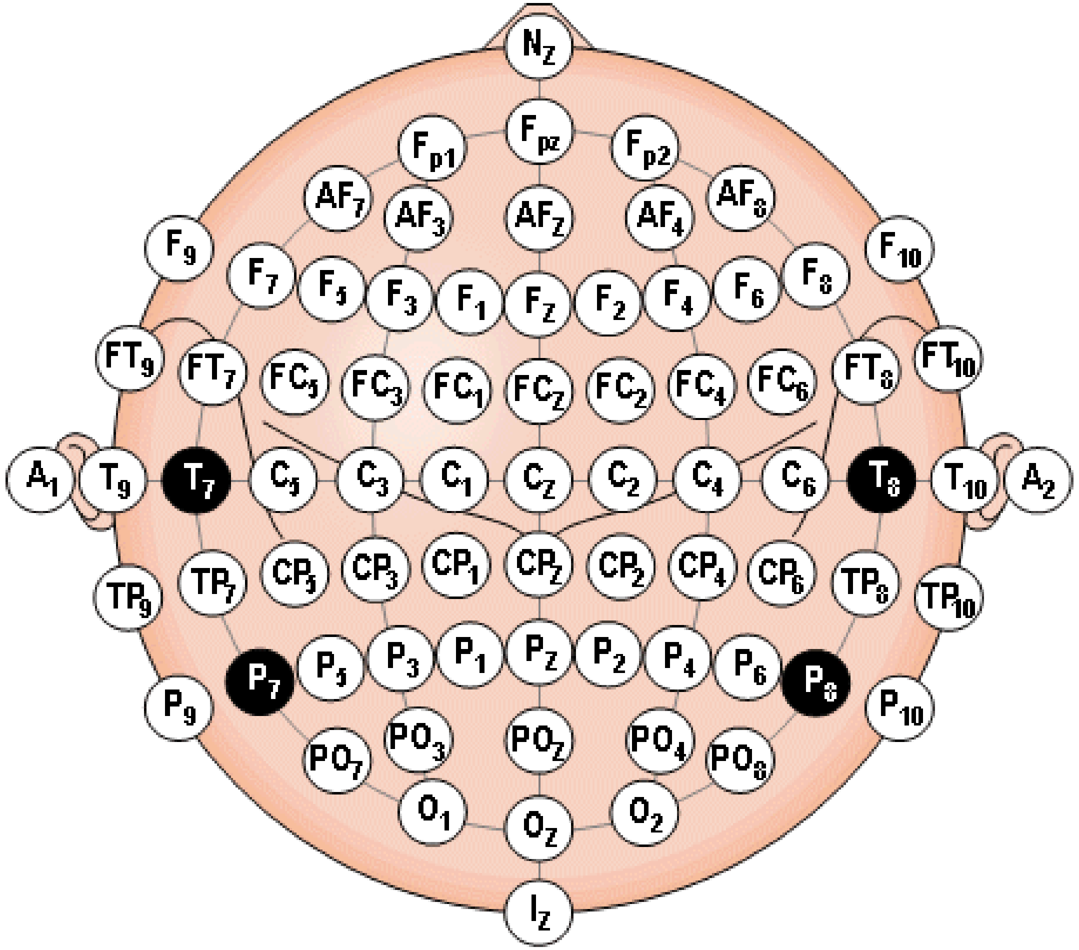


Click on the green arrow to apply the chosen name to the corresponding channel.

Now apply the changes

Using the acquisition server

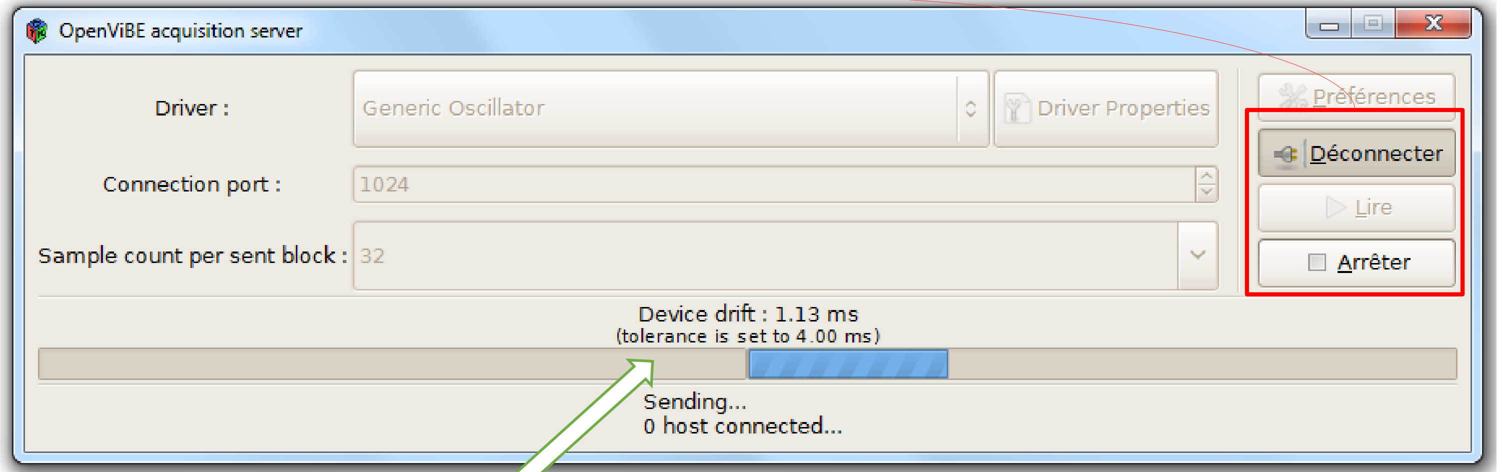
The international10-20 system normalizes the electrode names and locations.



Using the acquisition server

Click on *Connect* then *Play* to start reading data.

The data stream can be read by an acquisition client from the address *hostname:1024* on the same computer, it would be *localhost:1024*



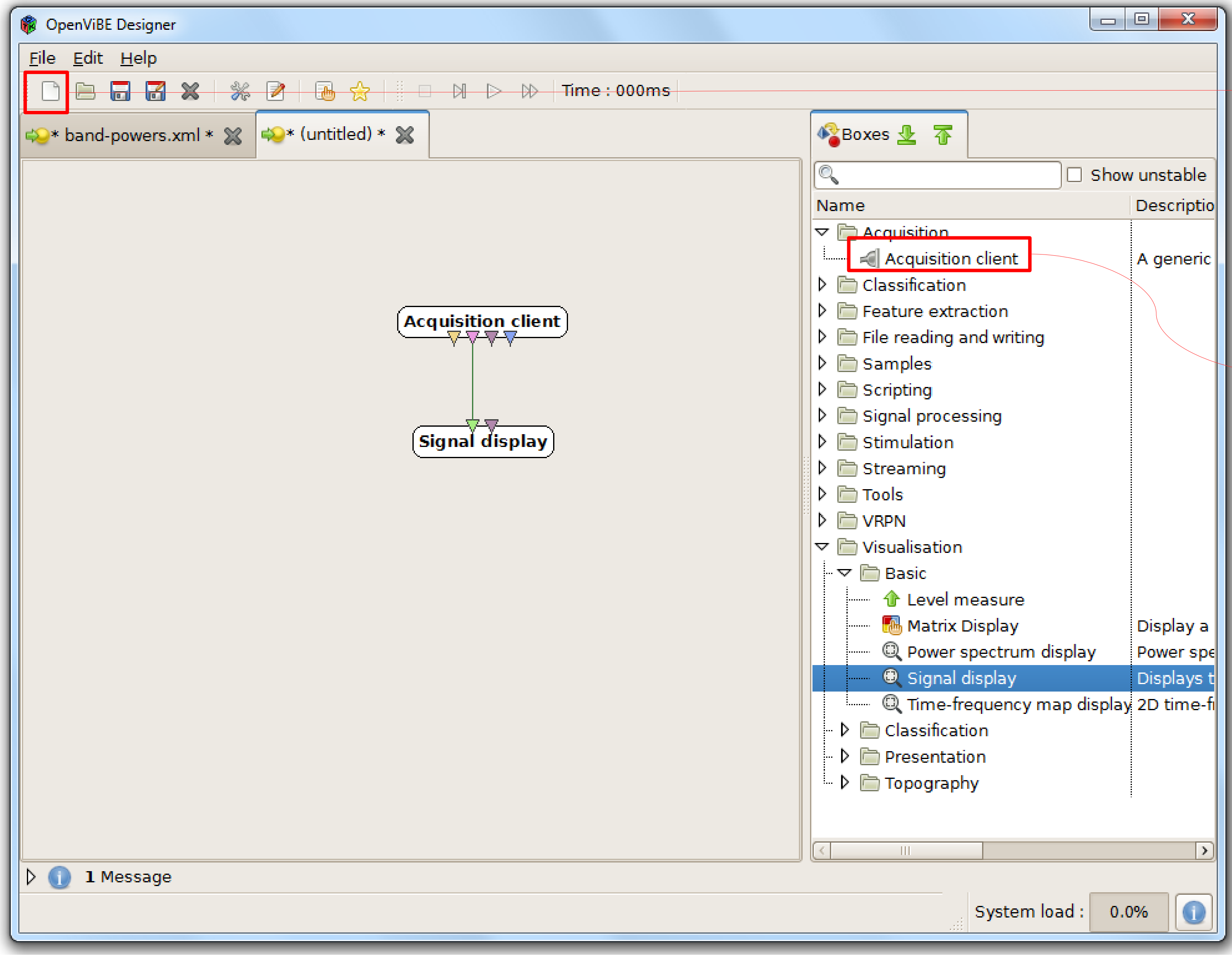
The acquisition reflects in realtime the device drift. The drift is the difference between the number of samples the driver sent to the acquisition server as compared to the number of samples it should have sent based on the elapsed time.

If the drift is too important, this can cause some timing and tagging issues, most particularly when it comes to ERPs such as P300, which requires a perfect match between the EEG stream and the event tagging. In this case, the server will apply a correction by removing or adding samples.

Visualizing the acquired signal in the designer

OpenViBE

Visualizing the acquired signal in the designer

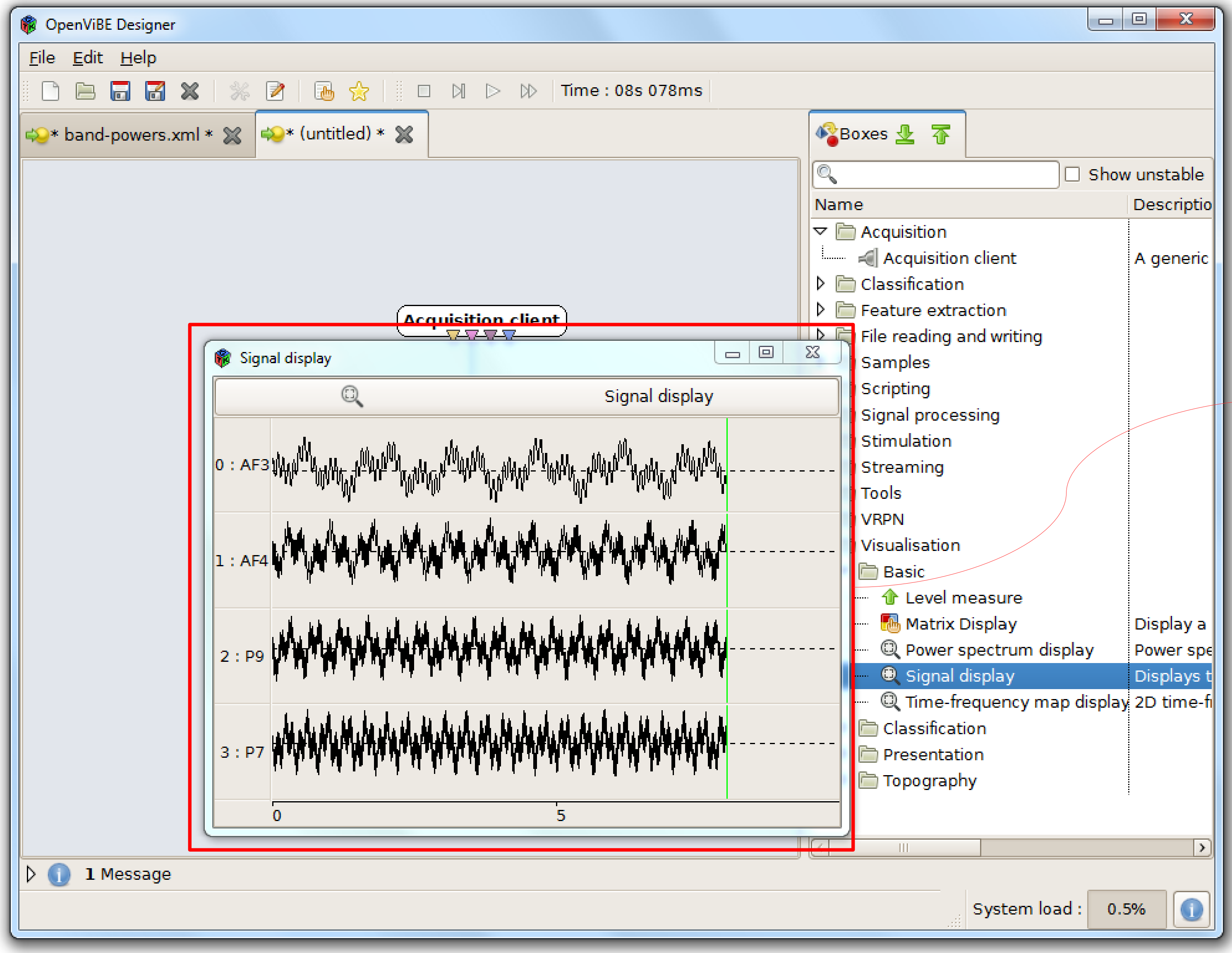


Create a new scenario

Drag & drop an *Acquisition Client* box and a *Signal Display* box.

Default settings will look for the acquisition server at *localhost:1024*

Visualizing the acquired signal in the designer

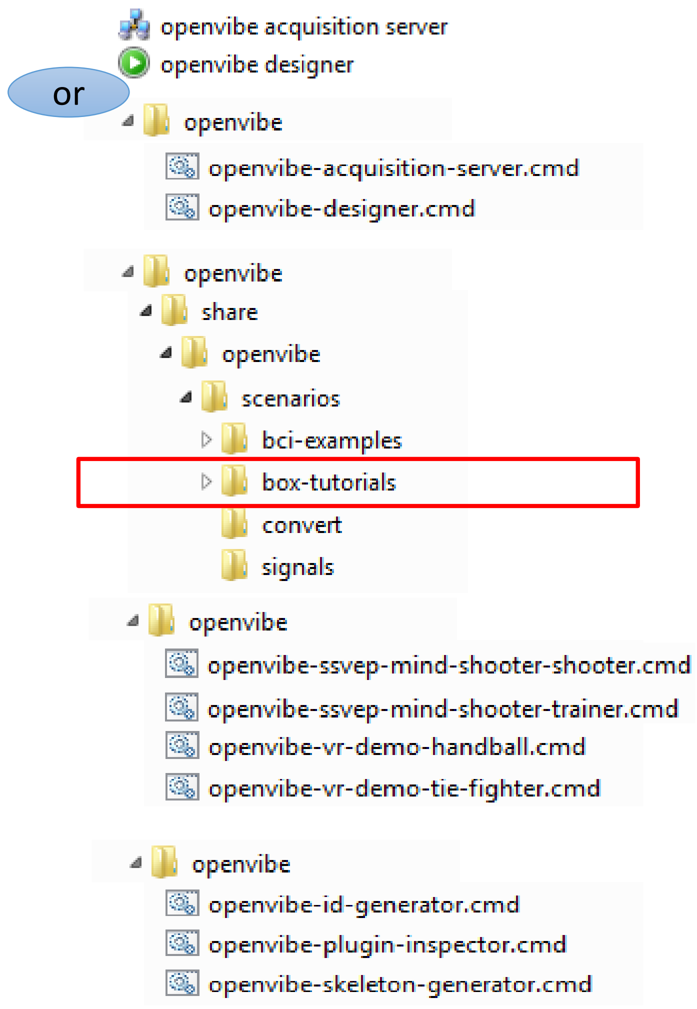


Press the *Play* button to see what's in the Acquisition Server stream

The box tutorials

OpenViBE

OpenViBE tree structure



Main applications :
Acquisition Server or Designer

Sample scenarios :
Basic tutorials (*box-tutorials*) and advanced scenarios (*bci-examples*), plus signal files and format conversion scenarios

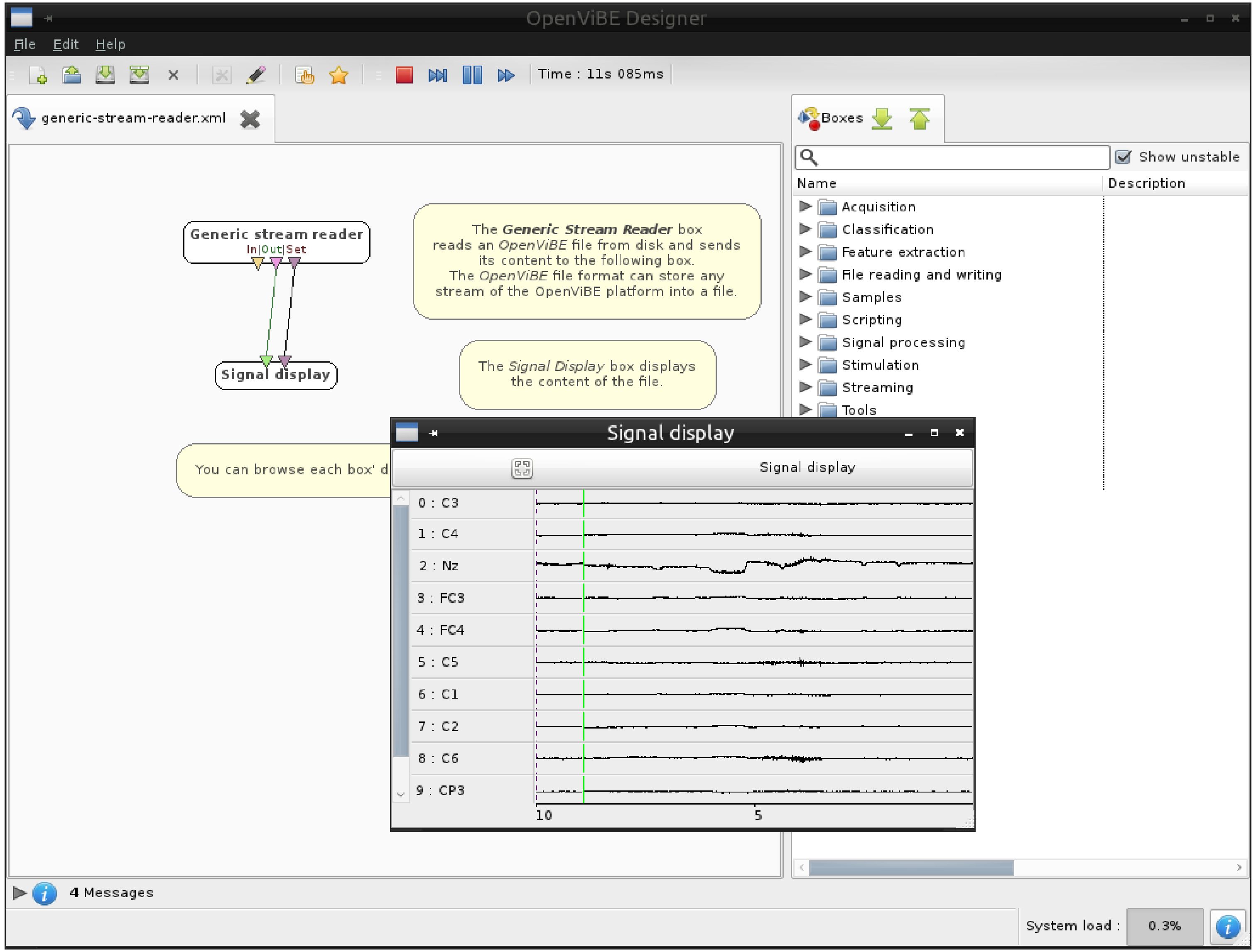
2D/3D Demo applications for advanced scenarios :
Handball, tie-fighter, ssvep shooter

Developers and contributors tools

Start the Designer :

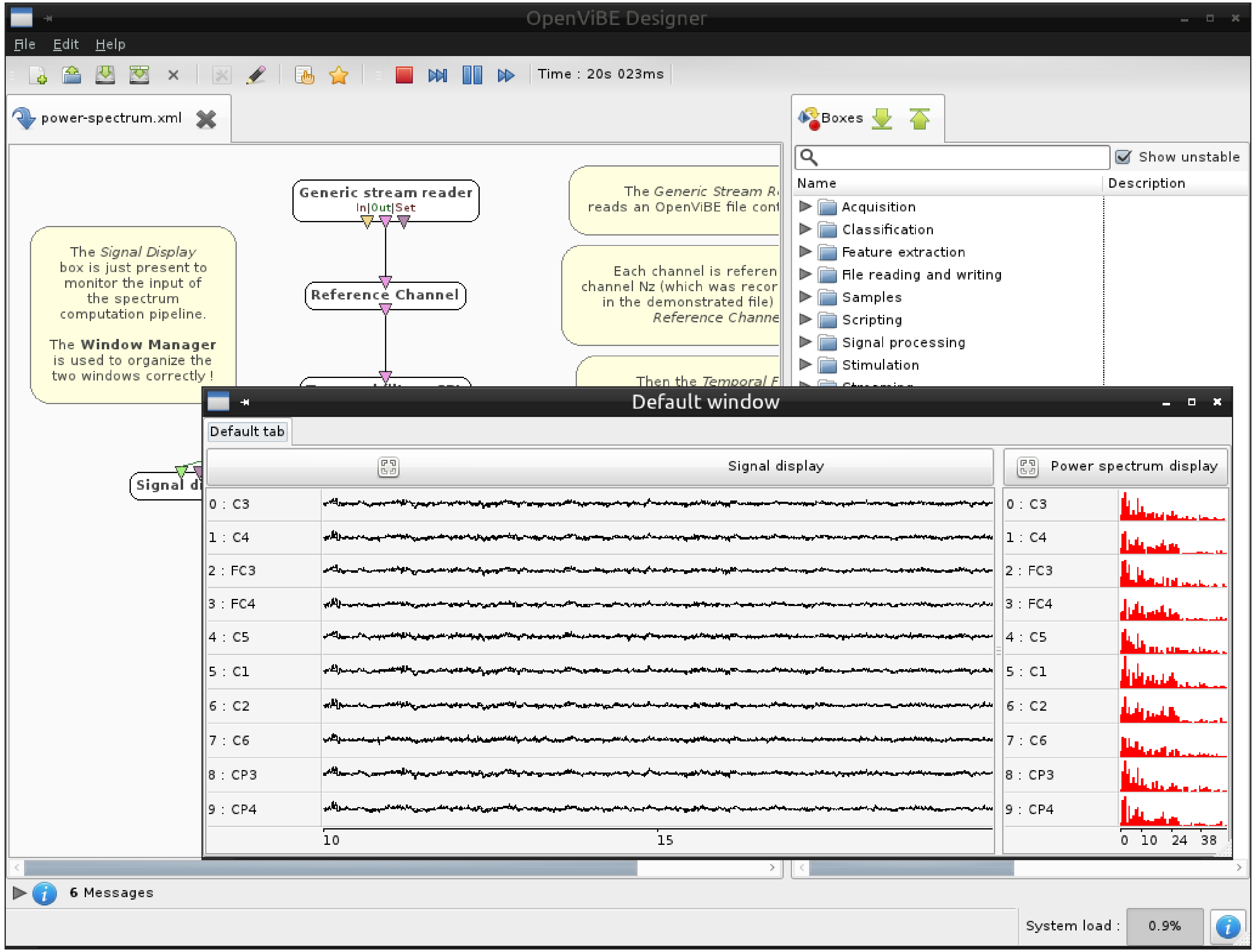
- **From Start menu (Start → OpenViBE → OpenViBE designer)**
- **From the file explorer, directly execute *openvibe-designer.cmd* in the OpenViBE folder**

Generic Stream Reader

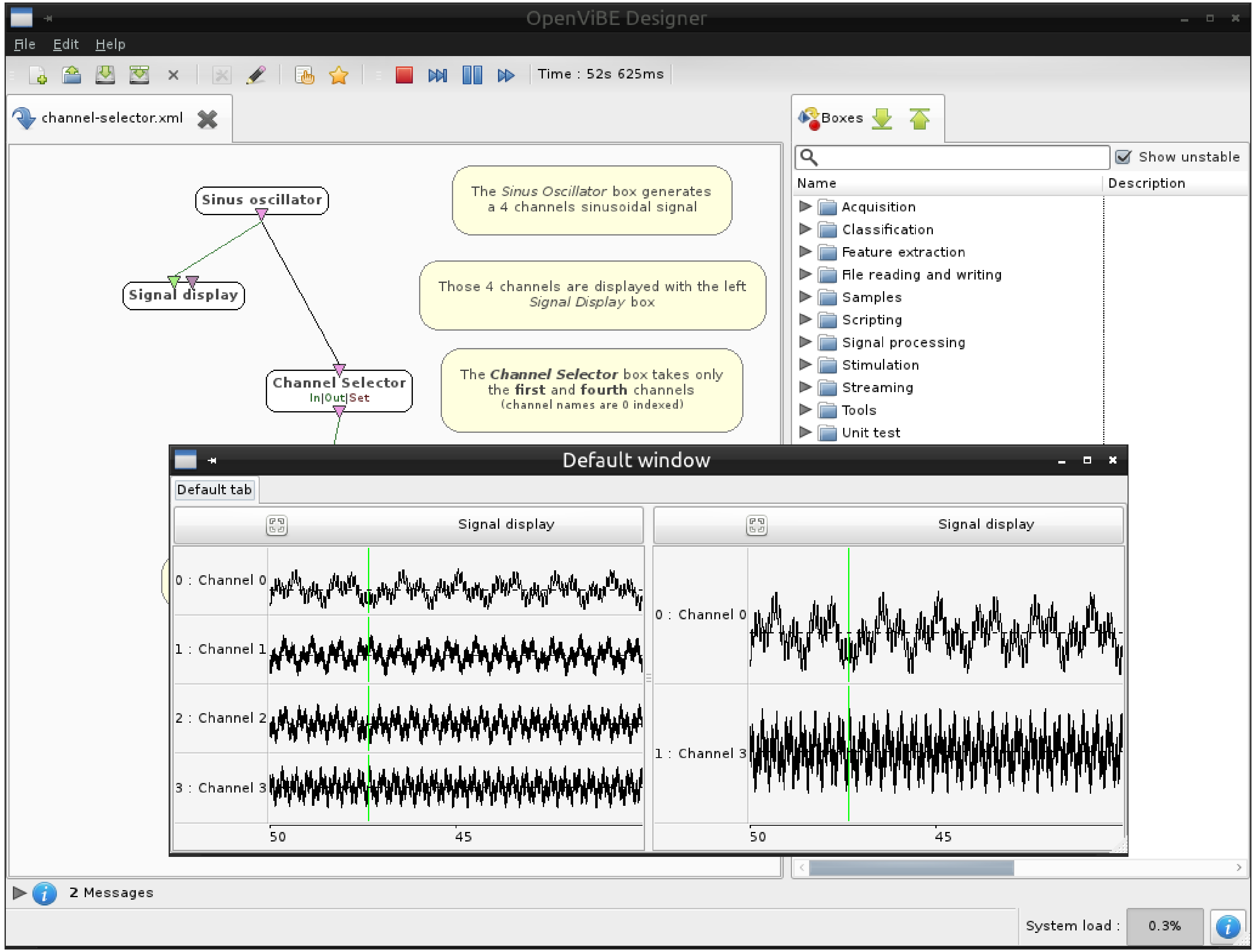


The OpenViBE file format (.ov) is simply the binary data stream written in a file, same format as what goes between boxes

Power Spectrum Display

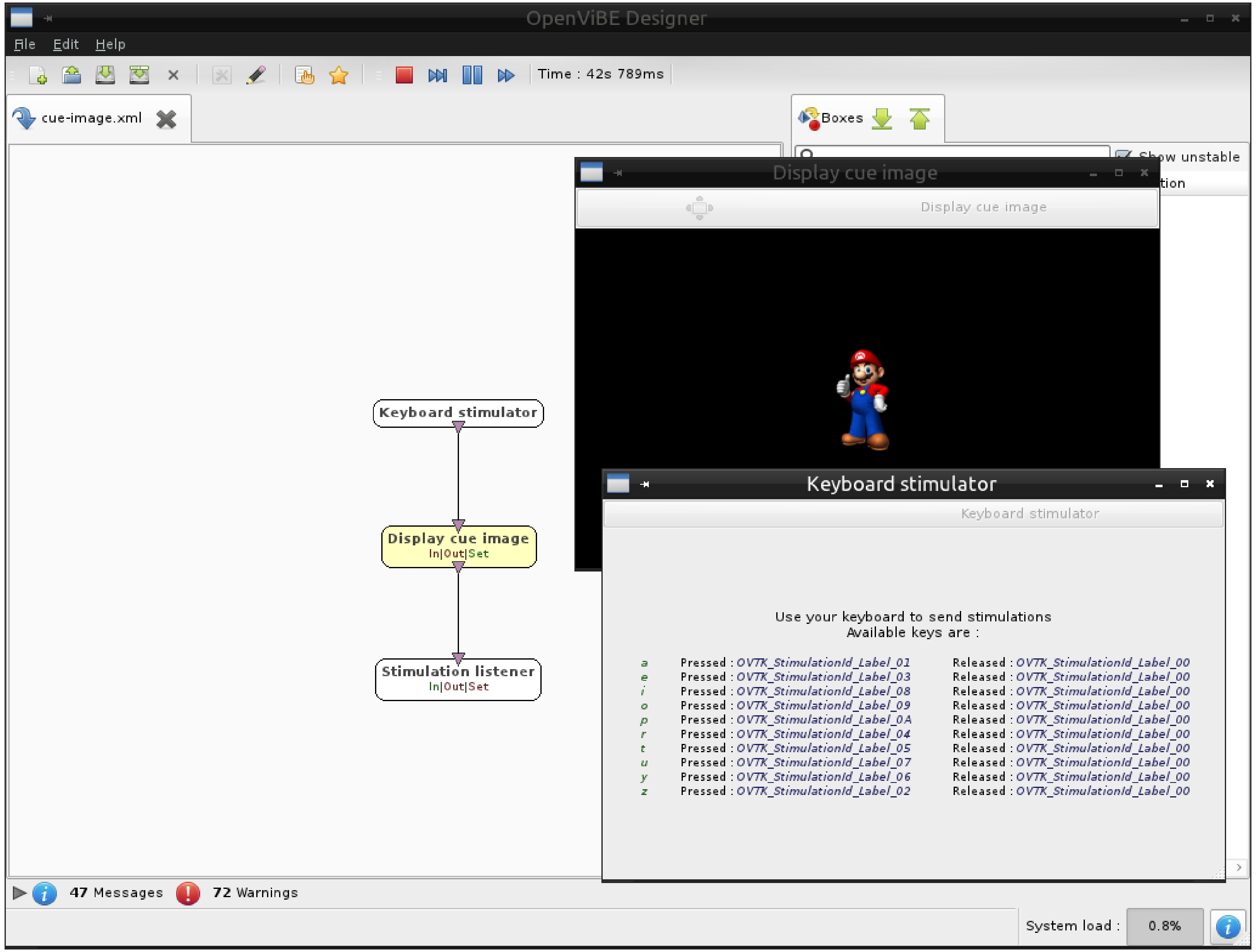


Channel Selector



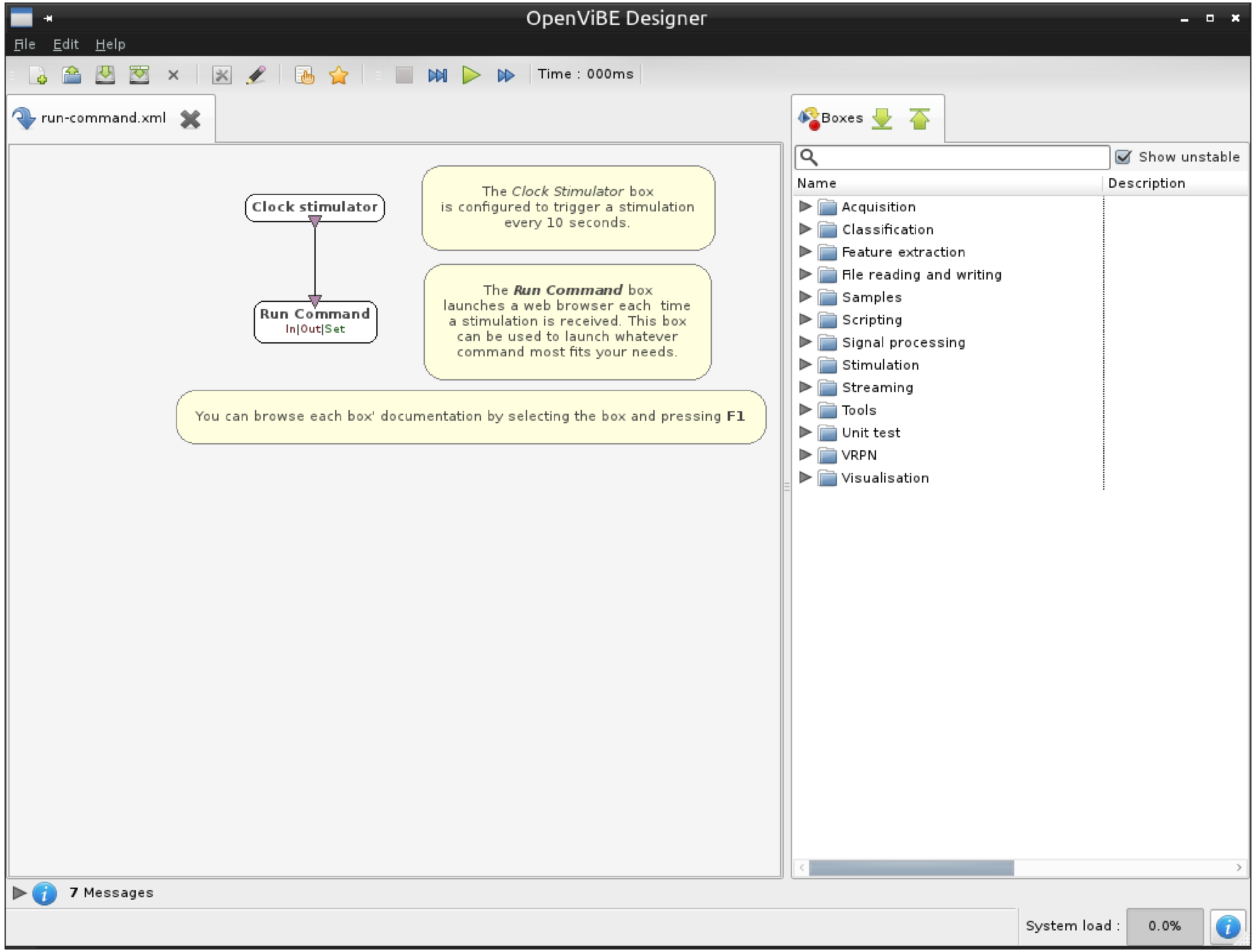
A very useful box to select or exclude specific channels based on their name or index

Display Cue Image



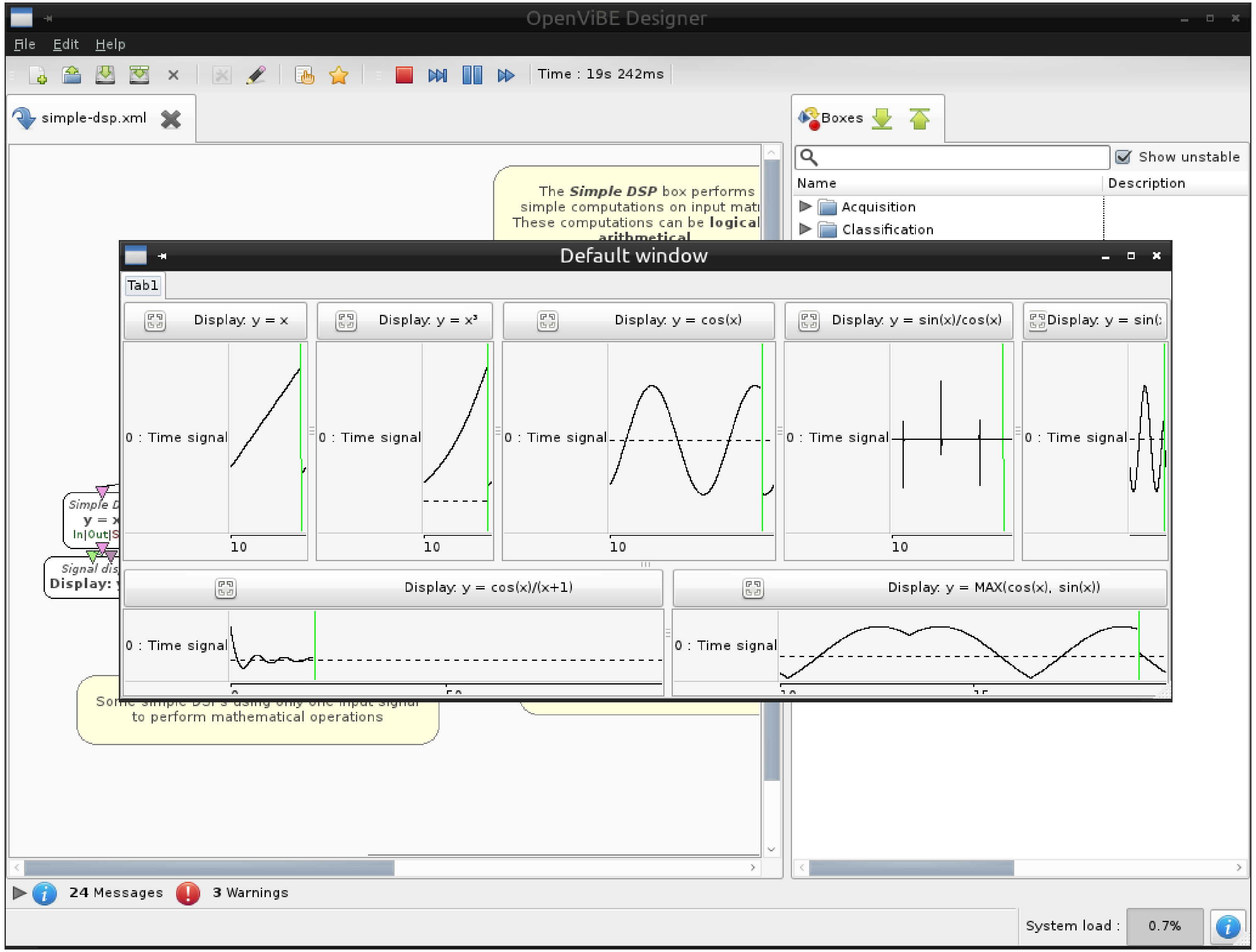
Can display feedback, rewards, or instructions for example.

Run Command



This box is able to run any system command, as you would do in a command prompt on Windows or Linux. For example, you can automatically start an external application that will connect to OpenViBE to display feedback

Simple DSP

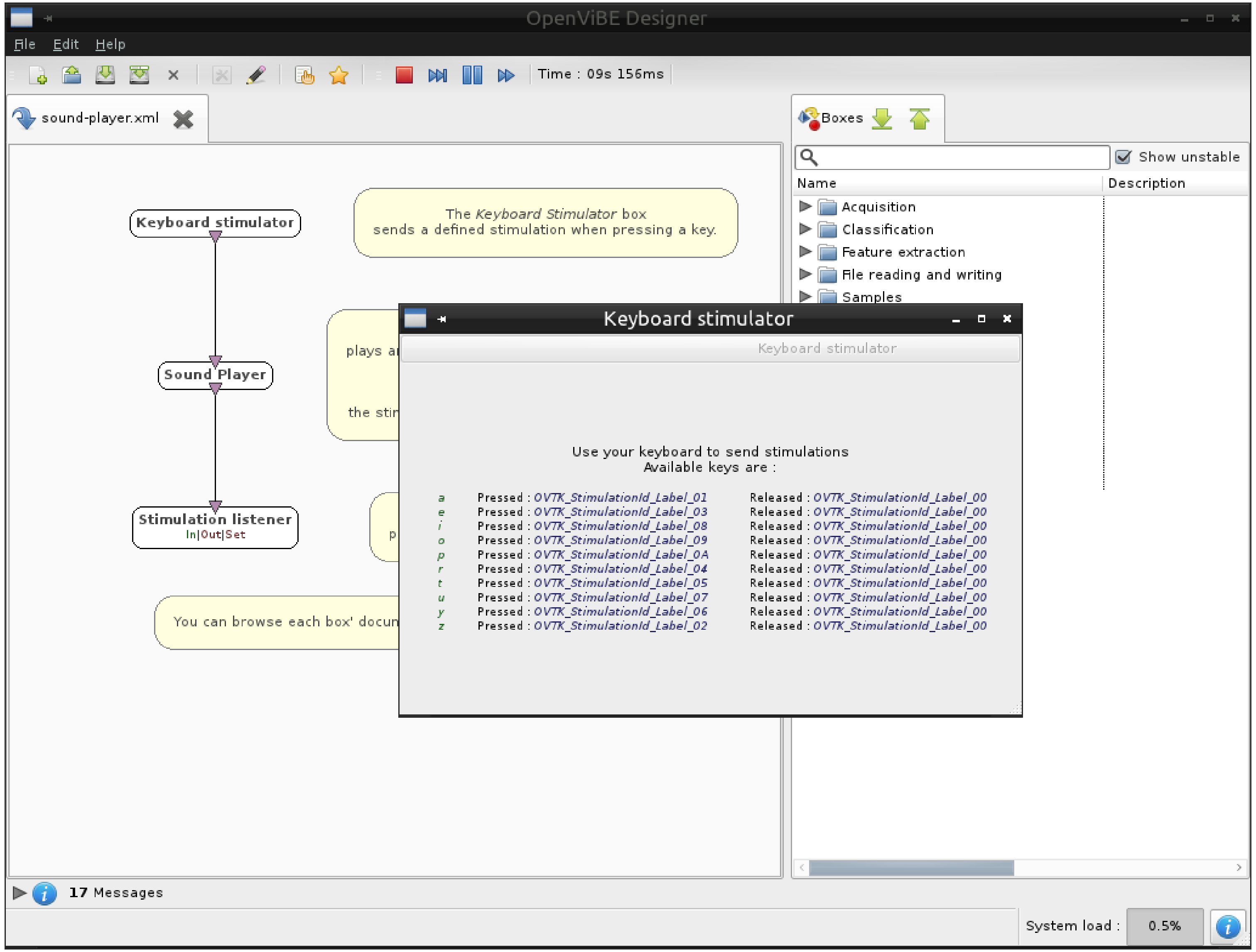


A very versatile box for basic mathematical operations on signals. See the box documentation for the complete syntax of the equation to set

Sign Change Detector

Can be used to detect if a signal is above a specific threshold for example

Sound Player

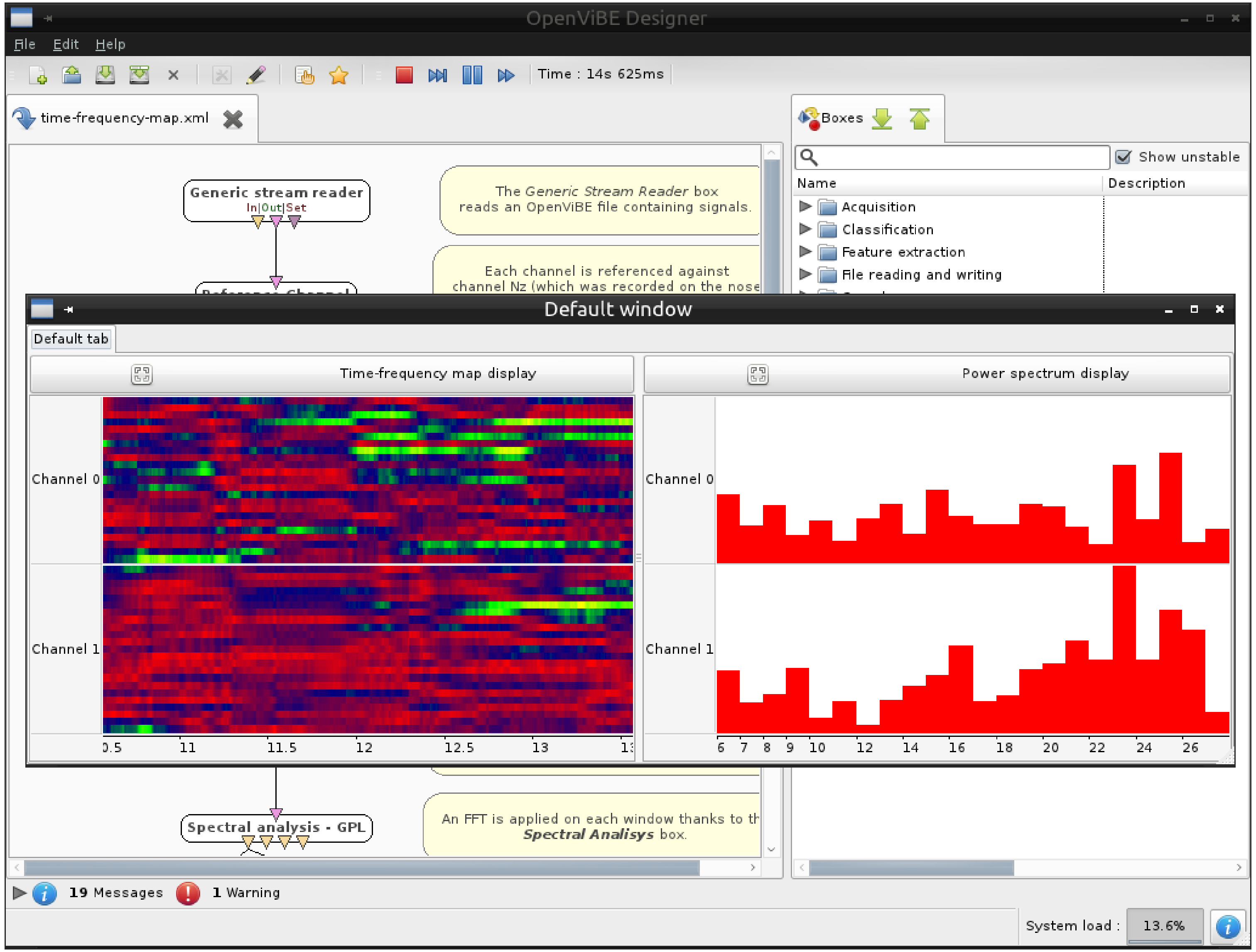


The Sound Player is based on the OpenAL library and can read files in .ogg and .wav formats

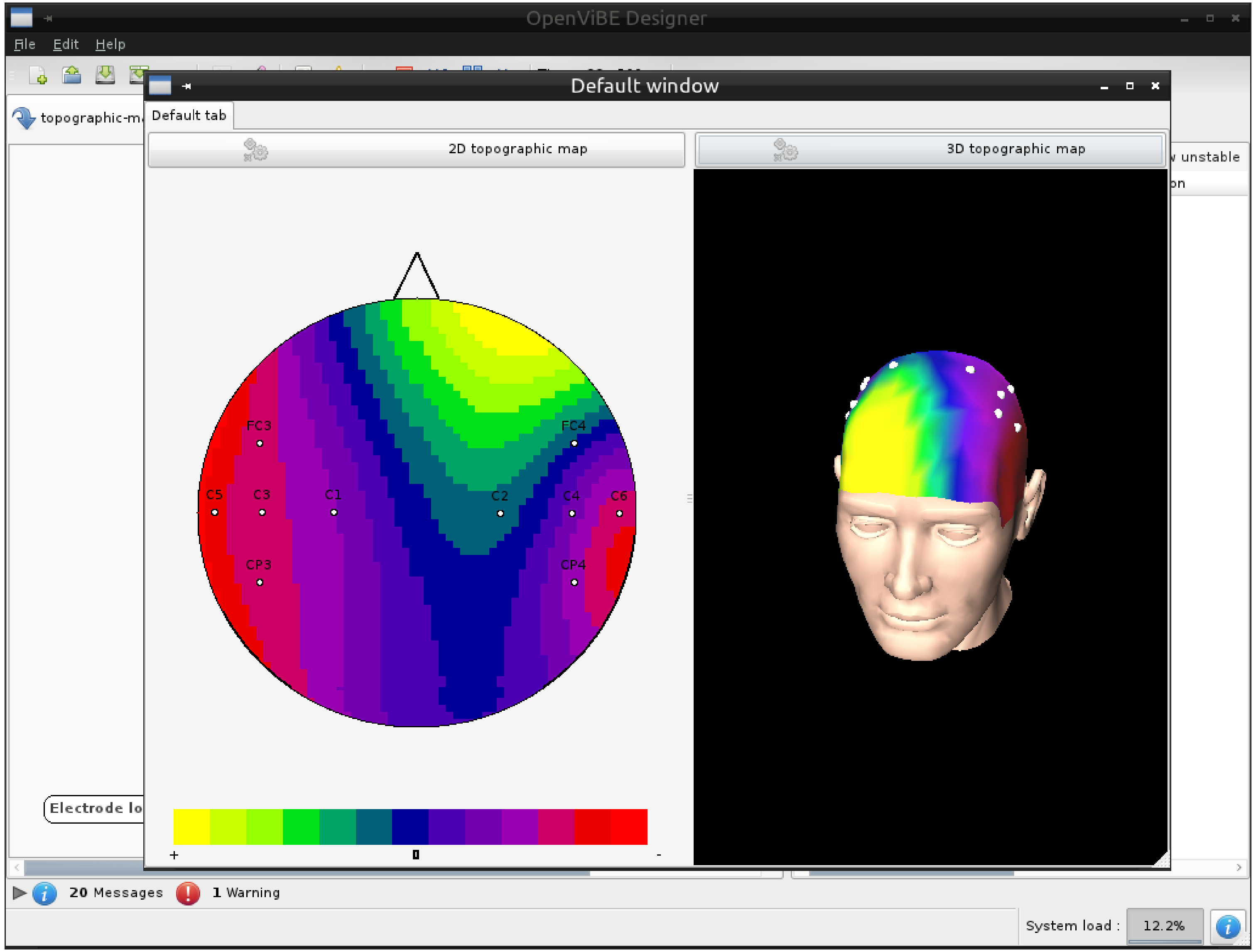
Stream Switch

Can be used to redirect an input signal to one or another processing pipeline depending on the context.

Time Frequency Map Display



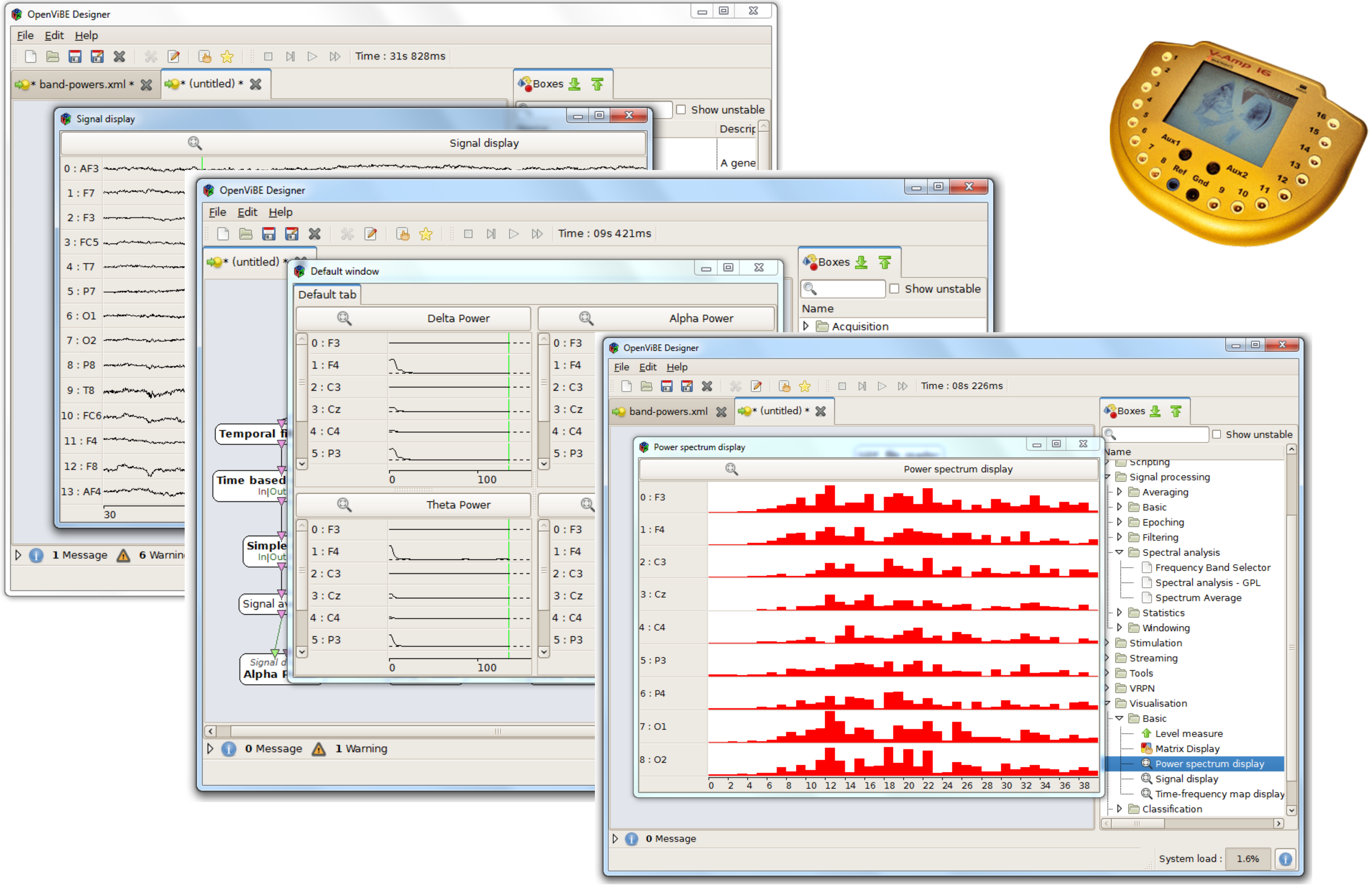
Topographic Maps



Moving ahead: reading signals from a real device

OpenViBE

Reading signals from a real device



Exercices

1. Display alpha activity on a topographic map with alpha levels in occipital areas and observe the effects of eyes open eyes closed
2. Send BIPS at a regular pace and observe the averaged EEG response on every channels

Contacts :

David Ojeda, R&D Engineer

do@mensiatech.com

Louis Mayaud, CSO

lm@mensiatech.com

+33.(0)6.50.66.34.91

http://www.mensiatech.com