NIRx Trigger Manual

R2.1 – 2018-11-22

Table of Contents

1. Introduction 2
   1.1 General introduction 2
2. Adding event markers to fNIRS data by hand 3
   2.1 Adding event markers while recording data 3
   2.2 Adding event markers after recording data 4
      2.2.1 The NIRStar *.evt file 4
      2.2.2 The Aurora fNIRS *.nirs file 5
      2.2.3 Editing event markers in analysis platforms 5
3. Adding event markers to fNIRS data using hardware 6
   3.1 Trigger input connector 6
   3.2 Trigger remote control 7
   3.3 Parallel port 7
   3.4 Parallel port replicator 9
   3.5 Cedrus c-pod or StimTracker 9
4. Adding event markers to fNIRS data using software 10
   4.1 NIRStar 10
   4.2 NIRStim 10
   4.3 Lab Streaming Layer (LSL) 11
      4.3.1 LSL in NIRStar 15.2 11
      4.3.2 LSL in Aurora fNIRS 13
   4.4 The Trigger.nrx file 13
5. How to send triggers 14
   5.1 NIRStim 14
   5.2 PsychoPy 14
      5.2.1 Sending triggers in PsychoPy over a parallel port. 15
      5.2.2 Sending triggers in PsychoPy using the Cedrus c-pod 16
      5.2.3 Sending triggers in PsychoPy using LSL 16
   5.3 Alternative options for Stimulus presentation 22

1. Introduction

1.1 General introduction

fNIRS experiments are most often based on the analysis of hemodynamic changes that occur as a result of behavioral changes (e.g. motor task performance) or changes in the environment (e.g. presentation of visual stimuli). Depending on the nature of the experiment, these changes can be either investigated in series (i.e. block-design) or individually (i.e. event-related design) (Fig. 1). Independent of the experimental design chosen, it is important that the onset of stimuli is accompanied by a corresponding event marker, so that different behavioral or stimulus conditions can be distinguished from one another, and data analysis techniques can be used to assess whether accompanying hemodynamic changes statistically differ.

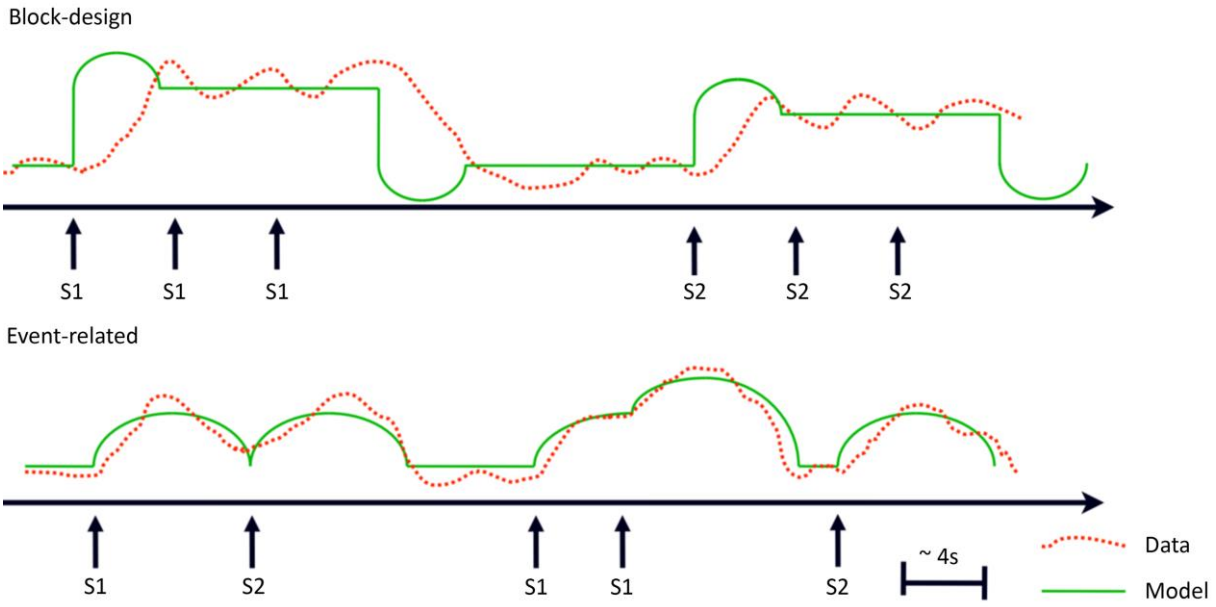

Figure 1: Schematic display of a block-design (top) and event-related design (bottom). 'S1'and 'S2' correspond to differing stimulus types. Note that in a block-design, the statistical model is used to investigate series of stimuli, whereas in the event-related design it is used to investigate single instances of each stimulus.

To correctly register the onset of behavioral and stimulus conditions, event markers have to be used. Event markers, oftentimes referred to as triggers, mark the period of time corresponding to the onset of the activity of interest. During data acquisition in NIRStar or Aurora, event markers will appear on the data recording screen as a dashed line (Fig. 2). These markers will be stored in a trigger event file (NIRStar: *.evt file; Aurora: *_config.json file) which will be saved in the data folder of the recording.

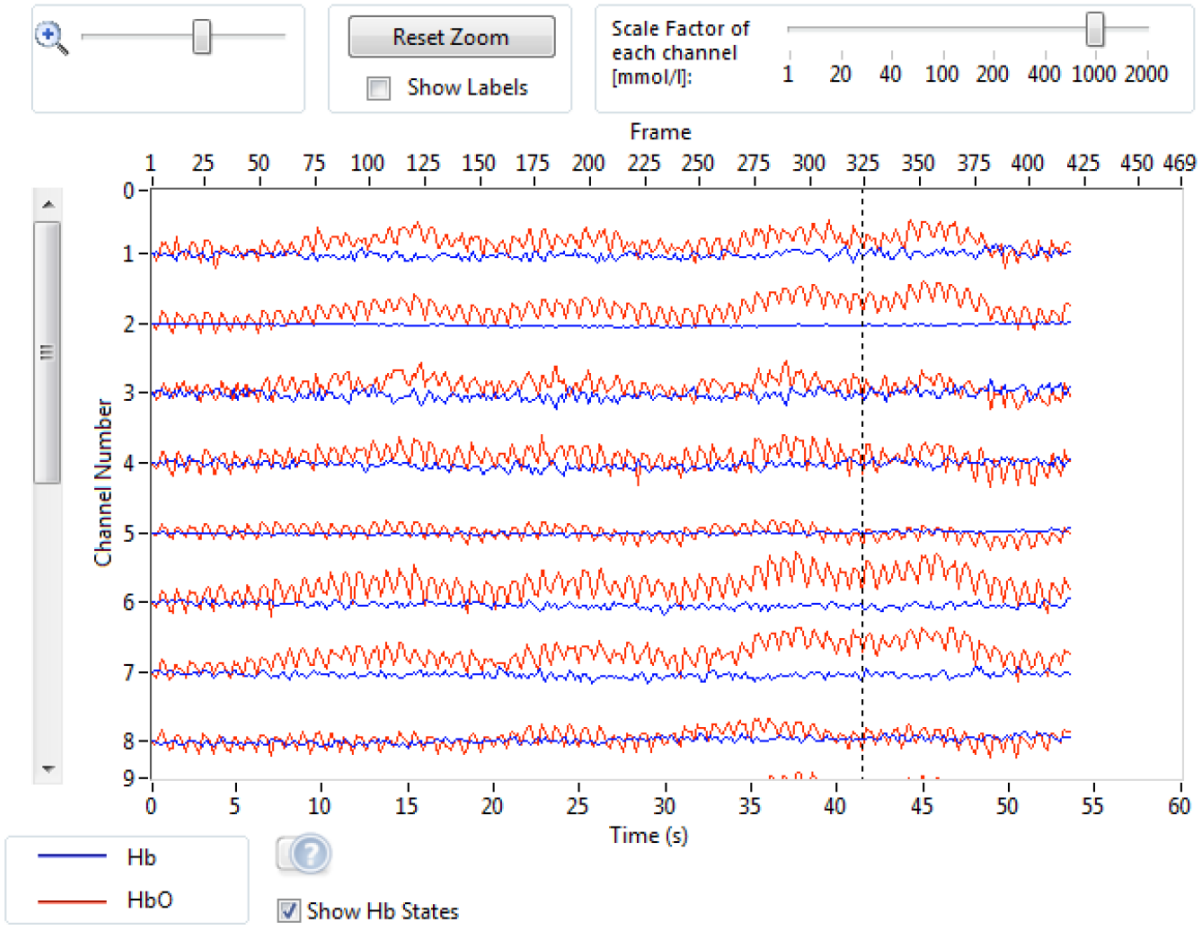

Figure 2: Example of an event marker placed during recording in NIRStar (at ~41s). Note that the corresponding number of the event marker is not displayed. However, this information is stored in the event file.

The current document outlines the various possibilities of obtaining event markers in the data recorded by NIRx fNIRS instruments – both in terms of hardware as well as software options. Specifically, the following chapters cover how event markers can be added to the data manually, or by sending triggers using various types of hardware, by using specific software, or by using a combination of hardware and software.

2. Adding event markers to fNIRS data by hand

2.1 Adding event markers while recording data

Although event markers are typically obtained in recorded data by means of external trigger signals, the NIRStar software also allows the user to set event markers by sending triggers manually during the course of the experiment. For both the NIRScout and the NIRSport, a manual marker can be recorded by either clicking the respective button (see Fig. 3) or by pushing the corresponding keys on the keyboard (i.e. F1,F2,…F8). For the NIRSport, four marker numbers are available, allowing the user to distinguish between different types of event. For NIRScoutX systems, eight manual markers are available. Note that each marker can be used as many times as the user desires.

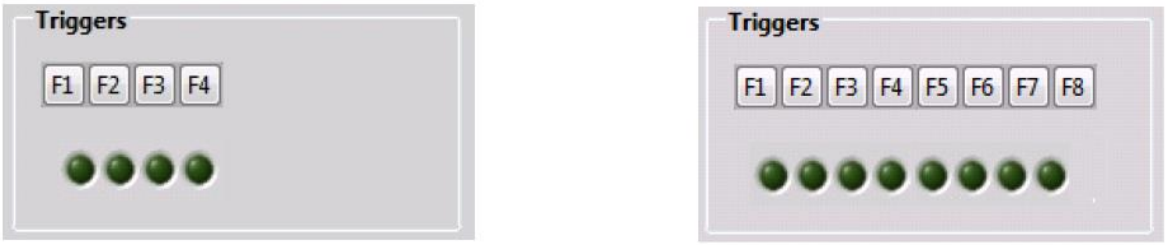

Fig. 2: Different trigger displays: For a NIRSport system, four manual markers are available (left), whereas for the NIRScoutX eight manual markers are available (right).

Tip: While adding event markers manually may not be the most precise way to discriminate between experimental conditions, it is very useful for marking unexpected experimental events (e.g. motion artifacts, changes in environmental light, etc.). A lab journal, or the experimental notes window in the setup screen, may be used in conjunction. Importantly, when sending triggers by hand, the users should ensure that the event markers that are used, are not already assigned to experimental conditions.

For more information on how triggers and event markers are implemented in NIRStar, please consult Section 10.3 of the NIRStar User Manual.

For Aurora fNIRS (Version 1.0), there is currently no option to manually add markers to the experiment.

2.2 Adding event markers after recording data

The preferred method is to register event markers by sending triggers during the experimental session. However, it is also possible to manually add or remove event markers after the recording, during analysis. For NIRStar, event markers are stored in the *.evt file. For Aurora fNIRS we currently advice to use the *.nirs file for data analysis. If no event markers have been recorded during the experiment, the data folder will still contain a file that is intended to log temporal information of the experimental session, but it will be empty. Below more explanation is given for the *.evt file and *.nirs file, respectively.

2.2.1 The NIRStar *.evt file

The *.evt file records the event marker channel code and frame number in a format compatible with NIRx NIRSport and NIRScout systems. Each row of the tab-separated table corresponds to a recorded trigger event. The first column contains the frame number at which a trigger was received. Columns 2-9 record the trigger markers, for up to 8 trigger input channels. ‘1’ indicates that an input received a trigger signal. Because NIRScout and NIRSport systems have a maximum of 4 digital input lines, columns 7-10 are always 0 (as no trigger signal can be received on these input lines).

In table 1 below, which is identical to the tab-separated table in the *.evt file, digital input 1 (DI1) was triggered during frame 18, DI4 was triggered during frame 27, DI15 during frame 31, etc.

18	1	0	0	0	0	0	0	0
27	0	0	0	1	0	0	0	0
31	1	1	1	1	0	0	0	0
35	0	0	0	1	0	0	0	0
50	1	1	1	1	0	0	0	0

*Table 1: Example tab-separated table containing event marker information, as in the *.evt file.*

Note: In addition to the *.evt file the *.hdr file also contains information about event markers too – in case triggers have been sent. In the '[Markers]' section of the *.hdr file, a tab-separated table consisting of 3 columns contains the onset time in seconds in the first column, the event marker in the second column, and the time in frames in the third column. This information may help to identify event markers in case reading the *.evt file is found difficult. However, for data analysis in nirsLAB only the information in the *.evt file will be used. Finally, trigger information can be stored in the *.nirs file. This is a Matlab file that will be created if the "Export data to Homer2 format" feature has been enabled under File Options (see Section 13.4.5 of the NIRStar User Guide). The *.nirs file contains information about acquired data and montage, and can be directly imported into Homer2. For more information see section 2.2.2 below.

2.2.2 The Aurora fNIRS *.nirs file

Although other data files in the Aurora fNIRS data folders contain information about triggers too, we currently advice the user to use the *.nirs datafile created. The *.nirs file is a matlab file that can be seen as the most basic unit for Homer2 analysis. The file will contain several fields, of which one field is *SD*. *SD* is a structured variable which contains both information regarding triggers as well as information regarding the source/detector geometry (i.e. the montage). Trigger information is stored inside the *s* variable. The *s* variable specifies the time points and conditions (event markers) of stimulus onset, with <number of time points> and <number of conditions> as dimensions. More information on how to work with triggers (conditions) in Homer2 can be found in the Homer2 User Guide.

2.2.3 Editing event markers in analysis platforms

While the user may add information to this file manually, analysis platforms also contain tools to edit event marker information. For example, in the nirsLAB software, an editor is available that allows the user to modify information of the *.evt file, or add information in case the file is empty (i.e. no event markers were recorded). Please see section 2.4 of the nirsLAB User Manual for more information. Another analysis platform, Homer2, has a similar function. Using the *stimGUI* from the Homer2_UI *Tools Menu*, stimulus marks (event markers) can be added, deleted and edited. See section 4.2 of the Homer2 User Guide for more information.

Importantly, it is generally not recommended to edit measurement-timing information in the data during analysis. Not only can this be a lot of work, it is also very sensitive to mistakes, thus raising the

possibility that results and conclusions are influenced. It is therefore strongly advised that the option to manually add or edit event markers is only used when the set of recorded event markers is incomplete, or not entirely accurate. This may be a result of the experimenter pressing the incorrect trigger button (and thus logging a false event marker), or pressing the trigger button only after the behavioral or stimulus condition of interest occurred.

3. Adding event markers to fNIRS data using hardware

3.1 Trigger input connector

Each NIRx instrument has a trigger input connector (Fig. 3). This input connector serves to receive and recognize triggers, that are sent according to a LPT port protocol. These triggers will be translated into event markers by Aurora fNIRS or the NIRStar software (depending on which instrument is used). In order for triggers to be successfully recognized, the triggers should have the following requirements:

i) TTL level (0 – 5.0V) with positive edge (low-to-high transitions);
ii) The duration of the pulse (high, 5.0V) should be of at least 10ms;
iii) The timing pulses should be separated by at least 100ms to be distinguished.

To connect hardware generating trigger output, use the accompanying trigger cable that is delivered with every NIRSport, NIRSport 2 and NIRScout instrument. This trigger cable can connect to any device that sends triggers according to a LPT port protocol. In the following paragraphs, several options for devices sending triggers over the LPT port protocol will be briefly discussed. In addition, one exception to devices connecting with the trigger cable is discussed, which is the Trigger remote control (see Section 2.2). For more information on the trigger input connector of NIRx fNIRS instruments, please consult the NIRSport User Guide, the NIRScout User Guide or the NIRSport 2 Hardware Getting Started Guide.

Figure 3a: Trigger input connector for NIRSport (left) and NIRScout Extended (right).

Figure 3b: Trigger input connector for NIRSport 2 (bottom).

3.2 Trigger remote control

The trigger remote control (Fig. 4) has been designed for mobile measurements with the NIRSport.

During these measurements, the NIRSport is connected to a tablet PC, and both are placed in a backpack worn by the subject. The trigger remote control allows the subject to send triggers, for example when they perform certain movements.

The Trigger remote control does not operate using a LPT port or protocol, as it connects directly to a NIRSport device. It does not necessarily have to be used during mobile studies; it can also be used as hardware-based alternative to manually sending triggers from the acquisition software (See section 2.1). This may be useful when both the experimenter and the subject are in a moving environment (e.g. a driving car) and it the push-buttons of the Trigger remote control are more reliable than keyboard keys (i.e. F1, F2, …F8). Note that the Trigger remote control cannot be used with the NIRScout or NIRSport 2.

Figure 4: NIRSport Trigger remote control.

3.3 Parallel port

The most common way to send triggers to the trigger input connecter of a NIRx fNIRS instrument, is to send them over the LPT port protocol, as outputted by a parallel port. Like NIRx fNIRS instruments, many psychophysiological or neuroscience research equipment (e.g. EEG equipment) uses the LPT port protocol to send triggers. The parallel port of a PC is located on the back of the chassis (Fig. 5). The trigger cable of NIRx fNIRS instrument is simply connected to this port. Note that in some instances the Parallel port needs to be set up in Windows Hardware configuration.

Figure 5: Parallel port of a computer, located on the back of the chassis. Note: The Parallel port is not always colored purple.

Sometimes Desktop PCs come without a parallel port. This issue may be solved by purchasing a parallel port card, which can be connected to the motherboard of the PC. Importantly, the user should make sure that drivers are installed correctly, in order for the parallel port to function correctly. Please carefully follow the instructions of the manufacturer of the Parallel Port.

In contrast to Desktop PCs, the vast majority of Laptop PCs or Tablets are not equipped with a parallel port, and here it is impossible to install a parallel port card. An alternative for this is a parallel port express card (Fig. 6, right). The parallel port express card can be plugged into an express card slot, thereby making direct contact to the motherboard of your Laptop PC. This ensures that the sending of triggers is be done in an accurate (timely) manner. Another apparent option is a parallel port USB adapter (Fig. 6, left). Importantly, however, a parallel port USB adapter is not suitable to send experimental triggers. While a parallel port USB adapter may seem easier to set-up, and they are often times less expensive, these adapters do not process the information fast enough in order to accurately send trigger. In case your PC, laptop or tablet does not have a express card slot, or the option to install a parallel port, a dedicated trigger option for sending triggers should be considered, such as the Cedrus StimTracker or Cedrus c-pod (see next 3.5).

Figure 6: Parallel port express card (left; correct) and parallel port USB adapter (right; incorrect).

3.4 Parallel port replicator

In some experimental set-ups, the user wishes to combine several research modalities (e.g. concurrent NIRS-EEG, concurrent NIRS-MRI, etc.). This is often times referred to as multimodal research. In multimodal research it is even more important to adequately send triggers, as it will otherwise become very difficult (if not impossible) to match the several data streams. A dedicated hardware solution offered by NIRx to simultaneously send triggers to several devices is the Active Parallel Port Replicator (Fig. 7). The Parallel Port Replicator allows to split a single Parallel Port Input into up to 4 Parallel Port Outputs. As such, it is an ideal option to simultaneously send triggers to several devices. The Parallel Port Replicator is USB powered and also includes a 1 x Auxiliary BNC input for trigger 0 (Pin 2) input. The rear part comes with LED indicators, that inform the user whether the device is powered/ connected, and which triggers are being sent/ received by the device (up to 8 bits).

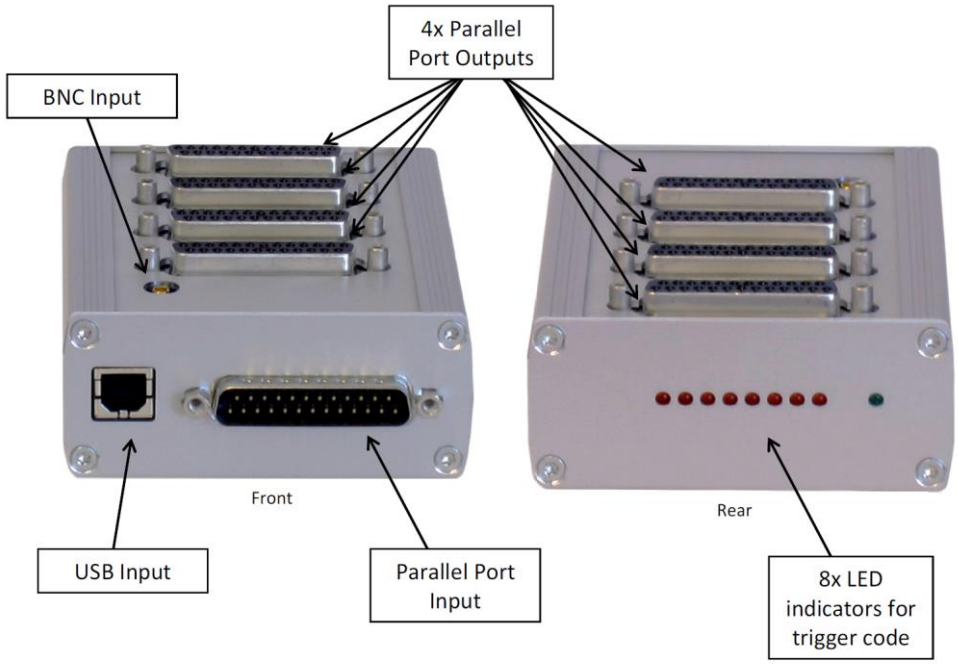

Figure 7: The NIRx Parallel Port Replicator.

3.5 Cedrus c-pod or StimTracker

If the PC, Laptop or Tablet intended to send triggers does not have a free slot for a LPT card on the motherboard, or a express card slot available, it may be good to consider a dedicated solution for sending experimental triggers.

In this case NIRx advices the user to look into stimulus presentation devices from Cedrus. For example, the Cedrus c-pod for NIRx (Fig. 8) has been tested and validated extensively. The c-pod allows stimulus presentation software to send up to 8 bits of event markers over the LPT protocol. It can be easily connected using a USB cable, and instead of a regular parallel port USB adapter (see section 3.3), the c-pod will have no more than 2ms delay, thus promising accurate registration of event markers.

Figure 8: Cedrus c-pod.

In addition to the c-pod, Cedrus also offers a m-pod, or a StimTracker, devices also used to send event markers, but with advanced functionality. Should you be interested, please contact your local distributor or NIRx for more information and purchasing options.

4. Adding event markers to fNIRS data using software

4.1 NIRStar

As written in section 4.2, NIRx acquisition software (NIRStar) allows the user to manually add markers to the data stream using the mouse and the keyboard. These markers can become important to register unexpected events during the measurement (e.g. body movement or any source of distraction) that may generate artifacts to the physiological data. However, for registration of event markers related to conditions (see Section 1.1) manually adding markers in NIRStar is not adviced.

4.2 NIRStim

In addition to the NIRStar acquisition software, NIRStim is a stimulus software also developed by NIRx that allows the user to program presentations to send trigger markers to NIRStar. Although the capabilities of NIRStim can be considered limited to alternative options for stimulus presentation (discussed below), an important capacity of NIRStim is that it can send triggers over a shared variable. With this shared variable, a direct connection between NIRStim and NIRStar can be established – both on a single PC as well as over a Network. This means the NIRStim stimulus software can send experimental markers without the need of additional hardware, such as a parallel port or Cedrus device (see Sections 3.3 and 3.5). As a result, NIRStim distinguishes itself from other stimulus presentation software (discussed in Chapter 5), and may be interesting to consider.

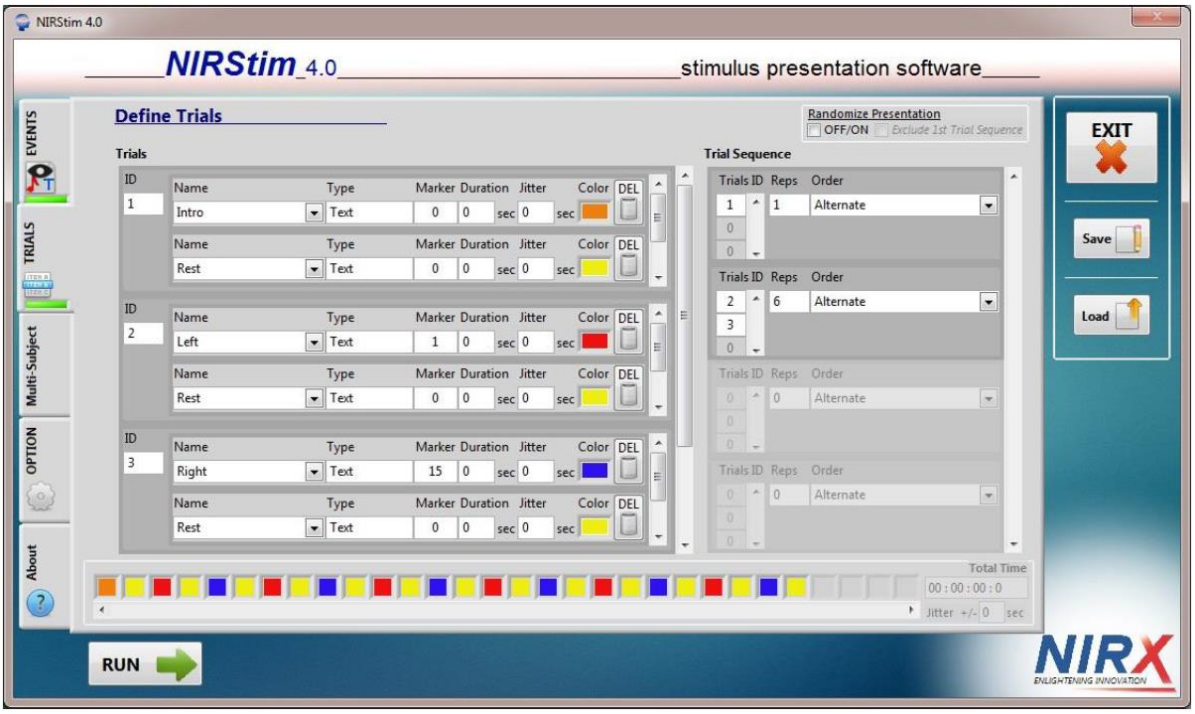

Figure 9: The NIRStim stimulus presentation software.

In addition to sending triggers over the shared variable, NIRStim does also allow to output triggers over a parallel port or Cedrus StimTracker device. For more information on how to set-up an experiment in NIRStim, and send triggers to the NIRStar acquisition software, please consult the NIRStim manual.

4.3 Lab Streaming Layer (LSL)

Lab Streaming Layer (LSL) is a protocol that allows for sharing of both recorded data as well as for experimental triggers in real-time, meaning it can synchronize easily between stimulus PC – fNIRS device. In addition, it allows for integration with data from many other systems and software packages (EEG, Eye-Tracking, Turbo-Satori, PsychoPy, etc.). LSL is based on a transport library (liblsl) that can be called and controlled from several language interfaces (C, C++, Java, C#, Python and Matlab) and under different platforms (Mac, Windows, Unix).

The main advantage of using LSL as means of communication is that it guarantees synchronization of data recorded by different systems. Lab Streaming Layer is available on the Lab Streaming Layer GitHub webpage. Here the user can also find libraries to communicate with several modalities, as well as information on how to use the protocol. With the release of NIRStar 15.2, and Aurora fNIRS, LSL is incorporated in the acquisition software for NIRx fNIRS instruments. The sections below discuss how to handle LSL in NIRStar 15.2 and Aurora fNIRS respectively.

4.3.1 LSL in NIRStar 15.2

The option to receive triggers over LSL, as well as data streaming, can be activated in NIRStar by clicking on “Receive triggers” and “Enable LSL streaming” in the “Data Streaming” window of the Hardware Configuration Menu (Fig.10).

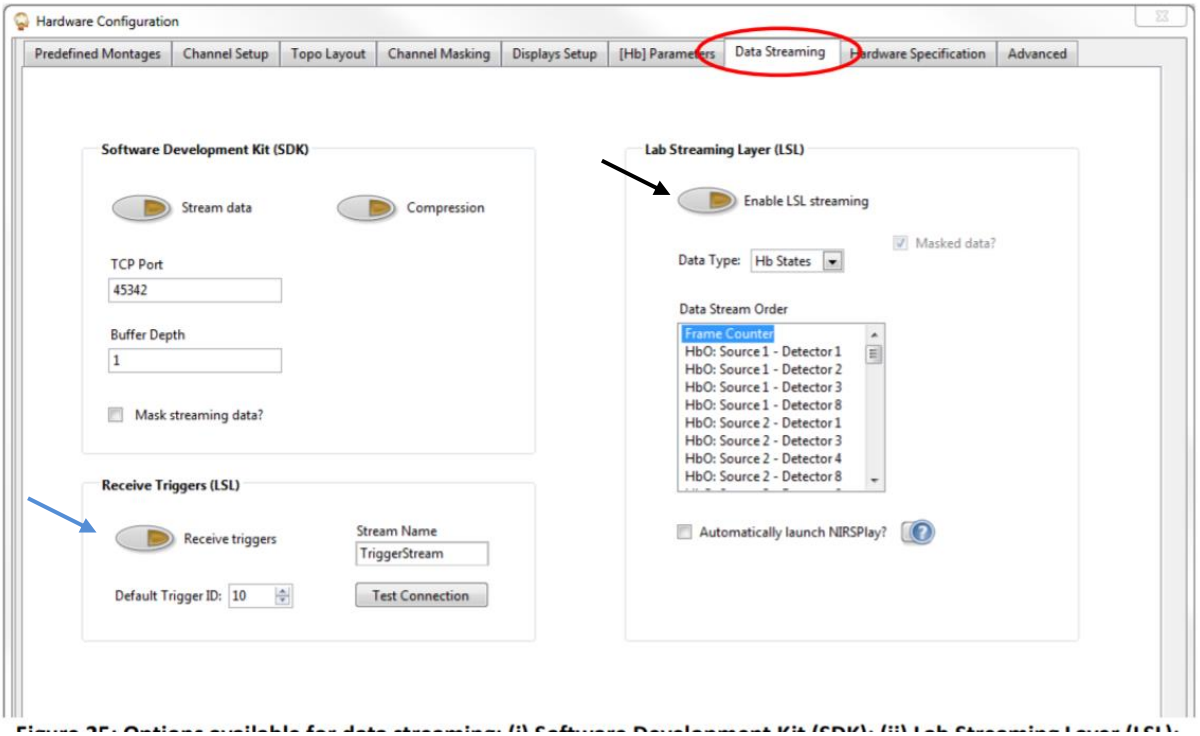

Figure 10: Lab Streaming Layer (LSL) in the NIRStar. The red oval shows the correct, tab, whereas the black and blue arrow indicate how to enable receiving triggers over LSL (blue) and turn on LSL streaming (black).

Once the "Receive triggers" capability is enabled, NIRStar will be able to retrieve triggers from the LSL stream. The trigger information NIRStar expects to receive is an integer corresponding to one of the markers available in the system (e.g. for NIRSport between 1 and 15, for NIRScoutX+ up to 255). If NIRStar receives an unexpected trigger format (e.g. a string), its corresponding time stamp will be stored with the trigger marker defined in this field. By default, this value is set to 10 (Fig. 10). In addition, the default name for the LSL stream where the software will look for triggers is set to "TriggerStream".

Before attempting to run an experiment, please make sure that the stream name is set correctly and that the stream outlet is readily available. This can be done by clicking the "Test Connection" button. It is strongly advised to test the stream connection anytime changes are made to the Hardware Configuration, since otherwise errors may occur when the measurement is started (e.g. "LSL stream cannot be found"). By clicking the "Test Connection" option, NIRStar will check to see whether the defined LSL stream is available (i.e. "TriggerStream" by default) and inform the user of the result. In case of success, the user may confirm the hardware settings and proceed with recording data. If the stream cannot be found, please double check the stream name and make sure that the stream is being outputted by the software/ hardware outputting the trigger information.

In addition to receiving triggers, NIRStar can also send triggers that are received manually or over other hardware/software options over the LSL protocol. This can be done by activating the "Enable LSL Streaming" option (Fig. 10). Once enabled, event marker information will be automatically streamed as triggers to LSL. This allows the user to conduct multi-modal research, without the need of a hardware option such as the Parallel Port Replicator (see Section 3.4). For more information on LSL functionality in NIRStar, please consult the NIRStar User Manual.

4.3.2 LSL in Aurora fNIRS

The option to receive triggers over LSL, as well as data streaming, can be activated in Aurora fNIRS for each configuration separately. To do so, choose to Edit a configuration and click on the "LSL stream names" window (Fig. 11). Here the stream names for data streaming ("Data out", default name: "Aurora") and receiving triggers ("Trigger in", default name: "Trigger") can be set. By clicking the "Test Connection" option, Aurora fNIRS will check to see whether the defined LSL stream is available.

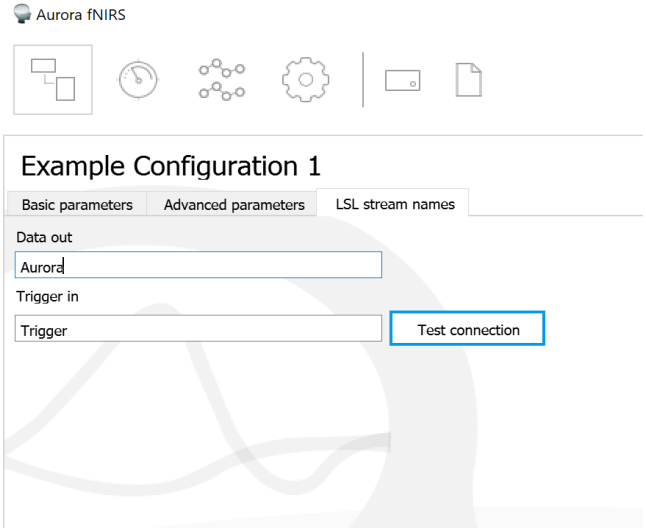

Figure 11: Lab Streaming Layer (LSL) in the Aurora fNIRS.

4.4 The Trigger.nrx file

Finally, the NIRStar has the capability to add event markers to the data stream based on triggers that are saved in a text file. If the user creates a file "Trigger.nrx" under "C:\NIRx\Commands", NIRStar will periodically check if this file contains new trigger markers. This feature allows for remote triggering that can be sent via a local network given that access to the acquisition computer is provided. An extension of this capability is that triggers can be sent from within Matlab, without the requirement of additional parallel or serial cables. An example of a Matlab function prototype along with its application to sent trigger marker '1' can be seen in Figure 12 below.

```
%% Send Trigger Marker programatically
marker = 1; %select trigger marker
if isnumeric(marker)
    marker = max(min(marker,15),1);
    Trigger_Control(marker);
end
```

```
function Trigger_Control(marker)

pathname = 'C:\NIRx\Commands\';
filename = 'Trigger.nrx';

if exist(pathname,'dir') ~= 7
    mkdir(pathname);
end

fid = fopen([pathname filename],'at');
fprintf(fid,'%d\n',marker);
fclose(fid);

end
```

Figure 12: Example of Matlab code and function prototype to send trigger programmatically to NIRStar.

5. How to send triggers

In case the user wishes to send triggers over a LPT port (using additional hardware), or over Lab Streaming Layer (LSL), an additional software package that sends the triggers is required. This is typically done through presentation software, which allows to synchronize the presentation of stimuli or executed task conditions with trigger markers. Below several software options for stimulus presentation and the sending of triggers are briefly discussed.

5.1 NIRStim

NIRStim is a stimulus software also developed by NIRx that allows the user to program presentations to send trigger markers to NIRStar (see Section 4.2). NIRStim allows to send triggers to NIRStar via a shared variable, or over a parallel port (LPT) or StimTracker device. NIRStim does not work with LSL.

5.2 PsychoPy

A common and rapidly becoming more popular choice to build and/or code experiments is PsychoPy. PsychoPy is written in the Python language and available freely in open source format here: http://psychopy.org/. The software allows users to build an experiment in a PowerPoint-like 'Builder' application (Fig. 13). Experiments created in Builder can then be converted into PsychoPy code, which can then be edited if desired. Alternatively, the user can write the experiment entirely in script format (from scratch). Importantly, it is however not possible to convert coded experiments in Builder view.

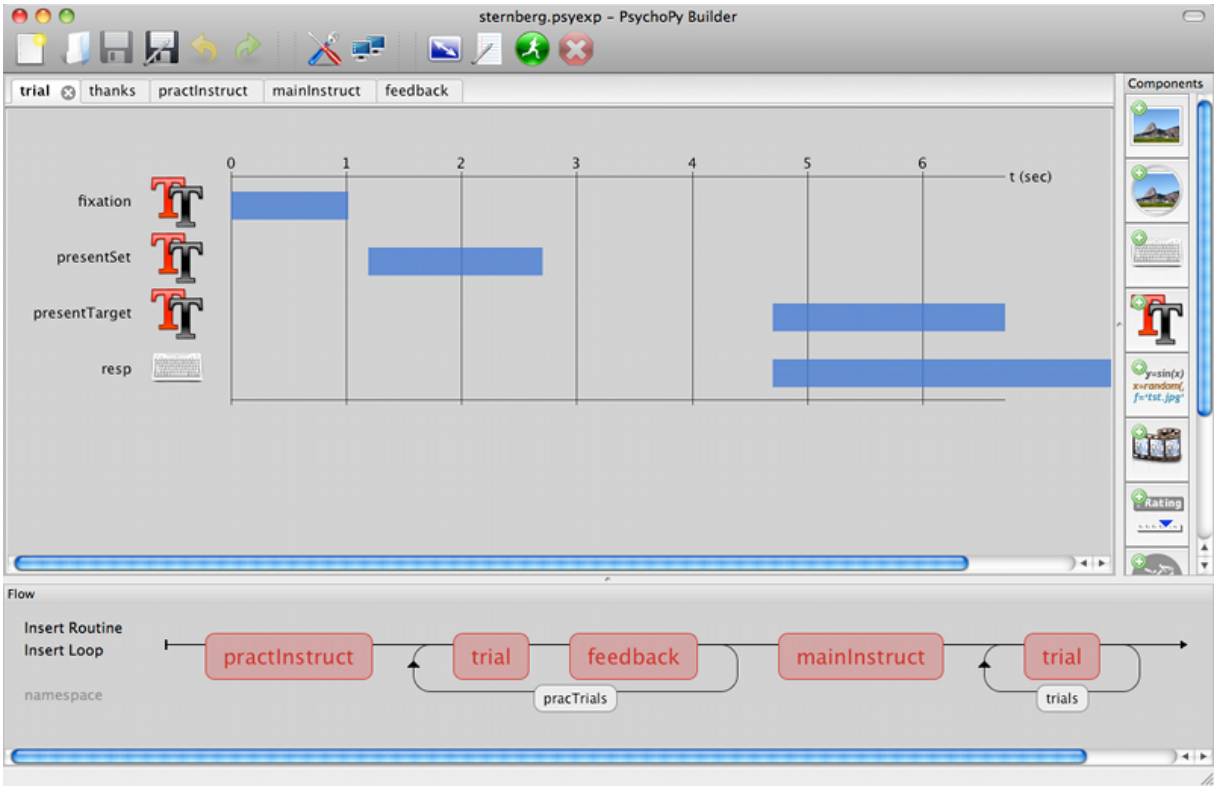

Figure 13: Example view of the PsychoPy Builder window.

The PsychoPy website is a great resource for educational material on how to build experiments (both in the Builder mode as well as in the Coder mode). For example, the Documentation section should prove to be of help when the user wishes to start creating experiments in PsychoPy. This section can be found here: http://www.psychopy.org/documentation.html. To assist the user, below references and explanations can be found on how to send triggers in PsychoPy, both over a parallel port, c-pod and through LSL.

5.2.1 Sending triggers in PsychoPy over a parallel port.

In case a parallel port is used to send triggers through PsychoPy, the user can configure this both in the Builder mode as well as in the Coder mode. In the Builder mode, the Parallel Port Out Component would have to be chosen, and added to the routine in order to send triggers. More information can be found here: http://www.psychopy.org/builder/components/parallelout.html. Alternatively, the user can switch to the Coder view after having built an experiment in PsychoPy, and then proceed to manually add code to sent triggers. More information on the various functions is available here: http://www.psychopy.org/api/parallel.html. In addition, a good way to get started is to load the "parallelPortOutput/py" demo experiment (Fig. 14).

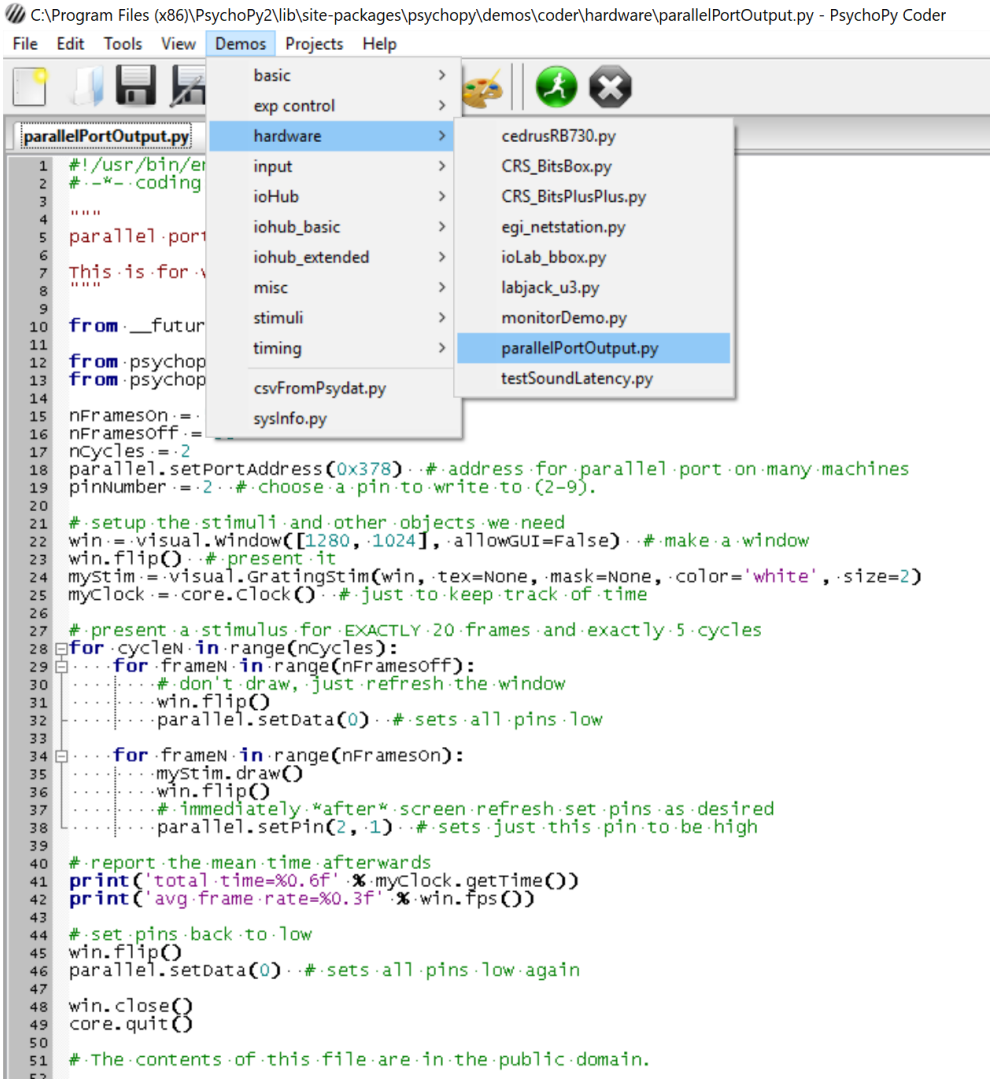

Figure 14: The PsychoPy "parallelPortOutput.py" demo experiment (in Coder view).

5.2.2 Sending triggers in PsychoPy using the Cedrus c-pod

A convenient way of sending trigger is the Cedrus c-pod (see Section 3.5). While the c-pod can work with the PsychoPy software, the user needs to take some steps in order to get this to work. Provided PsychoPy is already installed and functioning correctly, the user should proceed to download a Python library from the Cedrus website. Specifically, this concerns the pyxid_impl.py file, which can be downloaded here: https://github.com/cedrus-opensource/pyxid. Once downloaded, this file should replace the default "pyxid_impl.py" file, which can be found under "PsychoPy3\Lib\site-packages\pyxid".

Once the pyxid_impl.py file has been replaced, the following code should be added to your experiment:

```
import pyxid

import time

devices = pyxid.get_xid_devices()

assert len(devices) > 0

d = devices[0]

d.init_device()

d.set_pulse_duration(1000)


# one sweep through all lines

#for i in range(1,9):

#    d.activate_line(lines=[i])

#    time.sleep(1.2)
```

This code sets up the c-pod as a device to send triggers over. Ideally this code is added to the start of your experiment, so it is immediately loaded. Once this code has been added, triggers can be send relatively easily with the following piece of code: **`d.activate_line(lines=[5])`**. In this example, trigger number 5 (event marker 5) is sent. NIRx also offers an example builder and coder experiment, which can be downloaded from our Support website here:

5.2.3 Sending triggers in PsychoPy using LSL

In addition to sending triggers over hardware (i.e. the parallel port and Cedrus c-pod), PsychoPy is also able to send triggers over the LSL protocol (see Section 4.3). The main advantage of using LSL together with PsychoPy is that this solution for stimulus presentation is without any added costs for stimulus presentation or hard used to send triggers. Since this method also allows to send triggers wirelessly, it is a preferred way for mobile studies. However, since PsychoPy is relatively new in incorporating LSL, at present, sending triggers in PsychoPy requires the user to manually install the Python interface for LSL, and manually add code components to a builder experiment or coder experiment. In the following subsections, both steps are explained in detail.

5.2.3.1 Installing the Python interface for LSL

While the PsychoPy software is rapidly developing and will likely include the Python interface for LSL by default in the future, currently the user needs to manually install this interface. To install this interface to an existing install of PsychoPy, please follow all steps below.

1. Download the Python interface for LSL here: https://pypi.org/project/pylsl/#files.
2. Make a note of the location of where the interface file is downloaded. By default, the download folder is 'Downloads', meaning the location would be C:\Users\NIRx\Downloads\ pylsl-1.12.2-py2.py3-none-any.whl.
3. Open the Windows command prompt as Administrator by typing 'cmd' in the search bar and pressing ctrl–shift + enter to launch Windows Command Prompt as Administrator. Please choose 'Yes' if you receive a message from Windows asking you whether Windows Command is allowed to make changes to your device.
4. In the Command Prompt window, navigate to the installation folder of PsychoPy. By default, this folder is C:\Program Files (x86)\PsychoPy3. Navigating to this folder can be done by entering the following command:

   ```
   cd C:\Program Files (x86)\PsychoPy3
   ```

5. Set the installation path to the PsychoPy folder with the following command:

   ```
   set PATH ="C:\Program Files (x86)\PsychoPy3"; %PATH%
   ```

6. Install the Python interface for LSL by running the following command:

```
Python -m pip install C:\Users\NIRx\Downloads\pylsl-1.12.2-py2.py3-none-any.whl
```

7. You will receive a message that the Python interface for LSL is successfully installed (Fig. 14).

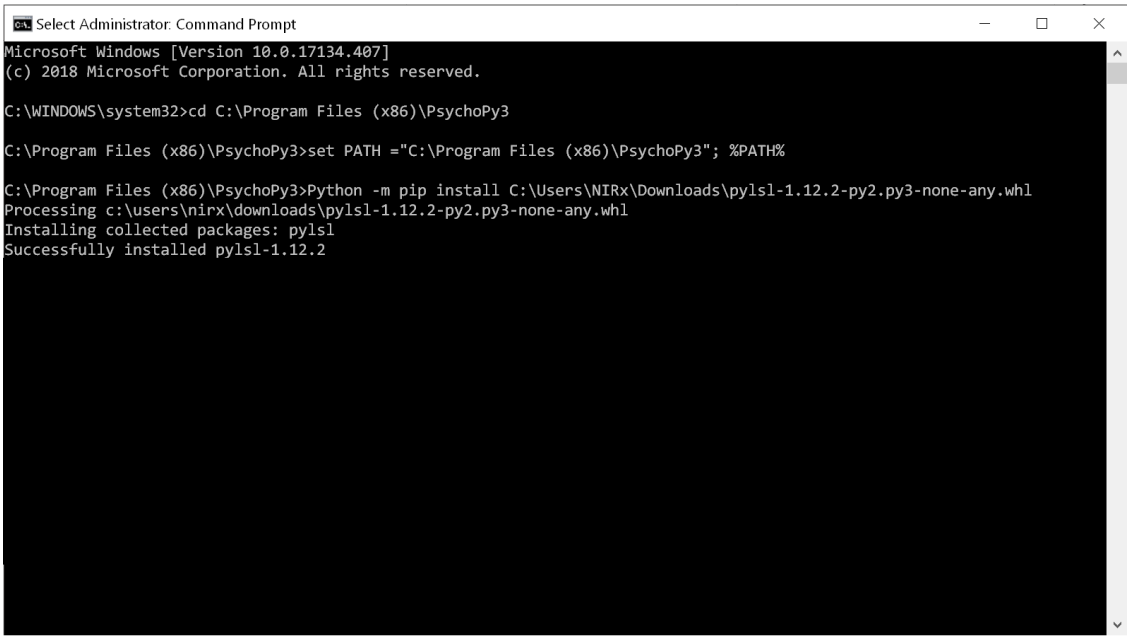

Figure 14: Installing the Python interface for LSL using Windows Command Prompt as Administrator.

5.2.3.2 Adding LSL code components to a builder experiment

Once the Python interface for LSL is installed, LSL code components can be added to experiments created either in PsychoPy builder or PsychoPy coder. For the moment, the present guide only covers how to add the code component to experiments in builder, given that the code is equal for experiments created in PsychoPy coder and under the assumption that users using PsychoPy coder to create experiments are familiar enough with the Python language to add this code themselves.

To start adding LSL code components to a builder experiment, please load a PsychoPy builder Demo experiment. To do so, navigate to 'Demos' in the menu bar of the PsychoPy builder view and load the 'stroop' demo; stroop.psyexp (Fig. 15).

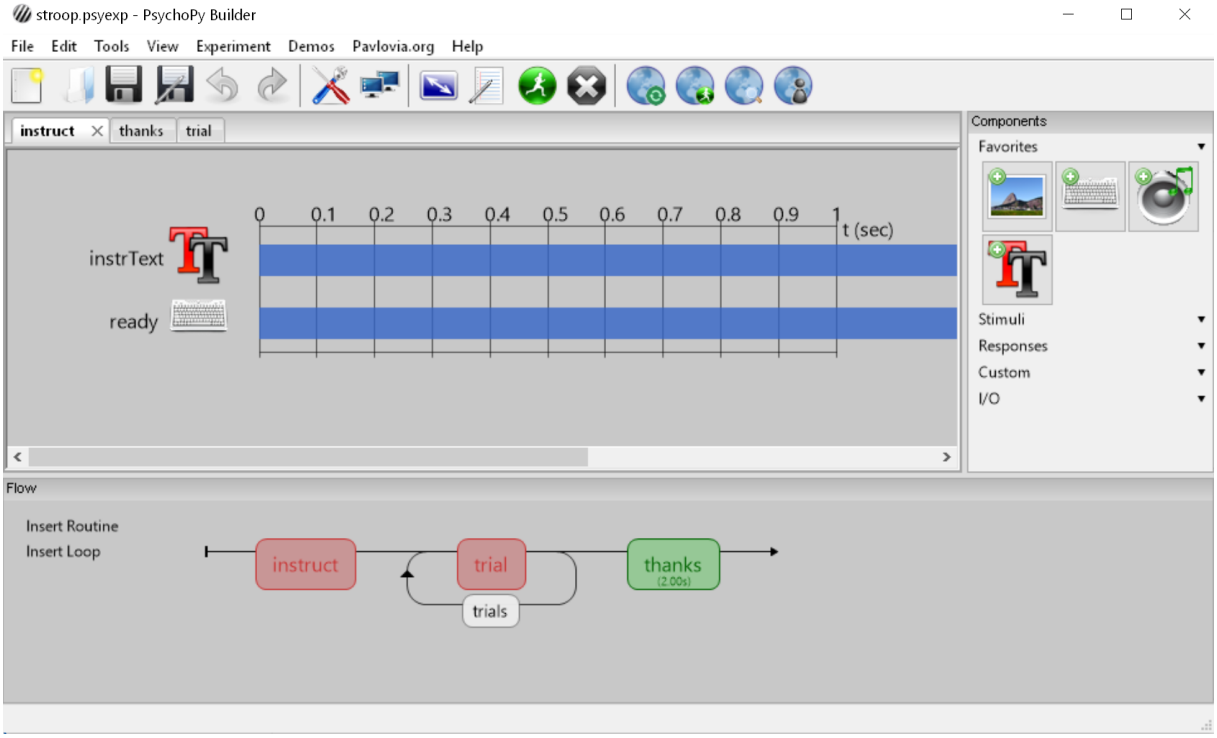

Figure 15: The PsychoPy stroop demo experiment.

Before adding the LSL code components to this demo experiment, the user is advised to run the experiment once or twice to make sure everything is working correctly. For detailed information on the 'stroop' experiment in PsychoPy, please consult the PsychoPy online documentation which is available here: http://www.psychopy.org/builder/concepts.html. In addition, there is a tutorial video which explains the basics of PsychoPy builder as well as how to build the Stroop task from scratch in PsychoPy: https://www.youtube.com/watch?v=VV6qhuQgsiI.

Once the basic concept of the 'stroop' demo experiment and PsychoPy builder are clear, the user can proceed to add the LSL code component to the experiment. This consists of two main steps: (i) adding code so that PsychoPy loads the Python interface for LSL at the start of the experiment and sets the variables, and (ii) sending a trigger during every trial.

In order to run code from (almost) the beginning of the experiment, navigate to the 'instruct' tab (active in Fig. 15). Next, from the 'Components' menu on the right, choose 'Custom' and click on the 'Code' button (Fig. 16). This will cause a new pop-up window to open in which the user can enter the code which should be run at a chosen moment of the experiment ('Begin Experiment', 'Begin Routine', 'Each Frame', etc.). Select the 'Begin Experiment' tab and copy the following code below:

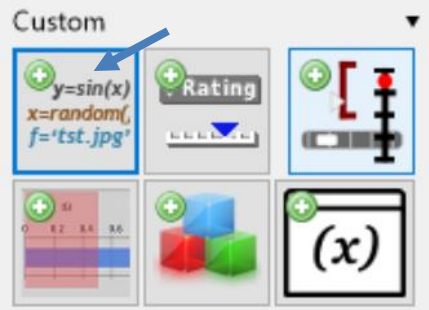

Figure 16: The 'Code' button.

```
from pylsl import StreamInfo, StreamOutlet # import required classes

info = StreamInfo(name='Triggerstream', type='Markers', channel_count=1,
channel_format='int32', source_id='Example') # sets variables for object
info

outlet = StreamOutlet(info) # initialize stream.
```

The first line of code imports the classes 'StreamInfo' and 'StreamOutlet'. The user is not required to make any changes to this line. The second line of code defines the object 'info', which (in the third line) will be used to provide the stream information to the outlet of the stream. The info object is generated by using the 'StreamInfo' function, and will store a declaration of the data stream. Typically, it represents the following information:

- Stream data format (#channels, channel format)
- Core information (stream name, content type, sampling rate)
- Optional meta-data about the stream content (channel labels, measurement units, etc.)

Note that in this case the 'info' object is set-up so that the stream name is set-up as 'TriggerStream', the default name for NIRStar 15.2 (See section 4.3.1). For Aurora, the stream name should be changed to 'Trigger' (See section 4.3.2). If different names are specified in either NIRStar 15.2 or Aurora, the user should specify these names in the code component. The other information can stay as is:

- 'type' sets the content type of the stream, in this case (event) markers), which is the preferred way to find streams on LSL.
- 'channel_count' sets the number of channels per sample, this can be left to the default of '1' as only 1 trigger will be sent per sample (timepoint).
- 'channel_format' sets the format/type of each channel, cf_int32 is the value format that is preferred for application event codes.
- 'source_id' sets a unique identifier of the stream's source, an optional piece of information that, if available, allows the recording program to re-acquire a stream automatically.

Detailed information can be found in the pylsl.py documentation, available here:
https://github.com/sccn/labstreaminglayer/blob/master/LSL/liblsl-Python/pylsl/pylsl.py

Once the above code has been added to the 'Begin Experiment' tab, click 'Ok' and navigate to the 'trial' tab. Here only a single line of code needs to be added, in the 'Begin Routine' tab. This will send an event marker at the start of each trial (Fig. 17).

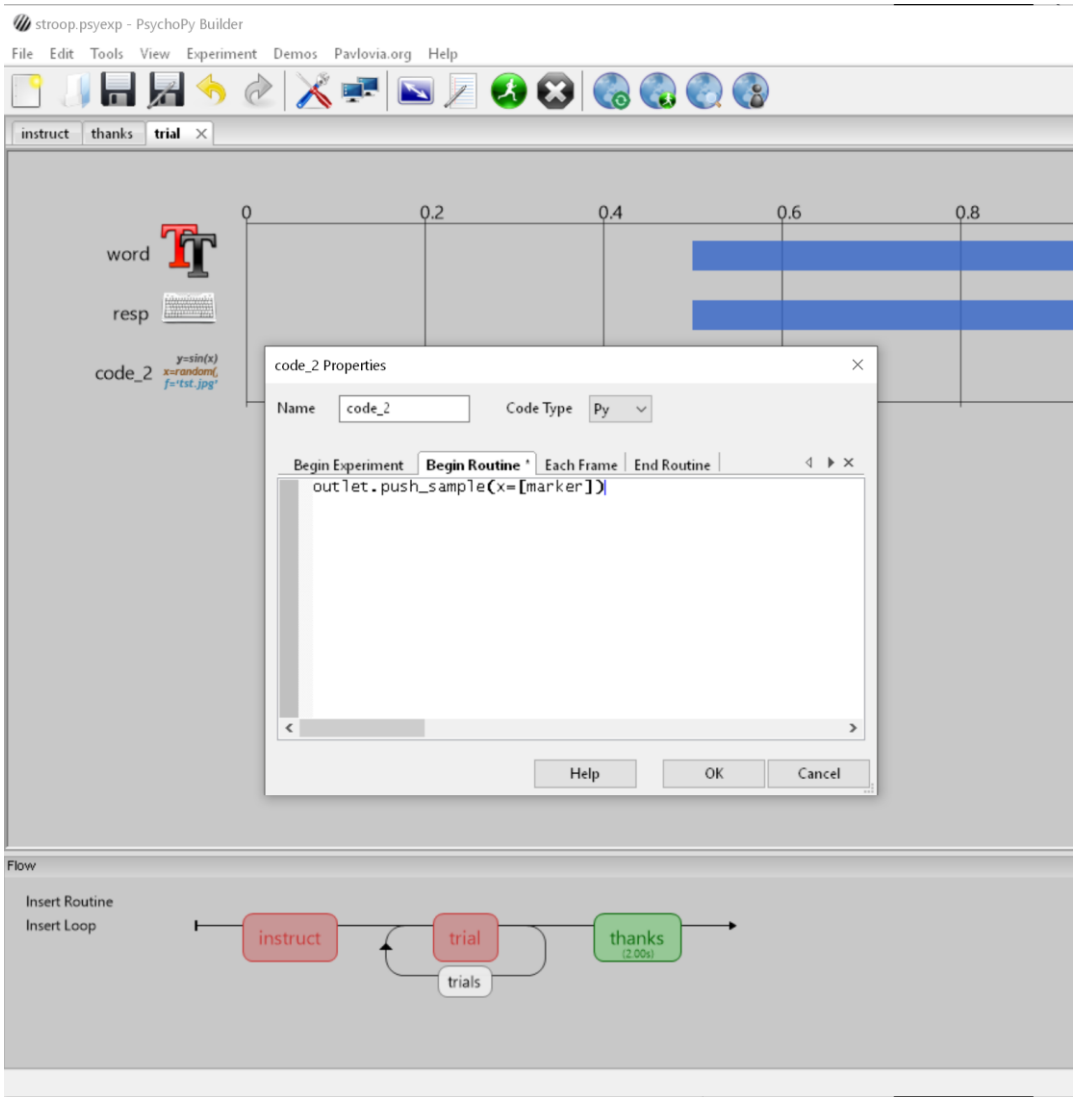

Figure 17: Adding relevant code to the 'Begin Routine' tab to send a marker at the start of each trial.

The code that should be added can be copied from below:

```
outlet.push_sample(x=[marker])
```

This code sends a marker x; which in this case is set to the column 'marker' in the 'trialTypes.csv' spreadsheet. This spreadsheet can be found in the folder where the 'stroop' demo experiment is stored (Fig. 18). For more information on this file, please see here: http://www.psychopy.org/builder/flow.html?highlight=trialtypes

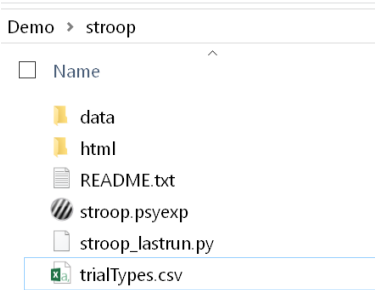

Figure 18: 'TrialTypes.csv'

The 'trialTypes.csv' spreadsheet can be opened in several applications, for example in Excel or Notepad. When viewing the data in the spreadsheet the user should see a table like this:

text	letterColor	corrAns	congruent
red	red	left	1
red	green	down	0
green	green	down	1
green	blue	right	0
blue	blue	right	1
blue	red	left	0

The 'trialTypes.csv' spreadsheet contains all individual trials which will be randomized (and repeated) during the experiment. In order to add event markers to these trials, we should add fifth column, which contains the event marker numbers:

text	letterColor	corrAns	congruent	marker
red	red	left	1	1
red	green	down	0	2
green	green	down	1	1
green	blue	right	0	2
blue	blue	right	1	1
blue	red	left	0	1

In the example above, marker '1' is sent for each congruent trial (the text and letterColor are identical, e.g. green), and marker '2' is sent for each trial that is incongruent (the text and letterColor are not identical, e.g. blue). Once this column is added, please save the file and return to PsychoPy builder.

After editing the 'trialTypes.csv file', the 'stroop' demo experiment should be set-up so that it sends triggers to NIRStar 15.2 or Aurora. To check this, please launch either acquisition software, and go to the LSL configuration section (see Section 4.3.1 or 4.3.2).

In a separate window, or on a separate computer connected to the same network, run the Experiment in PsychoPy, either by clicking the 'Run' button or by pressing Ctrl + R on the keyboard. The experiment will run, and after a dialog window appears briefly the instruction window is presented on the screen (Fig. 19).

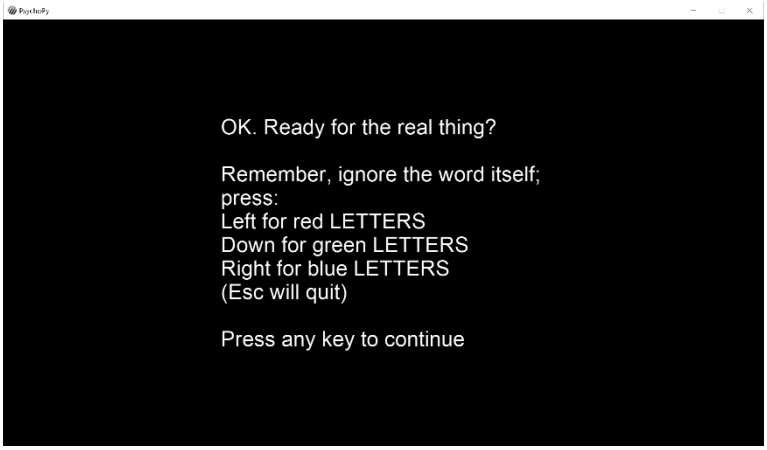

Figure 19: Instruction window for the 'stroop' demo experiment.

Once the instruction window is presented on the screen, please switch to either the NIRStar 15.2 or Aurora Software, and click the 'Test Connection' button in the LSL configuration section. If the connection is not successful, do not start the experiment as triggers will not be sent. Please check the name of the name of the stream ('TriggerStream' in this example) and note that this name is Case Sensitive. In case the connection is successfully established a dialog, window will pop-up (Fig. 20), and you can proceed to start the experiment in PsychoPy.

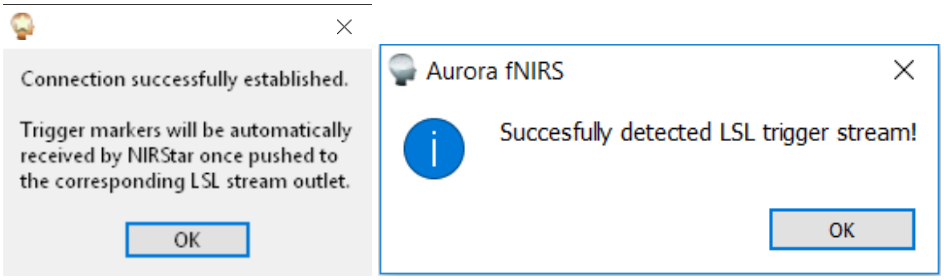

Figure 19: Dialog window for a successful LSL Trigger connection in NIRStar 15.2 (left) and Aurora (right)

Note: An alternative possibility is to convert a PsychoPy experiment created in Builder mode to a code version that can be edited in Coder mode, and add the LSL code components here. However, the user should be aware that afterwards the experiment can not be converted back into an experiment that can be viewed or edited in Builder mode. We therefore strongly advice users intending to use the Builder mode to completely finish their experiment, before manually adding the necessary code to send triggers. This applies both for sending triggers over hardware (i.e. parallel port or c-pod) as well as through LSL.

5.3 Alternative options for Stimulus presentation

Besides NIRStim and PsychoPy, there are several other software options that allow to synchronize the presentation of stimuli or executed task conditions with trigger markers. These options include (but are not limited to) E-prime, SuperLab and Presentation. Which software suits the user best is dependent on the various options they offer as well as experience. Importantly, these are all commercial alternatives. NIRx therefore kindly asks the user to contact the support team of the relevant stimulus software, for any questions pertaining to sending triggers.