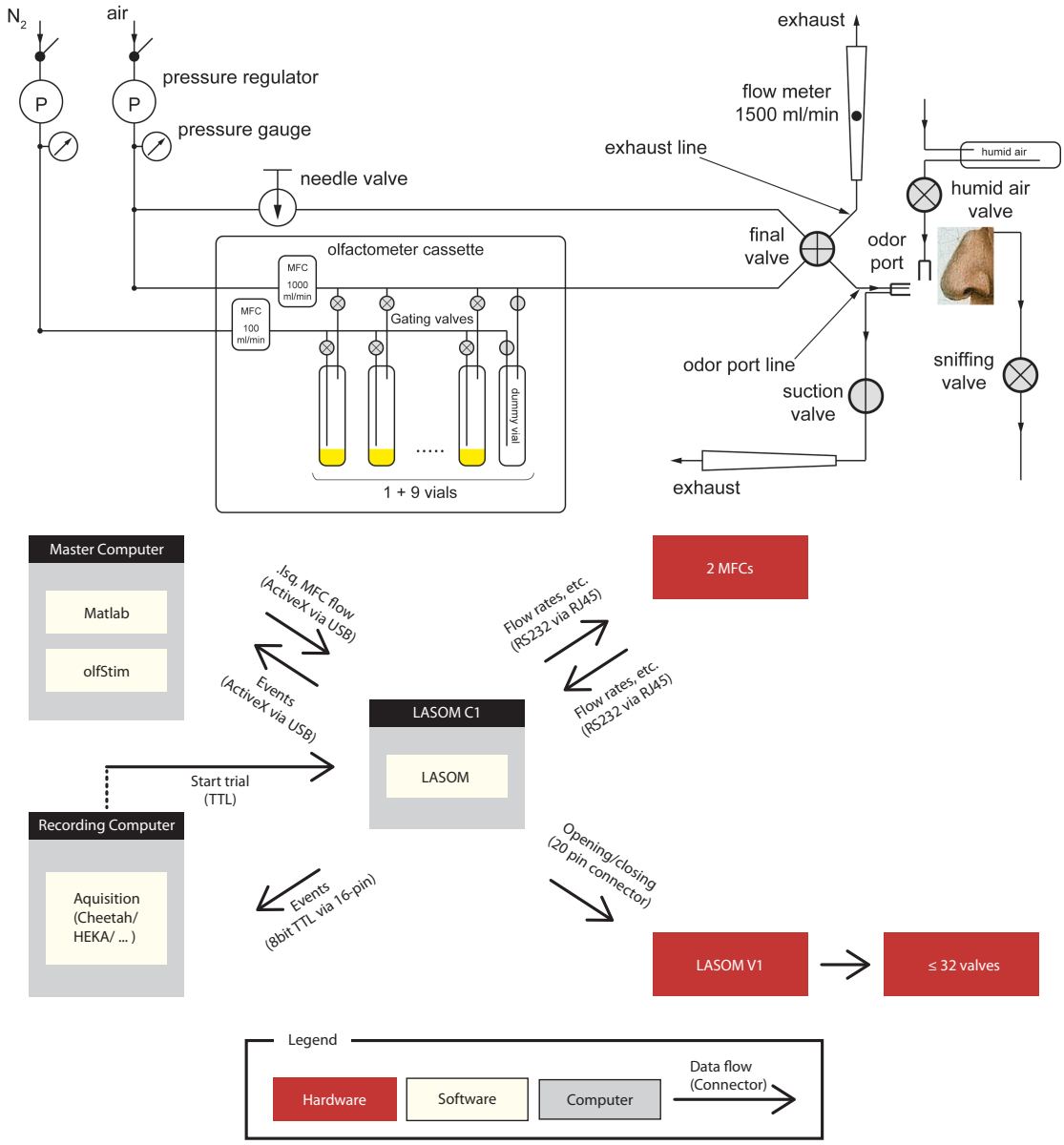
olfStim

Table of Contents



Set air and Nitrogen pressure to 2 bar.

List of hardware

Explain what this “LASOM” actually is.
LASOM_C1
LASOM_V1

Order List

Name	Part number	Vendor	How many	Unit price
Gas filter column	10270412	Fisher Scientific	2	245,70 €
Column clamp for 250/300 ml flask	10156272	Fisher Scientific	2	37,80 €

Valves, MFCs

Name	Part number	Vendor	How many	Unit price
Valve manifold (5NC valves)	161T102HH2	NResearch	2 or 4	323$
Valve manifold (4NC valves, 1NO valve)	161T102HH	NResearch	2	323$
MFC (1.5l/min Air, 2 bar, 20ºC N.C. Control Valve)	IQFD-200C-AAD-00-V-A	Bronkhorst	1	
MFC (100 ml/min N2, 2bar, 20ºC N.C. Control Valve)	IQFD-200C-AAD-00-V-A	Bronkhorst	1	

Setting up the hardware

Connecting the valves

LASOM makes assumptions about which valves are connected to which connectors on the LASOM2_V1 board (the board with the connectors for the valves). Therefore always connect the valves gating the dummy vial (they're normally open) to V1 and V2. From V3 to V20 connect the other odor vial gating valves. Connect the gating valves from the N2 stream to the even-numbered connectors, and the gating valves from the air stream to the odd-numbered connectors. The two valves gating one vial have to be connected to two consecutive connectors.
Example: the valve gating the air stream for the first vial has to be connected to V3, and the valve gating the N2 stream for the first vial, has to be connected to V4. For the second vial, the gating valve for the air stream goes into V5 and the N2 equivalent into V6.

The other valves (final valve or custom valves you want to control) go into valve connectors V21 to V32. You have to define the mapping of valves handled by the olfStim software to the connectors on the LASOM2_V1 board. For the default valves in olfStim the mapping is as follows:

Valve	Connector
Final valve	V31
Sniffing valve	V32
Suction valve	V28
Humidity valve	V27

These definitions are just part of the sequencer building blocks for every action.

For powering the final valve looks something like this:

Gate, slaveIndex, 31;

If you want to change the mapping for the default valves just change the numbers in the parameter definition. 31 in this example means that finalValve is connected to connector 31.
If you want to add new valves and control them through olfStim, read on in the section for developers.

Connecting the Mass Flow Controllers

There are two mass flow controllers, one for regulating the flow of the air stream, the other one for the N2 stream. olfStim currently only supports mass flow controllers (MFC), which can be directly controlled by LASOM control board. These currently include Bronkhorst-type mass flow controllers (MFC), using the digital RS232 interfaces. Tested were the Bronkhorst MFCs IQ-Flow. See parts list for details.

There are two RS232 connectors on the LASOM_C1 board: J21 and J22. The MFC #1 goes into J21 the MFC #2 goes into J22. OlfStim expects MFC #1 to be the air regulating MFC, and MFC #2 to be the N2 regulating MFC.

MFC #	MFC type	Connector
1	Air	J21
2	N2	J22

Olfactometer - computer connection

Get the latest LASOM package, which includes the driver by connecting to RPMetrix' ftp server (using an ftp client like FileZilla):

ftp.rpmetrix.com
user: support4rpm
password: U32x44rawvb9

Download the LASOM2 zip file with the latest timestamp. Unzip the folder. Open the command line, cd to the "RP Metrix" folder in the downloaded LASOM2 folder and execute the following statement:

regsvr32 LASOMX.ocx

You should receive a success pop-up message.

Note: You should also download a LASOM1 folder from the RPMetrix ftp server as the documentation for the sequencer codes are only contained in this package.

Required testing

Before starting to use the olfactometer with olfStim some initial testing has to be done to make sure, everything works as expected. For this use the LASOM control software “LoadBoard” which can be downloaded from the RPMetrix ftp server.

Go to:

Understanding the sequencer code

The microprocessor of the LASOM_C1 board understands a certain set of commands. For a description of this sequencer language see the “LASOMSequencerOperation.pdf” in the Documentation folder.

Here is a further description of some commands used by olfStim:

Odor,slaveIndex,odorValveIndex	Command will open the two valves gating the vial defined by odorValveIndex and will automatically close the two valves gating the dummy vial in the slave defined by slaveIndex. ATTENTION: As the two valves gating the vial #2 are connected to V3 and V4, the odorValveIndex of the first vial is 2. The reason is because the dummy valves are connected to connectors V1 and V2. Therefore all valve indices are shifted by 1.
Gate,slaveIndex,valveIndex	Command will open the valve connected to the connector defined by valveIndex on the slave defined by slaveIndex. Gate,1,3 will open the valve connected to connect V3 on slave 1.
StartTimer, 2	Starts an internal Timer at timer index 2 (timer indices 2-7 are generally usable, timer indices 0 and 1 are reserved for loops.
WhileAlways SinceTimer, 2, 3 CheckLapse, 3, someTime IfLapse, 3, @lapsedOut	Enter an infinite while loop, for every iteration, compute the relative time since the timer was started for timer index 2, write that relative time into timer index 3. Check whether the time in index 3 exceeds the time in ms defined in someTime, store the result. IfLapse checks whether timer index 3 did exceed the defined time value, if so the sequencer jumps out of

EndWhile	the while loop to label @lapsedOut.

The commands have to be provided by the master computer running olfStim to the LASOM_C1 board in the form of sequencer files. Sequencer files, also called lsq files, are txt files with the .lsq extension. They typically include the setting of parameters and action commands to execute.

Sequencer Files in olfStim

One of olfStim's core functions is buildTrialLsq.m. It uses the trial information present in smell to construct a sequencer file, which can then be sent to the LASOM_C1 for execution.

There are two important keywords in sequencer files (also lsq files) which buildTrial.m uses for parsing the text.

`eof`

This marks the end of the sequencer code and is contained at the end of the core.lsq file. buildTrialLsq.m adds the sequencer building blocks at the location of 'eof'.

`bof`

bof marks the step in the sequencer code where time is set to zero. This is where the clock of the sequencer starts to count. Relative to this clock all actions are triggered. The sequencer file sent to LASOM (trial.lsq) always contains some parameter definitions at the beginning, and depending on the use sometimes for instance an endless loop which checks for a

For users

Installation

Download or checkout olfStim from the github repository:
https://github.com/lorenzpammer/olfStim

Add the olfStim folder to the Matlab path. Only select the parent folder, as all necessary subfolders will be added to the path once olfStim is being called.

Necessary customization

It was a design goal to make olfStim work out of the box, but there are several customization options, which allow users to adapt olfStim for their particular needs without having to alter code.

Odorant library

All odorants you want to present have to be entered into the odorLibrary, which is just a text file with definitions for each odorant. The purpose is, that all information, which characterizes an odorant is entered once for each odor in use.
You can find the file under the following path: olfStim/odorLibrary/odorLibraryGenerator.m

There are a number of default odorants in the odor library. When you start using olfStim yourself you will either have to update them, for instance because you use odor X with a different purity, or just delete all entries and start from scratch.

Let's say we want to add Cineole to our library of odorants. We'll first add the odor name to the list of odor names:

```
odorNames = {'Benzyl alcohol','Butanone','Camphor','Citral', ...
    'Eugenol','Geranyl acetate','Hexanal','1-Hexanol','2-
Hexanone','Isoamyl acetate', 'Vanillin','Turtle food','Paraffin
oil','Mineral oil','Ethanol', 'Water','Empty', 'Cineole'}; % Cell
array of strings with names of odorants in Library
```

Then we'll add an entry to the list of odors. To do this just copy the template, which you can find at the top of paragraph C in the odorLibraryGenerator.m file, paste it to the end of the file (above the last 'end') and update the input to each field for the odorant you are using.

```
odor = 'Cineole';
for i = 1 : length(odorLibrary)
    if strcmp(odorLibrary(i).odorName, odor) % at the respective odor
write the following properties to the structure:
        odorLibrary(i).iupacName = '1,3,3-trimethyl-2-
oxabicyclo[2,2,2]octane'; % the name of the molecule following IUPAC
convention as a string
        odorLibrary(i).CASNumber = '470-82-6'; % the CAS registry
number identifying the molecule as a string
        odorLibrary(i).producingCompany = 'Sigma'; % the name of the
company that produced the odorant: 'Sigma', 'Roth', etc.
        odorLibrary(i).odorantPurity = 0.99; % purity of the molecule
given as a fraction, usually above 0.95: 0-1
        odorLibrary(i).state = 'liquid'; % string describing whether
the state of the molecule is liquid or solid: 'liquid' or 'solid'
        odorLibrary(i).odorantDilution = 0.1; % volume fraction
(v(odorant)/v(dilutive solution)) of odorant in the vial after
diluting it with water or oil: 0-1
        odorLibrary(i).dilutedIn = 'Paraffin oil'; % in which
solution the odor was diluted: 'Water' 'Paraffin oil' 'Mineral oil'
or [] if no dilution
```

```
        odorLibrary(i).concentrationAtPresentation    =    0.005;    %
concentration as volume fraction (v/v) of saturated headspace, which
is presented to the animal: 0-1
        odorLibrary(i).inflectionPointResponseCurve    =    [];    %
concentration (v/v, of saturated head space) of presented odorant at
which the response curve as measured in olfactory nerve has its
inflection point.
        odorLibrary(i).vaporPressure = [253.3 25]; % vapor pressure
in Pascal @ 25?C. For mixtures use Raoult's law. To convert mmHg into
Pascal multiply by 133.322
    end
end
```

Configuring valves

Regarding valves several separate properties can be configured.

1. Valve actions
2. Valve connector index
3. Action properties

Valve actions

Actions are loosely defined. Basically an action is just a block of sequencer code, which will then be used by buildTrialLsq.m to construct the sequencer file for the current trial. This can be very simple such as opening or closing one specific valve, but it can just as well be a complex sequence of commands for the sequencer. If you want to add a new valve action create a text file in the olfStim/lsq directory. Change the filename (no spaces!) to the name of the new action and add the .lsq extension. The file must contain the sequencer code for that action.

Valve connector index

Valves are connected to the connectors of the LASOM2_V1 board. If you add a new valve you have to tell the sequencer to which connector the new valve is connected. This information has to be provided in the block of sequencer code for the specific valve action (see the above paragraph).

Say you want to add a new action, which opens the 'xyz' valve connected to the connector V30 on the first slave LASOM2_V1 board. You have to add a text file with a name like openXyz.lsq to the olfStim/lsq/ directory. This text file should contain something like:

```
Gate, 1, 31; Opens the xyz valve
```

The Gate command tells the sequencer to power the valve connected to the valve connector #31 (second argument) on slave #1 (first argument). See the section Understanding sequencer code.

Action properties

All other customization and configuration for valve actions can be set in the main configuration file (olfStim/configuration/olfStimConfiguration.m) under the 'Valves' section. Every action you want to be able to access, alter and use from the gui, has to be defined by one entry in each of the various configuration variables (actionNames, values, used, etc.) and the order of entries in each variable has to be

the same. The purpose of each variable is described in detail in the configuration file.
One can rename, add or delete valve actions, by altering, adding or deleting the corresponding entries in the variable. The list of action names in the 'actionNames' variable have to match the filename (without the .lsq) of the valve action, as described above.

As an example not all users will need a "sniffing valve". We would therefore delete 'openSniffingValve' and 'closeSniffingValve' from the actionNames variable. Also in all other variables of the Valves section in the configuration file, one would have to remove the entry

Configuring I/O actions

The configuration and customization of I/O actions follows the same logic as configuring valve actions.
Add snippets of sequencer code (lsq files) for each I/O action to the olfStim/lsq/ioCodes/ folder in the same way you add valve actions.
The default parameters for the actions can be set in the olfStimConfiguration.m file under the "I/O" section.

Configuring the look of the odorSelectionGui

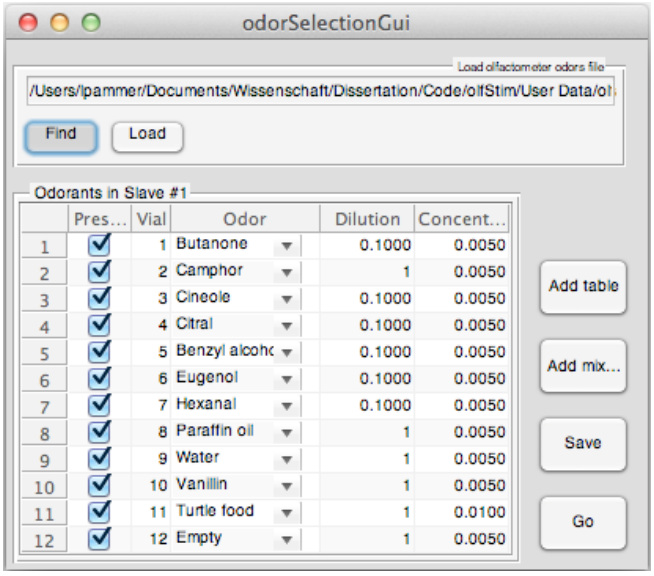
In the olfStimConfiguration.m file under the rubric "odorSelectionGui" some default parameters for the odorant vials setup can be entered. You'll find a detailed description of all parameters in the configuration file. For instance the number of vials displayed in the odorantSelectionGui can be set. This might be useful if you want to load more than the current default of 9 odor vials into the olfactometer.

Running olfStim from the gui

In the Matlab command line type

```
initOlfStim
```

This will result in the odorSelectionGui to open.



For every slave, choose the odorant in every connected vial. In the first column check whether you want to use this odor in the session. In the third column choose the odorant contained in the vial from the dropdown menu – the second column specifies the vial number.
Make sure that the dilution of the odorant in the vial (eg you diluted Cineole in Paraffin oil) is specified correctly in the 4th column. For the standardStim protocol it is necessary to set the concentration at presentation (v/v) in the 5th column.

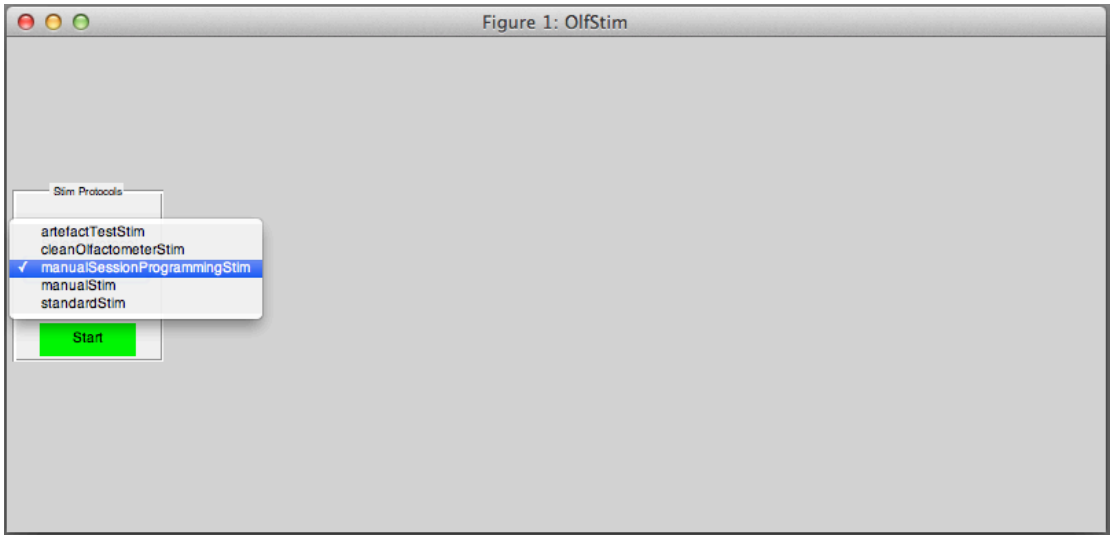
In order to avoid having to enter the loaded odorants every time, you can save the settings to a file by clicking the “Save” button. Previously saved settings can be loaded by specifying the location of the file with the “Find” button and loading it with the “Load” button.

NOTE: odor mixtures can not yet be used.

Once the Gui contains the information which odor is contained in which vial click the “Go” button.

You can change the default look of the odorSelectionGui by changing parameters in the olfStimConfiguration.m file section “Odorants”

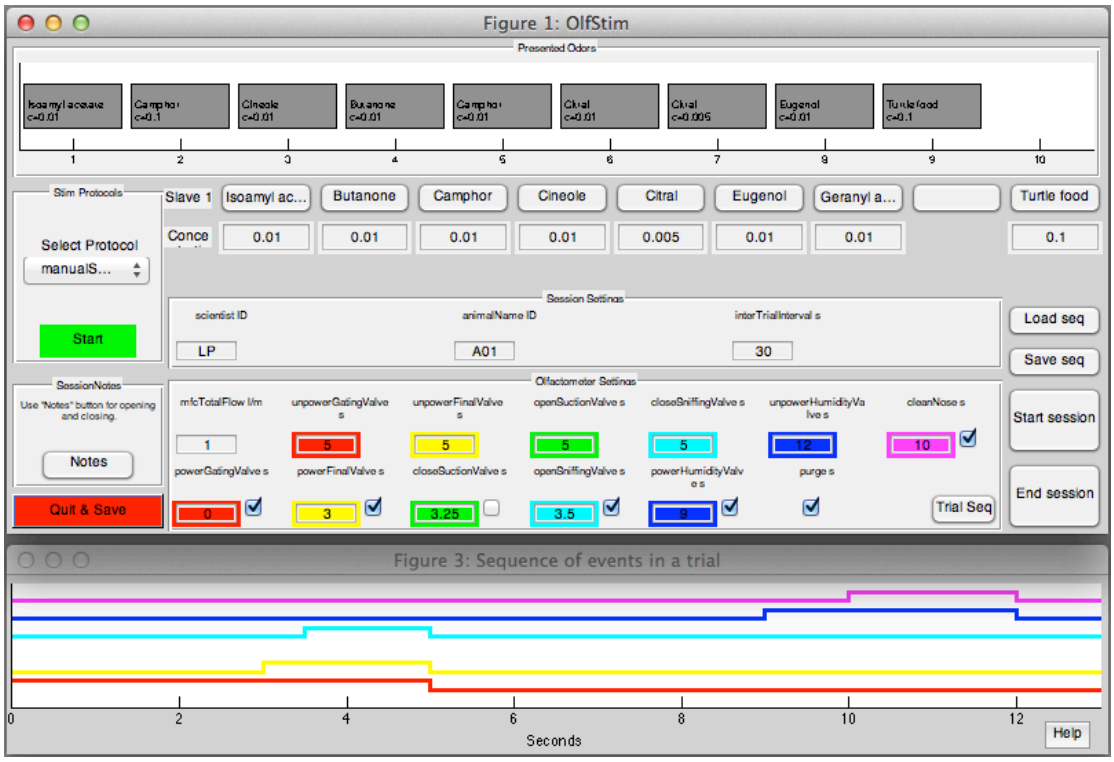
The main gui will now open and you have to choose which stimulation protocol you want to use. See details about the protocols below. Then click the “Start” button.



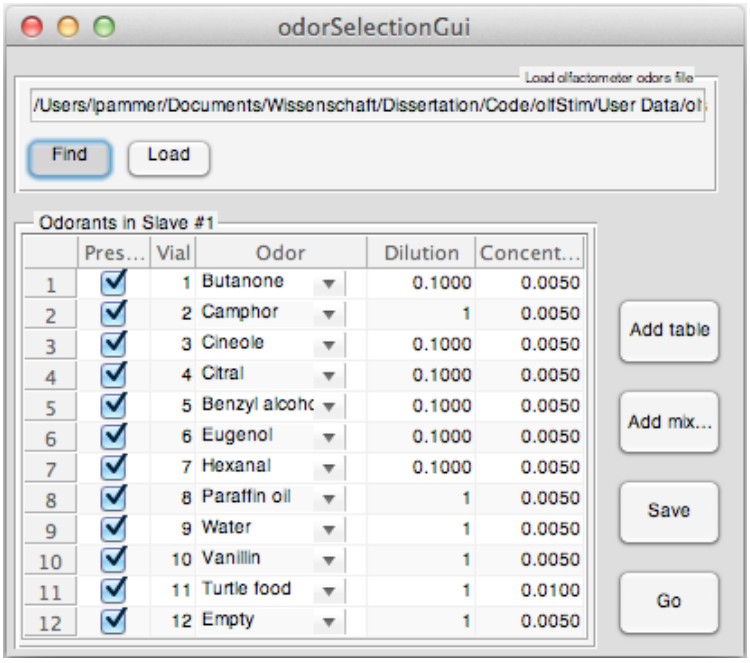
This results in the gui being populated by all features of the chosen protocol. You are now ready to start presenting odors! For the explanation of the protocols' features read the protocols section in this document.

Features

In this section olfStim's general features are documented.



Odorant Selection



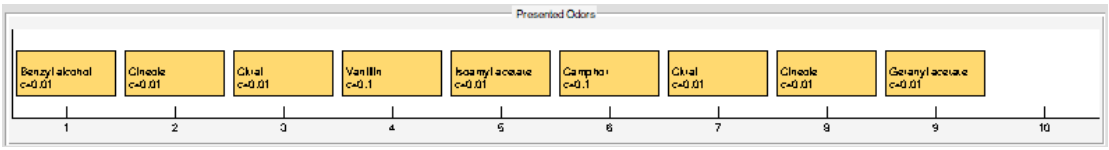
The gui for odor selection is opened automatically, when starting olfStim in Gui mode with initOlfStim. However it can also be opened by itself, by calling

```
odorSelectionGui
```

For every slave, choose the odorant in every connected vial. In the first column check whether you want to use this odor in the session. In the third column choose the odorant contained in the vial from the dropdown menu – the second column specifies the vial number.
See that the dilution of the odorant in the vial (eg you diluted Cineole in Paraffin oil) is specified correctly in the 4th column. For the standardStim protocol it is necessary to set the concentration at presentation (v/v) in the 5th column.

In order to avoid entering the loaded odorants every time, you can save the settings to a file by clicking the “Save” button. Previously saved settings can be loaded by specifying the location of the file with the “Find” button and loading it with the “Load” button.

Progress panel

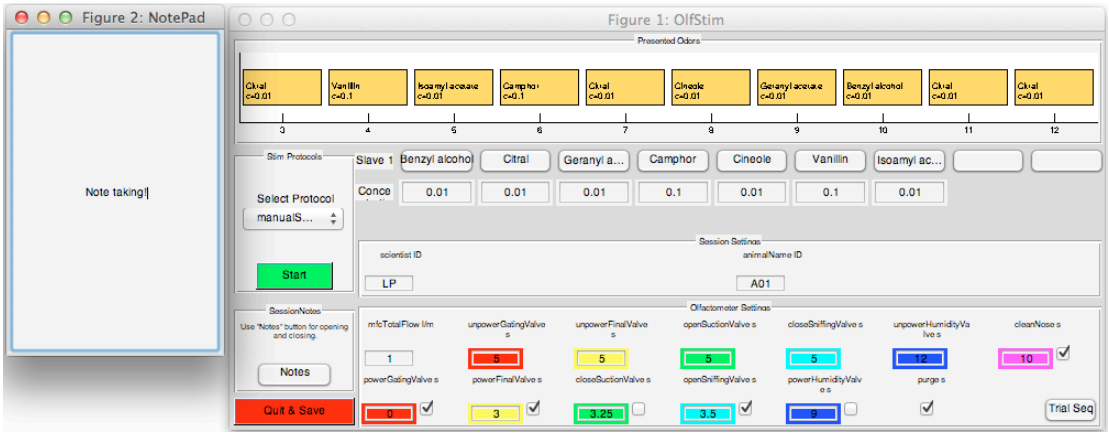


The progress panel is at the top of the GUI and shows the history of presented odors, identifying each trial with the odor name and the concentration.

Notes

Clicking on the “Notes” button on the left lower part of the GUI will open a new figure – the note pad - for note taking. In this figure you can write your notes for the current

trial. When the trial is triggered the notes are extracted and associated with the trial in the output data structure. At the same time the figure will be cleaned of the old notes. Use the “Notes” button both to open and close the note pad.



Olfactometer Settings

In the olfactometerSettings panel the times of valves opening & closing can be entered manually for each trial. Also multiple opening and closing times are allowed. This can be done by entering two values in square brackets (matlab vector notation). For instance entering [3 6] in the open sniffing valve and [5 8] in the close sniffing valve field will result in the sniffing valve opening at 3 s, closing at 5s, opening again at 6 s and closing again at 8 s.
Note however that at the end of the trial the resting state has to be reestablished. Which means you have to close/unpower every valve you open/power.

I/O and execution flow

Protocols

Cleaning

Start by filling all vials with Ethanol.

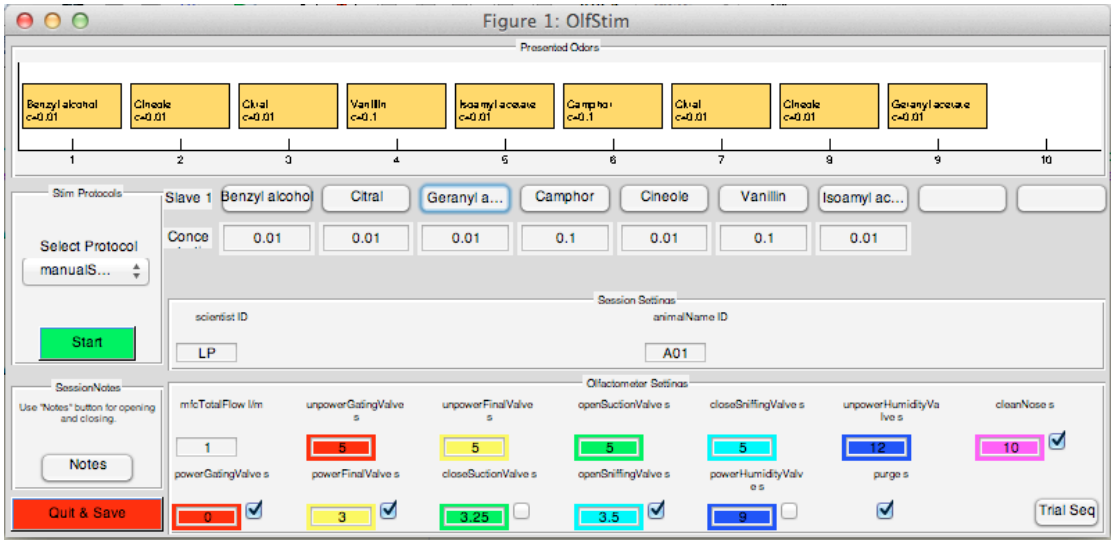
manualStim

The manualStim stimulation protocol is simple and a good starting point to test the olfactometer but also for exploratory experiments. It allows you to set the concentration of the odor presentation, control the total flow rate and the opening and closing times of relevant valves. Trials are triggered by manually clicking a button in the gui.

Every odor you have defined in the odorSelectionGui will show up in form of a button in the first row underneath the progress panel. Vial numbers ascend from left to right.

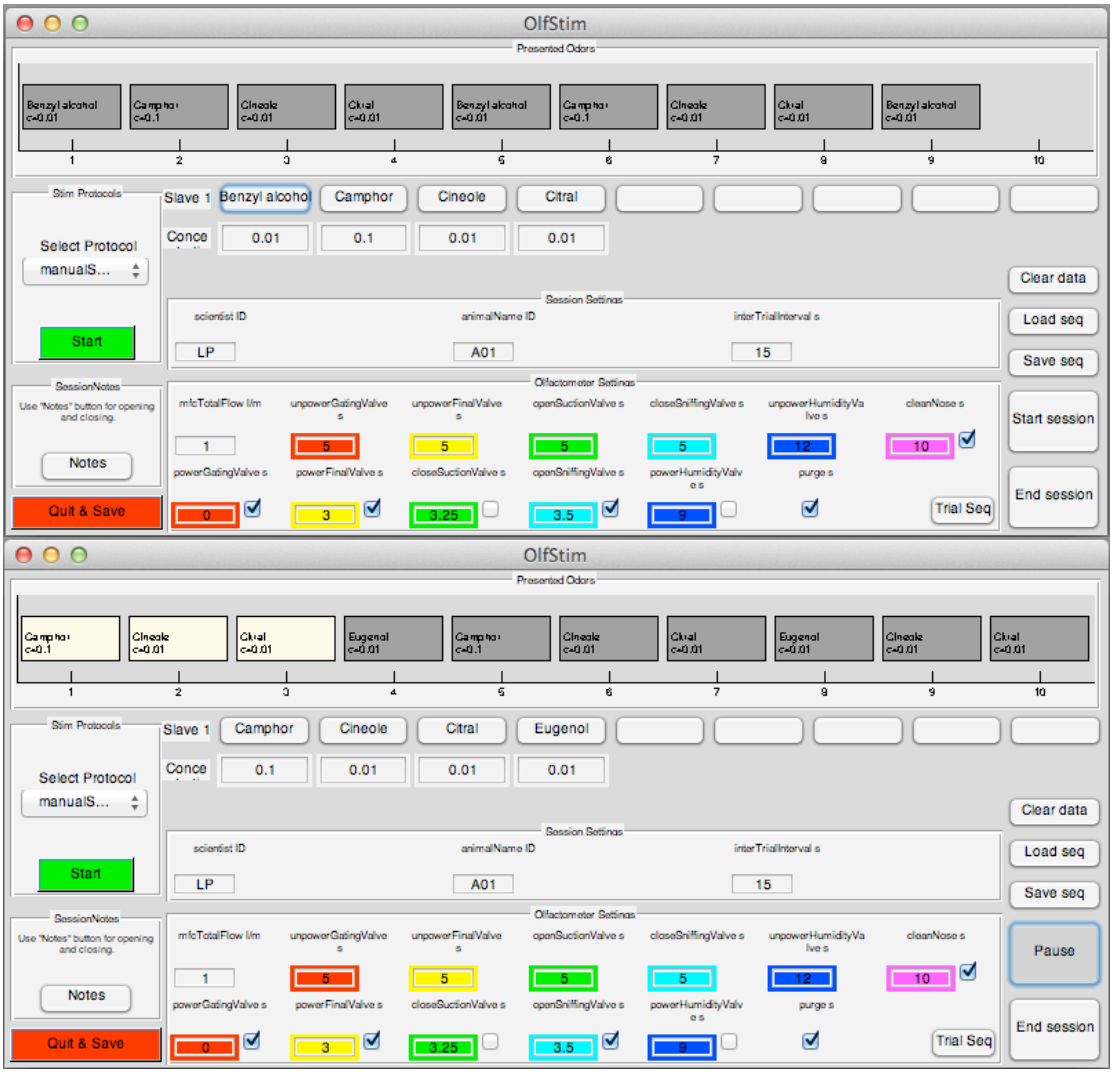
Under every odor button a concentration field is present. In the concentration field you can set the concentration (= fraction of saturated headspace) at which the odor will be presented. If you have set the dilution of the odorant in the vial correctly in the Odorant Library or the odorSelectionGui, olfStim will automatically adjust the flow rates accordingly.
If you enter concentration values, which cannot possibly be reached given the specification of the MFCs and your chosen total flow you will get a warning via pop up. For example, if you want to present an undiluted odor at a flow rate of 1 liter per minute and at a concentration of 0.3 you will get a warning, because the Nitrogen MFC has a maximum flow rate of 100 ml per minute. Another second error check will warn you if you enter a concentration, so low that the

Once you have set the concentration and the other relevant parameters for the trial you can press the odor button and the trial will start. Executed trials are shown in the progress panel.

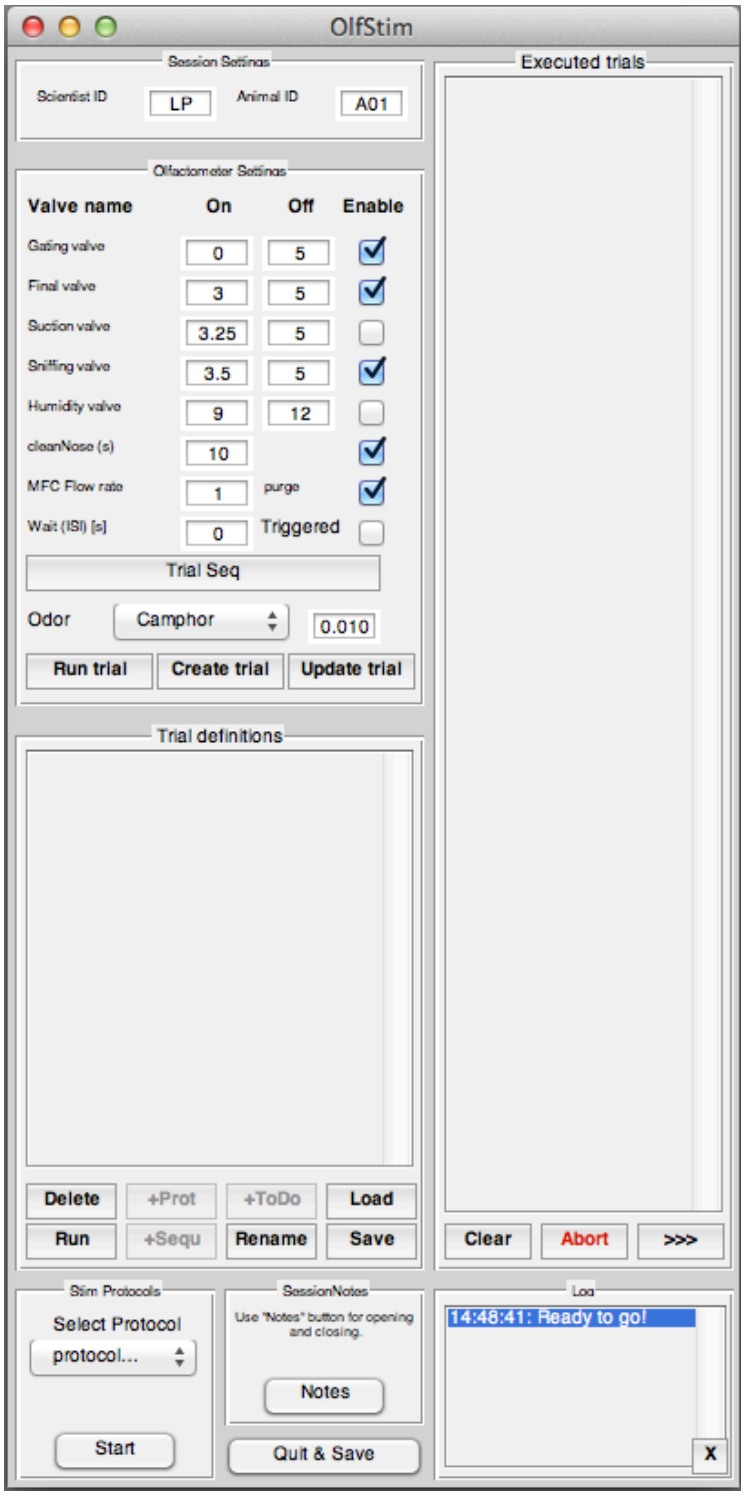


manualProgrammingStim

All the functionality described in manualStim are also available in manualProgrammingStim.


OlfStim
Presented Odors
Benzyl alcohol c=0.01
Camphor c=0.1
Cineole c=0.01
Citral c=0.01
Benzyl alcohol c=0.01
Camphor c=0.1
Cineole c=0.01
Citral c=0.01
Benzyl alcohol c=0.01
1 2 3 4 5 6 7 8 9 10
Stim Protocols
Select Protocol
manualS...
Start
SessionNotes
Use "Notes" button for opening and closing.
Notes
Quit & Save
Slave 1
Benzyl alcohol
Camphor
Cineole
Citral
Conce
0.01
0.1
0.01
0.01
Session Settings
scientist ID
LP
animalName ID
A01
interTrialInterval s
15
Olfactometer Settings
mfcTotalFlow l/m
1
unpowerGatingValves s
5
unpowerFinalValves s
5
openSuctionValves s
5
closeSniffingValves s
5
unpowerHumidityValves s
12
cleanNoses s
10
powerGatingValves s
0
powerFinalValves s
3
closeSuctionValves s
3.25
openSniffingValves s
3.5
powerHumidityValves s
9
purges
Trial Seq
Clear data
Load seq
Save seq
Start session
End session
OlfStim
Presented Odors
Camphor c=0.1
Cineole c=0.01
Citral c=0.01
Eugenol c=0.01
Camphor c=0.1
Cineole c=0.01
Citral c=0.01
Eugenol c=0.01
Cineole c=0.01
Citral c=0.01
1 2 3 4 5 6 7 8 9 10
Stim Protocols
Select Protocol
manualS...
Start
SessionNotes
Use "Notes" button for opening and closing.
Notes
Quit & Save
Slave 1
Camphor
Cineole
Citral
Eugenol
Conce
0.1
0.01
0.01
0.01
Session Settings
scientist ID
LP
animalName ID
A01
interTrialInterval s
15
Olfactometer Settings
mfcTotalFlow l/m
1
unpowerGatingValves s
5
unpowerFinalValves s
5
openSuctionValves s
5
closeSniffingValves s
5
unpowerHumidityValves s
12
cleanNoses s
10
powerGatingValves s
0
powerFinalValves s
3
closeSuctionValves s
3.25
openSniffingValves s
3.5
powerHumidityValves s
9
purges
Trial Seq
Clear data
Load seq
Save seq
Pause
End session

protocolsSJ_Stim



Running olfStim in test mode

In the Matlab commandline type

```
initOlfStim('testing')
```

or

```
initOlfStim('test')
```

This will start the test version of olfStim, which does not require the computer to be connected to an olfactometer. Most features not directly linked to olfStim-olfactometer interaction are active and can be tested. The test mode is useful if you want to start playing with the software but don't have an olfactometer connected, or if you're coding/hacking olfStim and want to test your code on your development machine. Keep in mind however that the test mode does not emulate the olfactometer's microprocessor, therefore not all of your code's potential bugs are caught.

Scripting olfStim

olfStim can also be scripted. This means if you don't want to use the GUI to control the olfactometer you can also do so by calling the functions of olfStim. This can be practical in several use cases, for instance if you have stereotypical stimulation sessions or if you want to access olfStim functionality from another program.

Initiation script

First open olfStim/scripting/olfStimScripting.m, the entry script into olfStim's scripting mode. It sets all necessary variables and paths and will then trigger the execution of the scripting protocol.
Go through the script and change all the marked lines to configure the scripting of olfStim for your computer and experiments.

Start scripting by executing the initiation script and providing some arguments:

```
olfStimScripting(protocol,#OfTrials,scientist,animalID,interTrialInterval)
```

Scripting protocol

Decide which scripting protocol you want to use and provide its name as an argument to the initiation script. The scripting protocols handle the definition of events (which odor to present at what concentration, when to open which valve, etc.) for every trial and trigger the trials.
Currently there is only one protocol provided in olfStim: 'randomOdorPresentation' - this protocol will randomly draw one of the odorants listed in the loaded odorant library file (structure called olfactometerOdors) and present it at the defined concentration.

You can add your own scripting protocols by writing a Matlab function. The procedure is simple:

1. Add a folder with the name of your protocol prefixed with a plus sign into olfStim/scripting/+scriptingProtocols. For example choose the folder name '+myScriptingProtocol'
2. Add a Matlab m-file into this folder and name it identically, only omitting the plus sign. For the example above this file would be called 'myScriptingProtocol.m'
3. When you now call the initiation script from the command line, provide 'myScriptingProtocol' as an argument and your new function will be called after the initiation script has set up the necessary variables and paths.

For Developers

If using LASOM it is necessary to set variable 1 ($Var1) to 0 before the trial begins:

```
SetVar, $Var1, 0
```

After the trial begins the variable has to be set to 1 and this updated status has to be emitted to the host computer:

```
SetVar, $Var1, 1
EmitStatus
```

At the end of the trial set the variable back to 0 and emit the new value:

```
SetVar, $Var1, 0
EmitStatus
```

This is necessary for the startTrial.m file to know in which state olfactometer is. EmitStatus causes the sequencer to emit the updated status to the host computer.

Adding new protocols:
Try to use the existing blocks of code in +protocolUtilities as much as possible. You have access to all gui components through the olfStimH handles struc. This means you can rearrange and change everything, but the functions will still be able to access the information without writing new code.