

MATLAB code/GUI setup & task running overview:

1. Please download MATLAB (version 2021b for consistency) and Arduino IDE.
2. Open the headfix_GUI.m file in MATLAB and change the file main path in line 70 -72 to desired folder saving path-one example path shown here. Please make sure to include a subfolder named “data” in the main folder. This is where all the data will be saved.

```
mainPath = 'C:\Users\mzhou\behavior';  
addpath(mainPath)  
saveDir = [mainPath '\data\'];           % where to save data
```

Figure 1 Set main path and data saving subfolder in MATLAB code

3. Create an “uploads” folder under the main folder and update the Arduino hex file upload locations in the “uploadButton_Callback” function (currently in lines 303-317).

```
function uploadButton_Callback(hObject, eventdata, handles)  
% hObject    handle to uploadButton (see GCBO)  
% eventdata  reserved - to be defined in a future version of MATLAB  
% handles    structure with handles and user data (see GUIDATA)  
portList = get(handles.availablePorts,'String'); % get list from popup menu  
selected = get(handles.availablePorts,'Value'); % find which is selected  
port      = portList{selected}; % selected port  
  
basecmd = strcat('C:\Program Files (x86)\Arduino\hardware\tools\avr\bin\avrdude' '-C"C:\Program Files (x86)\Arduino\hardware\tools\avr/etc/  
selectedmode = get(handles.experimentmode,'Value');  
  
if selectedmode == 1  
    [status,cmdout] = dos(strcat(basecmd,'C:\Users\mzhou\behavior\uploads\Namlab_behavior_cues.ino.hex','i'));  
elseif selectedmode == 2  
    [status,cmdout] = dos(strcat(basecmd,'C:\Users\mzhou\behavior\uploads\Namlab_behavior_randomrewards.ino.hex','i'));  
elseif selectedmode == 3  
    [status,cmdout] = dos(strcat(basecmd,'C:\Users\mzhou\behavior\uploads\Namlab_behavior_lickforreward.ino.hex','i'));  
elseif selectedmode == 4  
    [status,cmdout] = dos(strcat(basecmd,'C:\Users\mzhou\behavior\uploads\Namlab_behavior_decisionmaking.ino.hex','i'));  
elseif selectedmode == 5  
    [status,cmdout] = dos(strcat(basecmd,'C:\Users\mzhou\behavior\uploads\Serial_port_testing.ino.hex','i'));  
elseif selectedmode == 6  
    [status,cmdout] = dos(strcat(basecmd,'C:\Users\mzhou\behavior\uploads\Namlab_behavior_ramptiming.ino.hex','i'));  
elseif selectedmode == 7  
    [status,cmdout] = dos(strcat(basecmd,'C:\Users\mzhou\behavior\uploads\Namlab_behavior_delaydiscounting_automated.ino.hex','i'));  
end
```

Figure 2 Create uploads subfolder and change the address in the MATLAB code

4. Open each Arduino task script .ino file in Arduino IDE. Open “Preferences” under the “File” menu bar and select “Show verbose output during upload”.

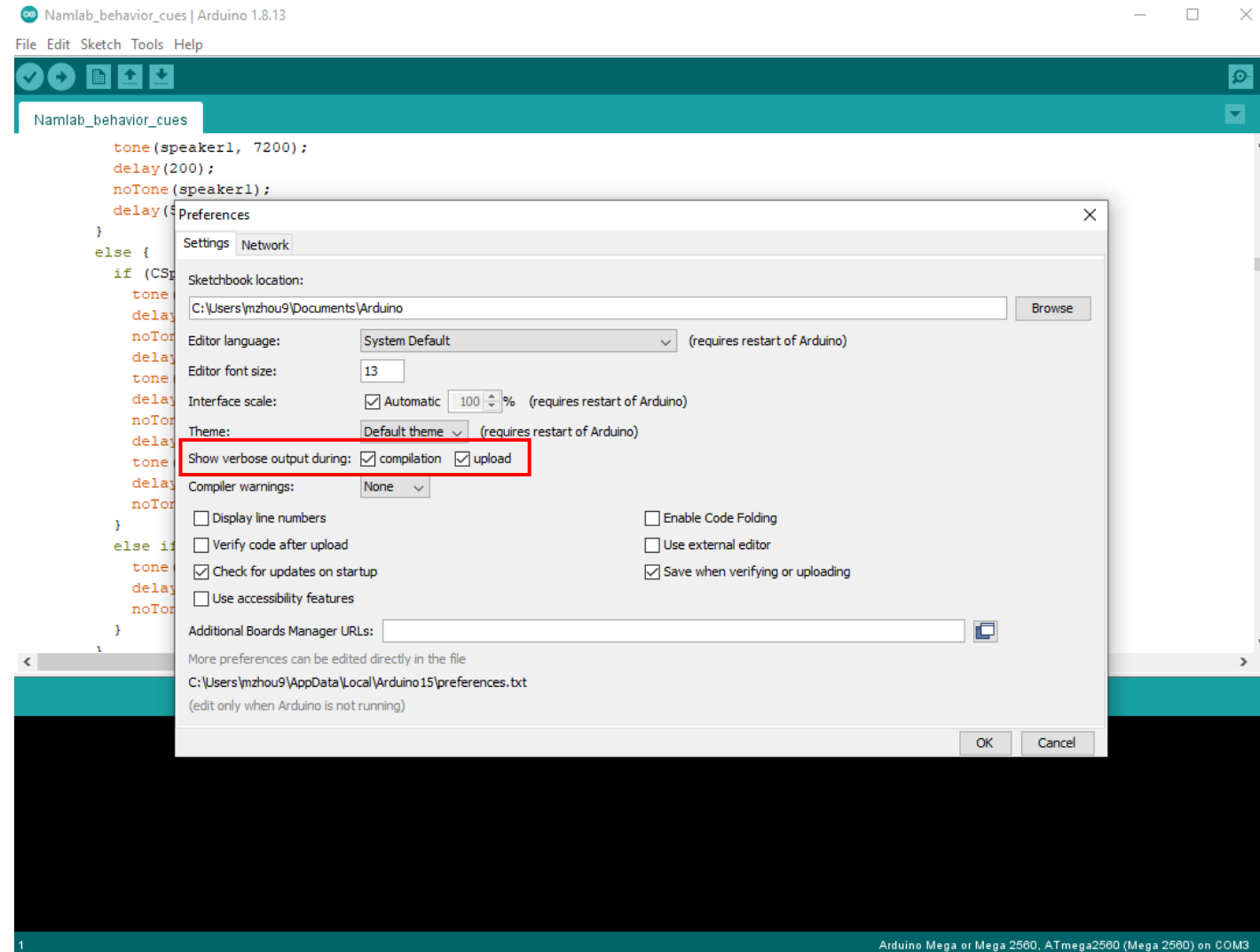


Figure 3 Preference window

5. Under the “Tools” menu bar, select board for Arduino mega 2560 board and select a vacant Arduino board in port. Hit the upload button to upload this file. Note down the uploaded file path. There are many online websites explaining how to get this hex file location for a code. An example is here: <https://www.instructables.com/HOW-TO-GET-HEX-FILE-FROM-ARDUINO-/>
6. Go to the uploaded folder path. Copy and paste the .hex file into the “uploads” folder created in Steps 2 and 3. (Repeat for all Arduino task scripts).

Name	Date Modified	Size	Kind
Namlab_behavior_cues.ino.hex	Oct 12, 2021 at 4:35 PM	64 KB	Document
Namlab_behavior_decisionmaking.ino.hex	Oct 6, 2021 at 10:29 PM	66 KB	Document
Namlab_behavior_d...g_automated.ino.hex	Oct 6, 2021 at 10:30 PM	46 KB	Document
Namlab_behavior_lickforreward.ino.hex	Oct 6, 2021 at 10:31 PM	44 KB	Document
Namlab_behavior_ramptiming.ino.hex	Oct 6, 2021 at 10:31 PM	61 KB	Document
Namlab_behavior_randomrewards.ino.hex	Oct 6, 2021 at 10:36 PM	44 KB	Document
Serial_port_testing.ino.hex	Oct 6, 2021 at 10:38 PM	7 KB	Document

Figure 4 Uploads folder example

7. Make sure the conditioning_prog.m file is in the same folder as the headfix_GUI.m and headfix_GUI.fig files. This is for saving the experiment data and parameters and plotting real-time events during the experiment.

Task running instructions:

1. Open the headfix_GUI.m and hit “Run” function under the “Editor” menu.
2. Select available COM ports (corresponds to which Arduino board to connect to).

Available ports:

COM3

Figure 5 Choose an available Arduino port number

3. Select the experiment mode to run (1=run with cues, 2=only with random rewards given, 3=lick dependent rewards, 4=decision making task, 6=ramp timing task, 5=testing serial port connection) and press the “Upload to arduino” button to upload this selected experiment hex file to the selected Arduino board. If successful, the

button text will display as “Successfully uploaded”. If unsuccessful, the button text displays “Unable to upload”; in that case, please check the file location and the corresponding code in the MATLAB code line 303-317.

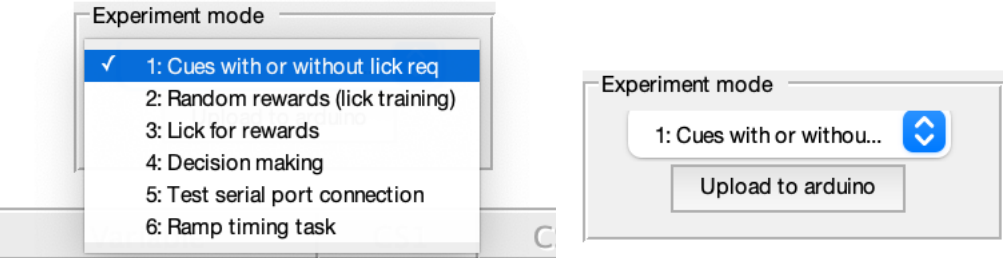


Figure 6 Select experiment mode and upload to Arduino

4. Hit “Link” to connect the computer and Arduino board with the serial port. Different sections of parameter editing will become active based on the experiment mode selected. (see [GUI parameter full explanation section](#) and [possible task section](#) below for how to set different parameters in different tasks).

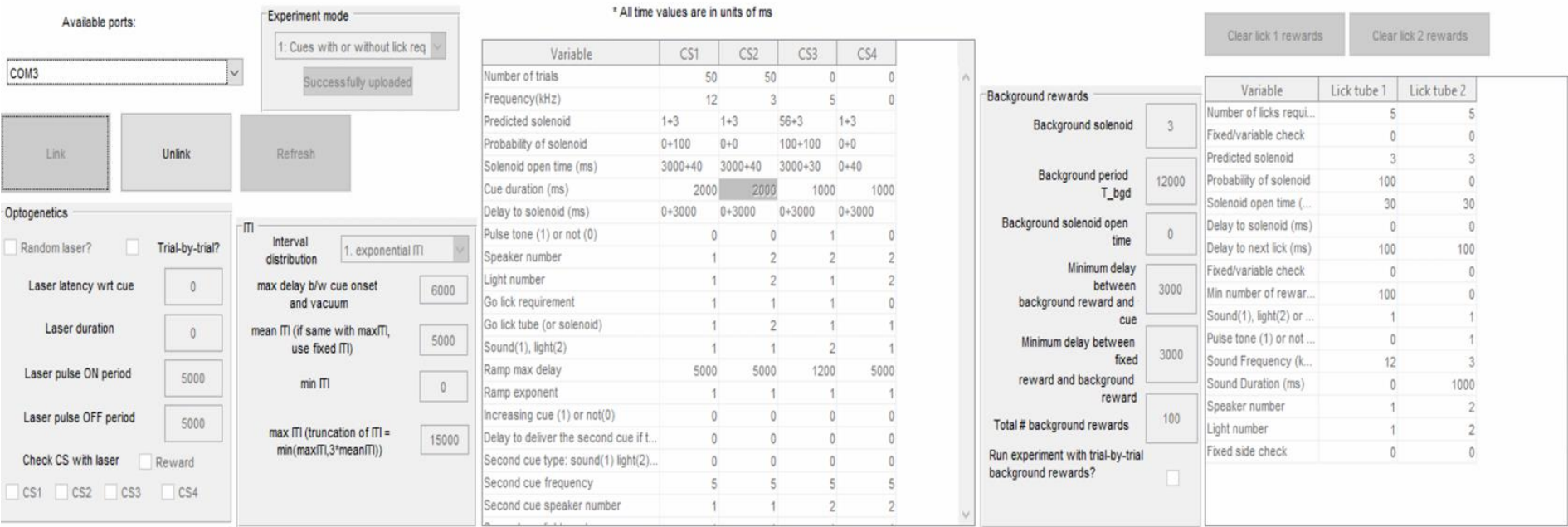


Figure 7 Parameter setting section

5. After setting up all the parameters correctly, hit the “Send” button to send all parameters to the Arduino board.

Connected to:

port not selected

Save as:

enteranimalname_

Send

Start

Stop

Figure 8 Buttons for sending parameters to Arduino, starting experiment and stopping experiment

6. Test hardware components before starting the experiment if needed. For example, pressing “Test CS1”, “Solenoid 2 Manual”, “Lick retract solenoid 1 Manual” will turn on the hardware for a set length. Pressing the “Prime” button under any solenoid section will keep that particular solenoid open until the button is unclicked.

Test CS1

Test CS2

Test CS3

Test Serial Port

Test LASER

Solenoid 1

Manual

Prime

Lick retract solenoid 1

Manual

Prime

Solenoid 2

Manual

Prime

Lick retract solenoid 2

Manual

Prime

Solenoid 3

Manual

Prime

Vacuum

Manual

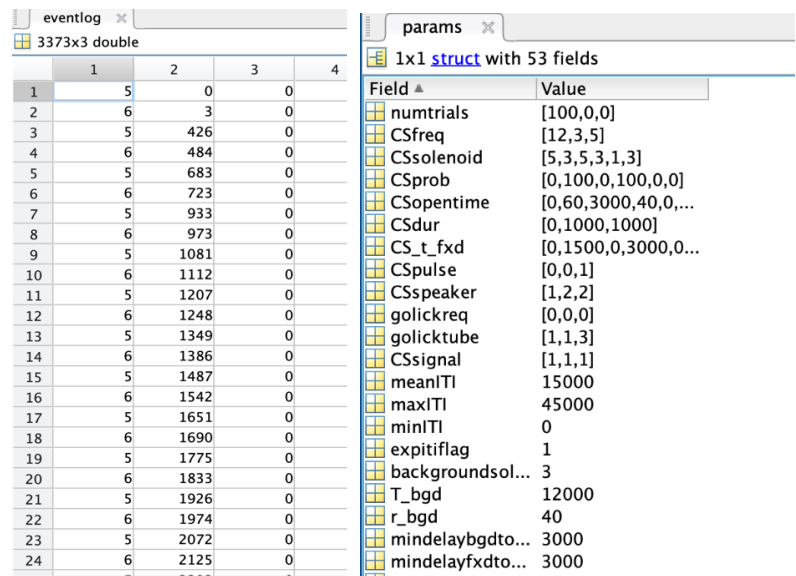
Solenoid4

Manual

Prime

Figure 9 Testing hardware section

7. Start the experiment by hitting the “Start” button in **Figure 7**.
8. When the session ends, the experiment will stop automatically and be saved to the folder saving path (set during Step 2) as a .mat file with two variables: eventlog and params as a .mat file.
- a. eventlog contains three columns: 1st column is all the event identifiers; 2nd column is time stamp of the event; 3rd column is identifier flag for either reward delivery status of the solenoid: 0=given, 1=missed, or for identifying the cue order: 0=first cue, 1=second cue).
 - b. params is a struct with all parameter names and values set for this experimental session for future reference.



The figure displays two MATLAB variable previews. The 'eventlog' variable is a 3373x3 double matrix. The 'params' variable is a 1x1 struct with 53 fields.

Field	Value
numtrials	[100,0,0]
CSfreq	[12,3,5]
CSsolenoid	[5,3,5,3,1,3]
CSprob	[0,100,0,100,0,0]
CSopentime	[0,60,3000,40,0,...]
CSdur	[0,1000,1000]
CS_t_fxd	[0,1500,0,3000,0,...]
CSpulse	[0,0,1]
CSspeaker	[1,2,2]
golickreq	[0,0,0]
golicktube	[1,1,3]
CSsignal	[1,1,1]
meanITI	15000
maxITI	45000
minITI	0
expitflag	1
backgroundsol...	3
T_bgd	12000
r_bgd	40
mindelaybgdto...	3000
mindelayfxdto...	3000

Figure 10 Saved MATLAB file data preview containing two variables: eventlog and params

Graphical User Interface Parameters Explanation

Graphical User Interface Sections	MATLAB saved parameter [# of items for this parameter]	Corresponding Arduino parameter	Description of parameters
CS properties window	numtrials[4]	numtrials[0-3]	Number of trials for CS1-4
	CSfreq[4]	CSfreq[0-3]	Sound frequencies for auditory cues (in kHz)
	CSsolenoid[8]	CSsolenoid[0-7]	Designated two solenoids for each of the three cues 1-4:solenoid 1-4; 5/6: retraction solenoid 1/2; 56: retraction solenoid 1&2
	CSprob[8]	CSprob[0-7]	Solenoid open probability
	CSopentime[8]	CSopentime[0-7]	Solenoid open time
	CSdur[8]	CSdur[0-3]	Cue durations
	CS_t_fxd[8]	CS_t_fxd[0-7]	Fixed delay from cue to solenoids
	CSpulse[4]	CSpulse[0-3]	Whether CS is a pulsed cue or not, 0=not pulsed, 1=pulsed
	CSspeaker[4]	CSspeaker[0-3]	CS1-4 first cue speaker number
	golickreq[4]	golickreq[0-3]	For cue-action-reward task, number of lick requirements met on selected tube to obtain the second solenoid for each cue
	golicktube[4]	golicktube[0-3]	designated lick tube for cue-action-reward task
	CSsignal[4]	CSsignal[0-3]	CS first cue signal type, 1=sound, 2=light
	CSlight[4]	CSlight[0-3]	CS1-4 first cue light number
	CSrampmaxdelay[4] †	CSrampmaxdelay[0-3]	ramp max delay for ramp timing task for CS1-4
	† see legend at the bottom of the table about parameters in brown text		

Variable	CS1	CS2	CS3	CS4
Number of trials	25	25	25	25
Frequency(kHz)	12	3	12	3
Predicted solenoid	56+3	56+3	56+3	56+3
Probability of solenoid	100+100	100+100	100+100	100+100
Solenoid open time (ms)	3000+40	3000+40	3000+40	3000+40
Cue duration (ms)	1000	1000	1000	1000
Delay to solenoid (ms)	1000+4000	1000+4000	1000+4000	1000+4000
Pulse tone (1) or not (0)	0	0	1	0
Speaker number	1	2	2	2
Light number	1	2	1	2
Go lick requirement	1	1	1	1
Go lick tube (or solenoid)	1	1	2	2
Sound(1), light(2)	1	1	1	1
Ramp max delay	5000	5000	1200	5000
Ramp exponent	1	1	1	1
Increasing cue (1) or not(0)	0	0	0	0
Delay to deliver the second cue if t...	1000	1000	1000	1000
Second cue type: sound(1) light(2)...	1	1	1	1
Second cue frequency	12	3	3	12
Second cue speaker number	1	2	2	1
Second cue light number	1	1	1	1

	CSrampexp[4]	CSrampexp[0-3]	ramp exponential factor for ramp timing task for CS1-4
	CSincrease[4]	CSincrease[0-3]	Increasing or decreasing frequency for CS1-4
	delayforsecondcue[4]	delayforsecondcue[0-3]	Time between first and second cue if there is second cue for each CS1; solenoids are delivered based on second cue onset
	CSsecondcue[4]	CSsecondcue[0-3]	Check if there is a second cue for each CS, 1=sound, 2=light, 0=no second cue
	CSsecondcuefreq[4]	CSsecondcuefreq[0-3]	Second cue frequency if it is a sound cue
	CSsecondcuespeaker[4]	CSsecondcuespeaker[0-3]	Second cue speaker number
	CSsecondcuelight[4]	CSsecondcuelight[0-3]	Second cue light number
<div>Inter-trial Interval Section</div> <div><div>ITI</div><div>Interval distribution1. exponential ITI</div><div>max delay b/w cue onset and vacuum6000</div><div>mean ITI (if same with maxITI, use fixed ITI)30000</div><div>min ITI0</div><div>max ITI (truncation of ITI = min(maxITI, 3*meanITI))90000</div></div>	meanITI	meanITI	average inter-trial-interval (ITI) for exponential trials
	maxITI	maxITI	max ITI for exponential ITI trials, equals to truncation of ITI = min(maxITI, 3*meanITI)
	minITI	minITI	minimum ITI for uniform distribution trials
	expitiflag	intervaldistribution	exponential ITI indicator, unticked(0) = uniform, ticked(1) = exponential
	maxdelaycuetovacuum	maxdelaycuetovacuum	maximum delay to vacuum after cue turns on. Change this if different cues have different delays to reward

<div>Background rewards section</div> <div><div>Background rewards</div><div><div>Background solenoid</div><div>3</div></div><div><div>Background period T_bgd</div><div>12000</div></div><div><div>Background solenoid open time</div><div>0</div></div><div><div>Minimum delay between background reward and cue</div><div>3000</div></div><div><div>Minimum delay between fixed reward and background reward</div><div>3000</div></div><div><div>Total # background rewards</div><div>100</div></div><div><div>Run experiment with trial-by-trial background rewards?</div><div><input type="checkbox"/></div></div></div>

Variable	Lick tube 1	Lick tube 2
Number of licks requi...	5	5
Fixed/variable check	0	0
Predicted solenoid	3	3
Probability of solenoid	100	0
Solenoid open time (...)	30	30
Delay to solenoid (ms)	0	0
Delay to next lick (ms)	100	100
Fixed/variable check	0	0
Min number of rewar...	100	0
Sound(1), light(2) or ...	1	1
Pulse tone (1) or not ...	0	1
Sound Frequency (k...	12	3
Sound Duration (ms)	0	1000
Speaker number	1	2
Light number	1	2

lickprob[2]	lickprob[0-1]	probability of solenoid opening upon completing required licks on side lick tube 1 and 2
lickopentime[2]	lickopentime[0-1]	solenoid open time (ms)
delaytoreward[2]	delaytoreward[0-1]	delay to solenoid/reward after the last required lick
delaytolick[2]	delaytolick[0-1]	delay to start counting the next lick after reward consumption
minrewards[2]	minrewards[0-1]	minimum number of rewards to give for each session
signaltolickreq[2]	signaltolickreq[0-1]	cue type signaling lick requirement met, 1=sound, 2=light, 3=both
soundsignalpulse[2]	soundsignalpulse[0-1]	whether the cue signal is pulsed or not, 0=not pulsed, 1=pulsed
soundfreq[2]	soundfreq[0-1]	auditory signal frequency (in kHz)
sounddur[2]	sounddur[0-1]	both sound and light cue duration (ms)
lickspeaker[2]	lickspeaker[0-1]	speaker number for auditory cues
variableratioflag[2]	variableratioflag[0-1]	flag for variable ratio schedule for lick requirements, if=1, then the task is a variable ratio schedule, if=0, fixed ratio schedule
variableintervalflag[2]	variableintervalflag[0-1]	flag for variable interval schedule for lick requirements, if=1, variable interval schedule task, if=0, fixed interval schedule
fixedsidecheck[2]	fixedsidecheck[0-1]	for delay discounting task, if=0, selected lick tube is not a fixed side, give rewards opening time

			varies from 0-60ms, if=1, selected lick tube is a fixed side with opening of 30ms.
	licklight[2]	licklight[0-1]	light number for light cues
<div> <div>Laser / Optogenetics section</div> <div>Optogenetics</div> <div> <input type="checkbox"/> Random laser? <input type="checkbox"/> Trial-by-trial? </div> <div> Laser latency wrt cue <input type="text" value="0"/> </div> <div> Laser duration <input type="text" value="0"/> </div> <div> Laser pulse ON period <input type="text" value="5000"/> </div> <div> Laser pulse OFF period <input type="text" value="5000"/> </div> <div> <input type="checkbox"/> Check CS with laser <input type="checkbox"/> Reward </div> <div> <input type="checkbox"/> CS1 <input type="checkbox"/> CS2 <input type="checkbox"/> CS3 <input type="checkbox"/> CS4 </div> </div>	laserlatency	laserlatency	laser latency with respect to cue (ms)
	laserduration	laserduration	laser on duration (ms)
	randlaserflag	randlaserflag	flag to run sessions with laser turning on randomly if==1
	laserpulseperiod	laserpulseperiod	period for which laser is on in a cycle (ms)
	laserpulseoffperiod	laserpulseoffperiod	period for which laser is off in a cycle (ms); If equal to laserpulseperiod, duty cycle is 50%
	lasertrialbytrialflag	lasertrialbytrialflag	flag to turn laser on a trial-by-trial basis
	CS1lasercheck	CS1lasercheck	laser-on flag for CS1, 1==laser on, 0==laser off
	CS3lasercheck	CS2lasercheck	laser-on flag for CS2, 1==laser on, 0==laser off
	CS3lasercheck	CS3lasercheck	laser-on flag for CS3, 1==laser on, 0==laser off
	CS4lasercheck	CS4lasercheck	laser-on flag for CS4, 1==laser on, 0==laser off
	Rewardlasercheck	Rewardlasercheck	laser-on flag for reward, 1==laser on, 0==laser off

*Order of the parameters saved in MATLAB/Arduino generally follows this chart with some exceptions of newly added parameters saved at the end.

†Brown inked parameters were not demonstrated in the behavioral tasks listed in the paper but can be useful for other tasks and manipulations

General Programming Logic

In general, the idea is that the Arduino controls each experiment. Thus, the primary code is the Arduino code for each experiment. In this code, all processing is event-based. Thus, the Arduino keeps track of time since the start of the experiment and checks if the time for a new event has come and if so, it executes that event, updates time for the next events, and goes back to check whether it is time for each new event, etc. To make the program as easy as possible for a user to run (i.e., without programming knowledge), the user-facing part of the code is only the MATLAB GUI. In this GUI, the users can set the experiment mode, parameters for the experiments, test hardware, receive timestamps for events from Arduino and visualize the behavior in real time. If one separate MATLAB process is opened per behavioral box and each Arduino is connected via a separate USB cable to the computer running the MATLAB instances, the task can be run in parallel for multiple behavior boxes.

Possible tasks

Here, we list the possible behavioral tasks that can be run using B-CALM, including a few planned extensions being worked on currently. Please note that the MATLAB GUI is currently built using GUIDE. However, we are porting this code away from GUIDE to increase compatibility across MATLAB versions. All current codes were tested on MATLAB v2021b.

Pavlovian conditioning

Pavlovian delay or trace conditioning with arbitrary reward magnitude, probability, or delays, and two possible reward/punishment identities for up to three trial types (one trial type per conditioned stimulus). We are currently extending this to a fourth conditioned stimulus. Please also note that the reward identities can be changed by simply changing the liquid in the containers attached to the central liquid dispensing solenoids. Though only two separate liquids can be delivered within a single behavioral session, different liquids can be delivered across different sessions. Possible liquids include water, sucrose, flavored sucrose, flavored milk, quinine, alcohol, orally deliverable drug solutions etc. Other currently possible parameters include:

- Intertrial interval (ITI): currently, we can set the intertrial interval to be exponentially distributed (i.e., with flat hazard function), uniformly distributed, exponentially distributed between cue-to-cue (instead of reward-to-cue), exponentially distributed and independent for both cue and reward (this is Rescorla's "truly random control"), or exponentially distributed between reward-to-reward (with cue coming after reward, i.e., backward conditioning). These are available in the "interval distribution" drop down menu.
- Stimulus (sound/light) properties: The frequency of the sound, whether or not it is pulsing (turns on and off with a period of 200 ms; hardcoded into the corresponding Arduino program), or whether it is a chirp with a frequency increase or decrease ("Increasing cue(1) or not") can be set. For a light, one of two possible LEDs can be turned on ("Light number").

- Unpredictable rewards during ITI for contingency degradation: It is possible to deliver unpredictable rewards during the ITI at a defined rate to reduce the contingency between a cue and reward. This is editable in the “Background rewards” section of the GUI (see below).
- Optogenetics laser: It is possible to control when a laser turns on relative to the cue onset, its duration, pulse on and pulse off period, which trial types the laser is activated on, whether the laser should be delivered pseudorandomly only in half of the trials (“Trial-by-trial?” checkbox), and whether the laser should be delivered randomly during the ITI. These are available in the “Optogenetics” panel.
- Multiple cues on the same trial: In some cases, it is necessary to deliver two cues on the same trial. This is the case for blocking, sensory preconditioning, occasion setting, or biconditional discrimination. For these experiments, set CS second cue type to 1= sound or 2 = light. Indicate the delay for second cue—this is the delay between first cue onset and second cue. In this case, the cue duration for both stimuli are set to be equal (set in “Cue duration”), and the “delay to solenoid” is measured from the second stimulus. Set the second cue properties such as second cue frequency, speaker number or light number accordingly. Please note that all solenoids are delivered after second cue onset which means delays to solenoids should account for the delay between first cue and second cue and cannot be shorter than delay to second cue.

Operant conditioning

Operant conditioning with licking on left/right lick tubes to get rewards on the center lick tube. This design allows operant action (lick left/right) to be separated from the consummatory action (lick center). Possible schedules include fixed ratio, variable ratio, fixed interval, variable interval, or their mixtures. In self-administration like variants, a conditional stimulus can be delivered immediately prior to reward delivery. These parameters are set in the sections of the GUI shown below.

Variable	Lick tube 1	Lick tube 2
Number of licks requi...	5	5
Fixed/variable check	0	0
Predicted solenoid	3	3
Probability of solenoid	100	0
Solenoid open time (...)	30	30
Delay to solenoid (ms)	0	0
Delay to next lick (ms)	100	100
Fixed/variable check	0	0
Min number of rewar...	100	0
Sound(1), light(2) or ...	1	1
Pulse tone (1) or not ...	0	1
Sound Frequency (k...	12	3
Sound Duration (ms)	0	1000
Speaker number	1	2
Light number	1	2
Fixed side check	0	0

Figure 11 Operant conditioning task parameter setting example

Cue-action-reward tasks

A discriminative stimulus based operant conditioning task (more generally, a cue-action-reward task) can be run such that an animal needs to perform an operant action (lick left/right) after receiving a specific cue to obtain reward. This is set by changing the “Go lick requirement” (how many licks to get reward) and “Go lick tube (or solenoid)” (which lick tube to lick to meet the requirement; could be left, right, or center). One can also set the go lick requirement to 0 and set the reward probability to 100 but deliver a punishment (e.g., quinine) in the central lick tube. Under these settings, the animal must withhold licking to avoid receiving the punishment.

Each trial type (listed under the CS columns) can cause two solenoid openings. This is useful for either multiple reward deliveries in a trial (possibly different identities) or presenting the operant lick tubes (left/right), which are themselves mounted on solenoids that allow presentation or retraction of the lick tubes. Therefore, the solenoid related entries in the following GUI section contain a “+” sign. By activating (or not) the retraction solenoids, the operant lick tubes can either be made available for licking only during the trial period, or for the entire session. For the former scenario, set the first of the predicted solenoid (i.e., the one before the “+”) to be the solenoid controlling the retractable lick tubes for the appropriate duration of opening (corresponding to how long it should stay available during the trial). Importantly, we assume that the second solenoid (the one after the “+”) is the reward delivery solenoid. It is this solenoid whose opening is contingent on meeting the “Go lick requirement”.

These parameters are set in the sections of the GUI shown below.

Variable	CS1	CS2	CS3	CS4
Number of trials	50	50	0	0
Frequency(kHz)	12	3	5	0
Predicted solenoid	5+3	6+3	1+3	1+3
Probability of solenoid	100+100	100+0	0+0	0+0
Solenoid open time (ms)	3000+40	3000+40	0+30	0+40
Cue duration (ms)	2000	2000	1000	1000
Delay to solenoid (ms)	0+3000	0+3000	0+3000	0+3000
Pulse tone (1) or not (0)	0	0	1	0
Speaker number	1	2	2	2
Light number	1	2	1	2
Go lick requirement	1	0	0	0
Go lick tube (or solenoid)	1	2	3	1
Sound(1), light(2)	1	1	1	1
Ramp max delay	5000	5000	1200	5000
Ramp exponent	1	1	1	1
Increasing cue (1) or not(0)	0	0	0	0
Delay to deliver the second cue if t...	0	0	0	0
Second cue type: sound(1) light(2)...	0	0	0	0
Second cue frequency	5	5	5	5
Second cue speaker number	1	1	2	2
Second cue light number	1	1	1	1

Figure 12 Cue-action-reward task parameter setting example

Timing tasks

Multiple timing tasks are possible. The first is the task shown in the manuscript. Here, rewards are delivered at fixed delays between each other and during the test phase, some rewards are omitted to test whether the animals perform the consummatory action at the expected reward time. To run this, set the cue duration to zero, delay to solenoid to be “0+0”, probability of solenoid to be “0+100”, and set the ITI to be uniform and at the desired inter-reward duration (e.g., for 10s inter-reward interval, set the ITI to be 9990-10010 ms).

In the second timing task, it is possible to make animals wait a desired duration after a cue until they lick for the first time. This “ramp timing” task can be controlled using the “ramp max delay” and “ramp exponent” parameters. A variant of this task can also be run without cues to produce internally generated well-timed licking. We will publish these tasks separately, as they are novel tasks and require further characterization than is possible here.

These parameters are set in the sections of the GUI shown below.

Variable	CS1	CS2	CS3	CS4
Number of trials	100	0	0	0
Frequency(kHz)	12	3	12	3
Predicted solenoid	1+3	5+3	5+3	5+3
Probability of solenoid	0+100	100+100	100+100	100+100
Solenoid open time (ms)	0+40	3000+40	3000+40	3000+40
Cue duration (ms)	0	1000	1000	1000
Delay to solenoid (ms)	0+0	0+3000	0+3000	0+3000
Pulse tone (1) or not (0)	0	0	0	0
Speaker number	1	2	1	2
Light number	1	2	1	2
Go lick requirement	1	1	1	1
Go lick tube (or solenoid)	1	1	2	2
Sound(1), light(2)	1	1	1	1
Ramp max delay	5000	5000	1200	5000
Ramp exponent	1	1	1	1
Increasing cue (1) or not(0)	0	0	0	0
Delay to deliver the second cue if t...	1000	1000	1000	1000
Second cue type: sound(1) light(2)...	1	1	1	1
Second cue frequency	12	3	3	12
Second cue speaker number	1	2	2	1
Second cue light number	1	1	1	1

Figure 13 Interval timing task parameter setting example

Variable	CS1	CS2	CS3	CS4
Number of trials	80	20	0	0
Frequency(kHz)	12	3	5	0
Predicted solenoid	5+3	5+3	1+3	1+3
Probability of solenoid	0+100	0+0	0+0	0+0
Solenoid open time (ms)	0+60	0+60	0+30	0+40
Cue duration (ms)	1500	1500	1000	1000
Delay to solenoid (ms)	0+1500	0+1500	0+3000	0+3000
Pulse tone (1) or not (0)	0	0	1	0
Speaker number	1	2	2	2
Light number	1	2	1	2
Go lick requirement	0	0	0	0
Go lick tube (or solenoid)	1	1	3	1
Sound(1), light(2)	1	1	1	1
Ramp max delay	5000	5000	1200	5000
Ramp exponent	1	1	1	1
Increasing cue (1) or not(0)	0	0	0	0
Delay to deliver the second cue if t...	0	0	0	0
Second cue type: sound(1) light(2)...	0	0	0	0
Second cue frequency	5	5	5	5
Second cue speaker number	1	1	2	2
Second cue light number	1	1	1	1

Figure 14 Ramp timing task parameter setting example

Choice tasks

For the choice task shown in the paper, animals can be presented with both operant lick tubes (left/right) for the duration of the task, with different associated reward requirements or properties. For instance, each side can have different magnitudes of rewards with different delays (delay discounting), probabilities (two-armed bandit or probability discounting), or ratio requirements (effort “discounting”). It is also possible to give choices between different types of rewards of differing magnitudes to assay subjective reward preference.

Importantly, during this experiment, the animals may sometimes develop a side preference for the left or right lick tube. If so, the buttons “Clear lick 1 rewards” or “Clear lick 2 rewards” are helpful for shaping. If these buttons are engaged, licks on the corresponding lick tube (we set lick tube 1 as the left lick tube and 2 as the right) never result in reward even if the operant criterion is met. This allows the user to rapidly adjust the task to mitigate any side preference. It is also possible to counterbalance which side the higher reward is delivered. This is controllable using the “Fixed side check” option.

These parameters are set in the sections of the GUI shown below.

Clear lick 1 rewards

Clear lick 2 rewards

Variable	Lick tube 1	Lick tube 2
Number of licks requi...	5	5
Fixed/variable check	0	0
Predicted solenoid	3	3
Probability of solenoid	100	100
Solenoid open time (...)	50	20
Delay to solenoid (ms)	0	0
Delay to next lick (ms)	100	100
Fixed/variable check	0	0
Min number of rewar...	40	20
Sound(1), light(2) or ...	1	2
Pulse tone (1) or not ...	0	1
Sound Frequency (k...	12	3
Sound Duration (ms)	1000	1000
Speaker number	1	2
Light number	1	1
Fixed side check	0	0

Matched to sample choice tasks

Matched to sample choice task utilizes both cues within each trial for the animal to evaluate the identity of the two cues. If the second cue’s identity matches the first cue’s identity, animal is supposed to lick on one side of retraction lick tube (e.g., left) to obtain

a reward—this lick tube is set in the “Go lick tube (or solenoid)” parameter, 1=left lick tube, 2=right lick tube; on the contrary, if the second cue doesn’t match the first cue (light vs sound or two sound cues with different frequencies), animal needs to lick on the other side of retraction lick tube (e.g., right) to obtain the reward. Side retraction solenoids are extended upon the onset of second cue and last for a set duration for the animal to decide within this time. Delay to those solenoids is set to equal the delay from first cue to second cue. First lick made on either side lick tubes is registered as the animal’s choice. Upon first lick, both side retraction lick tubes are retracted back. Reward is given immediately through the middle lick tube if the animal made the correct choice and no reward if otherwise. If no lick is made during the length of side lick tube extension, side lick tubes will be retracted back once time passes the duration. No reward will be given in this case and trial will end to enter the inter-trial interval.

These parameters are set in the sections of the GUI shown below.

Variable	CS1	CS2	CS3	CS4
Number of trials	25	25	25	25
Frequency(kHz)	12	3	12	3
Predicted solenoid	56+3	56+3	56+3	56+3
Probability of solenoid	100+100	100+100	100+100	100+100
Solenoid open time (ms)	3000+40	3000+40	3000+40	3000+40
Cue duration (ms)	1000	1000	1000	1000
Delay to solenoid (ms)	1000+4000	1000+4000	1000+4000	1000+4000
Pulse tone (1) or not (0)	0	0	0	0
Speaker number	1	2	1	2
Light number	1	2	1	2
Go lick requirement	1	1	1	1
Go lick tube (or solenoid)	1	1	2	2
Sound(1), light(2)	1	1	1	1
Ramp max delay	5000	5000	1200	5000
Ramp exponent	1	1	1	1
Increasing cue (1) or not(0)	0	0	0	0
Delay to deliver the second cue if t...	1000	1000	1000	1000
Second cue type: sound(1) light(2)...	1	1	1	1
Second cue frequency	12	3	3	12
Second cue speaker number	1	2	2	1
Second cue light number	1	1	1	1

Matched-to-sample trial

Unmatched-to-sample trial

Possible hardware modifications

Currently, our hardware is set up to perform behavioral experiments in head-fixed mice. Nevertheless, since our software is agnostic to the exact hardware connected to each Arduino pin, it is possible to change any of the hardware inputs/outputs. For instance, the side lick tubes can be swapped out with levers that can send in a TTL input for

every lever press to the Arduino. Another possibility is that these inputs could correspond to infrared beam breaks corresponding to nose poke entries in freely moving animals. Similarly, the solenoid control outputs that deliver rewards or retract lick tubes can be swapped with TTL-controlled motor/solenoid mechanisms that deliver solid or liquid rewards, or control retraction of levers respectively. Similarly, the TTL outputs for turning on the lights can be connected to deliver other types of sensory stimuli that take TTL inputs for turning on/off. If more complex stimuli are needed, one possibility is to control them with a separate microcontroller and have them turn on/off based on a TTL input. In this case, one would need to synchronize the clocks of the two microcontrollers. That can be done using a similar approach as shown in Figure 3E. In all these cases, the text on the GUI will need to be modified to match the external device. For instance, if the light is swapped with a smell, the text in the GUI may need to be switched for consistency. This can be done by opening the GUI file ("headfix_GUI.fig") in GUIDE and changing the corresponding texts. Please note that in the future, we are planning to port our code away from GUIDE for better compatibility across MATLAB versions.