Tripoke Manual

Feb, 2015

Nathan Insel

Contents:

I. Summary

The Tripoke task was originally designed to train mice to orient toward and approach specific sound and light cues. The name "tripoke" comes from the use of a triangular arrangement of nosepoke ports. Like many other conditioning paradigms, it relies on Pavlovian ("classical") conditioning methods; i.e., pairing an arbitrary, conditioned stimulus with an intrinsically valued, unconditioned stimulus (liquid food reward) to create an unconditioned response. There is also an element of instrumental conditioning, as the natural behavior of the animals toward a food source is refined to optimize capture of the liquid. The software running the task is capable of training mice on place-, response-, and sequence-learning tasks, as well as working-memory paradigms (by using sequences with repeated-elements). Although not presently implemented, the effects of conflict could be examined by presenting light and sound cues from different ports, or pitting cue- and spatial-contingencies against one another. A future version may also incorporate an odor delivery system, to add dynamically-changing contextual cues. The task is similar to rat-based "y-mazes", allowing greater freedom of movement at the cost of a defined "decision point." Although the current version uses a chamber with walls, the task could also be set-up on a platform, to encourage spatial strategies with distal landmarks.

II. Contributions and Updates

Feb '15, v1.0: Original task & document (NI)

(Add dates and names for future updates here)

III. Overview of System components

The experimental apparatus and its control software can be described as a combination of 5 main components:

1) Conditioning chamber (Section V-A)
   A hexagon with 3 long sides and 3 half-length sides, nosepoke holes (with IR sensors and feeder spouts) are located in the center of the half-length sides, as are small LED lights for visual cue presentation and magnetic speakers for audio presentation.
2) Electronics linking the chamber and Arduino (Section V-B)
   Includes cabling as well as a circuit board for blinking the LEDs, an Arduino Uno for controlling sounds, and an optical isolator (relay) for controlling the feeder solenoids.
3) Arduino (Section V-C & VI-A)
   We use the Arduino Mega, loaded with the sketch_tripoke code.
4) Computer running MATLAB (Section V-D & VI-B)
   We currently use a Linux, Ubuntu OS, but the system works equally well with Windows. The version of MATLAB we use is R2014B. The computer is connected to the Arduino via a USB port (set as a virtual serial port)

5) Camera & Camera recording system (Section V-E & VI-C)
We have used a USB2 camera and a GigE, Allied Vision Technologies camera
(2/15: both still lack proper acquisition software)

We have also begun developing additional components for future instantiations of that apparatus:
F1) Odor delivery system (Section VII-A)
F2) Open-Ephys Data Acquisition Board (DAQ; Section VII-B)

IV. Running Tripoke

The software interface for the Tripoke chamber is displayed in Figure 1.

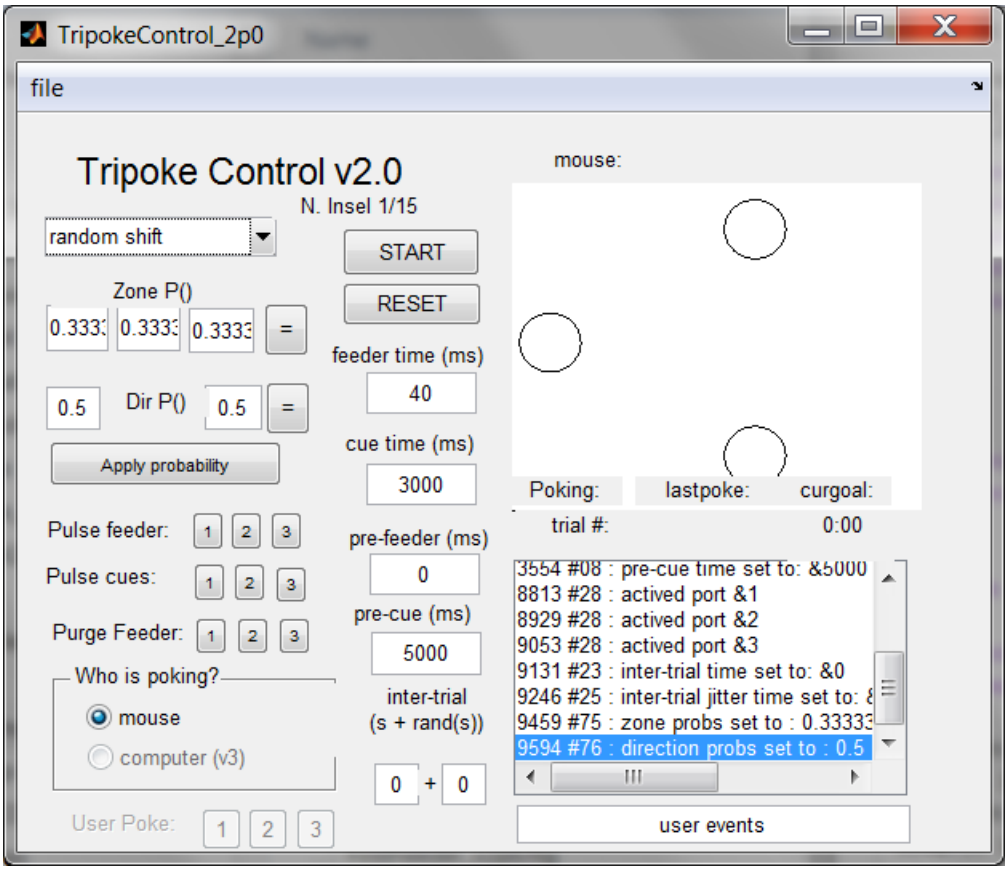


Figure 1: GUI interface for Tripoke. This image shows the task after "random shift" has been selected from the "select task" drop-down menu (upper left, under task title). Selecting this option reveals the Zone P() and Dir P() edit boxes (below menu) where the user can adjust the probability of zones and directions that become active from one trial to the next. Feeders and cues can be controlled manually using the "Pulse feeder", "Pulse cues", and "Purge Feeder" buttons (below the probability selection box). Task timers can be adjusted by using the feeder time, cue time, pre-feeder, pre-cue, and inter-trial edit boxes (middle of the window, under the START and RESET buttons). When ready to start, the user can click the "START" button (top middle), which will then change color, begin a trial counter and session timer (located in the middle right, below the axis), and initiate the session. Activated zones are displayed in the white, axis panel that shows three circles (upper right), timestamped events from the Arduino are displayed below. The user can also input notes using the user events edit box (bottom

right). Not shown in this image are settings associated with other tasks; e.g., if the user selects the sequence task, an edit box will appear that allows the user to explicitly write out a list of zones that will be activated in a repeated sequence across trials. Also not shown are file menu options: "edit mouse info," allowing the user to change the mouse and exposure number that is set upon starting the session (if inputted incorrectly the first time), "Load Parameters," allowing the user to load a previously-save set of task parameters, such as zone probabilities and timer times, and "Save Parameters" allowing the user to save the parameters selected on a previous day.

A. Order of operations:

- MATLAB is opened
- "Tripoke" is entered at the MATLAB command prompt, running the script that initiates the control software (e.g., TripokeControl_1p9)
- If the computer is not properly connected to the Arduino, the user will receive an error and a dialog box explaining possible ways to fix the problem
- To load pre-set tasks and values, the user clicks the "File" menu and selects "Load pre-set" Otherwise…
    - A task is selected by clicking on the Select Task menu. The "SinglePort" task is typically used for pre-training on the cues, the "RandShift" task is the primary Tripoke task, the "Sequence" task allows the user to specify a sequence of places. These are described in more detail in the next section.
    - When "RandShift" is selected, several editable text windows appear that allow the user to modify the zone probabilities (Zone P() boxes) and the direction probabilities (Dir P() boxes). If these values are changed, then the "Apply probability" button must be clicked.
    - Parameters are checked or edited (e.g., feeder time, cue time, etc.)
- Feeders are typically "primed" (liquid food reward allowed to flow to the bottom of the feeders) by clicking and then unclicking the "Purge feeder" buttons one at a time.
- The Start button is clicked. This will open a dialog box where the mouse number and mouse exposure number are entered.
- The mouse will begin. The program will keep track of both time and trial number on the bottom of the screen.
- When the session is over (typically 15 minutes), the "stop" button (formerly start button) is pressed.
- The start button is again clicked to initiate the next mouse's session. This is continued until all mice have completed.
- When all mice have completed the task, feeders are cleaned by clicking "feeder purge" and filling feeder syringes with warm water. (Pressure-cleaning can also be done using a syringe plunger.)
- The software maintains an autosave throughout the task, and performs a final save when the software is closed.

B. Tasks

Single Port:

Cues are presented, typically at the same time that the nosepoke becomes active (i.e., pre-cue = 0). The mouse can nosepoke at any time from cue initiation to receive liquid food reward. A new trial will not initiate until the inter-trial interval is complete (set

with the two inter-trial edit boxes, one for fixed interval, one for the additional jitter interval (random time selected from a uniform distribution between 0-X number of seconds).

RandShift:

The primary Tripoke task. Cues are presented from a zone, the mouse pokes at that zone or an alternative. If poking at the cued zone, mouse receives food reward; if poking at an alternative, the trial completes without reward delivery (no error cue is currently given). Whichever port the mouse pokes is deactivated for the next trial, and one of the two other ports becomes active. The "pre-cue" parameter (modified through the edit box) determines the time between trial initiation and cue presentation. Mice responding before the cue can be said to be guessing, following memory of spaces, or following an action "habit."

Sequence:

Allows the user to enter a sequence of zones that will be activated. Repeated experience with the sequence should allow mice to quickly follow the zone transitions. Repeated elements in the sequence allow working memory to be tested.

V. Hardware details

A. Conditioning chamber

i. Geometry

A hexagon with 3, 20 cm sides and 3, 10 cm sides. Nosepoke holes are located about 0.8 cm from the bottom of the chamber in the center of each 10 cm wall. Figure 2 A & B show an outline of the apparatus.

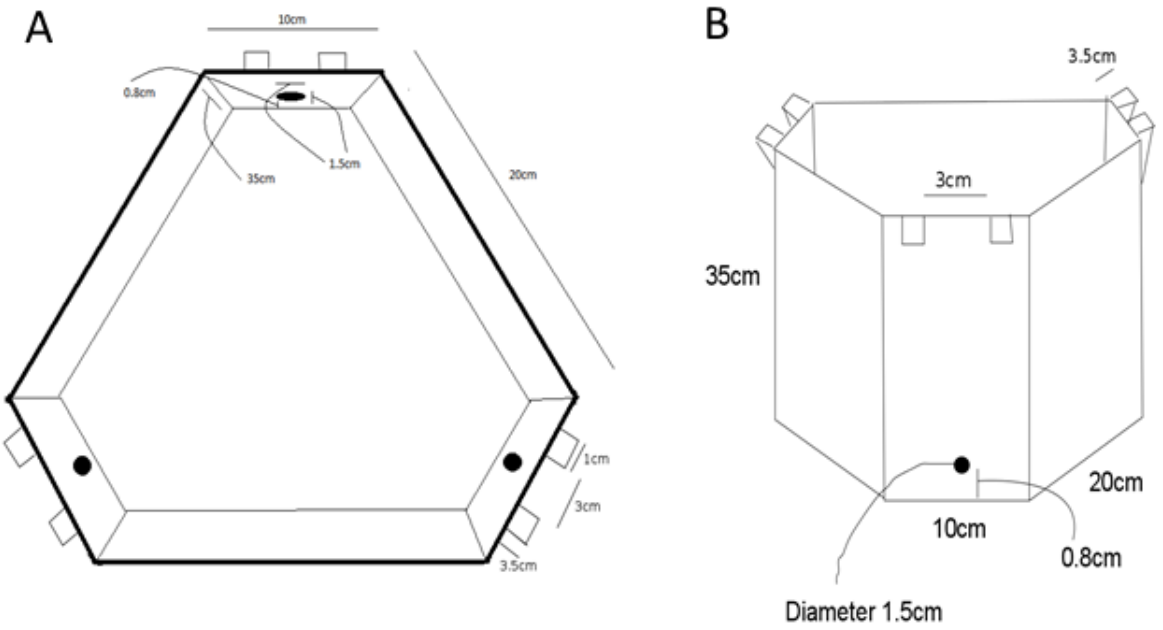


Figure 2. Diagram of conditioning chamber. As described in text, the conditioning chamber is a hexagon with a nosepoke hole in each of the three narrow walls. The two squares protruding from the top of the narrow walls are for hanging the feeder syringes.

ii. Lights

LED lights are bright blue, positioned just above the nosepoke hole, through a small, drilled hole

iii. Speakers

Magnetic (8 Ohm .3 W, 87DB 15mm; digikey: 668-1132-ND, manufacturer: AST-01508MR-R) positioned further above the nosepoke holes, emitting sounds through several drilled holes in the chamber wall.

iv. IR sensors

1 cm width (photointereupter otpic slot 10mm pcb digikey: 425-1925-5-ND, manufacturer: GP1A57HRJ00F)

v. Feeders

Comprised of several parts:

- The solenoid valve (currently using ASCO mini valve 1/8 12 DC U89256A4V12DC ordered from Tubefit)
- Solenoid tube connectors (Hose barbe 1/4" x 1/8 MNPT bag of 10 PBF1BAGOF10 from Tubefit)
- Tubing (1/8 inner diameter, tygon tubing (vinyl from the hardware store also fine))
- A 60 mL syringe
- A nozzle (Hamlet STST HOSE BARB TUB STUB 130LTSS0202 from Tubefit)

vi. Nosepoke plug

- Sensor and feeder nozzle fit into nosepoke plug: a plastic, 3-D printed, cylinder-shaped part that fits into the 1.5 cm nosepoke hole in the chamber wall (Figure 3).

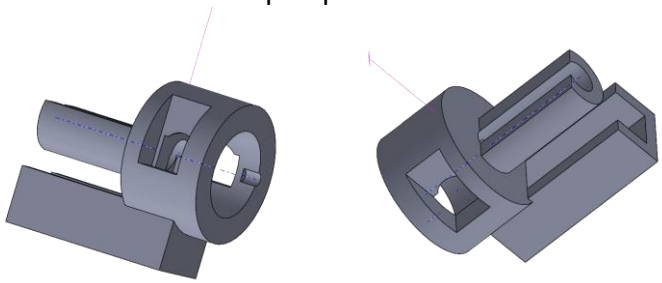
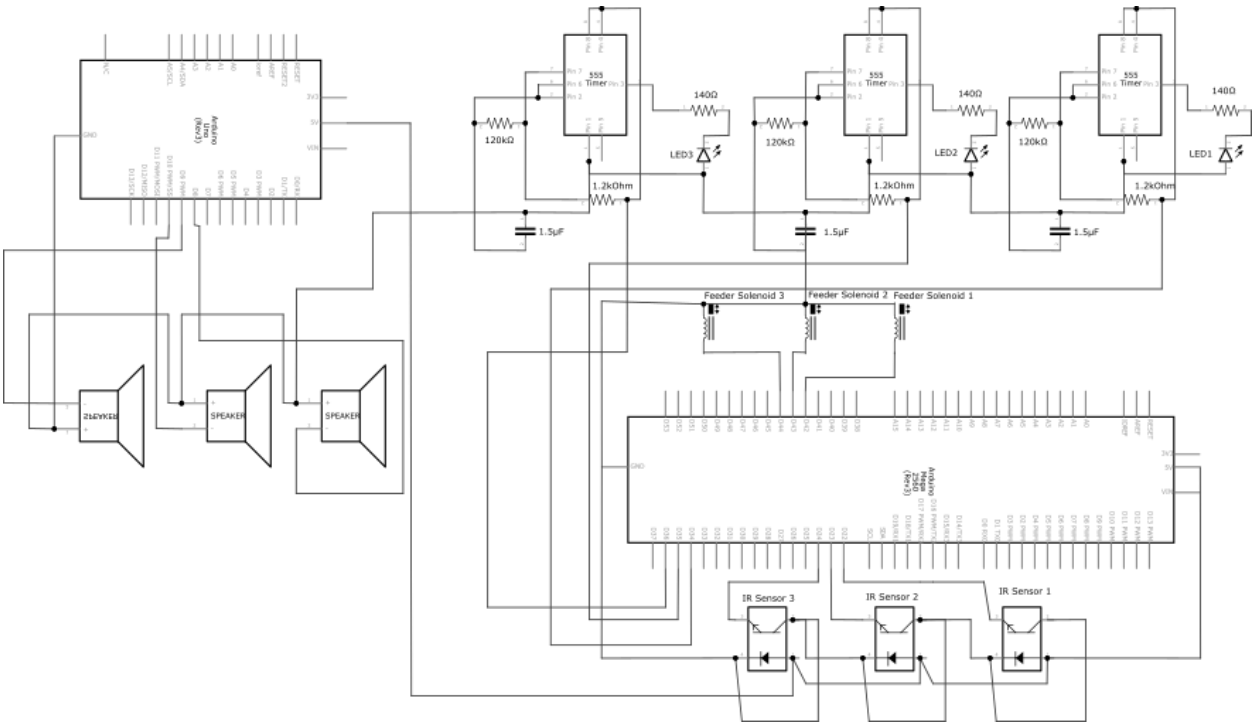


Figure 3. Nosepoke plug. Left and right: two angle views of the nosepoke plug that fits into the 1.5 cm diameter hole of the chamber wall and holds both the IR sensor and feeder nozzle. The larger cylinder fits into wall of the chamber, the slots are spaced to allow the U-shaped, IR sensor to slide in and detect beam-breaks within the slot. The elongated, narrow cylinder is shaped so that the feeder nozzle can be angled into the nosepoke hole to allow mice to drink liquid food reward. The rectangular slot on the bottom helps capture extra reward liquid.

B. Electrical connections

All electrical connections are described by Figure 3.



<u>Figure 4</u>. Tripoke electronic connections. The Arduino mega is depicted in the lower right corner (large rectangular box above the three small boxes depicting the IR sensors). 555 timer circuits to control the blink of the LEDs are located in the upper right. The sounds are controlled by an Arduino Uno (upper left box), which signals to three speakers (lower left). The feeder solenoids (middle right) require an optical relay (not shown) that converts the 5 V TTL signal from the Arduino into a 12 V pulse.

i. LEDs

On the electronics board is a solderless breadboard that contains an array of 555 timers configured to generate ~4 Hz blinks when receiving 5 V signals from the Arduino. An additional resistor (currently 50 Ohm) added in serial to circuit limits LED brightness.

ii. Speakers:

Control of sounds is offloaded from the main Arduino (Mega) board to a second (Uno) board. The software loaded on the Uno board currently converts a ttl signal from the Arduino mega into a 7 kHz tone at very low (still unmeasured) decibel level, achieved by adding a 1000 Ohm resistor to the output of the Uno.

iii. IR sensors:

Sensors are connected to 5 V, ground, and digital input pins on the Arduino Mega.

iv. Feeder:

Each solenoid is wired to a Greyhill, optical isolator relay (mounted on a Greyhill board). The isolators are connected to a 12 V power supply. The particular modules used are closed when inputs are ON, so Arduino digital output pins are set to 5V except when feeders are signalled to open.

C. Arduino

Arduino Mega microcontroller programmed with “sketch_schema” (e.g., sketch_schema_1p8). Serves as an interface between the computer and the chamber devices. Wires are connected

into the pin slots of the Arduino and connect to the electronics as described above. The specific pin numbers used as inputs and outputs are described in the software code.

D. Computer

No particular hardware specifications should be necessary for the computer.
We use a Linux OS, but the software is designed to be compatible with Windows.

VI. Software code details

A. sketch_schema (Arduino)

i. Inputs, outputs, timers

- The list of all pins used can be found at the beginning of the sketch_schema. E.g.:
  # define FEEDER_PIN_1 43
- The list of all MATLAB input signals can be found near the beginning of the TripokeControl code. E.g.,:
  #define CMD_START_TASK 2
- Besides providing an interface between computer and electronics, the Arduino also maintains several millisecond-precision timers. The timers are as follows:
  cues – turns off cues when time has elapsed
  precue – turns on cues when time has elapsed
  feeder – turns off feeder
  prefeeder – turns on feeder
  intertrial – initiates the pre-cue timer and activates nosepokes
  synchcues – turns on and off the LED for video synchronization (currently unused)
  reserved for future versions:
  odor – turns off odor delivery system

ii. Sequence of events

- The Arduino operates by continuously looping through the "main" function.
  1. The serial buffer is checked for new inputs, responding according to the command
  2. Each timer is checked.
  3. If nosepokes are active, the state of each sensor is checked. If there has been a change in the state, an "inp" signal is sent to the computer.

iv. Nosepoke conditionals

All nosepoke ports are read if the nosepokes are active, "if" conditionals only take place for those specific ports that are active.
- If the particular sensor detecting a poke ("nosepoked") matches the goal port ("nosepokeByte"), the pre-feeder time is begun.
- If the particular sensor detecting a poke does not match, an error is signaled, and a new trial initiated.
- If the detecting sensor stops detecting a poke during the pre-feeder time ("fixation break"), the pre-feeder timer is reset.
- If the feeder timer completes, or an error trial is registered, a "NEWTRIAL" signal is sent to the computer, requesting signals to set the identity of the next cues, feeders, active ports, and a new trial command.

B. TripokeControl (MATLAB)

i. relevant classes

1) hObject and the handles struct

MATLAB's GUIDE, GUI-builder program uses a handles structure to keep the handles and of all GUI components, along with any additional information that has to pass between functions. It is used in the place of a long list of global variables. Changes in handles can be saved and retrieved using the function guidata.

2) timer

The TripokeControl software uses two timers. The "mainloop" function is set as a timer "ml" that repeats every 20 ms. The actions of mainloop are described below in Section VI. B. ii

3) schema_arduino

The MATLAB interface to the Arduino hardware. The "schemauino" instance of the schema_arduino sets-up a child, serial object that communicates with the Arduino. It then contains a series of protocols for sending information from the TripokeControl GUI to the Arduino and for storing information received from the Arduino, including current and previously activated sensors, and whether a new trial flag has been received.

4) serial

Child of schema_arduino for communication over the serial port.

ii. sequence of events

TripokeControl is built using GUIDE.

- The opening function sets all Arduino parameters, and sets-up the gui handles structure to hold all parameters and gui data
- The primary loop is a timer object called "mainloop". This provokes the schema_arduino object to read the serial buffer (using is child serial object) and set parameters. Depending on the state of the task, it will also read the current nosepoke status and whether there is a new trial flag, both properties of the schemauino instance of the schema_arduino class. Finally, it will load and record inputs from the Arduino and initiate the autosave function (set to every 20 s)
- If the new trial flag is active, the setnewtrial function is called. The process by which a new goal zone is selected, and which ports are inactivated, cued, etc, is dependent on the specific task selected in the task drop-down menu.

VII. Future development components

A. Odor delivery system

B. DAQ

www.open-ephys.org/acq-board/

Used for synchronization. Receives input signals from the Arduino in binary