

Precise Dispenser for Canine Numerical Discrimination Tasks
20 September 2021
V0.4

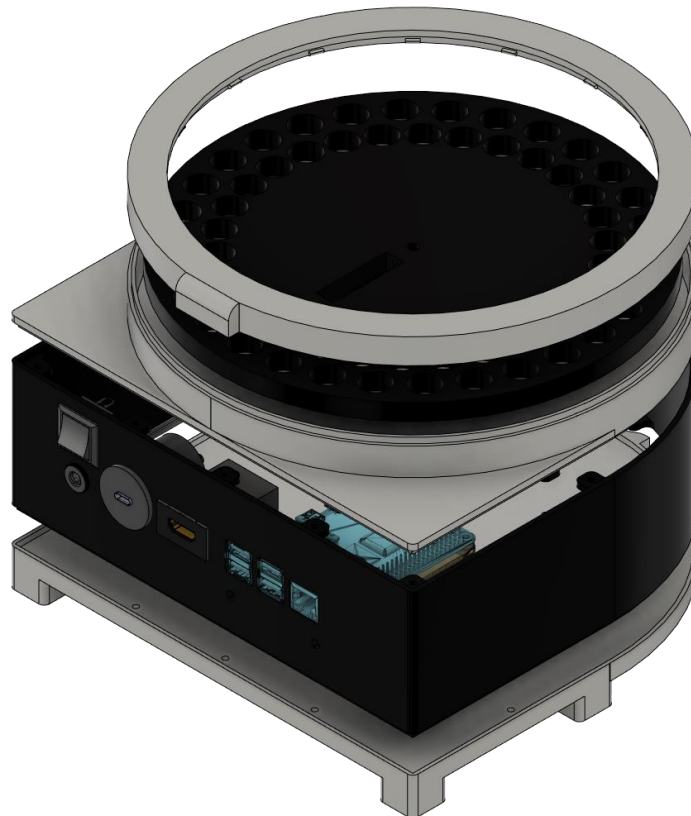


Figure 1 – Dispenser v84 Render

Overview

The performance of canine numerical discrimination tasks can be hindered by a lack of appropriate instrumentation. Using manual, researcher-controlled instrumentation can introduce a reaction time variable that could impose undue variability in the task results. Therefore, the creation of devices that can automatically dispense precise numbers of treats at the introduction of a predefined stimulus can be justified.

Dispenser Operation

The precise dispenser has a wheel on the top of it that holds up to 59 treats. The stepper motor shaft interfaces to the dispenser wheel through the center hole and by actuating the motor, the holes for the treat lines up with the treat exhaust and falls to the bowl below. When the treat goes through the exhaust it breaks an infrared break beam and the Raspberry Pi detects the treat dispensing.



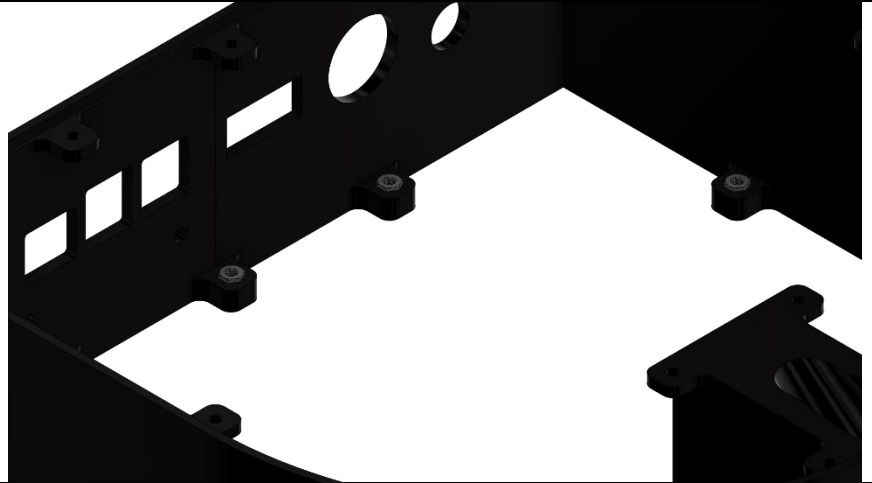
Figure 2 – Images of the assembled dispenser, fully loaded with treats



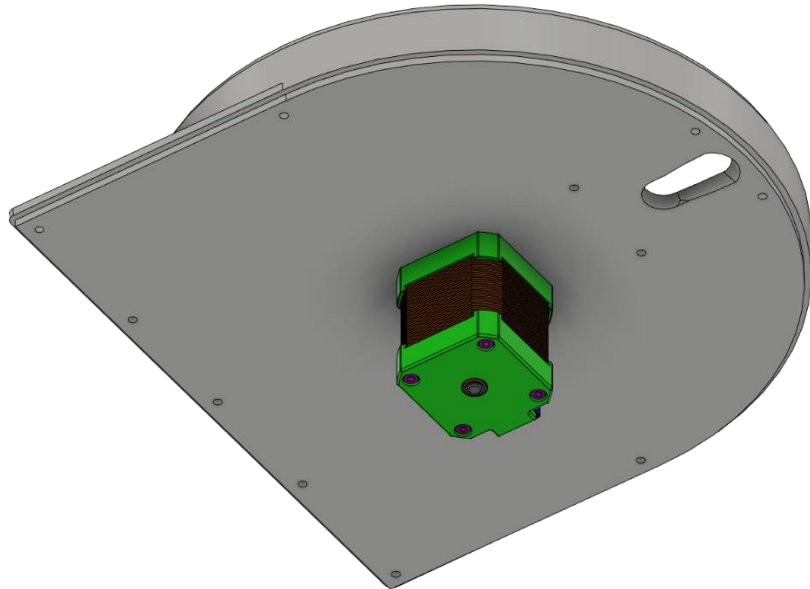
Figure 3 – Assembled dispenser connector backplane

Build Instructions

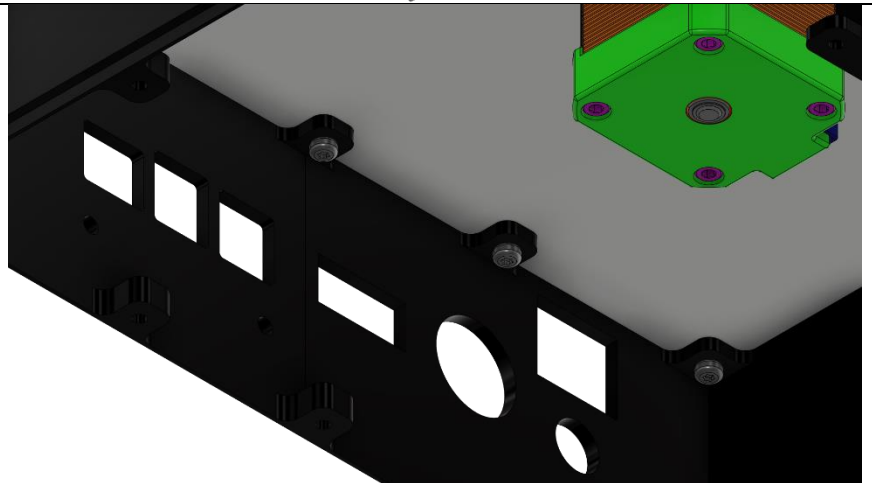
Observe the M3 hex nut capture slots on the dispenser walls. In each of these, super glue an M3 hex nut. Let it sit for 24 hours before continuing.



Mount the Nema 17 stepper motor using pan head M3 screws. Ensure the connector is point towards the right side, as shown.



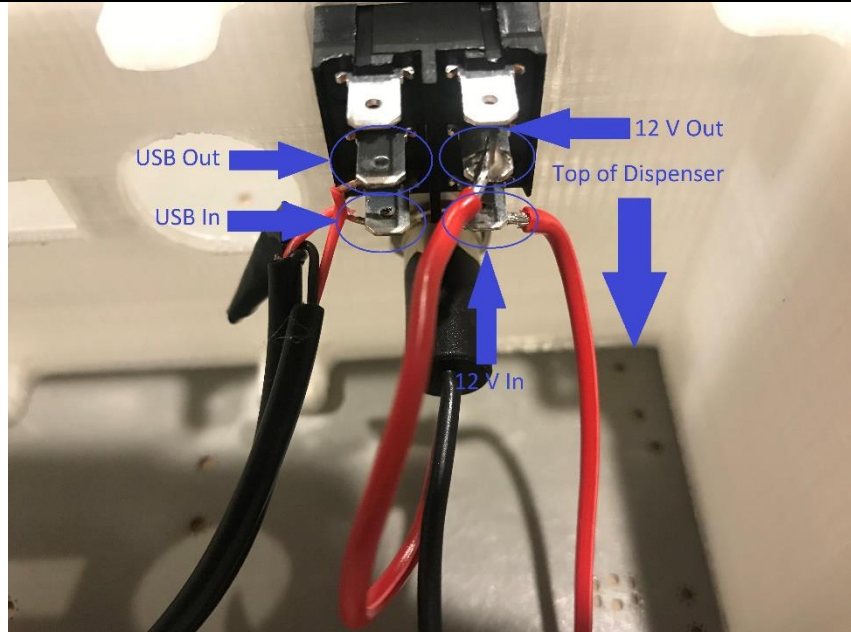
At this point, use 6mm M3 screws to mount the walls to the top of the dispenser. Mount the dispenser jogger by press fitting it onto the stepper motor shaft. Ensure the motor is supported during this, as it will break through the print when pressing the jogger down.



Now, mount the Raspberry Pi to the right angle mount using four M3 screws. Then, mount this into the walls using a 10mm M3 screw with hex nuts. Then mount the other panel mount connectors into the walls.



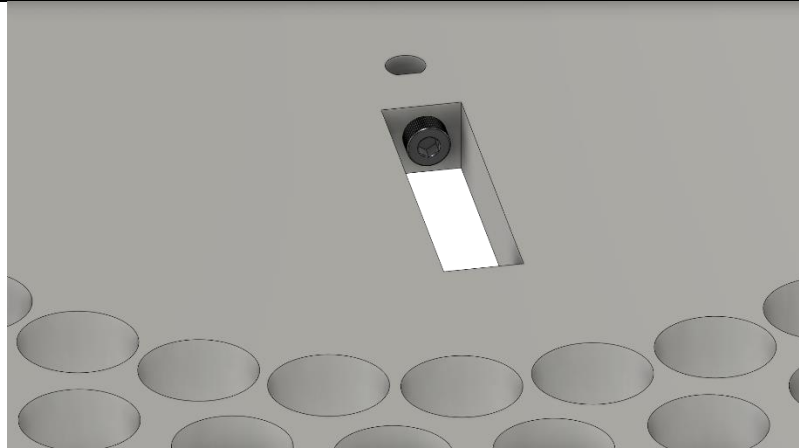
With the DPDT switch mounted, cut a USB C to USB C cord and solder the grounds back together. Cut the barrel jack panel mount connector's red wire, with one fifth of it being close to the connector and the other four fifths being used to connect to the Pi HAT. Solder according to image.



Mount the IR phototransistor elements to the side of the pellet exhaust using 9mm M3 screws.

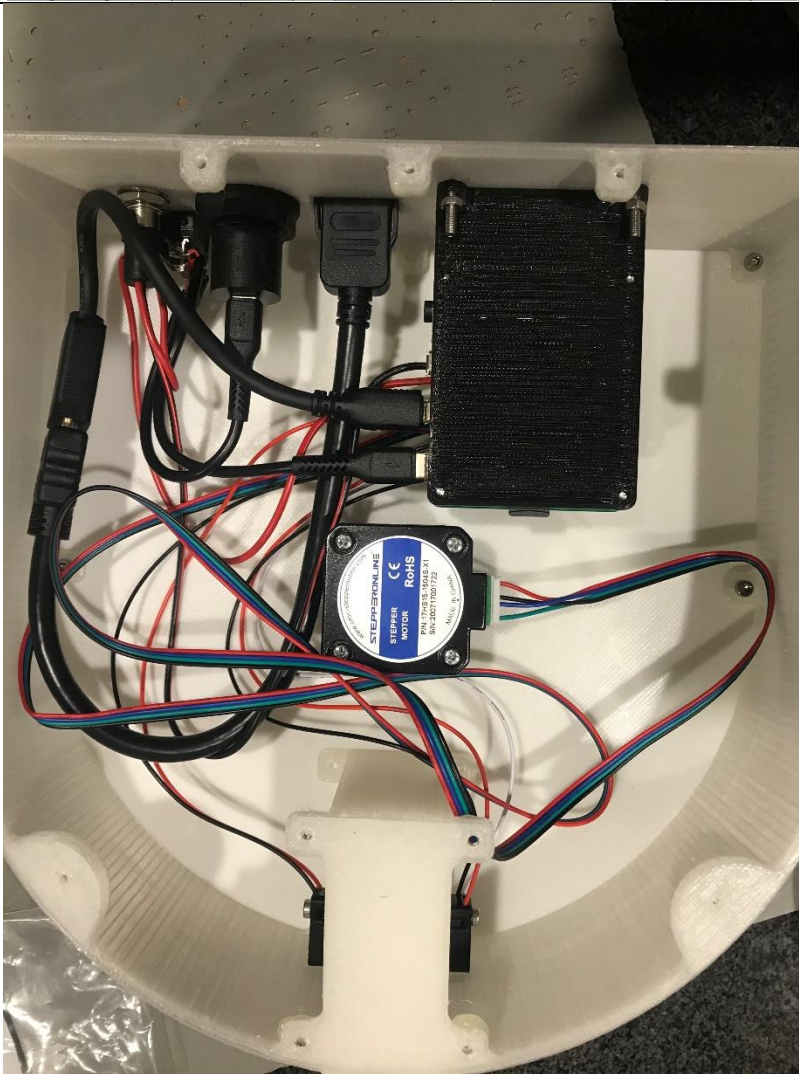


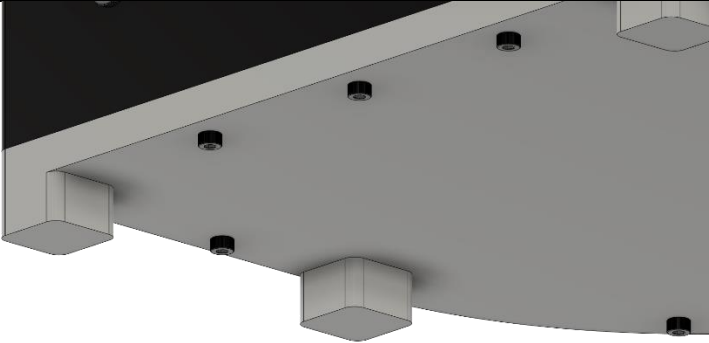
Place a 12mm M3 in the set screw hole of the treat jogger and screw it down to hold the jogger to the motor shaft.



Wire up the remaining pieces as shown below.

Secure the wires using the square zip tie mounts.



<p>Screw on the bottom plate using 6mm M3 screws.</p> <p>It is recommended to use rubber anti slip pads on the legs of the dispenser.</p> <p>Place the acrylic disk into the cover and place the cover onto the dispenser.</p>	
--	--

Testing Operation

1. Place treats into all the holes of the treat jogger and manually rotate the wheel to ensure all treats fall out of the holes. If some stick, then make sure any strands are removed from the print.
2. The `dispenser_test.py` script can be used to ensure the break beam works and that treats are dispensed as expected.

Troubleshooting

1. If the dispenser under dispenses, ensure the treat jogger is mounted securely to the motor shaft. If the set screw strips the hole, use super glue to affix the wheel to the shaft.
2. If the dispenser over dispenses, ensure that the jogger is perfectly aligned over the pellet exhaust before use.

Python Class Definition

```
class PreciseDispenser
    self.loaded_treats
        The number of treats remaining the in the jogger
    self.dispensing_timeout
        The amount of time between IR break beam events before an exception is
        thrown
    self.dispensing
        A Boolean to track if the dispenser is currently dispensing
    __init__(loaded_treats, timeout=1)
        Initializer for the canine treat dispenser, configures control variables
        and Pi GPIO.
        :param loaded_treats: The number of treats that are loaded into the
        dispenser for this session. Defaults to max, 59 treats.
        :param timeout: The amount of time before the dispenser throw an error
        between dispensation events
    dispense_treat()
        Function to dispense a single treat. Should not be used, use
        dispense_treats for error handling.
        :returns: True for successful dispensation of a single treat, False for a
        failure to dispense the treat.
    dispense_treats(num_treats)
        Dispenses the specified number of treats. If the requested number of
        treats is greater than the remaining treats, a ValueError is thrown. If
        the treat dispenser loop times out, a ValueError is thrown.
        :param num_treats: A number between 1 and the remaining number of treats.
        :returns: None. Throws ValueError when an error event is encountered.
    treat_dispersed(channel):
        A callback function for when a treat is dispensed. This is detected on
        the falling edge of the IR break beam.
        :param channel: The channel to identify the callback source.
        :returns: None.
    close()
        Closes the dispenser object and cleans up the GPIO assignments.
        :returns: None.
```

Interfacing Example

```
from precise_dispenser_driver import PreciseDispenser

dispenser = PreciseDispenser()
try:
    dispenser.dispense_treats(10)
    print("Successfully dispensed ten treats.")
except ValueError as e:
    print(e)
dispenser.close()
```

Raspberry Pi 4 as an Experiment Platform

The experiments on canine operant conditioning are run using a Python package called PsychoPy. It essentially acts as a development environment for psychology experiments and provides a variety of utilities for user interface development, trigger conditions, experiment logging, and so on. Our aim is to integrate this directly into a Raspberry Pi 4, to allow the entire setup to become a mobile testing unit. To do this, we will have a few requirements:

1. Wireless Access Point
 - a. The Raspberry Pi 4 should host its own WiFi access point (AP), which allows the experimenter to connect to the Pi and run experiments, download results, and interact with the main system.
2. Decoupled Interfaces
 - a. The screen the experimenter will use, and the screen used by the canine will be separate and require no extra hardware to achieve. By connecting to the Pi using SSH (Secure Shell), the command line can be exposed to run the experiments. The canine screen can be connected to the HDMI0 of the Pi and the SSH session can be started in a separate area.
3. Simple Setup
 - a. We want to make sure this is easy to setup and maintain. This means that we would want to use robust packages such as ‘pi-ap’ and SD card backups to easily restore the device and get new devices up and running easily.

There are two ways to setup a new Raspberry Pi to operate the canine dispenser:

1. Flash a new SD card with an image from the iso_images folder
 - a. Using a Raspberry Pi with a fresh installation of Raspbian Buster, there are built-in tools to achieve [this](#). “The SD Card Copier application, which can be found on the Accessories menu of the Raspberry Pi Desktop, will copy Raspbian from one card to another. To use it, you will need a USB SD card writer. To back up your existing Raspbian installation, put a blank SD card in your USB card writer and plug it into your Pi, and then launch SD Card Copier. In the ‘Copy From Device’ box, select the internal SD Card. This could have a number of different names and may have something like (/dev/mmcblk0) in its entry, but will usually be the first item in the list. Then select the USB card writer in the ‘Copy To Device’ box (where it will probably be the only device listed). Press ‘Start’. The copy, depending on the size of the SD card, can take ten or fifteen minutes, and when complete you should have a clone of your current installation on the new SD card. You can test it by putting the newly-copied card into the Pi’s SD card slot and booting it; it should boot and look exactly the same as your original installation, with all your data and applications intact.”
2. Follow the proceeding instructions to setup the Raspberry Pi manually.
 - a. Flash a new copy of Raspbian Buster onto a micro SD card and place an empty file named “ssh” onto the boot partition. There is no file extension.
 - b. Connect the Raspberry Pi to your local network using an Ethernet cable and power up the Pi. You can connect a monitor, keyboard, and mouse and use `ifconfig` to find the IP address of the Pi.
 - c. Using this IP address, go to PuTTY and type in this IP address and connect to the Pi. The default password for “pi” is “raspberry”.
 - d. Once in, the [pi-ap software](#) can be installed to create the access point. Note, the dog_operant repository does contain a copy of this, but the main repository will contain the most recent version.
 - e. The [dog_operant](#) repository can be cloned using the `git clone` command.
 - f. Run the command `sudo apt-get update`

- g. You will need to upgrade pip using

```
python3 -m pip install --upgrade pip
```

which will allow it to index some of the following packages.
- h. Run the command

```
python3 -m pip install psychopy==3.2.4
```

which will install all the needed packages. Using the older version of PsychoPy will prevent a 'Segmentation fault' caused when rendering a new Window.
- i. You may have errors from installing PyQt5, to remedy this, try the following:

```
sudo apt-get update
sudo apt-get install qt5-default pyqt5-dev pyqt5-dev-tools
```
- j. You may have errors importing Pandas due to an Import Error from NumPy, this can be fixed by doing:

```
sudo apt-get install libatlas-base-dev
```
- k. You may get an error saying that there is no package called 'wx', in which case you need to run the command

```
python3 -m pip install wxPython
```
- l. Step k may also error out, so issue the following command and then perform step k again:

```
sudo apt-get install dpkg-dev build-essential libjpeg-dev libtiff-dev
libsdl1.2-dev libgstreamer-plugins-base0.10-dev libnotify-dev freeglut3
freeglut3-dev libwebkitgtk-dev libghc-gtk3-dev libwxgtk3.0-gtk3-dev
```
- m. To install the audio tools, issue the following commands:

```
sudo apt-get install portaudio19-dev
python3 -m pip install pysine
```
- n. To install the serial tools, issue the following command:

```
python3 -m pip install pyserial
```
- o. If you want to run experiments, you will need to connect a screen to the Pi on HDMI0. An issue may present itself where no screen is presented when connected, which is caused by an incorrect /boot/config.txt file. To fix this do the following:
 - i. `sudo nano /boot/config.txt`
 - ii. Add the following to the file or uncomment from the file,
 - 1. `hdmi_force_hotplug=1`
 - 2. `hdmi_drive=2`
 - iii. If this does not fix the issue, add,
 - 1. `hdmi_safe=1`
- p. To run an experiment, use the following command

```
export DISPLAY=:0 ; python3 dog_operant/system_test.py
```

Note, this works from the command line, so run this when you are connected to an SSH session and with a screen connected to HDMI0 of the Pi and the experiment will display over HDMI0. Setting the DISPLAY variable to 0 will default it to the local display.
- 3. To transfer the results of the experiments, copy the contents of the data directory from the dog_operant directory. Use the following command structure:

```
scp -r user@ssh.example.com:/path/to/remote/source /path/to/local/destination
```

Raspberry Pi Interfacing and Passwords

The dispenser, when turned on, will create a WiFi access point called, “cchil-precise-stevens-x”, with “x” being the number, i.e. “1”. By connecting to this, the Raspberry Pi can be accessed through SSH using a program called [PuTTY](#).

Local Address	Password
pi@192.168.0.1	raspberry
pi@cchil-precise-stevens-1	raspberry

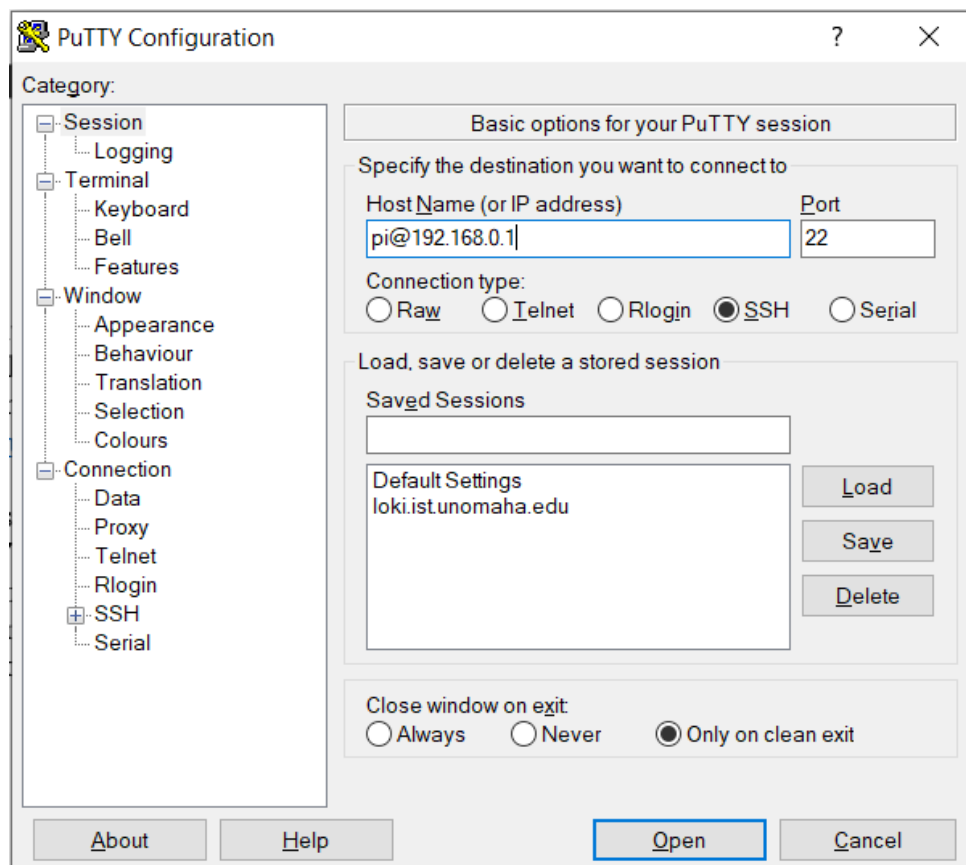


Figure 4 – PuTTY Terminal Configuration Window

Reliability Testing

We performed ten trials of increasing treat quantities: 1, 2, 3, ..., 9, and 10 treats. This totals 55 treats per trials for a total of 550 treats. The over- and under- dispensation rates, along with jamming events, are recorded in the following tables. The mean error rate for the six dispensers is calculated, along with the mean error for each trial amount. This procedure was performed for the five dispensers that were built for the Canine Cognition and Human Interaction Laboratory.

Table 1 – Results of the dispenser trials

	Total Treats Dispensed	Correct Number Dispensed	Over- Dispensed	Under- Dispensed
Dispenser One	550	542	8	1
Dispenser Two	550	543	7	0
Dispenser Three	550	549	1	0
Dispenser Four	550	548	2	0
Dispenser Five	550	549	1	0

Table 2 – Calculated accuracy and error rates of the dispensers

	Accuracy	Error
Dispenser One	92%	8%
Dispenser Two	93%	7%
Dispenser Three	98%	2%
Dispenser Four	99%	1%
Dispenser Five	98%	2%

Table 3 – Average accuracy and error across the five dispensers

Mean Accuracy	Mean Error
96%	4%

The trials show that any dispensing command has a 4% chance of failing. Considering that the number of treats affects the possibility of errors occurring, the accuracy and error across all ten treat quantities tested were calculated.

Table 4 – Cross quantity accuracy and error metrics

	Accuracy	Error
One Treat	100%	0%
Two Treats	100%	0%
Three Treats	100%	0%
Four Treats	100%	0%
Five Treats	100%	0%
Six Treats	98%	2%
Seven Treats	100%	0%
Eight Treats	84%	16%
Nine Treats	94%	6%
Ten Treats	86%	14%

The table above shows that dispensing less than eight treats has the highest likelihood of success in all cases. The raw data is available in the [GitHub repository](#).

Bill of Materials

Bill of Materials – Precise Dispenser			
Part	Quantity	Cost	Link
1 kg black filament	1	\$ 22.99	Filament
Raspberry Pi 4	1	\$ 55.00	Raspberry Pi
HDMI Panel Mount	1	\$ 7.95	HDMI
USB C Panel Mount	1	\$ 7.50	USB C
Barrel Jack Panel Mount	1	\$ 5.17	Barrel Jack
HDMI to Micro HDMI	1	\$ 8.69	Micro HDMI
Stepper Motor Power Supply	1	\$ 11.61	12V Power
Stepper Motor	1	\$ 8.99	Motor
Raspberry Pi Power Supply	1	\$ 8.00	Pi Power
Plastic Cover	1	\$ 9.95	8 in Cover
IR Break Beam	1	\$ 1.95	3mm IR
M3 Screws	1	\$ 26.99	M3 Pack
M3 Hex Wrench	1	\$ 14.99	Hex Set
Cable Ties	1	\$ 9.69	Zip Ties and Mounts
Micro SD Card	1	\$ 9.49	SD Card Pack
USB C Cable	1	\$ 6.99	USB C Cables

Bill of Materials – Precise Dispenser Pi HAT			
Part	Quantity	Cost	Link
PCB	3	\$ 28.45	[Add OSHPark Link]
A4988 Stepper Motor	1	\$ 4.69	Motor Driver
2 Pin Screw Terminal	2	\$ 2.20	2x 3.5mm Terminal
3 Pin Screw Terminal	1	\$ 1.35	3x 3.5mm Terminal
Capacitor	1	\$ 0.13	10u TH Cap
Resistor	1	\$ 0.10	10k TH Res
40 Pin Header	1	\$ 0.95	Pi Header
Stepper Motor Header	1	\$ 0.20	4 Pin Right Angle
A4988 Female Pins	2	\$ 1.30	2x Female Headers

Document Revision History

Revision, Name, and Date	Revision Description
V0.1, Walker Arce, 7 March 2021	First draft creation date for the precise dispenser
V0.2, Walker Arce, 3 April 2021	Added Python driver documentation
V0.3, Walker Arce, 9 April 2021	Added Python interfacing example and bills of material
V0.4, Walker Arce, 20 September 2021	Added build, test, and troubleshooting instructions