A Device for the Somatosensory Stimulation of Rodent Hindlimbs

---

Stephan Blanz[1], Royal Oakes[1], John Beckman[1], Jinyuxuan Guo[1]

1. University of Wisconsin - Madison, Department of Biomedical Engineering

---

Corresponding author and address:
Stephan Blanz
University of Wisconsin Madison, Department of Biomedical Engineering
1415 Engineering Drive
Madison, WI 53706
John Beckman
University of Wisconsin Madison, Department of Biomedical Engineering
1415 Engineering Drive
Madison, WI 53706

Declarations of interest: none

Abstract

Currently, no methods exist to test the efficacy of sciatic nerve repair on rodent models based on responsiveness to hindlimb somatosensory stimulation. The proposed device produces somatosensory stimulations through vibrations induced on the hindlimb of a conscious rat. By isolating the stimulation to either left or right hindlimb, the subject is presented a two-alternative forced choice (2AFC). A healthy rat is trained to conditionally respond to the stimulus by indicating on which limb the stimulus was felt. Once the rat is trained, the nerve is surgically deafferented and repaired. During recovery the rat is presented with the same 2AFC at various time points to determine the length of recovery of sensory function.

Keywords

Somatosensory Stimulation, Vibrotactile Stimulation, Two Alternative Forced Choice (2AFC) Testing, Sciatic Nerve Repair

Introduction

Allowing patients to fully recover following a severed nerve is extremely challenging. There are many studies regarding surgical methodologies as well as pre and post-operative care to repair the damaged nerves in the Peripheral Nervous System (PNS) [1]–[4]. One of the main task of these studies is analyzing the success of these surgical procedures to determine how effective the treatment was in repairing the neurological connection that was lost, or damaged. This work could translate into surgical procedures to help humans who have had their neurological connections damaged or severed, regain the control and sensations that they may have lost otherwise.

Damage to nerves in adult mammals have very different regeneration possibilities depending on where in the nervous system they occur. A damaged nerve in the Central Nervous System (CNS), is almost completely incapable of regenerating due to the neuroglia in the CNS [5]. Astrocytes in the CNS, after a nerve is damaged, form a barrier around the damaged nerves, which turns into astroglial scar tissue. In addition to the blockage of nutrients due to the astrocytes, and astroglial scar tissue, the myelin of the CNS releases growth-inhibiting proteins which block any regrowth of a nerve in the CNS [5]. However, in the PNS, damaged nerve cells undergo Wallerian Degeneration, a process which protects a portion of the damage peripheral nerve with Myelin, while the debris of the damaged section is cleared by macrophages. Wallerian degeneration is followed by axon regeneration, in which growth-promoting factors are released by the myelin and cell body, and help to guide the regrowing axon back to the cell body [4]. This process is only possible in the PNS and if the distal portion of the axon is not too far from the cell body [5]. In a scenario where the Wallerian Degeneration leaves the axon too far to reattach to the cell body, a surgeon can guide the

regrowing axon to the cell body by removing the damaged neurological tissue and scar tissue in the area. Additionally, if the gap between the axon and cell body is too large, a non-essential nerve may be sacrificed from elsewhere in the body [6]. Although these procedures are well documented, there currently exists no standardized way to test the effectiveness of these surgeries. If this solution can be shown to have a significant positive impact on the regeneration of nerve connections in rats, it could one day be implemented to help regrow human nerves as well.

Historically, a nerve of interest in repair studies has been the sciatic nerve. This nerve innervates from the buttocks into the leg and foot. It serves to connect the muscles of the leg and foot to the spinal cord and brain [7]. Thus, after surgical repair of this nerve, the success of the surgery must be assessed by stimulating this nerve, and observing a rats ability to distinguish between individual stimuli to each hindlimb. Conventional methods of nerve stimulation includes heat and electric shock [8], [9]. However, it may be detrimental to use these methods in animal models since they may have limited nerve function. Such a stimulus could then have the potential to harm the animal without without the subject showing signs of pain. Therefore, there is a need for developing a non-invasive, non-harmful way to stimulate the sciatic nerve in rats to determine the effectiveness of the surgery.

Different from heat and electric shock, vibration provides a safer way to stimulate the nerve of interest. Beneath the skin of mammals, Meissner corpuscles, Pacinian corpuscles, Merkel disks, and Ruffini corpuscles make up the mechanoreceptors that contribute to somatic sensation [10]. These mechanoreceptors are responsible for receiving touch and pressure stimuli and sending corresponding signals to the brain. Vibrations can cause pressure on the rodents' skin and the rodents will notice it through this mechanism. Mice exhibited the most noticeable transient responses to vibrational frequencies between 70-100 Hz, and at an acceleration of 1.0 m/s^2 [11]–[13]. Another study found that rats, after being stimulated within a range from 0 - 600 Hz, would attenuate with vibrations around 50 Hz [13]. Furthermore, the auditory perceptible frequency range in rats is from 200 Hz to 80 kHz [14]. Therefore, producing vibrations under 200 Hz would ensure the stimulus is limited to vibration rather than auditory perception. These different studies will be used to help determine the range of vibrational frequencies that will be available to stimulate the sciatic nerves of rats during surgical nerve repair experiments.

Materials

A plexiglass enclosure was used to house the rodent. The bottom of the plexiglass enclosure had a sheet of isoprene rubber coating the entire surface. The isoprene rubber provides a sanitizable surface and is an ideal model for transmitting low frequency vibrations [15]. The electronics unit is housed underneath the enclosure and includes an amplifier (Dayton Audio DTA-1 Class D AC/DC Battery Powered Mini Amplifier 15 WPC) [16] which is connected to two transducers (Dayton Audio DAEX13CT-8 Coin Type 13mm Exciter 3W 8

Ohm) (Figure 1) [17]. The transducers are part of two column assemblies which include: an acrylic standoff, the transducer itself, ¾ inch PVC, filled with concrete (to increase stiffness), and two layers of isoprene rubber to act as a dampener underneath. The transducer was attached to the standoff and column using superglue. The column was attached to the dampener using epoxy, and the dampener was attached to the acrylic using rubber cement.

The isoprene rubber floor and the acrylic standoff were selected to increase vibration isolation to the target area. Acrylic was chosen due to its light weight and ability transmit vibrations from the differing geometries of the circular transducer to the rectangular hindlimb pad [15]. The isoprene rubber was chosen due its ability to isolate vibrations, especially at low frequencies [15], [18].

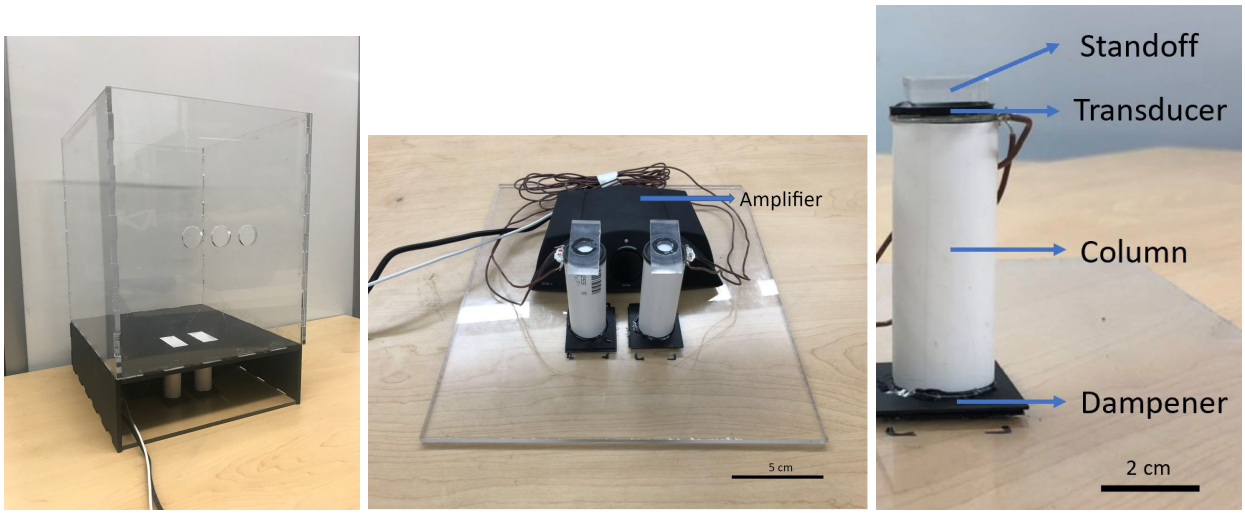


Figure 1 (Left). The finished device. Features: top enclosure with visible placement marks (white) for the rodent's hind limbs and nose holes through which the rat can nose poke based on stimulus and training received. The electronics assembly is housed underneath. **(Middle).** The electronics assembly which includes an amplifier and pedestals with transducer. **(Right).** Expanded view of the pedestal assembly featuring an acrylic standoff, transducer, ¾" PVC column filled with concrete, and isoprene rubber as a dampener.

For testing two Arduino UNOs were used in conjunction with MATLAB to collect data, along with a Sparkfun TAL221 Miniature Load Cell [19], a Sparkfun HX711 Load Cell Amplifier [20], [21] and a SparkFun LIS3DH Triple Axis Accelerometer [22].

Methods

Experimental Setup

Producing Transducer Inputs: For all experiments, the method of producing input to the transducer was the same. The transducer accepts an input sinusoid and vibrates at the

frequency and intensity of the input sinusoid. We used the MATLAB file Pruebasonido.m to produce the input sinusoid for all experiments. The amplitude of the sinusoid was either 0.1 or 0.2 and the frequency ranged from 20-200 Hz. Exceeding these limits could damage the transducer.

Frequency Testing: A test was performed using a Logitech H390 USB Headset to determine the relationship between input and output frequency of the transduce (Figure 2). The microphone from the Logitech headset was placed 1/4 inch above the surface of the transducer. Frequencies of 50, 100, and 150 Hz were used as inputs and 6 seconds of audio was captured for each frequency. The audio file was analyzed using MATLAB's Fast Fourier Transform (FFT) to determine major frequency components.

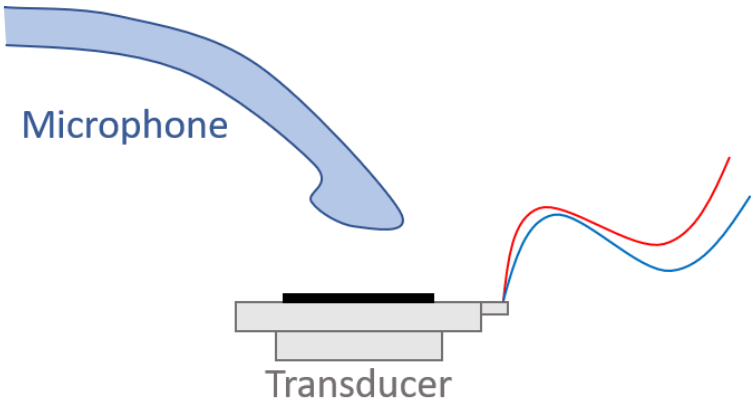


Figure 2. Testing setup of frequency test. A microphone is used to collect the signal from the transducer. A MATLAB code is then used to verify the output frequency matches input frequency through FFT.

Displacement Testing: A test was performed to determine the displacement of the transducer at various input frequencies and voltages (Figure 3). A thin white plate was placed on top of the transducer to provide contrast with a dark background. A ruler was placed next to the transducer to provide a reference. A camera was used to take a 32 to 36 second video of the transducer while it was producing vibrations. The video was processed in MATLAB by averaging all the frames in the video to a single image. The resulting blur was then measured in ImageJ for its thickness to determine peak-to-peak displacement.

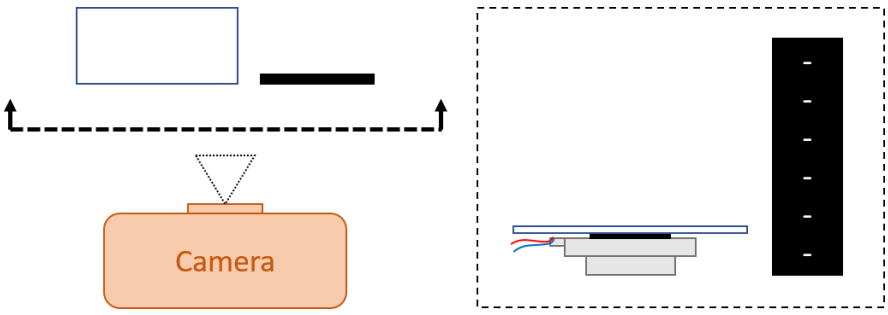


Figure 3. Testing setup of displacement test. A camera is used to record the videos of a vibrating transducer. A ruler (black) is placed aside to indicate scale.

Bleed Testing: A test was performed in the final design cage to determine the amount of off-target stimulation produced when stimulating a single hindlimb. A Sparkfun ADXL335 accelerometer was placed directly on the platform being vibrated (direct-target configuration) and recorded using an Arduino. Sampling was at 1000 Hz for 0.6 seconds. Then testing was repeated with the accelerometer on the platform adjacent to the platform being vibrated (off-target configuration). The magnitude of the acceleration vector was computed and was analyzed using MATLAB's FFT. At each input frequency, the amplitude of the FFT data for the off-target configuration was divided by the amplitude of the FFT data for the direct-target configuration to obtain the bleed percentage.

Load Cell Testing: A test was performed to determine the force exerted by the transducer using a load cell (Figure 4). The load cell was fixed in the cage such that the free end was above the transducer. There was a sheet of rubber between the load cell and the transducer. The load cell was then used to measure the force produced by transducer due to acceleration at various frequencies and amplitudes.

Figure 4. Testing setup of load cell test. The load cell (blue) is placed on either the vibrating pedal or the non-vibrating pedal. The force generated by the transducer is collected and analyzed.

Data was collected from the accelerometer using an Arduino UNO and was sent to MATLAB for processing (Appendix B). First, the motor was powered and 128 samples of the acceleration vector were collected from the accelerometer at 1 kHz. This number of samples was chosen due to memory limitations of the Arduino UNO. Second, the data was imported to MATLAB. Third, the magnitude of the acceleration vector was computed. Fourth, the Fourier transform of the accelerometer magnitude was computed using the MATLAB fast Fourier transform [13]. This method was used rather than sending the data one sample at a time because literature suggests that our method leads to more accurate data [23].

Results

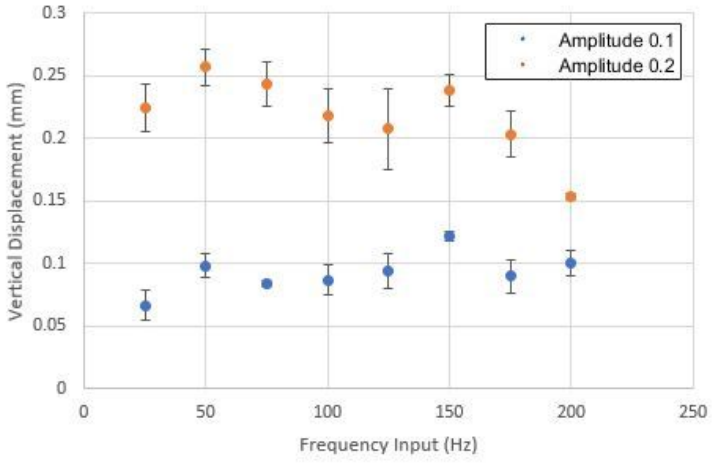


Figure 5. Relationship between displacement and frequency. The vertical displacement at an amplitude of 0.1 and 0.2 was computed at frequencies between 25 Hz and 200 Hz at 25 Hz intervals (n=3 for each trial).

From the displacement test, the displacement data at amplitude 0.1, 0.2 and frequencies from 25 Hz to 200 Hz was collected. To find out the relationship between frequency and amplitude, two Chi-square test is conducted on the results at 0.1 and 0.2 amplitude of the displacement test. The vertical displacement at an amplitude of 0.1 and 0.2 was computed at frequencies between 25 Hz and 200 Hz at 25 Hz intervals (n=3 for each trial). The Chi-square test is then conducted to check whether there is significant difference between observed results and the expected value. In this case, the observed results are displacement data at different frequency and the expected value will be the overall average displacement at 0.1 and 0.2 amplitude. The null hypothesis is that there is no significant difference between displacement results obtained from different frequency and overall average displacement, the alternative hypothesis is there are significant differences. Both of the results of the two Chi-square test at 0.1 and 0.2 amplitude support our null hypothesis. There is no statistically significant difference in displacement at various frequencies at 0.1 amplitude, X^2 (df = 7, N = 24) = 0.01858, p > 0.99, or at 0.2 amplitude, X^2 (df = 7, N = 24) = 0.03269, p > 0.99.

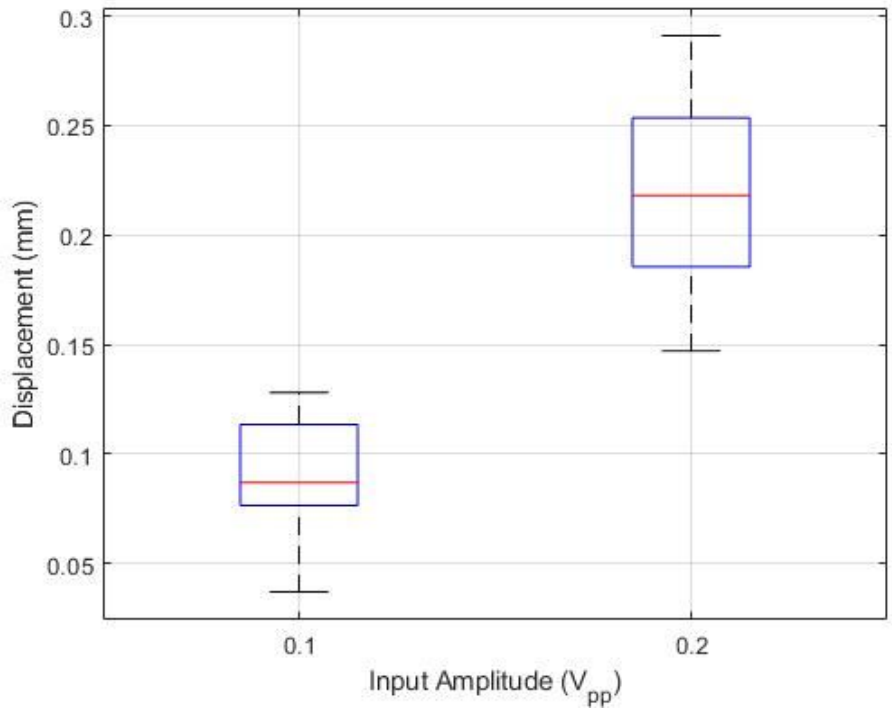


Figure 6. Relationship between displacement and voltage. A summary of the results of the uncoupled displacement measurements of the transducer. These results are sampled at a range of different frequencies (25-200 Hz), and highlights the statistical results of figure 3.

The frequency amplitude test proves there is minimal relationship between frequency and amplitude, while another test is needed to prove that the displacement of the transducer increases as the input amplitude is turned up. A voltage displacement test is conducted on the transducer to figure out the relationship between input voltage and displacement. At 0.1 amplitude, the measured displacement varies from 0.025 to 0.125 mm . At 0.2 amplitude, it varies from 0.15 to 0.39 mm. As amplitude increases from 0.1 to 0.2, the average displacement of the transducer also increases, from to . There is positive correlation between input amplitude and displacement measured on the transducer. A one-tailed paired t-test was also run on the data corresponding to the 0.1 and 0.2 input amplitudes, the results show a strong statistical difference between the displacements and input amplitudes with $p < 0.001$ (N=24).

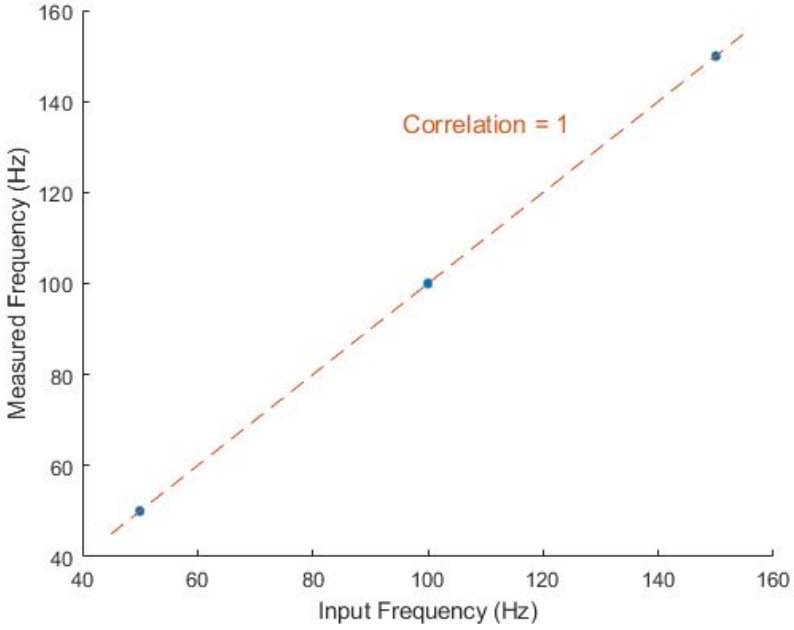


Figure 7. Frequency verification. Samples were taken at 50 Hz, 100 Hz, and 150 Hz (n=5 for each). There is a positive correlation between the input frequency and the measured frequency.

In order to justify that the output frequency is similar or equal to the input frequency, the frequency verification test is designed and accomplished. The transducer is running at 50 Hz, 100 Hz and 150 Hz while this signal is recorded. A MATLAB code is then used to measure the output frequency of the transducer through FFT. The measured average frequency is 50 Hz at 50 Hz input frequency, 100 Hz at 100 Hz input frequency and 150 Hz at 150 input frequency. The out frequency is positively correlated with the input frequency. The pearson's correlation coefficient is 1.

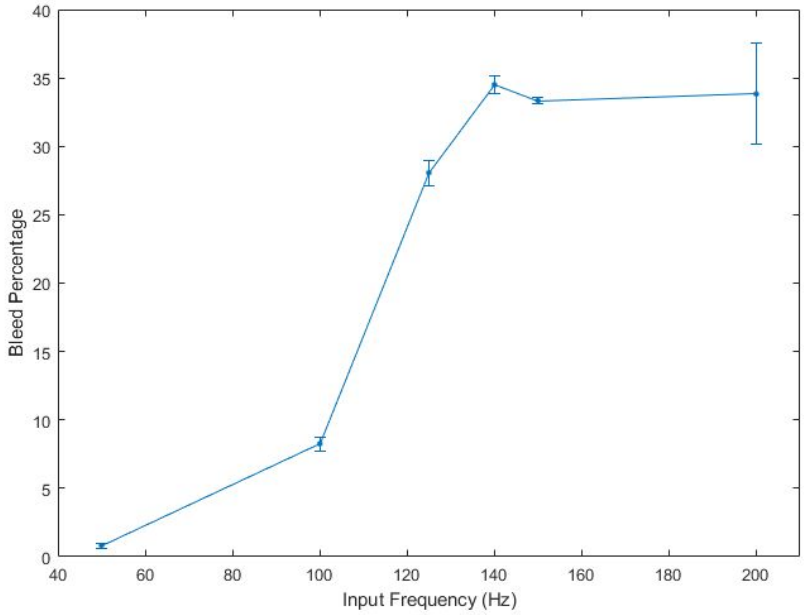


Figure 8. Bleed between platforms. The percent bleed between the two platforms was calculated at several frequencies (n=5 for each). There is a noticeable drop off in percent bleed between 100 Hz and 125 Hz. Error bars represent ± 1 SE.

Another parameter that needs to be tested is the isolation of the vibrations from two pedals. This is very important because the device should be able to stimulate only one of the rodent's hindlimbs without stimulating the other. A bleedover test is conducted to see how much vibration exists on non-vibrating pedal. The results are shown on the figure. Each data point represents the off-target stimulation at that frequency, normalized by the on-target stimulation at the same frequency. At frequencies under 100 Hz, the bleed percent is under 10%, while the bleed percent can go over 30% at frequencies above 125 Hz. Bleed percent increases as the input frequency becomes higher. At high frequencies over 100 Hz, the isolation of vibrations from two pedals is partially lost.

Discussion

Engaging a rodent by applying a vibrational stimulus is a far more humane method of stimulation than subjecting the animal to a painful stimulus. Conventionally, rodents have been stimulated by the use of heat, cold, or electric shock. These stimulation methods can have

potentially detrimental effects by applying a dangerous stimulus to an animal without the ability to pull away or respond to the stimulus.

By applying a vibrotactile stimulus to the hindlimbs of rodents rather than a potentially harmful mode of stimuli the researchers using this tool will be able to explore the large parameter space that this device encompasses. In previous iterations of this design, there was a common phenomenon being seen in the use of different motors, such as an LRA or ERM, amplitude-frequency coupling. However, when using the transducer as the displacement motor the user has control of both output displacement, and output vibrational frequency. This gives the user more freedom to determine the most effective parameters for stimulating the hindlimbs in the rodents.

In the final design of this tool, a layer of isoprene rubber was laid on the bottom of the enclosure in order to protect the electronics housed underneath the portion where the animals will be placed. However, the results from the bleed testing show the problem with using this rubber layer to insulate against biological waste from the animals. By covering the entire floor of the enclosure with a sheet of rubber, there is bleed between the active and non-active platforms. This is significantly more apparent at frequencies over 100 Hz. This is most likely due to the increased acceleration in the motor needed to reach the same displacement in a shorter period of time, which then impacts the other platform. There was the possibility in adding some modularity to this section of the enclosure, however, safety of the electronic components and simplicity of design were prioritized over minimal bleed. Additionally, the very low bleed, approximately 10% at 100 Hz impacted this decision, as to which the recommendation is not to use frequencies above 100 Hz.

Future work will include performing animal testing with our cage. The rats will be trained to position itself with one foot standing on one platform and one foot on the other. There will be 3 holes on the side of the cage. The rats will be trained to use these holes to tell whether they feel the vibration and in which foot. If they feel the vibration through their left foot, they will put their nose into the left hole, and right hole for right foot. If the rat is unsure or feels nothing, they will put their noses into the middle hole. Researchers can use this method to determine the function of the reconnected sciatic nerve of the rats. In addition to examining the entire parameter space of this research tool.

Acknowledgments

Funding for this project was provided by Dr. Aaron Suminski, a scientist with the University of Wisconsin, department of Biomedical Engineering and Dr. Aaron Dingle, an associate scientist with the University of Wisconsin, Department of Surgery, Poore lab. The Poore lab’s research focus is clinical and experimental microsurgery, with an emphasis on peripheral nerve regeneration and repair. Together they are researching the effectiveness of various treatments and methods of nerve repair.

Additionally, this project could not have been completed without the help and mentorship of Dr. Jeremy Rogers.

Appendix

A. Preliminary Design Specifications

Function:

Peripheral nerve injuries are common, debilitating and costly. Approximately 2.8% - 5% of all trauma patients in the US sustain such an injury. Many peripheral nerve injuries are a result of amputations, which affect an estimated 185,000 people in the US each year. Prosthetics are continually improving, but a large issue that remains is the patient's lack of tactile perception. Many researchers, including Dr. Aaron Dingle, are designing devices to solve this problem. The functional outcome of these devices can be assessed in humans by asking the patient questions, but this technique is not an option in animal models. Rats are commonly used as animal models as a precursor to human subject testing. In order to receive functional outcome data from rats, a healthy rat can be trained to respond in a certain way to a somatosensory stimulus. A peripheral nerve can then be surgically cut and the novel device implanted. The device can then be used to apply what should be recognized as the same somatosensory stimulus the rat was trained with. Observations on the percent of correct reactions can be used to determine success.

This project aims to design the somatosensory stimulation device used to train the rats. The device should be able to apply a graded stimulus to at least two limbs individually. The device will consist of a cage or cage insert as well as a microcontroller to control the stimulus grade.

Client requirements:

- Must house one rat
- Must be able to individually stimulate each hind limb of the rat
- Must be completely controlled by a computer (no user input other than to start the device and configure settings)
- Must be sterilizable
- Mouse must be visible while in the cage

Design requirements:

1. Physical and Operational Characteristics

a. *Performance requirements:* Multiple experiments may be run per day. The device should be capable of repeated use and cleaning between trials.

b. *Safety*: The device should not harm the animals in any way.

c. *Accuracy and Reliability*: The device should accurately and reliably produce a graded stimulation of the user's choice.

d. *Life in Service*: Trials would last approximately 15 minutes in duration.

e. *Shelf Life*: A 10 year shelf life is prefered.

f. *Operating Environment*: Room temperature (25°C) and pressure (1 atm).

g. *Ergonomics*: The rodent should have enough room to rear on its hind legs and interact with the forward facing wall.

h. *Size*: No smaller than 30 cm x 30 cm x 60 cm. No larger than 120 cm x 120 xm x

120 cm

i. *Weight*: No heavier than 11 kg.

j. *Materials*: Plexiglass.

k. *Aesthetics, Appearance, and Finish*: Must be able to observe test subject through wall material.

2. Production Characteristics

a. *Quantity*: One unit.

b. *Target Product Cost:* < $100

3. Miscellaneous

a. *Standards and Specifications*: The device should be approved by the UW-Madison Research Animal Resources and Compliance (RARC) center.

b. *Competition*: None

B. MATLAB and Arduino Code

Accel_read.ino

```
#include <MsTimer2.h>

#define BUFF_SIZE 384

/* The function that will read data from the arduino. */
void getData();

/* The analog inputs. */
int xin = A0;
int yin = A1;
int zin = A2;

/* Buffer to hold values from accelerometer. */
unsigned short buff[BUFF_SIZE];

/* A count of the number of elemets in buff. */
unsigned short cc = 0;

/* The variables to control the motor. */
int motor = 9;
int power = 252;

void setup() {
  analogWrite(motor, power);

  Serial.begin(9600);   // Start serial communication.

  pinMode(xin, INPUT);
  pinMode(yin, INPUT);
  pinMode(zin, INPUT);

  Serial.println("D");

  MsTimer2::set(1, getData);
  MsTimer2::start();
}

void loop() {
  int matCmd = 0;
```

```
  if (cc >= BUFF_SIZE) {
    MsTimer2::stop();
  }

  // If matlab needs more data, restart the program.
  matCmd = Serial.read();
  if (matCmd == 49) {
    Serial.println('1');

    for (int i = 0; i < BUFF_SIZE; i++) {
      Serial.println(buff[i]);
    }

    cc = 0;
    MsTimer2::start();
  }
}

void getData() {
  buff[cc++] = analogRead(xin);
  buff[cc++] = analogRead(yin);
  buff[cc++] = analogRead(zin);

  if (cc >= BUFF_SIZE) {
    MsTimer2::stop();
  }
}
```

Load Cell Calibration

```
#include <HX711.h>

/*
 *
Source code provided by
 By: Nathan Seidle
 SparkFun Electronics
 Date: November 19th, 2014
 License: This code is public domain but you buy me a beer if you use this and we meet
someday (Beerware license).

 Expanded by:
 Stephan Blanz

 This is the calibration sketch. Use it to determine the calibration_factor that the main example
uses. It also
 outputs the zero_factor useful for projects that have a permanent mass on the scale in between
power cycles.

 Setup your scale and start the sketch WITHOUT a weight on the scale
 Once readings are displayed place the weight on the scale
 Press +/- or a/z to adjust the calibration_factor until the output readings match the known
weight
 Use this calibration_factor on the example sketch

 This example assumes pounds (lbs). If you prefer kilograms, change the Serial.print(" lbs"); line
to kg. The
 calibration factor will be significantly different but it will be linearly related to lbs (1 lbs =
0.453592 kg).

 Your calibration factor may be very positive or very negative. It all depends on the setup of your
scale system
 and the direction the sensors deflect from zero state
 This example code uses bogde's excellent library: https://github.com/bogde/HX711
 bogde's library is released under a GNU GENERAL PUBLIC LICENSE
 Arduino pin 2 -> HX711 CLK
 3 -> DOUT
 5V -> VCC
 GND -> GND

 Most any pin on the Arduino Uno will be compatible with DOUT/CLK.

 The HX711 board can be powered from 2.7V to 5V so the Arduino 5V power should be fine.
```

```
*/

#include "HX711.h"

#define DOUT  3
#define CLK  2

HX711 scale;

float calibration_factor = 11800; // 11800 in cage for g

void setup() {
  Serial.begin(9600);
  Serial.println("HX711 calibration sketch");
  Serial.println("Remove all weight from scale");
  Serial.println("After readings begin, place known weight on scale");
  Serial.println("Press + or a to increase calibration factor");
  Serial.println("Press - or z to decrease calibration factor");

  scale.begin(DOUT, CLK);
  scale.set_scale();
  scale.tare(); //Reset the scale to 0

  long zero_factor = scale.read_average(); //Get a baseline reading
  Serial.print("Zero factor: "); //This can be used to remove the need to tare the scale. Useful in
permanent scale projects.
  Serial.println(zero_factor);
}

void loop() {

  scale.set_scale(calibration_factor); //Adjust to this calibration factor

  Serial.print("Reading: ");
  Serial.print(scale.get_units(), 1);
  Serial.print(" g"); //Change this to kg and re-adjust the calibration factor if you follow SI units
like a sane person
  Serial.print(" calibration_factor: ");
  Serial.print(calibration_factor);
  Serial.println();

  if(Serial.available())
  {
    char temp = Serial.read();
```

```
    if(temp == '+' || temp == 'a')
      calibration_factor += 25;
    else if(temp == '-' || temp == 'z')
      calibration_factor -= 25;
    else if(temp == 'w')
      calibration_factor += 100;
    else if(temp == 's')
      calibration_factor -= 100;
          else if(temp == 'e')
      calibration_factor += 1000;
    else if(temp == 'd')
      calibration_factor -= 1000;
    else if(temp == 't')
      scale.tare();
  }
}
```

Load Cell Continuous Measurement

```
/*
 Example using the SparkFun HX711 breakout board with a scale
 By: Nathan Seidle
 SparkFun Electronics
 Date: November 19th, 2014
 License: This code is public domain but you buy me a beer if you use this and we meet
someday (Beerware license).

 This example demonstrates basic scale output. See the calibration sketch to get the
calibration_factor for your
 specific load cell setup.

 This example code uses bogde's excellent library: https://github.com/bogde/HX711
 bogde's library is released under a GNU GENERAL PUBLIC LICENSE

 The HX711 does one thing well: read load cells. The breakout board is compatible with any
wheat-stone bridge
 based load cell which should allow a user to measure everything from a few grams to tens of
tons.
 Arduino pin 2 -> HX711 CLK
 3 -> DAT
 5V -> VCC
 GND -> GND

 The HX711 board can be powered from 2.7V to 5V so the Arduino 5V power should be fine.

*/

#include "HX711.h"

#define calibration_factor 11800.0 //This value is obtained using the
SparkFun_HX711_Calibration sketch

#define DOUT  3
#define CLK  2

HX711 scale;

void setup() {
  Serial.begin(9600);
  // Serial.println("HX711 scale demo");
```

```
  scale.begin(DOUT, CLK);
  scale.set_scale(calibration_factor); //This value is obtained by using the
SparkFun_HX711_Calibration sketch
  scale.tare(); //Assuming there is no weight on the scale at start up, reset the scale to 0

  // Serial.println("Readings:");
}

void loop() {
  Serial.print(scale.get_units(), 1); //scale.get_units() returns a float
  // Serial.print(" g"); //You can change this to kg but you'll need to refactor the calibration_factor
  Serial.println();
}
```

Movie Frame Averager

```
%% Movie Averager
% By: Stephan Blanz
% Portions of code found at:
https://ww2.mathworks.cn/matlabcentral/answers/59194-reading-video-file-to-matlab?s_tid=gn_loc_drop

% This code will ask the user to select a file. The program will then
% average the frames and both display and save the mean frame as well as an
% index frame (which can be selected by the user in the code).
%
% It is not recommended to use the matlab generated pictures for imageJ
% analysis as I have found there seems to be a compression that causes
% blur. It is also not recommended to use matlab to zoom in on generated
% pictures, but to let imageJ do this instead, as this has also been
% observed to cause blur.
%
% This program will show the images more as a sanity check than an
% analytical tool. Ensure you have selected the correct file and that your
% reference frame is free of blue and conversely, your mean frame has blur.

%%
clc;    % Clear the command window.
close all;  % Close all figures (except those of imtool.)
imtool close all;  % Close all imtool figures.
clear;  % Erase all existing variables.
workspace;  % Make sure the workspace panel is showing.



% Set this to the index of the frame you want to use as reference. Recommend
% to use one shortly before start of video since there is no motor motion
% yet and also no artifact from pushing record
indexGrabLocation = 5;

strErrorMessage = sprintf('Please select a file you want to analyze.');
response = questdlg(strErrorMessage, 'Choose a File', 'Select File', 'Cancel', 'Select File');
if strcmpi(response, 'Select File')
   [baseFileName, folderName, FilterIndex] = uigetfile('*.mov'); % Change this if using a different
file format
   if ~isequal(baseFileName, 0)
      movieFullFileName = fullfile(folderName, baseFileName);
   else
      return;
   end
```

```
else
    return;
end

try
    videoObject = VideoReader(movieFullFileName)
    % Determine how many frames there are.
    numberOfFrames = videoObject.NumberOfFrames;
    vidHeight = videoObject.Height;
    vidWidth = videoObject.Width;

    numberOfFramesWritten = 0;

    % Loop through the movie, getting mean frame.
    for frame = 1 : numberOfFrames
        % Extract the frame from the movie structure.
        thisFrame = read(videoObject, frame);

        % Uncomment if you want to go frame by frame and watch
        % WARNING: TAKES REAAAAALLY LONG.
        % %     % Display it
        % %     hImage = subplot(1, 2, 1);
        % %     image(thisFrame);
        % %     axis('on', 'image');
        % %     caption = sprintf('Frame %4d of %d.', frame, numberOfFrames);
        % %     title(caption, 'FontSize', fontSize);
        % %     drawnow; % Force it to refresh the window.

        % Calculate the mean frame.
        if frame == 1
            sumFrame = double(thisFrame);
        elseif frame == 5
            indexFrame = thisFrame;
        else
            sumFrame = sumFrame + double(thisFrame);
        end
        meanFrame = uint8(sumFrame / frame);
        %     % Uncomment this to plot as you go
        %     % Plot the mean gray levels.
        %     hPlot = subplot(1, 2, 2);
        %     imshow(meanFrame);
        %     axis('on', 'image');
        %     caption = sprintf('Mean of %d frames', frame);
        %     title(caption, 'FontSize', fontSize);
        %     drawnow;
```

```
    %
    %     % Update user with the progress.  Display in the command window.
    %     progressIndication = sprintf('Processed frame %4d of %d.', frame, numberOfFrames);
    %     disp(progressIndication);
  end

  hImage = figure(1);
  imshow(indexFrame);
  % Uncomment if you want it zoomed in. Not recommended for image
  % processing as MATLAB screws with the formatting and you get a blur
  %axis([1000 2000 700 1000]);
  caption = sprintf('%s     Index Frame', baseFileName);
  title(caption);

  hPlot = figure(2);
  imshow(meanFrame);
  %axis([1000 2000 700 1000]);
  caption = sprintf('%s     Mean of %d frames', baseFileName, frame);
  title(caption);

  %axis([hImage hPlot],[370 570 190 250],'on','image')


  % IMPORTANT: This is where images are stored.
  fileName1 = sprintf('IndexFrame_%s.png', baseFileName);
  fileName2 = sprintf('MeanFrame_%s.png', baseFileName);
  imwrite(indexFrame,fileName1);
  imwrite(meanFrame,fileName2);

  % Alert user that we're done.
  finishedMessage1 = sprintf('Done!  It processed %d frames of\n    "%s"', numberOfFrames,
movieFullFileName);
  finishedMessage2 = sprintf('Files saved as\n "%s" \n and \n "%s"', fileName1, fileName2);
  disp(finishedMessage1); % Write to command window.
  uiwait(msgbox(finishedMessage1)); % Also pop up a message box.
  uiwait(msgbox(finishedMessage2)); % Also pop up a message box.

catch ME
  % Some error happened if you get here.
  strErrorMessage = sprintf('Error extracting movie frames from:\n\n%s\n\nError: %s\n\n)',
movieFullFileName, ME.message);
  uiwait(msgbox(strErrorMessage));
end
```

MatlabSerialArduino.m

```
clear
close all
clc

% The number of samples in a single package from the Arduino.
PACKAGE_SIZE = 128;
% The number of packages recieved from the Arduino.
PACKAGE_NUM = 1;
% The number of values in each sample from the Arduino.
SAMP_DIM = 3;

xyzData  = zeros(PACKAGE_SIZE, SAMP_DIM);
xData    = zeros(PACKAGE_SIZE * PACKAGE_NUM, 1);
xDataFFT = zeros(PACKAGE_SIZE * PACKAGE_NUM, 1);
t = 1:PACKAGE_SIZE;
mag = zeros(PACKAGE_SIZE * PACKAGE_NUM, 1);
magFFT = zeros(PACKAGE_SIZE * PACKAGE_NUM, 1);

% Initialize communication with Arduino.
try
   s = serial('COM5', 'BaudRate', 9600);
   fopen(s);
   fscanf(s); % Get the first byte to acknowledge Arduino.
catch
   warning('Could not establish connection with Arduino.');
   fclose(s);
end

% Get data from Arduino a certain number of times.
for n = 1:PACKAGE_NUM
   try
      % Send a signal to the Arduino to get another data set.
      fwrite(s, '1', 'uint16');

      fscanf(s);

      % Read data from arduino.
      for i = 1:PACKAGE_SIZE
         for j = 1:SAMP_DIM
            val = fscanf(s);
```

```
            xyzData(i,j) = str2double(val([1:length(val) - 2]));
        end
    end

    % Get the magnitude of the acceleration vectors.
    for i = 1:PACKAGE_SIZE
        mag(((n-1)*PACKAGE_SIZE) + i) = norm(xyzData(i, :));
    end

    xData([(((n-1)*PACKAGE_SIZE) + 1):(((n)*PACKAGE_SIZE))]) = xyzData(:,1);
    n

  catch
    warning('Something happened.');
    fclose(s);
  end % End Try-Catch
end % End for loop

fclose(s); % Close serial communication with Arduino.

magFFT = fft(mag);
xDataFFT = fft(xData);
```

PlotsPlotsPlots.m

```
% The number of samples in a single package from the Arduino.
PACKAGE_SIZE = 128;
% The number of packages received from the Arduino.
PACKAGE_NUM = 1;
% The number of values in each sample from the Arduino.
SAMP_DIM = 3;

% x1 = abs(fftshift(csvread("cyl_dir_mag_102.csv")));
% x2 = abs(fftshift(csvread("cyl_dir_mag_127.csv")));
x3 = abs(fftshift(csvread("cyl_dir_mag_152.csv")));
x4 = abs(fftshift(csvread("cyl_dir_mag_177.csv")));
x5 = abs(fftshift(csvread("cyl_dir_mag_202.csv")));
x6 = abs(fftshift(csvread("cyl_dir_mag_227.csv")));
x7 = abs(fftshift(csvread("cyl_dir_mag_252.csv")));

freq = ((0:(length(x7)-1))-(PACKAGE_SIZE/2)) ./ (0.001*PACKAGE_SIZE);

figure
hold on
% plot(freq, x1, 'color', [1 63/255 85/255])
% plot(freq, x2, 'color', [1 178/255 53/255])
plot(freq, x3, 'color', [1 74/255 186/255])
plot(freq, x4, 'color', [215/255 106/255 1])
plot(freq, x5, 'color', [116/255 116/255 1])
plot(freq, x6, 'color', [6/255 233/255 0])
plot(freq, x7, 'color', [0.0000 115/255 14/255])
hold off

legend('3.0V','3.5V','4.0V','4.5V','5.0V')
axis([0 250 0 50])
xlabel('Frequency (Hz)')
ylabel('Amplitude')
```

Multiplots.m

```
% The number of samples in a single package from the Arduino.
PACKAGE_SIZE = 128;
% The number of packages recieved from the Arduino.
PACKAGE_NUM = 1;
% The number of values in each sample from the Arduino.
SAMP_DIM = 3;

% x1 = abs(fftshift(csvread("mag_102.csv")));
% x2 = abs(fftshift(csvread("mag_127.csv")));
x3 = abs(fftshift(csvread("cyl_dir_mag_152.csv")));
x4 = abs(fftshift(csvread("cyl_dir_mag_177.csv")));
x5 = abs(fftshift(csvread("cyl_dir_mag_202.csv")));
x6 = abs(fftshift(csvread("cyl_dir_mag_227.csv")));
x7 = abs(fftshift(csvread("cyl_dir_mag_252.csv")));

% y1 = abs(fftshift(csvread("coin_direct_mag_102.csv")));
% y2 = abs(fftshift(csvread("coin_direct_mag_127.csv")));
y3 = abs(fftshift(csvread("noise1_mag_153.csv")));
y4 = abs(fftshift(csvread("noise1_mag_178.csv")));
y5 = abs(fftshift(csvread("noise1_mag_203.csv")));
y6 = abs(fftshift(csvread("noise1_mag_228.csv")));
y7 = abs(fftshift(csvread("noise1_mag_253.csv")));

% z1 = abs(fftshift(csvread("coin_noise_mag_102.csv")));
% z2 = abs(fftshift(csvread("coin_noise_mag_127.csv")));
z3 = abs(fftshift(csvread("noise_mag_153.csv")));
z4 = abs(fftshift(csvread("noise_mag_178.csv")));
z5 = abs(fftshift(csvread("noise_mag_203.csv")));
z6 = abs(fftshift(csvread("noise_mag_228.csv")));
z7 = abs(fftshift(csvread("noise_mag_253.csv")));

freq = ((0:(length(x7)-1))-(PACKAGE_SIZE/2)) ./ (0.001*PACKAGE_SIZE);

figure
subplot(3,1,1);
hold on
% plot(freq, x1, 'color', [1 63/255 85/255])
% plot(freq, x2, 'color', [1 178/255 53/255])
plot(freq, x3, 'color', [1 74/255 186/255])
plot(freq, x4, 'color', [215/255 106/255 1])
plot(freq, x5, 'color', [116/255 116/255 1])
```

```
plot(freq, x6, 'color', [6/255 233/255 0])
plot(freq, x7, 'color', [0.0000 115/255 14/255])
hold off

title('Direct Placement')
legend('3.0V','3.5V','4.0V','4.5V','5.0V')
axis([0 250 0 50])

subplot(3,1,2);
hold on
% plot(freq, y1, 'color', [1 63/255 85/255])
% plot(freq, y2, 'color', [1 178/255 53/255])
plot(freq, y3, 'color', [1 74/255 186/255])
plot(freq, y4, 'color', [215/255 106/255 1])
plot(freq, y5, 'color', [116/255 116/255 1])
plot(freq, y6, 'color', [6/255 233/255 0])
plot(freq, y7, 'color', [0.0000 115/255 14/255])
hold off

title('Dampened')
ylabel('Amplitude')
axis([0 250 0 1000])

subplot(3,1,3);
hold on
% plot(freq, z1, 'color', [1 63/255 85/255])
% plot(freq, z2, 'color', [1 178/255 53/255])
plot(freq, z3, 'color', [1 74/255 186/255])
plot(freq, z4, 'color', [215/255 106/255 1])
plot(freq, z5, 'color', [116/255 116/255 1])
plot(freq, z6, 'color', [6/255 233/255 0])
plot(freq, z7, 'color', [0.0000 115/255 14/255])
hold off

title('Off-target Stimulation')
axis([0 250 0 1000])
xlabel('Frequency (Hz)')
```

Pruebasonido.m

```
function varargout = pruebasonido(varargin)
% PRUEBASONIDO MATLAB code for pruebasonido.fig
```

```
%      PRUEBASONIDO, by itself, creates a new PRUEBASONIDO or raises the existing
%      singleton*.
%
%      H = PRUEBASONIDO returns the handle to a new PRUEBASONIDO or the handle to
%      the existing singleton*.
%
%      PRUEBASONIDO('CALLBACK',hObject,eventData,handles,...) calls the local
%      function named CALLBACK in PRUEBASONIDO.M with the given input arguments.
%
%      PRUEBASONIDO('Property','Value',...) creates a new PRUEBASONIDO or raises the
%      existing singleton*.  Starting from the left, property value pairs are
%      applied to the GUI before pruebasonido_OpeningFcn gets called.  An
%      unrecognized property name or invalid value makes property application
%      stop.  All inputs are passed to pruebasonido_OpeningFcn via varargin.
%qa
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help pruebasonido

% Last Modified by GUIDE v2.5 17-Apr-2019 21:19:32

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @pruebasonido_OpeningFcn, ...
                   'gui_OutputFcn',  @pruebasonido_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT
```

```
% --- Executes just before pruebasonido is made visible.
function pruebasonido_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to pruebasonido (see VARARGIN)

% Choose default command line output for pruebasonido
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes pruebasonido wait for user response (see UIRESUME)
% uiwait(handles.figure1);


% --- Outputs from this function are returned to the command line.
function varargout = pruebasonido_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;


% --- Executes on button press in load.
function load_Callback(hObject, eventdata, handles)
% hObject    handle to load (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
myGui=guidata(handles.figure1);
%[file,path] = uigetfile('*.mp3','Seleccione un archivo de sonido');
%[x,fs] = audioread([path file]);
fs = 20500;
myGui.freqSam=fs;
freq = get(handles.slider1,'value');
```

```
amp = get(handles.slider2,'value');
%t = linspace(0, fs*2, fs*8);
%s = amp*sin(2*pi*t*freq);
duration=5;
values=0:1/fs:duration;
s = amp*sin(2*pi*freq*values);

left = get(handles.checkbox1, 'Value');
right = get(handles.checkbox2, 'Value');
if left == 1 && right ==0
   Out = [ s; zeros(size(values)) ]';%zeros(size(t))
elseif right == 1 && left ==0
   Out = [ zeros(size(values)) ; s ]';
else
   Out = [ s ; s ]';
end

%Out = [  zeros(size(values));s ];%zeros(size(t))
myGui.datasound=Out;
myGui.player=audioplayer(myGui.datasound,myGui.freqSam);
myGui.flag=2;
guidata(handles.figure1,myGui)


% --- Executes on button press in play.
function play_Callback(hObject, eventdata, handles)
% hObject    handle to play (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
myGui=guidata(handles.figure1);
if(myGui.flag==2)
   myGui.flag=1;
   disp('2');
   play(myGui.player);
else
   if(myGui.flag == 1)
      disp('1');
      myGui.flag=0;
      pause(myGui.player);
   else
      disp('0');
      myGui.flag=1;
```

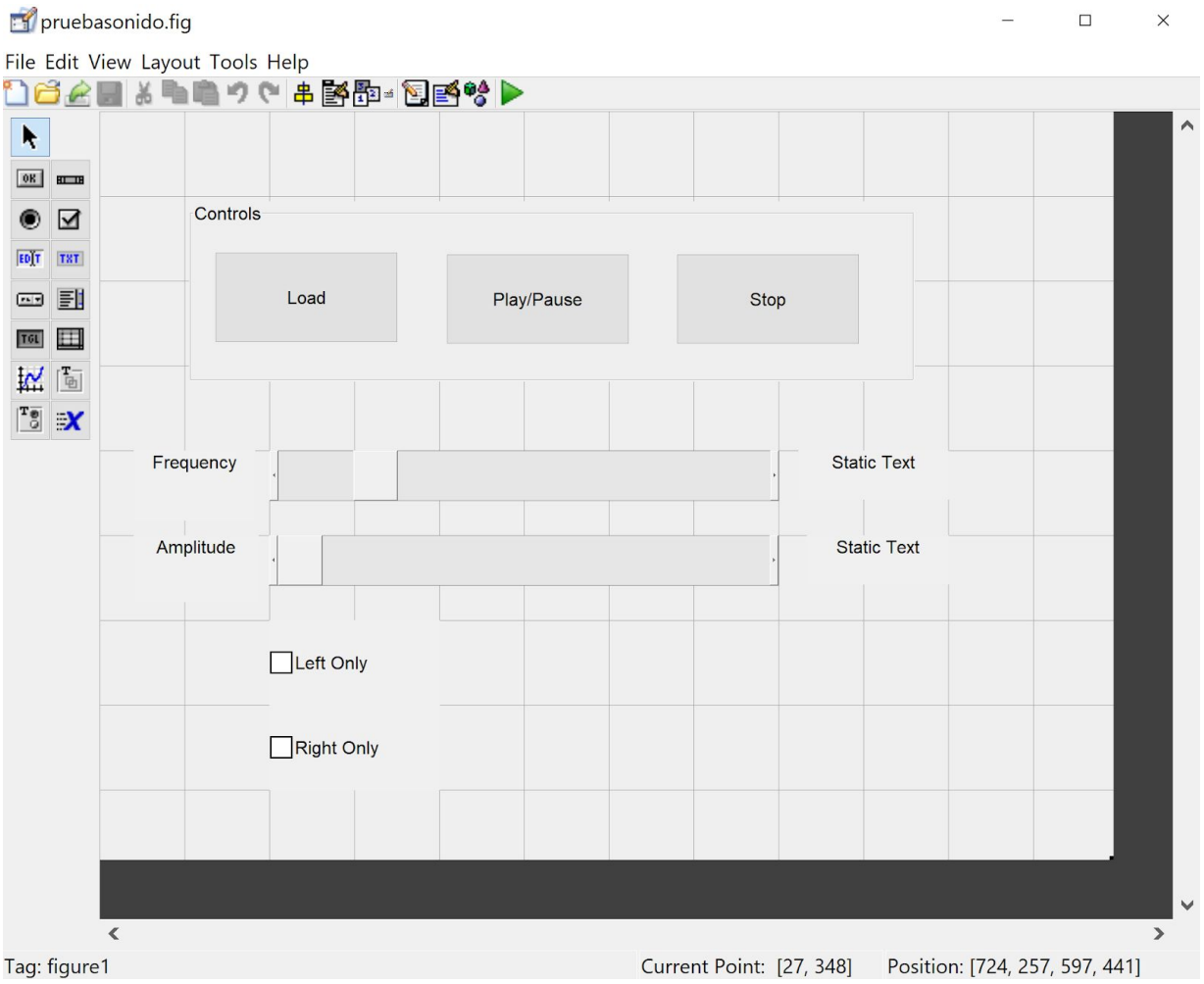
```
        resume(myGui.player)
    end
end
guidata(handles.figure1,myGui);



% --- Executes on button press in stop.
function stop_Callback(hObject, eventdata, handles)
% hObject    handle to stop (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
myGui=guidata(handles.figure1);
myGui.flag=2;
stop(myGui.player);
guidata(handles.figure1,myGui);



% --- Executes on slider movement.
function slider2_Callback(hObject, eventdata, handles)
% hObject    handle to slider2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'Value') returns position of slider
%        get(hObject,'Min') and get(hObject,'Max') to determine range of slider
b = get(handles.slider2,'value')
set(handles.text2,'String',num2str(b));

% --- Executes during object creation, after setting all properties.
function slider2_CreateFcn(hObject, eventdata, handles)
% hObject    handle to slider2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: slider controls usually have a light gray background.
if isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor',[.9 .9 .9]);
end
```

```
% --- Executes on slider movement.
function slider1_Callback(hObject, eventdata, handles)
% hObject    handle to slider1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'Value') returns position of slider
%        get(hObject,'Min') and get(hObject,'Max') to determine range of slider
a = get(handles.slider1,'value')
set(handles.text1,'String',num2str(a));

% --- Executes during object creation, after setting all properties.
function slider1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to slider1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: slider controls usually have a light gray background.
if isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor',[.9 .9 .9]);
end



% --- Executes on button press in checkbox1.
function checkbox1_Callback(hObject, eventdata, handles)
% hObject    handle to checkbox1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of checkbox1


% --- Executes on button press in checkbox2.
function checkbox2_Callback(hObject, eventdata, handles)
% hObject    handle to checkbox2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
```

% handles   structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of checkbox2

Pruebasonido.fig



C. Figures and Tables

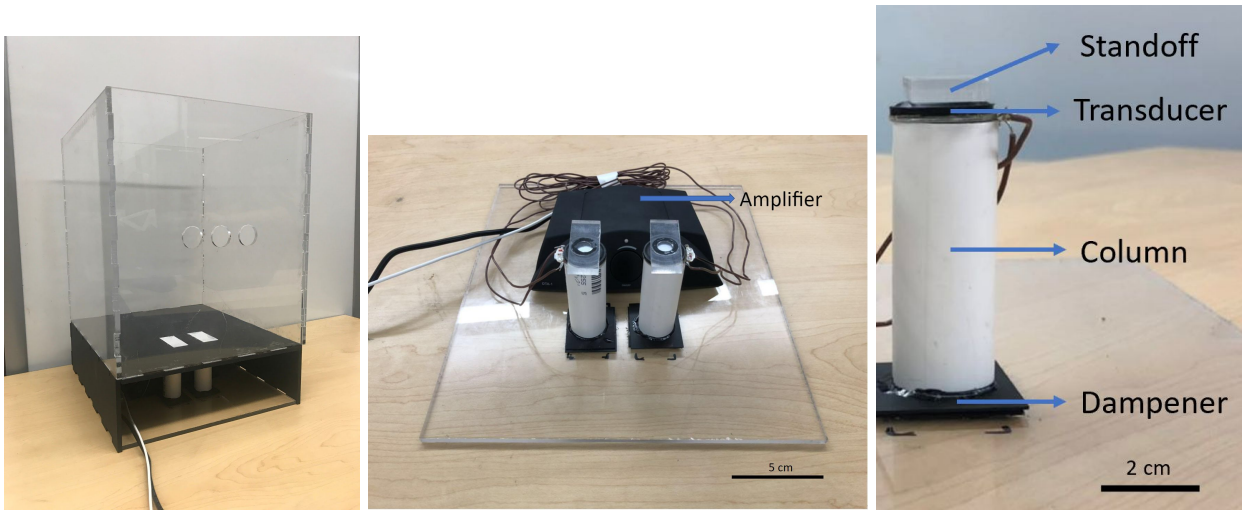


Figure 1 (Left). The finished device. Features: top enclosure with visible placement marks (white) for the rodent's hind limbs and nose holes through which the rat can nose poke based on stimulus and training received. The electronics assembly is housed underneath. **(Middle).** The electronics assembly which includes an amplifier and pedestals with transducer. **(Right).** Expanded view of the pedestal assembly featuring an acrylic standoff, transducer, ¾" PVC column filled with concrete, and isoprene rubber as a dampener.

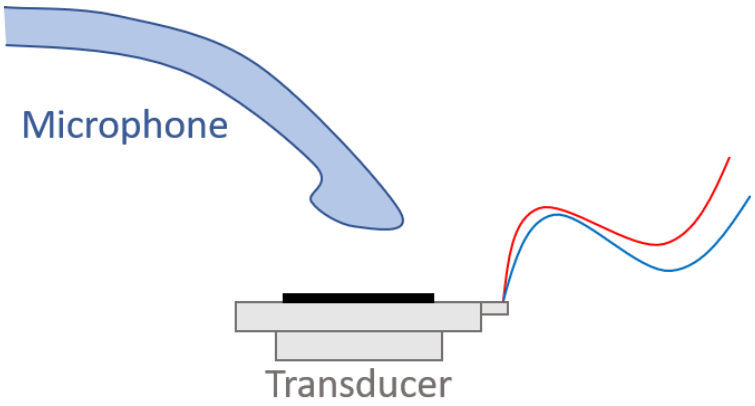


Figure 2. Testing setup of frequency test. A microphone is used to collect the signal from the transducer. A MATLAB code is then used to verify the output frequency matches input frequency through FFT.

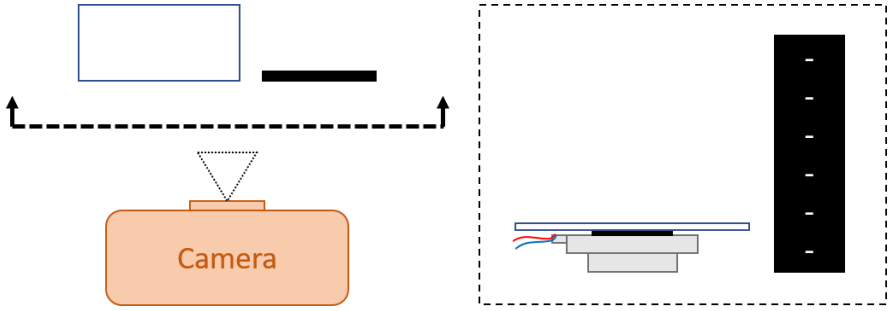


Figure 3. Testing setup of displacement test. A camera is used to record the videos of a vibrating transducer. A ruler (black) is placed aside to indicate scale.

Figure 4. Testing setup of load cell test. The load cell (blue) is placed on either the vibrating pedal or the non-vibrating pedal. The force generated by the transducer is collected and analyzed.

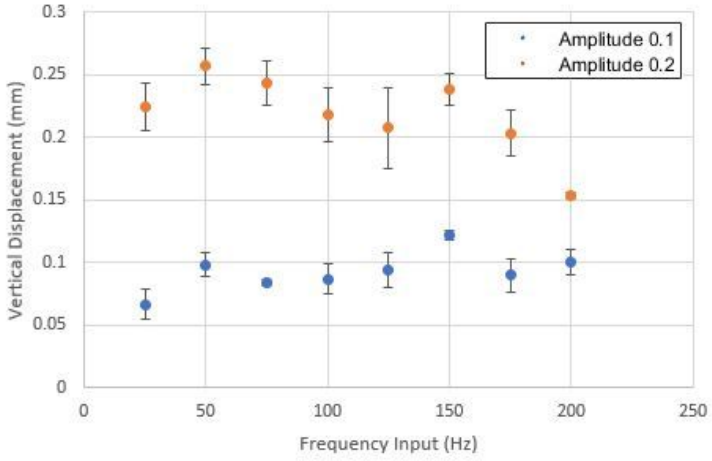


Figure 5. Relationship between displacement and frequency. The vertical displacement at an amplitude of 0.1 and 0.2 was computed at frequencies between 25 Hz and 200 Hz at 25 Hz intervals (n=3 for each trial).

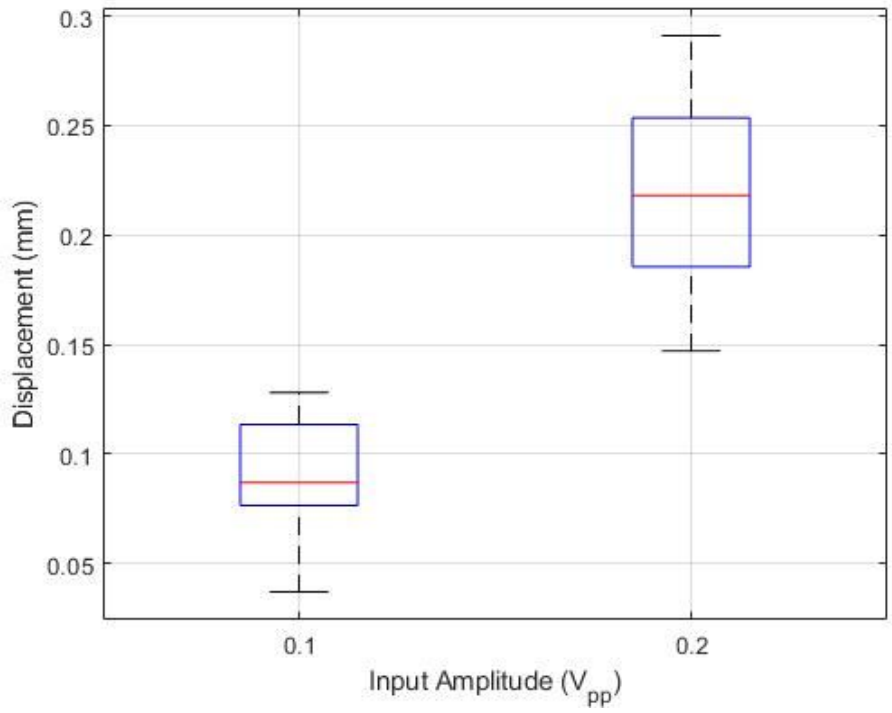


Figure 6. Relationship between displacement and voltage. A summary of the results of the uncoupled displacement measurements of the transducer. These results are sampled at a range of different frequencies (25-200 Hz), and highlights the statistical results of figure 3.

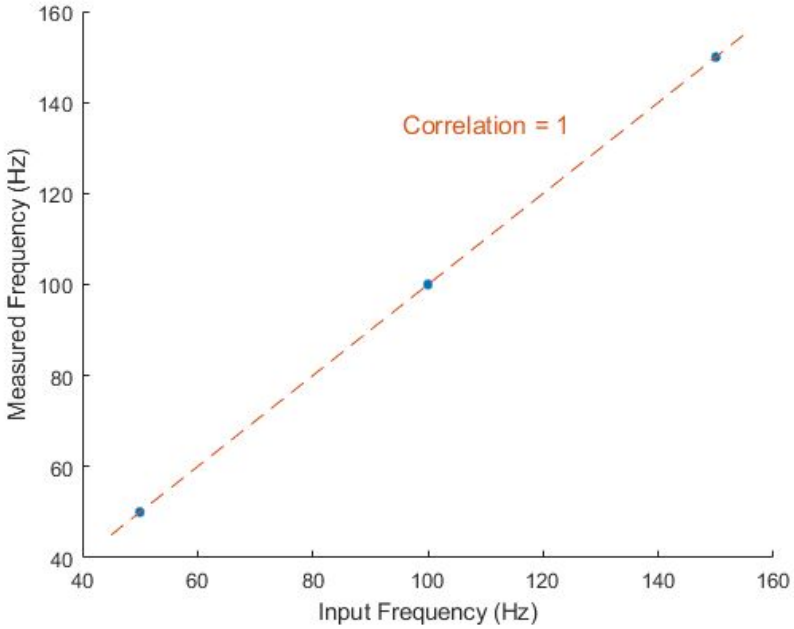


Figure 7. Frequency verification. Samples were taken at 50 Hz, 100 Hz, and 150 Hz (n=5 for each). There is a positive correlation between the input frequency and the measured frequency.

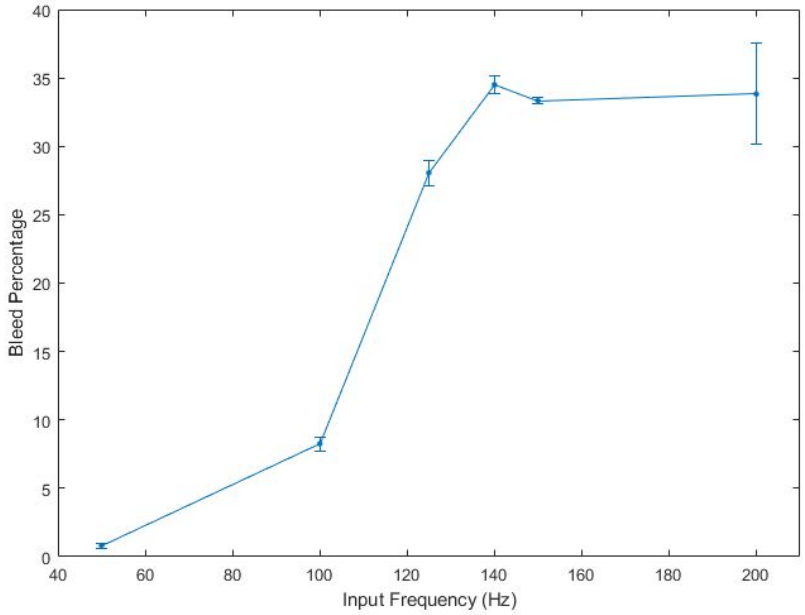


Figure 8. Bleed between platforms. The percent bleed between the two platforms was calculated at several frequencies (n=5 for each). There is a noticeable drop off in percent bleed between 100 Hz and 125 Hz. Error bars represent ± 1 SE.

D. BME 400 Report

Materials

A plexiglass enclosure was used to house the rodent. This allowed easy cleaning and being able to view the animal during the procedure. Currently, the vibrational platform is 3D printed SLA using a Form 2 3D printer. Two different sources of vibration were used (uxcell a11101800ux0165 DC 3V 1100RPM 0.2A High Speed Mini Vibration Motor for DIY Toys, Adafruit VIBRATION MOTOR 11000 RPM 5VDC) [8, 9]. A rubber adhesive pad covers the platform cutouts and provides adhesion for the platform. Future iterations of this prototype will ensure material is both waterproof and easy to sanitize.

Methods

Circuit and Arduino code has been developed to control the motor such that it can generate a vibration at desired frequency and amplitude. The motor itself is attached to the bottom of the platform. Two of such platforms are integrated into the bottom of a cage, which is large enough to hold one rat. The whole cage and the platforms are made from waterproof materials as the rats may urinate during the experiments.

Experimental Setup

The final prototype consists of a plexiglass rodent enclosure with two platform cut-outs designed to accommodate the hindlimbs of a rearing rat. The cut-ous are covered with a foam adhesive, underneath which the platform is attached to. The platform itself houses both the linear resonant actuator (LRA) and the eccentric rotating mass (ERM) motor (Figure 8.)

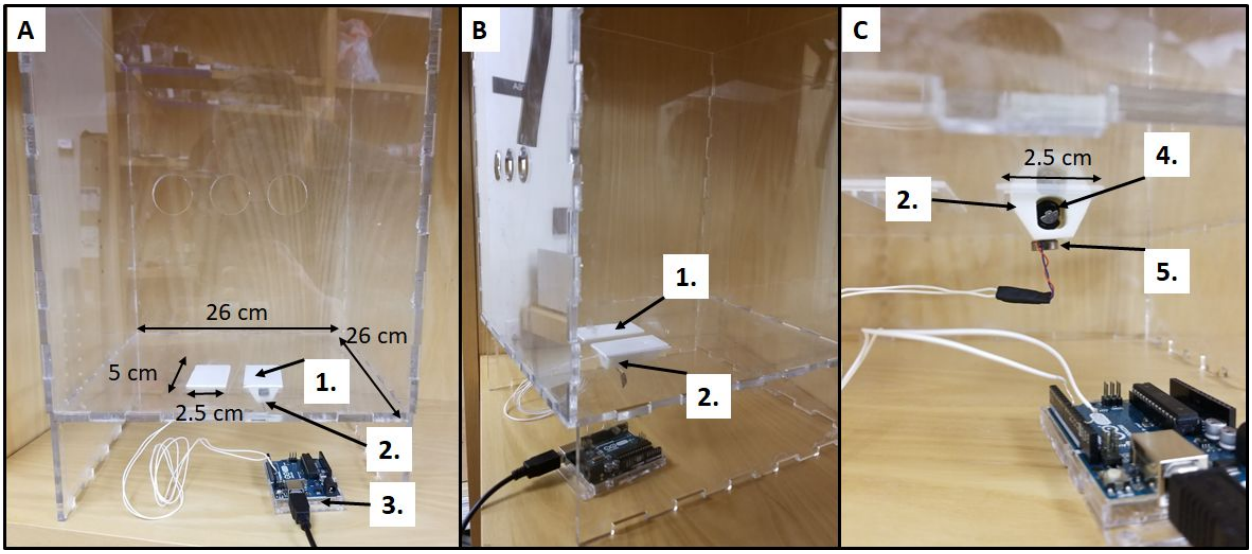


Figure 1. Rodent enclosure from various angles and enlargements. **(A)** The rodent enclosure in a front-on view. Visible are the two white foam covered cutouts on which a rodent will be trained to place its hind limbs while rearing (1.), and the platform housing the vibrational motors (2.) An Arduino microcontroller controls the vibrational frequency of the motors (3.) **(B)** Angled view displaying the cut-outs (1.) and motor(2.) **(C)** Detailed front view featuring the platform housing the vibrational motors (2.), the ERM cylindrical motor(4.) and the LRA coin motor (5).

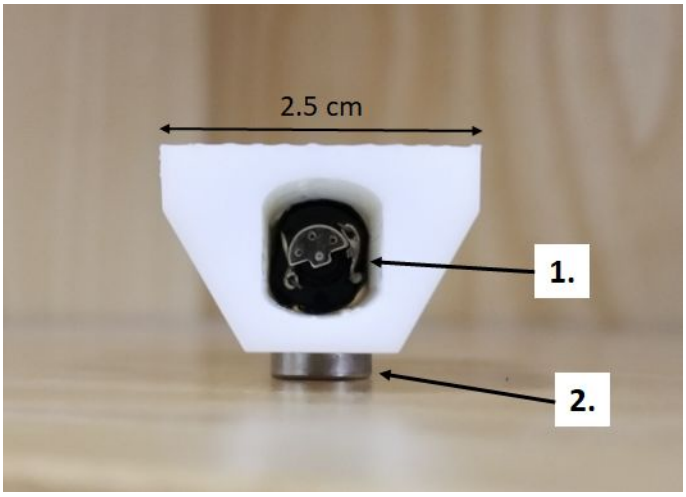


Figure 2. Close-up of the platform housing the vibrational motors. The ERM cylindrical motor (1.) is nestled in the middle of the platform. The off-centered rotating mass responsible for the vibrations is visible from this angle. The LRA coin motor (2.) is attached to the bottom of the platform. Together these motors cover the prescribed frequency range.

Testing

An experiment was conducted to determine the response of the LRA motor and ERM motor under different conditions. Each of the two motors were placed into three different positions with an accelerometer (Figure 8). The direct placement configuration was used to determine the motor's response to different voltages, since there may be discrepancies in actual performance from the datasheet. The dampened configuration was used to determine what the rodent might perceive from the leg being stimulated by the motor during use of the cage. The off-target configuration was used to determine what the rodent might perceive from the leg not being stimulated during use of the cage.

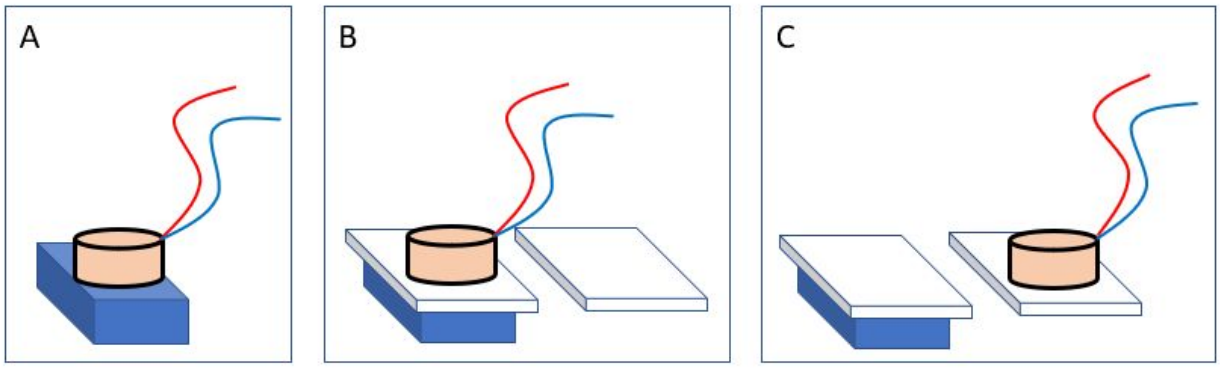


Figure 3. The configurations of the accelerometer (blue) and the motor (beige) used in our experiment. (A) The direct placement configuration. (B) The dampened configuration. There is a foam pad (white) separating the accelerometer and the motor. (C) The off-target stimulation configuration. The motor is placed on the adjacent foam pad to the accelerometer.

In each configuration, a motor was vibrated at various voltages. The voltages tested in each motor spanned the range specified in the motor's datasheet at 0.5 V increments. Each

motor was powered using a PWM signal with a max voltage of 5 V. Thus, a 2.5 V signal in this experiment corresponds to a duty cycle of 50% and a 3.5 V signal corresponds to a 70% duty cycle.

Data was collected from the accelerometer using an Arduino UNO and was sent to MATLAB for processing (Appendix B). First, the motor was powered and 128 samples of the acceleration vector were collected from the accelerometer at 1 kHz. This number of samples was chosen due to memory limitations of the Arduino UNO. Second, the data was imported to MATLAB. Third, the magnitude of the acceleration vector was computed. Fourth, the Fourier transform of the accelerometer magnitude was computed using the MATLAB fast Fourier transform [7]. This method was used rather than sending the data one sample at a time because literature suggests that our method leads to more accurate data [10].

Results

Testing demonstrated when the motors are directly placed on the accelerometer, the measured frequency generated by ERM (60 Hz, 175 Hz) is slightly higher than that generated by LRA (50 Hz, 150 Hz). When the motors are damped by the platform, the range of frequencies generated by both motors shrinks (100-150 Hz for LRA and 75-125 Hz for ERM). Moreover, both of the motors show an acceptable off-target vibration with an amplitude less than 25% of the amplitude obtained from the damped motor test.

Table 1. The results from the Fourier Transforms run on the different different motors at the different locations. The motor dampened by platform results are the most relevant to the frequencies that will be transferred to the legs of the rats during testing. The results from the Transforms show that the motors can achieve frequencies of approximately 75 Hz and 100 - 150 Hz.

	LRA	ERM
Direct Placement of accelerometer on motor	50 Hz, 150 Hz	60 Hz, 175 Hz
Motor damped by platform Accelerometer placed on platform	100 - 150 Hz	75, 125 Hz
Amplitude of off-target vibrations felt on adjacent pad *	25%	10% - 25%
* Based on selected prominent frequencies		

Beside frequency, another important factor that describes how an object vibrates is amplitude. In this case, the amplitude of device's vibration is directly correlated to the force generated by the Micro Vibration Motor with a eccentric disk. Since the inner structure of LRA is not accessible, only the force generated by ERM can be calculated, using the formula below:

$$F = m\, r\, \omega^2$$

In this formula, m is mass of eccentric disk, obtained from a micro scale. r is the radius of the orbit that the eccentric disk spins on. This is obtained by calculating the distance between the spinning rod and the center of mass of the eccentric disk. ω is the angular velocity

calculated from the motor's speed in round per second. Finally, F is for force generated by the motor and the result is 3.73e-4 N.

With force generated by the motor, the acceleration of the device can be determined using Newton's Second Law of Motion, shown below:

$$F = m\,a$$

F is the generated force and m is the weight of whole device, including the motor and a pedal. The weight of these to parts can be obtained using a micro scale. The calculated acceleration of the device is 0.02243 m/s^2, which is 2.289e-3 G.

Discussion

Engaging a rodent by applying a vibrational stimulus is a far more humane method of stimulation than subjecting the animal to a painful stimulus. Conventionally, rodents have been stimulated by the use of heat, cold or electric shock. The current prototype emits vibrations at 75, 100 and 125 Hz. The amplitude at these vibrational frequencies is limited by the weight of the platform. Furthermore, the amplitude in an ERM motor is coupled to the rotational velocity. It would be beneficial to create a device that can overcome these limitations to fully explore the parameter space of both frequency and amplitude in vibrotactile stimulation. At the date of publication no known studies exist examining frequency and amplitude independarly as a method for hindlimb somatosensation in rodents. Given these limitations, such a device, would lead to novel findings and help further research in multiple fields. This information would additionally benefit the project by identifying ideal stimulation parameters, as well as establish threshold parameters. Threshold parameters will be needed to ensure that the stimulation at one site does not cause off-target stimulation and/or perception at an adjacent site.

Currently, our data is only preliminary. A greater sample size will be needed to establish confidence intervals for the frequencies and amplitudes the device can produce. Additionally, the statistical power of the experiment should, by definition, be calculated prior to engaging in further data collection to ensure our tests correctly reject the statistical null hypothesis which states that vibrational frequencies are the same at various voltages.

Future work will include performing animal testing with our cage. The rats will be trained to position itself with one foot standing on one platform and one foot on the other. The will be 3 holes on the side of the cage. The rats will be trained to use these holes to tell whether they feel the vibration and which foot. If they feel the vibration through their left foot, they will put their nose into the left hole, and right hole for right foot. If the rat is not sure or feel nothing, they will put their noses into the middle hole. Researchers can use this method to determine the function of the reconnected sciatic nerve of the rats.

References

[1] E. Fernandez *et al.*, “Effects of L-carnitine, L-acetylcarnitine and gangliosides on the regeneration of the transected sciatic nerve in rats,” *Neurol. Res.*, vol. 11, no. 1, pp. 57–62, Mar. 1989.

[2] A. S. P. Varejão, P. Melo-Pinto, M. F. Meek, V. M. Filipe, and J. Bulas-Cruz, “Methods for the experimental functional assessment of rat sciatic nerve regeneration,” *Neurol. Res.*, vol. 26, no. 2, pp. 186–194, Mar. 2004.

[3] L. B. Dahlin, “Techniques of peripheral nerve repair,” *Scand. J. Surg. SJS Off. Organ Finn. Surg. Soc. Scand. Surg. Soc.*, vol. 97, no. 4, pp. 310–316, 2008.

[4] S. Rotshenker, “Wallerian degeneration: the innate-immune response to traumatic nerve injury,” *J. Neuroinflammation*, vol. 8, p. 109, Aug. 2011.

[5] E. A. Huebner and S. M. Strittmatter, “Axon Regeneration in the Peripheral and Central Nervous Systems,” *Results Probl. Cell Differ.*, vol. 48, pp. 339–351, 2009.

[6] J. K. Terzis, B. Faibisoff, and B. Williams, “The nerve gap: suture under tension vs. graft.,” *Plast. Reconstr. Surg.*, vol. 56, no. 2, pp. 166–170, 1975.

[7] S. Yeomans, “Sciatic Nerve and Sciatica,” *Spine-health*. [Online]. Available: https://www.spine-health.com/conditions/sciatica/sciatic-nerve-and-sciatica. [Accessed: 28-Apr-2019].

[8] J. R. Deuis, L. S. Dvorakova, and I. Vetter, “Methods Used to Evaluate Pain Behaviors in Rodents,” *Front. Mol. Neurosci.*, vol. 10, 2017.

[9] R. T. Green, “Threshold for electric shock of the laboratory rat,” *Anim. Behav.*, vol. 6, no. 1, pp. 72–76, Jan. 1958.

[10] D. Purves *et al.*, Eds., *Neuroscience*, 6 edition. New York: Sinauer Associates is an imprint of Oxford University Press, 2017.

[11] A. M. Garner, J. N. Norton, W. L. Kinard, G. E. Kissling, and R. P. Reynolds, “Vibration-induced Behavioral Responses and Response Threshold in Female C57BL/6 Mice,” *J. Am. Assoc. Lab. Anim. Sci. JAALAS*, vol. 57, no. 5, pp. 447–455, Sep. 2018.

[12] J. N. Norton, W. L. Kinard, and R. P. Reynolds, “Comparative Vibration Levels Perceived Among Species in a Laboratory Animal Facility,” *J. Am. Assoc. Lab. Anim. Sci. JAALAS*, vol. 50, no. 5, pp. 653–659, Sep. 2011.

[13] K. N. Rabey, Y. Li, J. N. Norton, R. P. Reynolds, and D. Schmitt, “Vibrating Frequency Thresholds in Mice and Rats: Implications for the Effects of Vibrations on Animal Health,” *Ann. Biomed. Eng.*, vol. 43, no. 8, pp. 1957–1964, Aug. 2015.

[14] R. R. Fay, *Hearing in Vertebrates: A Psychophysics Databook*. Winnetka, Ill: Hill-Fay Assoc, 1988.

[15] “Buyers Guide - Exciters.” [Online]. Available: https://www.parts-express.com/resources-buyers-guide-exciters. [Accessed: 28-Apr-2019].

[16] Parts Express, “‘Dayton Audio DTA-1 Class D AC/DC Battery Powered Mini Amplifier 15 WPC’ from www.parts-express.com!” [Online]. Available: https://www.parts-express.com/dayton-audio-dta-1-class-d-ac-dc-battery-powered-mini-amplifier-15-wpc--300-380. [Accessed: 30-Apr-2019].

[17] Dayton Audio, “DAEX13CT-8 Coin Type 13mm Exciter 3W 8 Ohm.” [Online]. Available: http://www.daytonaudio.com/index.php/daex13ct-8-coin-type-13mm-exciter-3w-8-ohm.html. [Accessed: 30-Apr-2019].

[18] Vijayan, V, “Material Selection of Compliant Mechanism for Vibration Isolation,” *Mech.*

Mech. Eng., vol. 18, no. 2, pp. 121–134, 2014.
[19] “Mini Load Cell - 100g, Straight Bar (TAL221) - SEN-14727 - SparkFun Electronics.” [Online]. Available: https://www.sparkfun.com/products/14727. [Accessed: 30-Apr-2019].
[20] “SparkFun Load Cell Amplifier - HX711 - SEN-13879 - SparkFun Electronics.” [Online]. Available: https://www.sparkfun.com/products/13879. [Accessed: 30-Apr-2019].
[21] “Load Cell Amplifier HX711 Breakout Hookup Guide - learn.sparkfun.com.” [Online]. Available: https://learn.sparkfun.com/tutorials/load-cell-amplifier-hx711-breakout-hookup-guide?_ga=2.116053509.2015357144.1556677851-1375936112.1556677851. [Accessed: 30-Apr-2019].
[22] “SparkFun Triple Axis Accelerometer Breakout - ADXL335 - SEN-09269 - SparkFun Electronics.” [Online]. Available: https://www.sparkfun.com/products/9269. [Accessed: 30-Apr-2019].
[23] A. Hjort and M. Holmberg, *Measuring mechanical vibrations using Arduino as a slave I/O to an EPICS control system*. 2015.