BIRD BOX PROJECT FINAL REPORT

May 2, 2019

Kevin Chen, María Nacenta Fernández, Michael Zhang
ECE 445 - Spring 2019
TA: Christopher Horn
Team 39

1 Abstract

The Bird Box Project is a project that is sponsored by University of Illinois at Urbana-Champaign Graduate Student Researcher Shelby Lawson for the purpose of streamlining trials conducted for bird behavior research. The project consists of both hardware and software components. The hardware of this project is designed to both allow the bird to provide responses to audio stimuli, reward/punish according to those responses, and monitor the bird. The software of this project is designed as a user interface easy-to-use for the researcher conducting the trials. It is designed to receive sound file inputs as well as trial parameters to compile into a cohesive trial program. At the end of the trial, the software will output a csv/excel sheet that records all the activities that occurred during the trial start. The Bird Box Project is optimized to alleviate hours of extraneous work on the researcher while maintaining cost-efficiency.

Contents

2 Introduction

2.1 Objective

Observation of animal behaviors and responses to certain audio stimuli can become the backbone for how our technologies are shaped or how our secret codes are constructed. Researchers at the University of Illinois are working to uncover patterns from these behaviors among birds through the means of conditioning. However, researchers are faced with a dire problem - there is no existing system that would perfectly cater towards their research needs and is cost-efficient. Thus, the need to build a system suitable for their research necessities becomes increasingly apparent.

The solution would be a system comprised of a a hardware and software interface. The software interface will accept parameters fed through a Graphical User Interface (GUI) to construct a unique file (with a certain number of trials) set by the researcher. The hardware would then respond to the data provided by the research in the GUI to play audio sounds for the bird to respond to. The bird would provide responses though color-differentiated buttons and trigger certain outcomes from the system. The bird would solely interact with the hardware side of the system. The system will be designed to reward the bird with food upon favorable action and punish the bird by turning the lights off upon unfavorable action. At the end of the research period, an excel sheet wold be generated documenting the results of each unique response that the bird provided.

2.2 Background

Modern technology has evolved at an incredible rate and digital signal processing is no exception to this rapid growth. With this growth in technology, it is important to also observe natural aspects regarding the field to draw more inspiration for advancements in signal processing. Thus, the analysis of bird behavior and responses to certain audio stimuli becomes a valuable observation for furthering knowledge in this field of study.

To highlight the problem, there will be varying tiers of impacts to provide emphasis on the scale in which the project contribution can help with understanding this field. Creating a product to suit the needs of researchers will unlock further contribution towards various insights within the field. The trials the system would help conduct further enables understanding of bird communication, which can be applied to save certain endangered specifies upon identifying a certain cry - from a bird, or, in a broader sense, this can contribute towards how language is perceived among birds - how the communicate amongst each other and how certain sounds are assigned meanings [1].

2.3 Experiment Procedure and Various Terminology

The experiment is composed of a variable amount of trials specified by the researcher. Each trial can be classified according to if can be called a sham. A legitimate trial will eventually play an audio track that differentiates from the background sound while a sham trial will never play this differentiating cue and continue to play the background sound until the next trial is started.

2.4 High Level Requirements

- The overall device must dispense food to the tested subject within a latency period of 2 seconds of successful trial completion - A trial is counted as successful if the trial is not designated as a sham and the subject correctly presses the Trial Attempt Button after a specified audio cue.
- The overall device must shut of cage lighting in the event that the tested subject failed a trial. Trial failure is defined strictly as the subject pressing the Trial Attempt Button during a trial specified as a sham.
- The device must not punish inaction from the subjects end, and, in the event of extended inaction, must alert the researchers that the trials have failed to initiate before shutting itself off (leaving a light on to not stress the bird).
- The software must be capable of accepting .wav file inputs for audio signals and produce an excel sheet briefing the results from the subject-system interaction as an output.
- The project must be cost-efficient and within a $500 budget as existing devices exist for a far more extravagant price.

3 Hardware Design

The hardware component can be broken into four major modules as shown in Figure 1. Additionally, all hardware, with exception of the computer speakers, will be inside of the testing environment. It is assumed that a host-computer will always be within proximity of the testing environment to support our block diagram. We have informed our sponsor of this development and have reached an agreement with regards to this.

The connections in Figure 1 do not necessarily denote data buses for data connections or the data and power outputs managed by a singular input. For the sake of simplicity and compactness, the block diagram merely denotes a connection between one node to another with the connection intent denoted. Specifics of each connection are purposefully left unspecified and are detailed in greater length in each individual section via schematic diagram.

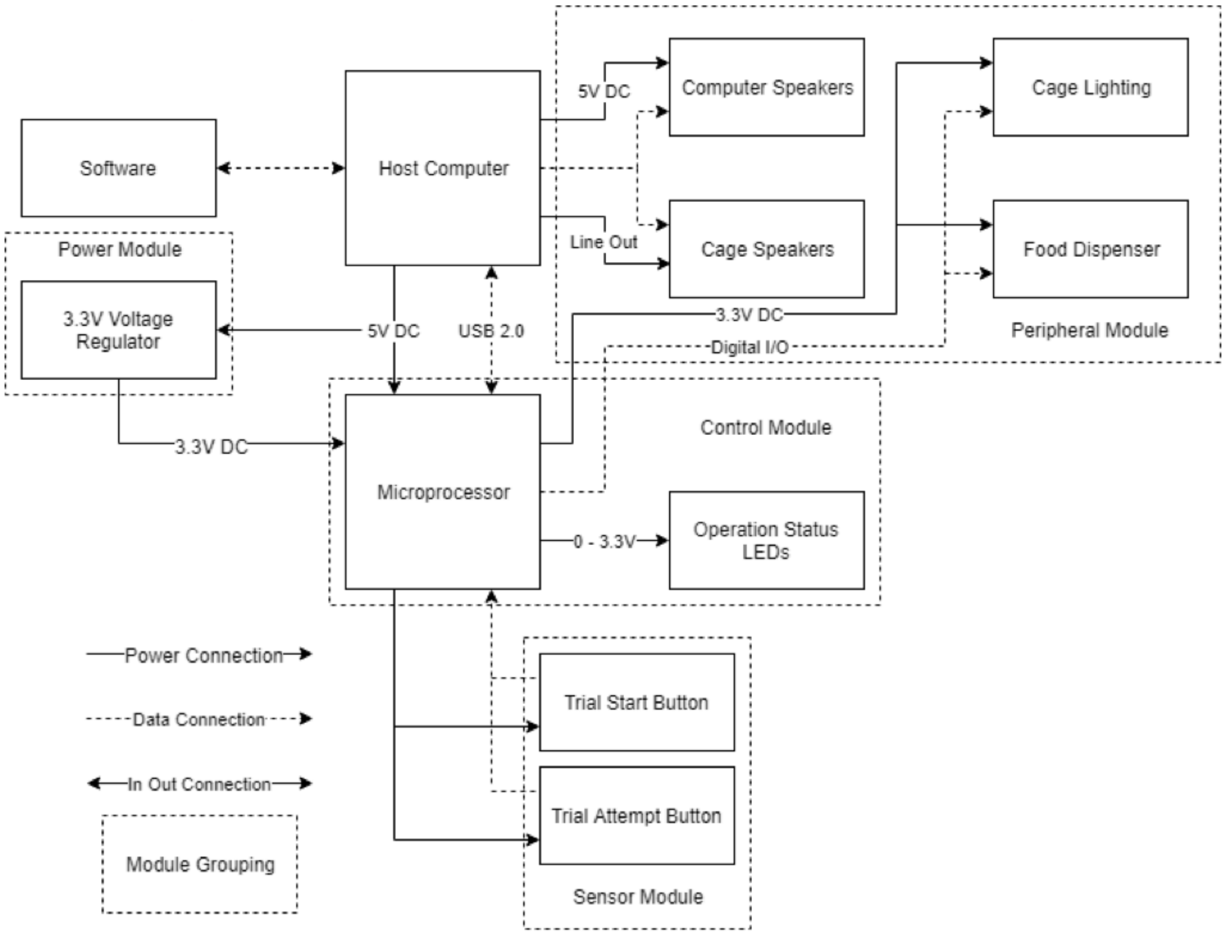

Figure 1: Block Diagram of Hardware Modules and Component Nodes

3.1 Control Module

The control module handles communication between the onboard devices as well as connected devices and software. It is powered by the power module and will disable itself and the device if voltage and current are significantly out of operation bounds or if device temperature is not within safe levels for the chip. This module consumes approximately 50mA $\pm$ 5% between its blocks.

3.1.1 Microcontroller

This project is designed to be used with the microcontroller ATmega16U2 because of its data retention (20yr 85C) as well as its ability to communicate with a USB 2.0 interface. This chip contains two SPI interfaces for required peripherals and a singular UART interface. Additionally, the microprocessor monitors onboard voltage and current and will disable functionality if out of operational bounds. 16kB of on chip flash allows for programming without an external NAND flash chip. But this microcontroller has a big inconvenient, it is a TQFN package which means that this microcontroller instead of leads it has pads under it and it is very small with a size of 9x9mm [4]. This makes soldering it really difficult and it needs special tools that we did not have in our laboratory. The final schematic for the final project will be as shown in Figure 2.

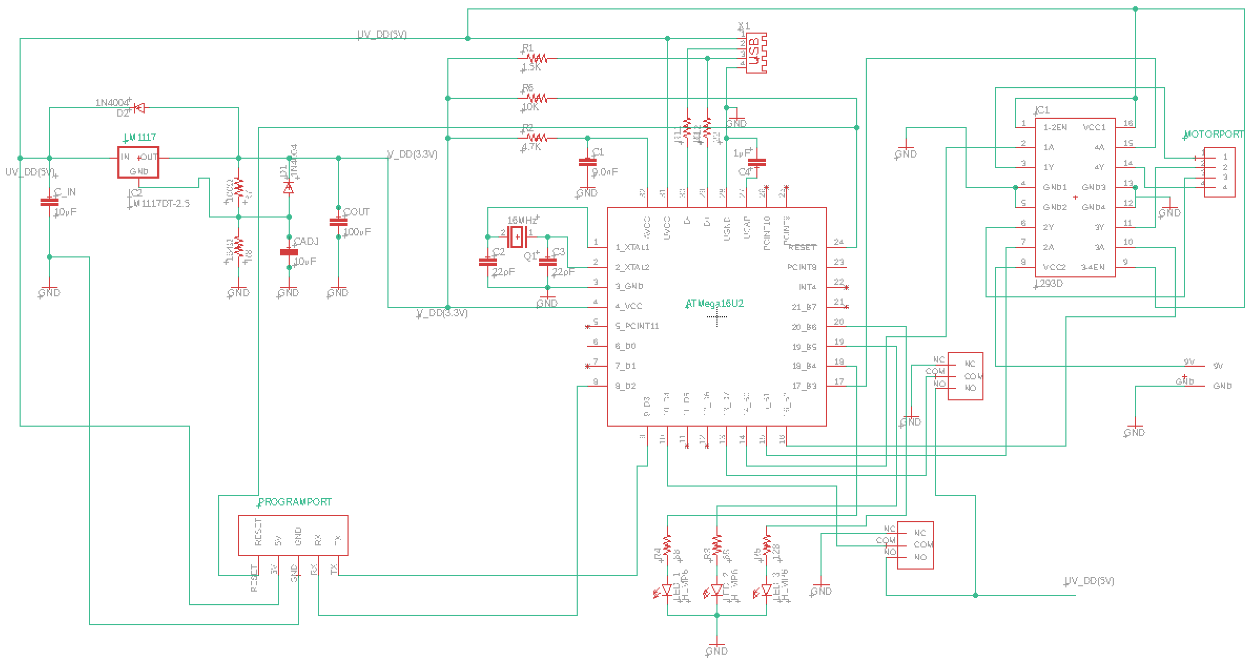

Figure 2: Final schematic of the PCB. Using ATmega16U2

The design of our final PCB will be the one shown in Figure 3. The PCB has a with of 10mil for all the wires because the current will not be higher than 1A in any party of the circuit. This PCB has two layers and the vias used have a drill size of 13.77953 enough for the width of the wires. Something important to know is that all the connections between the USB and the microcontroller must be short, with similar lengths and near as possible. To end with the PCB we have done an isolation of the PCB to ground with 50mil as it can be seen in Figure 3.

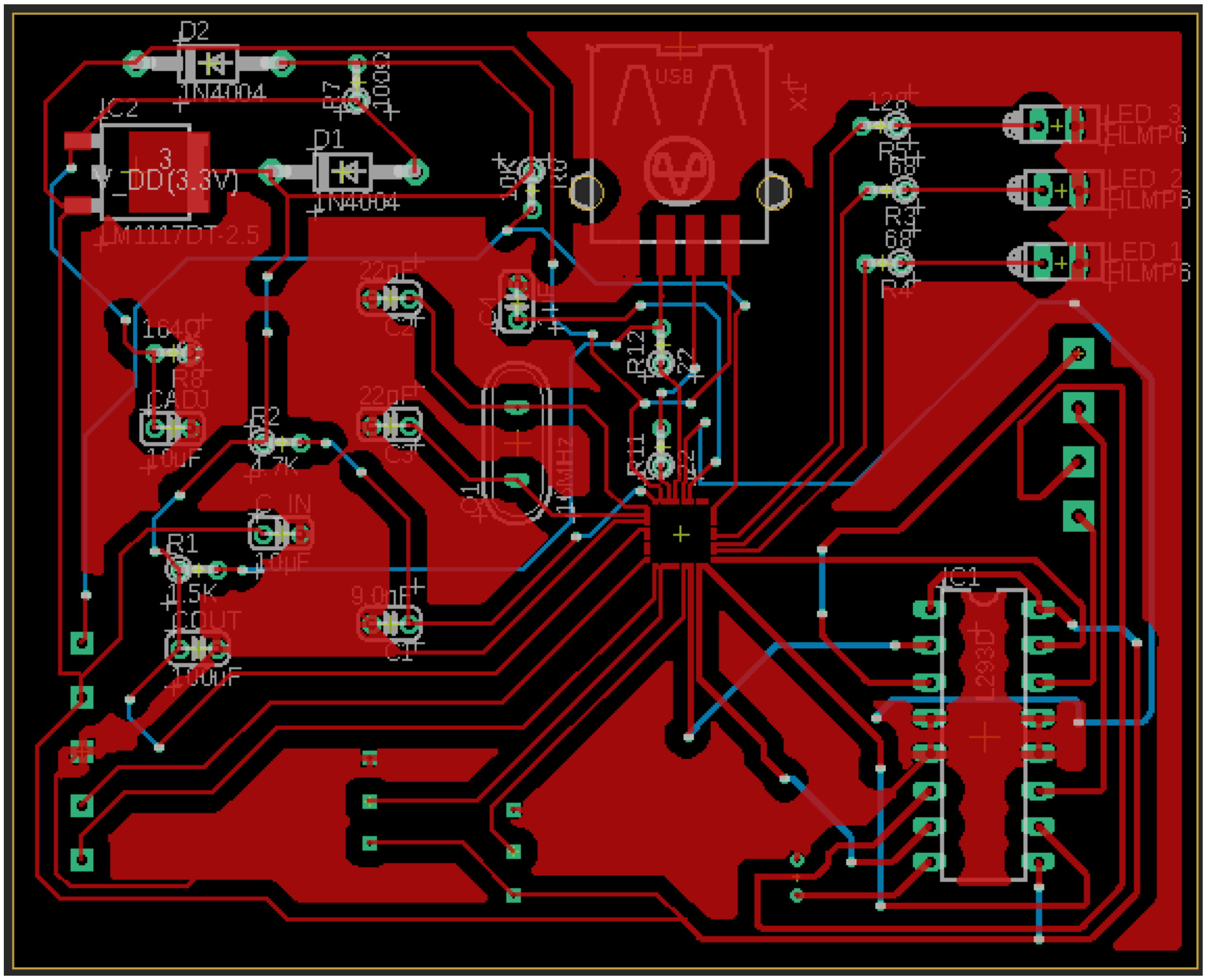

Figure 3: Final design for the PCB

For our prototype we decided that the best way to demonstrate how our project would work with the tools at our disposal was by using an ATmega328 [3]. This microcontroller is able to handle processing commands from both input sensors from the sensor module, as well as commands from software to active components in the peripheral module. The communication with the sensor module is done with UART and the communication with the peripheral module is done with SPI. It is because of this that we finally need to use a protoboard instead of our PCB. The schematic of the cirtuit of the protoboard for the prototype will be the one shown in Figure 4.

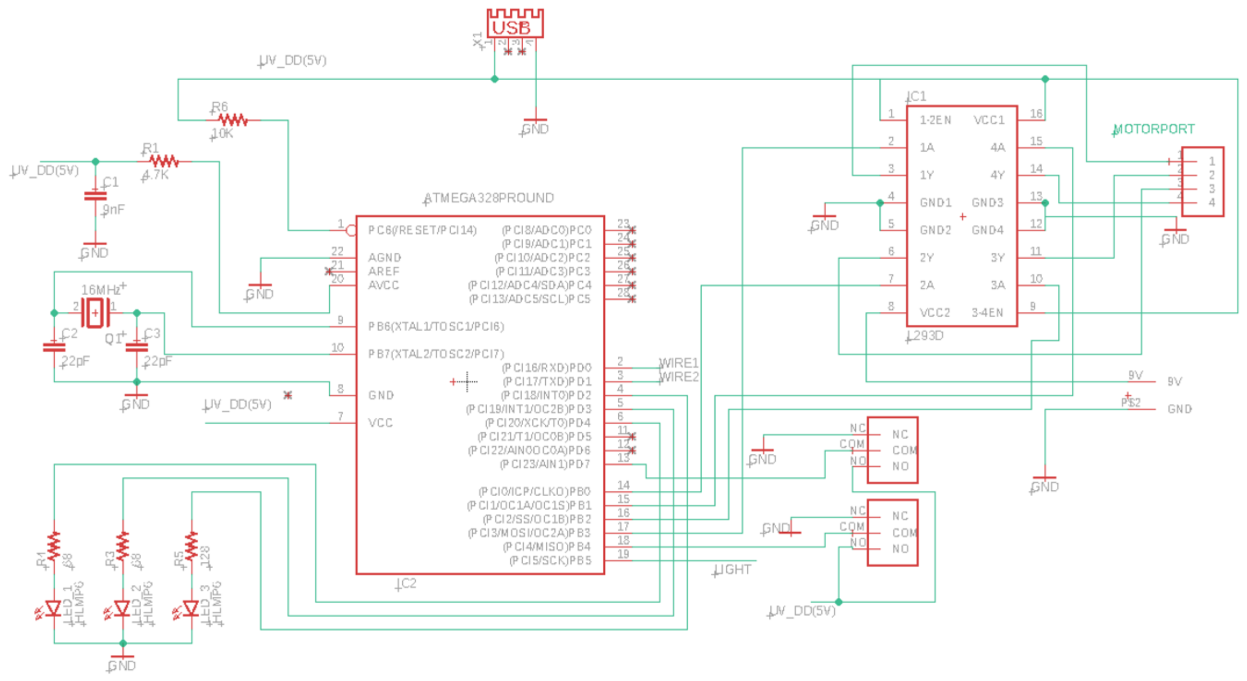

Figure 4: Final Schematic for the prototype. Using ATmega328

The problem with ATmega328 is that it does not have USB connection so it is only useful as a prototype but not for the final product. In Figure 4 the pins PD0 and PD1 are connected to two different wires that are going to help us to demonstrate how the signal from the host computer to the microcontroller will work if we had the USB connection. When the signal is 00 it means idle state, if it is 01 it means that we are playing a real audio and if it is 10 it means that we are playing a sham audio. The LEDs in the control module indicate in which state we are: yellow LED in idle state, red LED when a sham is playing and green LED when a real audio is playing.

In the following sections we are going to explain each module of the final schematic more deeply. The only difference between the final schematic and the prototype schematic aside from the USB connection is the power module as is explained in the following section.

3.2 Power Module

This module was created for the final design because the ATmega16U2 needs an input voltage of 3.3V but with our prototype as we are using an ATmega328 the input is 5V so we do not need the the voltage regulator. The power must convert standard USB 5V into 3.3V through a voltage regulator while still allowing a 5V channel to connect to the control module to facilitate data transfer between the software and control module. To execute that operation, the external power is supplied with USB 2.0 (we will assume that the device is always connected to this power source during operation). With this design consideration in mind, we eliminate the need for a battery or power storage component.

Our power consumption is averaged at around 2.325W with an average of 250mA at 3.3V and 300mA at 5.0V. The 3.3V consumption is justified because it is consistent with a majority of our hardware, encompassing the peripheral, sensor, and control modules. The 5V consumption is chosen to supply power to facilitate the data transfer between the software and control module. These values adhere to the maximum load able to be drawn from a USB 2.0 port - 500mA at 5V or 2.5W.

The low dropout regulator supplies 3.3V for the corresponding components in which it is required from an input voltage of 5.0 ± 0.25V. The classic LM1117 must be able to handle an input voltage at the theoretical low for USB 2.0 (4.75V) as well as the maximum (5.25V) at peak current draw (500mA) [2].

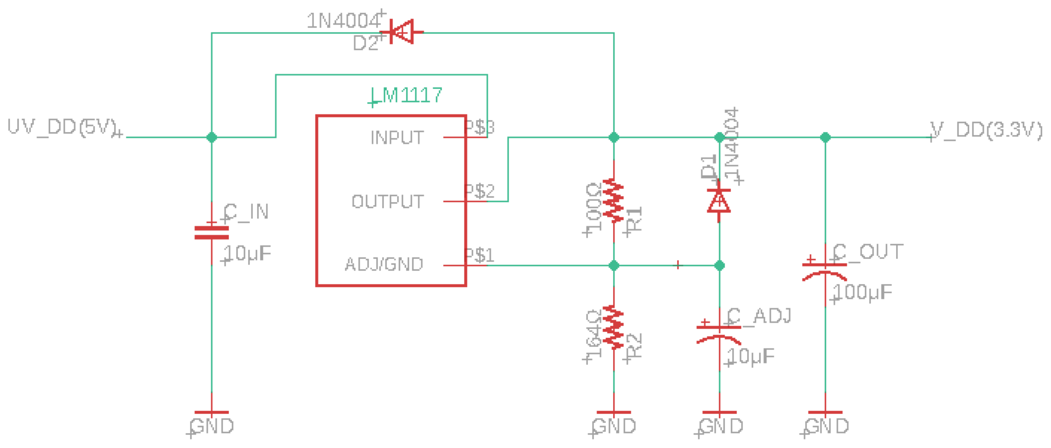

Figure 5: Circuit Schematic of the Power Module

Figure 5 outlines the external components needed to properly regulate this module to work in accordance with the rest. C_{in} and C_{out} are set according to the datasheet requirements for the LM1117 chip that we choose to utilize. C_{ADJ} helps improved ripple rejection when set to its state capacitance. D_1 helps prevent breakdown of our device if V_{DD} is shorted due to capacitative charge while D_2 does the same in case if UV_{DD} does the same.

3.3 Sensor Module

The sensor module primarily consists of the Trial Start button and the Trial Attempt button. The functions for these two buttons are briefed below.

The Trial Start button is a green lever for the bird to peck to indicate interest in beginning the trial. During this stage, there would be nothing happening in terms of food dispensing or punishment. When the green lever is pressed, the trial will officially begin and re-pressing the green lever will have no effect on future processes of the trial.

The Trial Attempt button is a red lever that takes in the birds input during the various states. If the bird presses the red lever during a Sham state, the system will punish the bird. And if the bird presses the lever during a reward state, the system will reward the bird. Pressing the lever during the idle state will not result in anything.

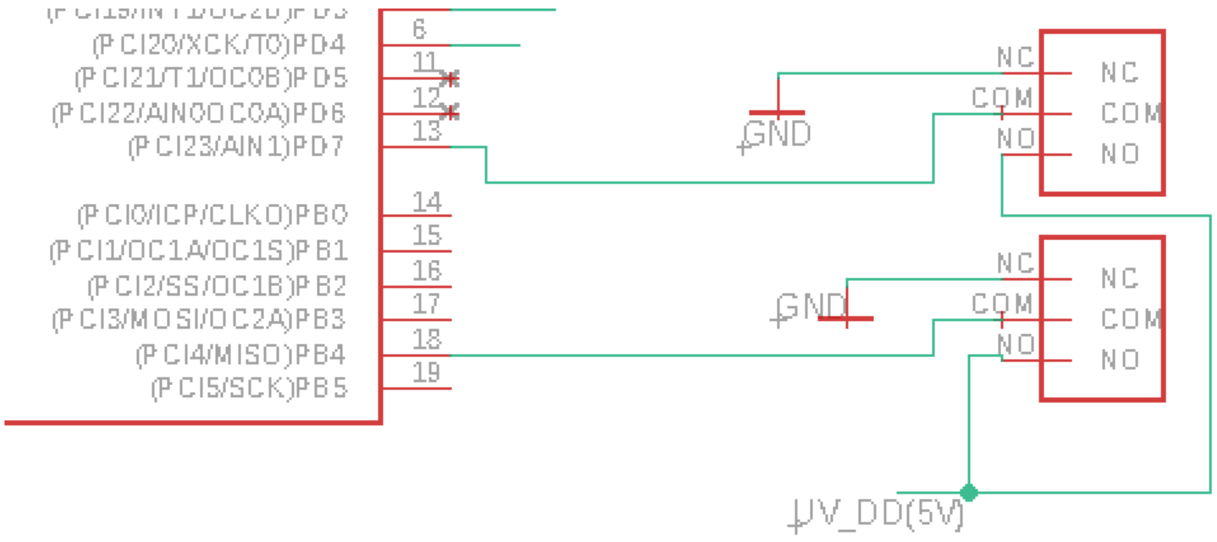

Figure 6: Trial Start and Trial Attempt Buttons

3.4 Peripheral Module

The peripheral module consists of a stepper motor driven by a H-Bridge. The overall design of the food dispenser additionally consists of an off-the-shelf Plastic PetSmart Bird Feeder, a cylindrical gear, and a funnel-cut piece of Teflon.

The logic of the food dispenser design is as follows: The cylindrical gear would attach to the stepper motor in a method similar to how an individual would put on a glove. Then, in between the cogs of the gear would be just enough allotted space for one seed. Thus, each full rotation of the stepper motor in theory would dispense exactly one seed. Next, the Teflon funnel would slowly funnel exactly one seed down upon spinning of the gear (because previous seeds would already be in the rotation).

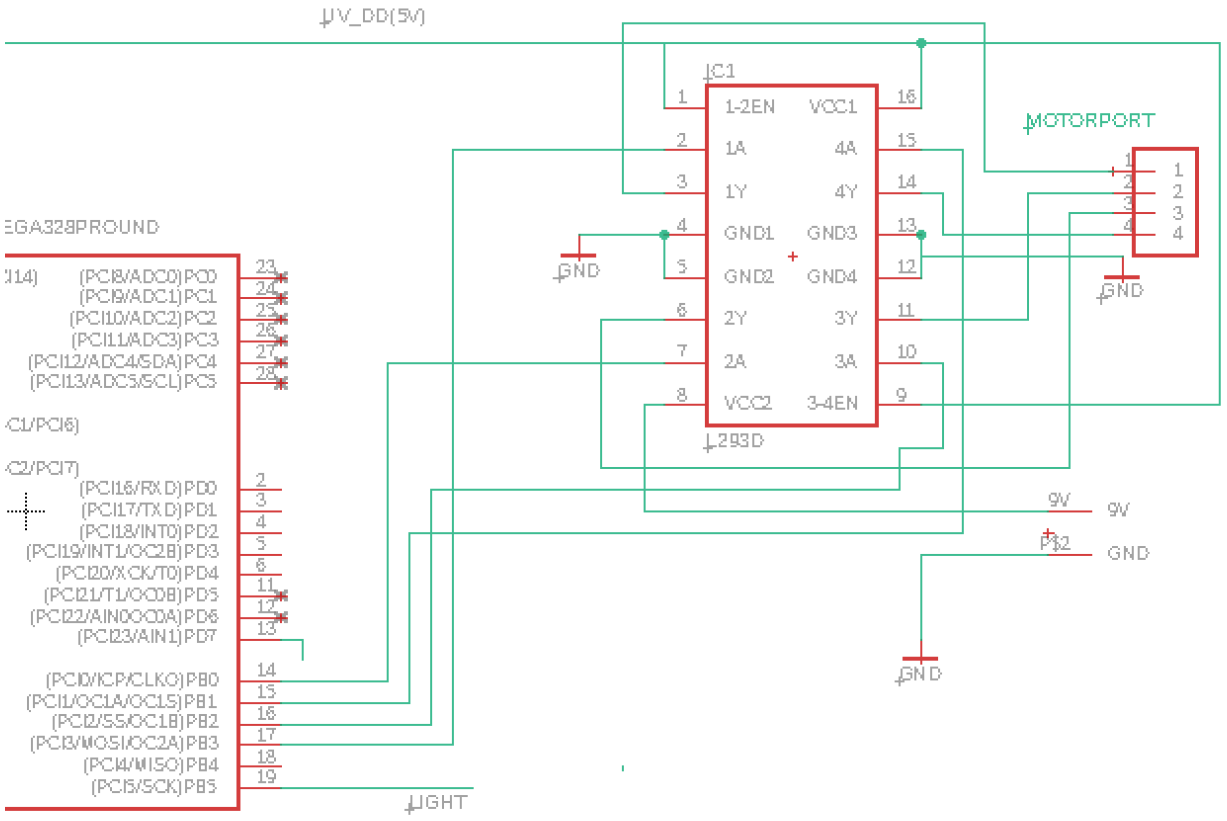

Figure 7: Motor for the food dispenser with H-Bridge and 9V battery and Light connection

4 Verification

4.1 Power Module verification

In Figure 8 it is shown how we verified that our LM1117 works properly.For that the LM1117 should provide a $3.3 \pm 5\%$ output from a $5.0 \pm 5\%$ input source. We used a voltmeter and measure the voltage at the input and the output of the device as we established in the our requirements and verifications in Table 3.

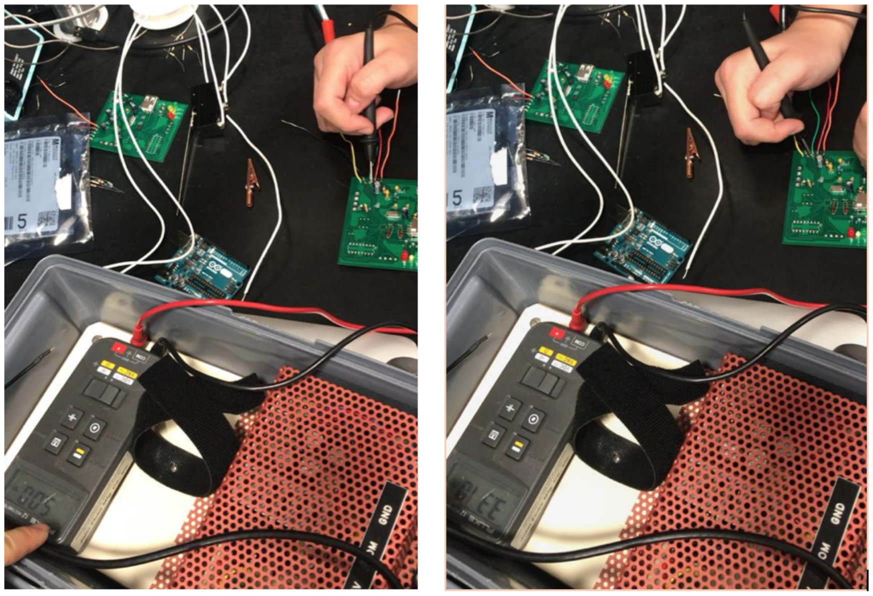

Figure 8: Verification of the input voltage of 5V(left) and output voltage of 3.3V(right) in the power module

In Figure 8 we can see how the input voltage is 5.00V when we give 5V in other part of the schematic with the same voltage. We can also see how the output voltage is of 3.310V. This means an error of 0.303% which is lower than the 5% of our requirements.

5 Software Design

This section will follow suit and explain the design of the software components. The importance of the module is in the fact that most of the processing and presentation is housed within this subsystem. Thus, it is necessary to create a fully functioning and fleshed out system while also being presentable to the user.

As of now, most of the back-end components of our application are completed and are mostly subjected to minor revision and commenting. The application is coded entirely in Python, using various modules to create a functioning portable executable file for Windows 10. A object oriented approach was taken in regards to the entire component. Naturally, the descriptions and documentation for the custom classes will be provided in the appropriate sections

For the interface component of the program, we use a python package that is able to suit the needs of our application. This is currently the main focus of development until the developer for the microprocessor is able to specify how specific information is encoded. The package is also heavily object-oriented with a heavy emphasis on abstract elements and classes and thus, requires the developed to create custom classes suited to the needs specified. The user interface is a simple drop-down file selection with custom dialog boxes akin to those present in older versions of Microsoft Office. Additional details on the graphical interface along with consistency requirements are detailed further on.

Lastly, the connection between the software and the hardware components is specified to be a USB 2.0 B-port connection. Originally, this was intended to be the sole connection between the operating machine and the hardware, however additional requirements from the sponsor have changed the requirements slightly and thus a standalone USB 2.0 connection would not meet our power consumption needs. Thus, additional connecting elements were asked to be connected to the host machine. Unfortunately, specifics behind the communication process are still quite hazy, however, they will still be outlined in the respective section.

5.1 Back-end Structure

The majority of the back-end coding and architecture is in place and is mostly subject to minor revisions and commenting or simple additions. The functional components of this portion of the subsystem revolve mostly around an object that contains a list of another object used to represent an individual trial.

5.1.1 Trial Class

The most fundamental and basic unit of the testing procedure is of course the individual trial.

This object is mostly an information dump for the testing procedure. A variety of parameters are fed during the generation process that is further detailed in the following section and stored as parameters to avoid directly interfacing with the variables and to enforce constraints on the various inputted values. A few of the parameters are specified to read-only such as the name of the researcher who performed the trial and the date said event was undergone to avoid modification of data and fabrication

Aside from the necessary initialization, getters and setter functions, the trial object only has one main method to run itself. The running process consists of effectively six stages detailed below.

- A continuous loop of the background sound until a trial is requested
- The delay period outlined in the trial generation until the target sound can be played
- The first instance of the target sound and the corresponding response window outlined as a parameter
- A buffer background sound between the first and second target sounds
- The second instance of the target sound and the final response window
- Return to the continuous background loop while recording results

This was the approach to programming the trialing procedure and is done so such that multiple trials can be done in relatively quick succession. Additionally, during the continuous looping described in the first and final steps, the user can specify a desired timeout duration that if exceeded, will terminate the trialling process while also alerting the user.

5.1.2 TrialSet Class

The trialset object contains architecture and functions to properly manage the trial objects described in the previous section.

This object should be the only method in which trials are generated as per our sponsor's specifications as trials are generated by block. The user specifies the number of blocks that they wish to generate or add as well as the block size if the task is initial generation or the previously specified block size. From this, each block is required to contain at least one instance of each specified target sound of which, the user can specify a number up to the block size for. The remaining spaces in the block, if any, are they randomly assigned a target sound.

The generation process also requires the probability for a sham trial as well as a minimum number of sham trials. If the entire process doesn't generate the necessary number of sham trials, then the entire procedure is repeated.

If the trialset meets the specified parameters and constraints, then the contained trials are randomly sorted such that the resulting target sounds do not follow a specified pattern outlined by the generation process. An option to reseed the probability distribution is then given to the user if they are dissatisfied with the generated set, and if so, the entire set is iterated and only the boolean denoting trial legitimacy is changed

5.1.3 Profile Class

Must like the trial class, this object is mostly a parameter dump for the running statistics for a testing subject. It contains mostly integer values regarding to trial completion and other such metrics as well as a comprehensive table of trials undertook by the subject. Naturally, the class contains a method to both load and save but does not allow direct modification to the parameters.

The parameters of this class are set to read only essentially to avoid potential tampering and are only modifiable by the main program's run function

5.2 Front-end Structure

This portion of the software is the current subject of much of the development time. Most of the components are currently either being actively developed, planned for development after more pressing components are integrated, or subjected to revisions and debugging. Nevertheless, a general outline of the components and relative progression is detailed as well as distribution methods and interface consistency guidelines.

5.2.1 Interface Components

We choose to use WxPython to power the interface. The program consists of a single ancestor frame that contains elements that allow the user to interact and modify back-end data via specified function by dialog or menu items.

The central frame contains a scrolling panel to display a loaded or generated trialset. It displays the information in a grid and allows the user to modify its contents via combo-boxes and check-boxes as well as modifier buttons to its right. Currently, this section of the main frame is mostly complete with only supplementary additions and revisions subject to addition such as right-clicking capabilities as opposed to the button menu. A few optimization revisions will most likely be performed should time allow it as the panel tends to hand should the user scroll through the contents too quickly for an extended time.

In addition to the information display, the main frame also contains a general statistics panel for the loaded trialset containing information such as number of probability reseeds and sham trials, separated according to the various target sound assigned upon generation. Likewise there is a corresponding panel for the statistics of the loaded profile, display total runs and other useful numbers.

Several dialog boxes are completed such as the ones required for generating new trials and running them. More will be developed as features are requested from the sponsor.

5.2.2 Distribution Method

We use the Pyinstaller module to distribute an executable file. The convenience behind this method is great as it's akin to deploying a normal executable using C/C++ code. The executable that the packager makes contains the necessary python packages needed to run the original code. Thus, the resulting file is self sufficient as well as portable.

5.2.3 Consistency

In order to preserve a presence of professionalism and order in our software, some guidelines were designated for future elements. For example, in the back-end components, consistency mainly revolves around the object oriented approach to development and the adherence to parameters instead of public attributes that can be directly modified. Likewise, in the front-end, we strive to meet the same end goal.

Before outlining the various guidelines we set for now, a few metrics were taken since they are rather unintuitive. Buttons in our interface package strangely have a horizontal and vertical buffer row and columns for some reason and thus must be centered (-1, -1) relative to the desired position. Additionally, the specified frame resolution includes the various menu items and default windows taskbar but fails to mention this properly in the documentatation. As such, the guidelines are as follows:

- The total pixel space of the taskbar at the top is 49px and directly subtracts from the vertical resolution of our window.
- The total pixel span of the various borders provided by windows totals to around 8px and directly subtracts from the horizontal resolution
- Dialog boxes should prioritize horizontal spacing over vertical spacing in the sense that dialog box frames should attempt to adhere to the relative resolution
- Sizers used for the default frame should prioritize horizontal layout over horizontal flexibility. Borders should be preserved as well
- Accelerators in the table should adhere to traditional bindings
- Dialog buttons should be of size (80, 26)px, check-boxes of (16, 16)px and square buttons (26, 26)px. Other UI elements must have appropriately sized features

6 Cost

The total cost of the project would be \$13,858.78 which is the sum of the Labor costs and Parts costs.

6.1 Cost of Labor

We assume that the labor costs of the design would be \$45 USD per partner and 10 hours a week. We will consider 63% of the semesters weeks into this calculation. Thus, labor costs for each partner in this project would be encompassed with the following calculation:

$$\$45/\text{Hour} \cdot 3 \text{ partners} \cdot 10 \text{ hours/week} \cdot 0.63 \cdot 16 \text{ weeks} = \$13,608$$

6.2 Cost of Parts and Materials

Below is a table of the cost of our components

Table 1: Cost of the materials

Part	Cost	Cumulative Total
Female USB Input	\$5.79	\$5.79
LM1117 Voltage Regulator	\$1.10	\$6.89
ATMEGA16U2A2 Microprocessor	\$2.52	\$9.41
OrionFan OD4020 Series	\$4.74	\$14.15
Vifa Compact HiFi Bluetooth Speaker	\$229.00	\$243.15
Passive Components	\$7.63	\$250.78

7 Conclusions

7.1 Accomplishments

The final project displayed incredible accuracy with the peripheral module. Having seed-by-seed accuracy helps increase the flexibility of food dispensed for any subject (granted that different species of birds consume different portions of food).

The software of this project followed a user-oriented design and improved on week-by-week iterations of collaborations between the team and the sponsor, building out various additional functionalities requested by the sponsor, such as trial reseeding and bird profiles.

7.2 Uncertainties

Moving forward with the project, because the microcontroller choice was ultimately abandoned due to time, it was undetermined whether or not the microcontroller we chose would be able to communicate via USB to the final software interface.

7.3 Ethical considerations

Following the ACM code of ethics section 2.3: "Know and respect existing rules pertaining to professional work", we need to learn existing guidelines and construct our project around and according to them [12]. Before any work with animals, it is mandatory to submit IACUC protocols and they adhere to nationwide rules for animal care and research. Our project fulfils all these requirements and has been approved already [11].

The birds are rewarded with specific amount of food for successful trials, and, we shut off the cage light in an ethical way to not harm the subject either physically or mentally. The light shut off is transient and only serves as an indicator that the subject has done something wrong, and as such, does not frighten or stress the subject. The timeout function ensures that the subject does not necessarily have to undergo testing if it does wish to do so in accordance with ACM code of ethics section 1.2: "Avoid harm", we are treating the subjects in such a manner that we avoid any potential harm that may befall them otherwise [12].

In the design of our project, we will follow the guidelines set by the IEEE code of ethics section 6: To maintain and improve our technical competence and to undertake technological tasks for others only if qualified by training or experience, or after full disclosure of pertinent limitations [13]. So we will follow every guideline that the doctorate responsible of this project tell us to be sure that we give the right amount of food to the bird and how much time the bird can be with the light off. Following this code, we need to make sure that we have an user friendly interface that everyone without programming knowledge can understand. We need to make sure that our project works seamlessly and without error because the integrity of our system will also reflect the reputation of those who use it, following the IEEE code of ethics section 9.

Our design must account for certain ethical risks. All the wires must be outside the cage and hidden such that the bird will not be subject to electrical damage. If any wires are required within the cage, they must be concealed by opaque material to guarantee risk aversion. All hardware that the bird interacts with must be attached in a manner in which is irremovable by the bird. A proper speaker frequency range is paramount to ethical hardware design with the bird. Given that bird songs typically range from 1,000Hz to 8,000Hz, our design will adhere to that range to avoid any stress on the bird [15].

7.4 Future work

USB Communication

Our software will have to communicate with the microprocessor we selected via USB. As such, we must construct a module that will handle this method. Other existing methods may also work such as Python USB packages, but need to be testing when the control module becomes constructed. That aside, threads will be used to communicate between the main program and the data poll.

Training Program

Our sponsor has previously hinted that a training module for the birds would be appreciated. However, this is currently treated as a luxury that can be developed after core features have been fully fleshed

Logging Method

A thread devoted to logging program progression will be created at a future date. It is not an immediate concern as we have easy debugging tools as a developer, however, before delivering the product, a proper logging thread will be constructed.

References

[1] R. O. Davis *Computer Methods and Programs in Biomedicine. Digital signal processing in studies of animal acoustical communication, including human speech.* Volume 23, Issue 3, December 1986, Pages 171-196

[2] Texas Instruments, *LM1117 800 mA Low-Dropout Linear Regulator,* LM1117 datasheet, February 2000. [Revised January 2016].

[3] Microchip Technologies, *8 bit AVR Microcontroller with 32K bytes In-System Programmable Flash,* ATMega328P. Datasheet, 2009.

[4] Microchip Technologies, *8 bit AVR Microcontroller with 16K bytes of ISP Flash and USB Controller,* ATMega16U2 Datasheet, 2010.

[5] Orion Fans, *OD4020 Series,* OD4020 Datasheet, Publishing Date Unknown.

[6] eswitch, *LP11 Series Pushbutton Switch,* LP11EE1NCSRGB Datasheet. September 5, 2018.

[7] eswitch, *LP11 Series Pushbutton Switch,* LP11EE1NASRGB Datasheet. September 5, 2018.

[8] Broadcom, *4mm Oval Precision Optical Performance LED,* HMLPLG71XD0DD Datasheet. March 30, 2012.

[9] VCC *4303F Series Solid State LED T1 (3mm),* 4303F5 Datasheet. Publishing Date Unknown.

[10] Sparkfun, *4UCON Technology Inc,* Audio 3.5mm Datasheet. August 2004. [Revised in May 2005].

[11] *Illinois Institutional Animal Care and Use Committee (IACUC). Regulatory Compliance & Safety*, 2016. Available at: http://research.illinois.edu/regulatory-compliance-safety/research-integrity-and-ethics

[12] *ACM.org ACM Code of Ethics and Professional Conduct*, 1992. [Online]. Available at: https://www.acm.org/code-of-ethics

[13] IEEE.org *IEEE Code of Ethics*, 2016. [Online]. Available at https://www.ieee.org/about/corporate/governance/p7-8.html. (Accessed February 4, 2019).

[14] Ritchison, G. (n.d.). Retrieved February 20, 2019, from http://people.eku.edu/ritchisong/birdbrain2.html

[15] *All About Birds*, Do Bird Songs Have Frequencies Higher Than Humans Can Hear? https://www.allaboutbirds.org/do-bird-songs-have-frequencies-higher-than-humans-can-hear/

APPENDIX A. Requirements and Verifications Table

Table 2: System Requirements and Verifications

Requirements	Verifications	Verification status(Y or N)
- Control Module. Shutoff at nonstandard voltages and currents	- Use a voltage generator and set voltage out of bounds (> 5.25V) and ensure nono power is supplied by the microcontroller to other components - Use a current generator and set current out of bounds (> 500mA) and ensure no power is supplied by the microcontroller	N
- Control Module.Can accurately transfer 32 bits of data via USB within 10ms	- Connect the microcontroller with a host computer and open a terminal on said host - Send a 32 bit character from computer - Using the computer, Request a 32 bit character from the microcontroller Check the data's accuracy as well as the time stamps	N
- Control Module. All LEDs must be visible from 3m away	- Drive each LED circuit with an equivalent 3.3V source - Stand a distance of 3m form the LED and ensure it is visible	Y

Table 3: System Requirements and Verifications. Continuation

Requirements	Verifications	Verification status(Y or N)
- Power Module. Outputs a maximum of 500mA at 5 ± 0.25V at all times when connected to an external device	- Attach equivalent load to when device is at peak power consumption to USB port Measure current using an ammeter is series - Measure open circuit voltage	Y
- Power Module. Provides a $3.3 \pm 5\%$ output from a $5.0 \pm 5\%$ input source	- Use a voltmeter and measure the voltage at the input and the output of the device	Y
- Power Module. Operates with a current draw between 0 - 300mA	- Using an oscilloscope as a resistive sweep, measure the current with a varying resistance until it reaches the lowest that the device does naturally	Y
- Power Module. Maintains a temperature below 80C during trialing	- Use an IR thermometer to ensure device temperature is below upper bound after a standard testing session (About an Hour).	Y
- Sensor Module. Buttons distinguishable between them	- Place in separate environment with the Trial Attempt Start - Reward subject if they can distinguish between the two buttons	Y
- Sensor Module. Sized appropriately such that the test subject can identify and press	- Place button alone with subject in separate environment and reward if button is successfully pressed	Y
- Food dispenser. Delivers expected food mass set within 5% error upon trial success	- Simulate a successful trial, weight the output of the dispenser	Y
- Food dispenser. Test subject must not be able to tamper with the dispenser and obtain food at any time	- Reward subject in separate environment and note through camera is subject is able to tamper with the dispenser to obtain additional reward.	Y

Table 4: System Requirements and Verifications. Continuation 2

Requirements	Verifications	Verification status(Y or N)
- Food dispenser. Delivers absolutely no reward under any other outcome other than a success	- Simulate a failure and both inaction clauses and note if any food is dispensed	Y
- Cage Speaker. Accurately reproduce hi-fidelity audio with less than 10% difference between given and played spectrograms when noise is filtered	- Create spectrogram of audio sent through 3.5mm line out - Record playback audio from speakers and compare to previous spectrogram and ensure that it is accurate enough	—
- Cage Speaker. Must have a standby noise of less than 10dB	- Set speakers to desired volume used in trials - Measure the standby playback of the speakers. Ensure standby is less than 10dB after filtering noise and accounting for recording errors	—
- Cage Speaker. Speakers must have a relatively uniform amplification in the frequency band of 100Hz to 20kHz	- Using a function generator, sweep the input signal from 100Hz to 20kHz and record the output waveform from the speaker - Compare the resulting Fourier Transforms of the recorded signals and ensure that the spectral peaks all have relatively the same height after accounting for recording microphone bias.	—
- Cage Light. Brightness of the light must not affect the subjects ability to correctly perform the test	- Progressively increase resistance of series resistor stopping when subject first displays signs of discomfort or limit imposed by sponsor as to not harm subject	Y
- Audible from a 10m distance	- Play a preset audio file and ensure that it is audible from a distance of 10m	Y