VoxBox Robo-Drummer

Team 27: Craig Bost, Nicholas Dulin, Drake Proffitt
ECE 445 Final Report Spring 2018
TA: Zhen Qin

Abstract

This report provides an overview of the design, implementation, and performance validation of a device which translates human vocal “beatbox” performances into an analogous machine performance by means of sound recognition and electromechanical actuation in real-time. This device is intended to be operated in a manner similar to a conventional musical instrument, whereby users may “tune” the device to their voice in order to optimize the device’s performance. Users may tune and operate 3 distinct drums using this system.

Table of Contents

1. Introduction

1.1 Objective

Our design enables a human user to convert live "beatboxing" performance into real-time physical performance by a robot. As such, our design is best thought of as musical instrument in its own right: when "played" as intended, our design responds by instructing a robot drummer to strike the corresponding drum kit component. Correctly "playing" our system amounts to the human user constraining their vocal performance to a subset of sounds which can be identified as comprising conventional "beatboxing". This mapping of performative sounds to the correct corresponding robotic performance comprises the core challenge that our design achieved successfully.

Our design is distinct from existing designs which similarly process "beatboxing" into equivalent musical forms, as these existing systems typically only convert recordings of "beatbox" performance into drum samples for music production; prime examples include "The Beatbox Machine" [1] and "Vocal Beater" [2]. As such, these existing systems are purely software-based and non-real-time, whereas our design features real-time performance, driven by hardware-software embedded systems.

For the purpose of establishing a standard for "real-time," our design used as reference the International Telecommunication Union's recommendation for maximum acceptable lag time between image and sound of 185 milliseconds [3]. We chose this standard because our design has a visual aspect to it, such that we recognized the necessity of avoiding noticeable delay between the robot actuation from the moment of user vocal input, in order to maintain the subjective impression of "real-time" behavior. The ITU recommendation thereby provided us with a de facto objective measurement to quantitatively ascertain that aspect of our design's success.

1.2 High Level Requirements

To sufficiently meet the objective of project, our design must satisfy the following high level requirements:

1. Device must be able to receive beatbox audio input from a human user/performer and distinguish between 3 different key sounds. This implies that there are 4 valid inputs in total: clap, snare, bass kick, and no input.

2. Device must be able to perform requirement (1) in real-time (defined as above in the Objective as within 185 ms) and drive a machine based on said inputs.

3. The machine will have 3 drums, one for each input, and will strike the correct drum which corresponds to its input, with a measurable sound-pressure intensity above 40 dB.

1.3 Block Diagram

Figure A.1 provides a graphical depiction of our complete system from a high-level perspective, detailing the functional constituent components of each sub-block and the nature of their connections.

Starting with the Input Subsystem: its role is to convert sound waves into analog voltage signals. This starts with the mic block, which is the input transducer for electromechanical conversion of sound waves to voltages. The Amp block contains amplification circuitry designed to boost and offset the mic-level signal into a more suitable range for sampling. The LPF (low pass filter) block provides low-pass filtering for improving signal quality, which is passed to the Control Subsystem.

The Control Subsystem contains the DSP block, whose principal tasks involve the analog-to-digital conversion of the Input Subsystem's output signal, processing of the digitized input, and subsequent response to qualifying stimulus by providing the trigger signals to the Electromechanical Conversion Subsystem. These output signals pass through Circuit Protection B, which protects the DSP block from any possible back EMF from the solenoid drives. A voltage limiter is placed between the DSP block and the LPF block to ensure that input signal stays within the safe limits for the ADC input pins. Adjustment Potentiometer and Tuning Button blocks provide the user interface to the the DSP block, along with the LED block and its indicator lights.

The Electromechanical Subsystem converts the trigger pulses from the Control Subsystem into solenoid driver outputs, by means of the Square-Wave Amplifier and Solenoid & Driver (x3) blocks. The Circuit Protect A and Circuit Protect B blocks provide protection from over-voltage and over-current, respectively.

The Mechanical Output Subsystem is essentially the output transducer mechanism: the Armatures block translates the mechanical motion of the solenoids into a drum strike, and the drums translate the kinetic energy of the drum strike into sound waves through mechanical vibration.

Lastly, the Power Subsystem supplies power to all other subsystems. Linear Regulator blocks provide the necessary conditioning of the Li-Ion power supply voltage to voltage levels appropriate to their respective subsystem demands.

2. Design

2.1 Power Module Design

The entire system contains several circuits that require a DC voltage supply as well as solenoids that require large instantaneous current draws. For these reasons as well as the desire to have a mobile device, we chose to use a battery as the power source. Two linear regulators (Figure A.2 and A.3) supply power to the input and control modules. An outline of the components used are shown in Table A.21.

2.2 Input Module Design

The input module is comprised of a microphone (Behringer Ultravoice XM8500 Dynamic Coil Microphone), microphone pre-amplifier, and a low-pass filter.. The purpose of this module is to convert human voice input into an electrical signal that can be processed by the microcontroller's ADC. The amplifier is a necessary component because the output from the microphone is too weak for the microcontroller to adequately process the signal. The filter is a necessary component because it prevents aliasing.

2.2.1 Amplifier Design

The microphone pre-amplifier takes the microphone output (mic level) signal as input, and amplifies it to a suitable level (line level) for ADC sampling. The requirements for the amplifier are shown in Table 2.1 below. The voltage gain was empirically derived from the test data for the microphone output voltage amplitude, and the microcontroller ADC analog reads. The bandwidth requirement covers the range of beat box from human voice. Constant group delay is desired because a linear phase system is distortionless.

Table 2.1: Amplifier Requirements

Requirement	Value
Voltage gain	80 V/V
Bandwidth	100Hz -10kHz
Group delay	Constant

The amplifier schematic is shown in Figure A.4. This is a 2-stage inverting op-amp circuit where each stage has identical gain. The op-amps are LM741. Two stages were chosen as opposed to a single stage due to the insufficient gain-bandwidth product rating of the LM741. A multi-stage design allows each stage to have a smaller gain while maintaining the net gain of the system. As shown in Figure A.4, there are three 10k resistors in shunt at the input, between the stages, and at the output. These resistors are necessary to eliminate the travelling DC offset that occurs when the circuit is first powered on.

The voltage gain for a single inverting op-amp stage is given by [4] :

$$\frac{V_{out}}{V_{in}} = -\frac{R_{feedback}}{R_{in}} = \frac{9k}{1k} = 9\ V/V. \quad (2.1)$$

Then the net gain of the amplifier is 81 V/V which is calculated by multiplying the gain of each individual stage.

2.2.2 Filter Design

The anti-aliasing filter receives input from the microphone-amplifier chain, and outputs the filtered signal to the Teensy ADC for sampling. The filter was designed using Texas Instrument's FilterPro software [5]. The active analog filter design settled on was a Sallen-Key style, 5-stage, Butterworth filter with -3dB cutoff frequency at 23kHz. The original circuit schematic presented in the original Design Document is in Appendix (Figure A.5, Table A.6).

2.2.2.1 Filter Design Considerations

Filter requirements were dictated by the results of our spectrogram analysis of the human voice, which revealed that this class of audio signals are principally bandlimited to 10kHz. Given the expected sampling rate of our microcontroller at 44.1 kHz, noise suppression in the frequency regime beyond the upper range of human voice was warranted. As such, we designed an analog filter to achieve this performance with maximally flat gain in the passband of 0 to 10 kHz, with linear phase characteristics, to avoid unwanted distortion of the desired frequency content of the input signal.

We used the Butterworth filter design topology to achieve the desired performance. The principal downside that our filter design needed to account for is that the Butterworth filter experiences non-linear phase near its cutoff frequency (-3 dB attenuation). If the cutoff frequency is set too low phase distortion would affect the upper frequencies of our passband; as such extra care was needed to determine the appropriate cutoff frequency.

One workaround which we employed in our filter design was to "drag-out" the passband far enough for the phase distortion to clear the upper frequencies of interest, but without introducing aliasing. At a sampling frequency of F_s = 44.1 kHz, the max resolvable frequency is $f_{max} < ½\ F_s$ = 22.05 kHz; any frequency below this value will be sampled correctly, without aliasing. However, as our signal is inherently bandlimited, aliasing within the range of its upper limit of 10 kHz and the max allowable of 22.05 kHz is permissible.

We can exploit this as follows: knowing that ½ F_s = 22.05 kHz corresponds to the discrete frequency $\omega = \pi$, and spectral copies occur every 2π, we can solve for the permissible bandwidth that prevents aliasing in the 0 to 10 kHz range.

$$F_s\ \omega = \Omega \quad (2.2)$$

where ω is the discrete radial frequency, Ω is the continuous radial frequency, and F_s is the sampling frequency. Using F_s = 44 kHz and Ω = 2π × 10 kHz:

$$\frac{\mathbf{10}\ kHz \times \mathbf{2}\pi}{F_S} = \omega = \frac{\mathbf{5}}{\mathbf{11}}\pi \tag{2.3}$$

Knowing the spectral copy centered at *2π* can jut into the spectral copy centered at zero all the way to,

$$2\pi - \frac{5}{11}\pi = \frac{17}{11}\pi = \omega_{bandwidth} \tag{2.4}$$

Converting back to continuous frequency yields

$$\frac{(\omega_{bandwidth} \times F_s)}{2\pi} = 34\ kHz = f_{bandwidth} \tag{2.5}$$

Thus, our max permissible bandwidth is 34kHz.

Knowing this we designed the filter to have -3 dB cutoff of 23 kHz, this resulted in a -30 dB attenuation at the 33 kHz mark- a point where we defined the signal to be "sufficiently bandlimited". The 33 kHz band limit will result in aliasing in all frequencies above 11 kHz when sampled, but all frequencies below that will be preserved. The -3 dB cutoff at 23kHz also meant that phase was for the most part linear throughout the 0 to 10 kHz. Thus, all design goals were achieved: maximally flat (property of the Butterworth filter), linear phase, and 0 to 10 kHz preservation.

2.3 Control Module Design

2.3.1 Hardware

As the DSP aspect of the design is crucial to the underlying functionality of the project, we needed hardware capable of: 1) performing high-precision digitization of analog signals; 2) sampling at a rate sufficient to avoid high-frequency aliasing; and 3) producing frequency-domain data (i.e. FFT's) with minimal lag.

To meet these specific requirements, we selected the Teensy 3.6 hobbyist development platform to fulfill this role as our DSP-capable microcontroller. The Teensy's capacity to sample at 44.1 kHz [6] up to 13-bit precision [7] satisfies the first two requirements, while its main clock speed of 180 MHz [8] and library of FFT functions [9] satisfied the final requirement. In addition to these features, we chose the Teensy 3.6 because it can be programmed in C/C++ using a straightforward IDE, allowing for rapid software development. As a final design consideration, the Teensy can be mounted to a PCB via header-pins, allowing quick replacement in case of accidental blow-out; a potentially significant time-saver.

In implementation, the microcontroller has two analog inputs, four digital inputs, one analog output, and seven digital outputs, as shown in schematic Figure A.10.

One analog input is the audio voltage signal from the Input Subsystem. The other analog input is the voltage level coming from an Adjustment Potentiometer that the user uses to manually adjust the sensitivity of the system to audio input; this voltage level is supplied by the one analog output, which is feeding a 50% duty-cycle PWM signal to the potentiometer, from which its voltage level is sampled and mapped to a threshold level in software.

The four digital inputs consist of momentary push buttons which the user uses to send request commands to the system; namely, requests for which drum they wish to "tune." The fourth button is used only for debugging purposes, and is not used by the user.

Of the seven digital outputs, three correspond to trigger command signals to the Electromechanical Conversion Subsystem, as these constitute the stimulus for the corresponding drum to strike. These trigger commands are also connected to indicator LEDs, which provide a redundant visual cue as to which drum was identified and triggered. The remaining four digital outputs are to additional indicator LEDs, which provide information to the user as to which "take" of the tuning process they are on, as well as peak-indicator light, indicating whether their sound input was loud enough to be detected by the system.

2.3.2 Software

The software running the system's control flow has two primary modes of operation: a "Tuning Mode" and a "Performance Mode." As shown in Figure A.11, the system program will not enter into the latter mode until the former has been completed: once the user has provided training data to the system, in the form of 10 "takes," (i.e. examples of their vocal performance), per drum they are "tuning." Once in Performance Mode any vocal input will be analyzed by the system program to determine which of the 3 drum sound profiles constructed in Tuning Mode the new input event best matches, and trigger the best matching drum accordingly.

Various methods of sound identification were researched including Principal Component Analysis [10] and Mel-Frequency Cepstral Coefficients [11]. Due to impending deadlines, computational complexity and the need for real-time implementation, we decided to forego these methods in favor of an algorithm of our own design. From spectrogram analysis of the sound set of interest, we determined that the sounds are "visually" discernible (Figure 2.1 below). This combined with the fact that our problem set (3 sounds to identify) is relatively small makes this implementation feasible.

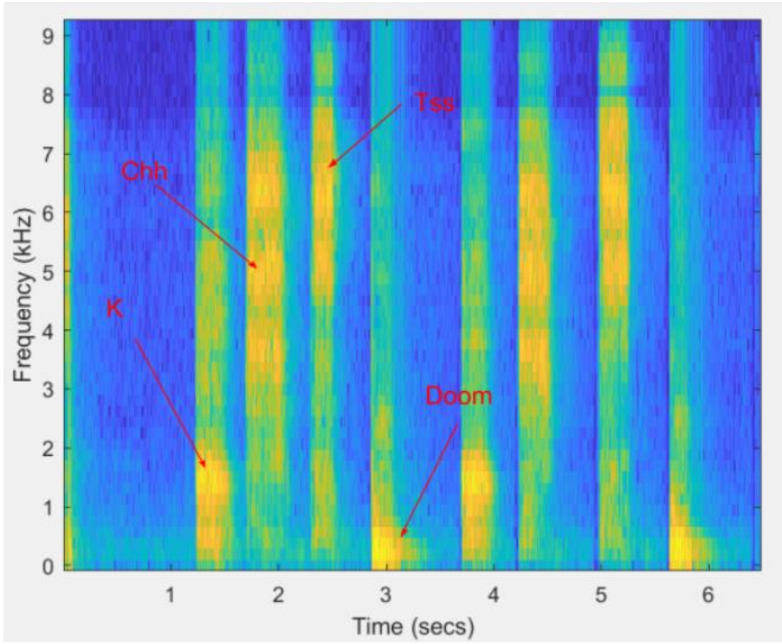


Figure 2.1 : Spectrogram of Beatbox sounds

For tuning, our algorithm requires the user utter each beatbox sound 10 times to develop a base for comparison. In these tuning sessions data is collected on sound "start," as opposed to the entire sound. This is required for a real-time implementation, because a reaction *after* a full utterance is too late. For every 512 samples a 1024-point FFT is taken from which only the dB value is of interest ($20log(|FFT|)$). *From here on "FFT" will be used to refer this dB converted quantity.* The first five *voiced* FFT's of a sound correspond to a sound start. Mathematical analysis has revealed 3 features of interest within each FFT: 1) The frequency energy centroid; 2) the low frequency energy ratio; and 3) the top 3 peak locations.

The frequency energy centroid is functionally equivalent to a center of gravity calculation applied to an FFT (Figure A.7): it identifies the center of the energy distribution within the FFT frame, where the center in question corresponds to a frequency bin value:

$$Frequency\ Energy\ Centroid\ = \frac{\sum_{i=0}^{511} \ FFT[i]\times i}{\sum_{i=0}^{511} \ FFT[i]} \tag{2.6}$$

where i = frequency bin or index.

The low-frequency energy ratio (LFER) quantifies the ratio of energy occupying the frequency band 0 - 1 kHz, relative to the total unique energy of the FFT (frequency bins 0 through 512). The explicit formulation of this calculation is shown below:

$$LFER = \frac{\sum_{i=0}^{25} FFT[i]}{\sum_{i=0}^{511} FFT[i]} \tag{2.7}$$

where i = frequency bin or index.

To calculate the top 3 peak locations, the frequency spectrum is quantized into 1 kHz wide frequency bands, from which the three frequency bands with the greatest peak values are selected (Figure A.9). These three peaks are then encoded into a numeric codeword. A graphic example can be seen in Figure 2.2 below:

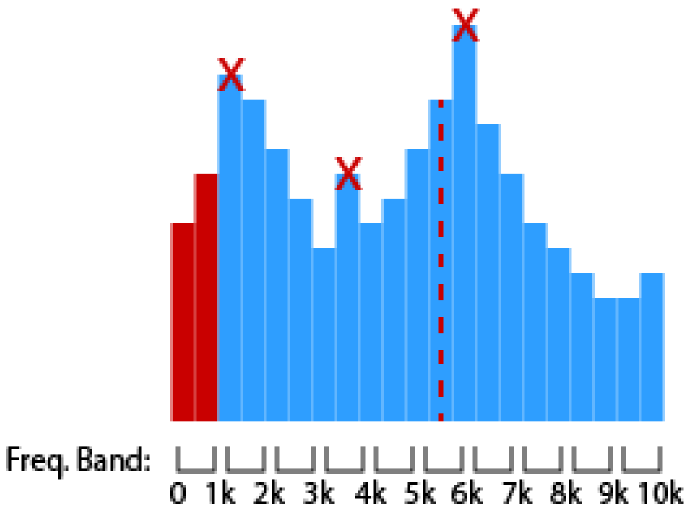


Figure 2.2: Example of of FFT frame with top 3 peak codeword of "246"

When the system program is running, for every sound event detected, the system program captures 2 to 5 FFT frames of data, which corresponds to the "attack" of the sound, and extracts the aforementioned features of interest, storing them into an event buffer.

In Tuning Mode, 10 of these event buffers are cached and processed to construct the sound profile for the drum being tuned. This profile consists of an average of the average centroid and low-frequency energy ratio, and a probability distribution matrix. To construct the distribution matrix, the system program counts all the instances of top 3 peak combination per frame position to first build a histogram, which is normalized into a probability. Thus, a probability distribution relative to each frame position is generated. Figure 2.3 provides a graphic representation of this process.

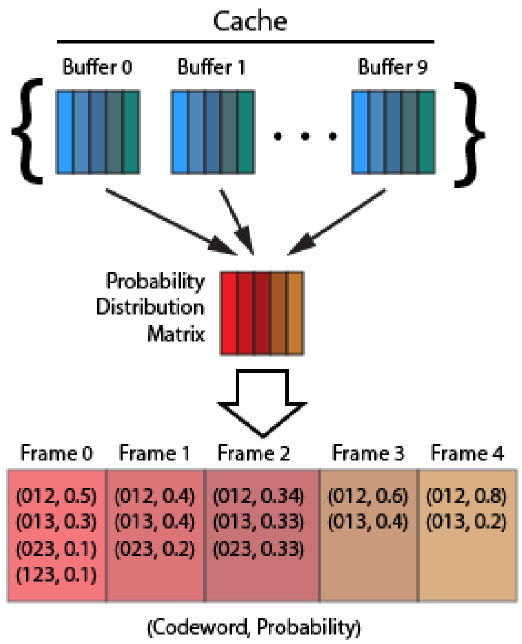


Figure 2.3: Probability Distribution Matrix Construction Process

In Performance Mode, the event buffer populated from the new user audio input signal is compared against the 3 drum sound profiles previously constructed in Tuning Mode, resulting in a 3 parameter score: 1) the difference error between average centroids; 2) the difference error between average low-frequency energy ratios; and 3) the aggregate probability derived from comparing frames between the event buffer and the sound profile. To compute the aggregate probability, the probability distribution matrix functions as a look-up table, whereby for a given frame top 3 peak combination an associated probability is returned; the results for all 5 frames are summed to produce an aggregate probability.

To choose which drum to trigger, a voter compares these 3 parameter scores, identifying which profile corresponds to the minimum centroid error, the minimum low-frequency energy ratio error, and the maximum aggregate probability for each. If the same profile is selected for both minimum error selections, the voter chooses that profile as the best match winner; if they do not agree, the system resorts to choosing the profile with the highest aggregate probability.

Once the winner is selected, the system program sends a trigger signal to the corresponding drum, in the form of a digital logic high signal.

2.4 Electromechanical Conversion (EMC) Module Design

The purpose of this module is to provide a mechanism by which to actuate the “arms” of the drummer. We chose to use solenoids to achieve this goal because they have good speed performance and are easily implemented. Speed is the most important design parameter for the EMC module. A high-level requirement is that the system operates in real-time so the EMC device is designed to be fast. Solenoids actuate faster than servos but why not use pneumatics? The answer is simple, ease of implementation. The solenoids are powered from a DC source and require only a driver circuit so that they do not damage any other parts of the electrical hardware.

2.4.1 Solenoid Design

Figure A.18 shows an image of one of the solenoids. Each solenoid is homemade and a components list is outlined in Table A.22. The solenoids are wrapped in 22 AWG magnet wire with 640 turns each. The solenoids are designed to meet the following two requirements: 1 inch throw and 100ms response time.

2.4.2 Solenoid Driver Circuit Design

The solenoid driver circuit (Figure A.19) is comprised of two parts, the square-wave amplifier and the current driver. The current driver consists of a power FET used to switch current and a flyback diode whose purpose is to provide inductor over-voltage prevention. A sufficient gate voltage must be supplied to the power FET so that its drain-source resistance is minimized and maximum current can flow through the solenoid. This is where the square-wave amplifier is utilized. The purpose of this circuit is to amplify the 0-3.3V digital signal from the microcontroller to a 0-11.1V signal. This circuit also limits the current draw from the microcontroller pins to protect the microcontroller from damage.

2.5 Mechanical Output Module Design

The purpose of this module (Figure A.20) is to create three distinct drum sounds when triggered by the microcontroller. Precision was not taken into consideration for this design. The stage and drum ensemble is made of the following materials: wood, PVC (used for the drum sticks), and various containers and bottles (used for the drums). In addition to providing the drummer, this module also houses the project circuitry.

3. Verification

3.1 Power Module Verification

The power supply module is designed to supply 8V and 5V from two linear regulators as well as supply 8A to each solenoid. These requirements have been achieved. The linear regulators have been verified by measuring the output voltage from each one while the entire system is powered on and operating. The solenoid power requirement has been verified by measuring the current draw during an actuation. The battery voltage does not drop while it supplies 8A to a solenoid.

3.2 Input Module Verification

3.2.1 Amplifier Verification

The amplifier requirements are outlined as follows: 80 V/V gain, 100Hz-10kHz bandwidth, and linear-phase. To verify these parameters we used the following procedure: bias the amplifier with a DC power supply, send a single tone input into the amplifier by means of a signal generator, measure the output with an oscilloscope, record the gain and group delay, repeat for frequencies between 1 kHz and 10 kHz. A plot of the amplifier verification measurements is shown in Figure 3.1. The amplifier achieved all requirements.

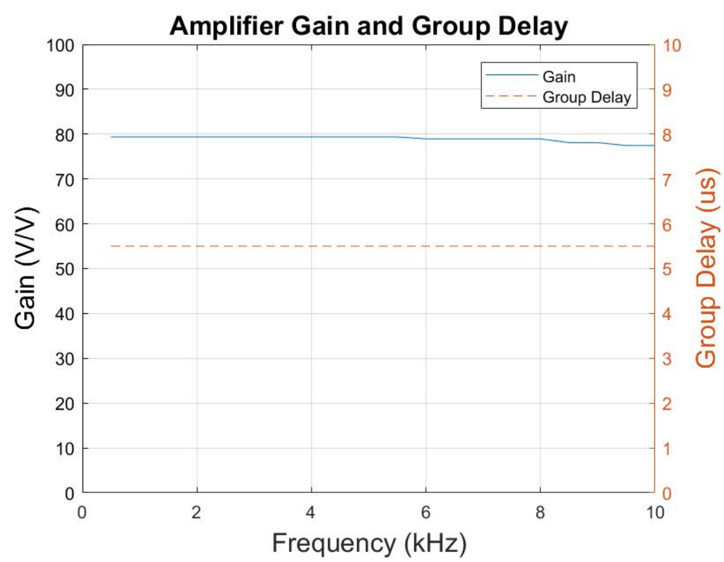


Figure 3.1: Amplifier Verification Measurements (Gain and Group Delay)

3.2.2 Filter Verification

Phase and gain of the filter were measured by performing as sine sweep on the input and measuring the time delay and amplitude of the output result. The gain is easily calculated as the ratio between output and input amplitudes. The unwrapped phase is calculated with knowledge of the time-shifting property;

$$Time\ Delay\ =\ Output\ time\ -\ Input\ time \tag{3.1}$$

$$Unwrapped\ Phase\ =\ \frac{-Delay}{input\ period} \times 360 \tag{3.2}$$

In the final design, near unity gain and near linear phase were achieved in the passband from around 0 to 21 kHz. The output was sufficiently bandlimited, with an attenuation of -27.95 dB at the 34 kHz mark. All filter design specs were met, as can be seen from Figure 3.2 below

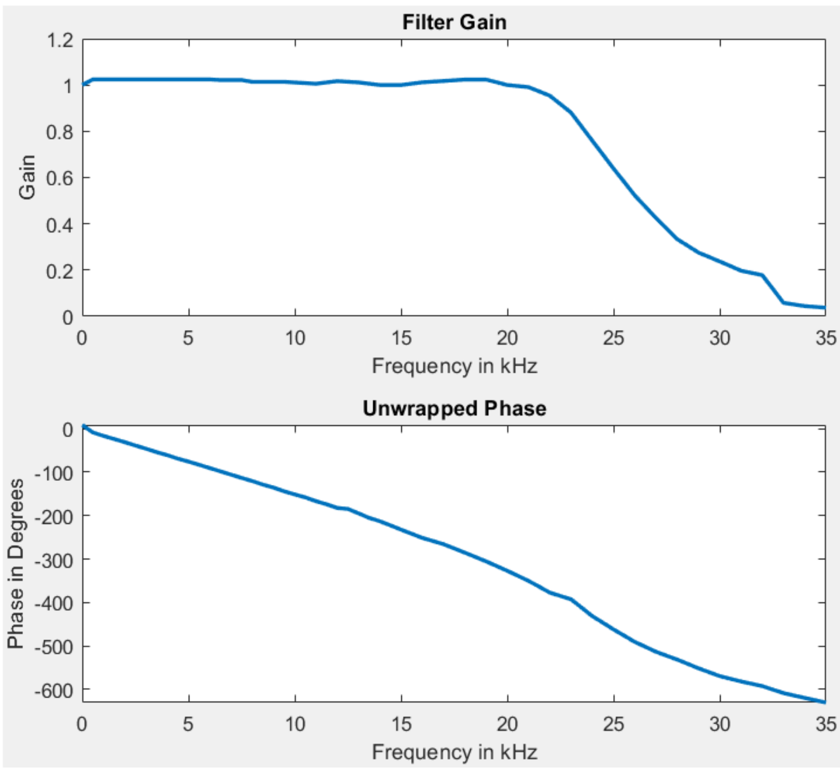


Figure 3.2: Anti-Aliasing Filter Verification Measurements (Gain and Unwrapped Phase)

3.3 Control Subsystem Verification

Verification for the Control Subsystem consists of two types of verification: confirmation of the Teensy 3.6 microcontroller's key functionality requirements, and validation of the system program algorithm by means of assessing the operational accuracy of the system.

To verify the microcontroller's analog signal digitization, voltages within the allowable range for its ADC input pins was applied, and the digital values assigned was observed and compared to the expected values. For input voltages from 0 to 3.3 VDC, at the 13-bit default precision setting, we expected a linear mapping to digital values of 0 to 2^{13}. As shown in Figure A.14, the microcontroller did maintain a linear relationship between input voltages and reported digital values, with an average error of 3.6%, a figure well within the tolerance for its performance demands.

For confirmation of the sampling rate, the microcontroller's analog input was supplied with a sine wave sweep over the range of 20 - 25 kHz. As expected, we observed "wrapping" of the peak frequency reported around 22.05 kHz, as expected. We also confirmed the sampling rate by observing the minimum difference in frequency between two sine wave sources that the microcontroller could resolve, which is a difference of 88 Hz; a figure consistent with preserving one FFT bin width of separation.

For confirmation of the speed of FFT generation of the microcontroller, a timer software library was used to the log the start and end times of the FFT polling and report function returns, yielding results shown in Figure A.16. The average total reporting time returned by the microcontroller is 11.613 ms; at 44.1 kHz sampling rate, for a 1024-point FFT, this reporting time indicates that the microcontroller returns FFT's with a 50% overlap, which was well within the needs of our design.

For validation of the system program algorithm, we conducted accuracy tests in the form of a series of test beatbox patterns, from which we tallied the correct and incorrect sound identifications. A full breakdown of results is detailed in Table A.23. The overall accuracy of the system, based on these tests, is 85.3%, which is well above the 60% target we set for the High Level Requirements of our design.

3.4 Electromechanical Conversion Module Verification

3.4.1 Solenoid Verification

The solenoid has two design requirements: 1 inch throw and 50 ms response time. To verify the 1 inch throw, we measured the throw distance with a ruler. To verify the response time, we devised a test setup that closed a circuit each time the drumstick struck the drum. This circuit was then probe on the oscilloscope along with the command signal to actuate the solenoid and the time difference between the two is the speed of the solenoid. The results is shown in Figure 3.3 and it shows that the speed of the solenoid is 25.1 ms.

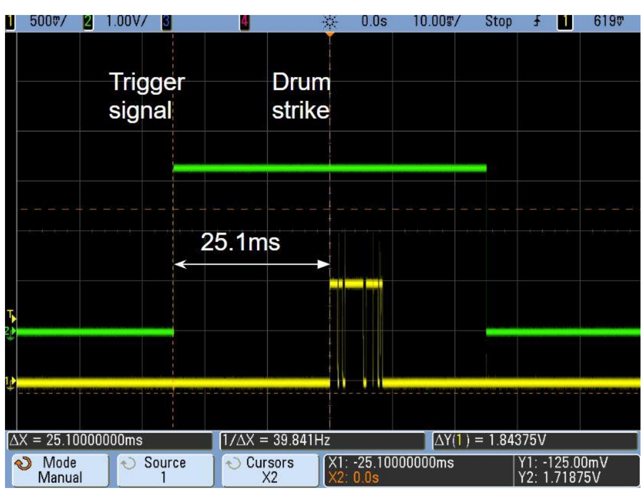


Figure 3.3: Solenoid Speed Verification

3.4.2 Solenoid Driver Circuit Verification

The solenoid driver circuit consisted of two parts, the square-wave amplifier and current driver. Among these two circuits, there were two requirements; amplify a digital signal from 0-3.3V to 0-11.1V and supply the solenoid with 8A of current. To verify these requirements we used the following procedure: power the solenoid driver and solenoid with a battery, use the signal generator to provide a 0-3.3V signal with a pulse width of 50ms, and probe the output of the square-wave amplifier along with the solenoid voltage and current. The results for these measurements are shown in Figure 3.4 and they indicate that the square-wave amplifier requirement has been verified. The solenoid and driver circuit was designed to push 8A through the solenoid however, the results show that the solenoid pulls only 7.45A. Even though this requirement was not attained, the overall EMC module still exceeded the 50ms response requirement that we set for it. The reason for this error may be due to the many approximations we made when calculating the design parameters for the solenoid.

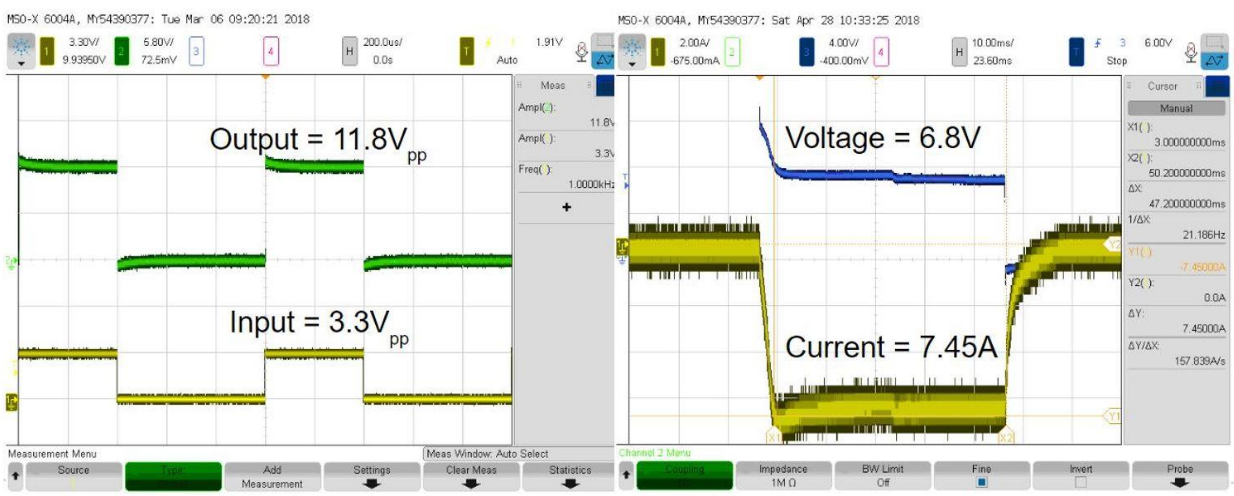


Figure 3.4: Solenoid Voltage and Current Draw

3.5 Mechanical Output Module Verification

The design requirements for the mechanical module are the following; three distinct drum sounds and the drummer operates in real-time (less than 185ms from voice input to drum strike). Both of these requirements have been verified. The actual speed of our system is 95ms from voice input to drum strike. That is 70ms FFT processing time plus 25ms solenoid actuation speed.

Furthermore, measurements of the loudness of the drum strikes observed a peak (maximum) level of 82.1 dB, as shown in Figure A.24. These results demonstrate that our design implementation successfully achieved the minimum output loudness of 40 dB that we sought to achieve.

4. Cost

Our development cost is calculated as follows:

$$\frac{\$30}{hour} \times \frac{16\ hours}{week} \times 15\ weeks \times 3 = \$21{,}600 \quad (4.1)$$

Table 4.1: Component Cost List

Component	Quantity	Cost
Various resistors, capacitors, and diodes		provided
Power Module		
Tattu LiPo battery pack 1300mAh 45C 3S	1	$13.99
LM317 Linear Regulator	2	$1.64
DCJ0202 Power Jack	1	$0.73
Input Module		
LM741 Op-amp	2	$1.46
LM6134 Op-amp	1	$4.85
LM6132 Op-amp	1	$4.00
Behringer Ultravoice XM8500 Dynamic Coil Microphone	1	$20.00
PCB mount 3.5mm stereo jack	1	$0.77
PCB	2	$2.00
Controller Module		
Teensy 3.6	1	$29.25
PCB	1	$1.00
LED	7	provided
Electromechanical Conversion Module		
Pen tube	3	$0.24
Steel rod	3	$6.42
PLA	-	-

Spring	3	$0.12
Shaft Collar	3	$6.00
Magnet wire		provided
IRF510 Power FET	3	provided
PCB	1	$1.00
Mechanical Output Module		
Wood		$20.00
PVC		$1.85
Containers (drums)		Free
Total		$115.32

5. Conclusion

5.1 Accomplishments

Our final design achieved all three of the high level requirements. The first being that it can take audio input from human beat box and distinguish between three distinct sounds. Secondly, our design operates in real-time (less than 185 ms latency from voice input to drum strike). Lastly, our design utilizes a mechanical drummer to produce audible output with at least a 40 dB sound-pressure intensity.

In addition to achieving the high-level requirements, our design also has the capability to tune to a user's voice. The tuning mechanism works by recording three sounds of the user's choice and then maps each one to a drum strike. This feature grants our design the ability to be utilized with a greater sound recognition rate opposed to untuned signal processing.

5.2 Future Improvements

One considerable obstacle we encountered that still needs addressing is the problem of acoustic feedback: depending on the acoustics of the room, the training profile constructed, and the acoustic properties of the drum sound, a drum strike occasionally triggers the system.

To mitigate this ramification we increased the distance between the microphone and drums which renders the microphone less sensitive to drum output. However, a more robust mechanism to solve this problem in software would be to add an exclusionary module. This module could be tuned to the sound of each drum strike so that when a drum is detected no output would be sent to the drummer. This would effectively change our design from a three sound detection system to a six sound detection system where only three sounds are mapped to a drum strike.

5.3 Ethical Considerations

We do acknowledge that, by the standard of the IEEE Code of Ethics [12], tenet #1, and the ACM Code of Ethics and Professional Conduct, General Moral Imperative 1.2 [13], our project does have the potential to injure and shock members of the public through misuse. As such, to the best of our skills and abilities we have endeavored to mitigate these risks to the end-user and public at large, as demonstrated by the explicit inclusion of safety measures in the more hazardous elements of our design, as well as the observation of appropriate work practices in the lab environment itself.

With respect to the former, our design was implemented with insulation for sensitive and potentially dangerous circuit components (capacitors, batteries, etc.) from the user by way of a protective, electrically insulated enclosure.

An additional concern for the end-user pertains to risks of hearing damage, as our device has an acoustic-production aspect to its operation. Per OSHA-issued noise safety standards [14], we advise the end-user and persons within its vicinity of operation to observe OSHA's recommendations for exposure-limits and/or to take appropriate measures with personal protective equipment (e.g. ear-plugs) as seen fit.

With respect to the latter, some specific way in which we consistently practice proper lab safety techniques includes: never rewiring a circuit while it's still powered, and always powering down our lab equipment (including soldering irons) before leaving the bench to attend to another task.

From the perspective of the development process itself, and in deference to the significance placed on the spirit of collegiality by the IEEE Code of Conduct, tenets #7 and #10, we acknowledge the value inherent to recognizing the skills and background of the team-members, and strove to provide sufficient opportunity to each team-member for contribution to the project, as well as to further their own professional growth. This also has implications for adhering to tenet #9, whereby misrepresentations of knowledge or ability may put members of the team, as well as the public at large, at risk, where the demands of a particular design task carry safety considerations. As such, we have sought out expert knowledge and ability to assist us during the design process, whenever such circumstances arose; and shall continue to adhere to these practices going forward as well.

From the perspective of issuing proper credit to third-parties' ideas and work, per the ACM Code of Ethics and Professional Conduct, General Moral Guideline 1.6, we have and shall continue to recognize the sources of inspiration for solutions to engineering challenges applied, as has been demonstrated throughout this document. It is not our intent to misrepresent our work, and have done our utmost to chronicle and cite the sources of our methodologies and implementations accordingly.

References

[1] Evolver.fm (2011). The Beatbox Machine Turns Your 'Beats' into Actual Drumbeats [Online]. Available: http://evolver.fm/2011/09/26/the-beatbox-machine-turns-your-beats-into-actual-drumbeats/

[2] Synthtopia (2010). New App Lets you Beatbox to MIDI On Your iPhone [Online]. Available: http://www.synthtopia.com/content/2010/09/07/new-app-lets-you-beatbox-into-your-phone-edit-the-beat-on-your-mpc/

[3] ITU Relative Timing of Sound and Vision for Broadcasting, ITU Recommendation BT.1359-1 (1998). Available: https://www.itu.int/dms_pubrec/itu-r/rec/bt/R-REC-BT.1359-1-199811-I!!PDF-E.pdf

[4] C. K. Alexander, M. N. O. Sadiku, "Fundamentals of Electric Circuits," 5th ed., 2013, pp. 181-182

[5] Texas Instruments (2018) "FilterPro" [Web Application]. Available: http://www.ti.com/design-tools/signal-chain-design/webench-filters.html

[6] PJRC.com. *ADC library update, now with support for Teensy 3.1* [Online]. Available: https://forum.pjrc.com/threads/25532-ADC-library-update-now-with-support-for-Teensy-3-1?p=48778&viewfull=1#post48778

[7] PJRC.com. *Teensy 3.6 ADC resolution* [Online]. Available: https://forum.pjrc.com/threads/41911-Teensy-3-6-ADC-resolution

[8] Freescale Semiconductor, Inc. (2016) Kinetis K66 Sub-Family: 180 MHz ARM Cortex-M4F Microcontroller [Online] Available: https://www.pjrc.com/teensy/K66P144M180SF5V2.pdf

[9] PJRC.com. *Audio System Design Tool for Teensy Audio Library* [Online]. Available: https://www.pjrc.com/teensy/gui/index.html?info=AudioAnalyzeFFT1024

[10] B. T. Last and K. K. Paliwal, "ROBUST SPEECH RECOGNITION USING SINGULAR VALUE DECOMPOSITION BASED SPEECH ENHANCEMENT," semanticsscholar.org, 1997. [Online]. Available: https://pdfs.semanticscholar.org/d64e/b07ded99ba07f36538f8cab2aa1f526d7d2b.pdf..

[11] R. Hasan, M. Jamil, G. Rabbani, and S. Rahman. (2004) "Speaker Identification Using Mel Frequency Cepstral Coefficients," *3rd International Conference on Electrical & Computer Engineering ICECE 2004*, 28 - 30 December 2004. [Online] Available: https://www.researchgate.net/profile/Golam_Rabbani4/publication/255574793_Speaker_Identification_Using_Mel_Frequency_Cepstral_Coefficients/links/55f05d5908ae0af8ee1d1894.pdf

[12] IEEE.org. (2018). *IEEE Code of Ethics.* [Online]. Available: https://www.ieee.org/about/corporate/governance/p7-8.html

[13] ACM.org (20180. *ACM Code of Ethics and Professional Conduct.* [Online]. Available: https://www.acm.org/about-acm/acm-code-of-ethics-and-professional-conduct

[14] *Occupational Noise Exposure*, OSHA Recommendation 1910.95 (Amended 2008). Available: https://www.osha.gov/pls/oshaweb/owadisp.show_document?p_table=STANDARDS&p_id=9735

Appendix A: Figures

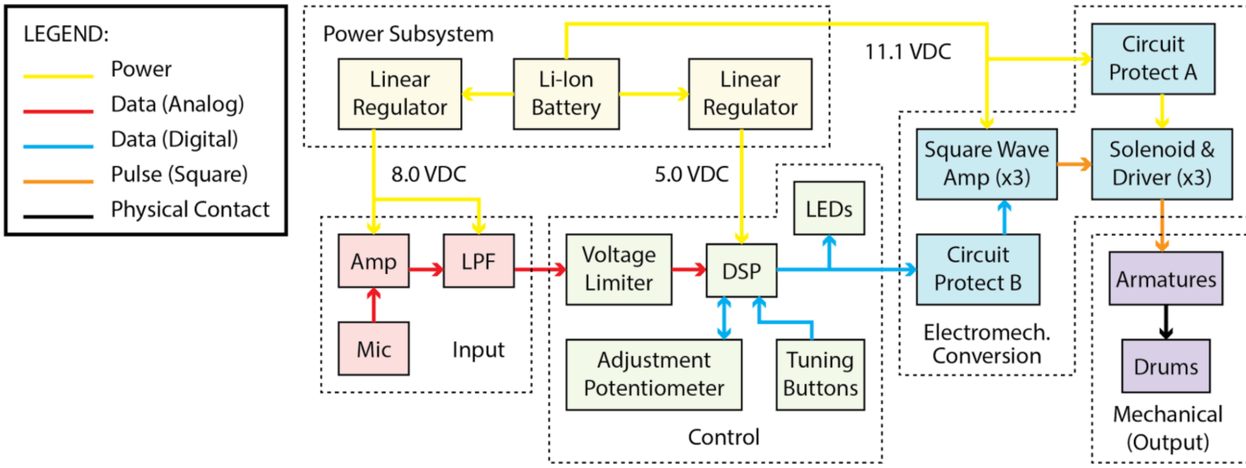
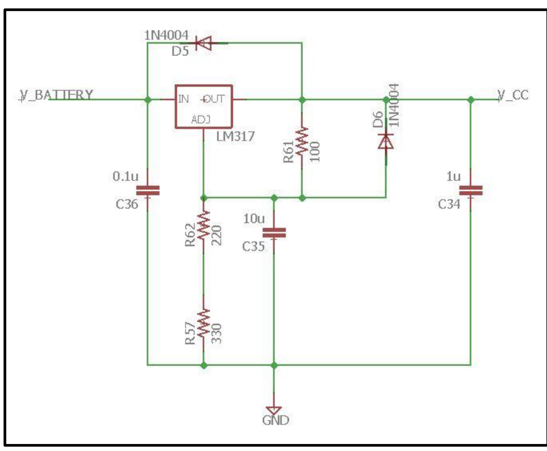


Figure A.1: High-Level Block Diagram



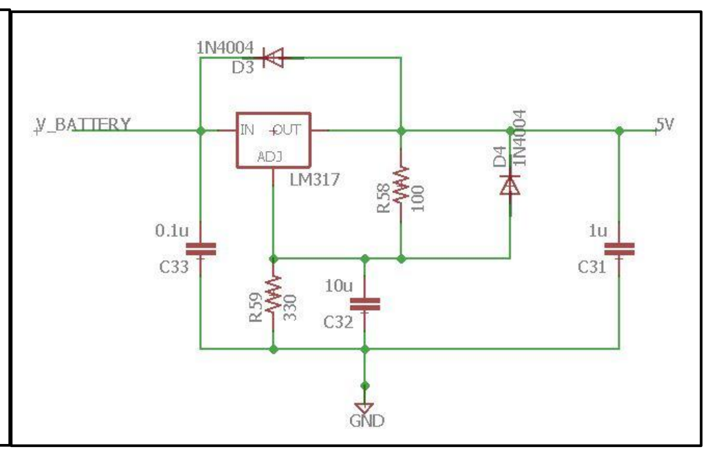


Figure A.2
Amplifier and Filter Linear Regulator Schematic

Figure A.3
Teensy 3.6 Linear Regulator Schematic

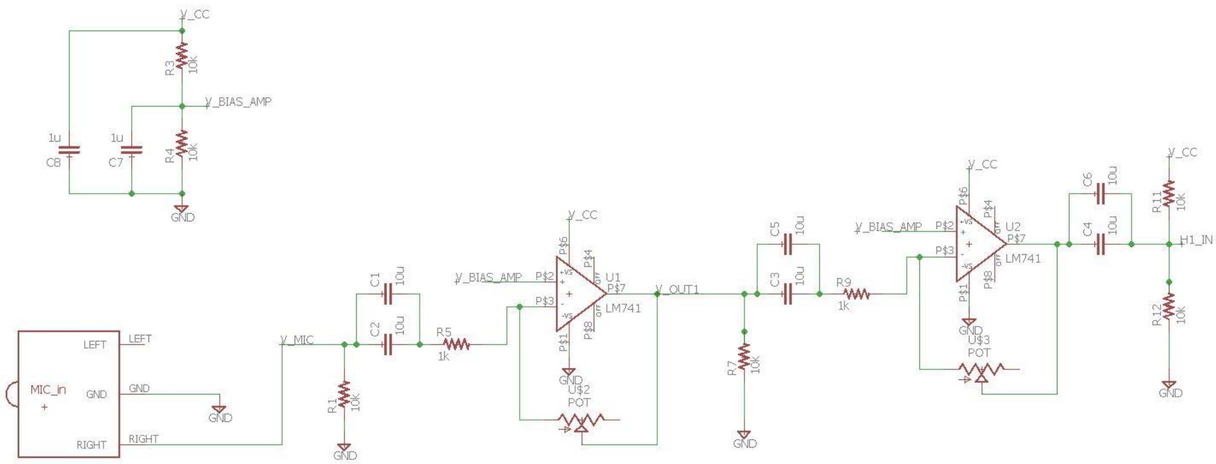


Figure A.4: Amplifier Schematic

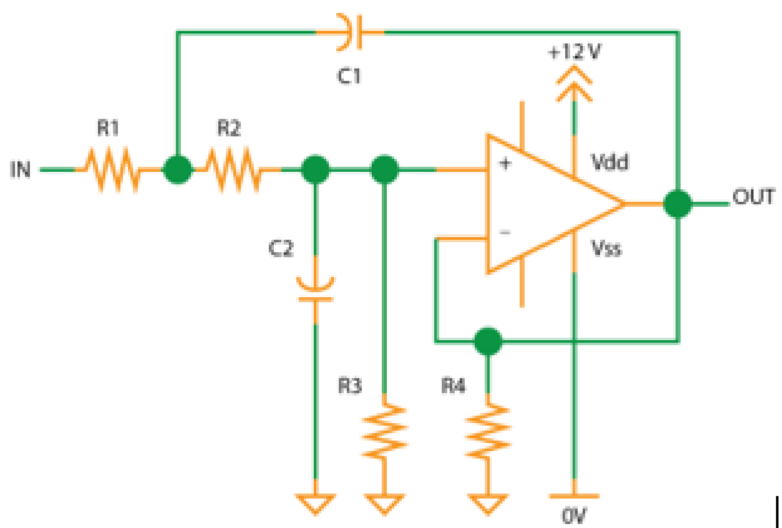


Figure A.5: Filter Circuit Stage (5 Stages total)

Stage	R1	R2	R3	R4	C1	C2
1	4.3 kΩ	9.1 kΩ	330 kΩ	330 kΩ	1.2 nF	1nF
2	3.6 kΩ	8.2 kΩ	330k	330k	1.5 nF	1nF
3	3.3 kΩ	6.2 kΩ	330k	330k	2.2 nF	1nF
4	2 kΩ	4.3 kΩ	330k	330k	5.6 nF	1nF
5	680 Ω	1.5 kΩ	330k	330k	47 nF	1nF

Table A.6: Filter Circuit Stage Resistor and Circuit Values

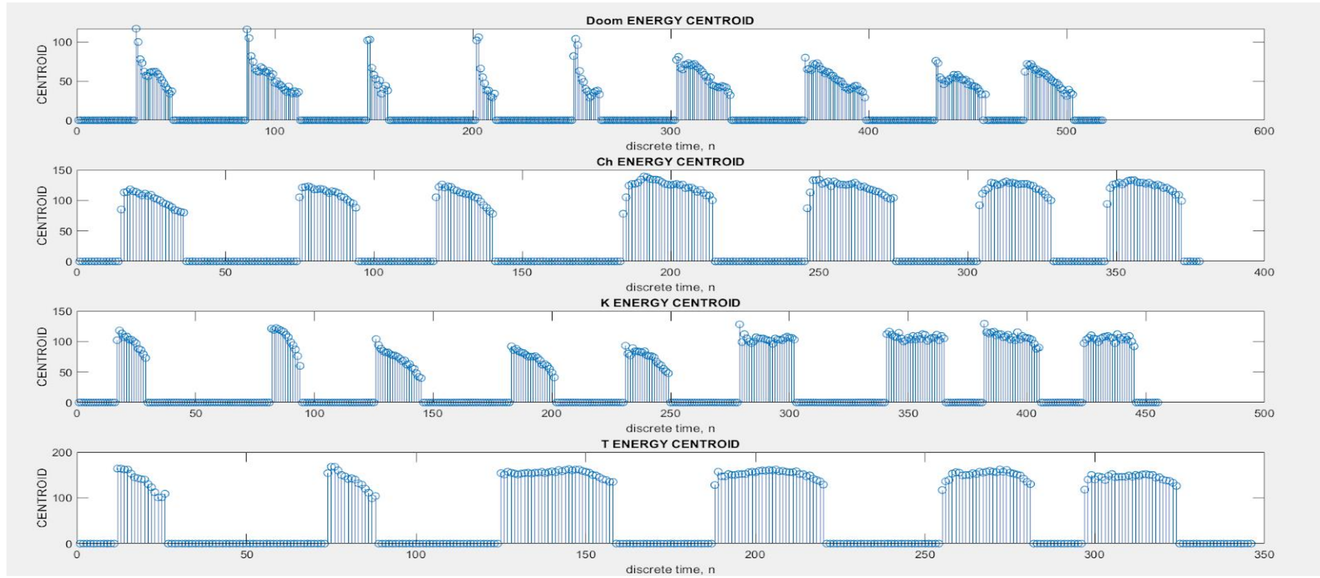
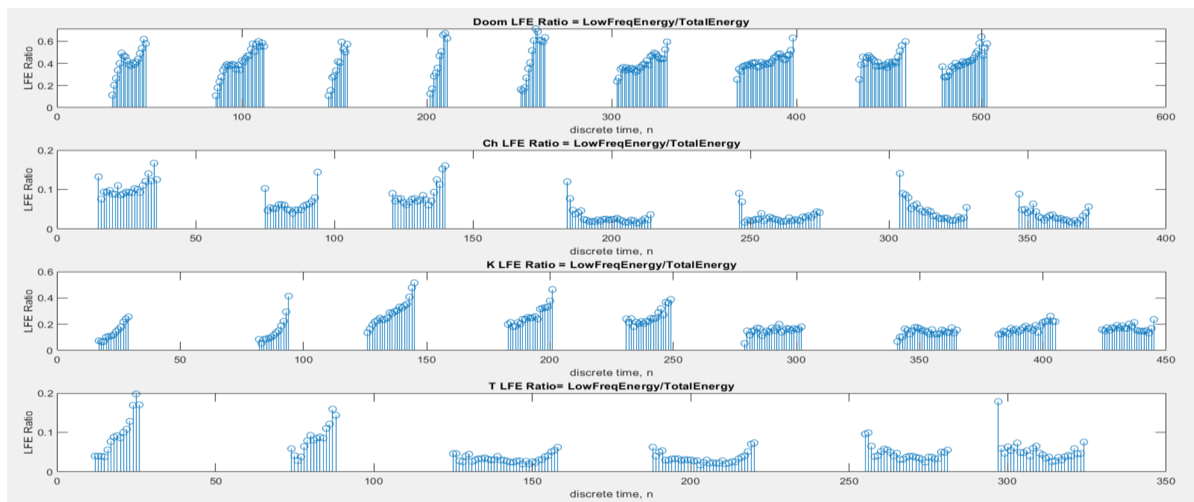


Figure A.8: Low Frequency Energy Ratio (LFER) for Various Beatbox Sounds ıts for Various Beatbox Sounds



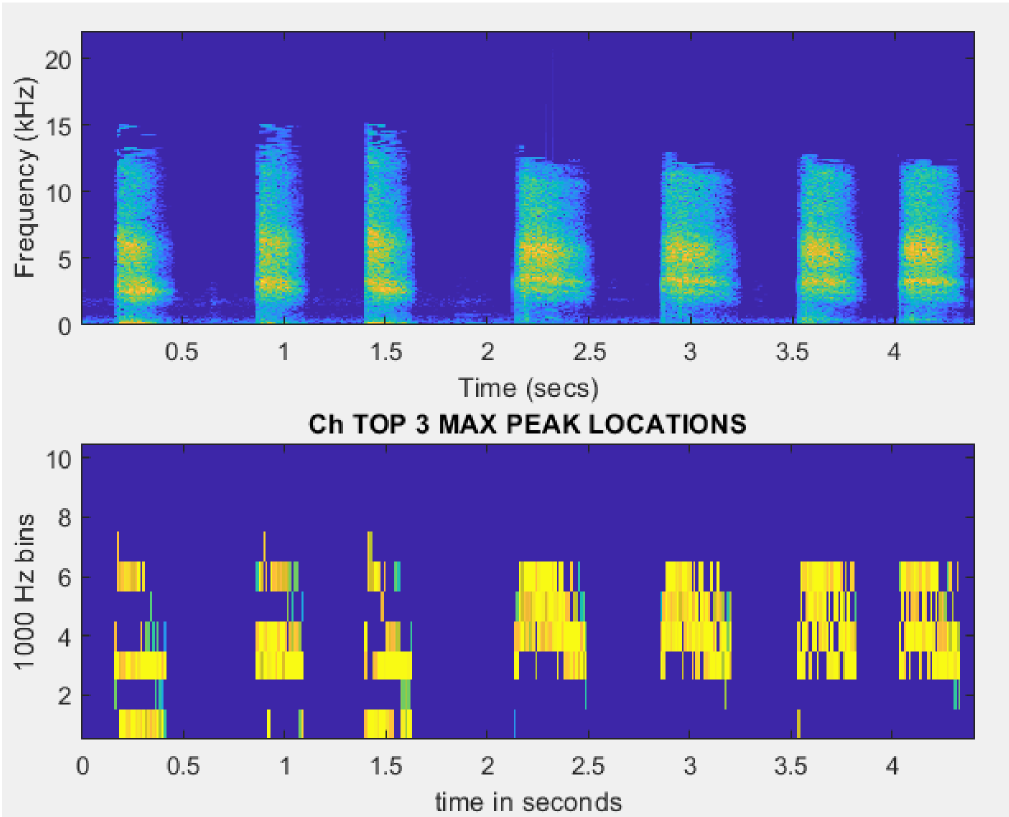


Figure A.9: Top 3 Peak Locations for the 'Ch' Sound
(top: Original Spectrogram, bottom: Top 3 peak spectrogram)

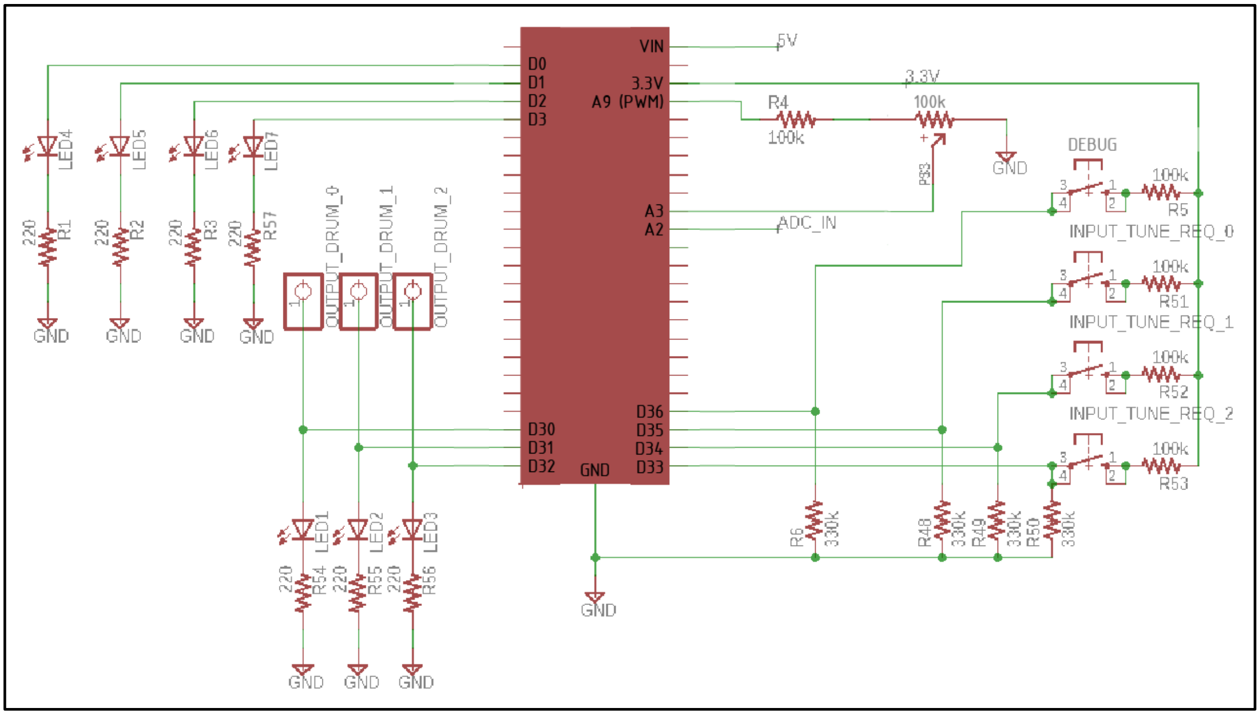


Figure A.10: Schematic for Control Subsystem

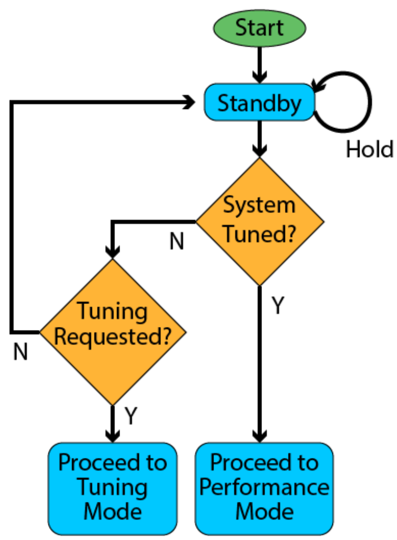


Figure A.11: Flowchart for System Software Operational Modes

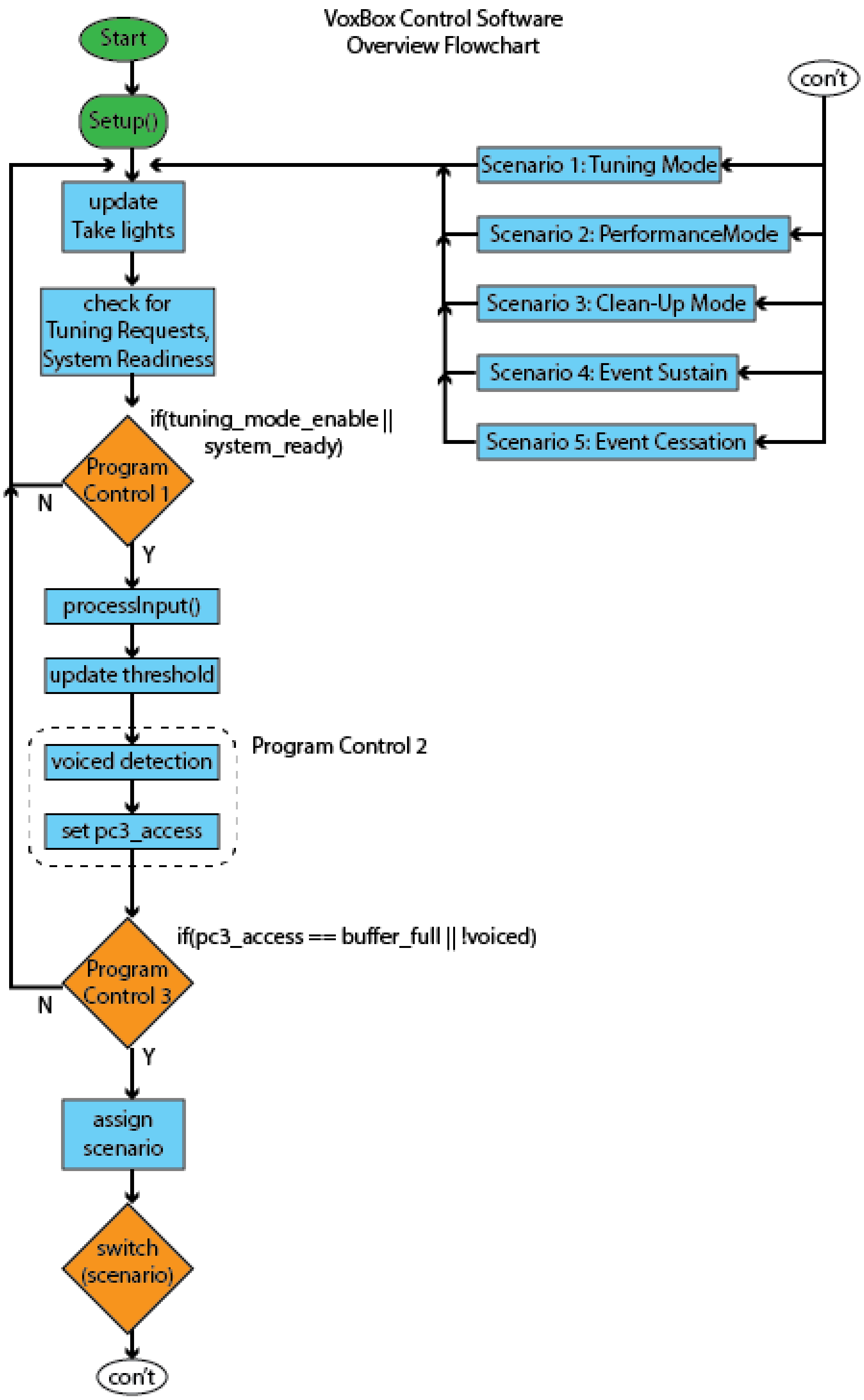


Figure A.12: System Program Flowchart

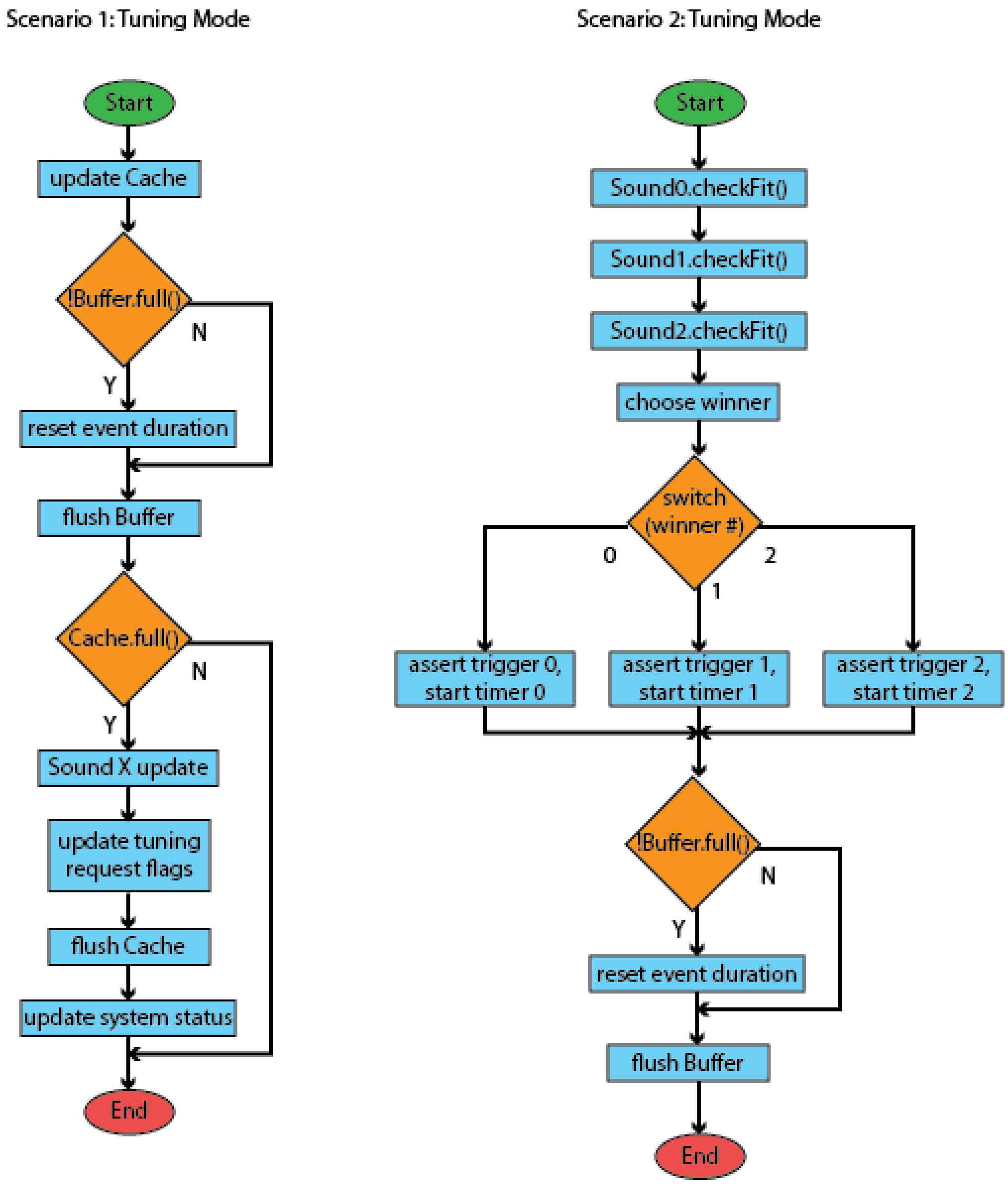


Figure A.13: Tuning Mode and Performance Mode Subroutine Flowcharts

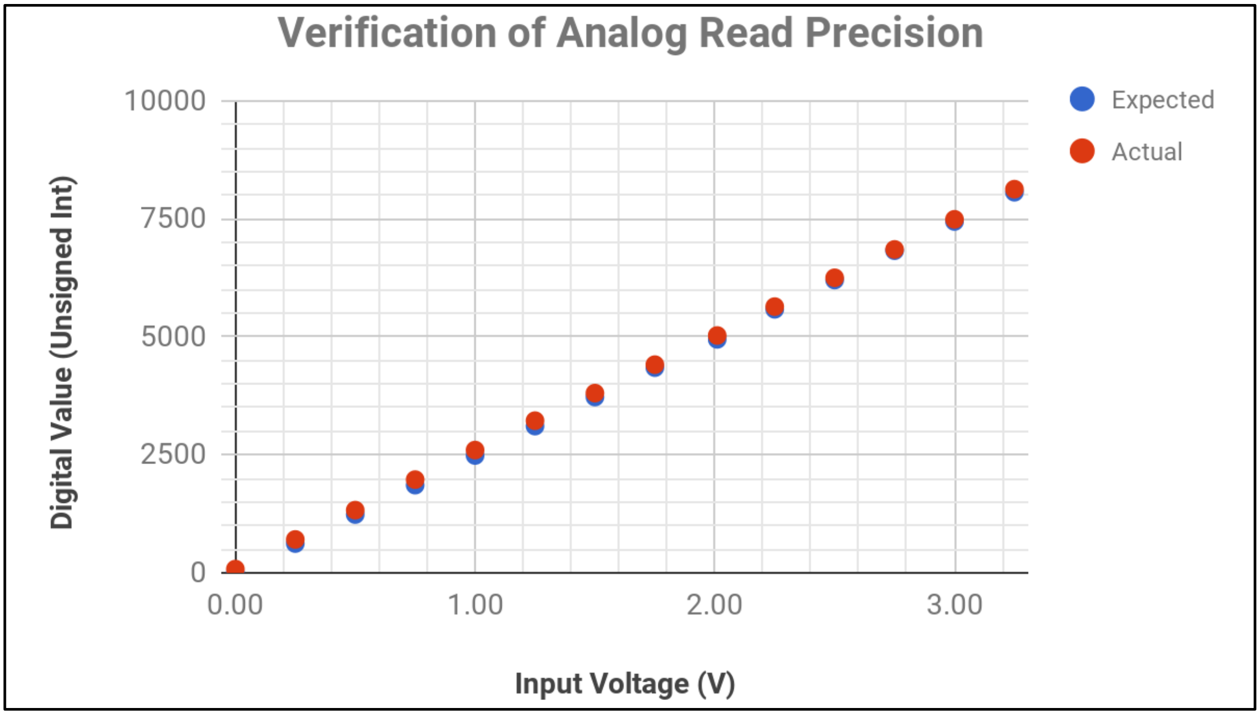


Figure A.14: Plot of Analog Read Precision Results

Table A.15: Experimental Results for Analog Read Precision

Input Voltage (V)	Expected	Actual	% Error
0.00	0	80	
0.25	621	708	0.140
0.50	1236	1329	0.075
0.75	1859	1976	0.063
1.00	2487	2602	0.046
1.25	3110	3225	0.037
1.50	3726	3809	0.022
1.75	4349	4412	0.014
2.01	4948	5030	0.008
2.25	5585	5642	0.010
2.50	6205	6253	0.008
2.75	6827	6854	0.004
3.00	7446	7492	0.006
3.25	8066	8131	0.008
		Average Error:	0.03675

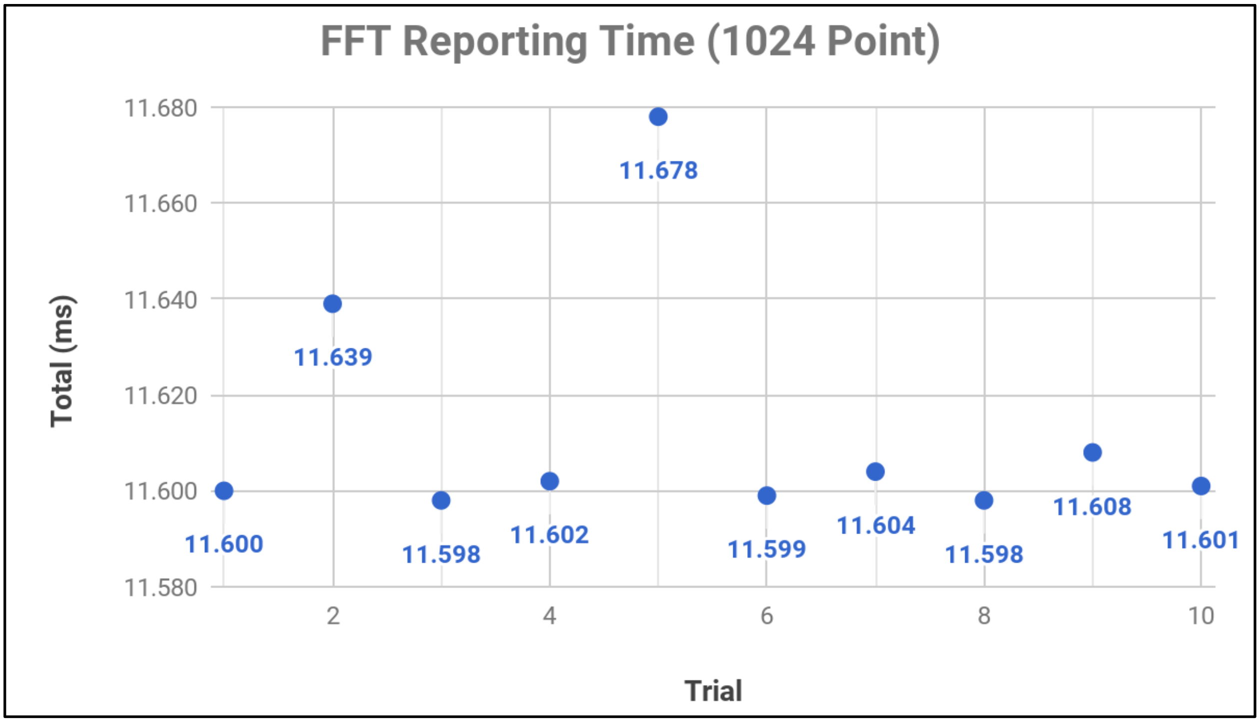


Figure A.16: Plot of FFT Reporting Time Results

Table A.17: Experimental Results for FFT Reporting Time

Trial	FFT.read() (ms)	FFT.available() (ms)	Total (ms)
1	11.586	0.014	11.600
2	11.585	0.054	11.639
3	11.584	0.014	11.598
4	11.588	0.014	11.602
5	11.585	0.093	11.678
6	11.583	0.016	11.599
7	11.590	0.014	11.604
8	11.584	0.014	11.598
9	11.586	0.022	11.608
10	11.587	0.014	11.601
		Average (ms):	11.613

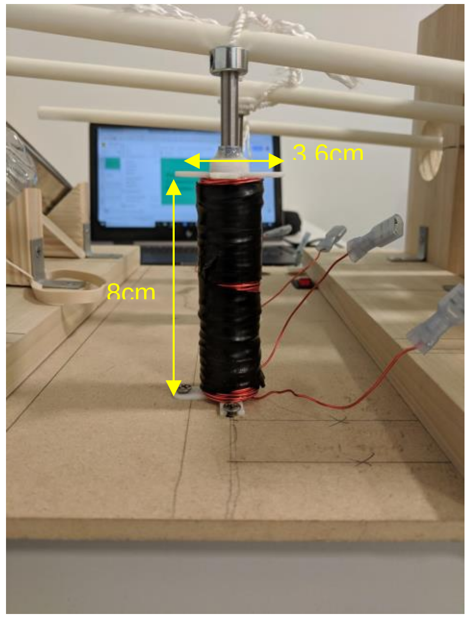


Figure A.18: Solenoid Image

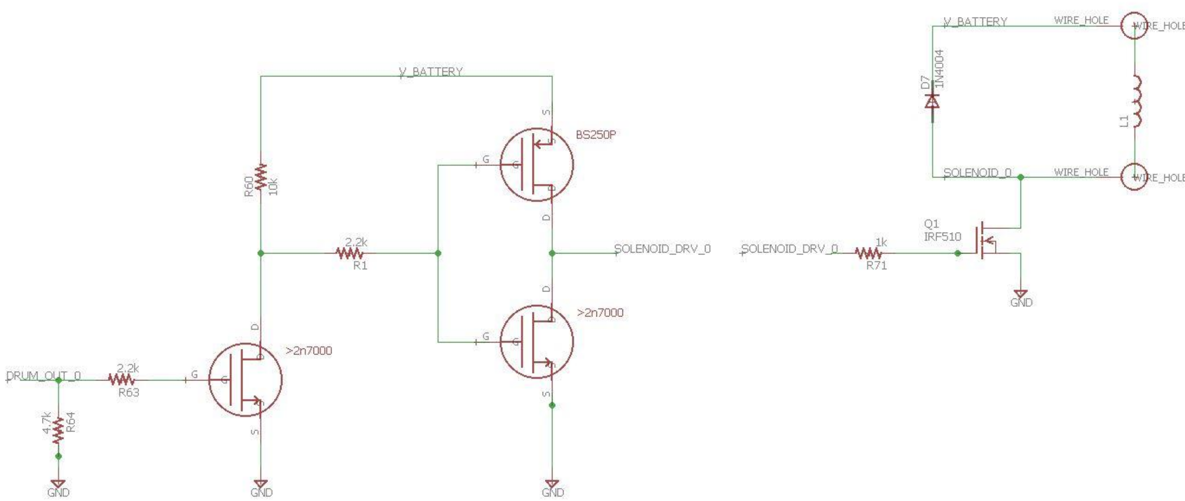


Figure A.19: Solenoid Driver Schematic

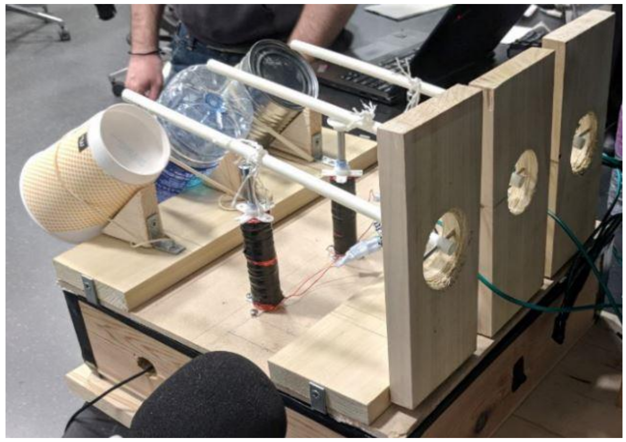


Figure A.20: Mechanical Drummer

Table A.21: Power Module Components and Power Draw

Component	Voltage	Current	Power
Tattu LiPo battery pack 1300mAh 45C 3S	11.1V	Dependent on solenoid actuation rate	
LM317	11.1V in, 8V out	15.5mA	172mW
LM317	11.1V in, 5V out	163mA	1.81W

Table A.22: Solenoid Parts

Component	Material	Dimensions
Guide tube	Pen tube	Length = 9cm
Plunger	Steel rod	Diameter = 3mm
Magnet wire		Size = 22 AWG, 640 turns
Guide tube caps	3D printed PLA	Diameter 3.6 cm
Spring		Length = 3 cm

Plunger cap	Steel collar	

Table A.23: Final Design Accuracy Results

Sound Type	Correct	Total	Accuracy
"Bass Kick"	78	85	0.9176470588
"Ts"	50	68	0.7352941176
"Ch"	40	44	0.9090909091
Total	**168**	**197**	**0.8527918782**

Figure A.24: Sound Meter Readings of Drum Strikes