ILLINI THEREMIXER

By

Karthik Achar
Shiv Kapur
Akhil Reddy

Final Report for ECE 445, Senior Design, Spring 2019
TA: Zhen Qin

1 May 2019
Project No. 15

Abstract

The Illini Theremixer is a project that aims to utilize the fundamental characteristics of a traditional musical instrument, the Theremin, to achieve modern music applications. By utilizing hand movements relative to the antennas on the Theremixer, users can mix music. The relative hand placement with the function select antenna (the antenna that is vertical) dictates the current filter applied to the live audio: bass boosting, mid boosting, treble boosting, or volume boosting. On the other hand, the relative placement with the magnitude antenna (the antenna that is horizontal) dictates the magnitude at which the current filter is amplified or attenuated. This report entails the research and development of the Illini Theremixer, and the various successes and challenges during the course of this project. Our primary results included various modules that functioned as per requirement, but we faced several challenges in properly integrating all parts together.

Contents

1 Introduction

Throughout human history, innumerable musical instruments have been created to allow people to express themselves in their own ways. One such instrument is the Theremin, which is notable as an instrument that can be operated entirely without touching it. Its ingenious design utilizes two antennae, which individually control the pitch and volume of the output sound based on the distance of the player's hands from each antenna. The Theremin is named after its inventor Léon Theremin, who received a patent for the device in 1928.

Theremin's original design used vacuum tubes and because of this operated at very high voltages; since its invention, multiple new designs have been created to adapt to modern tastes and technologies. The antennae in almost all designs function as variable capacitors, with the antenna itself serving as one plate and the user's hand (or any other closer object) serving as the second, grounded plate. Modern analog Theremin designs often use dual oscillators and a heterodyne detector to extract pitch as a difference frequency, and while digital Theremin are now common and generally cheaper than their analog counterparts, analog Theremins are generally considered to have better sound quality and are preferred by serious musicians.

1.1 Purpose

After learning more about the Theremin and its history, we decided to bring a more modern light to this often-overlooked instrument. The Theremixer is an effort to repurpose analog Theremin technology to modify existing music, as one would with an equalizer or DJ controller. As music technology has advanced, the price and complexity of necessary equipment has also increased steeply. However, by taking advantage of the Theremin's gesture-based controls, we hoped to design a simplistic and intuitive mixer capable of performing basic functions such as editing bass, midrange, treble, and volume in real time.

1.2 Functionality

To accurately measure the success of our project, we developed three high-level requirements that encompass the critical features of this project.

1. The output signal from the magnitude oscillator will have a frequency of 142 kHz ±10%.
2. The output signal from the function select oscillator will have a frequency of 430 kHz ±10%.
3. Teensy Microprocessor will apply filters to increase and decrease bass, treble, mid, and volume levels of the imported song based on the output signals of the function and magnitude oscillators.

The first two high-level requirements are important as they establish the control signals for the microprocessor's filters and are critical to determine changes in relative hand positioning near the Theremixer's antennas. The third high-level requirement is important to the success of this project because it implements the DSP filters on the microcontroller to provide the Theremixer mixing capabilities.

A fully integrated Theremixer will use the output waveforms from the magnitude and function select oscillator as control signals to apply the various filters. Figure 1 indicates the physical design of the Theremixer. The function select antenna protrudes vertically out of the housing, while the magnitude detection antenna protrudes horizontally from the housing. The two antennas were placed on different axes to avoid unintended interference while playing the Theremixer. To play the instrument, a user places one hand in the desired function region near the function select antenna and another hand in the desired magnitude region near the magnitude detection antenna. For example, to decrease bass to a 25% level, a

user should place one hand near the function select antenna in the “Bass” region (purple ovals) as depicted in Figure 1 and one hand near the magnitude detection antenna in the “25%” region.

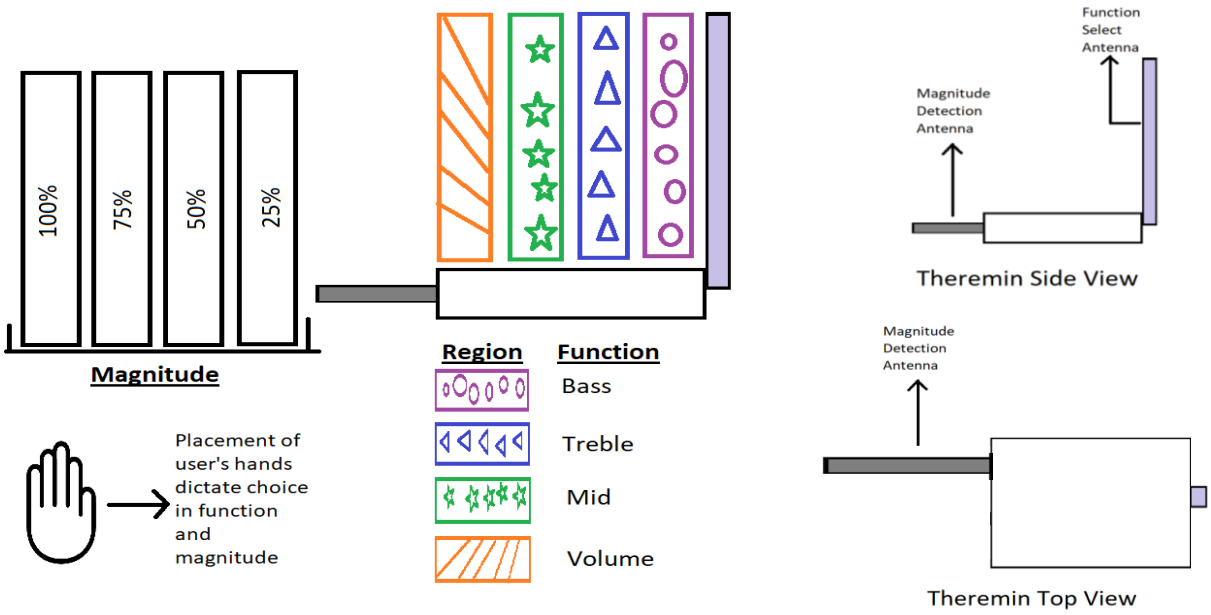

Figure 1: Theremixer Physical Design and Functionality

1.3 Subsystem Overview

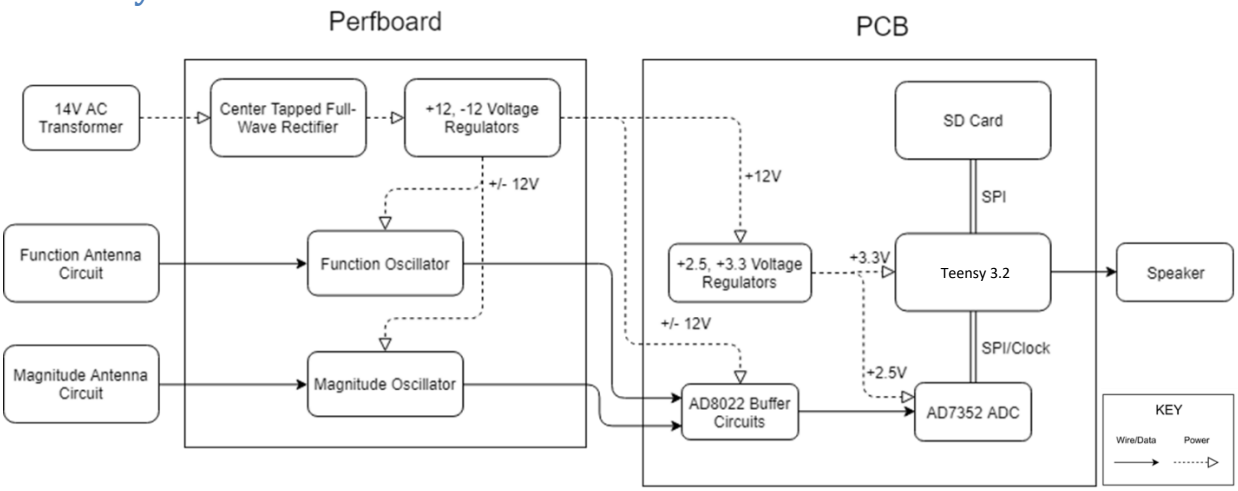

Figure 2: Block Diagram of Theremixer Modules

As Figure 2 illustrates, the Theremixer comprises of several subsystems that work together to provide necessary functionality. These subsystems include the Power Module, the Hardware Module, the Sampling Module, and Software Module. The Power Module includes the 14 V AC Transformer which is off board, the Center Tapped Full-Wave Rectifier and ± 12 Regulators on a perfboard, and the +2.5 V Regulator and + 3.3 V Regulator on the PCB. The Hardware Module consists of the function and magnitude antenna circuits which are off board and the function and magnitude oscillator circuits which are on the perfboard. The Sampling module includes the AD8022 Buffer Circuits and the AD7352 Analog to Digital Converter that are placed on the PCB. The Software module includes the Teensy Microprocessor that supports an input from an SD card and output to a speaker, all on the PCB.

2 Design

We built the Theremixer in four separate subsystems. The first subsystem is the power module which provides power to the other subsystems. The second subsystem is the hardware module which includes the antenna circuits and the oscillator circuits. The hardware portion is critical to establish the control signal for the software portion. The oscillator circuits create a sinusoidal waveform with a frequency that is altered by capacitance changes relative to hand positioning near the antenna. The outputs (two sinusoidal waveforms) are fed into the sampling components. The sampling components are the third subsystem in our design. This portion consists of buffering circuits and our ADC (AD7352). This subsystem processes the control signal and converts the analog input into a digital output that is processable by the software subsystem. The software subsystem consists of the Teensy 3.2 microcontroller which uses SPI communication to read the input from the sampling subsystem. The software subsystem also includes appropriate code to perform audio filtering operations and output the altered audio signal to a speaker.

2.1 Power

The power supplies we created worked through a 14 V AC Wall Transformer. Using this input, we created $\pm$ 12 V power rails to run the oscillator circuits, buffer circuits, + 2.5 V power supply, and + 3.3 V power supply. The + 2.5 V supply provided power to the AD7352. The + 3.3 V supply provided power to the Teensy 3.2 Microcontroller. In designing these power supplies, we kept in mind that we needed stable and consistent voltage outputs to avoid damaging parts. For this reason, we included several bypass capacitors at the inputs and outputs of our regulators and used parts that were properly rated for our use.

2.1.1 14 V AC Transformer

In this project, we used the Hammond Manufacturing 14 V AC Transformer BPD2EE to convert the input voltage to be used by a center-tapped full-wave rectifier. This center-tapped full-wave rectifier is based from the Etherwave Theremin Design and is intended to convert the half cycles of the 14 V AC signal into a pulsating DC signal with current flow in a single direction into the voltage regulator circuit using two D1N4001 diodes. The output of the voltage rectifier is the voltage regulator. We chose to also include a switch between the 14 V AC Transformer's positive line and the input of the rectifier to make powering the circuit easier. Figure 3 in Appendix B is the circuit schematic of the 14 V AC Transformer and Figure 4 is the circuit schematic of the Center-Tapped Full-Wave Voltage Rectifier.

2.1.2 $\pm$ 12 V Regulator

The $\pm$ 12 V regulator creates power rails that are used by both oscillator circuits and the + 2.5 V regulator and + 3.3 V regulator. The voltage regulator circuit uses the LM7912 and LM7812, 3-Terminal Positive/Negative Regulators, to create the positive and negative voltage rails. The regulator schematic is adapted from the component data sheet to reflect the intended use. Figure 5 in Appendix B is the circuit schematic for this component.

2.1.3 + 2.5 V Regulator

The + 2.5 V Regulator powers the AD7352 ADC. It uses bypass capacitors at the inputs and outputs to reduce ripple voltage and smooth out the incoming and outgoing signals. The + 2.5 V Regulator also uses pin 4 to provide more ripple reduction and smoothing. We set tied the Enable Pin (Pin 3) to Vin to ensure that the voltage regulator was always functioning while the circuit was powered. The + 2.5 V Regulator used was the LP2985 regulator by Texas Instruments. The regulator schematic is adapted from the component data sheet to reflect the intended use. Figure 6 in Appendix B is the circuit schematic for this component.

2.1.4 + 3.3 V Regulator

The + 3.3 V Regulator powers the Teensy 3.2 microcontroller. It uses bypass capacitors at the inputs and outputs to reduce ripple voltage and smooth out the incoming and outgoing signals. The regulator used in this project was the LP2950 by Texas Instruments. The regulator schematic is adapted from the component data sheet to reflect the intended use. Figure 7 in Appendix B is the circuit schematic for this component.

2.2 Hardware

The Hardware Subsystem consists of the antenna circuits and oscillator circuits. The overarching purpose of the hardware subsystem is to create analog control signals to be sampled by the sampling subsystem. The antenna circuits and oscillator circuits are based from the Etherwave Theremin design [2]; however, parts were replaced based on availability. The oscillators are designed to create a stable high frequency sinusoidal waveform that has an increasing frequency when a hand comes near the oscillator's respective antenna. These high frequency sinusoidal waveforms serve as the control signals for the Theremixer.

2.2.1 Antenna Circuits

The antenna circuits connect to the respective oscillator circuits as indicated in the circuit schematics. Our antennas are designed based on the fundamental concept of the Etherwave Theremin design. In the Etherwave Theremin design, a copper plated tube serves as an effective capacitor with the nearest object as the ground plane. As a user's hand comes near the antenna, the effective capacitance of the antenna increases due to the distance between the antenna and ground plane reducing. The function select antenna was positioned vertically on the left side of the Theremixer, while the magnitude detection antenna was positioned horizontally on the right side of the Theremixer. We aligned the two antennas on different planes to avoid interference while playing the instrument.

2.2.1.1 Function Select Antenna Circuit

The function select antenna consists of a wire soldered onto a ¾ inch copper plated tube connected to four 10 mH inductors in series. Figure 8 in Appendix B is the circuit schematic for this component.

2.2.1.2 Magnitude Detection Antenna Circuit

Our magnitude detection antenna consists of a wire soldered onto a ¾ inch copper plated tube connected to two 2.5 mH inductors and one 5 mH inductor in series. We also use a 1N4148 and a 1000 pF capacitor to help regulate the output. Figure 9 in Appendix B is the circuit schematic for this component.

2.2.2 Oscillator Circuits

The oscillator circuits were adapted from the Etherwave Theremin design; however, parts were replaced based upon availability. The oscillator circuits are connected to their respective antenna circuits. As the antennas increase their effective capacitance, the effective impedance of the entire system is affected. As a hand comes near the antenna, the frequency of the oscillators increases with the increase of antenna capacitance.

We chose to use the Etherwave Theremin design for our oscillators because it seemed advantageous over other oscillator designs. The oscillator circuits are fundamental in establishing a control signal that can be used to affect filters. To be used as a control signal, the oscillators must be receptive to hand movements near the respective antenna. We considered using the Shockley and Harley oscillators instead of the Etherwave Theremin oscillators. However, after performing simulations we could not find a feasible way to connect the antenna such that the oscillator frequency would reliably change due to hand movements. In these simulations, we modeled our antenna as a capacitor and stepped through various capacitance

values. In the Etherwave Theremin design simulation, we saw a significant frequency change at small capacitance changes. For this reason, we chose to build our Theremixer using the Etherwave Theremin design, however we realized the design had many unnecessary components. Since we only needed to establish a control signal, we chose to use the variable pitch oscillator circuit as our function select oscillator and the volume oscillator circuit as our magnitude oscillator. These were the only circuits from the Etherwave Theremin design that we used. We considered using two variable pitch oscillator circuits or two volume oscillator circuits instead of using one of both. After building and testing each, we found that there were no significant advantages of either. Since both oscillator circuits worked as needed and provided smooth waveforms, we decided to stick as closely with the original design to use one of both type of oscillator for the Theremixer.

In both simulations and testing, we trialed with different inductor values across the first transistor Collector and the + 12 V Rail. We found that increasing the inductance value decreased the oscillation frequency and output voltage. We considered changing this inductance value however upon further research we found that increasing the inductance would decrease the overall effect of hand movements near the antenna. Although we were not able to physically see the difference in our built circuit, in LTSpice simulations we confirmed this to be true. While LTSpice was an immense help in simulating the design of these oscillator circuits we found many inconsistencies when testing the design on a protoboard. Primarily, the oscillator frequencies and peak-to-peak voltages were higher in simulation than real testing. This did not influence our design or implementation, however showed that simulations do not always relay to applications identically.

2.2.2.1 Function Select Oscillator Circuit

The function selects antenna circuit feeds into this oscillator at the first transistor's collector to create the sinusoidal pitch waveform. This waveform will be sensitive to hand positioning near the pitch antenna. The transistors we used were 2N3904 NPN bipolar transistors by ON Semiconductors. The function select oscillator is based on the pitch oscillator of the Etherwave Theremin design. The function select oscillator operates at an average frequency of 431.15 Hz with an average peak-to-peak voltage of 3.51 V. Figure 10 in Appendix B is the circuit schematic for this component and Figure 11 in Appendix B is a simulation of this circuit.

2.2.2.2 Magnitude Detection Oscillator Circuit

The magnitude detection oscillator will create a sinusoidal waveform with varying frequencies based on hand positioning near the volume antenna. The transistors we used were 2N3904 NPN bipolar transistors by ON Semiconductors. The magnitude detection oscillator is based on the magnitude oscillator of the Etherwave Theremin design. The magnitude detection oscillator has an average frequency of 141.76 kHz and an average Peak-to-Peak of 761.29 mV. The magnitude detection oscillator had a slightly odd waveform; however, it was periodic and reliable. Figure 12 in Appendix B is the circuit schematic for this component and Figure 13 in Appendix B is the simulation of this circuit.

2.3 Sampling

The Sampling subsystem consists of the buffer circuits and the Analog to Digital Converter. The purpose of the Sampling subsystem is to convert the analog signals from the oscillators into digital signals that can be sent to the microprocessor and was necessary because the input oscillator signals had frequencies too high for the Teensy 3.2's onboard ADC to reliably sample. The AD7352 requires low impedance inputs for fully reliable operation, and so it was necessary to include buffer circuits as per the AD7352's datasheet. The buffer circuits are fed the analog input from the oscillator circuits and output signals to the ADC. The AD7352 then converts these analog inputs to digital outputs to be sent to the microprocessor.

2.3.1 Buffer Circuits

Our PCB design includes two sets of buffer circuits, one for each control signal from the oscillator circuits. They are direct replicas of the recommended layout from the AD7352 datasheet, using AD8022 dual op-amp chips arranged such that any bipolar single-ended input signal is converted to a pair of differential unipolar signals that can be sent to the ADC's input pins. Figure 14 in Appendix B includes a picture of the PCB.

2.3.2 Analog to Digital Converter

Our original plan for this submodule was to use the AD7367 1 MSPS ADC, but we eventually decided against this component because we wanted to make sure our high-frequency input signals would be accurately sampled (at the very least at the Nyquist rate) in case our oscillators ended up operating at a higher frequency than we intended. We finally settled on the AD7352 12-bit ADC as the best option for our project's requirements, as it takes in differential inputs and samples at 3 MSPS. Differential inputs add the benefits of significant common-mode voltage rejection and help to cut out any RF noise that could be introduced by the rest of our system, and the high sampling rate would be useful for providing better resolution when processing the digital signals.

2.4 Software

The Software subsystem consists of the implementation of SPI Protocol, FFT, and Audio Filters. The purpose of this subsystem is to read the digital output from the Sampling subsystem using the SPI protocol, interpret the frequency of these signals using an FFT, and then apply the necessary audio filters. The SPI protocol simultaneously reads input from the ADC for both control signals and outputs the values to the serial interface for the Teensy microprocessor. Then, the microprocessor utilizes these values to determine the current filter to be applied to the live audio. Appendix B contains the program code that was used to implement these functions.

2.4.1 SPI Protocol

The AD7352 transmits output data via SPI, and so it is necessary to implement SPI protocol to allow the Teensy Microprocessor to be able to read information from the ADC. This protocol reads 32 bits of information for each sample. In these 32 bits, two 12-bit values dictate the converted digital signals. The remaining 8 bits indicate the beginning of a new sample and separate the data for the two signals. To properly utilize SPI protocol, the software capitalizes on the Teensy SPI library to properly retrieve information from the ADC.

2.4.2 FFT Implementation

After receiving the digital signal from the ADC via SPI, the Teensy Microprocessor should use an FFT implementation to properly interpret the frequency of the input control signals. During the course of this project, we were unable to properly implement the FFT on the digital signals from the ADC because of a limitation with the Teensy FFT library. The FFT library only supports 512 bins with each representing 44.1 Hz, which means that the maximum frequency that can be sampled using this FFT library is 22.6 kHz. This introduces a problem, since our oscillator circuits produce signals with frequencies above 100 kHz. As a result, we researched alternative methods to implement the FFT. The first method is to develop our own FFT implementation without using a library. Another method involves heterodyning our control signals with a fixed oscillator signal and then passing these values through a lowpass filter to lower the overall frequency of the control signals. This method is actually very similar to the Moog Etherwave design that our oscillators are based on: the pitch frequency is the difference frequency between a variable and fixed oscillator, sent through a small detector circuit. Unfortunately, we were unable to implement either of these methods due to time constraints.

2.4.3 Audio Filters

The audio filters serve the purpose of providing mixing capabilities for the Theremixer. Theoretically, we would interpret the frequency of the input control signals properly using an FFT implementation and then use these to dictate the filter being applied. But, as a result of not getting the FFT implementation to work properly, we mimicked the input frequencies using a signal generator to determine the current filter being applied and the magnitude to which it is being applied. Essentially, the function select control signal allows a user to dictate the audio filter they would like to apply. These audio filters include capabilities to modify the bass, mid, treble, or volume of the output audio. On the other hand, the magnitude control signal allows the user to either amplify or attenuate the audio filter that is selected. We were able to successfully implement these audio filters.

2.4.3.1 Bass Boosting

Bass Boosting serves the purpose of amplifying or attenuating the bass in an output audio signal. This is implemented on the Teensy Microprocessor using a Low Pass Filter from the audio library at a cutoff frequency of 100 Hz. The magnitude signal dictates the extent to which the bass is amplified or attenuated.

2.4.3.2 Mid Boosting

Mid Boosting serves the purpose of amplifying or attenuating the midrange in an output audio signal. This is implemented on the Teensy Microprocessor using a Band Pass Filter from the audio library, using a cutoff frequency range of 400 to 600 Hz. The magnitude signal dictates the extent to which the midrange is amplified or attenuated.

2.4.3.3 Treble Boosting

Treble Boosting serves the purpose of amplifying or attenuating the treble in an output audio signal. This is implemented on the Teensy Microprocessor using a High Pass Filter from the audio library at a cutoff frequency of 800 Hz. The magnitude signal dictates the extent to which the treble is amplified or attenuated.

2.4.3.4 Volume Boosting

Volume Boosting serves the purpose of increasing or decreasing the output volume of the audio signal. This is implemented on the Teensy Microprocessor by altering the baseline volume by a scalar factor based on the magnitude signal.

3. Design Verification

3.1 Power

Each power module functioned as expected and was able to provide power to the necessary components within the proper tolerances. The + 12 V Regulator powered the oscillators and other voltage regulators with + 12.08 V. The – 12 V Regulator powered the oscillators with -12.07 V. The + 2.5 V Regulator powered the AD7352 ADC chip with + 2.6 V. The + 3.3 V Regulator powered the Teensy 3.2 microcontroller. This information is summarized in Table 1.

Table 1: Power Module Verification

	Required Value	Actual Value
+ 12 V Regulator	+ 12 V ± 5 %	+ 12.08 V
- 12 V Regulator	- 12 V ± 5 %	- 12.07 V
+ 2.5 V Regulator	+ 2.5 V ± 5 %	+ 2.6 V
+ 3.3 V Regulator	+ 3.3 V ± 5 %	+ 3.33 V

3.2 Hardware

3.2.1 Antenna Circuits

Each antenna circuit functioned as expected and were affected by hand placement near the antenna. To test the Antenna circuits, we used an LCR Meter to measure the effective capacitance of each antenna. The positive lead was connected to the bottom of the antenna through a wire and the negative lead was held in the tester's hand. By holding the negative lead, the tester becomes the ground plane. Using a ruler, the tester placed his hand 20 cm away from the antenna. After 10 trials, at 20 cm the Magnitude Oscillator Antenna had an average capacitance of 2.31 pF. The Variable Oscillator Antenna had an average capacitance of 5.52 pF 20 cm away. Shown in Figure 15 and Table 2, both antennas had an increased effective capacitance as the tester's hand approached the respective antenna. After 10 trials, at 1 cm away the Magnitude Oscillator Antenna had an effective capacitance of 5.33 pF. The Variable Oscillator Antenna had an effective capacitance of 9.06 pF at 1 cm.

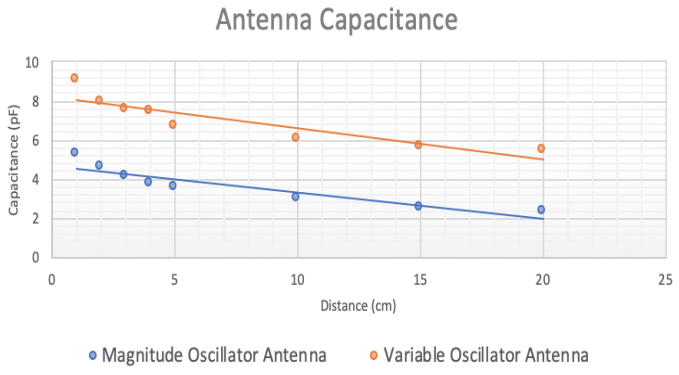

Figure 15: Antenna Effective Capacitance vs. Distance of Hand (graph)

Table 2 : Antenna Effective Capacitance vs. Distance of Hand (values)

Distance (cm)	0	1	2	3	4	5	10	15	20
Magnitude Oscillator Antenna Capacitance (pF)	87.17	5.33	4.59	4.14	3.82	3.6	3.01	2.59	2.31
Variable Oscillator Antenna Capacitance (pF)	120.01	9.06	7.93	7.62	7.48	6.75	6.07	5.67	5.52

3.2.2 Oscillator Circuits

Each oscillator circuit created a very stable and reliable frequency with low standard deviations. For the Magnitude Detection Oscillator Antenna, we met the desired oscillation frequency requirement of 142 kHz ±10% at an average of 141.76 kHz. The standard deviation of this frequency was 354.91 Hz. The average peak-to-peak value was 761.29 mV. For the Function Select Antenna Circuit, we met the desired oscillation frequency requirement of 430 kHz ±10% at 431.31 kHz. The standard deviation of this frequency as 727.70 Hz. The average peak-to-peak value was 3.51 V. To verify these frequencies, we used an oscilloscope to probe the base of the second transistor of each oscillator across ground. Shown in Figure 16 is the Magnitude Detection Oscillator scope with statistics over 1.006 thousand samples. Shown in Figure 17 is the Function Select Oscillator scope with statistics over 1.522 thousand samples.

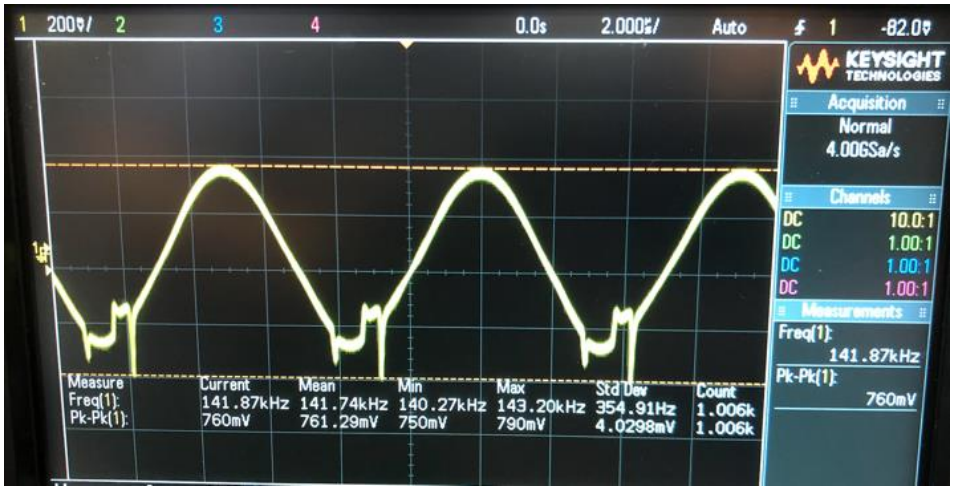

Figure 16: Verification of Magnitude Detection Oscillator Circuit

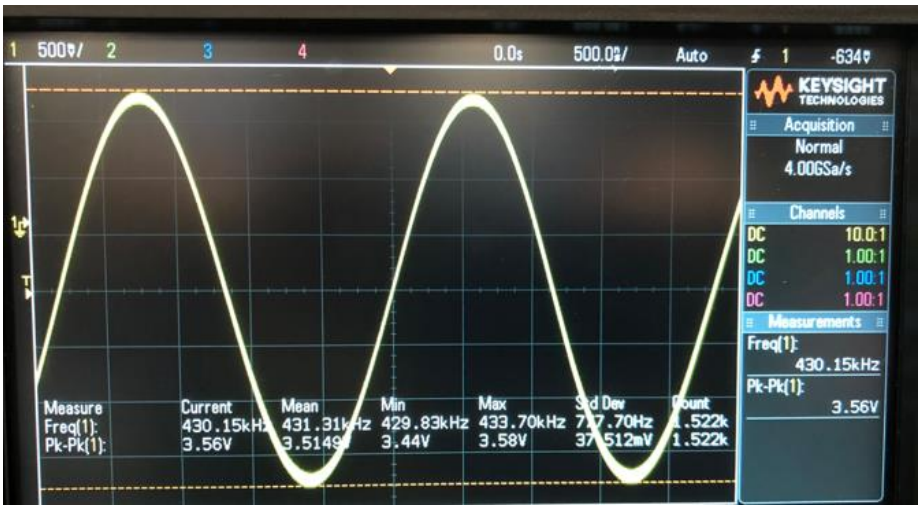

Figure 17: Verification of Function Select Oscillator Circuit

3.2.3 Integration

Although each antenna circuit functioned as expected and each oscillator circuit outputted a very stable waveform at the desired frequency, we were unable to achieve full integration of these two parts. An increase of capacitance in the antenna should result in an increase in frequency of the oscillation, however we were unable to visualize this on the oscilloscope. We may not have been able to visualize the change on the oscilloscope since the intended frequency change for the oscillators were less than 1 kHz.

Specifically, we expected a frequency shift of 500 Hz for the magnitude detection oscillator; and we expected a frequency shift of 300 Hz for the function select oscillator. These small fluctuations may be the reason we could not visualize the change on the oscilloscope. One way we could potentially visualize these frequency changes is through proper FFT windowing on the software end through the Teensy 3.2 microcontroller. By using a larger FFT window, we theorize that it would be possible to see these subtle changes because they will be occurring over a longer period.

3.3 Sampling

Using our base PCB design, the buffer circuits did not function as intended. However, we quickly realized that we had based our PCB design on the wrong schematic in the AD7352 datasheet, using Figure 21 rather than Figure 22 in the datasheet. Thankfully, the two layouts were very similar and only differed in a grounded resistor and a few changed component values. We were able to correct them using wires and through hole resistors, and after making these changes, we were able to verify that each buffer output was both unipolar and 180 degrees out of phase with each other as the ADC required. After verifying the buffer outputs and completing the SPI submodule, we connected the Teensy 3.2 Microprocessor to the rest of the PCB and were able to simultaneously view both outputs as changing integer values. However, we wanted to further verify that the sampling subsystem was working since due to the high frequency nature of our input signals it was difficult to know whether the oscillators were being properly sampled. To do this, we replaced the inputs with very low frequency (0.5-1 Hz) sine waves from a function generator. By doing this we were able to see the output integer values oscillate very reliably on the Teensy. The final PCB is pictured in Figure 18 in Appendix B, and the original board layout is shown in Figure 19.

3.4 Software

In the Software subsystem, there are many components that were successfully implemented. In fact, all the requirements that we specified in the R & V Table in Appendix A for the Teensy Microcontroller were fulfilled. To summarize the results, we were able to properly implement all the audio filters using a control signal that mimicked the signal from the function select oscillator, and then properly amplify or attenuate this audio filter using a control signal that mimicked the signal from the magnitude oscillator. Figure 20 in Appendix B indicates a frequency analysis of a snippet of audio from the Teensy Microcontroller. For this same snippet of audio, we applied different filters separately to determine if they were properly being selected and applied. As noted in Figure 21, when Bass Boosting is chosen, a low pass filter is applied to the audio output which eliminates frequencies above 100 Hz. It also indicates that the frequencies under 100 Hz are amplified. Also, in Figure 22, it is noticeable that when Mid Boosting is chosen, a band pass filter is applied to the audio output which eliminates frequencies below 400 Hz and above 600 Hz. It also expresses that frequencies between 400 Hz to 600 Hz are amplified. Additionally, Figure 23 indicates that when Treble Boosting is chosen, a high pass filter is applied to the audio output which eliminates frequencies below 800 Hz. It also shows that frequencies above 800 Hz are amplified. Lastly, Figure 23 indicates that when Volume Boosting is chosen and the value of the magnitude control signal changes, the baseline volume either increases or decreases. This is noticeable in decibel level changes in Figure 23 that were monitored by increasing the magnitude by 100 Hz every second. Overall, all the audio filters properly functioned because they were able to modify live audio output using the signals that were mimicked with the signal generator. Although we were able to read the digital output of the ADC through an SPI protocol, we were unable to properly compute the FFT of the digital signal. This is primarily because we were constrained for time in implementing alternative methods that were outlined in section 2.4. If alternative methods were used to implement the FFT, we would have been able to properly integrate the Theremixer.

4. Costs & Schedule

4.1 Parts

The cost of parts for this project is in Table 3.

Table 3 : Costs of Parts

Part Name	Part Number	Manufacturer	Unit Cost ($)	Quantity	Total Cost ($)
AC Transformer	BPD2EE	Hammond Manufacturing	17.58	1	17.58
Teensy 3.2	Teensy 3.2	PJRC	29.25	1	29.25
NPN Transistor	2N3904	ON Semiconductor	0.07	8	0.56
+ 12 V Regulator	LM78L12	Texas Instruments	0.79	1	0.79
- 12 V Regulator	LM79L12	Texas Instruments	0.79	1	0.79
+ 2.5 V Regulator	LP2985	Texas Instruments	0.88	1	0.88
+ 3.3 V Regulator	LP2950	Texas Instruments	0.96	1	0.96
Power Supply Diode	1N4001	ComChip Technology	0.17	2	0.34
Detector / Oscillator Diode	1N4148	ON Semiconductor	0.10	2	0.20
AD8022 Dual Op-Amp Chip	AD8022	Analog Devices	6.18	3	18.54
AD7352 12-bit ADC	AD7352	Analog Devices	24.40	1	24.40
10 mH Inductor	74404086103	Wurth Electronics, Inc	0.65	4	1.30
47 uH Inductor	74404065233	Wurth Electronics, Inc	0.67	5	1.34
5 mH inductor	74404023433	Wurth Electronics, Inc	0.55	2	1.10
2.5 mH Inductors	74404076811	Wurth Electronics, Inc	0.52	2	1.04
3300 pF Capacitor	GRM2165C1H332JA01D	Murate Electronics	0.87	2	1.74
15 pF Capacitor	CL05C150JB5NNNC	Samsung Electro Mechanics	0.35	2	0.70

1000 pF ceramic Capacitor	GRM033R71E102KA01D	Murata Electronics	0.72	2	1.44
22 uF tantalum Capacitor	GRM027R1E102KA01D	Murata Electronics	0.76	1	1.52
1800 pF film Capacitor	GRM076F76E00KA01D	Murata Electronics	0.80	1	1.60
2200 uF Aluminum Electrolytic Capacitor	GRM897T6F2155LA01D	Murata Electronics	0.56	2	1.02
0.1 uF Ceramic Capacitor	GRM762F8T200MA01D	Murata Electronics	0.42	2	0.84
Copper Plated Tubing for Antennas	1-36A-C	BrassCraft	7.05	2	14.10
Printed Circuit Board		PCBWay	50.00	1	50.00
Total					172.03

4.2 Labor

The cost of labor for this project is as follows:

$$Team\ Members\ (3) * Hourly\ Rate\ (\$50) * Hours\ per\ Week\ (12) * Total\ Weeks\ (16) * 2.5 = \$\mathbf{72,000}$$

4.3 Schedule

The schedule our team followed during the course of this project is in Table 4.

Table 4: Weekly Schedule

Week	Akhil	Karthik	Shiv
01/14/19 - 02/04/19	• Conduct research for project proposal • Create project proposal	• Conduct research for project proposal • Create project proposal	• Conduct research for project proposal • Create project proposal
02/11/19	• Work on design document • Complete eagle assignment	• Work on design document • Complete eagle assignment	• Work on design document • Complete eagle assignment
02/18/19	• Complete Design Document • Order Parts	• Complete Design Document • Order Parts	• Complete Design Document • Order Parts
02/25/19	• Build filter prototype on Teensy microcontroller	• Complete PCB Design • Build Magnitude Detection	• Complete PCB Design • Build Function Select Antenna

		• Conduct Antenna Impedance Testing	• Conduct Antenna Impedance Testing
03/04/19	• Complete Soldering Assignment • Complete Teamwork Evaluation I • Build software prototype to utilize multiple inputs for choosing filters	• Complete Soldering Assignment • Antenna Oscillator Testing & Calibration • Complete Teamwork Evaluation I	• Complete Soldering Assignment • Antenna Oscillator Testing & Calibration • Complete Teamwork Evaluation I
03/11/19	• Test and calibrate filters using simulated inputs	• Construct housing design and submit to ECE shop • Connect and test Theremixer hardware subsystem	• Construct housing design and submit to ECE shop
03/18/19	**SPRING BREAK**		
03/25/19	• Connect and test Theremixer hardware subsystem • Submit individual progress reports	• Connect and test Theremixer hardware subsystem • Submit individual progress reports	• Connect and test Theremixer hardware subsystem • Submit individual progress reports
04/01/19	• Test and interface all hardware and software components	• Test and interface all hardware and software components	• Test and interface all hardware and software components
04/08/19	• Develop project demonstration • Further verification and calibration	• Develop project demonstration • Further verification and calibration	• Develop project demonstration • Further verification and calibration
04/15/19	• Develop project demonstration • Further verification and calibration	• Develop project demonstration • Further verification and calibration	• Develop project demonstration • Further verification and calibration
04/22/19	• Work on final paper and presentation	• Work on final paper and presentation	• Work on final paper and presentation
04/29/19	• Finalize and submit the final paper, presentation, lab notebook, lab checkout, and Teamwork Evaluation II	• Finalize and submit the final paper, presentation, lab notebook, lab checkout, and Teamwork Evaluation II	• Finalize and submit the final paper, presentation, lab notebook, lab checkout, and Teamwork Evaluation II

5 Conclusion

Senior design proved to live up to its reputation and challenged each of us a great deal. We grew both our interpersonal and professional skills in working on this project. We did not manage to achieve full integration of each module in our project; however, we were immensely proud of the work we accomplished. We managed to create functioning power sources, oscillators, antennas, buffer circuits, and filters. We managed to properly sample our outputs and communicate to our microcontroller. We struggled on integration of the antennas with their respective oscillators due to much uncertainty. We also had set backs integrating the sampled output with our microcontroller filters. At this stage, there are still many improvements that we can make the Theremixer more functional and user-friendly.

5.1 Accomplishments

Although we lacked in fulfilling full integration, we succeeded in developing individual components. We created functioning power supplies that properly operated to power each respective component. In the hardware subsystem, we met the requirements for creating both working antennae and created two functioning oscillators. We struggled in proving the integration of each oscillator with their respective antenna. Our ADC properly sampled the output from the oscillators and the Teensy 3.2 microcontroller was able to read this input via SPI communication. The Teensy 3.2 microcontroller was also able to properly apply each audio filter using mimicked control signals.

5.2 Uncertainties

One of our main shortcomings in completing our project involved implementing the FFT subsystem to determine the frequency of our oscillators, since the Teensy 3.2's native FFT algorithm only provides frequency data up to 22.6 kHz which is far below our signal frequencies. Professor Oelze suggested that we heterodyne our signals with other sine waves close to the frequency of the input signals to create a low frequency signal that can be read by the Teensy's onboard ADC, and after considering this further we realized that the Etherwave design uses a very similar method to extract its pitch frequency, even using a capacitor in parallel to apply a low-pass filter to the modulated signal. There was also significant uncertainty with our antenna operation, specifically the effect of changing antenna capacitance on the oscillators' frequencies. We are still unsure whether or not our current design actually has a measurable difference in frequency within a reasonable range of motion, but we theorized that this might be partly or even fully fixed with a better FFT implementation.

5.3 Future work

Improvements to hardware can enhance the overall experience of using the Theremixer. Firstly, we recommend enhancing the effect of the antenna capacitance on oscillator frequency. This change would allow us to notice a larger difference on the oscillator frequency when a user's hand changes placement. Additionally, it would be ideal to redesign the Printed Circuit Board (PCB) to reflect design changes throughout the semester. Specifically, many changes were made to the PCB throughout the semester using through hole components that were soldered directly onto the PCB. Rather, it would be more appropriate to redesign the PCB to accommodate for the changes in the buffer circuits and the ground loops in the mixed signal design. Also, several changes on the software module can help enhance the Theremixer's capabilities. It is vital to implement Fast Fourier Transform (FFT) on the input signals from the ADC to measure the frequencies of both control signals. This will help complete the integration in this project so that the change in hand placement will apply a filter to the audio being played. Some additional features can also improve the user experience. These features include the integration of a potentiometer for selecting the cut-off frequency for different filters, utilizing buttons for functions such as reset, pause, play, and skip, creating a visual display of current filters, and introducing additional mixing effects. All these implementations can help make the Theremixer a truly unique product.

5.4 Ethical considerations

There are several ethical and safety matters to consider throughout the development of our project. Firstly, the IEEE Code of Ethics #5 states that the goal is "to improve the understanding by individuals and society of the capabilities and societal implications of conventional and emerging technologies, including intelligent systems" [1]. Our project encompasses replicating a traditional Theremin design to develop an alternative operation of mixing music using output from the Theremin. This aligns closely with the IEEE Code of Ethics #5 because we want to help individuals understand alternative uses of traditional technologies in the modern day. We also hope to accept honest criticism of our work throughout the semester to help guide our project and make fixes where necessary in accordance with IEEE of Ethics #7[1]. Another critical foundation of our project is to work closely with each other to develop professionally, and always uphold the IEEE Code of Ethics. The Code of Ethics is not only a statement of what to do on paper, but also the framework for an engineer's mindset to always be solution oriented while upholding certain principles.

The Theremin we hope to construct would be a modern replication of the traditional Theremin. We are going to be building a Theremin that is an adaption of an Etherwave Theremin that has been developed my Moog Music previously [2]. Fortunately, Moog Music has published various open-source information to help guide the design of the Theremin [3]. We hope to use this design as a foundation for further developing a modern Theremin and provide an alternative use with music mixing. Additionally, another ethical point to concern is developing a platform to alter music. Although the music is the property of artists, we will follow similar practices as those DJs in today's day and age who mix music at clubs. We hope to access the music we play legally through a streaming service or through purchase of music albums. Overall, our project does not pose any significant ethical concerns that might impact its success. This is primarily because we are taking a traditional design and making a modern use of it.

6 References

[1] IEEE, 'IEEE Code of Ethics', 2014. [Online.] Available: https://www.ieee.org/about/corporate/governance/p7-8.html. [Accessed: 01 - Feb - 2019].

[2] Robert Moog, Electronic Musician, 'Build the EM Theremin', 1996. [Online.] Available: https://www.cs.nmsu.edu/~rth/EMTheremin.pdf. [Accessed: 01 - Feb - 2019].

[3] Moog Music Inc, 'Understanding, Customizing, and Hot-Rodding Your Etherwave Theremin', 2003. [Online.] Available: http://www.suonoelettronico.com/downloads/HotRodEtherwav.pdf. [Accessed: 01 - Feb - 2019].

Appendix A: Requirement and Verification Table

Table 5: System Requirements and Verifications

Requirement	Verification	Verification status (Y or N)
1. 14 Volt AC Transformer a. Provide 14V AC ±5% at 60 Hz to voltage rectifier.	1. Use specification sheet to confirm rating. a. Measure output of transformer (input of voltage rectifier) using oscilloscope while connected to power supply circuit. b. Verify from the waveform that the transformer is providing 14V ±5% AC at 60 Hz.	Y
2. ± 12 Voltage Regulator a. Provide ±12V ±5% to the positive and negative power rails. b. Provide ±12V ±5% to power the AD8022 buffer op amps.	2. Measure the output voltage at the positive rail using a multimeter. a. Verify that the positive rail provides +12V ±5%. b. Measure the output voltage at the negative rail using a multimeter. c. Verify that the negative rail provides -12V ±5%. d. Verify that each AD8022 opamp is receiving ±12V ±5% to properly power on.	Y
3. + 3.3 Voltage Regulator a. Provide +3.3V ±5% to the Teensy Microprocessor	3. Measure the output voltage of the voltage regulator using a multimeter a. Verify that the teensy is receiving +3.3V to properly power on.	Y
4. + 2.5 Voltage Regulator a. Provide +2.5 V ±5% to power the AD7352 chip.	4. Measure the output voltage of the voltage regulator using a multimeter a. Verify that the AD7352 chip is receiving +2.5 V ±5%.	Y
5. Magnitude Detection Antenna a. Antenna effective capacitance of 1-8 pF ±10% b. Capacitance increase of 5-7pF ±10% when a hand approaches the antenna.	5. Measure the effective capacitance of the antenna using an LCR Meter, while your hand is far from the antenna. a. Verify that the total capacitance is in the range of 1-3 pF ±10%. b. Measure the effective capacitance of the antenna	Y

	using an LCR Meter, while your hand is near the antenna. c. Verify that the total capacitance has increased by 5-7 pF ±10%.	
6. Magnitude Detection Oscillator a. Maintain a baseline frequency of 142 kHz ±10%. b. Be responsive to hand movements. i. When a hand approaches the antenna the oscillator's frequency should increase by 500 Hz. ii. When a hand is away from the antenna the variable frequency should maintain the baseline frequency 142 kHz ±10%	6. Using an oscilloscope, measure the frequency of the waveform when a hand is far from the antenna. a. Verify that the frequency is near 142 kHz ±10%. b. Using an oscilloscope, measure the frequency of the waveform when a hand is near the antenna. c. Verify that the frequency is near 142.5 kHz ±10%.	N
7. Function Select Antenna a. Antenna effective capacitance of 1-8 pF ±10% . b. Capacitance increase of 5-7pF ±10% when a hand approaches the antenna.	7. Measure the effective capacitance of the antenna using an LCR Meter, while your hand is far from the antenna. a. Verify that the total capacitance is in the range of 1-3 pF ±10%. b. Measure the effective capacitance of the antenna using an LCR Meter, while your hand is near the antenna. c. Verify that the total capacitance has increased by 5-7 pF ±10%.	Y
8. Function Select Oscillator a. Maintain a baseline frequency of 430 kHz ±10%. b. Be responsive to hand movements.	8. Using an oscilloscope, measure the frequency of the waveform when a hand is far from the antenna. a. Verify that the frequency is near 430 kHz ±10%.	N

i. When a hand approaches the antenna the oscillator's frequency should increase by 300 Hz. ii. When a hand is away from the antenna the variable frequency should maintain the baseline frequency 430 kHz ±10%	b. Using an oscilloscope, measure the frequency of the waveform when a hand is near the antenna. c. Verify that the frequency is near 430.3 kHz ±10%.	
9. Analog to Digital Converter with Input Buffers (ADC) a. From two bipolar signal inputs from oscillator circuits, create two differential unipolar input pairs to feed into ADC using opamp buffers b. Properly sample both oscillator signals at least the Nyquist rate of the highest frequency signal (860 kHz)	9. While feeding oscillator signals into PCB, probe all four buffer opamp outputs. a. Verify that all outputs fed into the ADC match input signal frequency and that voltage values remain within a range of 0<V<2.048. b. Verify from the specification sheet that the AD7352 has a maximum sampling frequency of 3 MSPS. This is much higher than the required sample rate of 860 kSPS. c. Check Teensy Serial Monitor output to verify that oscillators are being sampled through SPI input	N
10. Teensy Microcontroller – Bass Boosting (Low Pass Filter) a. Select Low Pass Filter for frequency range of 50 to 300 Hz of the input control signal b. Amplify frequencies of 0-100 Hz and attenuate frequencies above 100 Hz c. Utilize magnitude control signal from 0 to 1200 Hz to amplify or attenuate altered signal	10. Load test audio file a. Use spectrogram analysis program to verify that frequencies from 0 to 100 Hz are amplified and frequencies above 100 Hz are attenuated.	Y

11. Teensy Microcontroller - Treble Boosting (Band Pass Filter) a. Select Band Pass Filter for frequency range of 301 to 600 Hz of the input control signal b. Amplify frequencies of 400 to 600 Hz, attenuate frequencies outside of this range c. Utilize magnitude control signal from 0 to 1200 Hz to amplify or attenuate altered signal	11. Load test audio file a. Use spectrogram analysis program to verify that frequencies from 400 to 600 Hz are amplified and frequencies outside this range are attenuated.	Y
12. Teensy Microcontroller - Mid Boosting (High Pass Filter) a. Select High Pass Filter for frequency range of 601 to 900 Hz of the input control signal b. Amplify frequencies above 500 Hz, attenuate frequencies below this frequency c. Utilize magnitude control signal from 0 to 1200 Hz to amplify or attenuate altered signal	12. Load test audio file a. Use spectrogram analysis program to verify that frequencies above 500 Hz are amplified and frequencies below 500 Hz are attenuated.	Y
13. Teensy Microcontroller - Volume Boosting a. Apply a 5-dB increase to the baseline volume by increasing the magnitude from 800 Hz to 900 Hz b. Apply a 3-dB decrease to the baseline volume by decreasing the magnitude to 800 Hz to 700Hz	13. Utilize Visual Audio program to determine the dB increase as you increase the magnitude control signal from 800 Hz to 900 Hz a. Utilize Visual Audio program to determine the dB decrease as you decrease the magnitude control signal from 800 Hz to 700 Hz	Y

Appendix B: Circuit Schematics, Graphs, and Data

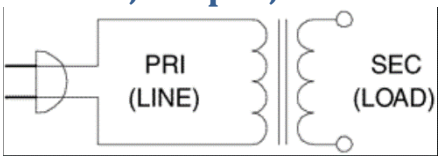

Figure 3: 14 V AC Transformer

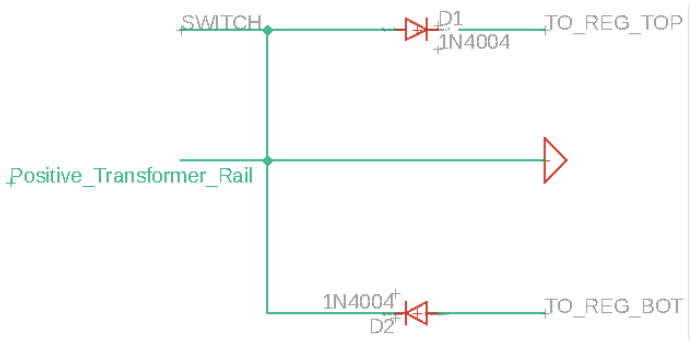

Figure 4: Center-Tapped Full-Wave Voltage Rectifier

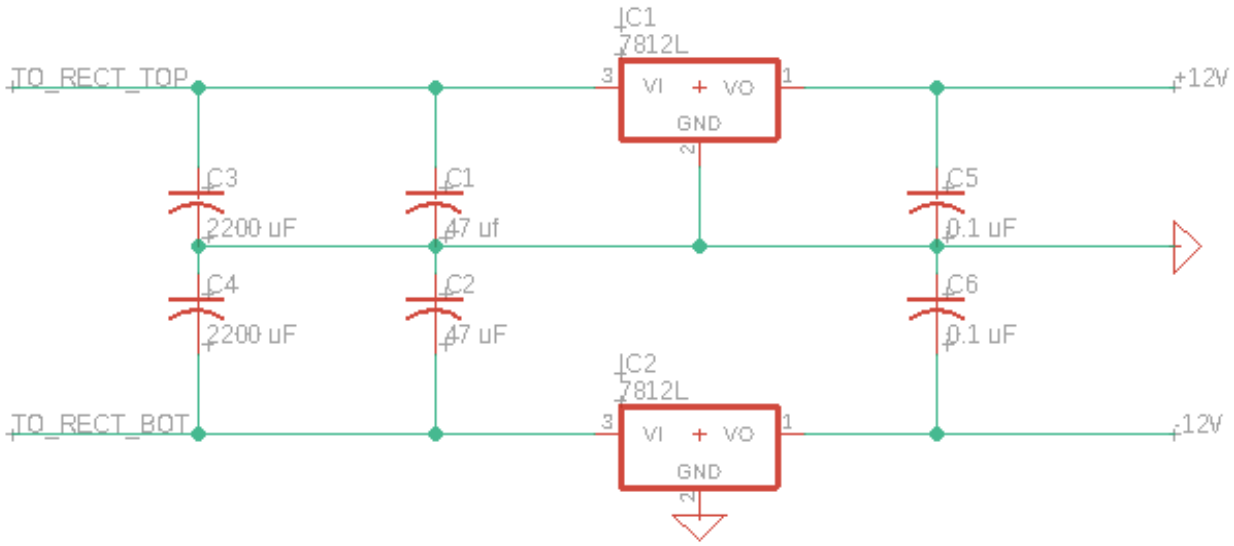

Figure 5: ± 12 Voltage Regulator Circuit

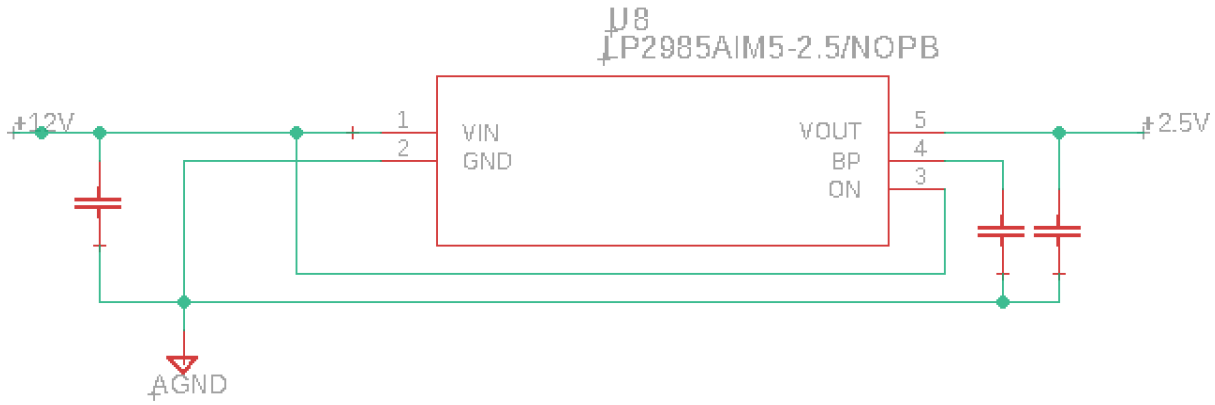

Figure 6: +2.5 Voltage Regulator

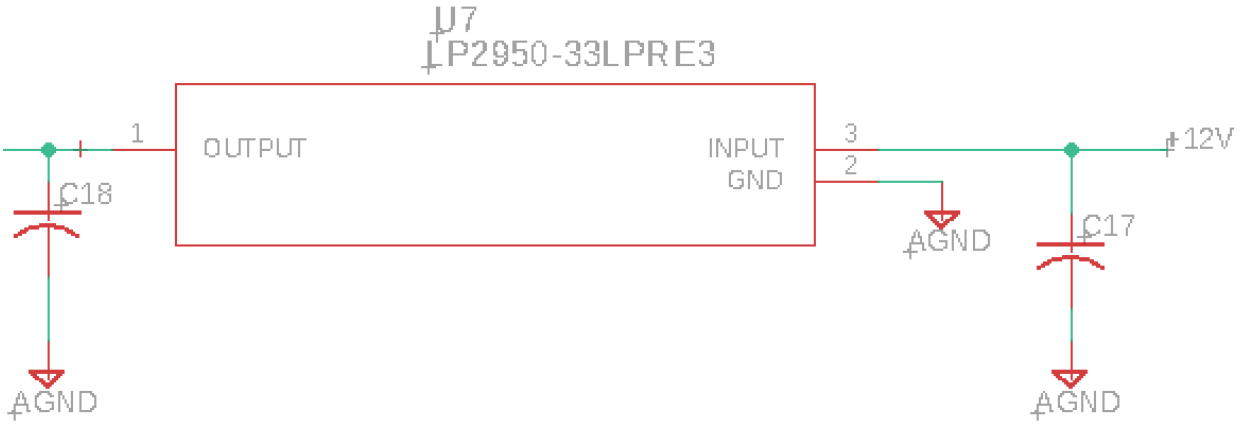

Figure 7: +3.3 Voltage Regulator

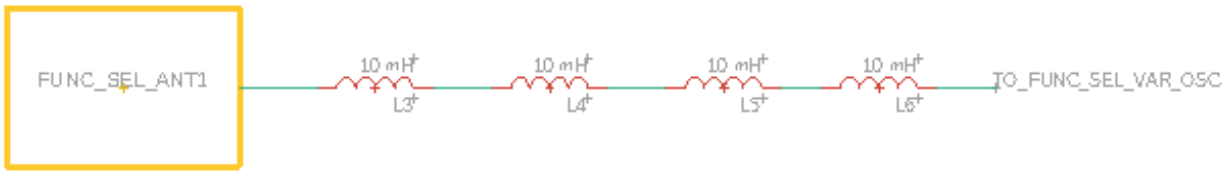

Figure 8: Function Select Antenna Circuit

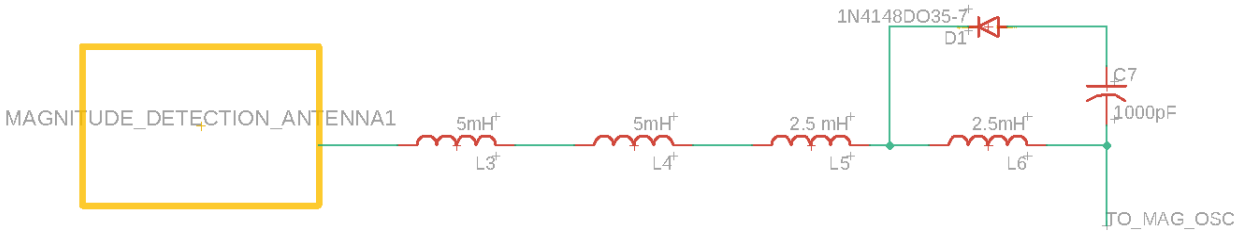

Figure 9: Magnitude Detection Oscillator Circuit

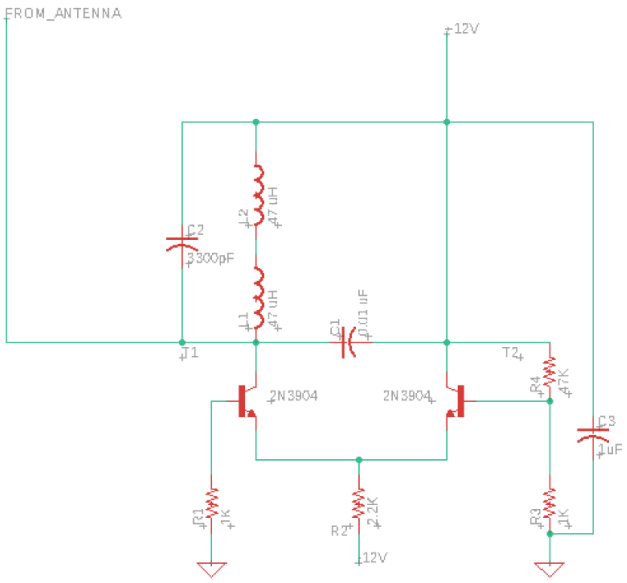

Figure 10: Function Select Oscillator Circuit

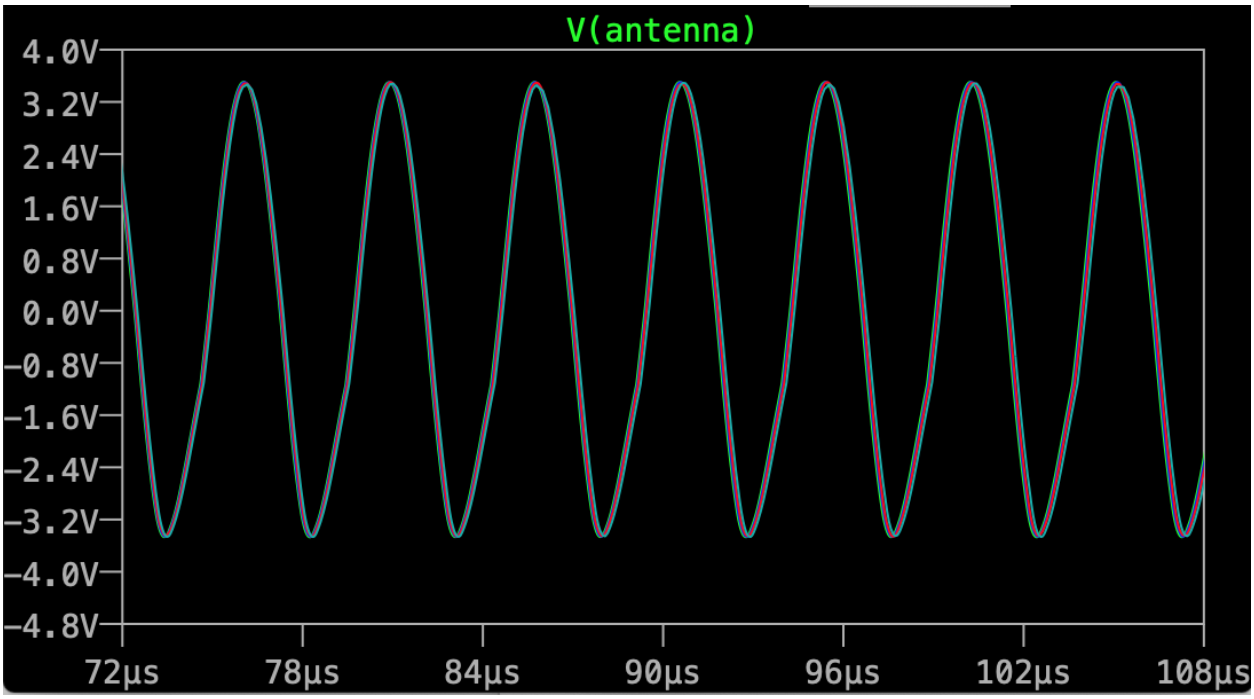

Figure 11: Simulation of Function Select Oscillator Circuit

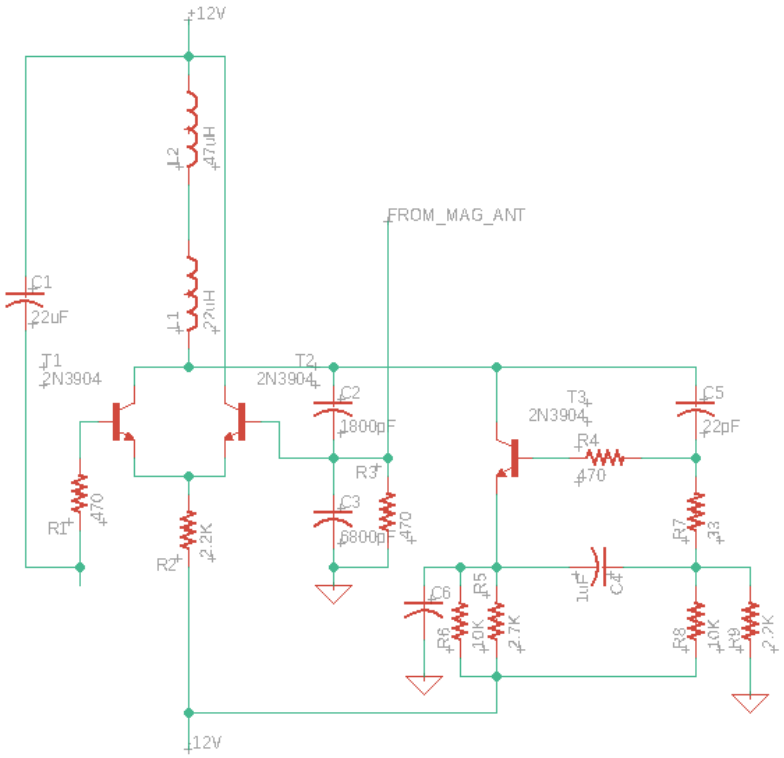

Figure 12: Magnitude Detection Oscillator Circuit

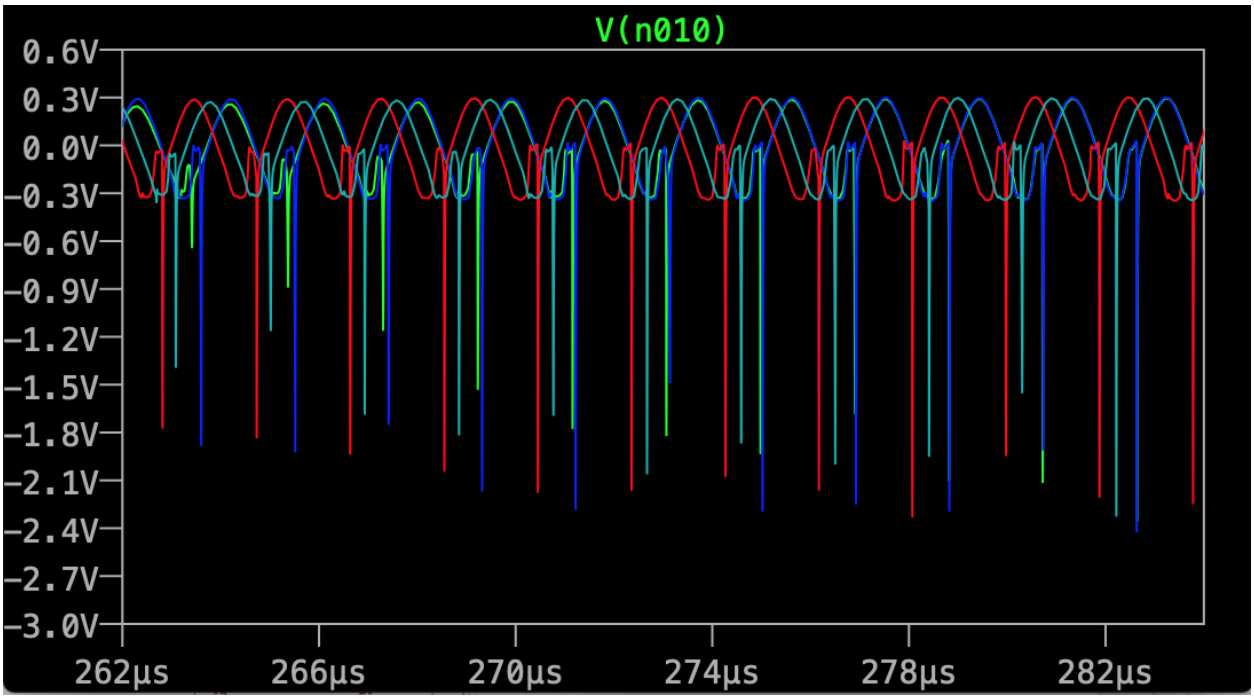

Figure 13: Simulation of Magnitude Detection Oscillator Circuit

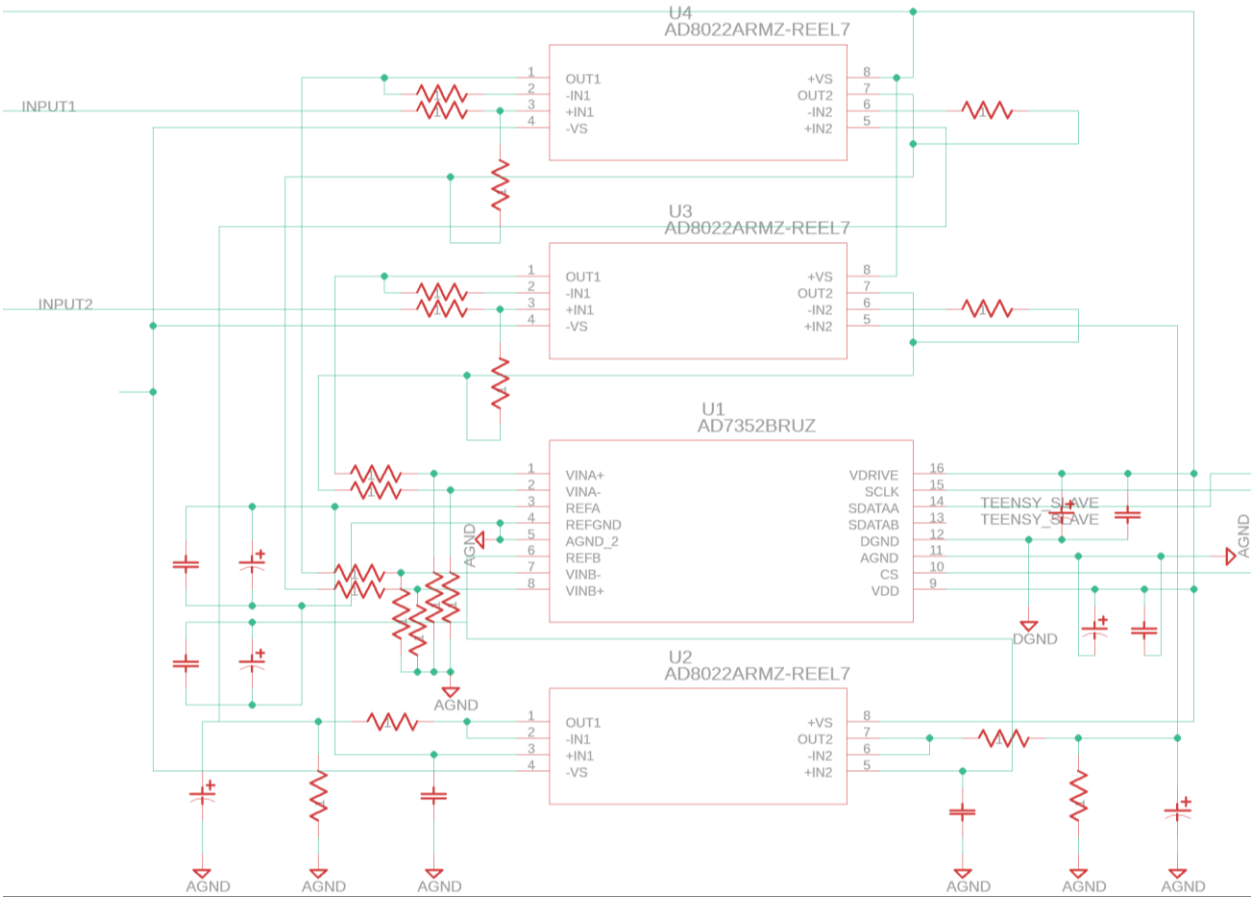

Figure 14: Sampling Module Schematic

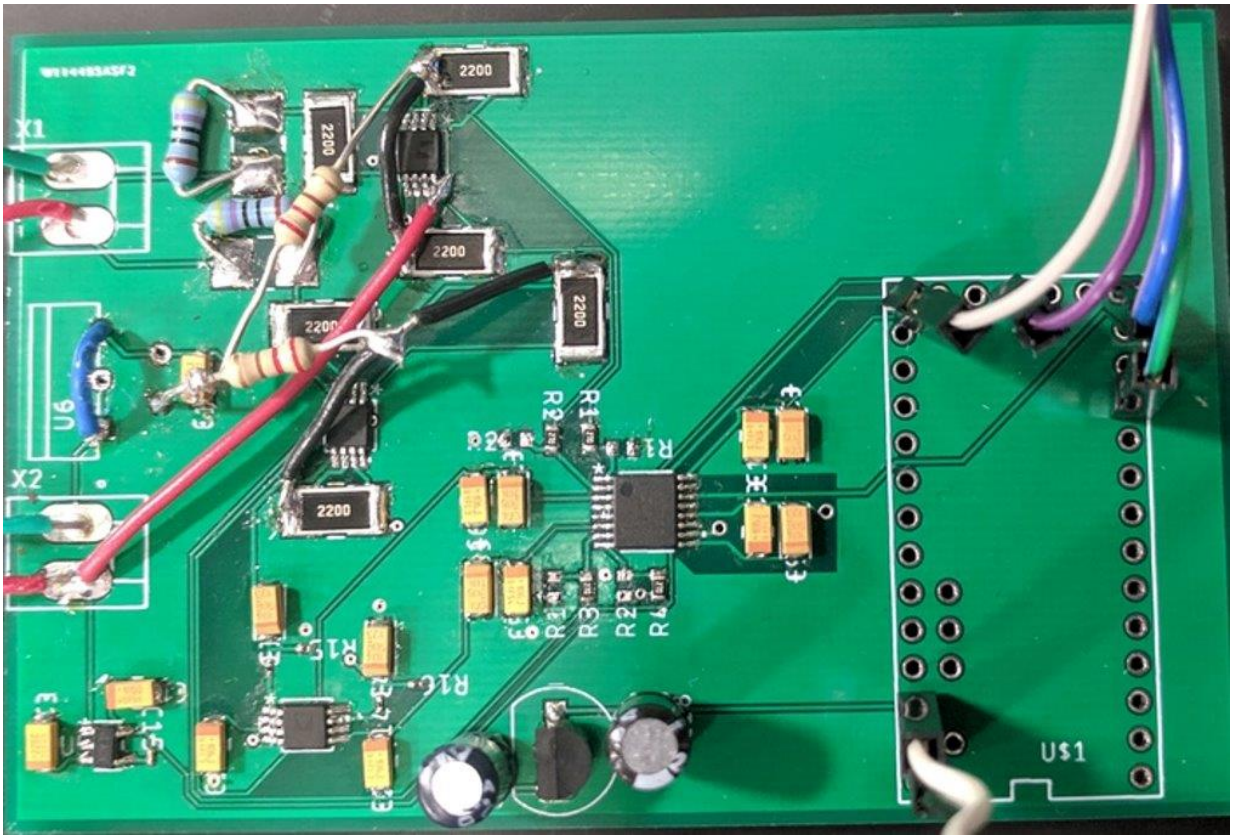

Figure 18: Printed Circuit Board (PCB)

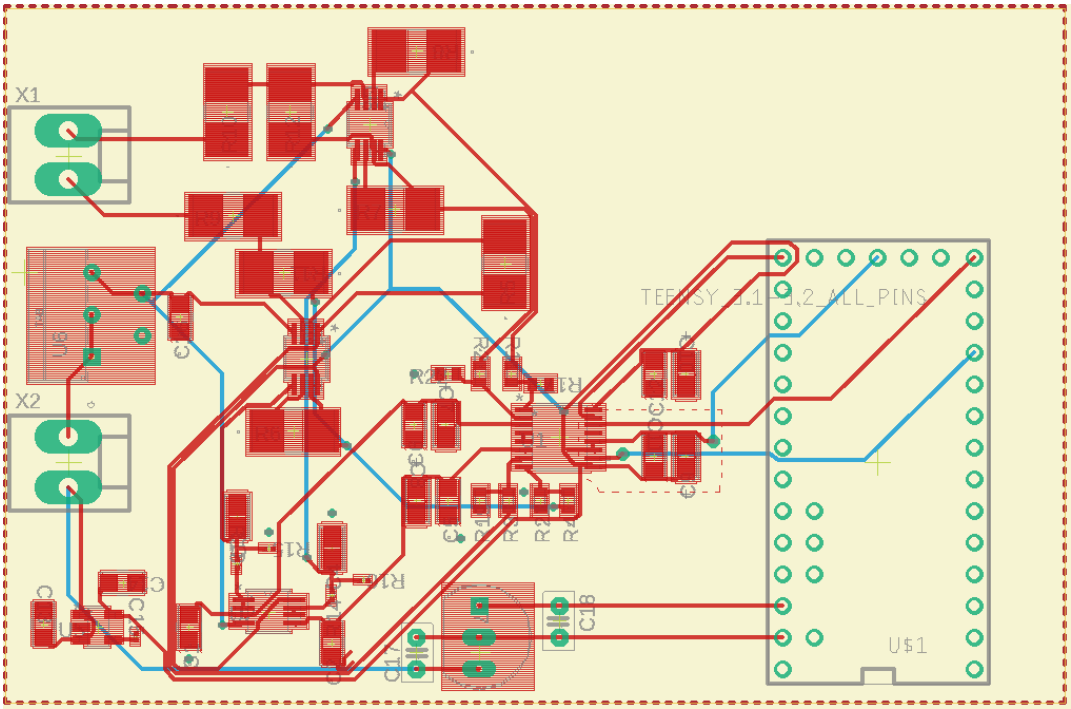

Figure 19: Board Layout for PCB

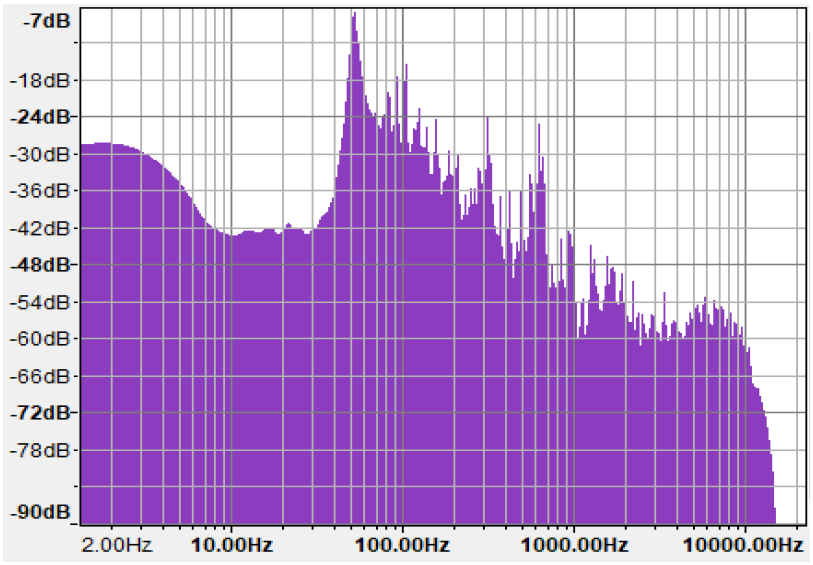

Figure 20: Original Audio Clipping

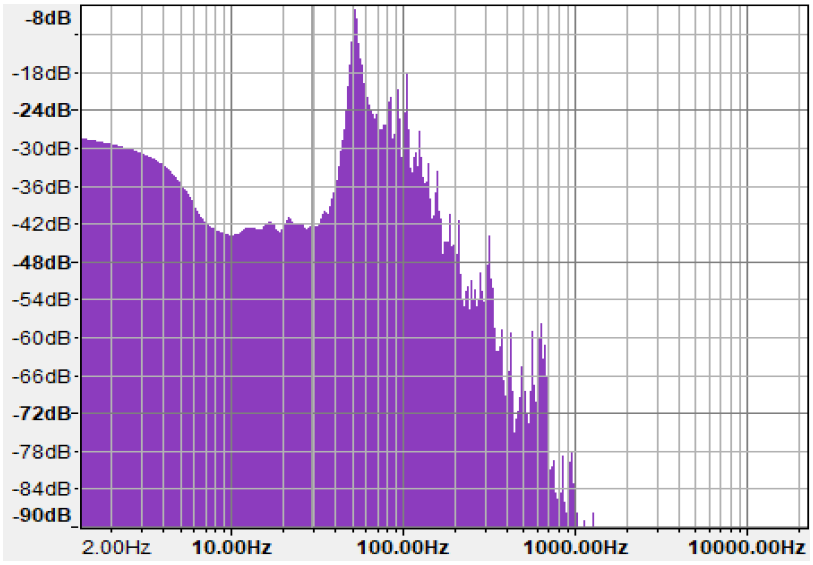

Figure 21: Bass Boosting - Low Pass Filter

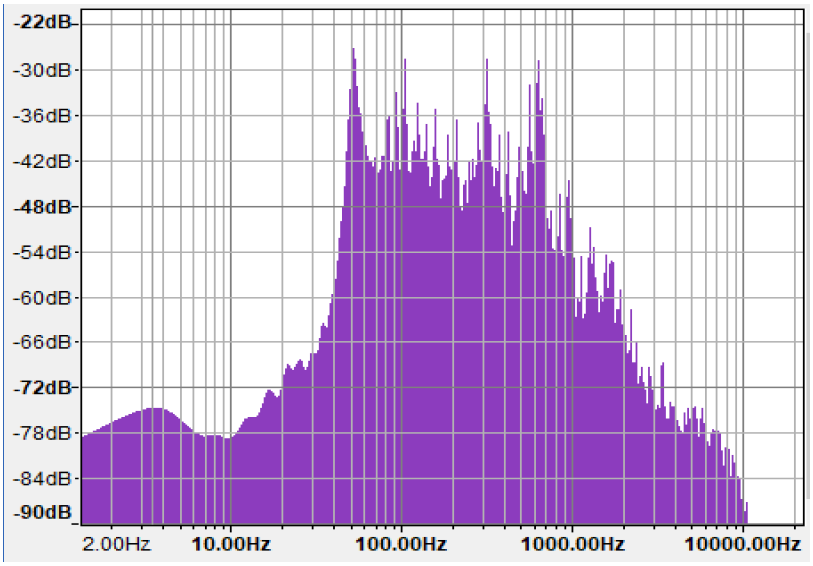

Figure 22: Mid Boosting – Band Pass Filter

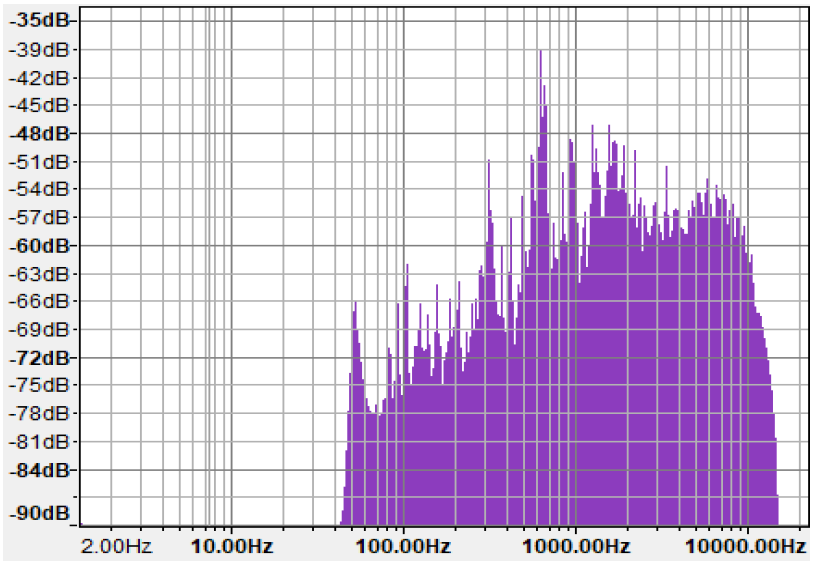

Figure 23: Treble Boosting – High Pass Filter

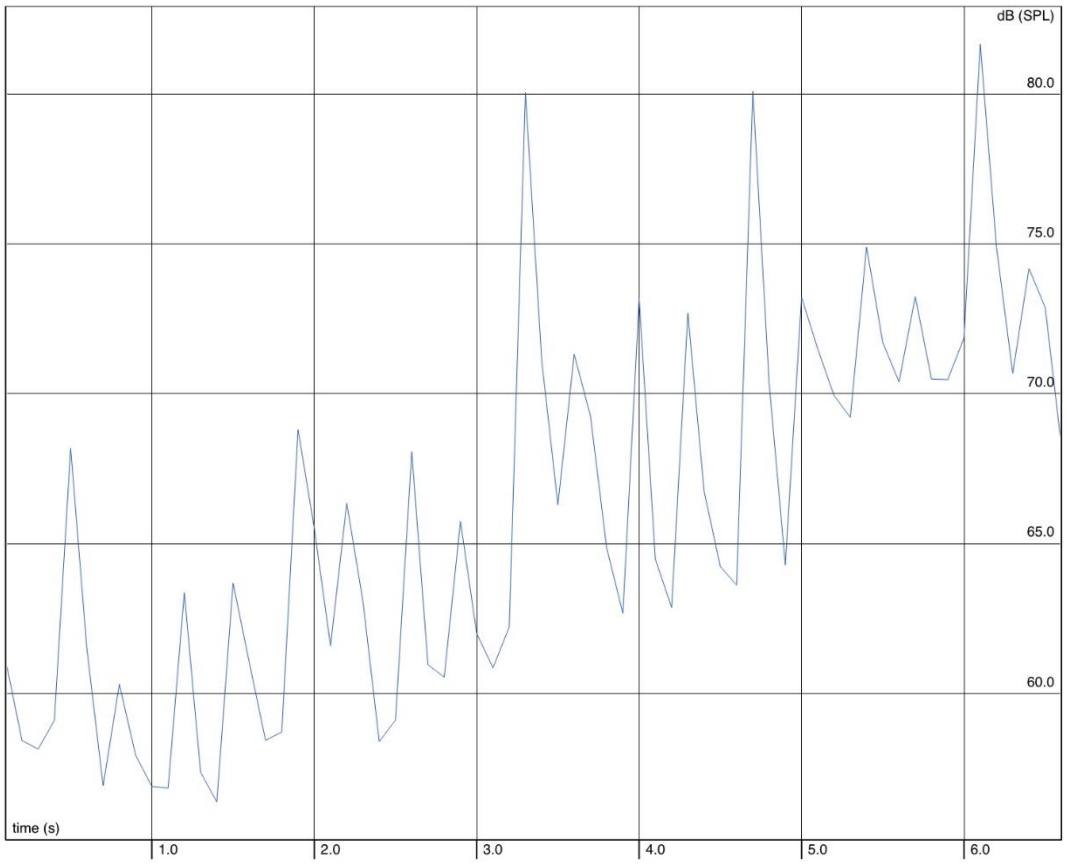

Figure 24: Volume Boosting

Appendix C: Software Program Code

Audio Filter Program

```
#include <Audio.h>
#include <Wire.h>
#include <SPI.h>
#include <SD.h>
#include <SerialFlash.h>
#include <Bounce.h>

const int myInput = AUDIO_INPUT_LINEIN;

AudioPlaySdWav          playSdWav1;     //xy=369,162
AudioFilterStateVariable filter2;       //xy=555,232
AudioFilterStateVariable filter1;       //xy=557,141
AudioMixer4             mixer1;         //xy=750,143
AudioMixer4             mixer2;         //xy=753,235
AudioOutputI2S          i2s1;           //xy=937,167
AudioConnection         patchCord1(playSdWav1, 0, filter1, 0);
AudioConnection         patchCord2(playSdWav1, 1, filter2, 0);
AudioConnection         patchCord3(filter2, 0, mixer2, 0);
AudioConnection         patchCord4(filter2, 1, mixer2, 1);
AudioConnection         patchCord5(filter2, 2, mixer2, 2);
AudioConnection         patchCord6(filter1, 0, mixer1, 0);
AudioConnection         patchCord7(filter1, 1, mixer1, 1);
AudioConnection         patchCord8(filter1, 2, mixer1, 2);
AudioConnection         patchCord9(mixer1, 0, i2s1, 0);
AudioConnection         patchCord10(mixer2, 0, i2s1, 1);
AudioControlSGTL5000    sgtl5000_1;

AudioInputI2S         audioInput;
AudioAnalyzeFFT1024   myFFT;
AudioConnection patchCord11(audioInput, 0, myFFT, 0);
AudioAnalyzeFFT1024   myFFT2;
AudioConnection patchCord12(audioInput, 1, myFFT2, 0);

#define SDCARD_CS_PIN    10
#define SDCARD_MOSI_PIN  7
#define SDCARD_SCK_PIN   14

float magnitude = 1.0;
int currentfilter = 0;
int filter1seen = 0;
int filter2seen = 0;
int filter3seen = 0;
int currentsong = 0;

void setup() {
  pinMode(0, INPUT); //Initialize Button 0
```

```
 pinMode(1, INPUT); //Initialize Button 1

 //Audio Shield and SD Card Initialization
 Serial.begin(9600);
 AudioMemory(30);
 sgtl5000_1.enable(); //Enable Audio Shield
 sgtl5000_1.inputSelect(myInput);
 sgtl5000_1.volume(1.0); //Set Initial Volume
 myFFT.windowFunction(AudioWindowHanning1024);
 myFFT2.windowFunction(AudioWindowHanning1024);
 SPI.setMOSI(SDCARD_MOSI_PIN);
 SPI.setSCK(SDCARD_SCK_PIN);
 if (digitalRead(SDCARD_CS_PIN)) {
  Serial.println("SD Card is high");
 } else {
  Serial.println("SD Card is low");
 }
 if (!(SD.begin(SDCARD_CS_PIN))) {
  while (1) {
   Serial.println("Unable to access the SD card");
   delay(500);
  }
 }
 //Set fourth mixer input to zero
 mixer1.gain(3, 0.0);
 mixer2.gain(3, 0.0);
 delay(100);
}

void loop() {
 if(digitalRead(0) == HIGH){
  currentfilter = 0;
 }
 if ((playSdWav1.isPlaying() == false)) {
    playSdWav1.play("SONG1.wav");
    delay(100);
 }

 float n;
 int i;
 if (myFFT.available()) {
  for (i=0; i<40; i++) {
    n = myFFT.read(i);
    if((i >= 0) && (i <= 1) && (n > 0.1)){        //  Reset Functionality
       currentfilter = 0;
    }
    if((i >= 2) && (i <= 8) && (n > 0.1)){        //  50 to 300 Hz, 0 <= i <= 8
       currentfilter = 1;
```

```
    }
    else if ((i >= 9) && (i <= 15) && (n > 0.1)){   //  301 to 600 Hz, 9 <= i <= 15
      currentfilter = 2;
    }
    else if ((i >= 16) && (i <= 22) && (n > 0.1)){  //  601 to 900 Hz, 16 <= i <= 22
      currentfilter = 3;
    }
    else if ((i >= 23) && (i <= 29) && (n > 0.1)){  // 901 to 1200 Hz, 23 <= i <= 29
      currentfilter = 4;
    }
   }
  }
  if (myFFT2.available()) {
   for (i=0; i<40; i++) {
    n = myFFT2.read(i);
    if((i == 2) && (n > 0.3)){        //100 Hz = 10% magnitude
      magnitude = 0.1;
    }
    else if ((i == 5) && (n > 0.3)){   //200 Hz = 20% magnitude
      magnitude = 0.2;
    }
    else if ((i == 7) && (n > 0.3)){  //300 Hz = 30% magnitude
      magnitude = 0.3;
    }
    else if ((i == 9) && (n > 0.3)){ //400 Hz = 40% magnitude
      magnitude = 0.4;
    }
    else if ((i == 12) && (n > 0.3)){   //500 Hz = 50% magnitude
      magnitude = 0.5;
    }
    else if ((i == 14) && (n > 0.3)){  //600 Hz = 60% magnitude
      magnitude = 0.6;
    }
    else if ((i == 16) && (n > 0.3)){ //700 Hz = 70% magnitude
      magnitude = 0.7;
    }
    else if ((i == 19) && (n > 0.3)){ //800 Hz = 80% magnitude
      magnitude = 0.8;
    }
    else if ((i == 21) && (n > 0.3)){ //900 Hz = 90% magnitude
      magnitude = 0.9;
    }
    else if ((i == 23) && (n > 0.3)){ //1000 Hz = 100% magnitude
      magnitude = 1.0;
    }
    else if ((i == 26) && (n > 0.3)){ //1100 Hz = 5000% magnitude
      magnitude = 2;
    }
```

```
    else if ((i == 28) && (n > 0.3)){ //1200 Hz = 50000% magnitude
       magnitude = 3;
    }
  }
}

if(currentfilter == 0){
  mixer1.gain(0, 1);
  mixer1.gain(1, 1);
  mixer1.gain(2, 1);
  mixer2.gain(0, 1);
  mixer2.gain(1, 1);
  mixer2.gain(2, 1);
  filter1seen = 0;
  filter2seen = 0;
  filter3seen = 0;
}

if(currentfilter == 1){
  // Low Pass Signal
  mixer1.gain(0, magnitude);
  mixer1.gain(1, filter2seen);
  mixer1.gain(2, filter3seen);
  mixer2.gain(0, magnitude);
  mixer2.gain(1, filter2seen);
  mixer2.gain(2, filter3seen);
  sgtl5000_1.volume(1);
  float freq1 = 100;
  filter1.frequency(freq1);
  filter2.frequency(freq1);
  filter1seen = 1;
}

if(currentfilter == 2){
  // Band Pass Signal
  mixer1.gain(0, filter1seen);
  mixer1.gain(1, magnitude);
  mixer1.gain(2, filter3seen);
  mixer2.gain(0, filter1seen);
  mixer2.gain(1, magnitude);
  mixer2.gain(2, filter3seen);
  sgtl5000_1.volume(1.0);
  float freq2 = 500;
  filter1.frequency(freq2);
  filter2.frequency(freq2);
  filter2seen = 1;
 }
```

```
 if(currentfilter == 3){
  //High Pass Signal
  mixer1.gain(0, filter1seen);
  mixer1.gain(1, filter2seen);
  mixer1.gain(2, magnitude);
  mixer2.gain(0, filter1seen);
  mixer2.gain(1, filter2seen);
  mixer2.gain(2, magnitude);
  sgtl5000_1.volume(1.0);
  float freq3 = 1500;
  filter1.frequency(freq3);
  filter2.frequency(freq3);
  filter3seen = 1;
 }

 if(currentfilter == 4){
  sgtl5000_1.volume(magnitude);
  mixer1.gain(0, 1);
  mixer1.gain(1, 1);
  mixer1.gain(2, 1);
  mixer2.gain(0, 1);
  mixer2.gain(1, 1);
  mixer2.gain(2, 1);
 }
 delay(200);
}
```

SPI Protocol Code

```
// include the SPI library:
#include <SPI.h>
#include <FreqCount.h>

// SD Pins 7, 10, 14

const int CS = 9;      // Chip Select from Teensy to ADC -  The pin on each device that the master can use to enable and disable specific devices.
const int SDATA = 12;   // Data from ADC to Teensy - MISO (Master In Slave Out) - The Slave line for sending data to the master
const int SCLK = 13;    // SCLK from Teensy to AD - The clock pulses which synchronize data transmission generated by the master
// set up the speed, mode and endianness for ADC
SPISettings settingsA(48000000, MSBFIRST, SPI_MODE3);
//ADC - AD7352 Settings: 48 kHz max , Most Significant Bit in data first, and Mode 3 based on CPOL: 1 & CPHA: 1

void setup() {
 // set pin modes properly
 pinMode (CS, OUTPUT);     // Set up pin mode for chip select as output
```

```
  pinMode (SDATA, INPUT);
  pinMode (SCLK, OUTPUT);
  //Initialize pins
  digitalWrite(CS,  HIGH);  // Set output on chip select as high to initialize so that input from ADC is
turned off
  digitalWrite(SCLK,  HIGH);  //Set output on clock to hight to initialize
  // initialize SPI:
  Serial.begin(57600);
  FreqCount.begin(1000);
  SPI.begin();
}
uint16_t sdata_a, sdata_b;
uint32_t count = 0;
uint32_t counta = 0;
uint32_t countb = 0;
uint32_t tracker = 0;
uint16_t sdata_a_out;
uint16_t sdata_b_out;
void loop() {
  // read bytes from ADC
  //assign settings for ADC and begin transaction
  SPI.beginTransaction(settingsA);
  digitalWrite(CS, LOW);
    count = SPI.transfer(0);
    count <<= 8;
    count |= SPI.transfer(0);
    count <<= 8;
    count |= SPI.transfer(0);
    count <<= 8;
    count |= SPI.transfer(0);
    count = count >> 1;
    tracker = count;
    counta = count >> 16;
    countb = tracker << 16;
    countb = tracker >> 16;
    sdata_a = counta;
    sdata_b = tracker;
    sdata_a_out = sdata_a >> 2;
    sdata_b_out = sdata_b >> 2;
  digitalWrite (CS, HIGH);
  SPI.endTransaction();
  Serial.print("sdata_a_out = ");
  Serial.print(sdata_a_out);
  Serial.print("      ,      ");
  Serial.print("sdata_b_out = ");
  Serial.println(sdata_b_out);
}
```