Final Report

ECE 458, Spring 2020

Group 9 - Acoustic Awareness Enabler

UNIVERSITY OF MASSACHUSETTS DARTMOUTH

Professor: Dr. Paul Fortier
Customer: Dr. Karen Payton
Advisor: Dr. Vance McCollough

Group Members:

Drew Martins: Team Lead and Programmer
Stephen Felix: Test Engineer
Ryan Dumont: Electrical Design Engineer
Michael Benker: Systems Engineer
Jared Alves: Signal Processing Engineer

Due date: April 28, 2020

Table of Contents

List of Figures

List of Tables

1. Project Overview

1.0 Background

The idea of the Acoustic Awareness Enabler is reported to have resulted from an incident in which the customer was watching a movie on her laptop with headphones. Her husband had called for her to respond, but she was unaware of this, her attention drawn to her computer. She thought then, what if there was a way for her headphones to let her know if someone was trying to get her attention? This was the apparent beginning of the Acoustic Awareness Enabler. Senior Design Project Group 9 seeks to realize this idea to a working device, capable of alerting a user wearing headphones to an external acoustic event.

1.1 Project objectives and Customer Requirements

The primary objective of the Group 9 Senior Design Project is to build a device known as the Acoustic Awareness Enabler, which is described as follows. The Acoustic Awareness Enabler is a device that is positioned between two auxiliary cables leading from an audio source to headphones. The device listens to outside noises and alerts the user (i.e. wearing headphones) to the unusually loud sound by turning off the audio throughput. The device uses three sensitivity settings, which provide the user with a selection of necessary threshold sound level (relative to surrounding environment) needed to pause the audio. The following were outlined as customer requirements for the Acoustic Awareness Enabler:

2. Customer Requirements

The customer requirements for the Acoustic Awareness Enabler are given in Table 1. These were the required conditions set by the customer early in the project lifespan and have not changed. The seventh requirement was added to the project a few months into design. From these requirements, the engineering requirements are derived. The requirements shown in red were affected by COVID-19. Because of a lack of access to components, it was not possible to test the battery life of the device with all components in place, as evidenced by the test plans section. A 3.5mm audio jack for the input and output to the device was also not necessary as no device was constructed. Customer Requirement 7 was also not implemented.

Table 1 Customer Requirements

Cust. Req. #	Customer Requirements Description
1	Self-containment of power source (rechargeable) and battery life of 8 hours
2	3.5mm audio jack for input and output
3	Must have 3 different levels of interruption noise sensitivity (High, Medium and Low sensitivity)

4	When interruption is detected, the volume of the noise going through users' headphones must be muted or lowered
5	Must be able to distinguish between ambient and interrupt with a 1% false interrupt detection rate
6	Must have a reset button to restart audio
7	Separate audio cable between device and audio source

3. Engineering Requirements, Constraints and Applicable Standards

3.1 Engineering Requirements

The Engineering requirements are shown in Table 2 below. These requirements were also affected by the COVID-19 pandemic. As outlined in the customer requirements, requirements 1, 2 and 7 could not be fully demonstrated as no physical device could be completed. However, a theoretical approach was taken to meet the other requirements. Three levels of interruption noise sensitivity were demonstrated within the "AA_Enabler" program in the appendix and in section 6.6.2. The processing in MATLAB done in the subsequent sections of 6.4 satisfy 3.1 and the requirements under Requirement 5. Requirements 3.2 and 4.1 are satisfied by the demonstration of the Acoustic Awareness Enabler prototype. This also satisfies all of the requirements under Requirement 6.

Table 2 Engineering Requirements

	Customer Requirements	Engineering Requirements
1	Self-containment of power source (rechargeable) and battery life of 8 hours	**1.1** 8 hours power supplied to device
2	3.5mm audio jack for input	**2.1** Audio Source input 3.5mm
		2.2 Audio Source Output 3.5mm
		2.3 I/D Interface – digital pot/switch
3	Must have 3 different levels of interruption noise sensitivity	**3.1** Record data and process using MATLAB
		3.2 Three Separate Sensitivity Buttons
4	When interruption is detected, audio is muted	**4.1** Use of potentiometers/switches to attenuate audio
5	Must be able to distinguish between ambient and interrupt with a 1% false interrupt detection rate	**5.1** Data analysis for algorithm to achieve a 1% false detection rate or better
		5.2 Lowest Ambient Noise of 30 dB
		5.3 Must have 100 Hz Sampling frequency
		5.4 Need a sound level detector
6	Must have a reset button to restart audio	**6.1** Physical reset button
		6.2 Switches resume audio after being turned off
		6.3 Recalibrate device, delay further interrupts for 20 seconds while gathering data
7	Separate audio cable between device and audio source	**7.1** Included with final product

3.2 Constraints

The following constraints have been imposed on the project. These have mostly remained constant throughout the course of the design project. A major area of concern for the physical device was portability factor, as the device was meant to be used at home as well as on the go. Another constraint was form factor or shape of the device. This would have been a necessary consideration when factoring in the device fitting into the enclosure. Another area of concern was how available components were for assembly. Due to COVID-19 many of these constraints were not constrictive. For example, the group was well under the original budget of $500 because no physical PCB was realized.

- Portability
- Form Factor (maximum size constraint of 3 x 2.5)
- Availability of Components
- Budget of $500 supplied by the customer
- Time: Project must be completed by Spring 2020

3.3 Standards

The list below is a compilation of ethical standards that the project adhered to. The IEEE code of ethics contains 10 ethical statements that must be adhered to for practicing engineers. For communicating with the microcontroller, the C programming language is used, which contains its own standards. The ports of the device are standard 3.5mm jacks. Sound levels must comply with a few standards listed below. I^2C and SPI standards are relevant to hardware interfaces for seven bit addressing. The enclosure and battery were not used for the project, but the standards for these are listed.

- IEEE Code of Ethics
- Microcontroller Coding – ATMega328pb, C Programming AVR
- 3.5mm Auxiliary Port
- Sound Level – IEC 652 TYPE2 and ANSI S1.4 TYPE 2
- I^2C and SPI Standards: 7 bit addressing
- IEC62133 – Battery Safety for Lithium Polymer Batteries
- Electrical Enclosures – IP40, NEMA 1

3.4 Ethical Issues

COVID-19 presents many new issues to the forefront which were not present before the pandemic occurred. The evacuation of on-campus students and faculty has effectively altered the expected outcome of the final design product. Due to a lack of availability of physical components, emphasis has shifted towards a more theoretical approach. Design tends to differ from manufacturing and therefore a necessarily accurate comparison between the two cannot be

made. The situation also presented scheduling issues for the final few months of the project and require restructuring of previous plans. Zoom software was used to abide by campus and societal rules of social distancing.

3.5 Concept of Operations

In terms of user interaction with the Acoustic Awareness Enabler, there arise only the following situations: turning the device on or off, changing the sensitivity threshold and resetting the device to start a new cycle of data collection or to turn the audio back on after it was turned off as a result of an interrupt.

Behind the scenes, the Acoustic Awareness Enabler is taking in acoustic data and testing whether a recent level exceeds a threshold level calculated from the average and standard deviation of a longer period of data. If the device finds that the sound level has not been exceeded, it will replace the oldest data set with the most recent and calculate the average and standard deviation continuously and test if a yet more recent sound will exceed a new threshold. If the threshold is exceeded, the device will turn off the audio throughput to the headphones. The Software Diagram flowchart outlines how the device works "under the hood". In attempt to outline how the Acoustic Awareness Enabler works functionally, a matrix was constructed to outline what events would be triggered at which time under all possible conditions. Finally, a high-level schematic was designed to give an overview of what hardware would be used to build the Acoustic Awareness Enabler.

Figure 1. AAE Comic Strip

4. Functional Overview

The Acoustic Awareness Enabler's function is to detect sudden increases in environment sound levels and mute audio throughput from one's phone or computer to headphones when this sudden increase is detected. There are three sensitivity settings, which determine how loud a sound needs to be to pause one's audio.

Six parts or subsystems make up the Acoustic Awareness Enabler. These subsystems are the Acoustic Information Subsystem, Processing Subsystem, Setting Information Subsystem, Auxiliary Switching Subsystem, Power Supply Subsystem and the Enclosure & Formfactor Subsystem.

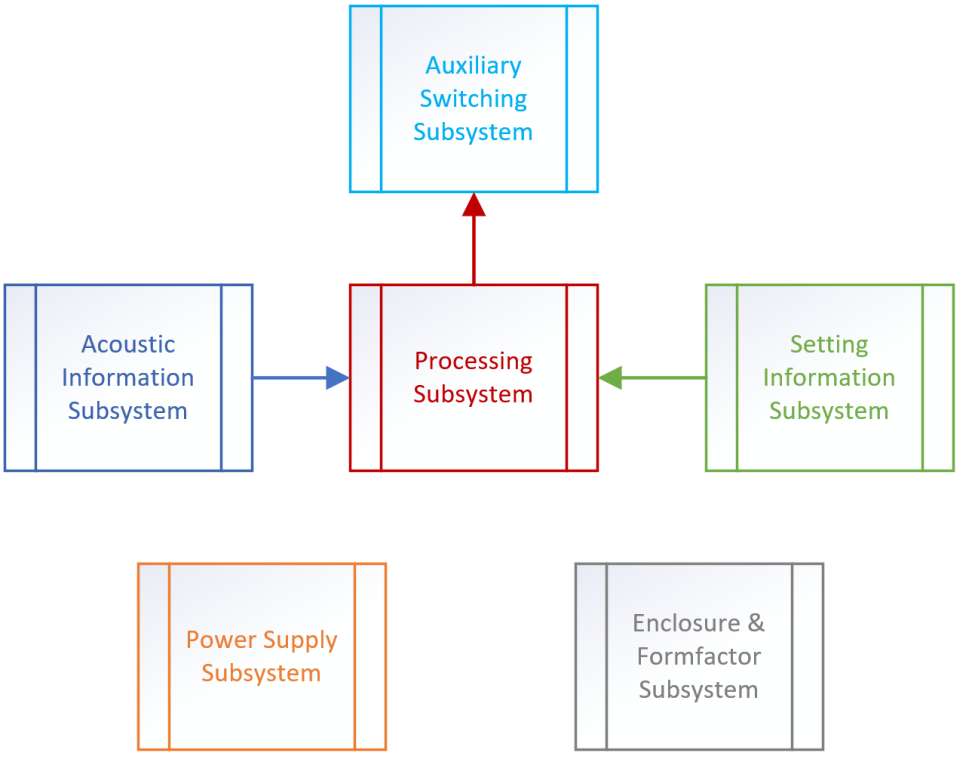

Figure 2. Subsystems of the Acoustic Awareness Enabler

A microphone is used to collect environment sound levels. The sound level information is amplified and sent to the microprocessor, which analyzes this data to determine if an interrupt is achieved. The ON/OFF buttons RESET button and sensitivity setting buttons convey information to the microprocessor as to the current mode of operation. When an interrupt is detected, the microprocessor communicates this information to the digital switch, which will mute the audio throughput to the headphones. A battery power source is needed as well as an enclosure and a printed circuit board as a form factor apparatus to confine the components into the enclosure. Below is a high-level schematic that provides an overview of how these main subsystems interact.

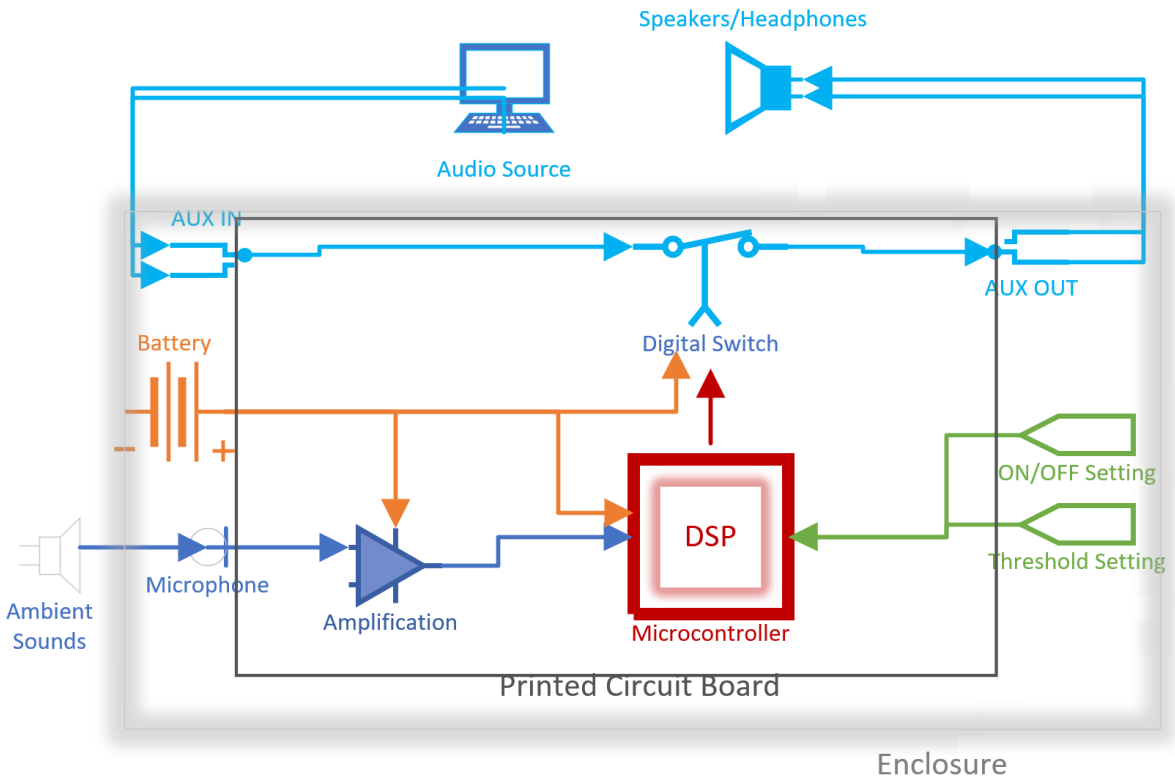

Figure 3. High-Level Schematic

The following table outlines the various modes of operation that the Acoustic Awareness Enabler is found, as well as identifying which possible changes would be needed to alter the current mode of operation.

CURRENT MODE		EVENTS (2 SECOND AVERAGE)				BUTTONS					
STATE	SETTING	AMBIENT DATA	"QUIET INT" DATA	"MED INT" DATA	"LOUD INT" DATA	ON	OFF	MIN	MED	MAX	RESET
PLAYING & COLLECTING WITHOUT 20 SEC OF DATA	ON, NOT INTERRUPTED, MIN	Averaging of ambient for 20 sec	Averaging of ambient for 20 sec	Averaging of ambient for 20 sec	Averaging of ambient for 20 sec	nothing	change state: Playing & not collecting	nothing	change setting to MED. change state: PLAYING & COLLECTING WITHOUT 20 SEC OF DATA (from beginning of 20 secs)	change setting to MAX. change state: PLAYING & COLLECTING WITHOUT 20 SEC OF DATA (from beginning of 20 secs)	change state: PLAYING & COLLECTING WITHOUT 20 SEC OF DATA (from beginning of 20 secs)
	ON, NOT INTERRUPTED, MED	Averaging of ambient for 20 sec	Averaging of ambient for 20 sec	Averaging of ambient for 20 sec	Averaging of ambient for 20 sec	nothing	change state: Playing & not collecting	change setting to MIN. change state: PLAYING & COLLECTING WITHOUT 20 SEC OF DATA (from beginning of 20 secs)	nothing	change setting to MAX. change state: PLAYING & COLLECTING WITHOUT 20 SEC OF DATA (from beginning of 20 secs)	
	ON, NOT INTERRUPTED, MAX	Averaging of ambient for 20 sec	Averaging of ambient for 20 sec	Averaging of ambient for 20 sec	Averaging of ambient for 20 sec	nothing	change state: Playing & not collecting	change setting to MIN. change state: PLAYING & COLLECTING WITHOUT 20 SEC OF DATA (from beginning of 20 secs)	change setting to MED. change state: PLAYING & COLLECTING WITHOUT 20 SEC OF DATA (from beginning of 20 secs)	nothing	
PLAYING & COLLECTING WITH 20 SEC OF DATA	ON, NOT INTERRUPTED, MIN	Compare with running ambient. Average into running ambient.	Averages into ambient level	Averages into ambient level	change state: NOT PLAYING & NOT COLLECTING (interrupted)	nothing	change state: Playing & not collecting	nothing	change setting to MED. change state: PLAYING & COLLECTING WITHOUT 20 SEC OF DATA (from beginning of 20 secs)	change setting to MAX. change state: PLAYING & COLLECTING WITHOUT 20 SEC OF DATA (from beginning of 20 secs)	change state: PLAYING & COLLECTING WITHOUT 20 SEC OF DATA (from beginning of 20 secs)
	ON, NOT INTERRUPTED, MED	Compare with running ambient. Average into running ambient.	Averages into ambient level	change state: NOT PLAYING & NOT COLLECTING (interrupted)	change state: NOT PLAYING & NOT COLLECTING (interrupted)	nothing	change state: Playing & not collecting	change setting to MIN. change state: PLAYING & COLLECTING WITHOUT 20 SEC OF DATA (from beginning of 20 secs)	nothing	change setting to MAX. change state: PLAYING & COLLECTING WITHOUT 20 SEC OF DATA (from beginning of 20 secs)	
	ON, NOT INTERRUPTED, MAX	Compare with running ambient. Average into running ambient.	change state: NOT PLAYING & NOT COLLECTING (interrupted)	change state: NOT PLAYING & NOT COLLECTING (interrupted)	change state: NOT PLAYING & NOT COLLECTING (interrupted)	nothing	change state: Playing & not collecting	change setting to MIN. change state: PLAYING & COLLECTING WITHOUT 20 SEC OF DATA (from beginning of 20 secs)	change setting to MED. change state: PLAYING & COLLECTING WITHOUT 20 SEC OF DATA (from beginning of 20 secs)	nothing	
PLAYING & NOT COLLECTING	OFF	Not collected	Not Collected	Not Collected	Not Collected	change state: PLAYING & COLLECTING WITHOUT 20 SEC OF DATA	nothing	nothing	nothing	nothing	nothing ?
NOT PLAYING & NOT COLLECTING	ON, INTERRUPTED	Not collected	Not Collected	Not Collected	Not Collected	nothing	change state: Playing & not collecting	nothing	nothing	nothing	change state: PLAYING & COLLECTING WITHOUT 20 SEC OF DATA (from beginning of 20 secs)

Figure 4. Functional Flowchart

5. Alternatives

Early in the design project, many alternatives were explored with options weighed based on specific criteria. The major factors considered were choice of microcontroller and method of threshold calculation. Towards the middle of the design process, several new factors were considered for both the microcontroller and the threshold method. These will be explored in the subsection below.

5.1 Alternative 1 – Microcontroller Selection

At the start of the project 3 different microcontrollers were examined and from there the best choice between the microprocessors were chosen from.

Microcontroller	ADC Resolution	Architecture	Cost*	SRAM	Voltage Range
ATmega328PB	**10-bit**	**8-bit AVR**	**$1.41**	**2 kB**	**1.8 V – 5.5 V**
ATmega2560	10-bit	8-bit AVR	$12.35	8 kB	4.5 V – 5.5 V
ATSAMD21G18	14-bit	32-bit ARM	$3.22	32 kB	1.62 V – 3.63 V

*Cost Information is based on the price of the device as of November 2019.

Alternative microcontrollers had their advantages to the ATmega328PB, mainly both devices had more RAM meaning the CPU could store more onto memory, which means that faster algorithms could be used to compute standard deviation and 2s Averages. Since adding 200x 10-bit integers can potentially leave a number that exceeds the highest 16-bit integer (65535), the ATSAMD21G18 having 32-bit registers and 32-bit operations means that we could add the average of all 200 samples instead of adding 50 samples and adding that to the 2s Average 4 times before actually averaging the number which is what was done in the program for the ATmega328PB. Another advantage is ADC resolution, higher resolution means being able to distinguish audio levels better, which the ATSAMD21G18 was best suited for among the three.

There are also multiple reasons for choosing the ATmega328PB, the main reason is familiarity, one of the team members has experience with using the CPU to write programs on. The other two CPUs that was not the case, especially with the ATSAMD21G18 which is also a different CPU architecture, which would have required additional time to learn to write programs for it. One additional reason is flexibility on voltage. The ATmega2560 needs at least 4.5V, the ATmega328PB could operate at 3.7V should the power savings be needed, giving this CPU the widest range of voltage levels to choose from.

The other reason is cost, it was much cheaper to get the ATmega328PB as a chip compared to getting the AT2SAMD21G18 or an ATmega2560 and it was also cheaper to purchase development boards (such as the ATmega328PB XPlained MINI) than it is to get development boards for other CPUs.

5.2 Alternative 2 – Threshold Calculation

When considering the method of interrupt threshold calculation, three methods were identified. These methods were the “Static dB Level”, “dB Interrupt-Factor Method” and “Data Analysis Method.”

The first method, the “Static dB Level” identifies a specific sound level which, when exceeded an interrupt is identified. For instance, one may select 80 dB to be the interrupt threshold. If a sound level is detected that exceeds 80 dB, then the audio throughput to one’s headphones is muted. This method was not selected for use in the Acoustic Awareness Enabler. The main issue identified with this method is that average sound levels vary for different environments. The average sound level in a lab is much quieter than in a car, for instance although both scenarios may be situations identified as candidates for the Acoustic Awareness Enabler. If this method were to be implemented, a way for the user to manually select the threshold level may be used, however this would invalidate the purpose for three threshold sensitivity settings.

The “dB Interrupt-Factor Method” identifies a ratio between an average environment level and the interrupt. For instance, one might say that an interrupt is approximately 1.4 times the level of the ambient. This method solves the issue with the first method with regards to the setting levels. The main issue with this method is that, given the level of fluctuation of environments, a dB-factor may be achieved regularly in some environments, while it may never be achieved in others. The issue about fluctuation of environments raises concerns about achieving a 1% false detection rate, which is an engineering requirement. For this reason, this method was not selected.

The “Data Analysis Method” analyzes data of an environment and compares it to a normal Gauβian distribution. The threshold is identified as a certain number of standard deviations from the average sound level. This method was found to be consistent with collected data on several environments. Further

simulations suggested that a 1% false detection rate is best achieved using this method and for this reason, this method was selected for use in the Acoustic Awareness Enabler.

6. Technical Design Descriptions

6.1 Subsystem 1 – Analog Electronics

The Analog Electronics Subsystem can be divided into two parts, consisting of the microphone and the amplifier. These are two critical components of the Acoustic Awareness Enabler needed to get optimal results. The microphone takes in the ambient noise level and interrupts in the current environment where the user is located. These signals are fed into the amplifier subsystem so the signal can be amplified and analyzed by the microcontroller's ADC. Considering that the project progress was restricted by COVID-19, the results below are mostly theoretical.

6.1.1 – Microphone Subsystem

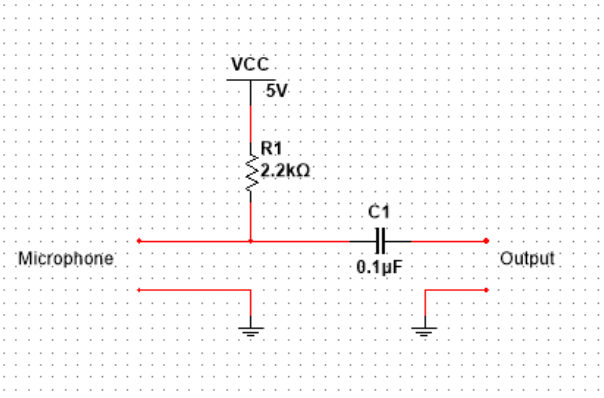

Figure 5- Microphone Subsystem

The microphone subsystem consists of a simple two pin microphone with the positive pin pulled high by a 2.2kohm resistor to a positive voltage rail of 5vdc. The value for the resistor was chosen based on the manufacturers specs to achieve nominal impedance within the microphone. A .1 microfarad capacitor was placed in parallel with the microphone and is known as a coupling capacitor. Its function in the circuit is too block the DC signal from the source voltage and only allow the AC component of the input signal to pass. Test conducted with this circuit yielded voltage outputs of 28-34mv. Below is an oscilloscope output viewed on an Analog Discovery 2 of the microphone taking in ambient noise and an interrupt represented by the spike in the middle of the graph.

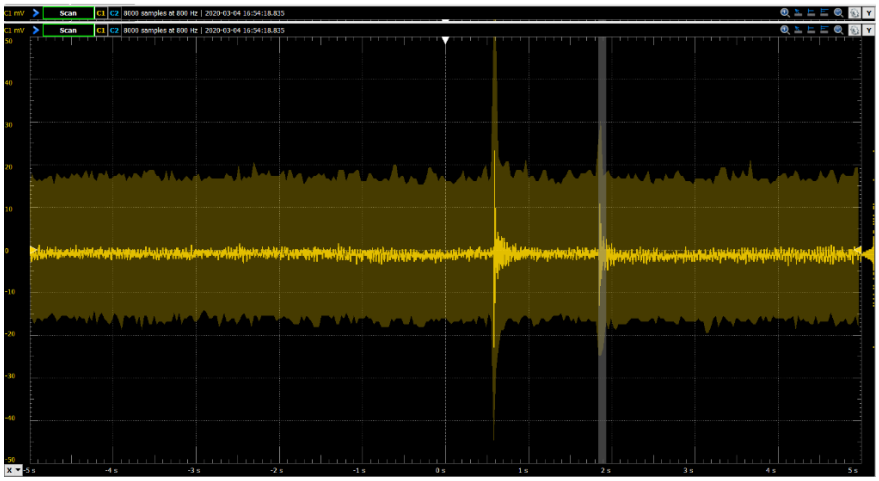

Figure 6 - Microphone Output

6.1.2 – Amplifier Subsystem

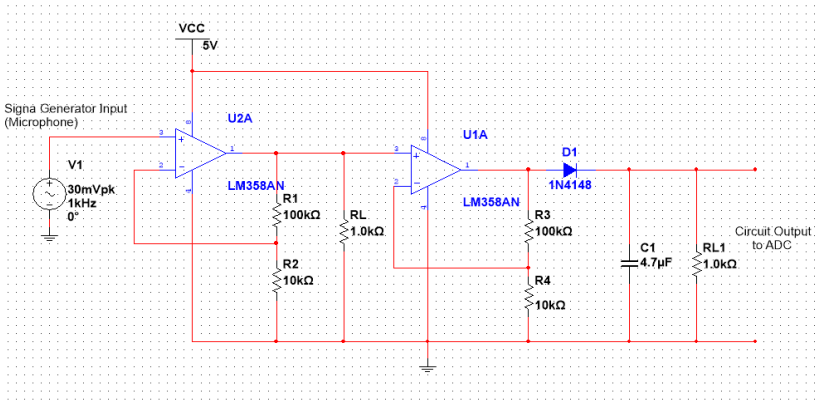

Figure 7 - Amplifier Subsystem

The amplifier subsystem consists of two operational amplifiers cascaded together, with the output of the second amplifier feeding into a rectifier circuit consisting of a diode and a capacitor. The amplifiers used are LM358 op-amps, which are in a non-inverting configuration. Luckily, two identical op-amps are housed within one IC chip allowing more flexible design of the PCB. The amplifiers are cascaded together to allow the proper gain needed of the signal coming from the output of the microphone. Due to their flexibility with voltage requirements, they utilize the same 5 vdc voltage rail as the microphone circuit. At the output of the second amplifier, amplified signal leads into a half-wave rectifier circuit consisting of a 1N4148 diode and a 470uF capacitor. The signal coming from the microphone is an AC signal and needs to be converted to DC to be processed by ADC within the microcontroller. Below is a simulation in Multisim where the output signal is probed before (yellow sine wave) and after (red dc output) the rectifier output to show how it works. Before the diode there was a voltage of 3.450vdc and after the diode the simulation showed 2.721vdc. The voltage drop is normal, and this proves that the amplification circuit is working considering that the input signal was only 30mvdc.

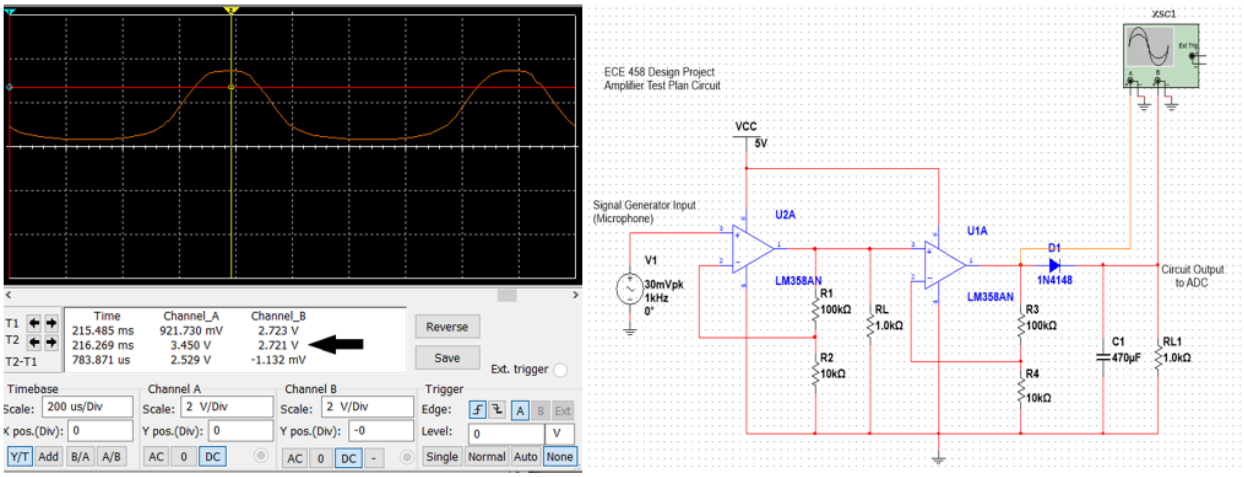

Figure 8 - Amplifier Circuit Output

6.2 Subsystem 2 – Auxiliary Switching

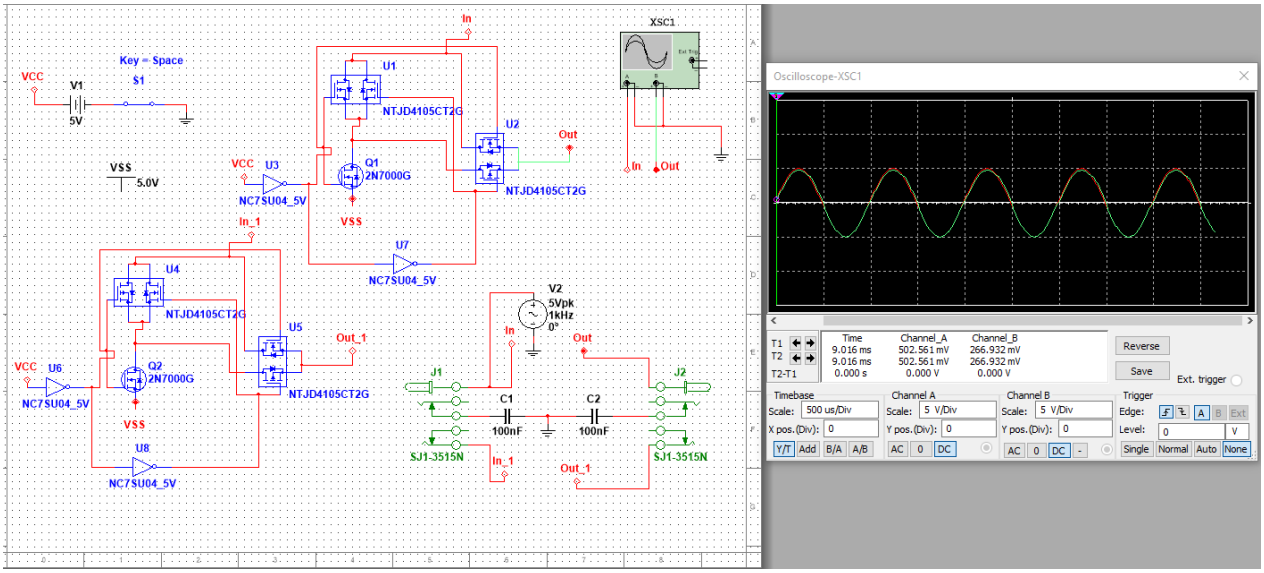

Figure 9. Aux w/ Digital Switch

By reading the digital switch data sheet, the team were able to emulate the inner circuit of one of the switches in the chip. The way the CD4066B CMOS Quad Bilateral Switch work is that inside the chip contains four switches that are each controlled by a controller pin that will close the switch when it receives a voltage signal and remains open when there is no signal connected to the pin. Now the idea is that the processor will send a signal to the controller pin to open the switch. When the processor uses the microphone to detect an interrupt in the environment, a signal will be sent to the controller pin to open the switch cause the flow of sound waves to not go through the aux port. While physically performing this circuitry in the lab, it turned out successful.

Initially the auxiliary ports had both grounds connected to each other instead of a common power rail as to reduce noise. However, this was revised as the noise wasn't entirely cleared than if connected directly to the GND rail. The first revision includes connecting a capacitor to each auxiliary ground pin and connect them both together in parallel to the resistor and the resistor goes to ground, but the noise still retained. The second revision removes the resistor and just connects the capacitors straight to ground (*Figure 6*) and this solves the noise issue.

The Auxiliary Switch control pins are connected to PORTB1:0 on the ATmega328PB. One switch is used for each side of the headphone hence the need for two switches and two ports. To turn on both switches, pins PB1:0 are both on (sending current to the auxiliary port), and to turn both switches off, pins PB1:0 are both off. The Acoustic Awareness Enabler toggles both ports on and off depending on the standard deviation.

6.3 Subsystem 3 – Setting Information

What is important is creating a user interface that is simple to understand and get the jobs done. The main component is three buttons that set the threshold sensitivity, the three sensitivities are discussed further in detail in section 6.6.2. Aside from that is the power button, of course that turns the device off and on. Aux pot is included in both sides for input for headphones and output for device. Also, a reset button that will recalibrate the device for better calculation of the ambient noise.

Acoustic Awareness Enabler has 5 Buttons, an on/off button (connected to PORTC4), a reset button (connected to PORTC3) and three threshold sensitivity buttons (connected to PORTC2:0). PORTC2 sets the threshold sensitivity level to high, PORTC1 to medium and PORTC0 to low. On/Off Button sets the device into sleep mode and the auxiliary ports stay connected. When the button is pressed again while in sleep mode, the device restarts to initialization of the program. The Reset button restarts the program back to initialization.

Pin Layout for the Buttons:

ATmega328PB I/O	Function	Description
PORTC4	On/Off Button	Turns the device Off and back On
PORTC3	Reset Button	Resets the Program
PORTC2	High Sensitivity Button	Sets Sensitivity Mode to High
PORTC1	Medium Sensitivity Button	Sets Sensitivity Mode to Medium
PORTC0	Low Sensitivity Button	Sets Sensitivity Mode to Low

For debugging purposes (not incorporated into the PCB design), 3 LEDs were set to PORTD3 and PORTD1:0. LED lights indicate what sensitivity mode is activated.

Pin Layout for the LEDs:

ATmega328PB I/O	Function	Description

PORTD3	High Sensitivity LED	If on, device is on High Sensitivity Mode
PORTD1	Medium Sensitivity LED	If on, device is on Med. Sensitivity Mode
PORTD0	Low Sensitivity LED	If on, device is on Low Sensitivity Mode

6.4 Subsystem 4 – Processing Subsystem

The Processing Subsystem can be subdivided into four multiple subsections: The intake to the A/D converter data, threshold and standard deviation calculation, False Detection rates and environment data plots.

In the microcontroller, the whole processing subsystem takes place within a timer interrupt service routine that occurs once every 10ms. At the beginning of the ISR the program reads the ADC and stores a number into an array. After storing the first 50 numbers they are added to the 2s Average and the process continues until 200 numbers total are added to the 2s Average. From there the 2s Average is computed and stored into an array of 2s Averages and added to the 20s Average. The process repeats until there are ten 2s Averages and then the 20s Average is computed. With the 20s Average and 2s Average array the standard deviation is calculated. If the standard deviation exceeds the threshold (which varies depending on the sensitivity mode), the auxiliary ports are disabled, else the auxiliary ports are reenabled.

6.4.1 – ADC Intake

The Analog to Digital Converter was set to a reference voltage of 5V and input was set to a varying voltage in order to test the value presented by the 10-bit ADC. As shown in the table, as the input voltage on pin C5 increases towards the reference voltage, the ADC value increases up to the maximum 10-bit integer value (1023).

ADC Test Code (CT_01)			
Vref (Aref)	Vin (DC)	adc_value (Decimal)	adc_value (Binary)
5.0 V	0 V	0	0b0000000000
5.0 V	1 mV	0	0b0000000000
5.0 V	3 mV	0	0b0000000000
5.0 V	5 mV	1	0b0000000001
5.0 V	10 mV	2	0b0000000010
5.0 V	15 mV	4	0b0000000100
5.0 V	20 mV	6	0b0000000110
5.0 V	30 mV	8	0b0000001000
5.0 V	40 mV	11	0b0000001011
5.0 V	50 mV	13	0b0000001101
5.0 V	60 mV	16	0b0000010000
5.0 V	80 mV	18	0b0000010010
5.0 V	100 mV	23	0b0000010111
5.0 V	150 mV	31	0b0000011111
5.0 V	200 mV	44	0b0000101100
5.0 V	300 mV	68	0b0001000100
5.0 V	400 mV	86	0b0001010110
5.0 V	500 mV	108	0b0001101100
5.0 V	600 mV	130	0b0010000010
5.0 V	800 mV	167	0b0010100111
5.0 V	1 V	208	0b0011010000
5.0 V	1.25 V	258	0b0100000010
5.0 V	1.5 V	314	0b0100111011
5.0 V	1.75 V	367	0b0101101111
5.0 V	2 V	412	0b0110011100
5.0 V	2.25 V	465	0b0111010001
5.0 V	2.5 V	515	0b1000000011
5.0 V	2.75 V	569	0b1000111011
5.0 V	3 V	620	0b1001101100
5.0 V	3.25 V	670	0b1010011110
5.0 V	3.5 V	723	0b1011010011
5.0 V	3.75 V	767	0b1011111111
5.0 V	4 V	828	0b1100111100
5.0 V	4.25 V	877	0b1101101101
5.0 V	4.50 V	929	0b1110100001
5.0 V	4.75 V	986	0b1111011010
5.0 V	5 V	1023	0b1111111111

Figure 10 ADC Test Code

6.4.2 – Thresholds and Standard Deviations

The thresholds and standard deviations were the main area of concern for the processing subsystem of the project. Several program simulations and methods were used to generate accurate standard deviations for the various threshold levels. For the device, three user selected thresholds were needed, and these were all functions of the standard deviation and average ambient level of a particular environment. Environments were collected using a sound meter and plots were constructed in MATLAB demonstrating each data point along with overlapping curves for each threshold.

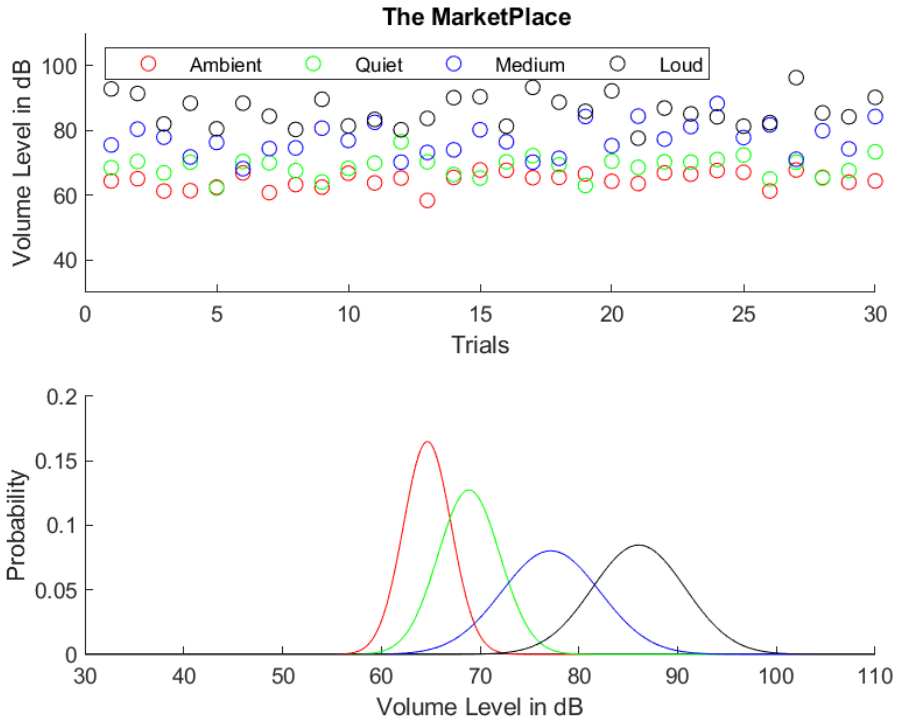

Figure 11 Marketplace

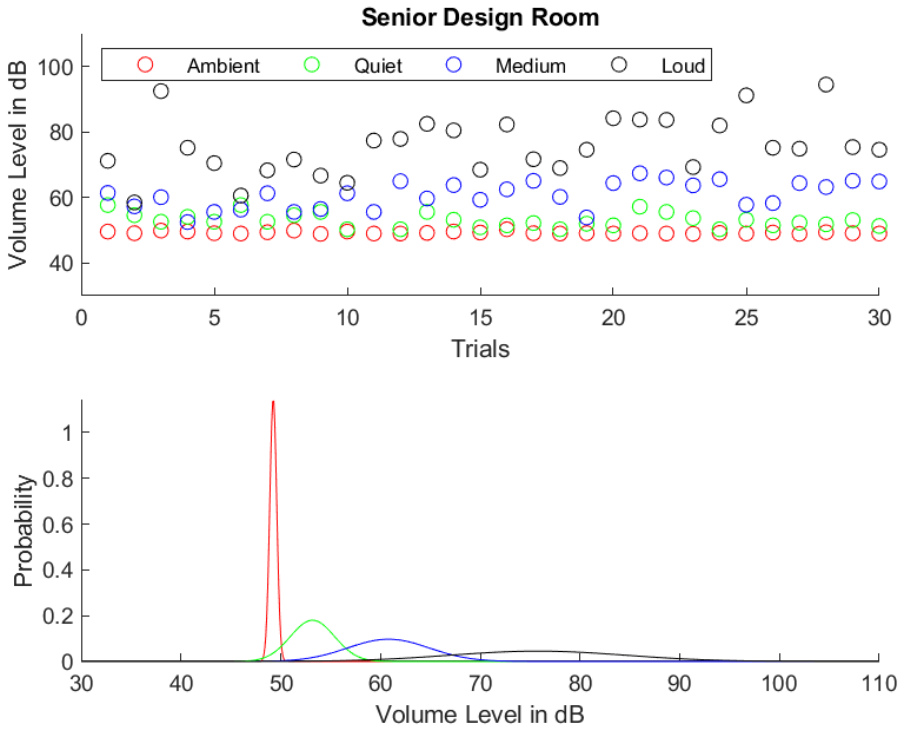

Figure 12 Senior Design Room

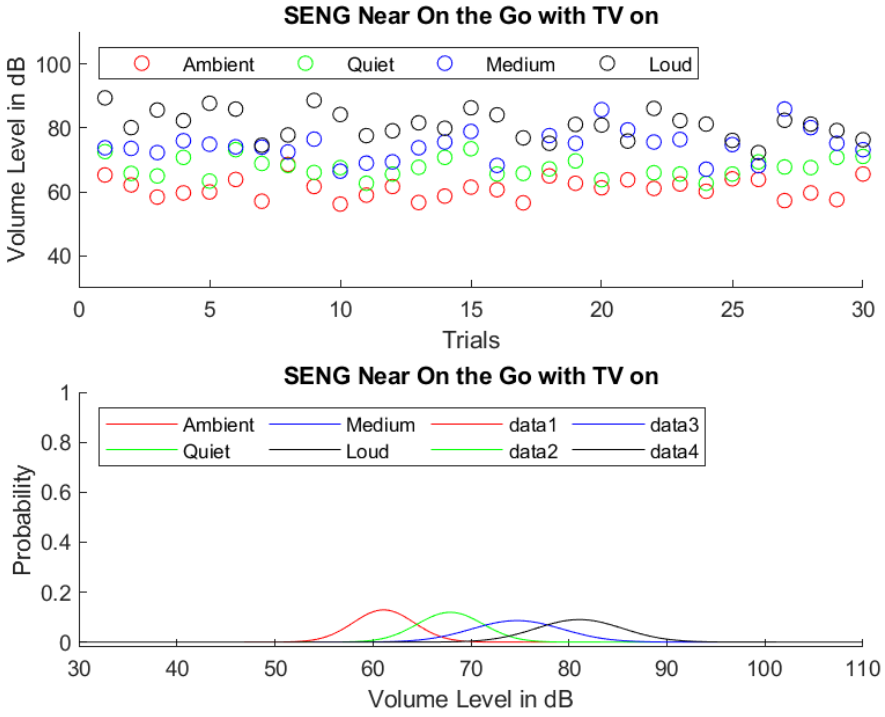

Figure 13 SENG

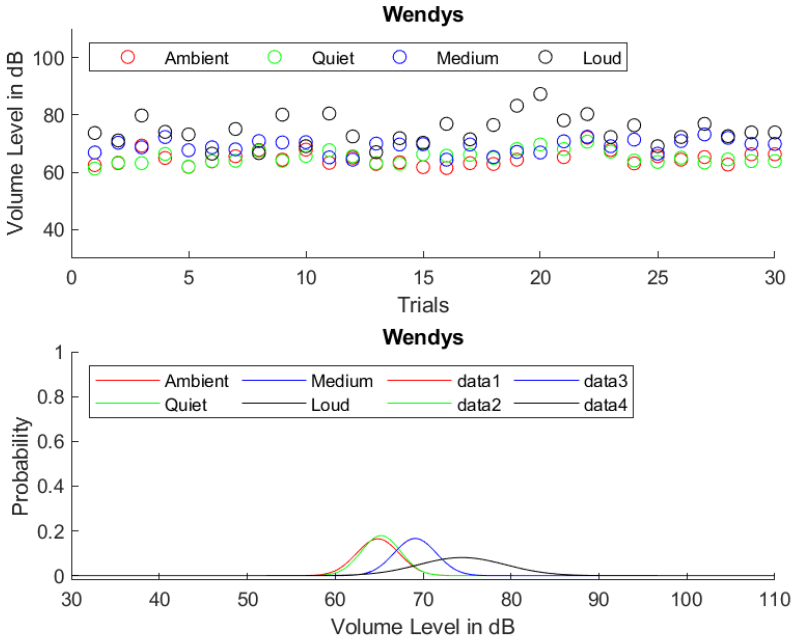

Figure 14 Wendy's

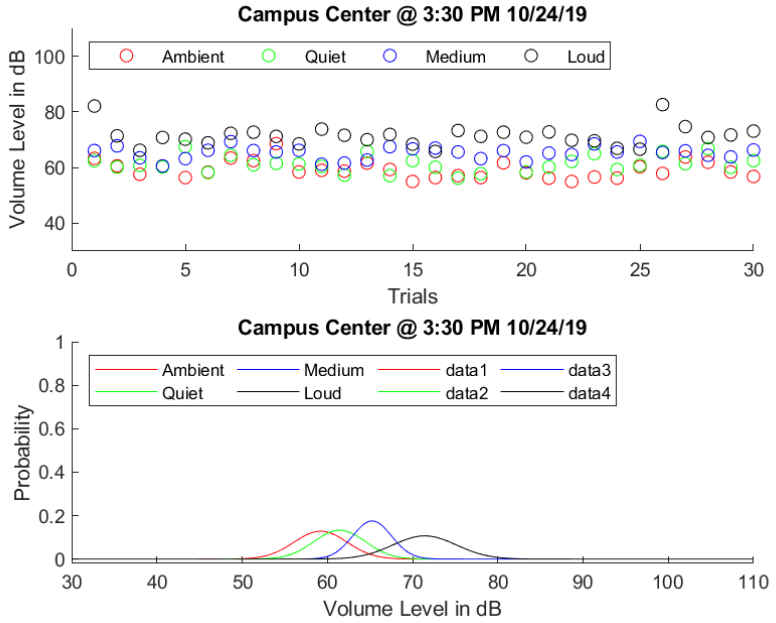

Figure 15 Campus Center

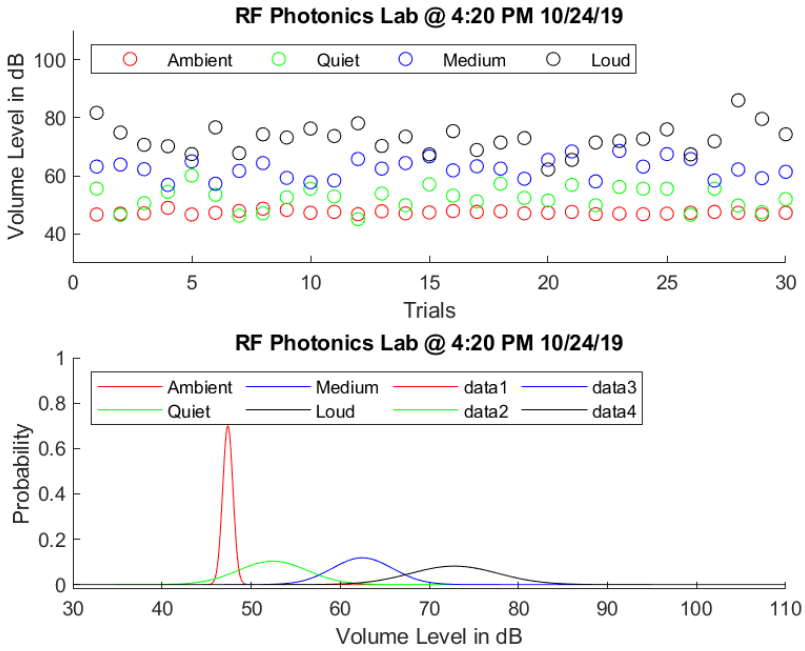

Figure 16 RF Photonics Lab

In each environment, thirty data points were collected for each category of ambient, low, medium and high interrupt levels. These are shown within each scatter plot as separate points. The full MATLAB code is included in the appendix. Gaussian or Normal Distribution functions are used for each threshold.

Additionally, a program in C# was constructed which utilized the samples collected with the sound meter to test out various standard deviation multiples for each threshold. The program used ClosedXML provided by NuGet to read the Excel files into a program and manipulate the data. The user could select which environment to read data from and then a threshold level was provided by the program depending on which level was inputted. A sample output is provided below, with the full program given in the appendix section.

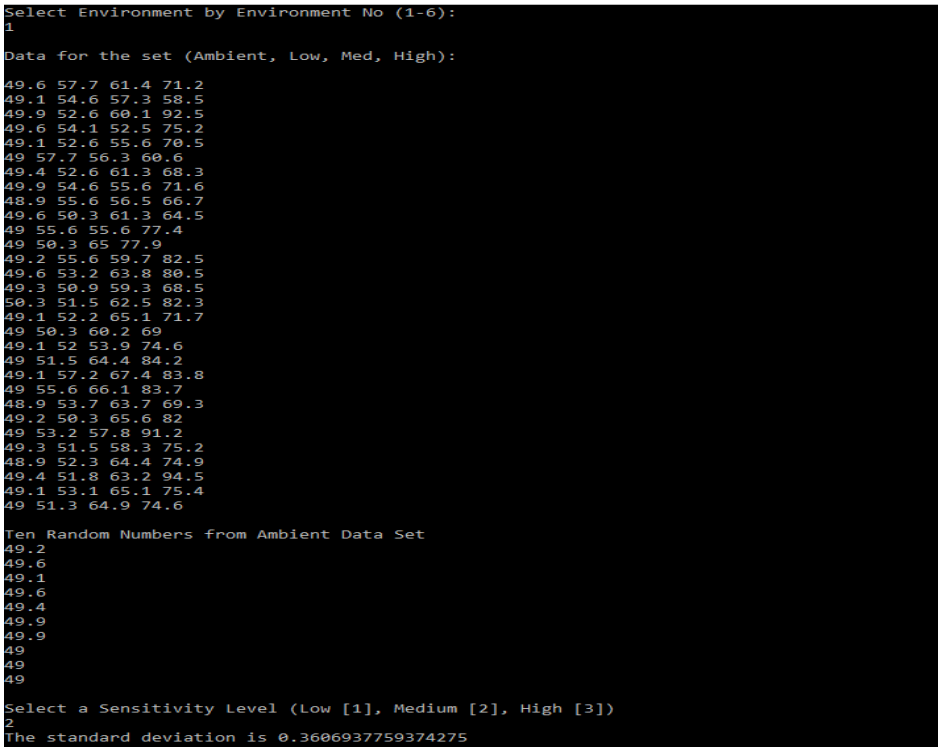

Figure 17 Program Sample

6.4.3 – False Detection Rates

A false detection, with respect to the Acoustic Awareness Enabler is when an interrupt is detected when there is no physical reality that would suggest that an interrupt had occurred. A false detection may be frustrating for the user of the Acoustic Awareness Enabler as this means that their audio was turned off for no good reason. In order to assess the performance of the Acoustic Awareness Enabler, a demonstration of a sufficiently low (less than 1%) false detection rate is needed, as per engineering requirement (XYZ). The following information is needed to determine a false detection rate (FDR).

Table 3. Detection Rates Matrix

Variable	Name	Description

Environment Data (Ambient)	n/a	Collection of data points. Running intake information to the AAE.
2 second average	2secAve	The most recent 2 seconds of data (typically 200 data points) are averaged to determine an instantaneous sound level.
20 Seconds Averages Array	Data20sec	The 20-second averages array is typically made of 10 numbers, which are 2-second averages of the most recent 20 seconds of data.
20 Second Average	20secAve	The average sound level from the last 20 seconds of data.
20 second Standard Deviation	20secStd	The standard deviation of the environment data is calculated.
Threshold Setting	T	The threshold setting is High, medium or low as selected by the user. The threshold setting is not a sound level, but a is the number of standard deviations from an ambient average that will theoretically trigger an interrupt.
Running Threshold	RT	The running threshold is the instantaneous sound level required for an interrupt to be detected. This value is updated after each new 2-second average. The running threshold is calculated using the following formula: $RT = 20secAve + 20secStd * T$
Interrupt detected	ID	A detected interrupt means that the running threshold has been exceeded by the instantaneous 2 second average. $ID = \begin{cases} 1 & 2secAve > RT \\ 0 & 2secAve < RT \end{cases}$
Real Interrupt	RI	A real interrupt is an event that may be detected. This is the case in which the source of an interrupt is worthy of being detected. The value of RI is determined by previous knowledge of the event, which was acquired through data collection. If the interrupt is real, RI=1. If the interrupt is not real, RI=0.
True Interrupt Detection	TID	A true interrupt detection, TID happens when an interrupt is detected that, through collected data is found to be a verified real interrupt. $TID = \begin{cases} 1 & ID = RI = 1 \\ 0 & otherwise \end{cases}$
False Detection	FD	False detection is an event where an interrupt is detected, while the interrupt is unfounded. $FD = \begin{cases} 1, & ID = 1\,, RI = 0 \\ 0, & otherwise \end{cases}$
False Negative	FN	A false negative means that an interrupt was not detected, despite the existence of a real interrupt.

		$FN = \begin{cases} 1, & ID = 0, RI = 1 \\ 0, & otherwise \end{cases}$
False Detection Rate	FDR	The false detection rate is the probability of the event $RI = 0$, given that $ID = 1$. $\boldsymbol{FDR = Pr[(RI = 0)\|(ID = 1)]}$

In order to calculate a false detection rate, a data set is required that possesses both real interrupts and interrupt detection. Data samples were collected for six environments containing samples for all possible events. Using data from these environments, false detection rates may be calculated. One parameter that the designers of the Acoustic Awareness Enabler may easily manipulate is the Threshold setting (Ţ) values. There exist three values that are used for the threshold setting, depending on the user-selected sensitivity setting of high, medium or low.

$$Ţ = \begin{cases} 4.00, & High \\ 12.00, & Medium \\ 24.00, & Low \end{cases}$$

As shown next, the false detection rate (FDR) is a function of Ţ. The false detection rate should be as low as possible, though it is an engineering requirement that it be below 1%. Since Ţ is an array of three integers, there will also exist three different false detection rates.

$$FDR = Pr[(RI = 0)|(ID = 1)] = Pr[(RI = 0)|\big(2secAve > (20secAve + 20secStd * Ţ)\big)]$$

The six environments were simulated in MATLAB using the following procedure. Ten random numbers from the collected data were allocated to the array signifying the previous 20 seconds. This data would be used to analyze how that collection of data will respond to other data points. The set of data is simulated against a controlled sequence of ambient-level sounds, then a series of quiet-level, medium-level and loud-level interrupts. Given knowledge of which levels are supposed to trigger interrupts, it was calculated whether and when an interrupt was detected. Using the definition of variables above, a false detection rate as well as a false negative rate was determined each time the program ran. The following figure is an example of the plotting of true interrupt detections (TID). The first 10 samples were the data points that supplied the basis for which the following were tested against. The following 15 were further ambient-level data points, where a true interrupt detection would be impossible (because real interrupt RI = 0). The remaining points were 25 each in the order of quiet interrupts, medium interrupts and loud interrupts.

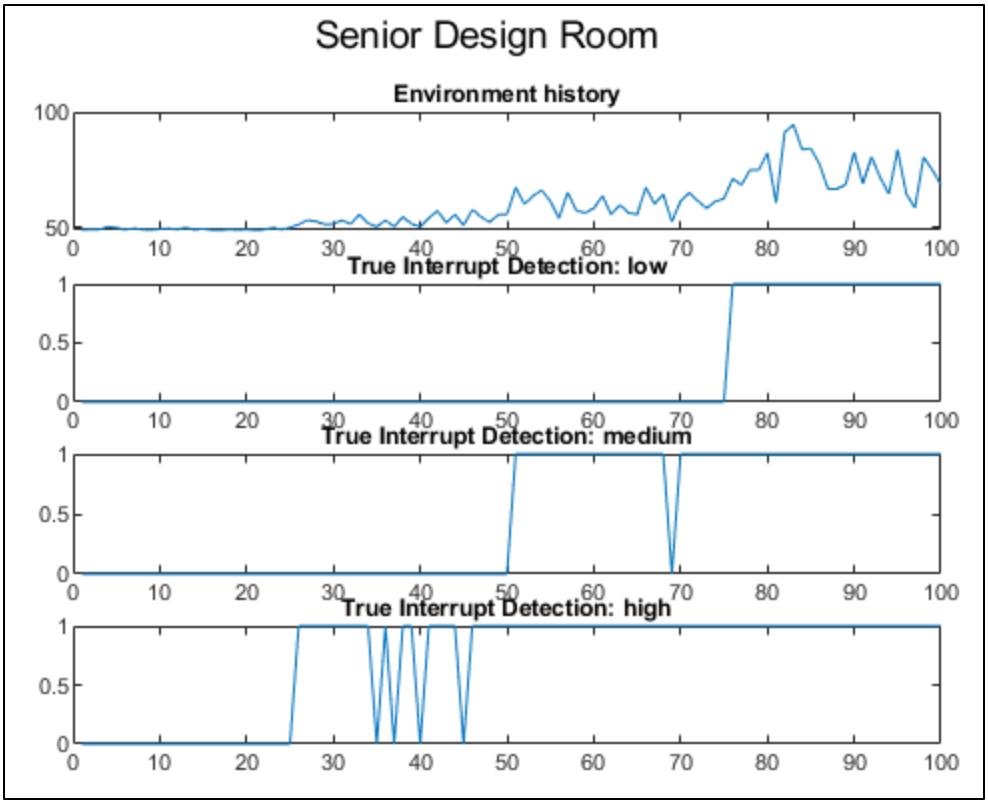

Figure 18. True Interrupt Detection in Senior Design Room

The number of standard deviations to threshold (Ṭ), as mentioned previously has significant influence on the rate of detection and therefore the rate of false detection and false negatives. In order to analyze which number of standard deviations to threshold (Ṭ) were suitable for various environments, a parameter sweep of Ṭ is performed in a new MATLAB program for many iterations of simulated environments to determine which values for Ṭ can result in the lowest false detection rate. The results are recorded in the table below.

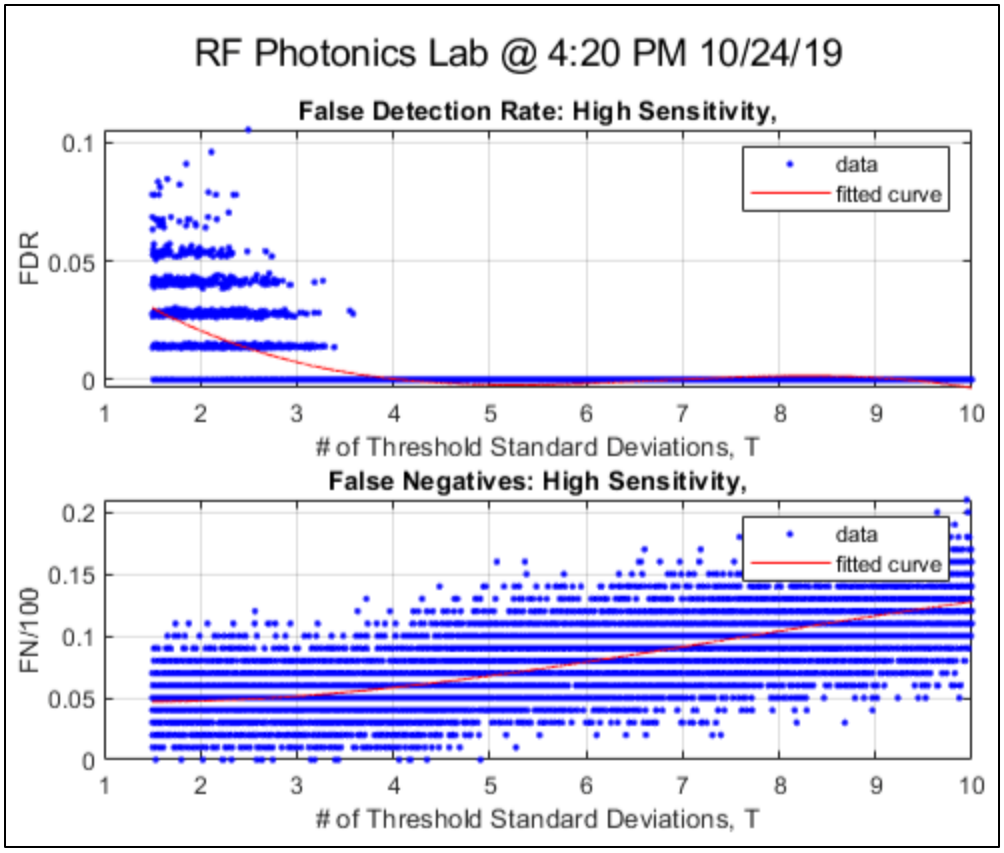

Figure 19. False Detection Rate vs. Threshold Ṭ

Table 4. Detection Rates from Environments

Data Environment	Threshold Sensitivity Setting	Ṭ	Average False Detection Rate	Average False Negative Rate
RF Photonics Lab	High	3.5	0.0018	0.052
RF Photonics Lab	Medium	24.5	0.0058	0.094
RF Photonics Lab	Low	40	0.041	0.064
The Marketplace	High	2	0.00	0.21
The Marketplace	Medium	5.75	0.00	0.12
The Marketplace	Low	11.5	0.01	0.22
Campus Center	High	4	0.00	0.39
Campus Center	Medium	5.5	0.00	0.39
Campus Center	Low	8	0.00	0.23
Wendy's	High	3.75	0.00	0.63
Wendy's	Medium	4.5	0.00	0.43
Wendy's	Low	8.5	0.00	0.24
SENG/On the Go	High	3.25	0.00	0.25
SENG/On the Go	Medium	6	0.00	0.32
SENG/On the Go	Low	9	0.00	0.24
Senior Design Lab	High	4	0.006	0.03

Senior Design Lab	Medium	30	0.003	0.10
Senior Design Lab	Low	55	0.02	0.04

A challenge is posed then, since numbers of standard deviations to threshold Ţ are at varying levels for optimization, how is the most appropriate number, Ţ selected in order to produce the best performance? There is an inevitable tradeoff between performance in quieter areas with lower standard deviations and performance in areas with high volume levels and higher standard deviations. Once a threshold number is chosen for each setting, there will need to be a method of analyzing the system performance. While the False Positive Rate or False Detection Rate is more important than the False Negative Rate, it would not suffice for the False Detection Rate to be zero, but the False Negative Rate is 100%; this case would render the device useless. One method of analyzing both qualities is the receiver operator characteristic plot.

6.4.4 – Receiver Operator Characteristic

A receiver operating characteristic describes how a system behaves with respect to its false positive (or false detection) rate and true positive (or true interrupt detection) rate. Depending on the type of system, a false negative may be more desirable than a false positive. In a choice between a false negative and a false positive, the Acoustic Awareness Enabler prefers a higher false negative rate over a higher false positive rate. This means that it is preferred that it is preferable that the AAE will not stop audio throughput even when it should have, as opposed to the case where it stops audio throughput when it shouldn't have. The receiver operator characteristic may be shown as a scatter plot with the false detection rate on the x-axis and the true positive rate on the y-axis. Each point indicates a trial in which both the false positive and true positive rates were calculated. As demonstrated previously, it is possible to perform a simulation of the process of the AAE and identify the false positive and false negative rates. Therefore, the operator characteristic can be known.

As discussed previously, the false detection rate and false negative rate are a function of the number of standard deviations to each threshold level (Ţ). This figure can be altered to produce a more desirable receiver operator characteristic. There are two approaches of accomplishing this, however.

For one, the number of standard deviations to the threshold levels (Ţ) can be optimized separately for the environments available. This approach was considered less in the early development stage, as it was considered complicated to accomplish and the potential benefits were not explored for this reason. Simulations at the later stages of development also demonstrated how particular the system may be and how much of an impact the number of standard deviations to threshold (Ţ) may have on the false detection rate, which should be 1% or less. Although it may complicate the problem, the results may be improved as it would allow the AAE to have better performance in each environment. One method of allowing this type of optimization for each environment would be to allow the number of standard deviations to the threshold (Ţ) to be a function of the ambient average volume level and standard deviation, since these are the main differentiating qualities of the environments. A piecewise function or if-then statement implementation could be problematic, as it would require testing of scenarios where the environment is categorized on the boundary of each if-then statement and prone to fluctuating between identified environments. Therefore, it would be proposed that the number of standard deviations to the threshold (Ţ) would have to be continuous and either a linear, quadratic or higher-order equation. Also note that, T being three separate solutions for

each threshold settings, there would be three equations for each sensitivity setting. Further, it may require a larger selection of environment samplings to produce an accurate quadratic or higher order Ţ function.

The other approach is a compromise, to select a static value for Ţ that will benefit some environments in terms of the Receiver Operator Characteristic and produce sub-optimal results in others. It is also the case that some environments are expected to be more likely scenarios for the use of the Acoustic Awareness Enabler. A weighting system can be used to determine a preferable performance for the more important locations. The weighting factor multiplies the thresholds and is then divided to produce a weighted result:

Table 5. Weighted Calculation for Threshold

Location	Importance Weighting Factor	High Sensitivity threshold	Medium Sens. Threshold	Low Sensitivity Threshold
Senior Design Lab	5	4	30	35
RF Photonics Lab	5	3.5	24.5	40
Campus Center	3	4	5.5	8
Wendy's	3	3.75	4.5	8.5
The Marketplace	2	2	5.75	11.5
SENG/On the Go	1	3.25	6	9
Total	19	**3.57**	**16.84**	**24.02**

Receiver Operator Characteristics are now plotted for all environments to assess their performance. This approach is to be implemented with a final product Acoustic Awareness Enabler to assess its performance in a variety of scenarios. The program in Appendix 1234567789 may be used to analyze further data collected using a final product Acoustic Awareness Enabler. The theoretical performance using the weighted sums is summarized as follows.

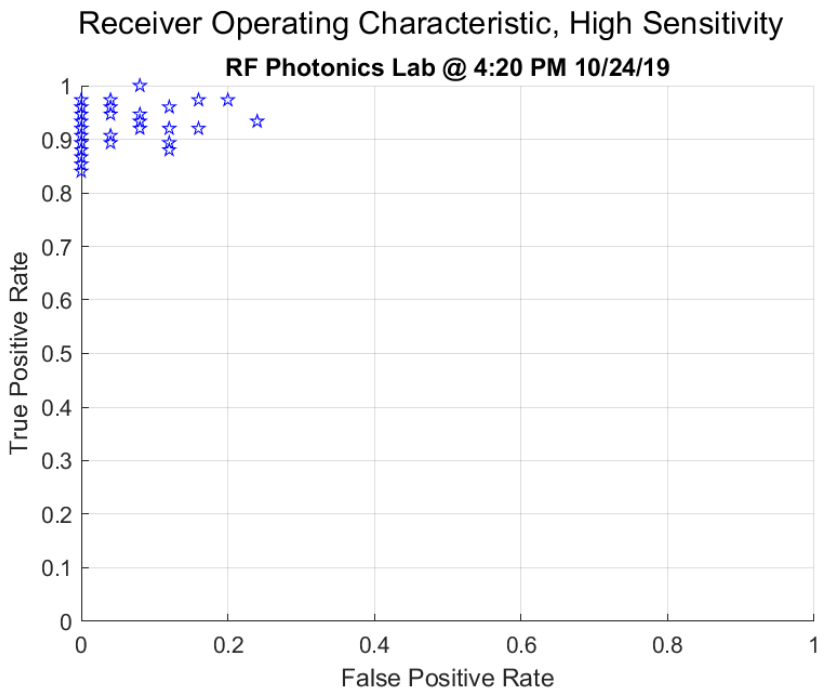

Figure 20. Receiver Operating Characteristic - RF Photonics Lab

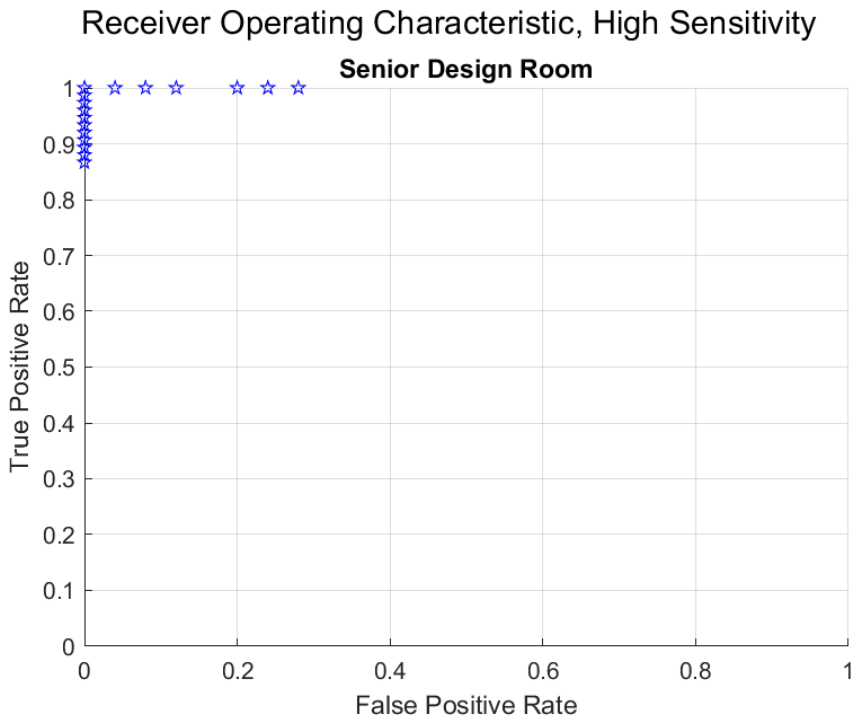

Figure 21. Receiver Operating Characteristic - Senior Design Room

The Senior Design Room and the RF Photonics Lab locations were deemed as more likely scenarios where a user is expected to employ the Acoustic Awareness Enabler, since they are more like a quiet home or office environment. Performance in these locations was reasonably good, with a great majority of false detection rates over many iterations of the program below 1%. If it was found that even better device performance is sought in these locations, increasing the thresholds (Ţ) would be a simple way to achieve that performance increase in quieter, low fluctuating environments. The values are set lower for the consideration of device performance in other environments.

We propose that these tests be performed using an integrated. It would be important to ensure that the microphone we are using behaves as the sound meter, for instance. Also, worth mentioning is how the number of data point and averaging of the microcontroller may influence accuracy. The data collected currently is comprised of data points taken at random times. This contrasts with a simulation of a continuous stream of data collected from an environment, which is what the microcontroller can do. We also propose that improvement of performance be focused on a few likely environments of use, simulating office and home environments. Following the same procedure of identifying what levels a true interrupt may be in an environment will be key in determining the false detection rate of the device.

6.5 Subsystem 5 – Enclosure and Form Factor

6.5.1 – Enclosure

Due to constraints brought upon by the COVID-19 pandemic, the project did not make it to the point of utilizing an actual enclosure to house the final circuit board and all the peripherals. Instead a mock enclosure was made to show how components were planned to be placed if the project ever reached that point. Side A and B both have a panel mount 3.5mm jack installed for the audio input and output, respectively. As a note, the red square on Side A is where a micro-usb port would have been located to charge the internal battery. Side C consists of a SPST push-button switch to turn the Acoustic Awareness Enabler on or off and a momentary push-button switch to reset the device. Side D has three momentary pushbuttons to select the appropriate sensitivity level for the environment that the user is in.

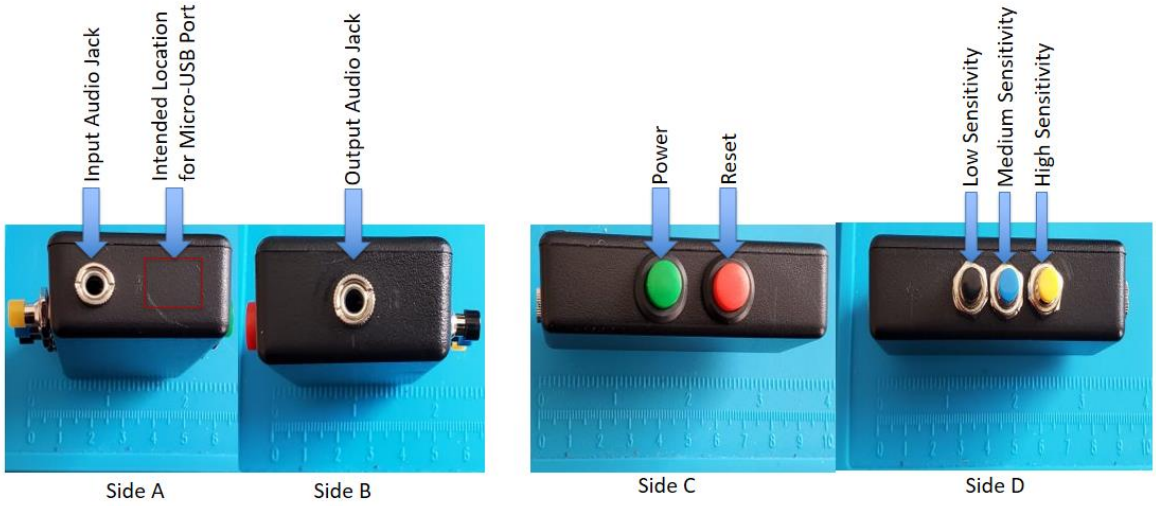

Figure 22 - Mock Enclosure Design

6.5.2 – Printed Circuit Board

As a result of the constraints which resulted from the COVID-19 pandemic, less emphasis was made on the final product integration. Furthermore, access to lab equipment, essential to producing a final approved printed circuit board layout was largely restricted. Electronics testing that would be necessary for a timely printed circuit board was delayed. Despite the issues that prevented the team from developing a printed circuit board, we have demonstrated that the printed circuit board is manufacturable in the constraint size of 3”x2.5”. A test layout (figure below) was developed on a board layout of the maximum size constraint to prove that this constraint is not an issue for the printed circuit board.

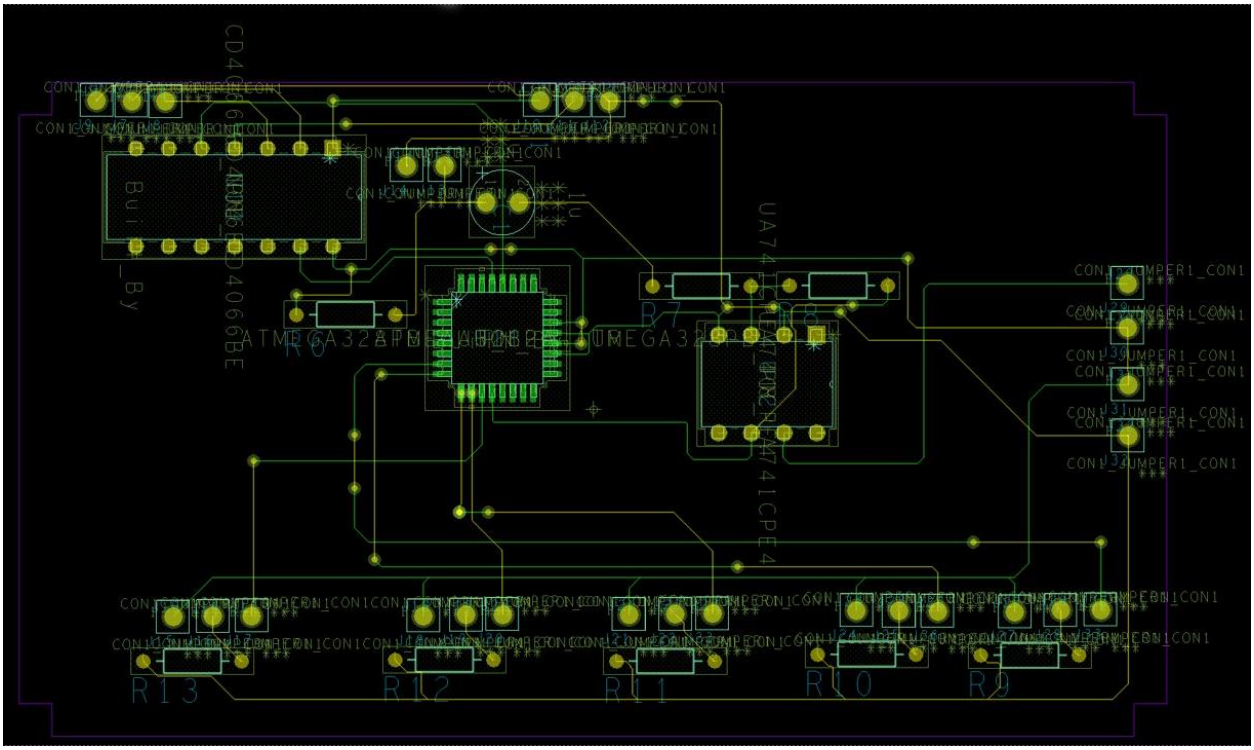

Figure 23. Initial PCB for Size Constraint Test

For a final printed circuit board, the following schematic should be used, were a printed circuit board to be manufactured. This schematic which includes pin payouts is proposed based on computer simulation and limited electronics testing. It is advisable that the schematic be tested before building a printed circuit board.

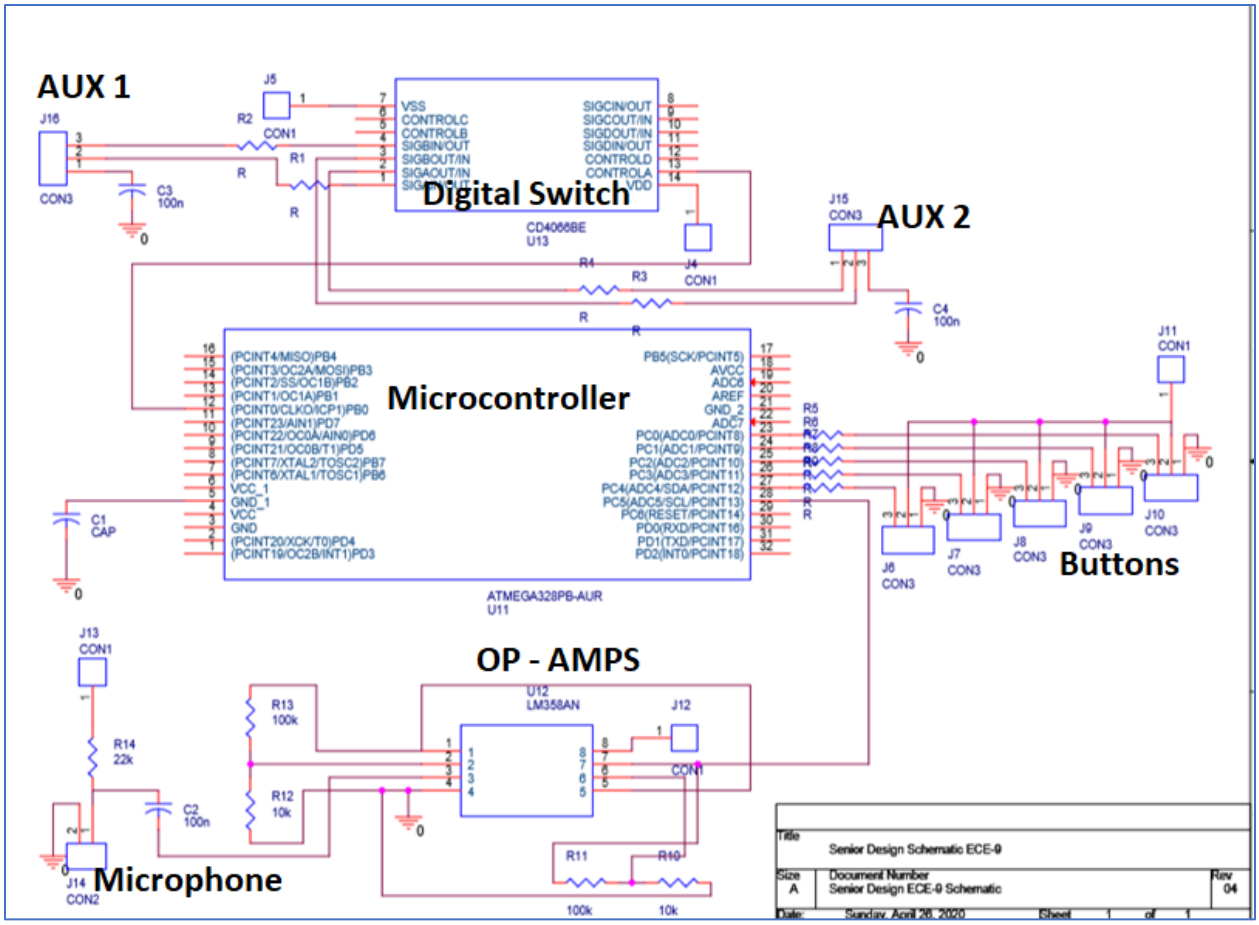

Figure 24. Proposed Schematic Layout for PCB

6.6 Subsystem 6 Power Supply

The power supply design has not been finalized and that would depend on two factors, the physical size of the device and how much current the whole system requires total. Early in the project the PRT-13854 battery was selected (3.7V 850 mAH) although it takes two of them in series to reach or exceed 5V. One alternative there would be to run the ATmega328PB at 3.7V this way two batteries could be set up in parallel. In series, two batteries amount to 7.4V 850 mAH and in parallel it's 3.7V 1700 mAh, the latter giving the system double the battery life.

To determine the power supply needed for the device the total current of the whole system must be measured. The whole system's total current can be measured by connecting a multimeter between the power supply and the high voltage rail that components such sa the CD4066BE and the buttons. Due to unavailability of lab equipment the total current of the system could not be measured which puts a hold on choosing the power supply.

7. Test Plan and Results

7.1 VCRM (Verification Cross Reference Matrix)

The Verification Cross Reference Matrix (VCRM) illustrates the specific test plans which satisfy the Customer and Engineering Requirements. The table categorizes each requirement as an inspection, analysis, demonstration or test. As stated in the Customer Requirements and Engineering Requirements section, some of the requirements were not demonstrable.

Table 6. Verification Cross Reference Matrix

Cus. Req. #	Eng. Req. #	Engineering Requirement Description	Related Test Plan Codes	Verification				Where met in design
				I	A	D	T	
1	1.1	8 hours of power supplied to device.	ET_09, ET_13, ET_15			X	X	Power supplied by battery.
2	2.1	Audio Source Input 3.5mm	ET_04, NT_02	X		X		Interface of enclosure and electronics.
	2.2	Audio Source Output 3.5mm	ET_04, NT_02	X		X		
	2.3	I/O interface – digital switch	ET_03, ET_07,	X			X	Digital Switch found in electronics and controlled by µC.
3	3.1	Processing includes average and standard deviation calculations to determine threshold	PT_01, PT_02, PT_04		X	X		Process first validated in external software then programmed in µC.
	3.2	Three separate sensitivity buttons	ET_05, PT_01		X		X	Interface of enclosure and electronics.
4	4.1	Digital switches turn off audio throughput when interruption threshold is exceeded.	ET_10, ET_11, ET_12				X	µC reads acoustic signals and alerts when threshold exceeding, sending a signal to the digital switch.
5	5.1	Digital signal processing is invoked to determine the presence of an outlier in a set of acoustic level readings.	PT_04, CT_01, CT_04, CT_05			X		Process first validated in external software then programmed in µC.
	5.2	Lowest Ambient Noise of 30 dB based on ANSI standards	ET_01, PT_01		X		X	Microphone meeting requirement is selected, signal processing design in accordance.
	5.3	Must have 100 Hz Sampling frequency to detect averages in sound (not reconstructing signal)	CT_05			X		At the interface of the ADC on µC and analog input from microphone and analog electronics.
	5.4	Need a sound level detector	ET_01				X	Microphone is used.
6	6.1	Physical reset button will be present on Acoustic Awareness Enabler to reset acoustic environment data.	ET_05, NT_02			X	X	Button pressed sends signal to µC to turn switch on after being off. If switch is on, clear processing data.
	6.2	Use of switches to resume audio after reset button has been pressed.	ET_10				X	Button pressed sends signal to µC to turn switch on after being off.

	6.3	Button will recalibrate device and will delay further interrupts for 20 seconds while gathering data	ET_11, PT_04			X	X	If switch is on, clear processing data. Set 20 second pause on interrupts.
7	7.0	Audio (auxiliary) Cable Included with Enclosure	ET_04	X				Include test auxiliary cable.

7.2 Test Cases and Test Results

The flowchart below outlines all test plans that were proposed at the beginning of the semester. The arrows indicate prerequisite plans for subsequent test plans. The test plans in this flow chart are color-coded by subsystem. These test plans have mostly remained the same since the test plan and test report, although some test plans were performed under less than ideal conditions as a result of lab availability. A few test plan prerequisites were forgone as a result of the pandemic constraints, which resulted in a "partial completion" condition. For instance, although a test plan such as PT_07 technically required the completion of ET_13, certain aspects of the test plan were attempted and in some cases partially completed. The next flowchart is color coded according to the level of completion. Solid green boxes indicate test plans that were considered a "Pass." Light green boxes indicate that the test plan was partially completed, meaning that either only some parts of the test were attempted or they were performed without first completing prerequisites, which were unable to complete due to the pandemic constraints. Finally, test boxes colored black were not completed or were unable to be attempted.

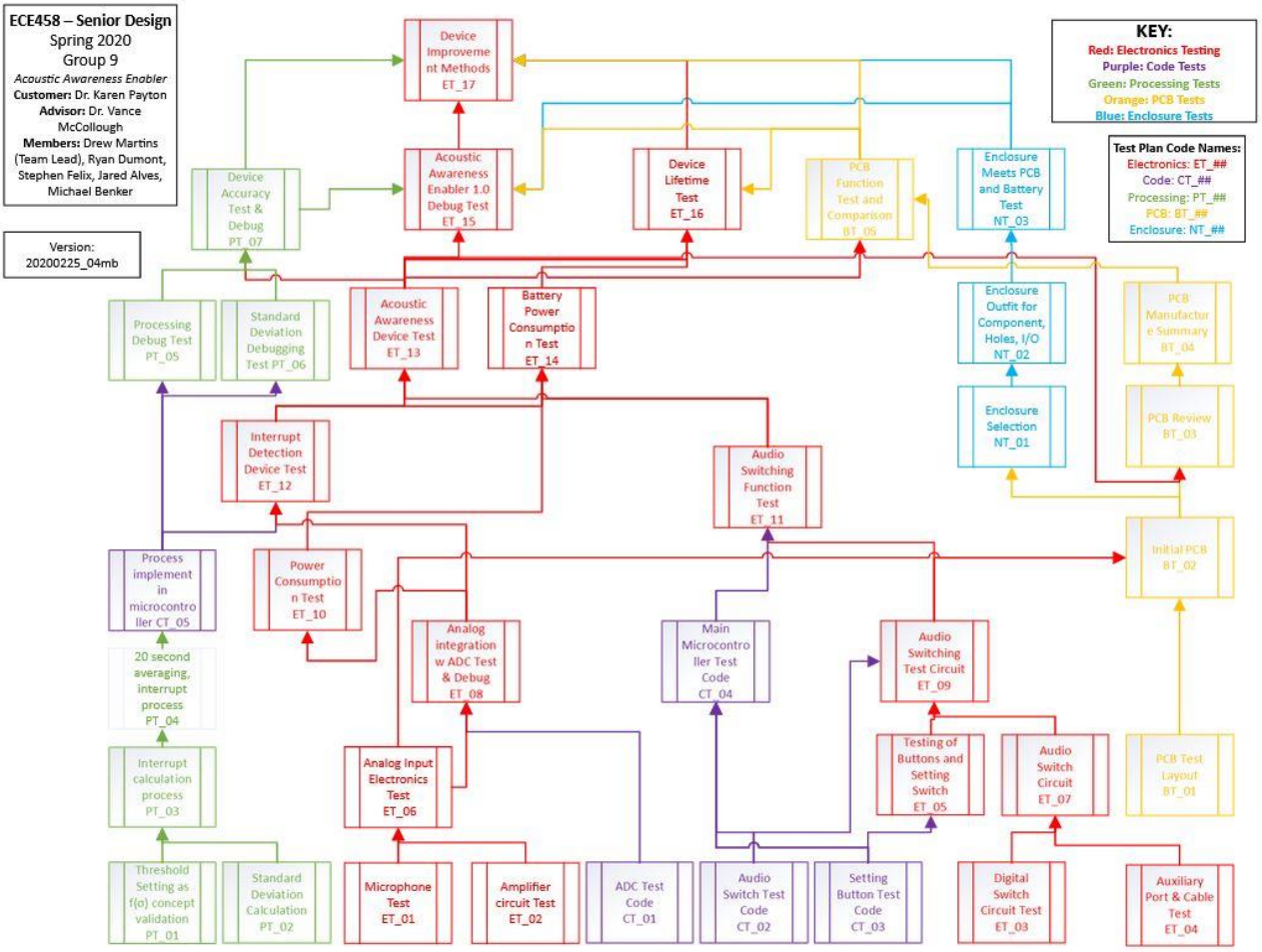

Figure 25. Original Test Plan Flowchart

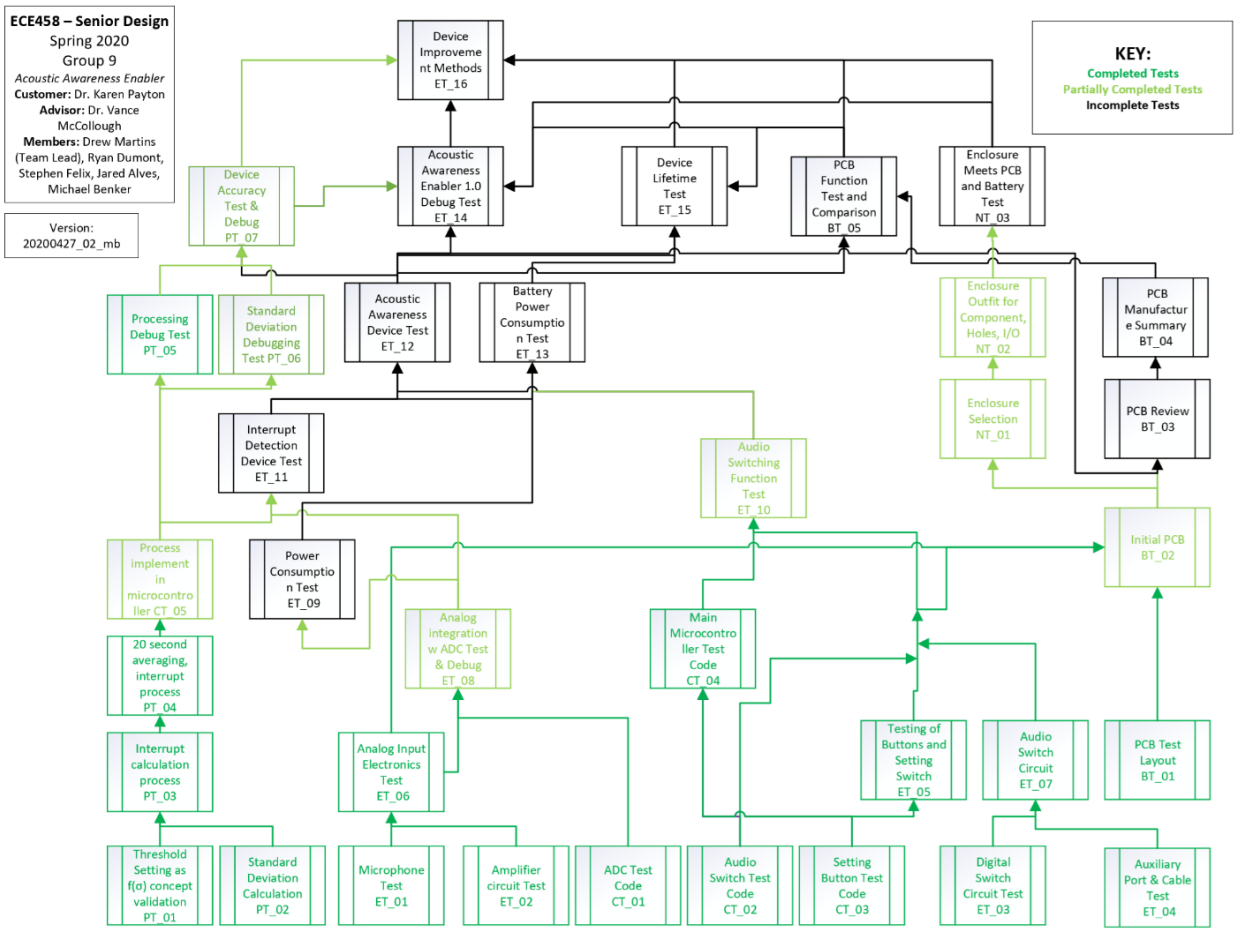

Figure 26. Test Plan Flowchart with Completion Status

7.3 Test Summary

The following table outlines test plans and their corresponding completion status; pass, partial completion, not performed (fail). Test plans that were impacted by constraints of lab equipment are noted.

Table 7. Test Summary

Eng. Req. ID	Test Code	Test Name	Pass/ Fail	Comments
5.2/5.4	ET_01	Microphone Test	Pass	The microphone used about 28-35mV of electricity depending on Decibels of noise heard.
5.4	ET_02	Amplifier Circuit Test	Pass	The amplifier successfully amplified a signal that was at peak 17.6 mV to 197 mV.

4.1	ET_03	Digital Switch Test	Pass	Disconnecting the switch from vdd still left the switch on until it was connected to ground.
2.1,2.2,5.1,6.3,7.0	ET_04	Auxiliary Port and Cable Test	Pass	The headphone jack fit in the Aux port and successfully was able to send audio from one aux to the other.
3.2, 6.1	ET_05	Testing of Buttons and Switch	Pass	The buttons and switch worked perfectly.
5.4	ET_06	Analog Input Electronics Test	Pass	Combining the microphone and amp allowed for a increase in the voltage being read.
2.1, 2.2, 2.3, 4.1	ET_07	Audio Switch Test	Pass	*Due to COVID-19, a DC power source was inaccessible, the workaround was to add 2x 3.7V batteries in series and a voltage divider to create a 5V DC source.
3.1, 5.1, 5.2, 5.3, 5.4	ET_08	Analog Integration w ADC Test and Debug	Partial Completion	Constricted by lab availability due to COVID-19
1.1	ET_09	Power Consumption Test	Not Performed	Constricted by lab availability due to COVID-19
3.2	ET_10	Audio Switching Function	Partial Completion	Constricted by lab availability due to COVID-19
3.1, 4.1, 5.1, 5.4	ET_11	Interrupt Detection Device	Not Performed	Constricted by lab availability due to COVID-19
	ET_12	Acoustic Awareness Device	Not Performed	Test prerequisites incomplete.
1.1	ET_13	Battery Power Consumption	Not Performed	Test prerequisites incomplete.
All	ET_14	Acoustic Awareness Enabler 1.0 Debug Test	Not Performed	Test prerequisites incomplete.
1.1	ET_15	Device Lifetime Test	Not Performed	Test prerequisites incomplete.
	ET_16	Device Improvement Methods	Not Performed	Test prerequisites incomplete.
5.1	PT_01	Threshold Setting as f(σ) Concept Validation	Pass	This data validates the concept of threshold as a function of standard deviation. Further tests improve accuracy of standard deviation.
5.1,5.2	PT_02	Standard Deviation Calculation	Pass	The plot shows that there is a general positive correlation between ambient level and standard deviation.
5.1,5.2,5.3	PT_03	Interrupt Calculation Process	Pass	
5.1,5.2,5.3	PT_04	20 Second Averaging, Interrupt Process	Pass	The program “interrupt.cs” only provides a simple initial test, since loud instantaneous noises tend to trigger the interrupt to occur, which is undesirable. This is because the program takes a two second average by taking only a minimum and maximum sound level and averaging those two discrete levels. Highest threshold is difficult to trigger, especially when standard deviation is high (>1).
5.1	PT_05	Processing Debug Test	Pass	Within the program at least, implementing a 30 second average vs a 3 second average did not improve the averaging process. An accurate standard deviation or average could not be obtained (See Figure 19). Recursive

				averaging does not change average calculation in a noticeable way. The iterative averaging method will be used. The timing yields a wide variance of ticks (between 4000 and 6000 ticks but can be as low as 500). This is done using the Stopwatch class in C#
5.1	PT_06	Standard Deviation Debugging	Partial Completion	Constricted by lab availability due to COVID-19
5.1,5.4	PT_07	Device Accuracy Test & Debug	Partial Completion	Constricted by lab availability due to COVID-19
5.1	CT_01	ADC Test Code	Pass	See Figure 32
2.3, 4.1, 6.2	CT_02	Audio Switch Test Code	Pass	The Auxiliary Ports (Input and Output) both needed a 1kΩ resistor to GND.
3.2, 6.1, 6.3	CT_03	Setting Button Test Code	Pass	The code was tweaked in order to get the on/off button operating.
2.3, 3.2, 4.1, 6.1, 6.2, 6.3	CT_04	Main µC Test Code	Pass	No Comment
6.2, 6.3	CT_05	Process Implement in µC	Partial Completion	Partial Completion
Constraint of maximum size: 3.5” x 2”	BT_01	PCB Test Layout	Pass	Test proves that electronics and PCB can be fabricated under maximum constraint size.
6.1,6.2,6.3, Constraint of maximum size: 3.5” x 2”	BT_02	Initial PCB	Partial Completion	Partial Completion: PCB not being ordered. Schematic for layout completed.
6.1,6.2,6.3, Constraint of maximum size: 3.5” x 2”	BT_03	PCB Review	Not Performed	Test prerequisites incomplete.
7.0, Constraint size	BT_04	PCB Manufacture Summary	Not Performed	Test prerequisites incomplete.
6.1, 6.2, 6.3	BT_05	PCB Function Test and Comparison	Not Performed	Test prerequisites incomplete.
7.0, customer req. 1	NT_01	Enclosure Selection	Pass with Failure criteria	Partial Completion with both Pass and Fail designations. Enclosure has been ordered despite test prerequisite completions.
1.1	NT_02	Enclosure Outfit for Components, I/O	Partial Completion	Test prerequisites incomplete.
1.1	NT_03	Enclosure Integration with Battery and PCB	Not performed	Test prerequisites incomplete.

8. Risk Discussion

Present Risks within the Project:

Risk	Eng. Req.	Likelihood	Consequence	Risk Score	Possible Method to Mitigtae Risk

Complications with accuracy of microphone communication with ADC (resolution issue).	5.1	Possible	Major	High	Proper amplification of mic signal testing mic/ADC.
Complications or inconsistency in achieving <1% false alarm rate.	5.1	Unlikely	Major	High	Troubleshooting of both hardware and software.
Complications in manufacture of PCB.	Not Req.	Possible	Moderate	High	Have professionals review PCB design and get it out for manufacture early.
Battery Life is Insufficient (< 8 Hours)	1.1	Possible	Moderate	High	Pre-calculations, Actual battery life tests.
Complications in creating a proper plastic/physical enclosure.	Not Req.	Unlikely	Moderate	Med.	Use pre-made case and modify.
Interruption Threshold will never be exceeded in an environment with high fluctuation (high standard deviation).	5.1	Possible	Minor	Med.	Possible limitations of device applications may alter threshold setting to operate differently in these settings (e.g. max ambient standard deviation).
Failure to get components in a timely manner.	All	Rare	Insignificant	Low	Most parts in possession, no rare parts.

Due to COVID-19, none of these risks have been fully solved, although there are plans to mitigate these risks that would theoretically work.

Risks	**Possible Method to Mitigate Risk**	**Methods taken to Mitigate Risk**
Complications with accuracy of microphone communication with ADC (resolution issue).	Proper amplification of mic signal testing mic/ADC.	Test Plans for constructing an accurate ADC reading and microphone measuring (incomplete).
Complications or inconsistency in achieving <1% false alarm rate.	Troubleshooting of both hardware and software.	Signal processing of recorded data performed, and receiver operating characteristic plotted for environments. Lab constraints hampered ability to test signal processing theory with fully integrated design.
Complications in manufacture of PCB.	Have professionals review PCB design and get it out for manufacture early.	PCB layout not sent to manufacture due to pandemic constraints.

Battery Life is Insufficient (< 8 Hours)	Pre-calculations, Actual battery life tests.	Battery life tests not performed due to lack of lab availability.
Complications in creating a proper plastic/physical enclosure.	Use pre-made case and modify.	Pre-made case ordered and modified.
Interruption Threshold will never be exceeded in an environment with high fluctuation (high standard deviation).	Possible limitations of device applications may alter threshold setting to operate differently in these settings (e.g. max ambient standard deviation).	Concluded to be a non-issue in a trade off between interrupt detection accuracy and false detection rates as demonstrated by signal processing.
Failure to get components in a timely manner.	Most parts in possession, no rare parts.	Not proven to have been an issue.

9. Plan, Schedule, and Costs

9.1 Plan and Schedule – Final Update

The schedule is outlined below. The dark gray represents actual time spent on tasks whereas the light gray represents what was planned throughout the semester. The actual and planned hours are outlined in the right- and left-hand columns respectively.

Task ID	Task Description	Engineering Requirements	Task Lead	Other Members	Planned Total Hours	1/25	2/1	2/8	2/15	2/22	2/29	3/7	3/14	3/22	3/28	4/5	4/11	4/18	4/25	5/2	5/9	Actual Total Hours
Assignments (A)																						
A1	Plan & Schedule		Drew	All	15			Due 2/4														20
A2	Test Plans		Drew	All	30						Due 2/27											
A3	Test Report		Drew	All	40										Due 3/27							
A4	Final Report		Drew	All	60															Due 4/28		40
A5	Final Presentation		Drew	All	60															Due 4/30		30
Tasks (T)																						
T1	Electronics Design				90																	
T1-1	Finalize Major Component Choices	1.1, 2.3, 3.3, 4.1 & 6.1	Drew	Ryan, Michael	25	5 4	10	10														4
T1-2	Analog Microphone Design	5.2,4 & 6.2	Ryan	Drew	25	5 4	10 5	10 4	 3.5													16.5
T1-3	Microphone to ADC Design	5.3,4 & 6.2	Ryan	Drew	15			5 4	5 3.5	5 3	 3	 6										24.5
T1-4	Auxillary Switch Design	2.1-3 & 6.2	Stephen	Drew	10	5 2	5 4.5															6.5
T1-5	Battery Integration	1.1	Drew	Ryan, Stephen	15				5	5	5											0
T2	Software and DSP				170																	
T2-1	Initial Microcontroller Programming	3.1 & 5.1	Drew	Jared, Michael	30	5 4	5 8	5	5 15	5	5											27
T2-2	Signal Processing Code	5.1	Jared	Michael	20	5 1	5 10	5 3	5													14
T2-3	Processing Code Debug	5.1	Drew	Michael, Jared	40						5	5 7	5	5 8	5 4	5	5	5 10				29
T2-4	Testing, Tuning other parameters	3.2, 4.1, 5.1, 6	Jared	Ryan, Michael	25									3		5	5	5	5	5		3
T2-5	Debugging of Microcontroller Prog.	3.2, 4.1, 5.1, 6	Drew	Jared, Michael	35							5 5	5 6	5 5	5 4	5 5	5	5 2	4			31
T2-6	Full Integration Code Debugging	3.2, 4.1, 5.1, 6	Drew	Michael, Jared	20											5	5	5	5 6			6

T3	Enclosure				21	
T3-1	Enclosure Size to Match PCB	1.1	Ryan	Michael	6	2
T3-2	Enclosure fitted PCB, Battery & buttons	1.1, 3.2, 6	Ryan	Drew	15	0
T4	Printed Circuit Board				76	
T4-1	Initial PCB Corrections	2, 6	Michael	---	25	6
T4-2	PCB minimum size test	2, 6	Michael	Drew, Ryan	25	11
T4-3	PCB design completion	2, 6	Michael	Drew	20	0
T4-4	PCB out for manufacture	2, 6	Michael	Ryan	6	0
T5	Test and Demonstrations				79	
T5-1	Analog Microphone Testing	1, 2, 5, and 6	Drew	Ryan	20	16
T5-2	Microphone to ADC Testing	1, 2, 5, and 6	Drew	Ryan	20	7
T5-3	Auxiliary Switch Testing	1, 2, 5, and 6	Drew	Stephen	30	10
T5-4	Testing Microphone + ADC Subsystems	1, 2, 5, and 6	Ryan	Stephen, Drew	9	0
T5-5	Testing Buttons + Switches Subsystems	1, 2, 5, and 6	Stephen	Ryan, Drew	9	10
T5-6	Testing Buttons, Switch, Microphone Subsystem	1, 2, 5, and 6	Drew	Ryan, Stephen	9	19
T5-7	PCB Testing	1, 2, 5, and 6	Drew	Ryan, Michael	8	0
T5-8	Practicing for Final Presentation		Drew	All	15	15
T6	Reviews				10	
T6-1	Compilation of Ethical Issues		Jared	Ryan	10	2
T6-2	Test Plans	1, 2, 5, and 6	Michael	ALL	14	61
T6-2	Test Reports	1, 2, 5, and 6	Michael	Jared	28	17

Figure 27 Plan and Schedule

9.2 Actual Hours vs Planned Hours

The Plan and schedule section demonstrates that several processes took for less time than what was expected. This was partially because many activities were interrupted by the COVID-19 pandemic. Most of the electronics and enclosure related tasks were discontinued due to lacking access to university facilities. It was also due to inaccurate planning in some instances. Some sections were not attempted because the prior tests could not be completed. This is outlined in the Test Summary. The necessity of scheduling throughout both last semester and the present semester provided insight into the importance of keeping a schedule as well as discrepancies that can occur as a result of doing so.

9.3 Cost Summary

The cost summary is outlined below. The budget of $500 was not exceeded. Costs such as the printed circuit board were not done, which reduced the total cost of the project.

Table 8. Cost Summary

Part Number	Seller	Price	Qty.	Total Price
458-1542-ND	Digikey	$1.11	5	$5.55
458-1536-ND	Digikey	$1.22	5	$6.10
ATMEGA328PB-XMINI-PD	Digikey	$9.26	2	$18.52
296-2061-5-ND	Digikey	$0.405	10	$4.05
DS18030-010+-ND	Digikey	$4.55	4	$18.20
1568-1495-ND	Digikey	$9.95	4	$39.80
Shipping Cost	Digikey	$8.99	1	$8.99
Tariff	-------	$0.41	1	$0.41
			Total	$101.62

10. Summary

The Acoustic Awareness Enabler is a device equipped to provide a user with a customizable filter for background noise when listening to continuous audio. Although many parts of the project could not be completed due to circumstance, the project will likely be resumed by a future group. Suggestions for future work would include the physical construction of a printed circuit board and device. It would also be helpful for subsequent teams to adjust

threshold levels if necessary. Threshold levels could be tailored to specific types of environments or could possibly cater to environments that are more realistic or expected for users of the device. Another area that could be expanded on would be doing spectral analysis on potential interrupts to filter based on frequencies. Certain characteristic sounds could be filtered out, such as unnecessary background noises which differ from human voices in terms of frequency spectrum. If a physical device could be completed, the customer could greatly benefit from having a convenient device with realistic sensitivity settings for attenuating outside audio. The device would be beneficial mostly on a personal level, for people who often use media devices on the go or around the home and would appreciate automatic interruptions when these are convenient.

11. Lessons Learned / Impacts due to COVID-19

COVID-19 presented many random variables into the equation which prevented the proper completion of the project. A physical device could not be obtained due to the circumstances. The final few months of the project proved to be quite a challenge. However once the team grew accustomed to remote meetings, it became second nature and a theoretical approach was taken.

Aside from COVID-19, many lessons were learned about teamwork and project planning. For many members of the team, this was the first time for completely independent group work. In the past, many group projects or assignments were highly structured and clear answers were obtained upon completion. The Senior Design project presented new challenges by forcing students to come up with answers independently, which provided appropriate preparation for the future workforce endeavors. It was also the first time that keeping a project schedule was required, which is very important for real world projects as well.

12. Documentation

Appendix A. C# Threshold Program based on Physically Collected Data

```
using System;
using System.Data;
using Excel = Microsoft.Office.Interop.Excel;
using ClosedXML.Excel;
using System.Linq;
```

```
//In order to use ClosedXML namespace, you must download nuget from nuget website
//To enable it, go to Tools -> Nuget package manager -> Manage nuget packages for
solution


namespace AA_Enabler

{
    class Program
    {

        /*
         * Select environment from the environments that we have data for.
         * Loads selected environment. Choose 10 random numbers from the list of 30
numbers from the column for Ambient data.
         * Display the 10 random numbers. We currently do this, but instead of typing in
10 numbers, we want the environment to be simulated in some level as though the last 20
seconds were in that environment.
         * Calculate average and standard deviation for the environment and display
threshold
         * Ask user to submit a new 2 second average for the environment.
         * Calculate if that new 2 second average is an interrupt for each threshold
setting
         *
         * average of each environments - real interrupt level
         *
         *
         **/

        static void Main(string[] args)
        {
            Console.WriteLine("Select Environment by Environment No (1-6):");
            string selection = Console.ReadLine();

            int select = int.Parse(selection);
            DataTable _dt = new DataTable(); //DataTable Stores values
            DataSet _ds = new DataSet(); //DataSet is used to display values

            string Book1 = @"C:\Users\Jared\source\repos\AA_Enabler\Book1.xlsx";
            string Book2 = @"C:\Users\Jared\source\repos\AA_Enabler\Book2.xlsx";  //Load
file from computer location, change as necessary based on where the file is
            string Book3 = @"C:\Users\Jared\source\repos\AA_Enabler\Book3.xlsx";
            string Book4 = @"C:\Users\Jared\source\repos\AA_Enabler\Book4.xlsx";
            string Book5 = @"C:\Users\Jared\source\repos\AA_Enabler\Book5.xlsx";
            string Book6 = @"C:\Users\Jared\source\repos\AA_Enabler\Book6.xlsx";


            switch (select)  //scan user input and select appropriate file
            {
                case 1:
                _dt = ImportSheet(Book1);
                    break;


                case 2:
                    _dt = ImportSheet(Book2);
                    break;
```

```
                case 3:
                    _dt = ImportSheet(Book3);
                    break;

                case 4:
                    _dt = ImportSheet(Book4);
                    break;

                case 5:
                    _dt = ImportSheet(Book5);
                    break;

                case 6:
                    _dt = ImportSheet(Book6);
                    break;
            }

            DataTableReader reader = _ds.CreateDataReader(_dt);
            Console.WriteLine("\nData for the set (Ambient, Low, Med, High):\n");
            PrintColumns(reader);

            //string[] str_col = new string[_dt.Rows.Count];
            double[] d_col_Ambient = new double[30];
            double[] d_col_Low = new double[30];
            double[] d_col_Med = new double[30];
            double[] d_col_High = new double[30];


            for (int index_r = 0; index_r < _dt.Rows.Count; index_r++)
            {
                int index_c = 0;

                d_col_Ambient[index_r] =
double.Parse(_dt.Rows[index_r][index_c].ToString());

            }

            for (int index_r = 0; index_r < _dt.Rows.Count; index_r++)
            {
                int index_c = 1;

                d_col_Low[index_r] = double.Parse(_dt.Rows[index_r][index_c].ToString());

            }

            for (int index_r = 0; index_r < _dt.Rows.Count; index_r++)
            {
                int index_c = 2;

                d_col_Med[index_r] = double.Parse(_dt.Rows[index_r][index_c].ToString());

            }

            for (int index_r = 0; index_r < _dt.Rows.Count; index_r++)
            {
                int index_c = 3;
```

```
                d_col_High[index_r] =
double.Parse(_dt.Rows[index_r][index_c].ToString());

            }

            //foreach (double q in d_col_Ambient)     //Test ambient double array
            //{
            //    Console.WriteLine(q);
            //}

            Console.WriteLine();

        double[] ten_rand_ambient = new double[10];

            ten_rand_ambient = ten_Random(d_col_Ambient);

            Console.WriteLine("Ten Random Numbers from Ambient Data Set");

            foreach (double i in ten_rand_ambient)     //Test random integer array of 10
            {
                Console.WriteLine(i);
            }

            Console.WriteLine();
            double ten_avg_ambient = ten_rand_ambient.Average();
            double std;


            Console.WriteLine("Select a Sensitivity Level (Low [1], Medium [2], High
[3])");
            string x = Console.ReadLine();
            int t = Int32.Parse(x);


            double sumOfSquaresOfDifferences = d_col_Ambient.Select(val => (val -
ten_avg_ambient) * (val - ten_avg_ambient)).Sum();
            std = Math.Sqrt(sumOfSquaresOfDifferences / d_col_Ambient.Length);

            if (std < 1)
            {

                ten_avg_ambient = ten_avg_ambient + 2;

            }

            double t1 = ten_avg_ambient + 1.5 * std;
            double t2 = ten_avg_ambient + 4 * std;
            double t3 = ten_avg_ambient + 7 * std;

            Console.WriteLine("The standard deviation is {0}\n", std);

            if (t == 3)
            {
                Console.WriteLine("The Instantaneous Threshold Level is: {0}", t1 );

            }
            else if (t == 2)
            {
```

```
            Console.WriteLine("The Instantaneous Threshold Level is: {0}", t2);
        }
        else
        {
            Console.WriteLine("The Instantaneous Threshold Level is: {0}", t3);
        }

        Console.WriteLine();


        while (true)
        {
            Console.WriteLine("Enter a two second average:");
            string str = Console.ReadLine();
            double two_avg = double.Parse(str);


            if (t == 3)
            {
                if (two_avg >= t1)
                {
                    Console.WriteLine("Interrupt has occurred");
                }
                else
                {
                    Console.WriteLine("No interrupt");
                }
            }
            if (t == 2)
            {
                if (two_avg >= t2)
                {
                    Console.WriteLine("Interrupt has occurred");
                }
                else
                {
                    Console.WriteLine("No interrupt");
                }
            }
            if (t == 1)
            {
                if (two_avg >= t3)
                {
                    Console.WriteLine("Interrupt has occurred");
                }
                else
                {
                    Console.WriteLine("No interrupt");
                }
            }
        }
    }



    public static DataTable ImportSheet(string fileName)
```

```
        {
            var datatable = new DataTable();
            var workbook = new XLWorkbook(fileName);
            var xlWorksheet = workbook.Worksheet(1);
            var range = xlWorksheet.Range(xlWorksheet.FirstCellUsed(),
xlWorksheet.LastCellUsed());

            var col = range.ColumnCount();
            var row = range.RowCount();

            //if a datatable already exists, clear the existing table
            datatable.Clear();
            for (var i = 1; i <= col; i++)
            {
                var column = xlWorksheet.Cell(1, i);
                datatable.Columns.Add(column.Value.ToString());
            }

            var firstHeadRow = 0;
            foreach (var item in range.Rows())
            {
                if (firstHeadRow != 0)
                {
                    var array = new object[col];
                    for (var y = 1; y <= col; y++)
                    {
                        array[y - 1] = item.Cell(y).Value;
                    }

                    datatable.Rows.Add(array);
                }
                firstHeadRow++;
            }
            return datatable;
        }
        static void PrintColumns(DataTableReader reader)
        {
            // Loop through all the rows in the DataTableReader
            while (reader.Read())
            {
                for (int i = 0; i < reader.FieldCount; i++)
                {
                    Console.Write(reader[i] + " ");
                }
                Console.WriteLine();
            }
        }

        static double[] ten_Random(double[] array)
        {
            //Gets 10 random integers from an integer array
            Random rand = new Random();

            double[] ten_array = new double[10];

            for (int i = 0; i < 10; i++)
            {
                int r = rand.Next(array.Length - 1);
```

```
                ten_array[i] = array[r];
            }

            return ten_array;
        }
    }
}
```

Appendix B. False Detection Rate & Receiver Operating Characteristic MATLAB Program

```
%ECE458 - Senior Design
%Michael Benker
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%% FALSE DETECTION RATES %%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
clf;clear all; clc; close all;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
samples=100;

%Threshold Levels. Begin and End for sweeping capabilities
T_h_begin = 3.57;
T_h_end = 3.57;
T_m_begin = 16.84;
T_m_end = 16.84;
T_l_begin = 24.02;
T_l_end = 24.02;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%VARIABLES
Ave2sec = 0;    %2-second average
Data20sec = zeros(10,1);
history = zeros(100,1);

RI_med = zeros(100,1); %1 if real interrupt, 0 if false interrupt
RI_high = zeros(100,1);
RI_low = zeros(100,1);
ID_low = zeros(100,1);   %1 if interrupt detected, 0 if interrupt not
detected
ID_med = zeros(100,1);
ID_high = zeros(100,1);
TID_low = zeros(100,1);    %1 if ID = RI = 1, 0 otherwise
TID_med = zeros(100,1);
TID_high = zeros(100,1);
FD_low = zeros(100,1);     %1 if ID = 1 & RI = 0, 0 otherwise
FD_med = zeros(100,1);
FD_high = zeros(100,1);
FN_low = zeros(100,1);     %1 if ID = 0 & RI = 1, 0 otherwise
FN_med = zeros(100,1);
FN_high = zeros(100,1);
FDR_low = 0;    %probability of false interrupt given interrupt detection
```

```
FDR_med = 0;
FDR_high = 0;
FDR_low_array = zeros(samples,1);
FDR_med_array = zeros(samples,1);
FDR_high_array = zeros(samples,1);
T_high_array = zeros(samples,1);     %standard deviations for high
sensitivity setting
T_med_array = zeros(samples,1);      %standard deviations for medium
sensitivity setting
T_low_array = zeros(samples,1);     %standard deviations for low sensitivity
setting
FN_low_array = zeros(samples,1);
FN_med_array = zeros(samples,1);
FN_high_array = zeros(samples,1);
Opti_low = zeros(samples,1);
Opti_med = zeros(samples,1);
Opti_high = zeros(samples,1);
TID_low_Array = zeros(samples,1);
TID_med_Array = zeros(samples,1);
TID_high_Array = zeros(samples,1);

TPR_low_Array = zeros(samples,1);
TPR_med_Array = zeros(samples,1);
TPR_high_Array = zeros(samples,1);

%IMPORT DATA
SoundData1 = 'Book3.xlsx'; %Read excel file in folder
DataMat = zeros(30,4); %Predefine Data Matrix
Ambients = xlsread(SoundData1, 'A2:A31'); %Ambient 1st col
Quiets = xlsread(SoundData1, 'B2:B31'); %Quiet is 2nd col
Mediums = xlsread(SoundData1, 'C2:C31'); %Medium is 3rd col
Louds = xlsread(SoundData1, 'D2:D31'); %Loud is 4th col
[k,DataLoc] = xlsread(SoundData1, 'E1:E1');

%Define past 20 seconds (ambients)


for s=1:samples
    T_high=T_h_begin+s*(T_h_end-T_h_begin)/samples;
    T_med=T_m_begin+s*(T_m_end-T_m_begin)/samples;
    T_low=T_l_begin+s*(T_l_end-T_l_begin)/samples;

    T_high_array(s,1)=T_high;
    T_med_array(s,1)=T_med;
    T_low_array(s,1)=T_low;

    RI_med = zeros(100,1); %1 if real interrupt, 0 if false interrupt
    RI_high = zeros(100,1);
    RI_low = zeros(100,1);
    ID_low = zeros(100,1);   %1 if interrupt detected, 0 if interrupt not
detected
    ID_med = zeros(100,1);
    ID_high = zeros(100,1);
    TID_low = zeros(100,1);    %1 if ID = RI = 1, 0 otherwise
    TID_med = zeros(100,1);
```

```
    TID_high = zeros(100,1);
    FD_low = zeros(100,1);      %1 if ID = 1 & RI = 0, 0 otherwise
    FD_med = zeros(100,1);
    FD_high = zeros(100,1);
    FN_low = zeros(100,1);      %1 if ID = 0 & RI = 1, 0 otherwise
    FN_med = zeros(100,1);
    FN_high = zeros(100,1);




for c =1:100
    Data20sec(c,1) = Ambients(randi([1 30],1,1),1);

end
Ave20sec = mean(Data20sec);
Std20sec = std(Data20sec);
RT_high = Ave20sec+Std20sec*T_high; %Running threshold level (high sens)
RT_med = Ave20sec+Std20sec*T_med; %Running threshold level (med sens)
RT_low = Ave20sec+Std20sec*T_low; %Running threshold level (low sens)

for c=1:25
    new = Ambients(randi([1 30],1,1),1);
    history(c,1)=new;

    if new>RT_high
            ID_high(c,1)=1;
            FD_high(c,1)=1;
    end
    if new>RT_med
            ID_med(c,1)=1;
            FD_med(c,1)=1;
    end
    if new>RT_low
            ID_low(c,1)=1;
            FD_low(c,1)=1;
    end

end

%Quiet interrupts
%Only high sensitivity should activate interrupt
for c=26:50
    new = Quiets(randi([1 30],1,1),1);
    history(c,1)=new;
    RI_high(c,1)=1;
    if new>RT_high
            ID_high(c,1)=1;
            TID_high(c,1)=1;
    else
        FN_high(c,1)=1;
    end
    if new>RT_med
            ID_med(c,1)=1;
            FD_med(c,1)=1;
```

```
    end
    if new>RT_low
            ID_low(c,1)=1;
            FD_low(c,1)=1;
    end

end

%medium interrupts
%Only high and medium sensitivity should activate interrupt

for c=51:75
    new = Mediums(randi([1 30],1,1),1);
    history(c,1)=new;
    RI_high(c,1)=1;
    RI_med(c,1)=1;
    if new>RT_high
            ID_high(c,1)=1;
            TID_high(c,1)=1;
    else
        FN_high(c,1)=1;
    end
    if new>RT_med
            ID_med(c,1)=1;
            TID_med(c,1)=1;
    else
        FN_med(c,1)=1;
    end
    if new>RT_low
            ID_low(c,1)=1;
            FD_low(c,1)=1;
    end

end



%loud interrupts
%all activate interrupt
for c=76:100
    new = Louds(randi([1 30],1,1),1);
    history(c,1)=new;
    RI_high(c,1)=1;
    RI_med(c,1)=1;
    RI_low(c,1)=1;
    if new>RT_high
            ID_high(c,1)=1;
            TID_high(c,1)=1;
    else
        FN_high(c,1)=1;
    end
    if new>RT_med
            ID_med(c,1)=1;
            TID_med(c,1)=1;
    else
```

```
            FN_med(c,1)=1;
        end
        if new>RT_low
                ID_low(c,1)=1;
                TID_low(c,1)=1;
        else
            FN_low(c,1)=1;
        end

    end

    %False Interrupt Detection Rate - Print all
    %probability of false interrupt given interrupt detection
    FDR_low_array(s,1) =  sum(FD_low, 'all')/(100-sum(RI_low, 'all'));
    FDR_med_array(s,1) = sum(FD_med, 'all')/(100-sum(RI_med, 'all'));
    FDR_high_array(s,1) = sum(FD_high, 'all')/(100-sum(RI_high, 'all'));

    TID_low_Array(s,1) = sum(TID_low,'all')/sum(ID_low, 'all');
    TID_med_Array(s,1) = sum(TID_med,'all')/sum(ID_med, 'all');
    TID_high_Array(s,1) = sum(TID_high,'all')/sum(ID_high, 'all');

    FN_low_array(s,1)=sum(FN_low,'all')/25;
    FN_med_array(s,1)=sum(FN_med,'all')/50;
    FN_high_array(s,1)=sum(FN_high,'all')/75;

    TPR_low_Array(s,1) = sum(TID_low, 'all')/(sum(RI_low, 'all'));
    TPR_med_Array(s,1) = sum(TID_med, 'all')/(sum(RI_med, 'all'));
    TPR_high_Array(s,1) = sum(TID_high, 'all')/(sum(RI_high, 'all'));

    end

    %!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
    DataLoc = char(DataLoc);

    figure(1)
    scatter(FDR_low_array,TPR_low_Array,'p','b')
    xlabel('False Positive Rate')
    ylabel('True Positive Rate')
    suptitle('Receiver Operating Characteristic, Low Sensitivity')
    title(DataLoc)
    xlim([0 1])
    ylim([0 1])
    grid on
    filenamelow = ['ROC_lowsens_standard' DataLoc '.png'];%
    saveas(gcf,filenamelow);

    figure(2)
    scatter(FDR_med_array,TPR_med_Array,'p','b')
    xlabel('False Positive Rate')
    ylabel('True Positive Rate')
    suptitle('Receiver Operating Characteristic, Medium Sensitivity')
    title(DataLoc)
    xlim([0 1])
    ylim([0 1])
    grid on
```

```
filenamemed = ['ROC_medsens_standard' DataLoc '.png'];
saveas(gcf,filenamemed);

figure(3)
scatter(FDR_high_array,TPR_high_Array,'p','b')
xlabel('False Positive Rate')
ylabel('True Positive Rate')
suptitle('Receiver Operating Characteristic, High Sensitivity')
title(DataLoc)
xlim([0 1])
ylim([0 1])
grid on
filenamehigh = ['ROC_highsens_standard' DataLoc '.png'];
saveas(gcf,filenamehigh);
```

Appendix C. Main Microcontroller Program (main.c)

```
#include <avr/io.h>
#include <avr/interrupt.h>
#include <avr/wdt.h>
#include <avr/sleep.h>
#include <math.h>


volatile uint16_t adc_num __attribute__((address(0x800100))); // Takes the number from the ADCH:ADCL register
volatile uint16_t adc_2sAvg __attribute__((address(0x800102))); // The 2 Second Average (average of 4x adc_num_arr_avg)
volatile uint16_t adc_20sAvg __attribute__((address(0x800104))); // The 20 Second Average (average of 10x adc_2sAvg)
volatile uint16_t adc_num_arr[50] __attribute__((address(0x800106))); // The adc_num_arr is an array of ADC values
volatile uint16_t adc_num_arr_avg __attribute__((address(0x80016A))); // Average of the adc_num_arr
volatile uint16_t adc_2sAvg_arr[10] __attribute__((address(0x80016C))); // Array of 2 Second Averages


volatile uint8_t adc_num_idx __attribute__((address(0x800180))); // Index of adc_num_arr (ADC array)
volatile uint8_t adc_2sAvg_idx __attribute__((address(0x800181))); // Index of adc_2sAvg (2s Avg array)
volatile uint16_t adc_sum_squared __attribute__((address(0x800182))); // The sum squared of the adc_2sAvg_arr
volatile uint16_t adc_std_dev __attribute__((address(0x800184))); // The square root of the adc_sum_squared divided by number of 2sAvgs (10)


volatile uint8_t thresh_sensitivity __attribute__((address(0x800190)));
// If thresh_sensitivity = 0, then Sensitivity is Low
// If thresh_sensitivity = 1, then Sensitivity is Medium
```

```
// If thresh_sensitivity = 2, then Sensitivity is High
volatile uint16_t thresh_high __attribute__((address(0x800191)));
volatile uint16_t thresh_medium __attribute__((address(0x800193)));
volatile uint16_t thresh_low __attribute__((address(0x800195)));


volatile uint8_t aux_switch __attribute__((address(0x8001A0)));
// If aux_switch == 3, Auxillary Ports are ON
// If aux_switch == 0, Auxillary Ports are OFF
volatile uint8_t sleep_code __attribute__((address(0x8001A1)));
// If sleep_mode = 0x00, then the device is powered on
// If sleep_mode = 0x0F, then the device restarts
// If sleep_mode = 0xFF, then the device is powered off


void WDT_init() __attribute__((naked)) __attribute__((section(".init3")));


void WDT_init()
{
   MCUSR = 0;
   wdt_disable();
   return;
}


int main()
{
   /* Initialization of Variables */
   aux_switch = 3;
   adc_num = 0;
   adc_2sAvg = 0;
   adc_20sAvg = 0;
   adc_num_idx = 0;
   adc_2sAvg_idx = 0;
   adc_sum_squared = 0;
   adc_std_dev = 0;
   sleep_code = 0x00;

   for(adc_num_idx = 0; adc_num_idx < 50; adc_num_idx++)
   {
      adc_num_arr[adc_num_idx] = 0;
   }
   adc_num_idx = 0;
   adc_num_arr_avg = 0;
```

```
    adc_num_idx = 0;
    for(adc_2sAvg_idx = 0; adc_2sAvg_idx < 10; adc_2sAvg_idx++)
    {
       adc_2sAvg_arr[adc_2sAvg_idx] = 0;
    }
    adc_2sAvg_idx = 0;

    thresh_sensitivity = 0; // Set sensitivity to High
    thresh_high = 44;       // High Sensitivity
    thresh_medium = 132;    // Medium Sensitivity
    thresh_low = 220;       // Low Sensitivity

    /* Initialization of I/O */
    DDRC &=~ 0x3F; // Set PINC4:0 as Button Inputs and PINC5 as Input for ADC
    DDRB |= 0x03; // Set PINB1:0 as Outputs for CD4066BE Switches
    DDRD |= 0x0B; // The Output showing the Threshold Sensitivity Mode.
    PORTD = 0x01;
    PORTB = 0x03;


    /* Initialization of ADC */
    ADMUX = (1 << REFS0)|(1 << MUX2)|(1 << MUX0);
    ADCSRA |= (0b111 << ADPS0);
    ADCSRA |= (1 << ADEN);

    /* Initialization of I/O Interrupt */
    PCICR = (1 << PCIE1);
    PCMSK1 = (1 << PCINT12)|(1 << PCINT11)|(1 << PCINT10)|(1 << PCINT9)|(1 << PCINT8);


    /* Initialization of Timer1 Interrupt*/
   TCNT1 = 12499; // Sets the timer to go off every 10 milliseconds.
   TCCR1A = (0b00<<COM1A0)|(0b00<<COM1B0)|(0b00<<WGM10);
   TCCR1B = (0b00<<WGM12)|(0b010<<CS10); // Set the Prescaler to 8
   TIMSK1 = (0b1<<TOIE1); // Enables interrupts for TOV1


    sei(); // Enable Interrupts
    /* Program Starts */
    while(1)
    {
       asm("NOP");
    }
}
```

```
ISR (PCINT1_vect) /* I/O Interrupt Routine */
{
   cli();
   if(PINC & (1 << PINC4)) // Power Off Button
   {
      if(sleep_code == 0x00)
      {
         sleep_code = 0xFF;
         } else if (sleep_code == 0xFF) {
         sleep_code = 0x0F;
      }
   }
   if(PINC & (1 << PINC3)) // Restart Button
   {
      wdt_enable(WDTO_250MS);
      // Perform a Soft Reset on the System
   }
   if(PINC & (1 << PINC2)) // High Sensitivity Button
   {
      thresh_sensitivity = 2;
      // High Sensitivity On
   }
   if(PINC & (1 << PINC1)) // Medium Sensitivity Button
   {
      thresh_sensitivity = 1;
      // Medium Sensitivity On
   }
   if(PINC & (1 << PINC0)) // Low Sensitivity Button
   {
      thresh_sensitivity = 0;
      // Low Sensitivity On
   }

   switch(thresh_sensitivity)
   {
      case 0:
      PORTD = 0x01; // Low Sensitivity
      break;
      case 1:
      PORTD = 0x02; // Medium Sensitivity
      break;
      case 2:
      PORTD = 0x08; // High Sensitivity
```

```
        break;
        default:
        PORTD = 0x0B; // Default (Error)
        break;
    }

    if(sleep_code == 0xFF) // If the device is powered down and the button is pressed again, the device restarts
    {
        sei();
        PORTD = 0x00;
        PORTB = 0x03;
        set_sleep_mode(2);
        sleep_mode();
        } else if(sleep_code == 0x0F) {
        sleep_disable();
        wdt_enable(WDTO_250MS);
    }
    sei();
}



ISR (TIMER1_OVF_vect)
{
// This interrupt service routine is non-atomic
    TCNT1 = 12499;
    /* Measure and Store ADC value at this point in time */
   ADCSRA |= (1 << ADSC);
   while(ADCSRA & (1 << ADSC));
    adc_num = ADC;
    adc_num_arr[adc_num_idx % 50] = adc_num; // Adds Element to the Array
    adc_num_arr_avg += adc_num_arr[adc_num_idx % 50];
    adc_num_idx++;

    /* Compute 2s Average */
    if((adc_num_idx > 0) && (adc_num_idx % 50 == 0))
    {
        adc_num_arr_avg /= 50;
        adc_2sAvg += adc_num_arr_avg;
        adc_num_arr_avg = 0;
    }

    if(adc_num_idx == 200)
    {
        adc_num_idx = 0;
```

```
        adc_2sAvg /= 4;
        adc_2sAvg_arr[adc_2sAvg_idx] = adc_2sAvg;
        adc_2sAvg = 0;
        adc_20sAvg += adc_2sAvg_arr[adc_2sAvg_idx];
        adc_2sAvg_idx++;
    }
    /* 20 Second Average and Standard Deviation Calculation */
    if(adc_2sAvg_idx == 10)
    {
        adc_std_dev = 0;
        adc_2sAvg_idx = 0;
        adc_20sAvg /= 10;
        for(adc_2sAvg_idx = 0; adc_2sAvg_idx < 10; adc_2sAvg_idx++)
        {
            adc_sum_squared = adc_2sAvg_arr[adc_2sAvg_idx] - adc_20sAvg;
            adc_sum_squared *= adc_sum_squared;
            adc_std_dev += adc_sum_squared;
        }
        adc_std_dev = (uint16_t)sqrt(adc_std_dev / 10);
        adc_sum_squared = 0;
    }

    if(adc_20sAvg + (adc_std_dev * thresh_high) > adc_2sAvg_arr[adc_2sAvg_idx] && thresh_sensitivity ==
0) // Above the High Sensitivity Threshold
    {
        aux_switch = 0;
        // Disable Audio
    } else if(adc_20sAvg + (adc_std_dev * thresh_medium) > adc_2sAvg && thresh_sensitivity == 1) // Above
the Medium Sensitivity Threshold
    {
        aux_switch = 0;
        // Disable Audio
    } else if(adc_20sAvg + (adc_std_dev * thresh_low) > adc_2sAvg && thresh_sensitivity >= 2) // Above the
Low Sensitivity Threshold
    {
        aux_switch = 0;
        // Disable Audio
    }

    if(adc_20sAvg + (adc_std_dev * thresh_high) <= adc_2sAvg && thresh_sensitivity == 0) // Below or Equal
to the High Sensitivity Threshold
    {
        aux_switch = 3;
        // Enable Audio
```

```
   } else if(adc_20sAvg + (adc_std_dev * thresh_medium) <= adc_2sAvg && thresh_sensitivity == 1) // Below
or Equal to the Medium Sensitivity Threshold
   {
      aux_switch = 3;
      // Enable Audio
   } else if(adc_20sAvg + (adc_std_dev * thresh_low) <= adc_2sAvg && thresh_sensitivity >= 2) // Below or
Equal to Low Sensitivity Threshold
   {
      aux_switch = 3;
      // Enable Audio
   }

   if(adc_2sAvg_idx == 10)
   {
      adc_2sAvg_idx = 0;
      adc_20sAvg = 0;
   }


   PORTB = aux_switch; // Set Auxiliary Switch to 0
}
```

Test Plans: Electronics

Test Name	Microphone Test	Test Number	ET_01			
Requirement(s) Tested	5.4, 5.2	Verification Method	I	A	D	T
Test Setup	Fig. 6, 8 Oscilloscope Variable DC Power Supply Microphone Part Number: PM0F-6050P-36UQ					

Test Step	Action (Attach test data, diagrams, etc. as appropriate)	Expected Result	Observed Result	Pass Fail
1	Build Microphone Circuit as shown in Fig. 6	See Fig. 6	Circuit ready for testing	Pass
2	Turn on and set power supply to 5VDC and connect to Vcc on circuit diagram; connect ground.	5VDC on Vcc, verifiable with volt meter	Voltmeter displayed 5VDC	Pass
3	Turn on oscilloscope, probe output of microphone and connect your ground wire.	Voltage output should be in the microvolt to millivolt range.	Voltage reading:-645uV@52.3dB	Pass
4	Simulate a loud noise, such as banging your hand on the table.	dB level between 30 and 90 dB; appropriate for volume level	66.2dB	Pass
5	Record voltage output from microphone.	Between 1 and 500 mV	28.8mV	Pass
6	Simulate the same noise but try to make it louder	Voltage output should be between 10 and 500 mV and voltage output increases with volume level increase.	33.6mV@77.5dB	Pass

Comments

The microphone used about 28-35mV of electricity depending on Decibels of noise heard.

Date 3/07/2020 Test Engineer Ryan Dumont Witness Jared Alves

Test Name	Amplifier Circuit Test	Test Number	ET_02			
Requirement(s) Tested	5.4	Verification Method	I	A	D	T
Test Setup	See Figure 7 Variable DC Power Supply Oscilloscope					

Test Step	Action (Attach test data, diagrams, etc. as appropriate)	Expected Result	Observed Result	Pass Fail
1	Build Amplifier Circuit as shown in Fig. 7	See Fig. 7	Circuit ready for testing.	Pass
2	Turn on Power Supply and connect 5V DC to Amplifier supply rails (see Fig 7).	Voltage output verifiable with Volt Meter	Multimeter reads 5Vdc.	Pass
3	Turn on Function Generator and select (any) waveform with voltage range 0-500mV. Turn on Oscilloscope and connect probes to function generator output.	Oscilloscope window displays waveform.	Input set to 17.6mV peak.	Pass
4	Connect function generator output to input of amplifier circuit (see Fig. 7). Probe Channel 1 oscilloscope probes to amplifier input. Connect oscilloscope Channel 2 probes to amplifier output (see Fig 7).	Expected gain ~11.	Output measured at 197mv.	Pass

Comments

The amplifier successfully amplified a signal that was at peak 17.6 mV to 197 mV.

Date 3/07/2020 Test Engineer Ryan Witness Stephen

Test Name	Digital Circuit Switch Test	Test Number	ET_03			
Requirement(s) Tested	4.1	Verification Method	I	A	D	T
Test Setup	Figure 9 Digital Switch Part Number: CD4066BE DC Power Supply (For Switch's CTRL input and power source) Function Generator (For the input signal, an AC power source) Oscilloscope (Measuring the input and output signal of the Switch)					

Test Step	Action (Attach test data, diagrams, etc. as appropriate)	Expected Result	Observed Result	Pass Fail
1	Build Digital Switch Circuit as shown in Figure 9	See Figure 9		
2	Turn on power Supply and connect 5V DC to the VDD of the Digital Switch Circuit.	Voltage output verifiable with Volt Meter	The 5v did activate the chip	pass
3	Use the Function Generator to send out a 5V AC Sine Signal and then test with a 5V AC Noise Signal. The signal will go to the input of one of the Digital Switches on the chip.	Voltage output verifiable with oscilloscope.	Instead of using a AC generator we connected the aux cable and used a actuall sound signal.	Pass
4	Set the CTRL pins of a switch between 5V and GND. Then compare the results on Step 5.	Voltage output verifiable with Volt Meter.	While connecting it to 5V the sound went through the circuit and connecting it to ground stopped it.	Pass
5	Connect an Oscilloscope (1x) Probe to the input signal (Vin) and output signal (Vout) and compare the two signals. Connect the grounds as well.	Verifiable through Oscilloscope: If CTRL = 5V Vout = Vin If CTRL = GND Vout = 0V DC	Sound signal is very noisy but we did see a signal input and output and heard it as well	pass

Comments:

Disconnecting the switch from vdd still left the switch on until it was connected to ground.

Date 3/7/20 Test Engineer Stephen Witness Ryan, Michael

Test Name	Auxiliary Port & Cable Test	Test Number	ET_04			
Requirement(s) Tested	2.1,2.2,5.1,6.3,7.0	Verification Method	I	A	D	T
Test Setup	Audio Source (computer or phone) 3.5mm headphone jack(x2) Male-to-Male Auxiliary Cable Headphones					

Test Step	Action (Attach test data, diagrams, etc. as appropriate)	Expected Result	Observed Result	Pass Fail
1.	Attach the 3.5mm headphone jack to the input and output to make sure it fits.	The Auxiliary jack should fit any standard 3.5mm audio device.	The headphones did fit in the jack inputs and outputs	Pass
2	Connect Auxiliary ports together in series.			
3	Connect the M-M auxiliary cable from an audio source to the auxiliary ports and connect headphones to the output of the other auxiliary port. Observe audio output to headphones.	The audio signal would travel through the device with little delay and drop of signal.	The audio signal did travel through the device.	Pass

Comments

The headphone jack fit in the Aux port and successfully was able to send audio from one aux to the other.

Date 3/07/2019 Test Engineer Stephen Witness Ryan

Test Name	Testing of Buttons and Setting Switch	Test Number	ET_05			
Requirement(s) Tested	3.2, 6.1	Verification Method	I	A	D	T
Test Setup	Signal Detector (Multimeter or Oscilloscope) Voltage Signal					

Test Step	Action (Attach test data, diagrams, etc. as appropriate)	Expected Result	Observed Result	Pass Fail
1	Connect one pin of each button to a voltage signal (5VDC). Measure input and output voltage with multimeter.	When Button is pressed the open circuit should become closed and the voltage signal should go through.	The Voltage is approx. 5V when the button is pressed and 0V when not pressed	Pass
2	Connect the Switch to a voltage signal and ground. Measure input and output voltage with multimeter.	Just like the buttons, when the switch is pressed check if current flows.	The audio signal from input and output mat	Pass

Comments

The buttons and switch worked as expected.

Date 3/4/2020 Test Engineer Drew Martins Witness Ryan Dumont

Test Name	Analog Input Electronics Test	Test Number	ET_06			
Requirement(s) Tested	5.4	Verification Method	I	A	D	T
Test Setup	Fig. 6 7, 8 Sound Voltmeter Variable DC Power Supply Oscilloscope					

Test Step	Action (Attach test data, diagrams, etc. as appropriate)	Expected Result	Observed Result	Pass Fail
1	Verify Tests ET_01, ET_02. Build Microphone Test Circuit and Amplifier Test Circuit (see Fig. 6 and 7)	See Fig. 6 and 7		Pass
2	Connect output of Microphone Circuit to input of Amplifier Circuit.	See Fig. 8	Analog integration circuit ready for testing.	Pass
3	Turn on Power Supply and connect 5VDC to Amplifier supply rails (see Fig 8).	Voltage output verifiable with Volt Meter	Meter read 5Vdc	Pass
4	Turn on Oscilloscope and connect Channel 1 probe to Microphone output (e.g. Amplifier Input) and Channel 2 probe to amplifier output.	Observable amplification (A=11) of ambient.	Channel 1 read 13.073mV and Channel 2 read 138.11mV.	Pass
5	Simulate a loud noise, such as banging your hand on the table.			Pass
6	Record voltage output from microphone and from amplifier.	Expected gain ~11	Channel 1 read 63.947mVdc and Channel 2 read 1.1004Vdc	Pass
7	Simulate the same noise but louder.	Voltage increases linearly with sound level reading (dB).	Channel 1 read 61.27mVdc .687Vdc and Channel 2 read .687Vdc .	Pass

Comments:
Combining the microphone and amp allowed for an increase in the voltage being read.

Date 3/07/2020 Test Engineer Ryan Witness Stephen

Test Name	Audio Switch Circuit	Test Number	ET_07			
Requirement(s) Tested	2.1, 2.2, 2.3, 4.1	Verification Method	I	A	D	T
Test Setup	Fig. 9 Digital Switch Part Number: CD4066BE) DC Power Supply (For Switch's CTRL input and power source) A Male-to-Male Auxiliary Cable from a device to the circuit & a pair of headphone or speakers. Two Auxiliary Ports (One Aux-Input and One Aux-Output)					

Test Step	Action (Attach test data, diagrams, etc. as appropriate)	Expected Result	Observed Result	Pass Fail
1	Verify tests ET_03 and ET_04	See ET_03 and ET_04	Tests complete.	Pass
2	Connect a 5V DC power source to the VDD on the Switches	Voltage output verifiable with Voltmeter	Voltmeter gives an output of 5V*	Pass
3	Set up two 5V DC inputs for the CTRL pins, one for each switch	Voltage output verifiable with Voltmeter	The switches are activated when CTRL Pin set to 5V and deactivated when CTRL Pin is set to 0V/GND	Pass
4	Connect an audio source (auxiliary cable – male-male) to the Aux-Input port	Auxiliary cable inserted	Auxiliary cable is connected	Pass
5	Connect headphones or speaker to Aux-Output and begin any mp3 file.	Audio can be heard if switch is on.	Audio can be heard when the switch is activated.	Pass
6	Set the CTRL pins of each switch between 5V and GND and determine if the audio is getting through to the other side,	If CTRL = 5V Audio is enabled If CTRL = GND Audio is muted	Audio is enabled on 5V Audio is disabled on GND	Pass

Comments:
*Due to COVID-19, a DC power source was inaccessible, the workaround was to add 2x 3.7V batteries in series and a voltage divider to create a 5V DC source.

Date Test Engineer Witness

Test Name	Analog Integration with ADC Test & Debug	Test Number	ET_08			
Requirement(s) Tested	3.1, 5.1, 5.2, 5.3, 5.4	Verification Method	I	A	D	T
Test Setup	Integration of ET_06 and CT_01 ATmega328PB Variable DC Power Supply Atmel Studio					

Test Step	Action (Attach test data, diagrams, etc. as appropriate)	Expected Result	Observed Result	Pass Fail
1	Verify functionality of ET_06.	Complete ET_06	Complete ET_06	Pass
2	Connect the output of the op- amp to pin C5 on the microcontroller.	ET_06 will be able to communicate with microcontrollers' ADC.	ET_06 can communicate with the microcontroller's ADC	Pass
3	Run CT_01.	See CT_01		
4	Open the Atmel Studio software on your computer and use the debug function.			
5	View the values that are seen by the ADC as ET_06 is taking in the ambient noise.	The ADC number depends on the volume of the incoming audio to the microphone.	WIP	WIP*

Comments:
*Test Step 5 is currently a work in progress.
Due to COVID-19, the equipment and teamwork required to finish this test plan was inaccessible.

Date_____Test Engineer_____Witness_____

Test Name	Power Consumption Test	Test Number	ET_09			
Requirement(s) Tested	1.1	Verification Method	I	A	D	T
Test Setup	Integration of ET_08 and ET_05 ATmega328PB Power Supply Digital Multimeter					

Test Step	Action (Attach test data, diagrams, etc. as appropriate)	Expected Result	Observed Result	Pass Fail
1	Build ET_08 and load CT_05 in microcontroller.	Program successfully loaded. See ET_08.		
2	Set power supply to 5 Vdc and connect the positive lead to Vcc and negative lead to ground.	Verifiable with voltage meter		
3	Take digital multimeter and set it to read current.	Multimeter should switch to amperage mode.		
4	Disconnect positive lead of power supply and connect one end of meter to that wire and the other lead of meter to Vcc on the circuit.	This will complete the circuit and allow you to measure current draw.		
5	Run circuit for an hour and log the amperage output on multimeter for five-minute intervals	Current values should be roughly in the same range of each other.		
6	Take average of those numbers and that is roughly the current draw of circuit per hour. Multiply this number times 8 (# of hours device needs to run for) and record the number. This is the current rating you will need on your battery.	An average of the current levels will be obtained to help with battery selection.		

Comments:
Due to COVID-19, ET_08 remained incomplete which is a prerequisite of ET_09.

Date_____Test Engineer_____Witness_____

Test Name	Audio Switching Function Test	Test Number	ET_10			
Requirement(s) Tested	3.2	Verification Method	I	A	D	T
Test Setup	Variable DC Power Supply ATmega328PB Integration of CT_04 and ET_08					

Test Step	Action (Attach test data, diagrams, etc. as appropriate)	Expected Result	Observed Result	Pass Fail
1	Have ET_08 already built and CT_04 running on the microcontroller.	See ET_08 and CT_04.		
2	Implement the code from CT_04 with the circuit, testing the switch with the various sensitivity levels	Switch circuit functions with sensitivity levels accurately		

Comments:
Due to COVID-19, ET_08 remained incomplete which is a prerequisite of ET_10.

Date_____Test Engineer_____Witness_____

Test Name	Interrupt Detection Device Test	Test Number	ET_11			
Requirement(s) Tested	3.1, 4.1, 5.1, 5.4	Verification Method	I	A	D	T
Test Setup	Integration of ET_07 and CT_04 ATmega328PB Variable Power Supply Digital Multimeter Oscilloscope					

Test Step	Action (Attach test data, diagrams, etc. as appropriate)	Expected Result	Observed Result	Pass Fail
1	Make sure ET_07 is built and functioning correctly and CT_04 is running on the microcontroller.	See ET_07 and CT_04.		
2	Power up the variable DC power supply and set the output voltage to 5 VDC.	Verifiable with voltmeter.		
3	Connect the positive lead to the +5 VDC voltage rail and the negative lead to the ground rail of the circuit.	Circuit should have power at this point.		
4	Power on oscilloscope and connect test probes to channel one and channel 2.	Oscilloscope will be ready to view ambient noise.		
5	Probe the input and output with channel 1 and channel 2 respectively to make sure microphone and amplifier are reading the ambient noise in the environment.	Original and amplified signal should be in view on screen.		
6	Create a loud noise of your choosing for about 2 seconds to trigger an interrupt.	Loud noise audible.		
7	Verify that an interrupt is achieved, examining the output pin of the microphone designated for the interrupt.	Interrupt is achieved.		

Comments:
Due to COVID-19, the equipment and teamwork required to finish this test plan was inaccessible.

Date_____Test Engineer_____Witness_____

Test Name	Acoustic Awareness Device Test	Test Number	ET_12			
Requirement(s) Tested		Verification Method	I	A	D	T
Test Setup	Integration of ET_09, ET_10 and ET_11 Website: noises.online Sound Meter					

Test Step	Action (Attach test data, diagrams, etc. as appropriate)	Expected Result	Observed Result	Pass Fail
1	Complete Tests ET_09, ET_10 and ET_11.	Verify ET_09, ET_10 and ET_11 are functioning.		
2	Connect the interrupt output pin on the microcontroller from ET_12 to the input of ET_11. Monitor the output pin using a voltmeter or oscilloscope probe.	Follow circuit diagram for ET_13.		
3	Begin audio throughput through the auxiliary cables from the device source to the switching circuit and out to the headphones.	Sound coming from device source is on.		
4	Select an ambient noise level from website: noises.online as was done in ET_06. Allow device to maintain a constant ambient reading for 20 seconds.	Sound selection is audible for 20 seconds.		
5	After 20 seconds, rapidly increase the volume of the output of the sound to start an interrupt. Monitor the audio output to the headphones.	Audio to the headphones should turn off when an interrupt is started. Ambient sound selection is much louder than the previous step to trigger and interrupt.		
6	Repeat step 5 and record ambient levels before and after the interrupt while taking record of whether the audio was turned off at the time of the ambient volume increase.	Audio should turn off at volume increase IF the threshold has been exceeded.		
7	Repeat steps 5 and 6 for all threshold level settings (high, medium, low).	Record under which cases the audio was turned off and the setting, ambient and interrupting sound level.		

Comments
Due to COVID-19, ET_09, ET_10 and ET_11 remained incomplete which is a prerequisite of ET_12.

Date_____Test Engineer_____Witness_____

Test Name	Battery Power Consumption Test	Test Number	ET_13			
Requirement(s) Tested	1.1	Verification Method	I	A	D	T
Test Setup	Integration of ET_11 and ET_09 ATmega328PB Selected Lithium Ion Battery Stopwatch					

Test Step	Action (Attach test data, diagrams, etc. as appropriate)	Expected Result	Observed Result	Pass Fail
1	Using calculated current consumption in ET_09, choose a lithium ion battery pack from an online retailer that has at least that much or more capacity.	Battery has enough capacity for 8 hours.		
2	Connect battery to ET_11 and run CT_05 for 8 hours.	The device should run on battery power for at least 8 hours.		

Comments:
Due to COVID-19, ET_09 and ET_11 remained incomplete which is a prerequisite of ET_13.

Date_____Test Engineer_____Witness_____

Test Name	Acoustic Awareness Enabler 1.0 Debug Test	Test Number	ET_14			
Requirement(s) Tested	All	Verification Method	I	A	D	T
Test Setup	Integration of BT_05, ET_12, PT_07 and NT_03					

Test Step	Action (Attach test data, diagrams, etc. as appropriate)	Expected Result	Observed Result	Pass Fail
1	Ensure that all components are housed and properly mounted in the modified enclosure.	A “completed” device will be built.		
2	Turn on the Acoustic Awareness Enabler.	Device powers up.		
3	Use one aux cord to connect your computer to the input jack of the A.A.E	This is where the music input will come from		
4	Connect your headphones from the output of the A.A.E to your ears.	This will allow you to hear the audio signal.		
5	Test each sensitivity level individually by pressing the panel mounted button associated with it.	Threshold setting changes to desired threshold setting.		
6	Create various “interrupts” for each sensitivity level test.	Sound traveling through headphones should stop. Depending how well it stops may mean software adjustments for thresholds.		
7	Log the behavior of the device and see if fine tuning in software is needed.	Results will tell you what work still needs to be done.		

Comments:
ET_14 could not be satisfied since ET_12 is a prerequisite of ET_14 and remains incomplete.

Date_____Test Engineer_____Witness_____

Test Name	Device Lifetime Test	Test Number	ET_15			
Requirement(s) Tested	1.1	Verification Method	I	A	D	T
Test Setup	Completed Circuit Board Chosen Battery Pack					

Test Step	Action (Attach test data, diagrams, etc. as appropriate)	Expected Result	Observed Result	Pass Fail
1	Verify that BT_05 is satisfied	Circuit board should be populated and ready to test.		
2	Take chosen battery and wire the positive terminal to Vcc on the final circuit board and the negative terminal to ground. Keep battery attached for 8 hours and turn on circuit. Do various tests during the 8-hour interval.	Battery should not die for the full length of time.		

Comments:

Date_____Test Engineer_____Witness_____

Test Name	Device Improvement Methods	Test Number	ET_16			
Requirement(s) Tested	All	Verification Method	I	A	D	T
Test Setup						

Test Step	Action (Attach test data, diagrams, etc. as appropriate)	Expected Result	Observed Result	Pass Fail
1	Confirm NT_03, BT_05, ET_15, ET_14, PT_07	See NT_03, BT_05, ET_15, ET_14, PT_07.		
2	Verify each category for possible improvements: Enclosure, PCB and electronics	Final product		

Comments:
ET_16 could not be satisfied as ET_14 is a prerequisite of ET_16 and remains incomplete.

Date_____Test Engineer_____Witness_____

Test Plans: Processing

Test Name	Threshold Setting as f(σ) concept validation	Test Number	PT_01			
Requirement(s) Tested	5.1	Verification Method	I	A	D	T
Test Setup	MATLAB, Sound Meter					

Test Step	Action (Attach test data, diagrams, etc. as appropriate)	Expected Result	Observed Result	Pass Fail
1	Obtain data set from environment X with sound meter, collecting 30 samples from “quiet environment”. Collect ambient samples, simulate “Low”, “Med” and “High” Interrupts and check sound levels.	Ambient levels (dB) and sound levels of low, medium and high interrupts	Data collected using sound meter. Example: Figure 11.	P
2	Repeat step 1 for “Medium” environment and “Loud” Environment	New data sets with same value categories	Data collected for several environments.	P
3	Compare and compute standard deviations in MATLAB of various data sets	Standard deviations of data sets	Standard deviation and averages calculated (Figure 12).	P
4	Verify that threshold from experimental data is based on standard deviation	Threshold Setting as a function of the standard deviation	Average standard deviation from the ambient average is calculate over all environments for each threshold setting (Figure 14).	P

Comments:
This data validates the concept of threshold as a function of standard deviation. Further tests improve accuracy of standard deviation.

Date 3/4/2020 Test Engineer Michael Witness Jared

Test Name	Standard Deviation Calculation	Test Number	PT_02			
Requirement(s) Tested	5.1,5.2	Verification Method	I	A	D	T
Test Setup	MATLAB In conjunction with PT_01					

Test Step	Action (Attach test data, diagrams, etc. as appropriate)	Expected Result	Observed Result	Pass Fail
1	Generate a plot of ambient level vs standard deviation in MATLAB by comparing measured data and standard deviations for data sets	Determine the relationship between the standard deviation and the ambient level of the environment	Plot of ambient vs standard deviation (Figure 15)	P
2	Derive a formula for threshold levels based on ambient and standard deviation	Find relationship between threshold setting and standard deviation	Threshold Formula (Figure 16)	P

Comments:

The plot shows that there is a general positive correlation between ambient level and standard deviation.

Date 3/7/2020 Test Engineer Jared Alves Witness Michael Benker

Test Name	20 second averaging, interrupt process	Test Number	PT_03			
Requirement(s) Tested	5.1,5.2,5.3	Verification Method	I	A	D	T
Test Setup	Microsoft Visual Studio, C# See attachment Code (Interrupt.cs)					

Test Step	Action (Attach test data, diagrams, etc. as appropriate)	Expected Result	Observed Result	Pass Fail
1	Write and compile code in C# that simulates the collection of 20 seconds or more of data at a rate of 100 samples per second.	Stores 20 seconds of data.	2-second averages are recorded and saved in the program. When 20 of data is present, newest data set replaces oldest.	Pass
2	Implement standard deviation calculation to determine if a 2 second interval after 20 seconds of data is within a number of standard deviations. If the most recent 2 seconds is above a number of standard deviations, the program should indicate it.	Standard deviation calculation of interrupt is verifiable through calculation.	After 20 seconds of data present, standard deviation is calculated for newest data set.	Pass
3	If not present in the test code thus far, modify the code so that the oldest number in the data set is replaced by the newest average if the newest average is within a number of standard deviations.	After 20 seconds, data stored should always be from the most recent 20 seconds if there were no interrupts. See attachment code for final example of implementation.	Present in program.	Pass
4	Program says when newest data set falls out of threshold range [number of standard deviations], which is set in the program.	“Interrupt” is displayed when number of Threshold standard deviations * sigma + average is equal to newest 2 sec average.	Program displays when interrupt is achieved and was found to be accurate in testing with #std=4, average = 56dB, sigma = 1.0 and newest 2 sec average is 63dB.	Pass

Comments

Date 3/7/2020 Test Engineer Michael Witness Jared

Test Name	Interrupt Environment Test	Test Number	PT_04			
Requirement(s) Tested	5.1,5.2,5.3	Verification Method	I	A	D	T
Test Setup	Sound level meter Visual Studio (C# language) - “interrupt.cs” (see attached code) See Figure 18					

Test Step	Action (Attach test data, diagrams, etc. as appropriate)	Expected Result	Observed Result	Pass Fail
1	Load “interrupt.cs” on Visual Studio	See PT_03.		Pass
2	Measure ambient sound levels for 20 seconds using the sound meter and record the values in the program “interrupt.cs”.	Program displays recorded values after entering.		Pass
3	Generate interrupts and monitor with sound meter	Various interrupt dB levels from meter		Pass
4	Load measured interrupt values into program and test if an interrupt is triggered within program	Program will display whether an interrupt should occur in software	Interrupt is triggered with loud instantaneous noises (Figure 18)	Pass
5	If discrepancies occur, program must be edited	Optimized program	Possible changes to be made to highest threshold (T3)	Pass

Comments
The program “interrupt.cs” only provides a simple initial test, since loud instantaneous noises tend to trigger the interrupt to occur, which is undesirable. This is because the program takes a two second average by taking only a minimum and maximum sound level and averaging those two discrete levels. Highest threshold is difficult to trigger, especially when standard deviation is high (>1).

Date 3/7/2020 Test Engineer Jared Alves Witness Michael Benker

Test Name	Process Debug Test	Test Number	PT_05			
Requirement(s) Tested	5.1	Verification Method	I	A	D	T
Test Setup	Microsoft Visual Studio, C# “interrupt.cs” program					

Test Step	Action (Attach test data, diagrams, etc. as appropriate)	Expected Result	Observed Result	Pass Fail
1	Test different averaging times (30 second average vs 3 second average) within C# by updating program	Data on 30 second average and 3 second average	Figure 19 – 20 vs 2 second average is best choice	Pass
2	Implement recursive averaging in software (C#)	Comparison of recursion to multi-averaging method	Figures 20 through 24 – No discernible difference between built in C# averaging and recursion methods	Pass
3	Test averaging to see if a time delay occurs when taking the average using Visual studio due to instruction timing	Time delay confirmation	Figure 25 – Iterative averaging yields wide variance in ticks	Pass

Comments
Within the program at least, implementing a 30 second average vs a 3 second average did not improve the averaging process. An accurate standard deviation or average could not be obtained (See Figure 19). Recursive averaging does not change average calculation in a noticeable way. The iterative averaging method will be used. The timing yields a wide variance of ticks (between 4000 and 6000 ticks but can be as low as 500). This is done using the Stopwatch class in C#

Date 3/22/2020 Test Engineer Jared Alves Witness Michael Benker

Test Name	Standard Deviation Debugging	Test Number	PT_06			
Requirement(s) Tested	5.1	Verification Method	I	A	D	T
Test Setup	Microsoft Visual Studio, C#, program: “interrupt.cs”, “environment_interrupt.cs”					

Test Step	Action (Attach test data, diagrams, etc. as appropriate)	Expected Result	Observed Result	Pass Fail
1	After ET_11, see if number of standard deviations must be fine-tuned (bypassed due to COVID-19)	Result of ET_11		
2	Use environmental-based interrupt program to test sound meter data if interrupts occur correctly	Determination of whether the standard deviations should be altered	High standard deviation environments are way off	
3	Test various standard deviation multiples close to the values originally chosen to see if different multiples of the standard deviation give more accurate threshold based on environmental data	Fine-tuned standard deviation multiples		

Comments:
Due to COVID-19, ET_11 remained incomplete which is a prerequisite of ET_12.

Date_____Test Engineer_____Witness_____

Test Name	Device Accuracy Test & Debug	Test Number	PT_07			
Requirement(s) Tested	5.1,5.4	Verification Method	I	A	D	T
Test Setup	Integration of BT_05, PT_05 and PT_06					

Test Step	Action (Attach test data, diagrams, etc. as appropriate)	Expected Result	Observed Result	Pass Fail
1	Confirm that Customer Requirement 5 is met by checking that the device is accurate 99% of the time through repetitive testing of device	Requirement 5		
2	Repeat testing of device as many times until 99% accuracy is achieved, fine tuning PT_05 and PT_06 as necessary	99% efficiency and Customer Requirement 5 met		
3	Change to new environment with a wider variance of ambient values and repeat process	Accurate device in varying environments		

Comments

Date_____Test Engineer_____Witness_____

Test Plans: Coding

Test Name	ADC Test Code	Test Number	CT_01			
Requirement(s) Tested	5.1	Verification Method	I	A	D	T
Test Setup	Microsoft Visual Studio, C See Appendix Code (ADC Test Code)					

Test Step	Action (Attach test data, diagrams, etc. as appropriate)	Expected Result	Observed Result	Pass Fail
1	Begin Microsoft Visual Studio and select C as language	N/A		Pass
2	Initialize the I/O and interrupt header files	Allows for I/O and interrupt libraries to be used		Pass
3	Write a function to clear two ports (ex. B and D)	Function is usable in main		Pass
4	Within main function, set Data Direction registers B, C and D as necessary	Data Direction Registers configured	PINB1:0 = ADC High PIND7:0 = ADC Low	Pass
5	Initialize ADMUX and ADC Status and Control Registers (reserved voltage and ADC channel)	Correct ADC channel and voltage configuration		Pass
6	Enable global interrupts and set timer1 using TCNT1 and TMSK1 as well as control registers	Interrupts enabled to be used for step 7	Timer1 is activated once every second	Pass
7	Write Interrupt Service Routine to display highest two digits and lowest 8 digits of ADC	Tested ADC	The ADC is read from PINC5 and sent to PB1:0 and PD7:0 (See Figure 32)	Pass

Comments
See Figure 32.

Date_____Test Engineer_____Witness_____

Test Name	Audio Switch Test Code	Test Number	CT_02			
Requirement(s) Tested	2.3, 4.1, 6.2	Verification Method	I	A	D	T
Test Setup	Processor: ATmega328PB Program: Test Code for Buttons References: ATmega328PB Datasheet Hardware: CD4066BE Quad Bilateral Switch (14-DIP) Chip					

Test Step	Action (Attach test data, diagrams, etc. as appropriate)	Expected Result	Observed Result	Pass Fail
1	Write C code that tests the CPU and Switches			Pass
2	Set PINB1:0 as outputs to each Switch (PB1 for Aux-Left and PB0 for Aux-Right).	PINB1 -> Aux Left PINB0 -> Aux Right	PINB1:0 controls the CD4066BE Bilateral Switch	Pass
3	Run a loop that counts PINB1:0 from 0 to 3, increment by 1 per second.	t = 0s, PB1:0 = 0 t = 1s, PB1:0 = 1 t = 2s, PB1:0 = 2 t = 3s, PB1:0 = 3 t = 4s, PB1:0 = 0	t=0,PB1=0,PB0=0 t=1,PB1=0,PB0=1 t=2,PB1=1,PB0=0 t=3,PB1=1,PB0=1 t=4,PB1=0,PB0=0	Pass
4	Test Circuit and Code	The sound should turn on each side of Aux depending on the state of PB1 and PB0.	The sound turns on each end of the headphone	Pass

Comments:
The Auxillary Ports (Input and Output) both needed a 1kΩ resistor to GND.

Date_____Test Engineer_____Witness_____

Test Name	Setting Button Test Code	Test Number	CT_03			
Requirement(s) Tested	3.2, 6.1, 6.3	Verification Method	I	A	D	T
Test Setup	Processor: ATmega328PB Program: Test Code for Buttons References: ATmega328PB Datasheet Hardware: • 5x Push Buttons w/4.7kΩ Pull-up Resistor. • 3x LED lights w/330Ω Resistor.					

Test Step	Action (Attach test data, diagrams, etc. as appropriate)	Expected Result	Observed Result	Pass Fail
1	Write C code that tests the CPU and Buttons			Pass
2	Set PIND2:0 as output for the LED, each LED represents different threshold levels	PD2:0 is the output	PD2:0 is set as the output	Pass
3	Initialize PINC4:0 as the pull-up input for each button.	PINC4:0 are now pull-up inputs for the buttons to use	PC4:0 are set as button inputs	Pass
4	Implement button PINC3 as the RESET button.	PINC3 restarts the device.	PC3 button restarts the device	Pass
5	Implement button PINC2:0 as Threshold Sensitivity buttons. Sets the Sensitivity levels.	PINC2 = High PINC1 = Medium PINC0 = Low	PC2 is High Thresh. PC1 is Med Thresh. PC0 is Low Thresh.	Pass
6	Implement button PINC4 to set the device to sleep mode and restart the program if the button is pressed again.	Sleeps on 1 button press and restarts the device on the next press.	PC4 Button toggles the device on and off	Pass
7	Test Circuit and Code	PC4 Button turns the power on/off PC3 resets the program PC2:0 changes threshold levels	PC4 – On/Off PC3 – Reset PC2:0 – Threshold Buttons	Pass

Comments:
The code was tweaked in order to get the on/off button operating.

Date_____Test Engineer_____Witness_____

Test Name	Main Microcontroller Test Code	Test Number	CT_04			
Requirement(s) Tested	2.3, 3.2, 4.1, 6.1, 6.2, 6.3	Verification Method	I	A	D	T
Test Setup	Processor: ATmega328PB Program: Integration of CT_02 and CT_03 References: ATmega328PB Datasheet					

Test Step	Action (Attach test data, diagrams, etc. as appropriate)	Expected Result	Observed Result	Pass Fail
1	Verify Tests CT_02 and CT_03	See Tests CT_02 and CT_03	CT_03 Works CT_02 Works	Pass
2	Initialize PIND2:0 as outputs (for the LED lights)	PIND2:0 are outputs	LED Outputs Represents Threshold Levels	Pass
3	Initialize PINC4:0 as the pull-up input for each button	PINC4:0 are now pull-up inputs for the buttons to use	PC4:0 are buttons set as an input	Pass
4	Initialize PINB1:0 as inputs (for the Bilateral Switches)	PINB1 and PINB0 controls each switch	PB1:0 are audio control switches	Pass
5	Run a loop that counts PINB1:0 from 0 to 3, increment by 1 per second.	t = 0s, PB1:0 = 0 t = 1s, PB1:0 = 1 t = 2s, PB1:0 = 2 t = 3s, PB1:0 = 3 t = 4s, PB1:0 = 0	t= 0s, PB1:0 = 0 t= 1s, PB1:0 = 1 t= 2s, PB1:0 = 2 t= 3s, PB1:0 = 3 t= 4s, PB1:0 = 0	Pass
6	Test Circuit and Code (Switches)	The sound should turn on each side of Aux depending on the state of PB1 and PB0.	The sound turns on and off Aux Left and Right (see test Step 5)	Pass
7	Test Circuit and Code (Buttons)	PC4 Button turns the power on/off PC3 resets the program PC2:0 changes threshold levels	PC4 – On/Off PC3 – Reset PC2:0 – Threshold Buttons	Pass

Comments

Date 3/4/2020 Test Engineer Drew Martins Witness Ryan Dumont

Test Name	Process Implement in Microcontroller	Test Number	CT_05			
Requirement(s) Tested	6.2, 6.3	Verification Method	I	A	D	T
Test Setup	Atmel Studio, Microsoft Visual Studio, AVR programming, C#, Oscilloscope an probes					

Test Step	Action (Attach test data, diagrams, etc. as appropriate)	Expected Result	Observed Result	Pass Fail
1	Perform demonstration PT_04.	See PT_04 Test Plan.		Pass
2	To implement process PT_04 in microcontroller, match variables that are defined using I/O such as analog input and threshold setting to microcontroller memory.	All variables from PT_04 should be accounted for, including cascaded averages, 20 second standard deviations.	See Main Microcontroller Code	Pass
3	Without connecting any I/O such as microphone and analog circuitry, digital switch and threshold switches, use the microcontroller to step through the process using user-defined 2 second ambient level averages.	Monitor 2 second averages saved to microcontroller.	Each 2s Avg is saved into an array.	Pass
4	Verify that after 20 seconds, the oldest 2-second average gets replaced by the newest if the newest is below the interruption threshold.	Verification complete.	The oldest 2s Avg is replaced every 20s.	Pass
5	Verify that after 20 seconds, if the newest 2-second average is above the threshold standard deviation, averaging stops and a high signal voltage is output to the digital switch output pin (not connected) using oscilloscope and probe.	Verification complete.		WIP*
6	Verify that the RESET variable resets the 20 seconds of averaging data when activated and no interrupt is allowed for 20 more seconds.	Verification complete.	The reset button restarts the program from the beginning.	Pass
7	Verify that the OFF button will tell the microcontroller to manually shut down, as opposed to cutting the voltage supply.	Verification complete.	The off button shuts down the microprocessor	Pass

Comments:
*Due to COVID-19, the lab equipment required for this part of the test plan was not present.

Date_____Test Engineer_____Witness_____

Test Plans: Printed Circuit Board

Test Name	PCB Test Layout	Test Number	BT_01			
Requirement(s) Tested	Constraint of maximum size: 3.5” x 2”	Verification Method	I	A	D	T
Test Setup	Cadence OrCAD Suite Eagle Autodesk PCB Designer Tools					

Test Step	Action (Attach test data, diagrams, etc. as appropriate)	Expected Result	Observed Result	Pass Fail
1	Run Cadence OrCAD: Capture CIS and construct a circuit with estimated components necessary for the PCB, including Microcontroller, Digital Switch, approximate analog electronics selections, OP-AMP and connections.	Circuit Layout	Schematic generated.	Pass
2	Import necessary library and padstack files (website: ultralibrarian.com) for components that are not standard in the program.	All components accounted for.	All expected components available with necessary libraries.	Pass
3	Export the schematic layout to a netlist file and load the netlist into OrCAD PCB Editor with a board layout size at the maximum constraint size 3.5” x 2”	Netlist file generated without errors or warnings. Board size in the Editor is 3.5” x 2”. Verify that components can be placed using the netlist.	All components successfully imported into OrCAD PCB Editor. Board size chosen to be maximum constraint size.	Pass
4	Place components on the board in the software and connect wires to demonstrate that all components fit in the maximum constraint size.	All components fit on the 3.5” x 2” layout.	All components with possible trace schemes fit on maximum constraint.	Pass

Comments
Figure 22. PCB Layout: Maximum Size Constraint. Purpose of test: to see that electronics will fit in max constraint size. Figure depicts rectangle of max size on top of pcb to show that board size is the max constraint size.

Date 3/20/2020 _____Test Engineer:_____ Michael Benker _____
_Witness:_____ Stephen Felix _____

Test Name	Initial PCB	Test Number	BT_02			
Requirement(s) Tested	6.1, 6.2, 6.3, Constraint of maximum size: 3.5” x 2”	Verification Method	I	A	D	T
Test Setup	Cadence OrCAD Suite Eagle Autodesk PCB Designer Tools Serpac Enclosures Website					

Test Step	Action (Attach test data, diagrams, etc. as appropriate)	Expected Result	Observed Result	Pass Fail
1	Verify Exact component selections and circuit layouts in ET_06 and ET_08.	See ET_06 and ET_08	Circuit layouts for ET_06 and ET_08 have not been verified completely.	Fail
2	Create new schematic or modify layout from BT_01 to include final component selections. Verify that all components and wire paths are correct.	Schematic components and wire pathways are correct.	In progress as of 3/26/2020.	
3	Run Cadence OrCAD: Capture CIS or Eagle Autodesk PCB designer tools and construct a circuit with final components choices and correct wire pathways.	Circuit layout is in accordance with schematic.		
4	If using Capture CIS, import necessary library and padstack files (website: ultralibrarian.com) for components that are not standard in the program. If using Eagle, ensure that all layout components are available.	All components accounted for.		
5	Export the schematic layout to a netlist file and load the netlist into OrCAD PCB Editor or import the schematic to Eagle PCB Designer and ensure all layout components are available.	Netlist file generated without errors or warnings or all layout components available.		
6	Reduce size of board as much as possible while preserving circuit layout on the board.	Board size is reduced from maximum constraint size.		
7	Increase the board size to fit snug in the next largest available enclosure from Serpac enclosures website. Maintain equal margin sizes at each edge of the board.	Layout size fits in chosen enclosure.		

Comments
Due to Lab restrictions (COVID-19), a decision is being made to produce rough-estimate PCB boards that may be edited using solder, component insertions, etc.

Date_____Test Engineer_____Witness_____

Test Name	PCB Review	Test Number	BT_03			
Requirement(s) Tested	6.1, 6.2, 6.3, Constraint of maximum size: 3.5” x 2”	Verification Method	I	A	D	T
Test Setup	Cadence OrCAD Suite Eagle Autodesk PCB Designer Tools					

Test Step	Action (Attach test data, diagrams, etc. as appropriate)	Expected Result	Observed Result	Pass Fail
1	Complete BT_02 Initial PCB	See BT_02		
2	Review component choices and pathways on PCB layout with teammates.	Component choices and pathways match schematic layouts.		
3	Seek brief consultation with University sources including students, technicians and professors to confirm that PCB is acceptable and that parameters such as linewidth are appropriate for manufacture.	PCB is satisfactory or adjustments must be conducted.		
4	If PCB design is less than satisfactory, adjust as necessary according to advice sought for PCB.	PCB is satisfactory for manufacture.		
5	Seek methods of PCB manufacture and place order.	PCB ordered for manufacture.		

Comments

Date_____Test Engineer_____Witness_____

Test Name	PCB Manufacture Summary	Test Number	BT_04			
Requirement(s) Tested	7.0, constraint size.	Verification Method	I	A	D	T
Test Setup	PCB(s)					

Test Step	Action (Attach test data, diagrams, etc. as appropriate)	Expected Result	Observed Result	Pass Fail
1	Complete BT_03, send PCB out for manufacture and receive manufactured PCB(s).	PCB(s) is mailed out and in possession of the team.		
2	Inspect all components on PCB to verify that the components are as expected.	Components are as expected.		
3	Inspect all pathways on PCB to verify that they are as expected.	Pathways as expected.		
4	Inspect all input and outputs paths on the PCB.	Inputs and outputs as expected.		
5	Verify that PCB fits in selected enclosure.	PCB fits in enclosure with selected battery source(s).		

Comments

Date_____ Test Engineer_____ Witness_____

Test Name	PCB Function Test and Comparison	Test Number	BT_05			
Requirement(s) Tested	6.1, 6.2, 6.3	Verification Method	I	A	D	T
Test Setup	PCB Power Supply Oscilloscope and probes Website: noises.online Sound meter					

Test Step	Action (Attach test data, diagrams, etc. as appropriate)	Expected Result	Observed Result	Pass Fail
1	Complete tests ET_12 and BT_04.	See ET_12 and BT_04.		
2	Install program code to the ATmega328PB on the PCB.	ATmega328PB on PCB is responsive and code is loaded.		
3	Connect wires to the connection ports on the PCB for each connector and connect power supply, ground and probes to respective ports on the PCB as per schematics used in ET_12.	PCB function is comparable to ET_12 function.		
4	Conduct test ET_12 but for PCB and record experimental results.	PCB function is comparable to ET_12 function.		
5	Compare results of PCB function test with ET_12.	PCB function is comparable to ET_12 function.		

Comments

Date_____Test Engineer_____Witness_____

Test Plans: Enclosure

Test Name	Enclosure Selection	Test Number	NT_01			
Requirement(s) Tested	7.0, Customer Req. 1	Verification Method	I	A	D	T
Test Setup	~~OrCAD Design~~ ~~Battery/Circuit Board Dimensions~~ Jameco.com					

Test Step	Action (Attach test data, diagrams, etc. as appropriate)	Expected Result	Observed Result	Pass Fail
1	Determine possible enclosure sizes from enclosures website.	Several enclosure sizes that meet size constraint.	Enclosure chosen and ordered: Hammond 2138222 (see datasheet figure)	Pass
2	Verify Battery Selection following battery consumption test ET_09.	Battery is chosen and ready.	Due to lab availability constraints due to COVID-19, the decision has been made to choose the enclosure without completing this test.	Fail
3	Verify size of PCB initial layout	Size should be within customer's constraints.	PCB layout will be built under the constraint of enclosure choice.	Pass
4	Once length and width measurements of PCB are verified, pick an enclosure with correct length and width to house the board.	Length/Width of PCB and battery should be smaller than enclosure.	PCB layout will be built under the constraint of enclosure choice.	Pass
5	Make sure enclosure has enough height space within to allow battery and PCB to be placed on top of one another.	Enclosure should be high enough to hold battery and circuit board when stacked on top of each other.	Under consideration. Battery may be selected given form factor constraint.	Pass

Comments

Date 3/25/2020 Test Engineer Ryan Dumont Witness Michael Benker

Test Name	Enclosure Outfit for Component, Holes, I/O	Test Number	NT_02			
Requirement(s) Tested	1.1	Verification Method	I	A	D	T
Test Setup	Drill Various Drill Bit Sizes Enclosure All Switches and Exterior Components					

Test Step	Action (Attach test data, diagrams, etc. as appropriate)	Expected Result	Observed Result	Pass Fail
1	Refer to figure 10 for tentative component placement.	A visual of proper placement will be obtained.		
2	Drill the Proper size holes for each component in their appropriate place designated in figure 10.	Proper mounting locations will be created.		
3	Mount all the components and make sure none interfere with each other.	All exterior components will be placed on enclosure.		

Comments

Date_____Test Engineer_____Witness_____

Test Name	Enclosure Meets PCB and Battery Test	Test Number	NT_03			
Requirement(s) Tested	1.1	Verification Method	I	A	D	T
Test Setup	Finalized Circuit Board Battery Enclosure					

Test Step	Action (Attach test data, diagrams, etc. as appropriate)	Expected Result	Observed Result	Pass Fail
1	Place and secure battery in enclosure laying it flat on the bottom.	Battery should fit		
2	Place and secure circuit board on top of battery.	Battery and circuit board should fit together		

Comments

Date_____Test Engineer_____Witness_____

Figures: Electronics Testing

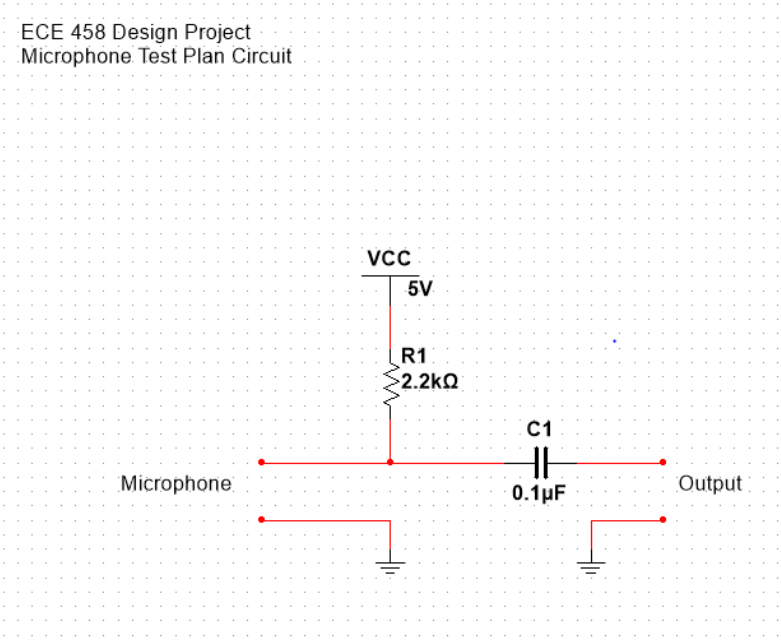

Figure 28 Microphone Circuit

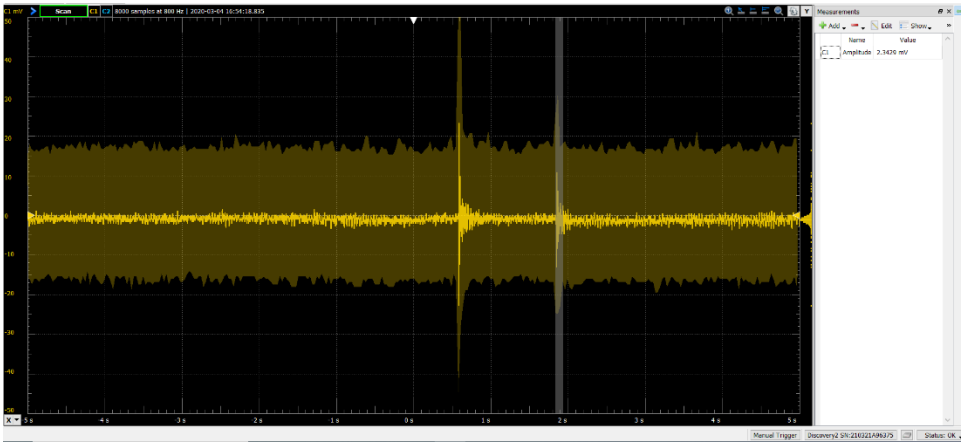

Figure 29 Microphone Test

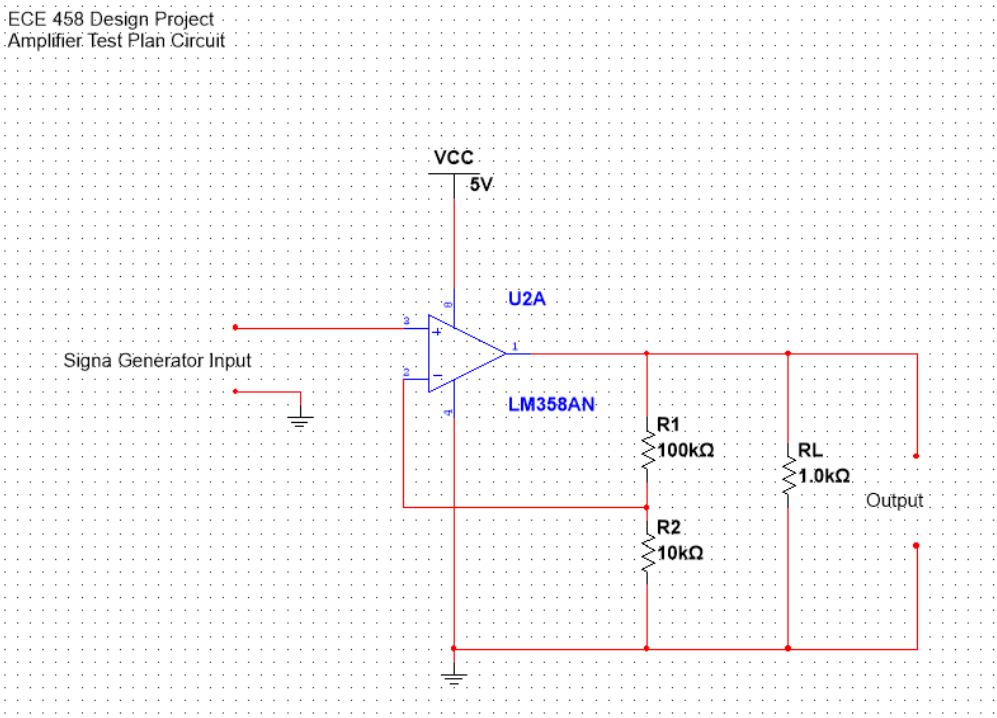

Figure 30 Amplifier Test Plan Circuit

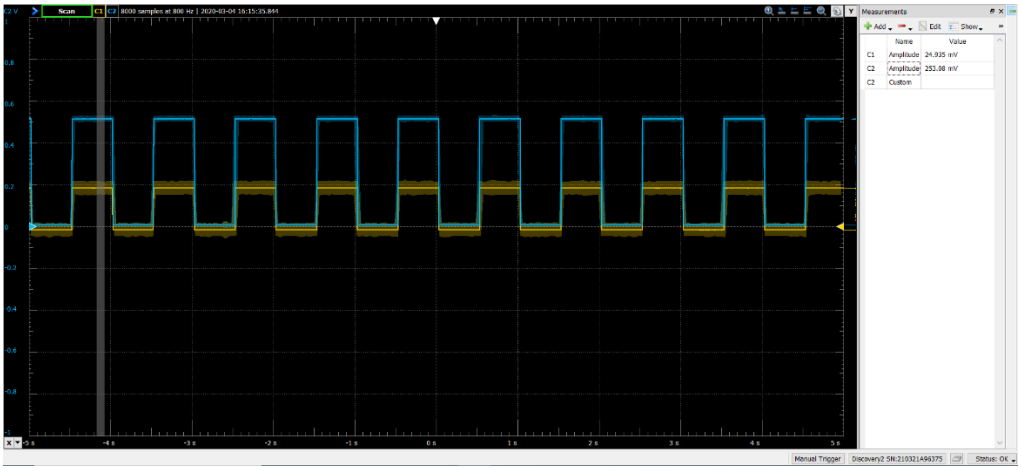

Figure 31 Amplifier 50mV Input

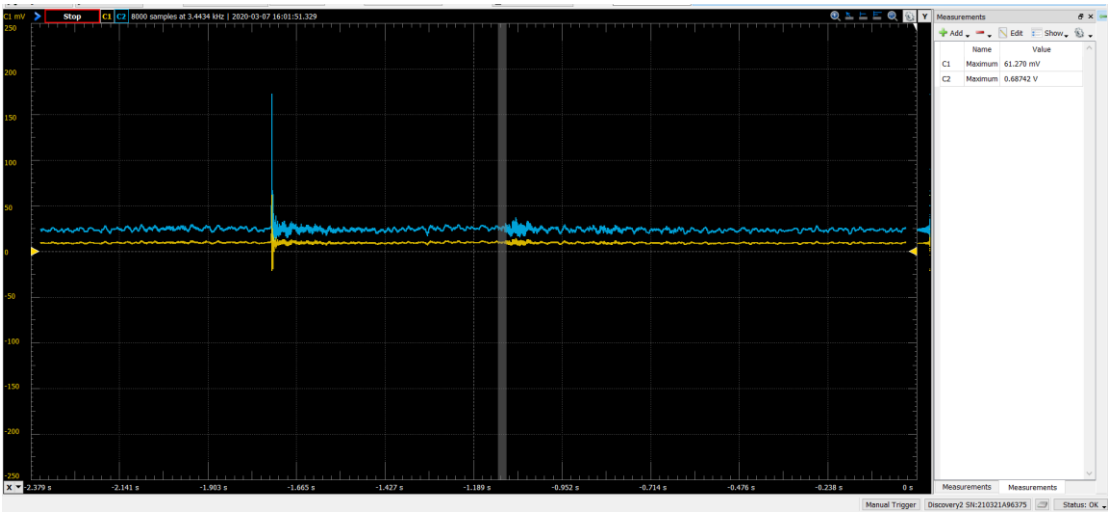

Figure 32 Microphone & Amp Integration

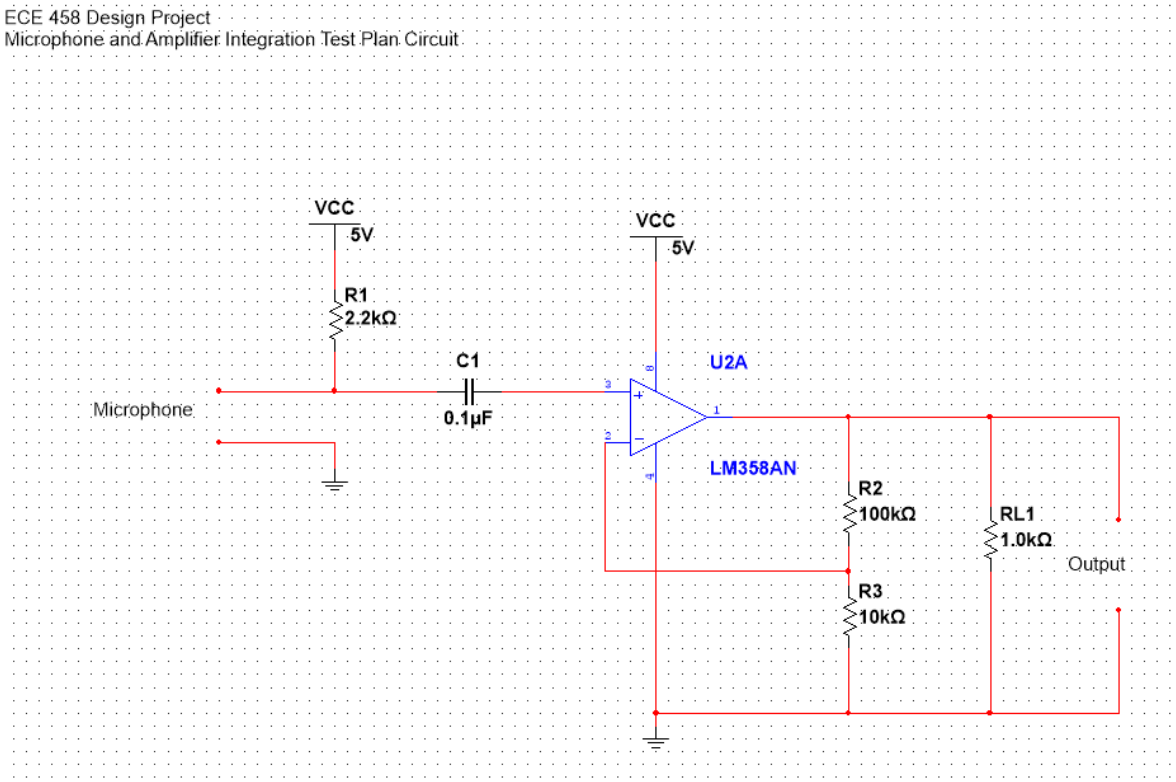

Figure 33 Microphone and Amplifier Integration Circuit

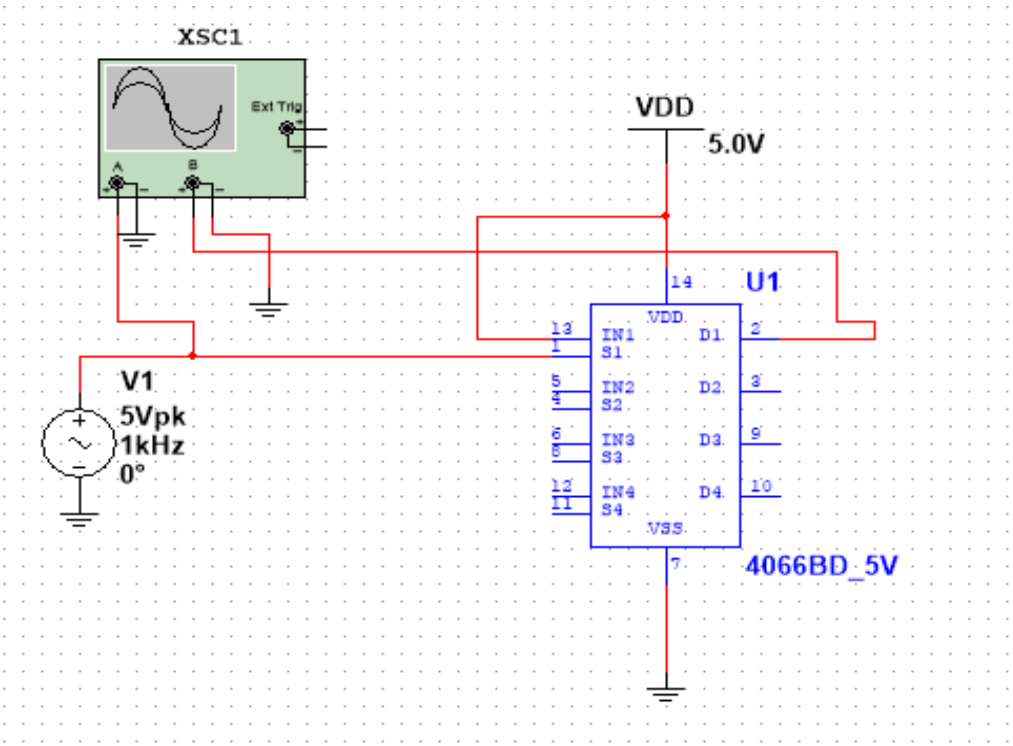

Figure 34 Digital Switch Testing Circuit

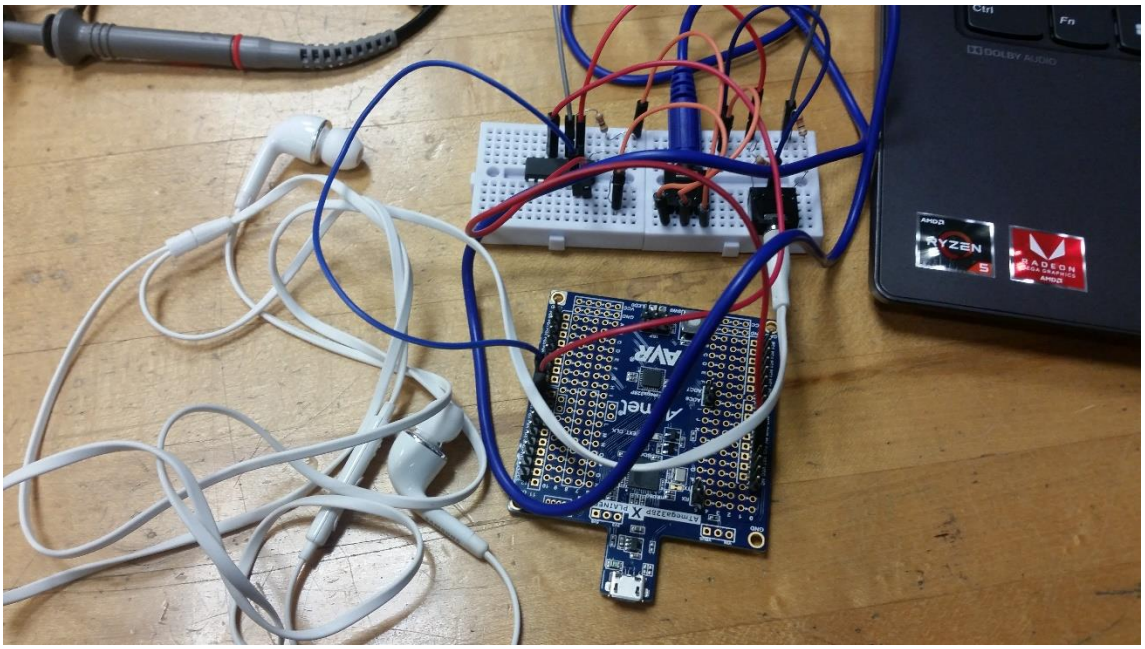

Figure 35 Auxillary Switch Test

Figures: Processing Testing

Ambient	Quiet	Medium	Loud	Senior Design Room
49.6	57.7	61.4	71.2	
49.1	54.6	57.3	58.5	
49.9	52.6	60.1	92.5	
49.6	54.1	52.5	75.2	
49.1	52.6	55.6	70.5	
49	57.7	56.3	60.6	
49.4	52.6	61.3	68.3	
49.9	54.6	55.6	71.6	
48.9	55.6	56.5	66.7	
49.6	50.3	61.3	64.5	
49	55.6	55.6	77.4	
49	50.3	65	77.9	
49.2	55.6	59.7	82.5	
49.6	53.2	63.8	80.5	
49.3	50.9	59.3	68.5	
50.3	51.5	62.5	82.3	
49.1	52.2	65.1	71.7	
49	50.3	60.2	69	
49.1	52	53.9	74.6	
49	51.5	64.4	84.2	
49.1	57.2	67.4	83.8	
49	55.6	66.1	83.7	
48.9	53.7	63.7	69.3	
49.2	50.3	65.6	82	
49	53.2	57.8	91.2	
49.3	51.5	58.3	75.2	
48.9	52.3	64.4	74.9	
49.4	51.8	63.2	94.5	
49.1	53.1	65.1	75.4	
49	51.3	64.9	74.6	

Figure 36. Data Collected using Sound Meter in One Environment

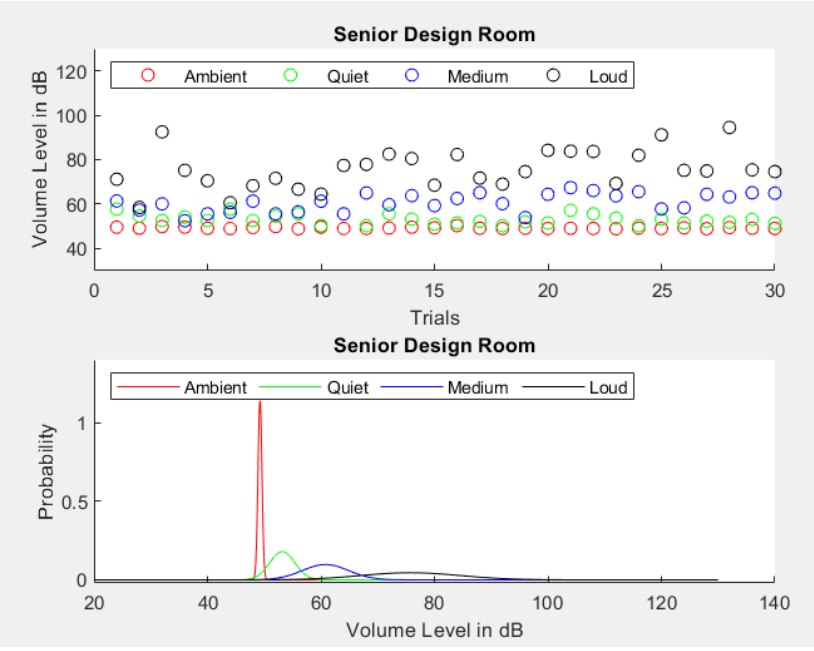

Figure 37. Data Plot with Sound Meter

T =

4×3 table

Labels	Averages	Stdevs
'Ambient'	[49.2533]	[0.3471]
'Quiet'	[53.1833]	[2.2130]
'Medium'	[60.7967]	[4.0985]
'Loud'	[75.7600]	[8.7974]

Figure 38. Standard Deviations of Environment

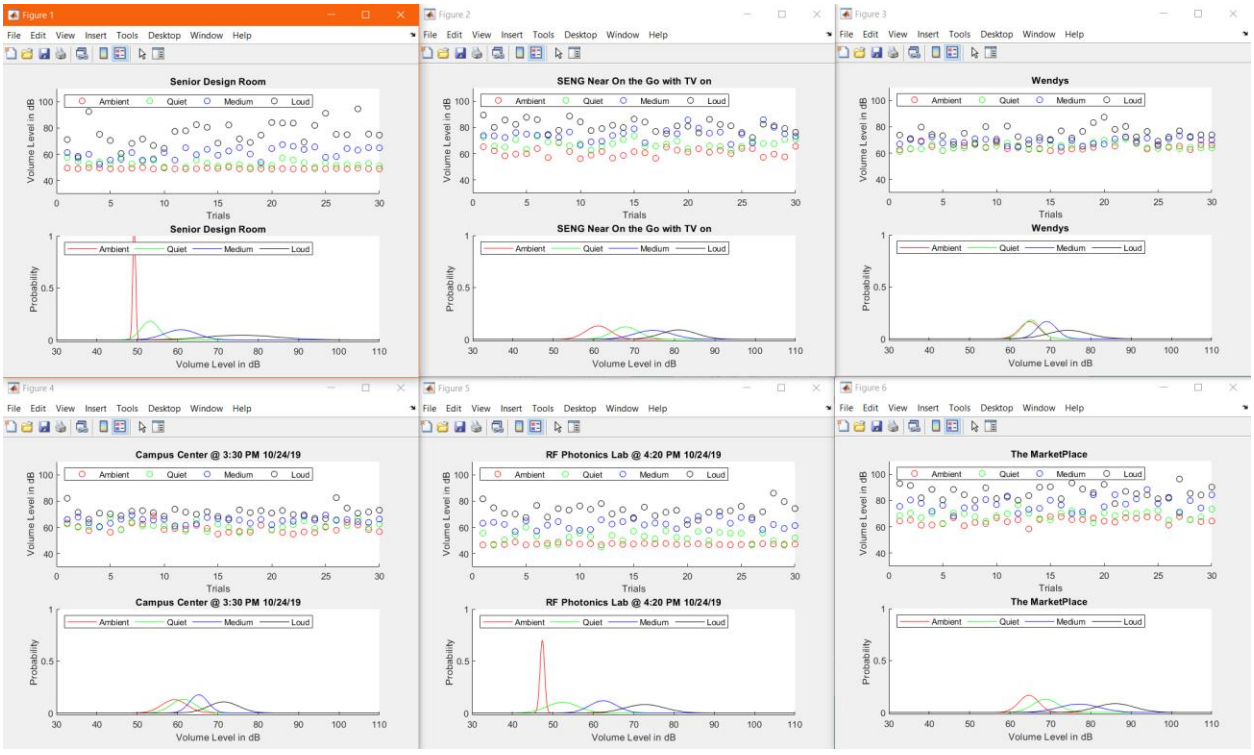

Figure 39. Multiple Environments

```
QuiStdfromAmb =

    4.1601


MedStdfromAmb =

   12.1691


LouStdfromAmb =

   24.0285
```

Figure 40. Average Standard Deviations of Each Threshold Level

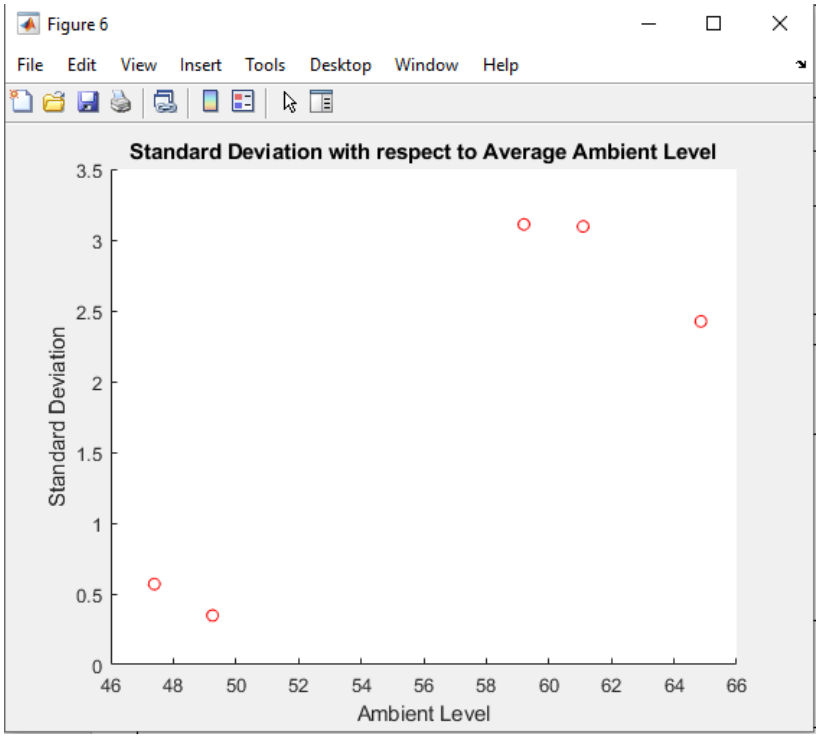

Figure 41 Standard Deviation vs Ambient

$$T_i = \sum_{1}^{n}((Ave_k - Ambient)/\sigma)/n$$

Figure 42 Threshold based on Standard Deviation

```
37.075
36.99
38.63
-----------------------------------------


Time is 20 seconds
Enter a minimum value (dB) between 30 and 90:
30
Enter maximum value (dB) between 30 and 90:
45
The two second average is 37.005
37.005
36.965
40.505
42.55
37.205
41.09
37
37.075
36.99
38.63
The twenty second average is 38.50149999999999
The twenty second standard deviation is 1.999412476203947
-----------------------------------------


Time is 22 seconds
Enter a minimum value (dB) between 30 and 90:
```

Figure 43 PT_04 Successful Interrupt

```
-----------------------------------------


Time is 30 seconds
Enter a minimum value (dB) between 30 and 90:
35
Enter maximum value (dB) between 30 and 90:
40
The three second average is 36.92333333333333
36.946666666666665
36.943333333333335
36.89
36.903333333333336
36.846666666666664
36.94
36.97666666666667
36.986666666666665
36.943333333333335
37.02333333333333
36.92333333333333
0
The thirty second average is 33.860277777777775
The thirty second standard deviation is 10.209353142560314
-----------------------------------------
```

Figure 44 30 second average vs 3 second average

```
int n = inputdata_twenty.Length;
twenty_avg = average(inputdata_twenty, n);
```

Figure 20 Call Recursion

```
twenty_avg = inputdata_twenty.Average();
Console.WriteLine("The twenty second average is {0}", twenty_avg);
```

Figure 46 Call Iterative

```
----------------------------------------


Time is 20 seconds
Enter a minimum value (dB) between 30 and 90:
35
Enter maximum value (dB) between 30 and 90:
37
The two second average is 35.495
35.495
35.535
35.475
35.53
35.54
35.495
35.52
35.49
35.46
35.48
The twenty second average is 35.501999999999995
The twenty second standard deviation is 0.026191601707417397
----------------------------------------
```

Figure 47 Recursive Average Result

```
----------------------------------------


Time is 20 seconds
Enter a minimum value (dB) between 30 and 90:
35
Enter maximum value (dB) between 30 and 90:
37
The two second average is 35.505
35.505
35.495
35.5
35.47
35.485
35.475
35.51
35.53
35.53
35.425
The twenty second average is 35.49249999999999
The twenty second standard deviation is 0.02960152023123229
----------------------------------------
```

Figure 48 Iterative Averaging Result

```
static double avgRec(double[] inputdata_two, int i, int n)
{

    // Last element
    if (i == n - 1)
        return inputdata_two[i];

    // When index is 0, divide sum
    // computed so far by n.
    if (i == 0)
        return ((inputdata_two[i] + avgRec(inputdata_two, i + 1, n)) / n);

    // Compute sum
    return (inputdata_two[i] + avgRec(inputdata_two, i + 1, n));
}

static double average(double[] inputdata_two, int n)
{
    return avgRec(inputdata_two, 0, n);
}
```

Figure 49 Recursive Methods

```
Time is 0 seconds
Enter a minimum value (dB) between 30 and 90:
35
Enter maximum value (dB) between 30 and 90:
40
Time Taken: 3032
The two second average is 37.115
37.115
```

```
Time is 0 seconds
Enter a minimum value (dB) between 30 and 90:
35
Enter maximum value (dB) between 30 and 90:
40
Time Taken: 6303
The two second average is 36.925
36.925
```

```
Time is 0 seconds
Enter a minimum value (dB) between 30 and 90:
35
Enter maximum value (dB) between 30 and 90:
40
Time Taken: 4123
The two second average is 37.075
37.075
```

Figure 50 Average Timing

Figures: Coding Testing

ADC Test Code (CT_01)			
Vref (Aref)	Vin (DC)	adc_value (Decimal)	adc_value (Binary)
5.0 V	0 V	0	0b0000000000
5.0 V	1 mV	0	0b0000000000
5.0 V	3 mV	0	0b0000000000
5.0 V	5 mV	1	0b0000000001
5.0 V	10 mV	2	0b0000000010
5.0 V	15 mV	4	0b0000000100
5.0 V	20 mV	6	0b0000000110
5.0 V	30 mV	8	0b0000001000
5.0 V	40 mV	11	0b0000001011
5.0 V	50 mV	13	0b0000001101
5.0 V	60 mV	16	0b0000010000
5.0 V	80 mV	18	0b0000010010
5.0 V	100 mV	23	0b0000010111
5.0 V	150 mV	31	0b0000011111
5.0 V	200 mV	44	0b0000101100
5.0 V	300 mV	68	0b0001000100
5.0 V	400 mV	86	0b0001010110
5.0 V	500 mV	108	0b0001101100
5.0 V	600 mV	130	0b0010000010
5.0 V	800 mV	167	0b0010100111
5.0 V	1 V	208	0b0011010000
5.0 V	1.25 V	258	0b0100000010
5.0 V	1.5 V	314	0b0100111011
5.0 V	1.75 V	367	0b0101101111
5.0 V	2 V	412	0b0110011100
5.0 V	2.25 V	465	0b0111010001
5.0 V	2.5 V	515	0b1000000011
5.0 V	2.75 V	569	0b1000111011
5.0 V	3 V	620	0b1001101100
5.0 V	3.25 V	670	0b1010011110
5.0 V	3.5 V	723	0b1011010011
5.0 V	3.75 V	767	0b1011111111
5.0 V	4 V	828	0b1100111100
5.0 V	4.25 V	877	0b1101101101
5.0 V	4.50 V	929	0b1110100001
5.0 V	4.75 V	986	0b1111011010
5.0 V	5 V	1023	0b1111111111

Figure 51 ADC Test Code

Figures: Printed Circuit Board Testing

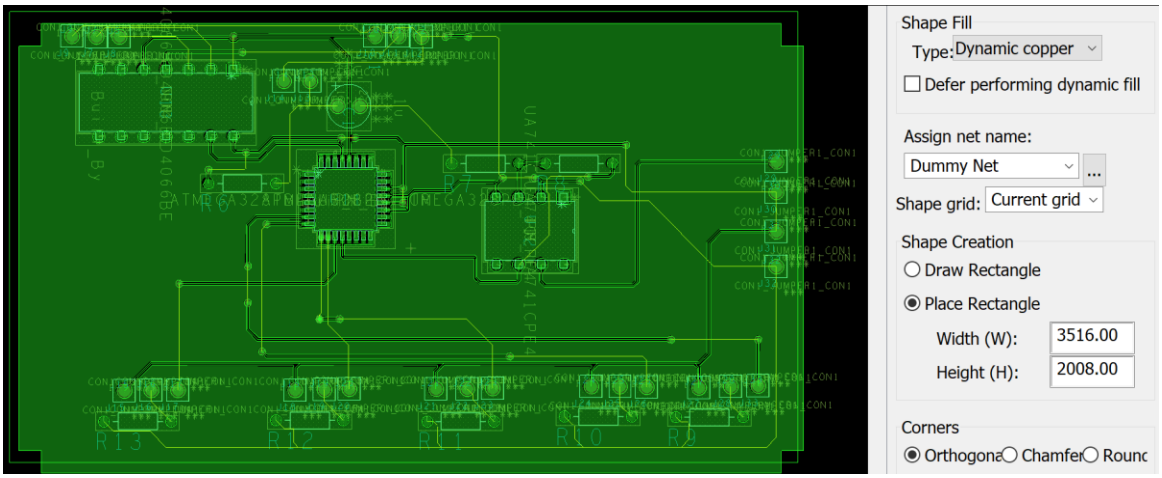

Figure 52. PCB Test Layout for Maximum Constraint Size (3500mil x 2000mil

Figures: Enclosure Testing

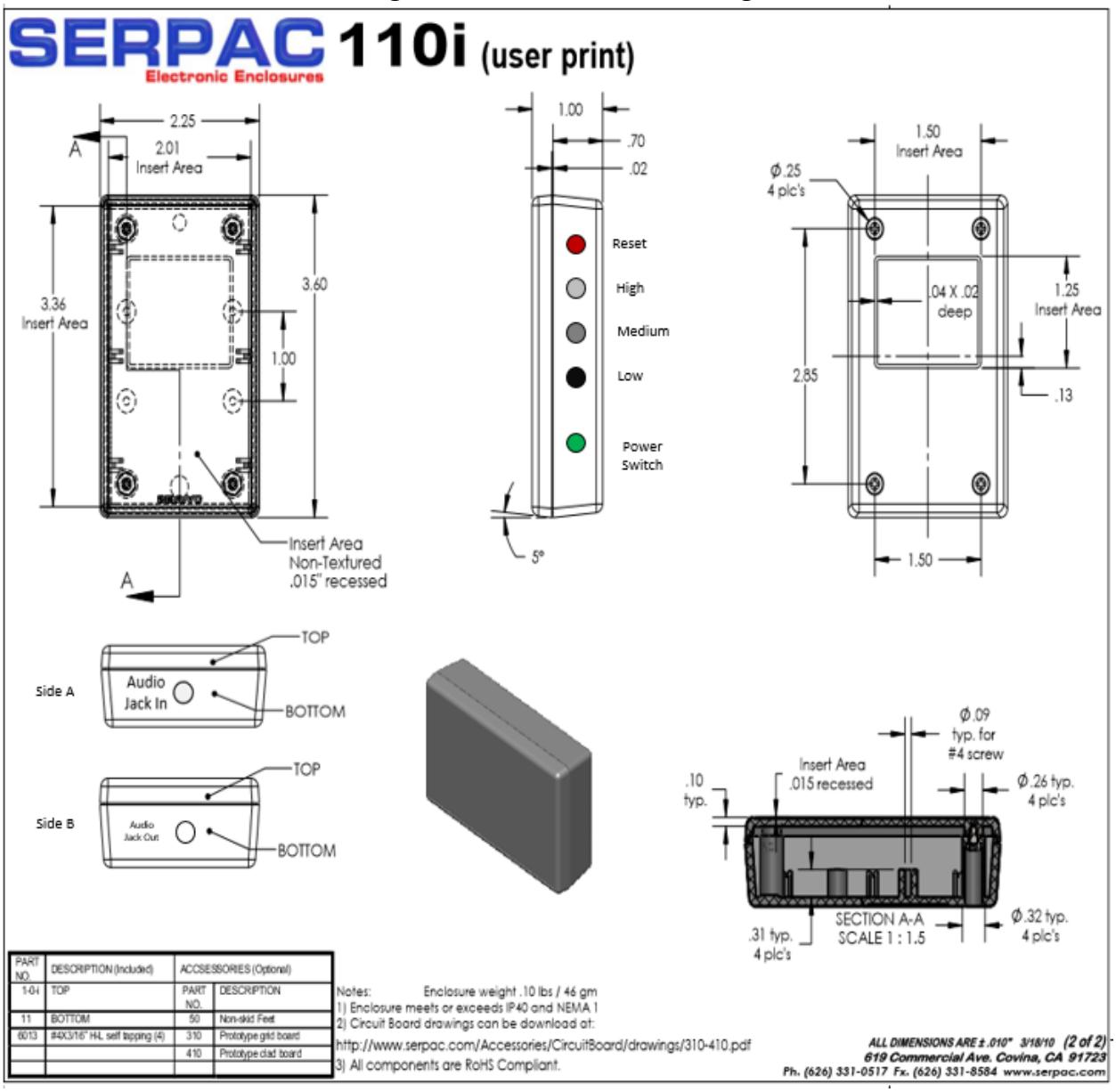

Figure 53 Enclosure Outline

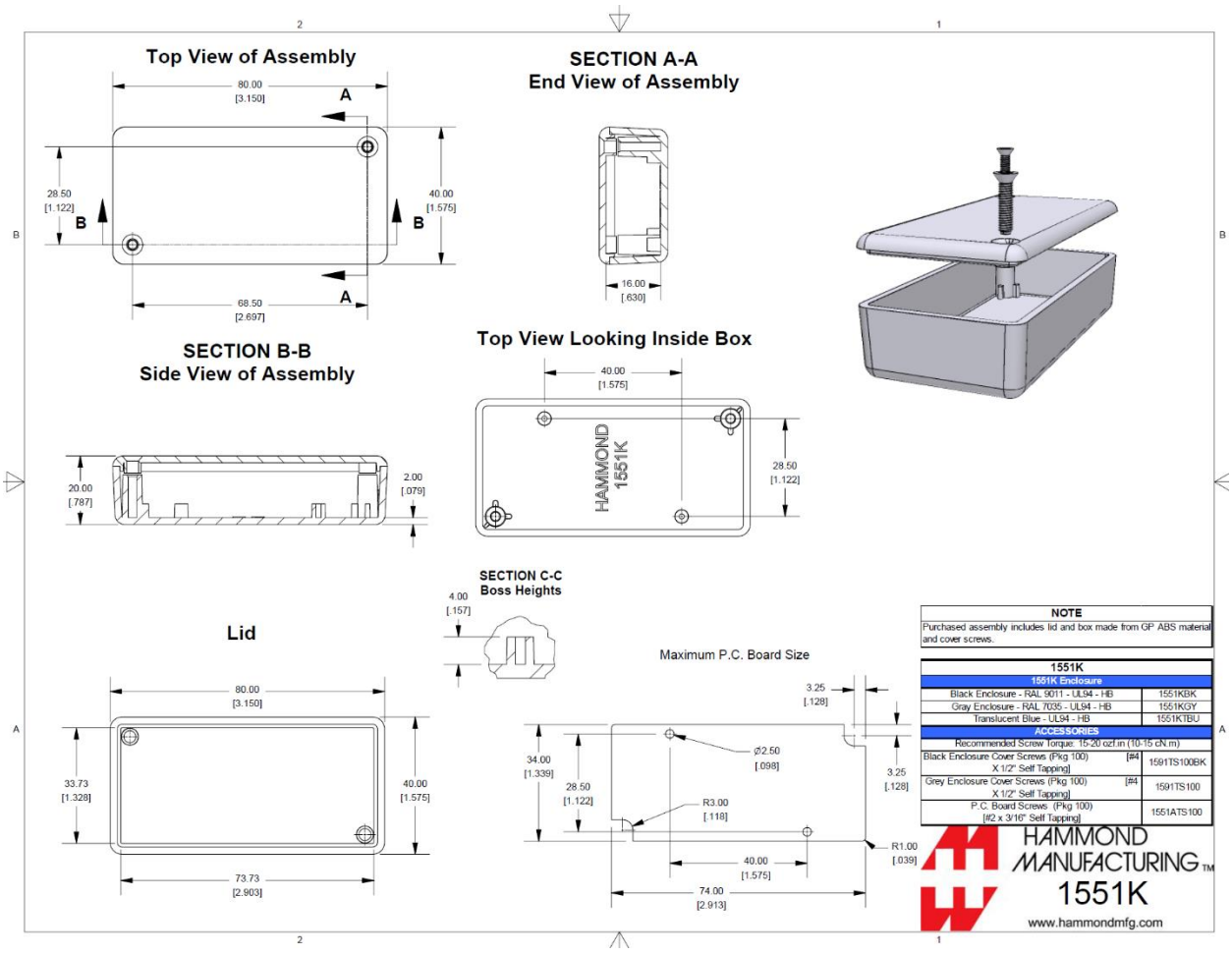

Figure 54. Selected Enclosure: Hammond 2138222