March 9, 2016

Dr. Andrew Rawicz
School of Engineering Science
Simon Fraser University
Burnaby, British Columbia
V5A 1S6

Re: Design Specification for AlarmSense System

Dear Dr. Rawicz

Attached you will find the document ‘Design Specification for AlarmSense System’, which outlines the design for our ENSC 440W project. Our goal is to design a system protocol to allow workers in noisy industrial settings to remain safe and work efficiently while protecting their hearing.

The design specification includes the details of implementing the requirements laid out in our functional specification. The design includes the physical design, hardware, and software of individual units, as well as the design of a charging station and app for controlling the system.

Ekho Systems is composed of four fourth year engineering students: Russell McLellan, Taylor Robson, Gordon Ho, and Adrian Tanskanen. If you have any questions or concerns, please contact us my email at rmclella@sfu.ca.

Sincerely,
Russell McLellan
Ekho Systems

[illegible]

Design Specification for AlarmSense System

Group 18: Russell McLellan
Taylor Robson
Gordon Ho
Adrian Tanskanen

Contact Person: Russell McLellan
rmclella@sfu.ca

Submitted: March 9, 2016
Revision: 1.11

Abstract

The AlarmSense system, a system of noise cancelling hearing protection has been designed for increasing safety in industrial workplaces while not compromising auditory alarm systems or worker to worker communication. Workers wearing AlarmSense units will be able to hear alarms and communicate with other workers, but harmful industrial noise will be cancelled out. The AlarmSense System is comprised of the individual headphone units, a charging briefcase, and a mobile app for system control.

The unit is primarily comprised of noise-cancelling headphones, a noise cancelling microphone, and a microcontroller to do audio processing. The unit has hardware and software filters to detect when certain frequencies are read by the microphone. These frequencies, corresponding to the frequencies of alarms, will be passed through the noise cancelling headphones. As the microcontrollers used in the system us different voltage ranges than those outputted by the microphone or required by the headphones, an analog circuit has been design to interface these components.

The unit will also read in and wirelessly transmit speech from the noise cancelling microphone to other AlarmSense units. Using an onboard GPS receiver, the distance between any two units will be known to the units. This distance will be used to attenuate the volume of the speech received by the unit, making communication more natural.

As the AlarmSense System is designed for use in industrial worksites with many workers, a central charging hub is needed. A briefcase with the capability to charge up to 12 units has been designed. The briefcase also offers the ability to control what alarm frequencies are being detected by the unit.

For easy control of the system by a foreman, a mobile app has been designed. The app will allow a foreman to see the battery life and connection status of all units on the network. The app will also allow the foreman to measure the frequency of alarms and send this value wirelessly to all units within range.

The design for the AlarmSense System is for both a prototype system to be demonstrated in April 2016, and for a full system to be put into market. This specification deals with both version of the product, and specifies what parts of the design are for either the prototype or complete unit.

Table of Contents

List of Figures

List of Tables

Glossary

Term	**Meaning**
Alarm	An auditory alert of a specific frequency present in a worksite
AlarmSense System	The complete product, including the unit, briefcase, and app
App	A smartphone program used for managing the AlarmSense System
Briefcase	A portable charging station for AlarmSense System units
Foreman	A manager of industrial workers responsible for safety
PPE	Personal protective equipment, including hearing protection
Unit	The part of the AlarmSense system worn by every worker, including headphones, a microphone, and a microcontroller
Worker	An individual using an AlarmSense System unit
Worksite	An area where loud industrial work is carried out

1. Introduction

The AlarmSense System is a system of active noise cancelling headphones to be used as PPE for hearing protection in industrial worksites which will allow workers to hear auditory alarms and communicate with each other despite the background noise. The goal of the system is to create safe, efficient worksites. The system comprises the unit, worn by each worker, and a briefcase for charging and an app for control of the system. The units consist of active noise cancelling headphones, a noise cancelling microphone, and a microcontroller for real time audio processing. The app will allow the foreman to check the status of the system and update the frequencies of the alarms. This design specification outlines the design of the unit, briefcase, and app.

1.1 Scope

This document details the design of the AlarmSense System. The design of the system corresponds to the functional requirements laid out in the 'Functional Specification for AlarmSense System' [1]. References to requirements in this document are listed in the functional specification. Included in this document are both the designs for a prototype system for live demonstration and the designs for a full system for mass production. This document will specify whether each design is for the prototype or full system.

1.2 Intended Audience

This document is to be used by Ekho Systems engineers. Engineers are advised to consult this document when implementing the system or interfacing subsystems, as well as when testing the system. The attached Test Plan details the tests to be run to ensure the system meets its requirements.

2. Overall System Design

Noise cancelling headphones and microphones will be used for this project. As these technologies have already been developed by companies such as Bose and Sony, we will not be developing them.

Our product seeks to combine noise cancelling headphones and microphones with a system of hardware and software to allow auditory alarms and worker communication past hearing protection. These two goals are independent, and have independent subsystems. The full system design for a unit is shown in Figure 1.

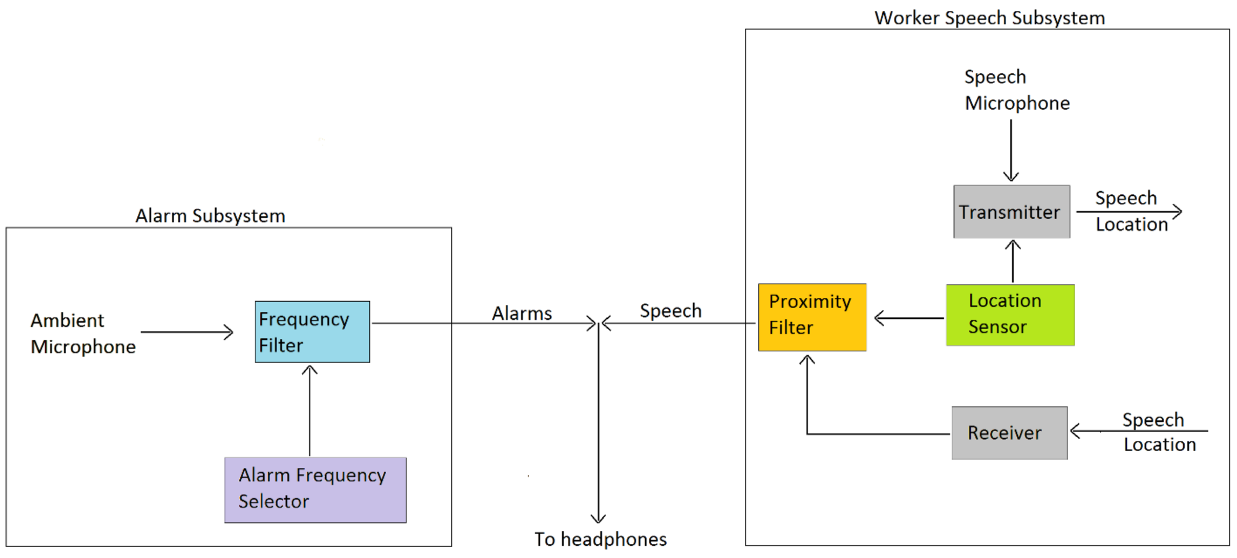


Figure 1: Overall system design

The alarm subsystem contains a non-noise cancelling microphone which will pick up all ambient noise and filter out the alarm frequency based on its frequency. The frequency of the alarm will be selected through either a mobile app, detailed in Section 5 or through a charging briefcase, detailed in Section 6. The frequency filter will be a combination of hardware and software, detailed in Section 3.2.3 and Section 3.3.4.

The worker speech subsystem contains a noise cancelling microphone which will pick up the worker's speech and not the ambient noise. This speech signal will then be wirelessly transmitted to all other units within range, as well as the position from which the signal was sent. Mirroring the transmitter is a receiver, which will pick up the signals sent out by other units. A proximity based filter will attenuate the incoming speech signal based on the distance between the two devices as picked up by the location sensors. The design of the transmission and reception protocols is discussed in Section 3.2.3, the location sensor and proximity filter in Section 3.3.10 and Section 3.2.3, and the microphone detailed in Section 3.1.3.

The prototype under construction for demonstration will not be of the whole system. The charging briefcase will not be developed and the app will not have all features detailed. The unit will not be complete as shown in Figure 1 but will be demonstrated with three microcontrollers, two for the alarm subsystem and the receiving half of the worker speech subsystem, and one for the transmitting half of the worker speech subsystem. The prototype system is shown in Figure 2.

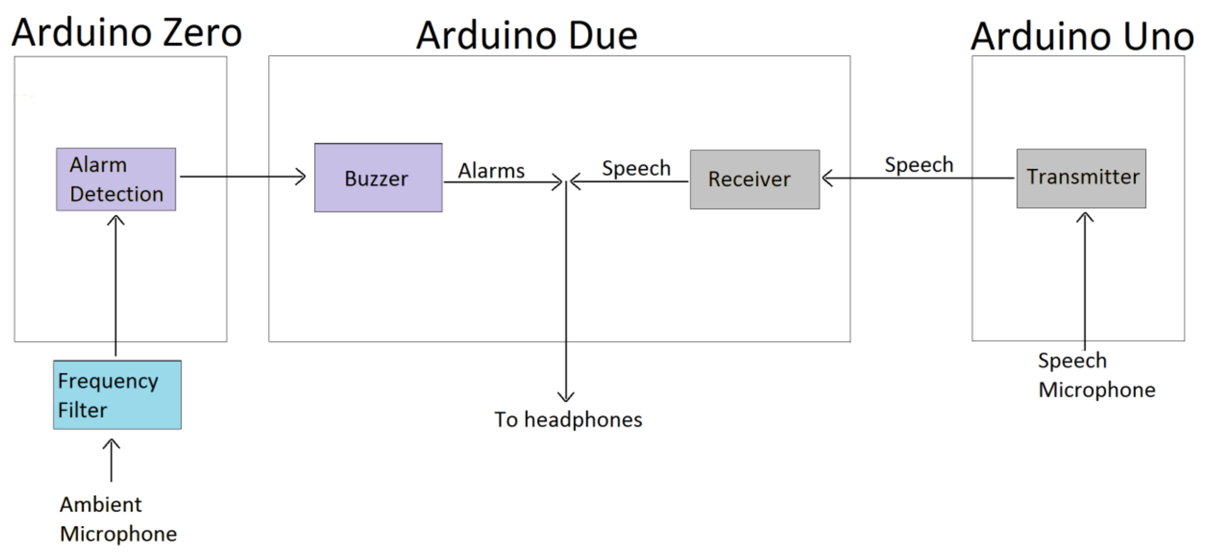


Figure 2: Prototype system design

The difference between the full system and the prototype system is broken down part by part in the following sections.

3. Unit Design

3.1 Mechanical Design

The AlarmSense System is designed to adhere to a strict set of durability, sustainability, and ergonomic requirements. This section documents the physical design of the headset unit and details how these requirements are integrated into the design.

3.1.1 Headphone Safety Design

The AlarmSense System is designed specifically as a safety device, and this section details how the headphones are designed to ensure the safety of the user. The headphones are designed so as to have an electrical and a mechanical component, both of which dampen harmful industrial noises. The mechanical component of the headset is based off of a standard set of hearing protection, commonly found on worksites. This hearing protection is designed to follow safety standard [R1-II], which states that all exterior noise will be reduced to at most 85dB regardless of whether the interior electronics are working. The active noise cancelling resulting from the electrical component is then integrated into the protective hearing to provide another layer of safety. The design of the electrical component is detailed in the Hardware Design section.

3.1.2 Sustainability Design

The headphones make use of an innovative and practical design in order to maximise sustainability. The final design will be simple, with only one large screw on each earmuff holding together the internal parts of the headphones. Once manufactured, all of the electrical hardware for the headphones will be integrated into the earmuff. This placement of the hardware will provide an additional noise dampening. If the electronics are broken, they will be simple to access and replace, and will not require replacing the whole unit. This replacement will be possible because of the innovative single screw assembly, which is shown in Figure 3.

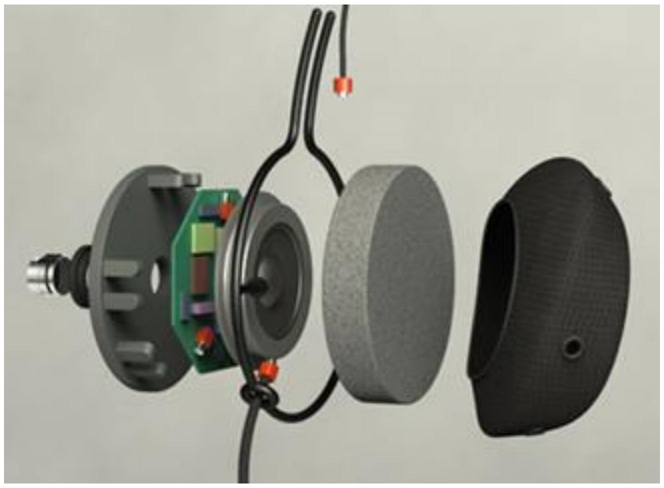


Figure 3: Simple design to allow parts to be replaced [2]

Fig.3 shows the interior of one earmuff from a headset. All of the interior parts can be accessed and replaced by removing one large aluminum screw [2]. This design will greatly increase the lifespan of the unit, especially considering the rough industrial environments that the product will be used in. The lifespan requirement is 5 years, as given by [R6-III].

The unit will be manufactured using sustainable materials. The plastic parts of the AlarmSense System will all be constructed using starch-based completely biodegradable polymers (SCBPs). SCBPs are a widely used for applications in biomedical and environmental instruments due to their robustness [3]. This robustness makes it the perfect material for our applications in heavy industrial settings.

3.1.3 Microphone design

As per requirement [R31-II], the microphone is designed to allow hands free communication. The microphone will be attached to the headphone and is voice activated so the user can operate it regardless of what their hands are doing. Additionally, the microphone volume control will be large enough to control with a pair of gloves. The full unit with the microphone is shown below in Figure 4.

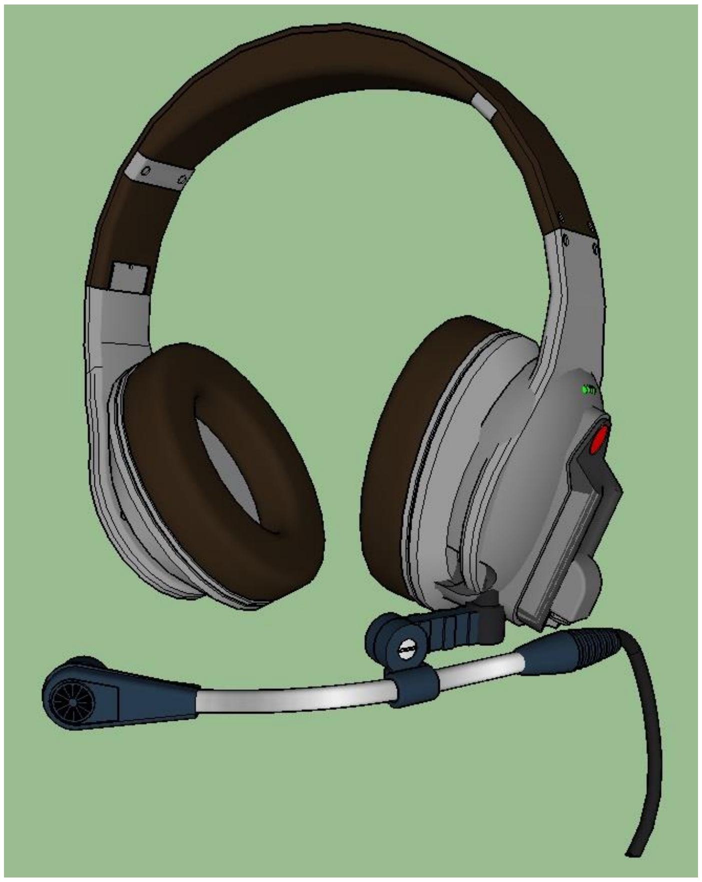


Figure 4: Unit with voice activated microphone

3.2 Software Design

The software in the unit must detect alarm frequencies and transmit speech wirelessly. Due to the fact that processing will be done in real time, the software must be optimized for speed as per requirement [R27-I]. This optimization also reduces the computing required resulting in longer battery life, which leads to a smaller battery being required to meet the 16 hour battery life requirement [R33-II]. The main components of software were active noise cancelling, selecting and updating the alarm frequency to be detected, detecting alarms, transmitting wireless signals, and proximity sensing. Some of these functions have been developed by other companies, such as active noise cancelling headphones, and these functions are incorporated into the design.

3.2.1 Active Noise Cancelling

The purpose of the active noise cancelling is to provide a more comfortable environment for the end user. The active noise cancelling is done in the headphones. For the prototype, Bose QuietComfort 25 headphones will be utilized. These headphones were selected as they are high quality and are proven to cancel out drones such as the ones that workers can hear when wearing standard hearing protection. Tests will be run to ensure compliance with [R17-II], which states that noise must be reduced by 30dB over standard hearing protection.

3.2.2 Selecting an Alarm Frequency

The mobile application, designed in Section 5, sends the alarm frequency that needs to be detected to the Arduino Zero. The Bluetooth serial input will then interrupt the current process and update the global variable for the alarm frequency, update the bandwidth of frequencies to be read in, and update the pins to select the required hardware filter. The Arduino Zero will communicate with the Arduino Due to select the output alarm format. The bandwidth selection is important as it reduces the processing power required, giving a longer battery life and will help ensure a 16 hour battery life in compliance with [R33-II]. Selection of the hardware filter is done through checking to ensure the frequency is within an acceptable range, and then sending a signal to a multiplexer to select the filter that has the correct passband. After receiving the alarm frequency value from the mobile application, the Arduino Zero causes an interrupt on the Due to reinitialize it with new variables. The Arduino Due has pre-programmed alarms that create unique tones that are sent to the user through the headphones, and these tones will trigger when the Arduino Zero detects an alarm frequency. The interactions between the devices for initial frequency setup can be seen in Figure 5.

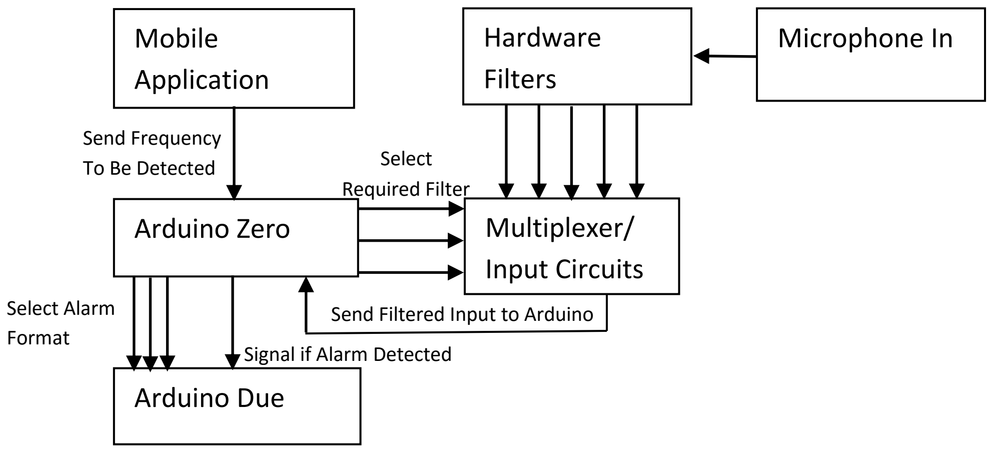


Figure 5: Block diagram of alarm detection system

3.2.3 Alarm Detection

The Arduino Zero comes with a standard library that can be utilized to detect frequencies. The Arduino Zero will receive ambient noise from a microphone in discrete samples. The main loop in the code detects the frequencies in the noise. If the alarm frequency is detected, the Arduino Zero then waits 10μs and detects the frequency again. If the alarm frequency is not detected again, the Arduino Zero exits the loop and starts the initial loop again. In the case that the Arduino Zero detects the alarm frequency for the second time, it sends a signal to the Arduino Due to indicate an alarm, and starts a clock. The Arduino Zero continues to loop and check for alarms, resetting the clock every time the alarm is detected. If an alarm is not detected for one second, then it discontinues the signal to the Due, and re-enters the main loop of the program.

3.2.4 Transmitting/Receiving Software

[R11-II], [R21-I], and [R22-II] lay out that the units should be able to sense when they are in range of another unit, and the transmission protocols. The hardware used to transmit the signal is detailed in Section 3.3.9.

For the prototype system, worker speech is read into the Arduino Uno and must be transmitted to the Arduino Due. Both the transmitter and receiver are set up using the same addresses in the TMRh20 library [4].

The Arduino Uno will read in 2-byte audio samples, which are the smallest sample size that can reproduce human speech. The samples will be read in at 8.3 kHz. 8.3 kHz was chosen as human speech does not exceed 4kHz, and the Nyquist-Shannon sampling theorem states that in order to recreate an analog signal from digital signals, the sample rate must be higher than twice the highest frequency seen in the analog signal [5].

Once 16 samples are taken by the Arduino Uno, the packet will be sent to the Arduino Due. Once out of every 100 packets, the Arduino Uno will wait for the packet confirmation from the Arduino Due. This wait is done every 100 packets to keep the system speed high, but also to allow the microcontroller to sense when it is out of range of the remainder of the system. The unit must know when it is connected to another unit so it can display this information to the worker, as laid out in [R11-I].

Each packet will be read into the Arduino Due as the packet is available. The Arduino Due will output the data at the same rate it was read in. This will be accomplished by having an interrupt running at 8.3kHz which outputs the next value in sequence.

The Arduino Due and Arduino Uno must remain synchronized during these operations. Table 1 shows the length of time the microcontrollers take to execute different operations, and Figure 6 shows what operation each microcontroller will be performing at different times.

Table 1: Processing time required for different operations

Microcontroller	Function	Time to execute (µs)
Uno	Measure a 2-byte sample from the microphone	113
Uno	Send a 32-byte packet to the transmitter	150
Uno	Send a 32-byte packet to the transmitter and wait for confirmation from the receiver	650
Due	Write a 2-byte sample to the headphones	4
Due	Read a 32-byte packet from the receiver	171

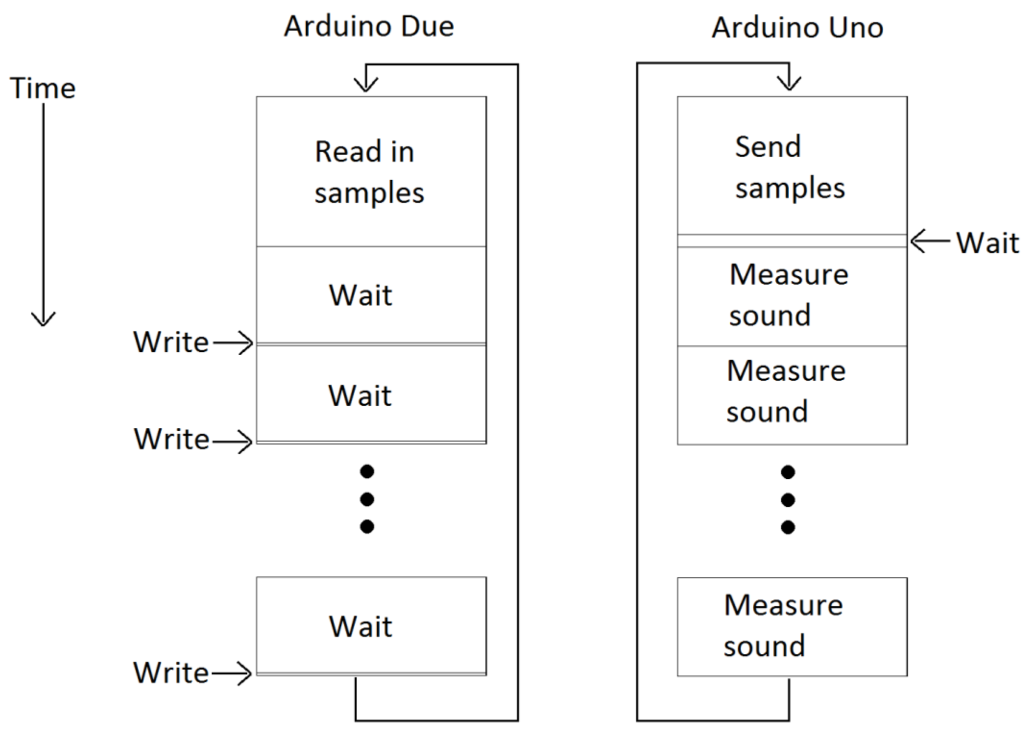


Figure 6: Synchronized processing on the Arduino Due and Arduino Uno

The large amount of time the Arduino Due spends waiting will allow it to perform calculations required by the proximity filter and alarm generation.

3.2.5 Proximity Filter

The proximity filter used in a full production system shown in Figure 1 has three inputs: the incoming speech signal, the position the incoming speech signal was broadcast from, and the position of the receiving unit. The proximity filter will work by applying the Equation 1 when the units are more than 5m away from each other. Equation 1 is based on [R25-I], which states that worker speech should attenuate in the same way as if there was no industrial noise. This attenuation is -6dB every time the distance between the units doubles, starting at 5m [6].

$$V_f = V_i \cdot 10^{-\frac{3}{2}\frac{d}{5}} \tag{1}$$

In Equation 1, V_f is the voltage level fed to the headphones, V_i is the voltage level read by the receiver, and d is the distance between the two units.

The distance between the two devices can be found by the difference between the two positions given by the position sensor. The calculation of distance is done by Equation 2, which approximates the distance between two points on the surface of the sphere when the two points are close together [7].

$$d = R \cdot \sqrt{(\Delta\theta)^2 + (\Delta\phi \cdot \sin(\theta))^2} \tag{2}$$

In Equation 2, d is the distance between the two units, R is the radius of the Earth in metres, Δθ is the difference in latitude in radians and Δϕ is the difference in longitude in radians.

The value of θ used in the sin(θ) term is measured with 0° at the north pole. While the sine function is a very slow operation, the controller will only perform the operation when the unit is powered on. The value of sin(θ) will not appreciably change across the worksite, so a constant value can be used.

Equation 2 is too slow to use in real time applications as written, as R is a very large number to read from memory when expressed in metres, and Δθ and Δϕ will be very small numbers when expressed in radians. As the GPS protocol gives latitude and longitude in terms of degrees, arcminutes, and arcseconds, it is natural to convert Equation 2 to these units, shown in Equation 3.

$$d = \frac{R \cdot \pi}{180 \cdot 60 \cdot 60 \cdot 10 \cdot 10} \cdot \sqrt{(\Delta\theta)^2 + (\Delta\phi \cdot \sin(\theta))^2} = 0.309 \cdot \sqrt{(\Delta\theta)^2 + (\Delta\phi \cdot \sin(\theta))^2} \tag{3}$$

In Equation 3, d is expressed in metres, and Δθ and Δϕ in terms of hundredths of an arcsecond. The choice of hundredths of an arcsecond was made because reasonable values will fit easily into a 2-byte variable. The maximum 2 byte number is 2^{16}, or 65536. 65536 hundredths of an arcsecond change in latitude corresponds to 20km regardless of initial position. The distance due to a change in longitude depends on the latitude from which the measurement is chosen. At an extreme latitude of 80° north or south, 65536 hundredths of an arcsecond corresponds to a distance of over 2km, well within the limits of the transmitter system.

The proximity filter will not be included in the prototype design due to time constraints. For the prototype demonstration, a simpler proximity filter will be applied. Two transmitters will be connected to the Arduino Uno, one with 100m range and one with 1km range. The Arduino Uno will send packets through the short range transmitter, and if it is out of range it will send packets through the long range transmitter, but only if the speech is loud enough. This design will create a two-level proximity filter, which will demonstrate the concept behind the proximity filter which will be included in the full system. The logic for this switch is shown in Figure 7.

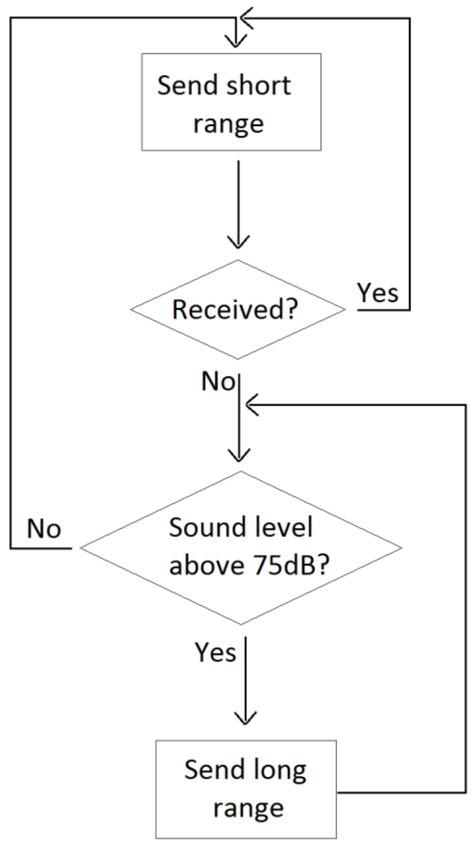


Figure 7: Flow chart for prototype proximity filter

The system will move through Figure 7 every time the system checks for connectivity, which is every 100 packets, or 2ms.

3.2.6 Writing to Headphones

Once the information is received by the Due it is processed and sent to the headphones. If the speech received from the Uno is not within the proximity range the information is not sent to the headphone. When the Uno is within range the speech is then passed directly to the headphones as long an alarm is not detected. When an alarm is detected the associated alarm signal is overlaid onto the speech signal, and the volume of both is cut in half so that both are audible but do not present any dangers to workers hearing. Once the alarm shuts off the speech signal is then passed though to the headphones at the normal amplitude.

3.3 Hardware Design

3.3.1 Analog Circuit Design

Analog circuitry is required to interface the headphones, microphones, and microcontrollers together. Analog circuitry is also used as a hardware frequency filter for alarm detection. This section details and justifies the design of the various stages of the input and output circuitry used for the selective noise cancelling headphones component of this project.

3.3.2 Input Circuitry

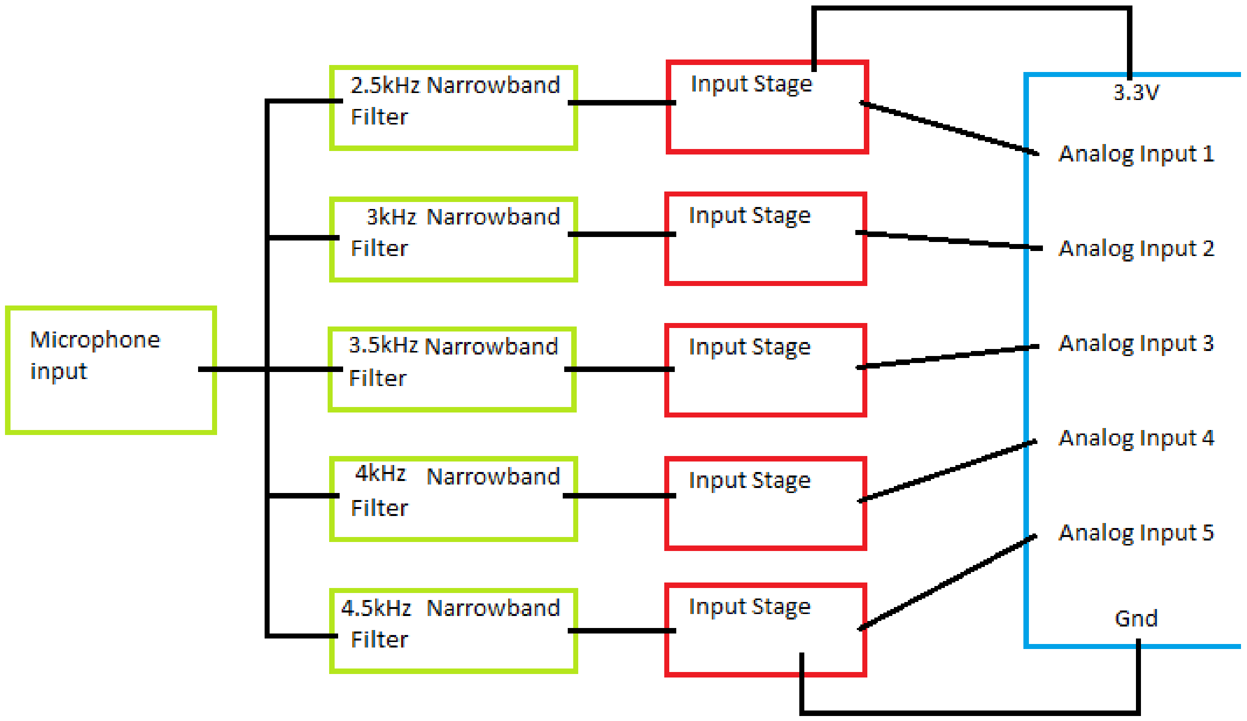


Figure 8: Block diagram for the input circuitry into Arduino Zero

In order to allow for selective noise cancelling, an array of input circuits is. Figure 8 shows a block diagram for the array of input circuits leading to the analog inputs of the Arduino Zero. Three important stages are considered: the microphone input stage, an active filter stage, and an input stage. The reasoning behind the array of input stages is to cover each frequency that an alarm could reasonably sound. For our prototype design, the input array will only be designed for five different common worksite alarms as shown in Figure 8. A full design could easily be manufactured with as large an array as desired, considering that the circuits act in parallel and will not slow down the processing power of the microcontroller. Additionally, while the array may seem bulky due to the large number of blocks, the full manufactured hardware could all easily be fit onto a single microchip. Each stage and their design are detailed in the following sections.

3.3.3 Microphone Input Circuit

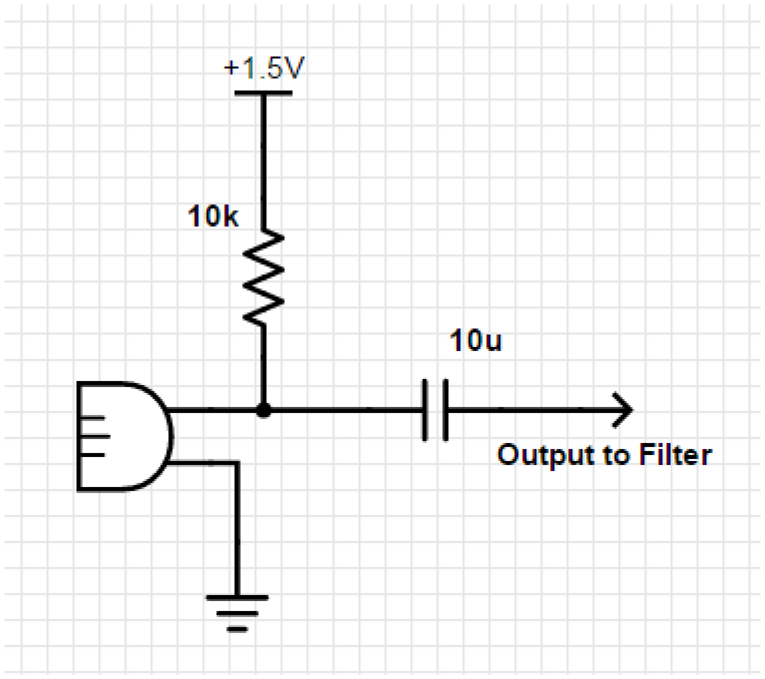


Figure 9: Microphone input circuit

Figure 9 shows the microphone input circuit. This simple circuit allows for our product to turn ambient sounds into voltage pulses for processing. The 10uF capacitor only allows AC signals to pass through into the rest of the circuit. This circuit is outputted into array of narrowband filters and does no further signal processing. The values of the resistor and capacitor are taken from the datasheet for a CEM-C9745JAD462P2.54R electret microphone [8].

3.3.4 Narrowband Filter

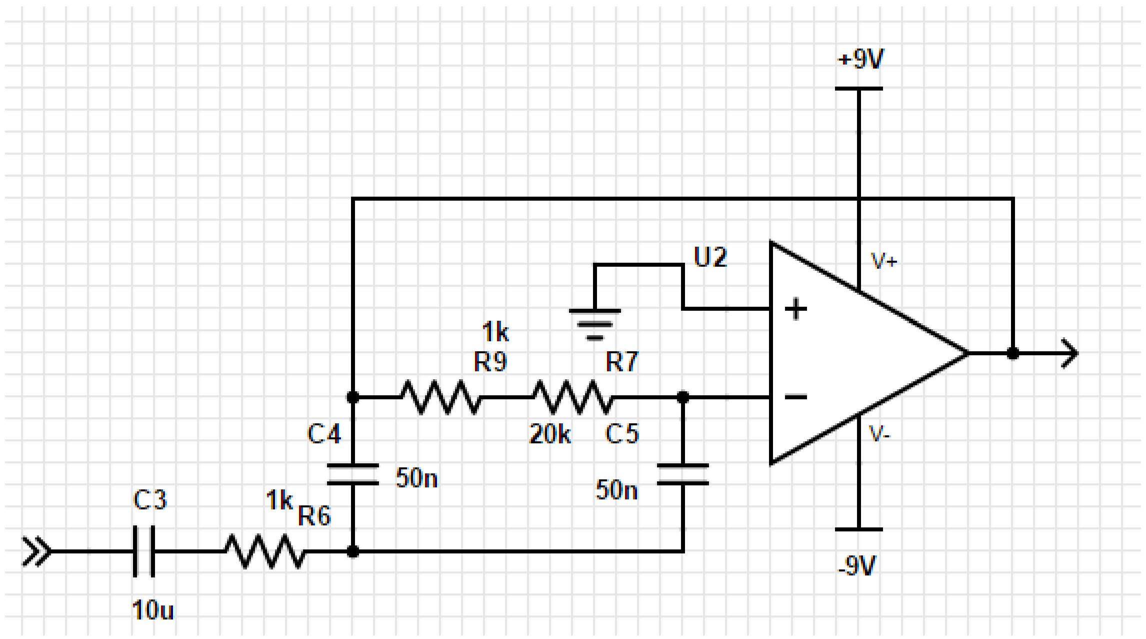


Figure 10: Active narrowband filter

The next stage is that narrowband filter stage. An array of these circuits is implemented into the full design, but only the filter designed for a center frequency of 3kHz is shown in Figure 10. This circuit amplifies a single frequency and attenuates all other frequencies. The centre frequency is the alarm frequency. The frequency response of the filter shown in Figure 10 is shown in Figure 11.

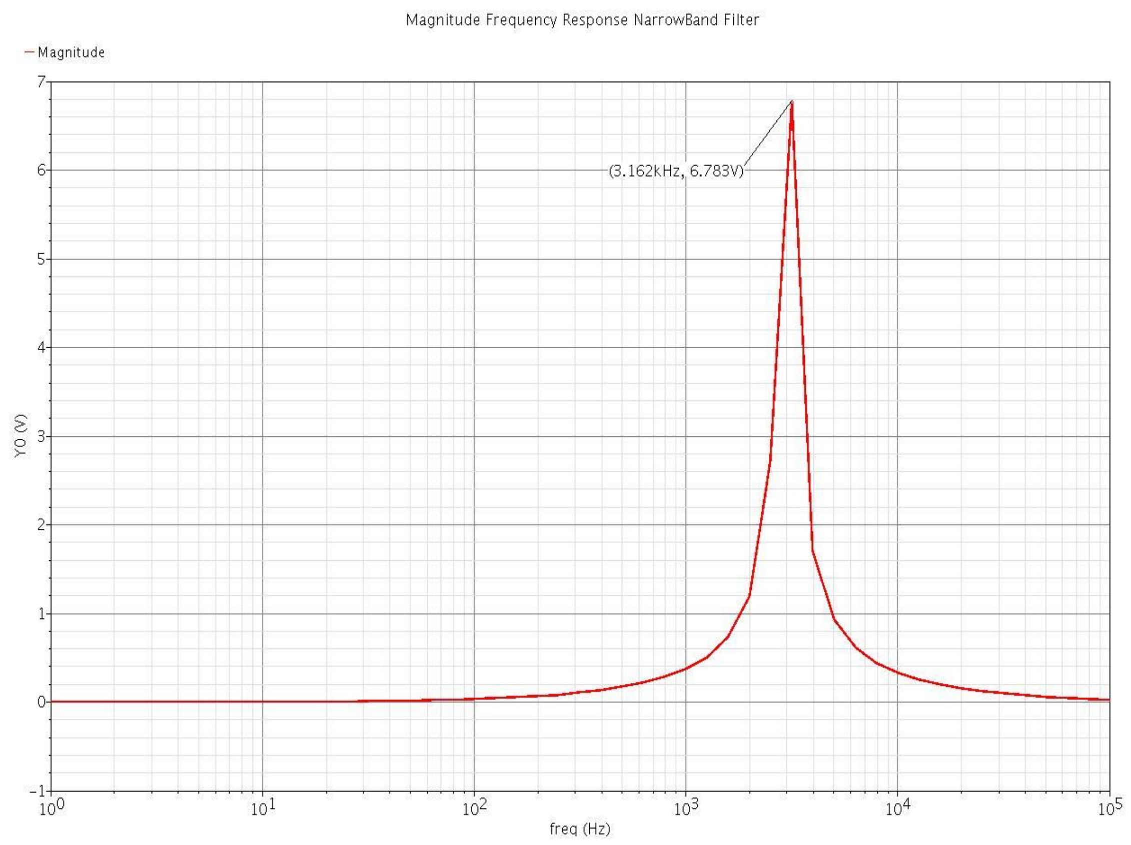


Figure 11: Frequency response of a 3kHz narrowband filter

The frequencies close to 3kHz will be significantly amplified above other frequencies. Most alarm frequencies are between 3kHz and 5kHz, and this means that an array of narrowband filters between these frequencies will be adequate. A narrowband filter is used over a wideband due to the shape of the alarm frequency spectrum, seen in Figure 12.

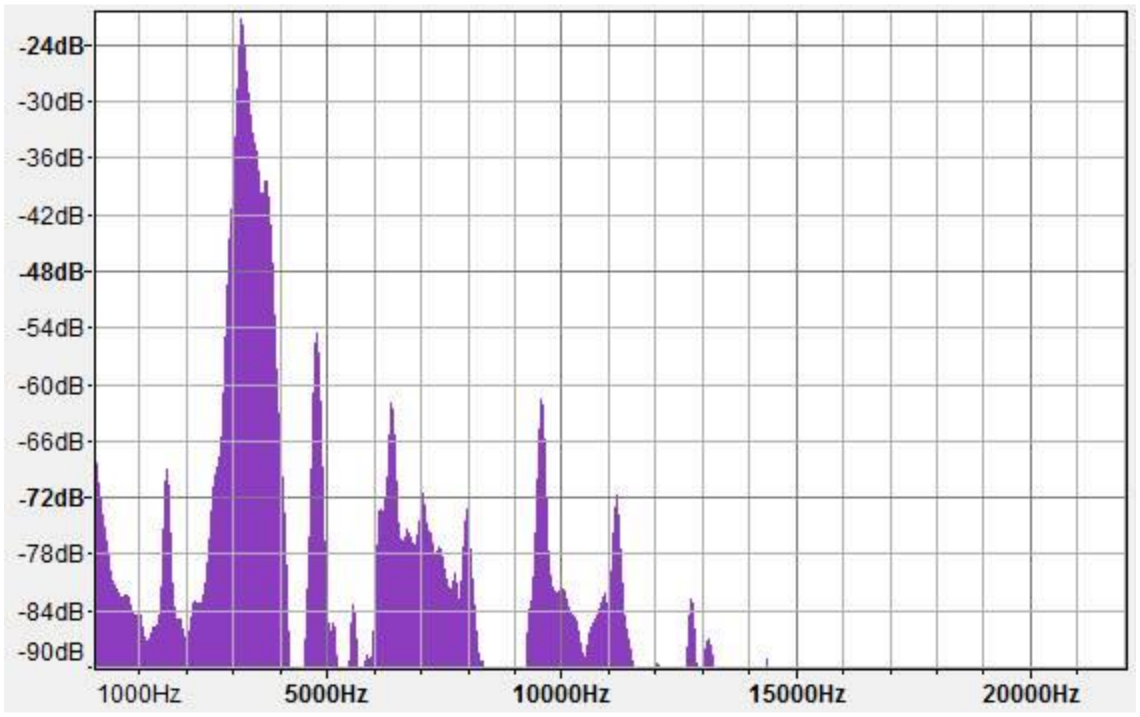


Figure 12: Spectrum of a 3kHz alarm with ambient noise

Figure 12 depicts how an alarm will be picked out through use of a narrowband filter. By applying the circuit to an alarm similar to the one shown above, the only noise that should pass through will be the alarm.

3.3.5 Input Stage Circuit

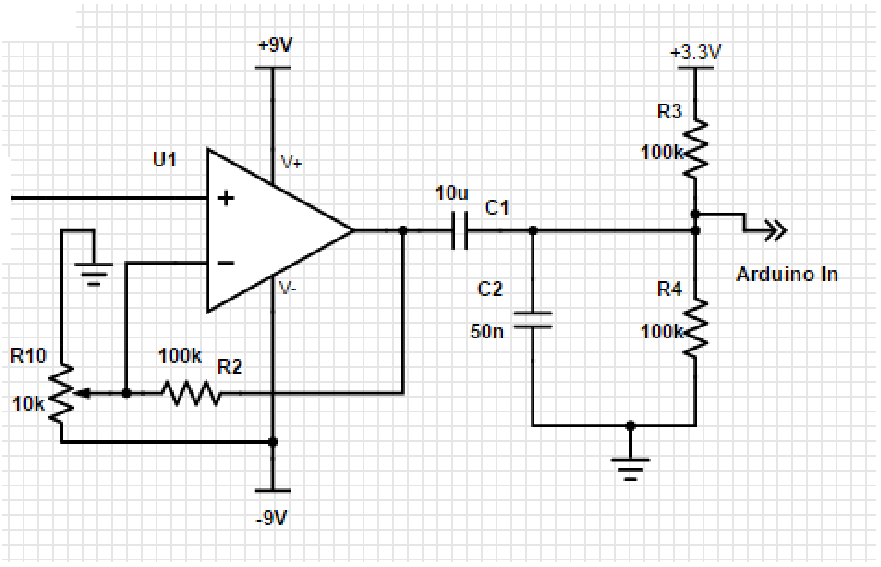


Figure 13: Input stage circuit

The next stage in the full input circuit is the input stage, seen in Figure 13. This circuit is necessary in order to interface the Arduino module with the microphone. An Arduino module reads in a signal centred at 2.5V with 2.5V amplitude, but the microphone output is centred at 0V. The input stage circuit is comprised of three parts: an amplifier, a capacitance stage, and a voltage divider. The amplifier allows tuning of the input voltage levels to allow for the maximum swing in the Arduino. The capacitance stage removes any DC voltage and only allows AC signals through. The voltage divider centers the AC signals at 2.5V.

3.3.6 Full Input Circuit

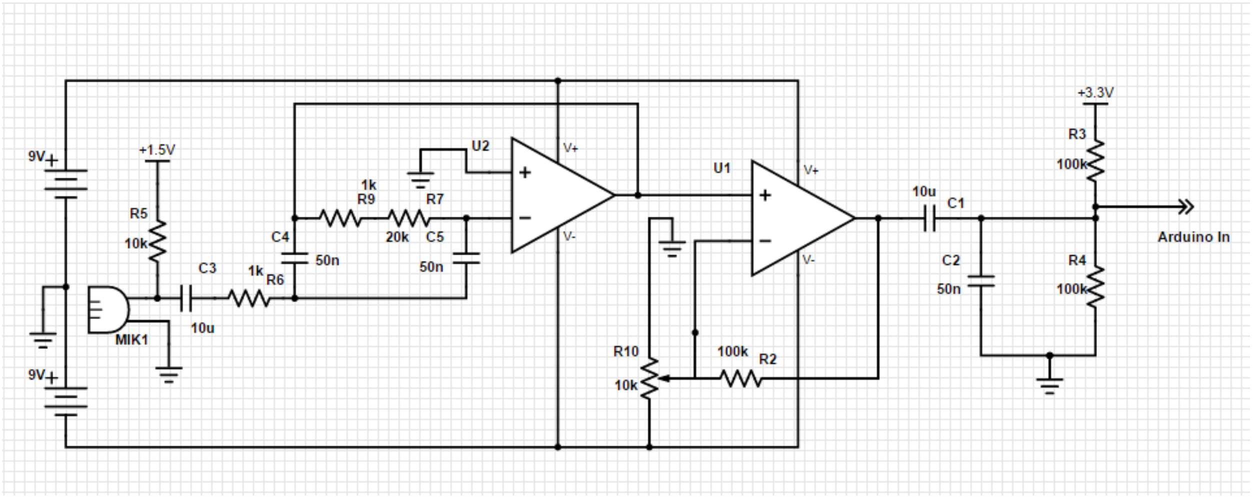


Figure 14: Full input circuit for one frequency

Figure 14 shows the full input circuit made by cascading the different stages together. An array of these circuits will be implemented in order to allow for multiple alarms to be selectively passed through to the user of the AlarmSense System. This full circuit allows the filtering of alarm out from excess ambient noise, and the amplification and offsetting for input to an Arduino module.

3.3.7 Output Circuit

In addition to the extensive input circuitry required for this project, a small output circuit is also required and is shown Figure 15.

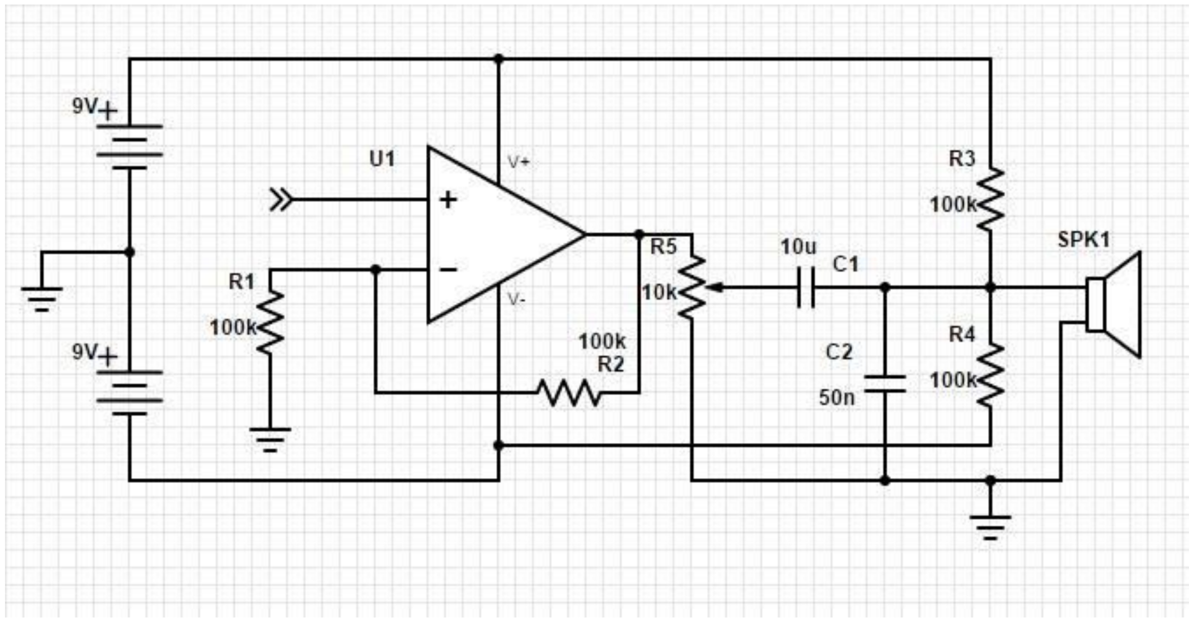


Figure 15: Output circuit

This circuit is similar to the input stage circuit shown in Figure 13, but in reverse. Rather than changing the signal for processing by an Arduino, this circuit takes an already processed signal and then modifies it so that the signal can be outputted into the headphones. This circuit is comprised of an amplifier, a capacitance stage, and another voltage divider, all of which serve the same purpose as those in the input circuit, but it also includes a potentiometer used as a “volume control”. By tuning the value of the potentiometer, the user can either increase or decrease the volume of any of the signals being output by the Arduino. This volume control allows us to easily tune the volume so as to adhere to requirement [R1-II] for the case that there is electrical power. Additionally, this circuit acts as a buffer so as to not wreck the sensitive hardware of the noise cancelling headphones.

3.3.8 Microcontroller Selection

This project utilizes multiple microcontrollers. The signal processing, wireless transmission, and audio output are split between two microcontrollers because of the limited amount of processing power on Arduino modules. The Arduino modules have many multiplexers and countless input and output pins. All of these allow for various peripherals and many tasks that can be completed, but also slow down any processing, especially inputting and outputting signals. This can make writing a value take one hundred cycles instead of four. With a final product a microcontroller could be designed specifically for the tasks needed, bringing it down to one microcontroller per user.

The Arduino Zero was chosen to do the frequency detection because of the existing libraries available. With a clock speed of 48MHz it was deemed fast enough to process the noise but not to receive a wireless signal and output an audio signal. Another issue with the Zero is the lack of a true analog out pin. Without this pin, a digital to analog converter would be needed, which would slow down the output processing.

The Arduino Due was chosen as the main microcontroller because of its 84MHz clock speed. This higher clock speed allows for more processing to be accomplished and allows for 40kHz sampling of the input and output signals. Another advantage of the Due is the two true analog output pins. Having these output pins removes the need for an external digital to analog conversion.

The Arduino Uno was chosen for transmitting the audio signal to the Arduino Due because of its low cost. For the prototype system, it is not necessary that the transmitting module be able to do any audio processing. The module needs only to read in and transmit signals, which the Uno is capable of doing at a low cost.

At a higher level, the Arduino platform was chosen instead of a Raspberry Pi or dedicated audio microprocessor for various reasons. Familiarity with C++ and the Arduino interface made programming and debugging more simple. The dedicated audio microprocessor was either more expensive or did not have all of the required features. Learning the Raspberry Pi architecture was daunting, considering the minimal potential gains from using that platform instead of an Arduino. Unfortunately, the Arduino modules were not computationally powerful enough, and so two were required to complete all of the necessary functionality.

3.3.9 Transmitter/Receiver Selection

The transmitter and receiver radio selected is the NRF24L01+ [9]. This transmitter has an open air range of 1 km, and a 2 MB/s data transfer rate. Any two NRF24L01+ radios can be configured to receive and transmit to each other. An RF24 radio can receive and differentiate between six other NRF24L01+ radios.

NRF24L01+ modules are light, weighing under 25g. This weight is low enough to satisfy [R13-II], which states that the units cannot weigh above 500g.

NRF24L01+ radios operate on the Serial Peripheral Interface (SPI) pins of an Arduino module, which are present on the Arduino Uno and Arduino Due. The details of the software used to run the radio are given in Section 3.2.3.

3.3.10 Location Sensor Selection

The position sensor used will be the EM-506 GPS receiver. This module was chosen due to its high accuracy, low weight, low cost, and fast start-up time [10]. The relevant specifications are shown in Table 2.

Table 2: Specifications for EM-506 GPS receiver

Specification	Value	Relevant product requirement
Accuracy	±2.5m	[R37-I] – the accuracy should be ±1m
Weight	30g	[R13-II] – the unit should weigh less than 500g
Cost	$39.95	[R51-III] – the unit must cost under $150
Start-up time	25s	
Voltage required	5V	

While the start-up time and required voltage are not explicitly listed in the functional requirements, the unit should not have an excessively long start-up time and 5V is the same voltage level that the Arduino modules require.

While [R37-I] states that the accuracy of the sensor should be ±1m, the sensor being used has an accuracy of ±2.5m. GPS sensors with higher accuracy are more expensive, heavier, and require a higher input voltage. The GPS sensor picked is a trade-off between these factors.

4. Briefcase Design

The charging briefcase will be not be implemented for the prototype system. The design presented here is for the full version of the system, to be developed after the prototype. A concept of the briefcase is shown in Figure 16.

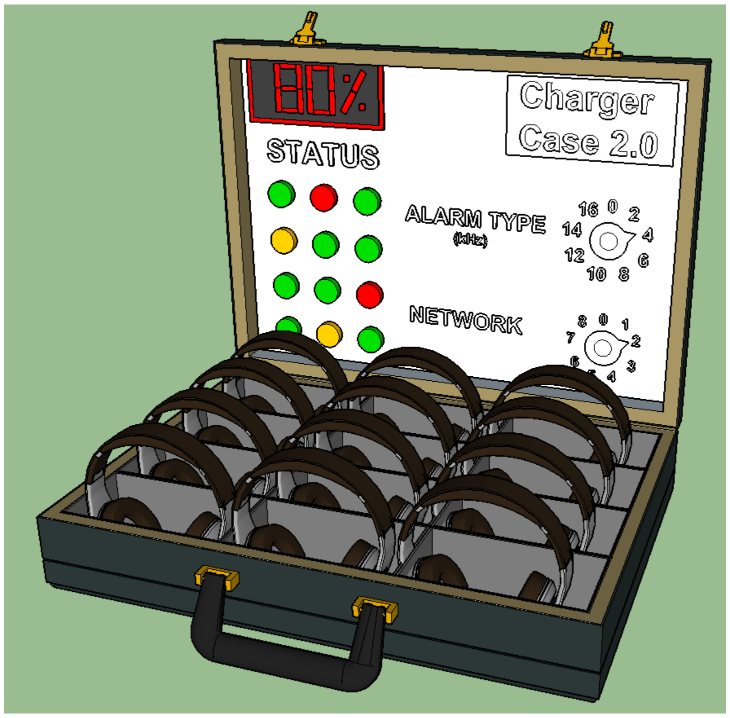


Figure 16: Concept for charging briefcase

The briefcase must be able to charge multiple units, and set alarm frequencies for multiple networks of units as per [R38-II], [R29-II], and [R40-II].

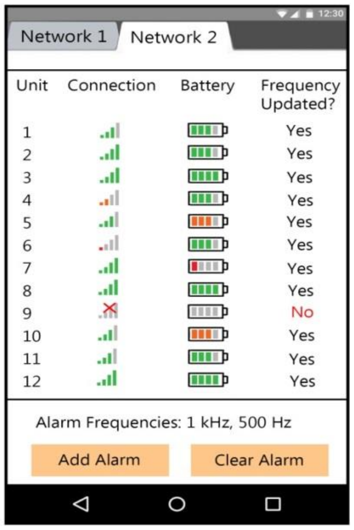
The charging briefcase will be plugged into a wall socket, which is a 120V AC signal. A step down transformer and AC to DC converter will be used to change this signal to a 9V DC signal, which is the voltage level of the batteries used in the unit.

A microcontroller will be present in the briefcase to track the frequencies of the different networks. The microcontroller will track the frequency associated with the different networks and upload them to the units when they are connected to the briefcase. As seen in Figure 16, the frequency and network values will be controlled by knobs on the inside of the briefcase.

5. App Design

5.1 Design and Features

The mobile application will be developed using the Android Studio Development Suite. The purpose of the application is to allow the foreman to wirelessly connect to the network of units. Once connected, the foreman will be able to check the battery and connection status of all devices on the network, as well as verify that each unit has the correct alarm frequency. The application will also allow the foreman to select alarm frequencies and send these frequencies to the units on the network. As with the briefcase detailed in Section 4, multiple networks of units will be supported to allow one foreman to manage multiple worksites. The home screen of the app is shown in Figure 17.



Network 1 | Network 2

Unit	Connection	Battery	Frequency Updated?
1	🟩	🟩	Yes
2	🟩	🟩	Yes
3	🟩	🟩	Yes
4	🟧	🟩	Yes
5	🟩	🟧	Yes
6	🟥	🟩	Yes
7	🟩	🟥	Yes
8	🟩	🟩	Yes
9	✗	⬜	No
10	🟩	🟧	Yes
11	🟩	🟩	Yes
12	🟩	🟩	Yes

Alarm Frequencies: 1 kHz, 500 Hz

Figure 17: Application home screen

For the prototype design to be demonstrated in April, not all features of the application will be developed. The application will only be for Android systems, not iOS as required by [R50-III]. The battery life and connectivity meters will not be displayed in real time. The prototype application will be able to manually set frequencies, detect frequencies, and send these frequencies to the prototype unit.

5.2 Manual User Input

The application connects to the unit with Bluetooth 2.0 serial Rx protocol with the Arduino Zero. Once a foreman submits a new alarm frequency, the application will send a packet to the Arduino Zero serial and interrupt the current process. The frequency in the code will be updated to the new value and the Zero will resume its normal cycle. As the number of instructions required to update the frequency is low, the interrupt will not appreciable slow down the Arduino Zero processing.

5.3 Automatic Detection of Alarm Frequency

In addition to manually updating the frequency, the application will support an ‘auto detect’ feature. If the foreman does not know the alarm frequency, they can hold the application next to the alarm when it is sounding. The application will analyze the spectrum of incoming noise and detect the frequency. This frequency will then send the value to the Arduino Zero in the identical fashion described in Section 5.2. In order for the detection to be as accurate as possible, the user should run the detection with minimal background noise. A spectrum of noise when an alarm is sounding is shown in Figure 18.

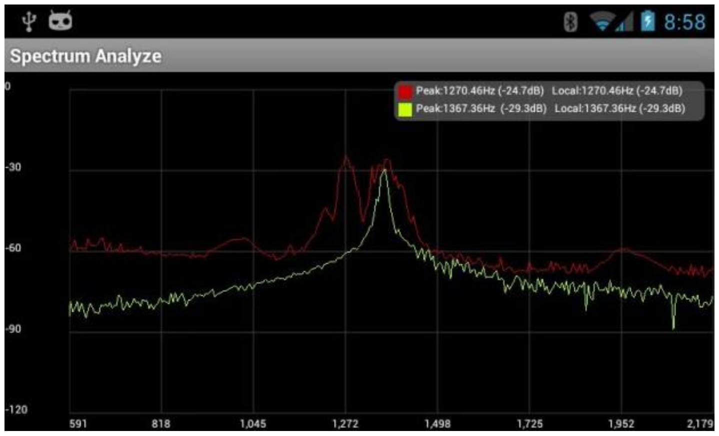
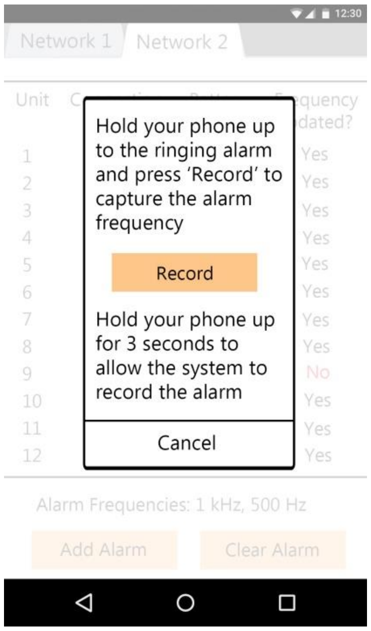


Figure 18: The frequency spectrum of an alarm, in green, and an alarm with background noise, in red

The screen which is displayed when the foreman is detecting an alarm is shown in Figure 19.



Hold your phone up to the ringing alarm and press 'Record' to capture the alarm frequency

Record

Hold your phone up for 3 seconds to allow the system to record the alarm

Cancel

Figure 19: The screen allowing a foreman to detect the alarm frequency

6. Conclusion

The document describes the design solutions to accommodate the functional requirements for the AlarmSense System. Engineers in Ekho Systems will use this document as a guideline for developing both the prototype system and the complete system.

Individual unit design is broken down into alarm detection and speech transmission subsystems each with hardware and software designs. The software design includes both the individual design of the subsystem, as well as how the two subsystems will interface with each other. The hardware design includes extensive analog circuitry for alarm detection and interfacing between the microcontrollers and microphones and headphones, as well as the selection of the hardware peripherals for location sensing and data transmission.

The charging briefcase design lays out how a single device will allow a network of units to simultaneously charge, as well as allowing a foreman to control the network. The charging briefcase is necessary for worksites where the units are owned by the company rather than individual workers, as there may not be the required number of outlets to charge the whole network individual.

The mobile application design details how the application will allow a foreman to quickly check the status of the network, which includes the remaining battery life on the units. The mobile application will allow a foreman to control the network without specialized training.

The attached Test Plan provides the framework for ensuring that the functional requirements are met. When the prototype has been assembled, this plan will be used to ensure the system is functioning properly.

All components of the AlarmSense System are detailed in their respective sections. This document lays out clear goals and plans for the remainder of the AlarmSense System development in each of these sections.

References

[1] Ekho Systems, "Functional Specification for AlarmSense System", 2016.

[2] PeoplePeople.se, “Not Another Headphone”, [Online]. Available: http://www.peoplepeople.se/not-another-headphone/. [Accessed:12-Feb-2016].

[3] D. R. Lu, C. M. Xiao, S. J. Xu, “Starch-based completely biodegradable polymer materials” in eXPRESS Polymer Letters. College of Material Science and Engineering of Huaqiao University, Quanzhou. Vol.3, No.6 (2009) 366–375.

[4]"Optimized High Speed NRF24L01+ Driver Class Documenation: Optimized High Speed Driver for nRF24L01(+) 2.4GHz Wireless Transceiver", *Tmrh20.github.io*, 2016. [Online]. Available: http://tmrh20.github.io/RF24/. [Accessed: 07- Mar- 2016].

[5]C. Shannon, "Communication in the Presence of Noise", *Proceedings of the IRE*, vol. 37, no. 1, pp. 10-21, 1949.

[6]"Voice Level and Distance", *Engineeringtoolbox.com*, 2016. [Online]. Available: http://www.engineeringtoolbox.com/voice-level-d_938.html. [Accessed: 07- Mar- 2016].

[7]"Spherical Coordinates -- from Wolfram MathWorld", *Mathworld.wolfram.com*, 2016. [Online]. Available: http://mathworld.wolfram.com/SphericalCoordinates.html. [Accessed: 07- Mar- 2016].

[8] Challenge Electronics, “Schematic Drawing,” CEM-C9745JAD462P2.54R datasheet, Jan. 2010.

[9] Nordic Semiconductor, nRF24L01 Single Chip 2.4GHz Transceiver datasheet, Jul. 2007.

[10] GlobalSat, EM-506 datasheet, Nov. 2013.

Appendix: Test Plan

Test 1

Test Name	Manual alarm change through mobile app
Purpose	Check the standard operation of the mobile application
Steps	1. Connect the app to the unit via Bluetooth 2. Manually type in a new alarm frequency from the standard available frequencies 3. Play an alarm of that frequency in the background
Expected Result	User will be notified when an alarm of that frequency is ringing

Test 2

Test Name	Automatic alarm change through mobile app
Purpose	Check the alarm detection of the mobile application
Steps	1. Connect the app to the unit via Bluetooth 2. Set an alarm frequency using the alarm detection feature of the app 3. Play an alarm of that frequency in the background
Expected Result	User will be notified when an alarm of that frequency is ringing

Test 3

Test Name	Alarm detection
Purpose	Check the standard operation of the alarm detection system
Steps	1. Play alarm noise through external speakers 2. Put ambient noise microphone beside the alarm 3. Listen to audio output from Arduino Due through headphones
Expected Result	Hear alarm through headphones attached to the Arduino Due

Test 4

Test Name	Multiple alarm frequency detection
Purpose	Check that the system accepts multiple frequencies from the app
Steps	1. Connect the app to the unit via Bluetooth 2. Manually type in two alarm frequencies separated by a comma 3. Play an alarm of the first frequency 4. Play an alarm of the second frequency
Expected Result	User will be notified of the alarm with higher priority if both are at the same time. User will be notified of each alarm if they go off separately. Priority of alarm to be indicated by the frequency of the beeps passed through the headphones.

Test 5

Test Name	Alarm detection with ambient noise
Purpose	Check the operation of the alarm detection system in the environment it will be used in.
Steps	1. Play industrial soundtrack at reasonably loud levels through external speakers 2. Play alarm noise through external speakers 3. Listen to audio output from Arduino Due through headphones
Expected Result	Hear alarm through headphones attached to the Arduino Due

Test 6

Test Name	Wireless speech transmission
Purpose	Check that speech can be transmitted without pushing a button
Steps	1. Play industrial soundtrack at reasonably loud levels through external speakers 2. Speak into microphone attached to Arduino Uno 3. Listen to audio output from Arduino Due through headphones
Expected Result	Hear clean speech with minimal background noise through headphones attached to the Arduino Due

Test 7

Test Name	Wireless speech transmission and alarm detection
Purpose	Check that alarms can be detected and speech can be overlaid to allow for communication in emergency situations
Steps	1. Play alarm noise through external speakers 2. Speak into microphone attached to Arduino Uno 3. Listen to audio output from Arduino Due through headphones
Expected Result	Hear clean speech with minimal background noise, and clearly hear the alarm.

Test 8

Test Name	Wireless speech transmission with proximity filter
Purpose	Test volume-based proximity filter
Steps	1. Speak softly into microphone attached to Arduino Uno 2. Increase the distance between the two units to over 100m 3. Speak softly again 4. Speak loudly, over 75dB
Expected Result	Hear speech through the Arduino Due when the two units are close together, hear nothing when the units are far apart and speech is quiet, and hear the speech when the units are far apart and the speech is loud.