Shush! The Parametric Speaker Alarm Clock

Ryan J. Coleman, EE, Christos P. Lemonias, EE, Shaaz Z. Salim, EE, Timothy W. Reardon, CSE

***Abstract*—We introduce Shush!, a parametric speaker alarm clock that wakes up one person. Shush consists of a parametric array, microcontroller, and circuitry contained within a mountable box as well as an Android app for selecting alarm times and tones, among other things. Shush uses specialized transducers for a directive cone of sound. Only when situated within the cone will the original sound be heard, outside of the cone nothing is heard.**

# I. INTRODUCTION

One of the most common complaints that college students have during undergraduate educations takes place in early mornings. That is, dealing with other people’s alarm clocks. Chances are, a student and his/her roommate will not have synced schedules everyday of the week. One might have 8am classes and wake up at 7am, or just be an early riser. On the other hand, this persons roommate may have their first classes start at 11am and may consistently intend on waking up on the late side of the morning, like 10am. With this in mind, chances are that the alarm clock belonging to the person who wakes up earlier will most likely wake up the other unintentionally, especially with the tight quarters of dorm living. When this happens, it is also very likely that one of two things will occur, or both. The person that wakes up early feels very bad for the roommate they routinely awaken inadvertently, or the roommate getting awakened too early gets very annoyed and frustrated with this living situation and that pesky alarm.

Looking beyond our personal experiences of college dorm living, this issue reaches out to all who share a sleeping space with another. This amounts to a very large scale problem that needs attention. Preventing this unsynchronized scheduling dilemma amongst roommates would be an almost impossibly scalable task. But, dealing with this complication at the source, the alarm, is a less challenging goal. Although the idea of an alarm clock has been a foundational tool that has not changed much since its inception, in recent years many attempts have been made that acknowledge this same very problem, all proposing a variety of solutions. Some of these have been to wear earphones in while sleeping so the alarm will enter directly in one’s ear or to even wear something like a wristband that will vibrate to awake you. Although these solutions work and will only wake up one person, they introduce another problem of not being optimally comfortable

Ryan J. Coleman from Billerica, Ma (email rcoleman@umass.edu)

Christos P. Lemonias from North Andover, Ma (email clemonias@umass.edu)

Shaaz Z. Salim from Canton, Ma (email szsalim@umass.edu)

Timothy W. Reardon from Southwick, Ma (email treardon@umass.edu)

while sleeping which could take away from your sleeping experience. Many people do not like wearing anything that resembles jewelry, earbuds, or earphones when falling asleep for obvious reasons. Thus, Shush! was born. Our solution for this problem does not require someone to wear anything uncomfortable, yet will still only wake up one person without inconveniencing any roommates involved.

Shush! uses parametric circuitry along with transducers to produce this alarm clock signal at ultrasonic frequencies that demodulate through the air. This technology can be thought of as a “laser beam” of sound that aims the sounds waves in a very directive manner. This directed beam of sound can be pointed toward the person sleeping so they will be the only ones in the room to hear it. A list of the system specifications are listed in Table I.

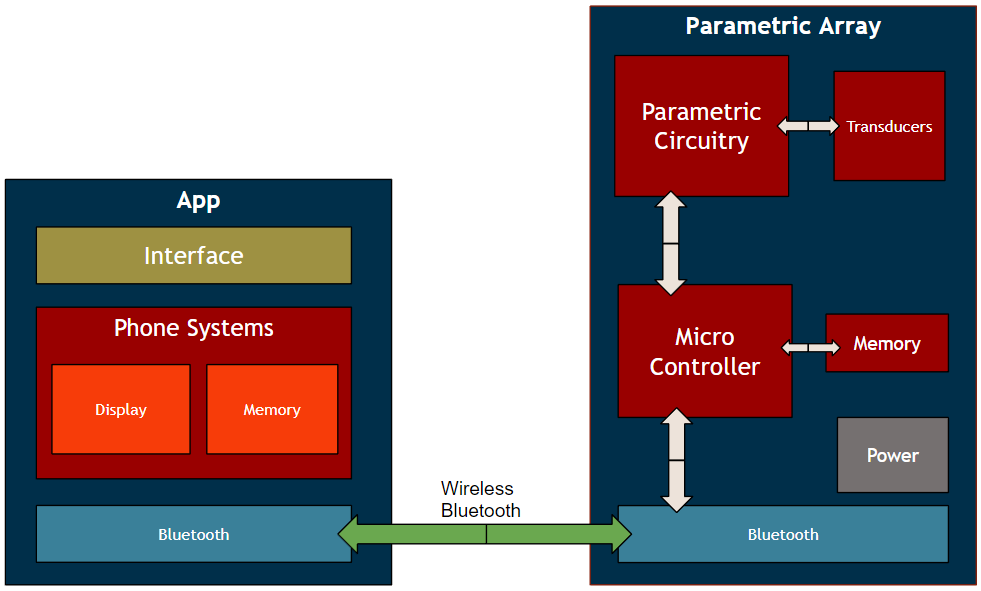
|  |  |
| --- | --- |
| Requirements | Specifications |
| Works at a reasonable distance from bed | <= 1 meter from bed |
| Loud enough to wake someone up | 70-90 dB |
| Audible anywhere on a pillow | 20x28in cone |
| Lightweight | Total weight under 1.5lbs |
| User friendly App | App is able to be used by anyone effectively after only just picking it up. |

*Table 1: List of system requirements and specifications*

# II. Design

## A. Overview

In order to solve this problem we will create an alarm clock that will only wake up one person at a time. To accomplish this we use parametric circuit (a circuit that brings the audio signal to proper specs for our use), consisting of a schmitt trigger, half bridge driver, power MOSFETs, and some voltage regulators, along with transducers to send this alarm signal at ultrasonic frequencies to wake up a single person. With this technology, the sound that travels in the air to the user will be directive, traveling in the shape of a cone to the pillow area on the bed. This will solve the problem because only one person, whoever this beam of sound is pointed at, will hear the alarm. This will leave the other roommate or roommates unable to hear it and no one will wake up with the exception of the intended user. We feel as though this method and implementation is more effective than other ways that we considered.[1][2] Mainly due to the fact that sleeping while wearing things such as earphones/earbuds or wristband to wake you up is uncomfortable and will not guarantee customer satisfaction.

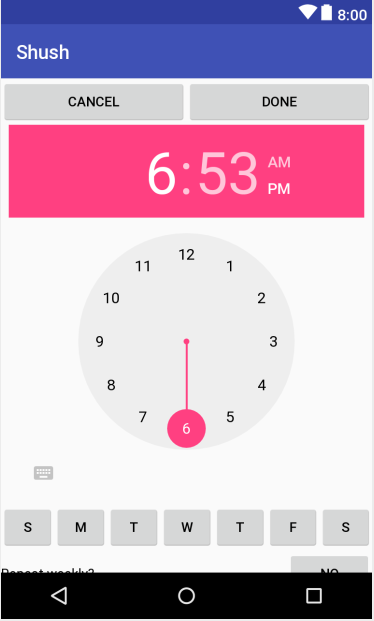


*Fig 1: The block diagram of Shush! This diagram depicts the two separate subsystems that make up Shush! The App and Parametric array subsystems communicate through bluetooth.*

In figure one you can visually see how we will be implementing this technology with user satisfaction and ease of use in mind. Looking at this block diagram as a reference, we will have two main modules for the design: the phone application and the parametric array. Looking at Table 1, each block’s system specifications are listed. As the control device, the phone application will allow for the user to easily choose when to wake up, snooze the alarm, and select the music/tone to play with a convenient user interface. The other block in the diagram, the microcontroller along with the other hardware contains most of the technology to ensure a directive sound will not only wake you up, but keep your roommates sleeping.

## B. Android App

The Android App is the main user interface. When opening the App it prompts the user to choose which bluetooth connection it will be using when sending alarm data. Using the App, the user is able to create new alarms. When creating a new alarm the user is able to choose what time it will go off, on what days it will go off, if it will repeat weekly, and the tone of the alarm. Once a user has at least one alarm set they will see them on the home screen. From this screen the user can click on an alarm and decide to edit it or to delete it so it will no longer play. Also on the home screen there is a button to connect to the bluetooth module in our parametric circuitry just in case of a disconnect after starting the app. The App updates the memory of our parametric circuitry every time an alarm is added, edited, or deleted. We did this to keep the amount of information in a single update very limited so that it would transfer quickly. Below is the top of the new alarm screen.



*Fig 2: New Alarm page is an example of one of the user interfaces. This page is where a user will be able to set all of the settings for their alarms.*

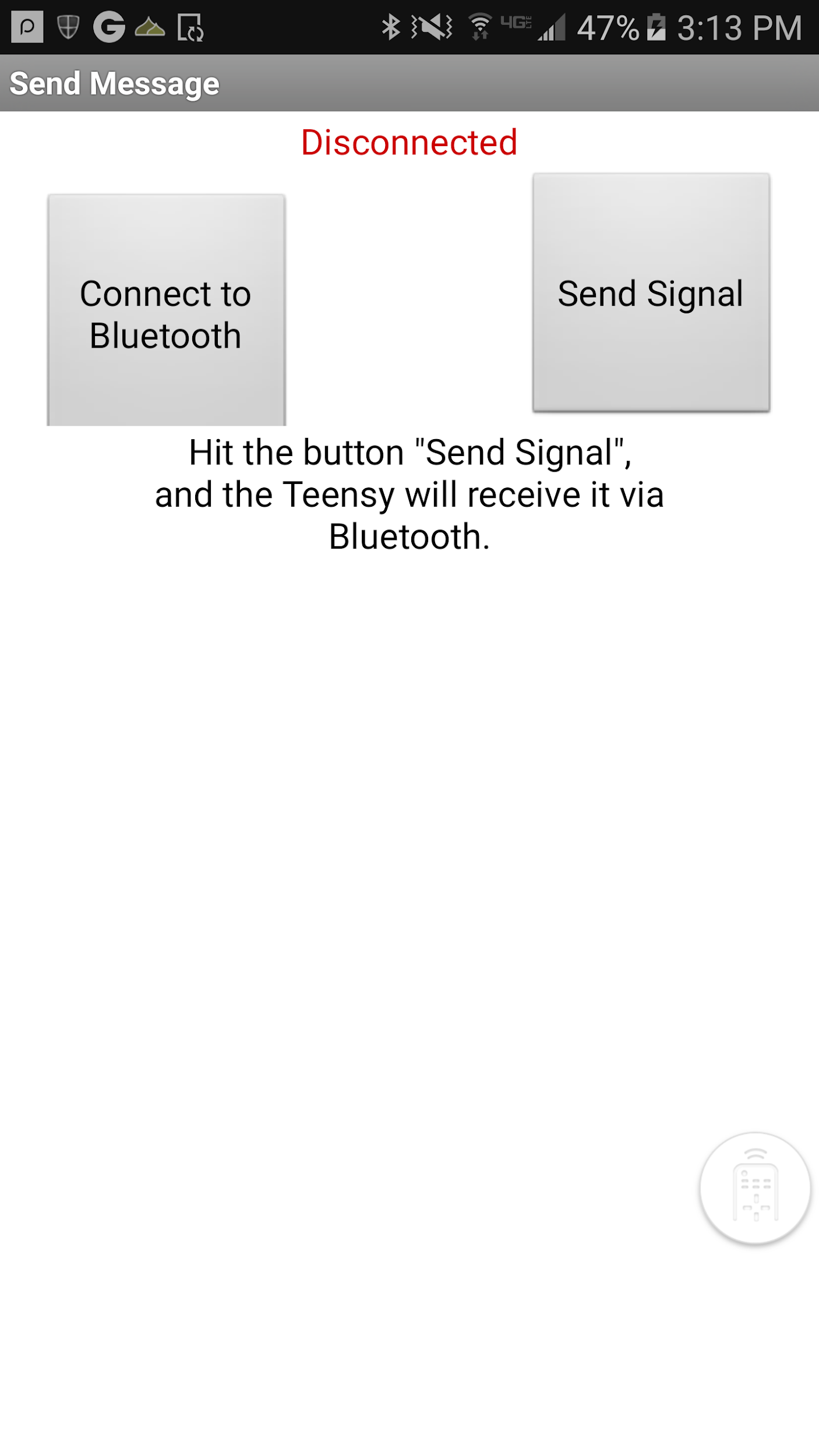
The Android App is being developed in the Integrated Development Environment (IDE),[3] Android Studio. All of the App’s functionality is programmed in Java and the App utilizes XML formatting to design the user interface layouts. Many of our previous courses have taught various portions of programming in Java. A few courses have taught techniques to integrate databases and data structures into the functionality of our codes.

In finishing the App we learned how to integrate different device functionalities such as the speakers, vibration, and memory. We choose not to integrate the device’s speakers or vibration into our final prototype but we do still use the memory to store the alarm tones so that the user can see what alarms they have set. We also learned how to package the alarm information and send it over a bluetooth connection to the bluetooth chip in our parametric circuit.

During development the App has been in an agile workspace where it quickly goes from development to test and back again. We do this by developing a piece of the App and then using an Android emulator to simulate a real device which then allows us to test what we just developed. At certain stages we have also downloaded our progress to a Galaxy S5 phone to test the App on a real device. Testing this way simply allows us to verify that the layout is correct and that buttons are clickable and do as they should when clicked. Part way through our testing we realized we could no longer use an emulator to test on. This is because the emulator we had did not have access to a bluetooth connection and therefore we would not be able to test our bluetooth communications.To test bluetooth we downloaded the App onto two different cell phones and began sending information to our parametric circuitry. Based on the parametric circuit we decided the best possible formatting for our communications. The test was considered passed when we could repeatedly send alarm information and each one showed up as it should in the parametric circuits memory. We also tested by sending alarms for a minute or two away on the day of testing to see if the alarm would play at the specified time. Then we would add alarms two or three minutes away and delete one of them. we did this to show that we can send an alarm and it will play but we could also test that the alarm delete messages were being sent and received successfully and to rule out a communication error that never set the alarm.

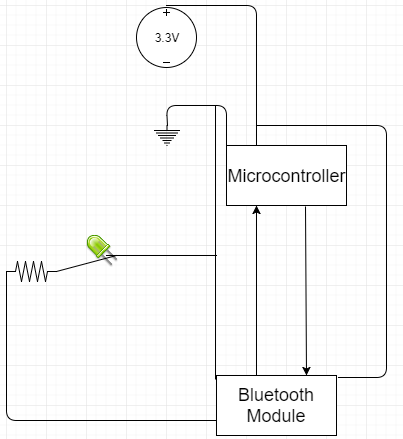
## C. Bluetooth

The Bluetooth integration in this project is especially important, as this forms the critical path between user inputs on their smartphone, and the Teensy microcontroller2 [4] which will process the inputs to control the rest of the system. To tackle this block, we started by developing an extremely lightweight mobile application separate from our main mobile application to test connectivity. This app, built using MIT’s App Inventor3 [5] software, has 2 main functions: finding and connecting to a Bluetooth device, and then sending a 1 or 0 to the device.



*Fig 3: Bluetooth App Home screen. The app has two easy to identify buttons, a colored indicator showing whether Bluetooth is connected, and an instructions.*

The Bluetooth device we chose is the HC-054, [6] which worked well for our project’s purposes. To test that the HC-05 is properly receiving the signal, and to test that the Teensy can process the signal and understand how to use it, we built the circuit below, which consists of the Teesny 3.5, the HC-05, an LED, and a resistor. When the app is connected to the HC-05, and ‘Send Signal’ is pressed, the HC-05 should receive a ‘1’ and transmit this signal to the Teensy. The Teensy is constantly checking for a change in signal from the HC-05, and when it receives the ‘1’ it will then jump into an ‘if’ conditional in its programming, and will know to digitally write a high signal to the LED.



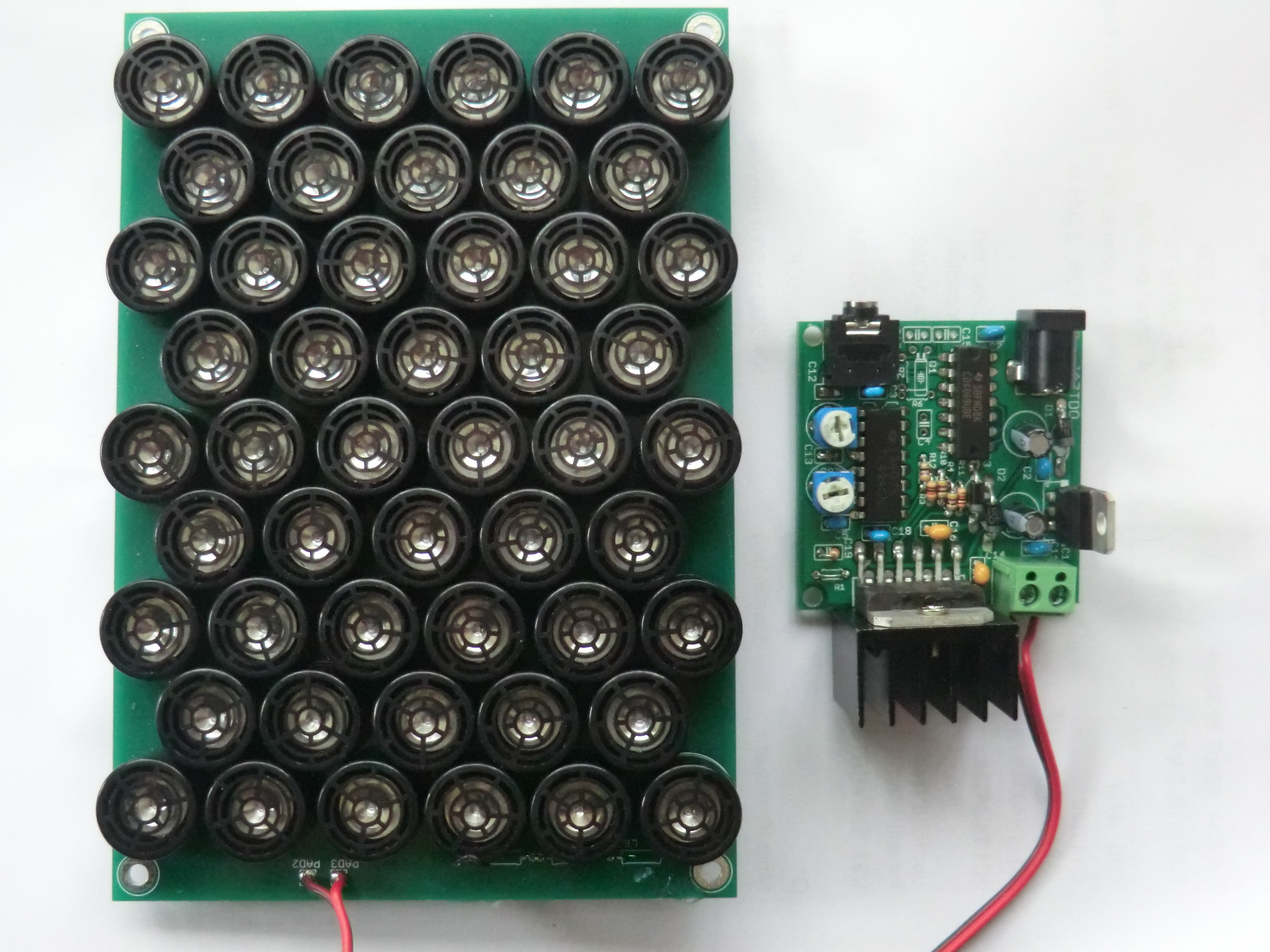
*Fig 4: Bluetooth Test Diagram. The Bluetooth module and microcontroller are connected to a supply voltage and ground, and are connected to communicate with one another.*

On observation, we were successful in connecting our microcontroller to our application with Bluetooth. The next step was to redesign the app with the Android Studio IDE to integrate it with our primary app. This way, we can stretch past the initial proof of concept and test the Teensy to manage receiving alarm tones and times. A simple strategy we used is to send a string which includes information such as the time and tone of the alarm, and then parse the string using delimiters such that it can differentiate between tone signals and time signals which are received on the same pin. This was especially difficult as the open BluetoothAdapter object did not carry across activities, and it was necessary to redesign the process by which we connect to Bluetooth such that the connection was opened for each alarm added.

We were successful in implementing Bluetooth to send all of the necessary information. To do this, it was important to understand how to use MIT’s App Inventor to build an app with Bluetooth connectivity, as well as how to do the same function with Java code. It was also important to understand to how use and code Arduino libraries and integrate our written program with the Teensy to guarantee compatibility across the two platforms. Our experience as Undergraduate Instructional Assistants for the lab portion of ENGIN 112 was particularly useful in ensuring this portion of the project was done in an efficient manner.

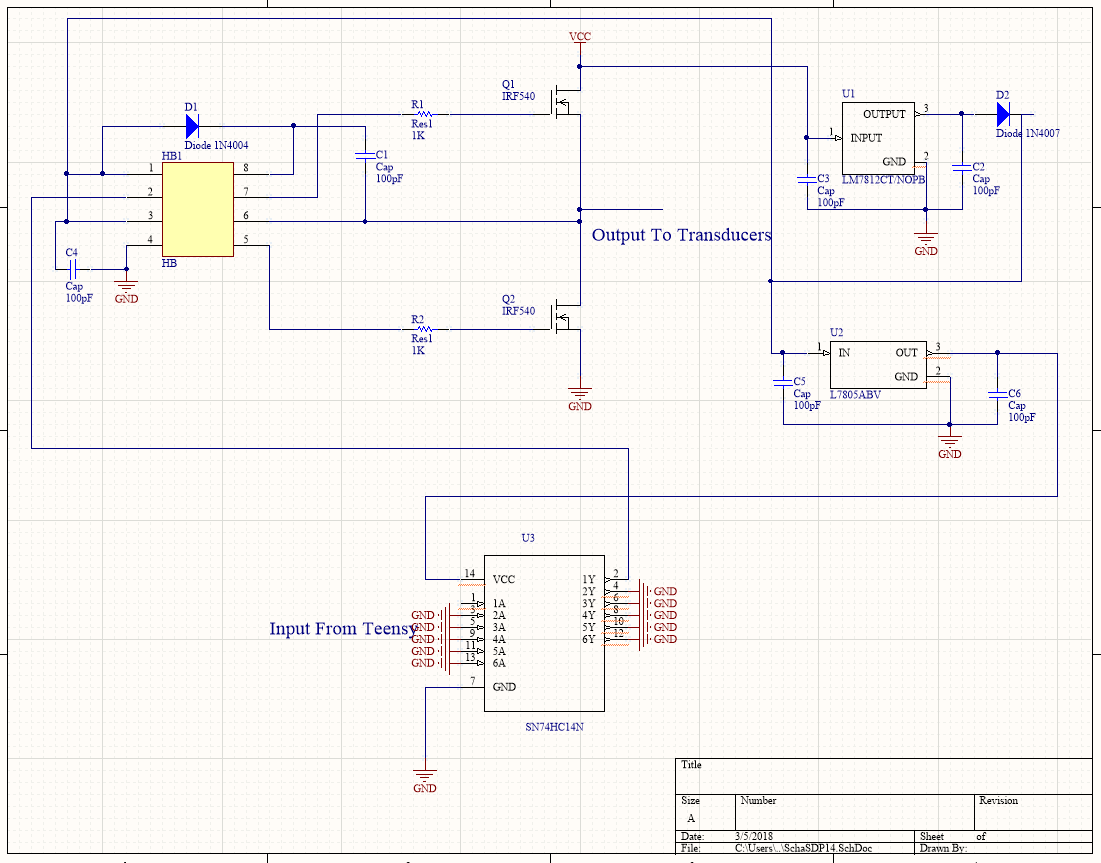
## D. Parametric Speaker

The parametric speaker is what makes this project special. Any normal speaker will be audible radially outward in all directions, but with a parametric speaker, you get a directive cone of sound heard by only people within the cone. The way parametric speakers works are by pulse width modulating the song or tone you want to hear onto a 40KHz carrier signal. This frequency is ultrasonic and is very directional. This signal is fed into a special speaker made from an array of transducers. When the 40K signal emits from these transducers it keeps its directive property, but as it passes through the air the air acts like a low-pass filter that demodulates the signal leaving us with the original signal in the cone and nothing outside of it. The parametric speaker is split into two parts consisting of the circuitry and the array. An example array is depicted below.



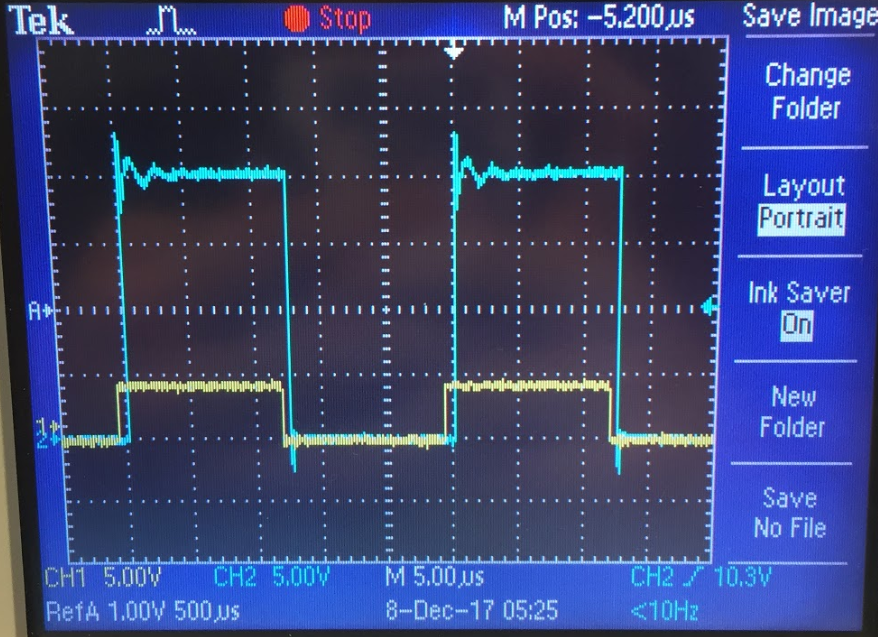
*Fig 5: Parametric Array. This parametric array uses 50 ultrasonic transducers in a honeycomb pattern. This patterns works best for achieving a directive cone of sound. The distance between the transducers effects how wide the cone is as well as how moise the beam is.*

The array is made from a collection of 16mm piezoelectric transducers. The transducers are wired in parallel in a honeycomb pattern. Our research into parametric arrays showed that this would be the optimal positioning because it minimizes the minimum spacing between transducers. [7] Next, we need to decide the number of transducers and how spaced they are from each other. Nonoptimal placing can cause signals to interact destructively and produce a worse signal. We also found that the more transducers you have the more directive our array is and the closer they are the more directive. This is not always a good thing because we specify a range for our cone at 1m and nonoptimal amounts or placing can move us away from our goal. Our MDR setup used a breadboard for placing our transducers and because of that the optimal spacing could not yet be achieved, but we have since designed, acquired and setup our PCB’s to gain better results. Even with a perfectly setup array we can’t just feed any modulated signal into our transducers. The transducers require 24V to be driven, but our microcontroller only produces 3.3V. This is where our parametric circuitry is needed to continue. Below is the parametric circuitry schematic.



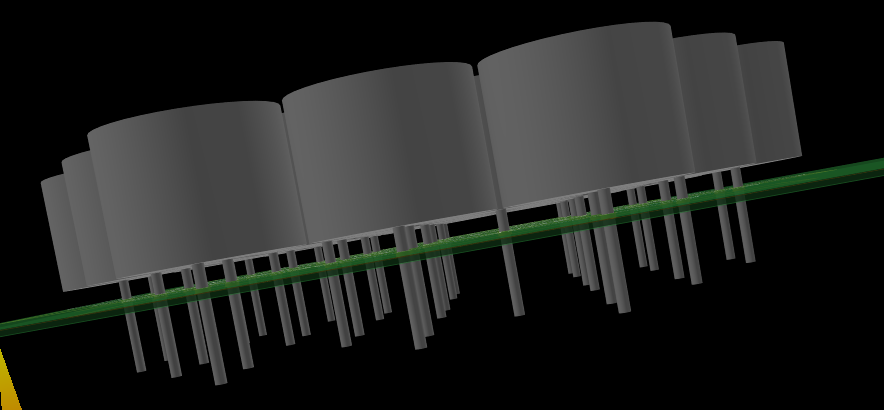
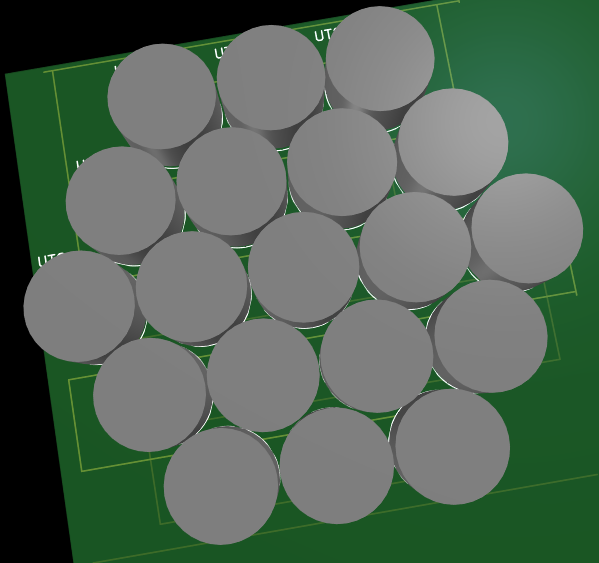
*Fig 6: The Current Parametric Circuitry for Shush! This designs consists of a Schmitt Trigger, a Half Bridge Driver, 2 n-channel mosfets and two voltage regulators as well as some resistors, capacitors, and diodes. The H-bridge uses the two n-channel mosfets that create a 24V replica of the 3.3V signal coming from the microcontroller.*

The parametric circuit needs to take the 3.3V 40KHz signal and output the same signal to the array at 24V and with enough current to drive the array. Our circuitry begins with a Schmitt Trigger, this helps square out the signal coming from the Teensy 3.5. We then use a half-bridge driver to up our voltage to 24V. The driver takes in the 3.3V and is connected to two n-channel mosfets. When our 3.3V signal is high, the top mosfet pulls the output up to the required 24V while turning the other mosfet off. When the input goes low, the bottom mosfet is turned on and our output is pulled down to 0V while the top mosfet is off. This setup provides us with in output identical to our input, but with a voltage of 24V opposed to 3.3V. The oscilloscope reading below shows our circuitry working.



*Fig 7: The output (blue) of the parametric speaker compared to the input (yellow). The input signal represents the 3.3V pulse width modulated signal that will come from our microcontroller, after passing through the parametric circuitry we receive the same signal amplified to 24V*

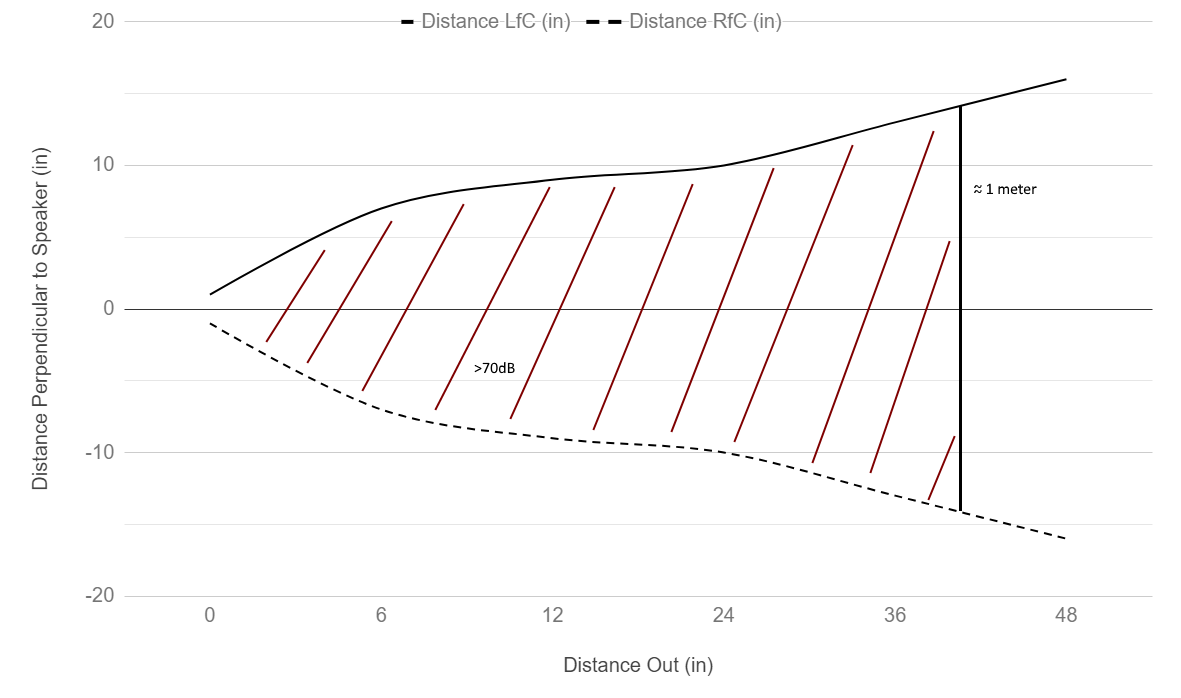
We were required to design and create a printed circuit board for this project and in the end we needed two. One PCB for the Transducer array and another for the Parametric Circuitry. For the designing of our PCB we used Altium PCB Design Software.[8] First we designed the Transducer array which consisted of 19 of our 16mm piezoelectric transducers. Altium comes with an extensive library of parts, but sadly our Transducers were not included. We instead designed the transducer models themselves before designing and routing the PCB. Below is a top and side 3D view of the transducer PCB.



*Fig 8: After designing and routing our PCB we get a 3D view of what the real life finished PCB will look like. We modelled the transducers ourselves within altium.*

After the transducer PCB comes the component PCB which the schematic for is found in Fig 6. This PCB was also designed in Altium and most of its components were either found in Altiums libraries our were available online for download. The Half Bridge Driver was the only component we were forced to design ourselves.

Once our speaker was working we needed to test it’s range. We used a decibel meter to measure sound volume at specific distances from the array and how far out on all sides you continue to hear the signal at those distances. Whenever we experienced a substantial drop in dB we marked that at the edge of the cone. Below is the graph of our results.

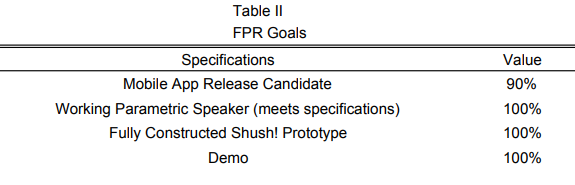


*Fig 9: Distance out from the beams center vs distance from the beams source. Starting close and incrementing out from the speaker we marked where considerable drops in dB were recorded. This provides us with a relatively accurate graph of our cone.*

We were able to achieve a range of 26in x 26in at 1m which is very close to our specifications. Over time we saw lots of improvement as we fine tuned our design and finally placed it on our PCB.

# III. Project Management

Table II shows the list of our FPR deliverables and their level of completion.

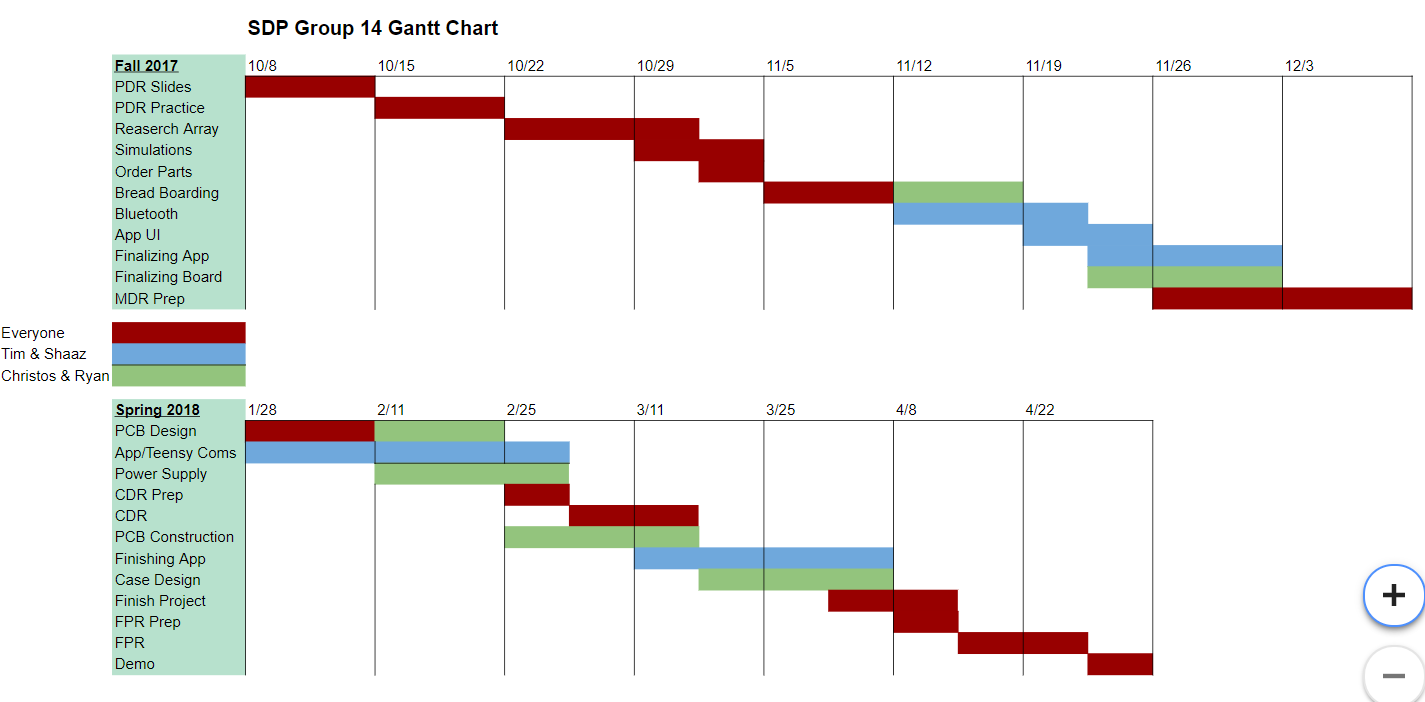


*Table 2: List of FPR deliverables and their completion at the time of FPR. The Mobile App Release Candidate was the only deliverable that wasn’t 100% complete.*

The goals that we set for ourselves at the time of CDR were mostly accomplished by our demonstration at FPR and Demoa Day. We showed off our Shush! android app and it’s ability to add and delete multiple alarms through the use of a bluetooth signal that is sent to the Teensy 3.5 Microcontroller. We also provided a working parametric array and circuitry that could take in a 40 KHz signal and output a very directive cone of sound in which the original audible sound can be heard. Finally we showed off the finished enclosure for our Shush! device and the final working prototype.

The Demo for our FPR deliverables consisted of a demonstration of the Parametric Speakers Specs and the working app. The only set back we experienced was a bug in our app that we were unable to fix. The parametric speaker met all specifications and performed as intended on both FPR and Demo Day.

So far we have worked very well together as a functioning group. Though we split our deliverables between us, no section of this project was completed without all of us having at least some kind of input. Our background in different areas of electrical and computer engineering allowed us to help each other whenever someone became stuck as well as effectively bounce ideas and solutions off each other whenever we all feel stumped. We were able to meet all together at least once a week as well as communicate and worked on the project most days when one or two group members were busy. We met with our advisor, Professor Eslami, every week and he provided us with lots of useful insight while also making sure we kept on track with our goals. For this first semester Timothy Reardon , a CSE, and Shaaz Salim, an EE, focused on the coding and software side of our project. They honed in on our bluetooth connectivity and app framework. While they were focusing on the software side of Shush, Our project manager Ryan Coleman and Christos Lemonias, two more EEs, picked up the reigns for the hardware portion of the project. They built the parametric circuitry consisting of a half bridge driver and optoisolators, the parametric array of ten 16mm piezoelectric transducers, and attempted to pulse width modulated signals from our teensy 3.5 microcontroller. Our Gantt chart can be found below in figure 3.



*Fig 10: SDP 14’s Gantt Chart. As a team we have followed the gantt chart thoroughly and stayed on track.*

# IV. Conclusion

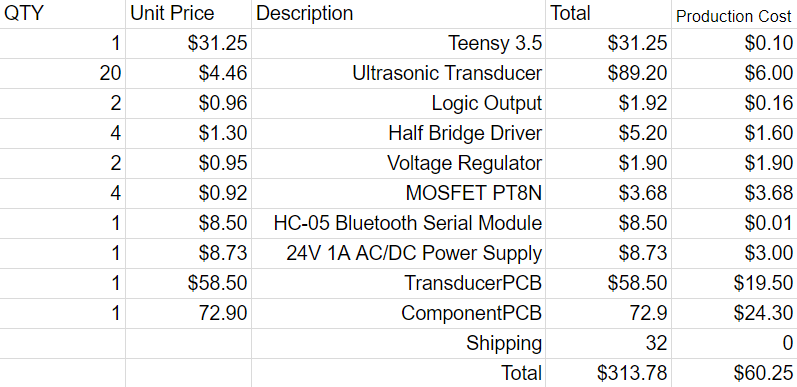
As of the conclusion of our senior design project, we have met all of our goals, and our specifications have all been met. Throughout the year we have managed to stick to our gantt chart indicated above. We managed to complete both the hardware and software tasks that we we had laid ahead of us in the beginning of the fall. This included completing a functioning phone application that works with the android platform. We have also designed our PCB with our hardware that amplifies the signal from the teensy microcontroller to create the 40 kilohertz frequency for our transducers.Lastly, we were able to implement full communication between these two modules via bluetooth in which packets of information allow for the Shush! alarm clock to work within specs and be easy to use for customers/users of the system. Our prototype has now been put in its sleek casing design and can be mounted on walls to wake up potential users, and hopefully be the alarm clock of choice for the early birds of future generations.

Some design flaws that still remain in the system that could be fixed in the future are fixing a bug in the software for the application and also allowing for louder and higher quality alarms. The software bug can be fixed by allowing for multiple alarms to be added and deleted before closing the app in between each use. Also, if more time allowed we would have implemented an audio shield that would allow for higher quality sounding alarms and even let users play music instead of just tones.

Appendix

## A. Shush! Cost

We are allowed to spend $500 for this project as a budget. Below is a chart of our total expenditures and the cost to buy all our parts in bulk of 1000. The bulk purchase cost is definitely a reasonable price for a market device and we are happy with our 313.78$ cost. Even with that we are still far under our limit.



*Fig 11: Table showing our project expensive with the cost of bulk included. We were able to stay far under our 500 limit and in bulk the Shush device is much cheaper.*

Acknowledgment

We would once again like to acknowledge how helpful Professor Eslami has been throughout the project as well as thank our evaluators Professor Hollot, Professor Janaswamy, and Professor Ganz. We also would like to thank SDP group 19 for allowing us to use their function generator and power supply.

references

[1] “Order The Shock Clock + Wristband.” *PAVLOK Store - Break*

*Bad Habits With Pavlovian Conditioning*, buy.pavlok.com/products/shocking-alarm-clock?gclid=Cj0KCQiA2NXTBRDoARIsAJRIvLwMBBdPevlb9NGs5fPuYox-2rIpCYl-i63FGQh74FgOELm6SN29MwYaAgTSEALw\_wcB.

[2] “11 Sleep Gadgets to Help You Snooze More Restfully, Wake Up

More Refreshed.”*Gadget Review*, 3 Aug. 2015, www.gadgetreview.com/11-sleep-gadgets-to-help-you-snooze-more-restfully.

[3] “Meet Android Studio.” *Android Developers*, 16 Jan. 2018,

developer.android.com/studio/intro/index.html.

[4] “Kinetis K64F Sub-Family Data Sheet.” *Pjrc.com*,

FreescaleSemiconductor, Inc., Sept. 2015,

www.pjrc.com/teensy/K64P144M120SF5.pdf.

[5] “MIT App Inventor.” *MIT App Inventor | Explore*

*MIT App Inventor*, MIT, 17 Dec. 2017,

appinventor.mit.edu/explore/.

[6] “HC-05.” *Electronicaestudio*, ITead Studio, 18 June

2010, www.electronicaestudio.com/docs/istd016A.pdf.

[7] Bratoszewski, Piotr, et al. “Measurements and

Simulations of Engineered Ultrasound Loudspeakers.”

*CMST.eu*, 7 Sept.

2015, cmst.eu/wp-content/uploads/files/10.12921\_cmst.2015.

1.03.006\_Bratoszewski.pdf.

[8] “PCB Design Software.” *PCB Design Rules Reference | Online*

*Documentation for Altium Products*, www.altium.com/.