Shush! The Parametric Speaker Alarm Clock

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

***Abstract*—We introduced Shush!, a parametric speaker alarm clock that will only wake up one person. Shush will consist of a parametric array, microcontroller, and circuitry contained within one mountable box as well as an android app for selecting alarm times and tones, among other things. Shush uses specialized transducers that will emit a audible frequency PWM onto an inaudible ultrasonic frequency allowing for a very directive cone of sound to form. Only when situated within the cone will the original sound be heard, outside of the cone nothing will be heard at all.**

# I. INTRODUCTION

One of the most common complaints that college students will have as we endure our undergraduate educations takes place in the early mornings in the dormitories. 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 humans across the globe that share sleeping space with another person or even have multiple roommates in one bedroom. This amounts to a very large scale affair 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

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

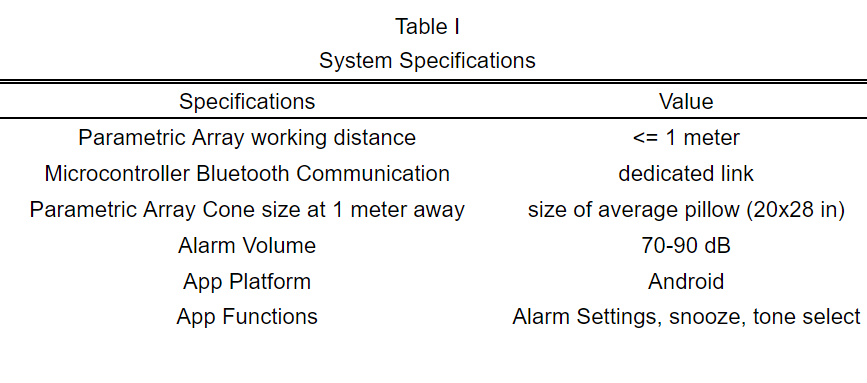
Christos P. Lemonias from North Andover, Ma (email clemonias@umass.edu)

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

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

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 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 does not require someone to wear anything uncomfortable, yet will still only wake up one person without inconveniencing any roommates involved.

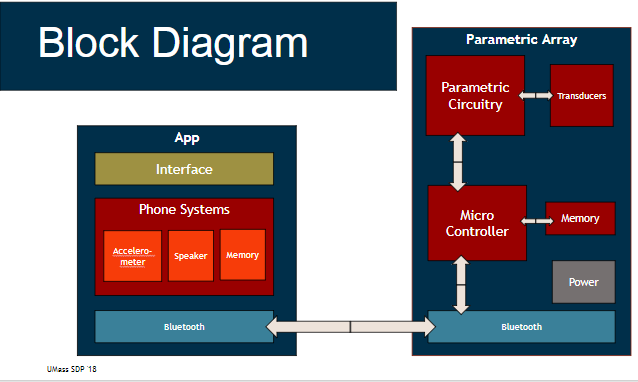
Shush! uses parametric circuitry along with transducers to produce this alarm clock signal at ultrasonic frequencies that demodulate through the air. This technology can be thought of as a “laser beam” of sound that aims the sounds waves in a very directive manner. This directed beam of sound can be 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.



# II. Design

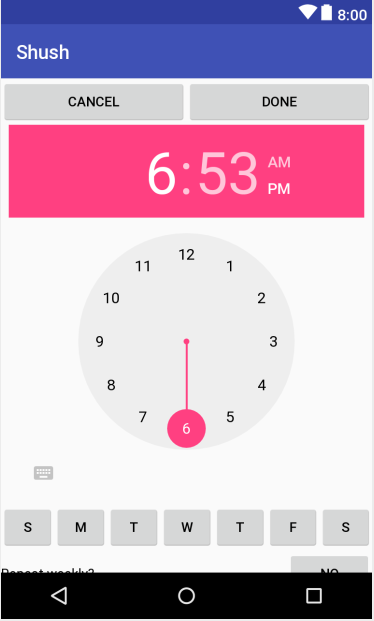
## A. Overview

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 will use parametric circuitry 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 extremely 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. 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.



Above 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 will be the main user interface of our product Shush! Using the App, the user will be able to create new alarms or lullabies. When creating a new alarm the user will be able to choose what time it goes off, on what days it goes off, if it will repeat weekly, the type of alarm, the tone of the alarm, and finally the user can give the alarm a name. When creating a new lullaby the user has all the same options except for the alarm type. That is because the alarm type setting for the alarm allows the user to choose if a tone will be played, if the phone will vibrate, or if the tone will play and the phone will vibrate. Once a user has a few alarms or lullabies set they will see them on the home screen and be able to toggle them on or off or delete them from that screen. Also on the home screen will be a button to connect to the bluetooth module in our parametric circuitry. Once the user has all their alarms and lullabies set and connected the device to the parametric circuitry the app will then push all the updated alarm and lullaby information to the circuitry so that the parametric circuitry as all the data it needs to function as the user desires. Below is the top of the new alarm screen. 

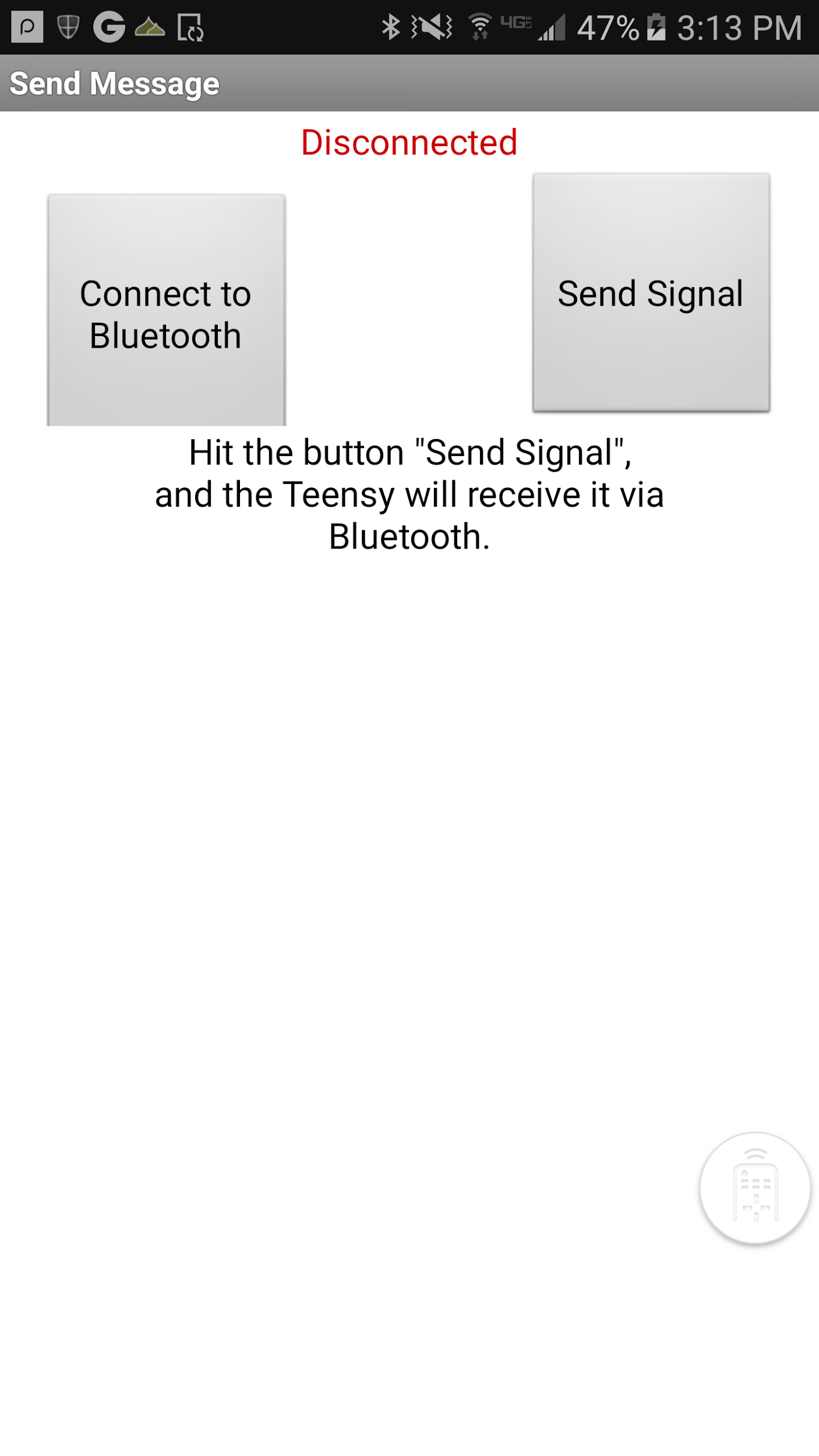
The Android App is being developed in the Integrated Development Environment (IDE), Android Studio. All of the App’s functionality is programmed in Java and the App utilizes XML formatting to design the user interface layouts. Many of our previous courses have taught various portions of programming in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

To complete the Android App we will need to learn how to utilize device functionality such as the speakers, vibration, and memory and then integrate all of that with the code and interface of the App. Also we will need to learn the proper formatting to send all of the alarm and lullaby information via bluetooth to the parametric circuity in a usable manor.

During development the App has been in a kind of 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 use to verify that the layout is correct and that buttons are clickable and do as they should when clicked. Another test that we will need to run is the packaging of data to be transmitted via bluetooth. This will be done by sending test data over a bluetooth link and checking to see if the data is usable on the other side. If it is usable then the App passed the test.

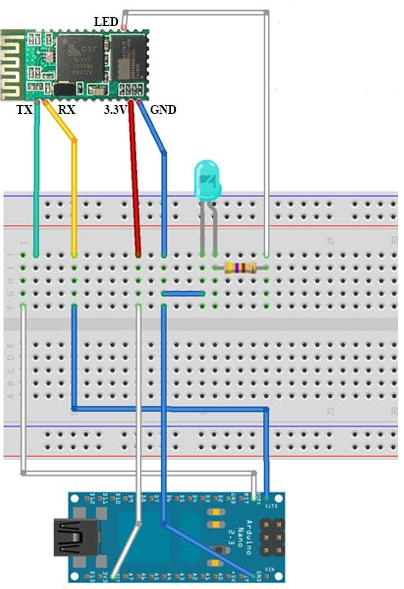
## C. Bluetooth

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

*Bluetooth App Home screen*

The Bluetooth device we chose is the HC-05, 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.

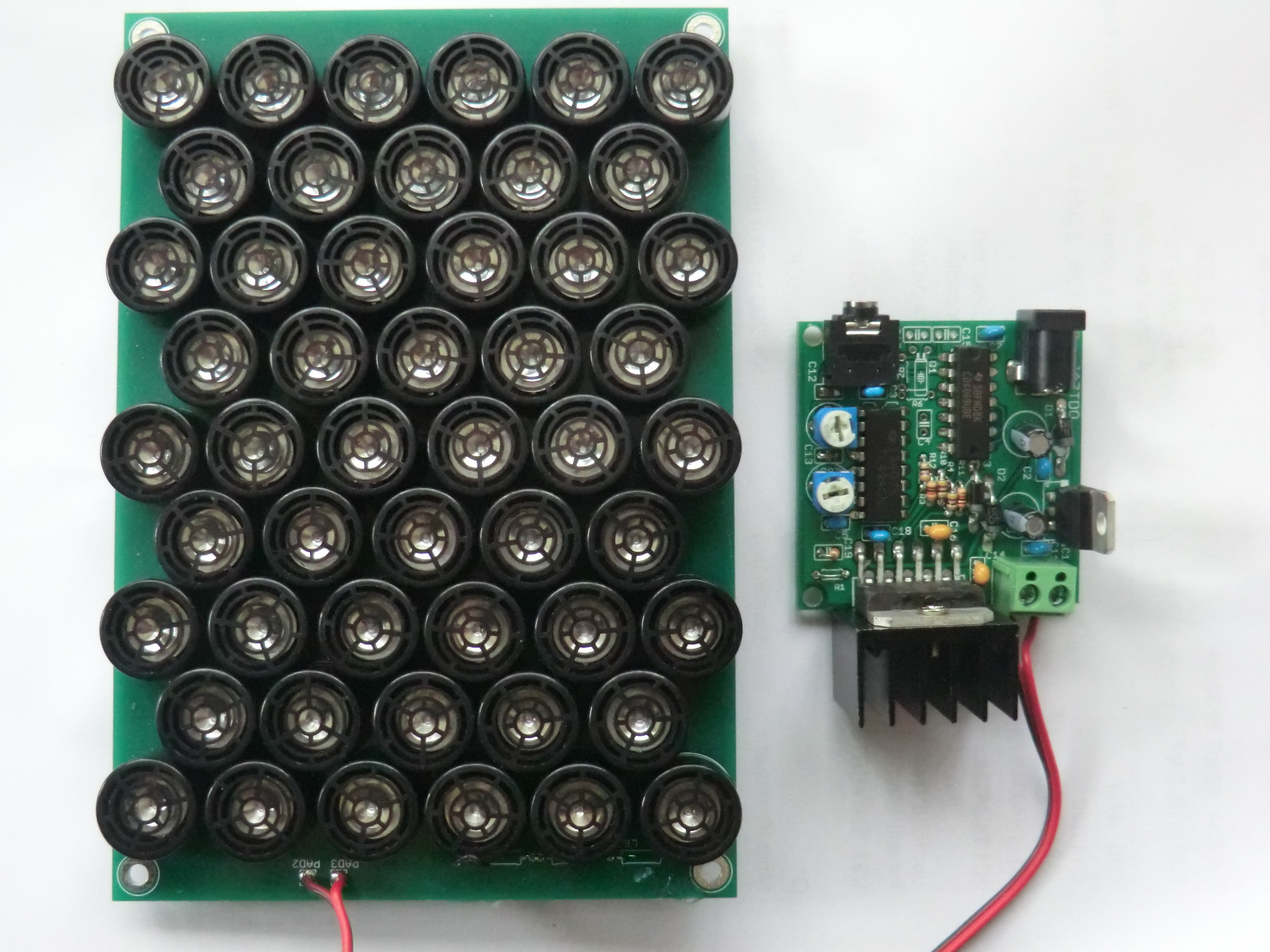
*Bluetooth Test Schematic*



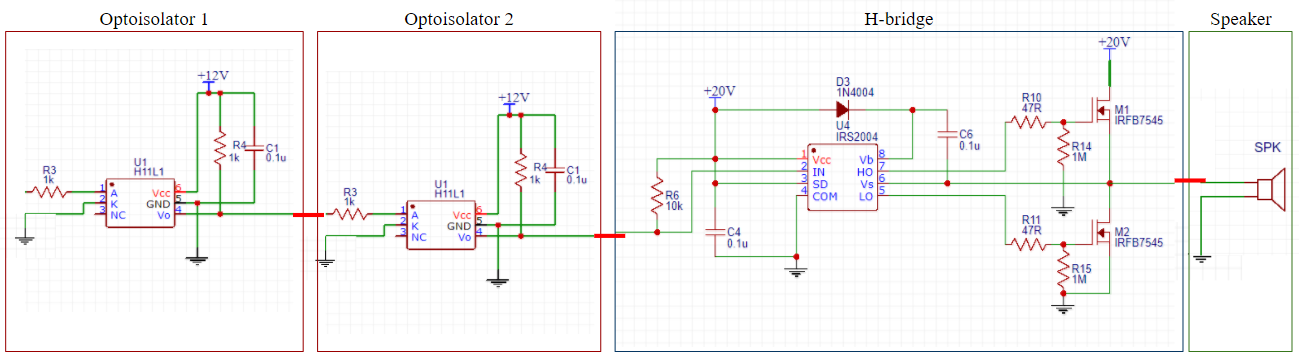
On observation, we were successful in connecting our microcontroller to our application with Bluetooth. To ensure our success, it was important to understand how to use MIT’s App Inventor to build an app with Bluetooth connectivity. It was also important to understand to how use and code Arduino libraries and integrate our written program with the Teensy to guarantee compatibility across the two platforms. My experience as an Undergraduate Instructional Assistant 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

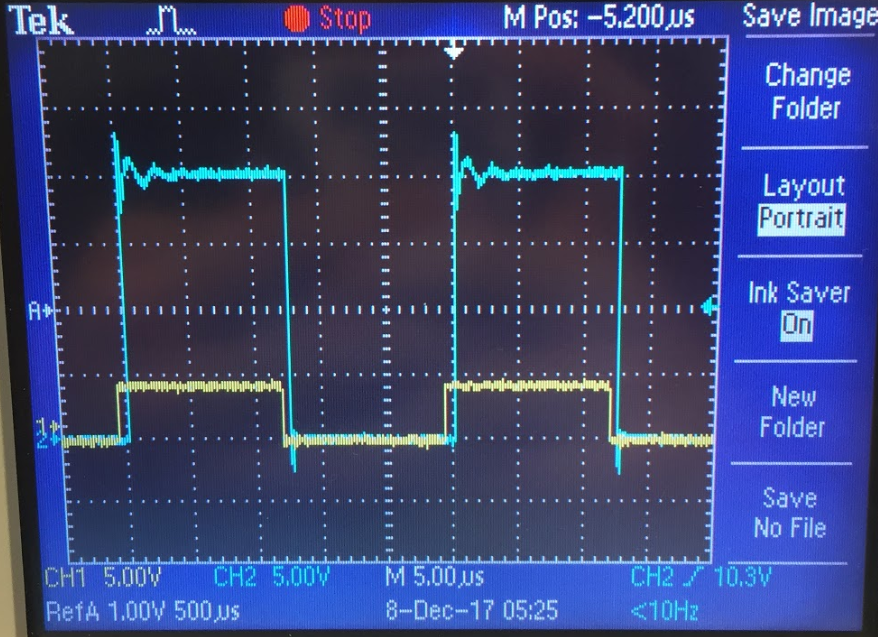
The parametric speaker is what makes this project special. Any normal speaker someone uses will be heard all throughout the room, but with a parametric speaker, you get a directive cone of sound heard by only people within the cone. The way a parametric speaker works are by pulse width modulating the song or tone you want to hear onto a 40KHz carrier signal. This frequency is ultrasonic and is very directive, but just playing the 40K signal will not provide us with any sound because we can’t hear 40KHz. 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.



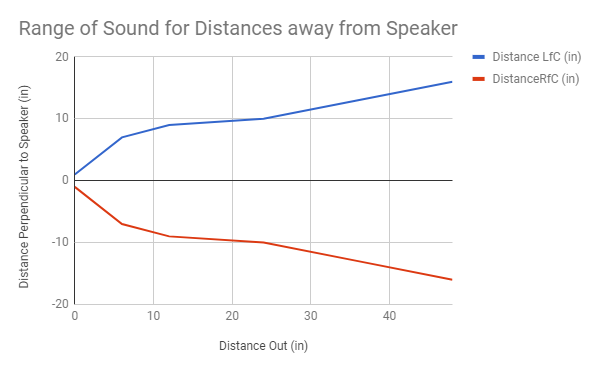
The array is made from a collection of 16mm piezoelectric transducers. The transducers are wired in parallel in a honeycomb pattern. Our research into parametric arrays showed that this would be the optimal positioning. Next, we need to decide the number of transducers and how spaced they are from each 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 current MDR setup using a breadboard for placing out transducers and because of that our optimal spacing can not yet be achieved, but we will only get better as time goes forward and we acquire our pcb. Even with a perfectly setup array we can’t just feed any modulated signal into our transducers. The transducers require 20V to be driven, but our microcontroller only produces 3.3V. This is where our parametric circuitry comes in. Below is the parametric circuitry schematic.



The parametric circuit needs to take in our 3.3V 40KHz signal and output the same signal to the array at 20V and with enough current to drive the array. Our circuitry begins with an optoisolator, this acts as protection for our microcontroller to keep any harmful currents from going backwards into our teensy 3.5. We then use a half-bridge driver to up our voltage to 20V. 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 20V 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 set up provides us with in output identical to our input, but with a voltage of 20V opposed to 3.3V. The oscilloscope reading below shows our circuitry working.



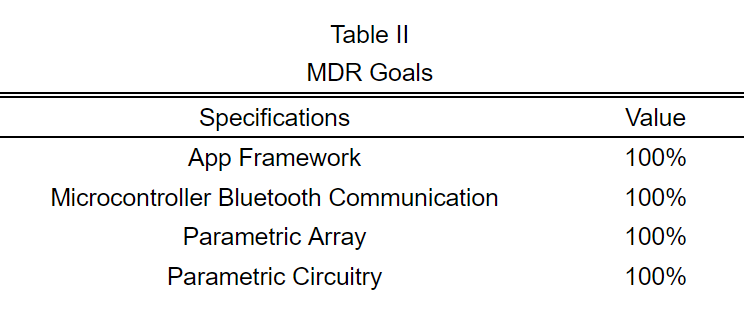
Once our speaker was working we needed to test it’s range. As opposed to just using our ears we used a decibel meter to measure volume at specific distances from the array and how far out on all sides you continue to hear the signal at those distances. Below is the graph of our results.



We were able to achieve a range of 26in x 26in at 1m which is very close to our specifications. This will only improve as we optimize our placement.

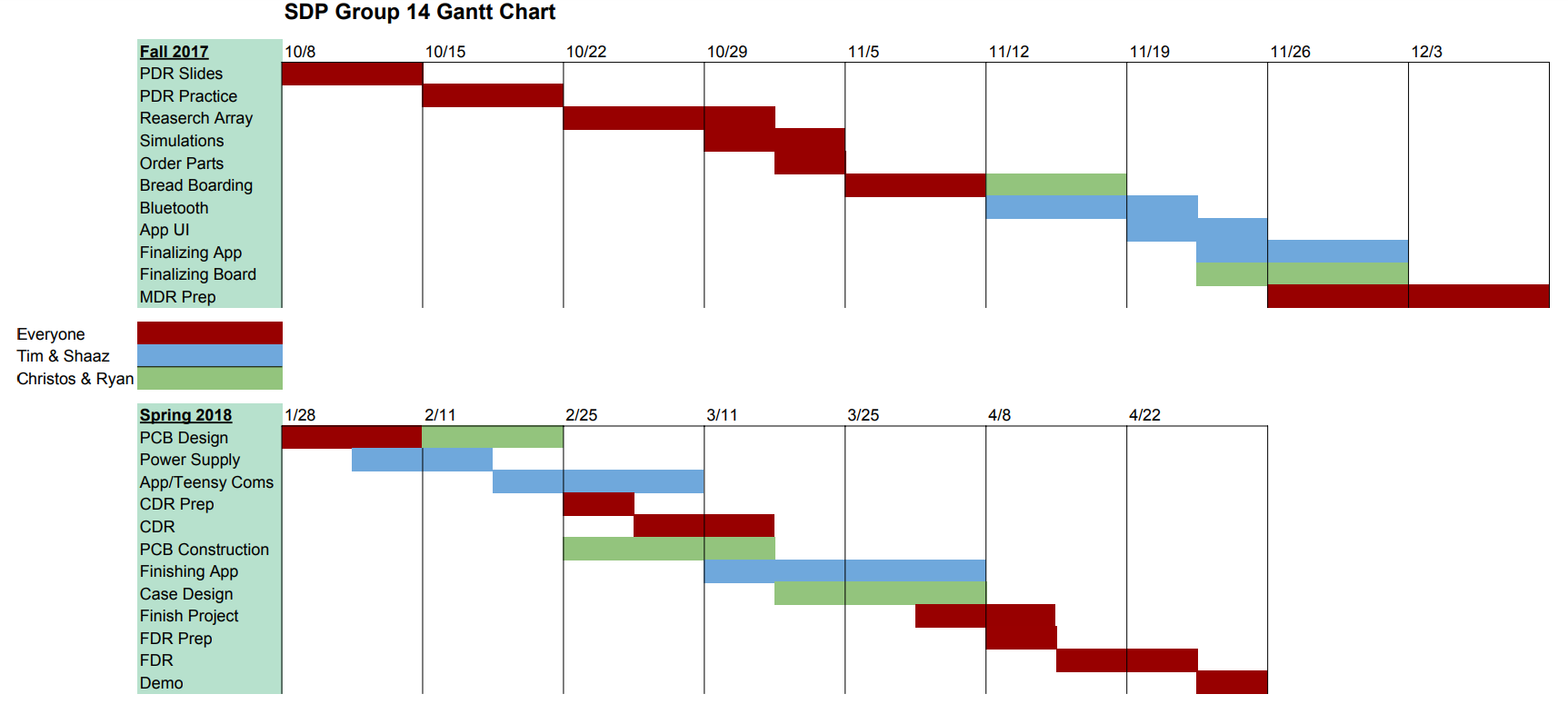
# III. Project Management

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



The goals that we set for ourselves at the time of PRD were accomplished by our demonstration at MDR. We showed that we could communicate with our microcontroller via bluetooth, which will be important for the app going forward. We also demonstrated the framework for our android app and how everything will be laid out for the user. Finally we were able to build 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. For our next semester we will need to bring the bluetooth over into our app as well as flesh out the way our app and microcontroller will communicate. The microcontroller will need to store songs and settings which we will do using a microsd card. We must also work on getting our teensy to output a PWM signal to our circuitry as opposed to a function generator. Next we need to focus on what kind of voltage supply our circuitry will need and how we can implement that into our design without adding any unnecessary weight. Finally we need to design our PCB that will go into our final product.

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 modulate signals from our teensy 3.5 microcontroller. Our Gantt chart can be found below in figure 3.



# IV. Conclusion

Summarize the current state of the project. How did you get there? Where are you going?

Describe your plans for the future? What difficulties do you expect? How will you get there?

As of the conclusion of the fall semester, we have met all of our MDR goals, but there is still significant work to be done to stay on track with our projections for the spring semester. So far, we have been successful in terms of staying on schedule and keeping up with our deliverables indicated on our Gantt chart.

Looking ahead to next semester, there a number of steps necessary to stay on track. Starting with the Bluetooth implementation, it is important 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 are considering for the Teensy’s software is to employ bit-masking, such that it can differentiate between tone signals and time signals which are received on the same pin. Another important step to cover is bridging the gap between our stored wav. files and the input signals of the array, which will require more research on our end. One possible solution is to integrate an audio shield into the project to process the wav. files, and directly feed the output into the parametric circuitry.

When these two steps are resolved, we can then test the entire critical path of the project by providing input to the mobile app, receiving it over Bluetooth, testing to see if has been properly processed by the Teensy and parametric circuitry, and lastly outputs sound correctly through the speaker. When this is done, we can begin designing our final product, which will include designing the main PCB and delivering and regulating power to the overall system. We hope to have these completed by CDR. Finally, after CDR our goal is to tie up any remaining loose ends such as delivering a finished app, and build the enclosure for our system to present on Demo Day.

Appendix

## A. Shush! Cost

We are allowed to spend $500 for this project as a budget. Below is a chart of our current expenditures and possible future purchases. Even with that we are still far under our limit.



Acknowledgment

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