ACOUSTIC MOTION TRACKING

By

Hojin Chun

Sean Nachnani

Final Report for ECE 445, Senior Design, Spring 2018

TA: Yuchen He

02 May 2018

Project No. 35

Abstract

The purpose of this project is to create a device capable of identifying a user's hand motion gestures using acoustic sound wave reflections. This device uses a centrally located speaker to send out sound pulses, and capture the subsequent audio using a four-input microphone array. The data is sampled by a microcontroller that then sends it through a wireless chip to a server. This server handles the signal processing that estimates the three-dimensional location of the user's hand at each sound pulse, and aggregates those vectors over time to identify a gesture.

Contents

1. Introduction

1.1 Problem Statement and Purpose

People are always striving for a better life and are trying to make each day more comfortable than the last. As technology advances further, and AI comes more into focus, we try to satisfy the comfort needs by building autonomous cars, creating smart cities, and building robots that can eventually replace humans. Even now, we have products like Alexa, Siri, Bixbi, and many more that can understand a person's vocal commands and get data or perform tasks for the user, without them having to lift a finger.

Our objective is to build a device that use sound rather than video as a means of motion recognition. Current devices mentioned above are limited to only using natural language processing to interpret a user's need. We want to expand upon this further and allow devices to perform commands using simple gestures. Not only our device will make people's life more comfortable with this feature, but also will help people with speech impediments or certain accents, allowing them to still use these devices.

1.2 Subsystems

Our project features 6 main components, as it is shown in Figure 1 below.

- Processing Module - The Microcontroller in the Processing Module handles transferring the data collected from the Microphone Array to the Wireless Chip. It is to act as the bridge between all the hardware components and the software running on external device.
- Transmission Module - The Wireless Chip in the Transmission Module sends the collected audio data over 802.11b/g/n radio protocols to the server, which then estimates the motion of the user's hand over time to identify the gesture performed.
- Microphone Array Module - The Microphone Array Module collects the audio data while the Speaker is playing the sound pulses. It has 4-input microphone array that allows for at least a 48 Khz sample rate.
- Speaker Module - Speaker module reproduces sounds up to 24 kHz.
- Power Module - The power module consists of AC/DC convert and Buck convert and converts 120 VAC to 5VDC to supply the speaker and convert 5 VDC to 3.3VDC to supply rest of the modules.
- Signal Analysis Module - Signal Analysis Module has software implementation involved pulsing a pseudo-random wave, and calculated the distance of the hand by measuring the peaks of the Channel Impulse responses received from the microphone array.

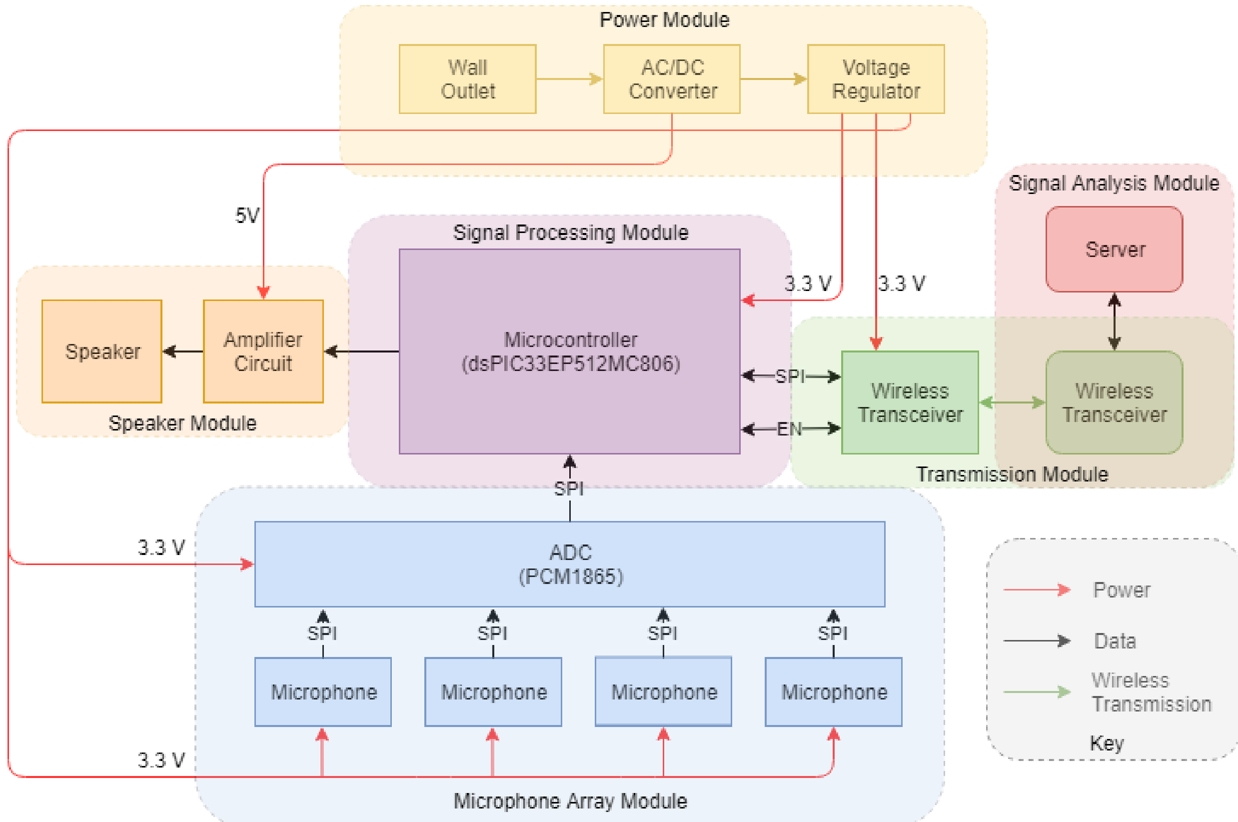

Figure 1. Block Diagram

2 Design

2.1 Micro Processing Module

This module contains both the MCU and the Wireless chip. We chose the dsPIC33EP512MC806 MCU for its 512kB onboard flash memory, its integrated digital signal processing capabilities, and its operating voltage of 3.3V. We chose this microprocessor though with the intention of using the integrated digital signal processing capabilities to pre-process the data and reduce the server overhead. We also started out the design by using 100.0ms long sound pulses, but later reduced the duration by half to reduce the overall runtime on the server. A simpler microcontroller can be chosen to act as the link between the Microphone Array and the wireless chip, if it contains enough onboard flash memory hold the audio data for one pulse. This however comes at the cost of requiring a much higher performing server to run the computations.

The microprocessor in this unit triggers the speaker to play a 50.0ms sound pulse every 0.5 seconds, and collects the digitally sampled data over SPI (Serial Peripheral Interface) from the Microphone Array for a 100.0ms duration. The data is then sent over SPI to the Wireless chip, and stored in its 4MB flash memory. Once the audio data for all four sound pulses has been collected, the wireless chip transmits the data over 802.11b/g/n radio protocols to the server for signal processing of the data.

2.1.1 Circuit Schematics

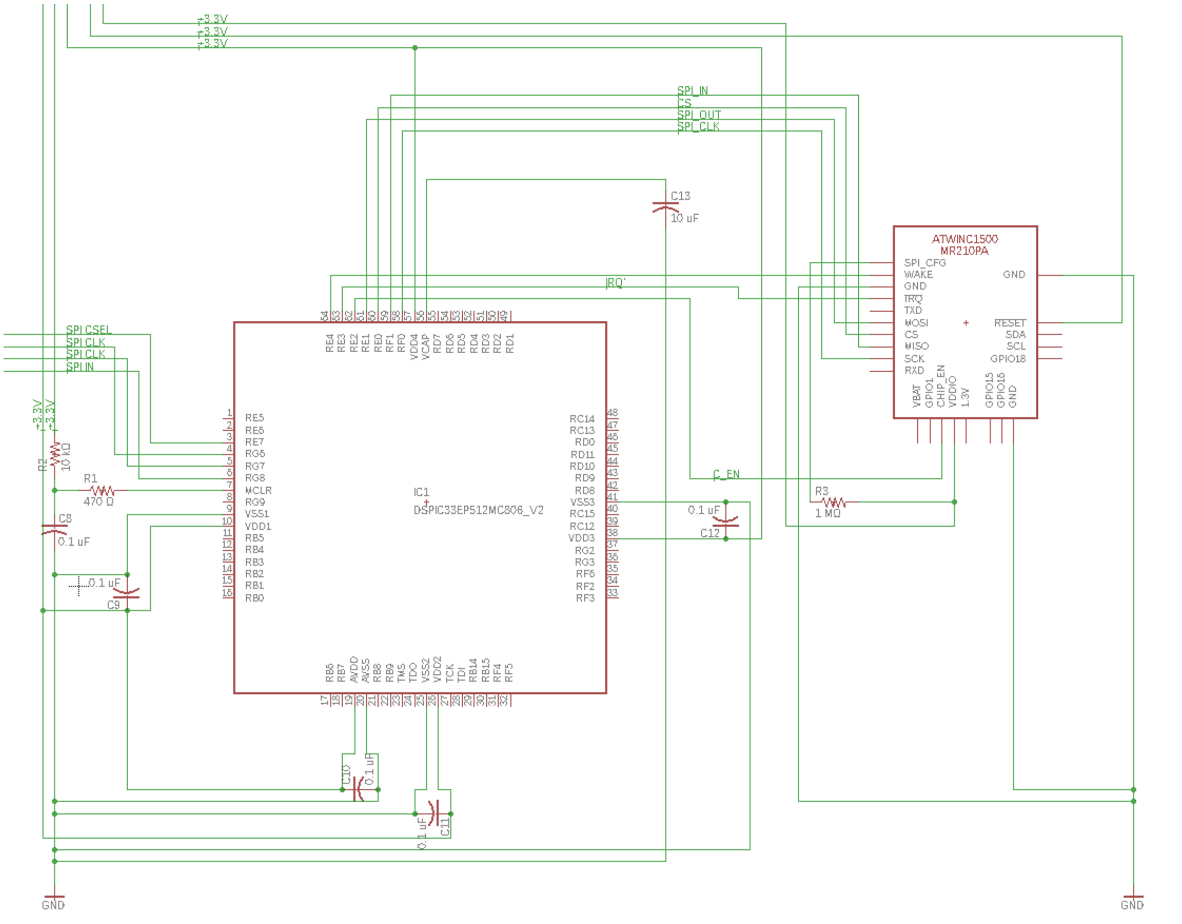

Figure 2: Processing Module Schematic

2.2 Microphone Array

The Microphone array consists of four analog MEMS microphones and an Analog to Digital Converter (ADC). As displayed in Figure 15. the microphones were placed on the corners of the board, to reduce the chance that a reflection is blocked by the speaker in the center. This module is responsible for capturing the sound pulse reflections off the objects in the room, and digitally sampling them to send over SPI to the MCU. Even if one microphone does not receive a proper reflection, the other three will always receive it, and a distance vector can always be calculated. These analog MEMS were chosen instead of digital MEMS microphones due to their increase in sensitivity around the 20kHz-24kHz range as seen below in Figure 3. The TI ADC was consequently picked for its 48kHz sampling rate and the ability to sample from each of these microphones simultaneously.

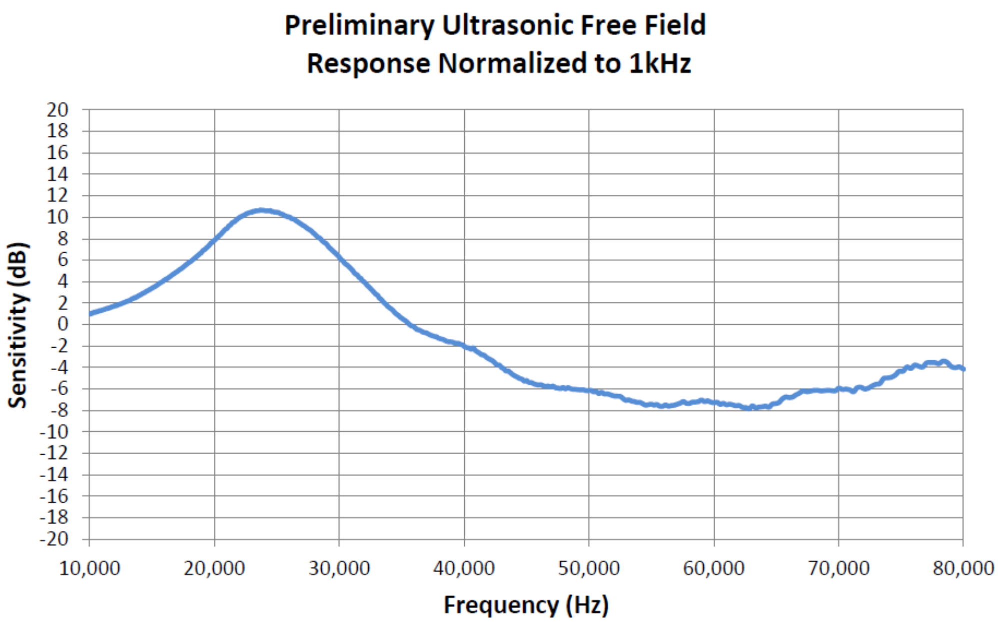

Figure 3. Microphone Sensitivity Curve

2.2.1 Circuit Schematics

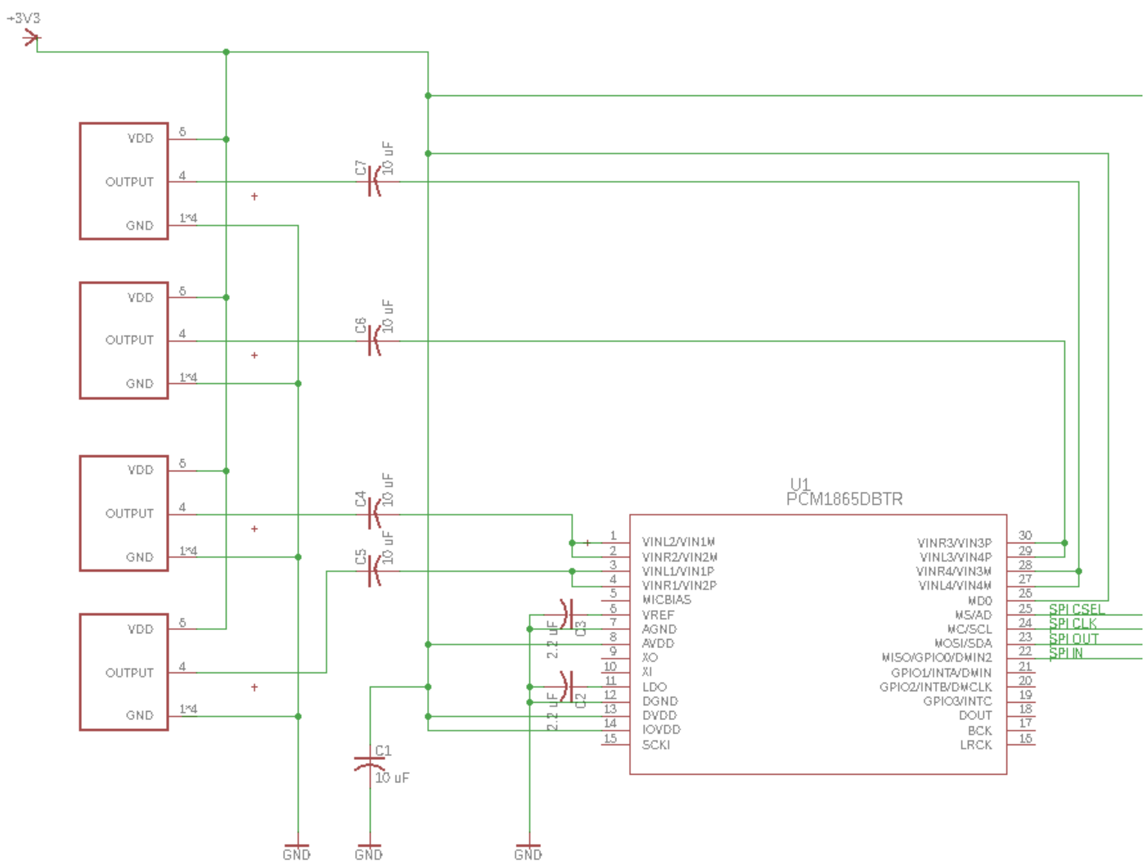

Figure 4. Microphone Array Schematic

2.3 Speaker Module

The XS-GTF1027 speaker outputs a sound pulse that spans a bandwidth of 20kHz -24kHz, and is centered atop the microphone Array, pictured in Figure 15. We purchased and used an off the shelf amplifier, MAX9744, to amplify the signals from the MCU. We chose this speaker specifically for it's ability to generate sounds above the audible hearing range, and for its significantly cheaper price point to other similarly capable speakers.

2.4 Power Module

We used a wall outlet to power the device, implementing an AC to DC adapter that can convert the 120V at 60hz to 5V DC voltage. Since the different parts of the device have different voltage requirements, we also used a voltage regulator to allow each component to receive the correct power specifications.

The Schematic for the power supply is displayed below in Figure

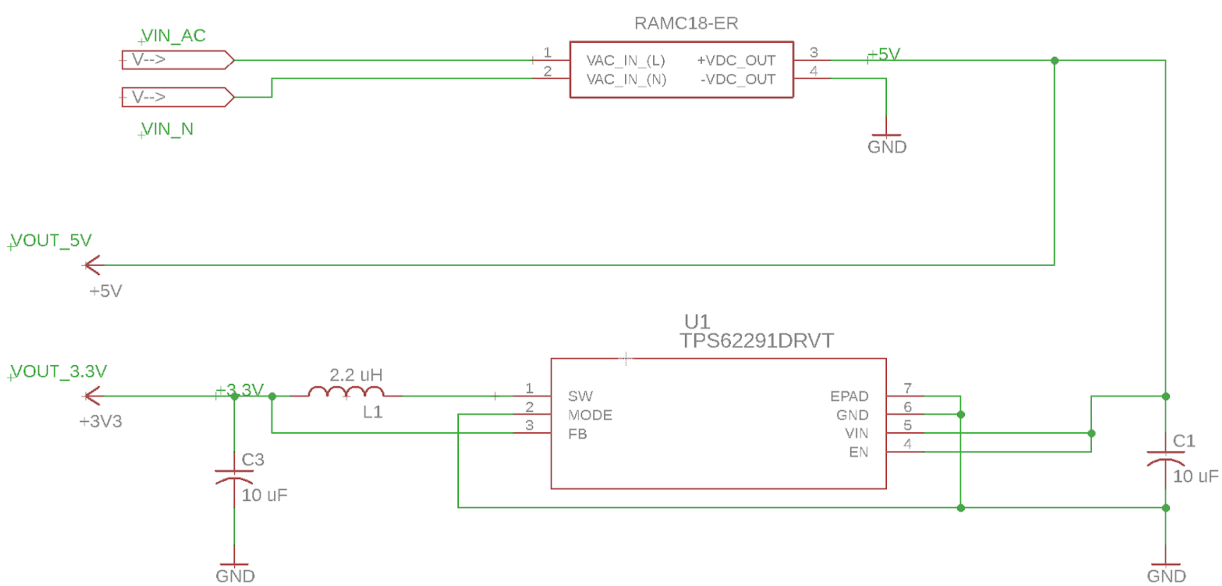

Figure 5. Power Supply Schematic

2.4.1 AC/DC Converter

The converter chosen was the RACM18-05SER, as it can converter from 120V AC to the 5V DC required for the speaker and amplifier components. It supplies at least 2.5A of current to the device, with the majority of that being used by the Speaker.

2.4.2 Voltage Regulator

Aside from the speaker, the rest of the components were chosen to run at 3.3V, so a buck converter was necessary to achieve that voltage requirement. We chose the LXDC55KAAA-205 as it can handle the peak current from the AC/DC converter and supply the remaining components with 3.3V and a current of up to 500mA needed. The calculation for the current drawn by these components are shown below in Table 1.

Table 1. Current Calculations

Component	Part Number	Max. Current Draw	Quantity	Subtotal
Microcontroller	dsPIC33EP512MC806	70mA	1	70mA
Wireless Transceiver	ATWINC1500	268mA	1	286mA
ADC	TI PCM1864	43mA	1	43mA
Microphones	SPU0410LR5HQB-7	5mA	4	20mA
Maximum output current requirement for DC/DC converter				419mA

2.5 Server

The Server is responsible for receiving the audio data and processing it to identify the user's hand gesture. We used a laptop to function as a server for this device. It can connect wirelessly to the device and run the Python code necessary to process the audio signals.

2.5.1 Software Signal Processing

The was written using Python, takes advantage of the robust signal processing libraries (i.e SciPy, NumPy, PyAudio). To calculate the time delay between the reflections of the objects in the room, the function calculates CIR for each pulse and series of reflections. We first split the signal received into sections, starting from the first part of a pulse and spanning the next 2401 samples to capture the reflections.

$$\begin{aligned} X_k &= \sum_{n=0}^{N-1} x_n \cdot e^{-i2\pi kn/N} \\ &= \sum_{n=0}^{N-1} x_n \cdot [\cos(2\pi kn/N) - i \cdot \sin(2\pi kn/N)], \end{aligned} \tag{1.1}$$

We then calculate the Channel Impulse Response by using Equation (1.1) to transform both the pulse section P_{iw} and the initial pseudo-random pulse played by the speaker S_w.

$$x_n = \frac{1}{N} \sum_{k=0}^{N-1} X_k \cdot e^{i2\pi kn/N}, \quad n \in \mathbb{Z}, \tag{1.2}$$

By dividing P_{iw} by S_w and using Equation (1.2) to transform it back into the time domain, we get the CIR. To see only the reflections due to the hand, we then divide this CIR by a calibration CIR that was done prior to the hand motion. To increase the accuracy of the peaks in the CIR, a pulse signal of large frequency bandwidth with unique phase is played from the speaker at repeating intervals. The distance from the hand to each microphone is calculated by using Equation (2).

(2)

$$Distance = \frac{Peak_1 - \ Peak_2}{Sample\ Rate} * 343\ \frac{m}{s}$$

Where, the values for $Peak_1$ and $Peak_2$ are their location in CIR array. Dividing that by the sample rate of the ADC (48 kHz) gives us the time, and multiplying by the speed of sound provides us the distance.

We calculate $Peak_1$ by finding the first location in the array that is above a certain threshold. The threshold is determined empirically over a series of test runs. After finding the first peak, the next one is calculated to be the largest magnitude peak that correlates to $Peak_1$ within range of 500 samples, as it corresponds to 3.5m. If no peaks are found within this range, then that is expanded further. The peak pairs are then correlated to the peak pairs in subsequent pulses to determine the correct pairs.

The Calibration CIR allows us to take advantage of the static reflections by the room's environment. Figure 6 below illustrates the ideal principle. The first shows all the reflections due to the environment, while the second was taken 0.5 seconds later, correlated to then subtracted by the first. The Y axis for both graphs is the magnitude of the signal, and the x axis is time.

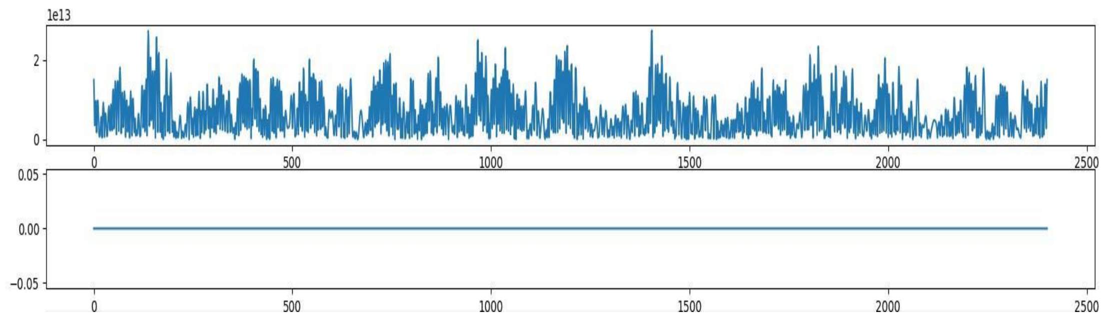

Figure 6. Ideal Calibration

2.5.2 Software Flowchart

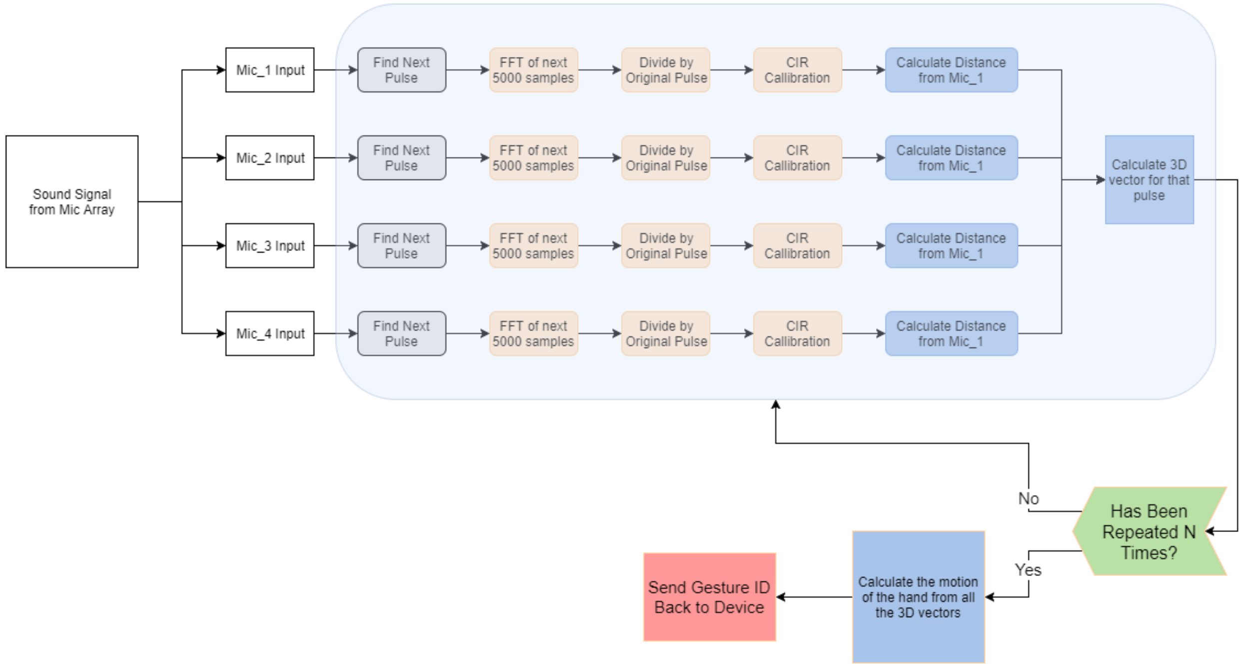

Figure 7. Software Flowchart

Figure 7 above, illustrates how the code functions. In the flowchart, N is the number of pulses. We found that 4 pulses provide us with enough information for the simple gestures of: horizontal motion, vertical motion, and Z-axis motion (moving the hand towards the speaker). In future work, we want to increase the number of pulses and execute more complicated gestures.

3. Design Verification

3.1 Micro Processing Module

We were not able to properly verify this module as a whole, due to the difficulties we ran into while trying to program the MCU outside of the starter kit. We were able to independently test the Wireless chip, by connecting it to the laptop used, and transfer data to the laptops terminal. We were not able to test the MCUs SPI or data transmission functionality though. We used MPLABX by Microchip to program the microprocessor, but it did not support capabilities of simulating SPI or reading in a file into the MCU. We were only able to show that it could connect over USB, which was not part of our requirements.

3.2 Microphone Array

For the verification of Microphone Array, we first had to solder the microphone MEMs and ADC first. We first had a trouble soldering the parts onto the PCB board as we have never soldered a surface mount chips before. We tried the original way first but it seemed to not work. We had to do some research and figured out that we had to use solder paste to solder it on.

The way we tested the microphone first was that we had to test to see if microphone registered sounds up to 24 kHz. We first had to download a sound that was above 24 kHz. Then we had to code in Arduino so that it showed signals that it picked up. Below are the signals that we tested for the microphones.

3.2.1 Signal Results

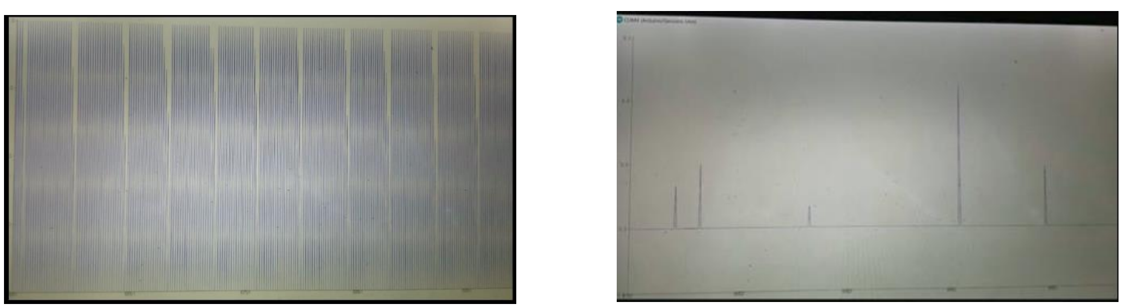

Figure 8. Signals from 3.3V (left) and GND (right)

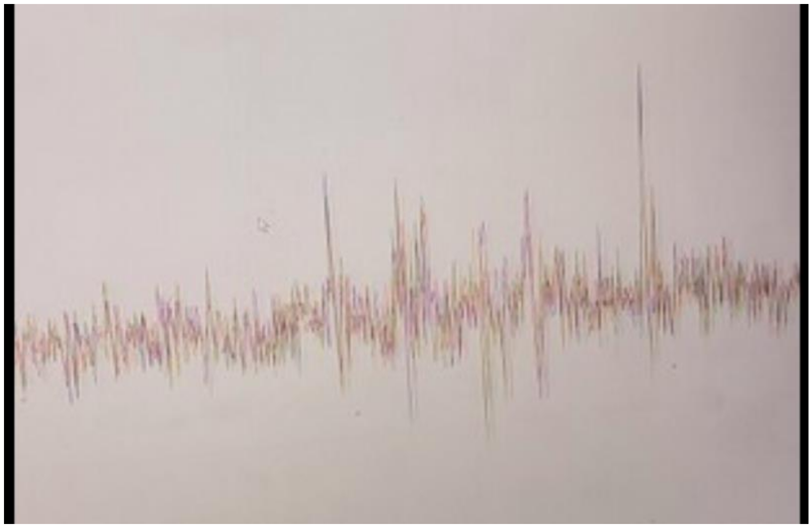

Figure 9. MEMS Microphones Signals

As seen in the graph above, there are clear distinct signals from the 3.3V, GND, and from the microphones. We can see that microphones picked up the sound above 24 kHz because when we beeped the sound above 24 kHz, you can see the peaks in the Figure 12. Individual signal from the microphone is explained more in the software section. For our project, we were not able to test our ADC due to time constraint.

3.3 Speaker Module

The Speaker was able to play sounds up to 24kHz and performed as intended. We performed a frequency sweep from 60Hz to 24kHz.

3.4 Power Module

For the verification of the power module, we first had to solder the buck converter and AC/DC converter. Power module was very straightforward to test. We just had to measure the output voltage from the AC/DC converter and from the buck converter. Our R&V table said that the output voltage from buck converter is 3.3V with errors less than 5%. However as seen in Figure 13, the voltage we measured is 3.7V. The reason why this might have happened is that we were supposed to use 2.2uH inductor. However, they ran out 2.2uH inductor. So, we had to use 2 1.1uH inductor and connect them in series. As seen in Figure 14, the way inductor is connecting, it is not completely in series. And when we put the voltmeter in between the inductors, we got some voltage drop across it. So that might have been the cause of the 3.7V.

3.4.1 Power Module Results

Figure 10. Output Voltage Buck Converter

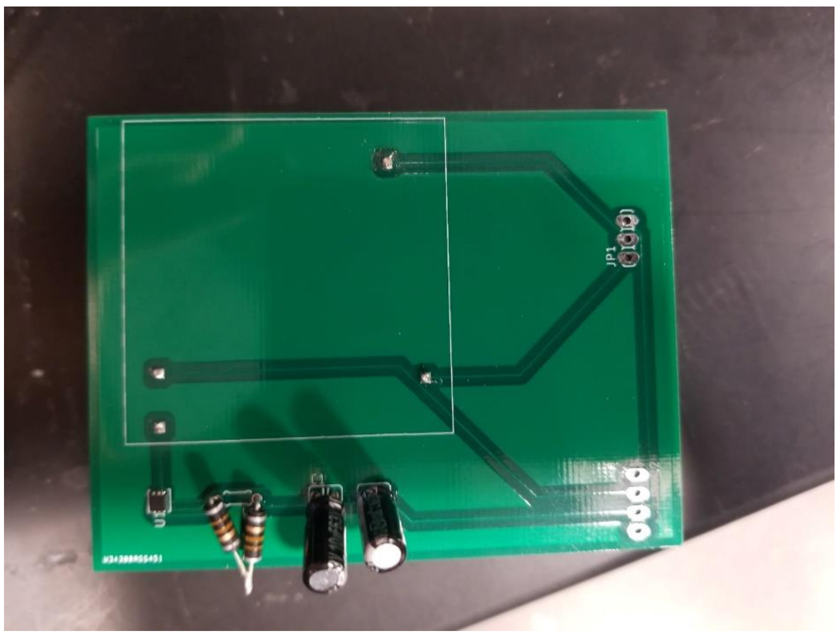

Figure 11. Power PCB Board

3.5 Server

As we were not able to fully integrate the MCU with the Wireless chip, we collected the microphone data using a Raspberry Pi and a 4-input Microphone Array expansion to test the software and gather results.

3.5.1 Software Signal Processing

While testing the ReSpeaker and Raspberry Pi in the lab, we found that there were considerable reflections due to all the surrounding objects, making it difficult to achieve a good Channel Impulse Response. We then moved to testing in a more open room, yet still found there were considerable reflections that appeared in the results of subsequent audio pulses. We therefore revised the algorithm slightly to use the CIR from the last pulse as calibration for the current pulse being processed.

Moreover, to reduce the noise in the signal we implemented a series of Weiner filters from the SciPy library. We applied a less aggressive filter on the initial CIR result for each pulse, taking the average across 10 samples, while applying a more aggressive average after subtracting the calibration CIRs, using an average of 20 samples. Figures 12 and 13 in Appendix B show the different results for the horizontal motion and vertical motion respectively. In Figure 13 the Wiener filter averages were taken across 20 samples and 30 respectively to show the even further reduction in noise. In each of the figures listed in Appendix B the topmost graph represents the CIR of the first pulse in blue, with the filtered version in orange. The remaining graphs display the filtered different of the CIRs. The red-highlighted sections of each represent the peaks that were found to correlate across the CIRs and were determined to be the peaks between the initial sound pulse and the reflection from the hand.

In Figure 13, the third graph from the top does not contain a red-highlighted section. This abnormality was because speaker tested with the Raspberry Pi blocked the reflection to the microphone at that specific hand location. In our own Microphone Array design, we had accounted for that issue by increasing the size of the board and elevating the speaker even more above the microphones. Even with this issue though, the algorithm was still able to identify the gesture, as this only affected one microphone. The results for which are below in Table 2.

Table 2. Horizontal Test Results

	Pulse 2	Pulse 3	Pulse 4
Mic 1	144.365cm	144.012cm	150.177cm
Mic 2	151.465cm	151.987cm	157.877cm
Mic 3	151.492cm	151.849cm	157.567cm
Mic 4	144.389cm	144.211cm	150.245cm

To determine a horizontal gesture, it was found that if the user was facing the device, the second and third pulse differed slightly, while the fourth pulse was much larger. For a vertical gesture, it was found to be a significant increase between each subsequent pulse. For the gesture of motion towards the device (Z axis gesture), it was found to be a significant decrease between each subsequent pulse.

4. Costs

4.1 Parts

Table 3. Parts Cost

Part	Manufacturer	Retail Cost ($)	Bulk Purchase Cost ($)	Actual Cost ($)
Mircrocontroller	Microchip	$7.42	$7.42	$7.42
Microphones	Knowles	$0.94	$0.94	$0.94
ADC	TI	$3.59	$3.59	$3.59
Speaker	Adafruit	$14.95	$14.95	$14.95
Amplifier	Adafruit	$19.95	$19.95	$19.95
Wireless Transmitter	Microchip	$7.84	$7.84	$7.84
AC/DC Converter	Recom	$37.00	$37.00	$37.00
Voltage Regulator	Murata	$4.14	$4.14	$4.14
PCBs	PCB Way	$10.00	$10.00	$10.00
Total		**105.83**		**105.83**

4.2 Labor

Table 4. Labor Cost

Team Member	Hourly Rate	Total Hours	Total Cost (Rate*Hours*2.5)
Sean Nachnani	$50.00	160	$20,000
Hojin Chun	$50.00	160160	$20,000

5. Conclusion

5.1 Accomplishments

Our accomplishments is that we were able to fully test individual part. We were able to show most of the data for the R&V table. The most notable accomplishment is with the MCU board. Although we couldn't use our PCB to test MCU, we were able to test with the Dev Kit we got. With that, we were able to get signals and data where it showed that it accurately detected the simple gestures and the distance from the hand.

5.2 Uncertainties

During the design, one of the big problem was to able to figure out how to solder the surface mount chips onto PCB. We had no experience before this project and we had to research in order to solder the surface mount chips. We had to use a solder paste and use heat gun instead of the regular solder iron. Another big problem we ran into was the MCU board. Because MCU board had so many pins and we had learned the language from the beginning, it took us a while to figure it out. Also, it didn't help to have the board arrive week before the final demo was due. Moreover, we imagined that when we bought the Dev Kit, we thought we could just detach the MCU chip from the board. However, as mentioned before, since the pins were so small, there was no way of us detaching it and soldering back onto our board.

5.3 Ethical considerations

For our project, there are few safety concerns. As mentioned above, our device will be plugged into a wall outlet. To be able to do that, we will have to have a AC/DC converter. The danger comes when we are dealing with wall outlet voltage at 120V AC and converting it to 24V DC. We will need to make sure the wall outlet contains a ground using one hand method and we will also need to make sure that AC/DC conversion is off limit from the user so they never have to come into contact with high voltages.

When dealing with high voltages, the concern of large current comes along with it. With large current, it can also dissipate heat. So when we are dealing with high voltages and high current, we will have to careful to the heat and we will make sure that the user will never be exposed to the excessive heat.

We are responsible for all decision we make for the design of our device and it is our responsibility to disclose any issues that might be dangerous to the user per Section 1 of the IEEE code of Ethics[4].

Lastly, since our device involves microphone and server to process the data, we need to be careful about privacy issues. As it was for the issue for Alexa, where the idea that Alexa is always listening or may somehow incriminate someone can be issue for owners of Alexa. To protect everyone's privacy, our device won't be on all the time, and will only take in data when you start the device, and it will mostly keep the data locally and not share or only upload anonymously or encrypted. We believed that if our device is properly and well designed, we will lessen these hazards to create an enjoyable experience for the user.

5.4 Future work

For our future work, we need to integrate MCU fully with the rest of the components. Due to time constraint and being stuck on MCU for a while we were not able to integrate everything together. For future work, we would like to fully integrate the device and have it to work. Also, we would like to use FMCW (Frequency Modulated Continuous Waveform) Radar Techniques instead of Psuedo-Random wave pulse. This will allow us to also take advantage of Doppler effects and calculate velocity, further increasing the accuracy of the device.

References

1. W. Mao, J. He, and L. Qiu, “CAT: High-Precision Acoustic Motion Tracking,” Proceedings of the 22nd Annual International Conference on Mobile Computing and Networking - MobiCom 16, 2016.

2. R. Nandakumar, V. Iyer, D. Tan, and S. Gollakota, “FingerIO,” Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems - CHI 16, 2016.

3. N. Roy, H. Hassanieh, and R. R. Choudhury, “BackDoor,” Proceedings of the 15th Annual International Conference on Mobile Systems, Applications, and Services - MobiSys 17, 2017

4. IEEE (2017). IEEE Code of Ethics. [Online]. Available: http://www.ieee.org/about/corporate/governance/p7-8.html [Accessed: 7- Feb- 2018].

5. Microchip.com, “16-Bit Microcontrollers and Digital Signal Controllers with High-Speed PWM, USB and Advanced Analog”, 2012. [Online]. Available: http://ww1.microchip.com/downloads/en/DeviceDoc/70616g.pdf

6. Microchip.com, “2 Mbit 2.3-3.6V SPI Serial Flash”, 2013. [Online]. Available: http://ww1.microchip.com/downloads/en/DeviceDoc/20005135B.pdf

7. Digikey.com, “Zero-Height SiSonic Microphone”, 2013. [Online]. Available: https://www.digikey.com/product-detail/en/knowles/SPU0410LR5H-QB-7/423-1139-1-ND/2420983

8. ti.com, “PCM186x 103 dB 2ch and 4ch Audio ADCs With Universal Front End (Rev. A)”, 2014. [Online]. Available: http://www.ti.com/product/PCM1864/datasheet/abstract#SLAS8312137

9. adafruit.com, “20W 4 Ohm Full Range Speaker - XS-GTF1027,” [Online]. Available: https://www.adafruit.com/product/1732

10. adafruit.com, “20W Stereo Class D Speaker Amplifier with Volume Control”, 2008. [Online]. Available: https://cdn-shop.adafruit.com/datasheets/MAX9744.pdf

11. microchip.com, “ATWINC15x0-MR210xB IEEE® 802.11 b/g/n SmartConnect IoT Module”, 2017. [Online]. Available: http://ww1.microchip.com/downloads/en/DeviceDoc/70005304B.pdf

12. recom-power.com, “RECOM AC/DC-Converter”, 2017. [Online]. Available: https://www.recom-power.com/pdf/Medline_AC-DC/RACM18-ER_W.pdf

13. murata.com, “LXDC Series”, 2014. [Online]. Available: https://www.murata.com/~/media/webrenewal/campaign/ads/america/udcdc/lxdc_product_brief.ashx

14. github.com, “ReSpeaker-4-Mic-Array-for-Raspberry-Pi”, 2017. Available: https://github.com/SeeedDocument/ReSpeaker-4-Mic-Array-for-Raspberry-Pi/blob/master/src/ReSpeaker%204-Mic%20Array%20for%20Raspberry%20Pi%20%20v1.0.pdf

15. wikipedia.org, “Discrete Fourier transform”, 2018. Available: https://en.wikipedia.org/wiki/Discrete_Fourier_transform

Appendix A Requirement and Verification Table

Microcontroller	Requirements	Verification
	1. The controller must support SPI and UART	1. Send a signal over SPI and UART and verify the output is the same as the signal transmitted
	2. The controller must have 256kb of onboard memory	Store 256kb of data and transmit it, checking to see if they are the same
	3. The controller must be able to transmit > 2Mbps	Transmit 20Mbps over SPI and verify the signal arrived in ≤10s

Microphone Array	Requirements	Verification
	1. A sample rate of at least 48kHz to allow transmitting in the inaudible range	Play a sound at 24kHz and check the signal outputted from the ADC to verify it is the same
	2. The MEMS Microphones must be able to pick up frequencies of at least 24kHz	Sweep the frequency from 100Hz to 24kHz and check the output from the ADC

Speaker	Requirements	Verification
	1. The speaker must be able to play frequencies at least up to 24kHz	Sweep the frequency up to 24kHz and record the sound played, then check if the recorded signal matches the original

Wireless Transmission	Requirements	Verification
	1. The wireless transmitter must be able to transmit over 2.4GHz channel losslessly at a rate of at least 17 Mpbs	Transmit a 2 Mb signal through the component over 2.4Ghz channel MAC protocol and verify it matches the original with a transmit time of 1 second
	2. The range must be at least 100ft	Perform the above transmission beginning next to the component then at 5ft increments until 100ft is reached

Power Supply	Requirements	Verification
	1. The AC/DC converter must handle converting from 120V, 60Hz to 5V DC	Supply a 120V, 60Hz AC voltage and measure the output to verify a consistent 5V DC
	2. The AC/DC converter must put out a minimum current of 2.5 A	Measure the output with an Ammeter to verify that it produces a stable 2.5A
	3. The voltage regulator must be able to supply a voltage of 3.3V with error of less than 5%	Measure the output of the Buck converter and verify the voltage stays within +/- 0.165V of 3.3V

Server	Requirements	Verification
	1. The server must be able to receive and transmit data wirelessly to the device	Receive a signal from the device, check the integrity, then transmit the same signal back and check if it is the same using the MCU

Software	Requirements	Verification
	1. The code must be able to: Perform DFTs and IDFTs	Perform DFTs and Inverse DFTs of known signals and verify the results
	2. The software code must be efficient enough to perform the calculations in under 7 seconds	Time how long it takes for the software to run with inputs of the same size as the device will send, and verify it is less than 7s
	3. The CIR Calibration must be able to remove the peaks in the CIRs due to the room environment	Measure the distance to the nearest reflective surfaces in the room and calculate their location in the CIR Array, to verify they have been removed.
	4. The 3D vector calculation must be accurate to within 3 cm at 2 m	Measure the location of the hand relative to the device, and repeat for 10 different locations, each within 2m of the device.

	5. The algorithm must be able to identify at least 3 different types of gestures with ≥ 95% accuracy	Perform each gesture at least 20 times, and record the number of times it is identified correctly.

Appendix B

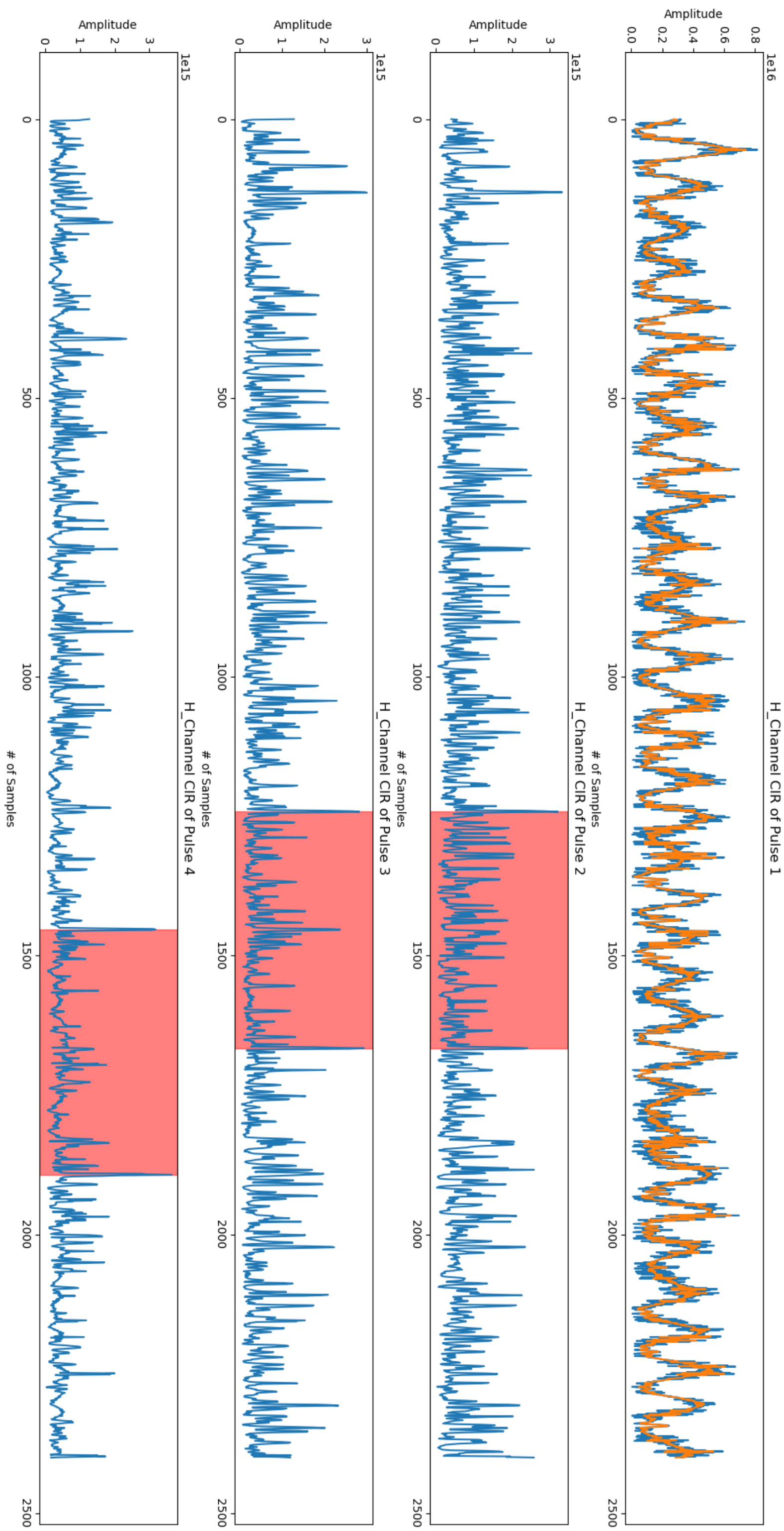

Figure 12. Horizontal Motion

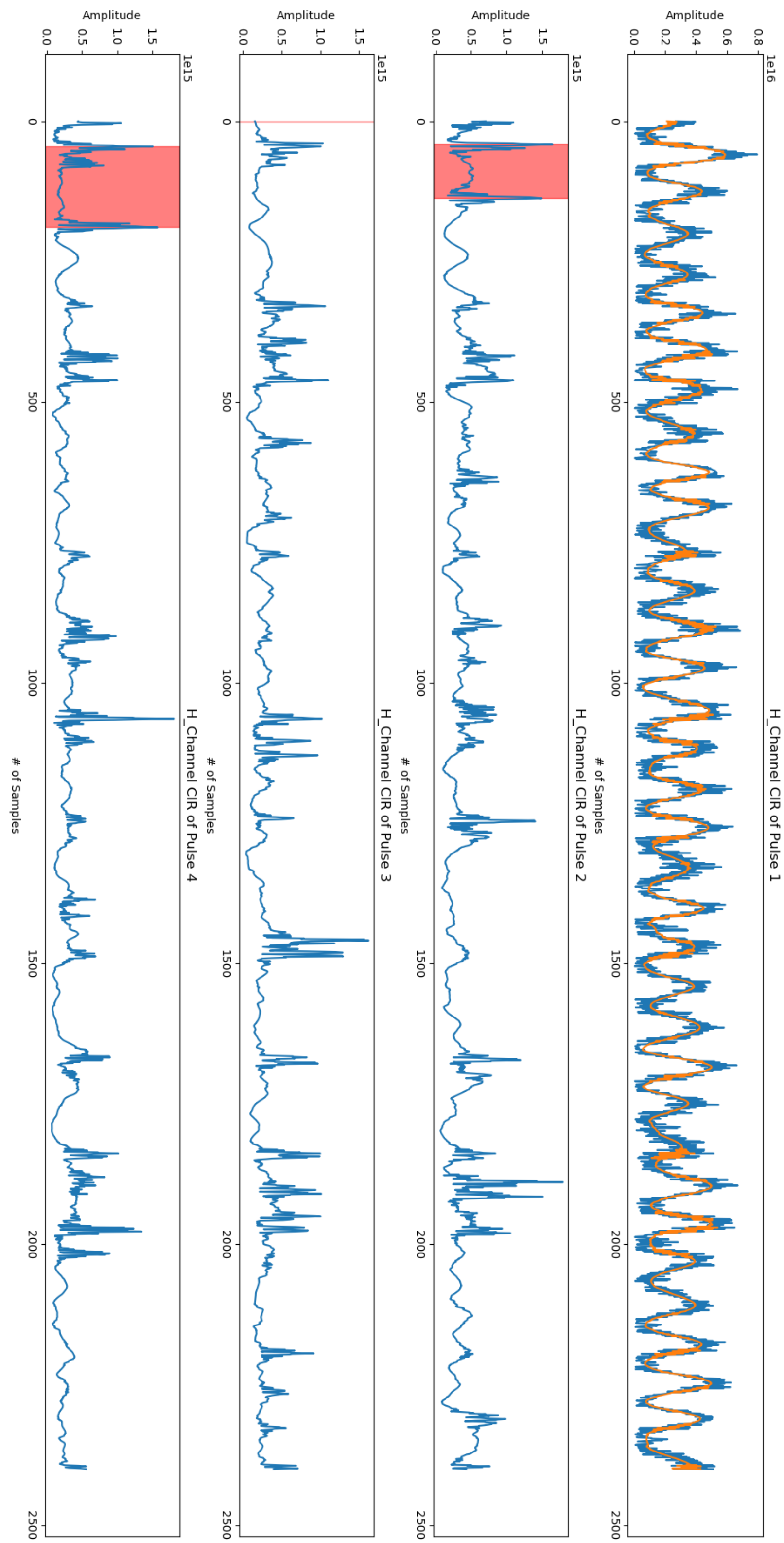

Figure 13. Vertical Motion

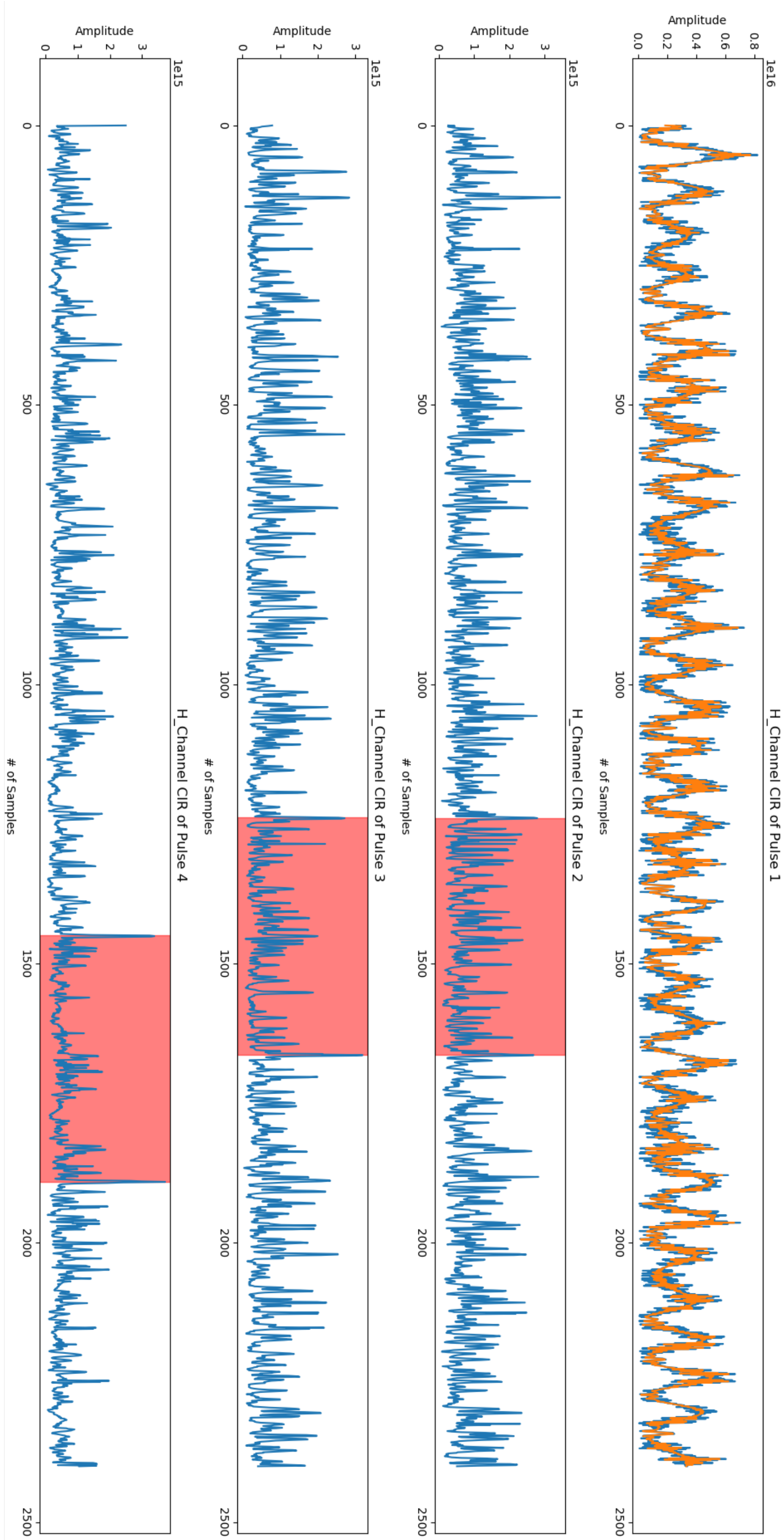

Figure 14. Horizontal Motion

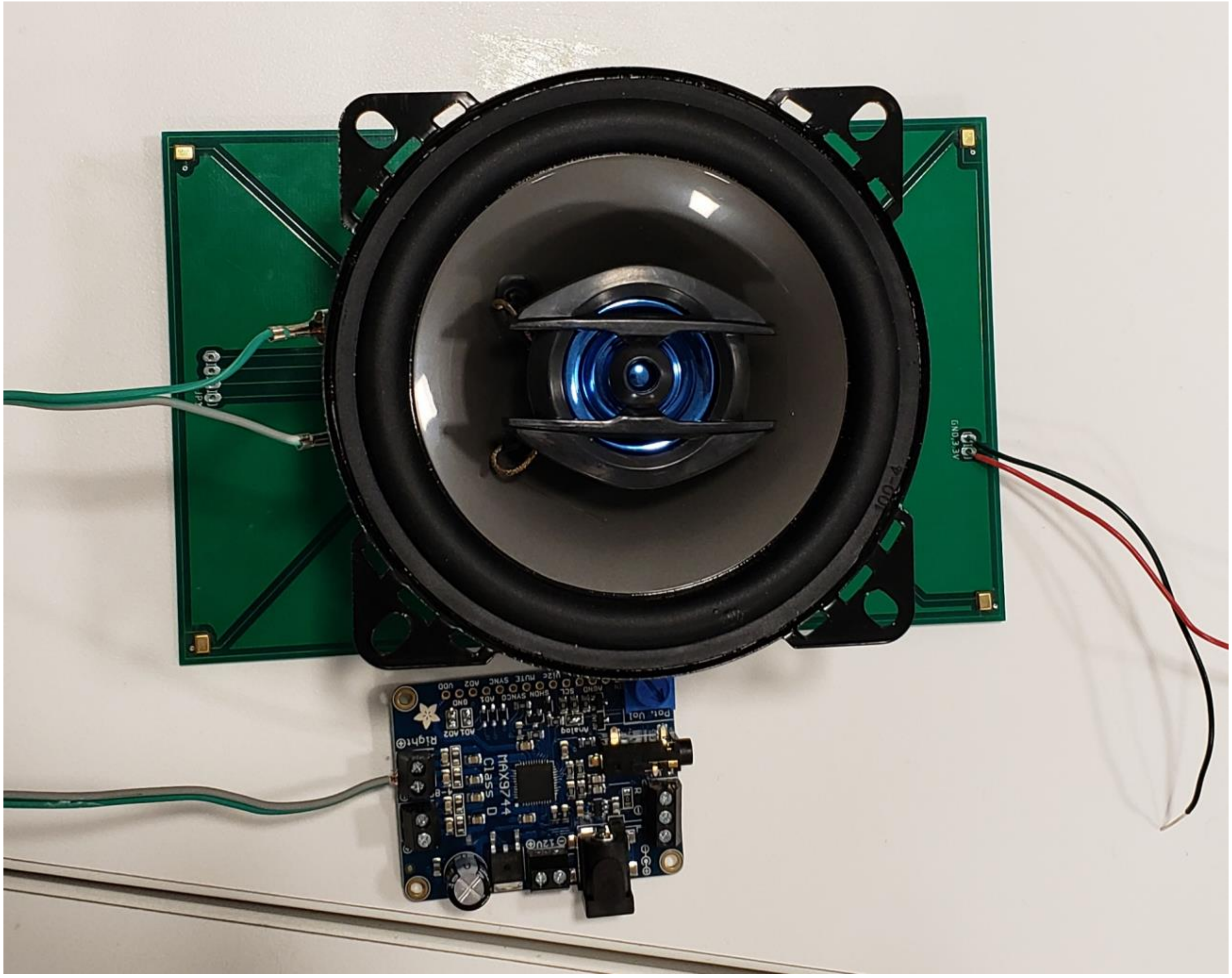

Figure 15. Speaker and Microphone Array