Multi-Microphone Array
Final Report

Team Members:
Kenneth Zhang
Sida Xiong
Thomas Kao

Teaching Assistant:
Michael Fatina

ECE 445, Senior Design
Project No. 87

May 3, 2017

Abstract

Acoustic scene analysis is a discipline that aims to process and interpret, from different perspectives, the acoustic information diffused in the environment. The primary method for obtaining this environmental sound data is to use a microphone array. The idea of large scale microphone arrays have existed since 1994. H.F Silverman of Brown Univ. and his team designed a 512-microphone array system but faced several limitations in portability and scalability, due to the hardware at that time. In this document, we present the design for a scalable and portable multi-microphone array. The central hub communicates with multiple microphone arrays, synchronizing and storing the collected sound data onto external storage. This document provides details for both the hardware and software design of the system and verification of finished modules.

Table of Contents

1 Introduction

1.1 Purpose

In 1994, scientists at Rutgers and Brown set out to build a versatile research platform for experimenting with a large microphone array. They were able to put together an array of 512 microphones, but due to the available technology, the project required an excessive amount of hardware, rendering it difficult to setup and use [1]. With the introduction of MEMS microphones and other advancements in hardware technology, recent systems have been able to improve their microphone density, detection sensitivity, and processing capabilities. Most notably, an array of 4096 microphones was built in 2014, by an Norwegian engineering firm, Sorama [2].

Our goal for this project was to create a system that could be used in a variety of environments while still providing fine detection capabilities. To accomplish this, we chose a modular design with individual microphone array modules that could be connected to the central hub in different ways based on the user's design choices. The ability to scale and reconfigure the system gives the user flexibility in where they can implement it. Our intent was to have the system take in the sound data from the array modules and save that data to a WAV file for future processing. To accomplish this, the system would consist of microphone array modules, which would serve as the front-end detection stage, and a central processing hub, which would synchronize the collection of sound data and convert it into the WAV file format.

1.2 Objectives

1.2.1 Goals and Benefits

- Easily reconfigurable parameters in software
- Portable and low power
- Expandable to fit application requirements

1.2.2 Functions and Features

- Microphone array modules that can be added to the system and rearranged based on application
- Processing hub that synchronizes the sound data and stores sound sampling into external storage

2 Hardware Design

2.1 Block Diagram

Our design consists of three main modules, as seen in Figure 1: the power supply, microphone array modules, and central processing hub. The power system supplies a steady voltage of 3.3 V to the entire system. The microphone array module consists of 4 analog MEMs microphones, 2 ADCs, and a micro-controller, as seen in Figure 2. These array modules connect to the central processing hub, which consists of one micro-controller and a USB drive, as seen in Figure 3.

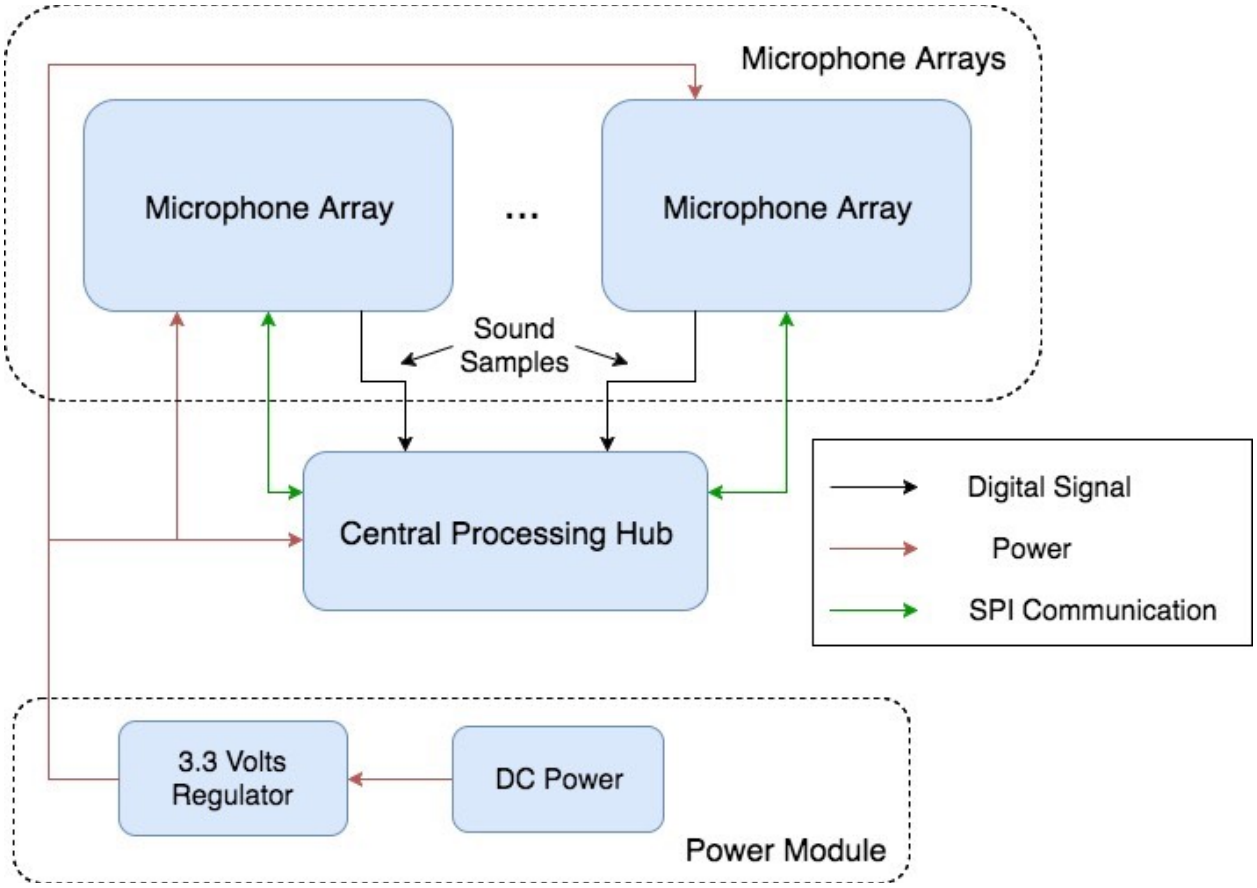

Figure 1: High level diagram of entire system

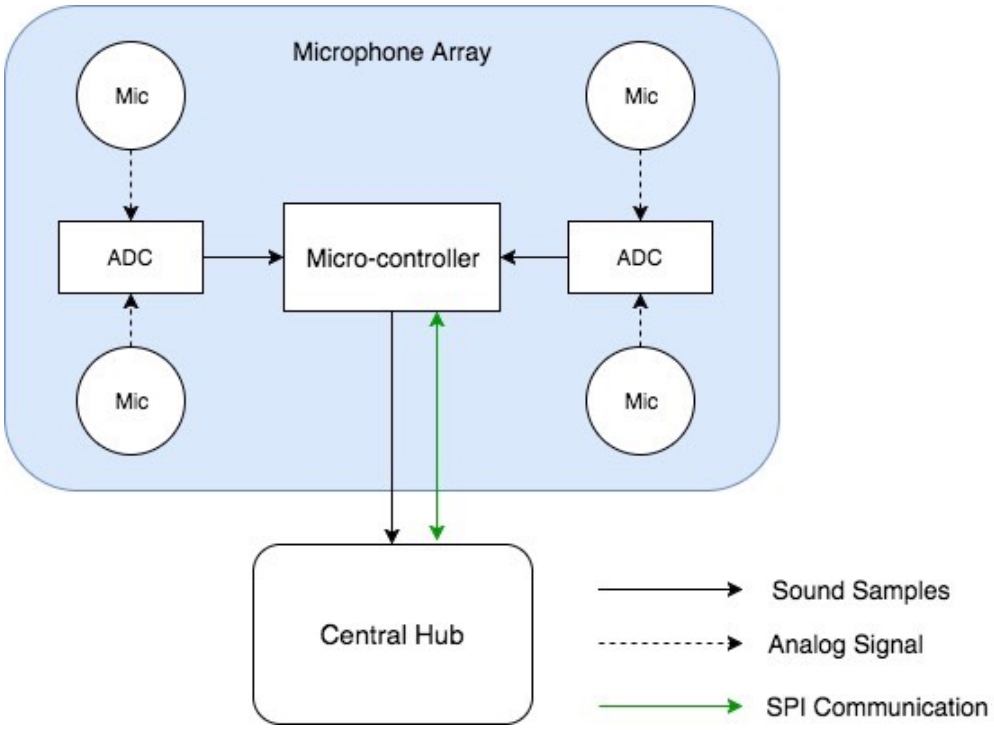

Figure 2: Block diagram of microphone array module

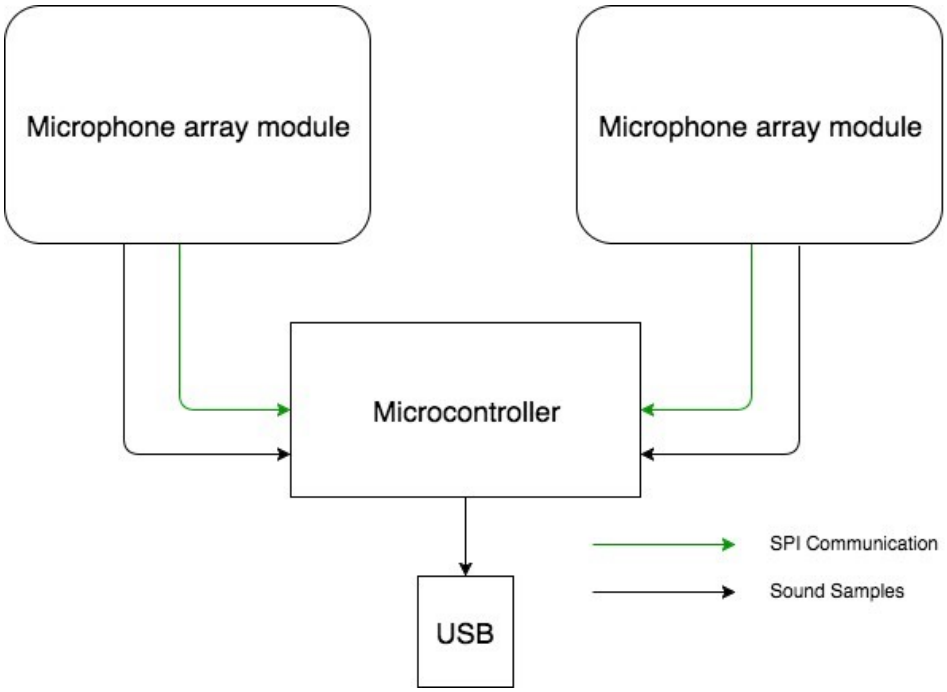

Figure 3: Block diagram of processing hub

2.2 Power System

2.2.1 Voltage regulator

In order to ensure a steady input voltage of 3.3 V to the system, we designed and built our own linear voltage regulator. Initially, we planned on using a buck converter, since it is more efficient than a linear regulator, which dissipates extra power as heat. However, we decided that the efficiency wasn't a huge concern for our application and we chose the linear regulator. The circuit consists of a MOSFET, a 2.8 V zener diode, and several resistors. With this voltage regulator, we were able to handle input voltages of 4 V - 6 V and output 3.3 V with deviations of up to .1 V. The current drawn from our voltage regulator reached a maximum of .3 A. The design for the voltage regulator is shown in Figure 4.

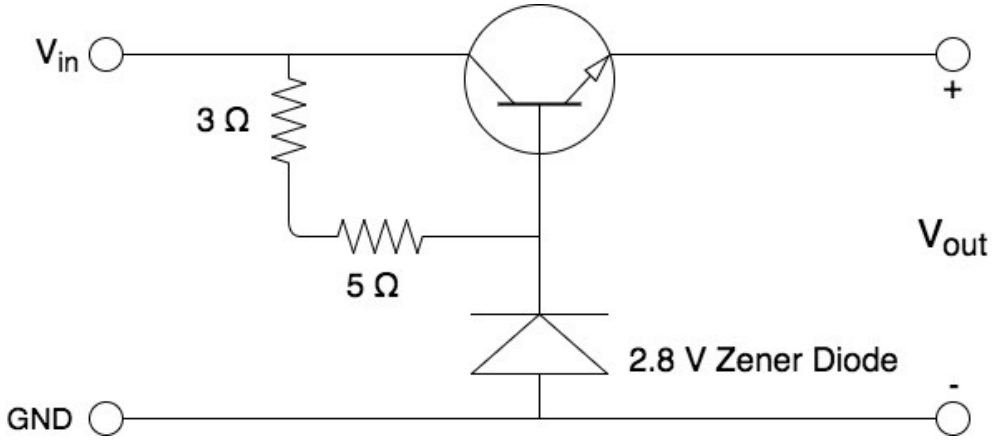

Figure 4: Voltage regulator schematic

2.2.2 Battery

Given our available input range for the voltage regulator, we decided to use standard 1.5 V AAA batteries. We connected them in series to produce a voltage of 4.5 V and then used that as the input to our voltage regulator. We decided against using a lithium ion battery because of the inherent safety hazards, such as the flammable electrolyte. AAA batteries were safer and suited our needs just as well as lithium ion batteries.

2.3. Microphone Array Module

For the microphone array module, we had several design decisions to make. The first consideration was how many microphones to put in each array. With more microphones placed in the array, the array can have more sensitive detection capabilities. This is because when microphones are placed closer together, they can detect shorter wavelengths. Ultimately, due to constraints from the parts that we chose for the module, we decided to have 4 microphones per array, which is enough for our applications. As seen from (1), where V is the speed of sound, f is the frequency of human speech, and

d is the distance between microphones. We set V to 340.29 m/s and f to 255 Hz, which is around the upper average frequency of human speech, which results in d comes out to 1.3344 m. Our microphones are placed much closer than this, allowing for even higher frequencies to be captured. With four microphones, we decided that the best configuration is a square. In this way, the array can detect symmetrically from every direction in a 2D plane.

$$d = \frac{V}{f} \tag{1}$$

2.3.1 Analog MEMS Microphones

We chose to use analog microphones because they are available in smaller packages than digital microphones. Size is such an important consideration for our microphone array modules, so we decided that using analog microphones would give us more flexibility with the design choices.

For our array, we decided to use the Invensense ICS-40720 analog MEMS microphones. These microphones have a frequency response from 75 Hz to 20 kHz, which encompasses 99.7% of the frequencies of the human voice. In addition, they are available in small 4.0 mm x 3.0 mm x 1.2 mm packages, which allows us to scale down the modules to a very small size [3].

2.3.2 Analog - Digital Converter (ADC)

Because we decided to use analog MEMS microphones, we needed to choose ADCs as well. There were many ADCs with advanced DSP functionalities built in, but they required much more configuration. To reduce the complexity, we chose to use the TI PCM 1804 ADC. This ADC is simple to configure and has the exact functionality that we needed, which made it an appealing option for us [4].

2.3.3 Array Micro-controller

When designing the system, we considered having more DSP capabilities so that we could have greater on-board processing power. However, we realized that given our application, we would only need a basic micro-controller with I^2S and SPI capabilities. On our array module, we would use the micro-controller to sample from multiple microphones/ADCs and then communicate with the micro-controller on the central processing hub. The ARM Cortex-M4 STM32F411CEU6 chip has all of these features and was a more affordable option compared to many other micro-controllers.

2.4 Processing Hub

2.4.1 Central Micro-controller

Considering that expandability of our microphone array is one of its main features, we needed to pick a micro-controller that has the flexibility to communicate with multiple micro-controllers that would be onboard the microphone arrays. With expandability in

mind, this micro-controller also needs to be able to synchronize and pull sound data from the multiple microphone arrays it is supporting, while also consuming low power as we also want the system to be portable so ideally does not require a huge external power supply.

Like the micro-controller we used for the microphone array, we decided to go with the ARM Cortex-M4 STM32F411CEU6 chip. This micro-controller has up to 5 SPIs in master and slave mode in full-duplex communication modes and 81 input/output (I/O) pins with interrupt capability. There are also 11 timers: up to six 16-bit, two 32-bit timers which would allow the central hub to keep track of the different sampling frequencies of the different microphone arrays [5]. Because synchronization is essential for the functionality of a large microphone array, we are limit by the number of timers that can track the different sampling frequencies of each array module. Thus limiting the central hub to communicate with up to 11 slave microphone arrays. As a result this micro-controller will be able to keep track of the handle our synchronization algorithm, which will be discussed in detail in the Section 3.

3 Software Design

3.1 Sampling Frequency

The sampling frequency depends what application we are sampling the sound for and the overall file size of the sound sample. Due to the importance scalability, we do not want to generate sound files that are sampled at higher frequencies than necessary. This issue can cause microphone arrays to become unsynchronized if they take too long sending the sampled sound files back to the central micro-controller hub.

Because the maximum frequency that is audible by humans is 20 kHz, the sampling frequency is set to 40 kHz to avoid aliasing based on the Nyquist sampling theorem [6]. The sampling frequency is defined in software, making it easily adjustable depending on the target sound source and application that needs to be ran on the collected sample.

3.2 Synchronization Algorithm

There were many important factors to consider when designing the synchronization for the microphone arrays, mainly how quickly can the sampling clocks synchronized and how accurately can we get to within a 10% margin of error with respect to the master clock. We also did not want the algorithm to increase the software complexity as well as consume unnecessary hardware resources. Therefore, we decided on using a Proportional Integral Derivative (PID) controller to handle the sampling clock synchronization. A PID controller by characteristic has low complexity, yet has the ability to automatically synchronize the sampling clocks via a feedback system. In theory, the proposed synchronization algorithm should effectively handle any major sampling clock offsets without hindering the system performance.

For our synchronization algorithm, we decided to use a serial peripheral interface (SPI) as our communication protocol between the central micro-controller hub and the micro-controllers on the microphone arrays. We also decided to just use the proportional control, Kp, to keep the simplicity of our software implementation. To test the performance of our algorithm, we emulated the sampling rate of the microphone arrays by having the micro-controllers hold counters which represented sound samples given a set sampling rate/frequency. The central micro-controller acted as the master and held the absolute count and the clock that the slave micro-controllers synchronized with. After the SPI communication is initialized, master controller sends an interrupt to slave controllers to begin counting. The master controller starts counting right after sending the interrupt, and continues counting for a set period after which the master controller stops counting and sends another interrupt to the slave controllers to stop counting as well. The slave controllers then sends over their counts to the master controller which are then compared to the absolute count. The master controller compute the error and if it was within 10%, which is the limit of noticeable difference given two sound samples,

the master will keep the current sampling rate. If the error is greater than 10%, the master multiples the error by a proportional value, Kp, as shown in (2).

$$y(t) = Kp \times e(t) \tag{2}$$

We defined Kp to be 0.1 because it wasn't large enough that a large error would cause the an overshoot and a smaller error would still allow a noticeable adjustment in performance. The master controller then adjusts the counting period with respect to the output of the PID controller. In theory with multiple microphone arrays, the master controller will sent interrupts to the slave controllers based on their adjusted sampling frequency and storing different absolute counts for each slave controller. This in turn synchronizes the sampling rate slave controllers with the clock on the master controller. A flowchart of our synchronization algorithm can be seen in Figure 3 [7].

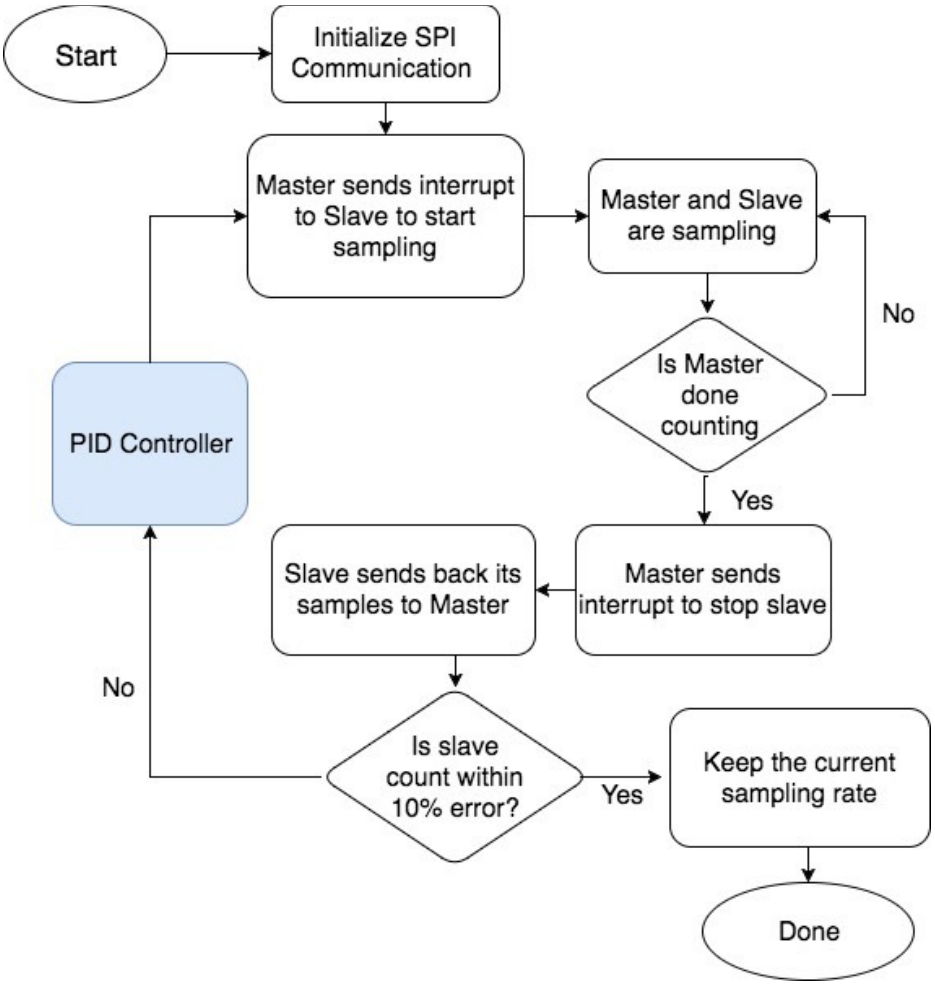

Figure 3: PID Synchronization Algorithm

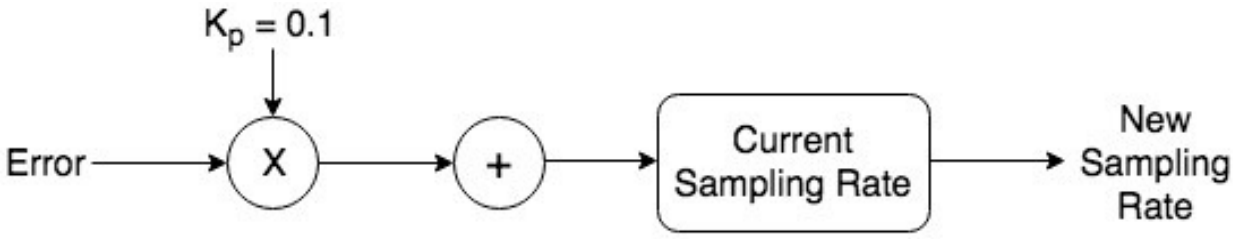

Figure 4: PID Calculation

4 Design Challenges

4.1 Industrial Micro-controller

During our design process, we considered many micro-controllers with the features that we needed for our application. We concerned ourselves more with how we would optimize the final product, as opposed to the steps we would need to take to arrive at that product. Due to this, we selected a more advanced and less user friendly micro-controller to use. Our group was inexperienced with micro-controller programming, so this greatly impeded our overall progress.

A better approach would have been to start with a more intuitive micro-controller, such as one from the Atmel AVR family. After building a simple prototype, it would have been easier to scale up, in terms of programming a more complex micro-controller and also adding more arrays.

4.2 Miniature Parts to Assemble

Although having small microphone packages was a design goal that we had, it also hindered our progress. Due to the size of the package, we weren't able to use the solder coil and solder iron to mount the package on the PCB. Instead, we were forced to buy solder paste, which was difficult to use.

Instead, we should have bought digital MEMS microphones in order to test our project. Because we were set on building the project with analog MEMS microphones and ADC's, we were blocked until those were ready. Similar to the micro-controller challenges, we should have taken more incremental steps, such as buying digital MEMS microphones, instead of trying to build the final product right from the beginning.

4.3 Properly Defining the Project Scope

One significant problem that we had was that throughout the duration of the project, our goals for the final product were constantly changing. Initially, we had wanted to have more on-board processing capabilities and we were going to have a peer-to-peer network to accomplish this. We wanted to be able to perform some real-time processing on the data before storing it. However, after discussing with our sponsor, we realized that these features weren't necessary. These constant pivots made it difficult for us to decide on the hardware that we wanted, which slowed down the progress that we could have made.

A better plan would have been to start with small, realistic goals and then adjust the goals as they are met. Instead, we focused more on the final product and not the steps that we would take to get there.

5 Verifications

5.1 Voltage Regulator

We verified our voltage regulator by probing the voltage output using a voltmeter. We supplied the circuit with 4 V up to 6 V using the DC power supply to emulate the voltage output from the AAA batteries. We tested our voltage regulator by supplying 4 V, 5 V and 6V and was about to read outputs of 3.28 V, 3.34 V, and 3.43 V respectively. We supplied the voltage for a minute for each input and the regulator was able to maintain the stated output voltages consistently; whereas, when we tested the design for our initial buck converter, the voltage oscillated too vastly to have a consistent output.

5.2 Microphone Array

Although we only built a prototype of our microphone array on a development board, we were still able to verify several aspects of that module. The microphone array should have been able to read in data from each of its ADC's at the proper sampling rate and then communicate that data to the processing hub. We simulated the ADC output using a waveform generator with a two square waves and found that our micro-controller was able to sample the data properly from both waves, which we verified by outputting the saved WAV files. We tested it further by modulating the frequency of the input waves and the output from our micro-controller responded accordingly.

5.3 Software Requirements

The save functionality of the main micro-controller hub was verified by using the onboard microphone from our development board to record audio and write that audio as a WAV file onto an USB drive. We spoke into the onboard microphone at a fixed distance each time and listened to the saved WAV file to make sure that speech was clear and there was no noticeable distortion in the audio during the writing process.

6 Cost Analysis

6.1 Labor

Name	Hourly Rate	Total Hours Invested	Total Cost = Hourly Rate x Total Hours Invested x 2.5
Kenneth Zhang	$30	10	$300.00
Sida Xiong	$30	450	$33750.00
Thomas Kao	$30	450	$33750.00
Total		**900**	**$67800.00**

6.2 Parts

Part	Part Number	Unit Cost	Quantity	Total Cost
ARM Micro-controller	STM32F411CEU6	$6.22	10	$62.22
Development Board	STM32F411-DISCO	$15.95	1	$15.96
Analog Microphone	InverSense ICS-40720	$3.38	10	$33.98
ADC	TI PCM1804	$5.20	10	$52.07
1.5 V AAA Battery	Energizer EN92	$0.73	8	$5.82
0.1 uF Capacitors		$0.24	25	$6.00
2.8 V 0.5 W Zener Diode		$0.70	1	$0.70
BJT Transistor	2N4921G	$0.60	1	$0.60
3 Ohms Resistor		$2.36	1	$2.36
5 Ohms Resistor		$2.58	1	$2.58
Total				**$182.29**

6.3 Grand Total

Parts	Labor	Grand Total
$182.29	$67800.00	$67982.29

7 Conclusion

7.1 Accomplishments

We were able to design a voltage regulator that output 3.3V within a 5% margin of error. Our microphone array was mainly build on the development board that we purchased. We were able to sample from two sources using a Waveform Generator, saving the samples into two WAV files. A change in frequency of the generated wave matched a change in pitch of the resulting sound sample which matches the change in frequency of a human as they talk. During our demo, we were able to present our voltage regulator working for a given voltage range.

7.2 Uncertainties

While we were mainly focusing on debugging our code for the micro-controller on the microphone array, we are currently unsure of how quickly and accurately our synchronization algorithm will perform. A large scale expandable microphone array relies heavily on synchronizing the sampling frequency of all the arrays quickly and as accurately as possible, otherwise producing incoherent sound samples.

7.3 Ethics

Our project follows the IEEE Code of Ethics with the following [10]:

3. “To be honest and realistic in stating claims or estimates based on available data.”
All the accomplishment stated above are based solely on results produced from our verifiable result and have been disclosed in this document.

5. “To improve the understanding of technology; its appropriate application, and potential consequences.”
The goal of our product is the provide a more modular and portable microphone array that can provide sound samples for a variety of applications, with parameters fine-tuned in software.

6. “To maintain and improve our technical competence and to undertake technological tasks for others only if qualified by training or experience, or after full disclosure of pertinent limitations”.
Design for our product has been undertaken only after we had a complete understanding of the details concerning the the goals and limitations.

7. “To seek, accept, and offer honest criticism of technical work, to acknowledge and correct errors, and to credit properly the contributions of others”.
This project has been reviewed by our Teaching Assistants, Professor, and our peers from the Senior Design class. All resources used and results outlined in this document are cited.

7.4 Future Work

In the future, we would like to integrate and test our sampling software with purchased digital microphones in order to verify that we are able to sample actual human speech. After that, we would like to implement the system on our own hardware (e.g. the parts we bought, and the PCBs we designed).

While the voltage regulator correctly outputs the defined voltage, we would like to connect our batteries to the voltage regulator to test the functionality of the entire power module. In addition, we still need to integrate the voltage regulator with the entire system to make adjustments to supply 3.3 V to all parts of the system.

We would like to test our implementation of our communication protocol as well. First, it would be beneficial to simulate the synchronization of sampling frequencies to see if it converges and how the parameters should be set in (2) for it to converge quickly. After that, we will test our implementation with two micro-controllers. Finally, we want to test the synchronization between the central processing hub and multiple microphone arrays.

Since the goal of this project was to build a scalable, modular design for a microphone array, our final objective is to be able to scale up our system to include many microphone arrays. Ideally, we would like to be able to handle thousands of microphones with our system.

References

[1] ieeexplore.ieee.org, “The huge microphone array”, 2017. [Online]. Available: http://ieeexplore.ieee.org/document/736423/

[2] conforg.fr, “Transient acoustic analysis of a run-up of a car using a modular 4096 channel MEMS microphone array”, 2017. [Online]. Available: http://www.conforg.fr/euronoise2015/proceedings/data/articles/000186.pdf

[3] invensense.com, “ICS-40720 Datasheet”, 2017. [Online]. Available: https://www.invensense.com/products/analog/ics-40720/

[4] ti.com, “Full Differential Analog Input 24-bit, 192-kHz Stereo A/D Converter”, 2007. [Online]. Available: http://www.ti.com/lit/ds/symlink/pcm1804.pdf

[5] st.com, “STM32F411xC, STM32F411xE Datasheet”, 2017. [Online]. Available: http://www.st.com/content/ccc/resource/technical/document/datasheet/b3/a5/46/3b/b4/e5/4c/85/DM00115249.pdf/files/DM00115249.pdf/jcr:content/translations/en.DM00115249.pdf

[6] berkeley.edu, “Aliasing”, 2017. [Online]. Available: http://redwood.berkeley.edu/bruno/npb261/aliasing.pdf

[7] ieeexplore.ieee.org, "Sampling Clock Synchronization with PID controller for optical OFDM systems”, 2017. [Online]. Available: http://ieeexplore.ieee.org/document/6737858/

[8] ti.com, “PCM1804”, 2017. [Online]. Available: http://www.ti.com/product/PCM1804

[9] utexas.edu, “Understanding PDM Digital Audio”, 2017. [Online]. Available: http://users.ece.utexas.edu/~bevans/courses/rtdsp/lectures/10_Data_Conversion/AP_Understanding_PDM_Digital_Audio.pdf

[10] ieeexplore.ieee.org, “IEEE Code of Ethics”, 2017. [Online]. Available: http://www.ieee.org/about/corporate/governance/p7-8.html

Appendix A: Requirements & Verification

Power Supply (10pts)

Requirements	Verifications
Voltage Source (10pts): Provides 3.3 V +/- 0.3V given a 4 - 6 V input source	Using a voltmeter to check the output of our voltage regular, verifying whether the voltage output is within 3.0V up to 3.6V.

Sensory Array Module (20pts)

Requirements	Verifications
Digital Microphones (7pts): Able to detect all frequencies of human voice (80 Hz - 20kHz)	1. Play audio recordings of known frequencies from 80Hz - 20kHz at amplitude of 70 - 75 decibels (normal human speech) 2. Check the output of the ADC using the oscilloscope to verify that the waveform generated has a frequency corresponding to the sound we played (80hz - 20kHz) and with wave amplitude 1.0V +/- 0.2V.
Microphone Array (13pts on the PCB / 10pts on the Discovery Board): The micro-controller will be able to sample from from two external I/O, generating two WAV file for each input	Test 1 (7pts): 1. Generate a 16kHz, 5Vpp wave on the waveform generator into two I/O pins on the Discovery Board 2. Output and save two WAV file onto the USB. Verifying by plugging the USB into the computer to play the sound. Test 2 (13pts): 1. Play an audio recording between 80Hz - 20kHz (normal human speech) 2. The two stereo microphones should record the sound and generate two WAV file (10pts on the Discovery Board / 13pts on the PCB) 3. Verify the two sound files by plugging the USB into the computer and playing back the sound

Processing System (20pts)

Requirements	Verifications
Saving Sound Data (10pts Max): Save sound data into WAV files into an external storage - 7pts for saving into USB on the Discovery Board / 10pts for saving onto SD on the PCB	1. Recording a sound file from the onboard microphone (for Discovery Board) or a microphone (for PCB) 2. Saving the sound file into a WAV file, verifying it by plugging the external storage into the computer and playing the audio back
Synchronizing Two Microphone Arrays (10pts): WAV files generated by the two microphone arrays are in sync	Compare the two WAV files (one file from each microphone array) using a C# script that will calculate an average per-sample difference. Every 10th sample from each file will be compared and if the difference is within a 7-10% error, it will be considered in sync.

Appendix B: Central Hub Schematic

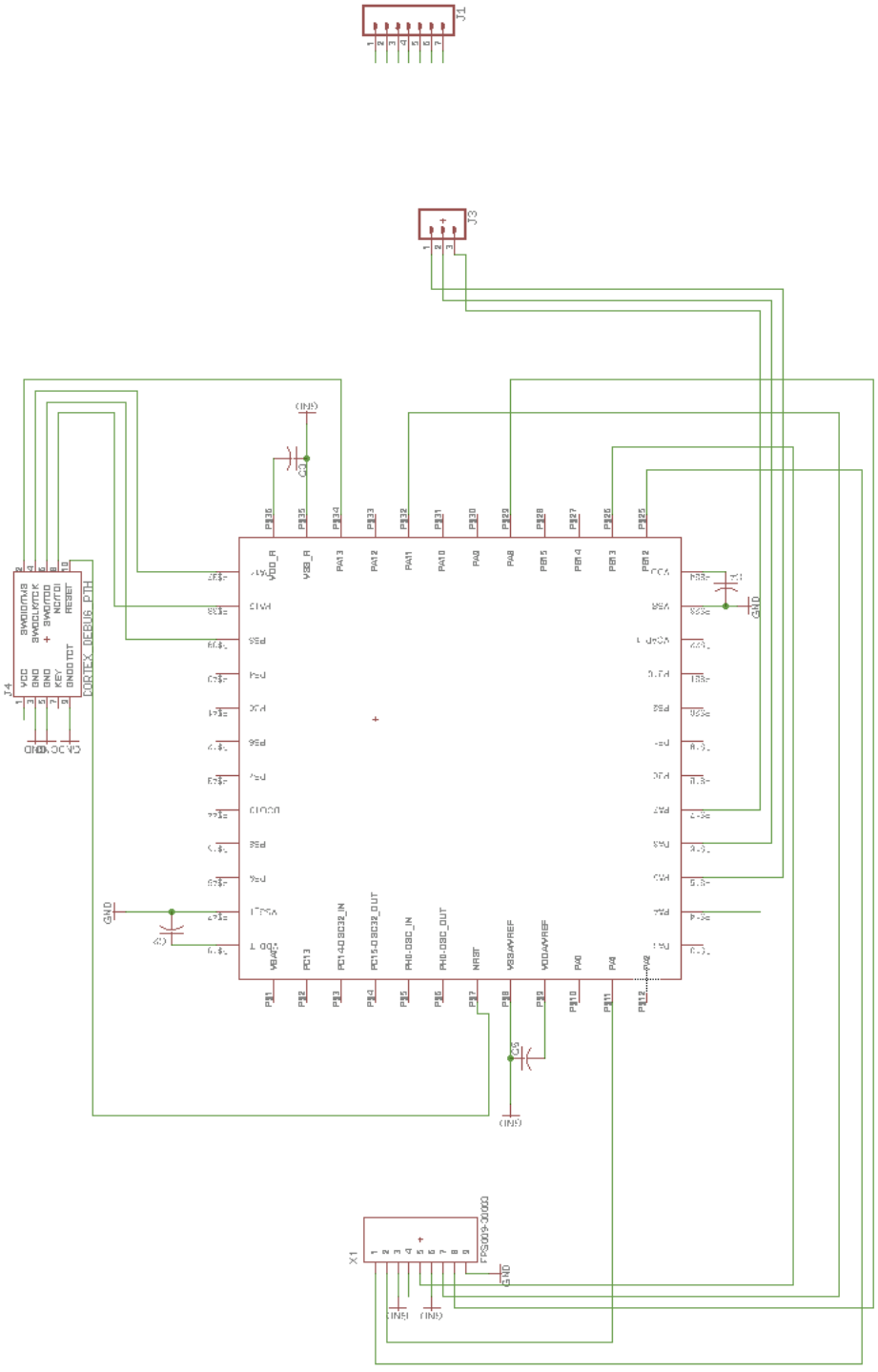

Figure 5: Circuit schematic of central processing hub

Appendix C: Microphone Array Schematic

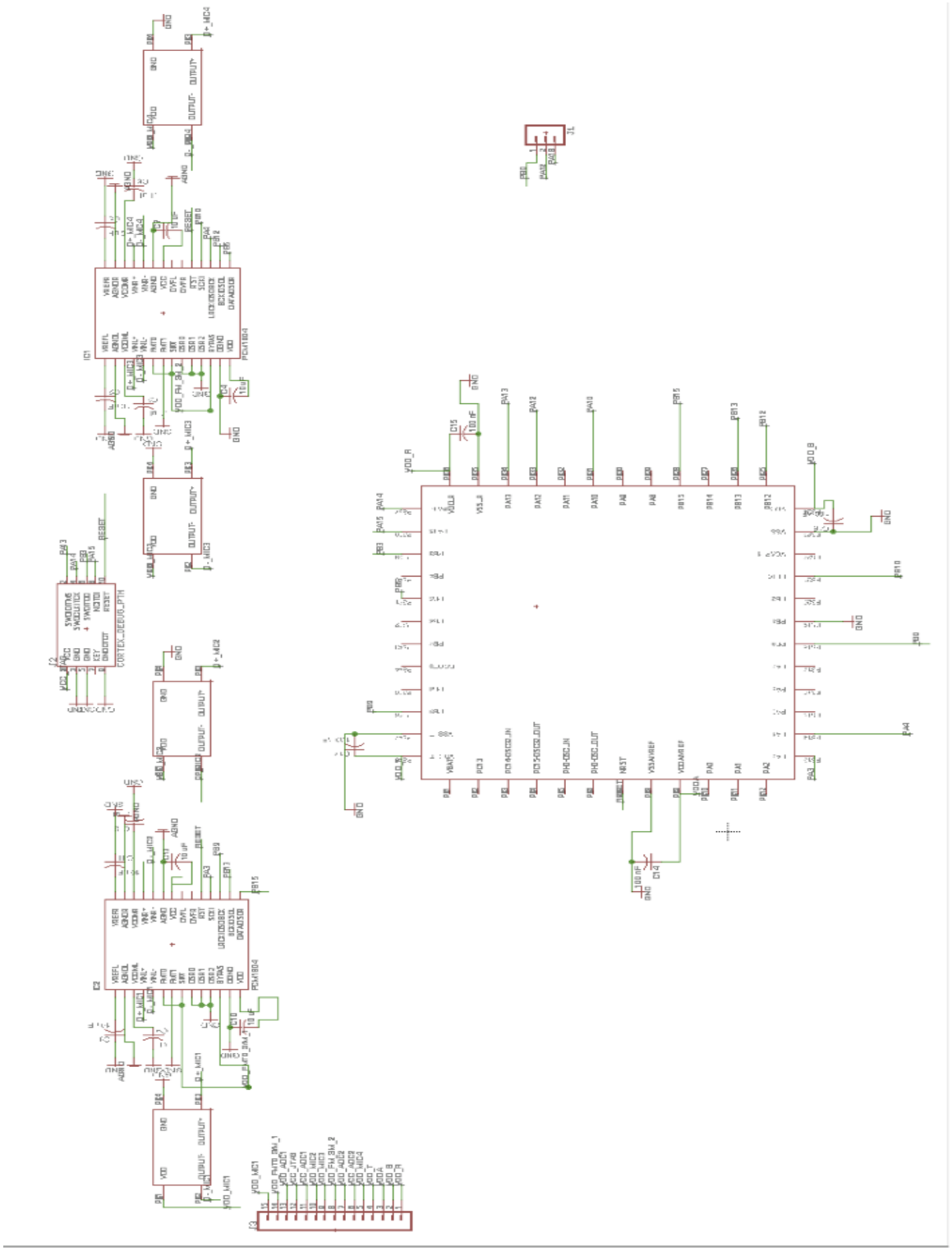

Figure 6: Circuit schematic for microphone array