Spatially Selective Audio Source Separation

Submitted To

Dr. Neal Hall

Prepared By

Blake Schwartz
Tanay Mannikar
Alex Zhang
Zachary Hestand
Ayan Basu

ECE464 Senior Design Project
Chandra Department of Electrical and Computer Engineering
University of Texas at Austin

2024

CONTENTS

TABLES……………………………………………………………………………………iii

FIGURES…………………………………………………………………………………..iv

EXECUTIVE SUMMARY……………………………………………………………...v

1.0 INTRODUCTION…………………………………………..…………………………...1

2.0 DESIGN PROBLEM……………..………………………………………………….1

3.0 DESIGN SOLUTION……………………………………………………………..2

3.1 Hardware Solution…………………………………………………………..3

3.2 Software Solution ……………………………………………………………...6

3.2.1 *Synthetic Audio Dataset*……………………………………………………6

3.2.2 *Beamspace Transform*……………………………………………………...7

3.2.3 *Convolutional Neural Network*……………………………………………9

4.0 DESIGN IMPLEMENTATION…………………………………………………10

4.1 Hardware Implementation………………………………………………………10

4.2 Software Implementation…………………………………………………..11

4.2.1 *Synthetic Audio Dataset Implementation*………………………………...12

4.2.2 *Beamspace Transform Implementation*…………………………………..12

4.2.3 *CNN Implementation*……………………………………………………...13

5.0 TEST AND EVALUATION……………………………………………………...14

5.1 Hardware Testing…………………………………………………………...14

5.2 Software Testing………………………………………………………………16

6.0 TIME AND COST CONSIDERATIONS………………………………………17

7.0 SAFETY AND ETHICAL ASPECTS OF DESIGN……………………………18

8.0 RECOMMENDATIONS…………………………………………………………18

9.0 CONCLUSIONS……………………………………………………………………19

REFERENCES……………………………………………………………………………20

APPENDIX A – TWO-CHANNEL MICROPHONE PCB SCHEMATIC………………..A-1

APPENDIX B – CLOCK DISTRIBUTION BOARD SCHEMATIC……………………...A-3

APPENDIX C – BILL OF MATERIALS…………………………………………………A-5

TABLES

1 *Deliverables Specifications*……………………………………………………………………2

FIGURES

1 *Block Diagram of the Design Structure*……………………………………………………….3

2 *MiniDSP UMA-16 v2 USB Mic Array*………………………………………………………...4

3 *Circuit Schematic for a Pair of Microphones*…………………………………………………4

4 *Two Daisy-Chained Microphone PCBs*……………………………………………………….5

5 *Completed 16-Channel Microphone Array*……………………………………………………5

6 *Diagram of Linear System Theory and Convolution*………………………………………….7

7 *Time-Frequency UCA Beamforming Results*………………………….……………………9

8 *Diagram of the CNN Architecture*…………………………………………………………...10

9 *Anechoic Recording of a Balloon Pop in Time and Frequency Domain*…………………….12

10 *Predicted and Ground Truth IRM Masks*…………………………………………………….16

11 *Training and Validation Loss from CNN Model*……………………………………………..16

12 *SDR Compared to Target and Noise Signals*………………………………………………...17

EXECUTIVE SUMMARY

The cocktail party problem states that it is difficult for both computers and people with hearing impairments to distinguish one speaker from multiple voices with overlapping time and frequency information. This design aims to solve a scaled-down version of this vast challenge by using digital signal processing and machine learning techniques. Our deliverables include a 16-channel microphone array, a synthetic audio dataset for training, and a source separation algorithm. This algorithm includes a beamspace transform and a convolutional neural network.

In the final design solution, a multichannel audio stream is sent to an STFT module, followed by a beamspace transform and a convolutional neural network. The output of the model is an ideal ratio mask that separates the target signal from the unwanted source. This solution has two main components: hardware and software. The hardware solution is a 16-channel uniform circular microphone array, which is mounted on a tripod via laser cut and 3D printed pieces. Eight PCBs with two microphones each are connected in series and controlled by two clock distribution boards. The audio is then sent to a USB streamer, which sends a Pulse Coded Modulation signal to a computer. The microphone array was used to record impulse responses in the team's target environment which were convolved with a public audio dataset to synthesize multichannel speech audio recorded in the room. The speech audio is modified with a beamspace transform. This operation decomposes the 16-channel signal matrix into a set of spatial directions as a form of plane-wave decomposition. Finally, the neural network predicts the ideal ratio mask which is applied to the optimal beamspace spectrogram.

The team discussed multiple solutions and experienced many challenges throughout the design process. It was too difficult to create an array completely from scratch. Therefore, the custom microphone array was adapted from an off-the-shelf, four-by-four rectangular array to use the pre-fabricated USB streamer and drivers. Other difficulties included a PCB manufacturing error and an issue with the clock distributing boards. The software team cycled through many iterations of the dataset, beamspace transform, and neutral network implementation. A significant portion of the project was dedicated to researching how the beamspace transform actually works and how to use TACC to train the model.

To test the microphone array, each microphone was individually tested using a Python script. The noise floor was analyzed, and it was determined that two of the microphones were outliers with an unreasonably high noise floor. After improvements, the array can record 16-channel audio at a sampling rate of 44.1 kHz. To test the convolutional neural network, multiple calculations were performed, including the Signal-to-Distortion Rate and the Hearing-Aid Speech Quality Index. Both metrics, along with the training and validation loss, show that the model can accurately predict ideal ratio masks without distorting the signals significantly.

The team encountered various time delays which inhibited the overall progress of the project. These include both shipping postponements and TACC queuing times. In contrast, there were no cost concerns, as the project was greatly under budget.

Although there is a slight ethical concern with the fact that this system could be used for spying or espionage, the system would require immense optimization to achieve such a level of source

separation. Therefore, the team is not worried about adverse usage of this design. There are also no safety concerns, as the array has no moving parts and does not emit sound or light.

In conclusion, this system does make progress toward solving the cocktail party problem by combining hardware and software elements. The design is designed to separate the target speaker from one unwanted source and additional noise. Further research includes extending the model to more scenarios and optimizing the software for real-time implementation.

1.0 INTRODUCTION

This document is an archival record of the team’s spatially selective source separation design and is intended for Dr. Neal Hall and any future researchers studying spatial audio. The following sections detail the design problem, solution, implementation, test and evaluation methods, time and cost considerations, safety and ethical aspects, and future recommendations.

The purpose of this design is to separate a target speaker from a two-speaker environment with additional noise. In doing so, the team addresses an aspect of the cocktail party problem. The design includes a custom 16-channel microphone array, a synthetic two-speaker speech audio dataset, and a source separation algorithm. After testing the solution, it was determined that the microphone array is fully functional and the algorithm can accurately remove the unwanted speaker, under certain conditions. There are many plans to continue this research, including enhancing the physical array design and training the separation model to be universal to more environments, sources, and source types.

2.0 DESIGN PROBLEM

This design aims to address the cocktail party problem, where human voices become difficult to separate in a noisy environment due to overlapping time and frequency information. Engineers have attempted to solve this effect with various acoustic, signal processing, and machine learning techniques; however, significant progress has mainly emphasized single-channel (monaural) audio. Our design addresses this by performing source separation on multichannel audio with an aspect of directionality. Table 1 details the team’s deliverables and specifications. These deliverables include a 16-channel, uniform circular microphone array, a tripod mount for the array, a synthetic audio dataset that simulates recordings from the target room, and a source separation model, which is optimized to maximize separation in the audible range, 20 Hz to 20 kHz. The scope of this project will be a two-speaker environment; this can be extended in future research.

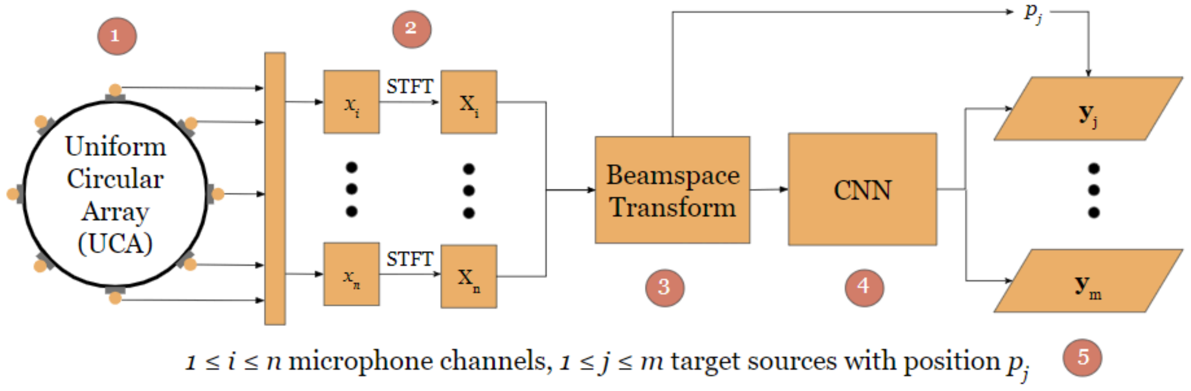
Table 1. Deliverables Specifications

Subteam	**Deliverable**	**Specification**
Hardware	Microphone array	16 channels, uniform circular geometry, capable of recording audio at 44.1 kHz
Hardware	Microphone array mount	Attachable to a tripod to simulate a user wearing the array
Software	Synthetic Audio Dataset	At least 100 hours of synthetic 44.1 kHz audio that simulates recordings in the target room
Software	Source Separation Algorithm	Separate the target speaker from the unwanted source and additional noise in a two-speaker recording. Optimized to maximize separation in the audible range, 20 Hz to 20 kHz

3.0 DESIGN SOLUTION

This section provides a detailed overview of the team's final design solution. The hardware component covers the design and construction of the microphone array. This portion will cover the materials used, discuss the inputs and outputs of the system, and describe the research used to arrive at the design decisions. The software component will cover data collection and the source separation model. This portion will explore considered data collection, and model options, the research done on each topic, and how the choices meet the specifications set above.

Figure 1 displays the high-level block diagram of the design. In the diagram, the microphone array records the speech as the signals $x_i,\ldots, x_n$ where $n = 16$. The system then performs the STFT and converts speech to the frequency domain. The signals are then passed to the harmonic-percussive separation module (HPS) and beamspace transform module (BT), encoding spatial information into the speech input. After preprocessing, the speech is inputted into a multichannel speech separation model to separate the individual sources $y_j, ..., y_m$ corresponding to the mth source where *m* is the total number of speech sources present. The separation model will generate a mask, which is then used on the beamspace input to generate the separate sources. The hardware and software components will be explained in detail in the following sections.



1
Uniform Circular Array (UCA)
2
x_i
STFT
X_i
x_n
STFT
X_n
Beamspace Transform
3
CNN
4
p_j
y_j
y_m
$1 \leq i \leq n$ microphone channels, $1 \leq j \leq m$ target sources with position p_j
5

Figure 1: Block Diagram of the design structure

3.1 Hardware Solution

In Figure 1, the microphone array module represents the hardware design for this project. The hardware design consists of a 16-channel microphone array with eight PCBs, each with two microphones, connected in series, and a USB streamer.

The final design was adapted from an off-the-shelf UMA-16 microphone array by MiniDSP, shown in *Figure 2* where the microphones were aligned in a uniform rectangular array (URA) fashion as a four-by-four matrix. However, this architecture did not allow for a cohesive collection of audio signals in surrounding areas such as to the sides of it. To fix this, we laid out the dual microphone PCBs in a circular fashion. The microphone array consists of 16 Knowles SPH1668LM4H-1 digital microphones to collect the speech audio. The microphones are implemented into a custom schematic including various other circuit components (resistors, capacitors, inductors, buffers, and voltage regulators) to manage the amount of current and voltage properly. The schematic, shown in Figure 3, describes a pair of microphones that translate the incoming speech audio into Pulse Density Modulation (PDM) signals, representing the set of sampled signals as a stream of single bits [1]. See Appendix B for a more detailed schematic of the custom PCBs created in EAGLE.

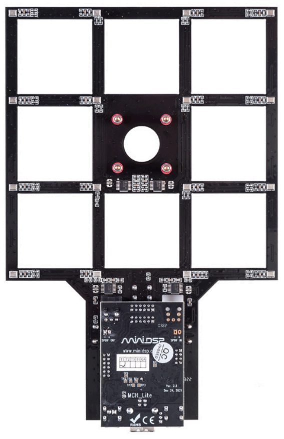


Figure 2: MiniDSP UMA-16 v2 USB Mic Array

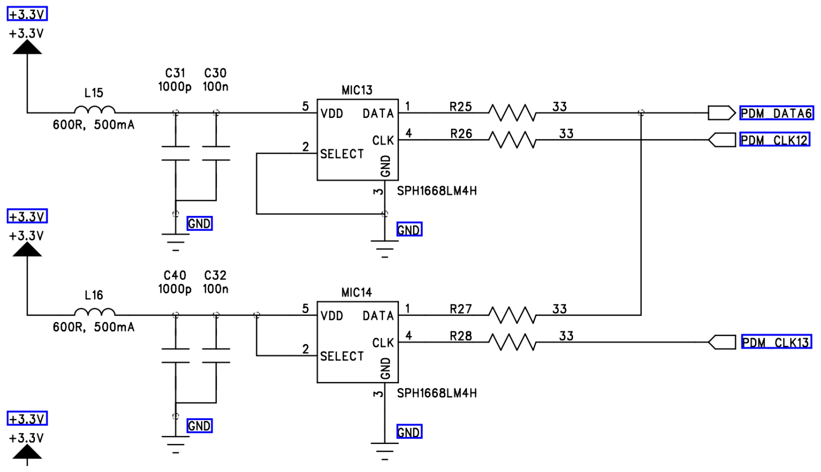


Figure 3: Circuit schematic for a pair of microphones

Eight individual PCBs, each with two microphones and eleven other surface-mount devices (SMD), are daisy-chained together, as seen in Figure 3 as well as a more detailed schematic in Appendix A, using ribbon cables to form the array. These are connected to two clock distribution boards via the same ribbon cables. Appendix B shows the schematic for the custom clock boards designed in EAGLE. This design allows us to easily experiment with different geometries or repair damaged microphones. The resulting geometry is a uniform circular array (UCA) with two rows and eight columns of microphones.

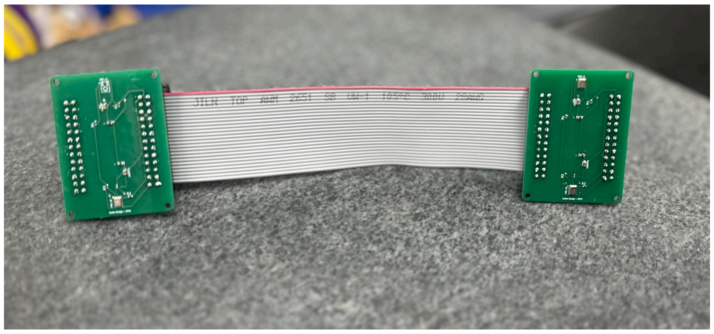


Figure 4: Two daisy-chained microphone PCBs

Next, the sampled signals are streamed into an asynchronous USB audio interface, which conducts PDM to Pulse Coded Modulation (PCM) signal conversion, allowing all 16 channels of raw audio to be recorded. This USB interface was removed from the original four-by-four array and wired to the new UCA.

Lastly, a mount was constructed for the microphone array. The base of the mount was made from laser-cut birch plywood, and the cases for the eight PCBs and two streamers were 3D printed. Figure 4 displays a daisy-chained set of microphone PCBs, while Figure 5 displays the final UCA mounted to a tripod.

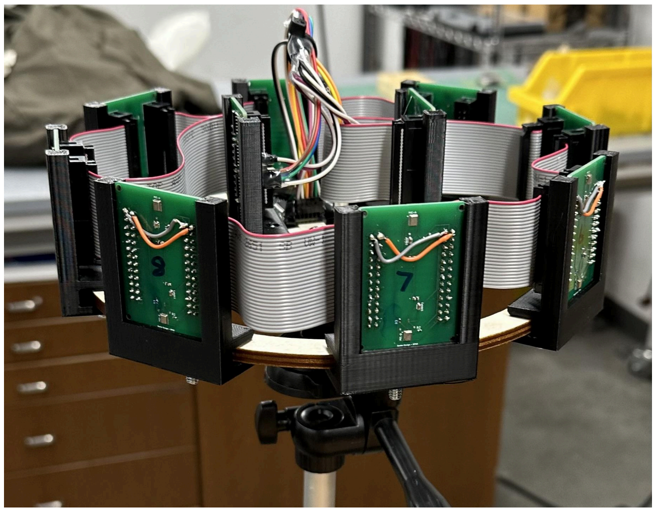


Figure 5: Completed 16-channel microphone array

3.2 Software Solution

First, the sixteen channels are passed through two preprocessing steps: an STFT module and a beamspace transform module. Then, the intermediate output is entered into the separation model and predicts a mask to be used on the beamspace transformed input.

There are three main components to the software design solution, which will be discussed in the following subsections in further detail: a synthetic audio dataset, the beamspace transform, and a convolutional neural network (CNN).

3.2.1 *Synthetic Audio Dataset*

Before training occurs, a large dataset of multichannel speech audio recorded in the target environment must be collected. However, it is not possible to record people talking in a room for a vast amount of time. Therefore, the team developed a synthetic dataset using recorded multichannel room impulse responses (RIR) at 32 locations. Next, the impulse responses were convolved with dry speech samples retrieved from the public *Expresso* dataset [2]. The dataset was downsampled to 44.1 kHz, which lowered the dataset size while still optimizing the model for frequencies between 20 Hz to 20 kHz due to the Nyquist-Shannon Sampling Theorem. This dataset was chosen as it consists of thousands of hours of high-quality, monaural speech audio. The result simulates how the raw speech would sound as if it was spoken in the target room and recorded with the microphone array. As shown in Figure 6, this method is directly derived from linear system theory, which states that an input convolved with the impulse response of the system results in the output of the system. The final dataset contains about 114 hours of multi-source, multichannel speech audio, which meets the design specification.

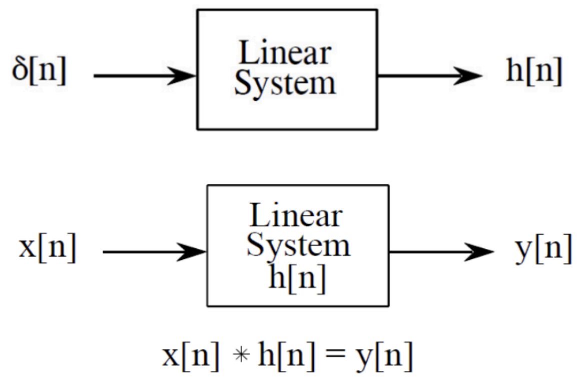


Figure 6: Diagram of Linear System Theory and Convolution

After convolution, the single-speaker synthetic audio is mixed with a second single-speaker audio clip from a different location. This mixing process is performed for every speaker in the *Expresso* dataset and for every RIR location. This audio is passed through the beamspace transform, described in the following subsection. The final training data consists of short segments of the beamspace transformed audio and the target is the ideal ratio mask (IRM), calculated using the equation,

$$\begin{aligned} IRM(t,f) &= \left(\frac{S^2(t,f)}{S^2(t,f)+N^2(t,f)}\right)^{\beta} \\ &= \left(\frac{SNR(t,f)}{SNR(t,f)+1}\right)^{\beta}, \end{aligned} \tag{1}$$

which isolates the target speaker [3].

3.2.2 *Beamspace Transform*

The beamspace transform decomposes the received 16-channel signal matrix into a set of spatial directions as a form of plane-wave decomposition by frequency. Once projected into these directions, or "beams," the received audio can be filtered by spatial location so that spectrograms corresponding to each desired separation angle in the azimuthal and elevation directions can be created for the neural network to train on. The system chosen implements a modified version of an N-beam FFT beamforming method which is implemented in the time-frequency domain based on circular convolution properties for efficient computation adapted from [4].

First, the beamforming matrix for the UCA geometry is defined. Figure 8 shows the beamforming matrix ***A*** steered towards a desired azimuthal and elevation angle at frequency $k = 2\pi f$ where N is the number of microphones in each circular array.

$$A(\phi, \theta) = \sum_{n=1}^{N} e^{-jkasin\theta cos(\phi - \frac{2\pi n}{N})} \tag{2}$$

The term in the exponent defines the values α_n needed to construct a complex weight matrix for n microphones in the UCA. In our implementation, we split both top and bottom arrays and performed this task on each before summing after beamforming has taken place. The first column of the N x N weight matrix $\boldsymbol{W}_N$ can then be used to perform circular convolution in the frequency domain with each incoming time sample of the N array elements ***x***, outputting the steered result ***y***. This is explained in the following equations:

$$\mathbf{W}_N = \begin{bmatrix} \alpha_0 & \alpha_1 & \dots & \alpha_{N-2} & \alpha_{N-1} \\ \alpha_{N-1} & \alpha_0 & \dots & \alpha_{N-3} & \alpha_{N-2} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ \alpha_1 & \alpha_2 & \dots & \alpha_{N-1} & \alpha_0 \end{bmatrix} \quad \begin{gathered} \mathcal{F}_N(\mathbf{w} \star \mathbf{x}) = \mathcal{F}_N(\mathbf{w}) \circ \mathcal{F}_N(\mathbf{x}) = \mathcal{F}_N(\mathbf{y}) \\ \mathbf{y} = \mathcal{F}_n^{-1}\{\mathcal{F}_N(\mathbf{w}) \circ \mathcal{F}_N(\mathbf{x})\} \end{gathered} \tag{3}$$

This process is applied to each individual frequency bin vector of an incoming spectrogram chunk, and the N outputs are averaged to output a spectrogram corresponding to each desired steering angle. We can precompute the beamspace weight matrix prior to real-time separation to save computing power and time. The results of the beamforming process are shown in Figure 7.

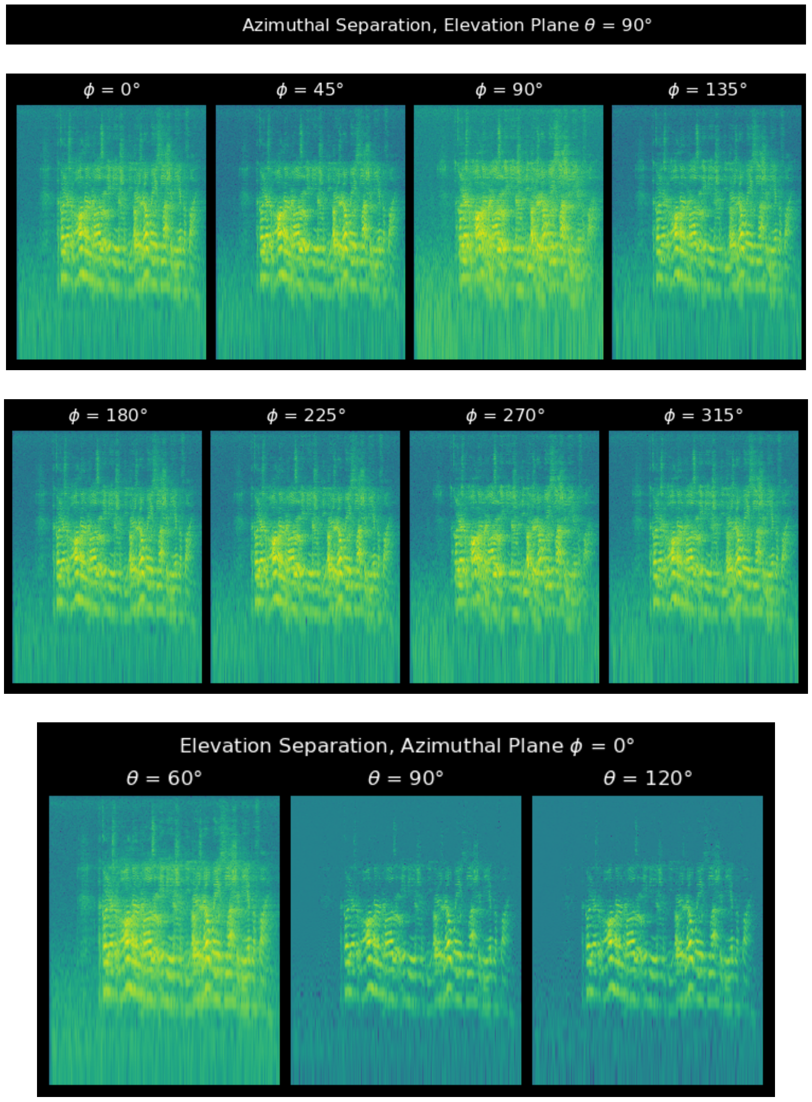


Figure 7: Time-Frequency UCA Beamforming Results

The beamspace information can be further manipulated to perform direction-of-arrival (DOA) calculations for up to N incoming sources by exploiting the UCA geometry [5]. The more advanced use of this framework is denoted by the position vector p_j in Figure 1 which the user may use to select a desired angle of arrival without knowing source locations beforehand.

3.2.3 *Convolutional Neural Network*

Next, a convolutional neural network (CNN), shown in Figure 8, consisting of three blocks of two 2D convolutional layers followed by batch normalization and a ReLU activation function, is utilized. The output is averaged, flattened, and sent to a fully connected layer also with batch normalization applied and a ReLU activation function. Lastly, the data is sent to a second fully connected layer with a sigmoid activation function producing the ideal ratio mask (IRM) $\hat{M}$, the estimated mask used to calculate $\tilde{Y} \odot \hat{M} = \hat{X}$, where $\odot$ is the Hadamard product, producing a signal $\hat{X}$, the predicted separated source signal [6]. The neural network will try to minimize the difference between the ground truth and predicted masks to recreate IRMs of speech sources accurately. Finally, the IRM is applied to the optimal beamspace transformed spectrogram to separate the target speaker from the unwanted source and any additional noise.

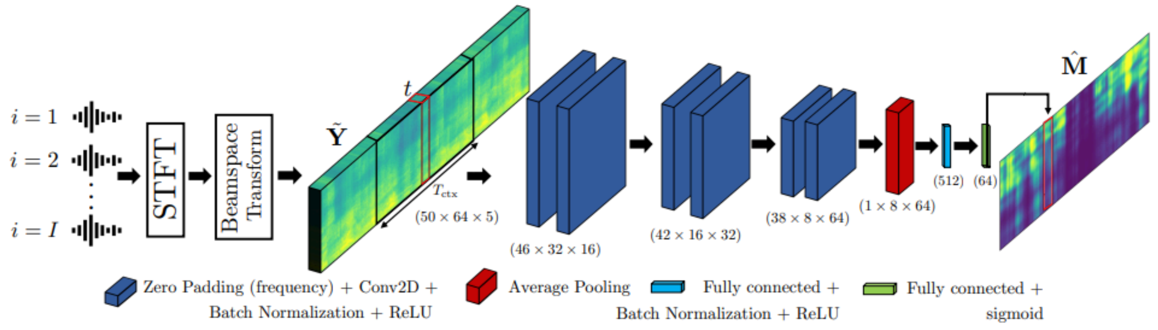

$i = 1$
$i = 2$
$i = I$
STFT
Beamspace Transform
$\tilde{\mathbf{Y}}$
t
T_{ctx}
$(50 \times 64 \times 5)$
$(46 \times 32 \times 16)$
$(42 \times 16 \times 32)$
$(38 \times 8 \times 64)$
$(1 \times 8 \times 64)$
(512)
(64)
$\hat{\mathbf{M}}$
Zero Padding (frequency) + Conv2D + Batch Normalization + ReLU
Average Pooling
Fully connected + Batch Normalization + ReLU
Fully connected + sigmoid

Figure 8: Diagram of the CNN Architecture

4.0 DESIGN IMPLEMENTATION

Throughout the design process, many challenges were encountered by both the hardware and software subteams. The following sections describe these obstacles and how the team adapted the design solution to overcome them.

4.1 Hardware Implementation

The first hurdle in the hardware design process was the construction of a multichannel microphone array. This was initially not given enough attention as it was believed that a simple microcontroller, like an Arduino with many analog inputs, would suffice in collecting audio data. However, after further research and consultation with others, it was realized that this would not easily work for our needs as we would not have an audio driver capable of reading the audio into a computer live. This led to the decision to use an off-the-shelf microphone array that includes audio drivers and adapt it to our project needs.

As talked about before, this off-the-shelf microphone array was a uniform four-by-four rectangular array. Because this would not be conducive to recording audio in 360 degrees, we needed to figure out a way to change this into a uniform circular shape. We wanted to use a similar schematic that the original array was built on but a rigid single PCB would not work. With this we had two ideas: either a single flexible PCB that could “wrap” into a circular shape with a rigid fixture to hold its shape or a chain of smaller individual PCBs mounted onto a circular shape. Ultimately we decided on the chained approach, as the cost for individual PCBs was much lower, and we had the flexibility to remove microphones or change the spacing between them as we pleased, as we could just extend the wiring between them.

Upon receiving the PCBs, a few defects were realized throughout assembly and testing that required us to change the implementation approach we had. The first issue was a mistake in the layout of the PCB boards themselves. The 3.3V input was crossed with a ground trace making a permanent short on every microphone board. Since there would not be time to order new boards, we took a scalpel and destroyed the connection between the two traces. Then, we took small hookup wires and remade the correct connections on each board. The other defect found was the clock distribution board would not send clock signals to microphones past the first eight. Since the software team needed to start recording audio the next day, the solution was to add a second clock distribution board that would power the last eight microphones separately. So, rather than a single clock board with 16 microphones chained together, the array uses two clock boards with eight microphones per board.

4.2 Software Implementation

All three aspects of the software design had revisions due to unforeseen hurdles. The synthetic audio dataset had to be created multiple times due to non-ideal conditions, the beamspace transform algorithm had many different implementations before it was finalized, and the team faced significant difficulties training the CNN.

4.2.1 *Synthetic Audio Dataset Implementation*

To create the first synthetic dataset, the team used claps as the impulse. However, it was quickly realized that this yielded inconsistent RIRs. To solve this, balloon pops were used for the second iteration. The team set a standard number of pumps, and the pop acted as the impulse.

Although this impulse is far more consistent, it is still not a perfect impulse, as seen in Figure 9. A perfect impulse in the frequency domain would show a constant magnitude at all frequencies. It is also clearly not a Dirac delta in the time domain, as there is a slight ripple after the spike. The team's solution was to normalize the RIRs using a frequency-domain deconvolution process with an anechoic recording of the balloon pop to negate modal spikes and dips in the frequency response of the imperfect balloon impulse. This recording was done in the UT Anechoic Chamber thanks to the permission of Dr. Michael Haberman, an acoustics professor in the Mechanical Engineering department.

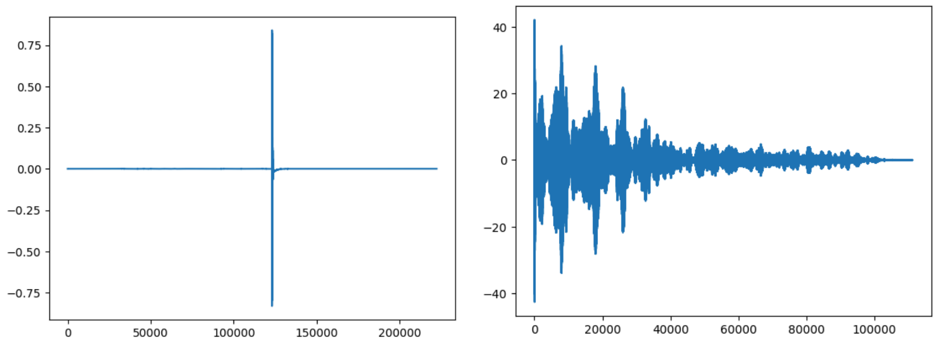


Figure 9: Anechoic Recording of a Balloon Pop in Time and Frequency Domain

4.2.2 *Beamspace Transform Implementation*

Initially, the beamspace transform was implemented using a form of Non-Negative Matrix Factorization (NMF) using orthonormalized array weighting factors applied to the spectrogram inputs [7]. However, this method did not result in the expected outputs, as documentation and discussion of this method were not clearly demonstrated by the references mentioned and publications using similar implementations. Compared to another well-known beamforming library in Python known as *Beamformers*, our method seemed to perform similarly. We also experienced issues with normalizing the responses by frequency to output a desirable spectrogram, since all papers surrounding this method only consider a single-frequency carrier assumption. As such, the weighting matrix used in the final implementation must be normalized for each frequency bin in order to ensure that higher frequencies are not undesirably amplified in the output, leading to a loss in low-frequency information and amplified high-frequency noise. Selection of the ideal STFT parameters such as FFT length and hop size also led to noticeable differences in both beamspace separation and model performance during training. Since higher frequency resolution (increased FFT length) is desirable to distinguish between nearby competing frequencies in overlapping speech data, the number of time samples given to the model had to decrease in order to preserve the amount of information given to the model input. This resulted in a tradeoff between time and frequency resolution which is discussed further in the CNN implementation section below. Since the final selection of the beamforming method allows for both time and time-frequency domain beamforming, our real-time model would implement the more efficient time domain formulation such as for use in a Max/MSP patch where these operations are easily translated.

Finally, a significant hurdle was approaching the definition of the beamspace transform itself. In papers we referenced, namely [4] and [6], this term was used very gently to refer to the orthonormalized FFT method of beamforming which results in a “beamspace manifold” onto which the incoming signals are projected. However, as outlined in [5], this method simply refers to the implementation of phase mode excitation beamforming in which the UCA geometry is projected onto a “beamspace” to administer source localization using URA-style subspace DOA algorithms such as MUSIC with this geometry.

4.2.3 *CNN Implementation*

While the main skeleton of our CNN model derives from [6], the team had to make substantial changes in order to utilize it for our implementation. Originally, we planned to modify the code directly from the paper and contact the authors. However, the researchers were not able to give the code to us. Instead, we recreated the design using Pytorch, a Python library used for creating and running machine-learning algorithms. Since our beamspace transform input is a different size than the original model, we adjusted the input-output structure for each of the modules. We did not adjust the number of blocks and planned to test different versions of the model before deciding on a final structure. Generally speaking, if our design needs to be used in a real-time application, the input and model sizes should be as low as possible to keep the processes down; however, our current priority is a working model capable of generating the correct IRM mask.

Next, the synthetic audio dataset needed to be preprocessed for the model. For this, we applied a log-mel spectrogram transform and generated the power density spectrum. Additionally, we performed tests directly inputting the beamspace input and tested between both input versions. So far, a high-resolution version of the direct beamspace input seems effective enough to generate the IRM mask. More testing still needs to be done on the log-mel spectrogram transformation.

After setting all the parameters, the model needed to be trained. To do this, we utilized the Texas Allocation Computer Center's (TACC) Lonestar6 Computing to train our model. This allowed us to iterate and improve models faster than training using our own resources. Using this resource was not straightforward, however, and required a deep understanding of distributed parallel learning and Slurm job usage to operate. In addition, TACC is especially crowded at the end of the semester and becomes difficult to use with increased network traffic. To use the time effectively, many simultaneous jobs were queued while the software team learned to use the system. One roadblock we had to overcome was limited training time. TACC only allows each user to train for two hours depending on allocation. Jobs need to take advantage of the time given by utilizing all GPUs and CPUs during training. Thus the training and preprocessing loops were tuned to work with TACC and take advantage of Pytorch's distributed learning module.

5.0 TEST AND EVALUATION

After designing the product and implementing the solution, extensive testing was conducted to ensure both the hardware and software components were effective. The following sections explain the team's testing methods and results.

5.1 Hardware Testing

After the PCBs were manufactured, a sample Python program was made to test the functionality of all the microphones individually and connected in series when daisy-chained together connected to the USB streamer. Test audio was recorded using MATLAB and played back using a Python script. The program was modified each run to select which channel to play; the top microphone on each PCB represents channel one, and the bottom microphone represents channel two. As more microphones are added, we incremented the channel numbers in the same fashion.

Initial testing was used to determine if the microphone array boards were assembled correctly by recording a short audio clip from each channel, confirming the functionality of all individual microphones. This testing found that the array would stop functioning after 8 channels (half of our 16-channel array). We were able to modify our design by using two clock distribution boards, one for each set of four PCBs, to allow the remaining 8 channels to also be recorded, as having all 16 channels was necessary to meet specifications detailed in Table 1. However, this testing strategy failed to ensure the microphones were 100% functional; while analyzing impulse response data, it was found that two microphones, numbers 9 and 12, had an abnormally higher noise floor compared to the other 14 microphones. These two signals were recovered by the impulse responses collected from their adjacent microphones, 10 and 11 respectively, providing no loss of information regarding azimuthal separation. While phase information regarding elevational differences is lost for these two microphones, the impact is negligible due to the information collected from all other microphones and our limited emphasis on elevational discrimination.

Further testing was conducted to ensure the noise floor for all microphones was identically between all microphones. To do this, the team set up all microphones and recorded a small audio clip in a relatively silent room. Then, a Python script was written to display the waveform data for each microphone individually so we could compare the amplitudes. This once again showed microphones 9 and 12 having orders of magnitude higher noise floors. Both boards were entirely disassembled and upon close inspection found that both microphone chips appeared to have had low amounts of solder on their solder pads. Once re-soldered, we tested them again and found all microphone noise floors were within family and acceptable for continued testing.

5.2 Software Testing

The CNN proved effective at predicting the general IRM mask but struggled with learning fine-grained details. As seen in Figure 10, the predicted IRM masks are similar to actual IRM masks, but there are some areas where the model is unable to learn the specifics of the true mask. Despite this, the overall loss of the model is low. The current model, after 250 epochs, had a final training loss of 0.054 and a validation loss of 0.203 as seen in Figure 11. This tells us that the model is capable of predicting the mask on new data with some effectiveness.

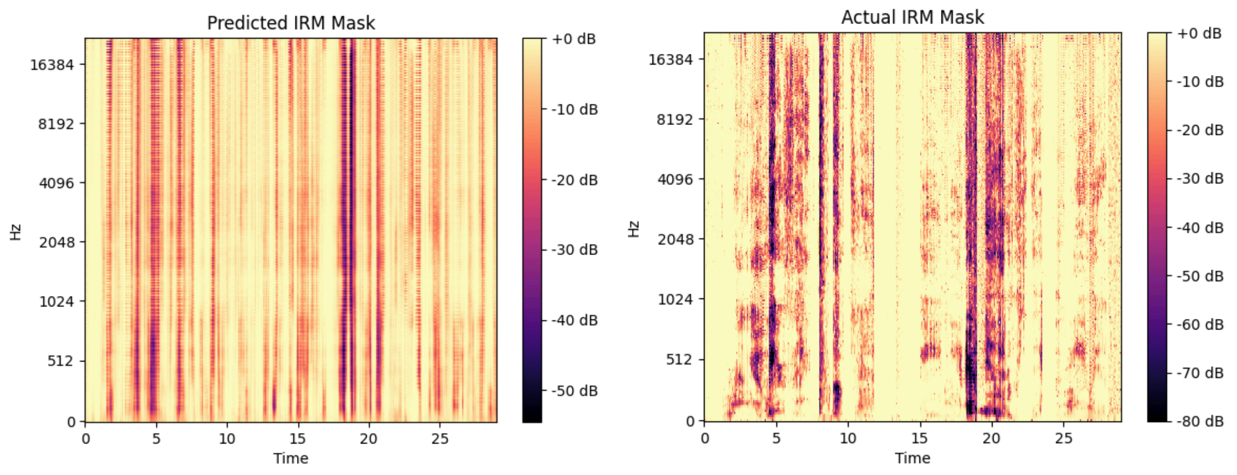


Figure 10: Predicted and Ground Truth IRM Masks

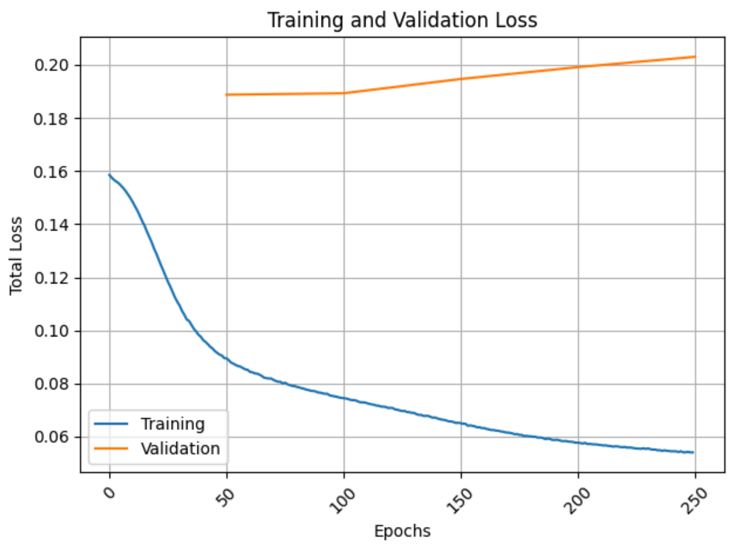


Figure 11: Training and Validation Loss from CNN Model

Additionally, the team compared the quality of the output to the Signal-to-Distortion Ratio (SDR) metrics and the Hearing-Aid Speech Quality Index (HASQI) [8]. SDR aims to quantify the quality of a signal by comparing raw audio to its noisy counterpart. The SDR ratio, plotted in Figure 12, demonstrates that the mask does not distort the original audio with the exception of some areas. We reason that the high SDR is caused by speech transients (short-duration speech sounds) and audio segmentation, or how the data is separated into blocks, during training.

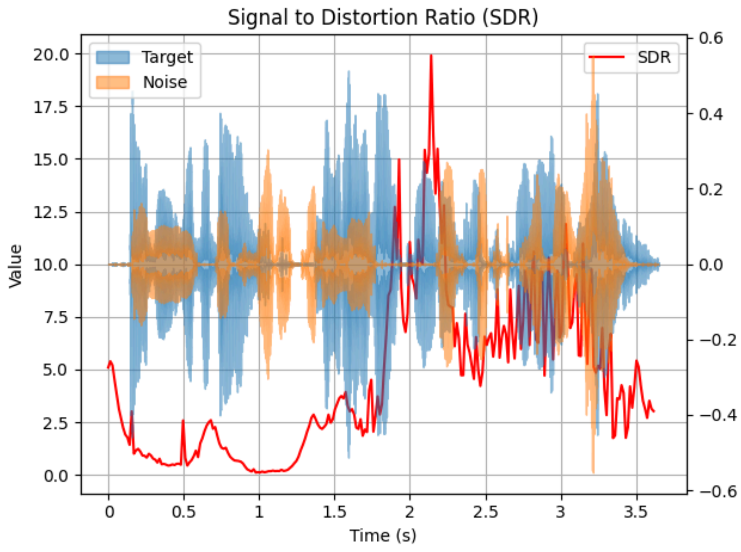


Figure 12: SDR Compared to Target and Noise Signals

The team also measured the audio quality using HASQI, a measurement used in hearing aids to determine speech quality and distortion. To an ear with no hearing loss, the IRM mask has a minor effect on the quality. Both expected and predicted IRM masks resulted in a HASQI score of about 0.81 and 0.82 respectively out of a maximum score of one. This tells us that the drop in quality and amount of distortion caused by the mask is low and still results in intelligible speech.

6.0 TIME AND COST CONSIDERATIONS

The team's first delay came from ordering the PCBs for the UCA. The boards and their components were delayed for weeks, which caused a domino effect. The dataset was necessary to train the model, and to create the dataset, RIRs needed to be collected using the array. Furthermore, the beamspace transform algorithm could not be completed or tested without knowing the final geometry of the array. To navigate around this, the software team developed a preliminary source separation model using the off-the-shelf MiniDSP microphone array as a proof-of-concept.

A second hindrance was due to TACC. Since the team was training the CNN at the end of the semester, and NeurIPS, a major data science conference, had a submission deadline close to the senior design showcase, TACC was overwhelmed with users. As a result, queuing times were long and GPUs were limited. Accessing the TACC service for debugging was also constrained as the number of active terminals was limited. The team was able to do a significant amount of training, but there are plans to continue training and testing the model further after the class ends.

The budget was never a concern. The possible costs were organized before ordering any equipment, so the team was under budget throughout the design process. See Appendix C for the bill of materials.

7.0 SAFETY AND ETHICAL ASPECTS OF DESIGN

There are no safety concerns with this design. However, the team has considered the possibility that, with additional enhancements, the solution could be used as a spying device by listening to a single conversation in a crowded environment. The UCA is large and would be easily spotted in a cafe or restaurant, and the model is trained for a room that is known a priori. Therefore, it is more suitable for a conference room-like setting.

8.0 RECOMMENDATIONS

There are many ways to improve on this design in future research. First, the microphone array could be adapted for a wearable format. It is already flexible, through the use of the individual PCBs connected with ribbon cables, so this is a feasible option. Second, the beamspace transform and model can be fine-tuned for real-time applications. The CNN has a lightweight architecture, so with the proper combination of the number of beamspace transformed spectrograms, frequency bins, and input length, the entire design could run in real-time. Further development also includes training on more environments, more types of audio sources, and a larger number of sources to make the model universal.

9.0 CONCLUSIONS

In conclusion, the team combined hardware and software elements to develop a system to tackle the cocktail party problem and separate two speakers. To this end, the team developed and tested a multichannel circular microphone array capable of reading audio in a 360-degree circle. The team also developed software to read multichannel audio and separate between two speakers by combining both beamforming and machine learning elements. From our project, we conclude that it is possible to develop a deployable system to separate audio and that, given enough time, an efficient, real-time system can be developed. Due to time limitations, we were not able to assemble a fully operational system; however, our work demonstrates that a system utilizing elements of our project is feasible. We recommend more testing and research on designing optimal array structures, testing alternative beamforming strategies, and running other machine-learning architectures to develop a real-time audio separator.

REFERENCES

[1] MiniDSP, "USB Audio Streaming : UMA-16 USB mic array product manual," MiniDSP.com.

[2] T. A. Nguyen, et al, "EXPRESSO: A Benchmark and Analysis of Discrete Expressive Speech Resynthesis," 2023.

[3] Y. Wang, A. Narayanan, & D. Wang, "On Training Targets for Supervised Speech Separation," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 22, no.12, pp. 1849–58, 2014.

[4] V. Ariyarathna, A. Madanayake, "Circular Array N-Beam Digital Beamformer Having Low Arithmetic Complexity: 16 Simultaneous 100 MHz Beams at 2.4 GHz," *IEEE Transactions on Radar Systems*, vol. 1, 2023.

[5] Y. Chen, et al., "DOA estimation for uniform circular array without the source number based on beamspace transform and higher-order cumulant," *Physical Communication*, vol. 59, 2023.

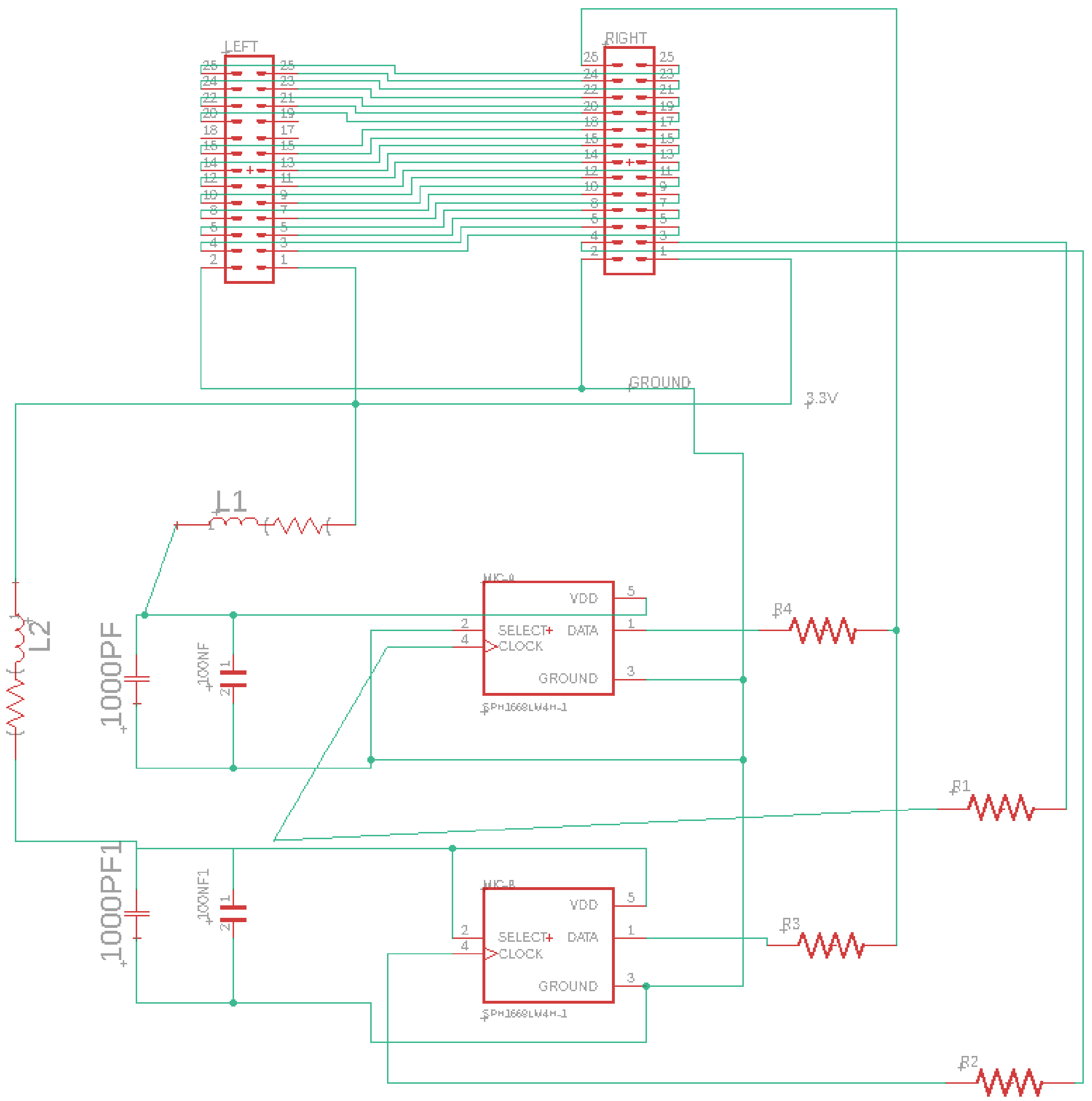
[6] M. Olivieri et al., "Real-Time Multichannel Speech Separation and Enhancement Using a Beamspace-Domain-Based Lightweight CNN," *ICASSP 2023 - 2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Rhodes Island, Greece, 2023, pp. 1-5.

[7] S. Lee, S. H. Park, & K. Sung, "Beamspace-Domain Multichannel Nonnegative Matrix Factorization for Audio Source Separation," *IEEE Signal Processing Letters*, vol. 19, no. 1, 2012.

[8] J. M., Kates & K. H. Arehart, "The Hearing-Aid Audio Quality Index (HAAQI)," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 24, no.2, pp. 354-65, 2016.

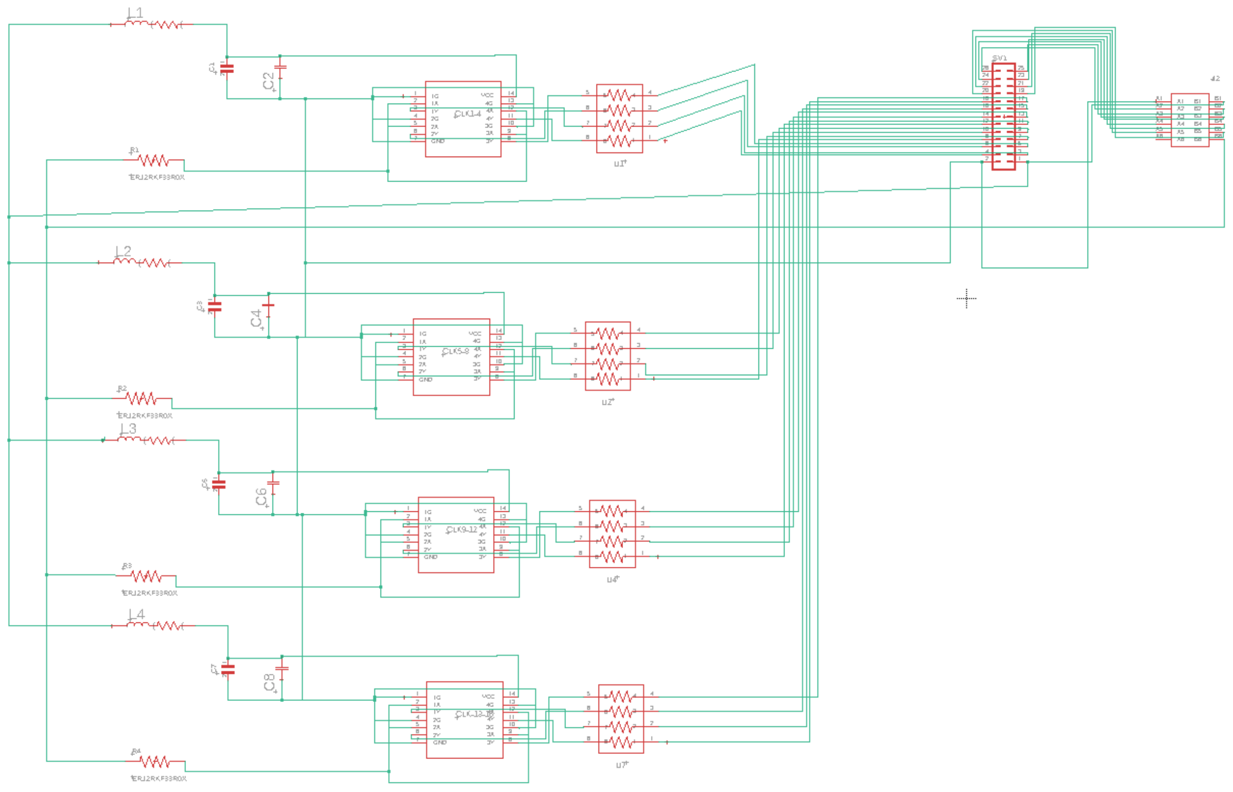
APPENDIX A – TWO-CHANNEL MICROPHONE PCB SCHEMATIC

APPENDIX A – TWO-CHANNEL MICROPHONE PCB SCHEMATIC



APPENDIX B – CLOCK DISTRIBUTION BOARD SCHEMATIC

APPENDIX B – CLOCK DISTRIBUTION BOARD SCHEMATIC



APPENDIX C – BILL OF MATERIALS

APPENDIX C – BILL OF MATERIALS

Item	Part #	Unit Cost	Quantity	Line Total
MiniDSP Microphone Array	UMA-16 v2 USB mic array	$275	1	$275
FERRITE BEAD 600 OHM	490-1014-1-ND	$0.0608	24	$1.46
CAP CER 1000PF 16V X7R	490-6147-1-ND	$0.0086	24	$0.21
CAP CER 0.1UF 25V X7R	490-16477-1-ND	$0.072	24	$1.73
IC BUFFER NON-INVERT 3.6V	497-13480-1-ND	$1.10	4	$4.40
RES ARRAY 4 RES 33 OHM	Y7330CT-ND	$0.079	4	$0.32
MIC MEMS DIGITAL PDM	423-1404-1-ND	$1.274	16	$20.39
RES SMD 33 OHM 1% 1/10W	P33.0LCT-ND	$0.0127	24	$0.31
2.54mm Pitch 26 pins IDC Female Connector	B00DR9J5E8	$12.85	1	$12.85
2.54mm Pitch 26 pins IDC Male Connector	B0BHY5LM7G	$8.45	1	$8.45
IDC 1.27mm Pitch IDC Cable	AZAD19022508	$11.99	1	$11.99
Microphone and Clock PCBs	Custom from JLCPCB	$31.12	1	$31.12
Total Cost				$368.23