

AUDIO RECORDING AND PROCESSING FOR THE HEARING IMPAIRED

Maciej Kaczorek

3rd Year Project Final Report

Department of Electronic &
Electrical Engineering

UCL

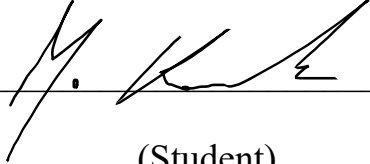
Supervisor: Dr Arsam Nasrollahy Shiraz

30 March 2022

I have read and understood UCL's and the Department's statements and guidelines concerning plagiarism.

I declare that all material described in this report is my own work except where explicitly and individually indicated in the text. This includes ideas described in the text, figures and computer programs.

This report contains 31 pages (excluding this page and the appendices) and 8395 words.

Signed:  _____
(Student)

Date: 30 / 03 / 2022

Table of Contents

1	Introduction.....	5
2	Research objective	6
2.1	Tasks.....	6
2.1.1	Hardware	6
2.1.2	Software.....	6
3	Background Theory.....	7
3.1	Human hearing and cocktail-party problem.....	7
3.2	Ways of tackling the cocktail-party problem.....	7
3.3	Beamforming	8
3.3.1	Frost beamformer	8
3.3.2	MVDR	10
3.3.3	Generalised Sidelobe Canceller.....	11
3.4	Theory behind hardware	12
3.4.1	Sampling, aliasing and anti-aliasing filter.....	12
3.4.2	Quantisation.....	12
3.4.3	Aliasing.....	12
3.4.4	Anti-aliasing filter	12
4	Literature review	13
4.1	Frost Beamformer	13
4.2	MVDR	13
4.3	GSC	14
4.4	Comparison.....	14
5	Methods and experimental setup.....	15
5.1	Hardware development	15
5.1.1	Microphones	15
5.1.2	Preamplifier.....	15
5.1.3	Anti-aliasing filter	17
5.1.4	Manufacturing PCB	18
5.2	Data acquisition	19
5.2.1	MSP432	19
5.2.2	Matlab.....	20
5.3	Beamforming algorithms.....	20
5.3.1	Frost beamformer	20
5.3.2	MVDR	20
5.3.3	Generalised Sidelobe Canceller.....	20
5.4	Experimental setup	20
5.4.1	Simulations.....	21
5.4.2	Physical experimental setup	21
6	Results.....	22

6.1	Validation of hardware	22
6.1.1	Low-pass filter	22
6.1.2	Microphone preamplifier	23
6.1.3	Microphone Array	24
6.2	Simulations	25
6.3	Physical experimental setup results	26
6.4	Analysis and discussion	27
7	<i>Conclusion and future work</i>	28
7.1	Future work	29
7.2	References	29
A	<i>Code for the MSP microcontroller</i>	34
B	<i>Matlab Code – Frost beamformer Simulation.....</i>	36
C	<i>Matlab Code – MVDR simulations.....</i>	38
D	<i>Matlab Code – GSC Simulations</i>	40
E	<i>Matlab Code – Frost.....</i>	42
F	<i>Matlab Code – MVDR.....</i>	45
G	<i>Matlab Code – GSC</i>	47

Audio recording and processing for the hearing impaired

Maciej Kaczorek

The report describes the design of a speech enhancement system that could be used to improve hearing aid devices in so-called cocktail party scenarios. The system is based on a microphone array consisting of 3 microphones, 3 preamplifiers, an MSP432P401R development board and a computer. Beamforming algorithms were tested and their results were presented. Firstly, in a simulated noisy environment on a simulated microphone array. Secondly, on designed hardware. In the simulated setup, Frost beamformer outperformed the other two tested algorithms attenuating noise by 21.16 dB. Testing algorithms on the design hardware showed different results. In this case, Generalised Sidelobe Canceller performed better than the other two tested algorithms attenuating noise by 14.69 dB. At the end of the report a possible explanation of these results is presented and future work is discussed.

1 Introduction

Hearing loss is one of the most common age-related disabilities. As of 2019, an estimated 1.57 billion people experienced some form of hearing loss globally. The data indicates that by 2050 2.45 billion people will have it, which would be a 56.1% increase in comparison to 2019 [1]. As populations age, the number of people with hearing loss will increase, therefore the demand for robust hearing aids will increase in the future. People with hearing loss suffer from the so-called “cocktail party problem”. It can be described as the problem of perceiving speech in noisy social settings such as a visit to a restaurant or a party [2]. Evidence suggests that abnormally broad binaural pitch fusion in people with impaired hearing is among the most important factors causing this problem [3].

Conventional hearing aids do not tackle that problem efficiently as their goal is to just amplify the input signal. This does not improve speech intelligibility as noise is amplified as well as the desired signal. Researchers over the past few decades tried to find different solutions for tackling this problem. Some of them include simple approaches like using Wiener Filters or Spectral Subtraction in the signal processing stage or using directional microphones, however, those approaches do not exhibit sufficient performance [4]. More advanced processing schemes include integrating auditory scene analysis in speech enhancement algorithms based on the Monte Carlo scheme or independent component analysis, however, the computational complexity of such algorithms makes them ineffective to use in present hearing aids [5].

One of the interesting solution that could tackle the cocktail party problem is incorporating beamforming into hearing aids. Beamforming is a technique of integrating sensor array outputs to generate a beam in a certain direction, altering the spatial directivity of sensors. Therefore, both hardware and software help tackle the described problem which helps reduce the computational complexity of the algorithms and provides sufficient speech intelligibility improvement [6]. Researchers have explored different array designs and approaches to beamforming. One of the most successful designs was developed by Bernard Widrow which consists of 6 microphones [6]. Most studies in the field of beamforming have only focused on exploring the performance of those algorithms with microphone arrays composed of 4 microphones or more, as more microphones enhance the performance of the system. However, more microphones mean also a higher cost of the hearing aid and decreased comfort of wearing

it as they occupy more space. Therefore, the report explores the possibility of developing such system using low-cost hardware consisting of only 3 microphones and a low-cost microcontroller. It could be an alternative to high-cost, sophisticated hearing aids and could make a progress towards increasing access to high-end solutions for people coming from low-income backgrounds. In order to find an answer to the stated research question, 3 different popular beamformers are evaluated and compared on the proposed low-cost hardware: Frost beamformer, the Minimum Variance Distortionless Response beamformer (MVDR) and the Generalised Sidelobe Canceller (GSC).

The report is structured in the following way. Firstly, the objective of the project is stated based on the problem discussed in this section. Secondly, the background theory is introduced in order to familiarise the reader with the field of speech enhancement in hearing aid devices and different beamforming algorithms tested in the project. Furthermore, literature is explored to further explore advances in research on selected beamforming algorithms and set the background for future work. Moreover, methods are presented, as well as results which are discussed and analysed. Finally, the conclusion is presented with a section indicating what aspects of the project could be further explored or corrected in the future.

2 Research objective

The main objective of the project is to compare the performance of 3 different beamforming algorithms and investigate possibility of developing such system on a low-cost hardware consisting of only 3 microphones. This approach could significantly reduce the cost of hardware and power consumption of the device.

2.1 Tasks

The tasks can be divided into two stages: hardware development and software development.

2.1.1 Hardware

- Selection of appropriate components such as amplifier chip and microphone types that will meet the requirements set in the WHO's 'Guidelines for hearing aids and services for developing countries' such as "basic frequency response of the system must be between 200 Hz and 4000 Hz" [7].
- Designing the circuit and investigating its performance.
- Developing the circuit on a printed circuit board (PCB).

2.1.2 Software

- Designing appropriate testing routine.
- Performing simulations of the noisy environment and testing the algorithms in the simulated environment.
- Investigate and compare performance of algorithms on a real signals captured using the developed hardware.

3 Background Theory

3.1 Human hearing and cocktail-party problem

As explained in [8] “Hearing is the process by which the ear transforms sound vibrations in the external environment into nerve impulses that are conveyed to the brain, where they are interpreted as sounds.” Human hearing range is from 20 Hz to 20 kHz with age hearing deteriorates and typical hearing upper limit for an adult human is 17 kHz [9]. Important role in human hearing plays the pinna located in the external ear, which helps capture sounds in the environment. Then the external ear canal transfers audio signal to the tympanic membrane. This membrane and three middle ear bones helps transfer sound pressure from the air into the fluid and tissue-filled inner ear [10]. This fluid set into motion hair cells which are “the sensory receptor cells of hearing and balance”. They convert that movement into neural information which is transmitted to the brain [11].

As described in [2] humans are able to distinguish speech sources “using primitive features such as spatial location and fundamental frequency.” People with hearing loss are unable to recognise those sounds on the same level as people with normal hearing. New research suggests that the origin of this problem might be due to abnormally broad binaural pitch fusion which leads to blending sounds coming into ears in a way that often makes them unintelligible [3]. Therefore, in recent years many effort has been made in order to improve hearing aids by implementing techniques that would eliminate cocktail party problem and significantly improve the satisfaction of hearing impaired people [12].

3.2 Ways of tackling the cocktail-party problem

Some speech enhancing methods include: use of directional microphones, microphone arrays and beamforming, use of adaptive noise reducing filters, multiband compression schemes and implementing machine learning algorithms [4], [12], [13].

Directional microphones capture sound incoming from the front of the listener while limiting amplification of sound arriving from the sides and back. Modern hearing aids often use adaptive directional microphones that change directional pattern based on the location of speech or noise source in the environment. However, not always directionality is desired, therefore using only directional microphones in the hearing aids limits the flexibility of the system [14], [15].

Adaptive noise reducing filters are also often introduced in the hearing aids. This approach will be explored more deeply in the next section of the report as those filters are also used in beamforming algorithms, however it is worth mentioning that they are frequently implemented as a speech enhancement technique without the addition of beamforming. Noise reducing filters work by obtaining an estimate of the noise spectrum and attenuating those frequency band in which noise exceeds the speech [16]. Some of the adaptive filters including algorithms like Normalized Least Mean Square (NLMS) or Recursive Least Mean Square (RLMS) are used to cancel the internal noise of the hearing aid caused by acoustic coupling between the microphone and the speaker which are very close to each other in small hearing aids [17].

Improvement of speech intelligibility can also be achieved by using multiband compression schemes. In this approach speech signal is segregated into a number of frames and split into different frequency bands. Then, compression factor is applied to the frequency samples of

each band [18]. It is a useful method of transferring the wide dynamic range of audio signals into a limited dynamic range of people experiencing hearing loss [19]. This approach is often combined with the use of noise reducing filters like Wiener filter used in [18].

Recent developments in machine learning field also led to advances in noise reduction techniques used in hearing aids [20]. Supervised deep neural networks are used in order to improve speech intelligibility and this approach was proven to perform very well also in environments where noise is non-stationary [12], [21]. Moreover, as stated in [12] implementing deep learning technology and bio-feedback into hearing aids will be able to provide solutions tailored to each individual user and every single listening environment. However, this approach usually increases the cost of hearing aid and computational complexity of the system which is undesirable [12].

As the report focused on the use of beamforming and microphone arrays in improving speech intelligibility. This approach will be discussed in depth in the next section.

3.3 Beamforming

Beamforming is the technique of integrating sensor array outputs to generate a beam in a certain direction, altering the spatial directivity of sensors. It is a technique for monitoring signals from a certain direction while dampening the signals from other directions [22]. This is particularly helpful in a cocktail party scenario as the desired signal (speech) usually comes from one direction at a time. There are two main features that determine the performance of a beamforming array: geometry of an array which is usually established by physical constraints (design of a hearing aid) and design of weightings at each sensor output [23]. Generally, they are divided into two groups: adaptive and non-adaptive beamformers. In non-adaptive beamformers, all sensor output weights are fixed according to some desired specification, whereas weights in adaptive beamformers are changing in real time based on implemented adaptive algorithms [4]. The robustness of beamformers can be improved using a method called diagonal loading [24]. This technique improves the performance of the algorithm if there is a mismatch between the steering direction and the actual direction of the desired signal. It is based on adding a loading factor to the covariance matrix of observations. This stabilises the matrix and makes the adaptive vectors more robust to those mismatches [25].

3.3.1 Frost beamformer

Frost beamformer was first developed by Otis Lamont Frost in 1972 under the name “Constrained Least Mean Squares Algorithm”. It is an improvement of Least Means Square Algorithms (LMS) developed by Widrow described in [26]. Major advantage of the algorithm is the fact that only two parameters have to be specified: the direction of arrival of the desired signal and a frequency band of interest. The algorithm gradually learns statistics of noise originating from directions other than the look direction throughout the adaptive process [27]. The main building blocks of the beamformer are: delay stage which steers the array in a desired direction and a bank of adaptive finite impulse response (FIR) filters.

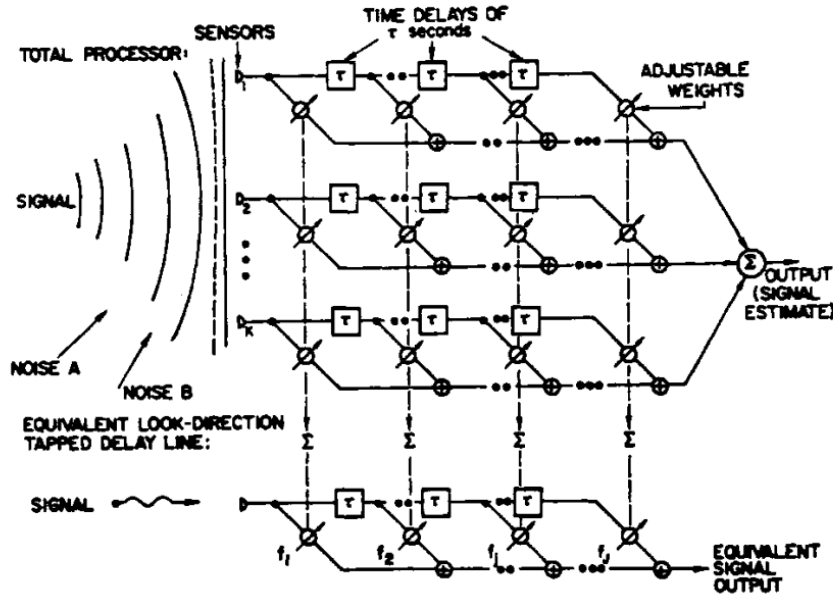


Figure 1: Diagram of the Frost beamformer. Source: [27]

To understand the Frost algorithms let us consider the vector of tap voltages at the k th discrete time sample $\mathbf{X}(k)$. The tap voltages are the sum of signals due to look-direction waveforms \mathbf{l} and non-look-direction noises \mathbf{n} , so that:

$$\mathbf{X}(k) = \mathbf{L}(k) + \mathbf{N}(k)$$

Equation 1: Equation showing tap voltages. Source: [28]

It is assumed that the vector of desired signal waveform is uncorrelated with the noise signals waveforms. Therefore, the output of the array at sample k is:

$$\mathbf{y}(k) = \mathbf{h}^T \mathbf{X}(k)$$

Equation 2: Output of the array at sample k . Source: [28]

,where $\mathbf{h} = [h_0 h_1 \dots h_{L-1}]^T$ is a FIR filter of length L .

Thus, expected output power of the array is defined as:

$$E[\mathbf{y}(k)] = E[\mathbf{h}^T \mathbf{X}(k) \mathbf{h} \mathbf{X}(k)^T] = \mathbf{h}^T \mathbf{R}_{xx} \mathbf{h} = J(\mathbf{h})$$

Equation 3: Expected output power of the array. Source: [28]

,where \mathbf{R}_{xx} is the correlation matrix, assumed to be full rank, of the vector of tap voltages at the k th discrete time sample $\mathbf{y}(k)$ [28].

Constraint on the filter \mathbf{h} is defined as:

$$\mathbf{C}^T \mathbf{h} = \mathbf{u}$$

Equation 4: Equation for constraint on the filter \mathbf{h} . Source: [28]

,where \mathbf{C} is the constraint matrix of size $L \times L_C$ and $\mathbf{u} = [u_0 u_1 \dots u_{L_C-1}]^T$ is a vector of length L_C containing weights of look-direction-equivalent tapped delay line (shown on Figure 1) [27].

Because the J restrictions fixes the look-direction-frequency response, minimising the non-look-direction noise power is the same as minimising the overall output power. Therefore, the following optimisation problem has to be solved:

$$\min_{\mathbf{h}} J(\mathbf{h}) \text{ subject to } \mathbf{C}^T \mathbf{h} = \mathbf{u}$$

Equation 5: Equation for optimisation problem to calculate Frost filter. Source: [28]

Using the method of Lagrange multipliers, the Frost filter is found as:

$$\mathbf{h}_F = \mathbf{R}_{xx}^{-1} \mathbf{C} (\mathbf{C}^T \mathbf{R}_{xx}^{-1} \mathbf{C})^{-1} \mathbf{u}$$

Equation 6: Equation for Frost filter. Source: [28]

The tapped delay line is implemented using basic Delay-Sum beamformer. Delay-Sum beamformer shifts signal at each input to the microphone array by some amount depended on the focus direction. As a result of that, some signal components are in phase (desired signal), therefore amplified, whereas other signals are out of phase [29].

3.3.2 MVDR

MVDR beamformer was proposed by Capon in 1967. Its structure is typical to a basic beamformer. A set of weights is applied to the inputs of the array. Then, the inputs multiplied by the set of weights are summed together to create an output. MVDR adjusts a weight factor to produce power with minimal interference and noise in the desired direction [30]. Advantage of this beamformer is its low computational complexity as its structure and performance is simpler than the other two beamformers discussed in this report [31]. However, this is also a disadvantage. Because of that MVDR is very sensitive to even small inaccuracies in the array steering vector, which is a case almost always due to the existence of array imperfection [32].

Derivation of the MVDR beamformer is presented below. Let us consider the signal input to the array:

$$\mathbf{x}(k) = a \mathbf{v}_0(k) + \mathbf{n}(k)$$

Equation 7: Equation showing signal input to the array Source: [32]

,where a is the amplitude, $\mathbf{v}_0(k)$ is the desired signal, \mathbf{n} is the noise and k is a discrete time sample. It is assumed that $\mathbf{v}_0(k)$ is deterministic.

Output of the beamformer can be written as follows:

$$\mathbf{y}(k) = \mathbf{h}^T \mathbf{x}(k) = \mathbf{h}^T a \mathbf{v}_0(k) + \mathbf{h}^T \mathbf{n}(k)$$

Equation 8: Output of the beamformer. Source: [32]

,where \mathbf{h}^T is a transpose of filter weights.

The goal of the beamformer is to minimise the variance of $\mathbf{h}^T \mathbf{n}(k)$, which can be calculated as:

$$\mathbf{E}\{|\mathbf{h}^T \mathbf{n}(k)|^2\} = \mathbf{E}\{\mathbf{h}^T \mathbf{n}(k) \mathbf{n}(k)^T \mathbf{h}\} = \mathbf{h}^T \mathbf{S}_n \mathbf{h}$$

Equation 9: Equation for variance of $\mathbf{h}^T \mathbf{n}(k)$ Source: : [32]

, while not affecting the desired signal i.e. keeping the gain of $\mathbf{h}^T a \mathbf{v}_0(k)$ equal to 1.

This can be written as:

$$\min_{\mathbf{h}} \mathbf{h}^T \mathbf{S}_n \mathbf{h} \text{ subject to } \mathbf{h}^T \mathbf{v}_0(k) = 1$$

Equation 10: Optimisation problem. Source: [32]

As in the case of Frost beamformer, this optimisation problem can be solved using the method of Lagrange multipliers, leading to:

$$\mathbf{h} = \frac{\mathbf{S}_n^{-1} \mathbf{v}_0(k)}{\mathbf{v}_0(k)^T \mathbf{S}_n^{-1} \mathbf{v}_0(k)}$$

Equation 11: Equation for calculating the weights of the MVDR beamformer. Source: [32]

As stated in [33] MVDR is “perhaps one of the most widely used adaptive beamformers”.

3.3.3 Generalised Sidelobe Canceller

Generalized sidelobe canceller beamformer has been widely used in digital hearing aids for its high performance and low computational complexity [34]. Structure of GSC was based on the structure of Frost beamformer. It consists of two parts: upper part and lower part. Upper part includes fixed beamformer (Delay-Sum beamformer discussed in the Section 3.3.1). Lower part contains blocking matrix with noise canceller [35]. Noise canceller in GSC is realised by using unconstrained least-means squared algorithm [36].

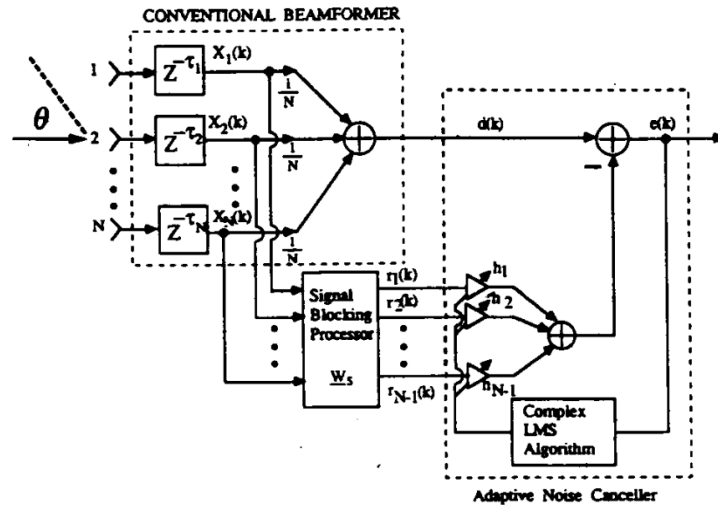


Figure 2: Structure of GSC beamformer. Source: [37]

The purpose of the blocking matrix is to block the desired signal from the lower path of the beamformer. Operation of blocking matrix is perfectly explained in [36] “since the desired signal is common to each of steered sensor outputs blocking is ensured if the rows of BM, sum up to zero”. Moreover the following condition has to be satisfied:

$$\mathbf{b}_m^T \mathbf{1} = 0$$

Equation 12: Condition for the blocking matrix. Source: [36]

,where \mathbf{b}_m is used to represent the m^{th} row of the blocking matrix. Moreover all \mathbf{b}_m are linearly independent.

As mentioned above noise canceller is realized using LMS algorithm. LMS algorithms was developed by Widrow and described in [26]. Since in GSC, output of the blocking matrix

contains only noise the LMS filter weights are updated using the following LMS modified equation [36]:

$$\mathbf{h}(k+1) = \mathbf{h}(k) + \mu \mathbf{e}(k) \mathbf{R}(k)$$

Equation 13: Equation for updating LMS filter weights. Source: [36]

,where $\mathbf{h}(k)$ are the filter weights, $\mathbf{e}(k)$ is the output of the beamformer shown of Figure 2, $\mathbf{R}(k)$ is the input to the noise canceller and μ is the step size with which the weights are updated.

3.4 Theory behind hardware

3.4.1 Sampling, aliasing and anti-aliasing filter.

Sampling is the process of converting continuous signals into a discrete sequences of numbers, while preserving the information present in those signals by measuring amplitudes of the signal with a speed specified by sampling frequency [38][39]. This process can be performed by a device called analogue to digital converter (ADC). Its sampling rate and resolution have a direct influence on how much of that original information is acquired. Sampling rate is the parameter specifying speed with which ADC is working. The faster the sampling rate, more discrete numbers define the recorded signal i.e. acquired waveform can be represented more accurately [38].

3.4.2 Quantisation

The process of transforming a discrete-time continuous-amplitude signal to a discrete-time discrete-amplitude signal is known as quantisation. Not only must we sample an analogue signal in time, but we must also round the signal amplitudes to a finite range of values [39]. ADC's resolution specifies to how many of those finite values we can round the signal's amplitude. The lower the resolution, the more of so-called "quantisation noise" is present in the recorded signal. That noise is caused by the difference between the actual sample's amplitude and the recorded one. Effect of it can be evaluated by calculating the signal-to-quantization noise ratio (SQNR) [39].

3.4.3 Aliasing

If the sampling frequency is slower than the frequency of the signal ADC tries to acquire, a phenomenon called aliasing occur. In this case high frequency signal is recorded as a low frequency signal and introduces distortion and artefacts in the recorded data [40]. To avoid that system's maximum sampling rate has to be equal to the Nyquist limit which is twice the highest frequency component that the system is designed to record [41]. This limit is also defined by Shannon's theorem described in [42]: "If a signal over a given period of time contains no frequency components greater than f_x , then all of the needed information can be captured with a sample rate of $2 \times f_x$ ".

3.4.4 Anti-aliasing filter

Aliasing can be avoided by filtering the analogue signal using low-pass filter before passing it to the ADC. In the described project this is achieved using low-pass second order Butterworth filter with a cut-off frequency of 20 kHz. There are different filter types such as Chebyshev, Butterworth or Elliptic. Chebyshev and Elliptic filters have sharper attenuation slopes than Butterworth, however they create ripples in the passband frequency range [43]. Butterworth

filters are designed to have as flat frequency response in the passband as possible, therefore this type of filter was used in the project as flat passband is desired when processing speech signals. Design of the Butterworth filter used in the project is explored more deeply in the methods section, however equation for the filter's cut-off frequency is worth mentioning at this point.

$$f_H = \frac{1}{2 * \pi * \sqrt[2]{R_1 R_2 C_1 C_2}}$$

Equation 14: Equation for Butterworth filter cut-off frequency. Source: [43]

,where $R_1 R_2 C_1 C_2$ are values of the resistors and capacitors included in the circuit. Circuit's diagram is presented in the Methods Section of the report.

4 Literature review

4.1 Frost Beamformer

As mentioned in the Theoretical Background Section Frost beamformer was based on LMS algorithms developed by Widrow. Widrow designed hearing aid system based on his algorithm. Its construction and principle of operation were described in [6], [14]. One of the disadvantages of Widrow's system is a lack of any further filtering besides beamforming. Therefore if there is an unwanted signal, in the same frequency as the desired signal, coming from the same direction as the steering direction that signal is not filtered out. Moreover, the mentioned algorithm requires prior knowledge of the noise statistics [27]. Frost improved Widrow's algorithm and presented it in [27]. This method does not require prior knowledge of the noise statistics, which is beneficial as the sound environment of the hearing aid user can change rapidly. However, the mentioned technique has the same disadvantage as Widrow's one. Namely, it is not able to filter out the noise of the same nature as the desired signal coming from the steering direction, therefore further filtering is required. Furthermore, it is assumed that "the non-look-direction noise voltages on the taps be uncorrelated with the look-direction signal voltage" [27]. If this requirement is not met, some of the desired signal may be filtered out making speech unintelligible. Some of the methods of improving Frost beamformer include the use of diagonal loading. Diagonal loading was implemented along with Frost beamforming in [44]. Diagonal loading improved the gain of an array by 0.25 dB. Moreover, adaptive diagonal loading was implemented based on Neighbourhood Field Optimization algorithm, which finds the diagonal loading factor and this approach improved the array gain by 0.29 dB. However, the estimated range of the steering direction has to be given which is a disadvantage of the proposed method.

4.2 MVDR

Some of the improvements of MVDR algorithms used in hearing aids include use of target activity detection. Such approach was proposed in [45]. This method is based on switching beamformer between the monaural MVDR and the binaural MVDR at each frequency bin depending on a proposed target activity detection system. The method showed improved robustness to errors in the estimated target direction of arrival in comparison with using either the monaural MVDR or the binaural MVDR on their own. However, the system was evaluated only in simulated environment which is a disadvantage. MVDR was also implemented in [46] alongside speech enhancement gain function based on Log-Spectral

Amplitude estimator. MVDR showed improvement to the system in comparison to using the gain function on its own, which again shows advantages of using MVDR in hearing aid applications. An interesting comparison was made in [47]. The study compares bilateral and binaural MVDR-based noise reduction algorithms in a presence of direction of arrival estimation errors. The simulation findings demonstrate that, in general, the binaural MVDR beamformer outperforms the bilateral MVDR beamformer, even at very low input SNRs. The difference between bilateral and binaural approach is the type of processing. In bilateral configuration the microphone signals are processed separately. This could indicate that processing 3 microphone inputs simultaneously, which was done in this report, is a correct approach.

4.3 GSC

As the beamformers in the project are tested on a simple low-cost hardware, evaluation of the algorithms in the presence of array's imperfection should be explored. This was done in [48]. It was shown that array imperfections lead to leakage of desired signal into the sidelobe cancelling path even despite the blocking matrix discussed in the Background Theory Section. As stated in [48]: "The imperfections were assumed to be due only to random element misplacement, obeying a two-dimensional zero mean Gaussian distribution whose radius had a standard deviation of 0.01". It was shown that this antenna element error variance can lead to even 39dB signal-to-noise ratio (SNR) loss compared with a perfect array. This mean that GSC is extremely sensitive to array imperfections. GSC was tested with an array including 5 microphones in [49]. It is shown that the system is able to attenuate noise sources at 60° angle from the array normal by 5dB. However, 60° angle is quite big, which could influence the result and was considered in designing experimental setup described in Section 5.4. Some attempts were made to improve GSC performance. Such attempt was described in [35], where Wiener filter was used after the GSC. Some improvement was shown, however it was not outstanding.

4.4 Comparison

Sum-delay, MVDR and GSC were compared in [50]. They were tested in the presence of babble, machinery and traffic noise. This study showed that the MVDR algorithm outperform the other two. Moreover, implementing a Wiener filter along with the MVDR technique further improves the performance of the system. In [51] MVDR and Frost beamformer were compared. MVDR showed better performance than Frost filter in the range of frequencies from 100 Hz to 4 kHz. Those two studies indicate that MVDR outperforms both Frost beamformer and GSC. Another example in which MVDR outperforms Frost beamformer was shown in [52]. In this study microphone array of 4 microphones is used with a spacing of 5 centimetres. It was shown that MVDR performs better for a larger values of desired source displacement. It is interesting to notice that the noise reduction is decreasing monotonically for both beamformers as the value of the displacement is increased. In contrast to those results [53] shows that Frost beamformer outperforms MVDR provided that the acoustic environment is time-invariant. However, they state that in slow-varying environments preferred beamformer remains an open question. Microphone array including 11 microphones was used. Results of those two papers could indicate that MVDR performs better on a microphone array with smaller number of microphones.

5 Methods and experimental setup

5.1 Hardware development

The system's hardware consists of three microphones placed five centimetres apart, three microphone preamplifiers, three anti-aliasing filters and a microcontroller that sends data through UART protocol to the computer for further processing. That is the simplest array on which beamforming algorithms can be tested. Firstly, the microphone and microphone preamplifier models were chosen in order to meet requirements specified in the previous parts of the report. Furthermore, the hardware was tested on a breadboard in order to investigate circuit's behaviour. Finally, circuit was developed and tested on a PCB.

Show 3 different microcontroller channels

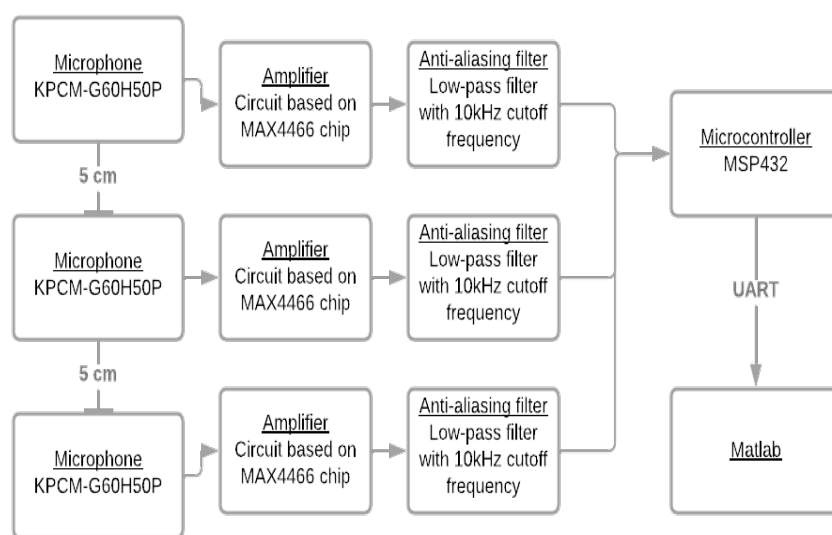


Figure 3: Hardware block diagram

5.1.1 Microphones

Microphones used in the project are electrostatic capacitor-based microphones (i.e. electret microphones). In hearing aids, this is the most commonly used type. Their cost is low as their design is not complicated and they used permanently charged material which eliminates need for polarizing power. Moreover, they are small and have flatter frequency response than the dynamic type of microphones, which is a significant advantage for audio processing of human speech [54]. Furthermore, they are significantly cheaper than MEMS microphones. The microphone model used in the system is KPCM-G60H50P. It is a low-cost omni-directional electret microphone. Its frequency range is between 100 Hz to 10000 Hz, which is sufficient for speech processing applications. It has high signal-to-noise ratio (SNR) (60dB) and sensitivity of -44dB [55]. Sensitivity is important in the next parts of the report as it has a significant role in calculating required amplification of the circuit.

5.1.2 Preamplifier

Signal amplification in the circuit is achieved using Maxim MAX4466 preamplifier integrated circuit (IC) chip. This preamplifier meets all the requirements of a good preamplifier for hearing aid application. It has excellent Power-Supply Rejection Ratio (PSRR) - 112dB and

very high Common-Mode Rejection Ratio (CMRR) - 126dB [56]. Moreover, it fits power requirements of the circuit as it can be supplied by a 3.3V DC source [56]. Furthermore, it is very small which makes it perfect for hearing aid designs. The amplification part of the circuit includes more components such as resistors and capacitors in order to set a required gain of the preamplifier or filter the noise from the power supply.

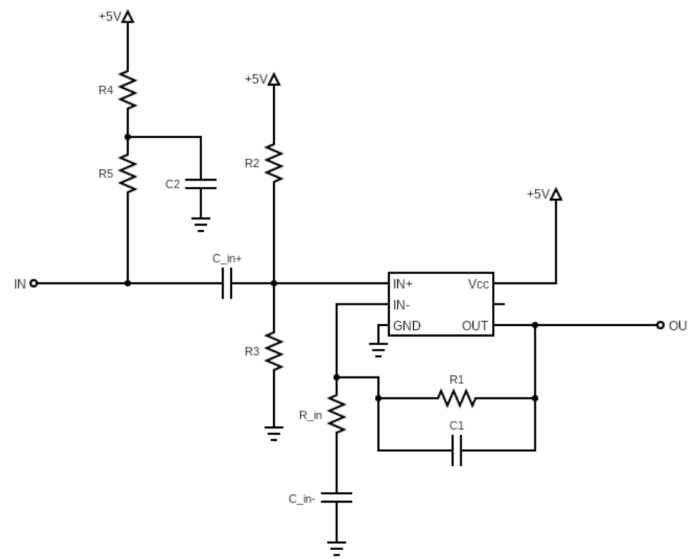


Figure 4: Preamplifier circuit diagram

The majority of the component values were selected based on the typical application circuit presented in the MAX4466 datasheet [56]. Namely, resistors: R_2 , R_3 , R_4 , R_5 and capacitors: C_1 , C_2 , C_{in+} , C_{in-} .

Values of all components included in the circuit are presented in the following table:

Component	Value
Resistor R_1	2.63M Ω
Resistor R_2	1M Ω
Resistor R_3	1M Ω
Resistor R_4	2k Ω
Resistor R_5	2k Ω
Resistor R_{in}	10k Ω
Capacitor C_1	100pF
Capacitor C_2	0.1 μ F
Capacitor C_{in+}	0.01 μ F
Capacitor C_{in-}	100nF

Table 1: Table presenting component values included in the preamplifier circuit

Two resistors were R_1 and R_{in} set the gain of the amplifier. The gain was calculated based on KPCM-G60H50P microphone's sensitivity and the desired maximum input sound pressure level (SPL) which was chosen to be 100 dB. The amplifier will saturate above that level. This value ranges from 90dB to 110dB in typical hearing aids [4]. Sensitivity of the selected microphone is -44dB. The desired gain of the amplifier was calculated in the following way.

Do not populate the equations, just show them and results.

$$\text{Microphone sensitivity} = 20 * \log (\text{transfer factor})$$

Equation 15: Equation for calculating microphone sensitivity

Knowing the sensitivity of KPCM-G60H50P is equal to transfer factor:

$$\text{transfer factor} = 6.3096 \text{ mV/Pa}$$

Therefore for every 1 Pascal, microphone's output increases by 6.3096mV.

Next step was converting the maximum SPL from decibels to Pascals in order to calculate the maximum output of the microphone before the saturation.:

$$SPL = 20 \log \left(\frac{p}{20 * 10^{-6}} \right)$$

Equation 16: Equation for converting Pascals to Decibels

,where p is SPL in Pascals.

$$p = 2Pa$$

Therefore, the output of the microphone for SPL of 100dB is:

$$\text{Microphone output} = 2Pa * 6.3096 \frac{\text{mV}}{\text{Pa}} = 12.6192 \text{ mV}$$

Therefore, the gain of the amplifier can be calculated, as we know that this value has to be amplified to 3.3V, which is the maximum voltage that can be read by the MSP432P4101 microcontroller's analogue to digital converter. Gain of the amplifier should be set to:

$$\text{Gain} = \frac{3.3V}{12.6192mV} = 261.506276$$

After taking into consideration the availability of the components and measuring its actual values, the following resistors were included in the circuit: $R_1 = 2.63M\Omega$ and $R_{in} = 10k\Omega$. Giving actual gain of:

$$\text{Gain} = \frac{R_1}{R_{in}} = 263$$

Therefore, the maximum SPL level applied to the microphone is slightly smaller than the desired 100 dB. The results of maximum SPL measurements were presented in Section 5.1.4 of the report.

5.1.3 Anti-aliasing filter

Role of the anti-aliasing filter is to attenuate frequencies above 20kHz before the signal is read by a microcontroller's analogue to digital converter (ADC). This was done in order to make sure to meet Nyquist's frequency limit. The theory behind aliasing and Nyquist theorem were explained in the background section. The filter used in the circuit is second-order low-pass unity-gain Butterworth filter with a -3db cut-off frequency of 20kHz. Operational amplifier (Op-amp) used in the circuit is Microchip Technology MCP6281. The advantage of that op-amp is the fact that it can operate from a single supply voltage as low as 2.2V. Moreover, it has high PSRR and CMRR: 90db and 80dB respectively. Butterworth type of filter was chosen as

it is designed to have frequency response that is as flat as possible in the passband, which is important requirement in the described system.

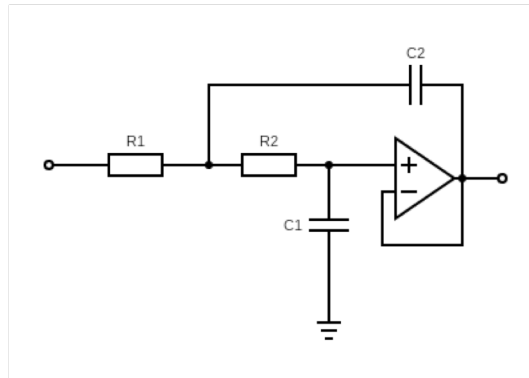


Figure 5: Anti-aliasing filter circuit diagram

As mentioned in the theoretical background section cut-off frequency of the second-order Butterworth filter can be calculated with the Equation 14. Values of resistors R_1 and R_2 were chosen to be the same, as well as capacitors C_1 and C_2 . Therefore, the formula for cut-off frequency can be reduced to:

$$f_H = \frac{1}{2\pi RC}$$

Equation 17: Equation for calculating cut-off frequency of the low-pass filter

Resistors values were chosen to be $10\text{k}\Omega$, therefore capacitors values can be calculated as:

$$C = \frac{1}{2\pi R f_H}$$

Equation 18: Equation for calculating capacitor value

,giving:

$$C = 0.79577472 \text{ nF}$$

After taking into account the availability of components in the lab and measuring the actual capacitance of the capacitors the following resistors were included in the circuit: $C_1 = 0.88\text{nF}$ and $C_2 = 0.88\text{nF}$.

5.1.4 Manufacturing PCB

All 3 circuit components: microphone, preamplifier and anti-aliasing filter were joined together to create one of the channels of microphone array. This was repeated 3 times in order to develop 3 channel microphone array. By manufacturing every channel on different PCB flexibility in microphone spacing was achieved. This way the default microphone spacing (5 cm) can be easily changed if needed in order to test the beamforming algorithms further. Circuit was tested by checking the saturation point which was proven to be a slightly smaller than expected 100 dB. The saturation point was measured to be 96 dB, which is still sufficient for hearing aid applications.

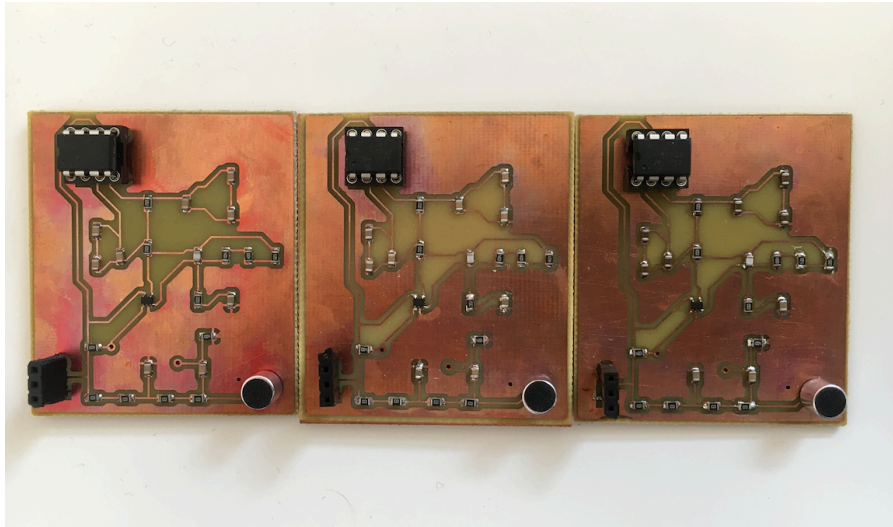


Figure 6: Picture presenting manufactured PCB

5.2 Data acquisition

5.2.1 MSP432

First stage of data acquisition is signal capturing using ADC. Second stage of that process is data transfer through chosen protocol to the computer for further processing. Texas Instruments MSP432P401R microcontroller was selected to perform this task. The reason for choosing this microcontroller was mainly its low cost (£23) and low power operation. It includes 48MHz ARM Cortex M4F processor. Its ADC can be programmed to work very fast (up to) as opposed to other popular microcontrollers such as Arduino UNO. Moreover, MSP432P401R microcontroller featured 3 most popular communication protocols, namely: Universal Asynchronous Receiver/Transmitter (UART) , Serial Peripheral Interface (SPI) and Inter-Integrated Circuit (I²C).

Its ADC range was set to 0V-3.3V. ADC's resolution had to be decreased from default 14 bit to 8 bit in order to match the size of the data packet that can be send through USB to the computer. Splitting one 14bit sample to two packets could not be achieved as that would significantly decrease the baud rate and as a result of that a decrease in sampling frequency would be also observed. Sampling frequency of the ADC was set to 132.3kHz. As three channels are connected to the ADC, this gives 44.1kHz sampling frequency per channel. The communication between microcontroller and UART is achieved by using UART protocol. This protocol was chosen as it can be directly used with a Universal Serial Bus (USB), which is very efficient as no adapters are needed to connect microcontroller to the computer. As UART data packet is made of 10 bits: 1 start bit, 8 data bits, 1 parity bit and a stop bit, the UART's baud rate was set to 1.4553 Mbaud/s. This ensures that in a second the whole data sample is send with the corresponding start and stop bits. Baud rate and sampling rate are generated from the internal clocks of the microcontroller. Testing proved that letting microcontroller sample and transmit data continuously leads to errors and the performance of the system is unstable. Moreover, the channels cannot be identified. Therefore, the sampling and data transfer begins when a character 'D' is received by the microcontroller and ends when a character 'S' is received. Code written to program microcontroller is included in the Appendix A.

5.2.2 Matlab

Matlab software is used in the last stage of data acquisition. Firstly, the software sends character 'D' to the microcontroller through UART to prompt start of the data acquisition. At the same time the message 'start' is displayed in the command window to indicate start of audio recording to the user. The data is received from serial port through function `fread()`. After gathering 396 900 samples, Matlab sends character 'S' to the microcontroller in order to stop the sampling process. 396 900 samples correspond to 3 seconds of acquired signal. When the process is finished the data is split between 3 channels, plotted and ready for further processing. Matlab code for acquiring the data from the serial port was presented in the Appendix B.

5.3 Beamforming algorithms

As mentioned in the previous parts of the report, the project focuses on testing and comparing 3 different beamforming algorithms: Frost beamformer, MVDR and GSC. Matlab software includes functions to perform those beamforming algorithms both on simulated and acquired data. They are included in the Phased Array System Toolbox.

5.3.1 Frost beamformer

Frost beamformer can be implemented in Matlab by calling a system object `phased.FrostBeamformer`. Few important parameters can be specified when calling the object. They are worth mentioning as they have direct influence on the beamforming results. Firstly, sensor array has to be specified by calling a sensor array object which included information such as type of the microphone used in the system, spacing between them, their number and their frequency range. Furthermore, signal propagation speed has to be declared (speed of sound), FIR filter length (having direct influence on the beamforming performance but also computational complexity), direction of the beam and sample rate of the audio signal. Values of those parameters will be specified and explained in the Section 5.4 of the report. Moreover, the `phased.FrostBeamformer` object can return beamforming weights which is particularly helpful when investigating beamformer's behaviour.

5.3.2 MVDR

Performing MVDR beamforming in Matlab can be done by calling the `phased.MVDRBeamformer` system object. Same parameters have to be specified as in the case of Frost Beamformer, therefore its explanation will be omitted. One parameter is not included: FIR filter length because this beamformer does not operate using FIR filters. The `phased.MVDRBeamformer` System object can also return the beamforming weights.

5.3.3 Generalised Sidelobe Canceller

As in the case of two previous beamformers, GSC can be implemented using `phased.GSCBeamformer` System object. Besides parameters mentioned when describing other two Matlab objects, one additional parameter can be specified for GSC, namely `LMSStepSize` which sets the adaptive filter step size that is used in the Least Means Squared algorithm.

5.4 Experimental setup

Experimental setup was designed in order to be efficient, important parameters easy to calculate and the experiment could be performed in home conditions. Firstly the 3 selected algorithms were tested on a simulated microphone array in a simulated noise environment. This was done in order to later compare the difference in performance between different algorithms

and investigate if when used with real audio signal this difference is the same. Furthermore, the experimental setup used in simulations was recreated in physical space. The beamforming algorithms were tested on 3 audio signals, namely 440Hz sine tone, 220Hz sine tone and 880Hz sine tone which is the signal of interest. Sine wave tones were chosen as their signal characteristic is the same which would also be the case of filtering particular person's voice from a mixture of other voices. Moreover, as they do not change in time it is easier to calculate array gain. Calculations of array gain will be explained in the next section.

5.4.1 Simulations

Firstly, the microphone array type, dimensions and type of microphones used in the array were declared in Matlab using phased.ULA object. ULA stands for uniform linear array, which is the type of microphone array used in the project. As in designed physical array spacing between the microphones was set to 5 centimetres. Microphones were simulated using phased.OmnidirectionalMicrophoneElement object and their frequency range was declared to match the frequency range of physical microphones used in the project (100Hz to 10000Hz). Furthermore, phased.WidebandCollector and dsp.SignalSource objects were used to simulate the noisy environment and its collection by the microphone array. Moreover, noise was added to the signal in order to simulate the internal noise of the microphones. The list of signals and their direction of arrival is presented below:

Signal	Direction [azimuth; elevation]
440Hz sine wave	[-45; -30]
220Hz sine wave	[45; 30]
880Hz sine wave	[0; 0]

Table 2: Table presenting signals used to test the beamforming algorithms and their direction of arrival

Matlab does not let user specify distance of the signal sources from the microphone array. This can be done approximately by changing intensity of the signal. Intensity of the 880Hz tone was set to be twice the intensity of other signals as in physical experiment this signal source is two times closer to the array than the others. The filter length for two algorithms that require that parameter (Frost beamformer and GSC) was set to 20. It provides efficient filtering without requiring too much computational complexity. Direction of the beam was set to [0; 0] which matches the direction of arrival of the desired signal. Sample rate of the signal was set to 44.4 kHz. LMSSetpSize in GSC was set to 0.1. Matlab code to generate the simulation was included in the Appendix C.

5.4.2 Physical experimental setup

Simulated environment was recreated in physical setting by placing 3 signal sources in house setting. The direction of arrival and elevation of the sources were the same as in simulations. Signals were played through speakers in specified positions. Diagram of the setup is presented below.

Put a small picture of actual physical setup within the figure.

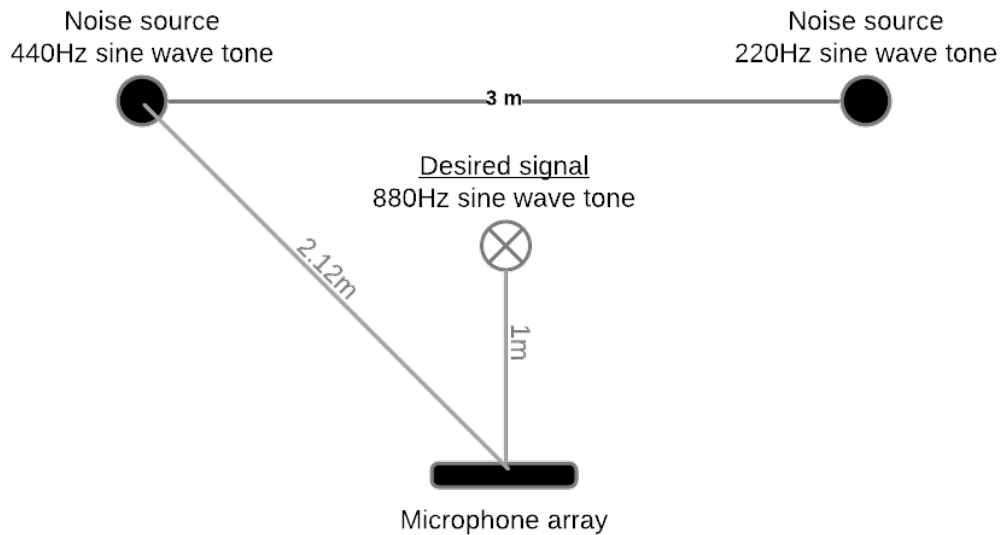


Figure 7: Diagram of the physical, experimental setup

In this case, signals were directly passed to the beamforming algorithms without use of the `Phased.WidebandCollector` and `dsp.SignalSource` objects. `Phased.ULA` object was also declared with the parameters matching physical array as the beamforming objects require that information. Filter length as in the simulations was set to 20. Direction of the beam was also left unchanged as well as `LMSSetStepSize`. When recording the signal, it was discovered that Matlab acquires audio with a slightly lower sample rate than specified, namely CHECK ON WINDOWS. Therefore, sample rate specified when declaring beamforming objects was changed accordingly. Matlab code used in this part of the project was included in the Appendix D.

6 Results

Firstly, the results of the simulations are presented and analysed. Array Gain is calculated and directivity patterns of the beamformers are presented. Furthermore, the same is done for the beamforming using real recorded signals acquired by a physical array. In the last section the results are compared and discussed.

6.1 Validation of hardware

6.1.1 Low-pass filter

The frequency response of the filter was measured and the results are presented in the following graph.

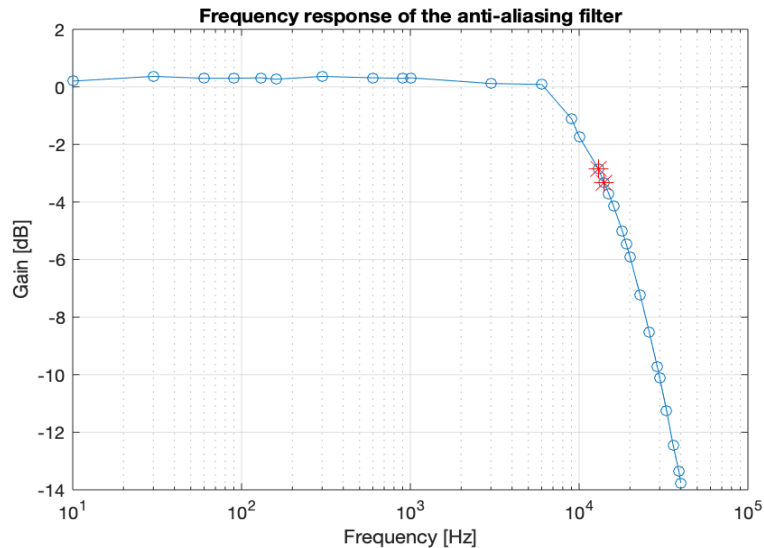


Figure 8: Frequency response of the anti-aliasing filter

As it can be seen the -3dB cut-off frequency is located between 13kHz and 14kHz (marked red on the graph), lower than the expected 20kHz. This can be due to the mismatch in components values. Considering the frequency range of the human voice, this should not influence the performance of the system. The frequency response in the passband region is very flat as expected for Butterworth type of filter.

6.1.2 Microphone preamplifier

Frequency response of the amplifier was tested in order to investigate its performance. Results are presented on the graph below.

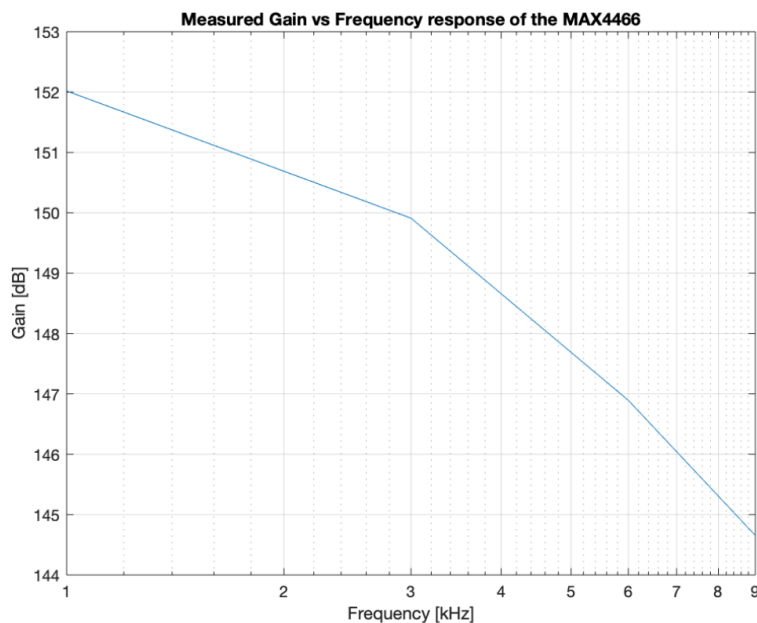


Figure 9: Frequency response of the MAX4466 microphone preamplifier

It can be seen that the frequency response of the amplifier follows the one presented in the datasheet. Therefore, it can be concluded that the design of the amplification circuit was

successful. In the human voice frequency range (0 Hz to 4 kHz), gain of the amplifier drops only by 3 dB.

6.1.3 Microphone Array

Below are presented directivity patters of the designed microphone array generated in Matlab.

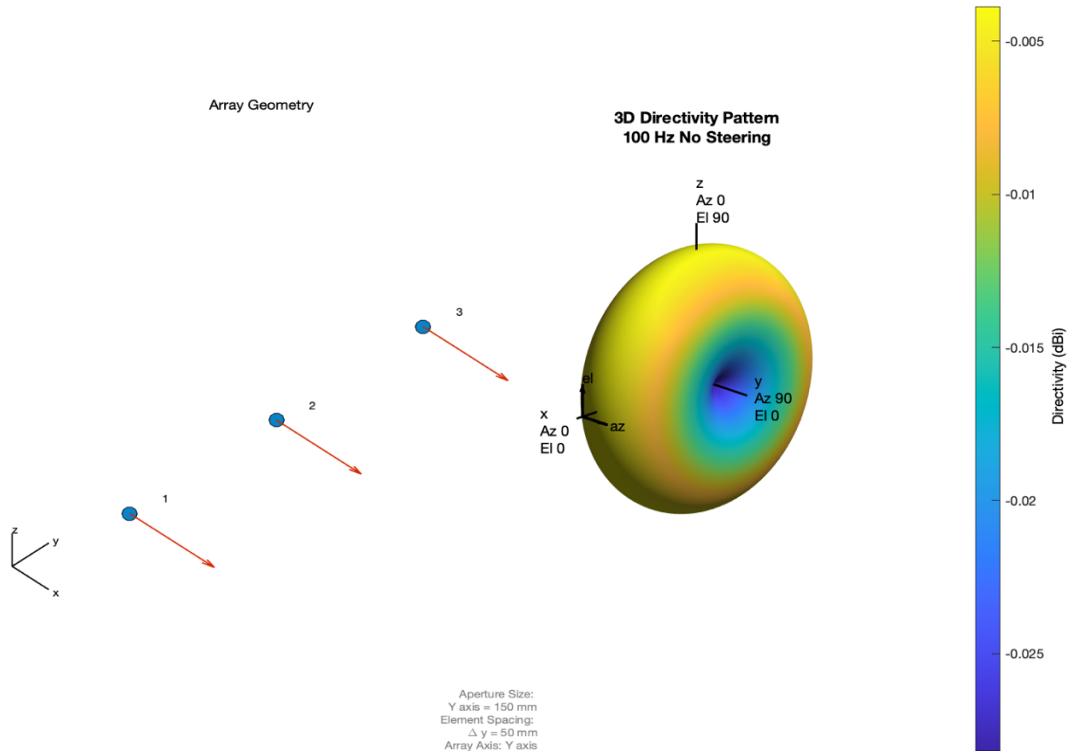


Figure 10: Array geometry and its 3D directivity pattern

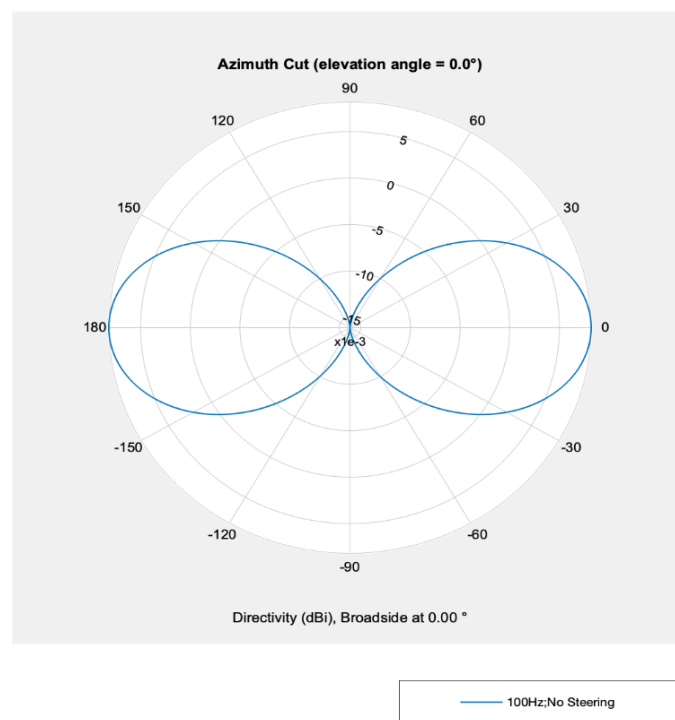


Figure 11: 2D Directivity pattern of the designed microphone array

It can be noticed that the directivity pattern follow the pattern of a typical microphone array designed using omni-directional microphones.

6.2 Simulations

During simulations all 3 beamforming algorithms performed very well. However, it can be noticed that the Frost beamformer heavily outperformed other two. Filtering is also easily noticeable by ear. Beamformers were evaluated by measuring the power of noise at the input to the beamformers and power of noise at the output of the beamformers. Ratio of those two values was called “Noise attenuation” and was expressed in decibels.

Results for Frost Beamformer:

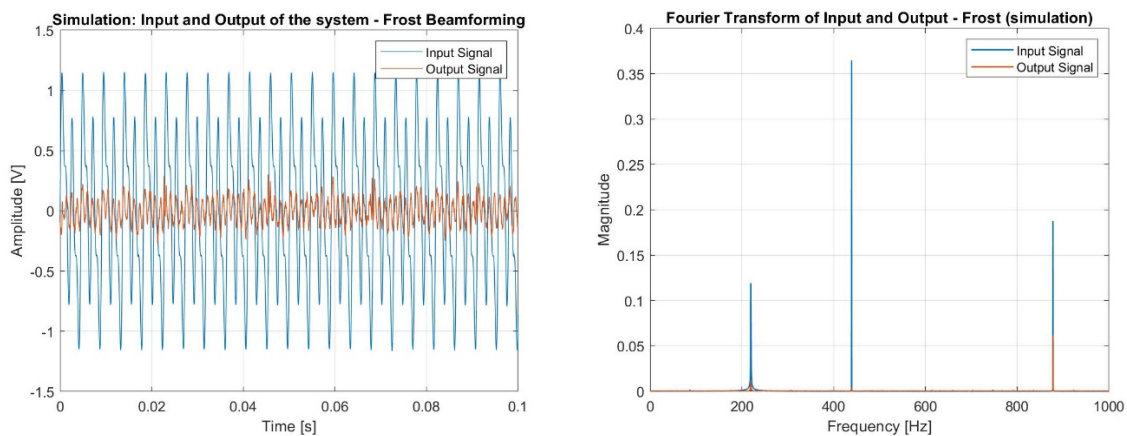


Figure 12: Input and output of the Frost Beamformer and their Fourier Transform

Noise reduction of the Frost Beamformer was measured as 21.16 dB.

Results for MVDR beamformer:

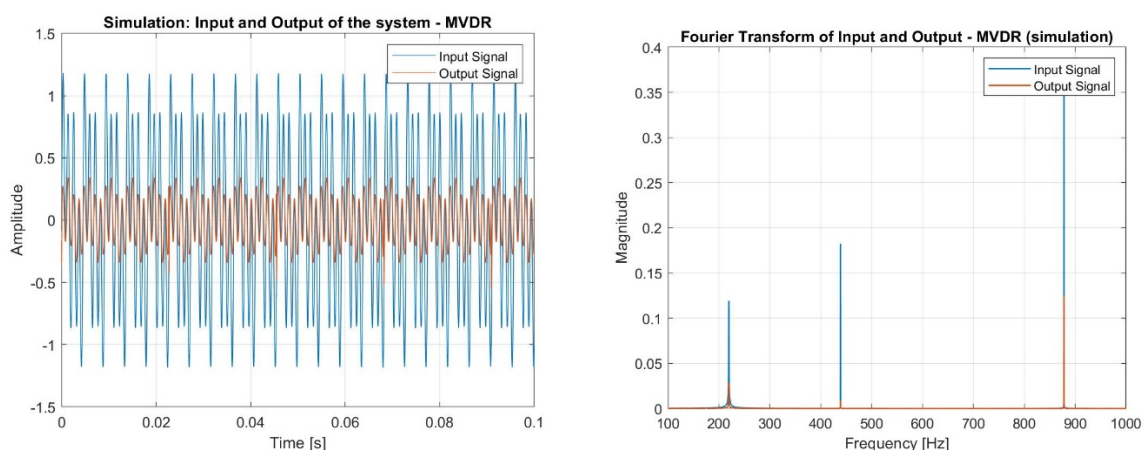


Figure 13: Input and output of the MVDR beamformer and their Fourier Transform

Noise reduction of the MVDR Beamformer was measured as 14.86 dB.

Results for GSC beamformer:

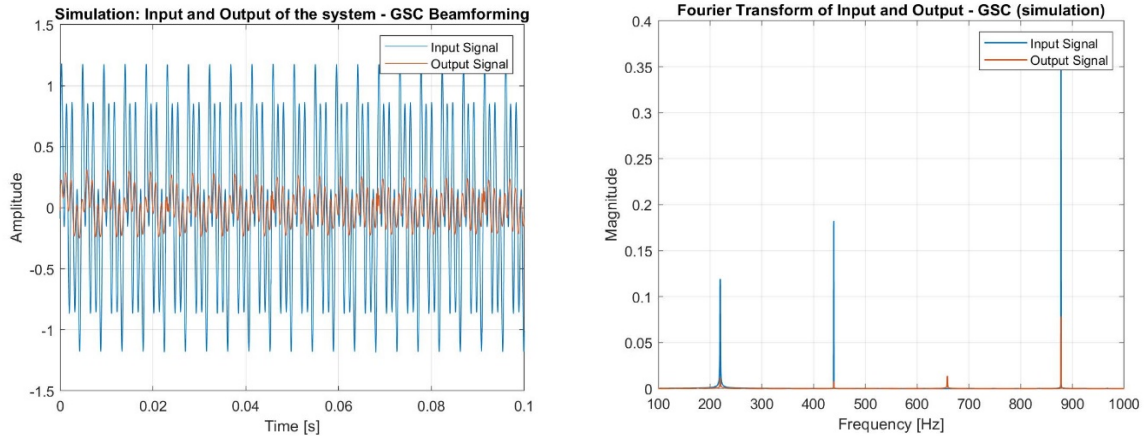


Figure 14: Input and output of GSC beamformer and their Fourier Transform

Noise reduction of the GSC beamformer was measured as 17.51 dB.

All array gain results are presented in the table below:

Beamformer	Noise attenuation (dB)
Frost beamformer	21.16
MVDR	14.86
GSC	17.51

Table 3: Array gains of 3 selected beamforming algorithms (simulation)

As we can see Frost beamformer heavily outperformed other two. Its array gain is twice the one measured for MVDR beamformer. It can be seen on Figure 8 how greatly spikes other than 880 Hz are attenuated, which is also the case for the GSC but not exactly for MVDR. On Figure 9 attenuation of other spikes is noticeable, however it is not as good as in other two beamformers.

6.3 Physical experimental setup results

Firstly, as in the case of simulations the Frost beamformer was evaluated. The results are presented on the graphs below.

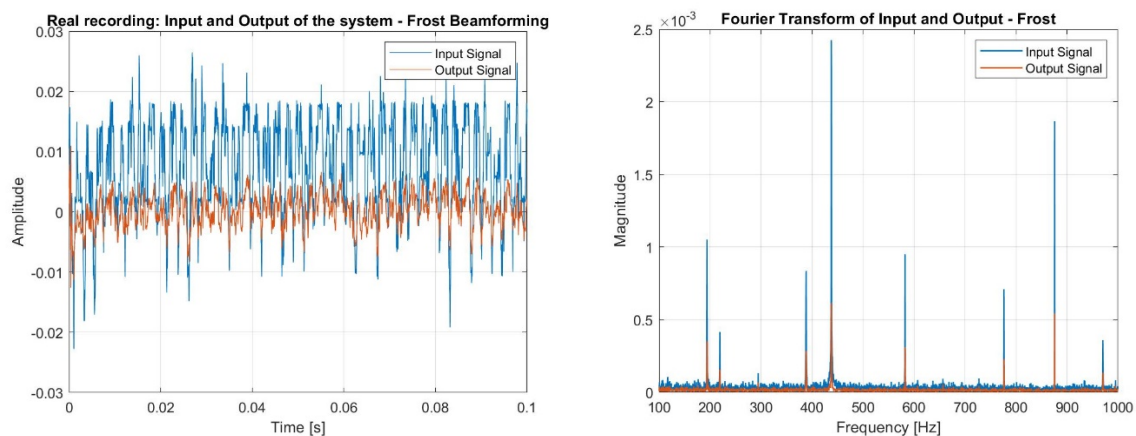


Figure 15: Graphs presenting Frost beamformer input and output and their Fourier transform

Results of MVDR beamforming are presented below.

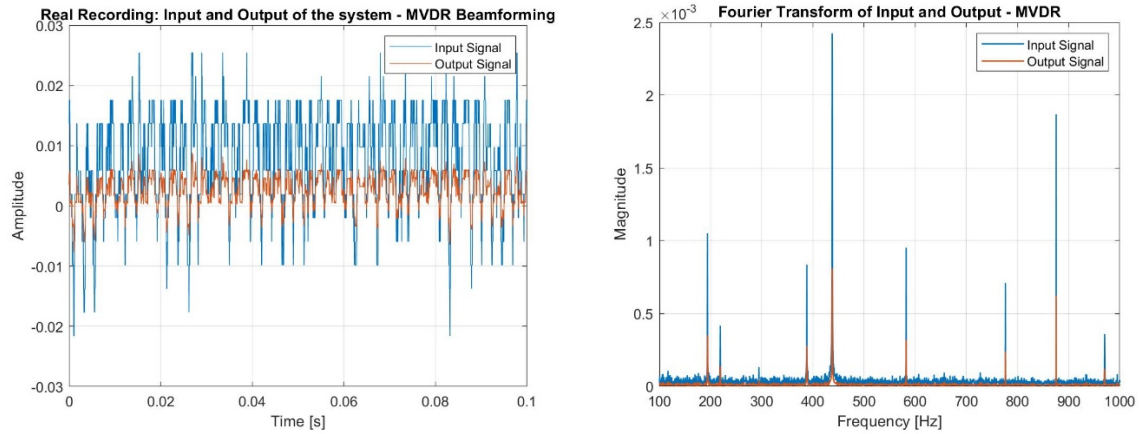


Figure 16: Graphs presenting input and output of MVDR beamformer and their Fourier transform

Results for GSC beamformer are presented below.

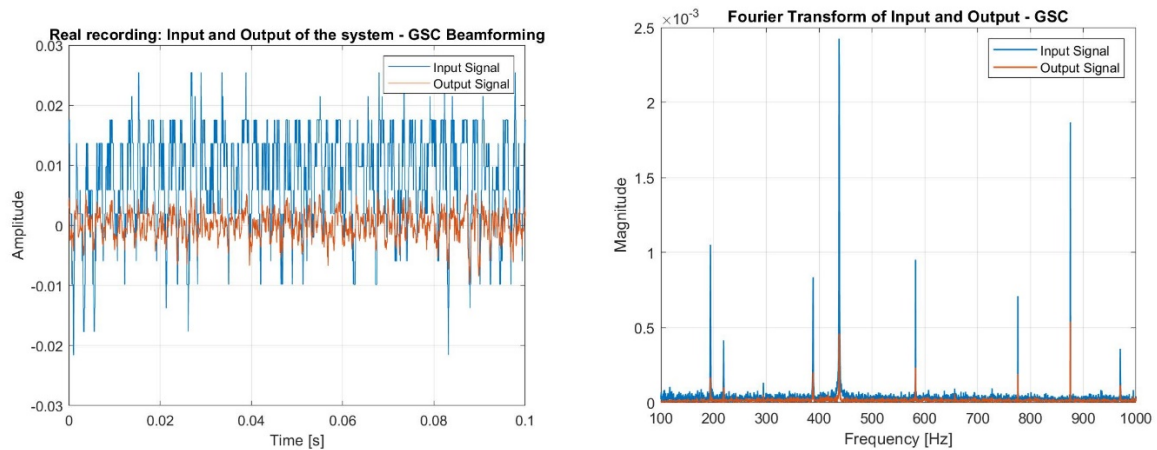


Figure 17: Graphs presenting GSC beamformer input and output and their Fourier transform

As it is clearly visible on the Fourier transform graphs, GSC outperformed other two beamformers drastically, heavily attenuating spike at 440 Hz.

Below table with noise attenuation is presented.

Beamformer	Noise attenuation (dB)
Frost beamformer	14.69
MVDR	9.57
GSC	15.75

Table 4: Table presenting results of experiments with real audio signals

6.4 Analysis and discussion

As it was shown in the previous section of the report Frost beamformer outperformed other two beamformers in a simulated environment. Performing beamforming on recorded audio signals using the designed hardware led to completely different results.

It is worth observing that the difference in beamformers' performance decreased. Particularly big difference in performance during simulations was observed between Frost beamformer and MVDR. This difference decreased now to around 5.12 dB compared to previously measured

6.3 dB. This change is not outstanding, however it may indicate that in a presence of array imperfections or mismatches in direction of the desired signal, those differences decrease.

It is worth noticing that in contrast to results of papers explored in the Literature Review section of the report, MVDR beamformer's performance was the worst, both in simulated and real environment. Therefore comment made in that section, that MVDR may perform better on smaller arrays compared to GSC, is invalid. This may be due to a simpler design of an MVDR beamformer. MVDR beamformer does not pre-steer the array, in contrast to other the two algorithms. This may increase influence of mismatches in the direction of signal of interest. It is worth noticing that majority of papers explored in the Literature Section performed analysis of the beamformers in simulated setting.

GSC outperformed other two beamformers when analysing real audio recordings. However, noise attenuation of GSC is close to the one exhibited by Frost beamformer. This may be explained by their similar design. They both work on a basis of LMS algorithms, one constrained, the other unconstrained. However, those results are also contradictory to papers explored in Section 4 as it was proven that the GSC is extremely sensitive to array imperfections, which, as shown, is not the case here. Its noise attenuation decreased by only 2 dB in comparison with simulations.

Unfortunately, a big influence of quantisation noise can be observed both on graphs presented in the last section as well as during subjective evolution of audio signals. This might also influence performance of tested algorithms.

It was proven that beamforming can be successfully implemented on a low-cost hardware consisting of only 3 microphones. In order to develop the system for professional hearing aids ADC with higher resolution should be implemented. This would approach would decrease the quantisation noise as well as improve the performance of algorithms further. Moreover, the beamforming algorithms should be implemented on a portable machine such as smartphone and data transfer should be realised using Bluetooth protocol.

7 Conclusion and future work

Three different beamformers were compared and possibility of implementing them on a low-cost hardware consisting of only 3 microphones was investigated. Firstly, the algorithms were tested in a simulated environment on a simulated microphone array in Matlab consisting of 3 microphones spaced by 5 centimetres, using recordings from Splice audio database. In this experimental setup Frost beamformer majorly outperformed other two beamformers. Its noise reduction was calculated to be 21.16 dB. The difference to other two beamformers, MVDR and GSC is large as their gain were calculated to be 14.86 dB and 17.51 dB respectively. Moreover, subjective evaluation showed noise reduction. GSC exhibited better performance to MVDR beamformer which is in contradiction to the results of scientific papers explored in the Literature Review section. In the second part of the project, physical microphone array was designed and manufactured on a PCB. The designed included 3 omni-directional microphones spaced 5 centimetres apart. MSP432P401R microcontroller was used to capture and transmit audio data from microphones to the computer running Matlab software. Again, the same 3 different beamforming algorithms were tested. In the case of physical microphone array GSC outperformed other two beamformers. Its noise attenuation was calculated to be 15.75. It was

shown that the beamforming with GSC can be successfully implemented on a low-cost, low-complexity system.

7.1 Future work

Future work will focus on developing the microphone array that could be used with professional hearing aids. This will include improving ADC resolution to at least 16 bits, which will improve performance of the beamforming algorithms and reduce quantisation noise. Moreover, beamforming algorithms should be implemented on a portable device such as smartphone instead of computer. Furthermore, data transfer should be realised using wireless transfer protocol such as Bluetooth.

Machine learning could be also implemented to reduce steering error of the array. Machine learning algorithms could scan the noisy environment, searching for the desired signal and estimate its direction of arrival. This would significantly improve performance of the beamforming algorithms as the steering error has big influence of the final outcome.

In order to further improve performance of the beamformers diagonal loading technique could be implemented. It is hard to choose a suitable diagonal loading factor, therefore machine learning could be also implemented to estimate that value and improve robustness of the beamformers.

7.2 References

- [1] L. M. Haile *et al.*, “Hearing loss prevalence and years lived with disability, 1990–2019: findings from the Global Burden of Disease Study 2019,” *Lancet*, vol. 397, no. 10278, pp. 996–1009, 2021, doi: [https://doi.org/10.1016/S0140-6736\(21\)00516-X](https://doi.org/10.1016/S0140-6736(21)00516-X).
- [2] A. Bronkhorst, “The Cocktail Party Phenomenon: A Review of Research on Speech Intelligibility in Multiple-Talker Conditions,” *Acta Acust. united with Acust.*, vol. 86, pp. 117–128, 2000.
- [3] L. A. J. Reiss and M. R. Molis, “An Alternative Explanation for Difficulties with Speech in Background Talkers: Abnormal Fusion of Vowels Across Fundamental Frequency and Ears,” *J. Assoc. Res. Otolaryngol.*, vol. 22, no. 4, pp. 443–461, 2021, doi: 10.1007/s10162-021-00790-7.
- [4] H. Dillon, *Hearing Aids*. Thieme, 2012, pp. 532 - 534.
- [5] J. Nix, M. Kleinschmidt, and V. Hohmann, “Computational scene analysis of cocktail-party situations based on sequential Monte Carlo methods,” in *The Thrity-Seventh Asilomar Conference on Signals, Systems Computers, 2003*, 2003, vol. 1, pp. 735-739 Vol.1, doi: 10.1109/ACSSC.2003.1292011.
- [6] B. Widrow, “A microphone array for hearing aids,” *Circuits Syst. Mag. IEEE*, vol. 1, pp. 26–32, 2002, doi: 10.1109/7384.938976.
- [7] World Health Organisation, “Guidelines for hearing aids and services for developing countries,” 2004.
- [8] J. E. Hawkins, “Human Ear,” *Encyclopedia Britannica*. [Online]. Available: <https://www.britannica.com/science/ear>.
- [9] D. Purves and S. M. Williams, *Neuroscience. 2nd edition*. Sinauer Associates 2001, 2001.

- [10] S. V. H. Robert A. Dobie, *Hearing Loss: Determining Eligibility for Social Security Benefits*. Washington (DC), 2004.
- [11] W. E. Brownell, "HOW THE EAR WORKS - NATURE'S SOLUTIONS FOR LISTENING," *Volta Rev.*, vol. 99, no. 5, pp. 9–28, 1997, [Online]. Available: <https://pubmed.ncbi.nlm.nih.gov/20585407>.
- [12] T. Zhang, F. Mustiere, and C. Michey, "Intelligent hearing aids: The next revolution," in *2016 38th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*, 2016, pp. 72–76, doi: 10.1109/EMBC.2016.7590643.
- [13] V. Hamacher *et al.*, "Signal Processing in High-End Hearing Aids: State of the Art, Challenges, and Future Trends," *EURASIP J. Adv. Signal Process.*, vol. 18, 2005, doi: 10.1155/ASP.2005.2915.
- [14] B. Widrow and M. Lehr, "Directional hearing system," US5793875A, 1997.
- [15] R. W. McCreery, R. A. Venediktov, J. J. Coleman, and H. M. Leech, "An evidence-based systematic review of directional microphones and digital noise reduction hearing aids in school-age children with hearing loss," *Am. J. Audiol.*, vol. 21, no. 2, pp. 295–312, Dec. 2012, doi: 10.1044/1059-0889(2012/12-0014).
- [16] H. Levitt, "Noise reduction in hearing aids: A review," *J. Rehabil. Res. Dev.*, vol. 38, pp. 111–121, Jan. 2001.
- [17] P. Rajesh, K. Umamaheswari, P. V. Babu, and S. S. Rao, "Application of Adaptive filter in digital hearing aids for cancellation of noise," in *2015 International Conference on Communications and Signal Processing (ICCSP)*, 2015, pp. 526–530, doi: 10.1109/ICCSP.2015.7322540.
- [18] R. S. Pujar and P. N. Kulkarni, "Cascaded Structure of Noise Reduction and Multiband Frequency Compression of speech signal to Improve Speech Perception for monaural Hearing Aids," in *2019 IEEE 16th India Council International Conference (INDICON)*, 2019, pp. 1–4, doi: 10.1109/INDICON47234.2019.9030366.
- [19] T. Schneider and R. Brennan, "A multichannel compression strategy for a digital hearing aid," in *1997 IEEE International Conference on Acoustics, Speech, and Signal Processing*, 1997, vol. 1, pp. 411–414 vol.1, doi: 10.1109/ICASSP.1997.599660.
- [20] G. S. Bhat, N. Shankar, and I. M. S. Panahi, "Automated machine learning based speech classification for hearing aid applications and its real-time implementation on smartphone," in *2020 42nd Annual International Conference of the IEEE Engineering in Medicine Biology Society (EMBC)*, 2020, pp. 956–959, doi: 10.1109/EMBC44109.2020.9175693.
- [21] Y.-H. Lai, W.-Z. Zheng, S.-T. Tang, S.-H. Fang, W.-H. Liao, and Y. Tsao, "Improving the performance of hearing aids in noisy environments based on deep learning technology," in *2018 40th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*, 2018, pp. 404–408, doi: 10.1109/EMBC.2018.8512277.
- [22] H. Krim and M. Viberg, "Two Decades of Array Signal Processing Research: The Parametric Approach," *Signal Process. Mag. IEEE*, vol. 13, pp. 67–94, 1996, doi: 10.1109/79.526899.
- [23] "Arrays and Spatial Filters," in *Optimum Array Processing*, John Wiley & Sons, Ltd,

2002, pp. 17–89.

- [24] M. T. Hossein, N. Ahmed, and M. S. Hossain, “Performance Investigation of Acoustic Microphone Array Beamformer to Enhance the Speech Quality,” in *2021 International Conference on Automation, Control and Mechatronics for Industry 4.0 (ACMI)*, 2021, pp. 1–4, doi: 10.1109/ACMI53878.2021.9528271.
- [25] “Optimum Waveform Estimation,” in *Optimum Array Processing*, John Wiley & Sons, Ltd, 2002, pp. 428–709.
- [26] B. Widrow, J. McCool, and M. Ball, “The complex LMS algorithm,” *Proc. IEEE*, vol. 63, no. 4, pp. 719–720, 1975, doi: 10.1109/PROC.1975.9807.
- [27] O. L. Frost, “An algorithm for linearly constrained adaptive array processing,” *Proc. IEEE*, vol. 60, no. 8, pp. 926–935, 1972, doi: 10.1109/PROC.1972.8817.
- [28] J. Benesty, J. Chen, and Y. Huang, *Microphone Array Signal Processing*. Springer Berlin Heidelberg, 2008.
- [29] B. D. Van Veen and K. M. Buckley, “Beamforming: a versatile approach to spatial filtering,” *IEEE ASSP Mag.*, vol. 5, no. 2, pp. 4–24, 1988, doi: 10.1109/53.665.
- [30] Y. Xiao, J. Yin, H. Qi, H. Yin, and G. Hua, “MVDR Algorithm Based on Estimated Diagonal Loading for Beamforming,” *Math. Probl. Eng.*, vol. 2017, p. 7904356, 2017, doi: 10.1155/2017/7904356.
- [31] L. Zhizhong, Y. Gang, J. Bin, L. Haitao, and X. Jingfeng, “Performance Simulation and Experimental Verification of Fast Broadband MVDR Algorithm,” in *2019 IEEE 11th International Conference on Communication Software and Networks (ICCSN)*, 2019, pp. 616–620, doi: 10.1109/ICCSN.2019.8905284.
- [32] E. N. Ibrahim and E. Khalil, “Improve the robustness of MVDR beamforming method based on steering vector estimation and sparse constraint,” in *2019 International Symposium on Advanced Electrical and Communication Technologies (ISAECT)*, 2019, pp. 1–5, doi: 10.1109/ISAECT47714.2019.9069701.
- [33] J. Benesty, J. Chen, and Y. Huang, *Microphone Array Signal Processing*. 2008.
- [34] B. Li and L.-H. Zhang, “An improved speech enhancement algorithm based on generalized sidelobe canceller,” in *2016 International Conference on Audio, Language and Image Processing (ICALIP)*, 2016, pp. 463–468, doi: 10.1109/ICALIP.2016.7846528.
- [35] S. Arote and M. Deshpande, “Multichannel Speech Dereverberation using Generalized Sidelobe Canceller and Post Filter,” in *2019 5th International Conference On Computing, Communication, Control And Automation (ICCUBEA)*, 2019, pp. 1–4, doi: 10.1109/ICCUBEA47591.2019.9129323.
- [36] L. Griffiths and C. Jim, “An alternative approach to linearly constrained adaptive beamforming,” *IEEE Trans. Antennas Propag.*, vol. 30, no. 1, pp. 27–34, 1982, doi: 10.1109/TAP.1982.1142739.
- [37] K. M. Kim, I. W. Cha, and D. H. Youn, “On the performance of the generalized sidelobe canceller in coherent situations,” *IEEE Trans. Antennas Propag.*, vol. 40, no. 4, pp. 465–468, 1992, doi: 10.1109/8.138853.

- [38] Y. Eldar, "Introduction," in *Sampling Theory: Beyond Bandlimited Systems*, Y. C. Eldar, Ed. Cambridge: Cambridge University Press, 2015, pp. 1–8.
- [39] T. K. Rawat, "2. Sampling and Quantization," *Digital Signal Processing*. Oxford University Press, 2015, [Online]. Available: <https://app.knovel.com/hotlink/khtml/id:kt00UPQ7V2/digital-signal-processing/sampling-quantization>.
- [40] T. K. Rawat, "2.3.1 Aliasing or Spectrum Folding," *Digital Signal Processing*. Oxford University Press, 2015, [Online]. Available: <https://app.knovel.com/hotlink/khtml/id:kt00UPQ7Z1/digital-signal-processing/aliasing-or-spectrum>.
- [41] M. S. Vijayalakshmi S. R., "6.2.9.4 Nyquist Limit," *Embedded Vision - An Introduction*. Mercury Learning and Information, 2020, [Online]. Available: <https://app.knovel.com/hotlink/khtml/id:kt012NHPE4/embedded-vision-an-introduction/nyquist-limit>.
- [42] M. Baker, "4.5 Nyquist and Shannon - Theoretical Limits," *Demystifying Mixed Signal Test Methods*. Elsevier, 2003, [Online]. Available: <https://app.knovel.com/hotlink/khtml/id:kt0090EN5B/demystifying-mixed-signal/nyquist-shannon-theoretical>.
- [43] L. Sevgi, "10.1 Filtering: Butterworth Filters," *Practical Guide to EMC Engineering*. Artech House, 2017, [Online]. Available: <https://app.knovel.com/hotlink/khtml/id:kt011LUQC1/practical-guide-emc-engineering/filtering-butterworth>.
- [44] X. Zhang, L. Zhuang, W. Liu, H. Qi, and X. Zhang, "Acoustic beamforming through adaptive diagonal loading," in *2019 IEEE Symposium on Product Compliance Engineering - Asia (ISPCE-CN)*, 2019, pp. 1–4, doi: 10.1109/ISPCE-CN48734.2019.8958621.
- [45] H. As'ad, M. Bouchard, and H. Kamkar-Parsi, "Robust Minimum Variance Distortionless Response Beamformer based on Target Activity Detection in Binaural Hearing Aid Applications," in *2019 IEEE Global Conference on Signal and Information Processing (GlobalSIP)*, 2019, pp. 1–5, doi: 10.1109/GlobalSIP45357.2019.8969387.
- [46] N. Shankar, A. Küçük, C. K. A. Reddy, G. S. Bhat, and I. M. S. Panahi, "Influence of MVDR beamformer on a Speech Enhancement based Smartphone application for Hearing Aids," in *2018 40th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*, 2018, pp. 417–420, doi: 10.1109/EMBC.2018.8512369.
- [47] D. Marquardt and S. Doclo, "Performance Comparison of Bilateral and Binaural MVDR-based Noise Reduction Algorithms in the Presence of DOA Estimation Errors," in *Speech Communication; 12. ITG Symposium*, 2016, pp. 1–5.
- [48] N. Jablon, "Adaptive beamforming with the generalized sidelobe canceller in the presence of array imperfections," *IEEE Trans. Antennas Propag.*, vol. 34, no. 8, pp. 996–1012, 1986, doi: 10.1109/TAP.1986.1143936.
- [49] M. W. Hoffman and K. M. Buckley, "Constrained optimum filtering for multi-microphone digital hearing aids," in *1990 Conference Record Twenty-Fourth Asilomar Conference on Signals, Systems and Computers, 1990.*, 1990, vol. 1, pp. 28–, doi:

10.1109/ACSSC.1990.523294.

- [50] N. Shankar, G. S. Bhat, and I. Panahi, "Comparison and real-time implementation of fixed and adaptive beamformers for speech enhancement on smartphones for hearing study," *Proc. Meet. Acoust.*, vol. 39, no. 1, p. 55009, 2019, doi: 10.1121/2.0001314.
- [51] S. S. Priyanka, "A review on adaptive beamforming techniques for speech enhancement," in *2017 Innovations in Power and Advanced Computing Technologies (i-PACT)*, 2017, pp. 1–6, doi: 10.1109/IPACT.2017.8245048.
- [52] E. A. P. Habets, J. Benesty, S. Gannot, P. A. Naylor, and I. Cohen, "On the application of the LCMV beamformer to speech enhancement," in *2009 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics*, 2009, pp. 141–144, doi: 10.1109/ASPAA.2009.5346463.
- [53] S. Markovich, S. Gannot, and I. Cohen, "A comparison between alternative beamforming strategies for interference cancelation in noisy and reverberant environment," in *2008 IEEE 25th Convention of Electrical and Electronics Engineers in Israel*, 2008, pp. 203–207, doi: 10.1109/EEEI.2008.4736688.
- [54] Š. J. G. and G. Svante, "Guidelines for Selecting Microphones for Human Voice Production Research," *Am. J. Speech-Language Pathol.*, vol. 19, no. 4, pp. 356–368, Nov. 2010, doi: 10.1044/1058-0360(2010/09-0091).
- [55] KEPO, "KPCM-G60H50P datasheet." [Online]. Available: https://static.rapidonline.com/pdf/710795_da_en_01.pdf.
- [56] Maxim Integrated, "MAX4466." 2012. Available: <https://www.maximintegrated.com/en/products/analog/audio/MAX4466.html>

Appendices

A Code for the MSP microcontroller

```
#include "msp.h"
#include <stdint.h>
#include <stdio.h>

int result1;
int result2;
int result3;

int main(void)
{
    // Stop watchdog timer
    WDT_A->CTL = WDT_A_CTL_PW |
                WDT_A_CTL_HOLD;

    // Configure GPIO
    P5->SEL1 |= BIT5 | BIT4 | BIT2;    // Enable A/D channel A0-A2
    P5->SEL0 |= BIT5 | BIT4 | BIT2;

    // Enable global interrupt
    __enable_irq();

    // Enable ADC interrupt in NVIC module
    NVIC->ISER[0] = 1 << ((ADC14_IRQn) & 31);

    // Turn on ADC14, change resolution and other important parameters
    ADC14->CTL0 = ADC14_CTL0_ON |
                ADC14_CTL0_MSC |
                ADC14_CTL0_SHT0__192 |
                ADC14_CTL0_SHP |
                ADC14_CTL0_CONSEQ_3 |
                ADC14_CTL0_SSEL_3;
    ADC14 -> CTL1 = ADC14_CTL1_RES_0;

    //set ADC inputs
    ADC14->MCTL[0] = ADC14_MCTLN_INCH_0;
    ADC14->MCTL[1] = ADC14_MCTLN_INCH_1;
    ADC14->MCTL[2] = ADC14_MCTLN_INCH_3 | ADC14_MCTLN_EOS;

    // Enable ADC14IFG.2
    ADC14->IER0 = ADC14_IER0_IE2;

    //Configure Flash wait-state to 1 for both banks 0 & 1
    FLCTL->BANK0_RDCTL = (FLCTL->BANK0_RDCTL & ~(FLCTL_BANK0_RDCTL_WAIT_MASK)) |
                        FLCTL_BANK0_RDCTL_WAIT_1;
    FLCTL->BANK1_RDCTL = (FLCTL->BANK0_RDCTL & ~(FLCTL_BANK1_RDCTL_WAIT_MASK)) |
                        FLCTL_BANK1_RDCTL_WAIT_1;

    //configure clocks
    CS->KEY = CS_KEY_VAL;                // Unlock CS module for register
access
```

```

CS->CTL0 = 0; // Reset tuning parameters
CS->CTL0 = CS_CTL0_DCORSEL_4; // Set DCO

CS->CTL1 = CS_CTL1_SELA_2 | // Select ACLK = REFO
           CS_CTL1_SELS_3 | // SMCLK = DCO
           CS_CTL1_SELM_3; // MCLK = DCO

CS->KEY = 0; // Lock CS module from unintended
accesses

// Configure UART pins
P1->SEL0 |= BIT2 | BIT3; // set 2-UART pin as secondary
function

// Configure UART
EUSCI_A0->CTLW0 |= EUSCI_A_CTLW0_SWRST; // Put eUSCI in reset
EUSCI_A0->CTLW0 = EUSCI_A_CTLW0_SWRST | // Remain eUSCI in reset
                 EUSCI_B_CTLW0_SSEL_SMCLK; // Configure eUSCI clock source
for SMCLK
EUSCI_A0->BRW = 1; // 12000000/16/9600
EUSCI_A0->MCTLW = (3 << EUSCI_A_MCTLW_BRF_OFS) |
                 (0x55 << EUSCI_A_MCTLW_BRS_OFS) |
                 EUSCI_A_MCTLW_OS16;
//UCA0MCTLW = 6;
EUSCI_A0->CTLW0 &= ~EUSCI_A_CTLW0_SWRST; // Initialize eUSCI

// Wake up on exit from ISR
SCB->SCR &= ~SCB_SCR_SLEEPONEXIT_Msk;

// Ensures SLEEPONEXIT takes effect immediately
__DSB();

while(1)
{ //wait to receive character to start conversion
  if(EUSCI_A0->RXBUF == 'S'){
    // Start conversion-software trigger
    ADC14->CTL0 |= ADC14_CTL0_ENC |
                 ADC14_CTL0_SC;
    __sleep();
    __no_operation();
  }
  //wait to receive character to end conversion
  if(EUSCI_A0->RXBUF == 'D'){
    ADC14->CTL0 &= ~ADC14_CTL0_ENC;
    ADC14->CTL0 &= ~ADC14_CTL0_SC;
  }
}

}

// ADC14 interrupt service routine
void ADC14_IRQHandler(void)
{
  if (ADC14->IFGR0 & ADC14_IFGR0_IFG2)
  {
    result1 = ADC14->MEM[0];
    result2 = ADC14->MEM[1];
    result3 = ADC14->MEM[2];
  }
}

```

```

        while(!(EUSCI_A0->IFG&0x02)) { }
        EUSCI_A0->TXBUF = result1;
        while(!(EUSCI_A0->IFG&0x02)) { }
        EUSCI_A0->TXBUF = result2;
        while(!(EUSCI_A0->IFG&0x02)) { }
        EUSCI_A0->TXBUF = result3;
    }
}

```

B Matlab Code – Frost beamformer Simulation

```

clear all

clc
close all
%defining microphone array
microphone = ...
    phased.OmnidirectionalMicrophoneElement('FrequencyRange',[20 10000]);

%specifying characteristics of the microphone array
Nele = 3;
ula = phased.ULA(Nele,0.05,'Element',microphone);
c = 340; % speed of sound, in m/s

%direction angles of the incoming signals
ang_440 = [-45; 0];
ang_880 = [30; 90];
ang_220 = [45; 0];

fs = 44000;
collector = phased.WidebandCollector('Sensor',ula,'PropagationSpeed',c, ...
    'SampleRate',fs,'NumSubbands',1000,'ModulatedInput', false);

%simulation time
t_duration = 3;
t = 0:1/fs:t_duration-1/fs;

prevS = rng(2008);
noisePwr = 0.000007;

%length of the signal
L = 3*fs;
f = fs*(0:(L/2))/L;

%defining variables
NSampPerFrame = 1000;
NTSample = t_duration*fs;
sigArray = zeros(NTSample,Nele);
voice_440 = zeros(NTSample,1);
voice_880 = zeros(NTSample,1);
voice_220 = zeros(NTSample,1);

```

```

%setting up AudioFileReader
player = audioDeviceWriter('SampleRate',fs);

file1 = dsp.AudioFileReader('440.wav', ...
    'SamplesPerFrame',NSampPerFrame);
file2 = dsp.AudioFileReader('880.wav', ...
    'SamplesPerFrame',NSampPerFrame);
file3 = dsp.AudioFileReader('220.wav', ...
    'SamplesPerFrame',NSampPerFrame);

%performing simulation
for m = 1:NSampPerFrame:NTSample
    sig_idx = m:m+NSampPerFrame-1;
    x1 = file1();
    x2 = 0.5*file2();
    x3 = 0.5*file3();

    temp = collector([x1(:,1) x2(:,1) x3(:,1)], ...
        [ang_440 ang_880 ang_220]) + ...
        sqrt(noisePwr)*randn(NSampPerFrame,Nele);
    %player(0.5*temp(:,3));
    sigArray(sig_idx,:) = temp;
    voice_440(sig_idx) = x1(:,1);
    voice_880(sig_idx) = x2(:,1);
    voice_220(sig_idx) = x3(:,1);

end

frostbeamformer = ...
    phased.FrostBeamformer('SensorArray',ula,'SampleRate',fs, ...
        'PropagationSpeed',c,'FilterLength',20,'DirectionSource','Input port',...
        'WeightsOutputPort',1);

signalsource = dsp.SignalSource('Signal',sigArray, ...
    'SamplesPerFrame',NSampPerFrame);

FrostOut = zeros(NTSample,1);
for m = 1:NSampPerFrame:NTSample
    [temp, w] = frostbeamformer(signalsource(),ang_880);
    %player(temp);
    FrostOut(m:m+NSampPerFrame-1,:) = temp;
end

%Plotting the results of the simulation
figure(1)
plot(t, sigArray(:,1)+sigArray(:,2)+sigArray(:,3))
hold on
plot(t, FrostOut)
title('Simulation: Input and Output of the system - Frost Beamforming')
legend 'Input Signal' 'Output Signal'
grid on
xlabel('Time [s]')

```

```

ylabel('Amplitude [V]')
xlim([0 0.1])

release(frostbeamformer);
reset(signalsource);

figure;
fourier = fft(sigArray(:,1)+sigArray(:,2)+sigArray(:,3));
P2 = abs(fourier/L);
P1 = P2(1:L/2+1);
fourier_frostOut = fft(FrostOut);
P2_2 = abs(fourier_frostOut/L);
P1_2 = P2_2(1:L/2+1);
plot(f,P1,'LineWidth',1)
hold on
plot(f, P1_2,'LineWidth',1)
legend 'Input Signal' 'Output Signal'
xlim([0 1000])

grid on
xlabel('Frequency [Hz]')
ylabel('Magnitude')
title('Fourier Transform of Input and Output - Frost (simulation)')

index_of_signal_frequency = find(f>877 & f<880);
P1(index_of_signal_frequency) = 0;
P1_2(index_of_signal_frequency) = 0;

pwrOfNoiseInput = sum(P1.*conj(P1));
pwrOfNoiseOutput = sum(P1_2.*conj(P1_2));

NoiseReduction = pow2db(pwrOfNoiseInput/pwrOfNoiseOutput)

```

C Matlab Code – MVDR simulations

```

clear all
clc
close all
%defining microphone array
microphone = ...
    phased.OmnidirectionalMicrophoneElement('FrequencyRange',[20 10000]);

%specifying characteristics of the microphone array
Nele = 3;
ula = phased.ULA(Nele,0.05,'Element',microphone);
c = 340; % speed of sound, in m/s

%direction angles of the incoming signals
ang_440 = [-45; 0];
ang_880 = [0; 0];
ang_220 = [45; 0];

fs = 44000;
collector = phased.WidebandCollector('Sensor',ula,'PropagationSpeed',c, ...
    'SampleRate',fs,'NumSubbands',1000,'ModulatedInput', false);

```

```

%simulation time
t_duration = 3;
t = 0:1/fs:t_duration-1/fs;

L = 3*fs;
f = fs*(0:(L/2))/L;

%defining variables
NSampPerFrame = 1000;
NTSample = t_duration*fs;
sigArray = zeros(NTSample,Nele);
voice_440 = zeros(NTSample,1);
voice_880 = zeros(NTSample,1);
voice_220 = zeros(NTSample,1);

%setting up AudioFileReader
player = audioDeviceWriter('SampleRate',fs);

file1 = dsp.AudioFileReader('440.wav', ...
    'SamplesPerFrame',NSampPerFrame);
file2 = dsp.AudioFileReader('880.wav', ...
    'SamplesPerFrame',NSampPerFrame);
file3 = dsp.AudioFileReader('220.wav', ...
    'SamplesPerFrame',NSampPerFrame);

%performing simulation
for m = 1:NSampPerFrame:NTSample
    sig_idx = m:m+NSampPerFrame-1;
    x1 = 0.5*file1();
    x2 = file2();
    x3 = 0.5*file3();

    temp = collector([x1(:,1) x2(:,1) x3(:,1)], ...
        [ang_440 ang_880 ang_220]);
    sigArray(sig_idx,:) = temp;
    voice_440(sig_idx) = x1(:,1);
    voice_880(sig_idx) = x2(:,1);
    voice_220(sig_idx) = x3(:,1);

end

signalsource = dsp.SignalSource('Signal',sigArray, ...
    'SamplesPerFrame',NSampPerFrame);

%beamformer
beamformer = phased.MVDRBeamformer('SensorArray',ula,...
    'PropagationSpeed',c, ...
    'Direction',ang_880,'WeightsOutputPort',true);

reset(signalsource);
MVDROut = zeros(NTSample,1);

for m = 1:NSampPerFrame:NTSample
    [temp,w] = beamformer(signalsource());

```

```

        temp = real(temp);
        MVDROut(m:m+NSampPerFrame-1,:) = temp;
    end

figure(1)
plot(t, sigArray(:,1)+sigArray(:,2)+sigArray(:,3))
hold on
plot(t, MVDROut)
title('Simulation: Input and Output of the system - MVDR')
legend 'Input Signal' 'Output Signal'
grid on
xlabel('Time [s]')
ylabel('Amplitude')
xlim([0 0.1])

figure;
fourier = fft(sigArray(:,1)+sigArray(:,2)+sigArray(:,3));
P2 = abs(fourier/L);
P1 = P2(1:L/2+1);
fourier2 = fft(MVDROut);
P2_2 = abs(fourier2/L);
P1_2 = P2_2(1:L/2+1);
plot(f,P1,'LineWidth',1)
hold on
plot(f, P1_2,'LineWidth',1)
legend 'Input Signal' 'Output Signal'
xlim([100 1000])

grid on
xlabel('Frequency [Hz]')
ylabel('Magnitude')
title('Fourier Transform of Input and Output - MVDR (simulation)')

%calculating noise reduction
index_of_signal_frequency = find(f>877 & f<880);
P1(index_of_signal_frequency) = 0;
P1_2(index_of_signal_frequency) = 0;

pwrOfNoiseInput = sum(P1.*conj(P1));
pwrOfNoiseOutput = sum(P1_2.*conj(P1_2));

NoiseReduction = pow2db(pwrOfNoiseInput/pwrOfNoiseOutput)

```

D Matlab Code – GSC Simulations

```

clear all
clc
close all
%defining microphone array
microphone = ...
    phased.OmnidirectionalMicrophoneElement('FrequencyRange',[20 10000]);

%specifying characteristics of the microphone array
Nele = 3;
ula = phased.ULA(Nele,0.05,'Element',microphone);
c = 340; % speed of sound, in m/s

```



```

%direction angles of the incoming signals
ang_440 = [-45; 0];
ang_880 = [0; 0];
ang_220 = [45; 0];

fs = 44000;
collector = phased.WidebandCollector('Sensor',ula,'PropagationSpeed',c, ...
    'SampleRate',fs,'NumSubbands',1000,'ModulatedInput', false);

%simulation time
t_duration = 3;
t = 0:1/fs:t_duration-1/fs;

%defining variables
NSampPerFrame = 1000;
NTSample = t_duration*fs;
sigArray = zeros(NTSample,Nele);
voice_440 = zeros(NTSample,1);
voice_880 = zeros(NTSample,1);
voice_220 = zeros(NTSample,1);

file1 = dsp.AudioFileReader('440.wav', ...
    'SamplesPerFrame',NSampPerFrame);
file2 = dsp.AudioFileReader('880.wav', ...
    'SamplesPerFrame',NSampPerFrame);
file3 = dsp.AudioFileReader('220.wav', ...
    'SamplesPerFrame',NSampPerFrame);

%performing simulation
for m = 1:NSampPerFrame:NTSample
    sig_idx = m:m+NSampPerFrame-1;
    x1 = 0.5*file1();
    x2 = file2();
    x3 = 0.5*file3();

    temp = collector([x1(:,1) x2(:,1) x3(:,1)], ...
        [ang_440 ang_880 ang_220]);
    sigArray(sig_idx,:) = temp;
    voice_440(sig_idx) = x1(:,1);
    voice_880(sig_idx) = x2(:,1);
    voice_220(sig_idx) = x3(:,1);
end

signalsource = dsp.SignalSource('Signal',sigArray, ...
    'SamplesPerFrame',NSampPerFrame);

%beamformer
beamformer = phased.GSCBeamformer('SensorArray',ula,'SampleRate',fs, ...
    'PropagationSpeed',c,'FilterLength',20,'DirectionSource','Input port');

reset(signalsource);

```

```

GSCOut = zeros(NTSample,1);
for m = 1:NSampPerFrame:NTSample
    temp = beamformer(signalsource(),ang_880);
    GSCOut(m:m+NSampPerFrame-1,:) = temp;
end

%Plotting the results of the simulation
figure(1)
plot(t, sigArray(:,1)+sigArray(:,2)+sigArray(:,3))
hold on
plot(t, GSCOut)
title('Simulation: Input and Output of the system - GSC Beamforming')
legend 'Input Signal' 'Output Signal'
grid on
xlabel('Time [s]')
ylabel('Amplitude')
xlim([0 0.1])

L = 3*fs;
f = fs*(0:(L/2))/L;
figure;
fourier = fft(sigArray(:,1)+sigArray(:,2)+sigArray(:,3));
P2 = abs(fourier/L);
P1 = P2(1:L/2+1);
fourier2 = fft(GSCOut);
P2_2 = abs(fourier2/L);
P1_2 = P2_2(1:L/2+1);
plot(f,P1,'LineWidth',1)
hold on
plot(f, P1_2,'LineWidth',1)
legend 'Input Signal' 'Output Signal'
xlim([100 1000])

grid on
xlabel('Frequency [Hz]')
ylabel('Magnitude')
title('Fourier Transform of Input and Output - GSC (simulation)')

%calculating noise reduction
index_of_signal_frequency = find(f>877 & f<880);
P1(index_of_signal_frequency) = 0;
P1_2(index_of_signal_frequency) = 0;

pwrOfNoiseInput = sum(P1.*conj(P1));
pwrOfNoiseOutput = sum(P1_2.*conj(P1_2));

NoiseReduction = pow2db(pwrOfNoiseInput/pwrOfNoiseOutput)

```

E Matlab Code – Frost

```

clear all
clc
close all

%defining microphone array
microphone = ...

```

```

    phased.OmnidirectionalMicrophoneElement('FrequencyRange',[20 10000]);

%specifying characteristics of the microphone array
Nele = 3;
ula = phased.ULA(Nele,0.05,'Element',microphone);
c = 340; % speed of sound, in m/s

%direction of source signal
ang_source = [0; 90];

fs = 39336;
collector = phased.WidebandCollector('Sensor',ula,'PropagationSpeed',c, ...
    'SampleRate',fs,'NumSubbands',1000,'ModulatedInput', false);

%simulation time
t_duration = 3;
t = 0:1/fs:t_duration-1/fs;

%defining variables
NSampPerFrame = 39336;
NTSample = t_duration*fs;
sigArray = zeros(NTSample,Nele);
voice_radio = zeros(NTSample,1);
%signal length
L = 3*fs;
f = fs*(0:(L/2))/L;

channel1 = dsp.AudioFileReader('channel1_recording.wav', ...
    'SamplesPerFrame',NSampPerFrame);
channel2 = dsp.AudioFileReader('channel2_recording.wav', ...
    'SamplesPerFrame',NSampPerFrame);
channel3 = dsp.AudioFileReader('channel3_recording.wav', ...
    'SamplesPerFrame',NSampPerFrame);

% performing simulation
for m = 1:NSampPerFrame:NTSample
    sig_idx = m:m+NSampPerFrame-1;
    x1 = channel1();
    x2 = channel2();
    x3 = channel3();

    sigArray(sig_idx,1) = x1-0.5;
    sigArray(sig_idx,2) = x2-0.5;
    sigArray(sig_idx,3) = x3-0.5;
end

%applying low pass filter
sigArrayFiltered = lowpass(sigArray, 10000,39336);

frostbeamformer = ...
    phased.FrostBeamformer('SensorArray',ula,'SampleRate',fs, ...
        'PropagationSpeed',c,'FilterLength',20,'DirectionSource','Input port');

signalsource = dsp.SignalSource('Signal',sigArrayFiltered, ...
    'SamplesPerFrame',NSampPerFrame);

```

```

%beamformer
beamformer = phased.FrostBeamformer('SensorArray',ula,'SampleRate',fs, ...
    'PropagationSpeed',c,'FilterLength',20,'DirectionSource','Input port');

reset(signalsource);
FrostOut = zeros(NTSample,1);
for m = 1:NSampPerFrame:NTSample
    temp = beamformer(signalsource(),ang_source);
    temp = real(temp);

    FrostOut(m:m+NSampPerFrame-1,:) = temp;
end

%Plotting the results of the simulation
figure(1)
plot(t, sigArray(:,1)+sigArray(:,2)+sigArray(:,3))
hold on
plot(t, FrostOut)
title('Real recording: Input and Output of the system - Frost Beamforming')
legend 'Input Signal' 'Output Signal'
grid on
xlabel('Time [s]')
ylabel('Amplitude')
xlim([0 0.1])

figure (2)
fourier = fft(sigArrayFiltered(:,1)+sigArrayFiltered(:,2)+sigArrayFiltered(:,3));
P2 = abs(fourier/L);
P1 = P2(1:L/2+1);
fourier2 = fft(FrostOut);
P2_2 = abs(fourier2/L);
P1_2 = P2_2(1:L/2+1);
plot(f,P1,'LineWidth',1)
hold on
plot(f, P1_2,'LineWidth',1)
legend 'Input Signal' 'Output Signal'
xlim([100 1000])
grid on
xlabel('Frequency [Hz]')
ylabel('Magnitude')

title('Fourier Transform of Input and Output - Frost')

%calculating noise reduction
index_of_signal_frequency = find(f>877 & f<880);
P1(index_of_signal_frequency) = 0;
P1_2(index_of_signal_frequency) = 0;

pwrOfNoiseInput = sum(P1.*conj(P1));
pwrOfNoiseOutput = sum(P1_2.*conj(P1_2));

NoiseReduction = pow2db(pwrOfNoiseInput/pwrOfNoiseOutput)

```

F Matlab Code – MVDR

```
clear all
clc
close all

%defining microphone array
microphone = ...
    phased.OmnidirectionalMicrophoneElement('FrequencyRange',[20 10000]);

%specifying characteristics of the microphone array
Nele = 3;
ula = phased.ULA(Nele,0.05,'Element',microphone);
c = 340; % speed of sound, in m/s

%direction of source signal
ang_source = [0; 90];

fs = 39336;
collector = phased.WidebandCollector('Sensor',ula,'PropagationSpeed',c, ...
    'SampleRate',fs,'NumSubbands',1000,'ModulatedInput', false);

%simulation time
t_duration = 3;
t = 0:1/fs:t_duration-1/fs;

%defining variables
NSampPerFrame = 39336;
NTSample = t_duration*fs;
sigArray = zeros(NTSample,Nele);
voice_radio = zeros(NTSample,1);

%length of the signal
L = 3*fs;
f = fs*(0:(L/2))/L;

channel1 = dsp.AudioFileReader('channel1_recording.wav', ...
    'SamplesPerFrame',NSampPerFrame);
channel2 = dsp.AudioFileReader('channel2_recording.wav', ...
    'SamplesPerFrame',NSampPerFrame);
channel3 = dsp.AudioFileReader('channel3_recording.wav', ...
    'SamplesPerFrame',NSampPerFrame);

% performing simulation
for m = 1:NSampPerFrame:NTSample
    sig_idx = m:m+NSampPerFrame-1;
    x1 = channel1();
    x2 = channel2();
    x3 = channel3();

    sigArray(sig_idx,1) = x1-0.5;
    sigArray(sig_idx,2) = x2-0.5;
    sigArray(sig_idx,3) = x3-0.5;
end
```

```

%applying low pass filter
sigArrayFiltered = lowpass(sigArray, 10000,39336);

signalsource = dsp.SignalSource('Signal',sigArrayFiltered, ...
    'SamplesPerFrame',NSampPerFrame);

%beamformer
beamformer = phased.MVDRBeamformer('SensorArray',ula,...
    'PropagationSpeed',c, ...
    'Direction',ang_source,'WeightsOutputPort',true, 'DiagonalLoadingFactor',0.5e-
3);

reset(signalsource);
MVDROut = zeros(NTSample,1);

for m = 1:NSampPerFrame:NTSample
    [temp,w] = beamformer(signalsource());
    temp = real(temp);
    MVDROut(m:m+NSampPerFrame-1,:) = temp;
end

figure(1)
plot(t, sigArray(:,1)+sigArray(:,2)+sigArray(:,3))
hold on
plot(t, MVDROut)
title('Real Recording: Input and Output of the system - MVDR Beamforming')
legend 'Input Signal' 'Output Signal'
grid on
xlabel('Time [s]')
ylabel('Amplitude')
xlim([0 0.1])

figure (2)
fourier = fft(sigArrayFiltered(:,1)+sigArrayFiltered(:,2)+sigArrayFiltered(:,3));
P2 = abs(fourier/L);
P1 = P2(1:L/2+1);
fourier2 = fft(MVDROut);
P2_2 = abs(fourier2/L);
P1_2 = P2_2(1:L/2+1);
plot(f,P1,'LineWidth',1)
hold on
plot(f, P1_2,'LineWidth',1)
legend 'Input Signal' 'Output Signal'
xlim([100 1000])
grid on
xlabel('Frequency [Hz]')
ylabel('Magnitude')

title('Fourier Transform of Input and Output - MVDR')

%calculating noise reduction
index_of_signal_frequency = find(f>877 & f<880);
P1(index_of_signal_frequency) = 0;
P1_2(index_of_signal_frequency) = 0;

pwrOfNoiseInput = sum(P1.*conj(P1));
pwrOfNoiseOutput = sum(P1_2.*conj(P1_2));

```

```
NoiseReduction = pow2db(pwrOfNoiseInput/pwrOfNoiseOutput)
```

G Matlab Code – GSC

```
clear all
clc
close all

%defining microphone array
microphone = ...
    phased.OmnidirectionalMicrophoneElement('FrequencyRange',[20 10000]);

%specifying characteristics of the microphone array
Nele = 3;
ula = phased.ULA(Nele,0.05,'Element',microphone);
c = 340; % speed of sound, in m/s

%direction of source signal
ang_source = [0; 90];

fs = 39336;
collector = phased.WidebandCollector('Sensor',ula,'PropagationSpeed',c, ...
    'SampleRate',fs,'NumSubbands',1000,'ModulatedInput', false);

%simulation time
t_duration = 3;
t = 0:1/fs:t_duration-1/fs;

%defining variables
NSampPerFrame = 39336;
NTSample = t_duration*fs;
sigArray = zeros(NTSample,Nele);
voice_radio = zeros(NTSample,1);
%signal length
L = 3*fs;
f = fs*(0:(L/2))/L;

channel1 = dsp.AudioFileReader('channel1_recording.wav', ...
    'SamplesPerFrame',NSampPerFrame);
channel2 = dsp.AudioFileReader('channel2_recording.wav', ...
    'SamplesPerFrame',NSampPerFrame);
channel3 = dsp.AudioFileReader('channel3_recording.wav', ...
    'SamplesPerFrame',NSampPerFrame);

% performing simulation
for m = 1:NSampPerFrame:NTSample
    sig_idx = m:m+NSampPerFrame-1;
    x1 = channel1();
    x2 = channel2();
    x3 = channel3();

    sigArray(sig_idx,1) = x1-0.5;
    sigArray(sig_idx,2) = x2-0.5;
    sigArray(sig_idx,3) = x3-0.5;
```

```

end

%applying low pass filter
sigArrayFiltered = lowpass(sigArray, 10000,39336);

frostbeamformer = ...
    phased.FrostBeamformer('SensorArray',ula,'SampleRate',fs, ...
        'PropagationSpeed',c,'FilterLength',20,'DirectionSource','Input port');

signalsource = dsp.SignalSource('Signal',sigArrayFiltered, ...
    'SamplesPerFrame',NSampPerFrame);

%beamformer
beamformer = phased.GSCBeamformer('SensorArray',ula,'SampleRate',fs, ...
    'PropagationSpeed',c,'FilterLength',20,'DirectionSource','Input port');

reset(signalsource);
GSCOut = zeros(NTSample,1);
for m = 1:NSampPerFrame:NTSample
    temp = beamformer(signalsource(),ang_source);
    temp = real(temp);

    GSCOut(m:m+NSampPerFrame-1,:) = temp;
end

%Plotting the results of the simulation
figure(1)
plot(t, sigArray(:,1)+sigArray(:,2)+sigArray(:,3))
hold on
plot(t, GSCOut)
title('Real recording: Input and Output of the system - GSC Beamforming')
legend 'Input Signal' 'Output Signal'
grid on
xlabel('Time [s]')
ylabel('Amplitude')
xlim([0 0.1])

figure (2)
fourier = fft(sigArrayFiltered(:,1)+sigArrayFiltered(:,2)+sigArrayFiltered(:,3));
P2 = abs(fourier/L);
P1 = P2(1:L/2+1);
fourier2 = fft(GSCOut);
P2_2 = abs(fourier2/L);
P1_2 = P2_2(1:L/2+1);
plot(f,P1,'LineWidth',1)
hold on
plot(f, P1_2,'LineWidth',1)
legend 'Input Signal' 'Output Signal'
xlim([100 1000])
grid on
xlabel('Frequency [Hz]')
ylabel('Magnitude')

title('Fourier Transform of Input and Output - GSC')

%calculating noise reduction

```



```
index_of_signal_frequency = find(f>877 & f<880);
P1(index_of_signal_frequency) = 0;
P1_2(index_of_signal_frequency) = 0;

pwrOfNoiseInput = sum(P1.*conj(P1));
pwrOfNoiseOutput = sum(P1_2.*conj(P1_2));

NoiseReduction = pow2db(pwrOfNoiseInput/pwrOfNoiseOutput)
```