Andreas Sønderland Skjong
Finn-Christian Eriksen
Torbjørn Inge Flor

Bachelor's thesis

Acoustic Condition Control

May 2020

NTNU
Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of ICT and Natural Sciences

NTNU
Norwegian University of
Science and Technology

Bachelor's thesis

2020

NTNU
Norwegian University of
Science and Technology

Andreas Sønderland Skjong
Finn-Christian Eriksen
Torbjørn Inge Flor

Acoustic Condition Control

Bachelor's thesis
May 2020

NTNU
Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of ICT and Natural Sciences

NTNU
Norwegian University of
Science and Technology

NTNU

Kunnskap for en bedre verden

Acoustic condition control

May 20, 2020

BACHELOR THESIS

Department of ICT and Natural Sciences

Norwegian University of Science and Technology

10001 - Andreas Sønderland Skjong

10072 - Finn-Christian Eriksen

10035 - Torbjørn Inge Flor

Supervisor 1: Ottar L. Osen

Summary

The purpose of the thesis is to research the fields of Acoustic Preventative Maintenance, APM. Generally, preventative maintenance is solved by high cost and low flexibility systems. These systems often have a need for multiple sensors which are directly connected to the equipment. To improve the system, the field of acoustics as a medium between software and monitored equipment was researched. The group has divided the task into four modules, where this thesis focuses on module two. By creating an APM system, the first issue is to extract data from a specific source. As preventative maintenance often happens to be in an industrial setting, noisy conditions are expected. Therefore the group has focused on removing noise, auxiliary sources and to locate sound sources. Specifically locate sound sources that are preferable within a room with numerous sources. The results in this thesis shows that acoustic preventative maintenance is possible. Topics within the fields of APM have been researched to build a system foundation. The algorithms tested by the group shows promising results as source localisation and source separation works as intended. But room impulse response and echo filtering is tested and researched, but still needs more research. With a combination of the remaining modules of the APM system, a working low-cost system is achievable.

Preface

This bachelor thesis is written by three students studying Bachelor of Science in Engineering - Automation, at NTNU Ålesund. The subject of this thesis is to research the field of acoustics data acquisition and how to process the recorded signals. This topic is mainly aimed at electronic and acoustic engineers. However, an opportunity to learn more about digital signal processing and APM presented itself, as well as a programming language we had little experience with. All of this made the bachelor thesis challenging, but in the end, the results were satisfying, and there was a lot of experience to obtain during this thesis.

We would like to thank our supervisors, Ottar Osen and Tom Jørann Giske, for being available and taking their time to guide us towards results in this thesis. We would also like to thank external mentors, e.g., our DSP tutor, Kai Erik Hoff for helping us with signal processing and MATLAB advice. Another thanks goes to Anders Sætersmoen for purchasing the equipment we needed, and advising us on what is available on the market. A thanks to Anders Ulstein who donated his time to advise us on how to improve our writing. Another thanks goes to Seaonics for letting us use their office space and giving us this thesis.

Andreas Sønderland Skjong

Finn-Christian Eriksen

Torbjørn Inge Flor

Contents

Terminology

Notation

L	Delay
dt	Time delay
c	Speed of sound waves
c_{20}	Speed of sound waves at 20 degree Celsius
l_{mic}	Microphone location
$l_{sources}$	Sound source location
l_k	Length from microphone to sound source
θ	Beamforming angle
r	Microphones radius
R	Radius source
θ_1	Beamformed angle one
θ_2	Beamformed angle two
Sk_θ	Beamformed signal in frequency domain
d	Distance
Δt	Time difference
E_i	Sum of reflected energy
E_r	Reflected energy
E_a	Absorbed energy
E_t	Transferred energy
$R_p(\omega, \phi)$	Function for incoming and the reflected amplitudal sound pressure
ω	Frequency
ϕ	angle incoming sound wave
f	Frequency denoted in kHz
R_p	Reflection factor
RT_{60}	Dissipation time
S_a	Absorption in Sabins

α_i	associated absorption coefficient
h	Humidity in percent

Abbreviations

BF	Beamforming
BSS	Blind Source Seperation
TOA	Time of Flight
DOA	Direction of Arrival
TDOA	Time Difference of Arrival
MUX	Multiplexer
UCA	Uniform Circular Array
ULA	Uniform Linear Array
GUI	Graphical User Interface
RIR	Room Impulse Response
UI	User Interface
ODAS	Open embeddeD Audition System
SRP	steered response power
APM	Acoustic Preventative Maintenance
GCC-PHAT	Generalized Cross Correlation - Phase Transform
SRP-PHAT	Steerd Response Power - Phase Transform
RPi	Raspberry Pi
EMI	Electromagnetic interference
PAST	Phased Array System Toolbox

List of Figures

List of Tables

Listings

Chapter 1

Introduction

This thesis will display a different method for preventative maintenance. By installing microphones in a room, it might be possible to obtain data on machinery condition from nearby sound sources. By recording the emitted sounds from sources in the room, it is possible to map the data and collect useful information about the condition of the equipment. This thesis will, therefore, be researching the possibility of implementing such a system.

1.1 Background

The background is to research an acoustic direction to preventative maintenance. The reason for this approach is to find a cost-efficient solution for monitoring equipment, e.g., a machine room. The company which requested this thesis, Seaonics, supply the marine and offshore market with handling and lifting equipment. The equipment, such as cranes and winches, are often a critical part of the marine and offshore operations for their customers. These systems need sensors to be able to control and monitor the machines. A skilled chief or machinist, use all their senses to survey their machines; by experience, from recurring surveys, they check if everything is normal and as expected in the actual model of the operation. Even with ear mufflers, they need to use their hearing when doing their survey.

As said before, it was requested that the focus would be on an acoustic approach with a cost-effective microphone installation. Vibration measuring sensors and equipment comes at a cost, even though it is has been a definite cost reduction for such equipment. A cheaper method that can be used in a broader range of installation could collect more experience data to do preventative actions. For Seaonics, in remote follow up, remote operation and autonomous operation, automated acoustic monitoring will add a broader picture of the process.

1.2 Problem Formulation

This thesis will give an outline of one of the modules of the APM system. Our main goal is to research if a low-cost solution to an APM system is achievable. As a result the group focuses on one of the tasks of an acoustic preventative system; how to gather data of the machinery in a noisy environment. Due to this action, the group takes the task of solving the second module in a four-part system for APM.

As seen in Figure 1.1, the group has created an example of an APM system. This system is split into four modules. The first module is about the user interface that connects the operators and techniques to the maintenance system. The second module is the part where hardware and software are combined, processing the analogue signal to convert it to data that can be later worked on. The third part is the communication link between the machines and the APM system. The fourth module is the anomaly detection or prediction estimation of the health of the system.

The group will look closer into the second module. In this module, four complications lie ahead, sound record, sound filtering, source localisation, and source separation. First, equipment and hardware need to get added together to form a prototype for research and development. Secondly, unwanted noises need to get removed by a filter to get a cleaner sound that will be easier to work with. Third, localisation of sound sources is necessary to pinpoint the position of sound sources. Fourth, a method of separating specified sound sources from other auxiliary sound emitters.

The group will also take into account the cost of the hardware used in the second module to create an APM system.

Figure 1.1: System modules

1.3 Objectives

The objectives for this thesis report are the four complications mentioned module two in Problem Formulation:

1. Sound recording. To determine the hardware and the type of configuration that would be optimal and a cost-effective APM system.
2. Sound filtering. To implementing techniques like echo cancellation, reverberation, and white noise removal to enhance audio quality.
3. Source localization. To apply ways to locate multiple sound emitters to get a location for source separation.
4. Source separation. To adopt techniques as Beamforming to focuses the receiver's angle and amplify the signal. Also implement Blind-Source-Separation to separate sound sources in the recorded input data.

1.4 Approach

The approach has been decided in the preliminary report, see Appendix A, which is a predetermined approach that holds the organisation part of the group's working conditions. The workflow of the project should follow the LEAN based model created by the group.

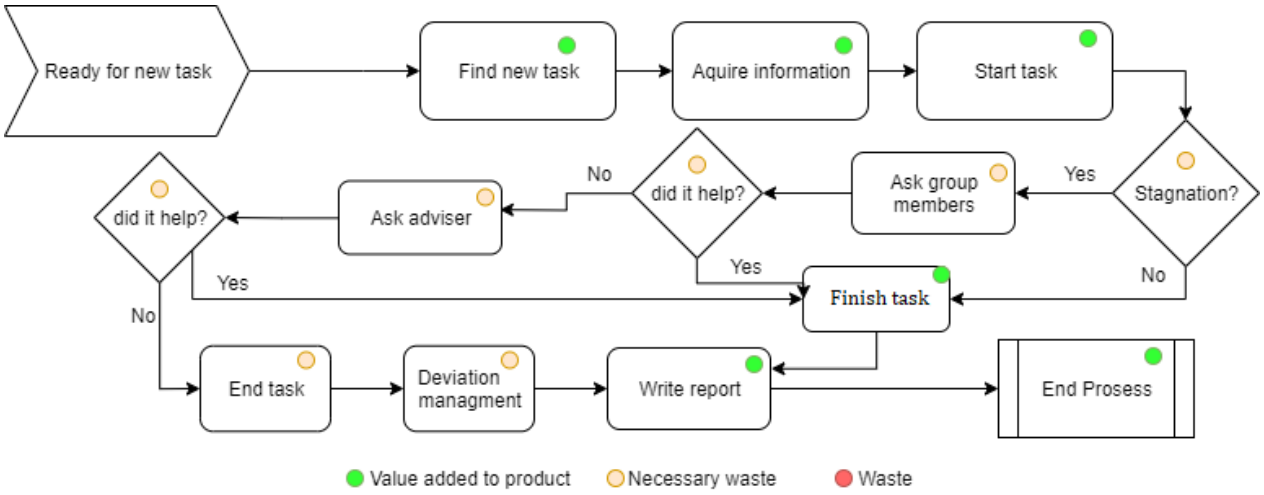


Figure 1.2: LEAN working flow

1.5 Report structure

The rest of the report is structured as follows.

Chapter 2 - Theoretical basis: Chapter two gives an introduction to the theoretical background.

Chapter 3 - Method: Chapter three contains a description of the methodology and materials that were considered.

Chapter 4 - Result: Chapter four contains a description of the finished results.

Chapter 5 - Discussion: Chapter five contains opinions of the results.

Chapter 6 - Conclusions: This chapter present an overall conclusion of the results in the thesis.

Chapter 2

Theoretical basis

2.1 Microphone types

2.1.1 Electret

The electret microphone has been around since 1962 and is the most commonly used microphone. Since it is easy to manufacture and performs, it makes it the right choice for the groups low budget. The microphone works almost like a regular condenser microphone except it does not need a bias charge; it uses a permanent charge from an electret matter. An electret is a ferroelectric matter that has been eternally electrically charged or polarized.

2.1.2 MEMS

The MEMS (Micro Electrical-Mechanical System) microphone is known for its small size, sound quality, reliability and low cost. Very tight sensitivity matching allows to optimize beamforming and noise-cancelling algorithms for multi-microphone arrays and is thus a good choice for testing microphone arrays. The MEMS is built upon the same techniques as the condenser microphones but uses a pressure-sensitive diaphragm etched directly into a silicon wafer. MEMS

microphones are used in the ReSpeaker core, which the group uses in the project.

2.2 Microphone configuration

(a) ULA configuration (b) UCA configuration

Figure 2.1: Microphone

There are multiple microphone configurations. The configuration affects the layout of the microphones are placed and how sound source detection and beamforming is calculated. Two of the most popular setups are ULA and UCA.

ULA is a one-dimensional array where the microphones are installed in a uniform, straight line, as shown in Figure 2.1a. With this configuration, it is easy to estimate the direction of the sound source, that is in the first or second quadrant. It is easier to estimate the DOA because the distance from the microphones and the centre point of the ULA is increasing for each microphone added. With ULA, it is possible to estimate the position of a source between 0 and 180 degrees.

UCA is a two-dimensional array when the microphones are installed in a uniform circular configuration, as shown in Figure 2.1b. With UCA, it is possible to estimate the position of a sound source between 0 and 360 degrees. This makes it possible to detect every sound sources around and above it.

2.3 Components

2.3.1 ReSpeaker Mic Array v2.0

The ReSpeaker is a hardware-based chip. The chip has four omnidirectional microphones, which means that they detect sound waves from all angles. The chip is intended initially as a voice-interface, e.g. IO modules. It is classified as a far-field microphone array and can detect voices up yo 5 meters away, even in noisy environments.

The board comes with pre-installed software. The software can detect several different sound sources, and the 12 RGB-LEDs lights up in the direction from which the most potent sound source originated. This means that that the ReSpeaker can detect, locate and suppress unwanted noise with beamforming and noise suppression algorithms. There is also a GUI from an open-source program which was made by the University of Sherbrooke, Canada [22]. This program is called ODAS and has a spherical representation from where different sound sources come from.

The chips compatibility and setup is well documented. The board is compatible with Linux, macOS and Windows, but Linux is the recommended operating system to use. The producer website has all the information needed to set up the device, all the documents required, support, and a forum which has FAQs and all issues you can bump into.

The ReSpeaker Mic Array v2 is a mic array with 4 MEMS microphones. It's onboard ADC has a max sample rate of 48kHz. Divided on each channel, one gets a sample rate of 16kHz each.

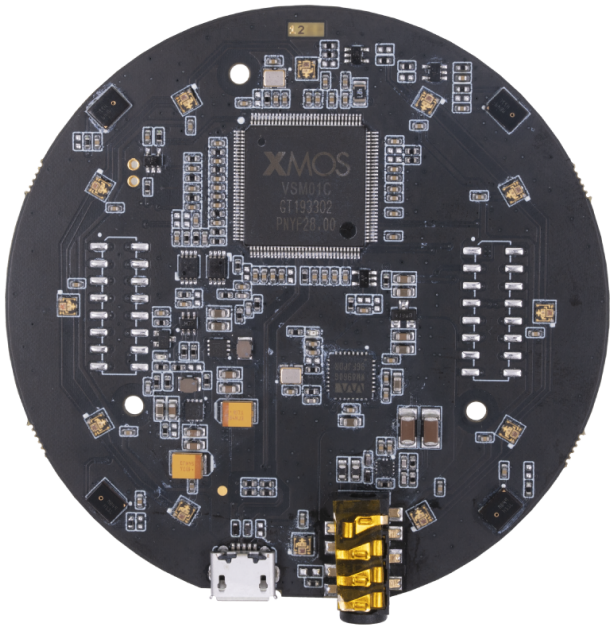


Figure 2.2: ReSpeaker Mic Array v2.0 [15]

Number of microphones	4 MEMS	ST MP34DT01TR-M
Max Sample Rate	48KHz	
Sensitivity	-26 dBFS	Omnidirectional
Mic radius	64.5mm	
SNR	61 dB	
Power Supply:	5V DC	

Table 2.1: ReSpeaker Mic Array v2.0 Specification

2.3.2 ReSpeaker Core v2.0

Seed's ReSpeaker Core v2.0 is a 6-piece microphone array. The board is designed for voice interface applications, and includes algorithms e.g. DoA, BF and AEC. The ReSpeaker Core v2.0 is micro-controller that runs on GNU or Linux, and is similar to other microphone arrays, except that is has a 1GB RAM, quad core 1.5GHz and has several communication protocols e.g. Ethernet, USB and WiFi. It is capable of recording a 16-bit resolution sound clip, and a sample rate of 96 kHz. [28].

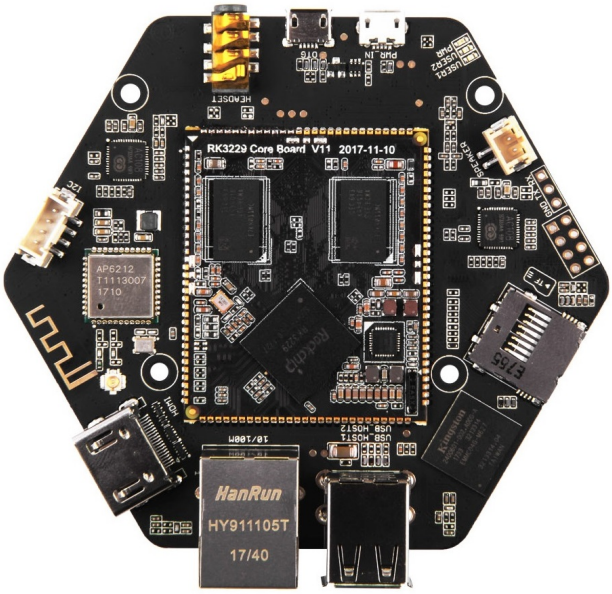


Figure 2.3: ReSpeaker Core v2.0 [13]

Number of microphones	6 MEMS	ST MP34DT01TR-M
Max Sample Rate	96KHz	
Sensitivity	-26 dBFS	Omnidirectional
Mic radius	32.3mm	
SNR	61 dB	
Power Supply:	5V DC	

Table 2.2: ReSpeaker Core v2.0 Specification

2.3.3 8-Ch 12-Bit ADC for Raspberry Pi

The ADC is a common accessory for Raspberry Pi. It is a cheap MCUs with a built-in ADC based on the STM32F030, which is a cost-effective, low-power ARM Cortex M0 MCU. It has an eight channels ADC from the MCU, and four integrated analogue Grove connectors so that it also can use analogue Grove modules with it [29].

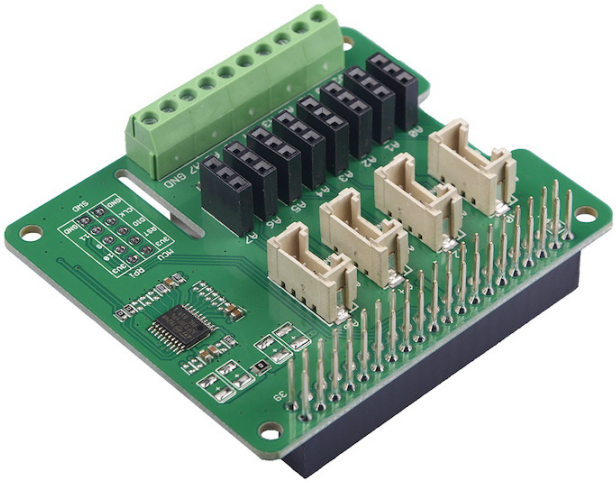


Figure 2.4: 8-Ch 12-Bit ADC for Raspberry Pi (STM32F030) [14]

2.3.4 MAX4466 microphone amplifier

The MAX4466 is an omnidirectional electret condenser microphone. The MAX4466 is a micro-power op-amp optimized for use as microphone pre-amplifiers. They provide a combination of an optimized gain-bandwidth product vs supply current and low voltage operation in ultra-small packages [21].

Figure 2.5: MAX4466 microphone amplifier [12]

2.4 Localization

2.4.1 Time-of-Arrival

One way to estimate the direction-of-arrival and the position is with TOA calculation. TOA is a way to calculate the position of a node by calculating the time differences between the signal sent, and the signal arrived from the node and multiplying the answer with the speed of sound, then comparing the results between multiple receivers to calculate the distance from the reference point to the node. This equation can do this.

$$distance = c(t_{received} - t_{transmitted}) \tag{2.1}$$

Where c is the speed of sound. When using this calculated distance from a number of receivers, a number of possible target locations can be mapped. In 2D, this yields a circle with the formula:

$$distance = \sqrt{(\Delta x)^2 + (\Delta y)^2} = \sqrt{(x_{ref} - x)^2 + (y_{ref} - y)^2} \tag{2.2}$$

where (x_{ref}, y_{ref}) is the known location of the reference spot. When enough reference spots have been calculated, a precise target position can be estimated to find a circular or spherical intersection of them all [23].

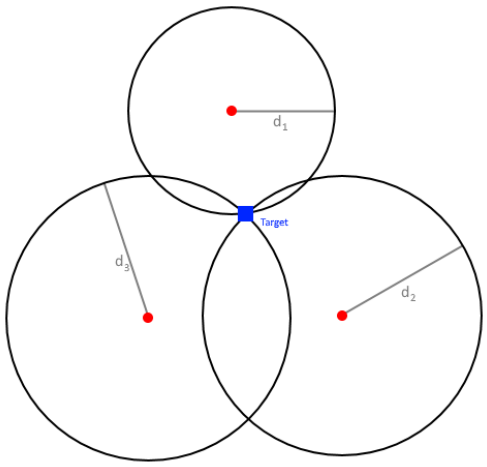


Figure 2.6: DOA estimation with TOA

2.4.2 Time Difference of Arrival

Another way to estimate DOA and position is with TDOA calculation. With TDOA it does not need the information from when the signal was sent, unlike TOA. But it lacks the knowledge of when the signal was received and at what speed the signal travel. When the signal is received, calculate the arrival time difference to get the time difference between multiple receivers and multiply it by the speed of sound to get the distance to the node [18]. The difference can be estimated by using the equation.

$$\Delta d_{n+1,n} = c\Delta T_n \tag{2.3}$$

where c is the speed of the sound travelling through the air and Δt_n is the arrival time difference at the reference point.

$$\Delta d_{n+1,1} = \sqrt{(x_s - x_{n+1})^2 + (y_s - y_{n+1})} - \sqrt{(x_s - x_n)^2 + (y_s - y_n)} \tag{2.4}$$

The conventional TDOA-based localization is a a set of hyperbolic equations such as the equation above. But this technique needs synchronisation between the receivers.

2.5 Algorithms - FFT/IFFT

A Fast Fourier Transform(FFT) is an algorithm to convert a signal from time or space domain into the frequency domain. The reason behind the conversion is to simplify the order or complexity of the targeted calculation process. The FFT technique is often in use in analysing music and its domains by breaking down complex sounds into frequency features. As the opposite of an FFT, IFFT converts a signal back from the frequency domain into the time domain.

$$Y(k) = \sum_{j=1}^{n} X(j) W_n^{(j-1)(k-1)} \tag{2.5}$$

$$X(j) = \frac{1}{n} \sum_{j=1}^{n} Y(k) W_n^{-(j-1)(k-1)} \tag{2.6}$$

$$W_n = e^{-2\pi i)/n} \tag{2.7}$$

2.6 Beamforming

Beam-forming is a technique that focuses on signal power in a specific direction. The receiving device, e.g., a microphone array, narrows in its detection field and records the signal from a particular sound source. This result is a faster and more reliable connection between the sender and receiver. The beamforming techniques have been around for a century but not used in our daily life before recent years. The method is now seen used frequently in Wi-Fi equipment and is a vital component in the coming 5G cellular network. All wave formed signals can use beamforming; this implies sound, radio, micro, and infrared waves.

$$Sn_1 = x1[n - L_1] + x2[n - L_2] + x3[n - L_3] + x4[n - L_4] + x5[n - L_5] + x6[n - L_6] \tag{2.8}$$

2.7 BSS

Blind source separation is a method to recover the original signals from a mixture of signal sources, without knowing the mixture pattern. Without the use of RIR algorithms or perfect controlled conditions makes the BSS algorithm challenging to implement into a real-world setting. Often the use of Artificial intelligence is preferred to solve problems like BSS but has an enormous cost of resources.

2.8 Echo cancellation

2.8.1 Adaptive filtering

In this thesis, microphones for information gathering are used. When using microphones in a room, there are some challenges with acoustics bouncing off surrounding walls. Challenges that do not occur when recording outside in the open air. The reason is that the sound sources emit acoustics ominously, which means in every direction. The microphone will firstly record the sound waves which have the shortest flight path, and then it records the echo that bounces off the walls. The echo will, therefore, have a small amount of delay in time and amplitude since sound travels at 343 m/s, and the amplitude weakens at the rate of 6 dB for every doubling of distance. This echo is an unwanted side effect. There are some implementations to put to good use to prevent any echo on the microphone's input. In our case, the echo cancellation is more complicated since eight microphones are being used, which means that every microphone needs its filter to cancel the echo out. This complexifies the code and takes up more resources from the micro-controller. Also, it is desirable to filter out the echo in real-time. This means that the real-time sound from the sound source needs to go into an adaptive echo filter. This, compared to running the code on a single already recorded sound clip, also takes up more computational resources from the micro-controller.

2.8.2 The room impulse response approach

Impulse response

An impulse response is a reaction of, e.g., a dynamic system and how it reacts to external change. The impulse response describes how the system reacts to changes in the time domain. This thesis is about acoustics and how the application responds to sound. So, to know how our system reacts to external acoustic changes is needed. Since sound is recorded, problems like echoes and reverberation within the room must be dealt with. First, starting with echo cancellation and how the impulse response of the room is mapped.

RIR

All rooms can have distinct room impulse responses (RIR). The RIR describes how a room reacts to external changes, e.g., with sound. The RIR is also called the room's signature, and it describes how the sound bounces off the walls until it reaches the microphones.

Echo

The first reflected sound wave is called an echo, which is a single reflection, the first reflection to return from a distant surface. The other reflections that bounce off the walls are called reverberations.

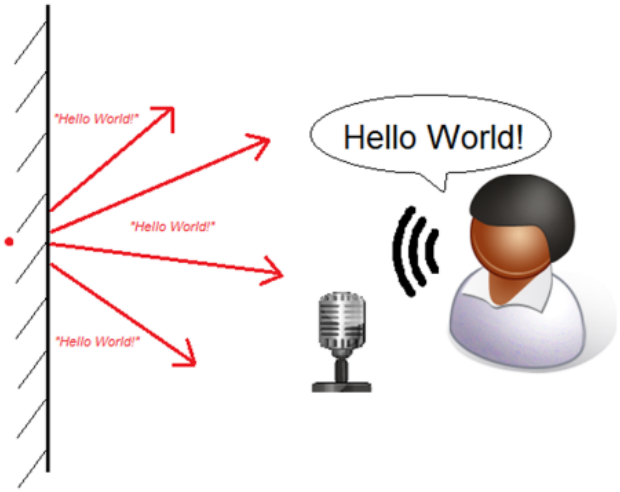


Figure 2.7: Sound reflection[32]

reverberation and the superposition principle

Reverberation is a significant acoustic property of a room. Knowing the reverberation time is essential in characterizing rooms, be they performance spaces, ordinary rooms, or open office spaces. Reverberation is a persistence of sound after the sound is produced. A reverberation is created when sound reflects numerous times between the reflective surfaces in, e.g., a room. Also called the superposition of echoes. Wave superposition of echoes is when more than one frequency adds together, creating a new frequency. Or in the case shown in figure 2.8, they

cancel each other's waves since the phase is shifted $\pi/2$ rads, and the amplitude are equal. These sound waves keep reverberating until the surroundings absorb the sound waves [20].

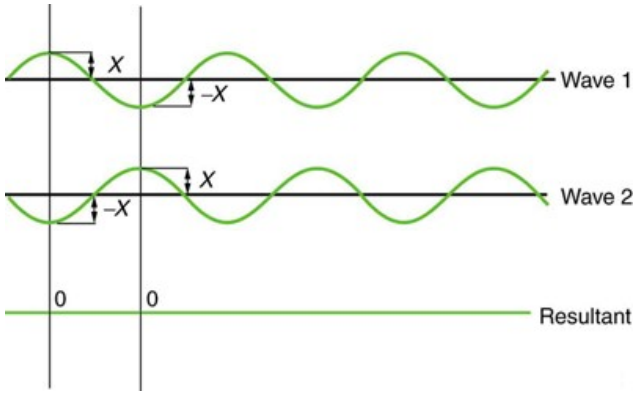


Figure 2.8: Superposition principle with two sound waves that cancels each other

Wall absorption

The absorption depends on the surrounding materials in the room. In record studios, the walls are usually covered with very absorbent material, like acoustic- foam or panels. One can even use felt cloth and Rockwool to efficiently and inexpensively eliminate noise in a room. Absorption is defined by a number between 0 and 1, and it indicates how much of the sound is absorbed compared to the reflected sound. A room with acoustic foam has an absorption coefficient close to 1, and a plaster wall is between 0.01-0.15, which makes it very reflective and therefore will affect the absorption time. Three things happen when a soundwave hits a wall. The sound wave is either reflected, absorbed, or it passes through the material. This balance in energy distribution is denoted.

$$E_i = E_r + E_a + E_t \tag{2.9}$$

The relationship between the incoming and the reflected amplitudal sound pressure is defined as the complexe pressure reflection factor.

$$R_p(\omega,\phi) = \frac{\hat{p}_r}{\hat{p}_i} = |R_p| * e^{j\delta} \tag{2.10}$$

The reflection factor is a function of the frequency, and the angle of the incoming sound wave called ϕ. Since the intensity of both the plane and spherical sound waves are proportional with the square sound pressure, the reflection factor based on intensity is given bu $|R_p|^2$. Thus the part of the incoming reflection's energy loss called the absorption coefficient α, can be denoted as

$$\alpha = 1 - |R_p|^2 \tag{2.11}$$

RT60 dissipation time

Absorption time of reverberation time is a measurement of sound decay. Reverberation time has the notation RT_{60}, and it is a time measurement of how long time it takes from the time of the first impulse to the time it takes the walls to absorb the equivalent of 60 dB. If the recorded sound is not as loud as 60 dB, one can optionally use RT_{20} instead. This means that you just multiply the RT_{20} -time by 3 to get an equivalent of RT_{60}. There are a few reasons to map the reverberation time of a room. The RT_{60} is useful when calculating the room absorption and room order automatically when placing the microphone in a new environment.

$$RT_{60dB} = \frac{24 \ln 10 \times Volume_{room}}{c_{20^\circ} \times S_{absorption}} \text{[31]} \tag{2.12}$$

Where n_sabins is calculated from experimental code in python that calculates RT_{60} with the impulse response and the sampling frequency [31]. For us to generate an RIR, the python library called "pyroomacoustics" [19] was implemented. This library creates an RIR based on the image source model (ISM). The absorption in Sabins is also given by

$$S_{absorption} = \sum_i \alpha_i * S_i \tag{2.13}$$

Is the sum of all the absorption E_i in the room and S is every wall area, and α is its associated absorption coefficient. Note that in large rooms, the total dissipation time has to take in air absorption in dB/m , and a fair approximation is given by

$$T_{60} = \frac{0,163 \times V_{room}}{S_{absorption} + 4 \times m \times V_{room}} \tag{2.14}$$

Where the air dissipation constant in dB is m, and is given by the formula

$$m = \frac{0.074}{h} \times f^2 \tag{2.15}$$

Where h is relative humidity in percent, and f is denoted as kHz. Note that the air absorption increases swiftly by an increase in frequency [31].

Image source model

ISM is a tool in geometrical room acoustics tools to simulate early reflections that bounces off surrounding surfaces. ISM can consider the absorption rate of the walls, and the directivity of the sound source and microphones. [24]. The disadvantage with the ISM, though, is the increase of the image source order proportionally affects the computation time, and when increasing the order, the computational time increases proportionally. With ISM, reflections can be mimicked by mirroring the source at the wall. The image source distance and angle concerning the receiving microphones are equal to the reflected sound path, but the angle is mirrored, and the amplitude is decreased.

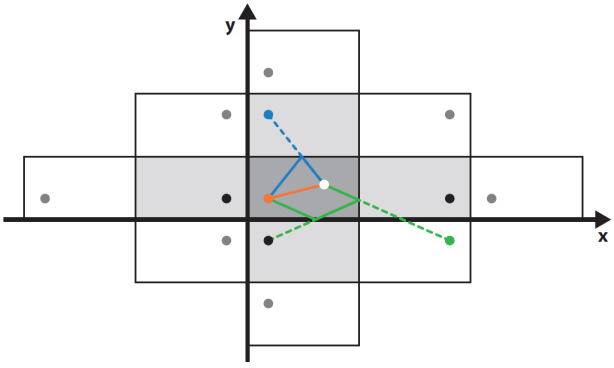


Figure 2.9: One can see several image sources of a rectangular room that is restricted to the XY-plane. The different image sources are given colours. The zeroth, first and second image order image sources are given by orange, blue and grey dots[11].

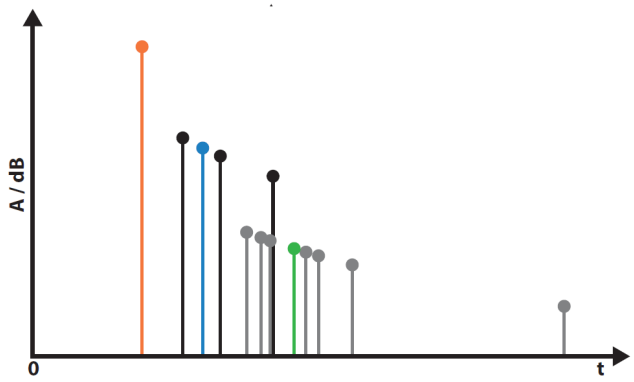


Figure 2.10: Corresponding impulse response to the image above [11].

Dry and wet sound

Wet and dry sound are terms in acoustic signal processing. Dry sound is the unprocessed audio without effects that comes directly from the sound source. Wet sound is the term if effects are added, like echo and reverb. Dry and wet sound plots are illustrated in the figured below [1].

Convolution and deconvolution

Convolution of dry signal and impulse response generates reverb. If the signal available is, e.g., recorded in a room with the absorption of 1, the walls do not reflect reverb or echo. Imagine you want to record in a studio, but also want to add the acoustics of a concert hall or a cathedral. To achieve this conversion of the sound, one can convolve the dry recorded reverb free sound and convolve the signal with the impulse response of the cathedral or concert hall. Below there are three pictures. The first, in orange 2.12, is a dry sound of a clap with no echo or reverb. The next picture, in blue 2.13a, shows the impulse response of the room. If one convolves these two $y[n] = x[n] * h$, it results in a wet sound with reverb, like the picture in red 2.13b. Note that the asterisk symbol $*$ does not mean standard multiplication.

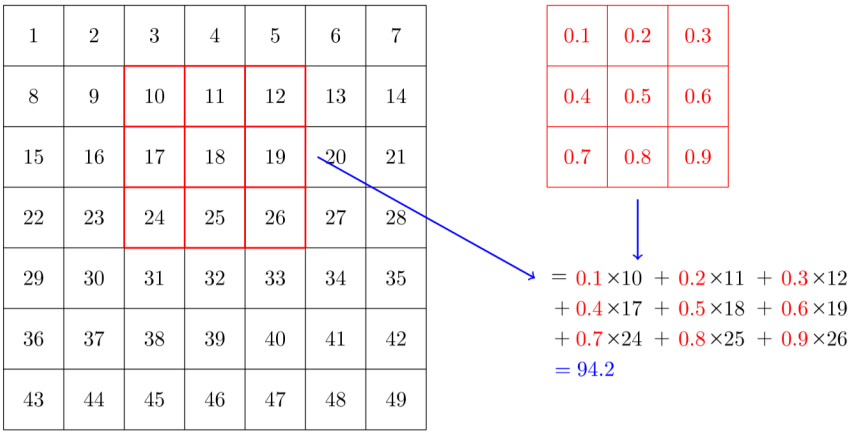


Figure 2.11: Illustration on how matrix convolution calculates [17]

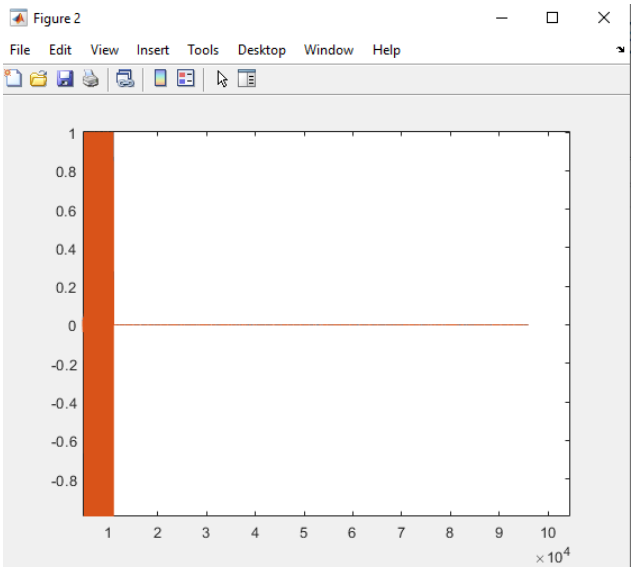
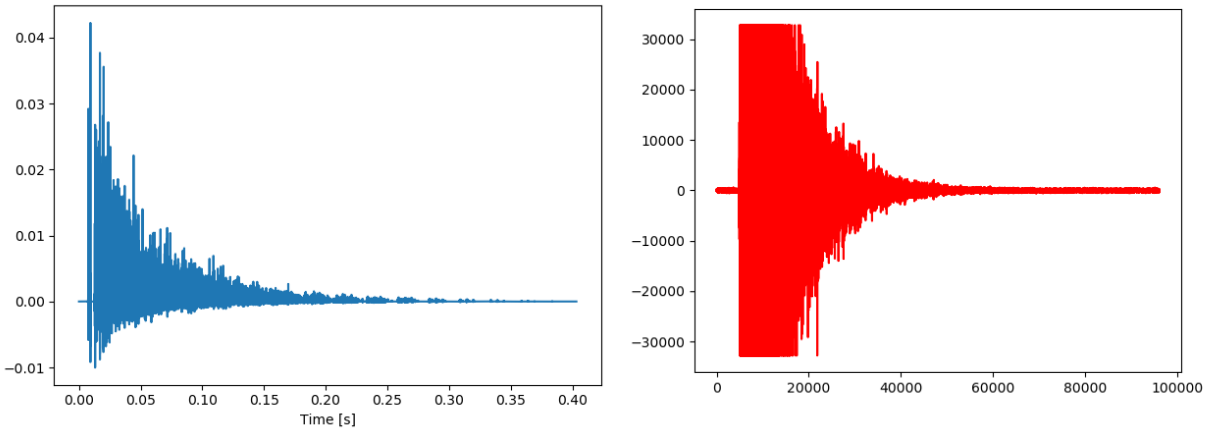


Figure 2.12: "$X[n]$" Dry single clap sound without reverb



(a) "h" Impulse response of the room

(b) "$Y[n]$" The "wet" convolved sound from the recording and the impulse response

Figure 2.13: Impulse response

Noise when deconvolving

Deconvolution does the opposite of convolution. Deconvolution is the inverse convolution that you can use to get the original reverb free sound from å sound with reverb. Since Y[n] above in red is the resultant of $X[n] * h$, the original sound $X[n] = Y[n] * h^{-1}$, but only theoretically. In real scenarios, the formula is closer to $X[n] = Y[n] * h^{-1} - \epsilon$, where epsilon is white noise that has entered the recorded signal. This may be because the microphones exert millivolts, which is sensitive when sending through wires and when close to electrical equipment. There is a different deconvolution that may improve the estimate of $X[n]$.

Wiener deconvolution with noise filtering

Wiener deconvolution improves the estimate of the original signal. By mapping what the white noise ϵ is, one can get rid of noise problems caused by the regular deconvolution. The Wiener deconvolution tries to reduce the effect of the deconvolved noise at frequencies which have low signal to noise ratio. The reason is that a low signal to noise ratio harms the deconvolution.

Chapter 3

Materials and methods

3.1 Project Organisation

The roles of each member were pre-elected in the feasibility study. The priority-based delegation of tasks in the project follows the chain of command to keep asynchronous working conditions. By keeping an organised tree-structure, the establishment helps the group in taking commands from external sources; this also keeps the tasks in the group intact and more directional focused.

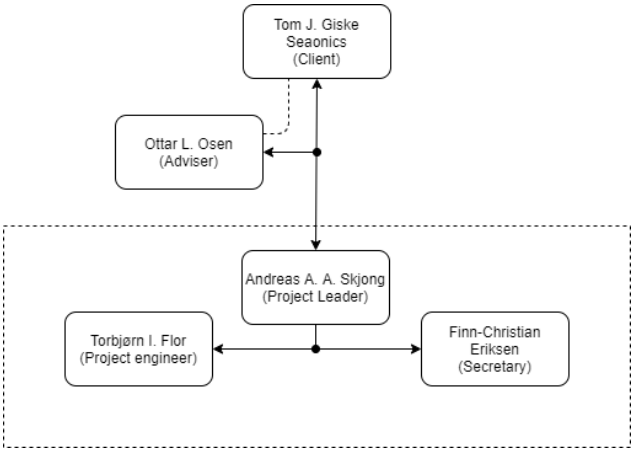


Figure 3.1: Project Structure

Routine meetings were held weekly to inform the different members of the group's organisatory tree structure. The group meetings were held Mondays to keep track of the work processes internally. Another meeting was also held every other week to keep our advisors and client informed. During this thesis, the communication between group members changed. Initially, the group worked at the same location provided at Seaonics, and the communication was transmitted directly within the group. Then the Covid-19 pandemic resulted in a relocation, so another type of communication was needed. The group then moved from a shared work space to using a digital platform called Discord. Discord, where voice, video and other media files could be shared. The change in working condition also changed other aspects of the organisation.

The unforeseen pandemic had an impact on the group's methods. The epidemic impacted how the group interacted with the projects advisor and client. With new guidelines to not interact with other people and hold a distance of two-meters apart, a new approach was at need. This resulted in a change to a digital solution to keep the counsellors informed. There was also a change of meeting frequency, which was changed from every other week to weekly meetings.

3.1.1 Data

The primary platform for the project was created as a SharePoint site. With SharePoint, every group member and advisers could access all of the projects files and keep track of the project. Other applications as Teams, OneDrive, Shift and Planner were also used. Shift is a working hour app, so each group member documents what has been done. Planner is an explanatory task manager to keep track of tasks and their progress. Teams was used to communicate between group members and to keep the apps in one place. It was therefore easy to integrate OneDrive, Shift and Planner directly into Teams.

GitHub was actively used to keep the version control of the code and other files [5]. The two main programming tools used in this project, PyCharm and MATLAB, is directly supported by Git.

3.2 Materials

The hardware selection was done by certain criteria. The hardware needed a frequency range between 20 - 20kHz, since this is the frequency range susceptible by the human ear. The next criteria was that the component had to have a sample rate that met the Nyquist sampling-theorem, which means twice as much as frequencies susceptible by the microphones. The hardware could not go above a certain price range, so it had to be relatively low cost. Lastly, the unit had to be flexible in terms of already available code.

ID	Supplier	Description	Quantity	Per piece (NOK)	Total (NOK)
301-09-501	Seeed Studio	ReSpeaker Core V2.0 [13]	1	912.00	912.00
301-09-517	Seeed Studio	ReSpeaker Mic Array V2.0 [14]	2	677.00	1354.00
301-35-129	Seeed Studio	8-Ch 12-Bit ADC for Raspberry Pi [15]	5	85.50	427.50
300-91-129	Adafruit	MAX4466 mikrofonforsterker [12]	16	68.00	1088.00
182-2096	Raspberry Pi	Raspberry Pi 4 Model B 4G SBC [27]	1	513.71	513.71
894953	JBL	JBL GO2 [26]	3	249.00	747.00
					5043.21

Table 3.1: Materials

Earlier in the project, a product called USB-AIO16-16F was taken into consideration. This is a 16 Channels USB Multi-functional Analog IO Device with a 16-Bit resolution, 1MHz sample rate. This device an AD-converter which convert a voltage to a digital signal [7].

Figure 3.2: USB-AIO16-16F

3.3 Hardware setup

3.3.1 ReSpeaker 4 Mic Array v2.0

In the beginning of the project, the ReSpeaker Mic Array v2.0 was used to record. It was first updated to the newest firmware by the instructions on the wiki page [9]. The mic array recorded with a sample rate of 16kHz on six channels, where four of them is raw microphone data streams. To increase the sample rate of the mic array the firmware was tried upgraded to 48kHz 6-channel by an experimental firmware, 48k_6_channels_firmware.bin [6].

3.3.2 8-Ch 12-Bit ADC for Raspberry Pi

A ReSpeaker ADC HAT for Raspberry Pi was bought. Along with it, the Adafruit microphones were connected. There were connectors soldered to each microphone, which were then connected to the RPi HAT. The HAT was then mounted straight onto the the RPi 24-pin connector.

The next step was to create a plate that held the HAT-configuration. A Plexiglas, that were drawn in Siemens NX 3.3, was shaped with a laser cutter, so the plate could hold the components. The plate was shaped as an ceiling tile, so it would fit into the roof frame at Seaonics, since this was a practical solution.

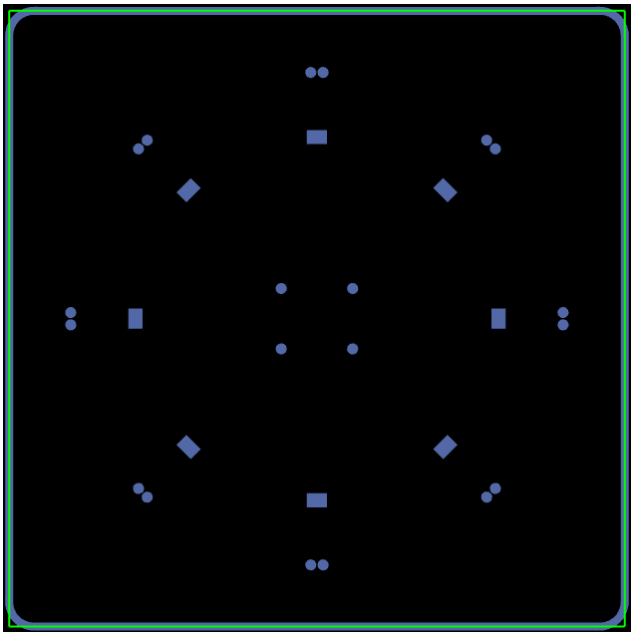


Figure 3.3: Siemens NX laser cut schematic

3.3.3 ReSpeaker Core v2.0

The ReSepaker Core v2.0 was first updated to the newest firmware and the installed recommended packages by the instructions on the wiki page packages [8]. To perform multi-channel recording with the ReSepeaker Core v2.0, two 4CH Audio ADC's samples the analogue signal from the six microphones with a set sample rate in the CLK SYNK Controler that trigger each channel in the ADC's. The analogue signals are then sent into a MUX that combines all the inputs to one output. Then the output signal can be retrieved by either a python code or directly via the terminal. It was chosen to use PyCharm to run programs on the Core instead of executing directly in the Core's terminal. This was because of the versatility and ease PyCharm. A Python script called record.py was used to record audio clips, see table 4.5.

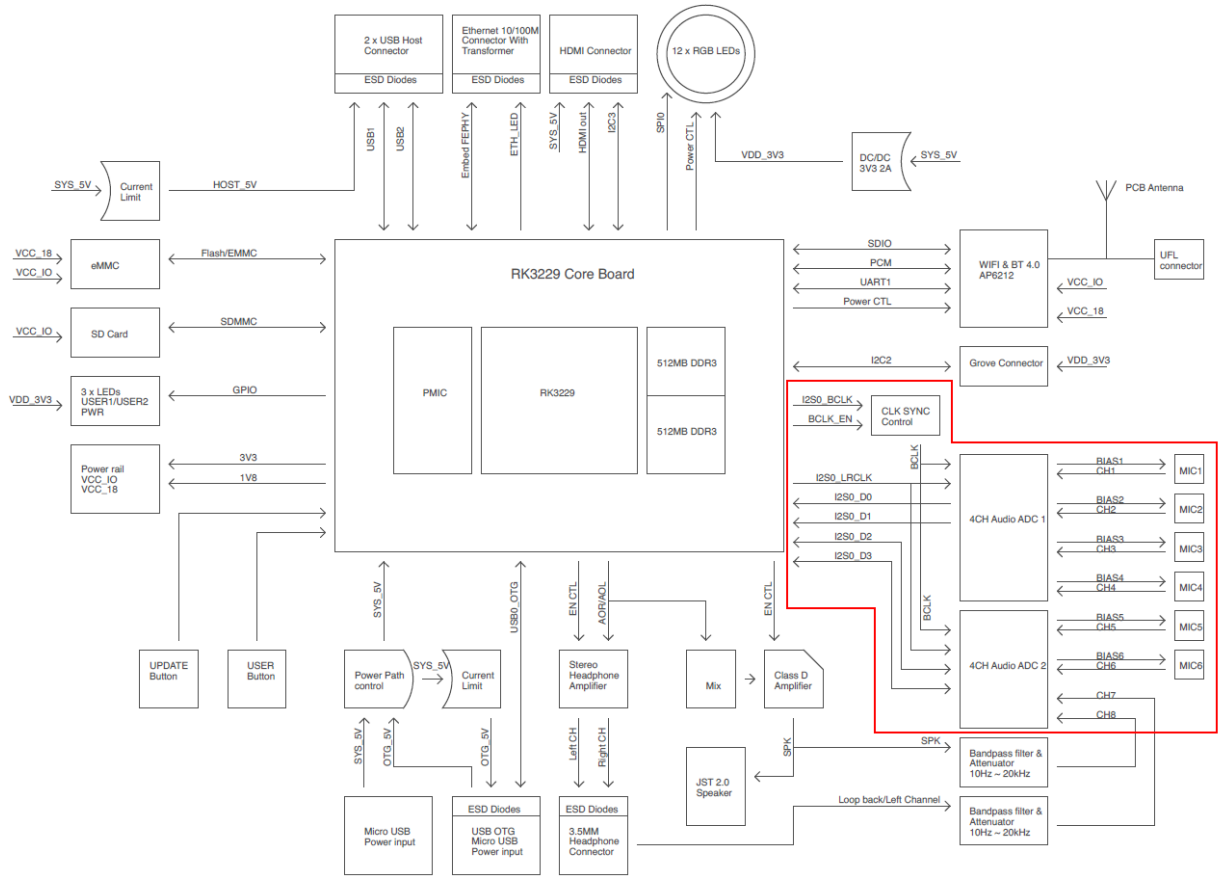


Figure 3.4: ReSpeaker Core v2.0 Schematic

3.4 Audio recording

To recording with the hardware,one initially connects to a computer with the USB interface. To extract and record sound, a python library called "pyAudio" was imported. PyAudio makes it possible to open a stream of data between the mic array and the computer.

The recorded signals are detailed and therefore large, and therefore divided up in chunks to limit the process power and memory consumption. Lastly, the data is summed up by appending the arrays into one file before saving it as a multi channelled wav-file. This was done by implementing a library called wave[10].

3.4.1 Facility

The recording facility was changed from the planned location. The new guidelines during the pandemic limited the use of the testing facilities at Seaonics. The location was changed from a machine testing facility to one of the group member's apartment. The new testing location was a 2.8m x 4.1m x 2.4m -sized bedroom with the microphone array placed on the bed.

Due to the room size and sound obstacles, the room was changed a third time. The new room's dimensions were 3.5m x 7.1m x 2.4m. A 3m circle was created on the floor, with markings on the tape-border with a resolution of $\frac{\pi}{34}$. With pre-marked angles, it could accurately record from the same location between each test.

3.4.2 Testing

During recording a wireless speaker was placed 1.5 meters away. The speaker was moved around the microphone array to create recordings of multiple angles to test DOA estimation. More recordings were created with two speakers to create test files with multiple angles to test beamforming and DOA.

The recorded files with two speakers were created by placing one speaker at 0 degrees 1.5 meters away, and moving the other speaker between the markings, which are $\frac{\pi}{34}$ in between. A 2 watt and a 1.5 watt speakers were used to simulate machines. With prerecorded sounds from machines on YouTube and music from Spotify, the two speakers could simulate sound sources. Sine waves were also used, since constant frequencies are preferred. During testing, the speakers were placed in different angles and each emitted an unique sound. The speakers used in the beginning of the testing were between 5-8 years old with a rubber membrane; this resulted in a malfunction. The speakers setup was therefore switched to three identical 3W speakers was made.

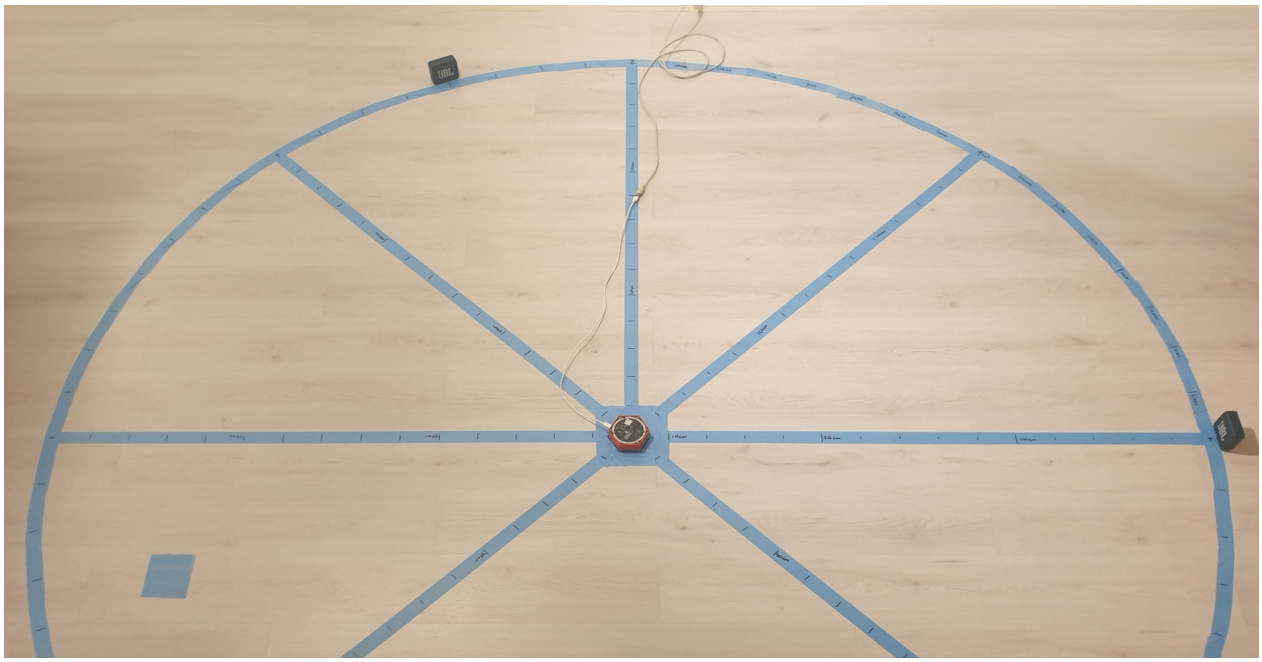


Figure 3.5: Recording room (living room)

Initially, Bluetooth was used to transmit audio signals from 2 phones to two old speakers. By using Bluetooth, synchronisation between audio signals might be difficult to achieve. The reason is that there is a small delay between each start, since starting the music simultaneously can be tough. The Bluetooth connection was replaced with a directly connected AUX cable, with Audacity to play the sounds. Three 3.5mm RCA cables were used with an RCA channel splitter to separate the left and right channel. The cable length was seven meters, from the PC to the speakers. With the new setup it was possible to play one audio clip on the left channel and another on the right channel, which finally synchronized the speakers.

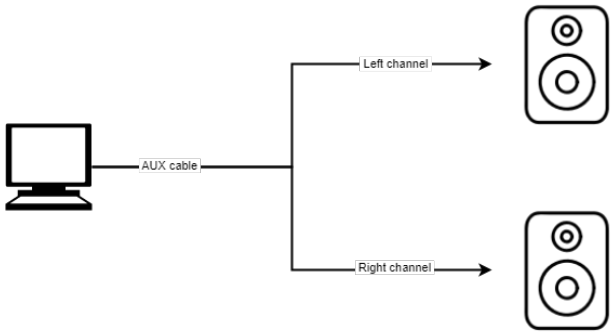


Figure 3.6: Splitted channels

3.5 Localization

An MATLAB script was created with a GCC-PHAT equation to estimate the source location. This was done by using Generalized-cross-correlation to estimate the shift of each microphone to calculate the position of the sound source.

```
% Constant variables
d = 0.0926;                          % Diameter
r = d/2;                             % Radius
M = 6;                               % Number of microphones
Tc = 20;                             % Room temperature
c = 331*sqrt(1+(Tc/273));            % Speed of sound with temperature Tc
[audiosignal,fs] = audioread(path); % The audio recording

% Generalized-Cross-Correlation
shiftMic = [];
[idx,lags] = xcorr(audiosignal(:,1),audiosignal(:,4),maxFrames);
[~, idx] = max(idx,[],1,'linear');       % Get extremal top point
shiftMic(1) = lags(idx)';
clear idx lags                           % Clear cache
[idx,lags] = xcorr(audiosignal(:,2),audiosignal(:,5),maxFrames);
[~, idx] = max(idx,[],1,'linear');       % Get extremal top point
shiftMic(2) = lags(idx)';
clear idx lags                           % Clear cache
[idx,lags] = xcorr(audiosignal(:,3),audiosignal(:,6),maxFrames);
[~, idx] = max(idx,[],1,'linear');       % Get extremal top point
shiftMic(3) = lags(idx)';
clear idx lags                           % Clear cache

% Converting the shift in samples to angle
tau = (shiftMic)/(fs*multiplier);
theta = atan(-tau*c/d);                  % DOA estimation result on rad
deg = rad2deg(theta);                    % DOA estimation result on degrees
```

Listing 3.1: DOA by using GCC-PHAT

This was later switched to using SRP-PHAT, see Section 3.7, because of multi-source source location.

3.6 Echo and Reverb

3.6.1 Adaptive filtering

An adaptive digital filter is changing its coefficients and converges to an optimal setup. The functionality for this adaptation compares the output of the filter to the desired output. Below is a adaptive filter diagram:

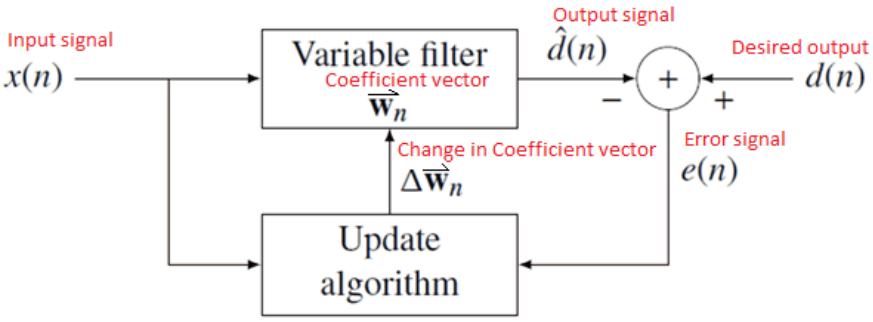


Figure 3.7: Adaptive filter diagram [25]

The diagram shows that the signal $x[n]$ is convolved with $\vec{w}_n$ to alter the output signal $\hat{d}[n]$. When the output signal $\hat{d}[n]$ is subtracted from the desired signal $d[n]$, from them the error signal $e[n]$ is obtained. $\vec{w}_n$ is a vector of coefficients, not a scalar, so the notation is not a written w_n. Since $\vec{w}_n$ changes for every n iterations, the coefficient is subscripted with n. When the error signal $e[n]$ is gathered, $\vec{w}_n$ is updated with an algorithm. If the input and output signal does not change over time, $\vec{w}_n$ will converge to the optimal filter and the output $\hat{d}[n]$ will look like the desired output $d[n]$ [25].

Echo elimination

A solution to echo cancellation can be presented in terms of an adaptive filter. An issue in creating a known optimal output from input, by locating the ideal filter that satisfies the input-output relationship. Specifically, when you fetch a headset and say "hey," the sound is received on the far end of the network, altered by the acoustic response of the surroundings, then reflected into the system and returned as an echo. However, since the system knows how "hey" initially sounded like and knows how the reverberated and delayed "hey" sounded like, the room's response can be by implementing an adaptive filter. Then the room's impulse response can be estimated, convolve all incoming signals with that impulse response, which would give an estimate of the echo signal, then subtract it from the microphone input of the individual you rang. The picture below shows another adaptive echo canceller 3.8.

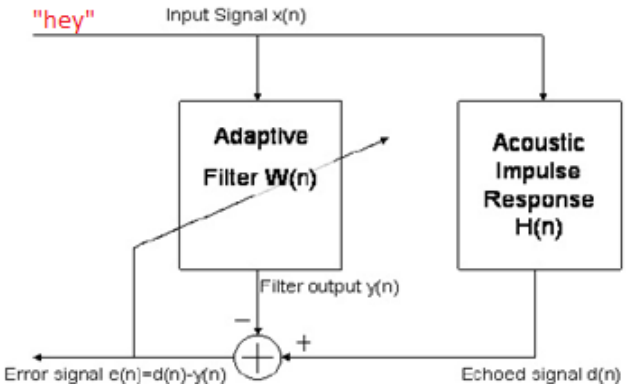


Figure 3.8: Adaptive filter diagram [25]

In this diagram, the "hey" is the input signal $x[n]$. After the "hey" is played out loud by a loudspeaker, the signal reverberates off the walls and gets picked up by a microphone and becomes the echoed signal $d[n]$. The adaptive filter $\vec{w}_n$ takes the input $x[n]$ and produces an output $y[n]$. $y[n]$ will after convergence ideally be following the echoed signal $d[n]$. Therefore $e[n] = d[n] - y[n]$ will converge to zero, given that nobody is speaking on the far-end of the line, which is usually the case when you have just picked up the headset and said, "hey." Which is not always true, and some non-ideal case consideration will be discussed later on.

Mathematically, a normalized least mean square adaptive filter can be implemented. With a

NLMS where $\vec{w}_n$ is updated each step by using the error signal $e[n]$ of the previous step.

$$\vec{x}_n = (x[n], x[n-1], x[n-2], ..., x[n-N+1])^T \tag{3.1}$$

N is the total number of samples in $\vec{w}_n$, e.g., $44100 samples/sec$.

Notice what samples of x are in reverse order.

$$\vec{w}_n = (w[0], w[1], w[2], ..., w[N-1])^T \tag{3.2}$$

Now the output $y[n]$ is calculated by convolving the inner product to find the dot product of $= \vec{x}_n$ and $= \vec{w}_n$: The only criteria is that both signals need to be real $\mathbb{R}$

$$\vec{y}_n = \vec{x}_n^T \vec{w}_n = \vec{x}_n \cdot \vec{w}_n \quad if\{\vec{x}_n, \vec{w}_n\} \in \mathbb{R} \tag{3.3}$$

the error can be calculated by using a normalized gradient descent method for minimizing it. The following update rule for $\vec{w}_n$ is as follows:

$$\vec{w}_{n+1} = \vec{w}_n + \mu \cdot \vec{x}_n \cdot \frac{e[n]}{\vec{x}_n^T + \vec{w}_n} = \vec{w}_n + \mu \cdot \vec{x}_n \cdot \frac{\vec{x}_n^T + \vec{w}_n - d[n]}{\vec{x}_n^T + \vec{w}_n} \tag{3.4}$$

Where μ is the learning rate/step size between $0 < \mu < 2$ [30].

3.6.2 Room impulse response

Acoustic signal processing library

A digital twin of the test room was created. To estimate how sound travels and gets reflected by colliding surfaces, a library called pyroomacoustics were imported [19].This library has a simulator that can mimic reverberations and add them to a sound signal to create a wet sound.

```
1 import pyroomacoustics as pra
```

Listing 3.2: Import of the pyroomacoustics library in python

Automatic absorption and max order-method – "inv_Sabine( )"

Automatic absorption and maximum order-algorithm. To install equipment in a new room, it was believed that the method was a good implementation to new rooms. Imagine having a process to calculate how the sound dissipates from the walls, and what the maximum calculation order need is. This calculation will perhaps prevent unnecessary python calculations of the RIR, and automatically calculate how much the walls absorb without looking it up online, or calculating them manually.

```
def getExperimentalAbsorptionAndRoomOrder_Sabine(t60, room_dim, c):
    """
    given desired t60, (shoebox) room dimension and sound speed,
    computes the absorption coefficient (amplitude) and image source
    order needed.
    """
    return absorption, max_order
```

Listing 3.3: Experimental code to approximate max room order and the absorption of the room

Instantiating room

There are specific parameters which are needed to calculate the rooms impulse response. The first step is to create the geometry of the walls. By adding the XY-coordinates of the corners into an array, it will give a 2D drawing of the room. Then the walls will be added as straight lines between the edges. The second step in modelling the room is to extrude the height of the walls, so that the room can be in 3D and that the generator can simulate sounds reflecting off the ceiling as well.

```
#The corners of test room
corners = np.array([[0, 0], [0, 1.8], [4.3, 1.8], [4.3, 0]]).T#[x,y]
```

```
roomHeight = 2.40
max_order = 6 #number of times the sound can reflect
room = pra.Room.from_corners(corners, fs=fs, max_order=max_order,
    absorption=absorption) #or pra.ShoeBox() method for square rooms
room.extrude(roomHeight)#Creates 3D room
```

Listing 3.4: Parameters needed to create the simulation room

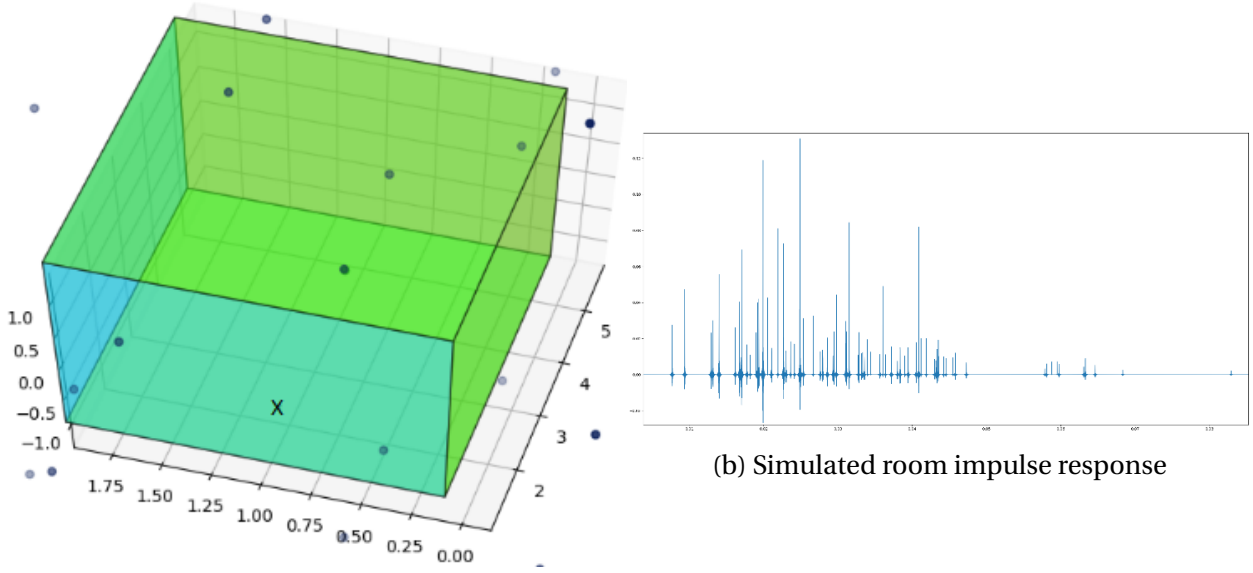
Shoebox alternative

Alternatively, one can recreate a cubical room if the room you want to simulate is just a cubical room, e.g., a 3-meter x 5 meters x 2.4 meters. A shoebox room is a parallelepipedic room with four or six walls, either in 2D or 3D, and all of the walls have 90-degree angles. The quickest method to simulate is to use the shoebox method because they are simple to define and efficient to simulate. The first parameter is simply the dimensions of the room put into an array, then attached to the room_dim= variable in the ShoeBox method. The second parameter is the sampling frequency the RIR will be generated, which is 96000 samples per second in this particular example. The next argument is the absorption of the walls. Typically these reflections are used to calculate the remaining amplitude of the signal after it reflects off a wall by the formula $Amplitude_{remaining} = 1 - absorption = 1 - 0.08 = 0.92$. Lastly, the fourth argument is the maximum number of reflections in the Image Source Model (ISM)

```
room_width = 4.3 #x length
room_length = 1.8 #y length
room_dim = [room_width, room_length, room_height]
fs = wav_file.getframerate() #Frequency samples
room = pra.ShoeBox(room_dim,
                   fs=fs,
                   absorption=absorption,
                   max_order=max_order)
```

Listing 3.5: Shoebox room parameters

Simulation plot



(a) Shoebox room

(b) Simulated room impulse response

Figure 3.9: a) The shoebox room with microphone "x" and the dots as the image sources which are reflections from the walls b) The room impulse response in regards to the position of a single microphone on room order = 5

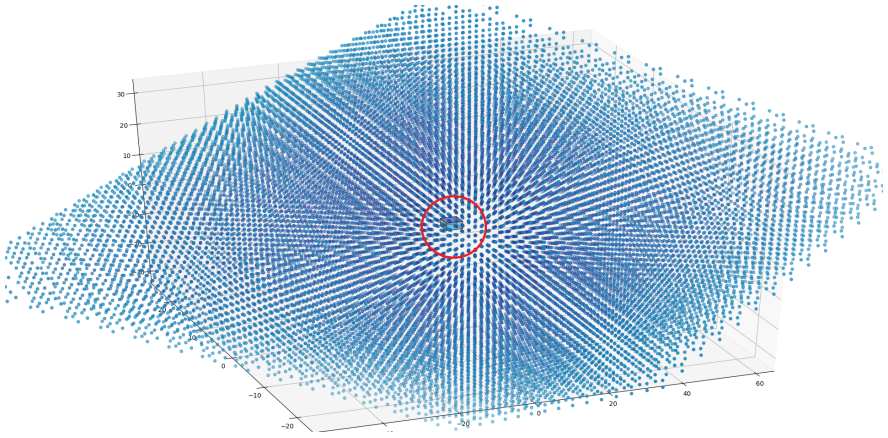


Figure 3.10: Max order set to = 30 before simulating. The red circle shows where the 3D room is plotted like in figure.3.9a Plot shows the number of wall reflections which travels outwards. All of these dots are calculated to reflect back to the microphones separately to create an impulse response.

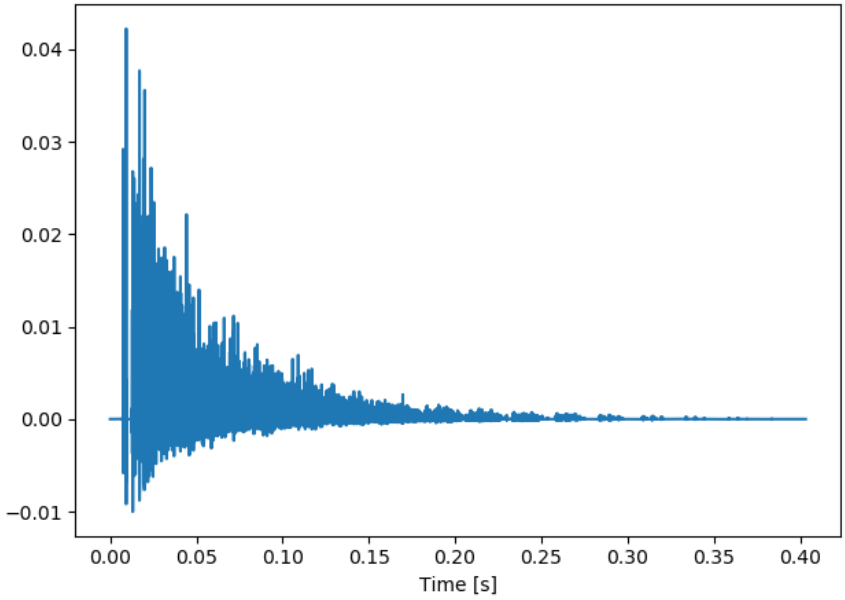


Figure 3.11: This impulse response shows how more detailed the impulse response gets when increasing the room order compared to figure 3.9b.

Attach sound signals

The next step is to add sources and what signal they emit. The method takes the location as a single argument and the sound the source emit. Optionally one can add the start time of the signal. The source location is a 2D or 3D array that must be within the coordinates of the room.

```
#Sound file location
file = "../../Recording/audiofiles/examples/echoEnhancedSounds/echoSound1.
    wav"
wav_file = wave.open(file, 'r')
soundData_Ch1 = wav_file.readframes(-1)# Extract Raw Audio from Wav
#Check if audio file is a valid stereo file
if wav_file.getsampwidth() == 2: #if it is 2 it is stereo/several channels
    soundData_Ch1 = np.frombuffer(soundData_Ch1, dtype='Int16')
else:
    raise RuntimeError("Unsupported sample width")
"""variables for source 2D and 3D locations (np.array[x, y, z])"""
 soundSource3DLoc = np.array([3.4, 0.9, 0.0])  # [X,Y,Z]
add source and set the signal to WAV file content. NB: SoundSource must ne
```

```
    np.array()
room.add_source(position=soundSource3DLoc, signal=soundData_i, delay=0.0)
```

Listing 3.6: Parameters to set where sound sources are and what sound signal it emits

On a higher level, a simulation scenario is instantiated firstly by defining a room that has a few sound sources, and a microphone array is mounted. The audio is attached to the source as raw audio samples. A RIR will be regenerated from the sounds that bounce off the walls, and when they will arrive at the microphones. Therefore, it needs to know where the sources are located so the timing, angle for the ISM will simulate accurately.

Iterated positioning of microphones

Since the microphone array's locations are placed out as uniformly spaced circular points in 2D, the site of the microphones varies on each recorded channel in the wav file. This means that the simulator needs to know the specific coordinate of the microphone that recorded the channel passing through the simulator. Therefore, the program changes the position iteratively in the code, and the timing and angle from the source are therefore correctly simulated on each channel so that a individual RIR is calculated.

```
def get_circular_2D_mic_array(center, numOfMics, angle1stMic, radius):
    """Creates an array of uniformly spaced circular points in 2D"""
    return micPosArray
#Generates room impulse responses
def createRIR(soundData_i, x_mic_coordinate, y_mic_coordinate):
    #SNIPPED CODE.
    micPosIterated = np.array([[x_mic_coordinate], [y_mic_coordinate],
                               [z_mic_center_coordinates]])
    # add one-microphone 3D XYZ array add coordinates of every mic
    room.add_microphone_array(pra.MicrophoneArray(micPosIterated,
                              room.fs)).
    #SNIPPED CODE
    def runFilter():
    #snipped code
```

```
13          i = 0
14          # micPosArray = np.array([])
15          for channel in deinterleaved:  # Get all channels
16              if i <= 0:
17                  micPosArray = get_circular_2D_mic_array(center=mic2DLoc,
                                        numOfMics=numOfMics,
18                                      angle1stMic=0, radius=radius)
19              x_coordinate = micPosArray[0, i]
20              y_coordinate = micPosArray[1, i]
21              filteredData = np.array(CreateRIR(channel, x_coordinate,
                            y_coordinate)).T
22              listFilteredSound.append(filteredData)
23              i = i + 1
```

Listing 3.7: Code iterate between and create impulse response from sound source and every microphone. A total of six RIRs are generated

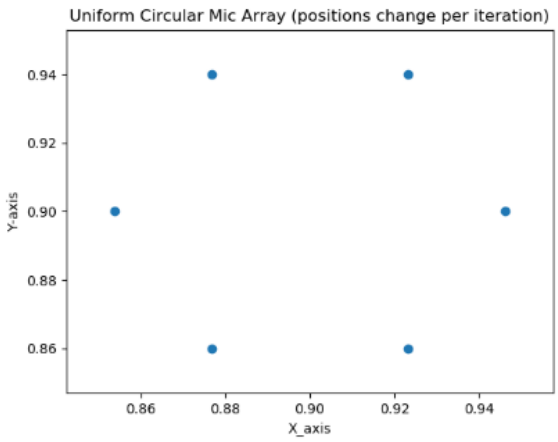


Figure 3.12: Plot of microphone positions. See animated iteration[16]. Description: First-order sound travel from a sound source to one microphone/channel. Orange is direct sound, and the two others are sounds bouncing off the walls of the room. The microphones are the six purple dots on the hexagon.

RIR simulation with ISM

When the position of image sources and their visibility from each microphone is determined, they can be used to generate the RIRs. The RIR is generated by calling the ISM-method. This method will construct all the images up to the chosen max_order and use them to create the impulse response between the sound source and microphone. For microphones placed at r, a real source s_0, and a set of visible image sources

$$a_r(s_0, n) = \sum_{s \in v_r(s_o)} \frac{(1-a)^{gen(s)}}{4\pi||r-s||} \cdot \delta_{LP}(n - F_s \frac{||r-s||}{c}) \tag{3.5}$$

Where gen(s) gives the reflection order of the source $(s, a) \in [0, 1]$ is the absorption factor of the walls, c is the speed of sound, and δ_{LP}. The windowed since function which separates frequencies. The microphone arrays are then made by convolving the audio samples associated with sources with the correct RIR. Since the simulation is done with discrete-time signals, a sampling frequency is specified for the room and the sources it contains.

RIR output

By invoking the pra.room.Room.simulate()-method, the source's recorded signal gets convoluted with the impulse responses to generate a new signal with more reverberation to simulate a more "wet" sound, "wet" means sound with more echo and reverb. The output will be summed at each of the microphones, which results in a 6-channel wave-file with more reverb than the original sound. Now that the RIR has been generated, and the original signal from the recorded signal is at hand. In theory, it might be possible to inverse-convolve the original signal with the RIR to get a reverb free sound [32]

3.7 Beamforming

In the project description, it is crucial to enhance the signals from a specific direction. Beamforming is an essential part when extracting a particular transmitted sound from the auxiliary sources. This is comparable to the human mind, where the phenomenon is called "The cocktail party effect", where humans blocks out sounds it does not want to listen to.

Beamforming is used when arrays of multiple sound transmitters are used. By manipulating the channels in the array towards a region containing the source location, the steered response will peak. SRP means a signal from a sound transmitter in the set specific location gets enhanced. A way to use beamforming is to manipulate the channels from each microphone. Compensating for the distance between them, and sum them up to form a reinforced signal. The algorithm is called delay-and-sum, which is the technique that was implemented in this project.

Since the microphones are distanced from each other, the sound signals will have separate arrival times. The delay-and-sum beamformer will compensate the difference by adding delays to the microphone channels. Once the signal is aligned, they are summed to create a single reinforced output signal. When summing the signals, white-noise in the data gets reduced. Where the summed white noise gets reduced due to its randomness. The technique can be used in every direction with a limit of array element size minus one; a unique beamformed signal is creatable for each source.

Beamforming towards multiple sources means different outputs on the same signal. The way this works is how signals arrive at the microphones. By knowing the delay from the sound source, aligning the channel to enhance one source distorts the others. This effect can be seen clearly in the illustration in figure 3.13.

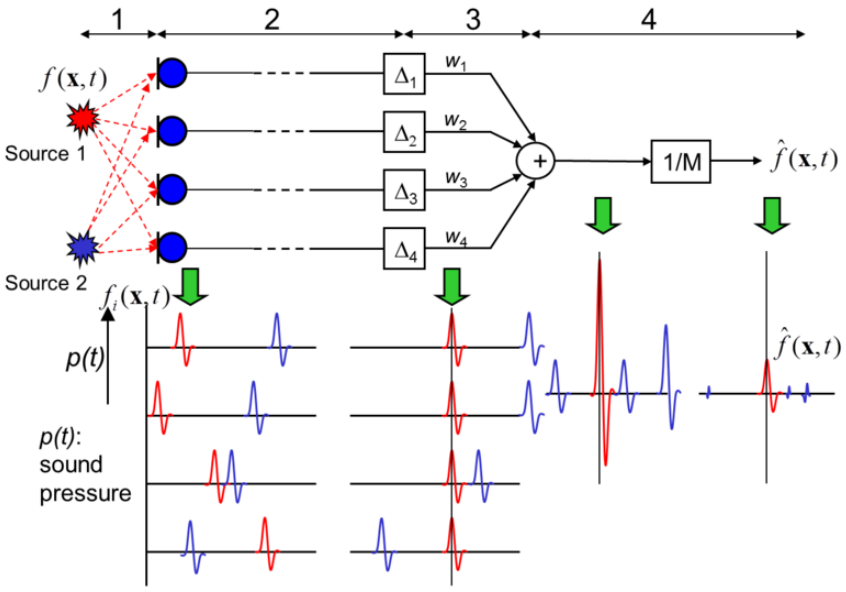


Figure 3.13: Beamforming with multiple sources

3.7.1 Starting phase / information gathering

Initially, there is a research phase where information on the subject gets gathered. Codes, libraries, theories and programs were tested. Some of theese libraries were Acoular, PyroomAcustic and Phased Array System Toolbox, PAST. The first two are libraries that can calculate beamforming algorithms in Python, and the last one is a MATLAB toolbox, where some were not applicable. Acoular, for instance, had a beamforming-algorithm, but only if the sound sources were located between the microphones. As the assignment had small microphone arrays which wanted to calculate sources from external locations placed around the microphones, Acoular was therefore not applicable. PyroomAcoustics was another library which was tested. This library was not applicable when it came to DOA, BSS and BF. The reason was that every individual sound signal had to be linked to a single source, but the sound that gets recorded is a combination several sources.

With both Acoular and PyroomAcustic tested, MATLAB's toolbox PAST was next in line. The toolbox has support for ULA, UCA and URA. It also has the functionality for beamforming, matched filtering, DOA estimation, and target detection. The toolbox was abandoned since the library

required that the microphones had to be separated by a minimum radius of 0.32m. Whereas the ReSpeaker Core v2.0 has a radius of 0.0463m.

After researching beamforming, it was decided to make an algorithm from scratch. When creating a beamformer from scratch, basic knowledge within signal processing and physics was implemented to handle the specific task. The new direction resulted in changing the IDE to MATLAB.

3.7.2 Creating a beamformer

There were tested two main approaches for the delay-and-sum beamformer. One of the methods was to calculate the cross-correlation between the channels. The second method is to calculate the delay by knowing the distance and angle from the source.

The xcorr function in MATLAB finds the highest sum of the signals. By delaying the channels between one predetermined input, the output from cross-correlation is a matrix of channels delayed against each other. By picking out the delay difference between the channels makes it possible to align the input signal in the time domain.

One of the difficulties by using xcorr when beamforming, is finding multiple sources. Since xcorr returns a matrix from a cross-correlated signal, using xcorr to find other global maximas is time consuming. Therefore, another approach was researched.

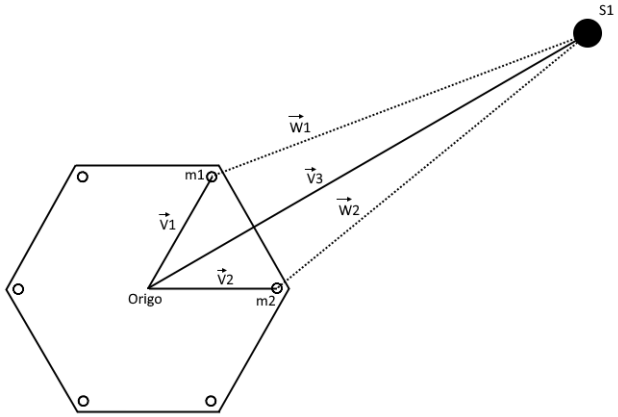


Figure 3.14: Vector displacement

```
MicPosition = [r*cos(-pi/3*micN), r *sin(-pi/3*micN)];
Source = [1.5 60]= 1.5*[cos(60) sin(60)]; %length and angle to xy-
    cordinates
LengthMicSource = Source-MicPosition;
```

Listing 3.8: Length Calculation

The first step in creating a beamformer is to calculate the distance from the sound source to the microphones. The solution was to use MATLAB for calculating vector positions of the microphones and the receiver. By predicting the angle and length to the sources, makes it possible to calculate each delay theoretically from the transmitter to each microphone. The calculation uses the speed of sound as a constant in the formula Distance= Time*Velocity. By turning the formula, the group can acquire the calculated time distortion to each microphone.

```
dt = LengthMicSource/c;
```

Listing 3.9: Length to sec

The speed of sound is at 343 meters in the second at 20 degrees Celsius, but temperature is not taken into account as a variable in this stage of the project.

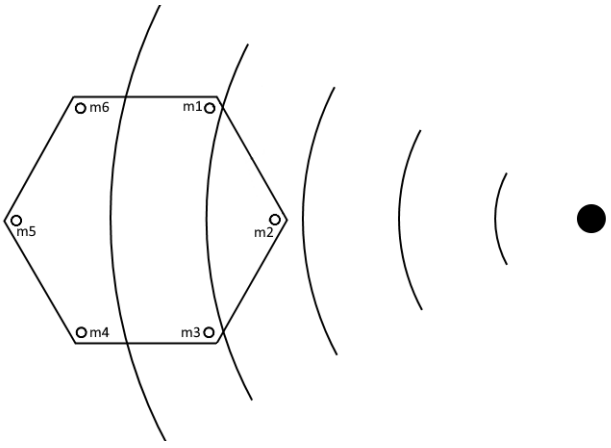


Figure 3.15: Microphone hit chronology

Time can further on be converted to samples. By delaying each channels array with zeroes according to samples delayed in time, the signals can get aligned. To work in time-domain is

preferred, as its simple to illustrate the procedure.

```
Timeshifts = round(timeDelay.*sampleFrequency);
delayedChannels = delayseq(rawData, Timeshifts);
```

Listing 3.10: Matlab delay channels

With each channel aligned, signals get added together to form the new beamformed signal. The resulting signal is then a result of the directional features from the sound source. By determining a location, it is possible to enhance the features of the signal from the targeted area. This can be used when BSS separates sources from each other. The following functions show the behaviour of the system mathematically.

$$BF(L_k) = \sum_{k=1}^{k=N} X_k[n - L_k] \tag{3.6}$$

$$L_k = dt_k fs \tag{3.7}$$

$$dt_k = \frac{l_k}{c} \tag{3.8}$$

$$l_k = l_{source} - l_{mic} \tag{3.9}$$

$$l_{source} = R\cos\theta + iR\sin\theta = Re^{i\theta} \tag{3.10}$$

$$l_{mic}(k) = r\cos(-\frac{\pi}{3}k) + ir\sin(-\frac{\pi}{3}k) = re^{-i\frac{\pi}{3}k} \tag{3.11}$$

$$BF(\theta) = \sum_{k=1}^{k=N} X_k[n - (Re^{i\theta} - re^{-i\frac{\pi k}{3}})\frac{fs}{c}] \tag{3.12}$$

3.7.3 DOA estimation

When creating the DOA estimation-formula, two approached were tested. One that is mentioned in Section 3.5, and another by using the features from the delay-and-sum beamformer. When using beamforming, DOA can be estimated by iterating between the angles 1-360. In each iteration, the gain improvement was plotted to form an SRP curve of the signal. By finding the peaks of the curve where the SRP was greatest, an angle of the most significant magnitude are

possible to extract. This lead to a simple approach to finding DOA.

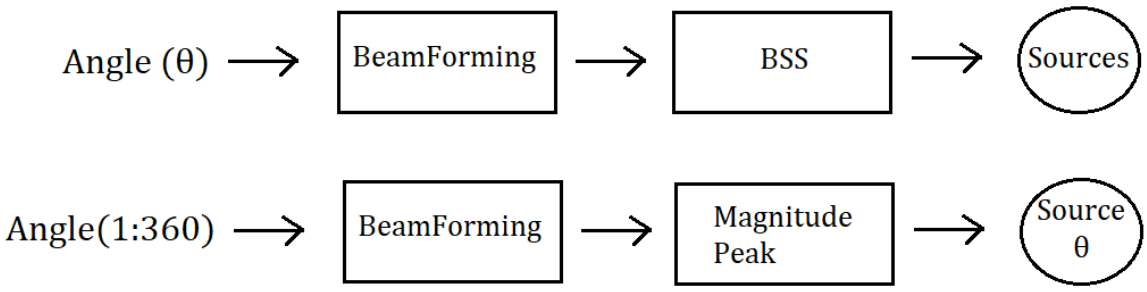


Figure 3.16: Block diagram of BSS and DOA estimation

3.7.4 BSS

As beamforming only enhances the desired source. It wont distinguish auxiliary sources in the recording. To separate or remove sources, the project needed to implement a BSS method. It was experimented then discovered a solution to perform BSS, by combining information from the beamformer.

To apply BSS, information of the beamformed sources is necessary. When beamforming towards a source, the signal from this area gets enhanced. By knowing which signal components in the particular direction get a gain, the sources were separated within the recording. Features from each source are possible to map by knowing the direction of the most significant enhancement. The BSS algorithm can be derived into four steps.

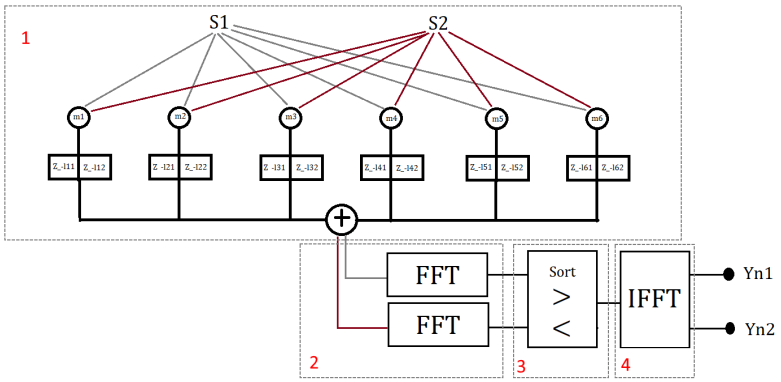


Figure 3.17: Block diagram BSS

1. Beamforming two or more sources
2. FFT the beamformed signals
3. Comparing and sort against each other
4. Inverse FFT

The group performed this by first doing a Fast Fourier Transform, FFT, of the output signal from the beamformer. By performing a FFT, the signal gets transformed to frequency components.

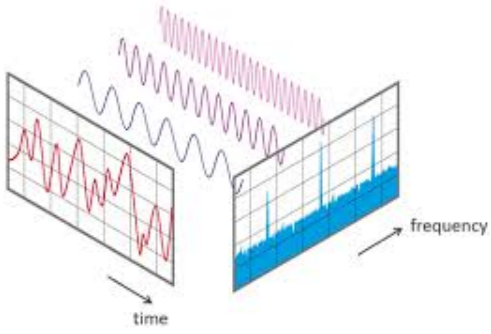
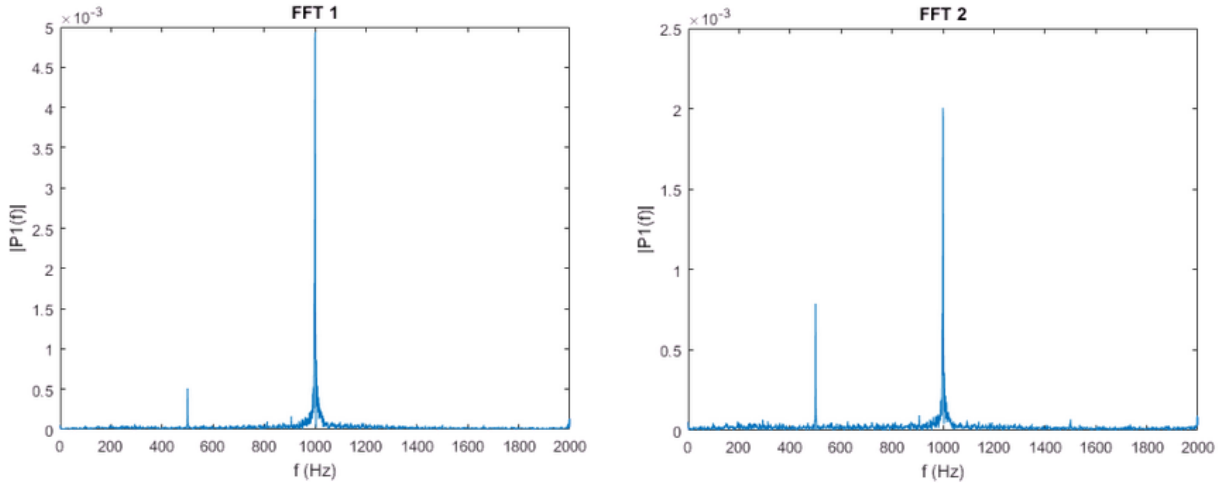


Figure 3.18: FFT

The second step was to compare the two-directional beamformed signals in the frequency plane. To have the signals in the frequency-domain makes it possible to sort each component by magnitude into directional-sources. By knowing which frequency component that's becomes enhanced, the system can separate one source from another. In Figure 3.19, an illustration of the

two beamformed signals in the frequency domain. In Figure 3.19a, an increase in the 1000Hz frequency component can be seen, but in Figure 3.19b, an identical rise in the 500Hz component can also bee seen. By knowing that Figure 3.19a is a directional beamformed signal against source one, the 1000Hz signal is concluded to belong to this direction.



(a) 1000Hz at angle 0

(b) 500Hz at angle 120

Figure 3.19: Directional beamforming

In the MATLAB code, the group did this by making a sorting algorithm of if-statements. The signal of the most significant magnitude gets to store the value in an empty array.

```
Sk1 = fft(Beam1); %FFT of the Directional beamforming against source 1
Sk2 = fft(Beam2); %FFT of the Directional beamforming against source 2

%Creation of emty list of the sorting alorythm
Yk1 = zeros(size(Sk1));
Yk2 = zeros(size(Sk2));

for i=1:length(Sk1)
    if abs(Sk1(i)) > abs(Sk2(i)) %if direction one has the greatest
        Yk1(i) = Sk1(i);         %Magnitude, its signal one.
    elseif abs(Sk2(i)) > abs(Sk1(i))
        Yk2(i) = Sk2(i);
    end
end
```

```
15
16 %inverse FFT
17 yn1= ifft(Yk1);
18 yn2= ifft(Yk2);
```

Listing 3.11: BSS

With two sets of divided frequencies, the group inversed the Fourier Transform to create output signals where the sources are divided. The code can be translated to the following mathematical equations as follows:

$$FFT(k) = \sum_{j=1}^{n} X(j) W_n^{(j-1)(k-1)} \tag{3.13}$$

$$IFFT(j) = \frac{1}{n} \sum_{k=1}^{n} Y(k) W_n^{-(j-1)(k-1)} \tag{3.14}$$

$$W_n = e^{(-2\pi i)/n} \tag{3.15}$$

$$BSS_\theta(n) = IFFT(yk_\theta) \tag{3.16}$$

$$Sk_\theta(k) = FFT(BF_\theta(j)) \tag{3.17}$$

$$yk_{\theta_1}(k) = \begin{cases} Sk_{\theta_1}(k), & \text{if } Sk_{\theta_1}(k) > Sk_{\theta_2}(k) \\ 0, & \text{otherwise} \end{cases} \tag{3.18}$$

3.8 GUI

In the preliminary report, it got decided to only create a technical graphical user application for testing and developing. The reason to create a more of a finished product and only creating a technical GUI is that this project is to test and understand the functionalities around using sound-based maintenance systems.

The GUI application first was created with the Python interface GUI toolkit TkInter. After working on the design and adding graphics that could be useful to display the information, it got decided to wait with the GUI until the basics of the code were working.

Because of the ease of use and graphical visualisation of data in MATLAB, it got decided to continue the rest of the code with MATLAB, version 2020a. Since the GUI was not finished in Python, a new GUI was created in MATLAB. It was created by using MATLAB's UI-Control modules. With this interface, it was possible to add buttons, sliders, graphs and other user controls. It also has support for Java which allows creating custom controls.

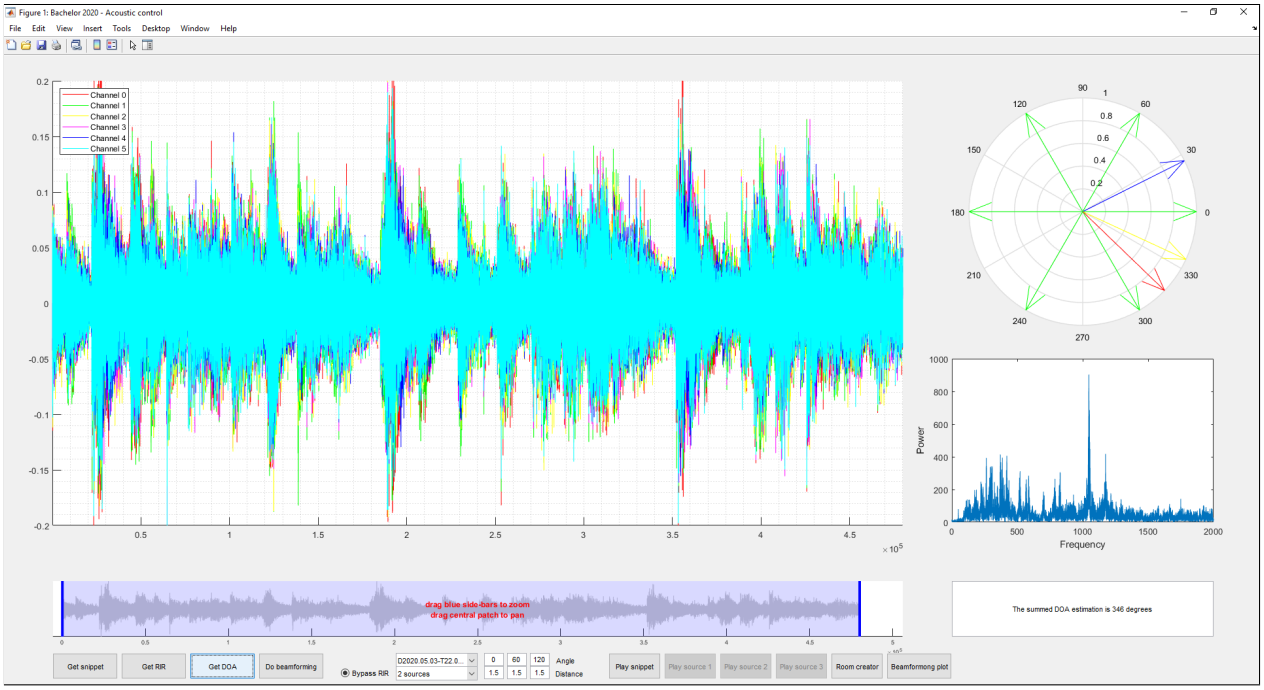


Figure 3.20: GUI application created with Java in MATLAB

The MATLAB version got updated to version, MATLAB 2020a Update 1, where the GUI application got a lot of errors and warnings because of the support for Java programming is getting removed from MATLAB. To get rid of the errors and warnings the GUI was rewritten in MATLAB App Designer. App Designer has drag and drop visual components. This made it fast and easy to move the GUI, as well made the code execution faster than using UI-Control with Java.

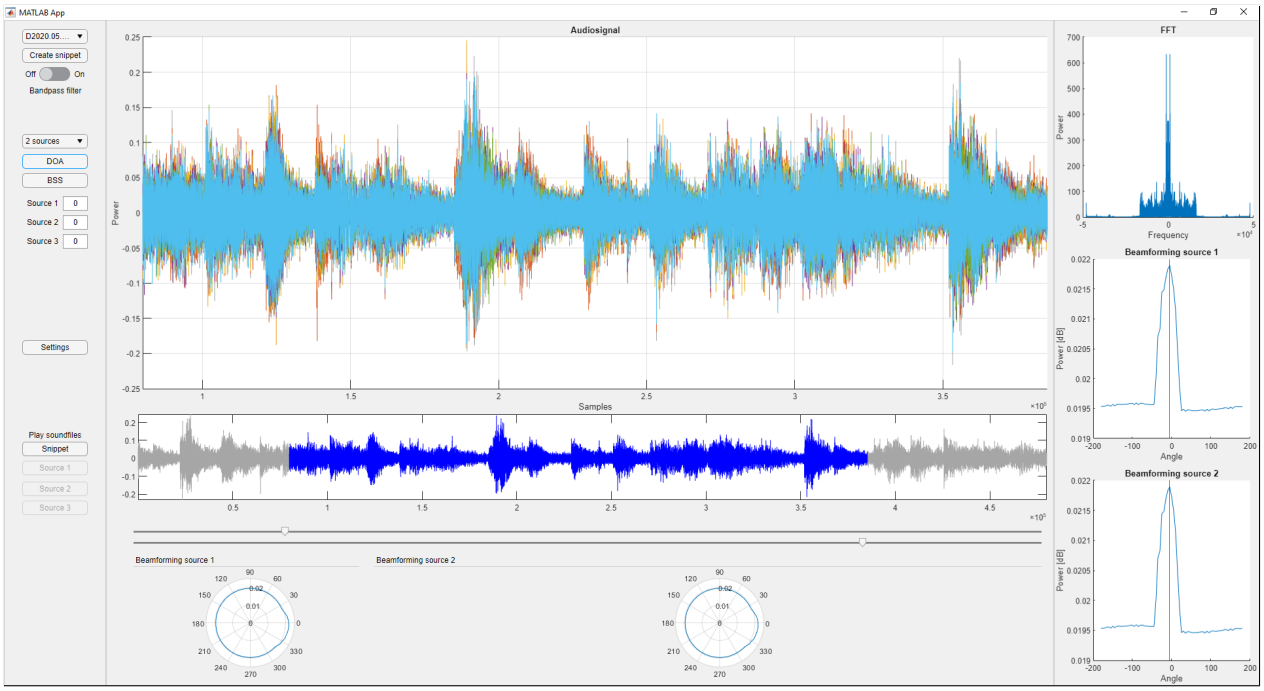


Figure 3.21: GUI application created with App Designer in MATLAB

3.9 IDE Applications

In the start of the project, the main application was created in PyCharm, which is a Python IDE. This language is a widely used language, and the group hoped to implement some of the available Python libraries. During testing of multiple libraries, the group concluded that just some of the libraries were useful. This resulted in a gradual implementation of a second IDE, with the language of MATLAB. The need for an IDE with graphical base functionality and illustrative possibilities made MATLAB preferable, as some of the methods needed in this project had to be made user made.

Program / addons	**Version**
MATLAB	R2020a
PyCharm	2020.1 (Professional Edition)
Python	3.5 32-bit
Python	3.6 64-bit

Table 3.2: Software

Chapter 4

Result

4.1 Components

In the thesis, the group ended up with a UCA formed array. Because a prolonged delivery and restricted Python support, the USB-AIO16-16F was not chosen.

4.1.1 ReSpeaker Mic Array v2.0

In the project two ReSpeaker mic array v2.0 was used for testing. It is a small and convenient shaped circular mic array used to record sound. The mic array usage is well documented and has already been used in a variety of project. The mic array was therefore selected for usage in the early test stages. The four microphones that are placed around its circular edges can record raw sound from the omnidirectional microphones. The four mics are combined in one channelled input to the computer by default. And by updating the firmware on the Mic Array, the single channelled input is separated into six channels. This separation makes the raw data from the microphones extractable.

An experimental firmware was tested for the ReSpeaker Mic Array. The updated firmware was to increase the sampling rate from 16kHz to 48kHz. The firmware update resulted in an unstable

sample rate because it did not sample steadily at 48kHz.

4.1.2 8-Ch 12-Bit ADC for Raspberry Pi

It got discovered that the Raspberry-Hat had speed issues. With 100Hz clock rate communication between the controller and the pi-hat, the A/D-converter got bottlenecked. When buying the Hat, only the specs for the multiplexer and the A/D converter was mentioned in the specsheet. So during the project, the ADC-hat was excluded as of the poor results.

Figure 4.1: Raspberry Pi4 with 8-Channel 12-Bit ADC for Raspberry Pi

4.1.3 ReSpeaker Core v2.0

The specifications on ReSpeaker Core v2 was superior compared to the other hardware in the same price range. The core seemed to have all the needed functions for an affordable price. Another benefit was that the microphones were uniformly aligned as a circle, which means the data can be processed to detect the direction of the sound signals.

With the ReSpeaker Core, the sample rate of the recordings could get up to 96kHz compared to the 16kHz sample rate of the Mic Array. This resulted in higher resolution of the recordings, which affected in more accurate calculations in the project. The ReSpeaker core also had a higher amount of microphones to record the surroundings. And since it is a micro-computer, it was the choice for the project, because it can also give extra functionalities like using WiFi to transfer the sound files.

With MEMS microphones mounted on the ReSpeaker, the characteristics were set to omnidirectional, this means a dome-shaped angle of perception of sound. The behaviour of the system was therefore set to the azimuth and a positive elevation plane. To simplify the project, the group decided to only work in a two-dimensional, which means the focus lies in the azimuth and radian plane.

4.1.4 Design

To hold our equipment from mechanical harm, the group produced two cases to our ReSpeaker Core v2.0 and the ReSpeaker Mic Array v2.0. The cases were 3D-printed in one of the 3D-printers at NTNU Ålesund.

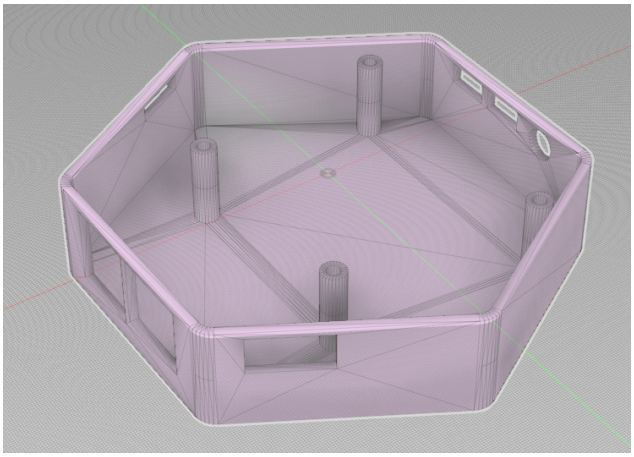


Figure 4.2: Respeaker Core v2.0 Cover[3]

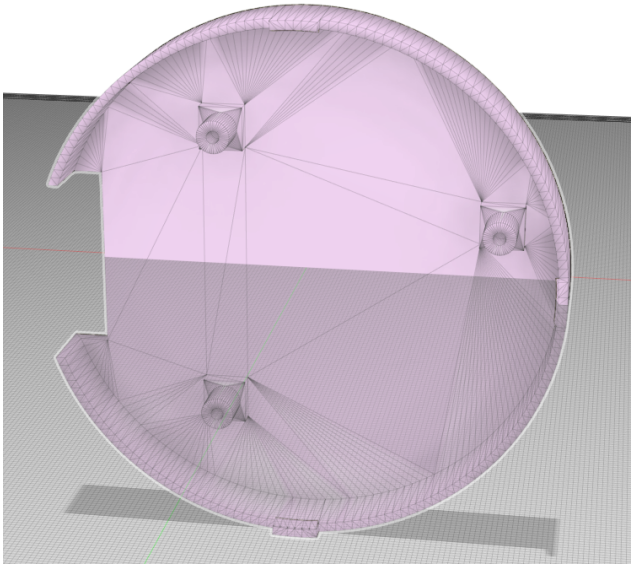


Figure 4.3: ReSpeaker Mic Array v2.0 Cover[4]

4.1.5 Recording

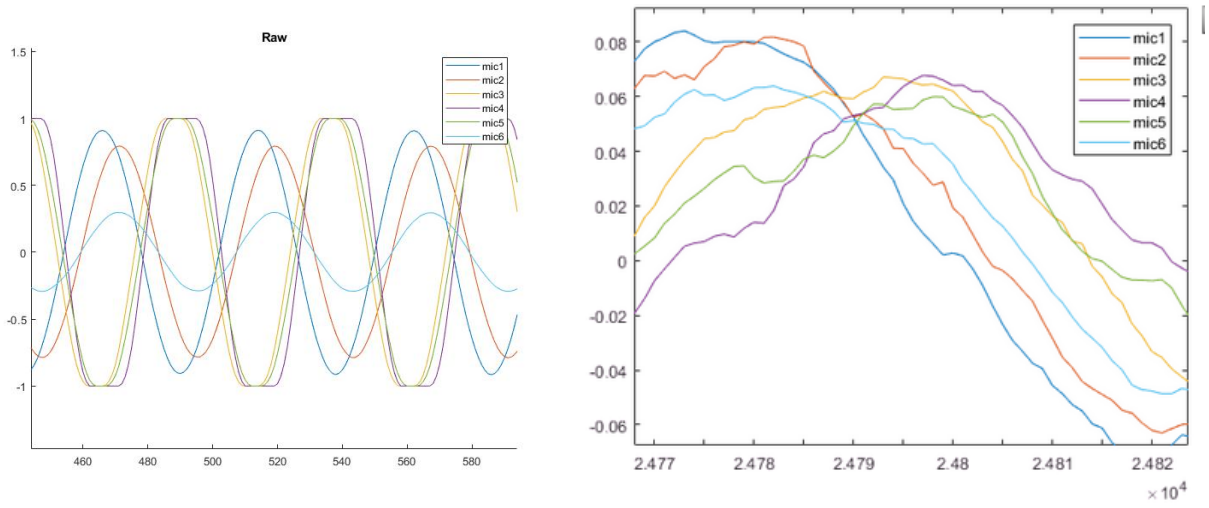
The recording are sets of multiple wave-files used for testing. In the recordings, different angles, sounds and number of sound sources were changed to create several unique recordings. Table 4.1 shows test-recordings created with multiple sources. Other recordings by using sinewaves with constant frequencies where also made, see Table 4.3 and GitHub for fill list [5].

Sound files	**Source 1**	**Source 2**	**Source 3**
multisource1.wav	Angle 0 1000Hz sinewave	Angle: 90 Eminem: Godzilla ft. Juice WRLD	N/A
multisource2.wav	Angle 0 1000Hz sinewave	Angle 60 500Hz sinewave	N/A
multisource3.wav	Angle 0 1000Hz	Angle 120 500Hz sinewave	N/A
multisource4.wav	Angle 0 Childish Gambino: 35.31	Angle 120 TIX: Karantene	N/A
multisource5.wav	Angle 0 SAINt JHN, Imanbek: Roses - Imanbek Remix	Angle 120 Drake: Tossie Slide	N/A
multisource6.wav	Angle 0 The Weeknd: In Your Eyes	Angle 60 Tones And I: Bad Child	Angle 120 Lil Mosey: Blueberry Faygo
multisource7.wav	Angle 0 Electric motor	Angle 120 Factory sounds	N/A
multisource8.wav	Angle 0 Factory sounds	Angle 60 Washing machine	Angle 120 Electric motor
multisource9.wav	Angle 0 LM4 - Electric Motor 05 - EDU172-AB	Angle 60 175 Regulator_empty long_closed valve 3_Sennheiser	Angle 240 MOT Electric scooter motor brushless accelerate decelerate and rev
multisource10.wav	Angle 0 Sound design Shepard tone - abrasive high pitched whine from an electric motor - ever increasing in pitch - loopable_	Angle 60 175 Regulator_empty long_closed valve 3_Sennheiser	Angle 240 MOTOR-FAN-015_int -mechanical_ELECTRIC -FAN_typeA_low_B_contact -mic
multisource11.wav	Angle 0 EFX EXT Pnumatic Nail Gun Series 02	Angle 60 LM4 - Electric Motor 02 - 60AKG-MS	Angle 240 motor electric medium start up run off slow down_03

Table 4.1: Recordings

4.1.6 Hardware malfunction

There were equipment issues during the test phase of the project. The recording equipment malfunctioned, and each channel had abnormalities in the amplitudes, plus some latency issues. The fault was detected when test data was gathered, and the abnormality was sporadic and had not been there before. In Figure 4.4 one of the problems are illustrated. An earlier recording is added besides to compare.



(a) hardware malfunction

(b) Before hardware malfunction

Figure 4.4: Before and after hardware malfunction

By comparing the two figures in 4.4 it's possible to draw some inequalities. Figure 4.4a has the most variation of its channels, where some channels have a low magnitude, and the bounds limits others. Another difference is in the distortion of the channel, where the results are not as expected. The error seems to be an hardware problem, as of most of new recordings had distortion. The problem got encountered at the end of the thesis and therefore is not solved.

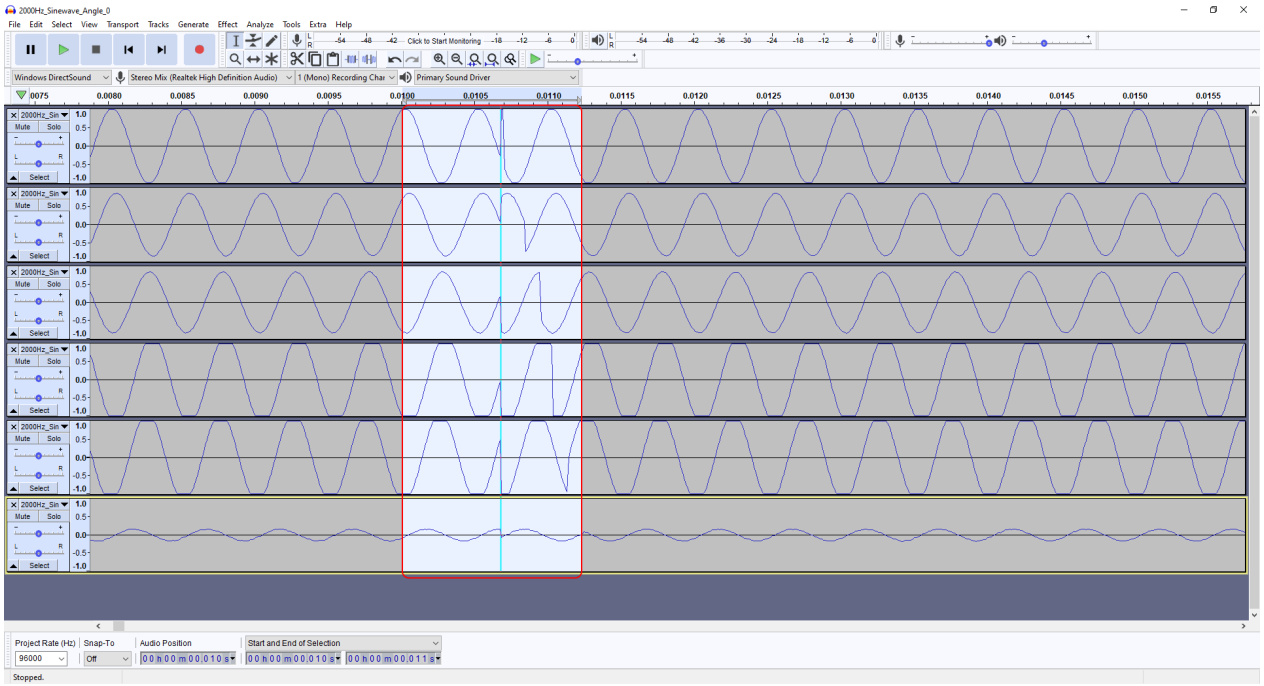


Figure 4.5: Bluetooth problems

In Figure 4.5 another error was discovered earlier in the project by analysing the recorded file in Audacity. This was found to be linked to Bluetooth delay. This resulted in a change to direct-connection through AUX cable.

4.2 GUI

The finished GUI was created with App Designer in MATLAB, shown in Figure 3.21. It has buttons and plots to display the information that is being calculated in other scripts. To select an audio clip to process, the dropdown textlist is used, where it is linked to a folder where the audio clips are saved. When a file is selected, the main plot updates and displays the audio waves. The folder path can be changed by clicking the settings button, and then change the folder path destination. Here it is also possible to change the UCA configuration.

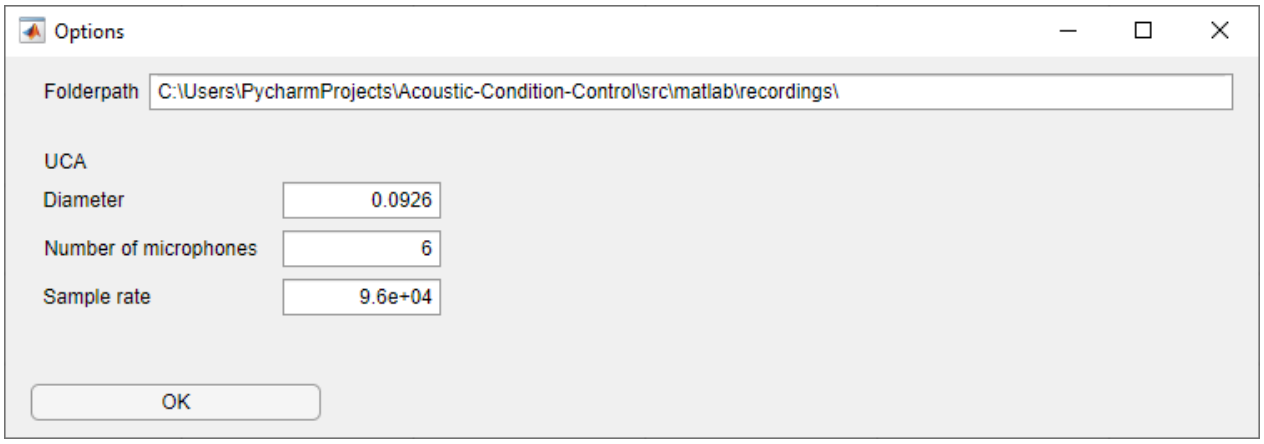


Figure 4.6: User application - Settings

4.2.1 Snippet

To shorten an audio clip use the two sliders on the bottom of the main plot to create a snippet of the audio clip by pressing the Snippet button. The main plot and the FFT plot will dynamically update while using the sliders to display the most optimal image of the sound wave. After pressing the Snippet button, it checks if the bandpass filter button is enabled. If enabled, the snippet is run through a bandpass. The bandpass filter is configured to the spesification of the microphones, 20Hz-10kHz. Then it is being saved as a wave-file called snippet.wav.

4.2.2 DOA

By pressing the DOA button, the DOA estimation is being calculation from the snippet file. The results from the estimations are being displayed in the plots on the right-hand side. There are also polar plots on the bottom part on the GUI.

4.2.3 BSS

Different types of BSS methods can be chosen by using the dropdown menu. Then to select which type of BSS method to use, select between two or three sources, also choose whether to

use chunk-splitting or not. The audio snippets gets split into chunks to increase performance. Then by pressing the BSS button, it gathers the angles from the textboxes below and runs it through the selected BSS method. Then the results are saved as its own wave-file.

4.2.4 Play sound clips

There are four buttons to play the audio clips. The first buttons are to play the audio snippet and can be used to check if the original audio is corrupted. The Tree other button called Source 1, Source 2 and Source 3 are linked to the BSS files.

4.3 Echo and reverb cancellation

4.3.1 Adaptive filter

Adaptive filter without reference sound is not obtainable within the time limitations. Since the project setup only uses microphones, a comparison of the output signal $\hat{d}(n)$ with the desired output $d(n)$ is not achievable. The error signal $e(n)$ which updates the coefficient vector for the filter is not calculable. According to the contacted professors at NTNU Trondheim, it is possible to create an adaptive filter without reference sound. Still, it is challenging but not achievable in this bachelor thesis.

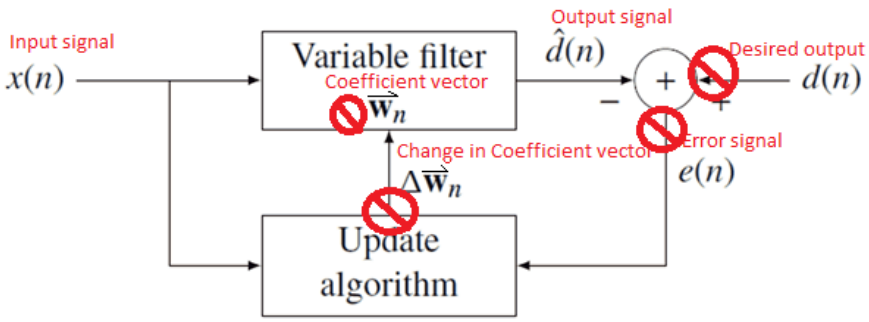


Figure 4.7: Diagramn of showing which reference sources are missing for completion of adaptive filtering

4.3.2 RIR

MATLAB with python implementation

Low coupling and high cohesion not implemented in Python RIR.py due to a MATLAB issue. There is a direct MATLAB-Python implementation in MATLAB. This means that MATLAB can run Python-code. The problem encountered was that all of the methods had to be in the same python file for MATLAB to run the it, which goes against low coupling and high cohesion.

Acoustic signal processing library

As mentioned before, the library called PyroomAcoustics was implemented. This library had an example on the Jupyter notebook that was rewritten to function in PyCharm. The reason for this is that running the PyroomAcoustics library online was faulty, and sometimes the Jupyter kernel would not start. This could perhaps be solved by renting a server on Jupyter with fewer limitations [19].

Test RIR accuracy

The Room Impulse Response with ISM-method is showing wanted results. The simulator is meant to create reverb from a dry sound without reverb. As shown in the figure, red is the recorded signal and green is the simulated signal. It is visible that the simulation creates almost a copy of the original code, which shows that the simulations estimation is fair. More accurate simulation is possible by increasing the order of the simulation.

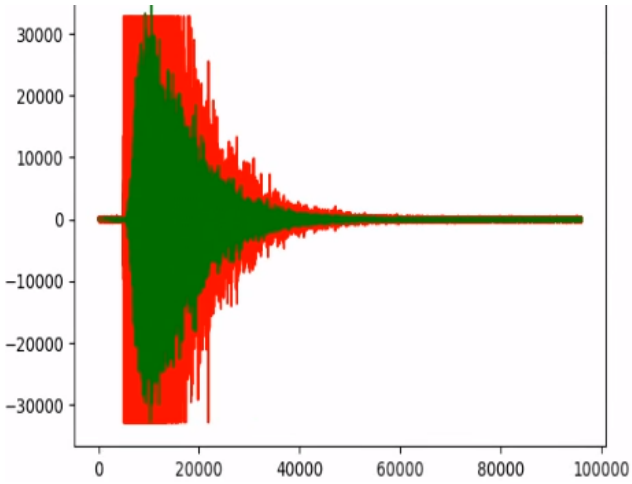


Figure 4.8: Original signal in red compared to the simulated sound in green

Test impulse response difference between mics

Test for checking that the RIR changes when iterating microphone locations. To check if the uniform circular array changes the position of the mics every iteration, which impacts the room impulse response between the source and mic, I use “import matplotlib.pyplot as plt.” Then I plot the mic positions and that they move every iteration.

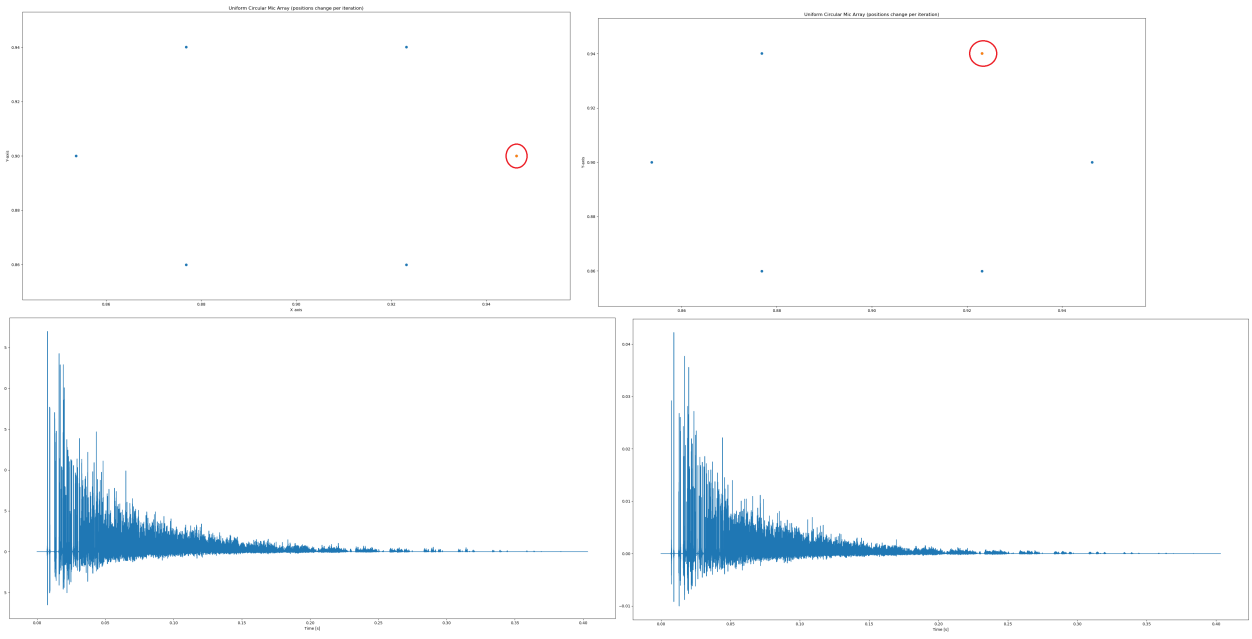


Figure 4.9: Two pictures that plots the microphone locations and the impulse responses changing

Testing of the denoise function

There is a function that applies a subspace denoising approach. It takes in the noisy sound array and tries to reduce the noise in the time domain. After adjusting and listening to the results, with many different thresholds like mu and loop-back, the achieved results were undesirable. An e-mail was sent to pyroomacoustics on how to use the filter, but there was no reply. Therefore, the approach was scrapped.

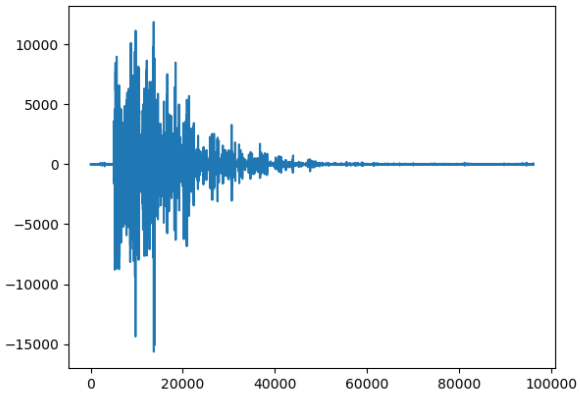


Figure 4.10: Indistinguishable denoised signal compared to the original

Test high-pass filter

High-pass filter denoising. Tried to remove white noise by merely creating a high-pass filter. The high-pass filter removed amplitudes below a certain high-pass and resulted in a less noisy sound. The approach was scrapped because the method may remove important information from the sound for later processing.

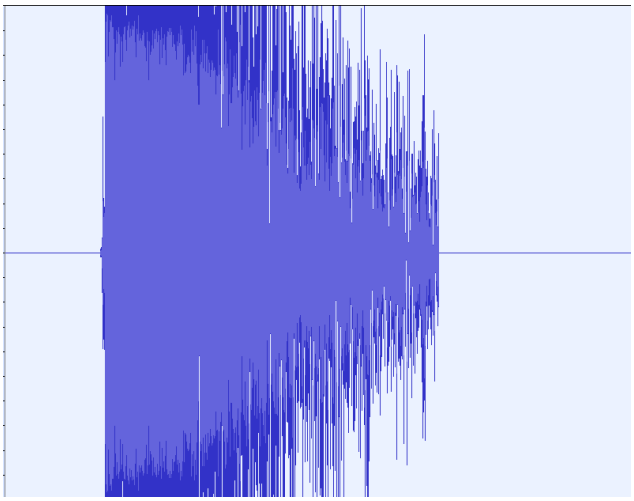


Figure 4.11: Flat line is the result of the high-pass filter

The experimental code returned to unreliable values. The calculations returned from this innovative method was an absorption coefficient that was neither between 0-1 or in “Sabins.” This experimental code is therefore not applicable in room setups. Will need further testing to achieve desired amplitude domain where it returns a coefficient between 0 and 1 or in Sabins.

Test automatic absorption method – “inv_Sabine( )”

```
def getExperimentalAbsorptionAndRoomOrder_Sabine(t60, room_dim, c):

    return absorption, max_order
```

Listing 4.1: Code to estimate max order and absorption of the room by calculating the RT_{60}

4.4 Beamforming

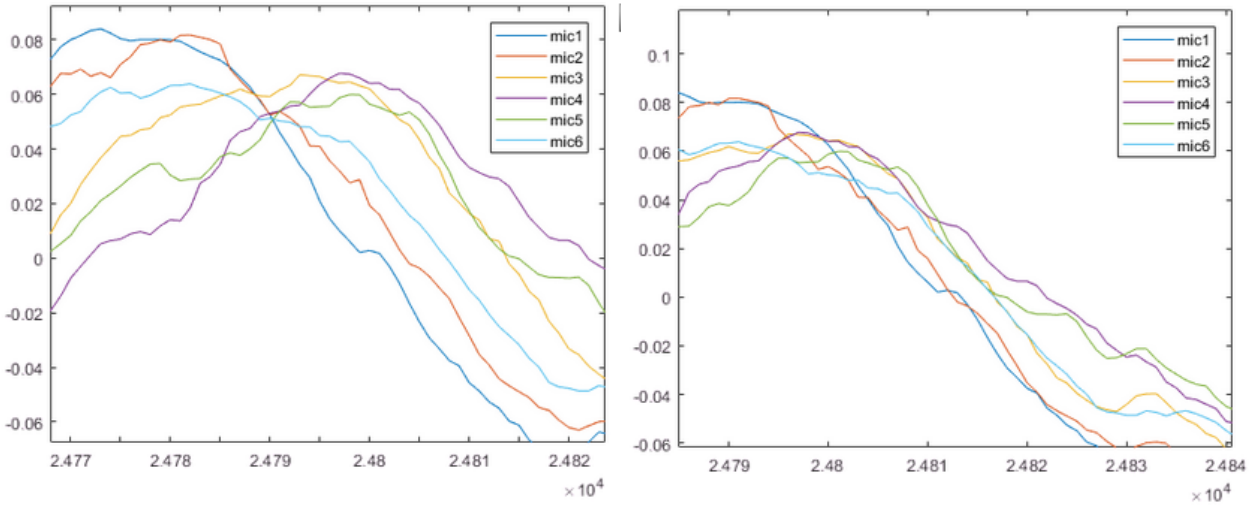
The beamforming results are a set of multiple software solutions. Some of the software is separated and is intended for multi-source recording. And other tools are for detection of multi directional sound transmitters.The software tools are suited to different purposes but built on the same BF technique.

Name	BSS	Beamforming	Source support	DOA	Comment	Input arguments
BSS2S.m	x	x	2		Simple separation	R,A
BSS3S.m	x	x	3		Simple separation	R,A
ChunkBSS2S.m	x	x	2		Separation for longer recordings	R,A
ChunkBSS3S.m	x	x	3		Separation for longer recordings	R,A
OneSourceFocus.m	x	x	1		Focus Beamforming in a specific angle	R,A
DOA.m		x	1	x	DOA estimation	R
DOAmulti.m		x	2	x	DOA estimation of two sources	R

*R-recording, A-angle

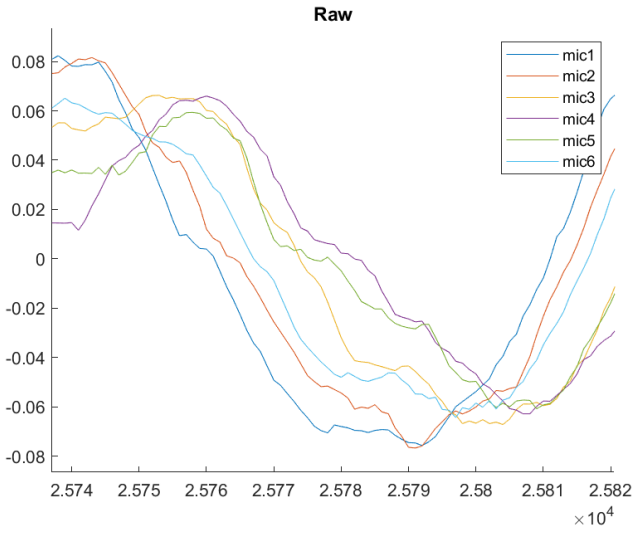
Table 4.2: Beamforming codes

The result of the beamforming technique in this project is the use of it in the BSS and DOA calculation. Figures below show some graphical examples of the approach.

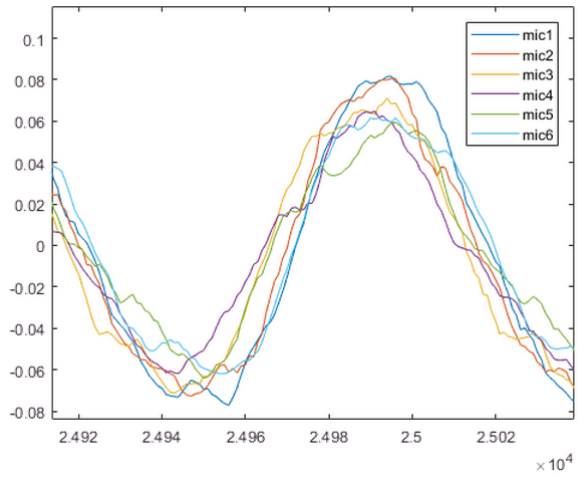


(a) Raw-data from the ReSpeaker Core (b) Theoretical aligned data

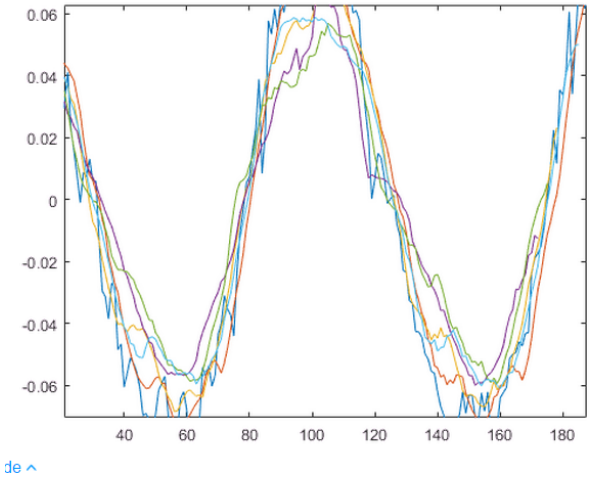
Figure 4.12: Before and after time delay of the channels



(a) raw input data



(b) Theoretical calculated time delay



(c) Cross-Correlated delay

Figure 4.13: Channel alignment

In Figure 4.13c, cross-correlation is used on the raw signal 4.13a to calculate the delays in the channels. By using the delays, the distortion can be compensated bu aligning the signals. The method works, but it has its limitations when it comes to multiple sources. In Figure 4.13b it is possible to see the different results of theoretically calculated signals, from the code 3.8, and both cross-correlation and theoretical calculation are quite alike. Each plot represents one channel from a recording.

delayed_channels

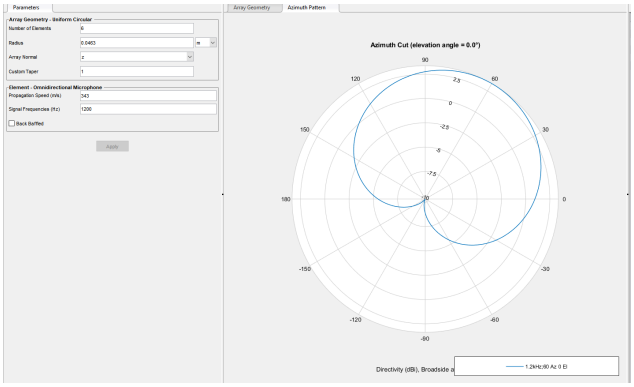
96000x6 double

	1	2	3	4	5	6
1	0	0	0	-3.0518e-05	0	0
2	0	0	0	-3.0518e-05	0	0
3	0	0	0	0.0027	0	0
4	0	0	0	0	0	0
5	0	0	0	0	0	0
6	0	0	0	0.0103	0	0
7	0	0	0	0.0111	0	0
8	0	0	-3.0518e-05	0.0130	-3.0518e-05	0
9	0	0	-3.0518e-05	0.0104	-3.0518e-05	0
10	0	0	-3.0518e-05	0.0098	-3.0518e-05	0
11	0	0	0	0.0089	0	0
12	0	0	0	0.0216	0	0
13	0	0	0.0545	0.0341	0.0321	0
14	0	0	0.0586	0.0480	0.0352	0
15	0	0	0.0583	0.0530	0.0383	0
16	0	0	0.0520	0.0444	0.0345	0
17	0	0	0.0431	0.0442	0.0393	0
18	0	0	0.0551	0.0242	0.0332	0
19	0	0	0.0539	0.0284	0.0301	0
20	0	0	0.0565	0.0457	0.0325	0
21	0	-3.0518e-05	0.0615	0.0603	0.0271	-3.0518e-05
22	0	-3.0518e-05	0.0508	0.0537	0.0486	-3.0518e-05
23	0	-3.0518e-05	0.0522	0.0552	0.0404	-3.0518e-05
24	0	0	0.0530	0.0573	0.0435	0
25	0	0	0.0672	0.0602	0.0356	0
26	0	0.0815	0.0605	0.0633	0.0400	0.0551
27	-3.0518e-05	0.0713	0.0661	0.0728	0.0411	0.0562

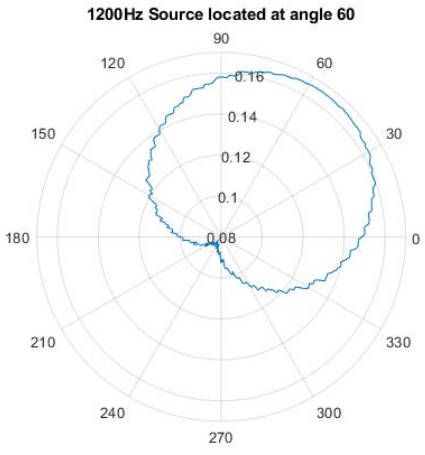
Figure 4.14: Delayed matrix of mic data

In Figure 4.14, a matrix of signal data has been delayed with zeroes at the beginning of the signal. Zeroes added to each channel represent time-shifts needed to align the data from each microphone.

In MATLAB, Sensor Array Analyzer can be used to predict the lobes of the beamformer. Lobe prediction can be made for frequencies in the microphone range. In the predictions variables as microphone sensitivity, positions and transmitter variables are filled in. With the ability to compare predictions with actual measurements, the group, can check the results against an simulated result.



(a) Simulated lobe with 1200Hz source and arrival angle 60



(b) actual lobe shape

Figure 4.15: System lobe shape test

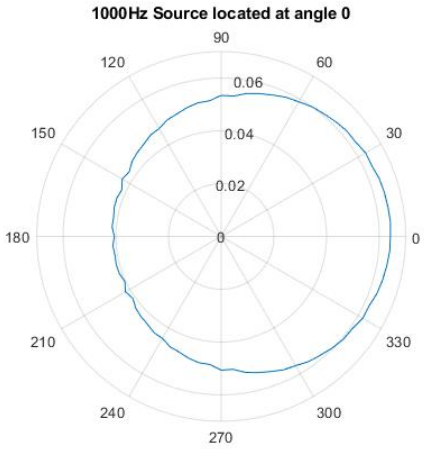
4.4.1 DOA

The logarithm's function is to use beamforming in all directions with a preset rendering. By gathering the SRP information of the signals in each direction, it is possible to point out sound sources in the surroundings. Output SRP is plotted for every direction into a pole plot graph to illustrate the direction to the audio transmitter. In the following table, the DOA.m and DOAmultisource.m have been used to estimate the DOA.

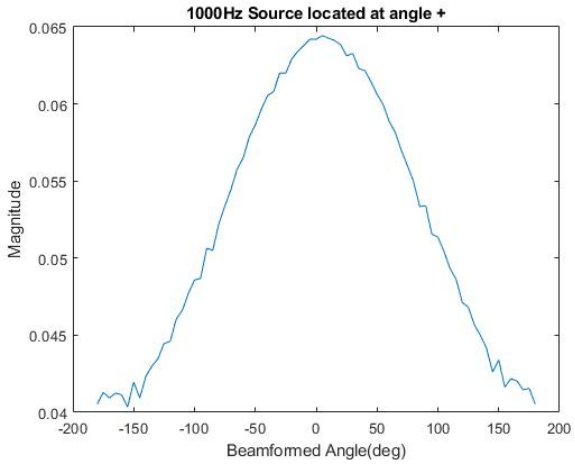
Recording	Angles(deg)	Magnitude Beamforming	DOA	Info	Comment	Figure
Multisource1.wav	0/90	0.1344/ 0.1364	5/100	Sinewave 1000Hz and music		DOA1
1200Hz_Multisource_0000.wav	0/-45	0.1344/0.1329	0/-45	Two sinewaves 1200 and 750Hz		DOA2
1200Hz_Multisource_5,625.wav	5.625/-45	0.1640/0.1533	0/-31	Two sinewaves 1200 and 750Hz		DOA3
500Hz_SinewaveAngle0.wav	0	0.2502	-10	500 Hz Sinewave		DOA4
500Hz_SinewaveAnlge60.wav	60	0.2293	55	500 Hz Sinewave		DOA5
500Hz_SinewaveAngle120.wav	120	0.2325	115	500 Hz Sinewave		DOA6
500Hz_SinewaveAngle180.wav	180	0.2418	175	500 Hz Sinewave		
500Hz_SinewaveAngle240.wav	240	0.2376	-125 = 235	500 Hz Sinewave		
500Hz_SinewaveAngle300	300	0.2536	-60 = 300	500 Hz Sinewave		

Table 4.3: DOA results

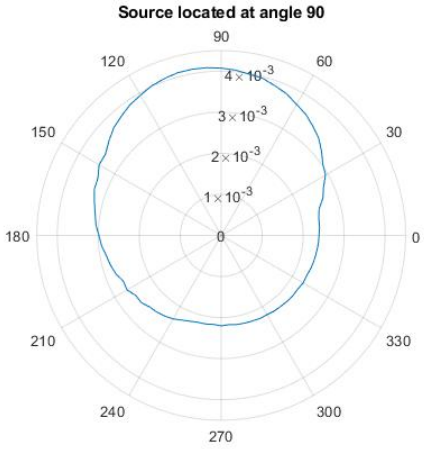
As seen above in Table 4.3, the prediction of the algorithm has some difficulties with hitting the exact angle. The test was done with the MATLAB codes DOA.m for single source and DOAmultisource.m for two sources. Results indicate the problems don't lie in the repeatably, but rather the accuracy. The single source DOA estimation has a flaw on -5 degrees, which can be eliminated with a bias. The multi-source DOA estimation has lower accuracy and repeatability.



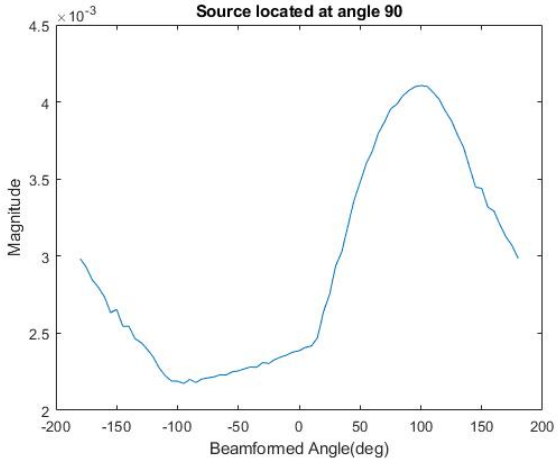
(a) Polar plot DOA estimation, source one



(b) Magnitude plot DOA estimation, source one



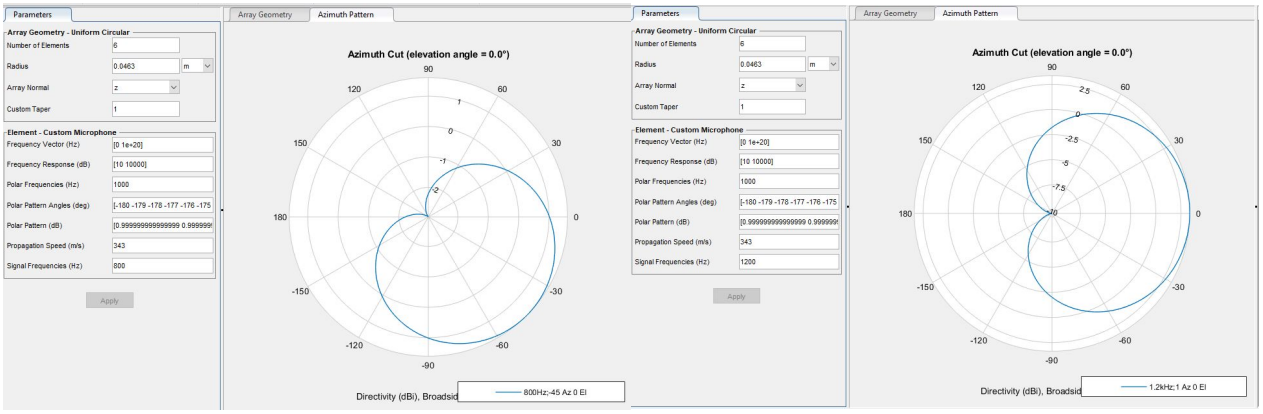
(c) Polar plot DOA estimation, source two



(d) Magnitude plot DOA estimation, source two

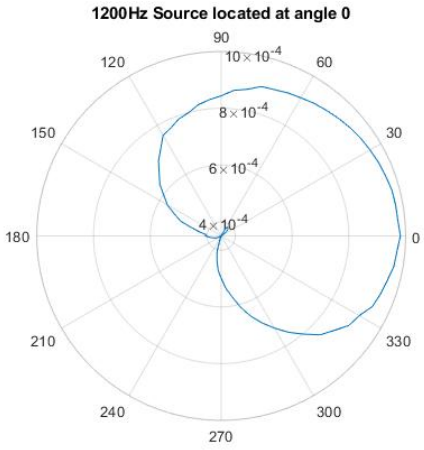
Figure 4.16: DOA1

In the subfigures 4.16, the recording of two sources can be identified. As mentioned in table 4.3, the illustrations shows the direction of arrival for each respected source. The transmitted sounds is a 1000Hz sine wave and a song. Original displacement of the sources is at angle 0 and 90 degrees, the estimated direction is 5 and 100 degrees.

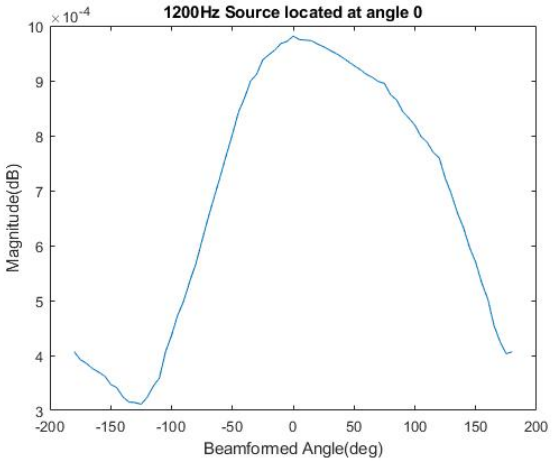


(a) Simulated lobe, 800Hz signal at -45 degrees

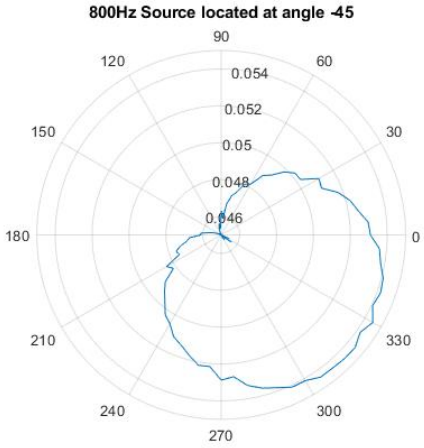
(b) Simulated lobe, 1200Hz signal at 0 degrees



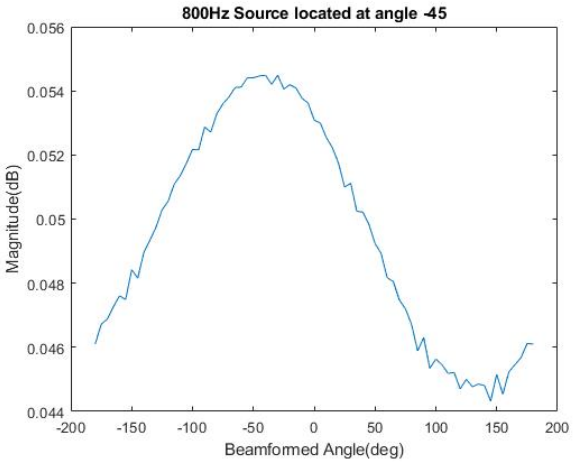
(c) Polar plot DOA estimation, source one



(d) Magnitude plot DOA estimation, source one



(e) Polar plot DOA estimation, source two

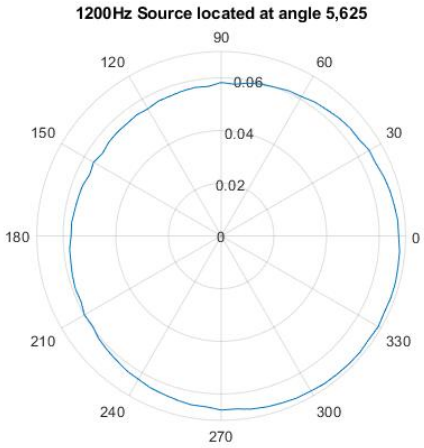


(f) Magnitude plot DOA estimation, source two

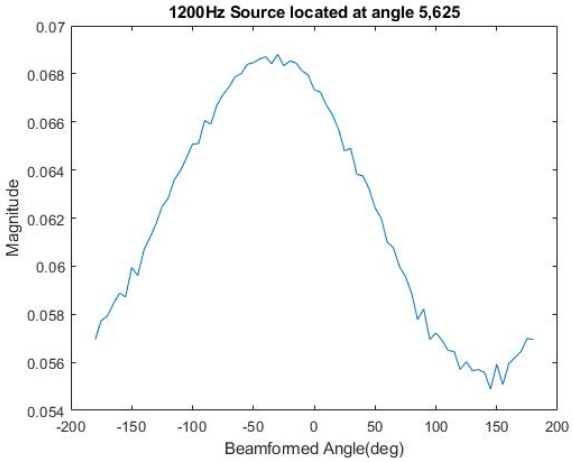
Figure 4.17: DOA2

In the subfigures 4.17, the recording of two sources can be identified. In plot 4.17a and 4.17b,

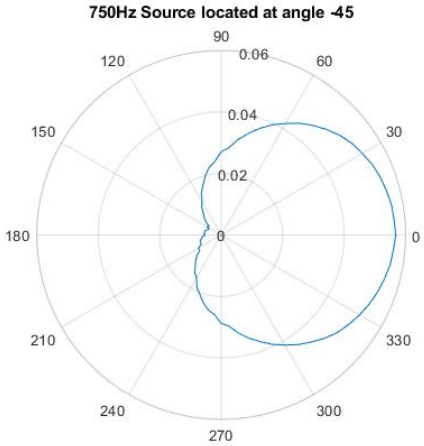
the simulated lobes illustrate the expected behaviours of the system with single sound sources. In Figure 4.17c -4.17f the system behaves as expected with an accurate DOA estimation. The Acoustic emitters are placed in angle 0 and -45 degrees, which is the same as the calculated displacement.



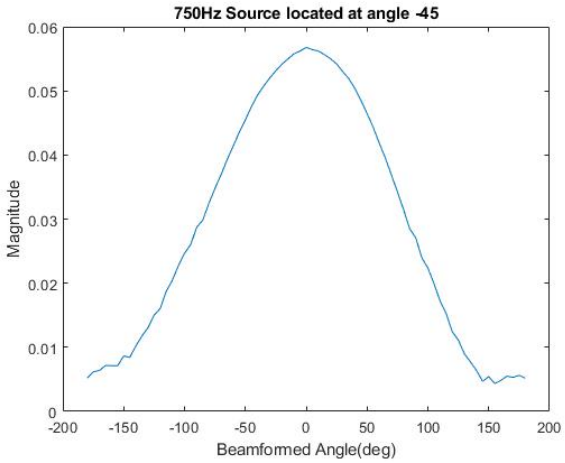
(a) Polar plot DOA estimation, source one



(b) Magnitude plot DOA estimation, source one



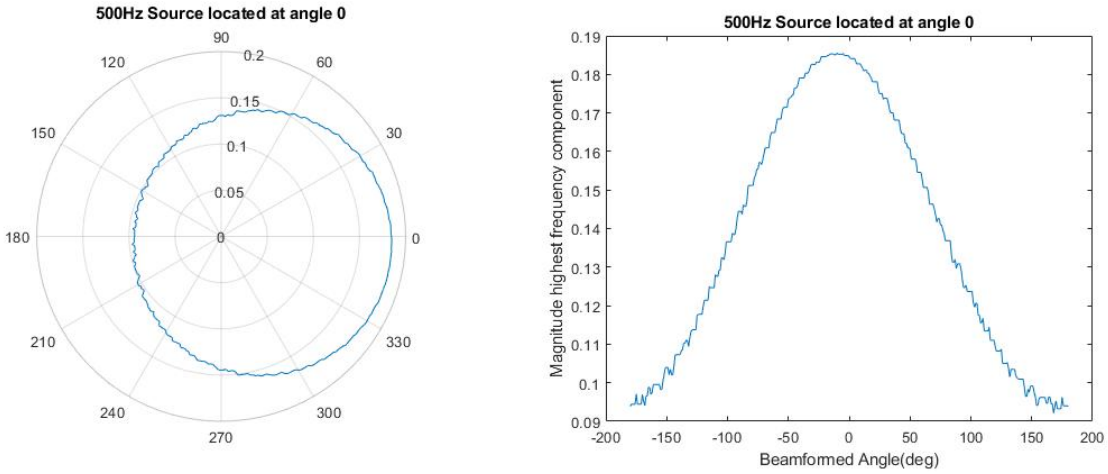
(c) Polar plot DOA estimation, source two



(d) Magnitude plot DOA estimation, source two

Figure 4.18: DOA3

In the subfigures 4.18, the recording of two sources with sine waves at 750 and 1000Hz is present. As mentioned in table 4.3 the estimated direction of arrival is at angle 0 and -31 degrees, with the respect of the true position, the DOA estimation error is 5.6 and 14 degrees. This can be linked to wave collision and/or the systems fragile handling of the imperfect testing condition. See 4.1.6.



(a) Polar plot DOA estimation

(b) Magnitude plot DOA estimation

Figure 4.19: DOA4

In the Figure 4.19 a 500Hz sinewave is located at angle 0. at 1:1: 360-degree resolution.

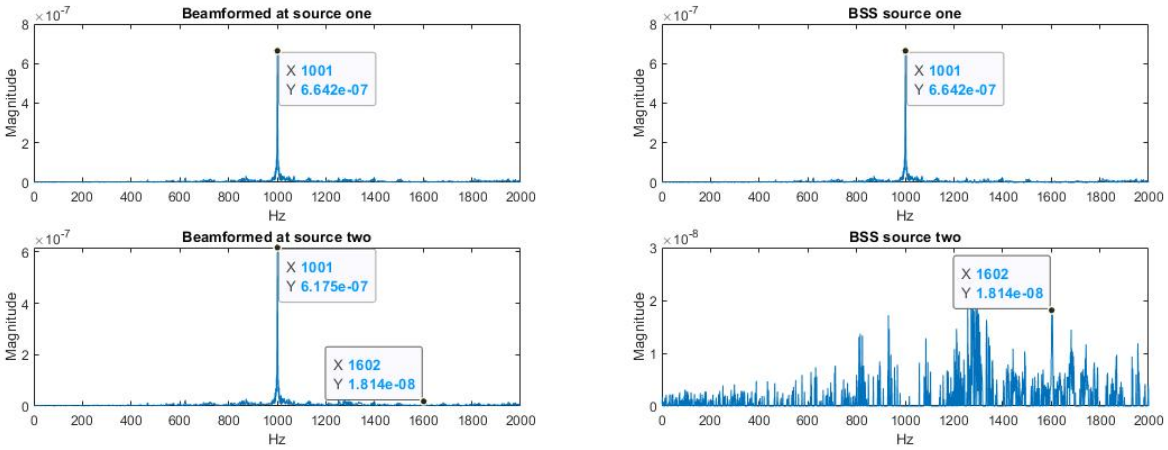
4.4.2 BSS

The result in this part of the project was a program that can source separate audio signals. The program was written in MATLAB and is at the moment only compatible with the right hardware and pre-specified variables. In Table 4.4 the SNR ratio of the separated sound sources are written.

Recording	Angles (deg)	Magnitude inputsignal	SNR BSS	Info	Comment	Figure
Multisource1.wav	0/ 90	0.1344/ 0.1364	63.05	Sinewave 1000Hz and music		BSS1
Multisource2.wav	0/ 60	0.0322/ 0.0265	41.28	Two sinewaves 1000 and 500Hz		
Multisource3.wav	0/ 120	0.0372/ 0.0347	1.34	Two sinewaves 1000 and 500Hz	Error	
1200Hz_ Multisource_0000.wav	0/ -45	0.1344/ 0.1329	38.98	Two sinewaves 1200 and 750Hz		BSS2
1200Hz_ Multisource_5625.wav	5.625/ -45	0.1640/ 0.1533	48.78	Two sinewaves 1200 and 750Hz		BSS3
1200Hz_ Multisource_11250.wav	11.250/ -45	0.1273/ 0.1301	49.69	Two sinewaves 1200 and 750Hz		
1200Hz_ Multisource_0000.wav						

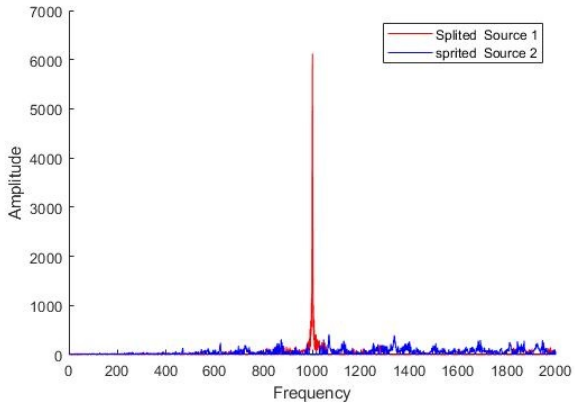
Table 4.4: SNR

$$SNR = \frac{P_{Singal}}{P_{Noise}} \tag{4.1}$$



(a) Beam-formed signal in the account of direction

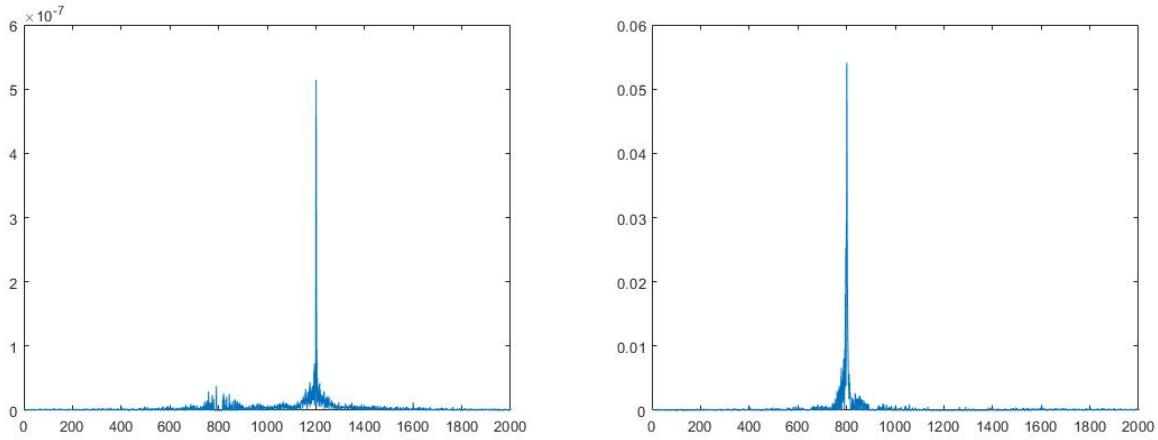
(b) BBS seperated sources



(c) BBS plotted together

Figure 4.20: BSS1

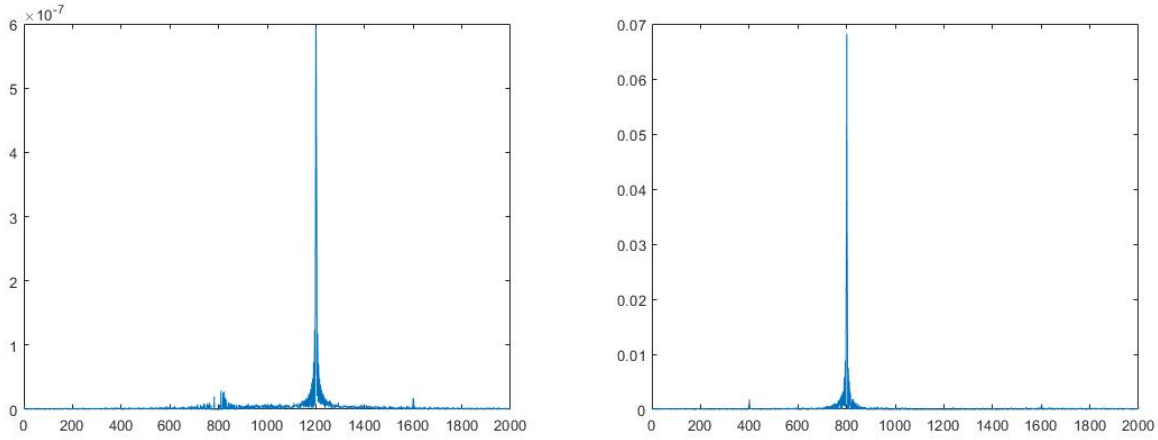
In the Figure 4.20 the recording contains a 1000Hz sine wave and music, with a displacement of 60 degrees apart. Figure 4.20a shows a beamformed signal against the two sources. By comparing the magnitudes in each frequency, it is possible to separate the sources in consideration of the direction of greatest magnitude. By looking at Figure 4.20b the sources are separated, where source one, 1000Hz sine wave, is only present in one of the audio arrays. The signal to noise ratio of the separation is 63.05.



(a) BSS source one (b) BBS source two

Figure 4.21: BSS2

In the above Figure 4.21, the separation of two sine waves are possible. The waves have a frequency of 1200Hz and 800Hz and are placed in angle 0 and -45 degrees. By studying the plots, it is possible to see some side frequencies that are not separated. Some of the reasons behind this are wave collision between the two sources.



(a) BSS source one (b) BBS source two

Figure 4.22: BSS3

In Figure 4.22 the displacement of sources angles is 5.6 and -45 degrees. With a better recording, the SNR ratio is increased with from 39 to 49, with almost the same scenario in 4.21.

4.5 Application

Script	Description
record.py	This is a Python script that is ran in the ReSepaker Core v2.0. This script creates a recording for x seconds and saves it on a shared folder on the network with the date and time as the filename
Application.mlapp	This is the main GUI application created in MATLAB App Designer
ApplicationSettings.mlapp	This is the settings GUI application created in MATLAB App Designer
BSS2S.m	This is a MATLAB function that preforms blind-source-separation on two sources.
BSS3S.m	This is a MATLAB function that preforms blind-source-separation on tree sources.
chunckBSS2S.m	This is a MATLAB function that preforms blind-source-separation on two sources split up into chunks.
chunckBSS3S.m	This is a MATLAB function that preforms blind-source-separation on tree sources split up into chunks.
DOAmultisource.m	This is a MATLAB function that preforms DOA estimation.
UCA.m	This is a MATLAB class. That defines the UCA configuration.

Table 4.5: Script description

4.5.1 Libraries

Name	Version	Language	Used	Comment
pyAudio	0.2.11	Python	Record.py	pyAudio is a library used to capture and record sound.
wave	21.5	Python	Record.py	A library used to store and capture sound as .wav files
soundDevice	0.3.14	Python	Record.py	A library used to find connected sound devices
PyroomAcoustics	0.3.1	Python	Echo-Canselation	Library with different solutions for audio array prosessing algorithms
Acoular		Python	N/A	An audio processing library for Python
Librosa		Python	N/A	LibROSA is a Python package for music and audio analysis
SimpleAudio		Python	N/A	Library for recording and playing wav files
numPy		Python	N/A	A powerfull calculation library for scientific use
Matplotlib		Python	N/A	A library used for plotting figures and interactive environments in Python
ReSpeaker GROVE		Python	N/A	ReSpeakers own library for hardware souch as the Core V2 and the Mic Array
Scipy.io		Python	N/A	A library for mathematics, Scientific and engineering tools
adaptfilt		Python	N/A	adaptfilt is an adaptive filtering module for Python
gpuRIR		Python	N/A	Library used for RIR calculation and GPU acceleration

Table 4.6: Libraries

Chapter 5

Discussion

5.1 Recording results

During testing, multiple hardware errors and malfunctions occurred. The first abnormality that was discovered when creating a test-file recording from numerous sources when using Bluetooth connected speakers. When this program was started, the results of the BSS and DOA algorithms varied. The algorithms had problems locating the sources when using periodical sinewaves as the emitted sound, which lead to error troubleshooting.

When the abnormalities were detected, even more troubleshooting was necessary to find the error. By performing a test recording, the file could be analysed in Audacity. Within the audio-files, it was possible to see audio cut off in multiple places, as seen in Figure 4.5. There where also variation in the synchronisation in the recordings linked to a problem using Bluetooth.

When Bluetooth is connected from two phones to two speakers, there will be a delay when pressing the play button on each phone. When playing a signal, e.g., a periodical sinewave, sound collisions can occur and might have been the reason for the error. With the right amount of delayed start, a magnification or cancellation started to happen between each sinewave. This sound wave combination occurred because of the superposition principle. The frequencies range used during the issue wherein 750Hz to 1200Hz.

When the Bluetooth connection was swapped with a cable, the audio cut off issue vanished. This due to a reduction in the delayed start when pressing play. However, a new problem occurred. The extended audio wires were old, and the old RCA switch created a degradation in sound quality when connected to only a PC. The problem was fixed by reverting to the phones, only with shorter AUX cable.

Another problem that occurred was the sound files used to create test recordings. Sinewaves, music and prerecorded machine sounds were used to develop these test files. When using sinewaves with constant frequencies, the two old speakers wore out. This resulted in purchasing three new speakers, but two of the new speakers also malfunctioned after playing sinewaves. But sinewaves were still used since they were easy to work with, and in a real scenario, there will not be speakers playing sound.

After switching rooms from the bedroom to the living room, another issue occurred. An issue with EMI might have affected the recording. The problem probably happened due to interference from the floor heating element. Time was consumed troubleshooting the abnormality when the hardware recorded sounds, and the group, therefore, came up with a hypothesis. With an electromagnetic field emitted by the heating cables underneath the floor, the recording equipment could take damage. The heating was therefore deactivated, but because of a smart functionality on the controller, it was impossible to disable the heating without disconnecting the fuse entirely. The controller has a timer which will automatically reactivate the heat after 6 hours. This might have affected the hardware over time, and at the end of the thesis, the equipment was damaged 4.1.6.

Earlier in the project, a product called USB-AIO16-16F was taken into consideration. It may be the a better solution for this project and we think this device should have been purchased. If we were not so determined to use Python as our main IDE, since Python has a lot of AI libraries, it would probably make us select this instead of the ReSpeaker ADC HAT. If this product were to have a complete Python supported system, it might have been an good implementation to our project. But instead the device supports C language, which would not have been a problem at all. The pandemic also had an impact, on the delivery time, since this is a product from USA.

5.2 Echo Constellation

The adaptive filter method is non-trivial. After five weeks of reading and research, the conclusion was to scrap the Normalized Least Mean Squares adaptive filter. Firstly, to implement this algorithm, the typical setup is a microphone $x(n)$, and a speaker $d(n)$. Since it is not intended to use speakers in the end module, but rather live sound sources, the error $e(n)$ cannot can not be subtract from the signal. The reason is that only microphones are being used, which creates a different system than the typical adaptive filter algorithm is intended for. And since time was lost before arriving at this conclusion, a final attempt was done, and the acoustic engineering professors at NTNU Trondheim were informed about the issue. After being in touch with professor Guillaume Dutilleux and another professor at NTNU, the result was to abandon the approach. Due to that an adaptive filter method without the reference from a speaker is possible, but complicated. And that it was unwise to pursue this solution in a thesis with such limited time. The direction of the adaptive filter was then rerouted towards deconvolving a room impulse response from the recorded signal.

RIR Simulator testing should be done in an isolated machine room to map the rooms impulse response. To remove reverb from a signal, there is a need to have a recorded dry sound to get the RIR, from e.g., a balloon pop. Since the recorded signal was recorded in a noisy environment, a good enough signal to get a deconvolved output was unobtainable. The reason is that the sound that gets sent through the simulator already had reverb in it. If the system were to be installed in a machine room, a hypothesis of covering the walls with sound absorbent, might get rid of all reverb in the recorded test signal. Followed, the simulator can get a short impulse response, from, e.g., a balloon pop to simulate how sound propagates in the machine room. Now that the rooms impulse is available and the simulator correctly set up, a recording of a single dry sound source is needed. This sound can be attached as a signal to a sound source in the code.

```
1 room.add_source(position=soundSource3DLoc, signal=soundData_i,
                  delay=0.0)
```

Listing 5.1: Code to add sound source to room and what signal it emits

The next step would be to remove all the wall absorption material and record the wet sound in

the machine room. Finally, one can deconvolve the wet sound with the rooms simulated impulse response and plot the data to see if the reverb is removed with the Wiener deconvolution, which filters away white noise and deconvolves.

Another approach to for deverberation is “Weighted Prediction Error," WPE. One could also try blind dereverberation to reduce the length of the reverb. The background noise and sound reverberation due to reflections in an enclosure are the two deteriorations within acoustic signal processing and far-field speech recognition. The WPE-method addresses the signal dereverberation approach, which is based on WPE for speech recognition and other far-field applications. WPE is a compelling method to blindly dereverberate acoustic signals based on long-term linear prediction [2].

Proper code should be low coupling, high cohesion. This was not fully implemented, this was beacuse the Pyhton files was started from MATLAB where it got some problems with importing libraries. This meant we had to move the functions needed from the libraries into the "RIR.py" file.

5.3 Source localization and separation

The group started the task of implementing beamforming into the thesis by obtaining information about the subject. As we wanted to make use of Python as the primary programming language, the research focused on the purpose of Python-beamforming and audio processing. The research lead to a a dead-end, as it was little to no information about the implantation of beamforming in the specifications of the project. After spending a lot of time without results, the group changed the strategy and redirected the focus to perform simple beamforming our self.

The group started to program a delay-and-sum beamformer from scratch. The group had more use of switching the IDE and programming language to something easier to test and visualise the data. The shift resulted in a change to MATLAB as the primary IDE and language. By calculating the distortion in each channel and summing up the signal, we ended up with an improved

directional signal. The output signal compared to the original signal was little to none, and not audible. However, by plotting this in the frequency domain, the group was able to spot the difference visually, witch opened for new ideas as to how to further enhance the signal.

As the group saw a visible difference in the beamformed signal, we started to compare two beamformed directional signals. By picking out the most significant frequency component and comparing each signal, the group could differentiate the sources and make two output signals. With this technique, frequency-components with the most significant magnitudes in each directional signal got separated. The separation of the source signals was a breakthrough for the project. Since the resulting output signals separated the two sources, and resulted in a far better output than expected. With some background noise, the separation was complete, and it was possible to listen to a specific source. The results from BSS is best noticeable when separating two or three sound sources containing music. This resulted in a close to fully separated output, which was a little muffled.

Further on, the group experimented in beamforming not only the direction of sources. By systematically calculating each angle in the azimuth orientation, the idea was to get a magnitude plot of the directional beamforming in each angle. In theory, the direction of a source should get the most considerable enhancement, and therefore a graph of each angle will form a wave-plot. By calculation or visibly finding the peak in the plot should result in finding the angle of origin of the source or also known as DOA.

The DOA estimation done by the group yielded promising results as we tried to increase the precision as high as possible. Especially single source calculation was showing good results. With a precision of -5 degrees, which could be removed by a bias, it was a better result than expected. However, with multiple sources, the calculation process started to get messier.

The multi-source DOA algorithm has some potential, but the lack of time prevented us in making a decent algorithm. Because of limited resources, the dependency on decent recordings was too high, which made the results undesirable. The code works partially, and sometimes with a high deviance when estimating the angle of arrival. One of the problems also lies in the ability to gather reliable recordings, as the equipment malfunctioned near the end of the thesis. A

solution to the problems is to use more time researching better algorithm.

5.4 Project evaluation

5.4.1 Workflow

This section is about how the group as a whole cooperated. When writing the preliminary report, the group imagined that we would work together on every task to solve issues efficiently. However, when the project evolved, the categories were split up into three different sections, filtering and echo cancellation, DOA triangulation and beamforming. These topics are so wast that the group members could not immerse themselves into every subject. This directly led to each member having their own subject to study. The split resulted in less cooperative teamwork as imagined, which is less effective in the long run. With proper collaboration, the group leader takes charge, motivates for progression and pushes for results on all topics.

On the other hand, each group member had their responsibility to achieve results. From the solitude, we learned to stay focused and motivate ourselves, which has led to a continuous workflow throughout the project. It is hard to say if we would be better off working together or single-handedly. The three group members are all stubborn, and does not give up easily when a topic is up for discussion. This always leads to the best solution this group can achieve. Maybe we needed more cooperation on every subject, to have come up with a better solution, or perhaps peace at mind for thought was more efficient.

The different working-structural software was used to get an overview of the group assignments. As mentioned in the preliminary report, the group used programs such as GIT, Microsoft Teams and GANTT. The use of software, as mentioned, has come in great use after the change in working conditions during the project.

5.4.2 Workflow during the Covid-19 pandemic

During the thesis, the Covid-19 pandemic broke loose. With an impact on the working conditions, the group made a risk evaluation of the situation during the project, see Appendix B. In the report, the group mentioned the change in working conditions and evaluated the loss of time, which is our most pressured resource. There were also other obstacles when dealing with the Corona-virus. The group had problems with the arrival time of the components. Since the components were shipped from China, the local Chinese shipments were delayed, and the ReSpeakers did not arrive until five weeks after the order was placed. This delay caused us to start our research on the hardware later than expected. Another obstacle was the universities, schools, kindergarten and our workplace closed. This prevented us from being able to simulate and create a test room at Seaonics or our university. The total time loss during the pandemic was calculated to 15% decrease, as in the report, but after the conditional changes, everything did not go as planned.

A few conditional changes were not foreseen in the COVID-19 report. For instance, the change in the test location was not foreseen, where the impact is mentioned in the report. Furthermore, the group had predicted a constant loss of effectiveness, but as seen at the end of the thesis, the efficiency is more like a curve, and lower than anticipated.

Chapter 6

Conclusions

The purpose of this project was to research if a low-cost solution to an APM system is achievable.

A goal was set on how to reach our objectives. Several ides were optimistic, but not all issues were foreseen, such as corona. Therefore, some reduction of objectives was necessary.

Hardware problems resulted to faulty recordings, and only a small data-set of correct test-data were made.

Adaptive filter without reference source are not achievable in this thesis because of time limitations and complexity. The RIR simulator needs a proper test facility and temporary wall absorbents. Wiener deconvolvution to get a reverbless sound and WPE needs more research.

More research on adaptive beamforming techniques are required. The BSS separation of the signals work, but the code can be improved since the algorithm is better in some scenarios than others. The DOA algorithm gives unstable results on multi-source recordings. But single source DOA estimates the exact angle within 5 degrees. The technical GUI with the combination of back-end gave ability to calculate and visual data. The application has the potential to perform the tasks of module two. With more time and resources, and by implementing the remaining modules, the possibility to produce a low-cost APM system is achievable.

6.1 Further work

Further work into module two of APM is first to research more into the fields of beamforming and echo cancellation. The group also recommends a better testing location and equipment. With better testing methods, topics as echo cancellation and source localisation could get an better result.

- Implement a better beamforming technique
- Set up a better test location
- Get new hardware for recording and testing
- Complete the echo cancellation method
- Implement the remaining modules

Appendices

A Preliminary report

Forprosjektrapport

Leanrapport

B Risikorapport og vurdering Corona

C Gantt diagram

D Timesheet

E Code

BBS2S.m

BBS3S.m

chunckBBS2S.m

chunckBBS2S.m

DOAmultisource.m

record.py

UCA.m

indexDevices.py

RIR.py

DOASingleSource.py

F Test reports

Test rapport Beamforming Acoular

Test rapport Beamforming Gain lobes

Test rapport BSS

Test rapport Odas_web

Test rapport Record From PC

Test rapport Record Respeaker

Test rapport Signal prosessering Delay

Test rapport Triandulering 1

Test rapport Triandulering 2

G Meetings controll group

Meeting report 2019.12.06

Meeting report 2020.01.14

Meeting report 2020.01.31

Meeting report 2020.02.10

Meeting report 2020.02.28

Meeting report 2020.03.17

Meeting report 2020.03.24

Meeting report 2020.03.31

Meeting report 2020.04.07

Meeting report 2020.04.14

Meeting report 2020.05.05

Meeting report with Kai 2020.03.25

H Internal weekly reports

Week 3

Week 4

Week 5

Week 6

Week 7

Week 8

Week 9

Week 11

Week 12

Week 13

Week 14

Week 15

Week 16

Week 17

Week 18

Week 19

Week 20

Bibliography

[1] Audio Engineering 101. Dry vs wet video, 2020. URL https://www.youtube.com/watch?v=vGIb48Gm1N8.

[2] Simon Doclo Ante Jukić. Speech dereverberation using weighted prediction error with laplacian model of the desired signal, 2020. URL https://ieeexplore.ieee.org/abstract/document/6854589?casa_token=_HmKdnOUv0oAAAAA:RVLu-cayfHyh74XxvIBMu5DaI6xRm6S8m4nTLx-GmtKJcP_IlHCFRH4ztadt9brSUwd89yQAjQ.

[3] bergos. Respeaker core v2 case, 2018. URL https://www.thingiverse.com/thing:3065699.

[4] ch0. Respeaker mic array v2 case, 2019. URL https://www.thingiverse.com/thing:3366507.

[5] ch0. Project on github, 2020. URL https://github.com/askjong/Acoustic-Condition-Control.

[6] ch0. Respeaker core v2 git, 2020. URL https://github.com/respeaker/usb_4_mic_array.

[7] ch0. Usb-aio16-16f, 2020. URL https://www.accesio.com/?p=/usb/usb-aio16-16a.html.

[8] ch0. Respeaker core v2 wiki, 2020. URL https://wiki.seeedstudio.com/ReSpeaker_Core_v2.0/.

[9] ch0. Respeaker mic array v2 wiki, 2020. URL https://wiki.seeedstudio.com/ReSpeaker_Mic_Array_v2.0/.

[10] cpython. Odas: Open embedded audition system, 2020. URL https://docs.python.org/3/library/wave.html.

[11] DAGA. Extending the closed form image source model for source directivity, 2020. URL https://www2.ak.tu-berlin.de/~akgroup/ak_pub/2018/000458.pdf.

[12] Elfa Distrelecs. Max4466 mikrofonforsterker, February 2020. URL https://www.elfadistrelec.no/no/max4466-mikrofonforsterker-5v-adafruit-1063/p/30091129.

[13] Elfa Distrelecs. Respeaker core v2.0, 2020. URL https://www.elfadistrelec.no/no/respeaker-core-v2-seeed-studio-102990883/p/30109501.

[14] Elfa Distrelecs. 8-ch 12-bit adc for raspberry pi, 2020. URL https://www.elfadistrelec.no/en/ch-12-bit-adc-for-raspberry-pi-seeed-studio-103030280/p/30135129?queryFromSuggest=true.

[15] Elfa Distrelecs. Respeaker mic array v2.0, 2020. URL https://www.elfadistrelec.no/no/respeaker-mic-array-v2-seeed-studio-107990053/p/30109517.

[16] Torbjoern Flor. Iterating uca, 2020. URL https://imgflip.com/gif/3x0152.

[17] Sylvain Gugger. Convolution in depth, 2020. URL https://sgugger.github.io/convolution-in-depth.html.

[18] Sangdeok Kim and Jong-Wha Chong. An efficient tdoa-based localization algorithm without synchronization between base stations. *International Journal of Distributed Sensor Networks*, 11(9):832351, 2015. doi: 10.1155/2015/832351. URL https://doi.org/10.1155/2015/832351.

[19] LCAV. pyroomacoustics, 2020. URL https://github.com/LCAV/pyroomacoustics.

[20] Lumen Learning. Superposition principle, 2020. URL Superpositionhttps://courses.lumenlearning.com/boundless-physics/chapter/interactions-with-sound-waves/.

[21] MAXIM. Low-cost, micropower, sc70/sot23-8, microphone preamplifiers with complete shutdown, 2020. URL https://cdn-shop.adafruit.com/datasheets/MAX4465-MAX4469.pdf.

[22] ODAS. Odas: Open embedded audition system, 2020. URL https://github.com/introlab/odas.

[23] Brian O'Keefe. Finding location with time of arrival and time difference of arrival techniques, 2020. URL https://sites.tufts.edu/eeseniordesignhandbook/files/2017/05/FireBrick_OKeefe_F1.pdf.

[24] Acoustic panels Review. Acoustic sound absorption, 2020. URL https://www.acousticpanelsreview.com/acoustic-sound-absorption/.

[25] Phonon. What methods can be used to identify and remove echo from an audio system?, 2020. URL https://dsp.stackexchange.com/questions/338/what-methods-can-be-used-to-identify-and-remove-echo-from-an-audio-system.

[26] Power. Jbl go2 bÆrbar bt hØyttaler navy, 2020. URL https://www.power.no/hoeyttalere-og-lyd/hoeyttalere/jbl-go2-baerbar-bt-hoeyttaler-navy/p-894953/.

[27] RS-Online. Raspberry pi 4 model b 4g sbc, 2020. URL https://no.rs-online.com/web/p/products/1822096/.

[28] Seed. Respeaker core v2.0, 2020. URL https://wiki.seeedstudio.com/ReSpeaker_Core_v2.0/.

[29] Seed. Respeaker 8-channel 12-bit adc for raspberry pi, 2020. URL https://wiki.seeedstudio.com/8-Channel_12-Bit_ADC_for_Raspberry_Pi-STM32F030/#feature.

[30] Håvard Straum. Prediksjon av romakustiske forhold i rom med ujevn absorpsjonsfordeling, 2020. URL ntnuopen.ntnu.no/ntnu-xmlui/bitstream/handle/11250/2370590/566547_FULLTEXT01.pdf?sequence=1&isAllowed=y.

[31] ThermaXx. Sabine's formula and the birth of modern architectural acoustics, 2020. URL https://www.thermaxxjackets.com/sabine-modern-architectural/.

[32] Vocal. Room impulse response, 2020. URL https://www.vocal.com/dereverberation/room-impulse-response/.

A

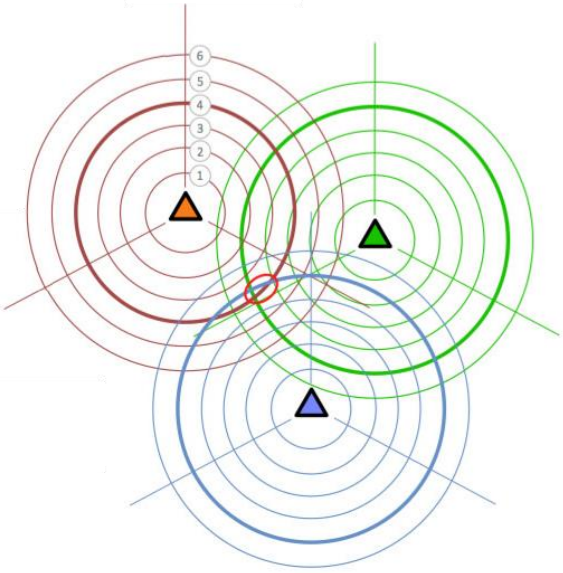
FORPROSJEKT - RAPPORT

FOR BACHELOROPPGAVE

NTNU
Kunnskap for en bedre verden

Forprosjektoppgave

IE303612 Bacheloroppgave



Forsposjektoppgave 2020

Andreas Skjong
Finn-Christian Eriksen
Torbjørn Flor

Totalt antall sider inkludert forsiden: 33

Ålesund, 30.01.2020

NTNU
Kunnskap for en bedre verden

Postadresse	Besøksadresse	Telefon	Bankkonto
NTNU i Ålesund	Larsgårdsvegen 2	73 59 50 00	7694 05 17431
N-6025 Ålesund	**Internett**	**Epostadresse**	**Foretaksregisteret**
Norway	www.ntnu.no	postmottak@ntnu.no	NO 974 767 880

FORPROSJEKTRAPPORT - BACHELOROPPGAVE

TITTEL:

Akustisk tilstandskontroll

KANDIDATNUMMER (NAVN):

Andreas Skjong
Finn-Christian Eriksen
Torbjørn Flor

DATO:	EMNEKODE:	EMNE:	DOKUMENT TILGANG:
30.01.2029	**IE303612**	**Bacheloroppgave**	**Åpen**

STUDIUM:	ANT SIDER/VEDLEGG:	BIBL. NR:
Automatiseringsteknikk	25 / 8	**Ikke i bruk**

VEILEDER:

Ottar L. Osen
Stig Espeseth
Tom Jørann Giske

SAMMENDRAG:

I denne rapporten har gruppen prosjektert, planlagt og spesifisert hvilket mål som skal oppnås i prosjektet. Gjennom god planlegging, bruken av verktøy som Gantt og Teams, skal gruppen gjennomføre prosjektet innenfor planlagte tidsrammer. De ulike modulene i oppgaven skal være planlagt på en slik måte at effektiviteten og gjennomføringsevnen blir opprettholdt for en god flyt gjennom semesteret.

Oppgaven gruppen ønsker å utforske muligheten til å bruke akustikk som et verktøy i industrien. Spesifikt skal vi se på mulighetene for å identifisere lydkilder og konsentrere enkeltkilder for videre bruk i tilstandskontroll. Hensikten med denne oppgaven er å få en forståelse og kunnskap til å kunne utvikle et produkt sammen med oppgavegiver på lang sikt. Gruppen skal studere mulighetene rundt temaet og lage et verktøy som kan videreutvikles.

Denne oppgaven er et forprosjekt utført av studenter ved NTNU i Ålesund.

Ålesund 30.01.2020

Andreas Skjong

Finn-Christian Eriksen

Torbjørn Flor

INNHOLD

INNHOLD - FIGURER

1 INNLEDNING

Et konsept som er populært for tiden er tilstandsbasert vedlikehold. Tanken bak denne oppgaven er å tilegne oss kunnskap om tilstandsbasert vedlikehold ved bruk av akustikk. Tilstandsbasert vedlikehold med bruk av akustikk er et lite utforsket felt, derfor ønsker gruppen å se videre på dette. Hensikten med et tilstandsbasert vedlikeholds-system kan være å redusere nedetid på anlegget, kunne planlegge reparasjon og gjøre kostnadsbesparelser.

Denne bacheloroppgaven er gitt av Seaonics. Oppgaven er et steg i en annen retning enn de mest brukte tilstandsbaserte vedlikeholds-metodene. Disse metodene er ofte kostbare, og krever mer utstyr. Ønsket fra oppdragsgiver er å utforske nye muligheter for å oppnå tilstandsbasert vedlikehold.

2 BEGREPER

Begrep	Beskrivelse
MS	Microsoft
VSC	Version Control System
AI	Artificial Intelligence

3 PROSJEKTORGANISASJON

3.1 Prosjektgruppe

Studentnummer(e)	Studentnavn
484219	Andreas Sønderland Skjong
484301	Finn-Christian Eriksen
483992	Torbjørn Inge Flor

3.1.1 Oppgaver for prosjektgruppen – organisering

Gruppa kommer til å benytte seg av Teams og Gantt-skjema. Det blir brukt for å delegere oppgaver etter prioritet, og for å synkronisere oss. Teams er et verktøy der man blant annet legger inn prosjektoppgaver med tidsfrister.

Navn	Rolle
Andreas Sønderland Skjong	Prosjektleder
Finn-Christian Eriksen	Prosjektsekretær
Torbjørn Inge Flor	Prosjektingeniør

3.1.2 Oppgaver for prosjektleder

- Hovedkontaktperson.
- Delegere oppgaver ved nødvendighet.
- Ansvarlig for progresjon.

3.1.3 Oppgaver for sekretær

- Vara for prosjektleder.
- Ansvarlig for møter (invitasjon, rapportskriving og romreservasjon).

3.1.4 Oppgave for alle medlemmer

- Hvert medlem er ansvarlig for å fullføre hver oppgave som er utdelt til dem.
- Hvert medlem har skal jobbe med bacheloren fra kl. 9-15. Med unntak undervisning i Industri 4.0.
- Hvert medlem skal skrive timeliste i Teams.

3.2 Styringsgruppe (veileder og kontaktperson oppdragsgiver)

Navn	E-post	Rolle
Ottar L. Osen	ottar.osen@ntnu.no	Førstelektor automatiseringsteknikk
Stig Espeseth	stig.espeseth@seaonics.com	Managing director Seaonics
Tom Jørann Giske	tom.giske@seaonics.com	Manager digital solutions

4 AVTALER

4.1 Avtale med oppdragsgiver

Etter møtet 14.01.2020 ble styringsgruppa og gruppemedlemmene enige om en avtale. Hovedfokuset blir flyttet bort fra, som først ble diskutert på møte den 06.12.2019, til at gruppa skal lage et system som kan selektere ut lyden på én maskin i rommet. Lyden skal deretter tas inn i et program som filtrerer bort all annen lyd i rommet. Hensikten med dette er at en operatør kan sitte på en annen lokasjon å lytte på en maskin og behandle resultatet. Grunnen for dette skiftet er at dette systemet har mer nytteverdi for oppdragsgiveren.

Gruppen får tilgang til et kontor med 4 sitteplasser, der vi kan jobbe på hverdager innenfor normal arbeidstid.

4.2 Arbeidssted og ressurser

- Kontor hos arbeidsgiver.
- Testlokale hos arbeidsgiver (Et simulert maskinrom).
- Materiale blir levert av NTNU Ålesund - IIR
- Tilgang til rådgiver, Ottar L. Osen.

4.3 Gruppenormer – samarbeidsregler – holdninger

Kjernetid mellom 9-15 mandag-fredag.
Ideell arbeidstid mellom 8:15-16 mandag-fredag.
Ferie: Påske mellom 4.-13. april.
Bachelor innleveringsfrist: 20. mai.

- Hvis et gruppemedlem ikke kan møte i kjernetiden satt av gruppen, må resten av gruppen bli informert om dette på forhånd. Det skal også gjøres opp eventuelle tap av arbeidstid.
- Arbeidstid skal loggføres ukentlig.

- Ukentlig møte mellom gruppen, dette skal avholdes i starten av uken for å avdekke arbeidsforhold og flyt.
- Industri 4.0 er prioritert fremfor bachelor-arbeid når forelesninger avholdes. Dette på grunn av obligatorisk oppmøte i faget.
- Det er viktig at oppgaver blir gjort i henhold til planen og tidsfrister. På grunn av dette må gruppens medlemmer være villig til å jobbe overtid slik at disse målene innfris. Dette skal være overhold av personen ansvarlig for oppgaven.
- Gruppens medlemmer må i sitasjoner der arbeidsflyt er hindret gi beskjed til ansvarlig av oppgaven eller eventuelt resterende medlemmer.
- Det er ønskelig at gruppens medlemmer har alle ansvar for en eller flere oppgaver til enhver tid, dette for å holde best mulig flyt i prosjektet.

Som ingeniør er sikkerhet den viktigste faktoren. Prosjektet skal utføres på en slik måte at liv og helse ikke står i fare. Utviklingen av ny teknologi er kontinuerlig, og dette medfører at som ingeniør, må man tenke på sikkerhet framfor noe annet. Et annen viktig faktor er de etiske og moralske aspektene ved utvikling av ny teknologi. Disse er det viktig å reflektere over når man utvikler et nytt produkt, slik at man ikke produserer noe som kan skade miljø, helse eller liv.

5 PROSJEKTBESKRIVELSE

5.1 Problemstilling - målsetting – hensikt

Gruppen har som mål å utforske mulighetene for bruk av akustikk til å vurdere tilstanden utstyr i et maskinrom. Det skal utforskes i bruk ulike metoder for filtrering av lyd for å lokalisere, detektere og isolere lydkilder i rommet. Gruppen skal også utforske muligheten ved feildeteksjon av utstyret ved hjelp av lyd.

Hensikten med denne oppgaven er å få en forståelse og kunnskap til å kunne utvikle et produkt på lang sikt. Gruppen skal studere mulighetene rundt temaet og lage et verktøy som kan videreutvikles.

Som videre utvidelser til oppgaven har gruppen ulike ideer som er interessante å bygge videre på. Dette er mål for et eventuelt produkt og ikke for selve gruppen. Dette som å lage et grafisk brukergrensesnitt for dataene av det overvåkede utstyret, implementere kunstig intelligens til å diagnostisere og prognosere utstyret og muligheten for videreutvikling av ideen til praktisk bruk.

Gruppen vil først oppnå:

- Detektere lydkilder
- Lokalisere lyd
- Isolere lydkilder
- Grafisk fremstilling av data
    - Varmekart
    - Lydkilder
    - Grafisk trender
- Se på mulighetene for Anomal tilstandskontroll
    - Prediksjon filter
    - Ekspert system
    - Neural Network

- Menneskelig deteksjon grafisk
- Menneskelig deteksjon akustisk

Senere utvidelser ved mulighet:

- GUI tilrettelagt bruker
- Diagnosesystem
- Prognosesystem
- Design av produkt

5.2 Krav til løsning eller prosjektresultat – spesifikasjon

Gruppen krever etter endt prosjekt å oppnå følgene: bekrefte muligheten for akustisk tilstandskontroll av maskinrom. Gruppen skal ha utforsket ulike metoder for å oppnå dette. Med denne bacheloren skal gruppen sitte igjen med kompetanse som skal kunne videreformidles i rapporten. Gruppen skal også prøve seg så lage et utkast av en programvare.

Gruppen skal ha studert feltene: behandling av lydsignaler, på en slik måte at deteksjon og isolasjon av lydkilder er mulig. tegning av lyd-omgivelsene for representasjon av lyd-rommet. Samt anomalt detektere feil eller mangler på overvåket utstyr.

5.3 Planlagt framgangsmåte(r) for utviklingsarbeidet – metode(r)

5.3.1 Planlegging

Planleggingen skal gjøres i Teams. I Teams kan vi legge inn arbeidstimene, planlegge hva som skal gjøres og tildele disse oppgavene til personer, legge inn Gantt-skjema, ha video-møter og legge inn filer. Dette programmet hjelper oss å overholde tidsfrister på oppgaver som må gjøres først, og er derfor en god måte å strukturere oss på.

5.3.2 Framgangsmåte

Prosjektet skal deles inn i ulike moduler og delmål som gruppen skal ta for seg. Det ønskes at gruppen alltid kjører to mindre oppgaver/moduler parallelt til enhver tid, for å holde en effektiv arbeidsprosess. Hovedmodulene i prosjektet består av: Hardware, Software, test, sluttprosess og rapport.

Gruppa ser for seg at det skal startes med å finne informasjon rundt temaet, grave litt i dybden slik at vi kan tegne oss et bilde av utstyret vi trenger. Utstyr som lett anvendbare mikrofoner, der vi vil ha ut ufiltrert lyd. Mikrofonen må ha egenskapene til å detektere frekvensen til ulikt utstyr som for eksempel kulelagrene på en elektromotor. I tillegg må lyden være enkel å hente ut og importeres i Python for filtrering. En siste ting er at sampleraten må være høy for å kunne detektere toppene i lydsignalet.

Videre ønsker gruppen å utforske muligheten for posisjonering, seperasjon og lytting på ulike lydkilder. Dette for å kunne filtrere ut uønsket støy og innhenting av spesifikk data. Vi ønsker å gjøre dette ved bruk av flere mikrofoner og triangulere disse mot hverandre. Det filtrerte signalet kan da kjøres gjennom ulike prosesser for å kontrollere tilstanden til lydkilden.

For å kontrollere tilstanden til utstyret har gruppen ulike teknikker som vi ønsker å utforske, en av dem er prediksjons filter. Dette filteret samens med er to alternativer til tilstandskontroll.

Underveis i prosjektet skal gruppen skrive rapport til oppgaven. Rapporten skal inneholde gruppens Teori, fremgangsmåte, resultater, konklusjon og annet innhold som timelister.

5.3.3 Refleksjon/ ettertanke fra resultatene

Effekter av å flytte mennesket kan gi muligheten for økonomiske besparelser. En besparelse er ved å bruke dette verktøyet i flere rom gjør det mulig for en operatør å overvåke flere rom samtidig. En annen besparelse er at operatører slipper å oppholde seg under støyende områder som kan vi hørselsbevarelser.

5.4 Informasjonsinnsamling – utført og planlagt

Hva vi har funnet:

- Lest enkelte prosjekter som ligger åpent på nett som omhandler temaet.
- Utført et første møte med styringsgruppen og oppdragsgruppen om ønsket resultat etter endt prosjekt.
- Utforsket og kontrollert testanlegg som skal brukes i prosjektet.
- Forhørt oss med rådgivere innom feltene Kunstig intelligens og signalbehandling.

Hva som gjenstår:

- Finne mer informasjon om kunstig intelligens og mulighetene til å bruke dette i prosjektet på best mulig måte.
- Finne den beste måten å behandle lydsignaler på, filtrere disse og bruke dem til ønsket formål.
- Forhøre oss om muligheten til lån av stasjonær datamaskin.

5.5 Vurdering – analyse av risiko

Frekvensen av tilfeller	Alvorlighetsgrad av konsekvens				
	Veldig lav	Lav	Middels	Høy	Veldig høy
Veldig høy					•Gruppen setter seg fast i prosjektet
Høy		• Støyskader • Øyeskader • Sykdom			• Oppgaven er for omfattende
Middels			• Mangel på ressurser	• Usette problemer	• Mekaniske klemskader • Brann i testanlegg
Lav		• Skader ved bruk av verktøy • Skader på arbeidsplassen	• Sen leveringstid • Stress og overarbeide	• Strømm-gjennomgang	
Veldig lav				• Flyvende objekter	

Personell Risiko	Forklaring	Tiltak
Brann i testanlegg	Brann oppstår i elektrisk utstyr på/i testutstyr	Vite nødutganger og nærmeste slokkeutstyr
Mekaniske klemskader	Klemskader mellom bevegelige deler	Jobbe på utlagt/frakoblet utstyr, mekanisk sperring av roterende utstyr ved drift.
Strømgjennomgang	Strømgjennomgang i kroppen under arbeid på elektrisk utstyr	jobbe sikkert ved å koble fra elektrisk
Øyeskader	Skader forårsaket av puss eller partikler som har kommet inn på øyet	bruk av briller ved saging, boring og filing
Støyskader	Støyskader ved arbeid over tid under støyende forhold	bruk av øreproppper ved støy over 56 dB eller mer
Flyvende gjenstander	mekanisk utstyr bryter under press og kommer i fart mot personell	bruk av mekanisk beskyttelse ved roterende deler
Skader ved bruk av verktøy	Bruk av verktøy som kniv, skrujern, baufil og annet utstyr kan føre til kuttskader	varsomt arbeid ved bruk av verktøy eller eventuelt bruk av hansker

Prosjekt Risiko	Forklaring	Tiltak	Handling
Oppgaven er for omfattende	Oppgaven er for ukonkret eller det er strid i oppfatning av oppgaven.	Bli enig på forhånd på hva som er et ideelt mål og hva som er realistisk mål.	Reduserer oppgaveomfanget/kompleksiteten

Gruppen setter seg fast i prosjektet	under prosjektet kan gruppen sette seg fast og ikke komme seg videre på grunn av mangel på erfaring	Ha en plan på hva man skal gjøre ved stagnering/sitter fast.	bruke rådgivere som har kunnskap innom feltene, K Erik Hoff og Ibrahim A. Hameed
Usette problemer	Problemer gruppen ikke kan forutse på forhånd som stagnerer flyten til gruppen	Planlegge prosjektet godt og legge til ekstra tid for usette problemer.	Bruk av rådgivere ved usette problemer
sykdom	Sykdom er en normal del av arbeidsdagen og kan ikke gjøres så mye med	God hygiene og ta pauser er viktige faktorer for å redusere sykdomsfravær	Personer som blir syk må holde seg hjemme for å unngå spredning
Mangel på tilgang på rådgiver	Hvis noe uforutsett skjer med rådgiver eller muligheten for å kontakte personen er redusert.	Ha flere kontaktpersoner for å redusere avhengigheten og opprettholde redundans.	Se på mulighetene for å kontakte noen innom fagfeltet eksternt.
Mangel på ressurser	Mangel på økonomisk støtte for innkjøp av utstyr er noe som vil påvirke prosjektets gang. Men kravet til utstyret kan reduseres etter tilgang på ressurser	Se på mulighetene for bruk av billigere hardware, eks å lage noe av det selv.	Se på andre muligheter for å få tilgang på ressurser.

5.6 Framdriftsplan – styring av prosjektet

Nr	Aktiviteter	Tid
A1	**Forprosjekt**	
A11	Skriving av forprosjekt rapport	4
A12	Undersøke - innhenting av informasjon	4
A13	Planlegge oppgaver, tidsbruk og …	3
A14	Implementere LEAN i bacheloren	2
A2	**Hovedprosjekt**	
A21	Skrive bachelor	14
A22	Rettskrive bachelor	2
A3	**Hardware**	
A31	Valg av hardware	1
A32	Bestille inn hardware	1
A33	Montere hardware	1
A4	**Software**	
A41	Funksjon for å hente inn lydsignal fra mikrofon for testing	2
A42	Finne ut om filtrering av støy/ekko	3
A43	Finne ut om triangulering	3

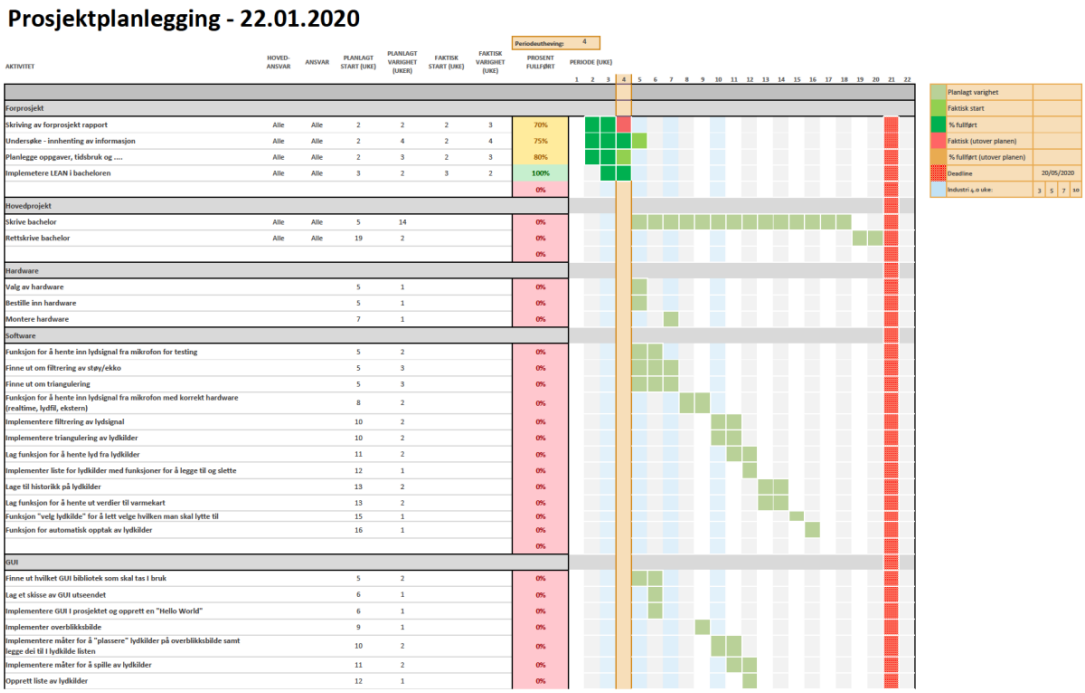
A44	Funksjon for å hente inn lydsignal fra mikrofon med korrekt hardware (realtime, lydfil, ekstern)	2
A45	Implementere filtrering av lydsignal	2
A45	Implementere triangulering av lydkilder	2
A46	Lag funksjon for å hente lyd fra lydkilder	2
A47	Implementer liste for lydkilder med funksjoner for å legge til og slette	1
A48	Lage til historikk på lydkilder	2
A49	Lag funksjon for å hente ut verdier til varmekart	2
A50	Funksjon "velg lydkilde" for å lett velge hvilken man skal lytte til	1
A51	Funksjon for automatisk opptak av lydkilder	1
A6	**GUI**	
A61	Finne ut hvilket GUI-bibliotek som skal tas I bruk	2
A62	Lag en skisse av GUI utseendet	1
A63	Implementere GUI I prosjektet og opprett en "Hello World"	1
A64	Implementer overblikks bilde	1
A65	Implementere måter for å "plassere" lydkilder på overblikks bilde samt legge dem til I lydkilde listen	2
A66	Implementere måter for å spille av lydkilder	2
A67	Opprett liste av lydkilder	1
A68	Implementere varmekart	2
A69	Lage til trend for historikk på lydkilder	2
A70	Implementer funksjon for "automatisk" opptak av lydkilder	1
A8	**Testing**	
A81	Testing henting av lyd	1
A82	Testing triangulering	1
A83	Fullskala test	1

5.6.1 Hovedplan

Det har blitt opprettet et Gantt skjemaet i Excel. Der står alle større arbeidsoppgaver, mens mindre arbeidsoppgaver står i Planner. Gantt-skjemaet er dynamisk. Det gjør at det grafikken oppdaterer seg automatisk i forhold til statusen på oppgava.

Det er viktig at gruppen følger arbeidsplanen nøye. Hvis den ikke er oppfølgt så kan andre oppgaver bli forskjøvet. Det kan føre til at bacheloren ikke møter tidsfristen.

Hvem som har ansvar for oppgavene velges underveis avhengig av belastning på gruppemedlemmene. Ansvaret for oppgavene blir fordelt av prosjektleder. Statusen på oppgavene tas opp på hvert mandagsmøte. Normalt er det en oppgave per person. Men hvis det blir for mye belastning på en person eller at en oppgave blir for vanskelig, settes to personer på oppgaven. Det blir gjort på denne måten for å oppnå LEAN med One-Piece-Flow (se. Vedlegg Lean).



Figur 1 Gantt skjema

5.6.2 Styringshjelpemidler

Gruppen vil bruke plattformen SharePoint for å samle alle dokumenter, rapporter og annen informasjon som omhandler prosjektet. Sammen med applikasjonen MS Teams kan vi gjør at all informasjon blir tilgjengelig for

alle som har tilknytning til prosjektet, og er hoved kommunikasjonsplattformen for gruppen.

I Teams brukes ulike tilleggsapper som Shift som er for å registrere arbeidstimer, fravær, ferie eller andre årsaker for at noen i gruppen ikke kan møte. Det hjelper oss med å planlegge oppgaver ettersom at personer ikke har muligheten.

Bruker et Gantt skjema i Excel som er tilknyttet MS Planner for en grafisk visning av arbeidsoppgavene. Gantt legges inn arbeids oppgaver som er direkte tilknyttet prosjektet angår programvare og utstyr. Bruker Planner for å beskrive lettere arbeidsoppgaver, og for å se hvilke oppgaver som er jobbes med. Samt andre oppgaver som ikke står i Gantt skjemaet som spørsmål til styringsgruppen.

Applikasjoner som er tilknyttet SharePoint:

- Teams
- OneDrive
- Shift
- Planner

5.6.3 Utviklingshjelpemidler

For hjelp med utvikling av systemet, er de viktig å ha god kontroll på versjoner og god kodestil med at dokumenterer koden og tenker på coupling og cohesion. Hjelpemidler som VSC og en god IDE gjør det kodingen lettere og mer effektivt.

Hjelpemidler:

- PyCharm IDE, som er et utviklingsverktøy spesielt utviklet for programmeringsspråket Python.
- GitHub, som er en VSC som har direkte støtte til PyCharm IDE.

5.6.4 Internkontroll – evaluering

Det blir en gjennomgang med styringsgruppen 14.01.2020 der vi går igjennom hva vi har gjort og hva som skal gjøres videre på prosjektet for å kunne oppnå målene.

Samt skal gruppen ha møte kvar mandag for å gå gjennom statusen på oppgavene, om vi ble ferdige eller kva som må til for å fullføre påbegynte oppgaver. For å forhindre at noen sitter fast på en oppgave eller om hva problemer som kan oppstå ved nye oppgaver, slik at vi ikke stopper opp og for en god flyt i prosjektet.

5.7 Beslutninger – beslutningsprosess

Hver større beslutning angående prosjektet blir satt av gruppen sammen med styringsgruppen, mens mindre spørsmål angående oppgaver blir satt av kun gruppen. Hvis en beslutning om å endre en oppgave blir satt, blir Gantt-skjemaet og Planner oppdatert.

6 DOKUMENTASJON

6.1 Rapporter og tekniske dokumenter

Det blir laget ferdige maler for møterapporter og test rapporter. I møterapporten så skal det stå hvem som var med på møtet og hva det inneholdt, samt statusen om prosjektet.

I test rapporten skal det stå hva testen omhandler og om testen er godkjent eller ikke, sammen med begrunnelse.

Alle dokumenter, bilder og annet innhold som er hentet fra andre, skal det oppgis kilder i formatet Chicago, femtende utgave.

7 PLANLAGTE MØTER OG RAPPORTER

7.1 Møter

7.1.1 Møter med styringsgruppen

Etter hvert møte med styringsgruppen skal det tas en gjennomgang i gruppa for å gå igjennom hva som diskutert. Det er for å sikre at alle forstod hva som ble gjennomgått og hva tiltak som må gjøres for å oppnå det som blei diskutert.

7.1.2 Ukes møter

Gruppen møtes hver mandag kl.09:15. for et statusmøte. Her skal vi diskutere hva som ble gjort tidligere uke, samt diskutere hva som gjenstår av påbegynte oppgaver, og hva som skal til for å oppnå dem. Skal også bli en gjennomgang av tidligere oppgavers påvirkning på kommende uke.

Møtet blir holdt av prosjektleder, mens sekretær skriver en møterapport. Alle rapportene blir lagret i SharePoint.

7.2 Periodiske rapporter

Det skal skrives en kort rapport fra hvert møte med styringsgruppen og dem ukentlige interne møtene, samt andre møter som oppstår underveis under prosjektet.

Formålet med rapportene er å finne ut hvor langt vi har kommet og hva som skal gjøres videre i prosjektet. Rapportene vil også hjelpe oss med å finne ut hva som eventuelt var en feil, hvorfor den oppstod og hvordan vi løste den, for å unngå nye feil av samme typen.

8 PLANLAGT AVVIKSBEHANDLING

Ved uplanlagte utfordringer eller hindringer av fremdrift skal disse stegene følges:

1. Gi felles beskjed for å opplyse gruppens medlemmer.
2. Kontakt nødvendig hjelp om det er mulig.
3. Gi beskjed til prosjektansvarlig.
4. Opplyse oppdragsgiver.

Ved nye endringer av oppgaven skal disse stegene gjennomføres.

1. Gruppen skal informeres og lage et utkast av endringene og hvilke konsekvenser det medbringer.
2. Rådgiver informeres.
3. opplyse oppdragsgiver om endringer.

9 REFERANSER

NTNU. *NTNU.* u.d. https://innsida.ntnu.no/wiki/-/wiki/Norsk/Laboratorie-+og+verkstedh%C3%A5ndbok (funnet 01 08, 2020).

10 VEDLEGG

Nr	Beskrivelse
1	LEAN rapport

LEAN- RAPPORT

FOR BACHELOROPPGAVE

NTNU
Kunnskap for en bedre verden

Lean

IP300416 Industri 4.0 (2020 VÅR)

Leanrapport

Andreas Skjong
Finn-Christian Eriksen
Torbjørn Flor

Totalt antall sider inkludert forsiden: 8

NTNU
Kunnskap for en bedre verden

Ålesund, 21.01.2020

Postadresse	**Besøksadresse**	**Telefon**	**Bankkonto**
NTNU i Ålesund	Larsgårdsvegen 2	73 59 50 00	7694 05 17431
N-6025 Ålesund	**Internett**	**Epostadresse**	**Foretaksregisteret**
Norway	www.ntnu.no	postmottak@ntnu.no	NO 974 767 880

INNHOLD

INNHOLD - FIGURER

TERMINOLOGI

Begreper

Begrep	Beskrivelse
5S	Grunnleggende filosofi og gode holdninger i produksjon
PDSA/PDCA	Plan – Do -Study -Act / Plan – Do - Check - Act
Kaizen	Stadige forbedringer
Gemba	Der det skjer. F.eks. Gå gemba i en fabrikk
Heijunka	Utjevne arbeid som er skjevfordelt mellom ansatte
Muda	Waste
JIT	Just in time

1 INNLEDNING

Problemstillingen gruppen har fått i bacheloren er å lage et verktøy for akustisk tilstandskontroll av maskineri. Prosjektet dreier seg i hovedsak om å utforske mulighetene for å flytte mennesket fra maskinrommet til et ekstert kontrollrom ved bruk av akustikk.

Bacheloroppgaven i prosjektet løses ved å dele opp den store oppgaven i mindre deloppgaver. På denne måten fokuserer gruppen på nettopp disse små oppgavene i Leanrapporten.

2 TEORETISK GRUNNLAG

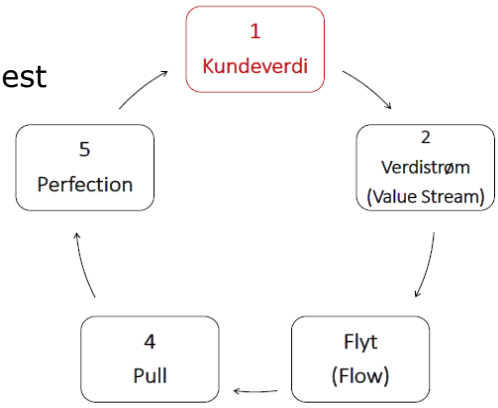
Lean er en filosofi som kort forklart er å forstå, skape og lære av arbeidsprosessene våre. Det å forstå hva som skjer i en arbeidsprosess. Klare å skille ut hva som er sløsing og hva som tilfører verdi til produktet. Skape flyt i arbeidsprosessen, lage en arbeidsprosess som ikke inneholder flaskehals. Ta lære av endringer som medbringer positive resultat, men også negative. LEAN er ikke å direkte øke produktivitet, men å fjerne sløsing.

Et av verktøyene som tar filosofien i bruk er 5S. 5S er Toyotas svar på LEAN og spesifiserer de ulike stegene for å oppnå filosofien. 5S står for: Sortere, Systematisere, Skinne, Standardisere og Sikre. 5S jobber for å fjerne såkalt "waste" (Muda), som er sløsing. Sløsing tilfører ikke verdi til produktet som blir laget, og burde derfor minimeres. Ved å fjerne sløsing, er det mulig å redusere kostnader som ikke er nødvendige for prosessen.

Figur 1 5S (Ask Lean u.d.)

3 METODE

For å finne ut hvordan gruppen skal jobbe mest mulig effektivt gjennom becheloren, ble det brukt Prosses Mapping. Dette ble gjort for å finne ut dei 5 ulike prosessene i prosjektet.



Figur 2, "Five Lean Fundamentals" (NTNU 2020)

1. **Kundeverdi**
   Raskere fremgang i prosjetet ved å bruke mindre tid på oppgaver som ikke gir verdi til prosjektet.

2. **Verdistrøm**
   Kartlegger alle prosesser hvordan oppgaver blir løst i prosjektet.

3. **Flyt**
   Bruker 5S for å fjerne sløsing og flaskehalser.

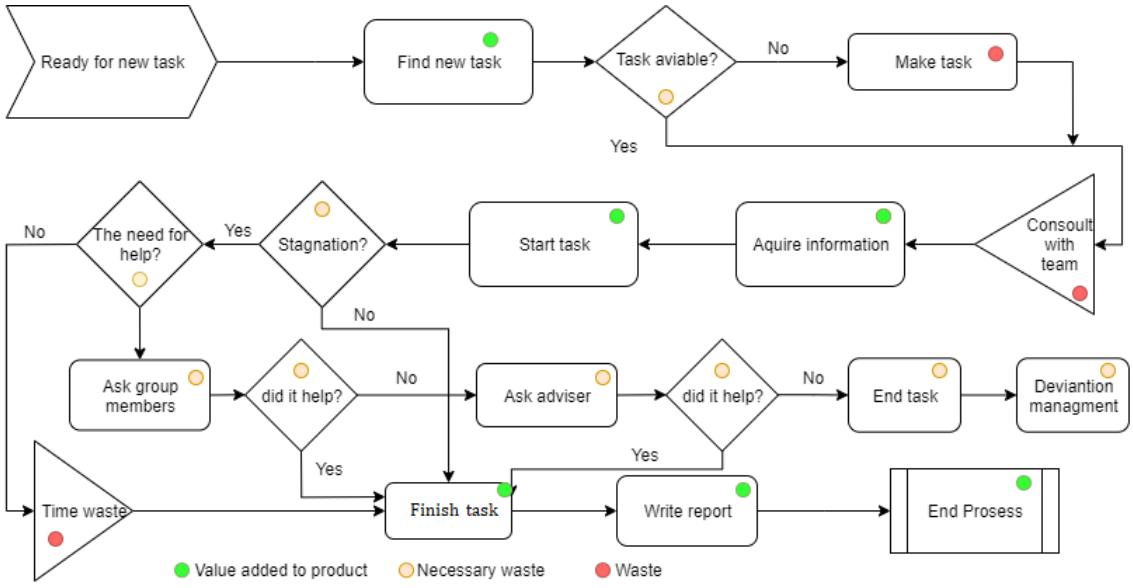
4. **Pull**

Kunden bestiller et produkt, og ikke før bestillingen har ankommet blir pull-produksjonen satt i gang f.eks. med prinsippet JIT.

5. Perfection

Ved å tenke på Kaizen under hele prosjektet, sett vi fokus på koninuerlig forbedringer undeveis. Gruppen har ukentlige møter for å diskutere oppgaver, problemer som kan oppstå og forbedringer.

Under ser vi et prosess kart av hvordan en typisk arbeidsprosess foregår i gruppen. Innholdet i kartet ble først ført opp på en slik måte gruppen utførte en vanlig arbeidsoppgave. Senere markerte vi hva som er viktig for produktet og hva som er sløsing "waste". I kartet kan vi se markeringene i rødt som direkte sløsinger, men vi har også gul markering som er indirekte sløsinger. Indirekte sløsinger er handlinger vi ikke kan eliminere, men bare forbedre. Den siste markeringen er grønn, denne er markert på oppgaver som legger til verdi i produktet som blir produsert.

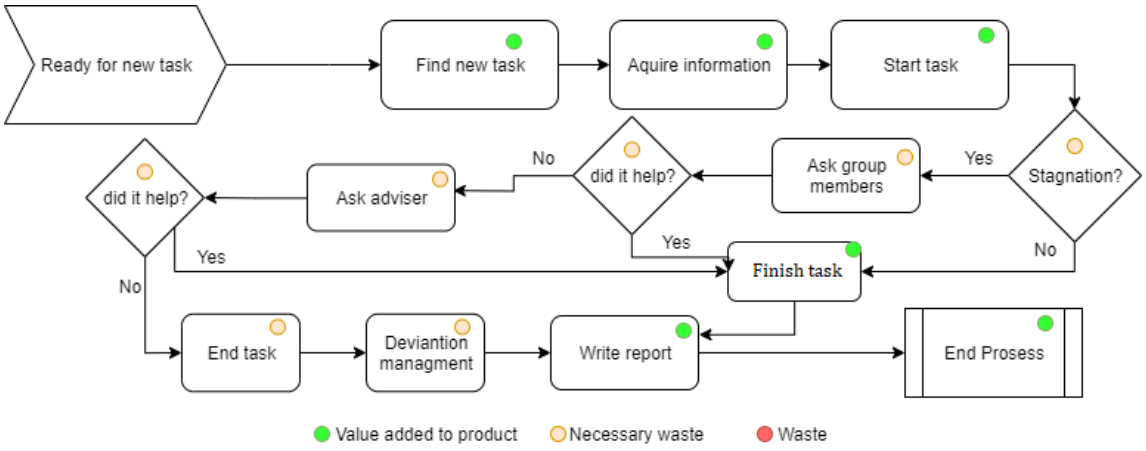


Figur 3 Prosesskart før Lean

Hensikten med Lean er å eliminere sløsinger i arbeidsprosesser. Vi eliminerer disse først ved å identifisere "waste", slik at vi har et mer konkret bilde av arbeidsprosessen. Når sløsing er identifiser kan vi iverksette endringer i de enkelte prosessene. I vårt tilfelle var dette å lage oppgaver til alle gruppemedlemmene og planlegge disse frem til prosjektslutt.

Endringene gruppen valgte å innføre var simple steg. Gruppen valgte å sette opp en grundig plan over arbeidsoppgaver, som var spesifikk tildelt en person som hadde ansvaret for den. Ved å innføre denne endringen slipper gruppemedlemmene å lete etter ny oppgave for hver gang en er ferdig med sin oppgave. På denne måten utjevnes arbeidsoppgavene, også kalt «heijunka». Det å tildele en person ansvar for en oppgave har også sine positive sider. Ved å ha ansvar for en oppgave slipper man å forhøre seg med gruppen når det kommer til å ta avgjørelser, som gjør det lettere. Den siste endringen er å kontakte hjelp med en gang stagnering av en arbeidsoppgave kommer. Stagnering skaper mye sløsing og det er mye bedre å skaffe seg hjelp for å komme seg videre.

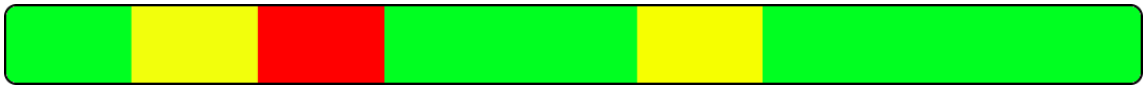
Under arbeid med Lean har gruppen brukt informasjonen som kommet frem i modul 1 i faget industri 4.0 til å utvikle et nytt prosesskart. Dette prosesskartet kan du se under, hvorav vi har fjernet tidssløsing som vi så på som unødvendig. Med simple endringer i arbeidsprosessene våre har vi redusert og effektivisert tids flyten i arbeidsoppgavene.



Figur 4 Prosesskart etter Lean

4 RESULTATER/ DRØFTING

Fra flytdiagrammene over har vi tatt ut hoved veiene og sett forskjellen i verdiskapning til produktet. Under kan du se dette litt klarere med effektbarene.



Figur 5 Effektbar før Lean

Figur 6 Effektbar etter Lean

Som vi ser i flytdiagrammene og effektbarene er at noe har endret seg. Prosessen er kortere og det er mindre steg som er angitt som sløsing eller nødvendig sløsing i hovedprosessen.

5 KONKLUSJON

Vi kan konkludere fra barene over at sløsingen i arbeidsprosessen har endret seg. Dette fordi stegene i prosessen er mindre og unødvendig sløsing er fjernet. Fokuset vårt i en bachelor oppgave ligger i å løse daglige oppgaver, som til slutt er satt sammen til et ferdig produkt. De daglige oppgavene er ofte veldig like, hvordan vi planlegger disse og hvordan oppgavene utføres. Dette gjør at gruppen anser dette som en viktig del av arbeidet vårt og er derfor vi har hatt søkelys på dette i denne rapporten.

6 REFERANSER

Ask Lean. u.d. https://asklean.com/5s-success-tips-apply-it-to-your-office-home-and-factory-too/.

NTNU. «Blackbord.» *Blackbord.* 21 01 2020. https://ntnu.blackboard.com/bbcswebdav/pid-881646-dt-content-rid-25285014_1/xid-25285014_1 (funnet 01 21, 2020).

—. *NTNU.* u.d. https://innsida.ntnu.no/wiki/-/wiki/Norsk/Laboratorie-+og+verkstedh%C3%A5ndbok (funnet 01 08, 2020).

B

NTNU
Kunnskap for en bedre verden

Risikorapport

IE303612 Bacheloroppgave angående Covid-19

Risikorapport

Andreas Skjong
Finn-Christian Eriksen
Torbjørn Flor

Totalt antall sider inkludert forsiden: xx

NTNU
Kunnskap for en bedre verden

Ålesund, 14.03.2020

INNHOLD

TERMINOLOGI

Begreper

Begrep	Beskrivelse
COVID-19	Coronavirus
Karantene	Ungå unødvendig kontakt med andre mennesker
Isolat	Ingen kontakt med andre mennesker
Epidemi	Økt hyppighet eller utbredelse av sykdom
Pandemi	Epidemi som brer seg over større landområder og flere tilfeller.

INNLEDNING

Under pandemi utbruddet Covid-19 viruset har skoler, barnehager og arbeidsplasser blitt stengt ned. Grunnet dette har gruppen fått uforutsette utfordringer som kan ha påvirkning på arbeidet i prosjektet. I denne rapporten skal gruppen risikovurdere viruset sitt utslag på arbeidet i bacheloromhandlingen.

1 STATUS

1.1 Hendelsesforløp

Allerede den 28. januar fikk gruppen problemer angående viruset. Det skulle bestilles inn hardware til prosjektet, men siden lageret befant seg i Kina var det uvist på når bestillingen kunne komme.

2 stk	ReSpeaker USB Mic Array	Elfa Distrelec	https://www.elfadistrelec.no/en/respeaker-usb-microphone-array-seeed-studio-107990193/p/30164996?q=respeaker&pos=6&origPos=6&origPageSize=10&track=true

27. februar ble diverse rettningslinjer fra det statlige helseorgan FHI ble innført av gruppen. Vasking og spriting av hender. (Folkehelseinstituttet(FHI) 2020)

12. mars fikk gruppen varsel fra NTNU, alle skoler blir sengt grunnet viruset fra 12. mars og til 27. mars, med mulighet for utvidelse. Dette påvirker gruppen muligheter for arbeidsplasser til oppgaven. I tillegg blir muligheten for bruk av LAB borte. Dette forbudet gjelder til og med 26. mars, og er stor sansynlighet for utvidelse. (NTNU 2020)

Regjeringen innfører en rekke nye forbud, en av dem er utenlandsreisende skal i karantene ved hjemkomst. Dette punktet treffer gruppen, da en av gruppemedlemmene var utenfor norden når forbudet tredde i kraft. (FHI 2020)

1.2 Nåværende status 14.03.2020

Gruppens nåværende status er å holde avstand og minske sosial kontakt med andre. Dette er da løst med å innføre hjemmekontor fra og med 13. mars – ubestemt tid. En av gruppens medlemmer er satt i karantene etter forbudet mot utlandsreiser. Karantenen gjelder F.o.m 13. mars til 27. mars og avholdes i et annet fylke. På grunn av restriksjoner mot reise kan splittelsen bli et problem i ubestemt tid.

1.3 Påførte tap

Gruppens tap av arbeidstid og effektivitet til nåværende tidspunkt er minimal. Gruppen anslår et tap i effektivitet ved hjemmearbeid på 10% per gruppemedlem over tid. I vurderingen ble muligheten for å jobbe sammen og ingen tilgang på enkelte hardware tatt i betrakning. Ved en dags arbeid den 13. mars fordelt på to gruppemedlemmer ligger et kalkulert tap på 1,4 timer.

2 RISIKOEVALUERING

Risikoanalyse

Frekvensen av tilfelle/r	Alvorlighetsgrad av konsekvens				
	Veldig lav	Lav	Middels	Høy	Veldig høy
Veldig høy					
Høy			Utvidet anbefalinger frå FHI (Hjemmekontor, ungå sosial kontakt)		
Middels	Utvidet restriskjoner frå Staten(kolektiv trasport)		Utvidet restriskjoner frå Staten(Ingen reiseing)		
Lav			Syk/fraværende rådgiver Beskymringer rundt viruset	Gruppemedlem blir syk av viruset	
Veldig lav					Død gruppemedlem

Prognose

Utvidet restriksjoner/retningslinjer fra FHI/NTNU - hjemmekontor			
Utvidet tid	Rammet arbeidsdager*	Tapsprosent	Total tap av tid**
12.03.20-27.03.20	11	10%	21,45 timer
12.03.20-30.04.20	35	10%	68,25 timer
12.03.20-20.05.20	47	10%	91,65 timer

*(6,5 timer) en arbeidsdag betegnes som de planlagte arbeidsdagene gruppen har valgt i forposjektet.

** Beregnet på 3 arbeidere

Syk gruppemedlem, isolasjon			
Antall medlemmer	Rammet arbeidsdager*	Tapsprosent**	Total tap av tid
1	14+	70-100%	64-91timer
2	28+	70-100%	128-182timer
3	42+	70-100%	192-273timer

*(6,5 timer) en arbeidsdag betegnes som de planlagte arbeidsdagene gruppen har valgt i forposjektet.

**Tapsprosent er beregnet etter muligheten for å jobbe hjemmefra.

3 DRØFTING

Gruppen velger å følge FHI sine retningslinjer sammens med statens og NTNU's restriksjoner. Dette blir avhold av gruppen så lenge restriksjonene/retninglinjene ikke oppheves. Gruppen velger dette på grunn av faren for smitte, en eventuell syk gruppemedlem kan påføre store tap av arbeidstid.

4 KONKLUSJON

Grunnet forholdene rundt epidemien, blir gruppen å ta i bruk tiltakene som hjemmekontor og mindre sosial kontakt. Splittelsen i prosjektgruppen vil være et problem som kan påføre prosjektet tap i senere tid, det er ubevist når gruppen kommer tilbake til normal drift. Hjemmekontor vil ha en effekt på arbeidseffektiviteten til gruppen. Gruppen ser på dette som et mindre tap og konkuderer derfor for å ikke søke om forlenget tid i bacheloren.

5 REFERANSER

FHI. *Smittevernsråd ved reise.* 12 03 2020. https://www.fhi.no/nettpub/coronavirus/fakta/reiserad-knyttet-til-nytt-koronavirus-coronavirus/ (funnet 03 14, 2020).

Folkehelseinstituttet(FHI). *fhi.no.* 14 03 2020. https://www.fhi.no/nettpub/coronavirus/fakta/fakta-om-koronavirus-coronavirus-2019-ncov/ (funnet 03 14, 2020).

Helsenorge. *Helsenorge: karantene og isolasjon corona.* 14 03 2020. https://helsenorge.no/koronavirus/karantene-og-isolasjon (funnet 03 14, 2020).

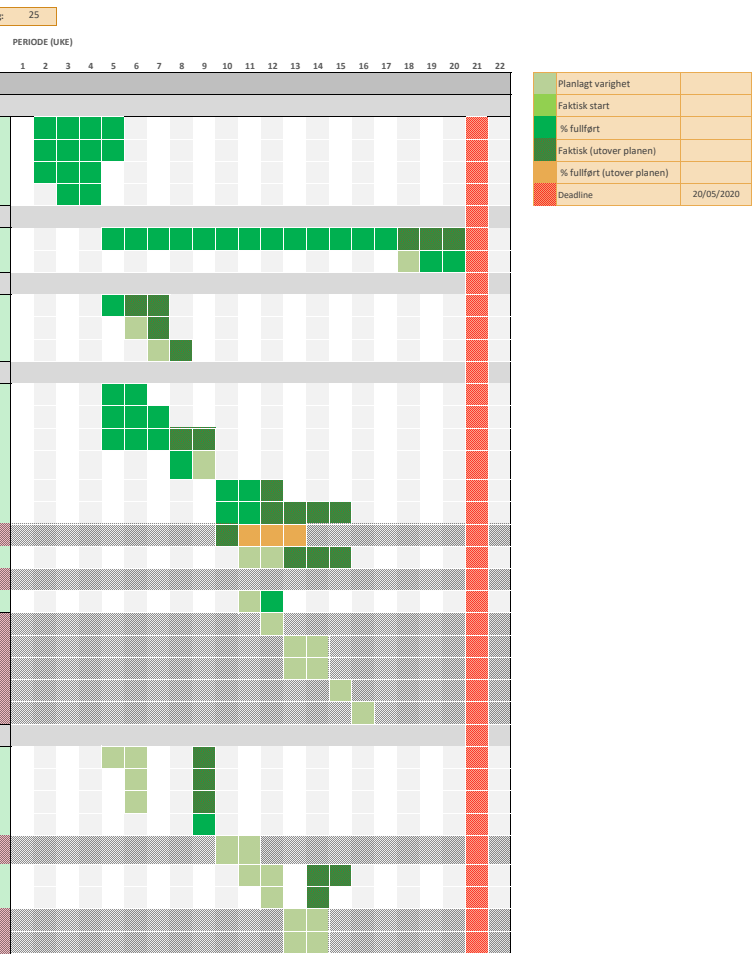
NTNU. *Retningslinjer for coronautbrudd.* 14 03 2020. https://www.ntnu.no/korona (funnet 03 14, 2020).

C

Prosjektplanlegging - 22.01.2020

Periodeutheving: 25

AKTIVITET	HOVED-ANSVAR	ANSVAR	PLANLAGT START (UKE)	PLANLAGT VARIGHET (UKER)	FAKTISK START (UKE)	FAKTISK VARIGHET (UKE)	PROSENT FULLFØRT
Forprosjekt							
Skriving av forprosjekt rapport	Alle	Alle	2	4	2	4	100%
Undersøke - innhenting av informasjon	Alle	Alle	2	4	2	4	100%
Planlegge oppgaver, tidsbruk og	Alle	Alle	2	3	2	3	100%
Implemetere LEAN i bacheloren	Alle	Alle	3	2	3	2	100%
Hovedprojekt							
Skrive bachelor	Alle	Alle	5	13	5	16	100%
Rettskrive bachelor	Alle	Alle	18	3	19	2	100%
Hardware							
Valg av hardware	Andreas	Alle	5	1	5	3	100%
Bestille inn hardware	Andreas		6	1	7	1	100%
Montere hardware	Finn		7	1	8	1	100%
Software							
Funksjon for å hente inn lydsignal fra mikrofon og grafisk fremstilling	Finn		5	2	5	2	100%
Finne ut om filtrering av støy/ekko	Torbjørn		5	3	5	3	100%
Finne ut om triangulering	Andreas		5	3	5	5	100%
Funksjon for å hente inn lydsignal fra mikrofon med korrekt hardware (realtime, lydfil, ekstern)	Finn		8	2	8	1	100%
Implementere filtrering av lydsignal, beamforming	Finn		10	2	10	3	100%
Implementere triangulering av en lydkilde	Andreas		10	2	10	6	100%
~~Implementere triangulering av flere lydkilder~~	Andreas		12	2	10	4	10%
Lag funksjon for å hente lyd fra lydkilder	Finn		11	2	13	3	100%
~~Implementere RIR i sanntid~~	Torbjørn						0%
Implementere filtrering av ekko med RIR	Torbjørn		11	2	12	1	100%
~~Implementer liste for lydkilder med funksjoner for å legge til og slette~~			12	1			0%
~~Lage til historikk på lydkilder~~			13	2			0%
~~Lag funksjon for å hente ut verdier til varmekart~~			13	2			0%
~~Funksjon "velg lydkilde" for å lett velge hvilken man skal lytte til~~			15	1			0%
~~Funksjon for automatisk opptak av lydkilder~~			16	1			0%
GUI							
Finne ut hvilket GUI bibliotek som skal tas i bruk	Finn		5	2	9	1	100%
Lag et skisse av GUI utseendet	Finn		6	1	9	1	100%
Implementere GUI I prosjektet og opprett en "Hello World"	Finn		6	1	9	1	100%
Implementer overblikksbilde	Finn		9	1	9	1	100%
~~Implementere måter for å "plassere" lydkilder på overblikksbilde samt legge dei til i lydkilde listen~~			10	2			0%
Implementere måter for å spille av lydkilder			11	2	14	2	100%
Opprett liste av lydkilder			12	1	14	1	100%
~~Implementere varmekart~~			13	2			0%
~~Lage til trend for historikk på lydkilder~~			13	2			0%



AKTIVITET	HOVED-ANSVAR	ANSVAR	PLANLAGT START (UKE)	PLANLAGT VARIGHET (UKER)	FAKTISK START (UKE)	FAKTISK VARIGHET (UKE)	PROSENT FULLFØRT	PERIODE (UKE) 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22
Implementer funksjon for "automatisk" opptak av lydkilder			16	1	16	1	**100%**	
Opprette GUI I MATLAB	Andreas		13	2	13	5	**100%**	
Testing								
Testing henting av lyd			6	1	6	1	**100%**	
Testing triangulering			8	1	9	2	**100%**	
Fullskala test	Alle		17	1	17	1	**100%**	

D

	Position	Total: Paid hours					
Total: Paid hours		1819.65					
Day notes							
Andreas Sønderland Skjong	Bachelor 2020	616.1666667					
Finn-Christian Eriksen	Bachelor 2020	599.4833333					
Torbjørn Inge Flor	Bachelor 2020	604					
	Mon 6 Jan	Tue 7 Jan	Wed 8 Jan	Thu 9 Jan	Fri 10 Jan	Sat 11 Jan	Sun 12 Jan
Andreas Sønderland Skjong	8:15-16:00	11:15-16:00	8:00-16:00	8:00-16:00	Unpaid		
Andreas Sønderland Skjong							
Finn-Christian Eriksen	10:15-16:00	11:15-16:00	8:00-16:00	9:00-16:00	8:00-15:00		
Finn-Christian Eriksen							
Torbjørn Inge Flor		11:15-16:00	8:00-16:00	9:00-16:00	7:15-16:00		
Torbjørn Inge Flor							
	Mon 13 Jan	Tue 14 Jan	Wed 15 Jan	Thu 16 Jan	Fri 17 Jan	Sat 18 Jan	Sun 19 Jan
Andreas Sønderland Skjong	Unpaid		Unpaid	Unpaid	8:15-16:00		
Andreas Sønderland Skjong							
Finn-Christian Eriksen	Unpaid	9:15-18:00	Unpaid	Unpaid	9:00-16:00		
Finn-Christian Eriksen							
Torbjørn Inge Flor	Unpaid	7:15-16:00	Unpaid	Unpaid	9:15-15:00		
	Mon 20 Jan	Tue 21 Jan	Wed 22 Jan	Thu 23 Jan	Fri 24 Jan	Sat 25 Jan	Sun 26 Jan
Andreas Sønderland Skjong	9:00-16:00	9:00-15:00	9:00-13:00				
Andreas Sønderland Skjong							
Finn-Christian Eriksen	9:00-12:00	9:00-15:00	9:00-13:00				
Finn-Christian Eriksen							
Torbjørn Inge Flor	Parental Leave	9:00-15:00	9:00-13:00				
	Mon 27 Jan	Tue 28 Jan	Wed 29 Jan	Thu 30 Jan	Fri 31 Jan	Sat 1 Feb	Sun 2 Feb
Andreas Sønderland Skjong	Unpaid	Unpaid	Unpaid	9:00-12:30	10:00-11:15		
Andreas Sønderland Skjong							
Finn-Christian Eriksen	Unpaid	Unpaid	Unpaid	9:00-15:00	10:00-11:15		
Finn-Christian Eriksen							
Torbjørn Inge Flor	Unpaid	Unpaid	Unpaid	9:00-15:00	10:00-11:15	Holiday	

	Mon 3 Feb	Tue 4 Feb	Wed 5 Feb	Thu 6 Feb	Fri 7 Feb	Sat 8 Feb	Sun 9 Feb
Andreas Sønderland Skjong	8:15-16:00	8:15-16:00	9:15-15:00				
Andreas Sønderland Skjong							
Finn-Christian Eriksen	8:15-16:00	8:15-16:00	9:15-15:00			19:00-20:35	19:00-20:30
Finn-Christian Eriksen	21:00-22:30	19:00-20:00	17:00-18:35				
Torbjørn Inge Flor	Parental Leave	9:00-15:00	Holiday	9:00-15:00	Holiday	Holiday	
	Mon 10 Feb	Tue 11 Feb	Wed 12 Feb	Thu 13 Feb	Fri 14 Feb	Sat 15 Feb	Sun 16 Feb
Andreas Sønderland Skjong	9:00-14:30	9:00-15:00	Unpaid	Unpaid	Unpaid		
Andreas Sønderland Skjong				10:00-11:00			
Finn-Christian Eriksen	10:00-16:00	9:00-16:00	Unpaid	Unpaid	Unpaid		
Finn-Christian Eriksen							
Torbjørn Inge Flor	9:00-16:45	9:00-16:00	Unpaid	Unpaid	Unpaid		
	Mon 17 Feb	Tue 18 Feb	Wed 19 Feb	Thu 20 Feb	Fri 21 Feb	Sat 22 Feb	Sun 23 Feb
Andreas Sønderland Skjong	9:15-15:00	9:15-15:00	9:15-15:00	9:15-15:30	9:15-15:00		
Andreas Sønderland Skjong							
Finn-Christian Eriksen	9:00-15:00	9:54-15:00	9:15-16:00	8:50-16:00	Holiday		
Finn-Christian Eriksen							
Torbjørn Inge Flor	9:00-15:15	9:00-15:15	9:00-16:00	9:00-16:00	9:00-14:45		
	Mon 24 Feb	Tue 25 Feb	Wed 26 Feb	Thu 27 Feb	Fri 28 Feb	Sat 29 Feb	Sun 1 Mar
Andreas Sønderland Skjong	9:15-16:00	9:15-16:15	9:15-15:00	9:15-15:00	9:15-16:00		
Andreas Sønderland Skjong							
Finn-Christian Eriksen	8:50-16:00	8:50-16:00	8:50-16:00	8:50-16:00	8:50-16:00		
Finn-Christian Eriksen							
Torbjørn Inge Flor	9:15-16:00	9:15-16:00	9:15-16:00	9:15-16:00	Unpaid		
	Mon 2 Mar	Tue 3 Mar	Wed 4 Mar	Thu 5 Mar	Fri 6 Mar	Sat 7 Mar	Sun 8 Mar
Andreas Sønderland Skjong	9:00-16:00	9:00-16:00	Unpaid	Unpaid	Unpaid		
Andreas Sønderland Skjong							
Finn-Christian Eriksen	8:50-16:00	9:00-16:00	Unpaid	Unpaid	Unpaid		
Finn-Christian Eriksen	19:00-20:00						
Torbjørn Inge Flor	9:15-15:00	Parental Leave	Unpaid	Unpaid	Unpaid		

	Mon 9 Mar	Tue 10 Mar	Wed 11 Mar	Thu 12 Mar	Fri 13 Mar	Sat 14 Mar	Sun 15 Mar
Andreas Sønderland Skjong	9:15-16:00	9:15-16:00	9:15-16:00	9:15-13:00	9:15-16:00		
Andreas Sønderland Skjong							
Finn-Christian Eriksen	9:00-16:00	9:00-16:00	8:30-16:00	Holiday	Holiday	12:30-16:30	
Finn-Christian Eriksen	19:00-22:00						
Torbjørn Inge Flor	9:15-16:00	9:15-16:00	9:45-16:00	9:15-13:00	9:15-16:00		
	Mon 16 Mar	Tue 17 Mar	Wed 18 Mar	Thu 19 Mar	Fri 20 Mar	Sat 21 Mar	Sun 22 Mar
Andreas Sønderland Skjong	9:30-16:00	9:15-15:00	9:15-15:00	8:30-16:00	8:30-16:00		
Andreas Sønderland Skjong							
Finn-Christian Eriksen	9:00-14:00	8:00-15:00	9:00-16:00	9:15-15:30	8:45-15:00		
Finn-Christian Eriksen	16:00-18:00						
Torbjørn Inge Flor	9:15-16:00	9:15-16:00	9:30-20:15	9:30-16:15	9:15-20:15	12:00-13:45	
	Mon 23 Mar	Tue 24 Mar	Wed 25 Mar	Thu 26 Mar	Fri 27 Mar	Sat 28 Mar	Sun 29 Mar
Andreas Sønderland Skjong	9:00-18:30	8:30-20:15	8:30-19:30	8:30-20:45	11:30-16:30		
Andreas Sønderland Skjong							
Finn-Christian Eriksen	10:00-14:00	8:30-16:30	9:00-16:00	13:30-20:30	8:30-16:00		
Finn-Christian Eriksen	14:00-20:00	21:00-22:30					
Torbjørn Inge Flor	9:15-16:00	9:15-0:00	11:00-17:45	9:15-16:00	9:15-22:00		
Torbjørn Inge Flor	21:45-23:00						
	Mon 30 Mar	Tue 31 Mar	Wed 1 Apr	Thu 2 Apr	Fri 3 Apr	Sat 4 Apr	Sun 5 Apr
Andreas Sønderland Skjong	9:30-20:30	8:15-19:45	9:15-18:30	12:15-0:30	12:15-7:30	16:00-19:30	
Andreas Sønderland Skjong							
Finn-Christian Eriksen	17:00-19:00	8:30-16:00	9:15-16:15	8:00-15:00	8:00-15:30		
Finn-Christian Eriksen							
Torbjørn Inge Flor	8:15-16:00	12:30-13:15	10:30-16:15	9:15-16:45	10:00-17:30		
	Mon 6 Apr	Tue 7 Apr	Wed 8 Apr	Thu 9 Apr	Fri 10 Apr	Sat 11 Apr	Sun 12 Apr
Andreas Sønderland Skjong	8:30-16:00	20:45-4:00	20:45-4:00				
Andreas Sønderland Skjong							
Finn-Christian Eriksen	8:30-15:30	9:00-16:15	9:00-16:00		18:30-21:30		
Finn-Christian Eriksen							

Torbjørn Inge Flor	Holiday	9:15-16:15	Unpaid	Holiday	Holiday		14:15-16:30
Torbjørn Inge Flor	14:15-21:30 Sett nøye gjennom den itererte og filtrerte lyden, og konkluderer med at mobilhøyttaler ikke er tilstrekkelig nok til å skape ekko som kan fjernes ved hjelp av inverskonvulering med RIR. Ba andreas om å ta opp nytt lydopptak med klapping for å få mer ekko og reverb.	9:15-22:30 Itererte over kanaler for å lage separate wav-filer. Andreas hjalp meg å lage kode som slo sammen kanalene i en stor wav fil.	9:15-12:00 Fikk PC tilbake fra Service. Så døde den igjen.		9:15-22:30 Itererte over kanaler for å lage separate wav-filer. Andreas hjalp meg å lage kode som slo sammen kanalene i en stor wav fil.		
	Mon 13 Apr	Tue 14 Apr	Wed 15 Apr	Thu 16 Apr	Fri 17 Apr	Sat 18 Apr	Sun 19 Apr
Andreas Sønderland Skjong							
Andreas Sønderland Skjong							
Finn-Christian Eriksen		9:00-16:23	9:00-19:00	8:00-15:00	9:00-16:00	10:00-16:10	
Finn-Christian Eriksen			21:00-23:00		18:00-22:00		
Torbjørn Inge Flor		9:15-18:30	9:15-18:30		9:15-18:30	9:15-18:30	
	Mon 20 Apr	Tue 21 Apr	Wed 22 Apr	Thu 23 Apr	Fri 24 Apr	Sat 25 Apr	Sun 26 Apr
Andreas Sønderland Skjong	10:00-17:00	8:00-14:10	10:00-17:00	10:00-17:00	10:00-17:00		
Andreas Sønderland Skjong							
Finn-Christian Eriksen	10:00-17:00	8:00-14:10	17:00-0:00	9:00-15:45	8:30-14:45		20:00-0:00
Finn-Christian Eriksen				18:00-23:00			
Torbjørn Inge Flor	9:00-15:45	9:15-16:00	9:15-16:00	9:15-16:00	9:15-16:00		

	Mon 27 Apr	Tue 28 Apr	Wed 29 Apr	Thu 30 Apr	Fri 1 May	Sat 2 May	Sun 3 May
Andreas Sønderland Skjong	10:00-20:00	10:00-20:00	10:00-18:45	10:00-20:00	10:00-20:00	10:00-20:00	12:00-22:00
Andreas Sønderland Skjong							
Finn-Christian Eriksen	10:00-16:40	10:00-14:15	11:00-16:40	10:00-20:00			10:00-11:00
Finn-Christian Eriksen		18:00-23:40	19:00-22:00				
Torbjørn Inge Flor	9:15-16:00	14:30-22:00	9:00-16:30	9:00-16:30	18:15-1:15	10:45-22:30	9:45-20:15
	Mon 4 May	Tue 5 May	Wed 6 May	Thu 7 May	Fri 8 May	Sat 9 May	Sun 10 May
Andreas Sønderland Skjong	10:00-22:30	10:00-20:45	10:00-22:30	10:00-23:45	10:00-22:30	12:00-17:15	10:00-22:30
Andreas Sønderland Skjong							
Finn-Christian Eriksen	10:00-22:30	9:00-19:10	10:15-19:00	13:00-20:00	10:15-16:00	12:00-16:00	16:00-19:00
Finn-Christian Eriksen					18:00-23:30		
Torbjørn Inge Flor	9:45-22:15	8:00-11:00	9:30-21:30	9:00-20:15	9:00-23:00	14:45-18:30	16:45-23:30
	Mon 11 May	Tue 12 May	Wed 13 May	Thu 14 May	Fri 15 May	Sat 16 May	Sun 17 May
Andreas Sønderland Skjong	10:30-22:30	11:00-23:00	11:00-0:00	15:45-22:00	12:00-20:00	12:00-0:00	21:00-0:00
Andreas Sønderland Skjong							
Finn-Christian Eriksen	9:00-17:30	9:30-20:00	17:30-1:45	10:30-23:30	12:00-20:00	10:00-14:00	21:00-0:00
Finn-Christian Eriksen	20:00-23:00					20:00-1:25	
Torbjørn Inge Flor	10:45-17:30	9:00-23:30	9:00-23:30	9:00-20:15	12:00-20:00	21:00-0:00	
	Mon 18 May	Tue 19 May					
Andreas Sønderland Skjong	11:00-2:00	11:00-23:45					
Andreas Sønderland Skjong							
Finn-Christian Eriksen	11:00-2:00	11:00-23:45					
Finn-Christian Eriksen							
Torbjørn Inge Flor	10:00-23:00	11:00-23:45					

E

```
function BSS2S(audiosignal, angle, uca)
% This code will take in a audiosignal, angle and a UCA microphone.

% Beamforming
% UCA variables
nMicrophones  = uca.M;      %Number of microphones
fs = uca.fs;     %Samplerate
Tc = 20;
c = 331*sqrt(1+(Tc/273));   % Speed of sound


% Calculating the distance of the sound sources for each microphone
sourcePos = [angle(:,1).*cosd(angle(:,2)) angle(:,1).*sind(angle(:,2))];
source1LengthShift = vecnorm((sourcePos(1,:)-uca.getPos)',2,1);  % Location 1
source2LengthShift = vecnorm((sourcePos(2,:)-uca.getPos)',2,1);  % Location 2


% Converting the distance from the sound source to the microphones to number of
% samples from the sound source to the microphones
source1SampleShift = round(source1LengthShift./c.*fs);
source2SampleShift = round(source2LengthShift./c.*fs);


% Subtracting the minumum samples between the microphones and the sound source and
% inverting the signal so the microphone that is the furtherst away is the
% closest.
p1 = max(source1SampleShift) + (max(source1SampleShift)-source1SampleShift);    %↙
To invert the signal
p2 = max(source2SampleShift) + (max(source2SampleShift)-source2SampleShift);    %↙
To invert the signal


% Aligning audio channels
delayed_channels1 = delayseq(audiosignal,p1);
delayed_channels2 = delayseq(audiosignal,p2);


% Converting from 6 channels to 1 channel by summing the aligned signal
sum1 = sum(delayed_channels1,2)./nMicrophones;
sum2 = sum(delayed_channels2,2)./nMicrophones;


% Converting the audio signal to frequency domain by performing FFT
Sk1 = fft(sum1);
Sk2 = fft(sum2);


% Creating empty arrays to prevent the array size changing each loop.
% This is only to make the code perform faster.
Yk1 = zeros(size(Sk1));
Yk2 = zeros(size(Sk2));


% Blind-Source-Separation
for i=1:length(Sk1)
    if abs(Sk1(i)) > abs(Sk2(i))
        Yk1(i) = Sk1(i);
    elseif abs(Sk2(i)) > abs(Sk1(i))
        Yk2(i) = Sk2(i);
    end
```

```
end


% Converting the audio signal back to time domain by performing IFFT
yn1 = ifft(Yk1);
yn2 = ifft(Yk2);


% Saving the beamformed arrays to saparated wav files
audiowrite('source1.wav',yn1,fs);
audiowrite('source2.wav',yn2,fs);


figure()
   t = 0:1/fs:1-0.001;
   x = yn1;
   xdft = fftshift(fft(x));
   df = fs/length(x);
   freq = -fs/2:df:fs/2-df;
   plot(freq,abs(xdft))
xlim([0 2000])

figure()
   t = 0:1/fs:1-0.001;
   x = yn2;
   xdft = fftshift(fft(x));
   df = fs/length(x);
   freq = -fs/2:df:fs/2-df;
   plot(freq,abs(xdft))
xlim([0 2000])
end
```

```
function BSS3S(audiosignal, angle, uca)
% This code will take in a audiosignal, angle and a UCA microphone.


% UCA variables
nMicrophones  = uca.M;      % Number of microphones in the UCA
fs = uca.fs;                % Samplerate of the UCA
Tc = 20;                    % Temperature of the room
c = 331*sqrt(1+(Tc/273));   % Speed of sound


% Calculating the distance of the sound sources for each microphone
sourcePos = [angle(:,1).*cosd(angle(:,2)) angle(:,1).*sind(angle(:,2))];
source1LengthShift = vecnorm((sourcePos(1,:)-uca.getPos)',2,1);  % Location 1
source2LengthShift = vecnorm((sourcePos(2,:)-uca.getPos)',2,1);  % Location 2
source3LengthShift = vecnorm((sourcePos(3,:)-uca.getPos)',2,1);  % Location 3


% Converting the distance from the sound source to the microphones to number of
% samples from the sound source to the microphones
source1SampleShift = round(source1LengthShift./c.*fs);
source2SampleShift = round(source2LengthShift./c.*fs);
source3SampleShift = round(source3LengthShift./c.*fs);


% Subtracting the minumum samples between the microphones and the sound source and
% inverting the signal so the microphone that is the furtherst away is the
% closest.
p1 = max(source1SampleShift) + (max(source1SampleShift)-source1SampleShift);    %↙
To invert the signal
p2 = max(source2SampleShift) + (max(source2SampleShift)-source2SampleShift);    %↙
To invert the signal
p3 = max(source3SampleShift) + (max(source3SampleShift)-source3SampleShift);    %↙
To invert the signal


% Aligning audio channels
delayedChannel1 = delayseq(audiosignal,p1);
delayedChannel2 = delayseq(audiosignal,p2);
delayedChannel3 = delayseq(audiosignal,p3);


% Converting from 6 channels to 1 channel by summing the aligned signal
sum1 = sum(delayedChannel1,2)./nMicrophones;
sum2 = sum(delayedChannel2,2)./nMicrophones;
sum3 = sum(delayedChannel3,2)./nMicrophones;


% Converting the audio signal to frequency domain by performing FFT
Sk1 = fft(sum1);
Sk2 = fft(sum2);
Sk3 = fft(sum3);


% Creating empty arrays to prevent the array size changing each loop.
% This is only to make the code perform faster.
Yk1 = zeros(size(Sk1));
Yk2 = zeros(size(Sk2));
Yk3 = zeros(size(Sk3));

```

```
58
59 % Blind-Source-Separation
60 for i=1:length(Sk1)
61     if (abs(Sk1(i)) > abs(Sk2(i)))&&(abs(Sk1(i)) > abs(Sk3(i)))
62         Yk1(i) = Sk1(i);
63     elseif abs(Sk2(i)) > abs(Sk1(i))&&(abs(Sk2(i)) > abs(Sk3(i)))
64         Yk2(i) = Sk2(i);
65     elseif abs(Sk3(i)) > abs(Sk1(i))&&(abs(Sk3(i)) > abs(Sk2(i)))
66         Yk3(i)= Sk3(i);
67     end
68 end
69
70
71 % Converting the audio signal back to time domain by performing IFFT
72 yn1 = ifft(Yk1);
73 yn2 = ifft(Yk2);
74 yn3 = ifft(Yk3);
75
76
77 % Saving the beamformed arrays to saparated wav files
78 audiowrite('source1.wav',yn1,fs);
79 audiowrite('source2.wav',yn2,fs);
80 audiowrite('source3.wav',yn3,fs);
81 end
```

```
function chunkBSS2S(audiosignal, angle, uca)
% This code will take in a audiosignal, angle and a UCA microphone.

% UCA variables
nMicrophones  = uca.M;      %Number of microphones
fs = uca.fs;     %Samplerate
Tc = 20;
c = 331*sqrt(1+(Tc/273));   % Speed of sound

CHUNK = 96000;
samples = length(audiosignal);


% Calculating the distance of the sound sources for each microphone
sourcePos = [angle(:,1).*cosd(angle(:,2)) angle(:,1).*sind(angle(:,2))];
source1LengthShift = vecnorm((sourcePos(1,:)-uca.getPos)',2,1);  % Location 1
source2LengthShift = vecnorm((sourcePos(2,:)-uca.getPos)',2,1);  % Location 2


% Converting the distance from the sound source to the microphones to number of
% samples from the sound source to the microphones
source1SampleShift = round(source1LengthShift./c.*fs);
source2SampleShift = round(source2LengthShift./c.*fs);


% Subtracting the minumum samples between the microphones and the sound source and
% inverting the signal so the microphone that is the furtherst away is the
% closest.
p1 = max(source1SampleShift) + (max(source1SampleShift)-source1SampleShift);    %↙
To invert the signal
p2 = max(source2SampleShift) + (max(source2SampleShift)-source2SampleShift);    %↙
To invert the signal


% Aligning audio channels
delayed_channels1 = delayseq(audiosignal,p1);
delayed_channels2 = delayseq(audiosignal,p2);


% Converting from 6 channels to 1 channel by summing the aligned signal
sum1 = sum(delayed_channels1,2)./nMicrophones;
sum2 = sum(delayed_channels2,2)./nMicrophones;

iterations = samples*2/CHUNK-2
windowHann = hann(CHUNK);


% Creating empty arrays to prevent the array size changing each loop.
% This is only to make the code perform faster.
yn1 = [];
yn2 = [];
halfChunk = CHUNK/2;


% Blind-Source-Separation
for i = 0:iterations
    lowLimit  = i*halfChunk+1;
    highLimit = i*halfChunk+CHUNK;

    Sk1 = fft(sum1(lowLimit:highLimit,1).*windowHann);
```

```
    Sk2 = fft(sum2(lowLimit:highLimit,1).*windowHann);

    Yk1 = zeros(CHUNK,1);
    Yk2 = zeros(CHUNK,1);

    for j=1:CHUNK
        if abs(Sk1(j)) > abs(Sk2(j))
            Yk1(j) = Sk1(j);
        else
            Yk2(j) = Sk2(j);
        end
    end

    % Converting the audio signal back to time domain by performing IFFT
    Yk1 = ifft(Yk1);
    Yk2 = ifft(Yk2);

    % Overlapping 50% of the signal
    if(i>0)
        oldData1   = yn1(1:i*halfChunk);
        summedata1 = yn1(i*halfChunk+1:i*halfChunk+halfChunk) + Yk1(1:halfChunk);
        futureata1 = Yk1(halfChunk+1:CHUNK);

        yn1 = [oldData1; summedata1; futureata1];

        oldData2   = yn2(1:i*halfChunk);
        summedata2 = yn2(i*halfChunk+1:i*halfChunk+halfChunk) + Yk2(1:halfChunk);
        futureata2 = Yk2(halfChunk+1:CHUNK);

        yn2 = [oldData2; summedata2; futureata2];
    else
        yn1 = Yk1;
        yn2 = Yk2;
    end
end


% Saving the beamformed arrays to saparated wav files
audiowrite('source1.wav',yn1,fs);
audiowrite('source2.wav',yn2,fs);
audiowrite('source3.wav',zeros(size(yn2)),fs);
end
```

```
function chunkBSS3S(audiosignal, angle, uca)
% This code will take in a audiosignal, angle and a UCA microphone.

% UCA variables
nMicrophones  = uca.M;      %Number of microphones
fs = uca.fs;     %Samplerate
Tc = 20;
c = 331*sqrt(1+(Tc/273));   % Speed of sound

CHUNK = 96000;
samples = length(audiosignal);

% Calculating the distance of the sound sources for each microphone
sourcePos = [angle(:,1).*cosd(angle(:,2)) angle(:,1).*sind(angle(:,2))];
source1LengthShift = vecnorm((sourcePos(1,:)-uca.getPos)',2,1);  % Location 1
source2LengthShift = vecnorm((sourcePos(2,:)-uca.getPos)',2,1);  % Location 2
source3LengthShift = vecnorm((sourcePos(3,:)-uca.getPos)',2,1);  % Location 3


% Converting the distance from the sound source to the microphones to number of
% samples from the sound source to the microphones
source1SampleShift = round(source1LengthShift./c.*fs);
source2SampleShift = round(source2LengthShift./c.*fs);
source3SampleShift = round(source3LengthShift./c.*fs);


% Subtracting the minumum samples between the microphones and the sound source and
% inverting the signal so the microphone that is the furtherst away is the
% closest.
p1 = max(source1SampleShift) + (max(source1SampleShift)-source1SampleShift);    %↙
To invert the signal
p2 = max(source2SampleShift) + (max(source2SampleShift)-source2SampleShift);    %↙
To invert the signal
p3 = max(source3SampleShift) + (max(source3SampleShift)-source3SampleShift);    %↙
To invert the signal


% Aligning audio channels
delayed_channels1 = delayseq(audiosignal,p1);
delayed_channels2 = delayseq(audiosignal,p2);
delayed_channels3 = delayseq(audiosignal,p3);


% Converting from 6 channels to 1 channel by summing the aligned signal
sum1 = sum(delayed_channels1,2)./nMicrophones;
sum2 = sum(delayed_channels2,2)./nMicrophones;
sum3 = sum(delayed_channels3,2)./nMicrophones;

iterations = samples*2/CHUNK-2;
windowHann = hann(CHUNK);


% Creating empty arrays to prevent the array size changing each loop.
% This is only to make the code perform faster.
yn1 = [];
yn2 = [];
yn3 = [];
halfChunk = CHUNK/2;


```

```
% Blind-Source-Separation
for i = 0:iterations
    lowLimit  = i*halfChunk+1;
    highLimit = i*halfChunk+CHUNK;

    Sk1 = fft(sum1(lowLimit:highLimit,1).*windowHann);
    Sk2 = fft(sum2(lowLimit:highLimit,1).*windowHann);
    Sk3 = fft(sum3(lowLimit:highLimit,1).*windowHann);

    Yk1 = zeros(CHUNK,1);
    Yk2 = zeros(CHUNK,1);
    Yk3 = zeros(CHUNK,1);

    for j=1:CHUNK
        if (abs(Sk1(j)) > abs(Sk2(j)))&&(abs(Sk1(j)) > abs(Sk3(j)))
            Yk1(j) = Sk1(j);
        elseif abs(Sk2(j)) > abs(Sk1(j))&&(abs(Sk2(j)) > abs(Sk3(j)))
            Yk2(j) = Sk2(j);
        elseif abs(Sk3(j)) > abs(Sk1(j))&&(abs(Sk3(j)) > abs(Sk2(j)))
            Yk3(j)= Sk3(j);
        end
    end

    % Converting the audio signal back to time domain by performing IFFT
    Yk1 = ifft(Yk1);
    Yk2 = ifft(Yk2);
    Yk3 = ifft(Yk3);

    % Overlapping 50% of the signal
    if(i>0)
        oldData1   = yn1(1:i*halfChunk);
        summedata1 = yn1(i*halfChunk+1:i*halfChunk+halfChunk) + Yk1(1:halfChunk);
        futureata1 = Yk1(halfChunk+1:CHUNK);

        yn1 = [oldData1; summedata1; futureata1];


        oldData2   = yn2(1:i*halfChunk);
        summedata2 = yn2(i*halfChunk+1:i*halfChunk+halfChunk) + Yk2(1:halfChunk);
        futureata2 = Yk2(halfChunk+1:CHUNK);

        yn2 = [oldData2; summedata2; futureata2];


        oldData3   = yn3(1:i*halfChunk);
        summedata3 = yn3(i*halfChunk+1:i*halfChunk+halfChunk) + Yk3(1:halfChunk);
        futureata3 = Yk3(halfChunk+1:CHUNK);

        yn3 = [oldData3; summedata3; futureata3];
    else
        yn1 = Yk1;
        yn2 = Yk2;
        yn3 = Yk3;
    end
end


% Saving the beamformed arrays to saparated wav files
audiowrite('source1.wav',yn1,fs);
audiowrite('source2.wav',yn2,fs);
```

```
audiowrite('source3.wav',yn3,fs);
end
```

```
function [magnitude1, magnitude2] = DOAmultisource(audiosignal, uca)
% global fig1 fig2 fig3 fig4
Tc = 20;                      % Temperature
c = 331*sqrt(1+(Tc/273));     % Speed of sound
fs = uca.fs;                  % Samplerate


magnitude1 = [];
magnitude2 = [];
frequencyStorage = [];


for x =0:5:360
    source = [1.5 x-180]; %R [length, angle]
    source = [source(1,1)*cosd(source(1,2)) source(1,1)*sind(source(1,2))];
    l = vecnorm((source(1,:)-uca.getPos)',2,1);  % Location 1

    dt = l./c;
    p_brut = round(dt.*fs);
    p_net = abs(p_brut-max(p_brut));

    delayed_channels = delayseq(audiosignal,p_net);

    sum1 = sum(delayed_channels,2);
    Sk = abs(fft(sum1)./fs);
    [mPK, freq] = findpeaks(Sk,'MinPeakDistance',500,'SortStr','descend');
    frequencyStorage = [frequencyStorage; freq(1) freq(2);];
    magnitude1 = [magnitude1 mPK(1)];
    magnitude2 = [magnitude2 mPK(2)];
end
end
```

```python
# Importing library's
import pyaudio
import wave
import numpy as np
from Recording.indexDevices import IndexDevices
from datetime import datetime

p = pyaudio.PyAudio()
uca = IndexDevices()
deviceIndex = uca.getIndex()
rate = uca.getSampleRate()
nChannels = uca.getChannels()

WIDTH = 2
FORMAT = pyaudio.paInt16
CHANNELS_TO_RECORD = 6


def record():
    now = datetime.now()
    current_time = now.strftime("D%Y.%m.%d-T%H.%M.%S")
    waveFileName = "/home/respeaker/Share/" + current_time + ".wav"
    secondsToRecord = 5

    stream = p.open(
        rate=rate,
        format=FORMAT,
        channels=nChannels,
        input=True,
        input_device_index=deviceIndex, )

    print("* recording")

    rawChannelFrames = []
    for i in range(0, secondsToRecord):
        data = stream.read(rate, exception_on_overflow=False)
        bufferedData = np.frombuffer(data, dtype=np.int16)  # interleaved channels
        separatedFrames = np.stack((bufferedData[0::nChannels], bufferedData[1::nChannels
],
                                    bufferedData[2::nChannels], bufferedData[3::nChannels
],
                                    bufferedData[4::nChannels], bufferedData[5::nChannels
]),
                                   axis=1)  # channels on separate axes
        rawChannelFrames.append(separatedFrames)

    print("* done recording")

    stream.stop_stream()
    stream.close()
    p.terminate()

    wf = wave.open(waveFileName, 'wb')
    wf.setnchannels(CHANNELS_TO_RECORD)
    wf.setsampwidth(p.get_sample_size(FORMAT))
    wf.setframerate(rate)
    wf.writeframes(b''.join(rawChannelFrames))
    wf.close()


if __name__ == '__main__':
    record()

```

```
classdef UCA
    %UCA Summary of this class goes here
    %   Detailed explanation goes here

    properties
        diameter    % Diameter of the UCA
        radius      % Radius of the UCA
        M           % Number of microphones
        fs          % Sample rate
        pos         % Positions of the mics
    end

    methods
        % Creating the UCA object
        function obj = UCA(diameter,M,fs)
            %UCA Construct an instance of this class
            obj.diameter = diameter;
            obj.radius = diameter / 2;
            obj.M = M;
            obj.fs = fs;
        end

        % Get the positions of the microphones
        function pos = getPos(obj)
            for i=0:obj.M-1
                obj.pos(i+1,:) = [obj.radius*cos(-pi/3*i), obj.radius*sin(-↙
pi/3*i)];
            end
            pos = obj.pos;
        end
    end
end


```

```python
import sounddevice as sd
import pyaudio


class IndexDevices:
    # A list of valid devices
    NAME_OF_MICROPHONE_ARRAY_LIST = [
        'ReSpeaker 4 Mic Array (UAC1.0)  (ReSpeaker 4 Mic Array (UAC1.0)), Windows WASAPI (6 in, 0 out)',
        'ReSpeaker 4 Mic Array (UAC1.0)  (2- ReSpeaker 4 Mic Array (UAC1.0)), Windows WASAPI (6 in, 0 out)',
        'seeed-8mic-voicecard: - (hw:0,0), ALSA (8 in, 0 out)']

    device = None

    # Init
    def __init__(self):
        self.getDevice()
        pass

    # Returns the device from the query list
    def getDevice(self):
        p = pyaudio.PyAudio()

        soundSourceList = str(sd.query_devices()).splitlines()
        for i in range(0, len(soundSourceList)):
            for micArray in self.NAME_OF_MICROPHONE_ARRAY_LIST:
                micNumber = soundSourceList[i].find(micArray)
                if micNumber >= 0:
                    self.device = p.get_device_info_by_index(i)

    # Returns the index number of the audio device
    def getIndex(self):
        return self.device['index']

    # Returns the sample reate of the sound device
    def getSampleRate(self):
        rate = self.device['defaultSampleRate']
        name = self.device['name']
        if name == 'seeed-8mic-voicecard: - (hw:0,0)':
            rate = 96000
        return int(rate)

    # Returns number of channels set in the device
    def getChannels(self):
        return self.device['maxInputChannels']


# Main code
def main():
    print(sd.query_devices())  # To get the full device list
    device = IndexDevices()
    micNumber = device.getIndex()
    print(micNumber)

    # If found, the device will be printed out.
    if micNumber is not None:
        p = pyaudio.PyAudio()
        devinfo = p.get_device_info_by_index(micNumber)
        print(devinfo)


if __name__ == '__main__':
    main()

```

```python
# Room Object for Simulating Room Impulse Responses (RIRs)
# Copyright (C) 2019  Robin Scheibler, Ivan Dokmanic, Sidney Barthe
#
# Permission is hereby granted, free of charge, to any person obtaining a copy
# of this software and associated documentation files (the "Software"), to deal
# in the Software without restriction, including without limitation the rights
# to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
# copies of the Software, and to permit persons to whom the Software is
# furnished to do so, subject to the following conditions:
#
# The above copyright notice and this permission notice shall be included in all
# copies or substantial portions of the Software.
#
# THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
# IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
# FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
# AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
# LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
# OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
# SOFTWARE.
#
# You should have received a copy of the MIT License along with this program. If
# not, see <https://opensource.org/licenses/MIT>.


"""
In this example, we construct an L-shape room, we place one source and one
microphones in the room and compute the room impulse responses. Then change to next
microphone and calculate the RIR.
Since there are 6 mics, we do this six times. At the end, the program adds all 6 channels
 back together into one wav file.
"""
import numpy as np
import wave
import matplotlib.pyplot as plt
from scipy import signal as sci
import pyroomacoustics as pra
from pyroomacoustics.denoise import apply_subspace
try:
    import mkl_fft._numpy_fft as fft
except ImportError:
    import numpy.fft as fft

# todo prøv scipy signal deconvolve, deretter pra.deconvolve, juster param på denoise
samtidig
np.seterr(divide='ignore', invalid='ignore')  # ignore these silly numpy division
warnings :p
"NB: Python does not have any private variables like C++ or Java does."
"""how many times the sound bounces off the walls"""
max_order = 30
visualizeRoom = True  # NB! Will plot once for every iteration
"""https://www.acousticpanelsreview.com/acoustic-sound-absorption/ #plaster walls
absorption"""
"""how much of the sound energy is absorbed by the walls every bounce
http://www.sengpielaudio.com/calculator-RT60Coeff.htm"""
absorption = (0.01 + 0.15) / 2  # Average value of plaster wall absorption
absorption_sabins = 194.921059  # calculated from formula https://www.thermaxxjackets.com
/sabine-modern-architectural/
room_height = 2.40  # z height
room_width = 4.3  # x length
room_length = 1.8  # y length
room_dim = [room_width, room_length, room_height]
"""variables for source 2D and 3D locations (np.array[x, y, z])"""
soundSource2DLoc = [3.4, 0.9]  # [X,Y]cic
soundSource3DLoc = np.array([3.4, 0.9, 0.0])  # [X,Y,Z]
"""For calculating coordinates of each of the 6 mics:"""
x_mic_center_coordinates = 0.9
y_mic_center_coordinates = 0.9
z_mic_center_coordinates = 0.0
mic2DLoc = [x_mic_center_coordinates, y_mic_center_coordinates]  # [X,Y]
room_mic_XYZ_center_locations = np.array(
    [[x_mic_center_coordinates], [y_mic_center_coordinates], [z_mic_center_coordinates
```

```
66 ]])  # [[x1,x2], [y1,y2], [z1,z2]]
67 diameter = 0.0926  # metres
68 radius = diameter / 2  # in meters. radius from centre of core to the microphone
69 pi = np.pi
70 speedOfSound = 343  # m/s
71 """[x,y] The corners of Andreas's hallway"""
72 corners = np.array([[0, 0], [0, 1.8], [4.3, 1.8], [4.3, 0]]).T
73 """Plot the room?"""
74 """ To get wav file with wave library"""
75 file = "../../Recording/audiofiles/examples/echoEnhancedSounds/echoSound1.wav"  #
   Torbjørns file location
76 # file = "C:\\Users\\Andreas\PycharmProjects\\Acoustic-Condition-Control\\src\\Recording
   \\audiofiles\\examples\\echoEnhancedSounds\\echoSound1.wav"
77 # file = sys.argv[1] # this is so you can run the python file. if u comment this,
   uncomment line 24
78 print(file)
79 wav_file = wave.open(file, 'r')
80 nChannelsMicArray = wav_file.getnchannels()  # gets the number of channels
81 numOfMics = nChannelsMicArray  # For good code understanding
82 fs = wav_file.getframerate()  # samples per second
83 nFrames = wav_file.getnframes()
84 soundData_Ch1 = wav_file.readframes(-1)  # Extract Raw Audio from Wav File and checks if
    it's valid
85 ir = []  # impulse response of room
86
87 # create method for stereo check:
88 if wav_file.getsampwidth() == 2:  # if it is 2 it is stereo/several channels
89     soundData_Ch1 = np.frombuffer(soundData_Ch1, dtype='Int16')
90 else:
91     raise RuntimeError("Unsupported sample width")
92
93 # http://schlameel.com/2017/06/09/interleaving-and-de-interleaving-data-with-python/
94 deinterleaved = [soundData_Ch1[idx::wav_file.getnchannels()] for idx in range(wav_file.
   getnchannels())]
95
96
97 def getExperimentalAbsorptionAndRoomOrder_Sabine(t60, room_dim, c):
98     """
99     given desired t60, (shoebox) room dimension and sound speed,
100    computes the absorption coefficient (amplitude) and image source
101    order needed.
102
103    parameters
104    ----------
105    t60: float
106        desired t60 (time it takes to go from full amplitude to 60 db decay) in seconds
107    room_dim: list of floats
108        list of length 2 or 3 of the room side lengths
109    c: float
110        speed of sound
111
112    returns
113    -------
114    absorption: float
115        the absorption coefficient (in amplitude domain, to be passed to
116        room constructor)
117    max_order: int
118        the maximum image source order necessary to achieve the desired t60
119    """
120
121    # finding image sources up to a maximum order creates a (possibly 3d) diamond
122    # like pile of (reflected) rooms. now we need to find the image source model order
123    # so that reflections at a distance of at least up to ``c * rt60`` are included.
124    # one possibility is to find the largest sphere (or circle in 2d) that fits in the
125    # diamond. this is what we are doing here.
126    import itertools
127    R = []
128    for l1, l2 in itertools.combinations(room_dim, 2):
129        R.append(l1 * l2 / np.sqrt(l1 ** 2 + l2 ** 2))
130
131    V = np.prod(room_dim)  # area (2d) or volume (3d)
132    # "surface" computation is diff for 2d and 3d
```

```
    if len(room_dim) == 2:
        S = 2 * np.sum(room_dim)
        sab_coef = 12  # the sabine's coefficient needs to be adjusted in 2d
    elif len(room_dim) == 3:
        S = 2 * np.sum([l1 * l2 for l1, l2 in itertools.combinations(room_dim, 2)])
        sab_coef = 24

    a2 = sab_coef * np.log(10) * V / (c * S * t60)  # absorption in power (sabine)

    if a2 > 1.0:
        raise ValueError(
            "evaluation of parameters failed. room may be too large for required t60."
        )

    absorption = 1 - np.sqrt(1 - a2)  # convert to amplitude absorption coefficient
    import math
    max_order = math.ceil(c * t60 / np.min(R) - 1)

    return absorption, max_order


"""Creates the Room Impulse Response"""


def CreateRIR(soundData_i, x_mic_coordinate, y_mic_coordinate):
    # global max_order, absorption
    """Room impulse response (RIR) generation and propagation simulation
    Using the Image Source Model (ISM) we can compute the impulse response
    for each microphone. From the Room constructor it is possible
    to set the maximum ISM order and the absorption coefficient.
    :param soundData_i: the iterated single channel from 6 channel array
    :param x_mic_coordinate: the X coordinate of the single microphone
    :param y_mic_coordinate: the Y coordinate of the single microphone
    :return: Returns the filtered sound from single channel """

    # set max_order to a low value for a quick (but less accurate) RIR
    # room = pra.Room.from_corners(corners, fs=fs, max_order=max_order, absorption=
absorption)
    # room.extrude(room_height)
    # much more computionally efficient that from_corners above
    room = pra.ShoeBox(room_dim, fs=fs, absorption=absorption, max_order=max_order)

    # add source and set the signal to WAV file content. NB: SoundSource must ne np.
array()
    # todo svakhet i programmet fordi vi kan ikke ha flere lydkilder i rommet. Hver
kilde må ha et eget lydsignal festet
    # til seg i det spesifikke punktet. Hvis vi skal sette dette produktet i et rom med
flere kilder, så vet vi ikke hva
    # slags lyd som kommer i fra hvilken kilde
    room.add_source(position=soundSource3DLoc, signal=soundData_i, delay=0.0)  # changed
 to Global var soundSource3DLoc
    micPosIterated = np.array([[x_mic_coordinate], [y_mic_coordinate], [
z_mic_center_coordinates]])
    # add one-microphone 3D XYZ array add coordinates of every mic
    room.add_microphone_array(pra.MicrophoneArray(micPosIterated, room.fs))
    room.image_source_model(use_libroom=False)
    """gets the impulse response between mic and source"""
    ir = room.compute_rir()
    """deconvolve test to try to get a more dry sound. Demands a very costly computation
"""
    # deconvulved = deconvulve(soundData_i=soundData_i, impulseResponse=ir[0][0]) #
Failed
    # deconvolvedWiener = normalize(wiener_deconvolve(soundData_i, ir[0][0]))
    # inverseDeconvulvedWierner = normalize(np.fft.fft(deconvolvedWiener))
    # plt.figure()
    # plt.plot(deconvolvedWiener[0:nFrames], 'b')
    # plt.plot(inverseDeconvulvedWierner[0:nFrames], 'r')
    # plt.show()
    """calculates the n samples before sound dissipates with 60 db"""
    t60_samples = pra.experimental.measure_rt60(ir)  # experimental code
    """calculates the time in sec before sound dissipates with 60 db"""
    t60_s = pra.experimental.measure_rt60(ir, fs=fs)  # experimental code
```

```
197     from numpy import log as ln
198     nAbsorption_sabins = (24 * ln(10) * room_width * room_length * room_height) / (
199                 speedOfSound * t60_s)  # formula found online
200     """Then the target RT60 and the estimated RT60 match really well! Now I haven't made
     this change just yet because
201     there are some caveats. First, I believe a in Sabine's formula relates to energy,
    while the reflection coefficient
202      is in amplitude (hence the original computation for the reflection). This might
    mean that there is a problem with
203       for example the estimation of the RT60 itself."""
204     # exprimtl_absorp_sab, max_order = getExperimentalAbsorptionAndRoomOrder_Sabine(t60=
    t60_s, room_dim=room_dim, c=speedOfSound)
205
206     # visualize 3D polyhedron room and image sources
207     if visualizeRoom:
208         fig, ax = room.plot(img_order=max_order)
209         fig.set_size_inches(5, 2, forward=False)
210         plt.show()
211     """Moreover, we can plot the RIR for each microphone once the image sources have
    been computed."""
212     if visualizeRoom:
213         plt.figure()
214         room.plot_rir()
215         fig = plt.gcf()
216         fig.set_size_inches(5, 2, forward=False)
217         plt.show()
218
219     """Moreover, we can simulate our signal convolved with these impulse responses as
    such:"""
220     # room.simulate(reference_mic=0, snr=0)  # this method must be called to get access
    to filteredSound
221     room.simulate(reference_mic=0)  # todo skriv om dette med snr og ref mic og
    hovorford et er viktig
222     """returns the simulated RIR sound to add to "dry" original sound"""
223     return room.mic_array.signals[0, 0:nFrames]  # added nFrames
224
225
226 def array_to_wav(audio, filename, fs, mono=False, norm=False, bitdepth=np.float):
227     '''
228     Save all the signals to wav files.
229
230     :param audio: the audi array that you want to save
231     :param filename: str
232     the name of the file
233     :param fs:
234     :param mono: bool, optional
235     if true, records only the center channel floor(M / 2) (default
236     `False`)
237     :param norm: bool, optional
238     if true, normalize the signal to fit in the dynamic range (default
239     `False`)
240     :param bitdepth:  int, optional
241     the format of output samples [np.int8/16/32/64 or np.float
242     (default)]
243     '''
244     from scipy.io import wavfile
245
246     if mono is True:
247         signal = audio[6 // 2]
248     else:
249         signal = audio.T  # each column is a channel
250
251     float_types = [float, np.float, np.float32, np.float64]
252
253     if bitdepth in float_types:
254         bits = None
255     elif bitdepth is np.int8:
256         bits = 8
257     elif bitdepth is np.int16:
258         bits = 16
259     elif bitdepth is np.int32:
260         bits = 32
```

```
261     elif bitdepth is np.int64:
262         bits = 64
263     else:
264         raise NameError('No such type.')
265 
266     if norm:
267         signal = normalize(signal, bits=bits)
268 
269     signal = np.array(signal, dtype=bitdepth)
270 
271     wavfile.write(filename, fs, signal)
272 
273 
274 def normalize(signal, bits=None):
275     '''
276     normalize to be in a given range. The default is to normalize the maximum
277     amplitude to be one. An optional argument allows to normalize the signal
278     to be within the range of a given signed integer representation of bits.
279 
280     :param signal: the array you want to normalize
281     :param bits: None
282     :return: Returns the normalized array
283     '''
284 
285     s = signal.copy()
286 
287     s /= np.abs(s).max()
288 
289     # if one wants to scale for bits allocated
290     if bits is not None:
291         s *= 2 ** (bits - 1) - 1
292         s = clip(s, 2 ** (bits - 1) - 1, -2 ** (bits - 1))
293 
294     return s
295 
296 
297 def clip(signal, high, low):
298     '''Clip a signal from above at high and from below at low.'''
299     s = signal.copy()
300 
301     s[np.where(s > high)] = high
302     s[np.where(s < low)] = low
303 
304     return s
305 
306 
307 def get_circular_2D_mic_array(center, numOfMics, angle1stMic, radius, plot=False):
308     """
309     Creates an array of uniformly spaced circular points in 2D
310 
311     ndarray (2, M)
312         The array of points
313         :param center: array_like
314         The center of the array
315         :param numOfMics: int
316         The number of points
317         :param angle1stMic:  float
318         The counterclockwise rotation of the first element in the array (from
319         the x-axis)
320         :param radius: float
321         The radius of the array
322         :return: returns an evenly spaced circular array of coordinates for circularly
    placed microphones
323     """
324     # print(center)
325     phi = np.arange(numOfMics) * 2. * np.pi / numOfMics
326     micPosArray = np.array(center)[:, np.newaxis] + radius * \
327                   np.vstack((np.cos(phi + angle1stMic), np.sin(phi + angle1stMic)))
328     # x_coordinate = micPosArray[0, i]
329     # y_coordinate = micPosArray[1, i]
330     # return x_coordinate, y_coordinate
331     if plot:
```

```
        plt.figure()
        plt.title("Uniform Circular Mic Array (positions change per iteration)")
        plt.ylabel("Y-axis")
        plt.xlabel("X_axis")
        plt.scatter(micPosArray[0], micPosArray[1])
        x_coordinate = micPosArray[0, 0]# change last pos to see pos of single mic
        y_coordinate = micPosArray[1, 0]# change last pos to see pos of single mic
        plt.scatter(x_coordinate, y_coordinate)# change last pos to see pos of single
    mic
        plt.show()
    return micPosArray


# ********************************************************************************************
    ************************************
def deconvolve(soundData_i, impulseResponse):
    """This is a test to see if the output sound is more dry.
    This is the test to inverse convolve the rooms impulse response with the original
    sound recorded """
    listImpResp = []
    listImpResp.append(impulseResponse)
    while len(impulseResponse) != len(soundData_i):
        listImpResp.append(0.0)
    inverse_impulseResponse = np.power(impulseResponse, -1)

    return sci.deconvolve(soundData_i, inverse_impulseResponse)

def wiener_deconvolve(y, x, length=None, noise_variance=1500., let_n_points=15,
    let_div_base=2):
    '''
    Deconvolve an excitation signal from an impulse response
    We use Wiener filter
    Parameters
    ----------
    y : ndarray
        The recording
    x : ndarray
        The excitation signal
    length: int, optional
        the length of the impulse response to deconvolve
    noise_variance : float, optional
        estimate of the noise variance
    let_n_points: int
        number of points to use in the LET approximation
    let_div_base: float
        the divider used for the LET grid
    '''

    # FFT length including zero padding
    n = y.shape[0] + x.shape[0] - 1

    # let FFT size be even for convenience
    if n % 2 != 0:
        n += 1

    # when unknown, pick the filter size as size of test signal
    if length is None:
        length = n

    # Forward transforms
    Y  = fft.rfft(np.array(y, dtype=np.float32), n=n) / np.sqrt(n)  # recording
    X = fft.rfft(np.array(x, dtype=np.float32), n=n) / np.sqrt(n)   # test signal

    # Squared amplitude of test signal
    X_sqm = np.abs(X)**2

    # approximate SNR
    SNR_hat = np.maximum(1e-7, ((np.linalg.norm(Y)**2 / np.linalg.norm(X)**2) -
noise_variance))  / noise_variance
    dividers = let_div_base**np.linspace(-let_n_points/2, let_n_points, let_n_points)
    SNR_grid = SNR_hat / dividers

```

```
    # compute candidate points
    G = (X_sqm[:,None] / (X_sqm[:,None] + 1./SNR_grid[None,:])) * Y[:,None]
    H_candidates = G / X[:,None]

    # find the best linear combination of the candidates
    weights = np.linalg.lstsq(G, Y, rcond=None)[0]

    # compute the estimated filter
    H = np.squeeze(np.dot(H_candidates, weights))
    h = fft.irfft(H, n=n)

    return h[:length]

def runFilter():
    """
    Takes a single channel and the single position of a microphone and calculates the RIR
    gets a filtered channel in return which it appends to an array and lastly creates a 6 channel array to a .wav file

    :return: Returns a completion message
    """
    global micPosArray
    listFilteredSound = []
    # listDenoisedSound = [] #did not work
    # subtractedSound = [] #did not work
    i = 0
    # micPosArray = np.array([])
    for channel in deinterleaved:  # Get all channels
        if i <= 0:
            micPosArray = get_circular_2D_mic_array(center=mic2DLoc, numOfMics=numOfMics, angle1stMic=0, radius=radius,
                                                    plot=visualizeRoom)

        x_coordinate = micPosArray[0, i]
        y_coordinate = micPosArray[1, i]

        print("Calculating RIR")
        filteredData = np.array(CreateRIR(channel, x_coordinate, y_coordinate)).T

        # denoisedSignal = apply_subspace(noisy_signal=channel, frame_len=256, mu=10, lookback=5, skip=2,
        #                 thresh=0.01, data_type=np.float32, plot=True) #did not work
        # deconvolved = deconvolve(channel, ir) #did not work
        print("deconvolving wiener")
        # deconvolvedSignal = wiener_deconvolve(channel, ir)
        listFilteredSound.append(filteredData)
        # for k in range(len(channel)): #did not work
        #     result = subtractedSound.append(channel[k] - filteredData[k]) #did not work
        if visualizeRoom:
            plt.title("red=Original sound signal. Green=Signal convolved with impulse response to simulate reverb")
            # plt.plot(subtractedSound, 'c') #did not work
            plt.plot(channel, 'r')
            # plt.figure()
            # plt.title("Signal convolved with impulse response to simulate reverb")
            plt.plot(filteredData, 'g')
            # plt.figure()
            # plt.plot(deconvolvedSignal)
            plt.show()
        # listDenoisedSound.append(denoisedSignal) #did not work
        i = i + 1
    # print("recording=", np.shape(deinterleaved),", filtered=", np.shape(listFilteredSound))
    # deconvolved = sci.deconvolve(deinterleaved[0], listFilteredSound[0])
    # print(np.shape(deconvolved))
    # print(deconvolved)
    # plt.plot(deconvolved[1], 'g')
    # plt.figure()
    # plt.plot(deinterleaved[1], 'r')
    # plt.show()
```

```
464     listFilteredSoundArray = np.array(listFilteredSound)
465     # listDeionisedSoundArray = np.array(listDenoisedSound)
466     # denoisedSoundName = "testDenoisedSignal.wav"
467     filteredSoundName = "testFilteredMat.wav"
468     # array_to_wav(deconvolved[1], denoisedSoundName, fs, norm=True, bitdepth=np.int16
    ) #did not work
469     array_to_wav(listFilteredSoundArray, filteredSoundName, fs, norm=True, bitdepth=np.
    int16)
470     print("Filtering done")
471     return "filter done"
472
473
474 if __name__ == '__main__':
475     runFilter()
476
```

DOA estimation

```
close all
clear all
clc
```

ReSpeaker configuration

```
d = 0.0926;                 % Diameter
r = d/2;                    % Radius
M = 6;                      % Number of microphones
Tc = 20;                    % Temperature
c = 331*sqrt(1+(Tc/273));   % Speed of sound
```

Get sound clips

```
path = '2000Hz_Sinewave_Angle_60.wav';      % Sound file

samples = [100,200];                        % Samples to get from the sound file
[audiosignal,fs] = audioread(path,samples); % Reading the sound file.
```

```
fs = 96000
```

```
recordedAngle = cell2mat(extractBetween(path,'Angle','.wav'));

multiplier = 1;
data = resample(audiosignal,fs,fs/multiplier);
```

The maximum amount of samples between the mics

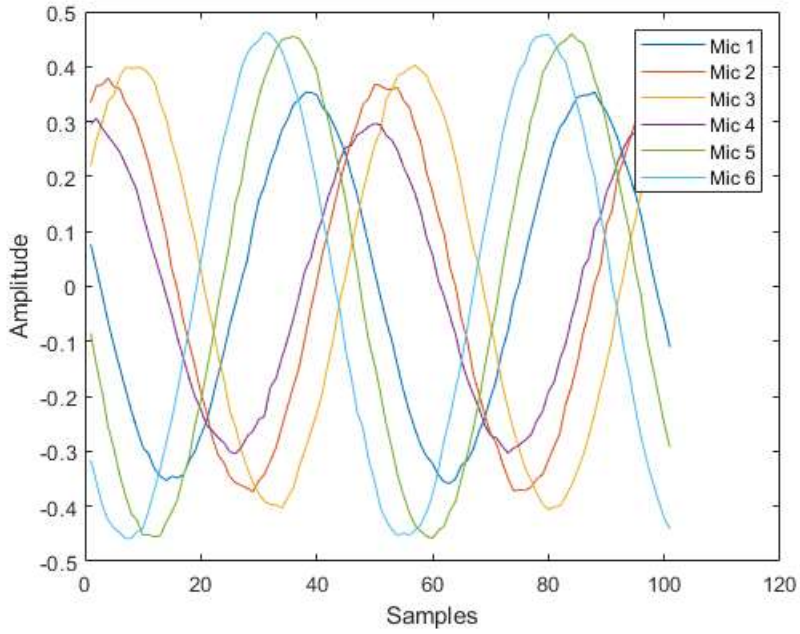
```
tau_max = d/c;

maxFrames = round(tau_max*fs*multiplier);
[-tau_max*fs*multiplier tau_max*fs*multiplier]
```

```
ans = 1×2
  -25.9240   25.9240
```

Plot audiosignal

```
figure()
plot(audiosignal)
xlabel('Samples')
ylabel('Amplitude')
legend('Mic 1', 'Mic 2', 'Mic 3', 'Mic 4', 'Mic 5', 'Mic 6')
```



Calculate Cross-Correlation

```
shiftMic = [];
[idx,lags] = xcorr(data(:,1),data(:,4),maxFrames);
[~, idx] = max(idx,[],1,'linear');            % Get extremal top point
shiftMic(1) = lags(idx)';
clear idx lags
[idx,lags] = xcorr(data(:,2),data(:,5),maxFrames);
[~, idx] = max(idx,[],1,'linear');            % Get extremal top point
shiftMic(2) = lags(idx)';
clear idx lags
[idx,lags] = xcorr(data(:,3),data(:,6),maxFrames);
[~, idx] = max(idx,[],1,'linear');            % Get extremal top point
shiftMic(3) = lags(idx)';
clear idx lags


tau = (shiftMic)/(fs*multiplier);
theta = atan(-tau*c/d);
deg = rad2deg(theta);

shift = [1 1 1];
addAngle = [0 0 0];
```

Correct the angles

```
if sign(shiftMic(1)) == -1 && sign(shiftMic(2)) == -1 && sign(shiftMic(3)) == 1
    shift = [-1 1 1];
    addAngle = [0 0 0];
elseif sign(shiftMic(1)) == -1 && sign(shiftMic(2)) == -1 && sign(shiftMic(3)) == -1
    shift = [-1 -1 -1];
    addAngle = [pi/2 pi/2 pi/2];
elseif sign(shiftMic(1)) == 1 && sign(shiftMic(2)) == -1 && sign(shiftMic(3)) == -1
    shift = [-1 1 1];
    addAngle = [pi/2 pi/2 pi/2];
elseif sign(shiftMic(1)) == 1 && sign(shiftMic(2)) == 1 && sign(shiftMic(3)) == -1
    shift = [-1 -1 -1];
    addAngle = [pi pi pi];
elseif sign(shiftMic(1)) == 1 && sign(shiftMic(2)) == 1 && sign(shiftMic(3)) == 1
    shift = [1 1 1];
    addAngle = [pi/2*3 pi/2*3 pi/2*3];
elseif sign(shiftMic(1)) == -1 && sign(shiftMic(2)) == 1 && sign(shiftMic(3)) == 1
    shift = [1 -1 -1];
```

```
    addAngle = [pi/2*3 pi/2*3 pi/2*3];
end
```

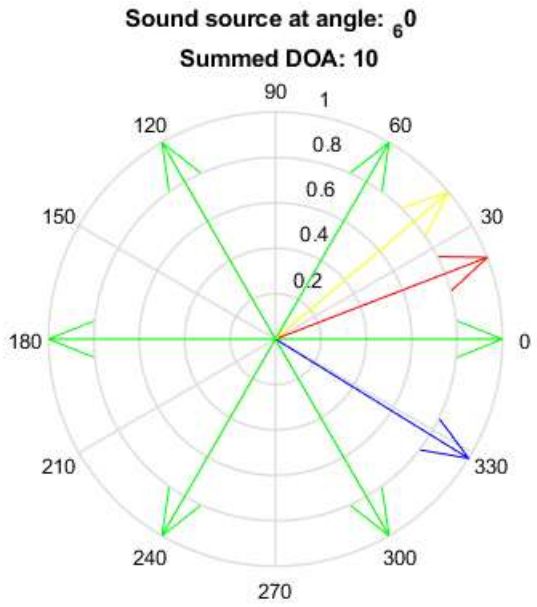
Show the estimated angles

```
angle =  [theta(1)*shift(1)+addAngle(1), theta(2)*shift(2)+addAngle(2), theta(3)*shift(3)+addAngle(3)];


summedDOA = rad2deg(sum(angle)./3); %Summed angle
if sign(summedDOA) == -1
    summedDOA = 360+summedDOA;
end


r = exp(1i * ([pi/3, pi/3*2, pi/3*3, pi/3*4, pi/3*5, pi/3*6])); % Placement of the microphones
r1 = exp(1i * (angle(1)));  % Cross-Correlated between mic 1 and 4
r2 = exp(1i * (angle(2)));  % Cross-Correlated between mic 2 and 5
r3 = exp(1i * (angle(3)));  % Cross-Correlated between mic 3 and 6

figure()
compass(r,'g')
hold on
compass(r1,'r')
compass(r2,'b')
compass(r3,'y')
title(sprintf('Sound source at angle: %s \n Summed DOA: %0.f', recordedAngle, summedDOA))
```



F

Test rapport Beamforming 1

Dato: 17.03.2020

Omhandler:

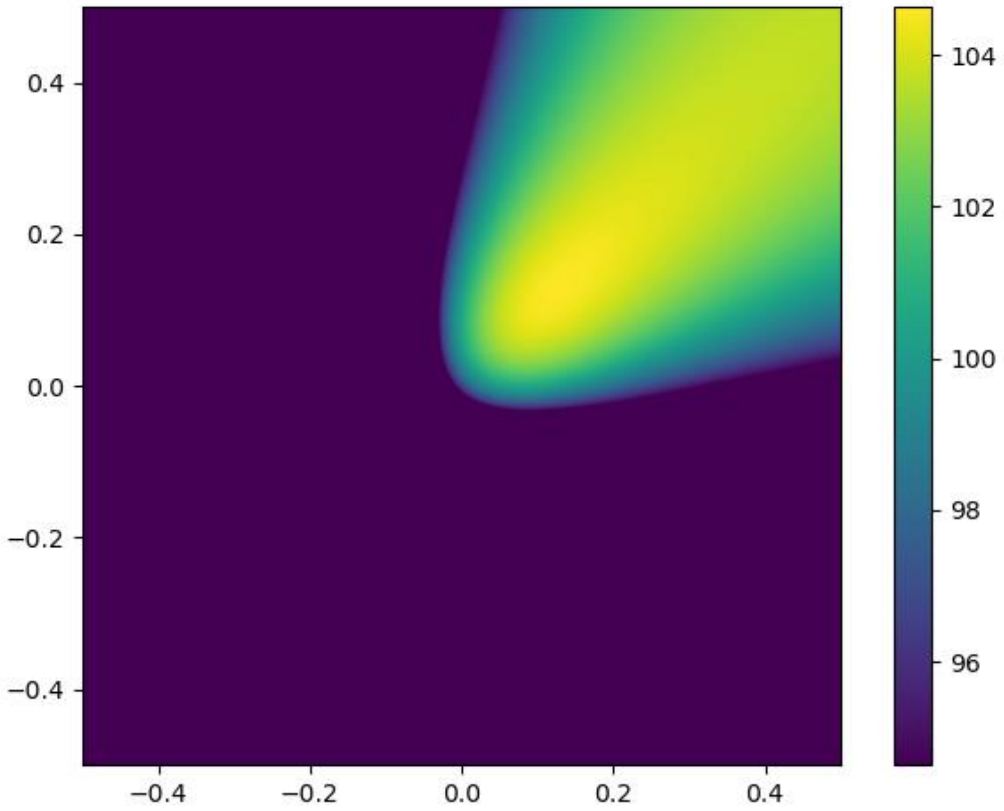
Test av beamforming. Underveis i oppgaven fant gruppen ut et bibliotek som kan kalkulere beamforming. dette biblioteket heter Acoular og er ment for python. Ved å sette posisjonene til mikrofonene, kan vi med hjelp av acoular kalkulere ut en beamform mot lydkilden. Under kan du se ulike tester hvor gruppen tar opp 1 sekund med lydsignal og kjører dette gjennom koden. Ut får vi et varmekart etter hvor beamformen ligger og i hvilken retning det lyttes på.

Programmvare	**Verson og beskrivelse**
acoular	
Numpy	
Pylab	

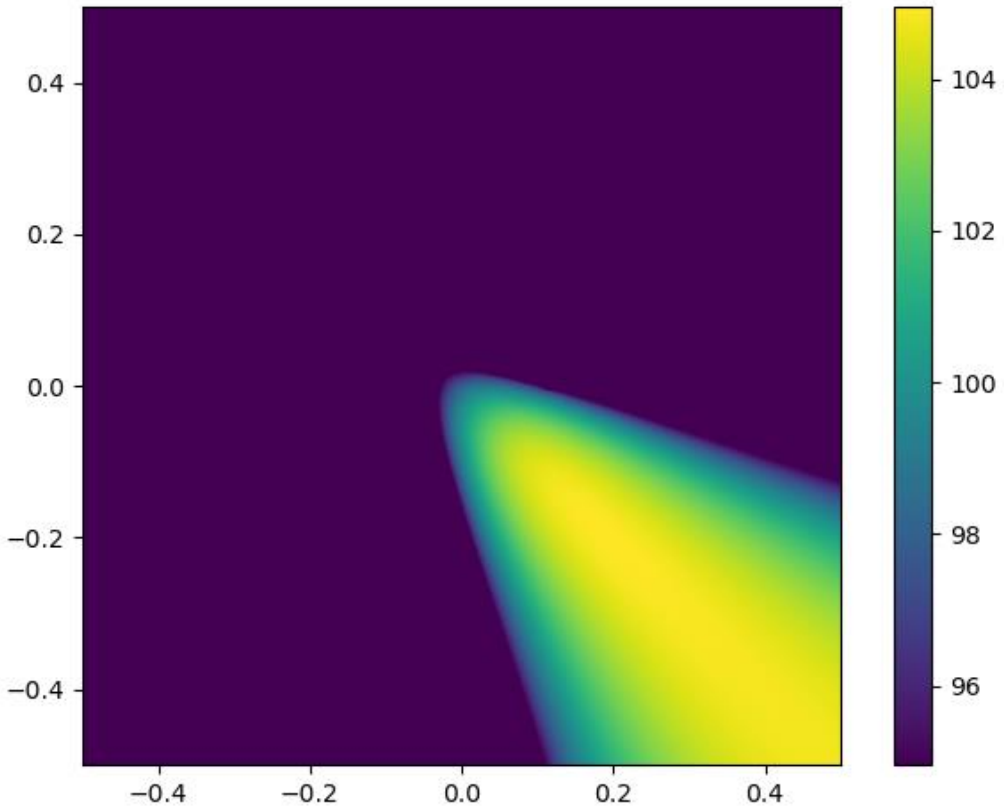
Resultat

Resultat	**Beskrivelse**
1	Test med nynning fra mic posisjon 1. ca 40cm unna.
2	Test med nynning fra mic posisjon 2. ca 40cm unna.
3	Test med nynning fra mic posisjon 4. ca 40cm unna og mobil med musikk avspilt fra mic 3. I testen kan du se refleksjon(ekko) diagonalt reflekteres i de 2 andre mikrofonene

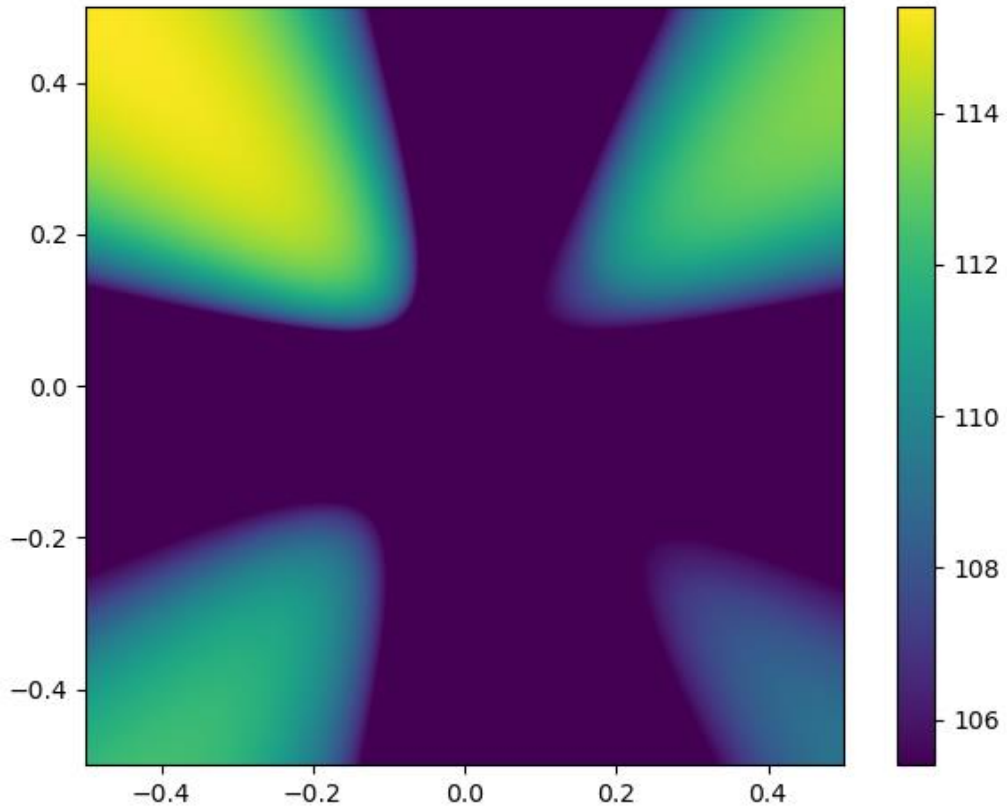
1.



2.



3.



	Ja	Nei	Beskrivelse
Blei testen gjennomført som forventet?		x	Viser ikke rett resultat, dette bare illustrerer en retning og ikke et beamformet resultat som gruppen ønsker.
Problem som oppstod underveis?	x		Tidligere har det blitt testet på 8000 i frekvens range. Denne gangen prøvde vi 4000 å det funket.
Feilkilder?	x		Feil frekvensrange
Forbedringer til neste test?		x	Dette kommer ikke til å bli brukt senere i oppgaven.

Test rapport Beamforming gain plots

Dato: 29.04.2020

Omhandler: Tester beamforming i 360 grader med oppløsning på 1 grad. Bruker dette for å finne rettninger til kilder og forsterkning med hjelp av beamforming. Flere kilder kan bli funnet med denne metoden, men på grunn av kodens tidlige fase er funksjonen svak for støy under opptakene.

Programmvare	Verson og beskrivelse
Matlab	2020a: BeamMultiSourceGain.m
Matlab: Sensor array analyzer	Brukt for å forutse fasong på loben i beamforming.

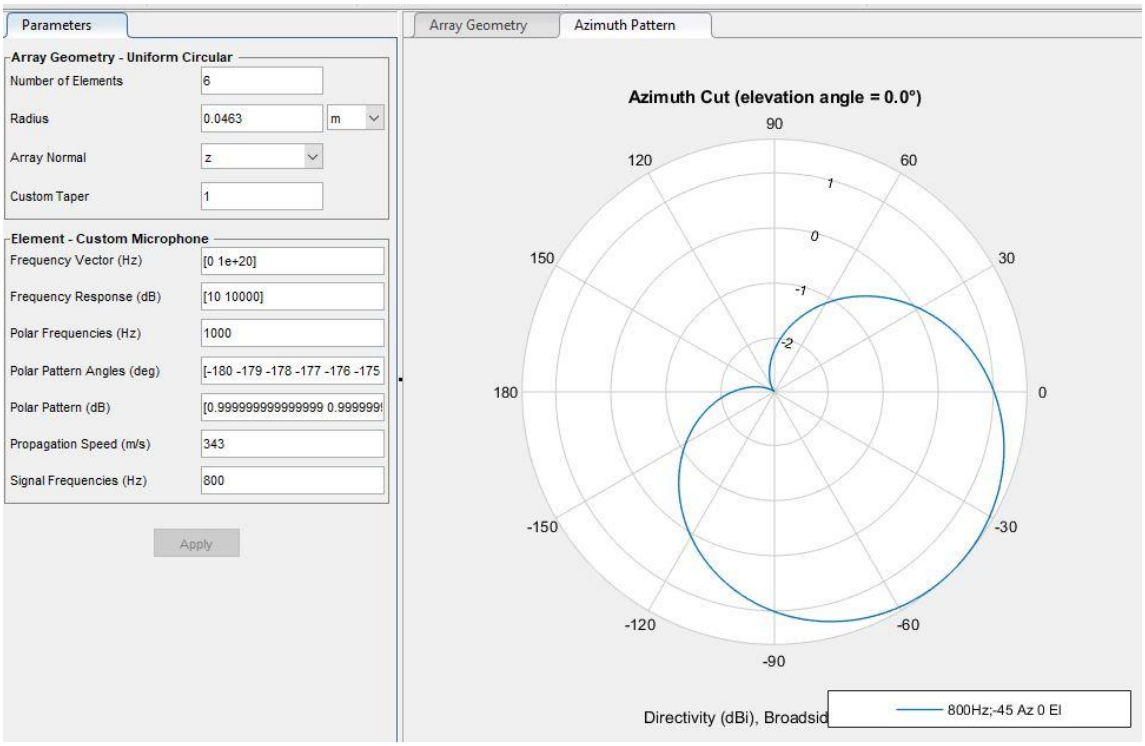
Resultat

Resultat	Beskrivelse
1	1200Hz_Multisource_0000.wav med kilder i vikel 0 og -45 grader. Finner begge. Kurvene viser forutsett fasong fra sensor array analyzer. Vinklene er som forventet.
2	1200Hz_Multisource_5625.wav med kilder i vinkel 5,625 og -45 grader. Kurvene viser forutsette fasonger. Men vinklene er ikke helt nøyaktig +-5 grader.
3	1200Hz_Multisource_11250.wav. vinklene er ikke som forventet og kurvene representerer ikke forventede fasonger. Dette skyldes dårlig opptak eller dårlige høyttalere.
4	

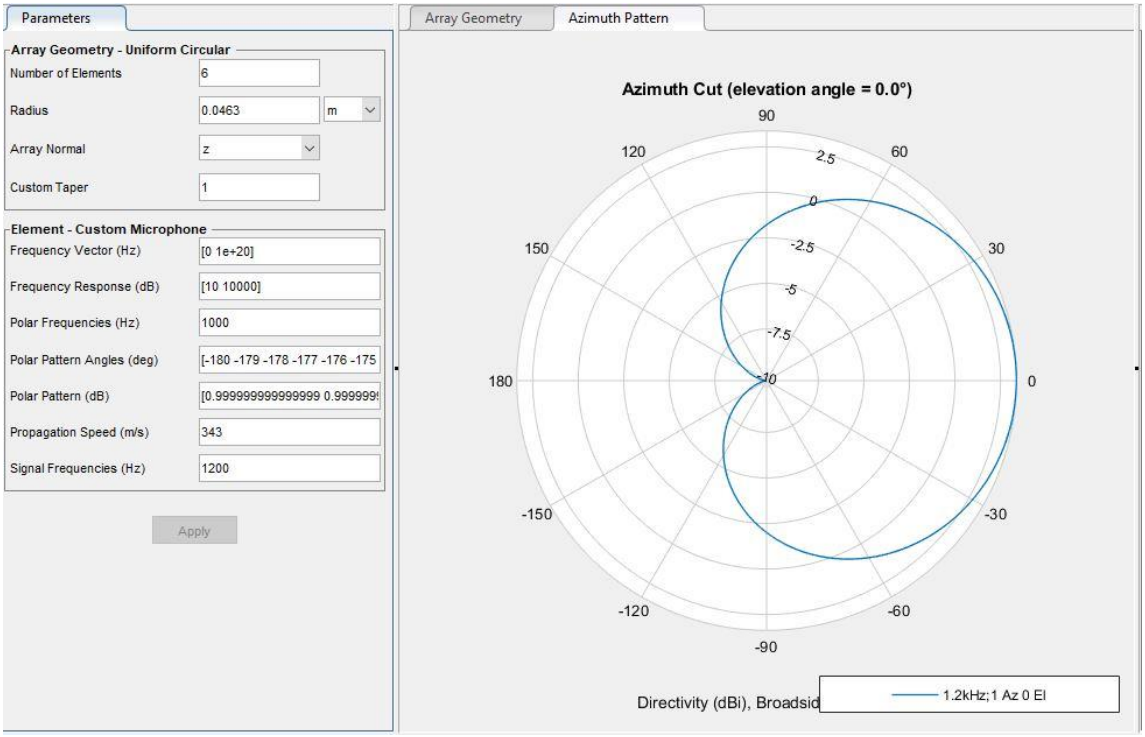
	Ja	Nei	Beskrivelse
Blei testen gjennomført som forventet?		x	Noen avvik pga feilkilder.
Problem som oppstod underveis?	x		Høytalere brukt til avspilling klarer ikke mer en noen sekunder før lyden fraviker fra ønsket verdi. under testing blir også høytalerne ødelagte pga ren sinusbølge.
Feilkilder?	x		Dårlige høyttalere, påvirkning av gulvvarme, rommets tempratur.
Forbedringer til neste test?		x	Ved bedre opptak kan en bedre test gjennomføres. Men under forholdene i bacheloren og tiden, er dette dessverre ikke mulig å gjennomføre. Innkjøp av nye høytalere, bedre testlokale…etc

Utdrag fra første test med lydfilen 1200Hz_Multisource_0000.wav:

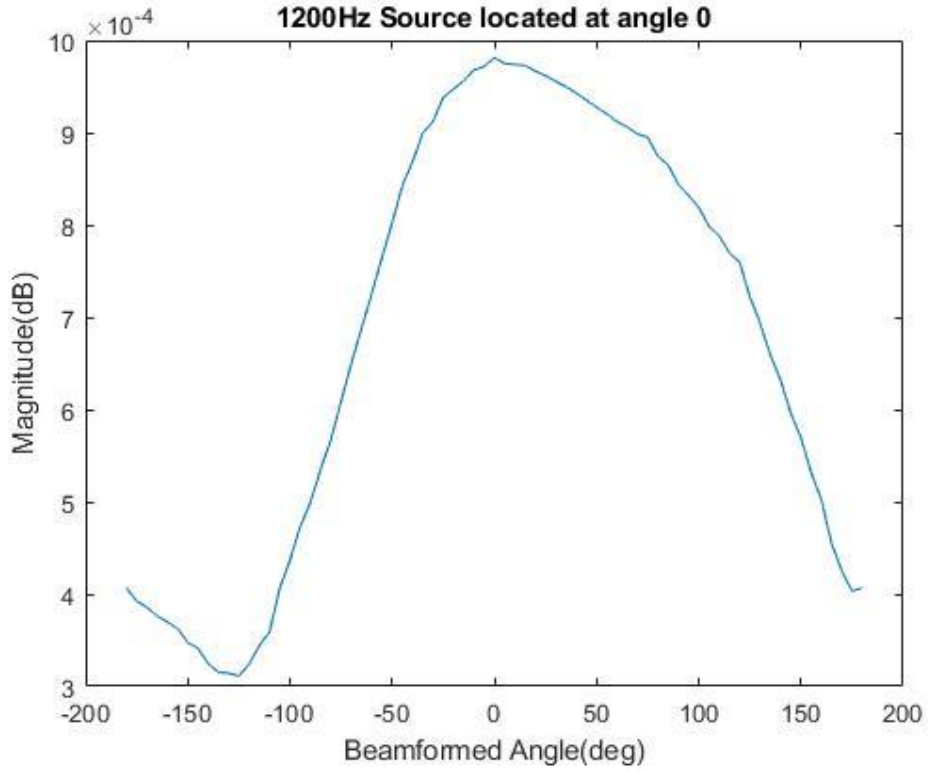
Forutsett utforming på lobene på 800Hz i vinkel -45

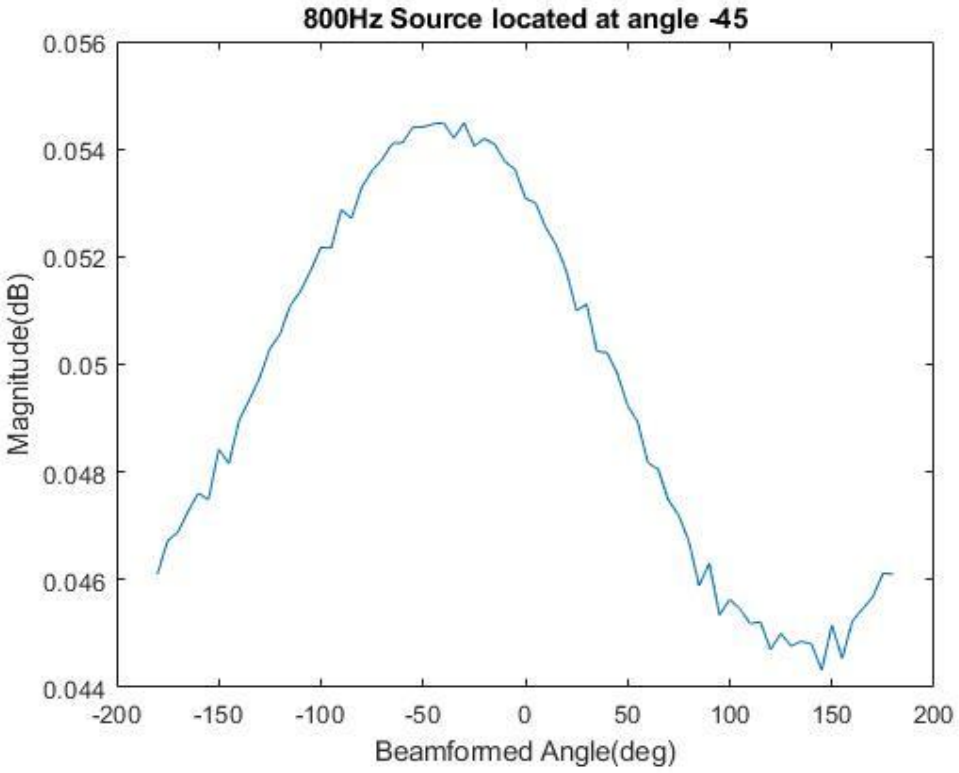


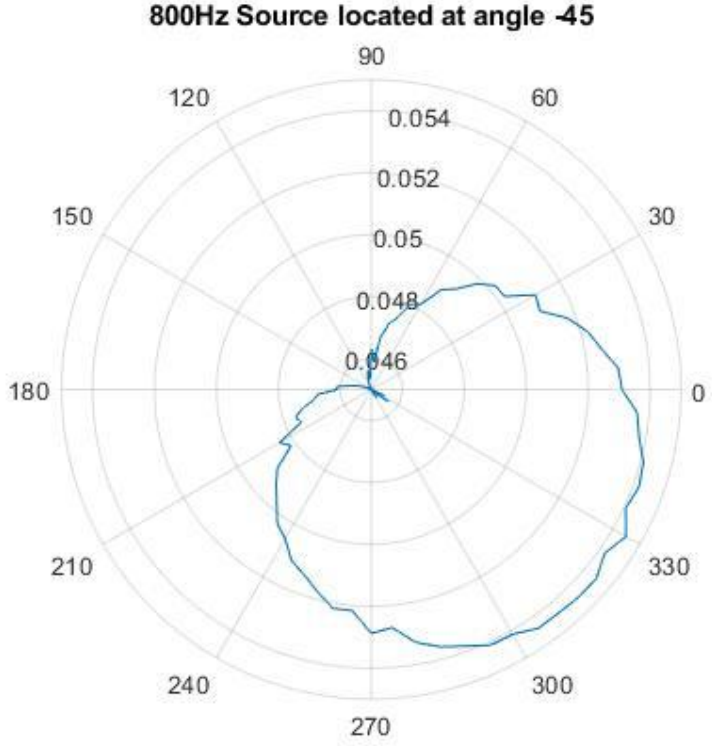
Forutsette fasong på loben med 1200Hz i vinkel 0

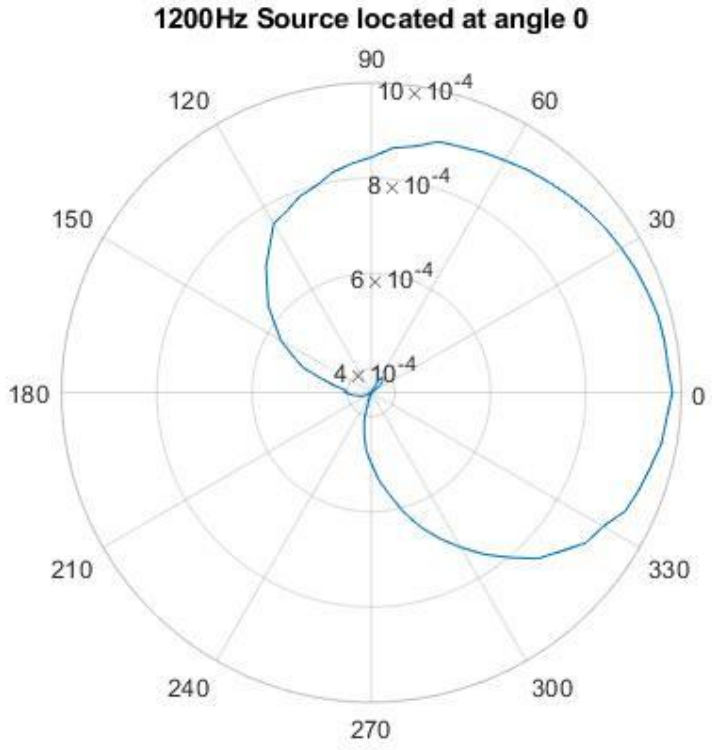


På grafene er det vist dB på y-aksen, dette er ikke tilfelle da det kun er plottet i magniutude.




800Hz Source located at angle -45
0.056
0.054
0.052
0.05
0.048
0.046
0.044
Magnitude(dB)
-200
-150
-100
-50
0
50
100
150
200
Beamformed Angle(deg)


800Hz Source located at angle -45
90
120
60
0.054
0.052
150
30
0.05
0.048
0.046
180
0
210
330
240
300
270


1200Hz Source located at angle 0
90
120
60
150
30
180
0
210
330
240
300
270
10×10^-4
8×10^-4
6×10^-4
4×10^-4

Test rapport Kildeseperasjon

Dato: 14.04.2020

Omhandler: Seperasjon av 3 kilder i et opptak

Programmvare	Verson og beskrivelse
Matlab	
ThreeSources	Matlabkode

multisource6.wav - Recording duration: 5s

Source 1:

1.5m distance from microphone

Angle: 0

The Weeknd: In Your Eyes Link

Source 2:

1.5m distance from microphone

Angle: 300

Tones And I: Bad Child Link

Source 3:

1.5m distance from microphone

Angle: 240

Lil Mosey: Blueberry Faygo Link

Resultat

Resultat	Beskrivelse
1 - Weekend	Gjenkjenner sangen, men lav og støyete
2 - Tones	Stemmen og enkelte bakgrunns lyder kommer fint frem. Beger preg av "muffling" og litt støy.
3 - lilMosey	Kommer veldig fint frem, litt støy men er veldig bra ellers.

Resultatene ligger under i vedleggsmappen "kildeseperasjon"

	Ja	Nei	Beskrivelse
Blei testen gjennomført som forventet?		x	Resultatet var bedre en formventet
Problem som oppstod underveis?	x		Under kodingen frem til at dette fungerte var det mange problemer. Men under testen var det ingen.
Feilkilder?		x	
Forbedringer til neste test?	x		Få koden til å kalkulere chunk-vis og finne en måte til å redusere støyet.

Test rapport Odas web

Dato: 19.02.2020

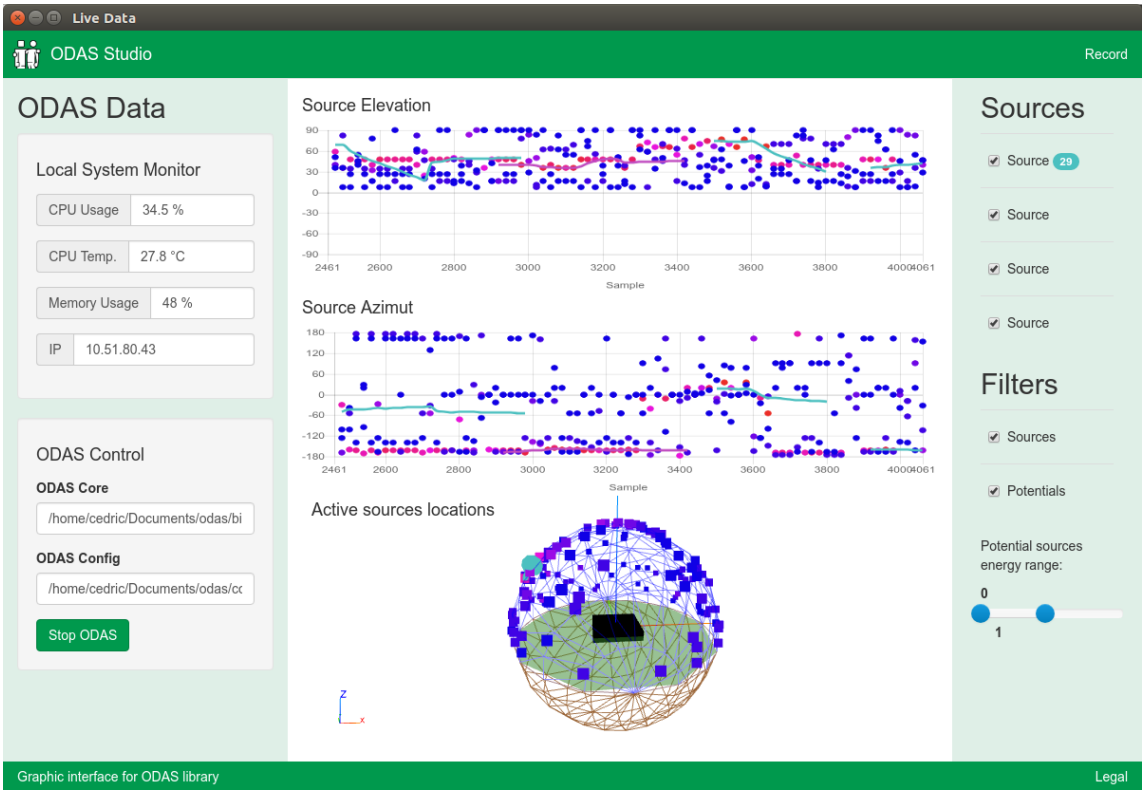
Omhandler: Test av Odas web sammens med Respeaker mic array v2

Programmvare	Verson og beskrivelse
Odas web	GUI applikasjon for å ta i bruk Odas, versjon v0.3
Odas library	Bibliotek for lokalisering, isolering av lydkilder. V1
Virtual box	Tatt i bruk pga Odas kun støtter ubuntu.
Ubuntu	OS for å kjøre odas, V18.4
Raspbian	Brukt for å oppdatere fireware, dette ble tatt ibruk pga det ikke fungerer å oppdatere drivere gjennom virtual machine og at det må brukes linux for å gjøre dette.

Resultat

Resultat	Beskrivelse
Hardware	Hardware fungerte fra start, problemet lå på software siden.
Software	Her lå det en del problemer
Kombinert resultat	Etter mye testing oppnådde gruppen ønsket resultat og programmet fungerte som det skal.

	Ja	Nei	Beskrivelse
Blei testen gjennomført som forventet?	x		Noen problemer underveis, men resultatet er som forventet, litt tregere en det burde men dette skyldes dårlig pc.
Problem som oppstod underveis?	x		Mange. først så fungerer ikke Odas på Windows. så gruppen utførte dette i Virtual Box, dette utløste flere problemer da tilkoblede USB enheter må gå gjennom windows og så inn på VB. Det lå også problemer da config fila til Respeaker er feil, som førte til sampling og kanal feil. Dette ble løst ved endring i filen.
Feilkilder?	x		Mulig dårlig pc
Forbedringer til neste test?	x		Bedre PC til å kjøre programmet, helst en dedikert pc som kjører linux.



Kilder:

https://github.com/respeaker/usb_4_mic_array

Test rapport Opptak av lyd

Dato: 03.02.2020

Omhandler: opptak av lyd via python og pc mikrofoner

Programmvare	Verson og beskrivelse
Python 3.8	
pyAudio 0.2.11	bibliotek som lar deg ta opp lyd
wave	Bibliotek som lar deg lagre et opptak som wav-fil

Resultat

Resultat	Beskrivelse
Opptak 1	Plystring ble tatt opp iløpet av 3 sekunder. Ble spilt av like etterpå
Opptak 2	hosting
Opptak 3	Plystring fra distanse

	Ja	Nei	Beskrivelse
Er det nokk tester til å trekke en konklusjon?	x		Yes. Programmet funker som forventet
Blei testen gjennomført som forventet?	x		Fikk mye brukelig nyttig informasjon om programmets funksjonalitet.
Problem som oppstod underveis?		x	
Feilkilder?		x	Nei. Men det kom frem under testing av mikrofon er dårlig.
Forbedringer til neste test?	x		Bruk av bedre mikrofoner, eller i så fall være obs på at mikrofonen ikke har lang rekkevidde.

Test rapport Opptak av lyd

Dato: 20.02.2020

Omhandler: Opptak av lyd ved hjelp av Respeaker Mic Array v2, 6 kanals opptak.

Programmvare	Verson og beskrivelse
Python	V 3.8
PyAudio	v0.2.11 behandle lyd
Wave	Lagring av lydfiler som wav fil
Numpy	V1.18.1 brukt til å ta ut enkelte kanaler fra innput stream, som består av 6.

Resultat

Resultat	Beskrivelse
Opptak kanal 1-4	Her ble det tatt opp rådataen fra mic 1-4 på Respeakeren
Opptak kanal 0	Her ble det tatt opp lyd fra alle kanalene samlet.
Opptak kanal 5	Her ble det tatt opptak av signaler sendt ut til mini-jacken på respeakeren, det ble ikke sendt noe ut til enheten og derfor ingen opptak.

	Ja	Nei	Beskrivelse
Blei testen gjennomført som forventet?	x		
Problem som oppstod underveis?	x		Litt problemer med instillingene og valg av index nummer til Respeakeren. Dette kan ungås ved å liste opp alle tilkoblede enheter.
Feilkilder?		x	
Forbedringer til neste test?		x	Alt fungerer som det skal.

Respeaker mic Array v2 er oppdatert firmware på, dette gjør at den har 6 kanaler i stedet for 1, som er fabrikkinstilling.

Test rapport for funksjonstest av matlab kode

Dato: 26.03.20

Omhandler: tidsforskyvning av rådata

Programmvare	Verson og beskrivelse
Matlab	

Resultat

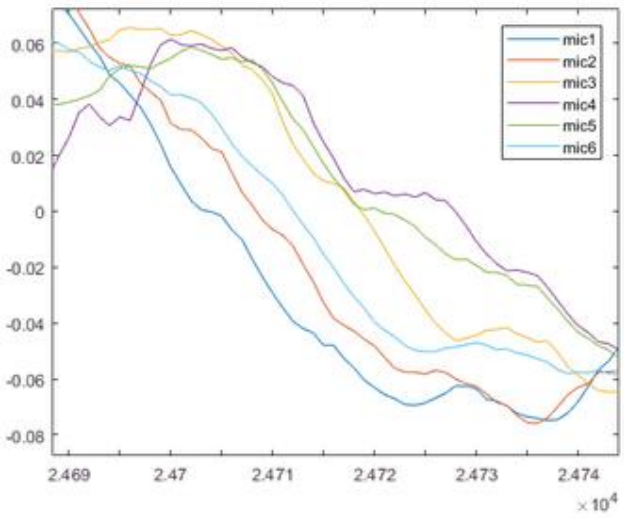
Resultat	Beskrivelse
1	Test ved hjelp av teoretisk kalkulering av tidsdelay
2	Test ved hjelp av "cross validation" av amplitudene i signalet

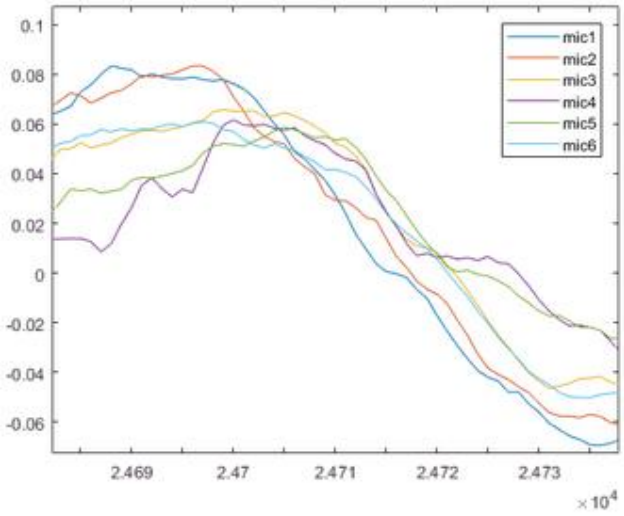
	Ja	Nei	Beskrivelse
Blei testen gjennomført som forventet?	x		
Problem som oppstod underveis?	x		Tidsforskyvning av signalet ved opptak, div ulike småfeil i koden.
Feilkilder?	x		For lav samplingsfrekvens
Forbedringer til neste test?		x	

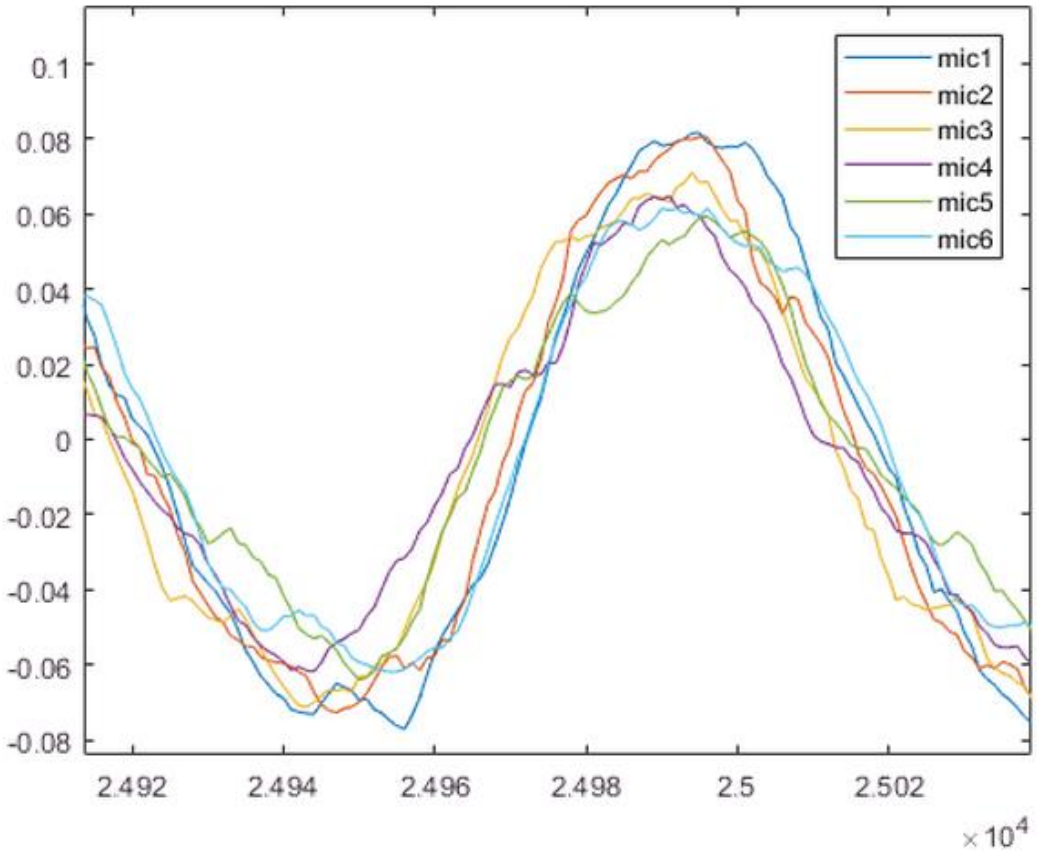
Under i testen ble det tatt inn 6 kanals rådata fra 6 mikrofoner. Signalene er forsinket i forhold til hverandre pga posisjoneringen til mikrofonene. Ved hjelp av 2 fremgangsmåter så skulle gruppen revansjere denne tids-differansen. To ulike fremgangsmåter ble gjort i testen. Den første metoden var å bruke avstanden mellom mikrofonene, sammen med avstanden fra lydkilden og noen andre kjente verdier til å kalkulere en teoretisk tidsforskjell mellom måletiden på mikrofonene. Under kan vi se at signalet blir mer konsentrert en det er opprinnelig i det ubehandlede signalet.

Test 1

Orginalt signal sammens med det tidsjusterte signalet.

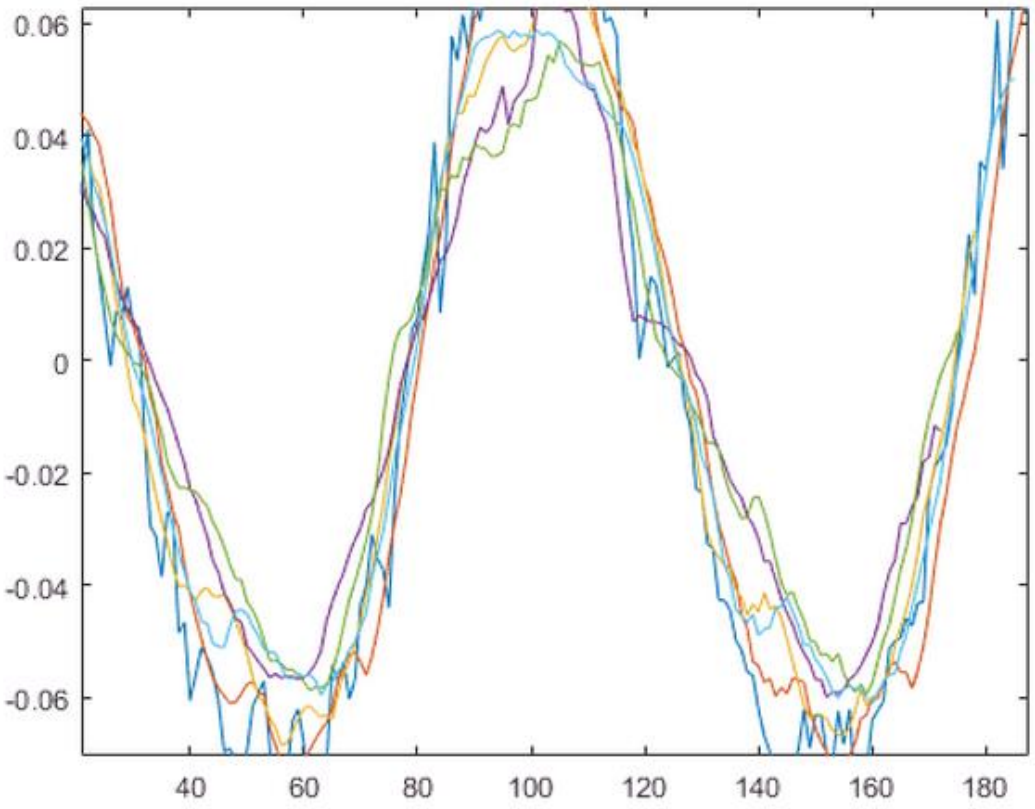





mic1
mic2
mic3
mic4
mic5
mic6
0.1
0.08
0.06
0.04
0.02
0
-0.02
-0.04
-0.06
-0.08
2.492
2.494
2.496
2.498
2.5
2.502
×10^4

I fremgangsmåte to brukte gruppen "cross-validation" til å kalkulere delayet mellom de ulike signalene. Denne metoden finner den høyeste summerte amplituden mellom et signal og et referanse signal. Dette kan da gruppen brukte til å finne samplings nummeret og forskyve signalet deretter. Ved å gjøre dette på alle signalene, vil disse befinne seg nærmest mulig referanse signalet.

Test 2



de ^

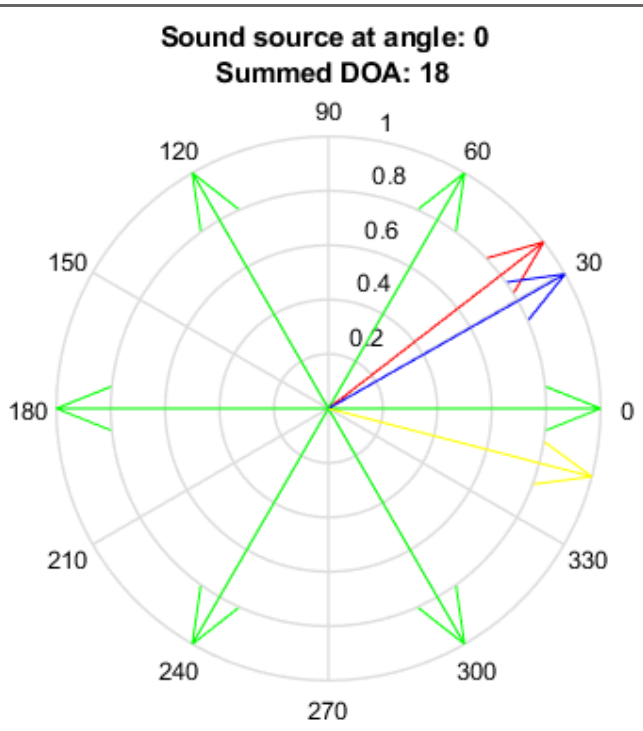
Test rapport triangulering

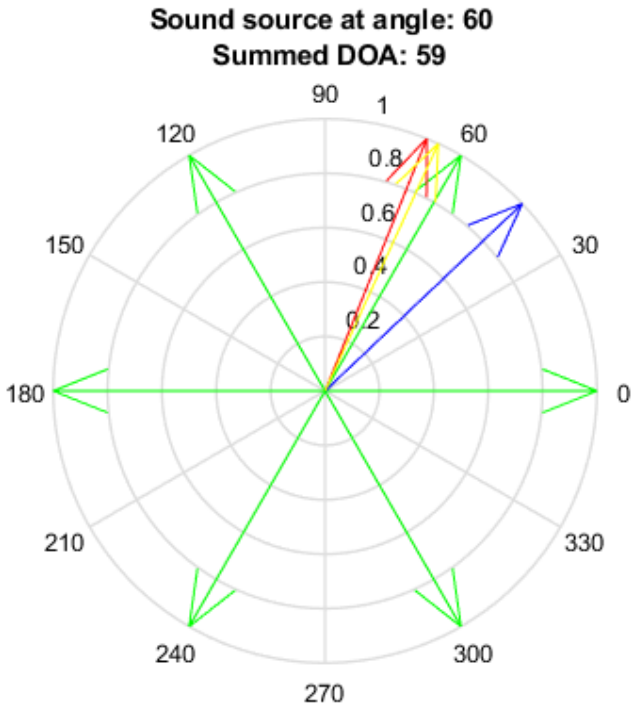
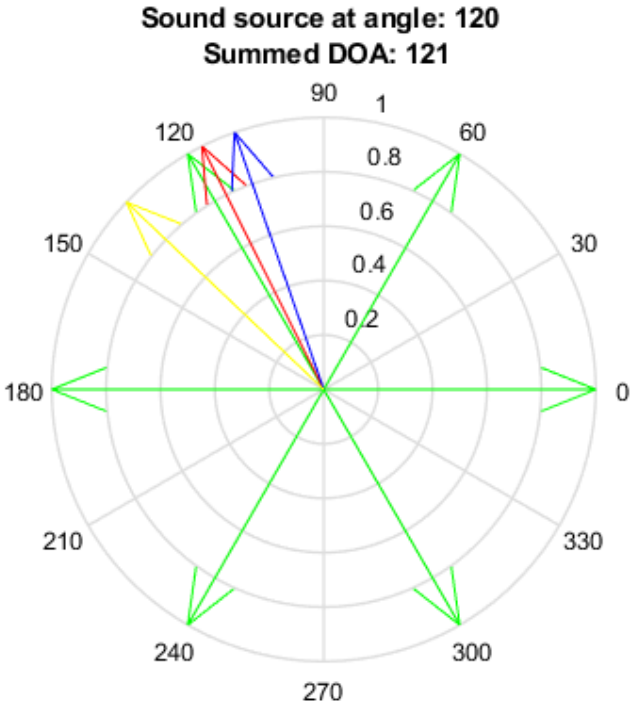
Dato: 06.04.2020

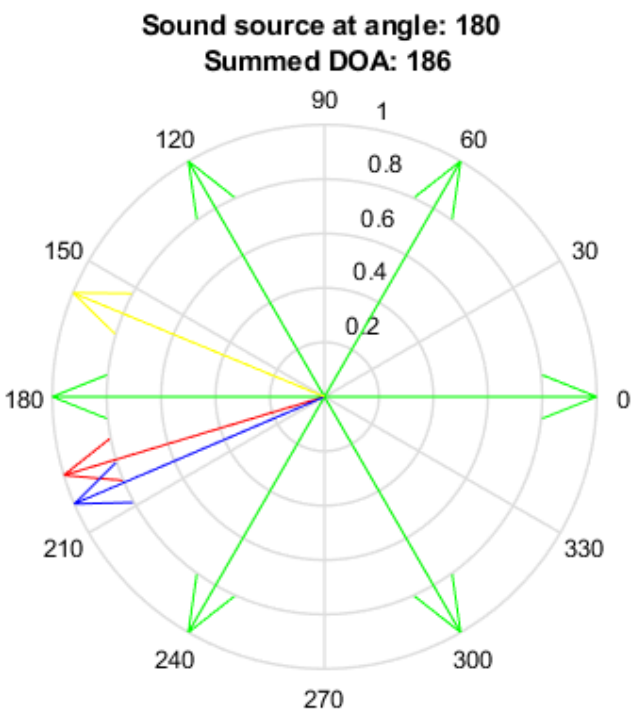
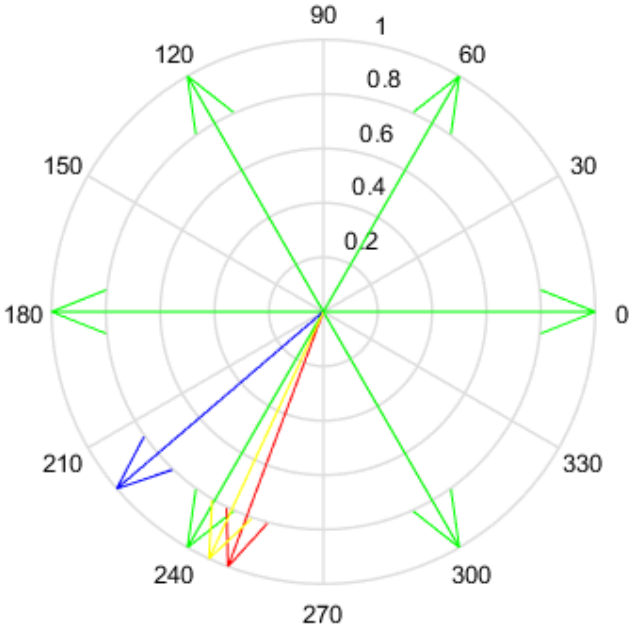
Omhandler: DOA estimering av en lydkilde

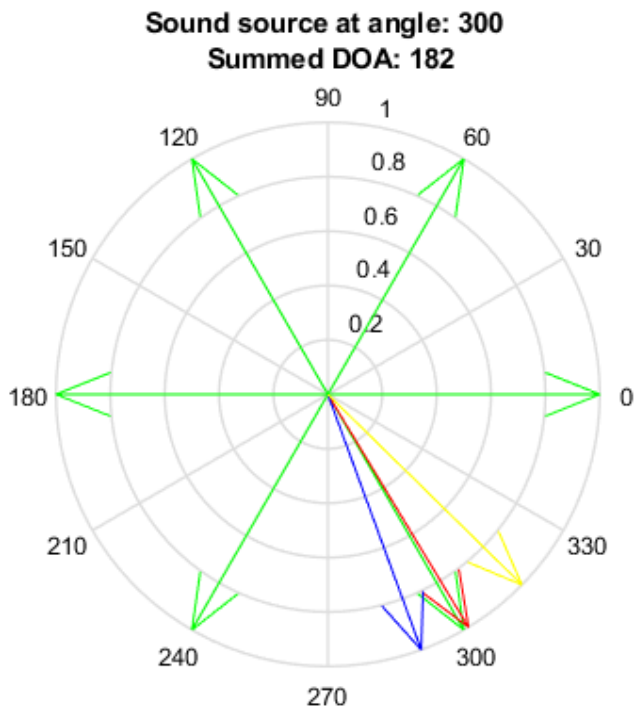
Programvare	Version og beskrivelse
Matlab	R2019a

Resultat

Resultat	Beskrivelse
1. Lydkilde plassert på 0 grader	

2. Lydkilde plassert på 60 grader	
3. Lydkilde plassert på 120 grader	

4. Lydkilde plassert på 180 grader	
5. Lydkilde plassert på 240 grader	

6. Lydkilde plassert på 300 grader	

	Ja	Nei	Beskrivelse
Blei testen gjennomført som forventet?		X	Den summerte vinkelen på alle steder uten om 0 og 90 grader ganske bra
Problem som oppstod underveis?	X		På grunn av cosinus så blir 0 og 90 grader veldig feil
Feilkilder?	X		Bruker cosinus for å beregne vinkelen
Forbedringer til neste test?	X		Bytte til tangens

Test rapport Triangulering

Dato: 02.03.2020 , Andreas og Finn-Christian

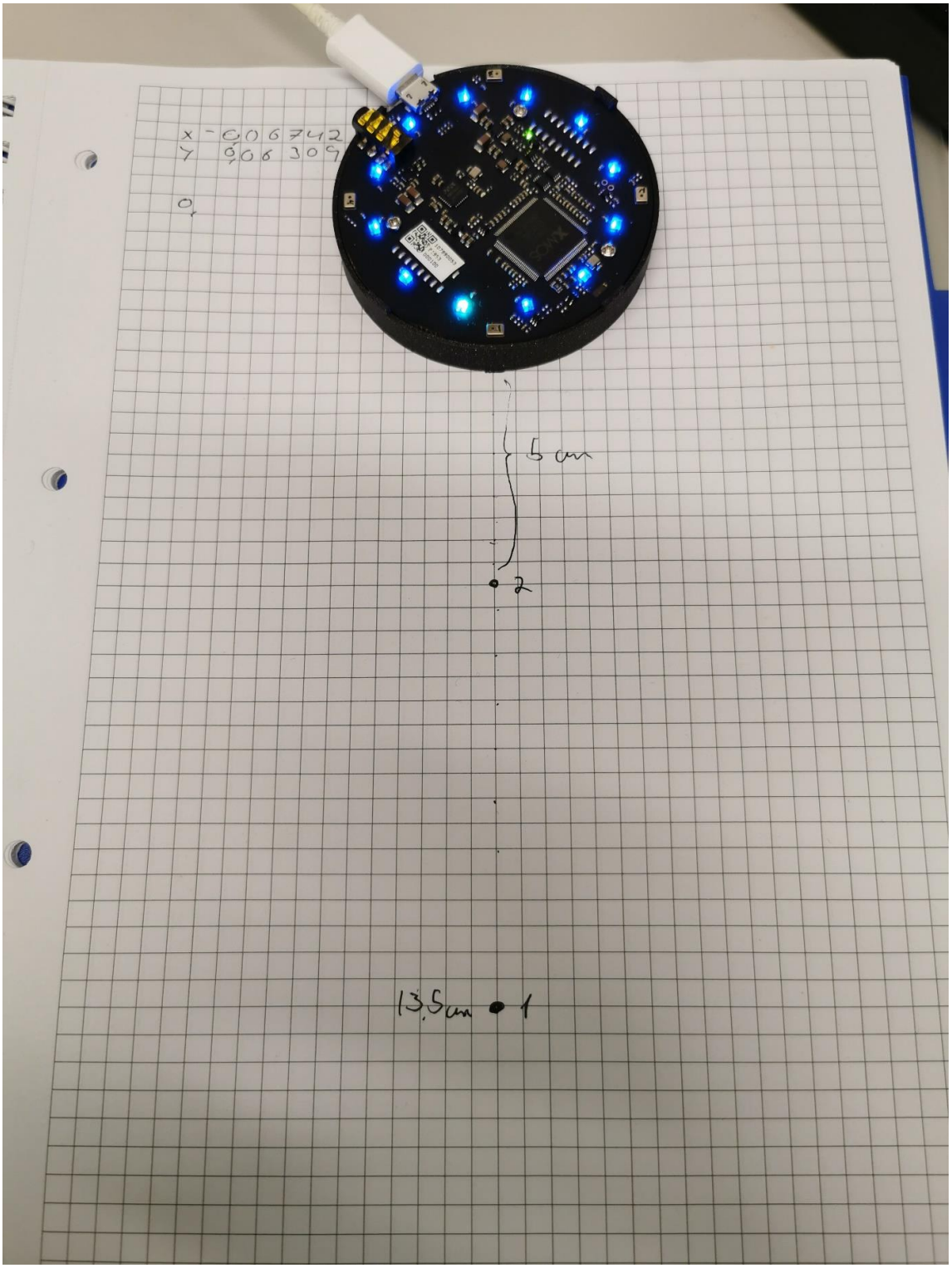
Omhandler: triangulering på hensyn med faseforskyvning mellom inngangssignal(magnitude).

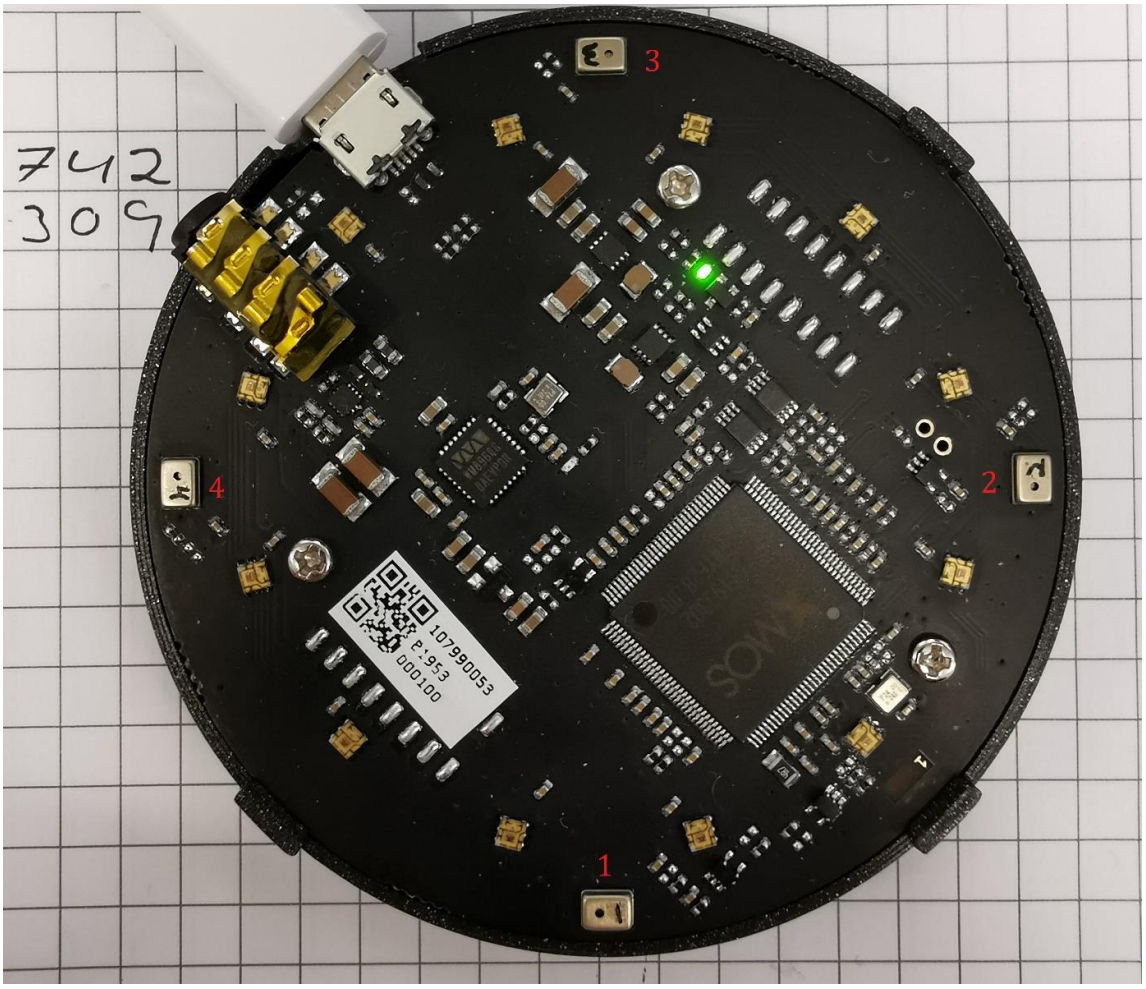
Forklaring:

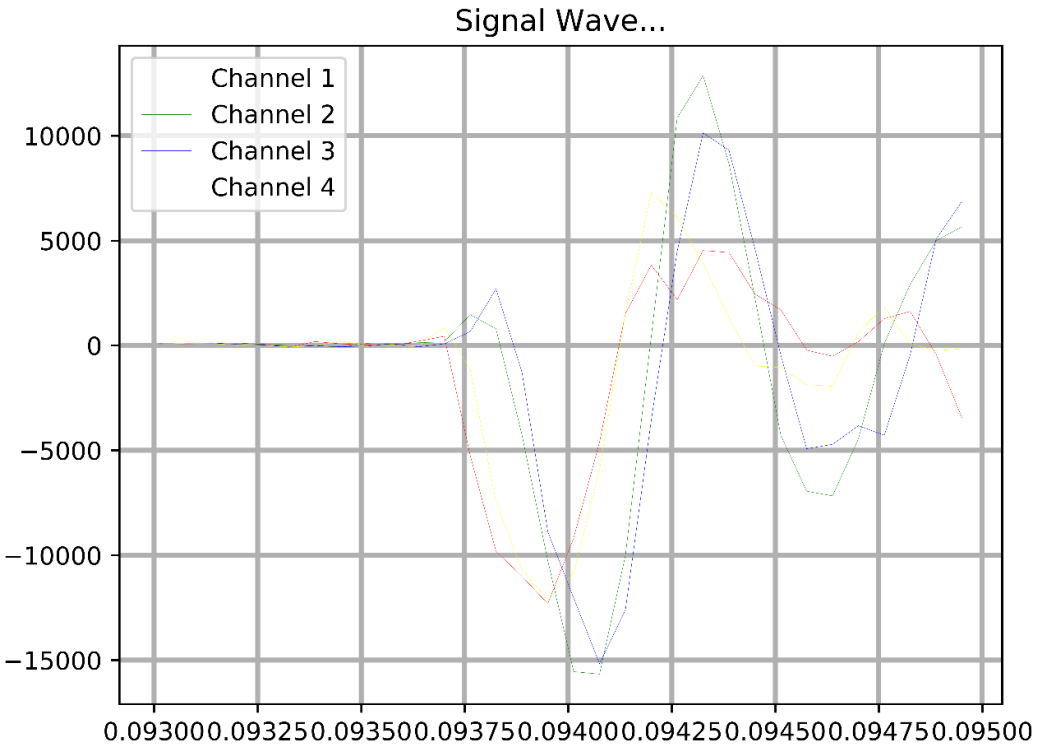
Gruppen ønsket å gjennomføre en veldig simpel test på hvor godt visuelle kart kan indikere retningen på en enkelt lydkilde. Det ble laget et enkelt skript i matlab som plottet sirkler rund mikrofonene på pucken som skulle indikere tiden(avstanden) fra en potensiell lydkilde og til hver enkelt mikrofon. Under kan du se bilder fra to figurer, hvorav de bruker ulike metoder for å kalkulere sirklene.

Programmvare	**Versjon og beskrivelse**
Matlab 2019	For å lage figurer og kalkulere tidsforskyvning
Pycharm	For å kjøre koden for opptak av lyd
RecordChannels code	For å ta opp lyd gjennom ReSpeaker 4 Mic Array og omgjøre det til wav

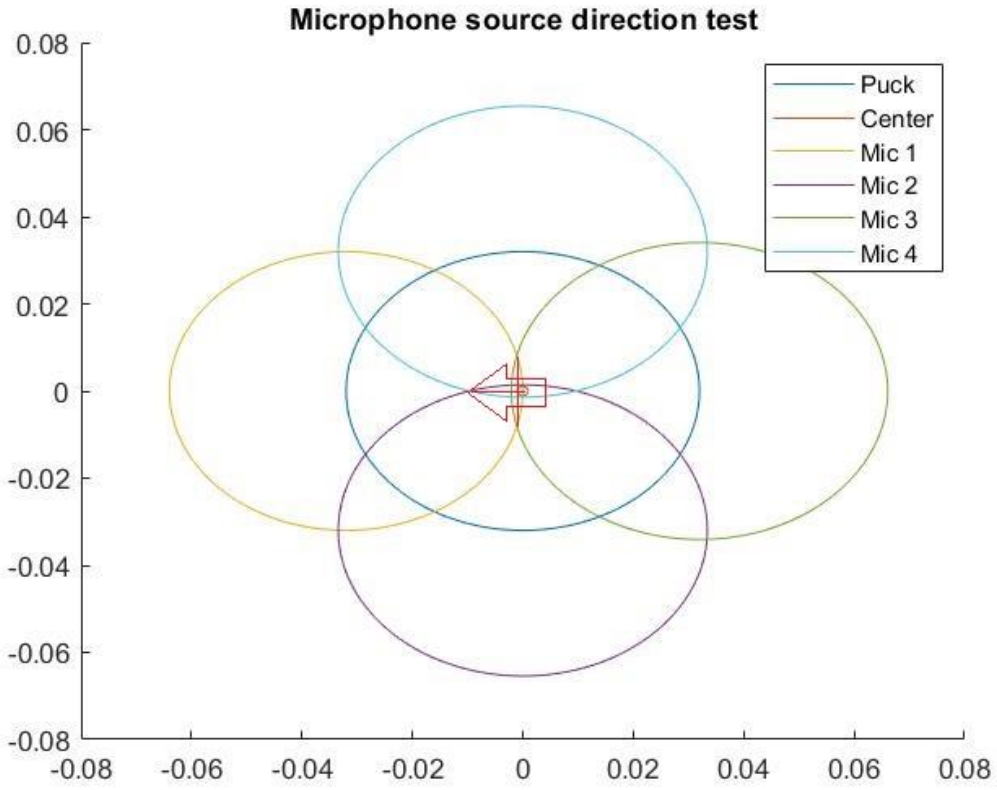
Hardware	**Verson og beskrivelse**
PC	
Respeaker 4 mic array	6 channel 4 mic array


x -0,06742
y 0,06309
0,
5 cm
2
13,5cm
1

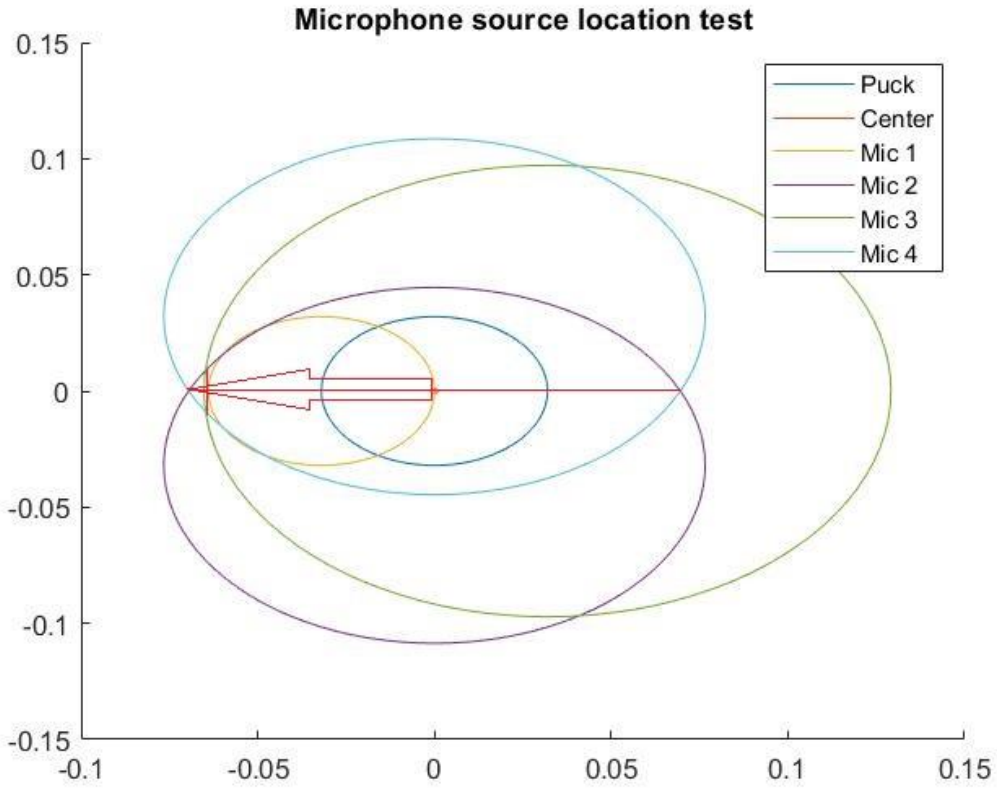

742
309
3
4
2
1
10799053
P:953
000100



Her kan vi se input signalene fra de 4 mikrofonene. Ved å måle tidsforskjellen til kurvene ved y=o kan vi se tidsforskyvningen mellom kurvene. Disse forholdstallene kan gruppen så bruke sammen med de fate lokasjonene til mikrofonene.



"Microphone source direction" brukte skalarer fra tidsforsinkelsen til å regne ut ringene



"Microphone source location test" her ble det kalkulert avstanden fra en gitt "master" mikrofon som var nærmest lydkilden og til de ulike mikrofonene. Dette ble gjort ved å angi tiden signalet treffer master til å være null og kalkulere tiden til de andre mikrofonene. Denne kalkuleringen bestemte så størrelsen på ringene.

Resultat

Under testen målte gruppen faseforskyvningen på de ulike mikrofonene for å indikere en vinkel lyden kommer fra, dette ved hjelp av å måle forskyvningen på hensyn av tid. Under målingene gjorde gruppen ulike innspillinger av lyd fra ulike retninger og distanse. Se bilder under.

Resultat	Beskrivelse
1 korrekt indikasjon på vinkel, avik på distanse	Måling med lyd fra 0 grader (Vinkelrett på mic 1 og 3) på en distanse på 13,5 cm, var det 4 cm avvik.
2 korrekt indikasjon på vinkel, avik på distanse	Måling med lyd fra 90 grader (Vinkelrett på mic 2 og 4) på en distanse på 13,5 cm, var det 2 cm avvik.
3 korrekt indikasjon på vinkel, avik på distanse	Måling med lyd fra 0 grader (Vinkelrett på mic 1 og 3) på en distanse på 5 cm, var det 1 cm avvik.
4 korrekt indikasjon på vinkel, avik på distanse	Måling med lyd fra 90 grader (Vinkelrett på mic 1 og 3) på en distanse på 5 cm, var det 1 cm avvik.
5 korrekt indikasjon på vinkel, avik på distanse	Måling med lyd fra 0 grader (Vinkelrett på mic 1 og 3) på en distanse på 13,5 cm, var det 2 cm avvik.

	Ja	Nei	Beskrivelse
Blei testen gjennomført som forventet?	x		Gruppen forventet mulig avvik på distanse pga. den simple formen for kalkulasjon.
Problem som oppstod underveis?	x		Gruppen støttet på noen små problemer ble rettet på underveis.
Feilkilder?	x		Det oppsto enkelte feilkilder pga. dårlig kode, dette var da forventet pga. tiden som ble lagt inn i koden før testing.
Forbedringer til neste test?	x		Kalkuleringen må forbedres, da dette baserte seg på simple visuelle metoder.

G

Rapport møte 06.12.19

Omhandler: Bachelor oppgave hos Seaonics

Deltagere: Finn-C, Torbjørn, Andreas, Ottar og Seaonics

Dato og klokkeslett: 11:30-12:30

Handling på møtet:

I. Hva Seaonics gjør
II. Hva Seaonics ser for seg i mulig oppgave
III. Innspill fra gruppen
IV. Omvisning av lokaler og test-lab
V. Konklusjon

Hva kom frem:

Temaer: AI- kunstig intelligens, Tilstandskontroll, mikrofon, lydsignaler, Maskinrom, frem produsering av feil, prognosebasert system, lyd kart og posisjonering

Info:

Under møtet kom det frem av Seaonics ønsker å finne ut om det er mulig å lage et system for tilstandskontroll av maskiner i et maskinrom, dette ved hjelp av lydspektret. Opptak av rommet skal utføres og dette skal kunne brukes i et program for å gjenkjenne type maskineri og hvordan tilstand det befinner seg i.

Det ble også pratet litt om posisjonering av hvor lyd kommer fra.

Konklusjon:

Seaonics sa at det var åpen for oss å utforske og gjøre hva vi hadde løst til i oppgaven. Det var åpen for oss å bruke besøkslokalene deres for arbeidsplass og testlabb nede i kjelleren til opptaksrom.

Gruppen ser for seg et program som bruker AI til å gjenkjenne og kontrollere tilstand til en maskin. Ved å frem produsere feil og mange opptak, skal gruppen klare å trene et neuralt nettverk til å gjenkjenne dette.

Rapport møte 14.01.20

Omhandler: Møte med styringsgruppen

Deltagere: Finn-C, Torbjørn, Andreas, Ottar og Seaonics

Dato og klokkeslett: 14.01 kl 10:30 – 12:30

Handling på møtet:

I. Går gjennom oppgaven gruppen har formulert
II. Styringsgruppen kommer med innspill
III. Endring i oppgaven
IV. Spørsmål gruppen har til styringsgruppen.
V.

Hva kom frem:

Temaer:

Anomal deteksjon, triangulering, mikrofoner, Raspberry Pi, støymåling, Endring av prosjektoppgave til å sette triangulering høyere, Arbeidsplass hos Seaonics, flere lydkilde lokalsering, lytting og seperasjon (seperasjonsfiltrering), Resurs giver, div valg i henhold til rapport.

Info:

Anomal deteksjon – Styringsgruppen stiller seg positiv til valget.

Triangulering – Ottar er positiv og ønsker å få dette til. Ønsket er å sette denne i første prioritet

Mikrofon – Styringsgruppen ønsker seg billig mikrofon. Burde drøfte posisjon av mikrofoner i oppgaven.

Støy - RSS (Root mean square)

Maskinrom – Styringsgruppen ønsker at vi definerer at jobben pågår i et maskinrom og ikke en vinsj. Siden vinsjen er vanskeligere.

Framproduksjon av støy - tannhjul, strips, skrujern og div andre ting.

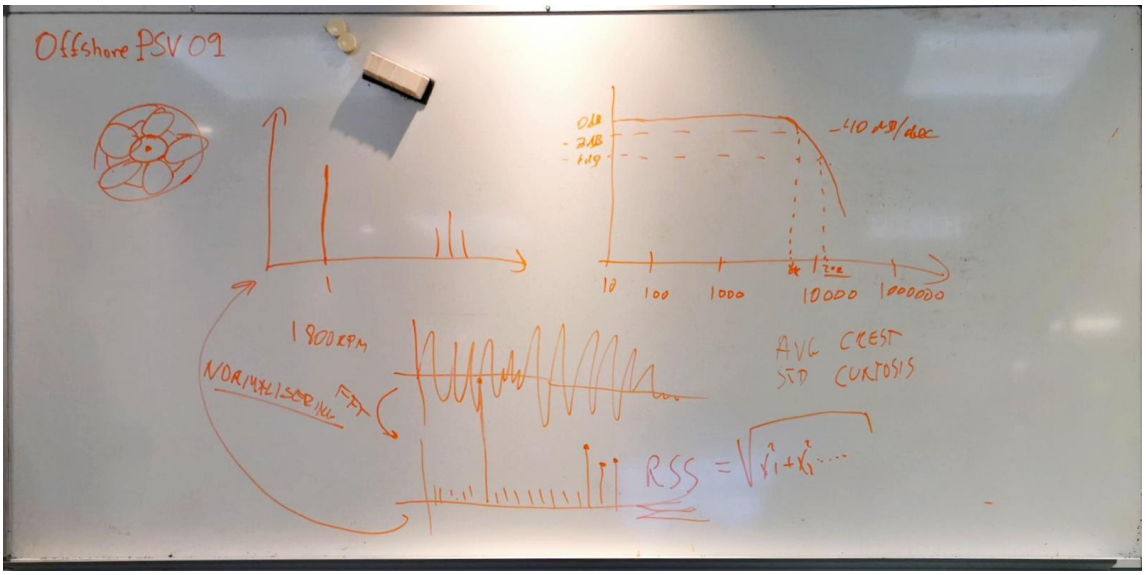
Ønsket er å sette høyere fokus på triangulering og bruke dette til å filtrere ut

uønskede lydkilder(seperasjonsfiltrering). Dette skal brukes til å hente ut data på spesifikke utstyr i et område.

Arbeidsplass hos Seaonics – Seaonics stiller med arbeidsplass som kan brukes av gruppen, diverse skjermer, stoler og annet utstyr er klart til bruk. Det er tilbud om kantine hos Seaonics, timeplan må gis når gruppen er til stede for å dimensjonere mat riktig.

Resursgiver – NTNU stiller som resurgiver til prosjektet, dette for å bestille div mikrofoner og div annet utstyr.

div valg i henhold til rapport – Engelsk, Overleaf og Grammarly.



Rapport møte 31.01.2020

Omhandler: Styringsmøte

Deltagere: Finn-C, Torbjørn, Andreas, Ottar og Seaonics

Klokkeslett: 10:00 – 11:00

Handling på møtet:

I. Introduksjon og fremvisning av fremgang
II. Samtale om mål fremmover
III. Innslag om hardware fra styringsgruppen

Hva kom frem:

Temaer: fokus på bachelor, status, hardware og teori rundt signalopptak, forsterkere, kanaler, frekvens.

Info: Det ble først en fremvisning av forprosjektet og gruppens meninger. senere ble det vist frem LEAN modulen til prosjektet. Det ble en kort samtale rundt temaet med oppgavegiver og rådgiver. Senere kom rådgiver frem med en rekke anbefalinger til gruppen om valg av hardware, og hvordan behandling av data gjøres på best måte. Forslag til hvordan vi skal tenke når vi skal velge hardware kom også frem.

Se vedleggsfil

Rapport møte 10.02

Omhandler: Statusmøte

Deltagere: Finn-C, Torbjørn, Andreas, Tom og Ottar

Dato og klokkeslett: Kl 10:00 – 11:00

Handling på møtet:

I. Vis av fremgang til gruppen
II. Mer prat om Hardware
III. Gruppens planlagte aktiviteter fremover

Hva kom frem:

Temaer: Fremgang til gruppen, Gantt, Hardware

Info: Det ble vist frem gruppens progresjon i prosjektet, hvorav vi henger litt etter på bestilling av hardware. Gruppen har kommet seg inn i prosjektet og driver enda med innhenting av informasjon.

Ottar og Gjøran har begge noen ideer for utforming og funksjonalitet til hardware, hvorav gruppen stiller seg enig i. Denne utformingen er å montere 8 mikrofoner i en sirkel på en rektangulær takplate. Det er ønskelig å teste denne utformingen med vinkling av mikrofoner.

Det ble også nevnt gruppens kommende aktiviteter på gantt skjemaet. Det ble foreslått at vi fokuserer på å holde planen slik at gruppen ikke henger etter planen.

Rapport møte 28.02.2020

Omhandler: Møte med styringsgruppen

Deltagere: Finn-C, Andreas, Tom og Ottar

klokkeslett: 12:00 – 13:00

Handling på møtet:

I. Info om hva gruppen har gjort siden sist
II. Gjennomgang av fremtidig plan
III. Planlegging av sluttresultatet

Hva kom frem:

Temaer: Focus(testing og triangulering fra scrach), jobbe videre, ikke fokusere på alle de bibliotekene der ute.

Info:

Under møtet kom det frem av gruppen har jobbet godt. Styringsgruppen ønsker at gruppen fokuserer mer på måling, testing og simple kalkuleringer, i stedet for det gruppen har fokusert på tidligere (bruk av bilbioteker og "backengineer" andres programmer). Dette skyldes stagnering og problemer gruppen støttet på.

Det ble nevnt at gruppen burde lage seg et testlokale med faste punkt for måling og lydkilder. Det ble også nevnt at vi må være forsiktig når det kommer til rene sinusbølger, PGA disse er skadelige for ørene i lengden.

Gruppen sliter med punktene Beamforming, echo-kanselering og DOA. Dette er punkter som er større en antatt. Mer tid disse punktene medfølger endringer i planen til gruppen, som må legges om for å fokusere på disse punktene.

Vedlegg:

Rapport møte 17.03.2020

Omhandler: Covid-19, fremgang

Deltagere: Finn-C, Torbjørn, Andreas, Tom og Ottar Skype møte

klokkeslett: 12:00

Handling på møtet:

I. Gjennomgang av Covid-19 rapport
II. Fremgangsrapport
III. Videre oppgaver

Hva kom frem:

Temaer: Beamforming, ekkokanselering, Triangulering, Covid-19 oppdatering.

Info:

Ta opp lydfil, bruk denne i beamforming for så å filtrere ekko, slik at vi kan se om det blir endring i beamformingen etterpå.

Søking om ekstra tid? Nei, mulig utvidelse av sluttdato.

Har mer hyppige møter fremmover pga corona. Så fra nå av har vi ukentlig møter

Rapport møte 24.03.2020

Omhandler: Ukesrapport med Styringsgruppen

Deltagere: Finn-C, Torbjørn, Andreas, Tom og Ottar

klokkeslett: 11:00

Handling på møtet:

I. Gjennomgang av forje uke
II. Tilbakemeldinger
III. Forslag til fremtidige oppgaver

Hva kom frem:

Temaer: Rapport fra forje uke, tilbakemeldinger på arbeidet vårt.

Info:

Raspbarry pi hatt fungerer ikke(sampleraten er ikke så høy som opplyst). Beamforming går ikke så bra som ønsket, derfor velger gruppen å lage en simpel delay-sum-algoritme. Dette gjør at gruppen lager all beamforming koden selv fra scrach med enkle teknikker i matlab.

Vedlegg:

Rapport møte 31.03.2020

Omhandler: Styringsmøte, med Kai som ekstra rådgiver

Deltagere: Finn-C, Torbjørn, Andreas, Kai og Ottar

klokkeslett: 12:00-13:00

Handling på møtet:

I. Gruppens ståsted
II. Innspill til endringer
III. Fremtidige mål

Hva kom frem:

Temaer: tidsforskyvning av kanaler. Nøyaktighet. Filtrering. Cross corelation.

Info:

Det ble under møtet avlagt en rapport om forje ukes fremgang og gruppens ståsted i oppgaven. Smått endringer ble fremmet av rådgiverne. Eks bruk av tangens i stedet for cosinus og sinus får mer sikker kalkulering.

Det ble lagt frem ønsker fra rådgiverne på hva som bør jobbes videre med, eksempel på dette er å lage en figur på nøyaktigheten til det teoretiske kalkuleringen av forskyvning.

Videre skal gruppen jobbe med å fremme det fokuserte signalet i beamforming med å ta i bruk filtringsmetoder og signalbehandling.

Rådgiverne mente også forslaget om å begynne avsluttingen av ekkokanseleringen er en god ide. Dette pga resultatene ikke går i retningen gruppen ønsker.

Rapport møte 03.04.2020

Omhandler: Ukesrapport med Styringsgruppen

Deltagere: Finn-C, Torbjørn, Andreas, Tom, Ottar og Kai

klokkeslett: 11:00-12:00

Handling på møtet:

I. Gjennomgang av gruppens fremgang
II. Diverse innspill fra styringsgruppen
III. Fremtidige gjøremål

Hva kom frem:

I møtet kom det frem mindre innspill fra styringruppen. Noen små innspill som eks bruk av tangens istede for cosinus. Det ble fremmet at gruppen tar påskeferie på røddagene.

Rapport møte 14.04.2020

Omhandler: Ukes møte med styringsgruppen

Deltagere: Finn-C, Torbjørn, Andreas, Tom og Ottar

klokkeslett: 12:00- 13:00

Handling på møtet:

I. Gjennomgang av resultater
II. Oppfordring til gruppen
III. Fremtidige gjøremål

Hva kom frem:

I møtet ble det frøftet resultatet til gruppen. Det ble utstedt et vedlegg med lydfiler som gruppen har produsert gjennom kildeseperasjon i matlab. Styringsgruppen syntes dette så lovende ut og syntes mer testing/resultater skulle produseres.

Det kom frem at Ottar anbefalte gruppen å ha minst 3 uker med skriving på bacheloren. Så gruppen må prøve å avslutte denne fasen av prosjektet i løpet av neste uke.

Vedlegg:

Rapport møte 05.05.2020

Omhandler: Møte med styringsgruppen

Deltagere: Finn-C, Torbjørn, Andreas, Tom og Ottar

klokkeslett: 13:00

Handling på møtet:

I. Gjennomgang av statusen til prosjektet
II. Video til presentasjon
III. Godkjenning av rapport, egenerklæring, publiseringsavtale.
IV. Presentasjon

Hva kom frem:

I. Vi gjekk igjennom kva vi har gjort til nå. Og forklarte kva vi har fjernet og lagt til. Vi har endret om til at vi kun jobber med bachelor skrivingen og video. Vi vil prøve å bli ferdig med eit utdrag av rapporten til slutten av uken for å sende til Ottar. Vi skal prøve å lage til deler av videoen løpet av uken.

II. Det kan være lurt i videoen å vise sammen ligninger mot kjente situasjoner. Situasjoner som konferanser eller cocktail party's der at det kan være vanskelig å høre, få fram at vårt prosjekt skal separere lydene slik at det blir lettere å høre

III. Malene vi har brukt er utdaterte og det finns flere versjoner, Ottar skal sjekke opp krav til frontside og finne ut om det er mulig å få egenerklæringen og publiseringssamtrykket på oppdatert på engelsk. Ottar skal også sjekke opp angående publisering. Det blir ikkje krav om å lage ein plakat.

IV. Vi skal lage ein video som vi skal sende til Ottar. Ottar skal sette opp ein egen fremviser på Zoom for å vise frem videoer slik at vi slipper problemer med bilde og lyd

Rapport møte 12.05.2020

Omhandler: Sluttmøte med styringsgruppen

Deltagere: Finn-C, Torbjørn, Andreas, Tom og Ottar

klokkeslett:

Handling på møtet:

I. Tekniske spørsmål om oppsett av bachelor
II. Litt info om fremføring
III. Generell kretikk om sammarbeid med styrinsgruppe under bachelor

Hva kom frem:

Temaer: prefase, bachelor oppsett, gruppe-erklæring, publiseringsavtale, generelt arbeid samens med styringsgruppen, vinklinger fra styringsgruppe.

Info:

Under møtet kom det opp spørsmål om oppsett og fremføring av bachelor. Publiseringsavtale ble skrapet siden dette medfølger med innlevering. meninger angående samarbeid med Seonics og rådgiver ble tatt opp, hvor positiver tilbakemeldinger var gitt til gruppen angående arbeidsinnsats.

Vedlegg:

Rapport møte 25.03.20

Omhandler: Rådgivning signalbehandling

Deltagere: Finn-C, Torbjørn, Andreas og Kai

klokkeslett: 16:00-17:00

Handling på møtet:

I. Gruppens ståsted
II. Spørsmål til kai
III. Rådgivning fra kai

Hva kom frem:

Temaer: Samplingsfrekvens, beamforming, triangulering, filtrering og ekkofiltrering.

Info:

Møtet ble avholdt over skype pga korona-krisen.

Kai mente at på grunn den korte avstanden på mikrofonarrayet må samplingsfrekvensen være høyere en tidligere antatt for å få noe nytteverdi av datane. Jo høyere samplingsfrekvens jo mer informasjon kan vi finne i signalet.

Spørsmål ble stilt til Kai som gruppen hadde. Dette skal Kai finne ut og komme tilbake til gruppen.

H

Ukentligrapport uke 3

Dato: 14.01.20

Deltakere: T, A og F

	Ja	Nei	Beskrivelse
Blei alle mål for forrige uka oppnådd?	x		Alt annet en møter med Ottar og Ibrahim, som ikke ble satt opp før tirsdag 14. januar 2020
Problem som oppstod underveis?		x	
Nokken oppgaver som må jobbes videre på eller trenger lengre tid		x	
Nokken problem som kan oppstå under denne ukens oppgaver?	x		Gantt skjemaet kan bli vanskelig å fullføre. Prosjektet er vanskelig å dele opp. Andre problemer er tid, som er lite av denne uka siden det er Lean med industri 4.0.

Andre beskjeder	
	2 planlagte møter denne uken, et med Seaonics og et med Ibrahim. Finn tar fri helga 12-16 Mars. Torbjørn jobber bortefra 1-9 feb

Ukentlig rapport uke 4

Dato: 20.01.2020

Deltakere: Finn, Torbjørn og Andreas

	Ja	Nei	Beskrivelse
Blei alle mål for forrige uka oppnådd?		x	Fikk ikke møtt Ibrahim pga endring fra hans side, dette passet likeså greit med de nye endringene i oppgaven.
Problem som oppstod underveis?	x		Det ble endring i oppgaven etter møtet med Seaonics, Fokuset ligger nå mer på filtrering av lydkilder. Andreas måtte også dra på jobb, som ikke var planlagt.
Nokken oppgaver som må jobbes videre på eller trenger lengre tid	x		Lage gantt skjema.
Nokken problem som kan oppstå under denne ukens oppgaver?	x		Tidspress. Vi må prøve å bli ferdige til onsdag, siden vi skal bort fra torsdag av. Samt kommer NITO årsmøte + stand iveien for noe av oppgavene på mandag.

Andre beskjeder	
	Planlagt fravær på gruppen onsdag og torsdag. Andreas blir samt borte mandag/tirsdag.

Ukentlig rapport uke 5

Dato: 30.01.2020

Deltakere: Finn, Andreas og Torbjørn

	Ja	Nei	Beskrivelse
Blei alle mål for forrige uka oppnådd?	x		De som kunne gjøres, en som ble utelatt, med det gjelder sammtale med Seaonics, som skal foregå 31.01
Problem som oppstod underveis?		x	Fikk mye ros for LEAN modulen, som kommer til å bli et vedlegg til bacheloren.
Nokken oppgaver som må jobbes videre på eller trenger lengre tid		x	
Nokken problem som kan oppstå under denne ukens oppgaver?	x		På grunn av Industri 4.0 faget, har gruppen problemer med å ha tid til prosjektet. Sammens med omskrivingen av oppgaven er gruppen litt bakpå når det gjelder start av hovedoppgaven.

Andre beskjeder	
	Blir noen jobbintervju på enkelte av gruppemedlemmene, dette skaper litt svinn av arbeidstid.

Ukentlig rapport uke 6

Dato: 03.02.20

Deltakere: Finn og Andreas

	Ja	Nei	Beskrivelse
Blei alle mål for forrige uka oppnådd?		x	Fikk ikke laget ferdig mal for testing. Tatt over tid når vi vet litt mer hvordan dette utføres. Det ble heller ikke bestilt deler, da dette skulle gjøres i forje uke. Dette blir gjort iløpet av uken når hardware er litt mer sikkert(rett valgt).
Problem som oppstod underveis?		x	Litt lite tid pga 3D-printing modul i faget industri 4.0
Nokken oppgaver som må jobbes videre på eller trenger lengre tid	x		Bestille deler
Nokken problem som kan oppstå under denne ukens oppgaver?	x		Må ta tid til å bestille deler.

Andre beskjeder	
	Torbjørn er borte hele uken, men jobber bortefra. Det samme gjelder Finn, som blir borte fra torsdag til fredag.

Ukentlig rapport uke 7

Dato: 10.02.2020

Deltakere: Finn, Andreas og Torbjørn

	Ja	Nei	Beskrivelse
Blei alle mål for forrige uka oppnådd?		x	Fikk ikke bestilt hardware. Videre kommunikasjon med forhandler må gjøres(Forsinkelser i forhold til Covid-19). Har også ikke fått utført valg av GUI bibliotek og utsende.
Problem som oppstod underveis?	x		Vanskeligere å finne hardware en først antatt.
Nokken oppgaver som må jobbes videre på eller trenger lengre tid	x		Hardware og GUI må gjøres ferdige.
Nokken problem som kan oppstå under denne ukens oppgaver?	x		Denne uka faller delvis bort pga Industri 4.0

Andre beskjeder	
	Må sikkert jobbe litt ekstra denne uken for å holde oss etter planen.

Ukentlig rapport uke 8

Dato: 17.02.2020

Deltakere: Finn, Andreas og Torbjørn

	Ja	Nei	Beskrivelse
Blei alle mål for forrige uka oppnådd?	x		Hardware ble bestilt.
Problem som oppstod underveis?	x		Hardware ble forskjøvet fra uken før
Nokken oppgaver som må jobbes videre på eller trenger lengre tid	x		Jobbe med GUI, dette er noe vi har forskjøvet siden det har vært litt uklart på hvordan bibliotek vi skulle velge.
Nokken problem som kan oppstå under denne ukens oppgaver?	x		Problemer med leveranse av hardware

Andre beskjeder	
GUI	Det må sjekkes på GUI og hvordan bibliotek vi skal bruke.

Ukentlig rapport uke xx

Dato:

Deltakere:

	Ja	Nei	Beskrivelse
Blei alle mål for forrige uka oppnådd?		x	Testing og triangulering av lyd ligger vi bakpå med, dette krever mer tid en antatt. GUI ligger også etter.
Problem som oppstod underveis?	x		Triangulering er mere innviklet ved flere lydkilder. Det er også problemer med at gruppen på bruke Linux for å kunne ta i bruk noen av bibliotekene.
Nokken oppgaver som må jobbes videre på eller trenger lengre tid	x		Triangulering og GUI trenger å bli brukt mer tid på. Andreas skal legge inn ekstra tid på Triangulering og Finn skal se mer på GUI så fort han er ferdig med kodebiten for opptak.
Nokken problem som kan oppstå under denne ukens oppgaver?		x	Vi har fått hardware og de bitene vi trengte for å kunne komme fult i gang.

Andre beskjeder	
	Torbjørn borte fredag, Ny innlevering i industri 4.0.

Ukentlig rapport uke 11

Dato: 09.03.2020

Deltakere: Finn-Christian, Andreas og Torbjørn

	Ja	Nei	Beskrivelse
Blei alle mål for forrige uka oppnådd?	x		Raspberry pi hatten fikk gruppen til å fungere, samt en simpel grafisk trigonometri.
Problem som oppstod underveis?		x	Uken var planlagt for Industri 4.0
Nokken oppgaver som må jobbes videre på eller trenger lengre tid	x		Som tidligere så må tiden på triangulering og filtrering utvides.
Nokken problem som kan oppstå under denne ukens oppgaver?		x	Utenom ferdigdefinerte risikoer så er det ikke det.

Andre beskjeder	
	Finn reiser bort torsdag og fredag denne uka

Ukentlig rapport uke 12

Dato: 16.03.2020

Deltakere: Finn, Andreas og Torbjørn. Dette over Discord(Voice chat)

	Ja	Nei	Beskrivelse
Blei alle mål for forrige uka oppnådd?		x	Henger bak på triangulering, beamforming og ekkokanselering. Karnsje fjerne/falle bort på ekkokanselering og GUI.
Problem som oppstod underveis?	x		Corona, som gjorde at innføring av hjemmekontor blir iverksatt. Ellers vanskelige oppgaver.
Nokken oppgaver som må jobbes videre på eller trenger lengre tid	x		på triangulering, beamforming og ekkokanselering.
Nokken problem som kan oppstå under denne ukens oppgaver?	x		Vi har et gruppemedlem som sitter i karantene, samt resten jobber med hjemmekontor. Dette har en påvirkning på arbeidet gruppen får gjort.

Andre beskjeder	
	Grunnet karantenen til et av gruppemedlemmene er muligheten for samarbeid redusert på ubegrenset tid. Det samme gjelder for retningslinjene gitt av staten for hjemmekontor dersom mulig. Nytt møte med styringsgruppen virtuelt, avtalt på slutten av uken.

Ukentlig rapport uke 13

Dato: 23.03.20

Deltakere: Finn, Andreas og Torbjørn

	Ja	Nei	Beskrivelse
Blei alle mål for forrige uka oppnådd?		x	Ligger fortsatt bakpå når det gjelder triangulering og ekkokanselering
Problem som oppstod underveis?		x	Filtrering av ekko har fått en ny vei.
Nokken oppgaver som må jobbes videre på eller trenger lengre tid	x		Triangulering, Ekkokanselering og beamforming (alle må kombineres)
Nokken problem som kan oppstå under denne ukens oppgaver?		x	

Andre beskjeder	
	Redefinering av arbeidsoppgaver i bachelor. Sitte lengre fremover.

Ukentlig rapport uke 15

Dato: 07.04.2020

Deltakere: Finn, Andreas og Torbjørn

	Ja	Nei	Beskrivelse
Blei alle mål for forrige uka oppnådd?		x	Gruppen har fortsatt problemer med de store punktene: Triangulering, beamforming og Filtrering.
Problem som oppstod underveis?	x		Torbjørn fikk problemer med PCen og måtte sende den inn til reparasjon.
Nokken oppgaver som må jobbes videre på eller trenger lengre tid	x		Triangulering og Beamforming er ting som er viktig for prosjektet og bør jobbes videre med.
Nokken problem som kan oppstå under denne ukens oppgaver?	x		Påskeferie/rødager gjør at tiden denne uken er mindre en ellers.

Andre beskjeder	

Ukentlig rapport uke 15

Dato: 07.04.2020

Deltakere: Finn, Andreas og Torbjørn

	Ja	Nei	Beskrivelse
Blei alle mål for forrige uka oppnådd?		x	Gruppen har fortsatt problemer med de store punktene: Triangulering, beamforming og Filtrering.
Problem som oppstod underveis?	x		Torbjørn fikk problemer med PCen og måtte sende den inn til reparasjon.
Nokken oppgaver som må jobbes videre på eller trenger lengre tid	x		Triangulering og Beamforming er ting som er viktig for prosjektet og bør jobbes videre med.
Nokken problem som kan oppstå under denne ukens oppgaver?	x		Påskeferie/rødager gjør at tiden denne uken er mindre en ellers.

Andre beskjeder	

Ukentlig rapport uke 16

Dato: 14.04.20

Deltakere: Finn-Christian, Torbjørn og Andreas

	Ja	Nei	Beskrivelse
Blei alle mål for forrige uka oppnådd?		x	Seperasjon av lydkilder ble oppnådd. Ekkofiltrering ble oppnådd. DOA med en lydkilde ble oppådd. Dette er grovt og ikke kombinerte programmer. Mangler fremdeles DOA for flere lydkilder.
Problem som oppstod underveis?		x	Ikke noe spesielle problemer, denne uken bydde på løsninger og mye fremgang. Men arbeidsdagene denne uken var få, siden det var røddager torsdag-mandag.
Nokken oppgaver som må jobbes videre på eller trenger lengre tid	x		DOA med flere kilder trenger lenger tid. Det må også finpusses og sammensettes koder hvis gruppen skal lage et "produkt"
Nokken problem som kan oppstå under denne ukens oppgaver?		x	Ingen forutsette problemer. Men deadline nærmer seg på fredag og da skal gruppen være ferdig med forskningsfasen.

Andre beskjeder	

Ukentlig rapport uke 17

Dato: 20.04.2020

Deltakere: Finn, Andreas og Torbjørn

	Ja	Nei	Beskrivelse
Blei alle mål for forrige uka oppnådd?	x		
Problem som oppstod underveis?	x		Overlapping av chunks, men fikk dette til på slutten av uken.
Nokken oppgaver som må jobbes videre på eller trenger lengre tid		x	
Nokken problem som kan oppstå under denne ukens oppgaver?		x	

Andre beskjeder	
	Gruppen går inn i en ny fase, dette innebærer at sluttspurt på enkeltoppgaver gjennomføres denne uken.

Ukentlig rapport uke 18

Dato: 27.04.2020

Deltakere: Finn, Andreas og Torbjørn

	Ja	Nei	Beskrivelse
Blei alle mål for forrige uka oppnådd?		x	Fikk ikke gjennomført SNR illustrasjoner.
Problem som oppstod underveis?	x		Testopptak ble ikke gjennomført som ønsket. Dette grunnet mye unøyaktigheter og problemer med sinusbølger som bryter hverandre ned. Det blir også ødelagt/brent høyttalere under testing, dette grunnet sinusbølgens destruktive evne ved avspilling.
Nokken oppgaver som må jobbes videre på eller trenger lengre tid	x		Få til SNR illustrasjonene for å visualisere programmets funksjon og en god test av programmet.
Nokken problem som kan oppstå under denne ukens oppgaver?	x		Videre problemer med hardware, men tanke på at vi ødelegger små høyttalere under testing.
Nåværende ståsted ferdig?	x		Ja gruppen skal gå over til skriving av rapport denne uken. Fokusområdet vil bli skriving, men testing og illustrasjoner vil bli jobbet med parallelt.
Nytt fokusområde	x		- Kartlegge hva som burde være med i hvert tema. - Lage illustrasjoner fra testing/resultater.
Er oppgavebesvarelsen i henhold til oppgaveteksten?			Her må gruppen identifisere hvordan vi ligger ann til oppgaveteksten. En liten gjennomgang mellom gruppen og sammens med rådgiver må til for å besvare spørsmålet.

Andre beskjeder	
	Finn er borte på fredag

Ukentlig rapport uke 19

Dato: 04.05.2020

Deltakere: Finn, Torbjørn og Andreas

	Ja	Nei	Beskrivelse
Blei alle mål for forrige uka oppnådd?	x		Avslutte alt av programmering er gjort.
Problem som oppstod underveis?	x		Opptak og testing fungerer ikke som det skal. RIR funker ikke som det skal.
Nokken oppgaver som må jobbes videre på eller trenger lengre tid	x		Noen få testinger må gjøres med de nye lydopptakene, som ikke er helt som det skal.
Nokken problem som kan oppstå under denne ukens oppgaver?	x		Skrivesperre, mangler på motivasjon.

Andre beskjeder	
Nye oppgaver	Finn: rapport covid19 fiks, beamforming, lage test resultater. Andreas: feilkilder, opptak, DOA. Torbjørn: Ekko test og konklusjon.

Ukentlig rapport uke 20

Dato: 11.05.20

Deltakere: Finn, Torbjør og Andreas

	Ja	Nei	Beskrivelse
Blei alle mål for forrige uka oppnådd?	x		Vi fikk inn all nødvendig test data og illustrasjoner
Problem som oppstod underveis?	x		Vi fikk hardwareproblemer som nevt forje uke. Dette må skrives inn i rapporten.
Nokken oppgaver som må jobbes videre på eller trenger lengre tid		x	Til dags dato er det viktig at vi fokuserer på skriving av bachelor rapporten. Dette skjer da litt sent, men gruppen jobber hardt med å komme i mål.
Nokken problem som kan oppstå under denne ukens oppgaver?	x		Skrivesperre og for liten tid kan være store problemer som gruppen kan støte på. Det er viktig for gruppen å fokusere på viktige emner i rapporten, grunnet dårlig tid.

Andre beskjeder	